Journal of Theoretical and Applied Information Technology
10th July 2013. Vol. 53 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PREDICTIVE LOAD BALANCING FOR DATA MINING IN DISTRIBUTED SYSTEMS

[1]M.EUGIN LILLY MARY, [2]PROF.DR.V.SARAVANAN

[1] Research Scholar, Bharathiar University, Coimbatore – 641 046, India
[2] Professor & Director, Department Of Computer Applications,
Sri Venkateswara College Of Computer Applications And Management,
Ettimadai, Coimbatore - 641 105, India

E-mail: [1] eugine_jeyaraj@yahoo.co.in, [2] tvsaran@hotmail.com

ABSTRACT

Load Balancing is the key attribute in distributed systems to ensure fast processing and optimal utilization of hardware components. The processing elements of varying hardware architecture and computing capabilities in a shared network system constitute the distributed system. In such a heterogeneous environment, load balancing and scheduling become complex and complicated. Dynamic feedback based techniques, predicting the processing time of a task in a node, will be a promising solution to resolve variations of this ubiquitous system. A collaborative agent system comprising mobile and stationary agents captures the feedback from the contributing nodes and the prediction for the proceeding time epoch is made by minimizing the gap between the predicted processing power and the actual processing power of the node in the preceding epoch. As a result, the actual processing time of a task recorded is very close to the predicted time, thereby reflecting the variations of the distributed system. This predictor enables the construction of load balancing algorithms suitable for a heterogeneous distributed environment.

Keywords: *Predictive Processing Power, Actual processing power, Next_Predicted_Time_Ratio, Actual_Time_Ratio, Rate of mining.*

1. INTRODUCTION

Today information technology has spread its wings to areas like molecular biology, weather forecast, cosmology, space research, nano technology, insurance, sensor networks, risk analysis, text mining, knowledge recovery etc. With the advent of massive storage devices the business industry stores and manipulates the data for their business activities and needs to process large volumes of data in the order of terabytes. At times the data sets to be processed may arrive at high speed in the form of streams, requiring fast and secure processing. Such activities include pattern recognition, identification of the underlying association, fraud identification, fault detection, etc., requiring very high computing power that can be achieved by utilizing a set of interconnected nodes forming clusters. These clusters constitute varying hardware resources, software sophistication, and quality of connectivity, etc. A large cluster is capable of contributing on par with a supercomputer at a fraction of the cost. [10]

The challenge is to distribute the tasks to the available computing nodes so that none of the nodes is overloaded or underutilized, and this art is termed as load balancing. Therefore, load balancing is the key attribute in a distributed system, to ensure fast processing and good utilization of the hardware components. Simple load balancing methods include the round robin, randomization approach, minimum connections, weight age methods, minimum misses, hashing etc., that are easy to implement, but are suitable only for homogeneous systems. The strategies can be divided into static or dynamic. Static load balancing distributes the tasks across the computing elements before execution, and the load distribution remains unchanged. In a distributed environment the state of the individual nodes varies and the knowledge of this variation during runtime forms the basis of dynamic load balancing. Therefore, dynamic load balancing is termed as an active technology, which provides the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

art of transforming the network traffic and scheduling. In addition, the quality of services viz., scalability, quick response time, failure handling and fault tolerance can be embedded in the dynamic load balancing strategies. A detailed analysis of the general load balancing algorithms has been made in [16]. A brief over view of the techniques extracted from the literature is given in Section 2. Although a plethora of load balancing policies are in use, simple methods [11] are adopted since they are easy to implement. However, these methods work well for homogeneous systems, and do not perform efficiently for heterogeneous elements or tasks requiring varying processing times. The present scenario is to manage the clusters of varying computing capabilities. To scale up such systems to perform at high level availability, and flexibility, the need arises for more sophisticated software architectures and adaptive methods. The adaptive load balancing strategies suitable for heterogeneous environments are complex and complicated, and need the prediction of processing time for any task to be processed in a computing node. Also the prediction policies are application specific. The existing scenario uses a generalized system for all types of applications [20]. As a result, the distributed network experiences excess traffic due to communication over load, but has very limited abilities to handle overloads, load imbalances, and compute-intensive transactions like cryptographic applications, data mining applications and multimedia processing which involve huge data.

