On the Planning of a Hybrid IT Infrastructure

Paulo Ditarso, Flavio Figueiredo, David Maia, Francisco Brasileiro, Álvaro Coêlho
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
Emails: {pmaciel, flaviov, davidcmm, fubica, degas}@lsd.ufcg.edu.br

***Abstract*—With the emergence of utility computing and the continuous search for reducing the cost of running Information Technology (IT) infrastructures, we will soon experience an important change on the way these infrastructures are assembled, configured and managed. In this paper we consider the problem of managing a hybrid high-performance computing infrastructure whose processing elements comprise in-house dedicated machines, a utility computing service provider, and idle machines from a best-effort peer-to-peer grid. This infrastructure supports the execution of both best-effort and real-time applications. Real-time applications use primarily computing power from the in-house machines and any processing power that can be attained from the best-effort grid. Extra capacity required to meet deadlines is purchased from the utility computing service provider. This extra capacity is reserved for future use through short term contracts which are negotiated with no human intervention. We take a business-driven approach for the management of this hybrid infrastructure and propose heuristics that can be used by a contract planner agent to reduce the cost of running the applications at the same time that guarantees that deadlines are met. In particular, we show that constructing an estimation for the behavior of the grid is essential for making contracts that lead to high efficiency in the use of the hybrid infrastructure.**

I. Introduction

Utility computing is becoming reality with several companies offering solutions for its implementation and others already providing computation on demand based on these solutions. Those promoting this kind of solution place as one of their main selling arguments the possibility of providing their customers with substantial reductions on the total cost of ownership (TCO) of their Information Technology (IT) infrastructures.

Although economical advantages certainly play an important role on the adoption of a technology, other factors also have their saying. For instance, it is likely that the migration of services that are supported by in-house dedicated IT infrastructure to one offered by an external utility computing service provider will face strong resistance from the internal IT management staff. Also, it may well be the case that running certain types of services on a utility computing infrastructure will not yield the cost reduction sought. More importantly, for some services it may not be desirable to have them executing on a third-party infrastructure (e.g. strategic, sensitive and critical services). Finally, having some in-house capacity may minimize possible undesirable effects of price fluctuations due to transient instabilities in the utility computing market.

On the other hand, the market-based utility computing model is not the only alternative to reduce TCO. Among other solutions proposed, peer-to-peer grid computing has been suggested as a way to enable a simpler economy for the trading of idle cycles [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 [2]. On the other hand, in these situations sharing systems may be efficient, as they can use information which is loosely structured and therefore easier to obtain, they can make use of social mechanisms for monitoring and enforcement, and they have lower marginal transaction costs [2]. However, unlike the solutions based on markets, these solutions generally give no guarantees on the quality of the service provided. Nevertheless, they have been successfully used to increase the cost effectiveness of IT infrastructures in a number of settings [3].

Given all that, in the near future we envision that many IT infrastructures will comprise a mix of computing power provided by in-house dedicated infrastructure and computing power provided from external utility computing service providers. Moreover, both kinds of approaches will provide different guarantees, varying from those with well defined quality of service to those provided in a best-effort basis.

This conjecture is also supported by our own experience. In our effort to disseminate the use of the OurGrid middleware we have fostered the creation of the OurGrid Community (http://www.ourgrid.org/), an open peer-to-peer grid that is in production since December, 2004. (See http://status.ourgrid.org/ for an up to date snapshot of the running system.) The OurGrid Community has been used in a variety of areas, from engineering [4] to bioinformatics [5], from computer science [6] to financial applications [7]. In particular, OurGrid supports the cooperative work of a community of meteorologists and hydrologists, both in the academia and in the government [4], [8]. Some members of this community provide daily weather forecasts as a public service (see, for instance, http://www.cptec.inpe.br/). Moreover, the capacity required at critical times (when real-time applications are run) is normally much larger than that required at other times. So, if for one side over provisioning the IT infrastructure of these public agencies to cope with the high demand at critical times

978-1-4244-2066-7/08/$25.00 ©2008 IEEE

is not cost effective, on the other side, relying on the best-effort guarantees of OurGrid is not acceptable. In this setting, it is likely that extra computing power required at critical times could be acquired on demand and with the required quality of service at a smaller cost than over provisioning the in-house dedicated infrastructure. Moreover, the amount of extra computing power required could be reduced if the idle cycles of the dedicated infrastructure could be traded in a peer-to-peer grid. So, in this setting, a hybrid IT infrastructure is highly desirable.

