Parallel OSEM Reconstruction Algorithm for Fully 3-D SPECT on a Beowulf Cluster

Zhou Rong, Ma Tianyu, Jin Yongjie
Department of Engineering Physics, Tsinghua University, Beijing, 100084
Email: zhourong00@mails.tsinghua.edu.cn, maty@tsinghua.edu.cn

Abstract—In order to improve the computation speed of ordered subset expectation maximization (OSEM) algorithm for fully 3-D single photon emission computed tomography (SPECT) reconstruction, an experimental beowulf-type cluster was built and several parallel reconstruction schemes were described. We implemented a single-program-multipledata (SPMD) parallel 3-D OSEM reconstruction algorithm based on message passing interface (MPI) and tested it with combinations of different number of calculating processors and different size of voxel grid in reconstruction (64 × 64 × 64 and 128 × 128 × 128). Performance of parallelization was evaluated in terms of the speedup factor and parallel efficiency. This parallel implementation methodology is expected to be helpful to make fully 3-D OSEM algorithms more feasible in clinical SPECT studies.

I. INTRODUCTION

In recent years, there has been a great deal of development of iterative image reconstruction algorithms for SPECT and PET. Modern iterative image reconstruction algorithms tend to provide images in high quality and quantitative accuracy, benefiting from the development of several techniques: Firstly, physical error correction methods are incorporated in the reconstruction algorithm, including the correction of attenuation, Compton scattering and collimator blurring response. Secondly, application of full 3-D reconstruction can compensate for photon crosstalk between transaxial slices. Thirdly, there is a trend to combine multi-image modalities in single studies. All these techniques contribute to improving the diagnostic accuracy. However, computational burden is increased at the same time, which leads to the inconvenience in clinical diagnosis. For example, in [1] it is reported that fully 3-D reconstruction of a 64 × 64 × 64 SPECT image with Monte Carlo based attenuation and scatter correction requires 30 minutes, which is not yet feasible for routine clinical use.

Several approaches contribute to reducing high computational burden of full 3-D SPECT and PET reconstruction algorithms. The first one is the employment of accelerated algorithms. The second choice depends on the hardware advances in semiconductor microelectronics to provide consistent improvements in computational speed and storage capacity. Thirdly, speedup can be obtained through implementation of parallel reconstruction algorithms on multiprocessor computing systems. In this paper, we pursue the third approach.

Recently, many authors have developed parallel processing method for emission tomography. Early works described approaches based on special purpose hardware designs, including multi DSPs [2] and fine-grained SIMD machines [3]. With the rise of beowulf clusters, which are based on commodity PC hardware, private network and open source software infrastructure, attention turned to implementation of parallelized reconstruction on such systems. In [4]–[6], parallel 3-D OSEM and fourier rebinning (FORE) algorithms for PET reconstruction were investigated.

This paper investigated the implementation of a parallel fully 3-D OSEM reconstruction algorithm for SPECT imaging on a 4-node, 8-processor beowulf type cluster. Single program multiple data (SPMD) model was used based on MPI programming. To evaluate the performance of parallel reconstruction, two measures including speedup factor and efficiency were used in the comparison of different node-to-node data communicating strategies and different configuration of calculating nodes and CPUs for both 64 × 64 × 64 and 128 × 128 × 128 image reconstruction.

II. METHODOLOGY

A. Parallelization of OSEM algorithm

The OSEM algorithm according to the formulation in [7] reads

\[
\tilde{\lambda}^{k+1}(n) = \frac{\sum_{j \in S_k} c_{ij} \lambda_j^k(n) \sum_{j \in S_k} c_{ij} p_j}{\sum_{j \in S_k} c_{ij} p_j}
\]  

where \( i, j \) are indexes of image estimate \( \lambda \) and measured projection \( p \) respectively, \( n \) indicates the iteration number, and \( \lambda_j^k(n) \) represents the updated image estimate after processing subset \( k \) in \( n \)-th iteration. \( p_j \) is the projection measurements in projection bin \( j \), \( c_{ij} \) represents the probability that photon emitted from voxel \( i \) can be detected in projection bin \( j \).

