Load Balancing Between Heterogenous Computing Clusters

Siu-Cheung Chau *
Dept. of Physics and Computing, Wilfrid Laurier University,
Waterloo, Ontario, Canada, N2L 3C5
e-mail: schau@wlu.ca

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

Abstract

More and more organizations have computing clusters located in different places. The distance between the computing clusters can be quite far away and the computing clusters can also differ in size. The load in one cluster may be very high while the other clusters may have nothing running on the system. A higher throughput can be achieved if load balancing is added. In this paper, we propose a simple and efficient load balancing method to balance load for computing clusters that are quite far away from each other. We assume that the computer clusters are connected in the form of a hypercube. We then use the load balancing method for hypercubes to balance the load between heterogenous computing clusters. After applying the propose method, the maximum load difference between any two computing clusters can be at most one more than the optimal solution.

Keywords: Load balancing, Computing Clusters, Hypercube.

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

1 Introduction

In a computing cluster, 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 a computing cluster. To maximize the benefit of computing clusters, the tasks assigned to each processing node should be roughly the same. Hence, load balancing is necessary to balance load between the processing nodes within a computing cluster. If an organization has more than one computing clusters and the computing clusters are located in different places, we also want to balance the load between the computing clusters to improve the overall throughput.

An example of such organization is Sharcnet. Sharcnet is a partnership between a number of universities and colleges in Canada. All the members of Sharcnet are located in southwestern Ontario, Canada. Currently, Sharcnet has five members with computing clusters. SHARCNET is structured as a ”cluster of clusters”.

Built on the latest Alpha processors, SHARCNET clusters consist of four-processor, 833Mhz, Alpha SMP (symmetric multi-processors) systems connected via Quadrics interconnection technology. The Alpha SMP are then connected together through Gigabit ethernet. Clusters of 24 SMPs (96 processors) are deployed at McMaster University and 27 SMPs (108 processors) are deployed at the University of Guelph. At The University of Western Ontario, there are three clusters, one of 12 SMPs (48 processors), one of 36 SMPs (144 processors) and one of 64 HP processors are deployed. The two SMP clusters are also connected using Nortel’s optical DWDM (Dense Wave Division Multiplexing). Smaller development clusters (8 processors) are deployed at the University of Windsor and at Wilfrid Laurier University. Depending on funding from the government, more computing clusters will be added to current member sites and to other universities such as the University of Waterloo, Brock University, and Trent University in the region.

The computing clusters within Sharcnet are quite far away from each other and not all of them are connected through dedicated fibre optic lines. In fact, only two computing clusters within Sharcnet are connected by fibre optics lines. The rest of them are connected through internet access line. For example, the distance between Wilfrid Laurier University and the University of Windsor is almost four hundred kilometers and there is no dedicated communication line between the two locations. Hence, the load balancing method to balance the load between computing clusters should have as few communications as possible.

Each computing cluster within Sharcnet is controlled by a master node that is outside of the cluster.

Hence, we can assume that the master node of each of the computing cluster are connected as a node of a hypercube. By assuming that the master nodes are connected as a hypercube, we can group clusters that are connected directly through the high-speed link in the same sub cube to take advantage of their connections. Furthermore, we can also make use of the topology of a hypercube to reduce the amount of communications during load balancing. To make things slightly simpler, we assume that the computing clusters are of the same size. We will relax this assumption later.

A n-dimensional hypercube is a network with $N = 2^n$ nodes. Each node can be coded by a binary sequence of length n. Two nodes are connected if their binary sequences differ in exactly one position. Each node $v = v_1v_2...v_n$ is connected to n nodes. Node v is connected to node $u = u_1u_2...u_n$ if $v_i \neq u_i$ and $v_j = u_j$ for $j \in \{1..n\}$ and $j \neq i$. The link between v and u is also called the link of dimension i.

2 Previously proposed load balancing schemes for hypercubes

If 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. Load balancing is not necessary. Unfortunately, there exists a large class of non-uniform problems with uneven and unpredictable computation and communication requirements. For these problems, we have to deal with the load imbalance dynamically.

