Hybrid MPI-OpenMP Programming for Parallel OSEM PET Reconstruction

M. D. Jones, *Member, IEEE*, R. Yao, *Member, IEEE*, and C. P. Bhole

***Abstract*—To improve the parallel efficiency (PE) of the ordered-subsets expectation-maximization (OSEM) algorithm for three-dimensional (3-D) positron emission tomography (PET) image reconstruction, we focused on reducing the computational imbalance among parallel processes and interprocess data exchange time which were the dominant limiting factors of PE when a large number of networked compute nodes were used. As clusters with multiple processors on each compute node have become increasingly common, we have aimed to take advantage of the load-balancing mechanism and the inherently lower latency of shared memory threads across processors within a single node. We, therefore, implemented the OSEM algorithm with a hybrid message passing interface (MPI) and OpenMP approach on the basis of a standard MPI implementation. The contributing components to the total reconstruction time for the hybrid technique were quantified and compared to that using only MPI. The hybrid MPI-OpenMP technique achieved a consistent PE improvement of approximately 7% to 17% compared to the pure MPI approach on the same number of compute nodes. As clusters of larger shared-memory multiprocessor (SMP) machines continue to become more cost effective, we expect this hybrid MPI-OpenMP approach to be increasingly valuable.**

***Index Terms*—Image reconstruction, parallel processing, positron emission tomography (PET).**

I. Introduction

In the last 10 years, iterative statistical image reconstruction has become the primary choice on positron emission tomography (PET) systems for its strong system and noise modeling capability [1]. However, to achieve high spatial resolution, modern PET systems use a large number of detectors and produce large volume images with small voxels. As the product of the detector-pair number and image size is the size of the system matrix that models the PET image formation process, the task of statistical iterative image reconstruction, i.e., solving the equation modeled by the system matrix iteratively, is very computationally intensive [2], [3]. It can take many hours to reconstruct a single frame whole body high-resolution PET image. Table I shows the number of detector-pairs and image volume of a few high-resolution PET systems to illustrate the computational magnitude of PET image reconstruction tasks.

Manuscript received December 25, 2005; revised July 19, 2006.

M. D. Jones is with the Center for Computational Research, State University of New York at Buffalo, Buffalo, NY 14222 USA (e-mail: jonesm@ccr.buffalo.edu).

R. Yao is with the Department of Nuclear Medicine, State University of New York at Buffalo, Buffalo, NY 14222 USA (e-mail: rutaoyao@buffalo.edu).

C. P. Bhole is with the Department of Computer Sciences and Engineering, State University of Buffalo, Buffalo, NY 14222 USA (e-mail: cpbhole@buffalo.edu).

Digital Object Identifier 10.1109/TNS.2006.882295

TABLE I
THE NUMBER OF DETECTOR PAIRS AND IMAGE VOLUMES OF 3 HIGH-RESOLUTION PET SCANNERS ARE SHOWN TO ILLUSTRATE THE MAGNITUDE OF THE COMPUTATION TASK OF IMAGE RECONSTRUCTION. THE ATLAS [16] AND MICROPET FOCUS 120 [17] ARE FOR ANIMAL STUDY AND HRRT [18] IS FOR HUMAN BRAIN STUDY

PET Scanner	Number of detector-pairs	Image size (voxel size)
ATLAS	$2.13\text{x}10^6$	$5.57\text{x}10^5$ (0.56x0.56x0.56 mm^3)
microPET Focus	$4.48\text{x}10^7$	$1.56\text{x}10^6$ (0.86x0.86x0.86 mm^3)
HRRT	$4.49\text{x}10^9$	$1.36\text{x}10^7$ (1.2x1.2x1.2 mm^3)

A. Prior Parallelized PET Reconstruction Work

Parallelizing the reconstruction task over a computer cluster with multiple processors is an effective mechanism to reduce the long reconstruction time [4]. In addition, tasks such as storing the system matrix in memory for PET reconstruction, which may not be practical on a single processing unit, can now be divided into smaller tasks and realized on clusters with enough compute units [5]. The implementation of a parallel reconstruction application depends strongly on the computing infrastructure: the software libraries available and the underlying hardware architecture. As briefly summarized by Jones *et al.* [6], various parallelization techniques have been investigated by different groups [7]–[11] as parallel computing platforms have evolved over the past two decades. Most recently, the networked low cost commodity PC clusters using the Linux operating system and message passing interface (MPI) [12] libraries have become the preferred platform for parallel PET reconstruction development [3], [13]–[15]. This is the platform that we study in this paper.

B. Parallelized OSEM

The ordered-subsets expectation-maximization (OSEM) [19] algorithm is an accelerated iterative statistical PET reconstruction algorithm. Its primary formula is given by

$$\theta_i^{k,s+1} = \frac{\theta_i^{k,s}}{q_i^s} \sum_{j \in J_s} \frac{C_{i,j} y_j}{\hat{y}_j^{k,s}} \tag{1}$$

$$q_i^s = \sum_{j \in J_s} C_{i,j} \tag{2}$$

$$\hat{y}_j^{k,s} = \sum_{i=1}^{I} C_{i,j} \theta_i^{k,s} \tag{3}$$

$$\theta^{k+1,1} = \theta^{k,S+1} \tag{4}$$

0018-9499/$20.00 © 2006 IEEE

where i is the image voxel index, θ_i is the estimated activity value in voxel i, j is the detector-pair (projection) index, y_j is the measured value in projection j, k is the iteration number. For OSEM, the projections are grouped into S subsets s is the subset index, $j \in J_s$ represents the collection of projections in subset, s, $C_{i,j}$ is the probability of activity in voxel i being detected by projection j; the collection of $C_{i,j}$ makes the *system matrix*, and I and J are the total number of image elements and projections, respectively.