Since IT management has been mostly studied considering a uniform way of providing IT services, interesting research questions arise from the new environment sketched above. In this paper we address the problem of how to improve the efficiency of running a hybrid IT infrastructure that supports the execution of real-time applications. In this infrastructure, any idle in-house capacity is donated to a best-effort grid and this allows donated cycles to be claimed back from the grid in the future, although with no guarantees of when they will be made available. Real-time applications are processed using the full local capacity, and any capacity that can be retrieved from the grid. Any extra capacity required to fulfill the workload by its deadline can be acquired on-demand from a utility computing service provider. This provider offers a contract interface that is used by an agent to automatically negotiate the amount and cost of the computing power used from the provider. The contract establishes guarantees and fees to be charged for both reservation and effective use of computing power. We assume that the cost of using the utility computing provider does not change over the relatively short period of time that a critical application run, but the cost of reserving computing power increases with the urgency: the sooner it is reserved, the cheaper is the reservation fee. The difficulty in establishing the appropriate contracts lies in the uncertainty on the amount of computing power that can be timely received from the best-effort peer-to-peer grid. The sooner a contract is established, the cheaper its cost, but the less accurate is the information on the amount of extra computing power required. We discuss several heuristics that can be used to drive the planning of contracts. To the best of our knowledge, this is the first work that explores this hybrid IT infrastructure.

The rest of the paper is organized as follows. In Section II we discuss related work. Section III gives a formal description of the problem. Since our solution follows a business-driven approach for management [9], in Section IV we describe a utility function for the execution of a real-time application over the hybrid IT infrastructure. The contract planning heuristics are presented in Section V and evaluated in Section VI. Finally, Section VII closes the paper with our concluding remarks and perspectives of future work.

II. Related Work

Similarities can be identified among our work, [10] and [11], as well as some points in which they may complement each other. The work in [10] focuses on the service provider operation, whilst [11] focuses on 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 as well as a service provider.

Popovici et al. [10] propose an economics-oriented approach for a service provider to follow, when choosing which customer's requests to accept. The difficulty is in selecting the requests, given the uncertainty on the availability of a resource provider service. This approach helps in defining admission control and scheduling algorithms that maximize profit. They extend the work presented in [12] and [13], by considering a service provider that offers job-based services to its clients and rents resources from a resource provider. The proposed algorithms take into account the uncertainty of the availability of resources from resource providers. The authors use risk-aware heuristics to maximize the utility obtained. In our work we consider a contract planner to a hybrid IT infrastructure, and the uncertainty comes from the best-effort nature of the grid behavior.

Buyya et al. [11] propose a scheduling algorithm that minimizes execution costs of workflows in a grid while meeting user's specified deadlines. The unreliability of the grid introduces uncertainty. This, in turn, is dealt with by re-scheduling tasks that fail in other grid resources available. Availability prediction and contract cost information of grid resources are used to select the most appropriate contracts and minimize cost. Our work uses a similar strategy, but applied to a different context.

As the necessity for computing services increases in the daily life of most organizations, and considering that utility computing is becoming reality, utility provider business models are extremely necessary. In [14], the author presents a general overview of how the business model for utility computing should look like, taking account characteristics such as necessity, reliability, usability, scalability etc. This approach is similar to the ones used for other utilities as, for example, water, telephone, Internet access, and electricity. The pricing model for the utility computing service provider that we use in this paper is also inspired on the pricing model of other utilities.

In [15] the authors provide some economic models for setting the prices of services based on *supply-demand*, such as commodity market, posted price and auction models. Based on the various possible models existing, a system architecture and policies for resource management in Grid infrastructure are described. In our work, we assume a generic pricing model composed of a reservation and a consumption fee, into which different provider's profile can be mapped.

Business-Driven IT Management (BDIM) [9] is a new area of research that takes a business perspective to the management of IT services. In this context, we propose planning algorithms for a hybrid IT infrastructure in order to maximize the utility obtained when running a real-time application. The utility metric can be easily linked with business metrics, such as profit, revenue or financial loss. To the best of our knowledge, no previous work on grid management, including ours in the context of the OurGrid system, considers business

objectives to drive management operations in a hybrid IT infrastructure.

III. Problem Statement

In this section we present a formal description of the problem. Firstly, we describe the considered application; secondly, we present the components of the hybrid IT infrastructure; and finally, the contract planning problem is stated.

A. The Application

For the sake of simplicity, we assume that during a time period Δt_{period} (typically a day), there is a single critical application to be executed. This application is characterized by a tuple $\mathcal{A} = \langle w, t_{ready}, t_{deadline} \rangle$, where $\mathcal{A}.w$ is an indication of the application's processing demand expressed as the number of cycles required to run the application in a reference machine; $\mathcal{A}.t_{ready}$ is the real time when the application is ready for execution (e.g. the time when data required to run the application was made available); and $\mathcal{A}.t_{deadline}$ is the real time by when the execution of the application must be finished. We will refer to the difference between the time the application must be completed and the time it is ready to start its execution as $\Delta t_{running} = \mathcal{A}.t_{deadline} - \mathcal{A}.t_{ready}$. Figure 1 depicts the above description.

