


## **Cost-based Virtual Machine Scheduling for Data-as-a-Service**


**Ana Cristina Alves de Oliveira**

(Mobile Applications and Innovation Laboratory (LAMPPIÃO), Federal Institute of Paraíba (IFPB), Campina Grande, PB, Brazil

 <https://orcid.org/0000-0003-0754-9058>, [ana.oliveira@ifpb.edu.br](mailto:ana.oliveira@ifpb.edu.br))


**Marco Aurélio Spohn**

(Federal University of Fronteira Sul (UFFS), Chapecó, SC, Brazil

 <https://orcid.org/0000-0002-9265-9421>, [marco.spohn@uffs.edu.br](mailto:marco.spohn@uffs.edu.br))


**Christof Fetzer**

(Systems Engineering Group, Technical University of Dresden (TU Dresden), Dresden, Germany

 <https://orcid.org/0000-0001-8240-5420>, [christof.fetzer@tu-dresden.de](mailto:christof.fetzer@tu-dresden.de))


**Le Quoc Do**

(Systems Engineering Group, Technical University of Dresden (TU Dresden), Dresden, Germany

 <https://orcid.org/0000-0002-1433-0217>, [dolequoc@tu-dresden.de](mailto:dolequoc@tu-dresden.de))

**André Martin**

(Systems Engineering Group, Technical University of Dresden (TU Dresden), Dresden, Germany

 <https://orcid.org/0000-0002-7135-6288>, [andre.martin@tu-dresden.de](mailto:andre.martin@tu-dresden.de))

**Abstract:** Data-as-a-Service (DaaS) is a branch of cloud computing that supports “querying the Web”. Due to its ultrahigh scale, it is essential to establish rules when defining resources’ costs and guidelines for infrastructure investments. Those decisions should prioritize minimizing the incidence of agreement breaches that compromise the performance of cloud services and optimize resources’ usage and services’ cost. This article aims to address the cost problem of DaaS by developing a model that optimizes the costs of querying distributed data sources over virtual machines spread across multisite data centers. We have designed and analyzed a cost model for DaaS, besides implementing a scheduling system to perform a cost-based VM assignment. To validate our model, we have studied and characterized a real-world DaaS system’s network and processing workloads. On average, our cost-based scheduling performs at least twice as well as the traditional round-robin approach. Our model also supports load balancing and infrastructure scalability when combined with an adaptive cost scheme that prioritizes VM allocation within the underutilized data centers and avoids sending VMs to data centers in the eminence of becoming over-utilized.

**Keywords:** Cloud computing, Data-as-service, Cloud Cost Model, cost optimization, virtual machine scheduling

**Categories:** H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

**DOI:** 10.3897/jucs.99223

## 1 Introduction

During the last decade, we witnessed a continuous and increasing growth of live data sources, such as smartphone applications and sensors. Processing such data allows companies to understand their customers and behavioral patterns and trigger actions in relevant situations. Examples of those applications range from recommender systems and clickstream analysis to fraud detection systems.

In addition to active and live data sources, information can be obtained by continuously crawling the Web. New content is published and made available every second through various channels, such as blogs and Web portals.

Big companies such as *Google* and *Yahoo!* have access to passive data sources as their business models rely on a continuous crawling and processing of the Web. Small to medium-sized enterprises (SMEs), whose primary business is often outside of IT, cannot access such data. Acquiring passive information requires appropriate technology for crawling, storing, and processing the data and sufficient computational and storage resources to perform those tasks.

Although cloud computing enables practically anyone to acquire resources on the fly on a pay-as-you-go basis without high up-front investments, crawling the (“whole”) Web and performing data processing is still challenging for many SMEs. First, the companies do not have the necessary knowledge and technology for crawling, storing, and processing the data. Second, keeping a copy of the “whole” Web requires massive resources that a single SME cannot afford.

In the following, we consider an alternative paradigm called Data-as-a-Service (DaaS), in which multiple SMEs share the costs needed for the infrastructure to crawl and store a copy of the whole Web (or at least a significant part of it) and execute their business-specific queries on the given data set.

In addition to the DaaS paradigm, we consider an inhomogeneous micro-cloud model rather than a single dedicated data center for crawling, storing, and processing data. Micro-clouds consist of a few nodes, typically comprising only as little as 20 nodes with different storage and processing capabilities, usually interconnecting through asymmetric links. The advantage of micro-clouds is that they can work as part of the heating system in residential houses in the northern hemisphere, which allows green cloud computing as promoted by companies such as *Cloud & Heat* [CloudAndHeat 2020].

Although the micro-cloud model provides many energy and environment-friendly advantages, scheduling tasks such as crawling and data processing impose new challenges. First, as mentioned above, micro-clouds are usually interconnected through low-bandwidth links, which do not allow data shuffling on the TeraByte range. Second, micro-clouds expose different processing capacities and incur additional costs, e.g., depending on the heating needs in a residential house. Hence, this requires scheduling strategies different from scheduling algorithms used for homogeneous data centers.

This paper extends the scheduling algorithm and cost model proposed by Oliveira *et al.* [Oliveira et al. 2015]. This work performs a real-world experiment analysis of the DaaS and micro-cloud paradigm, where tasks (abstracted as virtual machines - VMs), such as crawling and processing, are scheduled across a set of micro-clouds. Our algorithm reduces the cost to less than 50% of the traditional round-robin task assignment approach.

The remainder of this paper is organized as follows. Related work is presented in Section 2. The proposed Data-as-a-Service cost model is described in Section 3. The scheduling strategies are defined in Section 4. The cost-based scheduling is validated in Section 5. Section 6 discusses final remarks and future work.

## 2 Related Work

We found many works that propose cost models for cloud computing services and VM scheduling. However, they do not approach cost models for DaaS and do not present VM processing experiments to model workload in real data centers. Besides, there is a lack of work in pricing schemes for Data-as-a-Service specifically.

There are research lines that propose optimal cost analysis for power consumption. Li *et al.* [Li et al. 2018] studied computing and cooling energy to minimize the total energy. Kantarci *et al.* [Kantarci et al., 2012] proposed an inter-and-intra data center VM placement for large-scale cloud systems to minimize power consumption. The model approach also applied Mixed Integer Linear Programming (MILP). Dong *et al.* [Dong et al. 2015] proposed a VM migration technique to minimize the maximum link utilization to improve the network performance. This scheme makes a tradeoff between energy efficiency and network performance.

Xu *et al.* [Xu et al. 2018] analyzed a multi-objective minimization problem of VM scheduling by optimizing the incentives for both client and provider parties. They strived to maximize the successful execution rate of VM requests, minimize the cloud user cost, and minimize the deviation of profit, which is the incentive for the cloud provider.

Niyato *et al.* [Niyato et al. 2016] categorized and presented data pricing models for the Internet of Things (IoT). Such work is valuable for comprehending state of the art on pricing models.

Dehsangi *et al.* [Dehsangi et al. 2015] developed cCluster that implemented a VM classification on the fly that acts in the VM scheduling to reduce I/O response time and improve network throughput.

Zaman and Grosu developed a combinatorial auction-based resource allocation mechanism for dynamic VM provisioning and allocation in clouds that intends to minimize the total cost of VMs under the constraint that a particular job  $J$  needs to be finished by the deadline  $D$  [Zaman and Grosu 2013].

