Reserved, On Demand or Serverless: Model-based simulations for cloud budget planning

Edwin F. Boza*, Cristina L. Abad*, Mónica Villavicencio*, Stephany Quimba*, Juan Antonio Plaza†
*Escuela Superior Politécnica del Litoral, ESPOL. Campus Gustavo Galindo Km 30.5 Vía Perimetral, Guayaquil, Ecuador.
†Dátil. Av Miguel Alcívar, Edificio Torres del Norte, Torre A, oficina 308, Guayaquil, Ecuador.
eboza@fiec.espol.edu.ec, cabadr@espol.edu.ec, mvillavi@espol.edu.ec, struquim@espol.edu.ec, juanantonio@datil.co

***Abstract*—Cloud computing providers offer a variety of pricing models, complicating the client decision, as no single model is the cheapest in all scenarios. In addition, small to medium-sized organizations frequently lack personnel that can navigate the intricacies of each pricing model, and as a result, end up opting for a sub-optimal strategy, leading to overpaying for computing resources or not being able to meet performance goals. In this paper, we: (1) present the results of a study that shows that, in Ecuador, a considerable percentage of companies choose conservative pricing strategies, (2) present a case study that shows that the conservative pricing strategy is suboptimal under certain workloads, and (3) propose a set of models, a tool and a process that can be used by tenants to properly plan and budget their cloud computing costs. Our tool is based on $M(t)/M/*$ queuing theory models and is easy to configure and use. Note that, even though we are motivated by our study of adoption of cloud computing technologies in Ecuador, our tool and process are widely applicable and not restricted to the Ecuadorian context.**

***Keywords*—Cloud, reserved, on-demand, serverless, budget, simulation, queuing theory**

I. Introduction

The term *cloud computing* encompasses a myriad of services available to tenants, including—but not limited to—storage services like Dropbox, productivity suites like Office 365, computing offerings like those of Amazon Web Services (AWS), Microsoft Azure, Google cloud services and Rackspace, among others. The term can also include *private clouds*, or in-house datacenters where virtual machines (VMs) are provisioned on-demand to different organizational units.

In this paper, we concentrate on *utility computing* offerings, which include what providers often refer to as Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS). We bundle these types of offerings together because there is no clear line where to make the division between the two categories [4]. In addition, we also include in our analysis the recent development of *serverless computing*, which lets the tenant define functions to be executed as a response to client requests [14]. The reason to consider serverless computing in the same category as server-based offerings is that—in essence—both are computing resources. Furthermore, many microservices for web applications can easily be implemented using any of these types of cloud services (i.e., on VMs, containers, or as functions executed on demand). In short, even though "the cloud" encompasses many types of computation, storage, communication, and software services [4], in this paper we focus on the **computing** part of *cloud computing*.

978-1-5386-3894-1/17/$31.00 © 2017 IEEE

Organizations are increasingly adopting cloud computing due to benefits like reducing infrastructure costs, avoiding over-provisioning, scaling applications to handle variable demand, analyzing Big Data at a low cost, etc. [2], [5], [23]. These benefits are particularly attractive to small and medium-sized enterprises (SMEs), which can offer innovative products or services without large investments [11], [22], [25].

The main enabler of these benefits is the pay-as-you-go or utility model, in which you pay only for consumed resources. Examples include bid-based VMs, on-demand VMs or containers that are launched and released depending on tenant-defined elasticity rules to adapt to client demand, and functions that are executed only when a client invokes them.

Sadly, emerging markets—potential big winners of the cloud—have been slow in adopting it; and when they do, they tend to opt for the less flexible pricing models [12].

We are concerned that flexible cloud pricing options are not being sufficiently embraced, leading to overpaying when the client demand is variable (as is the case in most industries). We present a case study that shows that a conservative pricing strategy can be suboptimal under dynamic workloads, leading to significant increase in costs ($39.5\times$ in our example).

