Designing a Switching based Workflow Scheduling Framework for Networked Environments

Hamid Tabatabaee[1]
Department of Computer Engineering,
Quchan Branch, Islamic Azad University,
Quchan, Iran

Mohamad Reza Mohebbi[2]
Department of Computer Engineering,
Ferdows Institute of higher Education,
Mashhad, Iran

Hosein Salami[3]
Department of Computer Engineering,
Ferdows Institute of higher Education,
Mashhad, Iran

***Abstract*—Due to the dynamics of the power of resources in non-dedicated computing environments such as Grid, and on the other hand, the autonomy of these environments and, consequently, the impossibility of repeating the operating scenarios to compare the algorithms created in this context, creating an environment by providing such conditions is necessary. In this paper, a framework for evaluating workflow-scheduling algorithms has been created, focusing on the dynamics of the power of resources in distributed environments. This framework based on a switching model that is capable of considering the change in the processing power of resources with high precision. Using the ability of this framework, the effectiveness of several different workflow scheduling algorithms has been evaluated.**

Keywords*—Scheduling framework; workflow scheduling; grid; switching based framework

I. Introduction

The issue of finding suitable allocation of tasks to resources, also referred to as scheduling, is one of the issues that has long been considered by researchers, and so far many studies have been done about it. These studies have often presented several solutions considering different assumptions regarding the characteristics and structure of tasks (independent or heterogeneous [1]) or resources (homogeneous or heterogeneous [2] proprietary or non-proprietary, with complete or incomplete communication, etc.) and by pursuing different goals (reducing run-time [3], minimizing cost [4] and reducing energy consumption [2]). Due to the many applications in various areas, the scheduling of the workflows is still considered a matter of interest to researchers. Workflow scheduling is a process in which resources are assigned to the tasks included in the workflow. Grid is one of the most important computational platforms for deploying and executing workflows. Workflow management systems such as Pegasus, DAGMan/Condor and Karajan/Globus manage the definition, management, and execution of workflows on computational resources [5]. These software systems perform these tasks using the low-level services provided by the middleware of the grid. In general, the workflow characteristics are generated by the user using the workflow modeling tools. These attributes define workflow activities (tasks) and the data and control dependencies between them. At runtime, the grid workflow engine controls the execution of the workflow using the middleware of the grid.

For those who are not access to these platforms, creating such platforms is expensive and time-consuming. However, even those who have access to such platforms cannot use the resources to the desired extent, and the experiment is usually limited to a small number of resources. In addition, the use of these platforms requires empirical skills in deployment, high budget and long time to get results. It is also difficult to provide specific experimental scenarios, and even in some situations it is impossible, unrepeatable and uncontrollable. To overcome such constraints, creating a simulation framework is required. There are different simulators or frameworks for supporting simulation-based studies in the grid environment. A simulation framework is needed to accurately model the behavior of the environment in order to obtain logical results. One of the obvious features of the resources in the grid environment is fluctuation of the amount of their shared computing power for processing of the tasks assigned to the Grid environment, however, in current simulators, these resource fluctuations are not considered accurately. Given this, the results provided by these simulators lack the precision about the effect of the change in the power provided by the resources on the performance of scheduling algorithms.

In this paper, a framework is developed that provides an accurate assessment of the performance of work flow scheduling algorithms in distributed platforms with non-dedicated resources that are prone to encountering irregular changes in processing power. Unlike other existing simulation tools (Section 2), this framework is based on the theoretical model [6], which is based on linear switching state space. This model is able to represent and accurately describe the process of executing tasks in a distributed platform with arbitrary computing and communication properties and structures in the desired time scales (Section 3). In this paper, by introducing the above modeling method, we will discuss how to use it to create a framework that is capable of considering the dynamics of the power of the resources during the execution of workflows. The Structure of this paper is as follows: In section 2, some of the frameworks and simulation tools in the grid and cloud computing platforms are presented. In section 3, we introduce the linear switching model and then present the proposed simulator architecture based on this model. In section

4, the experiments are carried out and the results have been discussed. Finally, in Section 5, the conclusion of this paper is presented.

II. Literature Review

There are many different frameworks and simulators with different objectives for distributed platforms, such as grid and cloud .These tools have been created with the goals such as facilitating common tasks or providing the ability to perform experiments in a controlled and repeatable environment.

In general, the frameworks are seeking to facilitate the accomplishment of some common tasks that are needed in a field. For instance, in [7], the *signac* framework is proposed to facilitate the integration of different formats of specialized data, tools, and workflows. The signac framework provides all basic components required to create a well-defined and thus collectively accessible and searchable data space, simplifying data access and modification through a homogeneous data interface that is largely agnostic to the data source, i.e., computation or experiment. In recent years, due to the spread of cyber attacks and the importance of providing strategies and solutions for cyber-security, development and deployment of intrusion detection systems has gained special attention. These systems should permanently control ongoing operations in the environment in order to identify potential attacks, thus requiring high - power computational platforms to support this volume of computation. In [8], a framework is presented which provides the appropriate distribution of these calculations with regard to the security requirements and variable availability of the computational resources(including personal and enterprise resources, as well as cloud services), and also keeping in mind to minimize the cost associated with the use of external resources (cloud services).

