Cross-provider Cloud Cost Calculation

Christian Plewnia[1], Horst Lichter[1]

[1]RWTH Aachen University, Aachen, Germany

Abstract

Cloud computing grants flexible access to resources such as servers and storage to create compute infrastructures while only paying for what is used. Due to the complexity of usage-based pricing models, it is not feasible to obtain cost estimates for larger infrastructures manually. However, the available cost calculation tools mostly support calculations for one provider and are not open to importing existing infrastructure models, which makes cost estimations for large infrastructures of enterprises tedious. In this paper, we present a first version of a cross-provider cloud cost calculation model as a step towards enabling cost estimations across providers without requiring knowledge of the providers' pricing models.

Keywords

cloud computing, cost modeling, infrastructure as a service

1. Motivation & Related Work

Enterprises have widely adopted the use of Infrastructure as a Service (IaaS) cloud computing: in 2021, 81% of Europe's large businesses used IaaS [1]. One of the benefits of cloud computing is the flexibility to obtain computing resources, such as virtual machines (VMs) and storage, on demand without upfront cost, i.e., customers only pay for the time and extent of the resource usage. However, the usage-based pricing makes cost estimations for new cloud infrastructures, changes to existing infrastructures, and changes in cloud providers' prices difficult. This is especially true for large infrastructures of enterprises, a problem that was already addressed in 2011 by a cross-provider cost modeling tool that enabled the cost calculation for infrastructures modeled as UML deployment diagram [2]. Unfortunately, this tool is no longer accessible and the cloud providers' pricing models advanced since then; where this tool reportedly calculated the costs for all providers and services as products of usage and price, nowadays the calculations involve more than a simple product and differ per provider and service. To help with cost calculations, cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) offer calculators [3, 4, 5] that allow users to get a cost estimate, but only for the respective provider. There is also a cross-provider cost calculator, which is limited to a few providers and services without any possibility of extension [6]. All calculators require the user to model the infrastructure via its user interface; they neither allow importing existing infrastructure models nor can be accessed via an Application Programming Interface (API).

13th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA) 2023, May 11–12, 2023, Stockholm, Sweden

plewnia@swc.rwth-aachen.de (C. Plewnia); lichter@swc.rwth-aachen.de (H. Lichter)
https://swc.rwth-aachen.de (C. Plewnia); https://swc.rwth-aachen.de (H. Lichter)
0000-0003-0221-9842 (C. Plewnia); 0000-0002-3440-1238 (H. Lichter)

With this paper, we aim to revive the idea of a cross-provider model that can calculate a cost estimate for a given infrastructure and choice of cloud providers. For this, we first describe the selection of cloud providers and services, which served as a reference for the development of our model (section 2). Then, we present the first version of the model, which can consider VMs and block storage; this includes a notation for the infrastructure and the information on the cloud providers required for the cost calculation (section 3). In section 4, we discuss the plausibility of the cost calculations and the current limitations.

2. Reference for the Model Construction

To construct the cost calculation model, we need to study cloud providers. With this being initial work in this direction, we decided to focus on a small set of IaaS providers. We find it to be reasonable to start with the cloud providers that have the highest market share. The only market share information that we are aware of is from ranking reports by analysts such as Gartner [7], providing revenue and market share for 2021, and Acceleration Economy Network [8], providing revenue for 2022. While the rankings are inconsistent with respect to the considered providers, the rankings have in common that AWS [3], Azure [4], and GCP [5] are among the top five cloud providers. According to Gartner, these three covered 67.1% of 2021's market share. We decided to use these three providers as reference.

Another dimension relevant to our construction is the considered type of cloud service. By type, we understand services that are offered by cloud providers in a largely similar way. For example, AWS, Azure, and GCP all offer servers and block storage. We focus on servers and block storage as they are part of the infrastructure required for many software applications. For the model construction, we consider server offers from Elastic Compute Cloud (EC2) (AWS), Compute (Azure), and Compute Engine (GCP) as well as block storage offers from Elastic Block Storage (EBS) (AWS), Disk Storage (Azure), and Persistent Disk (GCP).

