# Advance Reservation for Parametric Job on Grid Computing

[1,2]Ardi Pujiyanta
*Electrical Engineering and Information*
*Universitas Gadjah Mada*
*Yogyakarta, Indonesia*
ardi.pujiyanta@mail.ugm.ac.id
[2]Information Engineering
Universitas Ahmad Dahlan
Yogyakarta, Indonesia

[3]Lukito Edi Nugroho
*Electrical Engineering and Information*
*Universitas Gadjah Mada*
*Yogyakarta, Indonesia*
lukito@ugm.ac.id

[4]Widyawan
*Electrical Engineering and Information*
*Universitas Gadjah Mada*
*Yogyakarta, Indonesia*
widyawan@ugm.ac.id

***Abstract*— Parametric jobs are similar jobs, differing only in arguments or input/output files. With the parametric type of job, most jobs can send as a single job. In science and engineering, parametric computing becomes very important as a means to explore the behavior of complex systems. Users can request resources to run jobs that they send in the future. The scheduler then looks for the availability of the requested resource in a predetermined time interval. If the required resources are not available, the request rejected. The flexible FCFS-LRH algorithm is proposed to overcome the amount of rejection, from the experimental results for scheduling parametric jobs, and it found that the FCFS-LRH advance reservation algorithm is better than without advance reservation.**

***Keywords—advance reservation, FCFS-LRH, grid computing, parametric job, scheduling.***

## I. INTRODUCTION

In recent years, increasing interest in technology integration, analysis, operation, and control of power systems has made it even more complicated. One solution to integrate large systems using grid computing. Computational resources from various geographical and administrative locations to complete shared tasks can be combined using grid computing[1]. Grid computing is derived from a new computing infrastructure for research and scientific cooperation that contains resources of a different nature and becomes a technology built to share large-scale distributed and integrated resources. Network development intended for diverse uses with efficient management, geographically distributed, and the availability of dynamic computing resources[2], is quickly becoming a significant research objective to offer users transparent access to resources. Transparency is a reason to use the term "Grid" which refers to the Electricity Network which only provides the demand for electrical power to all users, without requiring more insight into how and where the actual control has generated.

Similarly, grid computing provides computational power on demand, for all users without knowledge of the location of the allocated resource. In general, the grid usually denoted as sharing resources that are distributed geographically and are owned by different service providers and are in different administrative domains. Parametric jobs are similar jobs, differing only in arguments or input/ output files. The user can submit most types of parametric jobs as a single job. In experimental science and techniques, parametric computing becomes very important as a means to explore the behavior of complex systems. A grid system with traditional scheduling, jobs are sent and will be placed in a queue, waiting for resources to ordered, whether available or not. The scheduling algorithm used in a grid system can vary, for example, FCFS, SJF, EDF[3], by executing jobs based on different parameters, such as the number of resources, delivery time, and execution time. The traditional scheduling algorithm does not guarantee when a job will be carried out[4]. In rigid scheduling, when a user requests resources to carry out his work, three parameters are required, namely start time, execution time, and the number of resources[5]. The scheduler will look for the availability of resources needed in a predetermined time interval. If the source is not available, then the request will be rejected. If this happens often, the scheduler works hard to handle the same user request because the previous request rejected. In the end, it will cause resources to become idle between jobs. Hence resource utilization will decrease. This mechanism is known as rigid reservation (GARA)[6][7]. An elastic reservation is proposed [8] by taking user request parameters as a soft constraint. The reservation system instead rejects the request but provides choices to the user.

In his research [9] introducing slack-time, slack-time is the period for the start time of work, this new mechanism called FIRST (Flexible Reservation using Slack Time). An independent group of jobs [10] will schedule on the local scheduler scheduling, with restrictions on processing time (execution time, earliest start time, number of nodes) given by the user. All processing nodes are assumed to be identical, and the workload consists of batch jobs that require space-sharing execution. From the background of the problem, the proposed reservation scheduling model, in the Local Scheduler (LS) environment and work is an independent job, to overcome and ensure the availability of resources at a specific time in the future, thus providing guarantees that user jobs will be carried out.

