

1-1-2009

# Management of Cloud Infrastructures: Policy-Based Revenue Optimization

Tim Pueschel

*Albert Ludwigs Universitat Freiburg, tim.pueschel@is.uni-freiburg.de*

Dirk Neumann

*Albert Ludwigs Universitat Freiburg, dirk.neumann@is.uni-freiburg.de*

Follow this and additional works at: <http://aisel.aisnet.org/icis2009>

---

## Recommended Citation

Pueschel, Tim and Neumann, Dirk, "Management of Cloud Infrastructures: Policy-Based Revenue Optimization" (2009). *ICIS 2009 Proceedings*. Paper 178.

<http://aisel.aisnet.org/icis2009/178>

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 2009 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact [elibrary@aisnet.org](mailto:elibrary@aisnet.org).

# MANAGEMENT OF CLOUD INFRASTRUCTURES: POLICY-BASED REVENUE OPTIMIZATION

*Completed Research Paper*

**Tim Püschel**

Chair for Information Systems Research  
Albert-Ludwigs-Universität Freiburg  
Platz der Alten Synagoge,  
79085 Freiburg, Germany  
tim.pueschel@is.uni-freiburg.de

**Dirk Neumann**

Chair for Information Systems Research  
Albert-Ludwigs-Universität Freiburg  
Platz der Alten Synagoge,  
79085 Freiburg, Germany  
dirk.neumann@is.uni-freiburg.de

## **Abstract**

*Competition on global markets forces many enterprises to make use of new applications, reduce process times and at the same time cut the costs of their IT-infrastructure. To achieve this, it is necessary to maintain a high degree of flexibility with respect to the IT-infrastructure. Facing this challenge the idea of Cloud computing has been gaining interest lately. Cloud services can be accessed on demand without knowledge of the underlying infrastructure and have already succeeded in helping companies deploy products faster. Using Cloud services the New York Times managed to convert scanned images containing 11 million articles into PDF within 24 hours at a cost of merely 240 US-\$. However Cloud providers will only offer their services, if they can realize sufficient benefit. To achieve this, the efficiency of Cloud infrastructure management must be increased. To this end we propose the use of concepts from revenue management and further enhancements.*

**Keywords:** Cloud Computing, Revenue Management, Information Systems, Optimization

## **Introduction**

Due to high competition on global markets many enterprises face the challenge to make use of new applications and reduce process times on one side and cut the costs of their IT-infrastructures on the other side (Carr 2005).

To achieve this goal it is necessary to maintain a very high degree of flexibility with respect to the IT infrastructure. Facing this challenge the idea of Cloud computing has been gaining interest lately. The term Cloud Computing was initially coined by Amazon.com, which were among the first companies to offer Cloud services on a large scale. In the Cloud resources (e.g. processing power, storage and bandwidth) can be bundled as services, which are offered to Cloud users. These Cloud services can be accessed without knowledge of the underlying infrastructure. Cloud computing also allows the introduction of new products and services without large investments in installing or upgrading of the IT infrastructure. The New York Times managed to convert 4TB of scanned images containing 11 million articles into PDF files in very little time using Hadoop, a framework for distributed applications and Amazon's Cloud services. The actual conversion process took only 24 hours cost merely 240 US-\$ (New York Times Blog 2007). New Projects and websites like Justin.tv or SmugMug.com would not have been able to adapt to such a rapidly growing user base without the help of Cloud services (Vogt 2007). Don MacAskill, CEO of SmugMug, estimated his company saved about 339,430 US-\$ in 7 month, spending only 84,255 US-\$ on Amazons Simple Storage Service (SmugMug Blog 2006). According to analysts at Gartner "The projected shift to Cloud Computing will result in dramatic growth in IT products in some areas and in significant reductions in other areas." (Gartner 2008).

The more providers offer their resources or services, the more likely it is they can be accessed at competitive prices. Therefore it is important to attract more providers. However, providers will only offer their services if they can realize sufficient benefit. This contradiction can only be solved by increasing the efficiency of Cloud infrastructure management. To improve performance in the commercialization of distributed computational resources, decisions about the supplied resources and their management should be based on both technical and economic aspects (Kenyon and Cheliotis 2004). With state-of-the-art technology, this assimilation is hampered, as the local resource managers facilitating the deployment of the resources are not designed to incorporate economic issues (e.g. price).

The example of Amazon.com as a Cloud provider can be used to illustrate some of the challenges for Cloud providers. Amazon maintains a Cloud infrastructure that is used by its own shopping website but also offers different services for external use. This allows it to balance usage spikes to some extent and realize economies of scale in the maintenance of its infrastructure. However, the sale of services to external users should not have negative impacts on its core business. Especially it should not result in decreased resource availability for its shopping site. These effects can be avoided by maintaining a large buffer of spare resources or by an effective capacity and resource management.

In recent times, several research projects have started to develop price-based resource management components which could be applied to some extent for Cloud Computing. Those approaches are devoted to scheduling by utilizing the price mechanism. Clearly, this means that technical issues such as resource utilization are ignored for scheduling. In addition, resource management is much more comprehensive than just scheduling. For example Service-Level-Agreement (SLA) management is also part of resource management that is often omitted in economic approaches. This plays an important role when deciding which already ongoing jobs to cancel in overload situations to maintain system stability.

Technical resource management systems typically offer the possibility to include priorities for user groups. In purely price-based schedulers it is not possible to distinguish important from unimportant partners, as only current price matters for the allocation.

This leads to the overall research question which jobs Cloud providers should accept in order to comply with their business policies and maximize revenue. To solve this issue, providers need a decision model helping them to take this decision.

In this work it will be motivated that a policy-based revenue management mechanism should be used to decide on whether incoming job requests should be accepted or not. Furthermore we will introduce several enhancements like client classification and dynamic pricing, show how they can be used to improve the decisions and why they should be integrated into resource management systems. Essentially, there are three main reasons for the integration of client classification: First, it allows giving internal users the necessary priorities to maintain their service levels.

Second it permits the inclusion of long-term oriented relationships with strategically important customers so-called credential components. Finally, client classification can be used as an instrument of revenue management, which allows skimming off consumer surplus. The integration of dynamic pricing allows giving customers incentives to run their jobs during times of low utilization and thereby achieve a more even utilization. Furthermore, it can help achieve higher prices and therefore higher revenue in times of high demand. We will explore the use of a job acceptance model using enhancements such as client classification and dynamic pricing to resource management.