Patel and Sarje proposed a VM provisioning method to minimize SLA violations and improve the profits of cloud service providers. They developed a threshold-based load balancing among federated clouds. They assumed VM cost models for on-demand and reserved instances. The results showed that the model might decrease the Service Level Agreement (SLA) violation factor while hindering resource allocation and load balancing among data centers [Patel and Sarje 2012].

Li *et al.* developed an adaptive algorithm to find the best allocation plan to maximize resource availability in Infrastructure-as-a-Service (IaaS) clouds while avoiding overutilization. They argue that those practices are mainly responsible for increasing the profit [Li et al. 2012]. Xiong *et al.* [Xiong et al. 2011] also handled the system overload problem. They developed the ActiveSLA, a profit-oriented admission control framework for Database-as-a-Service systems to minimize SLA breaches and maximize profit. In another work, Xiong *et al.* [Xiong et al. 2011b] solved the prior goals with a different approach: a cost-aware resource management system.

## 3 Data-as-a-Service Cost Model

A *cloud* may be represented as a set of *micro-clouds*, each one holding physical hosts for the execution of VMs or data storage. As a preliminary description, the term *micro-cloud* represents a small set of processing elements in the same local area network (LAN) without necessarily meaning the infrastructure of a complex data center. However,

without loss of generality, we will use the term *data center* throughout this paper, either with the same concept of micro-cloud or as a regular data center. We may represent different cloud configurations in one or several administrative domains, including private, public, or federated clouds. In DaaS, one *query* runs on top of one or more VMs, requiring access to one *data source* from the storage area located in the data centers.

We modeled one query (A.K.A. *job*) as one or more *tasks*. Each task needs one VM to execute and one data source for processing. All data sources reside in the storage area, replicated over the data centers. Data sources are persisted in a *key/value (K/V) store*.

The VM instance reads and processes the data, then sends the (partial or final) query results to the destination through the communication network. The VMs perform the queries' computation following the MapReduce programming style [Dean and Ghemawat 2008]. In this context, *Map* is performing some computation on a data set, and *Reduce* is reducing the mapped data to obtain the final result. The *MapReduce* model is widely applied to process large amounts of data in a distributed fashion. Big Data processing is a good representative for using the MapReduce model [Hurwitz et al. 2013]. Figure 1 illustrates the scheduling process and presents the challenge of which data center to allocate the VM.

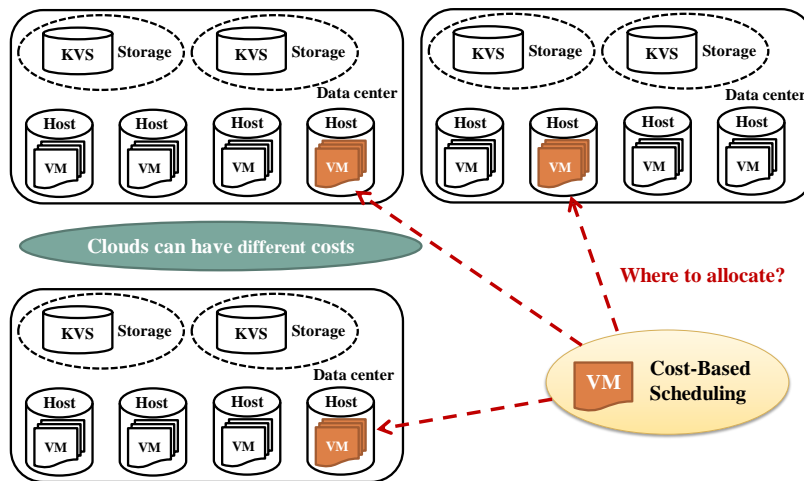


Figure 1: Scheduling decision process: Where to allocate the VM?

### 3.1 Cost Model

The DaaS cost model relies on two assumptions:

- i **VM and data are located in the same data center:** If the VM runs in the same data center that stores the required data source, i.e., in the same LAN, then there is

no network communication cost to read the data;

- ii **VM and data are located in different data centers:** If the VM does not run in the same data center storing the required data, it incurs network communication costs to read the input data.

The cost model components are defined as follows:

- Let  $D$  be the set of data centers:
  - $D = \{d_1, d_2, \dots, d_{|D|}\}$ ;
- Let  $U$  be the set of query VMs:
  - $U = \{u_1, u_2, \dots, u_{|U|}\}$ ;
- Let  $S$  be the set of data sources:
  - $S = \{s_1, s_2, \dots, s_{|S|}\}$ ;
- Let  $\vec{v}$  be the VM processing costs' vector for each data center in  $D$ :
  - $\vec{v} = \langle v_1, v_2, \dots, v_{|D|} \rangle$ ;
  - where  $v_i$  is the execution cost (i.e., CPU cost, considering the different speeds of heterogeneous resources) for a VM during 1 hour on data center  $d_i$ ,  $1 \leq i \leq |D|$ ;
- Let  $\vec{t}$  be the communication costs' vector for each data center in  $D$ :
  - $\vec{t} = \langle t_1, t_2, \dots, t_{|D|} \rangle$ ;
  - where  $t_i$  is the cost to transfer one gigabyte out of data center  $d_i$ ,  $1 \leq i \leq |D|$ .

Let us define the **independent variables** that model the query task that the VM  $u_z$  will execute,  $1 \leq z \leq |U|$ :

- $R_z$ : the number of gigabytes received by the query VM  $u_z$ ;
- $T_z$ : the number of gigabytes sent out of the query VM  $u_z$ ;
- $r_z$ : the reduction factor of VM  $u_z$ ; i.e., the ratio between  $T_z$  and  $R_z$ , where  $r_z = \frac{T_z}{R_z}$ ;
- $k_z$ : the time rate needed for processing one gigabyte received (input) within the query VM  $u_z$ , i.e., VM  $u_z$  needs  $k_z \cdot R_z$  hours to finish;
- $q_z$ : output rate (bytes per hour), where  $q_z = \frac{r_z}{k_z}$ .

We assume that the *cost to execute one query is the total cost composed by the VM execution cost* ( $Ce_z^i = k_z \cdot R_z \cdot v_i$ ) *plus the data communication costs* ( $Cd_z^i = T_z \cdot t_i$ ), taking into account the location of the VM and the data source.

