HASIL CEK_07 Advance

by 07 Advance

Submission date: 18-May-2022 09:32AM (UTC+0700)
Submission ID: 1838783403
File name: 07 Advance.pdf (103.93K)
Word count: 4918
Character count: 24494

Advance Planning and Reservation for Parametric (MPI) Jobs in a Cluster/Grid System

Rusydi Umar[2], Arun Agarwal[1], C. R. Rao[1]
[1]Department of Computer and Information Sciences
University of Hyderabad
Hyderabad, India
[2]Department of Informatics Engineering
Ahmad Dahlan University
Yogyakarta, Indonesia
rusydi_umar@rocketmail.com, {aruncs, crrcs}@uohyd.ernet.in

ABSTRACT: *Advance Planning and Reservation in a Grid System allows applications to request resources from multiple scheduling systems at a specific time in future and thus gain simultaneous access to sufficient resources for their execution. Existing advance reservation strategy will reject incoming reservation if requested resources are not available at that exact time. Therefore impact of advance reservations is decreasing resource utilization due to fragmentations. This paper proposes a novel advance planning and reservation strategy namely First Come First Serve Ejecting Based Dynamic Scheduling (FCFS-EDS) to increase resources utilization in a grid system for MPI jobs. MPI jobs are jobs that need more than one compute node of their execution and must start at the same time on each of the compute node. To achieve this we introduce a new notion that maps a user job to a virtual compute nodes (called logical view) which are subsequently mapped to actual compute nodes (called physical view) at the time of execution. A lemma ensures the success of such a mapping with increased resource utilization.*

Keywords: Advance Reservation, MPI Job, FCFS-EDS, Resource Utilization, Planning

Received: 1 July 2012, Revised 7 August 2012, Accepted 12 August 2012

1. Introduction

In most grid systems with traditional scheduler, submitted jobs will be placed in the wait queue if mandated resources are not available for them. Every grid system may use a different scheduling algorithm, for example First Come First Serve (FCFS), Shortest Job First (SJF), Earliest Deadline First (EDF), or EASY Backfilling [1] that executes jobs based on different parameters, such as number of resources, submission time, and duration of execution. With these scheduling algorithms, there is no guarantee about when these jobs will be executed [2].

To address the unwarranted waiting time issue and ensure that the specified resources are available for applications at a particular time in the future, we need an advance planning and reservation system [3]. Advance reservations [4] allow a user to request resources from multiple scheduling systems at a specific time in the future and thus gain simultaneous access to enough resources for their applications.

2. Literature Review

Many earlier literatures discuss about advance reservation strategy to increase resource utilization.We have attempted to broadly categorize these advance reservation strategies reported in the literature as follows:

2.1 Rigid Reservation

When users request for compute nodes, they must provide three parameters, start time, end time and number of compute nodes. The reservation system then searches for the availability of requested compute nodes in the specified time interval. If required nodes are unavailable, the request is rejected. This mechanism is known as rigid reservation and has been adopted into the Globus Architecture for Reservation and Allocation (GARA) [5], [6].

2.2 Elastic Advance Reservation

With rigid reservations, there is a shortcoming. If the users still want to reserve compute nodes then they must change the job parameters and again query for reservation until there is a match between the availability of compute nodes and the user's request. This mechanism brings an overhead due to communication traffic. Elastic reservation proposed by Sulistio et al. [7] takes the user request parameters as soft constrain. The reservation system instead of rejecting the request provides alternatives that can be selected by the user. Results show that their on-line strip packing (OSP) method performs better than first fit alternative (FF) and FF performs better than rigid reservation (RR). This strategy has been adopted in GridSim [8].

2.3 Overlapping/Relax Advance Reservation

In this strategy, user jobs are scheduled even if there are reservation violations in the deadlines because of overlapping of jobs. Peng Xiao et al. [9] used the overlapped time slot table, because they believed that applications tend to overestimate the reservation deadline to ensure their completion [10], [11]. They assumed that the real workload tends to overestimate the relative deadline with mean value by 35%. Experimental results show that this strategy can bring about remarkably higher resource utilization and lower rejection rate at the price of slight increase in reservation violation.

Peng Xiao et al. [12] added capacity issue in their relaxed model. They can limit the price of reservation violation as an impact of the relaxed reservation by adjusting parameter v^*.

Sabitha et al. [13] have introduced a relaxed resource advance reservation policy (RARP), which allows reservations to overlap each other under certain condition. But it is successful only in high reservation areas. So to cover both high and low reservation areas, trust is included in reservations. It can have better adaptation in the presence of both low and high reservation area when trust factor is considered.

