Impact of Instance Seeking Strategies on Resource Allocation in Cloud Data Centers

Hao Zhuang[1], Xin Liu[1], Zhonghong Ou[2], Karl Aberer[1]
[1] *École Polytechnique Fédérale de Lausanne, Switzerland*
[2]*Aalto University, Finland*
hao.zhuang@epfl.ch, x.liu@epfl.ch, zhonghong.ou@aalto.fi, karl.aberer@epfl.ch

***Abstract*—With the prosperity of cloud computing, an increasing number of Small and Medium-sized Enterprises (SMEs) move their business to public clouds such as Amazon EC2. To help tenants deploy services in the cloud, researchers either conduct performance evaluations or design mechanisms and software on seeking virtual machines of better performance. However, few studies have investigated the impact of instance seeking strategies on resource allocation in clouds if every tenant starts to apply the same method to find the better-performing virtual machine. In this paper, we propose a cloud and a tenant model in order to simulate the process of tenants' seeking better-performing instances in the cloud. We discuss, implement and evaluate six cloud resource allocation strategies and five instance seeking strategies. We perform the evaluation via simulation based on real data traces. Our results show that instance seeking strategies can cause the exhaustion of better-performing instances and significant request growth in the cloud. Furthermore, we find that tenants could save time and budget through collaborative seeking strategies. Finally, we discuss the implications of our findings from perspectives of both tenants and providers.**

***Keywords*-instance seeking strategy; resource allocation; performance heterogeneity;**

I. Introduction

Cloud computing has been popular in the past few years, ranging from industry to academia. In industry, a number of leading cloud providers, such as Amazon, Microsoft and Google, have built their cloud data centers to provide scalable and pay-as-you-go compute capacity. Cloud tenants rent virtual machines (VMs) according to their needs. With the mounting demands from tenants, it is inevitable for cloud providers to add new servers to their clouds due to the business expansion and hardware upgrade. Consequently, cloud data center becomes hardware heterogeneous, which results in performance variations. For example, Amazon uses EC2 Compute Unit (ECU), which is defined as *one ECU equals the CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor* [1], to evaluate the CPU capacity of VMs. It means that Amazon EC2 is a heterogeneous computing system with different CPU processors (Opteron and Xeon) and the performance variation of CPU capacity is about 20% (1.0 to 1.2 GHz). Thus, for cloud tenants, they always prefer to rent the instances[1] with faster CPU to achieve the best price-performance ratio.

Meanwhile, in academia, recent research suggests that there exist great performance variations of instances in cloud data centers. Our prior work [2], which was also reported by BBC, confirmed that hardware heterogeneity is the primary culprit for great performance variations. The variation of the same sub-type of instances, i.e. hosted by identical hardware, is relatively small, whilst the variation among different sub-types of instances, i.e. hosted by heterogeneous hardware, could reach up to 60%. [3] also found out that the performance heterogeneity exists in Amazon EC2 standard small instance. Also, existing works either give their suggestions [4] or design various mechanisms [2], [3] and software [5] for cloud tenants to find better-performing instances in public clouds. We define these mechanisms and software of detecting better-performing instances as *detection techniques*.

In this paper, however, we will challenge the effectiveness of these detection techniques if every tenant starts to use the same detection technique to seek better-performing instances. Cloud data center should support thousands of concurrent tenants. With the increasing number of tenants seeking better-performing instances, the probability of finding a better-performing instances becomes increasingly smaller. Thus, an immediate question is that "how could the tenant find better instances from the cloud with the acceptable time and cost?". Meanwhile, it would not be difficult for cloud providers to detect tenants' behaviors. Would they either prohibit it or change their resource allocation strategies? In this paper, instead of designing the methods or software on how to distinguish better instances from worse ones, we shift our focus to study the impact of the widespread use of these detection techniques on cloud resource allocation. To that end, five *instance seeking strategies*, which are the ways to request instances from the cloud, are designed for tenants when they use detection techniques. Our main contributions can be summarized as follows. (1) We propose a cloud model and a tenant model to evaluate the impact of instance seeking strategies based on real cluster data traces. (2) We find that the performances of instance seeking strategies differ widely. The *greedy* strategy outperforms the *moderate* which depletes the better-performing instances quickly. Moreover, cloud tenants could save more cost and time through the

[1] Hereafter, we will use VM and instance interchangeably

collaborative strategy. (3) Resource allocation strategies in the cloud exert a strong influence on the tenant requests' growth which is between 2 times to 42 times what is expected. The soaring request growth results in a heavy price on tenants seeking cost which may go up by 31 times.

The rest of the paper is structured as follows. In Section II, we present related literature. Section III describes our cloud data center and tenant models. Section VI details our simulation methods and evaluations, followed by a discussion of our findings in Section V. Section VI concludes the paper.

