International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:3, 2016

The Application of Bayesian Heuristic for Scheduling in Real-Time Private Clouds

Sahar Sohrabi

***Abstract*—**The emergence of Cloud data centers has revolutionized the IT industry. Private Clouds in specific provide Cloud services for certain group of customers/businesses. In a real-time private Cloud each task that is given to the system has a deadline that desirably should not be violated. Scheduling tasks in a real-time private CLoud determine the way available resources in the system are shared among incoming tasks. The aim of the scheduling policy is to optimize the system outcome which for a real-time private Cloud can include: energy consumption, deadline violation, execution time and the number of host switches. Different scheduling policies can be used for scheduling. Each lead to a sub-optimal outcome in a certain settings of the system. A Bayesian Scheduling strategy is proposed for scheduling to further improve the system outcome. The Bayesian strategy showed to outperform all selected policies. It also has the flexibility in dealing with complex pattern of incoming task and has the ability to adapt.

***Keywords*—**Bayesian, cloud computing, real-time private cloud, scheduling.

I. Introduction

THE propagation of Cloud data centers has radically changed the IT industry by providing services for businesses and individuals on Pay-as-you-go basis. Private Clouds in particular aim to serve the internal units in a business or tasks from similar businesses. Private Clouds, in comparison to public Clouds, have less resources available. Private Clouds are especially complicated to study when they have deadline restrictions on tasks. Such private Clouds are real-time. The aim in these Cloud systems is to serve the tasks within an acceptable time frame. However, a violation of deadline will not cause a meltdown in the system. The goal is to prevent the deadline violation where possible.

To serve the incoming tasks within the time frame scheduling policies are needed to determine the way available resources are dedicated to each task. The dynamic nature of the private Clouds and the uncertainty in future resource requests makes optimal scheduling a complicated job, especially when there is not an accurate information about the resource requirements in hand. Although there are examples of scheduling policies proposed for distributed systems, they usually rely on the existence of knowledge about resource requirements [1] or an approximated value of it [2], [3]. But such information is not always available to the scheduler. Moreover, differences in the values presumed by scheduler and the real resource requirements can potentially change the outcome for the policies relying on them. Therefore, there is a need for having scheduling policies that are capable of handling the scheduling in a real-time private Cloud without relying on the information on resource requirements. The earlier part of our research [4] showed that the applicable scheduling policies result in a substantially different outcome for the system according to the circumstances such as task arrival rate. A policy that outperforms all the others in a pressurized system falls short in a relaxed setting and the other way around.

Sahar Sohrabi is a PhD student at Swinburne University of Technology, Melbourne, Australia (e-mail: ssohrabi@swin.edu.au).

The sub-optimality of scheduling policies brought up the need for an adaptive scheduling mechanism that switches between scheduling policies according to the system's current status in order to get closer to an optimal outcome for the system. The system outcome can be measure as a combination of some criteria including energy consumption, deadline violation, execution time and the number of host switches.

Energy consumption is the major contributor in the cost of running a Cloud system. Deadline violation is also important because of the system's real-time aspect. Execution time provides insight into the system's behavior regardless of the deadlines. Host switches add to the energy consumption and execution time, and potentially the deadline violation, due to the start up energy peak and time delay. An accumulated measure of these criteria can be used to compare the performance of the scheduling policies in a real-time private Cloud.

An adaptive scheduling policy that leads to a more desirable measure needs to be adaptive and respond to the changes in the system status. It is required to change and learn based on the observed results. Bayesian heuristic has shown its potential in dealing with complicated systems. So, a Bayesian based mechanism that switches between the available scheduling policies is expected to result in objective measures that are closer to optimal measures.

The remainder is organized to include a short review of the related literature in Section II. Section III describes the way system status is defined and the scheduling policies for scheduling tasks in a real-time private Cloud. It also covers the criteria for system outcome and the constraints in the system. Then the Bayesian heuristic for scheduling in a real-time private Cloud is explained in Section IV. A set of experiments are carried out in Section V and the results are illustrated in Section VI. It is then followed by the conclusion and directions for future work in Section VII.

II. Related Work

Scheduling policies, in general, determine the way available resources are shared among tasks and their corresponding Virtual Machines (VM). The impact of scheduling policies on the results of provisioning in a private Cloud is tangible

due to its relatively limited resources. When the private Cloud has the limitation of being a real-time system, this impact is further highlighted.

