DLBEM: A Predictive Approach to Dynamic Load Balancing

Han Zhao, Xinxin Liu, Xiaolin Li
Scalable Software Systems Laboratory
Department of Computer Science
Oklahoma State University
Stillwater, OK 74048, USA
Email: {haz, xinxin, xiaolin}@cs.okstate.edu

October 8, 2007

Abstract

In this paper we propose a predictive approach for dynamic load balancing. This approach involves predicting unbalanced load distribution using Estimation Maximization (EM) algorithm, and migrating local jobs based on the estimation average. This strategy is an improvement to the existing approaches because by using EM algorithm, we only extract sample workload information which is assumed to follow mixture Gaussian distribution without keeping track of the whole workload information in the system. This method can give relatively accurate results and greatly reduces the communication efforts in distributed systems. After getting the estimation average load, a local job migration policy is invoked to balance the load distribution without oscillation. Simulation results show that our approach can get load distribution information

1 Introduction

Load balancing is one of the most important problems in achieving high throughput and good system performance in parallel computing. Typical load balancing strategies include unbalance load detection and job migration. Both of the two procedures need large number of communications in the whole system. While parallel computing develops very fast in recent years, characterized by heterogenous computing environment, large scale system and high frequency of communications. Such development brings new challenges for traditional dynamic load balancing strategies and calls for more elaborate algorithm design for even utilization of resources without much communication overhead. The problem is essentially
hard for the reason that load balancing algorithms need global information to make decisions. As the system grows larger, a large number of communications flood in the whole system and greatly impair the overall performance. The situation gets worse when algorithms are designed in a distributed fashion, in which each node is "blind" to system’s global information. However, distributed algorithms perform much better in scalability compared with centralized ones, makes it a better choice for dynamic load balancing in large scale parallel systems.

On the other hand, many numerical algorithms, which are built based on statistics models and probability theory, have been developed and proved to be effective on information prediction. One class of statistical methods, called Maximum Likelihood Estimation (MLE) [1] methods, calculate the best fitting model to sample data, and gives out the optimal estimation on current available data. A popular implementation of parameter estimation with maximum likelihood is Expectation-Maximization (EM) [2] algorithm. Its simplicity and widely applicable feature makes it an essential algorithm in Bayesian Network structure learning. When achieving convergence, it can give out good parameter prediction based on a small sample of data. Inspired by this algorithm, we introduce parameter estimation method into dynamic load balancing problem. By using EM algorithm to predict the average workload distribution based on a small number of sample data, we can substantially reduce communication in a parallel computing system and get relatively accurate information with a scalable distributed algorithm. Although MLE algorithm has been applied to many other fields, to our best knowledge, few MLE methods is introduced in scheduling scenario for information prediction.

In this paper, we not only give detailed information on using EM algorithm to detect unbalanced load distribution in a distributed system, but also design an effective method to perform job migration between processors in a parallel system.

The rest of the paper is organized as follows: Section 2 illustrates background knowledge of EM algorithm. In Section 3, we theoretically present our solution of unbalance detection and job migration policy. The result of simulation is shown in Section 4. In Section 5, we will discuss related work in predictive scheduling. Finally in Section 6 we will conclude the paper and discuss possible future works.

2 The Expectation-Maximization Algorithm

2.1 Introduction to EM Algorithm

In real world data modeling, data subject to analysis often presented in incomplete form, either due to the limitation of the sampling process or due to intentionally missing in purpose of simplifying likelihood function to tractable form. EM algorithm [2, 3, 4, 5] is an effective statistic method to maximize the parameters likelihood estimation in models where unobserved latent variables exist. The algorithm performs in an iterative way until it converges, and each iteration can be divided into two steps. The first step called the E step computes the expectation of complete data set likelihood given the model parameters. The expected
likelihood constructed is then used to compute the current maximum likelihood estimate of the model parameter, called the M step. The beauty of EM algorithm lies in its simplicity, and proved to be effective in Gaussian Mixture and Hidden Markov Models (HMM) [6]. A comprehensive discussion of EM algorithm can be found in the book edited by Watanabe and Yamaguchi [7].

