Characterizing energy consumption of IaaS clouds in non-saturated operation

Haleh Khojasteh, Jelena Mišić, and Vojislav B. Mišić
Ryerson University, Toronto, Canada M5B 2K3

***Abstract*—To save energy, physical machines (servers) in cloud data centers are partitioned in different pools, depending on whether they are kept on at all times and/or whether they have virtual machines instantiated. Partitioning (pooling) of servers affects the power consumption of the data center but also the performance and responsiveness to user requests. In this paper we examine the behavior of pool management scheme in different operating regions which correspond to linear operation, transition to saturation, and saturation, in particular the tradeoff between performance and power consumption. In addition, our results show that the offered load (analogous to the concept often used in networking) does not offer complete characterization of data center operation; instead, the impact of task arrival rate and task service time must be considered separately.**

***Index Terms*—cloud infrastructure; resource allocation; power consumption; performance evaluation**

I. Introduction

Energy efficiency is one of the top priorities in a cloud data center [10], which is why ways to reduce energy consumption without sacrificing performance are among the most important research topics. If the cloud data center is configured so that user tasks are provisioned on virtual machines (VMs) executing on servers or physical machines (PMs), energy efficiency may be improved by dividing the PMs in three groups or pools [4]. In this approach, PMs in the *cold* pool are normally kept switched off, while PMs in the *warm* and *hot* pool are normally switched on, the latter having virtual machines (VMs) already instantiated and ready to provision user tasks. When a user request that asks for one or more VMs arrives, the pools are checked in sequence from hot through warm to cold; if necessary, PMs are switched on and/or VMs instantiated. Upon termination of user tasks, hot PMs with idle VMs are moved back to the warm pool, while excess warm PMs are switched off and, thus, moved back to the cold pool.

In this manner, power consumption is made dependent on the data center load, which leads to improved energy efficiency and 'greener' computing [1]. At the same time, the manner in which the PMs are partitioned into hot, warm, and cold pools will affect the performance of the data center with respect to request response time and probability that a request will be blocked due to insufficient resource availability.

Earlier performance studies of pool management schemes [4] have focused on IaaS clouds operating in saturation regime where all the PMs are essentially busy (almost) all the time. This region is not really usable for cloud operators since blocking of user requests reaches values which are unacceptably high. Instead, in this paper we focus on linear and transition regimes which are more important in practice. We analyze the boundaries between the operating regimes with respect to overall system load, which is a function of the distribution of task arrival rate, task service time, and look-up overhead needed to decide whether the task can be accommodated or not. We evaluate the energy efficiency of this model in different scenarios.

Interestingly enough, our results indicate that characterizing the load with a single value, which is customary in the performance evaluation of communication networks [5], is not quite appropriate for cloud data centers. Namely, searching for the appropriate PM (or PMs) on which tasks from the current user request will be provisioned, necessitates a certain overhead which depends on the characteristics of the request but also on the current load of the system. Therefore, it is of interest to evaluate the performance of the scheme under both varying user request arrival rate and user task service time.

The rest of the paper is organized as follows: Section II surveys related work in cloud resource allocation. Section III presents the probabilistic model of the resource allocation in a pooled IaaS cloud and the manner in which it is solved to obtain full probability distribution of relevant performance indicators. Section IV presents the results of performance evaluation, while Section V discusses energy consumption and related issues. Section VI concludes the paper.

II. Related Work

Resource allocation in cloud systems has attracted the attention of a lot of research groups during recent years.

In [9], energy-efficient virtual resource allocation for the cloud has been formulated as a multi-objective optimization problem which is solved using an intelligent optimization algorithm. The work presented in [13] has proposed a congestion control method using an index for evaluating fair resource allocation in case of congestion. In [12], authors have proposed a dynamic resource allocation in a cloud environment which considers computing job requests that are characterized by their arrival and teardown times, as well as a predictive profile of their computing requirements during their activity period. Two algorithms to adjust resource allocation and task scheduling adaptively based on the actual task execution time have been proposed for in [7].

In [6], allocating VMs to applications with real-time tasks is formulated as a constrained optimization problem. Since an exhaustive search for solutions has exponential complexity, polynomial-time heuristic model was proposed to solve the

problem. Moreover, the cost obtained by this heuristic model was compared with the optimal solution and an Earliest Deadline First (EDF-greedy).

