


Article

A Cost Estimation Model for Cloud Services and Applying to PC Laboratory Platforms

KyungWoon Cho ¹ and Hyokyung Bahn ^{2,*} ¹ Embedded Software Research Center, Ewha University, Seoul 03760, Korea; cezanne@ewha.ac.kr² Department of Computer Engineering, Ewha University, Seoul 03760, Korea

* Correspondence: bahn@ewha.ac.kr; Tel.: +82-2-3277-2368

Received: 9 November 2019; Accepted: 4 January 2020; Published: 7 January 2020



Abstract: IaaS (Infrastructure as a Service) is a well-known computing service, which provides infrastructures over the cloud without owning real hardware resources. This is attractive as resources can be scaled up and down instantly according to the user's computing demands. Customers of such services would like to adjust the utilization policy promptly by considering the charge of the service, but an instantaneous response is not possible as it takes several hours or even a couple of days for cloud service providers to inform the billing information. In this article, we present an instant cost estimation model for estimating the cost of public cloud resources. Specifically, our model estimates the cost of IaaS by monitoring the usage of resources on behalf of virtual machine instances. As this is performed by generating a user-side metering daemon, it is very precise and thus similar to the resource usage evaluated by the cloud service provider. To validate our model, we run PC laboratory services for 50 students in two classes by making use of a public cloud during a semester. Experimental results show that the accuracy of our model is over 99.3% in comparison with the actual charge of the public cloud.

Keywords: IaaS cost estimation model; public cloud; cloud PC laboratory platform; CLABO

1. Introduction

IaaS (Infrastructure as a Service) is a new type of computing platform that provisions and manages hardware over the cloud [1]. As IaaS can quickly scale up and down resources according to user demands, it avoids the expense and complexity of buying and managing each user's own physical infrastructure. The billing of IaaS is determined based on the amount of resource usage, and thus IaaS is suitable for applications whose computing demands fluctuate dynamically [2]. Although this is the merit of IaaS, users cannot instantly recognize the cost of the resources used. Some IaaS consumers need to know the expected cost of the services as soon as possible and adjust their cloud utilization policies accordingly. However, cloud service providers (CSPs) do not inform the billing information instantly, making it difficult for users to coordinate their policies proactively [3]. In reality, evaluating the IaaS cost at the CSP level is a large-scale batch task, and thus it takes several hours or even a couple of days for IaaS users to be informed about the charge of the services.

To determine the IaaS cost, the usage of resources in the cloud should be measured and then evaluated [4,5]. The three representative resource types that need to be evaluated in IaaS are CPU, network, and storage. In the case of CPU, the basic price of a model is determined according to the computing power of the model, and the cost is charged based on the unit price of the model and the service time. In the case of network resources, the amount of data transferred over the network is evaluated and charged accordingly. In the case of storage resources, the combinations of the storage volumes used and the number of I/O operations are evaluated and charged. Although the cost of

IaaS is determined based on the aforementioned resource usage, this is totally the job of cloud service providers and users cannot examine the billing procedure; they just pay the cost requested by CSPs.

In this article, we present an IaaS cost estimation model for forecasting the costs of public cloud resources. Specifically, our model estimates the cost of IaaS instantly by monitoring the usage of resources on behalf of virtual machine instances. As this is performed by the user-side metering daemon we developed, it is very precise and thus similar to the resource usage evaluated at the CSP side. To validate our model, we run a PC laboratory service for 50 students in two classes by making use of a public cloud during a full semester. Experimental results show that the accuracy of our model is over 99.3% in comparison with the actual charge of the public cloud.

The remainder of this article is organized as follows. Section 2 briefly summarizes the related work of this study. Section 3 explains the cloud PC laboratory platform we developed for IaaS services. Section 4 describes the cost estimation model for IaaS in public cloud. In Section 5, we apply the cost estimation model to our cloud PC laboratory platform. Section 6 quantifies the accuracy of our model by applying it to two classes in the 2018 fall semester. Finally, Section 7 concludes this article.

2. Related Works

There have been a number of studies for the cost model of IaaS. Mazrekaj et al. perform the comparison studies of IaaS models and their pricing schemes from different CSPs [6]. Martens et al. present a formal mathematical cost model with the viewpoint of the TCO (total cost of ownership) in cloud computing [7]. They argue that the analysis of relevant cost types and factors in cloud computing is an important pillar of decision-making in cloud computing management. Belusso et al. propose a cost model based on a simple linear regression method [8].

Hedonic approaches for the IaaS cost model assume that the price should reflect embodied characteristics valued by some implicit nonfunctional features such as QoS (quality of service) [9,10]. Mitropoulou et al. show how to calculate and predict the cloud price accurately and how to avoid the price estimation bias [9].

Hinz et al. develop a cost model reflecting the usage of the processing power [11]. Aldossary et al. introduce a cloud system architecture and evaluate an energy-aware model that enables a fair attribution of a PM's energy consumption to homogeneous and heterogeneous VMs based on their utilization and size, which reflect the physical resource usage by each VM [12]. They also propose an energy-aware cost prediction framework that can predict the resource usage, power consumption, and estimate the total cost for the VMs during the operation of cloud services.

Sadeghi et al. propose a cost model considering hybrid cloud architectures [13]. They define cost factors for hybrid clouds and propose a resource allocation model that considers their cost model. Tang et al. try to solve the resource allocation problem with their cost model considering fairness [14].