The system outcome for a Cloud system covers a wide range of criteria including energy consumption [5], [6], cost (budget) [7], [8], total execution/completion time (make-span) [9], [10], number of host switches [11] and resource utilization/wastage [12] or a combination of criteria [13], [14]. Verma et al. [14], [15] proposed a bi-criteria Particle Swarm Optimization (PSO) algorithm to schedule tasks in the Cloud. They compared their algorithm with BHEFT proposed by Zheng and Sakellariou [16], which itself is an alteration of HEFT [17]. They have deadline and budget constraints as criteria. PSO is also used by Xiong and Wu [18], and Sridhar and Babu [19] for scheduling. However, these approaches mostly rely on knowledge about tasks' resource requirements, that is not necessarily available to the scheduler.

Among the criteria chosen by researchers some are particularly important in the context of real-time private Cloud. Energy consumption is the major component in cost of running a Cloud. Therefore, minimizing total energy consumption and subsequently total cost of private Cloud is important. An energy-aware scheduler in a real-time private Cloud should optimize total execution time because of its potential effect on total deadline violation. The importance of minimizing total execution time is highlighted when its effect on deadline violation is taken into account. The number of host switches also contributes to the cost of running the Cloud, because of the host start-up energy peak, and total execution time and/or deadline violation, due to the host start-up delay.

Kim, Beloglazov and Buyya [1] run a set of experiments on a private Cloud to decrease energy consumption. They assume complete knowledge about each incoming task's resource requirements. Therefore, they can change the CPU frequency of the host (using Dynamic Voltage and Frequency Scaling - DVFS) to meet task's associated deadline. Nevertheless, this prior knowledge of tasks' resource requirements are not always available to the scheduler.

In a study by Panda et al. [13] maximizing the profit and minimizing the task make-span was sought. They combined Profit Based Task Scheduling (PBTS) and Cloud Min-Min Scheduling (CMMS) [20]. They claimed improvements in average utilization. Because the presumed Cloud system was heterogeneous, they argued the host selection strategy changes the results.

Resource utilization, task run time and response time are studied by Tsafrir, Etsion and Feitelson when a back-filling mechanism is proposed to schedule tasks in distributed systems [2]. It requires prior knowledge/prediction about tasks' resource requirements to fill the remaining resources with suitable tasks. This knowledge might not exist or the predictions might not be accurate enough. Another scheduling policy is proposed by Kaur and Challa [21] where it aimed to minimize energy consumption and execution time. They considered prior knowledge about tasks' resource requirements as well.

Lee and Zomaya [22] proposed two energy-conscious scheduling policies ECTC and MaxUtil. They intended to maximize resource utilization and explicitly taking into account both active and idle energy consumption. For a given task, ECTC and MaxUtil check every resource and identify the most energy-efficient resource for it. The difference between the two is in their cost functions. The cost function of ECTC computes the actual energy consumption of the current task subtracting the minimum energy consumption. MaxUtil, on the other hand, is devised with the average utilization of resources and assigns the task to a host with maximum utilization. The results showed that ECTC and MaxUtil outperformed random scheduling algorithms. ECTC and MaxUtil focus on consolidation of as many tasks onto fewer hosts in order to decrease the number of active hosts. MaxUtil involves lower level of complexity as it uses average resource utilization. Although it was not designed for a real-time private Cloud, it has the potential to minimize the number of active hosts and subsequently decrease total energy consumption. Nonetheless, consolidating tasks to a fewer number of hosts in turn might cause resource conflicts and even increase total energy consumption and undesirably total execution time.

An energy-aware scheduling algorithm is proposed by Li et al. [23] to achieve a balanced load in a private Cloud. In their study, load balancing is defined as having an equal number of VMs on each processing node regardless of how much resources they might need. It makes this scheduling policy an applicable one in a real-time private Cloud where the tasks' resource requirements are unknown to the scheduler. However, because of their presumed Cloud system not being real-time, the scheduling policy was not devised to minimize deadline violation or completion time. We refer to this policy as Equal Load (EL). Even though EL was not designed to work in a real-time private Cloud, it has the potential to improve the system outcome.

In our earlier set of experiments, we proposed a scheduling policy for heterogeneous real-time Clouds called Energy-aware Deadline-Based Scheduling (EDS) [4]. This policy considers the limitations on all resources on the system including CPU, memory and network and substantially focuses on minimizing the total deadline violation. The same policy has the potential to lead to a desirable outcome for a real-time private Cloud.

