CHAPTER 4

GENERIC APPLICATION EVALUATION

In this chapter, the evaluation of the proposed algorithm by simulation is presented. The chapter is divided into 4 parts which are simulation tool, workflow scheduling algorithm simulation, experiment setting, followed by results and discussion in the last section.

4.1 Core Simulation Tool

The detailed evaluation of the workflow scheduling algorithm was performed on a modified version of a simulation system called **SimGrid** [16]. SimGrid is a toolkit that provides core functionalities for the simulation of distributed applications in heterogeneous distributed environments. The specific goal of SimGrid is to facilitate research in the area of distributed and parallel application scheduling on distributed computing platforms ranging from simple network of workstations to Computational Grids. The SimGrid software architecture comprises four main components: SURF, MSG, GRAS, and SMPI. The last three components provide APIs for implementing, simulating and/or deploying distributed applications. The first component, SURF, is a fast and accurate simulation engine.

As depicted by the following diagram, the SimGrid toolkit is basically three-layer which are described below.

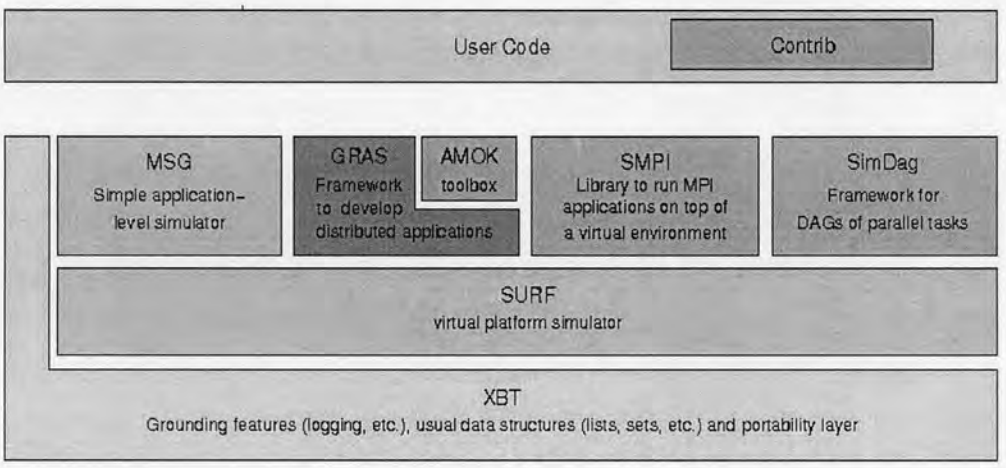


Figure 4-1 Relationships between the SimGrid components

4.1.1 Programmation environments layer

SimGrid provides several programming environments built on top of a unique simulation kernel. Each environment targets a specific audience and constitutes a different paradigm.

- **MSG** (Meta SimGrid: Rapid Distributed Algorithms Prototyping) is used to study a theoretical problem and compare several heuristics. It can be easily build rather than realistic multi-agents simulation. The MSG module provides an API for the easy prototyping of distributed applications by letting users focus solely on algorithmic issues. Simulations are constructed in terms of concurrent processes, which can be created, suspended, resumed and terminated dynamically, and can synchronize by exchanging data. A major difference between MSG and the next two modules, GRAS and SMPI, is that all simulated processes are in the same address space. This simplifies development of the simulation by allowing convenient communications via global data structures. In other words, while MSG can accurately simulate the interactions taking place in a distributed application, including communication and synchronization delays, the simulated application can be implemented with the convenience of a single address space.
- **SMPI** (Simulated Message Passing Interface: MPI Applications and Heterogeneous Platforms) programming environment, is used to study the behavior of a MPI application using emulation techniques. The Message Passing Interface (MPI) is the standard programming interface for parallel computational science applications. Many researchers and practitioners are interested in studying the behavior of MPI applications in various heterogeneous environments (both networks and CPUs), which they do not have at their disposal. SMPI is meant to enable such studies. Using the same benchmarking techniques as the ones developed in GRAS, users can annotate an existing application to indicate which portion of the application should be simulated, Application execution can then be simulated on arbitrary heterogeneous platforms.
- **GRAS** (Grid Reality And Simulation: Development of Production Distributed Applications) is a complete environment to develop a real-world distributed

application. This is an API for the realization of distributed applications. A major added functionality of GRAS, when compared to MSG, is that applications built with GRAS can run in simulated mode but also in the real world. Furthermore, this does not require any code modification. Key portions of the application code are automatically benchmarked so that the application simulation can be instantiated *on the fly*. As a result, GRAS provides a full-fledge environment to develop and deploy production distributed applications with the comfort of simulation for testing, debugging, and evaluation. GRAS provides a high-performance communication layer that allows easy and efficient cross-architecture communication of complex data structures. GRAS is portable and is available for Linux, Mac OSX, Solaris, AIX, IRIX and was validated on twelve CPU architectures.

4.1.2 Simulation kernel layer

The core functionalities to simulate a virtual platform are provided by a module called SURF. It is very low-level and is not intended to be used as such by end-users. Instead, it serves as a basis for the higher level layer.

- **SURF** (Simulation Kernel) main features are a fast max-min linear solver and the ability to change transparently the model used to describe the platform. This greatly eases the comparison of the several models existing in the literature. SURF considers the platform as a set of resources, with each of the simulated concurrent tasks utilizes some subset of these resources. SURF computes the fraction of power that each resource delivers to each task that uses it by solving a constrained maximization problem: allocate as much power as possible to all tasks in a way that *maximizes* the *minimum* power allocation over all tasks, also called *Max-Min* fairness. SURF provides a fast implementation of *Max-Min* fairness. Via *Max-Min* fairness, SURF enables fast and realistic simulation of resource sharing (e.g., TCP flows over multi-link LAN and WAN paths, processes on a CPU). Furthermore, it enables trace-based simulation of dynamic resource availability. Finally, simulated platform configurations are described in XML using a syntax that provides a unified abstract basis for all other components of the SimGrid project (MSG, GRAS, and SMPI).

4.1.3 Base layer

The base of the whole toolkit is constituted by the XBT (eXtended Bundle of Tools). It is a portable library providing some grounding features such as Logging support, Exception support and Configuration support. XBT also encompass the following convenient data structures: Dynar: generic dynamic array, FIFO: generic work queue, Dict: generic dictionary, Heap: generic heap data structure, Set: generic set datatype and Swag: O(1) set datatype.

4.2 Workflow Scheduling Algorithms Simulation

In order to simulate the proposed scheduling algorithm on SimGrid, a substantial enhancement was done. The simulation system consists of 4 main processes which are configuration process, simulation process, iteration process, and analytical process as shown in Figure 4-2.

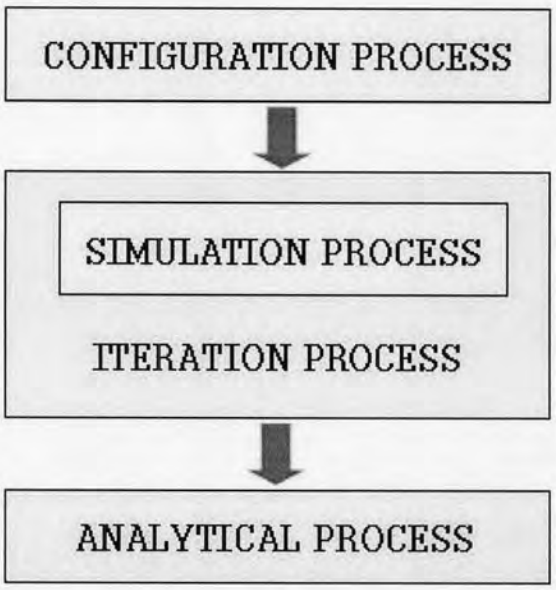


Figure 4-2 Simulation system

First, each parameter in the configuration file is set to initiate variables. After that, the simulation process starts and produces simulation result to be recorded. The simulation process is repeated until it reaches the number of iteration as set in the configuration file. Finally, due to the abundant number of result, an analytical process is needed to compute the average.

In order to study many cases, the configuration file has to be edited and then perform the procedure explained above. The detail of each process will be described elaborately as follows.