The typical gain of speed with OSEM relative to standard EM is between 10- and 100-fold. This algorithm is of considerable importance and is the primary technique used in three-dimensional (3-D) PET reconstruction. For high resolution 3-D PET systems, however, a further acceleration at this order is needed to reduce the image reconstruction time to minutes instead of hours [3], [13].

One parallelized form of OSEM is that the projections in each subset J_s are distributed over a total of N_p processors, i.e.

$$\theta_i^{k,s+1,n} = \frac{\theta_i^{k,s}}{q_i^s} \sum_{j \in J_{s,n}} \frac{C_{i,j} y_j}{\hat{y}_j^{k,s}} \tag{5}$$

and

$$\theta_i^{k,s+1} = \sum_{n=1}^{\mathrm{Np}} \theta_i^{k,s+1,n}. \tag{6}$$

This is the algorithm we used in this paper.

C. Parallel Programming Model

The message passing interface (MPI) is a library specification widely adopted as a standard for parallel computing. Its fundamental model is individual processes (each with its own address space) communicating via messages over an external network. The parallelized 3-D-OSEM above was initially implemented [15] using MPICH [20], a freely available, portable implementation of MPI [12], ported to Myrinet 2000 (Myricom, Arcadia, CA). In the following text, we refer this model as MPI or pure MPI.

In addition to MPI, OpenMP [21] is a specification for compiler directives, library routines (the OpenMP API) and environmental variables that can be used in Fortran and C/C++ programs to utilize *shared memory parallelism* on shared memory and distributed shared memory architectures. The primary advantage of using OpenMP directives lies in the ability of multiprocessors to access the same memory pool, without the costly communication overheads and network transit times found in message passing. As clusters of compute nodes with shared-memory multiprocessors (SMP) emerge as an attractive low-cost commodity off-the-shelf (COTS) platform for parallel applications, a natural mode of parallel programming on such clusters is a combination of using MPI between nodes and OpenMP within a node, for which the inherently lower overhead of the shared memory environment could be taken advantage. This mode is referred as hybrid MPI-OpenMP [29]–[32].

Fig. 1 illustrates the outline of the implementation of the parallelized OSEM reconstruction algorithm with MPI (without the lines) and hybrid MPI-OpenMP (with the lines). Note that the OpenMP directives in the framed boxes of Fig. 1 extend over the fine-grained loops of the forward and back projections. In this paper, we analyze the contributing factors to and the PE of the implementations with these two programming models.

II. Materials and Methods

A. Reconstruction Task

A miniature Derenzo phantom (Data Spectrum Corp., Chapel Hill, NC) scanned for 8 h (362 million counts) using the ATLAS [16] system was used as the reconstruction test case. The reconstructed image volume was $128 \times 128 \times 34$. The voxel size was $(0.5625\ \mathrm{mm})^3$. The number of iterations and subsets used were 8 and 9, respectively. The performance of the parallelized reconstruction algorithm used in this work in terms of image quality has been reported previously [5], [15], [22], [23]. In the current study, we focus on characterizing the contributing factors to reconstruction time and effects of parallel programming models. A parallel program visualization tool, JumpShot-4 [24], was used for doing visual postmortem performance analysis of the reconstruction program with different programming models, while the detailed performance data was obtained using the mpiP [25] MPI profiling library and the performance API (PAPI) [26].

B. Cluster of Shared-Memory Multiprocessors (SMPs)

We used a cluster of 300 SMP nodes, each with two Intel 2.4-GHz Xeon processors, interconnected by a low latency (approximately 9 microseconds using MPI), high bandwidth (about 240 MB/s, again using MPI) Myrinet 2000 network. A diagram of the cluster layout is illustrated in Fig. 2. MPI is the standard parallel API on such clusters. The architecture of this cluster reflects the growing trend of COTS high performance parallel compute clusters. The COTS approach is one in which relatively inexpensive individual units, compared to more traditional supercomputers that have more processors share memory per unit, are massed together and used for concurrent parallel computation.

C. Hybrid MPI-OpenMP and Pure MPI

The framed-lines starting with "OpenMP:" in Fig. 1 are the OpenMP compiler directives inserted for the loops which are responsible for carrying out the calculation of $\hat{y}_j^{k,s}$ in (3) and $\theta_i^{k,s+1,n}$ in (5). They instruct the compiler to parallelize the outer loop using multiple threads (private indicates the variable is not to be shared among threads) and performs a thread-level summation, with the function *reduction* $(+ : \hat{y}_j^{k,s})$, on the variable $\hat{y}$. When the OpenMP directives (framed-lines) are enabled by compilation, the hybrid MPI-OpenMP model is used for the program. The pure MPI mode is chosen when the program is compiled with the OpenMP directives disabled, or the number of OpenMP threads is restricted to one. The OpenMP implementation used is that of the Intel C/C++ compiler, version 8.1, which supports the OpenMP 2.0 Application Binary Interface (API).

The hybrid approach combines MPI for internode communication and OpenMP for parallelism within a node to take the advantages of shared memory, i.e., lower latency and higher bandwidth and more importantly, to decrease contention for internode resources. In this particular paper, we will examine in

