Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Systems Architecture 50 (2004) 177–192
JOURNAL OF SYSTEMS ARCHITECTURE
www.elsevier.com/locate/sysarc

Task migration in *n*-dimensional wormhole-routed mesh multicomputers

Gwo-Jong Yu [a], Chih-Yung Chang [b,*], Tzung-Shi Chen [c]

[a] *Department of Computer and Information Science, Aletheia University, Taipei 251, Taiwan*
[b] *Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei 251, Taiwan*
[c] *Department of Information Management, Chang Jung University, Tainan 396, Taiwan*

Received 3 July 2000; received in revised form 10 January 2003; accepted 19 June 2003

Abstract

In a mesh multicomputer, performing jobs needs to schedule submeshes according to some processor allocation scheme. In order to assign the incoming jobs to a free submesh, a task compaction scheme is needed to generate a larger contiguous free region. The overhead of compaction depends on the efficiency of the task migration scheme. In this paper, two simple task migration schemes are first proposed in *n*-dimensional mesh multicomputers with supporting dimension-ordered wormhole routing in one-port communication model. Then, a hybrid scheme which combines advantages of the two schemes is discussed. Finally, we evaluate the performance of all of these proposed approaches.

Keywords: Dimension-ordered routing; Mesh multicomputers; Gather-and-scatter; Task migration; Wormhole routing

1. Introduction

The simplicity and regularity of mesh topology make it popular in multiprocessor systems. Because it is suitable for VLSI implementation, several mesh-based commercial machines have been built in recent years. Examples includes Intel Touchstone Delta [7], Intel Paragon [8], and the Tera computer system [1]. In a mesh multicomputer, to perform a sequence of jobs needs to schedule submeshes according to some processor allocation strategy, that allocates each job to a submesh with appropriate size. Number of researches aimed at processor allocation and job scheduling schemes in hypercubes [5,9] and mesh multicomputers [2–4,6,10,11,14–19].

Given a job, the mesh system first allocates required processors, then execute job, and finally free processors. Although current submesh allocation schemes try to find an efficient way to allocate processor and to reduce the condition of fragment, however after a lot of processor allocations and deallocations, the mesh system still have a lot of fragments due to the unpredictable of the size and execution time of new incoming jobs. In such condition, a new job cannot be scheduled to execute on this mesh system due to the lack of large enough contiguous area of free processors, even if the number of free processors is sufficient.

* Corresponding author.
E-mail address: cychang@cs.tku.edu.tw (C.-Y. Chang).

1383-7621/$ - see front matter
doi:10.1016/S1383-7621(03)00098-5

So, a task migration which moves running jobs from source processors to target processors is needed for mesh subsystem to increase the system utilization. This work assumes that the locations of source and destination submesh are determined by some processor management scheme of the mesh system. Efficiency of task compaction is measured by the transmission latency of task migration. A lot of researchers design fast task migration schemes in hypercube by exploring disjoint paths between two subcubes. There are some differences between hypercube and mesh computer. The number of parallel paths is proportional to the degree of hypercube. However, the degree of each node is fixed to be $2n$ in an n-dimensional mesh system. In most cases, it is impossible to find out the parallel paths between two submeshes. The objective of this paper is to minimize the transmission latency of task migration by means of transferring the job in a way of several phases.

In recent parallel computer machines, wormhole routing [12,13] is the most important switching technique. In this paper, the target machine we discuss is an n-dimensional mesh multicomputers supporting one-port communication with dimension-ordered wormhole routing. We first present two task migration schemes in n-dimensional mesh multicomputers. Then a hybrid task migration scheme is presented. Finally, we use performance analysis to compare all of the proposed task migration schemes and inspect factors which affect the performance of those proposed schemes.

In the next section, we first introduce the system model of mesh multicomputer and describe the problem to be solved in this paper. In Section 3, two task migration schemes are presented in n-dimensional mesh multicomputers. Next, we proposed a hybrid approach which integrates these two schemes in Section 3.3. Performance analysis of various cases are demonstrated in Section 4. Section 5 includes extensions of the proposed approaches. Section 6 is the conclusion of this paper.

2. Preliminaries

In this section, we introduce our system model for designing the task migration schemes on an n-dimensional mesh multicomputer. The mesh multicomputer system consists of nodes, each node is a computer with its own processor, local memory, and communication links. Each directed link connects to two neighboring nodes through network [12,13]. A common component of nodes in a new generation multicomputer is a router. It can handle and control the message communication entering, leaving, and passing through the node. The architecture of the n-dimensional mesh network system used in this paper is to provide dimension-ordered wormhole routing with one-port communication, in which one node can only send and, simultaneously, receive a worm from the other respective node at the same time. The dimension-ordered routing used in this paper is assumed to route messages to destination nodes first along X_1-direction, then X_2-direction, and finally X_n-direction on the mesh.

An n-dimensional mesh system $M(\mathbf{s})$ consists of $s_1 \times s_2 \times \cdots \times s_n$ number of processors arranged in an n-dimensional grid, where $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ and s_i denotes the number of processors in dimension i. A processor in the grid is denoted by the coordinate $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where $0 \leqslant x_i \leqslant s_i - 1$. Let $M(\mathbf{s})$ denote the set of processors $\{(0,0,\ldots,0), \ldots, (s_1-1, s_2-1, \ldots, s_n-1)\}$. We define the submesh of $M(\mathbf{s})$ as $SM(\mathbf{x}, \mathbf{m})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denotes starting position (the smallest address of processors in submesh in lexicographical order) and $\mathbf{m} = (m_1, m_2, \ldots, m_n)$ denotes the size, $m_1 \times m_2 \times \cdots \times m_n$, of submesh. In other words, $SM(\mathbf{x}, \mathbf{m})$ denotes the set of processors with coordinates in $\{(x_1, x_2, \ldots, x_n), \ldots, (x_1 + m_1, x_2 + m_2, \ldots, x_n + m_n)\}$. Here we denote the source submesh by $SM(\mathbf{x}^s, m) = \{(x_1^s, x_2^s, \ldots, x_n^s), \ldots, (x_1^s + m_1, x_2^s + m_2, \ldots, x_n^s + m_n)\}$ and the destination submesh by $SM'(\mathbf{x}^d, m) = \{(x_1^d, x_2^d, \ldots, x_n^d), \ldots, (x_1^d + m_1, x_2^d + m_2, \ldots, x_n^d + m_n)$. These two submeshes with both of the same shape and size, $m_1 \times m_2 \times \cdots \times m_n$, are located in different starting locations, $\mathbf{x}^s = (x_1^s, x_2^s, \ldots, x_n^s)$ and $\mathbf{x}^d = (x_1^d, x_2^d, \ldots, x_n^d)$ with allowing them to be partially overlapped. One node $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in $SM(x^s, m)$ is needed to route its assigned subtask to the corresponding destination node $\mathbf{x}' = (x_1', x_2', \ldots, x_n')$ in $SM'(x^d, m)$, where

$$x_i' = x_i^d + (x_i - x_i^s), \quad i = 1, \ldots, n. \tag{1}$$

3. Task migration

In this section, two task migration schemes are first proposed in n-dimensional mesh multicomputers based on dimension-ordered wormhole routing. Then, we propose a hybrid scheme to minimize the total routing latency.

