# Resolving Load Balancing Issue of Grid Computing through Dynamic Approach

Er. Roma Soni
M-Tech Student

Dr. Kamal Sharma
Prof. & Director of E.C.E. Deptt.
EMGOI , Badhauli.

Er. Sharad Chauhan
Asst. Prof. in C.S.E. Deptt.
EMGOI , Badhauli

***Abstract***--Load balancing has been a key concern for traditional multiprocessor systems. The emergence of computational grids extends this challenge to deal with more serious problems, such as scalability, heterogeneity of computing resources and considerable transfer delay. Due to the dynamic property of grid environment, fixed-parameter prediction model cannot exert its forecast capability completely. To improve the global throughput of computational grid, effective and efficient load balancing algorithms are fundamentally important. A computational grid differs from traditional high-performance computing system in the heterogeneity of computing nodes, as well as the communication links that connect the different nodes together.

A dynamic and decentralized load balancing algorithm for computationally intensive jobs on a heterogeneous distributed computing platform is required. The time spent by a job in the system is considered as the main issue that needs to be minimized.
A resource queue length based solution to the grid load balancing problem has been proposed in this dissertation. An algorithm that is dynamic, decentralized and distributed for load balancing among the aggregated processing elements in the grid has proposed .The proposed algorithm balances the load in the grid based on the queue length of each processing element and transfer the task to the processing element having minimum queue length. The proposed Algorithm is implemented with the help of GridSim toolkit. The Simulations are performed for number of users and processing elements.

*****

## 1 .INTRODUCTION

This is an introduction to grid computing in the terms of its characteristics, problem areas, Grid architecture, load balancing, advantages, disadvantages, objective of dissertation and organization of dissertation.

### 1.1 GRID COMPUTING

The term grid is increasingly appearing in computer literature, generally referring to some form of system framework into which hardware or software components can be plugged, and which permits easy configuration and creation of new functionality from existing components.

Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering, and commerce. The term grid is chosen as an analogy to the electric power grid that provides consistent, pervasive, dependable, transparent access to electricity, irrespective of its source. Such an approach to network computing is known by several names: meta computing, scalable computing, global computing, Internet computing, and more recently Peer-to-Peer computing.

The concept of grid computing [1] started as a project to link geographically dispersed supercomputers, but now it has grown far beyond its original intent. The grid infrastructure [2] can benefit many applications, including collaborative engineering, data exploration, high throughput computing, distributed supercomputing, and service-oriented computing.

### 1.2 GRID ARCHITECTURE

Grids are usually heterogeneous networks.Grid nodes, generally individual computers that consist of different hardware and use a variety of operating systems and networking to connecting them vary in bandwidth. It is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across multiple domains based on their resources availability, capability, performance, cost and users requirement.

The components that are necessary to form a Grid (shown in Figure 1.3) are as follows:-

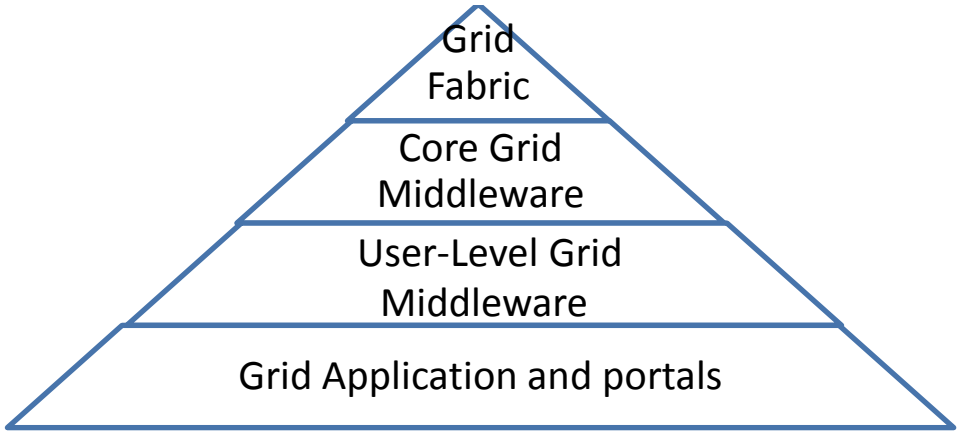


*Figure 1.2. A layered Grid architecture and components*

It is a system formed to share the resources among the various local domains and which owns the resources. These resources are used among the various projects. This forms the system as the aggregation of resources for a particular task i.e. virtual organization.

