A Prescriptive Analytics Approach for Energy Efficiency in Datacentres

John Panneerselvam

Doctor of Philosophy 2017

Table of Contents

List of Figures

List of Tables

Declaration

The study outlined in this dissertation were carried out in the Department of Engineering and Technology at University of Derby, under the supervision of Professor Nick Antonopoulos and Professor Lu Liu. This is to declare that the work stated in this thesis was done by the author, and no part of the thesis has been submitted in a thesis form to any other university or similar institution. No human or animal participation have been included in this research and the research presented in this thesis has been ethicaly approved. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others. Parts of this thesis have previously appreared in the papers listed in the list of publications.

John Panneerselvam
University of Derby
2017

List of Publications

Journals

1. **J. Panneerselvam**, L. Liu, Y. Lu and N. Antonopoulos, An Investigation into the Impacts of Task-level Behavioural Heterogeneity upon Energy Efficiency in Cloud Datacentres, *Future Generation Computer Systems,* 2018, inpress.
2. **J. Panneerselvam,** L. Liu and N. Antonopoulos, An Approach to Optimise Resource Provision with Energy-awareness in Datacentres by Combating Task Heterogeneity, *IEEE Transactions on Emerging Topics on Computing*, under review.
3. **J. Panneerselvam,** L. Liu and N. Antonopoulos, InOt-RePCoN: Forecasting User Behavioural trend in Large-Scale Cloud Environments, *Future Generation Computer Systems*, July 2017. Impact Factor: 3.997.
4. **J. Panneerselvam**, L. Liu, N. Antonopoulos and J. Hardy, Analysis, Modelling and Characterisation of Zombie Servers in Large-Scale Cloud Datacentres, *IEEE Access Journal*, 2017, in press. Impact Factor: 3.244.
5. **J. Panneerselvam**, J. Hardy, L. Liu, N. Antonopoulos, B. Yuan, Mobilouds: An Energy Efficient MCC Collaborative Framework with Extended Mobile Participation for Next Generation Networks, *IEEE Access Journal*, 2016. Impact Factor: 3.244.
6. Y. Lu, **J. Panneerselvam**, L. Liu, Y. Wu, RVLBPNN: A Workload Forecasting Model for Smart Cloud Computing, *Journal of Scientific Programming,* Hindawi, 2016. Impact Factor: 0.244.
7. **J. Panneerselvam**, A. Atojoko, K. Smith, L. Liu, N. Antonopoulos, A Critical Review of the Routing Protocols in Opportunistic Networks, *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems,* vol 1 (1), 2014.
8. X. Bai, L. Liu, M. Cao, **J. Panneerselvam**, Q. Sun, H. Wang, Collaborative Actuation of Wireless Sensor and Actuator Networks for the Agriculture Industry, IEEE Access Journal, 2017. Impact Factor: 3.244.
9. X. Shen, Q. Chang, L. Liu, **J. Panneerselvam**, Z. Zha, CCLBR: Congestion Control-Based Load Balanced Routing in Unstructured P2P Systems, *IEEE Systems Journal*, vol. PP(99), pp. 1 -12, June 2016. Impact Factor: 3.882.
10. J. Li, J. Zhao, Y. Li, L. Cui, B. Li, L. Liu, **J. Panneerselvam**, iMIG: Toward an Adaptive Live Migration Method for KVM Virtual Machines, *The Computer Journal*, July 2014. Impact Factor: 0.711.
11. X. Chen, L. Liu, Z. Zha, P. Gu, Z. Jiang, J. Chen, **J. Panneerselvam**, Achieving dynamic load balancing through mobile agents in small world P2P networks, *Journal of Computer Networks*, pp 134-148, vol 75 (A), 2014. Impact Factor: 2.516.

Conference Papers

1. **J. Panneerselvam,** L. Liu and N. Antonopoulos, Characterisation of Hidden Periodicity in Large Scale Datacentre Environments, *IEEE Int. Conf. Green Computing and Communications,* Exeter, June 2017. IEEE Press.
2. **J. Panneerselvam**, L. Liu, N. Antonopoulos, M. Trovati, Latency-Aware Empirical Analysis of the Workloads for Reducing Excess Energy Consumptions at Cloud Datacentres, *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 44 - 52, Oxford, March 2016. IEEE Press.

3. **J. Panneerselvam**, L. Liu, N. Antonopoulos, Y. Bo, Workload Analysis for the Scope of User Demand Prediction Model Evaluation in Cloud Environments, *Proc of IEEE/ACM Int. Conf. Utility and Cloud Computing*, pp 883-889, December 2014. IEEE Press.
4. Y. Fan, **J. Panneerselvam** and L. Liu, The Cost Function and Improvement Strategies of Service Quality of University Library under New Information Environments*, 2nd International Symposium on Real-time Data Processing for Cloud Computing,* Exeter, June 2017, in press. IEEE Press.
5. F. Tao, **J. Panneerselvam**, T. Holding, L. Liu, A Cloud-based Sustainable Business Model for Effective ICT Provision in Higher Education, *7TH IEEE Int. Sym. Service Oriented System Engineering,* San Francisco, pp. 222-228, March 2015. IEEE Press.
6. X. Bai, M. Cao, L. Liu, **J. Panneerselvam**, Collaborative Estimation and Actuation of Wireless Sensor and Actuator Networks for the Greenhouse Environment, *Proc. of IEEE/ACM Int. Conf. Utility and Cloud Computing*, Shanghai 2016, in press. IEEE Press.
7. K. Vilius, L. Liu, **J. Panneerselvam**, T. Stimpson, A Critical Analysis of the Efficiencies of Emerging Wireless Security Standards Against Network Attacks, *Int. Conf. Intelligent Networking and Collaborative Systems (INCOS)*, pp. 472 – 477, Taipei 2015. IEEE Press.
8. X. Sun, Y. Wu, L. Liu, **J. Panneerselvam**, Efficient Event Detection in Social Media Data Streams, *IEEE Int. Conf. Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, pp. 1711-1717, Liverpool 2015. IEEE Press.
9. H. Al-Jabry, L. Liu, Y. Zhu, **J. Panneerselvam**, A Critical Evaluation of the Performance of Virtualization Technologies, *9th Int. Conf. Communications and Networking in China (CHINACOM)*, pp. 606-611, Maoming, 2014. IEEE Press.
10. Y. Zhu, L. Liu, **J. Panneerselvam**, Z. Li, Credit-Based Incentives in Vehicular Ad Hoc Networks, *8th IEEE Int. Conf. Service Oriented System Engineering*, pp. 352 – 357, Oxford 2014. IEEE Press.
11. F. Hao, X. Mao, W. Wu, L. Liu, **J. Panneerselvam**, A Cloud-Oriented Services Self-Management Approach Based on Multi-Agent System Technique, *7TH IEEE/ACM Int. Conf. Utility and Cloud Computing,* pp 261-268, London 2014. IEEE Press.

Book Chapters

1. **J. Panneerselvam**, L. Liu and Yao Lu, Datacentre Event Analysis for Knowledge Discovery in Large-Scale Cloud Environments, *Cloud Computing, Computer Communications and Networks*, doi: 10.1007/978-3-319-54645-2_14, 2017.
2. **J. Panneerselvam**, L. Liu, R. Hill, An Introduction to Big Data, *Application of Big Data for National Security*, Butterworth-Heinemann, pp 113-139. ISBN: 9780128019672, February 2015.
3. **J. Panneerselvam**, L. Liu, R. Hill, Requirements and Challenges for Big Data Architectures, *Application of Big Data for National Security*, Butterworth-Heinemann, pp 113-139. ISBN: 9780128019672, February 2015.
4. T. Holding, **J. Panneerselvam**, L. Liu, Migrating to Public Cloud: From a Security Perspective, *Guide to Security Assurance for Cloud Computing,* Springer, pp. 31-50, March 2016.

Abstract

Given the evolution of Cloud Computing in recent years, users and clients adopting Cloud Computing for both personal and business needs have increased at an unprecedented scale. This has naturally led to the increased deployments and implementations of Cloud datacentres across the globe. As a consequence of this increasing adoption of Cloud Computing, Cloud datacentres are witnessed to be massive energy consumers and environmental polluters. Whilst the energy implications of Cloud datacentres are being addressed from various research perspectives, predicting the future trend and behaviours of workloads at the datacentres thereby reducing the active server resources is one particular dimension of green computing gaining the interests of researchers and Cloud providers. However, this includes various practical and analytical challenges imposed by the increased dynamism of Cloud systems. The behavioural characteristics of Cloud workloads and users are still not perfectly clear which restrains the reliability of the prediction accuracy of existing research works in this context.

To this end, this thesis presents a comprehensive descriptive analytics of Cloud workload and user behaviours, uncovering the cause and energy related implications of Cloud Computing. Furthermore, the characteristics of Cloud workloads and users including latency levels, job heterogeneity, user dynamicity, straggling task behaviours, energy implications of stragglers, job execution and termination patterns and the inherent periodicity among Cloud workload and user behaviours have been empirically presented. Driven by descriptive analytics, a novel user behaviour forecasting framework has been developed, aimed at a tri-fold forecast of user behaviours including the session duration of users, anticipated number of submissions and the arrival trend of the incoming workloads. Furthermore, a novel resource optimisation framework has been proposed to avail the most optimum level of resources for executing jobs with reduced server energy expenditures and job terminations. This optimisation framework encompasses a resource estimation module to predict the anticipated resource consumption level for the arrived jobs and a classification module to classify tasks based on their resource intensiveness. Both the proposed frameworks have been verified theoretically and tested experimentally based on Google Cloud trace logs. Experimental analysis demonstrates the effectiveness of the proposed framework in terms of the achieved reliability of the forecast results and in reducing the server energy expenditures spent towards executing jobs at the datacentres.

Acknowledgement

First and foremost, I would like to thank my supervisors Professor Lu Liu and Prof Nick Antonopoulos: it's difficult to express my gratitude to them for giving me the opportunity to join the research group at the University of Derby and supporting me in my research pursuits. It would have never been possible for me to grow as a researcher without the continuous and extensive support of Professor Lu Liu. I would also like to thank my examiners Professor Omer Rana and Professor Liangxiu Han for their valuable comments and constructive evaluation of my thesis.

I would also like to thank immensely my colleague James Hardy who acted as a huge inspiration and spent quality time with me during my journey in the past three years. I also want to extend my gratitude to my colleagues and friends within the college of Engineering and Technology at the Univesity of Derby.

Last but not least, I am indebted to my wife Nandhini, whose continous love and encouragement throughout my journey has been indispensable. I would like to thank my mum Malathi and my brother Mahesh for their support and love, allowing me to reach this life milestone.

1 Introduction

1.1 Outline

This chapter introduces the motivation, context and the aims and objectives of this research. These include introducing the context of energy efficiency in Cloud datacentre operation and the challenges in achieving a complete energy solution for operating Cloud resources without affecting the availed service quality. It further presents the methodological approaches adopted for accomplishing the research objectives and formally lists the contribution of this research, namely the development of novel prescriptive analytics approaches for Green Cloud Computing. The organisation of this thesis is presented at the end of this chapter.

1.2 Research Motivation

In the recent years, energy consumption of the Information and Communications Technology (ICT) is being researched widely from different perspectives with the motivation of achieving eco-friendly operation. The evolution of the Cloud Computing technology [2] in recent years has attracted various user groups and business domains to adopt the same, and as a consequence ICT is witnessing a rapid macro-level transition in the energy expenditures spent towards datacentre resources. There has been an increasing awareness [3] of the environmental implications of Cloud Computing around the world. Government of Japan has announced the Green Government Cloud, which is mainly targeted to reduce the energy consumption of the datacentre resources. Altogether, there is an unsettled issue on how and to what extent ICT affects society, environment and economy. The motivation of this research work originates [4] from the Gartner's blog 'sustainability strategy in datacentres will become a key component of corporate excellence', and from the commonly prevailing fact that good service quality and energy efficiency cannot co-exist. The former necessitates information and data based analytics for obtaining a comprehensive datacentre footprint that identifies sustainability as a core element. The latter necessitates analytics approaches capable of foreseeing the implications of reducing the level of resource utilisation upon achieving a smoother execution of user requests. While it is apparent to keep the IT resources running and energy costs down, comprising the service quality resulting from the unavailability or delaying the resource availability is undesirable for both Cloud providers and users. One approach to accomplish this requirement is to develop a methodology which can optimise energy efficiency whilst availing on-time availability of sufficient datacentre resources to ensure successful execution of user requests.

In brief, the focus of this research is on the energy efficient utilisation of datacentre resources by achieving maximum utilisation of minimal number of operating servers. Thus the challenge is to optimise the trade-off between energy efficiency of the server resources and the level of Quality of Service (QoS) offered to clients when executing their tasks at the datacentres. The QoS here is reflected on provisioning sufficient level of resources to user requests to ensure smoother execution.

1.3 Research Context

Cloud Computing is turning out to be a promising computing paradigm and stands as an alternative to costly deployments of hardware and software resources. Cloud Computing paradigm has emerged as an efficient approach which enables ubiquitous, on-demand [5] network access to a shared pool of flexibly reconfigurable computing resources including networks, servers, storage, applications, and services that can be rapidly deployed with minimal management effort or service provider interactions. The low-cost, any-time and any-where compute supplements of Cloud Computing has increased not only the users of Cloud but also Cloud providers in the commercial market. This naturally increases the deployments of Cloud datacentre resources across the world and IT-centric organisations are now deploying their own Cloud resources for transforming their existing business model to Cloud Computing rather than outsourcing their IT needs. This increasing deployment of datacentres leads to two immediate implications. The positive implication is that IT centric services are evolving into a new revolutionary era of narrowing the digital divide between urban and remote regions through distant datacentre services, so that remote location is no more a hindrance for IT services. The negative implication is that Cloud datacentres consume monumental amounts of energy and are becoming a major source of environmental pollution through their excessive carbon emissions. IT experts have issued a warning that datacentres are expected to consume three times as much energy in the next decade as of 2016, 416.2 terawatt hours of electricity has been used by datacentres across the world in the year of 2015, which is far too many than UK's total power consumption.

Cloud services are generally provided over the internet with the participation of clients (requesting services) and service providers. In this service concept, a mutual agreement called the Service Level Agreement [6, 7] (SLA) will be signed initially between service providers and Cloud users. SLA is a contractual agreement, in which, user demands in terms of resources, execution time, and the expected Quality of Service (QoS) will be negotiated. Cloud providers

offer their services within the bound limits of SLA, and maintaining the promises of this SLA will reflect the reputation of service providers. Cutting down the IT running energy costs generally involves the shutting down of idle resources in the datacentres, since idle resources will consume significant amounts of energy. This strategy also incurs an additional wait time when the turned-off resources are booted again to ensure resource availability during increasing resource demand. The wait time usually depends on the wake-up latency levels of the machines and directly reflects on the wait time of users before their requested resources become available. Breaching the SLA resulting either from service unavailability or delaying service availability affects the reputation of the providers, thus it is always a gamble for the providers to make a decision about shutting the server resources. But the providers are under immense pressure to reduce their carbon footprints for an eco-friendly datacentre operation.

Resource idleness usually results from the unutilised or under-utilised server resources during datacentre execution, such under-utilisation is a direct implication of over provisioning the resource levels for user requests than their actual requirement during execution. It is commonly witnessed that resource levels are often over-provisioned and majority of the resource time are actually wasted during task execution. Despite the emergence of Cloud Computing in the past decade, the characteristic features of Cloud workloads and behaviours of Cloud users are still not perfectly clear. Precise and prior knowledge of Cloud workload and user behaviours is crucial in important decision making whilst attempting to scale server resources and provisioning resource levels appropriately for user requests without incurring resource idleness and wastage of resource times.

With this in mind, this research is addressing the vision of achieving both energy efficiency and performance optimisation, driven by an in-depth prescriptive analytics of the Cloud systems. Prescriptive analytics is the act of identifying the best possible course of action for a given scenario. The analytics approach presented in this thesis exploits the historical trace profiles of individual jobs to extract insights about the past behaviours of jobs to anticipate their future possible behaviours at the datacentre. With the prior and anticipated behavioural inferences, the proposed approach is modelled to postulate the best possible course of energy management for individual job cases. The novelty beyond the state-of-the-art lies in the strategy of dynamically balancing the trade-off between energy efficiency and QoS. This strategy of availing Cloud Computing services will create a new innovation in high performance computing by leading the way for sustainable ICT for greening the datacentres.

1.4 Problem Statement

a) Is it possible to conduct an in-depth descriptive analytics of Cloud datacentre events for obtaining comprehensive inferences for characterising Cloud workload and user behaviours?
b) Is it possible to conduct an in-depth predictive analytics on Cloud workload and user behaviours to forecast their anticipated future events at the datacentres?
c) Is it possible to develop analytics methodologies for achieving an energy efficient scaling and provisioning of the server resources driven by the prescriptive knowledge of user and workload behaviours?

1.5 Research Aim and Objectives

This research is aimed at developing novel prescriptive analytics approaches to exploit the usage profiles of Cloud workloads to foresee their future behaviours at the datacentres, ultimately to reduce the energy expenditures spent towards the server resources whist ensuring uninterrupted and successful completion of incoming workloads. In accomplishing this research aim, the objectives of this research include the following.

a) A comprehensive literature review of the existing state-of-the-art methodologies in achieving energy efficiency in Cloud Computing with the motivation of identifying their drawbacks and prevailing issues. This includes reviewing the efficiencies of the existing techniques in achieving energy efficiency without degrading the performance quality and addressing the requirements towards an efficient energy saving approach with balanced energy and QoS trade-off.
b) Identification and quantification of the presence of proportional idle resource time during task execution resulting from over-provisioning of resources for task execution. This includes an empirical analysis of usage profiles of jobs processed in a real world Cloud datacentres, in order to expose the cause and implications of resource idleness and to project the scope of reducing the actually spent energy expenditures.
c) An in-depth analytics of the Cloud datacentre events and footprints to study and characterise Cloud workload and user behaviours. This includes exhibiting the intrinsic characteristics of Cloud workloads and user behaviours to extract and project the predictive properties of Cloud workload and user behaviours.
d) Development of novel prescriptive analytics approaches which exploits the datacentre footprints to predict the anticipated future behaviours of Cloud workload and user

behaviours, in order to drive important decision making in scaling server resources and provisioning resource levels for task execution.

1.6 Research Methodology

The research methodology of this work encompasses two integrated components.

a) Descriptive analytics of the datacentre footprints employs quantification and distribution analysis to characterise Cloud workload and user behaviours. This analysis is decomposed into four main areas: exposing and modelling the characteristics of zombie servers in a large-scale datacentre execution, characterising latency sensitivity of the workloads upon energy efficiency, exposing workload heterogeneity, extracting and modelling the inherent periodicity among Cloud workload and user behaviours.
b) An approach to address the energy implications of Cloud datacentres through the development of analytics approaches which leverages a range of statistical techniques and exploits the descriptive analytics to predict the anticipated Cloud workload and user behaviours. This includes predicting user behaviours in terms of their workload submission pattern, user session duration, number of anticipated job submissions, and forecasting the workload behaviours in terms of their resource consumption levels and execution behaviours during actual execution at the datacentres.

1.7 Major Contributions

a) The study and quantification of the energy profiles of operating servers within a datacentre environment for identifying the presence of proportional idleness among the provisioned resources and quantification of the energy wastages incurred by the under-utilised resources resulting from resource idleness during task execution. Further the inherent cause of such proportional resource idleness have been empirically exposed.
b) An in-depth methodical analytics of a large-scale datacentre events for identifying and modelling the predictive and periodical characteristics of Cloud workloads and users. An empirical analysis of the latency sensitivity levels of Cloud workloads and their corresponding impacts upon the undesirable datacentre energy consumption.
c) A novel analytics framework for predicting user behaviours in terms of their job submission interval, session duration and number of anticipated submission has been systematically developed, and its correctness has been verified.

d) A novel analytics framework for estimating the resource consumption levels of the arriving workloads at the back-end servers for postulating the most appropriate level of resource provision for task execution with minimal amounts of resource idleness during actual execution. This also includes the development of an integrated analytics framework to classify tasks within a given job based on their resource requirements, based on which the recommended level of resource provision for tasks have been optimised to reduce task termination probabilities.

1.8 Thesis Organisation

Chapter 2 presents a detailed survey of the state-of-the-art methodologies in reducing the energy consumption levels of the datacentres for the purpose of identifying the research gaps, critical issues and drawbacks in the existing techniques of energy efficient datacentre operation. Further, an in-depth literature and background study has been presented in the context of Cloud computing.

Chapter 3 presents an in-detail empirical analysis of the undesirable energy consumptions incurred by the proportional resource idleness resulting from over-provisioned resource levels for executing tasks at the datacentres. This chapter further characterises the periodicity of Cloud workloads and users and presents an empirical analysis of the latency sensitivity levels of Cloud workloads and their corresponding impacts on the undesirable energy consumptions at the datacentres.

Chapter 4 presents the developed tri-fold prediction framework for forecasting user behaviours in terms of their job submissions in large-scale datacentre environment. The dependability of this framework and the reliability of the forecast results have been empirically verified by training traces of Google Cloud datacentre events into the prediction framework.

Chapter 5 presents the proposed integrated analytics framework for estimating the resource consumption levels for user requests and classifying task behaviour categories in order to recommend the most appropriate level of resource levels to be provisioned for task execution.

Chapter 6 summarises the contributions of this research study. The remarks and conclusions highlights the research accomplishments, applicability and limitations of this research study and outlines the future research directions.

2 Literature Review

2.1 Outline

This chapter formally presents a literature review on Cloud Computing from the viewpoint of energy efficient computing and presents the survey of the existing energy efficient techniques used in Cloud datacentres. The state-of-the-art literature review of existing works concludes with critical requirements of this research study.

2.2 Cloud Computing

2.2.1 Cloud Datacentre Environment

A typical Cloud Computing process environment is composed of massive datacentres encompassing several numbers of servers hosting the operation of the Virtual Machines (VMs). Jobs arriving at a datacentre for processing are usually scheduled and provisioned with appropriate resources and are executed at the back-end servers. The server resources are orchestrated to execute tasks effectively within the allocated resource levels. A scheduler in the datacentre receives jobs, finds and allocate the most appropriate level of resources based on job requirements. Now-a-days multiple Cloud providers are joining hands to offer a collaborated Cloud service to the clients. This multi-provider service model facilitates the exchange of services or hiring an external service during service unavailability. The exchange of services within multiple Cloud datacentres are enabled by the meta-schedulers and Cloud-brokers [8-10] which are responsible for identifying and locating the desired services based on job requirements. Other than the compute and storage components, datacentres also comprise cooling components to maintain the reliability and longevity of the server resources such as air-conditioning, water-based cooling systems etc.

2.2.2 Virtualisation Technology

A key technology that enables implementation, application execution and services in the Cloud is Virtualisation [11-13]. Virtualisation is the act of creating substitutes for the real resources with the same functionalities and interfaces, but differ in size and performance capabilities. Such substituted virtual resources are utilised for the deployment of multiple emulated computing environments defined as Virtual Machines, which can be described as a software implementation of the computing resources deployed on a single physical host server. The virtualisation-based Cloud Computing provides a new supplement and delivery model for

delivering services over the Internet. Here, virtualisation refers to the abstraction of computer resources, such as the process of running two or more operating systems on a single set of physical hardware with strong isolation. Virtualisation technology enables a system administrator to combine disparate physical computing systems into virtual machines with maximally utilised host resources.

Virtualisation technology is used in several ways of achieving hardware abstraction and depending on the level of resource abstraction, virtualisation technology falls into three categories such as Full Virtualisation, Para Virtualisation and Operating System-level Virtualisation. Full Virtualisation is the concept of abstracting the underlying physical host hardware resources to encapsulate a number of VMs. Such VMs are deployed with strong isolation and are unaware of the virtualised host. The access to these VMs and their resource allocation are facilitated through a hypervisor or a Virtual Machine Monitor (VMM). Para virtualisation involves modification of the Operating System and the deployed VMs are aware of their virtualised environment. The resource demands and requirements of other running VMs are transparent to all other VMs encapsulated onto a single physical host. Operating System-level virtualisation involves virtualising the host OS rather than virtualising the actual physical host. All the deployed VMs share a homogenous OS and this type of virtualisation allows allocation of resources during the deployment of VMs and change them dynamically during run-time.

2.2.3 Cloud Workloads

Cloud based application workloads are very dynamic [14] and distinctive in nature, and the resources of the Cloud environments should exploit their native properties such as rapid elasticity, resource sharing, instant scaling etc., in such a way to handle the incoming workloads in an efficient way. Different workloads will have different processing requirements such as CPU, memory, throughput, latency, execution time etc. Cloud datacentres are facing more commercial and interactive workloads at higher frequencies unlike the traditional grids which usually encounter heavy computation-intensive scientific workloads. In other words, Cloud workloads often come from web services such as search and retrieval queries, online documentations, and Map Reduce jobs etc. Such real streaming workloads are usually shorter with their execution duration and are submitted at shorter intervals. In spite of their submission frequency, Cloud workload arrival pattern encompass higher level of noise than the traditional grid workloads.

A typical Cloud workload arrives at the Cloud datacentre in the form of jobs [15] submitted by users. Every job includes certain self-defining attributes such as the submission time, user identity and resource requirements in terms of CPU, memory and disk space. A single job may contain one or more tasks, which are scheduled for processing at the Cloud servers. A single task may have one or more process requirements. Tasks may have varied service requirements and characteristics such as throughput, latency, jitter, etc., even though they belong to the same job. Two jobs with the same resource requirements may not be similar in their actual resource utilisation levels because of the variations found among tasks contained within jobs. The provider generally records the resource utilisation levels of every scheduled task and maintains user profiles. Cloud workloads behave distinctively with different server architectures and this behaviour of workloads in the Cloud processing environment is more strongly correlated with the CPU cores than with RAM capacity of the machines at the server level. The resource utilisation patterns are usually more dynamic and vary abruptly with different workloads under different server architectures. The dynamic parameters of the server architectures are usually calculated as the measure of the number of cycles per instruction for CPU, and memory access per instruction for memory utilisations respectively.

Cloud workloads are governed by various intrinsic attributes which determines their characteristics. Job execution duration and number of tasks within jobs are the two explicit metrics defining workload characteristics. Job execution duration [16] is usually bimodal, with tasks contained within jobs either run for a shorter time or a longer time. Long running tasks can be further classified as user facing tasks and compute intensive tasks. The former runs continuously with quicker user interactions and the latter generally refers to the processing of the weblogs. Shorter duration tasks can be further classified as highly parallel user requests of both CPU and memory resources, shorter CPU and shorter memory intensive tasks respectively. Majority of the Cloud jobs run for less than 15 minutes and a very few number of jobs are more than 300 minutes in duration, with the duration of latency sensitive jobs [17] being less than 30 minutes on average. Task duration heavily depends on the nature of user behaviours and their interactions. Generally, a single job may contain tasks of both shorter and/or longer durations, and tasks running longer usually consume most of the allocated resources. It is worthy of note that jobs are generally governed by various constraints such as specified server and scheduling requirements. An efficient Cloud infrastructure effectively manages such constraints, still a few type of jobs can encompass more than 400 constraints [17] thereby affecting their execution duration. Most of jobs in a typical Cloud datacentres

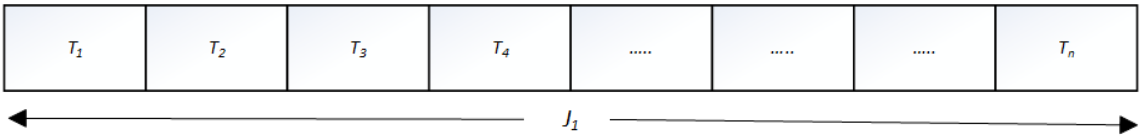


Figure 2.1. Logical Representation of Jobs and Tasks

encompass smaller to medium number (100 on average) of tasks, and a very few number of jobs have a single task. On the contrary, a very few jobs may also contain more than 2000 tasks. Thus majority of the Cloud users submit jobs with smaller number of tasks and a very few users submit jobs with larger proportion of tasks. Both the smaller number of jobs with increased number of tasks and the larger number of jobs with fewer tasks have distinctive impacts on the overall datacentre behaviour. A logical representation of a job J_1 encompassing n number of tasks from T_1 through to T_n is presented in Figure 2.1.

2.2.4 Cloud Users

Jobs submitted to a Cloud datacentre are usually driven [18] by Cloud users. In a Cloud datacentre, job submissions are usually characterised by associated user ID, assigned logical name and the corresponding resource requirements based on user needs. Jobs submitted under the same user name implies that all such jobs are submitted by a single user. This user name is a randomly allocated string and are uniquely assigned to users. A single user may also have various user profiles with different user IDs, which would lead to the generation of various user driven profiles for a single user. Such user profiles might exhibit similar user behavioural patterns since such multiple user profiles belong to a single user. Despite this inherent similarity among user profiles, matching the ownerships of jobs submitted under different user profiles is often tedious. This is because every single user will be assigned with a unique user ID, and when users have multiple profiles, a single user might have multiple user IDs. But jobs characterised with similar behavioural patterns closely correlating with noted user profiles can be treated in a common way in Cloud analytics to match the workload ownership.

Furthermore, the active number of concurrent users in an execution session is another important factor that affects resource utilisation. Users co-existing in a service session do not necessarily have similar resource requirements as they generate workloads of different business domains. Usually, Cloud providers employ a higher level of parallelism under an increased number of concurrent users requesting similar resources. The demand for CPU cores generally increases under an increasing number of concurrent users, with the demand for CPU being 20% under 100 concurrent users and 70% under 300 concurrent users presented [17] in a datacentre

analysis. User profiles are very dynamic in a way that every user has a potential access pattern and do not have a static IP address as it is dynamically assigned to users from a limited number of address pools. This often leads to the assignment of the same IP address to several users. User behaviours evolve over time, and thus the snapshots of user profiles obtained over a relatively shorter period are mostly imprecise.

2.2.5 Cloud Dynamicity

Both Cloud workloads and the datacentre resources [19] exhibit extreme dynamic characteristics. A Cloud datacentre environment is composed of heterogeneous servers operating at different utilisation levels, different Operating Systems, different processing capabilities, and encompass components from different manufacturers, etc. The multi-provider cloud service model with interoperable service features further increases the datacentre heterogeneity. Furthermore, the workloads being submitted show heterogeneities and dynamism in their actual resource consumption and resource utilisation. For instance, workloads with similar resource requests may not be similar in their actual resource utilisation patterns at the back end Cloud servers. Increasing levels of resource intensiveness have increased levels of energy implications by demanding increased allocation of resources. This diversity necessitates the need for understanding the characteristics of job submitted by users for the purpose of allocating tasks accordingly with optimum provisioning of the server resources.

The heterogeneity found among both Cloud users and the workloads causes the workloads to behave dynamically in a datacentre environment. With the Cloud server resources exhibiting heterogeneity among their operating conditions, process capabilities etc., dynamism [20] is evident among the workloads in terms of their arrival frequency, resource requests and utilisation levels. Such a dynamic nature of the datacentre process environments impose various levels of challenges in carrying out a real-time analytics of Cloud workload behaviours

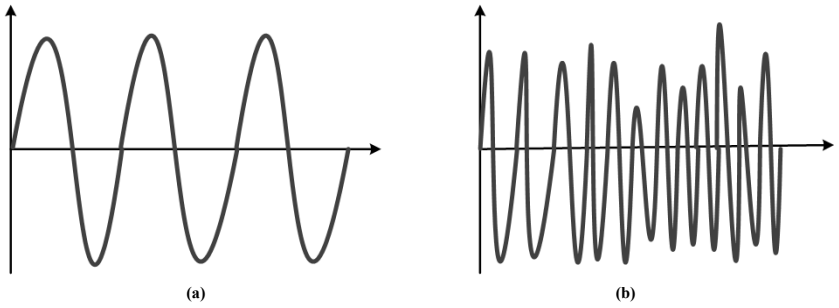


Figure 2.2. Cloud dynamism (a) Constant workload (b) Fluctuating workload

for driving effective decision making. The complexities in decision making driven by workload analytics can be demonstrated in two different scenarios, as shown in Figure 2.2. Firstly, job submissions varying in accordance with a periodic oscillation of constant amplitude. In such scenarios, predicting either the peaks or the valleys can be exploited to manage the datacentre resources by the way of scaling up/down the active resources at an optimum level based on the curve trend. Secondly, job submissions varying abruptly with the submission curve characterised by uneven amplitudes and very close occurrences of peaks and valleys. Given the two scenarios, the former allows reasonable time interval where a much better datacentre management can be achieved. But the later allows a lesser time scale to carry out the predictive analytics and further datacentre management driven by the prediction.

Another challenge imposed by the workload behaviour is job submissions from brand new users. Such newly arriving jobs may or may not have a pre-existing user or job profiles to which they can fit into. If they don't fit into an existing profile, then new user and job behavioural profiles should be created for every new users. Cloud workloads may also contain anomalies [21] and job submissions from malicious users. Such anomalies generally exhibit an abnormal behavioural pattern. Some of the runtime factors such as user access patterns, user concurrency, and resource usages often result in contextual anomalies which are unavoidable in Cloud environments. It is possible that these anomalies could also be categorised as newly arriving jobs submitted by brand new users. Conversely genuine workloads might also be classified as anomalies, which would result in a higher number of false negatives. Such a classification leads to wrong prediction, further causing service outages and execution failures.

2.3 Datacentre Energy Consumption

Generally, the energy consumption of the datacentres is determined by the operating resources and their corresponding workloads, particularly the CPU utilisation. In today's datacentres, processors [22] are the largest energy consumers among the operating resources and the choice of the CPU has a significant impact on the power consumption of the servers. It is worthy of note that CPU intensive workloads are resource hungry and thus account for more energy consumption than the memory counterpart. The power consumption of the servers is a proportional [23-25] function of the CPU utilisation, and is considerably low with minimal CPU utilisation. The number of working cores [26] in a multi-core processor has a major impact on the energy efficiency, as it heavily affects the processor power consumption. The datacentre energy consumption level is proportional to CPU workload, usually when the server

workload increases a corresponding increase in the server power can be witnessed. In other words, the degree of increase in the server power depends on the proportionality of the system workload, this progression is known as Moore's Law. This is due to the relationship between CPU clock frequency and the execution time determining the energy consumption. Furthermore, the processor throughput (measure of server capacity) is approximately proportional to the CPU frequency.

In the Cloud Computing service model, client demand requirements are satisfied by purchasing resources, this is typically at a level which exceeds the actual amounts of resources necessary to process their prospective job. Cloud providers are then obligated to provision the resources purchased by the clients, and as a consequence, resources are exceeding over-provisioned than the actual requirements in order to meet the SLA and to maintain QoS at the desired levels, resulting in most of the server resources operating below their actual capabilities. Another primary cause for excess energy consumption is that most of the resource utilisation is wasted due to early terminations of the workloads caused by both the providers and users. The commonly recorded termination events [17, 20, 27] are categorised as FAIL, KILL, EVICT and LOST events. FAIL events are the termination of the workloads due to failures and these jobs are generally rescheduled and restarted. FAIL event occurs when the specific resource requirements are not satisfied within the allowed processing time or due to natural machine failures. FAIL is a natural event which usually occurs without any external interventions from both providers and users. KILL event occurs when jobs are forced to be terminated, triggered by either users or providers. Usually a job or a task is forced to terminate or killed explicitly whenever the execution does not go in the desired direction, or when jobs or tasks exceed their allocated resource limits.

In a healthy Cloud environment, the relative probabilities of jobs being killed should remain stable and jobs should rarely fail, thus such environments are considered to be more energy efficient. Thus, a complete energy utilisation and resource utilisation profiles can only be obtained from the finished jobs. The total amount of resources consumed for successful completion of a job include those resources wasted due to the terminations of that particular job. Despite re-scheduling jobs facing termination, some of these jobs are actually dropped at the time of termination without attempting re-execution. All these termination and re-execution events account for undesirable energy and resource expenditures at the datacentres.

In addition, undesirable energy consumption is incurred in other active components of the datacentre infrastructures such as network components, air distribution, environment cooling system, etc., all [2] causing substantial amounts of CO2 emissions. Energy spent on the cooling system in order to maintain the optimum temperature of the operating CPUs cannot be undermined. It is being argued that a significant proportions of energy spent on running the server is required to cool the corresponding server and the server operating costs [28] are rapidly increasing.

2.3.1 Power Usage Effectiveness

Power Usage Effectiveness (PUE) [29] is an important metric to define the energy efficiency of the datacentres. PUE [30, 31] defines how effectively a server is using its electricity, which is always desirable to be at an optimum level to achieve energy-efficient computing. PUE is determined as the ratio of the amount of power fed into the datacentre to the amount of power used to run the computing infrastructure within the datacentre. The total facility power is usually the power used to run the power units such as UPS, remote power panels, battery backups, cooling units etc., and the IT equipment power is usually the power used to run servers, storage, communication equipment etc. Thus the PUE is computed as,

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power}$$

A PUE closer to 1.0 is very effective, implying that almost all the energy is transformed into computing power. A PUE of 2.0 means that every computationally useful watt of input power will require an additional watt for cooling, lighting, power distribution, etc. A PUE closer to 3.0 reflects that the datacentre has a very poor energy efficiency. With the world average PUE [32] being reported as 1.7, there is still a lot of scope for enhancing the PUE of datacentres.

2.3.2 Energy-Performance Trade-off

Cloud providers are contractually committed to provide services without violating the terms of the initially negotiated SLA (Service Level Agreement). The SLA is paramount in quantifying the attributes used to define, measure and maintain the Quality of Service (QoS) at a desired level and for setting and managing the Quality of Expectations (QoE) of the end users. While it is possible to simultaneously reduce environmental and energy impacts and the cost of service provision by reducing the active number of server resources at the datacentres, however this may result in lower levels of resource availability which could in turn causes temporal loss of service, jeopardising the agreed SLA, incurring financial penalties and damaging the

commercial credibility of the provider. Service unavailability is an unacceptable event for Cloud users as they have purchased a contract based on the structure and service appeal of Cloud Computing, namely an illusion of infinite resource availability. Though energy efficient computing is totally within the domain of the service providers, Cloud users are likely to be the victims of service outages which would generally occur when server resources are shut down early or started late in an effort to conserve energy. On the contrary, though any time availability of resource is the desire of users, running the server resources idle is not an ideal state of datacentre operation from the energy perspectives of the providers.

This forms the key research topic of this research study. A maximally utilised server resource can reduce the active number of servers and a prior knowledge of user demands can help the providers to avail effective services to their clients. Further, exposing the cause of excess energy consumption in Cloud datacentres is an integral requirement of the Cloud service model not only to achieve energy efficiency but also to maintain optimised performance quality.

2.3.3 Stragglers

Process level constrains are commonly witnessed during a task execution in a Cloud datacentre resulting from various reasons such as server heterogeneity, resource contention, data skew, incoming traffic queue, aggressive consolidation of tasks, data reliability, network constraints etc. Such process level constraints significantly affects the performance of the server nodes and thereby affects the progress of task being executed in the corresponding nodes. This phenomena usually causes tasks to run longer than their normal duration [33] termed as stragglers or long-tails. Since tasks within a single job are processed across a set of server nodes in Cloud datacentres, a few straggling tasks usually a smaller proportion within a single job considerably affect the completion of the entire job to an irresistible margin.

Stragglers are of two types based on their locality such as the node-level stragglers and task level stragglers. Whilst the former is usually identified among the running physical servers the latter results depending on the nature, characteristics and requirements of tasks themselves. Server nodes exhibiting poor performance and declining process capability can cause node-level stragglers, naturally tasks scheduled on to such node-level stragglers suffer performance constraints and face possible terminations and prolonged execution duration than anticipated. Tasks level stragglers naturally exhibit a varied behaviour than the other co-located tasks within the same job in terms of their resource consumption, execution duration etc. Whilst the node-level stragglers can be mitigated by avoiding scheduling of jobs onto such poor nodes, task-

level stragglers pose an increased management complexity since they can hardly be identified before they occur. Further the existence of the relationship between node-level and task-level stragglers is trivial, as our empirical analysis demonstrated that a common task-level straggler in two different execution instances of the same job has been scheduled onto different server nodes.

2.4 Cloud Workload Categorisation

Cloud workloads have been categorised in accordance to their incoming arrival pattern which includes the rate of arrival, frequency at which jobs are submitted and the nature of tasks in relation to user behaviours. User behaviour is the act of submitting jobs, which particularly refers to their submission trends, resource estimation ratios, session duration etc. The incoming workloads are categorised from the Cloud provider's point of view, with the vision of how the workloads affect resource provisioning at the datacentre resources. From the perspectives of the Cloud service providers, the incoming Cloud workloads can be categorised into five major types [34] as static, periodic, unpredictable, continuously changing, and once-in-a-lifetime workloads.

2.4.1 Static Workloads

Static workloads usually come from more or less constant user behaviours without any unexpected spikes in their arrival behaviour. Such workloads are characterised by flat utilisation profiles with defined boundaries, and fairly utilises the Cloud resources over the determined time period. Since such workloads are always bound to known bound limitations, rapid elasticity and sudden scaling of the Cloud resources are hardly required for executing the static workloads. Instant addition and removal of resources such as CPU, memory, network bandwidth etc., are not usually imposed by such workloads, since their incoming workload range is almost a constant or changes only by a small margin over time. Slightly over-provisioned resources than the current utilisation thresholds are always in the interests of the Cloud providers which enables them to satisfy user demands and also to reduce the energy consumption of their active resources. Examples of static workloads include private website queries, wikis, etc.

2.4.2 Periodic Workloads

Periodic workloads are characterised by a definite increase/decrease patterns in the arriving volume of workloads at regular time intervals, and this increase/decrease trend also reflects in

corresponding scaling up/down of the provisioned resources. Such workloads often results from the periodic day-to-day activities and repeating business behaviours. Though periodic workloads follow a definite incoming pattern, they might also turn out to be unpredictable occasionally. While provisioning the server resources in accordance to the arriving volume of workloads, sudden spikes in the periodic workloads demand rapid scaling of the resource levels. Rapid elasticity for increasing workload levels and decommissioning the resources in parallel to the diminishing workloads could benefit the Cloud providers with both effective task execution and efficient energy management. Examples of periodic workloads include weather forecasts, monthly bills, online ticketing during peak/off peak travel time-of-the-day etc. Such jobs often occur only at certain periods of the month or a day but include more number of people performing similar jobs during such peak periods casing sudden spikes in the periodic workloads.

2.4.3 Unpredictable Workloads

Similar to the periodic workloads, unpredictable workloads follow an increasing/decreasing pattern in the incoming workload volumes. Unpredictable workloads are characterised by random and unanticipated utilisation of the Cloud resources with the maximum risk of execution failures. Cloud providers are often faced with unplanned rapid provisioning of the resources in order to satisfy the process requirements of the unpredictable workloads. In the context of user behaviour modelling, such types of workloads are the most challenging and impose increased level of complexities in determining their resource requirement levels. Constant monitoring of the workloads and readily available state of the resources is one potential solution for executing the unpredictable workloads without affecting the QoS, but such strategies incur excess energy consumptions.

2.4.4 Continuously Changing Workloads

Continuously changing workloads are the type of workloads showing gradual increase/decrease trend in their incoming arrival pattern. The resource consumptions of such workload pattern are experienced by gradual expansions and shrinks in accordance to their traffic arrival rate. The variation in the traffic arrival pattern of such type of workloads can be determined easily and such variations are usually linear, non-linear, exponential etc. Continuously changing workloads allow easy resource provisions by constant monitoring of the incoming workload traffic. Since the variation of the workload traffic is a determinable factor and also not very intense, both the planned and unplanned resource provisions can be

easily achieved by the Cloud providers. Resource provisions closely elastic to the constant variation of the workload traffic would benefit the Cloud providers to easily satisfy the resource requirements and also to reduce the excess energy consumptions. Examples of such workloads include newly arrived products requiring more support in the beginning and diminishing as the product gets older.

2.4.5 Once-in-a-lifetime Workloads

Once-in-a-lifetime workloads exhibit the typical characteristics of constant arrival pattern for a prolonged time interval, together with a strong peak arrival rate of the workloads occurring rarely followed by prolonged constant arrival pattern of the workloads. Such pattern of workloads results in a rare peak utilisation of the Cloud resources in a range much greater than the range of the periodic workloads. Such type of workloads are generally predictable since user requirements are known in advance. But such type of workloads often challenges the availability of the resources for accomplishing large-scale resource provisions. Deploying a vast range of resources for these rare peaking workloads are not an ideal solution for the Cloud providers. One possible solution for such workloads is the resource pool sharing with inter-clouds by the way of exploiting the interoperable properties of Cloud Computing. Examples of such workloads include the yearly University admissions, sports events etc. Figure. 2.3 illustrates the arrival pattern of various workloads witnessed in the Cloud datacentres.

2.5 Cloud Workload Characteristics

In general, the different forms of workloads from the provider's perspectives include computation intensive requiring larger processing and smaller storage, memory intensive requiring larger storage and smaller processing, and workloads requiring both larger processing and larger storage, and communication intensive requiring more bandwidth capacities etc. Workloads are usually measured in terms of user demands, computational load on the servers, bandwidth consumption (communication jobs), and the amount of storage data (memory jobs)

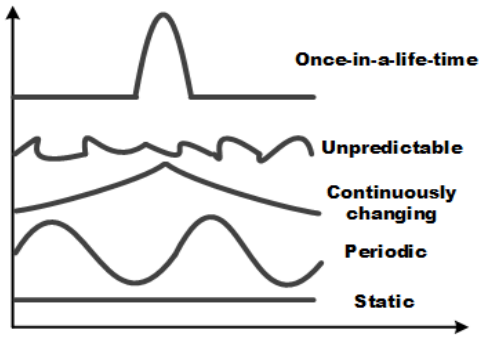


Figure 2.3. Cloud Workload Categorisation

etc. User demand behaviour modelling requires an in depth quantitative and qualitative analysis of the statistical properties and behavioural characteristics of the workloads including job length, job submission frequency, resource consumption of jobs etc., which insists that the initial characterisation of the workloads is more crucial in developing an efficient behaviour model. Rather than a stand-alone analysis of the above stated workload metrics, modelling the relationships between them across a set of workloads is more significant in order to achieve more reliable behavioural analysis. Statistical properties of the workloads are more significant since they remain consistent in longer time frames.

2.5.1 State Events

Generally, the execution of the workloads in the Cloud datacentres encompasses two different states such as the *session state* and the *application state*. *Statelessness* refers to the isolated portion of the running applications staying away from the execution. Cloud server resources requires this state information [35] about users and their applications for establishing a stateful connection between users and the Cloud resources in order to complete task execution.

Session state refers to the interaction of users handled by the applications. It depends on the nature of user behaviours and their purposes for interacting with the Cloud services. The duration of this session state is determined by the sequence of user actions performed for satisfying their job requirements. It is an important statistical metric in both the analysis of the workload traffic pattern and also in the context of user behaviour modelling. *Application state* depends on the data handled by the applications. Application state is determined by the nature of job requests such as CPU intensive, memory intensive, and communication intensive etc.

2.5.2 Job Length

Job length is the execution duration of user requests. It is usually measured as the time between job submission and the successful execution of that corresponding job. Job length for a particular job depends on a wide range of factors including the state information, job nature, availability state of the Cloud resources, user requirements, workload traffic etc. It is common that jobs and tasks often face both natural and forced terminations at the datacentres. Terminated tasks are usually resubmitted at the datacentres for completing their execution. Job terminations significantly affect the duration of tasks execution. Interestingly, one or few number of tasks fail acting as stragglers, and significantly affect the completion of the entire job. Such terminated jobs and jobs containing stragglers usually face an increased job duration than those instances of failure-free execution. In the case of jobs submitted with multiple tasks,

the execution duration of the entire job is actually the summation of all task duration contained within the corresponding job for cascaded jobs and job duration is computed as the duration of the long running task within the given job for parallel jobs. It is also possible that job encompass both cascaded and parallel tasks.

2.5.3 Job Submission Frequency

Job submission frequency is the time duration between the consecutive submissions of identical or similar jobs having similar characteristics such as operational requirements, user requirements etc. Job submission frequency is also one of the determining factors of the workload arrival pattern, leading to the categorisation of the workloads with an even popularity factor [36] defining how often and how many times a particular job arrives at the datacentre.

2.5.4 Job Resource Utilisation

Resource utilisation is usually measured from a two-fold perspective as the resource consumption level of the individual jobs and the machine level usages. Job level resource utilisation is usually the measure of resources consumed by the workloads whilst executed by the server resources. The resource consumption is generally measured based on the CPU usages, memory consumption, bandwidth usages in accordance to the nature of job execution. Machine level resource utilisation is measured as the ratio of the actual usage level of a machine to the maximum usage capacity of that corresponding machine. It is an observed fact that the CPU usage ratios are much lower than the memory usage ratios in case of both the machine level and job level resource utilisations.

2.5.5 Self-Similarity

Self-similarity is usually determined across a set of workloads arrived during a particular session. Web session workloads are addressed to be self-similar by a majority of previous studies [35]. The periodic components (arrival pattern, execution nature, user behaviours etc.) of the workload traffic are exploited for deriving the self-similarity across the arriving workloads. Self-similarity is an important metric, and effective analysis of which can enhance the modelling accuracy. Thereby, benefits the Cloud providers with efficient resource management for serving the incoming workload traffic by effecting elasticity and appropriate scaling of the resources with respect to the workload requirements. Also, determination of the self-similarity among the incoming workloads helps to aggregate the self-similar workloads onto a minimal set of Virtual Machine clusters with limited server resources that could handle the workloads.

2.5.6 Burstiness

Burstiness in the workloads [37] can be defined as the highly variable request arrival rate deviating from the average arrival intensity by a significant margin. It is measured as the ratio of the average peak arrival rate to the actual arrival rate of the workloads under normal time period. Burstiness usually depends on the self-similarity character found across the set of workloads during the time of observation. Burstiness in the workload arrival pattern is undesirable [38] in the Cloud datacentres as it often leads to over provisioning of the resources, thus causes excess energy consumption. It is one of the important factors causing peaking utilisation of the Cloud resources and also crucial in the behaviour modelling, since wrong depiction of the burstiness across the set of workloads often causes errors to an irresistible margin.

2.6 Cloud Analysis and State-of-the-art Energy Efficient Techniques

2.6.1 Cloud Workload Characterisation

Cloud workloads and the datacentre environments have been analysed from various perspectives for various reasons. Business-driven Cloud workloads [39] have been analysed for exhibiting the repeating pattern and the inter-arrival time of the incoming jobs. Cloud workloads have been primarily analysed from two different perspectives: firstly workload diversity which involves quantifying and characterising the amounts of workloads over time and secondly workload distribution which is the probabilities of workloads remaining consistent across different inter-arrival times. The statistical characteristics [40] of business critical workloads have been comprehensively studied for promoting resource management mechanisms for datacentres. Requested and actual resource usage patterns have been studied in terms of CPU, memory and disk and network I/O, and workloads are characterised as having cyclic patterns.

Workload behavioural patterns [1] have been analysed from the context of both users and their tasks. The relationship between users and their tasks has been characterised and validated based on the submission rate, and estimation ratios of CPU and memory resources for user dimensions and task length, and utilisation ratios of CPU and memory resources for task dimensions. Conducting inter and intra-cluster analysis, workload variability has been derived from realistic Cloud trace logs, but the cyclic and periodic behaviours of the workloads and users activities have not been derived to a certain depth. Based on the same user and task dimensions, workload diversity [19] has been analysis and modelled, where the characteristics

and behavioural patterns of users and task variability have been modelled across different observational periods, and user patterns are proved to be more diverse than task diversity across different observation periods. Statistical profiles [41] of the workloads based on their resource utilisation levels have been derived by clustering workloads of similar patterns. It has been deduced that the workloads are tri-modal based on job duration and workloads are categorised as short, medium and long jobs. Further jobs are sub-categorised as less, medium and high based on their resource consumption levels. Workload characterisation [42] has been derived based on different metrics including CPU/memory usage, throughput, response time etc., for the purpose of modelling the relationship among a set of workloads. Characterising workloads on individual VMs can lead into a pattern across a set of VMs by aggregating loads on the VMs, and this analysis is intended to determine a minimal set of VMs that can handle the workloads.

The intensity of the workload levels on VMs [43] has been characterised by exploring the cross-VM workload correlations resulting from the dependencies among the workloads across a set of VMs. Though temporal correlations have been validated at the operating VMs by capturing groups of VMs exhibiting repeatable patterns, modelling the behaviours of the workloads at job level is hardly evident. Job-level workload behaviour has been studied by exploring their resource utilisation [17] at the deployed server clusters, and further the machine maintenance events have been studied to characterise machines and workloads based on the Google cluster.

Table 2.1. Workload Characterisation Summary.

Authors	**Year**	**Workload**	**Characterisation**
Tankovic et al.	2015	Business Driven Workloads	Repeating pattern and inter-arrival time of incoming jobs
Shen et al.	2015	Business Critical Workloads	Requested and actual resource usage patterns in terms of CPU, memory and disk and network I/O
Moreno et al.	2013	Google Trace Version 2	User-Task relationship based on submission rate, task length CPU/Memory estimation and utilisation.
Moreno et al.	2014	Google Trace Version 2	Workload diversity in terms of behavioural patterns of users and task variability across different observational periods.
Alam et al.	2015	Google Trace Version 2	Statistical properties of workloads based on resource utilisation levels.
Mahambre et al.	2012	IaaS Cloud	Relationship among a set of workloads based on CPU/memory usage, throughput, response time.
Khan et al.	2012	Business applications production workloads	Workload intensity on VMs based on cross-VM workload correlation
Liu et al.	2012	Google Cluster	Job-level workload resource utilisation behaviour
Garraghan et al.	2013	Google trace logs	Server characteristics based on submission rates and resource utilisation levels

Similar analysis of characterising server resources [20] of server characteristics has been conducted based on the workload submission rates and the resource utilisation levels. Despite the existing works of Cloud based workload characterisation, Cloud workloads are yet to be characterised to accurately model the relationship between user activities and their corresponding workload behaviours at the datacentres for achieving reliable [44] prediction accuracy. A summary of workload characterisations conducted is presented in Table 2.1.

2.6.2 Datacentre Energy Expenditure Analysis

Usually, undesirable energy consumption is observed at various levels within the processes and components in a large-scale datacentre environment. Datacentre is a complex environment, in which the utility power entering the datacentre passes through a number of components including lighting equipment, fire suppression systems, networking components, and IT equipment such as server resources, switches, printers, storage and other service delivery components, cooling equipment such as chillers etc., all of them consume significant proportions of input power. Cooling systems are essential components in datacentres, since most of the input power is converted into heat. In addition to PUE, Data Centre Infrastructure Efficiency (DCiE) is also an important metric used to determine the datacentre energy efficiency, which is computed as the reciprocal of the PUE. Energy analysis can be carried out from various perspectives, this thesis addresses the analytics approaches which involves a posterior analytics of the datacentre footprints such as analysing the resource estimation and utilisation ratios of workloads, running proportions of workloads at the servers to their maximum capacity, proportions of idle resources during job execution etc.

The imprecise knowledge and understanding of the characteristics of both the datacentres and Cloud workloads are primary causes [45-48] for the less than perfect efficiency of existing methodologies of energy efficiency. Improvements can be made if the workload and datacentre behavioural characteristics and corresponding energy implication is quantified for the entire system. To this end, extensive analysis has been conducted in the recent past with the motivation of exploring the undesirable energy consumption within the Cloud server resources.

The impact of failure related energy consumption have been investigated [27, 49] , in which the various termination events and their consequent energy implications have been quantified based on the Google trace logs. Furthermore, the effects of task priority levels upon task termination events have been studied. This analysis insisted that large numbers of task kill and evict events are prevalent in Cloud datacentres, thereby causing undesirable energy

expenditures. The cause for such terminations has also been analysed with the motivation of exhibiting the failure and repair time incurred energy expenditures. Apart from this work, the undesirable energy consumption incurred by task failures and terminations have been extensively studied [50-55] from various perspectives in similar environments such as grids, large-scale MapReduce applications in Cloud environments etc.

