An Investigation into the Impacts of Task-level Behavioural Heterogeneity upon Energy Efficiency in Cloud Datacentres

John Panneerselvam[1], Lu Liu[1], Yao Lu[2] and Nick Antonopoulos[1]

[1]Department of Engineering and Technology, University of Derby, Derby, United Kingdom, DE22 1GB
[2]School of Computer Science and Telecommunication Engineering Jiangsu University, Jiangsu, China
Email: {j.panneerselvam; l.liu and n.antonopoulos}@derby.ac.uk
Corresponding Author: Lu Liu

Abstract

Cloud datacentre resources and the arriving jobs are addressed to be exhibiting increased level of heterogeneity. A single Cloud job may encompass one to several number of tasks, such tasks usually exhibit increased level of behavioural heterogeneity though they belong to the same job. Such behavioural heterogeneity are usually evident among the level of resource consumption, resource intensiveness, task duration etc. These task behavioural heterogeneity within jobs impose various complications in achieving an effective energy efficient management of the Cloud jobs whilst processing them in the server resources. To this end, this paper investigates the impacts of the task level behavioural heterogeneity upon energy efficiency whilst the tasks within given jobs are executed in Cloud datacentres. Real-life Cloud trace logs have been investigated to exhibit the impacts of task heterogeneity from three different perspectives including the task execution trend and t ask termination pattern, the presence of few proportions of resource intensive and long running tasks within jobs. Furthermore, the energy implications of such straggling tasks within jobs have been empirically exhibited. Analysis conducted in this study demonstrates that Cloud jobs are extremely heterogeneous and tasks behave distinctly under different execution instances, and the presence of energy-aware long tail stragglers within jobs can significantly incur extravagant level of energy expenditures.

Keywords: Energy-aware stragglers, Long-tails, Resource Idleness, Task Heterogeneity

1. Introduction

In the Cloud Computing service model, user submitted jobs are scheduled and processed in the Virtual Machines (VM) or Linux Containers (LXC) deployed on the physical server resources. Cloud datacentres are composed of massive number of heterogeneous servers providing supplements for accommodating (VMs) to process the user requests. Users submit their request in the form of jobs [1] at the datacentres, a single job may encompass one to several number of tasks. The scheduler [2] in the datacentre schedules and allocates the tasks belonging to a single job across the available server nodes based on the computational requirements of every individual tasks, unlike a typical Map Reduce platform [3] divides the job into multiple tasks for executing the tasks in an evenly distributed subsets. In other words, tasks belonging to a single job show increased level of heterogeneity [4, 5] in terms of their resource requirements, resource consumption and task duration etc., but a Map Reduce platform tries to achieve even execution profiles across the distributed tasks such as similar duration, resource consumption etc. This is not possible in heterogeneous Cloud jobs due to the task diversity and the variations found among the server node characteristics such as operating system, resource capacity such as CPU cores and memory, architecture and aging etc. 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 the task being executed in the corresponding nodes. This phenomena usually causes the tasks to run longer [6] than their normal duration, tasks exhibiting a runtime duration increasingly proportional to the average duration of the other co-located tasks are classified as stragglers. Since the 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 the 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 the tasks themselves. Server nodes exhibiting poor performance and declining process capability [7] can cause node-level stragglers, naturally tasks scheduled onto such node-level stragglers suffer performance constraints and face possible terminations and prolonged execution duration than anticipated. The existence of the correlation between the process capacity of the nodes and the task progress are beneficial in identifying the node level stragglers, where the task progress is determined by the 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 the tasks being executed in that particular node. Task-level stragglers naturally exhibit a varied and abnormal behaviour to that of the other co-located tasks within the same job. Whilst the node-level stragglers can be mitigated by avoiding poor nodes for job and task allocation, the 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, in such a way that a common task-level straggler can behave distinctively under different server nodes. Thus a generalised view of all the tasks within a job can mislead to derive unreliable inferences in spite of the task heterogeneity.

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 of the physical resources based on the requirements specified by the users. Tasks within a single job can be allocated onto different servers and can be provisioned with different resource levels particularly in the case of parallel tasks where the encompassing tasks within a single job are totally independent. This predefined level of resource provision is usually the maximum level of allowed resources for the LXCs or VMs. It is commonly being argued that the level of resource provisioning to process jobs usually far exceed [8] their actual requirements, which lead to undesirable energy expenditures by incurring larger proportions of idle resources. Furthermore, it is increasingly common in a datacentre that jobs and tasks face various types of terminations [9, 10]. Such terminations are usually caused by resource level breaches, resource unavailability, server outages, and machine failures etc. In Cloud datacentres, job terminations usually cause resubmissions of the entire job and task terminations within jobs cause resubmission of only the terminated tasks, both of which incur additional energy and resource expenditures [9]. This phenomena also affects the completion of the entire job, whereby the terminated tasks are reprocessed until they are executed successfully. Since resource level breaches are one of the major causes for task terminations, tasks characterising higher resource requirements are vulnerable for terminations potentially causing task-level stragglers. Existing works of straggler identification addresses task duration as the only major attribute to label tasks as stragglers, re-processing a few tasks within jobs resulting from tasks terminations have been ignored in straggler analysis [3, 11, 12] which leave the impacts of considerable proportions of tasks-level stragglers unnoticed. Whilst stragglers are approached to mitigate long tails for shorten durations, this paper addresses the phenomenon of task behavioural heterogeneity in terms of their resource intensiveness as one of the major causes for extravagant provisioning of resources. This behavioural heterogeneity has been approached from an energy efficiency perspective, in such a way that tasks naturally running longer and requiring more resources incur more energy expenditures than majority of the remaining co-located tasks within the same job.

To this end, this paper addresses tasks exhibiting a behavioural heterogeneity in terms of requiring more resource expenditures within a single job as energy-aware stragglers and further analyses the impacts of both long tails and energy-aware stragglers upon the overall energy efficiency. Important contributions of this paper include an empirical analysis of the execution behaviours of tasks within jobs. Task behaviours within their respective jobs are analysed from three different perspectives. Firstly, the execution trend of tasks within jobs have been analysed to exhibit the heterogeneity among task execution within respective jobs, this includes uncovering the relationship between CPU usage rate and task duration, along with studying the trend of task terminations within jobs. Secondly, the proportional presence of energy-aware stragglers and long tails within jobs have been empirically exposed. Finally, the actual amount 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 energy related implications of task-level stragglers, along with a comparative analysis of the execution characteristics of energy-aware stragglers, long tails and non-stragglers,.

