Value of information based scheduling of cloud computing resources

Ladislau Bölöni and Damla Turgut
Invited paper

Dept. of Computer Science, University of Central Florida

Abstract

Traditionally, heavy computational tasks were performed on a dedicated infrastructure requiring a heavy initial investment, such as a supercomputer or a data center. Grid computing relaxed the assumptions of the fixed infrastructure, allowing the sharing of remote computational resources. Cloud computing brought these ideas into the commercial realm and allows users to request on demand an essentially unlimited amount of computing power. However, in contrast to previous assumptions, this computing power is metered and billed on an hour-by-hour basis.

In this paper, we are considering applications where the output quality increases with the deployed computational power, a large class including applications ranging from weather prediction to financial modeling. We are proposing a computation scheduling that considers both the financial cost of the computation and the predicted financial benefit of the output, that is, its *value of information* (VoI). We model the proposed approach for an example of analyzing real-estate investment opportunities in a competitive environment. We show that by using the VoI-based scheduling algorithm, we can outperform minimalistic computing approaches, large but fixedly allocated data centers and cloud computing approaches that do not consider the VoI.

Keywords:

1. Introduction

We often forget that computation and networking costs money. We do not see the dollars counted down when we talk on the cellphone, have the hardware depreciation cost displayed on our laptops or the power bill showing up on the screen when we are playing on a gaming PC with a 700 Watt power source (of course, in a room illuminated by a 7W LED lightbulb). Even for high performance computing, where the computation and networking costs can be substantial, financial considerations used to come into picture only at the time of the investment decisions. Once a new supercomputer, data center or Beowulf cluster had been purchased, users were encouraged to utilize them to their maximum capacity. The grid computing model emerging in the late 1990s [1, 2] introduced the ability of requesting computational power on demand, on the analogy of the power grid. Nevertheless, all grid systems had been financed by national research foundations and thus, the objective of maximum utilization remained in place.

Cloud computing introduced a significant change. In some ways, cloud computing is fulfilling the promise of grid computing of providing on demand, on a very short notice, an amount of computational, storage and networking capacity, normally in the form of virtualized resources. For instance, computational capacity can be offered in form of virtual machines or containers. For instance, virtual machine on demand services provide the user with a remotely allocated virtual machine, in which the client can run its own operating system. These services can be *public clouds*, fully managed services offered by a third party vendor, such as in the case of Amazon's EC2 service [3, 4], Rackspace's OpenStack on Demand service or VMWare's vCloud Air service. In these cases, the cloud provider charges in actual dollars for the provided, metered computing capacity. The client makes a request for a computing unit, and in less then a minute, he can log in and start the computation. While computing, the client will pay an hourly fee. Once the computation is terminated, the client discards the virtual machine. Alternatively, companies can create *private clouds* in which they are offering similar services for their internal divisions. This allows companies to more efficiently share their existing computational resources. Large companies have developed their internal software architectures to provide computing on demand (this being, for instance, the casse of Google's Borg system [5]). Other companies can use available open source software to provide computing-on-demand solutions, for instance using OpenStack Nova system [6] for virtual machines on demand, or Apache Mesos or Google's Kubernetes for containers on demand. In Borg, company subdivisions need to purchase computing quotas with actual money, thus the principles of operation are similar to the one in public clouds.

In such systems, the amount of computation that a client can obtain is limited, in principle, only by the client's ability to pay. For instance, a client can pay $168, and run a parallel computation on 10,000 computer cores, a computing power that was out of reach to anyone without a million dollar investment. For variable and unpredictable loads, such as the sudden popularity and just as sudden fading of a game or app, cloud computing might be the only approach that can trace the variations of the computing demand. Of course, if one would use the 10,000 computer

Unit type	Virtual CPUs	Performance (ECU)	Cost per hour
m4.large	2	6.5	$0.12
m4.xlarge	4	13	$0.239
m4.2xlarge	8	26	$0.479
m4.4xlarge	16	53.5	$0.958
m4.10xlarge	40	124.5	$2.394
c4.large	2	8	$0.105
c4.8xlarge	36	132	$1.675
g2.2xlarge	8	26	$0.65
g2.8xlarge	32	104	$2.60

Table 1: The on-demand computation costs on Amazon EC2 at several representative compute units (as of April 2016). The mX models are general purpose, the cX models are compute optimized while the gX are GPU optimized

cores continuously for years, acquiring a private computer center would be cheaper. However, even for companies that own their data center, as Google's Borg example shows, it had been found that the fine grain measuring and accounting can better optimize the use of the computational resources.

1.1. Applications with elastic computational needs

The way we used to think about performing a computation, is that we have a certain input data X, a certain computing algorithm A and we expect that by performing the algorithm on the data we will create the output $B = A(X)$. For a given algorithm and input, there is a fixed set of computing resources C necessary to perform this computation. Depending on the actual hardware performing the computation, this might take a shorter or longer time, it might cost more or less, or use more or less energy.

There is, however, a very large class of applications that can be seen through a different model: by allocating a computational resource C_i to the process, we obtain an output $B_i = A(X, C_i)$. Naturally, if we perform two computations $C_1 > C_2$ then we expect the corresponding B_1 to be "better" than B_2 according to some quality measure. Many popular algorithms used today generate continuous improvements with additional computational power: we can run some more Markov Chain Monte Carlo network, run a particle filter with more particles, train a larger neural network, perform climate modeling at a finer spatial or temporal resolution and so on. There can be many reasons for not finishing a computation to the end: there can be limits on time, on cost, or simply there is no clear stopping point where we can say that the algorithm had finished. Most such methods eventually run in the problem of diminishing returns in the sense that after a while, adding additional computing power will not measurably improve or even change the outputs (for a given input). In this paper we will argue that we do not need to run the computation so far that no change is observed in the output. We can stop much earlier, at the point where further computation is not justified by the value of information that can be obtained from the output.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we work out a detailed model for VoI for analyzing investment opportunities. In Section 4 we discuss several possible scheduling algorithms for allocating computational power to the applications discussed in the previous section. We present the result of some simulation studies in Section 5. We conclude in Section 6.

