Business-Driven Management of Hybrid IT Infrastructures

Paulo Ditarso Maciel Jr.[a], Marcus Carvalho[a], Nazareno Andrade[a], Francisco Brasileiro[a], Ricardo Araújo[a], David Maia[a], Renato Miceli[a], Raquel Lopes[a], Miranda Mowbray[b]

[a] *Universidade Federal de Campina Grande*
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bloco CO
58.109-970, Campina Grande - PB, Brasil

[b] *Automated Infrastructure Lab*
Hewlett-Packard Laboratories Bristol
Long Down Avenue, Stoke Gifford
Bristol BS34 8QZ, UK

Abstract

With the emergence of the cloud computing paradigm and the continuous search to reduce the cost of running Information Technology (IT) infrastructures, we are currently experiencing an important change in the way these infrastructures are assembled, configured and managed. In this research we consider the problem of managing a hybrid high-performance computing infrastructure whose processing elements are comprised of in-house dedicated machines, virtual machines acquired from cloud computing providers, and remote virtual machines made available by a best-effort peer-to-peer (P2P) grid. Each of these resources has a different cost basis. The applications that run in this hybrid infrastructure are characterised by a utility function: the utility yielded by the completion of an application depends on the time

Email addresses: pmaciel@lsd.ufcg.edu.br (Paulo Ditarso Maciel Jr.), marcus@lsd.ufcg.edu.br (Marcus Carvalho), nazareno@dsc.ufcg.edu.br (Nazareno Andrade), fubica@dsc.ufcg.edu.br (Francisco Brasileiro), ricardo@lsd.ufcg.edu.br (Ricardo Araújo), davidcmm@lsd.ufcg.edu.br (David Maia), renato@lsd.ufcg.edu.br (Renato Miceli), raquel@dsc.ufcg.edu.br (Raquel Lopes), miranda.mowbray@hp.com (Miranda Mowbray)

Preprint submitted to Future Generation Computer Systems *August 25, 2009*

taken to execute it. We take a business-driven approach to managing this infrastructure, aiming to maximise profit, that is, the utility produced as a result of the applications that are run minus the cost of the computing resources that are used to run them. We assume that the cost of computing resources from the local in-house machines is unavoidable, i.e.. the in-house infrastructure has a fixed cost whether or not its resources are used. We also assume that the cost of computing resources from the P2P grid (when they are available) is negligible, because the grid is based on the exchange of spare resources between peers. Applications are run using computing power just from these two sources whenever possible. Any extra capacity required to improve the profitability of the infrastructure is purchased from the cloud computing market. We assume that this extra capacity is reserved for future use through short term contracts which are negotiated without human intervention. The cost per unit of computing resource may vary significantly between contracts, with more urgent contracts normally being more expensive. However, due to the uncertainty inherent in the best-effort grid, it may not be possible to know in advance exactly how much computing resource will be needed from the cloud computing market. Overestimation of the amount of resources required leads to the reservation of more than is necessary; while underestimation leads to the necessity of negotiating additional contracts later on to acquire the remaining required capacity. We propose heuristics to be used by a contract planning agent in order to balance the cost of running the applications and the utility that is achieved with their execution, with the aim of producing a high overall profit. We demonstrate that the ability to estimate the grid behaviour is an important condition for making contracts that produce high efficiency in the use of the hybrid infrastructure. We propose a model for predicting the behaviour of a P2P grid that uses a particular incentive mechanism, and assess the suitability of this model using field data. Our results show that the proposed model is able to predict the grid behaviour with an average error that is not larger than 16% for the scenarios evaluated, leading to a worst case efficiency of 85.32%.

Key words:
cloud computing, grid computing, peer-to-peer, business-driven IT management, capacity planning.

1. Introduction

A new business model is currently being adopted which is changing the way that Information Technology (IT) resources and services are deployed and used. In this model the acquisition of resources and services occurs whenever and wherever needed, and the amount charged is related to the amount of resources and services that are actually used. This model of IT sold as a service has been called *cloud computing*. One of its main selling points is the possibility of substantial reductions on the total cost of ownership of IT infrastructures.

Economic advantages certainly play an important role in the adoption of this business model, but there are other important factors to consider which are likely to result in organisations preserving at least some of their in-house infrastructure, rather than having all their applications run on computers owned by cloud service providers. For example, the retention of some in-house capacity may serve to cushion the effects of price fluctuations given by transient instabilities in the cloud computing market. The migration from services supported by in-house dedicated IT infrastructures to services offered by an external cloud computing provider is likely to face strong resistance from the internal IT management staff. For certain types of application the cost reduction resulting from a move to external providers may be low. Most importantly, organisations may decide that they do not wish to execute certain types of application on a computing infrastructure that they do not own or manage: these might include for example business-critical applications requiring very high availability, or applications that process sensitive data.

The market-based cloud computing model is not the only way to reduce total cost of ownership. Among other solutions proposed, peer-to-peer (P2P) grid computing has been suggested as a way to enable a simpler economy for the trading of computing cycles that would otherwise be idle [1]. Markets rely on the existence and efficiency of contract negotiation, norm enforcement, banking and accounting mechanisms. For several scenarios in distributed computing (and also outside computing), implementing such mechanisms is complex, costly or inefficient. In contrast, sharing systems may be efficient for these scenarios, as they can use information which is loosely structured and therefore easier to obtain, they can make use of social relations for monitoring and enforcement, and they have lower marginal transaction costs [2]. These solutions generally give no guarantees on the quality of service provided – indeed, they do not guarantee that the service will be provided at all. Nevertheless, they have been successfully used to increase the cost effectiveness of IT infrastructures in a number of settings [3].

We expect that in the near future many IT infrastructures will use both resources provided by in-house dedicated infrastructure and resources from external cloud computing providers. Moreover, spare capacity of the in-house infrastructure may be used to execute workload on behalf of other organisations, in exchange for the possibility of using these organisations' spare capacity to run part of the local workload in the future. The different components of this hybrid infrastructure will provide different guarantees, ranging from potentially very detailed quality of service guarantees, to no guarantees at all for best-effort components.

This expectation is supported by our own experience with the OurGrid middleware (http://www.ourgrid.org/). This middleware allows the deployment of open P2P grids. We have used the OurGrid middleware to foster the creation of the OurGrid Community, which has been used as a computing platform in a variety of application areas, including engineering, bioinformatics, computer science and financial applications [4, 5, 6, 7] (for a current snapshot of the running system, see http://status.ourgrid.org/). In particular, OurGrid supports the cooperative work of a community of meteorologists and hydrologists, both in academia and in government [4, 8]. Some members of this community provide daily weather forecasts as a public service (see, for instance, http://www.cptec.inpe.br/). The capacity required at critical times (when time-constrained applications are run) is normally much larger than that required at other times. Provisioning the IT infrastructure of these public agencies to cope with the high demand at critical times is not cost-effective. An agency will in general be able to obtain some additional capacity at a negligible cost from OurGrid at such a time; however, since OurGrid is best-effort, the agency cannot rely on OurGrid to always provide the capacity required. In this setting, it is likely that extra computing power that is not obtainable from OurGrid could be purchased on demand from cloud service providers at a smaller cost than the cost of provisioning the agency's in-house dedicated infrastructure to meet such demands. The presence of OurGrid reduces the amount of computing power which an agency will need to purchase at critical times. So, a hybrid IT infrastructure consisting of all three potential sources of computing resources (in-house, an external cloud computing provider, and a P2P grid) is highly desirable for this scenario.

Since most studies of IT management have assumed that IT services are provided by just one of these sources, interesting research questions arise from consideration of the hybrid IT environment just described. In contrast to existing work that focuses on the management of the cloud computing provider's infrastructure (e.g.. [9]), in this paper we take the point of view of the customer of a cloud computing service. More precisely, we discuss how this customer (the manager of the hybrid infrastructure) can make the best use of the dedicated in-house capacity, while judiciously using the other two sources of computing resources or services. We concentrate on the *contract planning* aspect of the IT infrastructure management [10], i.e., *given a utility function for an application and its predicted workload, how should one plan the contracts that will be made with cloud computing providers, in order to balance the cost of executing the application and the utility yielded by its execution?*

We study heuristics used by a contract planner agent which follows a business-driven approach to managing this infrastructure, meaning that it aims to maximise profit, where the profit in this case is the utility produced as a result of the applications that are run minus the cost of the computing resources that are used to run them. We assume that the agent does not have any say in which applications are to be run on the hybrid infrastructure, but can choose how much cloud computing capacity to reserve and when to make the reservations. In particular, we extend the work by Maciel Jr. et al. [11] where the hybrid infrastructure described above was first presented, and which highlighted the importance of an accurate prediction of the behaviour of the P2P grid. Here we extend the model to a larger spectrum of applications, and look more closely at the impact that an error in the prediction of the behaviour of the grid has on the overall efficiency of the system. In this direction, we propose a model for predicting the quality of service offered by a P2P grid that uses the incentive mechanism proposed by Andrade et al. [12], and we evaluate the model using data from the Grid Workload Archive [13].

For the applications that we had in mind when writing this paper, the computing resource used from the IT infrastructure is simple processing power. However, our analysis can also be applied to applications which use a different type or resource or service. For the rest of this paper we will write "cycle" as shorthand for a unit of computing services or resources: this could for example be an amount of computing power equivalent to a fixed quantity of CPU cycles on a reference machine when the application considered uses processing power from the infrastructure, or 1KB of storage when the application uses storage from the infrastructure, or one unit of a particular higher-level service. We will assume however that the application only requires one type of service or resource, and is sufficiently parallel that it can use any two cycles from any two sources interchangeably and in parallel. (These assumptions are true for most of the applications that are run using OurGrid.)

The remainder of the paper is organised as follows. In Section 2 we survey related work. Section 3 gives formal definitions of the application that we will consider and of the hybrid IT infrastructure. A formal description of the problem we target is given in Section 4. We present heuristics for contract planning in Section 5, and evaluate them in Section 6. In Section 7 we propose and validate the model for predicting the quality of service of the P2P grid. Finally, in Section 8 we give concluding remarks and discuss areas in which further research needs to be developed.

2. Related Work

In this paper, the applications that we are interested in are ones that are time-constrained. There are many examples of distributed applications in which customers need guarantees on the response time and on the allocation of resources. Examples of the domain areas of these applications include: remote medicine [14]; real-time control of sensitive sites, instruments and air traffic flow [15, 16]; multimedia and stream processing [17, 18]; environmental forecast [19, 4]; and e-Science experiments [20, 21]. There have been some efforts to develop real-time support in distributed applications [22, 23], and to improve quality of service guarantees of high-performance distributed computing systems by using *advance reservations* [24, 25, 26]. However, a better understanding of the quality of service requirements of these real-time applications is desirable. As yet there is not a complete understanding of how to generate functions that express the utility that customers gain from their applications given the length of the time that the application takes to complete. However, some work on determining these utility functions has been done under the assumptions that the utility functions are step functions or have a linear decay over time [27, 28, 9, 29]. In our work, we consider three time-constrained applications whose utilities to the customer are a step function, a linearly decaying function, and an exponential decaying function.