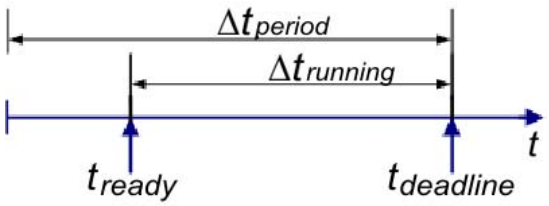


Fig. 1. The application timeline

B. The Hybrid IT Infrastructure

We consider a hybrid IT infrastructure which at any instant of time t is able to provide computing cycles that are equivalent to $d(t) + g(t) + u(t)$ computing cycles of a reference machine, where $d(t)$, $g(t)$ and $u(t)$ are the amount of equivalent cycles provided by, respectively, the in-house dedicated machines, the peer-to-peer grid and the utility computing service provider.

We assume that $d(t)$ is constant over time and is a function of the local capacity installed. On the other hand, $g(t)$ and $u(t)$ varies over time. For any t, such that $t < \mathcal{A}.t_{ready}$, we have $g(t) = u(t) = 0$, since there is no critical computation to be executed and therefore no need to seek for resources provided by the grid or the utility provider. During this time any idle cycles from the $d(t)$ cycles available are offered to the grid. Again, for the sake of simplicity we assume that for any t, $t < \mathcal{A}.t_{ready}$, all $d(t)$ cycles are idle, and are, therefore, donated to the grid. For all t, such that $\mathcal{A}.t_{ready} \leq t \leq \mathcal{A}.t_{deadline}$, both $g(t)$ and $u(t)$ may be greater than zero. The former will depend on the quality of service delivered by the grid at t, while the later will depend on the contracts that have been established with the utility provider prior to t.

In the absence of an established pricing model for the utility provider, we assume one that is a simplification of the pricing model used in the electricity market for large consumers. It is composed of two fees: a reservation fee and a consumption fee. To model the reservation cost, it is reasonable to think that the antecedence of a contract is a critical factor to price the resources, i.e. the earlier the contract is established, the cheaper will be the reservation cost. Moreover, the reservation cost is directly proportional to the amount of cycles reserved. Given that contracts are short term, we assume that the consumption fee for any contract is a fixed value that is charged for each cycle effectively consumed. It is important to note that the amount of consumed cycles from the provider can be less than the amount of reserved cycles, but never more than that.

A contract between the client and the utility provider is a tuple $\mathcal{K} = \langle init, rate, duration, price \rangle$ that establishes the instant of time after which the cycles will start to be consumed ($\mathcal{K}.init$), the maximum rate of cycle consumption per unit of time ($\mathcal{K}.rate$) and the duration of the contract ($\mathcal{K}.duration$). It also establishes the fees ($\mathcal{K}.price$) for the contract. When a contract is established the utility provider agrees in fulfilling it and the client agrees in paying the agreed fees. We assume that all contracts attempted are successfully established.

C. Planning and Scheduling

In the proposed environment the scheduling of the application is preceded by the acquisition of computing resources, known as *planning* [16]. For simplicity, we assume that the application has a workload that can be divided into very fine grains, so that at any instant of time t it is possible, if needed, to schedule work for the in-house dedicated infrastructure, the utility provider and the peer-to-peer grid that will consume, respectively, $d(t)$, $u(t)$ and $g(t)$ cycles. So, scheduling the application is extremely simple. Starting from t_{ready}, the scheduler simply uses all available cycles until the execution of the application is completed, giving preference to use first all in-house cycles available, then any cycles retrieved from the grid and, finally, any cycles that can be used from the utility provider.

Therefore, the actual problem is to execute the planning, i.e. when and how much extra resources needs to be acquired from the utility provider. In other words, we need to devise the algorithm that an agent must run to automatically define which contracts should be established with the utility provider, so that the application execution can be completed by its deadline at as small a cost as possible. More formally, we define a plan as a tuple $\mathcal{P} = \langle \mathcal{C}, \mathcal{U} \rangle$, where $\mathcal{C}$ is the set of contracts established and $\mathcal{U}$ is the total utility delivered by the plan. Our aim is to provide a mechanism that is able to produce efficient plans, i.e. plans that deliver as much utility as possible.

In the next section we model the utility obtained by the real-time application that is executed over the hybrid IT infrastructure.

IV. Utility Function of the Hybrid Infrastructure

In our model, we consider that each cycle that is used to process the application's workload yields a unit of utility. Furthermore, each cycle, whether it has been used to compute the workload or not, has an associated cost. Therefore, the total utility of the hybrid IT infrastructure for a given plan $\mathcal{P}$ and an application $\mathcal{A}$ is the difference between the application's workload and the sum of the cost of maintaining the infrastructure during Δt_{period} plus the cost of the cycles used to run A within $\Delta t_{running}$. This is expressed as:

$$\mathcal{P}.\mathcal{U} = A.w - (\gamma_d + \gamma_g + \gamma_p),$$

where γ_d is the cost of maintaining the dedicated in-house infrastructure, γ_g is the cost of using the grid, and γ_p is the cost of reserving and using cycles from the utility computing service provider.

The cost of the dedicated infrastructure is given by:

$$\gamma_d = d(t) \cdot \Delta t_{period} \cdot v_d,$$

where v_d, $0 < v_d < 1$, is the fixed cost of each cycle available on the dedicated infrastructure.

