An Improved Direct Dimension Exchange Method for Load Balancing in k-ary n-cube Multicomputers

Siu-Cheung Chau *
Dept. of Mathematics and Computer Science, University of Lethbridge,
Lethbridge, Alberta, Canada, T1K 3M4
e-mail: chau@cs.uleth.ca

Ada Wai-Chee Fu
Dept. of Computer Science and Engineering,
Chinese University of Hong Kong, Shatin, Hong Kong
e-mail: adafu@cse.cuhk.edu.hk

Abstract *This paper presents an improved direct dimension exchange method (DDE) for synchronous load balancing for k-ary n-cube multicomputers. The new method is based on the improved DEM for hypercube. The original DEM when extended to the k-ary n-cube requires many iterations in exchanging loads between neighboring nodes to converge to a balanced state. The original DDE does not requires iteratively exchanging loads between nodes. It can balance the load with fewer communication steps and can provide better load balancing quality than DEM. The improved DDE has better load balancing quality than the original DDE and has the same complexity of the original DDE.*

Keywords: dynamic load balancing, k-ary n-cube, multicomputer

*This research was supported by the National Sciences and Engineering Research Council of Canada

1 Introduction

In a highly parallel multicomputer, time critical tasks can be executed quickly by splitting a task into sub-tasks. The sub-tasks are then executed in parallel in different processing nodes of the multicomputer. To maximize the benefit of multicomputers, the tasks assigned to each processing node should be roughly the same. In this paper, we will address the problem of load unbalance in the processing nodes.

Suppose that we can estimate the execution time of any task accurately and we also know the total number of tasks in advance, we can use a static load balancing method to assign balanced load to each of the processing nodes. Unfortunately, there exists a large class of non-uniform problems with uneven and unpredictable computation and communication requirement. For these problems, we have to deal with the load imbalance dynamically.

The dynamic load-balancing problem in

multicomputers deals with the reallocation of tasks between processing nodes during execution of an application. The dimension exchange method DEM was proposed for the hypercube multicomputers [2, 7]. By exchanging load information and tasks between nodes in each dimension of a hypercube, the load of each node in the hypercube will be roughly balanced. The maximum load difference between any nodes is at most n. Cybenko [2] showed that the order in which dimension the exchange in load takes place has no bearing on the load balance quality. Chau and Fu[1] proposed the improved dimension exchange method for a hypercube multicomputer that reduce the maximum load difference between processing nodes from n to $n/2$. Our proposed load balancing algorithm in k-ary n-cube multicomputers use similar approach as the improved DEM for hypercubes proposed by Chau and Fu[1].

Hosseini et al [3] extended the DEM for arbitrary structures using the technique of edge-coloring of graphs. Unfortunately, the DEM only works well for hypercube multicomputers where nodes exchange load with nodes in the same dimension only once. For arbitrary structures, balancing the load between pairs of directly connected processors in each dimension once may not distribute the load evenly. The process of exchanging load between pairs of nodes in each dimension may have to be repeated s times before some satisfactory balanced state is reached where $s \in N^+$.

Xu and Lau [11] proposed the generalized dimension exchange method GDE which is also based on edge-coloring of graphs. Instead of splitting the workload between two directly connected processors equally in each step, GDE splits the workload non-equally using the exchange parameter λ. If $\lambda = 0$, the workload distribution would not change at all after each iteration. If $\lambda = 1$, an iteration step would cause a complete swap of the workload of every pair of nodes but would leave the workload distribution unchanged. The original DEM has $\lambda = 1/2$. They also showed that DEM is optimal for the hypercube when $\lambda = 1/2$. For some structures such as chains, rings, meshs, and torus, a non-equal spitting leads to a smaller value for s. They also showed that the the number of iterations required in the GDE depends on the value of λ. Assume that each node u has w_u tasks and the graph can be colored by c colors. Figure 1 shows the GDE algorithm for an arbitrary structure. Each pass of the for loop is called a sweep. Only one sweep is necessary for the original DEM algorithm for hypercubes. DEM or GDE for an arbitrary structure would require s sweeps. DEM or GDE terminates when the difference in the number of task between neighboring nodes is less than or equal to 1. Hence, the maximum difference between any pairs of nodes is at least n, the number of dimensions of the structure.