Many load-balancing algorithms for hypercube have been proposed. They fall into two types: Synchronous load balancing and asynchronous load balancing. Synchronous load balancing [2, 3] is one in which the balancing is done on a global synchronous manner. Asynchronous load balancing [4, 5] is one in which each processing node can initiate the load balancing at any given time.

Two examples of asynchronous load balancing are Receiver initiated diffusion (RID) and Sender initiated diffusion (SID) [4, 5]. In RID, an under loaded node requests extra task from overloaded nodes. In SID, an overloaded node tries to find an under loaded node and transfers some of its tasks to the under loaded node.

Willebeek-Lamair and Reeves [2] showed that RID always outperforms SID. In RID, when the load of an under loaded node drops below a predetermined value, the under loaded node starts to send load balancing requests to its neighbors. Its neighbors then send their current load information to the under loaded node. Based on the load information of its neighbors, the under loaded node calculates the task that each of its neighbors has to send to it.

Cybenko [1] showed that the diffusion methods are inferior to the dimension exchange method in

terms of their efficiencies and balance qualities. For the dimension exchange method (DEM) [7, 1, 2], all node pairs whose addresses differ only in the least significant bit, balance their load between themselves. Similarly, all node pairs whose addresses differ in only the second least significant bit, balance the load between themselves. The above process is repeated until each node has exchanged and balanced its load with its neighbors in all the dimensions. After execution of the DEM, the load difference between any pair of nodes is at most n.

Let w_u be the number of tasks in node u and $\oplus$ be the bitwise exclusive OR operator. Figure 1 shows the DEM algorithm for a hypercube.

DEM

for $k = 0$ to $n - 1$
node u exchange with node v the current value of w_u
and w_v, where $(u \oplus v) \oplus 2^k = 0$
if $(w_u - w_v) > 1$, send $\lfloor (w_u - w_v)/2 \rfloor$ tasks to node v
if $(w_v - w_u) > 1$, receive $\lfloor (w_v - w_u)/2 \rfloor$ tasks from node v
$w_u = \lceil (w_u + w_v)/2 \rceil$ if $w_u > w_v$
$w_u = \lfloor (w_u + w_v)/2 \rfloor$ otherwise

Figure 1: The DEM algorithm for hypercubes.

The cube walking algorithm (CWA)[3] can balance the load better than the DEM. After applying the CWA to a hypercube, the load difference between any nodes is at most one. The load difference is optimal because a load difference of zero can only happen if the total load is divisible by the total number of nodes N. The number of communication steps in the CWA is the same is DEM. However, CWA requires an additional $nN/2$ messages of size $O(n)$ to send the load information vectors where N is the number of nodes in an n-dimensional hypercube.

In order to reduce communications between computing clusters and get better load balancing quality that is comparable to CWA, we propose to use an improved version of the DEM instead of the CWA. In the next section, we describe the improved DEM for a hypercube.

3 Improved Dimension Exchange Algorithm for hypercubes

After the execution of the original DEM, the load difference between any two nodes is bounded by n. We can view the exchange of information and task migration in the least significant dimension of pairs of nodes as dividing the tasks among the two $(n-1)$-cubes roughly equally. However, the difference in the total number of task between the two $(n-1)$-cubes may be as high as $N/2$. This happens when the total number of tasks between every pair of nodes in the same dimension is an odd number and all the nodes in one sub-cube have one more task than the nodes in the other sub-cube.

Similarly, the exchange of load information and task migration in the second least significant dimension can be viewed as dividing tasks roughly equally between the two $(n-2)$-cubes in each of the $(n-1)$-cubes. The difference in the total number of tasks between two $(n-2)$-cubes in a $(n-1)$-cube is bounded by $N/4$.

