2009 IEEE Conference on Commerce and Enterprise Computing

A Heuristic Approach for Capacity Control in Clouds

Arun Anandasivam*, Stefan Buschek*, Rajkumar Buyya†
* Institute for Information Systems and Management
University of Karlsruhe, Germany
Email: anandasivam@iism.uni-karlsruhe.de
stefan.buschek@student.kit.edu
† Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
raj@csse.unimelb.edu.au

***Abstract*—Cloud resource providers in a market face dynamic and unpredictable consumer behavior. The way, how prices are set in a dynamic environment, can influence the demand behavior of price sensitive customers. A Cloud resource provider has to decide on how to allocate his scarce resources in order to maximize his profit. The application of bid price control for evaluating incoming service requests is a common approach for capacity control in network revenue management. In this paper we introduce a customized version of the concept of self-adjusting bid prices and apply it to the area of Cloud Computing. Furthermore, we perform a simulation in order to test the efficiency of the proposed model.**

I. Introduction

Since the idea of Grid Computing came up in 1998 by Foster et al. [1], the development of the infrastructure was mainly driven by scientific applications. Traditional resource management techniques focus on maximizing throughput or minimizing waiting time in a queue. In the recent years, markets for computer utility have become popular. Systems like Nimrod-G [2], Bellagio [3], Tycoon [4], Mirage [5], Gridbus [6] or the SORMA Open Grid Market [7] have stressed the usage of market mechanisms as an efficient way to allocate resources among the participants. While the former are more resource-centric approaches, markets try to optimize the usage from the users' perspective and to utilize resources in off-peak time. Since users in a distributed network act in a self-interested manner, the design of incentives is crucial for increasing efficiency in a network [8].

Economic concepts help to set incentives for resource owners to provide their resources. Cloud service providers offer their services similar to Grid Computing resources. However, contrary to the mainly scientific driven Grid scenarios, Clouds providers have to define Service Level Agreements and apply business models [9], [10]. The virtualization technologies in Clouds allow to define the exact resource usage for one product of a Cloud provider. Moreover, Grid participants are contemporaneously consumer and provider, whereas Cloud providers and consumers can be clearly distinguished. In Clouds resellers like Jollat[1] or RightScale[2] come into play by enhancing standard services from Amazon[3] with new services [11], [12].

In a market Cloud providers face dynamic and unpredictable consumer behavior. The way, how prices are set in a dynamic environment, can influence the demand behavior of price sensitive customers [13]. Consequently, customers with a low valuation for a service would use it in a cheap period. Business customers are willing to pay a higher amount for the usage. By identifying the right price for a customer and a requested service at a certain point in time, higher revenues can be achieved [14]. However, in some settings, it is difficult to change prices over time. For example, Amazon offers for its Elastic Cloud Computing[4] service a fixed price of $0.10 for a CPU hour without frequently changing the prices. Price changes can be realized on specific markets like auctions for computing resources like the solution from Zimory[5].

In this paper, we present a decision concept for a provider known from Revenue Management to accept or deny incoming requests for services in order to increase revenue in a scarce resource market. A provider offers several services, which use the same resources from his cluster. From his point of view he is interested in selling the more expensive services [15]. When a buyer requests a service with low revenue, the provider has the possibility to accept this request or to wait for prospective customer asking for high valued services. We analyze different decision rules well known from the Airline Industry and show how to apply Revenue Management concepts to Cloud Computing. Our contribution comprises a more efficient decision rule called *customized bid price policy*. The efficiency is proved via simulation.

[1] http://www.jollat.com/
[2] http://www.rightscale.com/
[3] Amazon Web Services (http://aws.amazon.com)
[4] A small Linux instance (http://aws.amazon.com/ec2)
[5] http://www.zimory.com

978-0-7695-3755-9/09 $25.00 © 2009 IEEE
DOI 10.1109/CEC.2009.20

II. Related work

A. Revenue Management for Cloud Computing

In Clouds, resources e.g. processing power, memory, storage, and bandwidth, can be bundled as services, which are offered to other Cloud users. Providers have to plan their resource usage carefully and be aware of dynamic changes of the incoming requests for their services. Examples of services on the infrastructure level (also known as Infrastructure as a Service) are Amazons Web Services like EC2, S3 or Simple DB. All of them are based on pure resources like CPU or storage. Although, advance reservation diminishes the unpredictable requests, users can still request important services on-demand and cancel pending requests or reservation. 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 (Figure 1).

