IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

FAILURE-RELATED ENERGY WASTE IN WIDE-RANGE CLOUD ENVIRONMENT

Pruthvin.R*
* Dept of ISE, BMS College of engineering, Bengaluru,India-560019.

ABSTRACT

Cloud computing providers are under great pressure to reduce operational costs through improved energy utilization while provisioning dependable service to clients; it is therefore extremely important to understand and enumerate the explicit impact of failures within a system in terms of energy costs. This paper presents the first inclusive analysis of the impact of failures on energy consumption in a real-world large-scale cloud system (comprising over 12,500 servers), including the study of failure and energy trends of the spatial and temporal environmental characteristics. Our results show that 88% of task failure events occur in lower priority tasks producing 13% of total energy waste, and 1% of failure events occur in higher priority tasks due to server failures producing 8% of total energy waste. These results highlight an unintuitive but significant impact on energy consumption due to failures, providing a strong foundation for research into dependable energy-aware cloud computing.

KEYWORDS: Cloud computing, energy-efficiency, failure analysis.

INTRODUCTION

PREFACE

Dependability research is greatly enhanced by analysis of failures within real-world systems, as researchers and practitioners are able to develop and tune highly effective dependability mechanisms based on realistic empirical data for a given domain. Such analysis is particularly crucial for the development of the energy-aware dependability mechanisms that are increasingly required in modern large-scale systems, particularly in the Cloud computing industry. According to the National Institute of Standards and Technology (NIST) [1], Cloud Computing is ``a model for enabling ever-present, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction". In such environments, practitioners are under great pressure to ensure Quality of Service (QoS) levels agreed with customers are fulfilled whilst reducing operational cost to maximize profit [2], [3].A substantial amount of this operational cost is energy waste [4], and a significant amount of this energy waste is produced by failures due to software and hardware crashes [5], [6]. Important work has been developed in power/energy modelling of large systems such as those presented by [7], [8]. Additionally, a significant number of resource management techniques such as [9], [10] have been proposed to reduce such energy waste while maintaining the expected performance levels. However, these approaches have relegated the impact of software and hardware crashes on the energy-efficiency of the analyzed systems. In this context we define software and hardware crashes as a fail-stop failure, i.e. processes or hardware fail by stopping without doing anything, and energy waste as the amount of energy expended on computational work that is lost due to failures. Crash failures result in a loss of work and require calculation to be repeated after recovery, thus any energy used prior to and including recovery is in fact wasted in contrast with failure-free execution. As mentioned in [11] and [12], another factor that increases resource waste in systems is the amount of rolled-back computation after failure. Recent theoretical analyses and studies have highlighted failures as a source of inefficiencies that increment energy consumption in large-scale data centres, and have proposed a number of mechanisms to reduce energy waste including resource selection [6], rollback [11] and resource scheduling [13]. Unfortunately, all of these analyses are derived from theoretical data and provide no insight about the true characteristics and dimensions of energy waste caused by failures.

On the other hand, current failure analyses of large-scale systems, limited due to confidentiality concerns, focus on analyzing failure statistics in terms of Time Between Failures [14], [15], root cause failure [16], [17] of systems, and performance implications. However, all of these practical analyses neglect quantifying the impact of failures in terms of energy. The gap between these two strands of (theoretical and practical) work highlight the urgent need for failure analyses that consider the impact of failures on energy consumption. Analyzing the impact of failures on energy consumption is significant for two main reasons. Firstly, it enables researchers and practitioners to understand the characteristics and dimensions of the problem in order to create concrete scenarios for decision-making. For example, in order to know what dependability mechanisms to use and when to apply them effectively, data centre administrators require awareness about the variables and conditions under which waste is produced. Secondly, it is important for practitioners (e.g. Cloud service providers) to understand the impact that both task and server failures have on data centre energy waste and operational efficiency. In this circumstance, we define a task as a program, possibly consisting of multiple processes running on a single machine assigned and performed within the Cloud. A server is defined as the hardware as well as the system software managing the hardware e.g. OS, software supporting communications, and networking. This understanding allows Cloud service providers to create a baseline and points of improvement to reduce the energy waste as long as the QoS offered to the customers is maintained. The major contributions of this paper are:

- The first practical task and server failure analysis of a large-scale heterogeneous Cloud computing environment that considers the impact of failures on energy waste.
- Quantification of energy waste produced by tasks and server failures within a large-scale system to support and boost the application of current theoretical models for energy savings under specific conditions.

