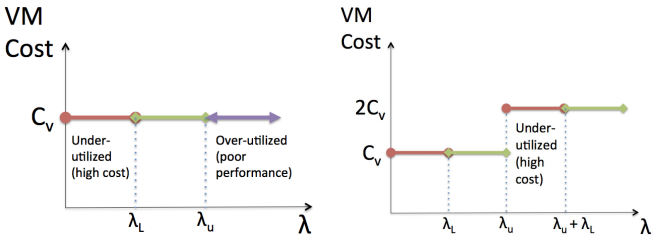
Optimal Pricing for Serverless Computing

Kunal Mahajan*, Daniel Figueiredo†, Vishal Misra* and Dan Rubenstein*
*Columbia University, †Universidade Federal do Rio de Janeiro
*{mkunal,misra,danr}@cs.columbia.edu, †daniel@cos.ufrj.br

***Abstract*—Serverless computing is an attractive cloud services paradigm, simultaneously promising reduced cost and greater flexibility for customers and increased revenues and higher resource utilization for cloud providers. In this paper, we present an analysis of the potential cost benefits of serverless computing for end customers and cloud providers. Using realistic cost models, queueing theoretic performance models, and a game theoretic formulation, we explore the tradeoffs between serverless computing (SC) and traditional cloud computing (virtual machine, VM). In the proposed framework, customers distribute their workload between SC and VM to minimize their cost while maintaining a particular performance constraint, while the cloud provider sets SC and VM prices to maximize its profit. We explore the impact of relative prices, customer workload, service capacity, and provider costs. Our main result is the identification and characterization of three optimal operational regimes for both customers and the provider that leverage either SC or VM only, or both, in a hybrid configuration. The various insights provided in this paper can help both cloud providers and customers better understand the tradeoffs and implications of a hybrid system that combines serverless and VM rental with corresponding pricing models.**

I. Introduction

Cloud computing is a disruptive technology that over the past two decades has dramatically changed how enterprises compute and how computation is monetized. Cloud computing platforms offer a plethora of computing paradigms and pricing models. The classic paradigm is the rental of virtual machines (VMs) for prolonged periods of time: the price depends on the VM resources (e.g., CPU, memory) and contract duration (e.g., month, year).



(a) Single VM (b) Multiple VMs

Fig. 1: Cost analysis for VMs

This VM rental paradigm can be inefficient for both the customer and provider since the reservation of resources is for a long duration. This inefficiency is explained in Figure 1: The left plot shows the cost as a function of the arrival load (e.g., jobs per second). Note that as the VM rental cost is fixed (C_v), the cost does not vary with the load when a single VM is rented (Figure 1(a)). There are two noteworthy issues: 1) when the load is too low, the VM is underutilized. Thus, the customer ends up paying a higher price for its effective computation; 2) when the load is too high, the VM is overloaded, and performance degrades to inadequate levels (e.g., very long response times). The second problem can be addressed by renting a second VM, as illustrated in Figure 1(b). However, the first problem persists. Note that if the first VM is overloaded by a small amount, renting two VMs can still leave the VMs underutilized.

A more recent paradigm is the rental of VMs under a contract that charges only for the times during which the VM is on. While this paradigm permits both the customer and provider to utilize resources with greater efficiency, there is still an overhead incurred, namely, time required to hibernate the VM, time to restore a hibernated VM, and user responsibility for turning the VMs on and off. In this paper, we focus on the more classic VM rental model, where the user pays a fixed rate independent of the state of the VM.

Even more recently, the Function-as-a-Service (FaaS) paradigm, also known as **Serverless Computing (SC)**, has emerged, in which users install *functions* in the cloud that are executed by their remote applications. The cost of a function call depends on its running time and resources consumed such as memory. From the provider perspective, since SC operates atop a *container* infrastructure as opposed to a more generalized VM infrastructure, the underlying hardware can be utilized more efficiently as it accommodates more containers than general purpose VMs [1]. This allows providers to increase profit by driving user loads onto SC, and to raise the respective price for unit time processing in SC.

