Scientific progress of design research artefacts

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

Arun Anandasivam
Universität Karlsruhe, arun.anandasivam@iism.uni-karlsruhe.de

Stefan Buschek
Universität Karlsruhe, stefan.buschek@student.kit.edu

Dirk Neumann
Albert-Ludwigs-Universität Freiburg, dirk.neumann@is.uni-freiburg.de

MAKING MONEY WITH CLOUDS:

REVENUE OPTIMIZATION THROUGH AUTOMATED POLICY DECISIONS

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

Arun Anandasivam, Universität Karlsruhe (TH), Institute of Information Systems and Management (IISM), Englerstraße 14, 76131 Karlsruhe, Germany, arun.anandasivam@iism.uni-karlsruhe.de

Stefan Buschek, Universität Karlsruhe (TH), Institute of Information Systems and Management (IISM), Englerstraße 14, 76131 Karlsruhe, Germany, stefan.buschek@student.kit.edu

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

Abstract

Many existing IT applications exhibit strongly varying demand patterns for resources. Accommodating an ever increasing and highly fluctuating demand requires continuous availability of sufficient resources. To achieve this state at reasonably costs, a high degree of flexibility with respect to the given IT infrastructure is necessary. Facing this challenge the idea of Cloud computing has been gaining interest. In so-called Clouds resources such as CPU, storage and bandwidth can be bundled into a single services, which are offered to Cloud users. These services can be accessed in oblivion of the underlying IT infrastructure. This way Cloud Computing facilitates the introduction of new products and services without large investments in the IT infrastructure.

Cloud Computing is a promising approach with a high impact on business models. One aspect of business models is clearly the revenue model, which defines how prices should be set to achieve predefined revenue level. The decision about accepting or denying requests has a high impact on the revenue of the provider. In this paper we analyze two approaches that support the cloud provider in its decision. We show that predefined policies allow increasing revenue compared to widely used technical models such as first-come-first-serve.

Keywords: Cloud Computing, Revenue Management, Decision Mechanisms

1 INTRODUCTION

Many existing IT applications (e.g. data mining, portfolio analysis or numerical simulations) exhibit strongly varying demand patterns for computing resources. Demand for computing resources is further increased by the introduction of new products and processes. Accommodating an ever increasing and highly fluctuating demand requires that sufficient resources are available. Building an adequate infrastructure requires significant investments in the infrastructure and instructing and hiring of personnel. However, many enterprises face increasing competition on global markets. This leads to the challenge to make use of new applications, reduce process times, and introduce innovative services for customers 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-$. New Projects and websites like Justin.tv or SmugMug.com would not have been able to adapt to such a rapidly growing user base without the help of Cloud services. Don MacAskill, CEO of SmugMug, estimated his company saved about 339,430 US-$ in 7 month, spending only 84,255 US-$ on Amazons Simple Storage Service. 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.

In order to remain attractive for customers Cloud services must meet user demands concerning price, availability and quality of service. The more providers offer their services, the more likely it is that they can be accessed in quantity and quality. Therefore it is important to attract more providers. However, Cloud providers will only offer their services if they can realize a sufficient level of profit. Providers have to plan their resource usage carefully and be aware of dynamic changes of the incoming requests for their services. With state-of-the-art technology, this assimilation is hampered, as the components facilitating the deployment of the services are not designed to incorporate economic issues (e.g. price, expected demand). 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).

In recent times, several research projects have started to develop price-based resource management components supporting the idea of grid or utility computing, which could be applicable for Cloud providers in some cases. Those approaches are devoted to scheduling by using price mechanisms such as auctions. Clearly, this means that technical issues such as resource utilization are ignored for scheduling. Furthermore, these approaches require a tight integration with the infrastructure of the provider, as well as the interface to the customer. In cases where a provider offers a wide range of services and customers might use different providers at the same time, it is often not feasible to undertake a complex and time consuming bidding process. Technical resource management systems typically offer the possibility to include priorities for users. Purely price-based schedulers do not distinguish important from less important partners, as only current price matters for the allocation. In this work we propose two mechanisms which increase revenue of Cloud providers by deciding which job or service requests are accepted and or rejected. The first mechanism provides an elegant way to introduce utilization based pricing in real world systems. Integration of concepts from revenue management in the second mechanism allow increasing overall revenue even in fixed-price scenarios which dominate Cloud Computing so far.

