Research on Novel Algorithm for Host Load Prediction

Yongjian Yang, Xiaodong Cao, Yanjun Chen and Shuqiu Li

College of Computer Science and Technology, Jilin University
Changchun, Jilin Province, China, 130012

yyj@jlu.edu.cn

Abstract

By analyzing a large of history data and predicting load variance trend of host, We can offer the important basis to load balance and decide load balance strategy for effective load balance. However most load prediction re-search focus on flow prediction of the network or load prediction algorithm it-self used classical time series prediction method, usually it is ignored on load balance after load prediction. Moreover, information exchange among nodes in distributed system through broadcast usually. It would be casting a lot of re-source of network. it is difficult to balancing load with whole system, because some node don't know load status and change of other network nodes well and exactly. Aiming at these disadvantages, we propose a novel algorithm for up-dating load information and prediction based on mobile agent called CPLBMA . This system is based on OpenMosix parallel distributed system platform, and used series prediction method combining with mobile agent technology to im-prove the performance from a simple load prediction algorithm. According to the prediction result, using stochastic interval and hierachical cluster analysis, selecting a matched node for overload node or underload node to dispatching and migrating task, it can overcome shortage of actual prediction technology, reach load balance and the performance of CPLBMA is improved.

Keyword: distributed system, load balancing, prediction, mobile agent, hierachical cluster analysis

I. Introduction

In the parallel distributed system with multi-processing element (PE), when it is non-uniform for task running on PE, there is load unbalance in every PEs. For effectively and reasonably utilizing resource of each PE, we introduce load balance mechanism to parallel distributed system. According to the controlling manner, load balancing can be classified into Centralized Controlling and Distributed Controlling. However, there are some problems in both manners. Centralized Controlling has bottleneck and low reliability. Distributed Controlling needs a number of messages to update load information and balance load among nodes[1].

Load information shows the status of nodes, including the utilization of CPU, Memory and so on. How this information is accurate and timely can affect the result of load balancing. Most traditional load balancing systems collect and update load information by broadcasting or polling. The former method will bring forth much added working for network and nodes. The latter one will decrease the efficiency of load information collecting. The both ways are realized by message-passing, which is

unreliable. Most of them are based on client/server model, which has fixed roles and single manner, so their flexibility is low[2].

Because task run time of a host is close relative to load of this host, we can predict task run time of the host by predicting load of the host. So load prediction of the host is an import substance on load balance system[3]. We can analyze a large collection of history data, and predict load variance trend of host. The result of prediction can offer the import basis to load balance, thus it decide load balance strategy for effective load balance.

At present, on load prediction research of cluster host, there are some problems in: main prediction method on load prediction is flow prediction of the network; load prediction system is not strongly adaptability; only a load index is used; the contradiction between prediction precision and system cost of load prediction is not effectively resolved; the most load prediction research focus on load prediction algorithm itself used classical time series prediction method, but it is ignored on load balance after load prediction.

From the discussion above, we can proposes a load balancing framework based on mobile agent named CPLBMA(Collecting and Predicting of Load Balancing Based on Mobile Agent). Mobile agent (MA) is a novel technology originated from distributed network and artificial intelligence [4][5]. We can use classical time series prediction method combining with mobile agent technology to improve the performance from a simple load prediction algorithm. Based on prediction result, using stochastic interval and hierachical cluster analysis, select a matched node for overload node or underload node to dispatching and migrating task, and it can reach load balance.

II. The System structure

To avoid the bottleneck of Centralized Controlling, Distributed Controlling is adopted by CPLBMA, and then every node receives tasks independently. To improve efficiency, adaptability and extensibility of system, a Control Node is used. It can collect and update the load information, and monitor the running state of system. It would adjust the strategy and structure of system if necessary. These operations are all completed through mobile agent. In this way, extra cost to balance load is low, and the performance and adaptability can be improved greatly.