There are some similarities between our work and that of Popovici and Wilkes [9] and Yu et al. [28], as well as some points in which our work complements these earlier papers. Popovici and Wilkes focus on the operations of a service provider, while Yu et al. look at the interaction between a customer and a grid service provider. Our work, on the other hand, investigates the interaction of a customer with both a best-effort grid and a cloud computing provider.

Popovici and Wilkes [9] propose an economics-oriented approach for a service provider, to solve the question of which customers' requests the service provider should accept. They extend works by Chun and Culler [30] and Irwin et al. [29], by considering a service provider that offers job-based services to its clients and rents resources from a resource provider. The difficulty in selecting the requests is that they assume that the service provider will have some uncertainty about the availability of the resources necessary to fulfil the requests. Popovici and Wilkes define risk-aware heuristics for admission control and scheduling that aim at maximising the service provider's utility, taking into account the uncertainty in resource availability. In our work we consider a contract planner for a hybrid IT infrastructure, and the uncertainty comes from the best-effort nature of the P2P grid.

Yu et al. [28] propose a scheduling algorithm that minimises the execution costs of workflows in a grid while meeting users' specified deadlines. The unreliability of the grid introduces uncertainty. This, in turn, is dealt with by re-scheduling tasks that fail, moving them to other available computers in the grid. Predictions of availability and information about the costs of grid resources under available contracts are used to select the most appropriate contracts and to minimise cost. Our work uses a similar strategy, but applies it to a different context.

Cloud computing providers require solid business models. Rappa [31] presents a general overview of what the business model for cloud computing might look like, taking account characteristics such as necessity, reliability, usability, and scalability. This model is similar to business models for the provision of utilities such as water, telephone, Internet access, and electricity. The pricing model for cloud computing resources that we use in this paper is based on pricing models used for other utilities, in which there is a charge for the reservation of resources.

Buyya et al. provide some economic models for setting the prices of services based on supply and demand. These include commodity markets, posted prices, and auctions [32]. Buyya et al. describe a system architecture and policies for resource management in grid infrastructures, based on the various possible pricing models. In this paper we assume a generic pricing model in which the cost of a cycle from a cloud computing provider is composed of a reservation fee and a consumption fee. Different providers' profiles can be mapped onto different reservation and consumption fees. Details of the pricing model are given in Subsection 3.2.

This paper is based on a previous work by Maciel Jr. et al. where the hybrid infrastructure was presented for the first time [11]. More specifically, Maciel Jr. et al. defined the hybrid IT infrastructure and its elements, proposed contract planning heuristics in order to decide which contracts to establish with the cloud computing provider, and evaluated the efficiency of these heuristics for a limited class of applications. Their main conclusion was that using information about the cloud computing provider's pricing model and accurately estimating the amount of cycles that will be reclaimed from the best-effort P2P grid are crucial to establishing contracts that produce a high profit from running applications on the hybrid infrastructure. In this paper we extend the work by Maciel Jr. et al. by: i) considering different utility functions for the applications (Section 3); ii) refining the function that represents the total profit yielded by running an application in the hybrid infrastructure (Section 4); further investigating the impact on the efficiency of the infrastructure of inaccurate predictions of the number of cycles that are reclaimed from the P2P grid over a given time interval (Section 6); and, iv) proposing and assessing the usefulness of a model for making this prediction for a P2P grid that uses the incentive mechanism proposed by Andrade et al. [12].

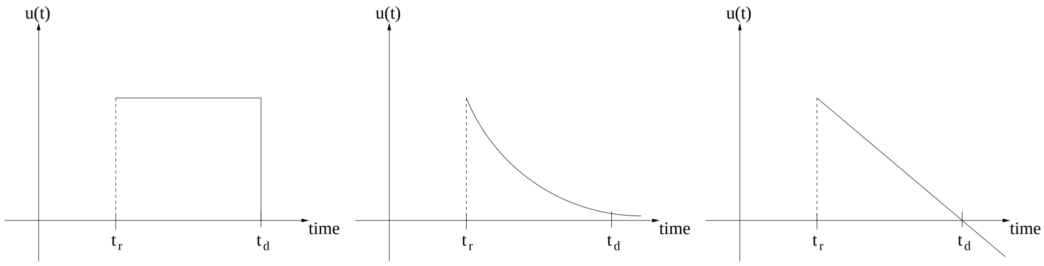
3. System Model

In this section we present the system model. First we describe the application that we will consider, and give a model for the utility yielded by its execution. Then we present the different components of the hybrid IT infrastructure, with a focus on modelling the costs of the cycles from the three different components.

3.1. The Application

For the sake of simplicity, we assume that during a time period Δ (typically on the order of a day), there is a single critical application to be executed. We characterise this application by a tuple $\mathcal{A} = \langle w, t_r, u(t) \rangle$, where w is an indication of the application's demand on the infrastructure, expressed as the number of the cycles it requires for completion; t_r is the instant of time, within Δ, when the application is ready for execution (for example the instant of time when data required to run the application becomes available); and $u(t)$ is the application's utility function. The function $u(t)$ specifies the total utility obtained by the owner of the hybrid IT infrastructure from the execution of the application, as a function of the time that the execution completes. Clearly $u(t)$ is only defined for $t \geq t_r$.

Figure 1 shows some examples of utility functions for time-constrained applications. For such applications the utility is a decreasing function of the instant of time at which the application completes its execution. Function $step(t)$, for instance, represents a real time application whose utility is positive and constant as long as its execution completes before a deadline; for completion times longer than that, the utility drops to zero. Function $expo(t)$ is exponentially decreasing, and tends to zero as the completion time increases. Finally, function $decay(t)$ starts positive and decreases linearly as the completion time increases, becoming negative after the completion time passes a deadline. The negative utility may represent a penalty incurred if the application cannot be completed by an agreed deadline.



(a) Step function (b) Exponential function (c) Decay function

Figure 1: Some examples of utility functions

In this paper we consider applications with the following common pattern: there is no utility gained if the application completes its execution after a predefined deadline (t_d), i.e. $u(t) = 0, \forall t, t > t_d$. We consider three different applications. The first application, called *All-or-Nothing* (AoN), has as its sole constraint to complete its execution before the deadline. For the second application, called *Linear-Decay-until-a-Deadline* (LDuD), the utility decays linearly until it reaches the value zero at t_d. Finally, we consider an intermediate application, called *Graceful-Degradation-until-a-Deadline* (GDuD), whose utility decays exponentially with time. The first two applications are modelled by the following utility function:

$$u(t) = \begin{cases} a \cdot (t - t_r) + b, & \text{if } t_r \leq t \leq t_d; \\ 0, & \text{otherwise,} \end{cases}$$

with $b > 0$ and $a = 0$ for the *AoN* application and $a = -b/(t_d - t_r)$ for the *LDuD* application. The third application is modelled by the following utility function:

$$u(t) = \begin{cases} a^t + b, & \text{if } t_r \leq t \leq t_d; \\ 0, & \text{otherwise,} \end{cases}$$

with $b > 0$ and $a < -1$.

3.2. The Hybrid IT Infrastructure

We consider a hybrid IT infrastructure which, during the period of time Δ, is able to provide $\mathcal{C}(\Delta)$ cycles, where

$$\mathcal{C}(\Delta) = \int_{\Delta} (i(t) + g(t) + p(t)) \cdot dt,$$

and $i(t)$, $g(t)$ and $p(t)$ are, respectively, the number of cycles available at time t from the in-house dedicated machines, the P2P grid and the cloud computing provider.

We will assume that all the cycles obtained from the cloud computing market are obtained from a single provider, but our analysis can be generalised for the case of several providers in a straightforward way.

The total cost of running the hybrid infrastructure during Δ is the sum of the cost of maintaining the in-house infrastructure, the cost of donating and receiving cycles to/from the best-effort P2P grid, and the costs arising from the use of cycles from the cloud computing provider's infrastructure. We assume that the cost of maintaining the in-house infrastructure is described by a fixed cost v_i for each cycle available from the in-house infrastructure, whether or not it is used. The cost of maintaining the dedicated infrastructure during Δ is therefore

$$\gamma_i(\Delta) = v_i \cdot \int_{\Delta} i(t)dt \tag{1}$$

We assume that the P2P grid works in an opportunistic way, with the peers only donating spare capacity to the grid which would otherwise be unused. Naturally, there is a cost v_g of donating a cycle to the grid; this is due mainly to the extra consumption of energy and to any extra security enforcements that are required before tasks can be executed on behalf of other peers using the spare capacity of the donating peer's local infrastructure. Nevertheless, these costs are likely to be very small compared to the other costs of operating and maintaining the local infrastructure (e.g. hardware, software, hosting, system administration, etc.). We therefore assume that $v_g \ll v_i$, and in the rest of the paper we ignore the cost of donating idle cycles to the grid. Moreover, there is no cost incurred for the cycles that are claimed from the grid, since they are donated cycles.

The costs of the cycles obtained from the cloud computing provider depend on the cloud computing provider's pricing model. At the time of writing, very few cloud computing providers offer a fully automated negotiation procedure. Most use a simple pricing model in which virtual machines are brought up and torn down at the wish of the customer, who pays a fixed price for each hour or fraction of hour effectively used. No reservation is needed, however there is only a limited number of virtual machines that can be simultaneously instantiated by a single customer. If a customer requires a larger number of virtual machines, she must have previously engaged in a human-to-human negotiation with the provider [33].

However, some cloud computing providers, such as Amazon, are starting to provide a pricing model that incorporates the notion of long-term reservation of resources prior to their use. In this case, a lower fee is charged for the use of resources if the customer has previously reserved them [34]. The cloud computing market is evolving rapidly, and we believe that it cannot be assumed that the pricing models currently used will be standard in the future. In fact, Amazon's introduction of reservation is an indication that, as more and more customers enter the market, providers will look for more information about customers' likely workloads in order to be able to operate their infrastructure efficiently. We predict that reservation will eventually become a common feature of cloud service pricing models, and that customers will be interested in software that provides some automation of the reservation and contract negotiation processes. We also believe that competition among providers will drive the providers to give customers more flexibility in their choice of contracts.

Based on these predictions, we define a pricing model that we believe is most likely to be adopted in the scenario that we consider in this paper, in which the customer's contract planning agent sets up short-term contracts. The pricing model involves two types of fees: a reservation fee and a consumption fee. It is reasonable to think that the urgency of a contract is a critical factor in setting the reservation fee: in general, contracts which are established only shortly before the computer power is due to be consumed will tend to have more expensive reservation fees per cycle than contracts which are established well in advance. For a fixed reservation and consumption time, we assume that the reservation cost is directly proportional to the number of cycles that are reserved. The maximum number of cycles that can be consumed at time t is the number that has been reserved for time t at times up to and including t, however it may be that not all of these will be consumed. The reservation fee is charged for all the cycles that are reserved, whether or not they are consumed, but the consumption fee is only charged for cycles that are consumed. We assume that the consumption fee associated with a contract is directly proportional to the number of cycles consumed under the contract. Note that this pricing model is similar to the one that Amazon has started to use; however, contracts are for much shorter periods than Amazon's contracts and are negotiated automatically.

