
ELSEVIER

Parallel Computing 24 (1998) 1499–1522

PARALLEL COMPUTING

An efficient four-connected parallel system for PET image reconstruction

Chung-Ming Chen [1]

Center for Biomedical Engineering, College of Medicine, National Taiwan University, No. 1, Sec. 1, Jen-Ai Road, Taipei, Taiwan

Received 15 January 1998; received in revised form 15 April 1998

Abstract

In this paper, we present an efficient parallel system with an interconnection network customized for Positron Emission Tomography (PET) image reconstruction. The proposed parallel reconstruction system has two distinguished features. On feature is that the interconnection network is optimal for *both* filtered backprojection and EM algorithms, rather than only for one of them. The other feature is that with only four-connectivity in contrast to log N-connectivity for a hypercube, the proposed parallel algorithms may accomplish the same performance in terms of order statistics as achieved by the optimal algorithms on a hypercube. The proposed parallel system has been realized using transputers.

Keywords: Positron emission tomography; Parallel image reconstruction; Filtered-backprojection algorithm; EM algorithm; Perfect shuffle network; Mesh; Hypercube

1. Introduction

Positron Emission Tomography (PET) is an imaging modality giving distribution of positron-emitting isotope-labeled chemicals in the human body. Unlike X-ray CT and MRI which provide anatomical data, PET reveals functional information on in vivo physiology and metabolism of the human body. Clinically, early detection of a disease before morphologically distinguishable may be achieved through PET by studying physiological or metabolic disorders. Hence, PET has become one of the most important imaging tools in modern diagnosis. However,

[1] E-mail: chung@lotus.mc.ntu.edu.tw

0167-8191/98/$ – see front matter
PII: S0167-8191(98)00068-4

even though PET may offer information not attainable by other medical imaging modalities, it requires much more reconstruction time for obtaining a high-quality 3D PET image. It is because a *high-quality* 3D PET image may not be reconstructed by simply stacking slices of 2D PET image as X-ray CT and MRI do due to its 3D nature. This can be easily understood by imagining that each pair of photons produced by positron–electron annihilation may fly in any direction in the 3D space. If only transverse planes are utilized, clearly, most photons would be wasted. Therefore, if a high-quality 3D PET image is desired, a true 3D reconstruction utilizing all available projection data needs to be employed, which undoubtedly would be far more computationally intensive than reconstruction of 3D X-ray CT or MRI.

To reconstruct a PET image in a reasonable time, various efforts have been made in the past to speed up both types of reconstruction algorithms, namely, analytic and iterative algorithms. Some of these works were based on special hardwares [21,38] to attain the required computation speed. Others took advantages of supercomputing [25] or parallel processing [4–11,22–24,30,33–36] to gain high computing power. For those works with special hardwares, two examples are the fractional address accumulator designed by Thompson et al. [38] using digital circuits and the slice-back-project engine proposed by Hartz et al. [21] making use of bit-slice technology. Although special hardwares could possibly meet the computational requirement of a particular PET system, these designs either have a limited obtainable speedup or are not generally applicable and extendible to other PET systems.

On the other hand, supercomputing and parallel processing both promise a fast and scalable reconstruction, e.g., it was shown in [25] that a 128×128 2D PET image can be reconstructed in several seconds on a Cray computer even using an EM algorithm. However, the high cost/performance ratio on a supercomputer is not preferable for most implementations.

In contrast to supercomputing, parallel processing offers more degrees of freedom, such as VLSI technology, interconnection networks, routing algorithms, task partitioning and so on, in achieving a low-cost high-performance implementation. Two classes of parallel implementations for PET image reconstruction may be found previously. One is taking advantage of general-purpose parallel systems [5–11, 19,33], such as Intel iPSC/2, iPSC/860, i860-based, workstation cluster, etc., which are commercially available. The other is resorting to dedicated parallel architectures, e.g., employing array processors [1], building a system based on special VLSI-based chips [23,24], transputers [2,4,35], DSP processors [34], and general processors [13–16], etc. Using a general-purpose parallel system, in general, has the merits of flexibility and easy implementation. However, the cost/performance might not be optimal even if the optimal parallelization efficiency is achieved on such a general-purpose parallel system. On the other hand, employing a dedicated parallel system may expect a high cost/performance since a customized hardware, e.g., processing units and interconnection network, may be used to minimize redundant computation and data sharing overhead. Moreover, the redundant hardware may be eliminated to reduce system cost. But, it generally requires an elaborate design to achieve a cost-effective system.

While both classes of implementations have their own merits and demerits, in our effort to support fast PET image reconstruction for the PET center in National Taiwan University, we have been interested in a dedicated parallel reconstruction system considering the cost-effectiveness in a clinical environment. Two classes of PET reconstruction algorithms are under considerations. The first class is the filtered-backprojection algorithm, which is an analytic algorithm. The filtered-backprojection algorithm is widely used in commercial PET system since it provides a faster solution than its counterpart. Various filtered-backprojection algorithms have been proposed previously [12,36], especially for 3D PET image reconstruction. However, a filtered-backprojection algorithm is usually composed of two stages, i.e., filtering followed by backprojection. The second class is the EM algorithm, which is an iterative method. The EM algorithm is interested for it generally promises a better reconstructed image. It is because more detailed physical processes have been modeled in the EM algorithm. But, it is much slower than the filter-backprojection algorithm. The EM algorithm was originally proposed by Shepp and Vardi [37] based on maximum likelihood estimates. Although there are many variations of the EM algorithms, most of them have the similar algorithmic structure as the Shepp and Vardi's EM algorithm. These EM algorithms usually include the Shepp and Vardi's EM algorithm and some additional steps either to speed up convergency [29] or to attain better quality of images [18,20,28].

In this paper, we propose an efficient parallel system with an interconnection network customized for the PET image reconstruction algorithms under consideration. The interconnection network is the static (i.e., consisting of point-to-point communication links among PEs as defined in [27]) perfect shuffle interconnection network. Compared to most previous dedicated systems, the proposed parallel reconstruction system has two distinguished features. On feature is that the interconnection network is optimal for *both* filtered backprojection algorithm and EM algorithm, rather than only for one of them. The other feature is that with only four-connectivity in contrast to log N-connectivity for a hypercube, the proposed parallel algorithms may accomplish the same performance in terms of order statistics as achieved by the optimal algorithms on a hypercube.

This paper is organized as follows. The algorithmic models of the PET image reconstruction algorithms are first defined in Section 2. For ease of presentation, the data sharing modes involved in both classes of parallel reconstruction algorithms are defined in Section 3. In Section 4, the dedicated parallel architecture is proposed along with the optimal data sharing algorithms for the four underlying data sharing modes. In Section 5, the parallel filtered-backprojection algorithm and its implementation results are presented. In Section 6, the parallel EM algorithm and its implementation results are provided. Conclusions are given in Section 7.

2. Algorithmic models of the PET image reconstruction algorithms

There have been many filtered-backprojection and EM algorithms proposed previously. To define the application scope of the proposed parallelization schemes,

the algorithmic models for the two classes of algorithms, i.e., the filtered-backprojection algorithm and the EM algorithm, are described in this section.

2.1. The filtered-backprojection algorithm

The filtered-backprojection algorithm considered in our system is composed of two stages, namely, filtering and backprojection. In the filtering stage, the projection data, i.e., the measured event counts in a PET system, are convolved with the filter. In the second stage, the filtered projection data are backprojected onto the image to be reconstructed. Varieties of filtered-backprojection algorithms with different filters may be found in the literature [12]. As an example, the original 2D filtered-backprojection for a parallel beam geometry may be expressed as

$$f(x,y) = \int_0^{\pi} Q_\theta(x\ \cos\ \theta + y\ \sin\theta)\ \mathrm{d}\theta, \tag{1}$$

where

$$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(w)|w|\mathrm{e}^{j2\pi wt}\ \mathrm{d}w. \tag{2}$$

Eqs. (1) and (2) are the backprojection and filtering stages, respectively. In these two equations, $|w|$ denotes the filter, $S_\theta(w)$ the frequency response of the projection data measured in angle θ, $Q_\theta(t)$ the filtered projection data, and $f(x,y)$ the reconstructed image. As another example, the filter for the True Three-dimensional Reconstruction Algorithm [12] is

$$H_\theta(\rho,\xi) = \begin{cases} 2\rho(\cos\ \xi\ \sin\ \beta), & 0 \leqslant \xi \leqslant \frac{\pi}{2} - \beta, \\ 2\rho(\sin\ \xi \cos\ \beta), & \frac{\pi}{2} - \beta < \xi \leqslant \frac{\pi}{2}, \end{cases}$$

where $H_\theta(\rho,\xi)$ denotes the frequency response of the filter in polar coordinate and β the angle limiting the slices sharing projection data [12]. After each view of 2D projection data are convolved with the filter, the filtered projection data are projected onto the 3D image to be reconstructed. Clearly, for both 2D and 3D reconstruction, the algorithmic model for the filtered-backprojection algorithm may be described as:

(1) *Filtering:* Each view of projection data are convolved with the filter independently, and

(2) *Backprojection:* each view of filtered projection data are independently backprojected onto the same image to be reconstructed.