The analysis is performed on the Google Cloud trace log which consists of over 12,500 servers over the period of 29 days containing over 25 million submitted tasks. In order to determine the impact of failures, we characterize all events that result in incomplete task termination in terms of elapsed time, server host, event type, and priority. We then further alter these events to identify task and server failures based on meaningful observations from the dataset and supporting literature, including the trace log specification. Finally, we conduct failure analysis of the temporal and spatial characteristics of tasks and servers, considering energy waste based on realistic energy models. The paper is structured as follows: Section 2 discusses related work; Section 3 introduces the dataset and coarse grain analysis of the trace log; Section 4 discusses the methodology used; Section 5 shows the general statistics of the failure-analysis; Section 6 describes the impact of energy waste due to failures within the system environment; Section 7 discusses the application of this work and Section 8 concludes the paper and outlines future research.

RELATED WORK

The importance of failure analysis and their impact on large-scale computing environments has been highlighted and partially addressed in previous work. Furthermore, the impact of failures on the waste of resources and increasing energy consumption has been identified as a critical factor and has been approached from different perspective. This section introduces the most relevant works in both areas. Schroeder, et al. [15] analyze the failure data of 29 different systems over the period of 9 years. They present the MTBF and MTTR of nodes as well as root cause of failures. Their results indicate that average failure rates vary across systems significantly, that there exists a relationship between machine failure rate and workload intensity and that time between failures and renewal times are well modelled by Weibull and lognormal distributions, respectively. Liang, et al. [18] presents a study of the composed RAS (Reliability, Availability, and Serviceability) event logs from BlueGene/L over a period of more than 100 days. The objective is to examine and distinguish the failure events as well as the correlation between fatal and non-fatal events to minimize the negative impact of system performance. Hui, et al. [19] analyze job failures in a large-scale data intensive Grid composed of 180 active sites and 34,121 servers over three time periods totalling 24 days. The objective is to characterize the interval times and life spans of failed jobs looking for cross-correlation structures and statistical models to fit the failure data and measure its impact on system performance. Ropars, et al. [11] underline the amount of resources wasted with respect to computing power or energy due to computation rolled-back after failures. The main objective of this work is the design of hierarchical rollback-recovery protocols based on a combination of coordinated check pointing and message logging to reduce the waste produced by redundant computation. Nguyen, et al. [6] discusses the inefficiencies and increment of energy consumption due to failures in large-scale data centres. Authors approach this energy waste problem by proposing a resource selection scheme to reduce the number of tasks re-submissions that result from failures during their life cycle. Quiane-Ruiz, et al. [13] discusses the wasted resources and performance impact caused by task failures in Map Reduce environments due to automatic re-scheduling. The main objective is to reduce the need for re-executing completed map tasks by re-computing only intermediate data that were processed by local reducers and hence does not require moving to another node for processing using fast tracking algorithms. From the literature examination and the current state of the art it is observable that there is a clear gap between failure analysis approaches and the design of mechanisms to reduce the energy waste caused by failures. On the one hand, the analyses are completely focused on how the failure characteristics are correlated to performance drawbacks and completely neglect the impact of those failures on the energy waste. On the other hand, theoretical approaches highlight the importance of reducing the waste produced by failures without providing any insight about the characteristics and dimensions of the addressed problem. As a result, this creates the requirement for complete failure analyses that also include the energy impact in real production environments.

DATASET

DATASET EXPLANATION

The Google trace log features 12,532 individual servers, spanning 29 days of operation, and includes over 48 million task submissions. Furthermore, the trace provides normalized CPU, Memory and disk utilization per task every 5 minutes. The trace log is available at [20], and information about the data schema and normalization process can be found in [21]. Due to the large volume and unprocessed nature of the data within the trace log (approximately 400GB of Comma Separated Files); it was necessary to deploy a 50 node Hadoop Map Reduce [22] cluster running HIVE [23], a data warehouse system, for storage and query computation. This allowed us to extrapolate data from the trace log in a timely manner (reducing individual query times from approximately 6 days to 15 minutes), as well as conduct event filtering and energy profiling of servers and tasks as described in Section 4.

