Smart and Dynamic Load Balancing Algorithm in Grid Computing Technology

Dr. Ramesh T. Prajapati[1], Dr. Dushyantsinh Rathod[2]
Department of Computer Engineering
[1]Indrashil Institute of Science and Technology, Rajpur, Kadi, Gujarat
[2]Aditya Silver Oak Institute of Technology, Ahmedabad

Abstract- *Grid Computing has emerged as a new large scale and important field and can be used to improve the performance of grid computing. Grid computing resources has enhanced the performance of computers and reduced their costs Grid computing has large and powerful applications of self-managing virtual computer out of a large collection of heterogeneous systems that sharing various resources which lead to the problem of load balance. The main goal of load balancing is to provide a distributed, low cost, scheme that balances the load across all the processors. We proposed a new algorithm of load balancing for dynamic behavior.*

Keywords- Grid computing, Load Balancing, Job Resources

I. INTRODUCTION

In Grid computing, individual users can access computers and data, transparently, without having to consider location, operating system, account administration, and other details. Grids tend to be more loosely coupled, heterogeneous, and geographically distributed. In Grid computing details are abstracted, and the resources are virtualized [2]. Grid Computing has emerged as a new and important field and can be visualized as an enhanced form of Distributed Computing. Sharing in a Grid is not just a simple sharing of files but of hardware, software, data, and other resources. Thus a complex yet secure sharing is at the heart of the Grid.

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce. Recent research on these topics has led to the emergence of a new paradigm known as Grid computing [2].

Grid Computing defined as applying resources from thousands or many computers in a network to a single problem, usually one that requires to access large amounts of data. Network [14] is sort of parallel and circulated framework that empowers the sharing, determination and collection of geologically conveyed assets progressively at run time contingent upon their accessibility, ability, and execution cost and client nature of administrations. Load adjusting is a system to upgrade assets, using parallelism, abusing throughput ad lib, and to slice reaction time through a proper dissemination of the applications [9].

Work movement is the main proficient approach to ensure that submitted employments are finished dependably and productively if there should be an occurrence of process disappointment, processor disappointment, hub crash, arrange disappointment, framework execution corruption, correspondence delay; expansion of new machines progressively despite the fact that an asset disappointment happens which changes the dispersed environment [11].

These are critical issues in Load Balancing: A surprising pinnacle can be steered to generally sit out of gear machines in the Grid. On the off chance that the Grid is as of now completely used, the least need work being performed on the Grid can be incidentally suspended or even scratched off and performed again later to make space for the higher need work.

There are two types of load balancing policy in grid environment: Static load balancing policy and Dynamic load balancing policy. The static load balancing policy is not well suited to grid environment because the load may vary with respect to time. Based on the load at the time it allocates the job to the nodes. Here the work stations are not constantly monitored. But in dynamic load balancing policy the workstations are constantly monitored. The selection of the policy is done at run time and also uses the current load information for decision making. Dynamic Load balancing policy will not require the priori task information to allocate/reallocate the resource. Dynamic load balancing algorithm will give better performance then static load balancing algorithm.

II. RELATED WORK

Load adjusting is a procedure to upgrade assets, using parallelism, misusing throughput act of spontaneity, and to slice

reaction time through a suitable appropriation of the application [6]. To limit the decision time is one of the goals for stack changing which has yet not been expert.

In cluster based load balancing algorithms [14], the computing nodes are partitioned into clusters on the basis of network transfer delay. In this proposed algorithm, each cluster and all computing nodes of its cluster are defined for making load balancing decisions thereby introducing considerable communication overhead.

In [17], a decentralized dynamic load balancing algorithm is proposed which neglected the overheads involved in collecting state information for load balancing. In this approach, the problem of frequent exchange of information is alleviated by estimating the load, based on system state information received.

In grid the load balancing algorithm will follow three policies there are Information policy, transfer policy, location policy and selection policy. The information policy specifies what workload information to be collected, when it to be collected and from where it to be collected. The resource monitoring system will monitor the status of the resource based on the resource load information the transfer policy will decide whether the resource have the eligibility to act as a sender or receiver. Sender means it will transfer the job to the resource. Receiver means it will receive the job from another resource [2].

Based on the transfer policy the location policy will select a suitable partner for sender or receiver. If the resource is an eligible sender the location policy seeks out an eligible receiver to transfer the job. Suppose the resource is an eligible receiver the location policy seeks out an eligible sender. Once we decide the resource as an eligible sender or receiver the selection policy. The selection policy defines which job should be migrated from heavily loaded node (source) to lightly loaded node (receiver) [2].

Static load balancing [9] on trees, assuming that the total load is fixed. Contrary to the traditional distributed systems for which a plethora of algorithms have been proposed, few of which were focused on grid computing. This is due to the innovation and the specific characteristics of this infrastructure.

Dynamic load balancing algorithms can provide a major improvement in performance over static algorithms. However, this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits [8].