But, this way may result in a new bottleneck in Control Node. Fortunately it can be overcome by mobile agent[6]. Mobile agent can distribute computation into all nodes, and a mobile agent can accomplish the work in one time that needs many interactions among nodes in traditional ways. This problem will be still discussed in the rest of this paper. Figure 1 illustrates the structure of CPLBMA.

In this structure, nodes can be classified into Serving Node fixed serving module to serve for tasks, and Controlling Node fixed controlling module to control the sys-tem. Controlling Node is the best node in the system. It can also receive and execute tasks. But to improve its reliability, fewer tasks should be allocated to it.

Because the controlling to system is completed by mobile agent in CPLBMA, the Controlling Node wouldn’t be the bottleneck. Moreover, CPLBMA can easily settle the trouble of Controlling Node as follows: Controlling Node chooses a standby node and fixes the controlling module beforehand. If the trouble appears in current Con-trolling Node, the standby node will act as a new Controlling Node, and then control the system to keep on working normally.

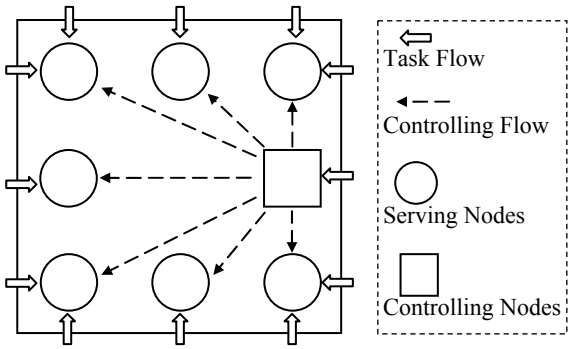

Figure 1. The structure of CPLBMA.

III. Collection and Prediction of Load Information

To achieve load balancing, load information of each node should be collected first of all. In traditional methods, each node sends its load information to a load balancer in Centralized Controlling, and then the balancer sends load information of all nodes to each node. In Distributed Controlling, every node broadcasts load information of itself to all the other nodes. However, both methods will consume plenty of band-width, and the former method will result in a bottleneck[7].

CPLBMA presents the following methods to collect and predict load information. Load Information Collector in every node collects current load information of local node periodically. The Mobile Agent for collecting load information dispatched by Control Node collects load information of every node in a cycle. At the same time, it releases the load information of other nodes stored in this Mobile Agent to this node, as figure 2. This information released to current node is up to date because it is collected by the Mobile Agent just now.

At present, on load prediction research of cluster host aim at the flow prediction of the network or load prediction algorithm itself, but it is ignored on load balance after load prediction.

By analyzing defect of load prediction research, we build up a Load Balancing Prediction System based on Mobile Agent----CPLBMA. These systems is based on OpenMosix parallel distributed system platform, and used classical time series prediction method combining with mobile agent technology, and improve from a simple load prediction algorithm. So it can overcome shortage of actual prediction technology, offer reliable information base to system balancing.

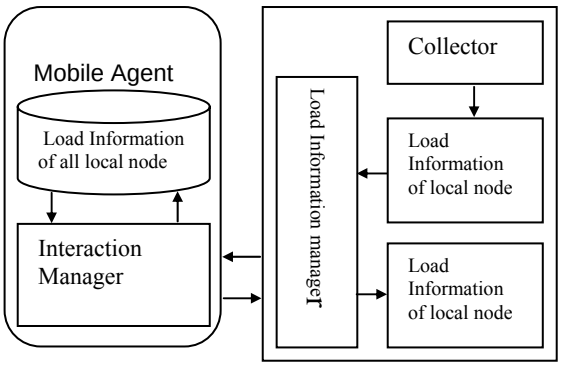

Figure 2. Load information collection on nodes.

IV. Algorithm for collecting load information

A. *Stochastic interval*

Often, stochastic values for a system characteristic are determined by examining a time series of previous values for that characteristic, for example, as made available by an online tool such as the Network Weather Service[8]. Given data in the form of a time-series, the simplest way to represent the variability of a stochastic value is as an interval.
We define the interval of a stochastic value X to be the tuple

$$X = [\underline{x}, \bar{x}] \ \{x \in X \mid \underline{x} \le x \le \bar{x}\} \quad (1)$$

The values $\underline{x}$ and $\bar{x}$ are called the endpoints of the interval. The value $\underline{x}$ is the minimum value over all $x \in X$ and is called the lower bound, and the value $\bar{x}$ is the maximum value, call the upper bound.