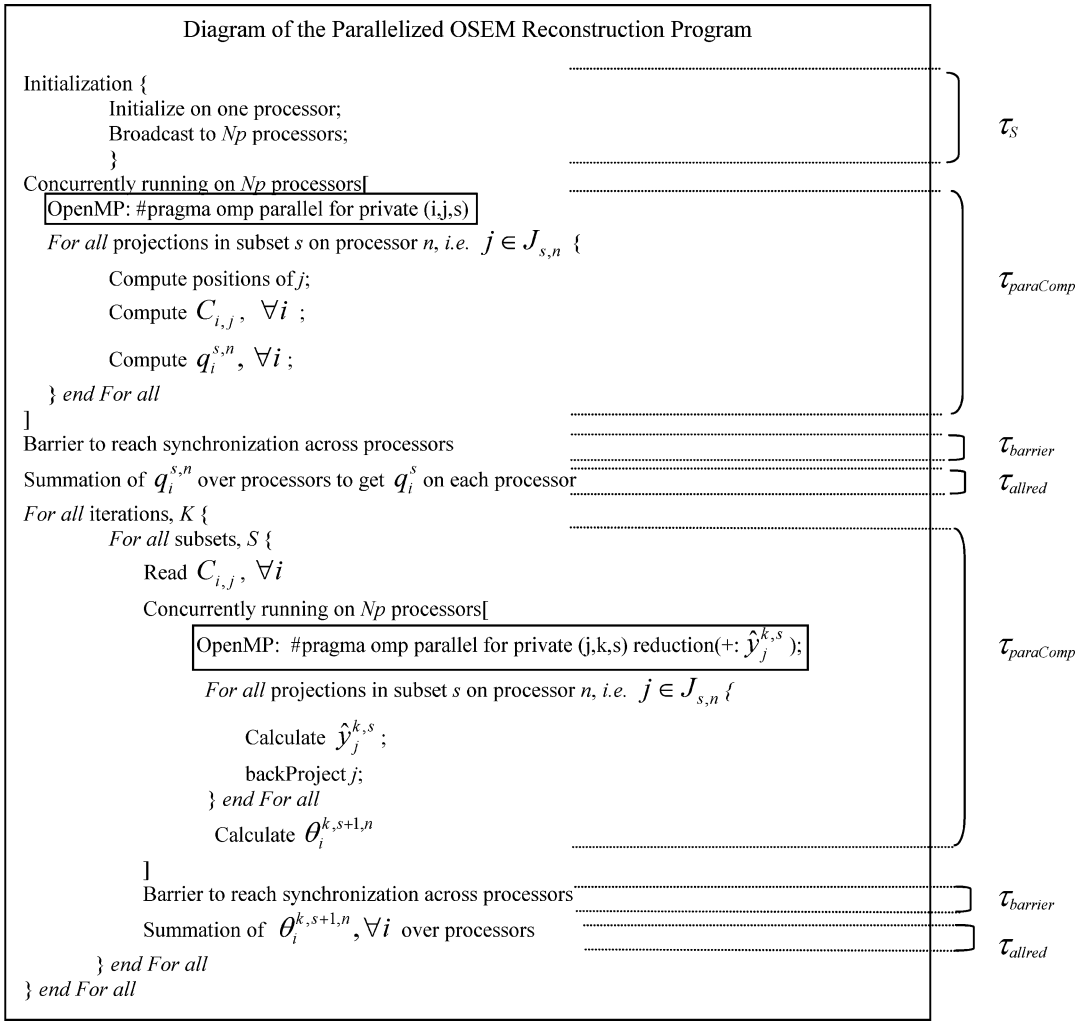

Fig. 1. A diagram illustrating the parallelized OSEM algorithm. The pure-MPI implementation is the context without the two framed lines starting with "OpenMP." The two framed lines are additions (compiler directives) for enabling OpenMP for intranode operations and, therefore, enabling the Hybrid MPI-OpenMP mode. The execution times in the program were categorized by the type of operation and labeled on the right side of the corresponding lines or blocks (see Section II-D for more details). The numbers of processors used here, N_p, are the power-of-two numbers between 1 and 64.

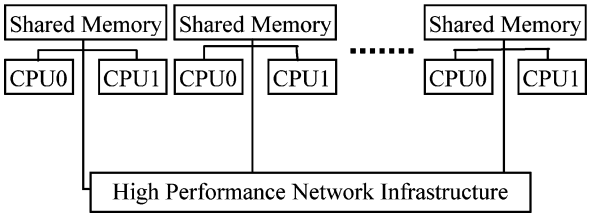

Fig. 2. Schematic layout of a cluster of two processor SMP machines interconnected with a high performance network, representative of the SMP cluster used in this study.

detail the contributions responsible for determining parallel performance, and analyze the suitability of the hybrid approach for OSEM reconstruction.

D. Performance Analysis

To evaluate the parallel efficiency, we postulate a simplified performance model. Let us denote the time spent in serial computation as τ_S, with τ_{paraComp} as the balance of time that can be parallelized. If we then define τ_{barrier} as the time required for all the processors to reach synchronization (using MPI_Barrier), and τ_{allred} as the collective exchange of information among the processors (using MPI_Allreduce), the total reconstruction time of a processor on a cluster with N_p processors is

$$\tau_{\mathrm{tot}}(N_p) = \tau_S + \tau_{\mathrm{paraComp}}(N_p) + \tau_{\mathrm{barrier}}(N_p) + \tau_{allred}(N_p) \quad (7)$$

where $\tau_{\mathrm{barrier}}+\tau_{\mathrm{allred}}$ is the load-balancing and communication overhead involved in parallel reconstruction. The parallelization speedup factor (SF), which is also the factor by which the execution time is reduced, is then given by

$$\mathrm{SF}(N_p) = \frac{\tau_{\mathrm{tot}}(1)}{\tau_{\mathrm{tot}}(N_p)} = \frac{\tau_S + \tau_{\mathrm{paraComp}}(1)}{\tau_S + \tau_{\mathrm{paraComp}}(1)/N_p + \tau_{\mathrm{barrier}}(N_p) + \tau_{\mathrm{allred}}(N_p)}. \quad (8)$$

PE is defined as

$$\mathrm{PE}(N_p) = \tau_{\mathrm{tot}}(1)/[\tau_{\mathrm{tot}}(N_p)N_p]. \tag{9}$$

It measures the efficiency of the parallel processing being used for the computation task. In an ideal setup, when the task is perfectly divided among processors with no communication overhead, $\mathrm{PE} = 1$.

