MEDWELL PUBLISHING
International Journal of Soft Computing 4 (2): 60-67, 2009
ISSN: 1816-9503

A Dynamic Priority Scheduler for Advance Reservation in Grid Computing

[1]Ravin Ahuja, [1]G. Gabrani and [2]Asok De
[1]Department of Computer Engineering, Delhi College of Engineering,
Bawana Road, New Delhi-110042, India
[2]Department of Electronics and Communication Engineering,
Delhi College of Engineering, Bawana Road,
Ambedkar Institute of Technology, New Delhi, India

Abstract: In the grid technologies, it has become possible to allow many users or applications having multiple jobs to seamlessly access and share huge and heterogeneous pools of computational data, storage and network resources that are geographically distributed. This research is into the domain of problems arising out of making resources available not only time specific but also to cater to the needs of specific users with different priorities at a particular time. In the present dynamic scenario when jobs (belonging to different users with different priorities) are aplenty, the situation arising confronts a challenge regarding scheduling of the incoming jobs after resolving their priorities and allocating them to the resources. These jobs can be allocated to the resources either immediately (current reservation) or for some time in future (advance reservation). This study presents, a Dynamic Priority Scheduler for Advance Reservation (DPSAR) in a space shared environment to coordinate the resource sharing in distributed grid computing environments. The DPSAR aims at analyzing the multifarious jobs and sequel to calibrating on the basis of priorities and protocols further reserving them to the respective resources. In this study, we have simulated the DPSAR and its results are analyzed to measure the performance issues. The results confirmed to an improved performance in terms of number of rejections and resource utilization over an existing scheduler based on advance reservation technique (without priority) and pit falls were found to be nominal.

Key words: Priority, scheduler, advance reservation, grid computing, gridlet

INTRODUCTION

Grid provides the ability to coordinate and share various resources connected through a network (Foster and Kesselman, 1999; Foster *et al*., 2002). The current trends for the management of these resources in the grid mostly employ variety of techniques viz. priority queues, backfilling techniques (Karatza, 2000) and methods like market based approach (Yeo and Buyya, 2004) such as those based on service level agreement and advance reservation. In the present networking technology and available bandwidth, the computing resources are aggregated together to form a grid computing environment. These computational resources may provide free or chargeable services. The resources may consist of one or more Processing Elements (PE) of same or different technologies. They may also have different hardware configurations that can vary from vector super computers to diskless clients. The grid also, has the capability to bring the computing power of various computing resources to work together thereby enhancing the overall computational power. This allows the grid to provide solutions to computationally intensive problems by giving an image of single system to the users (Li and Manish, 2005). The criterion for allocating jobs (or Gridlets) of users to the free PE's depends upon the paying capacity and protocol of the user (Li and Manish, 2005; Buyya and Murshed, 2002). The resources need to be monitored continuously to check their availability so that jobs can be assigned to them dynamically (Spooner *et al*., 2003). One of the major challenges is the task of resource allocation in a dynamic scenario when large number of users is joining the grid having numerous jobs to be performed.

To assign jobs to available resources, many schedulers have been proposed in the literature. These schedulers are based on various techniques such as FCFS, Round Robin, shortest job first etc. However, these

Corresponding Author: Ravin Ahuja, Department of Computer Engineering, Delhi College of Engineering, Bawana Road, New Delhi-110042, India

schedulers do not guarantee resource allocation at a specific timing i.e., the jobs scheduled by them are not allocated resources in advance. Therefore, in order to execute jobs on demand at a specified time, a scheduling system for Advance Reservation (AR) has been proposed (Foster *et al*., 2001; Sulistio and Buyya, 2004). This system consists of 2 components namely a scheduler and an advance reservation module. The function of scheduler is to schedule jobs on first come first basis and pass on these jobs to the advance reservation module. The advance reservation module does the reservation of jobs to the available resources so that the jobs are executed at a specified time in future. Thus, reservation ensures execution of a given job at a possible agreed upon time (between user and grid service provider) by a resource in advance.