This work is structured as follows. After the introduction some theoretical background and related work are presented. Subsequently the Job Acceptance Model and the policies are explained. Following a description of an Economically Enhanced Resource Manager which uses this model, we evaluate the model using simulations. Finally a conclusion and outlook on future work is presented.

## Theoretical Background

The key question is how to decide whether a job is accepted or not. The standard approach would be to always accept a job, if there is enough capacity available to fulfill it. Capacity is allocated on a first come first serve basis. However, this approach does not deliver the optimal allocation. Therefore it is necessary to take a closer look at the underlying decision problem.

### Theoretical Optimum

Assuming the provider is only interested to maximize its revenue and there are no further constraints, a simple instance of this problem would be:

$$\max_x \sum_{j \in J} x_j * fp_j \quad (O1)$$

Subject to:

$$\sum_{j \in J} c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C1.1)$$

Where T is the set of all regarded timeslots; J is the set of available jobs; R is the set of all resource types;  $fp_j$  is the price paid for job j;  $x_j$  is a binary allocation variable indicating whether job j was accepted or rejected;  $c_{jr}(t)$  is the capacity required by job j in timeslot t; and  $c_r(t)$  is the total capacity available for resource type r during timeslot t. (O1) is the objective function and represents the achieved revenue. The constraint (C1.1) is the capacity constraint. It assures that not more capacity can be allocated than is available.

In addition to the capacity constraints further requirements could exist, e.g. the requirement to have a certain amount of resources available for a certain client. The problem can be formulated and solved as a linear program. However, there are two problems when choosing this approach. The first one is that this is an instance of the knapsack problem (Kellerer et al. 2004), to be more specific the temporal knapsack problem. As such it is NP-hard and therefore computationally intractable. There are efforts of developing algorithms which can deliver exact solutions to the temporal knapsack problem faster but the problem still remains NP-hard (Bartlett et al. 2005). Therefore early on scientists started develop heuristics or approximations which can be solved in polynomial time (Ibarra and Kim 1975).

One option would be to use such an approximation. However, to solve the problem, whether with an exact or an approximative algorithm, it is necessary to have sufficient information on the available jobs, the capacity and runtime they require, the prices, etc. The fact that in general this information is not available before a job is submitted constitutes the second problem.

Instead of trying to predict the exact job information and then running an approximation to solve the knapsack problem a different approach is chosen in this work. The idea is to use policies as a heuristic for maximizing the revenue without having the exact information about future jobs. These policies can be based on requirements from contracts with clients, company policies and information gained from historic workload traces, utilization curves, prices, etc. Policies can also be based on information about the statistical distribution of job size, runtime and prices.

In systems with real-time allocation the policies are used as an additional requirement to the capacity requirement when determining whether to accept a job. In batch scenarios it is also possible to use the policies to select various job candidates and then select the jobs using a best myopic response approach.

### ***Related Work***

There are two main research streams dealing with the admission and management of job requests. The first one comes from the research area of computer science and focuses on the technical aspects while incorporating some economic aspects. Another stream aims to adapt concepts and from Revenue Management to job admission problem. Most existing work applies these concepts to Grid computing, Utility computing or the Software-as-a-Service business.

Ferguson et al. (1996) discuss the application of economic theories to resource management. Aiber et al. (2004) present an architecture for autonomic self-optimization based on business objectives. Elements of client classification such as price discrimination based on customer characteristics have been mentioned in other papers (Newhouse et al. 2004 and Buyya 2002). They did however not consider other discrimination factors. Chicco et al. (2006) describe data-mining algorithms and tools for client classification in the electricity grids but concentrate on methods for finding groups of customers with similar behavior.

An architecture for admission control on e-commerce websites that prioritizes user sessions based their intentions to buy a product is proposed by Poggi et al. (2007 and 2009). They analyzed customer behavior, such as navigational clicks, on an e-commerce web site. Based on this behavior predictions about the user's intention to buy a product can be made. Consequently, Quality of Service for user sessions is shaped based the user's intentions.

Boughton et al. (2006) present research on how workload class importance can be considered for low-level resource allocation. They focus on competing workloads in databases and investigate who business policies describing the relative importance of workloads can be used to efficiently allocate resources.

One approach to realize end-to-end Quality of Service is the Globus Architecture for Reservation and Allocation (Foster et al. 1999). This approach uses advance reservations to achieve QoS. Another way to achieve autonomic QoS aware resource management is based on online performance models (Kounev et al. 2007b). They introduce a framework for designing resource managers that are able to predict the impact of a job in the performance and adapt the resource allocation in such a way that SLAs can be fulfilled. Both approaches do not consider achieving QoS in case of partial resource failure. This work makes use of the second approach and adds the mechanism of Job Cancellation.

The introduction of risk management to the Grid (Djemame et al. 2006) permits a more dynamic approach to the usage of SLAs. It allows modeling the risk that the SLA cannot be fulfilled within the service level agreement. A provider can then offer SLAs with different risk profiles. However, such risk modeling can be very complex. It requires information about the causes of the failure and its respective probabilities. Clients need to have the possibility to validate the accuracy and correctness of the providers risk assessment and risks have to be modeled in the SLAs.

The use of concepts from Revenue Management for Internet Service Providers was researched by Nair and Bapna (2001). They considered the decision to accept or reject customers but did not take different service types or advanced reservation into account.

One of the first papers analyzing Revenue Management concepts for cluster systems was published by Dube et al. (2005). The suggested model offers one resource for different prices. By assuming the customer behavior follows a logit model, the authors analyzed an optimization model for a small number of price classes and provided numerical results.

Sulisto et al. (2008) analyzed how overbooking strategies can be used to mitigate the effects of cancelations and no-shows and thus potentially increase revenue. Different overbooking policies were tested and compared using simulations. Urgaonkar et al. (2002) showed that overbooking can be beneficial in shared hosting platforms, however only the throughput rate was considered for optimization.

A framework for the application of Revenue Management in the field of Grid Computing was introduced by Anandasivam and Neumann (2009). It outlines a theoretic model and certain requirements for pricing of bundles of resources.

The related work covers different aspects of related research problems. However, the proposed mechanisms are only applicable to the cloud market to some extent and an overall decision model taking the characteristics and requirements of the Cloud market into account is still missing.

## **Key Features and Job Acceptance Model**

To improve performance in the commercialization of distributed computational resources and increase the benefit of service providers the introduction of a job acceptance model is proposed. This model helps the provider to decide which incoming jobs to accept in order increase revenue, while still complying to additional requirements introduced by its business model.