The phenomenon of longs tail have been investigated [33] to demonstrate that the presence of smaller proportions of long tails can significantly impact the completion times of tasks. Long tails are the stragglers those significantly delay the completion of the entire tasks, thereby incurring excess energy consumption. The effects of performance interference [56, 57] among the co-located workloads which compete for similar resources from the same physical machine have been studied, and a decision making model have been proposed for selecting the best workload hosts from the pool of heterogeneous server resources. This work is aimed at reducing the energy wastes by allocating appropriate VM placements to the workloads. In general, aggressively consolidating workloads having similar resource intensiveness (CPU or memory) would lead to such performance interference effects.

Energy consumption of different run-time tasks [58] has been investigated by exploring the correlation between energy consumption and computational tasks. The energy-aware analysis of the computational correlations among tasks helps to achieve energy-efficient task placement and optimal resource management. Energy consumption of the sending and receiving network switches [59], and communication components [60] in Cloud datacentres has been studied. These communication components can consume considerable amounts of energy whilst connecting users to the Clouds and transmitting user workloads. The energy consumption of dormant server equipment [61] in Cloud datacentres has been analysed to develop a model for optimising energy consumption of the network components. This work states that individual server resources can achieve the lowest possible energy consumption state without affecting other components working under normal conditions. The effect of virtualisation [62] on overall datacentre energy consumption has been investigated by analysing server energy usage under various hypervisor configurations. All such works demonstrated that there is an increased scope for reducing the excess energy consumption prevailing at the Cloud datacentres.

From state-of-the-art research works on energy analysis, it is clear that there is no special emphasis given to energy consumption incurred by the zombie resources resulting from unutilised idle resources in the Cloud datacentres. Most studies are completely focused on

identifying the cause, effect and implications of task failures on the overall energy consumption. Given that server resources are increasingly being under-utilised in Cloud datacentres, idle resource times contribute a significant proportion of the overall energy consumption. The overall datacentre energy consumption is an accumulation of the energy consumed by various components and their corresponding events. Ignoring the energy consumption of zombie servers leaves a large proportion of undesirable energy expenditures unnoticed, and so a complete datacentre energy consumption profile cannot be obtained. This necessitates the need for an extensive analysis of the presence, cause, and implications of the zombie servers in large-scale datacentres in order to limit the effects of resource idleness upon the overall energy expenditures of the datacentres.

2.6.3 Hardware Based Energy Saving Techniques

The server power dissipation in a datacentre is of two types such as the static power and the dynamic power. Dynamic power is usually the power dissipation proportional to the workloads. Static power is usually the leakage currents [63] involved in the power dissipation. Though the leakage currents are considerably smaller than the dynamic power, they do have a measurable impact [64] on the overall power consumption. Most of the state-of-the-art techniques are focussed only to reduce the dynamic power dissipation of the servers undermining the leakage currents. Designing the processor to reduce the static power during the manufacturing phase is more costly and also this approach is advancing slowly. Dynamic Voltage Frequency Scaling (DVFS) [65, 66] has been the focus of many research works, which is the technique of allowing timely varying frequency and voltage levels of the server in accordance to the incoming workloads. But, the voltage and frequency levels of the hardware are not scalable out of their corresponding predetermined ranges.

Another hardware level energy management in the Cloud datacentres is the switching ON/OFF [67] strategy of the operating servers, targeted at reducing the power consumption of the idle resources. This is an efficient approach in reducing the both the static and dynamic power wastages of the servers, but this involves many practical challenges. Complexities include the resource scalability for the sudden fluctuations in the workload demands. In other words, if the idle servers are turned OFF to save energy when the workload is low, the servers must be efficient enough to wake up instantly once the workload increases. This switching between the server states should be achieved with low server wakeup latencies, which is the time delay that the servers incur during the switching process. Furthermore, frequent switching of the servers often degrades the hardware components and affects their reliability and lifetime. Also, the

server switching process consumes a measurable energy without contributing to the service offering. Thus, inappropriate server switching consumes additional energy and also degrades the QoS, and further risks on-time resource availability.

An effective hardware-level power management can be described as the ability of the service providers in serving more number of customers with limited resources. Generally, the server power cost varies as a function of its operational mode. Maintaining multiple power states of the server such as active, deep sleep, shallow sleep, soft off, mechanical off is effective in both achieving scalability and avoiding frequent switching. In this aspect, the decisions in choosing the right server for switching its state is crucial and choosing the right servers is even complicated in a heterogeneous environment. One approach that drives decision making in this server selection is to dynamically estimate user demands. If the expected future load is estimated a priori, then right servers can be switched states at the right time and in right number. But this prediction involves many practical challenges since inaccurate prediction often lead to misperceptions. The time interval between two consecutive invocations of the placements of the VMs is called as epoch and after each epoch changes are obvious in the resource availability. Associating the resource availability in terms of the server status with the anticipated workload levels is important after each epoch in datacentres.

2.6.4 Resource Allocation Strategies

Another perspective that drives energy management is the resource allocation strategy for the incoming workloads. This includes both the allocation of the incoming workloads and reallocation of the exiting workloads upon necessary. Automation in resource schedulers is key to enhance the both energy efficiency and performance quality. Automatic scheduling decisions on dynamically migrating and consolidating VMs are one of the key challenges prevailing in this kind of resource allocation policies. The scheduling policies should incorporate the global policies [68] with the consideration of multiple clouds rather than local scheduling policies. To this end, distributed scheduling [69, 70] policies are more focussed by Cloud researchers since conventional scheduling policies consider only the processing time delay during the workload allocation. For example, bin packing strategy is a resource allocation policy, in which the incoming workloads are simply allocated onto the available servers, similar to the packing up of an empty bin with materials. In other words, the VMs are the objects to fit into the bins and the physical servers are the bins. The order in which the workloads are allocated onto the hosts affects the efficiency of the resource allocation policies, because of the timely varying nature and demands of the workloads. Other existing allocation algorithms [71-

74] such as VM energy based scheduling, First Fit Decreasing (FFD), Best Fit Decreasing (BFD), and Modified Best Fit Decreasing (MBFD) are efficient only in a homogeneous server environment, and losing grip on heterogeneous environments.

2.6.5 Workload Consolidation

The virtualisation technology has led to the consolidation of the resources, by which, reducing the number of operating physical resources by merging the workloads together to operate in a single underlying physical server. Consolidation of the workloads is affected by various parameters that directly reflect on the overall energy consumption and the performance of the entire system. Generally, applications performs well when they are executed alone than in the consolidated mode. Consolidating similar workloads [75] often consumes more power and also degrades the performance quality since every consolidated workload competes for its own benefit. Also, the interferences between the co-located VMs affect the performance quality. Consolidation process should be given the preference to initially explore which workloads can co-exist well and are mutually exclusive. Aggressive consolidation [76] of the VMs only with the perspective of reducing the running number of operating servers would lead to performance degradation. When the VMs are consolidated [77] together to reduce the operating physical resources, the consolidated VMs share the available resources such as CPU, caches, buses and memory controllers etc. This sharing of the resources usually causes performance overheads which considerably affects the performance quality. And this performance degradation depends on the behaviour of the individual workloads.

It is not suitable to consolidate similar workloads together in a single platform because of their contention to share the same resources. For instance, CPU intensive workloads tend to consume more resources than memory intensive workloads. Here, consolidating CPU intensive workloads together would consume more energy and in turn degrades the performance of the workloads as both the workloads compete for CPU resources of the physical server. Mutual consolidation usually consolidate workloads of different resource requirements together for achieving better performance. If the consolidated number of VMs is larger than the number of available physical cores, it leads to core interference issues and thereby, affects the performance quality. With the emergence of multiple Cloud services, consolidation can also be benefitted by global consolidation strategies [78] across multiple servers rather than local consolidation policies. Another perspective of energy efficient consolidation is to consolidate workloads on to a smaller number of server with higher level utilisation rather than using a larger number of servers with lower utilisation levels.

2.6.6 Live Migration

Principally, there are two perspectives of migration [79] such as the suspend/resume migration and live migration. Live Migration is considered to be the finest approach to migrate the VMs in order to enable energy efficiency in the High Performance Computing environments due to the down time incurred in the suspend/resume migration. The underlying concept [80] behind energy efficient live migration is to migrate the running VMs to an optimum location. Whilst VMs operating in higher utilisation server to lower utilisation server to reduce the workload of the former, VMs operating on lower utilisation servers can also be migrated to switch the server off for energy efficiency. Live migration can highly benefit energy efficient datacentre management by reducing the operating temperature, reducing hot [81] (or red) spots in the servers, enhancing the cooling management, optimum scheduling of the workloads, and by achieving load balancing across the operating servers etc. Also, it is used to enhance the performance quality, in such a way that operating VMs facing resource scarcity can be migrated to an optimum location where additional resources can be availed. The basic principle of the live migration is to first select the VMs to be migrated followed by the determination of the appropriate location for migration, and then to trigger the migration. Live migration involves some critical decisions about *which VMs are to be migrated, when to trigger the migration and where to migrate the selected VMs?* Also, the migration downtime [79] during which the service might be interrupted should be maintained at the minimum level since this parameter has a direct impact on the performance quality. For example, in online [82] gaming applications, the migration process should be handled effectively even without the knowledge of users in order to offer a good service quality.

The pre-copy technology [83, 84] is one the migration techniques which first copies the memory pages from the source to the destination recursively, while running the VM services at the source. After successful copy of the memory pages, the execution of the instances will be shifted to the destination. The prime advantage of this approach is the lower migration downtime. Transferring the entire memory pages might turn out to be complex, particularly with the communication intensive instances requiring more bandwidth. In such cases, the migration process might incur additional communication overheads causing longer migration duration than usual, and thus results in increased migration latency and affects the QoS. Also in the live migration process, the generation of the dirty pages during the execution of the instances cannot be undermined. Migration of these dirty pages is inappropriate and leads to useless migrations. Another approach named the CR/TR motion [85] tends to compress the

memory pages in order to reduce the amount of data to be transmitted. The memory compression phase incurs a margin of QoS degradation and thus sacrificing the performance to an extent. Local migration, which is usually the migration within the same server, involves migrating only the local shared memory, thus eliminating the need for memory compression. But, the local migrations are not appropriate at all times when the resources are inadequate in the single server. Another pre-migration approach is the reservation strategy [83, 86], where the destination is first reserved before the migrating iterative copies of the memory pages. This approach employs a stop-and-copy strategy at the source and a commitment-and-activation strategy at the destination. This requires the creation of the checkpoint on a secondary storage such as Storage Area Network (SAN), and later retrieval of the VM image on the destination server. This way of pre-copy migration is beneficial for the communication intensive instances. But, this pre-copy live migration involves an additional time cost which is determined by the shutdown delay at the source, and migration duration and start up latencies at the destination.

The length of the migration, which is the migration duration is dependent on several factors such as network bandwidth, the size of the migrating VMs, the nature of the execution and so on. Usually, memory intensive instances tend to exhibit longer migration duration and also incur additional migration overheads than the CPU intensive instances. Mostly, the memory intensive migration is benefitted by incorporating a shared memory option such as SAN [87] and Network File System (NFS) in order to reduce both the migration overheads and the migration duration. In such a case, only the execution of the VMs is shifted to the new location, with both the locations referring to the shared memory option. In this sense, the storage options in the Cloud environments are classified as Instance storage and Shared storage. In the former, data is stored in the VM itself, while the data is stored in SAN or NFS in the latter. Thus the key participants of the live migration process are the source server, destination server, the VMs and the storage servers. Sometimes live migration of the VMs might turn out to be inappropriate for various reasons. From an energy efficiency perspective, migrations that do not cause any reductions in the active hosts or does not prevent over utilisation of the resources are counted as Useless Migrations [88]. Such useless migrations are inappropriate to be processed, since they incur both excess expenditures and undesirable performance overheads [45].

Generally, migration processes are handles in a cascaded fashion by the hypervisor. This incurs additional delays in the processing of the instances those are located behind the queue. To this end, migrations of multiple VMs have been introduced, which allows both the parallel

migration and the serial migration of the VMs. Migration mechanisms are handled at various levels such as the thread level, process level and the VM level. Super migration [89] is a technique of migrating groups of instances rather than migration of the instances individually and such migrations can reduce the network cost. A process is said to be at its centre of gravity of its communication load, if no possible migration of that process to any 1-hop neighbour can lead to any reduction in the network cost. Otherwise, the process is considered to be unbalanced. The success of this super migration depends on the identification of the set of instances that can be migrated together as a group. Inappropriate collection of instances often leads to the failure of the super process and this failure has a drastic impact on the entire performance when compared to the failures of individual migrations. Decision should be made upon whether single migration or super migration can potentially benefit both the energy efficiency and the performance quality. In a super migration, if the migration is not feasible because of inadequate resource availability at the destination, then the least beneficial instances are usually pruned in order to make the migration feasible. Though least beneficial, the dropped instances potentially affects the performance quality.

Threshold based migration [90] is another common approach in practice to enable the live migration of the VMs. The migration threshold is a crucial factor in this approach which usually depends on the methodology of computing the threshold. One of them is the server based threshold [91], where the migration is triggered when the operating physical server reaches either the upper threshold or the lower threshold. A VM based threshold approach [6, 92, 93] assigns energy budgets for the individual VMs with predefined thresholds and triggers migration when the pre-defined VM threshold level is breached. Live migrations based on predefined static threshold levels do not scale well in a heterogeneous datacentre environment, the initially assigned threshold levels should be dynamically scaled based on the various runtime factors such as server status, VM conditions, overall resource scaling etc.

2.6.7 Prediction-driven Techniques

Predictive analytics [94-98] are being carried out in Cloud environments for various purposes such as resource scaling, workload allocation, optimising elasticity etc. In general, there are two important phases of prediction analytics [46, 74, 99-101] in Cloud environments for energy efficiency, firstly forecasting the anticipated intensity of the arriving job submissions and secondly estimating the resource consumption levels of the arrived jobs. While the former benefits efficient scaling of the datacentre resources, the later helps with optimum level of resource provision for the incoming jobs. An approach for clustering tasks [102] of similar

characteristics has been proposed to estimate the resource requirements of newly arriving tasks by analysing clusters formed of historical information. This analysis further presents that only 20% of tasks exhibit periodicity, the degree of periodicity is crucial in determining the overall prediction accuracy. Since the incoming job trend exhibits better periodicity than tasks and tasks are actually contained within jobs, a hybridised clustering approach of both jobs and tasks might deliver a better estimation of resource requirements for workload execution. *k-means* clustering [19] is a well-known classification approach used for clustering observations, which divides *n* observations into *k* clusters based on the chosen parameters. The clusters are usually formed around the optimal centroids, but determining the optimal number of clusters is often complex. Another complexity prevailing in adopting *k-means* algorithm for classifying Cloud variables is that the algorithm does not scale well for global clusters and clusters of different size and density. Both these complexities necessitates analysing the characteristics of the incoming workloads before choosing the optimum number of clusters. With the Cloud workloads being increasingly heterogeneous, uniquely analysing the incoming workloads to choose the number of clusters might be tedious, and cluster selection based on qualitative metrics of the workloads generally introduce subjectivity in the computation accuracy.

A linear regression based prediction model (LRM) [103] has been proposed for benefitting autonomous resource scaling in Cloud datacentres, by predicting the number of service requests expected at the next interval based on an observation of linear trend of workloads in a relatively short period of time. Though LRM exhibits a lower prediction error, a simple linear approach may not scale well under fluctuating and dynamic workload arriving pattern in a longer time frame. Forecasting the workloads in a relatively shorter term may not allow enough time-scale for resource management due to the wake-up latencies of the machines. Generally in a simple LRM, the mean of the independent variable is expected as a linear combination of the regression coefficients and the predictor variables, and this mean is strictly linear due to fixed predictor values. This might prevent the regression from accurately modelling the dependency between the dependent and independent variables under dynamic workload fluctuations.

A Pattern matching workload prediction framework [104] for forecasting the resource usage patterns has been proposed by exploiting historical usage patterns those similar to the current trend. This framework identifies similar usage patterns from the past using Knuth-Morris-Pratt (KMP) algorithm for string matching, but jobs with similar characteristics not necessarily exhibit similar resource usage patterns. Furthermore, task failures within a single job execution significantly affect the overall resource usage levels of the workloads and tasks failure rates

vary dynamically during different execution instances. In addition, this algorithm calculates the acceptable error based on the desired number of matches, more matches are usually identified for larger acceptable error. Since the error margin is unique for every prediction match, deciding the trade-off between the desired number of matches and prediction accuracy is always questionable. Enhancing the precision of prediction usually restrain the number of matches identified by the algorithm, thus may not provide suffice historical evidences for estimating the future usage patterns. Addressing the error margin issues of this KMP algorithm, an improved KMP algorithm in combination with a linear regression model [105] has been proposed for load prediction. This model apparently chooses the linear regression model when the workload fluctuation is low, and chooses the KMP model when the incoming trend of workloads exhibits higher fluctuation. This scheme of alternative prediction model may fit the dynamicity of Cloud Computing. Though, it is not guaranteed that the observed current trend stays unaltered during the prediction time, thus the switching scheme may not scale well when the observed trend shifts suddenly in shorter time. A pattern matching scheme [106] for CPU workload sequence forecast has been proposed based on the KMP algorithm. This scheme benefits from a pre-processing phase of data analysis encompassing a time series analysis of the monitored data followed by a Kalman filter to approximate the true data based on observed data. In general, KMP algorithm suffers limitations such that the traditional approach can only match absolute values. Since the incoming traffic is actually sequence of data points, traditional KMP algorithm may not scale well for time series sequential data analysis.

An exponential smoothing (ES) [107] based prediction mechanism has been proposed to predict the future job arrival trend using historical information. ES approach, which predicts the trend of a time series, has been applied in this model to predict the attributes of the future trend by concurrent iterations. Though this approach benefits from the historical trend, limited usage of the historical traces in time may not provide suffice inferences, which necessitates the need of storing historical traces longer incurring additional storage costs. Since Cloud workload behaviours are bound to business hours, contradictions and inaccuracies might arise whilst using the peak time current iterations to predict the off-peak time future trend and vice versa.

A Hidden Markov Model (HMM) based prediction [43] of workload patterns has been proposed by exploring the temporal correlation among the workload behavioural changes, treating workload samples as time series. This model exploits the cross VM correlations resulting from the dependencies among the applications running in different VMs. This prediction framework is aimed at forecasting the workloads on individual VMs based on the

workloads groups witnessed in the previous process cycle. Analysing the workloads at the group level to predict the workload pattern on the individual VMs may not deliver precise prediction results. Also predicting the workload patterns should incorporate the knowledge of user behavioural patterns, since users are the actual drivers of the workloads. But, workload patterns on individual VMs are usually driven by the scheduling and job allocation mechanisms of the service providers not users. An off-line prediction [108, 109] has been conducted based on a pre-recorded resource usage data to forecast short-term resource usages based on a Markov chain model. Markov matrix [110] has been used to predict the future state distribution vector from the current state. In general Markov based approaches work with the fact that estimation of workload patterns exhibit prediction probabilities and the highest probability will lead to an effective prediction result. In general deterministic approaches might deliver better prediction accuracy than probabilistic approaches under the dynamic and timely varying nature of Cloud Computing. Motivated by the Markov matrix representation is a special class of Markov model [111] called the Markov Arrival Process (MAP), which is extensively used to describe the timely varying characteristics of the workloads by stochastically capturing the uncertain future evolutions of the workload features. MAP allows generalisation of the Markov Modulated Poisson Process (MMPP) framework, which is a class of continuous-time Hidden Markov Modelling (HMM) for prediction analysis. The active state of the HMM determines the current arrival rate of the workloads and models the arrival rate by Poisson process. MAP also captures the important workload features such as auto correlation, probability distribution, temporal and spatial parameters etc., and can generate the sample statistics such as Cumulative Distribution Function, joint moments of sample distribution, and auto correlation function coefficient etc. Markov based analysis involves the crucial error margin factor which usually determines the accuracy of the prediction results. If this associated error margin is significantly larger than the prediction level, then there is a higher possibility of wrong prediction results. Also, Markov model requires a significant number of samples for accurate prediction analysis, since a small number of samples are insufficient for capturing the complex dynamic features of the workloads. More number of samples can be generated by the state transitions defined by the weights of the Markov chain, which is required for the prediction analysis. Such a state transition can either be hidden or observable, with both being represented in the Markov matrix. The effectiveness of the prediction accuracy of the Markov model depends on the statistical analysis of the extracted statistical features, and the computational tractability of the MAP model facilitates such effective analysis.

The mean load prediction over a long-term interval has been proposed based on [112, 113] Bayesian model. With an estimated mean load at a given time, this model predicts the mean load into the future time for up to 16 hours. The mean load over consecutive time intervals is estimated an exponentially segmented pattern for the purpose of characterising the host load over a definite period of time. These prediction segments are transformed into a pattern with the heuristics that host load appears with higher correlation among the adjacent short term intervals. This may not necessarily be true in most of the occasions, since Cloud workloads and active users exhibit significant variations between peak and off-peak business hours. Thus long-term prediction based on adjacent segments might not be effective to deliver reliable level of prediction accuracy. Bayesian model [112, 113] works by the way of relating the prior and posterior event probabilities for computing the conditional probabilities. Furthermore, our empirical analysis on evaluating the efficiencies of HMM and Naïve Bayes model in predicting Cloud workloads revealed that both the models are susceptible [114] to an increased error percentage whilst predicting CPU and memory intensive Cloud workloads.

A predictive model based on a degree two polynomial regression [115] has been proposed for benefitting resource scaling in the datacentres. This model estimates the static and dynamic resource requests at the web server tier and the database tier for predicting the optimal configuration required for dynamically varying workloads in order to achieve an optimum provisioning of resources. This prediction model is built using the application performance statistics obtained while the application is still running. In general, polynomial regression models the relationship between the independent and dependent variables based on their non-linear dependence, since the polynomial functions are non-local the value of the independent variable strongly depends on the dependent variable. This prediction model works on real-time based on continuous iterations for estimating the over-provisioning of resources, this necessitates consistent monitoring of the workload intensity and resource consumption profiles which may impose additional overheads in the overall resource provisioning system. Furthermore, this model may suffer from complex time-cost since the overall prediction time is an accumulation of the status response time, process iteration and resource estimation time.

A modified best fit policy [116] has been proposed by treating physical machines as bins and virtual machines as items to be allocated onto the bins for the purpose of minimising the number of active physical servers. Before allocating VMs on to the physical servers, the future load anticipated on the physical machines is computed based on the load on the VMs to be allocated. This algorithm is merely a computation based on the current load rather than a prediction, since

the estimation is based on the known VM load. Estimating the anticipated incoming traffic a priori might help better resource management, rather than computing the anticipated load on the servers since this necessitates an additional computation time before the workloads can be allocated for processing. A virtual resource scheduling prediction scheme [117] has been proposed based on Support Vector Machine (SVM). The virtual resource requirements have been estimated by modelling the non-linear relationship between the inputs of the SVM. This model benefits by reconstructing the phase of the system as a time sequence. Phase is a state of the system at a given time, reconstructing the state as a time sequence might help modelling the linear dependence among the consecutive data values which can lead better prediction. But the degree of dependence among the consecutive values will dominate the prediction, and the efficiency of the SVM usually depends on the associated algorithm in learning the relationship inherent among the data values.

SPAR [118] (Spare Periodic Auto-Regression), an autoregressive based prediction model, has been proposed to forecast the workload patterns with the assumption that the dependent variable is highly correlated at every step under similar time period. Workloads driven by a variety of users co-existing with varied business needs may not satisfy this assumption in a Cloud environment. Users are characterised with unique patterns of job submissions and moreover, a single user might also submit different types of jobs in a single session. Thus modelling the incoming job trend as a simple auto regression within an observation period may not scale well under co-existing users with varied characteristics. RPPS [119], a prediction framework based on simple ARIMA technique, has been proposed to predict the future workload trends. The resource usage pattern has been fed as a time-series into the predictor to predict the workload pattern for a short-term time. ARIMA model is subjected to confidence limit bounds to the actual forecast determining the over and under-estimation errors of the forecast. Since Cloud workloads exhibit extreme dynamism in both the arrival frequency and the number of expected submissions, a simple ARIMA forecast may not deliver a precise forecast of the workload patterns in Cloud environments. The larger variance of the workload pattern resulting from the fluctuating job submission trend of users causes increased residuals in the training data which can significantly affect the prediction accuracy. A second order autoregressive method [120] is deployed to predict the workload patterns in Cloud environments for an effective resource management also suffers similar drawbacks.

Simple Moving Average (SMA) is an arithmetic moving average calculated as an unweighted mean of the previous n observed values, where n is the total number of historical usage profiles.

In the context of resource usage profiles of the historical samples, SMA assigns equal weights to all the historical usage profiles despite their association or timeliness (recent values) with the current sample. Exponential Moving Average (EMA) is a type of an infinite response filter and predicts the next anticipated value as an arithmetic mean of the historical samples by assigning exponentially decreasing weights to the older samples. Low Pass Filer (LPF) subjects the prediction of the previous iteration to a degradation function to predict the next possible value.

2.6.8 Straggler Mitigation Techniques

Straggler identification is growing importance as an integral component in the context energy management of the datacentre resources, since identifying the straggling tasks and treating them accordingly benefits not only to ensure completion of the straggling tasks but also to restrain the stragglers from consuming more server resources. Stragglers are usually identified based on a defined threshold to locate tasks exhibiting an abnormal execution behaviour than those of the other co-located tasks within the same job. One of the most commonly considered execution metrics of task execution is task duration for determining long tail stragglers. In practice, tasks exhibiting a runtime duration increasingly proportional to the average duration of the other co-located tasks are identified as long tail stragglers. Long tail stragglers are usually identified during the runtime by exploiting and comparing the current running duration of all the executing tasks within a given job. Most of the existing works adopts this duration threshold as a typical value of tasks exhibiting 50% longer [121] than the average duration to classify them as stragglers. Static threshold for straggler identification based on their execution duration may not scale well for Cloud workloads because of the increased level of process heterogeneity found among tasks within jobs. Tasks within a single job might exhibit varied duration and resource consumption pattern, so a generalised representation of all tasks within a job can often mislead to derive unreliable inferences about stragglers during runtime. Such static thresholds are commonly computed as a temporal difference of the duration between a running task and the average task duration during execution. Computing this temporal difference is only possible during the run time. Though identifying task-level stragglers before the actual execution might facilitate better management of stragglers, forecasting task-level stragglers before the actual execution is merely an aspiration to date.

Long tails [33, 122] usually result from the poor efficiencies of the nodes processing the corresponding tasks causing node-level stragglers, but the nature and intensity of task themselves can also lead to a straggling behaviour causing task-level stragglers. Task-level

stragglers have not gained suffice importance till now, as most of the existing works in the context of straggler identification focused on the node-level stragglers for long tail mitigation. Existing strategies of mitigating stragglers during runtime include speculative execution, which is commonly being used in Hadoop and Map Reduce environments and in production clusters [121] such as Google and Bing. This strategy increases the probability of early completion of the straggling tasks by creating multiple replicas of the straggling long tails, the replica completed first is stored for task completion and the remaining replicas are then terminated. Though, duplicating the same execution instance demands more resources allocation and hence incur excess energy consumption. All other replicas other than the one completed first are all terminated, so the resources spent on such terminated replicas are wastage of resources. Furthermore, the execution of such replicas are not know a priori which doubts their completion rate in a way that the created replicas might be even execute longer than the actual stragglers.

Another drawback of speculative execution is the definition of the threshold to trigger replicas. If the replicas are created earlier during the execution, there is still a possibility for the actual task to progress smoothly after the creation of the replicas, in which case the created replicas are simply terminated without extracting any information services. If the replicas are created later in the execution, changes are the created replicas may not finish on time causing unnecessary delays in the actual job completion, whereby wasting not only the resources spent on the straggling task but also the created replicas. Most of the existing works identifies runtime stragglers during the later stage of the actual execution's lifecycle, causing needless replicas. Identifying the threshold point for creating replicas is crucial in the success of speculative execution to avoid needless replicas, though such identification is only possible during runtime and hence a pro-active measurement is not possible. Furthermore creating replicas when the system utilisation [121] is higher may pose threat of straggling the created replicas. It is common for a data intensive tasks to progress at a slow phase than the other jobs, creating replicas for such jobs might not benefit quicker completion of tasks and simply create needless replicas to consume undesirable energy and resources without adding any benefit to the current execution.

A progress score based threshold [121] computes task progress score as the ratio of proportions of tasks completed and proportions remaining for execution, and classifies the corresponding task as a straggler when its progress score falls behind a defined threshold than the average progress score of the given job. Hadoop scheduler [123] uses this progress score for straggler classification and classifies tasks as straggler when their progress score falls behind 80% of the

average progress score. Both task progress rate and the process bandwidth within an execution phase [124] have been utilised to identify straggling tasks. The process speed of tasks has been predicted during the runtime and the remaining task execution time has been computed [125] for triggering a new set of speculative execution for maximising the cost performance. When the progress rate of tasks fall behind the other tasks, then such tasks are reported as stragglers. This progress rate has been determined by a slow node threshold typically at a rate when the progress rate of a given task falls behind the other co-located tasks by a margin of less than 50%. Whilst a higher threshold delays the straggler identification point and may not identify any stragglers at all, a lower threshold might increase the number of tasks identified as stragglers. LATE speculates tasks those estimated to finish farther into future to reduce job response time.

Again, the duration is computed based on the progress rate, all such computations are possible only during the runtime restraining the possibility of straggler identification before the actual execution starts. Since tasks within a single job do not progress at stable rate, process bandwidth might not provide suffice inferences for accurate estimation of stragglers. Furthermore, tasks within a single job may not start at the same time in the case of jobs with cascaded tasks, whereby a common progress rate cannot be applied to all tasks within a single job. Straggler identification based on the anticipated progress rate for tasks during execution may not scale well in a heterogeneous environment, in which the progress rate of tasks would be different under different execution instances affected by the bandwidth, throughput, server load, co-located tasks etc. Further evaluating the progress score against the mean of the remaining tasks within a job might introduce both additional overheads and relaxations for straggler identification during runtime.

Straggler tasks are characterised as exhibiting a normalised duration [126] as an increasing multiple of the median of the normalised duration of the other co-located tasks within the same job. Thus, when the normalised duration of a given task is increasingly proportional to the averaged mid-values of the normalised duration of the other co-located tasks, then the corresponding task is characterised as a straggler. The normalised duration is computed as the ratio of task execution time to the amount of work done, which is the completed proportion of job at the time of calculation. Assuming a static median value for forecasting task stragglers might not provide an accurate estimation in a dynamic Cloud environment. Mantri is a straggler mitigation approach [127, 128] focused on conserving the computing resources of the server nodes. It employs a strategy of backing up tasks for multiple execution at an early stage and

kills the original task instance when the cluster becomes busy and restarts task in a different node instance. Though Mantri addresses conserving energy by an early speculation of the straggling tasks, terminations are unavoidable at the process level. Furthermore, the newly created instances of the terminated tasks are not often guaranteed to complete within a defined time-scale, causing long tails of the replicas.

Addressing the issues of speculative execution, a co-worker based scheme [129] has been proposed for mitigating stragglers to shorten job completion time with less resource consumption. This approach transfers a portion of data from the straggling tasks to the co-worker, whereby the workload is shared by the co-worker. Data intensive tasks usually progress at a slow phase and this approach usually computes the progress rate after a few seconds of task initialisation. Whilst addressing energy efficiency, the data transfer cost and migration costs are unavoidable in this scheme. Another way of avoiding runtime stragglers is server blacklisting [130-132], which is the approach of avoiding nodes witnessed as stragglers in the past for task allocation. Though this scheduling strategy helps to avoid node-level stragglers, it is not always true that a straggler node in the past should remain to be a straggler in the future. Usually such poor server nodes are removed, upgraded and repaired and added back on to the server resources. Assuming the server nodes as stragglers from previous instances might cause false negatives, whereby a non-straggling server node will be wrongly categorised as straggler nodes. Most existing work on straggler identification are focused on identifying the stragglers during run time and mitigating them during the actual execution. It is commonly being argued that straggling tasks can only be identified during execution, and not before the start. Furthermore, the state-of-the-art straggler identification techniques are focusing more toward tasks exhibiting an execution time duration longer than the co-located tasks within the same job, leaving the resource utilisation levels of tasks unnoticed.

2.7 Summary

The state-of-the-art techniques focussed on the energy efficient datacentre execution are leaning towards energy conservation with compromising the performance quality to their respective margins. A wide range of research has been carried out to reduce the energy consumption of Cloud resources from various perspectives. A few include the efficient migration of the operating VMs, appropriate allocation of the workloads, enhancing the cooling management of the hardware, task consolidation to reduce the number of operating physical resources, and using techniques like Dynamic Voltage and Frequency Scaling (DVFS) [133]

to dynamically alter the voltage levels and frequencies of the servers. To this end, these techniques are witnessed as being either at the hardware level or at the software level. Software level approaches are mainly based on virtualisation and task consolidation. The DVFS and the cooling techniques [134] are dependent on the operating hardware. The underlying concept behind the hardware based approach is efficient server management. This server management is mostly achieved either by DVFS technique or the switching ON/OFF strategies of the physical servers. Most of the hardware based approaches help only in reducing the energy consumption of the physical resources and undermine system performance. A combined approach at both the hardware and software tier in a datacentre environment can be a potential solution for achieving both energy efficiency and performance optimisation.

Predicting the future behaviours of users in terms of their workloads submission and resource requirements can drive the hardware-based server management at the datacentres for energy efficient computing without degrading the service quality. But the efficiency of this approach directly rely on the level of achievable prediction accuracy for effective decision making in the server farm management. But the dynamic nature of both the workload and user characteristics impose various levels of challenges in developing an effective prediction model further to benefit energy management in Cloud environments. It is commonly evident that most of the existing prediction techniques utilise historical data for estimating the future trend. However, the correlation between the historical samples and current observations have not been effectively validated, choosing the most appropriate historical samples is very important in determining the prediction accuracy.

From the state-of-the-art techniques focused on straggler identification and mitigation, it is clear that an approach capable of predicting straggling tasks before the start of the actual execution has never been addressed. Furthermore, node operation status, particularly CPU usage rate of task execution have not gained enough importance in straggler identification. As task duration is crucial in determining long tails, CPU usage rate can cause task terminations leading to reprocessing the terminated tasks, thereby causing the terminated tasks to be potential stragglers. Terminated tasks lead to more energy implications than those of long tails causing stragglers and also affects the completion of the actual job. Whilst reducing the proportion of long tails within a single job shortens job duration, reducing the number of task resubmissions directly reduces energy expenditures.

This chapter has detailed the concepts behind Cloud Computing and the implications involved in achieving energy efficient datacentre execution. The importance and the need for an in-depth analysis of Cloud workload and user behaviours have been highlighted for the purpose of facilitating a more precise understanding of the Cloud systems, ultimately to explore the possibilities of predicting future behaviours of Cloud workloads in terms of their arrival trend, resource consumption trend and straggling behaviours at the datacentres. Enhancing the accuracy and reliability in predicting the anticipated future behaviour of Cloud workloads would highly benefit energy efficient management of Cloud datacentre resources.

3 Cloud Workload and Datacentre Analytics

3.1 Outline

This chapter includes a detailed description of the explored Cloud workload datasets and further presents a comprehensive analysis of the characteristics of Cloud workloads, exhibiting the impacts of latency sensitivity levels of Cloud workloads upon datacentre energy consumption. The undesirable energy expenditures spent on unutilised idle server resources have been empirically exhibited. Further, the hidden periodicity inherent among Cloud workloads and user behaviours have been uncovered to investigate the predictability of Cloud workload and user behaviours.

3.2 Dataset Overview

This research work deeply explores the Cloud trace logs [135] released by Google, featuring more than 650000 jobs submitted by users, spanning across 28 days of datacentre execution. The trace logs are investigated in order to explore and observe the patterns and behaviours of jobs, tasks and machines at the Cloud datacentre, along with observing the patterns of user resource requests, resource provision and resource consumption profiles of tasks during task execution. This analysis is intended to extract information pertinent to energy consumption of the server resources and the inherent periodicity in Cloud workload and user behaviours for driving prediction analytics. The event statistics observed from the trace log analysis are presented in Table 1.

3.3 Datacentre Events

A complete user workload can be witnessed as a job, and tasks are usually the integral components of jobs. In other words, a single job may contain several number of tasks, and the completion of a job can be assured only when all of its encompassing tasks are executed. Jobs

Table 3.1. Trace Log Statistics

Number of Days	28
Total Number of Job Submissions	650892
Total Number of Task Submissions	46093201
Number of Operating Servers	12500
Average Number of User per Day	190

may include either cascaded or parallel tasks or both. Jobs arriving at the datacentres submitted by users are usually scheduled on to the servers for processing. All tasks in a single job are scheduled accordingly onto the Cloud servers and allocated with resource levels in terms of CPU cores, memory and disk spaces. This allocated resource levels are usually determined based on the resource requirements of users and the computational intensity of jobs. The scheduler in the Cloud datacentre schedules jobs based on the resource availability of the servers and the priority levels of tasks.

The scheduled jobs are usually executed and finished smoothly when provided with the required level of server resources and generally incur no intrinsic and extrinsic termination causes. But the actual execution of tasks could face various types of terminations resulting in execution failures, such terminated jobs are most often resubmitted and rescheduled for processing. About 40.52% of the scheduled jobs are actually being killed [41] at least once in a lifetime before their successful execution. The frequency of jobs being killed is actually higher than those fail. Thus majority of the CPU cycles are wasted as they are put into jobs being killed and thus hugely account for excess energy consumption. However, these observations about the Kill event is quite unusual as the killed job percentage is relatively high. In general, killing job quite often from users incur undesirable costs and also this is unpleasant for the providers as well. Jobs are usually killed when the execution does not go in the desired direction or usually the test jobs are killed. In this sense, the higher percentage of kill events are attributed to the fact that the analysed trace logs are extracted from an exploratory testing architecture, which is a testing framework that explores interactions between distributed and concurrently-executing components. EVICT is a temporary termination event where a workload execution is paused and are generally resumed later or rescheduled to higher more efficient machines. Tasks are evicted whenever the current execution faces any resource constraints such as resource scarcity or if the current execution exceeds the pre-allocated resource limits.

It is common to evict tasks with low priority levels to accommodate tasks with higher level of scheduling priority. LOST event occurs when a job is terminated suddenly due to other events at the Cloud datacentres such as service outages, unexpected machine failures, shutdown etc. In regard to these events occurring at the Cloud datacentres, the current status of jobs and tasks [136] are classified as un-submitted, pending, running, and dead. This research analyses the various datacentre events at both job and task levels, detailed analysis of various datacentre events conducted on the studied trace logs are presented in the following sections.

3.3.1 Job Level Event Analysis

Figure. 3.1 presents the event analysis of the Cloud trace logs at job-level across the time period of 28 days, comprising a total of 650892 jobs submissions. As illustrated in Table 3.2 and Figure. 3.1, 57.64% of the total submitted jobs are processed smoothly without any interruptions or failures. Among the submitted jobs, users and providers have killed 41.03% of jobs and 1.31% of jobs have failed due to failures at the server level. Both the killed and failed jobs account for excess energy consumptions and such jobs are most often resubmitted again. The number of evicted and lost jobs is insignificant compared to those failed and killed events.

3.3.2 Task Level Event Analysis

Figure 3.2 presents task submission events along with the various types of termination events at task level for the observed period of 28 days, comprising a total of 46093201 encompassed within 650892 jobs. In general, providers often prefer to terminate tasks of a given job rather than the entire job during resource constraints. Even though this approach results in process delays, jobs involving task terminations are not necessarily considered as an entire failure.

Table 3.2. Job Level Event Statistics

Job Event	Number of Jobs
Submit	650892
Finish	375151
Kill	267079
Fail	8523
Evicts	22
Lost	16

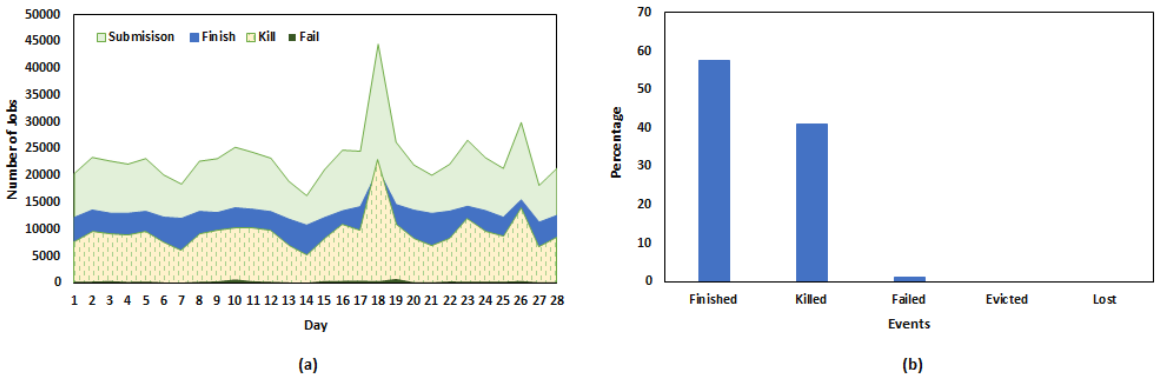


Figure 3.1. Job Events (a) Day Wise Analysis (b) Overall Percentage

Terminating entire job results in resubmission of the entire tasks contained within the corresponding job, which not only incurs additional process delays but also consumes excess energy and server resources. It is common that a few tasks contained within jobs act as stragglers and may face multiple terminations due to their process complexities. This event in the datacentre delays the completion of the entire job until the straggling tasks are executed successfully. Jobs analysed in the trace logs comprise only parallel tasks, thus tasks within jobs are independent to each other and do not impact each other.

As illustrated in Table 3.3 and Figure 3.2, an increased number of task failures can be witnessed at 29.34% of the submitted tasks. Further, the analysis illustrate 20.87% kills, 12.54% evicts and 0.02% lost events at task level respectively. These statistics demonstrate the existence of significant variations between the behaviours of Cloud workloads at job-level and at task-level accordingly. The kill events at task level are almost half of those at job level. This is postulated to the fact that users tend to kill their entire jobs rather than individual task. The rest of the kill events at task level usually results from the providers killing the straggling tasks. The number of tasks failed are significantly higher than those job level failures, insisting that tasks often face increased levels of natural terminations resulting in resubmissions. Further, with the number of evict events at job level being insignificant, analysis illustrates a considerable

Table 3.3. Task Level Event Statistics

Job Event	Number of Tasks
Submit	46093201
Finish	17176153
Kill	13525951
Fail	9619554
Evicts	5779568
Lost	8754

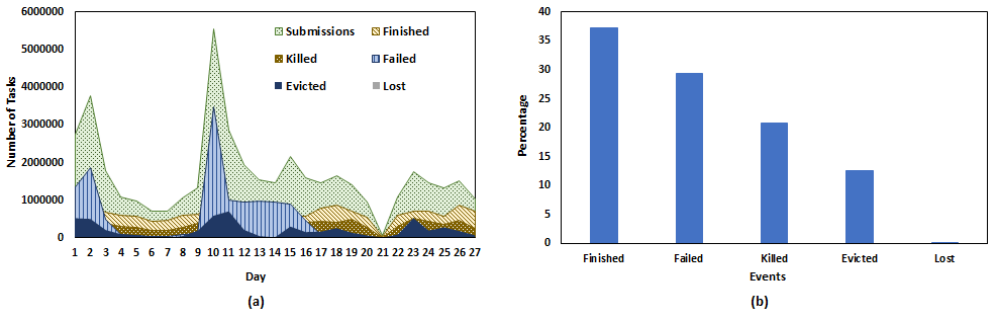


Figure 3.2. Task Events (a) Day Wise Analysis (b) Overall Percentage

number of evict and lost events at task level execution. Thus tasks are often evicted during resource constraints, and then resumed when the required resources become available ensuring the successful completion at job level. Lost events stays insignificant at both job and task level execution.

3.3.3 Machine Event Analysis

In addition to the datacentre execution events, Cloud server farm usually undergo various types of events pertinent to the availability status of the machines. It is natural in a Cloud processing environment that individual machines may become unavailable due to hardware failures, and server down-times during both hardware and software upgrades. About 2% [14] of the machines are constantly upgraded and about 2.5% of machines are removed and added back every day in a Cloud processing environment as they consistently report necessary updates. Such machines are usually taken offline to apply the necessary updates for the purpose of enhancing their process capabilities and to confront failures. Most of these machine downtimes are shorter and tasks being processed in such machines are stopped and rescheduled to other available machines. Some of the machines may face more frequent downtime than others, which could be attributed to their capacity, aging, etc.

Figure 3.3 illustrate the day-wise analysis of machine removals and machine updates during the observed period of 28 days respectively. On average, around 311 machines are removed due to failures and around 255 machines are updated every day at the studied datacentre. In some cases, users submitted jobs may characterise specific server requirements such as the server architecture, operating systems etc., and the requirements of such job types can only be satisfied by the specified architecture types. Beyond the architecture, tasks may also be governed by one or more constraints such as scheduling, resource availability, compatibility issues, core count, network capacity, platform etc. Thus the availability status of the machines in the server farm is crucial in timely scheduling and allocating the incoming workloads.

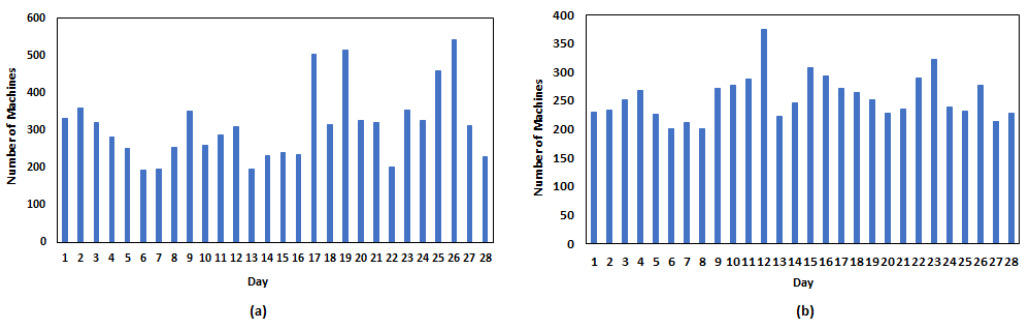


Figure 3.3. Machine Event Statistics (a) Machine Removals (b) Machine Upgrades

3.4 Workload Latency Sensitivity

Latency plays an important role at various levels of workload processing in a Cloud processing infrastructure. The most dominating types of latencies are the network latency and the dispatching latency, both of which actually result from the geographical distribution of users and the Cloud datacentres. Both these latencies depend on the Round Trip Time (RTT) [137], which defines the time interval between user requests and the arrival of the corresponding return response. Another type of latency existing in the process architecture is the computational latency which is the intra-cloud latency [138] found among the processing VMs located within the same datacentre. This latency depends on both the software and the hardware components [139, 140] such as CPU architecture, run-time environment, and memory, guests and host operating system, instruction set, hypervisor used etc. CPU architecture, Operating System and the scheduling mechanisms are the most dominating factors of this type of in-house computational latency. Jobs and tasks submitted at the Cloud datacentres characterise various levels of latency sensitivity levels depending on the nature of their process requirements and the end-user QoS expectations. A single definition of the computational latency cannot fit all types of jobs, since every job should be uniquely viewed at the datacentres. For instance, processing a massive scientific workload may span across several days or months, in which delays of a few seconds are usually acceptable. Common example of the latency sensitive workloads is the World Wide Web, among which different applications have different latency levels. The acceptable level of latencies usually depend on the measure of the end user tolerances. Workloads resulting from users surfing the internet are generally latency-insensitive. Jobs including online gaming, and stock exchange data etc., are the commonly witnessed latency sensitive applications. The level of sensitivity is determined by the allowed time-scale for the providers to provide an undisrupted execution of the workloads for delivering the desired levels of QoS, ranging from a few microseconds to a few tens of microsecond end-to-end latencies.

The taxonomy of the latency levels of Cloud workloads studied in this research are attributed from level 0 representing the least latency sensitive jobs through to level 3 representing the most latency sensitive jobs, which are provided explicitly in the analysed Google trace logs. Least latency sensitive jobs (level 0) are non-production tasks [11] such as development and non-business critical analyses etc., which do not have a significant impact on the QoS even if these jobs are queued at the backend servers. Level 1 jobs are the next level of latency sensitive

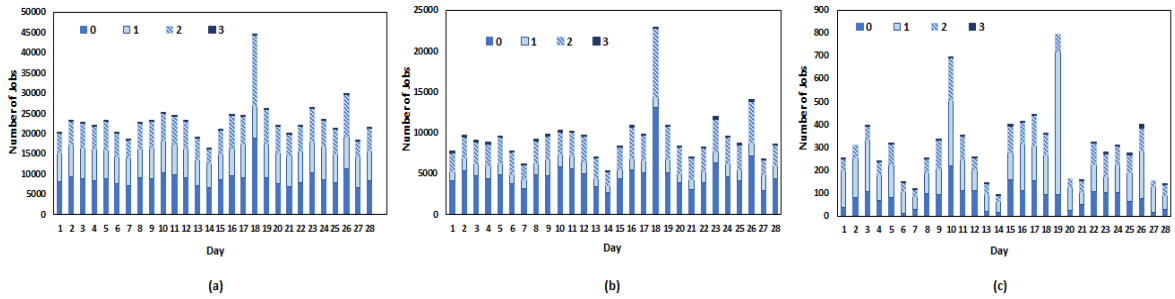


Figure 3.4. Latency Wise Job Events (a) Submission (b) Kill (b) Fail

jobs and are generally the machine interactive workloads. Level 2 jobs are the real time machine interactive workloads and the latency tolerance levels of these jobs stays at the tens of milliseconds. Level 3 jobs are the most latency sensitive jobs with latency tolerance levels at the range of sub-milliseconds, and are generally the revenue generating user requests such as stock and financial analysis etc. Workloads characterising increased level of latency sensitivity levels are treated with higher scheduling priorities at the datacentres. Latency analysis has a prime importance in greening the datacentre, since every job submitted to the Cloud datacentre has its own level of latency tolerances which has a direct impact on not only the various workload behaviours at the datacentres but also the end user QoS satisfaction. In general, least latency sensitive workloads are addressed to be less energy consuming and are relatively shorter in duration than jobs characterising higher levels of latency sensitivity. The failure probabilities of the latency insensitive workloads are also lower and are not resource hungry, thus exhibiting fair energy utilisation profiles. A detailed analysis of the various types of latency sensitivity levels and their corresponding impacts on the various datacentre events at both job-level and task-level are presented in the following sections.

3.4.1 Job Level Latency Sensitivity Analysis

Figure 3.4 presents the latency-wise Job-level events at the studied datacentre including job submission, job kill and job failures respectively. It can be clearly observed that most of jobs submitted to the Cloud datacentre are least latency sensitive accounting for 38.51% of the total job submissions, followed by level 1, level 2 and level 3 at 31.73%, 29.12% and 0.63% respectively. Interestingly, most latency sensitive job (level 3) submissions are insignificant supporting the fact that Cloud-based workloads computationally less latency sensitive as opposed to scientific grid workloads [18] which are usually resource intensive.

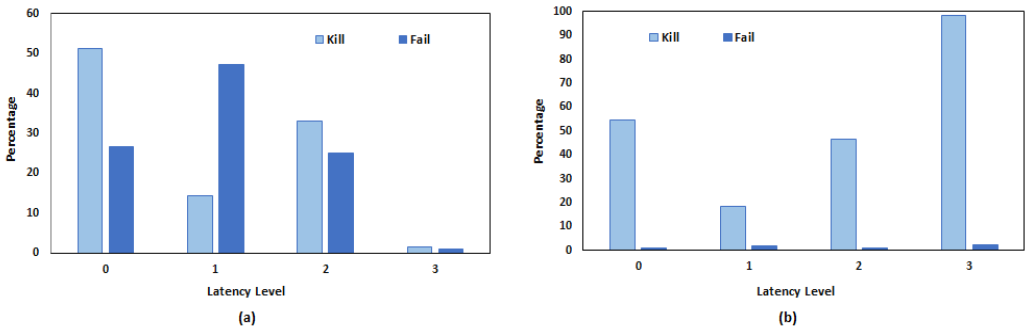


Figure 3.5. Job Termination Events (a) Against Overall Submission (b) Against Respective Latency Levels

Figure 3.5 illustrates the percentage statistics of the overall kill and fail events for different latency levels against the overall submission and against their respective latency level submissions accordingly. It can be observed that least latency sensitive jobs (level 0) are being killed the most, followed by level 2, level 1 and level 0 respectively. Level 1 latency sensitive jobs are failing the most naturally at the servers, followed by level 0, level 2 and level 3 respectively.

On a finer level analysis within each latency levels, most latency sensitive jobs (level 3) are most vulnerable both to be killed by users and to be failed at the servers, 98.21% of the submitted jobs have been killed, and 2.3% have failed. Least latency sensitive level 0 jobs account for 54.57% kills and 0.91% fails respectively. Following Level 0, Level 2 jobs account for 46.53% kills and 1.12% fails and level 1 jobs account for 18.43% kills and 1.95% fails respectively. It is evident that level 1 jobs are less vulnerable both to be killed and to be failed than any other latency sensitivity levels.

Though insignificant in number, most latency sensitive jobs are more vulnerable to consume excess energy witnessing the higher number of kills and fails. But their overall impact should considerably be lesser than the other lower latency sensitivity jobs, since they are submitted in a very few number accounting only for 0.63% of the total job submissions. The number of jobs being killed and failed shown in Figure 3.5 include jobs those are resubmitted following a kill or a failure event. Almost all jobs of latency level 3 are vulnerable both to be killed and to fail at least once before their successful completion. All these kill, fail and resubmission events account for excess energy consumptions to a considerable margin when compared to the actual kill-free and failure-free completion of jobs.

3.4.2 Task Level Latency Sensitivity Analysis

Figure 3.6 presents the day-wise statistics of latency-wise task submissions, along with the various termination events affected by different latency sensitivity levels accordingly. Similar to job-level statistics, most of task submissions are least latency sensitive accounting for 79.52% of the total number of task submissions, followed by level 1, level 2 and level 3 at 12.46%, 7.54% and 0.47% respectively. Further Figure 3.7 presents the percentage statistics of the termination events against the overall task submission and against their respective latency level submissions respectively. It is evident that least latency sensitive tasks are the most vulnerable workloads for all types of termination events. A significant number of failures of level 0 tasks are evident at 86.66%, followed by 10.22%, 2.50% and 0.62% fail events for level 1, level 2 and level 3 respectively, as shown in Figure 3.7(a). Further, an increased number of kill events of level 0 tasks are evident at 64.14%, followed by 28.05% kills for level 2, and 6.83% and 0.98% kills for level 1 and level 3 tasks respectively. Furthermore, 74.98% of evicts are identified among level 0 tasks, followed by level 1, level 2 and level 3 respectively at 19.19%, 5.11%, and 0.72%. This is because the providers naturally tend to terminate the least latency sensitive tasks whenever they face resource constraints, since the least latency sensitive task terminations do not have a worse impact on the QoS in comparison to evicting tasks characterising increased levels of latency sensitivity. But this privilege of terminating the least latency sensitive tasks should not exceed beyond the level of user tolerances, which then would affect the QoS.

