Technical Report No. 2013-613

Estimating Resource Costs of Executing Data-Intensive Workloads in Public Clouds[1]

Rizwan Mian, Patrick Martin, Farhana Zulkernine
School of Computing
Queen's University
Kingston, Ontario, Canada, K7L3N6
{mian, martin, farhana}@cs.queensu.ca

Jose Luis Vazquez-Poletti
Departamento de Arquitectura de Computadores y Automatica
Universidad Complutense de Madrid
28040. Madrid, Spain
jlvazquez@fdi.ucm

[1] A shorter version of this report has been published as Mian, R., Martin, P., Zulkernine, F., and Vazquez-Poletti, J.L. 2012. "Estimating Costs of Data-intensive Workload Execution in Public Clouds." *10th International Workshop on Middleware for Grids, Clouds and e-Science (MGC) in conjunction with ACM/IFIP/USENIX 13th International Middleware Conference 2012*, ACM, article. 3, Montreal, QC, Canada.

Estimating Resource Costs of Executing Data-Intensive Workloads in Public Clouds

Rizwan Mian, Patrick Martin, Farhana Zulkernine
School of Computing
Queen's University
Kingston, Ontario, Canada, K7L3N6
{mian, martin, farhana}@cs.queensu.ca

Jose Luis Vazquez-Poletti
Departamento de Arquitectura de Computadores y Automatica
Universidad Complutense de Madrid
28040. Madrid, Spain
jlvazquez@fdi.ucm

ABSTRACT

The promise of "infinite" resources given by the cloud computing paradigm has led to recent interest in exploiting clouds for large-scale data-intensive computing. In this paper, we present a analytical model to estimate the resource costs for executing data-intensive workloads in a public cloud. The cost model quantifies the cost-effectiveness of a resource configuration for a given workload with consumer performance requirements expressed as Service Level Agreements (SLAs), and is a key component of a larger framework for resource provisioning in clouds. We instantiate the cost model for the Amazon cloud, and experimentally evaluate the impact of key factors on the accuracy of the model.

Keywords

Cloud computing, cost model, resource provisioning.

1. INTRODUCTION

Public clouds, because of their pay-as-you-go flexibility and lack of up-front costs, are attractive to companies interested in lowering their operational IT costs. In making the decision to move to a public cloud, however, a company must be able to determine an appropriate configuration of cloud resources for an application and so predict the cost-effectiveness of moving the application. The cost-effectiveness is determined by the cost of the required resources and the application performance achieved with those resources.

We previously proposed a framework for resource provisioning of data-intensive applications in a cloud [1]. Given the cost structure from a cloud provider and the set of application workloads and the negotiated SLAs, our method determines a resource allocation with minimal cost for those workloads.

The cost model, which plays a key role in the decision-making process, produces a single dollar value that captures the cost-effectiveness of a particular configuration in terms of both the resources allocated and the applications' performance. The latter is represented as the penalty value imposed if SLAs associated with the workloads are not achieved. A configuration where more resources than needed are allocated pays a higher than necessary cost in terms of resources. A configuration where insufficient resources are allocated, on the other hand, pays a higher cost in terms of SLA penalties.

In this paper we discuss the different components of our cost model and explain how each can be determined for public cloud with a pay-as-you-go pricing strategy. We apply the cost model to the Amazon cloud [2] and present a set of experiments that investigate the impact of key factors affecting the cost model.

The remainder of the paper is structured as follows. Section 2 outlines related work. Section 3 discusses different resource types and pricing schemes in Infrastructure-as-a-Service (IaaS) clouds. Section 4 describes our cost model. Section 5 presents a set of experiments using the cost model with sample data-intensive workloads on Amazon EC2 and Section 6 concludes the paper.

2. RELATED WORK

The problem of resource provisioning in public clouds has recently received a great deal of attention. Vazquez-Poletti et al. [3] determine a suitable number of homogenous virtual machines (VMs) to execute a given workload in the Amazon cloud based on values of a novel cost-performance metric (C/P). Their method does not consider other resource costs such as storage or communication, and is applied to a workload consisting of a single work-unit, which is equivalent to a single query or a transaction. The C/P-based approach does not account for any SLAs, or its penalties in case of violations.

Tsakalozos et al. [4] use principles from microeconomics to dynamically converge to a suitable number of VMs for a workload given a user's budget. Their approach is used at runtime and cannot be used to provide an a priori prediction of resource allocations. Bicer et al. [5] also propose a runtime resource allocation framework to support time or cost constrained application execution in a hybrid cloud. Their cost model's parameters are acquired by monitoring an executing application.

Sharma et al. [6] develop a pricing model to provide "high" satisfaction to the users and the providers in terms of QoS guarantees and profitability requirements, respectively. The thrust of their work is towards *valuation* of cloud resources, and they employ financial option theory and treat the cloud resources as underlying assets.

Li et al. [7] propose a cost-effective data reliability mechanism to reduce the storage cost in a public cloud. Their mechanism checks the availability of replicas and reduces storage consumption up to one-third by making certain assumptions on the reliability. Assunção et al. [8] investigate the benefits that organizations can reap from a hybrid cloud. In particular, they offload work to a public cloud to reduce deadline violations and associated cost. Du [9] looks at maximizing revenue from cloud vendor's perspective by modeling hybrid and public cloud markets using Markovian traffics. Interestingly, her work suggests that the hybrid cloud is the most profitable model for cloud vendors.

Amazon's monthly calculator [10] estimates charges for Amazon EC2 resources, if they are used for an entire month. While the time-bound on a workload may be unknown in advance, we argue that the time-unit of a month for resource cost is excessively coarse-grained. The calculator does not have any knowledge of a workload and cannot account for application performance with a given set of resource allocations.