This scheduling system for Advance Reservation (Sulistio and Buyya, 2004) has limitations of not complying to react to the dynamic behaviour of huge number of users with distinct characteristics and requirements seeking services on the grid for computational requests. Also, as the jobs belonging to different users have different priorities therefore different jobs demand different treatment. These priorities are to be resolved in a dynamic manner, making the job of scheduling system extremely complex. Therefore, there is a need of different scheduling system capable enough to handle the priority of incoming jobs at the run time.

In order, to overcome the limitation in the scheduler proposed in Sulistio and Buyya (2004), we in the study have suggested a novel way of designing the first component of scheduling system called the scheduler, which can handle priorities and protocols of the jobs dynamically. This component is referred to as Dynamic Priority Scheduler for Advance Reservation (DPSAR). In this study, we have designed an algorithm for DPSAR and used gridsim tool for simulation. The gridsim tool provides a simulation of a real grid environment having heterogeneous resources. We have evaluated the performance of DPSAR. Further, we have compared our proposed scheduler DPSAR with the scheduler proposed in Sulistio and Buyya (2004).

GRIDSIM SIMULATION ENVIRONMENT

Gridsim (Buyya and Murshed, 2002) is a simulation test bed that provides a distributed grid environment constituting of heterogeneous computational resources. This simulation toolkit defines java based entities for simulation of resources, users, applications, schedulers etc. These jobs can be configured as systems using either space shared or time shared based techniques. In space shared technique, the jobs once scheduled are allocated to available resources that may constitute one or more PE's (Processing element). When the PE to which job is allocated is free, the job is executed. Whereas, in the time shared technique, the jobs to be executed on a PE are given a time slot. The PE is freed for that time slot and the given job is executed during that time slot.

Table 1: Indicating gridlet ID's with the burst time as expected time for which gridlet requires a PE and their respective arrival time Gridlet with G capital not small

GJ#	Burst time	Arrival
GJ1	5	0
GJ2	7	2
GJ3	10	3
GJ4	5	4

Table 2: Gridlets being allocated to the PE's showing the availability of Processing Elements PE1 and PE2 with respect to the arrival time of gridlets

PE1	GJ1		GJ3		
PE2		GJ2			GJ4
	2		9	14	

In this study, authors have considered the space shared technique for the proposed DPSAR. In this technique, whenever users joining the grid submit their jobs also called gridlets, the jobs are allocated and executed on a free PE. In case there is no free PE available, the job is placed in the waiting queue called resource queue. As soon as a job on a given PE finishes its execution, an internal event is generated to signify the completion of the job. In response to this, simulator frees the PE allocated to it and finds if there is any other job waiting in the resource queue. The pending jobs are then selected from the resource queue and are assigned free PEs based the job allocation policy. In order to understand the working of space shared technique, let us consider a grid scenario, where there are four gridlets and two number of PEs.

In Table 1, GJ# represents the Gridlet (Job) ID, Burst time represents total time (in ms) for which user requires PE and arrival time shows the time (in ms), at which job of a user arrives in a grid system.

Table 2 shows that GJ#1 is allocated to the PE1 as it is free at that time. At 2 ms, GJ#2 arrives and gets the processor PE2 as it is also free. Now gridlets GJ#3 and GJ#4 arrive at 3 and 4 ms, respectively. Neither of these gridlets will be allocated to any PE as none of the PE is free. GJ#1 finishes at 5 ms and GJ#2 finishes at 9 ms. Now at 5 ms, PE1 is free and will be allocated to GJ#3 and at 9 ms, PE2 is free and will be allocated to GJ#4. Therefore, all the Jobs will finish their execution within 15 ms according to space shared technique.

The jobs being executed can be allocated resources in advance for their execution in future at a specific time using advance reservation. Authors have considered the

scheduling of jobs using priority as the policy criteria with advance reservation mechanism followed by execution based on space shared technique. Advance reservation can be used to support co-scheduling, especially among diverse types of grid resources. In advance, reservation there is a set of resources with availability limited to a specific user or users, at a specific start time and for a specified duration.