II. Related Work

Our work is related to existing research efforts in the following three aspects:

Performance heterogeneity: Wang *et al.* [4] presented a measurement study on the impact of virtualization on Amazon EC2 platform. Their findings indicated that virtualization causes instability and variation to network throughput and packet delay. Our prior work [2] further confirmed that hardware heterogeneity is the primary culprit for great performance variations. Farley *et al.* [3] also found out that hardware heterogeneity exists in Amazon EC2 standard small (m1.small) instance, which results in performance variations.

Detection techniques: Li *et al.* [6] developed a performance and cost comparator, i.e. CloudCmp, to measure cloud services from different cloud providers. Their study demonstrated that there was no single winner who outperformed the other counterparts in all aspects of its cloud service offerings. [3] also formulated the customer-controlled placement gaming as a multi-armed bandit problem. [5] proposed ClouDiA, a software that helps the cloud users select server nodes with the lowest latency in the cloud, to minimize the latency between the servers, thereby improving the performance for latency-sensitive applications. However, they do not consider how cloud providers will react to tenants detection behaviors, which makes our work different.

Resource allocation policies and cloud simulation: [7] compared different VM allocation algorithms in the infrastructure clouds. We designed our VM allocation strategies inspired from this paper. Lee *et al.* [8] introduced a scheduling mechanism in the cloud that takes heterogeneity of the underlying platform and workloads into consideration . [9] analyzed the Google cluster trace data from a sizable multi-purpose cluster. We conducted our simulation with the real job statistics extracted from the Google cluster trace data. Rodrigo *et al.* [10] presented CloudSim to evaluate the performance of cloud provisioning policies, application workload models, and resources performance models. We developed our simulator based on CloudSim.

Our prior work [2] proposed a simple trial-and-error mechanism to achieve cost-savings based on hardware heterogeneity. In this paper, we focus on validating the effectiveness of these detection techniques [2], [3], [5] through designing instance seeking and resource allocation strategies from the perspectives of both tenants and cloud providers. To the best of our knowledge, there is no work focusing on instance seeking strategies of better instances in public clouds, which motivates our work in this paper.

III. Models

A. Cloud Data Center

we assume a cloud data center model, which is similar to the Amazon EC2, goes through several hardware upgrades due to the business expansion. Different generations of hardware co-exist in the cloud. Based on our prior work, the performance of instances hosted on the latest hardware are better than that of old hardware. In other words, cloud tenants could find better instances through checking the hardware information. Cloud data center provides several types of VMs with different capacities, which are suitable for various workloads. We only focus on the same type of instances with the same specifications of CPU, memory, network, disk and etc. All the instances are charged by hour and instances of the same type are charged with the same price c.

For the purpose of load balance and management, physical machines are organized in terms of *cluster*. The cloud data center consists of several clusters and each cluster has different quantities of physical machines. After receiving tenants' requests, the cloud data center will find and return the VMs to the tenants in the light of VM allocation strategies. Tenants can request instances at any time but each tenant can apply for only M instances at most. In Amazon EC2, M is 20. There are latencies for both requesting and terminating the instances.

1) Host Distribution: similar to Amazon EC2 [2], we design three kinds of host servers with different CPU types, namely E5430, E5507 and E5645 from Intel Xeon Processor 5000 Sequence[11]. Therein, E5645 is the most powerful and the latest hardware. In our study, we model a data center with three clusters. Furthermore, a discrete distribution ψ is defined to determine the distribution of each host in a cluster.

$$\psi = \{\mathrm{E}_i\text{: } \sum_i \mathrm{Pr}(\mathrm{E}_i)\text{=1, i=1..3}\}$$

where E_1, E_2, E_3 represents the distribution of E5430, E5507 and E5645 respectively. We study the effect of varying ψ. In particular, we analyze three representative distributions (1) *Minority* ψ_{min} = {0.3, 0.3, 0.4}, (2) *Majority* ψ_{maj} = {0.2, 0.2, 0.6} (3) *Dominant* ψ_{dom} = {0.1, 0.1, 0.8}. For example, Majority distribution means the latest hardware E5645 takes up as much as 60% in a cluster.

2) VM Resource Allocation Strategies: VM resource allocation is the process of mapping the virtual server onto physical host machines. The allocation policies of the commercial cloud are shadowed by their proprietary nature. However, we can be inspired from the prior studies [7] as well as from opensource cloud platform. OpenStack [12], for example, employs various filters to select hosts by computing capacities, CPU cores, RAM and image properties. After receiving requests from tenants, the data center controller will firstly apply these filters to select eligible hosts. After filtering, the scheduler selects the host that has the minimum weighted cost and returns it to end users. Virtualization also provides a VM migration mechanism to remap the virtual machines onto a less-loaded server after allocation. However, we regard VM migration as a costly operation and do not consider the migration.