We then propose a tool and process that can be used by companies to find out which cloud pricing model best suits their needs. Our tool is based on $M(t)/M/*$[1] queuing theory models and is easy to configure and use. Our approach provides a significant improvement over the current cloud budgeting process for small and medium enterprises [10], in which cost is estimated using simple math or online budgeting apps. In contrast, our tool provides better budget estimation via the support of dynamic workloads and generation of performance metrics; hence, the choice of a pricing model can be guided by exploring the cost versus performance tradeoff.

This paper is structured as follows. In Section II, we describe pricing models offered by cloud providers. In Section III, we present an overview of the adoption of cloud computing in Ecuador, focusing on the preferred payment models. In Section IV, we show the importance of choosing the right cloud pricing model. In Section V, we propose a set of models,

[1]Expressed in Kendall's notation, which describes the arrival process, the service time distribution and the number of servers, as detailed in Section V.

TABLE I: Pricing models supported by three cloud providers.

Pricing model	Amazon AWS	Microsoft Azure	Google Cloud Platform
Reserved VMs	×	×	×
On-demand VMs	×	×	×
Bid-based VMs	×		×
Serverless computing	×	×	×

tool and process to help users properly plan and budget their cloud infrastructure. In Section VI, we discuss related work. Finally, we present our conclusions in Section VII.

II. Background: Cloud Pricing Models

In this section, we describe the pricing models offered by Amazon Web Services, Microsoft Azure and Google Cloud Platform, for their different computing offerings. Table I indicates the pricing models supported by each provider[2].

On-demand VMs: On-demand VMs or instances, are VMs requested as needed, where only the used time is paid. They are suitable for short-term or new applications, unpredictable workloads, and for tenants wanting to minimize costs.

Reserved VMs: Provider gives preferential pricing based on long-term commitments or upfront payments. Suitable for stable applications or when provisioning for peak load.

Bid-based VMs: Tenant can define its capacity needs and the maximum per-hour price. Provider assigns resources when they are available at the specified price, thus providing an extreme way to get a regular VMs at the lowest price.

Serverless computing: The Function-as-a-Service (FaaS) model hides platform concerns from tenants, allowing them to run code in response to client requests. Tenant-defined functions are executed as needed; provider charges per request.

III. Adoption of cloud pricing models in Ecuador

In this section, we report on an ongoing study of cloud computing adoption in Ecuador[3], with regard to the payment models that Ecuadorian organizations use or plan on using.

Methodology: We designed a survey instrument with 29 questions in three sections: (1) information about the interviewee; (2) information about the organization; and (3) knowledge and state of adoption of cloud computing in the surveyed organization. A pilot test enabled us to improve the wording and sequence of the questions. The improved survey instrument was input into a LimeSurvey server and distributed via email. We solicited expanded feedback from 25 organizations via personalized emails, phone calls, or in-person interviews. The list of potential respondents was obtained from three sources: the superintendency of companies, ESPOL's Career Center, and the Ecuadorian Software Association; from them a random sample of 300 companies was taken to invite them to participate in the study.

[2]We do not discuss the pricing of renting hosts (vs. VMs) because this is only supported by one cloud provider, and its use and pricing is similar to that of VMs. The difference is in issues regarding licenses and performance.

[3]We analyzed the results obtained up to Jun 23, 2017.

TABLE II: Demographic profile

Organization Size	
Micro (< 10 employees)	25.00%
Small (10-49 employees)	18.48%
Medium (50-199 employees)	21.74%
Large (200+ employees)	34.78%
Organization Age	
< 5 years	22.83%
6-10 years	15.22%
11-15 years	14.13%
16-20 years	16.30%
21-25 years	9.78%
25+ years	21.74%
Organization Sector	
Information and communication	34.78%
Financial and insurance activities	11.96%
Education	8.70%
Wholesale and retail trade	6.52%
Construction	4.35%
Transportation and storage	4.35%
Administrative and support service activities	4.35%
Agriculture, forestry and fishing	3.26%
Professional, scientific and technical activities	3.26%
Other	16.48%

TABLE III: Pricing models vs. organization size

	Fixed	On-demand	Don't pay	Don't know	Total
Micro	9	9	3	2	23
Small	5	11	1	0	17
Medium	8	4	3	5	20
Large	12	8	3	9	32
Totals	34	32	10	16	92

Results: We ran the survey from May through June 2017 and obtained 92 responses. More than 40% of the participating organizations are micro or small enterprises. The demographics of this sample are presented in Table II. Regarding the pricing model, we can observe that micro and small enterprises prefer *on-demand* models (per use) while medium and large enterprises prefer the *fixed one* (See Table III). It is worth mentioning that 57.61% of the participants provided additional information justifying their payment model selection.

