QoS-aware Service VM Provisioning in Clouds: Experiences, Models, and Cost Analysis

Mathias Björkqvist[1], Sebastiano Spicuglia[2], Lydia Chen[1], and Walter Binder[2]

[1] IBM Research Zürich Laboratory
Rüschlikon, Switzerland
mbj,yic@zurich.ibm.com

[2] University of Lugano
Lugano, Switzerland
firstname.lastname@usi.ch

Abstract. Recent studies show that service systems hosted in clouds can elastically scale the provisioning of pre-configured virtual machines (VMs) with workload demands, but suffer from performance variability, particularly from varying response times. Service management in clouds is further complicated especially when aiming at striking an optimal trade-off between cost (i.e., proportional to the number and types of VM instances) and the fulfillment of quality-of-service (QoS) properties (e.g., a system should serve at least 30 requests per second for more than 90% of the time). In this paper, we develop a QoS-aware VM provisioning policy for service systems in clouds with high capacity variability, using experimental as well as modeling approaches. Using a wiki service hosted in a private cloud, we empirically quantify the QoS variability of a single VM with different configurations in terms of capacity. We develop a Markovian framework which explicitly models the capacity variability of a service cluster and derives a probability distribution of QoS fulfillment. To achieve the guaranteed QoS at minimal cost, we construct theoretical and numerical cost analyses, which facilitate the search for an optimal size of a given VM configuration, and additionally support the comparison between VM configurations.

Keywords: QoS, cloud services, VM provisioning, Markovian models

1 Introduction

Service systems are increasingly deployed in clouds due to the advantages of scalability and ease of management. In the cloud, a set of preconfigured VM instances is available at different costs (e.g., small, medium, large, and very large instances in Amazon EC2 [1]), and their corresponding hardware-related performance metrics are provided at best effort [15]. Meanwhile, service providers face ever more stringent QoS demands from users, in particular regarding the tail performance, e.g., 95^{th} percentile or higher response times. The difficulties of service management in clouds (i.e., selecting a VM configuration and dimensioning the system correctly) are further exacerbated when aiming at providing

QoS guarantees for both average and tail performance [8], while retaining the cloud advantages at the same time.

Several empirical studies [6, 16, 20] point out a common pitfall in clouds that the performance variability — in this case the response time of services — fluctuates significantly and tail latency degrades due to the heterogeneity of the underlying hardware and the workloads collocated on the same physical hosts. Although virtualization enables the efficient multiplexing of workloads across the ample hardware resource, performance isolation is limited [7], especially for applications that are not CPU intensive. While the performance variability persists in cloud platforms, little is known about the sensitivity of services on different VM configurations in terms of capacity[3] i.e., the maximum number of service requests that can be processed sustainably, and the aggregate impact of the capacity variability of a single VM on the QoS of the entire service cluster.

VM provisioning of service systems is typically based on the average capacity, which in turn is a good indicator for systems experiencing low variability and providing simple QoS guarantees [22], such as average throughput over a certain threshold. To avoid performance penalties due to variability in the cloud, selecting VMs with desirable performance becomes of paramount importance not only to reduce performance variability [9], but also to optimize cost [3, 5]. Consequently, empirical approaches are proposed to acquire VMs with higher capacities. However, due to the empirical nature of the proposed VM selection strategies, a QoS promise of satisfying a given target throughput is only attained at best effort. Moreover, the resulting cost minimization may be arbitrary, depending on the workload dynamics of the underlying cloud platform.

Our study aims to find the optimal VM provisioning for a service system, i.e., composed of an ideal VM configuration using a minimum number of VM instances, such that the required QoS properties are guaranteed for a certain fraction of time at minimal cost (e.g., 90% of the time the sustainable throughput should be at least 30 requests per second). To such an end, we study a wikipedia service [19] and first empirically quantify its capacity variability on different VM configurations, in the presence a daemon VM executing various benchmark workloads in a private cloud. Leveraging our empirical experience, we build a Markovian model which explicitly models the capacity variability of an entire cluster, and we derive the *probability distribution* of the delivered QoS for a given number of VMs of a certain configuration. Based on analytical solutions regarding the QoS fulfillment, we construct theoretical and numerical analyses to evaluate the tradeoff between cost and the fulfillment of QoS promises, (1) by comparing optimal provisioning to simple pessimistic and optimistic provisioning; (2) when provisioning based on the average capacity fails; and (3) when choosing a VM configuration that returns the best cost/service-availability ratio.

This paper is organized as follows: The capacity variability of a VM hosting a wiki service on different VM configurations is discussed in Section 2. The

[3] In this paper, we use the terms capacity and sustainable maximum throughput interchangeably.

proposed Markovian model and VM provisioning optimization is described in Section 3. Section 4 presents our cost analysis. Related studies are summarized in Section 5. Section 6 concludes this paper.

2 Capacity Variability of Service VM Configuration

In this section, we use a controlled cloud environment to study the capacity variability of service hosting on different VM configurations, i.e., the fluctuation of capacity, against single neighboring VMs executing various workloads. To such an end, our target service is a wikipedia deployed on a set of VM configurations and collocated with a daemon VM executing Dacapo benchmarks [2] in a private cloud. Essentially, we use the daemon VM to synthesize interference that can be encountered by a wiki VM in the cloud and parametrize the capacity variability, which is then used to build the QoS model for a service cluster in Section 3.