Let the function  $C(u_z, s_z, d_i, d_j)$ , or simply  $C_z^{i,j}$ , be the VM  $u_z$  cost (**dependent variable**) of executing the VM  $u_z$  at data center  $d_i$  and retrieving data from data center  $d_j$ ; where  $1 \leq i, j \leq |D|$  and  $1 \leq z \leq |S|$ , and this cost is defined as follows (Eq. 1):

$$\begin{aligned}
C(u_z, s_z, d_i, d_j) &= C_z^{i,j} \\
&= Ce_z^i + Cd_z^j \\
&= (k_z \cdot R_z \cdot v_i) + (T_z \cdot t_j) \\
&= k_z \cdot R_z \cdot v_i + R_z \cdot r_z \cdot t_j \\
&= k_z \cdot R_z \cdot v_i + R_z \cdot (q_z \cdot k_z) \cdot t_j \\
C_z^{i,j} &= k_z \cdot R_z \cdot (v_i + q_z \cdot t_j) \tag{1}
\end{aligned}$$

The cost to run VM  $u_z$  in data center  $d_i$ , if it contains one copy of the required data source (i.e.,  $d_i = d_j$ ) and, thus, there is no data communication cost ( $Cd_z^i = 0$ ), is simplified according to Equation 2.

$$C(u_z, s_z, d_i, d_i) = C_z^{i,i} = Ce_z^i = k_z \cdot R_z \cdot v_i \tag{2}$$

Although prominent, the VM instance can only download a data source from the storage data centers. We have defined a function called *contains* that must be employed when checking whether accessing a given data source is possible. If the data center  $d_j$  contains the data source  $s_z$ , the function will return 1; otherwise, it returns 0. The function is defined in Equation 3.

$$contains(d_j, s_z) = \begin{cases} 1 & \text{if } s_z \text{ is stored in } d_j \\ 0 & \text{if } s_z \text{ is not stored in } d_j \end{cases} \tag{3}$$

We have summarized the cost function for the VM costs in Equation 4 and exemplified the overall scheduling cost process in Figure 2. The scheduling cost is infinite if the data center  $d_j$  has no replica of the data source  $u_z$ .

$$C(u_z, s_z, d_i, d_j) = \begin{cases} k_z \cdot R_z \cdot v_i & \text{if } d_i = d_j \wedge contains(d_j, s_z) = 1 \\ k_z \cdot R_z \cdot (v_i + q_z \cdot t_j) & \text{if } d_i \neq d_j \wedge contains(d_j, s_z) = 1 \\ \infty & \text{otherwise} \end{cases} \tag{4}$$

#### 4 Formalization of the Cost-based Scheduling Problem

The cost-based scheduling problem is a minimization problem, where we look to minimize the cost of scheduling the query VMs. We place the query VMs into a queue and assign them to execute and retrieve the data source from the data center pair, which minimizes the cost.

The cost to execute the query VM on the data center holding the data source is not necessarily below the cost when retrieving the data from the network. If the communication and VM costs are below the cost of scheduling the VM to data centers holding the data, the scheduling service will retrieve the data through the network. It is worth noting that the cheapest data center for running the VM will host the instance.

The problem of cost-based scheduling is to minimize the cost of allocating one VM individually (greedy scheduling), following the order that the VMs are in the scheduling queue. We generically represent the minimum cost to process the VM  $u_z$  by:

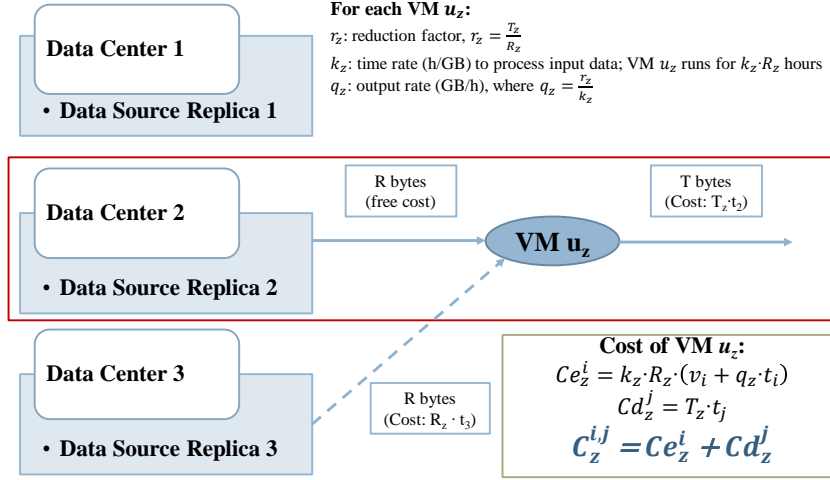


Figure 2: System model: Pricing scheme.

$$\min C_z = \min C_z^{i,j} = \min C(u_z, s_z, d_i, d_j); \forall d_i, d_j \in D$$

According to Equation 4, the cost of the VM  $u_z$  being hosted on data center  $d_i$ , and by retrieving data from data center  $d_j$  is:

$$C_z^{i,j} = t_j + v_i + q_z \cdot t_j \quad (5)$$

Mixed Integer Linear Programming (MILP) is an approach for solving scheduling problems. MILP is an application of mathematical programming (mathematical optimization). It optimizes a linear objective function, subject to linear equality and linear inequality constraints. A linear programming algorithm finds a point in the feasible region where the function has the smallest (or largest) value, if such a point exists, depending on the objective, whether it is minimizing or maximizing the objective function [Floudas and Lin 2005].

As discussed in previous sections, the scheduling problem we intend to solve has more assumptions than simply determining the minimum values for the coefficients that minimize the objective function. The locality of the data source replicas also impacts the final cost. Thus, this scheduling problem has more constraints than simple linear inequalities.

In the following sections, we will vectorize our variables and formalize the model using Linear Algebra.

#### 4.1 Definition of Constraints

Let us define the source matrix  $M$  of the query VM  $u_z$  to do this. It has  $|U|$  lines and  $|D|$  columns. The lines represent the query VMs, and the columns are the data centers that store the data sources required by the corresponding query. If the data center stores one copy of the required data, the value is set to  $\mathbf{1}$ ; otherwise, it is set to  $\mathbf{0}$ .

$$M_{|U| \cdot |D|} = \begin{bmatrix} 1 & 0 & \dots & 1 \\ 1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Given that, the vector that corresponds to all costs to execute the query VM  $u_z$  is defined as follows:

$$C_z = \vec{t}_z \cdot 1 + k_z \cdot (\vec{v} + q_z \cdot \vec{t})$$

Where:

- $\vec{v} = \langle v_1, v_2, \dots, v_{|D|} \rangle$  is the vector of VM execution costs for each data center in  $D$ ;
- $\vec{t} = \langle t_1, t_2, \dots, t_{|D|} \rangle$  is the vector of communication costs for each data center in  $D$ ;
- $M_z$  is the  $z$ -*esim* row of the source matrix (i.e., row describing which data centers contain a replica of the data source  $s_z$ );
- $\vec{t}_z = M_z \cdot \vec{t}$  (element-wise multiplication, for getting only the  $t$  costs from the data centers holding data replicas);
- Thus,  $t_z^j$  is the communication cost for retrieving the data source  $s_z$  from data center  $d_j$ ; it is set to  $\infty$  when the data is not stored on  $d_j$ .