3. Cross-provider Cost Calculation Model (CCCM)

The idea of the cross-provider cost calculation is for its users to be able to give it a description of an infrastructure and for it to return estimated cost, without the users needing to know the pricing mechanisms of the cloud providers. Since costs are incurred over time, we have to consider what time interval to calculate the costs for. In this context, *billing periods* must be considered as some effects, such as maximum cost per billing period, depend on this. The billing periods for AWS, Azure, and GCP correspond to calendar months. Since costs are accounted for by cloud providers separately per billing period, we subsequently consider only time intervals no longer than one billing period. Note that cost calculations for longer time intervals can still be achieved by summing up the cost per billing period.

Example Infrastructure To illustrate the CCCM and the choices we made for its construction, we define an example infrastructure for February 2023 with two resources, a VM as a server (r_{vm}) with additional block storage (r_{bs}) as the server's storage:

Offer details (as of March 17th 2023)				Resource cost calculation			
Off.	BI[a]	Component	Price	Res.	Component cost (*compCost*)		*resourceCost*
o_{vm}	60s/1s	Instance time	\$0.1536/h	r_{vm}	28d · 24h · \$0.1536	= \$103.22	\$103.22
o_{bs}	60s/1s	Storage size	\$0.0952/GiB-month	r_{bs}	(19d · 1TiB + 9d · 2TiB) · \$0.0952/(GiB · 28d)	= \$89.69	
		I/O operations	\$0.006/IOPS-month[1]		28d · 500IOPS · \$0.006/(IOPS · 28d)	= \$3.00	
		Throughput	\$0.048/MiB/s-month[2]		28d · 0MiB/s · \$0.048/(MiB/s · 28d)	= \$0.00	\$92.69

a: billing increment (first-time interval/following intervals); 1: only charged for the provisioned amount exceeding 3,000 IOPS, 2: only charged for the provisioned amount exceeding 125 MiB/s

Table 1
Details for AWS offers EC2 t4g.xlarge (=o_{vm}) and EBS gp3 (=o_{bs}) and the cost calculation for r_{vm} and r_{bs}.

r_{vm} A VM with 4 vCPU (logical CPUs) and 16 GB memory.

r_{bs} A block storage with initially 1 TiB of disk space, which is later increased to 2 TiB on February 20th 00:00. We further expect to require 3,500 IO operations/s (IOPS) and a throughput of 125 MB/s – numbers we could have obtained by benchmarking the software application we intend to deploy.

Total Cost Formally speaking, we define an infrastructure to be a set of resources R with R being a subset of the domain of all possible resources $\mathcal{R}$, i.e., $R \subseteq \mathcal{R}$. In the example, we define $R = \{r_{vm}, r_{bs}\}$. We then can calculate the cost for a set of resources R with

$$totalCost(R) : \mathbb{P}(\mathcal{R}) \rightarrow \mathcal{M} = \textstyle\sum_{r \in R} resourceCost(r), \tag{1}$$

where $\mathbb{P}$ denotes the power set and $\mathcal{M}$ is the domain of monetary cost.

Offers To calculate the cost for a resource r, we need to decide on an offer that provides this resource. For the example, we choose for r_{vm} the AWS EC2 VM type t4g.xlarge (=o_{vm}), which matches the needs for vCPU and memory, and for r_{bs} we choose AWS EBS gp3 (=o_{bs}), which provides the required storage. We encode this choice in $assignedOffer(r) : \mathcal{R} \rightarrow \mathcal{O}$, e.g., $assignedOffer(r_{vm}) = o_{vm}$ and $assignedOffer(r_{bs}) = o_{bs}$. Table 1 shows the offers and how the cost for r_{vm} and r_{bs} are calculated. The table also shows that both offers define a minimum usage time of 60s and after that charge for 1s increments — details we consider later.

