Holistic Energy and Failure Aware Workload Scheduling in Cloud Datacenters

Xiang Li[a] (corresponding author), Xiaohong Jiang[a], Peter Garraghan[b], Zhaohui Wu[a]
[a] College of Computer Science, Zhejiang University, Hangzhou 310027, China. {lixiang2011, jiangxh, wzh}@zju.edu.cn
[b] School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, UK. p.garraghan@lancaster.ac.uk

ABSTRACT

The global uptake of Cloud computing has attracted increased interest within both academia and industry resulting in the formation of large-scale and complex distributed systems. This has led to increased failure occurrence within computing systems that induce substantial negative impact upon system performance and task reliability perceived by users. Such systems also consume vast quantities of power, resulting in significant operational costs perceived by providers. Virtualization – a commonly deployed technology within Cloud datacenters – can enable flexible scheduling of virtual machines to maximize system reliability and energy-efficiency. However, existing work address these two objectives separately, providing limited understanding towards studying the explicit trade-offs towards dependable and energy-efficient compute infrastructure. In this paper, we propose two failure-aware energy-efficient scheduling algorithms that exploit the holistic operational characteristics of the Cloud datacenter comprising the cooling unit, computing infrastructure and server failures. By comprehensively modeling the power and failure profiles of a Cloud datacenter, we propose workload scheduling algorithms *Ella-W* and *Ella-B*, capable of reducing cooling and compute energy while minimizing the impact of system failures. A novel and overall metric is proposed that combines energy efficiency and reliability to specify the performance of various algorithms. We evaluate our algorithms against *Random*, *MaxUtil*, *TASA*, *MTTE* and *OBFIT* under various system conditions of failure prediction accuracy and workload intensity. Evaluation results demonstrate that *Ella-W* can reduce energy usage by 29.5% and improve task completion rate by 3.6%, while *Ella-B* reduces energy usage by 32.7% with no degradation to task completion rate.

KEYWORDS

Energy Efficiency; Thermal Management; Reliability; Failures; Workload Scheduling; Cloud Computing

1. INTRODUCTION

Cloud datacenters are core infrastructure required to provision digital services globally, forming compute facilities composed by thousands of interconnected servers. Such systems consume vast amounts of energy to operate - the carbon emission of datacenters is comparable to that of Argentina, and continues to grow by 11% annually [1]. Datacenter energy usage can be categorized as predominantly stemming from computing (processors, storage, etc.) and cooling (air conditioner, fans, etc.) [2]. In order to reduce computing energy, Virtual Machines (VMs) are commonly scheduled to consolidate workload onto the fewest servers as possible, and then shutting down idle servers [3]. Such scheduling is a direct threat towards minimizing cooling energy, which is highly affected by skewed spatial temperature distribution of the facility (i.e. the hottest server). This requires careful workload balancing [4] in order to distribute workload evenly amongst servers to minimize the highest temperature, avoid hot spots and reduce cooling energy [5]. However, the aims of these two scheduling approaches results in a contradiction that attempts to simultaneously consolidate workload onto as fewest active servers, yet attempts to reduce the formation of system hotspots directly created by such consolidation. This does not accord with our intuition, as it is typically assumed that computing energy and cooling energy are positively correlated to each other: more computing energy results in additional cooling requirement for computing infrastructure heat rejection (and vice versa). However, this does not occur when considering the reduced cooling efficiency when lowering the Computer Room Air Conditioner (CRAC) supply air temperature to address hotspot formation.

It is imperative for Cloud providers to schedule workload to minimize total system energy of both computing and cooling to reduce operational costs. This must be achieved whilst adhering to implicit and explicit user requirements for high levels of application performance and reliability. The complexity and scale of modern day Cloud datacenters has resulted in failures becoming the normal rather than the exception [6], manifesting within both system software and hardware. Such failures cause Service Level Agreement (SLA) violation, resulting in energy [7] and monetary loss [8]. Work in [7] demonstrates the prevalence of software and hardware failures within production Cloud datacenters, with server Mean Time Between Failure (MTBF) as short as 6.5 hours [9]. [6] declares that the success of large-scale (e.g. petascale) computing will depend on its ability to provide dependable service at scale.

A challenge arises within workload scheduling as the objective to minimize total system energy does not result in optimal task reliability. Assume that S_1, S_2, S_3 correspond to scheduling solutions that produce the minimal computing energy, cooling energy, and highest reliability as shown in Equation (1).

$$\begin{aligned} E_{computing}(S_1) &= \min\left[E_{computing}(S)\right], \\ E_{cooling}(S_2) &= \min\left[E_{cooling}(S)\right], \\ E_{reliability}(S_3) &= \max\left[E_{reliability}(S)\right]. \end{aligned} \tag{1}$$

Each scheduling solution S will produce corresponding values as functions $E_{computing}(S)$, $E_{cooling}(S)$ and $E_{reliability}(S)$. A solution producing a localized optimal value does not necessarily achieve global optimization (i.e. $S_1 \neq S_2 \neq S_3$). In other words, uncoordinated scheduling algorithms that consider each solution as isolated optimization problems may not achieve the best result, and importantly are unable to ascertain an optimal trade-off between system reliability and energy-efficiency.

There exists limited work that addresses this challenge in the context of distributed systems (and by extension Cloud computing). [6] proposed failure-aware node selection strategies for VM scheduling however does not consider energy usage. [10][11] studied trade-offs between computing energy and system reliability to optimize energy-efficiency under the constraints of task deadlines, however omits cooling energy (which contributes 34% of total system energy [2]). Workload scheduling that unifies this objective is challenging due to the complexity of coordinating multiple solutions, as well as limited availability of holistic models capable of successfully capturing datacenter temperature distribution, energy profiles, failure patterns and workload characteristics. In this paper we propose two new failure-aware energy-efficient VM scheduling algorithms for Cloud datacenters. A particular strength of these algorithms are their ability to perform scheduling optimization for compute and cooling energy in conjunction with the failure characteristics of computing components. The major contributions of this work are summarized as follows.

- *Unified models for energy and failure-aware workload scheduling in Cloud datacenters*. Models derived from empirical findings from workload and failure characteristics based on production Cloud datacenter operation are integrated and implemented within an event-based Cloud simulator. This paper introduces a simplified Computational Fluid Dynamics (CFD) based temperature model to achieve the trade-off between temperature modeling accuracy and completion time.
- *Problem formulization of VM scheduling comprising computing energy, cooling energy and server reliability*. We introduce a criteria metric that captures the values of all three aspects to provide a standard means for specifying the performance of numerous scheduling algorithms.
- *Two failure-aware energy-efficient VM scheduling algorithms*. We propose *Ella-W* and *Ella-B* – algorithms that allocate VMs based on the defined unified criteria metric, and evaluate their effectiveness against five representative energy-aware and/or failure-aware scheduling algorithms. We conduct simulations under different failure prediction accuracies and workload intensities, demonstrating their ability to reduce total system energy and improve task reliability.

The remainder of this paper is organized as follows. Section 2 surveys the related work on energy-aware and failure-aware modeling and workload scheduling. Section 3 presents the datacenter model definitions. Section 4 specifies the problem statement and challenges. Section 5 details our holistic energy and failure aware scheduling algorithms. Section 6 describes model construction and Section 7 details the algorithm evaluation. Finally, Section 8 summarizes our findings and presents future research directions.

2. RELATED WORK

There exists numerous work in datacenter resource management and workload scheduling that consider thermal management, computing energy and/or server reliability. This section classifies these works into two categories: (i) modeling of distributed systems, including tracelog analysis of real systems, function fitting of actual measurement, thermal modeling via CFD techniques, integration of models into a simulation platform, and (ii) workload scheduling based on constructed models and simulation platform.

2.1. System Modeling

Cloud computing workload characteristics has been studied by Garraghan *et al* [12]. They present a large-scale analysis of workload resource utilization and a characterization of a Cloud datacenter using tracelogs made available by Google. [13] presented a detailed analysis of failure characteristic in large-scale datacenters, and determined that there exists a strong correlation between server failure occurrence and number of disks (identified as the most frequent component failure within a server). [7] conducted further research on modeling the reliability of Cloud datacenters, presenting an analysis of failure patterns and repair times of a large-scale production system. Their results demonstrated that failure characteristics for workload and servers are highly variable and that production Cloud workloads can be accurately modelled by function fitting (e.g. Gamma distribution), which can be used to predict future failures. [14] presented a survey of online failure prediction methods and classified them into four categories: failure tracking, symptom monitoring, detected error reporting and undetected error auditing.

Server power consumption is primarily driven by its utilization, and it is often modelled by a linear function or a segmented linear function [15][16]. The CRAC power is affected by the amount of heat rejection and its cooling efficiency (captured by *coefficient of performance*). [5][17] demonstrate that setting a higher supply air temperature (T_{sup}) can improve CRAC cooling efficiency. However the setting of T_{sup} is restricted by the server temperature distribution (e.g. hotspots) within the datacenter. Therefore, given the status of cooling infrastructure and thermal characteristics of a specific datacenter, obtaining the server temperature becomes a critical problem which has been intensively studied using CFD techniques [18][19][20][21][22]. Most research in this field seeks to reduce energy consumption and optimize datacenter operation at the hardware-level. For example, Durand-Estebe *et al*. [18] introduced a Proportional Integral Derivative (PID) algorithm to control the fan speed, combined with modeling servers and air conditioners. They consider the overall material electric consumption in datacenters, and select the appropriate cooling temperature set point to reach the best compromise between the chiller and the server energy consumption. Different datacenter airflow configurations were studied by Shrivastava *et al* [21], who compared different configurations and concluded that the "raised floor & ceiling return" configuration was the most effective schema. Studies show that datacenter racks with vertically placed servers attain enhanced cooling efficiency when adopting vertical cooling schema [22].

Construction of temperature models using CFD is extremely time-consuming and computationally expensive, therefore it is typically not applied to online (real-time) modeling. [23] proposed a sensor-based thermal evaluation model that predicts temperature distribution in a fast and accurate manner taking into account the recirculation characterization of a datacenter topology, and has been widely adopted [17][24][25]. When exploiting CFD techniques, it is necessary to model the process of heat transfer within a rack at fine granularity which increases model complexity. Additionally, their work assumes stable thermal status (e.g. airflow, server power consumption) in the datacenter. In order to capture the dynamics of the airflow, [4] proposed a computationally efficient multivariable model that dynamically captures the effects of the CRAC unit blower speed and supply air temperature on rack inlet temperatures. This model was used to investigate datacenter cooling system design and analysis such as CRAC units, load balancing, and hotspot detection. CFD and its derived methods are typically used to model the temperature of air entering racks with CPU temperature identified as the most important indicator for thermal management [26][27]. The Resistor-Capacitor (RC) model is the most established means to obtain CPU temperature adopted within the literature [28][29][30], alongside other CPU temperature modelling approaches [31][32][33][34]. In order to provide a generalized and extensible simulation framework for simulating Cloud computing infrastructure, Buyya *et al*. proposed CloudSim: an event-driven software package for Cloud datacenter simulation and evaluating resource management algorithms [15]. The latest version of CloudSim supports the simulation of almost all components within a datacenter and additional work exist which further enhances its functionality [35][36].