Using the above reasoning, after the load exchange in the first dimension, one $(n-1)$-cube can have at most $T/2 + N/4$ tasks and the other one can have at least $T/2 - N/4$ task. After the load exchange in the second dimension, some $(n-2)$-cubes have at most $(T/2+N/4)/2+N/8$ tasks and some $(n-2)$-cubes have at least $(T/2-N/4)/2-N/8$ tasks. That is, some $(n-2)$-cubes have at most $(T/2^2+2N/2^3)$ tasks and some have at least $(T/2^2-2N/2^3)$ tasks. Hence, after the execution of the DEM, some nodes have at most $T/2^n+nN/2^{n+1}$ tasks and some have at least $T/2^n-nN/2^{n+1}$ tasks. After applying the DEM, the maximum load difference between any two nodes is at most n.

The maximum difference can be reduced by $n/2$ if we divide the tasks more carefully between the sub-cubes in the load exchange of each dimension. Consider two neighboring nodes u and v with tasks w_u and w_v. Suppose $(w_u - w_v) > 0$. Instead of simply sending $\lfloor (w_u - w_v)/2 \rfloor$ tasks to node v, we may want to send $\lceil (w_u - w_v)/2 \rceil$ tasks to node v. By having a better scheme on which node should send $\lfloor (w_u - w_v)/2 \rfloor$ tasks or $\lceil (w_u - w_v)/2 \rceil$ tasks to its neighboring nodes, we could get a tighter bound on the maximum task difference. The improved DEM is shown in figure 2.

Improved DEM

Let node $u = u_1u_2...u_n$ and $v = v_1v_2...v_n$.

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

node u exchange with node v the current value of w_u

and w_v, where $(u \oplus v) \oplus 2^k = 0$

if $k \neq n$

if $(w_u - w_v) > 0$ and $u_{k+1} = 0$,

send $\lfloor (w_u - w_v)/2 \rfloor$ tasks to node v

$w_u = \lceil (w_u + w_v)/2 \rceil$

if $(w_u - w_v) > 0$ and $u_{k+1} = 1$,

send $\lceil (w_u - w_v)/2 \rceil$ tasks to node v

$w_u = \lfloor (w_u + w_v)/2 \rfloor$

if $(w_v - w_u) > 0$ and $u_{k+1} = 0$,

receive $\lceil (w_v - w_u)/2 \rceil$ tasks from node v

$w_u = \lceil (w_u + w_v)/2 \rceil$

if $(w_v - w_u) > 0$ and $u_{k+1} = 1$,

receive $\lfloor (w_v - w_u)/2 \rfloor$ tasks from node v

$w_u = \lfloor (w_u + w_v)/2 \rfloor$

else

if $(w_u - w_v) > 1$

send $\lfloor (w_u - w_v)/2 \rfloor$ tasks to node v

$w_u = \lceil (w_u + w_v)/2 \rceil$

if $(w_v - w_u) > 1$

receive $\lfloor (w_v - w_u)/2 \rfloor$ tasks from node v

$w_u = \lfloor (w_u + w_v)/2 \rfloor$

Figure 2: The improved DEM algorithm for hypercubes.

Figure 3 shows an example of the DEM algorithm in operation. A total of 33 task migrations take place. The maximum difference between any two nodes is 2. Figure 4 shows an example of the improved DEM algorithm in operation. A total of 25 task migrations take place. The maximum difference between any two nodes is 0. The examples show that the improved DEM should be able to provide better load balancing quality with less communication cost.

Theorem 1: After applying the improved DEM to an n-dimensional hypercube, the load difference between any two nodes u and v is at most $n/2$.