In [14], an ad hoc parallel data processing framework has been presented to exploit the dynamic resource allocation for both task scheduling and task execution in IaaS clouds. Specific tasks of a processing job can be assigned to different types of VMs. The algorithm presented in [2] formed groups of VM instances according to their runtime deadlines and packed VMs in the same group on the same servers. Moreover, it shuts down some servers in time when the service request decreases in order to reduce energy consumption.

The work presented in [3] has proposed a resource allocation model using combinatorial auction mechanisms which uses energy parameters. Three algorithms have been introduced for different aspects of resource allocation.

However, no research has been done on the effects of the variation of offered load on cloud computing centers' behavior in linear and transition to saturation region. This is particularly important for congestion and admission control.

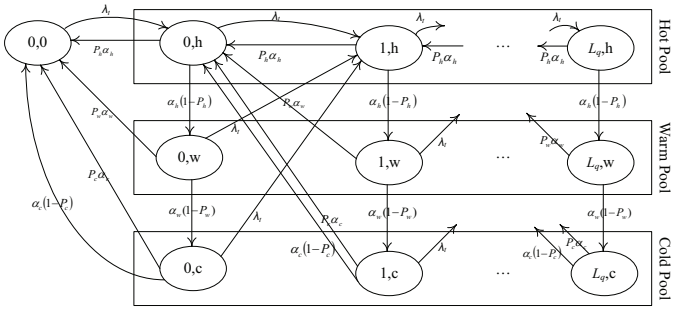
III. Analytical Model

We assume that the IaaS cloud center has a common input queue, while the three PM pools have separate queues of their own. A single PM can host a number of VMs simultaneously; this number is limited in order to ensure satisfactory performance level of the VMs [4]. Without loss of generality, we assume that all PMs are homogeneous as are the VMs. Furthermore, we assume that a single prebuilt VM image can satisfy all requests [4]; however, the model can easily be extended to cover PMs and/or VMs with different characteristics. To model the performance of an IaaS cloud center, we have constructed a probabilistic model that consists of three different submodels with one, three, and one instance each, respectively. Our model closely follows the one presented in [4], but our emphasis is on energy-related issues in the context of a non-saturated IaaS cloud center.

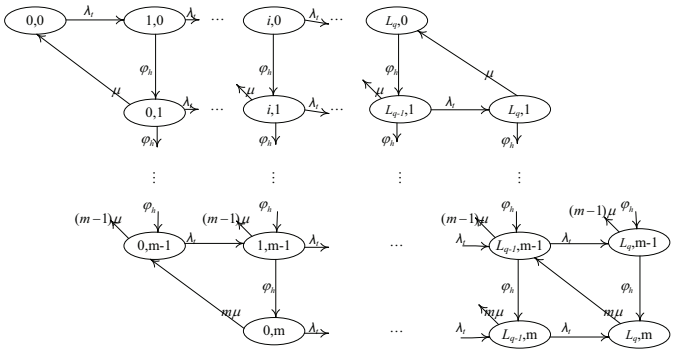
A. Resource allocation

Task requests arrive according to a Poisson process with arrival rate λ_t. An incoming request will be processed by the Resource Allocation module (RAM), shown with the two-dimensional Continuous Time Markov chain (CTMC) in Fig. 1(a). RAM checks the hot, warm, and cold pools (in that order) to find whether there is a sufficient number of PMs and idle VMs to accommodate the request; $1/\alpha_h, 1/\alpha_w$ and $1/\alpha_c$ are the mean look up delays in the respective pools. A hot PM can immediately begin service of a request, provided it has VMs to spare. If there is no hot PM with the required capacity, a warm PM may be used, but it must instantiate the required number of VMs before provisioning the request. If there is no warm PM either, a cold PM will be used, but must be switched on before instantiating the VMs. Obviously, the delays for the three types of PMs will differ, with hot PMs providing the shortest one.

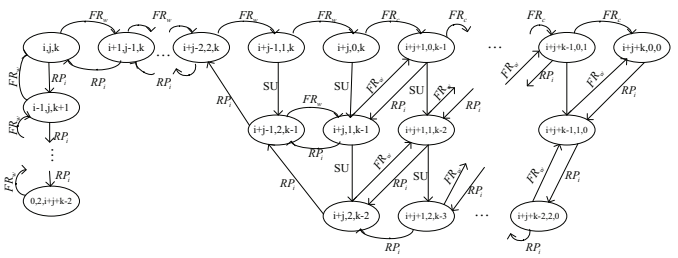
Once an idle VM is found, RAM will allocate the request. A request may be rejected if there is no space in the input queue,



(a) CTMC of the Resource Allocation module.