2. Related work

The problem of scheduling computational and networking resources to computations had seen a significant effort in the last quarter century. The exact optimization criteria, however, changed in function of both the nature of resources and the applications that are running on them. While early batch systems only ensured the sequential execution of a task on a single computational resource, the emergence of distributed computing systems with multiple, usually heterogeneous resources implied that multiple tasks can be allocated in parallel, and decisions need to be made about which resource is the best fit to which task [7]. The optimization criteria on these systems could be the expected time to completion of each task, the overall throughput of the system, as well as various fairness measures. If the executed tasks are actually collections of parallel and interacting tasks (meta-programs or workflows), new challenges had to be taken into consideration, for instance with regards to the scheduling of network connections or the storage of temporary data. A modern example of such a scheduler is the Hadoop scheduler YARN [8].

Cloud computing is a natural development of the previous models of distributed computing. Beyond the technical innovations related to virtualization, software defined networks and so on, cloud computing also represented a new model with regards to the ownership of the resources. In a typical public cloud, the owner of the computing resource is not the beneficiary of the computing applications. On the other hand, the actual beneficiary has little control over the scheduling decisions - for instance the customer of EC2 does not normally know where the virtual machine she requested is actually running.

Scheduling and resource allocation can be seen as an optimization problem, and thus such a wide range of optimization algorithms had been deployed. To sample just several recent examples done in the context of cloud computing, Zheng et al. [9] use a parallel genetic algorithm for the placement of virtual machines in a cloud computing system. Li, Tordsson and Elmroth [10] use linear integer programming to allocate cloud resources on public clouds, taking into account virtual machine migration as well as continuously changing pricing schemes, virtual machine types and performance. Pandey et al. [11] use particle swarm optimization to schedule applications on clouds taking into account both computation and data transfer costs. Abrishami, Naghibzadeh and Epema [12] schedule application workflows to cloud resources by analyzing the workflow as a graph and determining its critical path. Caron, Desprez and Mureşan [13] present an approach to predict the need for cloud resources by using a short-term memory of load patterns.

Whenever we are talking about economic models in terms of cloud or grid computing, we need to distinguish between systems that use a market model for efficient resource allocation versus systems where resources are allocated in exchange

for actual financial resources. It had been found that the introduction of a synthetic economic model can be actually useful even for system where no actual money is exchanged [14, 15]. More recently, Frincu [16] describes an approach for scheduling highly available applications using a cost model (which might not necessarily be actual financial cost).

The financial aspect of cloud scheduling had received comparatively less attention. Bittencourt and Madeira [17] consider scheduling on hybrid clouds, where some of the resources are allocated from the private cloud of the customer, while others are rented on the public cloud. The authors show that their proposed method can reduce the cost while achieving the predefined execution deadlines. Wang, Li, and Liang [18] consider the financial aspects from the point of view of the service provider, who needs to segment its offerings into different classes - the existing classes of the Amazon EC2 product and an auction market. They describe the optimal segmentation model as a Markov Decision Process.

Another aspect of the work described in this paper is that we assume a certain elasticity in the requirements of the user. Hwang et al. [19] describe a system where the customer trades off between long-term reserved resources and on-demand resources. To optimize its allocation between these two resource types the user uses a Kalman filter to predict its future resource needs. Yang et al [20] proposes a cost-aware auto-scaling approach based on a workload predicted using a linear regression model.

Our approach assumes an elastically scalable load where the computational performance of the application can be directly mapped to financial value. This allows us to define the value of information acquired through computation. The concept of value of information had been previously defined in game theory [21]. Recent work applied this concept to various situations where the cost of computation and communication must be traded against its benefits [22, 23, 24, 25].

Alicherry and Lakshman [26] consider resource allocation algorithms for cloud systems distributed over a large number of locations. The cloud users are running applications, for which they require a specific number of virtual machines and specify the communication requirements between them. The main optimization criteria for this paper is to minimize the communication latency between the VMs allocated to the same application. The paper proposes efficient 2-approximation algorithms for selecting the data centers, racks and servers on which the specific VMs will be allocated.

Cucinotta et al [27] assume a complex application that requires both computation and networking, and for whom a specific SLA is provided describing the quality of service requirements. The authors assume that there is a choice of network service providers and cloud (computation) service providers, each with their specific capabilities and prices. The task of the broker is to find an allocation of the application components and their communication paths in such a way that overall quality of service requirements are met. This can be formulated mathematically in the form of a mixed-integer geometric programming optimization problem, where the objective is to minimize the cost while meeting the SLA.

The paper [27] is similar to our approach in the sense that it takes into account the financial cost of operating the cloud system. However the optimization task is different - while [27] optimizes the cost for a given application with fixed needs, in our case we consider applications with elastic needs and optimize the ratio between the cost of computation and value created.

Konstanteli et al. [28] consider an approach where a set of horizontally scalable applications whose requirements might elastically grow or shrink need to be scheduled into a federated cloud. The scheduling is done through a probalistic optimization model which takes into account affinity and anti-affinity rules and optimizes for eco-efficiency and cost. The proposed optimization approach also allows the system to take into account predictions of scaling requirements (for instance based on historical data). The resulting system can be described through a mixed-integer linear programming problem. This paper shares with our paper the fact that it considers applications that have elastic requirements. In [28] the optimization is taken place outside the internal logic of the application: the elasticity is considered to be due to external factors. In contrast in our approach, we are taking the perspective of the application: an application might scale up because computation is momentarily cheap and the benefits of the computation can be high.

3. Application model: analyzing real-estate investment opportunities

3.1. Notation and terminology

Table 2 summarizes our terminology and notations used throughout the paper.