COARSE-GRAIN ANALYSIS

Table 1 depicts the general statistical properties of the trace log as well as the workload of the data centre. Workload is defined as the specific amount of work computed or processed in the data centre with defined resource requirements and consumption patterns. The overall workload in a data centre is commonly shaped by user submission patterns, task characteristics and scheduling policies. In the analyzed trace log, the workload is created by 930 different users submitting 650,000 different jobs comprising just over 25 million tasks.

Number of Users	930
Number of jobs	650,000
Number of submitted tasks	25,375,377
Number of scheduled tasks	47,351,173
Number of task events	144,010,768
Number of server events	37,780

Table 1. General trace log Statistics.

According to [21], a task in the system is represented as a Linux program consisting of one or more processes that can be encapsulated within a job. In this study, we focus on tasks in order to achieve fine granularity, and observe that there are over 47 million tasks scheduled; this is due to the possibility that a submitted task can be rescheduled multiple times. Moreover, there are 144,000,000 and 38,000 recorded events corresponding to tasks and servers, respectively. Users have different submission patterns ranging from a single task to 3.5 million tasks over the trace log time span. From Fig. 1(a), it can be seen that almost 94% of the users submit a small proportion of the total tasks within the Cloud environment, whilst the remaining 6% of users are responsible for the generation of the heavy workload due to their larger submission rate. In addition, Fig. 1(c) depicts the variability of task submission per day over the trace log time period, ranging from 678,929 to 4,940,423 tasks submitted daily. Each submitted task consumes CPU, memory and disk space at different rates and durations and is presented as normalized values within the trace log. All values of resource consumption are normalized based on the largest server capacity within the trace log. As observed in Fig. 1(b), approximately 98% of tasks gain small to moderate resource consumption while only 2% incur large resource demands.

Fig 1. Coarse-gain analysis of task

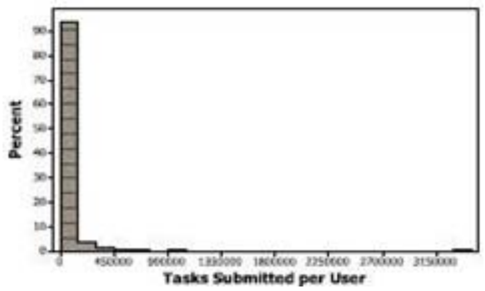

Fig 1(a) submission proportions

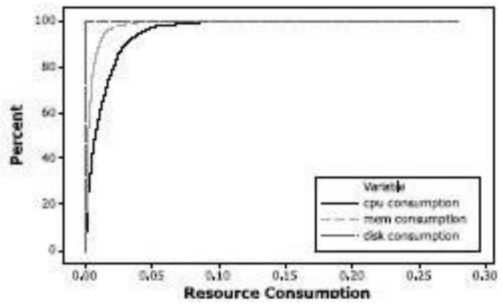

Fig 1(b) resource consumption

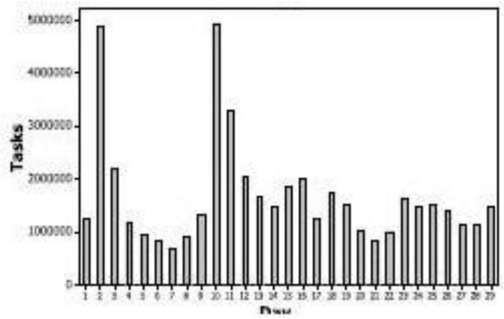

Fig 1(c) daily submissions

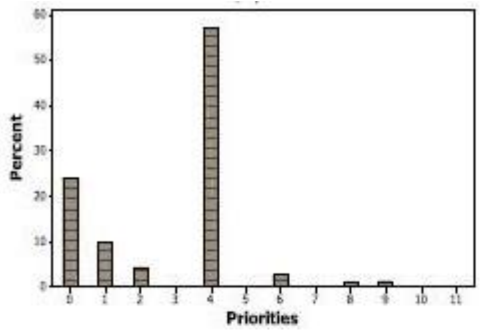

Fig 1(d) priority submissions

Furthermore, the proportion of resource consumption is strongly correlated to the duration of the task. Therefore, short tasks consume on average a smaller amount of resources in comparison to long tasks. The sum of the various utilization patterns and the placement constraints create a highly heterogeneous workload environment typical of multitenant Cloud data centres. The scheduling policy of the system is mainly based on task priorities. The priority scale for tasks ranges from 0 to 11 to indicate the lowest and highest priority, respectively. When required, low-priority tasks are evicted to produce resources to high-priority tasks creating further re-submission events that have a high impact on the overall workload. Fig. 1(d) illustrates the distribution of task priorities in the analyzed data centre, where it can be observed that lower priorities (0-8) represent the 99% of the total submitted tasks, with higher or production priorities (9-11) accounting for the remaining 1% approximately.