Our cost model accounts for all the resources needed (compute, storage and network) to execute a data-intensive workload consisting of multiple queries and transactions accessing multiple data partitions. Further, our cost model accommodates user-defined SLA and associated penalties. Moreover, the execution cost is provided at the granularity of an hour.

3. DIFFERENT RESOURCE TYPES AND PRICING SCHEMES IN IAAS CLOUDS

There are primarily three types of resources needed to execute a workload in an IaaS cloud, namely compute, storage, and network resources. For each resource type, there are different pricing schemes and sub types of resources. The resource sub types used in our cost model are identified in section 3.1. Meanwhile, we consider pay-as-you-go scheme in our cost model. This is because we find pay-as-you-go scheme is more in-line with the cloud philosophy proposed by Armbrust et al. [11] that includes: (a) no upfront commitment, and (b) pay-for-use pricing scheme. All major cloud vendors Amazon cloud [2], RackSpace [12], and GoGrid [13] offer infrastructural resources on a pay-as-you-go basis. We discuss diversity in resource sub types and different pricing schemes below, using Amazon cloud as an example.

3.1 Resource Types and Sub Types

The resource costs vary by the resource sub type and the usage duration. A VM is a typical compute unit in an IaaS cloud. VMs differ in their computational and memory resources, network bandwidth available to them and latency of disk I/O. Amazon offers a diverse portfolio of VM types [14] aimed at different classes of applications, as shown in the Table 1.

The storage also comes in different sizes and flavors as shown in Figure 1. Every VM has a local storage [15], which is usually in the hundreds of gigabytes and has no access costs. Data on the local storage is volatile, and the data is lost once the user gives back the VM to the vendor. All other storage types are usually accessed over the network. For example, Amazon S3 [16] stores data over multiple network devices. It is a key-value storage that can store a value sized from 1 byte to 5 terabytes. Similarly, DynamoDB [17] is a key-value storage with very low latency. However, it places a limit (64KB) on the size of the key and the value. In contrast, the Elastic Block Storage (EBS) volumes [18] are raw storage which are formatted and mounted, and appear as network disks on Amazon VMs. Their sizes can vary from 1 GB to 1 TB.

Table 1: VM Classes, their distinguishing features, example applications and cost bands.

VM Class	Distinguishing Features	Example Applications	Cost Band
Opportunistic	Small amount of CPU resources augmented with spare CPU capacity of the host server when available.	Lower throughput applications and web-sites that require additional compute cycles periodically, but are not appropriate for applications that require sustained CPU performance.	Very low cost
General purpose	Balance between compute, memory, and network resources.	Small and mid-size databases, data processing tasks that require additional memory, and caching servers.	Lower end
Compute optimized	Higher ratio of CPUs to memory than other VM classes.	CPU-bound scale out applications. Examples include high traffic front end fleets, web servers, batch processing.	Middle order
Memory optimized	Lowest cost per GB of memory among EC2 VM types.	Database applications and distributed caches.	Middle order
Storage optimized	Directly attached storage optimized for applications with specific disk I/O and storage capacity requirements.	NoSQL databases like Cassandra [19] and MongoDB [20] which benefit from very high random I/O performance and the low request latency of directly attached Solid State Drives (SSDs).	Additional premium on base VM cost
Cluster computing optimized	High core density and supports cluster networking.	Computational chemistry, rendering, financial modeling, and engineering design.	High end

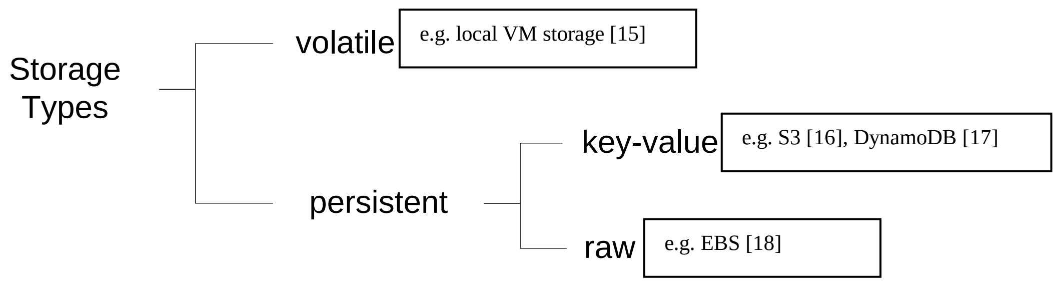

Figure 1: Examples of different storage types in the Amazon cloud.

Amazon also provides different network performance for different VM types and storage resources. However, bandwidth available to a VM is not expressed as a number or a range. Instead, Amazon categorizes the network performance of a VM type into four bands, namely *very low*, *low*, *moderate* and *high*. The notable exception is 10 Gb/s for some VM types such as storage optimized VMs. Fortunately, it is possible to acquire guaranteed bandwidth (500 and 1000 Mbps) on some VM types to access EBS at an additional premium.

The maximum reported bandwidth between EC2 VMs and S3 was around 21MB/s in 2008 as reported by Palankar et al. [21]. They also noted that as the number of threads on a VM increases, the per-thread bandwidth decreases but the aggregate bandwidth increases. With two machines running with six threads each, the average bandwidth was 17 MB/s for file sizes of 100 MB. Bergen et al. [22] note in 2011 that the average bandwidth reaches up to 12 MB/s for file sizes between 100MB and 5GB.

3.2 Pricing Schemes

The same resource sub type may also be offered using numerous pricing schemes. The pricing schemes usually differ in the resource acquisition method. For example, the popular schemes for acquiring VMs are: (a) pay-as-you-go, (b) auction-based, and (c) reserved.