Some research groups propose the price models for multi cloud environments. Wang et al. propose a taxonomy of pricing problems in the cloud resource and service markets, and demonstrate how relevant theories from the game theory, the control theory, and the optimization theory can be leveraged to address smart pricing problems for resource and service allocation of the cloud markets [15]. Xu et al. address the problem of multi-resource negotiation with the considerations of both the service-level agreement (SLA) and the cost efficiency [16].

Recently, IaaS cost estimation with the dynamic cost model have been proposed [17–21]. Agarwal et al. present a method for predicting spot prices using techniques of artificial neural network and show the experimental evaluation results on various instances of Amazon Elastic Compute Cloud (EC2) [17]. Meng et al. propose a parametric pricing approach in order to formulate pricing variables, which represent pricing factors and are calculated as well as a regression relation between the pricing variables and price [20]. They demonstrate the effectiveness of the proposed methodology with the real-world data of an organization in China. Their experimental results show that the proposed method achieves significant generalization performance with the best mean squared error (MSE) and reliable results in randomness of ensemble learning.

Most existing cost estimation models focus on the estimation of the cost that will be charged for a long-term period in the future based on the previous resource usage already reported. As this is the prediction of future resource usage, the model is complicated, and thus it is difficult to be implemented as an instant monitoring module. In contrast, our approach estimates the current charge of the cloud services promptly by injecting the estimation module in the user-side metering daemon, which incurs minimal overhead. Thus, the main focus of our design is in the minimization of the real-time estimation overhead by simplifying the formulation, which is the unique aspect of our cost estimation model that distinguishes itself from existing models.

3. The Cloud PC Laboratory Platform

Managing a PC laboratory for a specific programming class is not an easy matter as PCs in a laboratory are usually shared by other classes and users. In particular, requirements of multiple classes and users are difficult to be satisfied with the same PC because operating systems and the software stack of a PC should be setup with predefined configurations and then be fixed.

To resolve these issues, various ways have been attempted, including multi-booting, operating system streaming, and PC virtualization. However, none of these offers a complete solution for cost and management aspects. A public cloud can be an alternative solution to provide a unique environment for each student or class by supporting virtualized PC environments in a template-based fashion [22].

In this article, we develop a PC laboratory solution based on a public cloud called CLABO (CLOUD LABORatory) [23]. In CLABO, each student is assigned a virtual machine with one's own computing environment and configurations, which can be used regardless of locations. Instructors can build a customized virtual machine template for a class by defining virtual machine images and administrative attributes including access permissions. Once a template is created, instructors can generate and distribute a bulk of virtual machines for students in the batch. CLABO also supports an easy installation of libraries and applications required for classes, which can simply be mirrored to the virtual machine of each student.

If a virtual machine of a student is turned on but idle for a long time, CLABO stops it automatically to avoid unnecessary costs of public cloud. Maximum time for each student to use his/her virtual machine can also be set according to the cloud utilization policy. CLABO provides the estimated cost of public cloud for all students' virtual machines instantly based on the IaaS cost estimation model proposed in this article. We have developed and operated the CLABO service for three semesters at Ewha University and it is now publicly available on the web [23].

4. An IaaS Cost Estimation Model

In this section, we describe the cost estimation model for IaaS in public cloud. Each virtual machine in a cloud is called *instance* and the state of an instance can be either *active* or *inactive*. Thus, we estimate the cost of an instance differently by considering the state of the instance, i.e., the active instance cost $Cost_{inst}^{act}$ and the inactive instance cost $Cost_{inst}^{inact}$. An active instance cost $Cost_{inst}^{act}$ is the sum of the CPU cost, the storage cost, and the network cost as described in Equation (1). The CPU cost is determined by the instance type T and the usage time t of the instance. The storage cost is determined by the instance type T , the usage time t , and the storage usage function f_{sto} , which evaluates the total number of I/O activities for a given time. The network cost of an instance is determined by the usage time t and the network usage function f_{net} , which evaluates the total amount of data transferred over the network.

$$Cost_{inst}^{act}(t, T, f_{sto}, f_{net}) = Cost_{cpu}^{act}(t, T) + Cost_{sto}^{act}(t, T, f_{sto}) + Cost_{net}^{act}(t, f_{net}) \quad (1)$$

Now, let us see how the cost of each resource type can be defined. First, the CPU cost $Cost_{cpu}^{act}$ is determined by multiplying the unit price of CPU in instance T , i.e., P_{cpu}^T and the usage time t .

$$Cost_{cpu}^{act}(t, T) = P_{cpu}^T \cdot t \quad (2)$$

Second, the storage cost $Cost_{sto}^{act}$ is composed of the storage volume cost $Cost_{vol}^{act}$ and the storage I/O cost $Cost_{io}^{act}$. $Cost_{vol}^{act}$ is the cost of the storage space, which is expressed by multiplying the usage time t and the unit price of the storage volume in instance T , i.e., P_{vol}^T . $Cost_{io}^{act}$ is the cost for storage I/O activities, which is determined by multiplying the unit price of storage I/O, i.e., P_{io} and the number of I/Os during the usage time t , which is accumulated by f_{sto} that counts the number of I/Os at each monitoring interval.

$$Cost_{sto}^{act}(t, T, f_{sto}) = Cost_{vol}^{act}(t, T) + Cost_{io}^{act}(t, f_{sto}) = P_{vol}^T \cdot t + P_{io} \cdot \int^t f_{sto} \quad (3)$$