2.4 Flexible Advance Reservation (Static)

In flexible advance reservation, user jobs are scheduled within given flexible constraints. Moaddeli et al. [14] has examined the impact of backfilling algorithm in a flexible advance reservation. In their examination they distinguished between aggressive backfilling and conservative backfilling. Aggressive backfilling has higher utilization than conservative one.

Kaushik et al. [15] proposed a flexible reservation window scheme. Window size is the difference between the earliest start time of the user request and latest possible start time of the request. By conducting extensive simulations, they conclude that, when the size of the reservation window is equal to the average waiting time in the on-demand queue, the reservation rejection rate can be minimized close to zero and increase resource utilization.

Castillo et al. [16] has proposed a scheduling algorithm to increase system utilization using a concept from computational geometry. Here deadline is equal or larger than the execution time. If the deadline is the same as the length of the job, it is called as immediate deadline, and if the deadline is longer than the execution time, it is known as general job deadline. To schedule the general job deadline, they use the following strategies: Min-LIP, which minimizes the leading idle period; Min-TIP, which minimizes the trailing idle period; First-fit, which returns the first (i.e., earliest) feasible idle period regardless of the sizes of the leading and trailing idle periods; LACT (latest available completion time), where the scheduler assigns an arriving job to the server with the latest completion time that is earlier than the ready time of the new job. The result shows that Min-LIP gives the best system utilization compared to other strategies.

2.5 Flexible Advance Reservation (Dynamic, Physical View)

This strategy takes an advantage of shifting even earlier reservations made (subject to given flexible constraints) to make room for new incoming reservation. Chunming et al. [17] introduced time span when there is a time range for the starting time of the job. This time span is called slack-time. They proposed a novel mechanism called FIRST (Flexible Reservation using Slack Time). The advantage of slack-time is that the starting time of the job can be shifted to improve resource utilization and to reduce rejection rate. If new reservation comes the reservation system reschedules all unexecuted reservation by taking one by one, according to the rule of FIFO, min slack, min-min, min-max, and suffrage policy to see whether there is a solution or not. If there is no solution then the new reservation request will be rejected. The result shows when compared to rigid reservation, FIRST gives better resource utilization and min-min based gives better result than the rest.

In FIRST as mentioned before, every time a new task comes, it reschedules the entire task. According to Netto et al. [18] rescheduling the task will have a better system utilization if the user can wait until 75% of waiting time has been passed, compared to 50% and 25%. They observe that the longer a user waits to fix their jobs the better the system utilization. They use five different sorting techniques: Shuffle, First in First out (FIFO), Biggest Job First (BJF), Least Flexible First (LFF), and Earliest Deadline First (EDF). They concluded that EDF technique has the best result in increasing resource utilization.

Behnam et al. [19] have introduced scheduling algorithm for advance reservation called GELSAR (Gra-vitational Emulation Local Search Advance Reservation) in grid system where the reservations can have a deadline (*dj*) which can be equal or larger than a ready time (*rj*) + length of the reservation (*lj*). The Idea is to imagine that the search space is the universe and objects in this universe are the possible solutions. Every time the new reservation comes the GELSAR reschedule all reservation to find the best solution. If there is no solution the new incoming reservation is rejected. They compared their algorithm with other rescheduling algorithm (Genetic Algorithm, GA) and the found the result that their algorithm outperformed GA algorithm.

3. Proposed Advance Planning and Reservation Strategy

We propose a novel advance planning and reservation strategy namely First Come First Serve Ejecting Based Dynamic Scheduling (FCFS-EDS) to increase resources utilization in a grid system. To achieve this we introduce a new notion that maps a user job to a virtual compute nodes (called logical view) which are subsequently mapped to actual compute nodes (called physical view) at the time of execution. A lemma ensures the success of such a mapping with increased throughput. This approach can be categorized under Flexible Advance Reservation (Dynamic, Logical View).

3.1 Definition of Flexible Advance Reservation

Let us define "*flexible advance reservation*" as follows: "*Flexible advance reservations are reservations where the time between reservation request starting time* (t_{es}) *and reservation request end time* (t_d) *is longer than the execution time* (t_e) *of a job*" as shown in Figure 1.

