Effective Capacity Modulation as an Explicit Control Knob for Public Cloud Profitability

Cheng Wang, Bhuvan Urgaonkar, George Kesidis, Pennsylvania State University
Aayush Gupta, IBM Research Almaden
Lydia Y. Chen, Robert Birke, IBM Research Zurich

In this paper, we explore the efficacy of dynamic effective capacity modulation (i.e., using virtualization techniques to offer lower resource capacity than that advertised by the cloud provider) as a control knob for a cloud provider's profit maximization complementing the more well-studied approach of dynamic pricing. In particular, our focus is on emerging cloud ecosystems wherein we expect tenants to modify their demands strategically in response to such modulation in effective capacity and prices. Towards this, we consider a simple model of a cloud provider that offers a single type of virtual machine to its tenants and devise a leader/follower game-based cloud control framework to capture the interactions between the provider and its tenants. We assume both parties employ myopic control and short-term predictions to reflect their operation under the high dynamism and poor predictability in such environments. Our evaluation using a combination of real data center traces and real-world benchmarks hosted on a prototype OpenStack-based cloud shows 10-30% profit improvement for a cloud provider compared with baselines that use static pricing and/or static effective capacity.

Categories and Subject Descriptors: C.3.1 [**Distributed architectures**]: Cloud computing

General Terms: Design; Experimentation; Economics

Additional Key Words and Phrases: Cloud, tenant, dynamic pricing, effective capacity

ACM Reference Format:
Cheng Wang, Bhuvan Urgaonkar, George Kesidis, Aayush Gupta, Lydia Y. Chen and Robert Birke, 2016. Effective Capacity Modulation as an Explicit Control Knob for Public Cloud Profitability. *ACM Trans. Auton. Adapt. Syst.* 0, 0, Article 0 (2016), 26 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

It is well-known that many current data centers are under-utilized (around 6% to 12% in 2012 [NYTimes 2012]). Profitability concerns are changing this, e.g., best practices from Google suggest much improved utilization levels [Barroso and Hölzle 2013; Verma et al. 2015]. In a recent survey [Datacenter2025 2015] of 829 data center professionals, 72% of participants expected information technology (IT) resource utilizations to be at least 60% in 2025. As enterprises and businesses move increasing portions of their IT needs to various cloud computing platforms, the underlying data centers will be forced to operate at higher levels of utilization than seen currently. A key outcome would be increased occurrence of periods where *"demand" exceeds "supply"* (i.e.,

This work is supported, in part, by the following: NSF CAREER award 0953541, NSF CNS 1228717 and 1116626, Swiss National Science Foundation (projects 407540 and 167266), and an IBM faculty award.
Author's addresses: C. Wang, B. Urgaonkar and G. Kesidis, Pennsylvania State University; A. Gupta, IBM Research Almaden; L. Y. Chen and R. Birke, IBM Research Zurich.

DOI: http://dx.doi.org/10.1145/0000000.0000000

tenant needs exceed available resources). Generally speaking, utility providers employ two canonical approaches for incentivizing appropriate customer behavior during such periods: (i) dynamic pricing and (ii) dynamic service/commodity quality.

Utility providers have long employed *dynamic pricing* to signal supply-demand mismatch to their customers (e.g., real-time or coincident peak pricing used by electric utility companies [CoincidentPeak 2013]). Not surprisingly, dynamic pricing has emerged as an area of significant interest in the "computing utility" offered by public cloud platforms, the most prominent real-world example being Amazon EC2 spot instance markets [AmazonSpot 2016]. Numerous research papers have explored cloud's pricing design for profitability (more details in Section 5).

A second approach - the focus of this paper - is based on *dynamic service quality variation*. Towards this, we introduce the notion of the **effective capacity** of a cloud-offered resource type. We define the effective capacity of a resource type as its actual offered capacity (including all of its constituent/primitive resources if any) which may be different from its advertized capacity. [1] Several recent studies report temporal dynamism in effective capacity is already prevalent among public cloud VM offerings - see Sections 2 and 5 for more details.

However, such effective capacity dynamism is *generally a side-effect of other resource management decisions rather than being an explicit control knob* modulated based on anticipating the tenants' tolerance and response to it. The central thesis of this paper is that explicit control of VM effective capacity (complementing dynamic pricing) can help a public cloud provider improve its profitability and should, therefore, be a part of its overall resource management arsenal.

Figure 1 shows our vision of a public cloud that offers a variety of VM types with different degrees of dynamism *both* in prices and effective capacities. We show some examples of VM types offered by Amazon EC2 on this qualitative spectrum. Along the price dynamism axis, most of EC2 VM offerings have relatively static prices (change over months) except spot instances that exhibit frequently changing (over minutes) prices. Along the effective capacity axis, the regular on-demand and reserved instances are *claimed to be* provisioned with a fixed (advertized) capacity (although even these exhibit some effective capacity variation as our measurements in Section 2 show). The recently introduced "burstable" instances, on the other hand, only guarantee a certain baseline CPU capacity beyond which the CPU capacity becomes variable [BurstableVM 2016].

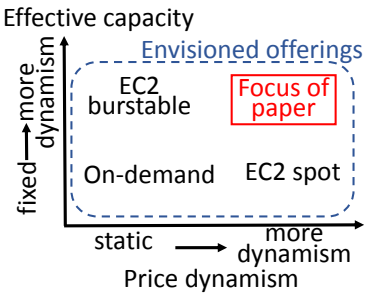


Fig. 1: Our vision of public cloud VM offerings with different degrees of dynamism both in pricing and effective capacity. Also shown are example VM offerings from Amazon EC2 and where they fit within our classification.

The Problem and Challenges: *How should a public cloud provider use dynamic effective capacity as an explicit and strategic control knob (complementing dynamic pricing) to maximize its profit?* The efficacy of explicitly controlled dynamic effective capacity would crucially depend upon the existence of (and adequate volume of) tenants that are capable of gainfully using such VMs and navigating the associated price-performance tradeoffs. We will offer evidence of such tenants both from related work and our own case studies. Equally crucial would be the ability of the cloud provider to estimate tenan-

[1] There could also be ways of expressing effective capacity in terms of workload-specific performance measures, e.g., the throughput offered by a database-as-a-service. Although we focus on virtual machines (VMs) as resource type, many of our ideas can be easily applied to other resources.

t's reaction to its offered style of dynamic effective capacity and incorporate this into its resource allocation decisions. We will offer evidence for feasibility of this via our empirical evaluation based on realistic tenant workloads.

Our Approach: We envision that complementing dynamic pricing, a futuristic cloud would offer a variety of VM types spanning the entire spectrum of Figure 1. Likely many service-level agreement (SLA) types will emerge corresponding to the capacity vs. price trade-offs that diverse tenants may find appealing. In fact, some of this already exists in the immense diversity of VMs offered by current public cloud providers. Many options exist for how these SLAs might express effective capacity dynamism - ranging from guaranteed/fixed capacity to purely best-effort with stochastically guaranteed capacities in between. Viewing SLA design as complementary to our work, we work with a fairly general and expressive SLA format. We assume that VMs in an SLA class are provisioned with a guaranteed baseline capacity plus a best-effort style variable capacity. The tenant chooses appropriate SLA classes for its VMs at a coarse time-scale, and carries out resource procurement within the SLA class (or a combination of multiple SLA classes) at a finer time granularity to leverage effective capacity v.s. price tradeoffs. At the cloud side, the cloud has to estimate the tenant's responses under dynamic pricing and dynamic effective capacity. We find that a useful way to model this setting is via a leader/follower game with the cloud provider being the leader and tenants the followers. The interaction between the cloud and the tenants is multi-step, capturing both the causal relationship and dynamism of price and effective capacity.

Contributions: We make the following contributions:

- We identify dynamic effective capacity modulation as a key explicit control knob, complementing dynamic pricing, for cloud's profit maximization. We show that the two are not interchangeable but complementary to each other through empirical evaluation (Section 4).
- We provide a variety of evidence that dynamic effective capacity already occurs in current clouds and argue for why it might grow and for its potential profitability (Section 2).
- Based on a general and systematic definition of SLA classes, we construct mechanisms for a cloud's decision making in the context of a set of tenants once they have chosen an appropriate SLA class. In particular, we propose a leader/follower game-based cloud control framework using both dynamic effective capacity modulation and pricing for a cloud provider's profit maximization (Section 3).
- We not only carry out trace-driven simulation using real-world tenant demand data, but also provide case studies on an OpenStack-based prototype cloud platform and conduct experimentation with realistic tenant benchmarks to help understand how the proposed system might work in the real world (Section 4). Our key findings are: (a) effective capacity modulation with dynamic pricing offers 10-30% profit improvement over static pricing and/or static effective capacity, (b) when lowering effective capacity, the provider should avoid exposing the tenant to operating regions where performance degradation is unacceptable, (c) inappropriate choice of SLA class may result in poor application performance and such a tenant may find it better to choose an SLA class with more guaranteed capacity and (iv) the provider may find it profitable to even provide differentiated service within the same SLA class if it has good estimates of tenant's performance/price sensitivities.

2. BACKGROUND AND SYSTEM MODEL

We offer several pieces of evidence for effective capacity dynamism in existing cloud systems and provide arguments for why we expect this trend to become more perva-

sive. We then present an SLA framework that our envisioned public cloud might use to systematize its VM offerings. We carry out our modeling and evaluation in Sections 3 and 4 in the context of this framework.

2.1. Explicit Effective Capacity Dynamism

Commercial clouds already offer VMs with some equivalent to our definition of dynamic effective capacity. For example, Amazon EC2 provides "burstable" VMs that only guarantee a certain baseline CPU capacity with the rest marked as "variable". The "pre-emptible" VMs from Google Compute Engine, which can be revoked at any time, and EC2's spot instances, which become unavailable once price exceeds tenant's bid, lead to dynamism in available resource for tenants.

In a alternative evolution of the cloud (than studied here), the provider could allow explicit tenant's participation in effective capacity modulation. E.g., Amazon EC2's C4 instances provide tenants the new option of controlling CPU C-state and P-state to meet their variable resource needs [VM-DVFS 2016]. In our work, tenants infer effective capacity dynamism created by the cloud provider and then respond to it. Regardless, these knobs offer supporting evidence for the growing occurrence of capacity dynamism in the public cloud.

Table I: Benchmarks and descriptions.