The proposed job acceptance model comprises various technical features as well as revenue management concepts. The main goals of these enhancements are to link technical and economical aspects of resource management and strengthen the economic feasibility of the Cloud. This can be achieved by establishing more precise price calculations for resources, taking usage of the resources, performance estimations and business policies into account. This chapter first describes the key features necessary to deal with the requirements of Cloud providers and customers. It then describes different policies which make use of these features and form the job acceptance model. The different policies and explained as well represented formally.

To adapt to different scenarios and business policies of different situations, it should be highly flexible and configurable via policies.

### **Key Features**

The motivational scenario in the introduction and the further requirements (Püschel et al. 2007) lead to four key features of the job acceptance model presented in this work. These should be able to deal efficiently with the motivational scenario presented in the introduction.

#### **Quality of Service.**

The first feature is Quality of Service (QoS). This is an important feature necessary to attract more customers. While some customers may accept best effort based services, a large share of customers considers the existence and fulfillment of adequate service levels a key factor in deciding which providers to choose.

QoS can be broken down into two aspects. The first aspect is to assure adequate performance during normal operation of the resources. Overload situations can lead to reduced overall performance (Nou et al. 2007) and thereby can result in breaking service level agreements between the provider and clients. Thus, it is necessary to have a mechanism that ensures that jobs will not be accepted if they result in an overload situation.

The second aspect of Quality of Service regards situations in which parts of the resources fail. To be able to fulfill all SLAs even in situations of partial resource failure, it would be necessary to keep an adequate buffer of free resources. Where this is not feasible there should at least be a mechanism that ensures that those SLAs that can be kept with the available resources are fulfilled. This can be done using the following feature which allows suspending or cancelling those jobs that cannot be finished in time due to the reduced availability of resources.

#### **Job Cancellation.**

Related to QoS is the feature automatic job suspension and cancellation. It is needed to ensure Quality of Service in situations where problems arise, i.e. parts of the resources fail or the estimations of the utilization were to optimistic. Cancellation of less important jobs to free capacity for incoming jobs with higher importance, i.e. jobs from a client with a higher classification or jobs that deliver significantly more revenue is also possible.

#### **Dynamic Pricing.**

Another enhancement is dynamic pricing based on various factors. Yeo and Buyya (2004) show an approach for a pricing function depending on a base pricing rate and a utilization pricing rate. The idea behind utilization based pricing is that when utilization is high, demand for the resources is high as well and therefore customers are willing to pay higher prices for resources. If utilization is low, lower prices are charged to attract more customers.

Pricing can also depend on other factors such as projected utilization, client classification, and projected demand. Pricing should also be contingent on the demand on the market.

### **Client Classification.**

To better integrate different groups of clients with varying requirements in a Cloud market it is beneficial to have some kind of classification. Certain classes of clients can then have different privileges. They can be discriminated regarding various factors, such as whether a job is accepted or not, pricing, Quality of Service, priority of the running jobs, and reservation of resources for certain clients when the system approaches high utilization.

As long as the availability of resources and the QoS is guaranteed it is not necessary to have different two separate infrastructures or keep a large buffer of idle resources for different client classes. Jobs from customers with lower requirements can be run on the spare resources and suspended once a job from an important customer arrives and needs the capacity. A Cloud that is able to accommodate users from different classes, promises better efficiency and may also be able to make better use of economies of scale.

In general, client classification allows better management of long-term relationships with strategically important customers by considering the client classification for the calculation of prices, scheduling of jobs and giving customers further privileges. Furthermore classification can enable the provider to improve revenue management, e.g. by skimming off customer surplus.

In some situations it might not be possible to use all aspects of client classification. If a provider receives all jobs through a market where clients have no influence on the decision to which provider their job is sent there is little benefit in using client classification. In this case the discrimination factors can still be used for product versioning. Obviously this does not allow improving customer relationship management and makes it more difficult to skim off customer surplus. However, it still can be useful to attract more customers and achieve a better utilization. The provider can offer different products with different levels of Quality of Service, access to resources and prices. Within the resource management component the same discrimination factors can be used, only in this case privileges are not based on client classification but on job or product identification. The different factors that can be used to differentiate various classes of clients or products are described in the following paragraphs.

**Price Discrimination.** The first factor is price discrimination. Customer-dependent pricing is one way to differentiate between different classes of clients. One idea to achieve this is introducing Cloud or Grid miles (Newhouse et al. 2004) in analogy to frequent flyer miles. Clients could be offered a certain amount of free usage of the resources or a 10% discount after spending a certain amount of money.

**Reservation of Resources.** For certain users it may be very important to always have access to the resources. This class of users could be offered a reservation of a certain amount of resources. One option is to reserve a fixed share of resources for a certain class of users another possibility is to vary this share depending on the usage of the system.

**Priority on Job Acceptance.** Another option is to use a client priority on job acceptance. When the utilization of the system is low, jobs from all classes of clients are accepted but when the utilization of the resources rises and there is competition between the clients for the resources, jobs from certain clients are preferred. There are two types of priorities: strict priorities and soft priorities.

- *Strict priority* means that if a job from a standard client and a client with priority compete for acceptance, the job from the client with priority always wins. For online allocation *strict priority* means that when a certain utilization threshold is reached only jobs from clients with this priority are accepted.
- *Soft priority* means jobs from clients with priority are generally preferred but standard clients have the chance to outbid clients with priority. For online mechanisms, standard clients can still submit jobs when the threshold is reached as long as they pay a higher price. Thus soft priority is essentially a discount on the reservation price or bid that may only apply in certain situation, i.e. when utilization exceeds a certain threshold.

**Quality of Service.** Another factor where differentiation for classes of clients is possible is Quality of Service. For some classes of clients Quality of Service is offered, for others not. Offering different levels of Quality of Service for different classes of clients is also possible. An example for this would be offering different risk levels (Djemame et al. 2006).

## Policies

The mentioned features allow the provider to include important factors when taking the decision which jobs or service requests to accept. However, it still needs a decision model to help take the actual decision which jobs to accept. To this end the following policies are introduced. They represent the some of the basic options - no enhancements, dynamic pricing with reservation prices, client classification with fixed reservation, client classification with strict priority, and client classification with soft priority:

- A simple first come first served policy (I) is used as a benchmark. Any incoming job is accepted if there is enough capacity available. This type of policy represents a simple system without any enhancements. The following mathematical formulation can be used to represent the FCFS Policy:

$$\max_x \sum_1^j \left(\frac{1}{2} * x_j\right)^j \forall j \in J \quad (O2)$$

Subject to:

$$\sum_1^j c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C2.1)$$

Where T is the set of all regarded timeslots; J is the set of available jobs; R is the set of all resource types;  $x_j$  is a binary allocation variable indicating whether job j was accepted or rejected;  $c_{jr}(t)$  is the capacity required by job j in timeslot t; and  $c_r(t)$  is the total capacity available for resource type r during timeslot t. The objective function (O2) is a generating function which is used to represent the sequential nature of the policy. Constraint (C2.1) is the capacity constraint which assures that not more capacity is allocated than is available for each resource type. The same generating function and capacity constraint is used for all policies.

