An Efficient Dynamic Load Balancing using the Dimension Exchange Method for Balancing of Quantized Loads on Hypercube Multiprocessors*

Hwakyung Rim
Dept. of Computer Science
Sogang University
Seoul Korea 121-742
ackyung@arqlab1.sogang.ac.kr

Ju-wook Jang
Dept. of Electronic Eng.
Sogang University
Seoul Korea 121-742
jjang@ccs.sogang.ac.kr

Sungchun Kim
Dept. of Computer Science
Sogang University
Seoul Korea 121-742
ksc@arqlab1.sogang.ac.kr

Abstract

Dynamic load balancing on hypercube multiprocessors is considered with emphasis on quantized loads. Quantized loads are divisible only in a fixed size. First, we show that a direct application of the well-known Dimension Exchange Method(DEM) to quantized loads may result in difference in assigned loads to processors as large as $\log N$ *units after balancing for a hypercube of size* N*. Then we propose a new method which reduces the maximum difference by half to* $\lceil \frac{1}{2} \log N \rceil$*. The claim is proved both by analysis of possible cases incrementing the difference on each phase of balancing and by enumerating all possible combination of load for hypercubes of limited sizes using a computer. To estimate the accumulated effect of balancing instances under real-world parallel processing environment, a simulation for hypercube multiprocessors using SLAM II tool is performed. The result shows about* 30% *improvement in speedup which results from reduced processing time, which in turn results from reduced nonuniformity.*

1 Introduction

The balancing of loads has been an important matter in parallel and distributed processing which greatly affects the speedup in processing time. The underlying network topology or architecture, communication overheads involving transfer of data between processors, the size of load, the size of smallest unit of load and etc. should be carefully considered to obtain an efficient solution to this problem. From the viewpoint of balancing load among processors, most of the real-world processing loads can be classified into one of the following three categories.

- divisible in arbitrary sizes
- divisible in multiples of a fixed size (quantum)
- indivisible

Great part of real-world load falls into the second category[2][4] , which we call quantized loads. However many previous balancing methods fail to address the effect of quantized loads on their balancing performance. In this paper, we consider dynamic load balancing based on the DEM(Dimension Exchange Method)[3] on hypercube multiprocessor (or multicomputer)with emphasis on quantized loads.

In the DEM method, balancing is performed between two processors in such a way that the processor with larger load sends part of its load to the other processor so that they have as equal load as possible. The load distribution will be uniform after $\log N$ phases if the job is infinitely divisible. If we have quantized loads and the sum of quanta from two processors is odd, one processor should have one more unit of load. This effect can be accumulated to $\log N$ since there can be $\log N$ such phases. As the size of hypercube grows, the maximum difference of $\log N$ will also grow, which in turn increases the processing time.

In this paper we propose a new method which reduces the maximum difference by half to $\lceil \frac{1}{2} \log N \rceil$. The result shows about 30% improvement in speedup which results from reduced processing time, which in turn results from reduced nonuniformity. The rest of the paper is organized as follows. The DEM method

*This research is partly supported by Ministry of Informations and Communication. republic of Korea

is analyzed for its performance on quantized loads in Section 2. Our method is presented in Section 3 and simulation results are presented in Section 4. Section 5 concludes this paper.

2 The Dimension Exchange Method (DEM) for load balancing on hypercubes

In this section, we provide a brief explanation of the DEM method and show that a direct application of the DEM method on quantized loads in a hypercube of size N may result in nonuniform distribution with difference as large as $\log N$.

Assume we have an n-dimensional hypercube of N processors, where $N = 2^n$. Given an arbitrary distribution of quantized loads over the N processors, the balancing problem is to redistribute the quantized loads as uniform as possible. Let P_k (k is called processor id) and $W_{k,i}$ denote the k-th processor and the job assigned to the processor P_k before balancing phase i, where $k = 0, 1, 2, \ldots, N-1$ and $i = 0, 1, 2, \ldots, \log N - 1$. $W_{k,i}$ is a nonnegative integer representing the number of quantum loads. $W_{k,\log N}$ represents the load on P_k after completion of balancing. The load balancing is performed in $\log N$ phases. In phase i, balancing is performed along dimension i, where $i = 0, 1, 2, \ldots, \log N - 1$. For $j, k = 0, 1, 2, \ldots, N-1$, balancing is performed between P_k and P_j where the binary representation of k and j differs only in the i-th bit position.

