ISSN 2320 - 2602
Volume 2, No.7, July 2013
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/pdfs/2013/ijacst03272013.pdf

CPU Load Predictions on the Computational Grid Using Distance Based Algorithm

Rahul Nayak, Prof. Rashmi Gupta
TIT,RGPV Bhopal(M.P.), India ,rahuln61@gmail.com
TIT,RGPV Bhopal(M.P.), India, rgupta6773@gmail.com

ABSTRACT

Need of computational resources and consumers of these resources are increases rapidly, to fulfill the desired computational demands up-gradation are necessary time to time. Efforts to obtain high computational units using grid computation are possible now in these days. But due to incremental request increases the complexity of grid additionally many problems arises on hardware as well as software level conflicts occurs. To prevent these faults in the complex grid predictive methods are helpful for plan and prevent these problems in advance. This paper provide the CPU load prediction method over multiple clustered grid environments, where more than one CPU is participating on the grid computation and load parameters are calculated, to justify the proposed method and their performance parameters.

Key words: Grid Computing, CPU Load Prediction, one step backward, BPN, predictive algorithm.

1. INTRODUCTION

Grid is an administrative collaboration to solve the similar computational goal in efficient manner, where task is discredited and organized in divide and conquer manner. Grid computing in general is a special kind computer organization of parallel computing that relies on complete computers (with on-board CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus [1].

Grids are used for many application areas, such as physics, bioinformatics, earth sciences, life sciences, finance, space engineering, etc. Grids have strengthened the change in science and engineering of complementing the theory and experimentation with computational and intensive discovery [2].

Grids are collections of resources ranging from clusters to supercomputers. Many types of jobs have been tried on grids, from sequential to parallel, from compute-intensive to data-intensive, and from massive coordinated applications to bags of independent tasks. A typical grid-based experiment requires the repeated execution of a computational task on different sets of input parameters or data; thus, many grid workloads are dominated by applications with a bag of tasks structure. The grid resource providers and the grid resource consumers (users) are of 7 different entities. The grid resource providers decide on the resource management policies, and provide only minimal, generic job management services. To simplify management, Virtual Organizations (VOs) group administratively users or resource providers.

The distributed computing environments to which most users have access consist of a collection of loosely interconnected hosts running vendor operating systems. Tasks are initiated independently by users and a rescheduled locally by a vendor supplied operating system. There is no global scheduler that controls access to the hosts. As users run their jobs the computational load on the individual hosts changes over time. Deciding how to map computations to hosts in systems with such dynamically changing loads (what we will call the mapping problem) is a basic problem that arises in a number of important contexts, such as dynamically load-balancing the tasks in a parallel program, and scheduling tasks to meet deadlines in a distributed soft real-time system [3].

Each node on the grid receives a piece of the problem, which consists of a collection of original problem cells (OPCs). An OPC is the smallest piece into which the problem is divided, and each one needs to communicate and share data with its neighbors. Optimal Grid automates this communication and attempts to minimize the amount of network communication needed to solve a problem. When the program for the application is loaded, the middleware automatically partitions the problem using the following procedures:

1. Determine the complexity.
2. Identify the number of nodes available.
3. Use algorithms to predict the optimal number of grid nodes needed to solve the problem.

4. Optionally interact with the user to divide the problem into an optimal number of pieces. Whether the user or Optimal Grid partitions the problem, the middleware predicts the computation time for the problem.

5. Partition the application data into OPCs. Allow the user the option to customize the data. In assessing stress on an airplane wing, for example, the user might decide to remove one or two rivets from a particular place.

6. Launch the program.

In this section of paper provides the general introduction about grid computing in the next section paper contains the background work of experimental setup.

1.1 EXPERIMENTAL SETUP

In this section we include the configuration and model by which we simulate the compound working of the system more over it here we provide the computational parameters by which we analyse the system which make us enable to predict the future work load of the complete grid nodes.

