Analyzing The Impact of Provisioning Overhead Time in Cloud Computing Centers

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

***Abstract*—In this paper, we analyze an efficient pool management model for cloud systems that partitions the servers (physical machines) into a hot and cold pool to improve energy efficiency. Servers are moved them from one pool to the other as needed to fulfill the incoming task requests, which are provisioned on virtual machines running on the servers. The model features two levels of task admission control: one at the global input, the other at each server separately. We examine the behavior of the cloud system in the regions of linear operation, transition to saturation and saturation, from the viewpoint of task rejection rates and energy consumption. Furthermore, we evaluate the sensitivity of task blocking probability and energy consumption to the pool partitioning threshold and the value of mean look up time in hot and cold pools, respectively, in different test scenarios.**

I. Introduction

In an Infrastructure-as-a-Service (IaaS) cloud, an incoming task service request undergoes several processing steps before being actually provisioned on a Virtual Machine (VM) executing on a designated server or Physical Machine (PM) [3]. These processing steps generally pertain to the search for a suitable PM that will provision the task; moving the task to the selected PM; and, finally, waiting for the instantiation of the VM on which the task will execute. Admission control through task rejection or blocking is thus performed at two steps along the way: at the global input queue and at the input queue of the selected PM.

To reduce energy expenditure incurred by PMs running idle, PMs are often partitioned into pools. For example, the model presented in [4] deals with three pools, hot pool with PMs that are always on and VMs already instantiated, warm pool with PMs that are on but without any instantiated VMs, and cold pool that consists of PMs switched off. In this setup, PMs are moved from cold to warm to hot state in order to fulfill the incoming task requests, and moved back to warm or cold pool when there is idle capacity in order to reduce energy consumption. The performance of the cloud center is then evaluated using the tools of probabilistic analysis and queueing theory. However, the emphasis was on the performance in saturation region, which is not the preferred operational regime for cloud providers as the achievable performance levels, esp. task blocking rate, is not very good; moreover, the analysis in that paper does not consider energy consumption in detail.

An extension of that analysis is presented in [5] where the focus was on linear and transitional regime, rather than on saturation regime. An interesting finding was that the aggregated metric of offered load, similar to the metric used to characterize networking systems, is insufficient to characterize the behavior of the cloud system. This is caused by the overhead incurred in processing the incoming task requests and admission control activities, which are dependent on the task arrival rate but not on the task service time.

In this paper, we consider a simpler model in which partitioning uses two pools only: the hot pool in which PMs are running with the maximum number of VMs instantiated and ready to provision user requests, and the cold pool in which PMs are turned off. We assume that the number of PMs in the hot pool does not fall below a predefined threshold, which serves to maintain the performance at an acceptable level. Our focus is again on the linear and transitional operational regimes which are much more interesting to cloud service providers, rather than the saturation regime in which task blocking reaches values that are unacceptably high in practice. We evaluate the performance of the cloud system expressed as task blocking rate and energy expenditure. Furthermore, we evaluate the sensitivity of the energy expenditure on the partitioning threshold, as well as on the mean look up time in the hot and cold pools.

The rest of the paper is organized as follows: In Section II, we survey related work in cloud resource allocation. Section III presents the stochastic sub-models and their interactions. Section IV presents the numerical results obtained from the analytical model. At last, Section V concludes the paper.

II. Related Work

Several research groups have been working on resource allocation and provisioning issues in the cloud.

In [6], authors have proposed two adaptive algorithms for resource allocation and task scheduling that adjust the resource allocation adaptively based on the actual task execution time.

The work presented in [8] has proposed a dynamic resource allocation in which job requests are characterized according to their arrival and teardown times, and allocation is made on the basis of a predictive profile of their computing requirements during their activity period.

In [1], an algorithm has been presented which forms groups of VM instances based on their runtime deadlines and packs VMs in the same group on the same servers. In order to reduce energy consumption, the algorithm can shut down some servers when the service request rate decreases.