2.1 Experiment Setup

From our private cloud environment, we chose two IBM System x3650 M4 machines, *gschwend* and *nussli*, each with 12 Intel Xeon E5-2620 cores running at 2.00GHz, and 64 and 36 GB of RAM, respectively, for running our experiments. We use KVM on gschwend for hosting our target and daemon VMs, and nussli for generating the Apache JMeter workload requests for our target wiki VM.

The target wikipedia system is based on a subset of 500000 entries from a *pages-articles.xml* dump downloaded on October 12, 2012. The wiki VM is a Debian 7.0 system running an Apache 2.4.4 web server, the PHP 5.4.15 server-side script engine, MediaWiki 1.21 as the web application, and the MySQL 5.5.31 database. The number of threads employed by Jmeter is configured such that the maximum throughput of the wiki VM is reached. As for the workload on the daemon VM, we selected the following benchmarks from the Dacapo benchmark suite: (1) `fop`, a lowly threaded CPU-intensive benchmark; (2) `luindex`, a lowly threaded IO-intensive benchmark; (3) `sunflow`, a highly threaded CPU-intensive benchmark; (4) `lusearch`, a highly threaded CPU- and IO-intensive benchmark; and (5) `tomcat`, a network-intensive benchmark. We refer readers to [7] for the detailed threading behaviors and characterization of the Dacapo benchmarks.

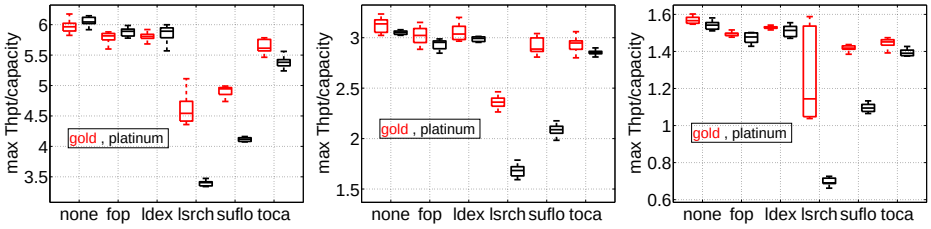
We consider four types of VM configurations, with CPUs and memory sizes as listed in Table 1, which are comparable to VM offerings in Amazon EC2 [1]. We use three configurations for the wiki VM (bronze, silver, and gold), and two configurations for the daemon VM (gold and platinum). Based on experimental evaluation, we use two, four, and eight threads when running Jmeter against a wiki running on a bronze, silver, and gold VM instance, respectively. In total, we evaluate the amount of performance interference experienced by the wiki under 36 scenarios, i.e., three configurations of wiki VMs $\times$ six types of daemon workloads (5 DaCapo benchmarks and no workload) $\times$ two daemon VM configurations.

Table 1. VM configurations and naming conventions

	Bronze	Silver	Gold	Platinum
No. processing units	1	2	4	8
RAM (GB)	4	8	16	32

The target wiki performance statistics are collected from the Apache log files which record the current time, the requested URL, and response time for each request. After a warmup period for the wiki VM, Jmeter, the daemon VM and the DaCapo benchmark, we start collecting statistics for five minutes for each of the 36 scenarios, each of which is repeated ten times. We summarize the results of $36 * 10 = 360$ runs using box plots in Fig. 1. One can straightforwardly find that the capacity variability of the wiki, i.e., the difference between no workload and different DaCapo benchmarks running on the daemon VM, can vary significantly depending on VM configurations and the characteristics of the DaCapo benchmark.

For further analysis, we take the median of the repeated runs of all scenarios and compute the average of the normalized throughput, compared to the scenario with no daemon VM neighbor. We thereafter categorize the results by target VM type, daemon VM type, and benchmark.



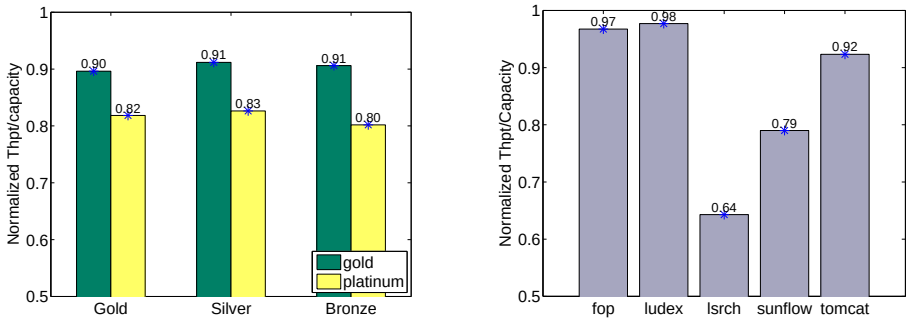
(a) Wiki on gold (b) Wiki on silver (c) Wiki on bronze

Fig. 1. Capacity variability of a wiki running on different VM configurations against `fop`, `luindex`, `lusearch`, `sunflow`, and `tomcat`, hosted on gold and platinum VMs: box plots based on 10 repetitions.