The agent based systems emerging recently are able to resolve the above mentioned issues. An agent is an autonomous system capable of performing action on behalf of its users. Agents are capable of providing optimized services according to the changing environment. The characteristics of an agent include pro activity, autonomy, goal oriented local views and decentralization [2]. An agent system itself possesses intelligence. In addition, it is capable of making decisions on its own. This social ability enables the multi agent system to handle operations which are impossible for a single agent [4]. The feedback from the resources in the distributed environment can thus be effectively obtained in a dynamic environment adopting MAS, and this feedback forms the basis for predicting the computing time required for the next task to be processed. The computation intensive mining operation can be processed at the nodes by means of the stationary agents residing there. A dynamic prediction based collaborative agent system has been proposed in this paper, to achieve load balancing in a ubiquitous system employing mobile and stationary agents. The rest of the paper is organized as follows. Section-2 narrates the related work, Section-3 the proposed approach, Section-4 proposed system over-view, Section-5 the experiments and results, and Section-6 gives the conclusion.

2. RELATED WORKS

Load balancing strategies should be devised to get a quick response from the clusters, and effective utilization of the hardware components. A plethora of load balancing methodologies are discussed in the research papers. In this section, brief outlines of the research techniques are discussed.

Load balancing may be static or dynamic. Static load balancing methods adopt a predefined strategy for the distribution of loads. [5]. Iterative load balancing methods are discussed in [19]. A comparative study of the static and dynamic load balancing methodologies has been carried out by Willebeek –Lemair and Reeves [18]. Load balancing becomes complex and complicated in a heterogeneous environment. CFS is a framework which allocates the number of virtual servers, in proportion to the capacity of the computing nodes [8]. Though this system provides a simple solution to shed the overloaded nodes, thrash occurs when some of the virtual servers are removed. Adler et al [1] proposed a framework which organizes the nodes as the leaves of trees that in turn, effect load balancing based on the joining or departure of the leaf nodes. However, aspects, such as the heterogeneity of the nodes and varying load distribution have not been discussed. Karger and Ruhl[6] illustrated methods for balancing loads in a heterogeneous environment, by reassigning the loads of the heavily loaded nodes to the lightly loaded ones. There is a possibility of bounds at the heavily loaded nodes [7] upon load movement, and it is unclear whether the techniques are efficient in practice.

The methods discussed so far, adopt the message exchange paradigm to collect the information regarding the loads. The process of the message passing paradigm cause network traffic, [14] and this can be resolved by employing agents. Recently, dynamic load balancing approaches using agent systems have been evolved. Agents are capable of their performing various kinds of operations on behalf of its users. They are employed to move data, intermediate results and models between clusters so that the network load

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

gets reduced. [10] The system proposed by Frank Lingen uses mobile agents for the execution of jobs submitted by hand held devices. A combination of two agents LGA and RLBA has been engaged in a framework [7], in which the LGA collects the load status, while the RLBA enables load balancing when a node is overloaded. Ezhumalai et al [7] proposed a framework which consists of a monitor agent to collect information from the nodes, a supervisor agent to analyse the overloaded nodes, and a locate agent to locate the node with a light load. However, the experimental setup for this framework has not been discussed. A framework named EALMA utilizes mobile agents for updating the load information of the cluster [9]. In an agent based load balancing architecture, TRAVELLER, [15] the resource brokers are responsible for collecting the load status of the computing nodes of the distributed environment. The framework, MESSENGERS [19] supports load balancing and dynamic resource utilization with the aid of agents. In the agent based framework, PMADE [14], the agents residing at the computing nodes initiate the load balancing, as and when a node is overloaded. MAS [4] is a framework which organizes the migration of mobile agents to cope with the current load status. Flash [12] is designed to apply load balancing in a cluster system. In the above mentioned agent based frameworks the changing load conditions of the nodes in the distributed environment form the basis for load balancing.

The data mining operations to be performed are both process intensive and data intensive, involving a huge volume of data and varying operations on those data. The agents are capable of performing mining operations on behalf of their users.

The cluster based approach; CLARA (Cluster Based Active Router Architecture) [17] performs multimedia transcoding tasks on a cluster of nodes instead of on the router. It is shown experimentally that prediction based 1oad balancing performs efficiently. A prediction based 1oad balancing, based on the confirmation mechanism discussed in [20] performs the adjustment of loads, when the predicted measure and observed value differs significantly. It is shown experimentally that only 4% of the processes need adjustment while the remaining 96% of the processes are in the steady state. But this method holds good for the uniform distribution of loads. A Grey Dynamic model based load balancing mechanism [GMLBM] has been proposed in [12], which predicted the load data according to the grey theory.

The overview of the dynamic prediction based load balancing system for processing datasets in a cluster based distributed environment, employing a collaborative agent system is reported in section 3 and the architecture over view is given in section 4.

3. THE PROPOSED WORK

The main focus of this research is to propose a centralized prediction-based dynamic load balancing algorithm that is suitable for the distributed heterogeneous environment. 'Heterogeneous environment' refers to the cluster of computing nodes of varying computer hardware architecture, computing capabilities, operating system and resource availability etc. The load balancers in existence, such as the round robin, tree based algorithms, or data parallel applications are suitable only for homogeneous tasks.