We first define a matrix of known coefficients called  $\theta$  to solve the problem. This model has three lines (i.e., the number of products on the cost function) and  $D$  columns corresponding to the data centers. The  $\theta$  matrix, and its transpose matrix  $\theta^T$  are defined in Equations 6 and 7, respectively.

$$\theta = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ q_z & q_z & \dots & q_z \end{bmatrix}_{3 \cdot |D|} \tag{6}$$

$$\theta^T = \begin{bmatrix} 1 & 1 & q_z \\ 1 & 1 & q_z \\ \dots & \dots & \dots \\ 1 & 1 & q_z \end{bmatrix}_{|D| \cdot 3} \tag{7}$$

Before we delve into the definition of the cost matrix,  $C_z$ , of the query VM  $u_z$ , we will define another matrix,  $X$ . It contains the cost vectors of the data centers that are required to compute the overall cost. The  $X$  matrix is defined in Equation 8.

$$X = \begin{bmatrix} \vec{t}_z \\ \vec{v} \\ \vec{t} \end{bmatrix} = \begin{bmatrix} t_z^0 & t_z^1 & \dots & t_z^{|D|-1} \\ v_0 & v_1 & \dots & v_{|D|-1} \\ t_0 & t_1 & \dots & t_{|D|-1} \end{bmatrix}_{3 \cdot |D|} \tag{8}$$

#### 4.2 Minimization Problem

The cost matrix,  $C_z$ , of the query VM  $u_z$ , is finally defined in Equation 9. We attest that the product of the matrices  $\theta^T$  and  $X$  results in the cost matrix.



$$\begin{aligned}
C_z &= \theta^T \cdot X \\
C_z &= \begin{bmatrix} 1 & 1 & q_z \\ 1 & 1 & q_z \\ \dots & \dots & \dots \\ 1 & 1 & q_z \end{bmatrix} \cdot \begin{bmatrix} t_z^0 & t_z^1 & \dots & t_z^{|D|-1} \\ v_0 & v_1 & \dots & v_{|D|-1} \\ t_0 & t_1 & \dots & t_{|D|-1} \end{bmatrix} \\
&= \begin{bmatrix} t_z^0 + v_0 + q_z \cdot t_0 & t_z^0 + v_1 + q_z \cdot t_1 & \dots & t_z^0 + v_{|D|-1} + q_z \cdot t_{|D|-1} \\ t_z^1 + v_0 + q_z \cdot t_0 & t_z^1 + v_1 + q_z \cdot t_1 & \dots & t_z^1 + v_{|D|-1} + q_z \cdot t_{|D|-1} \\ \dots & \dots & \dots & \dots \\ t_z^{|D|-1} + v_0 + q_z \cdot t_0 & t_z^{|D|-1} + v_1 + q_z \cdot t_1 & \dots & t_z^{|D|-1} + v_{|D|-1} + q_z \cdot t_{|D|-1} \end{bmatrix} \quad (9)
\end{aligned}$$

On the other hand, the scheduling decision involves finding the minimum cost among the possible choices. Before applying the scheduling function, we set the costs of all non-possible options to infinite. We perform the scheduling decision over the cost matrix  $C_z$  using the *argmin* function available in all Linear Programming frameworks. We define how to represent the scheduling decision function in Equation 10.

$$\{i, j\} = \underset{1 \leq i, j \leq |D|}{\operatorname{argmin}} C_z \quad (10)$$

Where:

- *argmin* returns the coordinates' indices of the minimum value for the cost. We used the NumPy framework that implements this function [NumPy 2020];
- $i = \lfloor \operatorname{argmin}(C_z) \div |D| \rfloor$  (row);
- $j = \lfloor \operatorname{argmin}(C_z) \% |D| \rfloor$  (column).

### 4.3 Cost-based Scheduling Algorithm

The scheduling decision is the ordered pair of data centers  $\{d_i, d_j\}$  that minimize the cost function over all data centers in  $D$ , as stated in Equation 11. The scheduling mechanism would propose an empty set result if no data sources were available. Therefore, the VM could not run until the execution conditions were satisfied.

$$\operatorname{schedule}(u_z, s_z, D) = \begin{cases} \{d_i, d_j\} = \operatorname{argmin}_{d_i, d_j \in D} C(u_z, s_z, d_i, d_j) & \text{if } C(u_z, s_z, d_i, d_j) \neq \infty \\ \emptyset & \text{otherwise} \end{cases} \quad (11)$$

## 5 Validation

We shall specify the VM queries, and processing and communication costs to validate this proposed DaaS cost model. We analyzed how to characterize these parameters using actual queries and costs.

The data underpinning the analysis reported in this paper are deposited at a public repository [Oliveira 2023].

## 5.1 Workload Definition

We conducted actual DaaS experiments to characterize workload parameters for VM queries. We monitored and collected CPU usage metrics and network traffic traces from the VM queries' data received and sent out. We have also searched for processing and communication costs for cloud infrastructure services. We will explain those assignments in the following sections.

### 5.1.1 Query VM Characterization

We experimented with estimating the processing and network metrics. The experiment consisted of an application to *crawl the Web* using the MapReduce programming model for data processing. We used the Hadoop framework [Hadoop 2010] to implement the crawling software.

The mappers and reducers are in three micro clouds located in 3 German cities: Dresden, Münster, and Hamburg, as shown in Figure 3. The execution time lasted for 24.79 hours. The query VMs remained active throughout the experiment. During the execution of the experiment, we gathered memory, CPU, and network usage metrics.

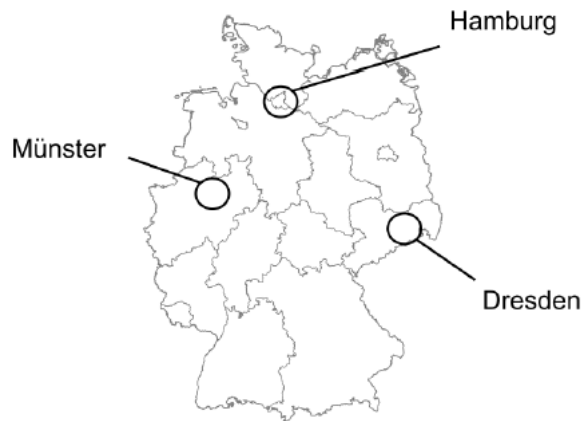


Figure 3: Distributed crawling [Quoc et al. 2015].

### 5.1.2 Network Traffic Characterization

Network traffic was collected using the *tcpdump* tool. Each VM's received and sent traffic was processed, and we graphically plotted the time series of the throughput of each VM. In Figure 4, the incoming and outgoing traffic throughput was grouped by the hour to provide more understandable information about the execution of virtual machines.

VM N<sup>o</sup> 1 ( $u_1$ ) has an outgoing traffic volume higher than the initial input. They become equivalents after the 10<sup>th</sup> hour. This is also the moment when both directions of traffic have their peak hours, with about 20 GB for the input volume and 30 GB for the output volume. VM N<sup>o</sup> 2 ( $u_2$ ) has an input volume higher than that of output during

practically all its execution. With a maximum input volume of almost 20 GB in 10<sup>th</sup> hour and 8.65 GB in hour 17<sup>th</sup>. The VM N<sup>o</sup> 3 ( $u_3$ ) has comparable volumes for incoming and outgoing traffic. The maximum input volume was 17.41 GB at 8<sup>th</sup> hour and 17.82 GB in 10<sup>th</sup>.