2.2. The EM algorithm

The EM algorithm generally consists of two primary steps in an iteration. One is forward projection and the other is backprojection. In the forward projection, the algorithm simulates physical processes, e.g., photon generation and detection, of a

PET system based on a certain model to generate the estimated projection data. Then, the estimated projection data would be compared with the measured projection data. In the backprojection, the discrepancy between the estimated and measured projection data is backprojected onto the image to be reconstructed. Then, the backprojected data would be used to modify the image obtained in the previous iteration. Additional steps may be inserted between these two primary steps, e.g., to speed up convergency or to get a better quality. A typical example is the EM algorithm proposed by Shepp and Vardi [37], which may be described as follows. In the EM algorithm, the object of interest is decomposed into a number of small cubical *boxes* (voxels). A pair of detectors defines a parallelepiped-like space called *tube*. Note that only those boxes inside the disk in each layer are to be reconstructed. The correction equation for the Shepp and Vardi's EM reconstruction algorithm can be written as:

$$\lambda^{\text{new}}(b) = \lambda^{\text{old}}(b) \sum_{t=1}^{T} \frac{n(t)p(b,t)}{\sum_{b=1}^{B} \lambda^{\text{old}}(b)p(b,t)}, \tag{3}$$

where $\lambda(b)$ is the number of photon pairs emitted from box b (the image to be reconstructed), $n(t)$ the number of photon pairs detected by tube t (projection data), $p(b, t)$ the probability that a photon pair emitted from box b is detected by tube t, T the total number of tubes, B the total number of boxes.

For each iteration, Eq. (3) can be decomposed into the following steps, where n, $\tilde{n}$, ε, δ and λ are the vector forms of $n(t)$, $\tilde{n}(t)$, $\varepsilon(t)$, $\delta(b)$, and $\lambda(b)$, respectively, and P the matrix form of $p(b, t)$, which is very sparse:

(1) $n = \lambda^{\text{old}} P$,

(2) $\varepsilon(t) = n(t)/n(t)$ for all t,

(3) $\delta = P\varepsilon^{\text{T}}$,

(4) $\lambda^{\text{new}}(b) = \delta(b)\lambda^{\text{old}}(b)$ for all b.

To avoid computing $p(b, t)$ on the fly which requires enormous computation, $p(b, t)$ is usually precomputed for a given system geometry. In these four steps, steps (1) and (3) are the forward projection and backprojection, respectively. Step (2) is for computing the correction factors between the estimated projection data and the measured projection data. Step (4) is to modify the image obtained in the previous iteration.

As a summary, the EM algorithm considered in this study may be modeled as four steps in an iteration.

(S1) *Forward projection:* Simulate the physical processes of a PET system to generate estimated projection data.

(S2) *Filtering and correction factor estimation:* Compute the discrepancy between the estimated and measured projection data. Generate the *correction factor* for each tube. Filtering may be applied to the estimated projection data or the correction factors to take into account some desired models, e.g., for a fast convergency or for a

better quality of image. It is assumed that these operations may be accomplished for each tube independently.

(S3) *Backprojection:* Backproject the correction factor to the boxes passed through by each tube. The backprojected datum on each box is called the *update factor*.

(S4) *Modification:* Modify the image obtained in the previous iteration according to the update factors. Filtering may be applied to the reconstructed image or the update factors to take into account some desired models, e.g., for a fast convergency or for a better quality of image. It is assumed that these operations may be performed for each box independently.

With the assumptions in steps (S2) and (S4), computations in steps (S1) and (S2) may be carried out for each tube independently and those in steps (S3) and (S4) may be executed for each box independently.

3. Data sharing modes

Defining data sharing modes is essential for the design of an efficient interconnection network. Since both the filtered-backprojection and EM algorithms are data-parallel algorithms, it is reasonable to replicate shared data in each PE such that each PE may perform computations until data coherence needs to be ensured. At this moment, all copies of replicated shared data need to be summed up and the results should be redistribute to all PEs to maintain data coherence. In addition, each PE needs to acquire initial data from the host in the beginning and report the final results to the host at the end. As a result, the data sharing modes employed in the proposed parallel algorithms are defined as follows. Throughout this paper, it is assumed that there are N PEs in the system and the size of the shared data is M. Moreover, without loss of generality, the shared data are divided into N segments, each with (N/M) shared data.

- *One-to-all broadcasting:* In this mode, one PE, e.g., PE_0, sends the entire shared data to all other PEs.
- *Scattering:* In this mode, PE_0 distributes the ith segment of the shared data to PE_i, for $0 \leqslant i < N$.
- *Integration:* Suppose the shared data are replicated in all PEs. In this mode, also known as *multinode accumulation* in [27], the replicated data in all PEs are summed up element-wise and the sum of the ith segment of all the replicated data will be assigned to PE_i, for $0 \leqslant i < N$.
- *All-to-all broadcasting:* In this mode, PE_i sends the ith segment of the shared data to all other PEs, for $0 \leqslant i < N$.

Among these four modes, the first two are to be used in the initialization stage of our parallel implementations. For examples, one-to-all broadcasting may be used for broadcasting the filter used in the filtered-backprojection algorithm to all PEs and scattering for distributing projection data of different views to different PEs. The third mode, integration, serves two purposes. The first purpose is to maintain data coherence of the replicated data. The second purpose is to maximize parallelization

efficiency by generating well-partitioned integrated data, i.e., the sum of all copies of replicated data, and tasks in each PE for the computations following integration. Note that since only partial integrated data would be utilized by each PE in these computations, it would be redundant to use such operation as *all_reduce* as defined in Message Passing Interface (MPI) standard [32] to maintain data coherence. All_reduce makes each PE have an entire copy of the integrated data. After these computations, which use the integrated data as input, have been completed, the results obtained in each PE are distributed to all other PEs by all-to-all broadcasting.

4. Parallel architecture and data sharing algorithms

The parallel system model considered in this study is an Multiple instruction, multiple data (MIMD) message passing model. In design of our dedicated parallel architecture, the ultimate goal is to attain the optimal parallelization efficiency subject to such constraints as hardware cost, clinically acceptable reconstruction time, system developing time, expandability, etc. The essential approach to achieve this goal is to exploit the parallelism involved in the reconstruction as much as possible in the architecture design. In this section, we present the proposed parallel architecture for PET image reconstruction and the data sharing algorithms tailored to the architectural features.

4.1. Parallel architecture

Based on the parallel system model, two levels of parallelism might be utilized to maximize parallelization efficiency. In the lower level, a special processor may be designed to take advantage of parallelism in the instructions. A typical example is the VLSI architecture proposed by Jones [24]. On the other hand, in the higher level, an interconnection network customized to all data sharing activities may be employed to exploit algorithmic parallelism to minimize data sharing overhead among PEs. This approach has been attempted by several previous works, e.g., the transputer-based system proposed by Atkins [2].

Theoretically, the highest parallelization efficiency may be obtained by fully utilizing both levels of parallelism. However, development of a new processor would cause a much higher hardware cost and a longer system developing time. On the contrary, using commercially available processors, including special-purpose processors (e.g., DSP processors) and general-purpose processors (e.g., Pentium processors), would avoid these two problems and, more importantly, could make use of the state-of-the-art processor technology. Hence, we propose to build a dedicated parallel system maximizing parallelization efficiency by (1) taking advantage of the commercially available processors, and (2) designing an interconnection network optimized for the underlying data sharing modes.