One of the objectives of the simulation tools is the study of architecture, components and functionality of the simulated platform. Bricks simulator [9] is a Java simulation framework used to evaluate the performance of programs and scheduling algorithms in Grid environments. The Bricks simulator includes a discrete event simulator, simulation of computing environment and grid data, as well as network components. This simulator provides an analysis and comparison of different scheduling algorithms on simulated grid settings, taking into account the impact of network components on overall efficiency. The GridSim simulator [10] is also a simulation model for Grid and Grid applications. The simulator consists of a network simulation component that is used to simulate network topologies, connections and switches, as well as resource failures in Grid applications simulation. GridSim simulator lacks the direct support to schedule workflows. The BeoSim simulator [11] is implemented with the purpose of studying the scheduling algorithms of parallel tasks in the field of multi-cluster computational Grid. This simulator can be driven by real or artificial load. Simbatch simulator [12] provides the ability to evaluate scheduling algorithms for batch schedulers. The Monarc 2 simulator [13] is a simulation framework designed to provide a design and optimization tool for large-scale distributed computing systems. Although Monarc 2 has provided two simple scheduling modules, its main purpose is to provide a realistic simulation of distributed computing systems designed to process physics data, and to propose a flexible and dynamic environment to evaluate the performance of a range of data processing architectures. The GSSIM simulator [14] has been developed based on the GridSim toolkit. The simulator provides a simple Grid scheduling framework which capable of simulating a wide range of scheduling algorithms in heterogeneous Grid infrastructures in several levels. However GSSIM faces problems such as slow execution and scalability.

None of the aforementioned simulators have direct support for workflows, so in recent years several simulators have been developed for such applications. TSM-SIM [15] is a simJava-based simulator that supports the simulation of dynamic resource grid and tasks and provides an interface for sending workflow programs as a unit. The tGSF tool [16] extends the Teikoku Grid scheduling framework, which provides a platform for simulating the scheduling of workflows and parallel tasks in the trace-based grid. Supporting workflow scheduling is loosely coupled in which work, selection, and assignment strategies are performed independently. Another example of grid scheduling simulators is WorkflowSim [17]. The focus of this simulator is on failures and on cluster-based scheduling algorithms. The WorkflowSim simulator is based on the features and services provided by CloudSim [18].

Among the simulators above, only the TSM-SIM simulator [15] explicitly considers the dynamics of resources by employing a background load generator and taking into account statistical distributions, while other simulators considered the computing power of resources at a constant value which is based on the average amount of resources processing power. In the next section, the system is firstly modeled using linear switching, and then the structure and performance of the proposed simulator are discussed.

III. The Proposed Simulator

In this section, at first, the system modeling will be introduced by the linear switching model that defines the basis for the proposed framework and then we will look at the structure and function of the proposed simulator components.

A. Modeling Task Scheduling problem in State Space

The problem of scheduling tasks in a heterogeneous resource environment has a nonlinear nature, but it can be transformed into a linear space switching model with a number of nonlinear constraints on the input vector. Due to the nature of the scheduling problem, there are complex and numerous relationships for the space state form of the system that may lead to complexity of design based on them, so the scheduling problem is represented in the form of relation (1)(At each step k).

$$X[k+1] = \max\left(0, X[k] - \Delta T * \mathrm{Diag}\left(\left[\frac{1}{\theta_{ij}}\right] * U\right)\right) \quad (1)$$

In the following sections, we define variables used in the above modeling of task scheduling problem.

1) Status Variable (X): The variable X is a vector whose number of elements is equal to the number of task in input workflow and the value of each element is a number between 0 and 1 that determines the amount of work remained for that

task in step k. The initial value of the state variable is usually equal to one. The system state variable with n task is displayed as follows:

$$X[k]=\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_{n-1} \\ x_n \end{bmatrix} \quad x_i \in [0,1] \subset R \quad ,i=1,2,\ldots,n \qquad (2)$$

For example, suppose that state variable of a system with 5 tasks in step k is as follows:

$$X[k] = \begin{bmatrix} 0 \\ 1 \\ 0.4 \\ 1 \\ 0.8 \end{bmatrix}$$

The values in the above state variable states that task 1 is fully executed, tasks 2 and 4 have not yet been executed, and 40% and 80% of tasks 3 and 5 is remained respectively.

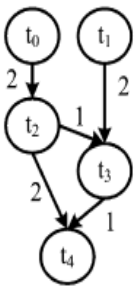
2) Control Vector(U): The control vector U is a matrix with m rows (number of resources) and n columns (number of tasks).

The element $u(i,j)$, $(i = 1,2,\ldots,m, j = 1,2,\ldots,n)$ determine the amount of use of the resource i for the task j, which is a number between zero and one. In each step, maximum number of none zero elements in each row of this matrix is one, and the remaining elements of that row are all zero. This restriction implies that at every step, each resource is only able to perform one task.

3) Runtime Length Matrix (θ): θ is a matrix with n rows and m columns, which its element θ_{ij} specifies the execution time of the task i on the resource j.(m is the number of system resources and n the number of tasks in the workflow).