The time spent in unavoidable serial computation is primarily used to set up the reconstruction: the initialization of scanner parameters, the measured data read-in, and determination and allocation of parallel workload.

In the optimistic scenario in which $\tau_{\mathrm{barrier}}(N_p)$ and $\tau_{\mathrm{allred}}(N_p)$ in (8) can be neglected, the idealized PE becomes Amdahl's Law [27]

$$\mathrm{PE}(N_p) = \frac{1}{1 + f(N_p - 1)} \tag{10}$$

where $f = \tau_S/(\tau_S + \tau_{\mathrm{paraComp}}(1))$. On large numbers of nodes, however, $\tau_{\mathrm{barrier}}(N_p)$ and $\tau_{\mathrm{allred}}(N_p)$ are not negligible, and can be a significant contributing factor relative to τ_S and $\tau_{\mathrm{paraComp}}(1)/N_p$.

Three components $\tau_{\mathrm{paraComp}}(N_p), \tau_{\mathrm{barrier}}(N_p)$, and $\tau_{\mathrm{allred}}(N_p)$ account for the execution time in parallelized mode. Since the load distribution over processes is not perfectly balanced most of the time, the τ_{paraComp} of each process varies depending on its local load and computation capacity. We, therefore, use $\mathrm{avg}(\tau_{\mathrm{paraComp}})$ and $\mathrm{std}(\tau_{\mathrm{paraComp}})$ to describe the average and standard-deviation, respectively, of τ_{paraComp} over multiple processes. The same symbols also apply to τ_{barrier} and τ_{allred}. Knowing that τ_{barrier} is the complementary component of τ_{paraComp} to reach synchronization of the parallel processes, it is intuitive to infer that the variations of τ_{paraComp} and τ_{barrier} are closely related and $\mathrm{avg}(\tau_{\mathrm{barrier}})$ correlates to $\mathrm{std}(\tau_{\mathrm{paraComp}})$.

With the specially inserted synchronization step (MPI_Barrier), the global summation operation MPI_Allreduce, which collects the values in the designated buffers belonging to each processor involved in the reconstruction task and is represented by τ_{allred}, is isolated from load-imbalance, i.e., no variation inherited from the variation of τ_{paraComp} or τ_{barrier}. The number of MPI_Allreduce calls is determined by the total image size divided by the chosen buffer size. The optimal choice of this buffer size is discussed in Section III-B.

In this paper, we compare the hybrid MPI-OpenMP model with two MPI configurations. The hybrid model consists of using one MPI task per cluster node (node, in this context, is a single computer containing two processors with associated onboard disk, memory, and network interfaces) plus two OpenMP threads. This is compared to the 'pure' MPI approach of using one MPI task per node (leaving the second processor of each node idle) and two MPI tasks per node (the more conventional method of filling available processors with MPI tasks). The pure MPI case in which we idle the second processor (hereafter referred to as MPI $\mathrm{ppn} = 1$) on each node is artificial (seldom will end users essentially throw away half of the available computational resources), but can be quite useful as a reference point, as we will discuss in Section III.

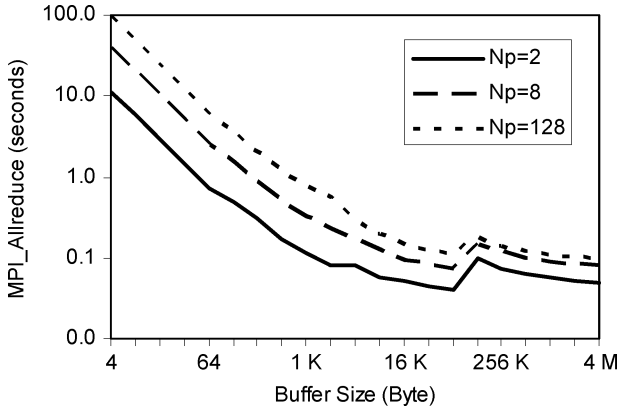

Fig. 3. MPI_Allreduce contributions to the global summation time from the Pallas MPI benchmark, as a function of buffer size on 2, 8, and 128 processors. The full size of the benchmark data set was 4 MB. Note that MPI_Allreduce reaches a minimum at a buffer size of about 64 KB. Larger buffer sizes did not further reduce this contribution.

III. Results

A. The Serial Computation Component τ_S

An analysis of a 32-processor run of the 3-D-OSEM MPI code revealed the initial setup phase of the code's execution τ_S is about 0.4% of total computation time. Although this may become a limiting factor when a large number of processors are used, we consider it to be a negligible factor for current typical computation environments ($<= 128$ processors) and make no further endeavor to optimize it in this paper.

B. The Choice of Buffer Size in Global Data Exchange

Here we consider the optimal choice of buffer size in the global data exchange of summation in (6). Fig. 3 shows the time required by MPI_Allreduce (proportional to τ_{allred}) as a function of buffer size on different number of processors (2, 8, and 128) from the Pallas MPI benchmark [28]. Other processor numbers show very similar behavior. An algorithm change was present in the MPICH global summation function for buffer sizes of 256 KB and above, which caused the slight nonmonotonic behavior for buffer sizes near 256 KB. It should be noted that this particular behavior can be dependent on the MPI implementation and even the underlying network hardware. The current implementation of the OSEM reconstruction uses a buffer size of 8 Bytes multiplied by the second and third dimensions of the image volume, which is approximately 34 KBytes. The maximum possible buffer size is the image size (in units of 8 Bytes), just over 4 Mbytes in this case. The current choice is quite close to optimal, as can be seen from Fig. 3, and only minimal gains (on the order of 5% or so) could be made by maximizing the buffer size used in global exchange.

