Job Scheduling Algorithm in Windows Grid

Akash Bhatt[#1], Nitin Pandya[*2]

[#1] *M.E. in Department of Computer Engineering (Wireless and Mobile Computing) GTU PG School, GTU, Ahmedabad*
[1]akashbhatt2010@gmail.com

[*2] *Assistant Professor, Department of Computer Engineering, Shankarsinh Vaghela Bapu Institute of Technology, Gandhinagar*
[2]nitinpandya.85@gmail.com

Abstract—Modern Science and technologies are collaborate with each other to achieve some different goals, performance and greater impact all over the globe. These collaborations are multidisciplinary and geographically distributed environments. With the help of this kind of environment we can utilize the unutilized resources as maximum and implement a different grid computing environment. A computational Grid is a combination of hardware and software infrastructure that provides dependable, reliable and inexpensive access to high end computational capabilities. Computational grid is used by a user to solve large scale problem by spreading a single large computation across multiple machines of physical location. Because of this, grids are basically dynamic, shared and distributed environments, handling these kinds of platform efficiently are extremely complex. One of the major problems is scheduling. Scheduling in grid environment is very important to achieve effective utilization of the available resources. So, the job scheduling algorithm for windows grid computing which is going to implement by us is provides reliability, minimize execution time and increase throughput using our Grid toolkit Alchemi and our proposed scheduling algorithm. Our goal to establish this research is to explore grid computing in such way that the available resources are utilized as much as possible and also we are implementing job scheduling algorithm which can reduce the execution time as well as the reduce cost of execution to completion. Also we can use this research for windows environment which can help much powerful way to industry or organization which are using grid computing environment.

Keywords—*Grid computing, Alchemi framework, Job scheduling Algorithm, Computational Grid*

I. INTRODUCTION

Grid computing is an interesting research area that integrates geographically distributed computing resources into a single powerful system. There are many applications can benefit from such integration. Examples are collective applications, remote visualization and the remote use of scientific instruments.

Grid computing is the utilization of the "unused" capabilities of a large [2], ad-hoc collection of compute nodes distributed across a network. The basic premise of a grid is that compute nodes work together to achieve a common goal.

Grid computing is a kind of parallel computing that enables the sharing, selection, and aggregation of geographically distributed autonomous resources, at runtime, as a function of availability, capability, performance, cost, and users quality-of-service requirements. One of the services that a Grid can provide is a computational service. Computational services execute jobs in a distributed manner [1]. A Grid providing computational service is often called Computational Grid.

Research efforts have focused on the development of Grid computing, a fundamentally new set of technologies that create large-scale distributed computing systems by interconnecting geographically distributed computational resources via very high performance networks. The advanced applications being developed to execute in Grid environments include distributed collaboration across the Access Grid, remote visualization of terabyte (and larger) scientific data sets, large-scale scientific simulations, Internet telephony, and multimedia applications.

In grid environments, there are three types of schedulers to meet different performance goals. Resource schedulers coordinate user requests for accessing a given resource to ensure fairness and to optimize utilization. Application schedulers promote the performance of individual applications by optimizing performance measures such as execution time and speedup.

Job schedulers aim to optimize the overall performance of a system, e.g., minimizing the average job response time and maximizing the number of jobs executed in certain period of time.

A job is typically divided into sub jobs which are assigned to different machines on a computational grid for execution. The actual execution time of a job is determined by the way of processor allocation, namely, the number and sizes of the sub jobs. Hence, the processor allocation strategy plays an important role in a job scheduler for grid applications. However, it is still not clear how to model the effect of inter-machine communications, so that the actual execution time of a job can be determined when processor allocation is given.

Types of GRID:

(1) **Computational Grid**: This grid is used to allocate resources specifically for computing power. The resources are usually high-performance machines.
(2) **Data Grid**: This grid is used for housing and providing access to data over multiple organizations.

(3) **Scavenging Grid**: It is one type of Computational grid which uses the unused resources for computational power. It steals CPU cycles when CPU is idle, so it is also called Cycle Scavenging Grid.