Respondents who selected *fixed payment for reserved or fixed service* justified their decision based on the fact that an "annual budget must be approved at the beginning of the fiscal year, allowing the organizations to deal with fixed monthly payments". This seems to indicate that they are used to paying fixed values for the IT services they receive (i.e. licenses, maintenance, software development, Internet, etc.). Furthermore, preferring annual contract-based pricing may indicate a lack of clarity on the cloud's financial model [12].

On the other hand, respondents that indicated that their organizations were comfortable using on demand models, provided justifications that demonstrate a careful analysis of this decision, based on their organizational needs as well as a solid understanding of the benefits of the pay-as-you-go approach. Some examples of the answers provided are:

- "Our business is project-based (or client-based); payment must be in accordance to this model" (2 respondents).
- "We have periods of peak or seasonal activity; we need more resources at peak times" (6 respondents, though one said they would consider changing to fixed monthly payments if their needs were to become more constant).
- "We like to pay only for what we use" (11 respondents).

In addition, one client indicated that even though they believe in the pay-as-you-go model, they have a cap on their cloud expenses, regardless of their client demand.

Finally, less than 2% of the respondents mentioned the use of serverless computing (lambdas or functions).

Takeaway 1: A considerable percentage (36.96%) of Ecuadorian organizations opt for fixed monthly payments instead of on-demand models, to facilitate annual budgeting.

IV. Case Study

In this section we show that choosing the wrong type of cloud computing offering can lead to increased costs.

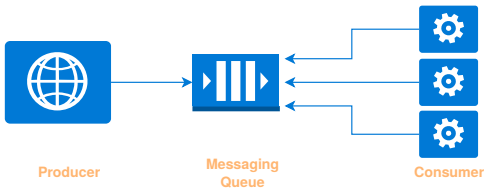
Consider a file compression service that can be consumed, for example, by: (1) A service that bundles invoice files (e.g., PDF, XML and verified XML), such as those used by Ecuadorian companies to fulfill requirements of the Internal Revenue Service (SRI), and sends them to clients via email. (2) A learning platform used by university students to submit project files that are compressed into a single bundle. Or, (3) the call detail records (CDR) reporting module of an IP switching system that compresses files for download.

We chose the first motivating example for our case study. In this example, the invoices are converted to the proper formats and compressed promptly after being entered into the system. These compressed files are later consumed by an email invoice delivery service and a web interface (which shows them to the clients as files that can be downloaded for their convenience).

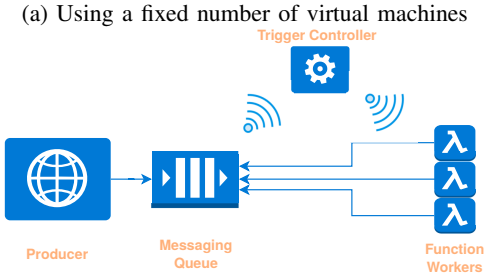
We are interested in the workflow of the compression service, which—following industry best practices—can be implemented as follows. First, the compression request is inserted into a message queue. Next, a consumer service performs the compression work. The exact details of how this workflow is implemented depend on the computing service being used. Lets consider two possible scenarios: (a) Using a fixed number of VM worker nodes, or (b) using on-demand worker functions. Figures 1a and 1b depict these architectures.

Based a real workload from Datil, a company that has an invoicing-as-a-service cloud product implemented on top of a public cloud provider, we know that the load perceived by the system varies depending on the time of the day, with most of the invoicing requests being processed during retail commercial hours; non-commercial hour requests come mainly from gas stations, which operate 24×7. Figure 2 shows the system's average request rate, for every hour in a day.

