International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 3
ISSN: 2088-5334

Job Scheduling Strategies in Grid Computing

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

[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*
*Corresponding author: *ardi.pujiyanta@mail.ugm.ac.id*

Abstract— **Grid computing can be thought of as large-scale distributed cluster computing and distributed parallel network processing. Users can obtain enormous computing power through network technology, which is challenging to get from a single computer. Job scheduling in grid computing is a critical issue that affects the overall grid system capability. In traditional scheduling, jobs are placed in queues, waiting for the availability of resources. Reservations reject if the required resources not obtained at the specified time. The impact that arises is the reduced use of resources. The scheduling algorithm and the parameters used to perform the work may vary, such as execution time, delivery time, and the number of resources. There is no guarantee when the job will execute using the scheduling algorithm. Therefore, it is necessary to improve resource utilization in the grid system and ensure that jobs will be carried out. This paper proposes a reservation scheduling strategy for MPI work, First Come First Serve Left Right Hole (FCFS-LRH). MPI jobs execute simultaneously, using more than one resource for implementation. When Completed, user MPI jobs will be scheduled on virtual compute nodes and mapped to actual compute nodes. The experimental results show that the increase in resource utilization strongly influenced by time flexibility.**

Keywords— **Advance reservation; MPI Job; FCFS-LRH; grid systems.**

Manuscript received 22 Oct. 2019; revised 13 Dec. 2020; accepted 25 Jan. 2021. Date of publication 30 Jun. 2022.

I. INTRODUCTION

The availability of high-speed networks and the effective use of computers have increased the demand for grid computing. Cluster computing is a form of large-scale distributed parallel network processing and can be thought of as grid computing[1]. Users can obtain enormous computing power through network technology, which is difficult to obtain from a single computer. In grid computing, resource allocation and job scheduling are critical issues affecting overall grid system capability. Jobs in traditional scheduling will be placed in queues waiting for required resources. Parameters used in the execution of work may vary, such as execution time, delivery time, and required resources[2][3]. There is no guarantee when the work will execute using the scheduling algorithm[4].

Start time, execution time, and the number of resources are parameters that need to be provided in a rigid scheduling system, if a user requests a resource to do his job[5]. The consequence of a rigid reservation mechanism will cause idle resources between jobs, this is due to the unavailability of the required resources within a predetermined time limit. The impact of idle resources, will cause resource utilization to decrease. Shi, *et al.*[6] propose the use of a new elastic resource, the job with the highest priority will be executed first. Sulistio et al. [7] proposes an elastic reservation with the user query parameter used as a soft constraint. The request is not rejected, but the user will be given an alternative choice of time by the reservation system. After the user selects one of the given alternative options, the user resubmits the request. However, this time, the request was sent using the rigid reservation method, because there is a guarantee of resource availability. Overlapping timeslots are proposed to overcome the problem of decreasing resource utilization [8],[9],[10].

Job start time period used by Chunming et al. [11] in his research called slack-time. This mechanism is called FIRST (Flexible Reservation using Slack Time). Slack-time can reduce rejection rates and improve resource utilization; in a way that the work start time can be shifted. The system will reschedule all non-executed reservations one by one if a new reservation arrives, according to FIFO (First In First Out) rules. A new reservation request is rejected if a solution is not found. Use of overlapping time slots [8] address the problem of decreasing resource utilization caused by reservations.

Time spent on a job tends to exceed the reservation time limit in getting the job done. User jobs will still be scheduled, even if there are overlapping job orders. In flexible reservation, the user's work is planned and given flexible constraints with varying start times and in certain time intervals [12], [13],[14],[15],[16].

Eliza et al. [17] propose checking for empty slots on available resources. If there is no empty slot during the reservation request, the available empty slot will be reserved. First Come First Serve Ejecting Based Dynamic Scheduling (FCFS-EDS) is used to improve resource utilization in the grid system on the local scheduler[18]. The disadvantage of FCFS-EDS is that if the previous job cancels the job before it is executed, the future job cannot occupy the space still used by the canceled job then FCFS-EDS can only shift to the right. Grandinetti et al. [19] have investigated a group of independent jobs scheduled, with user-provided processing time constraints. All processing nodes are assumed to be identical. Workloads consist of batch jobs that require the execution of space sharing. Thus, queued work can be started if there are nodes that match the required capacity.

Research[20][21][22][23][22][24] researched the impact of using backfilling algorithms in flexible reservations. The backfilling strategy proposed is to make reservations early by making space for new reservations to be allocated. The drawback of backfilling is no certainty when the job is executed because the next job must wait until the previous job is executed. Anju Shukla et al.[25] proposed an algorithm to reduce the average waiting time of a queued job. The job with the least workload will execute first. The algorithm will determine the least resources in the execution of the work. So that the resulting schedule with the application of work based on the shortest workload. If no resource is available as needed, the job is placed in a queue until the resource is found. To ensure there is a guarantee that the work will be executed in the future. The reservation scheduling strategy is proposed based on a virtual view instead of the physical view reported in the literature.

II. Material and Method

Experiments have carried out on planning and scheduling strategies for MPI work. Characteristics of the workload in this experiment are as follows [18][19][26][27][28][29].

- Reservation requests (μ=3 and 4), follow the Poisson distribution.
- Request Execution time (t_e), they uniformly distributed.
- The earliest start time requests (t_{esr}) uniformly distributed between 0 and 24.
- The percentage of flexible reservation users is randomly selected.
- Request reservation time (t_f) is between 1 and 12, evenly distributed.