- The second policy (II) uses dynamic pricing based on different reservation prices for different utilization levels.

$$\max_x \sum_1^j \left(\frac{1}{2} * x_j\right)^j \forall j \in J \quad (O2)$$

Subject to:

$$\sum_1^j c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C2.1)$$

$$(1 - H_1(p_j - p_{p1})) * \frac{\sum_1^j c_{jr}(t) * x_j}{c_r(t)} \leq l_{p1} \forall t \in T, \forall r \in R \quad (C2.2)$$

Constraint (C2.2) introduces utilization based pricing based on a certain reservation price  $p_{p1}$  that has to be achieved once the utilization surpasses the threshold  $l_{p1}$ . The variable  $p_j$  represents the price per unit and timeslot for job j.  $H_1(n)$  is the Heavyside step function.  $H_1(n)$  is 0 for  $n < 0$  and 1 for  $n \geq 0$ .

- Reservation of a fixed share of the resources is used in policy III. This is one of the simplest forms of client classification. It is included for comparison.

$$\max_x \sum_1^j \left(\frac{1}{2} * x_j\right)^j \forall j \in J \quad (O2)$$

Subject to:

$$\sum_1^j c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C2.1)$$

$$\frac{\sum_1^j c_{jr}(t) * cc_j * x_j}{c_r(t)} \leq l_c \forall t \in T, \forall r \in R \quad (C2.3)$$

$$\frac{\sum_1^j c_{jr}(t) * (1 - cc_j) * x_j}{c_r(t)} \leq (1 - l_c) \forall t \in T, \forall r \in R \quad (C2.4)$$

This policy essentially splits the resources into two pools. Constraint (C2.3) assures that certain fraction of the resources  $l_c$  is reserved for gold clients, whose jobs have the binary client classification variable  $cc_j$  set to 1. The rest of the resources can only be used by non-gold clients as can be seen in (C2.4). This policy is modeled after the separation of first class and economy class used in the airline industry.

- Client classification with strict priority is the proposed policy IV:

$$\max_x \sum_1^j \left(\frac{1}{2} * x_j\right)^j \forall j \in J \quad (O2)$$

Subject to:

$$\sum_1^j c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C2.1)$$

$$(1 - cc_j) * \frac{\sum_1^j c_{jr}(t) * x_j}{c_r(t)} \leq l_c \forall t \in T, \forall r \in R \quad (C2.5)$$

- The last basic policy, policy V, considered is client classification with soft priority. This type of policy is often referred to as nested pricing.

$$\max_x \sum_1^j \left(\frac{1}{2} * x_j\right)^j \forall j \in J \quad (O2)$$

Subject to:

$$\sum_1^j c_{jr}(t) * x_j \leq c_r(t) \forall t \in T, \forall r \in R \quad (C2.1)$$

$$(1 - H_1(p_j - p_{p1})) * (1 - cc_j) * \frac{\sum_1^j c_{jr}(t) * x_j}{c_r(t)} \leq l_c \forall t \in T, \forall r \in R \quad (C2.6)$$

In addition to these basic policies two mixed policies were evaluated.

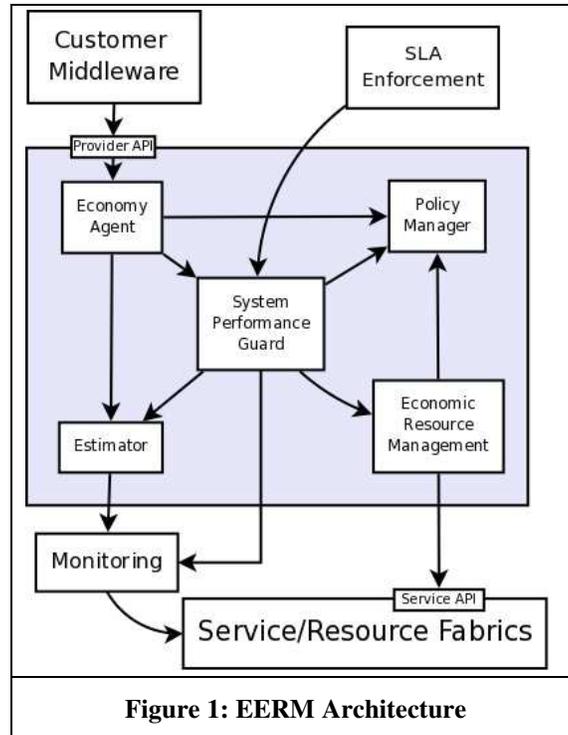
- Policy VI is a combination of dynamic pricing and strict priorities. This policy could be used providers selling spare capacity. Internal users can be classified as gold users and get strict priority. Instead of the extra charge for priority internal users would get a discount on job prices. It uses objective function (O2), and the constraints (C2.1), (C2.2) and (C2.5).
- A combination of dynamic pricing with soft priorities, as used in policy VII, would be suitable for Cloud providers with preferred clients. These gold clients get soft priority but standard clients can get the same priority for a job by sending sufficiently high bids. It uses objective function (O2), and the constraints (C2.1), (C2.2) and (C2.6).

## Architecture of an Economically Enhanced Resource Manager

To show the validity and feasibility of the approach the features and policies which form the model are integrated in the architecture of an Economically Enhanced Resource Manager (EERM). This section describes the architecture of

this EERM. It includes a description of the components with the aid of sequence diagrams. The EERM is not meant to run on each node in the Cloud. One instance of the EERM manages a complete cluster of systems or can even manage various clusters.

In a first step, it is necessary to register resources with the EERM. The estimator component has to be provided with the necessary technical information about the resources and the respective policies for the resources are uploaded to the Policy Manager. If new resources become available they are registered with the EERM in the same manner. If it is planned to take resources offline, they should be deregistered with the EERM first. This way the reduced capacity can be taken into account for price calculation and job acceptance. If resources are not deregistered but simply taken offline, the Monitoring component will notify the EERM about the resource failure. Subsequently the EERM will take the necessary adjustments for the reduced capacity. This however can mean that it is necessary to cancel jobs. An overview of the architecture of the EERM can be seen in Figure 1.