Usage The cost calculation of a resource also requires us to provide the relevant usage parameters, e.g., the time we intend to use r_{vm} as well as the storage, IOPS, and throughput for r_{bs}. We identify each usage parameter by the respective resource's *component*. We define a component c to be an element of the domain of all components $\mathcal{C}$, i.e., $c \in \mathcal{C}$. In our example, we have the components c_{time}, c_{stor}, c_{iops}, and c_{tp} (throughput). The complete picture of all components for the reference set is given in table 2.

The usage parameter for a component is a time series of usage values. The time series allows us to consider changes in the usage, such as the increase in disk storage of r_{bs}. We define a usage to be $u_c = (u_{c,1}; \ldots; u_{c,m}) : \mathcal{U}$ with $c \in \mathcal{C}$ and $\mathcal{U}$ being a variable-length tuple of real numbers ($\mathbb{R} \times \cdots \times \mathbb{R}$). For the usage encoding to be unambiguous, we specify the time resolution and

	Server			Block storage											
				AWS				Azure				GCP			
Component	AWS	Azure	GCP	gp2	gp3	io1	io2	SH[a]	SS[b]	PS[c]	U[d]	Std.	SSD	Bal.	Ext.
Instance time c_{time}	✓[1]	✓[2]	-												
vCPU c_{vcpu}	-	-	✓												
Memory c_{mem}	-	-	✓												
Storage size c_{stor}				✓[1]	✓[1]	✓[1]	✓[1]	✓	✓	✓	✓	✓	✓	✓	✓
I/O operations c_{iops}				-	✓[1,2]	✓[1]	✓[1]	✓	✓[3]	✓	✓	-	-	-	✓
Throughput c_{tp}				-	✓[1,2]	-	-	-	-	✓	✓	-	-	-	-

a: Std. HDD, b: Std. SSD, c: Premium SSD v2, d: Ultra; 1: defines a $minUsage_{c,o}$, 2: defines an $adjust_{c,o}$, 3: defines a $maxCost_{c,o}$

Table 2
Overview of the components for servers and block storage for AWS, Azure, and GCP.

unit of usage. For the time resolution we use 1s, which is the smallest billing increment we have observed across the cloud providers; thus, the length of u_c for any $c \in \mathcal{C}$ for February 2023 is $|u_c| = 2{,}419{,}200$ (28 d in seconds). For the unit of one usage value $u_{c,t}$, we use the smallest common base unit for the respective component, e.g., 1s for instance time, 1 GiB for storage sizes, and 1 IOPS for IO operations. Thus, each value $u_{c,t}$ encodes the usage quantity in the second t of the current billing period. For example, the usage of block storage size c_{stor} for r_{bs} is $u_{c_{stor}} = (1024 \times 1{,}641{,}600; 2048 \times 777{,}600)$ (in GiB), where $a \times b$ means to repeat the value a for b many times in the tuple; in this example, the series of 1024 corresponds to the 1 TiB required from February 1st until February 20th 00:00 (in seconds: $19\text{d} \cdot 24\text{h} \cdot 60\text{m} \cdot 60\text{s} = 1{,}641{,}600$) and the later series of 2048 corresponds to the 2 TiB required until the end of the month (in seconds: $9\text{d} \cdot 24\text{h} \cdot 60\text{m} \cdot 60\text{s} = 777{,}600$). Another example is the usage $u_{c_{time}} = (1 \times 2{,}246{,}400)$ for r_{vm}, where each usage value $u_{c_{time},t} = 1$ encodes a one second usage of the VM in second t.

To organize the encoding of the usage parameters per resource and component, we define the function $resourceUsage : \mathcal{R} \rightarrow (\mathcal{C} \rightarrow \mathcal{U})$. It provides for a resource r a mapping of the applicable components to the respective usage time series. We denote this intermediate mapping returned by *resourceUsage* as function $compUsage : \mathcal{C} \rightarrow \mathcal{U}$. For example, the aforementioned usage of block storage size c_{size} for r_{bs} can be accessed with $(resourceUsage(r_{bs}))(c_{stor}) = u_{c_{stor}}$.