The percentage sliding window of 12-time slots(1 hour) calculated resource utilization. The proposed method is compared with no reservation. The total amount of resources used is 30, the percent flexibility is between 25%-100%, and the number of jobs used is between 615 and 800.

A. Proposed Advanced Reservation Strategy

The proposed reservation strategy, named First Come First Serve Left Right Hole Scheduling (FCFS-LRH), is used to improve resource utilization in the grid system. The user job is scheduled on the virtual compute node before the job is executed. A flexible reservation is an execution time (te) less than the execution time interval (t_{esr}) to the end of execution time (t_{esl}), which is shown by the time diagram in Fig 1. Jobs sent with parameters (JumCN, t_{esr}, t_{lsr}, t_e). Once the reservation is received, it looks for free space on the virtual compute node with the earliest start time. If there is, then the job is placed at test time. Resources allocated. If no timeslot is found, the job will shift to the job start time limit. The notification interval is the difference between t_{lsr} and t_{esr}.

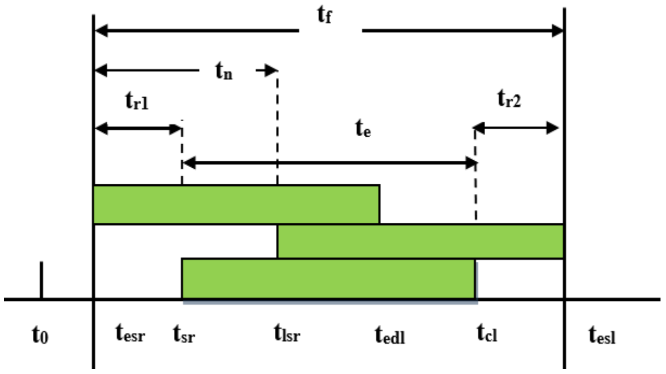


Fig. 1 Proposed flexible scheduling of reservations

t_0: Current time
t_{esr}: Lower limit of the job start time.
t_{sr}: Time to start job ($t_{esr} \leq t_{sr} \leq t_{lsr}$)
t_{lsr}: last start time, defined as $t_{lsr} = t_{esl} - t_e = t_n$
t_{esl}: Upper limit for ending time running a job
t_e: Time of execution of job.
t_n: Notification time.
t_{r1},t_{r2} : t_{r1}(left hole), t_{r1}(right hole), t_r defined by $t_r = t_{r1} + t_{r2} = t_{esl} - t_{esr} - t_e$
t_{edl}: The lower limit, defined as $t_{edl} = t_{esr} + t_e$
t_{cl}: Time to get the job done ($t_{edl} \leq t_{cl} \leq t_{esl}$)
t_f: Flexibility time, $t_f = t_{esl} - t_{esr}$
f: Level of flexibility, set as $f = t_f / t_e$, with $f \geq 1$, (if $f = \infty$, a job considered a not job reservation mode, if $t_{esr} = t_0$ and f=1. reservation considered with the highest priority leads to a direct scheduling mode [18].
userId: User identification
jobId: Job identification
JumCN: The number of computer resources needed
MaxCN: Total amount of computer resources.

The function of t_{r1} (left hole), is to provide free space, if there is the next job in, then the previous slot can be shifted to the right so that work that requires the next free space can occupy it.

The t_{r2} (right hole) function, for example, user1 needs 5 minutes of work execution time, starting from t=10 to t=11, while user2 needs time to execute the work for 15 minutes, the earliest start time of execution can start from t=10 and the last time the job starts is t=12 (Fig. 2). Then user1 cancels his work, while user9 enters the execution of his work for 10 minutes, starting from t=13 to t=15, then user2 can be shifted left to give user3 space to occupy the space t=13 to t=15 (Fig. 3).

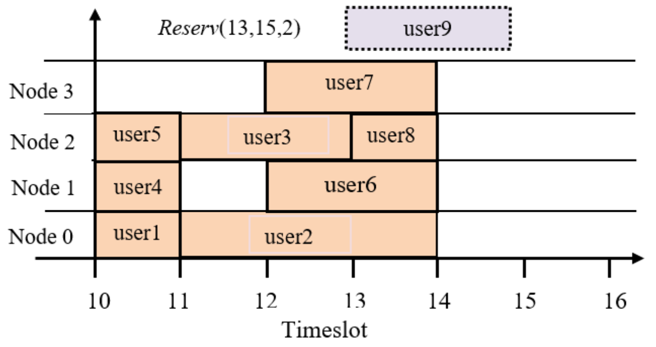


Fig. 2 Time slot diagram with four compute nodes, where eight users have allocated each time slot.

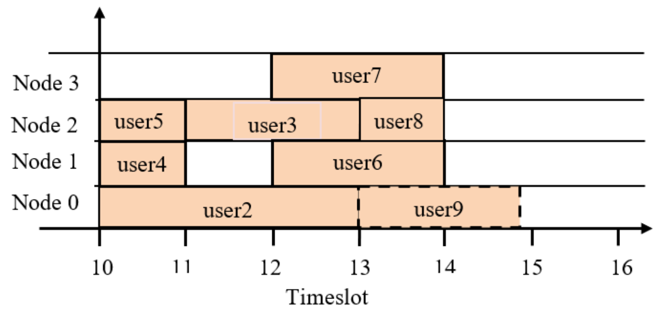


Fig. 3 The reservation is flexible. A dotted box shows the new reservation allocation.

