Using Revenue Management to Determine Pricing of Reservations

Anthony Sulistio[1], Kyong Hoon Kim[1,2] and Rajkumar Buyya[1]
[1]Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
{anthony, jysh, raj}@csse.unimelb.edu.au
[2]Dept. of Information Science
Gyeongsang National University, Korea
khkim@gnu.ac.kr

Abstract

Grid economy provides a mechanism or incentive for resource owners to be part of the Grid, and encourages users to utilize resources optimally and effectively. Advance reservation technique allows users to request resources in the future. However, few research has been done on determining pricing of such reservations.

In this paper, we present a novel approach of using Revenue Management (RM) to determine pricing of reservations in Grids in order to increase profits. Hence, the aim of RM is to periodically update the prices in response to market demands, by charging different fares to different customers for a same resource. We evaluate the effectiveness of RM and show that by segmenting customers, charging them with different pricing schemes and protecting resources for them who are willing to pay more, will result in an increase of total revenue for that resource. Moreover, using RM techniques ensure that resources are allocated to applications that are highly valued by the users.

1 Introduction

Grid [7] and peer-to-peer (P2P) [19] network technologies enable the aggregation of distributed resources for solving large-scale and computationally-intensive applications. Managing various resources and applications in highly dynamic Grid environments is a complex and challenging process. Resource management is not only about scheduling large and compute-intensive applications, but also the manner in which resources are allocated, assigned, and accessed. In most scheduling systems, submitted jobs are initially placed into a queue if there are no available resources. Therefore, there is no guarantee as to when these jobs will be executed. This causes problems in time-critical or parallel applications, such as task graph, where jobs may have interdependencies.

To address these issues and to ensure the specified resources are available for application's consumption when required, researchers have proposed the need for advance reservation (AR) [8, 17, 24, 28]. Common resources that can be reserved or requested are compute nodes (CNs) and network bandwidth. AR in a scheduling system solves the above problem by allowing users to gain *simultaneous* and *concurrent access* to adequate resources for the execution of such applications [28]. Currently, several Grid systems are able to provide AR functionalities, such as GARA [8], and ICENI [17].

Buyya et al. [4] introduced a Grid economy concept that provides a mechanism for regulating demand and supply of resources, and calculates pricing policies based on these criteria. With this concept, it offers an incentive for resource owners to be part of the Grid, and encourages users to utilize resources optimally and effectively, especially to meet the needs of critical applications.

Regulating demand and supply is an important issue in AR, because a study done by Smith et al. [24] showed that providing AR functionalities increases waiting times of applications in the queue by up to 37% with backfilling. This study was conducted, without using any economy models, by selecting 20% of applications using reservations on across different workload models. The finding implies that without economy models or any set of AR policies, a resource accepts reservations based on a first come first serve basis and subject to availability. Moreover, it also means that these reservations are treated similarly to high priority jobs in a local queue.

Several studies have been done to improve handling and scheduling of reservations in Grid systems with some degree of flexibilities using different techniques [11, 22, 23, 26]. However, [22, 23, 26] provide a simple pricing model to determine the usage cost of each reservation. Resources might need to adopt a more complex method to increase their incentives or profits. In order to address this problem, we incorporate *revenue management* (RM) techniques for determining the pricing model in our on-line algorithm for elastic Grid reservation-based systems [26].

The goal of elastic Grid systems is to provide users with suitable reservation options so they can *self-select* offers ac-

cording to their Quality of Service (QoS) parameters, such as deadline and budget. Similarly, the main objective of RM is to maximize profits by providing the right price for every product to different customers, and periodically update the prices in response to market demands [20]. Therefore, a resource provider can apply RM techniques to *shift demands* requested by budget conscious users to off-peak periods as an example. Hence, more resources are available for users with tight deadlines in peak periods that are willing to pay more. As a result, the resource provider gains more revenue or contributions in this scenario.

Numerous economic models for resource management have been proposed in the literature. These include: commodity market models [3], tendering or contract-net models [13], auction models [21], bid-based proportional resource sharing models [12], and cooperative bartering models [5]. From these models, RM is more suited to the commodity market one, where it complements the commodity's pricing. So far, RM techniques have been widely adopted in various industries, such as airlines, hotels, and car rentals [16].

The rest of the paper is organized as follows. Section 2 mentions an overview of RM techniques. Section 3 explains the overall model and how a RM system can be incorporated into an existing Grid system. Section 4 describes the tactics of RM. Section 5 conducts a performance evaluation, whereas Section 6 concludes the paper and suggests some further work to be done.

2 Revenue Management Techniques and Strategy

Revenue management (RM) is applicable when the following requirements are met [20]:

- capacity is limited and immediately perishable. For example, an empty hotel room of today cannot be stored to satisfy future demand.
- customers book capacity ahead of time to guarantee its availability when they need to consume it.
- the seller manages a set of fare classes and updates their availability based on market demands.

From the above criteria, RM is suitable in determining the pricing of reservations in Grids, as computing powers can be considered perishable. To successfully adapt RM, a resource provider needs to have an initial *strategy*, establishes a system that handles *bookings* and updates its *tactics* periodically based on demands [20]. These aspects are discussed next.