3.2. Investment opportunity at single timepoint

The scenario we are considering is that of an investor analyzing investment opportunities. Let us assume that the investor is offered to buy an asset for the price V_{cost}. Let us assume that the investment horizon is a year, after which the investor can sell the investment for V_{future}. The alternative of buying an asset is for the investor to put the money into a safe investment choice which guarantees a return of R_{safe}. If the investor spent C_{cost} on computation to analyze the investment opportunity, and took the opportunity, his profit will be:

$$Pr_{invest} = V_{future} - V_{cost} - C_{cost} \tag{1}$$

If he did not ta the opportunity, his profit would be:

$$Pr_{decline} = R_{safe} - C_{cost} \tag{2}$$

To understand how the investor will make this decision, we need to understand that at the time of investment, the investor does not know the V_{future} value, only an estimate V_{est}, which leads to the corresponding estimate of the invest profit Pr'_{invest}. Thus the investor will calculate

$$\Delta Pr' = Pr'_{invest} - Pr_{decline} = V_{est} - V_{cost} - R_{safe} \tag{3}$$

A simple decision model would be for an investor to invest if $\Delta Pr' > 0$ and to decline otherwise. Note that the investment

Table 2: Notation and Terminology

V_{cost}	the purchase cost of the investment
V_{future}	the future value of the investment
V_{est}	the estimated future value of the investment
V_{high}	the high estimate of the future value of the investment
V_{low}	the low estimate of the future value of the investment
R_{safe}	the safe return on the investment
C	the amount of computation analyzing the investment (in ECU hour)
C_{cost}	the cost of computation analyzing the investment
Pr_{invest}	profit if investing
$Pr_{decline}$	profit if not investing
σ	the standard deviation of the estimated future value
k	the confidence level for investment decision level ($k \cdot \sigma$)
k_1, k_2 and k_3	constants in the function $\sigma(C)$
p_{loss}	probability of loosing a pending opportunity to the competitor, a metric of competition intensity

decision does not depend on the expended computational cost, although that comes into the calculation of the actual profit. At the time of expenditures, the computation is a *sunk cost*, which should not influence the outcome (although psychological studies show us that it often does [29]).

The critical variable influencing the decision is the estimate V_{est}. The estimate is a result of computational analysis. Let us consider that the investment opportunity is a piece of real estate. The input of the analysis would be public information available about the real estate (location, square footage, amenities etc.), information about the environment (comparables, commercial potential, traffic patterns), information about the markets present and future (demand for locations, loan interests, taxes, incentives, trends for demand) and private information the investor might hold (eg. customers looking for similar real estate and so on). This information is often available in form of probability chains: if a certain politician is elected as mayor ($p = 0.60$), it will implement a change in the zoning ordinance ($p = 0.80$), which can trigger the sale of the building as a commercial space ($p = 0.75$), increasing its value. Such chains are too complex for a closed form solution, but are well suited to particle filter / Markov Chain Monte Carlo methods. One advantage of such techniques is that they not only generate an estimate of the value, but they also can infer, through the distribution of the particles, the probability distribution of the estimate. Monte Carlo methods have been pioneered in financial risk calculation by Boyle [30]. Several recent examples dealing with property value calculations [31, 32, 33]. For the remainder of this paper, we will make the assumption that the analysis will create a normal probability distribution, which will allow us to describe the uncertainty with a single value, the standard deviation σ[1]. In general, the choice of the normal distribution is a justified choice taken both by the general investment community (eg the Black-Scholes equation for pricing options [34]) as well as it is empirically justified by the profile of simulation result histograms (eg. in [33]).

[1]In analysis which involve binary choices, such as the election of a politician, it can might happen that the distribution will be bi- or multi-modal.

With this assumption, the decision model becomes somewhat more complex. The V_{est} value only denotes the mean of the Gaussian distribution, which means that if $\Delta Pr' = 0$, there will be an even chance of loosing or gaining a profit. Investors typically require a higher confidence in success before they invest. One way to achieve this is to use a "low estimate" for the estimated value, lowering it with a certain number of σs: $V_{low} = V_{est} - k \cdot \sigma$. With this approach, if we choose $k = 1$ we will have a 68% confidence in making a profit, while with $k = 2$ a 95.4% confidence. Similarly we can define a "high estimate" $V_{high} = V_{est} + k \cdot \sigma$.

With this approach, the investor can proceed as follows. It starts by choosing a certain confidence level k - a choice of $k = 2$ yielding a higher than 95% confidence might be a good starting point. The investor performs the computational analysis, yielding a V_{est} and σ, and implicitly, the V_{low} and V_{high}. If the estimated profit using the low estimate is positive, the investor decides to invest. If the estimated profit using the high estimate is negative, the investor declines the investment opportunity. In between these two values there is a $2k\sigma$ range where the investor cannot make a determination with the desired confidence (see Figure 1). If no other options are available, these opportunities will also not be taken.

Of course, this approach raises the question of how large is the σ value. The investor will only take opportunities where the $k \cdot \sigma$ is larger than the expected profit, thus large σs can effectively starve the investor of opportunities to invest. For a given amount of known data, the value of σ depends on the amount of computational power spent on the analysis.

Let us develop a plausible model of the σ function of the amount of computation. In Monte Carlo integration [35], a problem closely related to ours, the error is estimated as $\sigma = c/\sqrt{N}$ where N is the number of tests, proportional with the computational power C. We will need to generalize this formula to take into account some of the specifics of our problem. For a given computational power C (assumed to be measured in ECU-hours) we will assume that the standard deviation takes a form:

$$\sigma = k_1 + \frac{k_2}{1 + C^{\frac{1}{k_3}}} \tag{4}$$

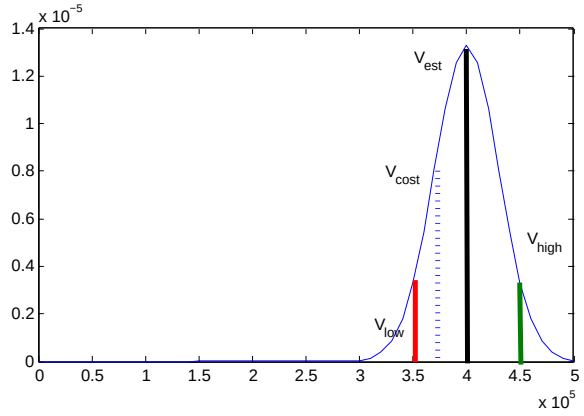


Figure 1: The average, high and low estimates for an investment opportunity (with $k = 1$, versus its cost. In this case, the investor cannot make a determination to invest or not with the desired confidence.

