Decoupled Scheduling via Receiver Initiated Approach

Ponsy R.K. Sathia Bhama [1],Thamarai Selvi Soma Sundaram[1],

Sivakama Sundari .R[2], Bakiyalakshmi.R[2],Thamizharasi.K[2]

Abstract
Grid scheduling is the process of mapping grid jobs to resources over multiple administrative domains. Traditionally, application-level schedulers have been tightly integrated with the application itself and were not easily applied to other applications. This design is generic that decouples the scheduler core (the search procedure) from the application-specific (e.g. application performance models) and platform-specific (e.g. collection of resource information) components used by the search procedure. In this decoupled approach the application details are not revealed completely to broker, but customer will give the application to resource provider for execution. The resource providers are clusters which is a collection of nodes. Moreover, to avoid fault occurrence and insecure conditions, job migration is performed with in the clusters. In a decoupled approach, apart from scheduling, the resource selection can be performed independently in order to achieve scalability.

Keywords:
Meta, grid scheduling, application-level scheduler, decouple, scheduler core, cluster and performance model

1. Introduction

GRID is a system for management and aggregation of autonomous, heterogeneous, computational and storage resources across geographical and administrational boundaries. Grid computing is a relatively new distributed computing paradigm that is gaining importance. It offers a solution to the increasing demand of highly computational and storage power, without requiring any extraordinary investments in the hardware infrastructure. However, in many cases the grid is not utilized properly without further optimization such as scheduling mechanisms for efficient assignment of application to available resources [4]. So application scheduling is the key issue for deploying parallel and distributed applications at large scale in grid.For the purpose of application scheduling, the problems of discovering available resources, selecting an application-appropriate subset of those resources, and mapping of data and tasks onto selected resources are addressed. This scheduler design seeks flexibility through modularity. And that module will explicitly *decouples* the scheduler core (the search procedure) from application-specific (e.g. performance models) and platform-specific (e.g. resource information collection) components used by the search procedure.

This scheduling approach focuses onminimizing the execution time of a single application on a set of potentially shared resources. This approach has been termed application-level scheduling [2]. In scheduling a large number of user applications for parallel execution on an open-resource Grid system, the applications are subject to system failures or delays caused by infected hardware, software vulnerability, and distrusted security policy. This paper considers the risk and insecure conditions in grid scheduling with in the clusters through job migration.

2. Decoupled Scheduling

This section describes the decoupled scheduling approach. To provide context for this description, the detailed scheduling scenario is addressed. A user has an application and wishes to execute that application on computational grid resources. The application is parallel and may involve significant inter-process communication. The target Computational Grid consists of heterogeneous workstations connected by LANs and/or WANs [2]. When the user is ready with the application, the broker is contacted to submit the requirements of the application. The resource provider publishes themselves to Broker declaring their resources and the type of application that they can execute. Broker then finds the suitable provider for the user's requirements. Broker return the domain address of the provider to user. Using the domain address, user will then contact the resource provider. The provider then retrieves CPU speed, memory, cache size and load average from its nodes. Calculates the completion time for the given application in all nodes. The least suffered node (least completion timed node) is selected to execute the application.

Manuscript received May 5, 2025
Manuscript revised May 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.5.14

Table I: Proposed algorithms

Algorithms	Description
Dynamic Fastest Processor Task First (DFPTF)	This algorithm provides fastest processors to largest task. It is dynamic and avoids starvation. The metric considered for this algorithm is task size.
Sufferage	This retrieves the least suffering machine for a particular job. Metrics considered are processor speed, memory capacity, cache size and load average.
Max-Min-Max	Both Max-Min and Min-Min algorithms are coupled. This algorithm prefers either large or small application based on number of large or small applications. It is dynamic and the parameter considered for this algorithm is execution time.

3. Phases of Scheduling

The scheduler performs the following sequential tasks

- Phase1: Resource Discovery
- Phase2: Resource Monitoring
- Phase3: Resource Selection.
- Phase4: Job Scheduling
- Phase5: Job execution

The scheduler is responsible for resource discovery, resource monitoring and resource selection. During resource discovery, lists of authenticated resources that are available for job submission are identified. In order to cope with the dynamic nature of the Grid, a scheduler will have dynamic state information about the available resources into its decision-making process. The resource selection algorithm is responsible for selecting the resource providers that is capable of executing the application. The scheduling algorithm will make the decisions of which task is to be run in which node under the resource provider. This includes ordering the list of machines in a resource provider for executing the task. The monitoring part handles the issue of fault tolerance by broadcasting the status periodically between the nodes.