Using pay-as-you-go scheme, a VM is available within a few minutes when requested, and the user pays by the hour. This frees the user from planning and long-term commitment. Pay-as-you-go scheme is also known as an *on-demand* scheme.

The storage usage is usually billed by the month, though charges for partial usage may apply. In addition, there are network costs associated with accessing the storage, which is typically measured by the number of accesses. Further, transferring data in and/or out of a public cloud usually incurs costs. This cost is estimated by the amount of data transferred. The storage cost may also vary by the size of storage acquired. Amazon decreases the cost rate for S3 as the storage size increases. This decrease follows a step function instead of a linear function.

The auction-based or spot schemes enable a user to bid for the unused VMs. The spot price fluctuates periodically depending on the supply and demand for the VMs. A user gets a VM only when the bid exceeds the current spot price. The user may willfully return the unneeded VMs or the vendor may grab the allocated VMs *forcefully* when the spot price exceeds the user's bid. This scheme is usually cheaper than the on-demand scheme, but the vendor may place a minimum or a baseline price for the bids. More importantly, the user may have to wait, or worse suffer a loss of a VM when the spot price increases. This adds complexity to the market model. Understanding how cloud vendors set the spot prices is useful for users, who can decide how much to bid. Ben-Yahuda et al. [23] find that Amazon seems to generate spot prices using a *hidden* reserve price function, which only complicates the market model.

A user may be willing to commit for a long period of time such as a month or a year. In this case, the reserve scheme is suitable and it costs less than the on-demand scheme. The money is paid upfront in a lump sump, and is usually non-refundable. Therefore, some planning is required to avoid wasting of money and/or unused VMs. A cloud provider may offer additional discounts on other resource types when subscribing a service for a long-term. For example, Amazon offers some percent discounts on the storage when reserving VMs. The reserved scheme is also known as the *pre-paid* scheme.

4. COST MODEL

Given a set of applications $\boldsymbol{A = \{A_1, A_2, \ldots, A_m\}}$, we say that a workload $\boldsymbol{W_i}$ for $\boldsymbol{A_i}$, is a set of requests that are issued by the set of clients of $\boldsymbol{A_i}$. Each request is an instance of a request type $\boldsymbol{R_{ij}}$ from a set $\boldsymbol{R_i = \{R_{i1}, R_{i2}, \ldots, R_{in}\}}$ for $\boldsymbol{A_i}$. The databases used by $\boldsymbol{A}$ consist of a set of data objects $\boldsymbol{D = \{D_1, D_2, \ldots, D_m\}}$. A request type $\boldsymbol{R_{ij}}$ for $\boldsymbol{A_i}$ accesses some data objects in $\boldsymbol{P_i} \subseteq \boldsymbol{D}$, and has a service level objective $\boldsymbol{SLO_{ij}}$. We call $\boldsymbol{P_i}$ a *data partition* and assume that $\boldsymbol{W_i}$ accesses data from $\boldsymbol{P_i}$. The SLA for $\boldsymbol{W_i}$ is composed of the set of all $\boldsymbol{SLO_i}$'s for the request types in $\boldsymbol{A_i}$. We need compute, storage and network resources to execute $\boldsymbol{W_i}$. A configuration $\boldsymbol{C}$ for a set of workloads, $\boldsymbol{W = \{W_1, \ldots, W_n\}}$, contains the following:

- A set of VMs $\boldsymbol{V = \{v_1, v_2, \ldots, v_r\}}$, where each VM $\boldsymbol{v_k}$ is a specific type (for example *small*, *large*, *xlarge*). Each VM type has a specific set of system attributes (e.g. OS, memory, cores), and a specific cost rate.
- A mapping of the workloads, $\boldsymbol{W}$, to VMs in $\boldsymbol{V}$ such that every workload is assigned to one VM.
- A mapping of data partitions used by $\boldsymbol{W}$ to VMs in $\boldsymbol{V}$ such that every data partition is assigned to at least one VM. The partitions are stored in the cloud storage. The partitions typically vary in sizes and have different access patterns, resulting in different storage and network costs. Overlapping partitions on the same VM share the same copy of the common data objects. Assignment to more than one VM involves *replication* of the partition, and we assume that the replicas are read-only.

An application that is executed with a given configuration in a public cloud results in a cost to the application owner. This cost, as previously noted, is made up of resource costs and penalty costs if SLAs are not met. The resource costs in a pay-as-you-go public cloud are primarily associated with compute, storage and network resources. Since a workload bound is not known in advance, we choose to represent the cost of a configuration per unit time, specifically in \$ / hour. The cost for configuration $\boldsymbol{C}$ in a pay-as-you-go IaaS cloud can therefore be stated as:

$$Cost\$(\boldsymbol{C}) = Compute\$(\boldsymbol{C}) + Storage\$(\boldsymbol{C}) + Network\$(\boldsymbol{C}) + Penalty\$(\boldsymbol{C})$$

A VM is a typical compute unit in an IaaS cloud. The VM types differ in their computational and memory resources [14]. Their price is generally metered by the hour and partial usage is rounded up to the next hour. The compute cost *Compute\$(C)* can be expressed as:

$$Compute\$(C) = \left\lceil \sum_{v \in V} VMCost\$(v) \right\rceil$$

where $\boldsymbol{V}$ is the set of VMs in the configuration $\boldsymbol{C}$, and *VMCost\$(v)* is the hourly cost of a VM *v*.