2.2. Workload Scheduling

We mainly focus on energy-aware and/or failure-aware workload scheduling policies designed for multi-node systems (e.g. clusters and datacenters) and introduce the representative works. Due to their inherit complexity, systems designers must consider a plethora of factors, including energy usage (servers, cooling), failures (servers, tasks, middleware, network), server temperature threshold and Quality of Service (QoS). Workload scheduling can be represented as an optimization problem (e.g. minimize energy usage, or maximize task completion rate) under specific constraints (imposed by an SLA).

Workload consolidation [37] is an efficient method to reduce computing energy by minimizing the number of active nodes [38]. Workload consolidation is commonly modeled as a bin packing problem, which has been proven to be NP-complete [39]. To address this problem, researchers tend to achieve the algorithm efficiency at the cost of reduced solution accuracy. Coffman *et al*. [40] proposed a simple and intuitive algorithm called *first-fit* decreasing, while Young *et al*. proposed *ECTC* and *MaxUtil* to consolidate workloads [37], with the former attempting to maximize duration of tasks running in parallel, and the latter maximizing average CPU utilization during execution.

Differing from the above works that solely considers workload consolidation to reduce computing energy, research conducted by Yoursri *et al*. [41] make substantial progress in combining computing energy reduction with CPU temperature control. Moore *et al*. [5] proposed a system-level solution to control heat generation through temperature-aware workload placement to reduce cooling energy. Tang *et al.* proposed the XInt algorithm [42] to achieve cooling efficiency via minimizing heat recirculation and peak inlet temperature. A proactive control approach is proposed in [43] that jointly optimizes the air conditioner compressor duty cycle and fan speed to reduce cooling cost and minimize risk of equipment damage due to overheating. Ayan *et al*. [17] proposed a hybrid method for coordinated cooling-aware job placement and dynamic cooling management. The algorithm allocates jobs to reduce the cooling demands of the CRACs, and then updates the CRAC thermostat settings based on temperature distribution models.

The above scheduling policies do not consider system failures. However, failure occurrences in these computing systems can induce substantial negative impact on system performance, deviating the system from our initial objectives [11], especially for those scenarios that must guarantee imposed constraints (e.g. a real time system may define strict task deadlines) [44]. To handle component failures within computing system, Song *et al*. defined a capacity–reliability metric to combine the effects of server capacity and reliability status during node selection [6]. They proposed Optimistic Best-Fit (*OBFIT*) and Pessimistic Best-Fit (*PBFIT*) algorithms to determine the best qualified servers on which to instantiate VMs to run user jobs. Experiments showed that the higher rate of successfully completed jobs was achieved by using *OBFIT* and *PBFIT* strategies. However, their scheduling algorithms do not consider the energy efficiency of a system. Altino *et al*. make further improvements [11] with their proposed algorithm leveraging proactive fault-tolerance techniques to deal with systems failures, and incorporating a metric termed power-efficiency to determine the best candidate server. Bahman [45] analyze energy and failure-aware workload scheduling at the level of hybrid/federated datacenters. They propose a scalable hybrid Cloud infrastructure as well as resource provisioning policies to assure QoS targets of the users, taking into account the workload model and failure correlation to redirect users' requests to the appropriate Cloud providers.

3. DATACENTER MODELING

This section details the datacenter models that our proposed algorithms are based on. Within a Cloud datacenter, users submit tasks deployed within VMs, and the scheduling system is responsible for VM placement, live migration and resource management. The datacenter dynamically adjusts the CRAC capacity in order to reduce cooling costs. It is possible to combine each of these models to describe datacenter operation holistically as illustrated in Fig. 1, and comprises four parts: (i) workload model, represented by

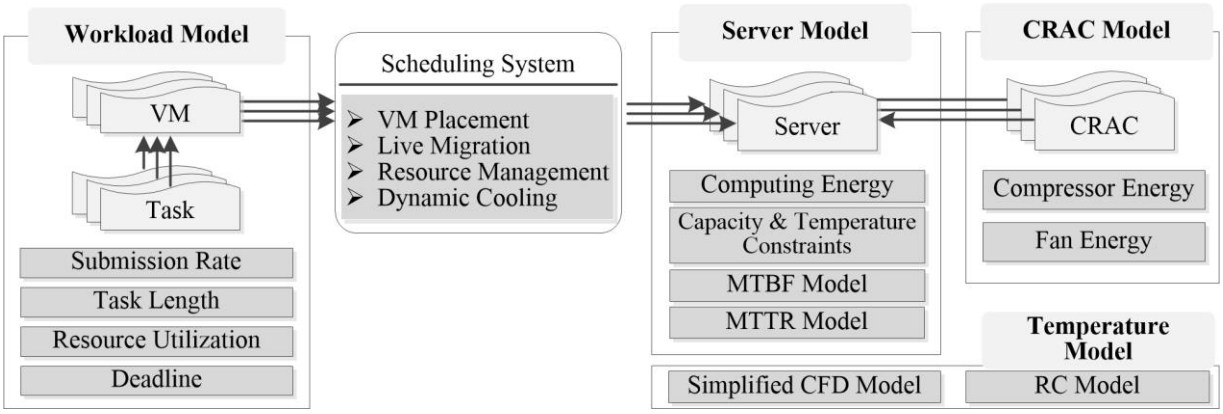


Fig. 1. Overall models of Cloud datacenter for workload scheduling

Symbol	Definition
E	Energy consumption [kWh].
k, p, a	Number of VMs submitted from users, number of PMs and CRACs in datacenter.
PM, VM, AC	Abbreviation of servers, virtual machines and air conditioners.
t	Time [s].
T	Temperature [K].
ϵ	Configuration / capacity of CPU, memory, etc.
w, u, d	Length [instructions], average CPU utilization and time deadline of a task.
P	Power consumption [W].
CoP	Coefficient of Performance, an index for evaluating CRAC performance.
Q_{AC}	Heat removed by CRACs [J].
C	(Specific) heat capacity [J/K, J/(kg*K)].
R	Thermal resistance [K/W].
T_{sup}	CRAC supply air temperature.
T_{in}	Rack inlet for cooling air temperature.
Ψ	Temperature raise caused by recirculation influence.

Table 1. Symbols and Definitions

submission rate, task length and resource utilization providing input to the scheduling system, (ii) server model, describing computing energy usage and failure patterns, (iii) CRAC model, profiling cooling energy usage (compressors and fans), and (iv) temperature model, that captures CPU temperature as a function of cooling capacity, server utilization and thermal characteristics.

3.1 Workload Model

Assume that a Cloud datacenter provides services using virtualization technology such as Xen, KVM and VMware. Tasks represent the most basic unit of computation, and are submitted by users through VM deployment. k is the total number of tasks submitted by users. We assume that each VM executes a single task. The request from users is represented as a sequence comprising k VMs, represented as $\boldsymbol{VM} = \{VM_1, VM_2, \ldots, VM_k\}$, where $k = |\boldsymbol{VM}|$. The capacity requirement of each VM is represented as $\epsilon_{VM\,i} = \{\epsilon_{core}, \epsilon_{memory}\}$ – a two-dimensional description considering the number of CPU cores (ϵ_{core}) and memory size (ϵ_{memory}). Each task deployed is captured by its workload length (w), average CPU utilization (u) and time deadline (d) (i.e. $task_i = \{w_i, u_i, d_i\}$). If a task fails during its execution (due to server failure), it will be re-submitted and re-executed within another available server [46]. A task completes its execution if it finishes prior to its deadline. Completion rate (η) is defined as the ratio of completed tasks with respect to the number of submitted tasks k [11]. Tasks are submitted according to submission rate λ, which is a function of time denoted by $\lambda = f_{submission}(t)$. The submission rate is assumed to be a continuous function. Therefore the total number of submitted tasks during time period $[t_1, t_2]$ is

$$k = |\boldsymbol{VM}| = \int_{t1}^{t2} f_{submission}(t)\, dt. \tag{2}$$

3.2. Server Model

The datacenter is composed of heterogeneous servers of various physical capacity, power profiles and temperature characteristics. The capacity of a physical server is represented by a fixed number of CPU cores, core processing speed and memory size. Assuming there are p servers within the datacenter, denoted by $\boldsymbol{PM} = \{PM_1, PM_2, \ldots, PM_p\}$, $|\boldsymbol{PM}| = p$. Server power consumption is primarily dependant on CPU utilization. The power model used within this work is based on the HP ProLiant ML110 G4 and G5 servers [15]. The server status is defined as *active* and *inactive,* with a server utilization greater than 0% indicating that a server is active and thus consumes power. An inactive server (i.e. sleep mode, turned off) results in power usage equal to 0. Power usage of i-th server is a

function of time t, denoted by $P_{PM\,i}(t)$, and its energy consumption is obtained by the integral of power usage on the time interval [t_1, t_2]. The summation of energy consumption of all servers is regarded as computing energy $E_{computing}$.

$$E_{computing} = \sum_{i=1}^{p} E_{PM\ i} = \sum_{i=1}^{p} \left(\int_{t_1}^{t_2} P_{PM\ i}(t)\, dt \right). \tag{3}$$

Within a Cloud datacenter, the VM scheduling solution is a map from ***VM*** to ***PM***. Let *Map* be the map and *Map*(VM_i) be the placement server of VM_i. The scheduling solution *Map* is subjected to constraints of critical temperature and capacity. This results in

$$\forall t \geq 0, \forall i \in [1, 2, ..., p] \quad T_{PM\ i}(t) \leq \overline{T}_{PM\ i} \text{ and } \sum C_{VMs\ \text{in}\ PM\ i} \leq C_{PM\ i}, \tag{4}$$

where $T_{PM\,i}(t)$ is the CPU temperature of i-th server, which is a function of time t. $\overline{T}_{PM\ i}$ is the critical temperature designed by the manufacturer which cannot be violated. Numerous works [17][23][24] consider the inlet air temperature as T_{PM}, while [27][29] consider the CPU temperature as T_{PM}. We argue that the inlet temperature is not a sufficient performance indicator, as the thermal management target is predominantly dictated by the CPU temperature [29][30]. As a result, this paper focuses on CPU temperature as the analysis objective. $\sum C_{VMs\ in\ PM\ i}$ represents the summation of the required capacity of VMs within i-th server, the physical capacity of which is denoted by $C_{PM\ i}$. It is possible for servers within the datacenter to experience failures resulting in executing VMs to terminate which are then re-submitted by the system scheduler onto other servers for re-execution. In this context, we define a failure as a crash-stop caused by a hardware or software error that makes compute nodes unavailable [47]. When a failure occurs, all VMs within the node will provide incorrect service (i.e. no response) [6][11]. The server failure model is captured by the distribution of the Mean Time Between Failure (MTBF). After a failure event occurs, a period of time is required until recovery, during which the server is unavailable. The length of time period to repair this server is captured by a distribution of Mean Time to Repair (MTTR).