There is a small cost v_g of donating an idle cycle to the grid. Assuming that $v_g \ll v_d$, hereafter we do not consider the cost of donating idle cycles to the grid. Moreover, there is no cost for the cycles that are claimed from the grid, since they are donated cycles. Thus, $\gamma_g = 0$.

To compose the cost of the computational power acquired from the service provider, we define β as the ratio between the reservation and the consumption fees, i.e., $\beta = 3/4$ means that 75% of the cost is due to the reservation fee, and the other 25% is due to the consumption fee. We believe that β should be at least 0.5 to mitigate loss of revenues when cycles are reserved and not consumed by the clients. Moreover, we define a factor φ to reflect the variation of the reservation cost with time. Therefore, the cost incurred from the cycles reserved and used from the utility computing service provider is given by:

$$\gamma_p = \sum_{\forall \mathcal{K} \in \mathcal{P}.\mathcal{C}} v_u \cdot \{\beta \cdot \mathcal{K}.duration \cdot \mathcal{K}.rate \cdot \varphi + (1-\beta) \cdot c^{\mathcal{K}}_{used}\},$$

where v_u is the provider's cost per cycle, and $c^{\mathcal{K}}_{used}$ is the number of cycles effectively used under contract $\mathcal{K}$.

At this point, we can define two different types of "bad contracts". The first one is when the amount of consumed cycles is smaller than the amount reserved, and the consumer has paid for the reservation of more cycles than needed. The other one is when the amount of cycles reserved is smaller than the total number of cycles needed, and a new contract should be made to meet the deadline. In this case, the new contract will be more expensive due to the variation of the reservation cost with time. In other words, the perfect contract is one that is established as soon as possible and in which the amount of reserved cycles is exactly the amount of cycles needed from the service provider.

V. Planning Algorithms

We assume that λ, $0 < \lambda < 1$, is the capacity ratio of the hybrid infrastructure, and expresses the ratio between $d(t)$ and the fraction of the workload that needs to be processed at each instant of time t, i.e. $\lambda = 0.5$ means that the in-house capacity can compute 50% of the application's workload[1]. Thus, the number of extra cycles (c_{extra}) that are required from the grid and the utility computing service provider is given by:

$$c_{extra} = \left(\frac{1}{\lambda} - 1\right) \cdot d(t) \cdot \Delta t_{running}.$$

As mentioned before, the best contract can only be established if the amount of cycles that will be received from the grid during the interval $\Delta t_{running}$ is known. The number of cycles received from the grid will depend on two factors. Firstly, the amount of "credit" that the client has in the grid, i.e. the amount of idle cycles that have been donated by the client to the grid and have not yet been reclaimed. Note that idle cycles from the client are only consumed by the grid if there is a demand for them when they are available. Moreover, since the grid only guarantees that cycles will eventually be payed back [17], within the critical window $\Delta t_{running}$ there is a possibility that not all of the idle cycles donated will be reclaimed. On the other hand, if the contention for idle resources is low, and there are many participants with idle resources, then there is a possibility that more cycles than those previously donated will be reclaimed. We model this by considering a grid quality of service parameter g_f that gives an indication of the proportion of idle cycles donated that can be reclaimed from the grid during $\Delta t_{running}$. Thus, the number of cycles received from the grid during $\Delta t_{running}$ is expressed in the following way:

$$c_{grid} = (\Delta t_{period} - \Delta t_{running}) \cdot d(t) \cdot g_f.$$

If $c_{extra} \leq c_{grid}$, then there is no need to acquire cycles at the utility provider. Otherwise, exactly $c_{extra} - c_{grid}$ cycles must be reserved and used from the utility computing service provider.

However, due to the grid's best-effort nature, there is no precise method of knowing how many cycles will be received during a certain time interval, i.e. g_f can at best be estimated. In this section, we propose planning algorithms that can be used in order to maximize the utility of running a hybrid infrastructure considering the uncertainty of the grid. Since the cost of the in-house capacity is fixed, utility can only be maximized by minimizing the cost of contracts made with the utility computing service provider. Ideally, a planner can make use of estimates of the amount of cycles that will be received from the grid. For example, this estimation may be achieved by analytical models of the grid's behavior and validated by historical-analysis of the grid. Considering that this information is known, a planning heuristic can be applied

[1]Note that when $\lambda = 0$, there are no in-house resources, and when $\lambda = 1$ there is an over provision of resources. Neither of these scenarios are of interest in our study.

to efficiently establish the appropriate contracts for the hybrid infrastructure.

For the sake of argumentation we will only consider a subset of the possible heuristics that make at most two contracts within each Δt_{period} time intervals. These contracts are established at well-defined times, namely: at the beginning of the time line (as soon as possible); or at the last time moment just before the deadline in which a contract can be made - let us denote this time instant $t^{-}_{deadline}$. The last contract always reserves all remaining extra cycles required at the time the contract is established. Since any contract that is attempted is successfully established, the resulting plans guarantee that the application always meet its deadline. The basic functioning of the planner is described in Algorithm 1.