However, SC can only handle a certain class of workload (different cloud providers offer different APIs and languages to instantiate the functions) and VMs will continue to be offered by cloud provider. The price of renting the VMs is then decided by market competition between the cloud providers, and so the providers cannot arbitrarily increase the price of SC (or offer only SC) as the rational user always has the option to switch to (the cheaper) VM rental.

In this paper, we explore the question of how a provider can *relatively* price SC, in order to maximize profit when facing rational users that minimize their costs (under some performance constraint). We combine simple cost models

This work was funded in part by the NSF through awards CNS-1717867 and CNS-1618911, and in part by CNPq and FAPERJ, Brazil. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

grounded in existing cost profiles offered by real cloud providers and traditional queueing theory and game theory to identify optimal strategies for both the user and provider. In particular, we determine how a user should purchase cloud services to minimize its cost and serve its workload under a given performance constraint. Moreover, we determine how the provider should set prices for cloud services in order to maximize profits. More specifically, we make the following contributions:

- We leverage simple queueing models for VMs and SC to characterize their performance under some given load (see Section II).
- We analyze the model and determine the optimal allocation between VMs and SC as a function of model parameters, such as prices for VMs and SC set by the provider, service capacity of VMs and containers, and user performance constraint (see Section II-A).
- Three possible operational regimes emerge under minimum user cost: use SC only, combine SC and VM, and use VM only.
- We propose a simple cost model for the provider to offer VM and SC, where its revenue is identical to the user's cost. Under this consideration, we formulate a game between the provider and the user, while for former sets a price and the latter decides on VM and SC (see Sections III and III-A).
- We identify the analytical expression for a Nash equilibrium of the game, using a strategy where the user minimizes its cost (under a given price) and the provider maximizes its profit (see Section III-B).

II. Cloud Services Model: User perspective

Consider cloud computing services where a client (user) may utilize both serverless computing (SC) and virtual machines (VMs). We apply the traditional cost model of such services where VMs are rented at a fixed price per unit time (denoted by α_v), irrespective of the loads placed on them and the running times of jobs they process. In contrast, SC also has a fixed unit time price (denoted by α_s). However, the cost of SC is proportional to sum of running times (duration) of jobs processed by the service.

Each rented VM is modeled as a single server system with a fixed capacity. However, due to variability in job sizes, we assume the time required to service a job can follow any distribution with average $1/\mu_v$. Note that μ_v can be interpreted as the average job service rate and is a parameter of the VM.

SC is modeled as an infinite server queue with fixed capacity, as the cloud provider dynamically allocates resources to service each job within a given performance profile (e.g., provided enough memory and CPU to process at a given rate). More specifically, the provider dynamically allocates *containers* (a lightweight VM at OS-level) to service a job arriving to its SC platform [1]. Although a container is given a fixed processing capacity (and memory), the variability in job sizes leads to an arbitrary distribution for the job service time. We assume that the average job service time in SC is $1/\mu_s$. Again, μ_s can be interpreted as the average job service rate and is a parameter of the SC platform.

Consider a client that uses cloud services to process a continuous arrival of jobs, with an average rate denoted by λ (e.g., $\lambda = 10$ jobs per minute), and would like to minimize its cost subject to some performance constraint, such as mean response time of jobs. In particular, we assume the client can leverage multiple VMs and SC simultaneously, splitting its load across the contracted services. We assume that the arriving jobs are mutually independent, i.e. the execution of a job does not depend on the execution of any other job[1].

We assume client load arrives according to a Poisson process with rate λ and is split randomly[2] among the rented resources, such that a thinned Poisson process arrives at VMs and SC. This permits us to model each VM as an M/G/1 queue, while SC as an M/G/∞ queue.

Recalling that the client has an explicit performance constraint when using cloud services, we assume this constraint is an upper bound on the average response time of jobs sent to the cloud for processing. For the SC model, this constraint translates directly to the average service time parameter, $1/\mu_s$. For the VM model, we employ a utilization constraint: every rented VM must have a utilization that is smaller than ρ_t, a parameter set by the client. In this form, using the Pollaczek-Khinchine formula [2], the response time constraint applied to an M/G/1 queue maps to a utilization constraint given the mean and second moment of the workload distribution. Moreover, several other performance constraints, such as the 95^{th} response time, can also be mapped to some maximum utilization in this model.