(b) CTMC of the Virtual Machine Provisioning module of the hot pool.



(c) CTMC of the Pool Management module.

Fig. 1. Analytical model of the IaaS provisioning (adapted from [4]).

or a suitable PM can't be found. The probability of the former event is $P_{bq} = \pi(L_q, h) + \pi(L_q, w) + \pi(L_q, c)$, where $\pi(i, j)$ denotes the probability of pool j having i requests while L_q is the capacity of the input queue; the probability of the latter is $P_{br} = \sum_{i=0}^{L_q} \frac{\alpha_c(1-P_c)}{\alpha_c + \lambda_t} \pi(i, c)$. Total blocking probability is, then, $P_{blk} = P_{bq} + P_{br}$.

If we define the probability generating function (PGF) for the number of tasks in the queue [11] as

$$V(z) = \pi(0,0) + \sum_{i=0}^{L_q} (\pi(i,h) + \pi(i,w) + \pi(i,c)) z^i, \quad (1)$$

mean waiting time can be calculated using Little's law [5] as

$$\overline{w_t} = \frac{\overline{v}}{\lambda_t (1 - P_{bq})}, \quad (2)$$

and mean look up time among pools can be obtained [4] as

$$\overline{lu_t} = \frac{1/\alpha_h + (1-P_h)((1/\alpha_w) + (1-P_w)(1/\alpha_c))}{1 - P_{bq}}. \quad (3)$$

B. Virtual machine provisioning

The request then waits in the PM input queue until a VM is ready to provision it. Provisioning is described with a Virtual Machine Provisioning module (VMPM) modeled as another CTMC, shown schematically in Fig. 1(b). The complete analytical model employs three such modules with identical structure corresponding to hot, warm, and cold pools, respectively.

Let ϕ_h as the rate at which a VM can be provided on a PM in the hot pool, and let μ be the service rate of each task. The arrival rates can be calculated as

$$\begin{aligned} \lambda_h &= \frac{\lambda_t(1-P_{bq})}{N_h} \\ \lambda_w &= \frac{\lambda_t(1-P_{bq})(1-P_h)}{N_w} \\ \lambda_c &= \frac{\lambda_t(1-P_{bq})(1-P_h)(1-P_w)}{N_c} \end{aligned} \tag{4}$$

for hot, warm, and cold pool, respectively, where N_h, N_w, and N_c denote the number of PMs in the respective pools. $P_h = 1 - (P_{na}^h)^{N_h}$ denotes the probability of a PM in the hot pool accepting the task; the complementary probability may be calculated as $P_{na}^h = \sum_{x\in\eta} \pi_x^h$. In the last expression, $\eta = \{(i,j,k)|i = L_q\}$ denotes a subset of VMPM states in which a task may be blocked.

By the same token, success probability for provisioning a task in warm and cold pool is $P_w = 1 - (P_{na}^w)^{N_w}$ and $P_c = 1 - (P_{na}^c)^{N_c}$, respectively.

C. Pool management

Provisioning of requests may require that PMs are moved between pools under the control of the Pool Management module (PMM), modeled with a CTMC shown in Fig. 1(c). The transition from a warm to hot PM occurs at a rate of $FR_w = (\lambda_t P_{bq}(1-P_h) + (1/SU_w))^{-1}$, where $1/SU_w$ is the mean time required for a warm PM to switch to hot state. Similarly, a cold PM can be moved to the hot pool at a rate of $FR_c = (\lambda_t P_{bq}(1-P_h) + (1/SU_c))^{-1}$, where $1/SU_c$ is the mean time needed for a cold PM to switch to hot state.

Conversely, if, upon a task ends execution and the number of idle hot PMs exceeds a predefined threshold, an idle hot PM can be moved to the warm pool to reduce energy consumption, or even to the cold pool if the number of PMs in the warm pool is above the predefined threshold, at a rate RP_i.

D. Integrated model

The overall model, thus, consists of three interactive stochastic submodels; this reduces complexity of the model itself but also the computational complexity of solving the model. It is, then, solved via successive fixed point iteration [8], shown as pseudocode in Algorithm 1. Iteration ends when the difference between the values of probabilities in successive iterations drops below a predefined threshold ($\Delta = 10^{-6}$).

