

Scheduling of Jobs in a Dynamic Heterogeneous Environment

Ponsy R.K. Sathia Bhama¹, Thamarai Selvi Soma Sundaram¹, Supriya Vasudevan²,
P.Ai Niyas², K.Swadha²

Department of Information Technology
Anna University
Chennai, India

Abstract - In a heterogeneous environment, resources are autonomous, distributed, dense, and dynamic, hence they should be effectively scheduled so that maximum utilization of the resources is possible. Also, the environment needs to be tolerant to network failures. In this paper we propose two algorithms for a grid scheduler which basically address the problem of fault tolerance by the periodic broadcasting of messages which makes the environment resilient to failures thus making it fault tolerant. The job migration policies follow dynamic parameters like network response time and the available start time, which enhances the performance of the individual tasks. An extensive comparative study is done based on how the three algorithms cope up with increasing loads. Also the grid scheduling policies are compared with that of centralized policies. The result based on the key metrics show that the performance of these algorithms is significantly greater than that of the centralized policies.

Keywords: Grid, Round trip time, Available start time, Push mode, Pull mode

1.0 Introduction

Grid computing is concerned with the sharing, coordinating distributed heterogeneous resources to work as a single computational resource. Grid applications uses high performance distributed resources like high performance systems, networks, databases, etc.. These are enabled via grid middleware such as Globus, Gridbus. The resources in grid are dynamic. Grid scheduling is the process of scheduling resources over multiple domains based upon task requirements, throughput, application performance, budget constraints, deadlines, etc.. Grid scheduling on the whole is a mapping of individual tasks to computer resources. After the resources have been identified, the jobs to be executed by the resources are to be scheduled. Then a job scheduler should be used to coordinate the execution of the jobs. It should also be noted that there could be different levels of schedulers within a grid environment. For instance, a cluster could be represented as a single resource. The cluster may have its own scheduler to help manage the nodes it contains. A higher level scheduler (sometimes called a meta scheduler) might be used to schedule work to be done on a cluster, while the cluster's scheduler would handle the actual scheduling of work on the cluster's individual nodes.

All grid scheduling systems work on the basic principle that the “new task” which needs to be executed has to make itself known to a “resource selector”. In current systems, the resource selector acts as a gateway to the grid. It will select resources from a global directory and then allocate the job to one of the available grid nodes. A scheduler that allows to request resources from more than one machine for a single job may perform load balancing of workloads across multiple systems. Each system would then have its own local scheduler to determine how its job queue is processed which requires advance reservation capability of local schedulers.

2.0 Scheduling Process

The scheduling architecture comprises of the following major components – local schedulers, local queues, grid scheduler, grid queue. As jobs arrive they are stored in the grid queue and then sent to the grid scheduler. The grid scheduler schedules the job according to the requirements and submits the job to local scheduler or to the grid scheduler of another system. Grid scheduling systems operate in dynamic environments subject to various unforeseen and unplanned events that can happen at short notice. Such events include the breakdown of computers, arrival of new jobs; processing times are subject to stochastic variations, etc. It turns out that the performance of a schedule is very sensitive to these disturbances, and it is difficult to execute a predictive schedule generated in advance. So, we go for scheduling with dynamic parameters.

2.1. Scheduler

The scheduler performs the following sequential tasks

- Phase 1: Resource Discovery
- Phase 2: Resource Selection
- Phase 3: Job Scheduling
- Phase 4: Job migration and execution
- Phase 5: Job monitoring

The scheduler is responsible for resource discovery, resource trading, resource selection, and job assignment. During resource discovery, information about list of authorized and available machines, resource access cost, and resource status information is maintained. The resource selection algorithm is responsible for selecting those resources that meet the constraints and then the job is executed. The scheduling algorithm makes the decision of which job is to be run in which system and when the job is to be executed. The job migration algorithm involves how to migrate the job in case the machine is not able to cater to the needs of the application. The monitoring part handles the issue of fault tolerance by broadcasting the status periodically between nodes.

2.2 Classification of Scheduling Algorithms

There are in general two broad classification of scheduling algorithms, viz.

- Centralized Algorithms, where a centralized manager takes care of the resource allocation and the execution of jobs. In these centralized policies, all jobs are submitted to a single grid scheduler which does not have an affinity to a specific local system
- Distributed Algorithms, where there is no central component. Each node is capable of handling the job execution or resource allocation. In the local scheduling algorithm, there are no grid schedulers. All jobs are submitted to local schedulers and execute on the compute server associated with each local scheduler.

