July 2012

Priority Based Mathematical Modeling for Grid Computing Environment

Abhinandan Datta

Information Technology Department, IEM, Saltlake, Kolkata, India, abhinandan_datta@yahoo.com

Imon Banerjee

Information Technology Department, IEM, Saltlake, Kolkata, India, monban@gmail.com

Follow this and additional works at: <https://www.interscience.in/uarj>

Part of the [Business Commons](#), [Education Commons](#), [Engineering Commons](#), [Law Commons](#), [Life Sciences Commons](#), and the [Physical Sciences and Mathematics Commons](#)

Recommended Citation

Datta, Abhinandan and Banerjee, Imon (2012) "Priority Based Mathematical Modeling for Grid Computing Environment," *Undergraduate Academic Research Journal*: Vol. 1 : Iss. 1 , Article 26.

DOI: 10.47893/UARJ.2012.1025

Available at: <https://www.interscience.in/uarj/vol1/iss1/26>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in Undergraduate Academic Research Journal by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Priority Based Mathematical Modeling for Grid Computing Environment

Abhinandan Datta & Imon Banerjee

Information Technology Department, IEM, Saltlake, Kolkata, India
E-mail : abhinandan_datta@yahoo.com, monban@gmail.com

Abstract - Grids are a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting together to perform very large tasks. This technology has been applied to computationally intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back office data processing in support for e-commerce and Web services. One of the main motivations of grid computing is resource sharing. Our main goal of this paper is to propose a new model that can assign priority of each user level job. We basically focused on mathematical modeling and pseudo coding of this system. Further we would like to simulate this environment using GridSim and compare this system with the existing one.

Keywords - Grid computing, priority based modeling, system modeling, Heirarchical modeling.

I. INTRODUCTION

The Basic idea between Grid Computing[3] is to utilize the ideal CPU cycles and storage of millions of computer systems across a worldwide network function as a flexible, pervasive, and inexpensive accessible pool that could be harnessed by anyone who needs it, similar to the way power companies and their users share the electrical grid. An ambitious and exciting global effort to develop an environment in which individual users can access computers, databases and experimental facilities simply and transparently, without having to consider where those facilities are located. [RealityGrid, Engineering & Physical Sciences Research Council,UK 2001] <http://www.realitygrid.org/information.html>

Grid computing[2][1] is focused on the ability to support computation across multiple administrative domains that sets it apart from traditional distributed computing. It offers a way of using the information technology resources optimally inside an organization involving virtualization of computing resources. Its concept of support for multiple administrative policies and security authentication and authorization mechanisms enables it to be distributed over a local, metropolitan, or wide-area network.

All the register users of this environment can submit multiple jobs to the shared resources by using Global grid resource broker. This entity is working as a communicator between user and resource owner. Users

submit the jobs to the resource broker, then resource broker allocates the jobs to the appropriate resource required by the user.

Our main motivation of work is to model an environment which can assign priority to each job from the user side and describe the mathematical model of the environment. This concept is mainly used to,

1. Minimized the conflict of multiple jobs for a single resource.
2. Minimized the workload of the Grid resource broker.
3. Optimized resource sharing benefit by getting the maximum payback for each resource every time.

II. MODEL OF THE GENERAL GRID RESOURCE SHARING ENVIRONMENT

Grid computing concerns the application of the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data[4][5]. Grid computing appears to be a promising trend in Resource Sharing for three reasons:

- 1) Its ability to make more cost-effective use of a given amount of computer resources,

- 2) It is a way to solve problems that can't be approached without an enormous amount of computing power.
- 3) It suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as collaboration toward a common objective.

Fig. 2 : Grid design for Resource Sharing

III. PROPOSED MODIFIED MODEL USING PRIORITY SETTING METHOD

The proposed system has Multiple Jobs (J) at the user side where single user can create single, multiple or no jobs at each time. These users have their own User Side System (USS) that describe the OS, processor, memory and other system configuration of the each user which could be different for each login for single user. As the user can be login by using his personal account from his own system or some other system. Each job has their own id, Time of submission, Deadline, resources requested, id of the user who has set the job and the budget offered from the user for resource allocation.