Table 1. An example of market segmentation in Grids for reserving jobs.

Class	User Category	Restrictions
1	Premium	none
2	Business	same VO, allow cancellation
3	Budget	same VO, non-refundable, only for a limited number of CNs

Table 2. Characteristics of different users.

Budget	Business & Premium
Relaxed deadline	Tight deadline
Run longer jobs	Run short/medium jobs
Highly price sensitive	Less price sensitive
Book earlier	Book later
More flexible	Less flexible
More accepting of restrictions	Less accepting

2.1 Market Segmentation

This is an initial step of RM that identifies different customer segments for a product, and applies different pricing to each of them. The resource provider only needs to come up with a strategy quarterly or annually. Note that a product means a reservation requested by a user.

The airlines industry is a well-known example that segments customers and offers them different fare classes based on when their book their flights prior to departure times. Each *fare class* is a combination of a price and a set of restrictions on who can purchase the product and when. For example, a customer that books a flight one day prior to a departure time can be identified as a business customer. The airline knows from historical data that business customers are less flexible to changes and less price sensitive than leisure customers who book a week before. Therefore, the airline can sell a higher price to business customers compare to leisure customers for seats in a same flight.

In Grids, resources can be part of one or more *virtual organizations* (VOs). The concept of a VO allows users and institutions to gain access to their accumulated pool of resources to run applications from a specific field [9], such as high-energy physics or aerospace design. Table 1 shows an example of market segmentation in Grids. The classifications are based on VOs and time of bookings prior to reservations. Moreover, we profile each user category in Table 2.

2.2 Price Differentiation

Once users' classifications and profiling are identified, restrictions can be introduced to create virtual products ori-

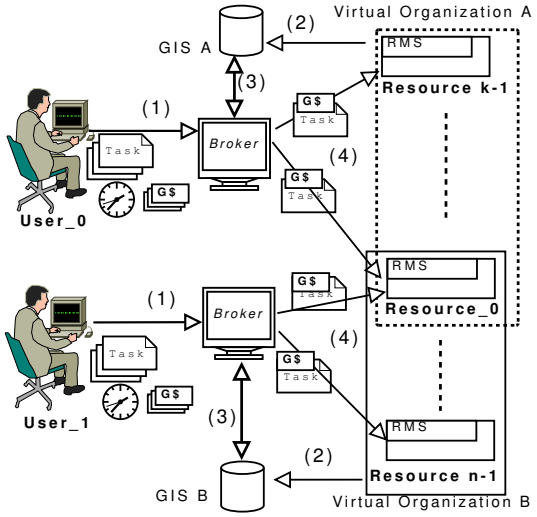

Figure 1. An overview of the model for two Virtual Organizations (VOs). Resource 0 is part of VO domain A and B.

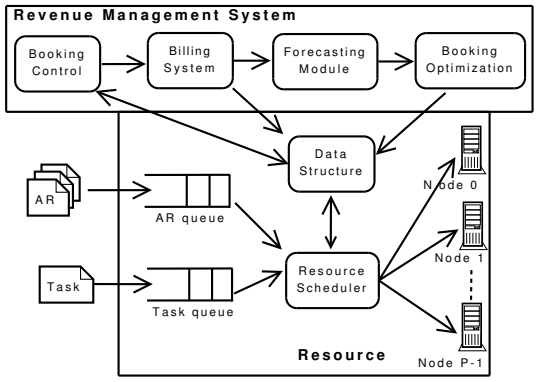

Figure 2. Revenue Management System as part of a Grid resource.

ented toward different market segments to make additional profits. As an example, products for the *Budget* users have many restrictions, as shown in Table 1, that make them unsuitable and unavailable to users with tight deadlines and from different VOs respectively. As a result, an *inferior* product can be sold to a more price-sensitive segment of the market [20]. Therefore, the resource provider can set prices for the same product to be: $p_1 > p_2 > p_3$, where p_1 denotes the price paid by the *Premium* (class 1) users and so on. This practice is commonly known in the economics literature as price differentiation or discrimination.

The main advantage of this approach is that these prices can be adjusted dynamically based on demands, since Grid resources are limited. Hence, by increasing the price to all classes during peak periods, it can shift some demands from the *Budget* users to off-peak periods. As a result, more resources are available for reservations for both the *Premium* and *Business* users.

3 Description of the Model

In our model, as depicted in Figure 1, each resource has a Revenue Management System (RMS). The RMS is responsible for handling requests and bookings. Also in the model, one VO consists of a Grid Information Service (GIS) and one or more resources and users. Figure 1 also shows the interaction between relevant components in our model. The explanation of these interaction steps are explained below:

1. *User_0* sends tasks and an initial fund, with a specified deadline time, to his/her broker. The same applies to *User_1*. In this model, the money is represented in Grid dollars (G$) term.

2. Each resource advertises its availability to a designated GIS. In Figure 1, *Resource_0* is part of VO domain A and B. Hence, this resource registers to both GIS of domain A and B.

3. The broker queries a list of available resources to the GIS. In Figure 1, the broker of *User_0* queries to the GIS of domain A, because it is running an application specific to domain A only. Likewise for *User_1* running an application in domain B.