(1) **Computational Grid:**
A Computational Grid consists of a set of resources, such as computers, networks, online instruments, data servers or sensors that are tied together by a set of common Services which allow the users of the resources to view the collection as a seamless computing/information environment [3].

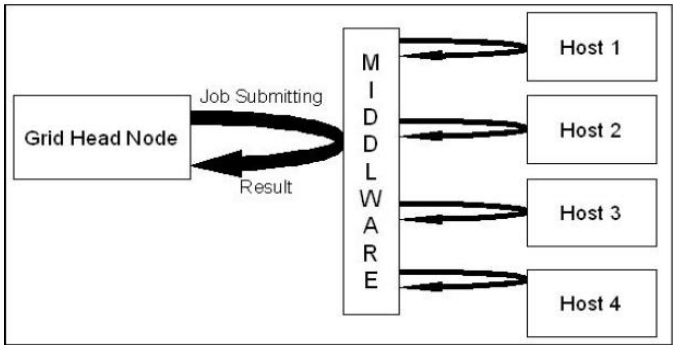

Fig. 1: Grid Computing Environment [3]

A grid also allows a single large computation to be spread across several machines, each of which is executing some portion of the computation. This can be done by:
- Breaking up the tasks into smaller tasks.
- Assigning the smaller tasks to multiple hosts to work on simultaneously, coordinating with each other.

Grid Layered Architecture:
Grid layered architecture defines how the grid layered works with different strategies.
 Fabric Layer: Interfaces to local control, including physical and logical resources such as files, or even a distributed file system.

 Connectivity Layer: Defines core communication and authentication protocol supporting Grid-specific network transactions.

 Resource Layer : Allows the sharing of a single resource and it builds on connectivity layer communication and authentication protocols to define protocols for secure negotiation, initiation, monitoring, and control of sharing operations on individual resources.

Collective Layer: Allows resources to be viewed as collections and sharing of resources. The Collective layer contains protocols and services not associated with any one specific resource but instead capturing interactions across the collections of resources.

 Application Layer: Use the appropriate components of each layer to support the application.

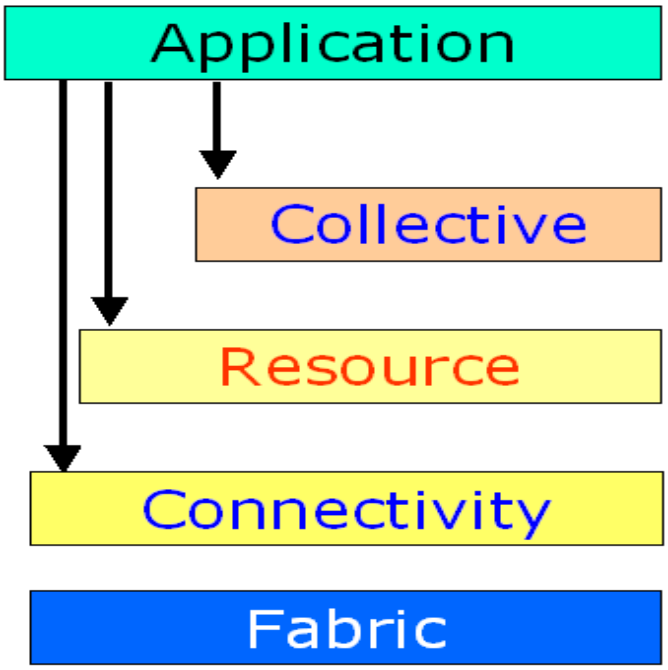

Fig. 2: the layered Grid Architecture

Each of these layers may contain protocols, APIs, and software Development Kits (SDKs) to support the development of Grid applications.

