# Virtual Machine Planning for Cloud Brokering Considering Geolocation and Data Transfer

Javier Alsina, Santiago Iturriaga, Sergio Nesmachnow
Universidad de la República, Uruguay, {jalsina, siturria, sergion}@fing.edu.uy

Andrei Tchernykh
CICESE, Ensenada, Mexico, chernykh@cicese.mx

Bernabé Dorronsoro
University of Cádiz, Spain, bernabe.dorronsoro@uca.es

***Abstract*—This article addresses a virtual machine (VM) allocation problem that appears in a novel business model for cloud computing. In this model, a cloud service broker owns a number of cloud reserved instances that outsources to its customers as cheap as on-demand VMs. The objective of the broker is to efficiently manage its reserved resources to maximize its revenue. We enhance the previous definition of the problem by considering more realistic parameters: geographical localization of resources and users, different types of applications, and data transfer costs. We propose a set of heuristics to solve the optimization problem of maximizing the cloud provider profit while offering appropriate Quality-of-Service to the users. The experimental analysis is performed over different scenarios using real data from cloud providers.**

***Keywords—virtual machine planning; cloud computing; Infrastructure as a Service, cloud brokering***

## I. INTRODUCTION

Cloud computing [1][2] has emerged as one of the main computing paradigms for large-scale data processing. It provides several important features, including elasticity, flexibility, and large processing power, among others. These features make cloud computing useful for big data processing and solving problems with large computing demands.

In the lower level of services provided by the cloud computing stack, Infrastructure as a Service (IaaS) is the model for delivering computing, storage, and other resources to users. The computing resources are based on virtual machines (VM) deployed over physical servers according to specific policies to satisfy user demands. A set of online services abstracts the details of the physical infrastructure to the user, handling the virtual resources with specific tools, such as hypervisors, to operate the VMs. Therefore, providers can manage the user requests and guarantee the elasticity properties of the cloud by using appropriate assignment policies. The model applied by providers to rent the resources is that of utility computing: the charging cost reflects the number of resources (computing power, storage, data transfers) requested and consumed by the users [3][4].

In our previous work [5][6], we propose and studied an IaaS-level business model for cloud computing, considering the cloud broker as an intermediary between cloud providers and users. Generally, the broker helps the users to find the best prices from different available clouds that fit the user requirements and to define possible configurations to deploy the user applications in the cloud [7]. In our model, the broker builds a virtual cloud using reserved VM instances (called RI). RIs are rented from traditional cloud providers for a large period of time (six months, one year, or longer), thus, the broker pays a reduced price. The broker then sublets the RIs to customers cheaper than on demand instances. This business model is feasible and profitable due to the price difference between reserved and on-demand VMs in the cloud [8]. When the broker does not have available RIs to deploy a customer VM request without violating the contracted Service Level Agreement (SLA), on-demand VMs must be bought to satisfy the demand, in order to keep a high reputation level. In this case, there is a profit reduction, because the broker pays the cloud provider more than the amount charged to its customer.

The related optimization problem is to efficiently allocate all customer requests into the available RIs (owned by the broker), in order to maximize the broker profit. Specific SLAs are defined for each VM request according to the features demanded by the user (i.e., computing power, storage, number of cores, etc.).

This resource allocation problem is a generalization of the multidimensional knapsack problem [9], which is proved to be NP-hard. Two situations should be avoided: i) having RIs that are either not sublet, or underutilized (i.e., sublet to a customer requiring a VM of less capacity), and ii) overbooking the available RIs. The first case reduces the revenue of the broker, while the second case also implies some additional cost, because the broker will be forced to reserve on-demand VMs from the cloud providers to guarantee the service.

The main contributions of the research reported in this article are fourfold. First, we extend the problem model presented in [5][6] to consider a more realistic situation by including geolocation of both providers and users, and data transfer. Second, we introduce a more realistic cost model, which takes into account data transfer costs according to rates of real cloud providers considering localization of resources and users. Third, we model two types of applications (one single application was considered in previous works): *computational applications*, deferrable according to some deadline limitations, and *web service applications*, which are not deferrable and must be executed in a period specified by the user, without flexibility.

2330-2186/16 $31.00 © 2016 IEEE
DOI 10.1109/CloudCom.2016.59

Forth, we propose six heuristics based on intuitive optimization strategies. They are evaluated on a realistic benchmark of instances gathered from real cloud providers, and compared against a recently proposed heuristic from the state of the art.

The paper is structured as follows. Next section presents the formulation of the optimization problem. The related work is reviewed in Section III. Section IV describes the proposed heuristics. The experimental evaluation of the studied strategies over a set of realistic workloads and scenarios using real data from actual cloud providers is reported in Section V. Finally, the conclusions and main lines for future work are formulated in Section VI.

## II. RELATED WORK

A large number of papers in the literature deal with cloud brokering techniques to benefit users. Two recent examples are the works by Wang et al. [10] and Hwang et al. [11]. These works focus on minimizing the price of on-demand VMs to users, and how users can combine the use of on-demand and reserved VMs to minimize their costs and cover their needs.

Zhang et al. [12] proposed a broker to deal with the allocation of on-demand VMs into the least possible number of reserved instances, with the consequent savings for its customers. This problem is related to the one we deal with, in which the broker needs to combine the use of on-demand and reserved VMs in order to serve its customers and minimize its cost. The difference between the mentioned work and ours is that the broker proposed by Zhang et al. books RIs according to the current demand. On the contrary, our broker does not know demand and needs to make the best possible use of the available resources, taking into account that at any time demand may increase and it should have the least possible impact on the budget. Lučanin et al. [19] consider geolocation and propose the migration of executing VMs in order to minimize the energy budget of a single cloud provider. In this work, quality of service is considered to be related to the number of VM migrations.