The data mining techniques to be implemented in the nodes are not only data intensive but computation intensive too. For the same size of input stream the processing time differs with respect to the methodology. Hence, the heterogeneity of the tasks plays an important role in forecasting the time units required to process a particular data stream in a node. In such a scenario, the prediction based load balancing strategies can enable effective load balancing and scheduling.

3.1 Constructing a Predictor Table

The objective is to process the stream of data on a cluster of computing nodes. Predicting the execution time of this parallel processing on a heterogeneous system becomes complex and complicated. There involves resolving two major issues. First, to construct a predictor table with initialized 'prediction values' when each of the computing nodes possesses varying capabilities. Second to forecast and update the 'rate of mining operation' on the individual nodes. 'Rate of mining' refers to the size of streams processed in a unit of time.

3.2 Initializing the Predictor Table

The open issue is how to initialize the prediction value. In [1], a prediction based load balancing strategy has been analyzed, where the performance of the load balancing strategies adopted for transcoding multimedia units in a cluster based computing environment has been compared, and it is recommended that prediction based load balancing strategies viz., prediction

Journal of Theoretical and Applied Information Technology
10th July 2013. Vol. 53 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

based least load first, and prediction based adoptive portioning algorithms yield more throughput and reduced jitter than the corresponding non predictive algorithms.

Experiments have been conducted to obtain an estimated value of the initial prediction time on a cluster of computing nodes in which a set of

N = {N1,N2,N3,…Nn}

'n' nodes are connected via network,

At any epoch, Tk a subset Nn are ready for handling the instructions.

Nn ={Ni\Fault index(i) < >0}

At each node the analytical techniques are specially deployed for processing a subset of data streams. Let 'm' streams, each of various sizes are routed to all the 'n' computing nodes individually. The time taken for processing a data stream, stream (i,j) at node Ni against the stream size has been observed. Let size (i,j) be the quantum of stream processed at time(i,j); the initial prediction value for node Ni has been arrived at using the equation (1).

$$\text{Initial-Prediced\_Val(i)} = \sum_{j=1}^{m} \text{size(i,j)} \Big/ \sum_{j=1}^{m} \text{time(i,j)} \quad \text{.....(1)}$$

3.3 Forecasting and Updating the Rate of Mining

Based on the experimental values discussed in the previous section, the estimated time for processing a stream in Node Ni, has been initialized. A predictor which is being incrementally built narrows the gap between the 'predicted time of processing a stream' and the 'actual execution time discussed in this section. In a framework GMLBM, (Grey Dynamic Model Based Load Balancing), the predictors are constructed linearly by the method of least squares, using the recently available four sets of data. [13]. Bravier et al illustrated that building a predictor model using the canonical least squares would be computationally too expensive. [3] They designed the predictor that approximates the linear model. Experimental proofs were given by them, and it was illustrated that the predictor offered better performance than the linear model. In our study, the estimation and forecast of the prediction value is dynamically carried out as per the following algorithm, depicted in table 2, and the definition of the parameters involved is depicted in table 1.

The forecast table T_Predictor constructed is being updated at the end of every epoch. The Actual_Time obtained as feedback from the feedback module of the computing nodes forms the basis for forecasting the processing time of the proceeding datasets. The prevailing gap between the predicted time and the actual processing time gets narrowed. As a result, the rate of processing, the Predicted_Time_Ratio approaches the actual rate of processing the Actual_Time_Ratio. A graph illustrating this phenomenon is given in figure 1. It is aimed to slim out the deviation of the measures as time progresses.

Let this deviation of Predicted_Time_Ratio from Actual_Time_Ratio be diff (T),

T_k, Tk_{+1}, T_{k+2} be the consecutive epochs and it is clear that

diff (T_{k+2}) , < diff(T_{k+1}) < diff(T_k)

In the grey dynamic model based mechanism [3] the prediction has been arrived at by maintaining the recent four sets of data. In our study, the cumulative measure of the previous records and the current records is considered as the two attributes in deciding the forecast.