DYNAMIC PRIORITY SCHEDULING SYSTEM FOR ADVANCE RESERVATION

The dynamic priority scheduling system for advance reservation mainly consists of two components viz. DPSAR and the Advance Reservation Module (ARM). The DPSAR does the scheduling of jobs by resolving jobs priorities dynamically, where as Advance Reservation Module (ARM) takes care of reservation of jobs that are scheduled by DPSAR. The proposed scheduler DPSAR basically acts as a prelude to the ARM as shown in Fig. 1.

In the given grid environment, a number of users having different priorities and protocols can join the grid. The priority of a user is derived from its paying capacity and ranking protocol. Each user with different priority submits one or multiple jobs. These jobs also, have a priority, which is based on two parameters, namely the job's length and the priority of user to which it belongs to.

Whenever, a user enters grid, it is enlisted by the DPSAR in a queue called user list. The users from the User list are sorted based on their priority and are placed in a queue called user queue. Thereafter, jobs that are generated by various users are ordered on the basis of their lengths and are placed in another queue named gridlet list. The jobs are now to be prioritized again. The DPSAR prioritizes them by using a gridlet sorting policy and Dynamic Priority Resolution (DPR) technique. The sorting policy decides on the number of jobs of each user to be prioritized and to be sent for reservation. Thus, the sorting policy ensures that jobs of various users are entertained in a way so as to avoid starvation among the jobs. On the other hand, the DPR technique is used to resolve the priority of jobs by considering the parameters associated with jobs priority. The DPR therefore, is consequential in determining the priority of incoming jobs having higher user priority and higher job priority based on the job length. The DPSAR finally stores prioritized jobs in a queue called Gridlet queue. The jobs thus, scheduled by DPSAR are submitted to ARM.

ARM maintains a list of scheduled jobs and reserves them. ARM then allocates resources to the reserved jobs to be executed at a specific time in future (referred to as start execution time). ARM, while reserving jobs also anticipates the tentative computational time required to execute the job so as to reserve forthcoming jobs. The jobs are executed based on space shared technique. In space shared technique, jobs are executed only if there is a free PE available otherwise they are placed in a resource queue maintained by ARM. Whenever, a job finishes its execution, the corresponding PE is freed and is allocated to the next pending job in the resource queue. For this an internal process of ARM continuously checks the resource queue, selects the next pending job and allocates it to a free PE.

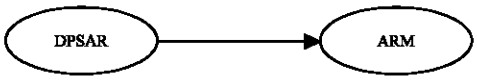

Fig. 1: DPSAR scheduler

DPSAR IN DETAIL

DPSAR state diagram: The details of DPSAR state diagram are shown in Fig. 2. It shows, the various state transitions the DPSAR goes through starting from the initial state to the request state. The initial state represents the users joining the grid, whereas in the Request state, the jobs, which are prioritized and scheduled by DPSAR are dispatched to ARM for reservation.

After the users have joined the grid, DPSAR goes through four states. In the first state, users are sorted according to their priorities. In the second state, the jobs of each user are sorted based on the lengths of jobs. In the third state, the jobs of various users are sorted in accordance with the job's priority based on job length and on priority of the user to which the jobs belong to. This is done by DPR technique as well as Gridlet Sorting Policy (GSP). The GSP based on a predefined value decides that how many jobs of a given user are to be scheduled and subsequently sent for reservation. Finally, in the last state, jobs are placed in a queue called Gridlet queue from where they are sent to ARM.

Algorithm for DPSAR: The algorithm followed by DPSAR can be broadly divided into two phases namely user Priority phase and Gridlet priority phase. The first phase deals with the user priority whereas the second phase takes care of job priority. Before explaining, the algorithm in detail, the following terminologies are first explained.

Various terminologies

User: Consumer or application seeking grid services.

User list: A list of users seeking grid services.

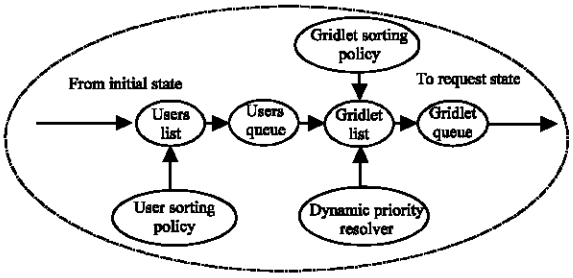

Fig. 2: State diagram for dynamic priority scheduler for advance reservation

User priority: Priority of the user according to its paying capacity and ranking protocol. It is represented by an integer value.

