Communication Optimization and Auto Load Balancing in Parallel OSEM Algorithm for Fully 3-D SPECT Reconstruction

Ma Tianyu, Zhou Rong, Jin Yongjie

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, a parallelizing scheme of OSEM reconstruction algorithm was implemented on an experimental beowulf-type cluster and impact factors on the parallel efficiency were investigated. Two approaches were employed to improve the efficiency: (1) the communication cost was minimized via overlapping communication with computation and (2) the idle time of processes was reduced by auto load balancing. Performance of the optimized parallel algorithm was evaluated in terms of computation time, speedup factor and parallel efficiency. Improvements were observed after optimization. The efficiency was raised from 83.86% to 92.07% in fully 3-D 128 × 128 × 128 SPECT reconstruction.

I. INTRODUCTION

Iterative image reconstruction algorithms for PET and SPECT are in rapid development in recent years. Modern iterative reconstruction algorithms tend to provide images in high quality and quantitative accuracy by means of incorporating more and more accurate model of the imaging process and using fully 3-D reconstruction strategy to compensate for photon crosstalk between transaxial slices, which also contribute to the increase of computational demands. Typically, fully 3-D reconstruction of a 64 × 64 × 64 SPECT image with accurate attenuation and scatter correction (based on Monte-Carlo simulation or Klein-Nishina formula) requires 30 minutes to 2 hours [1], [2], which is not yet feasible for routine clinical use.

Parallel processing is one of the effective way to solve this problem. Previously, various parallel processing approaches for emission tomography have been presented, including using special hardware designs such as multi DSPs [3] and fine-grained SIMD machines [4]. 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 [5]–[7], parallel 3-D ordered subset expectation maximization (OSEM) and fourier rebinning (FORE) algorithms for PET reconstruction were investigated. Speedups were achieved via the parallelization, however, the performance of parallel algorithms was influenced by the extra data communicating time due to the unavoidable synchronization step of the image data between computing nodes. Impact of load imbalance on the parallel efficiency was also reported which leaded to the idle time of processes and waste of their computing abilities.

In this work, we built an experimental 4-node, 8-processor beowulf type cluster and implemented a parallel fully 3-D OSEM reconstruction algorithm for SPECT imaging on it. Several job division and data communication strategies were investigated. Further optimization of the parallel algorithm was introduced to gain higher parallel efficiency.

II. PARALLELIZATION OF OSEM ALGORITHM

We begin our discussion with the original OSEM reconstruction formula [8]:

$$\bar{\lambda}^{k+1}(n) = \frac{\bar{\lambda}^k(n)}{\sum_{j \in S_k} c_{ij} \sum_{j' \in S_k} \sum_{i} c_{ij} \lambda^k(n)}$$

(1)

where \(i, j\) are indexes of image estimate \(\lambda\) and measured projection \(p\) respectively, \(n\) is the iteration number, and \(\lambda^k(n)\) represents the updated image estimate after processing the \(k\)-th subset in the \(n\)-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 \(j\), and \(\sum_{i} c_{ij} \lambda^k(n)\) is the reprojection of the image estimate \(\lambda^k(n)\) to the projection bin \(j\).

There are two parallelizable computational steps in OSEM: calculation of \(\sum_{i} c_{ij} \lambda^k(n)\), known as the reprojec-
tion step and that of \(\sum_{j \in S_k} c_{ij} p_j / p'_j\), known as the back-projection step, where \(p'_j = \sum_{i} c_{ij} \lambda^k(n)\). Each step is the pro-
duction of a matrix and a vector. The matrix-vector multi-
plication can be parallelized by dividing the matrix into strips corresponding to the vector and distributing the production over multiple processors, and then the computational results should be collected from all the processors. 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.

A. IPD/IPS

We use the term IPD/IPS to denote image and projection division / image and projection synchronization. Supposing there are \(M\) processors in the cluster and each processor runs one computing process, first each process is offered a complete copy of the image estimate \(\lambda^k(n)\), and computation of reprojection step is divided into \(M\) independent parts. The \(m\)-th part of the estimated projection \(\{p'_j\}_{j \in J_m}\) is computed in the \(m\)-th processor:

$$p'_j = \sum_{i} c_{ij} \lambda^k(n), \quad j \in J_m, m = 1, \ldots, M.$$  

(2)

Then a global data exchange is performed to provide each processor a complete copy of the estimated projection \(\{p'_j\}\).
Afterward, computation of back-projection step can be divided into M parts as well and computed in parallel similarly:

\[ \tilde{\lambda}^{k+1}_i(n) = \frac{\tilde{\lambda}_i^k(n)}{\sum_{j \in S_k} c_{ij} p_j} \sum_{j \in S_k} c_{ij} p_j, \quad i \in I_m, m = 1, \ldots, M. \]

(3)

Supposing that the normalization factor \( \sum_{j \in S_k} c_{ij} \) has been calculated prior to reconstruction, therefore calculation of (2) and (3) can be done with no inter-processor communication and a linear or near-linear speedup can be expected. However, projection synchronization procedure is necessary after the parallel calculation of (2), and image synchronization is required as the following step of (3) to ensure that each processor has complete data of both projection estimate and image estimate. The ID/IPS parallelization is natural to imagine and simple in programming, but it may suffer from relatively high communication burden. In addition, since the job division is according to the projection data vector in the projection step and to the image data vector in the back-projection step respectively, each process requires access to different values of the transition matrix \( \{c_{ij}\} \) in different calculating steps, therefore a full storage (or full on-the-fly calculation) of matrix values is necessary for all processes, which is not computationally feasible.