Let $s(\lambda) \leq \lambda$ denote the rate of load the client assigns to SC. Recall that in SC the client pays as long as the job is being processed. Thus, the average cost of processing this load is given by the average number of busy servers in the M/G/∞ queue, which is given by $s(\lambda)/\mu_s$, times the unit time cost, α_s.

Let $v(\lambda)$ denote the number of VMs rented by the client when its load's rate is λ. Due to the performance constraints, each VM can accept a maximum load of $\rho_t\mu_v$, and thus, the maximum load the rented VMs can jointly accept is $v(\lambda)\rho_t\mu_v$ at a cost of $v(\lambda)\alpha_v$, since the user pays per rented VM.

For a given load λ, the client chooses $s(\lambda)$ (how much load to send to SC) and $v(\lambda)$ (how many VMs to rent) to minimize cost:

$$c(\lambda) = \alpha_s s(\lambda)/\mu_s + \alpha_v v(\lambda) \tag{1}$$

subject to $\lambda - s(\lambda) \leq v(\lambda)\rho_t\mu_v$, which is the performance constraint for the rented VMs. Note that we assume that $1/\mu_s$ meets the average response time constraint for the client, since otherwise SC would never be considered.

[1]This assumption can be relaxed considerably to assuming only that two simultaneously running jobs are not dependent on one another.

[2]While this is a simple stateless load balancing policy, using more sophisticated load balancing mechanisms does not qualitatively change the nature of our results, it simply changes the threshold ρ_t used in our analysis

A. Analysis

We now investigate the tradeoffs and economic benefits in renting VMs and using SC. Intuitively, SC can be leveraged to reduce customer costs, especially for low loads or for loads just over the performance constraint.

Lets start by noting that the cost of serverless is linear in its load whereas the cost of VM is independent of its load. For small enough loads, serverless will always be cheaper. However, since this cost increases linearly with the load, it will eventually surpass the fixed cost of renting a VM. Let $L_s = \alpha_v \mu_s / \alpha_s$ denote the load at this crossover. When user load $\lambda < L_s$, the user should only use SC.

When $\lambda > L_s$, renting a VM is cheaper and the VM can handle loads $\lambda > L_s$ until the performance constraint is no longer met. Let $L_v = \rho_t \mu_v$ denote the load that a single VM can accept under the capacity constraint. Thus, for $\lambda \in [L_s, L_v]$, renting a VM is cheaper and can satisfy the performance constraint without invoking SC.

When $\lambda > L_v$, the user must either rent another VM or divert part of the load to serverless. Again, since serverless has a linear cost structure, it is cheaper to shed small overloads to SC for a small enough excess load. When the excessive load is sufficiently large, it will again be cheaper to rent a (second) VM, and SC will not be used.

This process repeats itself as λ increases, and the optimal allocation can be computed as follows. For a load λ, we need a total of $\lfloor \lambda / L_v \rfloor$ VMs since each VM can handle a load of L_v within the performance constraint determined by the user. Note the number of VMs is an integer. Since each VM can handle a load of L_v, the total load handled by the VMs is simply $\lfloor \lambda / L_v \rfloor L_v$. Given this value, the excess load is given by $\lambda - \lfloor \lambda / L_v \rfloor L_v$. Finally, if this excess load is smaller than L_s, we send it to serverless. Otherwise, we rent an additional VM to process this excess load, as it will be cheaper.

From the above calculations, in general we have the following optimal result for renting VMs:

$$v(\lambda) = \begin{cases} 0, & \text{if } L_v \leq L_s \\ \left\lfloor \frac{\lambda}{L_v} \right\rfloor, & \text{if } \lambda - \lfloor \lambda / L_v \rfloor L_v \leq L_s \\ 1 + \left\lfloor \frac{\lambda}{L_v} \right\rfloor, & \text{otherwise} \end{cases} \quad (2)$$

Moreover, we can determine the load diverted to serverless in the optimal allocation, which is given by

$$s(\lambda) = \begin{cases} \lambda, & \text{if } L_v \leq L_s \\ \lambda - \left\lfloor \frac{\lambda}{L_v} \right\rfloor L_v, & \text{if } \lambda - \lfloor \lambda / L_v \rfloor L_v \leq L_s \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Thus, equations (2) and (3) give the solution that minimizes the cost posed in equation (1).