GDE

While (not terminate)

for $l = 1$ to c

  for edge color l connecting nodes u and v

    node u exchanges with node v the current values of w_u and w_v

if $(w_u - w_v) > 1$,
  send $\lfloor(\lambda w_u - \lambda w_v)/2\rfloor$ tasks
  to node v
if $(w_v - w_u) > 1$,
  receive $\lfloor(\lambda w_v - \lambda w_u)/2\rfloor$ tasks
  from node v

$w_u = \lceil(1-\lambda)w_u + \lambda w_v\rceil$ if $w_u > w_v$
$w_u = \lfloor(1-\lambda)w_u + \lambda w_v\rfloor$ otherwise

Figure 1: The GDE algorithm for an arbitrary structure.

Xu and Lau [10] extended their GDE method to k-ary n-cube multicomputers. They derived the optimal λ's for k-ary n-cubes, rings, torus, and chains. They also showed that GDE is better than the diffusion method. After the application of GDE, the load difference between any pairs of nodes is bounded by $nk/2$ for k-ary n-cubes.

Wu and Shu[9] proposed the direct dimension exchange method (DDE) for load balancing in k-ary n-cubes. DDE can balance the load in one sweep and can provide better load balancing quality. After application of DDE, the load difference between any pairs of nodes is at most n. They also showed that DDE performs better than GDE in terms of the number of communication steps. Our improved DDE is based on DDE. It has the same complexity and communication cost of DDE but it can give even better load balancing quality. The maximum load difference between any pairs of nodes is bounded by $n/2$ instead of n.

The paper is organized as follows. Section 2 describes the direct dimension exchange method for chains and rings as proposed by Wu and Shu[9]. Section 3 describes the DDE method[9]. The improved DDE is presented in section 4. Section 5 concludes the paper.

2 DDE method for chains and rings

The DDE proposed by Wu and Shu[9] is based on the DDE method for chains and rings. Instead of balancing the load iteratively, the load of each ring in a k-ary n-cube is balanced directly. Figure 2 shows the DDE algorithm for chains and figure 3 shows the DDE algorithm for rings. After the application of DDE ring, the load difference between any pairs of nodes in the ring is at most 1.

DDE-chain

Let w_i be the number of tasks in node i, where $i = 0, 1, ..., k-1$.

1. Global information collection:
   Perform the scan with sum operation of w_i
   $W_i = \sum_{l=i}^{k-1} w_l$
2. Average load calculation:
   $T = W_0$, $w_{avg} = \lfloor T/k\rfloor$, and $R = T \bmod k$
3. Quota calculation:
   Calculate q_i, the quota of each node i
   $q_i = w_{avg} + 1$ if $i < R$
   $q_i = w_{avg}$ if $i \geq R$
   Calculate an accumulation quota of each node i
   $Q_i = \sum_{l=i}^{k-1} q_l$
4. Flow calculation:

Calculate the flow $x_{i,j}$ on each edge (i, j)
$x_{i-1,i} = Q_i - W_i$ for $i = 1, 2, ..., k-1$

Figure 2: The DDE algorithm for chains.

DDE-ring

Apply the DDE chain to the ring without considering the end-round edge $(k-1, 0)$.

If the flow from $x_{i,j}$ is from node i to node j, $x_{i,j}$ is negative. Otherwise, $x_{i,j}$ is positive.

Let n_p be the number of edges with $x_{i,j} > 0$
Let n_n be the number of edges with $x_{i,j} < 0$
Let n_z be the number of edges with $x_{i,j} = 0$

1. Let $m = \lceil k/2 \rceil$.
   If $n_n + n_z - n_p < 0$,
     let x_m be the m^{th} largest $x_{i,j}$ from all $x_{i,j} > 0$
   If $n_p + n_z - n_n < 0$,
     let x_m be the m^{th} smallest $x_{i,j}$ from all $x_{i,j} < 0$

2. For each edge, $x_{i,j} = x_{i,j} - x_m$

Figure 3: The DDE algorithm for rings.

3 DDE method for k-ary n-cubes

The DDE for k-ary n-cubes is similarly to the DEM for hypercubes. Instead of dividing a hypercube into 2 sub-cubes, we divide an k-ary n-cube into k k-ary $(n-1)$-cubes. This is done by applying the DDE ring for the most (or least) significant dimension. After applying DDE ring once, the maximum load difference between any pairs of k-ary $(n-1)$-cubes is at most k^{n-1}. The process is repeated for each dimension of an k-ary n-cube. Figure 4 shows the DDE for k-ary n-cubes. After the application of DDE, Wu and Shu[9] showed that the maximum load difference between any pair of nodes in an k-ary n-cube is bounded by n.

