Load Balancing Approaches in Grid Computing Environment

Neeraj Pandey
Department of Computer Science & Engineering
G. B. Pant Engineering College
Ghurdauri, UK, India

Shashi Kant Verma
Department of Computer Science & Engineering
G. B. Pant Engineering College
Ghurdauri, UK, India

Vivek Kumar Tamta
Department of Computer Science & Engineering
G. B. Pant Engineering College
Ghurdauri, UK, India

ABSTRACT

Grid computing is a kind of distributed computing that involve the integrated and collaborative use of distributed resources. It involves huge amounts of computational task which require reliable resource sharing across computing domains. Load balancing in grid is a technique which distributes the workloads across multiple computing nodes to get optimal resource utilization, minimum time delay, maximize throughput and avoid overload. It is a challenging problem that has been studied extensively is the past several years. This paper attempts to provide a comprehensive overview of load balancing in grid computing environment and also analyses the job distribution and system behavior. Furthermore, this survey various load balancing algorithms for the grid computing environment, identify several comparison metrics for the load balancing algorithms and carry out the comparison based on these identified metrics between them. it also reviews the latest research activities in the area of grid computing, including characteristics, capabilities, architecture, applications, design constraints, scheduling and load balancing and presents a set of challenges and problems.

Keywords

Survey, Grid Computing, Load Balancing, Scheduling, Job distribution, Performance evaluation

1. INTRODUCTION

In past several years the demand for computing power in computational grid environments is continuously increasing. The popularity of the Internet and the availability of powerful and high-speed computing resources and network technologies with optimal and low-cost changes the way of computing. Grid computing is concerned with the exchange of computer power, data storage, and access to large databases, without searching for those shared resources manually by grid user. Like the distributed system a grid system can also be accessed and operated over the internet or other networking technology and provides scalable storage and computational capability without the cost of available resources. Load balancing is one of the primary challenges of existing and future grid based applications and services.

The design of load balancing algorithm requires completed understanding of the grid system and the scheduling strategies used, the heterogeneity between available nodes, the challenges and issues, and their limitations within the grid Architecture. In this article, we analyze and focus on the various load balancing algorithm for grid computing environment and identify challenge and key issues related to them. In grid computing environment the problem of load balancing are closely related to scheduling of jobs to computing nodes because scheduling of jobs get affected by how and which manner a computing node (CN) is utilized. Grid environment generally consist heterogeneous networks and computing nodes. Computing nodes are individual computer or machine consists of different hardware and architectures and various operating systems. This form of grid system can be viewed as aggregation of resources dedicated to a particular task, e.g. virtual organization.

Various services such as resources allocation, job execution, scheduling, and security information etc. are required for a computational grid system. A grid-enable software tool, commonly known as grid middleware or simply middleware provide various amenities and link the resources to support distributed exploration. For various application including collaborative engineering, data exploration, high-throughput computing, and distributed supercomputing the grid infrastructure are great advantageous. [22].

This article review the literature over the period of 2000–2012 for load balancing problem. The survey is organized as follows. Section 2 covers all points related to the grid computing, followed by the load balancing in Section 3. The discussion of the various approaches used in load balancing, parameters etc. is provided in Section 4 and finally Section 5 conclude the paper, followed by the Appendix and References.

2. GRID COMPUTING CONCEPT

A computational grid environment consists of several software and hardware resources such as: computing nodes, storage system, databases, network resources and files system etc. A simple view of grid computing environment in given in figure 1. It consist 4 primary components: Grid User, Grid Resources, Resource Broker, and Grid Information Service (GIS). Initially the grid user interact with the Resource broker and send their task to computation. After then the discovery of the resources, scheduling strategies, and task processing is performed. The Grid information service (GIS) worked as an agent. It collects all the relevant information such as resource availability, node capacity etc. and provide it to the resource breaker to make the scheduling decision.

Resource broker has been implemented in the form of a Web service [45] and provide an abstraction to the complexity of grids by ensuring transparent access to computational resources for executing a job on grid [2]. Regarding system components the GIS provide access to all static and dynamic information. Grid resources are end system or entity that are able to execute multiple jobs on the behalf of the grid user and after execution of job provide computational result to the user.To create a computational grid system we generally require software services (e.g., Globus toolkit) on a set of networked computers. The software services provides facility to share resources, authentication and scheduling of job etc. There are various tools are available for computational grid such as Globus toolkit, and GridSolve etc. that provides solutions to load balancing problem. The main goal of these tools are to provide various functionality and easy access to shared resources. The tentative list of software tools with a small description for computational grid environment is given in table 1.