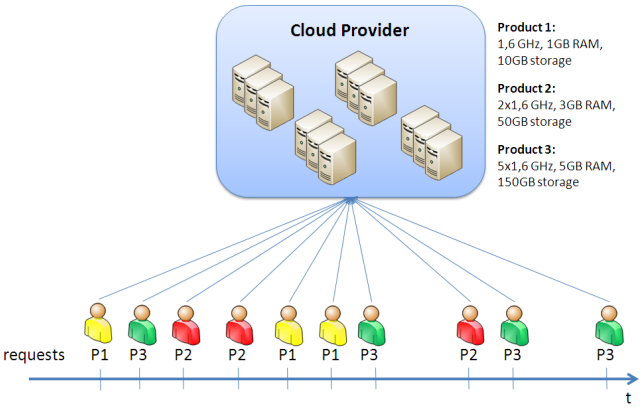

Fig. 1. Incoming requests for different services in different timeslots

The first paper analyzing Revenue Management concepts for cluster systems was published by Dube et al. [16]. In the suggested model one resource is offered at 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. Although the authors state that "in an on demand operating environment, customers and jobs, or service requests arrive at random", the behavior of price sensitive customers can be influenced by offering different prices for the same product, which in turn reduces the randomness [13], [17].

Cancelations and no-shows reduce the efficiency of resource usage. In [18] Sulisto et al. analyzed how overbooking strategies can be applied to maximize revenue. Different prices were charged for one resource and three overbooking policies were implemented and compared via simulation. The benefits of overbooking for shared hosting platforms was emphasized by Urgaonkar [19] as well. He did not optimize the revenue by classifying different services, but only the throughput rate.

Anandasivam and Neumann [20] 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. However, none of these papers analyzes capacity planning strategies of resource bundles via simulations.

Nair and Bapna [21] introduced Revenue Management concepts for a similar application domain of an Internet Service Provider. The provider has to decide whether to accept an incoming customer request or to reject it. 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.

B. Bid price control in Revenue Management

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 to prevent 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, memory, storage, or bandwidth. Due to multiple resources a nested booking limit control must be defined for each resource. This is called virtual nesting control [22], [23]. It is difficult to forecast demand appropriately for virtual classes. The requirement of mapping services to virtual classes increases complexity. Furthermore, the assumption that demand for low-class services occurs earlier than for high-class services is typical in Revenue Management. If demand arrives in a strict high-to-low order the provider simply needs to accept customer requests in a first-come-first-serve order to maximize his revenue. On the other hand when demand is stochastic the strict low-to-high order is also less appropriate. Therefore, we assume that demand for low-class services is more likely to arrive earlier and demand for high-class service is more likely to arrive later in time.

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 of the opportunity cost [24] of reducing the resource capacities, which are needed to satisfy incoming service requests. [25] 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 [22]. 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. That is, if a high amount of capacity has already been sold, the bid prices turn out to be higher.

Although prices which are charged from the customers are fixed for the booking period, bid price policies can be seen as some kind of (dynamic) pricing from the provider's interior point of view. The parameters of the bid price policy can be adjusted for subsequent booking periods due to changes in demand. In [26] linear functions are introduced to compute bid prices for each arriving request at time t. The essential requirements of Dynamic Pricing functions in [27] can also be transferred to bid price functions:

- Flexibility: The bid price function has to be configurable in an easy way to enable changes in the accept/reject decision policy by the provider.
- Fairness: Naturally, higher bid prices are attached to more expensive resources, and the opportunity costs of reserving resources of services with high resource demands are higher.
- Dynamic: Bid prices should not be static thresholds. To guarantee a certain precision they need to be adjusted to the current booking situation.
- Adaptability: Fluctuations in supply and demand have to be considered in calculation of bid prices.

III. Model

The decision of accepting or denying a request depends on the policy which applies the decision rule. Capacity control and dynamic pricing known from Revenue Management are an instance of linear programming models. Since speed of computation matters (especially for large resource/product settings), bid price control is an approximation method to quickly update the policies after the arrival of new requests. 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 [28].

In the following we assume that the provider has m resources $h \in \{1, \ldots, m\}$ available and offers n services $i \in \{1, \ldots, n\}$. Resources are CPU, memory, storage, and bandwidth, whereas services can be low, medium, and high instances like the Amazon Web Services. Resources have to be quantifiable, and dividable into discrete units. Matrix A describes the usage of resources by the services in the case of four resources and five services, what is illustrated in table I. An element a_{hi} represents the usage of resource h by one unit of service i. The total amount of capacity for each resource h is given by c_h. At a certain time $t \in \{T, T-1, \ldots, 0\}$ the reserved capacity of resource h is $\bar{c}_{ht}$. Selling one unit of service i yields a revenue of r_i. The decision of accepting a request is based on the bid price π_{ht} for resource h at time t as well as on the resource usage of the request: $r_i \geq \sum_{h \in A_i} a_{hi} \cdot \pi_{ht}$ must be fulfilled.