4. Before making a booking, the broker queries to each resource about future availabilities and their prices. Once the broker has decided on which offers to choose, it sends tasks and money to these resources.

3.1 Revenue Management System

Figure 2 shows how the RMS can be integrated into an existing Grid resource [26]. With the adoption of RMS, the *Booking Control* (BC) is now responsible for handling users queries and bookings. This is done by consulting and checking booking limits in the data structure. A *booking limit* is the maximum number of CNs that may be reserved at each fare class. Once the query yields a list of options, the *Billing System* (BS) calculates a fare class for each of them. Then, the BS sends this information to the user or his/her broker. The BS also handles the user payment and confirms his/her booking by submitting this information to the data structure.

Forecasting Module (FM) is responsible for generating and updating forecasts of demands in the future. Initially, the forecast can be done about two to three weeks prior to

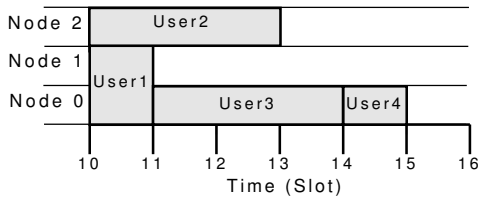

Figure 3. An example of existing reservation bookings.

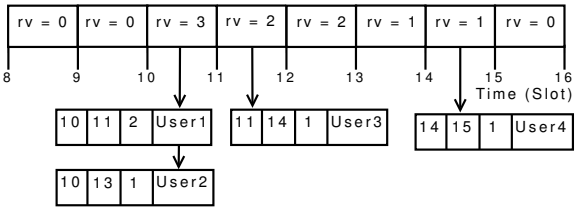

Figure 4. A representation of storing reservations with a sorted queue and δ = 1.

an opening of bookings. Then the FM updates this forecast frequently as bookings and cancellations are received over time from the BS.

These forecasts are then used as inputs by the *Booking Optimization* to re-generate booking limits for each user class. Hence, if the demands are deemed to be low, the booking limit for the *Budget* users is set to a higher number in order to increase the existing capacity. Forecasting and optimization will be discussed in more details in Section 4.

3.2 Resource

Figure 2 also shows the open queuing network model of a resource applied to our work. In this model, there are two queues: one is reserved for AR jobs while the other one is for parallel and independent tasks. Each queue stores jobs waiting to be processed by one of P independent CNs. All CNs are connected by a high-speed network. The CNs in the resource can be homogeneous or heterogeneous. In this paper, we assume that a resource has homogeneous CNs, each having same processing power, memory and hard disk.

A resource scheduler is responsible for managing incoming jobs and assigning them to available CNs. To prevent starvation among tasks or jobs that do not utilize reservations, the resource provider might want to partition the CNs initially. Then, the scheduler can use an EASY backfilling method [18] to schedule these jobs to empty CNs that are used for reservations.

3.3 Data Structure

A well-designed data structure provides the flexibility and easiness in implementing various algorithms. Hence, some data structures are tailored to specific applications, e.g. a tree-based data structure is commonly used for admission control in network bandwidth reservation [2, 29].

For our model, we use an array-based structure for administering reservations efficiently, as shown in Figure 4. It is a *time-slotted* structure, where each slot contains rv, the number of already reserved CNs, and a linked list for storing reservations that start at this time. Thus, it partitions dur into slots based on a fixed time interval δ. If dur spans multiple slots, rv on each of them is updated accordingly. Figure 4 shows how reservations are stored with a sorted queue and δ = 1 time interval, by using the example described in Figure 3. For enabling a fast $O(1)$ access to a particular slot, we use the following formula:

$$i = \left\lceil \frac{t}{\delta} \right\rceil \mod M \tag{1}$$

where i is the slot index, t is the request time (in minutes), and M is the number of slots in the data structure. Note that in order not to overlap reservation from different months, we assume that no reservations are made more than one month in advance. As a result, the data structure can be reused for the next month interval. Hence, it is only going to be built once in the beginning.

To incorporate RM functionalities into the data structure, each slot contains b_1, b_2, and b_3 denoting the booking limit for class 1, 2 and 3 respectively.

4 Revenue Management Tactics

RM tactics are used in a daily operational planning to calculate and update booking limits. For these tactics, we assume that class 3 (*Budget*) users reserve *before* class 2 users *before* class 1 users, as shown in Figure 5. This assumption is used so that once a booking limit for class 3, b_3, is reached, then users will be offered a fare class of the next one, i.e. class 2, and so on.

4.1 Protection Levels and Nested Booking Limits

When an initial demand is generated, the *Forecasting Module* sets protection levels, y_1 and y_2 for class 1 and 2 respectively. A *protection level* is required in order to make some CNs available for business and premium users that might book later in time, as shown in Figure 5.