The work by Alasaad et al. [13] deals with the resource allocation problem for web services in clouds, one of the two kinds of user requests we consider in our problem. In particular, they focus on media streaming services. Communications costs are considered by Lee et al. [14], where the authors consider how to deploy services in a hybrid private/public cloud in order to reduce communication costs and delays. A hybrid private/public cloud is also the scenario targeted by Van den Bossche et al. [15], where the authors propose three heuristics to schedule the resources of the system, taking into account the cost of both data transfer and computation in the public cloud. There are several similarities between the problem addressed in [15] and the one proposed in our work, if we consider the virtual cloud of the broker to be a private cloud. In contrast, in our work we also consider geolocation and two different kinds of user requests, namely web services and computational tasks, which present different requirements for the planning. We use the heuristics by Van den Bossche et al. as a reference baseline to compare the results computed by the heuristics we introduce in this article.

## III. THE VIRTUAL MACHINE PLANNING PROBLEM WITH GEOLOCATION AND DATA TRANSFER

The Virtual Machine Planning Problem with Geolocation and Data Transfers (VMPP-GDT) in cloud systems proposes optimizing the allocation of a set of user VM requests to a set of cloud resources (owned by the broker in the form of reserved cloud instances). A VM is rented for a fixed time span and its execution has to be completed before some hard deadline. Each VM request is described by its hardware requirements: processor speed, memory, physical storage capacity, and number of cores.

The main goal of the broker is to satisfy as many requests as possible by making the best use of its own reserved VMs to maximize its profit. When user requests cannot be handled using the available RIs, the broker may decide to reject the requests or lease on-demand VMs in the cloud. In our approach, the broker follows the second option. The main argument is that, even when the broker action implies a profit reduction, we consider that the reputation of the broker is of major importance to attract new customers and prevent existing ones from leaving. Therefore, it is desirable to keep a high reputation, at the cost of losing some money in some cases. Additionally, it provides a lower bound of the broker profit, which might be increased by rejecting some requests in overloaded scenarios.

The VMPP-GDT consider the following elements:

- A set of user requests $VM = \{v_1, \dots, v_n\}$, each one demanding a virtual machine for $T(v_i)$ time units. User requests arrive in batches. Each $v_i$ has an arrival time $A_i$, and specific hardware demands that must be satisfied: processor speed $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, number of cores $nc(v_i)$, and data transfer per time unit $DT(v_i)$.
- Two types of user requests: computation tasks (CT) and web services (WS). The execution of a CT must be finished before a deadline $D(v_i)$. There is flexibility on its starting time according to the scheduling decision. A WS is an interactive task (e.g., a service for selling concert tickets), thus both starting time $ST(v_i)$ and finishing time (deadline, $D(v_i)$) must be satisfied. WS applications cannot be deferred.
- A set of RIs owned by the broker $B = \{b_1, \dots, b_m\}$, $m \ll n$. Each RI has specific features including processor speed $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$ capacities (matching a predefined list of instance types $t(b_j) \in \{t_1, \dots, t_k\}$), location $z_j$, a cost function $C$ for RIs, and a cost function $COD$ for on-demand instances, with $C(b_j) \ll COD(b_j)$. These costs are provided on an hourly basis. The prediction of the number and type of RIs owned by the broker is out of the scope of this paper and it is considered as future work.
- A pricing function $p(b_j)$ that defines the hourly price for the RI of type $b_j$. In order to attract customers, the broker charges a lower price than the on-demand price for that

kind of instance, i.e., $p(b_j) < COD(b_j)$. Moreover, if the cheapest RI (i.e., type $b_k$) that can allocate a VM $v_i$ is not available, the broker can assign $v_i$ to a higher capacity RI, but charging the same amount as for $b_k$. This could decrease the revenue, but it prevents the broker from buying more expensive on-demand instance. At the same time, the customer will be pleased by the better performance offered.

- Each RI is located in a zone $z_i$ from a set of geographic zones $Z = \{z_1, \dots, z_l\}$. The cost function for data transfer $DTC(z_j, z_i)$ defines the price (per hour) for data transfer to a RI located in zone $z_j$ from users located in zone $z_i$. The portion of data transferred from request $v_j$ to each zone $z_i$ is defined by $0 \le DTD(v_j, z_i) \le 1$, where $\sum_{i=1}^{l} DTD(v_j, z_i) = 1$, for every $v_j$.

The objective of VMPP-GDT is to find a mapping function $f: VM \to RI$ to assign user requests to RIs in order to maximize the total profit of the broker without degrading the Quality of Service (QoS):

$$\max_j \sum_{j=1}^{m} \Bigg( \sum_{i:f(v_i)=b_j} \big(p(BF(v_i) - C(b_j)\big) \times T(v_i) + \sum_{h:ST(v_h)>D(v_h)} \big(p\big(BF(v_h)\big) - COD(BF(v_h))\big) \times T(v_h) + \sum_{i=1}^{n} \sum_{k=1}^{l} DTD(v_i, z_k) \times DTC(f(v_i), z_k) \times DT(v_i) \Bigg),$$

and subject to constraints

$$M(v_i) \le M(b_j), \qquad P(v_i) \le P(b_j),$$
$$S(v_i) \le S(b_j), \qquad nc(v_i) \le nc(b_j),$$

where $BF(v_k)$ gives the less expensive RI capable of executing the request $v_k$.

In our experiments, we consider the following geographical zones: South America, North America, Europe, Asia, and Oceania. The location is determined when the virtual machine is booked. On demand VMs can be located in and reallocated to any zone according to the broker decision.

## IV. The Proposed Scheduling Heuristics

We propose six heuristics extending ideas presented in our previous work [16] by considering VMs and users' localization, application types, and data transfer costs. Heuristics are conceived to provide a trade-off solution between the two groups of evaluation criteria: user centric and system centric. User centric evaluation criteria such as mean turnaround time, response time, and waiting time help to evaluate algorithms from the perspective of the satisfaction of users. System centric evaluation criteria (processor utilization, throughput, profit) help assess algorithms from the viewpoint of the system use.

We consider two optimization criteria: the broker profit, and the mean response time (i.e., the time spent between the start time and completion time of a VM request).