The network cost $Cost_{net}^{act}$ is determined by multiplying the amount of data transferred and the unit price of the network P_{net} , which consists of in-bound and out-bound costs. The costs depend on the unit price of the in-bound and out-bound data, i.e., P_{net}^{in} and P_{net}^{out} , respectively.

$$Cost_{net}^{act}(t, f_{net}) = P_{net} \cdot \int^t f_{net} = P_{net}^{in} \cdot \int^t f_{net}^{in} + P_{net}^{out} \cdot \int^t f_{net}^{out} \quad (4)$$

The active instance cost $Cost_{inst}^{act}$ can finally be expressed as Equation (5), where P_{cpu}^T and P_{vol}^T are fixed once the instance type T is determined, and P_{io} and P_{net} are also predefined by CSPs. Thus, the active instance cost can be determined by the usage time t and the amount of storage and network resources used, which are accumulated during the usage time.

$$Cost_{inst}^{act}(t, T, f_{sto}, f_{net}) = P_{cpu}^T \cdot t + P_{vol}^T \cdot t + P_{io} \cdot \int^t f_{sto} + P_{net} \cdot \int^t f_{net} = (P_{cpu}^T + P_{vol}^T) \cdot t + P_{io} \cdot \int^t f_{sto} + P_{net} \cdot \int^t f_{net} \quad (5)$$

Now, let us see how the inactive instance cost can be estimated. As the storage volume is the only resource that will be charged for an inactive instance, the cost of an inactive instance $Cost_{inst}^{inact}$ can be estimated by multiplying the inactive time t and P_{vol}^T as expressed in Equation (6).

$$Cost_{inst}^{inact}(t, T) = Cost_{sto}^{inact}(t, T) = P_{vol}^T \cdot t \quad (6)$$

Finally, the total cost of an instance, $Cost_{inst}$ is estimated by adding the active instance cost and the inactive instance cost as shown in Equation (7).

$$Cost_{inst}(t_{act}, t_{inact}, T, f_{sto}, f_{net}) = Cost_{inst}^{act}(t_{act}, T, f_{sto}, f_{net}) + Cost_{inst}^{inact}(t_{inact}, T) \quad (7)$$

5. Validating the Estimation Model

To determine the cost of an instance, metering of the instance's resource usage is necessary. This is the core process for billing IaaS, and thus CSP performs the metering and periodically reports the results to users [24]. However, as CSP does not perform real-time monitoring for metering, users cannot recognize the cost of IaaS instantly, making a quick decision of services difficult. CLABO provides the real-time resource monitoring feature via metering daemons running on a guest operating system.

The accuracy of CLABO's real-time usage monitoring can be validated by comparing it with the "AWS cost and usage report" [25]. To do so, we execute various instance scenarios on AWS and compare the resource usage of each scenario monitored by CLABO and that of the AWS usage report. Specifically, we generate 19 instance utilization scenarios as shown in Table 1.

Table 1. Scenarios of the cloud PC (Personal Computer) laboratory platform.

Scenario	Description
Lixm	Start Linux instance and shutdown after 20 min idle time; instance t2.micro used
Lixs	Start Linux instance and shutdown after 20 min idle time; instance t2.small used
Wixm	Start Windows and shutdown after 20 min idle time; instance t2.micro used
Wixs	Start Windows and shutdown after 20 min idle time; instance t2.small used
Lins	Connect via VNC; normal screen size; shutdown after 20 min idle time; instance t2.small used
Lifs	Connect via VNC; full screen size; shutdown after 20 min idle time; instance t2.small used
Wins	Connect via remote desktop; normal screen; shutdown after 20 min idle time; instance t2.small used
Wifs	Connect via remote desktop; full screen; shutdown after 20 min idle time; instance t2.small used
Ltxs	Connect via ssh; execute admin. commands for 20 min and shutdown; instance t2.small used
Lgxs	Connect via ssh; write about 200 lines of C code which sorts text lines; edit with vi and compile with gcc; repeat this for 20 min; instance t2.small used
Wvfs	Connect via remote desktop; write about 200 lines of C code which sorts text lines; edit and build with visual studio; repeat this for 20 min; instance t2.small used
Lwfs	Visit a popular web portal in Korea (http://naver.com) using Firefox; reading eight news and four comics with lots of animating images; instance t2.small used
Wwfs	Visit a popular web portal in Korea (http://naver.com) using Internet Explorer; reading eight news and four comics with lots of animating images; instance t2.small used
Lefs	Write three simple Java classes for simple text processing; build and run with Eclipse under Linux for 20 min; instance t2.small used
Wefs	Write three simple Java classes for simple text processing; build and run with Eclipse under Windows for 20 min; instance t2.small used
Lafs	Open a HelloWorld tutorial project with Android studio; modify an app design using a layout editor and launch a virtual device; test this for 20 min; instance t2.small used
LafM	Open a HelloWorld tutorial project with Android studio; modify an app design using a layout editor and launch a virtual device; test this for 20 min; instance t2.medium used
Wafs	Open a HelloWorld tutorial project with Android studio for Windows; modify an app design using a layout editor and launch a virtual device; test this for 20 min; instance t2.small used
WafM	Open a HelloWorld tutorial project with Android studio for Windows; modify an app design using a layout editor and launch a virtual device; test this for 20 min; instance t2.medium used

Each scenario is represented by the four classifiers, i.e., the operating system, the application characteristics, the instance access method, and the instance type. In the case of the operating system, “L” and “W” represent Linux and Windows, respectively, as listed in Table 1. In the case of the application characteristics, our scenario considers five commonly used applications in a software laboratory class: GNU development tools “g”, visual studio “v”, web browser “w”, eclipse “e”, Android studio “a”, terminal tasks “t”, and idle session “i”. The instance access method depends on how VDI is established, that is, “x” for no VDI connection, “n” for windowed VDI, and “f” for full-screen VDI. The instance type of a virtual machine is denoted by “m” for t2.micro, “s” for t2.small, and “M” for t2.medium.