2.3 Scheduling Decision Parameters

To make a scheduling decision, the scheduling system needs to take various parameters into consideration, which includes:

- Queue length of each node
- Free or Available Nodes

- Available start time (which is the sum of execution time of all the jobs being executed in the local system currently)
- $Available\ start\ time = \sum_{I=1\ to\ n}(execution\ time(job\ I))$
where there are 1 to n jobs being executed.
- Round trip time (in Terms of no of Sent and Received packets)between different nodes.

3.0 Scheduling Algorithm

As jobs arrive for execution, the scheduler selects the relevant resource required for processing the job from the available set of resources. The selection of the resource is based on the prediction of the computing power of the resource machine. Make span is a measure of the throughput of the heterogeneous computing system. The objective of the grid scheduling algorithm is to maximize the make span.

3.1 General Scheduling Algorithm

```

While there exists jobs to be scheduled
  For all jobs to schedule
    For all machines
      Calculate expected burst time
    End for
  End for
  Choose the best machine to execute the job
  (Minimal burst time)
  Assign job to the best machine
End while

```

The general scheduling algorithm does not consider any decision parameter, which affects its performance in a general grid environment. Regardless of their computing power request, some tasks may require high network bandwidth to exchange a large amount of data among processors, whereas others can be satisfied with low network bandwidth. Based on the general adaptive algorithm, 2 new scheduling algorithms have been proposed by considering the above decision parameters.

3.2 Proposed Algorithms

3.2.1 Push Mode Algorithm

In the push strategy, the grid scheduler sends the resource requirements of the job to its peers. Before a job begins to execute, it needs to wait in a local queue as well as transfer the input data to the other available systems. **Round trip time** is the *approximate data transfer time* from initial node to destination node. The scheduler decides which is the destination system based on which has the minimum round trip time. The job is then sent to the local scheduler or the grid scheduler of the destination machine according to the availability of resources and the requirements of the job.

- Request Resource status of J to the Partner Set of L (PS(L))
- Compute the Response time for all the machines available Find Machine M such that $N_J^M = \min(RT_J^L, RT_J^{PS(L)})$
where RT=Response time ,that is the Transfer Rate between L and PS(L)
- Move J to LQ^M

Figure 1 shows the activity diagram of the push mode algorithm

3.2.2 The Pull Mode Algorithm

The pull mode algorithm takes a more passive approach to job migration than the push strategy. Here, each system in the computational grid checks its own resource usage periodically at time interval θ . If the no of running jobs is below the threshold value θ , the machine volunteers itself for receiving jobs by informing its partner set of its low utilization. If a system has its threshold value crossed, it checks for the availability of any volunteer message and by comparing the Available start time (AST), the job is migrated.

Each Machine R checks own Resource Utilization Status (RUS^R) at time interval σ

- IF ($RUS^R < \delta$) send Availability Message (AM^R) and Available Start Time (AST) to Partner Set of R ($PS(R)$)
- If (Machine L with Job J in GQ^L receives AM^R)
Find Machine M such that $AST = \min (AST(L), AST(RR(L)))$
where $RR(L)$ is the set of Machines which have sent AM
- Move J to LQ^M

Figure 2 shows the activity diagram of the pull mode algorithm

Thus the proposed algorithms will maximize the utilization of resources and issue fault tolerance which are the primary objectives of grid scheduling.

