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

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

***Abstract*—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 an explicit control knob for a cloud provider's profit maximization complementing the more well-studied approach of dynamic pricing. 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. 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 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-world data center traces and 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.**

I. Introduction

It is well-known that many current data centers are underutilized (around 6% to 12% in 2012 [21]). Profitability concerns are changing this, e.g., best practices from Google suggest much improved utilization levels [25]. In a recent survey [8] 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., 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 [5]). 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. Numerous research papers have explored cloud's pricing design for profitability (more details in Section V).

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 resources if any) which may be different from its advertized capacity [1]. Several studies report temporal dynamism in effective capacity is already prevalent among public cloud VMs - Sections II and V offer 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 resource management arsenal.

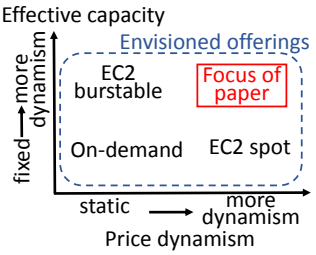

Fig. 1: Our vision of public cloud VM offerings with different degrees of dynamism both in pricing and effective capacity. We also show example VM types from EC2 and where they fit within our scheme.

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. Similar variety is exhibited by VM offerings from other major public clouds including Google, Microsoft, RackSpace, etc. 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 *advertized to be* provisioned with a fixed capacity (although even these exhibit some capacity variation as our measurements in Section II show). The recently introduced "burstable" instances, on the other hand, only guarantee a certain baseline CPU capacity beyond which capacity becomes variable [2].

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

[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.

of gainfully using such VMs and navigating the associated price-performance trade-offs. 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 tenant'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[2]. 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 model this setting is via a leader/follower game with the cloud provider being the leader and tenants the followers. The interaction these parties 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 (Section IV).
- 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 II).
- 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 III).
- 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 IV). 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]The variable capacity portion could be defined in stochastic terms, e.g., with mean and variance, which we leave to our future work.

II. Motivation 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 pervasive. We then present an SLA framework that our envisioned public cloud might use to systematize its VM offerings.

A. 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 [26]. 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.

B. 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) (a) when they are executed on different types of VMs procured from Amazon EC2 versus (b) when they are executed on similarly sized VMs created on a machine in our lab. We perform experiments on a large number of Amazon

TABLE I: Benchmarks and performance comparisons. The cases with higher performance variations in our lab have been shaded.

benchmark		description	dominant resource	coefficient of variation (%) (EC2, Lab)		
				t2.micro	m3.medium	m4.large
luindex		index a set of documents using lucene and measure execution time	CPU	(7.4, 3.92)	(7.55, 4.87)	(8.95, 3.38)
iperf		transfer large chunk of data and measure network bandwidth	Network b/w	(0.55, 0.34)	(21.15, 0.34)	(2.42, 0.34)
Stream	copy	perform simple memory operations (copy, add) on vectors and measure sustainable memory bandwidth	Memory bandwidth	(1.73, 0.78)	(4.85, 0.5)	(0.37, 0.92)
	add			(1.57, 0.63)	(9.88, 0.38)	(0.17, 0.81)
fio	read	generate random disk I/O workloads and measure iops	Disk I/O bandwidth	(25.74, 1.87)	(3.18, 1.87)	(15.49, 1.87)
	write			(25.96, 1.88)	(2.96, 1.88)	(15.58, 1.88)

