HASIL CEK_60030480

by Ardi Pujiyanta 60030480

Resource Allocation Model for Grid Computing Environment

Ardi Pujiyanta[a,b,1,*], Lukito Edi Nugroho[a,2], Widyawan[a,3]

[a] Department of Electrical Engineering and Information Technology, Faculty of Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia.
[b] Department of Informatics Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia.

[1] ardi.pujiyanta@mail.ugm.ac.id*; [2] lukito@ugm.ac.id; [3] widyawan@ugm.ac.id

* corresponding author

ARTICLE INFO

Article history

Received

Revised

Accepted

Keywords
FCFS-LRH
Grid computing
Resource allocation
Resource utilization
Waiting time

ABSTRACT

Grid computing is a collection of heterogeneous resources that is very dynamic and unpredictable. Usually used for solving scientific or technical problems that require a large number of computer processing cycles or access to large amounts of data. Various resource allocation strategies have used to make resource use more productive. So that the distributed environmental performance increases. The user sends the job by providing a predetermined time limit for running the job. Then the scheduler gives priority to work according to the request and scheduling policy and places it in the waiting queue. When the resource is released, the scheduler selects the job from the waiting queue with a specific algorithm. Requests will reject if the required resources are not available. The user can re-submit a new request by modifying the parameter until available resources can found. In the end, it causes the idle resources between work and resource utilization to decrease, and the waiting time will increase. An effective scheduling policy is needed to improve resource use and reduce waiting times. In this paper, the FCFS-LRH method proposed, where jobs received, will be sorted by arrival time, execution time, and the number of resources needed. After the sorting process, the work will place in a logical view. The job will be sent to the actual resource when it will execute. The experimental results show that the proposed model can increase resource utilization and reduce waiting time when compared to existing approaches.

1. Introduction

Grid computing is the application of resources in the network to solve one problem at the same time. We are usually used to solve scientific or technical issues, such as high energy physics, earth observation, and biological applications, which require many cycles of computer processing or access to large amounts of data. Grid computing can be considered as a large-scale distributed cluster computing and as a form of distributed parallel network processing [1]. The two most important issues in managing user work are resource allocation and scheduling of work based on the support needed. When a user's job is submitted, the situation will be handled by a resource broker, who must find and allocate the right resources for the job. After the resource allocation phase, jobs must be scheduled on existing resources and according to user needs.

In general, when a user sends a job, he requests several processors, memory and provides the maximum time limit needed to run the job. Then the scheduler gives priority to work, according to the scheduling requests and policies, and places it in the waiting queue. When the resource is released, the scheduler selects the job from the waiting queue with a specific algorithm. In rigid scheduling like FCFS, three parameters used to request resources, namely start time, execution time, and the number of resources used [2]. The scheduler will look for the availability of resources requested by users within the specified time interval. Requests will reject if the required resources are not available[3]-[5].

Inelastic reservations the user request parameter is used as a soft constraint. The reservation system instead rejects the request but provides an alternative that can be chosen by the user. This approach gives users the flexibility to select the best choice according to the needs of quality of Service (QoS) [6]-[8]. Relax Advance Reservation uses overlapping timeslots due to a tendency that the application exaggerates the reservation deadline to ensure its completion. User jobs scheduled even if booking violations occur because of overlapping jobs[9]-[13]. Anju Shukla et al. [14] proposed an algorithm with the primary objective being that if there is more than one resource chosen by a job, then the job that has the least workload will be executed first to reduce the average waiting time of the job queue. The algorithm will check the availability of resources that have the least load. In this rigid scheduling, work must wait in a queue to scheduled.

In a flexible reservation, the work user has a flexible start time and can vary within a certain time interval[15][16]. Moaddeli et al. [17] have examined the impact of the backfilling algorithm on flexible job ordering. In his research, aggressive backfilling and conservative backfilling compared. The result is aggressive backfilling has better benefits compared to conservative backfilling. Eliza et al. [18] propose checking free slots on available resources. If free slots are available, reservations scheduled. If free slots are not available during reservation requests, then the next available free slots will be reserved. The impact of the backfilling algorithm in flexible reservations has done by[19]-[22] research. Backfilling proposed to increase the utilization of the grid system. The advantage of this strategy is that it makes shift reservations early to make room for new reservations to allocated. The backfilling deficiency is that the next job must wait in line until the situation before it finished executed, so there is no certainty when the job will complete.

Netto et al. [14] conducted a study by rescheduling the allocated work. The experimental results show that the utilization of the system will be better if the user waits for up to 75% of the waiting time. Behnam et al. [15] introducing a reservation scheduling algorithm called GELSAR in the grid system. GELSAR will reschedule all new arrivals to find the best solution. New reservations will reject if there is no solution. The results of GELSAR outperform other Genetic Algorithms. Grandinetti et al.[25] have researched local scheduler, where a group of independent jobs has been scheduled, with processing time restrictions provided by the user. It assumed that all processing nodes are identical. Rusydi et al.[26], in their research, introduced First Come First Serve Ejecting Dynamic Based Scheduling (FCFS-EDS). The results of the FCFS-EDS experiments compared to FCFS without reservations are better in terms of resource utilization and also compared with studies[17],[23][24], having the same results. The advantage of FCFS-EDS is a one-time notification if a reservation is received because FCFS-EDS works in a logical view, In another approach, information has done whenever there is a revision made in the planning [23],[24]. In the FCFS-EDS strategy, incoming reservations will find an empty timeslot. If no timeslot found, the job will move to the upper limit of the execution, or if the old job has used the timeslot, then the old job will be shifted so that the new job can allocate. The impact of shifting the job to the right is a reduction in resource utilization and increased waiting time.