The fundamental goal of load adjusting strategies is to accelerate the execution of applications on assets whose workload changes at run time in unusual way. Subsequently, it is huge to characterize measurements to gauge the asset workload. Each dynamic load adjusting technique must gauge the opportune workload data of each asset. This is entering data in a heap adjusting framework where reactions are given to taking after inquiries:

The nature of resources is dynamic and shared which affects the performance of application. There are two important functions, Management of resource and Workload which provided at the service level of the Grid. The main objective of load balancing is to examine problems due to which load balancing technique is required in grid computing and also to transfer work load from heavy loaded nodes to lightly loaded nodes based on migration policy. This research evaluates the existing Load Balancing algorithm and find out the performance of grid. There are five policies for implementation of Load Balancing algorithms [6].

The usage of Information Policy, in current heap adjust calculation utilizes intermittent approach which is tedious.

The Triggering approach utilizes Queue length which isn't chosen how to stack adjusting process is finished.

For execution of calculation utilizing determination approach in view of Job length which can be utilized to settle on choice about choice of assignment for movement.

III. EXISTING ALGORITHM

In existing algorithm on the basis of load balancing in Grid environment, which dynamically balances the load. This algorithm used for migrating jobs for load balancing. Load balancing take place when some changes occurs in the load state. There are some particular tasks which change the load configuration in Grid environment which can be categorized as following:

- Any new job arrived
- Completion of execution of any job
- Any new resource arrived
- Any existing resource withdrawal
- Machine failure at any node

- Node become overloaded

On the basis of above mentioned activities happened we retrieve the current load information and the initial load information from the database.

Following is the existing algorithm for Adaptive Dynamic Load Balancing:

1. Initialize the status of all nodes.
2. Initial status=.Previous
3. While jobs=N
   N>0 do
4. if Current state is ready to change then
   Current = Get change state ();
   Start:
   If (Standard Deviation of Load of nodes < SD_Threshold)
   If (Load of any node is greater then average Load value of nodes)
   HeavilyLoaded_list= HeavilyLoaded_list + l (new selected node);
   End if
   Else (Load of any node is greater then threshold heavy load value)
   HeavilyLoaded_list= HeavilyLoaded_list + l (new selected node);
   Else if (Load of any node is less then threshold light load value)
   End
   Migrate jobs (Previous,Current); //Partitioning
5. end if
6. end while

Variables Used: These variables are used in the algorithm

- A job defines number of tasks running on grid.
- Current state indicates change in the state of load in grid.
- Current indicates current load status of all the nodes on grid.
- Previous indicates load status of all nodes on grid before the change in the state.

Methods used: These methods are used in the algorithm.

- Get change state () to get the current status of load on each node of grid.
- Migrate jobs (Previous, Current) to migrate jobs as per the current and previous status of nodes.
- Threshold heavy load and threshold light load is defined initially which depends on the traffic of application on the Grid

Load distribution is performed at a centralized controller or manager node. The central controller polls each workstation and collects state information consisting of a node's current load as well as the number of jobs in the node's queue. The polling is done on basis of occurrence of some defined activity. It is not done periodically. Periodic checking approach is used in Condor. In case of periodic approach Load Balancer collects Load sample periodically which is not required and infect creates an overhead also. In the proposed algorithm information is collected only if there is a change in configuration of Grid. This information is used to perform load balancing. Once information has been gathered then it is decided that load balancing is required or not. For this purpose application uses CPU utilization and queue length parameters. With help of these parameters we decide which resource is heavily loaded and which resource is lightly loaded. After selection of resource the application selects job out of n-jobs running on that resource. This selection is based upon on CPU consumption of different jobs. Least CPU consumed job will be selected for migration. When job is selected, application checks for available lightly loaded resource. If lightly loaded resource is available then migrate selected job from heavily loaded resource to lightly loaded resource.

IV. PROPOSED ALGORITHM

As indicated by the name dynamic load adjusting calculations takes choice at run time, and utilize present or late load data when settling on dissemination choices. In matrix environment with element stack adjusting designate/reallocate assets at runtime in light of no from the earlier assignment data, which decide when and which undertaking must be moved.

Segment of code related to Proposed Algorithm:

1. Construct a grid using Alchemi .Net
2. Create prime number application and submit the application to central node.
3. Check runtime load on each resource.
4. Check the node are heavily loaded or lightly loaded or failure node(CPU usage, Free memory and queue Length)
   Start:
   If (CPU Usage is Min and Queue length is Max)
   Heavily Loaded Node
   End if
   If (CPU Usage is Max and Queue length is Min)
   Lightly loaded Node
   Endif
5. If the load of particular node is heavily loaded or failure node then share the load of node to lightly loaded node
6. Analysis of lightly loaded node and heavily loaded node

and check the execution time of application.

In proposed algorithm we have created grid and created one application of prime number allocated to central node. Central node allocated this application to number of machine in which it connected to central node .in existing algorithm used five policies with period approach; queue length, threshold value and proposed algorithm used three policies with activity approach, CPU usage for balancing load. We checked how many nodes have heavy loaded and lightly loaded .if some nodes are lightly loaded then we assign load and balance the load in prime number application.