## 2. LOAD BALANCING PHENOMENA

This Chapter gives an overview of Load balancing, motivation for load balancing, its schemes, policies and challenges of Grid Computing.

### 2.1 INTRODUCTION

Distributed network computing environments have become a cost effective and popular choice to achieve high performance and to solve large scale computation problems. Unlike past supercomputers a cluster or grid or peer-to-peer system can be used as multipurpose computing platform to run diverse high performance parallel applications. Cluster computing [17] environment consist of Personal Computers that are interconnected using high speed networks and are located at same location where as grid computing involves coupled and coordinated use of geographically distributed resources for purposes such as large scale computation and distributed data analysis [18] [19].

A peer-to-peer system [20] is composed of participants that make a portion of their resources (such as processing power, disk storage, and network bandwidth) available directly to their peers without intermediary network hosts or servers. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model where only servers supply, and clients consume.

A collection of autonomous intelligent devices connected through a communication link is called Distributed System [21]. A major advantage of using distributed system is resource sharing. A major shareable resource is Central Processing Unit cycles. A distributed scheduler is a resource management component of a distributed operating system that focuses on redistributing the load of the system among the individual devices, such that the overall performance of the system is optimized.

Load Balancing in a distributed system is a process of sharing computational resources by transparently distributing system workload. A distributed computing system consists of software programs and data resources dispersed across independent devices. A workstation user may not use the machine all the time, but may require more than it can provide while actively working.Some devices may be heavily loaded, while other remains idle. Performance enhancement is one of the most important issues in distributed system. The performance of the system can often be improved to an acceptable level simply by.

## 3. GRIDSIM: GRID MODELING AND SIMULATION TOOLKIT

The GridSim toolkit provides a comprehensive facility for simulation different classes of heterogeneous resources, users, applications, and resource brokers. In Grid -like environment, resource brokers perform resource selection and aggregation depending on users requirements and hence they are user-centric in nature. Whereas, in single administrative domain PDC systems such as clusters, resource schedulers manage whole resource and hence they are generally system –centric in nature as they aim to optimize system throughput for enhancing utilization of the entire system[38].The GridSim toolkit can be used for simulating application schedulers for different parallel and distributed computing systems such as clusters and grids.

### 3.1 FEATURES OF GRIDSIM TOOLKIT

Salient features of the GridSim toolkit include the following:

- It allows modeling of heterogeneous types of resources.

 ISSN: 2321-8169
2572 – 2579

- Resources can be modeled operating under space-or time -shared mode.
- Resource capability can be defined (in the form of MIPS as per SPEC benchmark).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on resource's local time to model non- Grid (local) workload.
- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- Application tasks can be heterogeneous and they can be CPU and/or I/O intensive.
- It does not limit number of application tasks that can be submitted to a resource.
- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time -shared or space-shared. This feature helps in building schedulers that can use different market-driven economic models for selecting services competitively.
- Network speed between resources can be specified.
- It supports simulation of both static and dynamic schedulers.
- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim statistics analysis methods.

## 4. PROPOSED METHODOLGY

This chapter gives an introduction to generic model for Computational grid, proposed algorithm and working of proposed algorithm.

