A Framework for Analyzing Monetary Cost of Database Systems in the Cloud

Changbing Chen and Bingsheng He

School of Computer Engineering
Nanyang Technological University, Singapore

Abstract. In this paper, we propose to develop a framework to analyze the monetary cost of running database systems in the public cloud. The framework offers guidelines and methodologies in analyzing and estimating monetary cost of database systems. It consists of multiple components including categorizing database performance tuning knobs, benchmarking the price/performance of computation resources offered by the cloud provider, and building a monetary cost model. As a case study of our proposed framework, we conduct an in-depth study on two popular open-source database systems with respect to two cloud providers. We find that evaluating a query spans a wide range of monetary costs (with a difference up to 91%), and the experimental results demonstrate the accuracy of our monetary cost estimation.

Keywords: Database Systems, Cloud Computing, Performance, Monetary Cost, Framework.

1 Introduction

In the era of cloud computing, computation can be traded as a utility sold and bought according to predefined price schemes. Under *pay-as-you-go* charging, the monetary cost of running a database system in the cloud becomes an explicitly measurable metric. The *pay-as-you-go* feature is a clear distinction from previous price-oriented studies, for example, PennySort [3] and TPC-H including costs on hardware, software and shipping. Like the performance (in terms of throughput or response time) that has long been the core metric for system design and optimizations, the monetary cost has become an important metric for system design and optimizations [13,8].

To understand and optimize the monetary cost of running database systems in the cloud, we are particularly interested in the following questions:

- Where does the money go? We hope to identify the major cost component(s) in the total monetary cost.
- What is the correlation between performance and monetary cost of running database systems in the cloud?
- Current price schemes are mostly based on resource consumptions, with differential prices on virtual machine types. How do price schemes affect the system optimizations for monetary cost in the future cloud environment?
- What are the opportunities to reduce the monetary cost given certain workloads?

J. Wang et al. (Eds.): WAIM 2013, LNCS 7923, pp. 118–129, 2013.

It is a non-trivial task to answer the above questions. The monetary cost of running database systems in the cloud involves various complicated and often intertwined factors involving cloud providers (including price definitions as well as the virtual machine performance), and users (including how database systems are tuned and optimized). All these factors are able to significantly affect the monetary cost. Specifically, providers offer virtual machines with different capabilities at different prices. It is not clear about the monetary cost of running the same system on these offerings. Moreover, there is not yet a standard on the price scheme among different cloud providers. Clearly, different price schemes directly result in different monetary costs, leading to the differences in monetary cost. Even within a single price scheme, different virtual machines are charged at different prices; resource components (such as I/O, CPU and memory) can have different prices at different time periods (such as option pricing in Amazon). There have been only a few preliminary studies in understanding and optimizing the monetary cost of running database systems in the cloud [13,9]. Also, few studies systematically explore the interplay among database systems, prices and cloud offerings.

In this paper, we develop a framework to guide the study on the interplay among performance tunings, price structures and cloud offerings. In the framework, we develop methodologies in the following components of analyzing the monetary cost: 1) defining the space of prices under study, 2) assembling (micro) benchmarks for assessing the price and performance of resource components of virtual machines, 3) categorizing the tuning knobs within database systems with respect to the relationship between performance and monetary cost, and 4) developing a monetary cost model to estimate the monetary cost of query processing. Those components are designed as guidelines for analyzing the monetary cost of database systems in the cloud.

We evaluate our framework by running two popular open-source DBMSs (PostgreSQL and MySQL) on two different cloud providers, Amazon and Rackspace. Through the extensive study, we find that virtual machine selections and price-aware optimizations are key factors for reducing the monetary cost. Additionally, the experiments show that our monetary cost model accurately predicts the monetary cost with different virtual machine offerings and prices. That demonstrates the effectiveness of the proposed framework.

Organization. The remainder of this paper is organized as follows. We review the related work in Section 2. Section 3 presents the framework design, followed by the experimental results in Section 4. Finally, we conclude this paper in Section 5.