The proposed heuristics are:

- *Shortest Task First* (STF) gives priority to VM requests with shortest execution times (like the well-known STF strategy for response time minimization). The heuristic searches for the shortest unallocated VM request and lowest-cost RI that fulfills its hardware requirements. Unlike the previous versions of the heuristic [16][5], the assignment costs includes geolocation and data transfer.
- *Best Fit Resource* (BFR) assigns each VM request to the best-fit RI, which is defined as an RI with the closest number of requested cores, and closest amount of requested memory. This method tries to take advantage of assigning the requests to those RIs that fit the most, in order to make room for most restrictive requests to be executed in larger RIs.
- *Earliest Finish Time* (EFT) gives priority to those VM requests that can be finished the soonest. The availability of each suitable booked instance for the request is considered to compute the finish time. The main idea behind this heuristic is to take advantage of executing the fastest requests to increase the availability of RIs.
- *Earliest Deadline First* (EDF) gives priority to VM requests with the earliest deadlines, and assigns each request to the suitable instance with the earliest availability. The main idea is to take advantage of the early execution of the most restrictive requests regarding the deadline values trying to avoid the cost penalization of buying on-demand instances due to deadline violations.
- *Cheapest Instance* (CI) sorts VM requests by arrival time, and selects the cheapest RI that allows the execution of each request, similar to the well-known First Come First Served scheduling method. This heuristic aims at reducing the average response time, and cost due to booking and data transfer.
- *Shortest Request to Cheapest Instance* (SRCI) sorts VM requests by duration and selects the cheapest instance that allows the execution of each request, like the well-known *Shortest Job to the Fastest Resource* heuristic for response time optimization. Unlike the previous versions of this heuristic [16][5], geolocation and data transfer are considered to evaluate the assignment costs. As SJFR, the SRCI heuristic is intended to maximize the broker profit, as well as to minimize the mean response time. Shortest requests are assigned first, so they will be finished earlier, and users with short computing time demands will find that their requests are completed fast.

## V. Experimental Analysis

This section presents the experimental evaluation of the proposed algorithms.

### A. Problem instances

We developed a set of VMMP-GDT instances using real VM characteristics and pricing gathered from the Azure and Amazon cloud service websites. We constructed these instances modeling real workload data from our High Performance Computing facility [17] and from the Parallel Workload Archive [18].

A number of 400 problem instances are solved in the experimental analysis by combining 20 workloads and 20 RI scenarios with diverse characteristics. Each instance is defined by a workload and scenario.

a) *Workloads*: A workload defines a set of VMs requested by users. Each request includes: application type (CT or WS), memory and storage, processor speed, number of cores, data transfer estimation and the user geographical location.

We consider batches of 50, 100, 200, and 400 requests, with durations of 10–200 time units, and an application type rate of 70% CT and 30% WS. We defined two data transfer categories: High (50–250 GB), and Low (10–20 GB). Requests arrive according to a Poisson process per time unit, and we defined deadlines according to real data from grid and cloud logs.

b) *Scenarios*: Each scenario describes the set of RIs pre-booked by the broker described by available memory and storage, processor speed, number of cores, location zone and the cost (both pre-booked and on-demand) and price values.

We consider scenarios with 10, 20, 30, and 50 RIs by combining VMs from Amazon and Azure cloud computing services. The detailed characteristics are presented in Table I (price and costs are in US dollars, updated to May, 2016). We consider different configurations including small and average VMs (instances #1 to #3), large VMs (instances #4 and #6), and instances with large memory, CPU and/or storage (instances #5, #7 and #8). All scenarios were created randomly; we do not analyze the impact of the VM characteristics in the scenario.

TABLE I. TYPES FOR PRE-BOOKED VMS INSTANCES OF THE VIRTUAL BROKER

| # | VM ID | M(GB) | S (GB) | P(GHz) | nc | price | C | COD |
|---|---|---|---|---|---|---|---|---|
| 1 | m1.small (Amazon) | 1.7 | 160 | 1.0 | 1 | 0.048 | 0.027 | 0.06 |
| 2 | m1.medium (Amazon) | 3.75 | 410 | 2.0 | 2 | 0.096 | 0.054 | 0.12 |
| 3 | A2.medium (Azure) | 3.5 | 489 | 1.6 | 2 | 0.096 | 0.09 | 0.12 |
| 4 | m1.large (Amazon) | 7.5 | 850 | 2.0 | 4 | 0.192 | 0.108 | 0.24 |
| 5 | m2.xlarge (Amazon) | 17.1 | 420 | 3.25 | 2 | 0.192 | 0.136 | 0.24 |
| 6 | A3.large (Azure) | 7.0 | 999 | 1.6 | 4 | 0.328 | 0.18 | 0.41 |
| 7 | c1.xlarge (Amazon) | 7.0 | 1690 | 2.5 | 8 | 0.384 | 0.316 | 0.48 |
| 8 | A4.xlarge (Azure) | 14.0 | 2039 | 1.6 | 8 | 0.464 | 0.36 | 0.58 |

Our pricing policy considers that the price per time unit of a specific RI from the broker is 20% cheaper than the price of on-demand ($p(b_j) = 0.8 \times COD(b_j)$). This is a reasonable value for attracting users to the service, while obtaining reasonable profit values. The generated VMMP-GDT instances are available at http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP.

### B. Development and execution platform

The proposed heuristics are implemented in C using stdlib and GNU gcc. The experimental analysis is performed on a 24-Core AMD Opteron Magny-Cours Processor 6172 at 2.1GHz, with 24 GB RAM and CentOS Linux, from the Cluster FING HPC facility [17] from Universidad de la República, Uruguay (Cluster FING website: http://www.fing.edu.uy/cluster).

### C. Experimental results

We study the *profit* of the broker expressed in monetary units (US dollars) for each batch, which is the main objective of the optimization problem formulated in Section II. We also analyzed some QoS-oriented metrics and performed a comparison with state-of-the-art heuristic methods.