User sorting policy: It is policy used to sort the users.

Gridlet: Job of a user.

Gridletlist: A list of Gridlets.

Gridlet priority: Priority of the job, which is represented by an integer value. It can assume different values in different stages of the algorithm depending on either the job length or on both job length and user priority.

Gridlet sorting policy: It is the policy to sort the Gridlets I.

Significant Difference (SD): A predefined parameter that is based on difference between priorities of two jobs competing for resource reservation.

DPSAR algorithm: The algorithm for DPSAR has following two phases (Fig. 3):

User priority phase: As already mentioned, users having different priorities can join the grid. These users can have one or multiple jobs. In this phase, first the users are sorted and then the jobs of each user are sorted. The users are sorted in decreasing order of their priorities using user sorting policy. If the users have equal priorities they are arranged on first come first basis else insertion sort technique is used.

Once, the users have been sorted, jobs of each user are sorted on the basis of their lengths. The jobs are sorted using insertion sort technique. The jobs thus, sorted are placed in the Gridlet List. This technique is used as it saves time and therefore is more efficient. This is due to the fact that insertion sort has complexity of O (n), which is comparatively less than the time complexities O (nlog(n)) and O (n^2) of other sorting methods in the best or worst case, respectively.

Gridlet priority phase: In this phase, jobs already sorted in User priority phase are further prioritized based on two criteria. This criterion can either be User Priority or both job length and UserPriority (referred to as User High Priority (UHP) phase and User Job Priority (UJP) phase, respectively). DPR technique decides whether jobs will be sorted in UHP or UJP phase depending on the difference in priority of the jobs (that is user priority).

DPR technique takes two users at a time that are competing for reservation and compares their UserPriority. If the difference between user priority is larger than significant difference SD, only the jobs belonging to the higher priority user are sorted. The jobs belonging to lower priority user are queued up. These jobs are treated as pending jobs, which will further participate in the next sorting process in the user priority phase. Out of these sorted jobs of a higher priority user, only a predefined number of jobs are considered for reservation (as decided by pre-defined function) and rest are discarded. This is referred to as UHP phase.

Whereas, if the difference between the UserPriority of the two users competing for reservation is less than a significant difference SD, the priority of jobs is evaluated on the basis of their job length and User Priority. The jobs are sorted as the jobs keep arriving, the sorting process is repeated and the pending jobs of lower priority users are allowed to participate again and hence, they are prevented from getting starved. This phase is known user Job priority phase.

GRIDLET SORTING POLICY

The sorting technique used to sort the jobs in the above phases is decided by GSP. The GSP uses shortest first job technique for sorting the jobs of various users. This policy also uses a pre-defined function in UHP or UJP phase to decide as to how many jobs of high priority user are to be considered for reservation. The policy prevents all the jobs of high priority user from getting entertained and therefore avoids starvation of jobs of low priority users. The policy takes this decision on the basis of a pre-defined value.

Pseudo code for DPSAR

- Start
- Create a user list and add users to this list as users arrive and join the grid