Table 1. List of tools for grid computing

Tools	Description
Globus [52]	Packaged as a set of components that can be used either independently or together to develop applications.
NetSolve/ GridSolve [53]	A RPC based client/agent/server system that solve computational problem, and bring together disparate computational resources connected by computer networks.
EGI-InSPIRE [54]	Ideally placed to join together the new Distributed Computing Infrastructures (DCIs) such as clouds, supercomputing networks and desktop grids within the European Research Area.
Cactus [55]	An open source problem solving environment, enables parallel computation across different architectures and collaborative code development between different groups.
Legion [56]	An object-based, meta-systems software project at the University of Virginia.
Unicore [57]	(Uniform Interface to Computing Resources) A part of the European Middleware Initiative.
Condor [58]	Support High Throughput Computing (HTC) on large collections of distributively owned computing resources.
GridSim [59]	Java based Grid Simulation Toolkit For Modelling, Simulation, and Application Scheduling for Grid Computing

As given in figure 1 the interaction between various grid components are done using multiple steps[2], which are as follows:

- The grid user run their application and after analysis and specifying their requirement, submit their jobs to grid resource broker (GRB).
- GRB collects all resource information and perform resource discovery.
- After authorizing user and resource(s) GRB schedule the job to appropriate resource(s) or computing nodes.
- Resource(s) execute the job and return computational result to GRB.
- The GRB collects result and provide it to the grid user.

2.1 Grid Characteristics

Scalability, Heterogeneity, adaptability, and multiple administrative domain are the primary aspect to characterized a grid [23]. The primary characteristic of a computational grid system is described as follows:

- Heterogeneity: A grid system includes both software and hardware resources that are heterogeneous in nature.
- Scalability: Ability to handle a huge amount of job in a smooth and controlled manner.
- Transparent access: A grid might be seen as a single virtual computing node.
- Coordination: To provide aggregated computing capability, computing nodes must be coordinated.
- Consistent and Pervasive access: A grid system must be built with standard services, protocols, and interfaces and must grant access to available shared resources by adapting to a dynamic environment.
- Reliability: A grid system must be reliable in terms of node failure etc.

2.2 Grid Middleware

The grid middleware is backbone of the grid computing system which provides a set of core services such as resources authorization, authentication, and mechanisms for job submission, and file transfer etc. on the Grid. There are various core program such as 'e-Science' [9] that move forward the development of robust and generic Grid middleware in collaboration with industry.

A grid is created by installing software services, or middleware e.g. Globus Toolkit [52] on a set of networked computers to provides various facilities such as hardware and software resource location, user authentication, and distributed scheduling of resources and jobs.A view of grid

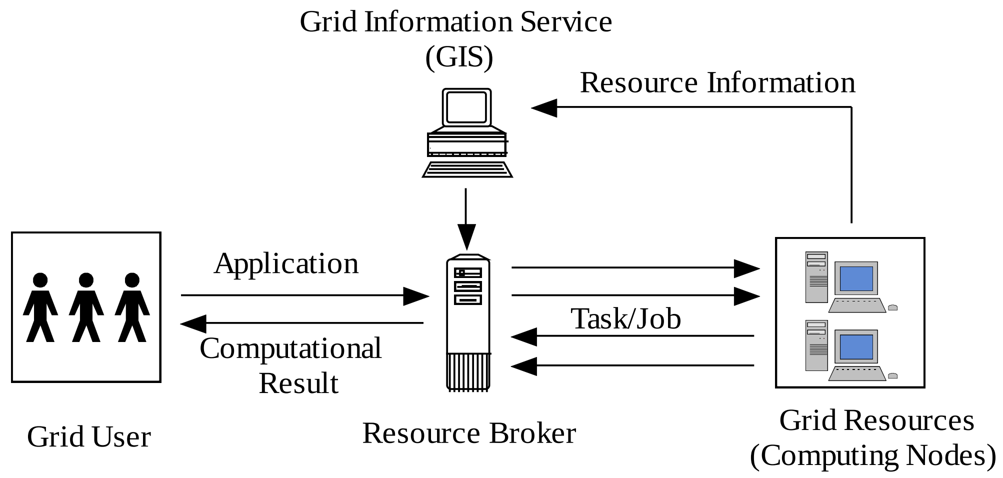

Fig 1:A Grid Computing Environment

middleware is shown in figure 2. It is a software tool that provide GUI to grid user to run their application program and consistent and homogeneous access to the shared resources. Bookkeeping or transection service is a daemon process running with root or administrator privileges on each computational node involve in the Grid. Middleware provides the access and information about the grid resources [50]. The Operations performed by the middleware are as follows:

- Map the resources
- Perform mutual authentication
- Provide secure and transparent access
- Resource allocation
- Job scheduling
- Initiate job process etc.

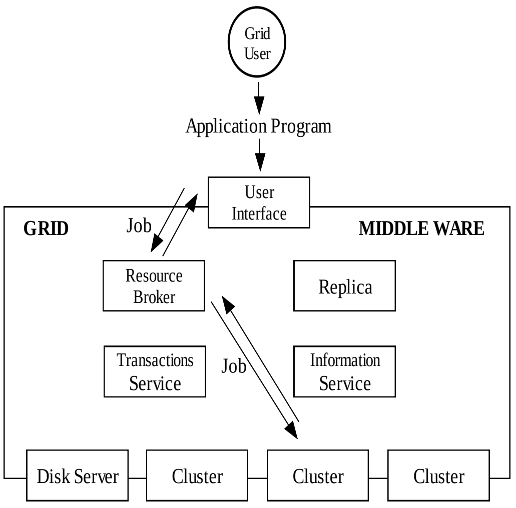

Fig 2: Grid Middleware

2.3 Grid Architecture

A grid architecture follows the "hourglass" model as described by Ian Foster. It identifies basic system components, specifies the purpose and function, and indicates how the interaction between these components are done. The layered architecture of grid system in given in the figure 3.