4) Example of problem solved by the proposed Model: In Figure 1, a directed acyclic graph (DAG) for an application and execution time of its tasks on two resources *P0* and *P1* are provided. The aim is to schedule this application by the controller. ($\Delta t = 1$)

a



b

Task	p0	p1
t0	5	8
t1	11	15
t2	4	7
t3	6	7
t4	1	2

Fig. 1. An Example of a DAG and the Corresponding Computational Cost Table.

Figs 2 and 3 show an assignment of runtime matrix (θ) and control vector (U) to clarify how a sample schedule can be represented using linear switching state space for above DAG. In Fig. 4, the Gantt chart corresponding to this schedule is shown. As can be seen, DAG execution ends in time of 25.

U(k) Matrix

k ≤ 8		8 ≤ k ≤ 11		11 ≤ k ≤ 15		15 ≤ k ≤ 16		16 ≤ k ≤ 22	
0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0

22 ≤ k ≤ 24		24 ≤ k ≤ 25		k ≥ 26	
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	0	0	0

Fig. 2. The U Matrix Generated based on the DAG Shown in Fig. 1.

θ Matrix

	P_0	P_1
T_0	5	8
T_1	11	15
T_2	4	7
T_3	6	7
T_4	1	2

Fig. 3. The θ Matrix Generated based on the DAG Showed in Fig. 1.

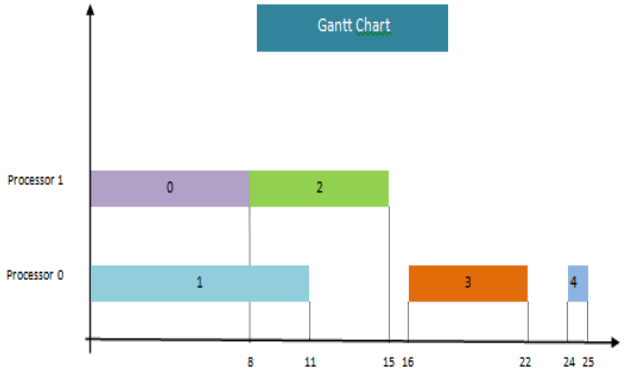


Fig. 4. Gantt Charts of Example Fig. 1.

5) Modeling the resource processing power uncertainty: Processing power of each resource determines the duration of each task's execution, therefore, uncertainty in the processing power of a resource can be shown by changing the time needed to execute tasks on that resource. Because this information is stored in the θ matrix, the change in this matrix corresponds to the change in the processing power of the resources. Therefore, the proposed model is capable of displaying uncertainty in processing power.

B. Architecture of Proposed Simulator

Due to the ability of the representation and modeling of uncertainty in the processing power of the resources available in the environment by linear switching state space model mentioned in previous section, we use it to create a tool to simulate execution of workflows in networked environments. To do this, we propose a simulator with layered loosely coupled architecture which is shown in Fig 5. In following we introduce each of the proposed simulator components.

Fig. 5. Architecture of the Proposed Simulator.

1) Event manager: The event manager is responsible for recording, storing and reporting the occurrence of internal events to other simulator components. The event manager uses SimJava [19], which provides an inter threaded interfacing framework that allows the sending of tagged events from one entity to another in a Java process. SimJava entities are connected through ports and can communicate with each other by sending and receiving objects related to tagged events.

2) Resource manager: The resource manager is almost the main component in the proposed simulator, which has several important responsibilities. This component acts as task acceptance manager by interacting with the higher layer of itself, and by accepting the ready tasks and their related mappings provides the conditions for executing them on the resources. More precisely, the resource manager, according to the amount of required processing of the tasks, and the current computing power of the resources on which each of the tasks is mapped, will determine the duration of their execution, and will notify their execution termination event to the event manager.

It is worth noting that in the proposed simulator, it is assumed that each resource can only accept a task at any time. On the other hand, this component is responsible for monitoring and tracking changes in each resource and, if any changes occur, will take appropriate action. One of these changes could be the change in resource computing power, resource failure, and resource release. Some of these events are fully managed by the resource manager, while for other events, after the necessary steps are taken, the event will be re-notified to other interested simulator components.

3) Workflow management layer: Topmost layer of the simulator is the workflow management layer which consists of three components of the workflow manager, the workflow engine and the scheduler.

The workflow manager is responsible for the interactions with below layer and the coordination of other components in this layer. The workflow engine operates similar to the workflowSim [17] workflow engine component. This component determines the ready tasks for execution, using the existing dependencies between the tasks as well as the tasks whose execution is completed, and then places them in the ready tasks repository.

The scheduler is considered as an abstract component, so it can be extended by user to schedule tasks in the desired method. For this purpose, the necessary information has been provided to the scheduler, which includes a ready- task repository, input workflow structure, specification and current status of the available resources in the environment. According to this information, scheduler should create proper mapping.

4) Adaptation of the linear switching state space model and proposed simulator: In the proposed simulator, the process of simulating a workflow is divided into several execution phases. In each execution phase, environment conditions are assumed to be constant. From environment conditions, we mean the resources computing power and the mapping of tasks to resources. A new execution phase will start with each event occurring. As mentioned before, the resource manager is the main component in the proposed simulator and a large part of the simulator function and consequently the implementation of the Linear switching state space model is implemented by it. One of the most important tasks of the resource manager is the reflection of the required changes in the model parameters at the end of each execution phase. In the following, we will focus more precisely on the functions of this component.