In this paper, the main focus is to overcome the problem of resource allocation on local scheduling in grid computing. The performance matrix used is resource utilization and job waiting time. The main contribution of this paper:

(1). Increasing resource utilization in grid computing scheduling.

(2). Reducing work waiting times with a comparison workload sent by users.

The next arrangement is, in section 2, contains the methods and algorithms proposed to solve the problem of resource allocation and waiting time. Section 3 describes the results and analysts comparing FCFS-EDS with FCFS-LRH. The last part includes conclusions.

2. Method

2.1. Proposed Advanced Reservation Strategy

In this study, a proposed reservation strategy model called First Come First Serving Left-Right Hole Scheduling (FCFS-LRH), is shown in Fig 1. Job requests are sent based on (JumCN, t_{esr}, t_{lsr}, t_e). Incoming user requests will be sorted by priority of start time of execution, time of execution, and the number of resources needed. Jobs will sort at each timeslot before being allocated to the virtual view. Work scheduled at the virtual computing node, with a first fit strategy. If an empty timeslot obtained between the start time (t_{esr}) and the upper limit to start the job execution (last start time) t_{lsr}, the user notified that the work has received. If there is no empty slot between t_{esr} and t_{lsr}, the job rejected. The user notified if the job rejected. The parameter t0 is the current time. The parameter t_n is the time of initial flexibility of work. The parameters t_{r1} and t_{r2} indicate that there are empty timeslots on the left and right. t_{el} shows the lower time limit for the last execution of the job. Work executed until the deadline (t_{esl}). The function of the t_f is to provide time flexibility on the job. Jobs that placed in a logical view are still fragmented. Recombination done when a job executed at the actual computing node[27][28].

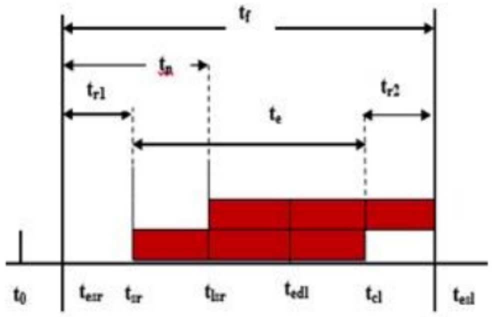


Fig. 1. Flexible reservation scheduling

2.2. Performance matrix of the FCFS-LRH method

The metrics considered for measuring the FCFS-LRH algorithm are Resource Utilization and Waiting Time. Average resource utilization calculated using formula 1. Formula(1) refers to the comparison of the number of Rj resources executed against the total amount of available resources. In the formula, s refers to the number of slots used by the resource. ts_i refers to the start time when jobs executed on R_j resources, and te_i refers to the final time that a task completed on R_j resources. The T parameter defined as the execution time of all jobs.

$$RU_j = \frac{\sum_{i=1}^{s}(te_i - ts_i)}{T} \tag{1}$$

Waiting time(WT) of the reservation is calculated. Sometimes resources are not available at the time of reservation requires. But resources can be used at different times. In this case, the difference between the expected start time(startR) and the actual start time (startN) is the waiting time.

$$WT = StartR - StartN \tag{2}$$

Total Wait Time(TWT) calculated as the sum of all waiting times at a particular timeslot.

$$TWT = \sum_{i=0}^{size} WT \tag{3}$$

Where size refers to the length of the reservation at a certain point in time. Then the average waiting Time (AWT) is

$$AWT = \frac{TWT}{No\ of\ reservation} \tag{4}$$

3. Proposed FCFS-LRH algorithm

The user needs to send a reservation, with parameters qReserv(t_{esr}, t_{lsr}, t_e,numC), to order resources(step 1). New reservation requests explained in the FCFS-LRH algorithm below, sort jobs based on arrival time (t_{esr}), execution time (t_e), and the number of resources needed (step1-step3).

Step4 is a loop to read the jobs that come in each timeslot and place them in a virtual view if there are empty timeslots. Step5 to step7 variable initialization, step8 is the process of finding an empty slot with a first fit strategy starting from CN=0 to CN, and timeslot starting from t_{esr} to the upper limit (d2). Insert job (Step9) if an empty slot found. Step10, if no empty slots found work on the moveJob() procedure, step11 end of the loop.

The moveJob() procedure, is used to move jobs because an empty timeslot is not found, until the upper limit of the start time of execution (t_{lsr}). step2 and step3 variable initialization. Step4 to step22 is a search loop that starts from the left to the timeslot, with relax=0 to t_r(flexibility value (t_{lsr} - t_{esr})). If an empty timeslot is found between the lower limit (t_{esr}) to the value of the finished variable, indicated by step23 to step28 (suc == true), insert jobId in the timeslot with the first fit strategy and save the number of jobsID received in the sList dynamic array and calculate waiting time. Step29 to step36 is used to calculate time slot utilization, where sList is a dynamic array, to hold the number of job IDs used at certain timeslot.