An ideal interconnection network for our system is expected to possess two essential properties. The first property is a constant connectivity. This property is desired so that the system can be expanded for a larger problem size easily in terms

of hardware cost and complexity. The second property is that it minimizes the data sharing overhead involved in the parallel algorithms. Taking these two properties into account, in this paper, we propose a four-connected parallel system using the static perfect shuffle network. As an example, the proposed parallel architecture with 8 PEs is illustrated in Fig. 1. Note that the perfect shuffle network has been unfolded in such a way that the left and right columns depict the ports 2 and 3 and the ports 0 and 1, respectively, of the original PEs for a better visualization of the network.

Although the static perfect shuffle network has been well studied in the literature [27,31], to our best knowledge, the optimal algorithms for integration and all-to-all broadcasting have not been investigated thoroughly. In the following, the optimal algorithms proposed in this paper for the underlying data sharing modes on the static perfect shuffle network are to be described. It is assumed that the host and the perfect shuffle based parallel architecture are connected through a single link between the host and PE_0.

4.2. Data sharing algorithms

The general block diagram for the proposed parallel image reconstruction algorithms may be sketched as in Fig. 2. In these algorithms, the data required for all

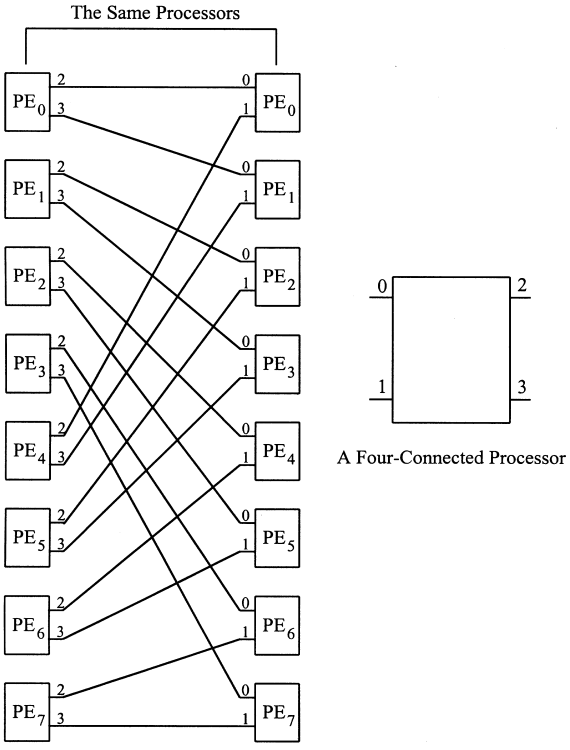

Fig. 1. 8 PEs interconnected by the static perfect shuffle network.

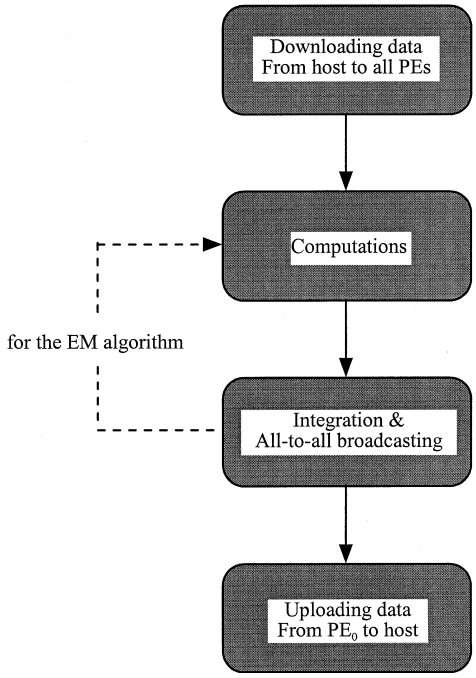

Fig. 2. Block diagram of the proposed parallel reconstruction algorithms.

PEs are first downloaded from the host. Shared data are replicated to each PE. At the end of the computation, all replicated data are integrated and broadcast such that all PEs have the same shared data for the following computation if any. When the program ends, PE_0 transfers the reconstructed image to the host.

Given an interconnection network, the optimal data sharing algorithms may vary with routing schemes employed, e.g., the store-and-forward routing or the wormhole routing. Although wormhole routing may largely eliminate the "distance" effect in data communication, this benefit greatly diminishes when all processors are involved in the data sharing activity, e.g., in integration and all-to-all broadcasting. As an example, in [27], it has been pointed that sending data through wormhole routing is not faster than through store-and-forward routing on a ring due to link contention. On the other hand, by deliberately scheduling the communication pattern, the communication links may be fully utilized and link contention may be minimized by using store-and-forward routing and the concept of pipelining. In this paper, we present the optimal data sharing algorithms in terms of order statistics for the four desired data sharing modes.

4.2.1. Downloading data

To optimize utilization of the perfect shuffle network for a high parallelization efficiency, the algorithms to download data and to integrate and broadcast shared

data, which fully exploits the topological feature of the perfect shuffle network, have been developed. For downloading data, there are basically two types of data sharing modes. One is one-to-all broadcasting and the other is scattering.

For scattering, a binary tree rooted at PE_0 embedded in the perfect shuffle network is employed as the downloading pattern. Moreover, the data are transferred in a pipelining fashion. The number of steps required to complete scattering is $\lceil \log(N + 1) \rceil$, where N is the number of PEs involved and "+1" is to account for the host. Fig. 3 illustrates the snapshots at the end of the four steps for scattering data from the host, which is not shown in this figure, to 8 PEs. Note that there are 9 PEs

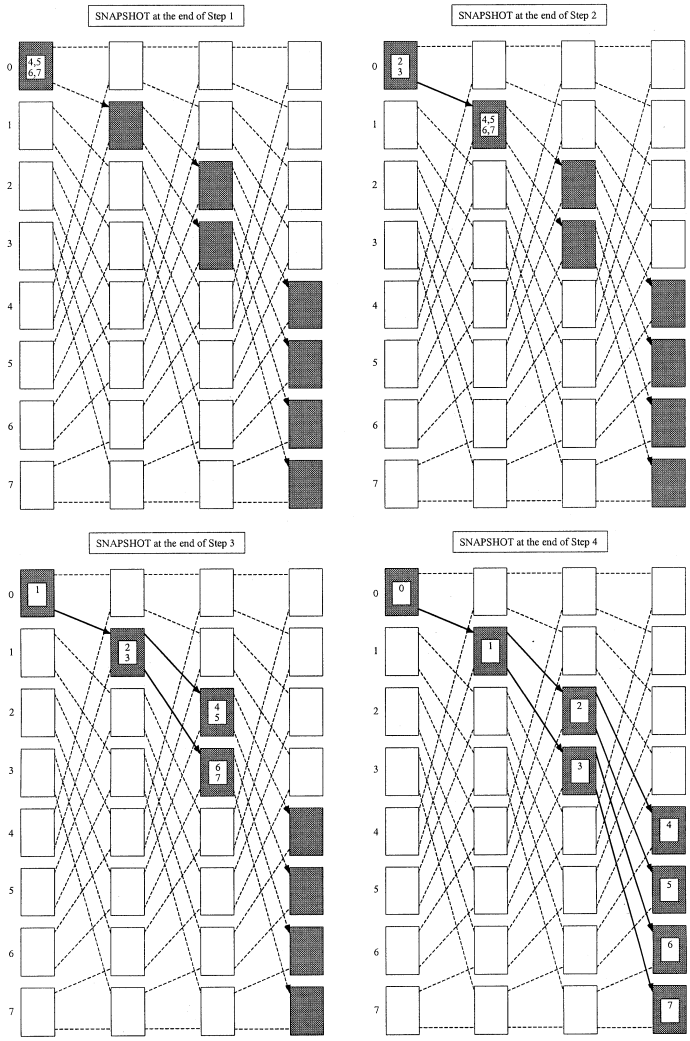

Fig. 3. The snapshots at the end of four steps for scattering data from the host to 8 PEs.

involved in total, including the host. Note that in Fig. 3, the perfect shuffle network is unfolded three times (i.e., the four columns of PEs in each snapshot are actually the same one,) to illustrate the embedded binary tree, in which the nodes are indicated by the dark blocks and the edges by the arrows. Moreover, in each snapshot, the numbers contained in the white block of each node represent the numbers of the segments arriving at that node. And, the line of an arrow changes from a dotted line to a thickened solid line if the corresponding edge is used in that step. Recall that in the scattering mode, PE_i is supposed to obtain the ith segment of data. These four steps may be summarized as below:

Step 1: The host sends a half of data, which are destined for PE_4–PE_7, to PE_0.