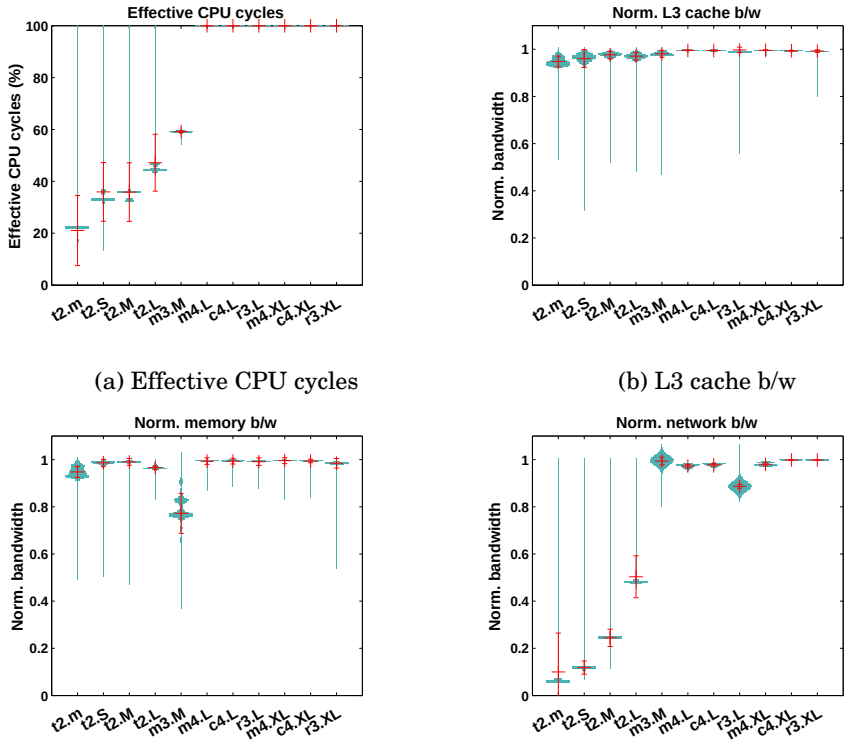
Benchmark	Description	Dominant resource
stress	spawn multiple workers spinning on sqrt()	CPU cycles
pmbw	measure the parallel L3 cache bandwidth of multi-core machines	L3 cache b/w
stream	perform simple memory operations on vectors and measure memory bandwidth	memory b/w
iperf	transfer large chunk of data and measure network bandwidth	network b/w

2.2. Capacity Dynamism in VMs with "Fixed" Capacity

Even for the VM types that are claimed to be provisioned with fixed (advertized) capacity, we still find evidence of effective capacity variation through our own measurements (below) and from a variety of related work, possibly due to the effect of multi-tenancy and the fact that some VMs types may be more aggressively packed than others. It is natural and reasonable to assume that as cloud's utilization level grows, this dynamism will continue to increase, taking the cloud closer to the full spectrum in Figure 1.

Evidence from benchmarking on EC2: For a diverse set of benchmarks with different bottleneck resources (see Table I), we compare the dynamism in their performance (across multiple runs) when they are executed on different types of VMs procured from Amazon EC2. For instances with more than one vCPUs, we run the multi-core versions of these benchmarks. We perform experiments on 11 instance types from Amazon EC2 in the availability zone *us-east-1c*; the `t2.x` instances are burstable while the rest of them are regular on-demand instances which are claimed to be provisioned with fixed capacity. For the burstable instances, we consistently find high variation in not only CPU cycles, but also other resources. In general, the regular on-demand instances have relatively lower capacity variation, with the exception of `m3.medium` (similar variation with burstable instances), indicating that such VMs may have been more aggressively consolidated than others. In addition, almost all regular on-demand instances exhibit high memory bandwidth variation, e.g., almost 50% variation for `r3.XL`, which implies possible larger dynamism in the offered resource capacities due to reasons such as multi-tenancy.

Evidence from other measurements: Plenty of recent work offers evidence of dynamic effective capacity on single or multiple instance types, within a single IaaS cloud



(a) Effective CPU cycles

(b) L3 cache b/w

(c) Memory b/w

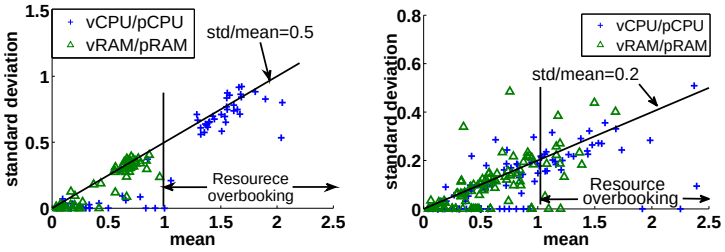
(d) Network b/w

Fig. 2: Violin plot of the resource capacity dynamism of different EC2 instance types inferred/measured from different benchmarks. `t2.x` types represent the burstable instances while the rest of them are regular on-demand instances. All the capacity values are normalized against the maximum capacity of each instance type measured across multiple runs. The effective CPU cycles are estimated using $1 - stolentime$ where the CPU stolen time [StolenTime 2013] reflects the percentage of CPU cycles that the hypervisor steals from the instance running the stress benchmark and could be measured by the Linux top command.

provider or across multiple cloud platforms, resulting in the "unobservability" problem described in [Gandhi et al. 2015]. These temporal/spatial performance variations have been observed across different resource types, e.g., CPU capacity [Wen et al. 2015; Farley et al. 2012], block I/O [Ghoshal et al. 2011; I/O 2012], network latency and bandwidth [Mogul and Kompella 2015; Wang and Ng 2010]. CloudLook [Cloudlook 2016] reports that even under similar advertized capacity, VMs from some providers (e.g., DigitalOcean) demonstrate 10% to 20% more performance variations than other providers, attributable to lower CPU effective capacity. Similarly, differences in CloudHarmony's [Cloudharmony 2016] CCU (CloudHarmony Compute Unit) scores across cloud providers are evidence of effective capacity variation across clouds.

2.3. Why Dynamic Effective Capacity Can Improve Cloud's Profitability

Ideas similar to dynamic effective capacity have long been explored in private settings wherein the provider and the tenants are part of the same organization and are willing to cooperate towards shared goals. Due to willingness to share information, the



(a) Tenant 1 (b) Tenant 2

Fig. 3: Statistics of resource allocation from a private hosted data center. "vCPU/pCPU" and "vRAM/pRAM" denote the ratios of number of vCPU vs. physical cores and (virtual) RAM capacity of all VMs vs. physical RAM capacity of a single physical machine.

workload properties and performance/cost trade-offs may be well-understood allowing the provider to carry out appropriate resource management.

As an example, we collect the resource allocation data of one tenant from a private hosted data center to demonstrate the existence of dynamic effective capacity. For each physical machine, we obtain timeseries of the ratio of number of vCPUs over number of physical cores, and ratio of allocated (virtual) RAM capacity over physical RAM capacity, and show the mean and standard deviation in Figure 3. We find that the effective capacity variation of VMs can be 20% (around the line of $std/mean = 0.2$) on a large portion of the physical machines, and that resource overbooking with dynamic consolidation is already common. Similar observations are seen from other tenants as well.

In a public cloud, however, the cloud's lack of knowledge of tenants' workloads and their resource needs makes the potential profitability opportunities of dynamic effective capacity unclear. Let us consider preliminary arguments to motivate: (i) why a public cloud provider might benefit from offering different effective capacities to different tenants, and (ii) why it might also benefit from varying effective capacities over time. Towards this, we create two tenants, each running Memcached [Memcached 2016] (a popular key-value store)[2] whose dominant/critical resource is the DRAM capacity. Each tenant runs a single VM with 4 vCPUs and 7GB RAM storing 6GB dataset, hosted on an OpenStack cluster described in Section 4.3. We use Yahoo! Cloud Serving Benchmark (YCSB) [YCSB 2016] closed-loop generator to send queries (100% GET) to the tenants. For tenant 1, the key popularity follows exponential distribution with 95% of requests going to 5% of the working set while for tenant 2 it follows Zipf distribution with Zipfian constant equals to 0.99. We vary the effective RAM capacity of the VM[3] and show the achieved peak throughput and the corresponding latency in Figure 4.

As we reduce the effective DRAM capacity of the VM, data items (originally in memory) will be moved to swap space of the host which has much higher latency than DRAM. Therefore, we observe degraded peak throughput and higher latency as effective capacity decreases for both tenants. Since tenant 1's key popularity distribution

[2]The benchmarks we use here and in evaluation are closed-loop workload generators; therefore, the achieved throughput could be less than the arrival rate under high load and low effective capacity in our measurements.

[3]There are various ways to achieve effective capacity modulation at the cloud side, e.g., the "cgroups" command, though created for a different purpose, can be used to change the resource limitations of a VM (as a process) on the hypervisor dynamically.

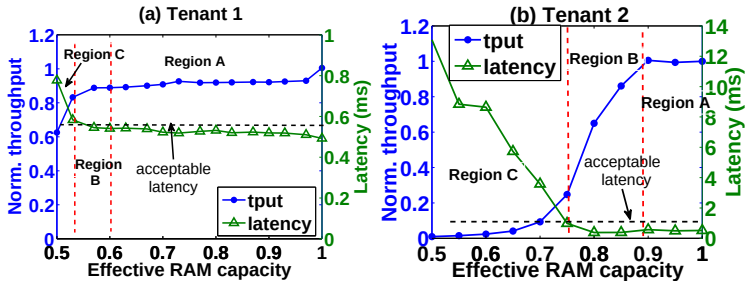


Fig. 4: Average throughput (norm. against peak arrival rate) and corresponding latency for a data caching benchmark with one memcached server and one YCSB workload client. (a) A tenant with exponential key popularity distribution: 95% requests go to 5% of working set (b) A tenant with Zipfian key popularity distribution.

is much more skewed than tenant 2's distribution, lesser data is moved to swap for the former tenant at the same effective capacity level. Thus tenant 1's performance degradation is much "slower" than that of tenant 2. In particular, we find three operation regions (**qualitatively**) for the tenants wherein tenants might react to effective capacity modulation in different ways depending on the perceived performance and their utility models:

- *Region A*: The performance degradation is (statistically) imperceptible to the tenant. It is, therefore, reasonable to expect that the tenant would operate **as if** there is no effective capacity modulation.
- *Region B*: The performance degradation is perceptible but acceptable if there exists a meaningful trade-off against the additional cost of procuring more resources for overcoming this performance degradation. The exact nature of such a trade-off would be tenant-specific.
- *Region C*: The tenant cannot tolerate such performance degradation due to unacceptable revenue loss.

The above regions would vary across tenants depending on their workload properties, e.g., acceptable latency levels, time-varying arrival rates. From Figure 4 we see that tenant 1 does not have region A but observes a very long region B up to 40% reduction of effective capacity. On the other hand, tenant 2 shows all three regions based on its relaxed acceptable latency level. This provides cloud an opportunity for under-provisioning resources without affecting tenant's performance. Furthermore, real-world workloads demonstrate time-varying arrival rates, hence the range of the above regions might also vary with different loads across time. These observations motivates the use of **dynamic** effective capacity by the cloud for its profit optimization. However, the **key challenge** for the cloud's control is dealing with its lack of knowledge of tenant's sensitivity to effective capacity modulation.