Generic Model for Computational Grid

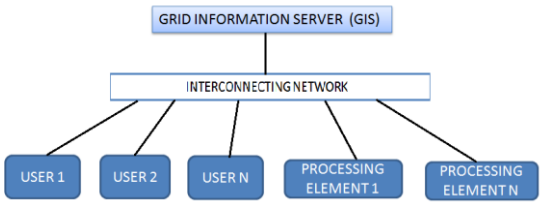


*Figure 4.1: Generic Model for Computational grid*

The Generic model for computational grid consists of Grid Information Server (GIS), users and processing elements.

### 4.1.1User

Each instance of the User entity represents a Grid user. Each user may differ from the rest of users with respect to the following characteristics:

- types of job created, e.g. job execution time, number of parametric replications, etc.;
- scheduling optimization strategy, e.g. minimization of cost, time, or both;
- activity rate, e.g. how often it creates new job;
- time zone; and
- absolute deadline and budget

### 4.1.2 Processing Element

Each instance of the Processing element entity represents a Grid processing element. Each processing element may differ from the rest of the processing element with respect to the following characteristics:

- number of processors;
- cost of processing;
- speed of processing;
- internal process scheduling policy, e.g. time-shared or space-shared;
- local load factor; and
- time zone.

The processing element speed and the task execution time can be defined in terms of the ratings of standard benchmarks such as MIPS and SPEC. They can also be defined with respect to the standard machine. Upon obtaining the processing element contact details from the Grid information service, balancers can query processing elements directly for their static and dynamic properties.

### 4.1.3 Grid Information Server

Providing processing element registration services and keeping track of a list of processing elements available in the Grid. The brokers can query this for processing element contact, configuration, and status information.

### 4.2 PROPOSED ALGORITHM

A Dynamic load balancing algorithm has been proposed that handles heterogeneous grid sites. Here, the heterogeneity only refers to the processing power of sites. The proposed algorithm balances the load in the grid based on the queue length of each processing element and transfer the job to the processing element having minimum queue length. The proposed algorithm is implemented using GridSim Toolkit.

The proposed algorithm for Dynamic load balancing in computational grid is as follows:

1. Begin the GridSim Toolkit Package.
2. Initialize the number of users and processing elements.
3. Generate users from 1 to 75.
4. Generate processing elements from 1 to 20.
5. Create tasks of each users.

6. Initialize the Queuelength of each processing element=0.
7. Now, Processing elements registers their information to Grid Information Server(GIS).
8. GIS checks the availability of processing elements and users send requests to processing elements to send its characteristics.
9. Afterwards, users are waiting to receive processing elements characteristics.
10. After receiving the processing element characteristics, the users find the
    processing element with minimum Queuelength.
    a) Initially assign the Queuelength of first processing element to the temporary variable (temp).
    b) for (i=2;i<total processing elements;i++)
    c) Check IF temp > Queuelength of next element THEN
       temp contains Queuelength of next processing element ELSE there is no change in temp value.
11. Allocate the task of user to Processing element (PE) with minimum Queuelength.
12. Queuelength of Processing element(PE) will become
       Queuelength PE=Queuelength PE +1.
13. Print the Submission time of User's task to the allocated processing element(PE).
       Submission time_task=GridSim.clock( ).
14. Nextuser=Nextuser+1.
15. Print the Execution time of user's task.
       Execution time_task=GridSim.clock( )-Submission time_task.
16. After Completion of Execution of above task. Now, Check for the arrival of any new task T' from processing element PE'.
17. IF no task arrives to the processing element PE' THEN go to step 19.
18. Queuelength of processing element PE' will become Queuelength PE'=Queuelength PE' -1.
19. Repeat steps 10 to 16 until Nextuser<=Number of users.
20. Print the Execution time of all tasks.

## 5. SIMULATION & RESULTS

This Chapter covers the different parameters considered for simulations and results analyzed through simulations for Dynamic Approach.