The remainder of this paper is organised as follows: Section 2 presents the background of the analysis presented in this paper and Section 3 studies the task behavioural heterogeneity in terms of their execution trend and termination pattern. Section 4 defines the characteristics of energy-aware stragglers and long tail stragglers along with presenting the empirical analysis of the same based on the studied jobs. Section 5 is covered with the energy implications of the task behavioural heterogeneity of tasks within jobs. Section 6 reviews the related works and Section 7 concludes this paper along with outlining our future research directions.

2 Background

2.1 Task Classification

Cloud jobs usually encompass a diverse group of tasks in respective to their execution behaviours, resource intensiveness, energy consumption levels, running duration etc. This paper postulates four different task groups within every jobs such as energy-aware stragglers, long tail stragglers, tasks characterising both energy-aware and long tail stragglers and non-stragglers. For ease of reading, we denote tasks characterising both energy-aware

Table 1: Job Profile Representation

Job Name	Encompassed Tasks
Job 1	50
Job 2	100
Job 3	200
Job 4	488
Job 5	1050

stragglers and long tails as task-level stragglers in this paper. The characteristics, execution behaviours and energy implications of such task groups are detailed later in this paper. This prior classification of task groups within a single job helps to uniquely treat each tasks group with optimum level of resource allocation based on their anticipated resource consumption levels and characteristic task behaviours in the server resources, so as to achieve possible level of resource conservation among the tasks encompassed within their respective jobs.

2.2 Analysis Sample

The investigation presented in this paper has been conducted based on the Cloud trace logs [13] released by Google, featuring more than 650000 jobs submitted by the 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 and the encompassed tasks at the Cloud datacentre. This analysis is intended to extract information pertinent to energy consumption of the server resources resulting from the behavioural heterogeneity among the tasks within jobs. Analysis of five different jobs representing different characteristics and behaviours have been displayed in this paper to present the extracted inferences about their behavioural heterogeneity upon energy efficiency.

2.3 Job Profiles

The execution profile of the 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 $c_{(c,m)}$ and job scheduling index J_{sh}, as shown in equation 1.

$$E_j = \{t_{sj}, J_i, J_n, n_t, c_{(c,m)}, J_{sh}\} \tag{1}$$

Since every task within jobs are processed individually, execution profiles of the 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 the task belongs, resource levels $c_{i(c,m)}$ of the i^{th} task within the job J_i and task priority T_p, as shown in equation 2.

$$E_t = \{t_{st}, T_i, J_i, c_{(c,m)i}, T_p\} \tag{2}$$

In terms of the resource requirements, job priority and the number of tasks encompassed within jobs are explicit, and the task length t_j can be calculated for the historical execution instances as the difference between the task completion time t_f and the time of task scheduling t_{sh}, as shown in equation 3.

$$l_j = t_f - t_{sh} \tag{3}$$

It is important to extract the competed execution profile of tasks, since profiles of terminated tasks cannot identically reflect the resource levels of the tasks. The trace logs presents a measurement period of 300 seconds while reading the execution parameters. Thus the total amount of resources consumed by a single task and its execution duration is given by the summation of all the measurement samples of the corresponding task, as shown equation 4 and equation 5.

$$R_T = \sum_{i=1}^{n} r_i \tag{4}$$

$$L_T = \sum_{i=1}^{n} l_i \tag{5}$$

where, R_T and L_T are the total amount 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*. It is obvious that 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. This paper presents the resource measurements of CPU usage levels and memory consumption levels within their respective execution. The CPU usage levels are measured in terms of CPU core-counts, the core-counts witnessed during any time of a given task execution is averaged and multiplied with the corresponding duration to obtain the total CPU consumption for that corresponding task. However, memory consumption is explicit as a static measurement in bytes for a given task execution in the analysed trace logs. The tasks are processed on isolated containers in the

analysed datacentres and the resource allocation and utilisation measurements are achieved within the respective containers processing the prospective tasks.

In spite of exposing the task heterogeneity, jobs encompassing different number of tasks have been presented in this paper as representatives of different job behaviours and task numbers, as shown in Table 1. Furthermore, the chosen jobs are increasingly heterogeneous in terms of the distribution of task length, resource consumption trend, termination pattern and the composition of energy-ware stragglers and long-tails. Detailed discussion of such heterogeneity are presented in Section 5. In addition, the analysed jobs are parallelised during execution, which means that the jobs only include parallel tasks and all the tasks within a given job are started at the same time. Thus, there is no linear dependencies among the tasks within a given job.

3. Task Execution Trend

3.1 Termination Pattern

Though it is desirable to process the tasks in one single execution instance, terminations are inevitable during the actual execution causing resubmissions. Whilst addressing over-estimation of the resource levels causing resource idleness, under-estimation of the resource levels leads to terminations whenever the execution exceeds the allowed or provisioned level of resources. Since resource level breaches are one of the important causes of task termination, CPU usage rate in terms of core-counts per second during any time of execution is crucial in determining the 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. The common termination events witnessed in the analysed datacentre include kill, fail, evict and lost events. Kill is a user triggered termination of the executing tasks, usually it does not involve the intervention of the service providers, whereas fail is a natural event which occurs in the datacentre due to several runtime causes such as resource scarcity and resource level breaches. Evict is a provider triggered event, where the execution of the running instances are paused temporarily to accommodate jobs with more priority, and later resumed when resources become available. Lost is an unpleasant event where the running tasks are completely lost due to service outages, machine failure, VM crashes etc.