Step 2: While PE_0 sends the half of data received at step 1 to PE_1, the host sends a quarter of data, which are destined for PE_2 and PE_3, to PE_0.

Step 3: PE_1 splits the data received previously into two halves and sends the first half to PE_2 and the second half to PE_3. Meanwhile, PE_1 and PE_0 receive data from PE_0 and the host, respectively.

Step 4: PE_2 and PE_3 split the data into two halves and send different halves to different successors. At the same time, PE_0–PE_3 receive the data destined for themselves from their predecessors, respectively.

For one-to-all broadcasting, the data are transferred in a similar pipelining fashion on an embedded binary tree, except that the entire data are sent to all PEs and each packet consists of $(1/\log N)$ of the data. As scattering, it also takes $\lceil \log (N + 1) \rceil$ steps, which are not illustrated in this paper for brevity.

4.2.2. Integration and all-to-all broadcasting algorithms

Integration is required to ensure data coherence of the replicated shared data. Moreover, integration provides a balanced load distribution for the computations following integration and using the integrated data as the input. On the other hand, all-to-all broadcasting allows all PEs to have the entire shared data after each PE generates a segment of shared data.

The integration algorithm proposed in this study is illustrated in Fig. 4 for 8 PEs. The integration algorithm takes $\log N$ steps to complete integration. In this algorithm, at step k, $1 \leqslant k \leqslant \log N$, each PE splits its most updated data into two halves and sends these two halves to its two successors, respectively. In other words, each PE sends $1/2^k$ of total data to each of its two successors at step k. At the same time, each PE receives the same amount of data from each of its two predecessors. All four links operate simultaneously. Then, each PE sums up the two sets of data received from its predecessors correspondingly. At the end of step k, each PE has only $1/2^k$ of total data, which are valid. At the end of integration, each PE would have a segment of integrated data of the same size.

To broadcast the segment of integrated data from each PE to all other PEs, the data may be transferred following a communication pattern, including the directions and the message sequences, reverse to that in the integration, except that no summation is needed. That is at step k, each PEs sends its own $1/2^{(\log N+1-k)}$ of integrated data to its two successors and receives the same amount of data from its two

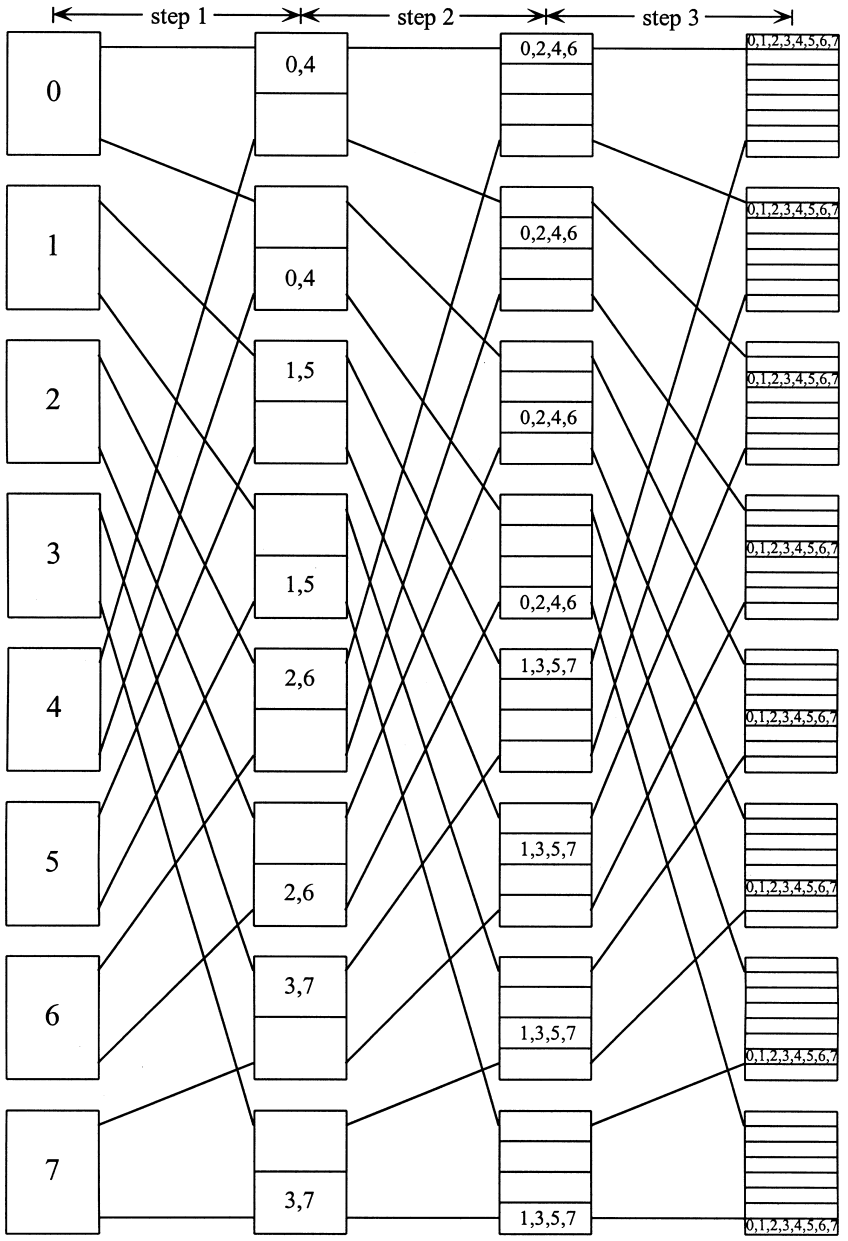

Fig. 4. An illustration for the integration algorithm for 8 PEs.

predecessors. At the end of step k, each PE has $1/2^{(\log N-k)}$ of integrated data. Note that the successors and predecessors of each PE during broadcasting are the predecessors and successors of this PE during integration, respectively.

4.2.3. Performance analysis and discussions

The performances of the proposed data sharing algorithms are to be analyzed in terms of order statistics to account for the various communication capabilities of the commercial processors. For example, some processors support full communication capability of all four links simultaneously, but others do not. In general, it is assumed that sending data of size m from one PE to another, both of which are directly connected, may be completed in time $O_{ls}(1) + O_{dx}(m)$. The subscript ls denotes link setup time, i.e., the time for establishing a link before data transfer, and the subscript dx stands for data transfer time, i.e., the time for the data travelling through the link. Moreover, for integration, the subscript sum indicates the summation time, i.e., the time for summing up two copies of shared data element-wise. Based on the simple model, the time analyses for the four data sharing modes are given in Propositions 1–4. Since we are interested in the performance of the static perfect shuffle, the communication time between the host and PE0 will be ignored in these performance analyses. Furthermore, for ease of analysis, it is assumed that the number of PEs, N, is a power of two.

Proposition 1. *The time required for the one-to-all broadcasting algorithm on the static perfect shuffle network is* $O_{ls}(\log N) + O_{dx}(M)$.

Proof. Recall that the data to be broadcast are divided into $\log N$ packets, each with $(M/\log N)$ elements. With N PEs, the number of pipeline stages is $\log N$. Therefore, the pipelining operations may be completed in $2 \log N - 1$ steps and each step can be accomplished in time $O(M/\log N)$. As a result, the total time required for the one-to-all broadcasting algorithm is $O_{ls}(\log N) + O_{dx}(M)$. □

Proposition 2. *The time required for the scattering algorithm on the static perfect shuffle network is* $O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$.

Proof. Since the step 1 of the scattering algorithm is for data transfer between the host and PE_0, the performance analysis will start with step 2 based on the assumption. From step 2, the size of data to be transferred across each link at step i is $(M/2^{i-1})$, for $2 \leqslant i \leqslant \lceil \log (N + 1) \rceil$. Excluding the step 1, the total number of steps required is $\log N$. Therefore, the total time required for the scattering algorithm is $O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$. □

Proposition 3. *The time required for the integration algorithm on the static perfect shuffle network is* $O_{ls}(\log N) + O_{dx}((1 - 1/N)M) + O_{sum}((1 - 1/N)M)$.