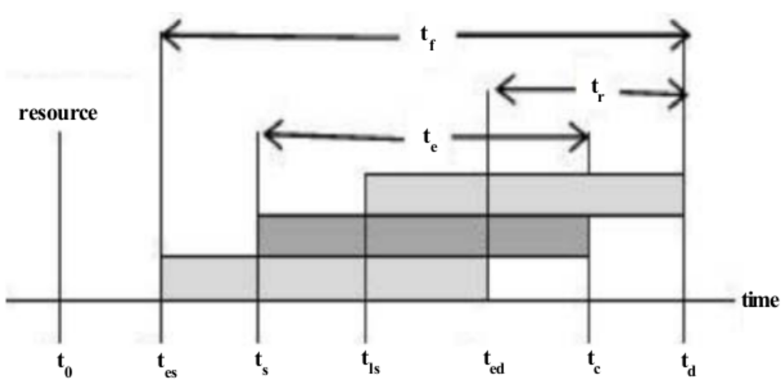

Figure 1. Flexible Advance Reservation

t_0 : current time

t_{es} : lower bound to start time of the job

t_d : upper bound to end time to execute the job

t_e : execution time of the job

t_r : relax time, define by $t_r = t_d - t_{es} - t_e$

t_{ls} : upper bound to start execution of the job, defined as $t_{ls} = t_d - t_e = t_{es} + t_r$

t_{ed} : lower bound to end time to execution of the job, defined as $t_{ed} = t_{es} + t_e$

t_s : start time to execute the job ($t_{es} \leq t_s \leq t_{ls}$)

t_c : completion time to execute the job ($t_{ed} \leq t_c \leq t_d$)

t_f: flexibility time, defined as $t_f = t_d - t_{es}$

f : the degree of flexibility, define as $f = \frac{t_f}{t_e}$, here $f \geq 1$, (if $f = \infty$, the job considered as non reservation job mode. If $t_s = t_0$ and f=1 the job for reservation considered with top most priority leading to immediate reservation mode i.e., scheduling mode)

UserId : identification of a user

jobId : identification of a job

numJ : number of jobs

numCN : number of compute nodes needed

maxCN : total number of compute nodes

3.2 FCFS-EDS Strategy

This strategy takes an advantage of shifting earlier reservations made (subject to given flexible constraints) to make room for new incoming reservation request. However the planning for reservation in our proposed model provides a logical view as against the physical view reported in the literature. Therefore we call our method as First Come First Serve-Ejecting Based Dynamic Scheduling (FCFS-EDS) strategy.

Let us assume that earlier "*n* - 1" reservation requests made to FCFS-EDS, specified by the following parameters: t_{es}, t_{ls}, t_e, user*Id, jobId, numCN* have been successfully scheduled. Definition of variables is explained in lines 4-9. Lines 10–14 initialize the variables.

Incoming "n^{th}" reservation request is scheduled based on first fit strategy (iteration of line 16-25). It tries to search for resources within the given constraints (t_{es}, t_{ls} and *numCN*) and without disturbing plan for previous "*n* - 1" reservation requests that were made. Let us call this as plan P_{old}. If within the given flexible constraints the search fails to allocate resources the algorithm tries to move around pre-vious "*n* - 1" reservations to accommodate "n^{th}" reservation request (lines 31-44). If the resources are found then the search is declared successful and the algorithm outputs a new plan P_{new} that depicts "*n*" reservation requests as a logical view. If the search is a failure then the "*n* - 1" reservation request plan has to be restored to its previous state i.e. P_{old}.

The time complexity of FCFS-EDS algorithm is $O(n \,.\, m)$ where n is t_r and m is t_s.

Algorithm FCFS-EDS

```
1 Function searchAndAlloc(userId, jobId, tes, tls, te, numCN : integer)→boolean
2 //search and allocate job with given tes, tls, te, numCN
3 Dictionary :
4 start : integer /*start time of the job*/
5 finish : integer /*finish time of the job*/
6 min : integer /*min free within interval start - finish*/
```

```
t : integer /*timeslot of minimum available node between start to finish*/
tr : integer /*relax time, length between of tes and tls*/
relax : integer /*different between start and tes time (start - tes + 1)*/
Algorithm :
tr ← tls – tes
succeed ← false
start ← tes
finish ← tes + te – 1
relax ← start - tes

while (!succeed and (relaxd ≤ tr)) do /*searching by first fit strategy*/
   /*searching minimum free node between start to finish*/
   min,t ← minFreeNode(start, finish)
     if(min ≥ numCN) then
       allocate(userId, jobId, tes, start, tls, te, numCN)
       succeed ← true
     else
       start ← t + 1
       finish ← start + te - 1
       relax ← start - tes
/*end while, the state is succeed=true or succeed=false (relax > tr)*/

start ← tes
finish ← tes + te – 1
relax ← start - tes
while (!succeed and (relaxdetr)) do
   /*searching minimum free node between start to finish*/
   min,t ← minFreeNode(start, finish);
   if(min ee numCN) then
      /*push or schedule the job to data structure using our lemma below
      and update free node between start to finish*/
      allocate(userId, jobId, tes, start, tls, te, numCN)
      succeed ← true
   else
     /*try to shift a job that start at t time slot*/
     if(!shiftNode(t, numCN-min)) then //can't be shifted, start equal t+1
        start ← t + 1
        finish ← start + te - 1
        relax ← start - tes
/*end while, the state is succeed=true or succeed=false (relax > tr)*/
if (!succeed)
   putBackAllShiftedJob()
return succeed
```