We calculated the annual computing cost of such compression service in a public cloud, for several pricing scenarios; our results are shown in Figure 3b. We obtained these costs using the Cloud Cost Calculator tool described in Section V.



(a) Using a fixed number of virtual machines



(b) Using on-demand function workers

Figure 1: Cloud architectures for the compression workflow.

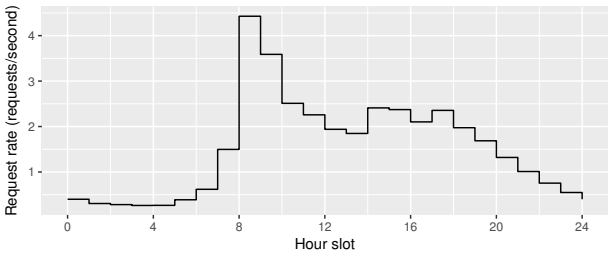


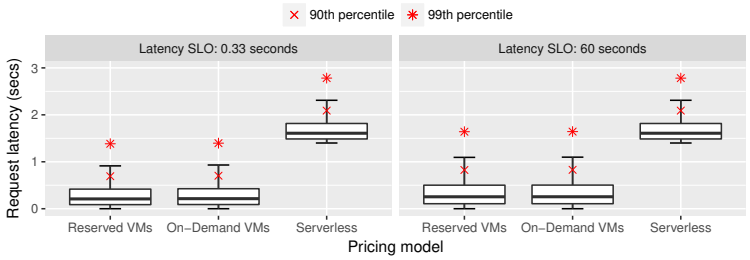
Figure 2: Average request rate, for every hour in a day.

If the service level objective (SLO) is 60 seconds, then opting for serverless computing leads to a 97.5% cost reduction with respect to reserved VMs. However, the reduced costs come at a penalty of increased latency: $1.7\,s$ vs. $0.36\,s$ average latency for serverless and reserved VMs, respectively (see Figure 3a). For the application under study, the increased latency is not an issue, as invoice emails can be sent within an hour after the client being invoiced, and still be within the service level agreement (SLA) that the tenants of the invoicing service are willing to sign. For other applications, such latency penalty ($1.26\,s$ for $90th$ percentile) may be prohibitive. In that case, the tenant could set a smaller SLO; for example, $0.33\,s$, in which case the best option is on-demand VMs.

Takeaway 2: More flexible pricing models can yield significant cost reduction at the cost of reduced performance.

V. Proposed Solution

Our solution is threefold: First, we propose a set of models that represent three alternatives to pay for computing resources in the cloud. Second, we present a software tool that can be used to estimate yearly cloud computing costs for each pricing model. Finally, we describe an easy-to-follow process for tenants to plan and budget their cloud computing costs.



(a) Average and tail (90 and 99 percentiles) latency or time between compression request being sent and the completion of the compression job.

	SLOs (secs)	
Pricing model	0.33	60
Reserved VMs	$595.68	$297.84
Required instances	2	1
On-demand VMs	$566.12	$411.72
Required instances	1.38	1
Serverless computing	(Exceeds SLO)	$7.54

(b) Yearly costs and required VMs for different pricing models and SLOs (0.33 and 60 secs average latency).

Figure 3: Latency and costs obtained using our tool configured as shown in Table IV, for two SLOs: average latency within 0.33 and 60 seconds. Table shows costs and VMs required (on average) to achieve SLO.

A. Model

We use a set of $M(t)/M/*$ queuing theory models with an arrival process fully characterized by a piecewise-constant arrival rate like the one shown in Figure 2.

In an $M(t)/M/*$ model, **arrivals** are governed by a non-stationary or non-homogeneous Poisson process (NSPP or NHPP), where arrivals at time t are a function of t: $\lambda(t)$.

We define $\lambda(t)$ with table of λ_h values, where $0 \leq h < 24$. For a time t in seconds, it follows that $\lambda(t) = \lambda_h$, where $h = t \bmod 3600$. This lets us consider daily patterns, where more requests are received during certain times of the day.