*Profit*. The average and best profit computed by each scheduling heuristic and the relative difference when comparing against the best result for each problem instance are reported in Table II (low data transfer) and Table III (high data transfer). We also report the GAP metric defined by Equation 1, where $best\ profit$ is the best profit value found by any heuristic for a specific problem instance, and $profit_H$ is the profit value obtained by a specific heuristic *H*.

$$GAP = \frac{best\ profit - profit_H}{best\ profit} \quad (1)$$

In addition, two other relevant metrics are reported: *#1* is the number of times that each heuristic is the best among the six heuristics compared, and *negative* is the number of times that a negative value is computed for the profit (i.e., the broker loses money when applying that strategy). The best result computed for each metric and each problem dimension is marked in bold font.

The results reported in Table II indicate CI is the best for VM planning in moderately loaded situations (50 and 100 tasks) and low data transfer. However, SRCI outperforms CI and computes the best results when considering a larger number of tasks. The results in Table III show a different situation when the applications make intensive use of data transfer: the SRCI heuristic consistently computes the best profit results for workloads with 100, 200 and 400 requests. In this situation, CI is the second best options.

SRCI and CI always outperform BFR, EDF, STF, and EFT in low data transfer scenarios, computing a profit gap difference between 46.3% up to 96.1%. In high data transfer scenarios the profit gap difference is much lower, especially for small scenarios, ranging between 2.2% and 20.2%. Nevertheless, BFR, EDF, STF, and EFT are unable to outperform SRCI and CI in any scenario.

TABLE II. PROFIT COMPARISON OF THE PROPOSED METHODS FOR LOW-TRANSFER SCENARIOS

| #W | profit | results | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | avg. | 80.4 | 42.8 | **81.8** | 44.0 | 36.4 | 35.4 |
| | best | **107.7** | 82.4 | **107.7** | 76.6 | 78.3 | 78.9 |
| | avg GAP | 2.5% | 50.6% | **0.7%** | 48.8% | 56.7% | 59.3% |
| | #1 | 54 | 0 | **73** | 0 | 0 | 0 |
| | negative | **0** | **0** | **0** | **0** | 1 | 6 |
| 100 | avg. | 139.0 | 48.9 | **142.5** | 51.3 | 33.1 | 35.5 |
| | best | **196.1** | 114.3 | **196.1** | 105.3 | 71.6 | 107.1 |
| | avg GAP | 2.9% | 67.6% | **0.6%** | 65.3% | 77.4% | 77.3% |
| | #1 | 25 | 0 | **78** | 0 | 0 | 0 |
| | negative | **0** | 5 | **0** | 4 | 5 | 10 |
| 200 | avg. | **219.0** | 87.2 | 218.3 | 89.2 | 85.6 | 75.8 |
| | best | **328.8** | 165.0 | 328.4 | 168.8 | 147.0 | 126.9 |
| | avg GAP | **1.2%** | 61.7% | 2.1% | 60.9% | 59.9% | 65.6% |
| | #1 | **55** | 0 | 46 | 0 | 0 | 0 |
| | negative | **0** | **0** | **0** | 2 | **0** | **0** |
| 400 | avg. | **273.5** | 76.7 | 257.8 | 87.4 | 91.7 | 73.2 |
| | best | 539.7 | 248.4 | **577.8** | 227.3 | 248.7 | 201.6 |
| | avg GAP | **0.4%** | 96.5% | 18.7% | 88.0% | 75.5% | 85.2% |
| | #1 | **80** | 0 | 21 | 0 | 0 | 0 |
| | negative | **0** | 25 | **0** | 21 | 14 | 20 |

TABLE III. PROFIT COMPARISON OF THE PROPOSED METHODS FOR HIGH-TRANSFER SCENARIOS

| #W | profit | results | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | avg. | **603.4** | 584.0 | 597.5 | 588.3 | 573.6 | 566.9 |
| | best | 737.0 | 703.4 | **746.8** | 714.8 | 714.0 | 717.9 |
| | avg GAP | **1.9%** | 5.0% | 2.9% | 4.1% | 6.5% | 7.9% |
| | #1 | **52** | 12 | 45 | 13 | 3 | 1 |
| | negative | **0** | **0** | **0** | **0** | **0** | **0** |
| 100 | avg. | **1102.7** | 1020.5 | 1085.7 | 1045.5 | 1014.1 | 985.8 |
| | Best | **1389.3** | 1355.1 | 1391.0 | 1364.1 | 1310.8 | 1335.5 |
| | avg GAP | **1.7%** | 10.2% | 3.8% | 7.1% | 9.9% | 12.6% |
| | #1 | **47** | 5 | 36 | 15 | 0 | 0 |
| | negative | **0** | **0** | **0** | **0** | **0** | **0** |
| 200 | avg. | **1977.1** | 1842.7 | 1895.2 | 1865.8 | 1919.2 | 1805.3 |
| | best | 2539.7 | 2523.4 | 2557.5 | **2566.2** | 2500.4 | 2429.8 |
| | avg GAP | **1.8%** | 10.1% | 6.9% | 8.8% | 5.0% | 10.9% |
| | #1 | **56** | 8 | 20 | 12 | 5 | 0 |
| | negative | **0** | **0** | **0** | **0** | **0** | **0** |
| 400 | avg. | **2832.9** | 2367.5 | 2497.3 | 2485.2 | 2699.5 | 2466.7 |
| | best | **4405.9** | 4169.2 | 4321.1 | 4310.6 | 4253.9 | 3901.8 |
| | avg GAP | **0.7%** | 20.9% | 15.9% | 15.4% | 5.5% | 13.6% |
| | #1 | **82** | 1 | 0 | 1 | 16 | 0 |
| | negative | **0** | **0** | **0** | **0** | **0** | **0** |

Results show low data transfer scenarios are less profitable than high data transfer scenarios. That is, when dealing with high data transfer scenarios, all heuristics are always able to compute profitable schedules. However, when dealing with low data transfer scenarios, SRCI and CI are the only heuristics capable of consistently computing profitable results for all scenarios (even in the most heavily loaded scenarios). Table II shows BFR, EDF, STF, and EFT compute between 14% and 25% of unprofitable schedules when scheduling workloads with 400 VM requests, meaning the broker loses money in these scenarios.