3.3 Illustration

To explain how FCFS-EDS works we introduce an example, If we have *maxCN* compute node (physical view), let *maxCN* is equal to 5 (*cn*0 - *cn*4) as physical nodes, then the number of virtual node (*v*0 - *v*4) is also 5 (*v*0 - *v*4). Sequence of incoming reservation is depicted in Table 1, where *numCN* ≤ *maxCN* and *numJ* are the number of jobs submitted by a *userId*. For example the given parameters for *userId* = 3 in the Table 1 implies the following: "*User* 3 *reserved* 3 *time slots at time slot* 12 *up to* 14, *needed* 1 *compute node for* 1 *independent job, and cannot be delayed/shifted.* (t_{es} , t_{ls} =12, t_e = 3, *numCN* = 1, *numJ* = 1)".

The result of FCFS-EDS is a plan shown in Figure 2, where *x* axis indicates time slot, and *y* axis indicates virtual compute node (**Logical View**). As there are 5 virtual compute nodes, they are shown along the *y* axis as *v*0, *v*1, *v*2,*v*3, and *v*4. The ten user reservations have been allocated during time slots 11 to 17. Consider *userId* = 9 from Table 1. The virtual computes nodes allotted to it are *v*4 at time slot 11(t_{es} = 11) and *v*3 at time slot 12 as only 1 job (requiring 2 execution time slots) has been submitted by this user.

userId	t_{es}	t_{ls}	t_e	*numCN*	*numJ*
1	11	11	1	2	1
2	11	11	3	1	1
3	12	12	3	1	1
4	15	16	1	1	1
5	15	15	1	2	1
6	11	11	2	1	1
7	13	13	1	1	1
8	16	16	2	1	1
9	11	11	2	1	1
10	15	15	3	1	1

Table 1. Parameters of reservation request

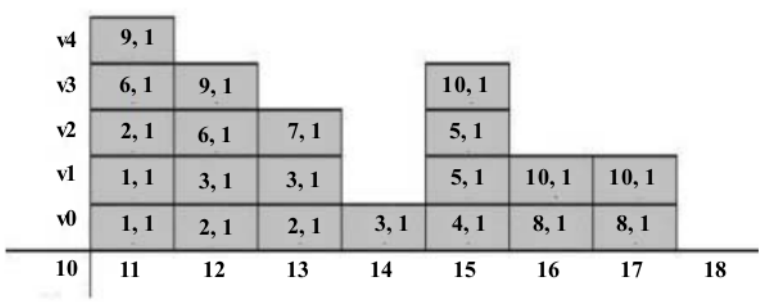

Figure 2. Ten reservations have been allocated (Logical View)

Suppose User 11 wishing to reserve 3 time slots from 12 up to 14, needs 2 compute nodes for 1 indepen-dent job and can be delayed up to time slot 14 (t_{es}= 12, t_{ls}= 14, t_e= 3, *numCN* = 2, *numJ* = 1). See Figure 3.

The result of FCFS-EDS is a plan for the new incoming reservation request from user 11 which is depicted in Figure 4. Using conventional reservation or rigid reservation job from user 11 will be rejected. From Figure 4 it is seen that the same job on different time slot is allocated on different virtual compute nodes. On successful reservation a notification is sent to the user only once (in our approach as we are working at a logical view) whereas in other approaches it has to sent every time a revision is made in the plan (P_{new}) (physical view: reassignment of physical resources) [18,19].

3.4 Mapping of the Plan (Logical View) to Actual Compute Node (Physical View)

We introduce a Lemma to guarantee that the plan (logical view) can always be mapped into actual compute node (physical view), and once a job is started at certain compute node, then it will be executed on the same compute node for the entire time slot. Applying the lemma to the plan as depicted in Figure 4, we guarantee that all the jobs will be executed as shown in Figure 5 (Physical View).