The ad hoc parallel data processing framework presented in [9] exploits 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.

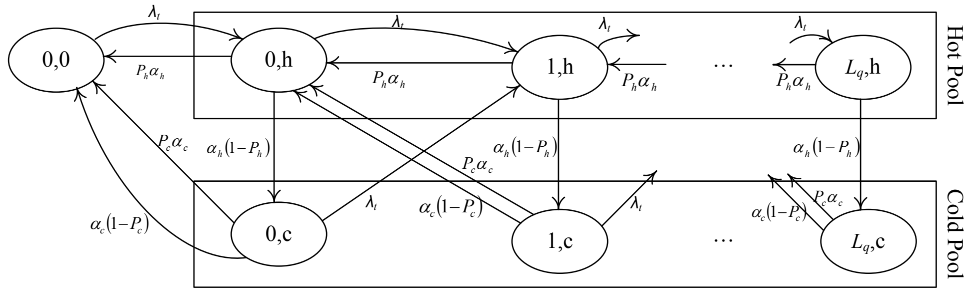


Fig. 1. Resource allocation.

In [10], the energy efficiency of virtual resource allocation for cloud computing has been formulated as a multi-objective optimization problem and the problem has been solved by intelligent optimization algorithm.

In [2], authors have 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, the impact of the provisioning time and, more specifically, the look up time needed to find a suitable PM in the appropriate pool has largely been ignored so far.

III. The model of the cloud system

An IaaS cloud center consists of a number of PMs (servers) running a number of VMs each that fulfill task requests submitted by users. All task requests are provisioned using a customized disk image [3]. For simplicity, we assume that the PMs are homogeneous as are the VMs, and that pre-built images satisfy all service requests. Also, we assume that each task request can be provisioned on a single VM.

A. Resource allocation

An incoming task request is first received in the global input queue, where it waits processing by a resource allocation module. This module will search the hot and cold pools for the required resources, i.e., PMs with idle capacity. If such a PM is found, the module will allocate the task to it, possibly instructing the PM to be switched on in the process; otherwise, the task may be rejected in what is effectively the first level of admission control. The operation of the resource allocation module can be described with a Continuous Time Markov Chain (CTMC) shown in Fig. 1. Task request arrivals are modeled as a Poisson process with rate λ_t. Task requests are lined up in the global finite queue with a maximum capacity of L_q to be processed. Each state of the CTMC in Fig. 1 is labeled as (i, j), where i denotes the number of tasks in the queue and j presents the pool on which the current task is under provisioning. Index h and c indicate hot and cold pool respectively. P_h and P_c present success probabilities of finding a PM that can accept the current task in the hot and cold pool respectively. $1/\alpha_h$ and $1/\alpha_c$ are mean look up times for finding an adequate PM in hot and cold pool respectively. The resource allocation module also calculates the task blocking probability, P_{bq}, which is the input parameter for the modules described below.

B. Virtual machine provisioning

If the task is allocated to one of the PMs, it is transferred to the input queue of the PM, which effectively implements the second level of task admission control. Once the task reaches the head of the said queue, the required VM is instantiated and the task is actually provisioned, i.e., the actual service starts. The operation of virtual machine provisioning module managing the hot PM pool can be described with a CTMC shown in Fig. 2. Each state of Markov chain is labeled by (i, j) in which i denotes the number of tasks in PM's queue and j is the number of VM that are already deployed on the PM. λ_h is the arrival rate to each PM in the hot pool and is in proportion to λ_t) and N_h. φ_h is the rate at which a VM can be deployed on a PM in hot pool and μ is the service rate of each VM.The total service rate for each PM is the product of number of running VMs by μ. The maximum number of VMs on a PM is equal to m and in this model, it is set to 10. The virtual machine provisioning module calculates the success probabilities (P_h and P_c) that at least one PM in a given pool (hot and cold, respectively) can be assigned to a task. Success probabilities are, then, used as input parameters for the resource allocation module and the other virtual machine provisioning module, as well as for the pool management module described below. In addition, the hot PM model provides the probability by which a hot PM becomes idle, P_i. An analogous CTMC, but with different transition rates, can be drawn for the corresponding module managing the cold pool.