## II. SCHEDULING PROPOSED MODEL

### *A. Proposed scheduling model*

In a flexible prior reservation, the user's work scheduled within flexible limits. Start time can vary within certain time intervals[11][12]. A flexible reservation is a reservation where the earliest request ($t_{esr}$) start time, and the last request request request ($t_{esl}$) is longer than the execution time ($t_e$) of a job, As shown in the timing diagram in Fig. 1. How requests handled. Job requests sent with parameters (JumCN, $t_{esr}$, $t_{lsr}$, $t_e$). After the job request is received, the scheduler will look for whether there is free space, if there is, the job will execute,

and resources will allocate. The time difference between $t_{lsr}$ and $t_{esr}$ is called the notification interval[13].

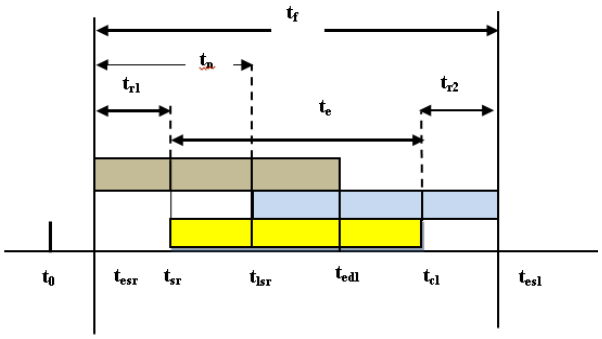


Fig 1. Proposed Flexible Scheduling of Advance Reservations.

$t_0$: Current Time
$t_{esr}$: lower limit for starting time from a job(the earliest start time)
$t_{sr}$: Time to start job($t_{esr} \leq t_{sr} \leq t_{lsr}$)
$t_{lsr}$: The upper limit for starting job execution (last start time), is defined as $t_{lsr}=t_{esl}-t_e=t_n$
$t_{esl}$: The upper limit to end the time to run the job
$t_e$: Time of execution of job
$t_n$: Notification time[13]
$t_{r1}$,$t_{r2}$ : $t_{r1}$(*left hole*), $t_{r1}$(*right hole*), $t_r$(Relax time), defined by $t_r = t_{r1} + t_{r2} = t_{resl} - t_{rsr} - t_e$
$t_{edl}$: lower limit until the time to end the execution of a job, defined as $t_{edl}=t_{esr}+t_e$
$t_{cl}$: Time to complete job ($t_{edl} \leq t_{cl} \leq t_{esl}$)
$t_f$ : Time of flexibility, defined as $t_f = t_{esl} - t_{esr}$
f : Level of flexibility, defined as $f = \frac{t_f}{t_e}$, where f≥1, (if f=∞, jobs considered a non-job reservation mode, if $t_{esr}=t_0$ and f=1 job found with the highest priority leading to direct scheduling mode)[14]

### *B. Proposed Advance Planning and Scheduling Strategies*

In this research, an advance reservation scheduling strategy called First Come First Serve Left Right Hole Scheduling (FCFS-LRH) to increase resource utilization on the grid system. Jobs sent by users will be mapped from the virtual computing node (called logical view) to the actual computing node (called physical view) at the time of execution. For example, determined:
p(t): job permutation matrix in timeslot t. $p^{-1}(t+1)$: permutation inverse matrix at timeslot t+1. Then B=p(t) x $p^{-1}(t+1)$, where B is a partial identity matrix (indicating that the job will be done on the same computational node from timeslot t to t+1). If B is not a partial identity matrix, work planning will come in time slot t+1.
For example, G(t+1) is a row vector at timeslot t+1. Then it can be determined[15]:
If G(t+1)(j)=G(t)(i) then
  B(i,j)=1
Else
  B(i,j)=0
B(i,j)=1 indicates that the work at time slot t has executed on the compute node i, and the work at time slot t+1 has executed at the compute node j.
If given LG(t) is a set of work plans with a time slot t at G(t), then:
LG(t)-LG(t+1) is a set of jobs done in slot t.
LG(t+1)-LG(t) is a set of jobs done in slot t+1.
LG (t)∩LG(t+1) is a set of jobs done in slot t until slot t+1.