3.3. CRAC Model

Typically, cold air supplied by the CRACs enters each rack through the inlet and flows out from the rear, removing the heat generated by computing servers as hot air. Fans are deployed within each rack in order to accelerate the airflow and heat transfer. The space between two inlet sides is known as the cold aisle, while the space between two outlet sides is known as the hot aisle [27][43]. Hot air is exhausted through ceiling return vents. Assuming the datacenter comprises a CRACs, represented by $\boldsymbol{AC} = \{AC_1, AC_2, ..., AC_a\}$, $|\boldsymbol{AC}| = a$. We consider the CRACs as the only cooling devices within the facility since it accounts for the majority of datacenter cooling energy [2]. The energy consumed by each CRAC during time interval [t_1, t_2] comprises compressor (i.e. a component of the CRAC, responsible for air compression) energy and CRAC fan energy [43].

$$E_{cooling} = E_{compressor} + E_{fan}. \tag{5}$$

The power consumption of a CRAC fan unit is proportional to the cubic of its rotation speed, and its energy consumption is obtained by time integral of the power. The energy consumed by the compressor is given by

$$E_{compressor} = Q_{\boldsymbol{AC}} / CoP\left(T_{sup}\right), \tag{6}$$

where Q_{AC} is the heat removed by the CRAC, which can be modelled according to server energy [24] or the heat disparity between datacenter out flow and CRAC supply air [43]. *CoP* is the Coefficient of Performance, denoted by $CoP(T_{sup})$ since it is a function of supply air temperature T_{sup}. A higher *CoP* indicates a more efficient process, requiring less work to remove a constant amount of heat. Research has demonstrated a positive correlation between *CoP* and supply air temperature [5][43] shown as follows:

$$CoP = 0.0068 \times T_{sup}^2 + 0.008 \times T_{sup} + 0.458. \tag{7}$$

Equation (7) implies that we can improve cooling efficiency by maximizing T_{sup} to reduce cooling cost. However, T_{sup} must be set low enough to maintain the CPU temperature under its critical threshold. In order to understand the relationship between CPU temperature and T_{sup}, we introduce our proposed temperature model.

3.4. Temperature Model

Our objective is to obtain the CPU temperature with the knowledge of CRAC settings, datacenter layout, server thermal characteristics and workload distribution. In order to provide support for online scheduling, the developed temperature model must be capable of performing online calculation within a timely manner. This paper proposes a two-step temperature model based on CFD model and RC model. This includes modeling (i) rack inlet temperature considering both CRAC status and datacenter layout, and (ii) CPU temperature when factoring server thermal characteristics and its respective workloads. For step 1, rack inlet temperature can be modelled as the summation of CRAC supply air temperature and datacenter recirculation [17][23][24].

$$\boldsymbol{T_{in}} = \boldsymbol{T_{sup}} + \boldsymbol{\Psi}. \tag{8}$$

$\boldsymbol{T_{in}}$, $\boldsymbol{T_{sup}}$ and $\boldsymbol{\Psi}$ are $r \times 1$-vectors representing the inlet temperature, nearest CRAC supply temperature and recirculation influence,

respectively (with r denoting the number of racks). The value of $\boldsymbol{\Psi}$ is identified in various scenarios, affected by rack power consumption since recirculation influence is produced by hot air from each rack. Without loss of generality, we assume that the outlet temperatures are identical amongst each rack within a certain range, representing different levels of server power usage and workload intensity. For each workload intensity within range, we monitor air temperature before entering each rack (i.e. $\boldsymbol{T_{in}}$, rack inlet temperature) to determine $\boldsymbol{\Psi}$ in Equation (8).

After obtaining the rack inlet temperature of each rack in Step 1. CPU temperature models are constructed in Step 2, representing the most important indicator of thermal management. The RC model is the most established means to obtain CPU temperature [29][30], modelled as

$$T_{cpu}(t) = PR + T_{in} + (T_0 - PR - T_{in}) \times e^{-\frac{t}{RC}}, \tag{9}$$

where R and C is thermal resistance and heat capacity of the server, respectively. We assume that all servers within a rack share the same inlet temperature. As described in Section 3.2, the power usage P for a given server is predominantly determined by its utilization. That is, P is affected by its hosted workloads (VMs and tasks). T_{cpu}, P and T_{in} here are not vectors and are instead values of a single server. It is noticeable that CPU temperature T_{cpu} is function of time t, from an initial temperature of T_0 and approaches (PR+T_{in}).

4. Problem Formulization and Challenges

4.1. Problem Formulization

The problem of failure-aware and energy-efficient workload scheduling targeted in this paper is formulated as follows.

$$\begin{aligned}
&\text{Given } \boldsymbol{PM},\ \boldsymbol{AC} \text{ and } \boldsymbol{VM},\\
&\text{find scheduling } Map,\\
&\text{that minimizes overall metric } H = \left(E_{computing} + E_{cooling}\right) \Big/ |\boldsymbol{VM}| + \omega(1-\eta),\\
&\text{subjected to } \forall t \geq 0, \forall i \in [1, 2, ..., p],\\
&\qquad T_{PM\ i}(t) \leq \overline{T}_{PM\ i} \text{ and } \sum \epsilon_{VMs \text{ in } PM\ i} \leq \epsilon_{PM\ i}.
\end{aligned} \tag{10}$$

Given a set of servers $\boldsymbol{PM}$, CRACs $\boldsymbol{AC}$ within the Cloud datacenter and requested VMs $\boldsymbol{VM}$ from users, the target is to find a scheduling solution represented by Map from $\boldsymbol{VM}$ to $\boldsymbol{PM}$ towards minimizing the overall metric H which considers computing energy $E_{computing}$, cooling energy $E_{cooling}$ and task completion rate η. The scheduling is subjected to constraints of critical temperature and server capacity. In order to apply the metric within the context of different workload intensities, we divide the energy usage by the number of requested VMs ($k = |\boldsymbol{VM}|$). The former term of the overall metric ($E_{computing} + E_{cooling}$) / $|\boldsymbol{VM}|$ is termed the *energy metric*, representing the energy efficiency of the scheduling, and the latter term $\omega(1 - \eta)$ is termed the *reliability metric*, indicating system reliability. We consider completion rate η as a critical indicator of the reliability and $1 - \eta$ is the ratio of failed tasks. The parameter ω is the weight configurable by the datacenter administrator, calculated before term (1-η) for two reasons: (i) *energy metric* and *reliability metric* have different units and massive disparity in value range - parameter ω is used for adjustment in units and values, and (ii) parameter ω provides flexibility in scheduling targets demanded by datacenter administrators (i.e. a larger ω value indicates that high task completion rate is prioritized over energy-efficiency of the entire system).

4.2 Challenges

Due to datacenter complexity and a multitude of interacting subsystems, most researchers perform evaluation for proposed approaches within simulation environments. For workload and server failure models, the main challenge is the limited availability of analysis data derived from real-world systems. To date there exists few works which accurately model workload and server failures [7][12][13]. CRAC and temperature models that traditionally use CFD has been intensively studied, however are not applicable for online scheduling due to significant computation completion time. Work such as [23][25] can provide rapid temperature distribution but are unable to capture server CPU temperature dynamicity. Furthermore, the models constructed should be implemented within a Cloud computing simulation platform in order to conduct research on algorithm design, implementation and evaluation.

Another challenge is the effective coordination of optimizing energy-efficiency and reliability. Total energy comprises computing energy and cooling energy. However, a contradiction exists as computing energy oriented strategies tend to consolidate workloads, while cooling energy oriented strategies attempts to balance workload for effective thermal distribution. Furthermore, energy-efficiency and reliability rate are two metrics proposed from different perspectives. Combination of such metrics becomes challenging since a highly energy-efficient scheduling solution could potentially results in decreased reliability, and vice versa. For example, energy-aware scheduling algorithm tends to allocate more workloads on energy-efficient servers, resulting in an increased probability of failure due to higher workload intensity [48][49]. Furthermore, characteristics of energy, workload and failure patterns can vary significantly within different datacenter environments. Therefore, proposed algorithms must be robust and flexible enough to handle different operational scenarios and conditions.

Additionally, comparisons between different algorithms in terms of energy usage and task reliability remains difficult due to a

lack of standard metrics to specify performance. [11] proposes a metric called working-efficiency calculated by multiplying a power efficiency metric with completion rate. However, this power efficiency metric does not explicitly consider the server energy usage. Furthermore, each datacenter administrator will pursue different objectives. For example, an administrator may seek to reduce energy expenditure at the risk of increasing task failures, while other systems could focus on increasing task completion rates in adherence to a specified deadline to avoid late-timing failures. Therefore, the metric should be defined with flexibility, allowing adjustment in evaluation of different systems.

5. TOTAL ENERGY AND FAILURE AWARE SCHEDULING

When scheduling VMs, it is important to consider the providers desire to enhance energy-efficiency, and user demand for reliable service. Intuitively, a datacenter with greater task duration and shorter MTBF is more likely to be affected by system failures. Such datacenters impose significant impact on task performance as server failures can interrupt tasks during their execution. Furthermore, failures result in waste in terms of resource usage and electrical energy [7]. Failure characteristics appear to vary considerably within different datacenter systems. For example, the MTBF in Los Alamos National Laboratory (LANL) HPC clusters system is 608 days [6], which is more than 40x of the MTBF in Google datacenter [7]. The average task execution time in LANL is 11 hours [6], being 6x than that of Google datacenter [12]. Therefore, a well-designed algorithm has to be heuristic to handle the diversity of task characteristic and system reliability.

In our proposed method, a server is selected in a greedy manner. We define an evaluation function comprising indexes of energy and reliability. For a submitted VM, the server with lowest value of the following function will be selected for VM placement. In order to meet the constraints of critical temperature and server capacity shown in Equation (10), the candidate servers are confined to those with (i) sufficient capacity to host the VM, and (ii) will not violate CPU critical temperature under CRAC maintenance (CRAC can dynamically increase its capacity when necessary).

$$\varepsilon_{overall\ (i,j)} = \alpha \times \varepsilon_{energy\ (i,j)} + (1-\alpha) \times \varepsilon_{reliability\ (i,j)}, \tag{11}$$

where $\varepsilon_{energy\ (i,j)}$ and $\varepsilon_{reliability\ (i,j)}$ is termed *energy index* and *reliability index* of PM_j for allocation of VM_i, respectively. Our algorithm holistically handles the characteristics with respect to server energy efficiency and reliability, and attempts to achieve an optimal trade-off between them. For example, if servers within the datacenter are frequently failing, the algorithm should be sensitive to server failure to avoid task failure and deadline violation. We achieve this by assigning different $\varepsilon_{reliability\ (i,j)}$ to each server and the latter term in Equation (11) (*reliability index*) will dominate the overall evaluation function $\varepsilon_{overall\ (i,j)}$. Another benefit of our approach is that parameter α is adjustable to achieve different trade-off points, as each datacenter provider will pursue different design objectives and business requirements. Generally, a smaller α indicates that the algorithm is more reliability biased, and greater α indicates more energy-efficiency biased.

