Planning and Scheduling Jobs on Grid Computing

Ardi Pujiyanta[1,2], Lukito Edi Nugroho[3], Widyawan[4]
[1]Electrical Engineering and Information, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2]Information Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[1]ardipujiyanta@mail.ugm.ac.id, [3]lukito@ugm.ac.id, [4]widyawan@ugm.ac.id

***Abstract*—Planning and scheduling in the Grid System allow applications to request resources from multiple scheduling systems at any given time in the future, by gaining simultaneous access to resources sufficient for their deployment. Existing scheduling strategies will deny incoming jobs if the requested resource is not available on time. Therefore, the job scheduling algorithm is one of the key areas in Grid Computing. This paper first describes the First Come First Serve Ejecting Based Dynamic Scheduling (FCFS-EDS), a job scheduling model used in Grid Computing environments, then discusses proposed job scheduling algorithms and system grid architectures as required. Finally, the proposed algorithm can perform job scheduling, as well as increase execution time for use in Grid Computing environments.**

Keywords—planning; scheduling strategy; grid system

I. INTRODUCTION

Large-scale distributed cluster computing with parallel network processing forms is referred to as Grid Computing [1]. The Grid contains resources of varying nature such as the CPU, network, data, or software [2][3]. The typical Grid mechanism is as follows [2]: The user sends the job through the Graphic User Interface, by providing some high-level specifications (e.g., the type of application to be used). The Grid plays the role of searching and allocating appropriate resources (computer, storage) to meet user demands. Then, the Grid monitors the correct job processing and informs the user that the required resources are available.

Grid resource management has several different layers from the scheduler. At the highest level is the management of global resources that have a more general view of resources, but it is very far from the source, where the application will be executed. At the lowest level is a local resource management system, which functions to manage certain resources and manage resources [4]. In the management of local resources, resources are accessed, assigned, and allocated in accordance with the Quality of Service (QoS) criteria, such as reservations and deadlines.

The work in the Grid system will be placed in a queue, waiting for available resources to become available [5], and jobs will be executed based on different parameters, such as the number of resources, and delivery and execution times. An efficient scheduling algorithm can utilize the capacity of the Grid system well, thus improving application performance [6]. First Come First Serve Ejecting Based Dynamic Scheduling (FCFS-EDS) is a strategy to increase resource utilization in the Grid, by way of user work mapped to virtual computing nodes, which are then mapped to actual computing nodes, at execution time [7]. In building the Grid system, planning and good job scheduling are required; therefore, the FCFS-EDS method will improve execution time performance, and hence scheduling performance.

II. MODEL MAPPING LOGICAL VIEW TO PHSYCAL VIEW

Job mapping is done by all job requests allocated first in a logical view (planning), then mapped to actual computing nodes (physical appearance), and job requests that have been placed on the logical view will be executed on a particular computing node for the entire time slot. To achieve this mapping, a lemma is used to ensure that the plan (logical view) can always be mapped to the actual computing node (physical appearance) [7].

If J(t) is defined as the magnitude of the value of the job planning array on MaxP (the maximum value of computing nodes) in the t slot, J(t)(i) is the i element on J(t), NJ(t) is the new array of job planning at time slot t after insertion of element J(t). The FCFS-EDS algorithm steps are as follows:

- Step1: Place the job in the logical view in matrix A, using the First Fit strategy
- Step2: Construct the permutation matrix A(t)
- Step3: Calculate the vector difference A(t) with vector A(t+1), if vector G is generated.
- Step4: Combine vector G with permutation matrix A(t), if generated complete permutation matrix H
- Step5: Transpose the complete permutation matrix H
- Step6: Multiply the vector A(t+1) with the complete transpose matrix in step5.
- Step7: Repeat step2 until step6 until finished.

III. PROPOSED SCHEDULING ALGORITHM

In this study, computing resources are only for local users (Fig. 1). Each site has its own workload that is not shared with other sites. For the simulation in this study, one external scheduler is used for one site. External Scheduler (ES): The user sends the job to an external scheduler. Then ES decides which site location to send the job to, which depends on the scheduling algorithm used. Local Scheduler (LS): When a job is placed to work on a particular site (sent to incoming work queue), it is then managed by the LS. The LS of a site determines how to schedule all jobs allocated to its resources [8].