2 Related Work

There have been a few studies on evaluating and optimizing user expenses in the cloud. Economics, particularly prices, have significant impact on cloud system designs [13,5]. Palankar et al. [9] studied the cost, availability and performance on Amazon S3 services, in the context of data intensive scientific applications. Kossmann et al. [8] studied the performance and cost of transactional workloads for different cloud providers. Tak et al. [11] studied different deployment choices for transactional workloads. Kllapi et al. [7] proposed optimizations for virtual machine selections to minimize the monetary cost of data flow programs. Assuncao et al. [2] evaluated the cost performance

of different scheduling strategies to combine a private (self-owned) cloud with public clouds. Compared with existing work, this study focuses on the interplay among database systems, prices and cloud offerings, and develops a monetary cost model for monetary cost optimizations. A number of studies [13,10,4] have been conducted to understand the performance variance on Amazon. We observed similar variance in the monetary cost, and our study covers the offerings from different cloud providers, and more virtual machine types on Amazon.

3 Framework Design

In order to analyze the monetary cost of database systems in the cloud, we develop a framework to guide the analysis of different aspects related to monetary cost. Particularly, the framework consists of the following components: 1) defining the space of prices, 2) assembling a series of micro benchmarks for assessing the price and performance of resource components of virtual machines, 3) categorizing the tuning knobs within database systems with respect to the relationship between performance and monetary cost, and 4) developing a monetary cost model by extending the performance model within database system. In the remainder of this section, we present the detailed design of each component.

3.1 Space of Prices

We observe *spatial* and *temporal* features of price definitions in the cloud. The spatial feature means different price definitions for different virtual machine instances and different cloud providers. The temporal feature means the temporally varying prices for the same virtual machine type, such as option pricing on Amazon. One example of spatial features is differential pricing for virtual machines in different regions of the world. Even for the same virtual machine, the reservation price is lower than that of on-demand virtual machines. Option pricing in Amazon is a temporal pricing feature. When the option price determined by supply/demand is lower than the user bid price, the virtual machine is granted to the user at the option price.

For the fair comparison on different virtual machines, we consider the offerings with the same operating system (Linux in our study). Moreover, the absolute prices are not a direct motivation for investigating the monetary cost of system optimizations. The interplay between the database system optimization and the price is on the *relative* prices among different resource consumptions, the *relative* prices for the same resource within the same provider or from different cloud providers and the *relative* prices on different temporal periods. We intentionally vary the price structures for sensitivity evaluations on the monetary cost.

3.2 Micro Benchmarks

We assemble the following micro benchmarks to measure the monetary cost of CPU and memory systems. Particularly, we use the Unix Benchmark Utility (Ubench) [12] to measure CPU capability and main memory performance. For I/O, we use the IOzone benchmark [6] to measure the sequential and random read performance on the persistent storage.

3.3 Categorizing Tuning Knobs

Many performance tuning knobs could have been added to DBMSs for performance evaluation. They cover different aspects of data management tasks, from query planning and execution to system wide configurations. Our studies identify three categories of tuning knobs, according to resource consumption based price schemes: (Category **A**) the knobs directly reduce the consumption of a resource component without increasing the consumption of other resource components (e.g., the buffer size of the DBMS), and (Category **B**) the knobs have the tradeoff among multiple resource components: reducing the consumption of a resource component but increasing the consumption of the other resource components. Tuning knobs in Category B require careful investigations on existing tradeoffs, which exist in various query optimizations such as access methods (e.g., sequential scans vs. index scans), and compression techniques. Beyond the existing tuning knobs for a specific virtual machine, we have other tuning knobs in the cloud, i.e., selecting different cloud providers and different virtual machine types from a specific provider (Category **C**). In this study, we investigate some typical tuning knobs that are common for PostgreSQL and MySQL, as shown in Table 1.

Table 1. Example tunings in each category

Category	Tunings
A	• tuning the buffer size;
B	• database compressions; • auxiliary structure (e.g., index and materialized view);
C	• selections of different virtual machines from the same provider; • selections of suitable cloud providers;

3.4 Monetary Cost Model

Since existing cost models in DBMSs are limited to query execution time, we develop a monetary cost model to facilitate virtual machine selections and price-aware tuning knobs. A natural direction is to adapt the existing cost model in DBMS with the awareness of virtual machines and prices. We have implemented the monetary cost model in both PostgreSQL and MySQL. Since their adaptation and experimental results are similar, we present the adaptation in details with PostgreSQL as an example.

We develop the monetary cost model in two steps, introducing the awareness of virtual machine performance differences and prices into the existing timing cost model of the DBMS.