This paper is novel because it proposes mechanisms for fixed price, as well as pay-as-you-bid scenarios which can be used in real-time and advance reservation scenarios. At the same time our mechanisms can be implemented in real world scenarios with relatively few changes to existing architectures. With increasing competition among Cloud providers work covering this topics is highly relevant in the research area of information systems. The paper is structured as follows. In the next section we introduce motivational scenarios and the requirements they impose. Subsequently we present work related to our model. Section 4 contains a description of our approaches and the models they are based on. An evaluation based simulations follows in section 5. The paper ends with a conclusion and brief outlook of future work.

2 MOTIVATIONAL SCENARIOS AND REQUIREMENTS

In this section we present three major real world scenarios, which motivate the use of automated policy decisions. The scenarios impose different requirements on the decision making unit concerning the timeframe, which is available for determining the decision.

2.1 Motivational Scenarios

Scenario I: A service provider offers a number of different cloud services on the market, similar to like Salesforce.com or mor.ph. Most of these services have to be accessible within very short time spans. Each service has a different profile of resource requirements. Some service are computationally intensive, others require large amounts of memory or storage. Currently, the provider has no mechanisms to adequately deal with utilization approaching maximum capacity. All incoming jobs are accepted as long as there is capacity left. Because demand for the different services vary greatly situations where one type of resources (e.g. CPU) operate at nearly full capacity while others still have a significant buffer left arise regularly.

Scenario II: A company Y has a lot of computationally intensive tasks in their business processes. They have built and maintain a number of large datacenters providing different services for their own business processes. Due to the nature of their business utilization of the resources is very uneven. The company has accumulated important knowledge in the management of their infrastructure and can provide the services efficiently. Providing these services has become a core competence for the company. Therefore, they consider offering these services to external customers as a new product. This allows them to enter a new business segment, achieve a more even utilization of their resources and at the same time take further advantage of economies of scale. However, the sale of services to external users must not have any negative impact on their own business processes. Especially, it should not result in decreased resource availability for internal users. Because of this, jobs users from within the organization should always be preferred when deciding which jobs are accepted.

Scenario III: The service provider offers a wide range of cloud services for different areas of application, such as data mining, numerical simulations or rendering. Like in the first scenario the profile of resource requirements varies greatly between these services. However, in contrast to scenario I the service provider has more time to take a decision whether to accept a job or not, since resources are reserved in advance. Furthermore jobs are significantly larger and cost more, justifying the introduction of more complex optimization methods. We assume that the provider has not the option to dynamically change his prices. He rather has fixed prices like Joyent.com offering a subscription model or Flexiscale.com with a pay-as-you-go model. An important aspect is to consider the time of arrival, when jobs come into the system. The request time for a service has a great impact on the forecasting models and the revenue expectations. We assume that large jobs are often well planned and scheduled more in advance than small jobs.

2.2 Requirements

From these scenarios the research question arises "What pricing mechanisms can be used to increase provider revenue?". These mechanisms have to consider different circumstances such as real-time decisions or further constraints like a strict minimum capacity available for internal users. Naturally there are myriads of options whose study covers a very wide research area. In this paper we investigate mechanisms, which try to increase revenue by accepting or denying service requests. We consider fixed price situations as well as simple pay-as-you-bid mechanism. We restrict our analysis to these types of mechanisms as they are relatively simple to implement in real world systems.

Scenario I imposes some requirements on a mechanism. To effectively deal with this scenario a mechanism must have very low computational cost; it simply needs to calculate the decision within a very short time span. Furthermore, the mechanism should take utilization of each resource type into account, when deciding whether or not a job is accepted. For this scenario we assume the application of a pay-as-you-bid mechanism. Scenario II adds the need to guarantee one class of users a certain capacity or implement some kind of prioritization of some users. The requirements for the third scenario are slightly different. In this case there is more time to take the decision. Therefore a mechanism can do a more complex optimization as long as it leads to higher revenue. Scenario III uses a fixed price mechanism. Using the requirements of all three scenarios it is obvious that it would be very difficult - if not impossible - to find a single mechanism which excels in all situations. Therefore we present two different mechanisms which are deemed promising to work in our scenarios. There exists no one-size-fits-all-mechanism. Efficiency is widely used as a requirement and metric. However, Cloud providers' main focus is on revenue not efficiency. Since they decide which mechanisms to implement we focus on revenue optimization in this paper.

3 RELATED WORK

There are two main streams of work existent that deal with the problem of admitting job requests. The first one considers technical and rule based optimization. The second one investigates the use of concepts from Revenue Management.