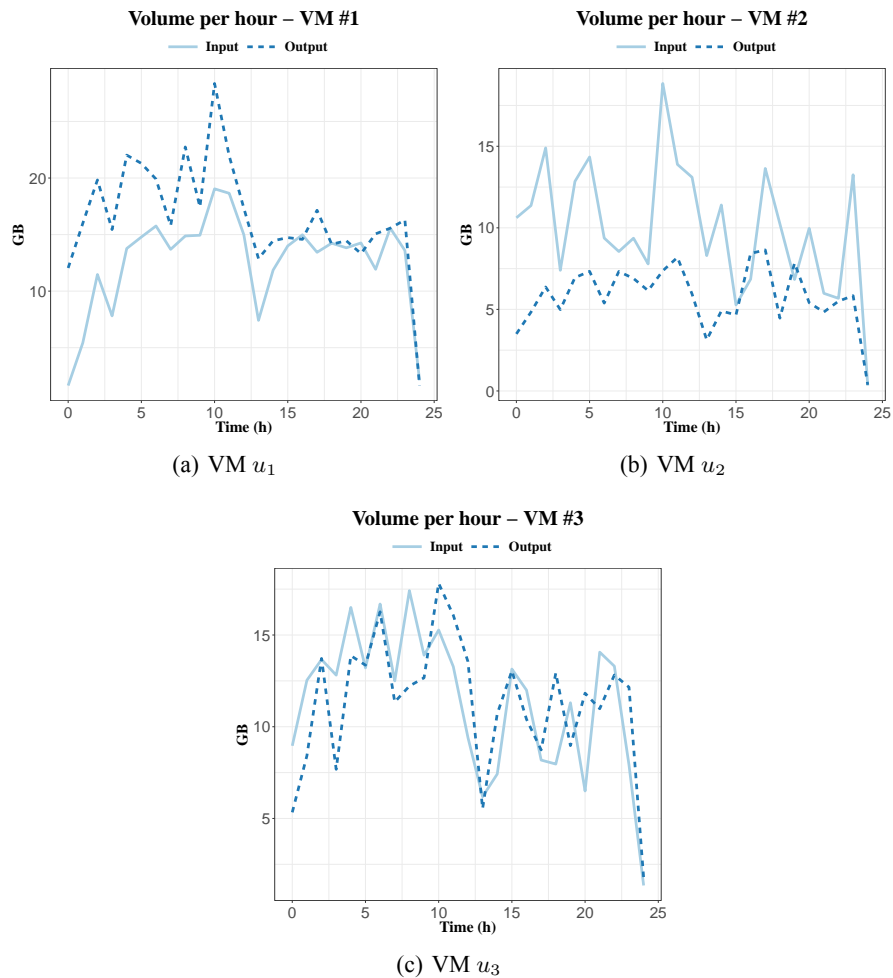


Figure 4: Input and output network volume per hour for each VM.

The variables  $R$  and  $T$  of the VMs are related to the network traffic behavior. In this sense, we have derived them from this DaaS application experiment. Figure 5 presents hourly estimates for the variables  $R$  and  $T$  for VMs  $u_1$ ,  $u_2$  and  $u_3$ . We used those values to validate the cost-based scheduling model afterward.

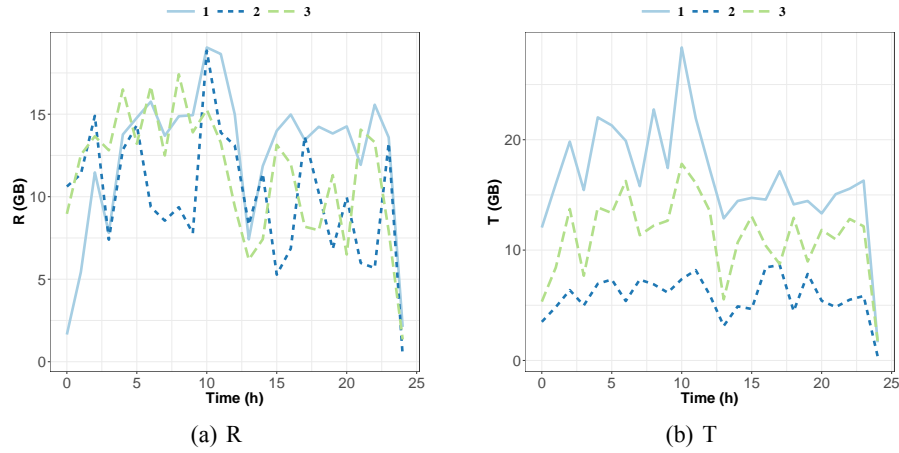


Figure 5: Estimated  $R$  and  $T$  values for the 3 VMs.

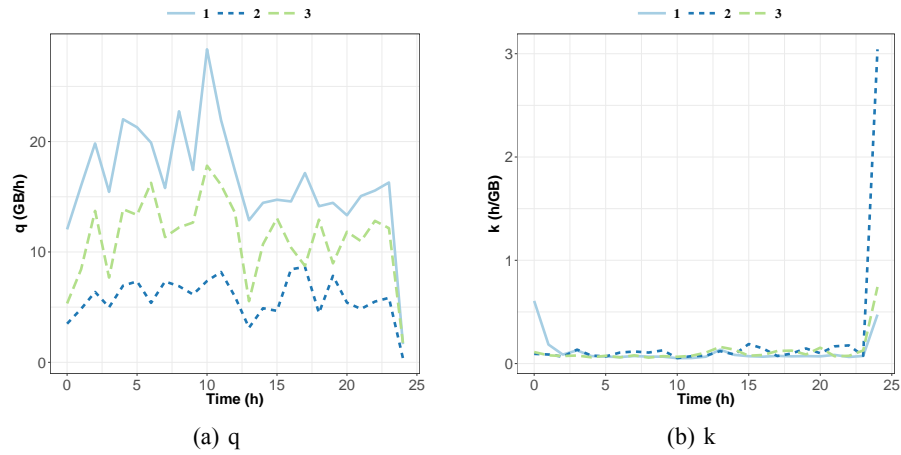


Figure 6: Estimated  $q$  and  $k$  values for the 3 VMs.

### 5.1.3 Data Processing Characterization

The variables  $q$  and  $k$  are derived from the collected processing metrics. We compute them after the measurements' analysis. Figure 6 shows each experiment hour's analytical distribution of the variables  $q$  and  $k$  for VMs  $u_1$ ,  $u_2$ , and  $u_3$  by each experiment hour.

### 5.1.4 Processing and Communication Cost Characterization

We set up the initial processing and communication costs according to the infrastructure costs applied by the Cloud & Heat [CloudAndHeat 2020] drop provider.

The chosen VM instances have a medium size (M). They contain four virtual CPUs, a virtual hard drive of 120 GB, 4 GB of RAM, and 4 TB of traffic, being charged € 0.08 per hour for each VM instance and € 0.02 per hour for each GB that exceeds this network quota. However, to simplify the analysis, such traffic in excess will not be charged for this assessment.

Three data centers will compose the cost model evaluation. The  $v$  and  $t$  costs to run a query VM in these data centers for 1h are presented in Table 1.

DC ID	Processing Cost Vector $\vec{v}$ (€)	Network Cost Vector $\vec{t}$ (€)
$d_1$	$v_1 = 0.09$	$t_1 = 0.02$
$d_2$	$v_2 = 0.1$	$t_2 = 0.03$
$d_3$	$v_3 = 0.07$	$t_3 = 0.05$

Table 1: Processing and Communication Costs for each Data Denter in  $|D|$ .

### 5.1.5 Workload Summary

Table 2 shows the variables calculated by the total time of the experiment for the sum of the sent ( $T$ ) and received ( $R$ ) amount of traffic, and the mean values of  $q$  and  $k$  for the VMs  $u_1$ ,  $u_2$  and  $u_3$ .