In this paper, we design a two-level resource allocation strategies $\phi_{host}^{cluster}$ [7]. For each tenant's request, the data center controller should decide which cluster and which host to run the VM. For the cluster level, we have three allocation strategies: (1) *Random (rand)* is to select the cluster randomly; (2) *Round Robin (rr)* selects the cluster in circular order; (3) *Least Full First (lff)* is to select the cluster with the minimal utilization. For the host level, two allocation strategies are provided (1) *Random (rand)* returns the host in a random way; (2) *Max Core First (mcf)* chooses the first node with maximal available computing cores. Combining both cluster and host levels, we have six VM resource allocation strategies:

$$\phi = \{\phi_{rand}^{rand}, \phi_{rand}^{rr}, \phi_{rand}^{lff}, \phi_{mcf}^{rand}, \phi_{mcf}^{rr}, \phi_{mcf}^{lff}\}$$

where ϕ_{rand}^{rand} is similar to the Chance scheduler [12] in OpenStack; Round Robin ensures the fairness of resource allocation; lff-based allocation considers the load balance during the resource allocation. Since we do not focus on resource allocation, other factors such as energy and cost efficiency are not considered.

3) Instance Performance Level:

Definition 1: **(Instance Performance Level)**. We assume the performance of each instance can be measured by detection technique. Let X be the set of the instance performance levels which is normalized to that of the worst performing instance and the instance performance level is defined as X = $\{x_i\}$.

Note that instances of the same type are charged with the same cost per hour, although they present different performance levels. Furthermore, we classify the instances running on different hosts into two categories by assigning a threshold value x_{thres}. If the performance level of a instance is greater than x_{thres}, we regard it as a *better-performing* instance. Otherwise, it is regarded as a *worse-performing* one. Tenants could define x_{thres} according to their detection techniques. Let B be the set of better-performing instance and W be the set of worse-performing. Thus, a data center can be defined as DC=$\{B,W,\psi\}$. It follows immediately that the total number of instances hosted by a cloud at time t is $N_t = |B_t| + |W_t|$. Furthermore, the probability of a tenant to get a better-performing instance is denoted as $p_t = \frac{|B_t|}{N_t}$.

B. Cloud Tenants

1) Job Modeling: tenants submit their requests to the cloud, indicating how many instances they will request and how long they will keep the instances. The duration and the number of instances depend on the characteristics of tenants jobs. To make our evaluation as realistic as possible, we extract the real job data from the Google Cluster traces [13] which provide trace data from 12 thousand machines over about a month-long period in May 2011. The distribution of job durations is similar to the heavy-tailed distribution shape [9]. The job duration ranges from tens of seconds to essentially the entire duration of the trace (about 700 hours). In our model, we define tenant jobs as a tuple $j_i = \langle q_i, d_i \rangle$, where the q_i is the number of instance and d_i is the duration of job runtime.

2) Instance Seeking Problem: We define Γ as the set of tenant τ_i :

$$\tau_i = \langle j_i,\ S_i \rangle$$

where S_i is the set of assigned instances s_i^j $(1 \leq j \leq q_i)$ from the cloud to execute the job $j_i = \langle q_i, d_i \rangle$. Naturally, the assigned instance set S_i consists a mixture of better-performing and worse-performing instances each with certain percentage.

We assume tenants submitting their requests follows a Poisson distribution at a request rate λ. Then, we can give the definition of a tenant to seek better-performing instances in cloud data center.

Definition 2: **(Instance Seeking Problem)**. Given a cloud data center DC=$\{B,W,\psi\}$ with resource allocation strategies ϕ and a group of tenants $\Gamma = \{\tau_i\}$ with seeking strategies Ψ, the instance seeking problem for each τ_i = $\langle j_i, S_i \rangle$ is to find each assigned instances set $S_i \subseteq$ B.
where we will further discuss instance seeking strategies Ψ.

3) Instance Seeking Strategies: to solve the instance seeking problem, different tenants adopt various seeking strategies. We divide tenants into two categories: *performance-oriented* and *budget-constraints*. For performance-oriented tenants, their applications need to run a really long time (maybe "forever"). Thus, they give the top priority to select instances of better performance from the cloud. In comparison with performance-oriented tenants with sufficient budget, budget-constrained ones prefer to make a compromise between the instance performance and the limited budget. For budget-constrained tenants and tenants who are unaware of instance performance variations, we provide the *normal* strategy, while we design three other types of strategies for performance-oriented tenants, namely *moderate, greedy* and *collaborative*.

Note that each tenant can only rent M instances at most. We suppose a tenant τ_i will request q_i ($q_i \leq$M) instances from the cloud and propose four strategies for τ_i.

i) ***Normal*** strategy Ψ_n is not targeted to solve the instance seeking problem. However, it is widely used by the most of the tenants who are not concerned with performance variations. They request the number of instances according to their demands and free instances immediately after the job completes.

ii) ***Moderate*** strategy Ψ_m is a three-step iterative process to acquire the desired number of better-performing instance: (1) request for the desired number q$_i$ of instances from the cloud; (2) sort the acquired instances by performance levels x$_i$; (3) keep the better-performing instances, terminate the worse ones immediately, and then apply for new instances from the cloud. Through iterating the procedure for multiple rounds, the tenant will eventually get the desired number of better-performing instances.