Let G (t) be the magnitude of the array of job planning plans in MaxP (the maximum value of computing nodes) in the t slot that can be seen in (1).

$$G(t) = \{g(t)_1 , g(t)_2 , g(t)_3, \dots . , g(t)_n \} \quad (1)$$

and p is a permutation on G(t) where p shows in (2).

$$p = \begin{Bmatrix} g(t)_1 & g(t)_2 & g(t)_2 & \cdots & g(t)_n \\ p(g(t)_1) & p(g(t)_2) & p(g(t)_3) & \cdots & p(g(t)_n) \end{Bmatrix} \quad (2)$$

formed matrix $A(p) = [a_{ij}(p)]$ with (3).

$$a_{ij}(p) = \begin{cases} 1, & if\ p(g(t)_i) = g(t)_j \\ 0, & if\ p(g(t)_i) \neq g(t)_j \end{cases} \quad (3)$$

A(p) is called the permutation matrix of p, then G(t)(i) is the i element of G(t), with job id worked on the computation node i at time slot t. NG(t) is a new array of job planning on time slot t after insertion on G(t).

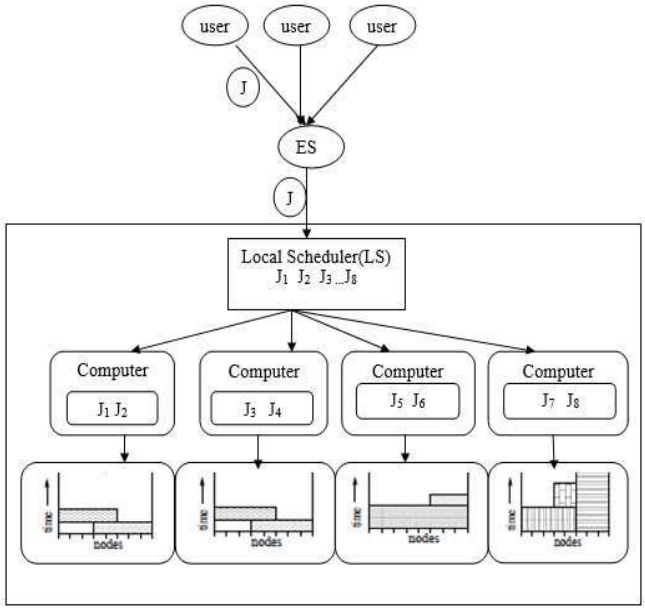

Fig. 1. Interaction between components on the grid

Thus, any finite working group G(t) can be represented by the set of permutation matrices if equation 2 is inversed that can be seen in (4).

$$p^{-1} = \begin{Bmatrix} p(g(t)_1) & p(g(t)_2) & p(g(t)_2) & \cdots & p(g(t)_n) \\ g(t)_1 & g(t)_2 & g(t)_3 & \cdots & g(t)_n \end{Bmatrix} \quad (4)$$

so, (5) is obtained.

$$a_{ij}(p) = a_{ji}(p) = a_{ij}^{-1}(p) \quad (5)$$

If given p(t) is the job permutation matrix in slot t and p-1(t+1) is the permutation inverse matrix in slot t+1, then the multiplication between p(t) and p-1(t+1) is the matrix partial identity B, where this partial identity matrix B shows that the work is done on the same computing node from slot t to slot t+1.

If (B partial identity matrix),

then the work on time slot t will be executed on the same computing node t+1

else the job planning will come at time slot t+1, multiplying the line vector G(t+1) with the complete inverse permutation matrix G-1 (t+1).

If G(t+1)(j)=G(t)(i) then

B(i,j)=1

Else

B(i,j)=0

B(i, j)=1 indicates that the job at time slot t has been executed on computing node i, and time slot t+1 has been executed on computing node j. If given LG(t) is the set of job planning with time slot t on G(t), then:

LG(t) - LG(t+1) indicates the list of job finishes at time slot t

LG(t+1) - LG(t) indicates the list of job start at time slot t+1

LG(t)∩LG(t +1) indicates list of a job continuing from time slot t until slot t+1.

Proposed algorithm steps:

- Step1: Place the job in the logical view in matrix A, using the First Fit strategy
- Step2: Construct the inverse permutation matrix $A^{-1}(t+1)$ Step3: Calculate the difference of the line vector A(t+1) with the line vector A(t), suppose G is generated.
- Step4: Construct the complete inverse permutation matrix $A^{-1}(t+1)$, suppose Y is generated.
- Step5: Multiply the vector A(t+1) with Y
- Step6: Repeat step2 until step5 to complete.

Example: If known 6 users submit job requiring 5 computer resources, then model/system will do random job fragmentation on computer resource, each slot 5 minutes wide.

Table I describes user job order placed on each logical view. The number of resources available initiates as R and user job time slot initiates as B(t).