First, we introduce the awareness of the performance differences of virtual machine offerings, and obtain an enhanced timing cost model. As demonstrated in Figure 2, different virtual machine types have quite different CPU, memory and I/O performance. The existing timing cost model of the DBMS uses some constants as the unit cost for calculating the query execution time. Figure 1 shows a fraction of the source code for the cost estimation of a sequential scan on a relation in PostgreSQL. Two constant parameters, “spc_seq_page_cost” and “cpu_tuple_cost”, are used to represent the I/O

and CPU unit costs per tuple in a sequential scan. We use some simple experiments to calibrate those unit costs in the target virtual machine types. Next, the calibrated values are plugged into the timing cost model. Thus, the timing cost model is able to predict the timing cost on different virtual machine types.

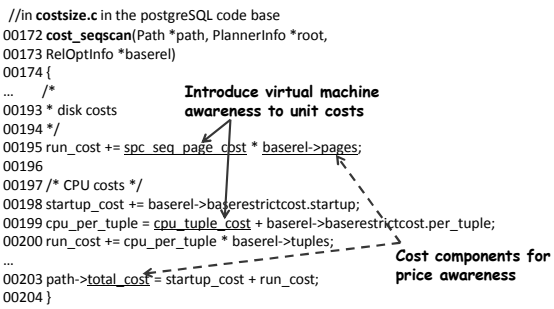


Fig. 1. An illustration of building monetary cost estimation: estimations for a sequential scan on a relation in PostgreSQL

Second, we introduce the awareness of price structures into the enhanced timing cost model. The enhanced timing cost model is able to predict the query execution time (denoted as T) on a certain virtual machine type. Moreover, from the timing cost estimation, we can obtain the estimated number of I/O operations incurred during query execution (denoted as $\#IO$). Figure 1 illustrates that "baserel$->$pages" and "total_cost" are used as estimations in PostgreSQL for $\#IO$ and T, respectively. Therefore, we derive the monetary cost model in Eq. 1, by summing up the execution, storage and I/O cost components. The price structure ($p_{execution}$, $p_{storage}$ and p_{io}) is defined as the prices for virtual machine execution, storage and I/O operations, respectively. The database storage size, S, is obtained from the database catalog.

$$MC = p_{execution} \times T + p_{storage} \times S \times T + p_{io} \times \#IO \quad (1)$$

4 Case Studies

In this section, we present case studies of the proposed framework by running PostgreSQL and MySQL on Amazon and Rackspace.

4.1 Experimental Setup

SUT (System Under Test). The SUT (PostgreSQL or MySQL) runs on a virtual machine in a public cloud. The database data are stored in the persistent storage offered in the cloud, e.g., EBS/RAID 0 in Amazon and the persistent storage of *cloud server* in Rackspace. Therefore, database data is persistent, even after the virtual machine is turned off.

Table 2. Instances on Amazon used in our experiments (Price: $ per hour, July 15, 2012, Linux, US-N. Virginia)

Name	Disk (GB)	CPU (CU)	IO	RAM (GB)	Price	Option price
A1	0	≤ 2	Low	0.6	0.02	0.008
A2	160	$1\times$ 1	Moderate	1.7	0.085	0.035
A3	350	$2\times$ 2.5	Moderate	1.7	0.17	0.06
A4	850	$2\times$ 2	Moderate	7.5	0.34	0.15
A5	1,690	$8\times$ 2	High	15	0.68	0.24

Table 3. Linux virtual machines on Rackspace used in our experiments (July 15, 2012)

Name	Disk (GB)	RAM (GB)	Price ($ per hour)
R1	80	2	0.12
R2	160	4	0.24
R3	320	8	0.48
R4	620	15.5	0.96

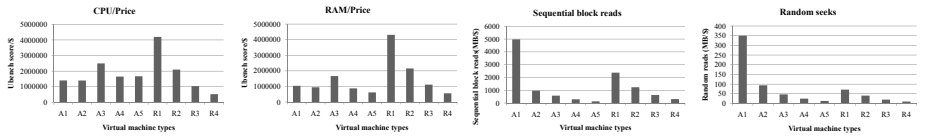
TPC-H Setup. We chose TPC-H as the benchmark workloads, since these queries represent commonly used data warehousing workloads. Moreover, they vary in complexity, which exercises different components including CPU and I/O in the virtual machine. We evaluate TPC-H with different scale factors, one and ten. The default scale factor is 10. In the experiment, we exclude TPC-H queries Q8, Q20 and Q21, since each of these queries ran longer than 3 hours.