Task waiting time is obtained as the sum of four components: waiting time in the global input queue, until the processing by RAM; RAM processing time; waiting time in the PM queue; lastly, the time for VM instantiation and deployment. Total response time is obtained by adding the waiting time to the duration of the actual service time.

Algorithm 1 Successive Substitution Method

```
Input: Initial success probabilities in pools: P_h0, P_w0, P_c0;
Input: Initial idle probability of a hot PM: P_i0;
Output: Blocking probability in common input queue: P_bq;
count ⟵ 0; maximum ⟵ 30; Δ ⟵ 1;
P_bq0 ⟵ RAM (Ph0, Pw0; Pc0);
[N_h, N_w, N_c] ⟵ PMM (P_h0, P_i0)
while Δ ≥ 10^-6 do
   count ⟵ count +1;
   [P_h, P_i] ⟵ VMM_hot (P_bq0, N_h);
   P_w ⟵ VMM_warm (P_bq0, P_h, N_w);
   P_c ⟵ VMM_cold (P_bq0, P_h, P_w, N_c);
   [N_h, N_w, N_c] ⟵ PMM (P_h, P_i)
   P_bq1 ⟵ RAM (P_h, P_w; P_c);
   Δ ⟵ |(P_bq1 − P_bq0)|;
   P_bq0 ⟵ P_bq1;
   if count = maximum then
      break;
   end if
end while
if count = maximum then
   return -1;
else
   return P_bq0;
end if
```