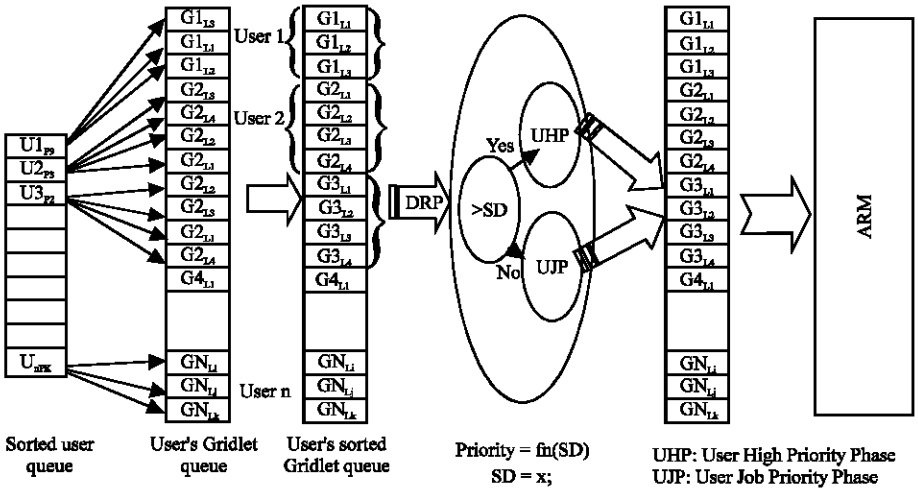

Fig. 3: DPSAR block diagram

- Sort the users in the user list based on assigned user priority using insertion sort following user sorting policy
- Create a Gridletlist and insert Gridlets in this list according to their user priority such that all gridlets of each user are placed in a sequence
- Sort the Gridlets of various users joined
- Define a Dynamic Priority Resolution (DPR) and is calculated at runtime by using

DPR = User[i]. priority-User [I+1]. priority

- Compare DPR with SD, which is significant difference between user priority.

If (DPR>SD)

User High Priority (UHP) phase: Sort the Gridlets vs Gridlets of each user according to shortest order and place into a user's gridlet queue. Consider only N number (decided by a predefined function) of shortest jobs of higher priority user to be sent for reservation schedule as decided by GSP.

else

User Job Priority (UJP) phase: Sort the Gridlets of users according to GSP, taking into account the user priority of both the users, which being compared, are placed into a queue. This leads to gridlets from both the users in the gridlet list.

- If (new user)
  Go to step 2
- If (new Gridlet)
  Go to step 4

Stop

DPSAR gives the best results when user priority is uniformly distributed among the users in the User list. Generally, SD is assigned as priority level/2. When a new user joins the grid, an event triggers scheduler thread jumps to step 7 or 8 depending upon the situations. This algorithm is explained using an example in the Fig. 3 having three users U1, U2 and U3 with priority 9, 3 and 2, respectively. U1 has 3 jobs viz. Gl_{L3}, Gl_{L1} and Gl_{L2} with length L_3, L_1 and L_2, respectively. It has been assumed that $L_i>L_j$ if i>j. Similarly, users U2 and U3 have four jobs each namely $G2_{L1}$, $G2_{L2}$, $G2_{L3}$, $G2_{L4}$ and $G3_{L1}$, $G3_{L2}$, $G3_{L3}$, $G3_{L4}$. In the second queue, (users sorted Gridlet queue), the jobs are sorted internally for each user. Thereafter, DPR compares SD for jobs competing and accordingly passes it to UHP or UJP phase, which constitutes the final queue called Gridlet queue.

Let us assume for example, user U1 has very high user priority as compared to user priority of U2. This means that the difference between user priority of two users U1 and U2 is larger than pre-defined value of SD (compared by DPR technique). Therefore, DPSAR enters UHP phase (otherwise it would have entered UJP phase). The jobs of U1 are now taken and sorted on the shortest job first basis whereas jobs of U2 are queued. According to gridlet, sorting policy only three jobs (a pre-defined value) of a higher priority user are at most entertained in order to prevent jobs of users of low user priority from starvation. Therefore, jobs $G1_{L1}$, Gl_{L2} and Gl_{L3} are scheduled for reservation and rest are discarded.

After the jobs of U1 having been considered, DPSAR now takes jobs of next two users namely U2 and U3. The priority between jobs of U2 and U3 is resolved by comparing the difference between user priority of U2 and User priority of U3. Let, us assume for instance that this difference is less than SD, therefore DPSAR enters UJP phase. As the user priority of U2 and U3 is comparable,

the priority between jobs of two users is again resolved on the basis of 2 parameters (instead of one parameter as in UHP) namely their user priority and job length. In this case, jobs of both the users U2 and U3 are considered. The jobs thus, prioritized are placed in Gridlet queue in accordance with the descending order of their priorities. The number of jobs of each user to be entertained and scheduled further for reservation is decided by GSP. Hence, the jobs $G2_{L1}$, $G3_{L1}$, $G2_{L2}$, $G3_{L2}$, $G2_{L3}$, $G3_{L3}$, $G2_{L4}$ and $G3_{L4}$ of U2 and U3, respectively are scheduled for reservation.