Fig. 1 presents the proportions of task terminations within their respective jobs. It can be observed that the task termination proportions are totally unique for jobs and there is no linear dependency between the termination proportions and the total number of encompassed tasks within a given job. For instance, Job 2 encompassing 100 tasks characterise an increased proportions of task terminations at 95% and Job 5 encompassing 1050 tasks only characterise 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 task terminations resulting in the resubmission of only the terminated tasks. In this regard of increasing task failure, Job 2 is an entire job failure. Whilst addressing the resource level breaches as the primary causes for task terminations, runtime execution factors such as CPU usage rate, task duration and the consumed level of memory resources can also be postulated as potential causes exerting terminations. The termination pattern of randomly chosen tasks within Job 5 has been displayed in Fig. 2, 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 Fig. 2, 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 the job, or when

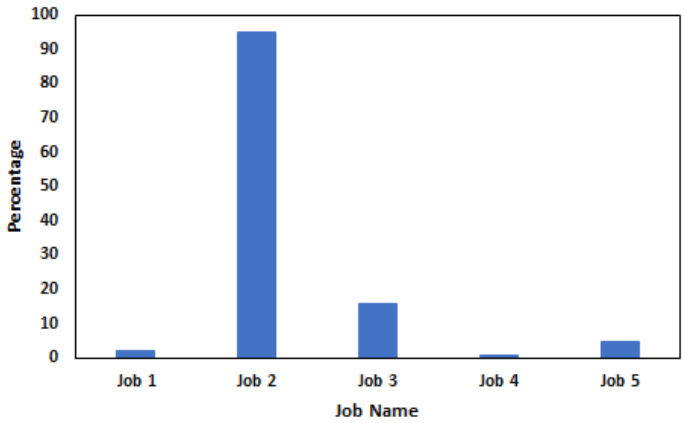

Fig 1. Task Termination Proportions

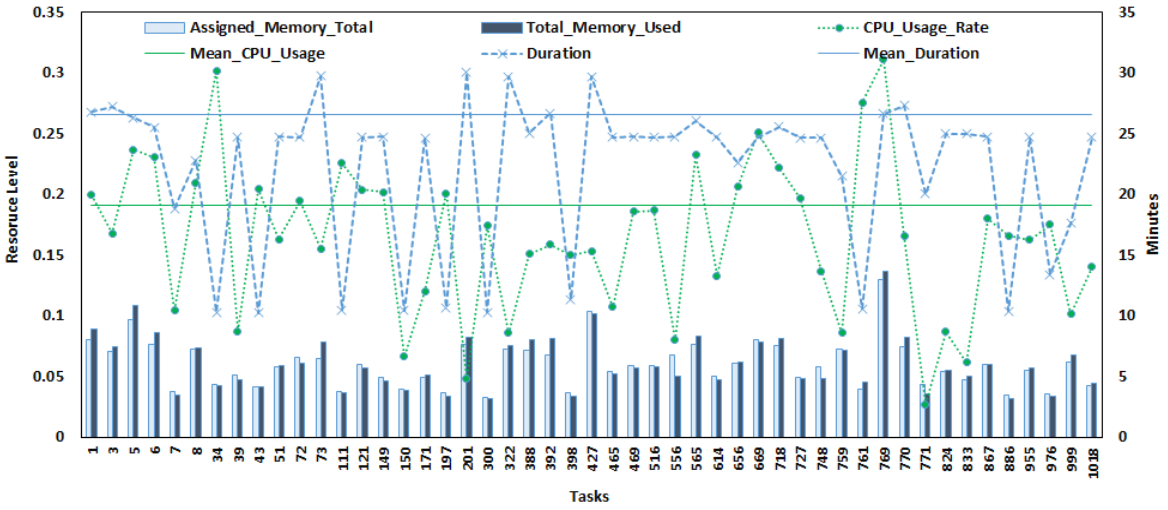

Fig. 2 Task Termination Pattern

the assigned level of memory resources are breached or when the CPU usage rate of a given task exceeds the mean CPU usage rate of the job. Mean CPU usage rate is the average CPU usage rate other than the highest peak of CPU utilisation level during a task execution session. 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.

3.2 Execution Trade-off

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 the 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 the task execution, memory usage usually remains fairly stable. The existence of the relationship [14] between the server node statistics such as CPU/memory resources and the task duration has been revealed to be non-trivial. The trade-off between task duration and CPU usage rate is crucial in determining the task execution efficiency. For instance, shortening the task duration may demand higher CPU usage rate and vice versa, but tasks with extravagant CPU usage rate are addressed 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.

Fig. 3 illustrates the trade-off between the CPU usage rate and the task duration for a selected non-terminated tasks within Job 1. 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 remained 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 the termination probabilities, such a trend of task execution satisfying the usage rate and duration trade-off can be regarded as a healthily execution. The term ‘healthily’ here refers to the satisfactory level of a task execution profile satisfying the usage rate-duration trade-off, where usually the usage rate and duration are inversely proportional to each other for a completed task profile. 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 6, u_i is the usage rate and d_i is the duration of the entire job.

$$opt(u, d) = \underbrace{mean}_{\forall t_i \in J} \{u_i, d_i\} \tag{6}$$

It is not a practical reality for 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. On a coarse grain, for a decreasing CPU usage rate the duration of the task usually increases and vice versa, however the proportional relationship in this trade-off is not obvious. Furthermore, there could be a few exceptions where the tasks are still completed without satisfying this trade-off between usage rate and duration. Fig. 4 illustrates a few examples of tasks within Job 1 where the usage-rate and duration trade-off is not satisfied,

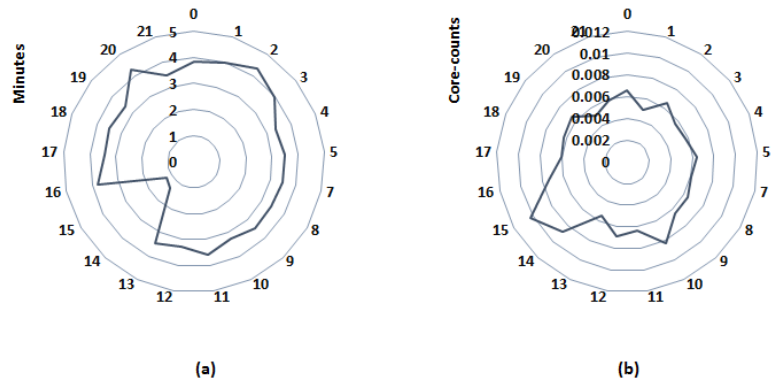

Fig. 3 Task Execution Pattern within Job 1 Satisfying Usage Rate Duration Trade-off (a) Task Duration (b) CPU usage rate