EC2 VM types and show a subset of our results (see our technical report for all our results [28]) for 3 diverse VM types: `t2.micro` (burstable with 1 vCPU and 1GB RAM), `m3.medium` (1 vCPU and 3.75GB RAM) and `m4.large` (2 vCPUs and 8GB RAM), all in the availability zone *us-east-1c*. We repeat our benchmarking on a machine in our lab (2.4 GHz with 12 cores and 16GB RAM) with the same resource allocations (using `cgroups`) as advertized for the EC2 VMs. As shown in Table I, we consistently find higher *cv* (coefficient of variation) for instances in the cloud than our lab-based host, with only a few exceptions. Although we can not know what precise resource management decisions on the machines hosting the EC2 VMs cause this higher level of dynamism, it is reasonable to interpret these findings as evidence for the following: when running in the cloud, the benchmarks experience a larger dynamism in offered resource capacities leading to higher dynamism in their performance. Even if advertized as having fixed capacity (as opposed to variable capacity for burstable VMs), some on-demand instance types might exhibit lower effective capacity than advertized. For example, we find `m3.medium` has 5% to 10% more *cv* for memory bandwidth (*Stream*) compared to other instance types, which suggests that such VMs may have been more aggressively consolidated than others. Similarly, for CPU-bounded benchmark *luindex*, we find the performance variation across all VM types in the cloud is more than twice compared to that on our lab-based host.

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 provider or across multiple cloud platforms, resulting in the "unobservability" problem described in [11]. These temporal/spatial performance variations have been observed across different resource types, e.g., CPU capacity [30], [9], block I/O [13], network latency and bandwidth [19], [29]. CloudLook [4] reports that even under similar advertized CPU 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 [3] CCU (CloudHarmony Compute Unit) scores across cloud providers are evidence of effective capacity variation across clouds.

C. 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 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 2. 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.

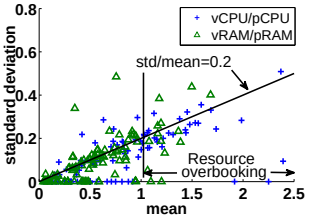

Fig. 2: Statistics of resource allocation from a private hosted data center. "vCPU/pCPU" and "vRAM/pRAM" denote the ratios of number of vCPU vs. physical cores and (virtual) RAM capacity of all VMs vs. physical RAM capacity of a single physical machine.

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 (a popular key-value store) 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 IV-C. We use Yahoo! Cloud Serving Benchmark (YCSB) 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 and show the average throughput and the corresponding latency in Figure 3.

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 throughput and higher latency as effective capacity decreases for both tenants. Since tenant 1's key popularity distribution is much more skewed than tenant 2's distribution, lesser data is moved to swap for

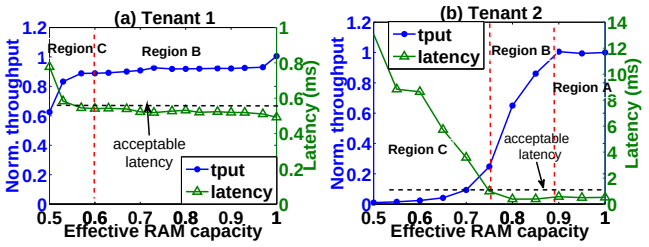

Fig. 3: Average throughput (norm. against arrival rate) and corresponding latency for a data caching benchmark with one Memcached server and one YCSB workload client. (a) A tenant with exponential key popularity distribution: 95% requests go to 5% of working set (b) A tenant with Zipfian key popularity distribution.

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:

1) *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.
2) *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.
3) *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 3 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.

D. Cloud's 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 II-A 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 II-B. 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**

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

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 [3]. 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.

Dynamic effective capacity should not be thought of as "cheating." One form it may take could indeed be as suggested in [19] 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.

III. System Design and Formulation

A. High-level Overview

Figure 4 shows a high-level overview of our cloud ecosystem. We choose to work with a time-slotted system wherein

[3] We assume a tenant workload to only have one bottleneck or "dominant" resource, allowing us to define effective capacity as a ratio (or scaling parameter for a resource vector as done in [12]). A more general treatment would consider multiple resources and we consider that part of our future work.