Technical and rule based optimization stems from the computer science perspective, as we will see in the following: Ferguson, Nikolaou, Sairamesh, and Yemini (1996) discuss the application of economic theories to resource management. Aiber, Gilat, Landau, Razinkov, Sela, and Wasserkrug (2004) present an architecture for autonomic self-optimisation 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, Napoli, and Piglione (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 behaviour. An architecture for admission control on e-commerce websites that prioritizes user sessions based on predictions about the user's intentions to buy a product is proposed by Poggi, Moreno, Berral, Gavaldà, and Torres (2007). Boughton, Martin, Powley, and Horman (2006) present research on how workload class importance should be considered for low-level resource allocation.

One approach to realize advance reservation and 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. 2007). 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 economic factors such as pricing.

As resources, e.g. processing power, storage and bandwidth, are usually bundled as services in the Cloud, providers have to plan their resource usage carefully and be aware of dynamic changes of the incoming request. Although, advance reservation diminishes the unpredictable requests, users can still submit important jobs on-demand and cancel queued jobs or their reservation (no-shows). The provider can apply revenue management strategies to enhance revenue and optimally allocate his resources to the consumers. He can set booking limits for his services and accept a certain amount of customers. An acceptance strategy is required due to the competition for the resources by the services. The resources in a cluster are accessed by different services. It is important to know which service request should be accepted now to gain enough revenue in the future. The described problem is an instance of the dynamic inventory network capacity control (NCC) for finite time horizon *T*. Therefore work about Revenue Management concepts for IT is a key part of our related work.

The first paper analyzing Revenue Management concepts for cluster systems was published by Dube et al. (2005). In the suggested model one resource is offered 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.

Cancelations and no-shows reduce the efficiency of resource usage. Sulisto et al. (2008) analyzed how overbooking strategies can be applied to maximize revenue. Different prices were charged for one resource. Three overbooking policies were implemented and compared via simulation. The benefits of overbooking for shared hosting platforms were emphasized by Urgaonkar (2002) as well. He did not optimize the revenue by classifying different services, but only the throughput rate.

Anandasivam and Neumann (2009) presented a framework for applying Revenue Management in Grid Computing. They gave an introduction and outlined some requirements, which have to be fulfilled. Their theoretic model comprises bundles of resources and shows how they can be priced. This framework can be adapted for Cloud Computing as well. However, none of these papers analyzes capacity planning strategies of resource bundles via simulations.

Nair and Bapna (2001) introduced Revenue Management concepts for a similar application domain: Internet Service Provider. The provider has to decide whether to accept an incoming customer or to reject. The application domain is different from Cloud Computing as it does not take advance reservation and bundles into account. Customers can only instantly get an internet access.

Policy based decision modeling and capacity planning for fixed prices and advance reservation have been partially studied. Nevertheless, in the context of cloud computing, requirements are different than the domains studied in the literature before. The settings were already outlined in the scenarios. In the next section we present our own model comprising QoS and client classification for online allocation and decision support for advance reservation.

4 THE MODEL

One of the key questions for resource providers is whether a job should be accepted at the time and for the price for which it was submitted. The standard approach used nowadays is 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, since our main objective is to maximize the revenue of the provider those jobs that achieve this objective should be chosen. This can be done by solving the following maximization problem:

$$\max \sum_{i \in J} p_i * x_i$$

Subject to:.

$$\sum_{i \in J}^{n} c_{ir}(t) * x_i \leq gc(t)\ \forall t \in T, \forall r \in R,$$

Where T is the set of all timeslots; J is the set of available job requests; p_i is the price paid for job i; x_i is a binary allocation variable indicating whether job i was accepted or rejected; $c_{ir}(t)$ is the capacity required by job i in timeslot t; and $gc_r(t)$ is the total capacity available of resource r during timeslot t.

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, to be more specific the temporal knapsack problem. As such it is NP-hard and therefore computationally intractable. Furthermore to solve the problem, it is necessary to have the information on the future jobs, the capacity and runtime they require, the prices, etc. In general this information is not available before a job is submitted.

We propose two approaches to solve this problem. The first approach uses policies as a heuristic. This approach has very low computational costs, decisions can be calculated very fast and useful policies can be already generated with very limited information about future demand. Therefore this approach can be used in a very wide range of scenarios, even real-time scenarios where decisions have to be taken immediately. The second approach introduces more complex mechanisms used in classic revenue management. It requires more time and resources for calculations than the first approach as well as a forecast for the demand. Therefore it cannot be used in such a wide range of scenarios as the first approach. In return, however, in those scenarios where it can be used it promises higher revenue than the first approach.