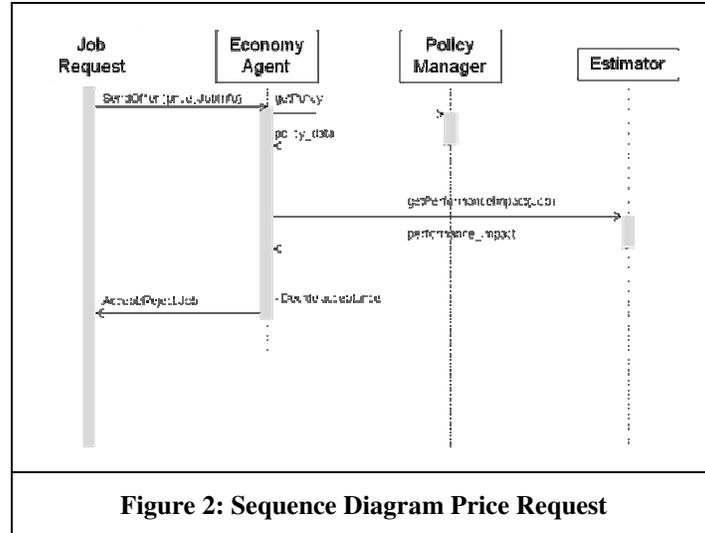


The EERM interacts with various other components, namely the Customer Middleware, a Monitoring component, the Resource Fabrics, and a component responsible for SLA Enforcement. The Customer Middleware accesses the Cloud via the Provider API. The Monitoring is responsible for monitoring the state and the performance of the resources and notifying the System Performance Guard in case of problems. Additionally data collected by the Monitoring is used by the Estimator component for its predictions. Resource Fabrics refers to the underlying software stack delivering the actual service. The SLA Enforcement component is necessary to monitor enforcement of the SLAs from an independent site. It can also supply the System Performance Guard with information whether the late delivery of results is acceptable for the client under certain conditions.

### ***Economy Agent***

The Economy Agent is responsible for deciding whether incoming jobs are accepted and for calculating their prices. These calculations can be used both for negotiation as well as bids or reservation prices in auctions. Figure 2 shows the sequence of a price request. First, the Economy Agent receives a request from a market agent. Then it checks whether the job is technically and economically feasible and calculates a reservation price for the job based on client category, resource status, economic policies and predictions of future job executions from the estimator component.

Once the Economy Agent has the information about utilization and the client classification it can determine whether the job is accepted. If the Job is accepted, the price for the job is calculated and sent back. If not, the job is rejected.



**Estimator.**

The Estimator component calculates the expected impact on the utilization of the resources. This is important to prevent reduced performance due to overload (Nou et al. 2007), furthermore the performance impact can be used for the calculation of prices. This component is based on work by Kounev et al. using online performance models (Kounev et al. 2007a).

**System Performance Guard.**

The System Performance Guard is responsible for ensuring that the accepted SLAs can be kept. In case of performance problems with the resources it is notified by the monitoring component. After checking the corresponding policies it determines if there is a danger that SLAs cannot be fulfilled. It then takes the decision to suspend or cancel jobs to ensure the fulfillment of the other SLAs and maximize overall revenue. Jobs can also be cancelled when capacity is required to fulfill commitments to preferred clients.

**Policy Manager.**

To keep the EERM adaptable the Policy manager stores and manages the different policies, such as policies concerning client classification and job cancellation. Policies are formulated in the SWRL, the Semantic Web Rule Language (Horrocks et al. 2004). All features of the EERM require the respective components to be able to communicate with the Policy Manager and base their decisions on the corresponding policies. A simple example from a pricing policy in SWRL is the following rule which expresses that if the utilization is between 71% and 100% there is a surcharge of 50:

```

Utilization(?utilization)ΔInsideUtilizationRange(?utilization, "71% - 100%")
    → SetSurcharge(?utilizationsurcharge, "50")
    
```

**Economic Resource Management.**

The Economic Resource Management is responsible for the communication with the local resource managers and influences the local resource management to achieve a more efficient global resource use.

**Experiment and Evaluation**

The evaluation of this proposal consists of two parts. First, a simple evaluation scenario based on generated workloads is presented. Then the results obtained in a more thorough evaluation based on real world workloads are shown. The example scenario and simulation setting are described, and the results are presented. For the evaluation a simulator called EERMSim was written in Java.

### Generated Workload Scenario

For the synthetic workloads 1500 jobs were generated. The arrival rate of jobs per timeslot was kept constant. Runtimes were generated based on a lognormal distribution (Feitelson 2009). Ten joblists were generated. These joblists were used to first evaluate three policies, namely dynamic pricing (policy II), soft priority (policy V) and a combination of both (policy VII). A simple first come first served policy (I) is used as a benchmark.

Table 1 shows a comparison of the simulation results for the different policies. It contains the policies,  $\mu$  revenue,  $\sigma$  revenue,  $\sigma/\mu$  revenue, and the revenue increase compared to the benchmark policy I.

<b>Table 1: Result of the simulation with generated workloads</b>				
Policy	$\mu$ revenue	$\sigma$ revenue	$\sigma/\mu$ revenue	revenue increase
I	1054006,89	66411,34	0,06300845	0,00%
II	1162197,56	58235,39	0,050108	10,26%
V	1200356,89	56138,72	0,04676836	13,89%
VII	1321126,33	65129,72	0,04929863	25,34%

The average revenue is  $\mu$  revenue,  $\sigma$  revenue denotes the standard deviation and  $\sigma/\mu$  revenue is the coefficient of variation for the revenue. This coefficient can be described as the relative standard deviation and gives a better insight how large the standard deviation is in relation to the mean.

It can be seen that the introduction of dynamic pricing delivers a revenue increase of roughly 10%. Soft priority shows a further increase in revenue, 13,89% compared to the benchmark. The combination of policy II and V even manages to outperform the benchmark by more than 25%. The order of the revenue achieved by these policies was always the same in the evaluation. In no case did the benchmark beat policy II, V or VII. As the results of the simulation with generated workloads were very interesting, a more detailed evaluation with all policies, different policy configurations was done with real workloads.

### Real Workload Trace Scenario

For the scenario based on real workloads, data from the Parallel Workload Archive (Feitelson 2009) was used. The SHARCNET log, which was graciously provided to the Parallel Workload Archive by John Morton (john@sharcnet.ca) and Clayton Chrusch (chrusch@sharcnet.ca), was used as a basis for this scenario. It contains 1,195,242 jobs sent to a set of 10 clusters in Ontario, Canada from a period starting in December 2005 and ending in January. The SHARCNET log was chosen as a basis because it contains a large variety of jobs with different runtimes, numbers of used cpus, and varying submit and start times. In contrast to the generated workload this scenario shows the variation in demand over time.