Last, note that if SC is much cheaper than VMs, then renting VM may never be more cost effective. This occurs when $L_v \leq L_s$, indicating that when the load at which the cost of serverless equals that of a single VM is larger than what a VM can handle, then the VM is just not cost effective. In this scenario, we have that $s(\lambda) = \lambda$ and $v(\lambda) = 0$, as indicated in the above equations.

B. Numerical evaluations

Figure 2 plots the optimal cost as function of the load λ for different ratios of α_s to α_v. The curves increase linearly with respect to λ when SC is used and flattens when an additional VM is rented. Note that the cost reduction when using SC strongly depend on the relative prices. If SC is relatively expensive ($\alpha_s = 10\alpha_v$), then the cost savings are marginal. On the contrary, if SC price is moderate, cost reduction is significant for a larger range of loads ($\alpha_s = 3\alpha_v$). Last, if prices are comparable ($\alpha_s = 1.3\alpha_v$), then SC always yields a lower cost.

Another important consideration is the relationship between VM price and capacity, as cloud providers offer a wide range of VM capacities at different prices. Figure 3 shows the optimal user cost for different price/capacity ratios. Note that as VM capacity increases (and thus its price), the cost reduction yielded by SC also increases. This occurs because L_s increases with the price of the VM. However, since the VM has more capacity, it also handles more load, inducing a larger L_v.

Finally, we investigate the price regions for optimal allocation of VMs and serverless. Note that given the model parameters (including λ) and a given pair prices α_s and α_v, either VM rental is cheaper, or serverless is cheaper (or the same). This induces regions in $\alpha_s \times \alpha_v$ plane where VM or serverless is more cost effective. Figure 4 illustrates these regions for different values for λ. Note that each curve (straight line) corresponds to the case where the costs are equal. Below and above the line correspond to the cases where SC and VM rental is cheaper, respectively. Last, as the load λ increases, the region where serverless is cheaper decreases, since a smaller time unit cost is required for serverless to be more cost effective.

III. Cloud Services Model: Provider perspective

The previous section analyzed how a user can minimize its cost by leveraging different cloud services to process its load. The reduction in cost attained by the user translates directly to a reduced revenue (and thus, profit) to the cloud provider. Therefore, it becomes essential for the cloud provider to optimally set prices for cloud services to ensure maximum profits.

The cloud provider can clearly influence how a rational client leverages different clouds services, as the provider sets the prices. For instance, if the price for SC is comparable to VMs, a rational client will divert all its load to SC, and never rent a VM (as shown in the previous section). On the other hand, if SC price is much larger than VM price, for almost all loads the client will prefer to use only VMs. Last, there is price range for SC and VM that the client will prefer to leverage both services. Hence, the relative pricing between SC and VM determines the services purchased by a rational client, and consequently the revenue of the provider.

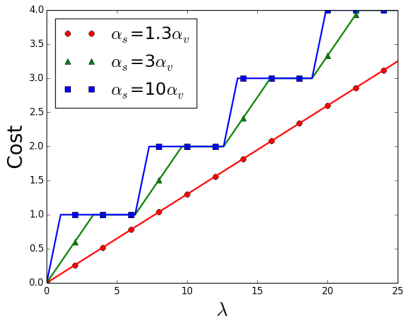


Fig. 2: Optimal customer cost with $\mu_s = 10$, $\mu_v = 7$, $\alpha_v = 1$.

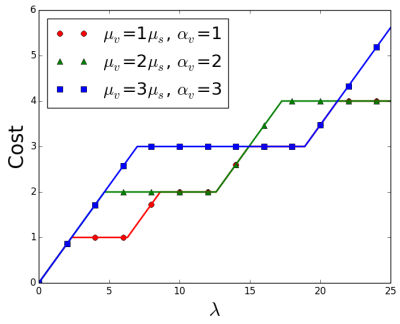


Fig. 3: Optimal customer cost with $\mu_s = 7$, $\alpha_s = 3$.

