International Journal of Computer Applications (0975 - 8887)
Volume 175 - No.32, November 2020

A Cloud Service Broker for Cost Effective Infrastructure Selection using Multiple Deployment Options

Raphael Gomes
Instituto Federal de Goiás
Goiânia - GO, Brazil

Geovany Rodrigues
Instituto Federal de Goiás
Goiânia - GO, Brazil

Gilberto Lobo
Instituto Federal de Goiás
Goiânia - GO, Brazil

ABSTRACT

The multiplicity of cloud service providers and the wide variety of resources types and regions makes selecting services a challenging task, which becomes even more complex when considering different cloud deployment models to meet the applications' specifics that will use these resources. For this, among the criteria used in selecting cloud resources, the cost is rated one of the most essential. Given this, this paper presents a cloud service broker's design and implementation for resource selection, taking into account different options, including the variation of cloud service providers, regions, and cloud deployment models. The proposed tool is based on other contributions that are also described in this work: 1) the design and construction of an ontology with concepts on the representation of computing resources, with associated reasoning processes; and 2) an on-premises infrastructure cost estimation strategy using a Total Cost of Ownership analysis. A qualitative evaluation considering productivity and accuracy is also presented, demonstrating the advantages of the proposed tool over other existing options.

General Terms

Cloud Computing, Resource Selection, Service Broker

Keywords

Cloud service selection, cost optimization, deployment models, ontology, TCO

1. INTRODUCTION

Nowadays, the use of Cloud Computing environments to serve infrastructure needs is a widely adopted strategy. Estimates indicate that 2/3 of Infrastructure as a Service (IaaS) spend goes toward compute [28]. In this scenario, cloud service discovery, i.e., the process of finding a Cloud Service Provider (CSP) who can best satisfy a consumer's needs, is carried out with cost optimization as one of the main focuses. However, the adoption of a single deployment option for this task is a limiting strategy. As an example, for some applications, the location of deployment is not important. For instance, in training neural network models, the geo-distribution of datacenters can be exploited as a strategy for allocating computational power at a lower cost. Furthermore, using another deployment model can also be an interesting strategy because, for specific scenarios, it can be more economically advantageous to create its own infrastructure instead of adopting public cloud environments. Other scenarios may require an on-premises solution such as those where legal constraints prevent public clouds.

Cloud service discovery taking only public clouds is already a challenging task since the market for cloud services is overwhelmed with a high number of heterogeneous cloud offerings, derived from the number of CSPs and the multiple regions in which they act. In addition to the wide variety of equivalent services, there is also the fact that each CSP adopts a specific nomenclature on the services it offers, given that due to commercial competition, there is no initiative on the part of them to create standardization [4]. Consequently, equivalent cloud services can be represented by different expressions depending on the provider, leaving the customer with the task of distinguishing and associating one service to another, using their specifications and descriptions. The use of general-purpose search tools is not an effective strategy to deal with this challenge as they are based only on the use of keywords and match between terms. The entire semantic and domain spectrum of the query is lost when weighing and selecting the results. Furthermore, considering that each CSP may expose their unique Application Program Interface (API), designing and developing an application to be deployed on a specific CSP does little to mitigate the development efforts to deploy the application on the cloud [11].

The use of Cloud Service Broker (CSB) solutions are commonly proposed as the most promising alternative in cloud service discovery [11, 13]. A CSB can be defined as a service that acts on behalf of a client to provide resources, including automatic resource provisioning across multiple clouds [13]. The use of a CSB makes it possible to deal with heterogeneity aspects. However, most existing solutions focus more on a wider range of functional and non-functional aspects [7, 33] rather than economical. Though, what makes a CSB eminent among others is a mechanism for optimizing customer value for money. Besides, some of these solutions do not offer enough abstraction to handle multiple CSPs [23] or are specific to the public cloud, not allowing analysis with other deployment models.

In view of that, this paper describes a CSB tool for cloud service discovery targeting cost reduction of computing resources on using different deployment options. The tool allows exploring the multiplicity of resource types and region of deployment. To this end, an ontology is proposed that aims to standardize terms related to infrastructure and, thus, address the highlighted semantic problems. Also, the tool allows the analysis of different deployment models by relying on an on-premises cost estimation strategy based on the

Total Cost of Ownership (TCO) analysis. The purpose of this CSB is not to optimize cloud service discovery at runtime, but rather to offer a tool that supports the search for resources with a given hardware configuration. Another goal is to offer a component that can be used in the construction of more complex solutions. Experiments demonstrate that the proposed tool offers a higher level of productivity and accuracy than existing alternatives, allowing more significant results in less time.
The remaining of this paper is organized as follows. Related work is discussed in Section 2. Section 3 presents the proposed ontology while the methodology for estimating on-premises infrastructure cost is discussed in Section 4. Based on these contributions, the tool architecture is presented in Section 5. The implementation of a prototype of the architecture is described in Section 6, and its qualitative evaluation, through an experiment, is discussed in Section 7. Final remarks are presented in Section 8.