One of the responsibilities of the resource manager is maintaining and keeping track of the status of the tasks. According to the linear switching state space model, the execution state of a workflow is shown at any time using the status vector X, so, the resource manager in order to handle this responsibility should update the values of this vector at any time in accordance with the current status of the tasks.

Another responsibility of the resource manager is to take appropriate actions in the face of changes in resource conditions, such as computing power fluctuation and failure. In the linear switching state space model, the matrix θ shows the amount of time required to execute each task according to the current processing power of each resource. As the change in the processing power of a resource is considered to change the amount of time required to execute the tasks on the resource, therefore, in order to reflect the desired change in the switching model, the elements related to the resource in runtime matrix must be changed. Given the pursuit of changes in resource computing power by resource manager, it will also be responsible for maintaining and enforcing the necessary changes to the θ matrix.

Now, we deal with the resource failure and how it reflects in the switching model. To do this, it should be noted that the failure of a resource in addition to the change in the resource condition (θ matrix) may also cause changes in the execution status of the tasks (X vector) as well. In order to reflect the failure of a resource, the resources manager set elements corresponding to the resource in question in θ matrix to infinity value. In addition, if the resource was executing a task, the status of the task will also be changed to its original value (value 1). If the failed resource is healed, the resource manager resets the elements corresponding to the resource in θ matrix to original processing power.

The third component of the switching model is the control matrix U. As discussed in the previous section, the control matrix determines the mapping of tasks to resources. Given that the mapping of tasks on resources is assigned to the scheduler, the determination of the values of this matrix will also be carried out by the scheduler. However, the actual maintenance of this matrix is carried out by the resource manager; In fact, the scheduler creates only its intended mapping and by providing this mapping to the resource manager resource manager, the resource manager will create a control matrix.

With these components, the resource manager at any time will be able to determine the state of execution. For this purpose, as state above, the resource manager needs to apply any change in conditions in the corresponding components. The resource manager at the end of each execution phase, with respect to the stability of the environmental conditions in the execution phase (values of the matrices θ and U) and given duration of the execution phase (ΔT), calculates the processing requirements of each task, and determine the new values of the status vector (X). In fact, the resource manager at the end of each execution phase determines the status vector values (X) before the change occurs in the components of the switching model (matrix θ or U). In Fig. 6, the function and role of each of the simulator components are displayed in the process of executing a workflow. It is necessary to express that in this diagram only a part of the function of the simulation components is shown.