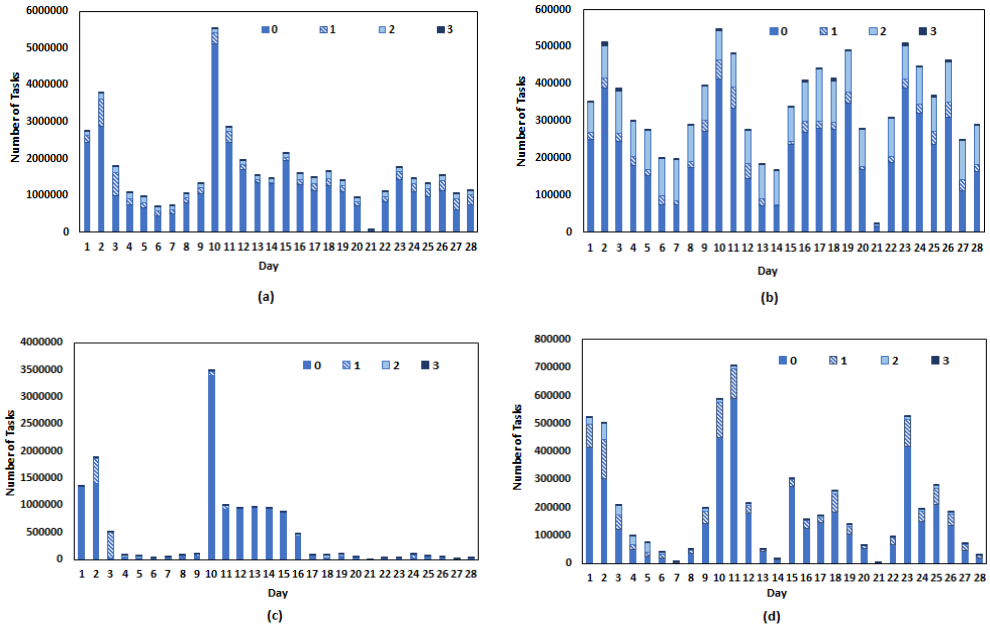


Figure 3.6. Latency Wise Task Events (a) Submission (b) Kill (c) Fail (d) Evict

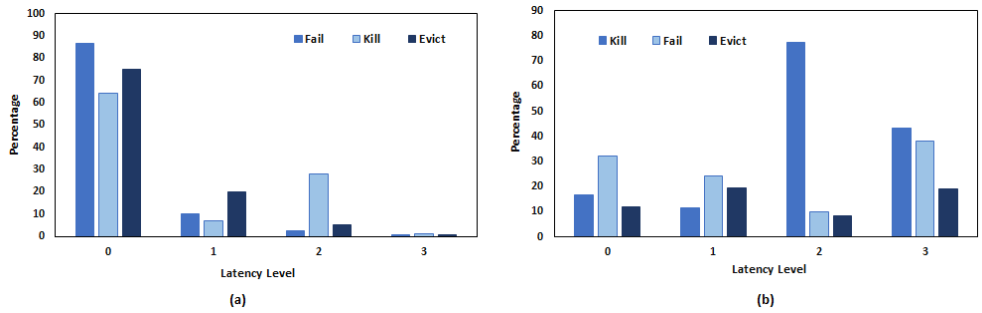


Figure 3.7. Task Termination Events (a) Against Overall Submission (b) Against Respective Latency Levels

On a finer level analysis within each latency level submissions, level 2 tasks encounter the most number of kill events at 77.61%, 9.75% failures and 8.50% evicts respectively, as shown in Figure 3.7(b). This implies that only a very few number of level 2 tasks are processed smoothly without facing constraints. Level 3 tasks face the most number of failures at 38.30%, 43.15% kills and 18.98% evicts respectively. These statistics demonstrate that level 3 tasks face an increased number of resource constraints such as unavailability, outages, or other unexpected terminations. Similar to level 2 tasks, only a few level 3 tasks are processed without facing resource constraints. Further, the analysis depict 16.83% kills, 31.98% fail events, and 11.82% evict events for level 0 tasks, and 11.44% kills, 24.06% failures and 19.31% evicts for level 1 tasks respectively. Thus, around 40% of level 1 and 45% of level 2 tasks are processed smoothly without any constraints, and thus account for much lesser resubmission events. The bottom line is, with more number of tasks facing various types of terminations at level 2 and level 3, around 95% of level 2 and almost all the level 3 tasks face terminations at least once before their successful execution, and thus lead to an increased number of resubmissions and incur additional undesirable energy expenditures.

3.5 Datacentre Undesirable Energy Expenditures

3.5.1 Zombie Servers

In the Cloud Computing service model, client demand requirements are satisfied by provisioning resources, this is typically at a level which far exceeds [102] the actual amounts of resource necessary to process their prospective job. Providers provision the resource levels for task execution based on the intensity of user demands. Since the provisioned resource levels are not often completely utilised, this directly causes the physical servers to operate below their actual capacities. This results in an increased proportion of server resources being idle and as a result of which, approximately 46% of machines [43] are operating below their maximum capability for a significant amount of time with their resources being reserved for the over-

estimated resource requirements. These resources remain idle and unutilised, consuming unnecessary energy and without contributing to the execution of the workloads. Servers with increased proportions of idle resources are termed as ‘comatose’ or ‘zombie’ [141] servers, which are usually powered on and consuming electricity but delivering no useful information services. Unfortunately zombie servers can remain unnoticed in datacentres since they do not have service affecting failures, their utilisation metrics do not generate exceedance alarms and they are therefore unlikely to appear in any top reporting lists. It has been shown that an idle server may consume approximately two-thirds of the energy [3, 62, 142, 143] used by the same server operating at full load. It is worthy of note that the power overhead incurred by resource idleness vary significantly for different hypervisors on alternative physical servers. Exposing the cause and impacts of zombie resources in Cloud datacentres is an integral requirement of the Cloud service model not only to achieve energy efficiency but also to maintain optimised performance quality. The cause, presence and impacts of Zombie servers on the datacentre energy efficiency are presented in the following sub-sections.

3.5.2 Workload Sampling

The analysed trace log data has been sampled on a per day basis with a single day spanning across 24 hours starting from 12.00 am for a given day for the purpose of characterising the zombie server energy consumption. The Cloud trace logs provide explicit information [36] of task events including start and end time, job ID, user, task priority levels, and resource requests in terms of CPU, Memory and Disk space, and the corresponding amounts of resources consumed during every execution session. Task usage information of the trace logs required for the zombie server analysis includes mean CPU usage rate, assigned CPU usage, assigned memory and maximum memory usage for every single task execution session. With the assigned and used amounts of resources being explicitly provided, the unutilised resources is computed. The elapsed time period of a task execution is calculated from the start and end time of an execution session. The elapsed time period are not necessarily the completion time of tasks, since a task execution session may be successful or might have faced a termination resulting in resubmission of the terminated task. Thus the elapsed time calculated is the time period when the status of the corresponding tasks is updated with either completion or termination. However, providers allocate resources for all task execution sessions, and resubmissions of tasks will cause the providers to allocate resources again for the terminated tasks.

3.5.3 Profiling Resource Utilisation

Resource utilisation levels has two measurement viewpoints. First is job level resource utilisation which is usually the measure of CPU, memory resources consumed by the workloads whilst executed in the VMs. Secondly, machine level utilisation is the measure of ratio of the actual usage level of a machine to its maximum usage capacity. An accumulation of job level utilisation of all the VMs directly reflects the machine level utilisation of the corresponding physical server. The resource utilisation for task execution sessions are characterised in terms of the CPU and memory resource consumption at task level. With the duration of task execution session being calculated for every task, the resource utilisation profiles of tasks are enumerated to exhibit the proportional idle resources over the actual level of resource allocation for every task execution session. Before quantifying the resource idleness, it is necessary to compute the total amount of resource time actually allocated for a task execution session, from which the idle resource times can be measured. The unutilised memory resources in a given task execution session is directly measured by subtracting the maximum memory usage from the assigned memory. The maximum CPU usage level assigned for an execution session is the maximum allowed level of CPU resources for that task execution. This maximum resource usage level is the highest peak of resource usage level in a given session and the mean resource usage is the mean value of the remaining usage level. As the maximum to mean resource usage ratio $r_{i(ratio)}$ increases, the presence of idle resource time in an execution session shows a corresponding increase. An execution session is measured as the duration between the start and finish time of a given task by either completion or termination. Conversely, lower values of $r_{i(ratio)}$ correspond to higher values of PUE of the servers. Now for profiling the CPU resource utilisation levels, the total amount of resource time consumed and the presence of idle resource time in a task execution session is calculated using equations 3.1 to 32.

$$t_r = \sum r_a \,.\, t \tag{3.1}$$

$$r_{i(ratio)} = \frac{(r_a - r_m)}{r_a} * 100 \tag{3.2}$$

where, t_r is the total resource time allocated for a task execution session, r_a is the maximum level of utilised resources, r_m is the mean resource usage level, $r_{i(ratio)}$ is the proportional idle resource time and t is the duration of task execution session. Due to data ambiguity in the allocated resource level for an execution session, the maximum level of utilised resources r_a is

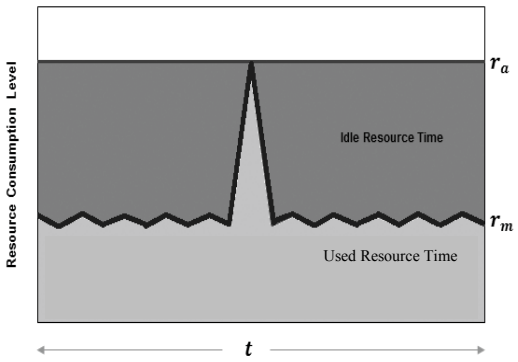


Figure 3.8. Idle Resource Time Proportion

assumed to be the allocated resource level in the analysis. The proportional presence of the idle resource time in a task execution session is illustrated in Figure 3.8.

3.5.4 Power Model

The energy consumption of the zombie servers incurred by the idle resource times is computed based on the unutilised amounts of resources in every task execution session. Such unutilised resource capacity might result from very low utilisation rate, in which case, the unutilised resources are referred as resulting from over-provisioned resources. Another factor causing the end of the execution session is the termination of tasks, where all the allocated resources lead to undesirable resource wastages. All the unutilised resources consume energy without contributing towards the actual task execution and are the primary sources of zombie servers. Potentially unrelated to the demand for energy efficient computing, Cloud service providers are also in the process of enhancing the energy profiles of their operating servers. The energy profiles of the servers developed by Oracle Corporation and IBM [144] are used in this analysis for the purpose of measuring the energy consumption of the zombie resources in order to match the profiles of datacentres in practice today. It is notable that the energy consumption of the

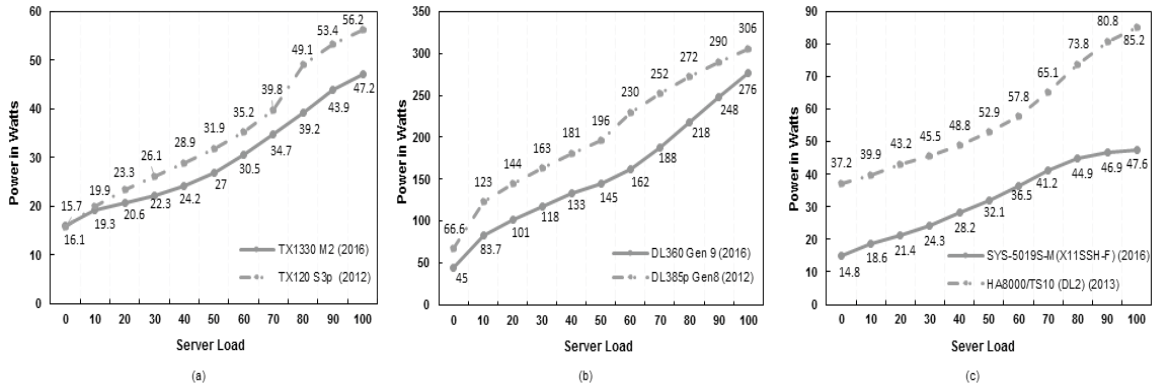


Figure 3.9. Power Profile Comparison of Servers (a) server A (b) server B (c) server C

Table 3.4. Server Capacity Comparison

Plat-form Type	2016					2012-2013				
	Server	**CPU**	**CPU Capacity (ssjops)**	**Total Memory (GB)**	**Max Temp (100% load)**	**Server**	**CPU**	**CPU Capacity (ssjops)**	**Total Memory (GB)**	**Max Temp (100% load)**
A	TX1330 M2	Intel Xeon E3-1240L v5	484,122	16	22.6	TX120 S3p	AMD Opteron 6380	1,660,274	8	20.6
B	DL360 Gen 9	Intel Xeon E3-1265V2	420,255	64	21.4	DL385 p Gen8	Intel E3-1260L v5	528,348	64	20.4
C	SYS-5019S-M(X11SSH-F)	Intel Xeon E5-2699 v3	3,159,419	16	21.2	HA8000/TS10 (DL2)	Intel Xeon E3-1280 v2	495,083	8	23.1

selected servers has reduced since 2012 despite their increasing capacities of CPU and memory resources. The improving energy efficiency of the server power profiles over the recent years is depicted in Figure 3.9, comparing the energy profiles of the current server profiles used in 2016 with those used in 2012. The server profiles for this comparative analysis have been selected based on a close match with their CPU type and capacity in order to compare their power consumption trend, as shown in Table 3.4.

The utilisation levels of the CPU is measured in terms of Server Side Java operations (ssjops) and the power consumption is depicted in Watts. Though the trace logs are obtained from Google servers, this analysis has been conducted based on three different in-practice server profiles to match the current energy consumption trend of the server resources according to the SpecPower 2008 benchmark [144]. The active idle power consumption of the servers is required to maintain the server turned on and the power consumption of the server increases with increasing percentage of the server load. A task execution will consume energy equivalent to the product of the respective proportional power of the current server load and the duration of task execution session. When there is an increase in the server load, the power consumption of the servers will increase with a decrease in the amounts of idle resources. Thus the total power consumed during a task execution session is computed using equation 3.3.

$$P(r) = \sum_{i=1}^{t} P(r_s) \tag{3.3}$$

where, $P(r)$ is the total power consumption during a task execution session and $P(r_s)$ is the server power proportional to current load and t is the total number of execution instances for

the corresponding task. Now, the presence of the proportional idle resource time (zombieness) and its corresponding power consumption $P(r_i)$, can be computed using equation 3.4 and 3.5.

$$t_{ri} = t * r_{i(ratio)} \tag{3.4}$$

$$P(r_i) = P(r_{ai}) * t_{ri} \tag{3.5}$$

where, t_{ri} is the proportional idle resource time resulting from the unutilised server capacities and $P(r_{ai})$ is the server power on active idle. The total amount of idle resource proportion for every task execution session is measured, further to compute the power consumed by the presence of idle resource time on a per day basis over the observed period of 28 days.

3.5.5 Idle Resource Time Analysis

This section analyses the presence of idle resource time among the allocated CPU and memory resources during task execution causing server zombieness, which can in essence, accommodate more workloads than they are currently processing. The idle CPU and memory resource times compared with the actual allocated resource time is illustrated in Figure 3.10. Figure 3.11 depicts the presence of idle CPU and Memory resource percentages over the observed period of 28 days.

The usage measurement sessions in the trace logs include the usage measurements for periods when no process belonging to task execution was running in task's container. For this reason, the values presented in Figure. 3.11 have been normalised by obtaining a mean value of the proportional resource time for every measurement period and normalise it against the overall duration for a given day in order to enhance the computation accuracy. It can be observed that an average of 75.6% of the resource execution time have been wasted caused by the idle CPU cores. This suggests that the CPU resources are commonly being over-provisioned, leaving many of the CPU cores active without any actual contribution towards task execution. The

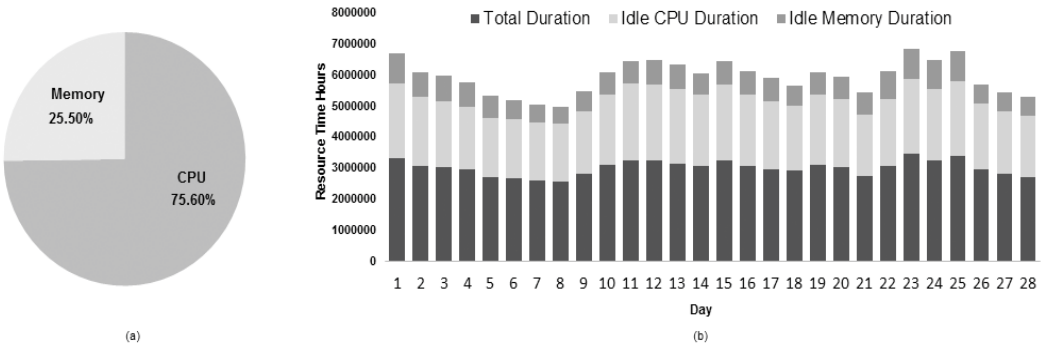


Figure 3.10. Idle Resource Time Analysis (a) Idle Resource Proportion (b) Day-wise Resource Time

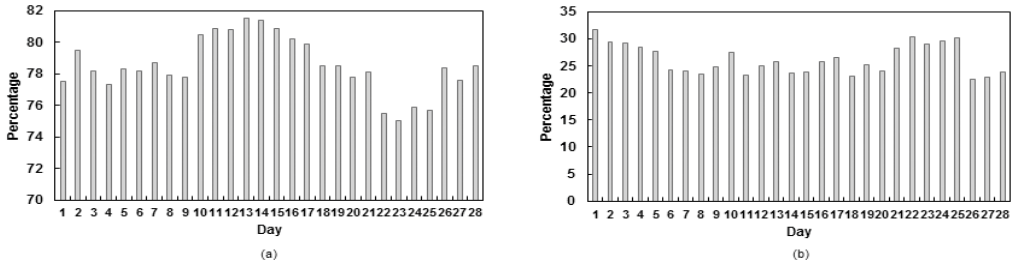


Figure 3.11. Day-wise Idle Resource Time Percentage (a) CPU (b) Memory

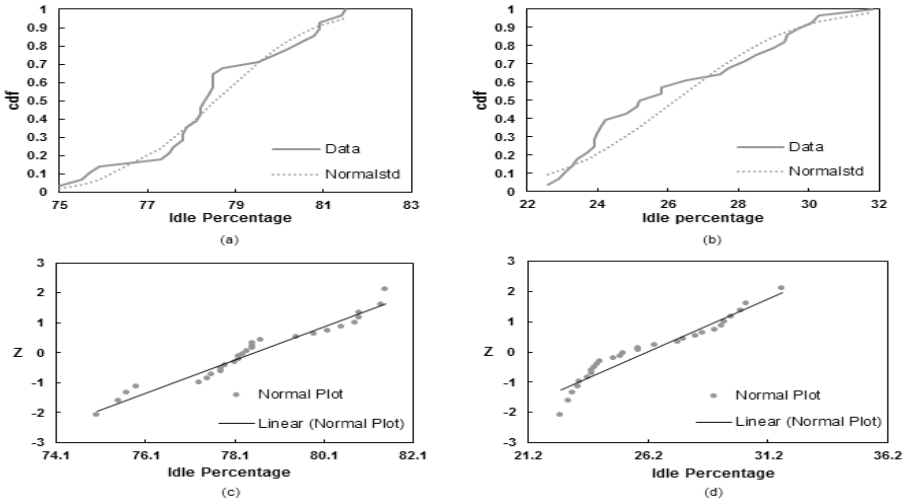


Figure 3.12. Idle Resource Time Distribution Analysis (a) Idle CPU Percent (b) Idle Memory Percent (c) Idle CPU AD Test (d) Idle Memory AD Test

immediate implication of the presence of excessive idle CPU time is that the server capability is always under-utilised resulting from the over-provisioned resource levels. For maximum greening in datacentres, server CPU resources should be allocated with a task load corresponding to their peak processing capability. This strategy will reduce the server zombieness and help the providers to achieve a PUE factor closer to 1. Furthermore, an average of 25.50% of idleness can be witnessed among the allocated memory resources. Though the wasted memory resources are considerably less than those of CPU, memory resources are still being over-provisioned. The actual CDF fitted with a normal standard distribution of idle resource time percentage for both CPU and memory resource is depicted in Figure. 3.12, along with the probability plot of the Anderson Darling (AD) Goodness of Fit (GoF) test. It is evident that the CDF distribution of both the CPU and the memory idle resource times are showing measurable fluctuation from that of the normal standard distribution. This supports the observations of day-wise loose correlation and increased fluctuations among both the CPU and memory idle resource times. The Anderson Darling test for data normality is repeated for both the CPU and Memory idle resource time distribution on a daily basis. The test shows that both

CPU and Memory idle presence follows a near to normal distribution, with the CPU distribution negatively skewed at -0.108 and the memory distribution positively skewed at 0.434.

3.5.5.1 Resource Time Fluctuation Analysis

From Figure 3.12, there is no correlation evident between the idle memory and CPU resource times, suggesting that the CPU and memory consumption levels of tasks are independent to each other, though this may be dependent on the nature of tasks. Since CPU resources are often the largest power consumer in the purely digital system, the increased presence of idle CPU resources will incur a more significant energy consumption. Further, the provision and consumption of CPU cores exhibits increased fluctuations. This increased fluctuation in the usage of CPU cores over the memory resources is illustrated using the standard deviation function of their respective idle resource times, shown in Figure 3.13. This fluctuation is measured as the function of deviation evident among the presence of idle resource time among the co-located execution session within every single day.

From Figure 3.13, the average standard deviation of the idle CPU and memory resource times are evident at 22.9 and 35.2, respectively, supporting the observation that CPU resource times has a closer correlation than memory resource times among the execution sessions within a single day. This would infer that the idle CPU resource time has a better trend of predictability than the memory counterpart. However, within the idle resource time behaviours of the entire month, there are abrupt spikes in the idle CPU resource time at uneven intervals, while the idle memory resource time has an almost saturated curve. Day 8, 14, 22 and 23 are examples of abrupt spikes in idle CPU resource times, their memory counterparts do not show any notable fluctuations. There is no temporal correlation evident between the idle resource time duration

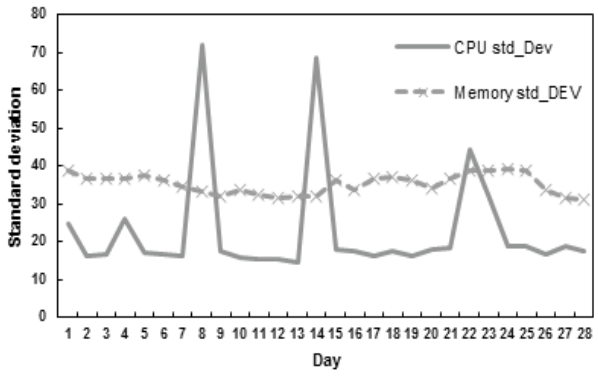


Figure 3.13. Idle Resource Time Fluctuation

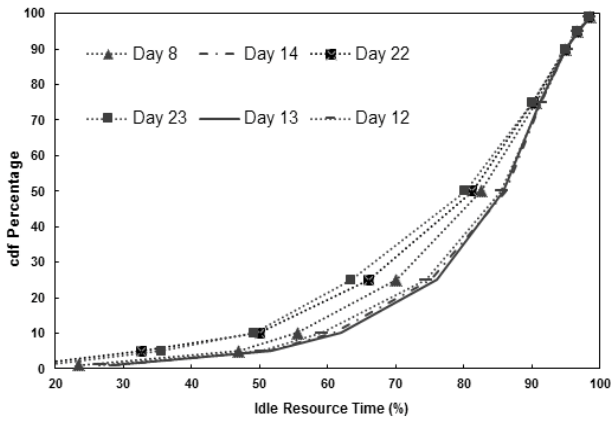


Figure 3.14. CPU Idle Time CDF Distribution

Table 3.5. CPU Idleness Statistics

Day	**Skewness**	**Median**	**Standard Deviation**
Day 13	-2.1519	85.87	14.5
Day 12	-4.4658	85.57	15.18
Day 23	-3416.4	75.03	32.28
Day 22	-492.85	75.48	44.12
Day 14	-5730.2	81.35	68.83
Day 8	-680.39	82.49	72.12

and task submissions. Furthermore, on a day-wise observation, the behaviour of CPU cores show unpredictable trend of resource consumption among the co-located execution sessions which is in contrast to the trend of the memory resources. For this reason, days exhibiting abnormal fluctuations of idle CPU resources among the co-located task execution session within the same day are subjected to further investigation. Days showing high resource time fluctuation (Day 8, 14, 22, 23) have been chosen along with days characterised by minimum fluctuations (Day 13, 12) in order to observe their corresponding behavioural trend of CPU idleness. Figure 3.14 illustrates the CDF of the idle CPU resource percentage and Table 3.5 presents the observed statistics for the selected days.

It is evident that the CDF curves of all the observed days are negatively skewed, which is significant for the days showing increased fluctuations among the co-located sessions of the same day. The curve skewness is insignificant for the Day 12 and 13, insisting that the distribution curve is deviating away from normal with increasing fluctuation in the idle resource time among the co-located execution session. The median is observed to be between

75 to 85 percent for all of the days of interest, insisting the fact that at least half of the allocated CPU resources suffer 75% of minimal idleness. Since task submissions are loosely correlated with resource idleness, user requests is not observed to be affecting the presence of proportional idle resources to a considerable margin.

3.5.6 Power Consumption Analysis

This section presents the analysis of the power consumption incurred by the presence of zombie server resources presented above. Such undesirable power consumption will increase the non-computing or overhead energy, which is an undeniable wastage of input energy. The power consumed by idle CPU and memory resources has been computed based on their respective proportions of idleness during task execution based on the server capacities as shown in Table 3.4. Based on the compute capacities of the servers, the server platforms A and C are classified as mid-range servers and platform B is classified as a high-spec server. From the trace log analysis, the idle resource times presented in section 3.5.5 are measured across a total number of 12,500 active servers, therefore 12,500 active servers are considered across the datacentre, for profiling the power consumption of the zombie resources for an entire datacentre. The measured power consumptions of the idle CPU and memory resources are presented for the three server profiles in Figure 3. 15 accordingly.

The average power consumption incurred by the idle CPU resource time across the observed period are 1.461, 4084 and 1343 (presented in kW-hours) respectively for platform A, B and C. It can be notable that the power consumption levels of the servers are correspondingly higher with increasing process capacity. An interesting observation is that the CPU compute capacity of server platform B is around 6 times higher than the other two servers, but the idle CPU power consumption of server profile B is measured at only around 3 times more than profile A

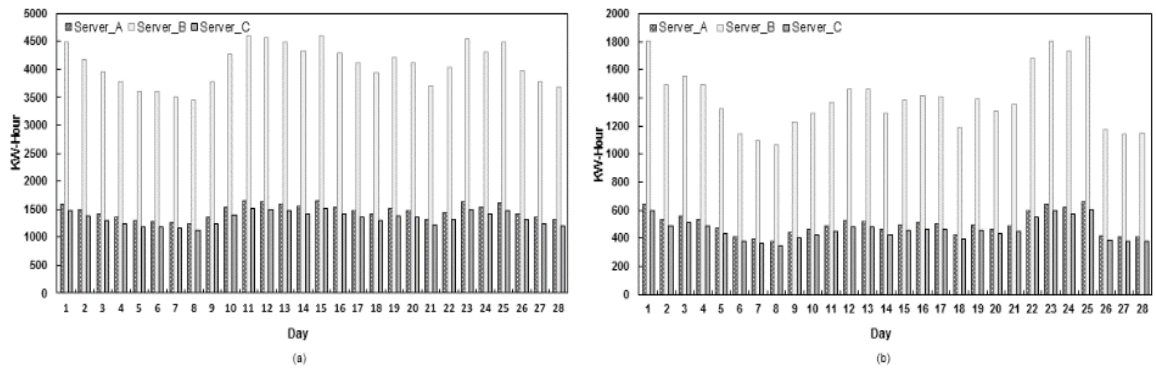


Figure 3.15. Power Consumption of Zombie Resources (a) Idle CPU (b) Idle memory

and C. This suggests that the power consumption levels of the active CPU resources are better optimised with increasing server capability, thereby proportionally reducing the power consumed. The idle resource time is the time during which the resource is either not being realised or is under-utilised, though the resource is still active and immediately available to use (not switched to any lower power "sleep" state). Such resource time could be effectively utilised by either increasing the intensity of task computation or increasing the server task load. Though both actions would increase the power consumption of the servers, their computation capacities can be effectively realised and an optimum utilisation of the available resources can be achieved.

The characteristics of idle memory resource time are viewed from a different perspectives to that of CPU, since the memory usage would most often exhibit less minimal fluctuation within a single execution session than those of CPU usage. The power consumption levels of the idle memory resource time are measured at 499, 1394 and 458 for server platform A, B and C respectively. It can be observed that the power consumption trend of memory resources is similar to that of CPU, with both exhibiting the trend of proportional power reduction of the active resources with increasing server capabilities. While the memory capacity of server platform B is 4 times larger than A and C, its idle memory resources are incurring a power consumption which is only around 3 times higher than A and C. Further, the power consumption levels of the idle memory resources are observed to be much lower than that of idle CPU due to the fact that the idle memory resource times are only one third of the idle CPU resource times. The input power to the servers also incurs power supply losses from AC/DC and DC/DC conversion, with AC/DC losses being much higher [145] than DC/DC losses. Besides the PSU, CPU and memory resources, the internal power consumption of the servers also includes power consumed by fans, drives, PCI cards, chip set etc.

3.5.7 Environmental Impacts

The power consumption of zombie resources has a direct environmental impactions through the means of their CO2 emissions. While the CO2 emissions created along the path of the power consumption cannot be completely avoided, there is seemingly always scope for reducing the level of emission. Some statistics insist that datacentres [146] can reduce CO2 emissions by an incredible 88% or more. This section presents the environmental implications of the zombie servers for investigating the potential for reducing the intensities of the datacentre CO2 emissions by the way of effectively reducing the presence of idle resources.

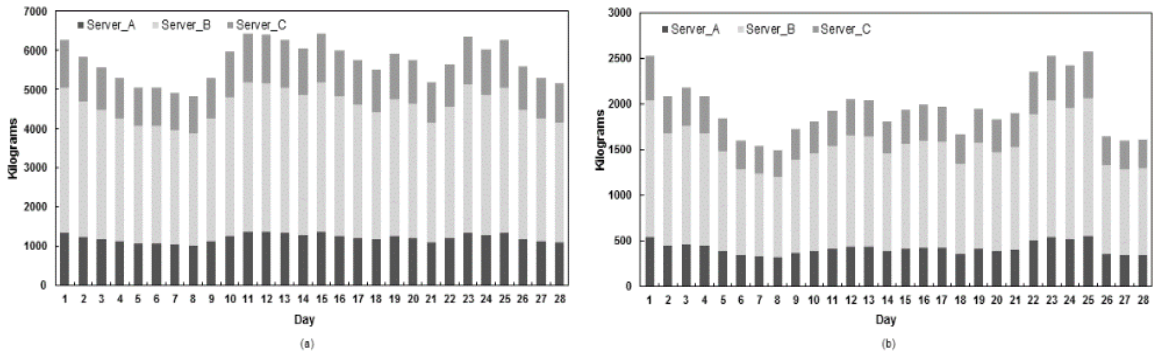


Figure 3.16. CO2 emissions of zombie resources (a) idle CPU (b) idle memory

Figure 3.16 presents the amounts of CO2 emissions incurred by the power consumed (kW-hours) by the idle CPU and memory resources time for the three server profiles under consideration. The CO2 emissions are measured according to the statistics [147] of the U.S. Energy Information Administration. The average CO2 emissions incurred by the power consumption of the idle CPU resource times are 1213, 3390 and 1115 (presented in kg) for server platforms A, B and C respectively. The average CO2 emissions of the memory counterpart are 414, 1157 and 381 respectively for the three servers. Findings presented in the earlier sections mean that it is no surprise that idle CPU resource time has a significantly higher level of environmental implication than those of idle memory resource times. The manufacturer data also confirms that high-spec servers consume more power than mid-range servers, thereby resulting in increased amounts of C02 emissions. The statistics presented in Figure 3.16 confirm that datacentres are one of the contributors toward global pollution, with their CO2 emissions are known to be in the range of thousands of tonnes per annum. The CO2 emissions are usually measured based on the actual method of power generation; the worst case of emission occurs when the electricity is generated with conventional coal combustion. It has been reported that there is a possibility of zero CO2 emissions [148] for the electricity being generated from renewable energy sources such as nuclear and hydro-sources, especially if the emissions related to plant construction, use and maintenance are neglected.

3.5.8 Application of the Analysis

Although the analysis presented in this section is based on the publicly available Google Cloud trace logs, the scope of the applicability of this analysis has been extended by using the trace logs as a baseline for profiling the energy characteristics of various commercial servers being widely used in 2016. The analysis presented in this section provides insightful observations and inferences for the target audience and for researchers promoting green computing. This

analysis can also be applied in similar computing environments such as transparent computing, grid computing and other parallel and distributed systems. The analysis presented in this section will find applications in the following ways.

a) To achieve an effective PUE. The PUE figure is usually the calculated ratio of the total facility power to the IT equipment power. By reducing the presence of zombie servers, Cloud providers can ensure there is no substantial overhead or non-compute energy incurred for achieving an effective PUE by transforming most of the power consumption into useful compute energy.
b) To achieve optimum resource management. Cloud providers are intent on maximising profits by minimising production costs. Spotting server zombieness and allocating tasks accordingly can help to reduce the presence of low-level server utilisations. Issues such as red spots can be avoided in the datacentre, these usually incur excess energy consumption resulting from the increased operating temperature levels of individual servers.
c) Reducing under-utilised resources and provisioning optimum level of resources within individual job execution without affecting job requirements can consolidate more number of workloads onto a minimal number of physical servers to achieve maximum utilisation.
d) Prediction analysis. The characterisation of user requests can be used to predict their resource requirements. The prediction of anticipated workloads in the near future would help providers with not only achieving energy-efficient resource scaling but also ensuring the availability of the requested resources to achieve effective QoS.

3.5.9 Summary

This section of this chapter presents extensive analysis of the energy related implications of the zombie servers caused by the presence of idle resource times whilst executing user requested tasks in large-scale Cloud environments based on the power profiles of current servers used in large-scale datacentres. Three such server profiles have been used as examples to determine power consumption incurred by the presence of idle CPU and memory resource times. Empirical analysis demonstrates that CPU resources account for the highest proportion of idle resource time at 75.6%, with memory resources exhibiting 25.5% of idleness, and the environmental implications of idle resource time in terms of CO2 emissions are highlighted. Idle resource time usually results from not fully realising the complete compute potentials of IT resources. The important inferences of the analysis include the following.

a) Idle resource time results from over-provisioned resources. The proportion of idle resource time is high in every individual task execution session. Over-estimated resources cause the providers to over-provision the server resources, which in turn increases the proportions of resource idleness.
b) Power consumption levels are proportional to the server capabilities. It has been observed that the higher-spec server consumed 3 times more power than the mid-range servers for the same level of idle resource time given the fact that their CPU compute capacities are 6 times higher than those of the mid-range servers. In some cases, the higher-spec server offers more effective consolidation of workloads than the mid-range server. As a result, resource idleness can be greatly reduced when the datacentre composition includes several higher-spec servers capable of effectively consolidating the workloads on to minimum number servers which are operating at near maximum utilisation levels.
c) CPU behaviour varies more abruptly than memory resource behaviour. Abrupt spikes are evident in the presence of idle resource times for CPU resources on a day-wise analysis, with the memory counter-part being almost saturated over the period of observation. Predicting the future workloads is a potential way of achieving optimum resource scaling by reducing idle resource times. However, while it offers the greatest benefit, predicting CPU idleness is more complex than predicting the memory idleness trend.

3.6 Periodicity Analysis

3.6.1 Cloud Periodicity Patterns

Periodicity in this thesis reflects the repeated submission of a given job. This repeated submission could be a recurring submission of a given job in which case job submission in just another submission of the same job. Or the repeated submission could arrive as a result of a resubmission of a failed job. Both the new submission of a previously submitted job and the resubmission of a failed job are expected to provide useful inferences about job behaviours and requirements. This notion of periodicity has been later exploited in this work for predicting the future trends of Cloud jobs at the datacentre.

Cloud providers usually maintain a track of their job execution profiles which include a generalised representation of users and their corresponding resource request and consumption patterns. In general, periodical pattern [42] is evident among job arrival frequency for user activities those associated with the operating business hours. Thus, Cloud workloads exhibit temporal and spatial correlations among them as they are driven by repeatable business

behaviours, and such workloads [149] are consistently predictable. For instance, creating weather reports is a typical example of timely recurring job submission. In this sense, future behaviours of the workloads could be related and extracted [86] from their past behaviours.

Pattern recognition among the workloads and users during a given session in a datacentre environment helps characterising their periodicity for various business functionalities. For instance, threshold pattern can be used to categorise the operating VMs according to their resource utilisation profiles, which helps scheduling and allocating jobs accordingly. Image similarity pattern studies the similarities among the VMs which help storing identical images together, thereby reducing the storage space of the VMs. Relationship pattern determines the degree of association between any two parameters and helps to identify the correlation among the workloads which helps grouping similar workloads together for efficient processing. Variability pattern studies the covariance among both workload and user groups. Periodicity pattern identifies the recurring behaviours among both the VMs and the workloads, which helps effective prediction analysis. Such pattern identification can be benefitted by extracting the statistical properties of job and user profiles which remain consistent for a significant amount of time.

Such a Cloud periodicity can be defined in relation with various time-bound periodical effects [35, 43, 143, 150] such as time-of-the-day, day-of-the-week, week-of-the-month, and month-of-the-year effects. Time-of-the-day effect defines the correlation of user behaviours with different business hours of a day. Such correlations are usually evident across user-driven job events at the Cloud datacentres. For instance, Cloud datacentres might face an increased number of job submissions during the peak business hours and a declining number of job submissions during off-peak business hours. Similarly, day-of-the-week effect is the day-wise correlated user behaviours where job event correlations are evident across the representative days of different weeks. Periodicity pattern among jobs arriving at the datacentres is evident across the submission time, submission frequency, arrival volume, resource requests and number of users who are actually submitting job for processing. Analysing such recurring patterns among job submissions assist modelling the workload behaviours in close correlation with user behaviours at the datacentres.

Categorising and characterising [48] users and Cloud jobs is crucial for driving effective decision making in Cloud environments. Validating the relationships [1] between user behavioural patterns and their corresponding job submissions facilitate better understanding of

user and job dynamism, which would benefit Cloud providers with necessary knowledge for achieving effective resource management, resource provision, workload scheduling and performance optimisation for energy efficiency without degrading the QoS. But modelling the behaviours of Cloud workloads in close correlation with user behaviours in spite of their heterogeneity is laborious, since a computational model cannot identically reflect the human behaviours. The dynamic nature of Cloud workloads demands an extensive and continuous analysis for characterising the workload and user behaviours in a datacentre environment. This section further presents the analysis of both the workload and user periodicity inherent among the studied datacentre trace logs with the motivation of demystifying the dynamism of Cloud Computing, by exhibiting the behaviours of users and their corresponding job submissions. The analysis presented in this section is conducted with the baseline of the periodicity characteristics of both users and jobs arriving at the datacentres.

3.6.2 Job Profile

Jobs submitted at the Cloud datacentres are usually assigned with a random string of Job ID. In general, a single job [15, 136] encompasses one or more tasks which are scheduled for processing at the datacentre. Every job submitted will be assigned with unique Job ID, furthermore a job resubmitted following a termination event will be assigned with a different Job ID to its original Job ID. Tasks encompassed within a single job will characterise the same Job ID, but are assigned with unique tasks IDs, and are usually scheduled either onto a single server or across a range of several servers based on the computational requirements of jobs. Different executions of the same Job will usually have a common logical job name. For instance, even though a resubmitted job will have a different Job ID to that of its actual submission, both the submissions will have the same logical job name, through which different execution instances of the same job can be identified in the analysed trace logs. Job Profile submitted by users at the datacentres is a composite consisting of Submission time t_s, user (name) n_u submitting job and the logical job name n_j, as shown in equation 3.6. Task profile is a composite consisting of submission time t_s , logical job name n_j, user name n_u and the corresponding resource request $r_{(c,m)}$, as shown in equation 3.7. The belongingness of a task to a given job can be determined through the logical job name and user name, both are explicit in the analysed trace logs.

$$J = \{t_s , n_u , n_j\} \tag{3.6}$$

$$T = \{t_s , n_j, n_u , r_{(c,m)}\} \tag{3.7}$$

3.6.3 Job Arrival Periodicity

Figure 3.17(a) presents the total number of job submissions across the observed period of 28 days, with the sample starting on Monday. On a coarse grain, it can be observed that the number of job submissions are reducing significantly over the weekends compared to those during weekdays. Further, a close correlation in the number of jobs submitted on representative days-of-the-week is evident over the entire four weeks. Day 18 is an exception Thursday, which consists of almost double the amounts of jobs submissions in comparison to the rest of the days. Further delving into the reason for this increased number of job submissions on Day 18, 81.21% of jobs have been submitted by a single user. This event can be categorised as a rare event, where a rare user suddenly submits enormous amounts of job unexpectedly. Neglecting all such once-in-a-lifetime workloads from this rare user, Day 18 can be regarded as a usual Day. Though a rare event, this event is a serious concern for the providers since the number of jobs arrived are significantly higher than the usual average, accounting for sudden resource scaling. This event can cause a disastrous impact up on the normal execution of other regular jobs at the datacentre, if the provider is not prepared for this mammoth number of job submissions. In the case of the arriving jobs being latency sensitive, job requirements do not often allow enough time to turn on the server resources characterising longer wake-up latencies. Most often, such type of workloads are efficiently managed by the providers upon providing advance notice of resource requirements, which allows the providers to scale additional resources as necessary.

Figure 3.17(b) presents the most number of submissions of a single job within a single day, which stays at 360 submissions per day on average over the period of 28 days. The submission of this job stays consistent over the entire days of observation, insisting that this particular job is characterised by a recurring submission. Despite this recurring submission, there are few abrupt spikes in Days 11, 19, 20 and 21. This is postulated to the unexpected job failures causing considerable resubmissions of that particular job during the corresponding days.

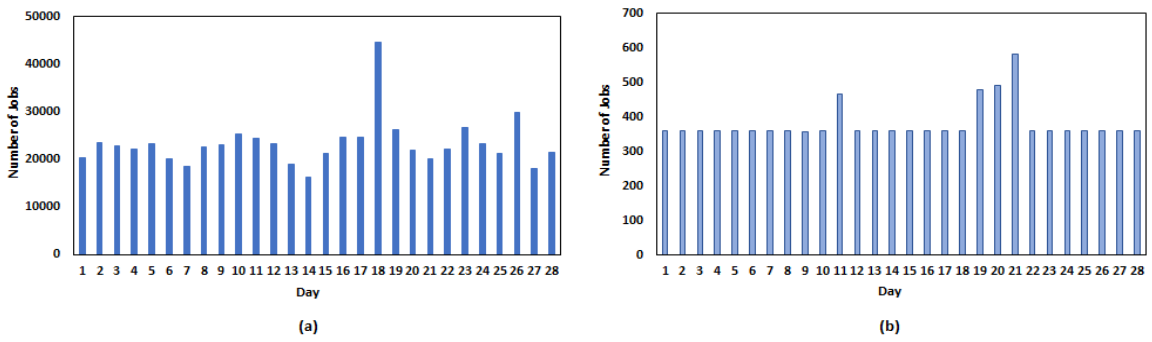


Figure 3.17. Job Arrival Trend (a) Submission (b) Most Number of Submission of a Single Job

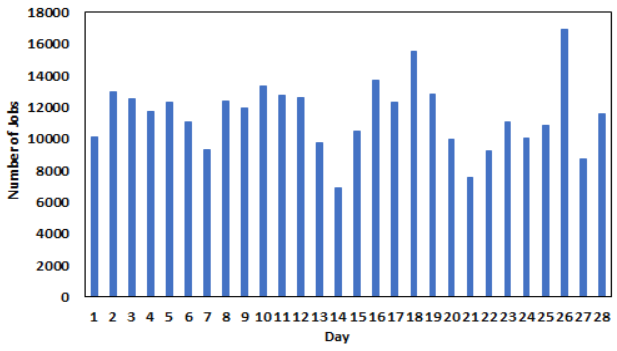


Figure 3.18. Job Diversity

3.6.3.1 Job Diversity

Figure 3.18 presents the total number of heterogeneous jobs submitted to the Cloud datacentre over the observed 28 days. For instance, Day 1 (Monday) comprises a total number of 20409 jobs submissions as shown in Figure 3.17(a), this total number of 20409 submissions are composed with 10100 number of heterogeneous jobs in Day 1, as shown in Figure 3.18. The number of arriving heterogeneous jobs over the observed period is exhibiting a similar behaviour with the number of actual job submissions, characterising weekend declines and increasing number of submissions over the weekdays. Weekend declines and weekday increase in both the number of arrived jobs and the number of diverse jobs reflects the fact that job arrival events are closely correlated with the business behaviours. A very few abrupt spikes (Example Day 26) are evident in Figure 3.18, insisting an increased diversity among the arrived jobs in those corresponding days. A close correlation can be observed between the number of job submissions and the total number of diverse jobs for the corresponding days, from Figure 3.17 and Figure 3.18. Thus, increasing diversity among the arriving jobs can be of the provider's expectation while witnessing an increase in the number of actual job submissions. It is worthy of note that despite the arrival of new jobs, execution failures and job terminations may cause resubmissions of the same jobs resulting in increasing job submissions. This

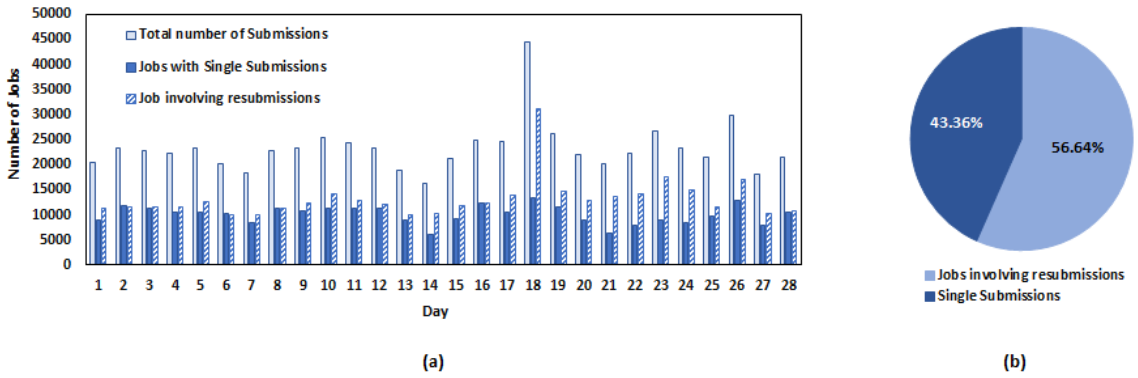


Figure 3.19. Submission Trend by Quantity (a) Day-wise Quantification (b) Overall Percentage

necessitates to analyse the resubmission trend of jobs resulting both from recurring submissions and job terminations.

A finer level analysis has been conducted to exhibit the submission trend in terms of the actual number of job submission and job diversity, resulting from both recurring and resubmissions at the Cloud datacentres. Figure 3.19(a) depict the number of jobs submitted at the datacentre involving single submissions and resubmissions, against the total number of actual submissions by quantity, and 3.19(b) depicts the overall submission percentage for 28 days. Figure 3.20(a) depicts the statistics of jobs involving single and resubmission by job diversity, and 3.20(b) presents the overall percentage statistics. From Figure 3.19 and Figure 3.20, it can be observed that a total of 56.64% of the total jobs have been submitted once and 43.36% of jobs characterise resubmissions. In terms of job diversity, 88.05% of the total job types have been submitted once and only 11.95% of the total heterogeneous jobs characterise multiple submission. In other words, only 11.95% of jobs types have resulted in 43.36% of the total job submissions. Considering Day 1, with a total number 20409 job submissions comprising 10100 heterogeneous jobs, the number of job types submitted once is 9057 and the number of job types involving resubmissions is 1043. In other words, 1043 heterogeneous jobs have resulted in 11352 submissions in Day 1, which is about 55.62% of the total job submissions in Day 1. This illustrates the fact that majority of job types are actually submitted once and a small proportion of the total distinctive jobs are resulting in majority of the resubmissions.

From these observations, it is clear that only a minority of job proportions are involving resubmissions either due to the recurring submissions or due to resubmissions resulting from job terminations. Such one-to-many submission trend of jobs can be exploited to achieve effective resource optimisation by the way of subjecting job submission trend to the periodical effects. Since only a small proportions of jobs are involving nearly half the resubmission events at the datacentre, charactering the periodical behaviours of such jobs allow the providers to

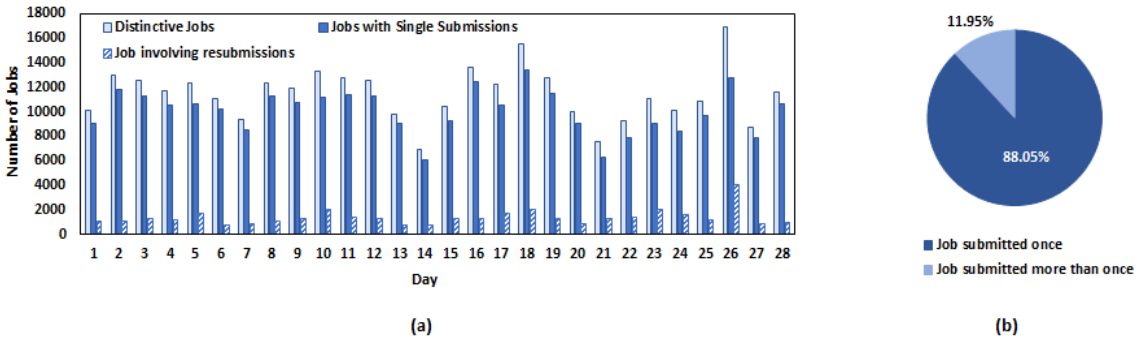


Figure 3.20. Submission Trend by Diversity (a) Day-wise Quantification (b) Overall Percentage

scale and manage their server resources in an energy-efficient way, in accordance with the periodicity exhibited by such jobs in terms of their arrival frequency, quantity and corresponding business hours.

This section presents the data distribution analysis for the studied periodicity patterns of job submissions over the period of 28 days. This distribution analysis in conducted by exploring the inherent data distribution of job submission trend and by fitting the data to the closest theoretical distribution using a Goodness of Fit test for data normality based on the Cumulative Distribution Function (CDF) of job submission trends, including the mean and standard deviation of the trend. Figure 3.21 presents the CDF fitted with the closest theoretical distribution for the monthly trend of jobs submitted once, jobs with resubmissions by quantity

Table 3.6. Data Distribution Fit Statistics for Job Submission Trend

Trend	Distribution	Parameters	Skewness
Jobs Submitted Once	Weibull	c=6.665 σ=10808	-0.3958626
Jobs with Resubmission by quantity	3P Lognormal	σ=0.9024 Θ = 9459.2 ς = 7.8384	3.46417959
Jobs with Resubmission by type	Gumbel	μ= 1131.9 σ=360.83	2.87461102
Job Diversity	Normal	μ= 11454 σ=2189.8	0.20511085

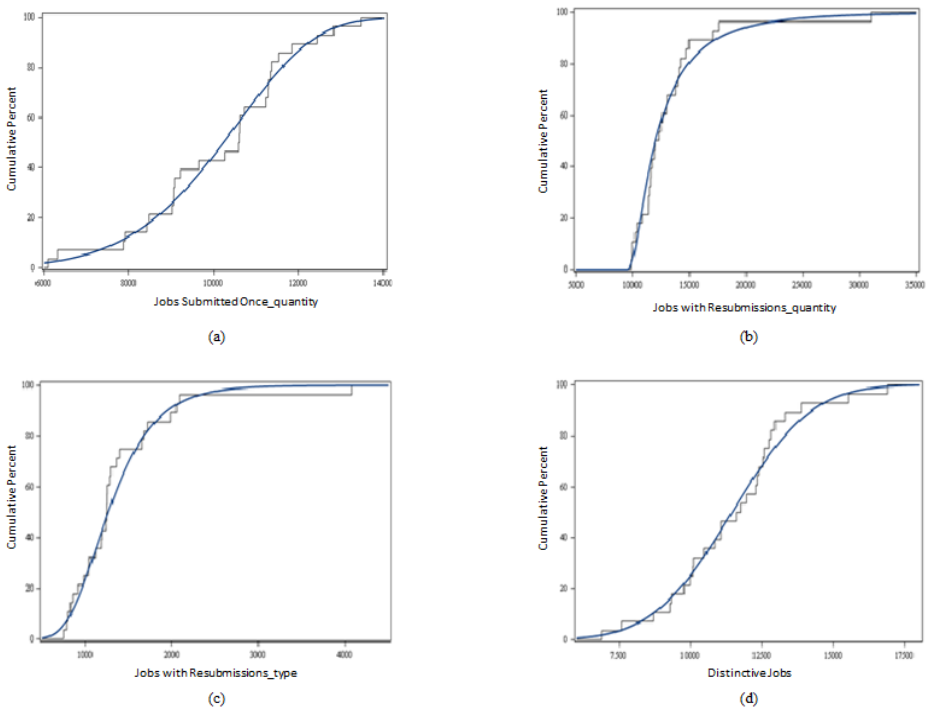


Figure 3.21. Data Distribution (a) Jobs Submitted Once (b) Jobs with Resubmission by Quantity (c) Jobs with Resubmission by Diversity (d) Job Diversity

and diversity respectively, and job diversity. Distinctive job trend in Figure 3.21 represents the total number of different job types submitted within the analysed trace logs. For instance, a given job submitted multiple times will be counted as a single job type.

Table 3.6 presents the best fit distributions and the corresponding skewness of the data trend. Jobs submitted once during the observation period predominantly flows Weibull distribution and the distribution curve is negatively skewed by a small margin. This shows that the trend of jobs submitted once is nearly equally distributed over the month. Further, jobs involving resubmissions by quantity and diversity follows Lognormal and Gumbel distributions and both the curves are significantly positively skewed respectively. This shows that majority of the Cloud workloads involves resubmissions both due to recurring submissions resulting from periodicity and resubmissions resulting from job terminations. Job diversity trend follows normal distribution with the curve exhibiting a slightly positive skewness, demonstrating that job heterogeneity is fairly equally distributed among the total number of jobs submitted at the datacentre.

3.6.3.2 Time-of-the-Day and Day-of-the-Week Effects

A deeper investigation is conducted on a randomly chosen Day 10 (Wednesday Week 1) Day 17 (Wednesday Week 3) and Day 21 (Sunday Week 3) for the purpose of exhibiting the day-of-the-week and time-of-the-day periodical effects on job arrival trend during week day and weekends respectively. Figure 3.22 presents the total number of jobs submitted on an hourly basis starting from 12 am on Day 10, Day 17 and Day 21 across a period of 24 hours respectively. It can be observed that the number of job submissions is fluctuating on a timely fashion within a single day. The number of job submissions is fairly less during the night time compared to those in the day-time submissions, implying a diurnal trend of job submissions on Day 10 and Day 17. Further, a significant increase in the number of job submissions is evident

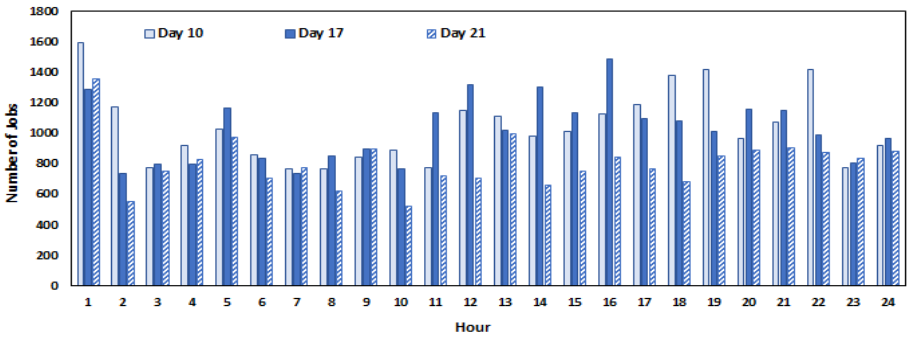


Figure 3.22. Hour-wise Quantification of Job Submission

during the peak time hours as the day progresses during Day 10 and Day 17. Job arrival rate is fairly exhibiting a random fluctuation for the whole 24 hours on Day 21, where a diurnal trend is hardly evident. This is postulated to the variations among user activities during weekends during which users may not have a timely trend of usage patterns whereas during week days user activities highly correlates with the business hours. The first hour (12 am – 1 am) of all the three observed days is showing an unusual pattern, since the number of jobs submitted are significantly higher than those in the remaining hours. Despite being a night-time hour, this increased number of job submission necessitates further investigation into user behaviours during this hour, which is presented section 3.6.4.