### *C. Algorithm*

The algorithm explanation below is as follows, lines 2 through 15 show declaration and initialization. User submits (tesr,tlsr,texe,JumCN), an empty timeslot will be searched between tesr to finishR, using the first fit strategy shown in lines 12 to 26. If an empty time slot is found then the job is placed in a virtual view, notifying the user the job has received. If in the search is not found an empty slot between the tesr to finishR, the algorithm will move the old allocated job by shifting it, to make space, so that new jobs can be inserted, shown in lines 28 to 43. If new jobs cannot insert, return it Old jobs that have shifted to their original place shown in lines 44 to 46. Notify user job rejected.

```
1 Function  Allocate(userId,tesr,tlsr,texe,jumCN)→boolean
2 tesr: The lower time limit starts from the job.
3 tlsr: The upper limit of time starting from the job
4 texe: Job Execution Time
5 finishR: time the job ends
6 minSlot: time shift between tlsr and tesr
7 flex: the difference between start and tesr
8 // Declaration and Initialization
9  minSlot←0;
10 minSlot←tlsr-tesr;
11 fix←false;
12   If (!fix) then
13        startR←tesr;
14        finishR←tesr+texe-1;
15        flex←startR-tesr;
16      while (flex<=minSlot and !fix)
17            minSlot←searchNode(startR, finishR);
18            If(minSlot>0) then
19               assign(userId,tesr,startR,tlsr,texe);
20               fix←true;
21            else
22        startR←time+1;
23              flex=startR-tesr;
24              finishR←startR+texe-1;
25            endif
26        endwhile
27   endif
28          startR←tesr;
29          finishR←tesr+texe-1;
30          flex←startR-tesr;
31        while(flex<=minSlot and !fix)
32          minSlot=searchNode(startR,finishR);
33          If(minSlot>0) then
34            assign(userId,tesr,startR,tlsr,texe);
35            fix←true;
36          else
37            If(!insRes(userId,jumCN-minSlot) then
38              startR←time+1;
39              finishR←startR+texe-1;
40              flex←startR-tesr;
41            endif
42          endif
43        endwhile
```

```
44            If(!fix) then
45              backJob();
46            endif
47      return fix;
48 endFunction
```

After a job has been successfully allocated to a logical view, the job can always be executed at the actual node, using the algorithm[15]

## III. FCFS-LRH PARAMETRIC JOB

### A. How it works Parametric Job

The following example will give how the parametric job works. If the system grid has computed nodes (physical view), as many as MaxC=5 (C0-C4), then we have a virtual node (Logical View) of 5 (V0-V4) as well. The order of arrivals can show in Table 1, Where JumCN≤MaxC and JumJob are the numbers of jobs sent by UserId. For example, given the parameter UserId3 in table 1. User3 orders 4 timeslot starting from timeslot 3 to timeslot 6, requires 1 compute node, for 1 independent job.($t_{esr}$=$t_{lsr}$=3, $t_e$, jumCN=1, jumJob=1).

TABLE I. JOB REQUESTS FROM PARAMETRIC JOB

| UserId | $t_{esr}$ | $t_{lsr}$ | te | JumCN | JumJob |
|---|---|---|---|---|---|
| 1 | 3 | 3 | 2 | 1 | 1 |
| 2 | 3 | 3 | 3 | 2 | 2 |
| 3 | 3 | 3 | 4 | 1 | 1 |
| 4 | 3 | 3 | 2 | 1 | 1 |
| 5 | 6 | 6 | 1 | 1 | 1 |
| 6 | 6 | 6 | 5 | 1 | 1 |
| 7 | 7 | 7 | 2 | 1 | 1 |
| 8 | 9 | 9 | 1 | 1 | 1 |
| 9 | 9 | 9 | 2 | 1 | 1 |
| 10 | 10 | 10 | 2 | 1 | 1 |
| 11 | 9 | 9 | 3 | 1 | 1 |

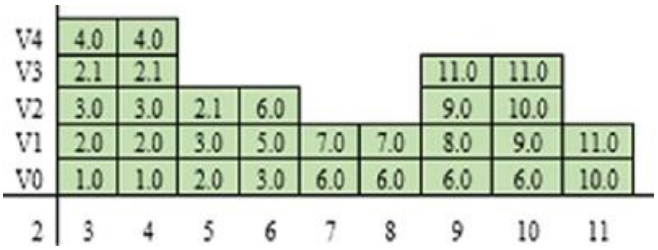


Fig 2. 11 Users have been allocated (Logical View) for parametric jobs.