METHOD

The main purpose of this paper is to summarize and enumerate the impact of failures (tasks and servers) in terms of energy waste within the analyzed trace log. This section describes the methodology of analysis which is shaped by three main steps: event sampling, failure event detection, and failure energy impact analysis.

EVENT SAMPLING

In order to analyze the energy waste produced by TEs and failures, it was necessary to characterize the events in the trace log by event type, elapsed time, task priority, and server host. While the event type, priority and server host are explicitly provided in the trace log, the elapsed time needs to be calculated based on the registered timestamps in the task event log. We refer to elapsed time as the period of time between at asks being scheduled and the occurrence of a TE. The elapsed time is calculated by subtracting the latest ``schedule" event timestamp from the TE timestamp. The sample for the analysis presented in this paper considers all the TEs except those that have trunked elapsed time i.e. all tasks that have ``schedule" events that occur before the trace log span. This condition mainly excludes the monitoring services running in the Cloud grouped in priorities 10 and 11 which have started running prior to the start of the trace log observation period. The principal reason for this exclusion is that these events do not allow the calculation of realistic energy consumption and can considerably skew the results. As a result, the sample consists of 25,927,826 TEs that represent 87% of the total TEs recorded in the trace log.

FAILURE EVENT DETECTION

Due to the vague definition and utilization of the different TEs, it is impossible to identify all the failures solely from the previous characterized sample. Therefore, it has been necessary to create a set of assumptions – supported by relevant literature and the evidence of system behaviour derived from data in the trace log - to distinguish failure events apart from TEs. Using this approach, we have been able to filter the event logs to identify two types of failures. Server failures are characterized as a software or hardware crash failure; based on the observations of the data, when a server failure occurs, all tasks executing on the server are subsequently terminated. As mentioned in [21], a REMOVE event for a server is the result of either a server failure or maintenance. Due to this uncertainty, it is not possible to distinguish server failure from planned server maintenance, as postulated within. However, data observations show that almost all REMOVE events are initiated with a number of tasks still running on the server. As a result, we classify all REMOVE events as server failures (agnostic of maintenance), as this event results in tasks deviating from correct service. Task failures are characterized as software crash failures. Task failures filtered from the task event log are identified by tasks that exhibit the FAIL event. In [21], FAIL events have been explicitly defined as the result of a software crash of the task. In addition, we have also filtered the task event log to identify TEs due to server failures. These tasks are of importance to this work to enable studying of the energy impact of failures within the trace log. This filter was applied to tasks that were rescheduled by KILL, EVICT or FAIL events whose timestamp resided within the calculated time period of server downtime. It has been well understood that the root causes of task and server failures might be physical, design (typically software), human-machine interaction faults, or even malicious attacks, or a combination of them. In reality, transient hardware faults, hardware design faults and software bugs often cause similar system behaviour. Within this work, we decided not to distinguish the root cause of a failure for servers due to uncertainty within the trace log. However, we are able to filter tasks failures that are resultant of hardware or software crashes within servers and software crashes within a task.

ENERGY ANALYSIS

The energy consumption per TE is calculated based on the power profile of the server where the tasks were running and the average CPU load imposed by tasks involved. Servers within the trace log are heterogeneous in nature and are aggregated in 3 main platforms which each contain a different combination of micro-architecture, chipset version, and CPU capacity. Each platform has one or more configurations that are merely variations of memory capacity. Precise details of servers and their platforms are masked in the trace log. Consequently, the characteristics and power profiles of real systems are considered from the results of SpecPower2008 benchmark. The actual architectural characteristics and power profiles of Google servers may vary from those presented in the SpecPower2008 results, therefore the best assumption that can be performed is matching the profiles of realistic data centre servers based on the CPU and memory capacity presented in the trace log. Namely, the selection of the specific profiles is based on the proportional similarity between obfuscated server capacities of CPU and Memory in the trace log and the actual server configurations provided in the SpecPower2008 results. This enables us to obtain energy calculations proportionally close to those that could be measured directly from the actual Google cluster. For example, ProLiant platform has approximately 25% of the CPU and memory of the PRIMERGY platform which is the same as Platform A that has 25% of CPU and memory capacity of Platform B in the trace log. The same case applies between 1022G-NTF platform and PRIMERGY in relation to the proportion of the obfuscated platforms B and C. The profiles presented by SpecPower2008 are preferred over other available server benchmarks because the results are obtained following a strict methodology of experimentation and monitoring.