II. Problem Statement

The main aim of Grid job scheduling is to increase the system throughput, decrease the execution time of jobs and to satisfy the job requirements from the available resources. Based on the literature review in current grid job scheduling algorithms, we find the limitations like:
 Use of Available Resource utilization is minimum

 System throughput is limited

 does Not Provide such Quality of service

 Require more Processing time

 Do Not Provide much Reliability
Using our new proposed scheduling algorithm we will try to minimize such problems as much as possible and provide better result.

III. Theoretical Background and Observations

Job Submission methods
Batch and Interactive Jobs

Generally, there are two kinds of job submission methods: batch execution and interactive execution. When a batch job [4] is submitted, it is submitted to a queue, where it waits until it reaches the top of the queue and the required resources become available.

An interactive job is a job where the nodes are reserved for users to log into an issue commands by hand. An interactive job is run immediately upon submitting if the specified resource is available. Since the interactive jobs never wait, and all the input and output are handled transparently, it is very useful for testing batch scripts and debugging programs. Although interactive jobs may decrease the utilization of the system, it is vital for a working scientific computing system.

Job Model

Jobs are submitted by independent users at the local sites. The scheduling problem is an on-line scenario without any knowledge of future job submissions. Our evaluations are restricted to batch jobs, a job request consists of several parameters as e.g. the requested run time and the requested number of resources. After submission a job requests a fixed number of resources that are necessary for starting the job. This number is not changed during the execution of the job. It is the task of the scheduling system to allocate the jobs to resources and to determine the starting time. The jobs are executed without further user interaction.

Basic Architecture of Grid

A grid application [9] is defined simply as an application that is to be executed on a grid and that consists of a number of grid threads. Grid applications and grid threads are exposed to the grid application developer via the object-oriented Alchemi .NET API.

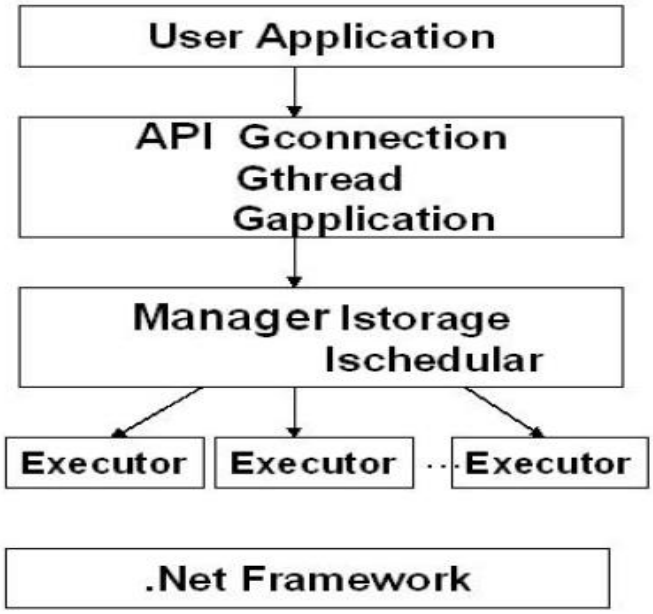

Fig. 3: Basic Architecture of Grid

IV. PROPOSED SYTEM AND SIMULATION ENVIRONMENT

Alchemi is a .NET based grid computing framework that provides run time machinery and programming environment required to construct the desktop grid and developed the grid application. It allows us flexible application composition by supporting the object-oriented grid programming model. Cross platform support is provided by Web- services and flexible execution model supports dedicated execution by grid nodes [6].

Alchemi Framework

The aim of Alchemi grid computing framework is not to develop the grid software as easy as possible but flexible, scalable, reliable, and extensible. The key features of the Alchemi are,

- Internet based cluster computing for desktop computer without a shared file system.
- Dedicated execution by cluster and individual nodes.
- Object-oriented grid thread programming model.

Layered Architecture of Grid Using Alchemi framework:

A grid is created by installing Executors on each machine that is to be part of the grid and linking them to a central Manager component. The Windows installer setup that comes with the Alchemi distribution and minimal configuration makes it very easy to set up a grid users can develop, execute and monitor grid applications using the .NET API and tools which are part of the Alchemi SDK. Alchemi offers a powerful grid thread programming model which makes it very easy to develop grid applications and a grid job model for grid-enabling legacy or non-.NET applications.