We experiment each scenario on AWS for 20 min and compare the results of our estimation model and the “AWS cost and usage report”. Figure 1 shows the instance usage time monitored by CLABO and that extracted from the “AWS cost and usage report”. In this graph, the results from the Windows platform are excluded as AWS EC2 charges the instance usage of the Windows platform on an hourly basis. As shown in the figure, the usage time monitored by CLABO and that of the “AWS cost and usage report” are very similar. Specifically, the accuracy of our model is shown to be 99.2% on average.

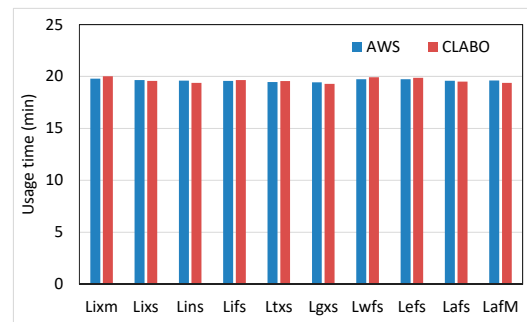


Figure 1. Comparison of the instance usage time for each scenario.

Figure 2 shows the instance usage time as time progresses for the Lixm scenario. The x-axis in the figure implies the execution scenario for the given time, and it may incur some variation of time while the user executes it. For example, booting, configuring, or shutdown of an instance can make variation of the usage time. In contrast, the time reported by AWS is the exact usage time of the instance, and the usage time monitored by CLABO may incur some inaccuracy as it is measured by the metering daemon every 20 s. As shown in the figure, the usage time monitored by CLABO is consistently similar to that of the AWS's report. Specifically, the accuracy becomes better over time; it was 99.7% when the execution time is 20 min but became 99.9% as the time is 70 min. In some cases, over-estimation by CLABO occurs as the detection of instance termination is delayed due to the periodic execution of the instance status checker. However, this error is very small compared to the total instance time and can be negligible as time progresses.

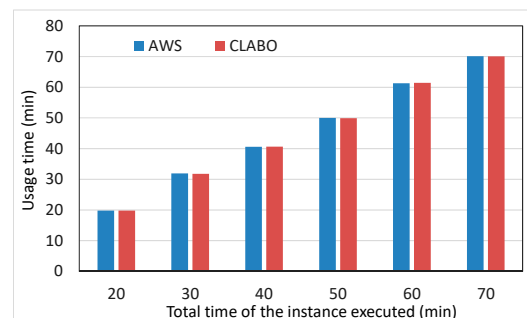


Figure 2. Comparison of the instance usage time as time progresses.

Let us now see the network resource usage. Figures 3 and 4, respectively, show the in-bound and out-bound network traffic for each scenario. As shown in the figures, the difference between the network traffic monitored by CLABO and that of the AWS's report is very small. In the case of the in-bound network traffic, the monitoring result of CLABO exhibits a little more than the AWS reports. Lwfs and Wwfs incur a large network traffic of 100 MB or more but the difference is still small enough.

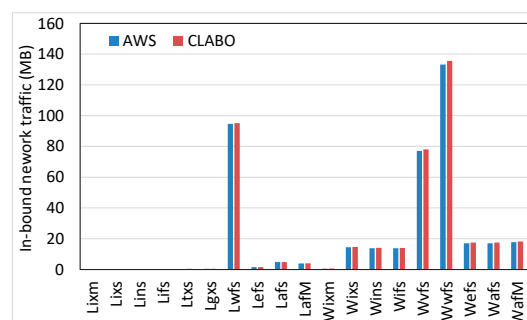


Figure 3. Comparison of the in-bound network traffic.

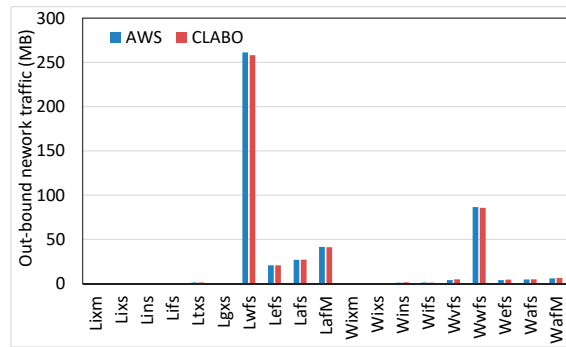


Figure 4. Comparison of the out-bound network traffic.

Figures 5 and 6 show the number of storage I/Os in terms of the read and write operations, respectively. Unlike other resource cases, storage usage monitored by CLABO and that reported by AWS have a certain amount of gap. This is because most CSPs use an I/O count as the cost metric of storage activity, but it is difficult to count the exact number of I/O operations at an instance level. In particular, recognizing the initial I/O activities is difficult in CLABO as its monitoring routine starts after the initial I/O activities. Moreover, I/O requests issued by an instance are usually merged by the storage-layer policies.