4.1 Policy based heuristic

The first approach is based on previous work presenting an earlier version of an architecture for an economically enhanced resource manager (Püschel et al. 2007). 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 Service Level Agreements (SLA), company policies and information gained from historic workload traces, utilization curves, prices, etc. If available, such policies can also be based on forecasts or information about the statistical distribution of job size, runtime and prices. This approach is especially well suited for motivational scenario I and II.

In this work we will extend two aspects of our policy based approach. The first aspect is dynamic pricing based on various factors, such as projected demand or utilization. The second aspect is job priority based on client classification. 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.

In scenarios where various services with different profiles of resource requirements are offered the (reservation) price can be calculated based on the overall utilization or on the utilization of each resource type (e.g, CPU, memory, storage). If prices are calculated based on utilization of the individual resource types those services that need a lot of capacity of scarce resources have a higher price increase than services which only need little. Such a pricing mechanism reduces the risk that some resources have a lot of free capacity but no jobs can be accepted because one type of resources is used to capacity.

For certain users it may be very important to always have access to the resources. This is the case in our third motivational scenario; sufficient resources have to be available for jobs from internal users. To achieve this a type of client priority on job acceptance can be used. 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 ore internal users are preferred. There can be two types of priorities: strict priorities and soft priorities. *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. Therefore clients can still submit jobs when the threshold is reached as long as they pay a higher price.

To keep the system adaptable it includes a Policy Manager, a component which stores and manages the different policies, such as policies concerning pricing and client classification. Policies are formulated in the SWRL, the Semantic Web Rule Language (Horrocks et al. 2004). SWRL combines the Web Ontology Language (OWL) and the Rule Markup Language (RuleML) and is de facto standard concerning rules languages that offer semantic integration. A simple example rule 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 5 on the general reservation price:

Utilization(*?utilization*) ^ *InsideUtilizationRange*(*?utilization,* ”71% – 100%”)

=> *SetSurcharge*(*?utilizationsurcharge,* ”5”)

The overall architecture for an economically enhanced resource manager is implemented in a prototype in an EU project; the extensions presented in this work are implemented in a simulator.

4.2 Bid price optimization

4.2.1 General description

Each offered service represents a booking class, which has a fixed price. The provider has to decide, if a service request should be accepted or denied. Thus, a limit defining how many requests are operable for each booking class has to be identified, which is known as capacity control. Nested booking limits allow preventing that bookings for services with high revenue are being rejected in favor of bookings with low revenue. They define how much capacity is reserved for a certain booking class. Every service has limited access to the resources like CPU, storage or bandwidth. Due to multiple resources a nested booking limit control must be defined for each resource. This is called virtual nesting control (Talluri and van Ryzin 2004). It is difficult to forecast demand appropriately for virtual classes. The requirement of mapping services to virtual classes increases the complexity. Furthermore, the assumption that demand for low-class services occurs earlier than high-class products is quite static, and when demand is stochastic the strict low-to-high order is less appropriate. It is assumed that customers book the services for usage at a certain time in the future. The low-before-high arrival is expressed by a high booking probability of low-fare services at the beginning of the booking period. This probability decreases during the time period. Contrarily, the probability of booking high-fare services is low at the beginning of the booking period, and increases until the end of the time period.

For the application of bid-price controls, at any point in time a simple threshold value for each resource has to be stored. Bid prices are interpreted as an approximation for the opportunity cost (Bertsimas and Popescu 2003) for reducing the resource capacities, which are needed to satisfy incoming service requests. Moeller et al. (2007) describes bid prices as monetary values of single capacity units, and the resource demands of a request weighted with the corresponding bid prices must be summed. If this sum exceeds the revenue yielded by the sale of one unit of the respective product, the request is rejected, otherwise it is accepted. Regular updating of bid-price values is necessary to guarantee a continuous precision of the bid prices. Less accurate bid prices can lead to accept/reject decisions of minor value. Continuously updated bid prices are based on the current booking situation at a certain point in time $t \in \{T, T\text{-}1, \ldots, 0\}$. That is, if a high amount of capacity has already been sold, the bid prices turn out to be higher.