B. *Prediction Window*

we can get a time series of load information, L1，L2，…Ln. The values in the time series can be used to make predictions. The period over which the time series is taken (T) may influence the set of values used to make a prediction. And then we introduce a coefficient Prediction Window (W). It include a time series L1，L2，…Ln and length of it is N = n. W may influence the set of values used to make a prediction too.

C. *description of load information using interval*

From the above definition, we can have a description of load information using inter-val. As follows:

Assume that load of system is L, so L is a stochastic interval and is up to definition of interval. We can make a definition for L as that:

$$L = [\underline{l}, \bar{l}] \ \{l \in L \mid \underline{l} \le l \le \bar{l}\} \quad (2)$$

$$\underline{l} = \mathrm{Min}\{L_1, L_2, \cdots, L_n\}$$

$$\bar{l} = \mathrm{Max}\{L_1, L_2, \cdots, L_n\}$$

There L1, … , Ln is a load state series of the node with a period of time. This period of time is a adjustable parameter.

We can make a safe conclusion that load state of every node in distributed system express as a two-tuples, but not a simple value. So there is a better description to load of every node.

D. *Collection of load information using mobile agents*

There is a Node Load State Table on every node in distributed system, and the struc-ture of this table is as table 1:

Table 1. Node Load State Table

Node num	Lmax	Lmin
i	LXi	LNi

There is a Node Load State Series Window: W = {L1,L2,…,Ln}, and this window will be updated at the end time of each collecting information periods.

At initialization, mobile agent will go out from a node and move between every node by engaged route policy. And it begins to collect the load information of the node. There is a Node Load State Table on mobile agent. When mobile agent move to a new node, load state two-tuples on new node will be read by mobile agent. Node Load State Table on mobile agent will be updated and than Node Load State Table and Node Load State Series Window on the node will be updated by mobile agent. So after a period of mobile agent moved, the task of collecting load information of node finish.

V. Host Load Prediction Algorithms

A. *Description of prediction algorithms*

Linear time series models such as those in the Box-Jenkins AR, MA, ARMA, and ARIMA classes might be appropriate for predicting load by analysis statistical properties of load[9][10].

(1) AR(p) models

The class of AR(p) models (purely autoregressive models) have $Zt = {}^{1}\!/_{\phi(B)} a_t + \mu$, where Φ(B) has p coefficients. Intuitively, the output value is a Φ- weighted sum of the p previous output values. The t+1 prediction for a load sequence is the Φ-weighted sum of the p previous load measurements. The t+2 prediction is the Φ-weighted sum of the t+1 prediction and the p-1 previous load measurements, and so on.

From the point of view of a system designer, AR(p) models are highly desirable since they can be fit to data in a deterministic amount of time. In the Yule-Walker technique that we used, The autocorrelation function is computed to a maximum lag of p and then a p-wide Toeplitz system of linear equations is solved. Even for relatively large values of p, this can be done almost instantaneously.

(2) MA(q) models

The class of MA(q) models (purely moving average models) have where θ(B) has q coefficients. Intuitively, the output value is the Φ-weighted sum of the current and the q previous input values. The t+1 prediction of a load sequence is the Φ-weighted sum of the q previous t+1 prediction errors. The t+2 prediction is the Φ-weighted sum of the predicted t+1 prediction error (zero) and the q-1 previous t +1 prediction errors, and so on.

MA(q) models are a much more difficult proposition for a system designer since fitting them takes a nondeterministic amount of time. Instead of a linear system, fit-ting a MA(q) model presents us with a quadratic system. Our implementation, which is nonparametric (ie, it assumes no specific distribution for the white noise source), uses the Powell procedure to

minimize the sum of squares of the t+1 prediction errors. The number of iterations necessary to converge is nondeterministic and data dependent.