We run TPC-H benchmark queries on two popular open-source data management systems, PostgreSQL 8.4 and MySQL 5.1. The page size of both systems is set to be 8KB. The buffer size is manually tuned for the best performance on different virtual machines. We examine the monetary cost of evaluating individual queries as well as multiple queries. In the evaluation of multiple queries, we evaluate 30 queries randomly selected from the TPC-H benchmark. Since we mainly focus on the monetary cost incurred by the resource consumption, we exclude installation costs on those systems.

Database Compressions. MySQL supports database compressions, and allows us to investigate database compressions as a tuning of Category B. We use *myisampack* to (de)compress the database. The total storage sizes of tables and their indexes with scale factor of ten are 11.4GB and 17.4GB with and without database compressions, respectively. With compression, the database storage size reduces 35%.

We conducted the experiments on virtual machines offered by Amazon and Rackspace. The operating system is Ubuntu Linux 10.04. The file system is *ext3*. We consider the five virtual machine types in Amazon with different CPU and I/O capabilities. Table 2 summarizes their basic properties listed in Amazon web site [1]. Virtual machine types A1–A5 correspond to *t1.micro*, *m1.small*, *c1.medium*, *m1.large* and *m1.xlarge* in Amazon's definition. The CPU capability is given in the form of ($\#core \times \#CUPerCore$).

We choose the virtual machine types R1–R4 on Rackspace, as shown in Table 3. Rackspace provisions the storage almost proportional to the amount of RAM.



(a) (b) (c) (d)

Fig. 2. Monetary efficiency of CPU and memory systems in different virtual machines in Amazon and Rackspace

While the result of a single run is not representative due to the cloud system dynamics, the results of many runs can capture the stability, i.e., forming a band in the measurement [10]. Thus, the average value of multiple runs tends to represent the performance in a long running scenario. In this study, we run each experiment on five virtual machine instances with ten times each, and report the average for the measurements.

4.2 Micro Benchmarks

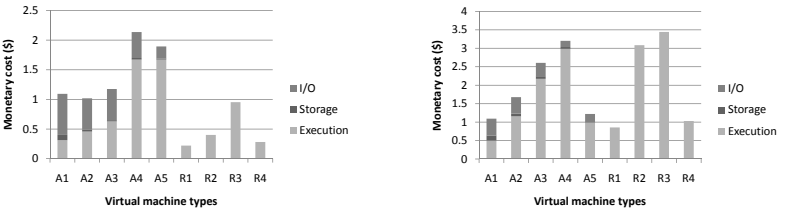
Figure 2 shows the monetary cost obtained from the micro benchmark results on different virtual machine instances. The block size of IOzone is set to the page size of data management systems (8KB). Comparing the relative monetary cost of different system components among the virtual machines, we find that the relative monetary cost of the CPU is similar to that of RAM; the relative monetary cost of the sequential block accesses is similar to that of random block accesses.

We observe significant differences among different virtual machines. On Rackspace, R1 achieves the lowest monetary cost on all the four micro benchmarks. This results in R1 being the most monetary efficient virtual machine on Rackspace, as we will see in Figure 3. Nevertheless, R1 does not necessarily achieve the lowest monetary cost for all workloads, due to its relatively small main memory capacity as well as storage capacity. On Amazon, there is not a specific virtual machine types dominating the four measured metrics.

4.3 Where Does the Money Go?

Overall, we observed similar monetary cost breakdowns on MySQL to those in PostgreSQL.