Proof. The proposed integration algorithm takes $\log N$ steps to complete integration. At step k, $1 \leqslant k \leqslant \log N$, each PE sends $(M/2^k)$ data to each of its two successors and receives the same amount of data from each of its two predecessors. Then, each PE sums up the two sets of data received from its predecessors correspondingly. Therefore, the total time required for the integration algorithm is $O_{ls}(\log N) + O_{dx}((1 - 1/N)M) + O_{sum}((1 - 1/N)M)$. □

Proposition 4. *The time required for the all-to-all broadcasting algorithm on the static perfect shuffle network is* $O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$.

Proof. Recall that the proposed all-to-all broadcasting algorithm follows a communication pattern, including the directions and the message sequences, reverse to that in the integration, except that no summation is needed. The same arguments for the proof of Proposition 3 may be applied. Therefore, the time required by the all-to-all broadcasting algorithm is the same as that required by the integration algorithm, except that the summation time should be excluded. That is, the total time is $O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$. □

As a comparison, the lower bounds of the data sharing times for the four data sharing modes on a mesh and a hypercube are provided in Table 1. The lower bounds for integration are given in [8]. They can also be obtained from [27] by considering integration as the dual of all-to-all broadcasting. The other lower bounds are derived according to the timing analyses in [27]. In [27], the exact timing expressions have been provided with the assumption of one-port communication. These expressions have been modified accordingly in terms of order statistics and simplified to a looser lower bound, if necessary, such that they are consistent with our analysis model. Except the lower bounds given in [8], all lower bounds derived from [27] have assumed the cut-through routing, which is supposed to yield a performance not worse than that based on the store-and-forward routing.

From Table 1, we can see that the proposed data sharing algorithms on a perfect shuffle network may achieve performance comparable to those on a hypercube with only four-connectivity in contrast to the $\log N$-connectivity in a hypercube. Compared to a mesh, the proposed data sharing algorithms and interconnection network obviously outperform those on a mesh. The outstanding performance of the proposed approaches arises from the fact that most data sharing algorithms for a hypercube may be easily mapped onto a perfect shuffle network, but much more difficult onto a mesh. Although efforts have been made to optimize such operation as global combine (also called all_reduce in the MPI standard) on a mesh by more complicated algorithms [3], the data transfer time and link setup time still cannot be optimized simultaneously.

When the link setup time is negligible, which may be due to a very small cost for link setup or a very large number of data to be transferred, the data sharing time is

Table 1
Lower bounds of data sharing time for the four data sharing modes on a mesh and a hypercube

	Mesh	Hypercube
One-to-all broadcasting	$O_{ls}(\log N) + O_{dx}(M)$	$O_{ls}(\log N) + O_{dx}(M)$
Scattering	$O_{ls}(\sqrt{N}) + O_{dx}((1 - 1/N)M)$	$O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$
Integration	$O_{ls}(\sqrt{N}) + O_{dx}((1 - 1/N)M)$ + $O_{sum}((1 - 1/N)M)$	$O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$ + $O_{sum}((1 - 1/N)M)$
All-to-all broadcasting	$O_{ls}(\sqrt{N}) + O_{dx}((1 - 1/N)M)$	$O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$

dominated by the data transfer time for all the four underlying data sharing modes. In this case, the hypercube, the mesh, and the perfect shuffle network are expected to have a similar performance. In addition to the remarkable performance, a ring may be easily found in the perfect shuffle network which is very useful in optimizing the parallel EM algorithms, e.g., for a perfect shuffle network with 32 nodes, a ring could be

16→0-1-2-5-18-4-9-19-25-12-6-13-26-29-30-31-15-7-3-17-8-20-10-21-11-22-27-23-14-28-24-16→0.

5. Parallel filtered-backprojection algorithm

One major goal in parallelization of sequential algorithms on multiprocessor systems is to achieve the largest *speedup* or highest *efficiency* such that the processing time is minimized. The *speedup* is defined as (the sequential processing time)/(the parallel processing time) and the *efficiency* as (the speedup)/(the number of PEs employed). The optimal performance, i.e., the largest speedup or the highest efficiency, may be attained by optimizing task partitioning and data sharing algorithms.

As defined in our algorithmic model, the filtered-backprojection algorithm is composed of two essential steps, namely, filtering and backprojection. For both 2D and 3D filtered-backprojection algorithms, the projection data in each view are first filtered with a kernel. Then, the filtered-projection data are backprojected onto the image space to be reconstructed. Since each view has about the same amount of computations in a filtered-backprojection algorithm, we propose to partition the task according to views. That is, given V views and N PEs, each PE would take care of computations for V/N views and computational loads are balanced in theory. At the beginning of the program, the host downloads V/N views of projection data to each PE to carry out filtering. Since the host would download different projection data to different PEs, the scattering algorithm is employed. In addition to projection data, the host also downloads filters to all PEs using the one-to-all broadcasting algorithm. Since computations of filtering for different views are completely independent, no data sharing would be required during filtering.

During backprojection, since all views need to backproject to the same image space, the image to be reconstructed is replicated entirely at the local memory of each PE such that all PEs may perform backprojection without communication with other PEs. However, to obtain the correct reconstructed image, the replicated images at the local memories of all PEs should be integrated after all PEs have finished backprojection. To integrate the replicated images, the integration algorithm is used. Since the integration algorithm would result in a scattered type of integrated image, i.e., each PE has only $1/N$ of the integrated image, the all-to-all broadcasting algorithm is used such that PE_0 would have the entire reconstructed image to send back to the host. Note that a simple binary-tree type of algorithm may also be used to collect all scattered integrated data to a single PE, which is expected to have the same performance as the all-to-all broadcasting algorithm. Another way to sum up all

copies of the replicated data is to use *reduce* operation as defined in MPI standard, which accumulates the final results into PE_0. Since integration allows a balanced computations on the integrated data after summation of all copies of shared data, which might be required for post-processing, integration is still preferred in this study, though integration and reduce operations have a similar performance.

The filtered-backprojection algorithm proposed in [36] has been parallelized on a transputer-based parallel system with the perfect shuffle interconnection network. Limited by the memory capacity available in each PE, i.e., 4M in a node PE and 8M in the host, we can only realize 2D filtered-backprojection. However, since the parallel algorithms for both 2D and 3D filtered algorithms are essentially the same, parallelizing a 2D algorithm would be sufficient to see the correctness and feasibility of the proposed parallel algorithm. Moreover, it is expected that the parallelization efficiency for the 3D parallel filtered-backprojection would be higher than that for the 2D case since the computation/communication ratio for the former is much higher than that for the latter.

Different numbers of PEs, i.e., 1, 2, 4, 8, 16 and 32 PEs, have been employed in the parallel implementations. The image sizes tested are 64×64, 128×128, 256×256, and 512×512. Table 2 gives the execution times of all tested cases. Besides, the downloading and uploading times are also listed in this table, which were obtained by eliminating the computation parts in the parallel algorithms.

From Table 2, it can be seen that the I/O time is approximately linearly proportional to the size of the image. And, it does not increase significantly as the number of PEs increases. This observation is consistent with the theoretical analysis for a negligible link setup time. To see the parallelization efficiency, the speedups are given in Table 3 and the efficiencies in Fig. 5. In both Table 3 and Fig. 5, the "+I/O" indicates that the speedups or efficiencies are calculated including I/O times and "−I/O" excluding I/O times.