B. *Proof of Concept FCFS LRH*

For example, maxS (maximum value of computational node) in slot t, is:

$$S(t) = \{s(t)_1\, , s(t)_2\, , s(t)_3, \dots , s(t)_n\,\} \qquad (1)$$

maxS is the value of the work planning array, which is shown by equation (1), where the i element of s(t) is s(t)(i), with the id job already executed at the computational node i on the timeslot t. pS(t) is the insertion of a new array of jobs in S(t) in timeslot t:

$$pS(t) - pS(t+1) \qquad (2)$$

Equation (2) is a Job executed in slot t

$$pS(t+1) - pS(t) \qquad (3)$$

Equation (3) is a Job executed in slot t+1

$$pS(t) \cap pS(t+1) \qquad (4)$$

Equation (4) is a Job executed in slot t to slot t+1.

$$Bm = m(t) \times m^{-1}(t+1) \qquad (5)$$

If the job permutation matrix in time slot t is m(t) and the inverse matrix of permutations in time slot t+1 is $m^{-1}(t+1)$. Then Equation (5) shows a partial identity matrix (Bm), where Bm: jobs executed at the same compute nodes from timeslot t to t+1.

```
If S(t+1)(j)=S(t)(i) then
        Bm(i,j)=1
Else
        Bm(i,j)=0
```

Bm(i,j) =1 : Jobs in time slot t are executed on resource i, and jobs in time slot t+1 are executed on resource j. For example, if it knows six users are sending jobs, then the system randomly breaks the role. Table 1 below illustrates each virtual resource that has allocated work.

TABLE I
ALLOCATION OF JOBS TO VIRTUAL RESOURCES

Time Slot	Resource				
	N1	N2	N3	N4	N5
t	3	7	4	5	8
t+1	8	-	7	6	4

From Table 1, it can see that the job from user4 has allocated to the timeslot t with the N3 resource number, and at timeslot t+1 in the N5 resource number.

S(t)=[3 7 4 5 8]
S(t+1)=[8 - 7 6 4]
pS(t) - pS(t+1)= [3 5]
pS(t+1) - pS(t) = [6]
pS(t) ∩ pS(t+1)= [8 7 4]
Solution:

a. Calculate the permutation matrix

$$S^{-1}(t+1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

b. Calculate the difference in pS(t+1) with pS(t) :
H = pS(t+1) - pS(t)

c. Combine H with the $S^{-1}(t+1)$, permutation matrix, obtained

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

d. Multiply pS(t+1) with the complete permutation matrix Y

$$pS(t+1)x \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} = [- \quad 7 \quad 4 \quad 6 \quad 8]$$

TABLE II
RESULTS OF CALCULATION OF DATA TO S(t +1)

	Node				
Timeslot	N1	N2	N3	N4	N5
t	3	7	4	5	8
t+1	-	7	4	6	8

Table 2 explains that userid3 job executed at N1 resource at timeslot t. Job userid4 is run at N3 from timeslot t to t+1. Job userid5 runs at the N4 at timeslot t. Job userid6 run by an N4 resource at timeslot t+1. userid7 executed by N2 from timeslot t to t+1, and job userid8 is run by resource N5 starting timeslot t to t+1.
So the partial matrix identity

$$I = S(t)\, x\, S^{-1}(t+1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Job userid4 userid7, and userid8 executed in the same resource from timeslot t to timeslot t+1 [26].

C. Algorithm FCFS-LRH

User-submitted parameter, qReserv(tesr, tlsr, texe, jumCN), to reserve resources (step1). The new reservation request is described in the FCFS-LRH algorithm below, and the parameter initialization shows step2. Suppose that there has been an old qReserv(n-1) reservation, which previously allocated using the FCFS-LRH algorithm. Then comes the new qReserv(n) reservation that requests resources for scheduled (step3 and step4). Without affecting the qReserv(n-1) reservation plan that has allocated. The new reservation will search for resources, up to the upper bound to start job execution (last start time), as step4 shows. If the required resource is found, the search will be successful. The algorithm will create a new successful plan described by the qReserv(n) reservation on the virtual node. If the requested resource is not found within the flexible limit, the algorithm will move the old qReserv(n-1) reservation shown in step5 and step6, to allocate qReserv(n). If the qReserv(n-1) reservation fails, then the qReserv(n-1) reservation plan must be returned to the original state shown in step7.

After the qReserv(n) reservation is scheduled on the virtual node, the next step is to recombine the fragmented job on the virtual node using an algorithm that refers to [26], so that it can execute on the physical node.

Algorithm I

```
Function searchSlotFree
Step1: Input (userId, jobId,tesr,tlsr,te,jumCN)
Step2: Parameter initialization
time=0, minSlot=0,seltr=tlsr-tesr,succ=false.
Step3: if(!succ), the condition not fulfilled go to step5
       Calculate start=tesr, finish=tesr+texe–1, flexible=start–
       tesr.
Step4: Calculate job shift
     while(!succ and flexible<=seltr), the condition is not
fulfilled go to step5
            minSlot = nodeFree(start, finish);
      if (jumCN <= minSlot )
      alloc(userId,jobId,tesr,start,tlsr,texe, jumCN)
      succ=true
      Otherwise Calculate start=time+1,
      finish=start+texe–1, flexible=start-tesr
Step5: Calculate start=tesr, finish=tesr+texe–1,
      flexible=start-tesr;
Step6: Calculate insertion and job shift
      while(!succ and flexible<=seltr), the condition is not
fulfilled go to step7
      minSlot=nodeFree(start, finish);
      if (jumCN<=minSlot)
      alloc(userId,jobId,tesr,start,tlsr,texe,jumCN)
      succ=true
      Otherwise
      if (!insRes(userId, jumCN-minSlot)
      Calculate start=time+1, finish=start+texe-1,
      flexible=start-tesr
Step7:    if(!succ) // return the job to its original
      place, because the job failed to shift
      backInsJob()
Step8: return succ
```

Algorithm 2 used to search for the minimum timeslot for qReserv(n) reservations that arrive. Step1 shows the initial value of the number of slots in pSlot[start]. Variable time used to receive the slot number value at the beginning of the search. Step2 searches for the minimum slot value required for the qReserv (n) reservation, starting at the start to finish interval. Step3 returns the minSlot variable value if the search is complete.