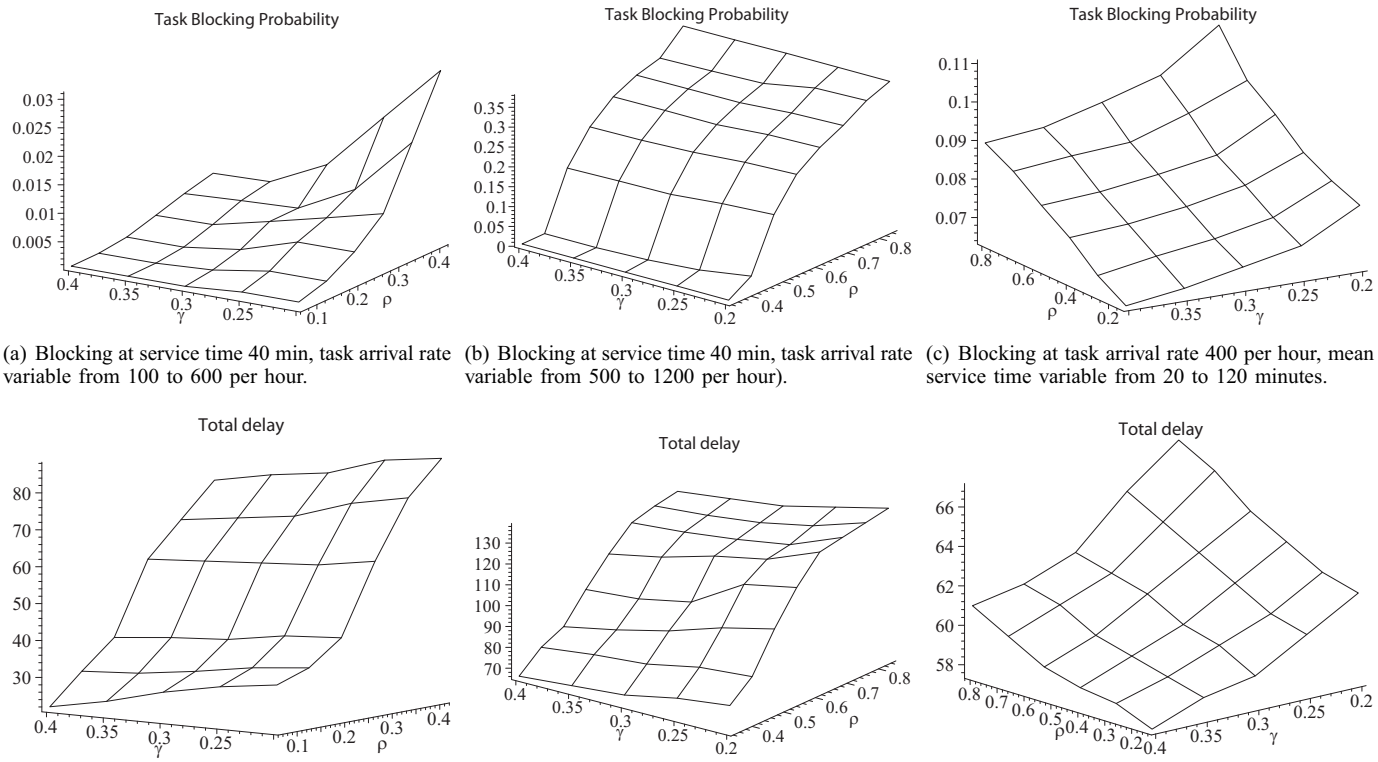
IV. Performance

To evaluate the performance of the pooled cloud system and to investigate the performance-energy tradeoff in more detail, we have solved the model for two scenarios. First, we kept the task service time fixed and varied task arrival rate. Second, we kept the task arrival rate fixed but varied task service time. In both cases, we have kept the total number of PMs constant at $N = 100$ while varying the initial proportion of PMs allocated to different pools: γN PMs in the hot and warm pools each, and $(1-2\gamma)N$ PMs in the cold pool. In addition, the number of PMs in the hot pool was kept at or above γN, while the other two pools were allowed to change. Transition to the warm pool was initiated when the number of idle hot PMs exceeded 2, and the same threshold was used for the transition from warm to cold pool.

Mean provisioning time for a task consists of the following components: mean waiting time in the input queue, mean time for pool look-up (which was shown to have a Coxian distribution [4]), mean waiting time in the queue of the allocated PM, and mean waiting time for VM provisioning. To obtain the total service time, we need to add the actual time of request execution.

A. Task blocking and total task delay

In Fig. 2, we analyze the effect of service time and task arrival rate on blocking probability and total delay for a task. To facilitate comparison under different combination of fixed and variable independent variables, we have plotted the diagrams as functions of γ and offered load calculated as $\rho = \frac{\lambda_t}{10N\mu_{tot}}$, where the maximum number of VMs running



(a) Blocking at service time 40 min, task arrival rate variable from 100 to 600 per hour.

(b) Blocking at service time 40 min, task arrival rate variable from 500 to 1200 per hour).

(c) Blocking at task arrival rate 400 per hour, mean service time variable from 20 to 120 minutes.

(d) Total delay (seconds) at service time 40 min, task arrival rate variable from 100 to 600 per hour.

(e) Total delay (seconds) at service time 40 min, task arrival rate variable from 500 to 1200 per hour).

(f) Total delay (seconds) at task arrival rate 400 per hour, mean service time variable from 20 to 120 minutes.

Fig. 2. Task blocking probability and total delay.

on a single PM is assumed to be 10. For clarity, we have divided the range of offered loads for the scenario with fixed service time into two sub-ranges; the corresponding results are shown in the diagrams in the leftmost and middle columns of Fig. 2.

As expected, both request blocking and total delay increase with load. Below the load of $\rho \approx 0.3$ to 0.4, Figs. 2(a) and 2(d), the cloud data center operates in linear regime with low blocking and reasonably small delay.

Beyond this load, however, blocking rapidly increases as does the delay. Figs. 2(b) and 2(e) show the increase of delay which appears to be gradual but only because a large number of task requests (over 10%) is rejected.

On the other hand, increasing the mean service time whilst keeping the task arrival rate fixed, shown in Figs. 2(c) and 2(f), results in an increase of blocking and delay which are much smoother. Note, however, that the data center operates in linear regime, well beyond the saturation limit, even though the value of ρ does increase above the threshold identified in the other four diagrams.