The parallelization efficiency of the proposed parallel filtered-backprojection algorithm may be modeled as

$$E = \frac{t_0}{t_0 + (2\alpha + \beta)(N-1)M + 2\gamma N \log N + Nt_{\mathrm{I/O}}}, \tag{4}$$

Table 2
Execution times including and excluding I/O time of the parallel filtered-backprojection algorithm

#PEs	64		128		256		512	
	Execution time	I/O time	Execution time	I/O time	Execution time	I/O time	Execution time	I/O time
1	6.17	0	4.91	0	394	0	3130.7	0
2	3.18	0.02	25.02	0.08	198.77	0.34	1584.95	1.34
4	1.7	0.03	13.02	0.10	102.33	0.37	811.89	1.47
8	1.04	0.03	7.73	0.10	59.91	0.39	471.98	1.56
16	0.59	0.03	4.12	0.10	31.03	0.40	241.40	1.59
32	0.37	0.03	2.29	0.11	16.42	0.41	124.73	1.61

Table 3
Speedups of the parallel filtered-backprojection algorithm

#PEs	64		128		256		512	
	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O
2	1.94	1.95	1.96	1.97	1.98	1.99	1.98	1.98
4	3.63	3.69	3.77	3.8	3.85	3.86	3.86	3.86
8	5.93	6.11	6.35	6.44	6.58	6.62	6.63	6.66
16	10.46	11.02	11.92	12.21	12.70	12.86	12.97	13.05
32	16.68	18.15	21.44	22.52	24.00	24.61	25.10	25.43

where t_0 is the sequential processing time, M the number of shared data to be integrated and N the number of PEs involved. α is the average time for transferring the data associated with one shared datum during integration and broadcasting, β the average time for integrating one shared datum, γ the average time for each link setup and $t_{I/O}$ the I/O time, including uploading and downloading data.

In general, Table 3 and Fig. 5 reveal that given a number of PEs, the larger the image size is, the more efficient the parallel filtered-backprojection algorithm would be. The reason is for 2D reconstruction, as the size of images increases, the amount of computational loads, i.e., t_0/N, increases with the third order while the size of the replicated images, i.e., M, and that of projection data increase with the second order. That is, the ratio of computational load to communication overhead is $O(N_I)$, assuming that the size of the 2D image to be reconstructed is $N_I \times N_I$. It is expected that the parallelization efficiency for 3D reconstruction would be much better than that for 2D reconstruction since the ratio of computational load to communication overhead for the 3D case is $O(N_I^2)$.

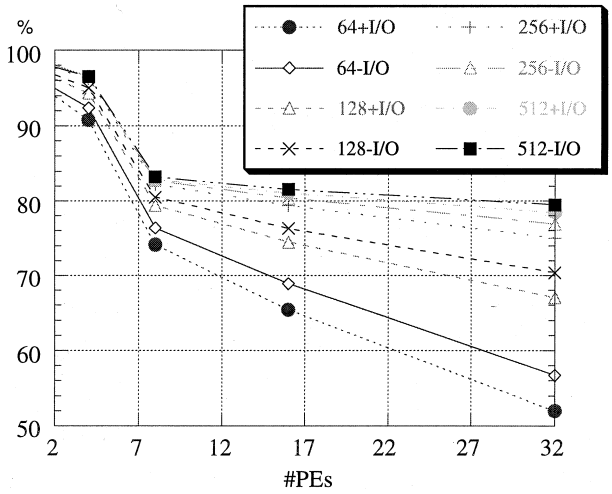

Fig. 5. Efficiencies of the parallel filtered-backprojection algorithm.

On the other hand, given an image size, the more PEs the system employs, the less efficient the parallel algorithm would be, though the speedup increases. Since the link setup is relatively negligible when N_I is large enough, the performance shown in Fig. 5 approximately exhibits the behavior of

$$E \approx \frac{1}{1 + kN} \tag{5}$$

for those cases marked "−I/O". Due to the effect of link setup term, workload variation, and measurement errors, k varies slightly with N and image size. Since M/t_0 inversely proportional to the image size, k is smaller for a larger image size. In other words, the efficiency degrades faster for a smaller image size.

From these experiments, we can also see that the time for downloading and uploading data is insignificant for a large image size, but is non-negligible for a small image size. It is mainly because the parallel processing time for reconstructing an image of a small size is very short.

6. Parallel EM algorithm

The EM algorithm is a large-scale data-parallel algorithm [6], which has very rich data parallelism. The two major computations in each iteration of the EM algorithm are large sparse-vector matrix multiplications in the steps (S1) and (S3). Although sparse matrix computation [17,26] has been studied extensively, minimizing the data sharing overhead involved in these two sparse-vector matrix multiplications is non-trivial due to the conflicting requirements for optimal task and data partitioning in both multiplications.

The major data parallelism in the EM algorithm is scalar multiplications. A reasonable way to utilize the data parallelism may be described by two spaces, namely, *box* and *tube* spaces. For each step, if a box (or tube) is assigned to a PE, all the tasks and data associated with this box (or tube) are also assigned to the PE. It is clear that in the steps (S1) and (S2), the computations for different tubes may be performed in parallel and the outputs are mutually exclusive. That is, it would be better to partition the tasks and data according to the *tube* space to achieve an exclusive partition in the steps (S1) and (S2) so that no overhead would be incurred to maintain data coherence. However, in the steps (S3) and (S4), the *box* space is preferred. It is this conflicting characteristic making the data sharing overhead inevitable in the parallelization of the EM algorithm.

In this study, we employ the *modified partition-by-box* scheme, which is proposed in one of our previous works [8], to partition the tasks involved in the EM algorithm. The modified partition-by-box scheme is a modified version of the partition-by-box scheme [8]. For ease of description, the partition-by-box scheme is described first. The partition-by-box scheme partitions the tasks and data according to the *box* space for both of the steps (S1) and (S3). The *box* space is partitioned such that all PEs have about the same number of box-tube pairs. Note that, for the Shepp and Vardi's EM algorithm, each box-tube pair is associated with one scalar multiplica-

tion in each of the steps (S1) and (S3). In the steps (S2) and (S4), each PE performs $(1/N)$ of the computations.

To avoid communication among PEs during the step (S1), the estimated projection data are replicated at the local memory of each PE. At the end of step (S1), to ensure the data coherence, the replicated estimated projection data are integrated using the integration algorithm. After integration, each PE has $(1/N)$ of integrated estimated projection data, which exactly matches the need for the step (S2). At the end of step (S2), since each PE needs almost all the correction factors in the step (S3), each PE broadcasts its own $(1/N)$ of correction factors to all other PEs using the all-to-all broadcasting algorithm.

The potential problem of the partition-by-box scheme is that the computational load might not be well balanced since the computational load associated with each box is different. To minimize the potential load imbalance problem, we further partition all the tubes associated with some boxes. More precisely, one can imagine that all boxes are arranged into a 1D array. The partition-by-box scheme divides this array into N segments such that the total number of box-tube pairs associated with each segment is as close as possible. Then, each PE takes care of computations for one segment. To balance the computational loads completely, the box-tube pairs associated with each box connecting two segments are further divided into two subsets, each assigned to one of the two adjacent segments. Again, to avoid communication during computation, these subdivided boxes are replicated to the two PE taking care of the two segments sharing the box. Therefore, at the end of step (S3), the two PEs sharing the same box need to exchange the computed box values to derive the correct one. In order to accomplish this procedure, all PEs need to be interconnected as a ring fashion (or at least as a linear array) in addition to be interconnected by the perfect shuffle network. This scheme is called the *modified partition-by-box* scheme.

Unlike the parallel filtered-backprojection algorithm, many data need to be downloaded to all PEs at the beginning of the proposed parallel EM algorithm. These data include projection data, probability matrix and its indices (since it is a sparse matrix and only non-zero elements are sent), and the indices for the boxes assigned to each PE. Among these data, the projection data are downloaded using one-to-all broadcasting algorithm and the others using scattering algorithm.

The Shepp and Vardi's EM algorithm [37] has been parallelized with different numbers of PEs, i.e., 1, 2, 4, 8, 16, and 32 PEs. To see the effect of I/O, including downloading and uploading, various number of iterations have been performed. They are 1, 2, 4, 8, 16 and 32 iterations. Again, limited by the memory capacity, the size of image and the size of the PET system that we could simulate are 64×64 and one ring with 96 detectors, respectively. Since, algorithmically, the EM algorithm is independent of dimensionality, parallelizing the 2D case would be adequate to see the feasibility for the 3D case, except the latter is expected to have a higher efficiency. For reference, the I/O times for using different number of PEs are listed in Table 4.

Table 4 indicates that the I/O time decreases first and then increases. This phenomenon may be ascribed to two important factors determining the I/O time, namely, the size of data sent to each PE and the number of stages the data binary

Table 4
The I/O times for using various number of PEs in the parallel EM algorithm

#PEs	1	2	4	8	16	32
Time (s)	0	3.53	2.64	2.17	2.17	2.34

tree has. As the number of PEs increases, the first factor decreases but the second one increases. The minimal point turns out to be somewhere between 8 and 16 PEs in our case.