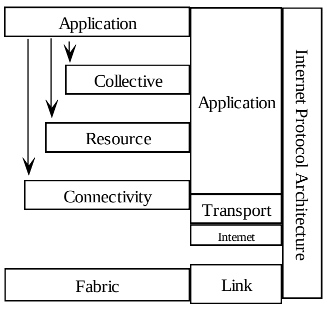

Fig 3: Layer of Grid Architecture [1]

It consists of five layers, which is described as follow:

- Application Layer: It operate in a virtual organization environment.
- Collective Layer: It provide various services such as monitoring, scheduling, broking and directory services etc.
- Resource Layer: Resource layer consist information and management protocol. Information protocol obtain all information related to the grid resources and management protocol used to negotiate access to a shared resources. Generally resource layer provide interactions with a single resource.
- Connectivity Layer: This layer consist several authentication and communication protocol required for communication and exchange of data between computing nodes.
- Fabric Layer: A grid system has several heterogeneous resources and it can be a single machine, cluster or distributed system. The layer provide shared access to the resources required for completion of a task

2.4 Grid Computing Issues and Challenges

All the resource available in the grid environment are basically owned and managed by multiple organization [44]. For the efficient operation of computational grid various factors must be considered such as resources sharing, and load balancing etc. The primary challenges that should be taken into account for grid system are as followed:

- Administration & Security
- Solution Development
- Resource heterogeneity
- Accounting infrastructure
- Programming for application development
- Resource Management

3. LOAD BALANCING

The gaining popularity of high speed and distributed system emerges the problem of load balancing. A load is the number of jobs in the waiting queue and can be light, moderate and heavy according to their work. Load balancing is a process of improving the performance of computational grid system in such a way that all the computing node involve in the grid utilized equally as much as possible. Load balancing is an important function of grid system to distribute the workload among available computing nodes to improve throughput, minimize execution times, maximize node utilization and overall system performance.

According to nature of the work load balancing algorithm is fall into two classes: static and dynamic. In static load balancing the decision information are made in advance. All the information related to scheduling such as node information are known previously before scheduling the job. In case of dynamic load balancing the scheduling decisions are made when there is need to schedule the job for further processing. The static load balancing algorithm are much simpler than dynamic in terms of implementation and load monitoring. A dynamic task scheduling can use either centralized or distributed control. In a centralized approach, all scheduling decision are made at one site and the failure in central site cause entire system down. In a distributed or non-cooperative approach, each site makes its own scheduling decisions and the control is much scalable as well as more reliable [26].There are some issues related to load balancing, some of them are as follows:

- Software solution design
- Partitioning of data or job
- Design of perfect or optimal load balancer
- Job assignment or load distribution
- Load estimation

3.1 Load Balancing Model

This paper present, a quite simple yet sufficiently realistic abstraction of load balancing model is presented. As given in figure 4 computational work or job arrived from grid user are first placed in a job queue. And according to the load balancing scheme the job scheduler schedule the job from job queue to appropriate computing node. The computing node execute the task and send computational result back to the associated grid user. Several parameters are used to measure the performance of the load balancing algorithm such as resource utilization, response time, throughput, waiting time, and reliability etc. Since nodes are heterogeneous in nature so every node has its own capability to process a job.

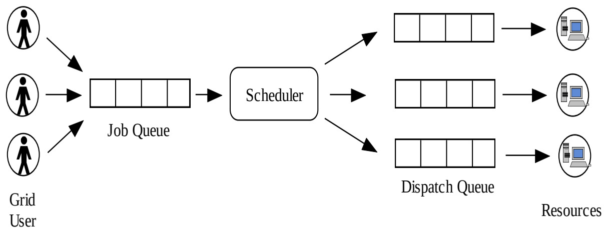

Fig 4: Load Balancing Model

In grid system a computing node can be a single machine, computer or a cluster and focus of a load-balancing strategies is to minimize the variance of the load among all the participating nodes. Various policies such as location policy, selection policy, and information policy has been proposed for load balancing to optimize system performance.

A computational grid system model as a queuing systems. As shown in figure 1, the system consist multiple computing nodes or resources which are connected by a communication network such as Ethernet. A network could be a private network, public network, or internet. Similarly a node could be a single computer/machine or a clusters (group of several computers). When a job is transferred from one node to another, a communication delay may occur. Several assumptions are taken and given as follow:

- Arrival rate (λ): Job arrive and enter into the system, according to a Poisson Process (exponential distribution) with the rate λ.
- Computing nodes are independent of one another.
- A job can be executed on any computing nodes.
- The service time distribution (the average time required to service one job) for completing jobs follow exponential distribution.

