ISSN 2442-6571

# 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, 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 April 13, 2020
Revised June 18, 2020
Accepted June 27, 2020
Available online July 12, 2020

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

ABSTRACT

Grid computing is a collection of heterogeneous resources that is highly dynamic and unpredictable. It is typically used for solving scientific or technical problems that require a large number of computer processing cycles or access to substantial amounts of data. Various resource allocation strategies have been used to make resource use more productive, with subsequent distributed environmental performance increases. The user sends a job by providing a predetermined time limit for running that 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 be rejected if the required resources are not available. The user can re-submit a new request by modifying the parameter until available resources can be found. Eventually, there is a decrease in idle resources between work and resource utilization, and the waiting time will increase. An effective scheduling policy is required to improve resource use and reduce waiting times. In this paper, the FCFS-LRH method is 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 be placed in a logical view, and the job will be sent to the actual resource when it executes. The experimental results show that the proposed model can increase resource utilization by 1.34% and reduce waiting time by 20.47% when compared to existing approaches. This finding could be beneficially implemented in cloud systems resource allocation management.

This is an open access article under the CC–BY-SA license.

## 1. Introduction

Grid computing involves the application of resources in a network to solve one problem at the same time. We are rather used to solving scientific or technical issues, such as high-energy physics, earth observation, and biological applications, all of which require many cycles of computer processing or access to large amounts of data. Grid computing can be considered to be a form of large-scale distributed cluster computing and a type 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 required. When a user's job is submitted, the situation will be handled by a resource broker, who must find and allocate the correct resources for the job. After the resource allocation phase, jobs must be scheduled on existing resources and according to user requirements.

In general, when a user sends a job, they request several processors, memory, and provide the maximum time limit required 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 such as first come first serve (FCFS), three parameters are 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 be rejected if the required resources are not available [3]–[5].

Inelastic reservations are used for the user request parameter 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 of the application to exaggerate the reservation deadline to ensure its completion. User jobs are scheduled, even if booking violations occur because of overlapping jobs [9]–[13]. 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 be 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, both aggressive and conservative backfilling are compared. The result is that aggressive backfilling is more beneficial. Gomes and Dantas [18] propose checking free slots on available resources; if free slots are available, reservations are scheduled, but if such 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 been analyzed in references [19]–[22]. Backfilling was 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 be allocated. The disadvantage of backfilling is that the next job must wait in line until it has finished execution; hence, there is no certainty regarding when the job will complete.

Netto *et al.* [23] conducted a study by rescheduling the allocated work. The experimental results show that the system's utilization will be better if the user waits for up to 75% of the waiting time. Barzegar *et al.* [24] introducing a reservation scheduling algorithm referred to as GELSAR in the grid system. GELSAR will reschedule all new arrivals to find the best solution. New reservations will be rejected if there is no solution. The results of GELSAR outperform other genetic algorithms. Grandinetti *et al.* [25] have researched a local scheduler, where a group of independent jobs has been scheduled, with processing time restrictions provided by the user. It is assumed that all processing nodes are identical. Umar *et al.* [26] introduced first come first serve-ejecting based dynamic scheduling (FCFS-EDS). The results of the FCFS-EDS experiments compared to FCFS without reservations are better in terms of resource utilization [17][23][24]. The advantage of FCFS-EDS is the provision of a one-time notification if a reservation is received because FCFS-EDS works in a logical manner. In another approach, information is processed 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 is found, the job will move to the upper limit of the execution, or if the old job has used the timeslot, then that old job will be shifted so that the new job can be allocated. The impact of shifting the job to the right is a reduction in resource utilization and increased waiting time.

In this research, 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 contributions of this paper are (1) increasing resource utilization in grid computing scheduling, and (2) reducing work waiting times with a comparison workload sent by users.

Section 2 contains the methods and algorithms proposed to solve the problem of resource allocation and waiting time. Section 3 describes the results and analyses comparing FCFS-EDS with FCFS-LRH. The last section provides conclusions.