Algorithm 1: Planner's Algorithm

```
begin
    At time = 0 do
    begin
        reserve cycles according to Heuristic
    end

    At time = t^-_deadline do
    begin
        reserve any remaining extra cycles needed
    end
end
```

To evaluate the execution of the application on the hybrid infrastructure, the heuristics described below were developed. A comparison of them is made in the next section.

A. Omniscient Heuristic

This heuristic produces an optimal plan. We assume that it has total knowledge of the grid behavior (i.e. it knows the exact value of g_f), and makes a contract with the precise amount of extra cycles needed from the service provider to compute the workload. As a result of this, it can reach the minimal cost of executing the application by taking benefit of the peer-to-peer grid. Thus, this heuristic always achieves the maximum utility of the hybrid infrastructure. We note that, as it uses the exact amount of received cycles from the grid to make the first contract, it never needs to make the second contract.

B. Oblivious Heuristic

This heuristic is oblivious of the existence of the grid. It always establishes a first contract for the whole extra cycles needed to compute the workload. Obviously, in this case there is also no need of making a new contract in a future time. This heuristic is a kind of lower bound[2] on the utility that can be attained from running the application on the hybrid infrastructure.

[2]This heuristic is not exactly a lower bound because it is always possible that a heuristic takes such bad decisions that its utility is even worse than that attained by the Oblivious one.

C. Averse Heuristic

This heuristic is similar to the Oblivious one, in the sense that it is completely averse to the risk of trusting in the best-effort grid. Therefore, it also reserves all extra cycles needed to complete the workload at the first contract. However, unlike the previous heuristic, grid cycles attained from the grid are used to run the application, possibly reducing the amount of cycles that are effectively consumed from the utility provider.

D. Greedy heuristic

This heuristic assumes full risk of trusting in the best-effort grid, and makes a contract with the utility service provider only in the last time instant, if needed. If the amount of received cycles from the grid is greater than the amount of extra cycles needed, no contract is made with the service provider.

E. Average heuristic

This heuristic is a compromise between the Averse and the Greedy heuristics. It makes a contract assuming that half of the extra amount of cycles needed to complete the workload ($c_{extra}/2$) will be reclaimed from the grid. Note that in this case there is a possibility of making two contracts in order to meet the deadline; this happens when the amount of received cycles is less than 50% of the extra amount needed.

F. Grid-Aware heuristic

This heuristic has some knowledge about the grid behavior. It does not know the exact value of g_f but it is able to build the probability distribution of g_f. This way, it assumes that the amount of cycles reclaimed from the grid is the mean of the probability distribution that represents the grid behavior. Note that, as happens with the *Average* heuristic, the establishment of two contracts may be necessary in order to meet the deadline.

G. Price&Grid-Aware heuristic

This heuristic has full knowledge about the provider's pricing model (β, φ and v_u) and, like the previous one, it also has knowledge about the grid's probability distribution behavior. Given this information, it assumes that the value of received cycles from the grid is one that equalizes the average penalty incurred by the two types of bad contracts, i.e. it minimizes the average penalty incurred due to a wrong estimation of g_f.

In summary the first two heuristics (*Omniscient* and *Oblivious*) produce useful benchmarks for the utility that can be achieved on the hybrid infrastructure. The next three heuristics (*Averse*, *Greedy* and *Average*) represent different degrees of trust on the best-effort grid. Finally, the last two heuristics (*Grid-Aware* and *Price&Grid-Aware*) utilize extra information about the grid and the provider's pricing model to help in better planning the establishment of contracts.

VI. Heuristics Evaluation

We developed a simulator to evaluate the heuristics described above. In this section we present the scenarios that we have investigated in our experiments and the results we have obtained from the simulation runs. To do such, we reference the parameters defined on the model described in Sections IV and V, and also, we define some new that are used in the investigation.

A. Scenarios Description

Based on the SegHidro experience [4], we assume a workload of an application running in a twenty-machines cluster during approximately 12 hours, on a daily basis. We define an efficiency coefficient e_p to establish the provider's cost per cycle. This coefficient relates the costs of running the utility provider and that of running the in-house infrastructure. For example, if we set $e_p = 0.5$, then we are considering that the provider's infrastructure is run 50% more efficiently than the in-house infrastructure. So, we consider $v_u = (1 - e_p) \cdot v_d$, with $0 \leq e_p \leq 1$. We have considered scenarios with $e_p \in \{0.5, 0.8\}$, which seemed reasonable values for the utility provider's efficiency.