Also, pool management module computes N_h and N_c as input for hot and cold PM sub-models respectively.

C. Pool management

As noted above, a cold PM is switched on and populated with the required number of VMs when there is no capacity in the hot pool to satisfy user requests. Conversely, if a hot PM is idle for a predefined time, it is switched off. However, the number of PMs is never below a predefined threshold in order to maintain acceptable performance. The management of PM pools is done by a separate module, the operation of which can be described by a two dimensional CTMC shown in Fig. 3. The system starts from the state (i, j) which indicates

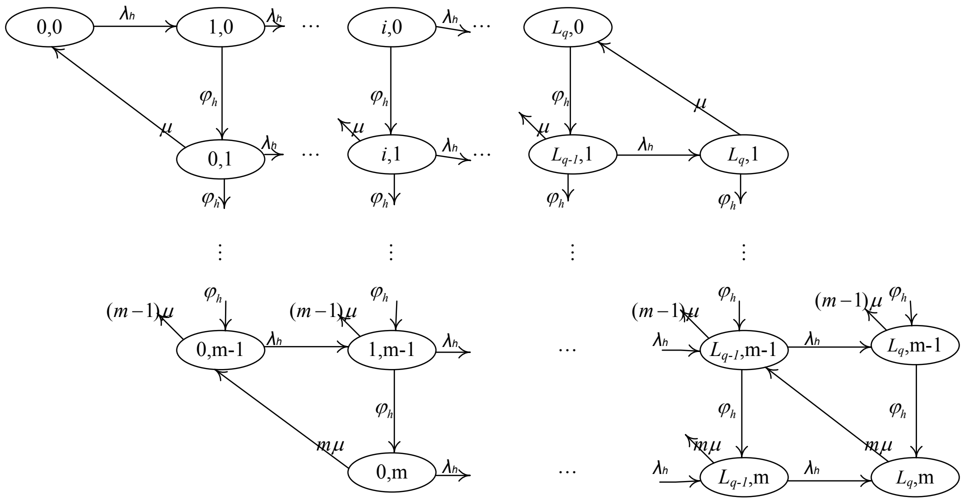


Fig. 2. Virtual machine provisioning in the hot pool.

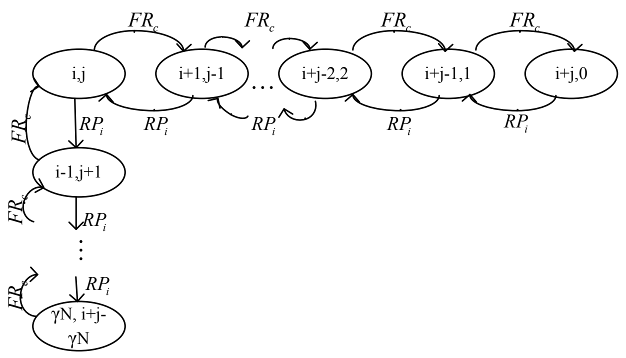


Fig. 3. Pool management.

that i and j PMs are in the hot and cold pool, respectively. If next search in the hot pool is successful (with probability of P_h), system will remain in the starting state, otherwise one of the cold PMs will be moved into the hot pool. This transition occurs with the rate of FR_c. With higher demand of PMs, cold PMs can become hot PMs gradually until all the PMs are hot. Also, due to the low demand of resources, idle hot PMs will be turned off gradually, but the number of hot PMs cannot get less than γN. The idle hot PM is moved to the cold pool by rate of RP_i.

D. Solving the integrated model

Our overall model thus consists of three inter-connected stochastic models which compute the cloud performance parameters such as task blocking probability and energy consumption. However, there is a cyclic inter-dependency among the modules, which can be resolved using fixed-point iterative method [7] through a modified version of successive substitution approach presented in Algorithm 1.