4.2.1 Configuration process

The configuration process is the first process to be started. In this process, each parameter has to be set according to the objective of the study. There are 3 types of parameters have been used in this simulator which are numerical parameter, path parameter, and character parameter. A numerical parameter is an unspecified quantity or even a selectable mode such as the number of workflow, the number of task, and type of scheduling. Path parameter is used to locate the file to be read such as task dependency file path, intermediate ranking file path, and final ranking file path. Character parameter is a member of selected character such as project name, and file name.

In this simulator, there are several parameters that can be set such as the number of workflow, task dependency file path, and project name etc. Each parameter is list as express below.

4.2.1.1 Workflow Configuration

- *The number of workflow*

The number of workflows to be simulated can be adjusted as an integer number from 1 up to 30,000. The number of workflows should be sufficiently large since it leads to a reliable result. However, the abundant number of workflow can take much resources. Therefore, it has to be adjusted to a proper value in order to achieve the greatest capability.

- *Workflow initial priority*

Workflow initial priority is a parameter to identify the initial priority for each workflow. The high priority can be specified by a small value. The priority value is set for all workflow that will be submitted in the before-evaluation-queue while it will not effect any workflow priority in the after-evaluation-queue.

- *Workflow computation size*

Workflow computation size is a computation size (in flops) of each workflow to be executed in the simulation system. The computation size can be calculated as a

multiplicity between workflow computation time needed (in seconds) and the speed of processor (in flops/second). In order to generate computation size for each workflow, an equation called "Random Gausian Distribution" is used to generate data by identifying the median data, and standard deviation. For the configuration, four parameters have to be specified in order to generate and set the computation size for each workflow.

a) Median computation size is the median value of workflow computation size.

b) Standard deviation is a statistic value to show how tightly all various data are clustered around the mean. If data points are close to the mean, then the standard deviation is small. In the case that need the same computation size for all workflows. The mean has to be set as the computation size needed, and the standard deviation has to be set to zero.

c) Percent error is an acceptable error of standard deviation from data generating process. The standard deviation estimated from all generated data might not be equal to the standard deviation that has been specified, so the acceptable percent error has to be set. The acceptable standard deviation can then be computed as shown in Equation 5-1

$$\sigma_{acceptable} = \sigma_{set} - \left(\frac{\%error \times \sigma_{set}}{100} \right) \qquad \text{(Eq 4-1)}$$

d) Computation size pattern is a pattern to set the computation size for each workflow which has different computation size. The computation size pattern is not necessary in the case when all workflow have the same computation size (Standard deviation = 0). There are seven types of pattern for defining workflow computation size which are

- forward order of intermediate ranking
- reverse order of intermediate ranking
- forward order of final ranking
- reverse order of final ranking
- forward order of workflow submission

- reverse order of workflow submission

- random

- *Workflow communication size*

Workflow communication size is data size (in bytes) to be transferred for each workflow. Since data transfer is not considered in this research, the workflow communication size is fixed to be equal to the number of task (communication size of each task = 1). However, in other cases, the communication size for each workflow can be generated by using the same procedure as the computation size which has been described in the previous section.

4.2.1.2 Task Configuration

- *The number of tasks*

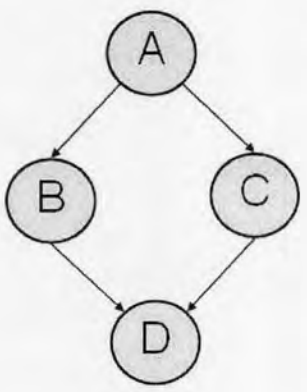
Generally, a workflow consists of several tasks. Since this research focuses on only parameter sweep applications, each workflow will therefore has the same number of tasks.

- *Task dependency description*

Each workflow composes of several related tasks. The relationship between tasks can be defined in the term of parent and child. This kind of description is also used in DagMan as described in Chapter 2. For the configuration, the number of relationship among tasks and their dependency have to be defined. In this research, XML is used as a file format of task dependency description as shown in Figure 4-3.

```
<task_dependency_id="1" parent="A" child="B"/>
<task_dependency_id="2" parent="A" child="C"/>
<task_dependency_id="3" parent="B" child="D"/>
<task_dependency_id="4" parent="C" child="D"/>
```

(a)



(b)

Figure 4-3 (a) Task dependency description (b) Dependency graph

In order to define the task dependency, three parameters have to be specified which are the task dependency number, parent task, and child task. For example, task dependency shown in Figure 4-3(a) can be represented as task dependency graph in Figure 4-3(b).

- *Task computation size*

Task computation size is the computation size to execute each task. Tasks in each workflow can be categorized into 2 groups which are before-evaluation tasks, and after-evaluation tasks. Before-evaluation tasks is a group of tasks that is executed before the evaluation point while after-evaluation tasks is a group of tasks that is executed after the evaluation point.

The procedure of assigning computation size are described as shown in Figure 4-4. After the workflow computation size was set, it is firstly divided into 2 parts in order to provide one part for a task at the evaluation point which will be described in the next section. Another part of the computation size will be assigned for both before-evaluation tasks and after-evaluation tasks. For the configuration, the computation size for each group of tasks has to be specified as percent of total computation size for both groups.

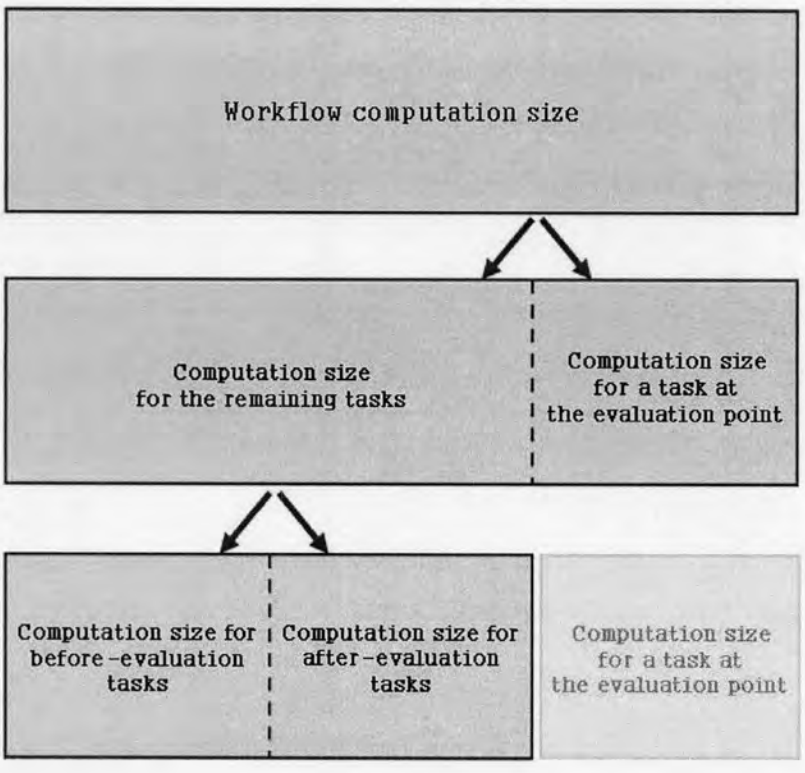


Figure 4-4 Procedure of assigning workflow computation size

- *Task communication size*

Task communication size is data size (in bytes) to be transfered for each task. Since data transfer is not considered in this research, the task communication size is fixed to be 1 byte. However, in other cases, task communication size can be evaluated by using the same procedure as the task computation size which has been described in the previous section.

4.2.1.3 Evaluation Point Configuration

- *The number of evaluation points*

This research considers only the case that has one evaluation point so that this parameter can not be adjusted.

- *Evaluation point*

Evaluation point is a reference point in which the intermediate result is produced. The evaluation point is set in term of task number, so that it can not be set higher than the number of tasks. It can be said that, each task in the workflow is executed until the task that has been set as an evaluation point is completed. Then, the workflow state is changed from before-evaluation-queue to after-evaluation-queue.

- *Computation size of evaluation point*

Computation size of evaluation point is a computation size (in flops) for executing the task at evaluation point. There are 4 types of setting to specify the computation size of evaluation point which are

- specify as percent of workflow computation size
- manually specify the computation size value (in flops). Note that, it has to be set less than the workflow computation size.
- specify equal to the computation size of each task in before-evaluation-queue
- specify equal to the computation size of each task in after-evaluation-queue

- *Communication size of evaluation point*

Communication size of evaluation point is a data size (in bytes) to be transferred for a task at the evaluation point. Since data transfer is not considered in this research, the communication size of evaluation point is fixed to be 1 byte.