The total execution times for different numbers of iterations are listed in Table 5. In the same table, the execution times excluding I/O time are also listed for analysis. As before, +I/O indicates with the I/O time and −I/O without the I/O time.

From Table 5, one can see that when the number of iterations is small, the overhead caused by downloading and uploading is quite significant. The extreme case is when only two PEs are used for one iteration. In this case, it takes even more time than using only one PE. The reason why the I/O time is so significant is due to the large amount of data to be downloaded and at the same time the image size which can be simulated is too small. As a result, the I/O operations take a great portion of the total execution time when the number of iterations or the number of PEs is small. Given a number of PEs, as the number of iterations increases, the overhead caused by I/O becomes less influential. This statement may be verified by Tables 6 and 7 in which the speedups and efficiencies for the cases with (+I/O) and without (−I/O) I/O times are listed, respectively.

The parallelization efficiency of the proposed parallel EM algorithm may be modeled as

$$E = \frac{n_i t_0}{n_i t_0 + n_i[(2\alpha + \beta)(N-1)M + 2\gamma N \log N] + N t_{\mathrm{I/O}}}, \tag{6}$$

where n_i is the number of iterations and other parameters are as defined in Eq. (4). Basically, Eq. (6) is quite similar to Eq. (4) except that the total time for computation, integration and all-to-all broadcasting are proportional to the number of iterations executed. From Tables 6 and 7, it is clear that given a number of PEs, the more iterations are performed, the higher efficiency the algorithm may achieve. However, the upper bound of the achievable efficiency for each given number of PEs

Table 5
Execution times including and excluding I/O time of the parallel EM algorithm

iter->	1		2		4		8		16		32	
#PEs	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O
1	5.17	5.17	10.34	10.34	20.68	20.68	41.36	41.36	82.00	82.00	162.4	162.4
2	6.35	2.82	9.15	5.62	14.74	11.21	25.92	22.39	48.29	44.76	93.02	89.49
4	4.3	1.66	5.77	3.13	8.75	6.11	14.66	12.02	26.45	23.81	50.08	47.44
8	3.46	1.29	4.43	2.26	6.38	4.21	10.28	8.11	18.09	15.92	33.62	31.45
16	2.97	0.8	3.57	1.40	4.77	2.60	7.17	5.00	11.94	9.77	21.50	19.33
32	2.80	0.49	3.20	0.89	4.00	1.69	5.60	3.29	8.80	6.49	15.17	12.86

Table 6
Speedups including and excluding I/O time of the parallel EM algorithm

iter-> #PEs	1		2		4		8		16		32	
	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O	+I/O	−I/O
2	0.81	1.83	1.13	1.84	1.40	1.84	1.60	1.85	1.70	1.83	1.75	1.81
4	1.20	3.11	1.79	3.30	2.36	3.38	2.82	3.44	3.1	3.44	3.24	3.42
8	1.49	4.00	2.33	4.58	3.24	4.91	4.02	5.10	4.53	5.15	4.83	5.16
16	1.74	6.46	2.90	7.39	4.34	7.59	5.77	8.27	6.87	8.39	7.55	8.40
32	1.85	10.55	3.23	11.63	5.17	12.24	7.39	12.57	9.32	12.63	10.70	12.63

Table 7
Efficiencies including and excluding I/O time of the parallel EM algorithm

iter->- #PEs	1		2		4		8		16		32	
	+I/O (%)	−I/O (%)	+I/O (%)	−I/O (%)	+I/O (%)	−I/O (%)	+I/O (%)	−I/O (%)	+I/O (%)	−I/O (%)	+I/O (%)	−I/O (%)
2	40.1	91.5	56.5	92.0	70.0	92.0	80.0	92.5	85.0	91.5	87.5	90.5
4	30.0	77.8	44.8	82.5	59.0	84.5	70.5	86.0	77.5	86.0	81.0	85.5
8	18.6	50.5	29.0	57.3	40.5	61.4	50.3	63.8	56.6	64.4	60.4	64.5
16	10.9	40.4	18.1	46.2	27.1	49.7	36.1	51.7	42.9	52.4	47.2	52.5
32	5.8	33.0	10.1	36.3	16.2	38.3	23.0	39.3	29.1	39.5	33.4	39.5

would be close to the efficiency computed by excluding the I/O time for a large number of iterations. For example, the maximal efficiency attainable with 32 PEs would be around 39.5%. This phenomenon may be clearly seen in Fig. 6.

It should be pointed out that the reason why the maximal achievable parallelization efficiencies for using a large number of PEs are not very high is due to the combining effect of the small problem size that we can simulate and the slow communication rate provided by the transputers. This combining effect results in a small computation/communication ratio. However, the parallelization efficiency may increase rapidly as the problem size increases. The reason is as follows. On one hand, for an $N_I \times N_I$ 2D image, the computational load is about $O(N_I^3)$. For an $N_I \times N_I$ 3D image, the computational load is about $O(N_I^5)$. On the other hand, as the number of iterations is large enough, the major overhead is caused by integration and all-to-all broadcasting. But, as pointed out before, the proposed integration and all-to-all broadcasting algorithms have been theoretically optimal which is linearly proportional to the size of data to be integrated and broadcast, i.e., $O(N_I^2)$ and $O(N_I^3)$ for 2D and 3D cases, respectively. Hence, as N_I increases, the computational load would increase $O(N_I)$ and $O(N_I^2)$ times faster than integration and all-to-all broadcasting for 2D and 3D cases, respectively. That is, the data sharing overhead would become much less significant for a larger problem size. As a matter of fact, there is nothing much we can do to further improve the performance of the proposed parallel EM algorithm algorithmically on the transputer system employed. In addition to increases the problem size, one way to get a higher parallelization efficiency is to use processors with fast communication capability.

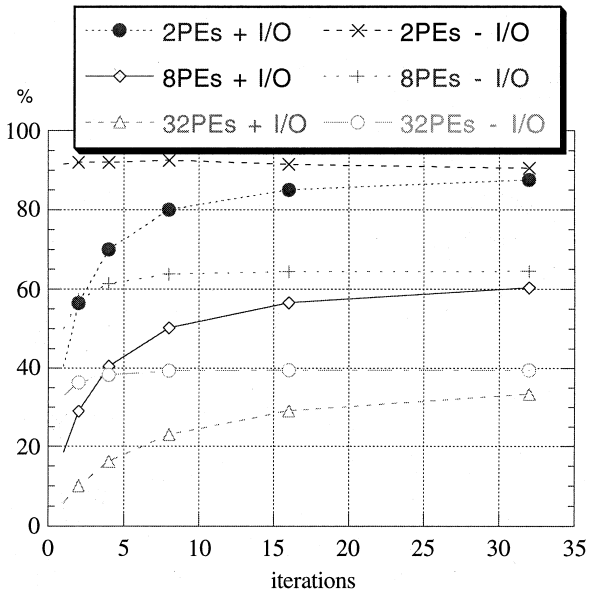

Fig. 6. Upper bounds of the achievable efficiencies by the proposed parallel EM algorithm on the transputer system employed.

7. Conclusions

From the experimental results obtained for using a single PE in this paper, the long computation time confirms that parallel processing is necessary for a practical 3D PET system. To speed up the reconstruction, we have designed a dedicated parallel system for PET image reconstruction based on the static perfect shuffle network. The distinguished feature of the perfect shuffle network is that with only four connectivity, it may accomplish all types data sharing activities involved in both filtered-backprojection and EM algorithms in a time comparable to that on a hypercube. Moreover, the proposed parallel system is optimal for *both* the filtered-backprojection and EM algorithms. In addition, a ring communication pattern can be easily embedded in the PPS network, which is very useful in optimizing the parallel EM algorithm.

Based on the topological feature of the perfect shuffle network, we have developed optimal data sharing algorithm for one-to-all broadcasting, scattering, integration and all-to-all broadcasting on the perfect shuffle network. The time required by the one-to-all broadcasting algorithm is, the times for the scattering and all-to-all broadcasting algorithms are the same, which is $O_{ls}(\log N) + O_{dx}((1 - 1/N)M)$, and the time for the integration algorithm is $O_{ls}(\log N) + O_{dx}((1 - 1/N)M) + O_{sum}((1 - 1/N)M)$.