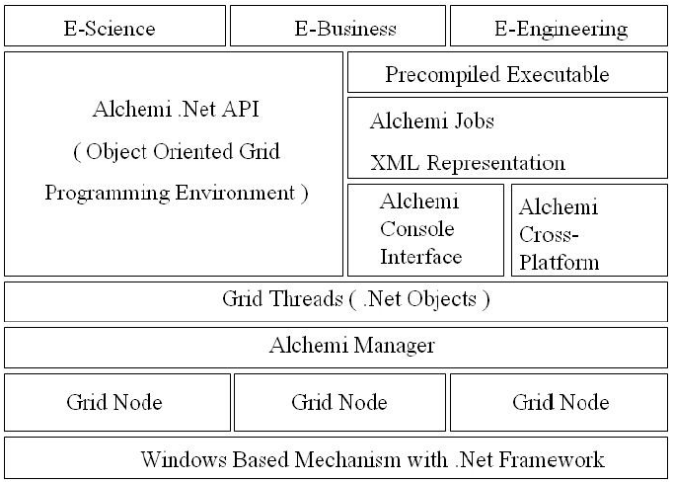

Fig. 4: Layered Architecture of Grid using Alchemi [6]

Proposed Technique:

A typical Grid consists of a number of services, applications and a number of physical resources, including compute resources that are capable of introducing these services as well

as storage resources, network resources etc. The job of the scheduler is to select a set of resources on which to schedule the tasks of an application, assign application tasks to compute resources, coordinate the execution of the tasks on the compute resources and manage the data distributions and communication between the tasks.
In this mechanism we can identify the historical information and set the scheduling of resources based on their avability. And after that we can pass thread to executor for their execution and after completion of task we can get the result.

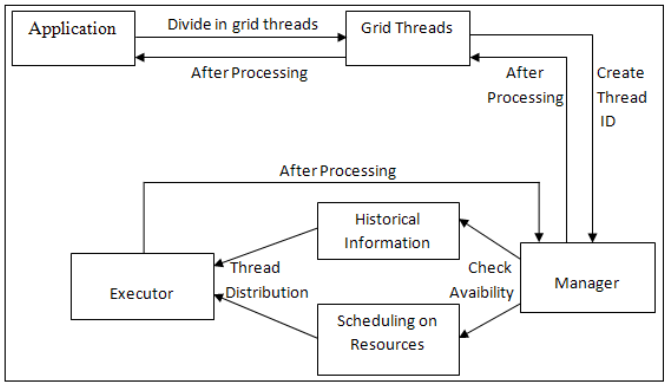

Fig. 5: New Scheduling Mechanism

Based on this scheduling mechanism we can try to find out the solution of our problem and we can get the better performance than existing mechanism using proposed algorithm and techniques.

Proposed Algorithm
Begin:
(1) First find all available resources
(2) Check Success Rate of Resources
(3) Find Jobs or Request from user
Divide Jobs into sub jobs or threads using Divide & Conquer Algorithm and calculate the percentage.
(4) Assign priority to sub jobs or threads and also check that the higher priority jobs in not depend on lower priority jobs
(5) Initially,
St=start; En=end, T=total, e=executor, P=problem, C=completed job, R=running job, I=incomplete job;
for (e=1; e<=n; e++) //no of executor
{
for (j=1; j<=n; j++) //priority check
{
If (e==idle)
{
If (e capacity==priority of job size)
{
Assign job to e;
}}}}
(6) if (e==idle)
{
Assign job j1, j2, … , jn using FCFS // Apply FCFS
}
(7) if (Any remaining incomplete jobs)
{
goto step 5;
}
(8) Merge answer of j1,j2,j3,…,jn;
(9) Now, Note End time
(10) Total Execution time= End time – Start time
(11) Stop.