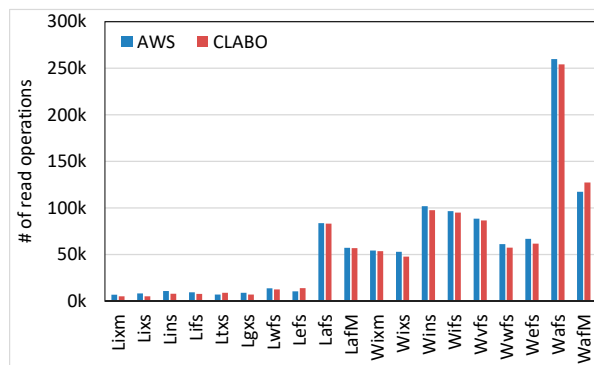


Figure 5. Comparison of the storage read counts.

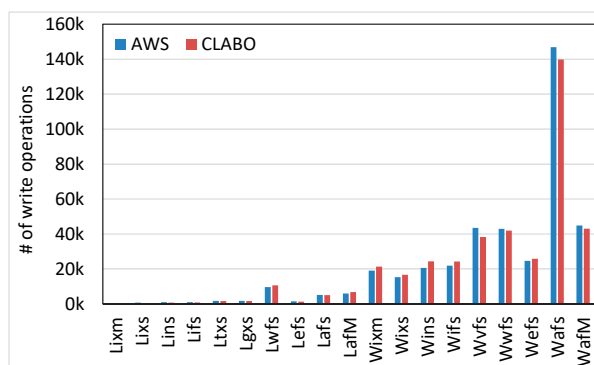


Figure 6. Comparison of the storage write counts.

We consider this by using an additional parameter to estimate the number of I/O operations that occur before CLABO monitoring starts, and accumulate them. This is derived by performing some dummy scenarios, which have almost no storage activities at the instance level, and compare the results of CLABO and AWS report. Even though the estimation of storage activities is not as accurate as other resource cases, the accuracy is greater than 95% when the usage of storage resources is over a certain level. In all Windows platform scenarios, the number of storage operations tends to be large because

Windows platforms have more I/O requirements of complex software stacks and incur additional paging I/Os due to the large memory usage. All numerical results obtained throughout experiments about CLABO monitoring comparison to AWS reporting are summarized in Table 2.

Table 2. Numerical results of instance usage time, network usage, and storage count for 19 scenarios.

Scenario	Usage (Min)		Net In (MB)		Net Out (MB)		Read Count (K)		Write Count (K)	
	AWS	CLABO	AWS	CLABO	AWS	CLABO	AWS	CLABO	AWS	CLABO
Lixm	19.78	20.02	0.02	0.04	0.02	0.05	6.94	6.80	0.36	0.35
Lixs	19.65	19.57	0.01	0.03	0.00	0.03	8.34	7.93	0.71	0.65
Wixm	-	-	0.57	0.64	0.08	0.13	54.35	52.89	19.04	21.32
Wixs	-	-	14.43	14.61	0.38	0.50	53.17	48.91	15.31	17.21
Lins	19.60	19.38	0.02	0.04	0.18	0.20	10.98	8.91	0.99	0.50
Lifs	19.57	19.66	0.14	0.16	0.38	0.42	9.50	8.88	1.02	0.82
Wins	-	-	13.82	14.04	1.11	1.58	102.00	98.28	20.61	23.89
Wifs	-	-	13.86	14.01	1.48	1.15	96.62	94.84	21.89	23.81
Ltxs	19.47	19.55	0.32	0.35	1.31	1.40	7.14	9.05	1.70	1.55
Lgxs	19.43	19.28	0.42	0.44	0.50	0.59	9.00	7.38	1.69	1.60
Wvws	-	-	77.04	78.09	4.26	5.07	88.60	84.81	43.47	38.94
Lwfs	19.73	19.93	94.73	95.12	261.20	258.05	13.76	12.31	9.62	11.24
Wwfs	-	-	133.24	135.54	86.60	85.84	61.16	58.10	42.96	41.68
Lefs	19.73	19.88	1.53	1.55	20.74	20.90	10.61	12.50	1.43	1.01
Wefs	-	-	17.01	17.45	4.25	4.54	66.91	62.19	24.57	24.10
Lafs	19.58	19.51	4.79	4.82	27.08	27.18	83.84	82.28	5.10	5.01
LafM	19.62	19.39	3.98	4.00	41.57	41.43	57.35	56.99	5.96	5.98
Wafs	-	-	17.08	17.50	4.75	5.07	259.69	252.75	146.87	140.01
WafM	-	-	17.68	18.17	6.17	6.45	117.25	125.13	44.91	42.06

6. Adopting the Model to Real-World Applications

CLABO can promptly track the cost of all created instances and also provide the cost breakdown analysis such as cost per user or cost per class. To quantify the accuracy of our cost estimation model, we utilized CLABO in two classes that require cloud PC laboratory platforms during the 2018 fall semester. The number of students was 50, and we generated 50 instances on AWS and compared the total cost charged by AWS and the cost estimated by our instant model.

Figure 7 shows the total costs charged by AWS for October, November, and December of 2018. As shown in the figure, CPU accounts for the largest portion of the cost for all months. The cost of the storage volume is varied significantly for different months, which accounts for 15% to 49% of the total cost. Note that the cost of the storage volume mostly resulted from the inactive instances, as the storage volume is charged although an instance is not active. Our analysis shows that over 95% of the storage volume cost occurred when the instances were inactive. As shown in the figure, the costs of the network and the storage I/O are not significant. The overall cost of the 50 virtual machines is less than 25 USD as the price of a t2.micro Linux instance is very low and CLABO automatically turns off the instances that are not used for a long time.