With the proposed data sharing algorithms, we have developed efficient parallel filtered-backprojection and EM algorithms taking advantage of the perfect shuffle

network. Although the parallelization efficiencies demonstrated in the experiments are not very high, they are mainly due to the combining effect of the small problem size we can simulated and the slow communication rate provided by the transputer system. This combining effect results in a small computation/communication ratio. It is believed that a high efficiency may be attained for a real 3D PET image reconstruction if a sufficient number of PEs and memories are available or if high performance processors are employed with fast communication capability.

References

[1] Analogic Corporation, Wakefield, MA, Modular Image Processor (MIP), IP-300 Technical Manual, 1981.

[2] M.S. Atkins, D. Murray, R.L. Harrop, Use of transputers in a 3-D positron emission tomography, IEEE Trans. Med. Imaging 10 (1991) 276–283.

[3] M. Barnett, R. Littlefield, D.G. Payne, R. Van De Geijn, Global combine algorithms for 2-D meshes with wormhole routing, J. Parallel and Distributed Comput. 24 (1995) 191–201.

[4] S. Barresi, D. Bollini, A. Del Guerra, Use of a transputer system for fast 3-D image reconstruction in 3-D PET, IEEE Trans. Nucl. Sci. 37 (1990) 812–817.

[5] K. Bastiaens, I. Lemahieu, P. Desmedt, On the use of a multi-threaded operating system for an efficient parallel implementation of the ML-EM algorithm for PET image reconstruction, IFIP Trans. A: Comput. Sci. and Technol. 44 (1994) 31–39.

[6] C.M. Chen, On minimizing data sharing overhead for large-scale data-parallel algorithms: Replication and allocation of shared data, Ph.D. Thesis, Cornell University, Ithaca, New York, 1993.

[7] C.M. Chen, S.-Y. Lee, Z.H. Cho, 3D PET image reconstruction on a mesh connected multiprocessor, in: 1992 Medical Imaging Conference Record, 1992.

[8] C.M. Chen, S.-Y. Lee, On parallelizing the EM algorithm for PET image reconstruction, IEEE Trans. Parallel and Distributed Systems 5 (6) (1996) 860–873.

[9] C.M. Chen, S.-Y. Lee, Optimal data replication: A new approach to optimizing parallel EM algorithms on a mesh-connected multiprocessor for 3D PET image reconstruction, IEEE Trans. Nucl. Sci. 42 (4) (1995) 1235–1245.

[10] C.M. Chen, S.-Y. Lee, Z.H. Cho, A parallel implementation of 3-D CT image reconstruction on hypercube multiprocessor, IEEE Trans. Nucl. Sci. 37 (1990) 1333–1346.

[11] C.M. Chen, S.-Y. Lee, Z.H. Cho, Parallelization of the EM algorithm for 3D PET image reconstruction, IEEE Trans. Med. Imaging 10 (1991) 513–522.

[12] Z.H. Cho, P.J. Jones, S. Manbir, Foundations of Medical Imaging, Wiley, New York, 1993.

[13] P.S. Crandall, C.W. Stearns, A scalable multiprocessor implementation of the reprojection algorithm for volumetric PET imaging, in: 1995 IEEE Nuclear Science Symposium and Medical Imaging Conference Record, vol. 2, 1995, pp. 1184–1188.

[14] V. Di Lecce, E. Di Sciascio, A.R. Manni, A pipeline backprojection for on-line 3-D PET, in: 1995 IEEE Nuclear Science Symposium and Medical Imaging Conference Record, vol. 2, 1995, pp. 1069–1073.

[15] V. Di Lecce, E. Di Sciascio, A.R. Manni, Parallelization of 3-D PET BpjF reconstruction on a DSP cluster, in: 1995 IEEE Nuclear Science Symposium and Medical Imaging Conference Record, vol. 2, 1995, pp. 1222–1226.

[16] M.L. Egger, S.A. Herrmann, C. Joseph, C. Morel, Fast volume reconstruction in positron emission tomography: Implementation of four algorithms on a high-performance scalable parallel platform, in: Proceedings of the 1996 IEEE Nuclear Science Symposium, Anaheim, CA, 1996, pp. 1574–1578.

[17] G.C. Fox, Load balancing and sparse matrix vector multiplication on the hypercube, Tech. Rep. C^3P-327, Caltech, 1986.

[18] P.J. Green, Bayesian reconstructions from emission tomography data using a modified EM algorithm, IEEE Trans. Med. Imaging 9 (1990) 84–93.
[19] T.M. Guerrero, S.R. Cherry, M. Dahlbom, A.R. Ricci, E.J. Hoffman, Fast implementation of 3D PET reconstruction using vector and parallel programming techniques, IEEE Trans. Nucl. Sci. 40 (4) (1993) 1082–1086.
[20] H. Hart, Z. Liang, Bayesian image processing in two dimensions, IEEE Trans. Med. Imaging 6 (1987) 199–206.
[21] R. Hartz, D. Bristow, N. Mullani, A real-time TOFPET slice-backproject engine employing dual Am 29116 microprocessors, IEEE Trans. Nucl. Sci. 32 (1985) 839–842.
[22] G.T. Herman, D. Odhner, K.D. Toennies, S.A. Zenios, A parallelized algorithm for image reconstruction from noisy projections, in: Large-Scale Numerical Optimization, 1989.
[23] W.F. Jones, L.G. Byars, M.E. Casey, Positron emission tomographic images and expectation maximization: A VLSI architecture for multiple iterations per second, IEEE Trans. Nucl. Sci. 35 (1988) 620–624.
[24] W.F. Jones, L.G. Byars, M.E. Casey, Design of a super fast three-dimensional projection system for positron emission tomography, IEEE Trans. Nucl Sci 37 (1990) 800–804.
[25] L. Kaufman, Implementing and accelerating the EM algorithm for positron emission tomography, IEEE Trans. Med. Imaging 6 (1987) 37–50.
[26] C.P. Kruskal, L. Rudolph, M. Snir, Techniques for parallel manipulation of sparse matrices, Theoretical Computer Science (1989) 135–157.
[27] V. Kumar, A. Grama, A. Gupta, G. Karypis, Introduction to Parallel Computing: Design and Analysis of Algorithms, Benjamin/Cummings, Menlo Park, CA, 1994.
[28] E. Levitan, G.T. Herman, A maximum a posterior probability expectation maximization algorithm for image reconstruction in emission tomography, IEEE Trans. Med. Imaging 6 (1987) 185–191.
[29] R.M. Lewitt, G. Muehllehner, Accelerated iterative reconstruction for positron emission tomography based on the EM algorithm for maximum likelihood estimation, IEEE Trans. Med. Imaging 5 (1986) 16–22.
[30] J. Llacer, J.D. Meng, Matrix-based image reconstruction methods for tomography, IEEE Trans. Nucl. Sci. 32 (1985) 855–864.
[31] J. Lopez, O. Plata, F. Arguello, E.L. Zapata, Unified framework for the parallelization of divide and conquer based tridiagonal systems, Parallel Comput. 23 (1997) 667–686.
[32] Message Passing Interface Forum, MPI: A Message Passing Interface Standard, 5 May 1994.
[33] S.P. Olesen, J. Gregor, M.G. Thomason, G.T. Smith, EM-ML PET reconstruction on multiple processors with reduced communications, Int. J. Imaging System and Technology 7 (3) (1996) 215–223.
[34] K. Rajan, L.M. Patnaik, J. Ramakrishna, High-speed computation of the EM algorithm for PET image reconstruction, IEEE Trans. Nucl. Sci. 41 (5) (1994) 1721–1728.
[35] F.U. Rosenberger, D.G. Politte, G.C. Johns, C.E. Molnar, An efficient parallel implementation of the EM algorithm for PET image reconstruction utilizing transputers, in: 1990 Nuclear Science Symposium Conference Record, 1990.
[36] L.A. Shepp, B.F. Logan, The Fourier reconstruction of a head section, IEEE Trans. Nucl. Sci. 21 (1974) 21–43.
[37] L.A. Shepp, Y. Vardi, Maximum likelihood reconstruction for emission tomography, IEEE Trans. Med. Imaging 1 (1982) 113–122.
[38] C.J. Thompson, T.M. Peters, A fractional address accumulator for fast backprojection, IEEE Trans. Nucl. Sci. 28 (1981) 3648–3650.