Algorithm 1 Greedy Algorithm

```
Require: $q_i$ : number of instances will request
Ensure: $S_i \subseteq B$ : assigned instances are all better ones
$S_i \leftarrow \emptyset$;
$request\_num = greedy(q_i)$;
while $|S_i| < q_i$ do
    $w_i \leftarrow requestInstance(request\_num)$;
    $S_i \leftarrow checkPerformanceLevel(w_i, x_{thres})$;
    $terminate(w_i, free\_latency)$
    if $|S_i| \geq q_i$ then
        $terminate(S_i, |S_i| - q_i, free\_latency)$
        $break$;
    else
        $request\_num = greedy(q_i - |S_i|)$
    end if
end while
return $S_i$
```

iii) ***Greedy*** strategy Ψ_g requests instances from the cloud in a greedy way. In this paper, we make a hybrid piecewise greedy policy according to the desired instance number.

$$greedy(q_i) = \begin{cases} q_i + \alpha, & 1 \leq q_i \leq \frac{M}{4} \\ q_i * \beta, & \frac{M}{4} + 1 \leq q_i \leq \frac{M}{2} \\ M, & q_i \geq \frac{M}{2} + 1 \end{cases}$$

where the value of α, β are tunable in an addition and multiplication way by the tenant. Algorithm 1 shows how tenants seek better-performing instances with the greedy strategy. For each tenant, they will apply the greedy policy according to their desired instance number firstly. In each iteration, all the requested instances are put in the set of worse-performing w$_i$. After checking the performance level in w$_i$, the better-performing instance will be removed from the w$_i$ and put into the S$_i$. The left worse-performing instances in w$_i$ will be terminated within a free latency. However, the time spent on the *checkPerformanceLevel* procedure depends on the detection techniques. In our implementation, we only check the hardware information of VM and then sleep for 100 seconds which we define as *tenant seeking time*. Finally, it will be decided that how many instances still need to be applied in the next round. If the number of better-performing instances is larger than the demand, it will terminate the extra better instances. However, the tenant could tune the value of α, β to decrease the number of extra instances.

iv) ***Collaborative*** strategy Ψ_c assumes not all the tenants are selfish and they could also cooperate. Ψ_c makes a difference on the way of tenants terminating the instance. The price model of cloud data center is charged per hour. Thus, if the tenant terminate the worse-performing instance at the end of the first hour rather than immediately, the number of worse-performing instance ($|B_t|$) will decrease at time t. Consequently, the probability of other tenants to acquire the better instance will increase. The cloud tenants could use Ψ_c strategy together with either Ψ_m or Ψ_g strategy. Thus, we have five tenant strategies:

$$\Psi = \{\Psi_n, \Psi_m, \Psi_g, \Psi_{cm}, \Psi_{cg}\}$$

Both Ψ_m and Ψ_g focus on the tenant request strategies, while Ψ_c pays attention to the tenant termination strategies. Thus, to combine the request- and terminate-strategy, we have two more strategies, Ψ_{cm} and Ψ_{cg}, based on Ψ_m and Ψ_g respectively.

IV. Evaluation

A. Simulator

To evaluate the impact of tenant strategies, we developed our simulator based on the CloudSim[10] which is a simulation toolkit that enables modeling and simulation of cloud computing systems based on discrete events. There are four layers in our model, namely Cloud, Cluster, VM and Broker. The Cloud layer provides Amazon EC2-like APIs for tenants to start and to terminate the instances. Cluster layer is responsible for cluster profiling and scheduling. It collects the statistical information of each cluster and returns the cluster in terms of cluster-level allocation policy. VM layer manages all physical machines within a cluster. It collects the utilization information (e.g. CPU, memory, disk, etc.) of each physical machine and filters machines by utilization information. In Broker layer, we implement five types of brokers on behalf of tenants with different strategies. Tenant brokers interact with the cloud through the APIs provided by Cloud layer.

B. Experiment Setup

We set up a data center with three clusters. The configuration of the data center consists of data center, cluster, VM and host level which are listed in table I. The data center level configures the maximum instance which a user can request, price, VM allocation strategy, request and free latency. For each cluster, we have a total of 1500 instances each with different host distribution ψ. The number of better-performing instances of the cluster with ψ_{dom} is twice as much as that of the cluster with ψ_{min}. The data center