2.4. Our SLA Model

There are many possible ways in which such a cloud interface could evolve. In particular, how and what the cloud provider discloses about effective capacity dynamism and what kinds of guarantees its SLA makes can take many forms. In our work, we assume that the cloud provider offers multiple VM SLA classes, each of which is defined as *a guaranteed baseline capacity plus variable capacity out of the advertized capacity*. Examples of today's offerings with explicit SLA guarantees in Section 2.1 are closer to what we advocate and are special cases of our proposal. Our SLA model is also general enough to capture VM types with best-effort capacity guarantees. Another direction

could be that the tenants infer effective capacity (out of claimed capacity) as in Section 2.2. Note that our focus is not SLA design; rather, we want to explore how the provider could leverage dynamic effective capacity for profitability once a specific SLA class has been chosen by the tenants.

We restrict our focus to a single SLA class under one advertized type of VM hosted on a homogeneous subset of physical machines (PMs). However, our model and formulation can be extended to incorporate multiple SLA classes and advertized VM types easily. We assume that other resources (e.g., network bandwidth, software services) to be either free and non-performance bottlenecks or not required for our workloads. We take one PM as one unit of physical resource and assume that the requested/advertized physical resource capacity of one VM is $r(\leq 1)$ (e.g., CPU in GHz, RAM in GB). We define **effective capacity**

$$\eta_{i,t} = \frac{r_{i,t}^{\text{alloc}}}{r}$$

as the ratio of actually allocated/offered resource over the advertized resource capacity, wherein $r_{i,t}^{\text{alloc}}$ is the amount of physical resource that is actually allocated to a VM of tenant i at time-slot t. For example, if the advertized RAM capacity for a certain type of VM is 2GB, but the provider limits the amount of RAM that can be used for a new VM to 1.5GB, then the effective capacity is 0.75. Furthermore, for the single SLA class used in our problem formulation, we denote as η^{LB} the guaranteed baseline capacity. The provider and the tenant may have different valuations for effective capacity. In the above example, it is possible that the tenant's application is not so sensitive to RAM capacity modulation, or that the tenant procures resource for its peak load which only lasts for relatively short duration. In such a case, the tenant might not even be aware of such effective RAM capacity modulation. The aforementioned two tenants can be viewed as a concrete example of different tenants' tolerance of such effective capacity variation.

Note that we are making the simplifying assumption that there is a single VM resource (which could itself be CPU capacity, memory capacity, memory bandwidth or IO bandwidth) that is a bottleneck in the following sense: for the assumed placement of workloads to a VM, this bottleneck resource saturates with spare capacity remaining for all other resources. In this context, η^{LB} becomes the baseline capacity of the bottleneck resource. For example, in our Web serving case study (Section 4), this bottleneck resource is CPU capacity (rate), whereas in our data caching case study it is memory capacity (space). In both case studies, we scale CPU capacity proportionally to memory capacity. This is also aligned with the fact that the CPU capacity of a VM is often strongly positively correlated with its memory capacity on public cloud offerings.

Dynamic effective capacity should not be thought of as "cheating." One form it could take could indeed be as suggested in [Mogul and Kompella 2015] wherein Google researchers propose to modulate network bandwidth differently (and potentially to an arbitrary large degree) for tenants with different network latency sensitivities to save cloud's costs. However, we pursue such modulation *in the context of SLAs* wherein certain aspects of the capacity dynamism are agreed upon over some time-scale and the actual dynamism is not arbitrary but rather limited to the extent stipulated in this SLA.

3. SYSTEM DESIGN AND FORMULATION

3.1. High-level Overview

Figure 5 shows a high-level overview of our cloud eco-system. We choose to work with a time-slotted system wherein the cloud provider and its tenants make strategic control

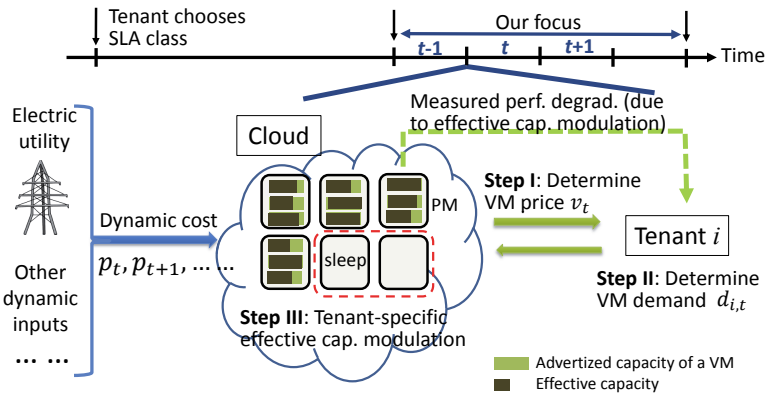


Fig. 5: Illustration of our cloud eco-system and the proposed game framework. Cloud's dynamic costs come from a combination of various dynamic inputs, e.g., dynamic energy price.

decisions at the beginning of each time-slot (e.g., one hour), indexed by t. Based on their relative temporal ordering, we identify three steps making up the overall decision-making in our cloud eco-system. In **step I**, the provider estimates the tenants' VM demands and reactions to changes in price and capacity and decides the VM price v_t to maximize its "myopic" profit. In **step II**, given v_t, previous performance observations and inferred effective capacity, tenant i decides and presents the provider with its VM demand $d_{i,t}$. In **step III**, once tenants have submitted their VM demands, the provider determines the effective capacity $\eta_{i,t}$ and implements appropriate control to enforce this effective capacity. Note that although all tenants see the same VM price during a given time-slot, the effective capacity could be different both across time and across tenants.

3.2. Game-based Cloud Control Framework

In this section, we expand the above ideas to devise a leader/follower game-based cloud pricing and effective capacity modulation framework and study the interactions between the participants (i.e., the provider and the tenants).

3.2.1. Step I: Cloud's pricing control. Denote as $d_t(=\sum_{i\in I} d_{i,t})$ the aggregate VM demand from all tenants during time-slot t, wherein I is the set of all tenants. At the start of time-slot t, the provider predicts its revenue $v_t\hat{d}_t - p_t m_t$ where $\hat{d}_t$ is the prediction of d_t, $p_t m_t$ is the operational cost with m_t denoting the number of active PMs and p_t the dynamic cost of keeping one PM operational which could come from multiple sources (e.g., energy costs charged by electric utility, network bandwidth costs charged by ISPs). We assume that the provider estimates the number of PMs needed during time-slot t as $m_t = \lceil \sum_i \hat{d}_{i,t} r \eta_{i,t} \rceil$.

Demand Prediction: The provider's prediction of tenant i's VM demand $d_{i,t}$ (denoted as $\hat{d}_{i,t}$) depends on (i) temporal effects (e.g., time-of-day or seasonal patterns), (ii) VM price (which affects the tenants' responses), and (iii) effective capacity $\eta_{i,t-1}$ (which affects tenant i's performance and thereby its response). Since the provider knows that the tenants have to *infer* effective capacity in previous time-slots from previous performance observations and then use these to predict effective capacity in the next time-slot (which will be implemented only after step III), it can simply assume that $d_{i,t}$ depends on VM price v_t and the previous effective capacity $\eta_{i,t-1}$; thus the estimated demand $\hat{d}_{i,t} = g_{i,t}(\{d_{i,s}\}_{s=1}^{t-1}, \{v_s\}_{s=1}^{t}, \{\eta_{i,s}\}_{s=1}^{t-1})$, a predictive function with

parameters recursively obtained from appropriate prediction techniques (e.g., machine learning or interpolation as used in our evaluation) with historical data[4]. We set s to take only one past time-slot $t-1$ throughout this paper for (i) keeping our analysis simple and also (ii) since older observations are likely to be much less important than the latest one in such a highly variable environment.

Provider's Pricing Design Problem. We assume that the provider optimizes its profits over $\tau \geq 1$ successive time-slots. Again, for ease of presentation and empirical analysis, we restrict ourselves to $\tau = 2$ throughout the paper while noting that extensions to larger τ are easily done using a standard dynamic programming framework. At the beginning of time-slot t, assuming that the provider is well-informed of its dynamic costs p_t and p_{t+1}, the provider's optimization problem can be written as follows (**Problem P1**):

$$\max_{v_t, v_{t+1}, \eta_{i,t}} v_t \sum_i \hat{d}_{i,t} - p_t m_t + v_{t+1} \sum_i \hat{d}_{i,t+1} - p_{t+1} m_{t+1}$$

Subject to

$$0 \leq \hat{d}_{i,s} = g_{i,s}(d_{i,s-1}, v_t, \eta_{i,s-1}),\ s = t, t+1,\ \forall i$$
$$\eta^{\mathrm{LB}} \leq \eta_{i,t} \leq 1,\ \ \forall i$$
$$m_s = \lceil \sum_i \hat{d}_{i,s} r \eta_{i,s} \rceil \leq M,\ s = t, t+1$$
$$\eta_{i,t+1} = \eta_{i,t},\ \forall i$$

wherein $\eta_{i,t-1}, d_{i,t-1}$ are known from the previous time-slot. M is the total number of PMs available ("alive" plus "sleeping"). The first constraint captures the provider's estimation of tenants' demands and responses given the VM prices, the effective capacity, and their resource demands. The second constraint captures that the provider would never allocate more resources than those requested by a VM, i.e., $r_{i,t}^{\mathrm{alloc}} \leq r$, and $\eta_{i,t}$ is lower bounded by η^{LB} (guaranteed baseline capacity of the SLA class considered). The third constraint guarantee that the resource capacity of the provider is not exceeded (if exceeded after real $d_{i,t}$ is revealed at the end of step II, the cloud has to do more aggressive effective capacity modulation in step III). Since $\eta_{i,t+1}$ will affect demand $d_{i,t+2}$ which is not considered in the myopic objective, we need to set a terminal condition for $\eta_{i,t+1}$ to avoid the trivial solution ($\eta_{i,t+1} = \eta^{\mathrm{LB}}$). We set $\eta_{i,t} = \eta_{i,t+1}$ in the last constraint which corresponds to setting the control horizon to 1 in model predictive control.

The provider only publishes the price v_t after solving the above optimization problem; the final effective capacity $\eta_{i,t}$ will emerge from a re-calculation in step III after tenants' demands are revealed in step II.

Computational complexity of the cloud's control problem. In general, **P1** is a nonlinear optimization problem due to the fact that the function $g_{i,s}(.)$ (relationship between the tenant's demand, the cloud's VM price and effective capacity) in the first constraint may take arbitrary form. In our evaluation, we approximate it using a linear regression model. Therefore, **P1** becomes a convex optimization problem with linear constraints, which can be solved efficiently. In particular, if the look-ahead window is τ (inspired by the model predictive control approach) and the number of tenants is I, we will have τ variables for the cloud's prices and $(\tau-1)I$ variables for the tenant-specific effective capacity ratio. In addition, we have $(\tau+1)I$ non-boundary constraints.

[4]Overheads of maintaining models for individual tenants can be reduced by aggregating tenants into groups based on their price/performance sensitivity.

If we view τ as a constant, the problem size increases linearly w.r.t. the number of tenants. We can further reduce the computational overhead by aggregating tenants into groups based on their price/performance sensitivity.