A. Demand

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 denoted as D_{it}, and the arrived demand until t $\hat{D}_{it}$ (see Figure 2).

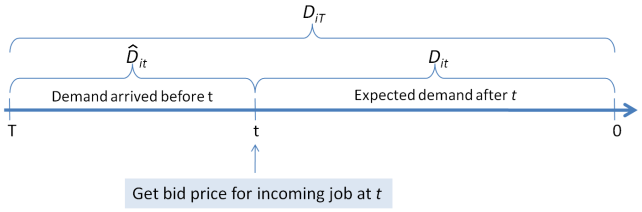

Fig. 2. Demand definition in a finite time

It is assumed that there can arrive at most one service request per discrete unit of time t, as for example in [29]. A request for service i at time t arrives with probability p_{it}, and thus, the arrival of a request for service i at time t is a random variable X_t with $X_t = \{0, i\} \, | i \in \{1, \ldots, n\}$, i.e. $X_t = 0$ if no request comes in at t. T is finite and countable, and thus, the arrival process of requests by the customers is a time-discrete stochastic process X, which is a sequence of random variables X_t. The value of the demand arrived from T until t changes from its previous value $\hat{D}_{it}$ at $t+1$ to $\hat{D}_{it}+1$ at t if a request for service i occurs in time slot t (note that time runs backwards).

It is assumed that customers book the services for usage at a certain time in the future. The low-before-high arrival (cf. [30]) 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.

Naturally, the condition $\sum_{i=0}^{n} p_{it} = 1$ must be valid at each point in time t.

B. Deterministic Linear Programming Model (DLP)

The NCC approximation method for bid prices assumes expected demand information and excludes the stochastic nature of the demand [31], [22]. Based on demand forecasts the expected aggregate demand-to-come $\overline{D}_{it}$ for the remaining booking periods is calculated, and it is assumed that the demand is equal to its mean values. An approximation for the objective-value function V is obtained by:

$$Max.\ V(x) = \sum_{i=1}^{n} r_i \cdot x_i \tag{1}$$

$$s.\ t. \sum_{i \in A^h} a_{hi} \cdot x_i \leq c_h - \bar{c}_{ht} \ \forall h \in \{1, \ldots, m\} \tag{2}$$

$$0 \leq x_i \leq \overline{D}_{it} \ \forall i \in \{1, \ldots, n\} \tag{3}$$

(1) is the objective function, which maximizes the total revenue. The total revenue results from the sum of the prices r_i

charged for each service multiplied with the number of units sold of each service in the booking period x_i. (2) ensures that enough capacity of each resource is available to satisfy the need for capacity by the number of allocated units of the services. (3) guarantees that the number of services sold are not below zero and do not exceed the expected demand-to-come.

The solution vector of the primal problem is discarded, and the variables of the optimal solution of the dual problem are used as bid prices [29] . The optimal solution can have multiple optimal dual bid price vectors. The DLP can either be solved at the beginning of the booking period with the given demand forecast by using static bid prices or by recalculating and updating the bid prices at certain data collection points during the booking period, what is advantageous in order to keep up a certain precision of the bid prices. The main benefit of the DLP model is that it can be solved computational efficiently, which makes it popular for practical applications. Its performance strongly depends on the size of the network as well as on reliability of the demand forecasts. However, this model does not imply any uncertainty in demand.

1) Randomized Linear Programming Model (RLP): The RLP model induces stochastic information. The expected demand as in the DLP case is replaced by a random demand vector D [23]. For instance, in [32] demand is modeled as a Poisson Process. The probability distribution of the demand for each service is used to generate executions of the demand-to-come. The optimal solution of this problem represents a random variable, which provides the approximation to the objective-value function V. The application of RLP leads to significant higher revenue than DLP [33].

C. Certainty Equivalent Control (CEC)

The accuracy of bid price values in the DLP model with updates during the booking period depends on how frequent these recalculations are performed. The most frequent calculation of bid prices is carried out by recalculating bid prices each time a request occurs. An approach called certainty equivalent control [24] extends the concept of bid prices and directly calculates an approximation of the opportunity costs. For this purpose it solves two instances of the DLP problem described above: The first instance solves the initial DLP problem (1) and the second instance also subtracts the amounts of resources demanded by the request from the remaining capacity of the resource

$$c_h - \overline{c}_{ht} - a_{hi}, \forall h \in \{1, \ldots, m\}\,. \tag{4}$$

The approximation of the opportunity cost of service i is then obtained by subtracting the objective function value of instance 2 ($V'(x)$) from the objective function value of instance 1 ($V(x)$). This approximation does not depend on the optimal dual variables, thus, the drawback of multiple optimal dual variables of the linear programming model is eliminated. The CEC policy requires forecasts for the total demand for each service, as well as forecasts for the expected demand-to-come ($\overline{D}_{it}$). One advantage of the CEC policy arises from the numerous and periodic updates of the approximation of the opportunity costs, thereby guaranteeing a certain accuracy. Because CEC is based on the DLP problem, it shares the same disadvantage of only incorporating expected demand and not considering uncertainty of the demand process.