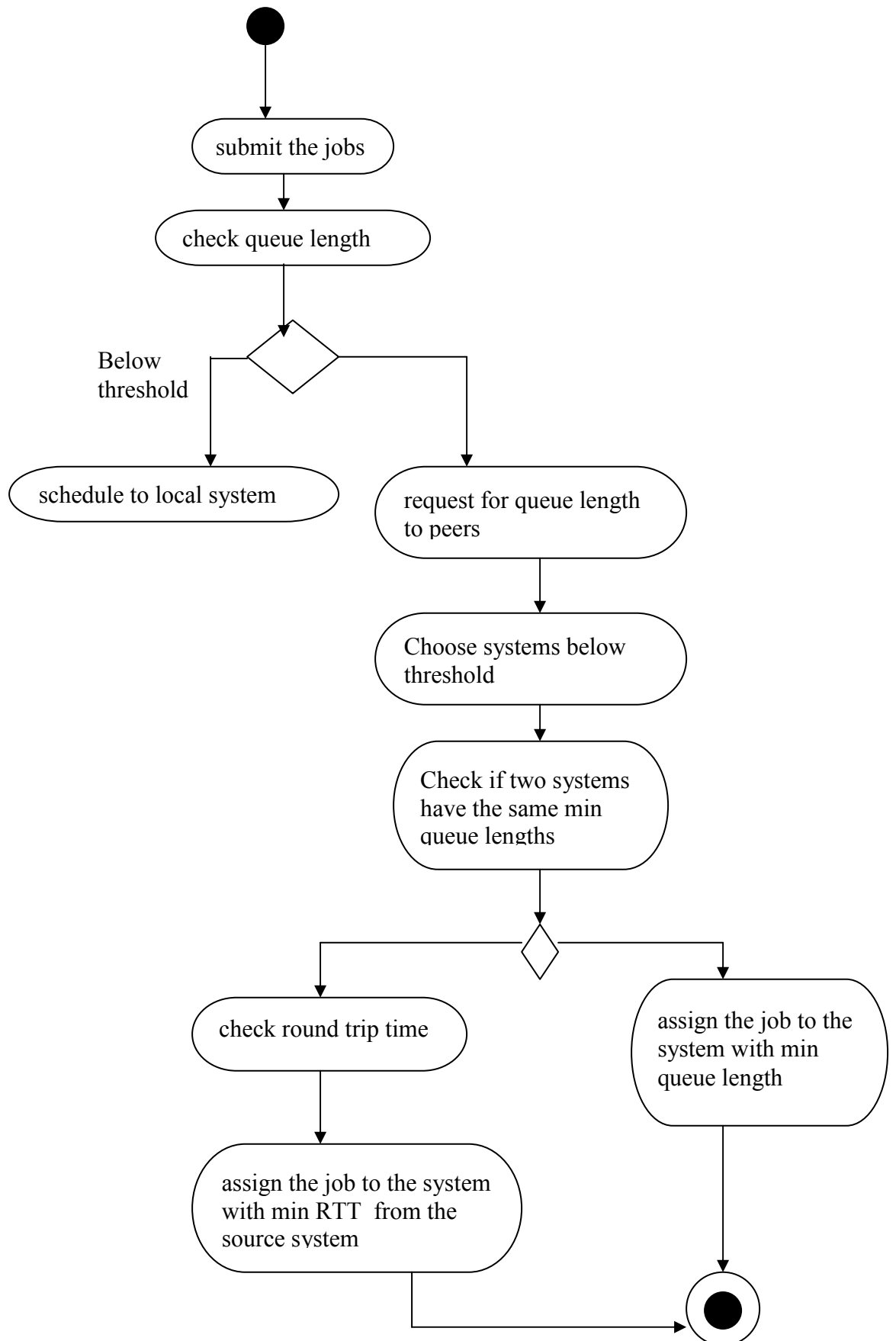


Figure 1: Push Mode Activity Diagram

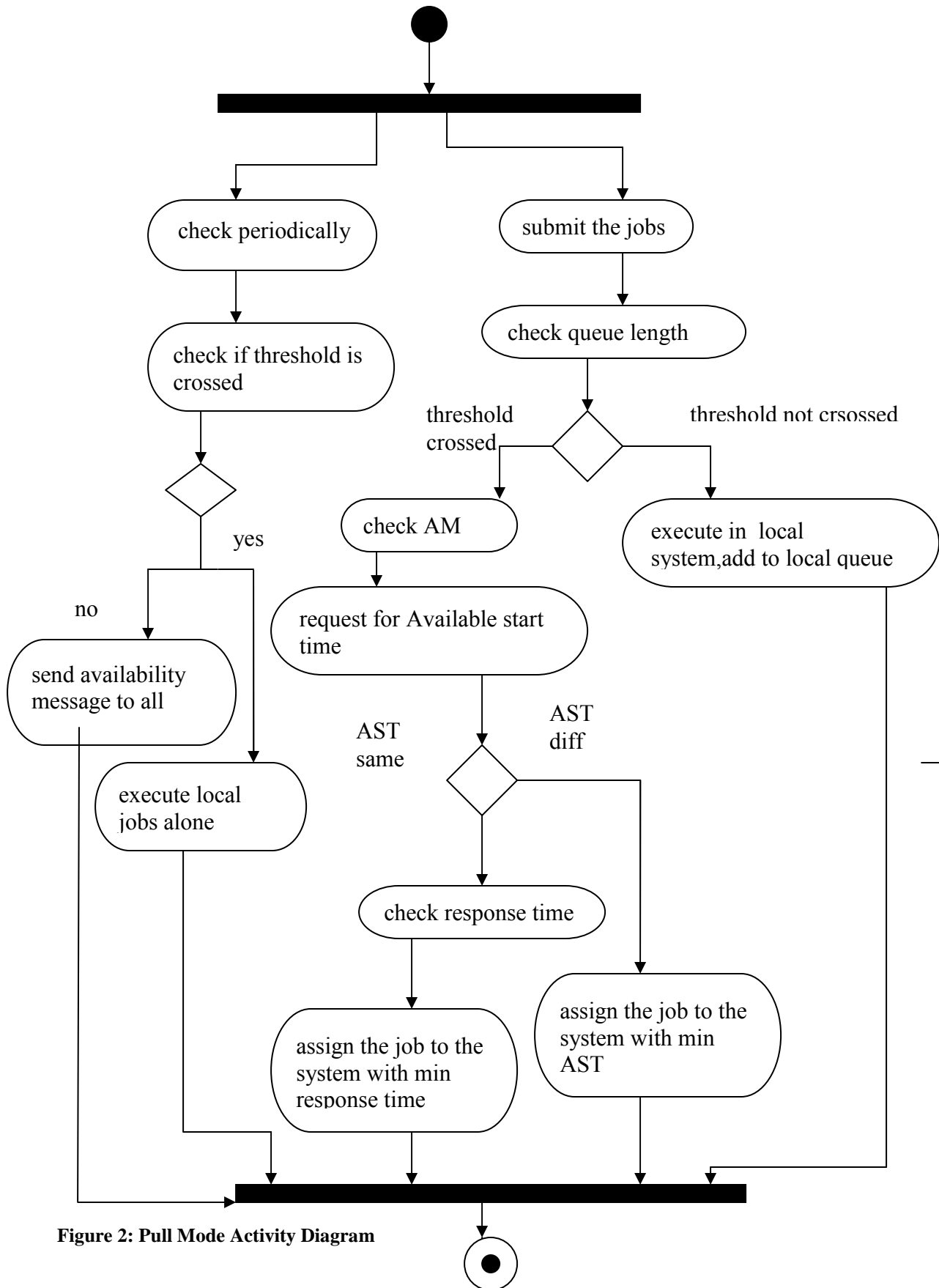


Figure 2: Pull Mode Activity Diagram

4.0 Experimental Results

This section presents and analyzes the simulation results of our job migration algorithms using the performance metrics described in Section 2.3.

To begin, we compare the performance of our push mode and pull mode job migration algorithms. The performance data for these algorithms is shown in figure . One of the most important metrics is the average response time because that is ultimately what users care about. We find that the Push Mode algorithm has the lowest average response time than the response