In order to prevent high-fare bookings are being rejected in favor of budget ones, a nested approach is used to determine b_i, where b_i denotes the booking limit for class i, as shown in Figure 5. With this approach, the booking limits

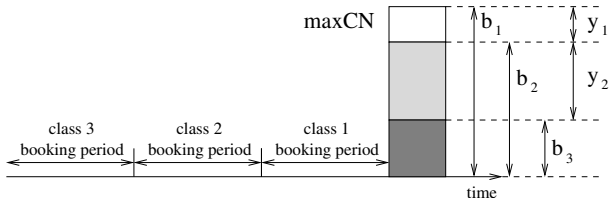

Figure 5. Protection levels (y_1, y_2) and nested booking limits (b_1, b_2, b_3) for each time slot.

are always nonincreasing, i.e. $b_1 \geq b_2 \geq b_3$. In addition, every class has access to all of the bookings available to lower classes. Hence, b_1 denotes the maximum number of CNs to be reserved.

4.2 Capacity Allocation Problem

The capacity allocation problem in RM is to decide the booking limit for each class user, in order to maximize the overall expected total revenue. If too many CNs are allocated to lower-class users during peak periods, we may loose a chance to earn more revenue from accepting future bookings from higher-class users. On the contrary, an insufficient quota for the lower-class users in off-peak periods, may lead to a lower resource utilization and revenue. Thus, finding an appropriate capacity allocation to each user class at different time periods is an important factor in RM.

Let p_i denotes the price of class i. Since the price of higher class is more expensive than that of a lower class, it follows that $p_i > p_{i+1}$. We assume that a cumulative distribution function of class i's demand is given by $F_i(x)$, because the capacity allocation analysis is based on forecasting future bookings [16]. Thus, $F_i(x)$ is the probability that the demand of class i user is less than or equal to x.

Let us first consider a two-class allocation problem for a given capacity C, where h denotes a higher class and l denotes a lower class. We assume that the current booking limit for the lower class is $b_l - 1$. The expected revenue, E, can be changed by increasing the booking limit from $b_l - 1$ to b_l, i.e. $IR(b_l)$, and depends on the demand d_l of the two classes. If $d_l \leq (b_l - 1)$, then the expected revenue is the same. However, if $d_l > (b_l - 1)$, then the revenue depends on d_h. In this case, the revenue can be increased by p_l, if $d_h \leq (C - b_l)$. On the contrary, if $d_h > (C - b_l)$, the resource provider will lose $(p_h - p_l)$. The expected revenue increase from $b_l - 1$ to b_l is defined by the following [20]:

$$\begin{aligned} E[IR(b_l)] &= (1 - F_l(b_l - 1)) \times \\ &\quad \{F_h(C - b_l)p_l - (1 - F_h(C - b_l))(p_h - p_l)\} \\ &= (1 - F_l(b_l - 1))\{p_l - (1 - F_h(C - b_l))p_h\} \end{aligned}$$

The algorithm to calculate b_l is shown in Algorithm 1,

Algorithm 1: BookingLimit ($C,\ p_h,\ p_l,\ F_h$)

$b_l \leftarrow 0$;
while $b_l < C$ **do**
  $b_l \leftarrow b_l + 1$;
  $E[IR(b_l)] \leftarrow (1 - F_l(b_l - 1))\{p_l - (1 - F_h(C - b_l))p_h\}$;
  if $E[IR(b_l)] \leq 0$ **then return** $b_l - 1$;
end
return b_l ;

where it starts from zero and keeps incrementing until the expected revenue becomes zero or negative. As a result of Algorithm 1, the protection level of a higher class is also determined by $C - b_l$.

Let us consider the capacity allocation problem of three classes in the RMS. We use an expected marginal seat revenue (EMSR) heuristic [1] to determine the booking limits of three classes, as shown in Algorithm 2. In order to determine b_3, the protection levels of class 1 and 2 need to be calculated first, as shown in Algorithm 2. Then, b_2 can be found by using the two-class problem with $C = maxCN - b_3$.

Algorithm 2: CapacityAllocation

$y_1 \leftarrow$ *maxCN* − *BookingLimit* (*maxCN*, $p_1,\ p_3,\ F_1$) ;
$y_2 \leftarrow$ *maxCN* − *BookingLimit* (*maxCN*, $p_2,\ p_3,\ F_2$) ;
$b_3 \leftarrow \max(0,$ *maxCN* $- y_1 - y_2)$;
$b_2 \leftarrow b_3 +$ *BookingLimit* (*maxCN* $- b_3,\ p_1,\ p_2,\ F_1$) ;
$b_1 \leftarrow$ *maxCN* ;

4.3 Cost and Variable Pricing

As mentioned previously, in this model, we differentiate jobs based on whether they are using reservations or not. Therefore, costs for executing these jobs would also be different. For non-AR jobs, we calculate the running cost as

$$Cost = dur * numCN * bcost \tag{2}$$

where dur denotes the job runtime, $numCN$ denotes the number of CNs used, and $bcost$ is the base cost of running a job at one time unit. Intuitively, the cost for jobs that use AR will incur higher due to the privilege of having guaranteed resources at a future time. Hence, the running cost for AR jobs is charged based on the number of reserved slots in the data structure. More precisely,

$$Cost_{AR} = numSlot * numCN * bcost_{AR} \tag{3}$$

$$bcost_{AR} = \tau * bcost * \delta \tag{4}$$

where $numSlot$ is the total number of reserved slots, $bcost_{AR}$ is the cost of running the AR job at one time slot, and τ is a constant factor ($\tau \geq 1$) to differentiate the pricing.