Algorithm II

```
Function nodeFree(start, finish)
Step1: Calculate minSlot = pSlot[start].getFree,
  time = start;
Step2: calculate the minimum timeslot at the pSlot
  For i=start,…,i<=finish
    If (pSlot[i].getFree() < minSlot)
      minSlot = pSlot[i].getFree
      time = i
Step3: return minSlot
```

According to the minimum timeslot that the user needs, algorithm 3 is used to allocate qReserv(n) to structured timeslot data. Step1 is used to calculate the value of the variable finish. Step2 is used to create a comp object that contains user information. Step3 is used to add the job (comp) component to the timeslot pSlot[start] because of the scheduled success. Step4 executes the AllocFF procedure to allocate user requests to the virtual node. Step5 updates timeslot on pSlot because of the addition of jobs(comp).

Algorithm III

```
Procedure alloc
Step1:    calculate finish=start+texe-1;
Step2:    create a comp object that contains (userId, jobId,
tesTime, startTime, tlsstartTime, execTime, endTime, jumCN);
Step3:    Append(start, comp);
Step4:  AllocFF(start, comp);
Step5:    calculate timeslot on pSlot using loops
    For i=start,…,i<=finish
      pSlot[i].setFree(pSlot[i].getFree-jumCN);
```

Algorithm 4 is used to allocate jobs ID on specific timeslot virtual nodes that paired with specific resource numbers, after successful job placement.

Algorithm IV

```
Procedure AllocFF
Step1: create cellx object containing (userID, jobID, startTime,
tlsstartTime)
Step2: calculate For c=0,…,c<comp.jumCN
Step3: calculate For k=0,…,k<comp.texe
Step4: calculate For j=0,…, j<maxCN
    If (cell[start+k][j].userID= =0)
      cell[start+k][j]= cellx,
      break.
```

Algorithm 5 used to reallocate the jobId in its original place because it has failed to shift right. Explanation of algorithm 5 as follows: step1 is used to create objects. Step2 is used to test the condition of the stack, and this stack contains jobs that have failed to shift right. If the stack is not empty, take the top stack, S1 (step 3), give the startTime attribute of S1 on the variable start. The loop on step4 looks for jobs that failed to shift, to return to their original place. Look for the same S1 (step5) value in timeslot pSlot[start], if found, delete the job in pSlot[start] and delete the job on the virtual node. Update startTime and endTime on comp.setstartTime(start-1). Return the job to the original timeslot by calling the Append (start-1, comp) procedure, because it has failed to shift right at timeslot pSlot[start]. The next step is to reduce the timeslot on the left side of the pSlot[start-1], with jumCN, and add the number CN to the right side of pSlot[endTime+1]. The break commands. If to exit the step4 loop. Continue the search until all S1 values are removed from the stack or until the stack is empty (step2).

Algorithm V

```
Procedure backInsJob
Step1: Create a stack object, listComp, comp, S1
Create stack=new Stack<Component>
Create listComp=new LinkedList<Component>
Create Component comp, S1
Step2: while(stack is not empty)
// The stack contains jobs that failed to move right
// take the top stack on the stack for example S1
Step3: Calculate S1=stack.pop, start=S1.startTime;
Step4: For i=0 to i<pSlot[start].listComp.size
Step5: If(pSlot[start].listComp(i)==S1)
    comp=pSlot[start].listComp.remove(i);
    delete cell objects on the virtual node
    comp.setstartTime(start-1)
   Append(start-1, comp)
   Reduce the timeslot on the left side of pSlot[start-1], with jumCN;
   add timeslot on the right side of pSlot[endTime+1], with jumCN;
      break;
```

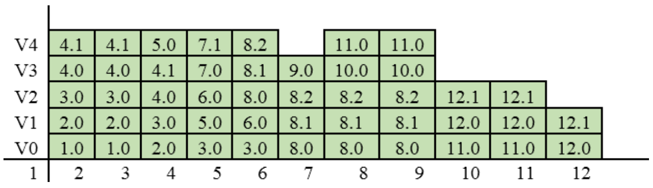
D. *Application of FCFS-LRH on MPI jobs*

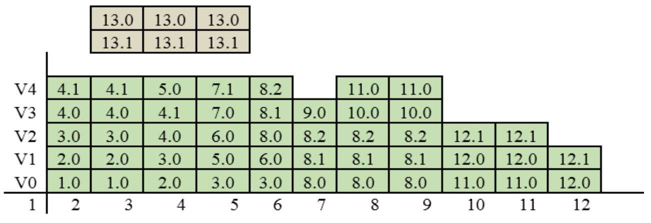
An example will given to explain the FCFS-LRH. The number of virtual nodes (v0-v4) used must be the same as the physical compute nodes MaxCN=5(c0-c4) used. Table 3 shows the order of reservation arrivals, where JumCN $\leqslant$ MaxCN and JumJob are the number of jobs users submitted. Suppose the parameters given by userId=4 as in Table 3 are as follows: userId4 orders three timeslots on timeslots 2 to 4, takes two compute nodes for 1 independent job, and cannot be shifted (t_{esr}=2, t_{lsr}=2 , t_e=3, jumCN=2, jumJob=1).