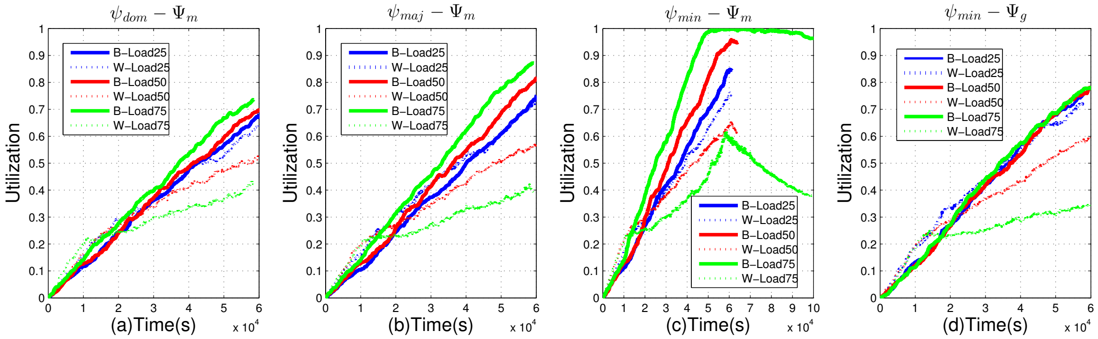

Figure 1. Utilization of better- and worse-performing instance in different clusters

provides tenants with a large VM instance which has 2 CPU cores, 3.7GB memory and 400GB disk. From the host configurations in table I, it can be seen that both E5430 and E5507 could host two VMs while E5645 could host three instances.

Level	Notation	Semantics(default value)
Data Center	M c ϕ Req_Latency Free_Latency	the maximum instance(20) cost in $ per instance per hour(0.3) VM allocation strategy (ϕ_{rand}^{rand}) request latency in seconds(80) free latency in seconds(60)
Cluster	N ψ	total instance in a cluster(1500) host distribution for each cluster ($\{\psi_{dom}$,ψ_{maj},$\psi_{min}\}$)
VM	<CPU,Mem,Disk>	VM instance with 2 cores, 3.7GB memory and 400GB disk
Host	E5430 E5507 E5645	4 CPU cores,8GB memory,1TB disk 4 CPU cores,8GB memory,1TB disk 6 CPU cores,16GB memory,1.6TB disk

Table I
DATA CENTER CONFIGURATION

C. *Performance Metrics*

Two metrics are defined for instance seeking problem: *seeking latency* and *seeking cost*. Seeking latency is the time which a tenant spends on solving the instance seeking problem whilst the seeking cost is the cost of requesting extra instances during the seeking latency. Usually, the running time of extra instances is less than one hour but it is billed as a full hour. From the perspective of cloud, we observe the *utilization* of better instances and worse instances in each cluster and also monitor the *request throughput*, which is the number of requests processed per unit of time in the cloud, to ensure the load balance in the cloud.

D. *Impact of Instance Seeking Strategy*

In this part, we keep the default VM resource allocation ϕ_{rand}^{rand} unchanged and vary instance seeking strategies Ψ. We simulate a group of 1000 tenants to request the instances from the cloud with a learning process, in which the first 200 tenants are unaware of seeking better instances so that they adopt Ψ_n strategy. Afterwards, we define three tenant loads: (1) Load75, (2) Load50, (3) Load25. The number is the percentage of performance-oriented tenants. Take Load75 for example, first 200 tenants use strategy Ψ_n whilst 75% of the following 800 tenants (that is 600) use one of the other strategies to seek better instances. The request arrival rate of tenants λ is 60 which means there is a request about every minute.

1) Resource Utilization: We run three kinds of loads with varying the Ψ. The utilization of both better- and worse-performing instances in three clusters are shown in the Fig. 1. Be noted that B-Load25 means the utilization of Better-performing instances with Load25 while W means the utilization of worse-performing instances. Several findings can be made from Fig. 1:

i) During the learning process (about first 3 hours), the increase rates of both B and W are similar in all three clusters since the first 200 tenants use Ψ_n. After that, with following tenants starting to seek better instances, the increase rate of W utilization has been significantly reduced while that of B keeps the stable growth. Also, the heavier the load is, the higher the utilization of B is but the lower the utilization of W is.

ii) From Fig. 1 (c), the utilization of B in the cluster with the distribution of ψ_{min} fluctuates at as high as 100% under the Load75, which means almost all the requests for better instances dispatched to the cluster ψ_{min} will no longer get better instances. The status of a cluster with depletion of all better instances is defined as *exhaustive*. Fig. 1 (a) and (b) show that the utilization of B in the clusters ψ_{dom} and ψ_{maj} are not much affected by the heavier workloads while the cluster ψ_{min} is exhaustive. When the cluster is in exhaustive status, tenants will spend more time on seeking better instances. Compared with the clusters ψ_{dom} and ψ_{maj}, the whole completion time (shown in x-axis) of all tenants requests rises 66.67% in the ψ_{min} cluster. However, for cloud providers, the cluster in exhaustive status still has a large proportion of worse-performing instances available. The scheduler will keep dispatching requests to this cluster which results in a sudden request growth. We will discuss the request throughput in the next subsection.

iii) Fig. 1 (d) shows the utilization in the cluster ψ_{min} with tenants using Ψ_g strategy. We set $\alpha = 3$ and $\beta = 2$ in the greedy policy. It is clearly seen that the cluster is not exhaustive with the utilization of B reaching at about 80%. Hence, from a tenant's point of view, the strategy Ψ_g outperforms Ψ_m under the same host distribution ψ, especially in the cluster which has a minority of better instances.