3.2.2. Step II: Tenants' Strategic Behavior. Upon receiving VM price v_t and using its own measurements of its performance metrics in the previous time-slot, each tenant has to determine the number of VMs it needs from the cloud based on the inferred effective capacity $\hat{\eta}_{i,t}$ and its own predicted raw demand. Denote as $D_{i,t}$ the raw physical resource demand of tenant i. We assume that any unsatisfied physical resource demand $(D_{i,t} - d_{i,t}r\eta_{i,t})^+$ will cause the tenant to incur a demand "dropping" cost due to performance degradation and/or revenue loss (e.g., being able to serve less clients' requests), which should be embedded into the tenant's utility function during evaluation. We also assume that admitting more than $D_{i,t}$ does not result in extra profit, thus $d_{i,t}r\eta_{i,t} \leq D_{i,t}$.

Tenant's Utility Model: A tenant's utility depends on the effective capacity it obtains from the provider, which affects the performance of the tenant's application (including the demand from its own clients). Denote as $U_i(d_{i,t}, \eta_{i,t})$ the tenant's utility function, which is non-decreasing and concave in both $d_{i,t}$ and $\eta_{i,t}$, e.g.,

$$U_i(d_{i,t}, \eta_{i,t}) = b_2 \log(b_1 d_{i,t} r\eta_{i,t} + 1) - b_0(D_{i,t} - d_{i,t}r\eta_{i,t})$$

wherein the first term implies that the revenue increases as the tenant procures more resources, and the second term reflects the penalty due to "dropping" demand. Such utility functions assuming diminishing marginal revenue are commonly assumed and even though we do not have access to exact forms for these, we expect to find qualitatively meaningful insights from using them.

Estimation of Effective Capacity: Denote as $\phi_{i,t}$ the performance metric (e.g., average latency) of tenant i during time-slot t, as $\lambda_{i,t}$ the corresponding average arrival rate per VM. We assume that the relationship between the performance metric and effective capacity can be captured by the function $\phi_{i,t} = F_i(\eta_{i,t}, \lambda_{i,t})$, which can be learned via suitable techniques. Note that $\eta_{i,t} \geq \eta^{LB}$ according to the cloud's SLA. Extensive related work exists on both offline/online profiling to obtain empirical relationships between latency vs. request arrival rate vs. effective capacity for e-commerce sites [Padala et al. 2007; Stewart and Shen 2005; Klein et al. 2014; Urgaonkar et al. 2002], throughput vs. effective capacity for MapReduce-like workloads [Herodotou and Babu 2011; Zhang et al. 2014], response time vs. effective capacity for streaming servers [Brandt et al. 1998] and many more. We assume that given the actual arrival rate $\lambda_{i,t-1}$ and measured performance value $\phi_{i,t-1}$, the tenant infers the $\eta_{i,t-1}$ via interpolation and estimates the new effective capacity via a suitable prediction technique, e.g., $\hat{\eta}_{i,t} = \beta_1\eta_{i,t-1} + \beta_2\eta_{i,t-2}$ if $\hat{\eta}_{i,t} \geq \eta^{LB}$ and $\hat{\eta}_{i,t} = \eta^{LB}$ otherwise, wherein parameters β_1, β_2 can be adaptively obtained by using historical data.

Tenant's Myopic Control Problem: Given the VM price v_t and estimated effective capacity $\hat{\eta}_{i,t}$, we express tenant i's profit maximization problem as follows (**Problem P2**):

$$\max_{d_{i,t}} U_i(d_{i,t}, \hat{\eta}_{i,t}) - v_t d_{i,t}$$

Subject to

$$d_{i,t}r\hat{\eta}_{i,t} \leq D_{i,t}$$

A real-world tenant might also define performance bounds (e.g., latency upper bounds) as part of its SLA requirement, instead of implicitly incorporating revenue

loss due to performance degradation into a "dropping" cost as we have done. We evaluate such tenants through our case studies in Section 4.

3.2.3. Step III: Provider's Additional Control. In this step, since the provider already observes the tenants' VM demands $d_{i,t}$, it can put $d_{i,t}$ and v_t back into **Problem P1** and solve it again to obtain more profitable (and feasible) effective capacities $\eta_{i,t}$. Thus tenant i's VMs will get capacity $r_{i,t}^{\text{alloc}} = \eta_{i,t} r$, and the total number of PMs needed in the **ideal case** (no migration costs) would be $m_t = \lceil \sum_i d_{i,t}\eta_{i,t} r \rceil$. We evaluate the proposed framework under the ideal scenario in Section 4.

3.3. Understanding "Optimal" Solutions

In this section, we derive the "best responses" for both the tenant and the cloud with simplifying assumptions to get preliminary understanding of the interactions between them. We present all proofs in the Appendix.

CLAIM 1. *Assume that (i) the tenant i is able to predict the effective capacity $\eta_{i,t}$ offered by the provider, (ii) its VM demand is continuous, and (iii) its utility function $U_i(d_{i,t}, \eta_{i,t}) = u_i(d_{i,t} r \eta_{i,t})$ is strictly concave and increasing. Then there exists the following unique optimal solution to* **Problem P2***:*

$$d_{i,t}^* = \frac{1}{r\eta_{i,t}} \min\{D_{i,t}, (u_i^{'})^{-1}(x_{i,t})\}$$

where $x_{i,t} = \frac{v_t}{r\eta_{i,t}}$; $(u_i^{'})^{-1}(\cdot)$ is the inverse function of $u_i^{'}(\cdot)$.

In particular, if $u_i(d_{i,t} r\eta_{i,t}) = b_2 \log(b_1 d_{i,t} r\eta_{i,t} + 1) - b_0(D_{i,t} - d_{i,t} r\eta_{i,t})$, the optimal solution becomes

$$d_{i,t}^* = \min\{\frac{D_{i,t}}{r\eta_{i,t}}, \frac{b_2}{v_t - b_0 r\eta_{i,t}} - \frac{1}{b_1 r\eta_{i,t}}\}$$

We obtain several useful insights from Claim 1 and the special case above: (i) The optimal amount of effective capacity $d_{i,t}^* r\eta_{i,t}$ only depends the ratio of $x_{i,t} = \frac{v_t}{r\eta_{i,t}}$ if there is large enough raw demand $D_{i,t}$. This implies that even if the cloud only provides very low effective capacity, the tenant might still maintain its demand of effective capacity as long as the VM price is also low to compensate for the its performance degradation. (ii) If the parameters (b_1, b_2) of tenant's utility function are smaller, which means the tenant makes less revenue by procuring VMs from the cloud, it will be more sensitive to the price/performance changes, and even a small rise in price might result in significant demand reduction.

CLAIM 2. *Assume that (i) the provider has a single tenant, (ii) the provider has perfect knowledge of its tenant's raw demand and optimal responses as in Claim 1, and (iii) the provider's control decision variables are continuous. Then its optimal solution (v_t, η_t) is not unique. However, there exists an optimal ratio of $x_t = \frac{v_t}{r\eta_t}$ that maximizes the cloud's profit.*

Claim 2 implies that a single control knob would be enough for the cloud's profit maximization, given that it has perfect knowledge of the single tenant's raw demand and optimal responses. However, the advantage of using effective capacity as a second control knob, even for the case where price and effective capacity are interchangeable, still holds since **it can serve as a correction to one already published price**. More importantly, with multiple tenants and less predictable tenant demands and their reactions to price/performance changes, the cloud may not be able to maximize its profits

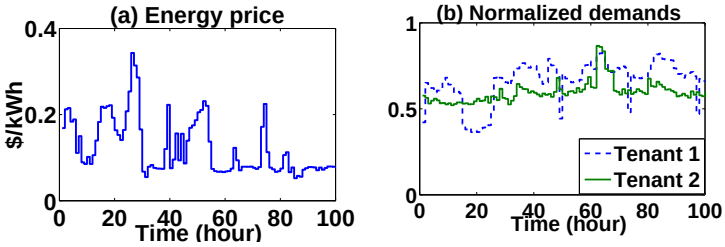


Fig. 6: (a) Hourly energy prices from an electric utility in [Ontario-electric 2014]. (b) Two tenants' normalized effective capacity demands (CPU usage in MHz) from a large commercial data center.

by only using a single control knob. We demonstrate this discrepancy between theory and practice in Section 4.

4. EVALUATION

We evaluate the proposed game framework in two complementary ways. In Section 4.1, we carry out trace-driven simulations using month-long tenant demand data from a large commercial data center. Since these traces do not provide performance measurements, we present two case studies deployed on an OpenStack-based prototype cloud platform in our lab involving live tenant workloads in Sections 4.2 and 4.3.

4.1. Trace-driven simulation

4.1.1. Experiment setup. **Provider Configuration**. We emulate a public cloud platform whose dynamic costs p_t of keeping a PM operational for an hour is proportional to its energy costs which are based on the day-ahead hourly energy price from an electric utility in [Ontario-electric 2014] (Figure 6(a)). We attribute 10% of the overall operation costs of the provider to these energy costs. Each PM has 8 cores and 16GB RAM. We assume that the full resource capacity of a PM is one unit, and the advertized capacity of each VM is $r = 1/4$ (i.e., a VM is assigned two cores), which means up to four VMs can be packed on the same PM with full capacity. The range of effective capacity $\eta_{i,t}$ is $[0.5, 1]$ with $\eta^{LB} = 0.5$. We predict tenants' aggregate responses to price and capacity variations, using scattered interpolation on the predictors described in Section 3. The training set does not overlap with the dataset that we use in our game.

Tenant Configuration. We pick two tenants' CPU usage (in MHz) time series (30-day trace) from a commercial production data center (Figure 6(b)) as their raw effective capacity demand. We choose the log-form utility function as described in Section 3.2. We assume that the tenant can directly predict effective capacity using the following predictor: $\hat{\eta}_{i,t} = 0.5\eta_{i,t-1} + 0.5\eta_{i,t-2}$. To differentiate their performance/price sensitivity, we set smaller b_2 for tenant 1 which implies that tenant 1 makes less revenue from VMs and thus is more sensitive to higher VM price and/or reduced effective capacity. We expect tenant 1 to be offered higher effective capacity than tenant 2.

Baseline. We create baseline "*OPT*" based on myopic objective assuming that the cloud has perfect knowledge of tenant's demands and responses.