Table 3. An example of variable pricing with different τ_1, τ_2, and τ_3 during the week.

Pricing Name	Day Period	Time Period	τ_1	τ_2	τ_3
Super Saver	Weekdays	12 am – 06 am	1.88	1.56	1.25
Peak	Weekdays	06 am – 06 pm	3.38	2.81	2.25
Off-Peak	Weekdays	06 pm – 12 am	2.63	2.19	1.75
Super Saver	Weekends	06 pm – 06 am	1.88	1.56	1.25
Off-Peak	Weekends	06 am – 06 pm	2.63	2.19	1.75

The above cost model is considered to be static because it does not consider the case where demand fluctuates over time in a predictable way. Hence, to increase profitability, a resource owner needs to consider variable pricing for different user segments and time period. Table 3 shows an example of setting different τ of equation (4), according to demands or daily arrival rate from several parallel and Grid workload traces [6, 15]. Note that τ_1, τ_2, and τ_3 denote τ for user class 1, 2 and 3 respectively.

4.4 Overbooking

Once users book a certain amount of CNs, the resource provider expects them to submit their jobs before reservations start. However, in a real-life scenario, users may cancel their jobs beforehand or by not submitting at all (*no-show*). *Overbooking* deals with these issues, and it can be effectively used to minimize the loss of revenue [16, 20]. In this paper, we present possible strategies for the overbooking problem in Grids. However, a detailed analysis will be conducted as part of future work.

- *A probability-based policy*: The amount of overbooking capacity is determined statistically based on the probability of cancellation (p_{cancel}) and no-show (p_{noshow}). Hence, the FM can determine the booking capacity to be $maxCN \ / \ (1 - p_{cancel} - p_{noshow})$.

- *A penalty-based risk analysis*: We can define various penalty policies to allow users to pay nominal fees for cancellations and/or no-shows.

- *A compensation-based risk analysis*: When the RMS accepts more bookings than the maximum capacity, the resource provider should offer compensations to the affected users for canceling their reservations. Thus, a risk analysis is required in order to increase the revenue and to minimize total compensation costs.

- *A hybrid scheme*: The total revenue can be improved by using a hybrid scheme based on the above methods.

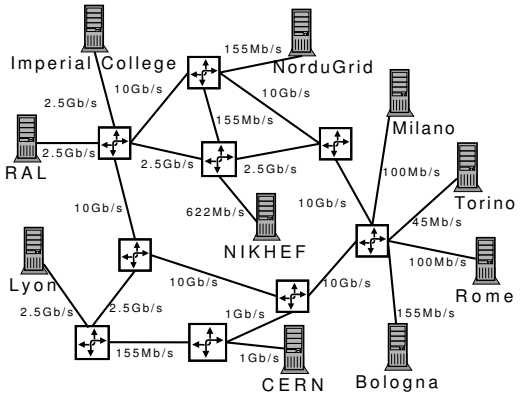

Figure 6. The simulated topology of EU DataGrid TestBed 1.

5 Performance Evaluation

We carried out the performance evaluation by using simulation, because we need to conduct *repeatable* and *controlled* experiments that would otherwise be difficult to perform in real Grid testbeds. Therefore, we use GridSim toolkit [27] to create an experiment based on EU DataGrid TestBed 1 [10]. The testbed topology is shown in Figure 6. The details of simulation parameters are discussed next.

5.1 Simulation Setups

Table 4 summarizes all the resource relevant information. In GridSim, a CPU rating of one node is modeled in the form of MIPS (Million Instructions Per Second) as devised by Standard Performance Evaluation Corporation (SPEC) [25]. The resource settings were obtained from the current characteristics of the real LHC testbed [14]. We took the data about these resources and scaled the number of nodes of each resource by 10. This is because simulating original computing capacities is not possible due to limited physical memory in a computer, since many resources and jobs need to be created during the simulation.

We divide the resources into four VOs based on their location, as shown in Table 4. Moreover, all of them use the same data structure with δ = 5 minutes, and has a fixed interval length of 30 days. To determine $bcost_{AR}$, we use τ from Table 3 for different time period.

We model incoming job traffic at three levels: resource, VO and Grid, by using a Poisson model with different lambdas for peak (λ_{peak}), off-peak (λ_{off}) and super saver (λ_{saver}) period, as depicted in Table 4 and 5. With these lambdas, we can set the peak period to be arriving more frequently than the off-peak period and so on. The lambdas for Grid and VO levels are taken from [15], where the authors

Table 4. Resource specifications and their jobs' inter-arrival rates (λ).