Multiple Queries. Figure 3 shows the monetary cost breakdown for running 30 queries in PostgreSQL and MySQL. Since Rackspace does not charge on I/O or storage, the monetary cost includes the execution cost only. In contrast, we observed significant differences in the breakdown among different virtual machines on Amazon. As the virtual machine capability increases, the total execution time decreases, and the amortized storage cost decreases. However, due to the increased price, the execution cost may increase or decrease. As for the I/O cost, as the virtual machine capability increases, the amount of main memory increases and the number of I/O accesses reduces, therefore reducing the I/O access cost. For virtual machines A1–A5, the storage cost is insignificant. The I/O cost and execution costs are two important components in the monetary



(a) PostgreSQL (b) MySQL

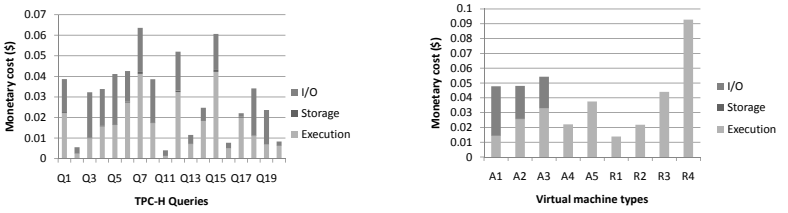
Fig. 3. The monetary cost breakdown of running 30 queries of PostgreSQL and MySQL on Amazon and Rackspace

cost of A1–A5. On the virtual machine with higher prices, the execution cost is more significant and the I/O access cost is less significant.

The time breakdown clearly shows that I/O and execution costs are the most significant components in the monetary cost. We could reduce I/O accesses and/or improve the query processing performance to reduce the monetary cost. They are usually consistent on data management systems, i.e., reducing I/O accesses usually results in performance improvement.

Individual Queries. Figure 4(a) shows the breakdown on monetary cost of PostgreSQL on running each query. Note, we restart the database system before measuring the cost of each query. The monetary cost breakdown of individual queries is similar to those in long-running scenario. The storage cost is insignificant, and the execution cost and the I/O cost are two significant components. Between these two components, the execution cost is the most significant component (more than 50% of the total monetary cost) on Queries Q1, Q6–7, Q12–18 and Q22. The I/O cost is dominated in other queries. Compared with the multi-query scenario, the I/O cost is more significant, because we flush the buffer cache before running individual queries. For other virtual machine types, we also observed similar results.

Different Scale Factors. Figure 4(b) shows the monetary cost breakdown of PostgreSQL running 30 queries with the scale factor one. We observe similar trend as the scale factor of ten. The execution cost is dominated in all virtual machine types, and the



(a) (b)

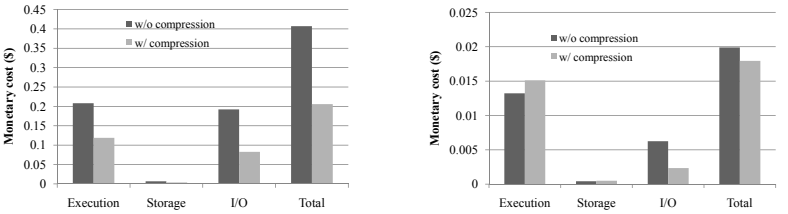
Fig. 4. The monetary cost breakdown: (a) running individual queries on A3; (b) running 30 queries of PostgreSQL with scale factor of one

I/O cost is significant for A1–A3. Since the entire database fits into the main memory of A4 and A5, the I/O access cost is small on those two virtual machine types.

4.4 Performance vs. Monetary Cost

Through a series of experiments on the three categories of tunings, we find that different categories of tuning have significant impact between performance and monetary cost of query evaluations. Category A improves both performance and monetary cost. Category B generally improves both performance and monetary cost with some exceptions, which require special care for the tradeoff between performance and monetary cost. Lastly, Category C usually has conflicts in optimizing performance and optimizing monetary cost. From the micro benchmarks, we have observed that different virtual machines from the same provider or from different cloud providers have different monetary cost on CPU and memory systems. These differences result in the differences in the monetary cost of running TPC-H workloads. Thus, we focus on the results for Categories B and C only.

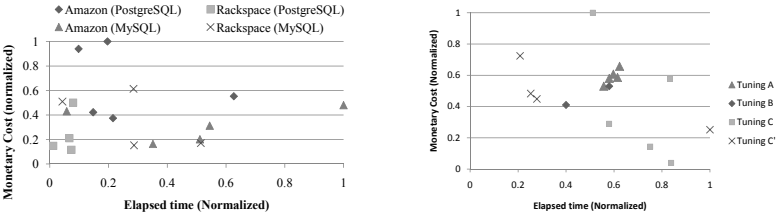
Tunings in Category B. Compressions. Figure 5 shows the performance and the monetary cost for the simple selection query with sequential scans and TPC-H Q5 with and without compressions on A3. Q5 is a complex five-way join query with sorting and aggregation. Database compressions reduce the I/O cost for all queries. With compression, the execution cost of TPC-H Q5 reduces, and the execution cost for the selection query increases, due to the decompression cost. Overall, database compressions improve monetary efficiency for both queries, but degrade the performance of the selection query.