Fig. 2 is the result of a proposed algorithm for a parametric work plan, where the x-axis shows a time slot, and the y-axis shows a virtual computing node (Logical View). Along the y-axis are displayed 5 virtual computing nodes, which shown as v0, v1, v2, v3, and v4. Eleven user reservations have allocated during timeslot 3 to 11. Consider userID = 3 from Table 1. The virtual computing node v2 used by userID=3 is timeslot 3 (tesr=3), timeslot 4, computing node v times timeslot 5 and computing node v times timeslot 6 because only 1 job (requires 4 timeslot time execution) that has been sent by the user. For example, userId = 12 wants to order 3 timeslots from 5 to timeslot 10, requires 3 compute nodes. Each job can be postponed until timeslot 10($t_{esr}$=5, $t_{lsr}$=10, $t_e$=3, jumCN=3, jumJob=3) (see Fig 3).

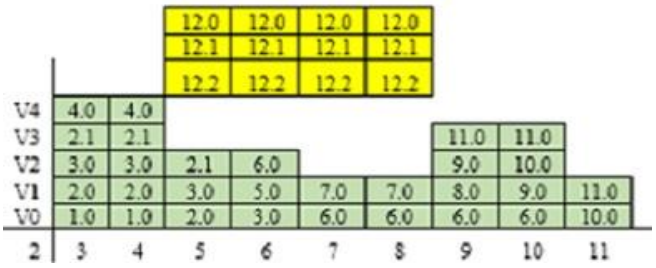


Fig 3. New users create job requests for parametric jobs

Fig.3 shows the results of a planned new incoming reservation request from user 12, for parametric work. Using conventional reservations or rigid reservations, only two jobs independent of user12 will allocate, and the other one will reject. Fig. 4 shows that the same job will assign to different time slots and virtual computing nodes. Successful reservation, a notification will be sent to the user only once (in the FCFS-LRH approach because it works in a logical view). If using another method, the information sent each time a revision is made in the plan[12][16].

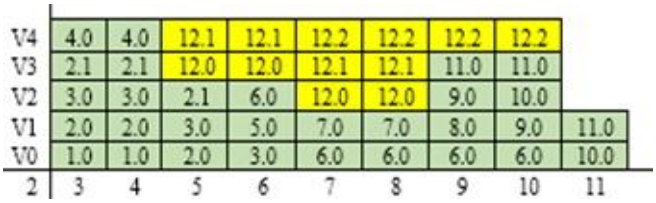


Fig 4. New users have been allocated using FCFS-LRH (Logical View) for parametric jobs.

### B. Planning Mapping to the Actual Computing Node

The FCFS-LRH algorithm is proposed to guarantee that logical display plans can always be mapped to the actual computing node, and once a job started at a specific logical node, it executed at the same actual node for all time slots. The work plan in virtual view as shown in Fig 5, will guarantee that all allocated jobs will be executed at the actual node, because the reservation system works on logical computing, as shown in Fig 6 (Physical View).

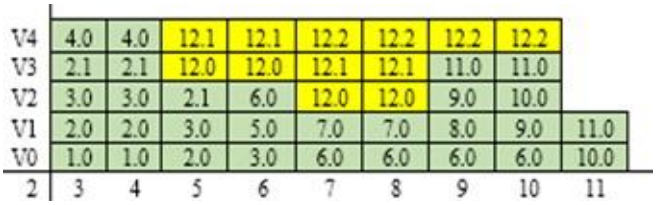


Fig 5. Allocation / parametric jobs (Logical View) planning for reservation requests using FCFS-LRH

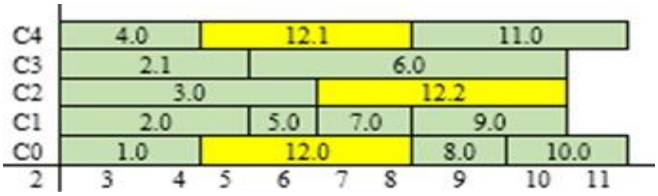


Fig 6. The mapping results on the actual node (Physical View) for parametric jobs

## IV. EXPERIMENTS AND RESULTS

Parametric job experiments have been carried out using proposed planning and scheduling strategies for FCFS-LRH reservations. The ratio of the utilization of the experimental results compared to the reservation strategy that does not

use planning. The workload or user request, for this experiment, has the following characteristics [10][14][17], are as follows:

- The level of incoming reservation demand is assumed to follow the Poisson distribution.
- Execution time($t_e$) for reservation requests distributed uniformly.
- The earliest start time ($t_{esr}$) for reservation requests distributed uniformly.
- Percentage of the flexible reservation is randomly selected.
- Time for flexibility ($t_f$) for reservation requests uniformly distributed.
- The amount of Timeslot is 5 minutes.