5.1. Energy Index Definition

The *energy index* ε_{energy} is defined as the increase in terms of total power consumption post VM allocation, or rather, the server with the least power increment after VM placement is regarded as the most energy-efficient candidate. This increment is obtained as follows: The power consumption prior to VM placement is the summation of power usage of all servers and CRACs termed *total power usage* (P_{total}), given by

$$P_{total} = \sum_{i=1}^{i=p} P_{PM\ i} + \sum_{i=1}^{i=a} P_{AC\ i}. \tag{12}$$

For any submitted VM_i, it is necessary to select a server for its placement. Assume that VM_i is allocated to PM_j, and the CPU utilization of server j after allocation is $u_{PM\ j}$'. The power cost of PM_j is calculated with an updated CPU utilization. CPU temperature is predicted using temperature models detailed Section 3.4 for the next time interval (e.g. 5 minutes), denoted by $T_{PM\ j}$'. If $T_{PM\ j}$' is lower than its critical temperature, the CRACs will not be adjusted. Otherwise, the cooling capacity of the nearest CRAC is gradually increased by decreasing its supply air temperature T_{sup}. For example, consider a scenario where the T_{sup} of a CRAC is 20℃ which decreases its T_{sup} by its minimum division value (e.g. 1℃) step by step. This decrease will continue until it reaches its lowest possible setting, or $T_{PM\ j}$' is lower than the critical temperature. If $T_{PM\ j}$' is greater than the critical temperature under the lowest possible T_{sup} setting, the energy index is defined as infinite to avoid temperature violation. Equation (12) is then reapplied with the updated servers and CRACs to obtain the power consumption after placing VM_i to PM_j, denoted by $P_{total\ (i,j)}$'. The energy index is therefore defined as

$$\varepsilon_{overall\ (i,j)} = P_{total\ (i,j)}' - P_{total}, \tag{13}$$

in which P_{total} and $P_{total\ (i,j)}$' are the total power cost before and after the placement of VM_i, respectively. A smaller value for $\varepsilon_{energy\ (i,j)}$ indicates less energy usage. The definition of ε_{energy} inherently considers energy draw of servers and cooling infrastructure, and can be used to calculate the total energy-efficiency of the entire datacenter.

5.2. Reliability Index Definition

We introduce two strategies *worst-fit* and *best-fit* to define the *reliability index*. The basic concept is to avoid server failures during task execution to increase task reliability. For *worst-fit* strategy, the *reliability index* for allocating VM_i to PM_j is as follows.

$$\varepsilon_{reliability\ (i,j)} = \begin{cases} \dfrac{h_i}{\delta_j} = \dfrac{w_i}{c_j \times u_i \times \delta_j} & \text{if } \delta_j \geq h_i \text{ and } d_i \geq h_i \\ infinity & \text{otherwise} \end{cases} \tag{14}$$

where δ_j is the predicted failure time of PM_j, and h_i is the duration of VM_i (i.e. *task i*, since each VM runs a single task). The duration h_i is defined as the ratio of task length w_i to utilized capacity (which is the product of server capacity c_j and average utilization of task *i* u_i), with d_i representing the deadline of task *i*. A server reflected by a smaller *reliability index* $\varepsilon_{reliability\ (i,j)}$ is more likely to be selected for placement. The *reliability index* is set to infinity to avoid selection when (i) the predicted server failure δ_j precedes the task duration h_i, meaning the task will fail before its completion, and (ii) the task deadline precedes the task duration, indicating that the task cannot finish prior to its deadline due to insufficient server capacity. One problem with *worst-fit* strategy is that it may assign short-running tasks to reliable and powerful servers and result in other long-running tasks unable to find suitable nodes to meet deadlines [6]. To handle this problem, *best-fit* redefines the *reliability index* as follows.

$$\varepsilon_{reliability\ (i,j)} = \begin{cases} \dfrac{\delta_j - h_i}{\delta_j} = 1 - \dfrac{w_i}{c_j \times u_i \times \delta_j} & \text{if } \delta_j \geq h_i \text{ and } d_i \geq h_i \\ infinity & \text{otherwise} \end{cases} \tag{15}$$

Similar to the reliability index definition of *worst-fit* shown in Equation (14), it attempts to guarantee that task *i* can complete execution without a failure by confining the candidates only to servers that $\delta_j \geq h_i$ and $d_j \geq h_i$.

5.3. Failure-aware Energy-efficient Algorithms

We combine the *energy index* and *reliability index* in order to form the corresponding proposed algorithms *Energy and reLiabiLity-Aware Worst-fit* (*Ella-W*) and *Best-fit* (*Ella-B*), respectively. In addition to energy efficiency, *Ella-W* attempts to select the most reliable and powerful server (smaller $\varepsilon_{reliability}$) when placing a VM to ensure task completion. Such placement does not necessarily result in the formation of a hotspot that consumes additional cooling energy, as the evaluation function in Equation (11) factors the energy metric into consideration, thus including cooling energy. Similarly, [11] proposes a different metric to implement the worst-fit strategy, called *reliability weight* when deciding which server to run VM_i. If the use of dynamic migration within the system is omitted, the *reliability weight* (R_{weight}) is defined as

$$R_{weight\ (i,j)} = \varepsilon_{reliability\ (i,j)} = 1 / 2^{d_i - \delta_j}. \tag{16}$$

In comparison to Equation (16), *Ella-W* determines its *reliability index* differently in two ways. First, task execution duration h_i is adopted instead of task deadline d_i, as the former parameter inherently considers server capacity. Secondly, the failure time of many servers can be much greater than d_i or h_i, especially for the datacenter that is comparatively reliable. The exponential characteristic makes R_{weight} of many servers close to 0 and decreases the significance of *reliability index* shown in Equation (16).

According to [6], best-fit strategies are relatively sensitive to the failure prediction accuracy as the server selected by *Ella-B* has "just sufficient" time execution to VM_i. Therefore the effectiveness of best-fit failure-aware scheduling is strongly dependant on the failure prediction accuracy. While there exists various definitions for prediction accuracy [14][46], this paper selects the definition adopted in [6][50] as it applies effectively within the context of our design evaluation.

5.4. Baseline Algorithms

To better illustrate the effectiveness of proposed algorithms *Ella-B* and *Ella-W*, we evaluate them against five state-of-the-art and representative scheduling algorithms. This includes (i) *Random* selection, (ii) *MaxUtil* [37] which allocates workload to servers with the maximum average utilization during execution, (iii) *TASA* (*Thermal Aware Scheduling Algorithm*) [27] which allocates workloads to the server with lowest CPU temperature, (iv) *OBFIT* (*capacity and reliability-aware Optimistic Best-FIT*) [6] that jointly considers server capacity and the reliability, and (v) *MTTE* (*Minimum Time Task Execution*) [11] that considers both computing energy efficiency and reliability. It is worth noting that the constraints of server capacity and critical temperature shown in Equation (10) apply to all selected algorithms.

- *Random selection*: to satisfy the constraints of capacity and critical temperature shown in Equation (10), we randomly select a server to instantiate the arriving VM. This is a very basic method for workload scheduling, as it does not consider reliability status of servers, and cannot guarantee that tasks will complete execution prior to the desired deadline.
- *MaxUtil*: this algorithm aims to consolidate workloads onto as fewest servers as possible to reduce computing energy. For a given task *i*, *MaxUtil* checks every server from the first rack to the last, and selects the server with the highest function value of $f_{i,j} = \sum_{t=1}^{h_i} u_j(t) / h_i$, where h_i is the processing time of task *i*, and $u_j(t)$ is the utilization of PM_j as a function of time *t*. The

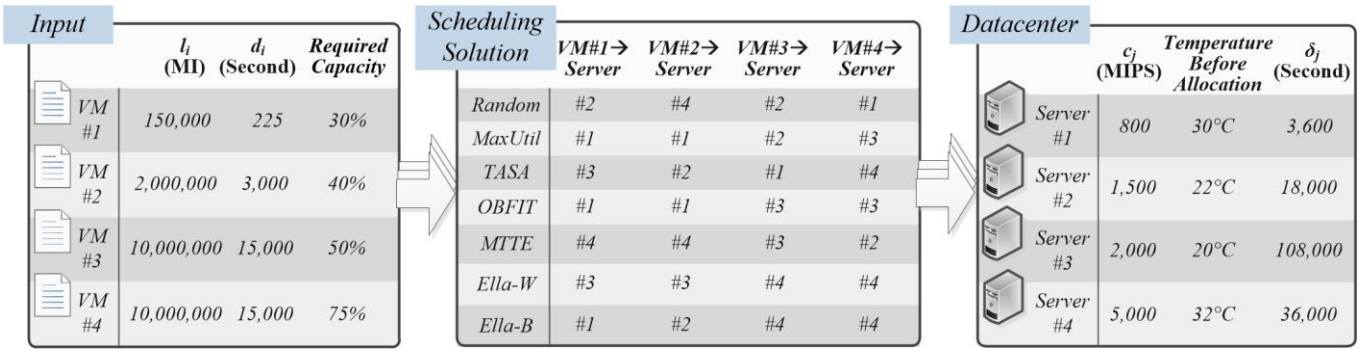


Fig. 2. A case study of VM scheduling with different algorithms

function value $f_{i,j}$ captures the average utilization during the execution of task i on PM_j.

- *TASA*: workload is scheduled uniformly to minimize the maximum temperature - a common approach to reduce the cooling capacity requirement and achieve high cooling efficiency. The main concept of *TASA* is to (i) schedule the task that will impose the largest temperature profile prior to the task with the lower temperature profile, and (ii) select the server with the lowest temperature as allocation target. In our experiment, the real-time scheduling system immediately allocates the submitted task to fulfil specified QoS. Therefore, we simplify *TASA* by allocating each task to the server with lowest CPU temperature without considering the task-temperature profile.
- *OBFIT*: consider a set of candidate nodes $S = \{PM_j \mid \delta_j \geq d_i \text{ and } c_j \geq w_i / d_i\}$. This algorithm selects the server with the lowest *capacity-reliability* metric shown as $cr = \alpha(c_j - w_i/d_i) / (w_i/d_i) + (1 - \alpha)(\delta_j - d_i) / d_i$, where α is an adjustable weight put on the two factors in making selection decisions. As α increases and approaches 1, *cr* becomes more capacity biased. If α approaches 0, *cr* is more reliability biased.
- *MTTE*: for the servers with sufficient capacity to host the arriving VM_i, *MTTE* estimates their power efficiencies according to $Eff = u_i (p_1 + p_2) / (p_1 + p_2 \times u_i)$, where p_1 and p_2 are parameters capturing the linearity of server power consumption as a function of server utilization u_i. That is, power consumption of PM_i is $p_1 + p_2 \times u_i$. If power efficiency is equal for a set of servers, the algorithm chooses the server that returns the lowest value of *reliability weight* as shown in Equation (16).

5.5 Case Study

In order to demonstrate the operation of these algorithms, we present a simple case study to illustrate the process of server selection for each algorithm: *Random*, *MaxUtil*, *TASA*, *OBFIT*, *MTTE*, *Ella-W* and *Ella-B*. The server characteristics for computing power and thermal are assumed to be identical. At time $t = 0$, the incoming request sequence comprises 4 VMs, and each VM runs a single task. Fig. 2 presents the scheduling solution produced by each algorithm. As described in Section 3.2, a scheduling solution is a map from VMs to servers. The middle scheduling solution table within Fig. 2 illustrates the placement of each VM.