Algorithm: Resource Allocation Algorithm

Input: Job (jobId,tesr,tlsr,te,numC)
Output: RU, AWT

```
1. For(j=0 to j< jumJob) do
2.   sort jobs by; tesr[j] >= tesr[j+1],te[j] >= te[j+1], numC[j] > numC[j+1]
3. End for
4. For i = 0 to i < jumJob do // jumjob is the number of jobs per timeslot
5.   cek←0,Jobid←jobId[i],tesr←tesr[i],tlsr←tlsr[i];
6.    te ← te[i],CN←numC[i];
7.    d2←tesr+te-1;
8. // Search timeslot free with First fit strategy.
9.   If (timeslot==empty); insert Jobid
10.    If (timeslot!=empty); movejob()
11. Endfor.
```

```
1. Procedure moveJob();
2. Initialization; finish←0,suc←false, start←tesr, finish←tesr+te-1.
3.  relax←start–tesr, tr←tlsr–tesr,CNs←0.
4. while (!suc and relax <=tr) do
5    For(cek=start to cek<finish)
6          CNs←0;
7        For s=0 to s<atrans.size() then
8          If atrans.get(s,cek)!=0
9            CNs++;
10         Endif
11       Endfor
12         sel←maxC-CNs; // maxC is the number of physical nodes
13       If sel>=CN then
14         t←start, suc←true;
15         Else
16           t←cek
17           finish←start+te-1
18           relax←start-tesr, suc←true;
19             If (start>=tlsr) break;
20       Endif
21   Endfor
22  Endwhile
23 If (suc==true) then
24         start←t+1, finish←start+te-1;
25         relax=start-tesr;
```

```
26     // insert JobID with the first fit strategy.
27    // calculate waiting time(AWT)
28 Endif
29 For y=0 to y<sList.size()  do
30    If (sList.get(y)!=0) then
31          p++;
32     Is←sList.get(y);
33          x1←(Is/node); utilc←utilc+x1;
34          RU←utilc/p;
35        Endif
36 Endfor
```

3.1. Illustration of FCFS-LRH

An example will give to explain FCFS-LRH. If it knows that the actual compute node maxC=6 (R0-R5) as a physical node, then the number of virtual nodes will have the same number of 6 (V0-V5). The order of arrival of the reservation illustrated in table 1, where numC≤maxC and numJob are the numbers of jobs sent by the user. For example, the parameters are given by userID=6 as in table 1. are as follows: userID6 orders 3-time slots starting from timeslot 1 to 6, it takes 2 computing nodes for 1 independent work and can be shifted (t_{esr}=1, t_{lsr}=6, t_e=3, numC=2, numJob=1).

Table 1. Reservation from the user

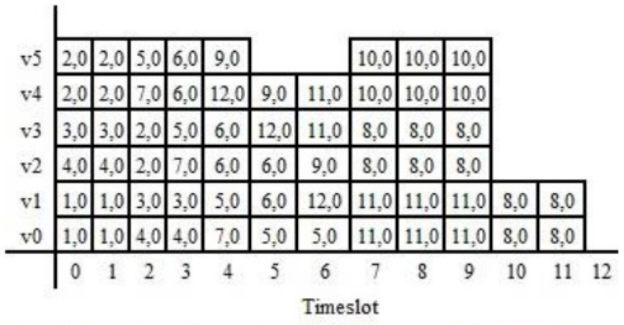
userID	tesr	tlsr	te	numC	numJob
1	0	0	2	2	1
2	0	3	3	2	1
3	0	1	4	1	1
4	0	0	4	1	1
5	1	5	5	1	1
6	1	6	3	2	1
7	1	6	3	1	1
8	2	9	5	2	1
9	2	9	3	1	1
10	3	8	3	2	1
11	2	8	4	2	1
12	2	8	3	1	1

Table 2. shows the userID jobs that have been sorted by t_{esr}, t_e, and numC. Fig 2 shows the logical view of the results of FCFS-LRH from Table 2, the x-axis shows the time slot, and the y-axis shows the virtual computing node. There are six virtual computing nodes, which displayed on the y-axis, 11 user reservations have allocated from timeslot 0 to 11. Consider userID6 from table 2 and Fig 2. The virtual node given to userID6 is in timeslot 3 with compute nodes v4, v5, timeslot 4 with compute nodes v2, v3, timeslot 5 with compute nodes v1, v2. One job that will be done by two computing nodes at a time slot, which has been sent by the user. For example, userID12 wants to order three timeslots starting from timeslot 2 to 8, requires one computational node for one independent job and can shift from start time to timeslot 8(t_{esr}=2, t_{lsr}=8, te=2, jumCN=2, jumJob=1), in Fig 2, userID12 starts the execution time from timeslot 4 to timeslot 6.