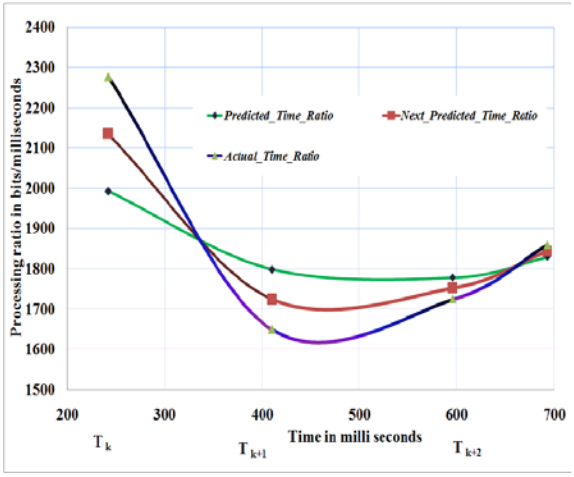

Figure 1: Chart illustrating the forecast

Journal of Theoretical and Applied Information Technology
10th July 2013. Vol. 53 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Definition of Parameters

Tot_Filesize	:	Total size of the datasets to be processed at that time epoch
T_{k+1}	:	Present time epoch
T_{k+1}	:	Proceeding time epoch
Ni	:	Node under consideration
j	:	The serial number of the task
Pre_Load_Stat(i)	:	The load forecasted at a node Ni
Predicted_Val(i,j, T_k)	:	Forecasted time of processing a task j at node Ni in the time epoch k
Predicted_Val(i,j, T_{k+1})	:	Forecasted time of processing a task j at node Ni in the time epoch k
Actual_Time(i,j, T_k)	:	Actual time observed for processing a task j at node Ni in the time epoch k
Predicted_Time_Ratio(i,j,T_k)	:	Predicted rate of mining for processing the task j at node Ni in the time epoch k
Predicted_Time_Ratio(i,j,T_{k+1})	:	Predicted rate of mining for processing the task j at node Ni in the time epoch k+1
Actual Time_Ratio(i,j,T_k)	:	Actual rate of mining observed for processing task j at node Ni in the time epoch T_k

4. PROPOSED SYSTEM OVERVIEW

The feedback being updated contributes a lot to the decision making process in dynamic load balancing. Thus, the forecast table being updated forms the basis for load balancing and scheduling. This can be achieved by employing a collaborative agent system, comprising of mobile and stationary agents. The proactive and reactive features of agents facilitate knowledge discovery, by effectively distributing it to the set of computing nodes.

The framework, Predictive Load Balancing (PLB), proposed in this paper, comprises a plethora of stationary agents and a mobile agent as depicted in figure 2.

The available computing power in the distributed environment is being categorized as monitoring and computing nodes. The monitoring node is responsible for the implementation of load balancing and scheduling policies, while the computing nodes perform the data mining operations. The monitoring node keeps track of the performance of the computing nodes, and suitably updates the predictor table, at the end of every epoch, as described in Section (3.3), and the history table as and when it is warranted.

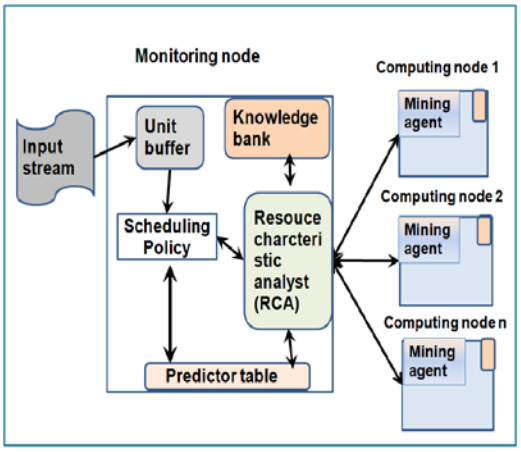

Figure 2: General Architecture

Let there be 'm' streams to be processed in 'n' computing nodes (m>n).

The various modules of the monitoring node performing load balancing and scheduling are described below.

Receiver module – collects and, sizes the tasks to be processed, grouped or sliced as the case may be, such that at any point of time, the size of the tasks should satisfy the condition in equation (2)

minsize < size (task i) < maxsize. ……. (2)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2 : Algorithm for constructing the forecast table

1. Collect the initial predicted value for node Ni, to initialize.
2. Compute the Predicted_Time_ Ratio for node Ni.
3. Fetch the actual time for executing a task j at node Ni.
4. Compute the Actual-Time_Ratio for executing the task j at node Ni.
5. Compute the difference between (2) and (4).
6. Arrive the Next-Predicted_Time_Ratio for processing the task(j+1) at node Ni in the proceeding epoch, by minimizing (5)
7. Assign the value obtained in (6) for the Predicted_Time_Ratio of processing the task (j+1) at node Ni for the proceeding epoch.
8. Repeat steps (2) to (7)

Input : Initial_Predicted_Val(i)
File_size(i,j, T_k)
Predicted_Val(i,j,T_k)

Output :
Predicted_Val(i,j,T_{k+1})