Algorithm Description
According to our algorithm and the Grid strategy the Failure of a resources while doing scheduling is not being considered at the time of allocating resources by the scheduler to the executor. Here, at the time of scheduling jobs, scheduler will consider only minimum cost of a resource along with MIPS of that resource. While doing scheduling, if resource fails to execute any job then such thing cannot be ignored when next time a job needs to be executed on that resource. So for a resource a new parameter is added as failure rate which will consider success rate of a resource. If a resource is having 100% failure rate then that means that whenever a job is scheduled on that resource then it will surely fail to execute that job on that resource. By default the value for failure rate should be zero. A history is maintained by the scheduler, which will keep track of this failure rate. If any failure happens then manager will update history of that resource, which will be considered at the time of next scheduling of jobs.

V. Results and Analysis

Alchemi Framework:
For our grid implementation purpose we can use Alchemi framework. Alchemi is a .NET based grid computing framework that provides run time machinery and programming environment required to construct the desktop grid and developed the grid application. The aim of Alchemi grid computing framework is not to develop the grid software as easy as possible but flexible, scalable, reliable, and extensible.
Performance Results:

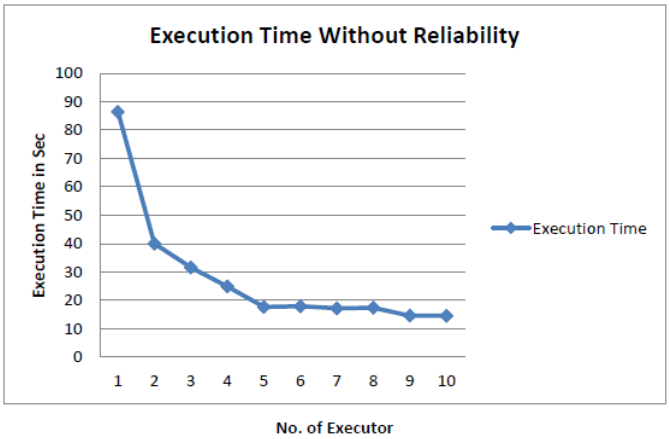

Fig. 6: Graph without Reliability Mode

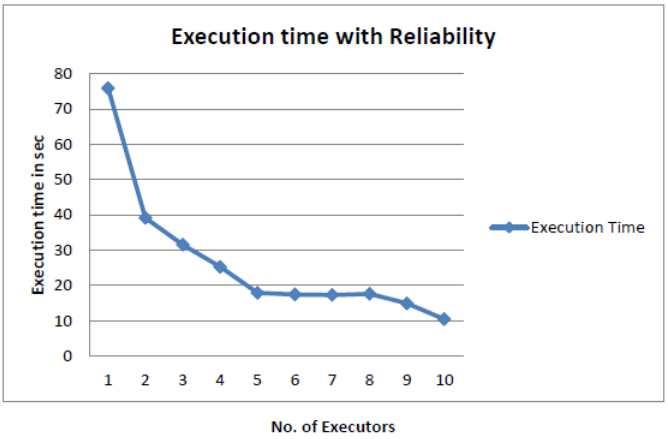

Fig 7: Graph with Reliability Model

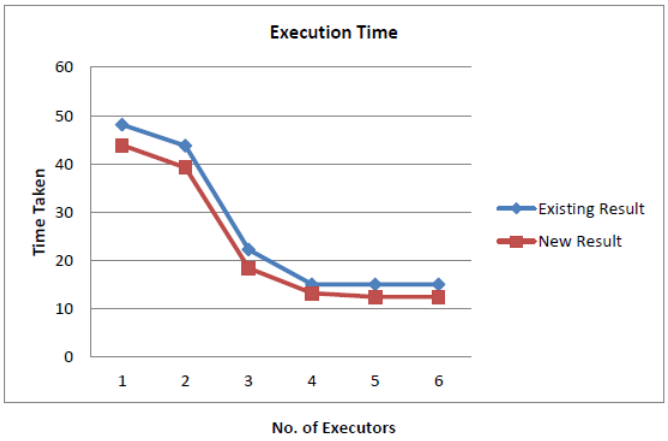

Fig 8: Comparison Graph of Existing and New Result of Application

Analysis of Result

As Shown from the various kind of graphs, the different parameters we are used for measuring the performance of the grid. In first Graph we have taken 10 nodes and take their execution time into consideration which is used without the reliability model. In Second Graph we have used the Reliability model and their algorithm and then again measure the result it shows that the execution time is taken less with reliability model as compared to first graph which is without reliability model.

So, after showing the results of reliability model graph we can say that using reliability model and programming method we can decrease the execution time of job and increase the performance of grid.

Also as shown in third graph we have define the comparison performance of existing and newly proposed algorithm application result. Using this graph we can clearly define that our proposed algorithm graph provide better result than existing algorithm.

So, we can say that using our proposed algorithm and from above measuring graphs that the our throughput is increase and also it take less execution time as compared to existing algorithms.

CONCLUSION AND FUTURE WORK