## 2. Method

### 2.1. Proposed advanced reservation strategy

In this study, a proposed reservation strategy model termed first come first served left-right hole scheduling (FCFS-LRH) is shown in Fig 1. Job requests are sent based on ($numC$, $t_{esr}$, $t_{lsr}$, $t_e$). Incoming user requests will be sorted by the priority of start time of execution, time of execution, and the number of resources required. Jobs will be sorted at each timeslot before being allocated to the virtual view. Work is scheduled at the virtual computing node with a first-fit strategy. If an empty timeslot is obtained between the start time ($t_{esr}$) and the upper limit to begin the job execution (last start time) $t_{lsr}$, the user is notified that the work has been received. If there is no empty slot between $t_{esr}$ and $t_{lsr}$, the job is rejected, and the user is notified. The parameter $t_0$ is the current time. The parameter $tn$ 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. $tel$ 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 are placed in a logical view are still fragmented. Recombination is achieved when a job is executed at the actual computing node [27][28].

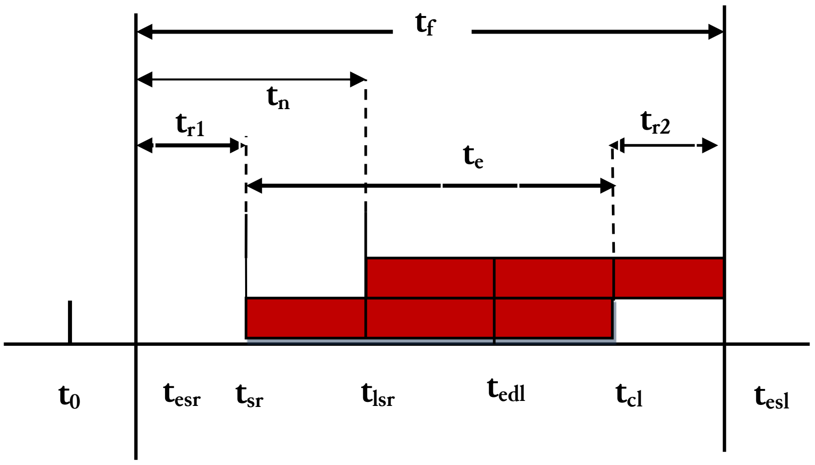

**Fig. 1.** Flexible reservation scheduling

### 2.2. Performance matrix of FCFS-LRH method

The metrics considered for measuring the FCFS-LRH algorithm are resource utilization and waiting time. Average resource utilization is calculated using (1). This formula refers to the comparison of the number of $R_j$ resources executed against the total amount of available resources. In the formula, s indicates the number of slots used by the resource. $ts_i$ refers to the start time when jobs are executed on $R_j$ resources, and $te_i$ refers to the final time that a task completed on $R_j$ resources. The $T$ parameter is 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. Occasionally, resources are not available at the time of reservation. However, resources can be used at different times. In this case, the difference between expected ($startR$) and actual ($startN$) start times is the waiting time.

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

Total Wait Time ($TWT$) is 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

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

### 2.3. Proposed FCFS-LRH algorithm

The proposed algorithm is shown in Fig. 2. The user needs to send a reservation, with parameters qReserv ($t_{esr}$, $t_{lsr}$, $t_e$, $numC$), to order resources. New reservation requests are explained in the FCFS-LRH algorithm below, and sort jobs are based on both arrival ($t_{esr}$) and execution ($t_e$) times and the number of resources required (line 1 to 3).

```
Algorithm: Resource Allocation Algorithm
Input: Job (jobId, $t_{esr}$,$t_{lsr}$,$t_e$, $numC$ )
Output: RU, AWT
1. For j=0:jumJob
2.    sort arrival jobs based on priority $t_{esr}$, $t_e$, $numC$
3. Endfor
4. For i=0:jumJob  // jumjob is the amount of job/timeslot
5.    calculate the value of  d2=$t_{esr}$+$t_e$-1
6.    Search timeslot free with First fit strategy
7.    IF (timeslot==empty) then insert Jobid  value
8.      IF (timeslot!=empty) then execution procedure movejob().
9.   Endfor

10. Procedure moveJob();
11. Initialization; finish=0,suc=false, start=$t_{esr}$, finish=$t_{esr}$+$t_e$-1.
12.  relax=start–$t_{esr}$, $t_r$=$t_{lsr}$–$t_{esr}$,CNs=0.
13. while (!suc and relax <=$t_r$)
14.      For cek=start:finish
15.           set the variable CNs=0
16.         For s=0:atrans.size()
17.             IF atrans.get(s,cek)!=0 then
18.                variable CNs increases by 1
19.             Endif
20.         Endfor
21.            calculate the variable sel=maxC-CNs // maxC is the number of physical nodes
22.         IF (sel>=CN) then
23.             calculate the variable t=start, suc=true
24.             Else
25.                calculate the variable t=cek, finish=start+$t_e$-1, relax=start-$t_{esr}$, suc=true
26.                IF (start>=$t_{lsr}$) then continuous to line 4
27.          Endif
28.    Endfor
29. Endwhile
30. IF (suc==true) then
31.    calculate the variable start=t+1, finish=start+$t_e$-1, relax=start-$t_{esr}$
32.    insert JobID with the first fit strategy
33.    calculate waiting time(AWT)
34. Endif
35.    For y=0:sList.size()
36.      IF (sList.get(y)!=0) then
37.         calculate resource utilization(RU)
38.      Endif
39.  Endfor
```