### 5.1 SIMULATION PARAMETERS

The performance of proposed algorithm under different system parameters using GridSim toolkit are studied. Following parameters are used during simulation of Dynamic load balancing algorithm:

*Table 5.1: Simulation parameters*

| | |
|---|---|
| **Simulation Runs** | **4** |
| **No. of Processing elements** | **5 – 20** |
| **No. of Users** | **10 – 75** |
| **No. of tasks** | **10 – 75** |
| **Gridlet Size (In MI)** | **10,000,000 - 750,000,000** |
| **Processing Power of processing elements ( In MIPS)** | **100 – 400** |

The Execution time of tasks corresponding to different users using Dynamic Approach and Static Approach is shown in Figure 5.1 and Figure 5.2. The graph shows the Execution time of tasks under Static approach is more than that of execution time of tasks with Dynamic Approach.

### 5.2 SIMULATION RESULTS

The Execution time of Dynamic Approach are analyzed for various users and processing elements for number of Simulations.

*Table 5.2: Execution times of users when number of processing elements = 3*

| | **Processing Elements =3** | | | |
|---|---|---|---|---|
| **Number of Users** | 10 | 25 | 50 | 75 |
| **Execution time of Dynamic Approach** | 2376104 | 11318257 | 35760370 | 82361354 |
| **Execution time of Static Approach** | 2980199 | 15610357 | 46021349 | 89670786 |

ISSN: 2321-8169
2572 – 2579

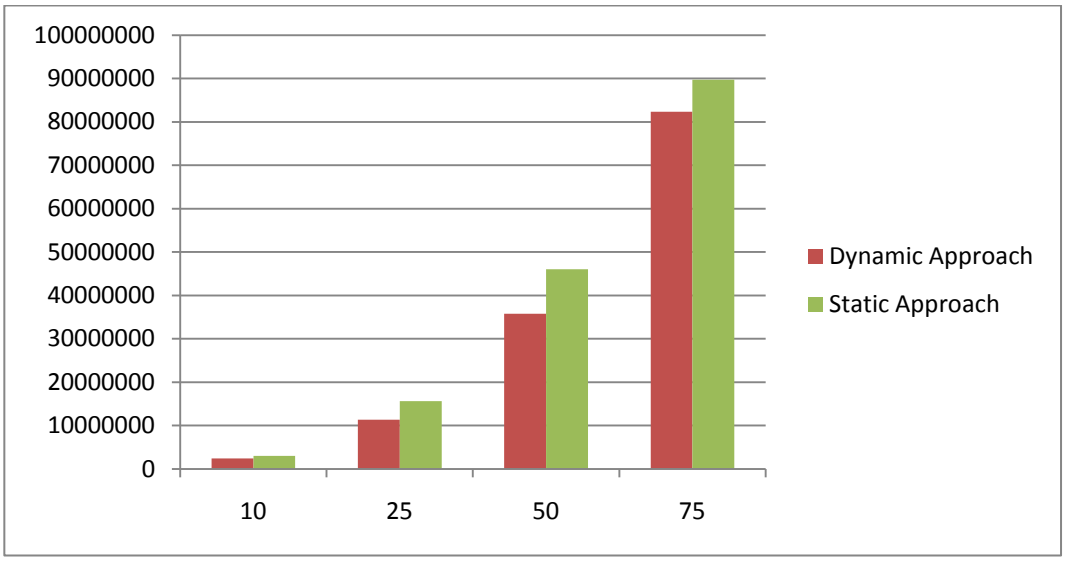


*Figure 5.1: Execution times of users when number of processing elements = 3*

*Table 5.3: Execution times of users when number of processing elements = 5*

| **Processing elements=5** | | | | |
|---|---|---|---|---|
| **Number of users** | 10 | 25 | 50 | 75 |
| **Execution time of Dynamic Approach** | 1753265 | 7312567 | 21678340 | 53612308 |
| **Execution time of Static Approach** | 2374163 | 8289650 | 27469234 | 59234871 |