Initialise :
When T_k = 1

Predicted_Val(i,j,T_k) = Initial_Predicted_Time(i)
Predicted_Time_Ratio(i,j,T_k) = File_size(i,j,T_k) /Predicted_Val(i,j,T_k)
While (m > n)
Get Actual_Time(i,j,T_k)
Actual_Time_Ratio(i,j,T_k) = File_size(i,j,T_k) /Actual_Time(i,j,T_k)
diff = Actual_Time_Ratio(i,j,T_k) - Predicted_Time_Ratio(i,j,T_k)
If (diff >0)
Next_Predicted_Time_Ratio = Predicted_Time_Ratio(i,j,T_k) + diff/2
else
Next_Predicted_Time_Ratio = Predicted_Time_Ratio(i,j,T_k) - diff/2
j = j +1
T_k = T_k +1 // next epoch
Predicted_Time_Ratio(i,j,T_k) = Next_Predicted_Time_Ratio
Predicted_val(i,j,T_k) = File_size(i,j,T_k)/ Predicted_Time_Ratio(i,j,T_k)

The minsize, refers to the size that the DataStream should be able to impart some pattern or knowledge. The maxsize, refers to the maximum computing power of the nodes in the distributed environment. The unit buffer stores and organizes these sized streams, in the FIFO order.

Scheduler module - refers to the predictor table, and allocates the data streams to the respective nodes as per the load balancing policy and the weights imparted from the performance table. The Weights associated with every node are maintained in the performance table.

Monitoring module – keeps track of the performance of the nodes under consideration, collects the feedback from the nodes and updates the predictor table. Also, watches the status of the nodes and updates the history table as and when the Fault_index(Ni) of a node equals zero.

Knowledge module – Collects the knowledge unearthed from the datasets mined at the computing nodes and consolidates the results.

Stationary agents named mining agents are deployed at the computing nodes. These agents receive the data stream, perform the mining operation and evaluate the parameters prevailing in the data Streams. The modules in the mining node comprise of the:

Receiver module – receives the data streams allocated to the particular node

Knowledge module – performs the mining operation and submits the knowledge unearthed from the local datasets.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3 : Simulated Values in a Cluster with 8 nodes

ID	Server_ Code	File_ Names	File_ Size	Predicted_Time_ Ratio	Predicted_ Time	Next_Predicted_T me_Ratio	Next_ Predicted_ Time	Actual_Ti me_ Ratio	Actual _ Time
58	8	Split8_Inpu t File.txt	50100	2686.25	19	2386.88	20.99	2087.5	24
59	7	Split1_Inpu t File.txt	50100	2203.27	23	2065.1	24.26	1926.92	26
60	4	Split3_Inpu t File.txt	55000	2366.22	23	2240.8	24.54	2115.38	26
61	5	Split4_Inpu t File.txt	75000	2573.19	29	2458.47	30.51	2343.75	32
62	6	Split5_Inpu t File.txt	101000	2241.35	45	2172.76	46.48	2104.17	48
63	1	Split2_Inpu t File.txt	80000	2191.67	37	2071.45	38.62	1951.22	41
64	2	Split8_Inpu t File.txt	50100	2090.91	24	1578.44	31.74	1065.96	47
65	3	Split6_Inpu t File.txt	80000	1815.91	44	1759.02	45.48	1702.13	47
66	8	Split1_Inpu t File.txt	50100	2386.88	21	2282.57	21.95	2178.26	23
67	7	Split2_Inpu t File.txt	80000	2065.1	39	1984.93	40.3	1904.76	42
68	1	Split1_Inpu t File.txt	50100	2071.45	24	2174.36	23.04	2277.27	22
69	3	Split2_Inpu tFile.txt	55000	1759.02	31	1796.18	30.62	1833.33	30
70	2	Split1_Inpu t File.txt	75000	1578.44	48	1587.09	47.26	1595.74	47
71	4	Split2_Inpu t File.txt	55000	2240.8	25	2178.09	25.25	2115.38	26
72	5	Split3_Inpu t File.txt	55000	2458.47	22	2538.76	21.66	2619.05	21
73	6	Split4_Inpu t File.txt	75000	2172.76	35	2128.05	35.24	2083.33	36
74	8	Split6_Inpu t File.txt	80000	2282.57	35	2317.76	34.52	2352.94	34

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Feedback module – keeps track of the parameters and the performance metrics such as the actual time taken for completing the assigned task and setting the flag, Fault-index (Ni) to zero, whenever the time of execution exceeds the max_threshold_value.

The mobile agent "Resource Characteristic Analyst" (RCA) – while roaming across the network collects and consolidates the knowledge unearthed.