The total number of compute nodes used in the experiment (jumCN=30), the level of reservation requests used (μ=3 and μ=4), the FCFS-LRH utilization factor and without a reservation will be measured. Fig. 7 shows the comparison of the percentage of utilization factors between FCFS-LRH scheduling with without reservation for parametric work, with μ=3 and percent flexibility=100%. Fig. 7 shows the results that the number of jobs received in the same timeslot is higher because the user request job can shift from the earliest start time to the upper end of the start time limit. Fig. 8. shows the results of the comparison of the percentage of FCFS-LRH scheduling benefits without reservation with μ=4 and Percentage of flexibility=100%. From Fig. 8 it appears that although the number of jobs entered is higher because μ = 4, the results still indicate that the amount jobs received at the same timeslot are higher because the user's job request can shift from $t_{esr}$ to the upper limit of $t_{lsr}$. Both fig. 7 and 8 show that FCFS-LRH results in better utilization than traditional strategies (without prior planning). In fig. 9 and 10. The percentage of flexibility is measured from 25% to 100% with an average arrival factor (μ=3 and μ=4). The use of FCFS-LRH is better than without reservation. Fig. 11 shows the average increase in the usefulness of the FCFS-LRH algorithm by 2.95% compared to without reservation.

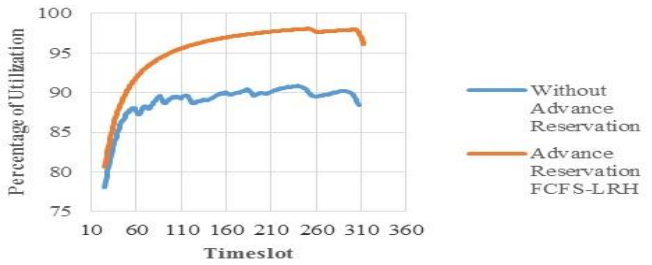


Fig 7. Comparison of reservation scheduling (FCFS-LRH) with without reservation, for Parametric job. The average arrival of μ=3 and Percentage flexibility=100%.


Fig 8. Comparison of reservation scheduling (FCFS-LRH) with without reservation, for Parametric work. The average arrival of μ=4 and Percentage flexibility=100%.

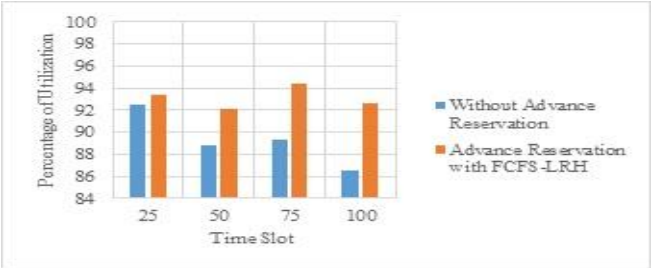


Fig 9. Percentage of utilization based on Percentage Flexibility with average arrival of (μ = 3).

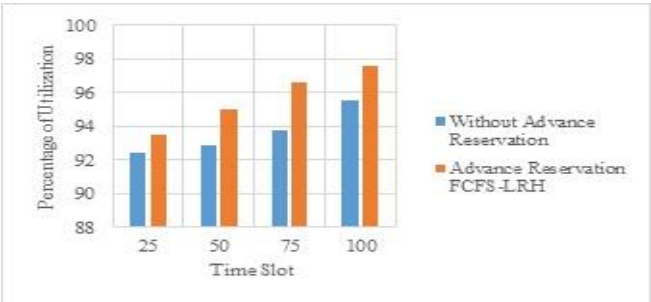


Fig 10. Percentage of utilization based on Percentage Flexibility with average arrival of (μ = 4).