V. COMPARISON

Input Range to find out prime no	Total No of Executer Running	Total Time to execute application after load balancing	
		Proposed LB Using CPU Utilization and Queue Length	Existing LB Using Threshold and Job length
10000000	6	00:06:0872	01:08:0899
	8	00:07:0967	01:10:0245
	10	00:08:1675	01:11:0891
	20	00:10:1290	02:20:0981
	25	00:09:7845	01:09:0834
	30	00:12:4378	02:89:1943
	35	00:14:4531	03:09:7190
	38	00:15:9081	03:39:8345

Table 1 Existing and Proposed Load balancing algorithm With different parameters

Total Execution Time: Differentiation of dissimilar No of Executer with dissimilar input area with regard to time and parameters.

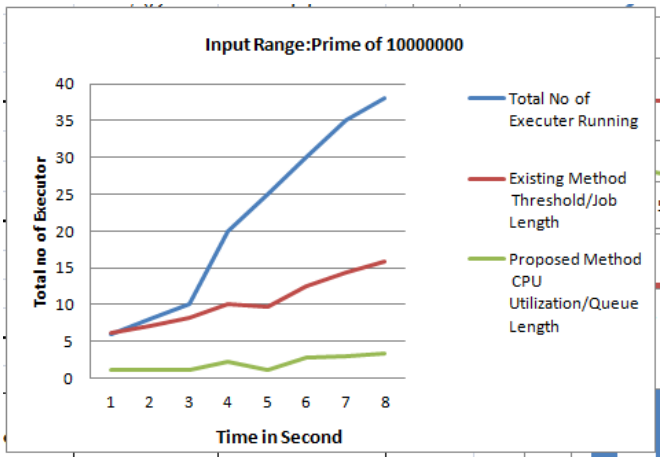

Figure:-1 Differentiation of no of executor running with regard to time.

In the graph fig-1 shows the existing and proposed algorithm result based on execution of application after load balance. In proposed algorithm it takes minimum time to execute application and reduce the communication overhead.

VI. CONCLUSIONS AND FUTURE ENHANCEMENTS

Grid Computing is dynamic structure so balancing load of computer is important. It is heterogeneous computing so ratio of failure is very high. So we proposed new algorithm for load balancing for balancing load of all processor and we are utilized idle processor. In this exploration we broke down existing Load Balancing calculation and proposed an upgraded calculation which all the more effectively actualizes three out of five strategies executed in existing Load Balancing calculation. Proposed Load balancing algorithm used to minimum time to execute job and min response time to execute application Based on above implementation we have executed prime no application with minimum execution time. In future we will implement load balancing in cloud.

REFERENCES

[1] Ramesh Prajapati "Fault Tolerance Mechanism for computational Grid using Checkpoint Algorithm", IJEDR,Aug-2013

[2] Nandagopal Malarvizhi, Rhymend Uthariaraj V, "Decentralized dynamic load balancing for multi cluster Grid environment", Berlin Heidelberg 2011, CCSIT, Springer-Verlag; 2011, Part III, CCIS 133, p. 149–60.

[3] Mr.Ramesh Prajapati,Dr.Samrat Khanna," Grid Computing with different Fault Tolerant Mechanisms", IJSRD,March-2014

[4] Mr.Ramesh Prajapati,Dr.Samrat Khanna," A Review Paper on Grid Computing",IJEDR,Sep-2013

[5] IBM .Redbooks paper

[6] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp and Ion Stoica, Load Balancing in Dynamic Structured P2P Systems.

[7] GRMS,http://www.gridworkflow.org/snips/gridworkflow/space/GRMS

[8] Paul Townend and Jei Xu, "Fault Tolerance within a Grid Environment" Proceedings of AHM2003

[9] Anh Nguyen-Tuong, "Integrating Fault-Tolerance Techniques in Grid Applications" PhD Thesis, University of Virginia, August 2000

[10] Krishna Nadiminti, Akshay Luther, Rajkumar Buyya, "Alchemi: A .NET based Enterprise Grid System and Framework" December 2005

[11] Raissa Medeiros, Walfredo Cirne, Francisco Brasileiro, Jacques Sauvé, "Faults in Grids: Why are they so bad and what can be done about it?" GRID'03

[12] Jonathan M. Smith Computer Science Department Columbia University New York,NY 10027 "A Survey of Process Migration Mechanisms" Technical Report CUCS 324-88.

[13] Inderpreet Chopra, "Fault Tolerance in Computational Grids", GCA'06,2006.

[14] Legrand A, Marchal L, Casanova H, "Scheduling distributed applications: the SimGrid simulation framework", In: Proceedings of the third IEEE/ACM international symposium on cluster computing and the Grid, 2003. p. 138–45.

[15] Mrs.Priya Patel,Mr.Ramesh Prajapati,Dr. Samrat Khanna," To Improve The Performance Of Computational Grid Using Fault Tolerant And Dynamic Load Balancing Algorithm For Grid Environment", IJTRE ,May-2016

[16] D.L. Eager, E.D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. In IEEE Trans. on Soft. Eng., volume 12(5), pages 662–675, 1986.

[17] L. Anand, D. Ghose, V. Mani, "ELISA: An estimated load information scheduling algorithm for distributed computing systems", Computers & Mathematics with Applications, Volume 37, Issue 8, April 1999, Pages 57–85