Table 2. Results of the arrival time of the userId

userID	tesr	tlsr	te	numC	numJob
1	0	0	2	2	1
4	0	0	4	1	1
3	0	1	4	1	1
2	0	3	3	2	1
7	1	6	3	1	1
5	1	5	5	1	1
6	1	6	3	2	1
12	2	8	3	1	1
9	2	9	3	1	1
11	2	8	8	2	1
8	2	9	5	2	1

10 3 8 3 2 1



v5	2,0	2,0	5,0	6,0	9,0			10,0	10,0	10,0			
v4	2,0	2,0	7,0	6,0	12,0	9,0	11,0	10,0	10,0	10,0			
v3	3,0	3,0	2,0	5,0	6,0	12,0	11,0	8,0	8,0	8,0			
v2	4,0	4,0	2,0	7,0	6,0	6,0	9,0	8,0	8,0	8,0			
v1	1,0	1,0	3,0	3,0	5,0	6,0	12,0	11,0	11,0	11,0	8,0	8,0	
v0	1,0	1,0	4,0	4,0	7,0	5,0	5,0	11,0	11,0	11,0	8,0	8,0	
	0	1	2	3	4	5	6	7	8	9	10	11	12

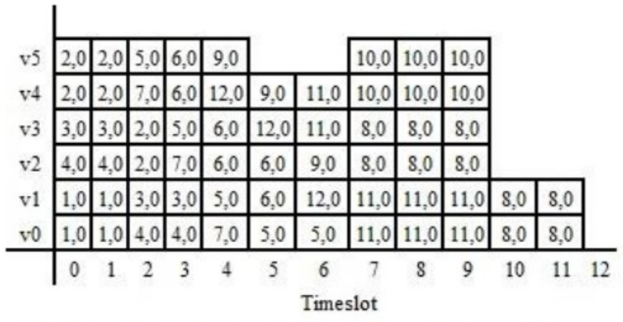
Timeslot

Fig 2. Job placement in a logical view with FCFS-LRH method.

In Fig 2, userID12 will reject if a reservation made using a rigid reservation. t_{esr} parameters in rigid reservations cannot shift. The userID12 job is placed on timeslot 4 through timeslot 6, on different virtual computing nodes. Notifications will be sent to users only once if the reservation is successful (FCFS-LRH works in the virtual display)[27].

3.2. Mapping from virtual nodes to actual computing nodes

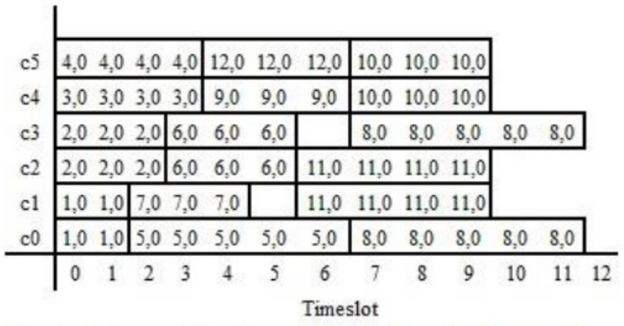
Fig 3 shows the job placement (logical view) from the results of table 2. In contrast, Fig 4 is the result of the recombination of logical views, which mapped to the physical view for all userID in table 2, which will guarantee that all jobs can execute on the actual node.



v5	2,0	2,0	5,0	6,0	9,0			10,0	10,0	10,0			
v4	2,0	2,0	7,0	6,0	12,0	9,0	11,0	10,0	10,0	10,0			
v3	3,0	3,0	2,0	5,0	6,0	12,0	11,0	8,0	8,0	8,0			
v2	4,0	4,0	2,0	7,0	6,0	6,0	9,0	8,0	8,0	8,0			
v1	1,0	1,0	3,0	3,0	5,0	6,0	12,0	11,0	11,0	11,0	8,0	8,0	
v0	1,0	1,0	4,0	4,0	7,0	5,0	5,0	11,0	11,0	11,0	8,0	8,0	
	0	1	2	3	4	5	6	7	8	9	10	11	12

Timeslot

Fig. 3. Job placement in a logical view with FCFS-LRH method.



c5	4,0	4,0	4,0	4,0	12,0	12,0	12,0	10,0	10,0	10,0			
c4	3,0	3,0	3,0	3,0	9,0	9,0	9,0	10,0	10,0	10,0			
c3	2,0	2,0	2,0	6,0	6,0	6,0		8,0	8,0	8,0	8,0	8,0	
c2	2,0	2,0	2,0	6,0	6,0	6,0	11,0	11,0	11,0	11,0			
c1	1,0	1,0	7,0	7,0	7,0		11,0	11,0	11,0	11,0			
c0	1,0	1,0	5,0	5,0	5,0	5,0	5,0	8,0	8,0	8,0	8,0	8,0	
	0	1	2	3	4	5	6	7	8	9	10	11	12

Timeslot

Fig 4. Job placement in a physical view with FCFS-LRH method.

3.3. Job workload

Each different grid model requires a different workload composition[29]. The characteristics of the workload to be used as simulation input in this experiment are as follows[26][30]-[32].

- The level of reservation requests (μ) is assumed to follow the Poisson distribution.
- Reservation requests distributed uniformly, which defined as the execution time (t_e).

- Reservation requests are between 0 and 24, uniformly distributed, which set as the earliest start time(t_{esr}).
- Percentage of flexible user reservations randomly selected.
- The flexibility time(t_f) for reservation requests is between 1 and 12 that uniformly distributed.

The flexibility time(t_f) for reservation requests is between 1 and 12 that evenly distributed. The percentage of resource utilization calculated in a sliding window of 12 timeslots (1 hour) and the results of the proposed FCFS-LRH method compared with FCFS-EDS. The utilization factor of the two strategies measured using the input characteristics above with the total number of computational nodes being 20, with reservation demand levels μ=2, μ=3, the number of jobs used between 300 and 800.