2.2 General Procedure of EM Algorithm

Let $\chi_{\text{obs}}$ denote incomplete data consisting of values of observable variables, and let $\chi_{\text{mis}}$ stands for missing data. The complete data form $\chi$ is given by

$$\chi = (\chi_{\text{obs}}, \chi_{\text{mis}})$$

Let $p$ be the joint density function of the complete data set $\chi$ with parameters $\theta$, by applying the Bayes rule, the E step of the algorithm calculate posterior possibility of missing data $\chi_{\text{mis}}$ given observed data $\chi_{\text{obs}}$ and parameter $\theta$

$$p(\chi_{\text{mis}}|\chi_{\text{obs}}, \theta) = \frac{p(\chi|\theta)p(\chi_{\text{mis}}|\theta)}{\int p(\chi_{\text{obs}}|\hat{\chi}_{\text{mis}}, \theta)p(\hat{\chi}_{\text{mis}}|\theta) d\hat{\chi}_{\text{mis}}}$$

The joint distribution $p(\chi|\theta)$ of observed data $\chi_{\text{obs}}$ and $\chi_{\text{mis}}$ can be defined as the likelihood function, and be expressed as follows

$$L(\theta|\chi) = L(\theta|\chi_{\text{obs}}, \chi_{\text{mis}}) = p(\chi|\theta) = p(\chi_{\text{obs}}, \chi_{\text{mis}}|\theta)$$

To ease the complexity of calculation, the likelihood function is often used in logarithm form $\log L(\theta|\chi)$ to generate easier formulas while attains the same maximum likelihood. The expectation of the log-likelihood given observed value $\chi_{\text{obs}}$ is expressed as

$$Q(\theta) = E[\log L(\theta|\chi)] = \int_{-\infty}^{\infty} p(\chi_{\text{mis}}|\chi_{\text{obs}}, \theta^n) \log p(\chi_{\text{obs}}, \chi_{\text{mis}}|\theta) d\chi_{\text{mis}}$$

After the function of $Q$ is found, the M step maximize the expectation at the current guess $\theta^n$

$$\theta^{n+1} = \arg \max_{\theta} Q(\theta)$$

By alternating between E step and M step, the EM algorithm iterates from initial estimation $\theta^0$ and converges to a local maximum of the likelihood function.

3 Dynamic Load Balancing Strategy with Prediction

In this section we will formally present our load balancing strategy design in two steps. First we will utilize the EM algorithm introduced in Section 2 to predict the mean workload in system. Step two will show how jobs are migrated according to predicted value get in the
estimation step. The system model and algorithm goals will be clarified before the formal strategy develops.

3.1 System Model

Before formally propose our dynamic load balancing strategy, we make the following assumptions. The system consists of multiple processors to process programs in parallel. Processors communicate with each other in a point-to-point network. Programs arriving to be processed can be partitioned into a batch of small jobs. Jobs are assigned to processors and can be migrated freely. Instead of implementing load balancer in every processor, we employ a hybrid system for better scheduling performance. All the processors are partitioned into zones. Each zone contains geographically nearby processors, means that processors communicate more easily within zones. In each zone there is a leading processor in response of collecting information, performing prediction, broadcasting and executing job migration. The processors are organized in this way to reduce computational overhead and number of messages. If every processor make balancing decisions with full system state information, it will require $O(n^3)$ messages delivered. In our hybrid organized strategy balancing decisions are made at each time new batch of jobs arrive at system, and each leading processor intentionally select a small subset of processors reporting their workload. In this way the communication overhead is cut down substantially.

Next we need to specify estimation process parameters: Suppose we have complete load information of all processors given by $\chi$. The reported workload information is $\chi_{obs}$ and the latent workload information is $\chi_{mis}$. We partition $\chi$ into $n$ groups. Each group consists of both observed data and missing data, which are all subsets of $\chi_{obs}$ and $\chi_{mis}$. Let $\chi_i$ stands for the $i$-th group, and each group is normally distributed. The assumption is reasonable because each group has its own mean and standard derivation. In this case all groups forms a mixture Gaussian model. Also let $p(x = i)$ denotes the probability of a particular load amount comes from the $i$-th group. The estimated parameter $\theta$ is given by

$$\theta = \{\mu_1, ..., \mu_n, \sigma^2_1, ..., \sigma^2_n, p(x = 1), ..., p(x = n)\}$$

3.2 Algorithm Goal

Our goal is to estimate mean system workload based on reported workload which is only a small portion of the complete load information. Each node can adjust its workload according to the mean value. The model intends to maximize the utilization of computing power while not incurring large communication overhead. Moreover, the algorithm pursues the following performance goals in parallel:

- Accommodating heterogeneity.
- Achieving good scalability in a hierarchical distributed way.
- adaptively load adjustments.
• Achieving stability by avoiding oscillating task migrations.

The merits of our model will be discussed in later part of this section accordingly.

3.3 Applying EM Algorithm in Scheduling

Compared with central control scheduling, distributed load balancing exhibits good scalability property when more nodes join the network. However, the distributed algorithm also introduces larger communication costs and leads to increasing computation complexity. Our method trades off between the two approaches by using a partly distributed organization. The workload information is recorded as CPU utilization normalized by CPU speed. In this way the algorithm accommodates heterogeneous computing environment, which is popular in today’s parallel systems.