VM ID	Sum of R (GB)	Sum of T (GB)	Time (h)	Mean k (h/GB)	Mean q (GB/h)
$u_1$	314.09	414.44	24.79	0.078917381	16.71971049
$u_2$	250.15	145.29	24.79	0.099095321	5.861165521
$u_3$	285.49	282.21	24.79	0.086834007	11.38412853

Table 2: Sum of network received ( $R$ ) and sent ( $T$ ) volumes; and average processing values of  $q$  and  $k$  for each VM during the period of the experiment.

## 5.2 Scheduling Strategies

We have compared the proposed cost-based scheduling for DaaS with the traditional round-robin strategy. We will define them in the following sections.

### 5.2.1 Round Robin Strategy

This strategy selects one pair of data centers that satisfies the conditions to execute the query VM in a round-robin fashion. The pool of resources is randomly organized.

### 5.2.2 Cost-based Strategy

The cost-based strategy decides on the pair of data centers that minimizes the cost to schedule the VM from the available data centers at the scheduling moment. It is relevant to emphasize that a pair of data centers may not represent the minimal cost among the initial set of data centers because, at some stage of the scheduling, the cheapest processing data center could be at total capacity (e.g., there is a lack of computing resources), or the data center that hosts a copy of the data source with minimal network cost is not available to data retrieval. In this case, the cost-based scheduling will propose the second cheapest cost and decide on another pair that satisfies the query VM assumptions.

### 5.3 Cost-based Scheduling Performance Evaluation

We have used R Programming Language and additional packages to implement the VM cost-based scheduling, to process the data, analyze the results, and generate the graphics [RProject 2021] [Wickham 2016] [Warnes et al. 2022]. We have also studied two Python frameworks for Linear Programming to implement the VM cost-based scheduling: Numpy [NumPy 2020] and SciPy [SciPy 2020].

We initially performed different experimental treatments to identify the effect of the factors in the final cost; however, we realized that the most significant impact (over 5% of significance) affecting results is the number of VMs that can be executed per data center. Then, we shaped our analysis and further investigated two experimental treatments.

In the first experiment treatment, one data center may process at most 20% of the scheduled queries; in the second, one data center can host up to 100% of the queries.

Suppose the data center that has the optimal cost is not available to allocate the query VM. The query VM will be allocated to the data center with minimal cost among the available set. The evaluated number of VMs per data center (DC) is shown in Table 3.

Nr.	Parameter	Factor Level
1	Number of VMs processed per DC	8
2	Number of VMs processed per DC	40

Table 3: Analyzed experiment treatments.

We have attributed to every data center two costs ( $v$  and  $t$ ) assigned according to the workload definition. To every query VM, we have also assigned two rates ( $q$  and  $k$ ) from within an analytical distribution. Every data source was replicated using a replication factor.

The number of data centers, queries, and sources was constant throughout the experiment. Those configuration variables are summarized in Table 4.

#### 5.3.1 Analysed Metric

We have analyzed a metric called **normalized cost**. It is a ratio between the cost obtained with the scheduling approach and the **optimal cost**.

Nr.	Parameter	Configuration
1	v (cpu cost)	See Section 5.1.4
2	t (bandwidth cost)	See Section 5.1.4
3	q	See Section 5.1.3
4	k	See Section 5.1.3
5	$ D  =  U  =  S $	3
6	Replication factor (data randomly placed into DCs)	2
7	Number of Query VMs	40

Table 4: Experiment plan.

The optimal cost to schedule the query VM  $u_z$  is the cost obtained when the pair of data centers that minimizes the cost to schedule the VM is chosen by the scheduler from the available set of data centers  $|D|$ .

Let  $C_D^{*z}$  be the optimal cost for running the query VM  $u_z$  on data source  $s_z$ , given the set of data centers  $D$ . The scheduling decision takes an ordered pair  $\{d_i, d_j\}$ , respectively, comprising the data center hosting the VM and the one for retrieving the data source. It minimizes the VM scheduling costs among all data centers belonging to the set  $D$ .

Note that the data centers with available resources to schedule a VM will vary according to the number of VMs running on them, their VM capacity, and if the chosen data center that hosts a copy of the data source is available to respond to the demand.

It is relevant to emphasize that if the pair of data centers that minimizes the cost is unavailable for hosting a new VM at the moment of the scheduling (e.g., there is a lack of computing resources), we will not name the optimal choice in this data analysis. Employing comparison, we will also present the optimal results as if we could choose the data centers without any contention of resources.

We analyzed the optimal resulting costs to verify the effectiveness of the cost-based strategy in choosing another pair of data centers when the cheapest pair is not available to attend the current query VM. It may also indicate the need to scale the infrastructure.

We formally define the normalized cost and the optimal cost as follows.

- **Normalized Cost:**  $NC(z)$

The normalized cost of the query VM  $u_z$  is defined as:

$$NC(z, e, r) = \frac{C_z^{i,j}}{C_D^{*z}} \quad (12)$$

Where:

- $d_i$  and  $d_j$  are the scheduler's decisions;
  - $C_D^{*z} = \min C_z^{i,j}; \forall d_i, d_j \in D$ .
- **Optimal Normalized Cost:**  $NC^*(z)$

The optimal normalized cost objective is 1, as stated in Equation 13. Even though it is only possible for cases when there is no contention of resources, i.e., if the cheapest DC is available when demanded. We analyzed this metric to check how often those events occur and whether significant losses justify oversizing the infrastructure.

$$NC^*(z) = NC(z, e, r) \rightarrow 1; \forall z, e, r \quad (13)$$

### 5.3.2 Discussion of the Results

The results showed no statistical difference between the cost-based and the optimal scheduling strategy. As a result, the cost-based scheduling resulted in the optimal choices for the VM scheduling in all scenarios under consideration. This improvement tends to increase when the processing time rate decreases ( $k$ ), which is acceptable and encouraging once the capabilities of the resources also increase. In contrast, new large-scale processing models are necessary.

For the first eight VMs scheduled on experiment Treatment N<sup>o</sup>1, the normalized cost is one because each DC supports hosting the processing of up to 20% of the query VMs (e.g., eight queries simultaneously scheduled from a demand of 40). Those results are presented in Figure 7. Note that it considers only processing conditions. It does not consider the data source transfer from the network.

Even when resource contention is present, the cost-based scheduling strategy performed, on average, three times better than round-robin. Besides, we confirmed statistically that the cost-based scheduling performed similarly to the optimal decisions even in cases of resource contention.

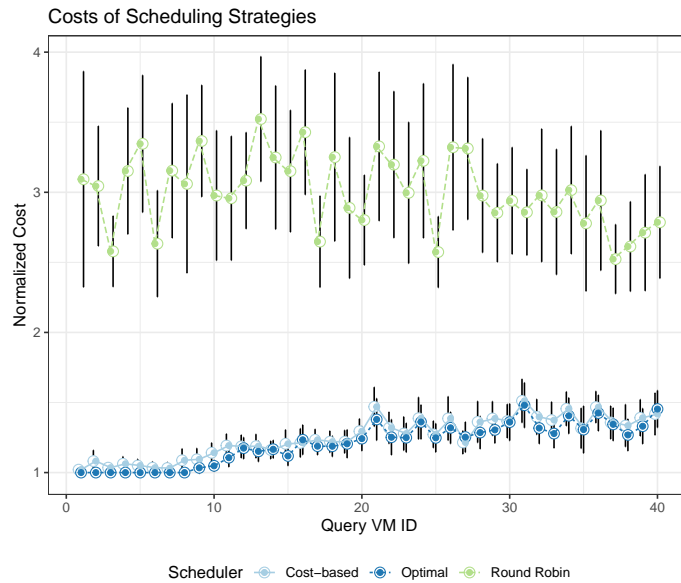


Figure 7: Obtained costs for the scheduling strategies when 20 % of the VMs may be hosted per data center.