D. Self-adjusting bid prices (SABP)

The idea of self-adjusting bid prices is to compute bid price functions for resources based on the amount of capacity already reserved as well as the expected demand-to-come [26]. Unlike the approaches described above SABP 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. SABP can be interpreted as some type of interior dynamic pricing by the provider, which was derived from the optimal dynamic pricing of a single product [32].

Our approach uses a customized version of the resource-oriented bid price function as proposed in [26], and further is abbreviated with SABP-c. The bid price of resource h at time t is calculated by the formula:

$$\pi_{ht} = (\overline{\pi}_h) + \alpha_h \cdot \frac{\overline{c}_{ht}}{c_h} - \beta_h \cdot \frac{u_{ht}}{U_{hT}} \tag{5}$$

The control variables $\overline{\pi}_h$, α_h, and β_h are determined via a genetic algorithm, which is described in section III-E. $\overline{\pi}_h$ is the base bid price and provides the basis for the bid price calculation. The value of $\overline{\pi}_h$ has a strong impact on the bid price. In [26] the base bid price is calculated by creating a random number and multiplying it with the minimum bid price of resource h. The minimum bid price is the value for which, if it is exceeded, requests for at least one service i are no longer accepted, and is computed by $\pi_h^{min} = min\left\{r_i/a_{hi} | i \in A^h\right\}$. In our approach the genetic algorithm uses the minimum bid price as upper bound for the base bid price.

The bid price function further is based on two parts. The first part ($+\alpha_h \cdot \frac{\overline{c}_{ht}}{c_h}$) is responsible for the increase of the respective bid price over time. $\overline{c}_{ht}$ is the amount of capacity of a resource h reserved at time t relative to the total capacity of resource h. If a request for a service i is accepted, the bid price of a resource increases by the delta of the value $\frac{\overline{c}_{ht}}{c_h}$, and also depends on the value of α_h. Thus, the bid price of a resource h only increases through the acceptance of incoming requests. This corresponds to the fact that available resources get less due to sales and hence become more expensive.
The second part of the formula ($-\beta_h \cdot \frac{u_{ht}}{U_{hT}}$) 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 expected demand until t $\hat{D}_{it}$ for a service $i \in A^h$ can be calculated by $\overline{D}_{iT} - \overline{D}_{it}$, what requires forecasts of the total demand per service i ($\overline{D}_{iT}$), as

well as forecasts of the demand-to-come ($\overline{D}_{it}$) for every point in time t until the end of the booking period. However, it is more intuitive to simply count the requests for each service i until t, what leads to an exact value of $\hat{D}_{it}$ for each i. u_{ht} in our case is calculated by

$$\sum_{i \in A^h} a_{hi} \cdot \hat{D}_{it}. \tag{6}$$

For every service its demand for resource h (a_{hi}) is multiplied with the demand arrived until t ($\hat{D}_{it}$), and the sum of these products is taken. U_{hT} is the capacity of resources which is needed to satisfy the total demand of the complete booking period. It is calculated by $\sum_{i \in A^h} a_{hi} \cdot (\overline{D}_{iT}$, where $\overline{D}_{iT}$ is the forecasted total demand for service i. U_{hT} can also be denoted as the total resource demand. It requires a forecast of the total demand for service i in the booking period. The quotient of u_{ht}/U_{hT} increases over time as more demand is realized, and hence, 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 $\overline{c}_{ht}$ 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 too frequent accept decisions of low-fare requests leading to an inefficient reservation for later arriving high-fare requests, what also implies potentially lost revenues. Because of these reasons, promising values for the control variables are obtained by the genetic algorithm.

The main advantage of self-adjusting bid prices is the very frequent recalculation of bid prices. Thereby, information about the current booking situation is always considered and the bid prices exhibit a certain precision. [26] states that this approach is robust to errors in the forecast. If the realized demand is less than the forecasted one, less capacity units are reserved to satisfy incoming requests. Thus, the increasing part of the bid price function is lower and more requests with lower revenues can be accepted. On the other hand, if realized demand is higher, the bid price increases stronger as more requests are accepted. Furthermore, a strict fragmentation of the services into only two subsets (S_1: accepted, i.e. $r_i \geq \sum_{h \in A_i} a_{hi} \cdot \pi_{ht}$); S_2: rejected, i.e. $r_i < \sum_{h \in A_i} a_{hi} \cdot \pi_{ht}$), is avoided to permanently update the bid prices and to guarantee a certain degree of accuracy.