TABLE III
Reservation parameters in MPI jobs

UserId	t_{esr}	t_{lsr}	te	JumCN	JumJob
1	2	2	2	1	1
2	2	2	3	1	1
3	2	2	5	1	1
4	2	2	3	2	1
5	4	4	2	1	1
6	5	5	2	1	1
7	5	5	1	2	1
8	6	6	4	3	1
9	7	7	1	1	1
10	8	8	2	1	1
11	8	8	4	1	1
12	9	10	3	2	1

Fig 4 shows the FCFS-LRH results for MPI jobs. The x-axis shows the timeslot, and the y-axis shows the virtual compute nodes. Since there are five virtual compute nodes, which will be displayed on the y-axis. Twelve user reservations have been allocated from timeslot 2 to 12. Consider userId6 from Table 3. The virtual node assigned to userId6 is in the timeslot (tesr=5) with compute node v2, and in timeslot 6 with compute node v1, it is only one job. (requires two-time slots) that the user has submitted. Suppose three timeslots ranging from timeslot 8 to 13 are ordered by userId13, require two compute nodes for one independent job and can be shifted to timeslot 13(tesr=8,tlsr=13,te=3,jumCN=2,jumJob=1). See Fig 5.


V4 4.1 4.1 5.0 7.1 8.2 11.0 11.0
V3 4.0 4.0 4.1 7.0 8.1 9.0 10.0 10.0
V2 3.0 3.0 4.0 6.0 8.0 8.2 8.2 8.2 12.1 12.1
V1 2.0 2.0 3.0 5.0 6.0 8.1 8.1 8.1 12.0 12.0 12.1
V0 1.0 1.0 2.0 3.0 3.0 8.0 8.0 8.0 11.0 11.0 12.0
1 2 3 4 5 6 7 8 9 10 11 12
Fig. 4 Allocation of ten reservations in virtual resources


13.0 13.0 13.0
13.1 13.1 13.1
V4 4.1 4.1 5.0 7.1 8.2 11.0 11.0
V3 4.0 4.0 4.1 7.0 8.1 9.0 10.0 10.0
V2 3.0 3.0 4.0 6.0 8.0 8.2 8.2 8.2 12.1 12.1
V1 2.0 2.0 3.0 5.0 6.0 8.1 8.1 8.1 12.0 12.0 12.1
V0 1.0 1.0 2.0 3.0 3.0 8.0 8.0 8.0 11.0 11.0 12.0
1 2 3 4 5 6 7 8 9 10 11 12
Fig. 5 Reservations from new users on MPI jobs

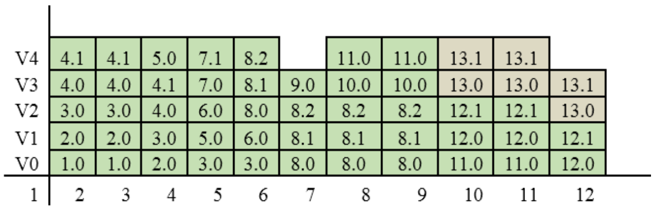
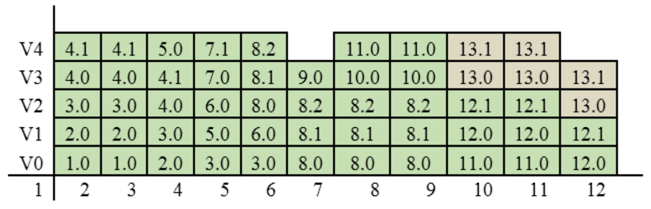

V4 4.1 4.1 5.0 7.1 8.2 11.0 11.0 13.1 13.1
V3 4.0 4.0 4.1 7.0 8.1 9.0 10.0 10.0 13.0 13.0 13.1
V2 3.0 3.0 4.0 6.0 8.0 8.2 8.2 8.2 12.1 12.1 13.0
V1 2.0 2.0 3.0 5.0 6.0 8.1 8.1 8.1 12.0 12.0 12.1
V0 1.0 1.0 2.0 3.0 3.0 8.0 8.0 8.0 11.0 11.0 12.0
1 2 3 4 5 6 7 8 9 10 11 12
Fig. 6 Reservations from new users have allocated using FCFS-LRH (Virtual)

Fig 6 shows that user13 will be rejected if the reservation is made using conventional reservation or rigid reservation. The same jobs in the same timeslot are allocated to different virtual compute nodes. If the reservation is successful a notification will be sent to the user only once. The FCFS-LRH scheduling approach works on the virtual view, whereas in other methods, it must send the revisions made[30].

E. *Mapping from Virtual Nodes to Actual Computing Nodes*

The FCFS-LRH application for a reservation using MPI jobs, shown in Fig 7, will be guaranteed that all posts to executed shown in Fig 8 (Physical).