2.2 (In)sensitivity of Capacity Variability

To compare the robustness of different target VM configurations, we normalize the throughput of the wiki VM by the throughput of the wiki without any neighbor for gold, silver, and bronze VMs. In Fig. 2(a), we present the average normalized throughput, a higher value of which means less interference is observed and the wiki VM is more robust. When collocated with a gold daemon VM, the difference between wikis running on different VM configurations is almost negligible. However, in our setup, when the daemon VM is more dominant, i.e., a platinum VM, a wiki on a silver VM seems to be slightly more robust than when on a gold or bronze VM. Such an observation can also be made for individual daemon workloads, see Fig. 1. Overall, our experiments show that a

wiki running on a silver VM is slightly more robust to noisy neighbors, and the capacity of the wiki can be throttled by 10-20% on average due to interference from neighboring VMs.



(a) Impact of different target VM configurations (b) Impact of different daemon workloads

Fig. 2. Average analysis of normalized throughput of target wiki.

2.3 A Really Noisy Daemon

We try to identify which type of workload represents the noisiest neighbor and causes high capacity variability for a wiki service collocated on the same physical machine. We compute the average normalized throughput across all target VM configurations for each benchmark, as presented in Fig. 2(b). One can clearly see three levels of performance variability: (1) mild interference from `fop`, `luindex`, and `tomcat`, where the capacity degradation is within 10%; (2) medium interference from `sunflow`, where the capacity is degraded by roughly 20%; and (3) high interference from `lusearch`, where the capacity degradation can be up to 35%. Clearly, `lusearch` is the noisiest neighboring VM for our wiki service, as they both compete for a similar set of resources, i.e., both CPU and IO. As both `fop` and `luindex` have limited concurrent threading, only limited performance interference is observed.

Up to this point, our experiments have addressed the variability of a wiki service hosted on a single VM. In the next section, we leverage Markovian modeling to capture the capacity variability of a wiki cluster consisting of multiple VMs.

3 Markov Chain Model for Service Cluster

In this section, our objective is to derive a rigorous mathematical analysis for answering the question, "what is the minimum size of a cluster whose VMs experience capacity variability such that the probability of achieving a target QoS is guaranteed?". We define the service capacity $C(n)$ as the total number of requests processed by a cluster of $n \in \mathbb{Z}$ VMs, its QoS target as C^*, the

fulfillment of which should be above a certain threshold ξ. Using Markov chain modeling, we obtain the steady-state distribution of QoS of a cluster with n VMs, and further search for the minimum n that satisfies the desired availability, $Pr[C(n) > C^*] > \xi$.

We start out the analysis by modeling the transition between high and low capacity of a single wiki VM, using values obtained in the previous section. Based on that, we develop a continuous-time Markov chain to model the service availability of the entire cluster. Finally, we show, by theoretical analysis and numerical examples, that the proposed minimum cluster size, n^*, indeed attains a good trade-off between cost and guarantee of service availability.

3.1 Single VM node

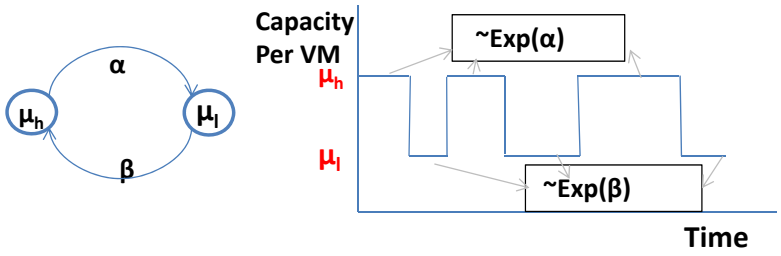


Fig. 3. Capacity variability of a VM: state diagram of high and low capacity (left) and illustration of time series (right)

We assume that a VM of a certain configuration (e.g., gold, silver, or bronze) alternates between states of high and low capacity, denoted by μ_h and μ_l, for exponentially distributed times with rate α and β, respectively. Examples of such values can be found in Fig. 1 for different VM configurations. We term the difference between μ_h and μ_l the capacity variability, and (α, β) the intensity of the variability. Fig. 3 illustrates the state transitions and time series of such a model. To capture the maximum variability possibly experienced by a VM, we only adopt two states of capacity, namely high and low, for different VM configurations. Their parameterizations can be carried out by our empirical analysis in Section 2. On the contrary, the values of α and β depend on the workload dynamics of the underlying cloud, and thus are assumed invariant to VM configurations. Note that one may find intermediate states in reality, i.e., the capacity is between $[\mu_l, \mu_h]$. Our proposed model can be further refined to accommodate multiple levels of capacities, albeit with a higher computation overhead for obtaining steady-state probability of service availability (see the next subsection).