We chose to model arrivals with an NSPP because this is simple enough to allow the user to incorporate dynamic, non-Poisson, features in an intuitive way without requiring extensive data collection or analysis [19].

We did not consider dependent arrival times (i.e., a non-renewal process like a Markovian Arrival Process) to avoid complicating the model and intimidating potential users. In other words, our goal was simplicity and not accuracy, since it is impossible to accurately predict a year's costs based on prior information. Our approach makes it easy to obtain an informed prediction; imprecisions in the estimate are better dealt with by including a contingency amount in the budget for errors and unexpected changes in workload.

Job **service times** are exponentially distributed, thus fully characterized by the mean service time, which can be obtained by the user by running a few tests and timing their service. Standard benchmarking tools can be used for this purpose.

The **number of servers** depends on the type of architecture and pricing model being simulated.

For *reserved VMs*, we use a $M(t)/M/c$ model where the number of servers c is given by $c = \textit{VMs} \times n$, where *VMs* is the number of reserved virtual machines, and n is the number of concurrent requests that can be serviced by each VM. We can find the proper number of servers to use by considering a tenant-defined service level objective (SLO) expressed in terms of the maximum average time in system (i.e., latency of requests), as configured by the user, and running simulations for an increasing number of servers until the target SLO is met. In other words, we solve a simulation-based optimization [7],

Algorithm 1: Exhaustive search (simulation optimization)

Input: Simulator, *Sim*; hourly arrival rates, $\lambda_h[\,]$; mean job service time, s; max. average latency, *SLO*
Output: Least number of servers such that average request latency (calculated by Sim) is $\leq$ *SLO*

```
c := 0;
latency := 0;
repeat
    c := c + 1;
    latency := Sim(c, λ_h[],s);
until latency ≤ SLO;
return c;
```

with a discrete decision variable (c) and a finite feasible set:

$$\underset{\mathbf{c}}{\text{minimize}} \quad f(c) \quad \text{subject to} \quad f(c) \leq \textit{SLO}, \textit{ and } c > 0,$$

where $f(c)$ is the average request latency, as reported by a simulation run of the $M(t)/M/c$ model configured with c servers, and a user-defined λ_h and average job service time. *SLO* is the maximum average latency tolerable by the tenant. By minimizing the number of servers, we are also minimizing the budget as the cost function increases with c.

Alternatively, we could find c using the approximation derived by Green and Kolesar [13]. However, we opted for the simulation optimization approach because the approximation works under the assumption that $\lambda(t)$ is sinusoidal [13].

To solve the optimization we do an exhaustive search using Algorithm 1. This approach is sufficient because we have one optimization variable, the optimization variable is discrete, and there are no local optima; thus, there is a small feasible set to evaluate and we can stop the search once a solution is found.

Note that the $M(t)/M/c$ model described above also works for statically provisioned on-demand VMs with no elasticity, and for the case of reserved hosts. These models only differ with respect to their hourly costs.

For *on-demand VMs with auto-elasticity*, we use a $M(t)/M/c$ model, where c changes dynamically during the simulation, based on the hourly request throughput and a configurable maximum average latency. We refer to this approach as a $M(t)/M/dyn$ model herein.

A *serverless computing* architecture is naturally modeled with an infinite number of servers, as the provider launches as many functions as needed; i.e., with a $M(t)/M/\infty$ model.

Mathematical analysis: The two main metrics that we are interested in are: average time in system and estimated costs.

The waiting time in an $M(t)/M/c$ system can be approximated using pointwise stationary approximation (PSA) [13]. The yearly costs can be calculated once c is known: $\mathit{Costs} = \mathit{VMs} \times \text{VMhourlyCost} \times 24 \times 365$, where $\mathit{VMs} = \lceil c/n \rceil$.