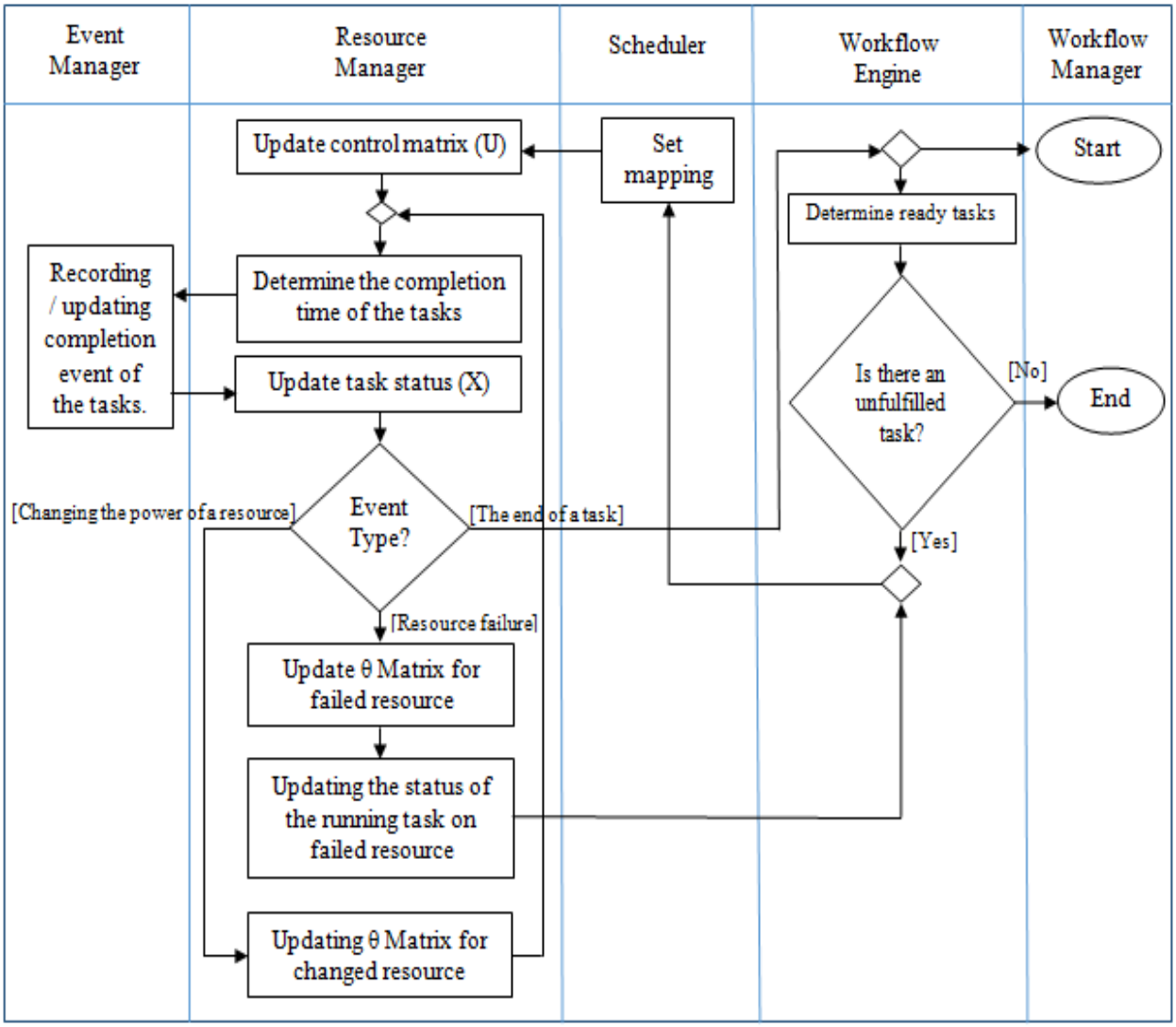


Fig. 6. The Functions of Each Simulator Component in the Process of Executing a Workflow.

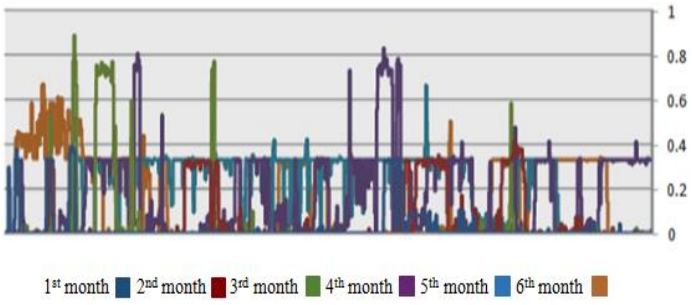


Fig. 7. The Amount of Resource No. 1515 Load at the Notre Dame University in the First Six Months of 2007.

IV. Experiments and Results

In order to evaluate the proposed simulator, five different scheduling algorithms, FCFS, MCT, MinMin, MaxMin and BNCP have been tested under different conditions of resource computing power. The NGB benchmark [20] has been used to investigate the effect of power resource fluctuations on different workflows. In the next section, a brief introduction of this benchmark are given at first, and then data set used in experiments to make changes in the conditions of the resources including computing power changes and failures are introduced. In the final section, the results of simulating the execution of this benchmark by the mentioned algorithms are presented and discussed.

A. NGB Benchmark

NGB benchmark, designed by NASA, is based on the NAS parallel benchmark [20]. These benchmarks are defined as dataflow diagrams, in which the nodes and edges respectively represent computations and communications. The NGB benchmark includes four families of problems include Embarrassingly Distributed (ED), Helical Chain (HC), Visualization Pipeline (VP) and Mixed Bags (MB)[15]. ED presents a class of grid programs called the study of parameters. HC models grid application with long chains of repeating programs, such as a set of flow computations executed in order. The VP models the workflow programs of the grid that combine multiple chains of composite processing. This benchmark, models Grid programs that the final step of their repeating tasks are visualization/analysis. MB models grid applications that combine their preprocessing tasks, calculations, and calculations of visualization, but combine with asymmetric communication.

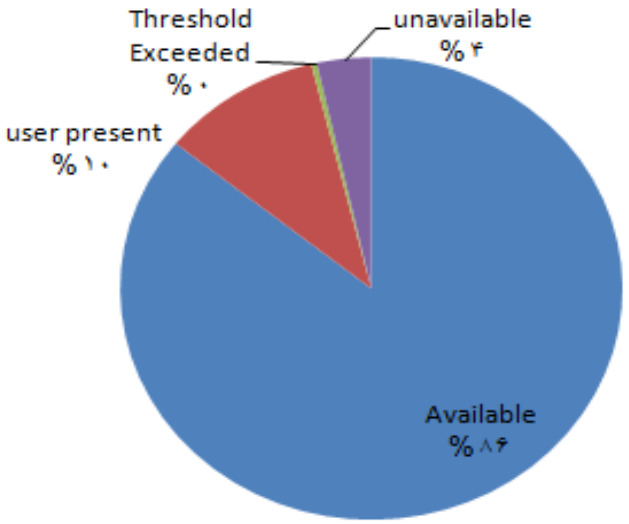
B. Used scheduling Algorithms

Four scheduling algorithms have been used to evaluate the performance of the proposed simulator: FCFC, MCT, MinMin, MaxMin and BNCP.