Although the concept shows some robustness against forecast errors as mentioned above, it does not imply stochastic information about demand. This assumption is not very realistic, and does also concern other deterministic models, such as DLP and CEC. The computational effort which is required for finding appropriate values for the control variables $\overline{\pi}_h, \alpha_h$, and β_h is significantly high, which mainly comes from the necessity for simulation-based optimization. Another weakness of the original model is, that it requires a high number of forecast values. For each point in time and for each product offered, the expected demand-to-come ($\overline{D}_{it}$) needs to be stored in order to be able to calculate the demand until t and the resource demands (u_{ht}) every time a request comes in. However, this can be overcome by simply counting the demand until t for each service i. Furthermore, some bid prices turned out to be negative, and the question of how to interpret negative bid price values comes up.

Note, that in [26] the SABP concept has been developed for managing resources in the context of airlines. Usually, in that case a sale of one flight means a change in capacity of one unit of the respective resource, i.e. a seat on a flight at a certain date. This is a major difference to the change in capacity in the context of resources and services in Clouds, in which a sale of a service means a change in capacities of several units of the resources involved in the service sold. Because of that, the usage of the original bid price function would lead to very high values of the decreasing part of the function in the setting of services in Clouds, and thus, some bid price values turned out to be inappropriate. This is avoided by using relative values in (5).

E. Genetic Algorithm

Genetic Algorithms belong to the class of evolutionary algorithms. They are optimization concepts for searching a solution space of a given problem for reasonable solution values [34]. By considering only a small part of the solution space, a simulation of an evolution is performed using the survival of the fittest strategy. Individuals which are fitter than other individuals have a higher probability of surviving and of still being persistent in the next generation in the evolution process.

A genetic algorithm involves some terminology from biology (cf. [34]), and consists of several elements. These have to be defined depending on the application context and the problem to solve.

The choice of an appropriate chromosome representation, which defines how many genes the chromosome contains, and which values the genes are allowed to obtain, is a key component. The control variables (π_h, α_h, β_h) in (5) are used as genes within the genetic algorithm and are optimized in the evolution process.

It is essential to specify the numerical range for the values of the control variables. As mentioned above, we use the minimum bid price π_h^{min} as upper bound for the base bid price. Thereby, the production of counterproductive values is prevented, as in the case if the base bid price exceeds the price of a service i ($\overline{\pi_h} > r_i$), which would lead to rejecting requests for all services i already at the beginning of the booking period. As a consequence, the genes containing the values of the base bid prices can lie in the interval $\left[0; \pi_h^{min}\right]$. The range for α_h and β_h is set to $[1; 2]$. This is because in some cases values below one have turned out to be too low and

may produce too low bid prices, what can lead to an inefficient accept/reject behavior.

The objective of the genetic algorithm is to find adequate values for the control variables of the bid price function. The genetic algorithm must simulate a complete booking period, which includes the demand expected for the given sales period based on the forecast by the provider.

The genetic algorithm performs the following steps:

1) At the beginning an initial population is created based on the chromosome representation as described above. This generation is numbered with generation 0.
2) In the second step the fitness value of all chromosomes in the population is calculated by the fitness function, which evaluates the expected demand with the gene values as control variables in the bid price function.
3) After steps 1 and 2 the evolution process is started. In each evolution phase the GA's selection operator randomly selects pairs or bigger subgroups of the population of chromosomes for reproduction.
4) The chromosomes selected by the selection function are reproduced, and then they are recombined (crossover) or mutated by the genetic operators defined.
5) Subsequently, some chromosomes of the current population are replaced by the new altered chromosomes, thereby creating a new generation.
6) Given the new generation, the generation enumerator is incremented by one.
7) Steps 2 to 6 are repeated until the maximum number of allowed evolutions is reached.
8) When the evolution process is terminated, the fittest chromosome, i.e. the set of control variables with the highest potential revenue, is taken as input for the self-adjusting bid prices function.