3.6.4 User Behavioural Periodicity

Workload behaviours are closely associated with user activities, since workloads originate from users. Thus it is necessary to analyse user behaviours in order to accurately validate the workload periodicity at the Cloud datacentres. Users are the actual responders who drive job submissions and task behaviours in terms of the resource requests and submission volumes. A Cloud user profile can be defined as an integrated composite of job and task profiles, consisting of the time of job submission t_s, and user name n_u, job name n_j and the associated resource demands in terms of CPU and memory $r_{(c,m)}$, as shown in equation 3.8.

$$U = \{t_s , n_u , n_j , r_{(c,m)}\} \tag{3.8}$$

In general, jobs originating from a single user might exhibit similar periodical trend at the Cloud datacentres. A single user is witnessed to be submitting approximately 18% [19] of the total jobs submitted in the analysed trace data, thus exhibits better predictability. Parameters used to identify periodicity among user behaviours include their submission time interval, resource demands, recurring application types, and user navigation etc. Think-time is a crucial metric in user behaviour modelling, which determines the average waiting time of users under delays, and it is directly related to the termination of the workloads triggered by users. Users causing increased terminations might cause an increased number of resubmission, thus contribute to an enormous amounts of workload arrival at the datacentres.

The active number of concurrent users during a service session is crucial in determining the level of resource scaling at the datacentres. The demand for CPU cores generally increases with the increasing number of concurrent users. It is worthy of note that users coexisting in a service session not necessarily demand similar amounts of resources. Usually, cloud providers employ a high level of parallelism under high level of concurrent users requesting diversified resources.

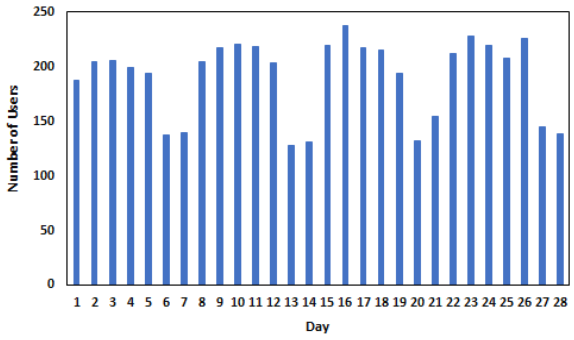


Figure 3.23. Active Users

3.6.4.1 Active Users

Figure 3.23 represents the total number of active users submitting jobs at the Cloud datacentre on a daily basis, illustrating an average of 190 users per day submitting jobs to the analysed datacentre.

Interestingly, a single datacentre encounters users characterising diversified range of resource demands. From the analysed datacentre, it can observed that variety of users submit single job through to a few hundreds of jobs within a single day. As shown in Figure 3.24(a), an average of 25 users are characterised with a single job submission per day. Though viewed as a single job, a single job submission might encompass one to several number of task requirements. Such type of users with limited job submissions impose increased complexities in accurately modelling their behaviours at the datacentre in spite of their occasional and rare submission behaviours. Also, it is important to differentiate these genuine limited users from anomalies since they can easily be judged as anomalies in spite of their distinctive and limited user profiles. In such cases of single job submissions, day-of-the-week effect might provide the necessary insights for deriving their user profiles. At a coarse grained analysis, a weekly periodical trend is evident among both the total number of active users and users with single

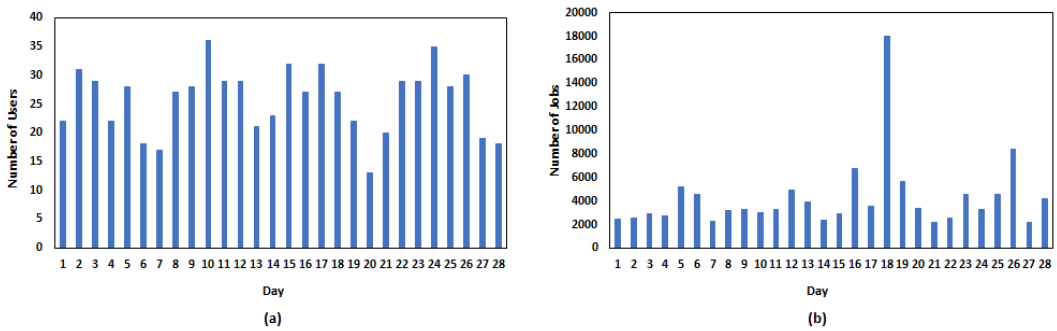


Figure 3.24. Users Statistics (a) Users with Single Job (b) Maximum Jobs from a Single User

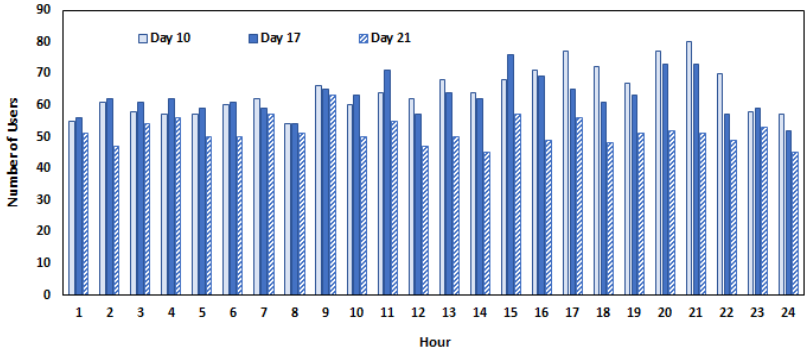


Figure 3.25. Hour-wise Quantification of Active Users

submissions, with the number of active users declining over the weekends and increasing during the weekdays. Figure 3.24(b) illustrates the maximum number of jobs submitted by a single user on a daily basis. It can be observed that Day 18 (Thursday) comprises the most number of job submissions from a single user, who has submitted 81.21% of the total jobs submitted on that particular day. This particular user has submitted 506 diverse job types across a total of 18024 job submissions. This corresponds to the workload periodicity analysis discussed earlier, where Day 18 exhibits an abnormal number of job submissions compared to the rest of the days of observation.

3.6.4.2 Time-of-the-Day and Day-of-the-Week Effects

Further to the day-wise periodicity of active users, it is essential to investigate the day-of-the-week and time-of-the-day effects on the active user in order to accurately model user periodicity. Figure 3.25 presents the hour-wise plot of the number of active users submitting jobs during Day 10 (Wednesday Week 2)), Day 17 (Wednesday Week 3) and Day 21 (Sunday Week 3) respectively. Clearly a diurnal trend is evident among the number of active users, with the number of users increasing during the day-time and declining during the night hours on Day 10 and Day 17. This support the early observations of diurnal trend of job arrival during weekdays, thus user activities are observed to be in correlation with the business hours during weekdays. Again diurnal trend is hardly evident on Day 21, and the number of active users is exhibiting a fairly random fluctuation throughout the day.

These observations are leading to the inference that job arrival pattern highly depends on the active users and their behavioural patterns, and both job and user behavioural patterns may not be similar for weekdays and weekends. From the observations presented earlier in Figure 3.22, an increase in the number of job submission has been witnessed between 12 am and 1 am during all the three observed days.

Table 3.7. Job Submission Statistics for Active Users during 12-1 am on Day 10, Day 17 and Day 21

Day 10 (Wed)			**Day 17 (Wed)**			**Day 21 (Sun)**		
User Name	**Jobs**	**Event (%)**	**User Name**	**Jobs**	**Event (%)**	**User Name**	**Jobs**	**Event (%)**
User A	361	22.59	**User A**	403	31.26	**User A**	367	26.98
User 2	339	21.21	*User 2.T*	211	16.36	*User 2.T*	225	16.54
User 3	153	9.57	*User 3.T*	76	5.89	**User B**	117	8.60
User B	127	7.94	**User C**	75	5.81	*User 3.T*	81	5.95
User C	75	4.69	User 5	52	4.03	**User C**	74	5.44
User 6	56	3.50	**User B**	45	3.49	User 6	50	3.67
User D	36	2.25	**User E**	44	3.41	User 7	42	3.08
User 8	36	2.25	**User D**	36	2.79	**User D**	36	2.64
User E	34	2.12	**User F**	32	2.48	**User F**	32	2.35
User F	32	2.002	User 10	31	2.40	**User E**	31	2.27

Table 3.7 presents the total number of jobs submitted by the first 10 users characterising maximum submissions during this observed hour of 12 am – 1 am on Day 10, Day 17 and Day 21. It can be observed that the first two users have submitted 361 and 339 jobs on Day 10, 403 and 211 jobs on Day 17, and 367 and 225 jobs on Day 21 respectively during this hour. These two users have contributed 43.8% on Day 10, .4762% on Day 17 and 43.5% on Day 21 respectively to the total number of jobs submitted during this hour. These two users have submitted most of their jobs during this hour, and have lesser number of job submissions during the other hours within the same day. Users with job submissions during all the three days are indexed with identical alphabets (bolded), and users with job submissions on two days are indexed with similar user name (italicised) and users with submission on only one of the three days are numbered based on their number of job submissions accordingly in Table 3.7. Interestingly, User A has submitted jobs during all the three days, who has submitted the most number of jobs during both the observed days, characterising nearly identical number of job submissions in Day 10 and Day 21 and increased number of submissions in Day 17. Further from Figure 3.17(b), User A has a characteristic submission of around 360 jobs every day between 12 am - 1 am, thus characterising a periodic submission behaviour. The increased number of job submission in Day 17 from User A is an unusual trend from this user. From the listed top 10 users in Table 3.7, a total of 6 users have submitted jobs during both Day 10 and Day 21 characterising similar number of jobs. Such users can be characterised as exhibiting better periodical trend of job submission and their behaviours show higher predictability. Behavioural similarity of users in Day 10 and Day 17 exhibits a better trend of day-of-the-week

effects and users with similar trend in all the three days exhibits a better trend of time-of-the-day effects.

3.6.5 Machine Usage Frequency

Server resources in a Cloud datacentre can be viewed as common and uncommon servers. Common servers are those resources used quite often and usually characterise low utilisation profiles, whereas uncommon servers show higher utilisation rates and are used occasionally. The term server usage here reflects the frequency count of the servers used for allocating jobs for execution in the datacentre, and server utilisation refers to the current level of utilisation to the maximum capacity of the corresponding server. Servers those used consistently for VM/LXC deployments for job execution are termed as common servers, whereas uncommon servers are usually used for occasional and rare job types characterising special and specified server needs. Servers in the datacentres are usually assigned with a permanent Machine ID, jobs arriving at the datacentres are allocated onto the temporarily deployed VMs onto these permanent machines. Schedulers in the datacentre receive jobs and find the appropriate physical machines to allocate tasks for processing. This process of selecting machines is usually governed by various factors such as the machine availability status, current server load, utilisation levels, nature and intensities of jobs etc. This section analyse the machine usage patterns based on the actual scheduling of tasks. For this analysis, Day 3, Wednesday, has been randomly chosen for investigating the machine events in order to exhibit the machine usage frequency patterns. A total of 12503 servers have been utilised for scheduling the incoming tasks during Day 3 across a period of 24 hours. Interestingly, an extreme dynamism is evident in the usage patterns of these 12503 servers as the usage frequency of the machines in the server farm ranges from once to a few hundred times. A total of 31 machines has been utilised only once for the whole day, such machines are generally the uncommon servers used occasionally. On the contrary, some of the machines have been consistently utilised for job scheduling

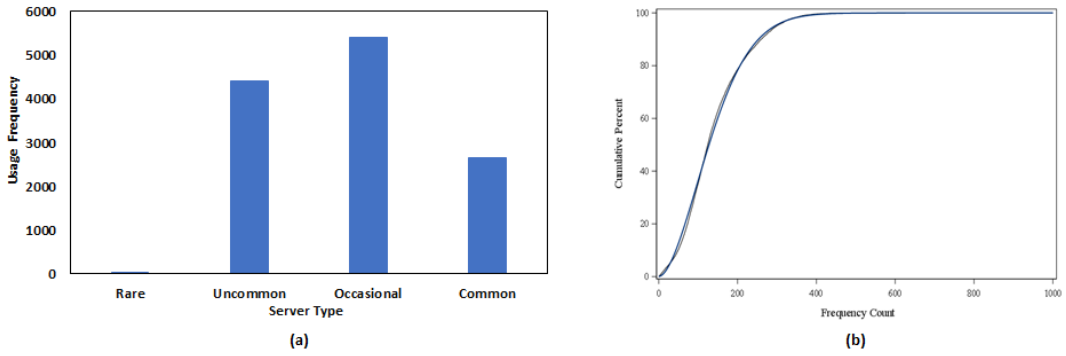


Figure 3.26. Machine Usage Frequency (a) Quantification (b) Data Distribution

Table 3.8. Data Distribution Statistics for Machine Usage

Distribution	Parameters	Skewness
Weibull	c=1,7518 σ=157.18	0.85857994
3P Gamma	Θ =26.88 α =4.109 σ = 40.69	0.85857994

throughout the entire day. In order to exhibit the usage patterns of the server, the machines are classified based on their usage frequency as rare (used once), uncommon (used for less than hundred times), occasional (used between 100 and 200 times) and common servers (used for more than two hundred times). Figure 3.26(a) presents the usage frequency patterns for the 12503 machines in the server farm during a period of 24 hours in Day 3. It can be observed that nearly half of the servers are occasional servers, accounting for 43.2%, followed by uncommon servers accounting for 35.21%, and common servers used for 21.27% and rare servers accounting only for 0.248% of the total machine usages. Figure 3.26(b) illustrates the distribution of the machine usage frequency pattern in terms of the cumulative distribution function fitted with the closest theoretical distribution. The machine usage frequency distribution predominantly fits Weibull and 3-Parameter Gamma distribution during Day 3. Table 3.8 present the statistics for the data distribution analysis for machine usage pattern over the 24 hours in Day 3. It can be observed that the distribution is positively skewed at 0.8585, insisting that a fewer proportions of the server farms characterise maximum utilisation within a single day, 21.27% of the total servers have been consistently utilised for majority of the incoming jobs within a single day. Turning the common servers off for energy conservation might compromise service quality since jobs are placed onto the common servers quite consistently. Thus switching the common servers off and on might incur undesirable wait time for users before their required resources become available. Thus uncommon and rare servers should be of the provider's target when energy management is attempted via the server switching strategy.

3.6.6 Summary

This section analysed the submission and execution events at a large-scale datacentre environment, with the motivation of exhibiting the dynamic nature of Cloud workloads and users, particularly the periodicity trend of Cloud jobs and users submitting jobs at the Cloud datacentres. Extensive analysis conducted on real-life Cloud trace logs reveal that both Cloud workload and user behaviours are highly dynamic and are bound to periodical effects in

accordance to the operating business hours. In general, Cloud users leave traces of their job submission trend in a typical Cloud datacentre, from which user behaviours can be modelled along with characterising their workload submission trends. Users with limited profiles or traces impose increased complexities in accurately modelling their behaviours at the Cloud datacentres than users characterising increased job submissions. Building user and jobs profiles by incorporating various time-bound periodical effects helps accurately model user and workload behavioural patterns in Cloud environments. Majority of job types are actually submitted once and a small proportion of the total distinctive jobs are resulting in majority of resubmissions, and certain jobs are characterised by every day submissions with similar number of arriving jobs. Arriving number of job submissions and active number of users follow a diurnal trend, and further corresponds to the operating business hours with week day increase and weekend declines. One or fewer number of users can contribute maximum number of job submissions during that particular session.

The characterisation of the workload and user behaviours presented in this chapter finds applications in prediction analytics, resource provisioning, server management, and job allocation etc., at the datacentres. The measure of inherent periodicity can influence the prediction accuracy. However existing works of predicting the incoming job trends for server scaling and resource provision have not given suffice emphasis to the inherent degree of periodicity among Cloud users and their corresponding workloads. Furthermore, this degree of inherent periodicity highly fluctuates across different workloads and users, which necessitates treating every job submitted by every user uniquely during an observation period. Treating all the incoming workloads in a common way during a given session might not help in understanding the characteristics of the incoming workloads for further predicting their future trend.

4 User Behaviour Forecasting Framework

4.1 Overview

Forecasting the anticipated future workloads would help the service providers to achieve an optimum energy-efficient scaling of the datacentre resources in accordance with the incoming workloads. As discussed in the previous chapter, Cloud workload and user parameters exhibit both temporal and/or spatial variations and correlations, which could be both significant positives (maximum correlations) and significant negatives (minimum correlations). Significant positives represents the persistence of a system metric to remain consistent over a period of time. The degree of such positive and negative correlations should be carefully incorporated in prediction modelling, since clusters of significant positives lead to effective prediction analysis whereas clusters of significant negatives heavily affects the prediction accuracy. These correlation metrics exhibit dynamic shifts in time, as the workloads usually fluctuate in time driven by user behaviours. Identifying the positive correlations among Cloud workloads and user behaviours over time helps to extract the hidden periodicity among the Cloud entities. Despite the existing works of prediction models in Cloud Computing to date, there is still a lack of an effective prediction model that can capture the inherent characteristic diversity and the correlations between users and their jobs submission trends.

To this end, this chapter presents a novel prediction framework named InOt-RePCoN (Influential Outlier Restrained Prediction with Confidence Optimisation) aimed at a tri-fold forecast of user behaviours, forecasting the anticipated number of job submissions in a session, session duration anticipated for users along with predicting their job submission trend in terms of the submission interval of consecutive submissions of the same jobs from users. This tri-fold forecast of user behaviours helps the service providers with a pro-active datacentre management for the purpose of achieving an optimum energy-efficient scaling of the server resources in accordance with the arriving workloads. The proposed prediction model exploits both the time-of-the-day and day-of-the-week periodicity effects for characterising user periodicity and predicts the future user behaviours based on a confidence optimised ARIMA forecast. This model uniquely analyses every single job belonging to a user to achieve a reliable level of prediction accuracy. The important contributions of this chapter include the following.

a) Analysis and extraction of the predictive features of both users and their corresponding job submissions to build the predictability profiles of users and jobs. By exploiting

periodicity effects, the proposed model computes the predictability weights for every single job submitted by users. This predictability weight has been exploited by the proposed model to reduce the average prediction error by uniquely treating jobs and users characterising different predictability weights.

b) A tri-fold prediction of user behaviour in terms of their job submission trends in Cloud environments. Firstly, forecasting the number of expected submissions of jobs for the target users. Secondly, forecasting the session duration for the anticipated users and finally, predicting job submission interval of consecutive submissions of the same job from users for an observed session.

4.2 InOt-RePCoN Framework

This section describes the integral components of the proposed prediction framework InOt-RePCoN, illustrated in Figure 4.1. InOt-RePCoN encompasses three integrated components such as a Rule Miner, a Validator and a Predictor. The integrated components of the proposed prediction model will have the following functionalities.

Rule Miner: The main functionalities of the rule miner are to select historical samples for training the prediction model and to compute the predictability weights for the target users and their jobs. The rule miner initially reads user profile from the incoming user request. By delving into the time-of-the-day and day-of-the-week effects, the rule miner selects historical samples based on the day and time of the current sampling period. The window length of the historical samples are chosen the same as the length of the currently interested sample. Based on the two aforementioned periodicity effects, the rule miner computes a predictability weight for user and their workloads in the current sampling period.

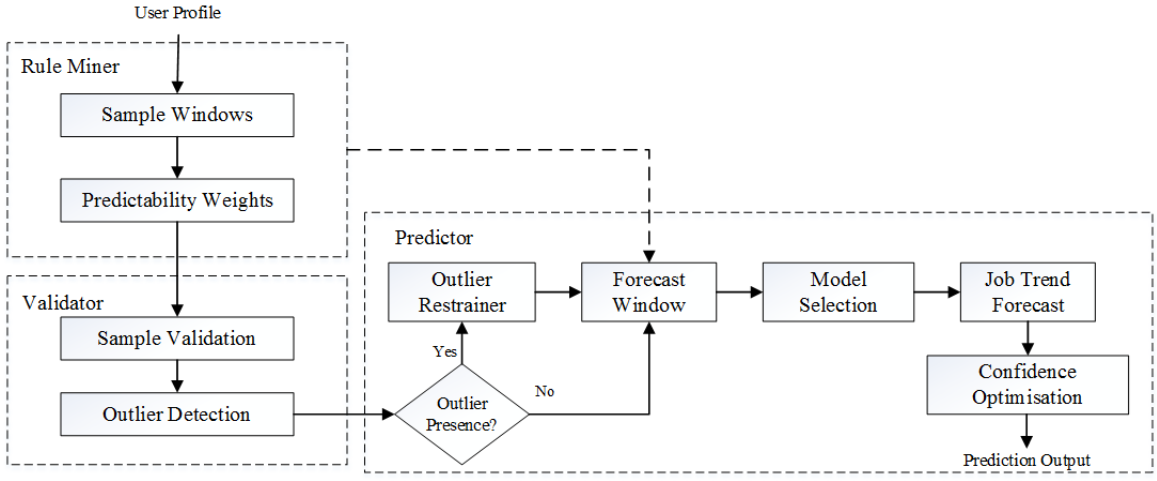


Figure 4.1. InOt-RePCoN Framework

Validator: The validator incorporates two sub-components: sample validator and an outlier detector. The sample validator chooses the most suitable historical samples from the samples initially chosen by the rule miner. This historical sample is chosen based on the measure of the degree of correlation between the current sample and the historical samples for a close enough match, this is done because the rule miner might return more than one historical samples. Further to this, another sample from the historical traces is chosen called the reference sample which is the next successive set of sample to the one chosen by the validator. The outlier detector measures the degree of residuals present in the prediction sample and in the chosen historical samples.

Predictor: The main functionalities of the predictor are to forecast the number of submissions, session duration and submission interval trend for the target users and jobs based on the samples chosen by the validator. It incorporates five sub-components such as an outlier restrainer, window forecaster, prediction model selector, predictor and a confidence optimiser. In the context of submission interval forecast, submissions from a given user characterising abnormal trend in terms of prolonged submission interval than majority of the submission interval within a session are characterised as outliers, such outliers usually exhibit a significantly deviating submission interval. Outlier restrainer checks for influential outliers present in the prediction sample and further restrains their impacts up on prediction accuracy. Window forecaster computes the number of submissions and session duration anticipated for the target users and jobs during an observation period. The model selector and predictor are modelled to forecast the anticipated future trend of job submission interval for the target users by selecting the most appropriate predictor values. Further the accuracy of the forecast is enhanced by optimising the confidence interval of the forecast.

4.3 Prediction Mechanism

4.3.1 Rule Miner

The rule miner receives the input consisting of the current sample of user trend and has two important functionalities. Firstly, the rule miner selects the prediction samples from the historical data, by the way of matching the start-end time and duration of the current sample such that the chosen historical samples are identical in duration and start-end time of the current sample. Two such historical samples are selected, one from the same representative day in the previous week of the current sample in order to validate dual effects of time-of-the-day and day-of-the-week effects collectively. The second historical sample is chosen from the previous

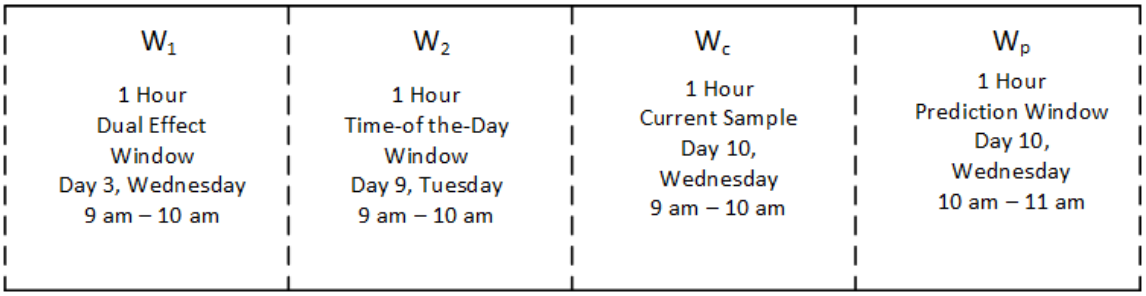


Figure 4.2. Rule Miner Window Illustration

day of the current sample to validate the time-of-the-day effects. After choosing the samples, the rule miner forms four different sample windows such as the current sample window (W_c) containing the current user trend from which the future trend is expected to be forecasted in the prediction window (W_p), window 1 (W_1) is built with the dual effect sample, and window 2 (W_2) for the time-of-the-day sample accordingly. Figure 4.2 illustrates the various sampling windows formed by the rule miner, in reference to a randomly chosen Day 10, Wednesday, 9 am – 10 am data contained in the current sample. A typical window size of one hour is adopted in this thesis, since the VM allocation duration of one hour is commonly witnessed in Cloud services such as Amazon EC2. However, the window size of the proposed model can be relaxed in actual deployment depending upon the granularity of the VM allocation policies of the Cloud services. Secondly, the rule miner computes a predictability weight P_s for every user contained in W_c, along with assigning predictability weights for all the type of jobs submitted by the target users. This predictability weight determines the degree of predictability of users and jobs depending on the current trend of users and jobs satisfying the sample window rules of the rule miner.

The predictability weight, P_s is assigned to every user and every job type belonging to a user by the measure of the W_c samples satisfying the day-of-the-week and time-of-the-day effects in accordance with W_1 and W_2. The rule miner assigns four levels of prediction weights to users and their jobs. A weight of level 3 is assigned to users and jobs satisfying both the dual effect window W_1 and the time-of-the-day window W_2. A weight of level 2 is assigned to users satisfying only the dual effect window W_1. A prediction weight of level 1 is assigned to users satisfying only the time-of-the-day window W_2. User not satisfying any of the two windows will be assigned with a predictability weight of level 0. Thus level 3 implies a higher degree of predictability through to level 0 implies a poor degree of predictability for users and jobs. By assigning predictability weights to users, InOt-RePCoN exploits the periodicity among user

behavioural pattern in terms of their submission trend during the two historic sample windows for choosing the most appropriate sample for prediction. The predictability weights are used to determine the error margin and forecast accuracy for users and jobs. Higher prediction weight implies better correlation of the predicted trend with the actual trend of the corresponding users and jobs. In other words, an increased level of predictability weight reflects the increased expectation of prediction accuracy. After computing the predictability weight for every users in the current sample window, the rule miner constructs a predictability weight table based on the list of users l_c, l_1 and l_2 respectively contained in the current sample window, window 1, and window 2. These processes are repeated for every job submitted by every user contained in the current sample window in order to assign the predictability weights to all jobs submitted by every users. The construction of the predictability weight table based on the predictability weight computation for users and jobs are detailed in section 4.4.3.

4.3.2 Validator

After assigning the predictability weights to both users and jobs in the current sample window, the proposed model further validates the similarities of user behavioural trend in the current sample window with both window 1 and window 2 respectively. Though the rule miner relies on both the time-of-the-day and day-of-the-week effects to assign the predictability weight, this similarity measure is conducted for the purpose of training the most suitable historical sample to the predictor from W1 and W2. Every single user and their jobs are analysed uniquely, since users co-existing in a given session usually exhibit varied job submission trend and service requirements. Thus the inferences obtained from individual user behaviours are not applicable to other co-existing users even they belong to the same window sample. The behavioural trend of user in terms of their job submission patterns in the three sample windows are analysed to measure the similarity of user behaviours in the current sample window with both the two historical windows.

4.3.2.1 Similarity Analysis

A single user might submit several jobs and thus characterised by multiple job submission trends. Thus the validator analyses the similarity measure for every individual job submitted by users. For this reason, the proposed model is aimed at forecasting user behavioural trend for individual jobs submitted by the corresponding users. Firstly, the submission interval S_i for a given job is calculated in the three sample windows using equation 4.1.

$$S_i = t_{j(i+1)} - t_{ji} \tag{4.1}$$

where, t_{ji} is the submission time of job j at time i, and $t_{j(i+1)}$ is the submission time of job j at time i+1, which is the next successive submission time of job j.

The validator measures the degree of correlation among the submission intervals of job j by validating the linear dependence of every consecutive submissions of job j in the current sample window with those in window 1 and window 2 respectively. The degree of association among the consecutive submission of job j is validated by the measure of correlation coefficient by modelling S_i of job j as a time series. Since Cloud workloads are dynamic in nature, submission interval of the same job by the same user within a single sample window might exhibit significant variation resulting in the presence of outliers in job submission interval. A submission sample is known to be containing outliers when a very few submission observations significantly deviate from majority of the remaining submission points. For instance, in the case of submission trend, when a submission characterise prolonged interval time than usual then it will be characterised as outlier in the proposed model. An increased presence of such outliers cause the submission interval pattern of the corresponding jobs to exhibit a non-linear trend and significantly affects the prediction accuracy. The greater the number of outliers the higher is the deviation of the observation from a linearity trend. Thus, the validator measures the presence of outliers in the submission interval trend of the target jobs from users for the purpose of measuring the degree of linearity in job submission interval. Presence of outliers is quite common in job submission trend owing to the increased dynamicity in Cloud environments. The characteristics of an outliers among the data points can be defined as in equation 4.2.

$$O_t = \begin{cases} 0 & if\ |r_i| \leq C(p) \\ 1 & otherwise \end{cases} \tag{4.2}$$

where, O_t is the outlier, r_i is the residuals with $i = 1, \ldots, n$ and $C(p)$ is the cut-off value for deciding the residuals. The observations falling beyond the cut-off value are determined as residuals and are characterised as outliers in the sample. The cut-off value is dynamically determined for the prediction samples, based on the spatial dependency among the individual submission observations with a given sample. A simpler correlation coefficient do not scale well and might result in an incorrect estimation of the association among consecutive submissions of a given job due to the presence of outliers. Based on the number of the outliers, the validator decides whether the prediction sample in the current sample window requires subjecting to outlier restrainer before training into the predictor (further detailed in section 4.3.3.2, 4.4.4 and 4.4.7). Now, the validator computes the confidence and prediction ellipses

for job submission trend of the target jobs from target users contained in the current sample window and window 1 and window 2 respectively for the purpose of accurately estimating the correlation coefficient and the presence of outliers among the submission trend. Confidence ellipse defines the event population mean and the prediction ellipse defines the confidence bounds of predicting the future observations.

A bivariate distribution has been adopted for analysing job submission trends, the validator contrasts job submission trend in the current sample window against those both in window 1 and window 2, but one at a time. The Confidence and Prediction ellipse computations are defined as follows. Let *Z* and *S* be the sample mean and covariance matrix of a random sample of size n with mean μ and covariance $\sum$. The variable $\mathrm{Z} - \mu$ is a bivariate distribution with zero mean and covariance of $(1/n)\sum$, and it is independent of *S*. Hotelling's T^2 statistic depicts equation 4.3, which presents a multivariate distribution particularly when the data characterise more than one parameter for each sample. Hotelling's T^2 statistic can be used to compare two sets of data to evaluate how well the two sets of samples are distributed.

$$T^2 = n(Z - \mu)' S^{-1} (Z - \mu) \tag{4.3}$$

Now a 100(1-α)% confidence ellipse for μ is computed using equation 4.4, where $F_{2,n-2}(1 - \alpha)$ is the $(1 - \alpha)$ critical value of a F distribution with degrees of freedom 2 and $n - 2$.

$$\frac{n}{n-1} (Z - \mu)' S^{-1} (Z - \mu) = \frac{2}{n-2} F_{2,n-2}(1 - \alpha) \tag{4.4}$$

The prediction ellipse estimates the new observations Z_n as a bivariate normal variate with zero mean and covariance $(1 + \frac{1}{n})\sum$, independent of S, given by equation 4.5.

$$Z_n - Z = (Z_n - \mu) - (Z - \mu) \tag{4.5}$$

Now a 100(1-α) % prediction ellipse is given by equation 4.6.

$$\frac{n}{n-1} (Z - \mu)' S^{-1} (Z - \mu) = \frac{2(n+1)}{n-2} F_{2,n-2}(1 - \alpha) \tag{4.6}$$

Both the generated confidence and prediction ellipses will have common centre (the sample mean), common major and minor axis. The degree of association between the consecutive job submissions and their submission interval is measured using Pearson correlation coefficients using equation 4.7. A positive correlation coefficient insists a close correlation between the two variables x and y. This correlation coefficient measures the degree of linear dependency between consecutive submission of jobs and their submission interval. By generating the

confidence and prediction ellipses, the validator presents the outliers contained in job submission trend, usually the outliers fall beyond the generated ellipses. Since outliers directly affect the prediction error margin, presence of such residuals cannot be ignored whist generating the prediction ellipses.

$$\rho_{xy} = \frac{Cov(x,y)}{\sqrt{V(x)V(y)}} = \frac{E((x-E(x))(y-E(y)))}{\sqrt{E(x-E(x))^2 E(y-E(y))^2}} \quad (4.7)$$

By generating independent prediction ellipses for the target jobs extracted from the current sample window, window 1 and window 2, the degree of similarity among the three sample windows is obtained, which is then used to select the most suitable historical sample for further prediction analytics. The correlation coefficient is determined by various intrinsic factors such as the time, user intention, business pattern etc. For instance, the current trend of the target users and jobs might have a close correlation with the time-of-the-day effects or with the day-of-the-week effects or both. Though window 1 satisfies both the time-of-the-day and day-of-the-week effects, window 2 will still comprise the most recent historical sample (just a day old), with window 2 comprising a week old sample. Thus validating the correlation coefficient for similarity measure between the current and historical samples is crucial in determining the prediction accuracy.

4.3.3 Predictor

This sections details the proposed prediction framework based on autoregressive integrated moving average (ARIMA) technique with an integrated outlier restrainer and confidence optimiser. Auto Regression scales well for prediction when the prediction samples characterise an inherent periodicity. The predictor models job submission trend of users as a time series to extract the periodical predictive characteristics from the current job submission trend of users. The integral components of the predictor are described along with their functionalities as follows.

4.3.3.1 Stationarity Test

Initially, the predictor conducts a stationarity test upon the submission trend of the target jobs and users in the current sample window for testing the degree of stationarity in the time series of job submission trend. The stationarity of the predictive sample are evaluated using an Augmented Dickey-Fuller (ADF) t-statistic test for stationarity by subjecting the submission trend for null-hypothesis. The degree of stationarity characteristics of job submission time and the submission interval are used to validate and select the appropriate ARIMA model for the

purpose of accurately forecasting the future observations. With job submission behaviour of users following a continuous time-series trend and expected to have a slow-turn around the data points, the ADF test is conducted using equation 4.8.

$$\Delta z_t = \alpha_0 + \Theta Z_{t-1} + \gamma t + \alpha_1 \Delta Z_{t-1} + \alpha_2 \Delta Z_{t-2} + \ldots + \alpha_p \Delta Z_{t-p} + \alpha_t \qquad (4.8)$$

The t-statistic on the Θ coefficient is used to evaluate the degree of stationarity and the submission trend is differenced when the trend exhibits non-stationarity. A more negative t-statistic depicts that the trend of the data does not need differencing. The null hypothesis of the ADF t-statistic test is given by,

$$H_o: \Theta \begin{cases} = 0 \; for \; data \; needs \; to \; be \; differenced \\ < \; 0 \; for \; the \; data \; trend \; is \; stationary \end{cases}$$

4.3.3.2 Outlier Suppression

Before attempting to forecast the anticipated submission trend, the prediction sample is subjected to robust regression for the purpose of restraining the effect of influential outliers up on forecast accuracy. The presence of the outliers in the submission trend is estimated based on equation 4.2, and the sample is subjected to robust regression depending on the degree of the presence of the outliers. Prediction samples suffering marginal or no outliers may not require robust regression. But, job submission trends of users usually suffer increased variance within an observed time period. Thus outliers and residuals are quite prominent in the trend of user submission behaviours in Cloud environments. The presence of such outliers in the prediction sample increases the error margin and often results in inaccurate prediction results. Thus it is essential to supress the influence of the presence of outliers for the purpose of achieving reliable level of prediction accuracy. Usually the presence of outliers is dominant in the Y direction of job submission trend, since the contamination of the data points resulting from the variances are mainly witnessed in the response direction. Thus the predictor initially estimates the presence of outliers by the degree of the variance in the submission interval, and further supresses the presence of such outliers by subjecting the prediction sample with robust regression as shown in equation 4.9 and equation 4.10. The estimation of outliers and suppression is illustrated in section 4.4.4 and 4.4.7 respectively. With the data contamination being witnessed in the response direction, robust regression algorithm computes the M estimates for regression based on iteratively reweighted least squares (IRLS). An IRLS fit is carried out in every iteration to apply a set of weights to the observations depending on the presence of outliers until convergence is achieved. The M estimator $\Theta_M \; of \; \Theta$ minimises the

sum of less rapidly increasing residual functions under residuals r_i, and Θ is the solution of the system of ρ equations.

$$Q(\Theta) = \sum_{i=1}^{n} \rho(\frac{r_i}{\sigma}) \tag{4.9}$$

where, ρ is the square function $\rho(z) = z^2$. If σ is known, then Θ_M is the solution for the system of ρ equations by the derivatives with respect to Θ.

$$\sum_{i=1}^{n} \Psi\left(\frac{r_i}{\sigma}\right) x_{ij} = 0, j = 1 \dots, p \tag{4.10}$$

where, $\Psi = \frac{\partial \rho}{\partial z}$, and the weight function depending on the residuals is $w(z) = \frac{\Psi(z)}{z}$. Robust regression is proceeded by alternately improving Θ in a location step and σ in a scale step, until convergence is achieved, whenever there is a relative change in the scaled residuals for the purpose of restraining the effect of influential outliers.

4.3.3.3 Forecasting Window

After supressing the effect of influential outliers in the previous step using robust regression, it is essential to initially forecast the characteristic number of job submission and session duration for the target users and jobs anticipated in the prediction window W_p. Session duration is the time interval between the first and the last submission of jobs from users during the period of observation. Another sample window named reference window W_r is introduced comprising historical samples from the next successive observation period of the historical window W_1 or W_2, whichever is finally validated by the sample validator. The length of this reference window W_r is adopted the same as the four sample windows initially constructed by the rule miner. After the construction of the reference window, the predictor generates a relative error margin E_1 for both the anticipated number of job submissions and the session duration for the target users and jobs contained in W_c in reference to the finally validated sample W_1 or W_2, using equation 4.11 (W_1 is considered as the validated window for the below descriptions).

$$E_{1n} = \frac{|V_{1n} - V_{cn}|}{V_{cn}} * 100 \tag{4.11a}$$

$$E_{1s} = \frac{|V_{1s} - V_{cs}|}{V_{cs}} * 100 \tag{4.11b}$$

where, E_{1n} and E_{1s} are the error percentage for the number of submissions and session duration respectively. V_{cn} and V_{1n} are the number of actual job submissions observed in W_c and W_1 respectively, and V_{cs} and V_{1s} are the session duration counter-part. In addition to E_1, another

expected relative error margin E_2 is computed based on the previously computed predictability weight P_s for the target users and jobs, using equation 4.12.

$$E_{2n} = \frac{V_{rn}}{100} * e_p \tag{4.12a}$$

$$E_{2s} = \frac{V_{rs}}{100} * e_p \tag{4.12b}$$

where, e_p is expected error percentage based on the predictability weights of job submission trend of user behaviours, set as 10, 15, 20 and 25 respectively for P_s values of 3, 2, 1 and 0, and V_{rn} and V_{rs} are the observed number of submissions and session duration for the target jobs and users in the reference window W_r. The final error margin E is computed for the forecasting window in terms of the anticipated number of job submissions and session duration based on equation 4.13. The error percentage is computed separately for the number of submissions and the session duration accordingly.

$$E = E_1 + E_2 \tag{4.13}$$

Now the anticipated number of submissions V_{pn} and session duration V_{ps} for the target users and jobs is computed for the forecasting window W_p, as $V_p = \{V_{pn}, V_{ps}\}$ with optimised error margin, using equation 4.14.

$$V_p = \begin{cases} V_r & for\ V_w = V_c \\ V_r - E\ of\ V_r & for\ V_w \ll V_c \\ V_r + E\ of\ V_r & for\ V_w \gg V_c \end{cases} \tag{4.14}$$

where, V_w is the value of the actual number of job submissions and the session duration for the target users and jobs observed in the historical window (W_1 or W_2) validated by the sample validator. In equation 4.14, $V_w \ll V_c$ insists a significant difference between the values in $W_{(1\ or\ 2)}$ and W_c, and a value is concluded to be significantly different if the difference is greater than half of E. If the difference is not significant, the values are considered to be equivalent.

4.3.3.4 Model selection and ARIMA Forecast

A random variable of a stationary time series has statistical properties over time with constant amplitude around the mean. But a non-stationary trend of a time series shows a more fluctuating amplitude and variance around its mean. A random variable of such series with non-stationarity usually incurs a combination of signal and noise. Though regression assumes a model for forecasting the future trend, it is essential to select the most suitable regression model with appropriate subset of predictor variables based on the trend of the variables contained in the

prediction sample. Based on the regression estimates on trend of the prediction sample, the predictor selects the appropriate ARIMA model for predicting the future trend of job submission interval. From the initial ARIMA identification for stationarity, the predictor also identifies the degree of Auto Regression and Moving Average processes required to be optimised depending on the autocorrelation and partial autocorrelation functions existing in the original series in the case of stationarity or in the differenced series in the case of non-stationarity among the data points. In a stationarised series, AR signatures associates a positive correlation to act as a partial difference in the forecasting equation whereas MA signatures associates a negative correlation to partially cancel the order of differencing in the forecasting equation. Thus in a stationarised series after differencing the original time series, an AR signature mimic the first difference and an MA term moderate the first difference, with a redundant AR-MA pair cancelling out the effects of each other. In general, ARIMA model delivers a forecast $\hat{y}_t$ for a stationary or a differenced time series, in which the predictors incur the lags among the dependent variable or the forecasting errors. A non-seasonal ARIMA model can be classified as ARIMA (p, d, q), where p is the autoregressive term, d is the number of non-seasonal differences required for stationarity, and q is the number of lagged forecast errors in the prediction equation. An ARIMA (p,d,q) with y denoting the d^{th} difference of Y, can be described using equation 4.15.

$$y_t = \begin{cases} Y_t, & if\ d = 0 \\ Y_t - Y_{t-1}, & if\ d = 1 \\ Y_t - 2Y_{t-1} + Y_{t-2}, & if\ d = 2 \end{cases}$$

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q} \quad (4.15)$$

This ARIMA forecast is expected to deliver the submission interval of consecutive submissions for the target users and jobs in the current sample window, with an upper and lower bound 95% confidence interval determined by the predictors.

4.3.3.5 Confidence Interval Optimisation

In general, the crispness of the ARIMA forecast heavily relies on the degree of seasonality existing among the consecutive submission observations. In a worst scenario, where the prediction samples contain minimal or no level of seasonality, then ARIMA model would most often deliver a linear forecast with a significantly larger confidence space. This type of forecasting trend would be of no benefit to Cloud providers whilst trying to predict the future submission intervals, since the linear forecast would insists that the submission interval

between consecutive submissions characterise zero units in time. This cannot always be true in Cloud datacentres, which necessitates to further optimise the forecast delivered by the ARIMA model for the purpose of achieving a more reliable optimum forecast for the submission window. In order to improve the prediction accuracy of the ARIMA forecast and to reduce the bound limits of the 95% confidence window delivered by the ARIMA predictors, the proposed framework further optimises the ARIMA forecast with a novel confidence interval optimiser. Since a continuous trend is expected for job submissions, submission interval between two consecutive submissions can only be a positive value in time. In order to satisfy this positive bound requirements of the confidence window, the optimiser nullifies the effect of the negative lower bounds against the corresponding positive upper bounds of the ARIMA confidence window to obtain the optimised window limits using equation 4.16.

$$L_m = \begin{cases} L_{li}\,, & for\ L_{li}\ significantly\ positive\ (i = 1, \dots, n) \\ U_{li} + L_{li} & for\ L_{li}\ significantly\ negative\ (i = 1, \dots, n) \end{cases} \tag{4.16}$$

where, L_m is the mean bound limits after nullifying the effects of the negative lower bounds of the ARIMA confidence interval and n is the number of submissions initially set by the window forecaster. Now the predictor further optimises the confidence interval against the actual forecast of the ARIMA model, by assuming a zero mean for the lower bound limits. With a zero mean for lower bounds in the confidence interval, the upper bound limits L_o are optimised based on the submission interval of the ARIMA forecast I_f, and the actual submission interval I_r observed in the reference window W_r, and the computed mean limit L_m, using equation 4.17.

$$L_o = \begin{cases} I_f, & for\ I_r\ not\ available \\ L_m + I_f, & for\ I_f < I_r \\ L_m - I_f, & for\ I_f > I_r \end{cases} \tag{4.17}$$

Thus the proposed model predicts job submission interval based on the forecasting window along with optimising the confidence interval of the ARIMA forecast. This optimised confidence interval reduces the interval bounds of the ARIMA confidence window for better accuracy and reliability in the prediction output. Usually in a Cloud Computing environment, under-prediction would have more disastrous effect on energy consumptions than an over-prediction from the energy perspectives. While the former results in additional wait-time for the service providers, the latter might lead to a quicker arrival of the anticipated job than expected. When jobs arrive quicker than expected, services can still be availed with a marginalised wait time for users. But jobs arriving later than expected might result in early provisioning of the resources causing undesirable energy expenditures. Thus InOt-RePCoN is

aimed at reducing the probabilities of under-predictions in job arrival trend delivered by the confidence interval optimiser of the forecasting framework. The proposed framework further optimises the bound limits to deliver the confidence interval W_{con}, for the purpose of reducing the probabilities of under-prediction using equation 4.18. Most often, Cloud user behaviour trend of job submission interval is governed by the presence of influential outlier. In this case, the anticipated trend is expected within the bounds of W_{con} and the zero mean lower bounds of the forecast. For samples with no or minimal influence of the outliers, the future trend is expected to be in correlation with W_{con}.

$$W_{con} = \begin{cases} I_f & for\ \frac{I_f}{2} > L_o > (I_f * 1.5) \\ L_o & otherwise \end{cases} \tag{4.18}$$

4.4 Model Validation

Model validation is the process of substantiating a computerised model to determine whether its applicability possess an acceptable range of accuracy and reliability in consistent with the intention of the model application. This section validates the proposed prediction model by the way of training a real-world Cloud datasets into the model for forecasting user behaviours.

4.4.1 Data Preparation

The dataset comprises job and task profiles across a period of 28 days of datacentre execution. The entire dataset is prepared with a day-wise sampling, with a single day spanning across 24 hours starting from 12.00 am for a given day. Then, Day 10 Wednesday has been randomly chosen as the training sample and InOt-RePCoN is expected to predict user behavioural trend during the period of 1 am - 2 am, using the sample of 12 am - 1 am as the current window W_c sample.

4.4.2 Sample Selection

After processing the raw datasets, input the samples are trained into InOt-RePCoN. Based on the prediction objective of forecasting user behaviours during 1 am – 2 am on Day 10, Wednesday, the rule miner constructs the sample windows. Now, the current sample window (W_c) comprises the data from 12 am – 1 am of Day 10 Wednesday, window 1 (W_1) comprises the data sample from 12.00 am – 1.00 am of Day 3 Wednesday and Window 2 (W_2) comprises the data sample from 12.00 am – 1.00 am of Day 9 Tuesday. The proposed prediction model is expected to predict user behaviours for the predictor window W_p, which is 1 am – 2 am of Day 10 Wednesday. Further to validate and optimise the confidence interval of the forecast

results and to set the forecast window, the rule miner further samples the data obtained from 1 am - 2 am, Day 9 Tuesday and 1 am – 2 am Day 3 Wednesday for the purpose of validating the reference window (W_r).

4.4.3 Predictability Weight Computation

The current sample window W_c comprises a total of 55 users (where every user is unique), implying all those 55 users have co-existed with their characteristic job submissions during the one hour observation period of W_c. The rule miner builds the historic sample windows as described earlier, and computes the predictability weight for all the 55 users. For the ease of reading, the 55 users of W_c are named as User 1 through to User 55, in the descending order of their corresponding number of job submissions. Table 4.1 presents the submission observations for the first 10 users from W_c.

Table 4.1. User Submission Statistics in Wc

User Name	Number of Job Submission	Event Proportion (%)
User 1	361	22.59
User 2	339	21.21
User 3	153	9.57
User 4	127	7.94
User 5	75	4.69
User 6	56	3.50
User 7	36	2.25
User 8	36	2.25
User 9	34	2.12
User 10	32	2.002

Table 4.2. User Predictability Weights

Users (n_u)	User Predictability Weight (P_{su})
46, 47, 16, 12, 34, 39, 26, 4, 48, 49, 17, 40, 9, 50, 5, 3, 27, 52, 53, 41, 25, 10, 32, 54, 31, 21, 7, 29, 33, 22, 38, 20, 18, 43, 11, 1, 55	Level 3
30, 28, 42, 13	Level 2
23, 14, 51, 36	Level 1
44, 45, 15, 2, 6, 35, 24, 37, 19, 8	Level 0

Now, the rule miner computes the predictability weight for all the 55 users, as shown in Table 4.2. Out of the total 55 users, 37 users are exhibiting a level 3 predictability weight, reflecting their increased predictability. Following level 3 predictability, 4 users are assigned with level 2 predictability weight, 4 users with level 1 predictability, and 10 users with level 0 predictability weight respectively. These 10 users with level 0 predictability weight exhibit a very low probability of accurate prediction, as they characterise no historical traces. The prediction accuracy for such brand new users will be enhanced after obtaining sufficient number of behavioural traces from which their user profiles can be built and recorded.

After assigning predictability weights to users, the rule miner individually explores jobs submitted by all the 55 users contained in W_c. For space limitations, only the computed predictability weights for all jobs submitted by User 1 are presented in Table 4.3. User 1 has submitted the maximum number of jobs in W_c. Despite User 1 exhibiting a higher level of

Table 4.3. Predictability Weight for Jobs submitted by User 1

Job Name	**Number of Submissions**	**Event Proportion (%)**	**Job Predictability Weight (P_{sj})**
Job 1	102	28.25	Level 3
Job 2	102	28.25	Level 3
Job 3	102	28.25	Level 3
Job 4	11	3.04	Level 3
Job 5	11	3.04	Level 3
Job 6	11	3.04	Level 3
Job 7	3	0.83	Level 1
Job 8	3	0.83	Level 1
Job 9	3	0.83	Level 1
Job 10	3	0.83	Level 1
Job 11	2	0.55	Level 3
Job 12	2	0.55	Level 3
Job 13	2	0.55	Level 3
Job 14	1	0.277	Level 0
Job 15	1	0.277	Level 0
Job 16	1	0.277	Level 0
Job 17	1	0.277	Level 3

predictability, not necessarily all jobs submitted by User 1 should exhibit a predictability weight of level 3. User 1 has submitted 17 different type of jobs across 361 submissions in W_c, as shown in Table 4.1. For the ease of readability, jobs submitted by User 1 are named as Job 1 through to Job 17, presented in the descending order of the number of submissions. Now, the rule miner constructs the predictability weight table for all jobs submitted by User 1. Table 4.3 presents the predictability weight table comprising all jobs submitted by User 1. From the 17 jobs submitted by User 1, 10 jobs are assigned with a prediction weight of level 3, 4 jobs with level 1, and 3 jobs with level 0 respectively by the rule miner.

Now, the following sections of model validation demonstrates the integral process of InOt-RePCoN aimed at predicting user behavioural trend of User 1 whilst submitting Job 1.

4.4.4 Outlier Detection

From the statistical analysis conducted on the submission of Job 1 from User 1, a total of 105, 80 and 102 submissions of Job 1 are observed in W_c, W_1, W_2 respectively. The validator computes the presence of outliers in the current window sample before performing the similarity analysis. Figure 4.3 presents the presence of outliers contained among the trend of consecutive submission of Job 1 submitted by User 1 in W_c detected based on equation 4.2. The rectangular box depicts the normal distribution of the data, with the solid line in the rectangular box representing the median, and the circles illustrate the outliers. The farther the presence of an outlier from the median, more is the deviation of that corresponding outlier from the actual observation. It is evident from Figure 4.3 that the submission interval trend of Job 1 from User 1 suffers from a significant proportions of outliers in W_c, insisting the need for robust regression process for the purpose of restraining the effects of the influential outliers.

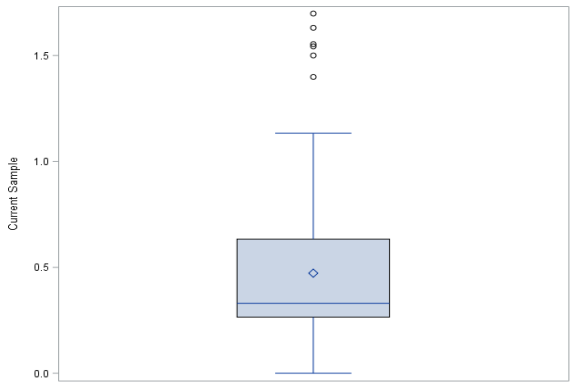


Figure 4.3. Presence of Outliers in the trend of Job 1 of User 1 in Wc

Table 4.4. Statistics of Prediction Ellipses

Sample	Number of Submissions	Minimum Interval (µs)	Maximum Interval (µs)	Pearson Coefficient	Mean	Standard deviation
Current	105	1939	101880483	0.16779	28347893	24153527
Window 1	80	1970	425730805	-0.01197	32610565	61543070
Window 2	102	2614	179952098	0.07393	27260821	27941765

4.4.5 Estimation of Ellipses

Now, the sample validator computes the confidence and prediction ellipses for the submission interval trend of Job 1 of User 1 in Wc, W1 and W2 respectively. The value of α in equation 4.4 and equation 4.5 are set as 0.10 and 0.20 respectively to generate the prediction ellipses of 90% and 80% confidences respectively. The effect of the presence of outliers in the window samples is directly proportional to the number of residuals falling beyond the prediction ellipses. Thus the outliers falling beyond the prediction ellipses are the influential outliers which significantly affect the prediction accuracy. Figure 4.4 presents the prediction ellipses generated for job submission trend of Job 1 submitted by User 1 in all the three Windows. The statistics of Pearson correlation coefficient analysis for job arrival interval of Job 1 of User 1

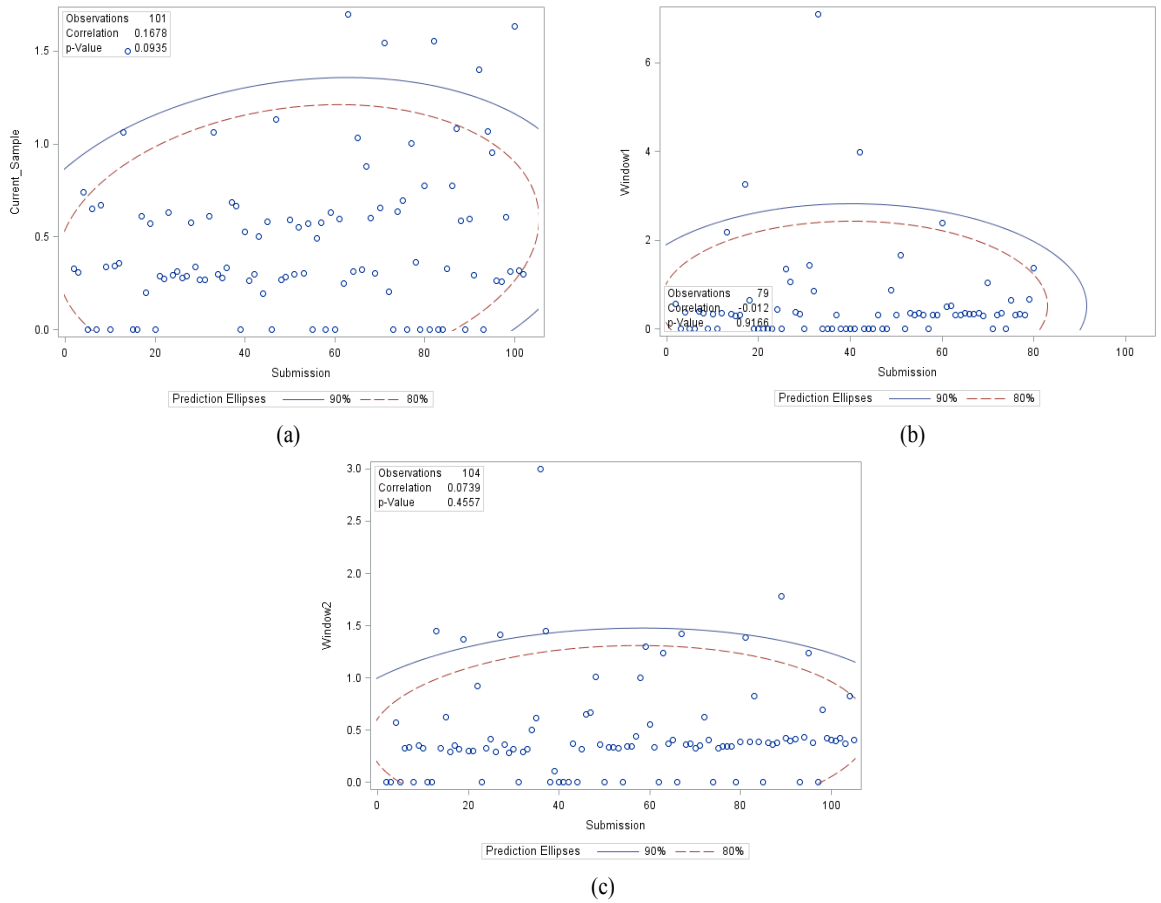


Figure 4.4. Prediction Ellipses of Job 1 of User 1 (a) Wc (b) W1 (c) W2

are presented in Table 4.4, along with the analysis results of the prediction ellipses. From Figure 4.4 and Table 4.4, it is evident that the submission interval of Job 1 of User 1 in W_c and W_2 is exhibiting a positive correlation and W_1 is exhibiting a negative correlation. From the prediction ellipses, a close correlation is evident between W_c and W_2 in terms of the prediction confidences and the presence of residuals, which is further validated by the Pearson correlation coefficient. Thus it can be concluded that the trend of Job 1 of User 1 in W_2 is exhibiting a more correlated behaviour with those in W_c than those of W_1, insisting that Job 1 of User 1 predominantly satisfies time-of-the-day periodicity effect. Based on this preliminary analysis for periodicity, the predictor relies on W_2 (validated by the sample validator) for the purpose of further training the most suitable historical sample into the predictor.