Resource Discovery and Monitoring

An information service is a vital component of the grid infrastructure. It maintains knowledge about resource availability, capacity, and current utilization. Within any grid, both CPU and data resources will fluctuate, depending on their availability to process and share data. As resources become free within the grid, they can update their status within the grid information services. The client, broker, and/or grid resource manager uses this information to make informed decisions on resource assignments. The information service is designed to provide:

- Efficient delivery of state information from a single source
- Common discovery and enquiry mechanisms across all grid entities

Information service providers are programs that provide information to the directory about the state of resources. Examples of information that is gathered includes:

1. Static host information: Operating system name and version, processor vendor/model/version/ speed/cache size, number of processors, total physical memory, total virtual memory, devices, service type/protocol/port
2. Dynamic host information : Load average, queue entries, and so on
3. Storage system information : Total disk space, free disk space, and so on
4. Network information Network bandwidth, latency, measured and predicted
5. Highly dynamic information Free physical memory, free virtual memory, free number of processors, and so on

The Grid Information Service (GIS), also known as the Monitoring and Discovery Service (MDS), provides the information services in Globus. The MDS uses the Lightweight Directory Access Protocol (LDAP) as an interface to the resource information. Monitoring and Discovery Service (MDS): MDS provides access to static and dynamic information of resources. Basically, it contains the following components:

1.Grid Resource Information Service (GRIS)

2. Grid Index Information Service (GIIS)

3. Information providers
4. MDS client

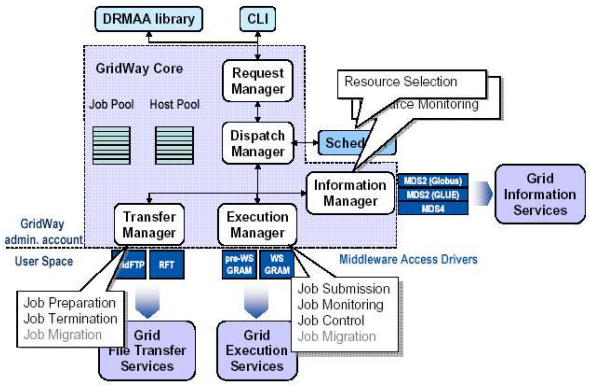


Figure 1: Grid Scheduling Infrastructures

Globus Resource Allocation Manager (GRAM) is part of the Globus Toolkit used for job submission. The Gram Job Launcher portlet allows a user to submit jobs to a Grid environment using the Globus GRAM protocol. For this the user must have a valid GSI Proxy Certificate which can be loaded through the Proxy Manager Portlet. GIS - Grid Information Service GIS is part of the Globus Toolkit used to manage resources information.

4. Scheduling

4.1 Resource Broker

Whenever the user wants to execute the application, resource broker is contacted for retrieving the resource provider's address. The resource broker will maintain a policy regarding the acceptance of application based on the cost criteria. Once it accepts the application, a global queue is maintained. The resource provider will provide the complete list of all machines available in grid. Since the grid is a dynamic environment, the broker will watch over the changes in environment and keeps updating.

The algorithm behind the global queue is Max-Min-Max. This algorithm is used for selecting an application from the global queue for allocation .Max-Min [1] is a static algorithm gives highest priority to largest application, whereas Min-Min [1] is a static algorithm which gives highest priority to shortest application. In order to avoid starvation, both algorithms are coupled. This proposed algorithm is called Max-Min-Max algorithm (Since preference is given to max-min algorithm, the name is max-min-max instead of min-max-min) which is described below.

Step 1: Start.

Step 2: Take the execution time of all the application in the queue. Let the number of applications be n.

Step 3: Compute the average execution time (E) that is E= (∑ execution time)/n.

Step 4: All application in the queue that has their execution time above E are considered as largest application. (L).

Step 5: All application in the queue that has their execution time below E are shortest application (S).