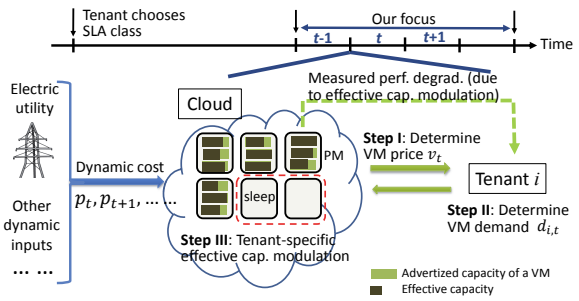

Fig. 4: Illustration of our cloud eco-system and the proposed game framework. Cloud's dynamic costs come from a combination of various dynamic inputs, e.g., dynamic energy price.

the cloud provider and its tenants make strategic control 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 responses 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.

B. 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).

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' reactions), and (iii) effective capacity $\eta_{i,t-1}$ (which affects tenant i's performance and thereby its reaction). 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

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

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 reactions 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 guarantees 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.

2) Step II: Tenants' Strategic Behavior: Upon receiving VM price v_t and, each tenant has to determine the number of VMs it needs from the cloud based on the inferred effective capacity $\hat{\eta}_{i,t}$ (using performance measurements in the previous time-slot) and its own predicted raw demand. Denote as $D_{i,t}$ the raw physical resource demand of tenant i. We assume that

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

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. Recall that we assume $\eta_{i,t} \geq \eta^{LB}$ 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 [22], [24], [17], throughput vs. effective capacity for MapReduce-like workloads [36], response time vs. effective capacity for streaming servers [1] 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 IV.

3) Step III: Provider's Corrective 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 IV, and leave more comprehensive control strategies to future work.

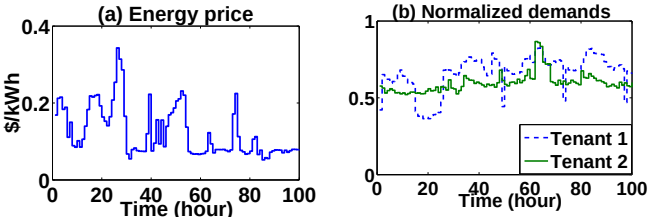

Fig. 5: (a) Hourly energy prices from an electric utility in [14]. (b) Two tenants' normalized effective capacity demands (CPU usage in MHz) from a large commercial data center.

To get a preliminary understanding of the interactions between the cloud and tenants, we derive "best responses" for both entities with simplifying assumptions [28]. Due to its idealized assumption of perfect prediction by both parties, our theoretic analysis suggests that dynamic pricing and effective capacity are interchangeable. However, in practice, capacity modulation acts as a **corrective** knob when prediction errors may cause the provider to publish sub-optimal prices. We illustrate this gap between theory and practice in Section IV.

IV. Evaluation

We evaluate the proposed game framework in two complementary ways. In Section IV-A, 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 IV-B and IV-C.

A. Trace-driven simulation

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 [14] (Figure 5(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 demands, using scattered interpolation on the predictors described in Section III. 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 5(b)) as their raw effective capacity demand. We choose the *log*-form utility function as described in Section III-B. We assume that the tenant

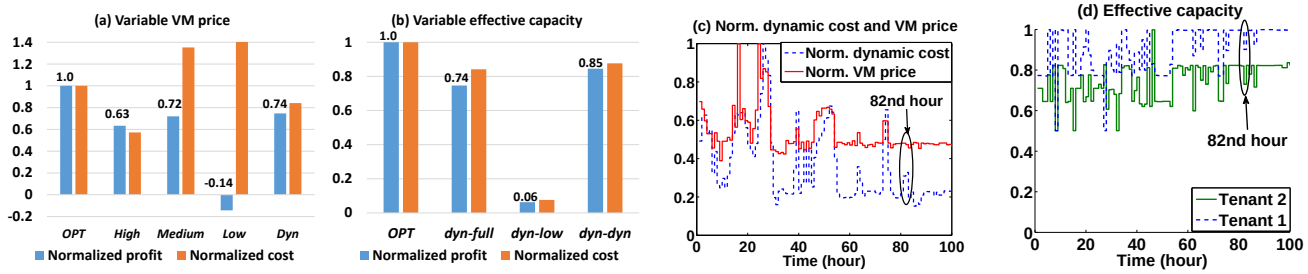

Fig. 6: Provider's normalized profit and cost under (a) dynamic vs. static pricing, (b) dynamic vs. static effective capacity with dynamic pricing. Profits and costs are normalized w.r.t. the profit and cost for *OPT*, respectively. *dyn*-{*cap*} represents strategies with dynamic pricing (*dyn*) where *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.