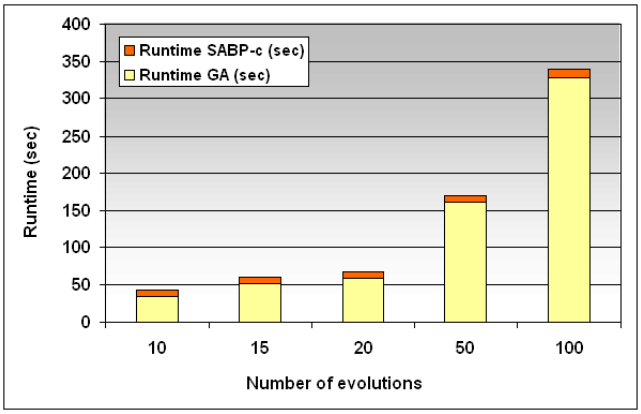

Fig. 3. Runtime analysis with chromosome size of 12 (control variables), population size of 3 and different number of evolution steps

The runtime analysis of the customized SABP policy clearly shows that the main amount of computational effort arises from the genetic algorithm. Naturally, the runtime is dependent on the population size, the number of variables to search for, and the number of evolution steps (figure 3). It is important to state that a longer evolution time does not necessarily lead to a better solution. Moreover, a raise of the population size leads to a longer runtime of the genetic algorithm, but it also adds diversity to the population, what in turn increases the probability of finding a better solution more quickly in terms of the number of evolution steps. Therefore, it is recommended to use a higher population size and a higher number of evolution steps in order to increase the probability of reaching a near-optimal solution.

TABLE I
RESOURCE USAGE BY THE FIVE SERVICES

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
CPU	2	4	3	8	10
Memory	2	8	3	4	10
Storage	8	2	4	4	8
Bandwidth	4	2	8	4	8
Price r_i	18.00	19.50	22.50	27.00	46.00

IV. SIMULATION

A. Setting

There are five different service offered by the provider in the context of this paper (see table I), and the provider has 1500 units of capacity of each resource. The demand for each of these services are independent, that is, a customer who has needs for service-1 will not book a higher and more expensive service. It is assumed that the provider has certain demand information from past booking periods, and is able to perform a more or less accurate demand forecast. Fluctuations in demand will be handled by passing through different demand scenarios in the simulation. We assume that customers who do not use the services booked do not get a refunding. Hence, if customers book a service they will have to pay for it.

TABLE II
DEMAND INTENSITIES OF THE DIFFERENT SERVICES IN EACH SCENARIO

Demand	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
Scenario0	375	184	156	91	108
Scenario1	382	168	149	89	104
Scenario2	378	148	174	91	91
Scenario3	352	154	166	94	86
Scenario4	387	158	160	98	82
Scenario5	376	166	156	93	86
Scenario6	373	169	155	96	73
Scenario7	377	171	148	90	72
Scenario8	369	176	151	95	67
Scenario9	353	156	175	92	64
Forecast	377	162	158	98	92

B. Demand scenarios

According to the arrival process described, numerous demand scenarios were created. A scenario represents an instance of the demand realized in the booking period. Because of the

random arrival of a request for service i the demand scenarios show variation for all services. Furthermore, we assume a booking period with length $T = 1.000$. The simulation is performed over ten different demand scenarios and one additional scenario, in which the realized demand equals the forecasted one. All ten scenarios are read and evaluated by every policy implemented. Each policy requires a forecast of the total demand for each service. The forecast corresponds to the expected demand and is created based on the same distributions as the demand scenarios. Table II comprises the demand intensities for the different services i in each scenario. All scenarios differ in the demand intensities as well as in the time of arrival of all services due to the random arrival of requests. The service provider achieves the highest profit with service $i = 5$. Scenarios 6 to 9 show the greatest variation compared to the forecast. This is done to test the robustness to forecast errors. In addition, table IV contains the results of only CEC and SABP-c in scenarios 10 to 14 in order to demonstrate the dependency of the CEC policy on the forecasted data. In scenarios 10 to 14 the demand intensity of services 1 to 4 exactly equals the forecasted one, but demand for the service $i = 5$, which yields the highest profit and also consumes the most resources, is explicitly deviated downwards from the forecast: in scenario 10 the demand intensity of service $i = 5$ deviates by approximately 10% from the forecast, in scenario 11 the deviation is 20%, and in scenarios 12 to 14 the deviation is increased by additional 10% per scenario, so that in scenario 14 the deviation is 50%.

TABLE III
SIMULATION RESULTS: ACHIEVED REVENUES BY EVERY POLICY

Demand	FCFS	DLP	RLP	CEC	SABP-c
Scenario0	3730.5	6123	6190.5	7473.5	7208.5
Scenario1	3670.5	6268.5	6069	7461	7364.5
Scenario2	3720	5959.5	6312	7392.5	7286
Scenario3	3730.5	6081	6463.5	7342	7356
Scenario4	3693	5947.5	6519	7291	7434.5
Scenario5	3687	6003	6190.5	7327	7418
Scenario6	3703.5	5851.5	6477	7264	7356
Scenario7	3616.5	6415.5	6415.5	7271.5	7405
Scenario8	3703.5	6037	6346	7255.5	7330.5
Scenario9	3703.5	6051	6415.5	7146	7345.5
Forecast	3676.5	6012	6199.5	7513	7456.5

C. Simulation results

The main results of the simulation are summarized in table III. For the purpose of comparison, all scenarios were also evaluated using the first-come-first-serve behavior. FCFS lead to poor results in revenue compard to all other policies due to almost only accepting requests for lower services, since these requests are at the beginning of the period.