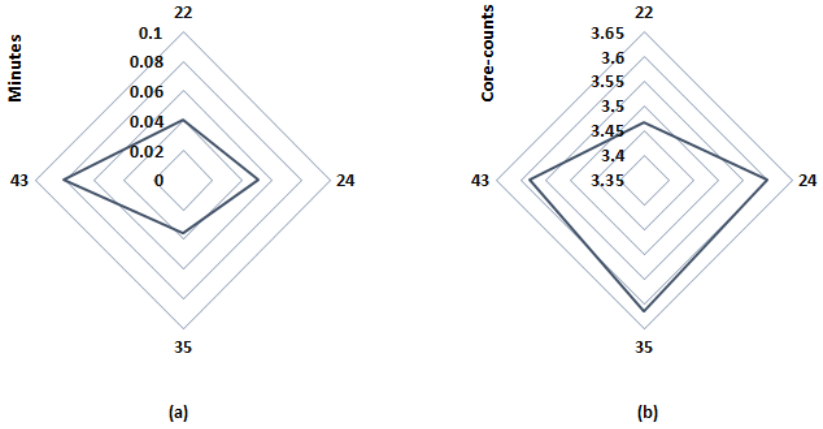

Fig. 4. Task Execution Pattern within Job 1 not Satisfying Usage Rate Duration Trade-off (a) Task Duration (b) CPU usage rate

however such tasks are terminated during execution. Here, all the tasks have consumed at least 0.03 core-counts but still characterise 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 task execution and can be ignored as they do not incur any notable level of excess energy expenditures, the latter are resource hungers and 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 the 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 duration trade-off. Jobs encompassing combinations of tasks satisfying and dissatisfying the usage rate and duration trade-off can be 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 to the actual execution.

4 Task-level Stragglers

4.1 Energy-aware Stragglers

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 not satisfying the usage rate and duration trade-off by characterising a combination of higher CPU 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 7.

$$S_t[i] = (U_C[i] > \mu) \cap (L_T[i] > \omega) \quad (7)$$

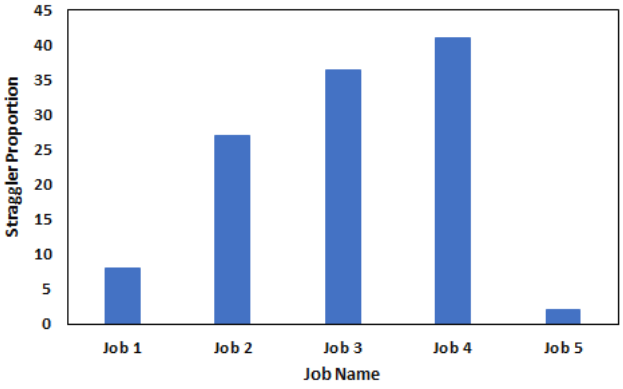

Fig. 5. Energy-aware Straggler Proportions

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 average job CPU usage rate and ω is the average duration of the respective job accordingly.

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 usage rate of μ and an average task length of ω, then 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 energy-aware stragglers vary from job to job. 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. Fig. 5 illustrates the proportions of the energy-aware stragglers within the studied jobs. The empirical analysis conducted in this research demonstrates a presence of 8% energy-aware stragglers within Job 1 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 the tasks are naturally expected to be resource hungry. Task-level stragglers pose an increased complexities in their identification and mitigation as their actual cause is often implicit. 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 job might turn out to be potential stragglers during the runtime. It is worthy of note that in some cases task-level straggler are unavoidable driven by the actual task requirements.

4.2 Long Tail Stragglers

By definition, long tail stragglers are the tasks those exhibit a runtime duration as an increased multiple of majority of the remaining tasks within the same job. Long tails stragglers are usually witnessed in a very few proportions of the total tasks within a given job, but their presence can significantly affect the overall completion time of the entire job since the job completion can only be ensured upon the completion of all the encompassed tasks. The impacts of long tails has different implications on cascaded and parallel jobs. In jobs comprising cascaded tasks, succeeding tasks are usually dependent on the outcome of the previous tasks, thus the long running tasks can potentially slow down the job progress rate, though the impacts of long tails are still trivial. But tasks in a parallel job are usually independent to each other, though long tails do not directly affect the progress rate of the other co-located tasks, long running tasks still delay the job completion to an irresistible margin. Most of the existing literature adopts a threshold value of 50% [3] to classify long tails, whereby tasks exhibiting a runtime duration 50% more than the average job duration are classified as stragglers, as shown in equation (8). In general, the identification efficiency of long tails depends on the time of classification. For instance, it is common for a data intensive tasks to progress at a slow progress rate than the other co-located tasks, classifying such data intensive tasks as long tails may lead to false positives, whereby normal tasks are wrongly classified as long tails. Other strategies of long tail identification include classification based on task progress score, progress rate, estimated task completion time etc., but analysing the efficiencies of such identification techniques to any further degree are considered to be out of scope of this paper as we are focused on exposing the energy implications of task-level stragglers.

$$d[i] > d_{avg} * 1.5 \tag{8}$$

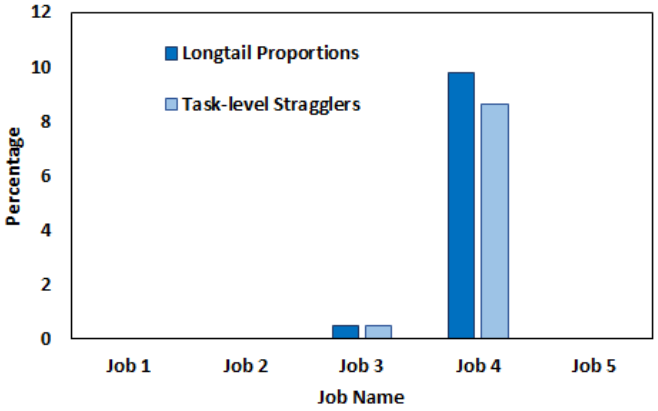

Fig. 6. Task-level Straggler Proportions

where, $d[i]$ is the runtime duration of task i, and d_{avg} is the runtime duration of the entire job, usually averaged across all the encompassed tasks. Fig. 6 presents the statistics for the proportional presence of long tails and tasks characterised as both long tails and energy aware-stragglers (task characterising both long tails and energy-aware stragglers are referred as Task-level Stragglers in Fig. 6) in the studied jobs. Among the studies jobs, Job 1, Job 2 and Job 5 comprise no long tails and any task-level stragglers, Job 3 comprise 0.5% of both long tails and task-level stragglers, in which sense Job 3 contains only a single task-level straggler. Job 4 exhibits an increased presence of long tails at 9.8% and task-level stragglers at 8.6% respectively. This few proportions of straggling tasks delay the job progress and significantly affect the Quality of Service (QoS). It is worthy of note that, this analysis has been conducted after the completion of the jobs, thus presenting actual observations than anticipations. Identification during runtime on different time-scale of execution may present us with different statistics of long tails and task-level stragglers. The reason for no long-tails and task-level stragglers are attributed to the distribution of the execution duration among the tasks encompassed within their respective jobs, which means that the tasks within a single job does not exhibit any notable deviation, as an increased multiple, than the average job duration, though the jobs may include tasks characterising low duration which may not have any notable impacts on long-tails.