We represent a contract between the client and the cloud computing provider by a tuple $\mathcal{K} = \langle t_e^{\mathcal{K}}, t_u^{\mathcal{K}}, c_r^{\mathcal{K}}, \beta^{\mathcal{K}} \rangle$, where $t_e^{\mathcal{K}}$ is the time at which the contract is established; $t_u^{\mathcal{K}}$ is the time at which the cycles are to be used; $c_r^{\mathcal{K}}$ is the number of cycles reserved; and $\beta^{\mathcal{K}}$ is a number between 0 and 1 expressing a relationship between the reservation and consumption fees per cycle, which is used to calculate the cost of cycles in a way detailed below. Since it does not make sense for the customer to reserve cycles for use outside the time interval $[t_r, t_d]$, we will assume that $t_u^{\mathcal{K}}$ lies in this time interval for all established contracts $\mathcal{K}$. When a contract is established, the cloud computing provider agrees to fulfil it and the customer (via the customer's agent) agrees to pay the agreed price. We assume that all contracts in compliance with the interests of provider and customer are successfully established and duly honoured by both parties involved.

We use the following model for the cost of c cycles under contract $\mathcal{K}$, where $0 \leq c \leq c_r^{\mathcal{K}}$. We write v_p for the total cost to the consumer (comprising both a reservation fee and a consumption fee) of reserving a single cycle at time t_p and consuming it at time t_d, where t_p is the earliest time at which a contract can be made for cycles to run the application, and t_d is the deadline for the execution of the application. We assume that this cost is independent of the choice of contract under which this reservation is made. (Obviously this contract has to be $\langle t_p, t_d, 1, \beta \rangle$ for some β: what we mean by the previous sentence is that the cost v_p is independent of the value of this β.) We we will use a function φ of the contract establishment time and the time of consumption which is independent of the choice of contract, which reflects how the reservation costs vary with these times. We assume that φ is a strictly decreasing function of the difference between the consumption time and the contract establishment time, so that it is larger for more urgent contracts, and that it takes value 1 when this difference is as large as possible, i.e., $\varphi(t_p, t_d) = 1$. The cost profiles of different providers can be modelled by different choices of $\beta^{\mathcal{K}}$, v_p and φ.

Given this, the cost of using c cycles under a contract $\mathcal{K}$ is given by:

$$\gamma_p^{\mathcal{K}}(c) = v_p \cdot \{\beta^{\mathcal{K}} \cdot c_r^{\mathcal{K}} \cdot \varphi(t_e^{\mathcal{K}}, t_u^{\mathcal{K}}) + (1 - \beta^{\mathcal{K}}) \cdot c\}, \tag{2}$$

This is the sum of the cost of reserving c_r^K cycles under the contract, and the consumption cost of consuming c of these reserved cycles.

Note that when both the reservation cost and the consumption cost are taken into account, it is not true in general that given any two contracts with the same time of establishment (i) if both reserve the same number of cycles, it is cheaper to use the less urgent contract, or that (ii) if both reserve the cycles for consumption at the same time, and both reserve at least as many cycles as are actually used at that time, it is cheaper to use the contract which reserves fewer cycles. As counterexamples, suppose that $t_r < t_1 < t_2 < t_d$ and set $\mathcal{K}_1 = \langle t_p, t_1, 1, 0.25\rangle$, $\mathcal{K}_2 = \langle t_p, t_2, 1, 0.875\rangle$, $\mathcal{K}_3 = \langle t_p, t_2, 2, 0.25\rangle$. Then $\gamma_p^{\mathcal{K}_2}(1)$ is greater than both $\gamma_p^{\mathcal{K}_1}(1)$ and $\gamma_p^{\mathcal{K}_3}(1)$. However, given any two contracts with the same time of establishment and the same value of $\beta^{\mathcal{K}}$, (i) and (ii) both hold. We allow this flexibility in our general pricing model, but in the evaluation sections of this paper we will assume that the service provider fixes a value β and only allows the establishment of contracts $\mathcal{K}$ for which $\beta^{\mathcal{K}} = \beta$. Thus in our evaluations (i) and (ii) hold, and the customer's choice of contract amounts to a choice of the time that the contract is established, the time that the reserved cycles will be consumed, and the number of cycles reserved. Notice also that if $\beta^{\mathcal{K}} = \beta^{\mathcal{K}'}$, $t_e^{\mathcal{K}} = t_e^{\mathcal{K}'}$, and $t_u^{\mathcal{K}} = t_u^{\mathcal{K}'}$, then for all $c \leq c_r^{\mathcal{K}}$, $c' \leq c_r^{\mathcal{K}'}$ we have $\gamma_p^{\mathcal{K}}(c) + \gamma_p^{\mathcal{K}'}(c') = \gamma_p^{\mathcal{K}''}(c + c')$, where $\mathcal{K}''$ is the contract $\langle t_e^{\mathcal{K}}, t_u^{\mathcal{K}}, c_r^{\mathcal{K}} + c_r^{\mathcal{K}'}, \beta^{\mathcal{K}}\rangle$. It follows that when calculating costs in our evaluations, we can assume without loss of generality that given any pair of times t_1, t_2, at most one contract is established at time t_1 for cycles to be consumed at time t_2.

We assume that the customer (that is, the manager of the hybrid infrastructure) runs a planning agent [10] that is in charge of establishing contracts with the cloud computing provider. The planning agent starts to run at time

t_p, the earliest time at which contracts can be established. Let the plan $\mathcal{P}$ be the outcome of a run of the planning agent for the time period $\Delta = [t_p, t_d]$ $\mathcal{P}$ is a set that contains the contracts that have been established between the customer and the cloud computing provider. Let $\mathcal{U}$ be the usage log for the execution of part of the application, which records the use of cycles from the cloud computing provider under plan $\mathcal{P}$. $\mathcal{U}$ is a set containing the values $c_u^{\mathcal{K}}$, for all $\mathcal{K} \in \mathcal{P}$, such that $c_u^{\mathcal{K}}$ is the number of cycles that have been consumed under contract $\mathcal{K}$ during the execution accounted by $\mathcal{U}$. The cost incurred from the cycles reserved by a plan $\mathcal{P}$ and used as recorded by $\mathcal{U}$ is given by:

$$\gamma_p(\mathcal{P},\mathcal{U}) = \sum_{\forall \mathcal{K} \in \mathcal{P}} \gamma_p^{\mathcal{K}}(c_u^{\mathcal{K}}). \tag{3}$$

From now on we will use the notation t^- (resp. t^+) to refer to an instant of time that is infinitesimally earlier (resp. later) than some instant of time t. During the time interval $[t_p, t_r)$, no cycles from the grid or the cloud computing provider are consumed. This is because there is no critical computation to be executed in this time interval, and therefore no need for the customer to seek resources from the grid or the cloud computing provider. During this period of time, any spare cycle from the in-house infrastructure is offered to the P2P grid. For the sake of simplicity we assume that for any t with $t_p \le t < t_r$, all $\int_{t_p}^{t_r^-} i(t) \cdot dt$ cycles are idle, and are, therefore, donated to the grid.

We now consider the number of cycles that will be available during Δ from the three different components of the hybrid infrastructure. Let t_c be the time that the application $\mathcal{A}$ completes its execution. Thus, the number of cycles that are available from the in-house infrastructure to run the application is given by $\int_{t_r}^{t_c} i(t)dt$.

On the other hand, the number of cycles that are available from the P2P grid during Δ will depend on the amount of resources that have been previously donated and the quality of service that the grid is able to deliver. The grid quality of service here is related to the probability of receiving back favors paid to other peers in the grid within a certain time horizon: the favors are donations of cycles. The more rapidly that favors are paid back, the better is the quality of service of the grid. In this paper we define the grid's quality of service to be

$$\Phi = \frac{\int_{t_r}^{t_d} g(t)dt}{\int_{t_p}^{t_r^-} i(t)dt}$$

Finally, the number of cycles that are available from the cloud computing provider at time t is defined by the contracts belonging to $\mathcal{P}$, and is given by

$$\sum_{\{\mathcal{K} \in \mathcal{P} | t_u^{\mathcal{K}} = t\}} c_r^{\mathcal{K}}$$

4. Problem Statement

The problem that we would like to solve is how to schedule $\mathcal{A}$ in the hybrid infrastructure, such that the associated *profit* to the owner of the hybrid infrastructure is maximised. We define the profit of running an application $\mathcal{A}$ in the hybrid infrastructure to be the difference between the utility obtained from running $\mathcal{A}$ and the cost of operating the infrastructure during Δ.

For simplicity, we assume that the application has a workload with very fine granularity, that it is sufficiently parallel that at any instant of time t it is possible, if necessary, to simultaneously schedule work using cycles available in the in-house dedicated infrastructure, in the cloud computing provider and in the P2P grid. Starting at t_r, the scheduler uses all available cycles until the execution of the application is completed, using as many in-house cycles as possible (whose cost is incurred whether or not they are used), then as many as possible of the cycles available from the grid (whose cost is negligible) and, finally, cycles available from the cloud computing provider. Following this scheduling algorithm, under the assumption that no two contracts in $\mathcal{P}$ specify the same consumption time $t_u^{\mathcal{K}}$, the usage log $\mathcal{U}$ associated with the execution of $\mathcal{A}$ in the hybrid infrastructure with the plan $\mathcal{P}$ is such that for all $c_u^{\mathcal{K}} \in \mathcal{U}$,

$$c_u^{\mathcal{K}} = max(0, min(c_r^{\mathcal{K}}, w - \sum_{\{c_u^{\mathcal{K}'} \in \mathcal{U} | t_u^{\mathcal{K}'} < t_u^{\mathcal{K}}\}} c_u^{\mathcal{K}'} - \int_{t_r}^{t_u^{\mathcal{K}}} (i(t) + g(t))dt))$$

Note that in the scenario that we propose, the scheduling of the application is preceded by the reservation of cycles from a cloud computing provider. This makes scheduling relatively simple, but requires a solution to the problem of executing the planning, i.e., deciding when and how many cycles to reserve. Different plans will lead not only to different costs, but also to different availability of cycles for running the application and, therefore, to different values of t_c. In turn, different values for t_c lead to different utilities. In summary, using the fact that cycles obtained from the P2P grid have negligible cost, the profit that is achieved by a plan $\mathcal{P}$ which allows the application $\mathcal{A}$ to be completed by time t_c (where $t_r \leq t_c \leq t_d$) is:

$$\text{Profit}(\mathcal{A}, \Delta, \mathcal{P}, \mathcal{U}) \quad = \quad u(t_c) - \gamma_i(\Delta) - \gamma_p(\mathcal{P}, \mathcal{U}), \tag{4}$$

where $\mathcal{U}$ is the usage log that results from applying the scheduling algorithm to the hybrid infrastructure under $\mathcal{P}$. The aim of the planning algorithm is to find the plan $\mathcal{P}$ that maximises *Profit*$(\mathcal{A}, \Delta, \mathcal{P}, \mathcal{U})$.

5. Planning Algorithms

In this section, we propose planning algorithms that can be used to maximise the profit achieved when running an application on the hybrid infrastructure. As such, the algorithm must first estimate the number of cycles that need to be acquired from the cloud computing provider and then establish the contract or contracts that will maximise the profit.