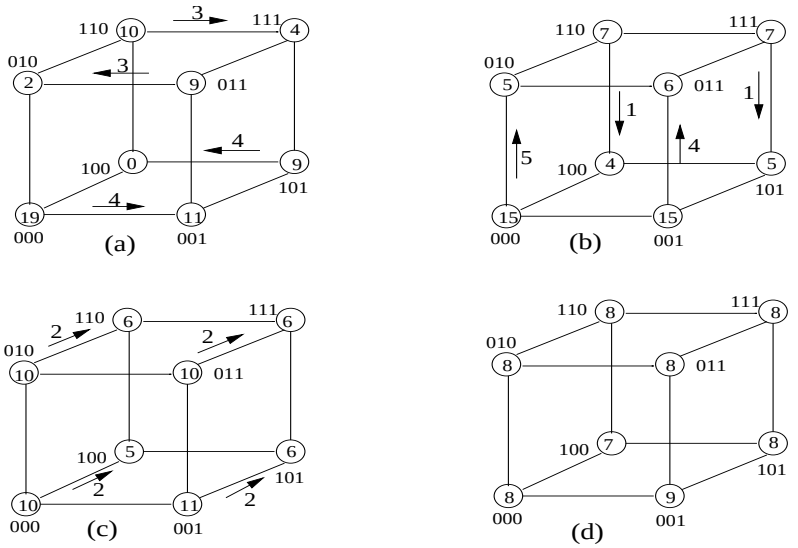

Figure 3. Example to show the execution of the DEM.

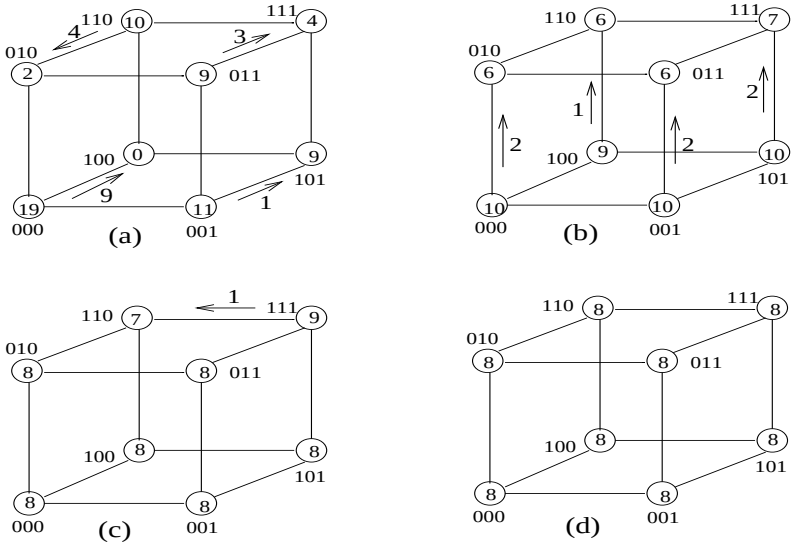

Figure 4. Example to show the execution of the improved DEM.

Proof: In the first step of the improved DEM, the load is divided roughly equally between the two $(n-1)$-cubes within the n-cubes. Due to the new conditions imposed in the improved DEM, at most half of the nodes in one of the $(n-1)$-cube has one more task than the nodes in the other $(n-1)$-cube. Hence, a $(n-1)$-cube has at most $N/2^2$ tasks more than the other $(n-1)$-cube.

Let T be the total number of tasks in the n-cubes. After the first application of the dimension exchange, an $(n-1)$-cubes with the most tasks $T_{max,1}$ will have at most N/4 more than the other $(n-1)$-cubes with the least tasks $T_{min,1}$. Hence, $T_{max,1}$ will have at most $T/2 + N/2^3$ and $T_{min,1}$ has at least $T/2 - N/2^3$ tasks.

Similarly, after the load exchange in the second dimension, an $(n-2)$-cubes has at most $T_{max,2}$ and at least $T_{min,2}$ tasks.

$$\begin{aligned} T_{max,2} &\leq (T/2 + N/2^3)/2 + N/2^3 * 1/2 \\ &\leq T/2^2 + 2N/2^4 \\ T_{min,2} &\geq (T/2 - N/2^3)/2 - N/2^3 * 1/2 \\ &\geq T/2^2 - 2N/2^4 \end{aligned}$$

After the load exchange in the ith application where $i \in \{1..n\}$, an $(n-l)$-cubes has at most $T_{max,i}$ and at least $T_{min,i}$ tasks.

$$\begin{aligned} T_{max,l} &\leq T/2^i + iN/2^{i+2} \\ T_{min,l} &\geq T/2^i - iN/2^{i+2} \end{aligned}$$