The decision of accepting or denying a request depends on the policy how the decision is made. Capacity control and dynamic pricing known from Revenue Management are an instance of linear programming models. Since speed of computation matters, bid price control is an approximation method to define policies very fast. It provides a good estimate, but not always an optimal solution. Especially in the NCC setting, the calculation of the optimum increases exponentially with the number of resources m and products n (Talluri and van Ryzin 2004). In the following we assume that the provider has three resources $h \in \{1\ldots m\}$ with m=3 available and offers three services $i \in \{1\ldots n\}$ with n=3. For example resources are CPU, storage and bandwidth, whereas services can be low medium and high instances like the Amazon Web Services. Resources have to be quantifiable. The usage of resources by the services is illustrated by the matrix A. The element a_{hi} represents the usage of resource h by one unit of service i. A_i shows all resources consumed by service i and A^h the services using resource h. The total amount of capacity for each resource h is given by c_h. For a certain time t the reserved capacity of resource h is $\bar{c}_{ht}$. Selling one unit of service i yields a revenue of r_i. The decision for accepting a request is based on the bid price π_{ht} for resource h at time t. $\bar{\pi}_h$ is the base bid price for resource h at beginning of the booking period T. At t the entire demand D_{iT} arriving in T for service i is divided into the demand arriving between the current time t and the end of the booking period, which is defined as D_{it} and the already known demand D_{iT} - D_{it}.

4.2.2 Self-adjusting bid prices

The idea of self-adjusting bid prices (SA-BP) is to compute bid-price functions for resources based on the amount of capacity already reserved as well as the expected demand-to-come (Klein 2007). Unlike the approaches described above SA-BP uses simple linear functions with parameters which can be easily kept track of during the booking period. The concept involves the determination of coefficients (control variables) via simulation-based optimization that are used for calibrating the bid-price functions adequately. In SA-BP two different linear bid-price functions are proposed. One is time-oriented and involves the time remaining to sell the products, and the other one is resource-oriented. However, the focus here is on the resource-oriented alternative, because it takes into account information about the future resource demands of incoming requests, and thus, can describe the current booking situation more accurately. The bid prices are calculated for each resource h every time a request occurs. The components of the bid-price function describe the current booking situation. Consequently, if many requests arrive the bid-price computations take place more frequently. Therefore, the accept/reject decisions directly affect the values of future bid prices. The formula of the bid-price function is: $\pi_{ht} = \bar{\pi}_h + \alpha_h \cdot \bar{c}_{ht} - \beta_h \cdot u_{ht}$.

$\bar{\pi}_h$ denotes the base bid price, which is the bid price of a resource h at the beginning of the booking period. It has to be specified in advance. Naturally, the choice of $\bar{\pi}_h$ influences further bid prices. The choice of the base bid prices are quite challenging. Precalculated random numbers as well as upper and lower bounds to calculate different sets of solutions including $\bar{\pi}_h$ within the simulation-based optimization are a solution proposed by Klein (2007). The base bid price can also be interpreted as the (initial) cost of reserving one unit of a particular resource.

The bid-price function is based on two parts. The first part ($\alpha_h \cdot \bar{c}_{ht}$) is responsible for the increase of the respective bid price over time. $\bar{c}_{ht}$ is the amount of capacity of a resource h reserved at time t. If a request for a service i is accepted, the bid price of a resource increases by the delta of the value $\bar{c}_{ht}$, which equals a_{hi} capacity units, and on the value of α_h . Thus, the bid price of a resource h only is only increased by the acceptance of incoming requests. This corresponds to the fact that the amount of available resources decreases due to sales and hence become more expensive.

The second part of the formula ($\beta_h \cdot u_{ht}$) decreases the bid price for every occurring request. A decrease is required because if some requests are rejected, some future requests can be accepted again. If there was no decrease in the function, the bid price only would rise, and from a certain point every future request would be rejected, and no more sales could take place. u_{ht} is the capacity required to satisfy the demand for the products $i \in A_h$ until t. The demand until t for a product $i \in A_h$ can be calculated by $\overline{D}_{iT} - \overline{D}_{it}$. It requires forecasts of the total demand per product i (D_{iT}), as well as forecasts of the demand-to-come (D_{it}) for every point in time t until the end of the booking period. u_{ht} is calculated by

$$\sum_{i \in A^h} a_{hi} \cdot (\overline{D}_{iT} - \overline{D}_{it})$$

For every product its demand for resource h a_{hi} is multiplied with the expected demand until t $\overline{D}_{iT} - \overline{D}_{it}$, and the sum of these products is taken. The values of u_{ht} increase over time as more demand is realized. The bid-price decreases with time proceeding. The two parts of the resource-oriented bid-price function make sure that the total value of a bid price π_{ht} increases only, if a request is accepted. The increase amplifies if the amount of reserved capacity c_{ht} is high.