A job possible could be either a dependent job or mutually independent. In a mutually independent job, there is no precedence constraint exists between jobs while in dependent jobs their exist a precedence.If jobs are arriving at the exponentially distributed rate λ, then the probability that there will be n jobs after time t is given as:

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \quad (1)$$

The inter-arrival time, $\mathcal{T}$, is the average time between job arrivals, measured in time per job and given as:

$$\mathcal{T} = 1/\lambda \quad (2)$$

3.2 Load Sharing

A workload or load simply defined as number of job in waiting queue and could be light, moderate, or heavy according to node status. The load index or capacity for any computing node belongs to capability of node and describe as a 3 tuple element $\{I_1, I_2, I_3\}$, where I_i is one of the following load matrices:

{Node utilization, Memory utilization, I/O utilization}

The definition of a load is depicted in table 2. The load sharing involve transferring the workload form heavily loaded computing node to lightly loaded node to improve the performance of grid system.

Table 2. Load Definition

	Load (L)	**Node Utilization**
Light Load	$L < L_{low}$	Low
Medium Load	$L_{low} < L < L_{max}$	Moderate
Heavy Load	$L > L_{max}$	High

Load sharing simply attempt to avoid ideal nodes and sometime known as process migration. The load sharing approach is quite simpler than load balancing as it attempts to migrate the heavily loaded node to lightly loaded nodes and ensure that no node is idle when heavily node exists in the system. Before scheduling the job it determine available nodes and then monitor it. At a given time stamp t the load Li on a computing node N is define as the sum of the loads at that time t. The processing speed of node generally define the capacity C_i of the node and given as:

$$\{ C_i \geq 1 | 1 \leq i \leq \text{number of nodes} \} \quad (3)$$

The node utilization U for a given node is generally defined as the ratio of busy time (T_{busy}) and sum of ideal time (T_{ideal}) and T_{busy} of that node and given as:

$$U = \frac{T_{busy}}{T_{ideal} + T_{busy}} \quad (4)$$

The disadvantage of load sharing approach is that more than one processor looks for job at the same time therefore a bottleneck exist there.

3.3 Job Characteristics

There are two types of jobs- sequential and parallel. A sequential job require a single node to execute whether a parallel job require more than one node to complete its operation and known as job parallelism. The degree of parallelism for a job J is given as number of tasks consists by J. The primary characteristic of a job is described as follows:

- Memory, I/O, and Network requirements.
- Job dependency (modeled as dependency graph).
- Runtime or execution time: a required time period. to process or execute the job.
- Arrival time: time at which a job arrives.
- Finish time: time at which a job leaves/complete its execution.
- Waiting Times: time spend at job queue.

3.4 Grid Agent

An agent collects the real time information used for processing of job and also monitors the waiting tasks [3]. The real time information includes the status of current task execution status, computing capabilities and node behavior etc. Each agent provides a high-level representation of a grid resource. It consist a layered architecture [10] with the following layer:

- Communication layer: provide communication using common data models and communication protocols and interface to heterogeneous networks and operating systems.
- Coordination layer: Accept request form above layer (e.g. service discovery) and act according to request.

- Local management layer: performs agent functions such as scheduling for local grid load balancing and provide information to above layer for decision making.

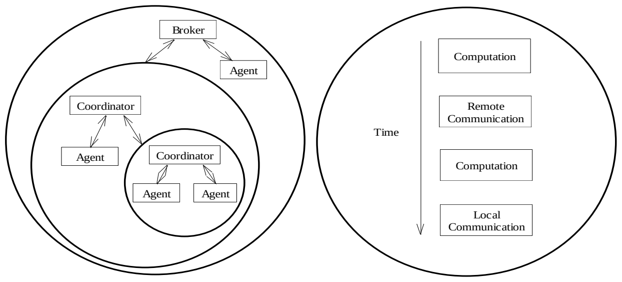

Fig 5: Hierarchy (a) and Structure (b) of Grid Agent

Since there may be multiple agent for a grid computing system, therefore all agents are organized in a hierarchical manner. A coordinator is an agent that heads a sub-hierarchy. The structure of a grid agent [28] and grid hierarchy [10] are given in figure 5.A agent process is distinct form a traditional process in various properties. Ka-Po Chow et al. in [28] define the following properties for a grid agent:

Table 3: Properties of grid agent

Properties	Meaning
Reactive	responds in timely fashion to change in the environment
Autonomous	exercises control over its own action
Temporally continuous	continually running process
Communicative	communicate with other agents
Adaptive	change its behaviour based on previous experience
Proactive	does not simply act in response to the environment
Cloning	duplicate itself to achieve better performance
Mobile	Transport itself from one machine to another

4. DISSCUSSION

In this section, we provide the discussion and comparative analysis of the load balancing algorithms. For comparison we take six algorithm and provide some small description and then compare these algorithm. The comparison chart for these algorithms is provided in table 6. The performance of load balancing algorithm is measured by various parameters, some of these parameters are given in table 4. The main purpose of all load balancing policies is distribute equal workload in each and every node as much as possible therefore it require exact state information of nodes.