First of all we create a grid environment to perform the calculation correctly. Therefore we connect ten traditional computer systems via a LAN. And the systems are contains new installed operating systems and not any software's and not any other kind of software's are installed. After installation of systems we check the communication between them using "PING" command and found all the systems are work over network properly. The network organization of the connected systems is given using Figure (1).

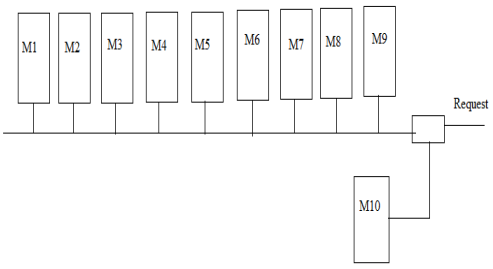

Figure 1: Shows the network arrangement of systems

All the systems denoted by M1….M9 are the host connected in parallel manner and the central monitor is installed on the top of the system M10.The connected machine contains a client agent which is responsible to obtain the host CPU load and the generated loaded is forwarded to M10, where M10 is used as server system and responsible to compute and predict the upcoming load on server according to time. For that purpose a server agent is installed on the monitor system.

1.2 CLIENT AND SERVER AGENT ARCHITECTURE

Client and server agents are the software programs where client agent is just collect the CPU load on the installed machine and using a simple multithreaded server socket program send it to the server end where server machine (monitor)identify performance parameters using their IP address and store into a local database(created using SQL server).

Server program access these data for training of machine learning algorithms and produces the upcoming loads using these trained algorithms. Trained algorithms are able to predict the upcoming accurate during experiments we found that the predication of algorithms are depends upon the algorithm and training.

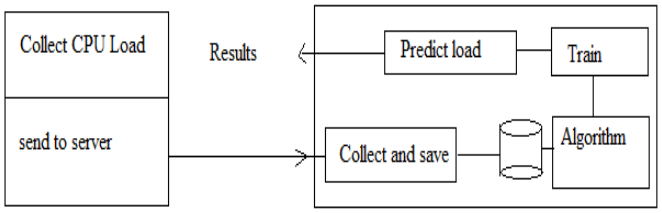

Figure 2: Shows the application architecture

In the above given diagram we prepare a master system and others are slave system master systems are responsible to accept the user request distribute the task and gather the data samples to make analysis. The complete process is performed in some basic steps these steps are:

1. A client application on all the server nodes.

2. Install server application on the master system.

3. Client application is responsible to collect the node load from the server system.

4. Use a simple multi-threaded server application to gather load parameters from server machine actually here we implement a client server program to get data from client machine.

5. The data found from different nodes is pre-process first and updated over data base. These load parameters that are generated during request is working as training data for our designed algorithm and BPN algorithm.

6. Now the data is ready to analysis. Finally on generated data set we apply BPN algorithm first and calculate the performance parameters.

7. After that we apply our designed algorithm to get performance parameters.

8. Finally we produce the predictive results.

2. ALGORITHM USED

In this section we provide the algorithms basic steps to analyses the produced load data to the system. Here we consume two algorithms for forecasting load on CPU first BPN and second our proposed algorithm.

Input: training samples (Dataset)

Output: trained model

1. **Initialize two vectors one input and hidden unit and second output unit.**
2. **Here first is a two dimensional array W_{ij} is used and output is a one dimensional array Y_i.**
3. **Initial weights are random values put inside the vectors after that the we calculate the output as**

$$x_j = \sum_{i=0} Y_i W_{ij}$$

Where y_i is the activity level of the j^{th} unit in the previous layer and W_{ij} is the weight of the connection between the i^{th} and the j^{th} unit.

4. **Next, activity level of y_i is calculated by some function of the total weighted input.**

$$y_i = [\frac{(e^x - e^{-x})}{(e^x + e^{-x})}]$$

5. **When activity of the all output units have been determined, the network computes the error E,**

$$E = \frac{1}{2} \sum_i (y_i - d_i)^2$$

Where y_i is the activity level of the j^{th} unit in the top layer and d_i is the desired output of the j_i unit.