The parameters α_h and β_h attach importance to variables. They have a strong impact on the accept/reject decisions. For instance, very high values for coefficient α_h and very low values for coefficient β_h lead to more frequent reject decisions because the increase of the bid-price function turns out too high. This would imply losses in revenues due to rare sales. In the opposite case, very low α_h values and very high β_h values result in an "accept all" policy, and thus, it can happen that capacity is mostly sold to low-fare classes (requests for low-level services) also leading to potentially lost revenues. Because of these reasons promising values for the control variables should be obtained by simulation-based optimization.

An enhanced approach to adopt the bid-price function is to artificially raise the bid prices depending on the current degree of reservation. Thereby, low-class requests occurring at the beginning of the booking period can be rejected explicitly in order to reserve capacity for requests yielding higher revenues. This kind of policy can be realized using a set of variables γ_i which depend on the average of the degree of reservation denoted by z_t calculated each time a request occurs by

$z_t = \frac{\sum_h z_{ht}}{h}$. Given z_t the provider can set utilization thresholds which specify when to use a specified γ_i, e.g. as follows:

- If $z_t < 0.4$: use values (0.12, 0.02, 0.01) for γ_i, $i \in \{1,2,3\}$.
- If $0.4 \leq z_t < 0.75$: use values (0.02, 0.025, 0.05) for γ_i, $i \in \{1,2,3\}$.
- If $0.75 \leq z_t \leq 1$: use values (0.005, 0.05, 0.1) for γ_i, $i \in \{1,2,3\}$.

γ_i is added to control variable α_h, thereby increasing the bid prices.

$$\pi_{ht} = (\bar{\pi}_h)^{z_{ht}} \cdot (\alpha_h + \gamma_i) - \beta_h \cdot \frac{u_{ht}}{U_{hT}}$$

In the case when demand strongly fits to the assumptions stated above, this contributes to more frequent rejections of service-1 requests at the beginning of the booking period, and to more frequent rejections of service-3 requests close to the end of the booking period.

5 EVALUATION

For the evaluation we considered the following three types of services with requirements for processing power, memory, and storage according to table 1. For each of the proposed mechanisms a scenario matching the requirements from the motivational scenarios was created and a simulation was run. The same service types and their resource profiles were used in both cases. The first simulation was based on real workloads, for the second mechanism a very challenging scenario using fixed prices fixed runtimes and relatively constant rates of arrival were used.

Service	CPU	Memory	Storage	Mean Price/Fixed Price
1	4	4	16	16
2	8	16	4	24
3	16	8	8	30

Table 1. Types of services and their resource profiles.

5.1 Policy based heuristic

The approach was evaluated using generated workloads as well as workloads based on real world scenarios, from the Parallel Workload Archive (Feitelson 2007). The setting of the evaluation matches the requirements for the first motivational scenario. The runtimes and offsets between submit and start time were generated using a lognormal distribution. For the workloads based on real world scenarios, 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. 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. Job runtimes where rounded down to full hours to allow a timeslot based allocation. After filtering invalid jobs and those jobs which were considered too long or too short 566,701 jobs were left and used for the scenario. For each of these jobs a service type was randomly drawn. Based on these jobs ten joblists with different prices were generated. A capacity of 8200 for each of the resource was used to be able to accommodate some of the larger jobs from the workload. A simple pay as you bid mechanism was assumed and pricing information was generated using a normal distribution with the mean price from table 1 and variance of 0.5.

Policy	FCFS	Utilization	Strict Priority	Soft Priority
Avg. Revenue	54190907,26	88697433,21	80480832,63	82579338,5
Percentual Increase		63,38%	48,51%	52,39%

Table 2. Revenue increase using policy based approach.