Algorithm 1 Iterative Algorithm for Substitution

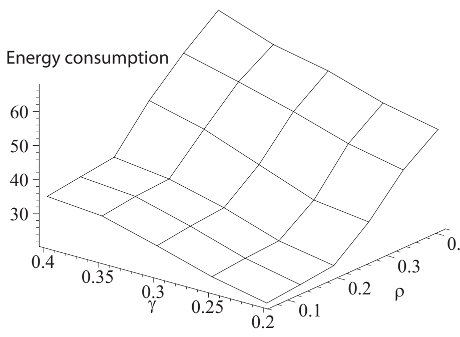
```
Input: Starting success probabilities in pools: $P_{h0}$, $P_{c0}$;
Input: Starting idle probability of a hot PM: $P_{i0}$;
Output: First level of probability of blocking: $P_{bq}$;
count $\longleftarrow$ 0; maximum $\longleftarrow$ 30; $\Delta \longleftarrow 1$;
$P_{bq0} \longleftarrow (Ph0, Pc0)$ (from resource allocation);
$[N_h, N_c] \longleftarrow (P_{h0}, P_{i0})$ (from pool management)
while $\Delta \geq 10^{-6}$ do
  count $\longleftarrow$ count +1;
  $[P_h, P_i] \longleftarrow (P_{bq0}, N_h)$ (from hot VM provisioning);
  $P_c \longleftarrow (P_{bq0}, P_h, N_c)$ (from cold VM provisioning);
  $[N_h, N_c] \longleftarrow (P_h, P_i)$ (from pool management)
  $P_{bq1} \longleftarrow (P_h, P_c)$ (from resource allocation);
  $\Delta \longleftarrow |(P_{bq1} - P_{bq0})|$;
  $P_{bq0} \longleftarrow P_{bq1}$;
  if count = maximum then
    break;
  end if
end while
if count = maximum then
  return -1;
else
  return $P_{bq0}$;
end if
```

IV. Performance Evaluation

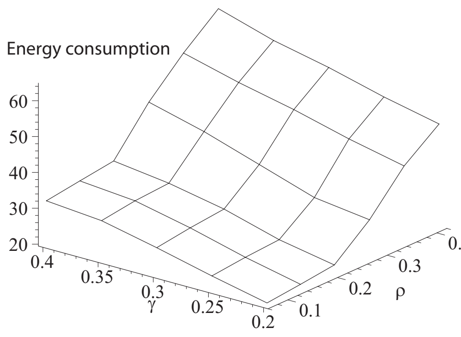
In our experiments, the total number of available PMs is maintained at a constant value of $N = 100$. The threshold coefficient γ determines the initial partitioning of the PMs between hot and cold pools, with γN PMs in the hot pool and $(1 - \gamma)N$ PMs in the cold pool. As noted above, the number of PMs in the hot pool can never drop below γN.

Due to space limitations, we do not show the actual equations describing the model, as they closely follow the analysis presented in [5]. Instead, we show just the results obtained from the solution of the model.

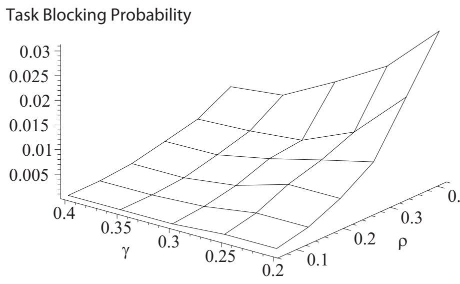
The response time or total delay is the sum of four provisioning times: waiting in the global queue, processing in the resource allocation module, waiting time in the PM queue, and



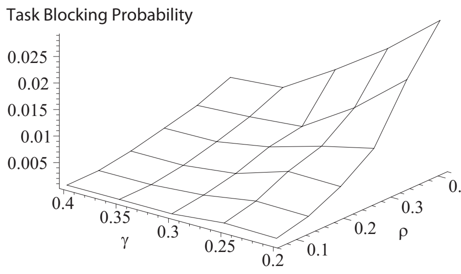
(a) Energy consumption, mean look up time in the cold pool 12 seconds.