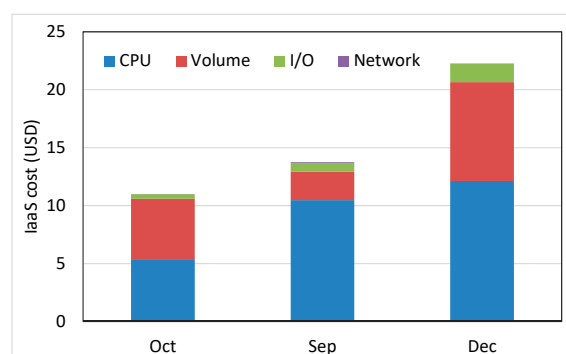
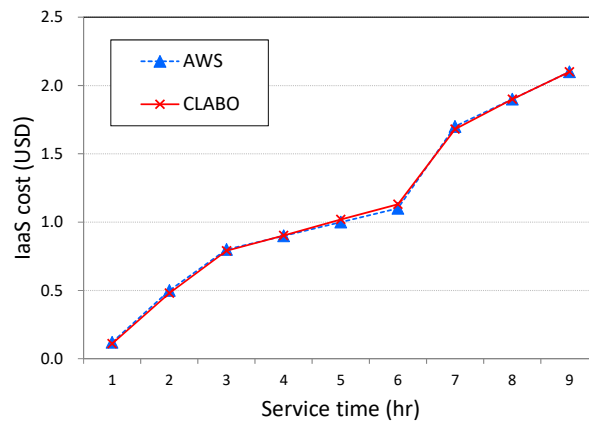
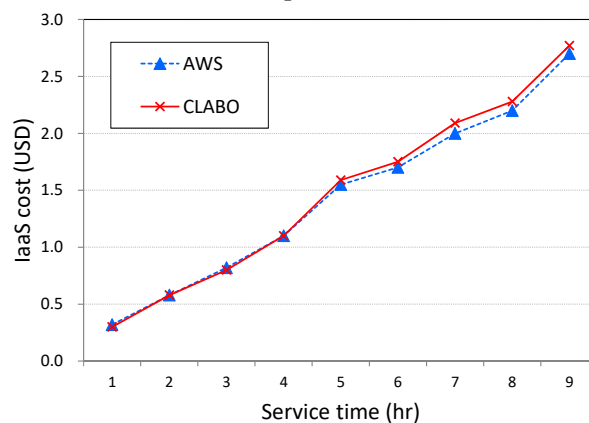


Figure 7. Per month IaaS cost for 50 virtual machines.

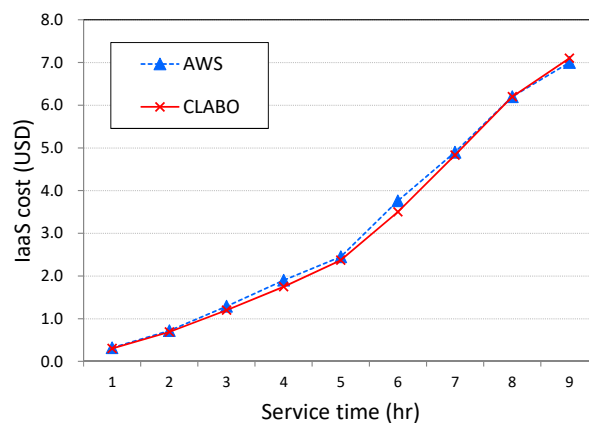
Figure 8 compares the total instance cost estimated by CLABO and the costs charged by AWS as time progresses. As the AWS usage report provides the billing information by the precision level of 1 h unit, we also compare the billing information provided by AWS and the estimated cost for each hour. As shown in the figure, the IaaS cost estimated by our model and the real cost charged are very similar. The error rate of the estimation is less than 1% for all cases although our estimation was performed instantly, and the accuracy is 99.3% on average compared to the actual charge of AWS.



(a) September



(b) October



(c) November

Figure 8. Comparison of IaaS (Infrastructure As A Service) cost estimated by the proposed model and that charged by AWS (Amazon Web Service).

We conduct some additional experiments to validate the proposed model. In particular, we execute two types of workloads, rendering and web service, and compare the estimated cost of our model with the real AWS charge. Note that rendering is a computing-intensive workload, whereas web service is an I/O-intensive workload, which are the two representative types of workload that can be executed on cloud. Figure 9a depicts the accumulated hourly costs of our model and AWS charge when web requests were generated by the Apache benchmark. As AWS adopts the auto-scaling functionality, the number of web servers increases or decreases according to the evolution of workloads. However, as shown in the figure, the errors of estimated costs by our model (denoted as Web) are less than 1% for all cases. Figure 9b shows the results for the rendering workload. This experiment was conducted by using eight rendering servers, which are set to terminate automatically when each assigned job is finished. Rendering is a computing-intensive workload and the cost is mainly dependent on the instance running time, and thus the cost graph is almost linear as time progresses before the completion of the rendering job. As shown in the figure, we can again see that our model estimates the real AWS charge precisely and the error rate is below 0.9% for all cases.