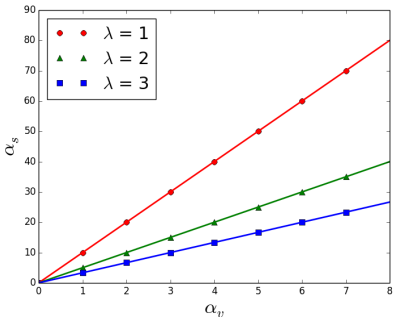


Fig. 4: Unit time price regions ($\mu_s = 10$, $\mu_v = 5$).

Clearly, a provider incurs costs when offering cloud services, which can depend on the kind of service. These differences in costs are due to differences in hardware and software, management practices, and necessary performance. Thus, the cost of offering a VM to client is different from the cost of offering SC. In particular, let β_v and β_s be the unit time cost for the cloud provider to offer a single VM and a single container for SC, respectively.

Note that a VM demands more resources from a physical machine than a container used in SC, which are more lightweight, as shown in the literature [1]. Therefore, a physical server can offer more CPU cycles *devoted to the user jobs* when leveraging containers. Thus, a provider can pack in more containers on a server as compared to VMs, and this reduces the operating costs for the provider on a per unit basis (hardware costs, power usage etc.). Consequently, the cost of offering a single VM is larger than that of offering a single container for SC, $\beta_s < \beta_v$. Moreover we assume the cost ratio to be inversely proportional to performance ratio, such that $\beta_s/\beta_v = \mu_v/\mu_s$. This is quite useful because it relates the service capacity of SC with its cost in comparison to VM.

Last, since containers can be implemented more efficiently in a given hardware, we assume that $\mu_s > \mu_v$. Thus, the cost to the provider as a function of λ is given by:

$$c_p(\lambda) = \beta_s s(\lambda)/\mu_s + \beta_v v(\lambda) \tag{4}$$

where $v(\lambda)$ is the number of VMs rented by the client and $s(\lambda)$ is the client load sent to SC. As SC service is modeled by a M/G/∞ queue we use its expectation to denote the number of containers needed in SC. Thus, $s(\lambda)/\mu_s$ is the expected number of containers required to offer SC for a load of $s(\lambda)$.

The provider profit is simply the difference between its costs and revenue for providing and selling a given service. Note that this revenue is exactly the client cost for using the cloud services, and is given by:

$$p(\lambda) = c(\lambda) - c_p(\lambda) = v(\lambda)(\alpha_v - \beta_v) + s(\lambda)/\mu_s(\alpha_s - \beta_s) \tag{5}$$

where $c(\lambda)$ is the cost for the client which is identical to the revenue of the provider, as given by equation 1.

Given this formulation, how should a provider set its prices to maximize profits under a rational client that minimizes its costs? In what follows we tackle this problem.

A. The Provider-Client Game

Consider the problem of a provider setting prices to maximize profits under a client that leverages cloud services to minimize its cost. We model this problem as a two-player game, using a game theoretic formulation [3]. In particular, we consider a sequential game with perfect information where the provider moves first and determines prices for its cloud services. The client moves second and determines the services that will be purchased. In our formulation, the provider must set α_s and α_v (prices) and the client must decide on $v(\lambda)$ and $s(\lambda)$ (number of VMs and load to SC). Note that being a game of perfect information, both players have access to all model parameters and equations, including λ.

We assume the client will purchase cloud services to process its load and meet its performance constraint. Under such consideration, a provider could simply set prices arbitrarily high to increase its profits. However, given a scenario with multiple cloud providers (as is the case in the current market), the client could simply purchase services from another cloud provider, in order to minimize its cost. In order to capture this phenomenon, we assume that the provider sets a fixed price for VM rental, due to strong market competition. In particular, we let $\alpha_v = 1$. Thus, our provider is left with choosing the price α_s for SC.