3.3.1 Node Selection

The first step in our method is to decide which processor should report their load information to the zone leading processor. The first method we use is a simple probability model called random pick. The zone leading processors decide the target percentage of active communication processors. Then processors within the zone randomly determine whether to report their workload. The method is easy to implement but hard to depict real workload distribution. Next we try to estimate workload with a roughly estimation of real workload distribution model. Suppose the EM algorithm executes with two Gaussian distributions. We randomly pick one zone and collect workload information within the zone to make an initial guess. The result is then broadcasted to all the other zones. Each processor then decide whether to report its load accordingly. The selection process will terminate when the desired proportion has reached. The method will introduce some communication overhead and increase implementation difficulties slightly compared with random pick selection. But the estimated accuracy will be more stable. Detailed analysis of both schemas can be found in Section ??.

3.3.2 Finding Maximum Likelihood Estimation of Mean Value

Suppose the network is divided into n zones. And each zone follows a Gaussian distribution. The first task is to calculate the probability of a particular observed data set $x_{obs} \in \chi_{obs}$ ($x_{obs}$ is a vector) comes from the $i$-th Gaussian.

$$P(x_{obs}|x_{mis} = i, \theta) = N(\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i^2} \exp\left(-\frac{(x_{obs} - \mu_i)^T(x_{obs} - \mu_i)}{2\sigma_i^2}\right)$$  \hspace{1cm} (1)

The expectation step is performed by using parameter value from last maximization step and (1) the estimation of the unobserved load information $y_{mis}$ is calculated by using the Bayes rule as illustrated in (??)
\[ P(x_{\text{mis}} = i | x_{\text{obs}}, \theta^t) = \frac{P(x_{\text{obs}} | x_{\text{mis}} = i, \theta^t) P(x_{\text{mis}} = i | \theta^t)}{\sum_{k=1}^{m} P(x_{\text{obs}} | x_{\text{mis}} = k, \theta^t) P(x_{\text{mis}} = k | \theta^t)} \] (2)

The expected log-likelihood function of the joint event is given by

\[ \mathcal{L}(\theta) = E_{x_{\text{mis}}} \left[ \lg \prod_{j=1}^{m} P(x_{\text{obs}}, x_{\text{mis}} | \theta) | x_{\text{obs}} \right] \]
\[ = \sum_{j=1}^{m} \sum_{i=1}^{n} P(x_{\text{mis}} = i | x_{\text{obs}}, \theta^t) \lg P(x_{\text{mis}} = i, x_{\text{obs}} | \theta) \] (3)

By expending the probability density function, and let the differential value of 3 equals to zero. We find the current maximum parameter for next iteration.

\[ \frac{\partial \mathcal{L}(\theta)}{\partial \mu_i} = 0 \] (4)
\[ \Rightarrow \mu_i = \frac{\sum_{j=1}^{m} P(x_{\text{mis}} = i | x_{\text{obs}}, \theta^t) x_{\text{obs}}}{\sum_{j=1}^{m} P(x_{\text{mis}} = i | x_{\text{obs}}, \theta^t)} \] (5)

The final result of mean load value is given by

\[ \bar{\mu} = \sum_{i=1}^{n} p(x = i) \mu_i \] (6)

3.3.3 Job Migration Policy

In previous we have illustrated how to detect unbalance between processors. Next jobs should be redistributed evenly within system. It is difficult for designing such policies for the following reasons:

- Bad policies will cause jobs migrating a lot. Each wave of migration will cause new unbalance, incurring a lot of communication overhead.
- Jobs may migrate between nodes continuously, causing oscillation phenomenon.

To overcome these difficulties, we perform job migration only in local zone extent. In each zone one lightly loaded processor will “marry” one heavily loaded processor. The extra workload from heavily loaded processor will migrate to lightly loaded processor. If one single processor can’t construct a pair with another one, it will wait until the first round of migration finished. The migration will terminated when all the nodes perform migration once. Our method involves less communication while still achieving good performance. Detailed analysis of our policy based on simulation results can be found in Section ??.
3.4 An illustrative Example

4 Simulation

In this section we present four different simulations to evaluate the performance of our strategy. To achieve this goal, a load generator program is designed and implemented in C as a testbed for analysis. The load generator program generates programs large enough on a 10\times10-two-dimensional grid. Each processor maintains its load in a normalized manner, that is, the load amount divided by the processor’s computational power. In order to perform load balancing between processors, we need to find out the load mean value with less communication overhead incurred. We also assume a multiprogramming environment, which means several small jobs run simultaneously on a single processor. The job migration is performed after a batch of jobs arrived and EM prediction is performed. The EM predictor and job migration policy are also implemented in C. In all simulations, the prior distribution being measured is assumed to be mixture Gaussian distributions.

The EM predictor aims at reducing communication overhead as many as possible while predicting mean load at an acceptable value. The first two simulations explore how the estimation accuracy can be affected by number of processors involved in the estimation process.