In comparison to other resources, CPU consumes the largest amount of the total power demand in physical servers. Therefore, it is assumed that all the con- figurations from the same platform share the same power profile as they share the same CPU micro-architecture and capacity.

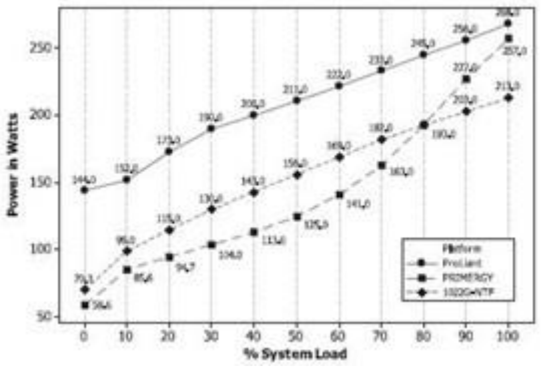

Fig 2. Power models of the selected platforms.

Each platform has a unique power profile that consists of 10 measurements ranging from 0% to 100% of system utilization with increments of 10% and their correspondent power consumption. While the system utilization is measured in terms of computed Server Side Java Operations (SSJOPS), power is measured in watts. The power models of the 3 selected platforms are presented in Fig. 2. With the energy calculated per event, the statistical analysis is conducted in two dimensions: First, we analyze the frequency and distribution of events per day to determine how the intensity of specific event type occurrence and their variation impact energy waste. Second, we analyze the elapsed time per event type and task priority to determine the contribution of each one to the total produced energy waste.

FAILURE ANALYSIS

The following two sections present the temporal and spatial characteristics of filtered FAIL events from the TEs and server REMOVE events to study the impact of energy wasted for tasks and server failures, respectively. Before presenting the subsequent failure analysis, it must be emphasized that each failure event for servers and tasks may not necessarily correspond to a unique failure, and that failure events that are temporally close together may be caused by the same failure. This vagueness is a result of the lack of precise data concerning failures, even after filtering failure events apart from TEs. However as stated in previous works [16] and [18] it is incredibly difficult to identify the root cause as well as the duration of a failure.

Furthermore, it is possible that the total energy waste calculated for tasks and servers could potentially be reduced when considering check pointing of tasks. However, when consulting the supporting literature of the trace log, as well as analyzing and characterizing task and resource utilization, we found no evidence of tasks exhibiting behaviour of check pointing. In addition, termination events result in work performed prior to termination to be lost. This behaviour in a subset of tasks is supported in [13] and [22] which states that ``a task failure is an intrusion on a running task, requiring the system to re- execute the interrupted task", indicating that a failure results in a task being restarted from the beginning of execution. Finally, assigning theoretical check pointing frequencies and overhead time to different task types would be subjective and arbitrary, distorting the behaviour of tasks within the system.

Total Server Failures	8954
Number of Servers Failed	5056
Task Failures	13,572,457
Unique tasks failed	829,738

Table 2. General failure Statistics

This also potentially leads to an additional increment in energy usage since some check pointing mechanisms, such as memory or HDD logging, consume a considerable amount of energy. Table 2 depicts general statistics of the failure events within the trace log after filtering failure events from TEs as described in Section 4.

It is observable from Table 2 that 3.26% and 39% of the total unique tasks and servers within the trace period experience one or more failures, respectively. This indicates that a relatively small number of unique tasks (3.26% of total tasks) account for a large amount (52%) of failures. This percentage of failure is comparable to other distributed systems studied in [19], where 5-8% and 2.4% of jobs failed after a period of execution, respectively.

ENERGY WASTE IMPACT ON THE SYSTEM

From Fig. 4(a) and (b), it is possible to observe that although in general terms tasks with priority 0 are the most affected by FAIL events in up to 80% of the cases, this value is inflated by very few days in which its proportion is extremely high in comparison to other days. This is clearly produced by the activity of specific users that not only submit a large number of tasks, but also introduce a large number of resubmissions of tasks that repeatedly fail as a result of crash-loops.