Table 4: Algorithm parameter

Parameters	Description
Nature	Static vs. dynamic
CPU overhead	Less to moderate or moderate to high
Node utilization	Low to high
Implementation	Easy Vs. complex
Cost	Less to high
Response time	Less to moderate
Algorithm reliability	Less to moderate or high
Decision information	After/before scheduling
Policy	Centralized Vs. de-centralized
Complexity	Easy Vs. Complex
Job behaviour	Pre-emptive Vs. non pre-emptive
Load monitoring	Continuous Vs. fitful
Fault tolerance	Yes or no
Cost	Low to medium or Medium to high
Clustering	Homogeneous Vs. heterogeneous

Since behavior of node in the system is not static due to node failure or node removal etc., therefore a temporal unbalance among the nodes exists. Due to dynamic nature of grid system a job replication approach [5] are suitable to deal with job processing. A job replication scheme is same as job distribution but send a copy of job to each processor for processing instead of different types of jobs. Tables 5 gives a comparative analysis of replication and distribution of job in compute grid system.

Table 5: Job replication Vs. Job distribution

Parameters	Job replication	Job distribution
Description	Same types of job are processed by each node.	Each node process different types of job.
Job type	Similar types of job (job replica)	Dissimilar
No. of copies of job	No. of node-1	1
Dataset appli-ed to each job	Same types	Different Types
Processing overhead	More	Less than replica-tion
Implementa-tion detail	Easy to implement	Between Easy (static) & complex (dynamic)
Speedup	Moderate	Moderate to high
Reliability	Moderate	Moderate to high

4.1 Fuzzy Based Approach

Fuzzy logic is a multivalued logic control and the rules are in the form of IF-THEN (fuzzy conditional statements). For example: **IF the node1 IS lightlyLoaded AND the node2 IS heavilyLoaded THEN the UnbalancedLoad**. The mapping of input to output is provided by fuzzy inference system (FIS). The advantage using fuzzy based approach is that, first it detect the imbalance between nodes and second avoid the unnecessary load. Fuzzy based load balancing firstly analyse information passed from the load monitor, and then make a decision. It use a domain expert's knowledge for creation of rule base. A fuzzy load balancer provide faster response time than average or periodic load balancer.

4.2 Genetic Algorithm Based Approach

For a large scale optimization problem the GA is a well-known and robust search technique. The primary goal of genetic based scheduling algorithm is to minimize the completion time of the job as well as the maximize the node utilization. The GA starts with randomly generated initial population called chromosome. Solutions from one population are taken and used to form a new population. After several generation final solution or optimal solution is generated. Three basic operation used in GA are: selection, crossover, and mutation.

4.3 Agent Based Approach

In agent based approach an agent hierarchy is exists there for agent based scheduling. A centralized control mechanism is used by agent. An agent search suitable nodes for execution of job. A system with multiple agent dispatched the agents to multiple nodes to obtain service. All agent have pre or earlier knowledge of all other agent and whenever an agent receive a job, it connect all the agent to determine the job execution time. Q. Long et al. [21] describe the 3 most important policy for agent based policy as follow:

- Migrated agent sets selection,
- Target containers selection, and
- Agent migration

4.4 Hybrid Approach

Sometimes a node may be in ideal state and sometime in busy state. To effectively utilize the participant node and overall system a hybrid approach is beneficial. In static approach there is no need to continuously collect system information . In other hand in dynamic approach to assign a task to appropriate node a continues monitoring of system information needed. The effectiveness of nodes may vary with time therefore a dynamic job assignment must be done there.

4.5 Policy Based Approach

The scheduling problem require best resource allocation to a set of job in an efficient way. A policy based approach handle different computation time of a job on various nodes. The initial execution time of a job set with the mean value. Mean value is taken by using the different time values on a set of available nodes. When the algorithm change its scheduling decision this time is updated by using iterative scheduling approach.

4.6 History Based Approach

In this approach the scheduler first estimate the start time for job and then allocates it to appropriated computing node. Estimation of start time is done using execution history. The scheduler contain various module such as resource select, reservation map, and information service. For a queuing based system the start time estimation following steps are taken:

- When a job complete its execution, the corresponding node send job execution status to grid agent.
- Agent stores the information, and provide it to scheduler to make decision

Execution status consists of all information related to that job including name, id, type, execution time taken and account information

5. CONCLUSION

In current scenario, compute grid involves sharing of variety of heterogeneous resources. To distribute the workload using any application effectively on grid computing systems, load balancing algorithm must be selected and design carefully. In this study, a survey of load balancing in the computational grid environment is presented. In addition, a qualitative comparison of load balancing algorithms is provided. The focus in this paper is to provide all relevant information related to load balancing in computational grid environment.

6. APPENDIX

This section gives some definition related to load balancing in computational grid environment used in this paper.