The concept of proof of our advance planning and scheduling strategy is assured due to the following Lemma:

Lemma: *If one can plan for scheduling a job on the consecutive time slot on virtual compute nodes (which are selected freely) (Logical View) then it guarantees that the job will be executed on a dedicated physical node for the required execution time*

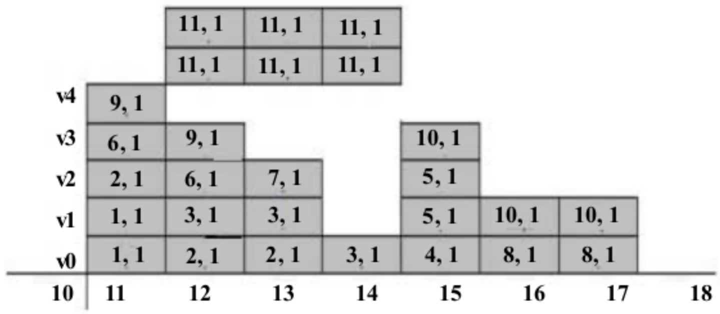

Figure 3. User 11 makes a request for reservation

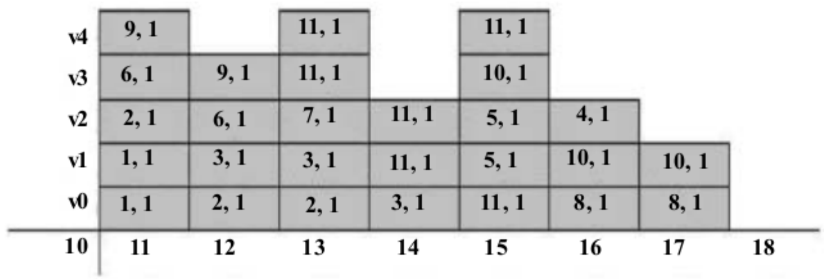

Figure 4. User 11 has been allocated using FCFS – EDS (Logical View)

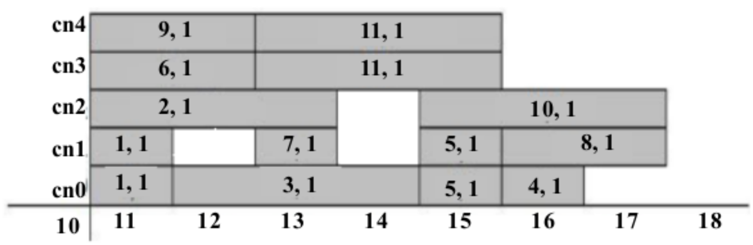

Figure 5. Actual Compute Node Mapping (physical view) at $t_0 = 18$

(*Physical View*).

Proof: Let $J(t)$ be is an array of size *maxCN* (maximum number of compute node) scheduling plan at time slot t. i^{th} element of $J(t)$ is the job id that is assigned to i^{th} compute node at time slot t.

Let $JS(t)$ be the list (set) of jobs planed for time slot t associated to $JS(t)$. Then

$JS(t) - JS(t+1)$ indicates the list (set) of job finishes at time slot t

$JS(t+1) - JS(t)$ indicates the list (set) of job start at time slot $t+1$

$JS(t) \cap JS(t+1)$ indicates list (set) of a job continuing from time slot t to $t+1$

A_M is t to $t+1$ assignment matrix of order *maxCN* (of zeros and ones). $A_M(i, j) = 1$ indicates that job at time slot t is executed at compute node i and at time slot $t + 1$ is executed at compute node j.

A_M is a partial permutation matrix. One can construct complete permutation matrix, say C_M (which is a non singular matrix).

Let C be equal to C_M then the multiplication between C_M and C is I

Thus the multiplication between A_M and C is PI

If the A_M is Partial Identity (PI) matrix,

Then the job at time t will be executed at the same compute node at time slot $t + 1$.

Else treat advance scheduling plan for t+1 time slot as by multiply J (t + 1) with C (one can easily verify that A_M for this will be PI).

4. Experiment

The comparison of our scheduling strategy FCFS-EDS is depicted in Table 2. It presents parameters that have been used by other researchers. The rows indicate possible parameter and column the scheduling strategies. The entries are notations/ symbols, policy and name wherever provided by the authors otherwise dash has been used. However, there is no clarity on whether their strategies accept MPI type of jobs. Our proposed strategy accepts those jobs and details are pointed below.