Resource & Component Cost We can now calculate the cost for a resource with

$$resourceCost(r) : \mathcal{R} \rightarrow \mathcal{M} = offerCost(assignedOffer(r), resourceUsage(r)). \tag{2}$$

We can then calculate the cost for an offer by summing up the component cost with

$$offerCost(o, compUsage) : \mathcal{O} \times (\mathcal{C} \rightarrow \mathcal{U}) \rightarrow \mathcal{M} = \textstyle\sum_{c \in comp(o)} compCost(c, o, compUsage(c)) \text{ and} \tag{3}$$

$$compCost(c, o, u) : \mathcal{C} \times \mathcal{O} \times \mathcal{U} \rightarrow \mathcal{M} = \min\left(cost(c, o, u), maxCost_{c,o}\right), \tag{4}$$

where min(...) optionally allows limiting the cost of a component per billing period to a value defined for the combination of the offer o and component c. Azure uses this for Standard SSD block storage's IOPS component, where a maximum price is given that is lower than summing up the hourly price for a month. The final piece of the cost calculation is

$$cost(c, o, u) : \mathcal{C} \times \mathcal{O} \times \mathcal{U} \rightarrow \mathcal{M} = \max\left(\textstyle\sum_{i=1}^{|u|} adjust_{c,o}(u_i), minUsage_{c,o}\right) \cdot pricePerUnit_{c,o}, \tag{5}$$

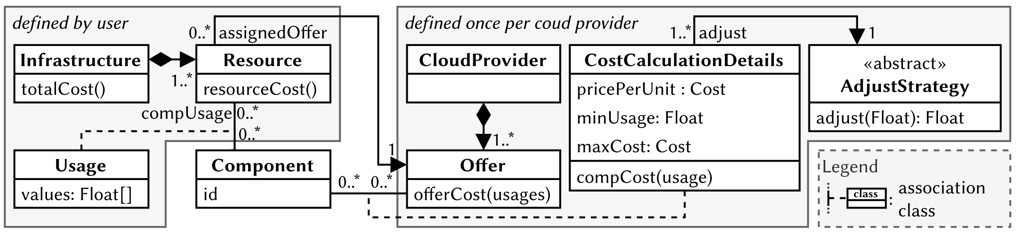

Figure 1: UML class diagram providing an overview of the CCCM.

where $adjust_{c,o}(u) : \mathbb{R} \to \mathbb{R}$ is a function for adjusting the usage values. It is defined per combination of offer $o \in \mathcal{O}$ and component $c \in \mathcal{C}$ and by default is an identity function $adjust_{c,o}(u) = u$. A different definition of the adjustment function allows to accommodate for free quotas, e.g., the free 3,000 IOPS for o_{bs} and c_{iops}, i.e., $adjust_{c,o}(u) = \max(u - 3000, 0)$. The $minUsage_{c,o}$ is a constant defined per combination of offer $o \in \mathcal{O}$ and component $c \in \mathcal{C}$ with a default $minUsage_{c,o} = 0$. It enables the definition of a minimum usage, e.g., $minUsage_{c,o} = 60$ s for the instance time of offer. The constant $pricePerUnit_{c,o}$ defines for each combination of offer $o \in \mathcal{O}$ and component $c \in \mathcal{C}$ the price per usage unit.

Overview Figure 1 shows a UML class diagram of the CCCM. The class and attribute names correspond to the previous descriptions. A key benefit of the CCCM is that the parts for the cloud provider only need to be modeled once. A user interested in calculating the cost of an infrastructure then can focus on modeling the set of resources R, the assignment of each resource to an offer via *assignedOffer*, and the usage *resourceUsage*.