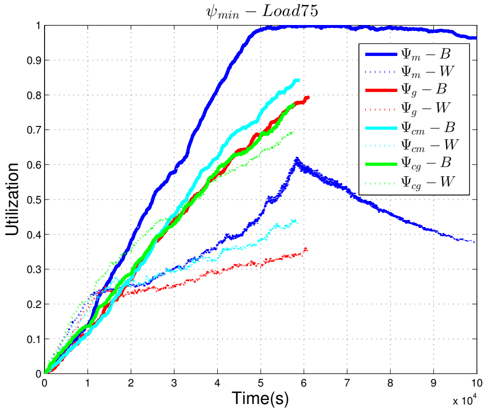

Figure 2. Utilization of ψ_{min} cluster with varying seeking strategy

To further exploit the impact of tenant strategies Ψ, we run the heaviest Load75 by varying Ψ. Fig. 2 shows the results in the cluster ψ_{min}. It is demonstrated that only tenants with Ψ_m strategy make the cluster ψ_{min} exhaustive quickly while those with other three strategies could finish the simulation in time. Tenants are in favor of that their requests are processed in the cluster with the low utilization of B, which increases the probability to acquire better instances. From this point, the collaborative strategies Ψ_{cm} and Ψ_{cg} outperforms the strategies without collaboration. Ψ_{cg} shows the similar utilization of B with Ψ_g but gives rise to high utilization of W which is almost twice that of strategies Ψ_g. This is because collaborative strategies do not free worse-performing instances immediately.

2) Seeking latency and cost: Fig. 3 depicts the cumulative distribution of the seeking latency and cost for all tenants in the cluster ψ_{min}. However, since the Load75 with Ψ_m strategy brings about exhaustion to the cluster, we select the Load50 with Ψ_m strategy to compare with other strategies. From Fig. 3, we can see although the strategy Ψ_m with less heavy workload (Load50), both of its latency and cost are the highest. For the seeking latency, the collaborative strategies Ψ_{cm} and Ψ_{cg} are better than those which are selfish. Also, the greedy-based strategies (Ψ_{cg}, Ψ_g) outperforms moderate ones (Ψ_{cm}, Ψ_m). Except tenants with Ψ_m, almost all of them could solve the instance seeking problem within half an hour. As for the seeking cost, we find similar trends to the seeking latency. The seeking cost of about 90% tenants is less than 10 dollars, which makes it affordable for both individuals and SMEs. Through the Fig. 3, we find Ψ_{cg} is the best strategy for tenants in terms of both seeking latency and cost. Also, we find the collaboration among tenants could save a lot of time and money on seeking better-performing instances.

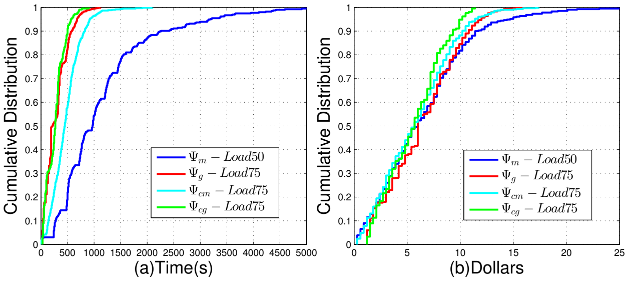

Figure 3. Seeking (a) latency and (b) cost in ψ_{min} cluster

E. Impact of Resource Allocation

1) Request Throughput: cloud data center should monitor the status of each cluster for the purpose of management and load balance. Hence, it would not be difficult for cloud providers to detect the sudden increase of tenants' requests. Fig 4 shows the request throughput in ψ_{min} cluster with Load75. We keep the instance seeking strategies Ψ_{cg} unchanged to observe the impact of different allocation strategies $\phi_{host}^{cluster}$. Normally, for every tenant, they only need to submit one request to acquire the desired number of VMs. In the cloud, there are four types of events in each tenant's life cycle: *VM_Create, VM_Create_ACK, VM_Destroy* and *VM_Destroy_ACK*. Hence, no matter how $\phi_{host}^{cluster}$ changes, the *expected request throughput* of all tenants using normal strategy should be 4 times the tenant request rate which is 4 *requests/minute*. However, with the extra requests for seeking better instances, the request throughput increases dramatically as is shown in the Fig. 4:

i) The growth rate of throughput is greatly different with the change of $\phi_{host}^{cluster}$. Compared with the expected request throughput, ϕ_{mcf}^{lff} allocation results in the most drastically increase as much as 42 times while even the minimal increase brought by ϕ_{rand}^{rr} is almost 2 times what the expected.