The total number of cycles needed from the cloud computing provider in order to complete the execution of the application $\mathcal{A}$ at some instant of time t_c is given by:

$$c_e(t_c) = max(0, w - \int_{t_r}^{t_c} (i(t) + g(t))dt).$$

Given the relatively short time for which the application is in execution, it is reasonable to assume that $i(t)$, for $t_r \leq t \leq t_c$, is known at time t_p, when the planner starts to run. Unfortunately, due to the uncertainty inherent in the best-effort P2P grid, it may not be possible at time t_p to predict $g(t)$ accurately for $t_r \leq t \leq t_c$.

Note that if it is possible at time t_p to estimate $c_e(t)$ accurately for any t in $[t_r, t_d]$, then, for the system model that we have defined in Section 3 with the additional assumption that $\beta^{\mathcal{K}}$ is the same for all contracts $\mathcal{K}$ that can be established, it is straightforward to find a plan that maximises the profit. There is a plan consisting of a single contract $\langle t_p, t_u, c_e(t_u), \beta \rangle$ for which the profit is maximal. A simple solver can be used to find the time t_u that specifies this contract. For instance, for the AoN application whose utility is a positive constant for any t_u, $t_u \leq t_r \leq t_d$, (and still making the assumption that only contracts $\mathcal{K}$ with $\beta^{\mathcal{K}} = \beta$ can be established), it is easy to see that the contract required is $\langle t_p, t_d, c_e(t_d), \beta \rangle$, because out of all the contracts that can be established which give rise to the maximum utility, this is the least urgent contract reserving the smallest number of cycles, and hence has the minimum cost.

In the approach sketched above, underestimation of the number of cycles that will be received from the grid leads to an overestimation of $c_e(t)$ and the reservation of more cycles than is necessary. It may be the case that this results in higher reservation costs than if only the cycles that will actually be used had been reserved, and in this case the profit achieved may not be maximal. (It is possible that it may still be maximal if the extra cycles from the grid enable the application to finish execution earlier than expected, resulting in a higher utility.) On the other hand, overestimation of the number of cycles that will be received from the grid leads to the underestimation of $c_e(t)$, and as a result, at time $t_u^{\mathcal{K}}$ there will be still some part of the application workload left to be processed. There will hence be unexpected costs of purchasing the additional cycles from the cloud computing provider required to complete the execution of the application (if this is possible), and the application will complete later than expected and so may produce a smaller utility than expected. Thus the profit will be lower than was expected by the planner at the time it established the relevant contracts, and it is possible that a different choice of contracts would have led to a higher profit.

Since it is not generally possible to predict with complete accuracy the number of cycles that will be available from the grid over a future time interval, we resort to heuristics that try to achieve profits that are as close as possible to the maximal achievable. These heuristics assume that $\beta^{\mathcal{K}} = \beta$ is the same for all contracts $\mathcal{K}$ that can be established, and also that for any t_e, t_u with $t_p \leq t_e < t_u$, $t_r \leq t_u \leq t_d$ and any positive integer c, it is possible to establish a contract $\langle t_e, t_u, c, \beta \rangle$.

Out of all possible heuristics, we concentrate our focus on those that make at most two contracts within the $\Delta = [t_p, t_d]$ time interval. In particular, we assume that one contract ($\mathcal{K}_1$) is always established as soon as possible, at time t_p, while the second contract ($\mathcal{K}_2$), when needed, is established at the instant of time at which the cycles reserved in the first contract were used, i.e., $t_e^{\mathcal{K}_2} = t_u^{\mathcal{K}_1}$. The reason is as follows. The heuristic runs a solver to find the contract $\mathcal{K}_1$ that maximises the profit, given an estimate c'_g (which depends on $t_u^{\mathcal{K}_1}$) of the number of cycles that are going to be received from the grid in the interval $[t_r, t_u^{\mathcal{K}_1}]$. By time $t_u^{\mathcal{K}_1}$ the heuristic can evaluate how accurate the estimate was. If the number of cycles that have been received from the grid in this time interval (we denote this number c_g) is greater than or equal to c'_g, then the execution of the application was completed at or before $t_u^{\mathcal{K}_1}$, and no further action is required. On the other hand, if $c'_g < c_g$ an additional contract may be necessary to complete the execution of the application. At this point the heuristic carries out a new optimisation, this time considering only the workload left to be processed. To avoid having to establish a third contract, the heuristic takes a conservative approach and assumes that no further cycles will be received from the grid before the execution of the application is completed.

In this paper we evaluate different flavours of this heuristic framework that differ from each other in how they produce the estimate c'_g. We discuss each of them in turn.

Omniscient Heuristic. This heuristic produces an optimal plan. We assume in this case that the heuristic has access to an oracle that is able to predict c_g with complete accuracy, i.e., we suppose that $c'_g = c_g$. This heuristic therefore always makes a single contract for the precise amount of extra cycles needed from the cloud computing provider to compute the workload. Thus, this heuristic always achieves the maximum profit for the hybrid infrastructure.

Averse Heuristic. This heuristic is completely averse to the risk of trusting the best-effort grid. Therefore it assumes that $c'_g = 0$. It always establishes a first contract for all the cycles needed to compute the workload, i.e., for the difference between w and the cycles that will be provided by the in-house infrastructure over the relevant time interval. Obviously, in this case there is also no need to make a second contract. Although this heuristic estimates that no cycles will be received from the grid, the scheduling algorithm uses the cycles that may in fact be provided by the grid, potentially reducing the number of cycles consumed from the cloud computing provider.

Oblivious Heuristic. This heuristic is oblivious to the existence of the grid, and therefore it also assumes that $c'_g = 0$. Moreover, when evaluating this heuristic we ignore $g(t)$ and instead assume that $g(t) = 0$ within Δ: another way of thinking about this is that we assume that the scheduling algorithm does not use any cycles available from the grid. Once again, in this case only a single contract is established. Under our assumptions on which contracts can be established and on the pricing model, this heuristic makes the optimal choice of contracts for a hybrid infrastructure that does not have access to a P2P grid. Thus, when we compare the other heuristics with this one, we can measure the value that the P2P grid adds to the hybrid infrastructure.

Predictive heuristic. This heuristic uses an oracle to obtain some knowledge about the grid's behaviour, as the *Omniscient* does, however the oracle used by the predictive heuristic is imperfect. We model this imperfection by associating a nonzero error ξ with the prediction that it makes, i.e. $c'_g = c_g \cdot (1 \pm \xi)$. Since this heuristic may need to establish a second contract to complete the execution of the application, the instant of time to use the cycles of the first contract has to be smaller than t_d, otherwise it would not always be possible to establish a second contract (recall that $t_e^{\mathcal{K}} < t_u^{\mathcal{K}}$).

In summary, the *Omniscient* and *Oblivious* heuristics produce benchmarks for the profit that can be achieved by the execution of the application on the hybrid infrastructure. The *Averse* heuristic reveals the value that the grid adds to the infrastructure. Finally, the *Predictive* heuristic allows us to evaluate the impact that the quality of the estimation of the amount of resources received from the grid has on the profit that can be achieved. Algorithm 1 is the pseudo-code for the planner framework just described.

6. Evaluation of the Heuristics

6.1. Evaluation Metric

To measure the efficiency of a given heuristic, we compare the profit yielded by this heuristic with that yielded by the Omniscient and the Oblivious heuristics. We define the *efficiency* achieved by a heuristic H when running an application $\mathcal{A}$ on the hybrid infrastructure as follows:

$$\mathcal{E}_H = 1 - \frac{Profit_{Omniscient}(\mathcal{A}) - Profit_H(\mathcal{A})}{Profit_{Omniscient}(\mathcal{A}) - Profit_{Oblivious}(\mathcal{A})},$$

where $Profit_{Omniscient}(\mathcal{A})$, $Profit_{Oblivious}(\mathcal{A})$ and $Profit_H(\mathcal{A})$ are, respectively, the profit achieved by the Omniscient heuristic, Oblivious heuristic and H, when scheduling A over the same hybrid infrastructure and under the same conditions.

Notice that the efficiency is not defined if the profit of the Omniscient heuristic is equal to that of the Oblivious heuristic. Note also that 1 is an upper bound for the efficiency, but there is no lower bound for it: the efficiency may be negative. This would be the case if the choice of contracts

Algorithm 1: Planner's algorithm

begin
  at *time* $= t_p$ **do**
  begin
    estimate c'_g according to $Heuristic$
    find $t_u^{\mathcal{K}_1}$ such that Equation 4 is maximised with
      $\mathcal{P} = \{\mathcal{K}_1\}$, $t_e^{\mathcal{K}_1} = t_p$ and $c_r^{\mathcal{K}_1} = w - c'_g - \int_{t_r}^{t_u^{\mathcal{K}_1}} i(t)dt$
    establish contract $\mathcal{K}_1$
  end
  at *time* $= t_u^{\mathcal{K}_1}$ **do**
  begin
    let w' be the number of cycles that still need to be processed
    if $(w' > 0)$
    begin
      find $t_u^{\mathcal{K}_2}$ such that Equation 4 is maximised with
        $\mathcal{P} = \{\mathcal{K}_2\}$, $t_e^{\mathcal{K}_2} = t_u^{\mathcal{K}_1}$ and $c_r^{\mathcal{K}_2} = w' - \int_{t_u^{\mathcal{K}_1^+}}^{t_u^{\mathcal{K}_2}} i(t)dt$
      establish contract $\mathcal{K}_2$
    end
  end
end

made by the heuristic was so bad that the resulting profit was even lower than that attained by the Oblivious heuristic.

6.2. Description of the Scenarios Evaluated

To evaluate the heuristics using the model presented in Section 3 we need to define scenarios by setting values for the constants and instantiating the functions that comprise the model. We first discuss the constants and functions related to the application, and then we discuss those related to the infrastructure.

Regarding the application, we need to define its workload (w), its utility function and its lifetime (t_r and t_d). Based on the experience of the e-Science applications currently used by the OurGrid Community, we set the workload to be that of an application running in a twenty-machine cluster for approximately 12 hours. Thus, we have $w = 864,000$ cycles and $t_d - t_r = 43,200$ seconds (12 hours). We set $t_p = 0$ and assume that Δ is a time interval of 24 hours. Therefore, $t_r = 43,200$ seconds and $t_d = 86,400$ seconds. For the utility function of the *AoN* and *LDuD* applications, the value b gives the *maximum* revenue that can be achieved with the execution of the application: if the execution of the application finishes exactly at t_r, the revenue obtained is b units of utility. We assume that the utility obtained by the completion of the application is directly proportional to its processing demand. Therefore, we set b to be $\mu \cdot w$ (with $\mu > 1$), where μ is the maximum profitability factor of the application. For example, $\mu = 2$ means that 2 units of utility is obtained as a result of the execution of the application for each unit of workload that the application contains. For the *GDuD* application we set $a = -1.00017$ and $b = 1.00054 \cdot \mu \cdot w$, which provides a utility that decreases slowly until about 8 hours after t_r, and has a sharp decrease after that.

We assume that the cost of the in-house infrastructure over a time interval is equal to the number of cycles that are available from the in-house machines over that interval, i.e. we set $v_i = 1$. We set v_p, the cost of reserving a cycle from the cloud service provider at time t_p and consuming it at time t_d, as a function of v_i. In the scenarios we evaluated we set $v_p = 0.5 \cdot v_i$, assuming that the cloud provider is able to operate its infrastructure more than twice more efficiently than the in-house infrastructure is operated.