After applying the improved DEM, the maximum difference between any pairs of nodes in an n-dimensional hypercube is at most D tasks.

$$\begin{aligned} D &= T_{max,n} - T_{min,n} \\ &= T/2^n + nN/2^{n+2} - T/2^n + nN/2^{n+2} \\ &= n/4 + n/4 \\ &= n/2 \end{aligned}$$

From theorem 1, the improved DEM has a better upper bound in load difference between any two nodes compared to the original DEM. Simulations were also done for both the DEM and the improved DEM for a hypercube from dimension 3 to dimension 12. The load are randomly assigned to each node of a hypercube and DEM or the improved DEM is applied to balance the load. The simulation is repeated 100,000 times. The results are listed in table 1 and table 2.

d	Max load difference between any two nodes								Avg
n	0	1	2	3	4	5	6	7	diff
3	780	49256	49170	793	0	0	0	0	1.50
4	0	12827	73855	13318	0	0	0	0	2.00
5	0	863	48776	49446	915	0	0	0	2.50
6	0	3	13053	73697	13237	10	0	0	3.00
7	0	0	864	48815	49461	860	0	0	3.50
8	0	0	9	14547	73490	11952	2	0	3.97
9	0	0	0	859	48865	49393	883	0	4.50
10	0	0	0	2	11820	73180	14991	7	5.03
11	0	0	0	0	905	49152	49024	919	5.50
12	0	0	0	0	4	13340	73414	13242	6.00

Table 1: The maximum load difference between any two nodes after applying DEM

From the data listed in table 1, after applying the original DEM to a hypercube, the expected load difference between any two nodes is roughly $n/2$. The difference is much lower than the upper bound of n. This can be explained by the fact that the upper bound can only happen for a special case.

From the data in table 2, after applying the improved DEM, the expected load difference between any two nodes is always less than 2. If the dimension of the hypercube is less than or equal to 12, the maximum load difference for any two nodes is at most 2. This happens because the load in each sub-cube of the hypercube would be roughly the same after one step of load balancing. The load balancing quality of the improved DEM compares favorably with the CWA where the maximum load difference is at most one.

The improved DEM requires only nN communications between the computing clusters. The CWA requires $nN/2$ more messages of size $O(n)$ compared to the improved DEM. For distributed load balancing with global information about load in all the nodes, an all-to-all message exchange between the nodes requires $N(N-1)$ communications. If we want to minimize the communications between the nodes, the improved DEM would be a very good alternative and the resulting load balancing quality is almost as good as the CWA or the optimal solution. Hence, the improved DEM is a good candidate for load balancing between homogeneous computing clusters of the same size.

	Max load difference between any two nodes								Avg
n	0	1	2	3	4	5	6	7	diff
3	9375	87483	3142	0	0	0	0	0	0.94
4	2030	87595	10375	0	0	0	0	0	1.08
5	104	79546	20350	0	0	0	0	0	1.20
6	0	69903	30097	0	0	0	0	0	1.30
7	0	60765	39235	0	0	0	0	0	1.39
8	0	52938	47062	0	0	0	0	0	1.47
9	0	47435	52565	0	0	0	0	0	1.53
10	0	43649	56351	0	0	0	0	0	1.56
11	0	39580	60420	0	0	0	0	0	1.60
12	0	34671	65329	0	0	0	0	0	1.65

Table 2: The maximum load difference between any two nodes after applying the improved DEM

4 Load Balancing Between Heterogenous Computing Clusters

Unfortunately, not all computing clusters within an organization are of the same size and consist of the same processing nodes. For example, the computing clusters within Sharcent are of different size and consist of different processing nodes. With a little bit of adjustment, the Improved DEM would also work for computing clusters of various sizes and consist of different processing nodes.