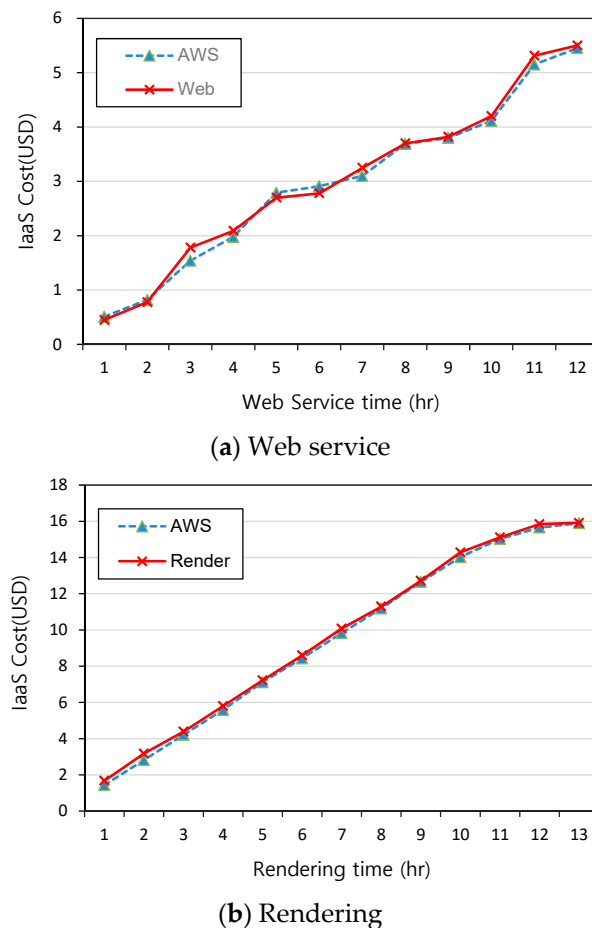


Figure 9. Validation of the proposed model under web service and rendering workloads.

We perform another experiment to compare our model with the existing model that estimates the cost of a certain period in the future based on the resource usage of the previous history. As existing cost models predict future resource usage by adopting complicated modeling, it is difficult to be implemented as an instant monitoring module, so prediction should be performed periodically. Figure 10 compares the estimated cost of the proposed model in comparison with the existing model that estimates the cost every 3 h. In this experiment, eight web servers were serviced during 12 h. As shown in the figure, our estimation model performs better than the existing model as our model estimates the resource cost instantly, but the existing model overestimates or underestimates the IaaS

cost depending on the previous usage of the resources even though it can be adjusted by periodic usage report. Note that this experiment does not imply the superiority of our model as it monitors the resource usage instantly, but the existing model predicts future resource usage beforehand, and thus the goals of the models are different.

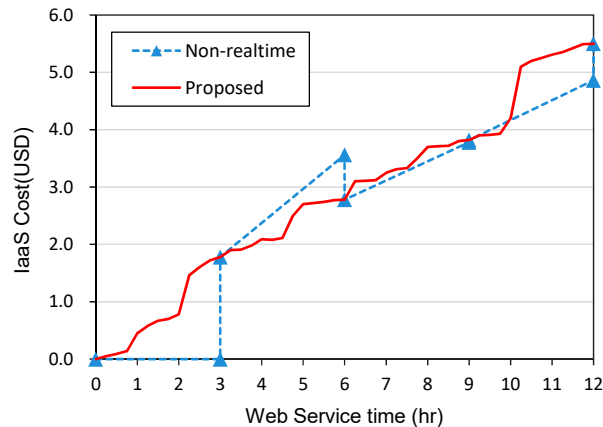


Figure 10. Comparison of the proposed model with the existing non-real-time model.

7. Conclusions

Cloud is an attractive computing platform as resources can be scaled up and down promptly according to the user's computing demands. However, unnecessary cost can be incurred due to excessive idle instances and/or temporarily spiked usage [7,24]. Thus, customers of cloud services must conduct the monitoring of the charge continuously. Unfortunately, this is difficult as informing the charge of the IaaS requires at least several hours or even a couple of days. This article presented an instant cost estimation model for IaaS resources, and adopted it to the real system called CLABO, which monitors the resource usage of public cloud by making use of a metering daemon. To validate our model, we ran PC laboratory services on AWS for a full semester. Experimental results showed that the accuracy of our model is over 99.3% in comparison with the actual charge of AWS.

One constraint of our model is that it is applicable only when the real-time resource usage monitoring is possible. If any of the metering components for IaaS cost estimation is not available, instant cost estimation will be difficult. However, as we have shown through AWS, monitoring of instance-level resource usage is quite accurate and feasible to track by utilizing the usage report of CSPs. In particular, major CSPs gradually offer resource metering information such as AWS CloudWatch and Azure Monitor. Thus, we plan to extend our cost model by using CSP-level metering facilities and further validate it via other public cloud platforms such as Azure and GCP.

As our approach is based on the user-side metering daemon, it incurs a slight overhead, which can be considered as a weakness of our model. However, we further observed that our metering daemon incurs less than 0.1% of the total cost as our design focused on the minimization of the estimation overhead by simplifying the formulation.