As indicated earlier, we assume that the cloud service provider sets a value β and only allows contracts $\mathcal{K}$ to be established for which $\beta^{\mathcal{K}} = \beta$. In order to evaluate different providers' profiles, we consider three different values for β: 1/2, 2/3, and 3/4.

The function φ, which reflects how the reservation fees vary with different contract establishment times and consumption times, is set as follows. We set $\varphi(t_1, t_2)$ to be a hyperbolically decaying function of $t_2 - t_1$ given by:

$$\varphi(t_1, t_2) = \frac{18,000}{t_2 - t_1 + 5,600} + 0.8,$$

which leads to a value for the contracts to be approximately 4 for the most urgent ones ($\varphi(t, t^+)$) and approximately 2, 5 for the contracts established at least 12 hours in advance. This allows us to investigate the behaviour of the heuristics in scenarios in which the cost of the contracts change substantially.

We specify three grid profiles by their quality of service Φ: a *bad* quality grid with $\Phi = 0.1$, a *medium* quality grid with $\Phi = 0.5$, and a *good* quality grid with $\Phi = 0.9$. For instance, for the good quality grid, the number of the cycles that are available from the grid during $[t_r, t_d]$ is 90% of the number donated to the grid during $[t_p, t_r^-]$ ($\int_{t_r}^{t_d} g(t) \cdot dt = 0.9 \cdot \int_{t_p}^{t_r^-} i(t) \cdot dt$). However, it is reasonable to suppose that the number of cycles available from the grid during a subinterval of $[t_r, t_d]$ is smaller for an earlier subinterval than for a later subinterval of the same length, because when the later time interval starts a smaller number of outstanding favors (donated resources) need to be paid back from the grid to the peer representing the in-house infrastructure. We therefore set $g(t)$ to be the following decreasing function of t:

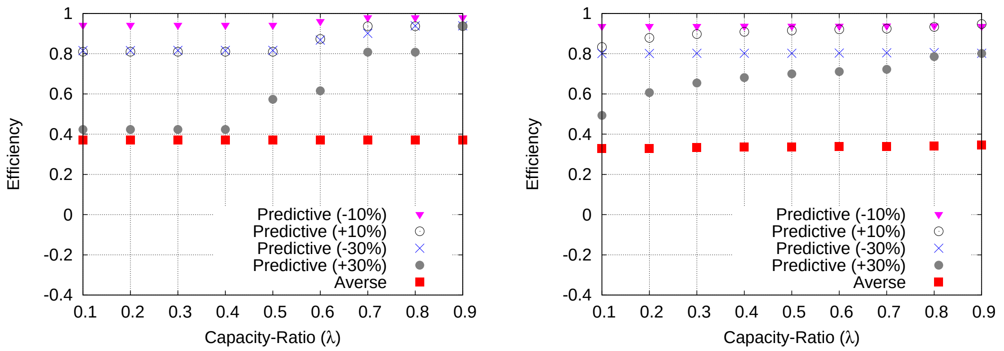
$$g(t) = 2 \cdot \Phi \cdot (t_d - t_r)^{-2} \cdot (t_d - t) \cdot \int_{t_p}^{t_r^-} i(t')dt'.$$

6.3. Numerical Results

For simplicity, we assume that the capacity of the in-house infrastructure does not change during the time interval Δ, and we introduce the *capacity ratio* λ ($0 < \lambda < 1$) which expresses how much of the workload w can be processed in-house. For instance, $\lambda = 0.5$ means that the in-house capacity can compute 50% of the application's workload. Note that when $\lambda = 0$ there are no in-house resources available during Δ, and when $\lambda = 1$ all the workload can be processed in-house. Neither of these scenarios are of interest to our study. The figures for this subsection show graphs of the efficiency versus the capacity ratio λ for $\lambda \in \{0.1, 0.2, \ldots, 0.9\}$. We will first describe the general results for all scenarios evaluated, and then discuss the impact of particular parameters on the efficiency obtained by the heuristics.

In general, t_c is calculated as a compromise between the loss of utility as t_c increases and the reduction in cost by avoiding to contract at the cloud provider the cycles that are obtained from the in-house infrastructure and the P2P grid during the time interval $[t_r, t_c]$. However, for the *LDuD* application, in all scenarios evaluated, all heuristics took the decision to contract all extra cycles to be used exactly at t_r. This is because for the *LDuD* application, as t_c increases, the loss in utility is more important the the cost reduction associated. Thus, in all scenarios the profit yielded by both the Omniscient and the Oblivious heuristics were the same and the efficiencies for the other heuristics were not defined. Therefore, in the following we discuss only the efficiency values for the *AoN* and the *GDuD* applications.

Figure 2 shows the average efficiency obtained by the heuristics on all the scenarios evaluated, Figure 2(a) for the *AoN* application, and Figure 2(b) for the *GDuD* application. These figures show the efficiency of the *Averse* heuristic, and of the *Predictive* heuristic for the cases where the heuristic's estimate of the number of cycles available from the peer-to-peer grid is 10% smaller, 10% greater, 30% smaller or 30% greater than the true number.



(a) *AoN* (b) *GDuD*

Figure 2: Average efficiency versus capacity ratio (λ).

As expected, for the relatively low errors in the estimation of the cycles received from the grid, the *Predictive* heuristic provides better results than the *Averse* heuristic, since it makes use of its partial knowledge of the grid's behaviour. When the *Predictive* heuristic underestimates the number of cycles available from the grid, the results are better than when it overestimates this number. This is because when it overestimates the number of cycles available from the grid it reserves fewer cycles from the cloud computing provider in its first contract than are necessary to complete the application, and has to establish a second contract later on. This results in higher reservation fees than if all the cycles necessary to complete the application had been reserved in the first contract.

For the *AoN* application (see Figure 2(a)), the *Predictive* heuristic has higher efficiency for larger values of λ, because the larger the in-house capacity is the more idle cycles it donates to the grid, and thus the more cycles the grid is likely to pay back within the time interval $[t_r, t_c]$. For the *GDuD* application (see Figure 2(b)) λ has less impact on the efficiency of the *Predictive* heuristic, because the completion time for the execution of the algorithm is generally earlier than t_d, and so the heuristic does not take advantage of all the in-house and grid capacity available before the deadline t_d.

The *Averse* heuristic ignores the grid and reserves all the cycles it needs (and which will not be provided in-house) from the cloud computing provider.

This results in a very low efficiency for all the scenarios we investigate. This efficiency is not affected by the value of λ. When λ increases, the number of cycles likely to be available from the grid increases, reducing the probable cost of running the application. However, this cost reduction is also achieved by the other heuristics. By taking a closer look at our results, we discovered that when λ increases the profits obtained by the *Omniscient*, *Oblivious* and *Averse* heuristics increase by the same proportion, leaving the efficiency for the *Averse* heuristic unchanged.

We evaluated the impact on efficiency of the quality of service Φ of the grid, setting $\Phi \in \{0.1; 0.5; 0.9\}$. Figure 3 presents the average efficiency obtained for each value of Φ, for both The *AoN* and the *GDuD* applications. Not surprisingly, the efficiency of the *Averse* heuristic is unaffected by Φ. On the other hand, the *Predictive* heuristic performs better as the grid's quality of service improves, and achieves efficiency 1 for some of the scenarios in which Φ and λ are large.

Figure 4 presents the average efficiency obtained for different values of β. It can be seen from this figure that as β increases the efficiency of the heuristics decreases, and that β has a greater effect on the *Averse* heuristic, which reserves more cycles from the cloud computing provider than the other heuristics do.

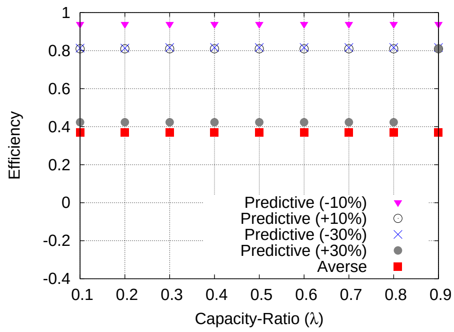
In summary, our results show that it is important to use at least knowledge about the grid behaviour when deciding which contracts to establish with the cloud computing provider. So, constructing an estimation for the behaviour of the grid is essential for making contracts that lead to high efficiency in the use of the hybrid infrastructure.

7. Estimating the Quality of Service of a P2P Grid

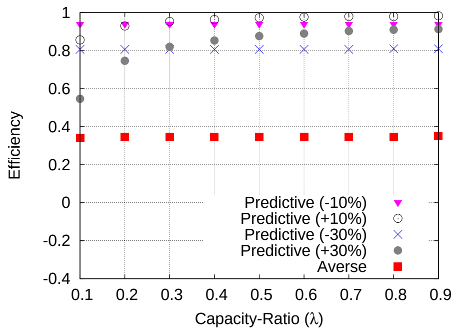
In this section we propose and evaluate an analytical model that can be used to estimate, at a given instant of time, the amount of resources that will be reclaimed from a best-effort P2P grid in the near future.

7.1. Model of the P2P Grid

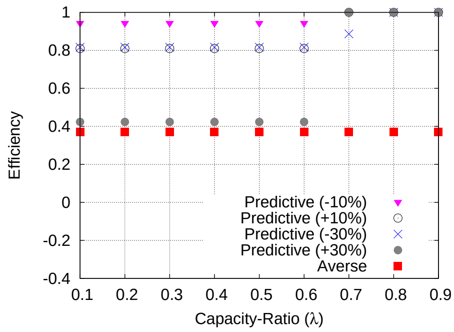
Our model is of a P2P grid that operates an incentive mechanism called the "Network of Favors" [12] which is used in the OurGrid middleware [1]. This incentive mechanism uses information that each peer gathers about its past interactions with other peers in the grid. For each peer p' with which it interacts, peer p records a *balance* that represents the difference between