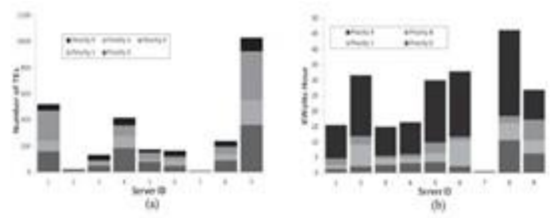

Fig 3. Top nine failed nodes (a) Failed tasks,(b)Energy waste.

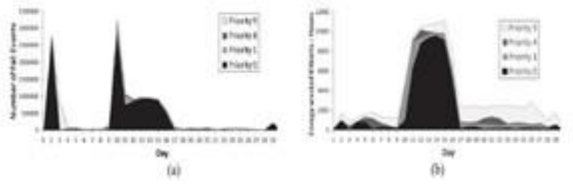

Fig 4. Daily task failure (a) Number of events (b) Energy waste.

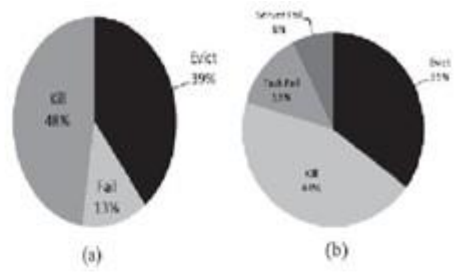

Fig 5. Energy Waste of trace log (a) TEs (b) TEs and failure events breakdown.

Without the influence of those users, the number of events and energy waste produced by fails on Priority 0 is very similar to the other days, even lower than Priority 9 that proportionally produce more energy waste. Filtering the proportion of EVICT and KILL events that relate to server failures, task and server failures contribute 13% and 8% of the total TE energy waste, respectively. This energy waste translates to 9.91% and 6.10% in the context of the total energy consumption for the entire data centre. As observed in Fig. 5, this is a substantial amount of energy waste in large-scale environments and represents an important point of improvement that can lead to reducing operational costs while the provisioned QoS is maintained. By minimizing the impact of server failures on long running tasks it is possible to reduce energy waste by as much as 8%. In this context, mechanisms such as those presented in [5] for check pointing can be applied in order to maintain the progress of long running tasks and avoid repeated computation. However, in the case of task FAIL events, check pointing can increase the waste of resources due to the high frequency of failure occurrence and the low priority of the majority of tasks affected. In this case, improved policies to avoid highly recurrent resubmissions such as those described in [4] can reduce the proportion of TE energy waste by up to 13%. From the presented analysis it is clear that the negative effect of failures on the energy waste for the studied environment is produced in two well defined scenarios. The first one involves task FAIL events that affect low priority tasks 80% of the time with a MTBF of 0.97 hours. Moreover, it is mainly driven by top 10 users which produce close to the 90% of FAIL events due to crash-loops, especially during days 2 and 10. The second scenario involves EVICT and KILL events produced by server failures, affecting 30% of tasks with high priority tasks with a MTBF of 58.72 hours. This scenario is driven by hardware and software failures of servers uniformly distributed according to the size of their population. That is, not one server type is affecting other types in proportion to the number of its available servers during the observation period.

APPLICATION OF WORK

Although the results obtained in this analysis are specific to the studied environment, the findings in this work can be used as a baseline of analysis for similar practical systems. Researchers and practitioners can use the derived observations and conclusions to develop, enhance and evaluate energy-aware dependable mechanisms as well as identify specific scenarios when and where failures have a significant impact on the energy waste within the system. To give a practical example, failure-aware scheduling focuses on developing more effective resource management to increase task reliability and system availability. Current techniques focus on improving the availability, reliability and performance of the system and do not consider energy-efficiency. As shown in Fig. 3, it is observable that different server platforms exhibit different energy profiles at the same system load, which can result in failure-aware algorithms scheduling tasks and jobs onto energy-inefficient servers as they only consider the server capacity. As a result, it is possible to develop a failure-aware scheduling algorithm that selects servers that provide a balance between ideal server reliability and energy-efficiency for executing a task. The work presented within this study can be leveraged in order to evaluate the effectiveness of such a mechanism and quantify the improvements in energy-efficiency based on empirical data as opposed to relying on theoretical values for energy waste of software and hardware in large-scale systems. Additionally, the findings presented in this work can be used in following ways:

- To support and adjust the claims of failure energy- aware mechanisms according to the characteristics of a real environment. Although the existing theoretical analyses remark that significant energy waste is produced by failures and propose elaborated mechanisms to address this problem, they do not present or discuss any insight into the actual amount of energy waste based on empirical findings. In fact, they never contrast the claimed improvements against quantified waste in real operational scenarios.
- To assist practitioners from similar environments to decide the appropriate dependability mechanisms and when to apply them in order to maximize the expected effectiveness. Failures within large-scale systems are the norm, rather than the exception. As a result, Cloud providers need to decide what type of faults they should invest time and resources in correcting when considering their system impact in terms of QoS, energy waste and development cost. For example, the work in [12] states that there are a number of limitations in applying fault tolerant run-time techniques such as check pointing, as they can potentially not only introduce high overhead, but more importantly difficulty in deciding when and where to apply such mechanisms. The results presented in this work provide quantified dimensions for Cloud failures and energy waste which can be leveraged when making decisions to deploy a mechanism such as check pointing, as well as on what type of workload.
- To delimitate the energy waste produced by the “normal" operational inefficiencies and those introduced by task and server failures. Understanding the sources and dimensions of these inefficiencies helps to identify the most effective courses of action to reduce their negative impact.

From the studied environment it is noticeable that although failures introduce close to the 21% of the total energy waste, this is still lower than the 79% introduced by scheduling operations such as KILL and EVICT. Such findings are critical in future research areas which aim to realistically model large-scale system environments.

CONCLUSION

In this paper we have presented and discussed the results of the first failure analysis on a large-scale heterogeneous Cloud environment that also quantifies the impact of task and server failures in matters of energy consumption. These results demonstrate that re-submissions caused by task and server crash failures create a significant amount of wasted computation that results in 21% of the total energy waste for ``Termination Events" (TEs) within the analyzed trace log. Additionally, the results also expose the existence of two main scenarios where failures lead to energy waste. The first is related to task failures frequently occurring on low priority tasks after a short MTBF and comprising 13% of the total energy waste by TEs. The second is related to server failures that seldom occur and affect high priority tasks after a long MTBFand comprises 8% of the total waste accounted by TEs. From these results a number of observations and important conclusions can be made: fi Energy waste can be produced by failures under very different conditions creating well defined scenarios depending on the system characteristics and user pat- terns. For the analyzed environment it is observed that the influence of very few users and the priority schema implemented by the scheduler affects the energy waste in the data centre significantly.

- Analyzing the impact of failures in energy consumption is critical to select the most appropriate mechanisms to reduce the waste while QoS is maintained. Depending on the characteristics of a scenario the introduction of inappropriate mechanisms may actually increase the waste rather than reduce it, for example the use of failure-aware scheduling and migration for highly frequent failures on low priority tasks.
- Temporal and spatial failure analysis can assist in deter-mining the most important factor that contributes to the produced energy waste. For the analyzed environment, the tasks' elapsed time has a decisive impact on the amount of generated waste. However, depending on the different energy profiles of the servers in the data centre, the location of the tasks also plays an important role.
- Filtering the failures from the general TEs in the data centre can lead to establish points of improvement to reduce the energy wasted not only by failures but also by the ``normal" system operational characteristics. In the analyzed environment, after failure filtering, EVICT and KILL events from the normal scheduling operation still account for 35% and 44% of energy waste, respectively.

As future work we plan to model the failure characteristics observed in this paper to enhance energy-aware mechanisms and evaluate those considering parameters from realistic environments using our approach for workload modelling and simulation. This is critical since current approaches have not presented fair comparison parameters in real production environments. Additionally, we are planning to analyze further the failure characteristics within the system. For example, we have observed that a small proportion of the failure events analyzed within the trace log are caused by abnormally high rates of CPU and Memory utilization. Finally, it is also important to design and implement mechanisms such as scheduling policies and fault tolerant strategies that can dynamically adapt to the different energy-failure scenarios which exist under real operational conditions.