As can be seen in table 2 the average revenue the benchmark, a simple first-come-first-serve (FCFS) policy, was 54190907,26. A policy using utilization based pricing managed to increase the revenue on average by 63,38% resulting in an average revenue of 88697433,21. Introducing strict priority without charging preferred users for the privilege obviously reduces the achieved revenue. However since the preferred users in our scenario are internal users it is reasonable that they get the privilege without a charge. The average revenue for a policy combining utilization based pricing with strict priority we achieved in our simulation is 80480832,63 which is an increase of 48,51% compared to the FCFS policy. Since soft priority allows outbidding preferred clients revenue is higher than in a policy using strict priority, however this results in slightly decreased resource availability for preferred users. A policy with soft priority and utilization based pricing resulted in a revenue increase of 52,39% compared to FCFS. Runtime for one simulation run with 566,701 job requests varied between 55 and 60 seconds, each job is accepted or rejected within fragments of a second. This makes the mechanism suitable for scenarios where job acceptance has to be decided in less than a second.

5.2 Bid price optimization

For the evaluation of bid price optimization we assumed the setting in scenario II. Instead of prices drawn from a normal distribution the prices were set according to table 1. Naturally, this leaves significantly less room for revenue increase compared to scneario I. Thus the bid price approach focuses more on forecasting and capacity planning. The bid prices are in every timeslot. Parameters like the current amount of capacity reserved, the total expected demand and the already realized demand are taken into account. For each incoming request at time t, the bid prices are updated. Bid prices increase, if the amount of reserved capacity rises due to the acceptance of requests. They decrease, if incoming requests are rejected and no additional capacity is being reserved. Given the bid prices, an approximation of the opportunity cost is calculated by weighting the resource demand of an incoming request with the bid prices. If the revenue yielded by the sale of one unit of the respective service exceeds this approximation, the request is accepted. This simulation setting differs from the policy-based approach described above, since the bid price optimization considers advance reservation of resources, while in scenario I resources are instantly consumed. The capacity of a resource provider was fixed to a maximum of 750 units for every resource. The simulation was performed with a booking period of length T=2000 for five job files. The SA-BP model yielded an average revenue of 1860.5. The average revenue of a FCFS policy was 1810.08 (increase of 2,79%). In the best case the revenue increase was 11.26%. The performance of SA-BP is based on the accuracy of the forecasting data. With inaccurate forecasting models, the revenue of SA-BP can even be lower than the FCFS. The runtime of SA-BP is similar to the FCFS computation. This is due to the heuristic approach of SA-BP. More complex revenue management models require long runtimes to calculate the optimal allocation. In both policies the runtime for 2000 jobs were about 6 seconds. In practice, a longer time horizon is necessary for calculation, which will increase the runtime exponentially.

The reason why SA-BP performed better than FCFS was due to rejections of most service-1 requests in the first third of the booking period. Thus, more capacity was kept free for later arriving service-2 and service-3 requests. The SABP3 policy accepted more service-1 requests in the second third of the booking period. Furthermore, the acceptance of service-3 requests decreased over time, thus enabling more accepts of later occurring service-1 requests. The FCFS policy accepted all incoming requests in the first third of the booking period until the capacity was completely reserved. SA-BP rejected early arriving service-1 requests, and thereby, later arriving requests for services 2 and 3 could be accepted.

5.3 Discussion

The evaluation shows that the proposed mechanism can increase revenue. The policy based mechanism fulfils all requirements imposed by the first two motivational scenarios. Revenue is significantly increased and computational costs are very low. The concept of strict priorities enables Cloud providers to guarantee availability of a predetermined amount of capacity for certain users. Unless these users are charged for this privilege it results in a revenue decrease compared to the pure utilization based pricing. Soft priorities can be used to mitigate this effect; however, they can result in a reduction of resource availability for preferred users. The bid prize optimization mechanism can be used to increase revenue in motivational scenario three. Even in a challenging setting with fixed prices and the same profit margin for all services it still managed to deliver revenue increases. Computational costs are higher than for the policy based approach but still acceptable for many scenarios.

6 CONCLUSION

Cloud Computing is a promising approach with a high impact on business models. One aspect of a business model is revenue and how prices should be set to achieve predefined revenue goals. The decision about accepting or denying requests has a high impact on the revenue of the provider. Revenue Management concepts known from the airline comprise interesting policies. In this paper we analyze two approaches. Predefined policies allow increasing revenue compared to static models like first-come-first-serve. Such policies include utilization based pricing as well as client classification.