Multiple jobs are submitted by the user (U) with their unique id, user side system (USS) id, and the id of the User Side Priority Manager (USP1) which sets the priority in which they are allocated resources. The User Side Priority Manager is divided into 2 levels. Level 1 (USP1) allots priorities to conflicting jobs on its own domain, while the Level 2 priority manager does the same for all the jobs received in the System domain. The Priority index tables list the jobs received according to their priority.

These jobs are submitted to the Grid Broker (GLO) who lists them based on their priority and then sends them accordingly to the Local Broker Manager (L) for allotment of the resources to the jobs. The Local Broker Manager (L) further distributes the jobs to the Local Brokers (LB) of each domain. Each Local Broker allocates the requested resources to the jobs based on the budget specified. Each resource performs the operations requested by the job, and sends back the output to the

User (U) and the next job in queue is allocated and so on.

Fig. 3 : Modified Grid design for Resource Sharing(Priority based)

For this environment, some faulty or exceptional cases like resource failure, resource being busy with some other internal jobs that prevents the resource to be allocated to the global jobs can be occurred. Then we suggest adding an entity, the Fault/Exception Manager (FM) which allocates the job to some other available resources. If even one resource cannot be allocated to a job sent by the Local broker, ONLY then the then the Fault/Exception Managers perform their operations. This is done by a bit in the Local Broker tuple, called *Exception_handler_check*, whose value, if 1, means there is a presence of an inability of resource allocation to a job, & hence FE comes into play. Similarly, FE tuple also has a bit called *L_chk_result* whose result, if 0, means the Exception handler need not be used, and if 1, its to be used.

The proposed system is entirely based on the understanding, that, when the Jobs are sent by the User, the resources needed to be allocated for the completion of the job are all predefined by the user.

IV. MATHEMATICAL MODELING FOR THE PROPOSED ENVIRONMENT

A mathematical model is a description of a system using mathematical concepts and language. The process of developing a mathematical model is termed mathematical modeling. In many cases, the quality of a scientific field depends on how well the mathematical models developed on the theoretical side agree with

results of repeatable experiments. Lack of agreement between theoretical mathematical models and experimental measurements often leads to important advances as better theories are developed.

The representation of each of the entity of this model can be mathematically represented as follows:

$$1) \text{ Job } (J_i) = J_i(id, T_{sub}, T_{deadline}, R[p][q], user_id, budg)$$

Where id = Job unique identification number

T_{sub} = Time of submission by the user

$T_{deadline}$ = Deadline of Job set by the user

$R[p][q]$ = Required resource array:

$$\begin{matrix} & \mathbf{C} & \mathbf{D} & \mathbf{S} \\ \begin{bmatrix} C_a & D_b & S_c \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} & & & \end{matrix} \quad \mathbf{p \times q}$$

Where the value of $q = 3$, as we have considered 3 types of resources.

C_a represents Computational resources,

D_b represents Data resources,

S_c represents Service resources requested by the Job J_i .

$p = \max(\text{No of } C_a, \text{No of } D_b, \text{No of } S_c)$

$R[p][q] = 0$, iff user does not need that resource of id $R[p][q]$.

$user_id$ = the unique id of the user from whom the job has originated.

$budg$ = the cost allocated to the job

$$2) \text{ User Side System}(USS) = USS_i(user_side_system_id, OS, memory, processor_req, Job[J_1, J_2, \dots J_n])$$

Where:

$user_side_system_id$ = the unique id of the User Side System which will interact with the User

OS = Operating System being used in the User Side System

$memory$ = amount of memory currently used by the user

$processor_req$ = The minimum processor frequency used by User U_i currently

$Job[n] = Job[J_1, J_2, \dots J_n]$ The array of Jobs requested to be processed by the User i

$$3) \text{ User}(U_i) = U_i(user_id, Priority_Manager_id, user_side_system_id, User_SS[USS_1, USS_2, \dots USS_n])$$

Where :

$user_id$ = the unique id of the user from whom the job has originated.