3.1. Diagonal scheme

In this subsection, a task migration approach which explores disjoint paths in mesh multicomputer with dimension-ordered wormhole routing is proposed. First, we use the following example of a three-dimensional mesh to illustrate the main idea of the developed task migration scheme. A task with several subtasks allocated to a $5\times4\times3$ submesh is needed to be migrated to another $5\times4\times3$ submesh on a mesh $\{(0,0,0),\ldots,(11,11,11)\}$ as depicted in Fig. 1. The task executed in the source submesh $\{(1,1,1),\ldots,(5,4,3)\}$ is needed to be migrated to the destination submesh $\{(6,7,8),\ldots,(10,10,10)\}$ in a $12\times12\times12$ mesh systems. Due to the maximum dimension of the submesh is five, the proposed scheme uses five phases to migrate the tasks distributed in nodes located in various X_1, X_2, and X_3 axis to the corresponding destination nodes. As shown in Fig. 2, in each phase 12 nodes migrate their subtasks to corresponding destinations in parallel, respectively. The schedule of the migration process depends on

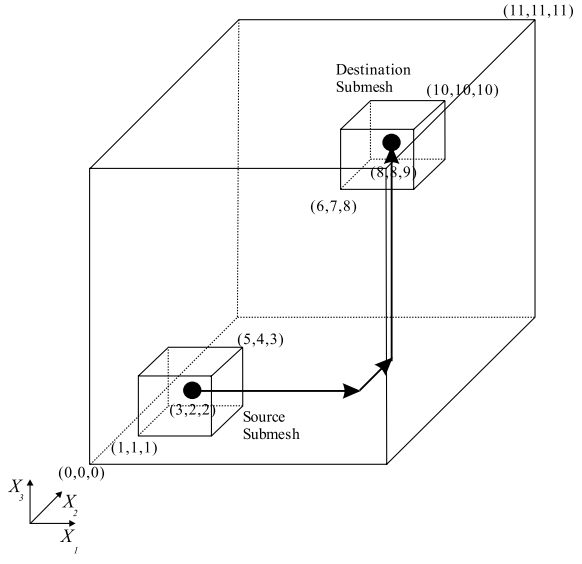

Fig. 1. Task migration between two $5\times4\times3$ submeshes in a three-dimensional mesh with dimension-ordered wormhole routing.

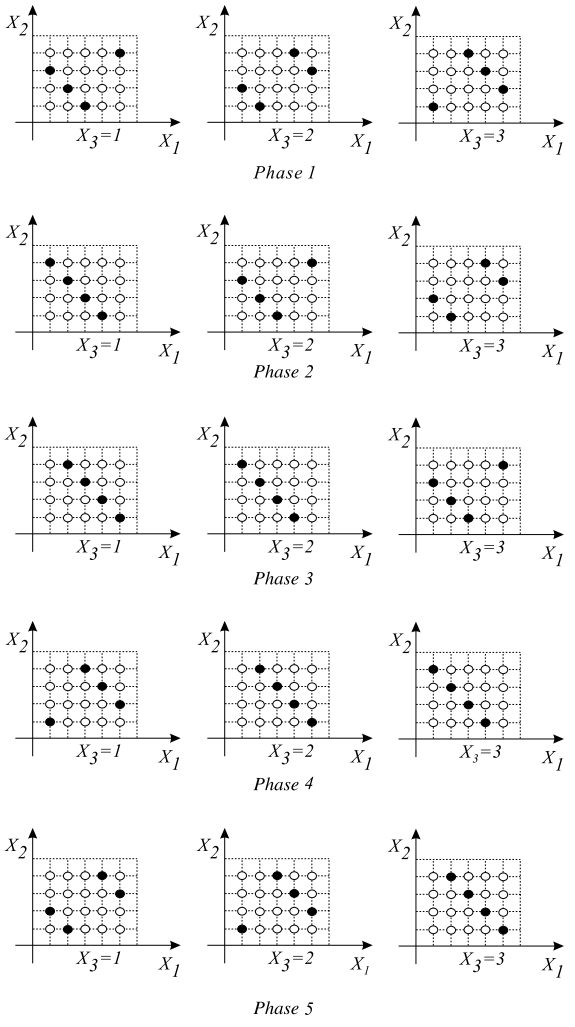

Fig. 2. Schedule of each phase in task migration process using the proposed diagonal scheme. Those solid nodes indicate processors that can be migrated to destination position in a congestion-free manner in each phase.

process coordinates. For example, the coordinates of the 12 nodes selected to migrate the tasks in the first phase are $(3,1,1)$, $(2,2,1)$, $(5,4,1)$, $(2,1,2)$, $(1,2,2)$, $(5,3,2)$, $(4,4,2)$, $(1,1,3)$, $(5,2,3)$, $(4,3,3)$ and $(3,4,3)$, respectively. The rule to choose these nodes is that their coordinates (X_1,X_2,X_3) fulfill

$$(X_1+X_2+X_3)\bmod 5=0.$$

Similarly, the rule to choose nodes in phase k is

$$(X_1+X_2+X_3)\bmod 5=k-1.$$

Note that each node migrates its subtasks first along X_1-direction, then X_2-direction, and finally X_3-direction. As we can check, no common links are used or shared in each phases. For example, in Phase 1 if we project those selected nodes (the solid nodes) to the X_2–X_3 plane, then all nodes will fall in different position. We can conclude that no two nodes will share or use the same link along X_1-direction. Similarly, it is easy to verify that no two nodes will share or use the same link along X_2-direction and X_3-direction. Therefore, in each phase, the routing is congestion-free based on dimension-ordered routing in one-port communication, in which one node can simultaneously send out and receive from one worm at a time. There are totally five phases needed to complete the task migration in this example.

Next, we generalize the above descriptions of the migration routing scheme as follows. In order to avoid congestion during performing dimension-ordered routing in some phase, the nodes in the source submesh should have different positions in $(n-1)$-dimensional subspace when we project a processor from n-dimensional space to $n-1$ subspace along direction X_i for all $1\leqslant i\leqslant n$. In what follows, we propose the *diagonal scheme* to migrate these subtasks. Here we only illustrate how to schedule the source nodes in some transmission phases. It is easy for nodes determine their routing path since the dimension-ordered routing protocol is applied. The source submesh $SM(\mathbf{x}^s,\mathbf{m})$ is assumed to be $\{(x_1^s,x_2^s,\ldots,x_n^s),\ldots,(x_1^s+m_1,x_2^s+m_2,\ldots,x_n^s+m_n)\}$. We state how to arrange the routing phases P_k for each k. The routing during migration is one-by-one from the first phase to the last phase. Let B be set to

$$B=\max_i\{m_i\}.$$

We will show that there are exactly B phases needed to migrate task to destination submesh. For each phase, P_k, all of nodes $\mathbf{x}=(x_1,x_2,\ldots,x_n)$, $x_i^s\leqslant x_i\leqslant x_i^s+m_i$, that are lying on hyperplanes

$$\sum_{i=1}^{n}x_i=c\cdot B+k, \tag{2}$$

are scheduled to simultaneously migrate their subtasks to their corresponding destinations, where c is integers and $1\leqslant k\leqslant B$. Clearly, we have a lot of parallel hyperplanes

$$\sum_{i=1}^{n}x_i=H',$$

where $\sum_{i=1}^{n}x_i^s\leqslant H'\leqslant\sum_{j=1}^{n}(x_j^s+m_j)$. From the above description, the entire task distributed to all nodes in the source submesh is migrated to destination submesh in B phases. We prove that the routing is congestion-free in each phase below. In addition, we prove that the number of routing phases is minimum with congestion-free routing in one-port communication model.

Theorem 1. *By applying the proposed diagonal scheme, the routing in each phase is congestion-free.*