4.1.2. Impact of Dynamic Pricing. In Figure 7(a), we evaluate the impact of dynamic pricing without varying effective capacity. We choose three static prices (*High*, *Medium*, *Low*) representing $(1.5, 1.0, 0.5)$ times the average price generated from *Dyn*, which employs dynamic pricing with full capacity. The profits have been normalized w.r.t. *OPT* profit. We observe that dynamic pricing is able to provide 74% of the profit as compared to the *OPT*. These static pricing schemes not only reduce the profits but also increase

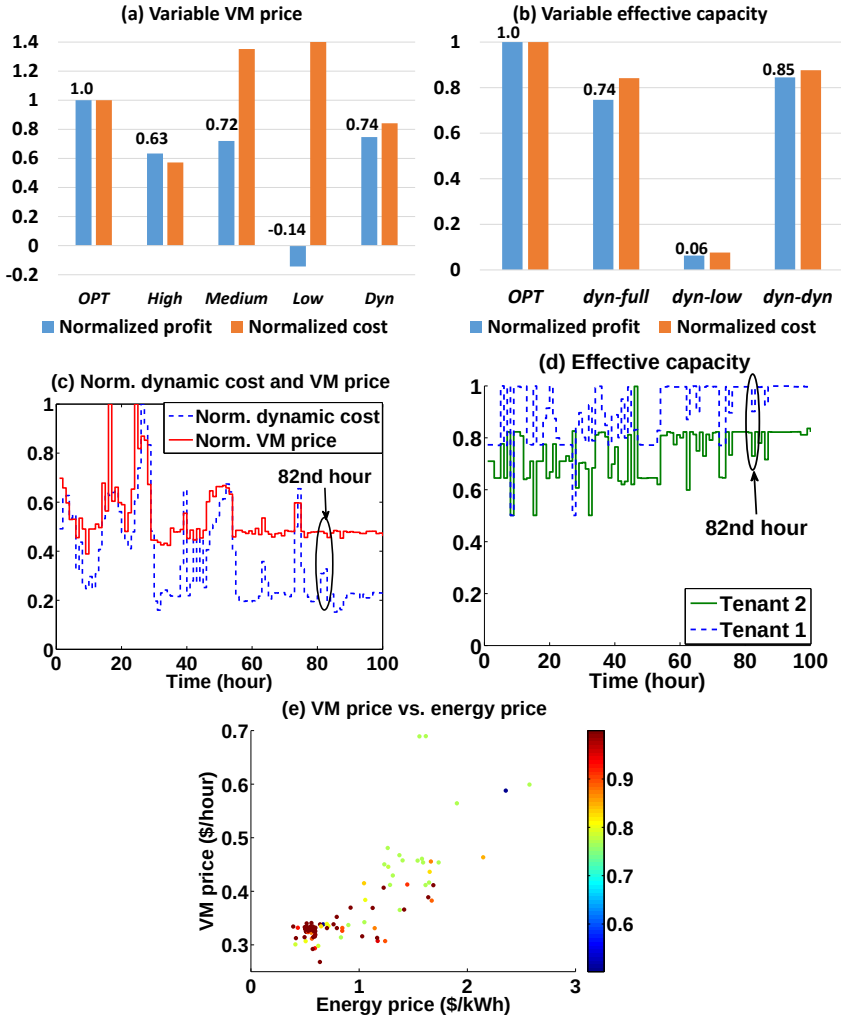


Fig. 7: Provider's normalized profit and cost under (a) dynamic vs. static pricing, (b) dynamic vs. static effective capacity with dynamic pricing. Profits and costs are normalized w.r.t. the profit and cost for *OPT*, respectively. *dyn*-{*cap*} represents strategies with dynamic pricing (*dyn*) where *cap* can be full (*full*), half (*low*) or dynamic (*dyn*) effective capacity. (c) VM price norm. against peak VM price. (d) Effective capacity offered to the two tenants. (e) VM price vs. energy price. The color of the points represents the effective capacity for tenant 1.

the provider's costs in some cases. For instance, *Low* hurts the profits by setting VM price to only half the average price of *Dyn*. The provider can attract a large demand, thereby incurring high cost. However, due to the corresponding cheap VM price, it is not able to recover the cost of provisioning VMs, resulting in a loss (-14%).

4.1.3. Impact of Effective Capacity Modulation. From Figure 7(b), we find that the game with dynamic effective capacity and prices, i.e., *dyn-dyn*, provides 11% higher profit than that from dynamic pricing only (*dyn-full*). This happens because by having the extra degree-of-freedom in control knobs, the cloud is able to lower VM prices, attracting more demand, yet maintaining costs similar to the strategy with just dynamic pricing.

Reducing the effective capacity significantly (*dyn-low*) causes tenants to reduce their demand significantly, resulting in almost no profit for the provider.

We observe provider's capacity violation in *dyn-full*, i.e., the total demand exceeds capacity, whereas no violation happens in *dyn-dyn*, which is because in step III of the game the provider is able to carry out effective capacity modulation after the real demand is revealed.

To see the effectiveness of the proposed framework without capacity limits (hence mimicking the behavior in an under-utilized cloud platform), we remove the provider's resource capacity constraint in another set of experiments, and we observe similar profit improvement over baselines.

4.1.4. A Closer Look at the Game Output. Figures 7(c)(d) show timeseries of VM price and effective capacity generated from the game. VM price tracks dynamic cost closely but is not exactly the same. This is expected since both VM price and effective capacity can impact tenants' demands (thereby affecting cloud's costs). For example, at around 82nd hour, although dynamic cost is high, the cloud still sets low VM price to encourage tenants' demand while lowering the effective capacity to reduce costs. We also observe that the effective capacity offered to tenant 2 is almost always less than that of tenant 1, which is because tenant 1 is more sensitive to performance/price (with smaller b_2).

In Figure 7(e), we show a scatter-plot of VM price vs. energy price with the colors of the points representing the effective capacity for tenant 1. We find that the cloud sets higher VM price as the energy price increases in general, since we assume that the cloud's Opex is proportional to the energy prices. In addition, when the energy price is low (e.g., less than $1\$/kWh$), implying low dynamic cost, the cloud tends to both set low VM price and allocate high effective capacity to attract more demands; however, there are exceptions where the cloud sets too low a VM price in Step I (hurting its profitability) and then compensates itself with low effective capacity for the tenant in Step III to make more profit (e.g., the green points when energy price is less than $1\$/kWh$). On the other hand, when the energy price becomes too high, the cloud could set high VM price in Step I, followed by low effective capacity in Step III if needed (e.g., the green points when energy price is greater than $1\$/kWh$), to reduce its Opex.

4.1.5. Tenant's Profitability. Although the focus of our framework design is to maximize the cloud's profit, it is still of interest to both the provider and the tenants to see the tenant's profitability in this framework. We take tenant 1 as an example and show its profits and costs under different strategies (normalized against those of *OPT*) in Figure 8. First, we fix the effective capacity to the full capacity and show the results in Figure 8(a). We find that the tenant obtains the highest profit under *Low*, which however leads to negative profit for the cloud. Similar to Figure 7(a), the tenant's profit and cost under both *Medium* and *Dyn* are close to those under *OPT*; however, the cloud would not be able to set the VM price for *Medium* without applying the framework in advance.

Second, we apply dynamic pricing with different strategies for effective capacity modulation and show the results in Figure 8(b). *dyn-low* results in the worst profit for the tenant, which is expected since the tenant's application performance would be greatly compromised. Surprisingly, *dyn-dyn* yields higher (and closer to optimal) profit than *dyn-full* which always provisions full capacity to the tenant. This is because of the flexibility in Step III: the cloud could set lower VM prices together with lower effective capacity to attract more tenant's demands (thus possibly more profit for the tenant) while still maximizing its own profit. However, in *dyn-full* the cloud has to stick to its inappropriate VM price (usually high price to avoid high Opex or due to misprediction) in Step I, which leads to higher costs for the tenant. Therefore, even from the tenant's

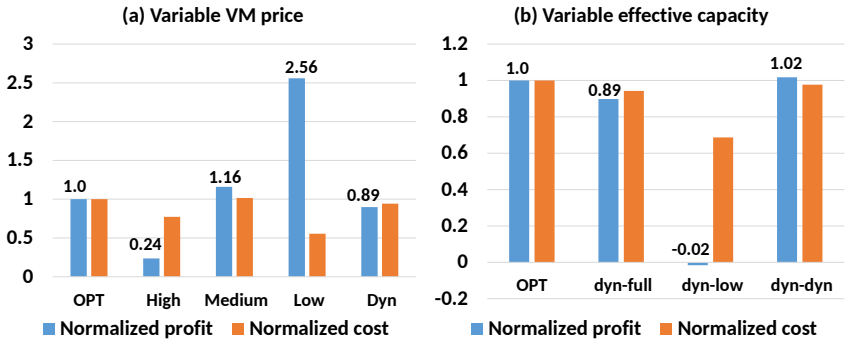


Fig. 8: Tenant 1's normalized profit and cost under (a) dynamic vs. static pricing, (b) dynamic vs. static effective capacity with dynamic pricing. Profits and costs are normalized w.r.t. the profit and cost for *OPT*, respectively.

perspective, it might be attractive if the cloud could use effective capacity modulation as a corrective knob in Step III.

Key insights: (i) Dynamic effective capacity, if combined with dynamic pricing, can further improve the provider's profitability. (ii) The cloud can avoid capacity violation by having effective capacity modulation after the real demand is revealed, which is advantageous over no effective capacity modulation when it is overloaded. (iii) Having learnt tenants' performance sensitivity, the provider can use it to provide differentiated service to tenants with the same guaranteed baseline capacity.

4.2. Case Study I: Web Serving

We present a case study based on the Wikipedia Web server benchmark in [MediaWiki 2016], as a cloud tenant whose dominant resource is CPU. Our goal is to provide guidance on how such a tenant can react to dynamic pricing and effective capacity modulation, as well as insights on how the application performance is affected by different operation regions as discussed in Section 2.

4.2.1. Experiment setup. **Provider Configuration**. We carry out our case study on a small cluster of servers with each PM having 8 cores and 16GB RAM.

Tenant Configuration. We assume that in addition to determining VM demand $d_{i,t}$, this tenant can also drop requests to improve performance of admitted requests when the predicted effective capacity is too low or price is too high. Denote as $\Lambda_{i,t}$ the raw request arrival rate during time-slot t, as $\lambda_{i,t}$ the admitted arrival rate of each VM, assuming perfect load balancing across replicated VMs (each VM has the same full database and the workload is read-only). Then the total admitted arrival rate is $d_{i,t}\lambda_{i,t}$ ($d_{i,t}\lambda_{i,t} \leq \Lambda_{i,t}$). We denote as $h_{i,t}$ the actually achieved throughput on a single VM, as $\phi_{i,t}$ the corresponding latency. We assume that $\Lambda_{i,t}$ is known or have been predicted using short-term predictive models. Given prediction of effective capacity

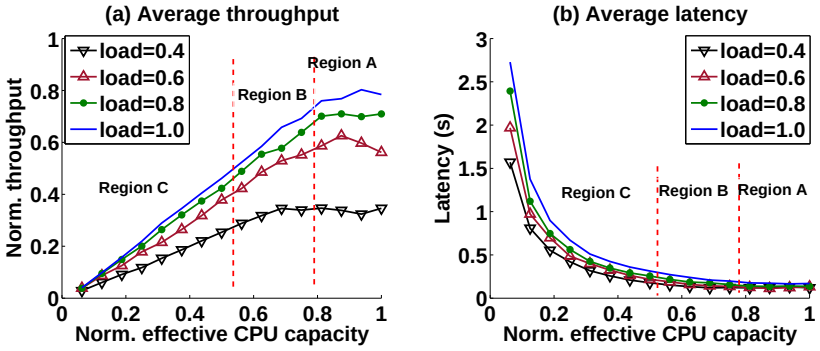


Fig. 9: Average throughput (a) and average latency (b) with a Wikipedia Web server under different load. "load" is norm. w.r.t. the peak arrival rate on a single VM with 8 vCPUs and 12GB RAM.