2. RELATED WORK

The foundations necessary for developing the solution proposed here are also dealt with in other work, but generally with different goals. More precisely, the analysis of the cost of clouds through TCO and the use of ontologies to represent cloud environments are the object of study in several works, among which can be cited, respectively, [7, 14, 22, 29] and [9, 17, 18]. However, concerning the first subject, TCO analysis has the sole objective of evaluating environments on-premises, without considering other cloud deployment models or multiple CSPs. Besides, the ontologies' proposition aims mainly to allow interoperability between CSPs without having cost optimization as a focus.
Among the work that most resemble the proposal presented here, Wang et al. [32] proposed the brokerage service to minimize the cost by exploiting different pricing offers of IaaS clouds to get maximum benefits from various offers. These resources are served to cloud customers as per their demands at a reduced cost. However, the approach is a dynamic solution to which, not in all cases, service cost can be reduced.
A semantic matchmaking algorithm was proposed by Modica and Tomarchio [21] to search cloud services that best meet the customer requirements. Experimental results show that semantic technologies can enhance the performance of supply-demand matchmaking. Nevertheless, cost optimization is not the main goal.
STRATOS [24] provides automated decision making for resource acquisition between different CSPs based on two steps: determining the number of resources required, and determining where to place the resources. One of the broker's goals is optimizing costs, which is expressed by taking the prices charged for different services into consideration. However, this optimization is performed by ignoring geographic datacenter placement.
Similarly, various cloud service discovery tools have been developed in the industry, such as Cloudorado Cloud Server Comparison [8] and RankCloudz [25]. However, these tools do not support multiple deployment models. Moreover, the RankCloudz selection process requires customers to determine the importance of a set of attributes, only for comparing CSPs and not performing cloud service discovery. In this same category, some CSPs themselves offer tools that assist in this task targeting cost optimization, with on-premises cost analysis. Examples are AWS Pricing Calculator [3], Azure TCO Calculator [20], and Google Pricing Calculator [12]. However, queries only consider resources from the provider itself.
Despite the similarities with some existing solutions, the tool proposed here aims to fill some gaps in these approaches' consultation process. The main outcome is that it allows a more precise analysis of the cost of resources when considering multiple CSPs, regions, and deployment models. With this, it is expected that the proposed tool allows us to offer more accurate insights into the costs related to the allocation of IaaS.

3. DRIVING AN UNIFIED IAAS NOMENCLATURE

Little has been done to tackle interoperability challenges in cloud computing environments. This is evidenced by the sparse adoption of mapping and translation libraries to abstract away differences between CSPs. Also, many frameworks assume that they will adopt a common API to collaborate to satisfy customer's requirements. In reality, the cloud market has demonstrated its resistance to allowing customers to move between competitors freely, and the vast majority of the literature assumes that data published by CSPs is comparable [11]. Further, some approaches assume that the CSPs will adopt standard offering descriptions (e.g., Open Cloud Computing Interface (OCCI) [10]), which is also unrealistic. The most promising solution widely discussed in the literature is the use of ontologies to handle these limitations.
As a first step in building an ontology-based solution, the main contributions related to compute services with a focus on cost optimization were analyzed. To this end, the analysis relies on the studies of Tankelevicienea and Damaseviciusb [30], where an evaluation framework was presented to be used during the process of building ontologies. On top of that, an assessment was carried out based on the following criteria:

— ***(I) Completeness:*** how well the ontology captures the domain knowledge.
— ***(II) Consistency:*** how free of contradictories and overlaps is the ontology.
— ***(III) Conciseness:*** how free of unnecessary and redundant information or details is the ontology.
— ***(IV) Preciseness:*** ontology's richness level.
— ***(V) Clarity:*** how well the ontology is clear and understandable for domain knowledge experts.

In the process, the contributions presented by Al-Sayed et al. [1] are reevaluated and extended. The evaluation had as scope only the aspects related to the goals of the proposed CSB so that other features of the ontologies are disregarded. Table 1 presents the taken solutions, as well as the satisfaction of the established criteria.
As shown in the Table 1, despite the wide applicability of the existing ontologies, the abstraction proposed by them is not suitable to adequately represent computing resources. Especially when considering the completeness criterion, only the mOSAIC and Han and Sim ontologies represent hardware components in the necessary granularity to develop the proposed CSB. Another important criterion, preciseness, is not adequately addressed. In reality, considering the target problem, no ontology completely meets all the criteria. Therefore, a new ontology is proposed, named ***Cloud Infrastructure Service Ontology (CISO)***, in order to overcome the limitations identified in the existing solutions.
The CISO has been developed in the Ontology Web Language (OWL) language by using Protégé 5.5[1]. As shown in Fig. 1, its structure includes classes that represent the highest level concepts such as Communication as a Service (CaaS), Compute Resource, etc. The meaning of each class is intuitive and can be extracted from the figure itself. It is important to notice that the ontology includes

[1]`https://protege.stanford.edu/`

Table 1. Ontologies and evaluation criteria

Ontology	***(I)***	***(II)***	***(III)***	***(IV)***	***(V)***
IBM [6]		✓	✓		
NIST [15]		✓	✓		
Alfazi et al. [2]		✓			
Lenk et al. [16]		✓		✓	✓
mOSAIC [22]	✓			✓	
CL-Ontology [7]		✓			✓
Tahamtan et al. [29]				✓	
Han and Sim [14]	✓	✓	✓		✓

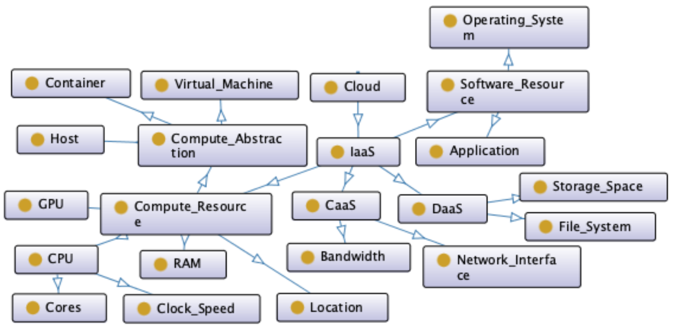

Fig. 1. Structure of the CISO ontology.