**Fig. 2.** Resource Allocation Algorithm

Line 4 is a looping to read the jobs that arrive in each timeslot and place them in a virtual view if there are empty timeslots. Line 5 calculate the value of d2, line 6 is the process of finding an empty slot with a first fit strategy starting from CN=0 to CN, and a timeslot starting from $t_{esr}$ to the upper limit (d2). Insert job (Line 7) occurs if an empty slot is found. Line 8, if no empty slots are found, works on the moveJob() procedure, and line 9 is the end of the loop.

The moveJob() procedure is used to move jobs if an empty timeslot is not found, until the upper limit of the start time of execution ($t_{lsr}$). Line 11 and Line 12 are variable initialization. Line 13 to 29 are search loop that begins 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 line 30 to 34 (suc == true), jobId is inserted in the timeslot with the first fit strategy, and the number of jobsID received in the sList dynamic array is saved, and the waiting time is calculated. Line 35 to 38 are used to calculate time slot utilization, where sList is a dynamic array used to hold the number of job IDs used at a certain timeslot.

### 2.3.1. Illustration of FCFS-LRH

An example will be provided to explain FCFS-LRH. If the model knows that the actual compute node $maxC$=6 (c0-c5) is 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 is illustrated in Table 1, where $numC \leq maxC$ and numJob are the numbers of jobs sent by the user. For example, the parameters are given by userID=6 as in Table 1 as follows: userID6 orders 3 time slots starting from timeslot 1 to 6, and it takes 2 computing nodes for 1 independent work, which can be shifted ($t_{esr}$=1, $t_{lsr}$=6, $t_e$=3, $numC$=2, $numJob$=1).

**Table 1.** Reservation from the user

| userID | $t_{esr}$ | $t_{lsr}$ | $t_e$ | 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. 3 shows the logical view of the results of FCFS-LRH from Table 2, the x-axis shows the time slot, and the y-axis displays the virtual computing node. There are six of these nodes, which are displayed on the y-axis; 11 user reservations are allocated from timeslots 0 to 11.

Consider userID6 from Table 2 and Fig. 3. The virtual node given to this user is in timeslot 3 with compute nodes v4 and v5, timeslot 4 with compute nodes v2 and v3, and timeslot 5 with compute nodes v1 and v2. Two computing nodes will do one job at a specific time slot sent by the user. For example, userID12 wants to order three timeslots starting from timeslots 2 to 8, requiring one computational node for one independent job and with the ability to shift from start time to timeslot 8 ($t_{esr}$=2, $t_{lsr}$=8, $t_e$=2, numC=2, jumJob=1); in Fig. 3, userID12 begins the execution time from timeslot 4 to timeslot 6.

**Table 2.** Results of the arrival time of the userId

| userID | $t_{esr}$ | $t_{lsr}$ | $t_e$ | 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 |