There are different types of storage with different properties and prices. We consider our data partitions are stored in a cloud storage, which is metered by the month. We prorate the monthly cost down to an hour. The hourly cost for the storage used in a configuration $\boldsymbol{C}$ is estimated by:

$$Storage\$(C) = \left\lceil \frac{\$q \times E}{month\_hours} \right\rceil$$

where *\$q* is the unit cost of storage (in dollar per gigabyte per month), E is the aggregated size of data partitions rounded up to the next gigabyte, and *month_hours* is the number of hours in a month (e.g. 24h ×30days). Any fractional cost is rounded up to the next cent.

There are two main sources of network costs in the data-intensive applications, namely transferring data in and/or out of the cloud and accessing storage. The transfer costs are difficult to capture in a general model since they are dependent on a number of specific characteristics of the application owner. We therefore assume that the data is already in the cloud, which is accounted for by the storage costs, and therefore, we only consider the network costs associated with accessing data storage. The network costs are therefore estimated by:

$$Network\$(C) = \left\lceil \sum_{v \in V} c_v \times \$s \right\rceil$$

where c_v is the estimated number of accesses to the network storage in a time-unit (hour) and *$s* is the unit network cost for accessing storage. Like storage costs, the network cost is rounded up to the next cent.

We propose a function that assigns a penalty each time-unit in which a breach occurs. This is reasonable since we use average performance metrics (such as average throughput) over a time-unit to detect a breach. For a particular configuration ***C*** and a request type *r*, the penalty incurred in a given time-unit (hour) is given by:

$$Penalty\$(C) = \sum_{r \in R} pcond(r, C) \times penalty(r)$$

where *penalty(r)* is the penalty value (in $) for requests of type *r* missing their SLO in a time-unit. The binary function *pcond* indicates whether or not an SLO defined over *r* and ***C*** has been violated. We provide examples of *pcond* when we instantiate the cost model for the Amazon EC2 public cloud in Section 5.

5. EVALUATION

We now examine the effectiveness of our proposed cost model for pay-as-you-go IaaS public clouds. We define an instance of our model for Amazon EC2 [2], which is currently a major IaaS cloud vendor [24], and consider possible configurations for a multitenant database application with different tenants, each with their own workload. We compare the cost estimates produced by our model with the actual costs incurred on EC2 for a variety of configurations of the multitenant database application. We find that our model produces accurate cost estimates in all cases.

We observe that there are three main variables that influence the cost for a configuration:

1. The VM types used in the configuration,
2. The mix of workloads or tenants involved in the configuration, and
3. The SLOs enforced in the configuration.

The network cost varies with the workload and the VM type, while the storage cost varies with the tenant type. We therefore present the results of three experiments where each factor is varied while holding the other two constant. We compare the estimated resource costs directly against the invoice rendered by Amazon. We calculate the penalties based on the measured metrics (throughput and response time). We determine the error in the cost estimate for each case.

5.1 Tenants and their Workloads

The tenants for the multitenant database application used in our experiments are described in Table 2. The tenants are based on well-known transactional (TPC-C and TPC-E), and analytical (TPC-H) benchmarks [25]. The tenants' workloads are made up of requests from the benchmarks and are chosen to exhibit different behaviours, namely read-only, write-heavy and mixed read/write.

Table 2. Example Application Tenants

Tenant	Workload	Data-bases	Request types
a	read-only	TPC-H	Q1, Q6
b	write-heavy	TPC-E	trade-order, trade-update
c	read-write (mixed)	TPC-H, TPC-C	Q12, Q21 (TPC-H), new-order, payment (TPC-C)

5.2 Cost Model for Amazon EC2

We examined the pricing structure offered by Amazon EC2 and assigned values to the cost variables in our cost model as follows.

Compute costs: We consider three VM types offered by EC2 in order to include cases of under, over and optimal resource provisioning for the example applications. We use the *small*, *large*, and *xlarge* VM types as shown in Table 3. The VM types vary in their price, processing power and their capacity to hold data in memory. For example, all three tenants fit in the memory of xlarge but none of them fit in the memory of small, and only two tenants fit in the memory of large. Studies have shown that EC2 does not always provide consistent performance so we chose to run our experiments in the region with least variance, namely US-East-1d [26].

Table 3. VM Types for Amazon EC2

VM Type	Cores (#)	Memory (gb)	Cost/hr($)[1]
Small	1	1.7	0.08
Large	2	7.5	0.32
Xlarge	4	15	0.64

Amazon EC2 provides the ability to place VM instances in multiple locations [2]. These locations are composed of *regions* and *availability zones* within them. Availability zones are distinct locations (presumably different data centers) that are engineered to be insulated from failures in other zones, and to provide inexpensive, low latency network connectivity to other zones in the same region.

Storage costs: We choose Elastic Block Storage (EBS) [18] to store tenant databases, primarily because EBS appears as a network mounted hard disk. We also find EBS convenient for the evaluation purposes. We created templates of tenant databases for TPC-C, TPC-H and TPC-E, called snapshots, prior to workload execution. These snapshots are booted to provide ready to use volumes (provisioned storage) for workload execution. We wrap up the tenant databases with

[1] Amazon has revised these costs since we started experimentation.

some binaries and settings, and store that as an *image*.[1] The major portion of binaries consists of a configured MySQL database management system (DBMS) hosting all the database tenants. This greatly simplifies the engineering process, and the workloads can start execution as soon as the compute and storage resources are available. The DBMS binaries and settings add a small amount to the storage cost compared to tenant databases, and are included in the storage costs.

The EBS storage cost consists of two parts, namely snapshot storage and provisioned storage. The snapshot storage cost occurs due to the images being stored in a permanent archive, S3 [16]. Meanwhile, the charge for provisioned storage is made when the VMs are booted with images stored in S3 [27]. The charges for provisioned storage accumulate on an on-going basis till the VMs are terminated.