$\hat{\eta}_{i,t}$, we maximize the tenant's profit as follows (**Problem P3**):

$$\max_{d_{i,t},\,\lambda_{i,t}} \; b_2 \log(b_1 d_{i,t} h_{i,t} + 1) - b_0(\Lambda_{i,t} - d_{i,t}\lambda_{i,t}) - v_t d_{i,t}$$

Subject to

$$h_{i,t} = F_i^h(\hat{\eta}_{i,t}, \lambda_{i,t})$$
$$d_{i,t}\lambda_{i,t} \leq \Lambda_{i,t}$$
$$\phi_{i,t} = F_i^\phi(\hat{\eta}_{i,t}, \lambda_{i,t}) \leq \bar{\phi}_i$$

The first two terms in the objective are the tenant's utility model, with the first term denoting the revenue collected by achieving actual throughput $d_{i,t}h_{i,t}$ (the $log(.)$ function is commonly used for capturing diminishing returns), and the second term denoting the revenue loss due to dropping requests. $F_i^h(.)$ and $F_i^\phi(.)$ denote the relationship of performance metrics vs. effective capacity and admitted arrival rates, respectively (as discussed in Section 3). $\bar{\phi}_i$ is the upper bound of latency; so the last constraint reflects the SLA requirement of this tenant.

Obtaining $F_i^h(.)$ and $F_i^\phi(.)$: We run the benchmark on a single VM with 8 vCPUs and 12GB RAM. A client generates Web page requests with Zipf distribution (Zipf constant=1). We vary the CPU resource allocated to this VM under different loads and show average throughput and latency in Figure 9, which we use to fit $F_i^h(.)$ and $F_i^\phi(.)$ via interpolation. We assume that this tenant can infer effective capacity from previous measurements by interpolation, i.e., $\eta_{i,t-1} = H_i(\phi_{i,t-1}, \lambda_{i,t-1})$ wherein $H_i(.)$ is the interpolating function also obtained from Figure 9. We scale the CPU usage (in MHz) demand of tenant 1 in Figure 6(b) to generate raw request rates $\Lambda_{i,t}$. The **work flow** of the tenant is: At the beginning of time-slot t, the tenant first predicts new effective capacity via techniques discussed above, then solve the optimization problem **P3** to determine number of VMs to procure and the request admission rates.

4.2.2. Provider's Profitability. Figure 10(a) shows the provider's normalized profits under different strategies. We observe effects similar to those in our trace-driven simulation: having dynamic effective capacity (*dyn-dyn*) improves the provider's profits by almost 10% over dynamic pricing with full effective capacity (*dyn-full*). Unlike the trace-driven simulation, we do not observe provider's capacity violation in *dyn-full* (assuming cloud has limited capacity). This is because the provider can still set much higher VM prices to discourage tenant's demand when it is overloaded. This is also con-

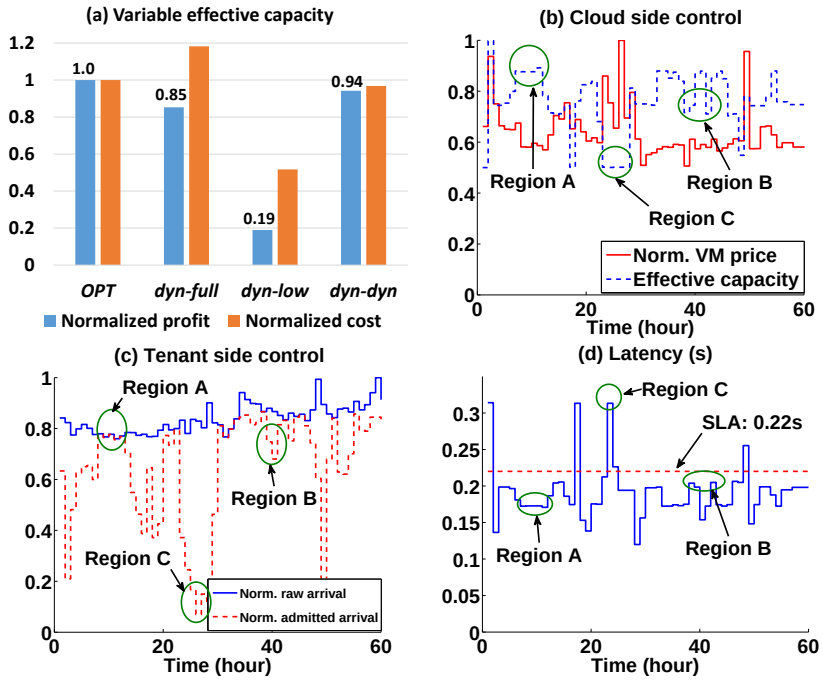


Fig. 10: (a) Normalized profits and costs of the provider with the Web serving tenant. (b) Normalized VM price (w.r.t. peak VM price) and effective capacity under *dyn-dyn*. (c) Web serving tenant's raw arrival and admitted rates. (d) Achieved latency. SLA requirement is 0.22s.

sistent with Claim 2 that a single control knob might be enough to regulate tenant's demand if there is only one tenant. However, in the next use case with two real-world tenants, we expect to see capacity violations again from *dyn-full*.

4.2.3. Tenant's Performance. We show timeseries of VM prices and effective capacity under *dyn-dyn* in Figure 10(b). The corresponding request admission rate and latency for the tenant are shown in Figures 10(c)(d), respectively. As discussed in Section 2.3, we qualitatively define three operation regions for the tenant's slackness in effective capacity in Figure 9, and also find examples of such regions in Figure 10. In region A, this tenant might not be aware of performance degradation down to 80% effective capacity, so it admits all raw arrivals as if the cloud is providing full capacity. However, in region B, the tenant starts to perceive degraded performance, e.g., achieved throughput is lower than expected due to cloud's effective capacity modulation, but the tenant's overall profitability is still acceptable. So the tenant starts to drop requests in order to meet its SLA requirement. Correspondingly, we don't observe any SLA violations in time-slots corresponding to regions A and B in Figure 10(d). When the effective capacity reaches region C, the tenant's performance (and revenue) starts to drop significantly; therefore, it has to drop large portions of raw arrivals to maintain profitability. However, whenever the tenant falls into region C, there is an SLA violation as in Figure 10(d). We do not observe any SLA violations with *OPT*, which implies that SLA violations might be further reduced if the tenant were able to better predict the effective capacity offered by the provider.

Even with *dyn-low*, wherein the cloud only provides half effective capacity, we don't observe any SLA violations, which appears unexpected. This is because the tenant

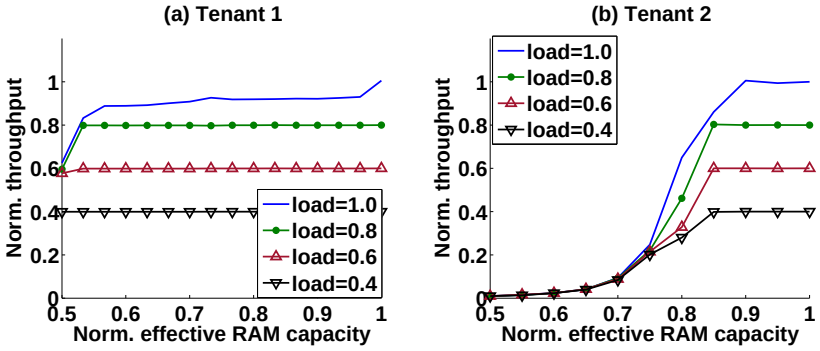


Fig. 11: Average throughput from a data caching benchmark with one memcached server and one YCSB workload client under different loads and key popularity distributions. Here "load" is normalized w.r.t. the peak load on a single VM with 4 vCPUs and 7GB RAM. (a) Exponential distribution with 95% request going to 5% keys. (b) Zipfian distribution with Zipf constant=0.99.

is able to perfectly predict this static effective capacity and reduce its VM demand and admission rates accordingly. So the "good" performance of *dyn-low* comes at the expense of low admitted demand (and low revenue for the cloud henceforth), which is also consistent with our observation in Figure 10(a) that *dyn-low* has the least profits and costs.

Key insights: (i) We find evidence that the provider's profitability can be improved by dynamic effective capacity modulation for a realistic performance-sensitive tenant. (ii) Tenant can use offline performance profiling to facilitate effective capacity inference. (iii) When lowering effective capacity for improving its profits, an important concern for the provider is to avoid exposing the tenant to operating regions where performance degradation is unacceptable. In general, detecting such regions would be tenant-specific and hence challenging.

4.3. Case Study II: Data Caching

We host two tenants based on a real-world key-value store benchmark with the same configuration in Section 2.3, for which the critical resource is memory capacity. The two tenants have different key popularity skewness, which results in different tolerance to effective capacity modulation. Our goal is to see how the provider treats tenants differentially, and how tenant's performance is affected by the provider's control.

4.3.1. Experiment Setup. **Provider Configuration**. We carry out our case study on a small OpenStack cluster consisting of 2.66GHz PMs with 12 cores and 16GB DRAM.

Tenant Configuration. We assume that both tenants can determine the number of VMs to procure and arrival rates to admit at the beginning of each time-slot. They have the same utility function (and parameters), and solve the same optimization problem (**Problem P3**), with the only difference in their $F_i^h(.)$ and $F_i^\phi(.)$ (due to different key popularity distribution). Such configurations help us focus on evaluating the impact of tenants' performance sensitivity on the provider's control.