Proof. We will show that in each phase, there is exactly one node occupying the X_1-direction communication channels. By projecting nodes that are scheduled in the same phase to hyperplane $X_1=0$, all nodes will have different positions. So we ensure that all these nodes are congestion-free along X_1-direction. We prove these arguments in what follows. We have hyperplanes

$$\sum_{i=1}^{n}x_i=H$$

for $x_i^s\leqslant x_i\leqslant x_i^s+m_i$, $1\leqslant i\leqslant n$, where

$$\sum_{i=1}^{n}x_i^s\leqslant H\leqslant\sum_{j=1}^{n}(x_i^s+m_i).$$

In phase P_k, nodes in hyperplane

$$\sum_{i=1}^{n}x_i=c\cdot B+k, \tag{3}$$

are scheduled to migrate subtasks to destination, where $B=\max_i\{m_i\}$. Assume $\mathbf{x}^*=(x_1^*,x_2^*,\ldots,x_n^*)$

is a node on the hyperplane in Eq. (3). Projecting $(x_1^*, x_2^*, \ldots, x_n^*)$ onto the hyperplane $x_1 = 0$ generates a corresponding position $(0, x_2^*, \ldots, x_n^*)$. If there are two or more nodes projected to the same location $(0, x_2^*, \ldots, x_n^*)$, these nodes will cause congestion during using X_1-direction communication channels. So, the goal we have to prove is that there are exactly one node in phase P_k projected onto the specific position $(0, x_2^*, \ldots, x_n^*)$. Assume nodes $(x_1, x_2^*, \ldots, x_n^*)$ are projected onto $(0, x_2^*, \ldots, x_n^*)$, then

$$x_1 + \sum_{i=2}^{n} x_i^* = c \cdot B + k.$$

That is,

$$x_1 = c \cdot B + \left(k - \sum_{i=2}^{n} x_i^*\right).$$

Because $x_1^s \leqslant x_1 \leqslant x_1^s + m_1$, the following inequations holds:

$$x_1^s \leqslant c \cdot B + \left(k - \sum_{i=2}^{n} x_i^*\right) < x_1^s + m_1 \tag{4}$$

$$\Rightarrow 0 \leqslant c \cdot B + \left(k - \sum_{i=2}^{n} x_i^* - x_1\right) < m_1 \tag{5}$$

$$\Rightarrow 0 \leqslant c \cdot B + \left(k - \sum_{i=2}^{n} x_i^* - x_1\right) < B. \tag{6}$$

Because $(k - \sum_{i=2}^{n} x_i^* - x_1)$ is a constant, there will be only one integer c satisfies (6). So, for fixed x_i^*, $2 \leqslant i \leqslant n$, only one x_1^* fulfills Eq. (3). If the projected positions of all of these nodes are all different, we can conclude that all of these nodes are congestion-free along X_1-direction. The same arguments can be applied to X_i-direction for $2 \leqslant i \leqslant n$. If all of n directions are congestion-free, we conclude that the migration process is congestion-free. □

Theorem 2. *The number of phases, B, with congestion-free routing is minimum based on dimension-ordered routing in one-port communication model.*

Proof. Based on dimension-ordered routing, there are at most $m_1 \times m_2 \times \cdots \times m_{i-1} \times m_{i+1} \times \cdots \times m_n$ nodes delivering data along x_i-direction simultaneously, for $i = 1, \ldots, n$. For a submesh $SM(\mathbf{x}^s, \mathbf{m})$, therefore, there are at most

$$\min_i \left\{ \prod_{j=1, j\neq i}^{n} m_j \right\}$$

nodes being simultaneously able to route their data in a way of congestion-free transmission. This makes there is a limitation on the size in each dimension of submesh $SM(\mathbf{x}^s, \mathbf{m})$. Therefore, the minimum number of routing phases are

$$\frac{\prod_{i=1}^{n} m_i}{\min_j \left\{ \prod_{k=1, k\neq j}^{n} m_k \right\}},$$

where $\prod_{i=1}^{n} m_i$ is the total number of nodes on the submesh $SM(\mathbf{x}^s, \mathbf{m})$. According to Theorem 1, each phase is congestion-free. We need exactly $B = \max_i \{m_i\}$ phases. Thus, the minimum number of routing phases,

$$\frac{\prod_{i=1}^{n} m_i}{\min\{\prod_{k=1, k\neq j}^{n} m_k\}} = \max_l \{m_l\},$$

whose value is minimum. □

3.2. Gathering–routing–scattering scheme

We next describe our second task migration scheme based on two collective communication schemes, gathering and scattering operations [12]. An illustration of gathering and scattering operation in eight processors is shown in Figs. 3 and 4, respectively, where the solid nodes represent target processor in gathering operation and source processor in scattering operation. Without loss of generality, we assume the source submesh has maximum length in X_1-direction. We apply the gathering operation on nodes in the same X_1-direction to collect subtasks into one node. After the node migrates the combined subtasks to the corresponding destination node, we use the scattering operation on nodes in a line along X_1-direction to disperse a couple of subtasks assigned to the destination node, to their respective nodes in the destination submesh. The gathering and scattering schemes can be referred to [12,16]. In Fig. 3, after the node in the end of line along X_1-direction

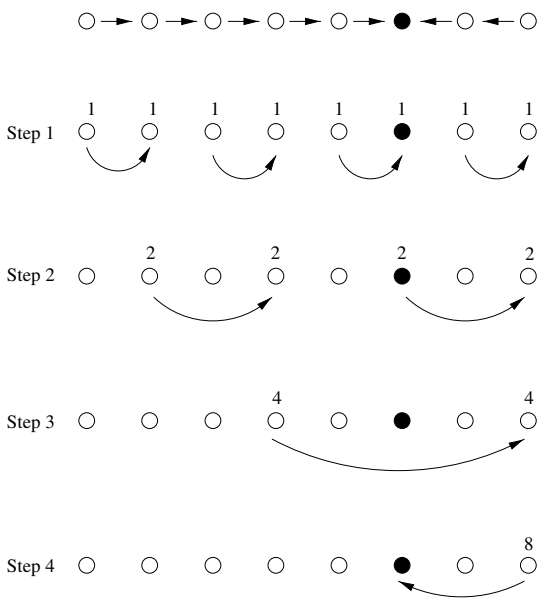

Fig. 3. Gather operation.

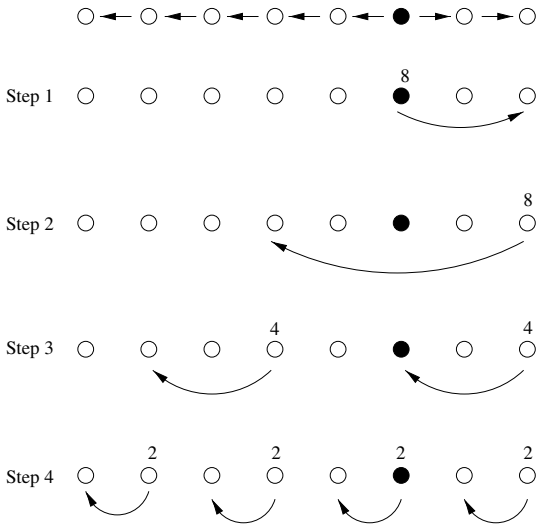

Fig. 4. Scatter operation.

receives all the subtasks, it sends all these subtasks to the scheduled solid node in the source submesh. In the scattering operation, an additional routing step is needed to route the data from solid node to the node in the end of a line along X_1-direction in the destination submesh. Fig. 4 shows the operation of scattering.

Without loss of generality, we assume the submesh has maximum length in X_1-direction. All of nodes $(x_1, x_2, \ldots, x_n)$ located on hyperplanes $\sum_{i=1}^{n} x_i = c \cdot B$ are gathering subtasks distributed within all of nodes in X_1-direction of $(x_1, x_2, \ldots, x_n)$. The node $(x_1, x_2, \ldots, x_n)$ next migrates the combined subtasks to the corresponding node $(x'_1, x'_2, \ldots, x'_n)$. The node $(x'_1, x'_2, \ldots, x'_n)$ finally scatters the combined subtask to the respective destination nodes on lines along X_1-direction to complete the task migration. An illustration of gathering–routing–scattering operation is shown in Fig. 5.