There are two parallelizable steps in OSEM algorithm: calculation of \( \sum_{i} c_{ij} \lambda_j^k(n) \) in the projection step and calculation of \( \sum_{j \in S_k} c_{ij} p_j / p_j' \) in the backprojection step, where \( p_j' = \sum_{i} c_{ij} \lambda_j^k(n) \). Each step is the production of a matrix and a vector. The matrix-vector multiplication can be parallelized by dividing the matrix into strips corresponding to the vector and distributing the production over multiple processors, and a global vector sum is collected afterwards. There are several schemes for the parallelization of OSEM algorithm, which differ as to how the computational tasks are assigned to multiple processors and how the results are collected.

1) IPD/IPS: We use IPD/IPS to denote image and projection division / image and projection synchronization. In this method, firstly each processor is offered a complete copy of
the image estimate $\tilde{\lambda}^k_i(n)$, and computation of reprojection step is divided into $M$ independent parts, where $M$ is the total number of processors. The $m$-th part of the estimated projection data $\{p^j_i\}_{i \in J_m}$ is computed in the $m$-th processor:

$$p^j_i = \sum_i c_{ij} \tilde{\lambda}^k_i(n), \quad j \in J_m, m = 1, \ldots, M. \quad (2)$$

Then a global data exchange is performed to each processor a complete copy of the estimated projection $\{p^j_i\}$. Therefore computation of backprojection step can be divided into $M$ parts as well and computed in parallel similarly:

$$\tilde{\lambda}^{k+1}_i(n) = \frac{\tilde{\lambda}^k_i(n)}{\sum_{j \in S_k} c_{ij} \sum_{j \in S_k} c_{ij}} \cdot p^j_i, \quad i \in I_m, m = 1, \ldots, M. \quad (3)$$

Supposing that the normalization factor $\sum_{j \in S_k} c_{ij}$ has been calculated prior to the reconstruction, calculation of (2) and (3) can be done with no interprocessor communication and therefore a linear or near-linear speedup can be expected. However, projection synchronization procedure is needed after the parallel calculation of (2) and image synchronization is required to follow (3) to ensure that each processor has complete data of both projection and image estimate. The IPD/IPS parallel implementation of equation (1) is the most natural and simplest in programming, but it suffers from relatively high communication burden. In addition, since the distribution of computation is according to the projection data vector in the projection step and to the image data vector in the backprojection step, each processor requires the access to different values of the matrix $\{c_{ij}\}$ in different calculating steps, therefore a full storage (or full on-the-fly calculation) of matrix values is necessary to all processors, which is not computationally feasible.

2) ID/IPS: In the ID/IPS (image division / image and projection synchronization) method, the image estimate is divided into $M$ parts, which is denoted by $\{\tilde{\lambda}^k_i(n)\}_{i \in I_m}$, and the computational task assigned to each of the $M$ processors is only related to one image part. The projection step for the $m$-th processor is:

$$p^j_{i,m} = \sum_i c_{ij} \tilde{\lambda}^k_i(n), \quad i \in I_m, m = 1, \ldots, M, \quad \forall j \in S_k. \quad (4)$$

The complete projection data is obtained by a global vector sum (rather than a global vector exchange in the IPD/IPS method) over all the processors:

$$p^j_i = \sum_m p^j_{i,m}. \quad (5)$$

Afterward each processor calculates its assigned part of the new image estimate:

$$\tilde{\lambda}^{k+1}_i(n) = \frac{\tilde{\lambda}^k_i(n)}{\sum_{j \in S_k} c_{ij}} \sum_{j \in S_k} c_{ij} p^j_i, \quad i \in I_m, m = 1, \ldots, M. \quad (6)$$

The ID/IPS method requires a global vector sum of the projection data and a global vector exchange of the image data, and each processor needs only to store (or calculate) $1/M$ of the full matrix $\{c_{ij}\}$.

3) ID/PS: The ID/PS (image division / projection synchronization) method is a modified version of ID/IPS. The parallelized projection and backprojection steps and the projection synchronization are the same as (4) and (5) in the ID/IPS method; however, the two methods differ in that the procedures of global image exchange are mostly removed in the ID/PS method. One can see from (5) that given complete projection data $p^j_i$, calculation of the estimated image value for one voxel has nothing to do with that for other voxels. Therefore the image synchronization is only necessary at the end of iteration to acquire final complete reconstructed images.