Resource Name (Location)	ID	# Nodes	CPU Rating	VO	$bcost$ (G$)	μ runtime	λ_{peak}	λ_{off}	λ_{saver}
RAL (UK)	$R1$	41	49,000	1	0.49	3 hours	0.01670	0.00835	0.004175
Imperial College (UK)	$R2$	52	62,000	1	0.62	3 hours	0.01670	0.00835	0.004175
NorduGrid (Norway)	$R3$	17	20,000	2	0.20	3 hours	0.00835	0.004175	0.0020875
NIKHEF (Netherlands)	$R4$	18	21,000	2	0.21	3 hours	0.00835	0.004175	0.0020875
Lyon (France)	$R5$	12	14,000	3	0.14	3 hours	0.00835	0.004175	0.0020875
CERN (Switzerland)	$R6$	59	70,000	3	0.70	3 hours	0.03340	0.00167	0.000835
Milano (Italy)	$R7$	5	7,000	4	0.07	3 hours	0.00418	0.0020875	0.00104375
Torino (Italy)	$R8$	2	3,000	4	0.03	3 hours	0.00167	0.000835	0.0004175
Rome (Italy)	$R9$	5	6,000	4	0.06	3 hours	0.00418	0.00209	0.001045
Bologna (Italy)	$R10$	67	80,000	4	0.80	3 hours	0.03340	0.0167	0.00835

Table 5. mean CPU rating for Grid and VO level and their jobs' inter-arrival rates (λ).

Level	μ Rating	μ runtime	λ_{peak}	λ_{off}	λ_{saver}
Grid	56,000	2 hours	0.13812	0.02290	0.01979
VO 1	56,000	5 hours	0.05087	0.02092	0.01913
VO 2	20,000	5 hours	0.05954	0.00537	0.00295
VO 3	60,000	5 hours	0.15901	0.00097	0.00046
VO 4	68,000	5 hours	0.07098	0.00672	0.00257

used a 3-stage Markov Modulated Poisson Process (MMPP) model in their workload analysis. For job runtime, we use an exponential distribution with different mean (μ) for each level. Since we are trying to simulate BoT applications, we set the number of reserved CNs to be 1 for all jobs.

We identify the Grid-level trace to be *Premium* users with a booking period of 2 hours and a search limit time (sl_{t1}) of 2 hours. The search limit time is used for finding alternative time slots if resources in the initial starting time are unavailable. We choose each resource-level trace to be *Business* users with sl_{t2} of 4 hours. Finally, each VO-level trace is set to *Budget* users with sl_{t3} of 24 hours because they are more flexible. All traces have the same booking period as in the Grid-level one.

For the *Premium* users, they will choose a resource from the Grid based on the earliest job completion time, whereas for the *Budget* users, they will submit jobs to a resource within the VO based on the cheapest price. Since all resources have different CPU ratings, we scale each job duration in the trace according to the μ rating found in Table 5. However each *Business* user is designated to submit to a particular resource, so no scaling is required. For all users, if a reservation for the current job can not be found, then we ignore this job and proceed to the next one. Overall, we simulate 15 traces in this evaluation for a period of 14 days.

The main objective of this experiment is to look at the impact of using RM in increasing the revenue of a resource. Therefore, we have two scenarios: in Scenario 1 (S1), we select $R1$ (RAL) and $R10$ (Bologna) to use a static pricing

Table 6. Total revenue for each resource.

Resource	S1 (x1000)	S2 (x1000)	% gain / loss
RAL	G$ 834	G$ 31,523	3,678.70
Imperial	G$ 66,662	G$ 61,645	-7.53
NorduGrid	G$ 2,570	G$ 4,638	80.44
NIKHEF	G$ 4,928	G$ 5,171	4.94
Lyon	G$ 684	G$ 742	8.37
CERN	G$ 101,997	G$ 103,529	1.50
Milano	G$ 170	G$ 171	0.57
Torino	G$ 10	G$ 13	26.46
Rome	G$ 114	G$ 119	4.07
Bologna	G$ 2,051	G$ 147,279	7,079.71

Table 7. Initial protection levels, y_1 and y_2.

Resource Name	Peak		Off-Peak		Super Saver	
	y_1	y_2	y_1	y_2	y_1	y_2
RAL	10	20	5	15	3	7
Imperial	12	27	6	20	2	11
NorduGrid	5	8	2	6	1	3
NIKHEF	5	8	2	6	1	3
Lyon	3	6	2	4	0	3
CERN	12	32	6	23	3	11
Milano	1	2	0	2	0	1
Torino	0	1	0	0	0	0
Rome	1	2	0	2	0	1
Bologna	15	35	8	25	4	12

method with τ_s = 1.9 (without RM), and $R2 - R9$ to adopt RM. Then in Scenario 2 (S2), all resources use RM. We compare these scenarios with the same set of parameters.