5. EXPERIMENTS AND RESULTS

In order to evaluate the above discussed forecast technique, experiments were conducted on a cluster with eight servers.

5.1 Experimental Evaluation

A Java program to sort and count the number of occurrences of keywords in a text of size ranging from 50KB to 185 KB, in order to construct a word vector, was executed in a cluster constituting eight nodes, and a portion of the forecast table constructed during the process is depicted in table 3.

The tabulated values are a part of the fore cast table constructed at run time, and the values correspond to the time epochs 9, 10 and 11. During these epochs all the 8 nodes, viz., N_1, N_2, N_3, N_4, N_5, N_6, N_7 and N_8 were active in the distributed environment. The measures corresponding to a particular node N_8 are illustrated here. ID 58, 66 and 74 displays the values pertaining to this node. The expected processing power (Next_Predicted_Time_Ratio) of this node N_8 forecasted for the time epoch 9 is 2386.88 bits/millisecond. The size of the dataset to be processed in the epoch 9 is 50100 bits. Therefore the time predicted is 19 milliseconds. The actual time taken for processing this dataset at node N_8, recorded as the feedback is 24 millisecond. Hence, the actual processing power worked out (Actual_Time_Ratio) is 2087.5 bits/millisecond. These two values, Next_Predicted_Time_Ratio (2386.88) and Actual_Time_Ratio (2087.5) of the present epoch 9 form the basis for predicting the processing power of node N_8 in the forthcoming epoch 10. Accordingly the processing power (Next_Predicted_Time_Ratio) forecasted for time epoch 10 is 2386.88 bits/millisecond (ID 66). Here again, the size of the dataset to be processed is 50100 bits. The predicted time is 21 milliseconds. The actual time taken for processing this dataset at node N_8 has been recorded as 23 milliseconds and the actual processing power, Actual_Time_Ratio, of this epoch 10 is 2178.26 bits/millisecond. Now the forecast for epoch 11 is made with the measures available measures at ID 66 and the forecasted processing power of node N_8 at the time epoch is 2282.57 bits/millisecond. At this epoch, the size of the dataset to be processed is 80000 bits. Therefore, the predicted time for processing the same at node N_8 is 35 milliseconds. It is inferred from the table that the prediction table is able to forecast the processing time of the proceeding task when executed in a node, more accurately. In the same way, the predictions are made for all the contributing nodes.

In order to evaluate the precision achieved by the algorithm, a comparison of the Predicted Time and Actual Time has been made, on observing the T_Predictor values when identical datasets of size 150100 bits are processed in node N_8. A bar chart depicting the Predicted Time and Actual Time is given in Figure 3. In the time epoch 8, the difference between the Predicted Time and Actual Time is noted as 32.65%. The same has been slashed down to 27.77% in the epoch 12 , and steps down to 8.79% in the epoch 16 and 3.44% in epoch 20; it is evident that the difference between the Predicted time and the actual time is slimming out as time progresses. That is the predictor is able to reflect the ubiquitous nature of the environment, since the same is constructed based on the feedback obtained from the contributing nodes.

The core objective of this research is to capture the variations in the distributed environment, and it is clear that the predictions made converge towards the actual measures. Load balancing and scheduling algorithms can be designed based on the predictor, ensuring fast processing and effective utilization of hardware components in the distributed environment.

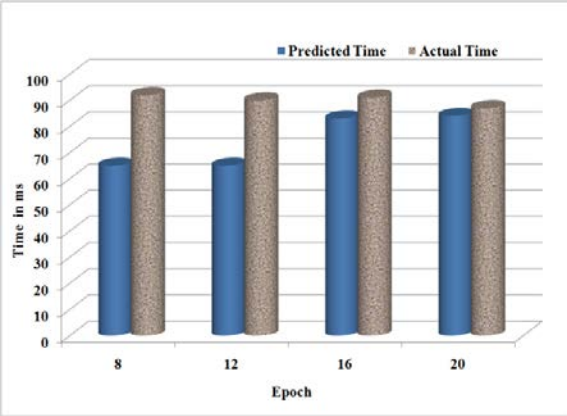

Figure 3: Comparison of the actual time and predicted time

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4: Improved Performance of the Prediction Based Algorithm

Cluster Size	Bytes processed per Millisecond		
	LLF	PRE_LLF	Improved Performance (%)
5	20047.16	29645.85	147.88
10	29342.79	43731.12	149.04
15	32325.83	48900.00	151.27
20	41103.87	63106.48	153.53
25	47299.06	74567.00	157.65

Experiments are conducted to evaluate the performance of the prediction based strategy when incorporated in load balancing and scheduling algorithms, and frame a comparison with non predictive algorithms. The study has been carried out in a distributed environment of various cluster sizes 5, 10, 15, 20 and 25. The application for classifying the datasets has been specially deployed in the nodes.