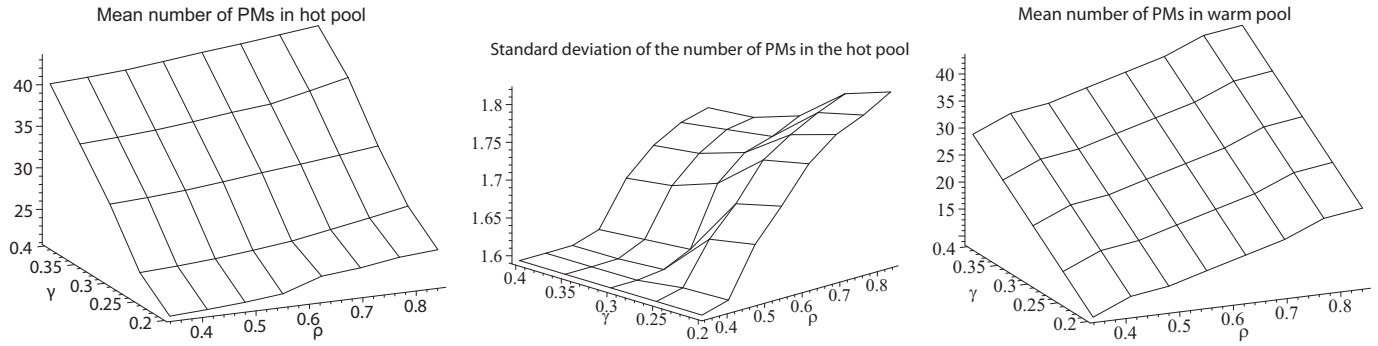
We note that for the same offered load, the cloud center appears to be more sensitive to the task arrival rate than to task service time, which is due to the overhead imposed by the provisioning process which increases with the number of tasks but is independent of the task service time.

B. Pool management

The observations above can be corroborated by calculating the steady-state number of PMs in different pools. The number of PMs in both hot and warm pools, shown in Figs. 3(a) and 3(c), exhibit a steady increase which is approximately linear function of the offered load, in one dimension, and similarly a linear function of the parameter γ. (We note that the distinction between variable arrival rate and variable service time does not apply here, since the overhead is incurred *before* the actual provisioning of tasks.) As can be seen, higher load leads to a shift in the partitioning of PMs, from the initial ratio defined by γ, towards an ever increasing number of PMs in the hot and warm pools, and the corresponding depletion of the cold pool.

However, the increase is far from being stationary, as witnessed by the diagram of standard deviation of the number of PMs in the hot pool in Fig. 3(b). As the offered load increases, the standard deviation exhibits a rapid increase, which means that the fluctuation of the number of PMs increases, as many PMs are being moved two and from the hot (and warm) pool in order to cater to the variable load.

At the same time, at low values of offered load, the mean number of PMs in the warm pool, Fig. 3(c), is noticeably lower than that in the hot pool, Fig. 3(a). Only when the load increases towards a rather high value of $\rho = 0.8$ do the two numbers begin to converge, meaning that most PMs are either hot or warm, and only a handful ever remains in the cold pool.

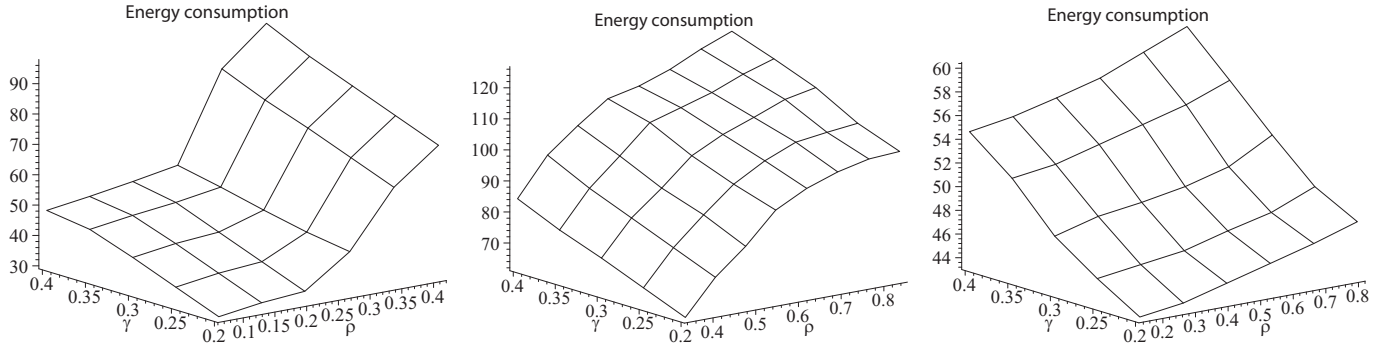


(a) Mean number of PMs in the hot pool.

(b) Standard deviation of the number of PMs in the hot pool.

(c) Mean number of PMs in the warm pool.

Fig. 3. Pertaining to steady-state partitioning of PMs into pools. Mean service time fixed at 40 minutes, task arrival rate variable from 500 to 1200 per hour).



(a) Energy consumption at service time 40 min, task arrival rate variable from 100 to 600 per hour.