However, in other cases, the communication size of evaluation point can be specified by using the same setting as computation size of evaluation point which has been described in the previous section.

4.2.1.4 Host Configuration

- *The number of hosts*

The number of hosts is the number of compute nodes that are simulated for executing tasks in workflows.

- *Host platform description*

Host platform description is used for defining the properties of each host such as host name, the speed of processor, bandwidth, latency, and network routing.

Since the platform generating process can be much complicated, two programs are used in order to generate host platform which are Tiers topology generator, and Tiers SimGrid. First, the number of host has to be set in the Tiers topology generator. The network connection is then generated as a graph which will be set as an input of another program, Tiers SimGrid. This program is used for generating host platform description by converting a graph format to XML format in order to be used in this simulation.

- *Host requirement description*

Host deployment description is used for selecting the functionality of each host between master, or slave. There is only one host that can be set as a master node which is the centralized scheduler for scheduling tasks to compute node. If a host is defined as a slave, it will be set as a compute node for executing tasks received from the scheduler in the master node. In this simulation, the host connections communicate on a LAN (Local Area Network).

4.2.1.5 Scheduling type

There are many possible approaches to scheduling work in order to accomplish various objectives. The more effective application can be enhanced by using the proper scheduling type. In this research, five different types of scheduling are compared which are First In First Out, Before-Evaluation-Queue-First, Weight Queue, Random, and Best-Intermediate-Result-First as described below.

a) First In First Out (FIFO)

The FIFO scheduling is a conventional approach to handling work requests. In this technique, the oldest submitted workflow in scheduling queue has to be executed first. Hence, the first submitted workflow is the first to be removed from the queue while the last submitted workflow is finally removed at the end of the scheduling process.

b) Before-Evaluation-Queue-First

In this scheduling policy, each workflow has to be executed in the before-evaluation-queue until it reaches the evaluation point. Then, it will be moved into the after-evaluation-queue and waits for other remaining queue to be executed. After all workflow are moved to the after-evaluation-queue, they are compared to each other in order to find intermediate ranking and set the priority. Since all workflow has been evaluated and ranked before the execution in the after-evaluation-queue can start, workflow in the after-evaluation-queue will therefore be executed in the same order as intermediate ranking.

c) Weight Queue

In this scheduling approach, the number of workflow in each queue has to be considered in order to select the queue to be processed. The queue which has more workflow is selected and a workflow in that queue will be executed.

d) Random

For the random scheduling policy, there is no pattern for scheduling a queue to be processed. Workflow in both before-evaluation-queue and after-evaluation-queue always have a chance to be processed.

4.2.1.6 Correlation Configuration

In this research, intermediate results and final results can be generated by specifying the correlation between intermediate results and final results, standard deviation of intermediate results, and standard deviation of final results into the statistic equation called "Random bivariate gaussian". The configuration of each parameter is described below.

- *Correlation between intermediate results and final results*

  Correlation is a value to represent the relationship between data, which in this research are intermediate results and final results. The correlation can be set from 0 to 1. If it is set close to 1, the intermediate results and final results will be generated close to each other. On the other hand, if it is set close to 0, the value of intermediate results and final results will be much different. However, in this research, good correlation should be close to 1.

- *Standard deviations of intermediate results and final results*

  Standard deviation is a statistic value to show how tightly all various data are clustered around the mean. If it is set to high value, the generated data will be much separated. In contrast, if it is set to a small value, the generated data will be close to each other. Therefore, the same value for all data can be created by setting the standard deviation equal to 0. In this research, two standard deviations have to be specified which are standard deviation of intermediate results and standard deviation of final results in order to generate intermediate results and final results respectively. However, in this research, both parameters are set equal to 1 throughout the experiment.

4.2.2 Simulation process

The Workflow Scheduler with Best-Intermediate-Result First Policy Simulation is implemented on the core simulation system called SimGrid as described in the first part of this chapter. In this research, the MSG module is used since it can provide simple and efficient APIs.

The simulation process is started by parsing the configuration file in order to create required components such as workflows, tasks, hosts, and scheduling queues. In this process, there are 3 queues which are before-evaluation-queue, after-evaluation-queue, and scheduling queue. The procedure of simulation process is described below which can be illustrated as shown in Figure 4-5.

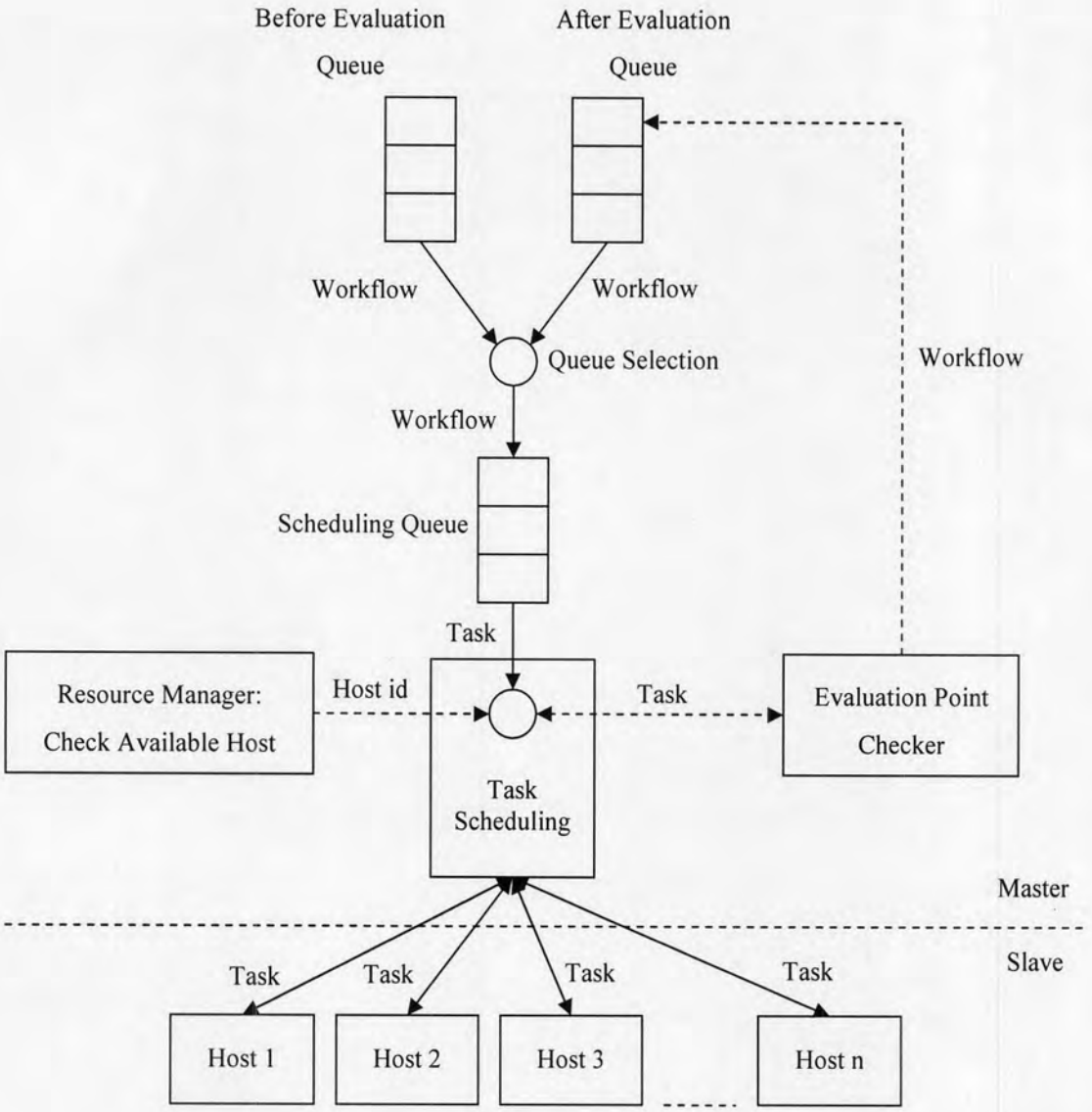


Figure 4-5 Simulation process

The before-evaluation-queue is firstly filled with all created workflows arranged in an arbitrary order that has been generated in the configuration process. At first, workflows in before-evaluation-queue are moved to fulfill the scheduling queue. In this queue, each workflow is viewed as several tasks. If there is an available host, a task in this queue will be selected and moved to be executed in the corresponding host. Here, a task will be assigned to each host until there is no host available. However, in order to select a task to be executed, it has to meet task dependency condition. When a task is completely executed, the host status will be back to

available for the next task to be executed. The task scheduling is continuously processed. If there is a workflow that reaches the evaluation point, it will be moved to the after-evaluation-queue. This will therefore provide a space in the scheduling queue. Then, queue selection will choose a queue between before-evaluation-queue and after-evaluation-queue depending on the scheduling type. A workflow in the selected queue is moved to replace that space in the scheduling queue. Moreover, when a workflow is completely executed, there will also be a space in the scheduling queue for the next workflow to be replaced. These simulation procedures are performed until all workflows are completely executed.