- **FCFC**: The initial version of the scheduling algorithm that used in this simulator. In this algorithm, each task is assigned to the next available source in the order of its entry, regardless of the expected completion time required. If multiple resources are available, one of them is selected at random.
- **MCT** (Minimum Completion Time): This algorithm [21] assigns to each task a resource with the best completion time in an arbitrary order.
- **MinMin**: This algorithm [22] initially begins with a set of ready tasks and then arranges them in the order of their completion time. In the following, a task with a minimum completion time is selected and assigned to its corresponding resource. Then, the task that has just been mapped will be sent to the queue and the process will be repeated until all ready tasks are scheduled. The idea of this algorithm is to create an optimal path to reduce the total runtime.
- **MaxMin**: Similar to the MinMin algorithm, the MaxMin algorithm selects tasks with a maximum completion time, and assigns it the best available resource. The idea of MaxMin is to avoid long-term tasks.
- **BNCP**: This algorithm [23] is a list-based algorithm which similar to other list-based algorithms composed from two steps include prioritizes tasks and assigning them to nodes. In the task prioritization phase, a simple mechanism is used in which critical tasks are selected as soon as they are ready. At the node assignment phase, in order to remove unnecessary delays that may occur due to the slowness of communication, a replication based mechanism is used, in which a resource is selected for the current assignment, which can significantly improve the execution time of the current task by replicating the critical task parent.

C. Changes in Resource Status

To simulate changes in resource status, including changes in processing power, and also failures and malfunctions, data from the resources trace of the Notre Dame University in early 2007, which is part of the FTA dataset [24] has been used. This trace includes the CPU load (percent) and the idle time (seconds). In Figure 7, the loading rate in the first 6 months of the resource number 1515 used in the experiments is shown. In Figure 8, the percentage of this resource is shown in each of the states in each month. It should be noted that in conducting experiments, the availability of a resource in any of status other than the Available status, is considered as non-Availability, in other words, failure.



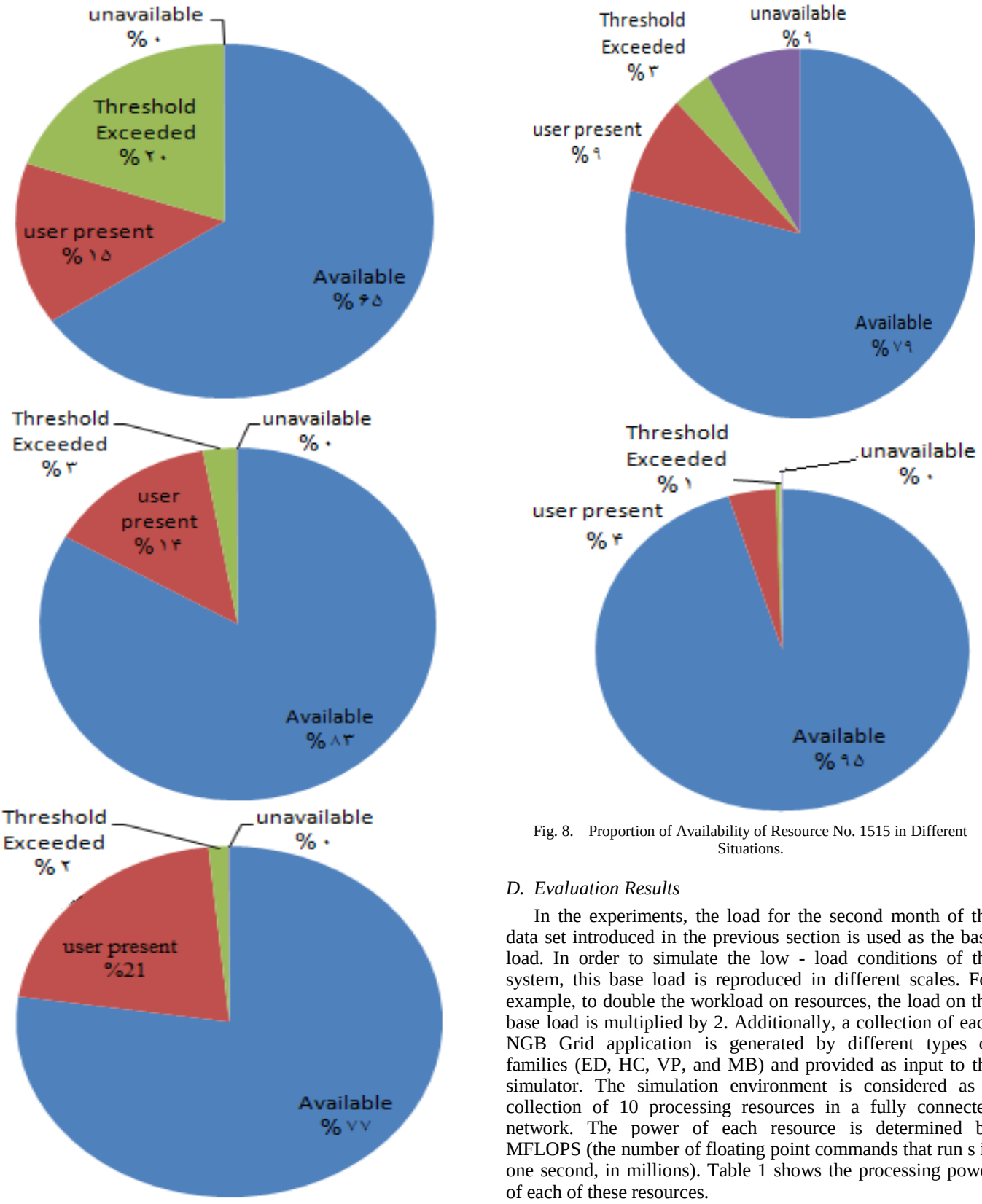


Fig. 8. Proportion of Availability of Resource No. 1515 in Different Situations.