Theorem 3. *The gathering–routing–scattering scheme is congestion-free in each phase.*

Proof. In this approach, all of the gathering and scattering steps are congestion-free [12,16]. The middle step, the solid nodes send the combined subtasks to the corresponding destination nodes, is also congestion-free, which has been shown in Theorem 1. Thus, the gathering–routing–scattering scheme is congestion-free in each phase. □

3.3. Hybrid scheme

In this subsection, we present a hybrid task migration scheme. We combine two schemes stated in the previous subsections to a hybrid one. We first partition the source submesh $SM(\mathbf{x}, \mathbf{m})$ into several n-dimensional subpartitions, which is also of submesh form with the size of $p_1 \times p_2 \times \cdots \times p_n$. That is, the number of subpartitions is $\lceil \frac{m_1}{p_1} \rceil \times \lceil \frac{m_2}{p_2} \rceil \times \cdots \times \lceil \frac{m_n}{p_n} \rceil$, where $\lceil \frac{m_i}{p_i} \rceil$ is the number of partitions in dimension i of the partitioned submesh. In the following, we show how the hybrid scheme works. We first partition the source submesh into subpartitions. We then use the gathering scheme to collect subtasks into nodes, located at the designated positions, depending on the size of m_i, $i = 1, \ldots, n$. This step is similar to the gathering–routing–scattering scheme we proposed. After that, we use the diagonal scheme to schedule $\max(\lceil \frac{m_1}{p_1} \rceil, \lceil \frac{m_2}{p_2} \rceil, \ldots, \lceil \frac{m_n}{p_n} \rceil)$ phases, to route the aggregated subtasks to their corresponding nodes. In this step, each subpartitions is represented as a supernode to transfer their subtasks on

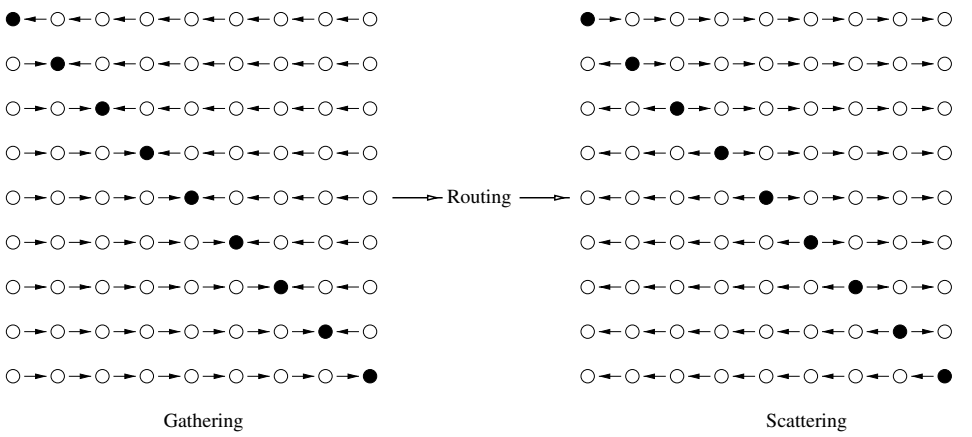

Fig. 5. The gathering–routing–scattering operation.

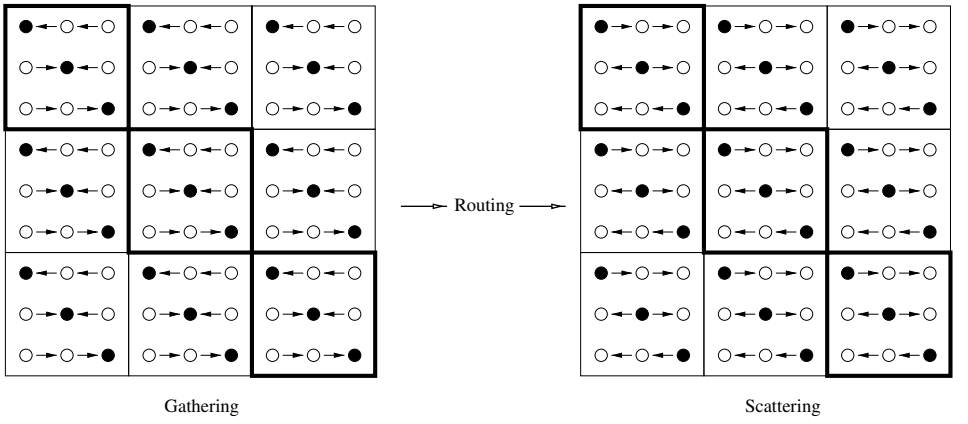

Fig. 6. The hybrid approach.

each node to its corresponding destination subpartition. Finally, we use the scattering scheme to distribute subtasks on the direction, which is with the maximum length in each partition, to complete the task migration. Fig. 6 shows the hybrid approach in a 2D mesh system.

Theorem 4. *The hybrid scheme is congestion-free in each phase.*

Proof. The gathering scheme is first used to collect subtasks in each subpartition on a specific X_i-direction; hence, it is congestion-free. It is also congestion-free that all nodes scheduled in the same phase in each subpartition migrates its combined subtasks to its corresponding node. By Theorems 1 and 3, we know that this step is also congestion-free. The next step is to route the combined subtasks, scheduled with $\max(\lceil \frac{m_1}{p_1} \rceil, \lceil \frac{m_2}{p_2} \rceil, \ldots, \lceil \frac{m_n}{p_n} \rceil)$ phases, to their corresponding nodes. Finally, the scattering scheme is applied to distribute each subtask in each subpartition in X_i direction; hence, it is congestion-free. Therefore, the proposed hybrid scheme is congestion-free in each phase. □

4. Performance analysis

In this section, we will evaluate the developed schemes by performance analysis. Some

parameters used in performance analysis are described in below. We assume that the startup latency is t_s and the transmission time of a flit (or byte) is t_x on a link between two neighboring nodes. Due to the delay in the intermediate routers is small, commercial systems [13] demonstrate that wormhole routing is distance insensitivity. We also assume that the size of subtask distributed in each node is the same, l flits for analyzed convenience. In general, in the wormhole-routing model, the latency for a node sending one message with l flits to another node is $t_s + t_x \times l$ [12,13].

Because the hybrid scheme is a combination of the diagonal scheme and the gathering–routing–scattering scheme, we first discuss the total transmission latency related to the hybrid task migration scheme in below. The hybrid task migration schemes needs gathering, diagonal routing, and scattering steps. In the gathering, it takes $\max_i \{\lceil \log p_i \rceil + 1\}$ time steps to collect these subtasks into one node. The combined message is with a double size compared with the previous one in each step and we need one extra step to transmit the final combined subtasks to the node located at the position in the designated phase. Thus, the size of the collected message is $\max_i \{(l \times 2^{\lceil \log_2 p_i \rceil})\}$ in final. This concludes the gathering step takes the time in below which can be obtained by the above derivations.

We observe that the quantities $\max_i \{p_i\}$ and $\max_j \{\lceil \frac{m_j}{p_j} \rceil\}$ are two important factors of the proposed task migrations schemes. Let $P = \max_i \{p_i\}$ and $Q = \max_j \{\lceil \frac{m_j}{p_j} \rceil\}$. That is, symbol P stands for the maximum number of processors in each dimension of each subpartition (the number of processors collected by gathering and scattering operation). The transmission latency in gathering operation depends on the maximum dimension P as follows:

$$T_{\text{gather}} = (\lceil \log_2 P \rceil + 1) \times t_s + (l \times 2^{\lceil \log_2 P \rceil}) \times t_x.$$

The cost T_{diagonal} depends on both P and Q as follows:

$$T_{\text{diagonal}} = Q \times (t_s + l \times P \times t_x).$$

The cost of T_{scatter} is the same as T_{gather}, i.e.

$$T_{\text{scatter}} = (\lceil \log_2 P \rceil + 1) \times t_s + (l \times 2^{\lceil \log_2 P \rceil}) \times t_x.$$

Thus, we have the total transmission time

$$T_{\text{hybrid}} = T_{\text{gather}} + T_{\text{diagonal}} + T_{\text{scatter}}.$$

We know that the first two schemes proposed in the previous section are the special cases of the hybrid scheme. The first task migration scheme, diagonal scheme, takes the transmission time in below not containing the gather-and-scatter steps for $p_i = 1$, $i = 1, \ldots, n$.

$$T_1 = \max_i \{m_i\} \times (t_s + l \times t_x).$$

The second task migration scheme, gathering–routing–scattering scheme, takes the transmission time in below, for $p_1 = m_1$, $p_2 = m_2$, …, and $p_n = m_n$.

$$T_2 = 2 \times (\max_i \{\lceil \log_2 m_i \rceil + 1\} \times t_s + \max_j \{l \times 2^{\lceil \log_2 m_j \rceil}\} \times t_x) + (t_s + \max_k \{l m_k\} \times t_x).$$

From the above analysis of time complexity, we know that the amount of startup latencies, $\max_i \{m_i\}$, in the diagonal scheme is larger than that of $2 \times (\max_i \{\lceil \log_2 m_i \rceil + 1\}) + 1$ in the gathering–routing–scattering scheme in general. That is the reason why we use the collective communication to reduce the total amount of startup latency in migrating a task. However, the transmission size of the message between two submeshes in the diagonal scheme is generally smaller than that in the gathering–routing–scattering scheme. The hybrid scheme, therefore, is proposed for optimizing the time of migrating one task. To minimize the total amount of transmission delay T_{hybrid}, it is necessary to derive the optimum partitioning with respect to the values of p_i, $i = 1, \ldots, n$ for a specific type of mesh architecture.