6. **Compute Error Derivative (EA) is the difference between the actual and the desired activity:**

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j$$

7. **Calculate the error changes as the total input received by an output changed**

$$EI_j = \frac{\partial E}{\partial X_j} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_j} = EA_j y_j (1 - y_i)$$

8. **Calculate the error changes as a weight on the connection into an output unit is changed:**

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial X_j} = \frac{\partial X_j}{\partial W_{ij}} = EI_j y_i$$

9. **Calculate the overall effect on the error:**

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_j EI_j W_{ij}$$

2.1 Proposed algorithm

Our proposed algorithm for prediction is working in two different phase's first training from the previous data and seconds the testing on real data.

Learning:

Input : training samples

Output: trained model

1. **read data set from database**
2. **Find the decision values from each attribute using the given formula**

$$D = \frac{1}{total\ number\ of\ values} \sum unique\ value$$

3. **Convert all attributes into binary**
4. **If value <= Decision parameter then**
   Value=0
   Else
   Value=1
   End if
5. **Now compute distance using hamming distance**

Prediction:

Input: load previous

Output: load one step ahead

1. **Collect data and convert into binary data**
2. **Get all rows which is better match with the pattern**
3. **Find average to get value**

3. RESULTS

After implementation we got the performance of results which is given in two parts first algorithms performance and second the load parameters predictions.

Finding load parameters on three different servers on the experimental setup is given as, the load calculated is provided on the basis of 5000 instances in data table for training of BPN and our proposed technique. The obtained results of BPN are given. The below given table shows the results obtained after training of algorithm as predictive values for 7 machines connected in network setup.

(Machines)	M1			M2			M3			M4			M5			M6			M7		
	R	B	P	R	B	P	R	B	P	R	B	P	R	B	P	R	B	P	R	B	P
5	2.3	3.2	2.1	4.2	7.1	3.9	7.3	6.2	7.3	2.4	2.5	3.9	19.2	10.2	29.1	28.1	21.9	27.1	12.1	29.1	12.9
10	4.6	2.1	3.9	2.5	2.9	2.4	3.2	2.8	4.5	7.4	7.0	2.1	6.7	5.8	5.9	14.4	13.1	14.4	23.7	14.1	22.7
15	5.3	5.1	3.2	2.1	2.1	2.1	2.2	2.6	1.5	19.3	3.2	18.2	4.8	8.1	5.1	4.3	4.6	3.7	3.6	3.7	3.1
20	4.2	3.8	4.7	5.1	4.2	5.8	3.2	3.6	7.8	32.2	32.3	28.2	7.3	3.8	7.1	4.8	4.2	3.9	6.1	5.2	6.9
25	5.6	5.1	5.2	4.9	2.4	7.3	7.5	4.3	3.4	1.2	1.7	.6	7.1	7.3	6.5	6.2	4.9	6.1	6.3	6.3	2.5
30	22.7	19.2	21.1	2.2	22.1	2.4	4.5	3.4	3.5	1.7	1.8	.2	3.8	4.2	3.5	7.1	8.2	6.9	7.2	6.2	6.9
35	3.7	2.2	10.2	2.9	4.2	2.4	7.6	3.7	6.5	5.9	7.1	2.3	3.2	3.9	8.1	4.1	6.9	7.1	8.3	5.8	6.2
40	8.3	8.1	8.4	4.4	3.9	4.1	4.1	4.3	8.6	4.3	3.8	4.1	3.9	3.1	2.8	4.7	5.5	10.2	2.3	1.7	2.6
45	12.1	12.5	12.5	8.6	9.1	8.5	4.5	8.2	5.3	7.2	9.8	2.6	3.3	2.6	3.1	4.9	3.4	4.3	4.2	4.2	3.9
50	2.4	1.9	2.2	21.1	20.8	19.2	4.5	2.3	4.1	6.6	6.1	6.6	7.1	6.8	8.8	7.9	3.1	2.2	6.3	4.7	8.4
55	20.1	19.2	20.6	28.1	27.9	28.4	4.4	23.2	5.2	8.4	5.1	7.3	2.9	6.4	5.3	3.1	7.9	8.4	2.7	3.2	2.8
60	29.9	28.1	29.4	42.9	39.8	41.2	8.6	41.1	3.1	26.2	21.1	21.1	12.5	23.1	4.4	4.5	6.2	2.3	5.2	4.1	6.5

Table 1: Shows load comparison on all 7 machine using Real, BPN and Propose Distance Based Method.