Random algorithm selects servers randomly amongst all servers under the capacity and temperature constraints shown in Equation (10). In this context, the required capacity of a VM is defined as the product of VM resource configuration and the task utilization. It is presented as the percentage regarding to server capacity, therefore the summation of *required capacity* of VMs within a server is not greater than 1. *MaxUtil* checks each server from the first to the last and selects the server with the maximum average utilization during task execution. *TASA* selects the server with the minimum CPU temperature (i.e. coolest server), therefore allocates VM_1 to PM_3 (20°C). After this placement, the predicted CPU temperature is updated with Equation (9) and then selects a server for VM_2, and so forth. *OBFIT* selects servers with the minimum *cr* metric. *MTTE* places VMs according to *Eff* metric. If *Eff* metric returns identical values, it allocates the VM to the server with minimum R_{weight} shown Equation (16). *Ella-W* and *Ella-B* make decisions based on the trade-off of reliability, cooling energy and computing energy. For *Ella-W*, we take the allocation of VM_1 as an example: PM_3 has the minimum (best) $\varepsilon_{reliability}$ of 0.023 according to Equation (14) and will become the allocation target server, assuming that the allocation will not cause critical temperature violation and additional cooling energy usage. Note that the decision making of *OBFIT*, *Ella-W*, and *Ella-B* is affected by the selection of trade-off parameter α, which is configured to 0.01, 0.5 and 0.5, respectively in this case study.

6. Model Construction

The complexity of Cloud computing, along with the increasing demand for energy-efficient IT technologies drives the necessity to conduct repeatable and controllable evaluation of algorithms prior to actual development of Cloud systems. A suitable alternative of real experiments is the use of simulation which enables the evaluation of hypothesis prior to software development. We use the CloudSim framework (Version 3.0.3) [15] that provides a generalized and extensible simulation enabling seamless modeling of emerging Cloud infrastructures. We implement the proposed *Ella-W*, *Ella-B* and other baseline algorithms into CloudSim to evaluate their performance in terms of energy-efficiency and task completion rate. In order to make our simulation more convincing, our temperature model is derived from CFD techniques. Furthermore, we use realistic datacenter operational conditions to define our models, such as workload distribution and failure patterns derived from the Google datacenter tracelog. In this section, we first present

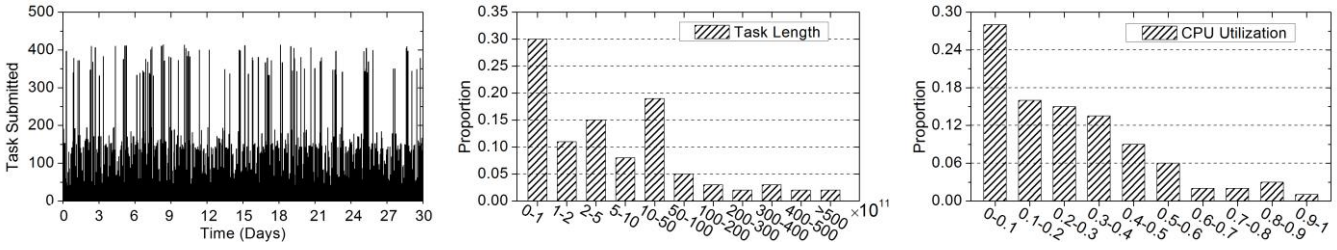


Fig. 3. Generated workload: (a) submission rate, (b) task length (instructions) distribution and (c) CPU utilization distribution

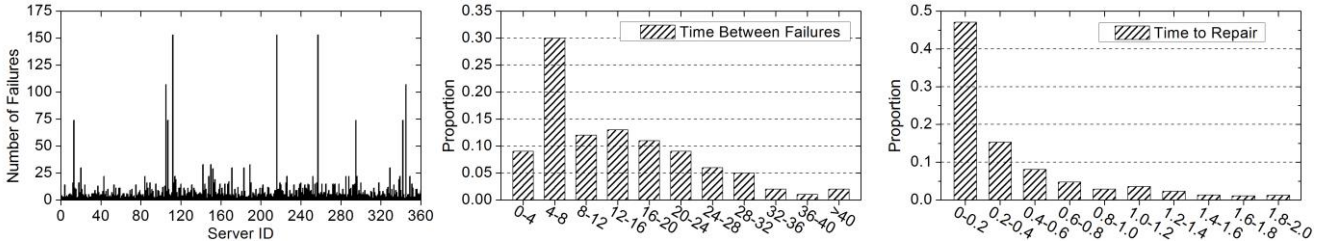


Fig. 4. Reliability pattern: (a) number of failures of all 360 servers, (b) distribution of time (hours) between two failures, (c) distribution of time (hours) to repair

the models of workload, failure and temperature. Note that these models are constructed offline and integrated into CloudSim for experiments.

6.1. Workload Generation

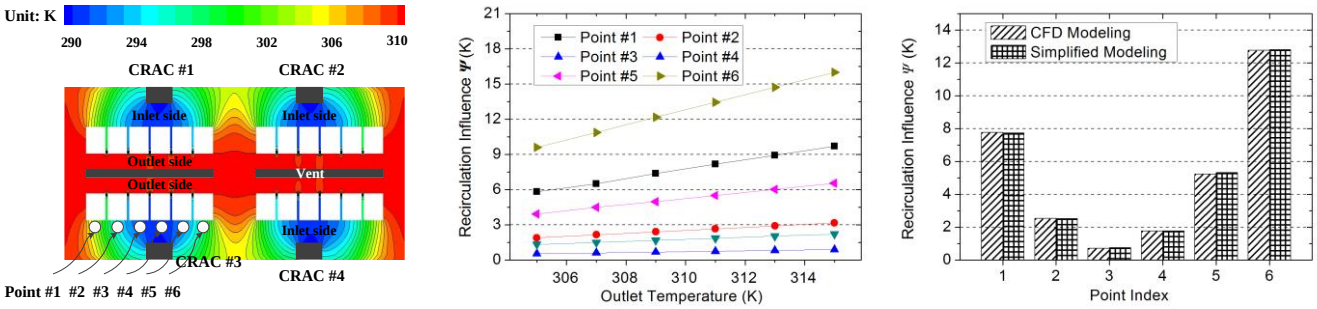
Cloud computing systems are required to fulfil diverse business objectives that result in workload diversity in terms of submission rate, task length and resource utilization patterns. This paper adopts the workload generated by previous work in [12], which presents a comprehensive analysis of the workload characteristics derived from Google datacenter that features approximately 25 million tasks. We model the submission characteristic of tasks through profiling the submission rate hourly. During each hour, we assume the submission rate follows a random distribution. Within [12], all users are classified into six clusters, and for our work we use the proportion of each cluster as the weight. The overall submission rate follows the distribution of the weighted summation of these clusters. Tasks are modelled in terms of length and CPU utilization with fitting functions including Lognormal and Loglogistic distributions. In order to capture the patterns of different workloads, we further linearly scale the parameters such as submission rate and CPU utilization. Fig. 3(a) shows the produced submission rate (hourly) in 30 days with a total number of 100,000. Fig. 3(b) illustrates the modelled distribution of task length. It is noticeable that the length of most (83%) tasks is between $0 \sim 50 \times 10^{11}$ instructions. Fig. 3(c) presents the distribution of produced task CPU utilization. It is observed that CPU utilization of the majority (72.5%) of tasks is under 40%.

Each task is associated with a completion deadline. Tasks that complete prior to the deadline are regarded as "completed tasks". To generate a deadline for each task, we utilize the approach adopted in [44][45]. First, a *baseline duration* is defined for each task. It is calculated as the ratio of task length to average capacity of servers within the datacenter. Completion deadline is defined as $(1 + x) \times$ *baseline duration*, where x is termed stringency factor, representing a degree of difficulty in meeting the deadlines. A larger x indicates more relax deadline. For the presented evaluation, the value of x is set to 1.2.

6.2. Failure Models

Our failure model is derived from the Google datacenter tracelog that features 12,532 servers spanning 29 days. Within the tracelog observational period, 8,954 server failures occurred within 5,056 servers with an average of 308 servers failing daily. This paper considers models of MTBF and MTTR. The former describes the frequency of server failures, and the latter illustrates the time servers take to recover from failures and re-join the system when possible. [7] classifies all servers appeared in the tracelog into 5 main categories. We generate the overall distribution according to their populations and the distribution of each category. Our modelled datacenter comprises 360 servers. Fig. 4(a) shows the number of failures of all servers within 30 days produced by our failure model. We can observe that a small proportion of servers exhibit high failure occurrence, and that most servers experience significantly less failure events. Fig. 4(b) presents the distribution of the MTBF, and the average value is 13.98 days. From Fig. 4(c), it is observable that the majority (78.1%) of servers can be repaired within an hour.

In the real world, a reliability-aware scheduling system makes decision based on predictions of the next failure [14], which differs from the actual failures. Correspondingly, it is necessary to model *actual failures* and *predicted failures* in simulation simultaneously. In this paper, the model described above is utilized to generate *actual failures*. For each server, once an actual failure is generated, it is assumed to be fixed for this server as failures demonstrate high spatial locality [14] (e.g. a server that fails frequently is highly likely to fail within the coming time period). Meanwhile, *predicted failures* are produced according to specific prediction accuracy. Similarly,



(a) (b) (c)

Fig. 5. Temperature model: (a) temperature contour in a stable status, (b) profiling recirculation influence $\boldsymbol{\Psi}$ for Points #1-6 under different outlet temperatures, (c) model accuracy when outlet temperature is set to 310K

the production of actual and predicted values for repair time adopts the same methodology. We generate synthetic traces of failure/repair predictions with different accuracy. Specifically, we adopt the method in [6] that

$$prediction\ error = \frac{actual\ time - predicted\ time}{actual\ time} \times 100\%,$$
$$prediction\ accuracy = 100\% - error. \tag{17}$$

Here *actual time* and *predicted time* are the occurrence time of actual failure and predicted failure. In [6], authors consider cases in which the predicted occurrence time of a future failure precedes the actual occurrence time. We remove this constraint since *predicted time* can be either less or greater than *actual time* in reality. When *predicted time* >= *actual time*, we define

$$prediction\ error = \frac{predicted\ time - actual\ time}{actual\ time} \times 100\%. \tag{18}$$

This means that *predicted time* can be obtained with the knowledge of prediction accuracy and actual failure generated. In this study, in half the case *predicted time* precedes after *actual time*.

6.3. Rack Inlet Temperature

In Section 3.4, we introduced our temperature model, which comprises two components: rack inlet temperature modeling and CPU temperature modeling. This section presents parameter identification for the former component, since CPU temperature modeling has been intensively studied [27][30]. The rack inlet temperature is described as the summation of CRAC supply air temperature and datacenter recirculation. However, the recirculation influence $\boldsymbol{\Psi}$ still remains unknown and requires identifying. $\boldsymbol{\Psi}$ is an r×1-vector - in order to identify $\boldsymbol{\Psi}$, we model a datacenter using CFD techniques. The size of our modelled datacenter is 15.8×6.5m^2, comprising 4 CRACs, 2 vents and 24 rack ×15 servers each rack (totalling 360 servers). The model is 2-dimensional and built using Gridgen, a complete meshing toolkit used to generate high quality grids for complex geometries in engineering. Fig. 5(a) presents the temperature contour within the datacenter in a stable status, with the outlet temperature and supply air temperature being 310K and 290K, respectively. The figure is obtained using Fluent, a well-known commercial CFD package. It is observable that the racks near a CRAC exhibit lower inlet temperature and recirculation influence, indicating higher cooling efficiency.