5 Energy Implications

5.1 Resource Heterogeneity

It is obvious that the resources provisioned at a level more than R_T (in equation 4) are over-commitment of resources. Fig. 7 presents the statistical observations of provisioned and utilised resources in terms of the CPU cores and memory bytes for the studied jobs. Idle-resource proportions of a given task execution are the proportions of resources those not exploited in that execution. In other words, such proportions of the resources could be released to accommodate other tasks, achieving maximum level of utilisation among minimal number of resources. 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 the tasks being executed. Interestingly, jobs comprising different number of tasks are behaving heterogeneously in utilising the provisioned resources. The impacts of the energy-aware stragglers can be well observed from Fig. 7(a), where Job 1 comprises a total of 4 energy-aware stragglers (task index: 22, 24, 35 and 43) and are clearly exhibiting an increased level of CPU consumption 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 the tasks within the corresponding job. This actually results in majority of the non-stragglers to leave considerable proportions of their provisioned level of resources unutilised, causing an overall 90.5% of CPU idleness in Job 1 across the encompassed tasks. Thus, energy-aware stragglers are 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 whilst the tasks are actually being executed.

It can be observed that the CPU and memory resources are left unutilised at an average of 90.5% and 23.5% by Job 1, 87.8% and 24.5% by Job 2, 88.8% and 42.5% by Job 3, 80.8% and 21.8% by Job 4 and 16.15% and 9.75% by Job 5 respectively. Job 5 have utilised the provisioned resources to a reasonable margin than the other jobs,

which further depicts the job heterogeneity in resource consumption. Table 2 presents the observed statistics of job execution including the total job duration, average task duration, and the total core-counts consumed across all the tasks encompassed within the corresponding jobs. The job duration has been measured as the length of the longest running task within the corresponding jobs. These observed statistics further prove the 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 exhibits a duration of 4.48 minutes and consumed 6.4 cores across all the encompassed tasks, but Job 5 comprising just

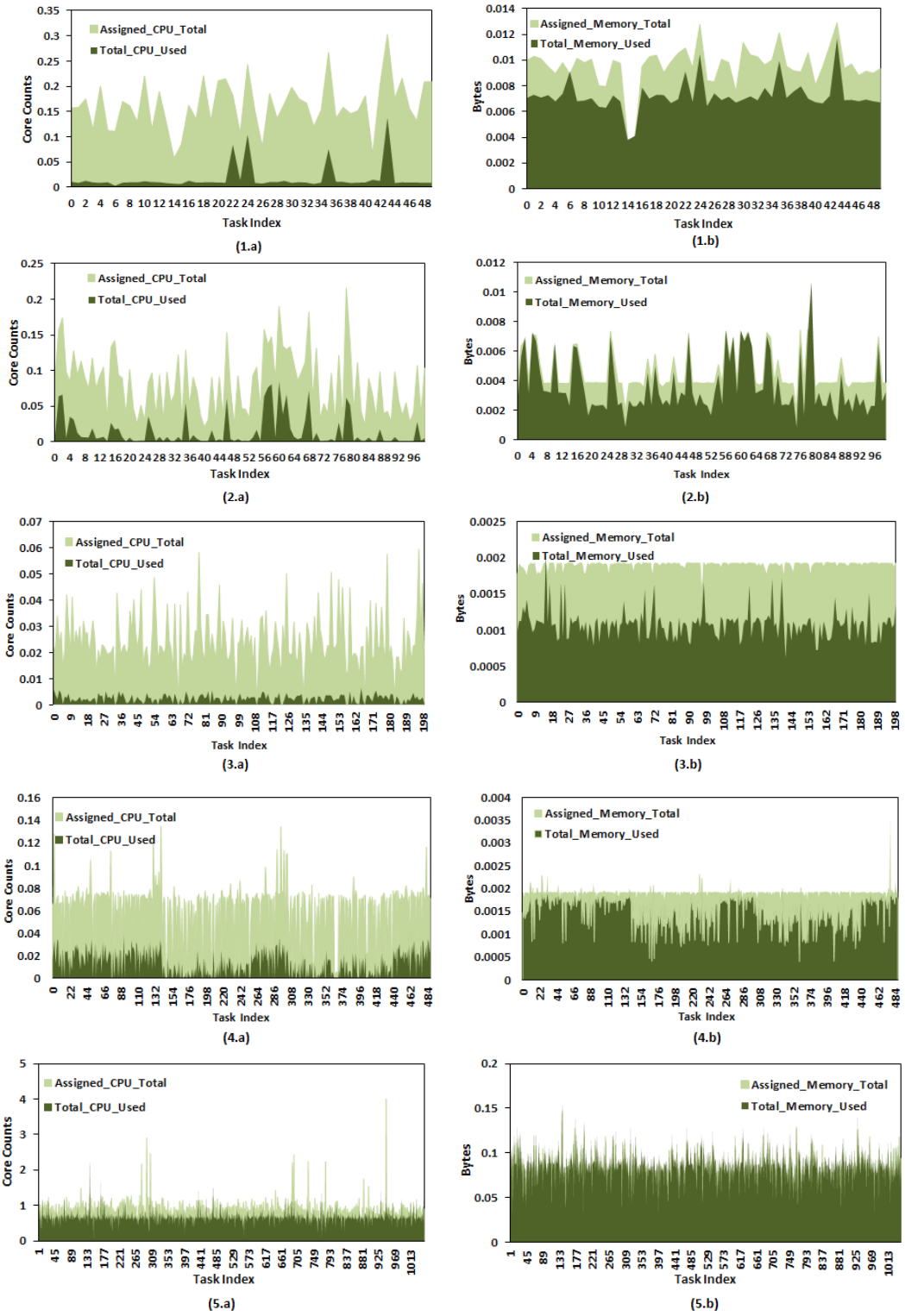

Fig. 7. Resource Provision to Usage Pattern (a) CPU (b) Memory {(1) Job 1 (2) Job 2 (3) Job 3 (4) Job 4 (5) Job 5}

Table 2: Job Execution Statistics

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

more than twice as many as tasks of Job 4 exhibits a job duration of 30 minutes (nearly 7 times of Job 4) and consumed 714.9 core counts (nearly 122 times of Job 5). Thus the encompassing number of tasks might not provide suffice inferences to characterise the 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.