4. Results and Discussion

For the simulation of the proposed resource allocation model, the software used is Eclipse IDE for Java Developers, Windows 8 operating system, Intel (R) Pentium (R) CPU B940 @ 2.00 GHz. 6.00 GB RAM configuration. The FCFS-EDS approach is used as a comparison because the FCFS-EDS approach has the advantage of working in a logical view environment, and notification to users only done once when jobs can allocate to logical views. On the other hand, FCFS-EDS has the disadvantage that tasks received or assigned to logical views not based on the priority of the start time of execution, the time of performance, and the number of resources needed. So it is possible that the use of resources is not efficient and the length of time the job is waiting. The resource utilization matrix and a job waiting time are used as a comparison of performance so that the use of resources becomes more efficient.

In this paper, a work allocation model for resources proposed to increase resource utilization and reduce the average waiting time. The order of job placement based on the time of initial execution, the time of the smallest work execution, and the number of lowest resource requirements prioritize so that the utilization of resources will increase, and the waiting time of the work can reduce. The parameters used in the experiment and for testing the performance shown in table 3.

Table 3. Parameters used by the experiment

Parameter	Range
Number of Jobs	300-800
Number of Resources	20
Rate reservation request	2-3
Percentage of flexibility	25-100

4.1. Comparison of FCFS-EDS and FCFS-LRH Methods

FCFS-EDS and FCFS-LRH methods included in the Flexible Advance Reservation Dynamic scheduling, sample table 1 above, will use to make it easier to compare resource utilization and an average waiting time for jobs. The results of job placement from table 1 in logical view using FCFS-EDS, shown in Fig 5, where the x-axis shows timeslot, the y-axis shows virtual nodes and the results of job placement in the physical view shown in Fig 6. Table 4. It indicates that the average waiting time FCFS-EDS is 0.83, and FCFS-LRH is 0.72.

Job waiting Time for FCFS-LRH is smaller than FCFS-EDS; this is because the start time of the situation can advance at an earlier timeslot, so the difference in waiting time (start- t_{esr}) is smaller. There are 3 FCFS-LRH userID whose advance start time is advanced, namely userID5 advanced 1 timeslot earlier, userID9 earlier 1 timeslot, and userID11 earlier 2 timeslots, so the total timeslot that can use earlier is 4 timeslot. Whereas FCFS-EDS only 1 userID can advance its execution time more first, namely userID10 of 1 timeslot. So the difference between timeslot FCFS-LRH with FCFS-EDS that can be improved earlier when the execution used is 3 timeslot (see Table 4), where the total waiting time (numWT) of FCFS-LRH is 21, and for FCFS-EDS is 24.

The impact of the start time that can place in an earlier slot, then the job waiting time can reduce. The use of timeslot FCFS-LRH is higher than FCFS-EDS; this is due to the use of timeslot earlier in its execution time (see Fig 4 and compare with Fig 6). Where userID5 allocated to timeslot 2, userID9 allocated to timeslot 4, userID11 is allocated to timeslot 6 using the FCFS-LRH method.

The utilization of FCFS-LRH timeslot is higher than FCFS-EDS starting from timeslot 2 in resource (see table 5), at timeslot 2 the level of FCFS-LRH usage is 100%. In comparison, FCFS-EDS timeslot utilization is 98%, until the end of the timeslot used for the execution time used by userID.

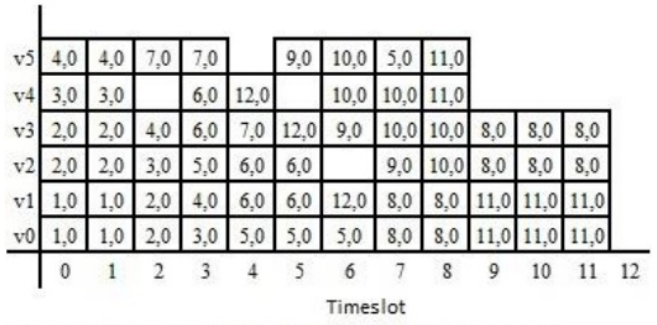

v5 4,0 4,0 7,0 7,0 9,0 10,0 5,0 11,0
v4 3,0 3,0 6,0 12,0 10,0 10,0 11,0
v3 2,0 2,0 4,0 6,0 7,0 12,0 9,0 10,0 10,0 8,0 8,0 8,0
v2 2,0 2,0 3,0 5,0 6,0 6,0 9,0 10,0 8,0 8,0 8,0
v1 1,0 1,0 2,0 4,0 6,0 6,0 12,0 8,0 8,0 11,0 11,0 11,0
v0 1,0 1,0 2,0 3,0 5,0 5,0 5,0 8,0 8,0 11,0 11,0 11,0
0 1 2 3 4 5 6 7 8 9 10 11 12
Timeslot

Fig. 5. Job placement in a logical view with FCFS-EDS method.