4. Discussion

For the CCCM to be of use, its calculated cost must be plausible, i.e., they must be equal to or sufficiently close to what the cloud providers would calculate. We realized the CCCM as a prototype in Python and implemented software tests to check for the plausibility of the calculated cost. As an initial effort to a systematic testing approach, we implemented at least one test case per combination of cloud provider and service type in our reference set. We obtained the expected cost using the cloud providers' own cost calculators.

We further conducted a small industry case study in which we assessed the cost of a potential cloud migration for multiple scenarios. The base scenario was a migration of an existing software application to a cloud infrastructure. Further scenarios were created to investigate the cloud infrastructure costs in case of a redesign of the application toward a microservice architecture. We provide more information on this case study as supplemental material[1].

The CCCM currently has multiple limitations that can lead to inaccurate cost calculations. The calculated cost for VMs from GCP is higher than it should be if a VM is used for more than 25% of a month, because the model does not yet provide a mechanism to consider the sustained use discount applied by GCP; this discount lowers the hourly cost over the course of a month

[1]see git repository https://github.com/swc-rwth/cross-provider-cloud-cost-model-paper

for continuously used VM vCPUs and memory. The CCCM also does not consider discounts from savings plan contracts. Further, server data transfer costs are not considered.

5. Conclusion & Future Work

We presented the Cross-provider Cost Calculation Model (CCCM), which enables a user to get the costs for a cloud infrastructure without requiring the user to know the cloud providers' pricing models, as long as someone modeled the providers' pricing models once. Compared to an early approach in this direction [2], the CCCM considers the advancements in the providers' pricing models and it works for the current pricing models of the providers AWS, Azure, and GCP for the service types servers and block storage and presumably also for other providers and service types. However, as discussed in section 4, this first version of the CCCM still has considerable limitations that may cause the calculated costs to be inaccurate in certain cases.

We make two suggestions for future work. First, to overcome the aforementioned limitations, to stay up to date with possible changes to pricing models, and to enable others to use the cost calculation model, we suggest a continuous open-source development for the cost calculation model. This would also allow to establish a library of modeled offers from cloud providers. Second, in the presented version of the model, the user has to define what resource is fulfilled by what cloud service offer, which can be difficult given the number of cloud providers and services on the market. This work can partially be automated by an optimization approach for finding the best matching offers with respect to an optimization objective for given requirements. The optimization objective can, for example, be to minimize cost. The requirements would express constraints to the acceptable offers and can cover technical aspects, e.g., a VM with at least 4 vCPU and 16 GB memory, but also other aspects, e.g., the chosen provider must have certain security certifications. Thus, a user would have to model resources, their requirements, and their usage. We currently work on conceptual and technical solutions for these suggestions.

References

[1] Cloud computing - statistics on the use by enterprises, Technical Report, Eurostat, 2022.

[2] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, P. Teregowda, Decision Support tools for Cloud Migration in the Enterprise, in: 4th International Conference on Cloud Computing, 2011, pp. 541–548. doi:10.1109/CLOUD.2011.59.

[3] Amazon Web Services Pricing Calculator, 2023. URL: https://calculator.aws/.

[4] Azure Pricing Calculator, 2023. URL: https://azure.microsoft.com/pricing/calculator/.

[5] Google Cloud Pricing Calculator, 2023. URL: https://cloud.google.com/products/calculator.

[6] Cloud Computing Comparison Engine, 2023. URL: https://www.cloudorado.com/.

[7] Gartner Inc., Gartner Says Worldwide IaaS Public Cloud Services Market Grew 41.4% in 2021, 2022. URL: https://www.gartner.com/en/newsroom/press-releases/2022-06-02-gartner-says-worldwide-iaas-public-cloud-services-market-grew-41-percent-in-2021.

[8] Acceleration Economy LLC, The World's Top Cloud Vendors, 2022. URL: https://accelerationeconomy.com/cloud-wars-top-10/.

All online references last accessed March 19th 2023.