The $M(t)/M/dyn$ system can be represented with a $M(t)/M/c(t)$ model, with a time-varying number of servers; $c(t)$ can be approximated using a stationary $M/M/c$ model for each hourly segment, and then combining the different c values in a stepwise function $c(t)$. The expected waiting time for this model can be then approximated using stationary backlog carryover (SBC) [24]. This approximation would yield a lower-bound on the real waiting time as in a cloud environment the number of servers per hour need not be fixed during a particular hour slot (if the cloud provider charges in a granularity smaller than one hour); tenant-configured elasticity managers could thus lead to waiting times larger than those estimated by this model. It follows that $\mathit{Costs} = 365 \times \sum_{h=0}^{23}(\mathit{VMs}_h \times \text{VMhourlyCost})$, where $\mathit{VMs}_h = \lceil c(h)/n \rceil$.

In a $M(t)/M/\infty$ system requests never queue up, so the average waiting time is 0 and the average time in system is the average service time. The yearly cost can be calculated by knowing that the expected number of arrivals by time t, $\Lambda(t)$ is: $\Lambda(t) = \int_0^t \lambda(s)\, ds$. Thus, for an arrival rate in requests per second, we can calculate the arrivals in a day as $\Lambda(23) = \sum_{h=0}^{23} \lambda(h) \times 3600$. It follows that $\mathit{Costs} = \Lambda(23) \times 365 \times (cr + cgs \times mem)$, where cr is the provider's cost per request, cgs is the provider's cost per GB/second, and mem is the average memory used by the function, in GBs/second.

We opted for simulating the systems instead of applying the formulas and approximations described above for several reasons: (1) the simulation results are more exact than those based on approximations, (2) the simulation code can be easily modified to do a trace-based study with real workloads in addition to generating model-based arrivals, and (3) the simulation approach lets the tenant obtain the waiting time distribution (and not only the expected waiting time), which can be extremely useful as tenants usually care about tail latencies and not only average latency. Furthermore, providers usually charge for use in discrete increments which are easier to handle in a simulation than in a mathematical analysis.

B. Cloud Cost Calculator (CLOUDCAL)

We implemented CLOUDCAL[4] using the SimPy discrete event simulation library,[5] with a generator `Process` for arrivals and `Resources` for computing units.

Generating the request interarrival times is not as simple as sampling from the hourly distribution because this naive approach ignores the changes of $\lambda(t)$ after $t = h$, thus introducing an inertia error [17]. While this approximation may be valid for cases when $\lambda(t)$ varies slowly, we make no such assumption on the behavior of the arrival process. Instead, we generate arrivals using the thinning method [18], which is basically an acceptance-rejection approach that discards (or thins) some events to correct the generation process. Our implementation is based on the code written by Nelson [20].

TABLE IV: Configuration parameters of our tool. The experimental values were those used in Section IV.

Parameter	Format	Experimental vals.
General configuration parameters		
Hourly arrival rates ($\lambda_h[]$, reqs/sec)	Float[]	values in Fig. 2
Latency SLO (max. avg. lat.) (secs)	Float	0.33 and 60
Simulation time (secs)	Integer	86400s
Reserved VMs		
Number of reserved VMs	Integer	1
Max. number of requests that can be serviced concurrently by a VM	Integer	2
Average service time (s, secs)	Float	0.3s
VM hourly cost	Float	0.034
Cost rounded to nearest (secs)	Integer	3600
On-Demand VMs		
Max. number of requests that can be serviced concurrently by a VM	Integer	2
Average service time (s, secs)	Float	0.3s
VM hourly cost	Float	0.047
Cost rounded to nearest (secs)	Integer	3600
Serverless computing		
Time to setup function (secs)	Integer	1.4s [14]
Average service time (s, secs)	Float	0.3s
Memory allocated to function (MB)	Integer	128
Cost per request	Float	0.0000002
Cost per GB per second	Float	0.00001667

Table IV lists the configuration parameters that need to be provided to be able to estimate the yearly cloud costs.

The simulation approach has an advantage over simple cloud cost calculators like the Amazon Web Services Simple Monthly Calculator[6] in that it lets user make their decision based on the cost and the predicted performance for each specific configuration. This tradeoff is important because, as we have shown in this paper, the reduced costs achievable with more dynamic pricing models come with a performance penalty that may not be tolerable to some applications.