The DLP policy was implemented using a dynamic version, i.e. with recalculation of the bid prices at each tenth part of the booking period, what means that the dual problem of the DLP policy is solved several times during the booking period. In scenarios 0 to 9 DLP on average yielded about 64.4% more revenue than a simple FCFS acceptance.

RLP was also implemented with recalculation of the bid prices at each tenth part of the booking period. Due to using the average of numerous instances of the demand distribution instead of the expected demand, RLP produced an average revenue increase of 4.5% compared to the DLP policy in scenarios 0 to 9.

The CEC policy showed a good revenue performance in scenarios 0 to 9, and on average yielded 20.5% more revenue compared to DLP and 15.5% more revenue compared to RLP. Furthermore, it achieved slightly better results in revenue than SABP-c in scenarios 0 to 2.

Overall, the customized SABP policy resulted in the best revenues. In the direct comparison between DLP and SABP-c, SABP-c outperformed DLP with an average revenue increase of 21.1%, and outperformed RLP with an average revenue increase of 16.0%. Compared to CEC the revenue yielded turned out higher in seven out of ten scenarios. Additionally, by looking at scenarios 10 to 14 in table IV, it is observable that when demand for high-class services deviates downwards, SABP-c is more independent from the forecast and can keep up a better revenue performance.

TABLE IV
SIMULATION RESULTS: ACHIEVED REVENUES - CEC AND SABP-C

Demand	CEC	SABP-c	Diff. in %
Scenario10	7396	7464.5	0.93
Scenario11	7306	7426.5	1.65
Scenario12	7189	7407.5	3.04
Scenario13	7072	7366.5	4.16
Scenario14	6955	7317.5	5.21

V. CONCLUSION

Commercial providers in a Cloud have to decide how to set their prices and how to plan the capacity. Efficiency and revenue play an important role. The decision about accepting or denying requests has a high impact on the revenue of the provider. Revenue Management concepts known from the airline industry comprise interesting policies. In this paper we analyze the policy concepts and present how to apply these concepts in the Cloud Computing domain. Furthermore, we propose a bid price based policy to determine the minimum price a consumer has to pay for requesting a service. The complexity arises when services utilize the same resources.

Evaluating incoming requests with the DLP policy can be critical, because the usage of the dual solution as bid prices turned out to be non-optimal and very inaccurate.

The customized version of the SABP concept is able to calculate promising values for the bid prices, but strongly depends on the configuration of the control variables. The revenue performance of the customized SABP policy clearly is better than the one of DLP and RLP policies. Moreover, it is more robust to deviations from the expected demand. The main computational effort of SABP-c arises from the

calibration of the control variables by the genetic algorithm. If the provider has sufficient information on future demand, and thereby is able to calibrate the bid price function appropriately, the computational effort only concerning the recalculation of the bid prices for each request arriving, is moderate and acceptable.

Furthermore, the CEC policy showed good results concerning revenue performance. But in cases, in which demand for the more expensive services is less than expected, CEC can be outperformed by SABP-c. It is important to note that the CEC policy does not use bid prices to estimate the opportunity cost. It directly calculates an approximation of the opportunity cost of reserving capacity required to satisfy an accepted request. Although CEC seems to compute an accurate approximation of the opportunity cost in the present setting, it is more dependent on the forecast than SABP-c.

Due to the reasons explained, the integration of the customized SABP policy into a Cloud system can be more efficient in evaluating customer requests than using the DLP or even the CEC approach in terms of revenue performance.

ACKNOWLEDGEMENT

This work has been partially funded by the EU IST programme under grant 034286 "SORMA - Self-Organizing ICT Resource Management" and by the Australian Department of Innovation, Industry, Science and Research (DIISR).

REFERENCES

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1998.

[2] D. Abramson, R. Buyya, and J. Giddy, "A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1061–1074, 2007.

[3] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat, "Resource allocation in federated distributed computing infrastructures," in *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*, 2004.

[4] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. Huberman, "Tycoon: An implementation of a distributed, market-based resource allocation system," *Multiagent and Grid Systems*, vol. 1, no. 3, pp. 169–182, 2005.

[5] B. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. Parkes, J. Shneidman, A. Snoeren, and A. Vahdat, "Mirage: A microeconomic resource allocation system for sensornet testbeds," *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on*, pp. 19–28, May 2005.

[6] R. Buyya and S. Venugopal, "The Gridbus toolkit for service oriented grid and utility computing: an overview and status report," in *Grid Economics and Business Models, 2004. GECON 2004.*, 2004, pp. 19–66.