We assume that the outlet temperatures are identical amongst each rack and ranges in E = [305K, 315K], to represent different workload intensities. For each of the outlet temperatures within range E, we monitor air temperature before entering each rack (inlet temperature) in stable status, and compute corresponding $\boldsymbol{T_{in}}$, to determine $\boldsymbol{\Psi}$ in Equation (8). Using this method, we analyze $\boldsymbol{\Psi}$ for Point #1-6 (illustrated in Fig. 5(a)) under various outlet temperatures within range E. These points are sufficient for capturing the inlet temperatures of all the racks since the datacenter layout is symmetrical. Fig. 5(b) presents the profile of $\boldsymbol{\Psi}$ of each point under different outlet temperatures. We fit each profile with a linear function via regression. Since we represent different workload intensities in the datacenter via outlet temperature, in practical terms it is necessary to map the workload intensities proportionally into E to determine the corresponding $\boldsymbol{\Psi}$. Fig. 5(c) shows a case study to demonstrate the practicality of our model. In this case, we set the outlet temperature and the initial inlet temperature to 310K. This figure presents the result of $\boldsymbol{\Psi}$ produced in the above linear regression and CFD modeling, with an average error of 1.6%, showing high accuracy of the simplified model.

7. EVALUATION

The evaluated datacenter configuration comprises 360 servers cooled by 4 CRACs. Servers are heterogeneous in terms of capacity, with each server configured as 2-8 Xeon3040 or Xeon3075 CPU cores with 1860-2660 MIPS. The heat capacity and thermal resistance of servers in Equation (9) are set to 340 J/K, 0.34 K/W, respectively [29]. CPU critical temperature is configured to 70°C. All the models and corresponding parameters are integrated into CloudSim V3.03.

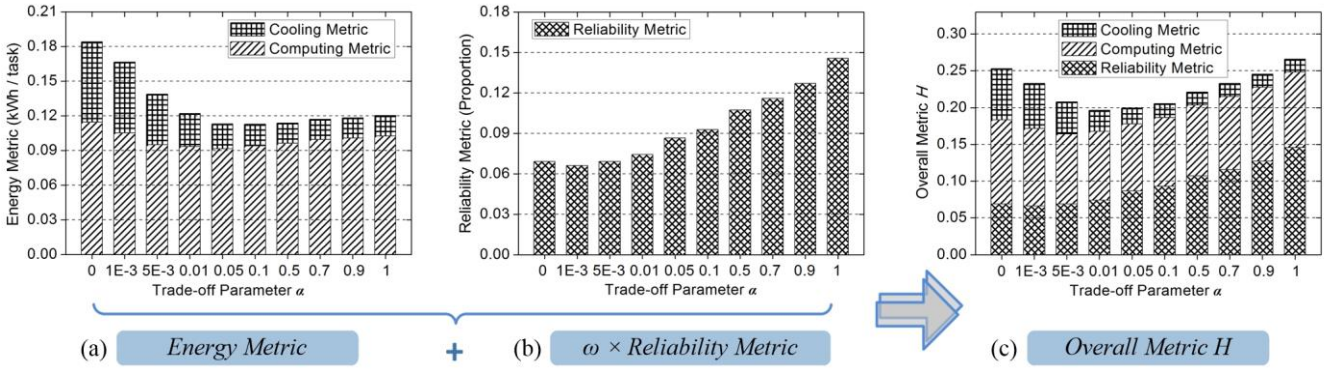


Fig. 6. Selection of trade-off parameter α for *Ella-W*. The overall metric is weighted summation of *energy metric* and *reliability metric*. The weight ω is set to 1.0.

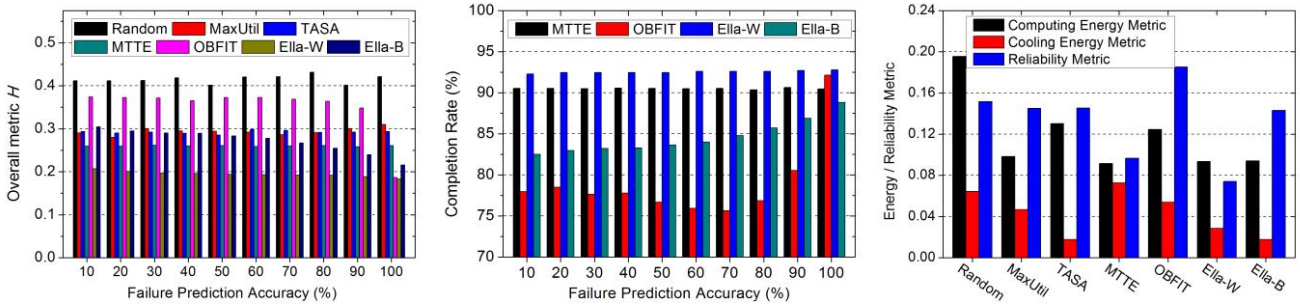


Fig. 7. Performance of different scheduling algorithms: (a) overall metric under different prediction accuracy, (b) completion rate of failure-aware algorithms (*Ella-W*, *Ella-B*, *MTTE*, *OBFIT*), (c) energy & reliability metric. The prediction accuracy is set to 80%.

As described in Section 5 (Equation (11)), *Ella-W* and *Ella-B* make decisions based on the trade-off between energy efficiency and reliability. First, we identify the trade-off parameter α in *Ella-W* and *Ella-B* through experiments to determine the optimal point. The target of this paper is formulated as Equation (10) in Section 4.1. The *overall metric H* is the summation of two components, *energy metric* $(E_{computing} + E_{cooling}) / |\boldsymbol{VM}|$ and *reliability metric* $\omega(1 - \eta)$. Parameter ω is set to 1.0 in our experiments. Next we present the methodology to identify the optimal trade-off point α in *Ella-W* and *Ella-B*, followed by the experimental results. We evaluate the performance with respect to the overall metric *H* proposed in section 4.1, including energy efficiency and reliability.

7.1 Trade-off Parameter Identification

Fig. 6 presents the identification of trade-off parameter α for *Ella-W*. From Fig. 6(a) we observe a sharp decrease in cooling energy when increasing α. This is due to *Ella-W* being cooling-aware thus saves cooling energy significantly with a greater α. Generally, smaller α indicates greater reliability biased and a higher α indicates greater energy-efficiency bias as described in Section 5. However, counterexamples exist where the system consumes slightly more computing energy when α becomes greater than 0.01. This is due to the energy biased (greater α) algorithm increasing the failure probability, leading to additional computing energy waste due to failed tasks (i.e. failed tasks are re-submitted and re-executed). Fig. 6(b) shows *reliability metric* when different α is configured. *Reliability metric* is obtained as $1 - \eta$, where η is the completion rate of all user tasks. Note that higher *reliability metric* indicates lower system reliability. It is observed that reliability biased algorithm can reduce failure occurrence significantly. For example, the *reliability metric* at $\alpha = 0$ is 0.069, indicating a task completion rate of 1 - 0.069 = 0.931, better than $\alpha = 1$ (0.146). The overall metric is the weighted summation ($\omega = 1.0$) of *energy metric* and *reliability metric*. We find that *Ella-W* provides the minimum overall metric (i.e. best performance) when $\alpha = 0.01$. Similarly, we conduct experiments to analyze the best α setting in *Ella-B*, results show that best performance occurs when $\alpha = 0.5$. In the rest of this section, we take the above values as the default setting for *Ella-W* and *Ella-B*.

7.2 Algorithm Comparison

After identifying the trade-off parameter in *Ella-W* and *Ella-B*, we compare them against other algorithms. Evaluations are conducted in a time period of 30 days with 100,000 tasks submitted. Fig. 7(a) shows the overall metric *H* as a function of failure prediction accuracy. We observe that *Ella-W* performs the best amongst all algorithms, 38.2% less than other algorithms in average with respect to the overall metric *H*. *MaxUtil*, *TASA* and *Random* are not failure-aware, and thus failure prediction accuracy makes no difference to their performance. However, the performance of failure-aware algorithms, especially best-fit algorithms (*Ella-B*, *OBFIT*), improves their performance due to an increase of failure prediction accuracy, which is in accordance with findings in [6]. This is because failure prediction accuracy affects the task completion rate. From Fig. 7(b) we observe that the improvement of failure

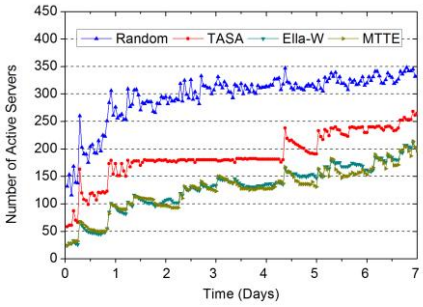
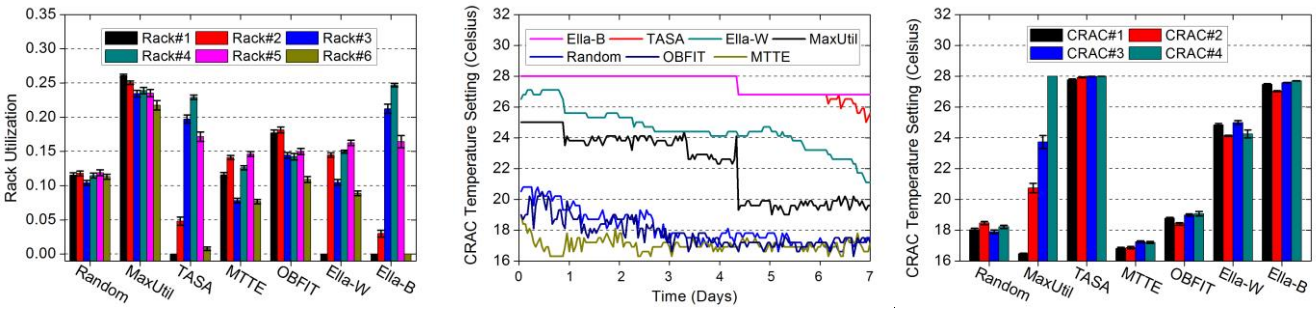


Fig. 8. Number of active servers with different scheduling algorithms



(a) (b) (c)

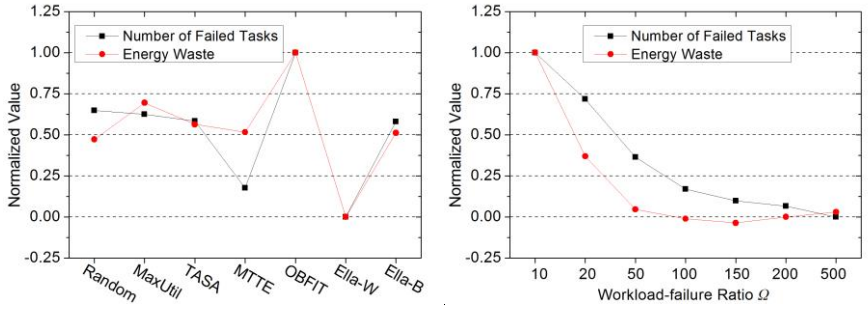
Fig. 9. Details in cooling performance of different scheduling algorithms: (a) utilization of Rack #1 - #6, (b) CRAC temperature setting in the first 7 days, (c) CRAC temperature setting of four CRACs.

prediction accuracy causes significant increase of task completion rate of *Ella-B* and *OBFIT*. This is due to best-fit algorithms are relatively sensitive to prediction accuracy.