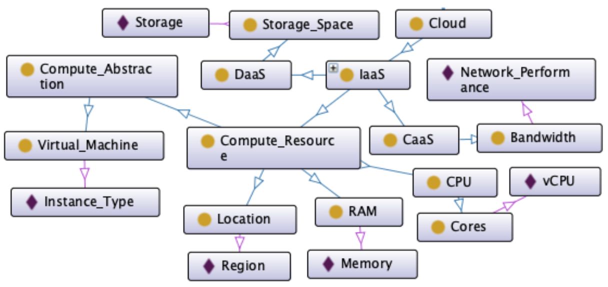

Fig. 2. Example of using CISO ontology to describe VM specification from AWS.

the main elements necessary for allocating computing resources, which were not adequately present in the analyzed ontologies.
To illustrate, a brief example is shown where the CISO ontology is used to describe VM specification from AWS. First of all, *Instance Type* has been defined as individual of *Virtual Machine* and *Region* as *Location*'s individual. The hardware properties are represented as individuals of the modeled classes, such as *vCPU* from *Core*, for instance. In Fig. 2, the individuals and the relationships are depicted.

4. ESTIMATING ON-PREMISES DEPLOYMENT COST

For the infrastructure cost estimate to be complete, it must also consider allocating resources using other deployment models which may be the case for an infrastructure on-premises. To this end, the Total Cost of Ownership (TCO) analysis technique is applied. This technique is generally used as a means of addressing the real costs attributing to owning and managing an IT infrastructure in business. A novel methodology to obtain this estimate is proposed, since the main existing strategies present particular assumptions, making it difficult to use in the scenario. Furthermore, although some CSPs offer tools for estimating TCO, they are skewed to privilege results in their environments.
The proposed TCO estimation methodology is based on Li et al. [17] and Cui et al. [9]. These two approaches complement each other since the first offers a broad methodology for resources in a datacenter, while the second is more focused on estimating the power cost in this environment. The following describes each of the components in the proposed methodology.

Amortization or depreciation of the cloud. The construction and maintenance of a datacenter require resources such as servers, cooling equipment, among others. These resources have specific depreciation rates making the cost of different metrics items can not be put together directly or comparatively fairly. To overcome this problem, a component is defined: amortization, which constitutes a monthly cost rate.
Typically, real estate is depreciated over periods of 10 years, and other items like server and facilities are depreciated over three years [26]. A cost amortization rate parameter is proposed, Arp, which can be applied to both one time purchases (server cost, facility cost, etc.) and operational expenses (power cost, cooling cost, etc.). The amount is calculated by Equation 1 considering the amortization period in hours (A_p) with a depreciation rate of 5%.

$$Arp = \frac{1.05 \times \text{time}}{A_p} \tag{1}$$

Server cost. For simplicity, it is adopted equipment with the same configuration. Each h physical server is characterized by the tuple $\{c_h, r_h, b_h, s_h\}$, where c_h is the number of CPU cores, r_h is the amount of RAM, b_h is the amount of bandwidth (BW), and s_h is the amount of storage. RAM, BW, and storage are measured in gigabytes (GB), gigabits per second (Gb/p), and gigabytes (GB), respectively. A fraction of server resources is allocated to VMs to fulfill the cloud customer demands. Usually, multiple VMs can be deployed on a single server. Each VM is characterized by a tuple with the same attributes as the description of a physical server. Thus, given n VMs with the same configuration, the number of physical servers, N_s, is calculated according to the Equation 2.

$$N_s = n \times \max\left\{\left\lceil\frac{c_{\text{VM}}}{c_h}\right\rceil, \left\lceil\frac{r_{\text{VM}}}{r_h}\right\rceil, \left\lceil\frac{b_{\text{VM}}}{b_h}\right\rceil, \left\lceil\frac{s_{\text{VM}}}{s_h}\right\rceil\right\} \tag{2}$$

The server cost, C_s, is estimated using N_s, the cost of each physical server (C_ps), and the amortization cost, using the Equation 3.

$$C_\text{s} = N_s \times C_\text{ps} \times Arp(\text{time}) \tag{3}$$

Network cost. The network cost is related to the number of switches, Network Interface Cards (NIC), and cables used to connect physical servers to the network. However, due to the negligible price compared to other equipment, cabling cost is neglected.
We initially calculate the number of switches (N_switch), using Equation 4, based on the number of physical servers (N_s), the number of NIC per virtualized server (N_NIC), the number of ports per NIC ($N_{p\text{NIC}}$), and the port number of a network switch (N_port).

$$N_\text{switch} = N_\text{NIC} \times N_{p\text{NIC}} \times \frac{N_s}{N_\text{port}} \tag{4}$$

Thus, the network cost, C_{net}, considers the switch number and the price of each switch (P_s) according to Equation 5.

$$C_{\text{net}} = N_{\text{switch}} \times P_s \times Arp(\text{time}) \tag{5}$$

Software cost. In addition to physical equipment, cost associated with the required software licenses is included. For this purpose, software is split into three categories:

— **Type I**: Operating Systems and any other software being licensed by suit;
— **Type II**: VM software, and any other software that is licensed by the processor number;
— **Type III**: management software, which is licensed by the processor number.