(b) Energy consumption at service time 40 min, task arrival rate variable from 500 to 1200 per hour.

(c) Energy consumption at task arrival rate 400 per hour, mean service time variable from 20 to 120 minutes.

Fig. 4. Energy consumption as function of offered load and partitioning of PMs into pools.

V. Energy consumption

The fluctuation of the number of PMs in the three pools is reflected on energy consumption of the cloud center. Let the power consumption of a hot PM be δ_p, while that of a warm PM be $W_c\delta_p$ (the power consumption of a PM in the cold pool is, obviously, zero). If we denote the mean time spent in each state of the CTMC for the PMM, Fig. 1(c), with $\overline{T_{st}}$, the total energy consumption is

$$E_c = \sum_{s\in\zeta}(N_{h_s} + W_c N_{w_s})\delta_p \overline{T_{st}} \tag{5}$$

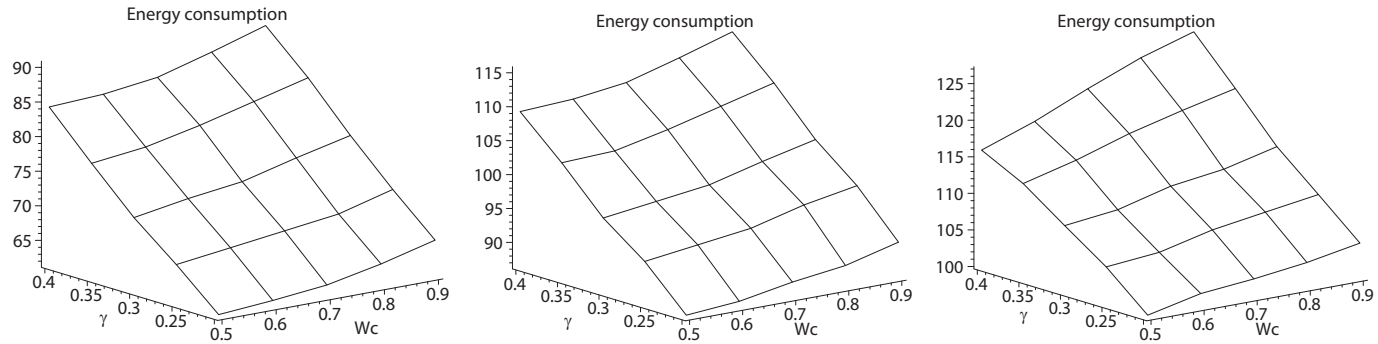
where ζ is the set of PMM states and N_{h_s} and N_{w_s} denote the number of PMs in hot and warm pool, respectively, in state s of the PMM.

Our first experiment follows closely the scenarios outlined in the previous Section, with the ratio of power consumption of a warm PM vs. that of a hot one fixed at $W_c = 0.5$. (For simplicity, we express all energy consumption values relative to the energy consumption of a hot PM with all 10 VMs instantiated.) The results are shown in Fig. 4; as before, we have split the range of observed values for the offered load in order to highlight the difference between linear and saturation regimes. As can be seen, the shape of surfaces obtained under variable task request arrival rate, Figs. 4(a) and 4(b), clearly indicate the boundaries of the linear regime in which the energy consumption is low and not very dependent on the offered load. Of course, if the initial partitioning of PMs gives preference to PMs in the hot and warm pools, higher energy consumption will result.

However, as soon as the task arrival rate exceeds the value of $\rho \approx 0.3$, energy consumption begins to rise at a considerable rate, due to higher proportion of PMs being in the hot pool, but also due to longer time spent in switching to and from hot state. (We assume that energy expenditure of a PM during switching is equal to that of a fully loaded hot PM.) Energy consumption appears to flatten at high loads above $\rho \approx 0.6$, but only because most of the PMs are in hot and warm pools most of the time, switching only occasionally to the cold state.

When the task arrival rate is fixed, energy consumption exhibits a nearly linear dependency on the mean task service time and the partitioning coefficient γ, as can be seen from Fig. 4(c). This behavior is similar to that observed for blocking probability and total task delay in Fig. 2.

Our final experiment involved energy expenditure under fixed task request arrival rate and mean service time (i.e., under fixed offered load), but with variable power consumption ratio W_c of a PM in the warm state vs. that of a PM in the hot state.



(a) Under mean arrival rate of 400 tasks per hour. (b) Under mean arrival rate of 700 tasks per hour. (c) Under mean arrival rate of 1000 tasks per hour.