Jobs running less than one hour or more than 10 days were filtered. Subsequently job runtimes were rounded down to full hours to allow a timeslot based allocation. After filtering invalid jobs and jobs which were considered to long or to short 566,701 jobs were left and used for the scenario. Based on these workloads ten joblists with different prices were generated.

Pricing information was generated using a normal distribution. Prices per unit were generated with a mean of 1 and a variance of 0.5. As a further condition a minimum unitprice of 0.1 was required. Full prices were calculated by multiplying unitprices with the number of cpus and timeslots and rounding the value up to the next integer. It was assumed that gold clients are willing to pay an extra charge for the priorities.

In this scenario, the capacity is determined by the number of cpus and limited to 2050. This capacity was chosen to accommodate the larger jobs from the SHARCNET log but still have time periods where demand exceeds supply. Jobs can either start in the same timeslot in which they are submitted or in a later timeslot. Using this scenario as a basis various policies are evaluated.

For each of the general policies, besides the benchmark case I, a number of configurations with different policy parameters, such as reservation prices and corresponding utilization levels, were tested. In total 88 configurations were used. Each of these configurations was tested with all ten joblists.

In the simulations with client classification, it was assumed that the gold clients are willing to pay an extra fee for the priority, if their requirements were met. The simulations were run on a virtual machine in the Amazon Elastic Compute Cloud (Amazon EC2). The virtual machine ran on the small instance size with the cpu power equivalent to a 1.0-1.2 GHz 2007 Opteron processor and 1.7 GB of memory. The runtime of the simulation for one parameter set and 566,701 jobs was usually between 120 and 130 seconds.

### **Results for the Workload Scenario**

This chapter starts with a general comparison of the results of the different general policies and then continues with a more in depth analysis of the results obtained with different parameters in each case.

The first come first served policy does not use any economic enhancements and does not have any parameters that can be adjusted. The mean revenue over the ten simulations is 6,004,783.6. This revenue serves as benchmark for all the other policies.

Table 2 shows a comparison of the simulation results for the different policies. In analogy to the first table, it contains the basic policies,  $\mu$  revenue,  $\sigma$  revenue,  $\sigma/\mu$  revenue, and the revenue increase compared to the benchmark policy I.

<b>Table 2: Result of the simulation with workload traces</b>				
Policy	$\mu$ revenue	$\sigma$ revenue	$\sigma/\mu$ revenue	revenue increase
I	6004783.6	34921.75	0.005816	0.00%
II	6794802.59	122430.51	0.018018	13.16%
III	4819177.66	539636.34	0.111977	-19.74%
IV	7385244.49	140937.44	0.019084	22.99%
V	7677053.68	76865.17	0.010012	27.85%
VI	7627947.58	243319.52	0.031898	27.03%
VII	7797407.76	291719.3	0.037412	29.85%

The mean revenue, standard deviation, and the coefficient of variation were calculated over all configurations and ten simulation runs for each policy.

With policy II, the policy using dynamic pricing, the average revenue is 13.16% higher than the benchmark.

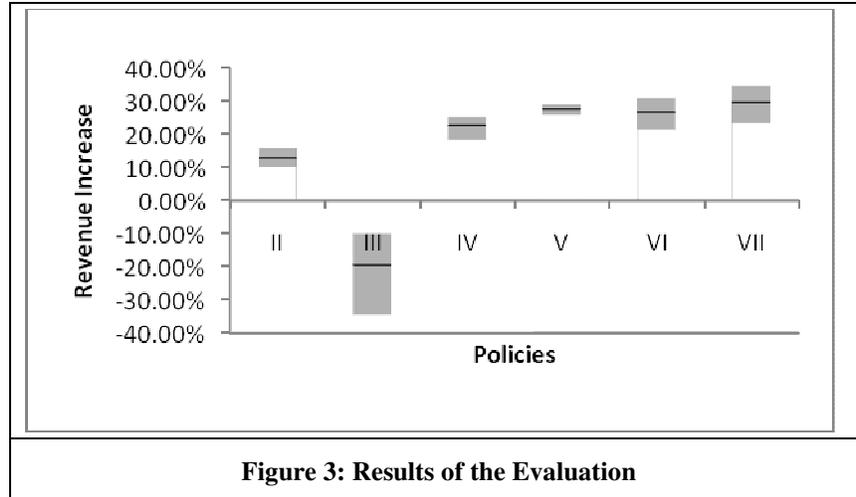
Policy III is not able to beat the benchmark, on average it delivers only 80.26% of the revenue achieved without the enhancements. This result was expected. Reservation of a fixed share of the capacity for one client or one class of clients only makes sense in certain situations. It can make sense when the gold users have a very even demand curve or are willing to pay high fees for their reservation. But the generally those assumptions cannot be expected to be true.

Policy IV which uses strict priorities performs significantly better than the benchmark. It also delivers a higher revenue than policy II. With this policy gold users have access to the complete grid and jobs from standard users are only rejected if the utilization exceeds the defined threshold.

In policy V standard users still can submit jobs during high utilization if the unitprice exceeds a certain reservation price. Due to the fact that more standard jobs with very high prices are accepted there is a further increase in revenue compared to policy IV.

Since both dynamic pricing and priorities resulted in increased revenue combinations between these policies were also considered. A combination between policies II and III was not considered because the mechanism of reserving a fixed share of the capacity did not increase the revenue. Both regarded combinations resulted in a further increase of revenue compared to their basic components.

The coefficient of variation,  $\sigma/\mu$  revenue, is very low in the results of all policies. This indicates that the influence of the exact prices on the results is limited as long as the distribution from which they are drawn remains the same.



**Figure 3: Results of the Evaluation**

Figure 3 shows the average revenue for all policies, the revenue for the best configuration for each policy and the revenue for the worst configuration for each policy. The ranking of the policies IV, V, VI and VII is not the same in each case; it depends on the configuration for each policy. However with the corresponding configurations policies with soft priorities always outperform their counterparts with strict priorities.

Different policies have different number of parameters and a different number of configurations were tested for various policies. Therefore, it is important to take a closer look at the simulation results for the different configurations to evaluate each policy.

For dynamic pricing five parameters can be specified: a general reservation price and two thresholds with respective reservation prices. In total 18 configurations were used for this policy.