4.4.6 Stationarity Test

Figure 4.5 presents job submission trend, auto-correlation (ACF) and partial-autocorrelation functions (PACF) estimated by the ADF test for the original series of job submission time. Figure 4.5 shows a gradual decaying ACF function and also a strong first lag in the PACF with no other significant lags. Table 4.5 presents the ADF statistics of the stationarity test for Job 1 of User 1. Tau is the test statistics of the ADF unit root test with a standard mean in the data points. The ADF test statistics leads to the inference that job submission trend is non-stationary since the P value is greater than 0.05, the null-hypothesis cannot be rejected and so the trend of job submission trend is non-stationary.

Table 4.5. ADF test for Job submission time

Type	Value
Tau (Single Mean)	1.56
Tau (Trend)	-1.58
P value (Single mean)	0.9994
P value (Trend)	0.7929

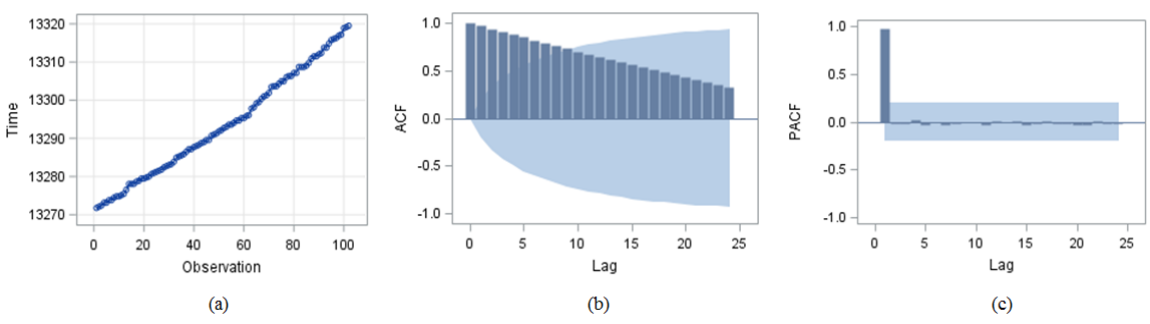


Figure 4.5. ADF Test for Job Submission Time (a) Submission Trend (b) ACF (c) PACF

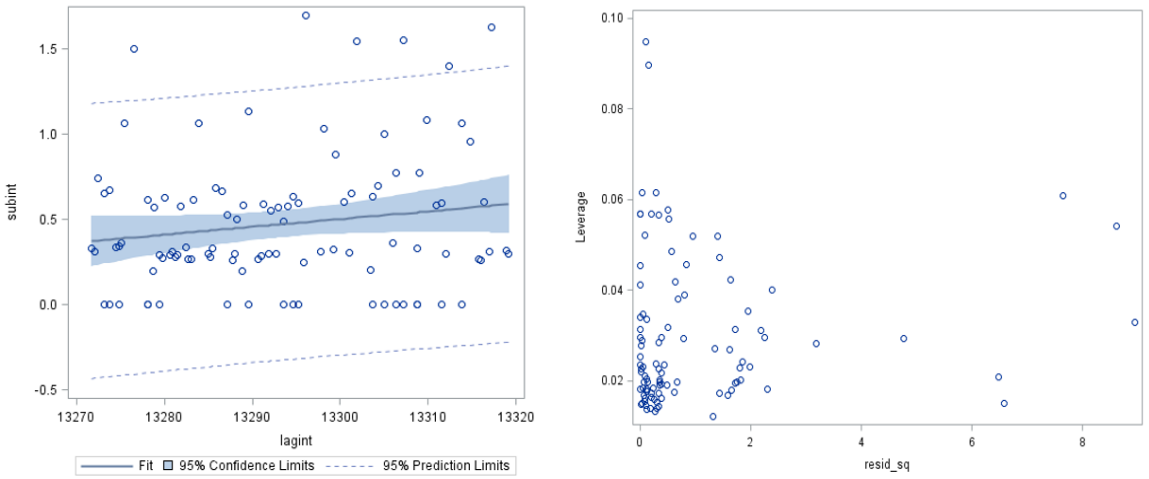
Table 4.6. Influential Outliers in the Submission Interval Trend of Job 1 of User 1

Submission	Time	Submission Interval	Cook Distance > 4/102	Weight
14	13278.06	1.50030	0.10112	0.33012
63	13297.82	1.69801	0.16488	0.35201
82	13308.70	1.55352	0.04620	0.36555
92	13313.76	1.39880	0.04795	0.41859
100	13318.86	1.63076	0.16428	0.31991

4.4.7 Outlier Suppression

Figure 4.6 illustrates the regression fit plot for job submission trend of Job 1 of User 1 in W_c, fitted with a 95% confidence and prediction limits for the submission interval trend along with the leverage-to-residual square plot. It can be easily observed that the submission interval trend of Job 1 of User 1 is heavily influenced by the presence of a significant number of outliers, which could lead to inaccurate prediction of job submission trend. The submission interval trend is suffering from both high leverages and large residuals, necessitating the need for suppressing the influence of the outliers with robust regression.

Table 4.6 further presents the observations of Job 1 of User 1 suffering significantly from the presence of outliers in the submission interval trend in W_c identified by the robust regression. The level of influence of the outliers over the data points are determined by the Cook's distance which is a combined measure of leverage and residuals present in the observation. Apart from the observations presented in Table 4.6, presence of all the outliers have a considerable impact



(a) (b)

Figure 4.6. Regression for Job 1 of User 1 (a) Fit plot (b) Leverage to residual square plot

on the overall prediction accuracy depending on their distance from the mean. Thus the predictor considers suppressing the influence of all the outliers by adjusting their weights depending on their corresponding influence on prediction accuracy.

Now the predictor subjects the prediction sample of Job 1 of User 1 in W_c to robust regression as described in section 4.3.3.2. Robust regression is now applied on the prediction sample by iterated re-weighted least squares based on the Huber weights. Observations highly suffering from the outliers are assigned with smaller weights by robust regression in order to suppress their influence on the prediction accuracy. After applying robust regression to supress the outliers in the prediction sample of Job 1 of User 1 in W_c, the outlier proportions has been reduced to 0.0594%. Table 4.6 further presents the weights assigned to the observations dominated heavily the influential outliers. It can be observed that observations influenced by severe outliers are assigned with lower weights through to a weight of one is assigned to the observations under minimum outlier influence.

Finally, the prediction sample is sorted ascendingly based on the weights assigned to the observations by robust regression for the purpose of restraining the influence of the outliers contained in the prediction sample.

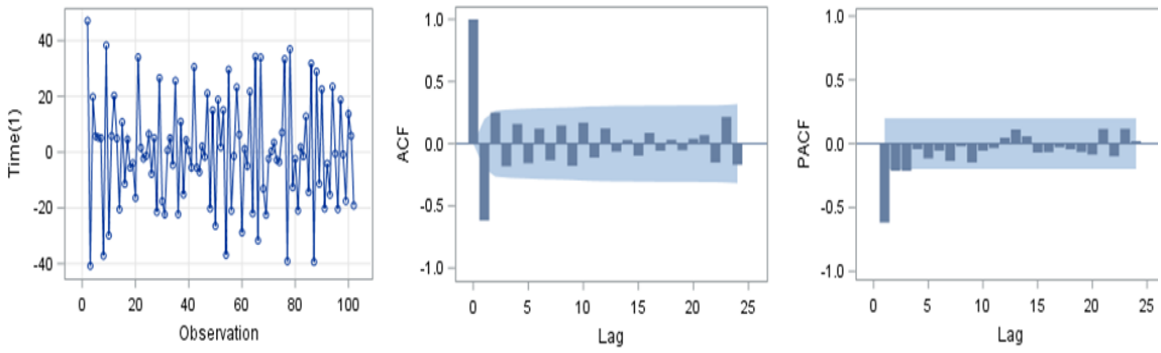


Figure 4.7. ADF Test for Differenced Variable (a) Trend (b) ACF (c) PACF

Table 4.7. ADF test for the Differenced Variable

Type	Value
Tau (Single Mean)	-21.45
Tau (Trend)	-21.33
P value (Single mean)	<.0001
P value (Trend)	<.0001

4.4.8 ARIMA Model Selection

Since the original variable of job submission time for Job 1 of User 1 is non-stationary, the predictor now computes the differenced variable of submission interval for Job 1 of User 1. Table 4.7 presents the test statistics of stationarity for the differenced variable after subjecting to robust regression. Figure 4.7 presents the trend, ACF and PACF statistics of the differenced variable for job submission interval after robust regression. From Table 4.7, the Tau value for the differenced variable is significantly negative with a very small p value, so that the null hypothesis is rejected and there is alternative hypothesis. It can also be observed that there is no gradual lag in the ACF function, and the PACF function is exhibiting a first positive significant lag followed with a second significantly negative lag and no other lags are significant. All these statistics conclude that the differenced variable which is the submission interval of job 1 of User 1 is now stationary. Furthermore, the first positive lag of ACF insist that there is an AR 1 process existing in this stationarised series. It is clear that the differenced variable is more suitable for predicting the future trend because of the degree of stationarity.

Now, the predictor estimates different ARIMA models for the purpose of training the predictor with the most appropriate predictor variables based on the trend of the stationarised differenced variable. Table 4.8 presents the estimates for various ARIMA models for job submission interval of Job 1 of User 1. From Table 4.8, it can be concluded that an ARIMA (1,1,1) model

Table 4.8. ARIMA Model Estimates for Job Submission Interval

ARIMA (p,q,d)	**Conditional Least Square Estimate**			
	Parameter	**Estimate**	**AIC**	**SBC**
ARIMA (1, 1, 0)	AR 1,1	0.57905	65.37405	70.60429
ARIMA (0, 1, 1)	MA 1,1	-0.36224	84.38915	89.61939
ARIMA (1, 1, 1)	AR 1,1	0.65974	31.35396	39.19932
	MA 1,1	1.00000		

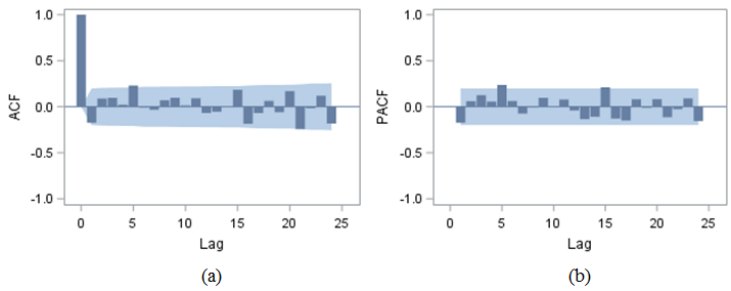


Figure 4.8. Predictor Plots of ARIMA (1,1,1) (a) ACF (b) PACF

is best suitable for predicting the trend of Job 1 of User 1, since both the AIC and SBC values are smaller than those of the other two models. Further the lags in the residual correlations of both ACF and PACF for ARIMA (1,1,1) are non-significant with only a first significant positive lag in the ACF plot for confidence, as shown in Figure 4.8.

4.4.9 Optimised Job Trend Prediction

The predictor choses the ARIMA (1,1,1) model for training the predictor for the trend of Job 1 of User 1. After choosing the ARIMA model, the anticipated number of submissions and the session duration for Job 1 of User 1 is estimated by the forecasting window to set the forecast window of the predictor based on section 4.3.3.3, explained as follows. Job 1 of User 1 spans across a total of 102, 105 and 21 submissions in W_c, W_2, and W_r respectively. The session duration of Job 1 of User 1 in W_c, W_2, and W_r are 47.71, 47.25 and 11.72 respectively. In other words, User 1 has submitted Job 1 for 102 times across a duration of 47.71 minutes in W_c. From W_r, predictor obtains the composite $V_r = \{21, 11.72\}$, where 21 is the number of submissions and 11.72 is the submission session duration for Job 1 of User 1 in W_r. Based on the computations presented in section 4.3.3.3, the window forecaster computes the anticipated number of submissions and the session duration for Job 1 of User 1 as a composite with the absolute values of $V_p = \{23, 11.72\}$ for the prediction window W_p.

Now the predictor predicts the submission interval of Job 1 of User 1 using the chosen ARIMA (1,1,1) model based on V_p. Figure 4.9 presents the ARIMA forecast for Job 1 of User 1 for the prediction window W_p. This is a linear forecast with a 95% confidence window for the submission interval of Job 1 of User 1 for the anticipated 23 submissions in W_r. It can be observed that the 95% confidence window spans across a significant interval bounds across the linear forecast.

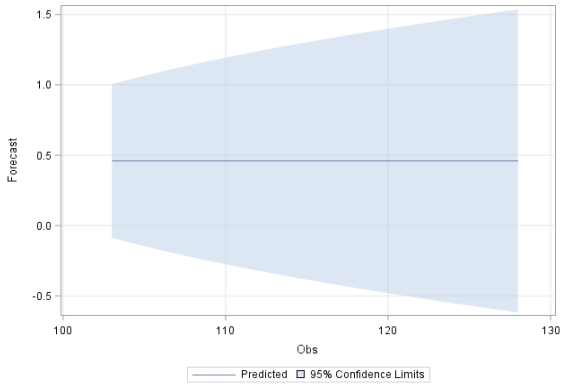


Figure 4.9. ARIMA Forecast for Job 1 of User 1

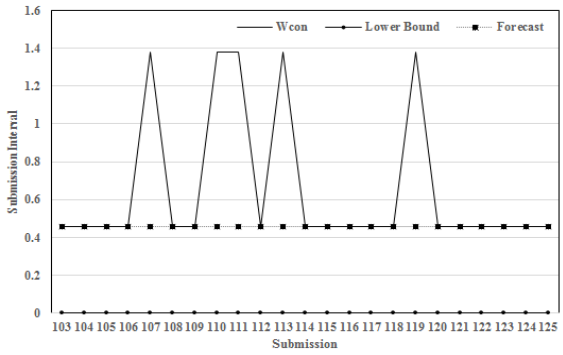


Figure 4.10. Optimised Confidence Window for Job 1 of User 1

InOt-RePCoN further optimises this 95% confidence interval of the ARIMA forecast in order to improve the prediction accuracy and to reduce the interval bounds around the linear forecast. Figure 4.10 presents the optimised confidence interval based on section 4.3.3.5. It can be observed that the ARIMA forecast is further optimised after nullifying the negative lower bounds with an optimised upper confidence interval. The upper confidence interval W_{con} is the optimised forecast with an error margin of the forecast expected with the bounds of W_{con} and zero mean lower bound, owing to the presence of outliers in the prediction sample. This insists that the future submission interval of Job 1 of User 1 is expected within the bounds of W_{con} and the zero mean lower bound limits.

4.5 Performance Evaluation

The efficiency of the proposed prediction model is evaluated by the measure of the forecast accuracy against the actual trend of user behaviours in terms of the anticipated number of submissions, session duration and the arrival trend for the target jobs from users. The efficiencies of our proposed model is evaluated under various scenarios of business hours in order to demonstrate the dependency of InOt-RePCoN under dynamic scenarios of Cloud Computing.

4.5.1 Week Day Off-Peak Time Prediction

4.5.1.1 Sample Containing Influential Outliers

The sample trained for validation in section 4.4 contains data from Wednesday during the business hours of 12 am to 1 am for the purpose of predicting the expected user behaviour from 1 am to 2 am, which is an off-peak business hour during a week day. Figure 4.11 illustrates the accuracy of the forecasting window, where the actual observations of the number of job submissions and the session duration are plotted against the forecasted values for Job 1 of User

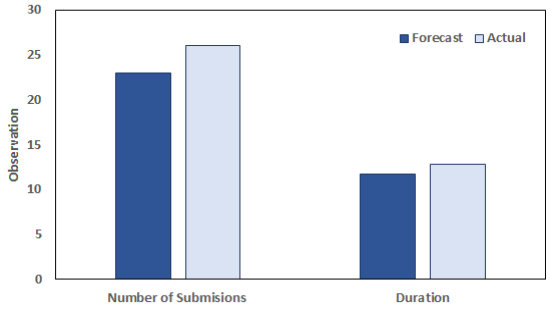


Figure 4.11. Forecast Window Observation for Job 1 of User 1

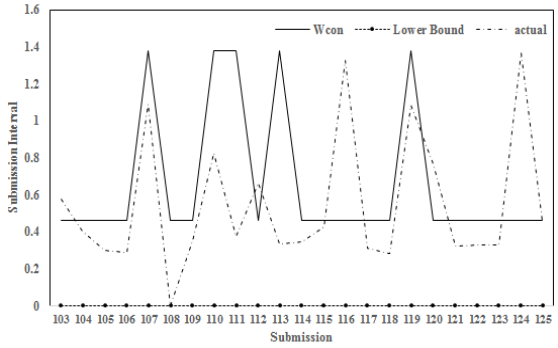


Figure 4.12. Optimised Confidence for Job 1 of User 1

1. It can be observed that an accuracy of 88.46% is achieved for the number of job submissions and an accuracy of 91.13% is achieved whilst predicting the session duration.

Figure 4.12 presents the optimised confidence interval of the InOt-RePCoN fitted with the actual trend of submission for Job 1 of User1. Since the prediction sample is heavily affected by the presence of influential outliers, the future trend is expected to be within the bounds of W_{con} and the zero mean lower bound error margin. It can be observed that 18 observation points out of 23 of the actual submissions are within the confidence interval bound limits predicted by the proposed framework. Thus the proposed framework achieves an accuracy of 78.26% whilst predicting the submission interval trend of Job 1 of User 1. The confidence bounds optimised by the proposed model significantly reduces the 95% interval of the ARIMA forecast, thus reducing the bound limits with reliable level of accuracy.

4.5.1.2 Samples Without Influential Outliers

The efficiency of the proposed model are further evaluated whilst forecasting user behavioural trend with the sample containing no influential outliers during off-peak time. Now the prediction model is trained with the data obtained from Wednesday during the business hours of 3 am to 4 am for the purpose of predicting the expected user behaviour from 4 am to 5 am. The rule miner forms W_1 and W_2 accordingly.

A total of 57 users are observed in W_c, with 39, 5, 4, and 9 users are assigned with a predictability weight of level 3, 2, 1 and 0 respectively by the rule miner. A randomly chosen user (named User 2) has been set as the target user for this forecast. User 2 has submitted a total of 3 job types across 36 submissions. Both User 2 and his three job types are assigned with a predictability weight of level 3. Now the objective is set to the predictor to forecast the trend of all the three job types belonging to User 2. Robust regression is not performed by the predictor in spite of the minimal presence and marginal influence of the outliers in job submission trend of the prediction sample. The parameter for the forecast window is computed as an absolute composite of V_p = {36, 56.89}, which means a total of 36 submissions are anticipated from User 2 in 56.89 minutes in W_p.

Figure 4.13 shows the ARIMA forecast for the submission interval trend of User 2 bounded with the 95% confidence interval, along with the forecast fitted with the actual observation. It can be observed that the occurrence of peaks and valleys of the forecast is closely correlating with the actual trend, but the forecast-to-actual values are still not accurately optimised. Furthermore, the 95% confidence of the ARIMA forecast is spanning across the forecast with a larger amplitude, which reduces the crispness of the forecast results.

Figure 4.14 illustrates the number of submissions and the session duration estimated by the forecasting window against the actual values observed. It can be observed that an accuracy of

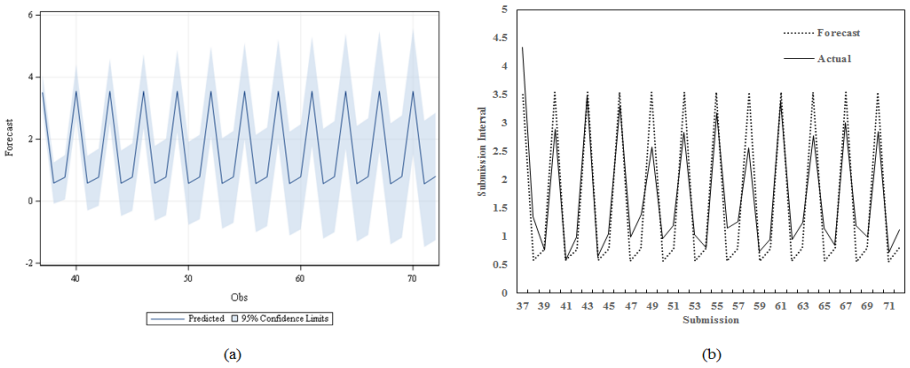


Figure 4.13. Forecast for All Jobs of User 2 (a) ARIMA forecast (b) Forecast vs Actual trend

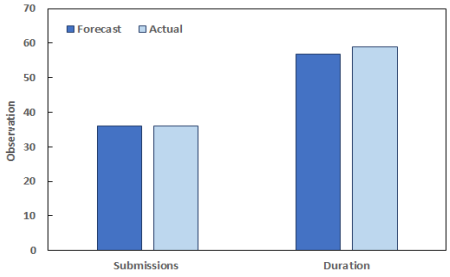


Figure 4.14 Forecast Window Observation for All Jobs of User 2

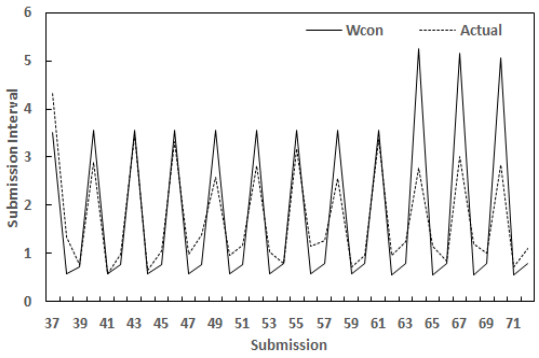


Figure 4.15. Optimised Confidence for All Jobs of User 2

100% is achieved in forecasting the anticipated number of submissions and an accuracy of 96.71% is achieved in forecasting the session duration for the trend of User 2. Figure 4.15 depicts the optimised confidence interval W_{con}, fitted with the actual submission trend of User 2. It can be observed that most of the actual trend of the submission interval of User 2 is closely correlating with the optimised confidence interval, with W_{con} achieving an accuracy of 73.23% delivered by the proposed prediction model. It is also evident that the optimised confidence interval of the proposed model significantly enhances the prediction accuracy of the initial ARIMA forecast.

4.5.2 Week Day Peak Time Prediction

Further, the efficiency InOt-RePCoN is evaluated whilst predicting the expected user behaviours under peak business hours during a week Day. Now, the objective is to forecast user behaviour from 11 am to 12 pm on a Monday morning. A total of 58 users are comprised in W_c, with 43, 4, 7, and 4 users are assigned with a predictability weight of 3, 2, 1, and 0 respectively. User (named User 3), with most number of submissions in W_c, has been set as the target user for this forecast. User 3 has submitted a total of 2 job types named Job 1 and Job 2 respectively, across a total of 75 job submissions in W_c. Both these two job types has been assigned with a predictability weight of level 3 by the rule miner, insisting an increased predictability. The event proportion for Job 1 and Job 2 is 60% and 40% respectively. Now predictor is set with an objective of forecasting the future trend of Job 1 of User 2. A submission interval trend of Job 1 of User 3 characterise a minimal influence of outliers in W_c. After generating the prediction ellipses, a close correlation is evident in the submission trend of Job 1 of User 3 between W_c and W_1. Thus the dual-effect window shows better correlation and exhibits a better trend of predictability for Job 1 of User 3. This is because the dual effect window contains the historic sample from the same representative Monday from the previous week.

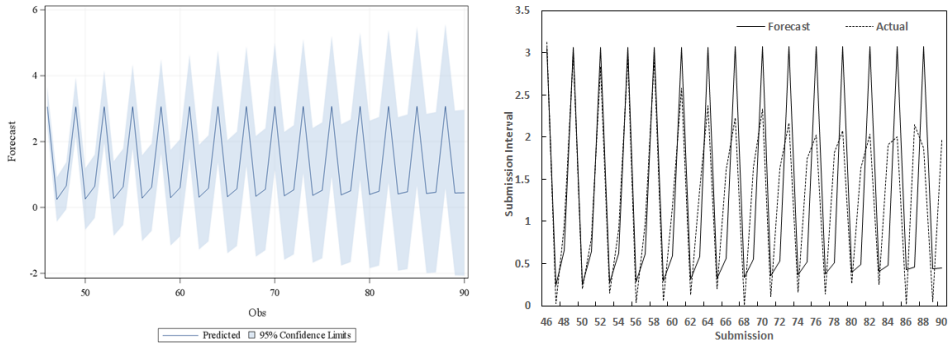


Figure 4.16. Forecast for Job 1 of User 3 (a) ARIMA forecast (b) Forecast vs Actual trend

But W_2, the time-of-the-day effect window, consists of sample from a Sunday (previous day of the current sample). Since W_2 contains week-end trend it is loosely correlating with the trend in W_c containing week-day trend. Thus the samples in W_1 is chosen and validated for similarity by the validator for the purpose of further training into the predictor. The parameter for the forecast window is computed as an absolute composite of V_p = {45, 58.09}, which means a total of 45 submissions are anticipated for Job1 of User 3 in 58.09 minutes in W_p. Figure 4.16 presents the ARIMA forecast output for the submission interval trend of Job 1 of User 3 with

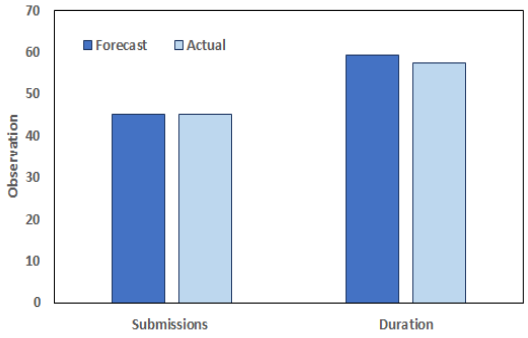


Figure 4.17. Forecast Window Observation for Job 1 of User 3

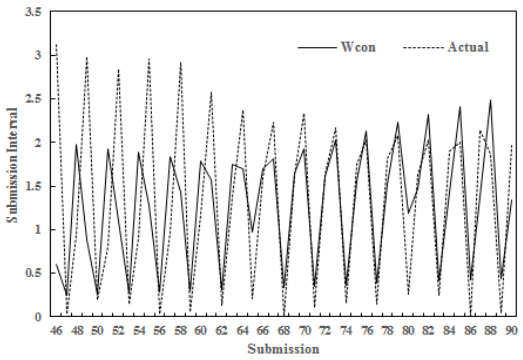


Figure 4.18. Optimised Confidence for Job 1 of User 3

the 95% confidence interval, alongside the forecast fitted with the actual observation of submission interval. Again, the occurrence of peaks and valleys of the forecast is closely correlating with the actual trend, but the forecast-to-actual values are still not accurately optimised. Figure 4.17 illustrates the number of submissions and the session duration predicted by our forecasting window against the actual values. It can be observed that an accuracy of 100% is achieved in forecasting the anticipated number of submissions and an accuracy of 96.76% is achieved in forecasting the session duration for the trend of Job 1 of User 3. Figure 4.18 depicts the optimised confidence interval fitted with the actual submission trend for Job 1 of User 3. It can be observed that most of the actual trend of the submission interval of Job 1 of User 3 are within the bounds of the optimised confidence interval, with W_{con} achieving an accuracy of 73.17% delivered by InOt-RePCoN.

4.5.1 Week-End Peak Time Prediction

Now, the objective is to forecast user behaviour from 11 am to 12 pm on a Sunday morning. A total of 55 users are comprised in W_c, with 41, 3, 7, and 4 users are assigned with a predictability weight of 3, 2, 1, and 0 respectively. The target is to predict the behaviours of a randomly chosen (named) User 4 who has submitted a total of 46 jobs in W_c.

A significant variance is evident in the submission behaviours of User 4 in W_c and are influenced by a significant proportions of outliers. After generating the prediction ellipses, a close correlation is evident in the submission trend of User 4 between W_c and W_2. Thus the time-of-the-day window shows better correlation and exhibits a better trend of predictability for User 4. Thus the samples in W_2 is chosen and validated for similarity by the validator for the purpose of further training into the predictor. The parameter for the forecast window is computed as an absolute composite of $V_p = \{25, 38.16\}$, which means a total of 25 submissions are anticipated from User 4 in 38.16 minutes in W_p. Figure 4.19 presents the ARIMA forecast

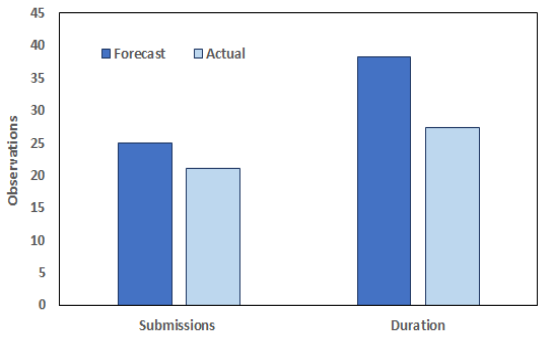


Figure 4.19. Forecast Window Observation for All Jobs of User 4

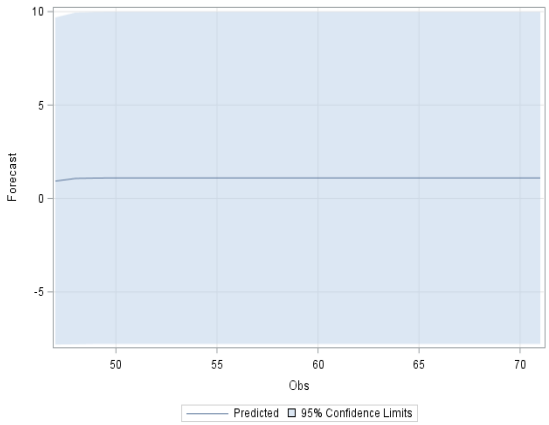


Figure 4.20. ARIMA Forecast for All Jobs of User 4

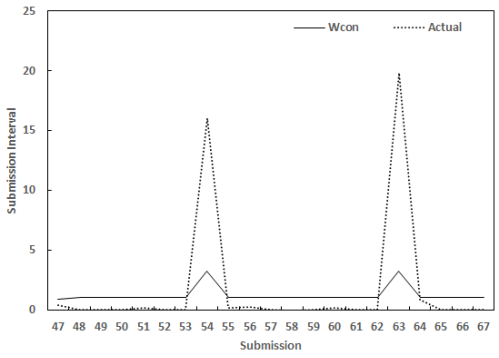


Figure 4.21. Optimised Confidence for All Job of User 4

output for the submission trend of User 4 with the 95% confidence interval. In this case, ARIMA present a linear forecast due to the presence of the influential outliers in the prediction sample. However linear forecast may not present a precise prediction inferences for the providers for resource management.

Figure 4.20 illustrates the number of submissions and the session duration estimated by the forecasting window against the actual values observed. It can be observed that an accuracy of 81.96% is achieved in forecasting the anticipated number of submissions and an accuracy of 60.21% is achieved in forecasting the session duration for the trend of User 4. The increased error percentage in forecasting the submission duration of User 4 is attributed to the presence of influential outliers. Figure 4.21 depicts the optimised confidence interval fitted with the actual submission trend for all jobs of User 4. It can be observed that most of the actual trend of the submission interval of User 4 are within the bounds of the optimised confidence interval, with W_{con} achieving an accuracy of 90.47% delivered by InOt-RePCoN.

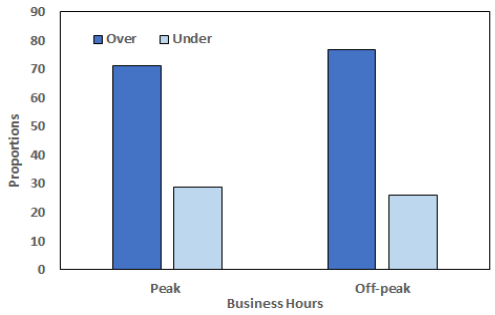


Figure 4.22. Over-to-under Prediction Ratio of Submission Interval

4.5.2 Reduction of Under-Prediction

From the perspectives of benefitting both users and the providers, InOt-RePCoN is aimed at reducing the probabilities of under-prediction, since it would cause a more disastrous effect on the overall energy efficiency. Figure 4.22 presents the over-to-under predicted ratio whilst forecasting the submission interval of jobs from users during peak and off-peak business hours. It can be observed that the proposed prediction model is effective in reducing the number of under-predictions, witnessed only at an average of 27.45% in comparison to the over-predicted observations witnessed at an average of 74.01%. Thus it can be concluded that the proposed prediction model is effective in reducing the probabilities of under-prediction.

4.5.3 Forecasting Efficiency of InOt-RePCoN

This section is aimed at demonstrating the forecast efficiency of InOt-RePCoN, by comparing the forecast accuracy of the proposed model against existing techniques with similar objectives of InOt-RePCoN which is to predict the intensity of the incoming job submissions. Firstly the accuracy of the statistical approach adopted by InOt-RePCoN has been compared with SPAR (Spare Periodic Auto Auto-Regression) which is an autoregressive based prediction model, an approach of predicting load on single VM based on HMM using single time series and a HMM based co-clustering technique of predicting workloads at group levels using multiple time series respectively, since all these techniques are aimed at predicting job arrival trend. This

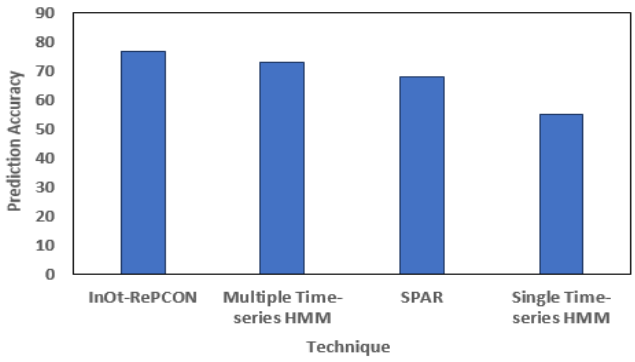


Figure 4.23. Confidence Optimisation Accuracy

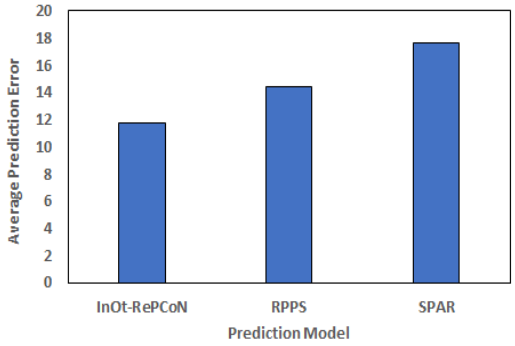


Figure 4.24. Prediction Efficiency

evaluation is intended to demonstrate the efficiency of the integrated novel Confidence Optimisation framework. Figure 4.23 illustrates the prediction accuracy of InOt-RePCoN against the compared techniques. It can be observed that the proposed model exhibits an average prediction accuracy of 76.73%, with accuracy of the multiple time series co-clustering HMM, Spare Periodic Auto-Regression, and single time series HMM technique being claimed at 73%, 68% and 55% respectively. By delivering a reliable level of accuracy for the confidence window, service providers can expect the consecutive submission of the corresponding jobs from users within the window intervals predicted by InOt-RePCoN. Secondly, the estimation accuracy of InOt-RePCoN is evaluated whilst forecasting the anticipated number of submissions and the session duration for users by the way of comparing the average prediction error of the proposed prediction model with the existing RPPS technique based on simple ARIMA, and SPAR which is based on periodic Auto-Regression, both aimed at forecasting the incoming job tend based on different adoptions of Auto-Regression.

The error percentage of the proposed prediction model is presented as a combination of the average prediction error whilst forecasting the anticipated number of submissions and the session duration. The prediction error of the other two models are presented as a combination of the average under and over prediction errors whilst forecasting future workload intensity. Figure 4.24 depicts the average prediction error of InOt-RePCoN, RPPS and the SPAR model respectively, it is evident that the proposed prediction model exhibits a better prediction accuracy with an average prediction error of 11.79, than the RPPS and SPAR models with their average prediction error being claimed at 14.39 and 17.68 respectively. With both the number of submissions and the session duration being estimated with a reliable level of accuracy, service providers can achieve an effective scaling of the server resources based on the anticipated intensity of the incoming job trend.

4.6 Summary

This chapter proposed InOt-RePCoN, a novel prediction model for forecasting the trend of user behaviours in large-scale Cloud environments. The proposed framework is expected to benefit the service providers in two different perspectives. Firstly, estimating the expected number of submissions and session duration for users helps the service providers to achieve an optimum resource management by scaling up/down the server resources in accordance with the window forecast. For instance, accurately predicting the peaks and valleys of the workload levels from users helps in effective switching of server resources for energy efficient server management. Secondly, the optimised confidence interval forecast for the submission interval trend of users provides useful inferences about the arrival frequency of jobs from users. This helps with an initial preparation for the scheduling and job allocation management in accordance with the arrival rate of workloads from users. The proposed model exhibits a characteristic reduction in the probabilities of under-predictions which helps to avoid the energy expenditures incurred by the early provisioning of resources for the anticipated job submissions. From the performance evaluations conducted based on real Cloud traces during different business hours, it can be concluded that the proposed prediction model achieves reliable level of accuracy in predicting the future job submission trend of users. One notable complexity of the model could be attributed to the need for storing the historical samples in the database, since the proposed model relies on historical samples to optimise the confidence interval of the ARIMA forecast. But the complexities in storing the historical samples is reduced to a minimum, since the proposed model exploits historical samples only from the previous one week. Since the proposed model relies on the inherent periodicity among users and their workloads for computing their predictability weights and further optimising the prediction confidence, the prediction of user behaviours without any degree of periodicity, essentially brand new users, cannot be further optimised by the proposed model. Though, recoding the traces of such users over a period may facilitate optimising their prediction confidence.

5 Optimised Resource Provision Framework

5.1 Overview

In the Cloud Computing service model, user submitted jobs are scheduled and processed in the VMs or LXCs deployed on the physical server resources. While forecasting the anticipated job arrival trend in terms of quantity and frequency has been proposed in the previous chapter, this chapter focuses on predicting the anticipated behaviours of the arrived jobs in terms their resource consumption levels and straggling behaviours during the actual execution. In general, the presence of stragglers may significantly affects the resource estimation accuracy, and such tasks should be treated with special consideration for resource prediction analytics. Tasks within a single job may exhibit increased fluctuations in their resource consumption levels, which might cause over-estimation of non-stragglers and under-estimation of stragglers. Resource prediction analytics should necessarily integrate a task classification analysis to pro-actively classify the stragglers and non-stragglers to optimise their estimated level of resource requirements. To this end, this chapter proposes an estimation of resource requirements of jobs, integrated with a classification framework to forecast straggling behaviours and a resource optimisation model to postulate the most optimum level of resource provisioning for tasks. Important contributions of this paper include the following.

a) Empirical analysis of the execution behaviours of tasks within jobs. Task behaviours within their respective jobs are analysed from three different perspectives. Firstly the actual number of resources provisioned and utilised in terms of CPU cores and memory bytes for every single task within a single job have been analysed to expose the proportional possibility of energy conservation for a given job execution. Secondly, the presence of energy-aware stragglers within jobs have been empirically analysed and their impacts on resource provisioning levels and incurred energy impacts have been exposed. Finally, the execution trend of tasks within jobs have been analysed to exhibit the heterogeneity among task execution within a single job, this includes uncovering the relationship between CPU usage rate and task duration, along with studying the trend of task termination within jobs.
b) An analytics approach has been proposed to estimate the resource consumption levels of the arrived tasks and further to identify energy-aware stragglers within a single job execution along with optimising the resource provisioning level for tasks to avoid resource idleness and task terminations.

5.2 Resource Profile Analytics

5.2.1 Methodology

A generalised description of jobs, tasks and user have been discussed earlier in Chapter 4. Now, the execution profile of jobs can be defined as a composite E_j, consisting of the submission time t_s, job index J_i, job name J_n, number of encompassed tasks n_t, resource levels $c_{(c,m)}$ (c and m are the CPU and memory resources accordingly), and job scheduling index J_{sh} as shown in equation 5.1.

$$E_j = \{t_{sj}, J_i, J_n, n_t, c_{(c,m)}, J_{sh}\} \tag{5.1}$$

Since every task within jobs are processed individually, execution profiles of tasks encompassed within jobs can be defined as a composite E_t, consisting of the submission time t_{st}, task index T_i, job index J_i to which task belongs, resource levels $c_{i(c,m)}$ of the i^{th} task within job J_i and task priority T_p as shown in equation 5.2.

$$E_t = \{t_{st}, T_i, J_i, c_{(c,m)i}, T_p\} \tag{5.2}$$

When the resource requirements, job priority and the number of tasks encompassed within jobs are explicit, task length t_j can be calculated for the historical execution instances as the difference between task completion time t_f and the time of task scheduling t_{sh}, as shown in equation 5.3.

$$l_j = t_f - t_{sh} \tag{5.3}$$

In spite of exposing task heterogeneity, jobs encompassing different number of tasks are presented in this chapter as representatives of different job behaviours and task numbers, as

Table 5.1. Job Profile Representation

Job Name	Encompassed Tasks	Periodicity	
		Day-of-the-week	Time-of-the-day
Job 0	50	Yes	Yes
Job 1	100	Yes	Yes
Job 2	200	Yes	Yes
Job 3	182	Yes	No
Job 4	488	Yes	Yes
Job 5	1050	Yes	Yes

shown in Table 5.1. It is important to extract information from the successfully executed profile of tasks, since profiles of terminated tasks cannot identically reflect the resource levels of tasks. The trace logs present a measurement period of 300 seconds when reading the execution parameters. Thus the total number of resources consumed by a single task and its execution duration is given by the summation of the all measurement samples of the corresponding task, as shown in equation 5.4 and equation 5.5.

$$R_T = \sum_{i=1}^{n} r_i \tag{5.4}$$

$$L_T = \sum_{i=1}^{n} l_i \tag{5.5}$$

where, R_T and L_T are the total number of resources consumed and task length for task T respectively, n is the total number of measurement sample, r_i and l_i are the resource consumption and duration of the corresponding task during the i^{th} sample period of task T. Hence a given task T would require a minimal level of R_T resources and can be expected to run for a minimum duration of L_T when provisioned with R_T.

5.2.2 Resource Provision and Consumption

The resources provisioned at a level more than R_T are over-commitment of resources. While the idle resource analysis presented in Chapter 3 has quantified the total proportion of idle resources on a daily basis over the entire days of analysis, this section presents energy impacts of over-commitment of every individual task within the studied jobs. Schedulers usually allocate the incoming tasks onto isolated containers with a pre-defined level of CPU, memory and disk space resources for the LXCs or VMs to consume physical resources. This predefined level of resource provision is usually the maximum level of allowed resources for the LXCs or VMs. Whilst is it commonly argued that the level of provisioned resources far exceed the actual requirements of tasks, this section projects the scope for proportional reduction in the actually provisioned resource levels for the deployed containers which would then facilitate the physical machine to accommodate more LXCs or VMs for the purpose of reducing the number of active physical resources. Figure 5.1 presents the statistical observations of provisioned and utilised resources in terms of the CPU cores and memory bytes for the studied. It can be observed that both the CPU and memory resources are commonly over provisioned. Whilst the provisioned memory resources have been utilised to a reasonable margin, CPU utilisation trend is leaving a significant proportion of the provisioned resources unutilised. Thus it is clear that the deployed LXCs and VMs are provisioned to consume extravagant amounts of physical

resources than actually required to process tasks being executed. Interestingly, jobs comprising different number of tasks behave differently in utilising the provisioned resources.

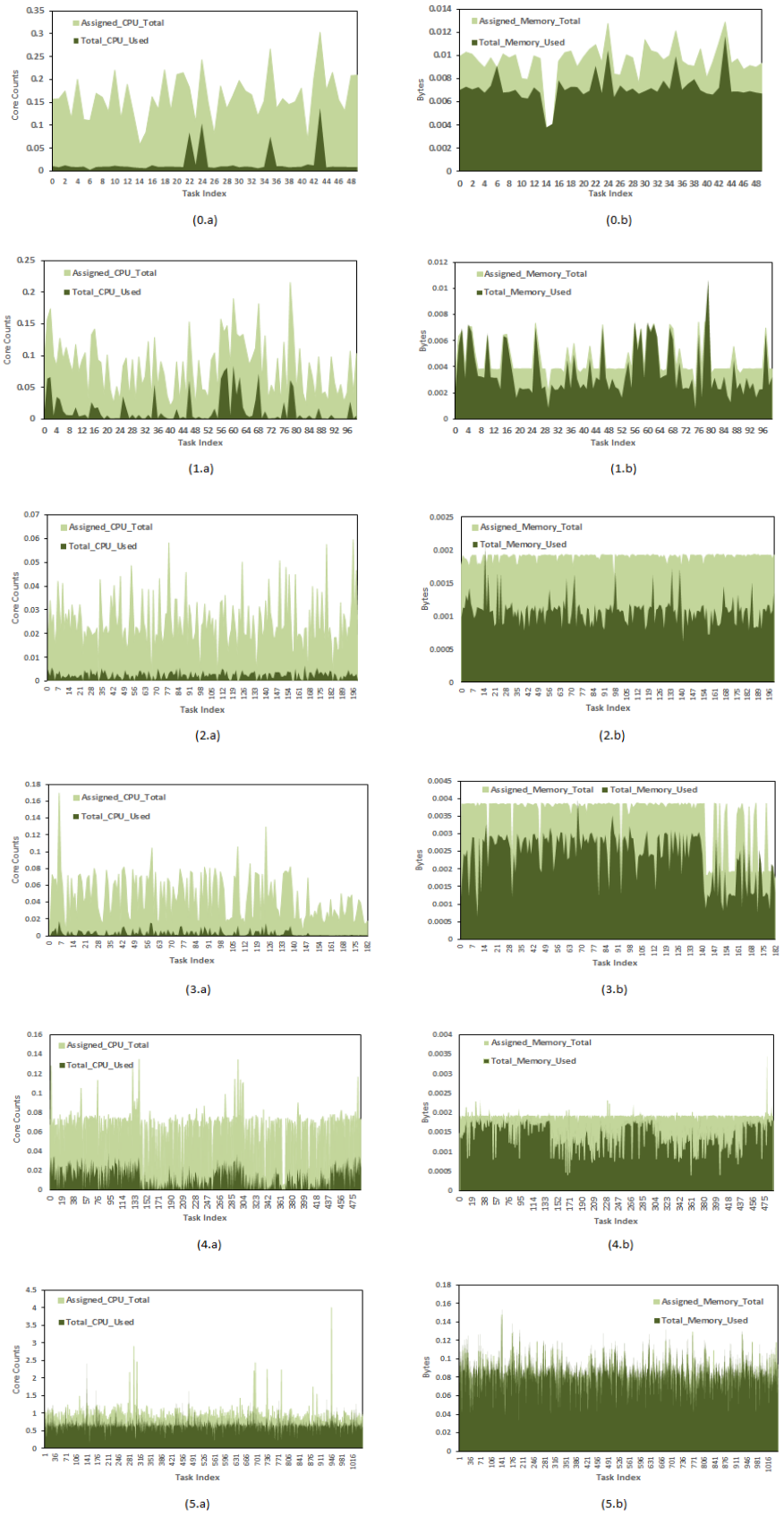


Figure 5.1. Resource Provision to Usage Pattern (a) CPU (b) Memory {(0) Job 0 (1) Job 1 (2) Job 2 (3) Job 3 (4) Job 4 (5) Job 5}

Table 5.2. Job Execution Statistics

Job Name	Job Duration	Average Task Duration	Total CPU Core counts
Job 0	4.33	3.43	0.8
Job 1	3.33	2.5	1.43
Job 2	1.65	0.96	0.5
Job 3	1.45	0.79	0.56
Job 4	4.48	2.7	6.4
Job 5	30	26.52	714.9

It can be observed that the CPU and memory resources are left unutilised at an average of 90.5% and 23.5% by Job 0, 87.8% and 24.5% by Job 1, 88.8% and 42.5% by Job 2, 91.6% and 31.2% by Job 3, 80.8% and 21.8% by Job 4 and 16.15% and 9.75% by Job 5 respectively. These observations further corresponds to the CPU and memory idleness observed for the entire month presented in Chapter 3. The lower proportions of the idleness in memory resources can be attributed to the fact that users tend to reserve more CPU for their prospective jobs, since CPU is obviously a more scarce resource than memory. It has also been signified [19] that the analysed trace log exhibits larger proportions of users requesting smaller amounts of memory and a only a fewer proportions of users requesting large amounts of memory resources, whilst the CPU resources have commonly been over requested by almost all users. Though not obvious, the trace log also includes storage-bound workloads which might also be the case for the fair utilisation profiles of the memory resources.

Job 5 have utilised the provisioned resource to a considerable margin than the other jobs, which further depicts job heterogeneity in resource consumption. Furthermore, job duration has been measured as the length of the longest running task within the corresponding jobs. Table 5.2 presents the observed statistics of job execution including the total job duration, average task duration, and the total core counts consumed across all tasks encompassed within the corresponding jobs. These observed statistics further proves job and task heterogeneity in terms of their execution profile. Though it is evident that jobs encompassing more tasks runs longer and consumes more resources, this trend is not absolutely linear. For instance Job 4 with 488 tasks runs for 4.48 minutes and consumes 6.4 core counts across all the encompassed tasks, but Job 5 comprising just more than twice as many as tasks of Job 4 exhibits a job duration of 30 minutes (nearly 7 times of Job 4) and consumes 714.9 core counts (nearly 122 times of Job 5).

Thus the encompassing number of tasks might not provide suffice inferences to characterise job duration and resource consumption levels. It can be postulated that every job should be uniquely treated for resource provision and further tasks encompassed within a single job also exhibits increased resource consumption diversity.

5.2.2.1 Distribution Analysis

To investigate job behavioural trend in utilising the provisioned resources, jobs are further subjected to a distribution analysis similar to the methodology presented in Chapter 3.

The distribution of the data trend of the provisioned and utilised resources have been evaluated against notable theoretical distribution such as Normal, Lognormal, Exponential, Gumbel, Gamma, Weibull etc., and further the best fit distribution is presented for the actual data trend in terms of their CDFs within every studied jobs. Firstly, the amounts of CPU and memory resources provisioned has been evaluated against the trend of actual resource consumption for every individual task within their respective jobs. Secondly, task duration within a single job has been analysed to observe task length heterogeneity within jobs. Figure 5.2 and 5.3 presents the CDF with the best fit distribution of the observed statistics including assigned-to-utilised CPU and assigned-to-utilised memory for every individual tasks within Job 0 and Job 5 respectively.