Fig. 5. Energy consumption as function of the ratio of warm-vs.-hot PM power consumption and partitioning of PMs into pools. Mean service time fixed at 40 minutes.

The resulting diagrams are shown in Fig. 5, where energy expenditure (relative to the energy consumption of a single fully loaded PM) is nearly linearly dependent on both γ and W_c. However, it is much more sensitive to the former than to the latter, as the consequence of the fact that the partitioning parameter imposes a lower bound on the number of PMs in the hot pool. Thus the energy expenditure will not drop as much when reducing the load, as in Fig. 5(a), since the minimum number of PMs in the hot pool is still limited by the value of the partitioning parameter γ. This hints that partitioning into three pools may be inefficient, and that better results with respect to energy consumption could be obtained by simply having two pools, a hot and cold one, despite performance degradation possibly incurred in this setup. This remains a promising direction for further research.

VI. Conclusion

In this paper we have examined the behavior of an IaaS cloud data center in which servers are partitioned into pools of hot (i.e., always on), warm, and cold machines (both of which are switched on if there is a demand). We have focused on the operation in the linear regime, and we have shown that the transition to saturation is clearly visible from the diagrams of task request blocking probability. We have shown that the manner in which servers are partitioned in the pools affects the performance, sometimes even more that the variations of offered load. We have also shown that the task arrival rate is more critical parameter affecting the performance of the cloud data center than mean task service time, due to the overhead incurred in resource allocation and provisioning. We have also shown that the energy expenditure is highly dependent on the manner in which servers are partitioned into pools, and that reduced power consumption of servers kept in the warm state does not result in commensurate savings in energy.

Our future research will focus on cloud data centers with servers in the hot and cold state only, as well as on the development of load-predictive algorithms for server pool management.

References

[1] J. Baliga, R. W. A. Jayant, K. Hinton and R. S. Tucker. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, **99**(1):149–167, 2011.

[2] L. Guan, Y. Wang and Y. Li. A Dynamic Resource Allocation Method in IaaS Based on Deadline Time. In *14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, September 2012.

[3] T. Huu and C. Tham. An Auction-based Resource Allocation Model for Green Cloud Computing. In *IEEE International Conference on Cloud Engineering*, pp. 269–278, March 2013.

[4] H. Khazaei, J. Mišić and V.B. Mišić. Analysis of a Pool Management Scheme for Cloud Computing Centers. In *IEEE Transaction on Parallel and Distributed Systems*, **24**(5):849–861, May 2013.

[5] L. Kleinrock. *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, 1975.

[6] K. Kumar, J. Feng, Y. Nimmagadda, and Y.-H. Lu. Resource allocation for real-time tasks using cloud computing. *20th Int. Conf. Computer Communications and Networks (ICCCN 2011)*.

[7] J. Li, M. Qiu, J. Niu, Y. Chen and Z. Ming. Adaptive Resource Allocation for Preemptable Jobs in Cloud Systems. In *10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 31–36, Nov 2010.

[8] V. Mainkar and K. S. Trivedi. Sufficient conditions for existence of a fixed point in stochastic reward net-based iterative models. In *IEEE Transactions on Software Engineering*, 22(9):640–653, September 1996.

[9] L. Xu, Z. Zeng and X. Ye. Multi-objective Optimization Based Virtual Resource Allocation Strategy for Cloud Computing. In *IEEE/ACIS 11th International Conference on Computer and Information Science*, pp. 56–61, May 2012.

[10] H. Yuan, C.-C. J. Kuo and I. Ahmad. Energy efficiency in data centers and cloud-based multimedia services: An overview and future directions, In *IEEE Int. Green Computing Conf.*, pp. 375–382, 2010.

[11] H. Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, 1991.

[12] D. Tammaro, E. Doumith, S. Zahr, J. Smets and M. Gagnaire. Dynamic Resource Allocation in Cloud Environment Under Time-variant Job Requests. In *Third IEEE International Conference on Cloud Computing Technology and Science*, pp. 592–598, Nov 2011.

[13] T. Tomita and S. Kuribayashi. Congestion control method with fair resource allocation for cloud computing environments. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, pp. 1–6, August 2011.

[14] D. Warneke and O. Kao. Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud. In *IEEE Transactions on Parallel and Distributed Systems*, 22(6):985–997, June 2011.