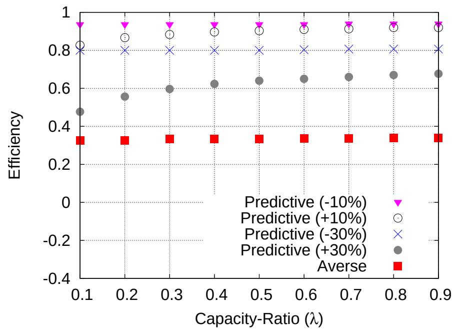
(a) $\Phi = 0.1$, AoN



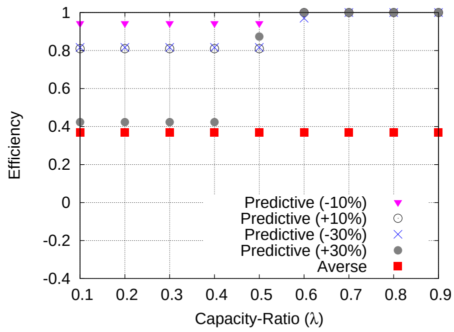
(b) $\Phi = 0.1$, $GDuD$



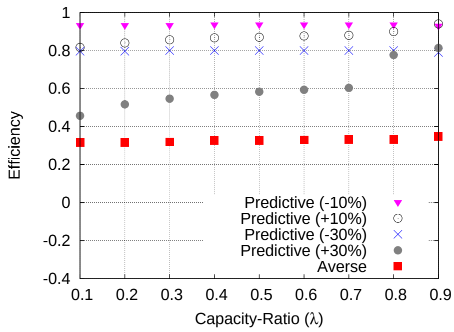
(c) $\Phi = 0.5$, AoN



(d) $\Phi = 0.5$, $GDuD$

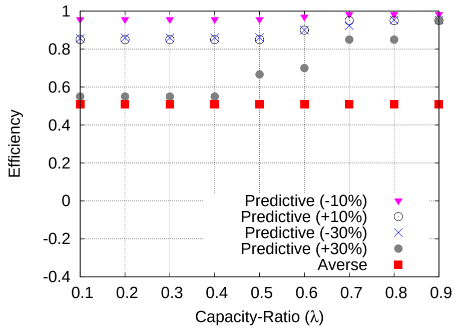


(e) $\Phi = 0.9$, AoN

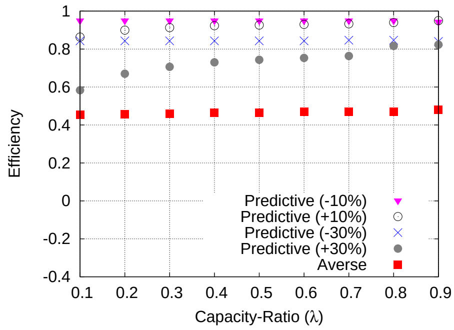


(f) $\Phi = 0.9$, $GDuD$

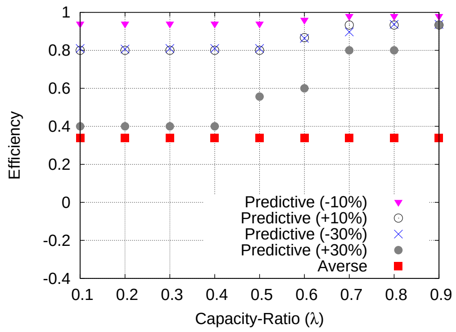
Figure 3: Efficiency versus capacity ratio (λ) for grids with different quality of service (Φ).



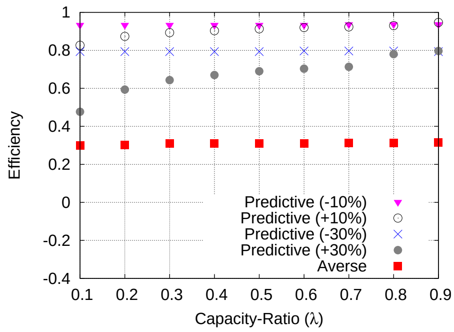
(a) $\beta = 1/2$, AoN



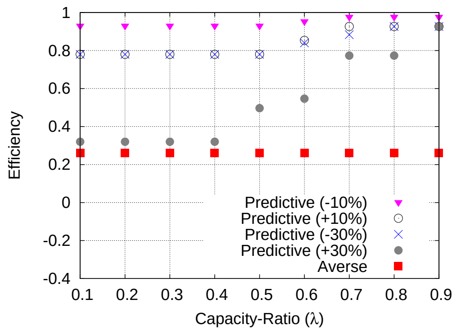
(b) $\beta = 1/2$, $GDuD$



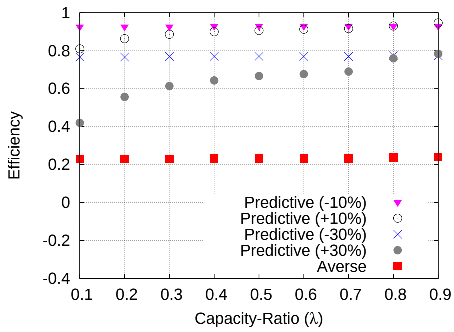
(c) $\beta = 2/3$, AoN



(d) $\beta = 2/3$, $GDuD$



(e) $\beta = 3/4$, AoN



(f) $\beta = 3/4$, $GDuD$

Figure 4: Efficiency versus capacity ratio (λ) for different values of β.

the number of resources that it has received from p' and the number of resources that it has donated to p'. This balance is zero if p and p' have never interacted. It increases when p donates resources to p' and decreases when p consumes resources from p', unless it is already zero: as a defence against whitewash attacks, the balance never takes negative values. Peer p allocates its idle resources proportionally according to the balances of the peers that request these resources. If all these peers have zero balances, p distributes its idle resources equally among them. This simple approach has been shown to be efficient at discouraging free riding.

The model we propose of the P2P grid is designed to be used by a peer p_0 (which we will call the "local peer") that wants to estimate the number of resources it will obtain from the grid during a given period of time in the near future, during all of which time it will request resources from the grid. In the eyes of the local peer, the grid is composed by a set of remote peers $G = \{p_1, p_2, \cdots, p_N\}$, where N is the number of peers in the grid other than p_0. Each peer can be either in *consuming* state - when it is requesting resources from the grid - or in *donating* state - when it has idle resources available for use by other peers in the grid. A peer will never be in both states at the same time, as we assume that peers only ask for remote resources if there are not enough local resources available. We write $G_c(t)$ for the set of peers in G that are in consuming state at time t. We assume that there is high contention for the grid's resources; specifically, we assume that every resource a donating peer makes available to the grid is consumed by some peer, and that if there are any peers in consuming state with positive balance then all the resources that the donating peer makes available will be consumed by these peers, with none left over for peers with zero balance.

Assuming that $b_k^i(t)$ is the balance peer p_i associates with peer p_k at time t, we can introduce the aggregate balance of all consuming peers other than the local peer p_0 on the provider peer p_i at time t, given by

$$B_i(t) = \sum_{\forall p_k \in G_c(t)} b_k^i(t)$$

Therefore, as a result of the Network of Favors mechanism, a donating peer p_i provides part of the $r_i(t)$ resources available on its infrastructure at time t to the consuming peer p_0 according to the following equation.

$$R_0^i(t) = \begin{cases} \frac{b_0^i(t)}{B_i(t)+b_0^i(t)} \cdot r_i(t) & if\ B_i(t) + b_0^i(t) > 0 \\ \\ \frac{r_i(t)}{|G_c(t)|+1} & otherwise \end{cases}$$

When resources are consumed, the balances from both consumer and provider are updated. For the sake of simplicity, we assume that all peers use the same accounting function, and that each donated resource results in an addition or subtraction of one unit from the relevant balances (with the exception that when a balance is zero no units are subtracted from it). Thus, the changes at time t in the balances b_0^i and b_i^0 for the consuming peer p_0 and a donating peer p_i are given by

$$\frac{b_0^i(t)}{dt} = -min(R_0^i(t), b_0^i(t)) \tag{5}$$

and

$$\frac{b_i^0(t)}{dt} = R_0^i(t)$$

The total amount of resources obtained from the grid by the local peer p_0 from all peers at time t is:

$$R_0(t) = \sum_{\forall p_i \in G \setminus G_c(t)} R_0^i(t) \tag{6}$$

However, in practice the local peer will not be able to apply Equation 6 to determine the number of resources that it will obtain from the grid, as the values of many parameters are difficult or even impossible for this peer to know. For instance, the aggregate balance $B_i(t)$ is a piece of information that is stored only at p_i. Instead of using Equation 6, we propose a prediction model that uses more generic parameters which can be estimated more easily. The prediction model takes into account the possible presence of A peers with altruistic behaviour (they are hardly ever in consuming state). The balances that altruistic peers record will be zero most of the time, as altruistic peers rarely (if ever) ask favors to the grid. Thus, altruistic peers will usually distribute their idle resources equally among all consuming peers, no matter whether one of the consuming peers donated more than others in the past.

For our model, we suppose that at any time t each peer has an independent probability ρ of being in donating state, and that the resources that a donating peer provides at time t is distributed independently of t and of the consuming peer's identity, and has mean $\bar{r}$. If the local peer p_0 is in consumption state at time t, it follows that the expected number of resources available from all the donating peers at this time is $\bar{r} \cdot N \cdot \rho$.

We estimate the number of resources obtained from the grid by the local peer p_0 at time t by:

$$E(t) = \begin{cases} \bar{r} \cdot (N \cdot \rho - A) \cdot \frac{\bar{b}_0(t)}{\bar{B}(t)+\bar{b}_0(t)} + \frac{A \cdot \bar{r}_A}{N \cdot (1-\rho)+1} & if\ \bar{B}(t) + \bar{b}_0(t) > 0 \\ \bar{r} \cdot N \cdot \rho \cdot \frac{1}{N \cdot (1-\rho)+1} & otherwise \end{cases} \tag{7}$$

where $\bar{b}_0(t)$ is the estimated aggregate balance of the local peer p_0 on all donating peers at time t; $\bar{B}(t)$ is the estimated aggregate balance of all consuming peers other than the local peer on all donating peers at time t; and $\bar{r}_A$ is the mean number of resources available from altruistic peers at time t.

Nevertheless, the local peer wants to be able to estimate not only the amount of resources it will obtain at time t, but more importantly the estimated amount of resources to be obtained in a time period $[t_s, t_f]$ in the near future, given by:

$$\Psi([t_s, t_f]) = \int_{t_s}^{t_f} E(t)dt \tag{8}$$

When applying Equation 8, we have to consider the change rate over time for $\bar{B}(t)$ and $\bar{b}_0(t)$. We make the simplifying assumption for our model that $\bar{B}(t)$ is constant over the time we are predicting. This assumption is based on the fact that, while consuming peers have their balances decreased by consuming resources, the providing peers have their balances increased by donating them. Thus, if peers frequently change their state from donating to consuming and vice-versa during the time period for which we are predicting, the aggregate balance of all peers will not change much over this short period of time, and we can use its mean value over this time period as an estimate of its value at any particular time during that period.

As mentioned before, the balance b_0^i will not increase, and may decrease, over a time interval during which the local peer p_0 is in consuming state. When this balance decreases, it affects the number of resources that the local peer can expect to be obtained from p_i. Thus, we need to model the dynamics of these balances. Since we do not know the value $R_0^i(t)$ we cannot use Equation 5 directly. However, we can estimate $R_0^i(t)$ by $E(t)$, using Equation 7, and use its result to estimate the decay rate of the aggregate balance $\bar{b}_0(t)$ while p_0 is in consuming state. We assume that no decrease to the aggregate balance $\bar{b}_0(t)$ will result from the donation of resources to p_0 by altruistic peers, because if p_i is an altruistic peer then b_i0 is very likely to be zero before (and after) the donation. Our estimate of the change rate of $\bar{b}_0(t)$ is therefore:

$$\frac{\bar{b}_0(t)}{dt} = -min\left(E(t) - \frac{A \cdot \bar{r}_A}{N \cdot (1-\rho) + 1}, \bar{b}_0(t)\right) \tag{9}$$

We can now use estimates of the aggregate balances $\bar{b}_0(t_s)$, $\bar{B}(t_s)$ at the initial time t_s, along with Equation 7 and the differential Equation 9, to calculate $E(t)$; and then apply Equation 8 to obtain an estimate of the total amount of resources that will be received by p_0 from the grid during the time period $[t_s, t_f]$, as required.