In the DEM method, balancing is performed between two processors in such a way that the processor with larger load sends part of its load to the other processor so that they have as equal load as possible. If the load is infinitely divisible, each processor can always take exactly the same amount of load. But this is not a practical assumption regarding reality in job distribution. It is more practical to assume that the job is divisible in a fixed quantum size and thus $W_{k,i}$ should be a nonnegative integer for all k, i. Since P_2 has larger load (14) before balancing than that of P_0 (7), P_2 sends 3 units to P_0, which results in $W_{0,2} = 10$ and $W_{2,2} = 11$. Balancing on a hypercube using the original DEM can be described as follows:

The original DEM

For $i = 0$ to $\log N - 1$ do
  For all pairs of processors P_k and P_j such that the binary representation of k and that of j differ only in the i-th bit position do *in parallel*
  if $W_{k,i} \geq W_{j,i}$ then
    $W_{k,i+1} = \lceil (W_{k,i} + W_{j,i})/2 \rceil$
    $W_{j,i+1} = \lfloor (W_{k,i} + W_{j,i})/2 \rfloor$
  else
    $W_{k,i+1} = \lfloor (W_{k,i} + W_{j,i})/2 \rfloor$
    $W_{j,i+1} = \lceil (W_{k,i} + W_{j,i})/2 \rceil$
  end if

The worst case in the original DEM method for quantized loads

One major problem of the original DEM method, when directly applied to quantized loads, is as follows. A balancing phase between two neighboring processors may cause one unit of difference if the sum of the loads from the two processors is not an even number. This difference can be accumulated to result in the difference of $\log N$ after the $\log N$ phases.

Figure 1 illustrates one of the worst cases for balancing quantized loads on a hypercube of 16 processors. The arrows represent the dimensions along which balancing is performed. One can note that the DEM method will not change the number of quantized loads on any processor all through the $\log N$ (4 in this example) phases in this particular example. The processor with the smallest load has a units, while the processor with the largest load has $a + 4$ units. The a can be any nonnegative integer. If we put $a + 1$ in place of a, we have another hypercube of 16 processors in which there exists also a maximum difference of 4. If we combine the two hypercubes to form a hypercube of 32 processors, the resulting hypercube will have the difference as large as 5. Induction using the size of hypercubes from 16, 32 to 2^n leads us to the proof that balancing using the DEM method on a hypercube of N processors may result in the difference as large as $\log N$. There also exist many cases in which maximum differences are $\log N - 1, \log N - 2, \log N - 3, \ldots,$.

If the size of hypercube is N, the processor with the largest load will finish in $a + \log N$ time, which determines the processing time for the job. Thus the impact of the maximum difference on the processing time depends on the ratio between a and $\log N$. If a is not large compared with $\log N$, the effect of the maximum difference can not be ignored. As the size of hypercube grows, the maximum difference of $\log N$ will also grow, which in turn increases the processing time. If balancing is initiated many times during the processing of the whole job, the effect of the maximum difference will be accumulated to increase the processing time. Later we will show from simulation how much the maximum increases the processing time, when accumulated. In the following section, we propose a new method which reduces the maximum difference by half.