The mean of all revenues and all configurations is 6794802.59. This is a 13.16% increase. With an average of 6877529.30 the configurations with a general reservation price of 0.3 deliver a higher revenue than those with 0.2, which had an average of 6815914.75 and 0.1 with a mean of 6690963.73. This indicates that increasing the reservation price further above 0.3 might lead to even higher revenues in our scenario.

Another important point can be seen from the results of the evaluation of policy II. Revenue is increased for a wide range of parameter values. Even with the worst of the tested configurations an increase in revenue of more than 10% can be observed. The risk of decreasing revenue with this policy is very low. Even with only very limited knowledge about the distribution of jobs and prices it is relatively easy to find configurations which increase revenue.

In policy III the only available parameters are the capacity reserved for the gold clients and the user IDs of the gold clients. The list of the gold clients was kept static throughout the simulation. As expected the policy with reservation of a fixed share of the resources leads to lower revenue than the benchmark. The main cause for this is the fact that the resource pool is essentially divided into two parts and therefore differences in the demand curves are not balanced. There can be situations where one group of users have low demand while others need more capacity. In this situation jobs from one class of users are rejected while resources reserved for other users remain idle.

The highest revenue, 5405299.20, was achieved when 1700 of 2050 cpus were reserved for gold clients. This is still only 90.02% of the benchmark. It is very likely that the revenue can be increased further by fine tuning the share of the reserved capacity. However this is not feasible in real life situations. Since the exact distribution of the incoming jobs is usually not known fine tuning the parameters to it is not possible. Therefore, a mechanism is only considered beneficial if it increases revenue for a wide range of parameter values or distributions.

Furthermore using strict priorities, as used in policy IV will always deliver better results than policy III. The policy of strict priorities also assures that a certain amount of capacity can be used exclusively by gold users. But in contrast to the previous situation gold users have access to the complete grid capacity. If they submit enough jobs to keep the utilization above the priority threshold over a long period of time this can lead to a situation where 100% of the capacity is used by the gold user. This policy leads to an average revenue of 7385244.49 which is a 22.99%

increase compared to policy 1. With the best configuration, a threshold of 1750, a revenue of 7522863.70 was achieved.

With the policy of soft priorities there is a further increase of the revenue to 7758685. This policy leads to higher revenue than strict priorities since it is now possible for standard clients to outbid gold clients.

As dynamic pricing as well as client classification with priorities showed an increase in revenue it is interesting to see whether the revenue can be further increased by combining the mechanisms.

The combination of dynamic pricing with strict priorities leads to an average revenue of 7627947,58. This is a revenue increase compared to policy IV. The highest achieved revenue is 7870507, an increase of 4,62% compared to the best tested configuration of policy IV. It was achieved with the following configuration:

The combination of dynamic pricing with soft priorities delivers the best results. Even with the worst configuration, the achieved revenue is 7402471.10. This is 23.28% more than the revenue without using any economic enhancements. With the best of the considered configurations the revenue is 8084992. This is an increase of 34.46% compared to the benchmark. The average of all configurations is 7797407.76, still a 29.85% increase in revenue.

## **Conclusion and Outlook**

In this work various economical enhancements for resource management are motivated. In the section on the theoretical background it was described how the allocation which delivers the highest possible utility can be found. However due to the complexity of this approach this approach and the fact that the necessary information is generally not available during runtime this approach cannot be used. The use of policies as a heuristic is proposed as a solution which is less complex and can be used with the available information.

Subsequently the key features are introduced and explained. The first two features, Quality of Service, and automatic job cancellation, comprise mechanisms for assuring Quality of Service and dealing with partial resource failure without introducing the complexity of risk modeling. The next feature is dynamic pricing based on various factors such as utilization and client classification. Utilization based pricing allows to adapt the price to set incentives for using resources during times of low utilization.

Client classification is related to all previous features. There are various factors which can be used to discriminate different classes of clients, such as price, Quality of Service and resource access. When discriminating on resource access strict or soft priorities are more flexible than a fixed reservation of a share of the resources. Therefore priorities should be preferred as discrimination factor. Subsequently an overall job acceptance model, which helps the provider decide which jobs to accept, was introduced. Several policies that can be used for this model were motivated and explained.

With information about the features and the choice of policies as a heuristic the architecture for an Economically Enhanced Resource Manager integrating these enhancements are introduced in chapter 5. The general architecture as and the different components of the EERM are described. To assess the benefit of the mechanisms an evaluation is presented. The evaluation was done with a simulator written in Java and using first generated and then real workload data as a basis. The results of the simulation show a clear benefit when using the enhancements.

With a dynamic pricing policy based on utilization an increase of 15.98% was achieved in the best case of the real workload scenario. The policy of fixed reservations was not able to beat the benchmark, however this could be expected. Since priorities are more flexible and offer no disadvantage compared to fixed reservation they should always be preferred. Both policies using strict and soft priority beat the benchmark. With soft priorities an increase in revenue of up to 29,07% was achieved. The combination of dynamic pricing and priorities leads to a further increase in of revenue of up to 34,64%. Improvements in revenue are achieved with a wide range of parameters, showing that increases in revenue can be achieved even if the optimal parameters for each policy cannot be found in advance.

The next steps include further integration of the EERM with different market mechanisms. The architecture was designed to be adaptable to a wide range of situations but the evaluation was only done with a very simple pay as you bid mechanism. The automatic generation, evaluation and optimization of policies based on historic data is also an interesting topic. Another next step is the projection of future demand and inclusion of this information in the policies. Finally, it should be considered how clients react and adapt their behavior if a provider implements such policies.