Definition 1: Since each task T has a start time T_s, end time T_e, arrival time T_a, and a computational duration T_d, therefor a task can be expressed as a four tuple element as:

$T = \{ T_s, T_e, T_a, T_d \}$(5)

Definition 2: The resource or computing node involve in grid system are heterogeneous in nature therefore each node has a its own capacity to execute a job. Since each job require some memory M, processing power Pw, and Input/output IO to complete its processing therefore the capacity of a node can be express as a three tuple element as:

$C = \{ M, Pw, IO \}$(6)

Definition 3: If the capacity of two nodes N_1 and N_2 are C_1 and C_2 respectively. Then a task which takes m units of time on the node N_1 to be processed would take m* (C_1/C_2) unit of time on the N_2.

Definition 4: The latest completion time (sum of execution time of all job processed) among all the node N involved in a grid system is known as makespan M.

$$M = \{ \text{Max} (\text{Load} (Ni), \forall\ 1 \le I \le \text{no of node}) \} \quad (7)$$

Definition 5: The node utilization (individual) Nu for a node is achieved by dividing the sum of completion time T_c by the makespan value.

$$N_u = \frac{\sum T_c}{makespan} \quad (8)$$

Definition 6: The node utilization (average) N_{avg} is achieved by sum of node utilization (individual) divided by the total number of node (m) involve in grid system.

$$N_{avg} = \frac{\sum_1^m N_u}{m}$$

(9)

Table 6: Load Balancing Approach : Comparative Analysis

Approach	**Description**	**Primary Consideration**	**Advantage**
Fuzzy Based [6,7,8,46]	Rule based methods use a domain expert's knowledge for the creation of rule base.	Membership function, Fuzzy rule, Fuzzy sets.	Easy to implement, Fastest response time, Better load balance.
Genetic Algorithm Based [3,4,25,56]	Search methods based on the principles of evolution and natural genetics.	Encoding scheme, Selection & fitness evaluation, Crossover & mutation	Work better when number of job increases, Better performance, Minimize job completion time.
Agent Based [10,21]	Agent hierarchy consists of a number of agents that work together to find load balancing solution.	Agent implementation Task Allocation, Scheduling Scheme, Partial information	Agent hierarchy consists of a number of agents that work together to find load balancing solution.
Hybrid Approach [11,13]	Combination of static and dynamic load balancing algorithm.	Job scheduling scheme, Re-distribution of job	More effective, Reduce job completion time Improved job execution performance
Policy Based [45]	Allocates grid resources to an application under the constraints with resource usage policies.	Resource allocation, Scheduling algorithm	Improves the completion time of Job, Optimized scheduling.
History Based [39]	A history of job-execution is used to estimate the start time of job.	Time estimation, Job allocation mechanism	Improve the utilization efficiency of computing nodes.

7. REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, The International Journal of High Performance Computing Applications, 15 (3) (2001) 200-222.

[2] Rajkumar Buyya, and Srikumar Venugopal, A Gentle Introduction to Grid Computing and Technologies, Computer Socity of India, CSI Communication, July (2005).

[3] Yajun Li, Yuhang Yang, Maode Ma, and Liang Zhoy, A hybrid load balancing strategy of sequential tasks for grid computing environments, Future Generation Computer Systems, 25 (2009) 819-828

[4] Albert Y. Zomaya, Yee-Hwei Teh, Observations on Using Genetic Algorithms for Dynamic Load-Balancing, IEEE Transections on Parallel and Distributed System, Vol. 12, No. 9, Sept. (2001) 899-911.

[5] Menno Dobber, Rob van der Mei, Ger Koole, Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison, IEEE Transections on Parallel and Distributed System, Vol. 20, No. 2, Feb. (2009) 207-218.

[6] Yu-Kwong Kwok, Lap-Sun Cheung, A new fuzzy-decision based load balancing system for distributed object computing, Journal of Parallel and Distributed Computing, 64 (2004) 238–253

[7] Mika Rantonen, Tapio Frantti, Kauko Leiviska, Fuzzy expert system for load balancing in symmetric multiprocessor systems, Expert Systems with Applications 37 (2010) 8711–8720.

[8] R.P. Prado, S. Garcia-Galan, A.J. Yuste, and J.E. Munoz Exposito, A fuzzy rule-based meta-schedular with evolutionary learning for grid computing, Engineering Applications of Artificial Intelligence 23 (2010) 1072–1082.

[9] Tony Hey, Anne E. Trefethen, The UK e-Science Core Programme and the Grid, Future Generation Computer Systems 18 (2002) 1017–1031.

[10] Junwei Caoa, Daniel P. Spooner, Stephen A. Jarvis, Graham R. Nudd, Grid load balancing using intelligent agents, Future Generation Computer Systems 21 (2005) 135–149.

[11] K.Q. Yan, S.C. Wang, C.P. Chang, J.S. Lin, A hybrid load balancing policy underlying grid computing environment, Computer Standards & Interfaces 29 (2007) 161–173.