3.2 Continuous Markov Chain Modeling of the Cluster

The single VM model naturally leads us to use a continuous-time Markov chain (CTMC) to describe the dynamics of available capacity in a cluster consisting of n VMs, experiencing high and low capacity. In the proposed CTMC, a state $i \in I = \{1, 2, \ldots n\}$ is defined as the number of VMs having low capacity, while the rest of $n - i$ VMs in the cluster have high capacity. Consequently, the corresponding capacity of state i in the systems is

$$C_i(n) = i\mu_l + (n - i)\mu_h.$$

Note that $C_i(n) \geq C_j(n)$, for $i \leq j$ — essentially, $C_i(n)$ monotonically decreases in i. When there are i VMs with low capacity, the system transpositions to state $i + 1$ with the rate $(n - i)\alpha$, and to state $i - 1$ with the rate $i\beta$. Fig. 4 illustrates such a Markov chain for a cluster of n VMs.

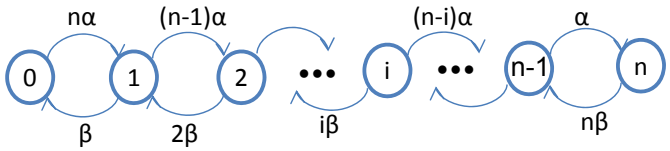


Fig. 4. Markov chain of aggregate VM capacities, where the state denotes the number of VMs experiencing low capacity.

We let $\pi = [\pi_0, \pi_1 \ldots \pi_n]$ denote the steady-state probability that the system has a service capacity of $C_i(n)$. One can solve the Markov chain in Fig. 4 by a set of balance equations [13], i.e.,

$$(n - i)\alpha\pi_i = (i)\beta\pi_{i+1} \; \forall i,$$

$$\sum_i \pi_i = 1.$$

Substituting all π_i as a function of π_n, we can then obtain the closed formed solution of π

$$\pi_n = \frac{1}{(1 + \frac{\alpha}{\beta})^n} \tag{1}$$

$$\pi_i = \binom{n}{i} (\frac{\alpha}{\beta})^{n-i} \pi_n, \; 0 \leq i < n.$$

Consequently, we can derive the probability that the service capacity is greater than the target

$$\mathbf{Pr}[C(n) > C^*] = \sum_{i \in \{I : C_i(n) > C^*, i \leq n\}} \pi_i. \tag{2}$$

To compute $\mathbf{Pr}[C(n) > C^*]$ for all $n \in \mathbb{Z}$, one shall first compute the values π_i, $\forall i$ using 1 for a given n, and the sum of π_i for the states i where the resulting capacity is greater than C^*, and then iterate the computation procedure for all values of n.

3.3 Trade-off between Cost and Service Availability

To find a minimum cluster size that ensures that a service capacity greater than the target capacity, $C > C^*$, is guaranteed for $\xi\%$ of time, we can formulate the following optimization after substituting Eq. 1 into the constraints and re-arrangements:

$$\begin{aligned} \text{minimize} \quad & n \\ \text{subject to} \quad & (n-i)\mu_h + i\mu_l \geq C^* \\ & \sum_i \binom{n}{i} (\frac{\alpha}{\beta})^{n-i} \frac{1}{(1+\frac{\alpha}{\beta})^n} \geq \xi \\ & i \leq n \end{aligned}$$

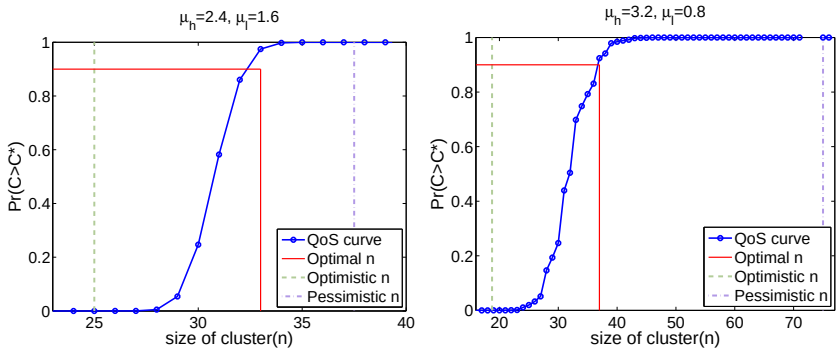
For given values of α, β, μ_h, and μ_l, $Pr[C(n) > C^*]$ is a function increasing in n, i.e., when $n_1 \geq n_2$, $\mathbf{Pr}[C(n_1) > C^*] \geq \mathbf{Pr}[C(n_2) > C^*]$, as self-explained in the second constraint in the above optimization. Consequently, one can straightforwardly find the optimal n^* by linearly searching through the possible values of $n \in \mathbb{Z}$ in an increasing order.

Note that the optimization is constructed implicitly depending on the workload intensity via the value of C^*. For a given period of time when the workload intensity is predicted as λ requests per second, one may want to keep the system 80% utilized, and set the target capacity to $C^* = \lambda/0.8$. The choice of the target capacity is out of scope of this work, and we direct interested readers to our prior work [4, 5].

n* **vs. Simple Solutions** Herein, we illustrate how n^* obtained through our proposed methodology attains a good trade-off between the cost and the guaranteed service availability, compared to simple optimistic and pessimistic solutions. One may optimistically think that all VMs have high capacity and only purchase $n^{opm} = \lceil C^*/\mu_h \rceil$ VMs by simply dividing the target capacity with the value of high capacity of a single VM. In contrast, a pessimistic solution would be to assume that all VMs have low capacity and purchase $n^{psm} = \lceil C^*/\mu_l \rceil$. As $\mu_h > \mu_l$, n^{psm} is greater than n^{opm}.