D. Evaluation Results

In the experiments, the load for the second month of the data set introduced in the previous section is used as the base load. In order to simulate the low - load conditions of the system, this base load is reproduced in different scales. For example, to double the workload on resources, the load on the base load is multiplied by 2. Additionally, a collection of each NGB Grid application is generated by different types of families (ED, HC, VP, and MB) and provided as input to the simulator. The simulation environment is considered as a collection of 10 processing resources in a fully connected network. The power of each resource is determined by MFLOPS (the number of floating point commands that run s in one second, in millions). Table 1 shows the processing power of each of these resources.

TABLE I. THE COMPUTING POWER OF THE RESOURCES USED IN SIMULATION

resource Number	Processing Power (MFLOPS)
1	9320
2	9320
3	10400
4	6400
5	6400
6	6400
7	12000
8	9320
9	12000
10	9320

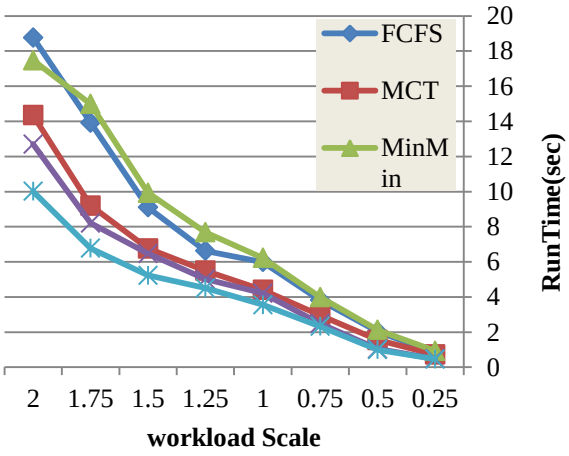


Fig. 9. The Effect of the Load Change on the VP Workflow Class.

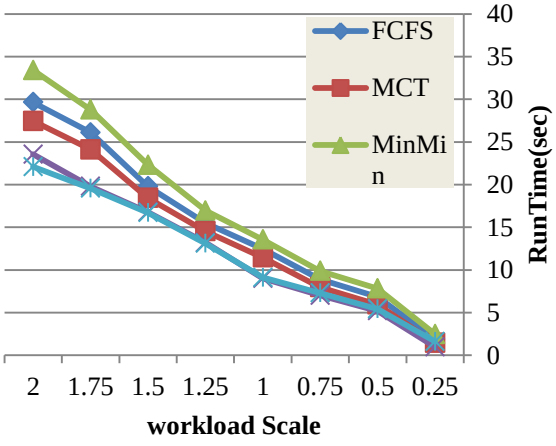


Fig. 10. The Effect of the Load Change on the MB Workflow Class.

In Figs 9 to 12, the results of the time required to complete each class of input flow types under different load conditions are shown with the five scheduling algorithms mentioned in the previous section.

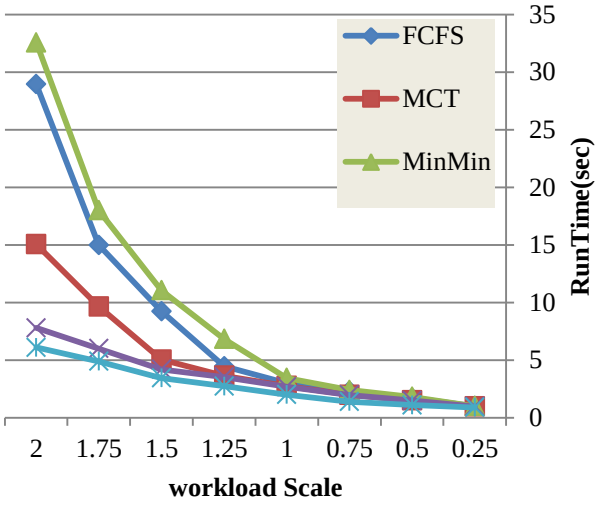


Fig. 11. The Effect of the Load Change on the ED Workflow Class.

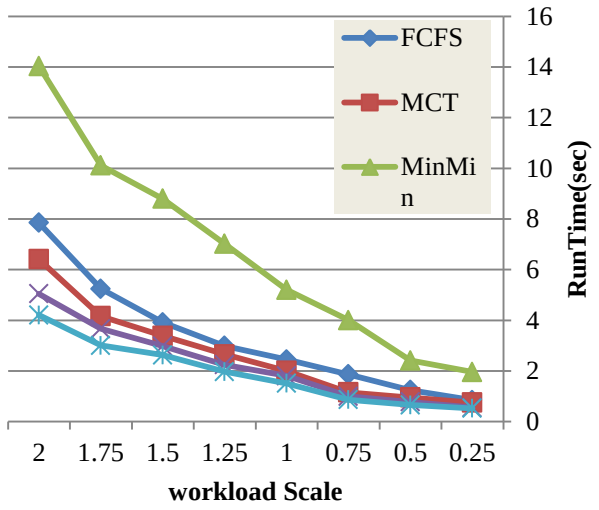


Fig. 12. The Effect of the Load Change on the HC Workflow Class.