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. We present the details of these parameters in our technical report [28].

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

2) Impact of Dynamic Pricing: In Figure 6(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 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%).

3) Impact of Effective Capacity Modulation: From Figure 6(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. Due to space limit, we omit such results in the paper.

4) A Closer Look at the Game Output: Figures 6(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).

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.

B. Case Study I: Web Serving

We present a case study based on the Wikipedia Web server benchmark in [31], 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.

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

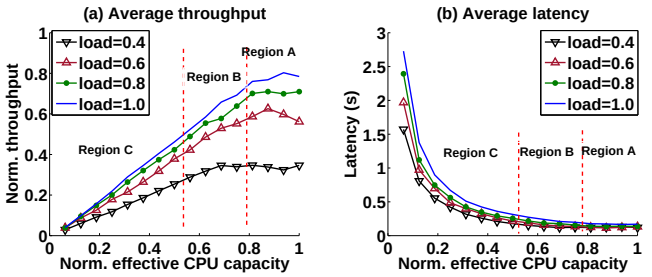

Fig. 7: Average throughput (a) and average latency (b) with a Wikipedia Web server under different load. "load" is norm. w.r.t. the peak arrival rate on a single VM with 8 vCPUs and 12GB RAM.

prediction of effective capacity $\hat{\eta}_{i,t}$, 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 III). $\bar{\phi}_i$ 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 7, 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 7. We scale the CPU usage (in MHz) demand of tenant 1 in Figure 5(b) to generate raw request rates $\Lambda_{i,t}$.

2) Provider's Profitability: Figure 8(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.

3) Tenant's Performance: We show timeseries of VM prices and effective capacity under *dyn-dyn* in Figure 8(b). The corresponding request admission rate and latency for the tenant are shown in Figures 8(c)(d), respectively. As discussed in Section II-C, we qualitatively define three operation regions for the tenant's slackness in effective capacity in Figure 7, and also find examples of such regions in Figure 8. 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 8(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 8(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.

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.

C. Case Study II: Data Caching

We host two tenants based on a real-world key-value store benchmark with the same configuration in Section II-C, with memory capacity as the critical resource. 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.

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 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 II-C. We vary the load and effective RAM capacity and show the measured average throughput in Figure 9. The corresponding average latency is not shown due to space limit. It is obvious that tenant 1 can tolerate more effective capacity modulation than tenant 2; therefore, we expect that the cloud

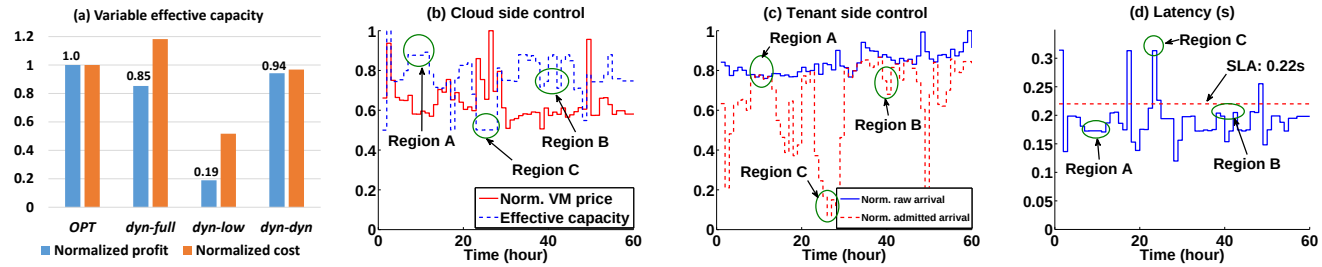

Fig. 8: (a) Normalized profits and costs of the provider with the Web serving tenant. (b) Normalized VM price (w.r.t. peak VM price) and effective capacity under *dyn-dyn*. (c) Web serving tenant's raw arrival and admitted rates. (d) Achieved latency. SLA requirement is 0.22s.