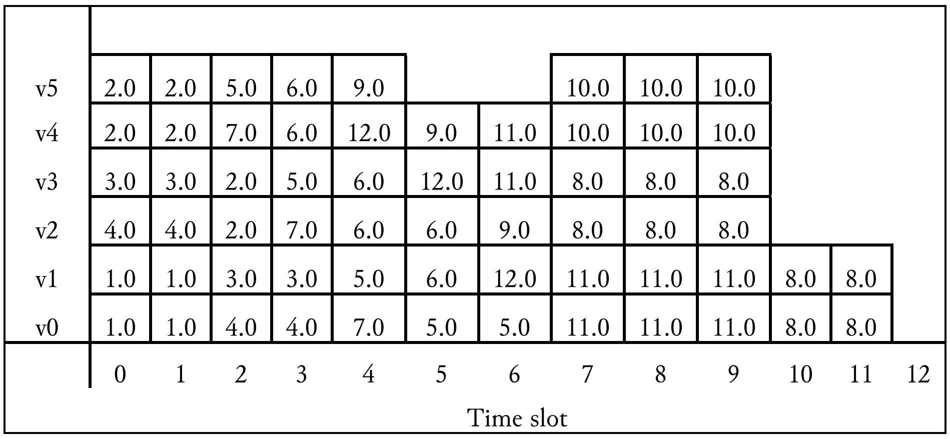
v5 2.0 2.0 5.0 6.0 9.0 10.0 10.0 10.0
v4 2.0 2.0 7.0 6.0 12.0 9.0 11.0 10.0 10.0 10.0
v3 3.0 3.0 2.0 5.0 6.0 12.0 11.0 8.0 8.0 8.0
v2 4.0 4.0 2.0 7.0 6.0 6.0 9.0 8.0 8.0 8.0
v1 1.0 1.0 3.0 3.0 5.0 6.0 12.0 11.0 11.0 11.0 8.0 8.0
v0 1.0 1.0 4.0 4.0 7.0 5.0 5.0 11.0 11.0 11.0 8.0 8.0
0 1 2 3 4 5 6 7 8 9 10 11 12
Time slot

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

In Fig. 3, userID12 will be rejected if a reservation is 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 only sent to users if the reservation is successful (FCFS-LRH works in the virtual view) [27].

### 2.3.2. 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 are mapped to the physical view for all userIDs in Table 2, an approach that guarantees that all jobs can execute on the actual node.

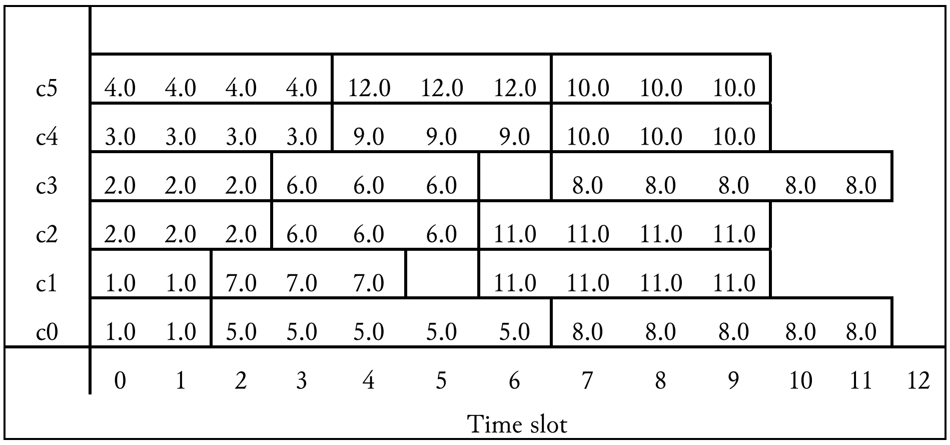
c5 4.0 4.0 4.0 4.0 12.0 12.0 12.0 10.0 10.0 10.0
c4 3.0 3.0 3.0 3.0 9.0 9.0 9.0 10.0 10.0 10.0
c3 2.0 2.0 2.0 6.0 6.0 6.0 8.0 8.0 8.0 8.0 8.0
c2 2.0 2.0 2.0 6.0 6.0 6.0 11.0 11.0 11.0 11.0
c1 1.0 1.0 7.0 7.0 7.0 11.0 11.0 11.0 11.0
c0 1.0 1.0 5.0 5.0 5.0 5.0 5.0 8.0 8.0 8.0 8.0 8.0
0 1 2 3 4 5 6 7 8 9 10 11 12
Time slot

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

#### 2.3.3. Job workload

Each 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 a Poisson distribution.
- Reservation requests are distributed uniformly, which is defined as the execution time ($t_e$).
- Reservation requests are between 0 and 24 hours, uniformly distributed, and they are 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 hours and is uniformly distributed.

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

## 3. Results and Discussion

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

In this research, a work allocation model for resources is proposed to increase resource utilization and reduce the average waiting time. The order of job placement is based on the time of initial execution, the time of the smallest work execution, and the number of lowest resource requirements prioritized so that the utilization of resources will increase, and the waiting time of the work can be reduced. The parameters used in the experiment and for testing the performance are provided in Table 3.