Step 6: If n (L)>=n(S) then start with max-min algorithm

Step 7: min-min and max-min algorithm will consecutively alternate it.

Step 8: Else start with min-min algorithm

Step 9: max-min and min-min algorithm will consecutively alternate it.

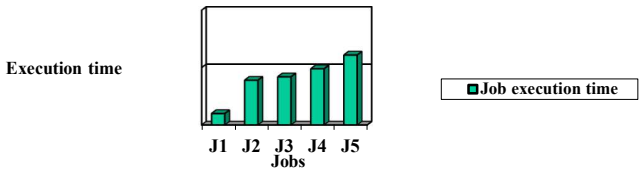
Step 10: If there are next set of applications go to step2 and repeat the steps 2-9.

Step 11: Else stop.

4.2 Algorithm Analysis (Max-Min-Max):

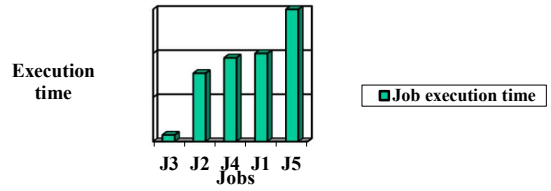
Table II: Arrival of Jobs

Jobs	Execution time
J1	10
J2	29
J3	3
J4	7
J5	12



First Come First Serve algorithm:

Average waiting time = (0+10+39+42+49)/5 = 28 milliseconds



Max-Min-Max algorithm:

Average waiting time = (0+3+31+38+40)/5 = 22.4 milliseconds

4.3 Provider publish to Broker

Resource provider publish themselves to Broker with their resources and declare the type of applications that it can execute.Provider consists of group of clusters. When many resource providers are capable of executing same type of application , then Broker will break the tie using parameters like free nodes, reliability of nodes and system properties like processor speed, processor load and memory.

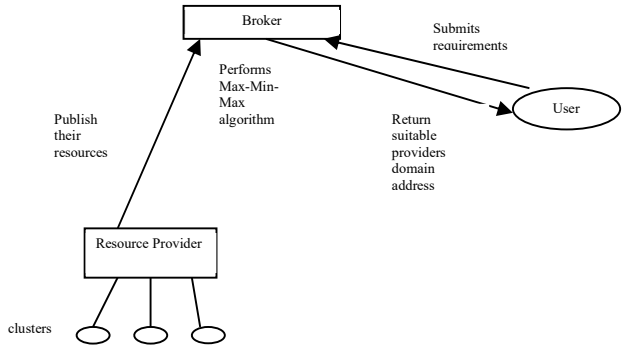


Figure 2: Requirement submission

4.4 Resource Provider (RP)

4.4.1 Sufferage

Customer now contains the application and provider's address. Within the provider which one is least suffered machine is selected using sufferage algorithm. The metrics considered for this algorithm are CPU speed, free memory, cache size, task size and load average [8]. The result of this algorithm is ordering of less suffering machines within a provider. The algorithm used for scheduling purpose is FPLTF (Fast Processor Largest Task First).

Step 1: Start

Step 2: Get the CPU speed, free memory, cache size, and load average of all machine under a selected provider.

Step 3: Calculate the completion time using formula,

Completion time = TBA + suffering time. (1)

Suffering time = Task size / (CPU speed * (free memory + cache size) * (1- load average ratio)) (2)

Where, TBA = time for that node to be available.

Step 4: The customer gives the application in terms of number of tasks along with their sizes. The task with largest size is selected first for scheduling hence framing out Fastest Processor Largest Task First algorithm.

Step 5: Least completion timed node is selected to execute the application.

Step 6: Stop.