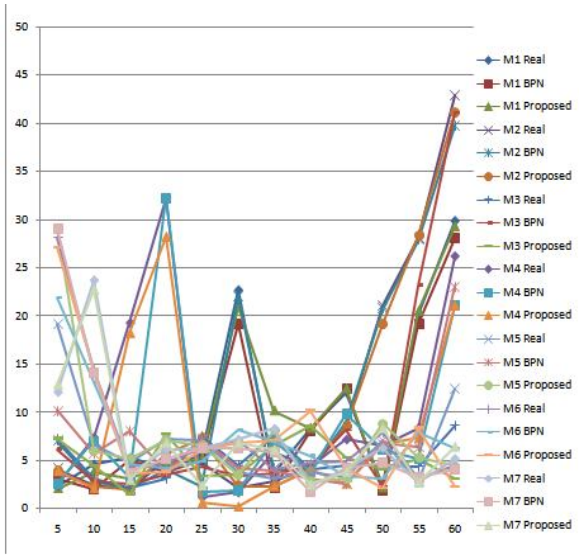

Figure 3: shows the CPU load on seven machines

In addition of that there are some additional parameters are obtained to get the performance of the system such as training time, prediction time by the algorithms which is listed in the blow discussion.

3.2 Training time: that is a measurement of time for performs training using the data available on database which is given in Figure (4).

No of rows	Proposed algorithm	Neural network
326	6 s	30 s
782	10s	40 s
1029	17s	47 s
2930	26s	52 s
3092	49s	59 s

Table 2 : shows Training time comparison between BPN and Propose Distance Based Method.

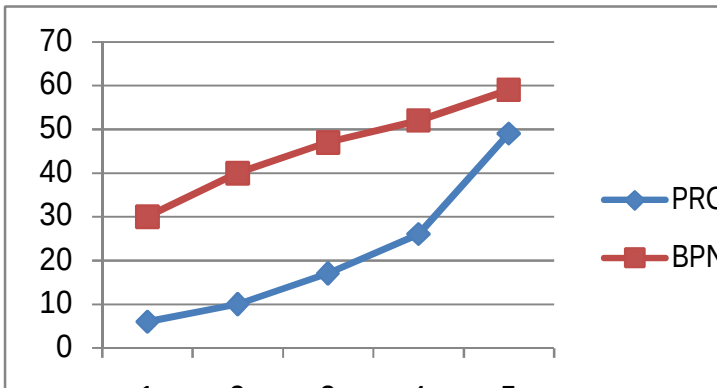

Figure 4: shows the training time of the algorithms

3.1 Decision time: decision time is a time required to predict values from trained data model.

No of rows	Proposed algorithm	Neural network
326	726 ms	38 ms
782	404 ms	42 ms
1029	217 ms	41 ms
2930	467 ms	82 ms
3092	493 ms	71 ms

Table 3: shows Decision time comparison between BPN and Propose Distance Based Method.

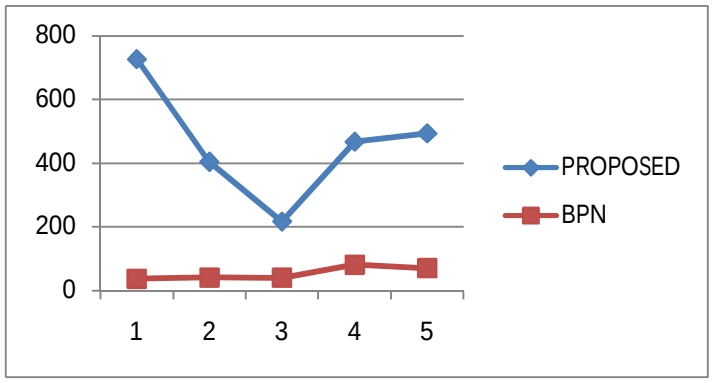

Figure 5: shows the decision time

The proposed system is providing the slow results then the BPN in decisions.