Here we give some assumptions to the parameters of system architecture for the use of analyzing the routing performance. This analysis is made on 2D (64×64), 3D ($64 \times 64 \times 64$), or 4D ($64 \times 64 \times 64 \times 64$) submesh accommodating a job needed to be migrated to another location. We have the message startup latency t_s is 1.0 μs for the small startup latency and is 10.0 μs for the large startup latency. The transmission time of a flit t_x on a link is 20.0 ns. The amount of a task in one node is assumed to be l flits; here different values of 100, 300, and 600 flits are discussed.

We first discuss the impact on the partitioning with the same shape and the same size as migrating a job between two 3D ($64\times64\times64$) submeshes. In the first case, we discuss the task migration with the same shape but with different orientations. For convenience, here we assume that the size of subpartition is $2\times16\times32$ with various orientations. The incurred transmission latencies are shown in Fig. 7. Here we observe that the transmission times are the same in all orientations. So, we will focus on various configurations (sharps) in the following analysis. However, if the length of all dimensions is different, this argument no more holds.

According to formula of transmission latency in each step of hybrid approach, to minimize the total transmission latency, T_{hybrid}, both of P and Q must be as small as possible. In the second case, we perform the task migration using various configurations such that $P = \max_i \{p_i\}$ is set to 64 and $Q = \max_j \{\lceil \frac{m_j}{p_j} \rceil\}$ has various size. Because Q corresponds to the number of phases needed in diagonal routing step. When P is fixed, small Q is corresponding to small total latency. The simulation results is shown in Fig. 8. From Fig. 8, the configuration $8\times8\times64$ corresponding to $P = 64$ and $Q = 8$ has the smallest transmission latency.

In the third case, we perform task migration in 3D submesh using various configuration such that all configuration corresponds to the same value of $Q = 64$ and various value of P. The value of P corresponds to the message size and the step needed in the gathering and scattering operation. To reduce transmission latency, the value of P is as small as better. It can be seen from Fig. 9 that small P corresponding to small transmission latency.

In the fourth case, we discuss the impact of uniform partitioning size using large and small startup latency on the transmission latency in 3D mesh system, i.e. we have different values of $p_1 \times p_2 \times p_3$. In order to balance the number of routing phases, P, and the maximum dimension of subpartition, Q, the subpartitions are with the shapes of the size of $1\times1\times1$, $2\times2\times2$, $4\times4\times4$, $8\times8\times8$, $16\times16\times16$, $32\times32\times32$ and $64\times64\times64$ used in this case of performance analysis. The configurations, $1\times1\times1$, corresponds to the diagonal scheme while the configuration, $64\times64\times64$, corresponds to the gathering–routing–scattering

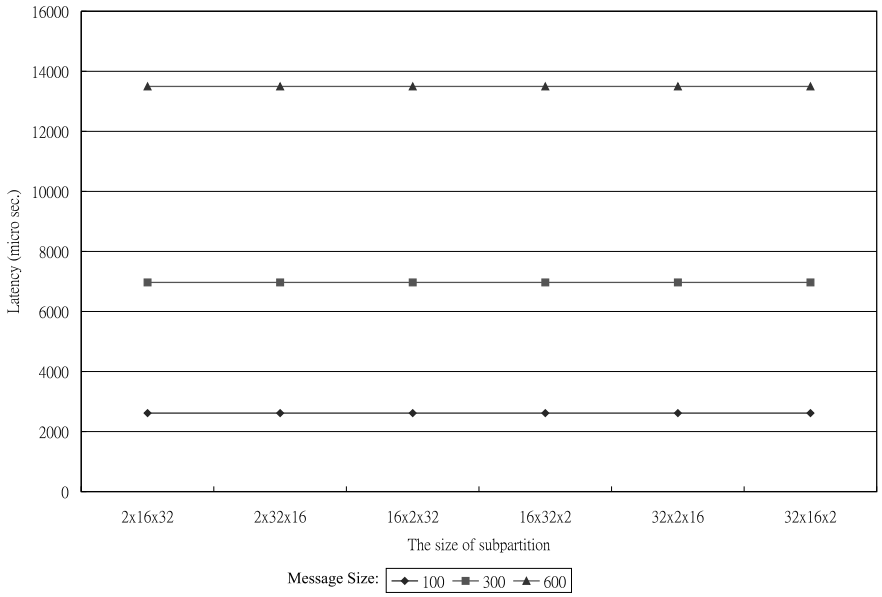

Fig. 7. Transmission latency of three-dimensional mesh system with large startup latency. Each subpartition is with the same shape but with different orientation.

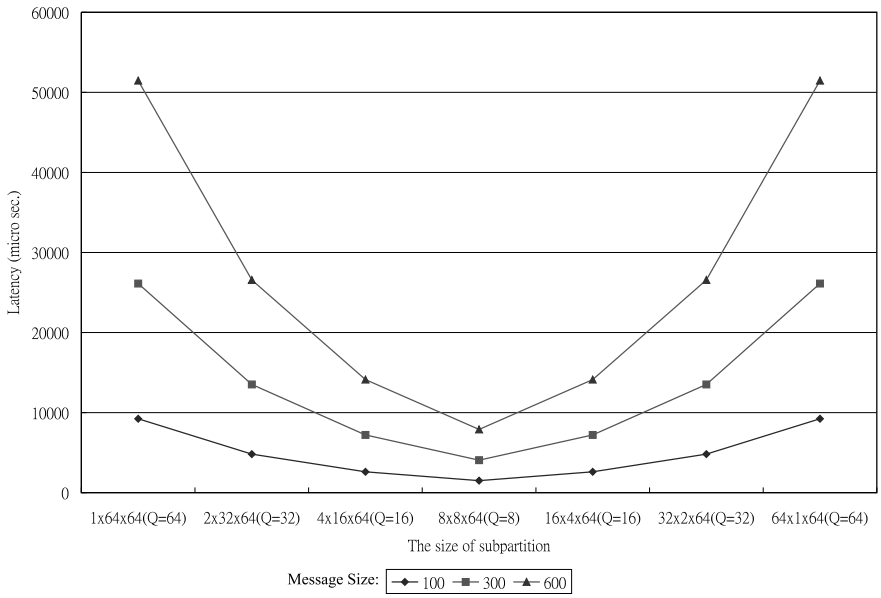

Fig. 8. Transmission latency of 3D mesh with large startup latency. Configurations have the same value of $P = 64$ and various Q.