C. Pure MPI Versus Hybrid MPI-OpenMP

The measured execution time of the three components $\mathrm{avg}(\tau_{\mathrm{paraComp}}), \mathrm{std}(\tau_{\mathrm{paraComp}}), \mathrm{avg}(\tau_{\mathrm{barrier}})$, and $\mathrm{avg}(\tau_{\mathrm{allred}})$ of the parallelized 3-D-OSEM computation task as

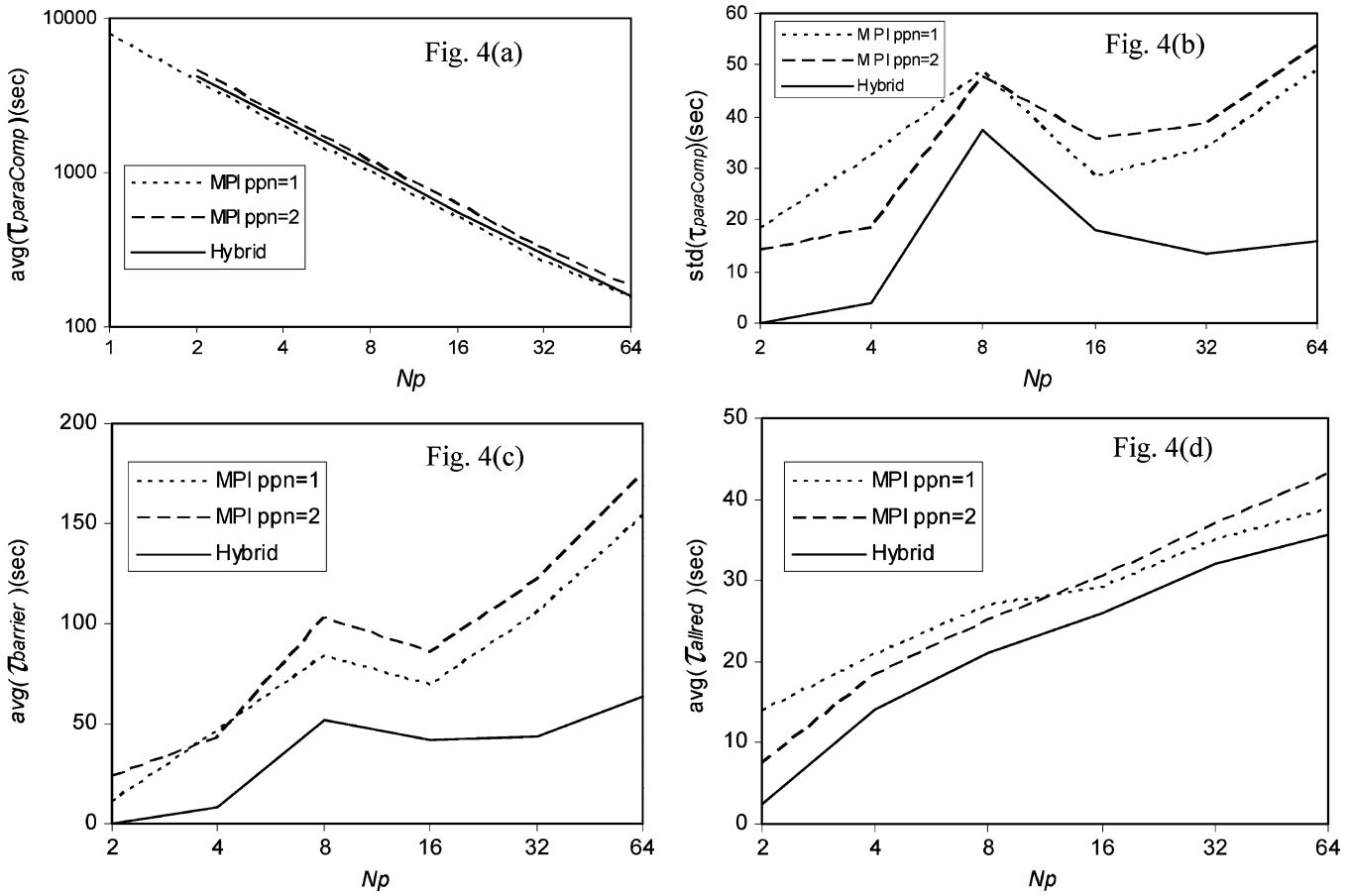

Fig. 4. The three components of the 3-D-OSEM reconstruction time τ_{paraComp}, τ_{barrier}, and τ_{allred} were measured for three modes of operation, i.e., MPI $\mathrm{ppn} = 1$, MPI $\mathrm{ppn} = 2$ and hybrid MPI-OpenMP, as a function of the total number of processors used for in the reconstruction job. The average and standard deviation of τ_{paraComp} are plotted in (a) and (b), respectively. The $\mathrm{avg}_(\tau_{\mathrm{barrier}})$ and $\mathrm{avg}_(\tau_{\mathrm{allred}})$ are shown in (c) and (d), respectively. The standard deviation plot of τ_{barrier} is very similar to the standard deviation plot of τ_{paraComp}, thereby accounting for the large variance seen in (b). The standard deviation of τ_{allred} is quite small, and is not shown in (d).

described in Section II-A, were plotted and shown in Fig. 4(a), (b), (c), and (d), respectively. The $\mathrm{std}(\tau_{\mathrm{allred}})$ is minimal and not presented. Note that $N_p = 1$ represents the serial reconstruction and, for reference, is plotted as a point of the MPI $\mathrm{ppn} = 1$ curve in Figs. 4(a) and 5(a) and (b).