TABLE I. PLACEMENT OF USER ORDER JOBS ON RESOURCES (LOGICAL VIEW)

	Resource				
Time Slot	***R1***	***R2***	***R3***	***R4***	***R5***
B(t)	5	2	7	1	3
B(t+1)	3	-	9	5	1

Solution:

- Inverse permutation matrix $B^{-1}(t+1)$ shown in (6).

$$B^{-1}(t+1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (6)$$

- Calculate the difference of matrix $B(t+1)$ and $B(t)$ shown in (7).

$$H = B(t+1) - B(t) \quad (7)$$

- Combine H with the inverse permutation $B^{-1}(t+1)$ shown in (8).

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (8)$$

- Multiply matrix B(t+1) with the complete permutation inverse Y shown in (9).

$$B(t+1) * \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (9)$$

In Table II it can be seen that job userid5 is done on resource R1, job userid2 is done on resource R2, job userid7 is done on resource R3, job userid9 is done on resource R3, job userid1 is done on resource R4, and job userid3 is done on resource R5.

TABLE II. RESULTS OF DATA CALCULATION TO B(T+1)

Resource					
Time Slot	***R1***	***R2***	***R3***	***R4***	***R5***
B(t)	5	2	7	1	3
B(t+1)	5	-	9	1	3

Order of job userid5 done on resource R1, userid2 job order done on resource R2, userid7 job order done on resource R3, userid9 job order done on resource R3, userid1 job order done on resource R4, and userid3 job order done on resource R5.

Permutation matrix $B(t)$ and inverse permutation matrix $B^{-1}(t+1)$ are calculated by the partial identity matrix equation shown in (10).

$$GI = B^{-1}(t+1) * B(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

and it can be said that job userid5 is done by the same computation node i.e. R1 from time slot t to t+1 as well as Userid1 done on R4 from time slot t until time slot t+1, and Userid3 is done on R5 from time slot t until time slot t+1.

IV. RESEARCH METHODS

A. Method of collecting data

To check the performance of the proposed scheduling and advance reservation scheduling strategy, this study uses data and workload generator to generate it. The output of the workload generator is used as an input to the proposed advance reservation scheduling. To generate a good workload generator, then we must determine the characteristics of the workload generator. Characteristics of the workload generator in this study[7][9][10], are as follows:

- The arrival rate of the work (time slot) that enters follows the Poisson distribution.
- Range of the execution time of each request reservations, distributed evenly.
- the earliest start time of each order, distributed evenly.
- The amount of resources required, distributed evenly.
- In this study the width of the 5 minute timeslot

Fig 2. The process model generates the advanced data workload on grid computing [10]. First, the user will submit details of the job (1). Based on the details of the user's work, as well as detailed grid information, will generate workload on the workload generator (2). The result of the workload generator will be sent back to the grid (3). The network environment is responsible for performing the work and returning the work (4). The resulting report in detail of the work, and. Finally, the user processes all the results in the post-production step (5).

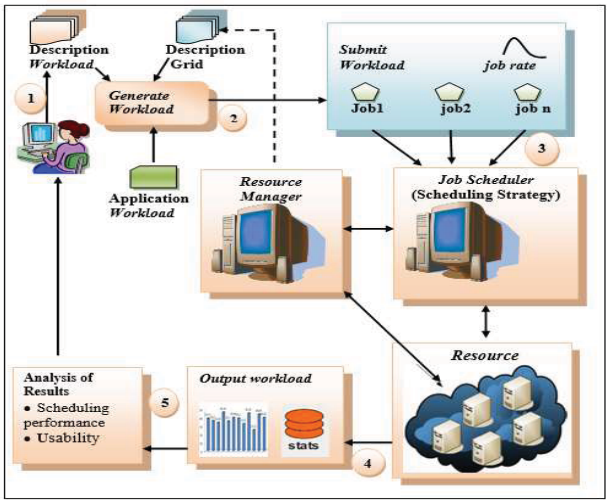

Fig. 2. The process model generates workload of advance reservation data on grid computing [9].

B. Grid System Architecture proposed

The proposed grid architecture shows the interaction between the various components in the advanced reservation scheduling model. The grid system architecture contains proposed scheduling model strategies, data structures, and accommodates reservations for serial and parallel jobs.

Interaction between the relevant components in the model shows in Fig. 3, the components involved in this model are:

Administrator, User, Module planning and reservation, Resource Allocation.

- The administrator sets the initial conditions for the scheduling model, and prepares the report.
- User submits a job description.
- The Planning Module schedules the incoming work from the user, then processes it using the proposed scheduling strategy.
- The ordering scheduling module maps the manual job to the required physical resource nodes, using the proposed system.