In order to analyze the details of each algorithm, we present the metrics of computing energy, cooling energy and reliability in Fig. 7(c) with 80% prediction accuracy. For computing energy, *MaxUtil*, *MTTE*, *Ella-W* and *Ella-B* perform effectively (0.094 kWh/task in average) in comparison with other algorithms. Fig. 8 illustrates the number of active servers within first 7 days of 4 representative algorithms (*Random*, *TASA*, *Ella-W* and *MTTE*). Results show that *Random* algorithm utilizes the most servers (295.8 in average), followed by *TASA* (188.8 in average), which attempts to balance temperature distribution with the purpose of reducing cooling capacity and energy draw. A straightforward conclusion can be drawn that computing energy is roughly proportional to the number of active servers used. The basic idea of TASA is to allocate more workloads to servers with better cooling efficiency and activate more servers to balance workloads and avoid hotspots. That is, the cost of reducing cooling energy for *TASA* is using more servers leading to more computing energy. *Ella-W* and *MTTE* perform best in terms of computing energy, using 130.8 and 127.6 servers in average, consuming 2140KWh and 2098KWh of electricity, respectively. They both tend to allocate VMs to servers with high utilization to consolidate workloads and are reluctant to start idle servers.

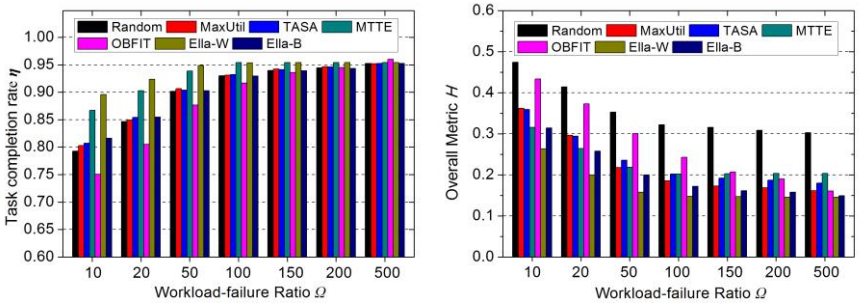
For cooling energy, results in Fig. 7(c) show that thermal-aware algorithms *TASA*, *Ella-W* and *Ella-B* consume 64.2% less cooling energy than the other non-thermal-aware algorithms *Random*, *MaxUtil*, *MTTE* and *OBFIT*. Cooling energy is affected by removed volume of heat produced by CRACs and corresponding *CoP*. Higher CRAC temperature setting leads to smaller *CoP*, indicating better cooling efficiency. Thermal-aware algorithms allocate workloads allowing for maximized CRAC temperature setting. In order to analyze this result, we further demonstrate the workload distribution (with standard error of mean, SEM) of each rack in Fig. 9(a). The indicator *rack utilization* is defined as the average utilization of all servers within a rack. We present the *rack utilization* of Rack #1 - #6, from which we can learn the workload distribution of all the 24 racks since our modelled datacenter is symmetric. For *Random* algorithm, rack utilization follows a uniform distribution as any server has the same probability being selected. *MaxUtil* selects servers in ascending order (i.e. from the rack to the last), making workloads consolidate more within the first rack and follow a roughly descended distribution. *MTTE* and *OBFIT* do not show any obvious features since their allocation is affected by the server reliability that is unrelated to rack location. However, temperature-aware algorithms *TASA*, *Ella-W* and *Ella*-B allocate more workloads to Racks #2 - #5, and less workloads to the bilateral racks (i.e. Rack#1 and #6). This is because the middle racks are closer to the CRAC (datacenter layout is illustrated in Fig. 5(a)), leading to higher cooling efficiency compared with bilateral racks. This results in thermal-aware algorithms performing better within the context of enhanced cooling efficiency.

Fig. 9(b) further illustrates the CRAC temperature setting during the first 7 days with different scheduling algorithms. Since there are four CRACs within the modelled datacenter, the results are presented in terms of average CRAC temperature setting of all CRACs. As described in Section 3.3, a higher temperature setting indicates better cooling efficiency and less cooling energy draw. It is noticeable that *TASA* performs best as it focuses on thermal management only. Both *Ella-W* and *Ella-B* show satisfactory performance. However since the trade-off parameter α of *Ella-B* (0.5) is greater than that of *Ella-W* (0.01), *Ella-B* is more energy biased, leading to better cooling performance in comparison with *Ella-W*. Although *MaxUtil* is not cooling-aware, it consolidates workloads to fewer



(a) (b)

Fig. 10. Energy waste (normalized value) by failed tasks with (a) different algorithms, (b) *Ella-B* under different workload-failure ratio Ω.



(a) (b)

Fig. 11. (a) Task completion rate η and (b) the overall performance of algorithms under different *workload-failure ratio* Ω.

servers (e.g. as shown in Fig. 9(a), rack utilization follows a descended distribution), causing the first CRAC to become over-utilized while underutilizing the remaining CRACs. This is demonstrated further by results shown in Fig. 9(c), which presents the temperature settings of each rack (average value over a time span of 30 days). We observe that the temperature setting of each CRAC under *MaxUtil* follows an ascending distribution, indicating a large deviation in CRAC cooling capacity. The temperature settings for thermal-aware algorithms *TASA*, *Ella-W* and *Ella-B* are significantly greater than the rest, which is consistent with the results shown in Fig. 9(b).

The above experiments are conducted in specific workloads and failure patterns. To comprehensively evaluate the performance of algorithms under different conditions, we conduct experiments with various workload intensities and failure frequencies. A parameter called *workload-failure ratio*, denoted by Ω is proposed to capture the impact of system reliability to overall performance. Ω is defined as the ratio of average time span between two failures (*MTBF*) to average workload length ($\overline{W}$) to represented as $\Omega = MTBF/\overline{W}$. Lower Ω indicates relatively longer task length and higher failure frequency, resulting in task execution that is more likely to be affected by server failures during task execution. In order to achieve different values of Ω, workload is fixed (amount of resource utilization and task length) and failure frequency is scaled linearly, making Ω range in 10 - 500. Prediction accuracy is set to 80% in the following evaluations.

On one hand, smaller Ω results in more task failures and worse *reliability metric*. This can be captured by task completion rate η as shown in Fig. 11(a). It is noticeable that *Ella-W* produces the highest completion rate of 94.1%, which is higher than the baseline algorithms by 3.6% in average. However *Ella-B* exhibits relatively poor completion rate (90.5%), which is roughly equivalent to the performance of the other baseline algorithms. Furthermore, a smaller Ω results in additional task re-submissions and more energy usage. Garraghan *et al*. analysed and quantified the explicit impact of failures within a system in terms of energy costs [7]. However, their evaluation did not consider the energy impact to cooling facilities. Therefore this paper presents the impact of task failures in terms of total energy waste. Two sets of parallel experiments are conducted with Ω = 20. Server failure is disabled (no failures occur) for one set and enabled for the other. Results produced from two sets are compared, and the energy difference between two sets is regarded as energy waste caused by task failures. Fig. 10(a) shows the number of failed tasks and the corresponding energy waste with

	Metric	*Random*	*MaxUtil*	*TASA*	*MTTE*	*OBFIT*	Average
***Ella-W* (%)**	*Energy Metric*	56.2	12.2	20.5	30.8	27.9	29.5
	Reliability Metric	32.5	30.5	30.3	8.3	36.1	27.5
	Overall Metric	52.0±1.2	20.8±3.0	26.2±1.8	25.7±1.5	33.3±4.8	31.6
***Ella-B* (%)**	*Energy Metric*	58.4	15.9	24.2	34.1	31.0	32.7
	Reliability Metric	-1.8	-1.2	-1.5	-17.6	10.0	-2.4
	Overall Metric	44.3±2.2	8.8±1.1	14.8±0.6	13.7±3.5	23.8±3.3	21.1

Table 2. Reduction of metrics by *Ella-W* and *Ella-B* in comparison with other algorithms.

different algorithms. Fig. 10(b) shows the corresponding energy waste by failed tasks of *Ella-B* under different Ω. The conclusion we draw from Fig. 10(a) and (b) is that wasted energy is roughly proportional to the number of task failures.

As a consequence, the impact of system reliability to the *overall metric* is two-fold, including the impact to *reliability metric* and *energy metric*. Generally, the overall metric decreases as Ω (workload-failure ratio) increases. This trend is illustrated in Fig. 11(b), which presents the overall metric of all algorithms under different Ω. It is observed that *Ella-W* and *Ella-B* exhibits the best performance under various conditions. Table 2 further shows the results of *Ella-W* and *Ella-B* in comparison with the rest of algorithms. The data is presented in terms of average reduced percentage of metrics under different Ω. *Ella-W* can reduce overall metric by 20.8% - 52.0% (31.6% in average) and *Ella-B* can reduce by 8.8% - 44% (21.1% in average). Specifically, *Ella-W* reduces 29.5% of energy usage and increases 27.5% of the *reliability metric* within the datacenter. In comparison, *Ella-B* reduces 32.7% of energy usage, however performs poorly in *reliability metric*: only 2.4% greater the average value of compared algorithms.

8. Conclusions and Future Work

This paper proposes two failure-aware energy-efficient algorithms (*Ella-W* and *Ella-B)* for workload scheduling within Cloud datacenters. This was achieved by capturing and modeling the behavioral characteristics of datacenter workload, server, CRAC and temperature. A novel performance metric is proposed to comprehensively evaluate algorithms in terms of energy consumption and reliability. To our knowledge, this is the first paper that captures computing energy, cooling energy and server failures in a holistic manner for scheduling decisions. Algorithm effectiveness was evaluated through simulation derived from realistic operational behavior against various energy-aware and/or failure-aware algorithms. Experiment results demonstrate that *Ella-W* and *Ella-B* reduces energy usage and improve system reliability significantly. From the observations and evaluation results presented in this paper, we draw the following conclusions.