[12] Jun Wang, Jian-Wen Chen, Yong-Liang Wang, Di Zheng, Intelligent Load Balancing Strategies for Complex Distributed Simulation Applications, 2009 International Conference on Computational Intelligence and Security, 2009 (182-186).

[13] Kuo-Qin Yan, Shun-Sheng Wang, Shu-Ching Wang, Chiu-Ping Chang, Towards a hybrid load balancing policy in grid computing system, Expert Systems with Applications 36 (2009) 12054–12064.

[14] Sonesh Surana, Brighten Godfrey, Karthik Lakshminarayanan, Richard Karp, Ion Stoica, Load balancing in dynamic structured peer-to-peer systems, Journal of Performance Evaluation 63 (2006) 217–240.

[15] Luis Miguel Campos, Isaac D. Scherson, Rate of change load balancing in distributed and parallel systems, Parallel Computing 26 (2000) 1213-1230.

[16] Karen D. Devine, Erik G.Boman, Robert T. Heaphy, Bruce A. Hendrickson, James D. Teresco, Jamal Faik, Joseph E. Flaherty, Luis G. Gervasio, New challenges in dynamic load balancing, Applied Numerical Mathematics 52 (2005) 133–152.

[17] Arjen Schoneveld, Peter M.A. Sloot, Martin Lees, Erwan Karyadi, A framework for dynamic load balancing: A case study on explosive containment simulation, Parallel Computing 26 (2000) 737-751.

[18] Xiao Qin, Performance comparisons of load balancing algorithms for I/O-intensive workloads on clusters, Journal of Network and Computer Applications 31 (2008) 32–46.

[19] Daniel Grosu, Anthony T. Chronopoulos, Noncooperative load balancing in distributed systems, Journal of Parallel and Distributed Computing 65 (2005) 1022 – 1034.

[20] Yin-Fu Huang, Chih-Chiang Fang, Load balancing for clusters of VOD servers, Information Sciences 164 (2004) 113–138.

[21] Qingqi Long, Jie Lin, Zhixun Sun, Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations, Simulation Modelling Practice and Theory 19 (2011) 1021–1034.

[22] Bruce Hendrickson, Karen Devine, Dynamic load balancing in computational mechanics, Computer methods in applied mechanics and engineering 184 (2000) 485-500.

[23] Mark Baker, Rajkumar Buyya, and Domenico Laforenza, Grids and Grid technologies for wide-area distributed computing, Software practice and experience 2002; (DOI: 10.1002/spe.488)

[24] Sara Kardani-Moghaddam, Farzad Khodadadi, Reza Entezari-Maleki, Ali Movaghar, A Hybrid Genetic Algorithm and Variable Neighborhood Search for Task Scheduling Problem in Grid Environment, Engineering 00 (2011) 3808-3814.

[25] Zhongju Zhang, Weiguo Fan, Web server load balancing: A queueing analysis, European Journal of Operational Research 186 (2008) 681–693.

[26] Jiannong Cao, Graeme Bennett, Kang Zhang, Direct execution simulation of load balancing algorithms with real workload distribution, The Journal of Systems and Software 54 (2000) 227-237.

[27] Vladimir V. Korkhov, Jakub T. Moscicki, Valeria V. Krzhizhanovskay, Dynamic workload balancing of parallel applications with user-level scheduling on the Grid, Future Generation Computer Systems 25 (2009) 28–34.

[28] Ka-Po Chow and Yu-Kwong Kwok, On Load Balancing for Distributed Multiagent Computing, IEEE Transactions on parallel and distributed systems, Vol. 13, No. 8, Aug. (2002) 787-801.

[29] Wang Lei, Chen Qing, Gao Zhanjun, Power Systems Fault Diagnosis Based On Grid Computing, The

International Conference on Advanced Power System Automation and Protection, IEEE (2011) 1557-1561.

[30] Ruchir Shah, Bhardwaj Veeravalli, and Manoj Misra, On the Design of Adaptive and Decentralized Load-Balancing Algorithms with Load Estimation for Computational Grid Environments, IEEE Transactions on parallel and distributed systems, Vol. 18, No. 12, Dec. (2007) 1675-1686.

[31] Alpana Rajan, Anil Rawat, Rajesh Kumar Verma, Virtual Computing Grid using Resource Pooling, International Conference on Information Technology, IEEE (2008) 59-64.

[32] Nirmalya Roy, and Sajal K. Das, Enhancing Availability of Grid Computational Services to Ubiquitous Computing Applications, IEEE Transactions on parallel and distributed systems, Vol. 20, No. 7, July (2009) 953-967.

[33] S. Luo, X. Peng, S. Fan, P. Zhang, Study on Computing Grid Distributed Middleware and Its Application, International Forum on Information Technology and Applications, IEEE (2009) 441-445.

[34] M. Ali, Z.Y. Dong, and P. Zhang, Adoptability of grid computing technology in power systems analysis, operations and control, IET Generation, Transmission & Distribution, Vol. 3, Iss. 10 (2009) 949-959.