In the simulation process, the main procedure is to select a queue between before-evaluation-queue and after-evaluation queue in order to move a workflow in that queue to a workflow scheduling queue. However, a workflow will be moved only when there is at least a space for a workflow to move into the workflow scheduling queue. The selection pattern can be arbitrary set according to the scheduling type that has been specified in the configuration process.

As mentioned above, there are 2 queues namely before-evaluation-queue and after-evaluation-queue which are first-in-first-out queue and priority queue respectively. Figure 4-6 shows input and output of each queue. The input of before-evaluation-queue is a set of workflows that have been submitted in the arbitrary order at the beginning of simulation. The output of before-evaluation-queue is a workflow that was firstly inserted into the queue. This output is then delivered to the next layer (Workflow scheduling). For the after-evaluation-queue, the input is a workflow that reaches the evaluation point while the output is a workflow that has the highest priority at that time.

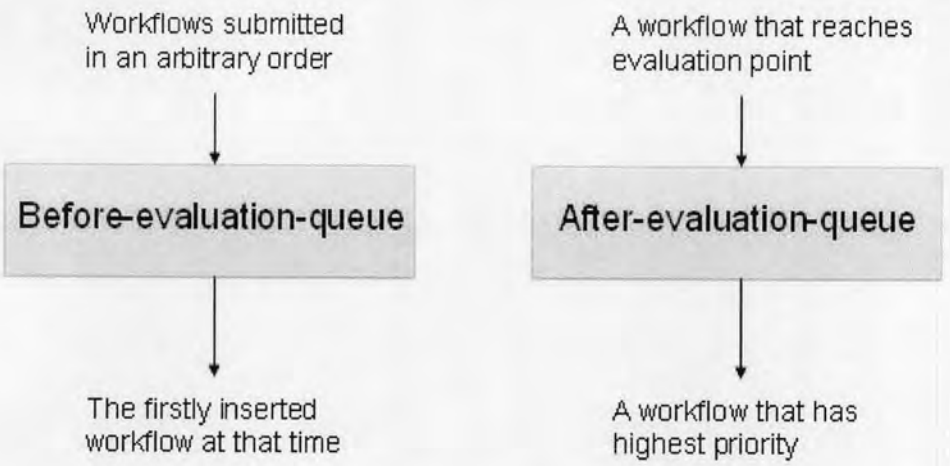


Figure 4-6 Input and output of before-evaluation-queue and after-evaluation-queue

Queue selection is a very important part in this research since it can effect a chance that high rank workflow will be selected to executed earlier. The basic idea of queue selection is a comparison between probability of before-evaluation-queue and probability of after-evaluation-queue. The queue which has higher probability will be selected. However, if both queues have the same probability, the before-evaluation-queue will be selected. Moreover, if there is no workflow in which queue, another queue will be selected. These conditions can be described as shown in

$$Selected\ queue = \begin{cases} Before - Evaluation - Queue; P_1 \geq P_2 \\ After - Evaluation - Queue; P_1 < P_2 \end{cases} \quad \text{(Eq 4-2)}$$

When P_1 = Probability of before-evaluation-queue

P_2 = Probability of after-evaluation-queue

This selection pattern can be arbitrary set according to the scheduling type that has been specified in the configuration process. In this research, there are 5 types of scheduling which are First-in-first-out, Best-evaluation-queue-first, Weight queue, Random, and Best-intermediate-result-first as described in the previous part (Configuration process).

The thing that differentiates queue selection pattern of each scheduling type from others is the probability setting. The probability of before-evaluation-queue and after-evaluation-queue for each scheduling type are listed as shown in Table 4-1.

Scheduling type	P_1	P_2
First-In-First-Out	0 *if* $N_2 > 0$ 1 *if* $N_2 = 0$	1 *if* $N_2 > 0$ 0 *if* $N_2 = 0$
Before-Evaluation-Queue-First	1 *if* $N_1 > 0$ 0 *if* $N_1 = 0$	0 *if* $N_1 > 0$ 1 *if* $N_1 = 0$
Weight queue	1 *if* $N_1 \geq N_2$ 0 *if* $N_1 < N_2$	0 *if* $N_1 \geq N_2$ 1 *if* $N_1 < N_2$
Random	0.5	0.5
Best-Intermediate-Result-First	$\frac{N_1}{(N_1 + N_2)}$	$\frac{N_2}{(N_1 + N_2)}$

Table 4-1 Probability of queue selection for each scheduling type.

When P_1 = Probability of before-evaluation-queue

P_2 = Probability of after-evaluation-queue

N_1 = The number of workflow in before-evaluation-queue

N_2 = The number of workflow in after-evaluation-queue

After the simulation process is completed, simulation results are collected which are completion time of each workflow, and queue selection history as shown in Table 4-2 below.

Workflow ID	*1*	*2*	*3*	*4*	.....	*N*
Completion time (hours)	30.451	48.136	41.258	25.541	.....	39.297

(a)

Scheduling #	1^{st}	2^{nd}	3^{th}	4^{th}	.....	N^{th}
Selected queue	0	0	0	0	.....	1

(b)

Table 4-2 Simulation results

4.2.3 Iteration process

In order to have more reliable results, the simulation process has to be repeated using different sequence of workflow submission that have been generated as workflow submission files in the configuration process. Hence, the simulation process has to be iterated until the number of iterations is equal to the number of workflow submission files.

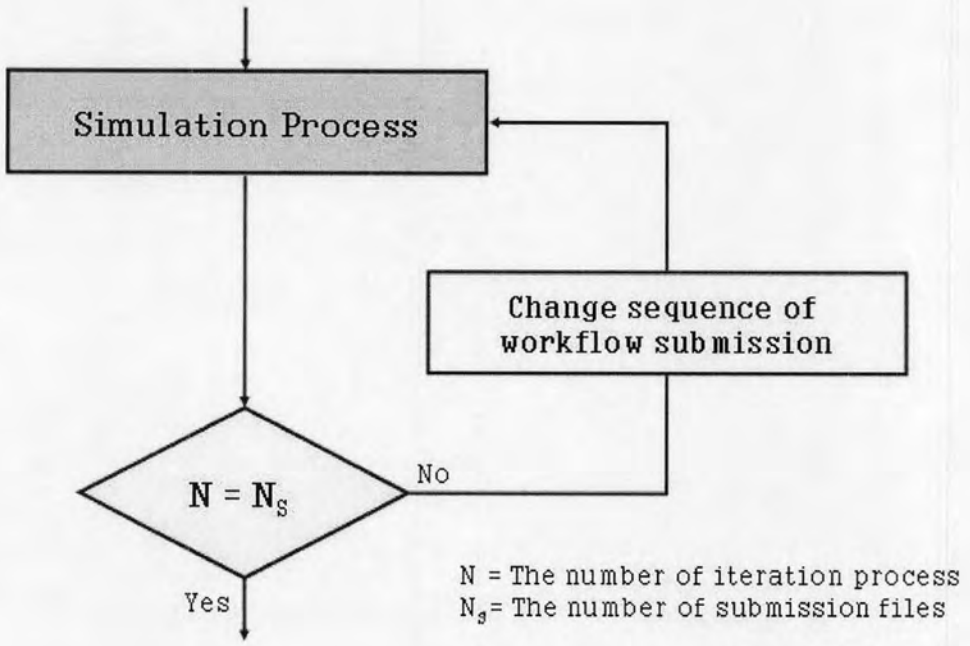


Figure 4-7 Flowchart of iteration process

When the iteration process is completed, the number of simulation results will finally be equal to the number of workflow submission files as illustrated below.