The average parallel computation time over N_p processors $\mathrm{avg}(\tau_{\mathrm{paraComp}})$ was inversely proportional to the number of processors, as shown in Fig. 4(a), regardless of whether MPI or hybrid MPI-OpenMP was used. In other words, the assumption of ideal speedup for this component of the execution time is reasonable, with less than 5% deviation from linear speedup. With the same number of processors, the MPI $\mathrm{ppn} = 1$ method, having less contention for local computing resources, used less reconstruction time as compared to the MPI $\mathrm{ppn} = 2$ and the Hybrid methods. As a measure of the computational imbalance, the standard deviation of τ_{paraComp} as a function of N_p is shown in Fig. 4(b). The Hybrid method had the lowest variation as compared to the two MPI implementations. From Fig. 4(b), it appears that $N_p = 8$ happened to have a relatively higher computation imbalance than other processor numbers for this specific reconstruction task, as all three methods showed this small anomaly for $N_p = 8$. The larger variations in computation time among processors also increases the synchronization time needed and, hence, leads to higher values of τ_{barrier}. This behavior is reflected in the nonmonotonically increasing trend of τ_{barrier} in Fig. 4(c). The two measures of the communication overhead, i.e., $\mathrm{avg}(\tau_{\mathrm{barrier}})$ and $\mathrm{avg}(\tau_{\mathrm{allred}})$, in Fig. 4(c) and (d), respectively, showed a general increase with increasing numbers of processors. The $\mathrm{avg}(\tau_{\mathrm{barrier}})$ of the hybrid method was lower than both MPI modes.

As the synchronization time due to work imbalance has been accounted for by τ_{barrier}, τ_{allred} accounts purely for the communication time required for the global summation function among the processors and does not have a large variance. As shown in Fig. 4(d), $\mathrm{avg}(\tau_{\mathrm{allred}})$ of the hybrid MPI-OpenMP for N_p processors is approximately equal to the $\mathrm{avg}(\tau_{\mathrm{allred}})$ of MPI $\mathrm{ppn} = 1$ or $\mathrm{ppn} = 2$ for $N_p/2$ processors.

The overall performance, in terms of SF and PE, of the MPI and hybrid implementations as a function of N_p, is compared in Fig. 5(a) and (b), respectively. The MPI $\mathrm{ppn} = 1$ results provide a useful reference, while the MPI $\mathrm{ppn} = 2$ results are a more practical measure (in which all the available processors are utilized). The PE and SF of the hybrid approach was 7% to 17% higher than MPI $\mathrm{ppn} = 2$, as the number of processors changed from 4 to 64. Next, we discuss the implications of these results in more detail.

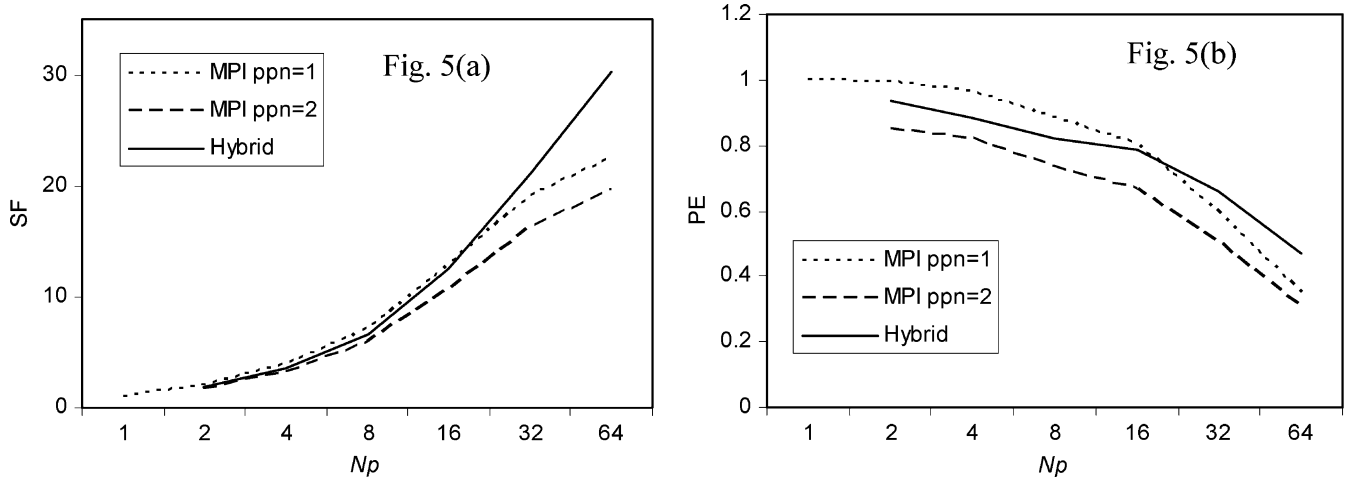

Fig. 5. Comparison of the overall performance of MPI ($\mathrm{ppn} = 1$ and $\mathrm{ppn} = 2$) and the hybrid models with different number of processors employed in the reconstruction job. The SF and PE results are shown in (a) and (b), respectively.

IV. Discussion

A. The Barrier Time τ_{barrier}

The barrier time was primarily due to load imbalance, i.e., the fundamental computation imbalance of the application among the multiple compute units, as illustrated by the correlation between Fig. 4(b) and (c). When the number of processors increases, the load imbalance does not change monotonically or linearly, hence, we see the nonmonotonic behavior of τ_{barrier} in Fig. 4(c), and representatively, at the point of $N_p = 8$.

The hybrid model with $N_p/2$ MPI processes shows a lower $\mathrm{std}(\tau_{\mathrm{paraComp}})$ and $\mathrm{avg}(\tau_{\mathrm{barrier}})$ than that of the MPI implementations with same or even lower number of processes. This is because the OpenMP approach has an intrinsic load-balancing mechanism [29] that effectively reduced the computation imbalance. The reduction would be mostly nonlinear because the distribution of projection data over multiprocesses is imaging object dependent.