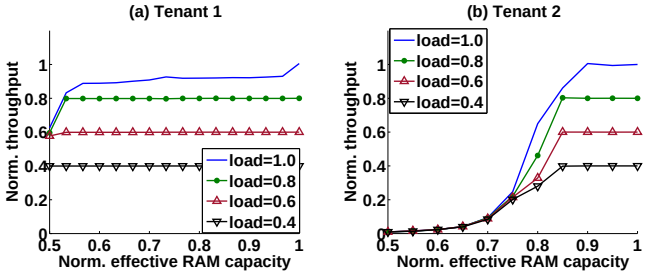

Fig. 9: Average throughput from a data caching benchmark with one memcached server and one YCSB workload client under different loads and key popularity distributions. Here "load" is normalized w.r.t. the peak load on a single VM with 4 vCPUs and 7GB RAM. (a) Exponential distribution with 95% request going to 5% keys. (b) Zipfian distribution with Zipf constant=0.99.

sets lower effective capacity to tenant 1. We use the same methodology as described in the Web serving case study to fit $F_i^h(.)$ and $F_i^\phi(.)$. We scale the demand traces in Figure 5(b) to generate raw request rates $\Lambda_{i,t}$ for the two tenants separately.

2) Provider's Profitability: From Figure 10(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 9. *dyn-full* is also not able to provide good profitability due to its inability to reduce costs.

3) Tenants' Performance: As we expected, the provider modulates tenant 1's effective capacity more aggressively (Figure 10(b)) since it is less sensitive to performance degradation. The consequence is, as shown in Figure 10(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 9(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. As an extreme, for some tenants expensive resource types with more guaranteed capacity might be the appropriate choice.

V. Related Work

Pricing design. There is a large body of related work in cloud pricing design, including techniques such as Nash games [20], [10], auction framework [18], MDP [34], non-optimization-based [16], [23], etc. A common assumption in prior work is that the cloud can predict well the tenant's demand and control for theoretical tractability. However, we look at a more general spectrum of tenant applications/workloads including real-world workloads and strategic control with poor predictability. Close to our work, [27] proposes a game-based cloud pricing framework which also explores cloud's profitability with poor predictability in workloads under dynamic pricing. However, it does not explicitly consider tenant's application performance in framework formulation.

Effective capacity modulation. Plenty of research works have shown the evidence of dynamic effective capacity on commercial IaaS cloud platforms (cf. Section II-B). However, to the best of our knowledge, the implications of 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 [7], [32], [33] (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, the only research work we find related to effective capacity is [19] wherein Google researchers discuss the potential benefit of modulating provisioned network bandwidth

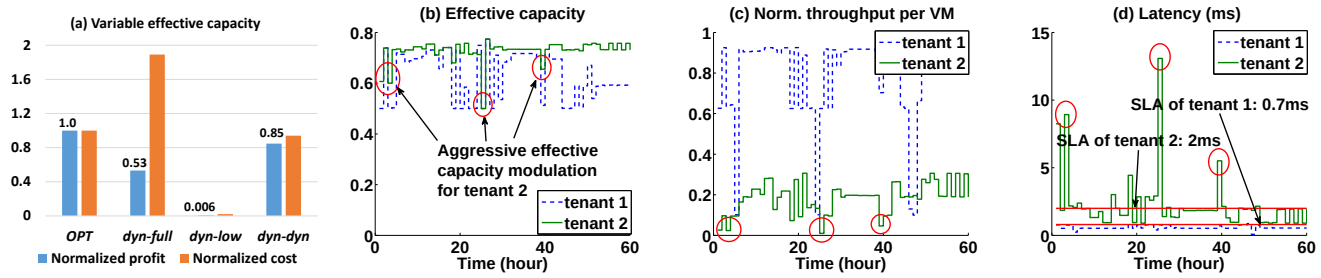

Fig. 10: (a) Normalized profits and costs of the cloud with the data caching tenants. (b) Normalized VM price and effective capacity under the full game *dyn-dyn*. (c) Achieved average throughput per VM. (d) Achieved latency. SLA requirements are 0.7ms and 2ms for tenant 1 and tenant 2, respectively. The circled areas reflect tenant 2's severe performance degradation due to aggressive effective cap. modulation.