(b) Energy consumption, mean look up time in the cold pool 7.2 seconds.



(c) Task blocking probability, mean look up time in the cold pool 12 seconds.



(d) Task blocking probability, mean look up time in cold pool 7.2 seconds.

Fig. 4. Performance of cloud data center at mean service time 40 minutes, mean look up time in the hot pool 18 seconds, task arrival rate variable from 100 to 600 per hour.

VM instantiation and deployment, to which the actual service time should be added [3], [4], [5]. We note that:

- Mean waiting time in global queue ($\overline{wt}$) is exponentially distributed with mean value of task arrival rate ($1/\lambda_t$).
- Mean look up time between the pools ($\overline{lut}$) has a Coxian distribution with two steps of look-up time between hot and cold pools [3].
- Mean waiting time in the input queue of the select PM ($\overline{PM_{wt}}$) is exponentially distributed for each pool with mean values of arrival rate ($1/\lambda_h$ and $1/\lambda_c$ that are in proportion to $1/\lambda_t$).
- Mean waiting time for VM provisioning ($\overline{pt}$) is exponentially distributed with mean value of task arrival rate ($1/\lambda_t$).

A. Cloud system behavior at low loads

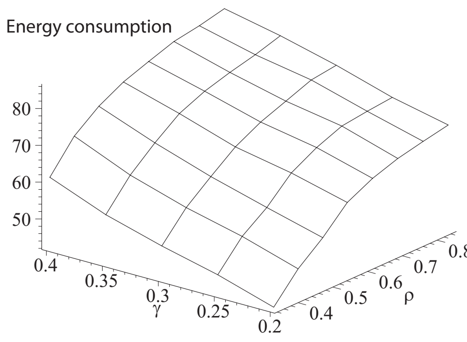
In our first experiment, we have investigated the impact of system load and pool threshold on energy consumption and task blocking, under the conditions of low system load (values of ρ up to 0.4). Mean task service time is assumed to be constant at 40 minutes, so that the adjustment of system load is achieved through the adjustment of task arrival rate. Task rejection probability is obtained from the model. Energy consumption is calculated by considering the mean time $\overline{T_{st}}$ spent in each state of the Markov chain in Fig. 3 and power consumption δ_p of a hot machine, as

$$E_c = \sum_{s \in \xi} N_{h_s} \, \delta_p \, \overline{T_{st}} \tag{1}$$

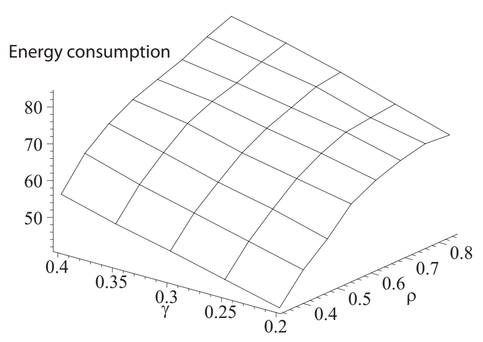
where ξ is the set of states of the Markov chain from Fig. 3, while N_{h_s} is the number of PMs in hot pool in state s. In this paper, δ_p is assumed to be 1, as all PMs are assumed to be homogeneous; the resulting values of energy expenditure may be interpreted as 'the mean number of PMs that are on.' The resulting diagrams are shown in Fig. 4.

As can be seen, energy consumption, shown in Figs. 4(a) and 4(b), is approximately linearly dependent on the pool threshold, as could be expected. However, it is initially nearly independent of the system load, up about $\rho = 0.2$, but then increases rapidly. This is caused by the fact that in the first part of the range of values for ρ, the current number of PMs in the hot pool suffices to service the incoming task requests. However, the increase of the load in second part of the range of values for ρ means that more PMs are switched on (i.e., moved into the hot pool), with the associated increase in energy consumption.