Iteration # \ Workflow ID	1	2	3	4	.....	N
1	30.451	48.136	41.258	25.541	.....	39.297
2	25.577	25.499	25.561	25.536		25.537
3	35.111	34.785	36.573	38.734		34.261
4	29.113	28.747	30.549	32.815		27.088
⋮						
N	28.901	28.448	30.656	33.355		27.033

(a) Workflow completion time

Iteration # \ Scheduling #	1	2	3	4	.....	n^{th}
1	0	0	0	0	.....	1
2	0	0	0	0		1
3	0	0	0	1		1
4	0	0	0	0		1
⋮						
N	0	0	0	0		1

(b) Queue selection order

Table 4-3 Results from iteration process

4.2.4 Analytical process

According to the numerous values of results from the iteration process, an average value has to be evaluated to represent each kind of result which are completion time of each workflow, and queue selection order as shown in Table 4-4.

Iteration # \ Workflow ID	1	2	3	4	.....	N
1	30.451	48.136	41.258	25.541	.....	39.297
2	25.577	25.499	25.561	25.536	.....	25.537
3	35.111	34.785	36.573	38.734	....	34.261
4	29.113	28.747	30.549	32.815	.....	27.088
⋮						
N	28.901	28.448	30.656	33.355	.....	27.033
	29.8306	33.123	32.9194	31.1962	.....	30.6432

(a) Workflow completion time

Iteration # \ Scheduling #	1	2	3	4	.....	n^{th}
1	0	0	0	0	.....	1
2	0	0	0	0	.....	1
3	0	0	0	1	.....	1
4	0	0	0	0	.....	1
⋮						
N	0	0	0	0	.....	1
	0	0	0	0.25	.....	1

(b) Queue selection order

Table 4-4 Analytical results

After the analytical results of workflow completion time has been computed as shown in Table 4-4(a), workflows can be rearranged into arbitrary order by following completion time, final ranking, intermediate ranking, or submission order. Moreover, the analytical results of queue selection order can be achieved as shown in Table 4-4(b) which can efficiently represent scheduling manner

4.3 Experiment

The proposed algorithm can be evaluated by using the simulation system that has been developed as mentioned in the previous chapter. The experimental environment, experiment setting, experiment design, and results are explained below.

4.3.1 Environment

The experiment is run on an Intel server with 3.2 GHz Genuine Intel processor. This simulation is executed on a small cluster which has 4 compute nodes. There is a batch scheduler called SGE (Sun Grid Engine) to schedule job in this cluster.

4.3.2 Experiment setting

Parameters are initiated as default values listed below. Some of these parameters will be varied.

Workflow Characteristics

- Correlation between Intermediate and Final results = 0.75
- Evaluation point = 50%
- Number of workflows = 5,000
- Number of tasks in a workflow = 20
- Workflow pattern or task dependency = Sequence
- Workflow computation size = 10 Teraflops

  Therefore, the computation size for each task is 0.5 Teraflops
- Workflow communication size = 20 bytes,

  Therefore, the communication size for each task is 1 byte.

Host or Compute Node Characteristics

- Number of Hosts or Compute Nodes in Simulation is 200 hosts.
- The hosts have same property such as processor, network.
- Processor type = Pentium 4 (3.06 GHz)
- Processor speed = 1.414 Gigaflops

- Network bandwidth = 0.937 Gbits/s

- Network latency = 0.0025 ms.

The workflow graph for the experiment is a simple sequence of tasks. The first reason is for simplicity and other reason is that it may represent the critical path of a more complex workflow.

4.3.3 Experiment design

The evaluation of this algorithm can be divided into 3 parts of experiment. For each part, a parameter is varied while others are fixed as defaults in order to study the effect of that parameter. The parameters that will be varied are application class, evaluation point, and workflow computation size.

4.4 Results and Discussions

The results of this simulation can be divided into 3 parts which are application class, the effect of the evaluation point, and the effect of workflow computation size.

4.4.1 Application class

This experiment is designed to measure the performance of proposed algorithm when applied with different types of application. This is due to the fact that each individual application may have a specific character, for example, one application may adopt intermediate results in order to precisely forecast the final result. On the other hand, some applications may not be able to use this intermediate result to predict the final outcome since these two results may not be co-related to each other. It should be noted that the relationship between these two results shows how accuracy the final results can be predicted from intermediate results. This relationship can be specified as a correlation between intermediate results and final results called "correlation of results". When the application has high correlation of results, the accuracy of prediction will be better.

In this experiment, seven different classes of applications are tested. Each class can be defined by setting different value of correlations. The application which has highest correlation is taken into account first which is 0.875 follow by 0.75, 0.625, 0.5, 0.375, 0.25, and 0.125 respectively. This will measure whether the proposed algorithm is able to work well when the correlation between intermediate

and final results is low. Also, this will help to conclude which class of application will be able to obtain benefit from this algorithm. The application class can be set as shown below.

Class A: application that has correlation between intermediate results and final results equal to 0.875

Class B: application that has correlation between intermediate results and final results equal to 0.75

Class C: application that has correlation between intermediate results and final results equal to 0.625

Class D: application that has correlation between intermediate results and final results equal to 0.5

Class E: application that has correlation between intermediate results and final results equal to 0.375

Class F: application that has correlation between intermediate results and final results equal to 0.25

Class G: application that has correlation between intermediate results and final results equal to 0.125

The experiment initiates from generating the value of intermediate results and final results for each class of application. To conduct this, statistic equation called "Random Bivariate Gaussian" is used as described in the previous part (configuration process). The generated intermediate results and final results for each class of application can be found in Appendix A.

In order to evaluate the effect of correlation, the experiment can be set as 2 sections which are the effect on completion order, and the effect on completion time. The detail of each section is explained as follows.

A) The effect of correlation on completion order

In this experiment, the proposed algorithm is compared to others by considering the completion order of each algorithm. The desired experimental result is to get workflows finish in the same order as their ranking. In the other word, the high-

ranked workflow should come out earlier than the lower one. Therefore, the purpose of this experiment is to find algorithm that meet the objective as described above.

From the experiment, several graphs can be plotted to show the relationships between average final ranking and completion order. From the experimental results below, it is obvious that the proposed algorithm tends to finish the high-ranked workflows first while the order of workflow completion of other two algorithms, FIFO and Random, is not related to the quality of the results. However, there are other 2 types of scheduling that provide the same results. Therefore, it can be concluded that there are 3 scheduling types which have outstanding performance in managing the completion order of this application.

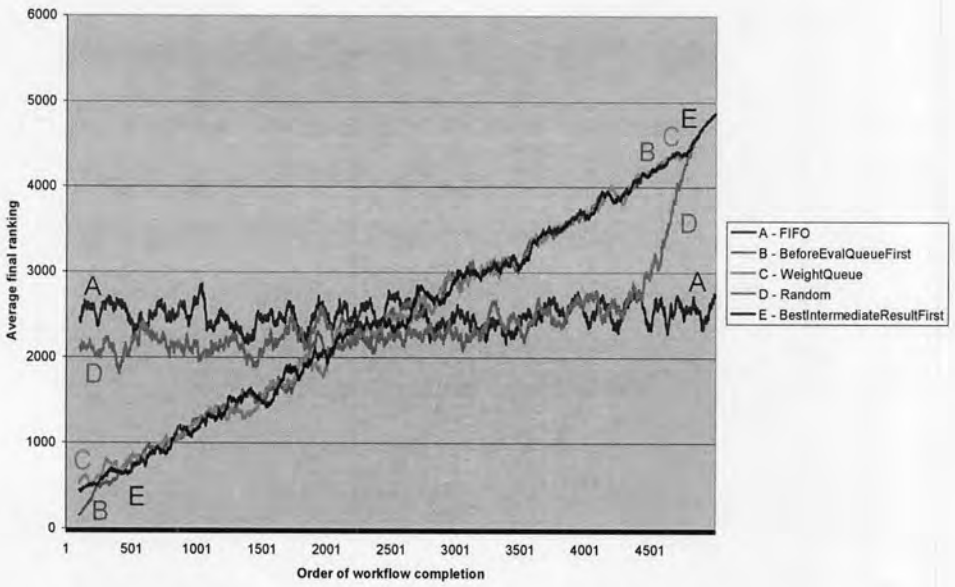


Figure 4-8 Relationship between average final ranking and completion order for class A application (correlation of results = 0.875)

Since the class A applications has the highest correlation between intermediate results and final results, compared to the other classes, The effectiveness of the algorithms that can exploit the intermediate results is also the highest. It can be seen from Figure 4-9 that for class D application, the effectiveness of BEQF, weight queue, and BIRF algorithm reduces, and finally in class G application, all algorithms are virtually the same.