$Priority_Manager_id$ = the unique id of the Priority Manager which will assign priorities to the jobs allocated by the User

$user_side_system_id$ = the unique id of the User Side System which will interact with the User

$User_SS[n] = User_SS[USS_1, USS_2, \dots USS_n]$ The array of User Side Systems which will interact with the user and accept the jobs. The User i can access any User Side System and supply jobs to be processed from anywhere.

$$4) \text{ User side Priority Manager Level 1 } (USP1_i) = USP1_i(USP1_id, User[U_1, U_2, \dots U_n])$$

Where

$USP1_id$ = the unique id of the User side Priority Manager level 1 that will assign priority to the jobs sent by the User

$User[n] = User[U_1, U_2, \dots U_n]$ is the list of Users that get their jobs prioritized by the User side Priority Manager Lv 1

$$5) \text{ User side Priority Manager Level 2 } (USP2) = USP2(UserSP1[USP1_1, USP1_2, \dots USP1_n])$$

Where:

$UserSP1[n] = UserSP1[USP1_1, USP1_2, \dots USP1_n]$ the list of User side Priority Manager Level 1s whose jobs get prioritized and sent to the GLO

$$6) \text{ Grid Information Server}(GIS) = GIS(G_id, Job[J_1, J_2, \dots J_n])$$

Where:

G_id = The unique Job ID of each Jobs sent to the Grid Broker

$Job[J_1, J_2, \dots J_n]$ = Stored job in the grid information server which have been submitted but not finished execution.

$$7) \text{ Local Broker Manager } (L) = L(Local_broker_id, Grid_jobs[J_id_1, J_id_2, \dots J_id_n], FE[FE_1, FE_2, \dots FE_n], LB[LB_1, LB_2, \dots LB_n])$$

Where:

$Local_broker_id$ = The id of the local broker to whose domain the selected $Grid_jobs[]$ are being sent

Grid_jobs[n] = Grid_jobs[J_id₁, J_id₂, ... J_id_n] are the jobs allocated to the Local Broker Manager by the Grid Broker to be distributed to the required resources.

J_id_i = It is the id of the individual job submitted by the user

FE[n] = FE[FE₁, FE₂, ... FE_n] The list of all the Fault/Exception handlers present in the System

LB[n] = LB[LB₁, LB₂, ... LB_n] The list of all the Local Brokers from each of the Domains

8) **Fault/Exception Handler (FE_i) =**
FE_i(FE_id, L_chk_result, Cs[a], Ss[b], Ds[c], JC[x₁],
JS[x₂], JD[x₃])

Where:

FE_id = the id of the Fault/Exception handler

L_chk_result = the result of Exception_handler_check of the Local Broker Manager. If it is 1, only then the Exception handler is accessed

Cs[a] = An array that consists of all the Computational Resources that are not being allocated to any jobs sent by the Grid broker and they are either under maintenance or used in their internal network

Ss[b] = An array that consists of all the Service Resources that are not being allocated to any jobs sent by the Grid broker and they are either under maintenance or used in their internal network

Ds[c] = An array that consists of all the Data Resources that are not being used by any jobs sent by the Grid broker and they are either under maintenance or used in their internal network

JC[x₁] = An array that consists of jobs distributed among the Computational Resources while functioning in their internal network

JS[x₂] = An array that consists of jobs distributed among the Service Resources while functioning in their internal network

JD[x₃] = An array that consists data obtained by Data Resources while functioning in their internal network

9) **Local Broker (LB_i) = LB_i(LB_id, domain_name,**
Exception_handler_check, Grid_Jobs[G_id₁,
G_id₂, ... G_id_n], C's[a], S's[b], D's[c],)

Where:

Exception_handler_check = 0 if the resources are all allowed to be allocated to the jobs listed in the Grid Information Server; and 1 if even any one of the resources cannot be used by the Grid Information Server.