We further delve into Job 4 for the purpose of conducting a comparative analysis on the energy implications of energy-aware stragglers, long tails, task characterising both long tails and energy-aware stragglers against those of the non-stragglers. Job 4 has been chosen for this analysis since it encompasses all the diverse group of task classification, a selection of 20 random tasks representing the four groups of task classification such as non-stragglers, energy-aware stragglers, long tails and task-level stragglers (denoted as both) respectively has been displayed along with the average values of all the encompassed tasks within Job 4. Fig. 8 presents the statistics of the idle CPU percentage witnessed during the actual execution for 20 random tasks from the four task groups, along with the average statistics for all the encompassed tasks within their respective groups. It can be observed that non-stragglers are exhibiting a significant increase in the proportions of incurred resource idleness than the other three task groups, at an average of 89.8%, 68.5%, 66.9% and 65.6% for non-stragglers, energy-aware stragglers, long-tails and task-level stragglers respectively. From Fig. 8(a), tasks within the their respective classification are exhibiting similar behaviours of incurring resource idleness, though there is a considerable difference in the observations of non-stragglers than the other three task groups, with the incurred idle CPU percentage for the other three task groups being very similar. Though all the task groups are incurring idle resources, the worse effect is witnessed among the non-stragglers, demonstrating more scope for resource conservation. It is clear that tasks within a single group can be treated in a similar way for task scheduling and resource provisioning, but this strategy requires a pro-active measure of forecasting the task classification prior to the task execution.

Fig. 9 presents the total consumption of CPU by the individual tasks within their respective classification, along with the overall average consumption among all the encompassed tasks within all the task group. In Fig. 9(b), the total CPU statistics reflects the total amount of cores consumed for the entire period of the task execution, whereas the CPU usage-rate reflects the mean of the CPU usage-rate witnessed at any time throughout the period of the task execution. Similar to the idle CPU statistics, a significant difference is evident between non-straggler and the other three task groups in terms of the total CPU consumption of individual tasks. But, we could witness a

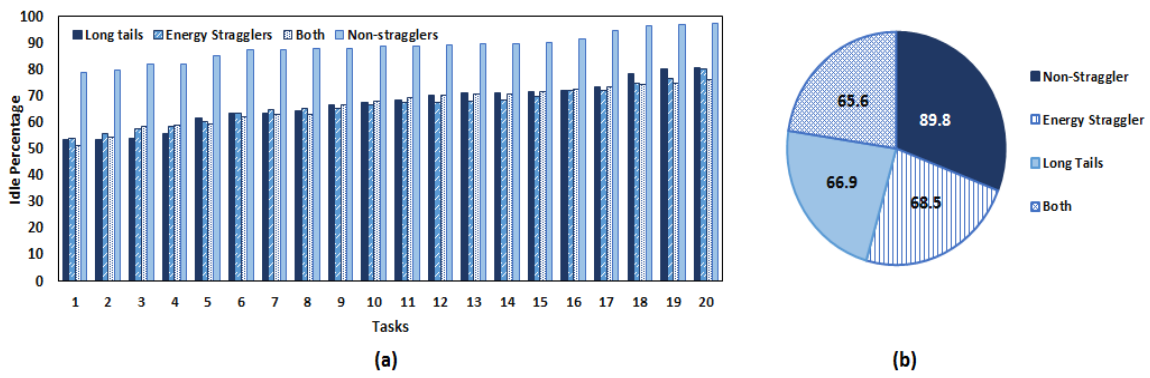

Fig. 8. Idle CPU Percentage in Job 4 (a) Individual Task (b) Overall Percentage

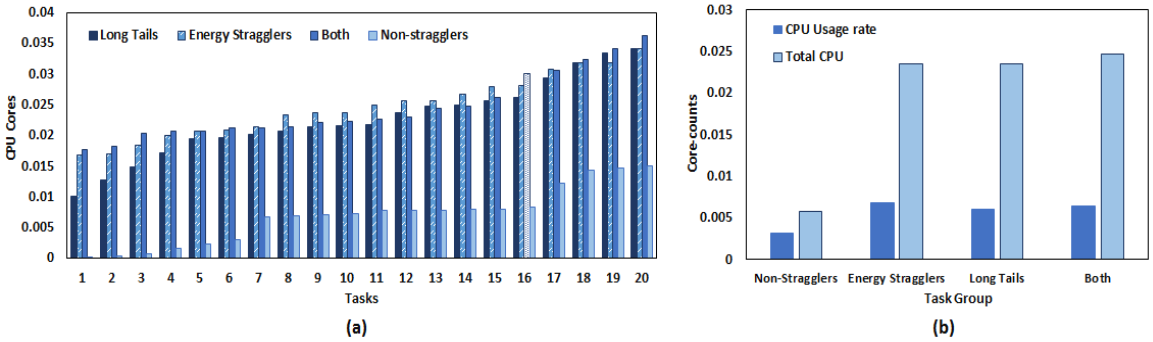

Fig. 9. CPU Consumption of tasks in Job 4 (a) Individual Task (b) Overall Average