Thus, considering the number of licenses of each type (N_*), their cost (C_*), and a subscription factor (percentage of unit price that yield annual fee), S_*, software cost, C_{sw}, is estimated according to the Equation 6.

$$C_{\text{sw}} = (C_o \times S_o \times N_{\text{oslic}} + C_{\text{ss}} \times S_s \times N_{\text{slic}} + C_m \times S_m \times N_{\text{mlic}}) \times Arp(\text{time}) \tag{6}$$

Support and maintenance cost. The support and maintenance cost (C_{sm}) involves software distribution and update, asset management, troubleshooting, server configuration, threat protection, and others. Its calculation is based on the number of administrators responsible for support and maintenance (N_{labor}), the average time spent on unit system under utilization (T_{use}), the time spent on all the idle systems (T_{idle}), and the rating number of salary averages (R_{salary}), using the Equation 7.

$$C_{sm} = N_{\text{labor}} \times (T_{\text{use}} \times N_s + T_{\text{idle}}) \times R_{\text{salary}} \tag{7}$$

Power cost. The power cost involves the consumption in kWh of the servers, cooling, network switches, and lighting, with the servers responsible for most of this consumption since they use up to 80% of peak power even at 20% utilization [31]. The Equation 8 estlabishes the power cost (C_{pw}), where $C_{\text{e-kwh}}$ is the electricity cost per kWh, PUE represents the total power utilization efficiency which is calculated using the analytical model provided in [9].

$$C_{pw} = C_{\text{e-kwh}} \times \frac{\omega_{\text{s}} + \omega_{\text{n}}}{1000} \times \text{time} \times \text{PUE} \tag{8}$$

Cooling cost. The cooling cost (C_{cool}) tends to account for 30% of the total power cost of a datacentre [31]. To estimate it, a parameter λ is applied, which represents how much that the cooling equipment consumes power for every 1W of heat dissipation in the datacenter. The other variables considered are β, Airflow Redundancy Constant (redundant airflow required to cool datacenter), and ϵ, Humidification Constant (redundant airflow to account for the burden of humidification). The calculation is performed according to the Equation 9.

$$C_{cool} = \lambda \times (1 + \beta) \times \frac{C_{\text{e-kwh}} \times \text{time}}{\epsilon} \tag{9}$$

Facilities cost. The cost of facilities (C_{fac}) comprises the expenses necessary for the execution of the equipment. It is calculated according to the Equation 10 and includes the number of racks (N_{rack}) and the cost of facilities per rack (C_{fp}).

$$C_{fac} = N_{\text{rack}} \times C_{\text{fp}} \times Arp(\text{time}) \tag{10}$$

Immovable cost. To estimate this component it is necessary to calculate the space taken up by all the racks under utilization (U_{space}), given by Equation 11, where N_{ft^2} is the number of square feet per rack, and ϕ_{space} is the percent of space taken by racks in all (<1).

$$U_{\text{space}} = \frac{N_{\text{ft}^2} \times N_{\text{rack}}}{\phi_{\text{space}}} \tag{11}$$

Immovable cost (C_{imm}) is then calculated by the Equation 5, being C_{ft^2} the cost per square foot to build the datacenter.

$$C_{imm} = C_{\text{ft}^2} \times U_{\text{space}} \times Arp(\text{time}) \tag{12}$$

Cloud's total cost. In the cloud's total cost, the total sum of all components is estimated as in Equation 13.

$$C_{total} = C_{\text{s}} + C_{\text{net}} + C_{\text{sw}} + C_{sm} + C_{pw} + C_{cool} + C_{fac} + C_{imm} \tag{13}$$

Using the proposed methodology, it is possible to estimate the total cost of implementing resources considering an on-premises infrastructure. This strategy was used in the construction of the CSB that will be presented in the next section.

5. TOOL ARCHITECTURE

The proposed ontology for cloud IaaS description, and the proposed strategy for on-premises cost estimation, establish the basis of the proposed tool. Although the proposed ontology allows the representation of multiple IaaS, the scope is set Virtual Machines (VM). Its architecture is illustrated in Fig. 3.

The interaction with the user is performed through the component named ***Query processing engine***. Through it the user describes the required resources, which include hardware characteristics and the number of requested VMs. The description is provided using logical and relational operators, which makes it possible to meet a broader set of requirements. Based on this description, the component generates the query in an internal language passed on to the ***Result aggregator*** component which is responsible for consolidating the results obtained based on the different cloud deployment models. Currently, the tool only considers public cloud and on-premises environments, but the architecture was designed to allow its future expansion to use other deployment models. The cost evaluation considering the two currently supported models is performed based on the generated query, through the components ***Public cloud model analyzer*** and ***Private cloud model analyzer***.

The component Private cloud model analyzer performs the cost evaluation for an infrastructure on-premises based on the proposed TCO estimation strategy. On the other hand, due to the CSPs multiplicity and heterogeneity, cost evaluation in public cloud is a challenger. Because of this, the component in charge of this task uses two auxiliary components. The first one, named ***Search agent***, is responsible for discovering resource information in the different supported CSPs. This task is performed in a specific way for each provider, through ***cloud provider adapters***. The ***Cloud information retrieval agent*** aggregates the results provided by these adapters, which stores them in a database named ***Cloud service offers***. Matching terms and establishing a common nomenclature is the second component's responsibility, named ***Knowledge agent***. Its main sub-component, ***Cloud service reasoning agent***, uses the proposed ontology to reason about the relationships among cloud services. To do so, it uses the ***Rating agent*** component, which helps to identify inconsistencies and selection in the case of equivalent results. The semantic results are kept in the ***Domain knowl-***

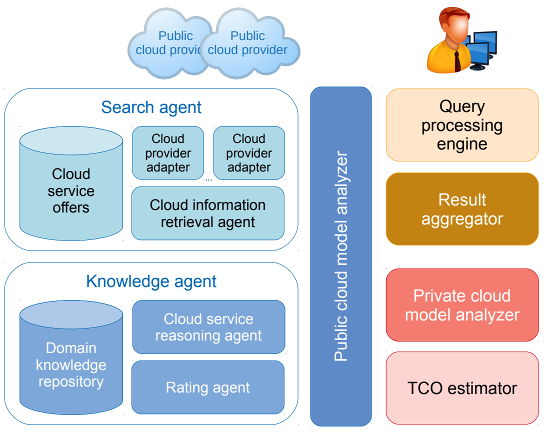

Fig. 3. Proposed tool architecture.

edge repository database, which can be updated in the face of new queries.