C's[a] = An array that consists of all the Computational Resources that can be allocated to any jobs sent by the Grid broker.

S's[b] = An array that consists of all the Service Resources that can be allocated to any jobs sent by the Grid broker.

D's[c] = An array that consists of all the Data Resources that can be used by any jobs sent by the Grid broker.

LB_id = the id of the Local Broker of that specific domain

Grid_Jobs[n] = Grid_Jobs[G_id₁, G_id₂, ... G_id_n] The Jobs sent to the Local Broker by the Local Broker Manager to be allocated to the resources

domain_name = name of the domain of the Local Broker

10) **Data Resources(D_i) =**
D_i(D_id, LB_id, domain_name, cost)

Where

D_id = the id of the Data Resource

LB_id = the id of the Local Broker of that specific domain to which the resource is present

domain_name = name of the domain of the Local Broker

cost = the cost of allocation of this resource by a job

11) **Computational Resources(C_i) =**
C_i(C_id, LB_id, domain_name, cost_per_sec)

Where:

C_id = the id of the Computational Resource

LB_id = the id of the Local Broker of that specific domain to which the resource is present

domain_name = name of the domain of the Local Broker

cost = the cost of allocation of this resource by a job per second

12) **Service resources (S_i) =**
S_i(S_id, LB_id, domain_name, cost_per_sec)

Where

S_id = the id of the Service Resource

LB_id = the id of the Local Broker of that specific domain to which the resource is present

domain_name = name of the domain of the Local Broker

cost = the cost of allocation of this resource by a job per second

13) Grid Broker(GLO) = GLO(GIS,L,USP2)

Where:

GIS = Grid Information Server

L = Local Broker Manager

USP2 = User Side Priority Manager Level 2

V. HIERARCHICAL REPRESENTATION OF THE MODEL

As pre our model of assigning priority to each of the submitted jobs, we can represented this above mentioned model in the hierarchical from.

Entity of each level of this tree submitted jobs to the upper level entity, at the time submission. Then the upper level entity either marked the job with some priority or allocates the job to the resources.

Fig. 3 : Hierarchical model of the environment

VI. PSEUDO CODE OF PRIORITY ASSIGNING ALGORITHM

A. Algorithm for User site priority manager level 1:

The algorithm for Priority scheduling for the selection of conflicting jobs on the priority basis for the Priority Index Table at level 1 User Side Priority Manager can be assigned the priority as follows where p and q denotes the row and columns of the Resource matrix in Job (J) tuple.

- 1) Begin
- 2) If no conflicting jobs are encountered Then $J_i (U_i, \text{priority}) = 0$
- else For $i = 0$ to $n-2$ do

```

For a = 0 to p do
  For b = 0 to q do
    If  $J_i(U_i, R[a][b]) = J_{i+1}(U_{i+1}, R[a][b])$  Then do
      If  $J_i(U_i, \text{budg}) < J_{i+1}(U_{i+1}, \text{budg})$  Then do
         $J_{i+1}(U_{i+1}, \text{priority}) = i$ 
      else if  $J_i(U_i, \text{budg}) > J_{i+1}(U_{i+1}, \text{budg})$  Then do
         $J_i(U_i, \text{priority}) = i$ 
      else do
        If  $[T_{\text{sub}}(J_{i+1}, U_{i+1}) - T_{\text{Deadline}}(J_{i+1}, U_{i+1})] < [T_{\text{sub}}(J_i, U_i) - T_{\text{Deadline}}(J_i, U_i)]$  Then do
           $J_{i+1}(U_{i+1}, \text{priority}) = i$ 
        else If  $[T_{\text{sub}}(J_{i+1}, U_{i+1}) - T_{\text{Deadline}}(J_{i+1}, U_{i+1})] > [T_{\text{sub}}(J_i, U_i) - T_{\text{Deadline}}(J_i, U_i)]$  Then do
           $J_i(U_i, \text{priority}) = i$ 
        else set random priority to both the jobs
      end If
    end If
  next b, end For
next a, end For
next i, end For
    
```

end If

3) End

B. Algorithm for User site priority manager level 2:

The algorithm for Priority scheduling for the selection of conflicting jobs on the priority basis for the Priority Index Table at level 2 User Side Priority Manager can be stated as follows .