This formula integrates several intuitions about the nature of analysing financial investments. The constant k_1 represents that there are uncertainties that no amount of computation can reduce - these include unknown factors and genuine random outcomes. The constant k_2 represents the amount of uncertainty that *can* be removed by more analysis. Finally, the constant k_3 models the fact that even for the reducible uncertainty, the improvement might not scale linearly with the amount of computation. Note that the formula uses the amount of computation C, not the cost of computation C_{cost}. How much money an investor uses to acquire a certain amount of computation depends on many factors. If he is purchasing them using the Amazon Elastic Compute Cloud, the cost can be calculated based on Table 1. With the computation measured in ECU-hours, if the user runs a c4.large machine (with ECU=8) for 10 hours, the amount of computation is going to be 80 hours.

If he is harvesting unused computing cycles on a company network, the cost can be zero. If a dedicated data center is used, the cost depends on the equipment amortization costs and the cost of maintenance.

One of the questions is: do we need a large amount of computation to make an informed decision, i.e. is C_{cost} at the same order of magnitude with the potential profit? Certainly, it is possible for an offer to be priced such it can be accepted or declined with minimal computation. However, in an efficient market, such offers will be very rare. We assume that the seller is just as competent as the buyers: there is no incentive for the seller to leave money on the table by pricing the offer too low, or to not sell by pricing it too high. In fact, the seller has a strong incentive to price the offer very close to the borderline of the predicted profit or loss. For such situations, the cost of estimating the profit will be in the same order of magnitude with the predicted profit.

3.3. Investment opportunity in time and scheduling the computation

Up to this moment we assumed that the investment opportunity appears as a decision made at a single timepoint. We have seen that by investing a certain amount of computational power C, the user approximates the future value with an uncertainty $\alpha(C)$. This value allows the investor to categorize the investments into three classes: those that clearly should be taken, those that clearly should be declined and those about which no confident decision can be made given the specific $\alpha(C)$.

One way to reduce this uncertainty is to use more computation C_2, with the hope that the uncertainty $\sigma(C + C_2)$ will be small enough such that a confident decision may be made. If a decision still cannot be made, the investor might choose to add additional computation C_3 in the next timestep and so on. Alternatively, the investor might choose to decline the investment opportunity and not spend additional computational resources on it.

Other than choosing its confidence level k, the choice of the *scheduling function* that determines the amount of computation C_t allocated to an opportunity at every timestep is the critical choice that the investor can make in order to impact its profit on the long run[2].

There are several considerations to be taken with regards to the scheduling function. First, allocating an overly large amount of computation immediately after the arrival of the opportunity might allow to make a decision quickly, but it can lead to an overly large computational cost - the investor might have been able to make a decision with a smaller outline. This consideration might make us conclude that adding up the computation gradually in small C_t increments would be a better choice.

A different consideration applies to the presence of the competition. If the investor is not alone in the market, the opportunity will be available to other investors as well. A way to model this is that at any timestep, there is a p_{loss} probability that a competitor would grab the opportunity, leading the investor with the $Pr_{decline}$ value, with all the invested computation cost wasted. In a highly competitive environment, it is a bad strategy to use too small C_t increments.

Figure 2 shows an example of the decision making using the scheduling process. The evolution of the estimate V_{est} is shown in the black line, the low estimate V_{low} in green while the high estimate V_{high} in red. The allocated computation is shown in the lower graph. As a result of these computations, the σ value decreases, and thus the lower and higher estimates are getting closer together. As long as the decision line indicating zero profit is between the high and the low estimates, the investor cannot make a decision. In this graph, at timestep 12 the green lower estimate line crosses the decision line - this gives the investor sufficient confidence that there will be a profit, and at this point the decision is made to make an investment. After the investment had been made, no further computation is scheduled.

4. Scheduling algorithms

4.1. Non-adaptive scheduling algorithms

Before we move on to the proposed VoI-based scheduling algorithm, let us discuss several non-adaptive scheduling algo-

[2]Of course, this statement assumes that each investor has access to the same algorithms and information

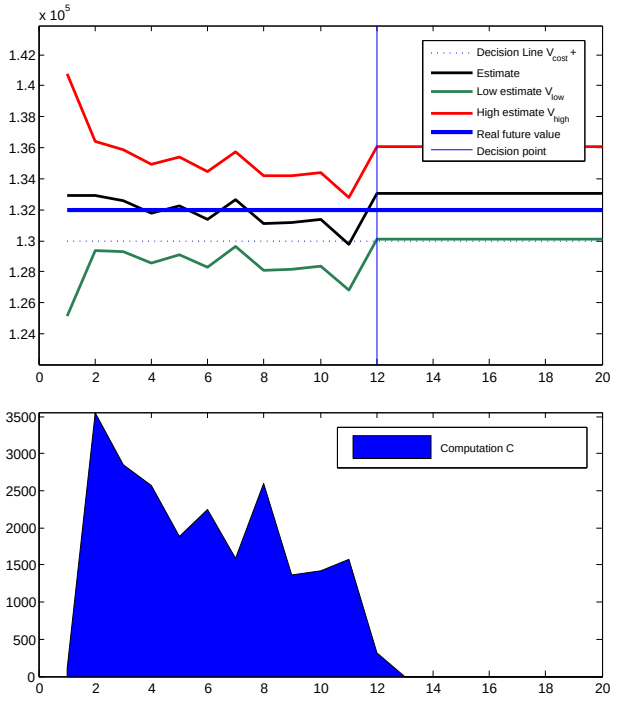


Figure 2: An investment analysis process ending in the decision to invest in the offered opportunity. Upper graph: the evolutions of the estimates in time. Lower graph: the allocated computation per time unit.

rithms the user might describe.

No-cost approach: this model assumes that the user does not invest in any computational power. Such a user can still perform the computation on his desktop or laptop machine, a situation we can approximate with zero cost (compared with the investments). Naturally, such a user is disadvantaged with regards to the speed of analyzing the investment opportunity. However, he has the advantage that computational expenses will not reduce his profit.