V4 4.1 4.1 5.0 7.1 8.2 11.0 11.0 13.1 13.1
V3 4.0 4.0 4.1 7.0 8.1 9.0 10.0 10.0 13.0 13.0 13.1
V2 3.0 3.0 4.0 6.0 8.0 8.2 8.2 8.2 12.1 12.1 13.0
V1 2.0 2.0 3.0 5.0 6.0 8.1 8.1 8.1 12.0 12.0 12.1
V0 1.0 1.0 2.0 3.0 3.0 8.0 8.0 8.0 11.0 11.0 12.0
1 2 3 4 5 6 7 8 9 10 11 12
Fig. 7 Allocation of (virtual) reservations for MPI jobs

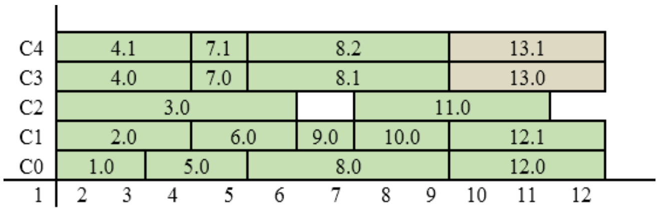


Fig. 8 Results of mapping on actual computational nodes for MPI jobs

III. Results and Discussion

The FCFS-LRH comparison results without a reservation for μ=3, percentage of flexibility=25%-100% shown in Fig. 9-12, which shows the benefits of FCFS-LRH, better than without a reservation. Similarly, Fig 13-16 results from the FCFS-LRH comparison with no reservation starting from μ=4, percentage of flexibility=25% -100% show better utilization of FCFS-LRH scheduling than strategies without reservation resource use strongly influenced by time flexibility(t_f). Table 4 summarizes experimental results with μ=3 and μ=4, percent flexibility between 25%-100%, the number of jobs between 615 to 800. The average usefulness of FCFS-LRH scheduling is better than the average utilization without reservation. Fig 17 shows where the highest percentage of utilization was 94.95%, obtained when the rate of flexibility was 75% with μ=3. Fig 18 shows where the highest rate of utilization was 97.43%, received when the rate of flexibility was 100% with μ=4. While Fig 19 indicates an increase in utilization by 3.97% using FCFS LRH got when the rate of flexibility is 75% with μ=3.

Fig 20 comparison of utilization between FCFS-LRH and FCFS-EDS. for μ=3. If it is assumed that 2% of jobs have scheduled to cancel their jobs, and then new jobs are entered, then resource utilization has increased by an average of 1.38%, from 2% of jobs that cancel their jobs. Likewise, Fig 21 for μ=3 shows that if it assumed that 2% of jobs scheduled to cancel jobs, then there are new jobs coming, then resource utilization has increased by an average of 1.79% 2% of jobs that canceled jobs.

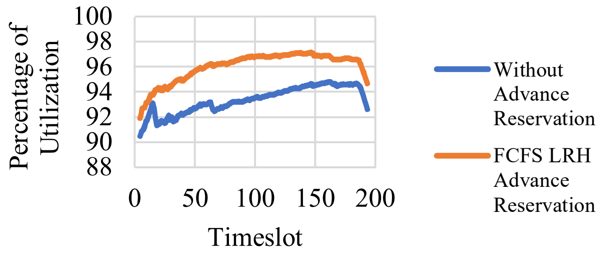


Fig. 9 Scheduling of Advance Reservation (FCFS-LRH) and without reservation, with μ=3, flexibility =25%

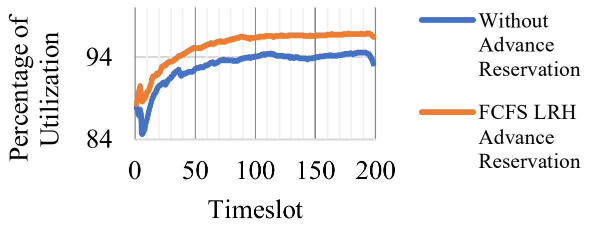


Fig. 10 Schedule of Advance Reservation (FCFS-LRH) and without reservation, with μ = 3, flexibility = 50%

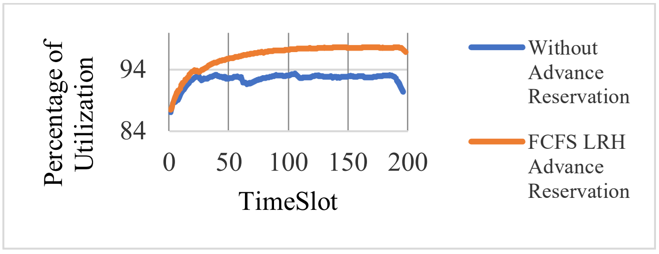


Fig. 11 Scheduling of Advance Reservation (FCFS-LRH) and without reservation, with μ = 3, flexibility =75%

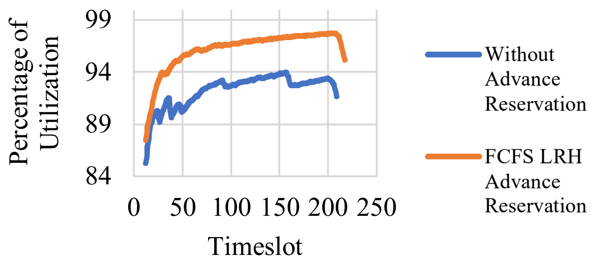


Fig. 12 Scheduling of Advance Reservation (FCFS-LRH) and without Reservation, with μ = 3, flexibility = 100%

TABLE IV
FCFS-LRH ADVANCE RESERVATION AND WITHOUT RESERVATION

Total Job	Flexibility Percentage (%)	(μ)	Without *Advance Reservation* (%)	*Advance Reservation* (%) FCFS-LRH
615	25	3	92.39	94.88
673	50	3	92.11	94.43
618	75	3	91.14	94.95
601	100	3	91.08	94.68
793	25	4	92.92	96.66
799	50	4	92.37	95.93
800	75	4	90.62	94.74
786	100	4	93.28	97.43