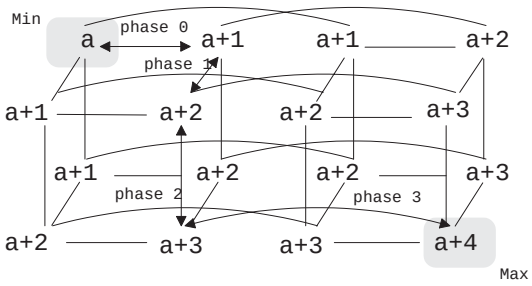

Figure 1. Balancing of quantized loads on a hypercube of 8 processors using the original DEM method: a worst case

3 Our method for balancing quantized loads on hypercube

In this section, we present a new method which improves the original DEM method especially for balancing quantized loads in hypercubes. In our method, we devised a scheme which prevents the difference from being incremented on every balancing phase. The following pseudo-code describes our method:

The our balancing method

For $i = 0$ to $\log N - 1$ do
  For all pairs of processors P_k and P_j
    such that the binary representation of k
    and that of j differ only in the i-th bit
    position with P_k, P_j having bit $0, 1$
    in the position, respectively do *in parallel*
    if $W_{k,i} + W_{j,i} = 2m + 1$ and m is
      an odd integer then
      $W_{k,i+1} = m$
      $W_{j,i+1} = m + 1$
    else if $W_{k,i} + W_{j,i} = 2m + 1$ and m is
      an even integer then
      $W_{k,i+1} = m + 1$
      $W_{j,i+1} = m$
        else
          $W_{k,i+1} = m$
          $W_{j,i+1} = m$
    end if

In this way, we increase the probability that loads of same type(odd or even) meet for balancing in the next phase. To see why, consider the two subcubes, 0-cube and 1-cube, where all the processors in the 0-cube(1-cube) have the i-th bit of the binary representation of the processor id equal to 0(1). The set of processors belonging to 0-cube(1-cube) will change as i goes from 0 to $\log N - 1$. Note that two processors which belong to the same subcube (0-cube or 1-cube) in phase i will meet for balancing in phase $i + 1$. Therefore if we route loads of same type (odd or even) to same subcube during balancing in phase i, it increases the probability that loads of same type meet for balancing in phase $i+1$. Furthermore, in our method it is guaranteed that the maximum difference in a balanced subcube is not incremented in immediate succession. Here a balanced subcube after phase i consists of 2^{i+1} processors whose binary representation of their id's differ only in the $i+1$ consecutive least significant bits. When the size of a balanced subcube reaches the size of the hypercube, the balancing is complete.

In our method, the difference may not be incremented in immediate succession. If the maximum difference in a balanced subcube is incremented in phase i, the maximum difference in a balanced subcube in phase $i + 1$ containing the half-size subcube is not incremented. One can think of several other cases based on the type (odd or even) of loads in the way loads meet for balancing. Those cases can be similarly verified for our claim that the maximum difference can not be incremented in succession. Assume we have four processors P_k, P_j, P_w, P_v where k and j (w and v) differ only in the i-th bit position and k and w have 0 in the i-th bit position. Further k and w (j and v) differ only in the $(i + 1)$-th bit position and k and j have 0 in the $(i + 1)$-th bit position. One example is when we have $k = 0, j = 1, w = 2, v = 3$. Table 1 shows 7 possible cases of incrementing difference based on the type of loads before balancing phase i. The values in the two rightmost columns show in which phase each case increments the maximum difference in a balanced subcube. Other cases can be similarly verified. Note that there is no case in which difference is incremented in immediate succession. This proves that the maximum difference using our method is less than or equal $\lceil \frac{1}{2} \log N \rceil$ for balancing of quantized loads in a hypercube of size N.

The worst case in the our balancing method for quantized loads

Even though using our method the maximum difference in a balanced subcube is not incremented in each phase, we found that it is possible for the maximum difference to be incremented again in every other phase. Figure 2 illustrates one worst case of load distribution before balancing. This proves that the maximum difference can be less than or equal to $\lceil \frac{1}{2} \log N \rceil$ for balancing of quantized loads in a hypercube of size N. This proof combined with the above proof leads us