User requests, which are input for our FCFS-EDS, are generated randomly. The input specifications are:

a. The rate of incoming reservation requests are assumed to follow poison distribution with mean 2.0,

b. Execution time (t_e) for reservation requests are between 5 to 15 timeslots distributed uniformly,

c. Earliest starting time (t_{es}) is between 0 to 24 timeslots distributed uniformly,

d. Percentage of user request that are for flexible advance reservation is assumed to at most 50% (selected randomly),

e. Relax time (t_r) is between 1 to 12 timeslots distributed uniformly and $t_{ls} = t_{es} + t_r$

f. Number of compute node needed ($numCN$) is between 1 to 5 compute nodes and distributed uniformly,

g. In the experiment it is assumed that a time slot is equal to 5 minutes (clock time).

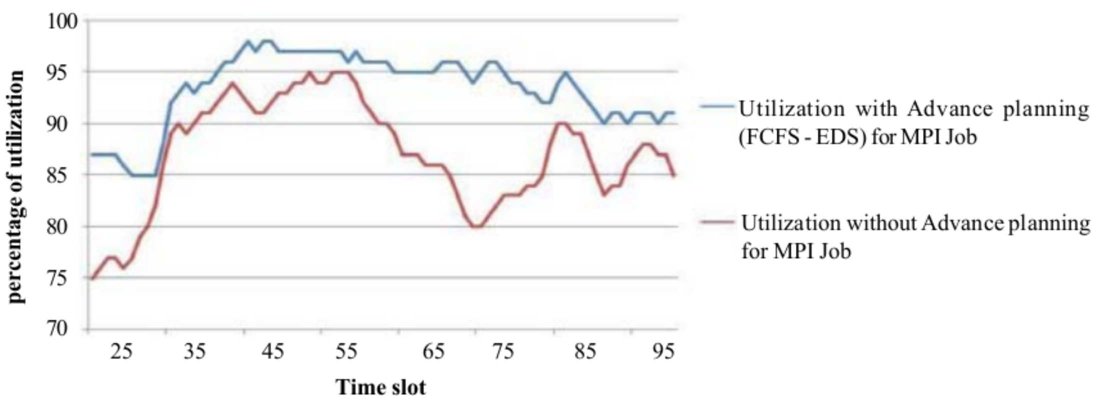

Figure 6. Comparison of percentage of utilization factor between scheduling with Advance Planning (FCFS – EDS) and flexible advance reservation without Advance Planning

We compare the performance of our method (FCFS-EDS with logical view) and one with only physical view. With above inputs and total number of compute node is 30 ($numCN$ = 30), the utilization factors of both strategies are measured. The comparison of resource utilization of both strategies is shown in Figure 6. Percentage of utilization factor is calculated within sliding window of size 12 timeslots (1 hour). Figure 6. shows that FCFS-EDS yields better utilization than the traditional strategy (with only physical view). We have compared the performance of the proposed method [20] (FCFE-EDS with logical advance planning) and an existing approach (flexible advance reservation strategy without advance planning) for parametric jobs and the data structure used is reported in [21]. The results showed that FCFS-EDS for parametric jobs, yields better utilization than the traditional strategy (without advance planning).

	Our work	Sulistio et al.	Castillo et al.	Kaushik et al.	Xiao et al.	Chunm-ing et al	Netto et al.	Behnam et al.	Moadelli et al.
Current Time	t_0	-	-	-	t_0	-	-	t = 0	-
Time of the earliest start time of the job	t_{es}	t_{is}	r_j	-	t_{start}	t_{is}	t_r	r_j	t_{rt}
start time to execute the job	t_s	-	-	t_{start}	-	t_{start}	t_s	-	t_{st}
completion time to execute the job	t_c	-	-	t_{end}	-	-	t_c	-	t_{ct}
the end time to execute the job	t_d	t_{ie}	d_j	-	-	-	d	d_j	t_{dt}
execution time of the job	t_e	dur	lj	dur	d	duration	t_e	L_j	t_{ad}
Relaxed time	t_r	-	-	W	-	slack	-	-	-
Degree of flexibility	f	-	-	-	-	-	-	-	-
Compute nodes needed	n	n	1	1	1	1	N		N
Name	flexible	elastic	general job deadline	unconstrained /flexible	relaxed	flexible	flexible	-	Flexible
Scheduling	FCFS-EDS	FCFS-OSP	FCFS Min-Lip	FCFS	FCFS	Min-min	EDF	GELSAR	FCFS

Table 2. Comparison between other scheduling strategy

5. Conclusion

In this paper we proposed a novel advance planning and reservation strategy FCFS-EDS for MPI jobs to increase resource utilization in a grid system. This strategy maps the job to a virtual node (logical view) and the Lemma guarantees that these jobs will be assigned to compute resources (physical view) for execution and will achieve higher resource utilization. We also experimentally compared the performance of the proposed method (FCFS-EDS with advance planning and reservation for MPI jobs) which showed better performance than an existing approach (flexible advance reservation strategy without advance planning).