5.2 Experiment Results

Table 6 shows the total revenue earned by each resource in both scenarios. $R1$ and $R10$ make a huge profit by adopting RM in S2, instead of using a static pricing in S1. This is because $R1$ and $R10$ protect some nodes for the *Premium* and *Business* users' bookings. Moreover, these users pay a higher rate of τ compare to τ_s and τ_3. However, RM also provide a limited number of available nodes with a cheaper price to the *Budget* users. According to Table 3, the average of τ_3 is 1.65 or at least 15% cheaper than τ_s. As a result,

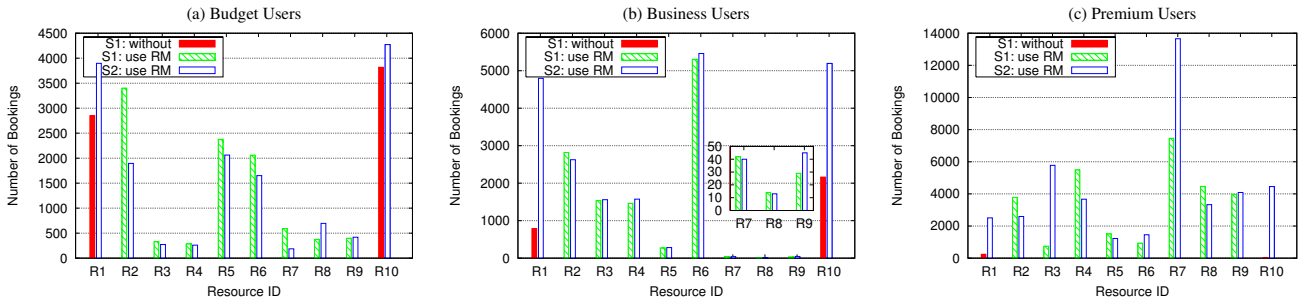

Figure 7. Total number of bookings.

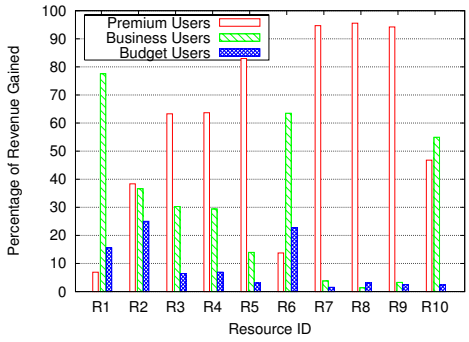

Figure 8. Percentage of income revenue in scenario 2 (S2 - all resources using RM).

RM is beneficial to both time- and budget-conscious users, and resource providers.

Table 7 shows the initial protection levels, y_1 and y_2 for the *Premium* and *Business* users respectively for S2. Based on this table, the *Budget* users are allocated to 25%, 50% and 75% of total capacity during peak, off-peak and super saver period respectively. Since y_1 and y_2 will be re-forecasted dynamically based on demand fluctuation, according to Algorithm 2, a resource provider is only required to give an initial estimation.

Figure 7 shows the total number of bookings made by each resource for different user classes. When $R1$ uses a static pricing (without RM) in S1, although $bcost$ of $R2$ is more expensive, the *Budget* users prefer to send their jobs to $R2$, due to a cheaper price overall in the VO, as depicted in Figure 7 (a). However, when $R1$ adopts RM in S2, more bookings from these users are being made. The same trend can be observed for $R10$ in `VO_4`.

Figure 7 (b) and (c) show the bookings made by the *Business* and *Premium* users respectively. Due to the fact that no protection levels are imposed on $R1$ and $R10$ in S1, when they want to book closer to the reservation time, no available nodes are found. As a consequence, the *Business* users have to cancel their bookings, and the *Premium* users have to use other resources in the Grid. This situation is called *dilution*, since $R1$ and $R10$ decrease the revenue they would have received from protecting additional nodes, y_1 and y_2, for these users.

When $R1$ and $R10$ utilizing RM in S2, the number of bookings are significantly grown for all user classes, especially the *Business* and *Premium* users. As a result, $R1$ and $R10$ are experiencing a huge increase in the revenue, as shown in Table 6. However, the increased number of bookings have an effect in other resources, as depicted in Figure 7 (a) and (c). This is because the *Budget* and *Premium* users can book to any available resources within the VO and Grid respectively. Among other resources, the impact was felt by $R2$ the hardest, with a decrease of 7.53% in the revenue as mentioned in Table 6, since $R2$ is located on the same VO as $R1$.

Figure 8 shows the percentage of incoming revenue for each user class. For smaller and medium-sized resources, such as Torino ($R8$) and NorduGrid ($R3$), the *Premium* users contribute more than 60% of the total revenue. On the other hand, the *Business* users contribute more than 50% on large-sized resources, such as CERN ($R6$) and Bologna ($R10$). Hence, from this figure, both the *Business* and *Premium* users are a major source of revenue for a resource. Therefore, in a competitive demand and supply market, a resource needs to differentiate itself among others to attract these users.

6 Conclusion and Future Work

Advance Reservation (AR) in a Grid system allows users to gain *simultaneous* and *concurrent access* to adequate resources for running their applications or jobs. Common resources that can be reserved are compute nodes and network bandwidth. In this paper, we present a novel approach of using Revenue Management (RM) to determine pricing of reservations in a Grid system.

The main objective of RM is to maximize profits by providing the right price for every product to different customers, and to periodically update the prices in response to

market demands. Therefore, a resource provider can apply RM techniques to *shift demands* requested by budget conscious users to off-peak periods as an example. As a result, more resources are available for users with tight deadlines in peak periods that are willing to pay more.

We evaluate the effectiveness of RM and show that by segmenting users, charging them with different pricing schemes, and protecting resources for those who are willing to pay more, will result in an increase of total revenue for that resource.