7.2. Evaluation of the grid model

We evaluate the grid model by comparing the prediction that it gives for the amount of resources that the local peer will receive with simulation results using field data. For our simulations we have used traces obtained from the execution of real grids, provided by the Grid Workloads Archive (GWA) [13, 35], an initiative of University of Delft for centralising access to workload traces from grid environments. Among the traces that GWA provides, we have chosen the ones from NorduGrid, which is a grid for academic researchers in nordic countries that has been operating since 2002 [36]. We have chosen this trace because, for the purposes of our simulations, it is the most suitable of all traces available: it has the highest number of sites (75 sites) with resource contention scenarios (781,370 tasks run), lasts for a long time (about 3 years) with most of the applications being bag-of-tasks.

As we wanted to simulate scenarios with a number of peers higher than the number of sites available on the trace, we divided the trace into time windows of 2 months and randomly selected sites from each time window, assigning a different peer to each site, until we had the desired number of peers. We then grouped together all the behaviours of the selected sites to make the workload. The number of peers (N) is a parameter of the simulator. We use a normal distribution to set the amount of resources for each peer, based on the work by Kee et al. [37], which models the number of nodes per cluster on computational grids. The mean amount of resources per peer ($\bar{r}$) used on the normal distribution is an input of the simulator, and the standard deviation value is set such that 99.7% of the distribution values are between $0.5\bar{r}$ and $1.5\bar{r}$. The workload was built based on the following job information available: submit time, run time, number of requested processors and job origin site for each task recorded in the trace. The workload was filtered in order to use only tasks that requested a single processor, since we are simulating a desktop grid infrastructure that only supports bag-of-tasks submissions. This filtering removed less than 1% of the trace from the simulation workload. For the sake of simplicity, we also considered that machines in the grid were homogeneous and each contained a single processor.

The prediction model given in Subsection 7.1 estimates the amount of resources that the local peer will be obtain from the grid in a specified time interval. However, a peer with low resource requirements (or with high resource requirements which are however almost all met by its in-house infrastructure) may consume fewer resources than are available to it from the grid. Since the prediction model does not consider the amount of resources requested from the grid by the local peer, we have to consider this issue in our evaluation. In order to do that, we evaluated the model by comparing the ratio between the estimated and requested amount of resources (ER) with the ratio between the obtained and requested amount of resources (OR). We set an upper bound of 100% for ER, meaning that if the amount of resources that the model estimated was higher than the amount of resources requested then we reset ER to 100%, as no more than the amount of resources requested can actually be used. Formally, given a simulation S and an estimate *estimated(S)* of the amount of resources that will be available from the grid during the time period simulated by S, where the estimate is obtained by using the prediction model, the values of ER and OR for this simulation are given by:

$$ER = min\left(\frac{estimated(S)}{requested(S)}, 1\right)$$

$$OR = \frac{obtained(S)}{requested(S)}$$

where *requested*(S) and *obtained*(S) are the amounts of resources that were requested and obtained over the course of the the simulation.

To evaluate the model for this simulation, we calculate an error which is defined as the difference between the ratio ER given by the model and the ratio OR given by the simulations:

$$\xi = ER - OR \tag{10}$$

If in some scenario the amount of resources available from the grid is larger than the amount that the peer requests, it is likely that for this scenario the error will be zero, because provided that the estimate of the amount of resources that will be available is at least as large as the amount requested, both ER and OR will take value 1. Since our prediction model assumes that there is high contention in the grid, we evaluate only the parts of the trace for which the amount of obtained resources is smaller than the amount requested, i.e., when the demand for resources outstrips the supply. When supply exceeds demand, all resource requests will be satisfied.

We simulated each scenario several times using each peer in turn as the local peer, setting different values for the number of peers N (the values used for N were 100 and 200) and for $\bar{r}$ (the values used for $\bar{r}$ were 10, 20, and 40). We divided the resulting 2-month-long simulations into shorter simulations with lengths of 1, 2, ... 50 hours. For each of these shorter simulations, we used the grid model to predict the amount of resources which would be available from the grid during the time period of the simulation, and calculated the prediction error ξ for this simulation. Fixing $N = 200$ and varying values for $\bar{r}$, then fixing $\bar{r}$ and varying values for N, and for each $\Delta t \in \{1, 2, \ldots 50\}$ we calculated the mean value of ξ for the set of simulations whose length was Δt hours. Then we calculated the mean $\bar{\xi}$ of these 50 mean values.

Table 1 summarises the results. The first column gives the number of peers N in the grid. The second column gives the mean amount of resources offered by a donating peer ($\bar{r}$). The third column is the mean of the mean overestimated prediction errors (positive errors) over all the shorter simulation lengths ($\bar{\xi}_+$). The forth column is the mean of the mean underestimated prediction errors (negative errors) over all the shorter simulation lengths ($\bar{\xi}_-$). The last column gives the frequency of positive errors, i.e., the number of simulations for which the model overestimated the amount of resources received from the grid, divided by the total number of simulations.

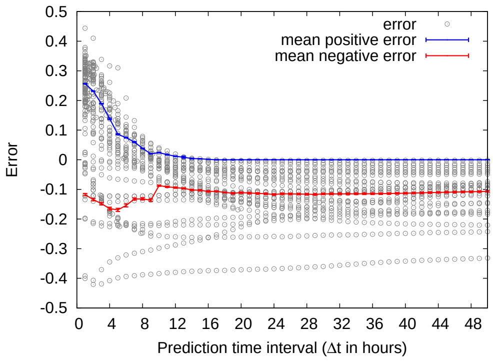
Out of the pairs of values for N and $\bar{r}$ for which we have run simulations, the one that gives rise the highest value of overestimated $\bar{\xi}_+$ is $N = 200$, $\bar{r} =$

N	$\bar{r}$	$\bar{\xi}_+$	$\bar{\xi}_-$	**%Overestimated**
100	20	2.43%	11.64%	13.06%
200	10	0.06%	4.27%	14.35%
200	20	1.74%	10.15%	18.26%
200	40	2.76%	11.09%	14.85%

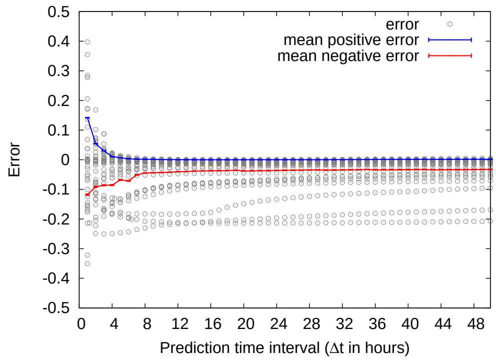
Table 1: Mean of mean overestimated ($\bar{\xi}_+$) and underestimated ($\bar{\xi}_-$) prediction errors

40, for which overestimated $\bar{\xi}_+ = 2.76\%$ and underestimated $\bar{\xi}_- = 11.09\%$. For the same value of N and the lower value 10 for $\bar{r}$, overestimated $\bar{\xi}_+$ and underestimated $\bar{\xi}_-$ are only 0.06% and 4.27%, respectively. It happens because when peers offer fewer resources, there is more contention in the grid. Since the prediction model assumes that the grid has high contention, the model is more accurate when $\bar{r}$ is low. The first and the third rows of the table show that when $\bar{r} = 20$, the values of overestimated and underestimated errors are larger for larger values of N. This is because the prediction model makes an error in estimating the amount of resources available from each peer, and (as can be seen from the fifth column) more often makes negative than positive errors. As a result, the sum of the errors in the estimate for each peer is larger when the number of peers is larger. From the fifth column, we can see that the frequency of overestimated errors lies between 13% and 19% for each of the pairs of values for N and $\bar{r}$ simulated. This result is relevant to situations in which overestimating the amount of resources to be obtained from the grid has worse consequences than underestimating this amount: one such situation is the contract planning presented in this paper.

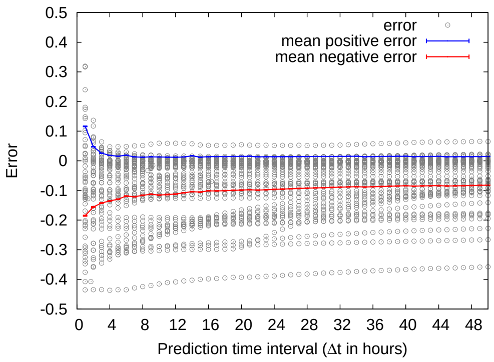
Figure 5 shows all prediction errors ξ for each value of Δt in all simulated scenarios. In addition to the errors, the figure also shows the mean of positive errors (overestimations) and negative errors (underestimations), with confidence interval bars for a 95% confidence level. It can be seen that most of the prediction errors ξ are negative for all scenarios. It is due to the pessimistic estimation made by the prediction model, which considers that overestimating is worse than underestimating for most of the cases, including the application presented in this paper. Moreover, the longer the prediction time interval is, the more accurate the prediction tends to be. We think the reason for this is as follows. When a peer is consuming, the balances that other peers record for it decrease until either they reach zero or the peer finishes consuming. When all these balances are zero, the peer can only



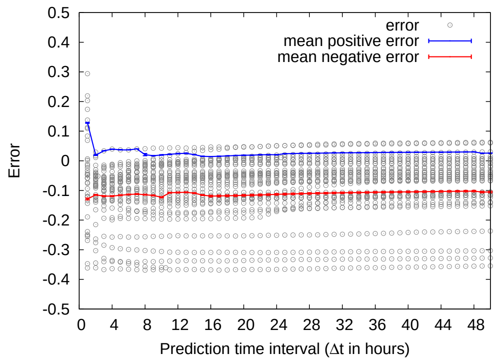
(a) $N = 100$, $\bar{r} = 20$



(b) $N = 200$, $\bar{r} = 10$



(c) $N = 200$, $\bar{r} = 20$



(d) $N = 200$, $\bar{r} = 40$

Figure 5: Absolute error values ξ, mean overestimated prediction errors $(\bar{\xi}_+)$ and underestimated prediction errors $(\bar{\xi}_-)$ for different prediction time interval sizes $(\Delta(t)$ in hours).

obtain resources from altruistic peers. Since the amount of resources it receives from altruistic peers does not change much over time, the model can estimate this value more accurately than the amount of resources received from non-altruistic peers, and so the model makes more accurate predictions in cases where consuming peer's balance is zero for a long period of time.

We now retrofit the prediction given by this model in the *Predictive* heuristic presented in Section 5. For a given grid prediction model, the average efficiency of the *Predictive* heuristic can be expressed as:

$$p \cdot \mathcal{E}_{Predictive}(\bar{\xi}_+) + (1 - p) \cdot \mathcal{E}_{Predictive}(\bar{\xi}_-),$$

where $\mathcal{E}_{Predictive}(\xi)$ is the efficiency of the heuristic for an error ξ, p is the probability of the prediction model overestimate the amount of resources to be received within a given time interval, and $\bar{\xi}_+$ (resp. $\bar{\xi}_-$) is the average overestimation (resp. underestimation) error.

For the *AoN* application, the prediction is made for a 12-hour interval. From our assessment, in this case, the grid model proposed has $\bar{\xi}_+ = 1\%$ and $\bar{\xi}_- = 9\%$, leading to an average efficiency of 96.69% in the worst case ($N = 200$ and $\bar{r} = 20$). On the other hand, the *GDuD* application has running times of less than two hours. In this case, $\bar{\xi}_+ = 16\%$ and $\bar{\xi}_- = 14\%$ which gives a worst case efficiency of 90.05%.