case	$w_{j,i} + w_{k,i}$	$w_{w,i} + w_{v,i}$	$w_{k,i+1}$	$w_{j,i+1}$
0	2m+1, m even	$2l$+1,l even	m	m+1
1	2m+1, m even	$2l$+1,l odd	m	m+1
2	2m+1, m odd	$2l$,l even	m+1	m
3	2m+1, m odd	$2l$,l odd	m+1	m
4	2m, m even	$2l$+1,l even	m	m
5	2m, m even	$2l$+1,l odd	m	m
6	2m, m odd	$2l$,l even	m	m
case	$w_{w,i+1}$	$w_{v,i+1}$	Inc i	Inc i+1
0	l	l+1	1	0
1	l+1	l	1	0
2	l	l	1	0
3	l	l	1	0
4	l	l+1	1	0
5	l+1	l	1	0
6	l	l	0	1

Table 1. The 7 possible cases of incrementing difference regarding the type of quatized loads in successive balancing phases (Inc i:increment of difference in phase i, Inc i+1:increment of difference in phase i+1)

to the proof that, in the worst case, our method will produce the maximum difference of exactly $\lceil \frac{1}{2} \log N \rceil$. Therefore, the maximum difference in our scheme is only about half of the maximum difference in the original DEM when applied to the quantized loads. In the next section, we will show how the maximum differences resulting from repeated balancing affect the total processing time. The SLAM II simulation tool[1] is employed to show how our scheme performs better in a practical sense.

4 The simulation

Simulation is performed to satisfy two purposes. First is to verify the combinatorial proof that the maximum difference in the worst case for our method does not exceed $\lceil \frac{1}{2} \log N \rceil$.we enumerated all possible distribution using the numbers in the range as initial loads on processors in hypercube. Table 2 shows the number of combination which produce certain maximum differences. As expected, our method has 0 combinations that produce the maximum difference exceeding $\lceil \frac{1}{2} \log N \rceil$.

The second and more practical purpose is to show that our method performs better especially on quantized loads than the DEM method. The purpose of this simulation is to estimate how the reduced maximum diffrence of our method will affect the job balancing performance such as average queue length and speedup in processing time. SLAM II[1] is used as a simulation tool. Following parameters are used for our simulation [5]:

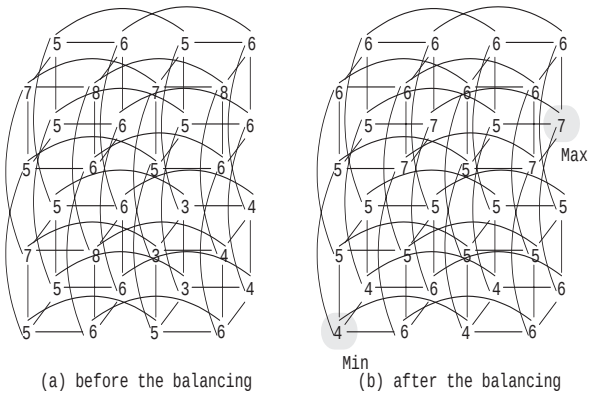

Figure 2. Balancing of quantized loads on a hypercube of 32 processors using our method: a worst case

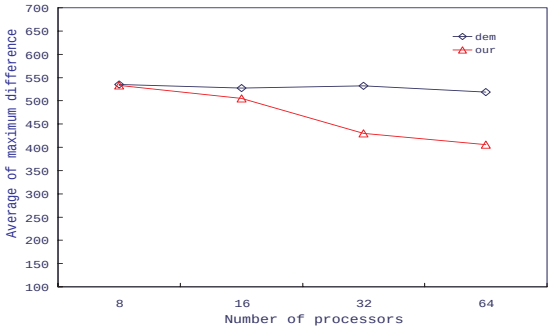

Figure 3. Average of the maximum difference for 1000λ

- The number of processors in a hypercube: 8, 16, 32, 64
- Architecure: loosely coupled multicomputer
- Number of processes(λ) generated in each processor: $1000\lambda, 5000\lambda$
- Arrival time: poisson distribution
- Service time: exponential distribution
- Threshold for overload: 50-60% CPU utilization
- No change in load distribution during balancing