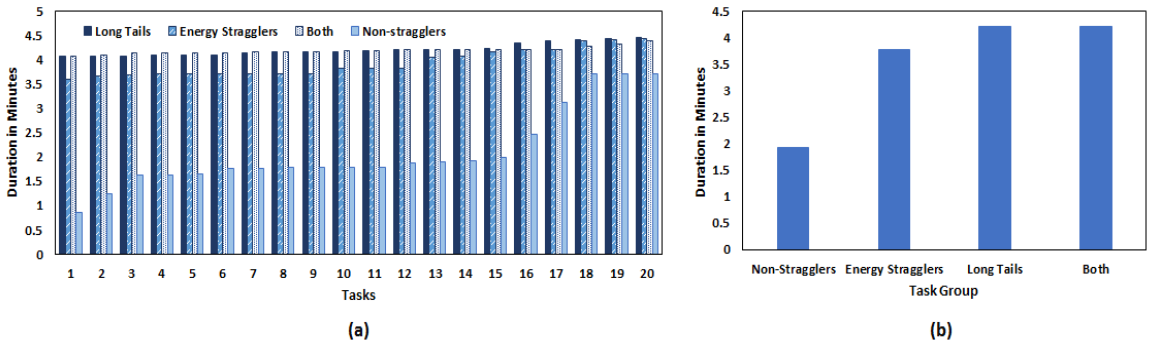

Fig. 10. Duration of tasks in Job 4 (a) Individual Task (b) Overall Average

significant heterogeneity in the core consumption among the tasks within every task group, unlike the CPU idleness statistics where tasks belonging to a single group exhibit similar percentage of idleness. In this regard, tasks are increasingly heterogeneous in terms of their resource requirements within Job 4, as we witness tasks consuming very low level of CPU to through to a higher level of CPU consumption within every task group. From the observations presented in Fig. 8 and Fig. 9, it can be postulated that service providers are already in the process of treating jobs according to requirements of individual jobs, in such a way that less resource intensive jobs are provisioned with lower level of resources which is evident in the similar level of CPU idleness among similar jobs. However, the heterogeneity among the task groups have not been effectively captured till now, which causes the providers to extravagantly over-provision the non-stragglers. Furthermore, from Fig. 9(b), the CPU usage **r**ate, which is measured as the core consumption of tasks in terms of the core-counts per second during the execution span, is remaining the nearly same for the three task group of stragglers, with the usage rate is slightly lower for the non-stragglers than the other three groups. Though, a significant difference is evident in the total amounts of CPU consumed between the non-stragglers and the other three task groups, depicting the fact that the three straggler groups are resource hungers. Given this observation, we postulate that the runtime duration of tasks is crucial in determining the overall resource requirements of individual tasks, with the presumption that tasks running longer may consume more server resources with a given job.

5.2 Duration Heterogeneity

Fig. 10 presents the observations of the total duration of individual tasks, along with presenting the overall average for all the task groups within Job 4. As expected, the long tails are exhibiting a runtime duration significantly more than majority of the non-stragglers. Surprisingly, a few proportions of non-straggler are also exhibiting a duration of more than three-thirds of those of the long tails. Furthermore the energy straggler groups are also exhibiting a runtime duration nearly equivalent to those of the long tail. There are a few immediate implications. Since non-stragglers characterise a lower CPU usage rate throughout their execution, their overall energy implications are still low despite a few tasks running longer. Though long tails run longer than the non-stragglers, all the long tails do not pose a significant energy threat, long tails exhibiting lower CPU usage rate throughout their duration can be treated as less resource intensive. But energy-aware stragglers pose significant energy implications since they do not only run longer, but also continue to exhibit an increased CPU usage rate throughout their execution duration. It is still a possibility that a few energy-aware stragglers may exhibit a shorter execution time, thus their overall CPU consumption might still be lesser than task-level stragglers, evident from the slightly lesser runtime duration of energy-aware straggler than those of the long tails and task-level stragglers, as in Fig.

10(b). The worse energy implications can be witnessed among the tasks characterising both long tails and energy-aware stragglers, since they not only run longer as equivalent to those of long tails, but also exhibit an increased CPU usage rate throughout their execution, thus their overall energy consumption level are significantly higher than the other three task groups respectively.

5.3 Findings

From the conducted analysis, Job 1 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. Despite this duration heterogeneity, Job 1 do not contain any long tails, since majority of the tasks exhibit long duration and only a minority of the tasks within Job 1 exhibit a lower duration, this phenomenon usually do not affect the completion time and service quality. Job 1 is not a resource hungry job, since they do not contain any long tails and characterise only a fewer proportions of energy-aware stragglers. But the implications of this very few proportions of energy-aware straggles in evident from the overall percentage of CPU idleness. Job 2 characterise nearly an even distribution of task length, with only a very few tasks characterise a lower duration and majority of the tasks runs between 120 and 200 seconds, in this regard task length is fairly homogeneous across the tasks within Job 2. But the resource intensity is extremely heterogeneous across the tasks encompassed within Job 2, with the resource intensity ranging from 14% through to 613% among the encompassed tasks. Job 2 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 the tasks within Job 2 would be fairly challenging in spite of the presence of 27% of energy-aware stragglers and the inherent heterogeneity of resource intensity. In terms of long tails, we a have similar phenomenon to that of Job 1. The task length of the encompassed task is evenly distributed in Job 3 ranging between 45 and 70 seconds, and further characterise fairly an even distribution of the task resource intensity ranging between 10% and 73%, a minority of the tasks exhibit lower level of resource intensity and majority of the tasks characterise evenly distributed resource intensity. Job 3 characterise 35% of energy-aware stragglers resulting from run-time execution factors and possibly node-level stragglers. In this regard, Job 3 is homogeneous in terms of the task length and heterogeneous in terms of the task resource intensity. But it also characterise a very few proportions of long tails, delaying the overall completion of Job 3. In fact, the worse energy and duration implications occur when the proportions of energy-aware stragglers and long tails are significantly lower. This is due to the fact that providers over-provision all the tasks to a level that can satisfy the requirements of a very few straggling tasks, and a very few tasks delaying the completion of the entire job.

The task length distribution 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%. Job 4 is an interesting sample since it comprises more than 40% of energy aware stragglers, which means nearly half of the tasks are energy intensive within Job 4. In this sense, 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 execution trend of Job 4 to project the scope for reducing the energy impacts of such a job kind. Job 5 is fairly homogeneous in terms of both the task length (at an average of 1578 seconds) and resource intensity distribution, and comprises only around 2% of energy-aware stragglers. It is worthy of note that the energy-aware stragglers have not exhibited any significant deviation in their resource intensiveness than those of 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 in only 16% of CPU resource idleness within Job 5. The relationship between task termination patterns and the proportional presence of energy-aware stragglers and long tails is trivial, as there is no linear dependency. For instance, Job 4 characterise and increased proportions of both long tails and energy-aware stragglers, but the terminations of tasks within Job 4 are significantly lower compared to the rest of the jobs those comprise minimal task-level stragglers. Thus, the presence of task-level stragglers are proven to influence the level of resource and energy consumption rather than exerting task terminations, particularly when the resources are provisioned at an exceeding level.

6 Related Works

Cloud workloads and the datacentre environments have been analysed from various perspectives for various reasons. Long tails have been approached by various research works [3, 6, 14-16] to mitigate their impacts on the overall job execution. Most of such works are aimed at only shortening the overall execution time of the job through early completion of long tails, but the energy implications of long tails are yet to be empirically exposed. Whilst long tails are commonly characterised as exhibiting a duration more than a defined threshold typically 50% more than the average job duration**,** straggler tasks are characterised as exhibiting a normalised duration [12] an increasing multiple of the median of the normalised duration of the other co-located tasks within the same job, with the normalised duration computed as the ratio of task execution time to the amount of work done, but assuming a static median may not scale well in spite of the task heterogeneity. Mantri is a straggler mitigation

approach [16, 17] 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 the 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.