Data center: this model assumes that the user has a data center with a fixed amount of computational power. A typical example would be a Hewlett Packard modularized data centers POD 240a which can host about 4400 servers. In general, data centers are most efficient if they are fully utilized, thus we will make the assumption that the user will choose to divide the computational power of the data center evenly among the pending opportunities. The data center incurs a constant cost, even if there is no computation performed. Another potential problem is that the more pending opportunities are there, the less computing power is available.

Cloud computing (on demand, but not adaptive): this model assumes that the user uses on-demand cloud computing instances to analyze the investment opportunity. One of the advantages of this model is that when there are no pending opportunities, no cost will be incurred. On the other hand, the user can analyze an arbitrary number of investment opportunities, and analyzing multiple opportunities does not limit the amount of computation the user can spend on each. Lacking other information about the opportunities, the best choice of the user is to allocate the same amount of computing power to each opportunity.

4.2. The VoI scheduler

Let us assume that we have a cloud computing account and a certain opportunity with cost V_{cost}. How much computation $C(t)$ should we buy from the cloud provider at time step t? Note that in the remainder of this discussion we assume that our cloud computing environment is predictable [36]. The cloud computing approach we discussed above would be to decide on a fixed value C and allocate the same value $C(t) = C \ \forall t$ until the opportunity is either taken, declined, taken by a competitor or timed out. Fixed allocation is prone to overallocation (allocating too much to an opportunity which will not give as a significant profit) and underallocation (allocation too little computation to a promising opportunity). Of course, overallocation and underallocation can only be defined from the perspective of an omniscient observer who knows V_{future}. The scheduler itself does not know the profit because this is exactly what it tries to calculate.

Let us analyze the scheduler's decision from the point of view of the value of information. The concept of the *value of information* we refer in this context was initially introduced by Bölöni and Turgut [37, 22] in the intruder tracking sensor networks and later extended into path planning [38] and scheduling [25, 24, 23] algorithms in the underwater sensor networks domain.

If the decision to invest had been reached, it means that the estimate with the sufficient σ had obtained a profit Pr_{invest}, which can be seen as the value associated with the information contained in the estimate. Similarly, if the decision to not invest had been reached, the value of information is the loss that had been avoided by declining the opportunity. We can avoid the over and underallocation of the computing resources if we make the cost of the computation allocated to the analysis a constant fraction f of the value of information.

At the first timestep, the scheduling algorithm has no information about the profit, thus it can do no better than allocating a constant initial value $C(0) = C_{init}$. Starting from timestep $t = 2$ we have an estimate of the future profit and while this estimate is probabilistic through the current values $V_{est}(t)$ and $\sigma(t)$, it can be used to make decisions about the schedule. To avoid loosing investment opportunities when the initial profit estimates are close to zero, in the calculation of the allocated computational power we use a value that overestimates the profit with an optimism term o measured in units of σ:

$$C_{cost}(t+1) = f \cdot (C_{est}(t) + o \cdot \sigma(t) - C_{cost} - R_{safe}) \qquad (5)$$

Notice that in contrast with other approaches, this model does not allocate computation, but *money to buy computation with*. The actual amount of computation $C(t+1)$ allocated at time $t+1$ will be "whatever we can buy with $C_{cost}(t+1)$ money".

Finally, the VoI model introduces a new reason to decline an opportunity. We have seen that a schedule-independent reason for declining an opportunity is that our estimates tell us with a predefined degree of confidence that the investment will not be

Parameter	Value
Scenario	
arrival rate	Poisson distributed with $\lambda = 0.01$ to 2.00 (opportunities / hour)
timespan	1000 hours
V_{cost}	uniformly distributed \$100,000 .. \$500,000
V_{future} - the future value of the investment	normally distributed in a range around V_{cost}
R_{safe}	0
competition	0.001 to 0.3 (probability / hour)
cost of cloud computation	Amazon EC2 large computing instances costs of April 2016
Estimation algorithm	
k_1	$0.1 \cdot V_{future}$
k_2	$1.0 \cdot V_{future}$
k_3	2
confidence k	2
No cost scheduler	
Free computation per opportunity per time	1 ECU
Data center scheduler	
Data center capacity	10,000 ECU
Cloud scheduler	
Computation per opportunity per time	10,000 ECU
VoI scheduler	
inital profit estimate $Pr(0)$	$0.001 \cdot V_{cost}$
optimism o	$0.4 \cdot \sigma$
fraction f	0.01

Table 3: Simulation parameters

profitable. With the VoI model, another reason to decline would be that the VoI decides that it is not justified to invest more money in computation for this opportunity. If the formula gives $C_{cost}(t) = 0$ for a given t, all the future $C_{cost}(t+1)$ values will be also zero, because with no added computation, the values will not change.

5. Experimental study

5.1. Scenario

We implemented the application model of analyzing investment opportunities as described in Section 3. Table 5.1 describes the simulation parameters.

5.2. Calculating the cost of computation

To calculate the cost of the cloud computing approach we will use the current cost of computing units from Amazon EC2. For the cost of the equivalent computing power obtained in a local data center, we will rely on the fact that many analysts assume that the the profit margins of Amazon AWS are about 50%[39], which makes the data center costs about 50% from the Amazon cost.

Another question is measuring the computation unit. This can be tricky as different performance factors in the system (cache size, memory amount, memory speed, GPU, storage amount and speed) might make a difference in the performance of a specific application. However, our scenario assumes that all the schedulers use the same software, which makes suitable to use a common metric. We will use the ECU metric introduced by Amazon [3]. We assume that the no-cost scheduler allocates 1 ECU per task.

5.3. Experiment 1: Dependence on the arrival rate

In this study, we measure the performance of the various scheduling approaches function of the arrival rate of the opportunities λ between 0.01 and 2.0 while keeping the competition level at 0.1. The experimental results for various metrics are shown in Figure 3. Obviously, the most important metric here is the total profit shown in Figure 3(a), the only of genuine interest for the investor. We find that for all the scheduling algorithms except NoCost the profit increases with the number of opportunities, with the VoI algorithm being clearly the best for all arrival rates, followed by the DataCenter and the Cloud schedulers. At this level of competition, the NoCost creates only minimal profit regardless of the arrival rate.