4. CONCLUSION

Load forecasting in grid environment is complex and interesting task. In this experiment we provide the calculations based on local setup of the computers and results are evaluated. To compute the load we apply two different algorithms and achieve results based on our experiments. During analysis we found that loads and load parameters are not only depends on previous patterns of load but CPU load also depends on other factors such as network, number of processes running on, memory available, background processes, frequency of requests, application servers and supporting applications too. Here we use sever machine for experimentations and evaluate the load on servers. after performance analysis we found the our algorithm produced much accurate results for predicting CPU load but training is completed in few time but to make a decision it consumes more time them other algorithms. In future we proposed and implement the same algorithm by which we get more accurate and efficient results.

REFERENCES

1. Manish Parasher, Craig A Lee**, "grid computing: introduction and overview"** IEEE 93(3):479–484, March 2005.
2. Alexandru Iosup and Dick Epema **,"Grid Computing Workloads: Bags of Tasks, Workflows, Pilots, and Others"**Parallel and Distributed Systems Group, Delft University of Technology, Mekelweg 4, 2628CD Delft, the Netherlands
3. Peter A. Dinda **"The Statistical Properties of Host Load (Extended Version)"** March1999, CMU-CS-98-175,School of Computer Science Carnegie Mellon University Pittsburgh, PA15213
4. I. Foster and C. Kesselman**, "The Grid: Blueprint for a New Computing Infrastructure",** Morgan Kaufmann Publishers, San Fransisco, CA, 1999.
5. P.A. Dinda, D.R. O'Hallaron**, "Host load prediction using linear models,"** Cluster Computing 3(4), pp. 265-280, 2000.
6. L. Yang, J.M. Schopf, and I. Foster**, "Conservative scheduling: Using predicted variance to improve decisions in dynamic environment,"** Supercomputing'03, pp. 1-16, 2003**.**
7. P.A. Dinda, **"A prediction-based real-time scheduling advisor,"** Proc. 16th Int'l Parallel and Distributed Processing Symp.(IPDPS 2002), pp. 35-42, 2002.
8. D. Lu, H. Sheng, and P. Dinda**, "Size-based scheduling policies with inaccurate scheduling information,"** 12th IEEE Int'l Symp. onModeling, Analysis, and Simulation of Computer and Telecommunications Systems,pp. 31-38, 2004.
9. S. Jang, X. Wu, and V. Taylor**, "Using performance prediction to allocate grid resources,"** Technical report,GriPhyN 2004-25, pp. 1-11, 2004.
10. L. Yang, I. Foster, and J.M. Schopf**, "Homeostatic and tendency-based CPU load predictions,"** Int'l Parallel andDistributed Processing Symp.(IPDPS'03), pp. 42-50, 2003.
11. R. Wolski, N. Spring, and J. Hayes**, "Predicting the CPU Availability of Time-shared Unix Systems on the Computational Grid,"** Proc. 8th IEEE Symp.on High Performance Distributed Computing, pp. 1-8, 1999.
12. M. Swany and R. Wolski**, "Multivariate resource performance forecasting in the network weather service,"** Supercomputing'02, pp. 1-10, 2002.
13. R. Wolski**, "Dynamically forecasting network performance using the network weather service,"** Journal ofCluster Computing, Vol.1, pp.119-132, 1998.
14. R. Wolski**, "Experiences with predicting resource performance on-line in computational grid settings,"** ACM SIGMETRICS Performance Evaluation Review, Vol.30,No.4, pp. 41-49, 2003.
15. P.A. Dinda**, "The statistical properties of host load,"** Technical report, CMU, pp. 1-23, 1998**.**
16. S. Akioka and Y. Muraoka**, "Extended forecast of CPU and network load on computational grid,"** 2004 IEEE Int'l Symp.on Cluster Computing and the Grid, pp. 765-772,2004.