Fast Compaction in Hypercubes

Nian-Feng Tzeng, *Senior Member*, *IEEE*, and Hsing-Lung Chen, *Member*, *IEEE*

Abstract—Compaction relocates active subcubes in a fragmented hypercube so as to produce a contiguous free region and eliminate the adverse impact of fragmentation on performance. The overhead of compaction is often contributed primarily by task migration, which makes use of disjoint paths for transmitting migrated data. Since task migration usually involves transmitting a large amount of data, the time required for migration with single paths is long, making compaction an undesirably lengthy process. This paper considers fast compaction through the use of *all* disjoint paths in existence for migration simultaneously from a source subcube to its target subcube, effectively reducing the size of data transmitted over a path and shortening the migration time. This approach leads to considerable savings in the compaction time for hypercubes which support circuit switching or wormhole routing, when compared with that using single migration paths.

Index Terms—Compaction, disjoint paths, fragmentation, hypercubes, subcubes, task migration.

1 INTRODUCTION

SUBCUBES in the hypercube are assigned to execute jobs according to a processor allocation scheme, with each job occupying one subcube of an appropriate size. Example processor allocation algorithms can be found in [1], [2], [3], [4], [5]. When a job completes its execution, the occupied subcube is released. After repeated subcube assignments and releases, it is likely for a hypercube to become fragmented so that an incoming job cannot be allocated even if a sufficient number of free nodes are present, because they are scattered around the hypercube. A highly fragmented system may exhibit poor performance and seriously delays the entry of an incoming job, in particular, when the job needs a large subcube. It might be advantageous for a highly fragmented hypercube to perform *compaction*, by moving active tasks to a specified area (for example, lower addressed nodes) such that all free nodes are at a contiguous region ready for upcoming jobs. In this case, compaction rearranges the assignment of subcubes to active tasks in an attempt to remove fragmentation. Compaction may also be useful in applications where hard real-time tasks (which must meet strict timing requirements) coexist with regular tasks. For such an application, hard real-time tasks have higher priority in getting resources and should be admitted to the system immediately. If hard real-time tasks arrive periodically, with their resource requirements known a priori (which is not uncommon) [6], compaction can be invoked to free subcubes of adequate sizes for these tasks before their arrival. In this case, compaction helps to ensure the successful completion of hard real-time tasks.

Compaction requires that all involved tasks be suspended and a new assignment of subcubes to the tasks be generated. The task assignment can be done using a processor allocation scheme, which extracts suspended tasks and puts them back to the front of the waiting queue, treating them as new tasks. These tasks are then assigned by the allocation scheme to subcubes in a contiguous area to avoid fragmentation. Many allocation schemes achieve this goal naturally [2], [4], [5]. The new task assignment tells the target subcube of every suspended subcube, and task migration is carried out to move processes from suspended subcubes to their respective target subcubes. After task migration is completed, the processes in target subcubes resume. The result of compaction in a fragmented hypercube is shown in Fig. 1.

Costs associated with compaction may include:

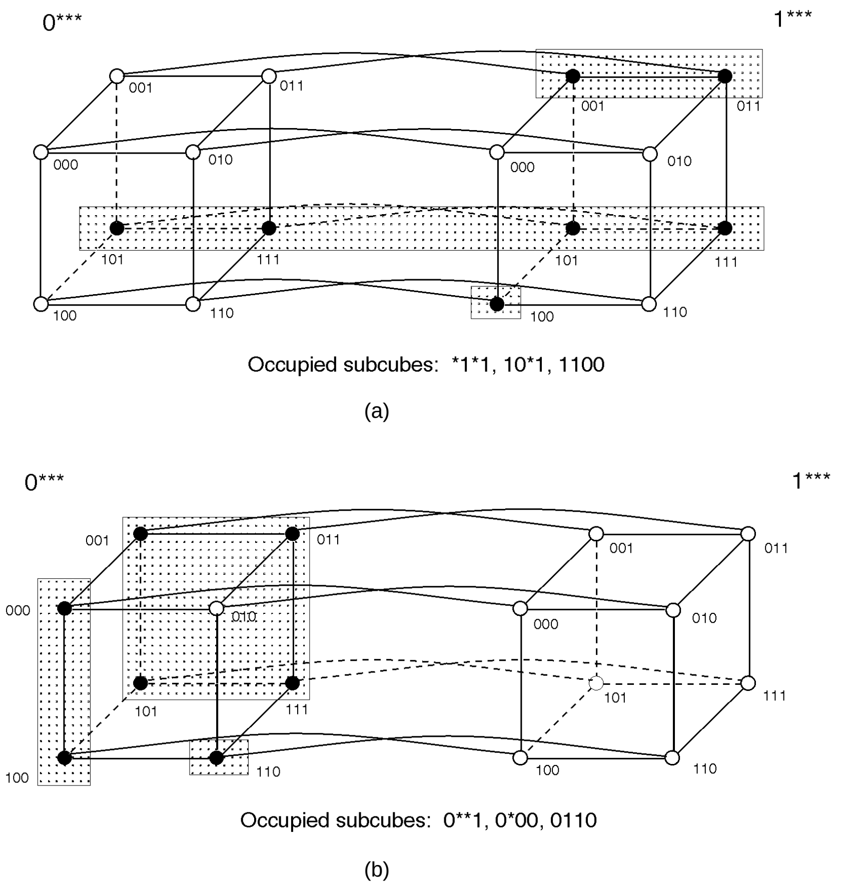
1) the cost of producing a new task assignment,
2) the cost of suspending tasks,
3) the cost of task migration, and
4) the cost of resuming tasks.