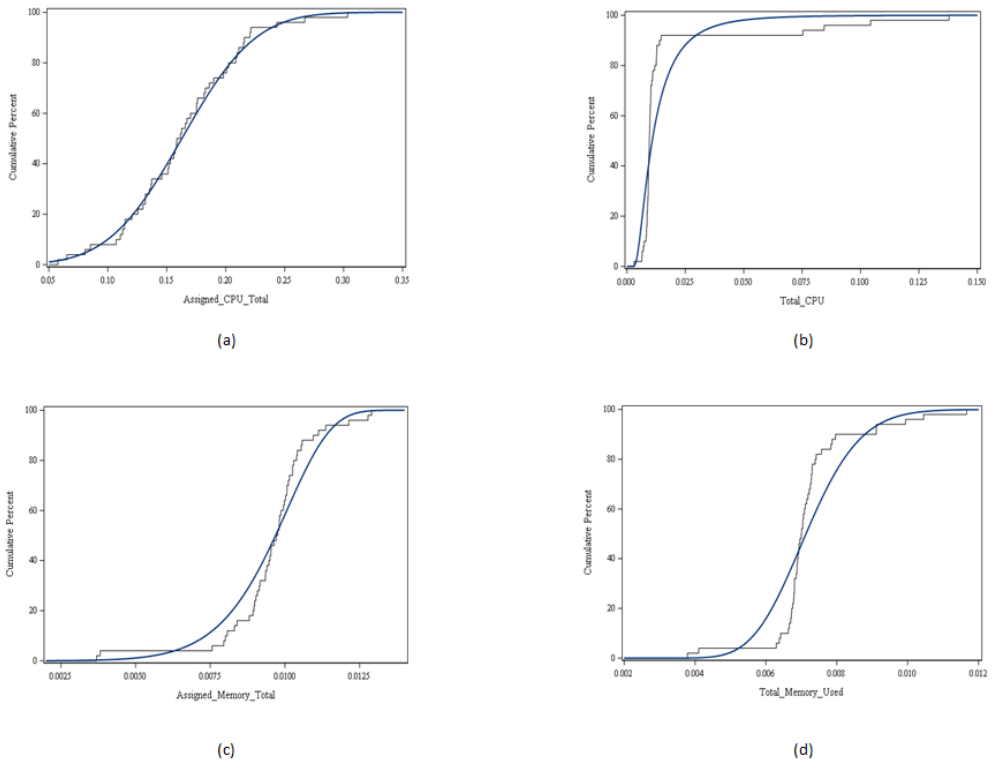


Figure 5.2. CDF of Job 0 (a) Assigned CPU (b) CPU Consumed (c) Assigned Memory (d) Memory Consumed

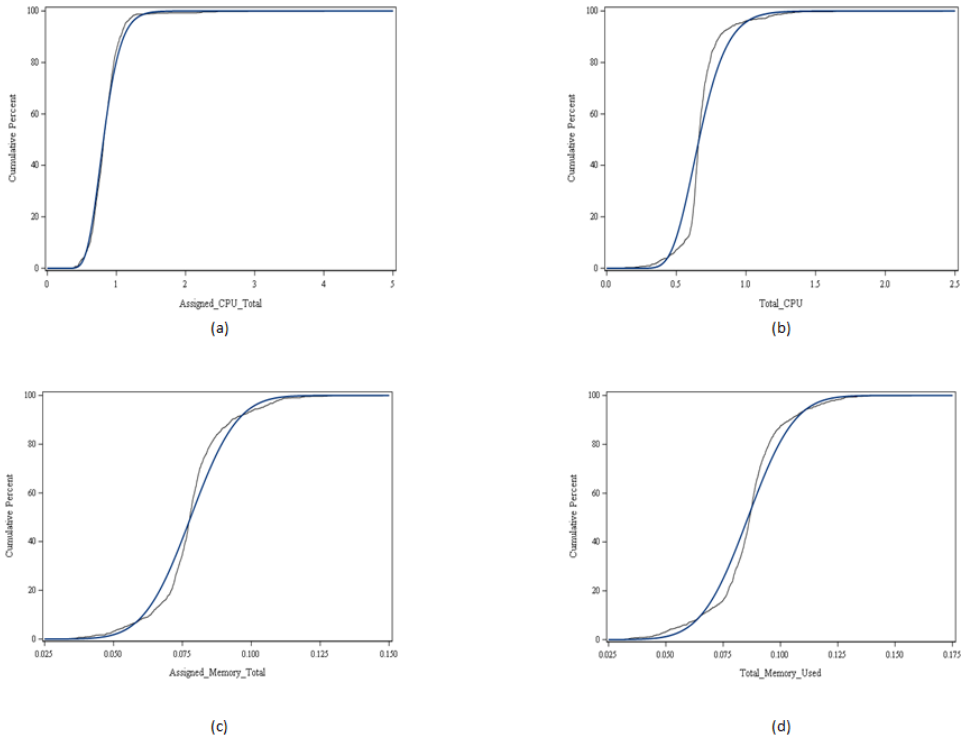


Figure 5.3. CDF of Job 5 (a) Assigned CPU (b) CPU Consumed (c) Assigned Memory (d) Memory Consumed

Table 5.3. Distribution Analysis for Job Resource Profile

Job Name	CPU					Memory				
	Assigned		Consumed		Proportion Wasted	Assigned		Consumed		Proportion Wasted
	Distribution	Parameters	Distribution	Parameters		Distribution	Parameters	Distribution	Parameters	
Job 0	Normal	μ=0.16276 σ=0.04914	3P Lognormal	Θ= 0.0025 ζ=-4.788 σ= 0.8408	90.5%	3P Weibull	Θ= 0.006 ζ=0.0166 c= 12.02	3P Lognormal	Θ= -0.003 ζ=-4.608 σ= 0.1204	23.5%
Job 1	Normal	μ =0.0822 σ = 0.0439	3P Lognormal	Θ= 0.0005 ζ=-5.56 σ=1.7625	87.8%	Gumbel	μ=0.004 σ=0.001	3P Lognormal	Θ=0.0002 ζ=-5.791 σ= 0.525	24.5%
Job 2	3P Lognormal	Θ=-0.033 ζ=-2.869 σ= 0.177	Normal	μ =0.0026 σ = 0.0015	88.8%	3P Lognormal	Θ= -0.018 ζ=-3.893 σ=0.0028	Lognormal	ζ=-5.791 σ= 0.525	42.5%
Job 3	Normal	μ = 0.0459 σ = 0.0266	3P Lognormal	Θ=376E-7 ζ=-6.471 σ=1.1861	91.6%	N/A	N/A	3P Weibull	Θ=-0.014 ζ=0.0618 c=31.124	31.2%
Job 4	Normal	μ = 0.0588 σ = 0.0261	Normal	μ = 0.0132 σ = 0.0102	80.8%	3P Lognormal	Θ= 0.0016 ζ=-8.162 σ=0.1728	Weibull	ζ=0.0016 c=5.3478	21.8%
Job 5	Lognormal	ζ=-0.208 σ=0.2412	Lognormal	ζ=-0.411 σ= 0.2398	16.15%	Normal	μ = 0.078 σ = 0.0134	Normal	μ = 0.0858 σ = 0.016	9.75%

Table 5.3 presents the distribution statistics for all the studied jobs along with the proportions of resources within every job execution. From Table 5.3, it can be observed that jobs are heterogeneous in effectively utilising the assigned resources without wastage of resources, predominantly following different distributions. Job 0 and Job 5 have been chosen to display their inner distributions of the analysed metrics, as they exhibit different extremism.

From Figure 5.2 and Table 5.3, the assigned CPU within Job 0 predominantly follows normal distribution and the consumed CPU predominantly follows 3P lognormal distribution and the curve is right skewed. Whilst the CPU cores has been provisioned and distributed equivalently across the encompassed tasks, a majority (around 90%) of the encompassed tasks consumed only a marginal proportion of the assigned resources and only a minority (around 10%) of tasks has consumed a reasonable margin of the provisioned resources. An immediate implication is that the former 90% of tasks are vulnerable to leave most of the provisioned resources utilised, causing 90.5% of CPU idleness since the provisioned and consumed curves are extremely heterogeneous. In addition, the memory assigned to tasks follows 3P Weibull and memory consumed follows 3P Lognormal distribution respectively, with both the curves are slightly left skewed. Idleness in the memory resources are witnessed at just around 23.5% since both the curves follow a similar distribution trend.

From Figure 5.3, both the CPU assigned and consumed curves predominantly follow Lognormal distributions and are left-skewed. Whilst more than 90% of the encompassed tasks within Job 5 are provisioned less than around 1.3 core counts, 90% of tasks has actually consumed less than around 1 core counts each. Thus the CPU assigned and consumed trend are nearly homogenous and the curves share a close enough distribution, whereby reducing the CPU idleness to 16.15%. Furthermore, a similar behavioural trend is evident in the trend of memory resources, since the memory assigned and consumed curves follow normal distribution. It can be arguably climbed that both the curves of memory trend is nearly identical, which has reflected in a significant reduction in the amounts of resources wasted accounting only for 9.75% of the provisioned resources. It can be postulated that the distribution trend of the assigned and utilised resources can directly reflect the proportional presence of idle resources. Thus jobs with heterogeneous distributions between the resources provisioned and resources consumed are vulnerable to leave most of the provisioned resources unutilised,

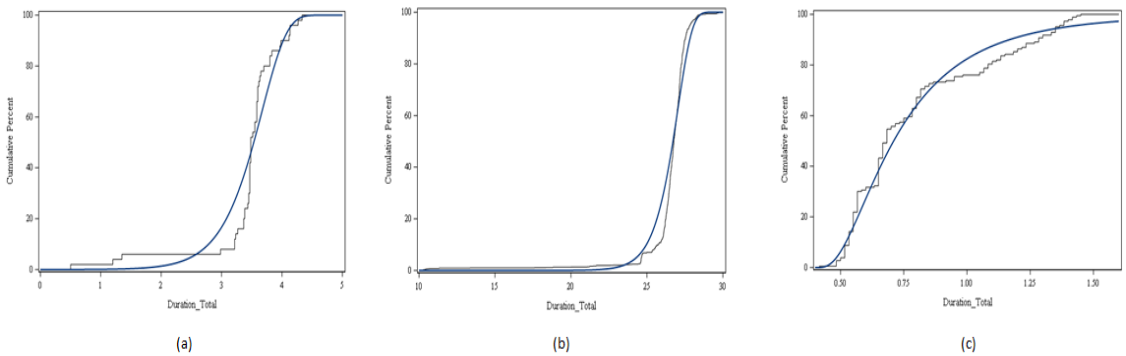


Figure 5.4. CDF of Task Duration (a) Job 0 (b) Job 5 (c) Job 3

Table 5.4. Distribution Statistics for Task Length within Jobs

Job Name	Duration	
	Distribution	**Parameters**
Job 0	3P Weibull	Θ=-21.25 ζ=24.92 c= 61.54
Job 1	Normal	μ = 2.5038 σ = 0.401
Job 2	3P Weibull	Θ= -0.594 ζ=1.644 c=8.4604
Job 3	3P Lognormal	Θ= 0.3881 ζ=-1.133 σ= 0.6901
Job 4	Weibull	ζ=3.0382 c= 2.2869
Job 5	3P Weibull	Θ=-57.39 ζ=84.448 c=89.825

causing a significant proportions of resource wastages. For achieving an energy efficient job execution, a close-enough distribution should be achieved between the trend of resources provisioned and resources consumed. Though, this is not always feasible in practice in an actual datacentre execution since the resource consumption trend is not known a priori before the actual execution.

Figure 5.4 displays the CDF with the best fit distributions and Table 5.4 presents the distribution statistics for the length of tasks encompassed within Job 0, Job 5 and Job 3. Both job 0 and Job 5 predominantly follows a 3P Weibull distribution and the cures are significantly left-skewed, insisting the fact that both jobs encompass tasks with shorter, medium and long running tasks causing an increased heterogeneity among task length within a single job. Within Job 0, 10% of tasks characterise a task length of around 3 minutes and another group of 10% of tasks characterise a task length of more than 4 minutes, and the remaining 80% of tasks runs between 3 and 4 minutes respectively. Within Job 5, 10% of tasks characterise a task length of around 26 minutes and another group of 10% of tasks characterise a task length of more than 28 minutes, and the remaining 80% of tasks runs between 26 and 28 minutes respectively. For both Job 0 and Job 5, task length is fairly homogeneous among 80% of tasks, the group of tasks with shorter length do not impact job completion time, but the group of tasks characterising longer duration than majority of tasks considerable affect job completion time since such long running jobs act as long tail stragglers within their respective jobs. Conversely, task length

distribution within Job 3 predominantly follows a 3P Lognormal distribution and the curve is right-skewed. Here task length is fairly heterogeneous across all the encompassed tasks within Job 3, with around 70% of tasks runs for less than a minute and the remaining 30% runs for more than a minute up to a maximum of 1.7 minutes. Here, isolating the group of long tails might not help early completion of job in spite of the heterogeneous distribution of task length within a single job.

From these observations, it is clear that jobs are increasingly heterogeneous further tasks encompassed within a single job exhibits an increased diversity in terms of their resource consumption pattern. Tasks may or may not exhibit homogeneity in terms of their running task length within a single job. Both jobs and every task within jobs should be uniquely treated whilst attempting to optimise their resource usage profiles for achieving energy efficiency. CPU resources provision are increasingly vulnerable to leave most of the provisioned resources unutilised and the provisioned memory resources are fairly utilised.

5.2.3 Task Termination Patterns

Though it is desirable to process job and tasks in one single execution instance, terminations are inevitable during the actual execution causing resubmissions. Whilst addressing over estimation of the resource levels causing idle resources, under-estimation of the resource levels leads to terminations whenever the execution exceeds the allowed or provisioned level of resources. The possible causes for tasks terminations have been discussed in Chapter 3, this section presents the trend of terminations of tasks within jobs with empirical analysis. Since resource level breaches are one of the important causes of task termination, the CPU usage rate in terms core-counts per second during any time of execution is crucial in determining task progress and terminations. Furthermore, given a task consuming a certain amount of resources during execution, its usage rate over the execution duration is important to characterise its resource intensiveness.

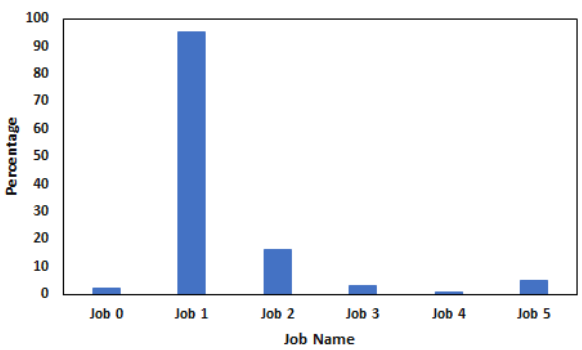


Figure 5.5. Task Termination Proportions

Figure 5.5 presents the proportions of task terminations within their respective jobs for the studied jobs. It can be observed that task termination proportions are totally unique for jobs and there is no linear dependency between the termination proportions and the total number of tasks encompassed within a job. For instance, Job 1 encompassing 100 tasks characterises an increased proportions of task terminations at 95% and Job 5 encompassing 1050 tasks only characterises 4.85% of task terminations. While the former is an entire job failure leading to the resubmission of the entire job, the latter is just a task terminations resubmitting only the terminated tasks.

Whilst addressing the resource related termination causes runtime execution factors such as CPU usage rate, task duration and the consumed level of memory resources can be postulated as potential causes exerting terminations. The termination pattern of all tasks encompassed within in randomly chosen Job 5 is displayed in Figure 5.6, where the duration is presented in minutes and the CPU usage rate is presented in core counts per second and the memory usage is presented in bytes. Job 5 encompasses a total of 1050 tasks, where 49 tasks have faced terminations. From Figure 5.6, it can be postulated that at least of one the aforementioned three factors can trigger a task termination event. In other words, a task is terminated when either a given task runs longer than the mean duration of job, or the assigned level of memory resources are breached or when the CPU usage rate of a given task exceed the mean CPU usage rate of job. However, a task termination without the involvement of these three factors is still possible as a rare phenomenon due to other run time factors such as LXC/VM crashes, hardware failures, hot spots, co-located VM or LXC influences etc.

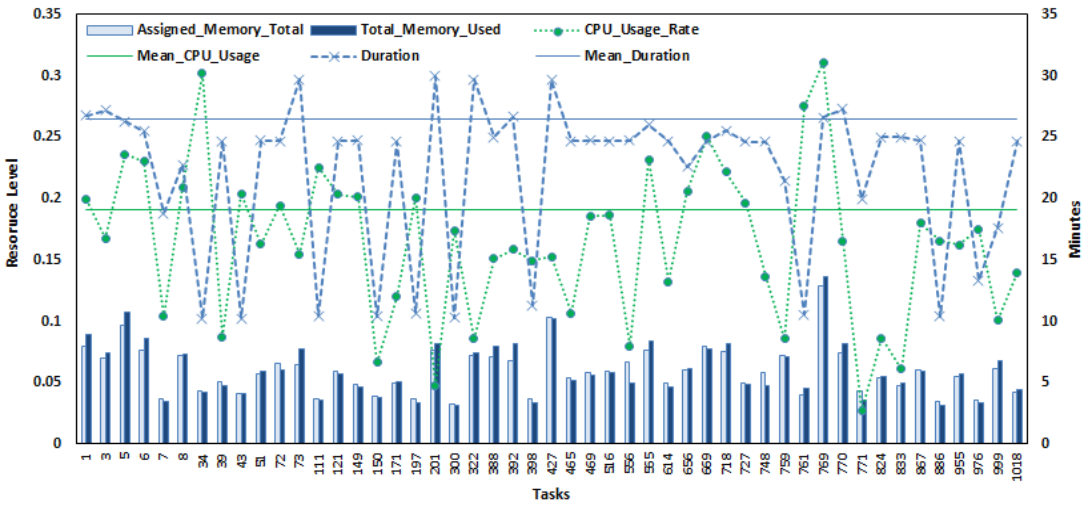


Figure 5.6. Task Termination Pattern

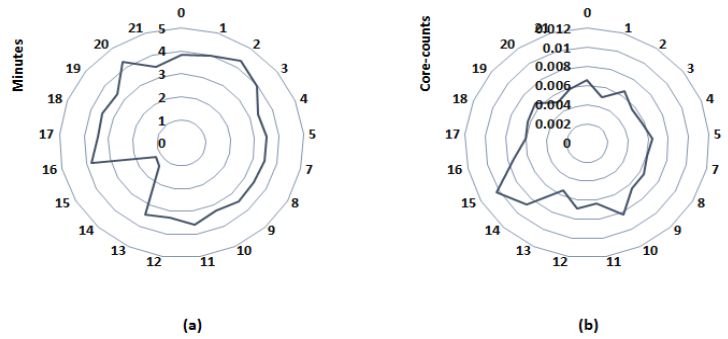


Figure 5.7. Task Execution Pattern within Job 0 Satisfying Usage Rate Duration Trade-off (a) Task Duration (b) CPU usage rate

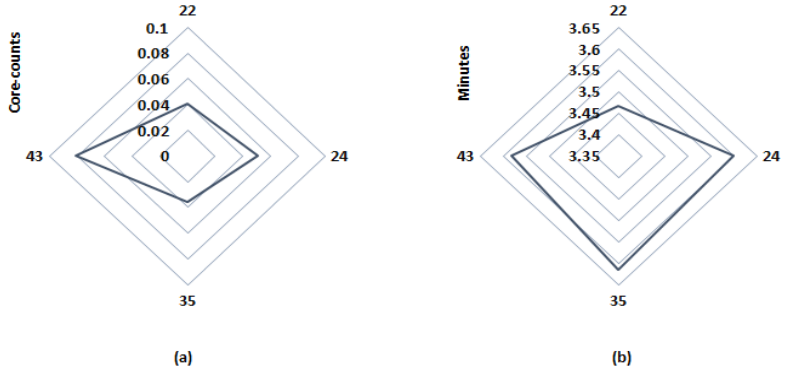


Figure 5.8 Task Execution Pattern within Job 0 not Satisfying Usage Rate Duration Trade-off (a) Task Duration (b) CPU usage rate

5.2.4 Task Execution Trend

It is quite an obvious fact that the process capacity of the resources should be increased for a task execution for the purpose of shortening task execution time. In other words, tasks can be executed either with a lower CPU usage rate and longer duration or with a higher CPU usage rate and shorter duration. With the CPU usage rate exhibiting increased fluctuations throughout task execution, memory usage is fairly remain stable. The existence of the relationship [130] between the server node statistics such as CPU/memory resources and task duration has been revealed to be non-trivial. The trade-off between task duration and CPU usage rate is crucial in determining task execution efficiency. For instance, shortening task duration may demand higher CPU usage rate and vice versa, but tasks with extravagant CPU usage rate are witnessed to be resource hungry. Though running tasks longer might characterise an optimum CPU usage rate, long running tasks might potentially act as long tails. Thus optimising this duration-usage rate trade-off should be considered as an essential criterion whilst achieving termination less energy efficient task execution. Figure 5.7 illustrates the trade-off between the CPU usage rate and task duration for selected non-terminated tasks within Job 0. It can be observed that the

trade-off between the CPU usage rate and duration are satisfied by most of the non-terminated tasks, such that tasks characterising higher CPU usage rate is exhibiting shorter duration and vice versa. For instance, Task 1 runs almost for 4 minutes and its corresponding mean CPU usage rate is quite low at 0.005 core counts per second. On the other hand, Task 15 is an example of shorter duration with higher usage rate characterising a duration of 1.2 minutes with a mean usage rate of 0.01 core counts per second. In spite of their termination probabilities, such a trend of task execution satisfying the usage rate and duration trade-off can be regarded as a healthy execution. However, there could be a few exceptions where tasks are still completed without satisfying this trade-off between usage rate and duration. Figure 5.8 illustrates a few examples of tasks within Job 0 where the usage-rate and duration trade-off is not satisfied, however such tasks are terminated during execution. Here all tasks have consumed at least 0.03 core counts but still exhibiting a duration longer than majority of other co-located tasks. Task execution can breach the usage rate and duration trade-off by characterising either shorter duration with lower usage rate or longer duration with higher usage rate. Whilst the former can be regarded as less resource intensive tasks and can be ignored as they do not impact the excess energy expenditures to any notable level, the latter are resource hungers and they naturally demand more resources at an alarming level than the remaining co-located tasks within the same job. However, this trade-off combination of tasks within a single job is not universal for all job types. A job is said to be exhibiting a homogeneous execution trend if all the encompassed tasks either satisfy or dissatisfy the usage rate and duration trade-off. Jobs encompassing combinations of tasks satisfying and dissatisfying the usage rate and duration trade-off is regarded as exhibiting heterogeneity in terms of their execution trend. Heterogeneous jobs usually pose increased complexities in achieving an optimum energy conserving resource provision across the encompassing tasks since the execution trend of the individual tasks are not known a priori before the actual execution.

5.2.5 Energy-aware Stragglers

Whilst the phenomenon of a few proportions of tasks running significantly longer than majority of tasks within a single job is defined as long tails, this chapter postulates a few proportion of tasks consuming significantly higher amounts of resources than majority of the remaining tasks as energy-aware stragglers. Long tails usually exhibit a duration as an increasing multiple of task length of the majority of the remaining tasks within the same job. Tasks not satisfying the usage rate and duration trade-off by exhibiting higher usage rate and longer task length are anticipated to be resource hungers than the other co-located tasks within a single job. To this

end, tasks characterising a combination of higher usage rate and longer duration than those of the average task CPU usage rate and duration are classified as energy aware stragglers, as shown in equation 5.6.

$$S_t[i] = (U_C[i] > \mu) \cap (L_T[i] > \omega) \quad (5.6)$$

where, S_t denotes tasks labelled as energy-aware stragglers, U_C is the CPU usage rate of task i, L_T is the length of the i^{th} task, and μ is the mean CPU usage rate of the entire job computed as the average of the mean CPU usage rate of all the encompassed tasks within the respective job, and ω is the average duration of the entire job computed as the average of the duration of all the encompassed tasks within the respective job accordingly. S_t is expected to deliver n number of stragglers within a given job.

Definition: *Energy aware stragglers*. For a job set $J = \{t_1, t_2, t_3, \ldots\ldots t_n\}$, where n is the total number of tasks within job J, with an average job CPU rate of μ and an average task length of ω, than tasks characterising both a CPU usage rate higher than μ and running longer than ω are termed as energy-aware stragglers within job J.

The presence and the proportions of stragglers vary from job to jobs. Though the proportions of the stragglers are expected to increase with increasing number of tasks within a given job, this proportion is not always linear. Figure 5.9 illustrates the proportions of energy-aware stragglers within the studied jobs. The empirical analysis conducted in this research demonstrates the presence of 8% energy-aware stragglers within Job 0 containing 50 tasks which is observed just to be around 2% within Job 5 containing 1050 tasks. Furthermore, Job 4 comprises more than 40% of tasks exhibiting the characteristics of energy-aware stragglers, this job can be classified as naturally resource intensive with all tasks are naturally expected to be resource hungry. The existence of the correlation between the process capacity of the nodes and task progress are beneficial in identifying the node level stragglers, where task progress is

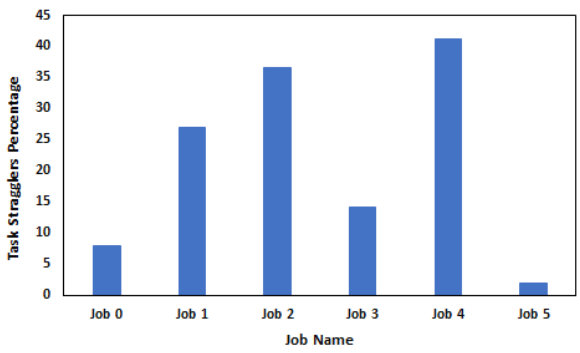


Figure 5.9. Energy-aware Straggler Proportions within Jobs

determined by node efficiency. Node-level straggles are caused by various explicit runtime events and process environments. Co-located tasks competing for similar resources on a node processing various task execution usually impact the node performance and increase the probability of the corresponding node becoming a straggler, potentially delaying all tasks being executed in that particular node.

Task-level stragglers pose an increased complexities in their identification and mitigation as their actual cause is often implicit. For instance, data-intensive tasks may characterise a slow progress rate and naturally run longer than less intensive tasks despite the node capacity. With the Cloud jobs being dynamic in terms of their resource requirements, tasks characterising higher level of CPU/memory requirements than the remaining tasks within the same jobs might turn out to be potential stragglers during the runtime. Furthermore, the inherent dynamicity of resource requirements among tasks within a single job naturally adds to the variations in the process behaviours of tasks belonging to a single job. It is worthy of note that in some cases task-level straggler are unavoidable driven by the actual task requirements.

The impacts of the energy-aware stragglers can be well observed from Figure 5.1(a), where Job 0 comprises a total of 4 energy aware stragglers (task index: 22, 24, 35 and 43) and are clearly exhibiting a higher CPU usage than the remaining tasks. In an attempt to satisfy the resource requirements of these smaller proportions of energy-aware stragglers providers overcommit the resource levels for all tasks within the corresponding job. This actually results in majority of the non-stragglers leaving provisioned resources unutilised, causing 90.5% of CPU idleness in Job 0. Thus, this chapter addresses task-level stragglers as one of the major causes for extravagant level of resource provision, and classifying energy-aware stragglers before the actual task execution might help to avoid over commitment of resource levels for non-stragglers in order to avoid resource wastages, and early and accurate identification of energy-aware stragglers during task execution benefits effective mitigation of the same.

5.3 Analytics Architecture

The proposed analytics architecture is aimed at estimating the resource consumption levels of the arrived tasks within jobs and further includes a classification framework to identify the energy-aware stragglers both before and during task execution and a resource estimation module to avail the most appropriate level of resource provision for tasks execution, with the motivation of reducing excess level of resource provisioning.

The proposed analytics architecture is illustrated in Figure 5.10. The proposed analytics architecture encompasses several components for functionalities such as sample selection, imputation, execution trend analysis, straggler classification, resource estimation and resource level optimisation. The primary purposes of these encompassed functionalities are detailed as follows.

Sample selection: The primary functionalities of the sample selection module is to choose the most appropriate samples from the historical traces for descriptive analytics and further to drive the predictive analytics. Most suitable historical samples are chosen and validated based on a statistical similarity measure for further analysis.

Imputation: It is common that the extracted historical samples might be incomplete in such a way that the execution profile for tasks within a given job might not be available. The imputation module is responsible to obtain a complete execution profile for jobs by inferring and imputing the missing values.

Execution trend analysis: An analysis in now conducted on the complete execution profile of the validated historical samples to observe the actual execution profiles of jobs. This analysis in intended to observe the usage rate duration trade-offs, job and task termination patterns, and resource consumption trend during historical execution instances.

Resource estimation: Driven by the descriptive analytics of the historical execution instance, the anticipated resource consumption levels of every individual task within a given job are estimated for resource provision.

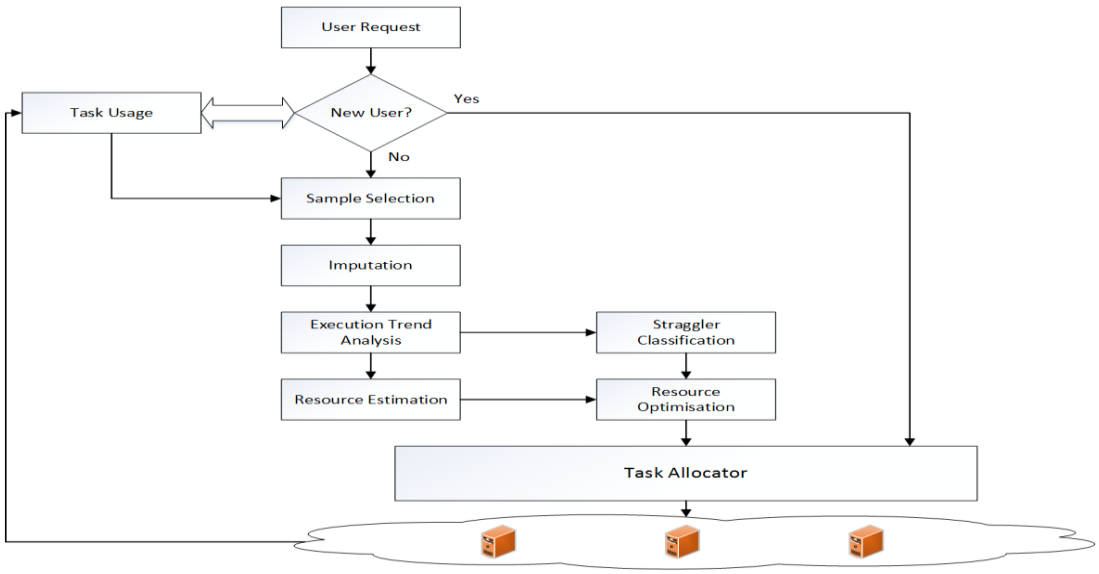


Figure 5.10. Analytics Architecture

Straggler classification: Based on their resource intensiveness, every tasks within a given job are subjected to a classification framework to forecast the anticipated execution behaviour for tasks within jobs. This classification is intended to isolate energy-aware stragglers within jobs from non-stragglers for further optimising their level of resource provisioning. Furthermore, appropriate thresholds for CPU usage rate and duration are determined to identify energy-aware stragglers during runtime.

Resource level optimisation: Based on the estimated resource levels and straggler classification, optimum level of resource provision for every individual tasks within a given job are determined considering various runtime and execution factors.

5.4 Resource Estimation Analytics

This sub section details the working mechanism behind the encompassing components of the resource estimation module in the proposed analytics architecture.

5.4.1 Sample Selection

Firstly, based on the currently arrived job profile similar execution profiles from historical instances are chosen for descriptive analytics by exploiting the inherent periodicity such as the time-of-the-day and day-of-the-week effects among user behaviours similar to the methodology discussed in section 4.3.1. It is always recommended to choose the complete execution profile which can only be obtained from finished jobs, however it is possible that jobs facing terminations may or may not be resubmitted again. In this event, the historical sample selection might return more than one execution instance. Choosing the execution profiles of the terminated jobs might not provide sufficient inferences for accurate analytics. Thus it is essential to validate the most suitable historical sample for a given job profile. A similarity weight is computed for the chosen historical samples by measuring the quantitative association of the statistical properties between the current and historical samples in terms of the number of tasks encompassed within jobs, resource requests, job scheduling priorities and task priority levels and termination pattern. A Profile Information (PI) table for similarity measure is constructed based on the values presented in equation 5.1 and 5.2. The execution instances exhibiting a close quantitative association with the currently arrived job profile are validated by the PI table and are then utilised for further descriptive and predictive analytics for respective jobs.

The construction of the PI table for Job 0 arrived during 12 - 1 am on Day 10 Wednesday of Week 2, is illustrated in Table 5.5. The statistical composite for the currently arrived job is extracted as $J0 = \{12.15\ am,\ 6323881198, 50,\ 0\}$, implying that Job 0 has been submitted at 12.15 am and has been assigned with a job index of 6323881198, encompasses a total of 50 tasks and has a scheduling priority of 0 (low latency sensitivity). Task profile, named *T0*, for job *J0* has been extracted as a statistical composite $T_0 = \{t_{st,}\ 6323881198,\ (0.03125,\ 0.007767),\ 4\}$ implying that task T_0 has arrived at a time t_{st} belongs to job 6323881198 characterise a CPU request of 0.03125 cores and a memory request of 0.007767 bytes, and includes a task priority level of 4. Task profile is constructed for all the encompassed tasks within a given job.

Now the sample selection module looks for similar job profiles from the historical samples. Three such similar job profiles have been selected from the historical traces for both the time-of-the-day and day-of-the-week samples. The PI table assigns a similarity weight for profile composite, identically associated metrics are assigned a weight of 1 and the metrics deviating from the current job profile are assigned with corresponding deviation for similarity measure.

Table 5.5. Profile Information Table for Job 0

Profile Composite / Similarity	**Day-of-the-Week Sample**			**Time-of-the-Day Sample**		
Sample Information	**Day 3 Week 1 Wednesday 12-1 am**			**Day 9 Week 2 Tuesday 12-1 am**		
Execution Instance	1	2	3	1	2	3
Job ID n_j	6275968532	6275636968	6275804419	6314856996	6314956139	6315082527
Total number of tasks n_t	50 / 0	49 / -1	48 / -2	48 / -2	48 / -2	48 / -2
Job Scheduling p_j	0 / 1	0 / 1	0 / 1	0 / 1	0 / 1	0 / 1
Task Priority p_t	4 / 1	4 / 1	4 / 1	4 / 1	4 / 1	4 / 1
CPU Requirements r_c	0.03125 / 1	0.03125 / 1	0.03125 / 1	0.03125 / 1	0.03125 / 1	0.03125 / 1
Memory Requirements r_m	0.007767 / 1	0.007767 / 1	0.007767 / 1	0.007767 / 1	0.007767 / 1	0.007767 / 1
Task Termination Profile	Evict / -1	Kill / -2	Kill / -2	Kill / -2	Evict / -1	Kill / -2
Similarity Score	3	1	0	0	1	0

This computation also includes the measure of task termination proportions in the historical execution instances and assigns a weight of 1 for instances without any task terminations and -1 for evicts, -2 for kill and fail events accordingly in the historical execution traces. A similarity score is generated as the summation of all the individual parametric score for all the identified execution instances. Based on the association measure, all the six execution instances can be validated as different execution instances of the same job profile, but the similarity score helps to choose the execution instance with minimal measurable deviation from the currently arrived job profile. Based on the similarity score, instance 1 of the day-of-the-week sample and instance 2 from the time-of-the-day sample are validated as the representatives of the two periodical effects respectively. In addition the day-of-the-week sample is exhibiting close association than the time-of-the-week sample, which will be later considered in the predictive analytics. Now the execution profiles of the validated execution instances are subjected to descriptive analytics and further utilised in the predictive analytics for estimating the resource consumption level of individual tasks within jobs.

5.4.2 Imputation

Imputation is a process of replacing or substituting missing data in statistical analysis, which is required for the execution samples due to the higher probability of the execution profiles including data ambiguity, missing data and possible anomalies. Profile incompleteness can be described from two different perspectives: firstly where the execution profile consisting of missing values leading to ambiguous profile, secondly where the execution sample containing task profile usages less than those of the currently arrived job resulting in incomplete profile of jobs. The missing values in the both ambiguous profile and the incomplete profile are estimated using Maximum Likelihood Estimation (MLE), as shown in equation 5.7. Tasks within a single job usually behaves distinctly and characterise a heterogeneous execution trend, MLE assumes X_i to be normally distributed around a constant mean and variance for a random sample $X_1, X_2, \ldots, X_n$. for estimating the value of missing X_i. MLE substitutes or estimates the missing value with the predicted value that maximises the probability (likelihood) and minimizes the imputation error.

$$L(\theta) = P(X1 = x1, X2 = x2, \ldots, Xn = xn) = f(x1;\theta) \cdot f(x2;\theta) \cdots f(xn;\theta) = \prod i = 1nf(xi;\theta) \quad (5.7)$$

The first equality is the definition of the joint probability mass function and the second equality comes with the consideration that the sample is a random function, implying Xi is independent. The last equality uses the shorthand mathematical notation of the indexed sample values.

Expectation-Maximization (EM) algorithm is used to estimate the maximum likelihood. EM converges in *n* number of iteration depending on the data sample, and estimates the unknown value through *n* number of equalities. Samples suffering incompleteness are subjected to the process of imputation. After this phase, complete execution profiles for the historical samples can be obtained.

5.4.3 Resource Estimation

The resource estimation module is primarily responsible for estimating the anticipated level of resource consumption for every individual task within a given job. The estimation module exploits the validated historical samples as the baseline for usage estimation in terms of the total amounts of CPU consumption and anticipated duration for every individual tasks. The CPU usage rate of tasks and their execution duration of the historical samples are used as input training sets for estimating the anticipated equivalents during the current execution. A dynamic weighing causal moving average (DWCMA) filter is adopted to estimate the CPU usage trend and duration of tasks. Traditional causal moving average filter assumes a given output sample depends only on the corresponding inputs occurred earlier and usually assigns more weights to the most recent samples, as shown in equation 5.8.

$$y(n) = b(1) * x(n) + b(2) * x(n-1) + \ldots + b(Nb+1) * x(n-Nb) \quad (5.8)$$

where *y(n)* is the output response depending on the previous occurrences based on *x(n), x(n-1)* etc., and *b(1), b(2)…b(n)* are the exponentially assigned weight functions assigned to the past occurrences. It is obvious that such a filter is linear and shift-invariant meaning that $y(n)$ is the output response to $x(n)$, then $y(n–k)$ is the response of the system to $x(n–k)$.

Definition. Linear shift-invariant. For input sets of variables $X = \{x_1, x_2, x_3, \ldots x_n\}$ and $Y = \{y_1, y_3, y_3, \ldots y_n\}$, the response variables within the output $Z = \{z_1, z_2, z_3, \ldots z_n\}$ are internally independent such that the response z_i depends only on its corresponding past instances x_i and y_i.

Hypothesis: Though task execution behaviours are dynamic with a given job in such a way that a given job might include energy-aware stragglers and tasks may not satisfy the usage rate-duration trade-off, it is initially assumed that tasks within a given job will behave normally as non-stragglers and will satisfy the usage-rate duration trade-off.

The proposed resource estimation module adopts the above hypothesis for initially estimating the resource consumption levels of tasks within jobs, however this hypothesis may not

necessarily be always true for job executions. Tasks failing to meet the hypothesis are moderated accordingly, discussed later in this chapter. Now, from the two sets of historical inputs the sample set exhibiting better association with the currently arrived job profile is naturally assigned more weights by the DWCMA filter. But this initially assigned weight is dynamically swapped for every individual task execution profile depending on several runtime factors. Unlike the traditional casual moving average filter, the proposed DWCMA filter assigns more weights to the most appropriate or (in-trend) sample for a given task execution from the two sets of historical input usage profiles. The term in-trend here refers to the satisfactory level of a task execution profile being a non-straggler and satisfying the usage rate-duration trade-off. As discussed earlier, usage rate and duration are inversely proportional to each other for a healthily executed task. Based on the actual usage behaviours of tasks within a given sample, an optimal value for usage rate-duration trade-off $opt(u, d)$ for all tasks within a job J is defined as in equation 5.9.

$$opt(u, d) = \underbrace{mean}_{\forall t_i \in J} \{u_i, d_i\} \tag{5.9}$$

It is not a practical reality that every individual task execution within a given job to exhibit the expected level of healthily execution trend. Hence measureable deviations are always evident among the individual task execution within a job. From task execution trend observations presented earlier in section 5.2.4, for a decreasing CPU usage rate the duration of task usually increases and vice versa, however the proportional relationship in this trade-off is not obvious.

In addition to the execution trend, task profiles extracted from the actual execution are treated with more weights than those imputed. The protocol for assigning weights to the two samples by DWCMA is presented in Figure 5.11. The weight assignment is executed in three sequential phases, Phase I verifies the Similarity Scores assigned by the PI table, Phase II verifies the correctness of the samples depending on the availability of the execution profile from actual execution and Phase III verifies the usage rate-duration trade-off for every task execution. The two samples sets are assigned with weights depending on the execution profile satisfying the three phases, task profiles within the respective two samples satisfying Phase I and Phase III are assigned with increasing weights and the weights are decreased with a degrading function when Phase II is violated. This is because the EM algorithm imputing the missing values with an overestimated value. The weights are assigned to the sample sets based on equation 5.10, where n is the total number of sample sets.

$$\omega = \begin{cases} \frac{1}{n}, & phase\ II\ violated \\ \frac{2}{n+1}, & otherwise \end{cases} \quad (5.10)$$

The CPU usage rate and duration and the total CPU consumption for every individual tasks within a given job are estimated by the dynamic weighing CMA filter as a tuple shown in 5.11. Whilst the total CPU consumption provides inferences for optimum level of resource provision, the estimated usage rate and duration are expected to provide inferences for straggler classification, dealt in section 5.5.

$$P_{out}[i] = \{u_i, d_i, c_i\} \quad (5.11)$$

where $P_{out}[i]$ is the estimated tuple for task i, encompassing its corresponding CPU usage rate u_i, duration d_i and total core consumption c_i.

5.4.4 Performance Evaluation

5.4.4.1 Compared Techniques

The efficiency of the proposed dynamic weighing CMA filter algorithm for estimating the resource requirements of jobs arriving at the datacentre has been evaluated against the existing state-of-the-art techniques including Simple Moving Average filter, Exponential Moving

Inputs: $J_1 = \{t_1, t_2, t_3, \ldots, t_n\}$ and $J_2 = \{t_1, t_2, t_3, \ldots, t_n\}$ → Historical input samples
S_s = Similarity scores of J_1 and J_2 respectively
Parameters: *opt(u, d)* → optimal values of usage rate-duration trade-off within J_1 and J_2
Output: $\omega(J_1, J_2)$ → Weight of J_1 and J_2 in DWCMA filter
For task t_i:

if $S_s[J_1] > S_s[J_2]$ *(phase I)*
  then $\omega \rightarrow J_1[t_i]$ and $(1-\omega) \rightarrow J_2[t_i]$
  if $J_1[t_i]$ imputed is true *(phase II)*
    then $\omega \rightarrow J_2[t_i]$ and $(1-\omega) \rightarrow J_1[t_i]$
    break
  else no change
  if $opt_1(u, d)$ is not true *(phase III)*
    then $\omega \rightarrow J_2[t_i]$ and $(1-\omega) \rightarrow J_1[t_i]$
  else no change
else $\omega \rightarrow J_2[t_i]$ and $(1-\omega) \rightarrow J_1[t_i]$
  (same steps in the previous loops are repeated)

Figure 5.11. Dynamic Weight Assignment Protocol

Average, Low Pass filter, Auto Regressing Moving Average (ARMA) and Linear Regression. The formulation of the compared techniques are briefed as follows.

$$Simple\ Moving\ Average = \frac{1}{n}\sum_{i=0}^{n} u_i \tag{5.12}$$

$$Exponential\ Moving\ Average = \alpha\, u_i + (1-\alpha)u_{i-1} \tag{5.13}$$

$$Low\ Pass\ Filter = \omega\, P_{n-1} + ((1-\omega)V) \tag{5.14}$$

$$(ARMA)\ \lambda(t+1) = \beta\, \lambda(t) + \gamma\, \lambda(t-1) + (1-(\beta+\gamma))(\lambda(t-2)) \tag{5.15}$$

$$(LR)\ Y_i = \beta_1 + \beta_2 X_i \tag{5.16}$$

where,

n - total number of historical usage profiles

u_i - usage profile in terms of resource consumption of the i^{th} sample

α - exponential weight assigned to the historical samples computed as $\frac{2}{n+1}$

ω - degradation constant

P_{n-1} - prediction of the previous run, and V - average observed value of the previous run

5.4.4.2 Evaluation metrics

The estimation efficiency of the proposed Dynamic Weighing Casual Moving Average Filer has been evaluated by the following evaluation metrics.

Coarse Grain Analysis presents an initial evaluation of the prediction techniques by plotting the predicted trend of the proposed and the aforementioned benchmark techniques against the actual trend of resource consumption of tasks encompassed within the studied jobs.

Over Prediction Ratio is a quantitative measure, computed as the ratio of the total number of over predicted tasks within a given job to the total number of tasks encompassed within the corresponding job. Similarly, *Under Prediction Ratio* is a quantitative measure, computed as the ratio of the total number of under-predicted tasks within a given job to the total number of tasks encompassed within the corresponding job. With both having respective consequences, over prediction is usually recommended for Cloud datacentres to avail suffice resource levels for tasks to achieve termination less execution. But the margin of over estimation is crucial in lowering the amounts of excessive resource provision and incurring resource wastages.

$$Over\ Prediction\ Ratio = \frac{Number\ of\ Over\ estimated\ Tasks}{Total\ Number\ of\ Tasks}$$

$$Under\ Prediction\ Ratio = \frac{Number\ of\ Under\ estimated\ Tasks}{Total\ Number\ of\ Tasks}$$

Average Accumulative Error (AAE) is a quantitative measure depicting the prediction efficiency of the aforementioned methodologies in terms of the degree of deviation of errors as shown in equation 5.17, where N is the total number of tasks including both the over and under-estimated errors, X_i and $\hat{X}_i$ are the actual and predicted amounts of resource consumptions respectively for task i. Given a job with n number of tasks, average accumulative error is computed as the average of the difference between the estimated and the actual value. Lesser the value of the average accumulative error, better is the prediction efficiency.

$$AAE = \frac{1}{N}\sum_{i=1}^{N}\left|X_i - \hat{X}_i\right| \tag{5.17}$$

5.4.4.3 Resource Prediction Analysis

This section presents the analysis of resource estimation for all tasks encompassed within a given job, this includes non-stragglers, energy-aware stragglers and long tails. The estimation efficiency of the proposed DWCMA protocol is presented alongside the aforementioned benchmark techniques against the actual trend of resource consumption for the studied jobs.

Figure 5.12 presents the resource estimation in terms of the estimated core counts against the actually consumed core counts for Job 0, Job 1, Job 2, Job 4 and Job 5 respectively. Since Job 3 does not satisfy the historical window requirements of the proposed methodology, evaluation

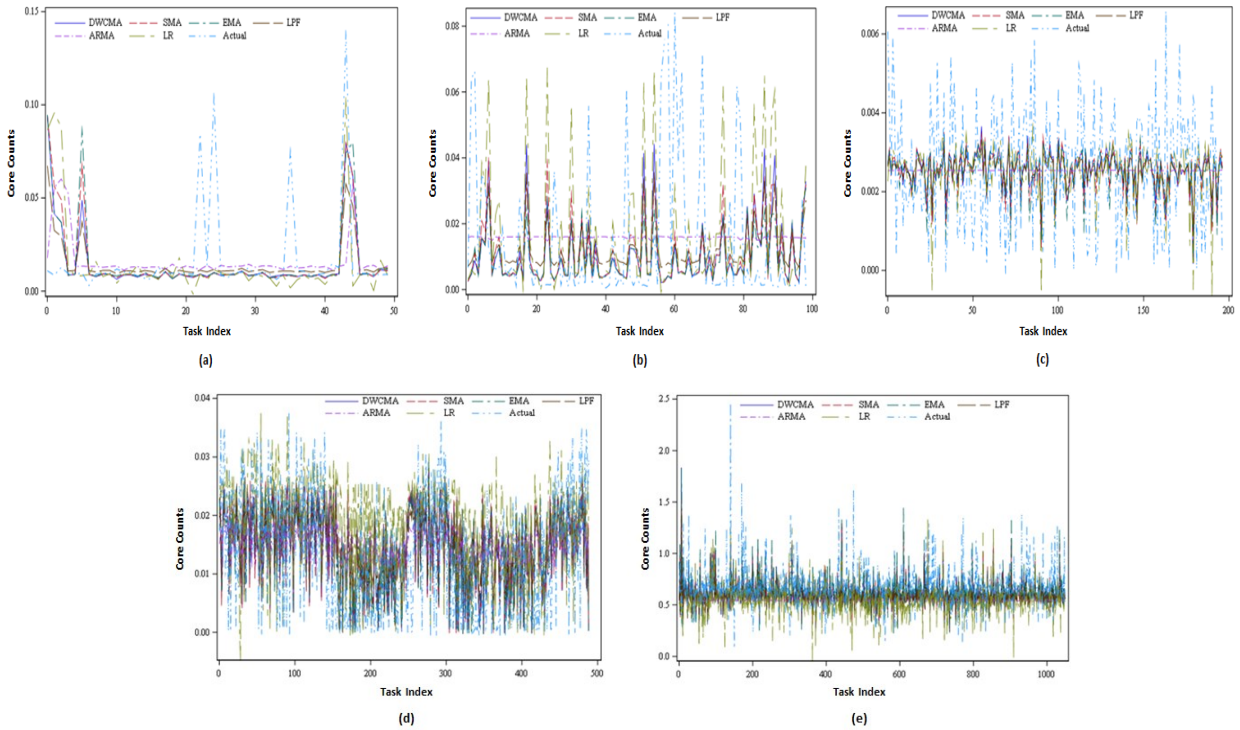


Figure 5.12. Resource Estimation Observation (a) Job 0 (b) Job 1 (c) Job 2 (d) Job 4 (e) Job 5

of Job 3 is not included. The unusual spikes in the actual trend illustrates the increased core consumption of the energy-aware stragglers. On a coarse grain, it can be observed that the resource estimation for non-stragglers by the evaluated techniques are in close correlation with the actual trend, but 3 out of 4 energy-aware stragglers are not captured by any of the prediction techniques. This exhibits the increased analytics complexity in capturing and predicting the resource requirements of energy-aware stragglers. Though all the techniques are observed to be closely predicting the resource requirements of the non-stragglers, the efficiency of such techniques are precisely evaluated later in this chapter for all the studied jobs. In the case of Job 1, ARMA presents a linear prediction for all the encompassed tasks, and it is over-predicting the resource requirements of non-stragglers by a considerable margin. This is due to the inefficiency of ARMA model to capture minute deviations among the sample observation, and ARMA model is vulnerable to cause increased energy wastages during task execution. In the case of Job 2, all the techniques are vulnerable to under-estimate the resource requirements of energy-aware stragglers and ARMA model delivers a flat prediction. Similar behaviors of energy-aware straggler prediction can be observed for Job 4, where a few of the non-stragglers are over estimated by a significant margin by all the prediction techniques. Interestingly, all the prediction techniques are closely estimating the resource requirements of tasks encompassed within Job 5. This is because the proportional presence of energy-aware stragglers are insignificant in Job 5, accounting only around 2%. Though marginal, the resource estimation of energy-aware stragglers needs more preciseness to avoid the probability of resource related terminations.

5.4.4.4 Prediction Ratio

This section evaluates the over and under estimation efficiencies of all the evaluated techniques while predicting the resource requirements in terms of the core counts for tasks encompassed within the studied jobs.

Figure 5.13 presents the over and under-prediction ratio of all the evaluated techniques, presented as an average of all the studied jobs. Considering all the type of tasks, the proposed DWCMA over-estimates 44.65% of tasks and under-estimates 55.34% of tasks accordingly. The over estimation of the evaluated benchmark techniques are observed at an average of 45.34%, 46.16%, 54.04%, 54.14%, 47.32%, and the under estimation is observed at an average of 54.65%, 53.83%, 45.93%, 45.58% and 53.67% respectively for SMA, EMA, LPF, ARMA and LR. The estimation efficiency of all such techniques are further evaluated by isolating the

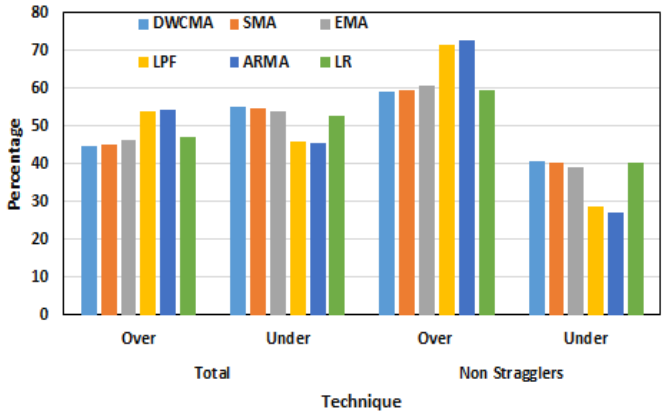


Figure 5.13. Over and Under Prediction Ratio

energy-aware stragglers and long tails, so as to estimate the resource requirements of the non-stragglers. Now, the proposed DWCMA over estimates 59.06% of tasks and under-estimates 40.93% of the non-straggler tasks respectively. Further, the over estimation of the benchmark techniques are observed at 59.51%, 60.75%, 71.40%, 72.72% and 59.71%, and the under estimation is observed at 40.48%, 39.24%, 28.59%, 27.27% and 40.28% respectively for SMA, EMA, LPF, ARMA and LR. It is clearly evident that the under prediction ratio for non-stragglers is much lesser than those of the total tasks (included energy-aware stragglers, non-stragglers and long tails), illustrating the impacts of energy-aware stragglers in prediction analytics.

The estimation ratio trade-off of the proposed DWCMA, SMA, EMA and LR fairly remains the same for both task classifications. But the over estimation ratio of LR and ARMA are higher than the remaining techniques by a considerable margin. Whilst is optimum for a prediction technique to over-estimate the resource requirements, this over estimation should always be marginally higher than the actual requirements. Over-estimating the resource level by a significant margin leads to resource wastages, which is actually the case of ARMA. In general, ARMA model fits well for time series trend prediction and usually presents the upper and lower confidence limits for prediction, but does not scale well for the context of resource requirement estimation. LPF presents a better over estimation ratio, however LPF depends on the average of prediction outcome of the previous iteration and adds a degradation function for the current sample. This increases the computational complexity by incurring multiple iterations of prediction analytics and the degradation function is vulnerable to over-commit the resource levels. Given such over and under prediction ratio, it is still unclear to conclude the optimum prediction technique. The error margin between the actual and predicted output of the resource

requirements is crucial in determining the estimation accuracy of the prediction methodology, which is dealt in the following section.

5.4.4.5 Average Accumulative Error

Energy-aware stragglers naturally characterise an increased resource consumption than those of the non-stragglers within a single job, thus pose an increased level of complexity in accurately predicting their resource requirements. Further tasks behaving as energy-aware stragglers are not known a priori before the actual execution. Thus the estimated level of resource consumption exhibit significant deviation from the actual consumption for estimated non-straggling tasks turning out to behave as energy-aware stragglers. For this reason, the energy-aware stragglers identified during the actual execution are eliminated for evaluating the prediction efficiencies of the aforementioned techniques. The resource estimation accuracy of the stated techniques are evaluated in terms of the average accumulative error for under-estimated and over-estimated tasks within a single job respectively. Figure 5.14 presents the

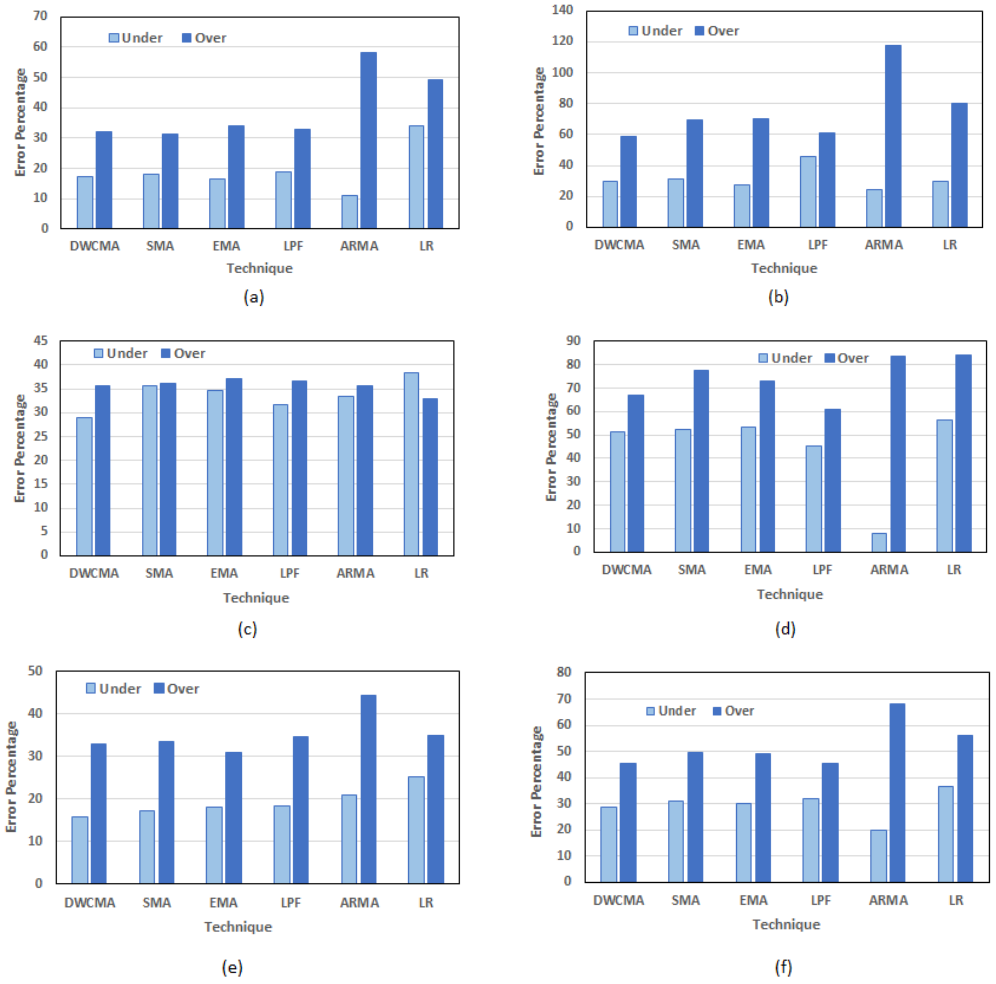


Figure 5.14. Average Accumulative Error for Resource Prediction (a) Job 0 (b) Job 1 (c) Job 2 (d) Job 4 (e) Job 5 (f) Average

accumulative error percentage for both the under and over-estimated resource levels for tasks within the studied job respectively.