For each pricing model, CLOUDCAL finds the configuration that yields the minimum cost while not exceeding the tenant's latency SLO. These configurations are Pareto optimal for their pricing models. The set of configurations with their expected cost and performance is presented to the user, who can select the most economical option or the highest in performance, depending on the tenant's internal utility function.

C. Cost budgeting process

We propose the following process to determine which pricing model is most adequate for a cloud tenant:

1) Configure CLOUDCAL.
2) Run CLOUDCAL; output is set of Pareto optimal configurations that minimize cost while meeting SLO.
3) Choose one of the Pareto optimal configurations; none of them outperforms the others in all criteria.

[4]Available at: https://github.com/ebozag/CLOUDCAL

[5]Available at: https://simpy.readthedocs.io/en/latest/

[6]Available at: https://calculator.s3.amazonaws.com/

4) Add a contingency amount to the cost obtained with CLOUDCAL; consider a 5 to 10% of unexpected costs due to changes in the workload or other issues[7].
5) Include the total cost in the yearly budget (cost + contingency amount). Also, remember to take into account the inflation rate for the whole IT budget.

D. Accuracy

The queuing theory models used on CLOUDCAL are accurate representations of the real system, with the exception of assuming exponential arrivals and exponential service times. We chose this approach as it makes it easy for the user to configure the system. However, if a more accurate representation is desired, an advanced user can modify CLOUDCAL to use any other distributions (including, an empirical distribution representing the actual observed system behavior)[8].

E. Running time

We ran CLOUDCAL ten times with the configurations used in this paper (including both SLOs), on a Linux machine with 8×1.4 GHz cores, and 8 GB of RAM. Our results show that running CLOUDCAL prior to budgeting yearly costs is not prohibitive, as it took, on average, $30.1\,s$ to calculate all pricing scenarios, including finding the proper number of servers to use via the simulation optimization approach.

VI. RELATED WORK

The problem of deciding on a pricing model for cloud providers has been studied in depth [1], [6], [16], [21], [28].

From the client's perspective, others have studied which is the best pricing model and number of VMs to select for large cloud deployments, given a particular application like MapReduce and a target deadline [8], [9], [27]. The Serverless model is a very recent cloud offering (end of 2014 for AWS Lambda), and was thus not considered in these studies.

Our tool provides a significant improvement over the current cloud budgeting process used by small and medium enterprises (SMEs), as described in [10]. SMEs that cannot afford large IT departments must resort to simplistic calculations or online budgeting tools to get rough cost estimates. Our approach has the advantage of providing a better budget estimation and the inclusion of performance metrics, hence the choice of a pricing model can be guided by the estimated budget and SLOs.

Finally, choosing the best VM for an application is an orthogonal problem [3], [26], [29]. Prior to configuring CLOUDCAL, the user can profile the application on all VM types, or do an instrumentation-based performance prediction [29].

VII. CONCLUSIONS

We presented a set of models that can be used to calculate the performance and costs of different cloud computing pricing models, for a specific service given a configurable workload. We simulate those models in a software tool that outputs the set of Pareto optimal cloud configurations that meet tenant-defined SLOs. Finally, we defined a process that can be used to properly plan the yearly cloud computing budget for a service.

[7]This applies even in the case of reserved VMs, as a tenant can decide mid-year to add one or more VMs to its server farm.

[8]We note that if we had opted for using mathematical approximations instead of simulations, then we could use Kingman's approximation to calculate the waiting times based on any general (non-exponential) distribution [15].

Acknowledgments: This research was supported in part by gifts from Microsoft, Google and Amazon. We thank our anonymous reviewers for their valuable comments which helped improve the paper.