In general, there are many scheduling policies for Cloud systems in the literature. However, the ones that do not rely on knowledge about tasks' resource requirements are applicable. Among them, it seems unlikely to find the best scheduling policy in all system settings. It is expected that the outcome of a scheduling policy is closer to the optimal value in some settings and not all. It brings the question about the possibility of having a single scheduling policy as an optimal policy that outperforms all the others in all settings of the real-time private Cloud system. If there is not such a policy, that seems to be likely, there is a need for developing a scheduling policy that learns and adapts. Such scheduling policy can use be the application of techniques such as Bayesian learning.

Bayesian learning is used in different areas. Its diverse set of applications in different disciplines represents its strength in dealing with a wide variety of problems. Bayesian method in Cloud computing is presented in a study by Sallam and

Li [24] where they carried out a multi-objective optimization when migrating VMs from a host to another. Their method is about selecting a VM for migration that satisfies objectives including load volume, power consumption, thermal state, resource wastage and migration cost. They are not concerned about the host selection process or initial scheduling policy.

Bayesian techniques are also applied for failure management [25], [26], mobile Cloud computing [27] and scheduling [28]. In the area of scheduling Wang et al. [28] used the concept of trust to provide a fair distribution of the load in the system or balance the load. It doesn't cover the other important objectives in the Cloud such as energy consumption and execution time. The goal of the study is a reliable scheduling policy that guarantees task execution even in the case of hardware failure. The degree of trust in each node is defined as its performance and the feedback from its performance in cooperation with other nodes. The framework is an extension of the Dynamic Scheduling framework presented by Harrison [29] in 1975. The data from a study by Calheiros et al. [30] are used for evaluation and they show that their approach can reduce the task's failure ratio.

Bayesian technique is widely used in Cloud computing in failure management. The reason can lie in the uncertainty of the hardware failure that can happen to any resource at any time. Bayesian techniques gives the system the ability to adapt to the changes in the system. A scheduling policy that is equipped with Bayesian learning technique is expected to provide a better system outcome for Cloud systems and real-time private Cloud in particular, because of its specific characteristics.

III. Optimizing Scheduling in Real-Time Private Clouds

The notion of a scheduling in a real-time private Cloud refers to the allocation of tasks to the hosts (Cloud resources). In the following subsections the formal annotations for real-time private Cloud scheduling optimization are given in details. These are then used to explain the Bayesian heuristic.

A. System Input Set

The optimized scheduling method can utilize the information from the system to facilitate scheduling in a real-time private Cloud. Available information include task arrival rate, average remaining deadline and current CPU utilization. System Input (SI) represents a set of value ranges. If $TAR = \{tar_0, tar_1, \ldots, tar_n\}$, $ARD = \{ard_0, ard_1, \ldots, ard_m\}$ and $CCU = \{ccu_0, ccu_1, \ldots, ccu_p\}$ Let $SI = (tar_2, ard_6, ccu_0)$ be an example of the system input.

- Task arrival rate (TAR) is the mean value for the tasks' arrival intervals. The longer the intervals and the more apart the tasks arrive, the bigger the TAR is. A small TAR value shows that the real-time private Cloud receives tasks relatively fast and is under pressure to schedule them. On the other hand a large TAR value means the system is relatively relaxed in scheduling the tasks.
- Average Remaining Deadline (ARD) is the averaged value of the deadlines for the tasks that are currently running in the system. It is worth noting that the real-time private Cloud is not aware of the exact resource capacities required for finishing the task. The remaining deadline is the only information about tasks that is available to the system.
- Current CPU Utilization (CCU) is the average CPU utilization over all hosts. It represents the way CPU in the system is being used. Although it is an averaged value, it can help in understanding the behavior of the system and predicting the system's energy consumption in particular.