[35] Kuo-Chan Huang, On Effects of Resource Fragmentation on Job Scheduling Performance in Computing Grids, 10th International Symposium on Pervasive Systems, Algorithms, and Networks, IEEE (2009) 701-705.

[36] Lizhe Wang, G. V. Laszewski, Dan Chen, Jie Tao, and M. Kunze, Provide Virtual Machine Information for Grid Computing, IEEE Transactions on systems, man, and cybernetics-part a: Systemand Humans, Vol. 40, No. 6, Nov. (2010) 1362-1374.

[37] P. G. S. Tiburcio, M. A. Spohn, ad hoc Grid: An Adaptive and Self-Organizing Peer-to-Peer Computing Grid, 10th International Conference on Computer and Information Technology (CIT) IEEE (2010) 225-232.

[38] Liang Bai, Yan-Li Hu, Song-Yang Lao, Wei-Ming Zhang, Task Scheduling with Load Balancing using Multiple Ant Colonies Optimization in Grid Computing, Sixth International Conference on Natural Computation (ICNC), IEEE (2010) 2715-2719.

[39] Y. Murata, R. Egawa, M. Higashida, H. Kobayashi, A History-Based Job Scheduling Mechanism for the Vector Computing Cloud, 10th Annual International Symposium on Applications and the Internet, IEEE (2010) 125-128.

[40] Alexandru Iosup and Dick Epema, Grid Computing Workloads, IEEE Internet Workloads March/April (2011) 19-26.

[41] Shuai Zhang, Shufen Zhang, The Comparison Between Cloud Computing and Grid Computing, International Conference on Computer Application and System Modeling (ICCASM), IEEE (2010) 72-75.

[42] Rajkumar Rajavel, De-Centralized Load Balancing for the Computational Grid Environment, Proceedings of the International Conference on Communication and Computational Intelligence, India (2010) 419-424.

[43] K. Hasham, A. D. Peris, A. Anjum, D. Evans, S. Gowdy, J. M. Hernandez, E. Huedo, D. Hufnagel, F. van Lingen, R. McClatchey, and S. Metson, CMS Workflow Execution Using Intelligent Job Scheduling and Data Access Strategies, IEEE Tranasaction on nuclear science, Vol. 58, No. 3, June (2011) 1221-1232.

[44] Naidila Sadashiv, S. M Dilip Kumar, Cluster, Grid and Cloud Computing: A Detailed Comparison, The 6th International Conference on Computer Science & Education (ICCSE), IEEE Aug. (2011) 477-482.

[45] Jang Uk In, Soocheol Lee, Seungmin Rho, and Jong Hyuk Park, Policy-Based Scheduling and Resource Allocation for Multimedia Communication on Grid Computing Environment, IEEE Systems journal, Vol. 5, No. 4, Dec. (2011) 451-459.

[46] W. Cheng, J. Congfeng, Liu Xiaohu, Fuzzy Logic-Based Secure and Fault Tolerant Job Scheduling in Grid, Tsinghua Science and Technology, Vol. 12 N0. S1 (2007) 45-50.

[47] Malcolm Irving, Gareth Taylor, and Peter Hobson, Plug in to Grid Computing Moving Beyond the Web, IEEE power & energy magazine, March/April (2004) 40-44.

[48] G.Manimaran, M. Shashidhar, Anand Manikutty, C.Siva Ram Murthy, Integrated Scheduling of Tasks and Messages in Distributed Real-time Systems, IEEE (1997) 64-71.

[49] Angelo Boccia, Gianluca Busiello, Luciano Milanesi, and Giovanni Paolella, A Fast Job Scheduling System for a Wide Range of Bioinformatic Applications, IEEE Transactions on Nanobioscience, Vol. 6, No. 2, June (2007) 149-154.

[50] M. Ali, Z. Y. Dong, X. Li, and P. Zhang, RSA-Grid: A Grid Computing based Framework for Power System Reliability and Security Analysis, IEEE (2006) 1-7.

[51] Zeng Zeng,and Bharadwaj Veeravalli, Design and Performance Evaluation of Queue-and-Rate-Adjustment Dynamic Load Balancing Policies for Distributed Networks, IEEE Transactions on Computers, Vol. 55, No. 11, Nov. 2006) 1410-1422.

[52] Globus, http://www.globus.org, accessed March 2013.

[53] NetSolve/GridSolve, http://icl.cs.utk.edu/netsolve, 2013.

[54] EGI - European Grid Infrastructure, http://www.egi.eu/about/egi-inspire//, 2013

[55] Cactus, http://cactuscode.org, 2013.

[56] Legion: A Worldwide Virtual Computer, http://legion.virginia.edu, accessed Jan. 2013.

[57] UNICORE-Distributed computing and data resources, http://www.unicore.eu, accessed Jan. 2013.

[58] Condor-High Throughput Computing, http://research.cs.wisc.edu/htcondor/, 2013.

[59] Grid Simulation Toolkit , http://www.cloudbus.org/gridsim/, 2013.