The first cost component depends on the complexity of the processor allocation scheme used. The second (or the last) one is due mainly to saving (or restoring) the execution status and registers after a suspending (or resuming) signal is received. Compared with these costs, the cost of task migration is usually dominating [7] because migration involves process movement over the communication channels and the sizes of programs/data to be transmitted are often large. To reduce the compaction duration, the third cost component has to be lowered effectively. It is common that processes in a subcube (which belong to one task) are moved in parallel, as addressed in [1], [8].

Hypercubes of the second generation support either circuit switching (such as Intel's *i*PSC/2 and *i*PSC/860), or wormhole routing (such as NCUBE's *n*-Cube/2) for efficient message transmission. Task migration in a hypercube with circuit switching or wormhole routing requires that the migration paths be totally disjoint from a subcube to its target subcube, because, under circuit switching, this allows a path to be set up successfully between every pair of corresponding nodes, whereas, under wormhole routing, this eliminates message blocking. The communication latency

• *N.-F. Tzeng is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504. E-mail: tzeng@cacs.usl.edu.*
• *H.-L. Chen is with the Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, Republic of China.*

Manuscript received 28 June, 1995.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 104588.



0***
1***
001
011
000
010
101
111
100
110
Occupied subcubes: *1*1, 10*1, 1100
(a)
Occupied subcubes: 0**1, 0*00, 0110
(b)

Fig. 1. (a) Initial task assignment; (b) task assignment after compaction.

under circuit switching and wormhole routing is distance-insensitive if no conflicts occur [9], [10]. It consists of three components: start-up latency, network latency, and blocking time. If migration paths are totally disjoint, the communication latency is contributed only by the first two components. Start-up latency is the time required to handle the message at both the source and destination nodes. Network latency equals the elapsed time after the head of a message has entered the network at the source until the tail of the message emerges from the network at the destination [10], and, under circuit switching, it also includes the time for path setup. Network latency is proportional to the message size, and it tends to dominate for large message transfers, such as those during compaction. To transmit a message of 100K bytes in *i*PSC/2, for example, the network latency accounts for 99 percent of total communication latency (which is roughly 36ms for such a transfer [9]; the start-up latency is only about 0.3ms). Consequently, network latency has to be reduced so as to lower communication latency effectively. This makes it highly desirable to utilize as many disjoint paths as possible for delivering migration data concurrently, reducing the size of data transmitted over a path.

In this paper, we investigate fast compaction by using *all* disjoint migration paths in a hypercube supporting circuit switching or wormhole routing. The paths employed may contain any links because all jobs are suspended during compaction. All nodes of a subcube transmit their data to corresponding nodes in the target subcube at the same time. After a subcube completes its migration, another subcube follows, until all subcubes involved in compaction are exhausted. Utilizing all disjoint paths shortens the migration time to the utmost extent, achieving fast compaction. To implement this compaction, every node is assumed to have a dedicated router permitting *all-port* communication [11], so that multiple messages can be transmitted concurrently from a node over different paths.

2 PRELIMINARIES AND EARLIER WORK

2.1 Nomenclature

Let H_n denote an n-dimensional hypercube, which consists of 2^n nodes, each labeled by an n-bit string, $b_{n-1}b_{n-2}\ldots b_1b_0$, where bit b_i corresponds to dimension i. A link joins two nodes whose labels differ in exactly one bit position. All links are bidirectional and full duplex, i.e., two messages can be transmitted simultaneously in the opposite directions of any link. A link $\lambda(x, y)$ is said to be along dimension i if x and y differ in that dimension. Every path between an arbitrary pair of nodes can be specified uniquely by an ordered sequence of links, and the length of a path is the number of its constituent links. A d-dimensional subcube in H_n is represented by a string of n symbols over set {0, 1, $*$}, where $*$ is a *don't care* symbol, such that there are exactly d $*$'s in the string.