Explanation of interaction steps between components as below:

- Step-1 (Initialization): The administrator initializes (1) the Planning module. In the "Planning" module, the Administrator determines the proposed scheduling strategy and defines the parameters for the job submission such as the earliest start time, start time, duration of job, UserId and Jobid, the number of computation nodes required and the capacity of the computing resources.
- Step-2 (User send job): User defines parameters for job scheduling. These parameters: UserId, JobId, initial start time, start time, duration of job, and the number of computing nodes are required. User submits his / her job profile in "Planning" Module.
- Step-3 (Planning): In the "Planning" Module, the job received is processed to see if the job is acceptable based on the job description/parameters. If it is accepted, the job details are submitted and sent to the "Reservation" module and the user is "confirmed" that the job is received. In addition, the job status is "Denied" because adequate resources (compute node) are not available. The job will be removed from the list.
- Step-4 (The process of scheduling execution): In the "Reservation" module, confirmed job on STEP-3 and scheduled on the required physical resources. According to the proposed scheduling strategy, the "Reservation" Module will always find the resources for the scheduled job in logical view, and then the "Reservation" Module executes the job on the actual computing node/resource.

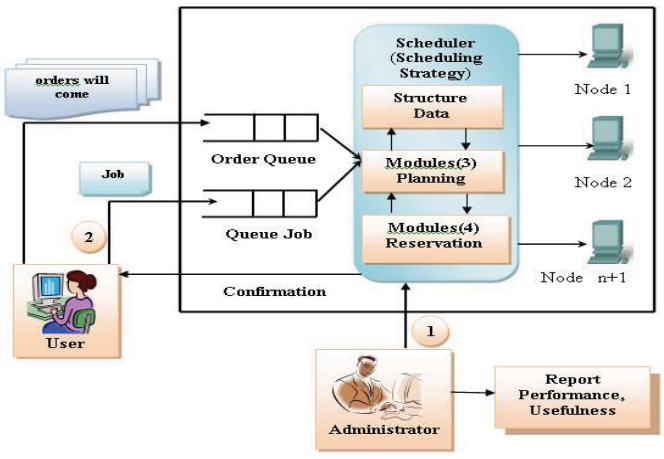

Fig. 3. Proposed resource model that supports the advance reservation on the grid.

V. RESULTS

The scheduling algorithm is one of the keys in grid computing. We compared the performance of the FCFS-EDS algorithm with the proposed algorithm, it appears that the proposed algorithm can reduce the calculation of the transpose matrix (in step 5), to prove the proposed algorithmic performance, tested using 5, 10, 15, 20 computational nodes the number of time slots used ranging from 10 to 30 time slots, with the magnitude of 1 time slot is 5 minutes. The x-axis shows the number of time slots used, the y-axis represents the execution time.

The results are shown in Fig. 4, Fig. 5, Fig. 6, and Fig. 7. When the number of computed nodes is 5 and the number of time slots used from 10 to 30-time slots (Fig. 4), the time required to execute the algorithm is faster than the FCFS-EDS algorithm. Similarly, when the number of computing nodes is changed to 10, 15 and 20 it appears in Fig. 4, Fig. 5, Fig. 6, and Fig. 7 that the proposed algorithm has a faster execution time since the proposed algorithm can reduce the transpose matrix of the FCFS-EDS algorithm, resulting in faster execution time.