Job Scheduling Algorithm in Windows Grid helps in improving the parameters like execution time, throughput, reliability, quality of services and maximize the use of available resources.In existing job scheduling algorithm the main factor is execution time and reliability is not provided in such a good manner with better throughput but using our proposed job scheduling algorithm and using reliability model and also with the help of Alchemi framework we can improve this parameters in such a good way. Using our proposed algorithm we can improve the performance of grid upto 20-25% as compared to existing in the terms of throughput, reliability and less execution time. Also, we can say that using reliability model and programming method we can decrease the execution time of job and increase the performance of grid, also making grid more reliable.

In future work we can implement job scheduling algorithm also for wireless grid computing and also our proposed algorithm we can modify according to current need of an industry and we can initiate new kind of Sensor Grid and Mobile Grid technology using this approach.

REFERENCES

[1] Sohil K. Patel1, Sanjay G. Patel, Reliable User Scheduler for Computational Grid, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 3, March 2013

[2] P. Koszek, K. Sandrasegaran, Grid Architecture Storage - Utilising Grid Computing for Dynamic Data Storage, Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)

[3] Hardik.M.Patel, Akash Bhatt, Design and Implementation of Computation Grid, Measure & Improve Performance Parameter:-A Survey, ISSN: 2277-3754 ISO 9001:2008 Certified International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7, January 2013

[4] G. Jaspher W. Kathrine, Mansoor Ilaghi U, Job Scheduling Algorithms in Grid Computing – Survey, International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 7, September – 2012, ISSN: 2278-0181

[5] Thanapal P, Dr. Gunasekaran G, A Distributed Job Scheduling on The Grid Using Particle Swarm Optimization (Pso) Algorithm, Thanapal et al., International Journal of Advanced Research in Computer Science and Software Engineering 3(1), January - 2013, pp. 279-286

[6] Zeljko Stanfel, Goran Martinovic, Zeljko Hocenski, Scheduling Algorithms for Dedicated Nodes in Alchemi Grid,2008 IEEE International Conference on Systems, Man and Cybernetics (SMC 2008)

[7] Sohil K. Patel1, Sanjay G. Patel, Reliable User Scheduler for Computational Grid, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 3, March 2013

[8] T Amudha# 1, T T Dhivyaprabha# 2, QoS Priority Based Scheduling Algorithm andProposed Framework for Task Scheduling in a Grid Environment, IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011 978-1-4577-0590-8/11/$26.00 ©2011 IEEE MIT, Anna University, Chennai. June 3-5, 2011
[9] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal, Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework, November 2007

WEB REFERENCES

[1] http://www.alchemigroup.com/
[2]http://www.cloudbus.org/~alchemi/doc/0_6_1/index.html
[3]
http://sourceforge.net/tracker/?group_id=66729&atid=515536
[4] http://ahmedhosnycs.com/blog/tag/alchemi/