In order to evaluate different provider's profiles, we vary the reservation/consumption ratio β for values in $\{1/2, 2/3, 3/4\}$. The variation on the utility provider's reservation cost with time φ is instantiated as follows. In the first contract we always have $\varphi = 1$. We then set φ in the second contract to be either 2.5 (when $e_p = 0.5$) or 7 (when $e_p = 0.8$). This allowed us to investigate the behavior of the heuristics in a scenario in which the cost of the second contract is moderately higher than the first, and in another in which this cost is substantially higher. Moreover, by matching different values of e_p and φ, we have, in both cases and as we vary β, interesting relations between the cost per used cycle for the provider's most expensive contract ($v_u \cdot [\beta \cdot \varphi + (1 - \beta)]$) and the cost per cycle of the in-house infrastructure (v_d). The first situation ($\beta = 1/2$) represents a provider whose cost per used cycle of the most expensive contract is greater than the in-house cycle cost. The second situation ($\beta = 2/3$) represents the case when both provider and in-house infrastructures have the same cost. The last one ($\beta = 3/4$) represents the situation when the provider's most expensive contract is lower than the in-house cost.

Andrade et al. [17] have shown that the incentive mechanism used by OurGrid leads most of the peers in the system to experience values of g_f that are close to 1, with an average smaller than 1 depending on the amount of free-riding in the system. However, that work did not consider a restricted window of time within which the donated cycles are reclaimed. Thus, in this more demanding scenario that we are studying, it is reasonable to consider that the grid factor assumes values between 0 and 1.1. So, we define three grid profiles according to its quality of service: a *bad* quality grid, a *medium* quality grid, and a *good* quality grid. To represent these behaviors the grid is modeled by a multi-modal distribution.

The *good* quality grid is one in which, with a probability of 90%, g_f is chosen from a uniform distribution in the interval $[0.8, 1.1]$, and with a probability of 10% g_f is chosen from a uniform distribution in the interval $[0, 0.8)$. The average value for the more frequent interval is less than 1 to account for some free-riding. Analogously, the *medium* quality grid is one that, with a probability of 90%, g_f is chosen from a uniform distribution in the interval $[0.55, 0.85]$, while with probability of 10% it is chosen from a uniform distribution in the intervals $[0, 0.55)$ and $(0.85, 1.1]$. Finally, the *bad* quality grid is one in which g_f has a probability of 90% of being chosen from a uniform distribution in the interval $[0, 0.3]$, and a 10% probability of being chosen from a uniform distribution in the interval $(0.3, 1.1]$.

To measure the performance of the proposed heuristics we compare them with the performance of the Omniscient and Oblivious heuristics. We define the *efficiency* of a heuristic H as follows:

$$1 - \frac{\overline{U}_{Omniscient} - \overline{U}_{H}}{\overline{U}_{Omniscient} - \overline{U}_{Oblivious}},$$

where $\overline{U}_{Omniscient}$, $\overline{U}_{H}$ and $\overline{U}_{Oblivious}$ are, respectively, the average utility attained by the Omniscient, H and Oblivious heuristics, in all scenarios considered.

With this, we can compare the performance of the heuristics related to the maximum reachable utility. We note that the efficiency is not defined for the cases that the average utility of the Omniscient heuristic is equal to that of the Oblivious one. This situation does not arise in any of the scenarios that we have simulated, since the probability of not receiving any cycles at all from the grid is very low. Note also, that 1 is an upper bound for the efficiency, but there is no lower bound for it, with negative efficiency values being possible. This occurs because the Omniscient heuristic always achieve the maximum utility, however, as pointed out before, the worst result is not always achieved by the Oblivious heuristic.

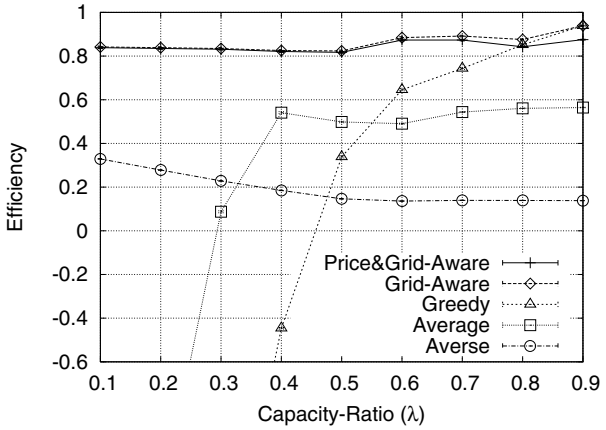
Since we assume that each cycle that is processed from the workload yields one unit of utility, the value of v_d can be inferred from the perceived profit of running the application. We note that for the particular example we are targeting, evaluating this value is not a trivial task, since many subjective criteria should be taken into account to measure it. Therefore, we have assumed $v_d \in \{0.1, 0.2, \ldots, 0.9\}$. However, since the simulations with different values of v_d did not show efficiency results substantially different, in the following we will only report the values for $v_d = 0.5$.

For all graphs presented in this work we plot the results of the efficiency versus the capacity ratio (λ). We consider that $\lambda \in \{0.1, 0.2, \ldots, 0.9\}$. Moreover, each point in the graphs represents the average of 30,000 runs, at the 95% confidence level with a negligible error.