![Figure 1: Estimated Accuracy affected by Number of Nodes](image1)

![Figure 2: Estimated Accuracy with Communication Overhead Reduction](image2)

In both simulations, the estimation accuracy is given by percentage of estimated quotient (estimated mean load dividing actual mean load), and used as the Y-axis of both figures. In figure 1, X-axis is number of processors used. While in figure 2, X-axis is corresponding communication overhead reduction according to number of processors involved in estimation process. Result shows EM algorithm can successfully reduce communication between nodes for information change while still maintains relative good results (90 percent above accuracy).
Another interesting observation is that when relatively less nodes are selected for prediction, the accuracy result exhibits more randomness. The result is fluctuating between better and worse result. The reason can be explained as follows: a small number of data can’t model distribution model accurately enough. Result shows more nodes can be selected to overcome this problem. In figure 2, 20 processors or more selected by using our node selection schema will make stable estimated accuracy increasing. figure 3 further illustrate the importance of node selection:

Figure 3: Node Selection: Random Pick VS. Proportionally Pick

In figure 3, Random Pick Selection exhibits more randomness in estimation. When processor number is small, in other words, communication overhead is small, random pick result is neither accurate nor stable compared with our method. Our method will guarantee the nodes selected roughly model the real load data distribution.

Finally, we will show the effectiveness of our job migration policy. The result is simulated as follows, at each time interval we measure the load information from each processor. The EM predictor is employed with 30 nodes selection to estimate the mean load. The 100 nodes are divided into 10 zones. We implement our job migration policy mentioned in Section 3.3.3. Standard derivation of the whole system is a good criteria to depict the effect of load balancing. The result is shown in figure 4. The observation of simulation result shows that our policy greatly decrease standard derivation at each time interval, causing load to be distributed more evenly among processors.

5 Related Work

This paper solves two critical problems in distributed dynamic load balancing: how to make scheduling decisions based on load estimation and how this technique is used to reduce the
communication overhead in a heterogeneous environment. The predictive scheduling is not a new concept, and it has been widely explored for more than a decade. An early paper [8] proposed a distributed drafting algorithm which used the load estimation messages to reduce the overhead caused by communication. However, distributed system technique was primitive at that time which largely confines the effectiveness of the algorithm. Later researches [9, 10] monitored the distributed system during runtime for estimation or for statistically partition load asymmetrically. But this kind of monitoring could cause serious overhead in performance. In recent years many innovative methods were introduced trying to make decisions based on workload estimation in a distributed fashion, including AI techniques [11], which predicts by using the PACE evaluation engine, P2P techniques [12], and bit vector approach specifically designed for distributed content based publisher/subscribe systems [13]. The problem of reducing communication overhead while maximizing processor utilization also gains widely research. The particles approach [14] places intensively communicating tasks close to each other to minimize communication delays. The MOSIX system [15] used a simple probabilistic model, which selects only a subset of hosts randomly to reduce communication overhead. A more recent paper published in 1995 [16] assumes a distributed asynchronous environment where communications between processes are only done by message passing. Another approach [17] makes the nodes communicate less, the accuracy of balancing decision with less updating is achieved by introducing a linear regression $L_2E$ predictive filtering model.

The powerful statistical tools based on Bayesian inference technique and parameter learning attracts researchers to model them with scheduling. The inference algorithm is employed to aid the wireless sensor network data collection process [18], marketing trends analysis [19] and grid resource scheduling [20]. A paper published in 2004 [21] proposes Bayesian decision networks as the paradigm to handle the scheduler’s uncertainty about system status. The simulation result shows the effectiveness of such models.

6 Conclusion

In this paper, a predictive approach of dynamic load balancing in distributed systems is described. EM algorithm is integrated in our strategy to get a relatively accurate estimate average of workload distribution. As we only need sample workload information to make prediction, our strategy substantially reduce communication overhead in distributed system. Based on the estimated workload average, we proposed a local job migration policy to achieve more evenly workload distribution among participant processors in the system. Our contributions are two folders. First, we make good unbalance detection and migration decisions with less information exchange. The sample data are collected by selecting data tally with real workload distribution. EM algorithm is then performed on these data to get the estimation of mean workload. After that, we consider communication overhead and oscillation in job migration. According to our policy, jobs migrate locally using pair-wise workload exchange. Second, our strategy unveils the potential of applying various developed statistical modeling methods in scheduling problem in distributed systems. The simulation results
validates our belief that the predictive approach based on statistical models and probability theory achieves good results in distributing workload evenly among processors. In addition, our simulation results show that when sample data is small, the estimation accuracy can be more randomness, and estimation accuracy can be affected by node selection strategy.

In our future work, we intend to refine the prediction procedure by employing different probability methods. More theoretical analysis on combining such methods and scheduling need to be conducted for investigation. Also we are aiming at implement more practical works for large parallel system application.

References