(3) ARMA(p,q) models

The class of ARMA(p,q) models (autoregressive moving average models) have where Φ-(B) has p coefficients and θ-(B) has q coefficients. Intuitively, the output value is the Φ-weighted sum of the p previous output values plus the θ-weighted sum of the current and q previous input values. The t+1 prediction for a load sequence is the Φ-weighted sum of the p previous load measurements plus the θ-weighted sum of the q previous t+1 prediction errors. The t+2 prediction is the Φ-weighted sum of the t+1 prediction and the p-1 previous measurements plus the θ-weighted sum of the predicted t+1 prediction error (zero) and the q-1 previous prediction errors, and so on.

By combining the AR(p) and MA(q) models, ARMA(p,q) models hope to achieve greater parsimony — using fewer coefficients to explain the same sequence. From a system designer's point of view, this may be important, at least in so far as it may be possible to fit a more parsimonious model more quickly. Like MA(q) models, how-ever, ARMA(p,q) models take a nondeterministic amount of time to fit to data, and we use the same Powell minimization procedure to fit them.

(4) ARIMA(p,d,q) models

The class of ARIMA(p,d,q) models (autoregressive integrated moving average models) implement Equation 0.1 for d = 1; 2;… Intuitively, the (1-B)d component amounts to a d-fold integration of the output of an ARMA(p,q) model. Although this makes the filter inherently unstable, it allows for modelling nonstationary sequences. Such sequences can vary over an infinite range and have no natural mean. Although load clearly cannot vary infinitely, it doesn't have a natural mean either.

ARIMA(p,d,q) models are fit by differencing the sequence d times and fitting an ARMA(p,q) model as above to the result.

B. Select migrated-task node matching algorithms

Because we transform the load state value of node into the two-tuples, migrated-task node matching algorithms can be looked as the tow-tuples matching algorithms. this problem is hierachical cluster analysis at the math and can be conclude with degree of differ. the distance is the best method of calculated degree of differ. Generally, we use to calculate by Minkowski distance.

$$d(i,j) = \sqrt[q]{(|X_{i1} - X_{j1}|^q + |X_{i2} - X_{j2}|^q + \cdots + |X_{ip} - X_{jp}|^q)} \quad (3)$$

there i =(Xi1,Xi2,…,Xip) and j = (Xj1,Xj2,…,Xjp)are two p dimensionality Ob-jects, q is a positive integer. It can make a better effect at compute distance with q = 2, and computational complexity is low.
So we can make a better matching algorithm:

1. Suppose that load state of every node is a two-tuples (L1i,L1j), (L2i,L2j), …,(Lni,Lnj).

2. Calculate load of a suppositional middle node M.

$$M = (\frac{1}{n}\sum_{K=1}^{n} L_{ki}, \frac{1}{n}\sum_{k=1}^{n} L_{kj}) \quad (4)$$

3. Currently node will calculate Minkowski distance d with middle node M by load state two-tuples of itself.

4. Calculate Minkowski distance between other node and middle node respectively, and we have a distance series d1, d2, …, dn-1.

5. And than calculate |d-d1|, |d-d2|, … |d-dn-1|, we can obtain a difference value series S1, S2, …, Sn-1.

6. The best matching node P can be calculated:

$$p \in \{1,2,\cdots,n-1\} \cup \mathrm{S_p} = Min\{S_1, S_2, \cdots, S_{n-1}\} \quad (5)$$

7. A migrated-task mobile agent will be started and migrate task to P node.

VI. Simulated Experiment

This section provides the results of our simulation experiment. We present some comparison between the method of Collecting and Predicting of Load Balancing Based on Mobile Agent(CPLBMA) and the Traditional Probabilistic Dissemination Algorithms (TPDA).