The observations as recorded are analyzed in two steps.

First, the system throughput is achieved, in terms of the bytes processed per second when a particular dataset is split up and scheduled to the nodes as per the Predictive Least Load First (Pre_LLF) algorithm. Second, the processing time of various data sets are observed, keeping the cluster size constant. About 400000 records are split and scheduled to the nodes duly incorporating the prediction based policies. Observations are recorded when scheduled to the least loaded node, reinforcing the predictive processing power of a node.

5.2 Results and Discussion

Throughput and scalability are the two major objectives in a distributed environment. Improved throughput in a distributed system is the main focus of prediction based load balancing strategies.

The throughput in terms of bytes processed per millisecond of the distributed environment is observed for varying sizes of clusters. The study has been extended to observe the behavioral pattern of the distributed system on processing datasets of varying sizes in a least loaded node, with prediction and without prediction.

Table 4 depicts the throughput observed when a file size of about 400000 records are scheduled to the nodes and the classification operation is performed

The measures in the column LLF are the processing powers of the contributing nodes when a dataset is scheduled to a lightly loaded node without prediction and the values in the column PRE_LLF are the processing powers when dataset is scheduled to a least loaded node, duly incorporating the predicted processing power of the particular node. It is evident, that the PRE-LLF shows an improved performance of 147.88% when scheduled to 5 nodes and this improvement shows an increasing trend, by recording 149.04 % for a cluster size of 10 and keeps on incresing with the cluster sizes. For a cluster size of 25 the maximun improvement of 157.65% is observed.

The feed back collected from the nodes reflects the heterogeneity of the ubiquitous distributed environment , and this forms the basis for forecasting the processing power of a particular node in the proceeding epoch. The PRE_LLF being formulated on this, could impart better performance since the variations in the distributed system are taken care of. This prediction would be more valuable for quite large numbers of nodes. Figure 4 depicts the comparison of the performance, and it is inferred that prediction based strategies prove to be more efficient in clusters of higher sizes.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

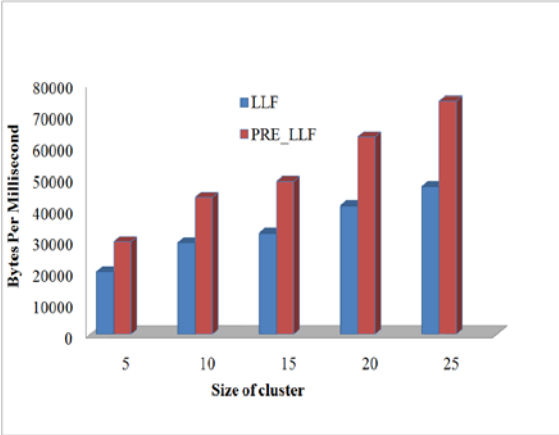

Figure 4: Comparative Performance of Prediction Based and Non Predictive Algorithms

The study has been extended to analyze the performance of prediction based strategies on processing varying file sizes on a cluster size of 20. The actual time taken for processing datasets of sizes varying from 14 MB, to 125 MB has been recorded for the LLF and PRE_LLF. Figure 5 depicts the comparison of the speed, in processing a task adopting Prediction based strategies. It is clear from the graph that prediction based strategies impart fast processing, as the size of the datasets increases, and it is inferred that the prediction based strategies prove to be more efficient in processing data sets of higher size of datasets also.

6. CONCLUSION

The construction of a predictor upon which load balancing techniques are constructed in a distributed system comprising heterogeneous computing elements is discussed in this paper. The feedback based strategies discussed, predict the processing power of a node in a distributed system, with reference to the actual execution time. A collaborative agent system, comprising stationary and mobile agents, captures the feedback from the contributing nodes, and updates the predictor table dynamically. And this dynamic updating enables the system in predicting the processing power of the computing nodes, closely adhering to the fluctuations of the ubiquitous distributed system.

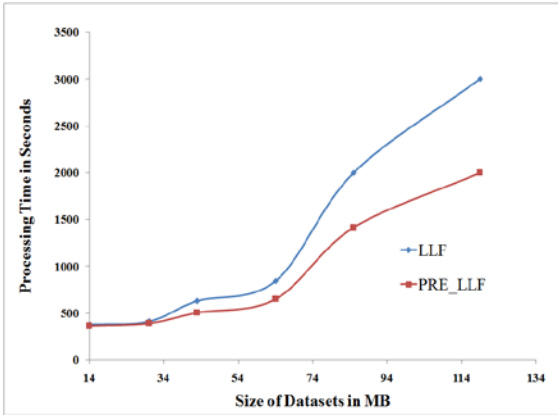

Figure 5: Comparative Processing Time

The predictor constructed during run time approaches the actual processing time, and this aspect forms the basis for designing load balancing algorithms reflecting the heterogeneity of the distributed environment.