B. Scheduling Assignments in Real-Time Private Clouds

Let $T = \{t_1, t_2, \ldots, t_n\}$, where $n \in \mathbb{N}$, denote the set of tasks in a workload and the parameters of the task be given as follows:

- Task Intensity, $TI = \{$CPU-intensive, memory-intensive, network-intensive, non-intensive$\}$, ti: $T \rightarrow TI$
- Task Deadline, td: $T \rightarrow \mathbb{N}$

Let $H = \{h_1, h_2, \ldots, h_m\}$, where $m \in \mathbb{N}$, denote the set of hosts in a real-time private Cloud and let its parameters be as follows

- Host Processing Speed, hps: $H \rightarrow \mathbb{N}$
- Host Memory, hm: $H \rightarrow \mathbb{N}$
- Host available bandwidth, hb: $H \times H \rightarrow \mathbb{N}$
- Host Maximum number of VMs, hmvm: $H \rightarrow \mathbb{N}$
- Host Status, hs: $H \rightarrow \{On, Off\}$

The assignment of tasks to the hosts can be defined as $A = \{a | a : T \rightarrow H\}$, where A is the set of all assignments of tasks to hosts. Note that, since T and H are finite, A is also finite, and represents the set of all possible assignments. A single deployment set can be $a_i = \{(t_1, h_i), (t_2, h_i), \ldots, (t_n, h_i)\} \in A$. Not all scheduling policies can be used in a real-time private Cloud. The followings are applicable to schedule tasks in a real-time private Cloud.

- MaxUtil [22] It schedules based on a host's average CPU utilization and focuses on consolidating as many tasks onto fewer hosts in order to reduce the number of active hosts.
- Intensity Based (IB) It follows the elementary idea of assigning an intensive task to a host with the most available capacity on that resource. A non-intensive task is randomly assigned.
- Greedy Deadline (GD) [4] It aims to decrease the deadline violation by deploying the task on a host with the least utilization. It does not turn on a new host unless the active hosts cannot accommodate any new task.
- Intensity-aware Greedy Deadline (IGD) IGD is similar to GD with the difference of how it deals with intensive tasks. If the host with maximum available resource A is running an A-intensive task, the host with the second maximum available resource A will be selected. Unless all hosts have the same number of A-intensive tasks. A non-intensive task is treated like a CPU-intensive one.
- Equal Load (EL) [23] It is an energy-aware scheduling policy proposed to balance the load in a private Cloud.

EL keeps the number of tasks on each host as equal as possible.
- Intensity-aware Equal Load (IEL) IEL acts like EL but puts into account the intensity of the tasks as described in IGD. If the selected host by EL is running an A-intensive task, another host will be chosen, unless all hosts are running the same number of A-intensive tasks.

The changes on the scheduling policies are expected to improve the measured criteria as they take into account the intensity of the tasks. Also, each policy results in a set of assignments. These assignments influence the performance of the system (system outcome). To find an optimal assignment and compare the performance of different scheduling policies, relevant criteria should be chosen.

C. Criteria

The criteria that are aimed to be optimized are denoted as $C : A \rightarrow \mathbb{R}$. To illustrate our approach we use four criteria energy consumption, deadline violation, execution time and number of host switches, $C = \{ec, dv, et, hs\}$.

- Energy Consumption (EC) Energy consumption contributes to carbon emissions and directly affects the cost of running the Cloud (both the hosts and the cooling devices). To calculate the total energy consumption, the energy model reported by Hsu et al [31] is applied. It provides a set of rules that determine the energy consumption for VM number i, V_i, at the time t using α, the idle energy consumption and $\beta = \alpha$ as in (1) in Watts (W).

$$E_t(V_i) = \begin{cases} \alpha & if\ idle \\ \beta + \alpha & if\ 0\% \leq CPUutil. \leq 20\% \\ 3\beta + \alpha & if\ 20\% < CPUutil. \leq 50\% \\ 5\beta + \alpha & if\ 50\% < CPUutil. \leq 70\% \\ 8\beta + \alpha & if\ 70\% < CPUutil. \leq 80\% \\ 11\beta + \alpha & if\ 80\% < CPUutil. \leq 90\% \\ 12\beta + \alpha & if\ 90\% < CPUutil. \leq 100\% \end{cases} \quad (1)$$

  EC values can be divided into levels where $EC = \{ec_0, ec_1, \ldots, ec_i\}$.
- Deadline Violation (DV) In a real-time system each task has a deadline. The resource allocation should aim to prevent or decrease the total deadline violation. Preventing the deadline violation in a real-time system is an important aspect of QoS. DV can sit in a range from $\{dv_0, dv_1, \ldots, dv_j\}$ set.
- Execution Time (ET) Total execution time for a given workload is another indicator of how well the system satisfies the task's resource requirements and is a measurement of QoS. ET values will be a member of $\{et_0, et_1, \ldots, et_k\}$ set.
- Host Switches (HS) The number of host switches is an explanation for a portion of delays and energy consumption. The energy needed for starting a host increases the energy consumption. Also, each host switch (on/off) creates a switching delay. By preventing unnecessary host switches, there is a chance to decrease the total energy consumption, deadline violation and execution time. HS can have ranges from $\{hs_0, hs_1, \ldots, hs_l\}$ set.