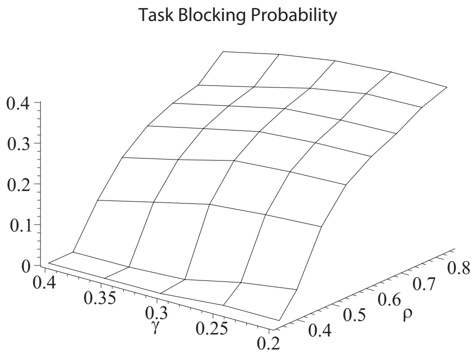
At the same time, the blocking probability in Figs. 4(c) and 4(d) is comparatively low, well below 1% in the most part of the observed range, but it does rise when the system load increases. Also, blocking probability remains low as long as the pool threshold is kept near the maximum value of $\gamma = 0.4$, but it does increase at lower values of γ. The rationale for such behavior is simple: when the pool threshold is low, most PMs are kept in the cold pool (i.e., they are switched off), and they are brought in to the hot pool only when there is a need. However, this action incurs a certain overhead which may cause some tasks to be rejected. Keeping more PMs in



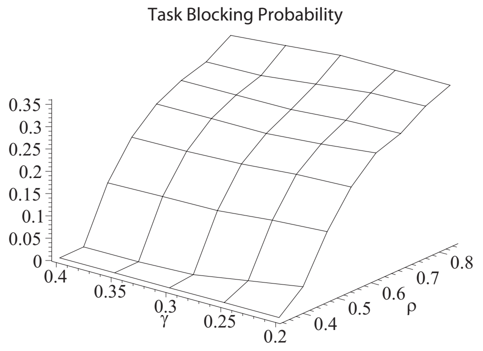
(a) Energy consumption, mean look up time in the cold pool 6 seconds.



(b) Energy consumption, mean look up time in the cold pool 4.5 seconds.



(c) Task blocking probability, mean look up time in the cold pool 6 seconds.



(d) Task blocking probability, mean look up time in cold pool 4.5 seconds.

Fig. 5. Performance of cloud data center at mean service time 40 minutes, mean look up time in the hot pool 7.2 seconds, task arrival rate variable from 500 to 1200 per hour.

the hot pool helps reduce the number of rejected tasks and keeps the blocking probability low; since the mechanism of moving the PMs between pools is continuously adaptive to the instantaneous load, the increase in blocking probability is gradual and does not exhibit a distinct threshold such as the one observed in the diagrams of energy consumption.

We have also used two values for the mean look up time in the cold pool which correspond to the mean look up rates of 300 (diagrams on the left) and 500 searches per hour (diagrams on the right). The mean look up time in the hot pool was kept constant at 200 searches per hour. We note that faster search reduces both the energy consumption and the task rejection probability, although the differences are not significant in either case.

B. *Cloud system behavior at high loads*

We have repeated the first experiment but with the range of system loads that is wider and extends well into higher loads, up to $\rho = 0.8$; the mean look up times have been reduced in order to make the system more responsive. As before, the changes in system load were accomplished by changing the mean task arrival rate while the mean task service time was kept constant. The results are shown in Fig. 5.

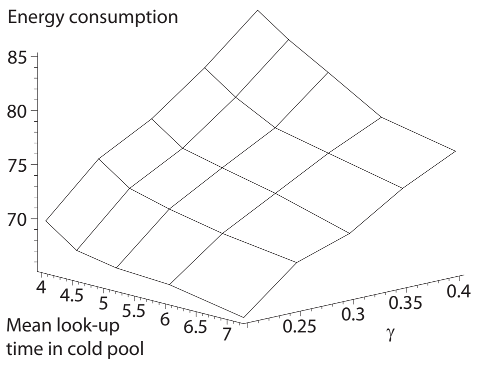
As can be seen, the higher values of system load lead to considerable increase in energy consumption as well as in task blocking. As the task arrival rates are higher, a distinct threshold may be observed in the diagrams for task blocking probability, Figs. 5(c) and 5(c). The corresponding thresholds in the diagrams for energy consumption can't be seen – in fact, they would be visible if the range of system loads were extended downward towards values of $\rho < 0.2$.