Snapshot storage has a cost of \$0.125/GB/month and provisioned storage has a cost of \$0.10/GB/month [18]. Both snapshot and provisioned storage are billed by the month, but are metered by the hour [28], so we can estimate their associated cost by the hour. Their cost is rounded up to the next integer cent. The hourly cost of the snapshot storage is estimated to the next cent by:

$$SnapshotStorage\$(C) = \left\lceil \frac{\$q \times E}{month\_hours} \right\rceil$$

where \$q is the unit cost of snapshot storage (\$0.125), E is the aggregated size of partitions rounded up to the next gigabyte, and month_hours is the number of hours in a month (24h ×30days). The provisioned cost is estimated using the same method, except the \$q becomes the unit cost of provisioned storage (\$0.10 per GB-month). We validate our method of estimating storage costs against the daily increments in the storage cost reported by Amazon Activity [29].[2]

Network costs: Amazon places numerous network charges such as data transfer in and out of clouds, and data transfer across zones. In our case, the data is already in the cloud and the clients and DBMS exist in the same zone so the only charges are for accesses to EBS storage.

We experimentally determine the number of accesses required for each workload on each VM type. We then estimate the number of storage accesses per hour for a mix of workloads on a VM type as the *average* of the number of accesses by each individual workload in the mix. When considering all workload combinations, we find that the error in the network cost provided by simple average varies from -\$0.04 to \$0.12 (range of \$0.16). We can possibly improve the accuracy of estimating the network cost by prorating the network access at the request level but this comes at the cost of increased complexity. We find that the simple average provides reasonable accuracy in most cases.

[1] Our image (ami-7bc16e12) is publicly available at: http://thecloudmarket.com/owner/966178113014. Once the image is instantiated, the clients can connect (ssh in) to the instance and access the MySQL DBMS as root user with wlmgmt password.

[2] We record Account Activity on a daily basis. Our daily records for April 12 to July 12 are available here: http://research.cs.queensu.ca/home/mian/index_files/Page485.htm

We need some method of aggregating experimental results, and realize that "average" is meaningful for a normal-like distribution. With the scarcity of training samples, we are unable to verify the distribution of results and resort to using average as the aggregation method. Average, after all, provides a "smoothing" effect over available measurements. Nonetheless, the reported results must be interpreted with caution.

Penalty cost: We calculate the penalties based on the measured throughput and response times. The binary function *pcond* is defined differently for response time and throughput. For response time on a request *r*, it is defined as

$$pcond\ (r, C) = \begin{cases} 1\ if\ avgRespTime(r, C) \geq threshold(r), \\ 0\ otherwise \end{cases}$$

where *avgRespTime(r,**C**)* is the average response time for request *r* in ***C***, and *threshold(r)* is the required response time specified in the SLO for the request *r*.

For throughput on a request *r*, *pcond* is defined as

$$pcond\ (r, C) = \begin{cases} 1\ if\ avgThroughput(r, C) \leq threshold(r), \\ 0\ otherwise \end{cases}$$

where *avgThroughput(r,**C**)* is the average throughput for transaction *q* in *C,* and *threshold(r)* is the required throughput specified in the SLO for the request *r.*

5.3 Experiments

All the requests in a workload execute concurrently until the specified time. A request type in a workload may have multiple instances that execute concurrently. Our workloads are bound by time. Until the completion time, a request instance is continuously re-submitted if finished early. This ensures that the *request mix* remains present at a DBMS throughout the time bound.

The workloads execute at the optimal Multi-Programming Level (MPL)[3] of a VM type. All workloads are presently weighted equally. Therefore, the optimal MPL is divided equally when multiple workloads are executing together. Any remainder MPL value is used by Q1 or Q12 query instances. This is an ad hoc choice to keep the DBMS under optimal load.

For example, suppose read-only and write-heavy workloads execute on the large VM type where the optimal MPL level is 75. There are four request types in the read-only and write-heavy workloads (Q1, Q6, trade-order and trade-update), each getting an equal MPL share (of 18) of the optimal MPL value (75). The remainder of 3 adds into the MPL share of Q1. The MPL share of a request type represents the concurrent instances of that request type in the request mix executed at a DBMS.

We parameterize our cost model by the hour, therefore, the same request mix executes for the entire duration. The *warm up* period is included in the measurements. This is important because the DBMS is populated with the workloads' data. This data is read off the network disk, which results in the network cost. This data is of considerable size. Once the data

[3] Conceptually, the throughput increases as the number of concurrent clients increase, up to a point where the multi-programming level (MPL) plateaus, and then it starts decreasing. We consider the optimal MPL value to be the beginning of the plateau. We determine the optimal MPL value for a VM type experimentally. The optimal MPL levels of *small*, *large* and *xlarge* are 14, 75 and 115, respectively.

is in the DBMS caches, much of the data needs can be served locally from various caches. Therefore, the warm up period is part of the measurement process in this case.

We perform sensitivity analysis, in which all but one variable is varied, and the remaining are kept constant. That is, we vary the user-controllable variables one-at-a-time. This allows us to evaluate the cost model with each variable individually. We also see the impact of each variable on the workload execution cost independently. The experiments are:

1. Varying VM Type (presented in section 5.3.1)
2. Varying Workload Mix (presented in section 5.3.2)
3. Varying SLA Penalties (presented in section 5.3.3)

5.3.1 VM Type

We first examine the effectiveness of the cost model as we vary the VM type used. We execute a combination of all the tenants (a, b and c from Table 2) with no SLOs defined, and hence no penalty costs, on each VM type. We compare the estimated and the measured costs for each case in Figure 2. We observe no errors in the VM and the storage costs. The average error in the network cost is about $0.01, which is about 6% (over-estimation)[1] of the total measured cost on average. The VM costs increase for more powerful VM types, but the network cost decreases. We attribute the reduced network costs to a larger buffer pool, which reduces the number of network accesses.