We furthermore present how to apply more complex revenue management concepts in Clouds. When price are relatively fixed, an optimization of the capacity over bid price concepts seems to be a promising approach. A bid price optimization enables to determine the minimum price a consumer has to pay for requesting a service. The complexity arises when services utilize the same resources. The bid price is applicable for a scenario, where advance reservation is possible. Our proposed policies and models were validated via simulation. For the first of our proposed mechanisms we were able to use real world workload traces. By using these real workload traces the models are applicable in practice to some extent. However we still had to generate pricing information, therefore complete external validity of our results cannot be assumed. We showed that well-defined pricing strategies can notably increase revenue. Future work will comprise an implementation in existing middleware considering architectural design and technical feasibility. Moreover, the heuristics and policies have to take learning algorithms into account to analyze, whether it will lead to better results.

References

Aiber, S., D. Gilat, A. Landau, N. Razinkov, A. Sela, and S. Wasserkrug (2004). Autonomic self-optimization according to business objectives. In Proceedings of the First International Conference on Autonomic Computing (ICAC'04), , pp. 206–213. IEEE Computer Society.

Anandasivam, A. and D. Neumann (2009). Managing revenue in Grids. In Hawaii International Conference on System Sciences, Proceedings of the 42nd Annual. Springer-Verlag GmbH,.

Bertsimas, D. and I. Popescu (2003). Revenue Management in a Dynamic Network Environment. Transportation Science, vol. 37, no. 3, pp. 257–277.

Boughton, H., P. Martin, W. Powley, and R. Horman (2006). Workload class importance policy in autonomic database management systems. In POLICY '06: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06), Washington, DC, USA, pp. 13–22. IEEE Computer Society.

Buyya, R. (2002). Economic-based Distributed Resource Management and Scheduling for Grid Computing. Ph. D. thesis, Monash University.

Carr, N. G. (2005). The end of corporate computing. MIT Sloan Management Review 46 (3), 32–42.

Chicco, G., R. Napoli, and F. Piglione (2006). Comparisons among clustering techniques for electricity customer classification. IEEE Transactions on Power Systems 21 (2), 933– 940.

Dube, P., Hayel, Y. and L. Wynter (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, vol. 4, no. 1, pp. 24–38.

Feitelson, D. G. (2008). Workload Modeling for Computer Systems Evaluation.

Ferguson, D. F., C. Nikolaou, J. Sairamesh, and Y. Yemini (1996). Economic models for allocating resources in computer systems. pp. 156–183.

Foster, I., C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy (1999). A distributed resource management architecture that supports advance reservations and co-allocation. In Proceedings of the 7th International Workshop on Quality of Service (IWQoS 1999), London, UK, pp. 62–80.

Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean (2004). Swrl: A semantic web rule language combining owl and ruleml. w3c member submission.

Kenyon, C. and G. Cheliotis (2004). Grid resource commercialization: economic engineering and delivery scenarios. Grid resource management: state of the art and future trends, 465–478.

Klein, R (2007). Network capacity control using self-adjusting bid-prices. OR Spectrum, 29(1):39–60.

Kounev, S., R. Nou, and J. Torres (2007). Autonomic qos-aware resource management in grid computing using online performance models. In The 2nd International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2007), Nantes, France.

Moeller, A., Roemisch, A. and K. Weber (2008). Airline network revenue management by multistage stochastic programming. Computational Management Science, vol. 5, no. 4, pp. 355–377.

Nair, S. and R. Bapna (2001) An application of yield management for Internet Service Providers. Naval Research Logistics, vol. 48, no. 5, pp. 348–362.

Newhouse, S., J. MacLaren, and K. Keahey (2004). Trading grid services within the uk e-science grid. Grid resource management: state of the art and future trends, 479–490.

Poggi, N., T. Moreno, J. L. Berral, R. Gavald`a, and J. Torres (2007). Web customer modeling for automated session prioritization on high traffic sites. In Proceedings of the 11th International Conference on User Modeling, Corfu, Greece.

Püschel, T., N. Borissov, M. Macías, D. Neumann, J. Guitart and J. Torres (2007). Economically Enhanced Resource Management for Internet Service Utilities. In Proceedings of the 8th International Conference on Web Information Systems Engineering WISE(2007)

Sulistio, A., Kim, K. and R. Buyya (2008). Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue. In Proceedings of the 2008 8th International Symposium on Cluster Computing and the Grid (CCGRID). IEEE Computer Society, 2008.

Talluri, K. and van G. Ryzin (2004). The Theory and Practice of Revenue Management. Berlin, 2004.

Urgaonkar, B., Shenoy, P. and T. Roscoe (2002). Resource overbooking and application profiling in shared hosting platforms. In OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation. New York, NY, USA, pp. 239–254.