**Table 3.** Parameters used by the experiment

| Parameter | Range |
|---|---|
| Number of Jobs | 300-800 |
| Number of Resources | 1-20 |
| Rate reservation request | 2-3 |
| Percentage of flexibility | 25-100 |

The FCFS-EDS and FCFS-LRH methods included in the flexible advance reservation dynamic scheduling, as in the sample in Table 1, will be used to make it easier to compare resource utilization and an average job waiting time. The results of job placement illustrated in Table 1 in a logical view using FCFS-EDS are 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 are provided in Fig. 6. Table 4 indicates that the average waiting time for FCFS-EDS is 0.83, and for FCFS-LRH, it is 0.72.

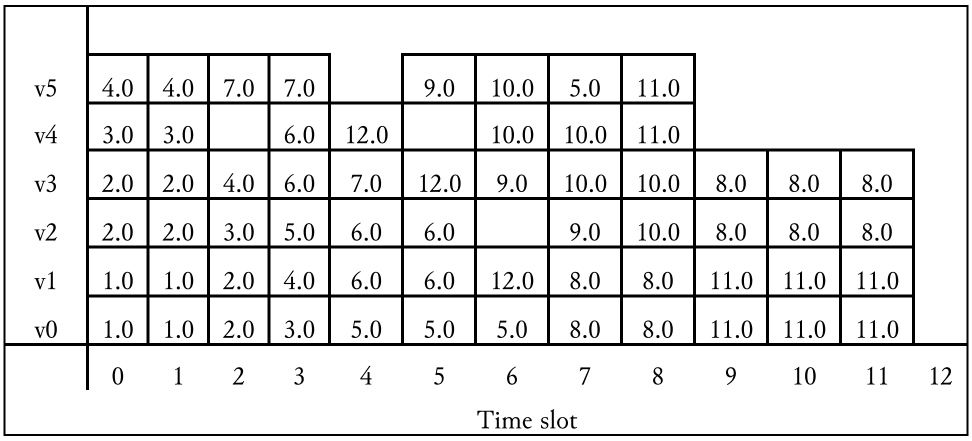
v5 4.0 4.0 7.0 7.0 9.0 10.0 5.0 11.0
v4 3.0 3.0 6.0 12.0 10.0 10.0 11.0
v3 2.0 2.0 4.0 6.0 7.0 12.0 9.0 10.0 10.0 8.0 8.0 8.0
v2 2.0 2.0 3.0 5.0 6.0 6.0 9.0 10.0 8.0 8.0 8.0
v1 1.0 1.0 2.0 4.0 6.0 6.0 12.0 8.0 8.0 11.0 11.0 11.0
v0 1.0 1.0 2.0 3.0 5.0 5.0 5.0 8.0 8.0 11.0 11.0 11.0
0 1 2 3 4 5 6 7 8 9 10 11 12
Time slot

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

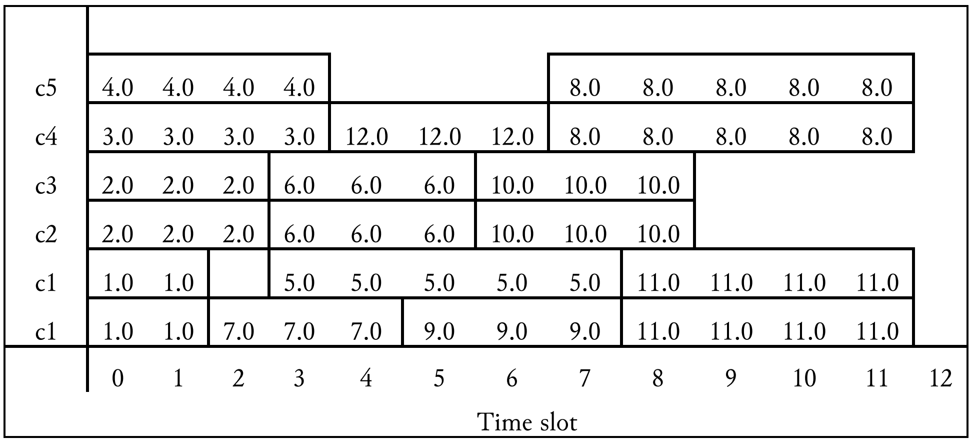
c5 4.0 4.0 4.0 4.0 8.0 8.0 8.0 8.0 8.0
c4 3.0 3.0 3.0 3.0 12.0 12.0 12.0 8.0 8.0 8.0 8.0 8.0
c3 2.0 2.0 2.0 6.0 6.0 6.0 10.0 10.0 10.0
c2 2.0 2.0 2.0 6.0 6.0 6.0 10.0 10.0 10.0
c1 1.0 1.0 5.0 5.0 5.0 5.0 5.0 11.0 11.0 11.0 11.0
c1 1.0 1.0 7.0 7.0 7.0 9.0 9.0 9.0 11.0 11.0 11.0 11.0
0 1 2 3 4 5 6 7 8 9 10 11 12
Time slot