To facilitate subsequent discussion, the following notations are adopted and they are used throughout this paper.

Let N denote the set $\{n-1, ..., 1, 0\}$. For any subcube S and integer $i \in N$, $S[i]$ indicates the ith bit of the address of S. Bit $\overline{b}$ represents the complement of b and is defined only for $b = 0$ or 1. Suppose subcubes S and T are the source and target subcubes, respectively. Let $D(S) = \{i \in N : S[i] = *\}$, $D(T) = \{i \in N : T[i] = *\}$, $I(S,T) = \{i \in N : S[i] = T[i]\}$, and $C(S,T) = \{i \in N : S[i] = \overline{T[i]}\}$, which mean that S and T have don't care bits at $D(S)$ and $D(T)$, respectively; S and T are bitwise identical at $I(S, T)$, and they are complementary to each other at $C(S, T)$. Note that the four sets $I(S, T)$, $C(S, T)$, $D(S) - D(T)$, and $D(T) - D(S)$ are pairwise disjoint, and their union is N, as stated in [8]. Furthermore, $|D(S) - D(T)|$ is always equal to $|D(T) - D(S)|$, which denotes the number of elements in set $D(T) - D(S)$.

2.2 Node Model

Each node in H_n has a dedicated router so that computation and communication can be overlapped at each node. The router supports either circuit switching or wormhole routing. It has multiple external channels for communication among nodes and internal channels for connecting the router with the local processor/memory. Every external channel is assumed to have a corresponding internal channel so that the node can send and receive on all its ports simultaneously, allowing *all-port* communication. All-port communication is realized in the n-Cube/2 hypercube and may become popular for future hypercubes, because it avoids any communication bottleneck between the router and the local processor/memory. With these kinds of routers, each node in H_n can handle up to n simultaneous paths, one along a pair of external input/output channels. Our proposed fast compaction requires source routing, i.e., the entire path of a message is determined and specified by the source node. When $\gamma (\geq 2)$ disjoint paths exist between any given source-destination pair, each of them conveys $1/\gamma$ of total migrated information, effectively reducing the network latency (as it approximates L/B, where L and B are the message length and the channel bandwidth, respectively).

2.3 Earlier Work

Chen and Shin developed a task migration method under Gray code subcube allocation [1]. The method is suitable for hypercubes with store-and-forward switching, requiring that migration be carried out by all participating nodes in locked steps, synchronously. In contrast, the algorithm by Chen and Lai [8] constructs parallel edge-disjoint paths between two subcubes for task migration under circuit switching. With this algorithm, totally disjoint paths are identified between nodes in a subcube and their corresponding nodes in the target subcube, and there is one migration path for each pair of nodes.

An *isomorphism* from S to T is a bijection from nodes of S to nodes of T such that two nodes in S are adjacent if and only if their images in T are adjacent. Chen and Lai defined an isomorphism f from S to T [8] such that, for any node u in S, the address of its image $f(u)$ in T is determined as follows:

$$f(u)[i] = \begin{cases} u[i] & \text{for } all\ i \in I(S,T) \\ T[i] & \text{for } all\ i \in C(S,T) \cup (D(S) - D(T)) \\ u[i] & \text{for } all\ i \in D(T) - D(S) \text{ and } u[\alpha^{-1}(i)] \neq T[\alpha^{-1}(i)] \\ \overline{u[i]} & \text{for } all\ i \in D(T) - D(S) \text{ and } u[\alpha^{-1}(i)] = T[\alpha^{-1}(i)], \end{cases}$$

where α is any arbitrary, but fixed, bijective function from $D(S) - D(T)$ to $D(T) - D(S)$ and α^{-1} is its inverse. This isomorphism yields that the Hamming distance between any node u in S and its image $f(u)$ in T is exactly $|C(S, T)| + |D(S) - D(T)|$. It is clear from this isomorphism that $C(u, f(u))$ is the union of three disjoint subsets: $D_f(S\,|\,u) \cup D_f(T\,|\,u) \cup C(S, T)$, where $D_f(S\,|\,u) = \{i \in D(S) : u[i] \neq f(u)[i]\}$ and $D_f(T\,|\,u) = \{i \in D(T) : u[i] \neq f(u)[i]\}$.

The regular path was defined in [8] as follows: A Hamming path between $u \in S$ and $f(u) \in T$ is *regular* if, on the path from u to $f(u)$, all links along the dimensions in $D_f(T\,|\,u) \cup C(S, T)$ precede all links along the dimensions in $D_f(S\,|\,u)$. It has been proven that *any two regular paths with different source nodes are totally disjoint*. If a regular path from each node u in S to its image $f(u)$ in T is chosen as the migration path, then all the migration paths are pairwise totally disjoint, since they are regular paths with different source nodes.