We compute the service availability curves by Eq. 2 for all values of n that fulfill the target capacity of $C^* = 60$ requests per second, using $\alpha = 60$, $\beta = 50$ and two sets of μ_h and μ_l, respectively. Fig. 5 summarizes the numerical results. Additionally, we also graphically illustrate the optimal provisioning of VMs (n^*) that fulfill the desired service availability, i.e., the cluster capacity is greater than 60 for $\xi = 90\%$ of the time, compared with pessimistic (n^{psm}) and optimistic (n^{opm}) solutions. We consider service availability curves in two cases of capacity

variability, namely with smaller and bigger difference between the high and low capacity of a VM. One can easily see that the optimal cluster size grows with the variability, indicated by a higher value of n^* in Fig. 5(b) than (a). When the variability of capacity is higher, the service availability curve increases slower in n than in the low variability case. Moreover, the pessimistic and optimistic allocations are even further away from the optimal one.



(a) Smaller variability (b) Bigger Variability

Fig. 5. Service availability curve, $Pr[C(n) > 60]$: the optimal number of VMs to achieve $\xi = 90\%$, the pessimistic, and the optimistic solution.

To proceed to cost comparison, we assume the cost of a cluster, $cost(n)$, is a strictly increasing function in n, i.e., $cost(n_1) \geq cost(n_2)$ when $n_1 \geq n_2$. Furthermore, due to the monotonicity of $Pr[C(n) > C^*]$ and $n^{opm} \leq n^* \leq n^{psm}$, we reach the following corollary:

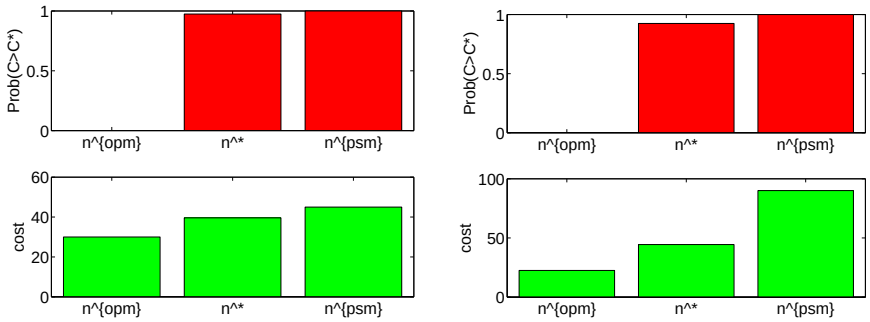
Corollary 31

$$\begin{aligned} &cost(n^{opm}) \leq cost(n^*) \leq cost(n^{psm}), \\ &\mathbf{Pr}[C(n^{opm}) > C^*] \leq \mathbf{Pr}[C(n^*) > C^*] \simeq \xi \leq \mathbf{Pr}[C(n^{psm}) > C^*]. \end{aligned} \tag{3}$$

Though the optimistic solution incurs lower cost, the QoS fulfillment threshold is not met. On the contrary, the pessimistic solution can achieve the service availability with 100% guarantee, but at a higher cost. The optimal provisioning of VMs, n^*, indeed achieves a good trade-off between cost and QoS fulfillment, compared to simple optimistic and pessimistic solutions. Note that n^* can result in a slightly higher value of $\mathbf{Pr}[C(n^*) > C^*]$ than ξ, due to the discrete choice of the number of VMs.

We further numerically illustrate how such a trade-off is affected by different levels of variability in capacity of a single VM. Using a simple linear cost function, i.e., $cost(n) = 1.2 \cdot n$, we construct two numerical examples in Fig. 6, following the parameters discussed in Fig. 5. Note that the cost here is defined as the cost per time unit, which can be aligned with the billing periods used in commercial clouds, e.g., one hour. One can see that n^* can improve the QoS fulfillment

drastically by increasing cost, compared to n^{opm}, and reduce cost significantly by allowing a fractional capacity degradation, compared to n^{psm}. The advantage of n^* in attaining a good trade-off is even more prominent in the case of bigger variability.



(a) Lower variability, $(\mu_l, \mu_h) = (1.6, 2.4)$ (b) Higher variability, $(\mu_l, \mu_h) = (0.8, 3.2)$