References

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

[2] Sulistio A., Buyya, R. (2004). A Grid simulation infrastructure supporting advance reservation. *In*: 16th International Conference on Parallel and Distributed Computing and Systems, p. 1-7, ACTA Press. Calgary.

[3] MacLaren, J. (2003). Advance Reservations: State of the Art. *In*: Working Draft, Global Grid Forum.

[4] Smith, W., Foster, I., Taylor, V. (2000). Scheduling with Advanced Reservations. *In*: 14th IEEE International Symposium on Parallel and Distributed Processing, p. 127-132, IEEE Press, Cancun

[5] Foster, I., Kesselman, C., Lee, C., Lindell, B., Nahrstedt, K., Roy, A. (1999). A Distributed Resource Management Architecture that Supports Advance Reservation and Co-Allocation. *In*: 7th IEEE International Workshop on Quality of Service, p. 27-36, IEEE Press, London.

[6] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., Tuecke, S. (1998). A resource management architecture for metacomputing systems. *In*: 4th Workshop on Job Scheduling Strategies for Parallel Processing. LNCS 1459, p. 62-82. Springer, London

[7] Sulistio, A., Kim, K. H., Buyya, R. (2007). On Incorporating an On-line Strip Packing Algorithm into Elastic Grid Reservation-based Systems. *In*: 13th International Conference on Parallel and Distributed Systems (ICPADS'07), p. 1-8, IEEE Press, Hsinchu

[8] Buyya, R., Murshed, M. (2002). GridSim: A Toolkit for the Modeling and Simulation of Distributed Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience. 14, 1175-1220

[9] Xiao, P., Zhigang, H., Xi, L., Liu, Y. (2008). A Novel Statistic-based Relaxed Grid Resource Reservation Strategy. *In:* 9th International Conference for Young Computer Scientists, p. 703-707. IEEE Press, Hunan

[10] Lee, C. B., Snavely, A. (2006). On the user-scheduler dialogue: Studies of user provided runtime estimates and utility functions, *International Journal of High Performance Computing Applications.* 20, 496-506

[11] Castillo, C., Rouskas, G., Harfoush, K. (2011). Online algorithms for advance resource reservations, *Journal of Parallel and Distributed Computing.* 71, 963-972

[12] Peng, X., Zhigang, H. U. (2009).Relaxed resource advance reservation policy in grid computing, *The Journal of China Universities of Posts and Telecommunications. 1*6, 108-113

[13] Sabitha R. B. S., Venkatesan, R., Ramalakshmi, R. (2011). Resource Reservation In Grid Computing Environments: Design Issues. *In:* 3rd International Conference on Electronics Computer Technology, p. 66-70, IEEE Press, Kanyakumari

[14] Moaddeli, H. R., Dastghaibyfard, G., Moosavi, M. R. (2008). Flexible Advance Reservation Impact on Back-filling Scheduling Strategies . *In*: 7th International Conference on Grid and Cooperative Computing, p. 151-159, IEEE Press, Shenzhen

[15] Kaushik, N. R., Figueira, S. M., Chiappari, S. A. (2006). Flexible Time-Windows for Advance Reservation Scheduling. *In*: 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, p. 218-225, IEEE Press, California

[16] Castillo, C., Rouskas, G., Harfoush, K. (2011). Online algorithms for advance resource reservations. *Journal of Parallel and Distributed Computing*. 71, 963-972

[17] Chunming, H., Jinpeng, H., Tianyu, W. (2006). Flexible Resource Reservation Using Slack Time for Service Grid. *In*: 12th International Conference on Parallel and Distributed Systems, p. 327-334, IEEE Press, Washington

[18] Netto, M. A. S., Bubendorfer, K., Buyya, R. (2007). SLA-based advance reservations with flexible and adaptive time QoS parameters. *In*: 5th International Conference on Service-Oriented Computing, LNCS 4749, p.119-131. Springer, Heidelberg