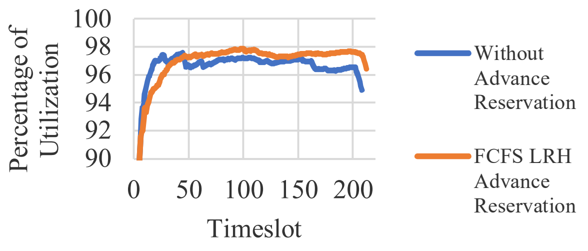


Fig. 13 Scheduling of Advance Reservation (FCFS-LRH) and without reservation, with μ=4, flexibility=25%

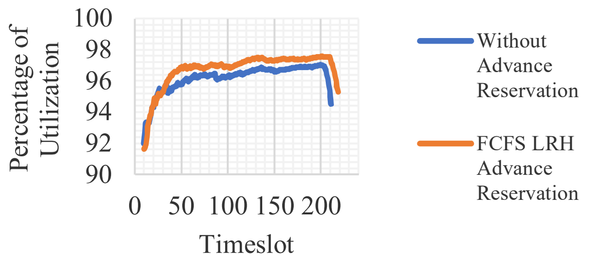


Fig. 14 Schedule of Advance Reservation (FCFS-LRH) and without reservation, with μ=4, flexibility=50%

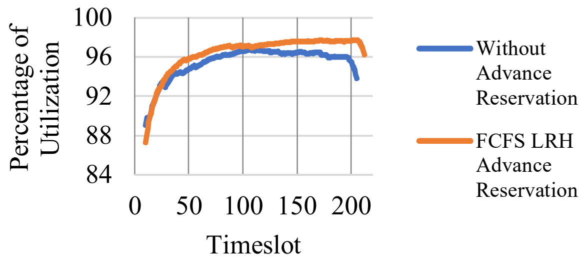


Fig. 15 Schedule of Advance Reservation (FCFS-LRH) and without reservation, with μ=4, flexibility=75%

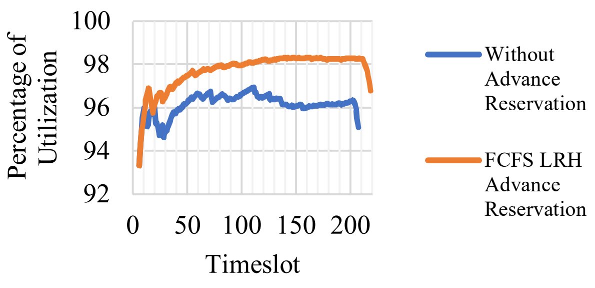


Fig. 16 Comparison of advance reservation (FCFS-LRH) and without reservation, with μ=4, flexibility=100%

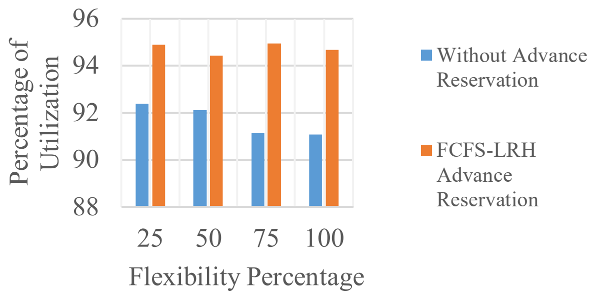


Fig. 17 Utilization based on percent flexibility with arrival factor (μ=3)

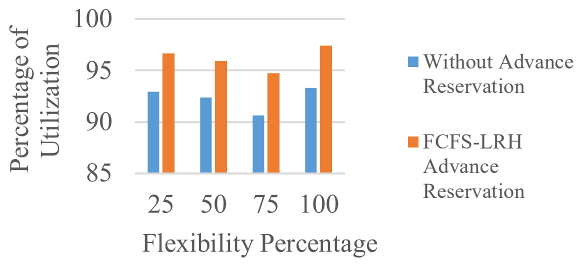


Fig. 18 Utilization based on percent flexibility with arrival factor (μ= 4)

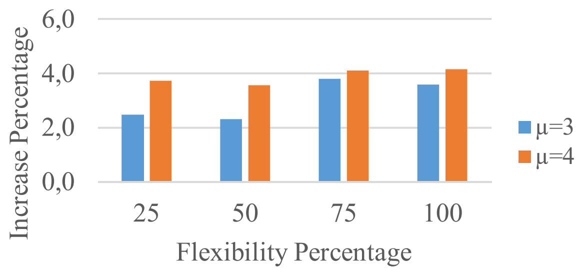


Fig.19 Percentage increase in utilization based on flexibility and arrival factors.

