Dynamic Load Balancing of Iterative Data Parallel Problems on a Workstation Clustering

Hye-Seon Maeng, Hyoun-Su Lee, Tack-Don Han, Sung-Bong Yang, and Shin-Dug Kim
Parallel Processing System Laboratory
Dept. of Computer Science, Yonsei University, Seoul, 120-749, Korea
\{carchi, minerva, hantack, yang, sdkim\} @kurene.yonsei.ac.kr

Abstract

Dynamic load balancing on the workloads of the clustered workstations has emerged as a powerful solution for overcoming load imbalance\[1,4,5,7\]. In order to detect such imbalances, some load balancing methods check the average idle-time of the workstations periodically\[1,5,7\]. But in these methods load balancing cannot be performed until the end of a period even if load imbalance has occurred in the middle of the period. In this paper, we present a new threshold load balancing method for workstations which process the jobs with relatively long execution times. The new method decides a proper time to perform load balancing and does perform the balancing right after the detection of the load imbalance. We also show that a static load balancing method with a long period is suitable if the workstations have to deal with the jobs having unpredictable arrival times and relatively short execution times. The performance of the methods presented in this paper is compared with the method without load balancing as well as with the periodic methods in \[1,5,7\]. The experiments were done with an iterative data parallel problem called ISING problem\[3\]. The experimental results show that our methods outperform all the other methods that we compared.

1. Introduction

An iterative data parallel problem assumes a discrete model for a physical system and calculates a set of values for each domain point in the model. When an iterative data parallel problem is mapped onto a workstation clustering, the model domain is partitioned into a set of regions. These regions are distributed into the workstations on the clustering. During an iteration each workstation computes its domain points. Then it communicates the results with the workstations that have computed the neighboring domain points.

It is natural that several other jobs (we called them external jobs) are competing for the same resources of a workstation. Therefore, during an iteration some workstations may terminate their computations much later than others, while some finish earlier and sit idle if there are no or a few external jobs to compute. If such imbalance occurs frequently during the process of computing the whole problem, the overall execution time of the problem would be significantly increased because the synchronization is required after every iteration.

Recently dynamic load balancing on the workloads of the clustered workstations has emerged as a powerful solution for overcoming such imbalances\[1,4,5,7\]. In dynamic load balancing some of the workloads of heavily-loaded workstations may be assigned to relatively lightly-loaded workstations during runtime. In order to detect load imbalances, some balancing methods check the average idle-time of the workstations periodically\[1,5,7\], i.e., at every $k$ iterations, where $k$ is a positive constant. Whether a load imbalance has occurred or not can be determined by the value of the average idle-time of workstations in a period. If a load imbalance is detected, they decide to perform load balancing after the period.

A workstation may process various types of jobs and their arrivals to the workstation may not have any specific distribution such as an exponential distribution. In this paper we restrict the types of jobs and their arrivals to a workstation in such a way that a workstation is said to be in stable state if it mainly computes the external jobs whose execution times are quite long and the workstation is said to be in dynamic state if it has to deal with the external jobs with unpredictable arrival times and relatively short execution times. In cases that the arrivals of jobs are predictable, determining load imbalances can be
done easily. However, periodic load balancing methods in [1,5,7] some drawbacks in performing load balancing for the workstations in both states. In stable state these methods can perform load balancing quite effectively because an imbalance would be lasted for a long time. But they cannot perform load balancing until the end of a period even if the imbalance has occurred in the middle of the period. In dynamic state these methods may perform load balancing rather frequently since the average idle-time may not indicate a proper time to perform balancing.

In this paper we present a new threshold load balancing method for the workstations in stable state. The new method decides a proper time to perform load balancing and does perform the balancing right after the detection of the load imbalance. We also show that a static load balancing method with a long period is suitable in dynamic state. The performances of the methods presented in this paper are compared with the method without load balancing as well as with the periodic methods in [1,5,7]. The experiments were done with an iterative data parallel problem called ISING problem[3]. The experiment results show that our methods outperform all the other methods that we compared.

The rest of this paper is organized as follows. In Section 2, we give the details of our dynamic load balancing methods. Section 3 provides the experimental results.

2. New Dynamic Load Balancing Methods

2.1 Stable State Load Balancing (SSLB)

For developing an efficient load balancing method, the idle-time of each workstation and the cost of load balancing have been regarded as the major factors in detecting load imbalances as well as in deciding the proper time to perform load balancing. For example, if the average idle-time of the workstations during a period is greater than a certain threshold value which usually is decided by considering the load balancing cost, then load balancing is performed after that period.

These methods have a couple of drawbacks for stable state. First, load balancing cannot be performed until the end of the period even though the load imbalance has occurred in the middle of the period. Second, if the average idle-time is a little bit smaller than the threshold value, we call this a minor imbalance, we have to ignore the imbalance of the workloads even though it will be prolonged for quite a long time. If we try to perform load balancing whenever a minor imbalance has been detected, too much frequent load balancing cannot be avoided.

Consequently, the overheads of such load balancing would be enormous.

Now, we define the following to illustrate a new balancing method, static state load balancing (SSLB) method. Let \( T_{\text{max}} \) be the maximum time required by the workstation clustering to complete the \( j \)th iteration of the problem, \( j \geq 1 \), \( T_{\text{av}} \) be the average time required by the workstations to complete the \( j \)th iteration, and \( T_{\text{av}}^{j} \) be the average idle-time for the \( j \)th iteration. Then \( T_{\text{av}}^{j} = T_{\text{max}} - T_{\text{av}}^{j} \). Let \( k \), \( k \geq 1 \), be the length of a load balancing period in the number of iterations and \( C \) be a threshold value, \( C > 0 \). So the average idle-time \( T_{\text{av}}^{j} \) during \( k \) iterations can be obtained from the following equation.

\[
T_{\text{av}}^{j} = \frac{\sum_{i=1}^{k} T_{\text{av}}^{i}}{k}
\]

Then the periodic methods in [1,5,7] detect load imbalance when \( T_{\text{av}}^{j} > C \). In the SSLB method, we introduce a new threshold called a static threshold \( C_{s} \). We use the average idle-time \( T_{\text{av}}^{j} \) of the workstations during an iteration rather than the average idle-time \( T_{\text{av}}^{j} \) to perform load balancing whenever an imbalance is detected. Figure 1 shows an example that a load imbalance can be detected with \( C_{s} \).