3 Basis of Migration Paths

This section describes the basis of migration paths for every subcube involved in compaction. The next definition is useful for subsequent description.

DEFINITION 1. *The* least coverage subcube *of S and T is the smallest subcube which contains both S and T, denoted by lcs(S, T).*

3.1 Bound on the Number of Paths

Naturally, it is interesting to find out the upper bound on the number of disjoint migration paths existing from every node in subcube S to its corresponding node in target subcube T. The bound actually depends on whether or not S and T are overlapped, as stated below, and its proof can be found in [12].

LEMMA 1. *Consider d-dimensional subcubes S and T in H_n. If S and T are nonoverlapped, every node in S has at most $(n - d)$ migration paths to reach its corresponding node in T, such that all the paths from S to T are pairwise totally disjoint.*

Next, if S and T are overlapped, let $S \cap T = P$, whose dimension is $d - \delta$, $0 \leq \delta \leq d$. The case of $\delta = 0$ is trivial and is omitted. For $\delta = 1$, it is clear that subcubes $S - P$ and $T - P$ are of the same dimension $d - 1$, where $S - P$ refers to the part in S but outside P. Excluding the overlapped part P, we consider the mapping from $S - P$ to $T - P$ and see how many migration paths exist between them. This is an optimistic scenario, because nodes in P are assumed not to require any migration paths. It is easy to verify that $C(S - P, T - P)$ consists of two elements; in other words, the addresses of $S - P$ and $T - P$ have exactly two complementary bits. Consider $S = *1*1$ and $T = 0**1$ in H_4 illustrated in Fig. 1. In this example, $P = 01*1$, $S - P = 11*1$, and $T - P = 00*1$, so $S - P$ and $T - P$ involve exactly two complementary bits (i.e., b_3 and b_2). Assume that $C(S - P, T - P) = \{i, j\}$. Every node in $S - P$ can obviously have exactly two paths (with one taking dimension i first, followed by dimension j, and the other taking dimension j first, followed by dimension i) within $lcs(S, T)$ to reach its corresponding node in $T - P$ such that all the paths are pairwise totally disjoint. We then examine the possible migration paths outside $lcs(S, T)$. Since the dimension of $lcs(S, T)$ is $d + 1$, each node in $lcs(S, T)$ has $(n - d - 1)$ outgoing links, suggesting that every node in $S - P$ has exactly $n - d - 1$ migration paths (formed by taking dimensions k, i, j, and, then, k in sequence, where k is the dimension of an outgoing link) outside $lcs(S, T)$ to reach its corresponding node, such that all the paths are pairwise totally disjoint. Thus, every node in $S - P$ has exactly $2 + (n - d - 1)$ migration paths, indicating that every node in S has at most $(n - d + 1)$ disjoint migration paths.

For $\delta > 1$, some nodes in S would have fewer disjoint migration paths than what was discussed for the case of $\delta = 1$ above, as described in the following lemma, whose proof is provided in [12].

LEMMA 2. *If the overlapped subcube P of S and T is of dimension $d - \delta$, $1 < \delta \leq d$, then not every node in S can have $(n - d + 1)$ paths to reach its corresponding node in T, such that all the paths are pairwise totally disjoint.*

From the above discussion, it is concluded that the theoretic upper bound on the number of disjoint migration paths from each participating node is $n - d$, unless the dimension of P is $d - 1$, in which $(n - d + 1)$ migration paths from each node in $S - P$ have been identified. The question next is: Can this upper bound be achieved and how can it be achieved? Interestingly, we have derived a systematic approach to achieving this bound. Our approach is based on two types of paths defined below. These definitions require that, for given S and T, every dimension in $C(S, T) \cup (D(T) - D(S))$ be assigned an arbitrary, distinct priority. Once priorities are assigned to dimensions in $D(T) - D(S)$, dimensions in $D(S) - D(T)$ are given priorities as follow: If $k = \alpha^{-1}(j)$, then dimension k is given the same priority as dimension j. Recall that a Hamming path between nodes u and $f(u)$ takes links along dimensions in $C(u, f(u))$, which equals $D_f(S \mid u) \cup D_f(T \mid u) \cup C(S, T)$. Under this priority assignment, no two dimensions in $C(u, f(u))$ are assigned the same priority. This is because, from the definition of isomorphism f for $j = \alpha(k)$, we have $k \in D_f(S \mid u)$ if and only if $j \notin D_f(T \mid u)$.