D. Constraints

Not all assignment candidates $a \in A$ represent feasible alternatives. For instance, placing all tasks to a single host might exceed the threshold for the maximum number of VMs on the host. The set of constraints $Omega$ is defined that considers only the limitation on the number of VMs that a host can run as $Omega = \{Omega_{hmvm}\}$. Note that any other constraint can be added to the system. Let $a^{-1} : H \rightarrow P(T)$ denote the inverse relation to $a \in A$, i.e. $a^{-1}(x) = \{t \in T | a(t) = h\}$. Then the constraint on the maximum number of VMs on a host $Omega_{hmvm} A \rightarrow \{true, false\}$ is defined as follows:

$$Omega_{hmvm} = \forall h \in H \sigma_{tina^{-1}(h)} 1 \leq hmvm(h)$$

IV. Bayesian Heuristic for Scheduling in Real-Time Private Clouds

An important aspect of scheduling optimization in real-time private Clouds is to realize how a certain setting in the input affects the criteria. Bayesian learning method requires the prior probability of a criterion value for a given system input. SI values in combination with the scheduling policy chosen can be used to calculate the conditional probabilities of the values for each criterion. These conditional probabilities then can be utilized to obtain posterior probabilities in scheduling tasks in a real-time private Cloud.

$P(ec_i)$ is the probability of the system output being ec_i for energy consumption. This is the prior probability. Prior probability is then used to calculate the conditional probability. $p(ec_i(SI_j\ \&\ SchedulingPolicy_k))$ is the probability of the energy consumption sitting in the ec_i range when SI_j is the system input and scheduling policy k is applied. Conditional probability should be calculated for range of each criterion as in Algorithm 3.

Data: Obtained results from scheduling policies
Result: Bayesian scheduling policy selection
Read Training DataSet;
Calculate the prior probability for currentSI (current system input set);
Calculate the prior probability for each scheduling policy, SP_i;
Calculate conditional probabilities;
posterior probability for each scheduling policy SP_i = $\frac{conditionalProbabilitySP_i \times priorProbabilitySP_i}{priorprobabilityforcurrentSI}$;
Select a scheduling policy based on the posterior probabilities;

Algorithm 1: Bayesian scheduler

The first stage of Bayesian heuristic is to calculate prior and conditional probabilities as in algorithm 1. In a real-time private Cloud the aim in to minimize the value of the outcome criteria, EC, DV, ET and HS. Therefore, for any given

Data: Obtained results from scheduling policies
Result: Prior probability for currentSI and each scheduling policy
for ***each*** *instance from TrainingDataSet* **do**
  increment totalCounter;
  if *instance's SystemInputSet equals currentSI OR currentSI doesn't exist in the TrainingDataSet AND instance's SystemInputSet's TAR equals currentSI's TAR* **then**
    increment counter;
  end
end
prior probability for currentSI = $\frac{counter}{totalCounter}$;
for ***each*** *scheduling policy,* SP_i **do**
  for *textbfeach instance from TrainingDataSet* **do**
    if *instance's scheduler equals* SP_i **then**
      increment k_i
    end
  end
end
prior probability for SP_i = $\frac{k_i}{totalCounter}$;
Algorithm 2: Calculating prior probabilities

Data: Obtained results from scheduling policies
Result: Conditional probability for each scheduling policy resulting in a Low criteria value
for ***each*** *instance from TrainingDataSet* **do**
  for ***each*** *scheduling policy,* SP_i **do**
    if *instance's scheduler equals* SP_i **then**
      increment $counter_i$;
      $totalCriteriaValue_i+=$ $instance'soutcomevalue$;
    end
  end
end
conditional probability for each scheduling policy, SP_i resulting in a 'Low' criteria value = $\frac{totalCriteriaValue_i}{counter_i}$;
Algorithm 3: Calculating conditional probabilities

SI set criteria values should be calculated based on which scheduling policy is applied. Then, the scheduling policy with the minimum value for the given setting will be chosen. It means the scheduling algorithm has the highest probability to lead to a more optimal outcome for the system. It is the posterior probability.