times resulting from the other two algorithms. The Pull Mode algorithm have the next highest response times.

Figure 3 shows the comparison of the performance of the centralized algorithm with that of the grid environment as well as with the sequential execution of jobs, where there is no job migration. Figure 4 shows the comparison of the performance of our migration techniques

In both the graphs the response time is plotted against the no of jobs, which demonstrates the scalability of the algorithms with the increasing no of jobs.

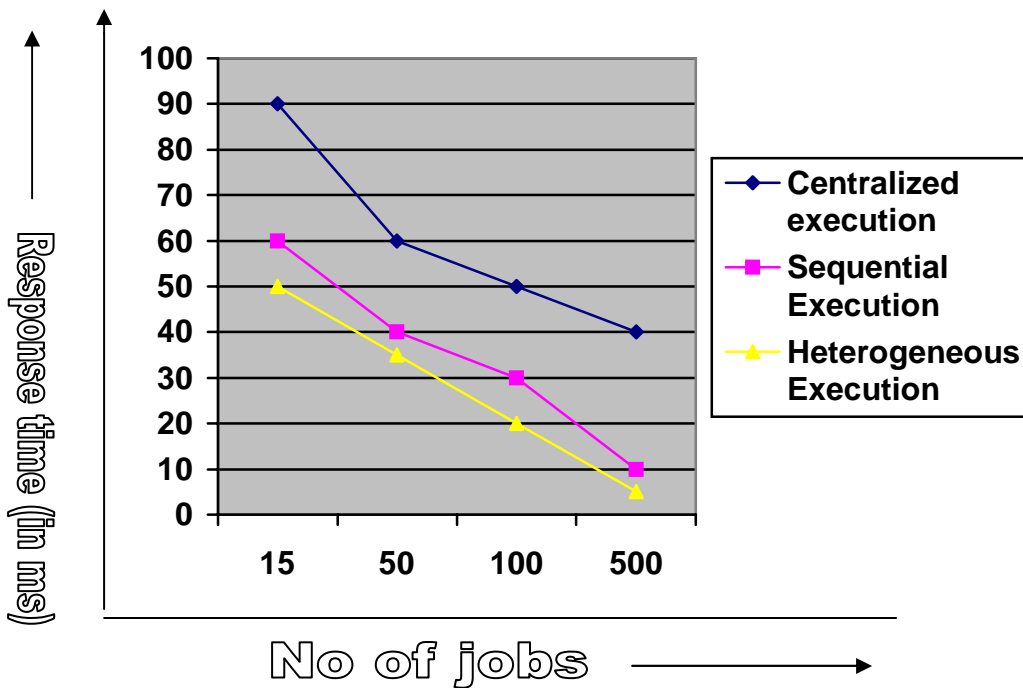


Figure 3: Comparison of the performance of the centralized algorithm with that of the grid and sequential execution