DEFINITION 2. *For any $j \in D_f(T \mid u) \cup C(S, T)$, a Hamming path between $u \in S$ and $f(u) \in T$ is* definite, *if it takes links along dimensions $\in C(u, f(u))$ in a decreasing priority order, starting with link j, and the link with the lowest dimension priority is followed by the link with the highest dimension priority.*

For example, if links along dimension b in $C(S, T) \cup (D(T) - D(S))$ are assigned priority b, $C(u, f(u)) = \{5, 3, 2, 0\}$ and $j = 2$, then the path formed by taking links along dimensions in the sequence of 2, 0, 5, 3 is a definite path. The path is definite because it is defined once the starting link is known. It is obvious that there are $|D_f(T \mid u) \cup C(S, T)|$ definite paths between u and $f(u)$. The following definition characterizes a path, which is not a Hamming path, between nodes u and $f(u)$.

DEFINITION 3. *For $j = \alpha(k)$ and $u[j] = f(u)[j]$, a path between $u \in S$ and $f(u) \in T$ is* extended *if it takes link j first, then takes links along dimensions $\in C(u, f(u)) - \{k\}$ in a decreasing priority order, then travels through link k and, finally, through link j.*

An extended path starts and ends with links along the same dimension which is not in $C(u, f(u))$. For an extended path, $j = \alpha(k)$ and $u[j] = f(u)[j]$ imply $j \in D(T) - D(S)$ and $u[k] \neq f(u)[k]$, or, equivalently, $k \in D_f(S \mid u)$. Since each k specifies one extended path, there are $|D_f(S \mid u)|$ extended paths in total between u and $f(u)$.

3.2 Properties of Paths

The proofs of the following lemmas are omitted (they can be found in [12]).

LEMMA 3. *Let u be any node in S. Any pair of definite paths between u and $f(u) \in T$ are totally disjoint.*

LEMMA 4. *Let u be any node in S. Any pair of extended paths between u and $f(u) \in T$ are totally disjoint.*

LEMMA 5. *Let u be any node in S. A definite path and any extended path between u and $f(u) \in T$ are totally disjoint.*

LEMMA 6. *Let u and v be any two nodes in S. A definite path from u to $f(u) \in T$ and any definite path from v to $f(v) \in T$ are totally disjoint.*

LEMMA 7. *Let u and v be any two nodes in S. A definite path from u to $f(u) \in T$ and an extended path from v to $f(v) \in T$ are totally disjoint.*

LEMMA 8. *Let u and v be any two nodes in S. An extended path from u to $f(u) \in T$ and any extended path from v to $f(v) \in T$ are totally disjoint.*

4 PROPOSED FAST COMPACTION

Compaction involves four actions: suspending active tasks, producing a new task assignment (without fragmentation) for suspended tasks, then, performing task migration, and, finally, resuming the tasks. Since task migration is often the dominating action among the four [7], we intend to minimize the time overhead of migration, accomplishing fast compaction. To this end, as many disjoint paths as possible are established between a source subcube to its target subcube. The hypercube under consideration (like most contemporary machines) implements circuit switching or wormhole routing, and the communication latency in such a hypercube is insensitive to the path length, provided no conflicts occur. As a result, migration paths can be of unequal length without affecting the communication time, and they may include any links.

Consider migration from d-dimensional subcube S to subcube T in H_n, with $S \cap T$ equal to $\varnothing$ or of dimension $d - \delta$, $1 < \delta \leq d$. (Note that the cases of $\delta \leq 1$ have been treated earlier, in Section 3.1.) The following explains how to determine $(n - d)$ migration paths from every node in S to its corresponding node in T such that all of the paths are pairwise totally disjoint. This is the largest number of migration paths to expect, based on Lemmas 1 and 2. Paths within $lcs(S, T)$ and outside $lcs(S, T)$ are identified in sequence. Let $lcs(S, T)$ be of dimension w.

The address of $lcs(S, T)$ can be represented uniquely by a string of n symbols over set $\{0, 1, *\}$, and w is the number of $*$ in the string. Every symbol i in the string is considered. It belongs to one of the following possibilities:

1) $i \in C(S, T)$,
2) $i \in D(S) - D(T)$,
3) $i \in D(T) - D(S)$,
4) $i \in D(S) \cap D(T)$,
5) $i \in I(S, T) - (D(S) \cap D(T))$.

For possibility 1, either $S[i] = 0$ and $T[i] = 1$, or $S[i] = 1$ and $T[i] = 0$. In either case, $lcs(S, T)[i]$ must equal $*$ so that $lcs(S, T)$ contains both S and T. For possibility 2, $S[i] = *$, indicating that $lcs(S, T)[i]$ must equal $*$, as, otherwise, $lcs(S, T)$ cannot contain S. Similarly, for possibility 3, $T[i] = *$, indicating that $lcs(S, T)[i]$ must equal $*$, as, otherwise, $lcs(S, T)$ cannot contain T. For possibility 4, $S[i] = D[i] = *$, thereby signifying $lcs(S, T)[i] = *$. For possibility 5, however, $lcs(S, T)$ cannot be $*$, since $S[i]$ and $D[i]$ are then both equal to 0 or 1.