6. TOOL IMPLEMENTATION

To demonstrate the proposed tool, a prototype of the architecture presented in the previous section was developed. The prototype is a web application implemented in Python using the Django framework[2]. The source code of the tool is available at `https://github.com/GeovRodri/tco-multicloud`. The details of the implementation are described following.

The CISO ontology's implementation, the main element of the Knowledge agent component, is based on the Resource Description Framework (RDF) specifications, part of a W3C specification family. The RDF/XML format was adopted using properties from the Ontology Web Language (OWL) [19], through the Owlready2 library[3], which performs the queries and returns the equivalent terms for each provider.

Regarding the Search agent component, since only a limited set of cloud providers offer service description APIs, web scraping techniques were used to extract information about resources. This alternative has the advantage of allowing the inclusion of any cloud provider in the tool. The tool's current prototype allows data from AWS, Microsoft Azure, Google, Alibaba Cloud, and Oracle. These providers were used because Gartner reported them as the main ones in its last report Magic Quadrant for Cloud Infrastructure as a Service [5]. In the prototype, only the on-demand pricing model was considered.

One of the challenges in implementing the Search agent component is related to the frequency with which the cloud providers adapters make queries. The most suitable solution would a real-time query to avoid using outdated information. However, this proved to be unfeasible due to the latency to get the results. By measuring the total time of data extraction, it was possible to identify a latency of 2.44 ± 0.18 hours with a confidence interval of 95%. This notable latency is mainly due to the data extraction performed with web scraping and huge volume of transmitted data. Even with the solution's improvement through parallel queries, the latency would still be high to be performed in real-time. Another observed feature was the variation frequency of resource description data. To demonstrate it, it was carried out data collection regarding VM offers and cost. Each VM offer concerns the description of the VM type (hardware characteristics), associated operating system, and region of availability. The data were collected twice a day in two different periods during an interval of 126 days (04/29/2020 to 09/02/2020). A single CSP, Microsoft Azure, showed daily variation in VM offers number and/or costs, without, however, presenting two variations on the same day. For the other CSPs, the variation in offer number and cost occurs at intervals greater than one day, as can be seen in Fig. 4. In this figure, the dates for which offers number and costs have been kept constant are omitted. Given that, it is possible to state that a daily query of resource description data is acceptable to keep the resource model updated.

[2]`https://www.djangoproject.com`
[3]`https://pypi.org/project/Owlready2`

7. QUALITATIVE EVALUATION

This section describes an experimental evaluation of the tool through a qualitative analysis that aims at evaluating users' productivity, in terms of the time required to query VM types, and accuracy, in terms of the accuracy of the results.

Experiment design. We compare the proposed tool against some other three similar CSBs: AWS Pricing Calculator [3], Azure TCO Calculator [20], and Cloudorado Cloud Server Comparison [8]. The selection of the first two alternatives is because they performed TCO analysis in conjunction with VM types querying, while the third one was chosen because it allows querying in multiple CSPs. It is adopted a controlled experiment strategy where participants are given a VM specification with hardware characteristics. A single specification was used per participant, aiming they use the tools held within reasonable experimental time and also to evaluate the accuracy of the tool when considering different queries. This experiment design is leveraged to evaluate the interaction of the users with the proposed tool. The analysis of this interaction enables identifying advantages and limits of it and improvements that can be introduced.

The experiment procedure lasts for a maximum of two hours per participant. All participants used the same powerful PC.

The setting of the VM specifications used in the experiment was designed to reflect real scenario queries. For this, it was adopted the resource requirements of applications that are typically deployed in the cloud. For simplicity, only CPU, RAM, and storage are used as hardware attributes.

Based on [27], a Pareto efficiency criterion was adopted using as preference criteria compute efficiency, measured as "compute ECU[4]/$- hr", memory efficiency, measured as "memory GB/$-hr", and storage efficiency, measured as "storage GB/$-hr". As a result, different hardware specifications were established, ranging from 1 CPU core, 1GB memory, no storage, to 32 CPU core, 64GB memory, 1TB storage.

Each participant should consult the assigned VM configuration on the four tools. In addition to the search results, they should collect the time taken to obtain the results. At the end of the experiment, the participant fills a questionnaire about their experience in cloud computing concepts, cloud service selection, TCO analysis, and resource selection modeling languages and tools. Also, each participant is asked to provide feedback on the proposed tool's usability and productivity and things to improve.