	8 processors		16 processors
	DEM	Our	DEM
diff=0	50,438	87,034	148,226
diff=1	819,747	925,739	17,593,176
diff=2	211,170	68,802	12,502,375
diff=3	220	0	100,973
diff≥4	0	0	77,005
	16 processors	32 processors	
	Our	DEM	Our
diff=0	476,485	55,412	889,092
diff=1	24,949,040	117,986,702	273,339,227
diff=2	4,996,230	220,341,830	63,571,635
diff=3	0	10,254,632	12,543,611
diff≥4	0	1,704,989	0

Table 2. The maximum difference after balancing

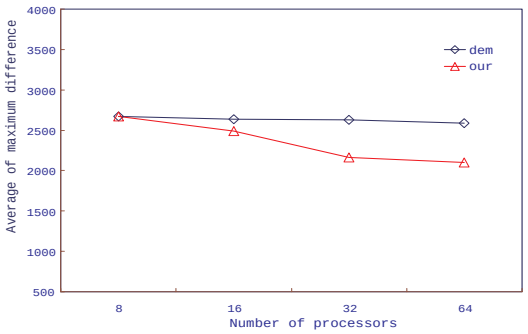

Figure 4. Average of the maximum difference for 5000λ

Figure 3 and Figure 4 shows average of maximum differences after balancing for $1000\lambda, 5000\lambda$, respectively. The average is taken over many balancing instances which happened during the 1,000,000 time units for the event-driven simulation. The maximum difference is directly related to the maximum of queue lengths in the hypercube multicomputer. Thus, the smaller the average is, the more uniform the load distribution is, which leads to reduced processing time. Comparison of our method against the DEM shows 30% improvement. Figure 5 shows improvement of our method in speedup over the DEM method. It shows about 30% improvement. The comparison also shows that improvement in speedup tends to grow for larger hypercube. This is as we expected. Since the processing time after balancing is $a + \log N$ for the DEM method and $a + \lceil \frac{1}{2} \log N \rceil$ for our method, when a is the minimum load after balancing which is assumed to

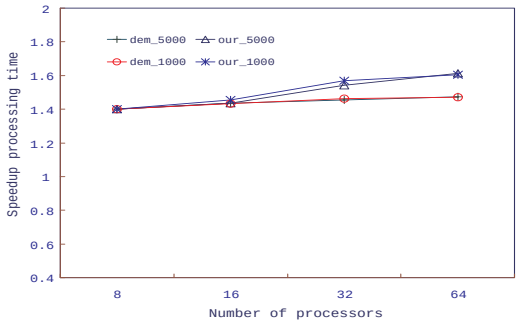

Figure 5. Balancing effect on speedup

take a time units to finish. Our method will take much less time for larger N.

5 Conclusion

A new method for dynamic load balancing on hypercubes is proposed which performs better especially on quantized loads than the well known dimension exchange method(DEM). The maximum difference in the number of load units(quanta of load) after balancing on a hypercube of size N is reduced from $\log N$ to $\lceil \frac{1}{2} \log N \rceil$. A simulation using the SLAM II on loosely coupled hypercube multicomputer shows about 30% improvement in speedup for processing time.

References

[1] A. Alan, B. Pritsker, *Introduction to Simulation and SLAM II*, John Wiley, 1986

[2] M. J. Berger, S. Bokhari, "A partioning strategy for non-uniform problems on multiprocessors, *IEEE Transactions on Computers*, C-26, pp. 570-580, 1987

[3] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessor," *Journal of Parallel and Distributed Computing*, 7, pp. 279-301, 1989

[4] G. Cybenko, T. G. Allen, "Parallel algorithms for classification and clustering," Proc. SPIE CAAASP, 1987

[5] M. H. Willebeek-Lemair, "Strategies for Dynamic Load Balancing on Highly Parallel Computers," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 4, No. 9, pp. 979-993, 1993