8. Conclusion

In this paper we have extended our previous research on business-driven management of a hybrid IT infrastructure [11], reporting extended models and new results on this topic. We believe that this work will assist the owners of IT infrastructures, by providing business-driven heuristics to decide when to use in-house resources, when to use P2P grid resources, and when to reserve resources from cloud computing providers.

A. Summary of Principal Notation

Symbol	**Meaning**
Δ	Time interval within which application should be executed;
$\mathcal{A}$	Application to be executed;
w	Application's processing demand in cycles;
t_r	Time the application is ready for execution;
$u(t)$	Application's utility function;
t_d	Deadline for the application;
t_c	Time the application is completed;
$i(t)$	# cycles available at time t from in-house infrastructure;
$g(t)$	# cycles available at time t from P2P grid;
$p(t)$	# cycles available at time t from cloud computing provider;
$\mathcal{K}$	Contract between customer and cloud computing provider;
$t_e^{\mathcal{K}}$	Time at which the contract is established;
$t_u^{\mathcal{K}}$	Time at which the cycles will be used;
$c_r^{\mathcal{K}}$	# cycles reserved under contract $\mathcal{K}$;
$\beta^{\mathcal{K}}$	Variable reflecting the relative costs of reservation and consumption;

$\gamma_p^{\mathcal{K}}(c)$	Cost of contract $\mathcal{K}$ for consuming c of the reserved cycles;
$\mathcal{P}$	The plan (set) of established contracts;
$\mathit{Profit}(\mathcal{A}, \Delta, \mathcal{P}, \mathcal{U})$	Profit from executing application $\mathcal{A}$, within Δ, under plan $\mathcal{P}$, with usage log $\mathcal{U}$;
$\gamma_i(\Delta)$	Cost of maintaining the dedicated in-house infrastructure;
$\varphi(t_e^{\mathcal{K}}, t_u^{\mathcal{K}})$	Function reflecting how the reservation fee per cycle varies with the urgency of the contract;
v_i	Fixed cost of each cycle available on the in-house infrastructure;
v_p	Cost of reserving a cycle from the provider at t_p and consuming it just before t_d;
$c_u^{\mathcal{K}}$	# cycles consumed under contract $\mathcal{K}$;
$[t_s, t_f]$	Time period over which the grid's behaviour is predicted;
b_k^i	Balance that peer p_i associates with peer p_k;
p_0	The local peer;
$\bar{b}_0(t)$	Estimated aggregated balance of the local peer on all donating peers at time t;
N	Number of peers in the grid other than the local peer;
$\bar{r}$	Mean amount of resources provided by a donating peer;

Table 2: Summary of Principal Notation.

References

[1] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, M. Mowbray, Labs of the world, unite!!!, Journal of Grid Computing 4 (3) (2006) 225–246. doi:10.1007/s10723-006-9040-x.
URL `http://dx.doi.org/10.1007/s10723-006-9040-x`

[2] B. Yochai, Sharing Nicely: On Shareable Goods and the Emergence of Sharing as a Modality of Economic Production, The Yale Law Journal 114 (2004) 273–358.
URL `http://www.yalelawjournal.org/archive_abstract.asp?id=94`

[3] F. Brasileiro, E. Araújo, W. Voorsluys, M. Oliveira, F. Figueiredo, Bridging the High Performance Computing Gap: the OurGrid Experience, in: Proc. 1^{st} Latin-American Grid Workshop (co-located with CCGrid'2007), Rio de Janeiro, Brazil, 2007.

[4] E. C. Araujo, W. Cirne, G. Mendes, N. Oliveira, E. P. Souza, C. Galvao, E. S. Martins, The SegHidro Experience: Using the Grid to Empower a Hydro-Meteorological Scientific Network, in: Proc. of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), Melbourne, Australia, 2005.

[5] A. Wilter, C. Osthoff, C. Oliveira, D. E. B. Gomes, E. Hill, L. E. Dardenne, P. M. Barros, P. A. A. G. L. Loureiro, R. Novaes, P. G. Pascutti, The BioPAUÃ Project: A Portal for Molecular Dynamics Using Grid Environment, Brazilian Symposium on Bioinformatics.

[6] A. N. Duarte, W. Cirne, F. Brasileiro, P. Machado, GridUnit: Software Testing on the Grid, in: Proc. 28th ACM/IEEE International Conference on Software Engineering (ICSE'06), Shanghai, China, 2006.

[7] C. Osthoff, F. Oliveira, C. Oliveira, A. Vassalo, W. Cirne, A Grid Computing Testbed for EM Algorithm Financial Market Applications, in: Third IFIP Conference on E-Commerce, E-Business and E-Goverment, Guarujá, SP, 2003, pp. 583–590.

[8] W. Voorsluys, E. Araujo, W. Cirne, C. G. ao, E. Souza, E. Cavalcanti, Fostering Collaboration to Better Manage Water Resources, Proceedings of GCE 2005: Workshop on Grid Computing Portals.

[9] F. I. Popovici, J. Wilkes, Profitable services in an uncertain world, in: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC'05), IEEE Computer Society, Washington, DC, USA, 2005, p. 36. doi:http://dx.doi.org/10.1109/SC.2005.58.

[10] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the Condor experience., Concurrency - Practice and Experience 17 (2-4) (2005) 323–356.

[11] P. D. Maciel Jr., F. Figueiredo, D. Candeia, F. Brasileiro, A. Coêlho, On the Planning of a Hybrid IT Infrastructure, IEEE/IFIP Network & Operations Management Symposium (NOMS'08).

[12] N. Andrade, F. Brasileiro, W. Cirne, M. Mowbray, Automatic grid assembly by promoting collaboration in peer-to-peer grids, J. Parallel Distrib. Comput. 67 (8) (2007) 957–966. doi:http://dx.doi.org/10.1016/j.jpdc.2007.04.011.

[13] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, D. H. J. Epema, The Grid Workloads Archive, Future Gener. Comput. Syst. 24 (7) (2008) 672–686. doi:http://dx.doi.org/10.1016/j.future.2008.02.003.

[14] S. Camorlinga, B. Schofield, Modeling of workflow-engaged networks on radiology transfers across a metro network, IEEE Transactions on Information Technology in Biomedicine 10 (2) (2006) 275–281.

[15] A. S. McGough, A. Akram, L. Guo, M. Krznaric, L. Dickens, D. Colling, J. Martyniak, R. Powell, P. Kyberd, C. Kotsokalis, GRIDCC: real-time workflow system, in: WORKS '07: Proceedings of the 2nd workshop on Workflows in support of large-scale science, ACM, New York, NY, USA, 2007, pp. 3–12. doi:http://doi.acm.org/10.1145/1273360.1273362.

[16] L. Weigang, M. V. P. Dib, D. A. Cardoso, Grid Service Agents for Real Time Traffic Synchronization, in: WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, Washington, DC, USA, 2004, pp. 619–623. doi:http://dx.doi.org/10.1109/WI.2004.75.

[17] R. Kuntschke, T. Scholl, S. Huber, A. Kemper, A. Reiser, H.-M. Adorf, G. Lemson, W. Voges, Grid-Based Data Stream Processing in e-Science, in: E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, 2006, p. 30. doi:http://dx.doi.org/10.1109/E-SCIENCE.2006.78.

[18] A. A. Sawchuk, E. Chew, R. Zimmermann, C. Papadopoulos, C. Kyriakakis, From Remote Media Immersion to Distributed Immersive Performance, in: ETP '03: Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence, ACM, New York, NY, USA, 2003, pp. 110–120. doi:http://doi.acm.org/10.1145/982484.982506.

[19] B. Plale, D. Gannon, D. A. Reed, S. J. Graves, K. Droegemeier, R. Wilhelmson, M. Ramamurthy, Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD, in: Computational Science - ICCS 2005, 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part II, 2005, pp. 624–631.

[20] A. Plaza, D. Valencia, J. Plaza, P. Martinez, Commodity cluster-based parallel processing of hyperspectral imagery, J. Parallel Distrib. Comput. 66 (3) (2006) 345–358. doi:http://dx.doi.org/10.1016/j.jpdc.2005.10.001.

[21] Control and monitor the high energy physics experiments.
URL `http://cmsdoc.cern.ch/cms/TRIDAS/Temp/CMS\_DAQ\_TDR.pdf`

[22] Y. Zhang, A. Sivasubramaniam, Scheduling best-effort and real-time pipelined applications on time-shared clusters, in: SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures, ACM, New York, NY, USA, 2001, pp. 209–219. doi:http://doi.acm.org/10.1145/378580.378646.

[23] S. Funk, J. Goossens, S. Baruah, On-line scheduling on uniform multiprocessors, in: RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01), IEEE Computer Society, Washington, DC, USA, 2001, p. 183.

[24] U. Farooq, S. Majumdar, E. W. Parsons, Engineering grid applications and middleware for high performance, in: WOSP '07: Proceedings of the 6th international workshop on Software and performance, ACM, New York, NY, USA, 2007, pp. 141–152. doi:http://doi.acm.org/10.1145/1216993.1217019.

[25] U. Farooq, S. Majumdar, E. W. Parsons, A framework to achieve guaranteed QoS for applications and high system performance in multi-institutional grid computing, in: ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing, IEEE Computer Society, Washington, DC, USA, 2006, pp. 373–380. doi:http://dx.doi.org/10.1109/ICPP.2006.8.

[26] I. Foster, A. Roy, A quality of service architecture that combines resource reservation and application adaptation, in: 8th Int. Workshop on Quality of Service, 2000, pp. 181–188.

[27] K. L. Mills, C. Dabrowski, Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace?, J. Grid Comput. 6 (3) (2008) 291–311.

[28] J. Yu, R. Buyya, C. Tham, Cost-based Scheduling of Scientific Workflow Applications on Utility Grids, Proceedings of the 1st International Conference on e-Science and Grid Computing (e-Science 2005) (2005).

[29] D. E. Irwin, L. E. Grit, J. S. Chase, Balancing risk and reward in a market-based task service, in: Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 160–169. doi:http://dx.doi.org/10.1109/HPDC.2004.5.

[30] B. N. Chun, D. E. Culler, User-centric performance analysis of market-based cluster batch schedulers, in: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02), IEEE Computer Society, Washington, DC, USA, 2002, p. 30.

[31] M. A. Rappa, The utility business model and the future of computing services, IBM Systems Journal 43 (1).

[32] R. Buyya, D. Abramson, J. Giddy, H. Stockinger, Economic models for resource management and scheduling in Grid computing, Concurrency and Computation: Practice and Experience 14 (13-15) (2002).

[33] Amazon's prices.
URL `http://aws.amazon.com/ec2/#pricing`

[34] W. Vogels, Introducing Amazon EC2 Reserved Instances – A way to further reduce IT costs.
URL `http://www.allthingsdistributed.com/2009/03/amazon_ec2_reserved_instances.html`

[35] Grid Workloads Archive.
URL `http://gwa.ewi.tudelft.nl/`

[36] Nordugrid.
URL `http://www.nordugrid.org/`

[37] Y.-S. Kee, H. Casanova, A. A. Chien, Realistic modeling and synthesis of resources for computational grids, in: SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing, IEEE Computer Society, Washington, DC, USA, 2004, p. 54. doi:http://dx.doi.org/10.1109/SC.2004.49.