[4]An Elastic Compute Unit (ECU) is a normalized unit of CPU integer processing power available in an AWS instance. According to AWS, one EC2 Elastic Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

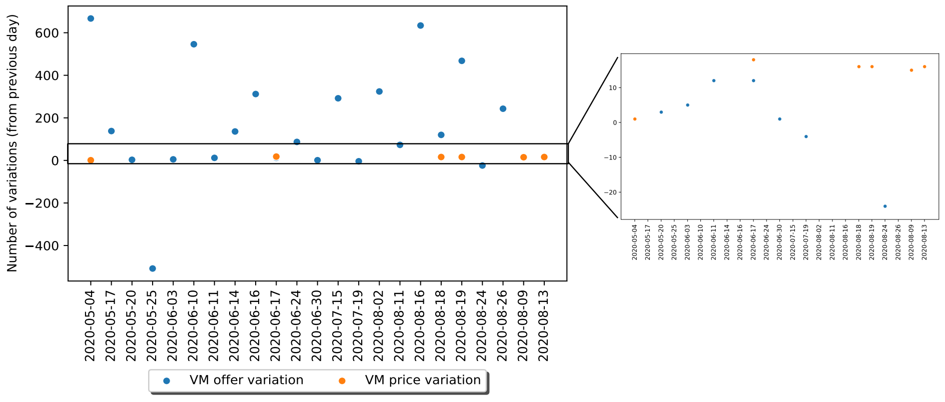

Fig. 4. Variation of VM offers and cost from 04/29/2020 to 09/02/2020.

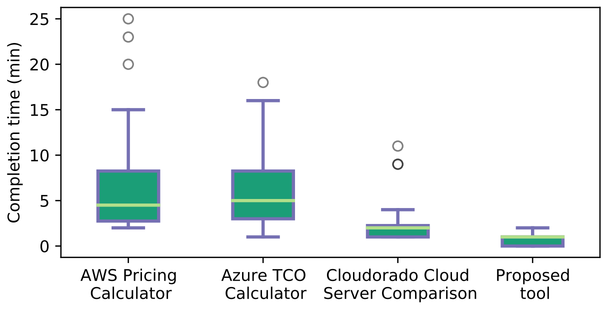

Fig. 5. Box-plot of time spent by participants to complete each search using either approaches.

Before the experiment starts, participants are introduced to relevant cloud computing, VM types, and cloud-based pricing concepts. They are also introduced to the experiment guidelines with a brief explanation of expected tasks (4-5 minutes). During the experiment, additional guidance is provided to any participant requiring assistance for following the experiment guidelines, but no guidance was provided about interacting with the tools.

Participants were recruited from Bachelor of Information Systems' students at the Federal Institute of Goiás, Brazil. An incentive for participation was offered in the form of extra points at the course's grade. Overall, 24 participants with varying expertise levels in programming and cloud systems were recruited. These were broken down as 17 with no knowledge about cloud computing, 6 with basic knowledge, and just 1 with practice knowledge. Further, 10 of them self-reported high knowledge about TCO analysis and 14 no knowledge. None of the participants have used the tools before.

Productivity results. The productivity of participants is evaluated by calculating the time spent to get results. This time is collected individually for each tool and includes the period spent specifying the VM configuration assigned to the participant. Figure 5 presents the box-plot of the distributions of the time spent by participants in the experiment. What can be clearly seen in this figure is that the proposed tool allows the resource search to be carried out easily, regardless of the participant. While the other tools have a time distribution with high variation, including the presence of outliers, for the proposed tool, the variation is small, with a maximum of 2 minutes to complete the search.

Table 2. Accuracy of service selection using the analyzed tools.

Tool	**Precision**	**Recall**	**F-meas.**
AWS Pricing Calculator	0.375	0.001	0.001
Azure TCO Calculator	0.625	0.001	0.002
Cloudorado Cloud Server Comp.	1	0.019	0.037
Proposed tool	1	0.986	0.993

Accuracy results. The accuracy of the selection is evaluated by calculating precision, recall, and F -Measure score considering the results returned by each tool, concerning the satisfaction of the queries carried out. For a given query, the Precision P is the proportion of the relevant VM types retrieved to all the retrieved VM types and is mathematically expressed as the Equation 14.

$$P = \frac{|relevant\ VM\ types \cap retrieved\ VM\ types|}{|retrieved\ VM\ types|} \tag{14}$$

Similarly the Recall R is the proportion of relevant VM types which have been retrieved to all the relevant VM types and is mathematically expressed as the Equation 15.

$$R = \frac{|relevant\ VM\ types \cap retrieved\ VM\ types|}{|all\ relevant\ VM\ types|} \tag{15}$$

Finally, it is computed the F-measure score, which is the harmonic sum of P and R which gives the accuracy of the tool and is described as the Equation 16.

$$F = 2 * \frac{|precision * recall|}{|precision + recall|} \tag{16}$$

As a relevant cloud service, were counted all types of VMs that precisely matched the specified queries. Some types returned by the AWS Pricing Calculator and Azure TCO Calculator tools have been classified as irrelevant because they do not match the hardware configuration specified in the query. It is also important to note that in addition to the CSPs supported in the proposed tool, the results of other CSPs (10 in total) supported in the Cloudorado Cloud Server Comparison tool were also included. The results using the considered metrics are presented in Table 2.

As can be seen in Table 2, as it relies on a single provider, the first two tools have very low accuracy. Another reason for this is that most of the results returned were classified as irrelevant. On the other hand, the Cloudorado Cloud Server Comparison tool has

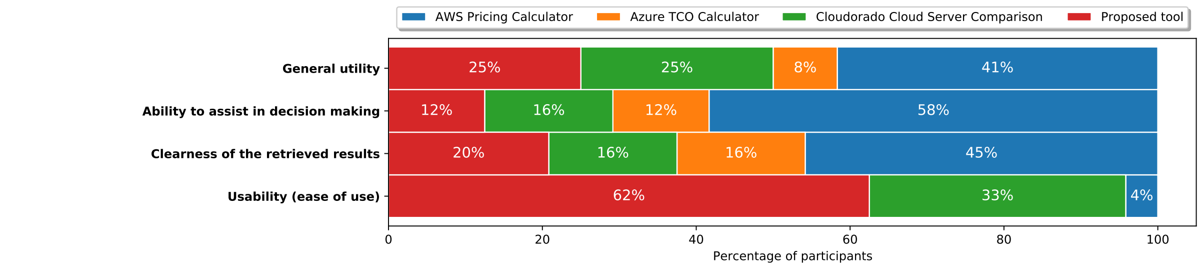

Fig. 6. Participant feedback on tool characteristics.