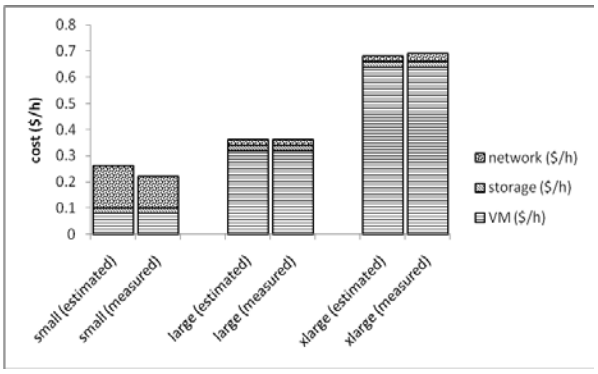

Figure 2: Estimated and measured costs for all workloads executing simultaneously on each VM type with no SLOs.

5.3.2 Workload Mix

We next examine the effectiveness of the model as we vary the mix of tenants running together on the same VM. We run different combinations of the tenants' workloads on a small VM with no SLOs defined, that is, no penalty costs. We compare the estimated and the measured costs in Figures 3 and 4.

Figure 3 shows that, as expected, we are able to accurately estimate the costs for the individual workloads because we collect the metrics for the workloads on an individual basis.

Figure 4 shows that when workloads are run in combination on a VM we provide accurate estimates for storage and compute costs but the network estimates differ from the actual costs. The average error in the network cost is about $0.03, which is about 13% of the total measured cost on average. We also see a large difference between the estimated and the measured cost for the bc workload. In this case, the network cost error is $0.12, which is about 55% of the total measured cost. We believe this is because the simple average does not take into account the change in workload intensity and its effects on buffering and lock contention when the workloads are executed simultaneously.

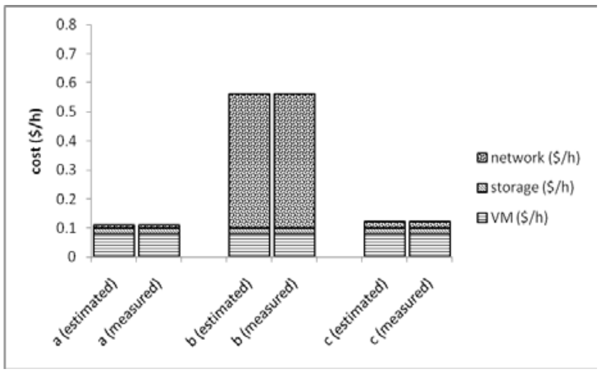

Figure 3: Estimated and measured costs for each workload executing on a small VM type instance with no SLOs.

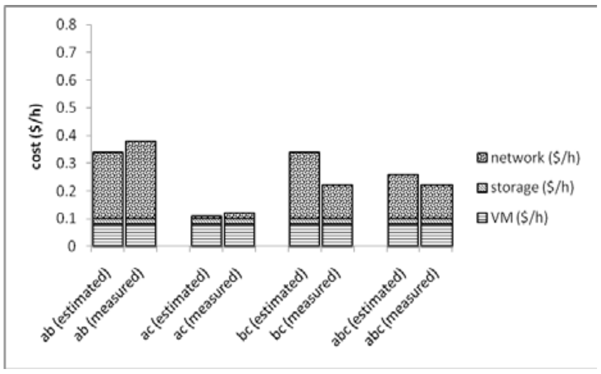

Figure 4: Estimated and measured costs for each workload combination executing on a small VM type instance with no SLOs.

5.3.3 SLA Penalties

Next we keep the workloads and the VM type fixed, but vary the SLOs. We choose small VM type instance to execute all workloads simultaneously (combination abc). We describe our rationale for SLOs' specification and penalties.

Table 4. SLOs for different requests.

Tenant	Request	threshold	penalty
c	Payment	50tps*	$0.10
b	Trade-update	0.04tps	$0.15
a	Q1	200s	$0.05

* transaction per second (tps)

Missing payments lead to monetary losses. Therefore, we associate a high penalty ($0.10) with the threshold value of 50tps as shown in Table 4. We also place SLOs on write-heavy and read-only workloads for trade-update and Q1 requests, respectively. The trade-update in a real stock market has a low frequency of arrival, but large monetary stakes. Therefore, we place an SLO on trade-update in the write-heavy workload by associating a threshold of 0.04tps but with a harsh penalty ($0.15). Meanwhile, Q1 is an analytical query with an expectation of larger execution time. Therefore, we associate a threshold of 200s with a minor penalty ($0.05) to discourage the embarrassingly large execution time. We compare the estimated and the measured costs in Figure 5.

The average error in estimating cost is about $0.04, which is about 12% of the total measured cost on average. After inspecting the evaluation results above, we note that the cost errors are usually in estimating network costs.

[1] %cost error = (predicted – measured)/measured

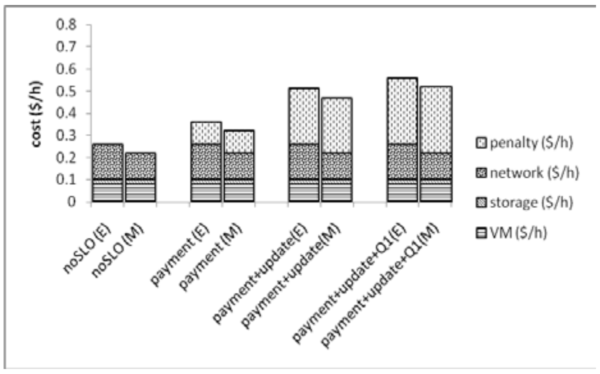

Figure 5: Estimated (E) and measured (M) costs for all workloads executing simultaneously on a small instance with varying SLOs.