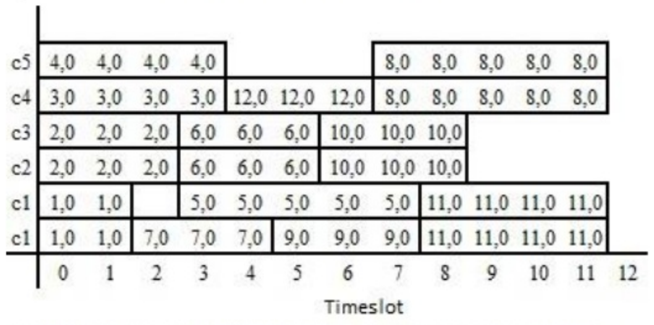

c5 4,0 4,0 4,0 4,0 8,0 8,0 8,0 8,0 8,0
c4 3,0 3,0 3,0 3,0 12,0 12,0 12,0 8,0 8,0 8,0 8,0 8,0
c3 2,0 2,0 2,0 6,0 6,0 6,0 10,0 10,0 10,0
c2 2,0 2,0 2,0 6,0 6,0 6,0 10,0 10,0 10,0
c1 1,0 1,0 5,0 5,0 5,0 5,0 5,0 11,0 11,0 11,0 11,0
c1 1,0 1,0 7,0 7,0 7,0 9,0 9,0 9,0 11,0 11,0 11,0 11,0
0 1 2 3 4 5 6 7 8 9 10 11 12
Timeslot

Fig 6. Job placement in a physical view with the FCFS-EDS method

Table 4. Comparison of FCFS-LRH and FCFS-EDS waiting times

FCFS-EDS								FCFS-LRH							
user ID	start	t_{esr}	t_e	Num te	wt	num WT	AWT	user ID	start	t_{esr}	t_e	Num te	wt	Num WT	AWT
7	2	1	3	3	1	1	0.33	5	2	1	5	5	1	1	0.20
5	3	1	5	8	2	3	0.38	7	2	1	3	8	1	2	0.25
6	3	1	3	11	2	5	0.45	6	3	1	3	11	2	4	0.36
12	4	2	3	14	2	7	0.50	9	4	2	3	14	2	6	0.43
9	5	2	3	17	3	10	0.59	12	4	2	3	17	2	8	0.47
10	6	3	3	20	3	13	0.65	11	6	2	4	21	4	12	0.57
8	7	2	5	25	5	18	0.72	8	7	2	5	26	5	17	0.65
11	8	2	4	29	6	24	0.83	10	7	3	3	29	4	21	0.72

Table 5. Resource utilization

	Timeslot												
Method	0	1	2	3	4	5	6	7	8	9	10	11	12
FCFS-LRH	100	100	100	99.5	98.9	98.38	98.1	98	97.2	96.3	94.9		
FCFS-EDS	100	100	98	97.25	96.4	95.5	94.7	94	94	93.6	93	92.4	91.38

Two experiments have carried out using the parameters in table 3. First, compare the FCFS-LRH waiting time with FCFS-EDS, (Backfilling, Aggressive backfilling, without reservation). The second is comparing the resource use between the FCFS-EDS and FCFS-LRH methods.

The first experiment was to measure FCFS-EDS, FCFS-LRH waiting times (backfilling and aggressive backfilling without reservation), whose results shown in table 6. Table 6 shows job waiting times for FCFS-LRH smaller than FCFS-EDS, Backfilling, and Aggressive backfilling without reservation. This is due to the time flexibility affects the actual start time, which can be close to the expected time when the user submits a job. The waiting time (backfilling and aggressive backfilling without reservation) is high because the next task must wait in a queue until the job before it finished executing. Thus the FCFS-LRH provides a better allocation policy because time flexibility(t_f) used to reduce reservation waiting time. The impact of a good allocation policy is that the waiting time value of FCFS-LRH be reduced compared to FCFS-EDS (Backfilling and Aggressive backfilling without reservation) for all conditions. The waiting time value based on the level of reservation arrivals and the number of jobs submitted by users (table 6 and Fig 7). For the μ=2 and μ=3, the average reduction in waiting time is 20.47%. While the average decrease of waiting time with a level of reservation requests (μ=2) was 36.8%, for the level of reservation requests (μ=3) waiting time reduction was 9.7% (table 7 and Fig 8).

The results of the second experiment shown in table 8 and Fig 9. From table 8, it can calculate, the average value of timeslot utilization has increased by 1.34%, while the average resource utilization is 1.17% for μ= 2 and 1.5% for μ=3 (table 9 and Fig 10). This increase in utilization was due to previous work placement starting from the left of the timeslot so that the use of timeslot increased. Jobs will be allocated first on the left-hand side.

Table 6. Comparison of FCFS-LRH and FCFS-EDS waiting times

	Number of jobs					
	μ=2			μ=3		
Method	383	402	421	601	618	673
FCFS-LRH	0,55	0,78	0,49	1,05	0,94	0,78
FCFS-EDS	0,78	1,03	0,69	1,09	1,07	0,88
Aggressive backfilling	2,97	1,11	1,84	1,77	4,22	3,33
Backfilling	3,25	1,42	2,45	2,27	4,91	3,6

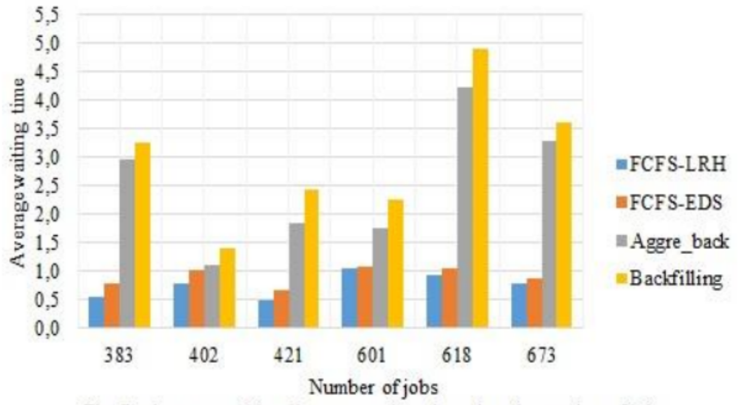


Fig. 7. Average waiting time comparison based on the number of jobs

Table 7. Average waiting time based on the average level of reservation request

	Rate reservation request	
Method	μ=2	μ=3
FCFS-LRH	0.61	0.92
FCFS-EDS	0.83	1.01
Aggressive backfilling	1.97	3.10
Backfilling	2.37	3.59