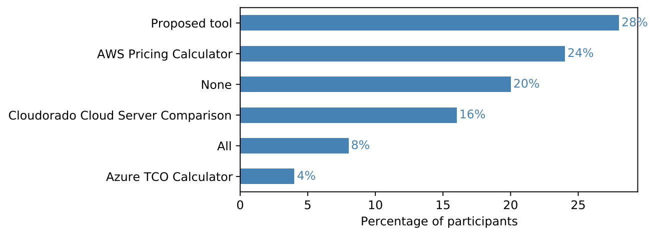

Fig. 7. Participant feedback on tool adoption.

maximum accuracy, but the recall and F-measure are low, even relying on more CSPs. This is because a single result is returned per provider. In contrast, the proposed tool returns all types that satisfy the search. This results in a remarkably better level of accuracy than the other tools.

Exit interview responses. After the completion of search in the analyzed tools, the participants were interviewed in order to survey their experience of using them. They were asked to answer six questions, five of which aimed to assess tools characteristics and one to assess tool adoption.
In the first four questions, participants were asked to indicate which tool best meets four quality criteria. The criteria, as well as the results, are shown in Fig. 6. For presenting a broader set of features, the AWS Pricing Calculator tool was designated as the best in three of the criteria. However, the proposed tool held the second position in two of them, and for the usability criterion, it was designated as the best. There is a need to improve the ability of the proposed tool to assist in decision making. This is mainly because the report provides only the search results without indicating more information about the provided resources.
In the fifth question, participants were asked to designate which tool or strategy they would use if they needed to find the lowest cost to allocate a given VM on the cloud. The result is shown in Fig. 7. Although the proposed tool was not designated as the best in the previous questions, it was pointed out as the one adopted by the majority (28%) of the participants, making AWS Pricing Calculator the second option. Interestingly, a large part of the participants (20%) chose not to use any of the tools directly, but to perform the search on general-purpose search tools, probably due to their greater familiarity with this strategy.
The participants were asked to present in open format reviews and suggestions about the proposed tool. In general, the reviews were positive, suggesting that the proposed tool has an intuitive interface, being simple, clear, and high usability. As negative aspects, the participants highlighted the lack of detail in the information returned. For example, they pointed out the need to explicitly explain the monetary and time unit and which CSPs are querying in the search. Participants also indicated that the tool does not fully assist the resource selection process because important features, such as cost projection over time, are not offered.

Experimental validity. As is common with experimental case study designs, external validity (i.e., the ability to generalize the results) is naturally impaired to an extent in order to attain higher internal validity (i.e., validating the cause-effect inference). However, since one of the main goals is to facilitate resource selection by users without prior knowledge, and since the searches were carried out using various VM configurations, it is possible to state that the results indicate the proposed tool's quality. However, there are some validity threats, such as the fact that the participants themselves collect time without using automated tools. Also, qualitative indicators can be questionable due to the participants' inexperience, which can be reinforced by the fact that many reject all the tools.

8. FINAL REMARKS

This paper presented a cloud service broker's design and implementation, whose goal is to provide a cost-optimized tool to search computing resources considering different cloud deployment models and multiple cloud service providers. The findings from an experimental case study suggest that the tool's raised level of usability and performance results in significant improvements in search productivity and accuracy in service selection. As future work, it is suggested the inclusion of new CSPs in the proposed tool and the implementation of new billing mechanisms and cloud deployment models. On top of that, a more in-depth analysis of the outcomes obtained with the tool should also be performed. Another outcome is putting the cost analysis tool in the practice of the customer's cloud environments and try to draw guideline rules from the analysis for realizing cloud practices with high economic efficiency.

9. REFERENCES

[1] Mustafa M Al-Sayed, Hesham A Hassan, and Fatma A Omara. Towards evaluation of cloud ontologies. *Journal of Parallel and Distributed Computing*, 126:82–106, 2019.

[2] Abdullah Alfazi, Quan Z Sheng, Yongrui Qin, and Talal H Noor. Ontology-based automatic cloud service categorization for enhancing cloud service discovery. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, pages 151–158. IEEE, 2015.

[3] Amazon. AWS Pricing Calculator. https://calculator.aws, Access on 09/11/2020, 2020.

[4] Ibrahim Attiya and Xiaotong Zhang. Cloud computing technology: Promises and concerns. *International Journal of Computer Applications*, 159(9):32–37, 2017.

[5] Raj Bala, Bob Gill, Dennis Smith, David Wright, and Kevin Ji. Gartner Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. https://pages.awscloud.com/GLOBAL-multi-DL-gartner-mq-cips-2020-learn.html?pg=WIAWS-tx, Access on 09/08/2020, 2020.

[6] Michael Behrendt, Bernard Glasner, Petra Kopp, Robert Dieckmann, Gerd Breiter, Stefan Pappe, Heather Kreger, and Ali Arsanjani. Introduction and architecture overview IBM cloud computing reference architecture 2.0. *Draft Version V*, 1(0), 2011.