DDE for k-ary n-cubes

---

for $l = 0$ to $n-1$

  Apply the DDE ring algorithm to k^{n-1} rings in the l dimension

  Exchange tasks according to the flow

  Each node i updates its weight w_i

Figure 4: The DDE algorithm for k-ary n-cubes.

4 Improved DDE method for k-ary n-cubes

The maximum load difference between any pair of nodes can be further reduced by $n/2$ if we divide the load more carefully between the k k-ary $(n-1)$-cubes in applying the DDE-ring. Suppose the total number of tasks in the rings of the first dimension of a k-ary n-cubes is T_l and $T_l \bmod k = R_l$ where $l \in \{0..k-1\}$. In DDE, we assign $\lfloor T_l/k \rfloor + 1$ tasks to the first R_l nodes and $\lfloor T_l/k \rfloor$ tasks to the rest of the nodes. Instead of assigning one more task to the first R_l nodes all the time, we can assign one more task to the last R_l nodes half of the time. By alternating where the R_l tasks go, the maximum load difference between any pair of the k k-ary $(n-1)$-cubes can be reduced from k^{n-1} to $k^{n-1}/2$. After applying the improved DDE, the maximum load difference can be improved to $n/2$.

The Improved DDE is basically the original DDE with adjustment in the original DDE-chain. The rest of the algorithm is exactly the same. The improved DDE-chain is showed in figure 5.

Improved DDE-chain

Let w_i be the number of tasks in node i, where $i = 0, 1, ..., k-1$.

Let node $i = i_0 i_1 ... i_{n-1}$

Let l be the dimension when DDE-chain is applied in the DDE algorithm

1. Global information collection:
   Perform the scan with sum operation of w_i
   $W_i = \sum_{l=i}^{k-1} w_l$

2. Average load calculation:
   $T = W_0$, $w_{avg} = \lfloor T/k \rfloor$, and $R = T \bmod k$

3. Quota calculation:
   Calculate q_i, the quota of each node i
   If i_{n-1} is even then
   $q_i = w_{avg} + 1$ if $i_l < R$
   $q_i = w_{avg}$ if $i_l \geq R$
   If i_{n-1} is odd then
   $q_i = w_{avg} + 1$ if $i_l + R \geq k$
   $q_i = w_{avg} + 1$ if $i_l + R < k$
   Calculate an accumulation quota of each node i
   $Q_i = \sum_{l=i}^{k-1} q_l$

4. Flow calculation:
   Calculate the flow $x_{i,j}$ on each edge (i,j)
   $x_{i-1,i} = Q_i - W_i$ for $i = 1, 2, ..., k-1$

Figure 5: The Improved DDE algorithm for chains.

Theorem 1: For an k-ary n-cube, after execution of the improved DDE, the load difference between any pairs of nodes i and j is bounded by $n/2$.

Proof: In the improved DDE, the first application of DDE-ring divide the load roughly equally between the k k-ary $(n-1)$-cubes. Due to the new conditions imposed in the improved DDE-chain, at most half of the nodes in one of the k-ary $(n-1)$-cube has one more task than the nodes in another k-ary $(n-1)$-cube. This happens when all the nodes with their rightmost bit being odd (even) have one more task than the nodes in another k-ary $(n-1)$-cube and all the other nodes with their rightmost bit being even (odd) have the same number of task as the nodes in another k-ary $(n-1)$-cube. Hence, one k-ary $(n-1)$-cube has at most $k^{n-1}/2$ tasks than another k-ary $(n-1)$-cube.

Let T be the total number of tasks in the k-ary n-cubes. Let t_l be the total number of tasks in the lth ring where $l \in \{1..k^{n-1}\}$. After the first application of the DDE-ring, an k-ary $(n-1)$-cubes with the most tasks $T_{max,1}$ will have at most $\lceil t_l/k \rceil$ tasks for odd l and t_l/k for even l. Since $\lceil t_l/k \rceil$ is less than or equal to $(t_l/k$ $+ (k-1)/k)$, $T_{max,1} \leq T/k + (k-1)/k * k^{n-1}/2$ tasks.