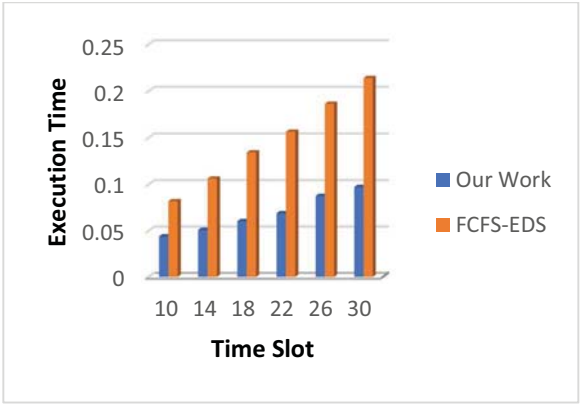

Fig. 4. Number of Computing Nodes 5

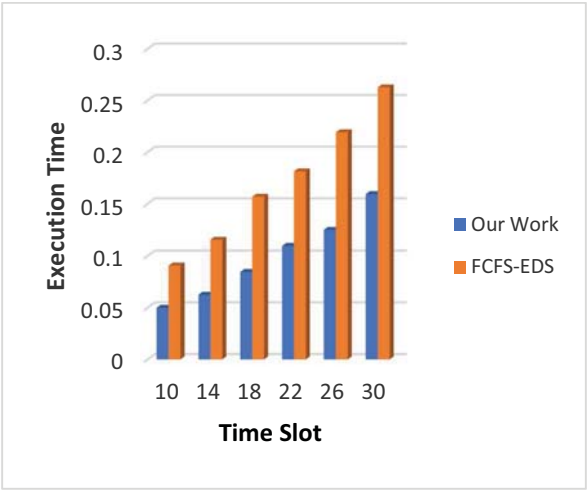

Fig. 5. Number of Computing Nodes 10

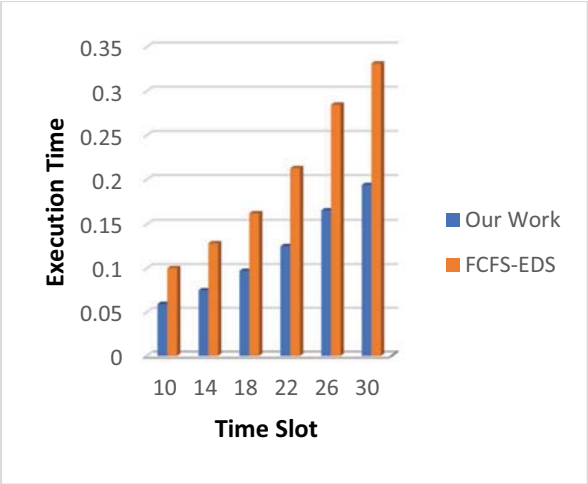

Fig. 6. Number of Computing Nodes 15

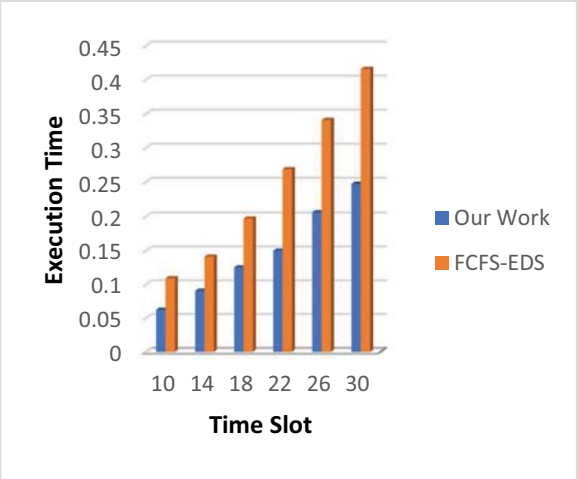

Fig. 7. Number of Computing Nodes 20

VI. Conclusion

In this research, we propose a scheduling algorithm on the local scheduler that maps logical view to physical view. From the experiment result, it can be found that the proposed algorithm can perform scheduling job on the local scheduler, and have faster execution time compared with the FCFS-EDS algorithm because the proposed algorithm can reduce the transpose matrix in the FCFS-EDS algorithm.

References

[1] C. Franke_, U. Schwiegelshohn, and R. Yahyapour, Grid scheduling by on-line rectangle packing, network an international journal,wiley, 2004.

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

[3] M. Livny and R. Raman. High-Throughput Resource Management. In I. Foster and C. Kesselman, editors, The Grid - Blueprint for a New Computing Infrastructure, pages 311–337. Morgan Kaufmann,1999.

[4] S. Uwe and Y. Ramin, "Attributes for Communication Between Grid Scheduling Instances," in Grid Resource Management: State of the Art and Future Trends, Norwell, MA, USA, Kluwer Academic Publishers, 2004.

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

[6] R. Garg and A. K. Singh, 'Adaptive workflow scheduling in grid computing based on dynamic resource availability', *Eng. Sci. Technol. an Int. J.*, vol. 18, no. 2, pp. 256–269, 2015.

[7] U. Rusydi, A. Arun and C. R. Rao, "Advance Planning and Reservation in a Grid System," in *NDT 2012. CCIS/LNCS, vol. 293, pp. 161-173. Springer, Heidelberg* , Dubai, 2012

[8] R. E. J. Munro and Y. Guo, 'Solutions for complex, multi data type and multi tool analysis: principles and applications of using workflow and pipelining methods.', *Methods Mol. Biol.*, vol. 563, pp. 259–271, 2009.

[9] Alexandru Iosup and Dick H.J. Epema, Shynthetic grid workloads with Ibis, Koala, and Grenchmark, Delft, The Netherlands.2007.

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