To evaluate the performance of our proposed algorithm, we have used following performance metrics: degree of balance of all nodes, the execution time of all tasks, and number of messages. These three parameters can reflect the performance of algorithms. Our goals are to make the degree of balance of all nodes of CPLBMA higher than that of TPDA, to make execution time of all tasks in whole system of CPLBMA shorter than that of TPDA, and to reduce the messages among nodes.

Before our simulation experiment, we have tested some correlative parameters, including network delay, execution time of task and so on. The experiment is implemented in Linux installed OpenMosix and Aglets. Then we apply these parameters and some experiential values into the simulation.

In our simulation, we compare that the average load rate of CPLBMA and TPDA. There are 100 nodes and 2000 tasks which are added to these nodes randomly. But the distribution of tasks on nodes is same in different algorithms. From Figure 3, we can conclude that the degree of balance of CPLBMA is higher about 2% than TPDA.

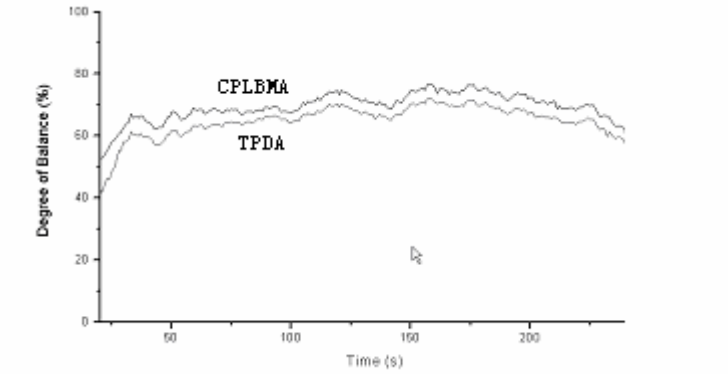

Figure 3. Comparison for Degree of Balance Between CPLBMA and TPDA

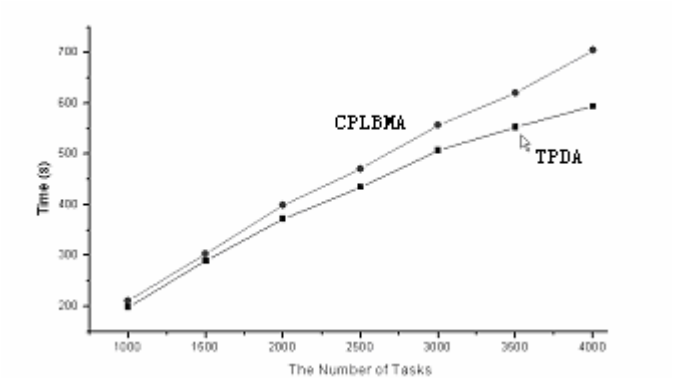

Figure 4. Comparison for execution time of all tasks between CPLBMA and TPDA.

From figure 4, we can see that the total execution time of all tasks in CPLBMA is shorter than that in TPDA. It is more evident when the number of tasks is larger.

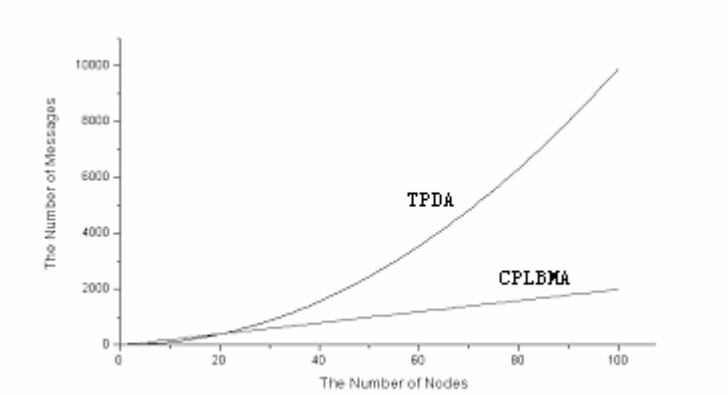

Figure 5. Comparison for the number of messages between CPLBMA and TPDA.