The above analysis of symbol i immediately leads to $w = |C(S, T)| + |D(T) - D(S)| + |D(S) - D(T)| + |D(S) \cap D(T)|$, which equals $|C(S, T)| + |D(T) - D(S)| + |D(S)|$. Since $|D(S)|$ is equal to d, the preceding equation implies $|C(S, T)| + |D(T) - D(S)| = w - d$. From Definition 2, there are $|C(S, T)| + |D_f(T | u)|$ definite paths between u and its corresponding node $f(u) \in T$. According to Definition 3 and its subsequent discussion, $|D_f(S | u)|$ extended paths exist between u and $f(u)$. All these definite and extended paths lie inside $lcs(S, T)$, and the total number of these paths amounts to $|C(S, T)| + |D_f(T | u)| + |D_f(S \mid u)|$, which equals $|C(S, T)| + |D(T) - D(S)|$ because α is a bijective function and $|D(S) - D(T)| = |D(T) - D(S)| = |D_f(S | u)| + |D_f(T \mid u)|$. As a result, every node in S has exactly $(w - d)$ migration paths within $lcs(S, T)$. Based on Lemmas 3, 4, 5, 6, 7, and 8, all migration paths selected are pairwise totally disjoint.

Next, migration paths outside $lcs(S, T)$ are determined. For any node u in S, choose an arbitrary definite path (or an arbitrary extended path, if there is no definite path) between u and $f(u)$, say formed by taking dimensions $r_1, r_2, ..., r_z$ in order. Since every node in $lcs(S, T)$ has $(n - w)$ outgoing links (for $lcs(S, T)$ is of dimension w), there are exactly $(n - w)$ paths traversing dimensions $k, r_1, r_2, ..., r_z, k$ from node u to its corresponding node, where dimension k is along an outgoing link of u. In other words, exactly $(n - w)$ paths from u to $f(u)$, one along each outgoing link of u, can be determined. These paths are outside $lcs(S, T)$ and are clearly pairwise totally disjoint. Consequently, every node in S has exactly $(w - d) + (n - w)$ migration paths to reach its corresponding nodes in T such that all the paths are pairwise disjoint. The next theorem follows immediately.

THEOREM. *Consider task migration from d-dimensional subcube S to target subcube T in H_n, with $S \cap T = P$. If $P = \emptyset$ or P is of dimension smaller than $d - 1$, then the number of disjoint paths present between every node in S and its corresponding node in T is $(n - d)$, and this number is optimal (i.e., maximum). In addition, all the paths between S and T are pairwise totally disjoint. If P is of dimension $d - 1$, however, it is possible to identify $(n - d + 1)$ migration paths between every node in S and its corresponding node in T, such that all the paths between S and T are pairwise totally disjoint.*

The following provides an example of migration using all disjoint paths in existence. Let $S = 10{*}1$ and $T = 0{*}00$ in H_4 depicted in Fig. 1. It is apparent that $S \cap T = \emptyset$ and $lcs(S, T) = H_4$, resulting in $n - d = w - d = 3$. Based on the above theorem, three migration paths exist between every node in S and its corresponding node in T. For this task migration, $D(S) - D(T) = \{1\}$, $D(T) - D(S) = \{2\}$, $C(S, T) = \{3, 0\}$, and $\alpha(1) = 2$. As mentioned earlier, dimensions in $C(S, T) \cup (D(T) - D(S))$ have to be assigned different priorities. If dimension b is assigned priority b (any other assignment will do as well), then dimension 3 has priority 3, dimension 0 has priority 0, and dimension 2 has priority 2. In addition, dimension 1, which belongs to $D(S) - D(T)$, is assigned priority 2, identical to that of dimension 2 because $1 = \alpha^{-1}(2)$.

Now, for $u = 1001$, we have $f(u) = 0100$ according to isomorphism f, given in Section 2, yielding $C(u, f(u)) = \{3, 2, 0\}$. Since $D_f(T | u) = \{2\}$ and $C(S, T) = \{3, 0\}$, from Definition 2, there are three definite paths between nodes 1001 and 0100: $1001 \xrightarrow{3} 0001 \xrightarrow{2} 0101 \xrightarrow{0} 0100$, $1001 \xrightarrow{2} 1101 \xrightarrow{0} 1100 \xrightarrow{3} 0100$, and $1001 \xrightarrow{0} 1000 \xrightarrow{3} 0000 \xrightarrow{2} 0100$, where the number above an arrow is the priority of the dimension taken. For instance, the first path takes dimension 3 first, followed by dimension 2, and then dimension 0. In this case, $D_f(S | u) = \emptyset$, signifying that there is no extended path present between the two nodes, as expected. The three identified migration paths are denoted respectively by Y^1, Y^2, and Y^3 in Fig. 2.