Fig. 6. QoS fulfillment vs. cost: $\mathbf{Pr}[C(n) > C* = 60] > \xi = 0.9$

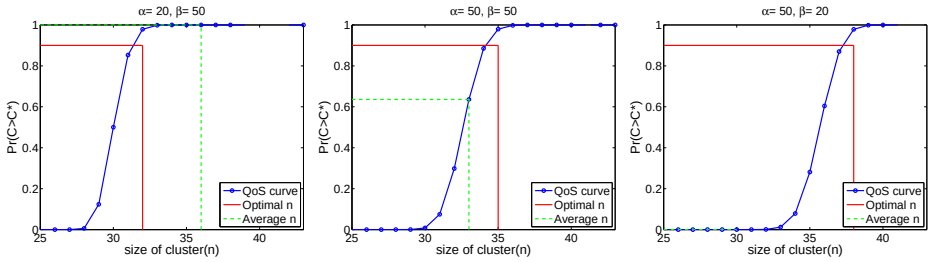
Why not Consider Average Capacity of a VM? In this subsection, we show that choosing n based on the average capacity of a VM cannot reach the optimal values nor guarantee QoS fulfillment at the target capacity level, using numerical examples. Recalling the state transition of a VM depicted in Fig. 3(a), the average capacity of a single VM, μ, and the VM provisioning based on the average capacity, n^{avg} are

$$\mu = \frac{\mu_h \alpha + \mu_l \beta}{\alpha + \beta}, \texttt{and } n^{avg} = \frac{C^*}{\mu},$$

respectively. Fig. 7 demonstrates that a cluster size based on the average capacity is not a reliable solution under three scenarios of (α, β), namely (a) often experiencing low capacity (b) alternating between high and low capacity equally, and (c) often experiencing high capacity. We let $\mu_h = 2.4$ and $\mu_l = 1.6$, as used in the case of small variability. Shown in Fig. 7(a), when $\alpha < \beta$, n^{avg} tends to overestimate and $\mathbf{Pr}[C(n) > C^*]$ is over the required values, $\xi = 0.9$. When $\alpha > \beta$, n^{avg} tends to underestimate and $\mathbf{Pr}[C(n) > C^*]$ is below the required values, indicated by the horizontal line overlapped on the x-axis in Fig. 7(c).

As for $\alpha = \beta$, we want to highlight that n^{avg} can achieve the target capacity roughly 50% of the time, for any capacity variability and target values. This observation can be explained by Eq. 1. When $\alpha = \beta$, the steady state of QoS fulfillment is greatly simplified to $\pi_n = 1/2^n$ and $\pi_i = \binom{n}{i}(1/2^n)$. Thus, substituting $n^{avg} = \lceil 1/2\mu_h + \mu_l \rceil$ can result in $\mathbf{Pr}[C(n) > C^*] = (50 + \epsilon)\%]$, where ϵ is a small positive fluctuation due to the ceiling operator on n^{avg}.

Observation 32 *When $\alpha = \beta$, n^{avg} can achieve $C(n) > C^*$ roughly 50% of the time, i.e., $Pr[C(n) > C^*] = 50 + \epsilon\%$, where ϵ is a small positive value.*



(a) $\alpha < \beta$ (b) $\alpha = \beta$ $\alpha > \beta$

Fig. 7. QoS fulfillment curves based on n^{avg} and n^*, under $\mu_h = 2.4$ and $\mu_l = 1.6$.

4 Choosing a VM Configuration

In this section, we compare different VM configurations in terms of their optimal cluster sizes and total cost, based on our proposed Markov chain model. Using theoretical and numerical analysis, we study if a cluster composed of more powerful VMs is always smaller than a cluster of weaker VMs. Due to the large number of parameters considered, we focus on providing a condition where weaker VMs imply a bigger cluster, and numerical counter examples where a cluster of weaker VMs can provide better service availability than a cluster of more powerful VMs.

4.1 Typical Case: Weaker VM Means a Bigger Cluster

Following the convention in Section 2, we consider three types of VM instances, namely gold, silver, and bronze. A gold instance is more powerful and implies a higher average computational capacity than a silver instance, whose average capacity is more than that of a bronze instance. All VM configurations experience high ($\mu_{h,type}$) and low capacity ($\mu_{l,type}$) for exponentially distributed durations with means equal to α and β, respectively. We can show the necessary condition for the *typical case*, meaning clusters of weaker VMs are bigger than clusters of more powerful VMs when achieving the same target of service availability.

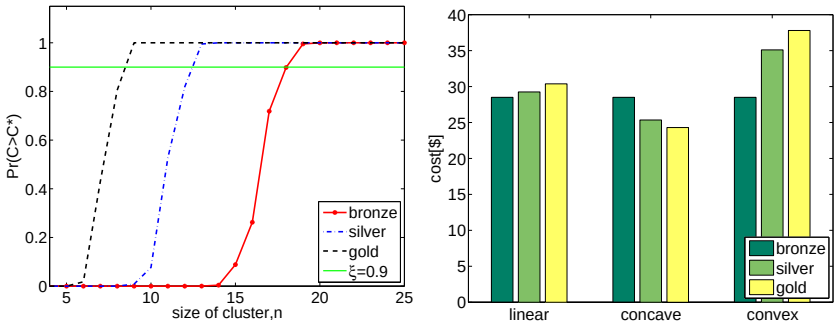
Theorem 41 *When experiencing the same α and β and aiming at the same service availability threshold, the cluster sizes of gold, silver, and bronze instances are*

$$n^*_{gold} \leq n^*_{silver} \leq n^*_{bronze}, when$$

$$\mu_{h,gold} \leq \mu_{h,silver} \leq \mu_{h,bronze}, and\, \mu_{l,gold} \leq \mu_{l,silver} \leq \mu_{h,bronze}.$$

The theorem follows straightforwardly from the monotonicity of $Pr[C(n) > C^*]$ in n. Due to the lack of space, we skip the proof. The theorem tells us that to guarantee the same level of service availability, one should definitely acquire a higher number of weaker VMs than powerful VMs, when the low and high capacity of weaker VMs are inferior to the low and high capacity of powerful VMs, respectively.