- *Worst-fit failure-aware algorithm performs better than best-fit failure-aware algorithms in terms of the reliability*. Two factors cause poor performance for best-fit algorithms: (i) they tend to allocate VMs to a server whose upcoming failure time is close to the VM completion time, therefore they are sensitive to failure prediction accuracy, and (ii) even though the prediction is set to relatively high accuracy (e.g. 80%), the worst-fit algorithm *Ella-W* still shows better performance (lower overall metric) under different *workload-failure ratio* Ω, as worst-fit algorithms tend to allocate VMs to the most reliable servers, leading to increased reliability.
- *Algorithms that consider multiple parameters are more effective towards improving overall system operation*. Amongst all algorithms studied within this paper, *Ella-W* and *Ella-B* produce the best result since it considers cooling energy, computing energy and failures holistically for decision making. Algorithms that randomly select servers without considering these factors together result in worst performance. Furthermore, adjusting the trade-off parameter α of *Ella-W* to neither 0 (reliability-aware only) nor 1 (energy-aware only) achieves the best result overall. Therefore, workload scheduling within Cloud datacenter has to take into account various factors and perform elaborate trade-offs.

Future work includes more holistic modeling and further improvement of workload scheduling performance. Specifically, the augmentation of modeling can be implemented from three perspectives: (i) inclusion of additional cooling components, such as water cooling devices and in-case fans, (ii) more find-grained CFD modeling, including 3-dimensional models that considers the height of datacenters and racks, and (iii) the impact of server temperature on system reliability, which is omitted within this paper. This entails exploration of the relationship between server temperature and failure patterns. Additionally, we plan to further explore scheduling algorithms with heuristic methods such as genetic algorithms and ant colony algorithms to enhance the system productivity. The method we currently use is a greedy-based server selection, which cannot guarantee the production of optimal result.

Acknowledgment

This work is supported by National High Technology Research 863 Major Program of China (No.2011AA01A207), National Science Foundation of China (No.61272128) and China Scholarship Council (CSC) program. This work is partly supported by the European Commission under H2020-ICT-20152 TANGO project.

References

[1] Jonathan K, Growth in datacenter electricity use 2005 to 2010. Oakland, USA, 2011, Analytics Press. http://www.analyticspress.com/datacenters.html.
[2] Barroso L A, Clidaras J, Hölzle U. The datacenter as a computer: an introduction to the design of warehouse-scale machines, Synthesis lectures on computer architecture, 2013, 8(3): 1-154.
[3] Ferreto T C, Netto M A S, Calheiros R N, et al. Server consolidation with migration control for virtualized datacenters, Future Generation Computer Systems, 2011, 27(8): 1027-1034.
[4] Zhou R, Wang Z, Bash C E, et al. Datacenter cooling management and analysis-a model based approach, Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2012: 98-103.
[5] Moore J, Chase J, Ranganathan P, et al. Making scheduling "cool": temperature-aware workload Placement in Data Centers, Conference on Usenix Technical Conference. 2005: 61-75.
[6] Fu S. Failure-aware resource management for high-availability computing clusters with distributed virtual machines, Journal of Parallel and Distributed Computing, 2010, 70(4): 384-393.
[7] Garraghan P, , et al., An analysis of failure-related energy waste in a large-scale Cloud environment, IEEE Transactions on Emerging Topics in Computing, 2014(2): 166-180.
[8] Vishwanath K V, Nagappan N. Characterizing cloud computing hardware reliability, Proceedings of the ACM symposium on Cloud computing, 2010: 193-204.
[9] Hsu C, Feng W, A power-aware run-time system for high-performance computing, Proceedings of ACM/IEEE Conference on Supercomputing, 2005: 1-9.
[10] Aupy G, Benoit A, Robert Y. Energy-aware scheduling under reliability and makespan constraints. International Conference on High Performance Computing,

2012: 1-10.
[11] Sampaio A M, Barbosa J G. Dynamic power-and failure-aware cloud resources allocation for sets of independent tasks, IEEE International Conference on Cloud Engineering (IC2E), 2013: 1-10.
[12] Moreno I S, Garraghan P, Townend P, et al. Analysis, modeling and simulation of workload patterns in a large-scale utility cloud, IEEE Transactions on Cloud Computing, 2014, 2(2): 208-221.
[13] Vishwanath K V, Nagappan N. Characterizing cloud computing hardware reliability, ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June. 2010:193-204.
[14] Salfner F, Lenk M, Malek M. A survey of online failure prediction methods, ACM Computing Surveys, 2010, 42(3): 1283-1310.
[15] Anton Beloglazov, and Rajkumar Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers, Concurrency and Computation: Practice and Experience (CCPE), 24(13): 1397-1420.
[16] Khosravi A, Garg S K, Buyya R. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers, Euro-Par 2013 Parallel Processing, Springer Berlin Heidelberg, 2013: 317-328.
[17] Banerjee A, Mukherjee T, Varsamopoulos G, et al. Cooling-aware and thermal-aware workload placement for green HPC data centers, Proceedings of Green Computing Conference, 2010: 245-256.
[18] Durand-Estebe B, et al. Data center optimization using PID regulation in CFD simulations, Energy and Buildings, 2013(66): 154-164.
[19] Radmehr A, Noll B, Fitzpatrick J, et al. CFD modeling of an existing raised-floor data center, Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), 2013: 39-44.
[20] Summers J, Kapur N, Thompson H. Design of data centre rack arrangements using open source software, Semiconductor Thermal Measurement and Management Symposium, 2013: 45-51.
[21] Shrivastava S, Sammakia B, Schmidt R, et al. Comparative analysis of different data center airflow management configurations, Proceedings of IPACK2005: 329-336.
[22] Songa Z, Zhangb X, Erikssona C. Data center energy and cost saving evaluation, International Conference on Applied Energy, 2015: 1255-1260.
[23] Tang Q, Mukherjee T, Gupta S K S, et al. Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters, International Conference on Intelligent Sensing and Information, 2006: 203-208.
[24] Pakbaznia E, Ghasemazar M, Pedram M. Temperature-aware dynamic resource provisioning in a power-optimized datacenter, Conference on Design, Automation and Test in Europe, 2010: 124-129.
[25] Tang Q, Gupta S K S, Varsamopoulos G. Thermal-aware task scheduling for data centers through minimizing heat recirculation, International Conference on Cluster Computing, 2007: 129-138.
[26] Vanderster D C, Baniasadi A, Dimopoulos N J. Exploiting task temperature profiling in temperature-aware task scheduling for computational clusters, Advances in Computer Systems Architecture, 2007: 175-185.
[27] Wang L, Khan S U, Dayal J. Thermal aware workload placement with task-temperature profiles in a data center, The Journal of Supercomputing, 2012, vol. 61(3): 780-803.
[28] Skadron K, Abdelzaher T, Stan M R. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management, HPCA, 2002: 17-28
[29] Mhedheb Y, Jrad F, Tao J, et al. Load and thermal-aware VM scheduling on the Cloud, International Conference on Algorithms and Architectures for Parallel Processing, 2013: 101-114.
[30] Zhang S, Chatha K S. Approximation algorithm for the temperature-aware scheduling problem, International Conference on Computer-aided Design, 2007: 281-288.
[31] Z Wu, X Li, P Garraghan, X Jiang, K Ye, Virtual machine level temperature profiling and prediction in Cloud datacenters, International Conference on Distributed Computing Systems, 2016 (in press).
[32] Ramos L, Bianchini R. C-Oracle: Predictive thermal management for data centers, International Symposium on High Performance Computer Architecture, 2008: 111-122.
[33] Hirokaw T, Hond K, Shibuy T. Mercury and freon: temperature emulation and management for server systems, ACM SIGARCH Computing Architecture News, 2006, 34(5): 106–116.
[34] Vanderster D C, Baniasadi A, Dimopoulos N J. Exploiting task temperature profiling in temperature-aware task scheduling for computational clusters. Advances in Computer Systems Architecture, 2007: 175-185.
[35] Li X, Jiang X, Huang P, et al. DartCSim: An enhanced user-friendly cloud simulation system based on CloudSim with better performance. International Conference on Cloud Computing and Intelligent Systems, 2012: 392-396.
[36] Li X, Jiang X, Ye K, et al. DartCSim+: enhanced CloudSim with the power and network models integrated. International Conference on Cloud Computing, 2013: 644-651.
[37] Lee Y C, et al. Energy efficient utilization of resources in cloud computing system, The Journal of Supercomputing, 2012, 60(2): 268-280.
[38] Ye Ke-Jiang, et al, Power management of virtualized cloud computing platform, Chinese Journal of Computers, 2012, 35(6): 1262-1285.
[39] Li X, et al. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center, Mathematical and Computer Modelling, 2013, 58(5): 1222-1235.
[40] Coffman et al. Approximate solutions to bin packing problems. WOE-29, Institut FR Mathematik B, TU Graz, Steyrergasse 30, A-8010. 2002.
[41] Mhedheb Y, Jrad F, Tao J, et al. Load and thermal-aware VM scheduling on the cloud, Algorithms and Architectures for Parallel Processing. Springer International Publishing, 2013: 101-114.
[42] Tang Q, Gupta S, Varsamopoulos G. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach, IEEE Transactions on Parallel and Distributed Systems, 2008, 19(11): 1458-1472.
[43] Lee E, et al. Proactive thermal management in green datacenters, The Journal of Supercomputing, 2012, 60(2): 165-195.
[44] Islam M, Balaji P, Sadayappan P, et al. QoPS: a QoS based scheme for parallel job scheduling. Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, 2010: 252-268.
[45] Javadi B, Abawajy J, Buyya R. Failure-aware resource provisioning for hybrid Cloud infrastructure. Journal of Parallel and Distributed Computing, 2012, 72(10): 1318-1331.
[46] Yu Z, Wang C, Shi W. FLAW: FaiLure-Aware Workflow scheduling in high performance computing systems. Journal of Cluster Computing, Kluwer Academic Publishers Hingham, MA, USA, 2010, 13(4): 421-434.
[47] Avižienis A, Laprie J C, Randell B, et al. Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 2004, 1(1): 11-33.
[48] Fadishei H, Saadatfar H, Deldari H, Job failure prediction in grid environment based on workload characteristics, Proceedings of 14th International Computer Conference, 2009: 329-334.
[49] Iyer R, Rossetti D, Effect of system workload on operating system reliability: a study on IBM 3081, IEEE Transactions on Software Engineering, 1985, 11(12): 1438-1448.
[50] Fu S. Exploring event correlation for failure prediction in coalitions of clusters. Conference on Supercomputing, 2007:1-12.

Xiang Li is a PhD student in the Department of Computer Science and Technology at Zhejiang University since 2011. He received his BSc degree from Liaoning University in 2011. He visited the School of Computing, University of Leeds as a visiting scholar in 2015. His research interests focus on Cloud computing, energy-aware workload scheduling and datacenter modeling. He is a student member of IEEE.

Xiaohong Jiang received her B.Sc. and M.Sc. degree in computer science from Nanjing University and the Ph.D. degree in Zhejiang University, China, in 2003. She is an associate professor at the Department of Computer Science, Zhejiang University. Her research focuses on distributed systems, virtual environment, cloud computing, and data service.

Peter Garraghan is a Lecturer in the School of Computing & Communications, Lancaster University. He has industrial experience building large-scale systems and his research interests include distributed systems, Cloud datacenters, dependability, and energy-efficient computing.

Zhaohui Wu is the President of Zhejiang University, He is a chief scientist in the 973 Project, an information expert in the 863 Project and the team leader of the National Panel of the Modern Service Industry. He is the winner of the first-tier talent in the National Talents Program. His major research focuses on service computing, Cloud computing and ubiquitous computing.