We see considerable penalty costs, in the case of payment+update+Q1 SLOs, due to lack of resources to avoid violations. This shows under-provisioning. We compare the costs of simultaneously executing all workloads on each VM type in Figure 6. We see that the overall cost reduces slightly for large VM type due to reduced penalties but higher VM costs. There are no penalties when the workloads execute on the xlarge VM type, but the overall cost is higher than any other resource configuration. In the case of the xlarge VM type, we observe that the penalties have been replaced by higher VM costs. This is an example of the trade-off between penalties and resource costs. The resource configuration with large VM type has the minimal dollar-cost, and therefore is the optimal configuration for the executing all the workloads given payment+update+Q1 SLOs. The average error in estimating costs varies by about $0.01, which is about 2% of the total measured cost on average.

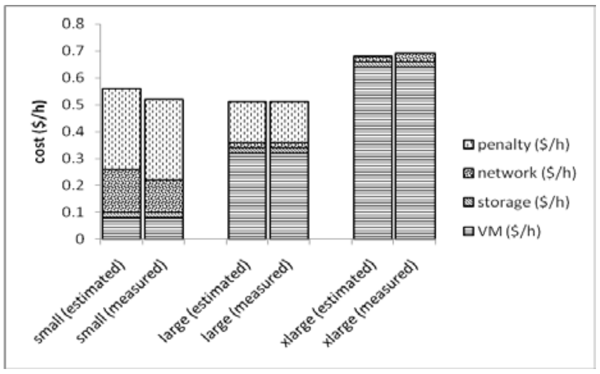

Figure 6: Estimated and measured costs for all workloads executing simultaneously with payment+update+Q1 SLOs on each VM type.

5.3.4 SUMMARY

In all the experiments, we observe low storage costs, especially compared to network and VM costs. This is because monthly storage costs is already low, and prorating it gives even lower hourly cost, which is then rounded up to the next cent. This relatively lower storage cost is in-line with the widening cost- value gap between storage and other computational resources including network and processors.

The evaluation cases considered gauge our cost model for each user-controllable variable. Admittedly, they are not exhaustive. Further, our workloads are distorted versions of the realistic workloads presented by Cooper et al. [30]. Nonetheless, we argue that they suffice for the evaluation. Also, we do not vary the instances of the same workload. This is unnecessary because our workloads do not have a fixed number of request instances in a request mix, and we scale a workload according to the optimal MPL value of a VM instance. Finally, our workload combination contains at most eight request types. This is reasonable since TPC-C and TPC-E benchmarks have five and ten transactions, respectively, although TPC-H has 22 queries. We believe a realistic DBMS is rarely a read-only or a write-only service. It usually serves a combination of transactional and analytical workloads [31].

Also, we do not consider large number of VM instances and tenant replica in evaluating our cost model. This is because the VM costs can be determined accurately using the published unit costs. We validated our method of estimating storage costs by inspecting Amazon’s Account Activity for a number of months.

6. CONCLUSIONS AND FUTURE WORK

We formulate the constructs for modeling the cost of workload execution in a public cloud. We present a cost model for workload execution, and evaluate it in the Amazon cloud. Our cost model is workload aware and provides cost at the granularity of an hour. More importantly, we explore methods for building and instantiating a cost model for workload execution in IaaS-based clouds. These methods are relevant for other IaaS GoGrid [13] or RackSpace [12]. We believe that our cost model provides a basis for modeling dollar-cost for executing any workload type in the Amazon cloud. We anticipate that the users considering clouds for executing their application would find the cost model useful.

Our cost model provides an hourly cost of workload execution, and assumes that the data already exists in the cloud. The experimental evaluation shows that our cost model is a suitable tool for estimating the cost of workload execution for the pay-as-go-scheme in the Amazon clouds.

We vary the use-cases in the user-controllable variables: (a) workloads, (b) VM types and the (c) SLOs’ specifications. Our evaluation workloads consist of analytical, transactional and mixed types. We consider different workload combinations on different VM types. We also specify SLOs on the transactions and the queries belonging to different tenants. The SLOs vary in their threshold and penalty values. The *absolute* average error in estimating configuration costs across all experiments is 6.28%, which is about $0.02 of the total measured cost of the configurations on average. With the scarcity of training samples, we are unable to verify the distribution of results and resort to using average as the aggregation method. Therefore, these results must be taken with caution.

Our cost model estimates the network cost using a simple average. We believe it is likely to become less effective when the optimal MPL value of a VM instance is not divided equally amongst the workloads executing on a VM instance. Further, all the request types present in a workload have equal presence. Changing the presence may also have impact on the estimation method. We intend to explore the suitability of simple average under unequal MPL share and/or unequal presence.

The current cost model, while adequate for workload execution in a single zone, needs to be expanded in order to deal with any inter-zone and inter-region communication costs. Also we do not address the cost of maintaining consistency between replicas, leaving it as future work.

7. ACKNOWLEDGMENTS

We acknowledge research support from National Science and Engineering Research Council of Canada (NSERC), Amazon

AWS, MEDIANET (Comunidad de Madrid S2009/TIC-1468), ServiceCloud (MINECO TIN2012-31518) and 4CaaSt (European Commission's IST priority of the 7th Framework Programme under contract number 258862).