We note that the typical case simply implies the order of n^* for different configurations, not the differences in their costs. Using three types of cost functions, namely linear, concave, and convex, we show that the costs of different types of VM clusters can vary a lot. In particular, the high and low capacities experienced by each VM configuration are listed under the typical case in Table 2, where (α, β) are (40,20). The linear/concave/convex cost function means the cost per VM instance is linearly/concavely/convexly proportional to the average capacity of single VM of a particular type. We set the cost per VM per time unit of (gold, silver, bronze) for linear, concave, and convex as (1.5, 2.25, 3.375), (1.5, 1.95, 2.7), and (1.5, 2.7, 4.2), respectively. Fig. 8(a) and (b) summarize the resulting service availability curves of different VM types and the resulting costs under different cost functions. One can see that although the bronze cluster is much bigger than the gold, the cost can still be lower when the cost per VM is linearly and convexly proportional to their average capacity. On the contrary, when there is a discount on computational capacity, i.e., when the cost per unit of computation decreases for gold, a gold cluster can be a cheaper option as shown by the case of a concave cost function.



(a) QoS Curves (b) Different cost functions

Fig. 8. QoS fulfillment and cost comparison for a typical case: comparison of gold, silver and bronze VMs.

4.2 Counter Example: A Cluster of Weaker VMs Can Be Smaller

Here, we show by some counter examples that the optimal size of a cluster with weaker VMs is not necessarily larger. The capacity parameters of gold, silver, and bronze instances used are listed under the counter example in Table 2. The average capacity, μ, is the average of high and low capacity, and grows with the VM configuration. However, the capacity variability, i.e., the difference between high and low capacity, is higher for more powerful VMs.

Fig. 9 summarizes the curve of QoS fulfillment of the three VM configurations. One can see that the QoS curve of the three types of VMs cross each other at $n = 15$. For a given size, the QoS of a gold VM is not necessarily higher than

Table 2. Capacity parameters of single VM for all VM types.

	Typical Case			Counter Example		
	μ_l	μ	μ_h	μ_l	μ	μ_h
Gold	3.75	4.5	5.62	0.26	2.65	5.03
Silver	2.25	3.00	3.75	0.95	2.30	3.64
Bronze	1.50	2.00	2.50	1.80	2.00	2.20

that of a silver or bronze VM. In particular, for $n \geq 15$, the QoS of a silver VM is higher or equal to a gold VM. As a result, depending on the threshold of QoS, ξ, the optimal cluster size of bronze VMs can be bigger, or smaller than that of gold VMs. To guarantee $Pr[C(n) > 30] \geq 0.85$, the optimal cluster size of all three types of VMs is 16. When such a threshold is higher than 0.85, the number of VMs in a gold cluster should be higher than in a bronze cluster. This leads us to conclude that not only the average, but also the variability in VM throughput is crucial in choosing and sizing VM clusters in the cloud.

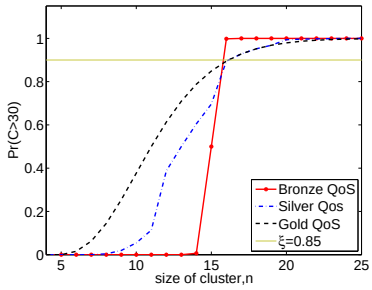


Fig. 9. QoS curve of different types of VMs, under $\alpha = \beta = 20$, $Pr[C > 30] > 0.85$.

Our proposed Markov model and solution provide an efficient means to explore a large number of parameters encountered, such as different cost functions, and exogenous variabilities and their intensity, when choosing the right VM configuration and deciding the cluster size. Numerical examples serve the purpose of illustrating how our solution robustly attains an optimal trade-off between cost and QoS fulfillment across different system parameters and VM configurations.

5 Related Work

Recent studies on QoS analysis for cloud services [18, 21, 22] are mainly driven by service compositions and service selection, using a Markovian decision process [14] or a Baysian network model [21]. In contrast, studies focusing on constant QoS value, e.g., Zheng [22] proposed a calculation method to estimate the probabilistic distribution of QoS. However, the impact on the QoS due to the underlying performance variability of the cloud is to a large extent overlooked.

Most existing studies on the performance variability of applications hosted in the cloud are based on empirical experiments, especially in terms of average and 95^{th} response time [16, 20], and aim to discover the root cause of such a phenomenon [10–12]. The observations made from cloud experiments are mainly based on a single type of configuration and simple benchmarks. A few studies [16, 17, 20] focus on multiple types of VM configurations and try to quantify the variability in their response times. Moreover, the variability in throughput is largely evaluated under a particular workload intensity, instead of using the maximum sustainable throughput, i.e., the capacity.