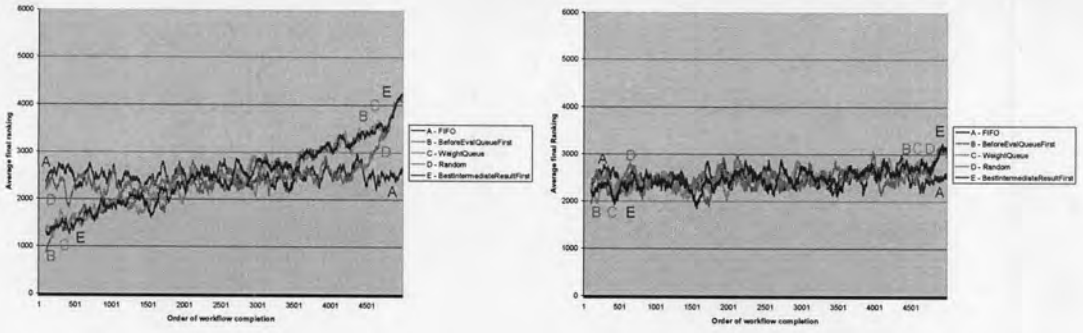


Figure 4-9 Relationship between average final ranking and completion order
(a) class D application (correlation of results = 0.5),
(b) class G application (correlation of results = 0.125)

B) The effect of application class on completion time

The objective of this algorithm is to minimizing the turnaround time of the workflows that give the best results. Therefore, the expected experimental result is to finish the high-ranked workflow as soon as possible. In the other word, the high-ranked workflows should have low completion time. An experiment was designed to measure the performance of the proposed algorithm by considering the completion time. However, as described above, the proposed algorithm may not provide benefit to some class of applications. As a result, different classes of applications are tested in order to show whether the proposed algorithm still able to work well with an application that has low correlation of results.

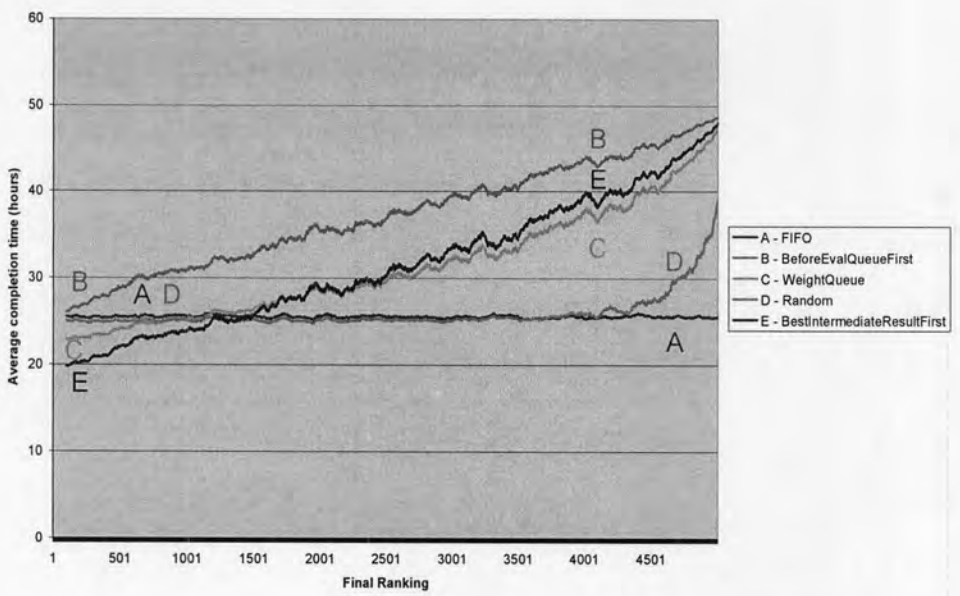


Figure 4-10 Relationship between average workflow completion time and final ranking for application class A (correlation of results = 0.875)

Figure 4-10 demonstrates that the Best-Intermediate-Result-First algorithm can finish the first 1250 ranked workflows by using less completion time than other algorithms. The second algorithm that can perform well is Weight Queue which can finish 1-156 ranked workflows about 3 hours later than the proposed algorithm. In the other word, the proposed algorithm can finish the high-ranked workflows earlier than other algorithms at least about 3 hours. Therefore, it can be concluded that the Best-Intermediate-Result-First algorithm is the most appropriate algorithm for Class A application.

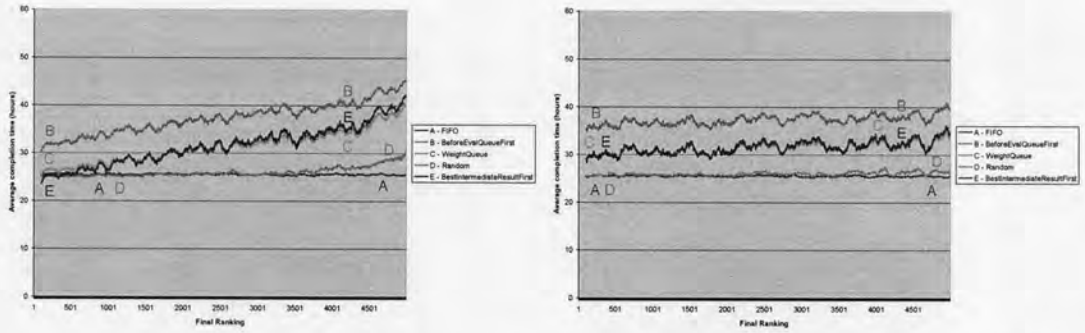


Figure 4-11 Relationship between average workflow completion time and final ranking
(a) application class D (correlation of results = 0.5)
(b) application class G (correlation of results = 0.125)

C) Summary

The purpose of experiment is to examine the efficient of proposed algorithm. As can be seen from the test, its result revealed the fact that although there are some variables in correlation value; the function of algorithm was still able to perform perfectly. When compare with other algorithms, the proposed algorithm can finish the high-rank workflows earlier.

According to the experimental results, the Best-Intermediate-Result-First algorithm can perform well when the correlation was higher than 0.5. However, the performance is clearly obvious when the correlation was higher than 0.625.

4.4.2 The effect of the evaluation point

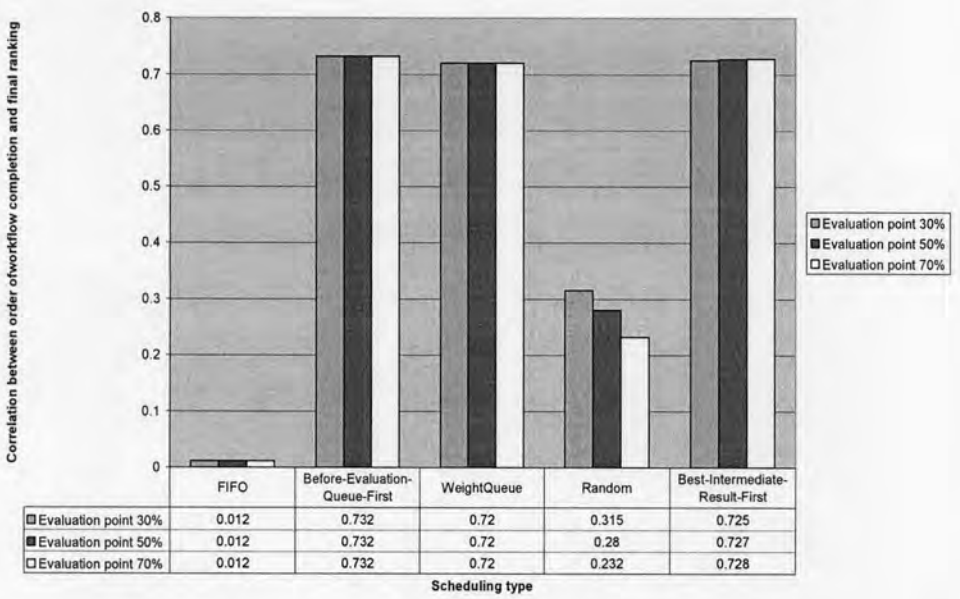
The objective of this experiment is to find an appropriate evaluation point of application when using the proposed algorithm. The evaluation point may depend on the application characteristic itself or can be partially selected. The evaluation point of

some applications may not be selected since it may have only one state that provides intermediate results. However, some applications may have several states that can provide intermediate results so that the most appropriate state should be selected. As a result, this experiment was designed to measure the performance of algorithm by varying the position of evaluation point. In this experiment, nine positions of evaluation point are tested which are at the 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%.