8. REFERENCES

[1] R. Mian and P. Martin, "Executing data-intensive workloads in a Cloud," *CCGrid Doctoral Symposium 2012 in conjuction with 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 758-763, Ottawa, ON, Canada.

[2] Amazon, "Elastic Compute Cloud (EC2)," [Online] Retrieved on 17th Dec, 2009; Available: http://aws.amazon.com/ec2/.

[3] J.L. Vazquez-Poletti, G. Barderas, I.M. Llorente and P. Romero, "A Model for Efficient Onboard Actualization of an Instrumental Cyclogram for the Mars MetNet Mission on a Public Cloud Infrastructure," *PARA2010: State of the Art in Scientific and Parallel Computing, Lecture Notes in Computer Science (LNCS)*, vol. 7133, 2010, pp. 33-42.

[4] K. Tsakalozos, et al., "Flexible use of cloud resources through profit maximization and price discrimination," *27th International Conference on Data Engineering (ICDE)*, IEEE, 2011, pp. 75-86, Hannover, Germany.

[5] T. Bicer, D. Chiu and G. Agrawal, "Time and Cost Sensitive Data-Intensive Computing on Hybrid Clouds," *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 636-643, Ottawa, ON, Canada.

[6] B. Sharma, R.K. Thulasiram, P. Thulasiraman, S.K. Garg and R. Buyya, "Pricing Cloud Compute Commodities: A Novel Financial Economic Model," *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 451-457, Ottawa, ON, Canada.

[7] W. Li, Y. Yang, J. Chen and D. Yuan, "A cost-effective mechanism for Cloud data reliability management based on proactive replica checking," *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2012, pp. 564-571, Ottawa, ON, Canada.

[8] M. de Assunção, A. di Costanzo and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," *Cluster Computing*, vol. 13, no. 3, 2010, pp. 335-347.

[9] L. Du, "Pricing and Resource allocation in a Cloud Computing Market," *Workshop on Cloud Computing Optimization (CCOPT 2012) in conjunction with 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid).* 2012, pp. 817-822, Ottawa, ON, Canada.

[10] Amazon, "Simple Monthly Calculator," [Online] Retrieved on 9th Jul, 2011; Available: http://calculator.s3.amazonaws.com/calc5.html.

[11] M. Armbrust, et al., "Above the Clouds: A Berkeley View of Cloud Computing," *Technical Report#: UCB/EECS-2009-28*, University of California at Berkeley, 2009 [Online] Retrieved on 13th Feb, 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[12] RackSpace, "Cloud hosting products-using the power of cloud computing by rackspace," [Online] Retrieved on 9th Aug, 2012; Available: http://www.rackspacecloud.com/.

[13] GoGrid, "Cloud hosting: Instant windows and linux cloud servers," [Online] Retrieved on 9th Aug, 2012; Available: http://www.gogrid.com/.

[14] Amazon, "EC2 Instance Types," [Online] Retrieved on 27th April, 2011; Available: http://aws.amazon.com/ec2/instance-types/.

[15] Amazon, "EC2 Instance Store," [Online] Retrieved on 27th Jul, 2013; Available: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html.

[16] Amazon, "Simple Storage Service (S3)," [Online] Retrieved on 2nd Dec, 2010; Available: http://aws.amazon.com/s3/.

[17] Amazon, "DynamoDB," [Online] Retrieved on 31st Jan, 2012; Available: http://aws.amazon.com/dynamodb/.

[18] Amazon, "Elastic Block Store (EBS)," [Online] Retrieved on 28th Aug, 2010; Available: http://aws.amazon.com/ebs/.

[19] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, 2010, pp. 35-40.

[20] K. Chodorow, *MongoDB: the definitive guide*, O'Reilly, 2013.

[21] M.R. Palankar, A. Iamnitchi, M. Ripeanu and S. Garfinkel, "Amazon S3 for science grids: a viable solution?," *Proceedings of the 2008 international workshop on Data-aware distributed computing*, ACM, 2008, pp. 55-64, Boston, MA, USA.

[22] A. Bergen, Y. Coady and R. McGeer, "Client bandwidth: The forgotten metric of online storage providers," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*, IEEE, 2011, pp. 543-548, Victoria, BC, Canada.

[23] O.A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, 2011, pp. 304-311, Athens, Greece.

[24] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," *10th IEEE/ACM International Conference on Grid Computing*, 2009, pp. 17-25, Banff, AB, Canada.

[25] TPC, "Transaction Processing and Analytical Database Benchmarks," [Online] Retrieved on 26th Jun, 2011; Available: http://www.tpc.org/information/benchmarks.asp.

[26] J. Schad, J. Dittrich and J.-A. Quiane-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proceedings of VLDB Endowment* vol. 3, no. 1-2, 2010, pp. 460-471.

[27] Amazon, "$0.10 per GB-month of provisioned storage," [Online] Retrieved on Aug 12, 2010; Available: http://forums.aws.amazon.com/message.jspa?messageID=190013.

[28] Amazon, "Undercharged for allocated ebs volume?," [Online] Retrieved on 6th May, 2012; Available: http://forums.aws.amazon.com/thread.jspa?messageID=285885&#285885.

[29] Amazon, "Account Activity," [Online] Retrieved on 27th Apr, 2011; Available: http://aws.amazon.com/account/.

[30] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears, "Benchmarking cloud serving systems with YCSB," *1st ACM Symposium on Cloud Computing (SoCC)*, ACM, 2010, pp. 143-154, Indianapolis, IN, USA.

[31] G. Paulley, "DBMS applications and workloads (Personal Communications)," Director, Engineering, *Sybase iAnywhere*, 2011.