V. Evaluation

The real-time private Cloud simulations are performed in a system with 20 homogeneous hosts. The reason for choosing a homogeneous system is to eliminate the effect of different host capacities on the system outcome. The initial available bandwidth to each host is set to 1000 Hertz. Different scheduling policies (discussed in Section III-B) are deployed.

A. Experimental Settings

System input set have the value ranges as follows:

- TAR To generate the task arrival intervals, Poisson distribution is used. Multiple mean values are assigned to this distribution as 2, 3, 4, 5 and 6. Therefore the TAR value range is a member of the TAR set with $\{2orshorter, 3, 4, 5, 6orlonger\}$ as members.
- ARD Averaged remaining deadlines is the set ARD that according to the simulations can have these members $\{Low, Medium, High\}$.
- CCU Current CPU utilization set is a set where the members are $\{Low, Medium, High\}$.

The value ranges representing each member in ARD and CCU are the first, second and third portion of the observed results.

B. Test Data Sets

For evaluation purposes, it is to test the proposed method on workloads from a real system. We used the workloads provided in CloudSim package [30], [32] in its later version used for evaluating consolidation algorithms [11]. These workloads are parts of the CoMon project, a monitoring infrastructure for PlanetLab [33]. Ten days were randomly chosen in March and April 2011 and we base our evaluation on the same chosen dates to facilitate the reproduction of the results. However, the workloads do not belong to a private real-time Cloud, so the tasks do not have a deadline. Uniformly randomly numbers are associated to them as deadlines.

VI. Results

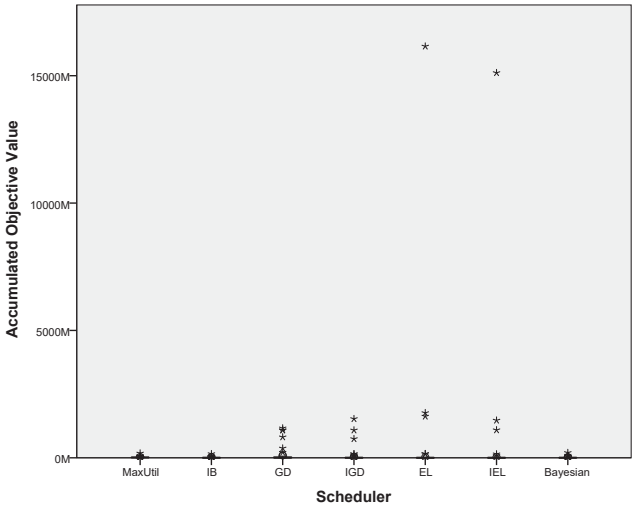
The results are then compared with scheduling policies in Section III-B. Fig. 1 illustrates the box plots for accumulated criteria values.

In Fig. 1 (a), IEL and EL have outliers that are substantially high in comparison to other schedulers. After zooming in, showed in Fig. 1 (b), GD and IGD exhibit non-optimal high measurements. It is in Fig. 1 (c) that Bayesian strategy shows its strength over the rest of the scheduling policies, MaxUtil and IB. Bayesian scheduler does not have the outlier values and resulted in a more optimal outcome for the system.

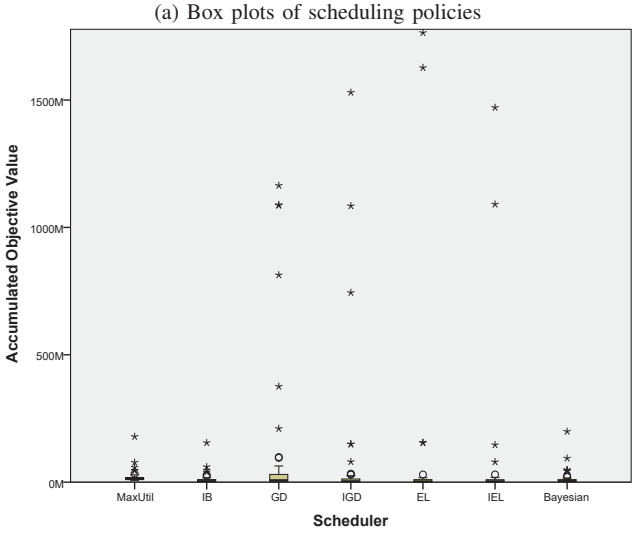
VII. Conclusions and Future Work

In a real-time private Cloud different scheduling policies might lead to a sub-optimal outcome for the system in specific settings of hosts, current VMs running on the hosts, arrival intervals, resource utilization and the deadline of the tasks. In this paper a Bayesian heuristic strategy is proposed for scheduling tasks in private real-time Clouds where it switches between available policies to come closer to the optimal outcome according to the system's current status.