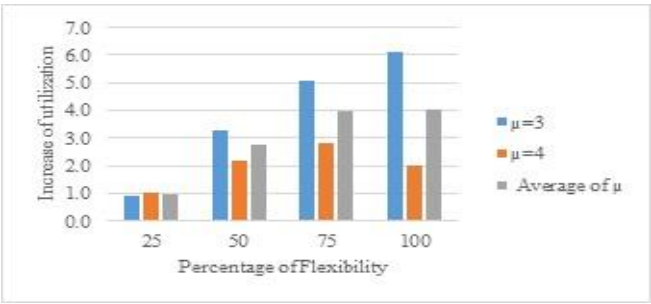


Fig 11. Increase Percentage of utilization based on Percentage of Flexibility and utilization factor

## V. CONCLUSION

In this study focused on the job scheduling model on grid computing, and the proposed scheduling algorithm on the local scheduler. From the results of the research, the proposed algorithm for parametric jobs can schedule jobs in the local Grid scheduler. In terms of utilization, the proposed algorithm has a better average percentage of utilization compared to conventional algorithms for parametric jobs.

## REFERENCES


[1] Amulya and P. S. Kulkarni, "Review of Grid Computing technology in electrical power systems," *Int. Conf. Electr. Power Energy Syst. ICEPES 2016*, pp. 487–491, 2017.

[2] I. Foster and C. Kesselman. editors. "*The Grid: Blueprint for a New Computing Infrastructure,"* San Fransisco, Morgan Kaufmann, 1999.

[3] A. W. Mu'alem and D. G. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 6, pp. 529–543, 2001.

[4] A. Sulistio and R. Buyya, "A grid simulation infrastructure supporting advance reservation," *Proc. IASTED Int. Conf. Parallel Distrib. Comput. Syst.*, vol. 16, pp. 1–7, 2004.

[5] W. Smith, I. Foster, and V. Taylor, "Scheduling with advanced reservations," in *14th IEEE International Symposium on Parallel and Distributed Processing*, pp. 127–132, 2002.

[6] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," *IEEE Int. Work. Qual. Serv. IWQoS*, no. 1, pp. 27–36, 1999.

[7] K. Czajkowski *et al.*, "A resource management architecture for metacomputing systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing. LNCS vol. 1459*, pp. 62–82, 2006.

[8] A. Sulistio, K. H. Kim, and R. Buyya, "On incorporating an on-line strip packing algorithm into elastic grid reservation-based systems," *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 1, 2007.

[9] C. Hu, J. Huai, and T. Wo, "Flexible resource reservation using slack time for service grid," *Proc. Int. Conf. Parallel Distrib. Systems-ICPADS*, vol. 1, pp. 327–334, 2006.

[10] L. Grandinetti, F. Guerriero, L. Di Puglia Pugliese, and M. Sheikhalishahi, "Heuristics for the local grid scheduling problem with processing time constraints," *J. Heuristics*, vol. 21, no. 4, pp. 523–547, 2015.

[11] C. Castillo, G. N. Rouskas, and K. Harfoush, "On the design of online scheduling algorithms for advance reservations and QoS in grids," *Proc. - 21st Int. Parallel Distrib. Process. Symp. IPDPS 2007; Abstr. CD-ROM*, 2007.

[12] M. A. S. Netto, K. Bubendorfer, and R. Buyya, "SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters,," in *5th International Conference on Service-Oriented Computing, LNCS vol. 4749*, pp. 119–131, 2007.

[13] C. Xu and J. W. Wong, "Scheduling algorithms for advance resource reservation," IFIP International Federation for Information Processing**,** pp. 659–671, 1998.

[14] R. Umar, A. Agarwal, and C. R. Rao, "Advance Planning and Reservation in a Grid System," *Commun. Comput. Inf. Sci.*, vol. 293 PART 1, pp. 161–173, 2012.

[15] A. Pujiyanta, L. E. Nugroho, and Widyawan, "Planning and Scheduling Jobs on Grid Computing," *Proceeding - 2018 Int. Symp. Adv. Intell. Informatics Revolutionize Intell. Informatics Spectr. Humanit. SAIN 2018*, pp. 162–166, 2019.

[16] B. Barzegar, A. M. Rahmani, K. Zamanifar, and A. Divsalar, "Gravitational emulation local search algorithm for advanced reservation and scheduling in grid computing systems," *ICCIT 2009 - 4th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, pp. 1240–1245, 2009.

[17] A. Sulistio, K. H. Kim, and R. Buyya, "Using revenue management to determine pricing of reservations," *Proc. - e-Science 2007, 3rd IEEE Int. Conf. e-Science Grid Comput.*, pp. 396–404, 2007.