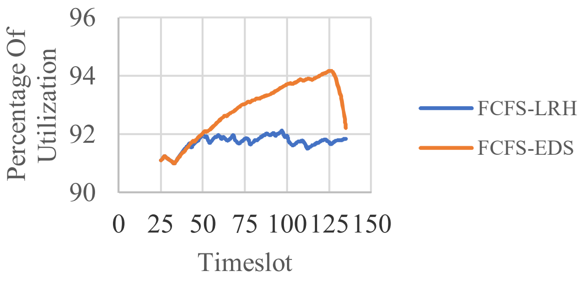


Fig. 20 Compares the utilization of FCFS-LRH with FCFS-EDS for μ=3, with 2% of jobs canceling

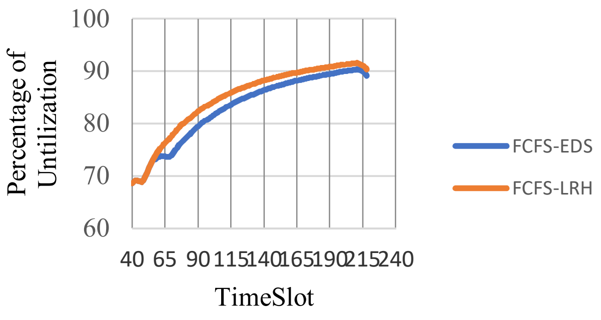


Fig. 21 Compares the utilization of FCFS-LRH with FCFS-EDS for μ=4, with 2% of jobs cancelling

IV. Conclusion

This paper proposes a reservation strategy called FCFS-LRH for MPI work. This strategy maps jobs from virtual nodes to physical nodes, ensuring that jobs allocated to virtual nodes will be executed on physical resources and obtain higher resource utilization. Increased use of resources is strongly influenced by time flexibility (tf). Experimentally the proposed method was compared with no reservation, where the results showed that the performance of the proposed method was better than the reservation strategy approach without reservation. Scheduling FCFS-LRH compared to FCFS-EDS in case of job cancellation found that resource utilization with the proposed method is better.

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, doi: 10.1002/net.20021.

[2] K. Czajkowski *et al.*, "A resource management architecture for metacomputing systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1459, pp. 62–82, 1998, doi: 10.1007/bfb0053981.

[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, doi: 10.1109/IWQOS.1999.766475.

[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," pp. 127–132, 2002, doi: 10.1109/ipdps.2000.845974.

[6] 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, doi: 10.1007/s10586-015-0530-0.

[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, doi: 10.1109/ICPADS.2007.4447738.

[8] 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, doi: 10.1109/ICYCS.2008.117.

[9] P. Xiao and Z. Hu, "Two-dimension relaxed reservation policy for independent tasks in grid computing," *J. Softw.*, vol. 6, no. 8, pp. 1395–1402, 2011, doi: 10.4304/jsw.6.8.1395-1402.

[10] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," *IEEE Int. Work. Qual. Serv. IWQoS*, vol. 2000-January, no. June, pp. 181–188, 2000, doi: 10.1109/IWQOS.2000.847954.

[11] C. Hu, "Flexible Resource Capacity Reservation Mechanism for Service Grid Using Slack Time," *J. Comput. Res. Dev.*, vol. 44, no. 1, p. 20, 2007, doi: 10.1360/crad20070103.

[12] 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," *2016 7th Int. Conf. Netw. Futur. NOF 2016*, 2017, doi: 10.1109/NOF.2016.7810118.

[13] 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, doi: 10.1016/j.comcom.2015.10.016.

[14] B. Li, Y. Pei, H. Wu, and B. Shen, "Resource availability-aware advance reservation for parallel jobs with deadlines," *J. Supercomput.*, vol. 68, no. 2, pp. 798–819, 2014, doi: 10.1007/s11227-013-1067-8.

[15] C. Castillo, G. N. Rouskas, and K. Harfoush, "Online algorithms for advance resource reservations," *J. Parallel Distrib. Comput.*, vol. 71, no. 7, pp. 963–973, 2011, doi: 10.1016/j.jpdc.2011.01.003.

[16] F. Camillo, E. Caron, R. Guivarch, A. Hurault, C. Klein, and C. Pérez, "Resource management architecture for fair scheduling of optional computations," *Proc. - 2013 8th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. 3PGCIC 2013*, pp. 113–120, 2013, doi: 10.1109/3PGCIC.2013.23.

[17] 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, doi: 10.1088/1742-6596/540/1/012002.

[18] 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, doi: 10.1007/978-3-642-30507-8_15.

[19] 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, doi: 10.1007/s10732-015-9287-0.

[20] 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, doi: 10.1109/ICACEA.2015.7164855.

[21] R. Istrate, A. Poenaru, and F. Pop, "Advance reservation system for datacenters," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2016-May, pp. 637–644, 2016, doi: 10.1109/AINA.2016.106.

[22] A. Sulistio *et al.*, "An Adaptive Scoring Job Scheduling algorithm for grid computing," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5, no. 1, pp. 68–72, 2015, doi: 10.1177/1094342006068414.

[23] O. Dakkak, S. Awang Nor, and S. Arif, "Scheduling through backfilling technique for HPC applications in grid computing environment," *ICOS 2016 - 2016 IEEE Conf. Open Syst.*, pp. 30–35, 2017, doi: 10.1109/ICOS.2016.7881984.

[24] 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.

[25] 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, 2019, doi: 10.22266/IJIES2019.0228.11.

[26] 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, doi: https://doi.org/10.1109/SAIN.2018.8673372.

[27] 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.

[28] A. Hirales-Carbajal, J.-L. González-García, and A. Tchernykh, "Workload Generation for Trace Based Grid Simulations," in *Procedding of the Ist international supercomputer conference in Mexico ISUM.*, 2010, pp. 1–9.

[29] A. Iosup, D. H. J. Epema, J. Maassen, and R. Van Nieuwpoort, "Synthetic grid workloads with Ibis, KOALA, and GRENCHMARK," in *Integrated Research in GRID Computing - CoreGRID Integration Workshop 2005, Selected Papers*, 2007, pp. 271–283, doi: 10.1007/978-0-387-47658-2_20.

[30] 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, doi: 10.1109/ICCIT.2009.319.