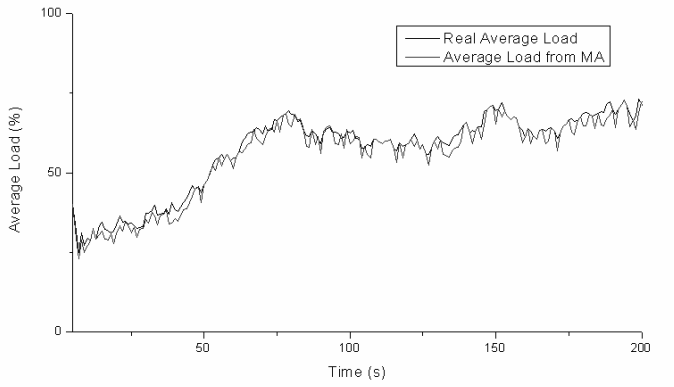

Figure 6. The degree of fitting on average load of CPLBMA

From figure 5, we can conclude that the more of nodes, the more of messages to update load information. But the increase in CPLBMA is slower than TPDA, because of the use of mobile agent. As same as mentioned before, when there are adequate nodes in a system, messages flood will degrade the performance of whole system.

In figure 6, we can find that the average load calculated by mobile agent is very close to the real one. So we can prove CPLBMA is feasible.

VII. Conclusions and Future Works

In this paper, we proposed a framework for load balancing using mobile agent named CPLBMA. CPLBMA can resolve some problems existed in traditional load balancing system, such as bottleneck, updating load information and prediction strategies.

According to analysis to CPLBMA and its simulation experiment results, we can conclude that the performance of CPLBMA is better than TPDA. CPLBMA can avoid messages flood, shorten the whole execution time of tasks, and improve the performance of load balancing.

References

[1] George Coulouris, Jean Dollimore, Tim Kindberg. Distributed Systems: Concepts and Design. 3rd Edition, Addison-Wesley, 2001.
[2] Jiannong Cao, Yudong Sun, et al, Scalable load balancing on distributed web servers using mobile agents. Journal of Parallel and Distributed Computing, 63(10), 2003, 996-1005.
[3] R Wolski. Dynamically forecasting network performance using the network weather service [J].Cluster Computing, 1998
[4] V.A.Pham, A.Karmouch, Mobile Software Agents: An Overview. IEEE Communications Magazine, 36(7), 1998, 26-37.
[5] D. Lange and M. Oshima, Programming and Deploying Java Mobile Agents with Aglets (Addison-Wesley, 1998).
[6] J. Gomoluch and M. Schroeder, Information agents on the move: A survey on load balancing with mobile agents. Software Focus, 2(2), 2001
[7] Luís Moura Silva, Guilherme Soares, Paulo Martins, Victor Batista, &Luís Santos, The Performance of Mobile Agent Platforms. First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents. Palm Springs, California, 1999, 270-271.
[8] A. Barak and O. La'adan. The MOSIX Multicomputer Operating System for High Performance Cluster Computing. Journal of Future Generation Computer Systems, 13(4-5):361-372, March 1998. 3.4.1 138.
[9] Box-Jenkins. Forecasting and Time-series Analysis. 1994, Third Edition.
[10] Martin Quinson. Dynamic Performance Forecasting for Network-Enabled Servers in a Metacomputing Environment[C] in:PMEO-PDS 02, 2002

Yongjian Yang，a Full Professor of Computer College, the vice Dean of Software College in Jilin University. He received the master degree of Computer Application from Beijing University of Posts and Telecommunications and the Doctor degree of Computer Software from Jilin University. He has been studying Computer Network Communications.

Xiaodong Cao is a graduate student, majoring in Distributed Network under the instruction of Yongjian Yang professor at present. His research interests are in Load Balancing, Load Prediction, Mobile Agent and Active Network. He has attended 3 important projects and

Yajun CHEN is a graduate student, majoring in Distributed Network under the instruction of Yongjian Yang professor at present. His research interests are in Load Balancing, Load Prediction, Mobile Agent and Active Network. He has attended 3 important projects and published 2 journal papers.