The solution to this game is a strategy for the provider and a strategy for the client. A Nash equilibrium in this game is a pair of strategies (one for the provider, another for the client) such that neither player can benefit from unilaterally changing strategies. A Nash equilibrium can be found using backward induction in the game tree [3], as follows. For every possible price set by the provider, the client minimizes its cost by allocating its load to VMs and SC. Among all this scenarios, the provider chooses the price that maximizes its profit. In the analysis that follows, we determine this Nash equilibrium.

B. Analysis

Recall that in Section II we determined the minimum cost for the client under a given price. However, recall that the optimal allocation strongly depends on model parameters and has three different regimes: SC only, SC+VM, and VM only. In what follows, we consider each of these regimes.

Case 1 (SC only): $\rho_t\mu_v \leq \alpha_v\mu_s/\alpha_s$. In this regime, we have that $p(\lambda) = (\lambda/\mu_s)(\alpha_s - \beta_v/\mu_s)$. This profit is maximized by setting α_s as large as possible within the constraint, since the profit grows linearly with α_s. Thus, the maximum value for α_s that satisfy the constraint, denoted by $\alpha_s^{(1)}$, is given by:

$$\alpha_s^{(1)} = \mu_s/\rho_t \tag{6}$$

Therefore, $\alpha_s^{(1)} = \mu_s/\rho_t$ is the price that maximizes the profit in this regime. Note that the price is proportional to the capacity of the SC container and inversely proportional to the target utilization set by the user, ρ_t. In particular, a user that has a more conservative performance constraint (e.g., lower target utilization) will observe a higher price, since resources cannot be used as efficiently. The maximum profit in this regime is given by

$$p^{(1)}(\lambda) = \lambda\left(\frac{1}{\rho_t} - \frac{\beta_v}{\mu_s^2}\right) \tag{7}$$

Case 2 (SC+VM): $\lambda - \lfloor\lambda/\rho_t\mu_v\rfloor\rho_t\mu_v \leq \alpha_v\mu_s/\alpha_s$. In this regime we have that

$$p(\lambda) = \left\lfloor\frac{\lambda}{\rho_t\mu_v}\right\rfloor(1-\beta_v) + \left(\frac{\lambda}{\mu_s} - \left\lfloor\frac{\lambda}{\rho_t\mu_v}\right\rfloor\frac{\rho_t\mu_v}{\mu_s}\right)\left(\alpha_s - \frac{\beta_v}{\mu_s}\right) \tag{8}$$

Again, this profit increases linearly with α_s and thus is maximized by setting α_s as large as possible within the constraint that determines this case. Thus, the largest possible value for α_s in this case, denoted by $\alpha_s^{(2)}$, is given by:

$$\alpha_s^{(2)} = \frac{\mu_s}{\lambda - \lfloor\lambda/\rho_t\rfloor\rho_t} \tag{9}$$

which maximizes the profit in this case. Note that the price is proportional to the capacity of the SC container and inversely proportional to load offered to SC. In particular, a user that sends more load to SC will observe a lower price, in order for the provider to maintain its revenue. The maximum profit in this regime is then given by

$$p^{(2)}(\lambda) = \left\lfloor\frac{\lambda}{\rho_t\mu_v}\right\rfloor(1-\beta_v) + \left(\lambda - \left\lfloor\frac{\lambda}{\rho_t\mu_v}\right\rfloor\rho_t\mu_v\right)\left(\frac{1}{\lambda - \lfloor\lambda/\rho_t\rfloor\rho_t} - \frac{\beta_v}{\mu_s^2}\right) \tag{10}$$

Case 3 (VM only): otherwise. In this case the profit is independent of α_s as only VMs are being used. In particular, if $\alpha_s \geq \alpha_s^{(2)}$ then only VMs will be used by the client. The profit is this regime is given by

$$p^{(3)}(\lambda) = \left(1 + \left\lfloor\frac{\lambda}{\rho_t}\right\rfloor\right)(1-\beta_v) \tag{11}$$