![Figure 1: Detection of load imbalance with \( C_{s} \)](image)

When the average idle-time is smaller than the value of \( C_{s} \), and the imbalance of the workloads cannot be ignored since it has been prolonged for a long time, there is no
proper ways to detect such imbalance unless we reduce the value of $C_i$. But reducing the value of $C_i$ may cause too much frequent load balancing.

In order to overcome this situation, we introduce another threshold called adjustable threshold $C_s(n)$ for the $n$th iteration. $C_s(n)$ is a decreasing function of $n$ and its initial value is set to $C_i$ as shown in Figure 2. If $T_{s}^\ast > C_i$ and $T_{s}^{\ast\ast} > C_s(n)$, then we perform load balancing after the $n$th iteration. After a load balancing is done, $C_s(n)$ is reset to $C_i$. Because $C_s(n)$ is decreased gradually in time, the SSLB method is able to detect any minor load imbalance which is prolonged for a long time and therefore avoids frequent load balancing. Figure 2 shows an example that a minor load imbalance has been detected with $C_s(n)$.

![Figure 2: Detection of load imbalance with $C_s(n)$](image)

So the SSLB method decides to perform load balancing after the $n$th iteration if $T_{s}^\ast > C_i$, or $T_{s}^{\ast\ast} > C_s(n)$ is satisfied.

### 2.2 Dynamic State Load Balancing (DSLB)

Many researchers have omitted their consideration on load balancing for the workstations in dynamic state[1,2,5]. The external jobs in dynamic state can make the workloads of workstations change abruptly. So the balance information gathered so far could be obsolete for predicting future imbalances. When dynamic external jobs are entering, the reaction to the load imbalance at a certain point may cause another load imbalance. In this state we should depend only on the assumption of a fixed characteristics of the workloads. When we observe dynamic external jobs for a long period of time, for example days or weeks, we can determine certain characteristics of a workstation.

In dynamic state, we should perform static load balancing with a long fixed period regardless of what the value of the average idle-time may be so that we can avoid the load balancing overheads. The length of a period in dynamic state can be chosen so as to reflect the workload characteristics.

### 3. The Experimental Results

In this section, we present the experimental results of an iterative data parallel problem under various types of workloads. The ISING problem[3] has been chosen as a sample data parallel problem. It simulates energy particles which are allocated in a 2-dimensional grid. Every particle in a grid point may move in one of the 4-directions (up, down, left, and right), or may just stay at the original grid point. The probability that a particle moves in one of the four directions is the same. The test was done on a workstation clustering of ten Sun SPARCstation-5s.

![Figure 3: A sample load function for a workstation](image)

We choose Zaki[6]'s external job modeling, which can represent both stable and dynamic states. In this modeling each workstation $i$ has an independent load function $\ell_i$, and two parameters for $\ell_i$. The first parameter is the maximum load $m_i$ that specifies the maximum amount of loads per workstation. The value of $\ell_i$ is obtained by using a pseudo-random number generator to get a value between 0 and $m_i$. The second one is the duration $t_i$ of persistence indicates the amount of time before the next invocation of the pseudo-random number generator. A small value for $t_i$ implies rapidly changing loads with dynamic external jobs, while a large value indicates relatively stable loads with static external jobs. Figure 3
shows a sample load function for a workstation.

The experiments were performed under different maximum loads on different number $p$ of workstations. The comparisons of the performances for stable state with $t_f = 50$ sec are shown in Figures 4 through 8. A load balancing period and the threshold value of the first periodic method (PD1) are chosen as in [7] and those of the second periodic method (PD2) are chosen as in [1,5]. The period of PD1 is 10 iterations and PD2 sets the period time so that the load balancing cost is 5% of the period time. Most of the tests use 1,500,000 particles as the problem size and a test was performed with various numbers of workstations.

Figures 4 and 5 show the test results with $p=5$ and $p=10$ respectively. In the figure NOLB represents the method without load balancing. Both figures show that the SSLB method effectively reduces the execution times.

Figure 4: $p=5$ in stable state

Figure 5: $p=10$ in stable state

Figure 6: Various maximum loads

Figure 7: Varying the number of workstations

Figure 6 shows the results with various maximum loads when $p=5$. Figure 7 shows the results with various numbers of workstations. In this test all workstations are assigned 300,000 particles in order to compare the influence of $p$. The SSLB method also shows the better execution times than others for all the cases.
Figure 8: workstations in dynamic state

For the workstations in dynamic state, we used a short duration, \( t = 3 \) sec, of persistence. In this experiment three workstations have heavy loads and the rest have relatively light loads. Figure 8 shows that PD1 and PD2 have longer execution times than NOLB. As shown in the figure, the DSLB method outperforms others.

5. Conclusion

In this paper we have presented a new dynamic load balancing method for stable state. It decides a proper time to perform load balancing with static and adjustable thresholds and does perform load balancing right after the detection of the load imbalance. We also have shown that a static load balancing method with a long period is suitable for dynamic state. We have demonstrated the superiority of our load balancing methods using the experiments.

6. References