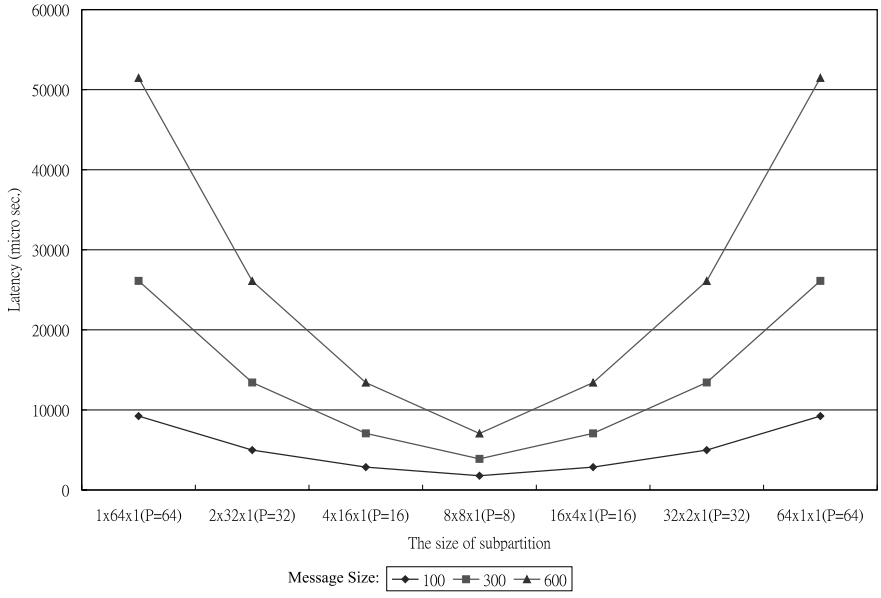

Fig. 9. Transmission latency of 3D mesh with large startup latency. Configurations have the same value of Q but with different value of P.

scheme. Figs. 10 and 11 are the incurred transmission latency of small and large startup latency, respectively. In small startup latency, as shown in Fig. 10, we prefer to use the diagonal scheme for task migration in most case. This is because in small startup latency, the impact of message size is

larger than the startup latency. In small startup latency, and message size 100, 300, and 600, diagonal scheme requires smaller transmission latency than gathering–routing–scattering scheme. On the other hand, in large startup latency, when the message size is large ($l = 600$), diagonal scheme performs better than gathering–routing–scattering scheme. However, when the message

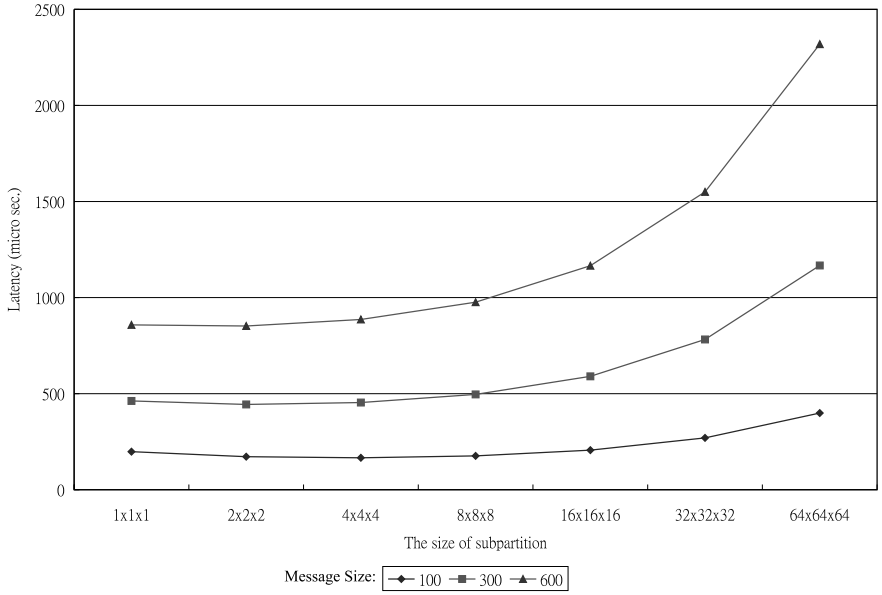

Fig. 10. Transmission latency in 3D mesh system with small startup latency. Each subpartition has various block size.

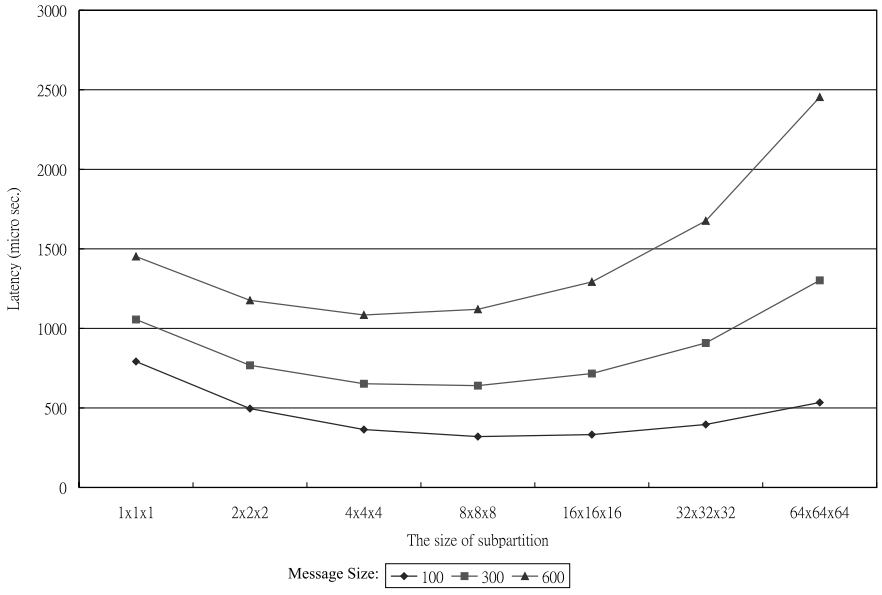

Fig. 11. Transmission latency for large startup latency in 3D mesh with various subpartition size.

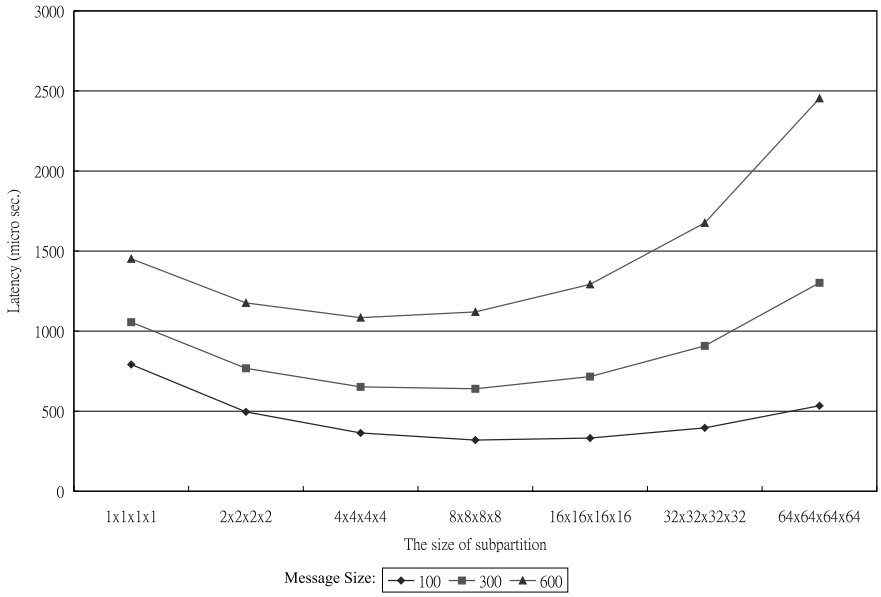

Fig. 12. Transmission latency for large startup latency with in 4D mesh with various partition size.

size is small ($l = 100$), gathering–routing–scattering scheme performs better. In Fig. 11, the configuration $4\times4\times4$ perform best when message size is 300 or 600. The configuration $8\times8\times8$ perform best in the case that the message size is 100.

To examine the system performance on various dimensions, we compute the transmission latency on 4D and 2D mesh system. In 4D mesh system, the size of source submesh is $64\times64\times64\times64$ and the size of subpartition ranges from $1\times1\times1\times1$ to $64\times64\times64\times64$. Fig. 12 shows the performance of various configurations. Compare Figs. 11 and 12 , we found that all configurations have the same transmission latency. This is because the wormhole routing is distance insensitive and the formula of T_{hybrid} is affected by P and Q, and is irrelevant to dimension. Fig. 13 shows the transmission latency of the hybrid approach on source submesh with size 64×64. We obtain the same results as in Figs. 11 and 12.