Furthermore, these results suggest that assigning each request to the cheapest RI capable of processing it is a simple but efficient strategy for moderately loaded RI infrastructures with low data transfer. A different behavior is observed when dealing with an overloaded RI infrastructure or with high data transfer scenarios. In these cases, assigning the shortest tasks first to the cheapest RI resources allows the broker to improve the profit results up to 18.3% when compared to the CI heuristic. These results match our findings in [5] where we do not consider task types or geolocation. Figure 1 shows the number of times that SRCI and CI compute the best profit for all studied instances.

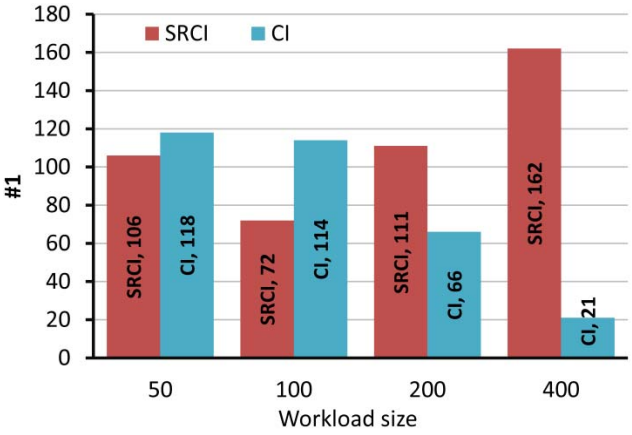


Figure 1: Number of times SRCI and CI compute the best profit (out of 200 instances)

Table IV presents the results of the Friedman statistical test for the profit results of the proposed methods. Table IV shows that SRCI is significantly better than the rest in heavily loaded scenarios with high data transfer rate. On the other hand, CI is the best in moderately loaded scenarios with low data transfer rate.

TABLE IV. MEAN RANKS AND P-VALUE RESULTS OF THE FRIEDMAN TEST OF THE PROPOSED IN HIGH- AND LOW- TRANSFER SCENARIOS

| #W | data rate | Mean rank (# pvalue < 0.01 out of 4) | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | low | 1.6 (0) | 4.0 (0) | 1.4 (0) | 3.7 (0) | 5.1 (0) | 5.2 (0) |
| | high | 1.8 (0) | 3.8 (0) | 2.3 (0) | 3.5 (0) | 4.7 (0) | 4.9 (0) |
| 100 | low | 1.8 (0) | 4.0 (0) | **1.2 (3)** | 3.9 (0) | 5.1 (0) | 5.0 (0) |
| | high | **1.8 (1)** | 4.0 (0) | 2.1 (0) | 3.4 (0) | 4.4 (0) | 5.3 (0) |
| 200 | low | **1.5 (1)** | 4.4 (0) | **1.5 (1)** | 4.2 (0) | 4.2 (0) | 5.2 (0) |
| | high | **1.8 (3)** | 4.4 (0) | 2.9 (0) | 3.8 (0) | 3.1 (0) | 5.1 (0) |
| 400 | low | **1.3 (2)** | 4.8 (0) | 1.7 (1) | 4.2 (0) | 3.9 (0) | 5.2 (0) |
| | high | **1.2 (4)** | 5.4 (0) | 3.6 (0) | 4.0 (0) | 2.3 (0) | 4.4 (0) |

*QoS metrics*. Next, we study the QoS of the computed schedules. We define the QoS of a schedule by means of the waiting time of its VM requests. The waiting time $w_i$ of request $v_i$ is defined by $w_i = ST(v_i) - A_i$, with $A_i$ being the arrival time and $ST(v_i)$ the starting time of request $v_i$. The shorter the waiting time of a VM request, the better its QoS.

Table V reports the average, standard deviation, and worst waiting time computed by each scheduling heuristic. In metric *#1* we report the number of times each heuristic computes the schedule with the lowest total waiting time. The best results are presented in bold font.

Results in Table V show SRCI outperforms all other heuristics in every scenario, always computing the lowest average waiting time and computing the lowest total waiting time more times than any other heuristic. These results show that scheduling the shortest results first significantly increases the QoS of the schedule and allows SRCI to outperform the others in every scenario. Furthermore, it is worth noting that VM requests assigned to on-demand instances have no waiting time and the best QoS, only requests assigned to RI may have a waiting time. Hence, it is most remarkable that SRCI maximizes the profit (i.e., minimizes the usage of on demand instances), while also successfully maximizes the QoS (i.e., minimizes the waiting time of the VM requests).

TABLE V. WAITING TIME COMPARISON OF THE PROPOSED METHODS

| #W | waiting time | results | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | worst | 17.7 | 22.2 | **17.4** | 24.1 | 23.2 | 23.6 |
| | avg | **8.3** | 10.9 | 11.2 | 11.1 | 11.5 | 12.3 |
| | std. dev. | **2.7** | 5.4 | 2.7 | 5.8 | 5.7 | 6.2 |
| | #1 | **126** | 21 | 15 | 27 | 7 | 11 |
| 100 | worst | **15.3** | 19.4 | 18.5 | 21.2 | 21.6 | 24.5 |
| | avg | **9.8** | 13.5 | 13.3 | 15.6 | 16.0 | 17.7 |
| | std. dev. | 2.5 | 2.1 | 2.3 | 2.7 | **2.0** | 3.1 |
| | #1 | **165** | 25 | 11 | 0 | 0 | 0 |
| 200 | worst | **17.5** | 24.2 | 23.3 | 26.7 | 24.0 | 25.9 |
| | avg | **12.2** | 17.6 | 17.0 | 20.2 | 20.2 | 20.9 |
| | std. dev. | **1.5** | 3.2 | 2.5 | 3.2 | 2.3 | 2.3 |
| | #1 | **184** | 19 | 3 | 0 | 0 | 0 |
| 400 | worst | **15.4** | 20.8 | 21.2 | 22.5 | 23.4 | 24.3 |
| | avg | **11.8** | 13.5 | 14.1 | 16.0 | 16.8 | 17.6 |
| | std. dev. | **2.1** | 4.4 | 3.9 | 4.1 | 3.7 | 4.1 |
| | #1 | **126** | 75 | 3 | 0 | 0 | 0 |

*Application type analysis*. We also analyze the on-demand cost of the computation tasks (CT) and web services (WS) for the computed schedules. The on-demand cost metric is the economic cost the broker must pay for renting on-demand instances to satisfy all VM requests. This metric is very relevant for the broker because it is a cost which is out of the planned budget, and because on-demand instances generate no profit.