**B. Assessment of parallel performance**

A performance model according to Amdahl’s law [8] is used to evaluate the efficiency of the parallel reconstruction algorithm. Assume $T(N)$ be the total computation time with $N$ processors. In the ideal case it would be expected that

$$T(N) = \frac{T(1)}{N}, \quad (7)$$

but actually not all the computation works are perfectly parallelizable. Generally each computation work can be divided into two parts: serial computation and computation that can be perfectly parallelized. Let $T_s$ be the execution time for serial computation and $T_p$ for parallelizable computation with one processor, then with $N$ processors, the total time $T(N)$ is given by

$$T(N) = T_s + \frac{T_p}{N}. \quad (8)$$

Moreover, there would be some additional works when the computation work is carried on with more than one processors. Some of them can be executed in parallel and the time cost is independent of the number of processors (e.g. the setup time spent by each processor), which is denoted by $T_{ip}$. But some other additional works may take even more time with the increase of number of processors, for example, the global vector sum or exchange required in the image and project data synchronization with all processors participating in the reconstruction. This part of computation time is represented by $T_{is}(N)$, and then (7) becomes:

$$T(N) = T_s + \frac{T_p}{N} + T_{ip} + T_{is}(N). \quad (9)$$

The speedup factor $S(N)$ of the parallel algorithm, which indicates how the execution time is reduced, is given by:

$$S(N) = \frac{T(1)}{T(N)} = \frac{T_s + T_p}{T_s + \frac{T_p}{N} + T_{ip} + T_{is}(N)}, \quad (10)$$

and the parallel efficiency $E(N)$ is defined as speed up factor over number of processors:

$$E(N) = \frac{S(N)}{N} = \frac{T(1)}{N \times T(N)}. \quad (11)$$
A beowulf type experimental cluster was built to evaluate the parallel reconstruction algorithm. The cluster consists of four nodes; each node is composed of dual 2.8 GHz Intel Xeon Nocona processors, 2 Gbytes of DDR RAM, 240 Gbytes SATA RAID0 hard disk system, and a gigabit ethernet NIC. The nodes are connected through a DELL gigabit ethernet switch.

In this study we used MPI programming library. MPI is a standard library which implements SPMD parallel programming model. in MPI programming each process has its own address space and communicates with other processes via messages. Our OSEM parallel program was written in C++ language with MPICH 1.2.6, compiled by gcc-3.4.3 and run on Redhat Linux 9.0 operating system. In particular, we used MPI_Allreduce function to implement the global vector sum and exchange. And the time is measured by MPI_Wtime function, whose accuracy is $10^{-6}$ second.

## III. RESULTS

### A. Reconstruction results

We used the Shepp-Logan phantom shown in Fig. 1. The phantom was digitized into 0.18-cm voxels in a $128 \times 128 \times 40$ 3-D image matrix. Monte Carlo simulation was performed using the SIMSET code [9] to model photon propagation in the phantom and generate the projection data. Influence of attenuation and Compton scatter was not considered. The simulated collimator was a low energy, high resolution(LEHR) parallel-hole collimator with a thickness of 30.8 mm and hexagonal holes with a hole diameter of 1.4 mm. The energy response of the camera was set to 10% and the intrinsic spatial resolution of the detector was taken to be gaussian with a FWHM of 3.8 mm. The energy window was 20% symmetric window centered at 140 keV. The radius of camera rotation was 11.52 cm. $1.9 \times 10^9$ histories of 140-keV $^{99m}$Tc photons were tracked and 120 projections (each projection is a $128 \times 128$ image) were generated over 360°. Two types of reconstruction were carried out: $64 \times 64 \times 64$ reconstruction using 60 projections over 360° collapsed to $64 \times 64$ images and $128 \times 128 \times 128$ reconstruction using all 120 $128 \times 128$ projections. Two corresponding transition matrices were produced prior to reconstruction, in which 3-D spatial response of collimator blurring effect was modelled as described in [10]. Both reconstruction results are shown in Fig 1 as well. The reconstruction results should be the same in same type of reconstruction with different configuration of calculating processors, but in fact we found that there were some very small differences in the image value between such results. We believe that the difference results from the MPI collective routines, where different order of floating point calculation may be used by MPI to achieve better parallelism when multi processors participate in the calculation, and these errors will not influence the accuracy of the results.