REFERENCES

[1] Al-Roomi *et al.*, "Cloud computing pricing models: a survey," *Intl. J. Grid and Distrib. Comp.*, vol. 6, no. 5, 2013.
[2] Alharbi *et al.*, "Strategic value of cloud computing in healthcare organisations using the balanced scorecard approach: A case study from a saudi hospital," *Proc. Comp. Sci.*, vol. 98, 2016.
[3] O. Alipourfard *et al.*, "CherryPick: Adaptively unearthing the best cloud configurations for Big Data analytics," in *Usenix NSDI*, 2017.
[4] M. Armbrust *et al.*, "A view of cloud computing," *Comm. ACM*, vol. 53, no. 4, 2010.
[5] S. Arvanitis, N. Kyriakou, and E. Loukis, "Why do firms adopt cloud computing?" *Telematics and Informatics*, 2016.
[6] H. Bhargava and M. Gnagwar, "Pay as you go or all you can eat? Pricing methods for computing and information services," in *HICSS*, 2016.
[7] Y. Carson and A. Maria, "Simulation optimization: Methods and applications," in *Proc. Winter Simul. Conf.*, 1997.
[8] N. Chohan *et al.*, "See spot run: Using spot instances for mapreduce workflows." in *USENIX HotCloud*, 2010.
[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Comm. ACM*, vol. 51, no. 1, 2008.
[10] A. Eivy, "Be wary of the economics of serverless cloud computing," *IEEE Cloud Computing*, vol. 4, no. 2, 2017.
[11] B. Fakieh *et al.*, "SMEs and cloud computing: The benefits to the national economy and global competitiveness," in *Proc. EMCIS*, 2016.
[12] A. Gohad *et al.*, "Cloud pricing models: A survey and position paper," in *IEEE Intl. Conf. Cloud Comp. Emerg. Mrkts.*, 2013.
[13] L. Green and P. Kolesar, "The pointwise stationary approximation for queues with nonstationary arrivals," *Mgmt. Sci.*, vol. 37, no. 1, 1991.
[14] S. Hendrickson *et al.*, "Serverless computation with open Lambda," in *USENIX Conf. Hot Topics in Cloud Comp.*, 2016.
[15] J. Kingman, "The single server queue in heavy traffic," *Mathematical Proc. Cambridge Philosophical Soc.*, vol. 57, no. 4, 1961.
[16] G. Laatikainen, A. Ojala, and O. Mazhelis, "Cloud services pricing models," in *Intl. Conf. Software Business*, 2013.
[17] L. Leemis and S. Park, *Discrete-event simulation: A first course*. Pearson Prentice Hall, 2006.
[18] P. Lewis and G. Shedler, "Simulation of nonhomogeneous poisson processes by thinning," *Naval Res. Logistics Qtly.*, vol. 26, no. 3, 1979.
[19] Nelson and Gerhardt, "Modelling and simulating non-stationary arrival processes to facilitate analysis," *J. Simulation*, vol. 5, no. 1, 2011.
[20] B. Nelson, *Foundations and methods of stochastic simulation: A first course*. Springer, 2013.
[21] N. Nguyen *et al.*, "Resource management in cloud networking using economic analysis and pricing models: A survey," *Com. Surv. Tut*, 2017.
[22] P. Priyadarshinee, M. Jha, and R. Raut, "Cloud computing adoption in SMEs: A literature review," in *AIMS Intl. Conf. Management*.
[23] Schniederjans and Hales, "Cloud computing and its impact on economic and environmental performance," *Decision Support Sys.*, vol. 86, 2016.
[24] R. Stolletz, "Approximation of the non-stationary $m(t)/m(t)/c(t)$-queue using stationary queueing models: The stationary backlog-carryover approach," *European J. Oper. Res.*, vol. 190, no. 2, 2008.
[25] Vajjhala and Ramollari, "Big data using cloud computing: Opportunities for small and medium-sized enterprises," *lib. euser. org*, 2016.
[26] S. Venkataraman *et al.*, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *Usenix NSDI*, 2016.
[27] A. Wieder *et al.*, "Orchestrating the deployment of computations in the cloud with Conductor," in *Usenix NSDI*, 2012.
[28] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Comp.*, vol. 1, no. 2, 2013.
[29] N. Yadwadkar *et al.*, "Selecting the best VM across multiple public clouds: A data-driven performance modeling approach," in *SOCC*, 2017.