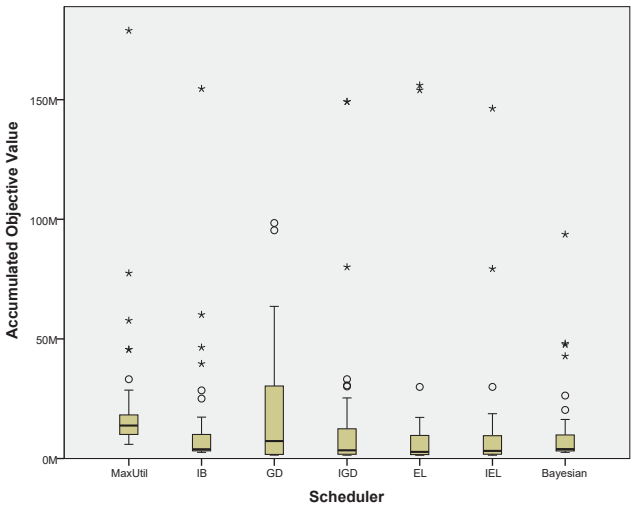
The results showed that the Bayesian heuristic strategy lead to a more optimal outcome for the system based on energy consumption, deadline violation, execution time and the number of host switches. It means that in any given real-time private Cloud the historical data for different deployed scheduling policy can be used in the Bayesian heuristic strategy to determine the best possible policy to choose according to the current status of the system. It



(a) Box plots of scheduling policies



(b) Re-scaled box plots of scheduling policies



(c) Re-scaled box plots of scheduling policies

Fig. 1 The box plots for the outcome of each scheduling policy in comparison to the Bayesian scheduler

also accentuates the potential for switching between policies to further optimize the outcome as a universally optimal scheduling policy is unlikely.

It is worth noting that this research is conducted using six scheduling policies. However, the Bayesian strategy is expandable to any number of deployed policies in the system as it uses the observed results of any given policy for training. Moreover, Bayesian strategy includes the recent results in its next decision. It provides the system with the ability to adapt to the changes in the system in a robust manner. The system can benefit from this in order to cope with the changes. It include but is not limited to the way this strategy deals with hardware failure. Nonetheless, its effectiveness in hardware failure needs to be evaluated.

This work can be further improved by experimenting other heuristic strategies for policy switching. Although there are widely used heuristics for scheduling, their performance in switching scheduling policy in a real-time private Cloud and the comparison with Bayesian strategy can be studied. The candidate heuristics, however, need to have the learning and adapting ability to suit the tests for hardware failure recovery.

Acknowledgment

This research is conducted with the financial support of SUPRA (Swinburne University Postgraduate by Research Award). Constructive reviews by Irene Moser from Swinburne University of Technology (Melbourne, Australia) and productive ideas by Aldeita Aleti from Monash University (Melbourne, Australia) are greatly appreciated and acknowledged.

References

[1] Kyong Hoon Kim, Rajkumar Buyya, and Jong Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In *CCGRID*, volume 7, pages 541–548, 2007.

[2] Dan Tsafrir, Yoav Etsion, and Dror G Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. *Parallel and Distributed Systems, IEEE Transactions on*, 18(6):789–803, 2007.

[3] Chuan-Feng Chiu, Steen J Hsu, Sen-Ren Jan, and Jyun-An Chen. Task scheduling based on load approximation in cloud computing environment. In *Future Information Technology*, pages 803–808. Springer, 2014.

[4] Sahar Sohrabi and Irene Moser. Energy-aware deadline-based scheduling in IaaS cloud with regard to the available memory. In *Proceedings of International Conference on Advanced Computing and Services*. World IT Congress, 2015.

[5] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10. San Diego, California, 2008.

[6] Nikzad Babaii Rizvandi, Javid Taheri, Albert Y Zomaya, and Young Choon Lee. Linear combinations of DVFS-enabled processor frequencies to modify the energy-aware scheduling algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 388–397. IEEE, 2010.