## References

- Aiber, S., Gilat, D., Landau, A., Razinkov, N., Sela, A., and Wasserkrug, S. 2004. "Autonomic self-optimization according to business objectives," *ICAC '04: Proceedings of the First International Conference on Autonomic Computing (ICAC'04)*, Washington, DC, USA: IEEE Computer Society, pp. 206–213.
- Anandasivam, A., and Neumann, D. 2009. "Managing revenue in Grids," *Hawaii International Conference on System Sciences, Proceedings of the 42nd Annual*, Springer-Verlag GmbH.
- Bartlett, M., Frisch, A.M., Hamadi, Y., Miguel, I., Tarim, S.A., and Unsworth, C. 2005. "The Temporal Knapsack Problem and Its Solution," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, [http://dx.doi.org/10.1007/11493853\\_5](http://dx.doi.org/10.1007/11493853_5).
- Bertsimas, D., and Popescu, I. 2003. "Revenue Management in a Dynamic Network Environment," *Transportation Science* (37:3), pp. 257–277.
- Boughton, H., Martin, P., Powley, W., and Horman, R. 2006. "Workload class importance policy in autonomic database management systems," *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, Washington, DC, USA: IEEE Computer Society, pp. 13–22.
- Buyya, R. 2002. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, Ph.D. thesis, Monash University.
- Campbell, D. E. 1987. *Resource Allocation Mechanisms*, London, UK: Cambridge University Press.
- Carr, N. G. 2005. "The end of corporate computing," *MIT Sloan Management Review* (46:3), pp. 32–42.
- Chicco, G., Napoli, R., and Piglion, F. 2006. "Comparisons among clustering techniques for electricity customer classification," *IEEE Transactions on Power Systems* (21:2), pp. 933–940.
- Djemame, K., Gourlay, I., Padgett, J., Birkenheuer, G., Hovestadt, M., Kao, O., and Voß, K. 2006. "Introducing risk management into the grid," *The 2nd IEEE International Conference on e-Science and Grid Computing (eScience2006)*, Amsterdam, Netherlands, p. 28.
- Dube, P., Hayel, Y., and Wynter, L. 2005. "Yield management for IT resources on demand: analysis and validation of a new paradigm for managing computing centres," *Journal of Revenue and Pricing Management* (4:1), pp. 24–38.
- Feitelson, D. G. 2009. *Workload Modeling for Computer Systems Evaluation*.
- Ferguson, D. F., Nikolaou, C., Sairamesh, J., and Yemini, Y. 1996. *Economic models for allocating resources in computer systems*, pp. 156–183.
- Foster, I., and Kesselman, C. 1997. "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications and High Performance Computing* (11:2), pp. 115–128.
- Foster, I., Kesselman, C., Lee, C., Lindell, B., Nahrstedt, K., and Roy, A. 1999. "A distributed resource management architecture that supports advance reservations and co-allocation," *Proceedings of the 7th International Workshop on Quality of Service (IWQoS 1999)*, London, UK, pp. 62–80.
- Gartner Inc. 2008. "Gartner Says Worldwide IT Spending On Pace to Surpass \$3.4 Trillion in 2008," Press Release, <http://www.gartner.com/it/page.jsp?id=742913>, accessed on August, 21 2009.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., and Dean, M. 2004. "Swrl: A semantic web rule language combining owl and ruleml," *w3c member submission*.
- Ibarra, O.H., and Kim, C.E. 1975. "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems," *J. ACM* (22:4), pp. 463–468.
- Kenyon, C., and Cheliotis, G. 2004. "Grid resource commercialization: economic engineering and delivery scenarios," *Grid resource management: state of the art and future trends*, pp. 465–478.
- Kellerer, H., Pferschy U., Pisinger D. 2004. *Knapsack Problems*. Springer Verlag.
- Klein, R (2007). "Network capacity control using self-adjusting bid-prices," *OR Spectrum* (29:1), pp. 39–60.
- Kounev, S., Nou, R., and Torres, J. 2007a. "Autonomic qos-aware resource management in grid computing using online performance models," *The 2nd International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2007)*, Nantes, France.
- Kounev, S., Nou, R., and Torres, J. 2007b. "Building online performance models of grid middleware with fine-grained load-balancing: A globus toolkit case study," *The 4th European Engineering Performance Workshop (EPEW 2007)*, Berlin, Germany.
- Litzkow, M. J., Livny, M., and Mutka, M. W. 1988. "Condor - A hunter of idle workstations," *Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS 1988)*.
- Moeller, A., Roemisch, A., and Weber, K. 2008. "Airline network revenue management by multistage stochastic programming," *Computational Management Science* (5:4), pp. 355–377.

- Nair, S., and Bapna, R. (2001) "An application of yield management for Internet Service Providers," *Naval Research Logistics* (48:5), pp. 348–362.
- Newhouse, S., MacLaren, J., and Keahey, K. 2004. "Trading grid services within the uk e-science grid," *Grid resource management: state of the art and future trends*, pp. 479–490.
- New York Times Blog 2007. TimesMachine use case, <http://open.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/> (accessed on August, 21 2009)
- Nou, R., Julià, F., and Torres, J. 2007. "Should the grid middleware look to selfmanaging capabilities?," *The 8th International Symposium on Autonomous Decentralized Systems (ISADS 2007)*, Sedona, Arizona, USA, pp. 113–122.
- Poggi, N., Moreno, T., Berral, J. L., Gavaldà, R., and Torres, J. 2007. "Web customer modeling for automated session prioritization on high traffic sites," *Proceedings of the 11th International Conference on User Modeling, Corfu, Greece*.
- Poggi, N., Moreno, T., Berral, J.L., Gavaldà, R., and Torres, J. 2009. "Self-adaptive utility-based web session management," *Comput. Netw.* (53:10), pp. 1712-1721.
- Püschel, T., Borissov, N., Macías, M., Neumann, D., Guitart, J., and Torres, J. 2007. "Economically Enhanced Resource Management for Internet Service Utilities," *Web Information Systems Engineering – WISE 2007*, [http://dx.doi.org/10.1007/978-3-540-76993-4\\_28](http://dx.doi.org/10.1007/978-3-540-76993-4_28).
- Rappa, M. A. 2004. "The utility business model and the future of computing services," *IBM Systems Journal* (43:1), pp. 32–42.
- SmugMug Blog 2006. Amazon S3: Show me the money, <http://blogs.smugmug.com/don/2006/11/10/amazon-s3-show-me-the-money/> (accessed on August, 21 2009)
- Sulistio, A., Kim, K., and Buyya, R. 2008. "Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue," *Proceedings of the 2008 8th International Symposium on Cluster Computing and the Grid (CCGRID)*, IEEE Computer Society.
- Talluri, K., and van Ryzin, G. 2004. *The Theory and Practice of Revenue Management*, Berlin, 2004.
- Urgaonkar, B., Shenoy, P., and Roscoe, T. 2002. "Resource overbooking and application profiling in shared hosting platforms," *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*. New York, NY, USA, pp. 239–254.
- Vogt, K., 2007. „Justin.tv and AWS“, presentation at Amazon Web Services The Start-Up Project - San Francisco, <http://www.slideshare.net/tracylaxdal/justintv-aws-the-startup-project>
- Wurman, P. R. 1999. *Market structure and multidimensional auction design for computational economies*, Ph. D. thesis, University of Michigan.
- Yeo, C. S., and Buyya, R. 2004. "Pricing for Utility-driven Resource Management and Allocation in Clusters," *Proceedings of the 12th International Conference on Advanced Computing and Communications (ADCOM 2004)*, Ahmedabad, India, pp. 32–41.