In Table VI (low data transfer) and Table VII (high data transfer) we report the average on-demand cost for satisfying CT requests, and the average on-demand cost for satisfying WS requests for each scheduling heuristic. In metric *#1* we report the number of times each heuristic computes the schedule with the lowest total on-demand cost. The best results are in bold font.

TABLE VI. ON-DEMAND COST COMPARISON OF THE PROPOSED METHODS FOR EACH APPLICATION TYPE IN LOW-TRANSFER SCENARIOS

| #W | on-demand cost | results | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | avg CT | 1.2 | 2.2 | **1.1** | 2.5 | 2.9 | 1.9 |
| | avg WS | 3.6 | 3.7 | **3.0** | 3.2 | 4.5 | 5.4 |
| | #1 | 32 | 37 | 39 | **46** | 33 | 32 |
| 100 | avg CT | 7.4 | 11.7 | **5.8** | 11.5 | 14.6 | 10.1 |
| | avg WS | 5.9 | 7.9 | **5.4** | 8.1 | 8.7 | 12.9 |
| | #1 | 17 | 8 | **41** | 18 | 8 | 4 |
| 200 | avg CT | 12.2 | 19.6 | **10.7** | 18.7 | 22.0 | 18.1 |
| | avg WS | 9.9 | 12.2 | **8.2** | 13.3 | 14.5 | 18.4 |
| | #1 | 4 | 1 | **64** | 0 | 0 | 0 |
| 400 | avg CT | 63.6 | 80.9 | **59.4** | 82.6 | 87.1 | 76.7 |
| | avg WS | 35.1 | 38.6 | **31.7** | 39.1 | 43.4 | 51.7 |
| | #1 | 0 | 0 | **89** | 0 | 0 | 0 |

TABLE VII. ON-DEMAND COST COMPARISON OF THE PROPOSED METHODS FOR EACH APPLICATION TYPE IN HIGH-TRANSFER SCENARIOS

| #W | on-demand cost | Results | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***SRCI*** | ***BFR*** | ***CI*** | ***EDF*** | ***STF*** | ***EFT*** |
| 50 | avg CT | 1.6 | 2.2 | **1.5** | 2.4 | 3.0 | 1.8 |
| | avg WS | 3.8 | 3.5 | **3.1** | 3.2 | 4.7 | 5.4 |
| | #1 | 33 | 38 | 37 | **44** | 35 | 35 |
| 100 | avg CT | 8.7 | 12.0 | **7.0** | 11.8 | 14.9 | 10.3 |
| | avg WS | 6.1 | 8.1 | **5.8** | 8.0 | 9.0 | 13.0 |
| | #1 | 17 | 10 | **37** | 17 | 8 | 4 |
| 200 | avg CT | 14.1 | 20.1 | **12.7** | 19.4 | 22.6 | 18.7 |
| | avg WS | 10.7 | 12.4 | **8.9** | 13.6 | 15.0 | 19.2 |
| | #1 | 3 | 2 | **57** | 0 | 0 | 0 |
| 400 | avg CT | 69.5 | 85.1 | **65.6** | 86.9 | 91.5 | 80.1 |
| | avg WS | 39.2 | 41.8 | **35.5** | 42.7 | 46.8 | 56.4 |
| | #1 | 0 | 0 | **85** | 0 | 0 | 0 |

Results in Table VI and VII indicate that CI outperforms all other heuristics, computing the lowest average and total on-demand cost in almost every scenario, especially in heavily loaded situations. When analyzing the on-demand cost by application type, results show CT and WS applications contribute evenly−with around 50% each−to the total on-demand cost, despite having a ratio 3 to 1 more CT than WS requests. This is expected considering WS applications have more constrained execution deadlines; hence they are scheduled using on-demand resources more frequently than CT applications. Although CI minimizes on-demand cost, it is SRCI which is able to maximize the overall profit. This issue is likely related to data transfer costs and should be further studied.

### D. Comparision with Queue scanning

Queue Scanning (QS) is an online scheduling heuristic introduced by Van den Bossche et al. [15]. It was proposed for solving the problem of scheduling VM requests in a hybrid infrastructure combining a private and a public cloud. The QS algorithm is a simple and effective online method that can be combined with different scheduling policies for managing VM requests. Van den Bossche et al. proposed two policies for combining with QS: Unfeasible-To-Public (UTP) and Cheapest-To-Public (CTP). When using the UTP strategy, unfeasible requests are simply redirected to the public cloud. On the other hand, when using the CTP strategy, the algorithm tries to schedule large expensive unfeasible requests to the private cloud by rescheduling small cheap requests from the private cloud to the public cloud.

The problem proposed by Van den Bossche et al. [15] is similar to the VMPP-GDT we address in this article. Thus, the QS algorithm is suited for solving our problem when considering the broker RIs as a virtual private cloud. In this subsection we compare the profit results computed by SRCI and CI, the most accurate of our schedulers, with those computed by QS-UTP and QS-CTP. Tables VIII and IX summarize the comparison, reporting the average, best, and average GAP profit, the number of times each algorithm computes the best profit in the comparison, and the number of times each algorithm computes a non-profitable schedule.