In order to evaluate the effect of evaluation point, the experiment can be set as 2 sections which are the effect on completion order, and the effect on completion time. The detail of each section is explained as follows.

A) The effect of evaluation point on completion order

This experiment was designed to evaluate the effect of evaluation point on completion order for each type of scheduling. The desired experimental result is to get workflows finish in the same order as their ranking. From the experiment, several graphs can be plot which shows relationship between average final ranking and completion order. These graphs can be found in Appendix B. However, the ability to control completion order can be calculated as a correlation between completion order and final ranking. Therefore, this correlation was computed to show the accuracy of each algorithm at each evaluation point as illustrated below.



	FIFO	Before-Evaluation-Queue-First	WeightQueue	Random	Best-Intermediate-Result-First
Evaluation point 30%	0.012	0.732	0.72	0.315	0.725
Evaluation point 50%	0.012	0.732	0.72	0.28	0.727
Evaluation point 70%	0.012	0.732	0.72	0.232	0.728

Figure 4-12 Correlation between order of workflow completion and final ranking

From the experimental results above, there are three algorithms that show outstanding performance in managing the completion order at every evaluation point which are Before-evaluation-queue-first, Weight queue, and Best-intermediate-result-first. Moreover, according to each algorithm, it is obvious that there is no significant change in correlation between completion order and final ranking at each evaluation point. Therefore, it can be concluded that evaluation point will not have any effect on the accuracy of the algorithm.

B) The effect of evaluation point on completion time

An experiment was designed to measure the performance of the proposed algorithm by considering the completion time. However, as described above, the proposed algorithm may not provide benefit when using inappropriate evaluation point. As a result, different evaluation points are tested in order to find the most appropriate evaluation point.

The experimental results can be analyzed by considering the relationship between final ranking and average completion time when using different evaluation points which are 30%, 50%, and 70%.

Figure 4-13 shows the relationship when using evaluation point at 30%. The graph demonstrates that the Best-Intermediate-Result-First algorithm can finish the first 1250 rank of workflow by using less completion time than other algorithms. The average completion time of the 1-156 rank of workflows is 18.15 hours, and of the 157-313 rank of workflows is 19.158 hours. However, the second algorithm that can perform well is Before-Evaluation-Queue-First. The completion time of first 156 ranked workflows is 18.7 hours which is only 0.55 hours later than in Best-Intermediate-Result-First. Therefore, if the evaluation point is set at some positions below 30%, the Before-Evaluation-Queue-First tends to performs better than the proposed algorithm.

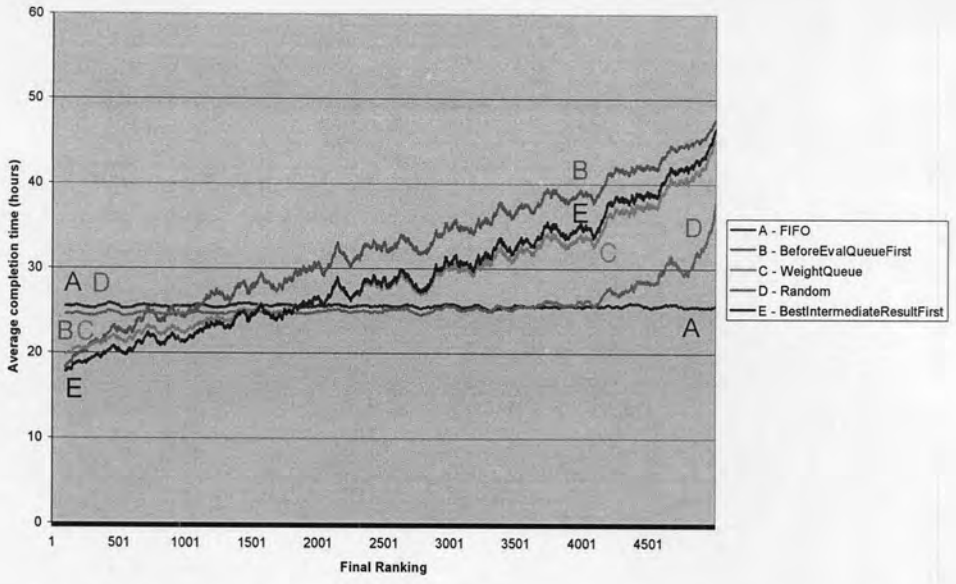


Figure 4-13 Relationship between final ranking and average completion time (hours) when evaluation point 30%

When the evaluation point is set at 50% as shown in Figure 4-14, the proposed algorithm will finish the high-ranked workflows earlier than other algorithms. The first 625 ranked workflows will have less completion time when using this algorithm. The average completion time of 1-156, 157-313, and 314-625 ranked workflows are 20.91, 22.08, and 23.3 hours respectively. The algorithm that can perform well next to the proposed algorithm is Weight Queue. The average completion time for those rank of workflow above are 4.07, 2.76, and 1.72 hours less than the Best-Intermediate-Result-First respectively.

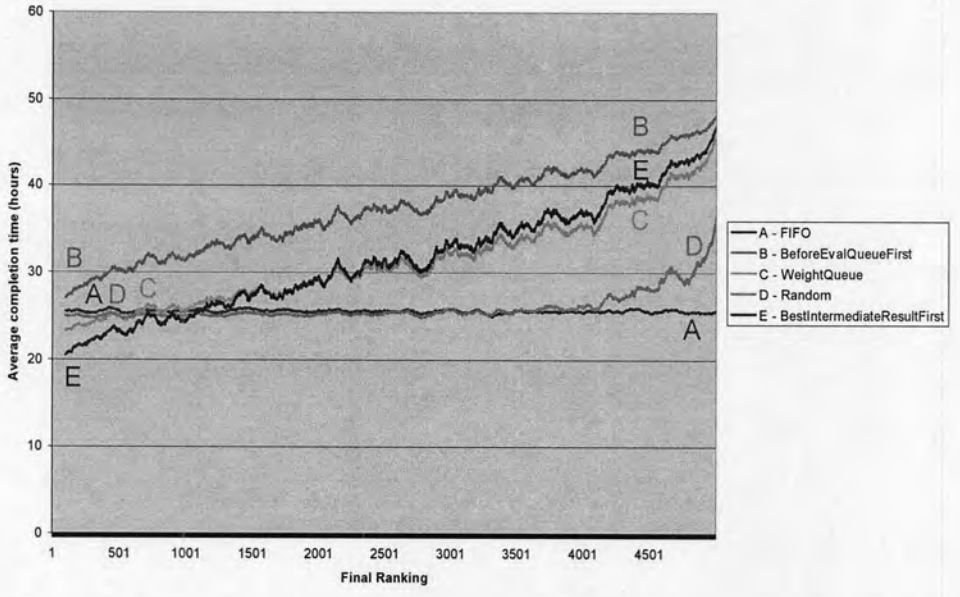


Figure 4-14 Relationship between final ranking and average completion time (hours) when evaluation point 50%

When the evaluation point is set at 70%, the experimental result shows that the Best-Intermediate-Result-First can finish the first 313 ranked workflows by using less completion time than other algorithms. The average completion time of 1-156 and 157-313 ranked workflows are 23.62 and 24.92 hours respectively.

The algorithm that can perform well next to the proposed algorithm is Random. The average completion time of 1-156 and 157-313 ranked workflows are 1.7 and 0.251 less than the Best-Intermediate-Result-First respectively. It is obvious that the differences are reduced when compare to those at 50%. Hence, if the evaluation point is set up to 80% or 90%, the Random algorithm tends to perform better than other algorithms.