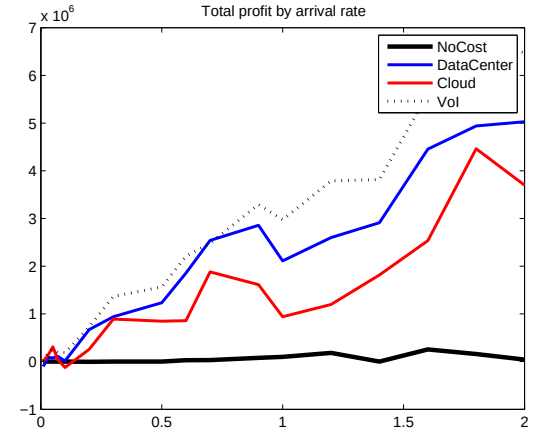
Figures 3(b), 3(c) and 3(d) help us understand the actual behavior of the user model. Figure 3(b) shows us the number of taken opportunities function of the arrival rate. We find that NoCost very rarely chooses to take an opportunity. What is interesting is that the VoI and the Cloud algorithms choose to take effectively the same number of opportunities. Despite this, the VoI created a larger profit due to two factors: it picked a higher fraction of the more valuable opportunities and spent much less on the less valuable ones. The DataCenter scheduler took a much smaller amount of opportunities, especially at higher arrival rates, where its computational power had to be divided between the various opportunities that need to be analyzed. Note, however, that, this lower acceptance rate yielded a higher profit, due to the overall lower cost of the computations.

Finally, Figure 3(c) and Figure 3(d) show the number of opportunities lost to the competion and their ratio to the total number of opportunities that had been presented. What we observe here is that, in general, the NoCost computing model looses the most opportunities, because its low computational investment, it takes a long time to reach a decision, during which time competitors have the chance to grab the opportunity. What is interesting here is that wastly lower ratio of lost opportunities presented by the VoI scheduler – this is due to the adaptability of the scheduler's allocation. If it sees a promising opportunity, the VoI scheduler will ramp up the computation much faster than the uniformly allocating Cloud scheduler; on the other hand, the VoI scheduler is more ready to decline an unpromising opportunity - both effects leading to a lower rate of opportunities lost to the competition.

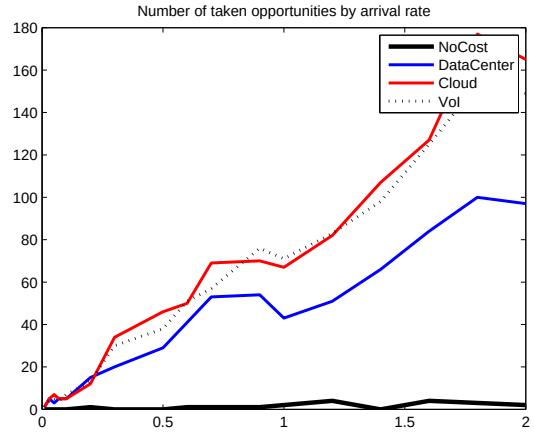
This experiment had shown us that the presence of competition significantly impacts the profits, leading us to the next experiment.

5.4. Experiment 2: Dependence on the competition

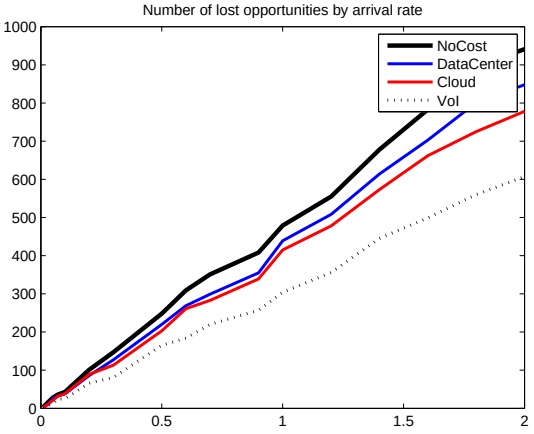
In this study, we vary the competition value p_{loss} which shows the chance for a given profitable pending opportunity to be taken by a competitor at any given time slot in a range of 0.001 to 0.3. For these experiments we kept the arrival rate λ constant at 1.0.



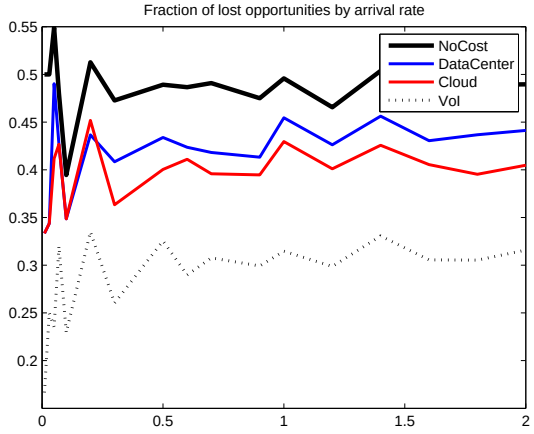
(a) Total profit function of arrival rate



(b) Opportunities taken function of arrival rate



(c) Opportunities lost function of arrival rate



(d) Fraction of opportunities lost function of arrival rate

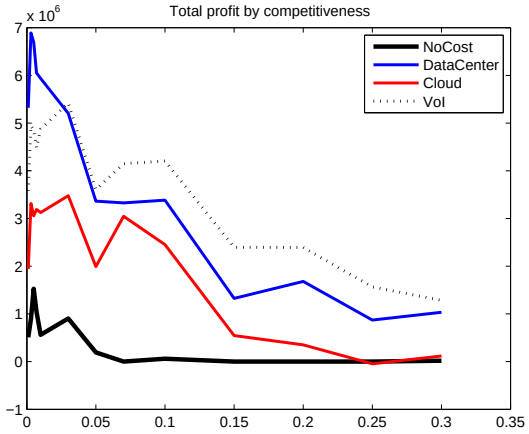
Figure 3: Performance of various scheduling algorithms function of the arrival rate of investment opportunities

Figure 4(a) shows the total profit. As expected, what we see here is that the higher the competition, the lower the profit for all scheduling algorithms. For very low competition even the NoCost algorithm can make profit - the decision will eventually be made, and no computational cost diminishes the cost. Also for a low competition, DataCenter approach can actually outperform VoI, primarily due to its cheaper computational power. Starting from a competition level of 0.05, however, the VoI approach significantly outperforms the other approaches. What actually see is that for very high competition, the Cloud approach can actually loose money! As Figure 4(b) shows, this is not due to its underperforming in its computations - in fact, it takes more opportunities than the VoI approach. The Cloud approach, however, looses a lot of money by analyzing opportunities which are then lost to the competition. Both the DataCenter approach with its fixed costs that do not depend on the number of opportunities and the VoI approach with its adaptive behavior perform better for high competition.