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 [27], [20], [35]. Many recent papers have also explored how to improve tenant's application performance or cost-efficacy using VM offerings with explicit effective capacity modulation [30] or implicit [9], [6], [15] dynamic capacity modulation. These are all complementary to our focus on the cloud provider's operation.

VI. 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. 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.

References

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

[2] Burstable vm. https://aws.amazon.com/ec2/instance-types/t2/, 2016.

[3] Cloudharmony. http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html, 2016.

[4] Cloudlook. http://http://www.cloudlook.com, 2016.

[5] Fort Collins Coincident Peak, 2016. http://www.fcgov.com/utilities/business/rates/electric/coincident-peak.

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

[7] A. Corradi, M. Fanelli, and L. Foschini. Vm consolidation: A real case based on openstack cloud. *Future Gener. Comput. Syst.*, 32, 2014.

[8] Data center 2025. http://www.emersonnetworkpower.com/en-US/Latest-Thinking/Data-Center-2025/Pages/default.aspx, 2015.

[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 *Proc. of ACM SOCC*, 2012.

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

[11] A. Gandhi, P. Dube, A. Karve, A. Kochut, and H. Ellanti. The unobservability problem in clouds. In *Proc. of IEEE ICCAC*, 2015.

[12] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. of NSDI'11*, 2011.

[13] D. Ghoshal, R. S. Canon, and L. Ramakrishnan. I/o performance of virtualized cloud environments. In *Proc. of DataCloud-SC*, 2011.

[14] Ontario electric utility, 2014. http://www.ieso.ca/Pages/Power-Data/.

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

[16] R. T. Kaushik, P. Sarkar, and A. Gharaibeh. Greening the compute cloud's pricing plans. In *HotPower*, 2013.

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

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

[19] J. C. Mogul and R. R. Kompella. Inferring the network latency requirements of cloud tenants. In *Proc. of HotOS XV*, 2015.

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

[21] New york times. http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html?pagewanted=all&_r=0, 2012.

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

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

[24] C. Stewart and K. Shen. Performance modeling and system management for multi-component online services. In *Proc. of NSDI*, 2005.

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

[26] Vm dvfs. https://www.cloudyn.com/blog/new-aws-c4-instance-and-the-complete-ec2-families-guide/, 2016.

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

[28] C. Wang, B. Urgaonkar, A. Gupta, L. Y. Chen, R. Birke, and G. Kesidis. Effective capacity modulation as an explicit control knob for public cloud profitability. Technical report, http://www.cse.psu.edu/research/publications/tech-reports/2016/CSE-16-001.pdf, 2016.

[29] G. Wang and T. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proc. of IEEE Infocom*, 2010.

[30] J. Wen, L. Lu, G. Casale, and E. Smirni. Less can be more: Micro-managing vms in amazon ec2. In *Proc. of IEEE ClOUD*, 2015.

[31] Mediawiki, 2016. https://www.mediawiki.org/wiki/MediaWiki.

[32] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy. Profiling and modeling resource usage of virtualized applications. In *Proc. of Middleware'08*, 2008.

[33] S. Xi, C. Li, C. Lu, C. Gill, M. Xu, L. Phan, I. Lee, and O. Sokolsky. Rt-openstack: Cpu resource management for real-time cloud computing. In *IEEE Cloud'15*, 2015.

[34] H. Xu and B. Li. Maximizing revenue with dynamic cloud pricing: the infinite horizon case. In *Proc. of IEEE ICC*, 2012.

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

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