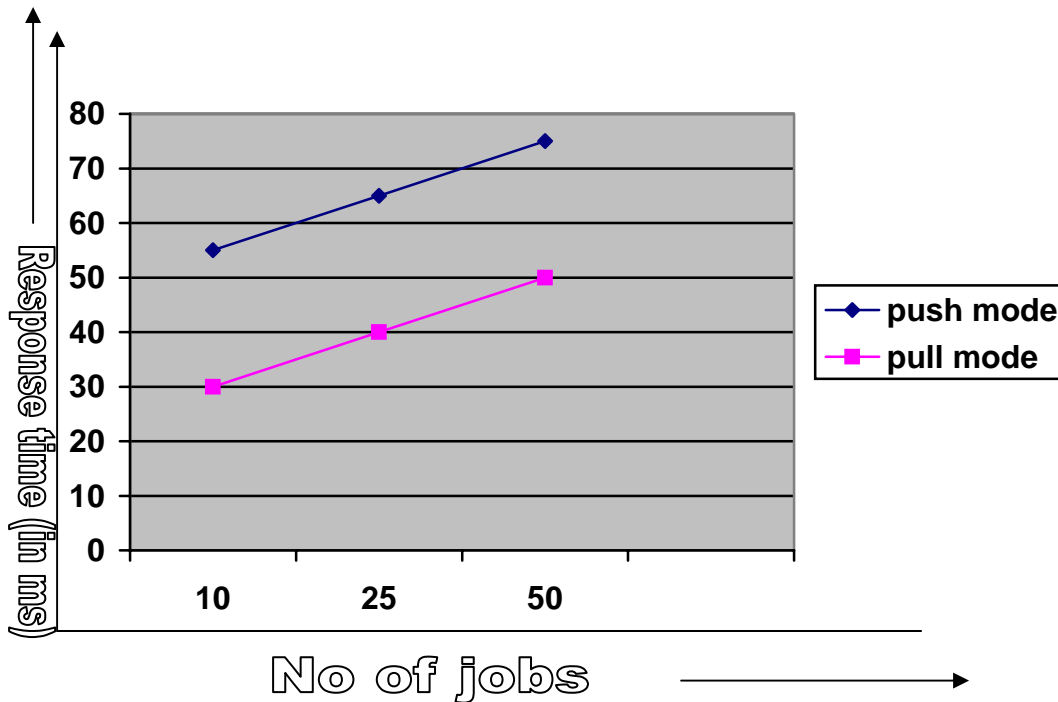


Figure 4: Comparison of the performance of our migration techniques

5.0 Conclusion

The grid provides an efficient platform to execute real time applications in an efficient manner and also it provides a way to maximize resource utilization. The algorithms that we have proposed will completely overcome the major drawback of general adaptive scheduling algorithm. The major scheduling algorithms suffer mainly from the network failures, which render them inefficient. In this paper, the algorithms which we have proposed help overcome the network failures and revert back the system to the original state by the periodic broadcasting of the status of each node. Also the dynamic parameters are considered which help in the speeding up the execution of tasks. Also the experimental results show that the concept of scalability has been incorporated as the algorithms scale well with increasing workload. The future work can include the usage of multiple heterogeneous resources as in this case we have considered CPU time as the only resource.

6.0 References

- [1] Rajkumar Buyya, David Abramson, Jonathan Giddy, "An Economy Driven resource Management Architecture for Global Computational Power Grids"
- [2] I. Foster, C. Kesselman, et.al., "The Anatomy of Grid" .
- [3] I. Foster and C. Kesselman, et.al., "The Physiology of the Grid" .
- [4] I. Foster, "What is the Grid? A Three Point Checklist"
- [5] Lijuan Xiao, Ynmin Zhu, Lionel M. Ni, Zhiwei XU, " GridIS: An Incentive-based Grid Scheduling", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium(IPDPS'05)
- [6] Enis Afgan , "Role of Resource broker in the Grid", ACMSE 2004
- [7] E, Huedo, Ruben S. Montero, Ignacio M. Liorente, " Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications ", Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (EUROMICRO-PDP'04)
- [8] I. Foster, Chuang Liu, et.al., "Design and Evaluation of a Resource Selection Framework for Grid Applications"