The gridlets thus, scheduled are reserved by ARM to be executed in future at a specific time. The ARM ensures that the scheduled jobs are allocated resources in advance. It is to be mentioned here that number of jobs to be considered for reservation may vary and is decided by the pre-defined function used by the reservation policy of ARM.

The ARM reservation policy decides the percentage of reservation for incoming jobs out of the total jobs of various users. The jobs can either be reserved immediately or in future as decided by the ARM reservation policy. The scheduled jobs are expected to start their execution on the resources reserved for them by ARM policy. But in practice it may happen that the execution of jobs gets delayed thereby impacting the performance of DPSAR. The effect of this delay on various performance parameters is studied by conducting various simulation experiments.

SIMULATION ENVIRONMENT AND RESULTS

The algorithms for comparison (the proposed DPSAR algorithm and simple Advance Reservation Without Priority (proposed by Anthony *et al.* (2004), referred to as ARWP)) are implemented on a machine based on Pentium 1.6 GHz with 80 GB HDD and RAM of 512 MB on Microsoft Windows XP. The experiments are performed on a simulated grid environment provided by grid sim. The performance model and simulation environment used here is similar to that used.

The simulation environment consists of 5 resources. These 5 resources have 47 PE's in total having different processing capabilities in terms of Millions Instructions Per Second (MIPS). However, PE's belonging to a particular resource have the same processing capability. For example, one of the resources may have 5 PE's with the processing capability of 417 MIPS. Further, it is assumed that there are 50 users with 100 jobs each. These 5000 jobs may join the grid simultaneously and request for reserving the resource. As the number of resources is far less than the number of jobs requesting them, it is imperative that not all the jobs will get the resource reservation. The percentage of jobs getting advance reservation is therefore varied (from 0-30%) in order to study its impact on various performance parameters (discussed below), such as like average waiting time, average offset, number of rejections and resource utilization.

The parameters used for performance comparison in this study are explained as follows:

Average waiting time: Average time for which a gridlet waits in order to get a resource using advance reservation technique.

Average offset time: Average difference between time at which a gridlet has been scheduled to start its execution and the time at which it actually starts its execution on an allocated resource.

Number of rejections: Number of gridlets getting discarded for not getting reservation on any resource.

Resource utilization: Measure of percentage of resource utilization when gridlets are running on a particular resource.

All the performance measures are probabilistic in nature, so value of these variables are taken by repeating the experiments several times. These values are then averaged over as many times as the experiments are performed. In this study, we have repeated the simulation experiments 300 times. All these simulations were carried out in gridsim using eclipse 3.2.

The various performance metrices of DPSAR that have been evaluated are now discussed below:

Average waiting time: Figure 4 shows graphical analysis between average waiting time and percentage of jobs reserved in advance for both DPSAR and ARWP algorithms. It is observed that as the percentage of job reservation is increased, the average waiting time increases for both DPSAR and ARWP. Since, the jobs are reserved for some specific time in future, it is obvious that average waiting time will increase with percentage of job reservation. It is noticed that when the percentage of reservation is varied from 0% (immediate reservation i.e. reservation starting with current time as start time) to 10% there is a sharp rise in average waiting time whereas this rise is not as sharp with the further increase in percentage of reservation. With the use of DPSAR scheduler, the average waiting time increases very insignificantly as compared to the ARWP. This is due to the fact that DPSAR takes priority of the gridlets into consideration before allocating resources.

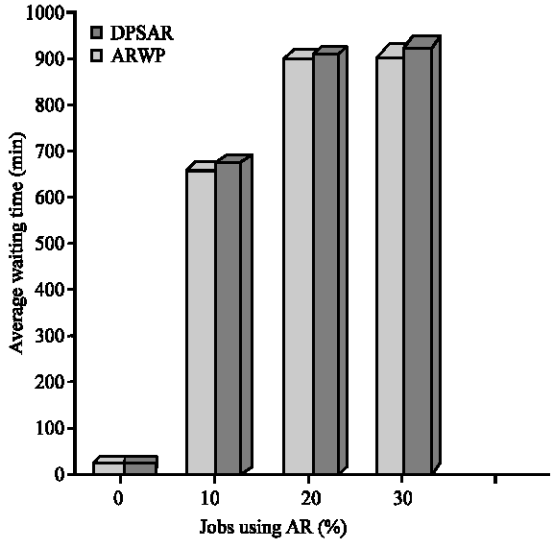

Fig. 4: Graph between average waiting time and percentage of jobs using DPSAR and ARWP