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

The job waiting time for FCFS-LRH is less than for FCFS-EDS because the start time of the situation can advance at an earlier timeslot; hence, the difference in waiting time (start- $t_{esr}$) is smaller. There are 3 FCFS-LRH userIDs for which the advance start time is advanced, namely userID5 advanced 1 timeslot earlier, userID9 earlier 1 timeslot, and userID11 earlier 2 timeslots. Therefore, the total number of timeslots that can use earlier is 4. In contrast, for FCFS-EDS, only 1 userID can advance its execution time more first, namely userID10 of 1 timeslot. Thus, the difference between the timeslot FCFS-LRH with FCFS-EDS that can be improved earlier when the execution is used is 3 timeslots (see Table 4), where the total waiting time (numWT) of FCFS-LRH is 21, and for FCFS-EDS it is 24.

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

| FCFS-EDS | | | | | | | | FCFS-LRH | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *user ID* | *start* | $t_{esr}$ | $t_e$ | *Num* $t_e$ | *wt* | *num WT* | *AWT* | *userI D* | *start* | $t_{esr}$ | $t_e$ | *Num* $t_e$ | *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** |

The impact of the start time is that it can be placed in an earlier slot. Then, the job waiting time can be reduced. The use of timeslot FCFS-LRH is higher than FCFS-EDS; this difference is due to the use of the timeslot earlier in its execution time (see Fig. 4 and compare with Fig. 6). The userID5 is allocated to timeslot 2, userID9 is allocated to timeslot 4, and userID11 is allocated to timeslot 6 using the FCFS-LRH method. The utilization of the FCFS-LRH timeslot is higher than FCFS-EDS starting from timeslot 2 in the 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.

**Table 5.** Resource utilization

| Method | Timeslot | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *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 were carried out using the parameters in Table 3. First, the FCFS-LRH waiting time was compared with FCFS-EDS (backfilling, aggressive backfilling, without reservation). The second experiment compared the resource use between the FCFS-EDS and FCFS-LRH methods. The first experiment involved measuring FCFS-EDS, FCFS-LRH waiting times (backfilling and aggressive backfilling without reservation), the results of which are shown in Table 6. This table demonstrates that job waiting times for FCFS-LRH are shorter than for FCFS-EDS, backfilling, and aggressive backfilling without reservation. This finding is due to the time flexibility affecting 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 has finished executing. Thus, the FCFS-LRH provides a better allocation policy because time flexibility($t_f$) is used to reduce reservation waiting time. The impact of a good allocation policy is that the waiting time value of FCFS-LRH can be reduced compared to FCFS-EDS (backfilling and aggressive backfilling without reservation) for all conditions. The waiting time value is based on the level of reservation arrivals and the number of jobs submitted by users (Table 6). For μ=2 and μ=3, the average reduction in waiting time is 20.47%, whereas the average decrease of waiting time with a level of reservation requests (μ=2) was 36.8%, and for the level of reservation requests (μ=3), waiting time reduction was 9.7% (Table 7).

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

| Method | Number of jobs | | | | | |
|---|---|---|---|---|---|---|
| | *μ=2* | | | *μ=3* | | |
| | *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 |

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

| Method | Rate reservation request | |
|---|---|---|
| | *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 |

The results of the second experiment are shown in Table 8. It can be observed that the average value of timeslot utilization has increased by 1.34% (Table 8), whereas the average resource utilization is 1.17%

for $\mu$= 2 and 1.5% for $\mu$=3 (Table 9). This increase in utilization was due to previous work placement starting from the left-hand side of the timeslot so that the use of the timeslot increased. Jobs will be allocated first on the left-hand side.