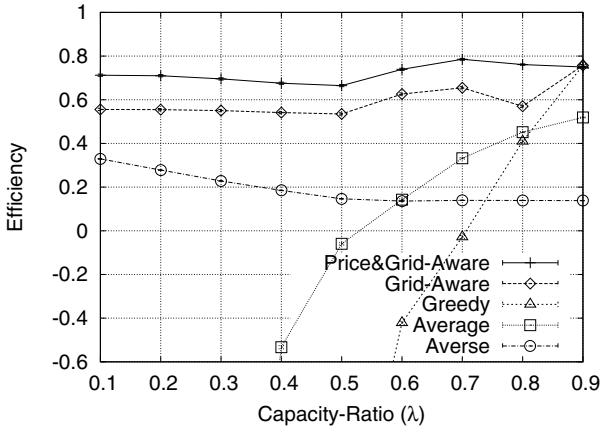
B. Performance Analysis

Figure 2 represents the average efficiency obtained by the heuristics on all scenarios simulated for $\varphi = 2.5$ and $e_p = 0.5$ (Figure 2(a)) and $\varphi = 7$ and $e_p = 0.8$ (Figure 2(b)). As expected, both Grid-Aware and Price&Grid-Aware heuristics give the best results, since they take into account

partial knowledge about the grid behavior and the provider's pricing model. These heuristics have a very stable behavior and are always able to attain more than 80% of the achievable efficiency when $\varphi = 2.5$. When φ increases, the efficiency of these heuristics decrease, due to the substantial increase in the cost of the second contract, specially for the Grid-Aware heuristic which does not consider the provider's pricing model. From the results we can also see that the heuristics that do not have knowledge about the grid perform, in most cases, very badly. The Greedy heuristic achieves very low efficiency with lower values of λ, and improves when λ increases. This occurs because the in-house infrastructure gets larger as λ increases, thus the number of extra cycles required diminishes at the same time that more idle cycles are donated to the grid, increasing the probability of having the few cycles needed reclaimed back on time. The Average heuristic has a similar behavior but, not so bad when λ is low and not so good when λ is high. As expected, both heuristics worsen when φ increases. The Averse heuristic has a more stable behavior and yields a very low efficiency in all scenarios investigated. This is because it does not take into account the grid and reserves the whole amount of extra cycles needed.



(a) $\varphi = 2.5$ and $e_p = 0.5$



(b) $\varphi = 7$ and $e_p = 0.8$

Fig. 2. Average efficiency versus capacity ratio (λ).

We have also evaluated the impact of the grid quality of service on the efficiency achieved. Figure 3 presents the average efficiency obtained in each different scenario of the grid's quality for $\varphi = 2.5$ and $e_p = 0.5$. Not surprisingly, the heuristic Averse has little impact on its efficiency in the different scenarios. The other heuristics perform better when the grid quality is at least medium, while the Average and Greedy heuristics are more sensitive to the grid quality than the others. This occurs because these heuristics take more risk than the others[3].

In summary, our results show that different planning of contracts yields quite different efficiencies. Moreover, a blind approach to the definition of such a plan may lead to very poor efficiency, in some cases even leading to negative utilities. So, it is important to use at least knowledge about the grid behavior when deciding which contracts to establish.

VII. Conclusion

In this paper we have defined a hybrid IT infrastructure that we consider will be a reality in a near future. The processing power of this infrastructure is provided by in-house dedicated resources, as well as by on-demand processing power purchased from a utility computing service provider, and idle cycles traded on a peer-to-peer grid.

We have described planning heuristics that can be used by a contract planner agent to reduce the cost of running real-time applications in this hybrid environment, at the same time that guarantees that deadlines are met. We also derived a utility provider pricing model and a utility function to evaluate the total utility obtained by executing an application in this hybrid infrastructure.

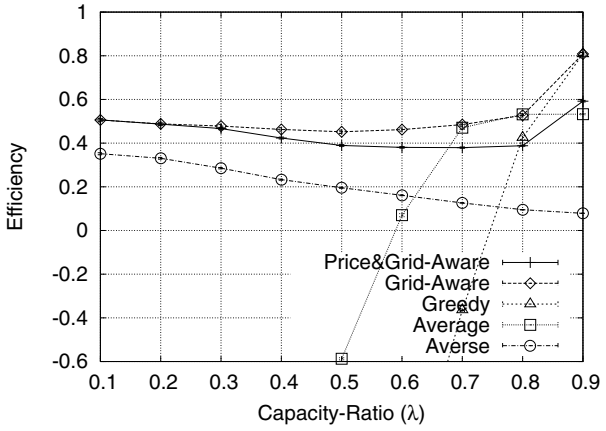
By the obtained results, we can conclude that the use of this hybrid IT infrastructure can take advantage from the low cost of the grid to improve the efficiency attained by running the application. However, it is not a good approach to fully trust the grid, except in the situations near to over provisioning of the in-house dedicated infrastructure. Furthermore, the advantage of using a best-effort grid can be maximized by using partial knowledge about the grid behavior. In particular, we show that constructing an estimation for the behavior of the grid is essential for making contracts that lead to high efficiency in the use of the hybrid infrastructure.