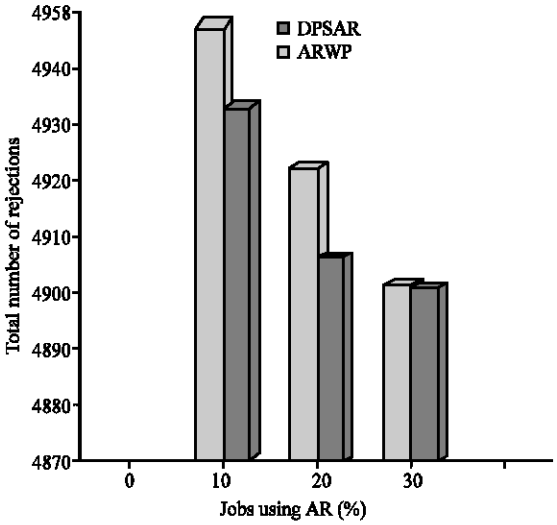

Fig. 6: Graph between total number of rejections and percentage of jobs reserved using DPSAR and ARWP

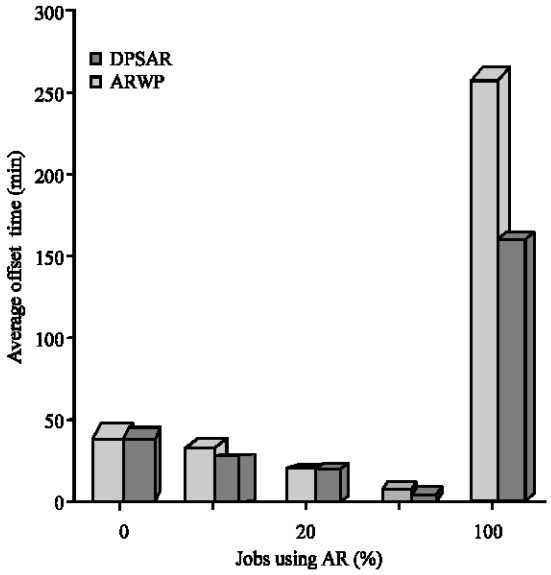

Fig. 5: Graph between average offset time and percentage of jobs reserved using DPSAR and ARWP

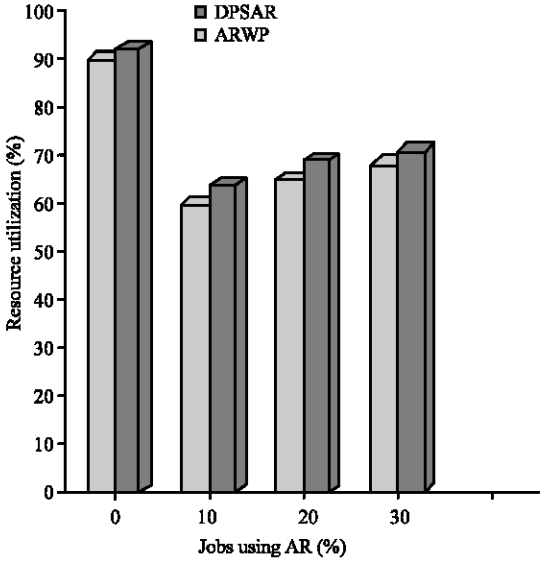

Fig. 7: Graph between percentage utilization and percentage of jobs reserved using DPSAR and ARWP

Average offset time: Graphical analysis between average offset time and percentage of jobs using advance reservation is demonstrated in the Fig. 5. As the percentage of reservation is increased, the average offset time decreases in both DPSAR and ARWP. It is evident, from the results that the performance of DPSAR is better than ARWP as it has lower average offset time indicating better accuracy as the gap between guaranteed start time (while, resrving) and actual execution time has decreased. It is observed that DPSAR not only reduces the average offset time, but also considers the selected preferential jobs for reservation.