### B. Computation time and performance measures

The reconstruction task was executed on the experimental cluster with various configuration of number of calculating nodes and that of processors. Since each calculating node had two processors, different choices of number of calculating nodes and of processors per node might result in same number of calculating processors in all, and we found the optimized configuration of calculating nodes were not the same in different types of reconstruction task. As shown in Table I, In $64 \times 64 \times 64$ reconstruction, communication time was reduced if we made the most of processors within each calculating node and used as small number of nodes as possible, for processes within one node communicated with each other via shared memory, rather than via much slower network messages. While in $128 \times 128 \times 128$ case, since the transition matrix was much larger than that in $64 \times 64 \times 64$ case (502.5 Mbytes vs. 46.8 Mbytes), memory accesses of various processes seemed to be the main computational burden and it was better to make use of more calculating nodes and assign less processes within one node.

Fig.2 illustrate execution time for various computation components with different combination of two parallelizing methods (ID/IPS and ID/PS), and using different number of processes (the optimized number of nodes were used in all cases). Time evaluated includes execution time for disk I/O operation $T_{disk}$, calculation time $T_{cal}$, which is the sum of execution time for both projection and backprojection step, communication time $T_{comm}$ for image and projection data synchronization, and total time $T_{total}$ is the sum of the above three time parts. To a first approximation, $T_{disk}$, $T_{cal}$ and $T_{comm}$ roughly correspond to $T_s$, $T_p$ and $T_{is}$ in (8) respectively. Fig.2 shows that the ID/PS method has

---

**Fig. 1.** Slices of the Shepp Logan phantom and reconstruction results. From left to right: the original phantom, $128 \times 128 \times 128$ reconstruction result and $64 \times 64 \times 64$ reconstruction result.

**Table I**

<table>
<thead>
<tr>
<th>Type of reconstruction</th>
<th>$p=4$</th>
<th>$p=6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$64 \times 64 \times 64$</td>
<td>32.75</td>
<td>33.77</td>
</tr>
<tr>
<td>$128 \times 128 \times 128$</td>
<td>1022.13</td>
<td>938.84</td>
</tr>
</tbody>
</table>
small advantage in calculating speed over the ID/IPS method, thanks to its relatively short communication time.

Shown in Fig. 3 is the speedup factor as a function of number of processors using the ID/PS method. In comparison the ideal linear speedup factors are drawn as well, which can be thought of a fundamental limitation. Fig. 4 shows the corresponding parallel efficiency. One can see that the efficiency achieved in $128 \times 128 \times 128$ case is higher than in $64 \times 64 \times 64$ case, we believe that it is because gain in communication time $T_{cal}$ is more rapidly than that of communication time $T_{comm}$ with the increase of the size of voxel grid in reconstruction, which results in relatively smaller ratio of $T_{comm}$ to total time $T_{total}$ in $128 \times 128 \times 128$ case. On the average, the parallel efficiency descends with the increase of number of processors due to the increased communication time $T_{comm}$, but it is noteworthy that the efficiency using 6 processors is even lower than using 8 processors. This phenomenon is related to the recursive-doubling algorithm used in MPI collective routines [11], where communication in power-of-two number-of-processors cases is more efficient than in non-power-of-two cases. While making full use of the cluster (4 nodes, 8 CPUs), the resultant speedup factors are 5.6211 in $64 \times 64 \times 64$ case and 6.6602 in $128 \times 128 \times 128$ case, and the corresponding parallel efficiencies are 70.27% and 83.25% respectively.

IV. DISCUSSION

Our initial motivation was to improve the speed of fully 3-D SPECT OSEM reconstruction and make it clinically feasible. We built an experimental beowulf type cluster and implemented the parallel OSEM algorithm on it. Performance of the reconstruction algorithm with various number of processors was studied according to a parallel performance model.

However, our present parallel algorithms have not considered the problem of load imbalance in division of computational tasks, which may lead to working disparity among processors and waste of computational resources. We anticipate that once the consideration of auto load-balancing is properly addressed, our algorithm will achieve even higher efficiency.

REFERENCES