As future work, we intend to develop a mechanism that is able to dynamically estimate the amount of cycles that will be received from the grid. We will then implement the planner agent and evaluate its functioning in a real setting. We also believe that BDIM aspects other than planning can be modeled in this hybrid IT infrastructure and we intend to broaden our investigation in this direction.

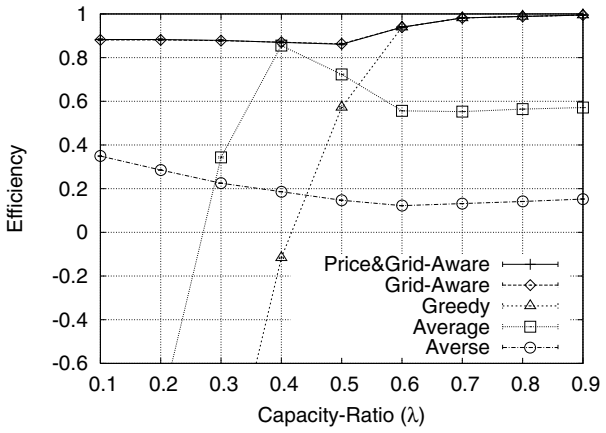
Acknowledgment

This work has been developed in collaboration with HP Brazil R&D. We would like to thank John Wilkes and Dejan

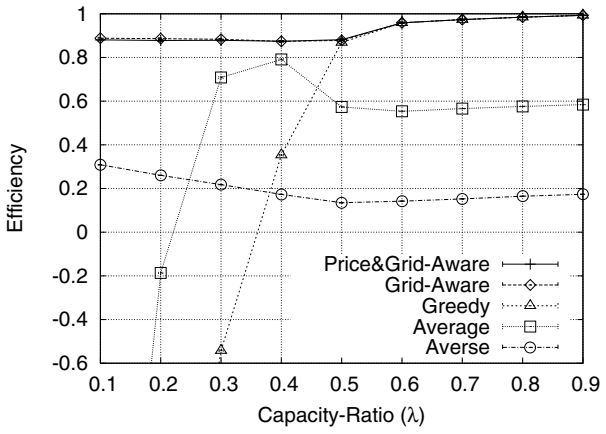
[3]The findings for the case in which $\varphi = 7$ and $e_p = 0.8$ are the same and are not shown.



(a) Bad quality grid



(b) Medium quality grid



(c) Good quality grid

Fig. 3. Efficiency versus capacity ratio (λ) for different grid's behavior, with $\varphi = 2.5$ and $e_p = 0.5$.

Milojicic for insightful suggestions in early versions of this paper.

REFERENCES

[1] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray, "Labs of the world, unite!!!" *Journal of Grid Computing*, 2006, available as technical report at http://www.ourgrid.org/twiki-public/pub/OG/OurPublications.

[2] B. Yochai, "Sharing nicely: On shareable goods and the emergence of sharing as a modality of economic production," *The Yale Law Journal*, vol. 114, pp. 273–358, 2004. [Online]. Available: http://www.yalelawjournal.org/archive_abstract.asp?id=94

[3] F. Brasileiro, E. Araújo, W. Voorsluys, M. Oliveira, and 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, May 2007.

[4] E. C. Araujo, W. Cirne, G. Mendes, N. Oliveira, E. P. Souza, C. Galvao, and 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, and P. G. Pascutti, "The biopauã project: A portal for molecular dynamics using grid environment," *Brazilian Symposium on Bioinformatics*, 2005.

[6] A. N. Duarte, W. Cirne, F. Brasileiro, and 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, and 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, and E. Cavalcanti, "Fostering collaboration to better manage water resources," *Proceedings of GCE 2005: Workshop on Grid Computing Portals*, November 2005.

[9] J. Sauve, A. Moura, M. Sampaio, J. Jornada, and E. Radziuk, "An Introductory Overview and Survey of Business-Driven IT Management," *Business-Driven IT Management, 2006. BDIM'06. The First IEEE/IFIP International Workshop on*, pp. 1–10, 2006.

[10] F. I. Popovici and J. Wilkes, "Profitable services in an uncertain world," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC'05)*. Washington, DC, USA: IEEE Computer Society, 2005, p. 36.

[11] J. Yu, R. Buyya, and 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)*, pp. 140–147.

[12] B. N. Chun and 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)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 30.

[13] D. E. Irwin, L. E. Grit, and 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)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 160–169.

[14] M. A. Rappa, "The utility business model and the future of computing services," *IBM Systems Journal*, vol. 43, no. 1, 2004.

[15] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[16] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[17] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray, "Automatic grid assembly by promoting collaboration in peer-to-peer grids," *J. Parallel Distrib. Comput.*, vol. 67, no. 8, pp. 957–966, 2007.