ii) For the same cluster level allocation, the throughput of random-based host level allocation ($\phi_{rand}^{cluster}$) is smaller than that of mcf-based ($\phi_{mcf}^{cluster}$). Since E5645 has two more CPU cores than the other two types, the mcf-based host allocation $\phi_{mcf}^{cluster}$ will firstly assign the E5645 to tenants. Thus, even tenants with normal strategy Ψ_n in the learning process could get better-performing instances in a first-come-first-served manner until all E5645 servers are consumed two more CPU cores. The number of better cores drops significantly in the learning process under mcf-based allocations, which greatly decreases the probability of

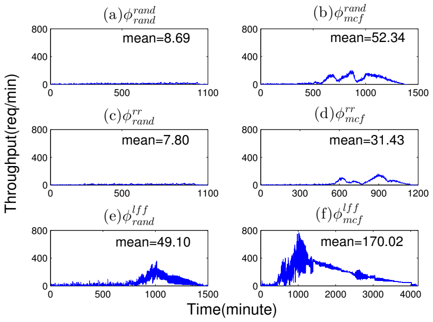

Figure 4. Throughputs of ψ_{min} cluster with varying allocation strategy

following tenants to acquire better instances. From Fig. 4 (a) and (b), with different host level allocations, the mean throughput of ϕ^{rand}_{mcf} is 6 times as high as that of ϕ^{rand}_{rand}.

iii) For the same host level allocation, the throughput of round-robin (ϕ^{rr}_{host}) is slightly smaller than that of random-based (ϕ^{rand}_{host}) but dramatically smaller than that of llf-based (ϕ^{llf}_{host}). The round-robin is more fair than random-based allocation to each cluster. Since we only simulate 3 clusters, it is reasonable to understand the similar performances of round-robin and random allocations. However, the essence of llf-based allocation is to keep the load balance among different clusters. It always selects the cluster with the lowest utilization first. Thus, the cluster with ψ_{min} distribution is the most likely to be selected, which makes the cluster exhaustive quickly.

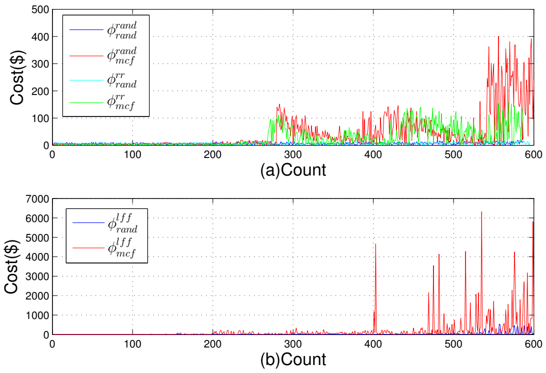

Figure 5. Impact of allocation strategies on seeking cost in ψ_{min} cluster

2) Seeking latency and cost: Fig. 5 shows the seeking cost of tenants with Ψ_{cg} to find better instances under the Load75. The seeking latency shows the similar trends and we will not shed light on it due to space limit. We can see that the seeking cost rises rapidly with a growing number of tenants' seeking instances. From Fig. 5 (a), the seeking cost under the ϕ^{rand}_{rand} and ϕ^{rr}_{rand} is relatively stable at about 20$ while that under ϕ^{rand}_{mcf} and ϕ^{rr}_{mcf} rises to around 100$ after about 260th tenants. For the last about 50 tenants under ϕ^{rand}_{mcf} , the seeking cost can be as high as around 300$. Also, it is interesting to observe a sharp-increase-then-decrease-gradual loop in the cost of tenants with ϕ^{rand}_{mcf} and ϕ^{rr}_{mcf}. It is also clearly seen from Fig 5 (b) that the costs of ϕ^{lff}_{rand} and ϕ^{lff}_{mcf} are much higher than that in Fig 5 (a). The average seeking cost of ϕ^{lff}_{mcf} , which is also the highest, is 31 times that of ϕ^{rand}_{rand}. Hence, we can see that the seeking cost of tenants is significantly affected by the resource allocation strategy. The tenant will be daunted by the high seeking cost and long seeking latency.

V. Discussion

Our study serves as a first step towards better understanding the impact of instance seeking strategies in public clouds. The results of simulation from this study provide insights that are valuable to both cloud tenants and providers. In this section, we will discuss the implications of our findings from the perspectives of both tenants and providers.