For performance profiling, we run the same benchmark and use the same workload configuration as described in Section 2.3. We vary the load and effective RAM capacity and show the measured average throughput in Figure 11. It is obvious that tenant 1 can tolerate more effective capacity modulation than tenant 2; therefore, we expect that the cloud sets lower effective capacity to tenant 1. We use the same methodology

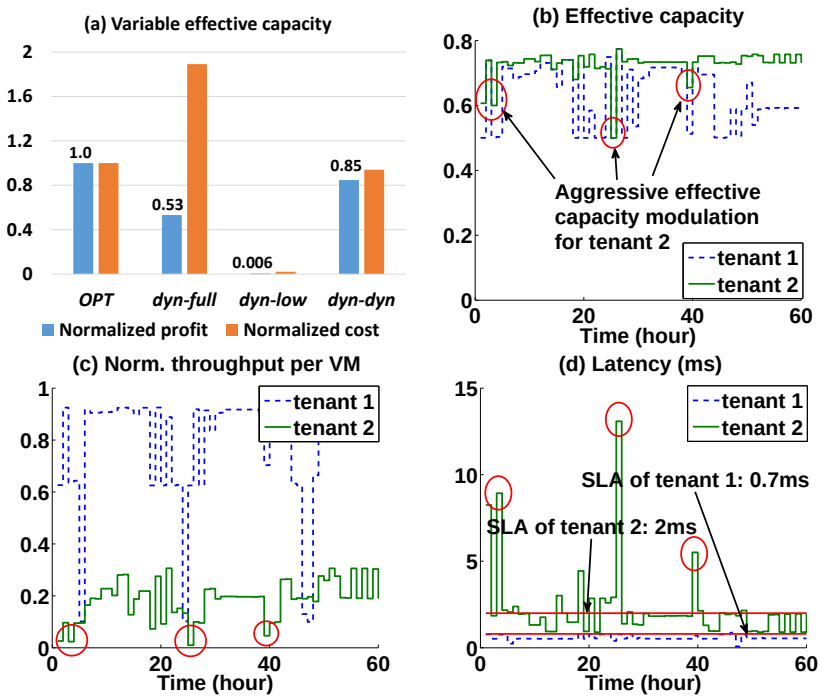


Fig. 12: (a) Normalized profits and costs of the cloud with the data caching tenants. (b) Normalized VM price and effective capacity under the full game *dyn-dyn*. (c) Achieved average throughput per VM. (d) Achieved latency. SLA requirements are 0.7ms and 2ms for tenant 1 and tenant 2, respectively. The circled areas reflect tenant 2's severe performance degradation due to aggressive effective cap. modulation.

as described in the Web serving case study to fit $F_i^h(.)$ and $F_i^\phi(.)$. We scale the demand traces in Figure 6(b) to generate raw request rates $\Lambda_{i,t}$ for the two tenants separately.

4.3.2. Provider's Profitability. From Figure 12(a), again we find that *dyn-dyn* provides much higher profits than other strategies. In particular, *dyn-low* (half effective capacity) has almost 0 profit due to the fact that both tenants experience severe performance degradation at $\eta_{i,t} = 0.5$ as shown in Figure 11. *dyn-full* is also not able to provide good profitability due to its inability to reduce costs.

4.3.3. Tenants' Performance. As we expected, the provider modulates tenant 1's effective capacity more aggressively (Figure 12(b)) since it is less sensitive to performance degradation. The consequence is, as shown in Figure 12(c)(d), that tenant 1 is still able to achieve high throughput with almost no SLA violations, whereas tenant 2 suffers from a low throughput (by dropping large portions of raw arrivals) but still has many SLA violations. This is because tenant 2 only has around 10% effective capacity slackness; if the effective capacity drops below 90%, its performance degrades severely as in Figure 11(b). Due to the short-term prediction and myopic control that the provider uses to estimate tenants' reactions, it becomes very difficult for the provider to know that it should not "steal" more than 10% effective capacity from tenant 2. On the other hand, since the provider always carries out real effective capacity modulation after the tenants reveal their VM demands, it is also difficult for the tenants to predict

the effective capacity. In fact, even a little bit of misprediction might be fatal to such performance-sensitive tenants.

Therefore, it may not be wise for tenant 2 that is so sensitive to effective capacity modulation to choose VMs of the SLA class in evaluation with $\eta^{LB} = 0.5$ for hosting its services. Rather, it might be a good idea for such tenants to pay higher costs for an SLA class with large baseline capacity or dedicated H/W resource with full capacity guarantees.

Key insights: (i) The provider offers differentiated effective capacity (in addition to the same baseline capacity) to tenants based on observations of their reactions to effective capacity modulation. (ii) Tenants can have different tolerance to effective capacity modulation. At an extreme, for some tenants expensive resource types with more guaranteed capacity might be the appropriate choice.

5. RELATED WORK

Pricing design. There is a large body of related work on the pricing design of various cloud resources/services via techniques such as Nash games [Tsai and Qi 2012; Feng et al. 2012], auction framework [Li et al. 2013; Shi et al. 2014; Zhang et al. 2015; Baranwal and Vidyarthi 2015], MDP [Xu and Li 2012], non-optimization-based [Kaushik et al. 2013; Sharma et al. 2012], etc. [Niu et al. 2012] proposes a novel model of cloud network bandwidth pricing in a Nash game with the goal of maximizing the cloud's and the tenants' social welfare. [Valerio et al. 2013] employs a two-stage Stackelberg game to determine the resource (VM) provisioning for Amazon EC2 on-demand and spot instances as well as the dynamic spot prices. [Xu and Li 2013] maximizes the cloud's profit by dynamically adjusting the VM prices using a stochastic dynamic programming approach. [Kantere et al. 2011] designs a price-demand model and a dynamic pricing scheme for a SaaS cloud which provides data caching services. However, a common assumption in prior work is that the cloud can predict well the tenants' demands and their reactions to price changes for theoretical tractability, whereas we look at a more general spectrum of tenant applications/workloads including real-world workloads with poor predictability. Close to our work, in [Wang et al. 2015] a game-based cloud pricing framework is proposed which also explores cloud's profitability with poor predictability in workloads under dynamic pricing. However, [Wang et al. 2015] does not explicitly consider tenant's application performance in either framework formulation or restricted trace-driven simulation.

Effective capacity modulation. Commercial public cloud providers already provision computing resources with dynamic effective capacities, with the most prominent example being the burstable instances, which have guaranteed base CPU capacity plus variable capacity, offered by Amazon EC2 and Google Compute Engine. Recent research work [Nasiriani et al. 2017] reveals that the network bandwidth of such instances is also burstable, although not explicitly exposed in their SLAs. Another example, albeit not quite intuitive, is the preemptible instances [PreemptibleVMs 2016] from Google Compute Engine and the spot instances from Amazon EC2, both of which might be revoked by the cloud due to the tenant's out-of-bid or the cloud's resource shortage, resulting variable (intermittent) capacity as opposed to the regular full capacity. Moreover, even the instance types which are claimed to have full capacity exhibit dynamic effective capacity to some extent (cf. Section 2.2). However, the efficacy of such offerings in improving the cloud's profitability is limited since the variable capacity/availability of such instances either depends on the tenant's bid (e.g., EC2 spot) or usage credits (e.g., burstable instances) or is simply a side-effect of the cloud's other resource management decisions instead of explicit control actions.

On the other hand, to the best of our knowledge, the implications of explicit effective capacity modulation on a provider's operation/profit (particularly taking into account

the tenants' response to such modulation), complementing dynamic pricing, has not been explored in existing work. Several specific forms of effective capacity modulation have been explored. As one example, a popular line of work focuses on cloud cost optimization via VM consolidation [Beloglazov and Buyya 2010; Corradi et al. 2014; Zhang et al. 2010; Meng et al. 2010] (mostly in private clouds), one way in which effective capacity modulation might be realized. However, such work does not consider tenant's strategic behavior (based on its utility model) in response to the degraded application performance, thereby losing the opportunity of further improving cloud's profit by exploiting the "slackness" in tenant's tolerance to effective capacity reduction. In public clouds, capacity modulation has not been explored much. [Niu et al. 2012] proposes to provision network bandwidth as fixed portion plus stochastically provisioned portion based on the the tenants' willingness to pay. [Mogul and Kompella 2015] discusses the potential benefit of modulating provisioned network bandwidth without tenants' awareness, which seems "cheating" on tenants and complicates/worsens their resource procurement strategies. However, in our work a tenant chooses from multiple SLA classes and a corresponding baseline capacity defined in the chosen SLA class is guaranteed.

Tenant operation with dynamic prices or capacities. There exists a plethora of related work that investigates how tenants react to dynamic pricing strategically in public clouds [Wang et al. 2015; Zafer et al. 2012; Niu et al. 2012]. In particular, many research studies investigate the cost-effective solutions for tenant's procurement of spot instances (with dynamic spot prices), for different workloads or applications, e.g., delay-tolerance batch jobs [Song et al. 2012; Subramanya et al. 2015; Menache et al. 2014], video streaming [He et al. 2014], data caching [Xu et al. 2016]. Towards the trade-off of price vs. effective capacity, some recent research works begin to exploit the cost-benefit of using instances with explicit temporal capacity variation but cheap prices, e.g. EC2 burstable instances, for executing jobs with intermittent resource needs [Wen et al. 2015; Nasiriani et al. 2017]. Research works have also explored tenants' cost-optimal instance selection and operations for heterogeneous clusters [Farley et al. 2012; Conley et al. 2015; Xu et al. 2013; Jayathilaka et al. 2015] using instances with implicit capacity modulation. These are all complementary to our focus on the cloud provider's operation.

6. CONCLUSIONS

We explored the efficacy of effective capacity modulation as a control knob for a cloud provider's profit maximization when tenants react to such modulation. Towards this, we proposed a leader/follower game-based cloud control framework with dynamic pricing and effective capacity modulation to maximize the cloud's profits. We investigated the feasibility of effective capacity modulation. We leveraged myopic control and short-term predictions to deal with the cloud's poor predictability of tenant's demand and price/performance sensitivities. Our evaluation showed 10-30% profit improvement compared with static pricing and/or static effective capacity. We also carried out realistic case studies on a prototype cloud platform with insights about performance in real-world settings.

APPENDIX

A.1. Proof of Claim 1

PROOF.

Denote tenant i's cost function as $J_t(d_{i,t}) := u_i(d_{i,t}r\eta_{i,t}) - v_t d_{i,t}$. Then we can obtain the following

$$\begin{aligned} J_t^{'}(d_{i,t}) &= r\eta_{i,t}u_i^{'}(d_{i,t}r\eta_{i,t}) - v_t \\ J_t^{''}(d_{i,t}) &= (r\eta_{i,t})^2 u_i^{''}(d_{i,t}r\eta_{i,t}) < 0 \end{aligned}$$

wherein the latter is due to the assumption that $u_i(.)$ is strictly concave. Therefore, $J_t(d_{i,t})$ is also strictly concave w.r.t. $d_{i,t}$, and there exists a unique optimal solution for the unconstrained optimization problem $\max_{d_{i,t}} J_t(d_{i,t})$. Now taking the first-order derivative of $J_t(.)$ and setting to 0, we can obtain the optimal solution for the unconstrained problem as follows:

$$\tilde{d}_{i,t}^* = \frac{1}{r\eta_{i,t}}(u_i^{'})^{-1}(\frac{v_t}{r\eta_{i,t}})$$

wherein $(u_i^{'})^{-1}(.)$ is the inverse function of $u_i^{'}(.)$. Now let us look at the original constrained optimization problem with boundary constraint $d_{i,t}r\eta_{i,t} \leq D_{i,t}$. Since $J_t(d_{i,t})$ is strictly concave, when $\frac{D_{i,t}}{r\eta_{i,t}} \geq \tilde{d}_{i,t}^*$, the optimal solution of the constrained problem is $d_{i,t}^* = \tilde{d}_{i,t}^*$; otherwise, $d_{i,t}^* = \frac{D_{i,t}}{r\eta_{i,t}}$.