B. Hybrid MPI-OpenMP

Since this model effectively reduced the number of compute units involved in communication to N_p/ppn, where ppn is the number of shared memory processors per node, based on Fig. 4(b) and (c), the time required to synchronize global data across nodes, i.e., τ_{allred}, was greatly reduced. Therefore the performance of the hybrid approach should be better, as reflected by the SF and PE results in Fig. 5(a) and (b). In other words, the hybrid MPI-OpenMP approach made better use of local node resources (disk, memory, bus) and reduced competition for network resources.

In Fig. 4(d), the MPI $\mathrm{ppn} = 1$ and $\mathrm{ppn} = 2$ curves crosses over between $N_p = 8$ and $N_p = 16$. This was likely an artifact of the particular MPI implementation used in this study (it has disappeared with a more recent MPI implementation). The impact of the artifact is small enough, as compared to τ_{barrier}, to be negligible.

The performance gains using the hybrid MPI-OpenMP approach were similar to previous studies [30]–[32], in that the relative performance advantage of the hybrid approach did depend on the algorithm under study and the network performance of the MPI implementation. The gains that we have shown here were due to the improved computing balance of shared memory and OpenMP compilers, as well as reduced internode communication.

The current trend of parallel computing favors small scale SMPs, in which manufacturing costs can be minimized to a certain extent by having several processing cores share the same fundamental infrastructure (memory, bus, i/o, etc.). Most Linux clusters in the high-performance technical computing (HPTC) market are based on this distributed shared-memory architecture. The hybrid reconstruction model is the logical fit to this architecture. According to the results of this study, a cluster of SMP nodes with more (e.g., 4 rather than 2) processors is expected to benefit further from applying the hybrid MPI-OpenMP approach.

C. OpenMP

Given the performance gains found for the hybrid MPI-OpenMP approach, an interesting question arises as to the viability of a pure SMP implementation (using either OpenMP or some other shared-memory API). While the main goal of this work was to study the speed advantage of introducing a hierarchical parallelization strategy, i.e., using OpenMP for intranode communication to complement the common MPI internode communication, a coarser-grained approach (rather than the fine-grained use of OpenMP) for the loops shown in Fig. 1 could be applied using purely OpenMP. There are other threading techniques such as Pthreads [33] available for the intranode task on a shared-memory multiprocessor node, but we have chosen OpenMP for its portability and ease of implementation [21]. The simplicity of our fine-grained approach can be seen in Fig. 1, where OpenMP is included by just adding the compiler directives without making any other changes to the original code. Using coarser-grained pure OpenMP on a shared memory architecture [34] is viable and has shown good performance in terms of PE. The downside of a pure SMP program, however, is that this approach is limited to expensive

large SMP computer architectures, and is currently too costly for many institutions.

V. Conclusion

We have studied the parallel efficiency of a 3-D OSEM algorithm implemented with a pure MPI and a hybrid MPI-OpenMP approach. Our initial motivation was to improve the efficiency beyond that of a strict message-passing approach. The hybrid MPI-OpenMP approach took advantage of the load balancing mechanism of OpenMP and the inherently lower latency of shared memory threads across processors within a node and showed a 7% to 17% improvement in terms of PE on 4 to 64 processors, as compared to the pure MPI $\mathrm{ppn} = 2$ approach. This hybrid approach is particularly important as trends towards multicore processors and larger SMPs continue to prove more cost-effective in cluster computing. While we have focused in this work on a particular application (a 3-D OSEM reconstruction code), we expect this hybrid approach to become increasingly valuable to accelerate applications with similar characteristics.

Acknowledgment

The authors would like to thank Dr. C. Johnson and Dr. J. Seidel at NIH for providing the initial 3JD reconstruction code, and the Derenzo phantom data, respectively. The authors would also like to thank the anonymous reviewers for their insightful suggestions. This work utilized the high-performance computational capabilities of the Center for Computational Research, State University of New York at Buffalo.

References

[1] R. M. Leahy and J. Y. Qi, "Statistical approaches in quantitative positron emission tomography," *Stat. Comput.*, vol. 10, pp. 147–165, 2000.

[2] W. F. Jones, L. G. Byars, and M. E. Casey, "Positron emission tomographic images and expectation maximization: A VLSI architecture for multiple iterations per second," *IEEE Trans. Nucl. Sci.*, vol. 35, pp. 620–624, 1988.

[3] S. Vollmar, C. Michel, J. T. Treffert, D. F. Newport, M. Casey, C. Knoss, K. Wienhard, X. Liu, M. Defrise, and W. D. Heiss, "HeinzelCluster: Accelerated reconstruction for FORE and OSEM3D," *Phys. Med. Biol.*, vol. 47, pp. 2651–2658, 2002.

[4] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. New York: Oxford Univ. Press, 1997.

[5] C. A. Johnson, J. Seidel, R. E. Carson, W. R. Gandler, A. Sofer, M. V. Green, and M. E. Daube-Witherspoon, "Evaluation of 3D reconstruction algorithms for a small animal PET camera," *IEEE Trans. Nucl. Sci.*, vol. 44, pp. 1303–1308, 1997.

[6] J. P. Jones, W. F. Jones, F. Kehren, D. F. Newport, J. H. Reed, M. W. Lenox, K. Baker, L. G. Byars, C. Michel, and M. E. Casey, "SPMD cluster-based parallel 3D OSEM," *IEEE Trans. Nucl. Sci.*, vol. 50, pp. 1498–1502, 2003.

[7] S. Barresi, D. Bollini, and A. Del Guerra, "Use of a transputer system for fast 3-D image reconstruction in 3-D PET," *IEEE Trans. Nucl. Sci.*, vol. 37, p. 812, 1990.

[8] C. M. Chen, S. Y. Lee, and Z. H. Cho, "Parallelization of the EM algorithm for 3-D PET image-reconstruction," *IEEE Trans. Med. Imag.*, vol. 10, pp. 513–522, 1991.