(a) TPC-H Q5 (b) Selection query without index

Fig. 5. Monetary costs of evaluating TPC-H Q5 and the selection query with and without compression on A3

Tunings in Category C. Figure 6(a) shows the performance and monetary efficiency of running 30 queries on PostgreSQL and MySQL. Data management systems have been optimized for the performance with various tunings in Categories A and B. The performance and monetary efficiency do not have a clear correlation between each other. The best performing virtual machine type is not the one with the best monetary cost. Moreover, the best performing type varies with different systems.



(a) (b)

Fig. 6. Performance and monetary cost: (a) on different virtual machines in Amazon and Rackspace, (b) by applying the three categories of tunings individually. Tunings A and B are performed on A3 on Amazon.

Put It All Together. Figure 6(b) illustrates monetary cost and performance (i.e., the elapsed time) for Q5 in TPC-H workload. We have observed similar results in other TPC-H queries. Both monetary cost and elapsed time results are normalized to their corresponding maximum value. Tunings A, B and C(C') belong to Categories A, B and C, respectively. Tuning A is to tune the buffer size within the same virtual machine (i.e., with RAM of 1GB, 2GB, 4GB and 6GB). Tuning B performs Q5 with and without database compressions. Tunings C and C' are to run the query on different virtual machine types offered by Amazon and Rackspace, respectively.

Overall, there is not a clear correlation between performance and monetary cost with all the tunings considered. The point with best performance is not the point with the highest monetary cost, and vice versa. Moreover, we have observed the three cases for the correlation between performance and monetary cost. As a result, the monetary cost varies significantly (the difference is as much as 91% for TPC-H Q5), even with similar performance.

4.5 Impact of Price Structures

Different Charging Methods. Figure 7(a) re-plots the monetary cost of running 30 queries in Amazon with charging according to the RAM hour. We use Rackspace's price, $0.06 per GB per hour. Clearly, different charging methods affect the monetary cost, since workloads have different consumptions on the resource. With charging on the RAM hour, A3 achieves the best monetary cost. One possible reason is due to the high monetary efficiency of RAM in A3, as shown in our micro benchmarks.

Different Prices among Virtual Machines. On Amazon, the price of execution on Ai is twice as that of A$(i-1)$ ($i \geq 3$). A similar tiering price ratio r exists in Rackspace. Figure 7(b) re-plots of the monetary cost of running 30 queries on A2–A5 in Amazon, varying the ratio, r, in the tiering price. We fixed the price of A5, and vary the ratio of r. As the r increases, the price of A2 decreases. Thus, the monetary cost of A2–A4 increases, and A5 has the same monetary efficiency. The main observation is that the comparison of monetary efficiency among A2–A4 varies with the ratio. For example, A2 has the best monetary cost when $r = 2$, whereas A3 has the best monetary efficiency when $r = 1.5$. Thus, the tiered prices affect the choice on the virtual machine with the best monetary cost.

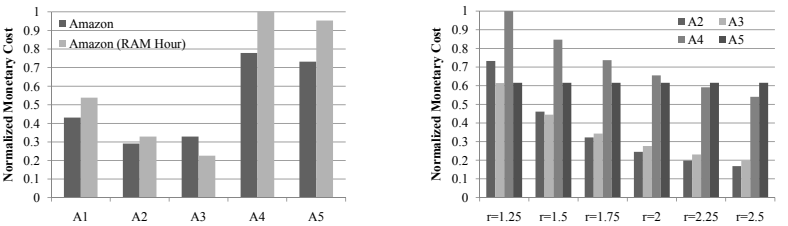


Fig. 7. Normalized monetary cost on A1–A5: (a) with different charging methods, (b) with different tiring price ratios

4.6 Cost Model Evaluations

We observed similar results on PostgreSQL and MySQL, and thus present results for PostgreSQL only. Figure 8(a) shows the monetary cost prediction of PostgreSQL on running each query on A3. The real and estimated monetary costs are normalized to their corresponding maximum values. Our monetary cost model can accurately predict the monetary cost of individual TPC-H queries. Thus, our monetary cost estimation is applicable to different queries.