Consider two nodes u and v within a hypercube. Let $w_u = 640$ tasks and $w_v = 960$ tasks. Suppose u is a computing cluster with $p_u = 64$ processing nodes and v has $p_v = 32$ processing nodes. Instead of sending (960-640)/2 tasks from v to u as indicated in the improved DEM, we should take into account the number of processing nodes in each computing clusters. After load balancing, the number of tasks that are assigned to each processing node in all the computing clusters should be roughly the same. The average load in each processing nodes in u, and v is $w_u/p_u = 640/64 = 10$ and $w_v/p_v = 960/32 = 30$ respectively. In order to balance the load between u and v, we should send $w_v - (w_u + w_v)/(p_u + p_v) * p_v = 427$ tasks from node v to node u. After the transfer, the average load in each processing nodes in u, and v will be around 17. Figure 5 lists the revised improved DEM that balances load between computing clusters of different size.

Revised Improved DEM

Let node $u = u_1u_2...u_n$ and $v = v_1v_2...v_n$.

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

node u exchange with node v the current value of w_u

and w_v, where $(u \oplus v) \oplus 2^k = 0$,

and the number of processing nodes p_u and p_v

if $k \neq n$

if $(w_u/p_u - w_v/p_v) > 0$ and $u_{k+1} = 0$,

send $\lfloor w_u - (w_u + w_v)/(p_u + p_v) * p_u \rfloor$ tasks to node v

$w_u = \lceil w_u - (w_u + w_v)/(p_u + p_v) * p_u \rceil$

if $(w_u/p_u - w_v/p_v) > 0$ and $u_{k+1} = 1$,

send $\lceil w_u - (w_u + w_v)/(p_u + p_v) * p_u \rceil$ tasks to node v

$w_u = \lfloor w_u - (w_u + w_v)/(p_u + p_v) * p_u \rfloor$

if $(w_v/p_v - w_u/p_u) > 0$ and $u_{k+1} = 0$,

receive $\lceil w_v - (w_u + w_v)/(p_u + p_v) * p_v \rceil$ tasks from node v

$w_u = \lceil w_v - (w_u + w_v)/(p_u + p_v) * p_v \rceil$

if $(w_v/p_v - w_u/p_u) > 0$ and $u_{k+1} = 1$,

receive $\lfloor w_v - (w_u + w_v)/(p_u + p_v) * p_v \rfloor$ tasks from node v

$w_u = \lfloor w_v - (w_u + w_v)/(p_u + p_v) * p_v \rfloor$

else

if $(w_u/p_u - w_v/p_v) > 1$

send $\lfloor w_u - (w_u + w_v)/(p_u + p_v) * p_u \rfloor$ tasks to node v

$w_u = \lceil w_u - (w_u + w_v)/(p_u + p_v) * p_u \rceil$

if $(w_v/p_v - w_u/p_u) > 1$

receive $\lfloor w_v - (w_u + w_v)/(p_u + p_v) * p_v \rfloor$ tasks from node v

$w_u = \lfloor w_v - (w_u + w_v)/(p_u + p_v) * p_v \rfloor$

Figure 5: The revised improved DEM algorithm for Computing Cluster of various sizes.

The method listed above can also handle computing clusters with different processing nodes. We can assign a value s_u to represent the processing power of a processing nodes in u. The value s_u would be used as a factor in adjusting the value p_u. The value p_u would be calculated by multiplying the number of processing nodes in u by the processing power s_u. We can then use the load balancing method as listed in Figure 5.

5 Summary

A method to balance the load between heterogenous computing clusters that are far apart is proposed. It is based on the improved dimension exchange method (DEM) for synchronous load balancing for hypercube architecture. Although, theoretically, the proposed method cannot provide the optimal load balancing quality, in practice, it is very close to the optimal if the number of computing clusters is less than 4096. In fact, the maximum load difference between any two computing clusters can be at most one more than the optimal solution. Furthermore, the proposed method requires a lot fewer communications between the computing clusters. In the near future, we will try to conduct simulation study of our load-balancing method and also try to conduct load balancing experiments between the computing clusters of Sharcnet.

References

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

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

[3] 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.

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

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

[6] 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.

[7] S. Ranka, Y. Won, and S. Sahni, Programming a hypercube multicomputer, *IEEE Software*, (24)1, September 1988, Pages 69-77.