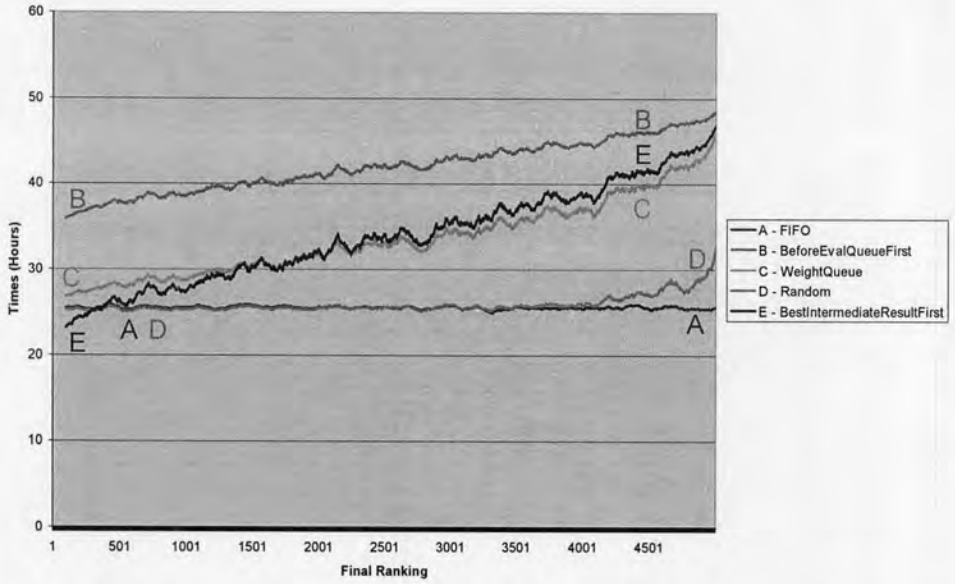


Figure 4-15 Relationship between final ranking and average completion time (hours) when evaluation point 70%

C) Summary

This experiment was conducted to find the most effective evaluation point of the proposed algorithms. From the experimental results, it can be concluded that the appropriate evaluation point for the proposed algorithm should be in the range of 30%-80%. It is also obvious that the most appropriate evaluation point is at 50%

4.4.3 The effect of workflow computation size

According to all experiments above, computation size of each workflow is equally set to be 10 Teraflops. However, in most applications, each workflow may

have different computation sizes. Therefore, an experiment is designed to show the effect of the case on the algorithm performance.

In order to perform this experiment, computation size of each workflow has to be generated. This can be achieve by using random normal distribution with the mean at 10 Teraflops and standard deviation at 2.5 Teraflops. After that, it will be randomly assigned for each workflow. The generated workflow computation sizes is illustrated in Appendix C.

A) The effect of using different computation sizes on completion order

This experiment was designed to asses the effect of using different workflow computation sizes on completion order for each type of scheduling. The desired experimental result is to get workflows finish in the same order as their ranking. From the experiment, several graphs can be plotted which show the relationship between average final ranking and completion order. These graphs can be found in Appendix A. However, the ability to control completion order can be calculated as a correlation between completion order and final ranking. Therefore, this correlation was computed to show the accuracy of each algorithm as illustrated below.

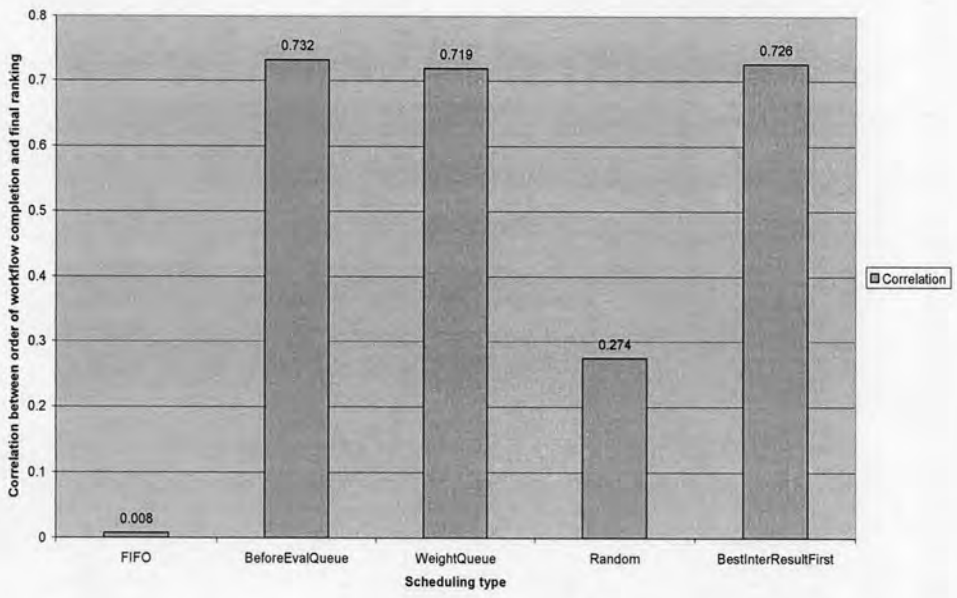


Figure 4-16 Correlation between completion order and final ranking when using difference workflow computation sizes

From the experimental results as shown in Figure 4-16, the algorithm that has the highest correlation between completion order and final ranking is Before-evaluation-queue-first which has a correlation of 0.732. Then, the second is Best-

intermediate-result-first (correlation = 0.726), follow by Weight Queue (correlation = 0.719), Random (0.274), and FIFO (correlation = 0.008) respectively. It is obvious that the performance of the proposed algorithm is not significantly different when comparing with Before-evaluation-queue-first algorithm. Therefore, it can be conclude that the proposed algorithm can still efficiently handle the completion order even when using different computation sizes.

B) The effect of using different computation size on completion time

This experiment was designed to measure the performance of the proposed algorithm when using with the application that has different workflow computation sizes. The experimental results are collected in order to show whether the proposed algorithm still able to work well in that case.

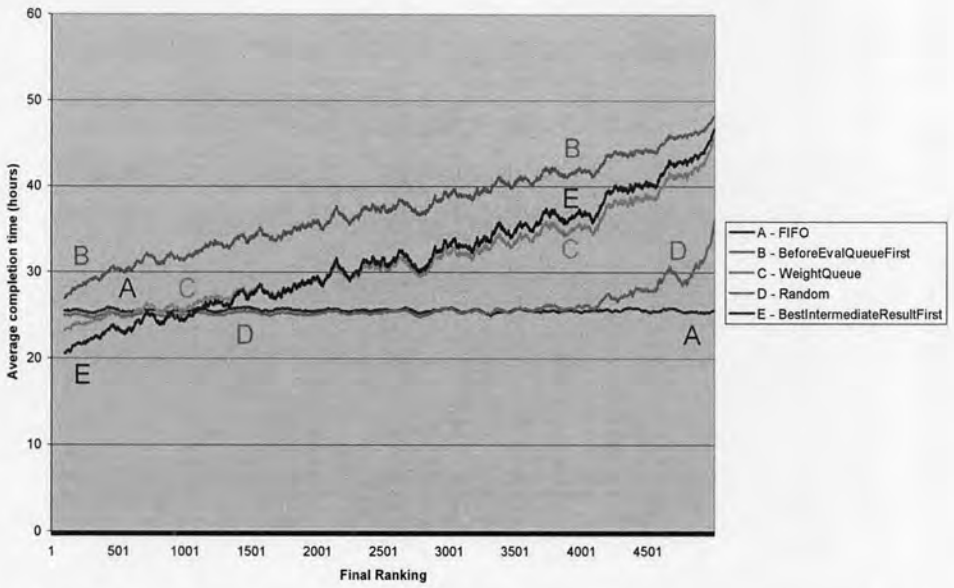


Figure 4-17 Relationship between final ranking and average completion time (hours) when using different workflow computation size

Figure 4-17 shows the relationship between final ranking and average computation time (hours) when using different workflow computation sizes. The graph demonstrates that the proposed algorithm can finish the high-ranked workflow out earlier than other algorithms. Therefore, it can be conclude that the proposed algorithm still able to work well even when using with the application that has different sizes of workflow computation.

C). Summary

From this experiment, computation size will be adjusted in different values in order to test the durability of proposed algorithm.

The experiment also demonstrates that the proposed algorithm remains good even though the workflow computation size varies. This result corresponds to other experiments.