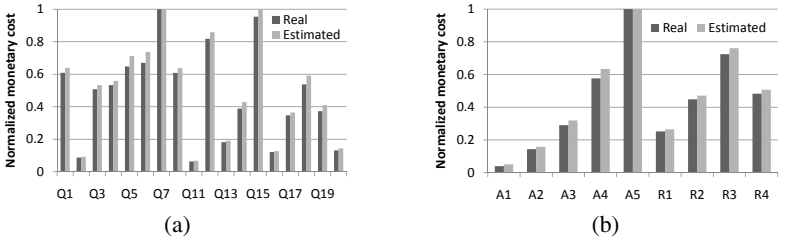


Fig. 8. Monetary cost measurement and estimation: (a) running individual TPC-H queries of PostgreSQL on A3; (b) running TPC-H Q5 of PostgreSQL on Amazon and Rackspace

Figure 8(b) shows the monetary cost prediction of running TPC-H Q5 in PostgreSQL on A1–A5 and R1–R4. Our monetary cost model is able to achieve a good prediction on the trend of the real monetary cost, regardless of virtual machine types.

4.7 Summary

The evaluation of our framework reveals the correlation between performance and monetary cost depends on workloads, price schemes and virtual machine types. We have the following two implications on DBMSs.

First, on the same virtual machine type with fixed prices, the most monetary efficient operating points are usually the best performing for different execution strategies. Our studies confirm that the traditional performance-oriented optimizations continue to

be effective on the same virtual machine. We do see exceptions caused by tunings of Category B on Amazon. One example exception is database compression.

Second, when different virtual machine types and different pricing features are considered, the most monetary efficient operating points are usually *not* the best performing. Our cost model can accurately predict the monetary cost comparison among different virtual machine types.

5 Conclusions

This paper proposes a framework for analyzing the monetary cost of running database systems in the cloud. We evaluate the effectiveness of the framework with database warehousing workloads with two popular open-source DBMSs, PostgreSQL and MySQL, on two cloud providers, Amazon and Rackspace. We find that monetary cost of the same system varies significantly on different virtual machine types and different price definitions. On a specific virtual machine type, the best performing configuration is usually the most monetary efficient. We further develop a monetary cost model with the awareness of virtual machine selections and prices. Our experiments demonstrate that the monetary cost model accurately predicts the monetary cost of query evaluations with different virtual machine offerings and prices.

References

1. http://aws.amazon.com/ec2/instance-types/
2. Assunção, M.D., di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: HPDC (2009)
3. Coates, J., Gray, J., Nyberg, C.: Performance/price soft and pennysort. Tech. Rep. MSR-TR-98-45, Microsoft Research (1998)
4. Gong, Y., He, B., Zhong, J.: CMPI: Network performance aware MPI in the cloud. IEEE TPDS (2013)
5. Ibrahim, S., He, B., Jin, H.: Towards pay-as-you-consume cloud computing. In: IEEE SCC 2011, pp. 370–377 (2011)
6. IOzone: http://www.iozone.org/
7. Kllapi, H., Sitaridi, E., Tsangaris, M.M., Ioannidis, Y.: Schedule optimization for data processing flows on the cloud. In: SIGMOD (2011)
8. Kossmann, D., Kraska, T., Loesing, S.: An evaluation of alternative architectures for transaction processing in the cloud. In: SIGMOD (2010)
9. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon S3 for science grids: a viable solution? In: DADC (2008)
10. Schad, J., Dittrich, J., Quiane-Ruiz, J.-A.: Runtime measurements in the cloud: Observing, analyzing, and reducing variance. In: PVLDB (2010)
11. Tak, B.C., Urgaonkar, B., Sivasubramaniam, A.: To move or not to move: The economics of cloud computing. In: HotCloud (2011)
12. Ubench: http://phystech.com/download/ubench.html
13. Wang, H., Jing, Q., Chen, R., He, B., Qian, Z., Zhou, L.: Distributed systems meet economics: Pricing in the cloud. In: HotCloud (2010)