**Table 8.** Percentage of utilization results.

| Method | Number of jobs | | | | | |
|---|---|---|---|---|---|---|
| | *μ=2* | | | *μ=3* | | |
| | *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** |

**Table 9.** The average level of reservation requests.

| Method | Rate reservation request | |
|---|---|---|
| | *μ=2* | *μ=3* |
| FCFS-EDS | 92.50 | 93.72 |
| FCFS-LRH | 93.67 | 95.22 |

## 4. Conclusion

Various resource allocation strategies have been used to make resource use more productive. Hence, distributed environmental performance was found to increase. An effective scheduling policy is required to increase resource use and reduce waiting times. In this work, a reservation scheduling strategy referred to as FCFS-LRH is proposed. Jobs that come in this strategy are sorted by priority first, and then jobs will be placed on virtual nodes. Jobs allocated to such nodes will be mapped to physical nodes when they are executed. Work that has been allocated on a virtual node will be guaranteed to be executed on physical resources. Experimentally, the FCFS-LRH method was compared with FCFS-EDS, backfilling, and aggressive backfilling without reservation. FCFS-LRH performance was found to increase in terms of resource utilization, and its use can reduce job waiting times. The results of this study can only be used in local scheduling in grid computing. The next research is to apply the FCFS-LRH method on cloud systems. The algorithm developed for the job on the global scheduler.

## Acknowledgment

The authors thank the Ministry of Research, Technology, and Higher Education (Ristekdikti) of the Republic of Indonesia through BPPDN doctoral scholarship within the 2015 to 2020 fiscal year with registration number 150529056601.

## Declarations

**Author contribution**. All authors contributed equally as the main contributor to this paper. All authors read and approved the final paper.
**Funding statement**. This research received a grant of the BPPDN doctoral program scholarship (2015-2020) from the Ministry of Research, Technology, and Higher Education (Ristekdikti) of the Republic of Indonesia with registration number 150529056601.
**Conflict of interest**. The authors declare no conflict of interest.
**Additional information**. No additional information is available for this paper.

## References

[1] M. Caramia, S. Giordani, and A. Iovanella, "Grid scheduling by on-line rectangle packing," *Networks*, 2004, doi: 10.1002/net.20021.

[2] W. Smith, I. Foster, and V. Taylor, "Scheduling with advanced reservations," in *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*, pp. 127–132, doi: 10.1109/IPDPS.2000.845974.

[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," in *IEEE International Workshop on Quality of Service, IWQoS*, 1999, doi: 10.1109/IWQOS.1999.766475.

[4] I. Foster and C. Kesselman, *The grid 2: Blueprint for a new computing infrastructure*, 2004, doi: citeulike-article-id:340626.

[5] K. Czajkowski *et al.*, "A resource management architecture for metacomputing systems," 1998, pp. 62–82, doi: 10.1007/BFb0053981.

[6] R. Buyya and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing," *Concurr. Comput. Pract. Exp.*, vol. 14, no. 13–15, pp. 1175–1220, Nov. 2002, doi: 10.1002/cpe.710.

[7] A. Sulistio, Kyong Hoon Kim, and R. Buyya, "On incorporating an on-line strip packing algorithm into elastic Grid reservation-based systems," in *2007 International Conference on Parallel and Distributed Systems*, 2007, pp. 1–8, doi: 10.1109/ICPADS.2007.4447738.

[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, Mar. 2016, doi: 10.1007/s10586-015-0530-0.

[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, Jun. 2001, doi: 10.1109/71.932708.

[10] C. Castillo, G. N. Rouskas, and K. Harfoush, "On the Design of Online Scheduling Algorithms for Advance Reservations and QoS in Grids," in *2007 IEEE International Parallel and Distributed Processing Symposium*, 2007, pp. 1–10, doi: 10.1109/IPDPS.2007.370226.

[11] P. Xiao, Z. Hu, X. Li, and L. Yang, "A Novel Statistic-based Relaxed Grid Resource Reservation Strategy," in *2008 The 9th International Conference for Young Computer Scientists*, 2008, pp. 703–707, doi: 10.1109/ICYCS.2008.117.

[12] B. S. S. Rani, R. Venkatesan, and R. Ramalakshmi, "Resource reservation in grid computing environments: Design issues," in *2011 3rd International Conference on Electronics Computer Technology*, 2011, pp. 66–70, doi: 10.1109/ICECTECH.2011.5941858.