But in either case, higher system load leads to higher energy consumption and considerably higher task rejection rate. The values seem to flatten at high system loads, but only because the system has effectively entered saturation, with a large number of incoming tasks rejected (values are well over 10%). Operation in this regime is most likely unacceptable in practice and should be avoided.

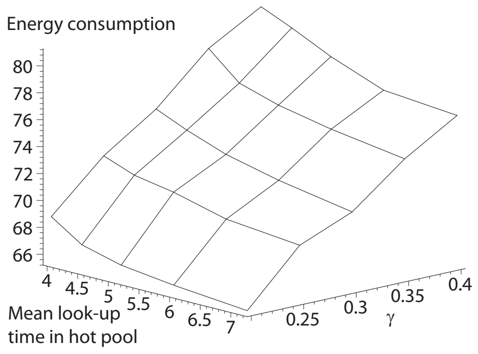
As before, shorter mean look up times (i.e., faster searches) give a slight improvement in performance. This has motivated us to conduct the experiments described in the next Subsection.

C. *The impact of look up times on energy consumption*

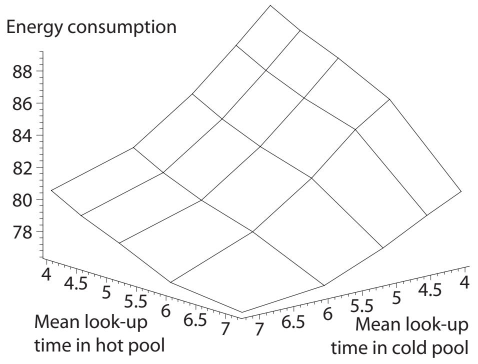
Finally, we have investigated the impact of look up times on energy consumption; the resulting diagrams are shown in Fig. 6. As can be seen, lower values of the mean look up time in the cold pool results in higher energy consumption which is due to faster searches and quicker bringing up of new PMs into the hot pool. The increase is somewhat higher than in the case of corresponding mean look up time in the hot pool,



(a) Energy consumption as the function of mean look up rate in the cold pool; mean look up rate in the hot pool constant at 7.2 seconds.



(b) Energy consumption as the function of mean look up rate in the hot pool; mean look up rate in the cold pool constant at 7.2 seconds.



(c) Energy consumption as the function of mean look up rates in the hot and cold pools; pool threshold kept at a constant value of $\gamma = 0.4$.

Fig. 6. The impact of look up times on energy consumption. Mean task arrival rate 1000 per hour, mean task service time 40 minutes.

which is caused by the additional overhead needed to turn on a cold PM. The difference in impact between the mean look up times in the cold and hot pools is confirmed in Fig. 6(c).

V. Conclusion

We have analysed the effects of pool threshold and mean look up time in hot and cold pools on the performance of an IaaS cloud system where PMs are partitioned into a hot and a cold pool. Our results confirm that the system should operate well below saturation in order to provide acceptably low task rejection probability and energy consumption. Also, we have shown that the performance is more sensitive to the mean look up time for the PMs in the cold pool.

In our future work, we plan to investigate the performance of the networked system as well as of the system with migration of live VMs between servers (PMs).

References

[1] 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.

[2] 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.

[3] 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.

[4] H. Khazaei, J. Mišić and V.B. Mišić. A Fine-Grained Performance Model of Cloud Computing Centers. In *IEEE Transaction on Parallel and Distributed Systems*, 24(11):2138–2147, Nov 2013.

[5] H. Khojasteh, J. Mišić and V.B. Mišić. Characterizing energy consumption of IaaS clouds in non-saturated operation. submitted to IEEE INFOCOM 2014 Workshop on Mobile Cloud Computing, Toronto, Canada, April-May 2014.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.