As for future work, we need to consider a scenario where the demands are dependent, such as budget users reserve compute nodes at a full-fare price, normally known as a *buy-up*. We also need to consider cancellation and overbooking strategies in the model.

Acknowledgment

This work is partially supported by research grants from the Australian Research Council (ARC) and Australian Department of Education, Science and Training (DEST).

References

[1] P. P. Belobaba. Application of a probabilistic decision model to airline seat inventory control. *Operations Research*, 37(2):183–197, 1989.

[2] A. Brodnik and A. Nilsson. A static data structure for discrete advance bandwidth reservations on the internet. In *Proc. of Swedish National Computer Networking Workshop (SNCNW)*, Stockholm, Sweden, Sep 2003.

[3] R. Buyya, D. Abramson, and J. Giddy. Nimrod-G: An architecture for a resource management and scheduling system in a global computational grid. In *Proc. of the 4th Intl. Conference & Exhibition on High Performance Computing in Asia-Pacific Region (HPC Asia)*, Beijing, China, May 2000.

[4] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[5] B. F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. In *Proc. of the 22nd Intl. Conference on Distributed Computing Systems*, Vienna, Austria, July 2–5 2002.

[6] D. Feitelson. Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload, 2007.

[7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[8] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proc. of the 7th Intl. Workshop on Quality of Service*, London, 1999.

[9] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, 15(3), 2001.

[10] W. Hoschek, F. J. Jaén-Martínez, A. Samar, H. Stockinger, and K. Stockinger. Data management in an international data grid project. In *Proc. of the 1st Intl. Workshop on Grid Computing*, Bangalore, India, Dec. 17 2000.

[11] N. R. Kaushik, S. M. Figueira, and S. A. Chiappari. Flexible time-windows for advance reservation scheduling. In *Proc. of the 14th Intl. Symposium on Modeling, Analysis, and Simulation (MASCOTS)*, California, USA, Sep. 11–13 2006.

[12] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical Report arXiv:cs.DC/0404013, HP Labs, USA, Apr. 2004.

[13] S. Lalis and A. Karipidis. JaWS: An Open Market-Based Framework for Distributed Computing over the Internet. In *Proc. of the 1st IEEE/ACM Intl. Workshop on Grid Computing*, Bangalore, India, Dec. 17 2000.

[14] LCG computing fabric. http://lcg-computing-fabric.web.cern.ch, 2005.

[15] H. Li and M. Muskulus. Analysis and modeling of job arrivals in a production grid. *SIGMETRICS Performance Evaluation Review*, 34(4):59–70, 2007.

[16] J. I. McGill and G. J. V. Ryzin. Revenue Management: Research Overview and Prospects. *Transportation Science*, 33(2):233–256, 1999.

[17] S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Workflow enactment in ICENI. *UK e-Science All Hands Meeting*, pages 894–900, Sep 2004.

[18] A. W. Mu'alem and D. G. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, 2001.

[19] A. Oram, editor. *Peer-to-peer: Harnessing the Power of Disruptive Technologies*. O'Reilly Press, 2001.

[20] R. L. Phillips. *Pricing and Revenue Optimization*. Stanford University Press, 2005.

[21] O. Regev and N. Nisan. The POPCORN Market - An Online Market for Computational Resources. In *Proc. of the 1st Intl. Conference on Information and Computation Economies (ICE)*, Charleston, USA, 1998.

[22] T. Roeblitz, F. Schintke, and A. Reinefeld. Resource reservations with fuzzy requests. *Concurrency and Computation: Practice & Experience (CCPE)*, 18(13):1681–1703, 2006.

[23] M. Siddiqui, A. Villazon, and T. Fahringer. Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *Proc. of the 2006 ACM/IEEE conference on Supercomputing (SC'06)*, Florida, USA, Nov 2006.

[24] W. Smith, I. Foster, and V. Taylor. Scheduling with advanced reservations. In *Proc. of the Intl. Parallel and Distributed Processing Symposium*, Cancun, Mexico, May 1–5 2000.

[25] SPEC. Standard Performance Evaluation Corporation. http://www.spec.org, 2007.

[26] A. Sulistio, K. H. Kim, and R. Buyya. On Incorporating an On-line Strip Packing Algorithm into Elastic Grid Reservation-based Systems. In *Proc. of the 13th Intl. Conference on Parallel and Distributed Systems (ICPADS)*, Hsinchu, Taiwan, Dec. 5–7 2007.

[27] A. Sulistio, G. Poduval, R. Buyya, and C.-K. Tham. On Incorporating Differentiated Levels of Network Service into GridSim. *Future Generation Computer Systems*, 23(4):606–615, May 2007.

[28] A. Sulistio, W. Schiffmann, and R. Buyya. Advanced Reservation-based Scheduling of Task Graphs on Clusters. In *Proc. of the 13th Intl. Conference on High Performance Computing (HiPC)*, Bangalore, India, Dec. 18–21 2006.

[29] T. Wang and J. Chen. Bandwidth tree – a data structure for routing in networks with advanced reservations. In *Proc. of the 21st Intl. Performance, Computing, and Communications Conference*, pages 37–44, Phoenix, USA, 2002.