TABLE VIII. PROFIT COMPARISION PROFIT COMPARISION: SRCI AND CI VS. QUEUE SCANNING [15] FOR LOW-TRANSFER SCENARIOS

| #W | profit | result | | | |
|---|---|---|---|---|---|
| | | ***SRCI*** | ***CI*** | ***QS-UTP*** | ***QS-CTP*** |
| 50 | avg. | 80.4 | **81.8** | 81.4 | 80.0 |
| | best | **107.7** | **107.7** | **107.7** | **107.7** |
| | avg GAP | 3.2% | **1.5%** | 1.9% | 3.8% |
| | #1 | 49 | **50** | 49 | 46 |
| | negative | **0** | **0** | **0** | **0** |
| 100 | avg. | 139.0 | 142.5 | **143.2** | 137.7 |
| | best | **196.1** | **196.1** | **196.1** | **196.1** |
| | avg GAP | 4.5% | 2.2% | **1.2%** | 6.0% |
| | #1 | 16 | 43 | **45** | 12 |
| | negative | **0** | **0** | **0** | **0** |
| 200 | avg. | **219.0** | 218.3 | 211.9 | 213.2 |
| | best | **328.8** | **328.4** | 319.6 | 325.2 |
| | avg GAP | **1.2%** | 2.1% | 5.7% | 4.4% |
| | #1 | **52** | 45 | 3 | 4 |
| | negative | **0** | **0** | **0** | **0** |
| 400 | avg. | **273.5** | 257.8 | 251.9 | 261.3 |
| | best | 539.7 | **577.8** | 552.0 | 536.4 |
| | avg GAP | **0.5%** | 18.8% | 22.4% | 9.7% |
| | #1 | **75** | 18 | 8 | 1 |
| | negative | **0** | **0** | **0** | **0** |

TABLE IX. PROFIT COMPARISION: SRCI AND CI VS. QUEUE SCANNING [15] FOR HIGH-TRANSFER SCENARIOS

| #W | profit | result | | | |
|---|---|---|---|---|---|
| | | ***SRCI*** | ***CI*** | ***QS-UTP*** | ***QS-CTP*** |
| 50 | avg. | **603.4** | 597.5 | 591.0 | 597.5 |
| | best | 737.0 | **746.8** | 737.0 | 737.0 |
| | avg GAP | **0.9%** | 2.0% | 3.2% | 2.0% |
| | #1 | **61** | 42 | 32 | 51 |
| | negative | **0** | **0** | **0** | **0** |
| 100 | avg. | **1102.7** | 1085.7 | 1078.8 | 1079.8 |
| | best | 1389.3 | **1391.0** | 1380.6 | 1389.3 |
| | avg GAP | **0.8%** | 2.9% | 3.3% | 3.2% |
| | #1 | **47** | 33 | 15 | 16 |
| | negative | **0** | **0** | **0** | **0** |
| 200 | avg. | **1977.1** | 1895.2 | 1837.8 | 1932.8 |
| | best | 2539.7 | **2557.5** | 2550.9 | 2539.7 |
| | avg GAP | **0.3%** | 5.4% | 8.5% | 2.7% |
| | #1 | **71** | 18 | 2 | 9 |
| | negative | **0** | **0** | **0** | **0** |
| 400 | avg. | **2832.9** | 2497.3 | 2398.6 | 2719.1 |
| | best | **4405.9** | 4321.1 | 4172.1 | 4172.0 |
| | avg GAP | **0.0%** | 15.2% | 18.8% | 4.2% |
| | #1 | **96** | 0 | 0 | 4 |
| | negative | **0** | **0** | **0** | **0** |

Results show QS-UTP and CI are the best schedulers for situations with light load and with low data transfer. However, SRCI is the heuristic that consistently allows computing the most profitable schedules in every other scenario. SRCI computes the best schedule as much as 96 out of 100 instances in high data transfer scenarios. Table X shows the results of the Friedman test for these results. These results show SRCI is indeed significantly more accurate than the rest of the proposed schedulers when addressing large workloads of tasks.

TABLE X. MEAN RANKS AND P-VALUE RESULTS OF THE FRIEDMAN RANK TEST FOR SRCI AND CI VS. QUEUE SCANNING [15]

| #W | transfer scenario | mean rank (# pvalue < 0.01 out of 4) | | | |
|---|---|---|---|---|---|
| | | ***SRCI*** | ***CI*** | ***QS-UTP*** | ***QS-CTP*** |
| 50 | low | 4.6 (0) | 4.4 (0) | 4.3 (0) | 4.7 (0) |
| | high | 4.2 (0) | 4.5 (0) | 4.9 (0) | 4.4 (0) |
| 100 | low | 5.0 (0) | 3.9 (0) | 4.2 (0) | 5.0 (0) |
| | high | **4.1 (1)** | 4.5 (0) | 5.0 (0) | 4.4 (0) |
| 200 | low | **4.0 (1)** | **4.1 (1)** | 5.2 (0) | 4.7 (0) |
| | high | **3.6 (2)** | 4.7 (0) | 5.7 (0) | 3.9 (0) |
| 400 | low | **3.6 (2)** | 4.6 (1) | 5.4 (0) | 4.5 (0) |
| | high | **3.2 (3)** | 5.1 (0) | 5.9 (0) | 3.8 (0) |

Figure 2 summarizes the number of times each heuristic computes the best profit.

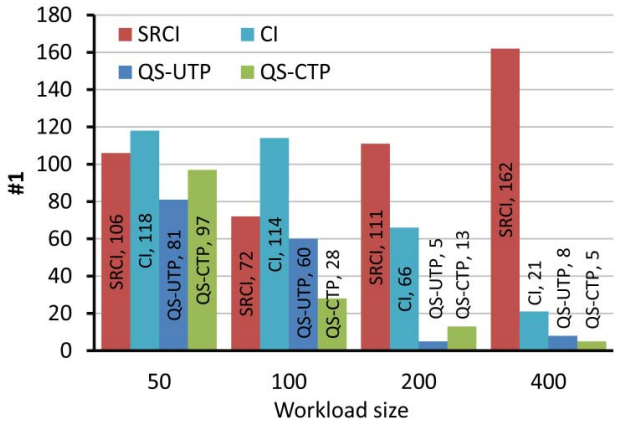


Figure 2: Number of times SRCI, CI, and QS heuristics compute the best profit values (out of 200 instances)

## VI. CONCLUSIONS

In this article we address the virtual machine planning problem that takes into account geographical localization of users and resources, data transfer costs, and two kinds of applications. We present six heuristics for optimizing the use of virtual machines that a broker owns and outsources to its customers.