Next, for $u = 1011$, we have $f(u) = 0000$ based on isomorphism f, leading to $C(u, f(u)) = \{3, 1, 0\}$. As $D_f(T | u) = \emptyset$ and $C(S, T) = \{3, 0\}$, according to Definition 2, there are two definite paths between nodes 1011 and 0000:

$$1011 \xrightarrow{3} 0011 \xrightarrow{2} 0001 \xrightarrow{0} 0000,$$

and

$$1011 \xrightarrow{0} 1010 \xrightarrow{3} 0010 \xrightarrow{2} 0000,$$

since link 1 is assigned priority 2. In this case, $D_f(S | u) = \{1\}$, indicating that there is one extended path present between the two nodes, based on Definition 3. The extended path starts and ends with links along dimension $j = 2$, which equals $\alpha(1)$, as:

$$1011 \xrightarrow{2} 1111 \xrightarrow{3} 0111 \xrightarrow{0} 0110 \xrightarrow{2} 0100 \xrightarrow{2} 0000,$$

since k is 1 and link 1 is assigned priority 2. Again, there are three migration paths in total between nodes 1011 and 0000, as illustrated in Fig. 2, where they are represented by Z^1, Z^2, and Z^3, respectively. As can be observed, these six migration paths between S and T are pairwise disjoint. The use of all available paths lowers the migration time to the full extent, leading to fast compaction.

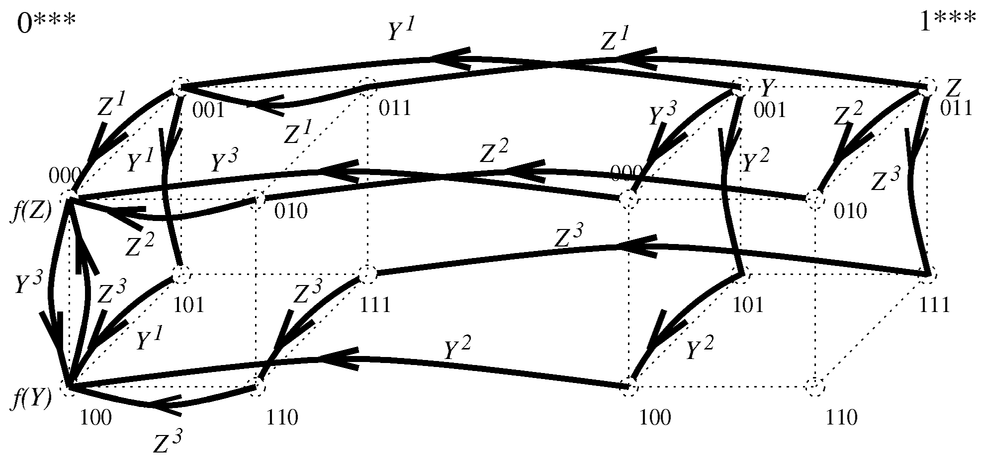


Fig. 2. Three disjoint paths present between every node in $S = 10{*}1$ and its corresponding node in $T = 0{*}00$.

5 Conclusion

We have proposed fast compaction by using *all* disjoint paths present between a subcube and its target subcube for task migration in the hypercube which supports circuit switching or wormhole routing. During compaction, the time spent for task migration often dominates because the amount of information transferred from each involved node is often large. In such a hypercube, the migration time is dictated mainly by network latency, which is proportional to the message size. When γ disjoint paths are utilized for migration, each path then delivers only $1/\gamma$ of total migrated information, shortening the network latency to $1/\gamma$ of that needed with a single path, if the router permits all-port communication. This results from the fact that under circuit switching and wormhole routing, the communication latency is insensitive to path length, so migration paths identified can be of unequal length while maintaining same efficiency.

The method for determining all migration paths is simple and can be carried out at each participating node in a distributed manner. It is applicable to any fragmented hypercube, reducing the migration interval drastically. This makes it possible to achieve compaction much faster than an approach based on single migration paths.

Acknowledgments

Nian-Feng Tzeng was supported in part by the U.S. Natonal Science Foundation under Grants MIP-9201308 and CCR-9300075, and by the State of Louisiana under Contract LEQSF(1994-96)-RD-A-39. Hsing-Lung Chen was supported by the National Science Council of the Republic of China under Contract NSC83-0408-E011-001. A preliminary version of this paper was presented at the Seventh International Conference on Parallel and Distributed Computing Systems, October 1994.