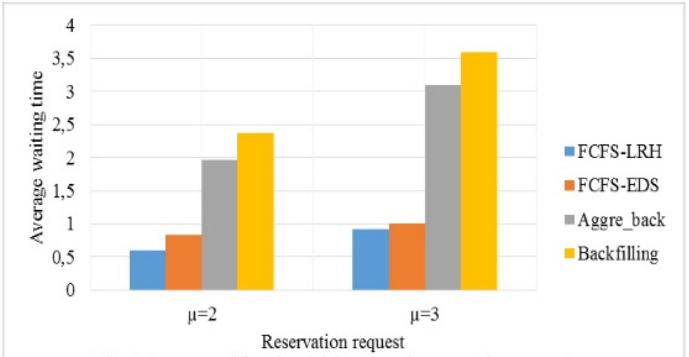


Fig. 8. Average waiting time job based on the reservation request.

Table 8 Percentage of utilization results

	Number of jobs					
	μ=2			μ=3		
Method	383	402	421	601	618	673
FCFS-EDS	89,96	94,94	92,60	93,40	93,92	93,83
FCFS-LRH	90,91	96,60	93,51	94,68	95,60	95,68

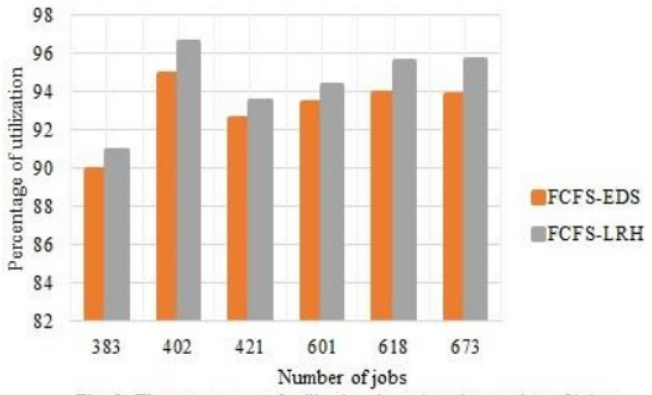


Fig. 9. The percentage of utilization based on the number of jobs

Table 9. The average level of reservation requests

	Rate reservation request	
Method	μ=2	μ=3
FCFS-EDS	92,50	93,72
FCFS-LRH	93,67	95,22

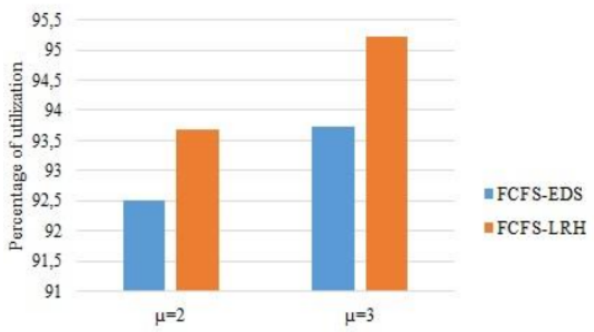


Fig. 10. Percentage of utilization based on the average level of reservation requests

5. Conclusion

Various resource allocation strategies have used to make resource use more productive. So that the distributed environmental performance increases. An effective scheduling policy is needed to increase resource use and reduce waiting times. In this paper, a reservation scheduling strategy called FCFS-LRH proposed. Jobs that come in this strategy are sorted by priority first, and then jobs will be placed on virtual nodes. Jobs allocated to virtual nodes will be mapped to physical nodes when they executed. Work that has allocated on a virtual node will be guaranteed to executed on physical resources. Experimentally the FCFS-LRH method is compared with FCFS-EDS, backfilling, and aggressive backfilling without reservation. FCFS-LRH performance increases better in terms of resource utilization and can reduce job waiting times. The results of this study can only used in local scheduling in grid computing.

REFERENCES

[1] M. Caramia, S. Giordani, and A. Iovanella, "Grid scheduling by on-line rectangle packing," *Networks*, vol. 44, no. 2, pp. 106–119, 2004.

[2] W. Smith, I. Foster, and V. Taylor, "Scheduling with advanced reservations," pp. 127–132, 2002.

[3] 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.

[4] I. Foster and C. Kesselman, editors, *The Grid 2 - Blueprint for a New Computing Infrastructure*, Los Altos, California, Morgan Kaufmann,2004.

[5] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S.Tuecke, "A resource management architecture for meta computing systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing. LNCS vol. 1459. Springer*, London,1998.

[6] Buyya, R., Murshed, M.: GridSim: A Toolkit for the Modeling and Simulation of Distri- buted Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience 14, 1175–1220,2002.

[7] 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.

[8] J. Shi, J. Luo, F. Dong, J. Zhang, and J. Zhang, "Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints," *Cluster Comput.*, vol. 19, no. 1, pp. 167–182, 2016.

[9] 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.

[10] 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.