The main results of the experimental evaluation indicate that SRCI is the best scheduler for VM planning in heavily-loaded low-transfer scenarios and in all high-transfer scenarios, computing the schedules with the best profit and QoS. Conversely, in lightly-loaded low-transfer scenarios, CI is the one which computes the best results. CI outperforms SRCI in profit values; nevertheless SRCI is still able to compute schedules with the best QoS values.

We performed a comparison of the best schedulers proposed in this article against two state-of-the-art heuristics: QS-UTP and QS-CTP, proposed by Van den Bossche et al. [15]. QS-UTP and QS-CTP compute competitive results in lightly-loaded scenarios. However, SRCI and CI quickly outperform both of them when dealing with a larger number of user requests. In heavily-loaded scenarios, SRCI computes the best results in 81% of the cases, while QS-UTP and QS-CTP barely compute the best result between 2% and 4% of the cases. These results confirm that SRCI and CI are accurate heuristics for solving the Virtual Machine Planning Problem with Geolocation and Data Transfers.

The main lines for future work include further studying the proposed methods, especially in overloaded situations; and modeling more features of the users' requests to deal with realistic applications (data, voice, video, etc.).

## ACKNOWLEDGMENTS

S. Iturriaga and S. Nesmachnow are supported by ANII and PEDECIBA, Uruguay. B. Dorronsoro is supported by MINECO, Spain (TIN2014-60844-R and RYC-2013-13355). A. Tchernykh is partially supported by CONACYT, grant no. 178415, Mexico.

## REFERENCES

[1] R. Buyya, J. Broberg, and A. Goscinski, Cloud Computing: Principles and Paradigms. Wiley, 2011.

[2] I. Foster, Y. Zhao, and S. Lu, “Cloud computing and grid computing 360-degree compared,” Grid Comp. Environments Workshop, 2008, pp. 1–10.

[3] M. Hu, J. Luo, Y. Wang, and B. Veeravalli, “Practical resource provisioning and caching with dynamic resilience for cloud-based content distribution networks,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 8, pp. 2169–2179, 2014.

[4] U. Schwiegelshohn, A. Tchernykh, “Online Scheduling for Cloud Computing and Different Service Levels,” IEEE 26th Int. Parallel and Distrib. Proc. Symposium Workshops & PhD Forum, pp. 1067-1074, 2012

[5] S. Nesmachnow, S. Iturriaga, and B. Dorronsoro, “Efficient heuristics for profit optimization of virtual cloud brokers,” IEEE Comp. Intelligence Magazine, vol. 10, no. 1, pp. 33–43, 2015.

[6] S. Nesmachnow, S. Iturriaga, B. Dorronsoro, E.-G. Talbi, and P. Bouvry, “Metaheuristics for the virtual machine mapping problem in clouds,” Informatica, vol. 26, no. 1, pp. 111–134, 2015.

[7] N. Grozev and R. Buyya, “Inter-cloud architectures and application brokering: Taxonomy and survey,” Software: Practice and Experience, vol. 33, no. 3, pp. 369–390, 2014.

[8] O. Rogers and D. Cliff, “A financial brokerage model for cloud computing,” J. of Cloud Comp.: Advances, Syst. and Applications, vol. 1, no. 2, pp. 1–12, 2012.

[9] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack Problems. Springer-Verlag, 2004.

[10] W. Wang, B. Liang and B. Li, "Optimal Online Multi-Instance Acquisition in IaaS Clouds," in IEEE Trans. on Parallel and Distrib. Syst., vol. 26, no. 12, pp. 3407–3419, 2015.

[11] R. H. Hwang, C. N. Lee, Y. R. Chen and D. J. Zhang-Jian, "Cost Optimization of Elasticity Cloud Resource Subscription Policy," in IEEE Trans. on Services Comp., vol. 7, no. 4, pp. 561–574, 2014.

[12] J. Zhang, S. Chen, H. Huang, X. Wang, D. Du, "Dynamic Resource Provision for Cloud Broker with Multiple Reserved Instance Terms," 15th Int. Conf. on Algorithms and Arch. for Parallel Proc., 2015, pp. 339–356.

[13] A. Alasaad, K. Shafiee, H. M. Behairy and V. C. M. Leung, "Innovative Schemes for Resource Allocation in the Cloud for Media Streaming Applications," IEEE Trans. on Parallel and Distrib. Syst., vol. 26, no. 4, pp. 1021–1033, 2015.

[14] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang and M. R. T. Lyu, "Towards Operational Cost Minimization in Hybrid Clouds for Dynamic Resource Provisioning with Delay-Aware Optimization," in IEEE Trans. on Services Comp., vol. 8, no. 3, pp. 398–409, 2015.

[15] R. Van den Bossche, K. Vanmechelen, J. Broeckhove, “Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds,” in Future Generation Computer Syst., vol. 29, no. 4, pp. 973–985, 2013.

[16] S. Nesmachnow, S. Iturriaga, B. Dorronsoro, E.-G. Talbi, and P. Bouvry, “List scheduling heuristics for virtual machine mapping in cloud systems,” VI High Performance Comp. Latin America Symposium, 2013, pp. 37–48.

[17] S. Nesmachnow. "Computación Científica de Alto Desempeño en la Facultad de Ingeniería de la Universidad de la República". Revista de la Asociación de Ingenieros del Uruguay, vol. 61, pp 12–15, 2010 (in Spanish).

[18] D. Feitelson, D. Tsafrir, D. Krakov, “Experience with using the Parallel Workloads Archive,” in J. of Parallel and Distrib. Comp., vol. 74, no. 10, pp. 2967–2982, 2014.

[19] D. Lučanin, I. Brandic, “Pervasive Cloud Controller for Geotemporal Inputs,” in IEEE Trans. on Cloud Computing, vol. 4, no. 2, pp. 180–195, 2016.