In Figure 8, we present the results from experiment Treatment N<sup>o</sup>2, where every data center can host the processing of up to 100 % of the query VMs. We do not have to allocate VMs to non-optimal-cost data centers in this case. Consequently, the normalized optimal cost is 1 for all scheduled VMs. The cost-based strategy was nearly 2.5 to 4 times cheaper than the round-robin approach, including the confidence intervals for the variability of the results.

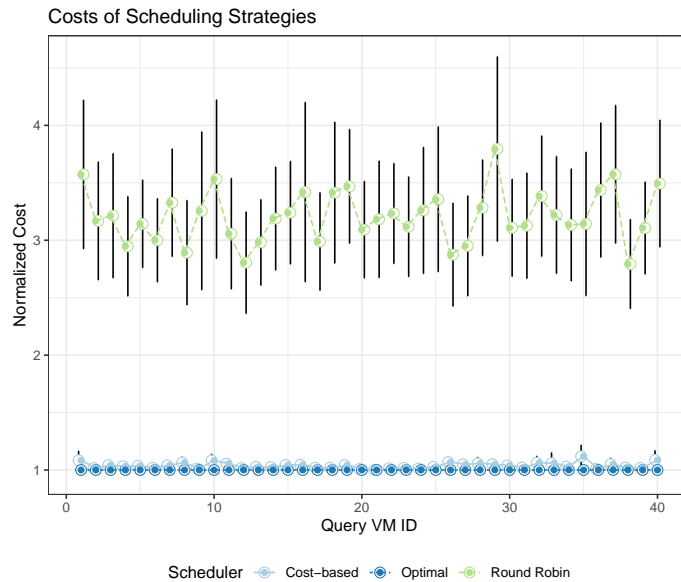


Figure 8: Obtained costs for the scheduling when up to 100% of the VMs may be hosted per data center.

The large variability of the round-robin strategy is that it eventually decides on the optimal cost; however, it presents the same probability of choosing the most expensive data center to allocate the query. The cumulative distribution function of the mean costs charged per scheduler strategy used to specify multivariate random variables' distribution shows that the cost-based strategies have correlated and close growth curves. In contrast, the round-robin costs grow more smoothly, presenting higher mean values. The CDFs for both experiment treatments are plotted in Figures 9 and 10.

We also concluded that there is no significant cost difference between the cost-based scheduling and the analyzed optimal scheduling.

Complementary, we studied the outliers present in the results. The outliers contributed to the variation in the cost values. Since there is a random assignment for four variables, their combination may lead to a broader time range and processing of consuming queries. We may identify those using boxplots (the line inside the box indicates the median value). The mean and median in our experiment have practically the same value. The median for the normalized cost for the cost-based strategy is around 1, and the round-robin median is approximately 3. The results are shown in Figure 11.

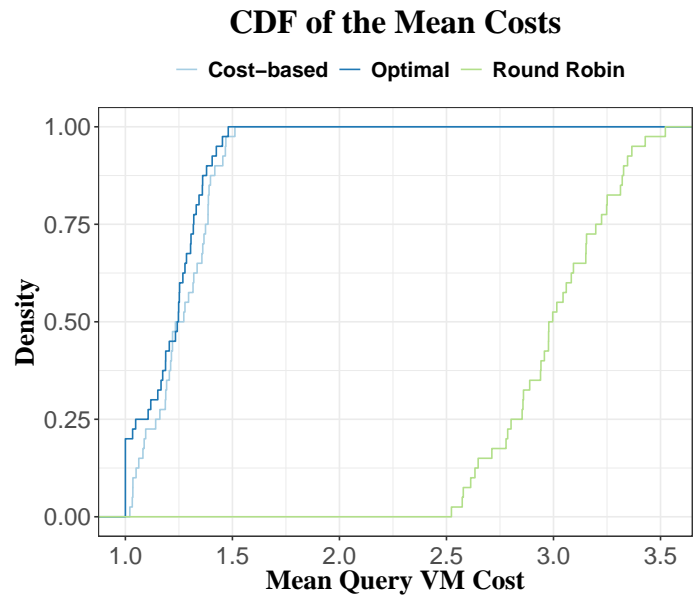


Figure 9: CDF for the scheduling strategies when up to 20 % of the VMs.

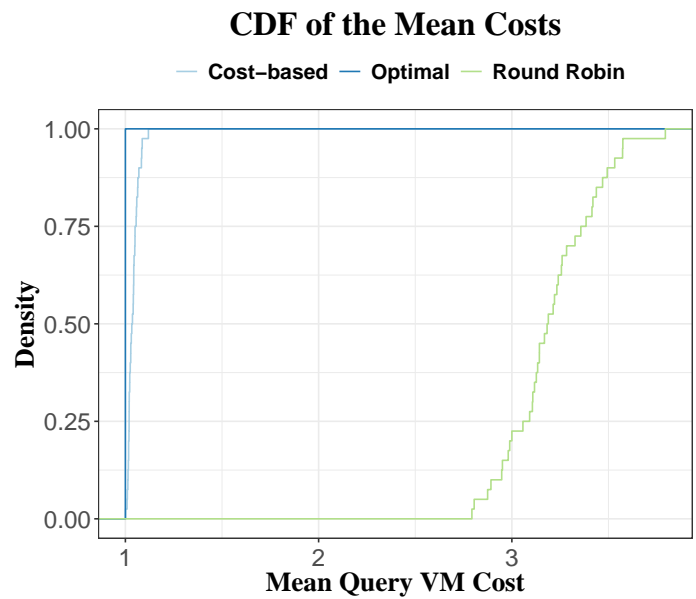


Figure 10: CDF for the scheduling strategies when up to 100 % of the VMs.

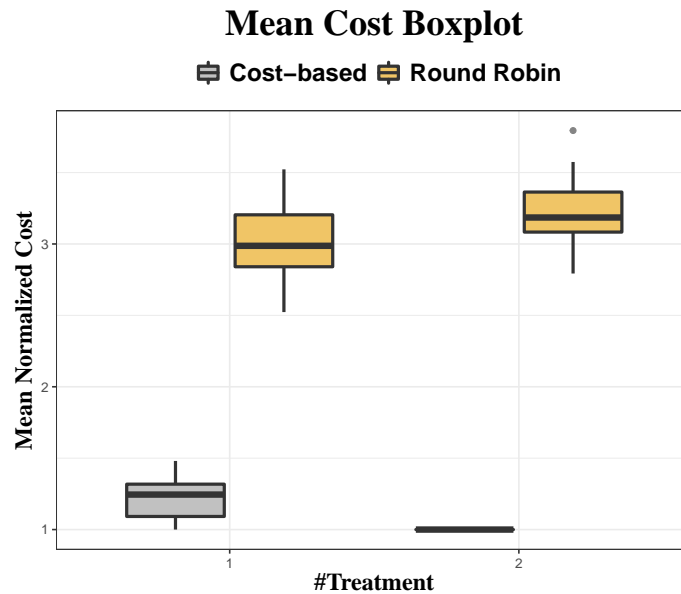


Figure 11: Boxplot of the query VM normalized costs.