To give a precise illustration of the proposed approach, we have constructed a simulator of multidimensional mesh multicomputers for empirical analysis of the task migration process. This simulator can be used to estimate the difference between the theoretical and empirical results. The parameters used in theoretical analysis are also used in empirical analysis for the sake of fair comparison. The empirical simulation results of transmission latency with 4D uniform shaped submesh with large startup latency are shown in Fig. 14. Through the comparison of Figs. 12 and 14, we observed that the tendency of theoretical results coincide with empirical results with minor difference. The small numerical difference comes from the extra communication steps in the beginning and ending of the pipeline transmission processes. In theoretical analysis, the extra steps in the beginning and ending of pipelining are omitted for clarity. We can thus conclude that the theoretical analysis is reasonable to the description of performance analysis.

Briefly, from the above analysis, we have the following comments and suggestions as performing task migration. Generally speaking, we use the diagonal scheme to perform task migration to gain the minimized transmission latency when the startup latency is smaller. The time by applying the second scheme is usually larger than that by applying other schemes as shown in Fig. 6. Thus, the second one is more unsuitable for executing task migration. The reason is that the collective com-

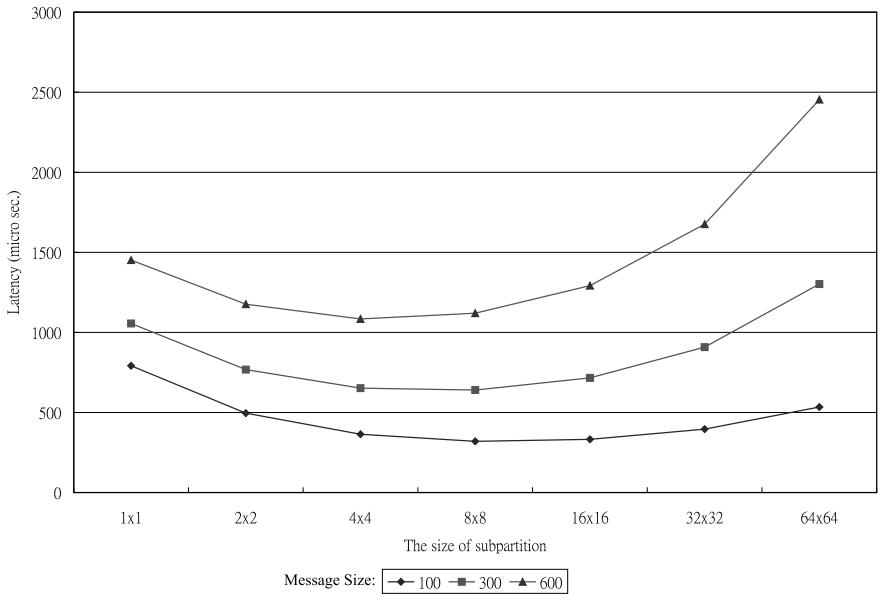

Fig. 13. Transmission latency for large startup latency in 2D mesh with various partition size using hybrid approach.

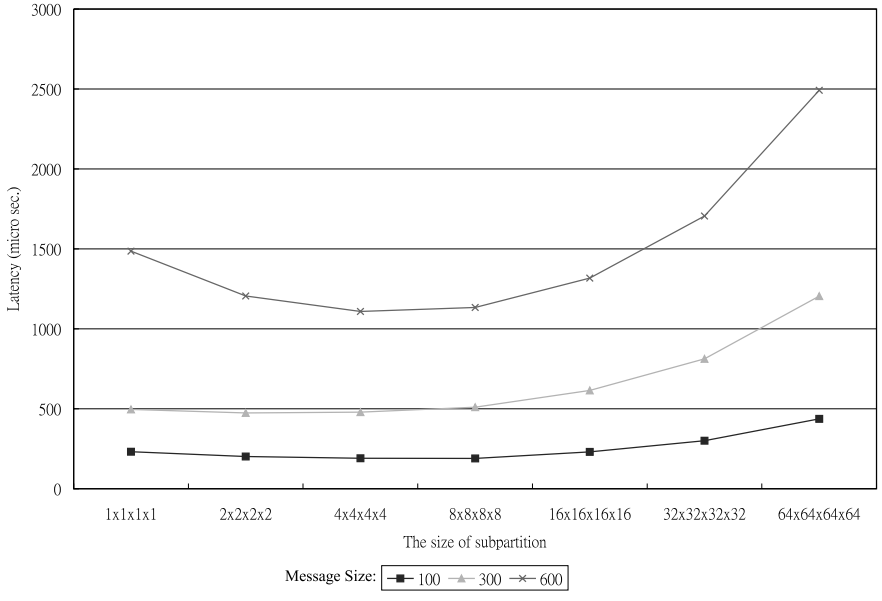

Fig. 14. Transmission latency for large startup latency in 4D mesh with various partition size obtained from simulator.

munication used in each X_i-direction takes a lot of time to collect a large amount of messages. However, we have to take into account the factors which affect the transmission latency, including the message size assigned in each node, the startup latency, as well as the partitioning size and shape together as the hybrid scheme is used. By evaluating these factors together, we are able to get the optimum solution with having the minimum time of T_{hybrid}.

5. Discussions

In this section, we take into account the cases as the noncontiguous processor allocation schemes [11] are used. We classify three cases to discuss the situations of migrating tasks as in the mesh system that allows the noncontiguous processor allocation. In the following discussions, assume that the number of processors are the same as that in the destination processors.

Case 1: A task is assigned to one submesh and then is needed to be reassigned to a couple of subframes which are of the cuboid form. In order to migrate the task into these subframes, we first partition the source submesh into pieces of subframes with the same size as corresponding to the destination subframes. Thus, we apply our proposed schemes to performing the task migration on each pair of subframes.

Case 2: A task is assigned to a couple of subframes and then is needed to be reassigned to one submesh. In order to migrate the task distributed at these subframes to one destination submesh, we first partition the destination submesh into pieces of subframes with the same size as corresponding to a couple of source subframes. Thus, we apply our proposed schemes to perform the task migration on each pair of subframes.

Case 3: A task is assigned to a couple of subframes and then is needed to be reassigned to another couple of subframes. In order to migrate the task distributed at these subframes to their corresponding destinations, we first partition each subframe in source into a few pieces of subpartitions with the size according to these distributed subtasks which are reallocated to the places in destination. Next, we apply the approach described in Case 1 to performing the task migration for each pair of subpartitions.

We demonstrate the schemes of the above three cases while a mesh system supports noncontiguous processor allocation schemes with three examples as shown in Fig. 15(a)–(c), respectively. As shown in Fig. 15(a), we have a task assigned to a source submesh that is needed to be migrated to three subframes. The source submesh is partitioned into three subframes with the same size with respect to the destination subframes. Then, we apply our proposed scheme to each pair of one destination submesh. The destination submesh is partitioned into three subframes with the same sizes with respect to the source subframes. Then, we apply our proposed scheme to each pair of subframes. Finally, we discuss the complex case as shown in Fig. 15(c). Prior to applying task migration, we examine each destination subframe and determine its location of the corresponding destinations at destination subframes. For example, one source subframes is partitioned to three subpartitions 1, 2, and 3, numbered, because of their corresponding destination subpartitions are located at different

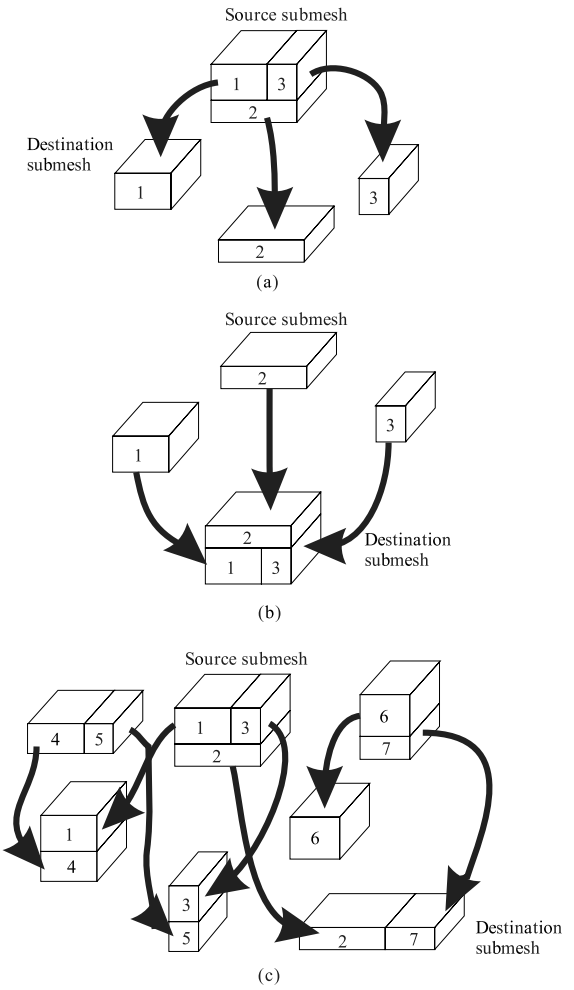

Fig. 15. Three examples for cases 1, 2 and 3 for noncontiguous processor allocation schemes: (a) an example of case 1; (b) an example of case 2; (c) an example of case 3.

place as depicted in Fig. 15(c). Then, we apply our proposed scheme to each pair of subpartitions.