[11] P. Xiao, Z. Hu, X. Li, and L. Yang, "A novel statistic-based relaxed grid resource reservation strategy," *Proc. 9th Int. Conf. Young Comput. Sci. ICYCS 2008*, no. 2, pp. 703–707, 2008.

[12] Sabitha, R.B.S., Venkatesan, R., Ramalakshmi, R.: Resource Reservation In Grid Compu- ting Environments: Design Issues. In: 3rd International Conference on Electronics Com- puter Technology, pp. 66–70. IEEE Press, Kanyakumari,2011.

[13] Peng, X., Zhigang, H.U.: Relaxed resource advance reservation policy in grid computing. The Journal of China Universities of Posts and Telecommunications 16, 108–113, 2009.

[14] A. Shukla, S. Kumar, and H. Singh, "An improved resource allocation model for a grid computing environment," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 104–113, 2019.

[15] M. Barshan, H. Moens, B. Volckaert, and F. De Turck, "A comparative analysis of flexible and fixed size timeslots for advance bandwidth reservations in media production networks," in *2016 7th International Conference on the Network of the Future (NOF)*, 2016, pp. 1–6.

[16] M. Barshan, H. Moens, J. Famaey, and F. De Turck, "Deadline-aware advance reservation scheduling algorithms for media production networks," *Comput. Commun.*, vol. 77, no. 2015, pp. 26–40, 2016.

[17] Moaddeli, H.R., Dastghaibyfard, G., Moosavi, M.R.: Flexible Advance Reservation Impact on Backfilling Scheduling Strategies. In: 7th International Conference on Grid and Cooperative Computing, pp. 151–159. IEEE Press, Shenzhen,2008.

[18] E. Gomes and M. A. R. Dantas, "Towards a resource reservation approach for an opportunistic computing environment," *J. Phys. Conf. Ser.*, vol. 540, no. 1, 2014.

[19] A. Mishra, "An enhanced and effective preemption based scheduling for grid computing enabling backfilling technique," *Conf. Proceeding - 2015 Int. Conf. Adv. Comput. Eng. Appl. ICACEA 2015*, pp. 1015–1018, 2015.

[20] O. Dakkak, S. Awang Nor, and S. Arif, "Scheduling through backfilling technique for HPC applications in a grid computing environment," *ICOS 2016 - 2016 IEEE Conf. Open Syst.*, pp. 30–35, 2017.

[21] S. Leonenkov and S. Zhumatiy, "Introducing New Backfill-based Scheduler for SLURM Resource Manager," *Procedia Comput. Sci.*, vol. 66, pp. 661–669, 2015.

[22] R. Istrate, A. Poenaru, and F. Pop, "Advance reservation system for data centers," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2016-May, pp. 637–644, 2016.

[23] Netto, M.A.S., Bubendorfer, K., Buyya, R.: SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 119–131. Springer, Heidelberg, 2007.

[24] Behnam, B., Amir, M.R., Kamran, Z.F., Azedah, D.: 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, pp. 1240–1245. IEEE Press, Seoul, 2009.

[25] 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.

[26] U. Rusydi, et al., "Advance Planning and Reservation in a Grid System," in *NDT 2012 CCIS/LNCS, vol. 293. Springer, Heidelberg* , Dubai, 2012.

1

[27] 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. Humanity. SAIN, 2018*, pp. 162–166, 2019.

[28] A. Pujiyanta, L. E. Nugroho, and Widyawan, "Advance Reservation for Parametric Job on Grid Computing," *Proc. 2019 4th Int. Conf. Informatics Comput. ICIC 2019*, 2019.

[29] R. V. Lopes and D. Menascé, "A Taxonomy of Job Scheduling on Distributed Computing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3412–3428, 2016.

[30] M. Carvalho and F. Brasileiro, "A user-based model of grid computing workloads," *Proc. - IEEE/ACM Int. Work. Grid Comput.*, pp. 40–48, 2012.

7

[31] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: Modelling the characteristics of rigid jobs," *J. Parallel Distrib. Comput.*, vol. 63, no. 11, pp. 1105–1122, 2003.

[32] A. Iosup, D. H. J. Epema, J. Maassen, and R. Van Nieuwpoort, "Synthetic grid workloads with Ibis, KOALA, and GRENCHMARK," *Integr. Res. GRID Comput. - CoreGRID Integr. Work. 2005, Sel. Pap.*, pp. 271–283, 2007.

HASIL CEK_60030480

ORIGINALITY REPORT

13%	6%	12%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ardi Pujiyanta, Lukito Edi Nugroho, Widyawan. "Advance Reservation for Parametric Job on Grid Computing", 2019 Fourth International Conference on Informatics and Computing (ICIC), 2019 Publication	4%
2	"Networked Digital Technologies", Springer Science and Business Media LLC, 2012 Publication	3%
3	Massimiliano Caramia. "Grid scheduling by on-line rectangle packing", Networks, 09/2004 Publication	1%
4	Jean-Marc Nicod, Laurent Philippe, Veronika Rehn-Sonigo, Lamiel Toch. "Using Virtualization and Job Folding for Batch Scheduling", 2011 10th International Symposium on Parallel and Distributed Computing, 2011 Publication	1%
5	www.dline.info Internet Source	1%

6	ijsrcseit.com Internet Source	1%
7	www.cs.huji.ac.il Internet Source	1%
8	Submitted to De Montfort University Student Paper	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	Off		