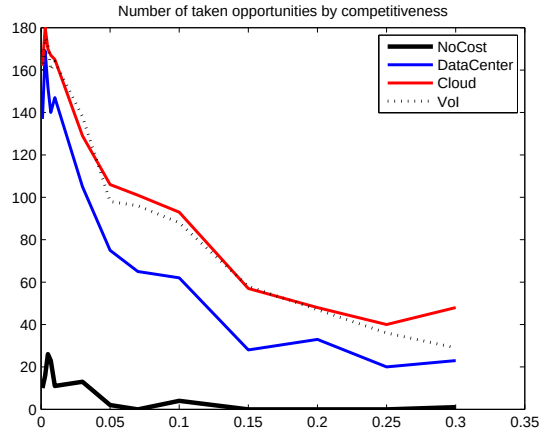
Finally, figures 4(c) and Figure 4(d) show a similar pattern to the ones in Experiment 1, with the VoI approach loosing the least opportunities to competition. An interesting pattern can be observed on Figure 4(d) while the other approaches stabilize at a high level of about 50% loss starting from a relatively mild competition of 0.05, the VoI approach stays lower, but it also increases towards this level. The reason of this increase is the special treatment of the first step in the VoI approach. For the timestep, the VoI approach has no estimate of the VoI, thus only makes a comparatively small initial investment. If the competition is high, there can be a significant chance that the system looses the investment in timesteps 1 or 2 before the VoI behavior "kicks in".

6. Conclusions

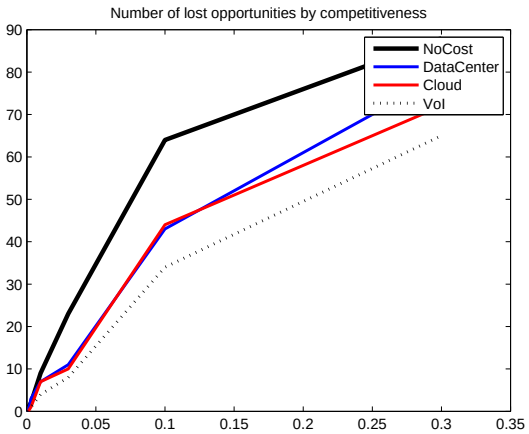
The amount of high performance computing performed at a research institution or business used to be limited by the available computational facilities. The decision to invest in such facilities were justified and made years in advance. The advent of cloud computing made the decision to perform high performance computation on thousands of computer cores a decision



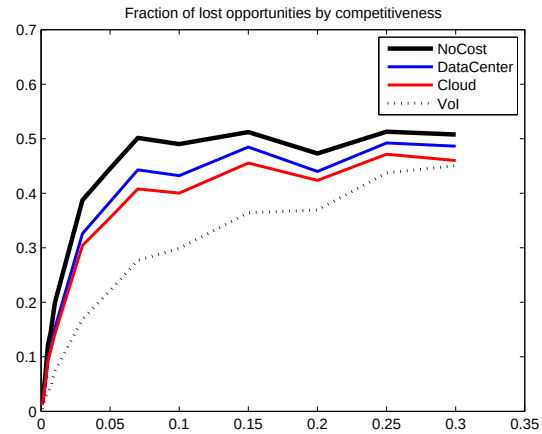
(a) Total profit function of the competition metric



(b) Opportunities taken function of the competition metric



(c) Opportunities lost function of the competition metric



(d) Fraction of opportunities lost function of the competition metric

Figure 4: Performance of various scheduling algorithms function of the intensity of the competition

that can be taken on a minute's notice. In this paper, we argued that the ability to make this decision very quickly does not reduce the need to analyze whether the computational expenses are justified or not. We discussed that many modern high performance computing applications are elastic in term of computational power - additional computation improves the quality of results, but often with a curve of diminishing returns. We argue that a convenient technique to create efficient decision making approaches is to use the concept of "value of information" - to try to quantify the amount of financial benefit a certain calculation can gain us, and use this value when making scheduling decisions.

We illustrate the proposed model with the example of an investor who is analyzing real estate investment opportunities. We compare approaches that assume minimal, no-cost computational analysis with the cost of maintaining a private data center, buying computational power on the cloud and, finally, with a VoI-informed, cloud-based scheduling approach. We find that the VoI approach clearly outperforms every other approach across a wide range of opportunity arrival rates and competition intensity values.

References

[1] F. Berman, G. Fox, A. J. Hey, Grid computing: making the global infrastructure a reality, Vol. 2, John Wiley and Sons, 2003.

[2] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International journal of high performance computing applications 15 (3) (2001) 200–222.

[3] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, A performance analysis of EC2 cloud computing services for scientific computing, in: Cloud computing, Springer, 2009, pp. 115–131.

[4] S. Garfinkel, An evaluation of Amazon's grid computing services: EC2, S3, and SQS, Tech. rep., Harvard University (2007).

[5] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, J. Wilkes, Large-scale cluster management at Google with Borg, in: Proc. of the European Conference on Computer Systems (EuroSys), 2015, pp. 18:1–18:17.

[6] X. Wen, G. Gu, Q. Li, Y. Gao, X. Zhang, Comparison of open-source cloud management platforms: OpenStack and OpenNebula, in: Proc. of IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'12), 2012, pp. 2457–2461.