[9] C. A. Johnson, Y. C. Yan, R. E. Carson, R. L. Martino, and M. E. Daube-Witherspoon, "A system for the 3D reconstruction of retracted-septa PET data using the EM algorithm," *IEEE Trans. Nucl. Sci.*, vol. 42, pp. 1223–1227, 1995.

[10] J. L. CruzRivera, E. V. R. DiBella, D. S. Wills, T. K. Gaylord, and E. N. Glytsis, "Parallelized formulation of the maximum likelihood-expectation maximization algorithm for fine-grain message-passing architectures," *IEEE Trans. Med. Imag.*, vol. 14, pp. 758–762, 1995.

[11] K. Rajan, L. M. Patnaik, and J. Ramakrishna, "Linear-array implementation of the EM algorithm for PET image-reconstruction," *IEEE Trans. Nucl. Sci.*, vol. 42, pp. 1439–1444, 1995.

[12] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference*, 2nd ed. Cambridge, MA: MIT Press, 1998, vol. 1.

[13] D. W. Shattuck, J. Rapela, E. Asma, A. Chatziioannou, J. Qi, and R. M. Leahy, "Internet2-based 3D PET image reconstruction using a PC cluster," *Phys. Med. Biol.*, vol. 47, pp. 2785–2795, 2002.

[14] R. E. Carson, W. C. Barker, J.-S. Liow, and C. A. Johnson, "Design of a motion-compensation OSEM list-mode algorithm for resolution-recovery reconstruction for the HRRT," in *IEEE Nucl. Sci. Symp. Conf. Rec.*, 2003, pp. 3281–3285.

[15] C. A. Johnson, J. Seidel, J. J. Vaquero, J. Pascau, M. Desco, and M. V. Green, "Exact positioning for OSEM reconstructions on the ATLAS depth-of-interaction small animal scanner," *Mol. Imag. Biol.*, vol. 4, p. S22, 2002.

[16] J. Seidel, J. J. Vaquero, and M. V. Green, "Resolution uniformity and sensitivity of the NIH ATLAS small animal PET scanner: Comparison to simulated LSO scanners without depth-of-interaction capability," *IEEE Trans. Nucl. Sci.*, vol. 50, pp. 1347–1350, 2003.

[17] R. Laforest, D. Longford, S. Siegel, D. F. Newport, and J. Yap, "Performance evaluation of the microPET®—Focus F120," in *IEEE Nucl. Sci. Symp. Conf. Rec.*, 2004, pp. 2965–2969.

[18] M. Schmand, L. Eriksson, M. E. Casey, M. S. Andreaco, C. Melcher, K. Wienhard, G. Flugge, and R. Nutt, "Performance results of a new DOI detector block for a high resolution PET-LSO research tomograph HRRT," *IEEE Trans. Nucl. Sci.*, vol. 45, pp. 3000–3006, 1998.

[19] H. M. Hudson and R. S. Larkin, "Accelerated image-reconstruction using ordered subsets of projection data," *IEEE Trans. Med. Imag.*, vol. 13, pp. 601–609, 1994.

[20] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high performance, portable implementation of the MPI Message Passing Interface standard," *Parallel Comput.*, vol. 22, pp. 789–828, 1996.

[21] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, pp. 46–55, 1998.

[22] R. Yao, J. Seidel, J.-S. Liow, and M. V. Green, "Attenuation correction for the NIH ATLAS small animal PET scanner," *IEEE Trans. Nucl. Sci.*, vol. 52, pp. 664–668, 2005.

[23] R. Yao, J. Seidel, C. A. Johnson, M. E. Daube-Witherspoon, M. V. Green, and R. E. Carson, "Performance characteristics of the 3-D OSEM algorithm in the reconstruction of small animal PET images," *IEEE Trans. Med. Imag.*, vol. 19, pp. 798–804, 2000.

[24] O. Zaki, E. Lusk, W. Gropp, and D. Swider, "Towards scalable performance visualization with jumpshot," *Int. J. High. Perform. Comput. Appl.*, vol. 13, pp. 277–288, 1999.

[25] J. S. Vetter and M. O. McCracken, "Statistical scalability analysis of communication operations in distributed applications," in *Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programm.*, 2001, pp. 123–132.

[26] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *Int. J. High. Perform. Comput. Appl.*, vol. 14, pp. 189–204, 2000.

[27] L. R. Scott, T. Clark, and B. Bagheri, *Scientific Parallel Computing*. Princeton: Princeton Univ. Press, 2005, pp. 44–45.

[28] Pallas MPI Benchmarks - PMB, Part MPI-1, GmbH Pallas Pallas GmbHTech, 2001.

[29] D. S. Henty, "Performance of hybrid message-passing and shared-memory parallelism for discrete element modeling," in *Proc. Supercomput. (SC)*, 2000, pp. 10–18.

[30] L. Smith and M. Bull, "Development of mixed mode MPI/OpenMP applications," *Sci. Programm.*, vol. 9, pp. 83–98, 2001.

[31] E. Chow and D. Hysom, Assessing performance of hybrid MPI/OpenMP programs on SMP clusters Lawrence Livermore Nat. Lab., Livermore, CA, Tech. Rep. UCRL-JC-143957, 2001.

[32] G. Mahinthakumar and F. Saied, "A hybrid MPI-OpenMP implementation of an implicit finite-element code on parallel architectures," *Int. J. High. Perform. Comput. Appl.*, vol. 16, pp. 371–393, 2002.

[33] B. Nichols, D. Butler, and J. P. Farrell, *Pthreads Programming*. Sebastopol, CA: O'Reilly, 1996.

[34] J. S. Kole and F. J. Beekman, "Parallel statistical image reconstruction for cone-beam X-ray CT on a shared memory computation platform," *Phys. Med. Biol.*, vol. 50, pp. 1265–1272, 2005.