[7] Gabriel G Castañé, Huanhuan Xiong, Dapeng Dong, and John P Morrison. An ontology for heterogeneous resources management interoperability and HPC in the cloud. *Future Generation Computer Systems*, 88:373–384, 2018.

[8] Cloudorado. Cloud Server Comparison. https://www.cloudorado.com/cloud_server_comparison.jsp, Access on 09/11/2020, 2020.

[9] Yan Cui, Charles Ingalz, Tianyi Gao, and Ali Heydari. Total cost of ownership model for data center technology evaluation. In *2017 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 936–942. IEEE, 2017.

[10] Andy Edmonds, Thijs Metsch, Alexander Papaspyrou, and Alexis Richardson. Toward an open cloud standard. *IEEE Internet Computing*, 16(4):15–25, 2012.

[11] Abdessalam Elhabbash, Faiza Samreen, James Hadley, and Yehia Elkhatib. Cloud brokerage: A systematic survey. *ACM Computing Surveys (CSUR)*, 51(6):1–28, 2019.

[12] Google. Google Cloud Pricing Calculator. https://cloud.google.com/products/calculator, Access on 09/18/2020, 2020.

[13] Nikolay Grozev and Rajkumar Buyya. Inter-Cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2014.

[14] Taekgyeong Han and Kwang Mong Sim. An ontology-enhanced cloud service discovery system. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 17–19, 2010.

[15] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. NIST cloud computing standards roadmap. *NIST Special Publication*, 35:6–11, 2011.

[16] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the Cloud? An architectural map of the Cloud landscape. In *2009 ICSE workshop on software engineering challenges of cloud computing*, pages 23–31. IEEE, 2009.

[17] Xinhui Li, Ying Li, Tiancheng Liu, Jie Qiu, and Fengchun Wang. The method and tool of cost analysis for cloud computing. In *2009 IEEE International Conference on Cloud Computing*, pages 93–100. IEEE, 2009.

[18] Benedikt Martens, Marc Walterbusch, and Frank Teuteberg. Costing of cloud computing services: A total cost of ownership approach. In *2012 45th Hawaii International Conference on System Sciences*, pages 1563–1572. IEEE, 2012.

[19] Deborah L McGuinness, Frank Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[20] Microsoft. Azure TCO Calculator. https://azure.microsoft.com/en-us/pricing/tco/calculator, Access on 09/11/2020, 2020.

[21] Giuseppe Di Modica and Orazio Tomarchio. A semantic framework to support resource discovery in future cloud markets. *International Journal of Computational Science and Engineering*, 10(1-2):1–14, 2015.

[22] Francesco Moscato, Rocco Aversa, Beniamino Di Martino, Teodor-Florin Fortiş, and Victor Munteanu. An analysis of mosaic ontology for cloud resources annotation. In *2011 federated conference on computer science and information systems (FedCSIS)*, pages 973–980. IEEE, 2011.

[23] Ioannis Patiniotakis, Yiannis Verginadis, and Gregoris Mentzas. Preference-based cloud service recommendation as a brokerage service. In *Proceedings of the 2nd International Workshop on CrossCloud Systems*, CCB 14, New York, NY, USA, 2014. Association for Computing Machinery.

[24] Przemyslaw Pawluk, Bradley Simmons, Michael Smit, Marin Litoiu, and Serge Mankovski. Introducing STRATOS: A cloud broker service. In *2012 IEEE fifth international conference on cloud computing*, pages 891–898. IEEE, 2012.

[25] RightCloudz. Cloud Service Providers Evaluation. https://rightcloudz.com/RankCloudzOnline, Access on 09/18/2020, 2020.

[26] Pierangelo Rosati and Theo Lynn. Measuring the business value of infrastructure migration to the cloud. In *Measuring the Business Value of Cloud Computing*, pages 19–37. Palgrave Macmillan, Cham, 2020.

[27] Willard Simmons. How to select the most efficient AWS EC2 instance types using Pareto front analysis. https://read.acloud.guru/selecting-the-most-efficient-aws-ec2-instance-types-using-pareto-front-analysis-3a5c81bae3a2, Access on 09/08/2020, 2017.

[28] Katy Stalcup. $14.1 Billion in Cloud Spending to be Wasted in 2019. https://www.parkmycloud.com/blog/cloud-spending/, Access on 09/17/2020, 2019.

[29] Amirreza Tahamtan, Seyed Amir Beheshti, Amin Anjomshoaa, and A Min Tjoa. A cloud repository and discovery framework based on a unified business and cloud service ontology. In *2012 IEEE Eighth World Congress on Services*, pages 203–210. IEEE, 2012.

[30] Lina Tankelevicienea and Robertas Damaseviciusb. Characteristics of domain ontologies for web based learning and their application for quality evaluation. *Informatics in Education*, 8(1):131, 2009.

[31] AF Thompson, FV Olofinlade, and B Bello. On Cost and Energy Efficiency of Security in Cloud Computing. *International Journal of Computer Applications*, 180(14):0975–8887, 2018.

[32] Wei Wang, Di Niu, Ben Liang, and Baochun Li. Dynamic cloud instance acquisition via IaaS cloud brokerage. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1580–1593, 2014.

[33] Denis Weerasiri, Boualem Benatallah, and Moshe Chai Barukh. Process-driven configuration of federated cloud resources. In Matthias Renz, Cyrus Shahabi, Xiaofang Zhou, and Muhammad Aamir Cheema, editors, *Database Systems for Advanced Applications*, pages 334–350, Cham, 2015. Springer International Publishing.