- 1) Begin
- 2) If no conflicting jobs are encountered Then $J_i (U_i, \text{USS}_i, \text{priority}) = 0$
- else For $i = 0$ to $n-2$ do


```

For a = 0 to p do
  For b = 0 to q do
    If  $J_i(U_i, \text{USS}_i, R[a][b]) = J_{i+1}(U_{i+1}, \text{USS}_{i+1}, R[a][b])$  Then do
      If  $J_i(U_i, \text{USS}_i, \text{budg}) < J_{i+1}(U_{i+1}, \text{USS}_{i+1}, \text{budg})$  Then do
         $J_{i+1}(U_{i+1}, \text{USS}_{i+1}, \text{priority}) = i$ 
      else if  $J_i(U_i, \text{USS}_i, \text{budg}) > J_{i+1}(U_{i+1}, \text{USS}_{i+1}, \text{budg})$  Then do
         $J_i(U_i, \text{USS}_i, \text{priority}) = i$ 
      else do
        If  $[T_{\text{sub}}(J_{i+1}, U_{i+1}) - T_{\text{Deadline}}(J_{i+1}, U_{i+1})] < [T_{\text{sub}}(J_i, U_i) - T_{\text{Deadline}}(J_i, U_i)]$  Then do
           $J_{i+1}(U_{i+1}, \text{USS}_{i+1}, \text{priority}) = i$ 
        else If  $[T_{\text{sub}}(J_{i+1}, U_{i+1}) - T_{\text{Deadline}}(J_{i+1}, U_{i+1})] > [T_{\text{sub}}(J_i, U_i) - T_{\text{Deadline}}(J_i, U_i)]$  Then do
           $J_i(U_i, \text{USS}_i, \text{priority}) = i$ 
        else set random priority to both the jobs
      end If
    end If
  next b, end For
next a, end For
            
```

```

         $J_i (U_i, USS_i, priority) = i$ 
    else set random priority to both the jobs
end If
    else  $J_i (U_i, USS_i, priority) = 0$ 
    next b, end For
next a, end For
next I, end For
end If
3) End

```

VII. CONCLUSION & FURTHER WORK

Here we have proposed a new priority based general Grid architectural model for resource sharing of different type such as data resource, computational resource & service resource.

This approach maximized the resource owner benefit by using priority assignment approach and it also handle the fault by using the fault handler.

In a Grid environment, it is hard and even impossible to measure performance of a scheduling algorithm. Investigating resource management and scheduling algorithm in Grid computing environments need a testbed infrastructure but there are no ready-to-use testbed infrastructures, building such a thing is expensive and time consuming. To overcome this limitation, Java-based Grid Simulation toolkit, called GridSim can be use for simulating scheduling algorithm. Further work will be to simulate our resources sharing and scheduling algorithm using GridSim[6].

REFERENCES

- [1] Foster I, Kesselman C (eds). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann: Los Altos, 1999; 259–278.
- [2] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling virtual organizations. International Journal of High
- [3] Foster I, Kesselman X, Nick J, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum, 22 June 2002; 1–5.
- [4] Balachandar R. Amarnath, Thamarai Selvi Somasundaram, Mahendran Ellappan I and Rajkumar Buyya, Ontology-based Grid resource management, Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.942.
- [5] Foster I, Kesselman C (eds). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann: Los Altos,
- [6] <http://www.buyya.com/gridsim/>