Clearly, we can easily use our proposed schemes to perform task migration on each pair of subframes for each case. In each case, however, there exist a lot of pairs of subframes needed to be executed to migrate their subtasks. It may increase the traffic congestion while task migration is executed. Certainly, it guarantees that the routing by our proposed schemes is deadlock-free by using the dimension-ordered wormhole routing.

As in the above discussions, we can easily apply our proposed schemes to a variety of cases, such as the destination submesh with rotated shape and noncontiguous source and destination submeshes. Consequently, our proposed task migration schemes are available for various processor allocation schemes, including contiguous and noncontiguous processor allocation approaches.

6. Conclusion

In this paper, two simple task migration schemes are proposed in n-dimensional mesh multicomputers with supporting dimension-ordered wormhole routing in one-port communication model. Furthermore, a hybrid task migration scheme was proposed with the attempt to minimizing the routing latency. Finally, we compare all of our proposed task migrations schemes via performance analysis. In addition, we discuss how to easily apply our proposed task allocation schemes to some different processor allocation schemes with contiguous methods. Our future work includes investigating the job scheduling approaches, integrating task migration and processor allocation together, and evaluating the entire system performance including system utilization, job response time and more.

Acknowledgements

This work was supported by the National Science Council of the Republic of China under grant NSC 91-2219-032-006, NSC 91-2213-E-032-036 and NSC 91-2213-E-156-002.

References

[1] R. Alverson, The Tera computer system, in: Proceedings of International Conference on Supercomputing, 1990, pp. 1–6.
[2] S. Bhattacharya, W.-T. Tsai, Lookahead processor allocation in mesh-connected massively parallel multicomputer, in: Proceedings of International Parallel Processing Symposium, 1994, pp. 868–875.
[3] G.-M. Chiu, S.-K. Chen, An efficient submesh allocation scheme for two-dimensional meshes with little overhead, IEEE Transactions on Parallel and Distributed Systems 10 (May) (1999) 471–486.
[4] P.-J. Chuang, N.-F. Tzeng, An efficient submesh allocation strategy for mesh computer systems, in: Proceedings of 11th International Conference on Distributed Computing Systems, May 1991, pp. 256–262.
[5] P.-J. Chuang, N.-F. Tzeng, A fast recognition-complete processor allocation strategy for hypercube computers, IEEE Transactions on Computers 41 (4) (1992) 467–479.
[6] J. Ding, L. Bhuyan, An adaptive submesh allocation strategy for two-dimensional mesh connected systems, in: Proceedings of International Conference on Parallel Processing, vol. II, 1993, pp. 193–200.
[7] I. Corp., Paragon XP/S product overview, 1994.
[8] I. Corp., A touchstone DELTA system description, 1991.
[9] J. Kim, C.R. Das, W. Lin, A top–down processor allocation scheme for hypercube computers, IEEE Transactions on Parallel and Distributed Systems 2 (January) (1991) 20–30.
[10] K. Li, K.H. Cheng, A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system, Journal of Parallel and Distributed Computing 31 (December) (1991) 79–83.
[11] V. Lo, K.J. Windisch, W. Liu, B. Nitzberg, Noncontiguous processor allocation algorithms for mesh-connected multicomputers, IEEE Transactions on Parallel and Distributed Systems 8 (July) (1997) 712–726.
[12] P.K. Mckinley, Y.-J. Tsai, D.F. Robinson, Collective communication in wormhole-routed massively parallel computers, IEEE Computers 28 (December) (1995) 39–50.
[13] L.M. Ni, P.K. Mckinley, A survey of wormhole routing techniques in direct networks, IEEE Computers 26 (February) (1993) 62–76.
[14] D.D. Sharma, D.K. Pradhan, Submesh allocation in mesh multicomputers using busy-list: a best-fit approach with complete recognition capability, Journal of Parallel and Distributed Systems 36 (January) (1998) 106–118.
[15] Y.-J. Suh, S. Yalamanchili, Configurable algorithms for complete exchange in 2D meshes, IEEE Transactions on Parallel and Distributed Systems 11 (April) (2000) 337–356.
[16] Y.-C. Tseng, S.-Y. Ni, J.-P. Sheu, Toward optimal complete exchange on wormhole-routed tori, in: Proceedings of the 1997 International Conference on Parallel and Distributed Systems, Seoul, Korea, December 1997, pp. 96–103.

[17] F. Wu, C.-C. Hsu, L. Chou, Processor allocation in the mesh multiprocessors using the leapfrog method, IEEE Transactions on Parallel and Distributed Systems 14 (March) (2003) 276–289.
[18] B.S. Yoo, C.R. Das, A fast and efficient processor allocation scheme for mesh-connected multicomputers, IEEE Transactions on Computers 51 (January) (2002) 46–60.
[19] Y. Zhu, Efficient processor allocation strategies for mesh-connected parallel computers, Journal of Parallel and Distributed Computing 16 (December) (1992) 328–337.

Gwo-Jong Yu received the B.S. degree in information computer engineering from the Chung-Yuan Christian University, Chung-Li, Taiwan in 1989, and the Ph.D. degree in computer science from National Central University, Chung-Li, Taiwan in 2001. From July 2000 to June 2001, he was also a research assistant in the Institute of Information Science, Academia Sinica, Taipei, Taiwan. Since August 2001, he has been an assistant professor in the Department of Computer and Information Science, Aletheia University, Tamsui, Taipei, Taiwan. His research interests include digital watermarking, parallel and distributed processing and wireless communication. Dr. Yu is a member of IEEE Communication Society.

Chih-Yung Chang received the Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in 1995. He joined the faculty of the Department of Computer and Information Science at Aletheia University, Taiwan, as an assistant professor in 1997. He was the Chair of the Department of Computer and Information Science, Aletheia University, from August 2000 to July 2002. He is currently an associate professor of Department of Computer Science and Information Engineering at Tamkang University, Taiwan. Dr. Chang is a member of Editorial Board of *Tamsui Oxford Journal of Mathematical Sciences* and a member of the IEEE Computer Society and IEICE society. His current research interests include Bluetooth radio systems, Ad Hoc wireless networks, sensor networks, personal communication systems, interconnection networks and mobile computing.

Tzung-Shi Chen received the B.S. degree in Computer Science and Information Engineering from Tamkang University, Taiwan, in June 1989 and the Ph.D. degree in Computer Science and Information Engineering from National Central University, Taiwan, in June 1994. He joined the faculty of the Department of Information Management, Chung Jung University, Tainan, Taiwan, as an Associate Professor in June 1996. Since November 2002, he has become a Professor at the Department of Information Management, Chung Jung University, Tainan, Taiwan. He was a visiting scholar at the Department of Computer Science, University of Illinois at Urbana-Champaign, USA, from June to September 2001. He is currently the Chair of Department of Information Management at Chung Jung University. He co-received the best paper award of 2001 IEEE ICOIN-15. His current research interests include mobile computing and wireless networks, Internet computing, web mining, and parallel and distributed computing. Dr. Chen is a member of the IEEE Computer Society.