[7] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, et al., A comparison of eleven static heuristics for mapping a class of independent

tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed computing 61 (6) (2001) 810–837.

[8] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, et al., Apache Hadoop YARN: yet another resource negotiator, in: Proc. of the 4th annual Symposium on Cloud Computing, ACM, 2013, p. 5.

[9] Z. Zheng, R. Wang, H. Zhong, X. Zhang, An approach for cloud resource scheduling based on parallel genetic algorithm, in: Proc. of IEEE International Conference on Computer Research and Development (ICCRD'11), Vol. 2, 2011, pp. 444–447.

[10] W. Li, J. Tordsson, E. Elmroth, Modeling for dynamic cloud scheduling via migration of virtual machines, in: Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom'11), 2011, pp. 163–171.

[11] S. Pandey, L. Wu, S. M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: Proc. of IEEE International Conference on Advanced Information Networking and Applications (AINA'10), 2010, pp. 400–407.

[12] S. Abrishami, M. Naghibzadeh, D. H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Generation Computer Systems 29 (1) (2013) 158–169.

[13] E. Caron, F. Desprez, A. Muresan, Forecasting for grid and cloud computing on-demand resources based on pattern matching, in: Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10), 2010, pp. 456–463.

[14] L. Rodero-Merino, E. Caron, A. Muresan, F. Desprez, Using clouds to scale grid resources: An economic model, Future Generation Computer Systems 28 (4) (2012) 633–646.

[15] X. Bai, D. C. Marinescu, L. Bölöni, H. J. Siegel, R. A. Daley, I.-J. Wang, A macroeconomic model for resource allocation in large-scale distributed systems, Journal of Parallel and Distributed Computing 68 (2) (2008) 182–199.

[16] M. E. Frîncu, Scheduling highly available applications on cloud environments, Future Generation Computer Systems 32 (2014) 138–153.

[17] L. F. Bittencourt, E. R. M. Madeira, HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds, Journal of Internet Services and Applications 2 (3) (2011) 207–227.

[18] W. Wang, B. Li, B. Liang, Towards optimal capacity segmentation with hybrid cloud pricing, in: Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'12), 2012, pp. 425–434.

[19] R.-H. Hwang, C.-N. Lee, Y.-R. Chen, D.-J. Zhang-Jian, Cost optimization of elasticity cloud resource subscription policy, IEEE Transactions on Services Computing 7 (4) (2014) 561–574.

[20] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, J. Chen, A cost-aware auto-scaling approach using the workload prediction in service clouds, Information Systems Frontiers 16 (1) (2014) 7–18.

[21] R. A. Howard, Information value theory, IEEE Transactions on Systems Science and Cybernetics 2 (1) (1966) 22–26.

[22] D. Turgut, L. Bölöni, IVE: improving the value of information in energy-constrained intruder tracking sensor networks, in: Proc. of IEEE ICC'13, 2013, pp. 6360–6364.

[23] F. Khan, S. Khan, D. Turgut, L. Bölöni, Scheduling multiple mobile sinks in underwater sensor networks, in: Proc. of IEEE LCN'15, 2015, pp. 358–365.

[24] S. Basagni, L. Bölöni, P. Gjanci, C. Petrioli, C. Phillips, D. Turgut, Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle, in: Proc. of IEEE INFOCOM'14, 2014, pp. 988–996.

[25] L. Bölöni, D. Turgut, S. Basagni, C. Petrioli, Scheduling data transmissions of underwater sensor nodes for maximizing value of information, in: Proc. of IEEE Globecom, 2013, pp. 460–465.

[26] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: Proc. of IEEE INFOCOM'12, 2012, pp. 963–971.

[27] T. Cucinotta, D. Lugones, D. Cherubini, K. Oberle, Brokering SLAs for end-to-end QoS in cloud computing., in: Proc. of IEEE CLOSER'14, 2014, pp. 610–615.

[28] K. Konstanteli, T. Cucinotta, K. Psychas, T. A. Varvarigou, Elastic admission control for federated cloud services, IEEE Transactions on Cloud Computing 2 (3) (2014) 348–361.

[29] H. R. Arkes, C. Blumer, The psychology of sunk cost, Organizational behavior and human decision processes 35 (1) (1985) 124–140.

[30] P. P. Boyle, Options: A Monte Carlo approach, Journal of financial economics 4 (3) (1977) 323–338.

[31] D. Gimpelevich, Simulation-based excess return model for real estate development: A practical Monte Carlo simulation-based method for quantitative risk management and project valuation for real estate development projects illustrated with a high-rise office development case study, Journal of Property Investment & Finance 29 (2) (2011) 115–144.

[32] O. Hosny, K. Nassar, P. A. Olusola, Decision support system for housing developers in developing countries under uncertain buyer behavior, Journal of Management in Engineering 28 (3) (2012) 311–323.

[33] M. Holtan, Using simulation to calculate the NPV of a project, `http://www.investmentscience.com/Content/howtoArticles/simulation.pdf`, accessed: 2016-09-20.

[34] F. Black, M. Scholes, The pricing of options and corporate liabilities, The journal of political economy (1973) 637–654.

[35] C. P. Robert, G. Casella, Monte Carlo integration, in: Monte Carlo Statistical Methods, Springer, 1999, pp. 71–138.

[36] M. García-Valls, T. Cucinotta, C. Lu, Challenges in real-time virtualization and predictable cloud computing, Journal of Systems Architecture 60 (9) (2014) 726–740.

[37] D. Turgut, L. Bölöni, A pragmatic value-of-information approach for intruder tracking sensor networks, in: Proc. of IEEE ICC'12, 2012, pp. 4931–4936.

[38] F. Khan, S. Khan, D. Turgut, L. Bölöni, Greedy path planning for maximizing value of information in underwater sensor networks, in: Proc. of IEEE P2MNet'14, 2014, pp. 39–44.

[39] S. O'Grady, AWS: Forget the revenue, did you see the margins?, `http://redmonk.com/sogrady/2010/08/04/aws-margins/`, accessed: 2016-09-20.