It can commonly be observed that all the evaluated prediction techniques are exhibiting a better Average Accumulative Error Percentage for the under-estimated tasks than those of the over-estimated tasks, though the over-estimation ratio is better for non-stragglers. In other words, a majority of the non-straggling task proportions are over-estimated with high error percentage and a minority of the non-straggling task proportions are under-estimated with minimum error percentage. An increased diversity in the prediction accuracy is evident among the different types of jobs. For instance, Job 0 and Job 5 are exhibiting a better prediction accuracy than the remaining jobs, one common similarity between these two jobs is that the proportional presence of energy-aware stragglers are insignificant. Job 4 is exhibiting the worse prediction accuracy among the studied jobs, which encompasses more than 40% of energy-aware stragglers. ARMA model is observed to present a better under-estimation accuracy, but its over-estimation error margin is irresistible in the cases of Job 0, Job 1 and Job 4. This corresponds to the earlier observation of ARMA model exhibiting a better over-estimation ratio. Whilst over-estimating, ARMA model is vulnerable to excessively estimate the resource requirements to a level that can leave a majority of the estimated resources unutilised. The LR technique is exhibiting a poor estimation efficiency in the context of resource level estimation, with both the under- and over-estimation error percentage is irresistible. Though exhibiting a similar prediction ratio with SMA and EMA, the proposed DWCMA protocol exhibits a slightly better error percentage than the two techniques. LPF is exhibiting a better error percentage for different types of jobs, but outperformed by the proposed DWCMA on average. The Average Accumulative Error for under-estimated tasks are observed at 28.59%, 30.9%, 30.04%, 32.01%, 19.63% and 36.72% for DWCMA, SMA, EMA, LPF, ARMA and LR respectively across all the diverse group of jobs. Similarly, the Average Accumulative Error for the over-estimated tasks are observed at 45.24%, 49.69%, 49.14%, 45.19, 67.98% and 56.24% respectively for DWCMA, SMA, EMA, LPF, ARMA and LR across the diverse groups of jobs. Overall, it can be concluded that the proposed DWCMA protocol achieves a better prediction accuracy trade-off between the under and over-estimated non-straggling tasks within the given jobs than the compared benchmark techniques with better Average Accumulative Error. Though marginal, the under-estimated tasks are vulnerable for terminations due to under-commitment of resource levels, thus needs further optimisation for resource estimation. Though it is being argued that prediction based on repeated job submissions can only be

accurate with 25% of jobs, analytics at task level based on the proposed methodology can benefit estimating the resource requirements with much better prediction accuracy. The prediction and resource optimisation considerations for energy-aware stragglers and resource optimisation for under-estimated non-stragglers and the energy impacts of over-estimated tasks are discussed as follows.

5.5 Straggler Classification Framework

After estimating all the resource requirements of tasks encompassed within a given job, it is vital to classify tasks within a single job based on their resource intensiveness. From the earlier analysis, energy-aware stragglers tend to be dynamic in a way that straggling tasks during a given execution instance may not behave the same in another execution instance. Though, the proportions of task-level energy-aware stragglers remain nearly consistent among the different execution instances of the same job. The proposed straggler analytics framework is shown in Figure 5.15, which comprise a combination of offline and online analytics of straggler classification. Whilst the offline analytics is aimed at straggler classification before the actual execution, online analytics is aimed at identifying energy-aware stragglers during the runtime.

From the analysis of the selected historical job profiles, two initial lists of energy-aware stragglers are generated as S_{t1} and S_{t2}, respectively for the day-of-the-week and time-of-the-day samples. Tasks commonly witnessed as energy-aware stragglers in the two generated list can anticipated to be a definite straggler during the actual job execution, as shown in equation 5.18, S_{th} is the initial list of task-level energy-aware stragglers anticipated during the actual job execution based on the historical task behaviours. It is also a possibility that S_{th} can be an empty set at this point if none of tasks overlap in the generated lists of historical stragglers.

$$S_{th}[i] = S_{t1}[i] \cap S_{t2}[i] \tag{5.18}$$

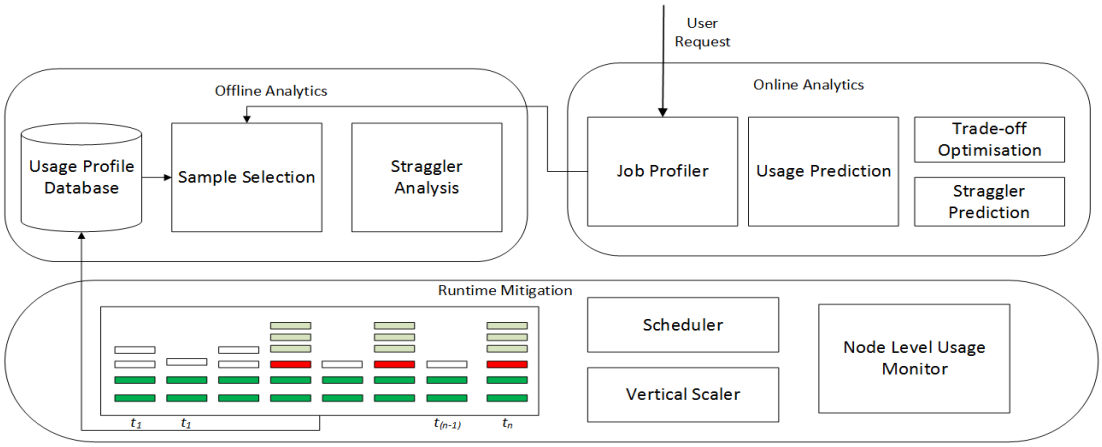


Figure 5.15. Straggler Classification Framework

5.5.1 Straggler Detection

This section presents the proposed analytics methodology for classifying the energy-aware stragglers before the initialisation of job execution. The output of the resource estimation module presents the anticipated usage profile for all tasks within the target job in terms of the CPU usage rate and the duration. Using a static mean value of the resource consumption of all tasks within a job may not benefit accurate estimation of a threshold for straggler identification. Based on the initial analysis of the straggler classification, an n^{th} percentile distribution of energy-aware straggles within the two historical samples is extracted. Now tasks not impacted by the abrupt behaviours of CPU usage rate and duration are isolated and categorised as non-stragglers based on the observations falling beyond the *(100-n)th* distribution, using equation 5.19.

$$N_{st} = W_{(1,2)}[i] \begin{cases} i_l < P_{(100-n)} \\ i_u < P_{(100-n)} \end{cases} \tag{5.19}$$

where, N_{st} is the sample containing non stragglers, $W[i]$ is the chosen two historical samples respectively, i_l is task duration, i_u is the mean CPU usage rate of the i^{th} task respectively, $P_{(100-n)}$ is the *(100-n)th* percentile value and *n* is the proportions of stragglers identified based on equation 5.18. After filtering out the energy-aware stragglers, threshold score for the CPU usage rate and duration for non-stragglers is obtained for the two samples using equation 5.20.

$$N_{s(\alpha_{(1,\ 2)},\ \beta_{(1,\ 2)})} = \left(\sum_{i=1}^{n} \frac{N_{st}[u_i]}{n} , \sum_{i=1}^{n} \frac{N_{st}[l_i]}{n} \right) \tag{5.20}$$

where, *n* is the total number of non-stragglers, α and β are the average values of the CPU usage rate and task duration of the non-stragglers with the two samples respectively. The non-straggler threshold values are computed separately for the two samples respectively as $N_{s1(\alpha,\ \beta)}$ and $N_{s2(\alpha,\ \beta)}$. These two values forms the upper and lower confidence limits for the average duration and CPU usage rate for the non-straggling tasks during the current execution. Now, this confidence limits are applied to the predicted output obtained in equation 5.11 to generate the list of non-stragglers bounded with a two-sided confidence limit for the target job respectively, as shown in equation 5.21 and 5.22 respectively.

$$Pu_{con}[i] = \alpha_1 < P_{out}[i] < \alpha_2, \qquad for\ CPU\ usage\ rate \tag{5.21}$$

$$Pl_{con}[i] = \beta_2 < P_{out}[i] < \beta_1, \qquad fortask\ duration \tag{5.22}$$

where, $Pu_{con}[i]$ and $P_{lcon}\,[i]$ are the list of tasks satisfying the confidence bounds of the average CPU usage rate and duration thresholds for the non-stragglers respectively for the predicted usage profile of the target job. Tasks anticipated to execute within the limits of P_{con} should satisfy the non-straggler criterion. However, the trade-off between task duration and the CPU usage rate is crucial in deciding the energy aware straggling behaviours of tasks. Furthermore, tasks anticipated to exhibit a duration less than the value projected by the confidence bounds are considered as not satisfying the non-straggler criterion, with the presumption that lower duration might characterise a higher usage rate and vice versa, thereby not satisfying the usage rate-duration trade-off given by equation 5.9. Anticipated task execution behaviours not falling within the healthily task execution criterion are vulnerable to become potential stragglers during the runtime. An optimised average CPU usage rate and duration for the non-straggling tasks during the actual execution is given by equation 5.24.

$$S^{th}{}_{(\alpha,\ \beta)} = \left(\frac{\sum_{i=1}^{n} P_{ucon}[i]}{n_u}, \frac{\sum_{i=1}^{n} P_{lcon}[i]}{n_l}\right) \tag{5.23}$$

where n_u and n_l are the total number of tasks in P_{ucon} and P_{lcon} respectively. The predicted output obtained from equation 5.11 is now subjected to this trade-off criterion and tasks not meeting $S^{th}{}_{(\alpha,\ \beta)}$ are further isolated and labelled as anticipated energy-aware stragglers during the actual execution. Now the final anticipated list of straggling tasks is given by equation 5.24. $S^{th}{}_{(\alpha,\ \beta)}$ is expected to present both the CPU usage rate and duration threshold values for both the usage and duration confidence limit samples. Ideally, α_1 and α_2 are the upper and lower thresholds for the CPU usage rate for non-stragglers for the predicted output, where α_1 is obtained from the P_{ucon} and α_2 is obtained from P_{lcon} respectively, and β_1 and β_2 are the duration counterparts, where β_1 is obtained from P_{lcon} and β_2 is obtained from P_{ucon} respectively, based on equation 5.23. The offline thresholds for non-stragglers anticipated in the predicted output is computed based on equation 5.24 and equation 5.25, which weighs α_1 and β_1 more than their counterparts α_2 and β_2, respectively.

$$S^{th}{}_{\alpha} = (\omega * \alpha_1) + (1 - \omega)\alpha_2 \tag{5.24}$$

$$S^{th}{}_{\beta} = (\omega * \beta_1) + (1 - \omega)\beta_2 \tag{5.25}$$

Now, an initial classification of the energy-aware stragglers anticipated in the current execution of the target job is achieved based on equation 5.26.

$$S_{t-off} = \{P_{out}[i]\{(i_u > \alpha)\ \cap\ (i_l > \beta)\} \cup (S_{tc}[i]) \tag{5.26}$$

where i_u is the CPU usage rate and i_l is task duration of i^{th} task contained in P_{out}, α is the optimised CPU usage rate and β is the optimised task duration respectively given by equation 5.24 and equation 5.25, and S_{th} is the initial list of stragglers given by equation 5.18.

S_{t-off} is the probability of stragglers identified from the offline descriptive analytics of the combination of historical events and predicted resource usage profiles of individual tasks within a given job. However, the actual task behaviour depends on several run-time factors such as the node efficiency, resource consumption fluctuation, running duration, task intensity etc., and are impacted by the sole effects of CPU usage rate and task runtime duration respectively. Thus, it is important to further optimise this offline stragglers list S_{t-off} through a categorical analysis of straggler probability by incorporating the behavioural heterogeneity for every individual tasks.

An initial classification of tasks are obtained based on the offline threshold to present an initial hypothesis for a task to behave as energy-aware stragglers or not during the current job execution, which is further subjected to Naïve Bayes classifier further to enhance the preciseness of the dependability of offline stragglers presented by S_{t-off}. A bayes rule scales well for a categorical classification when the dimensionality of the inputs is high. Now, P_{out} delivered by equation 5.11 with an initial task classification based on S_{t-off} is trained as the input set of data for Naïve Bayes classifier to obtain the final list of energy-aware stragglers anticipated in P_{out} during the actual job execution.

Definition. Conditional independence. For a set of predictor tuple $X = \{x_1, x_2, x_3, \ldots . x_k\}$, Naive Bayes classifier assumes that the effect of the value of a predictor x_i on a given class c is independent of the values of other predictors. Based on this conditional independence, the influence of the CPU usage rate and task duration are evaluated individually on a task being classified as energy-aware stragglers based on S_{t-off}. Naïve Bayes classifier estimates the posterior probability using equation 5.25.

$$P(c/x) = \frac{P(x/c)P(c)}{P(x)} \tag{5.25}$$

The overall influence of all the predictors for a given observation is given by equation 5.26.

$$P(C/X) = P(x_1/c)^* P(x_2/c) * P(x_3/c) * \ldots \ldots * P\left(\frac{x_n}{c}\right) = \prod_{k=1}^{n} P(x_k/c) \tag{5.26}$$

where, *P(c/x)* is the posterior probability of *class c* for a given predictor *x*, *P*(*c*) is the prior probability of *class c* (straggler or a non-straggler), *P*(*x/c*) is the likelihood for the probability of class *c* for a given predictor *x,* and *P*(*x*) is the prior probability of predictor *x*. After evaluating the posterior probability, Naïve Bayes classifier categorises a given observation belonging to the class of stragglers C_i or non-stragglers C_j using equation 5.27.

$$C(obs) = \begin{cases} C_i, & for\ P(C_i/X) > P(C_j/X) \\ C_j, & otherwise \end{cases} \tag{5.27}$$

The independent influence of the CPU usage rate, duration and the total CPU core consumption for the two historical usage profile and the predicted profile is evaluated using Naïve Bayes Classifier upon the initial off-line threshold based task classification on P_{out}. So, totally six predictors are trained in the classifier to foresee the outcome of a task to behave either as a non-straggler or an energy-aware straggler during the actual execution of a given job. Since the predictors including CPU usage rate, duration and total CPU consumption are numerical values, all such values should be discretised into respective categories before Naïve estimates the posterior probability. For discretisation, Naïve Bayes Classifier assumes a normal distribution for the observations within a given job, as shown in equation 5.28 to equation 5.30.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5.28}$$

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2\right]^{0.5} \tag{5.31}$$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5.30}$$

where, μ is the mean, σ is the standard deviation and *f(x)* is the normal distribution of the observations achieved based on the probability density function. The mean and standard deviation for every predictor are computed independently based on their prior classification of stragglers and non-stragglers respectively.

5.5.2 Runtime Mitigation

Stragglers in the past are not necessarily behave as a future straggler due to the runtime heterogeneity, simply classifying the energy-aware stragglers based on their historical behaviour may not be sufficient for Cloud executions. A task execution profile is consistent only when it exhibits statistical correlations among different execution instances. During

runtime, a task is identified to be a straggler when the actual execution fails to satisfy the usage rate-duration trade-off, $S^{th}{}_{(\alpha,\ \beta)}$ obtained in the offline analytics can be referred as the threshold point to label a running tasks as potential straggler when task execution breaches both α and β defined by $S^{th}{}_{(\alpha,\ \beta)}$. But the trade-off can only be calculated after the execution is completed. CPU usage rate is an important parameter to identify stragglers during runtime for the benefit of optimum vertical scaling, CPU usage rate can be monitored during runtime. The effectiveness of online analytics lies in the actual time of identification to trigger vertical scaling for the context of real-time elasticity provisioning. Due to the uncertainty in task execution behaviour, the proposed scheme alerts a task as possible energy-aware straggler when the CPU usage rate breaches its corresponding threshold. Long tail stragglers can be identified based on the duration threshold, though presented evaluating the duration threshold for long-tail straggler identification is not within the scope of this chapter. The threshold for CPU usage rate and duration is calculated by subjecting the corresponding threshold values of the non-stragglers obtained in the chosen historical sample analytics, the offline straggler threshold and the threshold of non-stragglers in the prediction output to a two-tier nested exponential smoothing filter, as shown in equation 5.31 and equation 5.32, in such a way that the factual values enjoy a better weighing than the anticipated values.

$$S_{ru} = \{\omega(\omega\ \alpha_1 + (1-\omega)\ \alpha_2) + \ (1 - \ \omega)\ (\omega\ \alpha_{off} + (1-\omega)\ \alpha_{pred})\} \tag{5.32}$$

$$S_{rl} = \{\omega(\omega\ \beta_1 + (1-\omega)\ \beta_2) + \ (1 - \ \omega)\ (\omega\ \beta_{off} + (1-\omega)\ \beta_{pred})\} \tag{5.33}$$

where, $\omega = 2/(n+1)$, α_1 and α_1 are the mean CPU usage rate of the non-stragglers identified in the historical sample analytics, α_{off} is the mean CPU usage thresholds used in the offline straggler identification based on $S^{th}{}_{(\alpha,\ \beta)}$ and α_{pred} is the mean CPU usage threshold of the non-stragglers in the predicted output (based on P_{con}), and β is the duration equivalent respectively.

Since Cloud workloads are dynamic in nature, the heterogeneity among Cloud workloads can be witnessed from two different perspectives: firstly tasks within a job characterising increased CPU usage rate fluctuation with fairly even distribution of task length, and secondly tasks within a job charactering increased CPU usage rate with uneven distribution of task length. Whilst the former do not have an impact on the overall completion time of job, the later can significantly impact the overall job completion time. In other words, the former is a job containing only energy-aware stragglers and the later consists of both the energy-aware

stragglers and long tail stragglers. In general, long tails depend on the runtime factors such as co-located tasks, node efficiency, node-level stragglers etc., thus it is optimum to mitigate the long tail stragglers during runtime rather than attempting to predict them before execution. The characteristics of long tails are postulated to exhibit an execution duration of 50% greater than the computed length threshold S_{rl}, as shown in equation 5.33.

$$S_{lt} = t_l[i] > 1.5 * S_{rl} \tag{5.33}$$

Thus during the actual job execution, the runtime stragglers are identified using equation 5.34.

$$S_t = \begin{cases} t_u[i] > S_{ru}, & for\ energy-aware\ stragglers \\ t_l[i] > S_{lt}, & for\ long\ tail\ stragglers \end{cases} \tag{5.34}$$

where $t_u[i]$ and $t_l[i]$ are the current CPU usage rate and task duration of the i^{th} task within a given job during execution. Since triggering a speculative execution is out of scope of this chapter, as the energy-aware stragglers are focused to be mitigated without creating replicas, the initially provisioned level of resources are postulated to runtime elasticity by provisioning virtual cores upon identifying energy-aware stragglers during runtime. The efficiencies of the proposed methodology in classifying energy-aware stragglers before the start of the execution and identification of energy-aware stragglers during the actual execution is presented as follows.

5.5.3 Performance Evaluation

5.5.3.1 Experiment Setup and Workload Generation

The efficiency of the proposed straggler identification methodology has been evaluated in two different phases. Firstly, the offline analytics has been evaluated to demonstrate the efficiency of the proposed methodology in predicting task-level energy-aware stragglers before the start of the actual job execution. Secondly, the identification accuracy and timeliness efficiency of the proposed methodology in detecting energy-aware stragglers during the actual execution has been evaluated. To facilitate the evaluation of runtime identification, the proposed framework along with the benchmarks techniques are modelled to identify stragglers among jobs being executed in a Kubuntu VM characterising 2 core processor and 1GB RAM, every tasks within a given job are executed on individual threads to reflect the isolated LXCs of a typical Cloud datacentre. The application use case for this evaluation system is chosen from the studied Google production workload traces comprising heterogeneous job types, evaluating all the studied jobs in this chapter. Job execution scenarios of the datacentre are replicated by creating

a multithread job containing *n* number of tasks, all such tasks belonging to a single job are executed across individual threads within a the VM. Tasks within a single job are parallelised and all the threads used for a single job execution are started at the same time to ensure an even start time for all tasks within a given job. Heterogeneity among the running tasks within a single job in terms of their progress rate is achieved by imposing various level of delays and computation intensities among tasks, whereby tasks are executed with varied resource consumption and completion times. After an initial random run of the chosen jobs in the evaluation system, the datacentre equivalent (near identical) execution time and resource intensity of the individual tasks within the respective jobs are achieved by moderating the initially imposed delay within task progress using equation 5.35.

$$d_{req} = (d_c * t_{req}) / t_c \tag{5.35}$$

where, d_{req} is the required delay within task progress to identically replicate the Cloud execution, d_c is the delay imposed during initial run, t_{req} is required task completion time of the Cloud execution and t_c is task completion time of the initial run respectively. Table 5.6 presents the replicated task completion time to that of the typical Cloud execution along with the heterogeneity in resource intensity for a randomly chosen 10 tasks from the studied jobs. The resource intensity percentage of tasks are presented in accordance with the intensity of the other co-located tasks within the same job.

Table 5.6. Replicated Cloud Execution Statistics for Experiments

Job Name	Task Index	Cloud Execution (seconds)	Experiment Duration (seconds)	Imposed Delay (seconds)	Resource Intensity (%)
Job 0	0	156	148.2	5.93	28
Job 1	12	178	177.9	4.1354	43
Job 1	91	161	161.07	2.68	22
Job 2	31	46	45.84	0.88	52
Job 2	191	50	49.80	1.13	44
Job 3	44	40	39.71	0.23	171
Job 4	138	229	228.17	3.93	58
Job 4	420	222	221.47	5.98	37
Job 5	233	1698	1689.97	67.55	25
Job 5	1008	1734	1729.26	86.4	20

5.5.3.2 Compared Methodologies

The straggler detection efficiency of the proposed methodology has been evaluated against the existing state-of-the-art threshold calculation methods including static threshold, progress score based threshold, task progress based threshold, estimated finish time based threshold and a normalised duration based threshold accordingly. The evaluated benchmarks techniques are briefed as follows.

Static Threshold identifies a running task as energy-aware straggler based on a static value obtained from the previous historical run. The point of CPU usage rate at which a task is identified as an energy-aware straggler during its recent historical execution instance is used as the threshold for straggler identification in the current execution.

Progress Score based Threshold classifies a running task as straggler depending on its computed progress score based on equation 5.36 through to equation 5.38, where $PS[i]$ is the progress score of task *i*, *M* is the proportions of task completed and *N* is the remaining proportions of task left for execution. Hadoop default scheduler adopts this progress score based threshold for triggering speculative replicas of straggling tasks when the progress score of a given task is less than 80% of the progress score of the entire job as shown in equation 5.38.

$$PS[i] = M/N \tag{5.36}$$

$$PS_{avg} = \sum_{i=1}^{n} \frac{PS[i]}{n} \tag{5.37}$$

$$For\ Task\ i,\ PS[i] < PS_{avg} * 80\% \tag{5.38}$$

Task Progress based Threshold identifies a running task as straggler when the progress rate of a given task falls behind a defined threshold level than a majority of the remaining tasks within the same job. This progress based threshold is most often used to identify long tails than the energy-aware stragglers. The progress based threshold adopts a typical value of 50%, thereby classifies a running task as a straggler it its progress rate if less than 50% of the average progress rate compared to its siblings, as shown in equation 5.39 and equation 5.40.

$$PR[i] = PS[i]/T \tag{5.39}$$

$$PR[i] < PS_{avg} * 50\% \tag{5.40}$$

Estimated Finish Time based Threshold calculates the estimated time to completion for a given task, and classifies the corresponding task as straggler if its estimated finish time is longer than

a certain percentage (again a typical value of 50% mostly adopted), as shown in equation 5.41, where PS_{avg} is the average progress score of the entire job, *PR[i]* is the progress rate of task *i*, and *TTC[i]* is the estimated time to completion for task *i* based on its progress rate and progress score of the entire job.

$$TTC[i] = \frac{(1-PS[i])}{PR[i]} \tag{5.41}$$

Normalised Duration based Threshold defines a task as straggler based on its normalised duration *nd(t)*, computed as the ratio of task execution time to the amount of work done by a given task *t* within a Job *j*. The corresponding task is classified as a straggler based on equation 5.42, where $nd(t_j)$ is the normalised duration of the entire job and $nd(t_i)$ is the normalised duration of task t_i, and β is a threshold value typically adopted as 1.3.

$$nd(t_i) = \beta \; X \; median\{nd(t_j)\} \tag{5.42}$$

5.5.3.3 Evaluation Metrics

True Positive evaluates the prediction accuracy of the evaluated methodologies in correctly classifying a straggler as a straggler, this is used to demonstrate the efficiency of offline analytics in classifying energy-aware stragglers before the actual execution. Higher the rate or proportions of true positives, better is the classification accuracy.

False Positive evaluates the prediction accuracy of the evaluated methodologies in not classifying a non-straggler as a straggler during offline analytics before execution, classifying a non-straggler as energy-aware straggler might mislead to wrong energy related inferences. Lower the rate or proportions of false positives, better is the classification accuracy.

Detection Time evaluates the timeliness of the evaluated methodologies in early identification of the stragglers during the actual execution of jobs. Early identification of the energy-aware stragglers during runtime benefit the providers to provision elasticity before the execution breaches the initially allocated level of resources. Earlier the identification of stragglers, better is the efficiency of the analytics methodology.

Time-Accuracy Trade-off is used to demonstrate the efficiencies of the evaluated methodologies in balancing the trade-off between detection time and the total proportions of true positives and false positives. Both the early identification of too many non-stragglers as stragglers and late identification of lesser number of energy-aware stragglers might lead to inefficient decision making in the context of energy conservation in datacentres. Whilst the

former causes unwanted provisioning of resources for non-stragglers through elasticity during runtime, the latter might cause task terminations due to the energy-aware stragglers breaching the allocated resource levels.

5.5.3.4 Classification Accuracy

This section present the analysis of the experiment results for the proposed energy-aware straggler classification technique evaluated against the aforementioned benchmarks. The results obtained from every studied jobs have been discussed individually to analyse the impacts of job heterogeneity in the classification efficiency of the studied techniques. Both the offline analytics and runtime identification of the proposed classification technique has been presented and analysed.

Figure 5.16 presents the classification efficiency of the proposed and the benchmark techniques in terms of the total number of correctly identified energy-aware stragglers and the true and false positive proportions of classified tasks respectively. Job 0 characterise an uneven distribution of task length between 30 and 256 seconds among the encompassed tasks, further the resource intensity among the encompassed tasks is extremely heterogeneous. It can be observed from Figure 5.16 that the proposed offline analytics identifies 3 out of 4 energy-aware stragglers even before the execution starts and the proposed runtime analytics threshold identifies all the energy-aware stragglers. In addition, the true positive rate of the proposed runtime threshold is significantly better than the compared benchmark techniques witnessed at 66.66% and further reducing the false positives rate down to 33.33%. Though the static mean threshold identifies all the energy-aware stragglers during runtime, the true positive and false positive rates are witnessed at 36.36% and 63.63% respectively, much worse than the proposed technique. It can further be confirmed that the rest of the benchmark techniques are not efficient in identifying energy-aware stragglers and further characterise significant proportions of false

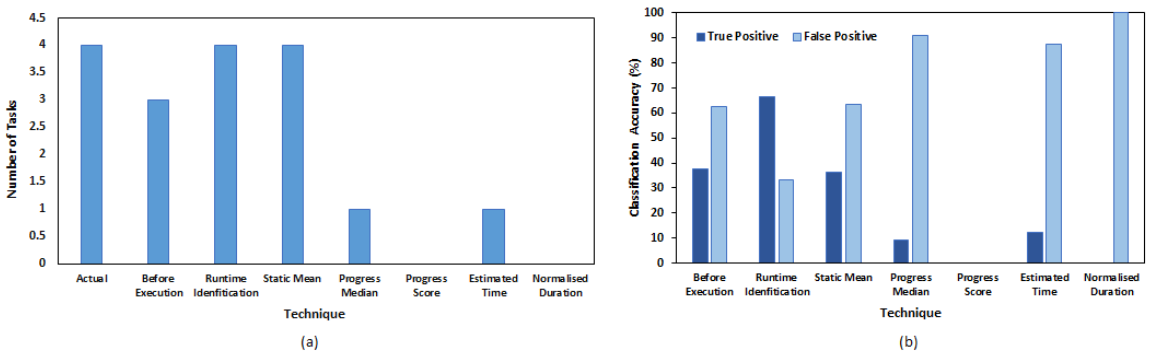


Figure 5.16. Classification Accuracy for Job 0 (a) Identified Stragglers (b) Error Proportions

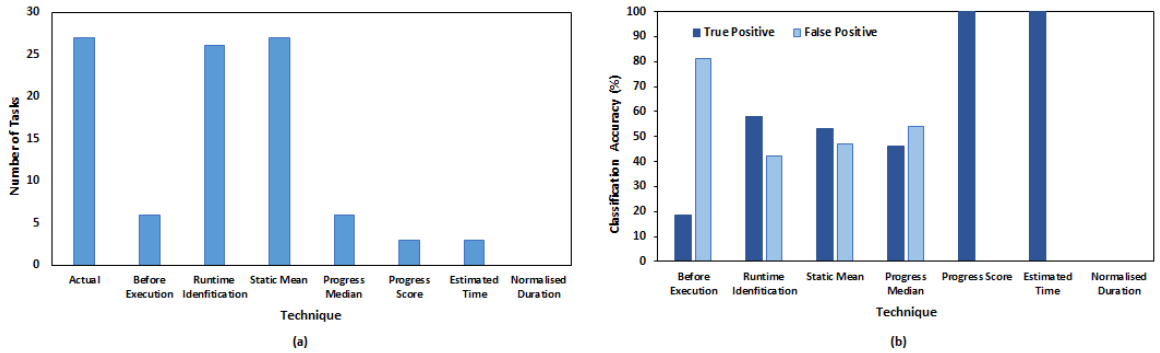


Figure 5.17. Classification Accuracy for Job 1 (a) Identified Stragglers (b) Error Proportions

positives, resulting in wrong classification of non-stragglers as energy-aware stragglers. Thus, it is clear that despite job heterogeneity, the proposed methodology is effective in accurate classification of energy-aware stragglers with minimal proportions of false positives.

Figure 5.17 presents the classification accuracy statistics for Job 1 respectively. Job 1 characterise nearly an even distribution of task length, with only a very few tasks characterise a lower duration and majority of tasks runs between 120 and 200 seconds, in this regard task length is fairly homogeneous across tasks within Job 1. But the resource intensity is extremely heterogeneous across tasks encompassed within Job 1, with the resource intensity ranging from 14% through to 613% among the encompassed tasks. This insists the fact that Job 1 is vulnerable for terminations and/or resource idleness due to this uneven distribution of resource intensity and further characterise a total of 27 energy-aware stragglers out of 100 total tasks. Achieving optimum provisioning of resources across all tasks within Job 1 would be fairly challenging in spite of the presence of 27% of energy-aware stragglers and the inherent heterogeneity of resource intensity. From Figure 5.17, the proposed offline analytics identifies only 6 out of 27 stragglers impacted by job heterogeneity, but 26 out of 27 stragglers are identified by the proposed runtime threshold, and all the energy-aware stragglers are identified by the static mean technique and the remaining techniques are not efficient in identifying energy-aware stragglers during the actual execution. Despite identifying all the stragglers, the static mean technique exhibits a higher percentage of false positives witnessed at 47.05% than the proposed technique exhibiting a false positive rate of 42.22% respectively. Furthermore, the true positive rate of the proposed and the static mean threshold are witnessed at 57.7% and 52.9% accordingly. Thus the static mean technique is vulnerable to wrongly classify non-stragglers as energy-aware stragglers by a margin of 5% more than the proposed technique despite identifying just one additional straggling task. In this sense, it can be postulated that

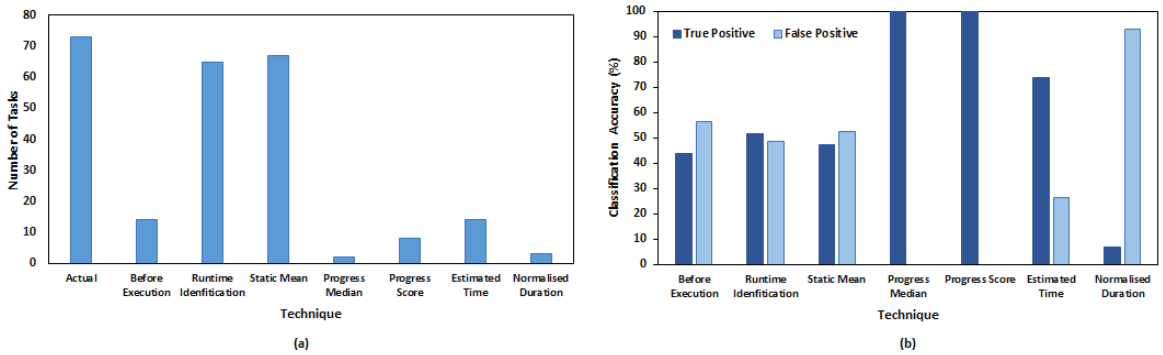


Figure 5.18. Classification Accuracy for Job 2 (a) Identified Stragglers (b) Error Proportions

the proposed technique is effective in correctly classifying energy-aware stragglers with reduced false positives despite task resource consumption heterogeneity.

Figure 5.18 presents the classification accuracy statistics of Job 2 for the proposed and benchmark techniques. Task length of the encompassed task is evenly distributed ranging between 45 and 70 seconds, and further characterise fairly even distribution of task resource intensity ranging between 10 and 73%, a minority of tasks exhibit lower level of resource intensity and majority of tasks characterise evenly distributed resource intensity. Job 2 is homogeneous both in terms of task length and task resource intensity. Despite being homogeneous, Job 2 can still characterise energy-aware stragglers whenever a given task runs longer and continues to consume more resources. From Figure 5.18, the proposed offline analytics identifies only 14 out of 73 actual stragglers before the execution starts, and the runtime threshold identifies 65 and the static mean threshold identifies 67 energy-aware stragglers respectively. Though, the classification accuracy of the proposed runtime threshold is better than the static mean threshold, with the true positive rate witnessed at 51.58% and 47.51% for the proposed runtime threshold and the static mean threshold respectively. Though the progress median and progress score based thresholds exhibits flawless true positive rates, they only managed to identify 2 and 8 energy-aware stragglers respectively, leaving majority of the stragglers unnoticed. The true positive rate of the estimated finish time based threshold is better than the proposed technique witnessed at 73.68%, again it only manages to identify 14 stragglers out of 73. The normalised duration based threshold identifies only 3 stragglers and exhibits a false positive of 92.85% at an irresistible margin.

Figure 5.19 presents the classification accuracy statistics of the evaluated techniques for Job 4. Job 4 is an interesting sample since it comprises more than 40% of energy aware stragglers, which means nearly half of tasks are energy intensive than the remaining half set of tasks within

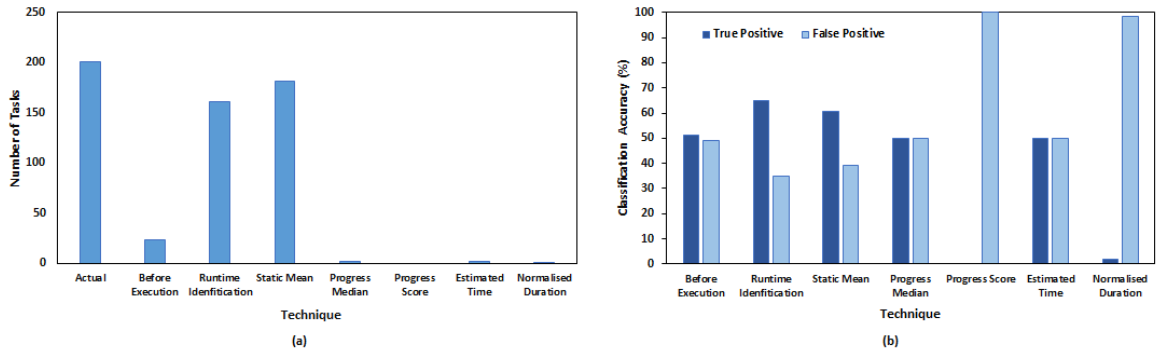


Figure 5.19. Classification Accuracy for Job 4 (a) Identified Stragglers (b) Error Proportions

Job 4. In this regard Job 4 itself can be regarded as an energy-intensive job rather than comprising energy-aware stragglers. However, it is still worthwhile to analyse the classification efficiencies of the studied techniques to project the scope for reducing the energy impacts of such a job kind. Task length distribution of tasks within Job 4 is extremely heterogeneous, ranging between as low as 4 seconds and 270 seconds. Furthermore, the resource intensity is also heterogeneous, ranging between 3% and 105%. From Figure 5.19, it is evident that the proposed classification technique outperforms the benchmark techniques in terms of the trade-off between identified stragglers and reduction in false positives, with the proposed technique identifying 161 out of 201 energy-aware stragglers at a true positive rate of 64.9%, against the static mean threshold identifying 171 stragglers at a true positive rate of 60.6%. The total number of correctly identified stragglers by the remaining techniques are very insignificant, and the benefits availed to resource provision is little of merit.

Figure 5.20 presents the classification efficiency statistics for Job 5. Job 5 is fairly homogeneous in terms of both task length (at an average of 1578 seconds) and resource intensity distribution, comprising only around 2% of energy-aware stragglers. It is worthy of note that the energy-aware stragglers do not exhibit any significant deviation of resource intensiveness than the non-stragglers within Job 5. In this sense, Job 5 can be regarded as mostly encompassing non-stragglers with the minority of the energy-aware stragglers do not having any notable impacts upon resource provision and consumption trend, this has reflected only 16% of resource idleness within Job 5 as discussed earlier. From Figure 5.20, all the classification techniques are exhibiting significant proportions of false positives, with the proposed technique performing better than the remaining techniques exhibiting a true positive rate of 16.48% as opposed to the static mean technique exhibiting 4.52% of true positives and the rest of the techniques identifying none of the stragglers. This could be because of the

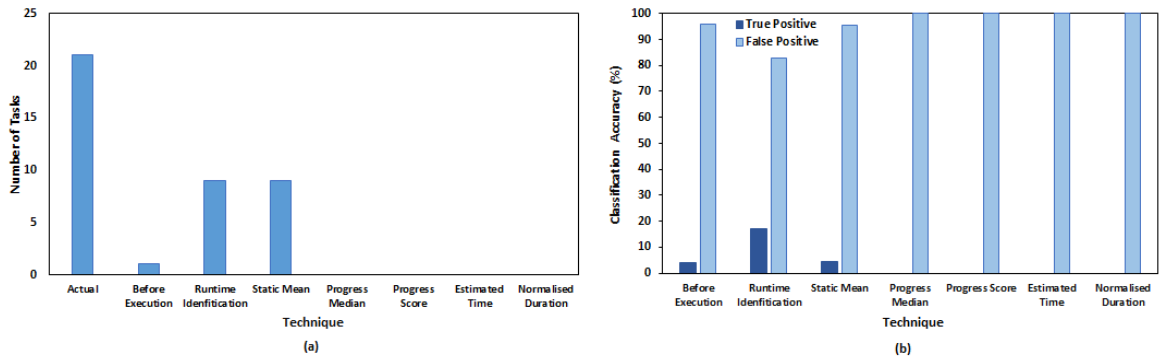


Figure 5.20. Classification Accuracy for Job 5 (a) Identified Stragglers (b) Error Proportions

homogeneity exhibited by tasks encompassed within Job 5 and the prediction techniques fairly believe that tasks within Cloud jobs are mostly heterogeneous. Furthermore, considering the impacts of the energy-aware stragglers upon resource consumption, Job 5 can be regarded as comprising no energy-aware stragglers, suggested by the analytics of the chosen two historical samples of Job 5 exhibiting only around 27 and 18 energy-aware stragglers respectively, with both the historical samples comprising 1050 tasks. Thus, it can be postulated that the straggler classification should also consider the historical distributions of the proportional presence and impacts of the energy-aware stragglers upon energy efficiency.

Figure 5.21 presents the classification efficiency of the proposed technique in terms of the true positive and false positive rates averaged across the studied jobs (excluding Job 5) presented against the accuracy proportions of Job 3. Job 3 is a job sample without sufficient historical traces which restrains the proposed analytics both in terms of resource level prediction and straggler classification. Whilst both the day-of-the-week and time-of-the-day samples have been available for analysis for the rest of jobs, Job 3 presents only the day-of-the-week sample for offline analytics. From Figure 5.21, the effects of this limited availability of the appropriate historical samples are clearly evident, with the true positive rate for the rest of jobs is witnessed

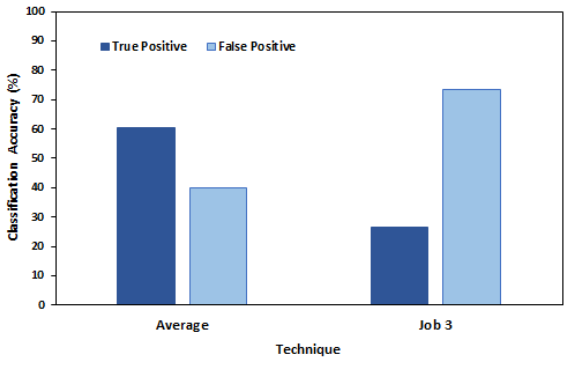


Figure 5.21. Classification Efficiency of the Proposed Technique

at 60.23% which reduces to only around 26.53% for Job 3. This proves the power of offline analytics based on the most appropriate historical samples for both prediction and straggler classification. Apart from this, it can be concluded that the proposed straggler analytics technique performs significantly better than the evaluated benchmark techniques. It is worthy of note that most of the evaluated benchmark techniques are focused on classifying tasks as stragglers whenever tasks exhibit prolonged execution duration than majority of tasks within the same job, leaving the resource intensity of tasks unnoticed restrains their capacities of identifying tasks with increased resource intensity.

The classification effectiveness of the proposed analytics technique can be attributed to the fact that it incorporates the combinational effects of both task duration and CPU usage rate of respective tasks. The energy-aware straggler prediction before execution is still believed to be ambitious to date, in which sense it can be postulated that the proposed straggler classification methodology is efficient in identifying task level stragglers before the start of the actual job execution.

5.5.3.5 Time-Accuracy Trade-off

Figure 5.22 presents the classification time of energy aware stragglers for Job 0, Job 1 and Job 4 for the proposed runtime threshold, static mean threshold and the progress based threshold respectively. It can be observed that the classification time of the proposed threshold is slightly higher than the static mean threshold for the corresponding tasks for Job 0 and Job 1, and the identification time of the progress based threshold is significantly on the higher side. There are two immediate implication, firstly late classification of the energy-aware stragglers into task execution is of no benefit for vertical scaling or speculative execution and secondly early classification of tasks might lead to needless vertical scaling or speculative execution. So, an effective classification technique should identify energy aware stragglers at an optimum time into execution.

The static threshold is vulnerable to early identification of non-stragglers as energy-aware stragglers causing increased false positives, and the progress score threshold is vulnerable for late identification in the cases of Job 0 and Job 1 and not effective in identifying stragglers in the case of Job 4. Thus it can be concluded that the proposed runtime threshold is effective in identifying energy-aware stragglers at an optimum time into execution as well as avoiding the proportions of false positives.

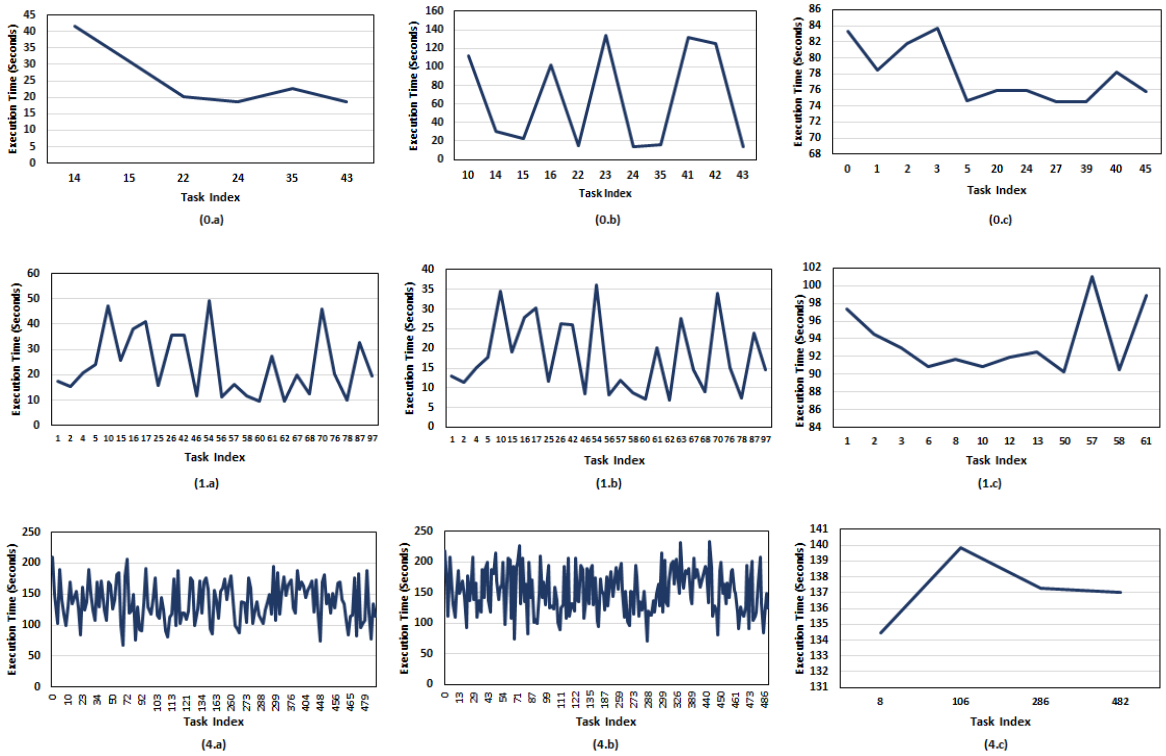


Figure 5.22. Classification Time (a) Proposed Threshold (b) Static (c) Progress Based {(0) Job 0 (1) Job 1 (4) Job 4}

5.6 Resource Provision Optimisation Module

The resource provisioning optimisation problem can be witnessed from two perspectives: firstly commitment of resources levels should not be vulnerable to cause resource idleness, and secondly the actual task execution should not breach the provisioned level of resources causing task terminations. From the resource prediction perspectives, the former usually results from the over-estimated resource levels and the latter results from the under-estimation. This section is focused on optimising the under-estimated resource levels delivered by the proposed resource estimation module in consideration of the straggler classification outcome. A resource usage and straggler-aware over commitment policy of resource levels has been proposed to meet the objectives of termination less execution with minimal resource idleness.

5.6.1 Task Categories

Based on the early discussed behaviours and characteristics of tasks within jobs, tasks are classified as non-stragglers and energy-aware stragglers for resource provision. A static resource provisioning policy for these two categories my not scale well for energy efficiency, since the resource consumption level of non-stragglers and stragglers are significantly different. Every individual task within a given job should be dynamically provisioned with a

resource level depending on their requirements. Tasks identified as non-stragglers and tasks classified as stragglers in the offline analytics facilitate deciding their resource provisioning levels before the actual execution starts. But the energy-aware stragglers identified during the runtime needs dynamic scaling of resources levels during runtime which can be achieved through dynamic vertical scaling.

Case (1): Non-stragglers

Non-stragglers are expected to execute without any notable abnormal resource consumption level during the actual execution, thus it is recommended to initially rely on the resource usage estimation whilst provisioning resource levels to non-stragglers. Thus the anticipated usage of non-stragglers is expected as per the usage prediction output, as in shown equation 5.43, where $R_{Pred}[i]$ is the initially predicted resource usage level for task *i* within a given job.

$$R_{ns(c)}[i] = R_{Pred}[i] \tag{5.43}$$

Case (2): Energy-Aware Stragglers

The resource consumption level for energy-aware straggles are usually expected to exceed the level of non-stragglers by a significant margin. Further to the usage prediction of stragglers, it is recommended to rely on the historical execution instances to determine the resource provisioning level of the classified energy-aware stragglers. Thus the anticipated resource consumption level of stragglers $R_{st(c)}$ would depend on the identified maximum CPU usage among the two historical samples and the resource prediction as shown in equation 5.44, where $R_{s1}[i]$ and $R_{s2}[i]$ are the corresponding consumption of the given straggling task *i* in the two historical execution profiles respectively.

$$R_{st(c)}[i] = \max(R_{s1}[i], R_{s2}[i], R_{Pred}[i]) \tag{5.44}$$

The actual usage level of tasks are affected by various runtime factors and also task heterogeneity restrains from using the resource levels as depicted by the above two equations. The extravagant heterogeneity of tasks within a job and the dynamic datacentre runtime environment is naturally insisting the need for overcommitting the resource levels for achieving termination-less execution, as this has a direct impact on QoS. To this end, a dynamic over-allocation policy has been further proposed for every individual tasks within a given job to further optimise the resource provisioning levels recommended by equation 5.43 and equation 5.44.

5.6.2 Over Commitment factor

A over commitment of factor γ is adopted for all task categories, whereby it is proposed to overcommit the resources by a margin of γ to the level insisted by the equation 5.43 and equation 5.44 accordingly. This over commitment factor is dynamically chosen for every individual tasks depending on three important factors: task consistency based on the process efficiency of tasks in the historical execution profiles, process capacities of tasks determined for the actual execution based on the resource prediction output, and task classification delivered by the straggler classification module. Three classes of reliability has been adopted for every individual factors determining the over commitment factor as high, medium and low, with high insisting the reliability measure is highly reliable through to low being less reliable for a task to behave normally during execution. Over allocation proportions for the three defined level of reliability classes are adopted as 30%, 40% and 50% respectively for high, medium and low classes of task reliability. Based on the defined three over-commitment factors and the respective reliability class of tasks, every individual tasks will be dynamically evaluated to moderate the over commitment factor, as explained below.

5.6.2.1 Process Efficiency

The reliability of resource consumption consistency of tasks is estimated based on the process efficiency of task execution during its historical execution instances. Despite the allocated level of resources, it is usual for the process efficiency of certain tasks to fall behind than a majority of tasks within the same job, impacted by task nature.

Definition: Process Efficiency. Given a Job set J with n number of tasks $J=\{T_1, T_2, T_3, \ldots\ldots, T_n\}$, executed for a duration $D=\{t_1, t_2, t_3, \ldots, t_n\}$ consumed a resource level of $R=\{r_1, r_2, r_3, \ldots . r_n\}$, where task T_i executed for a duration of t_i, and consumed r_i amounts of resources, task process efficiency Pe_i of task T_i is defined as the ratio of the executed duration to the amount of resources consumed during the actual execution, where job duration and resource consumption of job are given by $max(t_i)$ and $\sum_{i=o}^{n} r_i$ respectively. Task process efficiency and job process efficiency can be computed as shown in equation 5.45 and equation 5.46.

$$Pe_i = \frac{t_i}{r_i} \tag{5.45}$$

$$Pe_j = \frac{\sum_{i=0}^{n} Pe_i}{n} \tag{5.46}$$

In general, task process efficiency reflects the effectiveness of the resource provisioning level for every individual task within a single job. If certain tasks are allocated with less process efficiency within a single job, such tasks are vulnerable to behave either as long tails or energy-aware stragglers resulting from the node-level process efficiency. It is postulated that task efficiency of a given task is extremely low during an execution if its task efficiency falls below 50% (a value typically adopted for classifying straggler in the literature) of job process efficiency as shown in equation 5.47.

$$T_{le}[i] = \begin{cases} J[t_i] & if\ P_{ei}[i] < 0.5 * Pe_j\ is\ true \\ nil & otherwise \end{cases} \quad (5.47)$$

Now, based on task process efficiency of the two historical samples, the consistency of a given task is measured among the two historical usage profiles to evaluate the reliability class for every individual tasks within a given job. A task is considered to be highly reliable when the process efficiency of the corresponding task stays internally consistent among the two historical usage profiles. If task process efficiency is not consistent in both the two historical samples, it is less reliable and if task process efficiency is consistent only in one of the two historical samples it characterise a reliability class of medium.

5.6.2.2 Process Capacity

It is worthy of note that task efficiency is not only affected by the node level stragglers, tasks those are more resource intensive within a single job naturally demands more resources than the other co-located tasks

Definition: Process capacity. Given a Job set *J* with *n* number of tasks $J=\{T_1, T_2, T_3, \ldots\ldots, T_n\}$, predicted to run for a duration $D=\{t_1, t_2, t_3, \ldots, t_n\}$ anticipated to consume a resource level of $R=\{r_1, r_2, r_3, \ldots . r_4\}$, where task T_i is expected to run for a duration of t_{pi}, and to consume r_{pi} amounts of resources, task process capacity Pc_i of task T_i is defined as the ratio of the anticipated duration to the predicted level of resource usage, where the anticipated job duration and resource consumption of job are given by $max(t_i)$ and $\sum_{i=o}^{n} r_i$ respectively. Task process capacity and job process capacity can be computed as shown in equation 5.48 and equation 5.49.

$$Pc_i = \frac{t_{pi}}{r_{pi}} \quad (5.48)$$

$$P_{cj} = \frac{\sum_{i=0}^{n} P_{ci}}{n} \quad (5.49)$$

Process capacity will determine the efficiency of the provisioned level of resources in processing job within the determined time-scale. Individual tasks with process capacity below

job process efficiency are vulnerable to affect the overall job progress by potentially acting as long tail and/or energy-aware stragglers. Higher the value of $Pc_{i,}$ greater is the process efficiency and shorter is the execution duration for a given task. Lower the value of $Pc_{i,}$ higher is the possibility of the corresponding task to behave as a long tail straggler. Similar to task process efficiency, tasks with low process capacity $T_{lc}[i]$ are identified using equation 5.50.

$$T_{lc}[i] = \begin{cases} J[t_i] & if\ P_{ci}[i] < 0.5 * Pc_j\ is\ true \\ nil & otherwise \end{cases} \quad (5.50)$$

Tasks with lower process capacity identified by equation 5.50 are considered as less reliable for further optimise their initially estimated resource provision level.

5.6.2.3 Task Category

Furthermore, the over-commitment factor will give special emphasis to the offline straggler classification with the objective of overcommitting the resource levels for the predicted off-line stragglers. Over committing the resource level for stragglers help to resolve the resource constraints of the energy-aware and long-tail stragglers such that job delay caused by prolonged execution or terminations of the stragglers caused by resource scarcity can be significantly reduced. For a given task, if it behaved as non-straggler in the two historical usage profiles and further classified to be a non-straggler based on the resource prediction, it is highly reliable to stay as a non-straggler during the actual execution. For a given task, if it is classified as energy-aware straggler by the offline analytics, then it is less reliable to behave as a non-straggler during actual execution. For a given task, if it has behaved as energy-aware straggler during both its historical execution instances not classified as energy-aware straggler by the off-line analytics, then it is less reliable to behave as a non-straggler during actual execution. For a given task, if it has behaved as energy-aware straggler in either one of the two historical samples and not classified as energy-aware straggler by the off-line analytics, then it characterise a medium reliability to behave as non-straggler during actual execution.

5.6.3 Over-commitment Percentage

The above discussed factors determining the over commitment level of resources for task execution are expected to influence each other. For instance, an energy-aware straggler task with process efficiency consistency will enjoy a higher reliability weightage at step (5.6.2.1) and a lower reliability weightage at step (5.6.2.3), such that the level of over commitment postulated in the two respective levels are expected to cancel out the effects of each other so as to deliver a final optimised level of over-commitment. Thus the optimised over-commitment

percentage of resources OCF for a given task is achieved as an average of the over commitment level determined by the above three factors as shown in equation 5.51, where, ocf_a, ocf_b, ocf_c are the over commitment level of resources computed based on the three influencing factors respectively. The over-commitment protocol is illustrated in Figure 5.23.

$$OCF = \frac{1}{3}\,(ocf_a + ocf_b + ocf_c) \tag{5.51}$$

The final level of recommended resource provision for a task *i* is given by equation 5.52.

$$R_p[i] = \begin{cases} R_{ns(c)}[i] * OCF, & for\ non\ stragglers \\ R_{st(c)}[i] * OCF, & for\ stragglers \end{cases} \tag{5.52}$$

Input: U_1 and U_2 → historical usage profile of Job *J* encompassing execution duration *D* and resource consumption *R*.