Similarly, an k-ary $(n-1)$-cubes with the least tasks $T_{min,1}$ will have at least $\lfloor t_l/k \rfloor$ tasks for odd l and t_l/k for even i. Since $\lfloor t_l/k \rfloor$ is greater than or equal to (t_l/k - $1/k$), $T_{min,1} \geq T/k - 1/k * k^{n-1}/2$ tasks.

After the second application of the DDE-ring, an k-ary $(n-2)$-cubes has at most $T_{max,2}$ and at least $T_{min,2}$ tasks.

$$\begin{aligned} T_{max,2} &\leq (T/k + (k-1)/k * k^{n-1}/2)/k \\ &\quad +(k-1)/k * k^{n-2}/2 \\ &\leq T/k^2 + 2(k-1)/k * k^{n-2}/2 \end{aligned}$$

$$\begin{aligned} T_{min,2} &\geq (T/k - 1/k * k^{n-1}/2)/k \\ &\quad -1/k * k^{n-2}/2 \\ &\geq T/k^2 - 2/k * k^{n-2}/2 \end{aligned}$$

After the lth application of the DDE-ring where $l \in \{1..n\}$, an k-ary $(n-l)$-cubes has at most $T_{max,l}$ and at least $T_{min,l}$ tasks.

$$T_{max,l} \leq T/k^l + l(k-1)/k * k^{n-l}/2$$

$$T_{min,l} \geq T/k^l - l/k * k^{n-l}/2$$

After applying the improved DDE, the maximum difference between any pairs of nodes in an k-ary n-cube is at most D tasks.

$$\begin{aligned} D &= T_{max,n} \text{ - } T_{min,n} \\ &= (T/k^n + n(k-1)/k * k^{n-n}/2) \\ &\quad \text{ - } (T/k^n - n/k * k^{n-n}/2) \\ &= (n(k-1)/k + n/k)/2 \\ &= n/2 \end{aligned}$$

Hence, the maximum task difference between any pairs of nodes is at most $n/2$.

5 Conclusions

The DDE method extended the DEM algorithm to the k-ary n-cube. Compared to the GDE method, the DDE is faster, reduces communication, keeps better locality, and balances the load better[9]. The improved DDE retains all the good properties of the original DDE with the same communication cost and further improves the load balancing quality. After the applicationof the improved DDE, the maximum load difference between any pairs of nodes is most $n/2$. For the DDE, the load difference is at most n. Hence, the improved DDE has a much better load balancing quality than the original DDE.

References

[1] S.C. Chau and Adu Fu. An improved dimension exchange method for Load Balancing for hypercube multicomputers. manuscript.

[2] G. Cybenko. Dynamic load-balancing for distributed memory multicomputers, *Journal of Parallel and Distributed Computing*, (7)2, October 1989, pages 279-301.

[3] S.H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel*

and Distributed Computing, October 1990, Pages 160-166.

[4] S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software*, September 1988, Pages 69-77.

[5] K.G. Shin and Y. Chang. Load sharing in distributed real-time system with state-change broadcasts. *IEEE Transaction on Computers*, (38)8, August 1989, Pages 1124-1142.

[6] N.G. Shivaratri and P. Krueger. Load distributing for locally distributed systems, *IEEE Computers*, (25)12, December 1992, Pages 33-44.

[7] M. Willebeek-Lemair and A.P. Reeves. Strategies for dynamic load-balancing on highly parallel computers. *IEEE Transaction on Parallel and Distributed Systems*, (4)9, September 1993, Pages 979-993.

[8] M. Wu and W. Shu. A load balancing algorithm for n-cube. *Proceedings of the 1996 International Conference on Parallel Processing*, IEEE Computer Society, 1996, Pages 148-155.

[9] M. Wu and W. Shu. The Direct Dimension Exchange Method for Load Balancing in k-ary n-cubes. *IEEE Symposium on Parallel and Distributed Processing*, Oct. 1996, pages 366-369.

[10] C.Z. Xu and F.C.M. Lau. The generalized dimension exchange method for load balancing in k-ary n-cube and variants. *Journal of Parallel and Distributed Computing*, (24)1, January 1995, Pages 72-85.

[11] C.Z. Xu and F.C.M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, (16)4, December 1992, Pages 385-393.