[7] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 49. ACM, 2011.

[8] Jia Yu and Rajkumar Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3):217–230, 2006.

[9] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, and Shun-Sheng Wang. Towards a load balancing in a three-level cloud computing network. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 1, pages 108–113. IEEE, 2010.

[10] Linan Zhu, Qingshui Li, and Lingna He. Study on cloud computing resource scheduling strategy based on the Ant Colony Optimization Algorithm. *IJCSI International Journal of Computer Science Issues*, 9(5):1694–0814, 2012.

[11] 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*, 24(13):1397–1420, 2012.

[12] Aameek Singh, Madhukar Korupolu, and Dushmanta Mohapatra. Server-storage virtualization: integration and load balancing in data centers. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 53. IEEE Press, 2008.

[13] Sanjaya K Panda and Prasanta K Jana. A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment. In *Electronic Design, Computer Networks & Automated Verification (EDCAV), 2015 International Conference on*, pages 82–87. IEEE, 2015.

[14] Amandeep Verma and Sakshi Kaushal. Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pages 1–6. IEEE, 2014.

[15] Amandeep Verma and Sakshi Kaushal. Cost minimized pso based workflow scheduling plan for cloud computing. pages 37–43, 2015.

[16] Wei Zheng and Rizos Sakellariou. Budget-deadline constrained workflow planning for admission control. *Journal of grid computing*, 11(4):633–651, 2013.

[17] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *Parallel and Distributed Systems, IEEE Transactions on*, 13(3):260–274, 2002.

[18] Ying Yidu Xiong and Yan Yan Wu. Cloud computing resource schedule strategy based on pso algorithm. In *Applied Mechanics and Materials*, volume 513, pages 1332–1336. Trans Tech Publ, 2014.

[19] M Sridhar and G Babu. Hybrid particle swarm optimization scheduling for cloud computing. In *Advance Computing Conference (IACC), 2015 IEEE International*, pages 1196–1200. IEEE, 2015.

[20] Jiayin Li, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin, and Zonghua Gu. Online optimization for scheduling preemptable tasks on iaas cloud systems. *Journal of Parallel and Distributed Computing*, 72(5):666–677, 2012.

[21] Harmeet Kaur and Rama Krishna Challa. A new hybrid virtual machine scheduling scheme for public cloud. In *Advanced Computing & Communication Technologies (ACCT), 2015 Fifth International Conference on*, pages 495–500. IEEE, 2015.

[22] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[23] Jiandun Li, Junjie Peng, Zhou Lei, and Wu Zhang. An energy-efficient scheduling approach based on private clouds. *Journal of Information & Computational Science*, 8(4):716–724, 2011.

[24] Ahmed Sallam and Kenli Li. A multi-objective virtual machine migration policy in cloud systems. *The Computer Journal*, 2013.

[25] Qiang Guan, Ziming Zhang, and Song Fu. Ensemble of bayesian predictors for autonomic failure management in cloud computing. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[26] Qiang Guan, Ziming Zhang, and Song Fu. Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems. *Journal of Communications*, 7(1):52–61, 2012.

[27] Xianbin Wang, Guangjie Han, Xiaojiang Du, and Joel JPC Rodrigues. Mobile cloud computing in 5g: Emerging trends, issues, and challenges [guest editorial]. *Network, IEEE*, 29(2):4–5, 2015.

[28] Wei Wang, Guosun Zeng, Daizhong Tang, and Jing Yao. Cloud-DLS: Dynamic trusted scheduling for cloud computing. *Expert Systems with Applications*, 39(3):2321–2329, 2012.

[29] J Michael Harrison. Dynamic scheduling of a multiclass queue: Discount optimality. *Operations Research*, 23(2):270–282, 1975.

[30] Rodrigo N Calheiros, Rajiv Ranjan, César AF De Rose, and Rajkumar Buyya. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv:0903.2525*, 2009.

[31] Ching-Hsien Hsu, Kenn Slagter, Shih-Chang Chen, and Yeh-Ching Chung. Optimizing energy consumption with task consolidation in clouds. *Information Sciences*, 258:452–462, 2014.

[32] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of

resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[33] KyoungSoo Park and Vivek S Pai. CoMon: a mostly-scalable monitoring system for planetlab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, 2006.