REFERENCES

[1] P. Mell and T. Grance, “NIST definition of Cloud computing”, Nat. Inst. Stand. Technol., 2009.
[2] A. L. Freitas, N. Parlavantzas, and J. Pazat, “Cost reduction through SLAdriven self management”, in Proc. 9th IEEE Eur. Conf. Web Services, Sep. 2011, pp. 117fi124.
[3] U. Hoelzle and L. A. Barroso, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1st ed. San Rafael, CA, USA: Morgan and Claypool, 2009.
[4] M. Pedram,”Energy-efficient datacenters”, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 31, no. 10, pp. 1465fi1484, Oct. 2012.
[5] A. Gainaru, F. Cappello, M. Snir, and W. Kramer,”Fault prediction under the microscope: A closer look into HPC systems”, in Proc. IEEE Intl. Conf. High Perform. Comput., Netw., Storage Anal., Nov. 2012, pp. 1fi11.
[6] T. Nguyen andW. Shi,”Improving resource efficiency in data centers using reputation-based resource selection”, in Proc. IEEE Intl. Green Comput. Conf., Aug. 2010, pp. 389fi396.
[7] M. vor dem Berge, W. Christmann, E. Volk, S. Wesner, A. Oleksiak,T. Piontek, et al., “CoolEmAllfiModels and tools for optimization of data center energy-efficiency”, in Proc. Intl. Conf. Sustain. Internet ICT

Sustainabil., Oct. 2012, pp. 1fi5.
[8] X. Fan, W. Weber, and L. A. Barroso,"Power provisioning for a warehouse-sized computer",ACM SIGARCH Comput. Archit. News, vol. 35, no. 2, pp. 13fi23, Jul. 2007.
[9] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Future Generat. Comput. Syst., vol. 28, no. 5, pp. 755fi768, May 2012.
[10] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle,"Managing energy and server resources in hosting centers", ACM SIGOPS Oper. Syst. Rev., vol. 35, no. 5, pp. 103fi116, Dec. 2001.
[11] 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", in Proc. 17th Int. Conf. Parallel Process., 2011, pp. 567fi578.
[12] L. Yudan, R. Nassar, C. B. Leangsuksun, N. Naksinehaboon, M. Pfiaun, and S. L. Scott,"An optimal checkpoint/restart model for a large scale high performance computing system", in Proc. IEEE IPDPS, Apr. 2008.
[13] J.-A. Quiane-Ruiz, C. Pinkel, J. Schad, and J. Dittrich, "RAFTing MapReduce: Fast recovery on the RAFT", in Proc. IEEE 27th ICDE, Apr. 2011, pp. 589fi600.
[14] R. K. Sahoo, M. S. Squillante, A. Sivasubramaniam, and Y. Zhang,"Failure data analysis of a large-scale heterogeneous server environment", in Proc. IEEE Int. Conf. Depend. Syst. Netw., Jul. 2004, pp. 772fi781.
[15] B. Schroeder and G. A. Gibson,"A large-scale study of failures in highperformance computing systems",IEEE Trans. Depend. Secure Comput., vol. 7, no. 4, pp. 337fi351, Oct. 2010.
[16] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer,"Failure data analysis of a LAN of Windows NT based computers",in Proc. 18th IEEE Symp. Rel.Distrib. Syst., Oct. 1999, pp. 178fi187.
[17] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in Proc. 4th USENIX Symp. Internet Technol. Syst., USENIX Assoc., vol. 4. 2003, pp. 1fi15.
[18] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "Blue-Gene/L failure analysis and prediction models",in Proc. IEEE Int. Conf.DSN, Jun. 2006, pp. 425fi434.
[19] L. Hui, D. Groep, L.Wolters, and J. Templon, ``Job failure analysis and its implications in a large-scale production grid," in Proc. 2nd IEEE Int. Conf. Sci. Grid Comput., Dec. 2006, p. 27.
[20] Google. Mountain View, CA, USA, (2011).Google Cluster Data V2[Online]. Available: http://code.google.com/p/googleclusterdata/wiki/ ClusterData2011fi
[21] C. Reiss, J. Wilkes, and J. Helleirstein, Google Cluster-Usage Traces: Format C Schema. Mountain View, CA, USA: Google Inc., 2011.
[22] J. Dean and S. Ghemawat, ``MapReduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107fi113, Jan. 2008.
[23] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, et al., ``Hive: A warehousing solution over a map-reduce framework," in Proc. VLDB Endowment, vol. 2. 2009, pp. 1626fi1629.