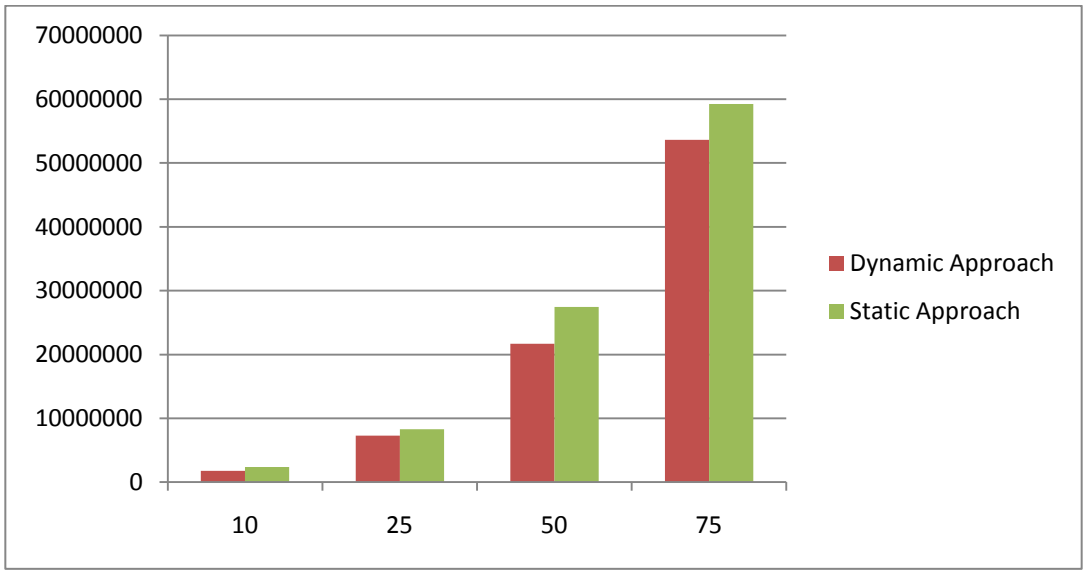


*Figure 5.2: Execution time of users when number of processing elements = 5*

ISSN: 2321-8169
2572 – 2579

The execution time of tasks corresponding to different processing elements using Dynamic Approach and Static Approach is shown in Figure 5.3 and Figure 5.4. The graph shows that execution time of tasks under Dynamic Approach is still less as compared to Static Approach even when number of processing elements are increased due to selection of only those processing elements which has minimum load.

*Table 5.4: Execution times of processing elements when number of users = 25*

| **Number of Users = 25** | | | | |
|---|---|---|---|---|
| **Number of PE's (Processing elements)** | 5 | 10 | 15 | 20 |
| **Execution time of Dynamic Approach** | 7376540 | 4523189 | 3741653 | 3467251 |
| **Execution time of Static Approach** | 8256471 | 5824198 | 4672587 | 4359823 |