[19] Behnam, B., Amir, M. R., Kamran, Z. F., Azedah, D. (2001). Gravitational Emulation Local Search Algorithm for Advanced Reservation and Scheduling in Grid Computing Systems. *In*: 4th International Conference on Computer Science and Convergence Information Technology, p. 1240-1245, IEEE Press, Seoul

[20] Rusydi Umar, Arun Agarwal, CR Rao. (2012). Advance Planning and Reservation in a Grid System. *In*: The Fourth International Conference on Networked Digital Technologies. CCIS/LNCS 293, p. 161-173. Springer, Dubai

[21] Rusydi Umar, Arun Agarwal, CR Rao.(2012). Data Structure for Advance Planning and Reservation in a Grid System. *International Journal of Computer Applications* (IJCA). NCRTC, 33-37

HASIL CEK_07 Advance

ORIGINALITY REPORT

17%	12%	13%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.cloudbus.org Internet Source	1%
2	www.ijcaonline.org Internet Source	1%
3	Peng XIAO, Zhi-gang HU. "Relaxed resource advance reservation policy in grid computing", The Journal of China Universities of Posts and Telecommunications, 2009 Publication	1%
4	Ojonukpe Sylvester Egwuche, Olumide Sunday Adewale, Samuel Adebayo Oluwadare, Oladunni Abosede Daramola. "Enhancing Network Life-Time of Wireless Sensor Networks through Itinerary Definition and Mobile Agents for Routing among Sensor Nodes", 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), 2020 Publication	1%
5	www.inderscienceonline.com Internet Source	1%

6	www.csc.ncsu.edu Internet Source	1%
7	Submitted to University of Bedfordshire Student Paper	1%
8	rouskas.csc.ncsu.edu Internet Source	1%
9	cloudbus.org Internet Source	1%
10	faculty.kfupm.edu.sa Internet Source	1%
11	students.engr.scu.edu Internet Source	1%
12	"Grid Resource Management", Springer Science and Business Media LLC, 2004 Publication	<1%
13	Behnam Barzegar. "Gravitational emulation local search algorithm for advanced reservation and scheduling in grid systems", 2009 First Asian Himalayas International Conference on Internet, 11/2009 Publication	<1%
14	Chunming Hu, Jinpeng Huai, Tianyu Wo. "Flexible resource reservation using slack time for service grid", 12th International Conference on Parallel and Distributed Systems - (ICPADS'06), 2006	<1%

	Publication	
15	pdfs.semanticscholar.org Internet Source	<1 %
16	"Applied Parallel Computing. State of the Art in Scientific Computing", Springer Science and Business Media LLC, 2007 Publication	<1 %
17	nozdr.ru Internet Source	<1 %
18	Manuel Ammann, Michael Verhofen. "The Effect of Market Regimes on Style Allocation", Financial Markets and Portfolio Management, 2006 Publication	<1 %
19	dcis.uohyd.ac.in Internet Source	<1 %
20	www.researchgate.net Internet Source	<1 %
21	Communications in Computer and Information Science, 2011. Publication	<1 %
22	seminar.ilkom.unsri.ac.id Internet Source	<1 %
23	Mondol, Md. Abu Sayeed, and Md. Mostofa Akbar. "A new approach to schedule workflow applications for advance reservation of	<1 %

resources in grid", International Journal of Grid and Utility Computing, 2014.
Publication

24	hdl.handle.net Internet Source	<1%
25	journalofcloudcomputing.springeropen.com Internet Source	<1%
26	www.amrita.edu Internet Source	<1%
27	Kleopatra Konstanteli, Tommaso Cucinotta, Theodora Varvarigou. "Optimum allocation of distributed service workflows with probabilistic real-time guarantees", Service Oriented Computing and Applications, 2010 Publication	<1%
28	Peng Xiao, Zhigang Hu, Xi Li, Liu Yang. "A Novel Statistic-based Relaxed Grid Resource Reservation Strategy", 2008 The 9th International Conference for Young Computer Scientists, 2008 Publication	<1%
29	Sudha Srinivasan. "Robust scheduling of moldable parallel jobs", International Journal of High Performance Computing and Networking, 2004 Publication	<1%

epdf.pub

30	Internet Source	<1%
31	mts.intechopen.com Internet Source	<1%
32	"Service-Oriented Computing – ICSOC 2007", Springer Science and Business Media LLC, 2007 Publication	<1%
33	"High Performance Computing and Communications", Springer Science and Business Media LLC, 2007 Publication	<1%
34	C. Castillo, G.N. Rouskas, K. Harfoush. "Online algorithms for advance resource reservations", Journal of Parallel and Distributed Computing, 2011 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		