## 6 Conclusion

We have designed and analyzed a cost model for DaaS, besides implementing a scheduling system to perform a cost-based VM assignment. The final costs obtained for the queries using the cost-based scheduling performed at least twice as well, on average, than the traditional round-robin approach.

The proposed cost-based scheduling model may also support load balancing and infrastructure scalability when combined with an adaptive cost scheme that prioritizes VM allocation within the underutilized data centers and avoids sending VMs to data centers in the eminence of becoming over-utilized. Such an approach is possible with simple changes in the cost model to adjust the data center costs based on the current allocation state.

In future work, we intend to benchmark the cost-based scheduling model with adaptive schemes, submit the system to SLA constraints, and attribute penalties for non-compliant agreements.

## Acknowledgements

This work was supported by the LEADS project funded by the European Commission under the Seventh Framework Program (grant 318809). This work was also supported by the TruEGrid project funded by the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES), the German Academic Exchange Service (DAAD), the German Society for International Cooperation (GIZ) under the New Partnerships (NoPa) program.

## References

- [CloudAndHeat 2020] Cloud&Heat, Cloud & heat iaas pricing (2015). <https://www.cloudandheat.com/en/pricing-iaas.html>
- [Dean and Ghemawat 2008] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, *Commun. ACM* 51 (1) (2008) 107–113. doi:10.1145/1327452.1327492. <https://doi.org/10.1145/1327452.1327492>
- [Dehsangi et al. 2015] M. Dehsangi, E. Asyabi, M. Sharifi, S. V. Azhari, ccluster: A core clustering mechanism for workload-aware virtual machine scheduling, in: 2015 3rd International Conference on Future Internet of Things and Cloud, 2015, pp. 248–255. doi:10.1109/FiCloud.2015.56.
- [Dong et al. 2015] J. Dong, H. Wang, S. Cheng, Energy-performance tradeoffs in iaas cloud with virtual machine scheduling, *China Communications* 12 (2) (2015) 155–166. doi:10.1109/CC.2015.7084410.
- [Floudas and Lin 2005] C. A. Floudas, X. Lin, Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications, *Annals of the Operations Research* 139 (2005) 131–162. doi:10.1007/s10479-005-3446-x. <https://doi.org/10.1007/s10479-005-3446-x>
- [Hadoop 2010] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1–10. doi:10.1109/MSST.2010.5496972.
- [Hurwitz et al. 2013] J. Hurwitz, A. Nugent, F. Halper, M. Kaufman, *Big data for dummies*, John Wiley & Sons, Indianapolis, 2013.
- [Kantarci et al., 2012] B. Kantarci, L. Foschini, A. Corradi, H.T. Mouftah, Inter-and-intra data center VM-placement for energy-efficient large-Scale cloud systems, in: 2012 IEEE Globecom Workshops, pp. 708–713.
- [Li et al. 2012] J. Li, Q. Wang, D. Jayasinghe, S. Malkowski, P. Xiong, C. Pu, Y. Kanemasa, M. Kawaba, Profit-based experimental analysis of iaas cloud performance: Impact of software resource allocation, in: Proceedings of the 2012 IEEE Ninth International Conference on Services Computing, SCC '12, IEEE Computer Society, USA, 2012, p. 344–351. doi:10.1109/SCC.2012.85. <https://doi.org/10.1109/SCC.2012.85>
- [Li et al. 2018] X. Li, P. Garraghan, X. Jiang, Z. Wu, J. Xu, Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy, *IEEE Transactions on Parallel and Distributed Systems* 29 (6) (2018) 1317–1331. doi:10.1109/TPDS.2017.2688445.
- [Niyato et al. 2016] D. Niyato, D. T. Hoang, N. C. Luong, P. Wang, D. I. Kim, Z. Han, Smart data pricing models for the internet of things: a bundling strategy approach, *IEEE Network* 30 (2) (2016) 18–25.
- [NumPy 2020] NumPy, Numpy web site (2023).<https://numpy.org/>
- [Oliveira et al. 2015] A. C. Oliveira, C. Fetzer, A. Martin, D. L. Quoc, M. Spohn, Optimizing query prices for data-as-a-service, in: 2015 IEEE International Congress on Big Data, 2015, pp. 289–296. doi:10.1109/BigDataCongress.2015.48.
- [Oliveira 2023] A. C. Oliveira, Cost-based virtual machine scheduling for data-as-a-service, *Tech. rep.*, Mendeley Data Repository (2023). doi:10.17632/47c6p6vhyr.1.
- [Patel and Sarje 2012] K. Patel, A. Sarje, Vm provisioning method to improve the profit and sla violation of cloud service providers, 2012, pp. 1–5. doi:10.1109/CCEM.2012.6354623.
- [Quoc et al. 2015] D. L. Quoc, C. Fetzer, P. Fellber, É. Rivière, V. Schiavoni, P. Sutra, Unicrawl: A practical geographically distributed web crawler, in: 8th IEEE International Conference on Cloud Computing (CLOUD'15), IEEE Computer Society, 2015.

- [RProject 2021] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2021). <https://www.R-project.org/>
- [SciPy 2020] SciPy, Scipy web site (2023). <http://scipy.org>
- [Warnes et al. 2022] G. R. Warnes, B. Bolker, L. Bonebakker, W. H. Robert Gentleman, A. Liaw, T. Lumley, M. Maechler, A. Magnusson, S. Moeller, M. Schwartz, B. Venables, T. Galili, Various R Programming Tools for Plotting Data, R Foundation for Statistical Computing (October 2022). <https://cran.r-project.org/web/packages/ggplots/ggplots.pdf>
- [Wickham 2016] H. Wickham, ggplot2: Elegant Graphics for Data Analysis, Springer-Verlag New York, 2016. <https://ggplot2.tidyverse.org>
- [Xiong et al. 2011] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, H. Hacigümüş, Intelligent management of virtualized resources for database systems in cloud environment, in: 2011 IEEE 27th International Conference on Data Engineering, 2011, pp. 87–98.
- [Xiong et al. 2011b] Xiong, Pengcheng, Chi, Yun, Zhu, Shenghuo, Tatemura, Junichi, Pu, Calton, Hacigümüş, Hakan, ActiveSLA: A Profit-Oriented Admission Control Framework for Database-as-a-Service Providers, in: Proceedings of the 2nd ACM Symposium on Cloud Computing, 2011, <https://doi.org/10.1145/2038916.2038931>.
- [Xu et al. 2018] H. Xu, Y. Liu, W. Wei, W. Zhang, Incentive-aware virtual machine scheduling in cloud computing, J Supercomput (74) (2018) 3016–3038. doi:<https://doi.org/10.1007/s11227-018-2349-y>.
- [Zaman and Grosu 2013] S. Zaman, D. Grosu, A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds, IEEE Transactions on Cloud Computing 1 (2) (2013) 129–141.