Regarding Figures 9 through 12, as expected, increasing the amount of load in the resources will increase the time needed to execute the workflow. However, among the scheduling algorithms used, the BNCP and MaxMin algorithms exhibit the best performance and MinMin's worst performance. According to the obtained results, between different types of workflow, HC and ED classes have demonstrated higher sensitivity to increasing the load level of their resources so that by increasing the load, the execution period of the work streams is substantially increased. These results are similar to those reported in [15].

V. Conclusion and Future Work

In this paper, a simulator for evaluating workflows based on linear switching model is presented. The linear switching model used is capable of accurate representation of variation in computational resources processing power, so the simulator has been able to accurately assess the performance of workflow scheduling algorithms in the face of variation in the power of resources. By using this simulator, five workflow scheduling algorithms were evaluated to execute workflows created on the basis of the NGB benchmark. Due to the high rate of failures in distributed environments, the issue of dealing with them has been an important place in the recent researches, and scheduling algorithms are introduced with the approach of dealing with the failures. As the future work, we can also add the possibility of supporting this kind of algorithms, which are for instance the check-pointing algorithms.

Acknowledgement

We are grateful to Islamic Azad University, Quchan branch authorities, for their useful collaboration.

References

[1] Q.Y. Chen, et al. “Research of Dependent Tasks Scheduling Algorithm in Cloud Computing Environments,” InITM Web of Conferences 2016 (Vol. 7, p. 08001). EDP Sciences.

[2] Nesmachnow S, et al. “Energy-aware scheduling on multicore heterogeneous grid computing systems,” Journal of Grid Computing. 2013 Dec 1;11(4):653-80.

[3] Dolkhani E, et al. “Assignment look-ahead HEFT for scheduling workflows of communicating tasks,” InTelecommunications (IST), 2016 8th International Symposium on 2016 Sep 27 (pp. 649-653). IEEE.

[4] S. Selvarani, G.S. Sadhasivam. “ Improved cost-based algorithm for task scheduling in cloud computing,” InComputational intelligence and computing research (iccic), 2010 ieee international conference on 2010 Dec 28 (pp. 1-5). IEEE.

[5] J. Yu and R. Buyya, “A taxonomy of workflow management systems for grid computing,” Journal of Grid Computing, vol.3, no. 3-4, pp. 171-200, 2005.

[6] H.tabatabaee, "(static and Dynamic) Task scheduling Modeling by linear switching state space ", PhD thesis, 2011.

[7] Carl S. Adorf, et al., “Simple data and workflow management with the signac framework”, Computational Materials Science, 220–229, 2018.

[8] José Francisco Colom, et al., “Scheduling framework for distributed intrusion detection systems over heterogeneous network architectures”, Network and Computer Applications, S1084-8045(18)30041-9, 2018.

[9] A. Takefusa, et al. “Overview of a performance evaluation system for global computing scheduling algorithms,” In High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on, pp. 97-104. IEEE, 1999.

[10] A. Sulistio, et al., “A toolkit for modelling and simulating data Grids: an extension to GridSim,” Concurrency and Computation: Practice and Experience, vol. 20, no. 13, pp. 1591-1609, 2008.

[11] W. M. Jones, et al., “Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters,” The Journal of Supercomputing, vol. 34, no. 2, pp. 135-163, 2005.

[12] Y. Caniou and J.-S. Gay. “Simbatch: An API for simulating and predicting the performance of parallel resources managed by batch systems,” in European Conference on Parallel Processing. 2008. Springer.

[13] C. Dobre, F. Pop, and V. Cristea. “A simulation framework for dependable distributed systems,” in 2008 International Conference on Parallel Processing-Workshops. 2008. IEEE.

[14] K. Kurowski, et al. “Grid scheduling simulations with GSSIM. in Parallel and Distributed Systems,” 2007 International Conference on. 2007. IEEE.

[15] M.A. Belkoura and N. Lopez-Benitez, “TSM-SIM: A Two-Stage Grid Metascheduler Simulator,” International Journal of Grid Computing & Applications, vol. 2, no. 4, pp. 11, 2011.

[16] A. Hirales-Carbajal, et al. “A grid simulation framework to study advance scheduling strategies for complex workflow applications,” in Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. 2010. IEEE.

[17] W. Chen and E. Deelman. “Workflowsim: A toolkit for simulating scientific workflows in distributed environments,” in E-Science (e-Science), 2012 IEEE 8th International Conference on. 2012. IEEE.

[18] R.N. Calheiros, et al., “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.

[19] F. Howell and R. McNab, “SimJava: A discrete event simulation library for java,” Simulation Series, 1998. 30: p. 51-56.

[20] M. Frumkin and R.F. Van der Wijngaart, “Nas grid benchmarks: A tool for grid space exploration,” Cluster Computing, vol. 5, no. 3, pp. 247-255, 2002.

[21] T.D. Braun, et al., “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems,” Journal of Parallel and Distributed computing, vol. 61, no.6, pp. 810-837, 2001.

[22] J. Blythe, et al. “Task scheduling strategies for workflow-based applications in grids,” in CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005. 2005. IEEE.

[23] A. Atef, et al. “Lower-bound complexity algorithm for task scheduling on heterogeneous grid,” Computing. 2017:1-21.

[24] B. Javadi, et al. “The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems,” Journal of Parallel and Distributed Computing, vol. 73, no. 8, pp. 1208-1223, 2013.