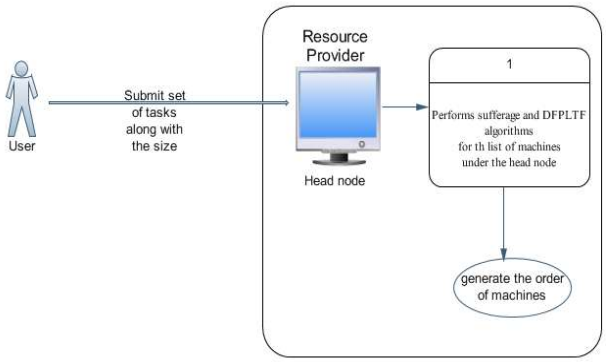
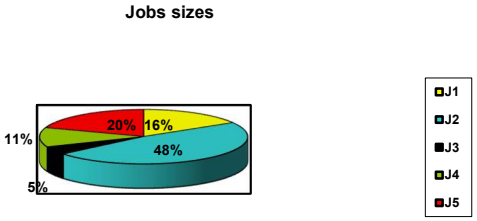


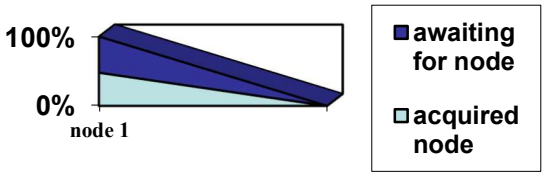
Figure 3: Application Scheduling and Execution

Suffering time is directly proportional to task size and inversely proportional to CPU speed, free memory, cache size and load balance. If the processor has more speed, more free memory and cache, more load balance (1-load average) then suffering time for that node to execute the given task is less. If the preferred node is not available or too busy, then head node will allocate the task to next preferred node.

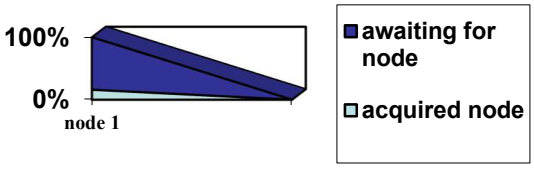
4.4.2 Algorithm Analysis (FPLTF)



Fastest Processor Largest Task First



First In First Out



- considering one node for five jobs to execute.

5. Job Migration

The resource provider generates the order in which the tasks have to be executed. As per that, the head node directs the tasks to respective sub-nodes in the order. If any sub-node fails to execute the tasks, the tasks will be migrated to other nodes in the cluster, in such away the tasks are replicated by the head node. The replication is performed only for the application that requires high security. To workout safe, the scheduler (head node) duplicates the job to be executed at multiple sites and stop all these replicas execution, once one of the replicated jobs is successfully carried out.

6. Conclusion

Thus a decoupled scheduling approach is considered for parallel applications in a computational grid environment. Moreover, an exhaustive search of machines in the grid for a similar kind of applications is also reduced, which will lead to less time consumption. And also the performance is evaluated based on the execution time, processor speed, memory capacity, bandwidth, and cache size and load average. Thus the application has been decoupled from scheduling and also dynamic information is exploited at run time for improved scheduling.

7. Results and Discussion

A pre-scheduling concept is used, since the application modules are not revealed to the broker. As per that, the limitation lies here is searching and retrieving more databases. Moreover, when the application comes for the first time, tedious search process will occur, which in turn consumes time. Thereafter when the application comes, this search process can be avoided using pre-schedule

References

[1] "Risk-Resilient Heuristics and Genetic Algorithms for Security-Assured Grid Job Scheduling " Shanshan Song, Kai Hwang, Fellow, IEEE, and Yu-Kwong Kwok, Senior Member, IEEE transactions on computers, vol. 55, no. 6, June 2006 .

[2] "A Decoupled Scheduling Approach for the GrADS Program Development Environment" Holly Dail, Henri Casanova and Fran Berman, IEEE 2002.

[3] "Grid Brokers and Meta schedulers Market Overview" Ilona Gaweda and Chris Wilk, Feb 2006.

[4] "Dynamic Scheduling in Grid Systems" Maria Chtepen ,sixth FirW PhD Symposium,Faculty of Engineering ,Ghent University,30th November 2005-paper nr.110.

[5] "Operating Systems Concepts" Abraham Silberschatz and Peter B. Galvin, fourth Edition. "The Anatomy of the Grid" Ian Foster, Carl Kesselman and Steven Tuecke.

[6] "The Physiology of the Grid-An Open Grid Services Architecture for Distributed Systems Integration" Ian Foster, Carl Kesselman, Jeffrey M. Nick and Steven Tuecke.

[7] "Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids" Daniel Paranhos da Silva, Walfredo Cirne, Francisco Vilar Brasileiro