S_{t-off} ← list of energy-aware stragglers delivered by offline analytics

S_{t-h1} and S_{t-h2} ← list of energy-aware stragglers in the two historical samples

P_{out} → Usage prediction for job *J* encompassing the anticipated duration *D* and predicted resource consumption level *R*.

Output: OCF → Optimized resource consumption level for all tasks within Job *J*

(1) Pe_i ← calculate task process efficiency $\underbrace{Pe_j}_{\forall t_i \in J}$ for U_1 and U_2

(2) Pe_j ← calculate job process efficiency for J is U_1 and U_2

(3) if $P_{ei}[i] < 0.5 * Pe_j$, then $P[i]$ in $T_{le}[i]$

(4) Pc_i ← calculate task process capacity $\underbrace{Pc_j}_{\forall t_i \in J}$ for P_{out}

(5) Pc_j ← calculate job process capacity for J is U_1 and U_2

(6) if $P_{ci}[i] < 0.5 * Pc_j$, then $P[i]$ in $T_{lc}[i]$

(7) if $T_{le}[i]$ of U_1 contains T_i
  then initialize $ocf_a = 1.4$
    if $T_{le}[i]$ of U_2 contains T_i
      then $ocf_a = ocf_a + 0.1$
    else retain ocf_a
  else if $T_{le}[i]$ of U_2 contains T_i
    then initialize $ocf_a = 1.4$
  else initialize $ocf_a = 1.3$

(8) if T_i contained in $T_{lc}[i]$
    then initialize $ocf_b = 1.5$
  else $ocf_b = 1.3$

(9) if T_i contained in S_{t-off}
      then initialize $ocf_c = 1.5$
      break
    else if T_i contained in S_{t-h1}
    then initialize $ocf_c = 1.4$
      if T_i contained in S_{t-h2}
        then $ocf_c = ocf_c + 0.1$
      else retain ocf_c
  else if T_i contained in S_{t-h2}
    then initialize $ocf_c = 1.4$
  else if initialise $ocf_c = 1.3$

(10) OCF ← calculate ocf as an average of ocf_a, ocf_b, and ocf_c

Figure 5.23. Over Commitment Protocol

5.7 Performance Evaluation

This section presents the efficiency of the proposed resource optimisation methodology, evaluated from the perspectives of reducing the server energy expenditures, reducing task terminations and mitigating the presence of energy-aware stragglers. The proportions of tasks benefitting from resource conservation and task proportions subjected to excess resource provision, and the failure probability in terms of under-estimated resource levels have been verified by theoretical analysis, and further the server energy expenditures of the actual and proposed provision of resource levels have been evaluated experimentally through simulations. GreenCloud [151, 152] simulation platform has been used to simulate tasks within jobs. Tasks are devised to be scheduled by the green scheduler of the GreenCloud across a selected range of servers. For energy management, DVFS has been enabled on the both the physical servers and the processing VMs to dynamically adjust the internal scheduling of tasks, so that the VMs try to extend task execution time to exploit DVFS in accordance to the workload intensity. The resource intensity of tasks has been reflected in task size (presented in bytes) and the proportional idleness has been reflected by imposing corresponding load on the server depending on the idle proportions incurred in the actual and proposed level of resource provisioning accordingly.

5.7.1 Resource Conservation

Figure 5.24 presents the projected proportions of resource conservation based on the proposed methodology for the analysed jobs against the respective proportions of idle resources during their actual execution. The resource conservation statistics are presented only for the over-estimated tasks within the respective jobs, since under-estimated tasks does not provide any proportions of resource conservation rather the resource levels of under-estimated tasks needs to be further increased. From Figure 5.24, a proportion of 82.24%, 79.93%, 84.005%, 62.54%

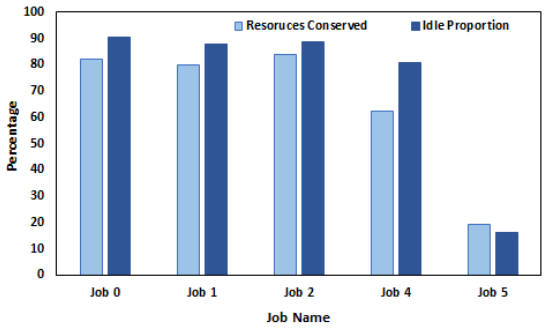


Figure 5.24. Resource Conservation Statistics

and 19.12% resources can be conserved against their initially provisioned level of resources respectively for tasks within Job 0, Job 1, Job 2, Job 4 and Job 5. It can be observed that a significant proportion of initially provisioned resources can be conserved exploiting the proposed analytics methodology. The resource conservation statistics are presented as an average across all the encompassed tasks within their prospective jobs.

Thus the projected proportion of resource conservation is not static across the encompassed tasks within a given job, every tasks within a given job enjoys different proportions of resource conservation. It is interesting to see that an increased proportions of resource conservations are projected for Job 5 than their actual level of resource idleness. This is because the projected proportions of resource conservation is presented only for those over-estimated tasks. It is possible that the resource estimation of stragglers driven by the usage profiles of non-stragglers might be under estimated. In addition, the projected proportions of resource conservation is not similar across the studied jobs, affected by job heterogeneity. The consequence of the presence of stragglers, job heterogeneity and under prediction probabilities are discussed as follows.

5.7.2 Straggler and Heterogeneity Consequence

Figure 5.25 presents the proportions of tasks achieving resource conservation and resource wastages within the studied jobs based on the proposed methodology. Whilst the conserved task proportions reflect the achieved reduction in the originally provisioned resource level for tasks within jobs along with the conversation percentage achieved, the wasted task proportions depict tasks for which the resources are over-estimated than the actual level along with the percentage of resources wasted for tasks within the studied jobs. None of the over-estimated tasks within Job 0 experience resource wastages, (i.e.,) the proposed resource provisioning level are much less than the actual level for all tasks within Job 0. Thus an average of 82.24% of resource levels are conserved across all tasks encompassed within Job 0, as presented in

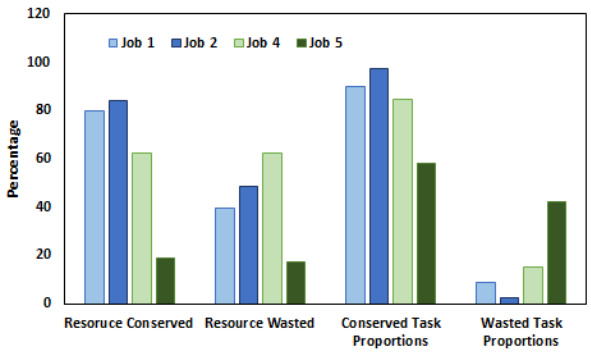


Figure 5.25. Straggler and Heterogeneity Consequence Statistics

Figure 5.24. From Figure 5.25, resource conservation is achieved for 90% of tasks with an average of 79.93% of conservation in Job 1, 97.43% of tasks with an average of 84% of conservation Job 2, 84% of tasks with an average of 62.54% of conservation in Job 4 and 58% of tasks with an average of 19.12% of conservation in Job 5 respectively.

Conversely, the proposed methodology has over-estimated the resource level than the originally provisioned level for 9% of tasks with an average of 39.35% of wastages for Job 1, 3% of tasks with an average of 48.36% of wastages for Job 2, 15% of tasks with an average of 62.65% of wastages for Job 4 and 42% of tasks with an average of 17.3% of wastages for Job 5 respectively. It is clearly evident that resource conservations are achieved for a majority of task proportions with significant reduction in the actually provisioned resource levels and a minority of task proportions are suffering marginal resource wastages. Job 5 is a exhibiting a different trend, since only around 16% of resource idleness are actually witnessed during job execution. Furthermore, the proportional presence of stragglers are very insignificant within Job 5 and also energy-aware stragglers do not show any notable increase in their resource consumption trend from that of non-stragglers. To this end, it can be postulated that jobs exhibiting fair resource consumption trend with less than 20% resource idleness can be ignored for further analysis for exploring the scope of resource conservation.

As expected, the proportional presence of energy-aware stragglers do have a considerable impact upon the efficiencies of resource provision optimisation. Job 0 with 8% of energy-aware stragglers is not suffering any excess resource wastages resulting from over estimation. From earlier analysis, a proportion of around 27%, 36% and 40% of energy-aware stragglers are witnessed in Job 1, Job 2 and Job 4 respectively, which has reflected in the wasted proportions of resources within the respective job. Such observations are reflecting that increasing proportions of energy-aware stragglers within jobs are resulting in respective increase in the proportions of resource wastages through over-estimation of resource levels based on the proposed analytics methodology.

However, the proportions of task for which their resource levels are over-estimated are still insignificant, whereby the impacts of the over-estimation factor has been maintained at the minimum possible level to reduce their energy impacts upon the overall energy consumption.

5.7.3 Failure Probability

Figure 5.26 presents the failure probability proportions of tasks within their respective job during the actual execution, resulting from the under-predicted resource levels by the proposed methodology leading to resource level breaches during job execution. The effectiveness of the proposed methodology is evaluated from two different perspectives: firstly, the failure probability has been evaluated for all tasks encompassed within a given job despite their resource consumption behavior, and secondly the actual and classified non-stragglers within jobs are isolated to evaluate the failure probability of non-stragglers. From Figure 5.26, around 11% of total tasks are vulnerable for resource related terminations which reduces to 9% for non-stragglers for Job 0. Similarly, the failure probability has been witnessed at 32% for all tasks and 5% for non-stragglers for Job 1, 30% for all tasks and 6% for non-stragglers for Job 2, 19% for all tasks and 7% for non-stragglers for Job 4 and 23% for all tasks and 20.64% for non-stragglers for Job 5. Again, Job 5 is not performing as expected exhibiting an increased proportions of job failure probability for all tasks and non-stragglers. This can again be attributed to the fair execution profiles of Job 5, again it can be recommended that jobs with fair execution trend do not project the scope for further reduction in resource expenditures. It is clearly evident that the failure probability for non-stragglers are significantly lower than the entire task group including stragglers, exhibiting the efficiencies of the proposed methodology in reducing the failure probability for non-stragglers. Stragglers are naturally expected to consume more resources and stragglers being identified during runtime are suggested to be provisioned with additional resources through vertical scaling to avoid terminations during actual execution. Failure probability can be reduced by increasing the margin of the over-commitment factor proposed in the over-estimation protocol, in such a way that the over-commitment factor for the discussed categories can be scaled up to further reduce the proportions of under-estimation of resources. However, increase in the over-commitment

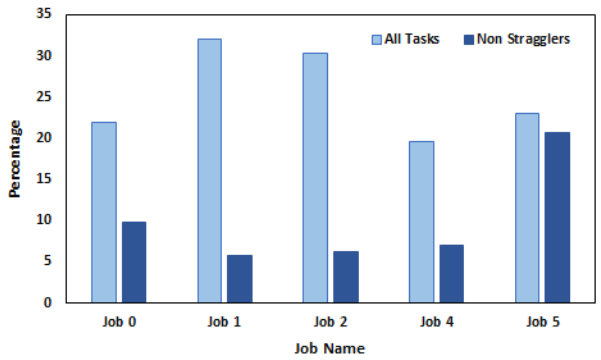


Figure 5.26. Failure Probability Proportions

margin will increase the resource provisioning levels of other task groups for which the resources have already over-estimated. This may lead to a reduction in the overall conservation of resources estimated by the proposed methodology.

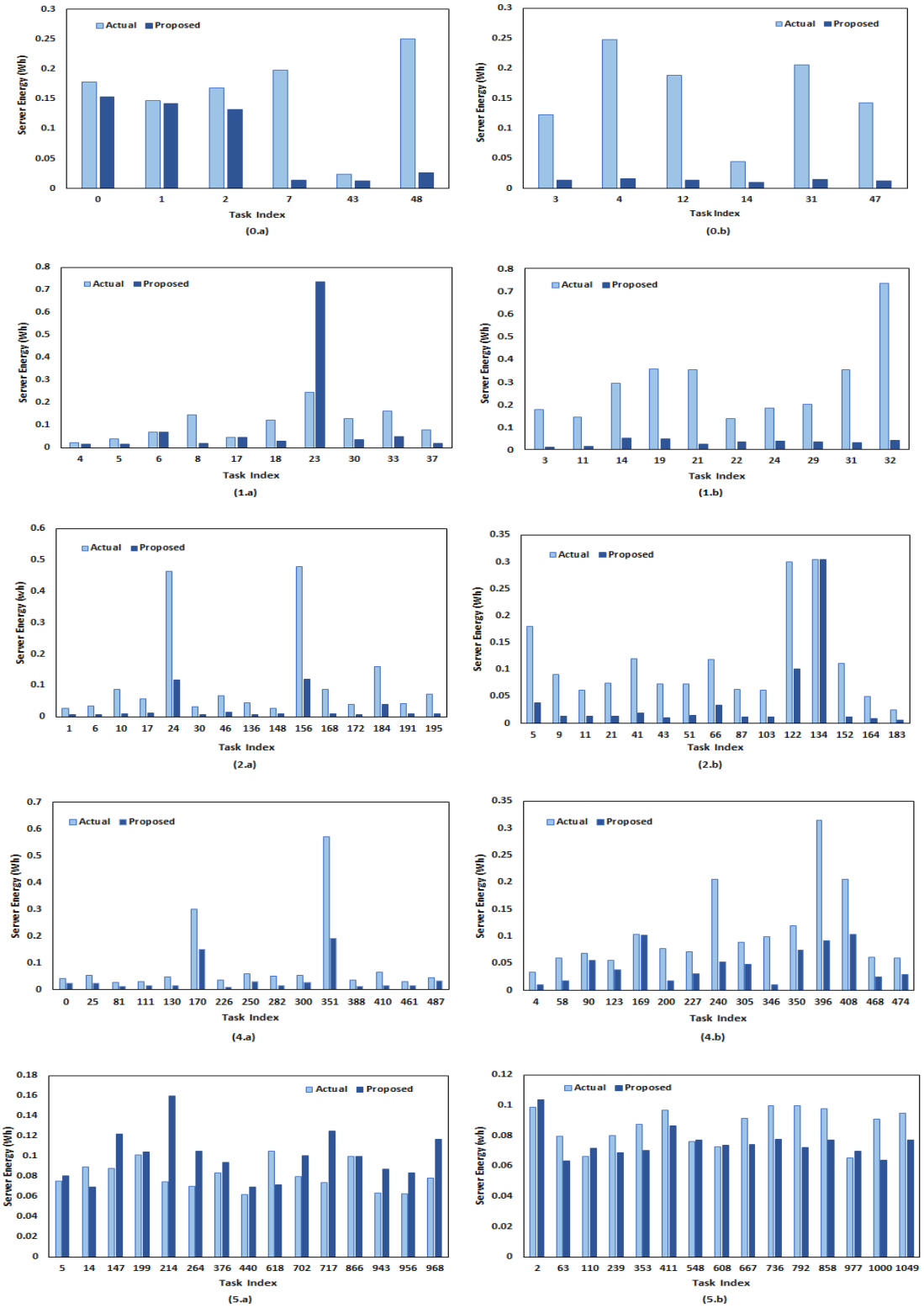


Figure 5.27. Server Energy Expenditures (a) Classified and Actual Stragglers (b) Classified Non-Stragglers {(0) Job 0 (1) Job 1 (2) Job 2 (4) Job 4 (5) Job 5

5.7.4 Energy Efficiency Analysis

This section is intended to exhibit the effectiveness of the proposed methodology in conserving the server energy expenditures achieved through the reduction in the amounts of resources spent on job execution. Figure 5.27 presents the energy expenditure statistics obtained from the simulation of Job 0 through to Job 5, for the actually assigned amounts of resources and the proposed resource provision. The effectiveness of the proposed methodology has been presented for the classified energy-aware stragglers and non-stragglers respectively in Figure 5.27, for an equivalent selection of random tasks representing the two group of classification. On a coarse-grain, it is clearly evident that the proposed methodology performs better in reducing the server energy expenditures for tasks classified as non-stragglers than those of classified as energy-aware stragglers within a given job. This is because of the straggler classification effectiveness of the proposed methodology well before the start of the actual execution. For instance in the case of Job 0, tasks 0, 1, 2, 7, 43, and 48 are classified as energy-aware stragglers by the proposed offline analytics. Though the proposed methodology proposes only a marginal reduction for tasks 0, 1 and 2 and 43, a significant reduction in server energy expenditures can be achieved for tasks 7 and 48 within the classified energy-aware straggler group. Despite being classified as energy-aware stragglers, the resource provisioning level of task 7 and 48 has been moderated by the proposed resource optimisation analytics strategy involving the analysis of process efficiency and process capacity of the respective tasks, both of which moderating the influence of these tasks being classified as energy-aware stragglers. For non-straggler tasks within Job 0, a significant proportions of energy conservation has been achieved fairly across tasks. Though a significant reduction in the energy expenditures has been achieved for non-stragglers within Job 1, only a marginal reduction in server energy expenditures has been achieved by the proposed methodology for most of the classified energy-aware stragglers.

Furthermore, task 23 within Job 1 is a typical example where the proposed methodology is estimating the resource provisioning level that exceeds the actual level of resources originally provisioned, in which sense this would incur energy wastages by incurring more proportions of resource idleness. This is attributed to the fact task 23 being inconsistent in its behavior during historical execution profiles in terms of its straggling behavior and process efficiency and further the process capacity being less reliable for the current execution based on the estimated resource consumption level, all these factors have contributed to the over-estimation of the resource levels for task 23. However, considering the overall impacts of this over-

estimation upon the server energy conservation for all tasks within Job 1, the excess energy consequence of task 23 is very insignificant. Job 2 is a typical example where task 134 has been estimated to consume more resources which is then turning out to be a non-straggler during the actual execution, again this is due to task 134 being less reliable in this straggler behavior, process efficiency and process capacity. But, a significant reduction in sever energy conservation has been achieved for both task groups based on the proposed methodology.

The influence of an increased proportions of straggling tasks within Job 4 can be observed from Figure 5.27, where only a marginal reduction in server energy expenditures has been achieved for most of the classified energy-aware stragglers. Furthermore, the achieved reduction in server energy expenditures for non-stragglers is much lower than those of the other studied jobs, however still better than those of the actually provisioned level of resources for non-stragglers within Job 4. Job 5 has been suggested not to reduce the actually provisioned resource levels based on the theoretical verification, driven by the effects of straggler consequence, failure probability and importantly the witnessed level of reduced resource idleness (being less than 20%) during the historical execution profiles. The simulation experiments are also presenting us with similar inferences, since most of the energy-aware stragglers have been over-estimated than the actual level of resource provision. Also, only a marginal reduction in the server energy expenditures can be achieved for the group of non-stragglers. However, reduction in server energy is still possible with the group of classified energy-aware stragglers being insignificant than those of the non-stragglers within Job 5.

Figure 5.28 presents the overall energy conservation probability across all the encompassed tasks within the studied jobs, comparing server energy expenditures incurred by the proposed level of resource provision against the actually provisioned level of resources. The statistics presented in Figure 5.28 has been moderated with the proportions of tasks with energy

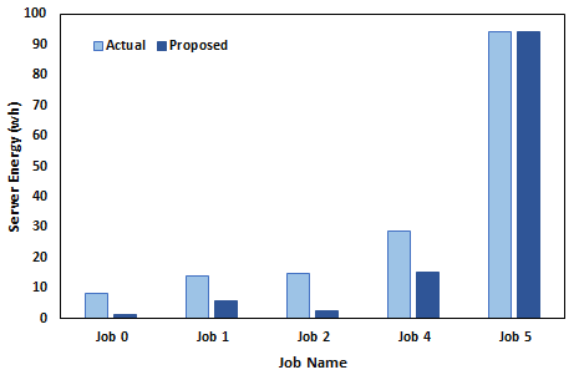


Figure 5.28. Overall Energy Conservation Statistics

conservation and excess energy expenditures within a given job accordingly. Thus, an overall reduction in the server energy expenditures can be achieved by a margin of 82.75%, 58.18%, 82.53%, 47.04% and 0.11% based on the proposed resource provisioning level across all tasks encompassed within Job 0, Job 1, Job 2, Job 3, Job 4 and Job 5 respectively. The difference in the achieved reduction in server energy expenditures across the studied jobs are attributed to job heterogeneity in terms of the straggler proportions, difference between the resource consumption level of stragglers and non-stragglers within a given job, task behavior consistency in terms of straggling behaviors, process efficiency and process capacity.

5.8 Summary

This chapter proposed a novel analytics driven resource optimisation framework to avail optimised level of resource provisioning to execute tasks with reduced resource wastages. The proposed framework includes three integral components such as the resource estimation module, straggler classification module and the resource optimisation module. Estimating the resource consumption levels of tasks encompassed within jobs before the execution, tasks within a given job are classified based on their resource requirements and execution trend. Further, the estimated resource levels are optimised based on their classified intensity and several runtime factors affecting task execution. The effectiveness of every integral module are evaluated both theoretically and through practical experiments, which proves the effectiveness of the proposed analytics methodology in estimating the resource requirements of tasks with reliable level of accuracy. The straggler classification module is efficient in classifying the energy-aware stragglers well before the start of the actual task execution, and also effectively identify the energy-aware stragglers during runtime. Furthermore, the resource optimisation module incorporates the descriptive knowledge of task execution and postulates a resource provisioning level for tasks within jobs, in such a way that the originally provisioned resource level are reduced to significantly reduce the proportions of resource idleness with minimal probability of task failures.

6 Conclusion

6.1 Overview

This section presents the major contributions of this thesis along with outlining the future research directions of this research study.

6.2 Major Contributions

The research objectives of this thesis are achieved through a combination of three core components such as the descriptive analytics, predictive analytics and prescriptive analytics. Extensive literature study conducted in Chapter 2 on energy efficient Cloud Computing included studying the concept of Cloud Computing, and focused on uncovering the hidden barriers of achieving energy efficient datacentre execution along with exploring the research gaps of existing state-of-the-art on energy efficient Cloud Computing, this has reflected the first objective of this thesis. The conducted literature review shaped this research across two core research dimensions: firstly, predicting the intensity of workload arrival in the next few hours at the datacentres benefits scaling up/down the active server resources based on the arrival traffic to reduce server idleness, secondly, predicting the behaviours of the arrived workloads at the datacentres helps to avail the most optimum level of resources to the arrived tasks to execute them with minimal proportions of resource idleness to achieve maximum utilisation of minimal number of active servers.

An extensive range of descriptive analytics has been presented in Chapter 3. By exploring a large-scale real Google Cloud trace logs, the trend of provision-to-consumption of resources within job execution has been presented on a day-wise analysis. This analysis is intended to highlight the proportion of resources spent on idle resource time during job execution and further to project the scope for proportional reduction in the server energy expenditures spent on such resource idleness. This analysis empirically suggested that CPU resource are more vulnerable for resource idleness witnessed at around 75% of idleness within a single day and its memory counter-part may display around 25% of resource idleness against their actual level of provisioned resources for job execution, this has reflected the second objective of this thesis. Furthermore, extensive descriptive analytics has been conducted on the characteristic behaviours of Cloud workloads and users with the motivation of exhibiting and exploiting their predictive properties for predicting their anticipated future behaviours, reflecting the third objective of this research. Firstly, the workload latency sensitivity has been deeply studied

along with exhibiting their energy related implications, which showed that Cloud workloads are mostly less latency sensitive and usually the most latency sensitive workload are the resource hungers and are more vulnerable for terminations. Secondly, the inherent periodicity among Cloud workload and user behaviours has been empirically uncovered which laid the foundation and presented important inferences for predictive analytics. This analysis has showed that Cloud workload and user behaviours are bound to various periodical effects, and such periodicity is usually existent among user usage patterns, workload arrival traffic etc. Furthermore, as discussed in chapter 3, most of the Cloud workloads incur multiple submissions in such a way that around 88% of job types account for nearly 50% of the arrived jobs, thus exhibit better repeating behaviour.

The fourth objective has been achieved across Chapter 4 and Chapter 5. Chapter 4 proposed a novel prediction framework, named InOt-RePCoN, to forecast the anticipated user behaviours at the datacentres in terms of their session usage, number of submissions and submission arrival trend. The outcome of this framework presents important inferences about the anticipated workload intensity at the datacentres which addressed the first research dimension of this thesis to benefit server scaling. The reliability of this proposed user behaviour forecasting framework has been empirically verified which showed that the proposed framework delivers the forecast with reliable level of accuracy better than the claimed accuracy percentage of the compared state-of-the-art in the same context. Furthermore, the dependability of the proposed framework has been tested by training real Google Cloud samples from different day-of-the-week and time-of-the-day comprising both peak-time and off-peak time workloads.

Chapter 5 proposed a novel framework to optimise the level of resources provisioned to task execution integrated with a resource estimation module and a straggler classification module, through a combination of descriptive analytics benefitting predictive analytics. The heterogeneity among tasks within a single job has been empirically exhibited in terms of their resource intensity, execution trend and straggling behaviours. This insisted that tasks within a single job may display different proportions of incurred idleness and consumption levels and may behave differently which insisted that tasks within a given job are highly heterogeneous and should be uniquely treated for resource optimisation. Exploiting the descriptive knowledge of the historical usage profile analysis, the anticipated resource consumption levels of the arrived tasks within jobs are estimated priori, and tasks within a given job are classified based on their resource intensity. Based on this estimation and classification, every individual task within a given job are uniquely treated to optimise their level of resource provision by

incorporating the effects of various runtime factors. The effectiveness of this proposed framework has been verified both theoretically and experimentally, which demonstrated that the proposed framework is effective in availing the most appropriate level of resources with reduced resource idleness. The resource estimation module and the straggler classification module has been proved to deliver better reliability and accuracy than the compared state-of-the-art of the same context.

6.3 Benefits and Overheads

The developed analytics model in this thesis presents a variety of scope for energy efficiency in Cloud datacentres. While the InOtRePCoN provides a prior knowledge about user behaviours in terms of their submission trend, the resource estimation and straggler analytics framework provides the prior knowledge about the anticipated behaviours of jobs at the datacentre. The benefits provided to the service providers from the proposed frameworks can be witnessed from two perspectives. Firstly, the former framework provides energy benefits pertinent to server management by providing a prior quantification and arrival trend of user jobs, which can be exploited to scale up and down the server resources in accordance with the estimation provided by the framework to conserve running server energy. The latter straggler framework helps the providers to achieve energy efficiency by providing a prior estimation of not only the resources required to process jobs but also the resource intensiveness of individual tasks within jobs. This information helps the providers to uniquely treat every task within a given job, which further helps to avoid both over-provision of less resource intensive tasks and also under-provision of the resource intensive tasks.

In practice at the real datacentre deployment, both the proposed frameworks should be integrated with the schedulers or the Cloud brokers so that the information extracted from the frameworks can be fed into the schedulers for achieving optimum placement of jobs with appropriate allocation of resources. The historical knowledge base required for performing the proposed analytics is limited to just a week, so that the storage requirements are fairly minimum. Though the proposed model incurs a few computation overheads and computation time, the benefits availed to energy efficiency cannot be undermined witnessing the higher percentage of resource wastages during the actual job execution. However, the proposed model may incur computation complexities without availing any notable energy benefits particularly when the incurred resource idleness are fairly low (Job 5 in Chapter 5 can be quoted as a representative of such a job kind). For such a job kind, the analytics for energy efficiency can

be avoided to leave a marginal proportion of resource wastage. But, such a job kind are very few in population in the analysed trace logs, thus from a wider perspective, the benefits availed to energy efficiency with the computation complexities of the proposed frameworks can still be lauded. Testing the efficiency of the proposed model in real datacentre deployments with an automated machine learning approach is considered to be beyond the scope of this research work, which is one of the future research plans.

6.4 Future Work

In the proposed InOt-RePCoN framework in Chapter 4, the effects of influential outliers in the prediction sample have been restrained to reduce the impacts of such outliers up on the forecast accuracy. Still, the submission trend forecast of the proposed framework for the samples containing no outliers is better than the forecast for samples containing significant proportions of outliers during both off-peak and peak times. For instance, the occurrence of peaks and valleys in the future submission intervals of prediction samples containing no outliers are forecasted more accurately than the samples influenced by the presence of outliers. In other words, the error margin for samples containing no outliers are much lower than the samples influenced by outliers. One of the future works of this research is to explore the possibility of enhancing the prediction accuracy of InOt-RePCoN, with the motivation of reducing the prediction error margin interval for outlier influenced samples.

The proposed approach of resource provision optimisation presented in Chapter 5 performs better for non-stragglers than energy-aware stragglers, in such a way that the resource estimation of non-stragglers presents better reliability. Further, the straggler classification framework includes a marginal proportion of false positive rate which may result in excess resource provision. Though marginal, reducing such false positive overheads helps to achieve better reduction in server energy expenditures. Investigating the possibility of enhancing the crispness of resource estimation of stragglers and reducing the false positive rates of the classification approach is one of the future research directions.

Though the proposed framework in Chapter 5 is effective in reducing resource idleness for majority of task proportion, a few tasks still suffer the probability of termination caused by resource level breaches. Furthermore, tasks terminations triggered by some runtime factors are inevitable. Whenever tasks suffer resource-related terminations, tasks are usually resubmitted and should be availed with better level of resources. Deciding this level of resource provision for resubmitted tasks is crucial such that the future termination probabilities should be reduced

and also the overall energy expenditure spent on terminated tasks should be maintained at the minimum possible level. To this end, integrating a cost-benefit analytics module to the proposed framework in Chapter 5 would be beneficial to decide the resource provisioning level for terminated tasks. This cost-benefit model should give special emphasis to the parameters such as task process revenue, RTT, energy and resources spent on the terminated execution instances, overall energy and resources spent on a given task so far etc. Developing this cost-benefit model is another future research direction of this research.

List of References

[1] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models," presented at the 7th International Symposium on Service Oriented System Engineering (SOSE), Redwood City, 2013.

[2] Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 2012–2017," Cisco, USA2013.

[3] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," presented at the Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010), Las Vegas, 2010.

[4] B. Tratz-Ryan, "Sustainability strategy in datacentres will become a key component of corporate excellence," ed: Gartner, 2013.

[5] L. Xu, G. Tan, X. Zhang, and J. Zhou, "Energy aware cloud application management in private cloud data center," in *Cloud and Service Computing (CSC), 2011 International Conference on*, 2011, pp. 274-279.

[6] C. Dupont, G. Giuliani, F. Hermenier, T. Schulze, and A. Somov, "An energy aware framework for virtual machine placement in cloud federated data centres," in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, 2012, pp. 1-10.

[7] A. García García, I. Blanquer Espert, and V. Hernández García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems,* vol. 31, pp. 1-11, 2014 2014.

[8] N. Bessis, S. Sotiriadis, F. Xhafa, F. Pop, and V. Cristea, "Meta-scheduling issues in interoperable HPCs, grids and clouds," *International Journal Web and Grid Services,* vol. 8, 2012 2012.

[9] S. Sotiriadis, N. Bessis, F. Xhafa, and N. Antonopoulos, "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud," presented at the 26th International Conference on Advanced Information Networking and Applications, Fukuoka, 2012.

[10] S. Sotiriadis, N. Bessis, and N. Antonpoulos, "Decentralized Meta-brokers for Inter-Cloud: Modeling brokering coordinators for interoperable resource management," presented at the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012), Sichuan, 2012.

[11] V. lunhao and L. Aili, "Research on the Application of Virtualization Technology in High Performance Computing," presented at the Symposium on Electrical & Electronics Engineering (EEESYM), 2012.

[12] L. Yan, "Development and Application of Desktop Virtualization Technology," presented at the International Conference on Communication Software and Networks (ICCSN), Xi'an, 2011.

[13] N. R. Challa, "Hardware based I/O Virtualization technologies for Hypervisors, Configurations and Advantages – A study," presented at the International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, 2012.

[14] H. Zhang, G. Jiang, and K. Yoshihira, "Intelligent Workload Factoring for A Hybrid Cloud Computing Model," presented at the World Conference on Services – I, Los Angeles CA, 2009.

[15] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards Understanding Heterogeneous Clouds at Scale: Google Trace Analysis," Intel Science and Technology Center for Cloud Computing, Pittsburgh,2012.
[16] A. K. Mishra, J. L. Hellerstein, W. Crine, and C. R. Das, "Towards Characterizing Cloud Backend Workloads: Insights form Google Compute Clusters," *ACM SIGMETRICS Performance Evaluation Review* vol. 37, pp. 34-41, March 2010 2010.
[17] Z. Liu and S. Cho, "Characterizing Machines and Workloads on a Google Cluster," presented at the 41st International Conference on Parallel Processing Workshops, Pittsburgh, PA, 2012.
[18] S. Di, D. Kondo, and W. Cirne, "Characterization and Comparison of Cloud versus Grid Workloads," presented at the International Conference on Cluster Computing, 2012
[19] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis Modelling and Simulation of Workload Patterns in a Large Scale Utility Cloud," *IEEE Transactions on Cloud Computing,* vol. 2, pp. 208 - 221, 02 April 2014 2014.
[20] P. Garraghan, P. Townend, and J. Xu, "An Analysis of the Server Characteristics and Resource Utilization in Google Cloud," presented at the International Conference on Cloud Engineering, Redwood City, CA, 2013.
[21] T. Wang, J. Wei, W. Zhang, H. Zhong, and T. Huang, "Workload-aware anomaly detection for web applications," *The Journal of Systems and Software,* vol. 89, pp. 19-32, March 2014 2014.
[22] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," *The Journal of Supercomputing,* vol. 65, pp. 445-468, 2013.
[23] C. Peoples, G. Parr, S. McClean, B. Scotney, P. Morrow, S. Chaudhari*, et al.*, "An Energy Aware Network Management Approach Using Server Profiling in 'Green' Clouds," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, 2012, pp. 17-24.
[24] D. Kusic, J. O. KephartJames, E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing,* vol. 12, pp. 1–15, March 2009 2009.
[25] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th annual international symposium on Computer architecture*, San Diego, 2007, pp. 13-23
[26] R. Jansen and P. R. Brenner, "Energy efficient virtual machine allocation in the cloud," in *Green Computing Conference and Workshops (IGCC), 2011 International*, 2011, pp. 1-8.
[27] P. Garraghan, I. S. Moreno, P. Townend, and J. Xu, "An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment," *IEEE Transactions on Emerging Topics in Computing,* vol. 2, pp. 166-180, 04 February 2014 2014.
[28] T.-T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, 2010, pp. 1-8.
[29] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen, "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, p. 22.
[30] Google. (2015). *Efficiency: How we do it*. Available: https://www.google.com/about/datacenters/efficiency/internal/

[31] 42U. (2016). *What is PUE / DCiE? How to Calculate, What to Measure*. Available: http://www.42u.com/measurement/pue-dcie.htm
[32] D. World. (2015). *Adder technology helps data Centre provider achieve highest environmental efficiency*. Available: https://digitalisationworld.com/article/46717/blogsall
[33] P. Garraghan, X. Ouyang, P. Townend, and J. Xu, "Timely Long Tail Identification through Agent Based Monitoring and Analytics," presented at the 18th International Symposium on Real-Time Distributed Computing, Auckland, 2015.
[34] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns*: Springer, 2014.
[35] S. Dick, O. Yazdanbaksh, X. Tang, T. Huynh, and J. Miller, "An empirical investigation of Web session workloads: Can self-similarity be explained by deterministic chaos?," *Information Processing and Management,* vol. 50, pp. 41–53, January 2014 2014.
[36] Y. S. Suna, Y.-F. Chena, and M. C. Chenb, "A Workload Analysis of Live Event Broadcast Service in Cloud," *Procedia Computer Science,* vol. 19, pp. 1028-1033, 2013 203.
[37] J. Yin, X. Lu, H. Chen, X. Zhao, and N. N. Xiong, "System resource utilization analysis and prediction for cloud based applications under bursty workloads," *Information Sciences,* vol. 279, pp. 338–357, 20 September 2014 2014.
[38] J. Tai, J. Zhang, J. Li, W. Meleis, and N. Mi, "ArA: Adaptive resource allocation for cloud computing environments under bursty workloads," presented at the IEEE 30th International Performance Computing and Communications Conference (IPCCC), Orlando, 2011.
[39] N. Tankovic, N. Bogunovic, T. G. Grbac, and M. Zagar, "Analyzing Incoming Worklaod in Cloud Business Services," presented at the 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM) 2015.
[40] S. Shen, V. V. Beek, and A. Iosup, "Statistical Characterization of Business Critical Workloads Hosted in Cloud Datacentrers," presented at the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015.
[41] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and Clustering of Worklaods in Google Cluster Trace based on Resource Usage," Cornell University7 January 2015 2015.
[42] S. Mahambre, P. Kulkarni, U. Bellur, G. Chafle, and D. Deshpande, "Workload Characterization for Capacity Planning and Performance Management in IaaS Cloud," presented at the International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, 2012.
[43] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," presented at the IEEE Network Operations and Management Symposium, Maui, HI, 2012.
[44] B. Ciciani, D. Didona, P. D. Sanzo, R. Palmieri, S. Peluso, F. Quaglia*, et al.*, "Automated Workload Characterization in Cloud-based Transactional Data Grids," presented at the 26th International Symposium on Parallel and Distributed Processing (IPDPSW), Shanghai, 2012.
[45] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826-831.
[46] I. S. Moreno and J. Xu, "Customer-Aware Resource Overallocation to Improve Energy Efficiency in Real-Time Cloud Computing Data Centres," presented at the International Conference on Service-Oriented Computing and Applications (SOCA), Irvine, CA, 2011.

[47] S. U. Sharma and Y. B. Gandole, "Virtualization Approach to Reduce Network Latency for Thin Client Performance Optimization in Cloud Computing Environment," presented at the International Conference on Computer Communication and Informatics, Coimbatore, 2014.
[48] B. Ciciani, D. Didona, P. D. Sanzo, R. Palmieri, S. Peluso, F. Quaglia*, et al.*, "Automated Worklaod Characterization in Cloud-based Transactional Data Grids," presented at the 26th InternationalParallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, 2012.
[49] P. Garraghan, P. Townend, and J. Xu, "An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment," presented at the 15th International Symposium on High-Assurance Systems Engineering, Miami Beach, 2014.
[50] T. Ropars, A. Guermouche, B. Uçar, E. Meneses, L. V. Kalé, and F. Cappello, "On the use of Cluster-based Partial Message Logging to Improve Fault Tolerance for MPI HPC Applications," presented at the 17th International Conference on Euro-Par 2011 Parallel Processing, Bordeaux, 2011.
[51] B. Schroeder and G. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *IEEE Transactions on Dependable and Secure Computing,* vol. 7, pp. 337 - 350, 6 February 2009 2009.
[52] T. Nguyen and W. Shi, "Improving resource efficiency in data centers using reputation-based resource selection," presented at the International Conference on Green Computing, Chicago, 2010.
[53] Jorge-Arnulfo, Quiané-Ruiz, C. Pinkel, J. Schad, and J. Dittrich, "RAFTing MapReduce: Fast Recovery on the Raftc," presented at the 27th International Conference on Data Engineering, Hannover, 2011.
[54] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "BlueGene/L Failure Analysis and Prediction Models," presented at the International Conference on Dependable Systems and Networks (DSN'06), Philadelphia, 2006.
[55] H. Li, D. Groep, L. Wolters, and J. Templon, "Job Failure Analysis and Its Implications in a Large-scale Production Grid," presented at the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06), Amsterdam, 2006.
[56] I. S. Moreno, R. Yang, J. Xu, and T. Wo, "Improved Energy-Efficiency in Cloud Datacentres with Interference-Aware Virtual Machine Placement," presented at the Eleventh International Symposium on Autonomous Decentralized Systems (ISADS), Mexico City, 2013.
[57] R. Yang, I. S. Moreno, J. Xu, and T. Wo, "An Analysis of Performance Interference Effects on Energy-Efficiency of Virtualized Cloud Environments," presented at the 5th International Conference on Cloud Computing Technology and Science, Bristol, 2013.
[58] F. Chen, J.-G. Schneider, Y. Yang, J. Grundy, and Q. He, "An Energy Consumption Model and Analysis Tool for Cloud Computing Environments," presented at the First International Workshop on Green and Sustainable Software (GREENS), Zurich, 2012.
[59] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," *Proceedings of the IEEE,* vol. 99, pp. 149 - 167, January 2011 2011.
[60] T. Lin, T. Alpcan, and K. Hinton, "Game-Theoretic Analysis of Energy Efficiency and Performance for Cloud Computing in Communication Networks," *IEEE Systems Journal,* vol. 11, pp. 649 - 660, June 2017 2017.
[61] S. Shaoling, Z. Jing, C. Moliang, R. Hui, C. Yi, and F. Xiaodong, "Network Energy Consumption Analysis and Dormancy Mechanism Based on Ant Colony Algorithm in Cloud Computing Environment for IOT Service and Real-time Embedded Industrial

Control System," presented at the 27th ChineseControl and Decision Conference (CCDC), Qingdao, 2015.
[62] Y. Jin, Y. Wen, Q. Chen, and Z. Zhu, "An Empirical Investigation of the Impact of Server Virtualization on Energy Efficiency for Green Data Center," *The Computer Journal,* vol. 56, pp. 977-990, 20 February 2013 2013.
[63] I. Takouna, W. Dawoud, and C. Meinel, "Energy efficient scheduling of HPC-jobs on virtualize clusters using host and VM dynamic configuration," *ACM SIGOPS Operating Systems Review,* vol. 46, pp. 19-27, 2012.
[64] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, and Q. B. Wang, "GreenCloud: a new architecture for green data center," in *sixth International Conference on Autonomic Computing*, Barcelona, Spain, 2009.
[65] Q. Chen, P. Grosso, K. v. d. Veldt, C. d. Laat, R. Hofman, and H. Bal, "Profiling energy consumption of VMs for green cloud computing," in *Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC*, 2011, pp. 768-775.
[66] I. Takouna, W. Dawoud, and C. Meinel, "Energy Efficient Scheduling of HPC-jobs on Virtualized Clusters Using Host and VM Dynamic Configuration," presented at the SIGOPS Operating Systems Review, 2011.
[67] Z. Zhang and S. Fu, "Characterizing power and energy usage in cloud computing systems," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 146-153.
[68] P. Raycrofta, R. Jansena, M. Jarus, and P. R. Brennera, "Performance bounded energy efficient virtual machine allocation in the global cloud," *Sustainable Computing: Informatics and Systems,* vol. 4, pp. 1 - 9, March 2014 2013.
[69] N. Bessis, S. Sotiriadis, V. Cristea, and F. Pop, "Modelling requirements for enabling meta-scheduling in inter-clouds and inter-enterprises," in *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, 2011, pp. 149-156.
[70] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *4th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012, pp. 82-89.
[71] Y. Kessaci, N. Melab, and E.-G. Talbi, "A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager," *Future Generation Computer Systems,* 2013.
[72] N. Kim, J. Cho, and E. Seo, "Energy-based accounting and scheduling of virtual machines in a cloud system," in *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, 2011, pp. 176-181.
[73] R. Jeyarani, N. Nagaveni, and R. Vasanth Ram, "Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence," *Future Generation Computer Systems,* vol. 28, pp. 811-821, 2012.
[74] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems,* vol. 28, pp. 755-768, 2012.
[75] P. Graubner, M. Schmidt, and B. Freisleben, "Energy-efficient virtual machine consolidation," *IT Professional,* vol. 15, pp. 0028-34, 2013.
[76] S. S. Manvi and G. Krishna Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of Network and Computer Applications,* vol. 41, pp. 424-440, May 2014 2013.
[77] A. Corradi, M. Fanelli, and L. Foschini, "VM consolidation: A real case based on OpenStack Cloud," *Future Generation Computer Systems,* vol. 32, pp. 118 - 127, 2014 2014.

[78] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart, and J. Torres, "Energy-aware scheduling in virtualized datacenters," in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*, 2010, pp. 58-67.

[79] K. Ye, D. Huang, X. Jiang, H. Chen, and S. Wu, "Virtual machine based energy-efficient data center architecture for cloud computing: a performance perspective," in *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, 2010, pp. 171-178.

[80] C.-T. Yang, K.-C. Wang, H.-Y. Cheng, C.-T. Kuo, and W. C.-C. Chu, "Green power management with dynamic resource allocation for cloud virtual machines," in *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, 2011, pp. 726-733.

[81] N. J. Kansal and I. Chana, "Cloud Load Balancing Techniques: A Step Towards Green Computing," *International Journal of Computer Science Issues (IJCSI),* vol. 9, 2012.

[82] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole, "Energy-efficient application-aware online provisioning for virtualized clouds and data centers," in *Green Computing Conference, 2010 International*, 2010, pp. 31-45.

[83] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach*, et al.*, "Live migration of virtual machines," in *2nd conference on Symposium on Networked Systems Design & Implementation*, 2005, pp. 273-286.

[84] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, Anaheim, 2005, p. 25.

[85] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proceedings of the 18th ACM international symposium on High performance distributed computing*, Garching, 2009, pp. 101 - 110.

[86] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual machines," *Future Generation Computer Systems,* vol. 28, pp. 469 - 477, 7 May 2011 2012.

[87] P. Raycroft, R. Jansen, M. Jarus, and P. R. Brenner, "Performance bounded energy efficient virtual machine allocation in the global cloud," *Sustainable Computing: Informatics and Systems,* 2013.

[88] M. M. Taheri and K. Zamanifar, "2-phase optimization method for energy aware scheduling of virtual machines in cloud data centers," in *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, 2011, pp. 525-530.

[89] N. Tziritas, S. U. Khan, C.-Z. Xu, T. Loukopoulos, and S. Lalis, "On minimizing the resource consumption of cloud applications using process migrations," *Journal of Parallel and Distributed Computing,* vol. 73, pp. 1690-1704, 2013.

[90] A. Beloglazov and R. Buyya, "Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers," presented at the MGC, Bangalore, 2010.

[91] C. Lefurgy, X Wang, and M. Ware, "Server-level power control," presented at the Fourth International Conference on Autonomic Computing (ICAC), 2007.

[92] VMware, "VMware distributed power management concepts and use," VMware Inc2010.

[93] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," presented at the Proceedings of the 1st ACM symposium on Cloud computing, 2010.

[94] N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting," presented at the 4th International Conference on Cloud Computing, Washington, 2011.
[95] A. A. Bankole and S. A. Ajila, "Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment," presented at the Seventh International Symposium on Service-Oriented System Engineering, Redwood City, 2013.
[96] J. L. Berral, R. Gavalda, and J. Torres, "Adaptive scheduling on power-aware managed data-centers using machine learning," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, 2011, pp. 66-73.
[97] D. Bacigalupo, J. Van Hemert, A. Usmani, D. Dillenberger, G. Wills, and S. Jarvis, "Resource management of enterprise cloud systems using layered queuing and historical performance models," in *International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010, pp. 1-8.
[98] R. Miana, P. Martin, and J. L. Vazquez-Poletti, "Provisioning data analytic workloads in a cloud," *Future Generation Computer Systems,* pp. 1452 - 1458, 82 - 84 2013.
[99] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments," presented at the Proceedings of the 9th international conference on Autonomic computing, San Jose, CA, 2012.
[100] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems,* vol. 28, pp. 155-162, Januray 2012 2012.
[101] Y. Jiang, C.-s. Perng, T. Li, and R. Chang, "ASAP: A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning," presented at the 11th IEEE International Conference on Data Mining, Vancouver,BC, 2011.
[102] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload Estimation for Improving Resource Management Decisions in the Cloud," presented at the Twelfth International Symposium on Autonomous Decentralized Systems, Taichung, 2015.
[103] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload Predicting-Based Automatic Scaling in Service Clouds," presented at the Sixth International Conference on Cloud Computing, Santa Clara, CA, 2013.
[104] E. Caron, F. Desprez, and A. Muresan, "Forecasting for Grid and Cloud Computing On-Demand Resources Based on Pattern Matching," presented at the 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010.
[105] T. Li, J. Wang, W. Li, T. Xu, and Q. Qi, "Load Prediction-based Automatic Scaling Cloud Computing," presented at the International Conference on Networking and Network Applications, 2016.
[106] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload Prediction for Cloud Computing Elasticity Mechanism," presented at the IEEE International Conference on Cloud Computing and Big Data Analysis, 2016.
[107] C.-F. Wang, H. Wen-Yi, and C.-S. Yang, "A Prediction Based Energy Conserving Resources Allocation Scheme for Cloud Computing," presented at the IEEE International Conference on Granular Computing (GrC), Noboribetsu, 2014.
[108] A. Mityakov, D. L. Petrov, M. Kostenko, and D. Butusov, "User demand prediction algorithm used in mobile device driver for cloud storage," presented at the 11th International Conference on ITS Telecommunications, St. Petersburg, 2011.
[109] S. Mallick, G. Hains, and C. S. Deme, "A Resource Prediction Model for Virtualization Servers," presented at the International Conference on High Performance Computing and Simulation (HPCS), Madrid, 2012.

[110] S. Mallick, G. Hains, and C. S. Deme, "A resource prediction model for virtualization servers," presented at the International Conference on High Performance Computing & Simulation (HPCS), Madrid, 2012.
[111] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian Workload Characterization for QoS Prediction in the Cloud," presented at the 4th International Conference on Cloud Computing, Washington, 2011.
[112] S. Di, D. Kondo, and W. Cirne, "Host Load Prediction in a Google Compute Cloud with a Bayesian Model," presented at the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Salt Lake City, UT, 2012
[113] S. Di, D. Kondo, and W. Cirne, "Google hostload prediction based on Bayesian model with optimized feature combination," *Journal of Parallel and Distributed Computing,* vol. 74, pp. 1820-1832, 1 January 2014 2014.
[114] J. Panneerselvam, L. Liu, N. Antonopoulos, and B. Yuan, "Workload Analysis for the Scope of User Demand Prediction Model Evaluations in Cloud Environments," presented at the 7th International Conference on Utility and Cloud Computing (UCC), Lodon, 2014.
[115] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multitier applications in the cloud", Future Generation Computer System," *Future Generation Computer Systems,* vol. 27, pp. 871-879, June 2011 2011.
[116] N. K. Gondhi and P. Kailu, "Prediction Based Energy Efficient Virtual Machine Consolidation in Cloud Computing," presented at the Second International Conference on Advances in Computing and Communication Engineering, 2015.
[117] Y. Shen, "Virtual resource scheduling prediction based on a support vector machine in cloud computing," presented at the 8th International Symposium on Computational Intelligence and Design, 2015.
[118] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao*, et al.*, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," presented at the Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, 2008.
[119] W. Fang, Z. Lu, J. Wu, and Z. Cao, "RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center," presented at the IEEE Ninth International Conference on Services Computing, Honolulu, HI, 2012.
[120] N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud using Predictive Models forWorkload Forecasting," presented at the IEEE 4th International Conference on Cloud Computing, Washington, DC, 2011.
[121] X. Ouyang, P. Garraghan, D. Mckee, P. Townend, and J. Xu, "Straggler Detection in Parallel Computing Systems through Dynamic Threshold Calculation," presented at the 30th International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, 2016.
[122] J. Rosen and B. Zhao, "Fine-Grained Micro-Tasks for MapReduce Skew-Handling," 2012.
[123] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "SAMR: A Self-adaptive MapReduce Scheduling Algorithm In Heterogeneous Environment," presented at the 10th IEEE International Conference on Computer and Infomation Technology (CIT 2010), Bradford, 2010.
[124] Q. Chen, C. Liu, and Z. Xiao, "Improving MapReduce Performance Using Smart Speculative Execution Strategy," *IEEE Transactions on Computers,* vol. 63, pp. 954-967, 24 January 2013 2014.

[125] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," presented at the 8th USENIX conference on Operating systems design and implementation, San Diego, 2008.
[126] N. J. Yadwadkar, G. Ananthanarayanan, and R. Katz, "Wrangler: Predictable and Faster Jobs using Fewer Resources," presented at the Proceedings of the ACM Symposium on Cloud Computing, Seattle, 2014.
[127] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: attack of the clones," presented at the Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation Pages, Lombard, 2013.
[128] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha*, et al.*, "Reining in the outliers in map-reduce clusters using Mantri," presented at the Proceedings of the 9th USENIX conference on Operating systems design and implementation Pages, Vancouver, 2010.
[129] S.-W. Huang, T.-C. Huang, S.-R. Lyu, C.-K. Shieh, and Y.-S. Chou, "Improving Speculative Execution Performance with Coworker for Cloud Computing," presented at the IEEE 17th International Conference on Parallel and Distributed Systems, Taiwan, 2011.
[130] N. J. Yadwadkar and W. Choi, "Proactive Straggler Avoidance using Machine Learning," University of Berkeley, University of Berkeley2012.
[131] X. Ouyang, P. Garraghan, C. Wang, P. Townend, and J. Xu, "An Approach for Modeling and Ranking Node-Level Stragglers in Cloud Datacenters," presented at the IEEE International Conference on Services Computing, 2016.
[132] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding Long Tails in the Cloud," in *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, Lombard, 2013, pp. 329-342.
[133] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time Cloud services," *Concurrency and Computation: Practice and Experience,* vol. 23, pp. 1491-1505, 2011.
[134] D. G.-d. Lago, E. R. Madeira, and L. F. Bittencourt, "Power-aware virtual machine scheduling on clouds using active cooling control and DVFS," in *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, 2011, p. 2.
[135] "Google Cluster Data V2," Google, Ed., 2 ed, 2011.
[136] C. Reiss, J. Wilkes, and J. Hellerstein, "Google Cluster-Usage Traces: Format + Schema," Google Inc6 May 2013 2013.
[137] Z. Wan, "Cloud Computing infrastructure for latency sensitive applications," presented at the 12th International Conference on Communication Technology, Nanjing, 2010.
[138] M. S. Bali and S. Khurana, "Effect of latency on network and end user domains in Cloud Computing," presented at the International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), Chennai, 2013.
[139] Z. Wan, "Sub-millisecond level latency sensitive Cloud Computing infrastructure," presented at the International Congress on Ultra Modern Telecommunications and Control Systems, Moscow, 2010.
[140] Z. Wan, P. Wang, J. Liu, and W. Tang, "Power-Aware Cloud Computing Infrastructure for Latency-Sensitive Internet-of-Things Services," presented at the UKSim 15th International Conference on Computer Modelling and Simulation, Cambridge, 2013.
[141] J. Whitney and P. Delforge, "Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers," NRDCAugust 2014 2014.

[142] Z. Lu, S. Takashige, Y. Sugita, T. Morimura, and Y. Kud, "An Analysis and Comparison of Cloud Data Center Energy Efficient Resource Management Technology," *International Journal of Services Computing,* vol. 2, pp. 32 - 51, December 2014 2014.
[143] M. Mazzucco and D. Dyachuk, "Optimizing cloud providers revenues via energy efficient server allocation," *Sustainable Computing: Informatics and Systems,* vol. 2, pp. 1-12, 2012.
[144] SPEC, "SPECpower_ssj2008 Results," SPEC.
[145] Vertatique. (2015). *Average Power Use Per Server*. Available: http://www.vertatique.com/average-power-use-server
[146] M. Golden, "Data Centrers can slash CO2 emissions 88% or more," Stanford Precout Institure for Energy18 July 2013 2013.
[147] EIA. (2016). *Independent Statistics and Analysis*. Available: http://www.eia.gov
[148] Infrarati. (2014). *Data Center CO2 Emissions*. Available: https://infrarati.wordpress.com/2014/03/25/data-center-co2-emissions/
[149] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers & Electrical Engineering,* 2013.
[150] F. Satoh, H. Yanagisawa, H. Takahashi, and T. Kushida, "Total Energy Management System for Cloud Computing," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, 2013, pp. 233-240.
[151] *GreenCloud 'Simulating Energy-Efficient Clouds'*. Available: https://greencloud.gforge.uni.lu/publications.html
[152] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing,* vol. 62, pp. 1263–1283, December 2012 2012.