REFERENCES

[1] Adler.M, Eran Halperin, R. Karp.M, and Vazirani.V, A, Stochastic process on the hypercube with applications to peerto-peer networks, in Proc. STOC, 2003.

[2] Bakri Yahaya, Rohaya Latip, Mohamed Othman, and Azizol Abdullah, Dynamic Load Balancing Policy with Communication and Computation Elements in Grid Computing with Multi-Agent System Integration, International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(3): 780-788 The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085).

[3] Bravier.A.D, Montz.A.B, Peterson.L.L, Predicting Mpeg Execution Times, Proceedings of international conference – Measurment and modeling of computer systems, SIGMETRICS, 1998.

[4] Byung Ha Son, Seong Woo Lee, Hee Yong Youn, Prediction-Based Dynamic Load Balancing Using Agent Migrationfor Multi-Agent System,2010 12th IEEE International Conference on High Performance Computing and Communications.

[5] G.Cybengo, Dynamic Load Balancing for Distributed Memory Multiprocessors, Journal of Parallel and Distributed Computing,Vol 7/1989

[6] David Karger and Matthias Ruhl, New Algorithms for LoadBalancing in Peer-to-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Peer Systems,,Tech. Rep. MIT-LCS-TR-911, MIT LCS, July 2003.

[7] R. Ezumalai, G. Aghila and R. Rajalakshmi, Design and Architecture for Efficient Load Balancing with Security Using Mobile Agents, IACSIT International Journal of Engineering and Technology Vol. 2, No.1, February, 2010 ISSN: 1793-8236.

[8] Frank Dabek, Frans Kaashoek, David Karger, Robert Morris, Simple Load Balancing for Distributed Hash Tables, in Proc.IPTPS, Feb. 2003.

[9] IBM WebSphere Edge Server Redbook. http://www.redbooks.ibm.com /pubs/ pdfs/redbooks /sg246511.pdf.

[10] Josenildo Costa da Silvaa, Matthias Kluscha, Stefano Lodi, and Gianluca Moro, Privacy-preserving agent-based distributed data clustering, Web Intelligence and Agent Systems: An international journal 3 (2006) 1–18.

[11] Katz E, Butler M, and McGrath R, "A Scalable Http Server: TheNcsa Prototype," Computer Networks and ISDN Systems, vol. 27,pp. 155-164, 1994.

[12] Kim Y H , "An Efficient Dynamic Load Balancing Scheme for Multi-agent System Reflecting Agent work load,cse, vol. 1, pp.216- 223, 2009 International Conference on Computational Science and Engineering, 2009.

[13] Liang-Teh Lee, Hung-Yuan Chang , Gei-Ming Chang and Hsing-Lu Chen, A Grey Prediction Based Load Balancing Mechanism for Distributed Computing Systems, ISCIT 2006, IEEE, 0-7803-9740-X/06.

[14] Neeraj Nehra, 2,R.B. Patel, 3V.K. Bhat, A Framework for Distributed Dynamic Load Balancing in Heterogeneous Cluster, Journal of Computer Science 3 (1): 14-24, 2007 ISSN 1549-3636.

[15] Server Iron Chassis L4-7 Software Configuration Guide. <http://www.foundrynet.com/services/documentati on -/sichassis/management.html>

[16] Shirazi B A , Hurson A R , and Kavi K M, Scheduling and LoadBalancing in Parallel and Distributed Systems. CS Press, 1995.

[17] Welling G, Ott M, and Mathur M, "A Cluster-Based Active Router Architecture," IEEE Micro, vol. 21, no. 1, Jan./Feb. 2001.

[18] Willeebeek M H -LeMair and Receves A P ,Strategies for load balancing on highly parallel computers, IEEE Transactions, Parallel and distributed systems, Vol IV.No9, 1993.

[19] Xu, C.-Z. and Wims B, 2000. Mobile agent based push methodology for global parallel computing. Concurrency and Computation: Practice and Experience, 14: 705-726.

[20] Yongjian Yang, Xiaodong Cao, Jiubin Ju, Yajiun Chen, Research on Prediction Methods for Load Balancing Based on Self-adaptive and Confirmation Mechanism, 2005 International Conference on Control and Automation (ICCA2005),June 27-29, 2005, Budapest, Hungary.