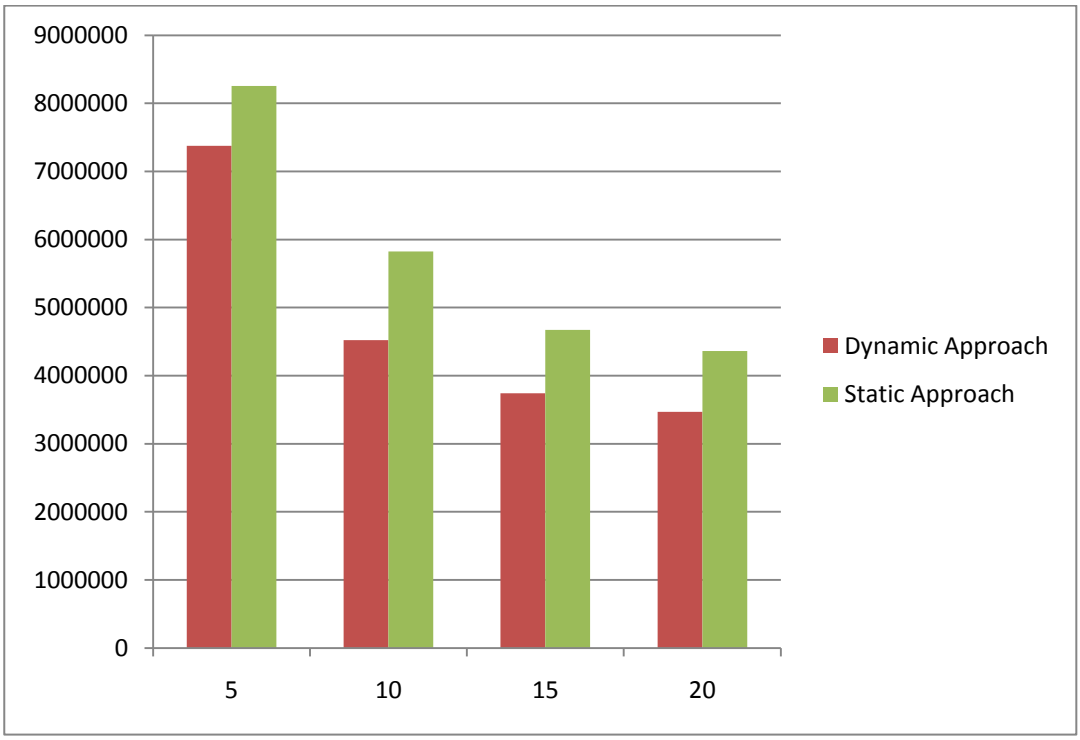


*Figure 5.3: Execution times of processing elements when number of users= 25*

*Table 5.5: Execution times of processing elements when number of users = 50*

| **Number of Users = 50** | | | | |
|---|---|---|---|---|
| **Number of PE'S (Processing elements)** | 5 | 10 | 15 | 20 |
| **Execution time of Dynamic Approach** | 22674905 | 13570349 | 10116890 | 83564216 |
| **Execution time of Static Approach** | 26537828 | 17823487 | 16583891 | 89346271 |

ISSN: 2321-8169
2572 – 2579

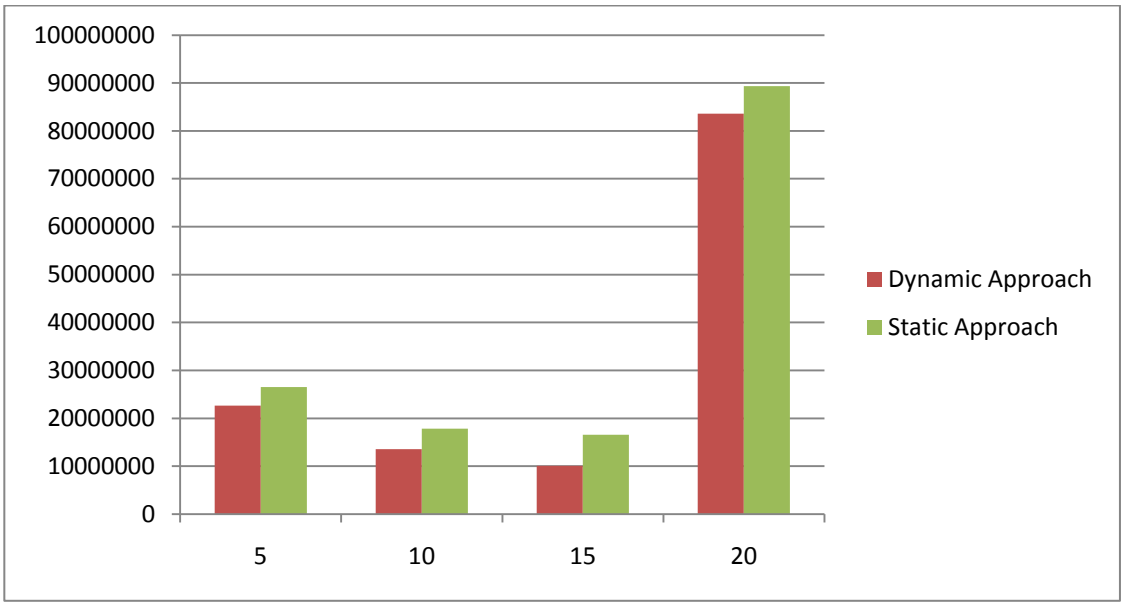


*Figure 5.4: Execution times of processing elements when number of users= 50*

The results show that Dynamic Approach is better than the Static Approach in all scenarios.