Cloud tenant: Cloud tenants always purse the high price-performance ratio. Given the same instance type, the price is the same for all tenants but the instance performance varies a lot. Consequently, it is natural for tenants to seek better instances by different strategies. Intuitively, we thought that moderate strategy Ψ_m could help tenants save more time and money. Our results, nevertheless, show that compared with Ψ_m, greedy strategy Ψ_g is less likely to drain of better-performing instances in the cluster under the same workload. The reason for this is, at any given time t, Ψ_g strategy could improve the probability to acquire better instances through the greedy policy. Therefore, tenants with Ψ_g strategy could solve the instance seeking problem with less time and cost. However, tenants occupying more resources in a greedy way significantly influence other tenants' probabilities to obtain better-performing instances. Our results also show that the collaborative greedy-based strategy Ψ_{cg} is the best choice for tenants in terms of both seeking latency and cost. Nevertheless, it is difficult to persuade tenants to collaborate with each other. Thus, most of the tenants are more likely to adopt moderate and greedy strategies, which may make cluster exhaustive quickly, especially for the cluster with a minority of better instances.

Another interesting finding is shown in the Fig. 6. If all the tenants use normal strategy Ψ_n, the probability to acquire a better-performing instance is the highest at any time. The collaborative-based strategies improve the probability greatly, whilst the greedy strategy makes the highest adverse effects on the following tenants who request later. Thus, our results give a good motivation for tenants to adopt collaborative strategies.

Cloud provider: Provisioning instances of homogeneous performance from heterogeneous hardware remains a tough problem. Other phenomena like CPU, disk and network contention also result in vast performance variations within

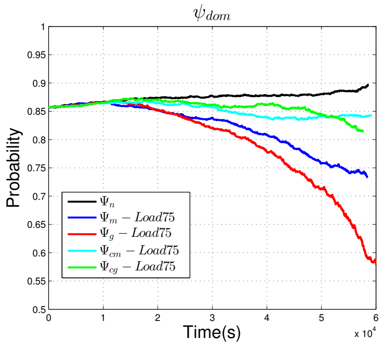

Figure 6. Probability to acquire a better-performing instance in ψ_{dom}

instances of the same type. We argue that this problem will be long-lasting and tenants will keep seeking better instances with different strategies. However, tenants' seeking instances leads to the quick growth of requests and low utilization for worse-performing instances. How would cloud providers react to tenants' seeking behaviors? It largely depends on their profits. The tenants pay the same price for either better- or worse-performing instances. Thus, if the operation cost of the latest hardware is smaller than that of the old ones, they are pleased to see that more latest hardware are found and used by tenants. If not, they, however, may change their VM allocation strategies or the price scheme to discourage the tenants' seeking behaviors, thereby maximizing their profits. As can be seen from Fig. 5, the seeking cost and latency largely depend on the VM allocation strategies. Cloud providers could easily control the situation by adjusting the allocation policies. Nevertheless, this also harms the fairness among cloud tenants. The optimal solution is to introduce more efficient price schema or to provide a more homogeneous platform. For example, they can give the discount for instances hosted by old hardware or even they can give tenants the right to choose the underlying hardware.

VI. Conclusion

In this paper, we present a cloud and a tenant model to simulate the process of tenants seeking the better-performing instances in the cloud. Six cloud resource allocation strategies and five instance seeking strategies are designed and evaluated based on the real job data from Google cluster traces. Through the simulation, we observe the cluster exhaustion and significant request growth in the cloud. We find that the greedy strategy outperforms the moderate one while collaborative-based strategies help tenants save more time and budget. Also, resources allocation strategies in the cloud exert a strong influence on the effectiveness of seeking strategies. The cloud provider could discourage the tenants' seeking behaviors with changing resource allocation strategies. Finally, we discuss the implications of our findings from the perspectives of both tenants and providers. In the future, we will introduce the game theory into our analysis and we may also design a tenant behavior aware resource allocation policy for heterogeneous computing environment.

Acknowledgment

This work was funded by EU project CloudSpaces (FP7-317555) and MAMMotH project (Dnro 820/31/2011) from Finnish funding agency for technology and innovation.

References

[1] "Amazon EC2," http://aws.amazon.com.

[2] Z. Ou, H. Zhuang, J. Nurminen, A. Ylä-Jääski, and P. Hui, "Exploiting hardware heterogeneity within the same instance type of amazon ec2," in *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*. USENIX Association, 2012, pp. 4–4.

[3] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. Bowers, and M. Swift, "More for your money: exploiting performance heterogeneity in public clouds," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 20.

[4] G. Wang and T. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[5] T. Zou, R. Le Bras, M. Salles, A. Demers, and J. Gehrke, "Cloudia: A deployment advisor for public clouds," *Proceedings of the VLDB Endowment*, vol. 6, no. 2, 2012.

[6] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th annual conference on Internet measurement*. ACM, 2010, pp. 1–14.

[7] K. Mills, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 91–98.

[8] G. Lee, B. Chun, and R. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in *3rd USENIX Workshop on Hot Topics in Cloud Computing (Hot-Cloud11)*, 2011.

[9] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 7.

[10] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[11] "Intel 5000 Sequence," http://ark.intel.com/products/family/28144.

[12] "Openstack Scheduler," https://github.com/openstack/nova/tree/master/nova/scheduler.

[13] "Google cluster trace," http://code.google.com/p/googleclusterdata.