[7] J. Nimis, A. Anandasivam, N. Borissov, G. Smith, D. Neumann, N. Wirstrom, E. Rosenberg, and M. Villa, "SORMA-Business Cases for an Open Grid Market: Concept and Implementation," in *Grid Economics and Business Models: 5th International Workshop, GECON 2008, Las Palmas de Gran Canaria, Spain, August 26, 2008, Proceeedings*. Springer-Verlag GmbH, 2008, p. 173.

[8] J. Feigenbaum and S. Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," in *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2002, pp. 1–13.

[9] C. Weinhardt, A. Anandasivam, B. Blau, and J. Stoesser, "Business models in the service world," *IEEE IT Professional, Special Issue on Cloud Computing*, vol. 11, no. 2, pp. 28–33, 2009.

[10] A. Weiss, "Computing in the clouds," *netWorker*, vol. 11, no. 4, pp. 16–25, December 2007.

[11] R. Buyya, C. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *10th IEEE International Conference on High Performance Computing and Communications*. Washington DC, USA: IEEE Computer Society, 2008, pp. 5–13.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep., 2009.

[13] G. Bitran and R. Caldentey, "An overview of pricing models for revenue management," *Manufacturing & Service Operations Management*, vol. 5, no. 3, pp. 203–229, 2003.

[14] S. Kimes, "Yield management: A tool for capacity-constrained service firms," *Journal of Operations Management*, vol. 8, no. 4, pp. 348–363, 1989.

[15] R. Phillips, *Pricing and Revenue Optimization*. Stanford Business Books, 2005.

[16] P. Dube, Y. Hayel, and L. Wynter, "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, 2005.

[17] R. Wilson, "Nonlinear pricing and mechanism design," in *Handbook of Computational Economics (Vol. 1)*, H. M. Amman, D. A. Kendrick, and J. Rust, Eds. Elsevier, 1995, pp. 253–294.

[18] A. Sulistio, K. Kim, and R. Buyya, "Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue," in *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. IEEE Computer Society Washington, DC, USA, 2008, pp. 267–276.

[19] B. Urgaonkar, P. Shenoy, and T. Roscoe, "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: ACM, 2002, pp. 239–254.

[20] A. Anandasivam and D. Neumann, "Managing revenue in Grids," in *Hawaii International Conference on System Sciences, Proceedings of the 42nd Annual*. Springer-Verlag GmbH, forthcoming.

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

[22] E. Williamson, "Airline network seat control," Ph.D. dissertation, MIT, Cambridge, 1992.

[23] B. Smith and C. Penn, "Analysis of alternative origin-destination control strategies," in *Proceedings of the 28th Annual AGIFORS Symposium*, New Seabury, MA, 1988.

[24] D. Bertsimas and I. Popescu, "Revenue Management in a Dynamic Network Environment," *Transportation Science*, vol. 37, no. 3, pp. 257–277, 2003.

[25] A. Möller, W. Römisch, and K. Weber, "Airline network revenue management by multistage stochastic programming," *Computational Management Science*, vol. 5, no. 4, pp. 355–377, 2008.

[26] R. Klein, "Network capacity control using self-adjusting bid-prices," *OR Spectrum*, vol. 29, no. 1, pp. 39–60, 2007.

[27] S. Yeo and R. Buyya, "Pricing for Utility-Driven Resource Management and Allocation in Clusters," *International Journal of High Performance Computing Applications*, vol. 21, no. 4, pp. 405–418, 2002.

[28] K. T. Talluri and G. J. van Ryzin, *The Theory and Practice of Revenue Management*. Berlin: Springer, 2004.

[29] K. Talluri and G. van Ryzin, "An Analysis of Bid-Price Controls for Network Revenue Management," *Management Science*, vol. 44, no. 11, pp. 1577–1593, 1998.

[30] P. P. Belobaba, "Application of a Probabilistic Decision Model to Airline Seat Inventory Control," *Operations Research*, vol. 37, no. 2, pp. 183–197, 1989.

[31] F. Glover, R. Glover, J. Lorenzo, and C. Mcmillan, "The passenger mix problem in the scheduled airlines," *Interfaces*, vol. 12, pp. 73–79, 1982.

[32] G. Gallego and G. van Ryzin, "Optimal dynamic pricing of inventories with stochastic demand over finite horizons," *Manage. Sci.*, vol. 40, no. 8, pp. 999–1020, August 1994.

[33] K. Talluri and G. van Ryzin, "A randomized linear programming method for computing network bid prices," *TRANSPORTATION SCIENCE*, vol. 33, no. 2, pp. 207–216, May 1999.

[34] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT: MIT Press (A Bradford Book), 1998.