The profit in this region is independent of α_s due to no load being sent to SC. Therefore, the value of α_s can be set arbitrarily large subject to the constraint $\alpha_s \geq \alpha_s^{(2)}$. While the optimal value of α_s is not relevant in this case, it is important when we consider distribution of workloads. Hence, in this region, we assume the optimal value of α_s as follows:

$$\alpha_s^{(3)} = \alpha_s^{(2)} \tag{12}$$

Since the provider wants the maximum profit, it can consider the maximum over the three possible regimes, which is given by:

$$p^*(\lambda) = \max\{p^{(1)}(\lambda), p^{(2)}(\lambda), p^{(3)}(\lambda)\} \tag{13}$$

Moreover, the price that achieves this optimum profit is given by:

$$\alpha_s^* = \alpha_s^{(o)}, \text{ where } o = \arg\max_{i=1,2,3}\{p^{(i)}(\lambda)\} \tag{14}$$

Theorem 1. *Consider the Provider-Client game where the strategy for the provider is a choice for α_s and its payoff given by eq. 5 and the strategy for the client is the choice of $v(\lambda)$ (number of rented VMs) and $s(\lambda)$ (load to SC) and payoff given by the negation of eq. 1. The choices given by eq. 14 (for the provider) and eqs. 2 and 3 (for the client) is a Nash equilibrium of this game.*

Proof. The proof is by construction of the equations that determine the Nash equilibrium via backward induction on the game tree. Note that for a given price α_s chosen by the provider, the strategy that minimizes the client cost (maximizes its payoff) is given by eqs. 2 and 3. Thus, the client cannot reduce its cost (improve its payoff) by deviating from this strategy. Similarly, given the client strategy of minimizing its cost as determined by eqs. 2 and 3 and the strategy of the provider of determining the price to maximize its profit as in eq. 13, the provider cannot improve its payoff by deviating from the price given in eq. 14. □

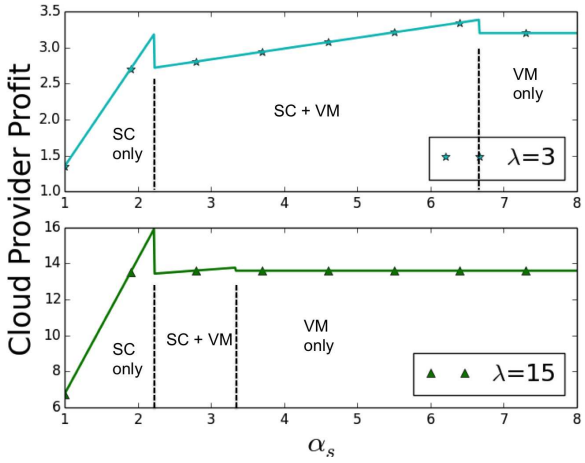


Fig. 5: Provider profit as a function of SC price, α_s.

1) Numerical evaluations: Figure 5 shows the provider's profit as a function of the SC price, α_s, for two values for load, corresponding to each plot. For both cases, as the price increases we observe three distinct regions, which correspond to the user configuration: SC only, SC+VM, VM only. The first region where $\alpha_s < 2.22$ shows a fast linear increase in the provider's profit since all the workload is served by SC. In this region, the profit rises simply because the user is paying a higher price to use SC for the same workload. At $\alpha_s = 2.22$, there is a sudden drop in the profit as the user reduces its cost by utilizing the hybrid model SC+VM, with the majority served on the VM and the remaining load on SC. The profit again starts to increase as the price increases because the same load served in SC is now being charged a higher price. For $\lambda = 3$, the profit increases linearly for $\alpha_s \in [2.22, 6.6]$, while for $\lambda = 15$ it increases in the range $\alpha_s \in [2.22, 3.4]$. Note that the profit slope in this range is not as steep as in the SC only regime since the workload on SC is much smaller (only the excess load). As the price continues to increase we observe a second sudden drop in the profit, at $\alpha_s = 6.6$ and $\alpha_s = 3.4$, respectively. This drop occurs because the user can reduce its cost by using a second VM to serve the excess load, not sending any more load to SC. After this point, increasing the price α_s has no bearing on the profit as the user has adopted a VM only configuration. Thus, note that as α_s increases from zero onward, the user reacts accordingly to minimize its cost and this has a direct influence on the provider's profit.