## 6. CONCLUSION AND FUTURE SCOPE

### 6.1 CONCLUSION

In Grid Environment, poor performance results due to uneven distribution of load among nodes in the system. Therefore, to fully exploit the computing power of such systems, it is crucial to employ a judicious load balancing strategy for proper allocation and sequencing of tasks on the computing nodes.

In this dissertation, effect of load balancing on tasks in terms of execution time is analyzed. Results show that the execution time of Dynamic Approach is less as every time the processing element with minimum queue length is selected for execution as compared to the execution time with Static Approach. The algorithm is tested under various load conditions in terms of task length varying from 10,000,000 - 750,000,000 (MI). The performance of Dynamic Approach is also better when the system is lightly loaded in terms of increasing the processing elements and keeping the number of users as fixed.

### 6.2 FUTURE SCOPE

The Simulations are performed for number of users and processing elements and results shows that our Dynamic Approach Algorithm works better than static approach. This dissertation work can later be extended for different simulation parameters and more better load balancing results can be achieved.

## 7. REFERENCES

[1] Yulai Yuan, Yongwei Wu, Guangwen Yang, and Weimin Zheng," Adaptive Hybrid Model for Long Term Load Prediction in Computational Grid," 8th IEEE International Symposium on Cluster Computing and the Grid, pp.340-347, August 2008.

[2] Youchan Zhu, Lei An, Shuangxi Liu," A Resource Discovery Method of Grid Based on Resource Classification," Proceedings of First International Conference on Intelligent Networks and Intelligent Systems, pp. 716-719, August 2008.

[3] C. Xu and F. Lau, "Load Balancing in Parallel Computers: Theory and Practice," Kluwer, Boston, MA, 1997.

[4] Yajun Li, Yuhang Yang, and Rongbo Zhu," A Hybrid Load Balancing Strategy of Sequential Tasks for Computational Grids," International Conference on Networking and Digital Society (ICNDS), 2009.

[5] M. Baker, R. Buyya, and D. Laforenza, "Grids and grid technologies for wide area distributed computing," International Journal of Software: Practice and Experience (SPE), vol. 32(15), 2002.

[6] B. Yagoubi, and M. Medebber, "A load balancing model for grid environment," Proceeding of 22nd International Symposium on Computer and Information Sciences (ISCISC 2007), pp. 1-7, 7 November 2007.

[7] C. Kim and H. Kameda," An algorithm for optimal static load balancing in distributed computer systems," IEEE Transaction on Computers, vol. 41(3), pp. 381-384, March 1992.

[8] K. Lu, R. Subrata, and A. Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites," Journal of Computer and System Sciences, vol. 73(8), pp. 1191-1206, December 2006.

[9] K. Lu, and A. Zomaya, "A Hybrid Policy for Job Scheduling and Load Balancing in Heterogeneous Computational Grids," Proceeding of 6th International Symposium on Parallel and Distributed Computing, pp. 19-26, 5 July 2007.

[10] M. Dobber, R. Mei, and G. Koole," Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison," IEEE Transaction on Parallel and Distributed Systems, vol. 20(2), pp. 207- 218, February 2009.

[11] J. Cao, D.P. Spooner, S. A. Jarvi, and G.R. Nudd," Grid Load Balancing Using Intelligent Agents," Future Generation Computer Systems, vol. 21(1), pp. 135-149, January 2005.

[12] Bin Lu, and Hongbin Zhang," Grid Load Balancing Scheduling Algorithm Based on Statistics Thinking," IEEE 9th International Conference, pp. 288-292, 2008.