References

[1] M.S. Chen and K.G. Shin, “Subcube Allocation and Task Migration in Hypercube Multiprocessors,” *IEEE Trans. Computers*, vol. 39, no. 9, pp. 1,146-1,155, Sept. 1990.
[2] J. Kim, C.R. Das, and W. Lin, “A Top-Down Processor Allocation Scheme for Hypercube Computers,” *IEEE Trans. Parallel and Distributed Systems*, vol. 2, no. 1, pp. 20-30, Jan. 1991.
[3] P.-J. Chuang and N.-F. Tzeng, “A Fast Recognition-Complete Processor Allocation Strategy for Hypercube Computers,” *IEEE Trans. Computers*, vol. 41, no. 4, pp. 467-479, Apr. 1992.
[4] Q. Yang and H. Wang, “A New Graph Approach to Minimizing Processor Fragmentation in Hypercube Multiprocessors,” *IEEE Trans. Parallel and Distributed Systems*, vol. 4, pp. 1,165-1,171, Oct. 1993.
[5] S. Rai, J.L. Trahan, and T. Smailus, “Processor Allocation in Hypercube Multiprocessors,” *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 6, pp. 606-616, June 1995.
[6] C. Shen, K. Ramamritham, and J. Stankovic, “Resource Reclaiming in Multiprocessor Real-Time Systems,” *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 4, pp. 382-397, Apr. 1993.
[7] C.-H. Huang and J.-Y. Juang, “A Partial Compaction Scheme for Processor Allocation in Hypercube Multiprocessors,” *Proc. 1990 Int’l Conf. Parallel Processing*, vol. I, pp. 211-217, Aug. 1990.
[8] G.-I. Chen and T.-H. Lai, “Constructing Parallel Paths Between Two Subcubes,” *IEEE Trans. Computers*, vol. 41, no. 1, pp. 118-123, Jan. 1992.
[9] O. Frieder et al., “Experimentation with Hypercube Database Engines,” *IEEE Micro*, pp. 42-56, Feb. 1992.
[10] L.M. Ni and P.K. McKinley, “A Survey of Wormhole Routing Techniques in Direct Networks,” *Computer*, vol. 26, no. 2, pp. 62-76, Feb. 1993.
[11] P.K. McKinley and C. Trefftz, “Efficient Broadcast in All-Port Wormhole-Routing Hypercubes,” *Proc. 1993 Int’l Conf. Parallel Processing*, vol. II, pp. 288-291, Aug. 1993.
[12] N.-F. Tzeng and H.-L. Chen, “Fast Compaction in Hypercubes,” Technical Report TR-95-8-4, Center for Advanced Computer Studies, Univ. of Southwestern Louisiana, 1995.

Nian-Feng Tzeng (S'85-M'86-SM'92) received the BS degree in computer science from National Chiao Tung University, Taiwan, the MS degree in electrical engineering from National Taiwan University, Taiwan, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1978, 1980, and 1986, respectively.

He has been with the Center for Advanced Computer Studies at the University of Southwestern Louisiana (USL), Lafayette, since June 1987. From 1986 to 1987, he was a member of the technical staff at AT&T Bell Laboratories, Columbus, Ohio. He is on the editorial board of *IEEE Transactions on Computers*, has served on program committees for several conferences, and was a distinguished visitor of the IEEE Computer Society. He was co-guest editor of a special issue of the *Journal of Parallel and Distributed Computing* on distributed shared memory systems, 1995, and was the newsletter editor for the IEEE Technical Committee on Distributed Processing. His research interests include parallel and distributed processing, high-performance computer systems, high-speed networking, and fault-tolerant computing.

Dr. Tzeng is a member of Tau Beta Pi, a member of the ACM, a senior member of the IEEE, and the recipient of the outstanding paper award of the 10th International Conference on Distributed Computing Systems, May 1990. He received the University of Southwestern Louisiana Foundation Distinguished Professor Award in 1997.

Hsing-Lung Chen (S'79-M'88) received the BS and MS degrees in computer science from National Chiao Tung University, Taiwan, in 1978 and 1980, respectively, and the PhD degree in electrical and computer engineering from the Illinois Institute of Technology, Chicago, in 1987.

From 1987 to 1989, he was an assistant professor in the Department of Mathematics and Computer Science at Clarkson University, Potsdam, New York. In 1989, he joined the Department of Electronic Engineering at Taiwan Institute of Technology, Taipei, Taiwan, where he is currently a professor. His research interests include parallel processing, distributed computing, and database systems.

Dr. Chen is a member of the ACM and the IEEE.