If we further define $x_{i,t} := \frac{v_t}{r\eta_{i,t}}$, the optimal solution of the constrained optimization problem can be re-written as follows:

$$d_{i,t}^* = \frac{1}{r\eta_{i,t}} \min\{D_{i,t}, (u_i^{'})^{-1}(x_{i,t})\}$$

which implies that the optimal amount of effective capacity $d_{i,t}^* r\eta_{i,t}$ only depends the ratio $x_{i,t}$ if there is large enough raw demand $D_{i,t}$.

□

A.2. Proof of Claim 2

PROOF. Assume that the cloud only has a single tenant and it has perfect knowledge of tenant's raw demand and optimal solution as in Claim 1[5]. When $\frac{D_t}{r\eta_t} \geq \tilde{d}_t^*$, the tenant's optimal demand is $d_t^* = \tilde{d}_t^* = \frac{1}{r\eta_t}(u^{'})^{-1}(x_t)$ where $x_t = \frac{v_t}{r\eta_t}$, and $m_t = \lceil d_t^* r\eta_t \rceil = \lceil (u^{'})^{-1}(x_t) \rceil$. Note that with perfect knowledge, the cloud does not have to consider consecutive time-slots in the optimization window, since the tenant's optimal demand only depends on the VM price and effective capacity ration in the current time-slot. Therefore, the cloud's optimization problem becomes

$$\max_{v_t, \eta_t} \; x_t(u^{'})^{-1}(x_t) - p_t\lceil (u^{'})^{-1}(x_t) \rceil$$

Subject to

$$\lceil (u^{'})^{-1}(x_t) \rceil \leq M$$

Clearly although there exists an optimal $x_t = \frac{v_t}{\eta_t}$ that maximizes the cloud's profit, the optimal solution (v_t, η_t) of the above optimization problem is not unique.

[5]We remove the subscript i since there is only one tenant.

On the other hand, when $\frac{D_t}{r\eta_t} < \tilde{d}_t^*$, we have $d_t^* = \frac{D_t}{r\eta_t}$, and $m_t = \lceil d_t^* r\eta_t \rceil = \lceil D_t \rceil$. The cloud's optimization problem becomes

$$\max_{v_t, \eta_t} \; x_t D_t - p_t \lceil D_t \rceil$$

Subject to

$$\lceil D_t \rceil \leq M$$

The optimal solution (v_t, η_t) is also not unique.

In sum, we conclude that there exists an optimal ratio of $x_t = \frac{v_t}{r\eta_t}$ that maximizes the cloud's profit, whereas the optimal solution (v_t, η_t) is not unique.

□

REFERENCES

AmazonSpot. 2016. (2016). http://aws.amazon.com/ec2/spot-instances/.

G. Baranwal and D. P. Vidyarthi. 2015. A fair multi-attribute combinatorial double auction model for resource allocation in cloud computing. *Journal of Systems and Software* (2015).

L. A. Barroso and U. Hölzle. 2013. The datacenter as a computer: An introduction to the design of warehouse-scale machines. (2013).

A. Beloglazov and R. Buyya. 2010. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proc. of MGC'10*.

S. Brandt, G. Nutt, T. Berk, and M. Humphrey. 1998. Soft real-time application execution with dynamic quality of service assurance. In *Proc. of IWQoS*.

BurstableVM. 2016. https://aws.amazon.com/ec2/instance-types/t2/. (2016).

Cloudharmony. 2016. http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html. (2016).

Cloudlook. 2016. http://http://www.cloudlook.com. (2016).

CoincidentPeak. 2013. (2013). http://www.fcgov.com/utilities/business/rates/electric/coincident-peak.

M. Conley, A. Vahdat, and G. Porter. 2015. Achieving cost-efficient, data-intensive computing in the cloud. In *Proc. of SoCC'15*.

A. Corradi, M. Fanelli, and L. Foschini. 2014. VM consolidation: A real case based on OpenStack cloud. *Future Gener. Comput. Syst.* 32 (2014), 118–127.

Datacenter2025. 2015. http://www.emersonnetworkpower.com/en-US/Latest-Thinking/Data-Center-2025/Pages/default.aspx. (2015).

B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift. 2012. More for your money: Exploiting performance heterogeneity in public clouds. In *Proc. of ACM SOCC*.

Y. Feng, B. Li, and B. Li. 2012. Bargaining towards maximized resource utilization in video streaming datacenters. In *Proc. of INFOCOM'12*.

A. Gandhi, P. Dube, A. Karve, A. Kochut, and H. Ellanti. 2015. The unobservability problem in clouds. In *Proc. of ICCAC'15*.

D. Ghoshal, R. S. Canon, and L. Ramakrishnan. 2011. I/O performance of virtualized cloud environments. In *Proc. of DataCloud-SC'11*.

J. He, Y. Wen, J. Huang, and D. Wu. 2014. On the Cost-QoE tradeoff for cloud-based video streaming under Amazon EC2's pricing models. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 4 (2014), 669–680.

H. Herodotou and S. Babu. 2011. Profiling, what-if analysis, and cost-based optimization of MapReduce programs. *Proc. of PVLDB'11* (2011).

EC2 I/O. 2012. http://blog.scalyr.com/2012/10/a-systematic-look-at-ec2-io/. (2012).

H. Jayathilaka, C. Krintz, and R. Wolski. 2015. Response time service level agreements for cloud-hosted Web applications. In *Proc. of SoCC'15*.

V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou, and A. Ailamaki. 2011. Optimal service pricing for a cloud cache. *IEEE Trans. Knowledge and Data Engineering* 23 (Sept 2011), 1345–1358.

R. T. Kaushik, P. Sarkar, and A. Gharaibeh. 2013. Greening the compute cloud's pricing plans. In *Proc. of HotPower'13*.

C. Klein, M. Maggio, K. Arzen, and F. Hernandez-Rodriguez. 2014. Brownout: Building more robust cloud applications. In *Proc. of ICSE'14*.

H. Li, C. Wu, Z. Li, and F. Lau. 2013. Profit-maximizing virtual machine trading in a federation of selfish clouds. In *Proc. of INFOCOM'13*.

MediaWiki. 2016. (2016). https://www.mediawiki.org/wiki/MediaWiki.

Memcached. 2016. (2016). http://memcached.org/.

I. Menache, O. Shamir, and N. Jain. 2014. On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud. In *Proc. of ICAC'14*.

X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. 2010. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proc. of ICAC'10*.

J. C. Mogul and R. R. Kompella. 2015. Inferring the network latency requirements of cloud tenants. In *Proc. of HotOS'15*.

N. Nasiriani, C. Wang, G. Kesidis, and B. Urgaonkar. 2017. Characterizing the Network Bandwidth Dynamism of Amazon EC2 Burstable Instances. In *Proc. of IC2E'17*.

D. Niu, C. Feng, and B. Li. 2012. Pricing cloud bandwidth reservations under demand uncertainty. In *Proc. of ACM SIGMETRICS'12*.

NYTimes. 2012. http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html?pagewanted=all\&_r=0. (2012).

Ontario-electric. 2014. (2014). http://www.ieso.ca/Pages/Power-Data/.

P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. 2007. Adaptive control of virtualized resources in utility computing environments. In *Proc. of EuroSys'07*.

PreemptibleVMs. 2016. (2016). https://cloud.google.com/preemptible-vms/.

B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya. 2012. Pricing cloud compute commodities: A novel financial economic model. In *Proc. of IEEE/ACM Ccgrid'12*.

W. Shi, L. Zhang, C. Wu, Z. Li, and F. C.M. Lau. 2014. An online auction framework for dynamic resource provisioning in cloud computing. In *Proc. of ACM SIGMETRICS'14*.

Y. Song, M. Zafer, and K. Lee. 2012. Optimal bidding in spot instance market. In *Proc. of INFOCOM'12*.

C. Stewart and K. Shen. 2005. Performance modeling and system management for multi-component online services. In *Proc. of NSDI'05*.

StolenTime. 2013. (2013). http://blog.scoutapp.com/articles/2013/07/25/understanding-cpu-steal-time-when-should-you-be-worried.

S. Subramanya, T. Guo, P. Sharma, D. Irwin, and P. Shenoy. 2015. SpotOn: A batch computing service for the spot market. In *Proc. of SoCC'15*.

W. Tsai and G. Qi. 2012. DICB: Dynamic intelliegnt customizable benign pricing strategy for cloud computing. In *Proc. IEEE ICCC'12*.

B. Urgaonkar, P. Shenoy, and T. Roscoe. 2002. Resource overbooking and application profiling in shared hosting platforms. *Proc. of OSDI'02* (2002).

V. D. Valerio, V. Cardellini, and F. L. Presti. 2013. Optimal pricing and service provisioning strategies in cloud systems: A stackelberg game approach. In *Proc. of IEEE CLOUD'13*.

A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proc. of EuroSys'15*.

VM-DVFS. 2016. https://www.cloudyn.com/blog/new-aws-c4-instance-and-the-complete-ec2-families-guide/. (2016).

C. Wang, N. Nasiriani, G. Kesidis, B. Urgaonkar, Q. Wang, Y. Chen, A. Gupta, and R. Birke. 2015. Recouping energy costs from cloud tenants: tenant demand response aware pricing design. In *Proc. of ACM e-Energy'15*.

G. Wang and T.S.E. Ng. 2010. The impact of virtualization on network performance of Amazon EC2 data center. In *Proc. of Infocom'10*.

J. Wen, L. Lu, G. Casale, and E. Smirni. 2015. Less can be more: Micro-managing VMs in Amazon EC2. In *Proc. of IEEE ClOUD'15*.

H. Xu and B. Li. 2012. Maximizing revenue with dynamic cloud pricing: the infinite horizon case. In *Proc. of ICC'12*.

H. Xu and B. Li. 2013. Dynamic cloud pricing for revenue maximization. *Cloud Computing, IEEE Transactions on* 1, 2 (2013), 158–171.

Y. Xu, Z. Musgrave, B. Nobel, and M. Bailey. 2013. Bobtail: Avoiding long tails in the cloud. In *Proc. of NSDI'13*.

Z. Xu, C. Stewart, N. Deng, and X. Wang. 2016. Blending on-demand and spot instances to lower costs for in-memory storage. In *Proc. of Infocom'16*.

YCSB. 2016. (2016). https://github.com/brianfrankcooper/YCSB.

Murtaza Zafer, Yang Song, and Kang-Won Lee. 2012. Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs. In *Proc. of IEEE CLOUD'12*.

Q. Zhang, L. Cheng, and R. Boutaba. 2010. Cloud computing: state-of-the-art and research challenges. *Proc. of JISA'10* (2010).

X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C.M. Lau. 2015. Online auctions in IaaS clouds: Welfare and profit maximization with server costs. In *Proc. of ACM SIGMETRICS'15*.

Z. Zhang, L. Cherkasova, and B. T. Loo. 2014. Optimizing cost and performance trade-offs for MapReduce job processing in the cloud. In *Proc. of NOMS'14*.

Received December 2016; revised July 2017; accepted August 2017