Meanwhile, another set of studies focus on developing solutions to reduce the performance variability in a best effort manner, from the perspective of service providers. Particularly, both [5,9] propose opportunistically selecting VMs which have high capacity, while discarding VMs with low capacity. Another type of solution is to try to figure out the underlying hardware and neighboring workloads, so as to select similar physical hosts [16] and influence the neighboring VMs [15]. As the methodology is trial and error, the QoS of the target application, e.g., the service availability, is not always guaranteed. Moreover, the cost analysis is over-simplified, without considering the performance variability.

Our study provides a complementary perspective to the related work by characterizing capacity variability experienced by a single VM, with respect to different types of workloads, and rigorously models its aggregate effect on multiple VMs in fulfilling sophisticated QoS while aiming at minimizing cost.

6 Conclusion

Using empirical experiments with a wikipedia system, as well as a Markovian model and numerical analysis, we demonstrate how QoS fulfillment can be best guaranteed with a minimum number of correctly configured VMs deployed in a cloud where VMs suffer from high capacity variability. Our experimental results show that different VM instance sizes can have varying degrees of capacity variability from collocated VMs and that workloads on collocated VMs can impact the capacity of the service VM by up to 35%. Our analytical and numerical results provide not only insight on how an optimal number of VMs should be chosen for a service cluster, but also give counter examples on why simple pessimistic, optimistic, and average-based provisioning of VMs cannot strike an optimal balance of cost and QoS fulfillment in the cloud where performance variability persists. Overall, we provide a systematic and rigorous approach to explore several crucial aspects of VM provisioning for service clusters, i.e., capacity variability, cost structure, and guarantees regarding QoS fulfillment.

7 Acknowledgements

The research presented in this paper has been supported by the Swiss National Science Foundation (project 200021_141002) and by the European Commission (Seventh Framework Programme grant 287746).

References

1. Amazon EC2. http://www.amazon.com/.
2. DaCapo suite. http://dacapobench.org/.
3. M. Björkqvist, L. Y. Chen, and W. Binder. Cost-driven Service Provisioning in Hybrid Clouds. In *Proceedings of IEEE Service-Oriented Computing and Applications (SOCA)*, pages 1–8, 2012.
4. M. Björkqvist, L. Y. Chen, and W. Binder. Dynamic Replication in Service-Oriented Systems. In *Proceedings of IEEE/ACM CCGrid*, pages 531–538, 2012.
5. M. Björkqvist, L. Y. Chen, and W. Binder. Opportunistic Service Provisioning in the Cloud. In *Proceedings of IEEE CLOUD*, pages 237–244, 2012.
6. G. Casale and M. Tribastone. Modelling Exogenous Variability in Cloud Deployments. *SIGMETRICS Performance Evaluation Review*, 40(4):73–82, 2013.
7. Y. Chen, D. Ansaloni, E. Smirni, A. Yokokawa, and W. Binder. Achieving Application-centric Performance Targets via Consolidation on Multicores: Myth or Reality? In *Proceedings of HPDC*, pages 37–48, 2012.
8. J. Dean and L. Barroso. The Tail at Scale. *Commun. ACM*, 56(2):74–80, Feb. 2013.
9. B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift. More for your Money: Exploiting Performance Heterogeneity in Public Clouds. In *SoCC*, pages 20:1–20:14, 2012.
10. K. R. Jackson, L. Ramakrishnan, K. J. Runge, and R. C. Thomas. Seeking Supernovae in the Clouds: A Performance Study. In *Proceedings of HPDC*, pages 421–429, 2010.
11. D. Kossmann, T. Kraska, and S. Loesing. An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. In *SIGMOD Conference*, pages 579–590, 2010.
12. M. Mao and M. Humphrey. A Performance Study on the VM Startup Time in the Cloud. In *IEEE CLOUD*, pages 423–430, 2012.
13. R. Nelson. *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*. Springer-Verlag, 2000.
14. R. Ramacher and L. Mönch. Dynamic Service Selection with End-to-End Constrained Uncertain QoS Attributes. In *Proceedings of ICSOC*, pages 237–251, 2012.
15. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of ACM CCS*, pages 199–212, 2009.
16. J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *PVLDB*, 3(1):460–471, 2010.
17. S. Spicuglia, L. Y. Chen, and W. Binder. Join the Best Queue: Reducing Performance Variability in Heterogeneous Systems. In *Proceedings of IEEE CLOUD*, 2013.
18. K. Tsakalozos, M. Roussopoulos, and A. Delis. VM Placement in non-Homogeneous IaaS-Clouds. In *Proceedings of ICSOC*, pages 172–187, 2011.
19. Wikipedia. http://www.wikipedia.org/.
20. Y. Xu, Z. Musgrave, B. Noble, and M. Bailey. Bobtail: Avoiding Long Tails in the Cloud. In *Proceedings of NSDI*, April 2013.
21. Z. Ye, A. Bouguettaya, and X. Zhou. QoS-Aware Cloud Service Composition Based on Economic Models. In *Proceedings of ICSOC*, pages 111–126, 2012.
22. H. Zheng, J. Yang, W. Zhao, and A. Bouguettaya. QoS Analysis for Web Service Compositions Based on Probabilistic QoS. In *Proceedings of ICSOC*, pages 47–61, 2011.