Apart from this, workload behavioural patterns [18] have been analysed from the context of both the 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 based on the task length and utilisation ratios of CPU and memory resources for task dimensions respectively. Based on the same user and task dimensions, workload diversity [4] has been analysed 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 the task diversity across different observation periods. Workload characterisation [19] 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. The job-level workload behaviour has been studied by exploring their resource utilisation [10] at the deployed server clusters, and further the machine maintenance events have been studied to characterise machines and workloads based on the Google cluster. The inherent periodicity existing among the Cloud workloads has been characterised in our earlier works [20] to model the recurring pattern among the Cloud jobs in terms of their arrival frequency and resource intensity. The impacts of latency sensitivity among the workloads upon energy efficiency has been studied in our earlier works [21] at both the job and task levels to uncover the energy implications of workload latency sensitivity, where it has been showed that higher level of latency sensitive workloads are more resource hungry and vulnerable for terminations. Furthermore, reducing the communication latency among multimedia workloads with non-uniform destinations has been the focus of the works of [22], for the benefits of performance scalability and flexible routing. The spatial traffic burstiness in communication networks has been captured based on their communication locality and the temporal traffic burstiness has been modelled by a markov-modulated poisson process (MMPP) [23] in multi-cluster environments. MMPP has also been employed to model and analyse bursty multi-media traffic [24] in software defined networks. Furthermore, a practical time series anomaly detection model has been developed based on a relaxed form of linear programming support vector data description [25] for monitoring the operations and traffic in cloud services. Despite the existing works of Cloud based workload behavioural analysis, the presence and impacts of energy-aware stragglers and long tail stragglers upon energy efficiency have not gained enough importance. Characterising energy-aware and long tail stragglers should necessarily be integrated into workload behavioural analysis, since they can remain unnoticed at the task level and can cause significant energy implications.

7 Conclusion

This paper presents an empirical analysis of the execution profiles of real-world Cloud jobs with the motivation of exhibiting the inherent execution patterns in terms of the execution trade-off, termination patterns and the presence and impacts of task-level stragglers within jobs upon the overall energy efficiency. The analysis conducted based on real-world Cloud trace logs insist that tasks within Cloud jobs are increasingly heterogeneous and may incur heterogeneous impacts upon energy consumption and resource requirements. From the analysis presented in this paper, it is clear that every Cloud task should be viewed uniquely for energy management despite belonging to a single job given consideration to their classification category presented in this paper. Jobs can behave distinctly under different server architectures and the process capacity of the nodes processing the tasks have a significant role in determining the execution efficiency. A single agenda of resource provisioning across all the tasks within a given job may not scale well, and may incur undesirable energy expenditures in such a way that the provisioned level of resources to non-stragglers may far exceed their actual level of resource consumption. A pro-active task classification prior to the actual task execution might provide intriguing insights to the providers for better energy management of task execution at the datacentres, though this includes various practical challenges due to the process and environment heterogeneity of Cloud datacentres and Cloud jobs. Furthermore, incorrect classification and resource estimation of Cloud jobs may incur disastrous effects in Cloud datacentres through task terminations resulting from under-estimation, and energy wastages resulting from over-estimation of resource requirements at an extravagant level.

The analysis presented in this paper finds applications in predictive analytics aimed at a prior estimation of the task resource requirements and potential behaviours, and in resource scheduling and task allocation frameworks where tasks under similar classification can be sent to batch processing in similar process scenario. This helps to avoid a generalised view of all the tasks within a single job and thus energy conservation can be achieved for a larger proportions of tasks within jobs. As a future work, we plan to develop an analytics model that can forecast the energy-aware stragglers along with estimating their resource requirements before processing the jobs, ultimately to reduce the level of resource idleness and undesirable energy wastages during task execution. In

addition, the analysed job in this paper encompasses only parallel tasks, so we also plan to analyse job encompassing serial jobs and their corresponding energy related implications in the future.

Acknowledgement

This work is partially supported by the National Natural Science Funds of China under Grants No. 61502209 and 61502207, the Natural Science Foundation of Jiangsu Province under Grant BK20170069, UK-China Knowledge Economy Education Partnership. Lu Liu is the corresponding author.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] P. Garraghan, P. Townend, and J. Xu, "An Analysis of the Server Characteristics and Resoruce Utilization in Google Cloud," presented at the International Conference on Cloud Engineering, Redwood City, CA, 2013.

[6] 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.

[7] J. Rosen and B. Zhao, "Fine-Grained Micro-Tasks for MapReduce Skew-Handling," 2012.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] "Google Cluster Data V2," Google, Ed., 2 ed, 2011.

[14] N. J. Yadwadkar and W. Choi, "Proactive Straggler Avoidance using Machine Learning," University of Berkeley, University of Berkeley2012.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. Panneerselvam, L. Liu, and N. Antonopoulos, "Characterisation of Hidden Periodicity in Large-Scale Cloud Datacentre Environments," presented at the 13th IEEE International Conference on Green Computing and Communications, Exeter, 2017.

[21] J. Panneerselvam, L. Liu, N. Antonopoulos, and M. Trovati, "Latency-Aware Empirical Analysis of the Workloads for Reducing Excess Energy Consumptions at Cloud Datacentres," presented at the IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, 2016.

[22] Y. Wu, G. Min, D. Zhu, and L. T. Yang, "An Analytical Model for On-Chip Interconnects in Multimedia Embedded Systems," *ACM Transactions on Embedded Computing Systems (TECS),* vol. 13, November 2013 2013.

[23] Y. Wu, G. Min, K. Li, and B. Javadi, "Modeling and Analysis of Communication Networks in Multicluster Systems under Spatio-Temporal Bursty Traffic," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, pp. 902 - 912, 14 July 2011 2013.

[24] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance Modelling and Analysis of Software-Defined Networking under Bursty Multimedia Traffic," *ACM Transactions on Multimedia Computing, Communications, and Applications,* vol. 12, December 2016 2016.

[25] C. Huang, G. Min, Y. Wu, Y. Ying, K. Pei, and Z. Xiang, "Time Series Anomaly Detection for Trustworthy Services in Cloud Computing Systems," *IEEE Transactions on Big Data,* vol. PP, 01 June 2017 2017.