Author Contributions: K.C. designed the architecture and algorithm and performed the experiments. H.B. supervised the work and provided expertise. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2016R1A6A3A11930295) and also funded by the ICT R&D program of MSIP/IITP (2019-0-00074, developing system software technologies for emerging new memory that adaptively learn workload characteristics).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Joshi, N.; Shah, S. A comprehensive survey of services provided by prevalent cloud computing environments. In *Smart Intelligent Computing and Applications*; Springer: Singapore, 2019; pp. 413–424.
2. Sushil, B.; Jain, L.; Jain, S. Cloud computing: A study of infrastructure as a service (IAAS). *Int. J. Eng. Inf. Technol.* **2010**, *2*, 60–63.
3. Chaudry, R.; Guabtni, A.; Fekete, A.; Bass, L.; Liu, A. Consumer Monitoring of Infrastructure Performance in a Public Cloud. In Proceedings of the International Conference on Web Information Systems Engineering, Thessaloniki, Greece, 12–14 October 2014; Springer: Cham, Switzerland, 2014.
4. Al-Roomi, M.; Al-Ebrahim, S.; Buqrais, S.; Ahmad, I. Cloud computing pricing models: A survey. *Int. J. Grid Distrib. Comput.* **2013**, *6*, 93–106. [[CrossRef](#)]
5. Julie, K.; Hyokyung, B. An Efficient Log Data Management Architecture for Big Data Processing in Cloud Computing Environments. *J. Inst. Internet Broadcast. Commun.* **2013**, *13*, 1–7. [[CrossRef](#)]
6. Artan, M.; Shabani, I.; Sejdiu, B. Pricing schemes in cloud computing: An overview. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 80–86.
7. Benedikt, M.; Walterbusch, M.; Teuteberg, F. Costing of cloud computing services: A total cost of ownership approach. In Proceedings of the 2012 45th Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2012.
8. Belusso, C.L.; Sawicki, S.; Basto-Fernandes, V.; Frantz, R.Z.; Roos-Frantz, F. A proposal of Infrastructure-as-a-Service providers pricing model using linear regression. *Revista Brasileira de Computação Aplicada* **2018**, *10*, 44–53. [[CrossRef](#)]
9. Mitropoulou, P.; Filiopoulou, E.; Nikolaidou, M.; Michalakelis, C. Pricing IaaS: A hedonic price index approach. In Proceedings of the International Conference on the Economics of Grids, Clouds, Systems, and Services, Biarritz-Anglet-Bayonne, France, 19–21 September 2017; Springer: Cham, Switzerland, 2017.
10. Wu, C.; Toosi, A.N.; Buyya, R. Hedonic pricing of cloud computing services. *IEEE Trans. Cloud Comput.* **2018**. [[CrossRef](#)]
11. Hinz, M.; Koslovski, G.P.; Miers, C.C.; Pilla, L.L.; Pillon, M.A. A cost model for iaas clouds based on virtual machine energy consumption. *J. Grid Comput.* **2018**, *16*, 493–512. [[CrossRef](#)]
12. Aldossary, M.; Djemame, K.; Alzamil, I.; Kostopoulos, A.; Dimakis, A.; Agiatzidou, E. Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. *Future Gener. Comput. Syst.* **2019**, *93*, 442–459. [[CrossRef](#)]
13. Sadeghi, S.; Tarokh, M.J. A Cost Model for Hybrid Cloud. *Int. J. Comput. Inf. Technol.* **2017**, *5*, 1–6.
14. Tang, S.; Niu, Z.; He, B.; Lee, B.S.; Yu, C. Long-term multi-resource fairness for pay-as-you use computing systems. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1147–1160. [[CrossRef](#)]
15. Wang, L.; Hu, G. Smart pricing for cloud resource and service markets: A brief overview. In Proceedings of the 2016 12th IEEE International Conference on Control and Automation (ICCA), Kathmandu, Nepal, 1–3 June 2016.
16. Xu, Y.; Yao, J.; Jacobsen, H.A.; Guan, H. Cost-efficient negotiation over multiple resources with reinforcement learning. In Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), Vilanova i la Geltru, Spain, 14–16 June 2017.
17. Agarwal, S.; Mishra, A.K.; Yadav, D.K. Forecasting price of amazon spot instances using neural networks. *Int. J. Appl. Eng. Res.* **2017**, *12*, 10276–10283.
18. Kumar, S.V.; Dutta, K. Dynamic price prediction for amazon spot instances. In Proceedings of the 2015 48th Hawaii International Conference on System Sciences, Kauai, HI, USA, 5–8 January 2015.
19. Turchenko, V.; Shults, V.; Turchenko, I.; Wallace, R.M.; Sheikhalishahi, M.; Vazquez-Poletti, J.L.; Grandinetti, L. Spot price prediction for cloud computing using neural networks. *Int. J. Comput.* **2014**, *12*, 348–359.
20. Meng, Q.N.; Xu, X. Price forecasting using an ACO-based support vector regression ensemble in cloud manufacturing. *Comput. Ind. Eng.* **2018**, *125*, 171–177. [[CrossRef](#)]
21. He, H.; Ma, Z.; Li, X.; Chen, H.; Shao, W. An approach to estimating cost of running cloud applications based on AWS. In Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference, Hong Kong, China, 4–7 December 2012; Volume 1.
22. Washington, E.; Sequera, J. Model to implement virtual computing labs via cloud computing services. *Symmetry* **2017**, *9*, 117. [[CrossRef](#)]

23. CLABO Service. Available online: <http://sys-sw.clabo.codemayo.com> (accessed on 18 October 2019).
24. Anwar, A.; Sailer, A.; Kochut, A.; Schulz, C.O.; Segal, A.; Butt, A.R. Scalable metering for an affordable it cloud service management. In Proceedings of the 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, USA, 9–13 March 2015.
25. AWS Cost and Usage Report. Available online: <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-reports-costusage.html> (accessed on 21 October 2019).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).