The shape of these two curves is not particular of this parameter configuration. Given our model formulation and the optimal prices and profit established in equations 14 and 13, there will always be two peaks corresponding to the transition from all SC to a hybrid SC+VM configuration and from the hybrid SC+VM to an all VM configuration. Moreover, since these are peaks, the optimal profit is bound to occur at one of these two peaks. Interestingly, the transition point that provides a higher profit does depend on the parameter configuration, as illustrated in Figure 5. However, as λ increases the optimal profit will be attained at the first peak, with the highest possible price within the all SC regime. As λ decreases, the optimal price is attained in the second peak, with the highest possible price within the SC+VM regime. Again, this is illustrated in Figure 5.

IV. Related work

A large body of work has explored different pricing schemes for different cloud services paradigms, considering both the perspective of the customer (how to minimize cost) and the cloud provider (how to maximize revenue or profit) [4]–[7]. Cloud providers also offer services under a dynamic pricing scheme (i.e., spot instances), and some recent works have focused on price bidding and price prediction and dynamic VM allocation to reduce costs [8]–[10]. A common line of work explores the combination of multiple cloud service paradigms, such as dynamic pricing, fixed pricing, and VMs that can be switched on and off, under different workloads, such as video streaming, and scientific computation, in order to reduce customer costs [7], [11]–[13]. Another area of research has focused on minimizing customer costs in hybrid (private/public) cloud providers, especially where they explore provisioning of private clouds and the offloading of excess computations to a public cloud [14], [15]. Although these prior works are similar in scope to the current paper, to the best of our knowledge, none of them have explored a hybrid approach that combines VM rentals and serverless computing (SC) to reduce customer cost and maximize provider profit.

V. Conclusion

This paper has addressed a hybrid system where a customer can split its workload between VM rental and SC in order to minimize costs while satisfying a given performance constraint. A key finding is the existence of three optimal operational regimes, where depending on system parameters, the Nash equilibrium of the game (i.e., the optimal strategy for the customer and provider) is for the customer to use only SC or only VM or a combination of both. Moreover, we analytically characterize these three regimes as a function of system parameters, such as determining the optimal price for SC. Finally, we believe the proposed framework to study SC pricing and the various insights provided in this paper can help both cloud providers and customers better understand the tradeoffs and implications of a hybrid system that combines SC and VM rental with their corresponding pricing structure.

References

[1] P. Sharma, L. Chaufournier, P. Shenoy, and Y. C. Tay, "Containers and virtual machines at scale: A comparative study," in *ACM International Middleware Conference*, 2016.

[2] L. Kleinrock, *Theory, Volume 1, Queueing Systems*, 1975.

[3] M. J. Osborne and A. Rubinstein, *A course in game theory*, 1994.

[4] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi, "A survey on spot pricing in cloud computing," *Journal of Network and Systems Management*, 2017.

[5] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, 2015.

[6] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[7] I. Menache, O. Shamir, and N. Jain, "On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud," in *USENIX International Conference on Autonomic Computing*, 2014.

[8] P. Sharma, D. Irwin, and P. Shenoy, "How not to bid the cloud," in *USENIX Workshop on Hot Topics in Cloud Computing*, 2016.

[9] S. Subramanya, T. Guo, P. Sharma, D. Irwin, and P. Shenoy, "Spoton: A batch computing service for the spot market," in *ACM Symposium on Cloud Computing*, 2015.

[10] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Transactions on Cloud Computing*, 2013.

[11] D. J. Dubois and G. Casale, "Optispot: minimizing application deployment cost using spot cloud resources," *Cluster Computing*, 2016.

[12] C. Wang, B. Urgaonkar, A. Gupta, G. Kesidis, and Q. Liang, "Exploiting spot and burstable instances for improving the cost-efficacy of in-memory caches on the public cloud," in *EuroSys*, 2017.

[13] Z. Xu, C. Stewart, N. Deng, and X. Wang, "Blending on-demand and spot instances to lower costs for in-memory storage," in *IEEE INFOCOM*, 2016.

[14] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *IEEE CLOUD*, 2009.

[15] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Generation Computer Systems*, 2013.