Number of rejections: Figure 6 depicts an analysis between number of rejections and the percentage of jobs using advance reservation. As more and more jobs seek reservation, the number of rejections goes up. As seen from the figure, the DPSAR gives improved results over

ARWP. This is due to the reason that the shortest jobs of higher priority users are entertained first by DPSAR. Rather it also handles, the job priorities dynamically at the run time.

Resource utilization: The graph in Fig. 7 shows variation in percentage of resource utilization with respect to the change in percentage of jobs using advance reservation in both DPSAR and ARWP. The resource utilization drops in both the techniques. The resource utilization in DPSAR shows better results when compared to ARWP, as the DPSAR schedules jobs in shortest job first order. DPSAR not only provides committed reservations to the jobs but also takes care of job priorities. Therefore, DPSAR ascertains job reservation to more deserving jobs thereby enhancing the over all utility of resources.

CONCLUSION

The DPSAR proposed in this study does job scheduling and allocation on demand. This allocation is done for a specified time in future. The job allocation takes place only after the dynamic analysis of jobs at run time in terms of their priorities and job lengths. DPSAR by incorporating a pre-defined function also limits the number of jobs of each user to be reserved. This therefore, helps to reduce the problem of starvation for jobs belonging to low priority users.

The performance of DPSAR has also been evaluated by conducting several experiments on a simulation tool called Gridsim. We have also compared the performance of DPSAR with an existing scheduler ARWP (it does not have feature of dynamic priority resolution). It is also, observed that the average offset time in DPSAR has reduced as compared to ARWP. Further, the number of rejections of jobs that do not get reservations also drops in case of DPSAR when compared to ARWP. In DPSAR the resource utilization drops at 10% reservation, but shows slight improvement at 20 and 30%, which is almost at par with ARWP. It is also observed that the average waiting time experiences an upward trend in DPSAR but that is not very significant.

Further, it is to be mentioned that the proposed DPSAR uses shortest job first technique for sorting user jobs. However, DPSAR can also be designed based on other techniques like FCFS or longest job first. Also, the proposed DPSAR reserves the scheduled jobs to different resources in a random manner. A different scheme of allocation of resources to the jobs can be devised and its effect on the performance of DPSAR can be studied as a further extension of this study.

REFERENCES

Buyya, R. and M. Murshed, 2002. Grid Sim: A toolkit for modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computing. Practice and Experience, 14 (13-15): 1175-1220.

Foster, I. and C. Kesselman, 1999. The Grid: Blueprint for a future computing infrastructure. San mateo: Morgan Kauf-mann Publishers.

Foster, I., C. Kesselman and S. Tuecke, 2001. The anatomy of the grid. Enabling scalable virtual organizations. Int. J. High Perform. Comput. Applications, 15 (3): 200-222.

Foster, I., C. Kesselman, J. Nick and S. Tuecke, 2002. The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure, Global Grid Forum.

Karatza, H.D., 2000. A simulation model of backfilling and I/O scheduling in apartitionable parallel system. Simulation Conf. Proc. Winter, 1: 496-506.

Li, X. and M. Parashar, 2005. Adaptive Runtime Management of Spatial and Temporal Heterogeneity for Dynamic Grid Applications. Proceedings of the 13th High Performance Computing Symposium.

Spooner, D.P., S.A. Jarvis, J. Cao, S. Saini and G.R. Nudd, 2003. Local grid scheduling techniques using performance prediction. Computers and digital techniques. IEE Proc., 50 (2): 87-96. http://www.csie.ntu.edu.tw/~ktw/uos/uos-ch6.pdf.

Sulistio, A. and R. Buyya, 2004. A grid simulation infrastructure supporting advance reservation. Proc. 16th Int. Conf. Parallel and Distributed Comput. Syst., pp: 1-7.

Yeo, C.S. and R. Buyya, 2004. A Taxonomy of Market-Based Resource Management Systems for Utility-Driven Cluster Computing, Software Practice and Experience, Published Online in Wiley Inter Science.