[13] P. Xiao and Z. Hu, "Relaxed resource advance reservation policy in grid computing," *J. China Univ. Posts Telecommun.*, vol. 16, no. 2, pp. 108–113, Apr. 2009, doi: 10.1016/S1005-8885(08)60213-7.

[14] A. Shukla, S. Kumar, and H. Singh, "An Improved Resource Allocation Model for Grid Computing Environment," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 104–113, Feb. 2019, doi: 10.22266/ijies2019.0228.11.

[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, doi: 10.1109/NOF.2016.7810118.

[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, pp. 26–40, Mar. 2016, doi: 10.1016/j.comcom.2015.10.016.

[17] H. R. Moaddeli, G. Dastghaibyfard, and M. R. Moosavi, "Flexible Advance Reservation Impact on Backfilling Scheduling Strategies," in *2008 Seventh International Conference on Grid and Cooperative Computing*, 2008, pp. 151–159, doi: 10.1109/GCC.2008.85.

[18] E. Gomes and M. A. R. Dantas, "Towards a Resource Reservation Approach for an Opportunistic Computing Environment," *J. Phys. Conf. Ser.*, vol. 540, p. 012002, Oct. 2014, doi: 10.1088/1742-6596/540/1/012002.

[19] A. Mishra, "An enhanced and effective preemption based scheduling for grid computing enabling backfilling technique," in *2015 International Conference on Advances in Computer Engineering and Applications*, 2015, pp. 1015–1018, doi: 10.1109/ICACEA.2015.7164855.

[20] O. Dakkak, S. Awang Nor, and S. Arif, "Scheduling through backfilling technique for HPC applications in grid computing environment," in *2016 IEEE Conference on Open Systems (ICOS)*, 2016, pp. 30–35, doi: 10.1109/ICOS.2016.7881984.

[21] S. Leonenkov and S. Zhumatiy, "Introducing New Backfill-based Scheduler for SLURM Resource Manager," *Procedia Comput. Sci.*, vol. 66, pp. 661–669, 2015, doi: 10.1016/j.procs.2015.11.075.

[22] R. Istrate, A. Poenaru, and F. Pop, "Advance Reservation System for Datacenters," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 2016, pp. 637–644, doi: 10.1109/AINA.2016.106.

[23] M. A. S. Netto, K. Bubendorfer, and R. Buyya, "SLA-Based Advance Reservations with Flexible and Adaptive Time QoS Parameters," 2007, pp. 119–131, doi: 10.1007/978-3-540-74974-5_10.

[24] B. Barzegar, A. M. Rahmani, K. Zamanifar, and A. Divsalar, "Gravitational Emulation Local Search Algorithm for Advanced Reservation and Scheduling in Grid Computing Systems," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009, pp. 1240–1245, doi: 10.1109/ICCIT.2009.319.

[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, Aug. 2015, doi: 10.1007/s10732-015-9287-0.

[26] R. Umar, A. Agarwal, and C. R. Rao, "Advance Planning and Reservation in a Grid System," 2012, pp. 161–173, doi: 10.1007/978-3-642-30507-8_15.

[27] A. Pujiyanta, L. E. Nugroho, and Widyawan, "Planning and Scheduling Jobs on Grid Computing," in *2018 International Symposium on Advanced Intelligent Informatics (SAIN)*, 2018, pp. 162–166, doi: 10.1109/SAIN.2018.8673372.

[28] A. Pujiyanta, L. E. Nugroho, and Widyawan, "Advance Reservation for Parametric Job on Grid Computing," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–5, doi: 10.1109/ICIC47613.2019.8985978.

[29] R. V. Lopes and D. Menasce, "A Taxonomy of Job Scheduling on Distributed Computing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3412–3428, Dec. 2016, doi: 10.1109/TPDS.2016.2537821.

[30] M. Carvalho and F. Brasileiro, "A User-Based Model of Grid Computing Workloads," in *2012 ACM/IEEE 13th International Conference on Grid Computing*, 2012, pp. 40–48, doi: 10.1109/Grid.2012.13.

[31] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," *J. Parallel Distrib. Comput.*, vol. 63, no. 11, pp. 1105–1122, Nov. 2003, doi: 10.1016/S0743-7315(03)00108-4.

[32] A. Iosup, D. H. J. Epema, J. Maassen, and R. van Nieuwpoort, "Synthetic Grid Workloads with Ibis, Koala, and Grenchmark," 2007, pp. 271–283, doi: 10.1007/978-0-387-47658-2_20.