B. ID/IPS

The ID/IPS (image division / image and projection synchronization) method is suggested in [6]. In this method, the image estimate is divided into M parts, which is denoted by \( \{\tilde{\lambda}_i^k(n)\}_{i \in I_m} \) and the computational task assigned to each of the M processes is only related to one part of the image. The projection step of the \( m \)-th process is:

\[ p_{j,m} = \sum_i c_{ij} \tilde{\lambda}_i^k(n), \quad i \in I_m, m = 1, \ldots, M, \quad \text{for all } j \in S_k. \]

(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:

\[ p_j' = \sum_m p_{j,m}. \]

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

\[ \tilde{\lambda}^{k+1}_i(n) = \frac{\tilde{\lambda}_i^k(n)}{\sum_{j \in S_k} c_{ij} p_j} \sum_{j \in S_k} c_{ij} p_j, \quad i \in I_m, m = 1, \ldots, M. \]

(5)

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 to store (or calculate) only \( 1/M \) of the full transition matrix \( \{c_{ij}\} \).

C. ID/PS

The ID/PS (image division / projection synchronization) method is a modified version of ID/IPS. Parallelized projection and back-projection calculation and projection data synchronization are the same as (4) and (5) in ID/IPS, but these two methods differ in that the procedures of global image exchange are removed in the ID/PS method. One can see from (5) that given complete projection data \( p_j' \), calculation of the estimated image value for one voxel has nothing to do with that for other voxels. Therefore image synchronization is only necessary once at the end of all iterations in order to acquire final complete reconstructed images.

D. PD/IS

The PD/IS (projection division / image synchronization) method is very similar to the ID/PS method, only differs in that the computational task is divided according to the projection data, and synchronization of image data is required at the end of the calculation of each subset.

III. IMPLEMENTATION AND PERFORMANCE OPTIMIZATION

We built an beowulf-type experimental cluster to evaluate the parallel reconstruction algorithm. As shown in Fig.1, The cluster have four computing nodes and a gigabit ethernet switch. Each node consists of two 2.8 GHz CPUs, 2 GB RAM, 240 GB hard disk and a gigabit ethernet NIC.

In this study we used MPI standard parallel API which implements single program, multiple data (SPMD) parallel programming model. in MPI programming each process has its own address space and communicates with other processes via messages. The parallelized OSEM reconstruction code 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 the ID/PS parallelized OSEM algorithm, projection data synchronization is required in the calculation of each
The synchronization was originally implemented using MPI_Allreduce function, which works in a blocking communicative mode. That is to say, in the communication step, after every process starts the data exchange routine and send out the data via messages, it will not return from the routine until all the other processes participating the data exchanging finish receiving the data. This mode of communication may cause waste of computing ability.

In addition, loading imbalance is unavoidable due to both the fundamental work imbalance in the application, i.e. the inherent nonuniformity of the image and projection data, and the multitasking nature of the individual cluster computers themselves. Therefore before the communication starts, additional idle procedure may occur in some processes waiting for some other processes to get data ready. As is shown in Fig. 2, blocking communication and load imbalance may both lead to decrease of parallel efficiency.

A. overlapping communication with computation

In this study, no-blocking communication in the ID/PS parallelized OSEM algorithm was implemented using MPI_Isend and MPI_Irecv functions. As shown in Fig. 2, every process starts the data exchanging routine, and then return immediately to carry out the following computational work, thus data communication and computation can be done in an overlapping way and communication time is reduced.

In the OSEM algorithm, calculations of each estimated projection pixel value in the projection step and image voxel value in the back-projection step are independent. Therefore synchronization of projection data in ID/PS (or image synchronization in PD/IS) can be splitted into several parts, and data transition of one projection part can be overlapped with computation of another projection part by means of no-blocking communication routines.

However, Fig. 2 is only a simplified illustration of the no-blocking communication program. In fact, there are two main problems to be addressed: first, the MPI_Isend and MPI_Irecv functions only implement an one-to-one communication. In order to make an all-to-all no-blocking communication like the MPI_Allreduce in the blocking communicative mode, we used a recursive doubling global data exchange algorithm [9] by calling MPI_Isend and MPI_Irecv multiple times. Second, the projection and back-projection computing steps were practically divided into several sub-steps. Since the computation of one step was independent of others, we could start the communication routine when the computation of one sub-step was completed, then returned immediately and started another computing sub-step. Therefore communication and computation could be overlapped.

B. dynamic load balancing

In order to make further improvement in parallel efficiency, a self-adaptive balancing method was applied which averaged the work load “on the fly” over all processes based on a job redivision strategy. Firstly a computing process was chosen to be the “root” process, then at the end of one iteration, each process collected the statistics of its execution time of both projection and back-projection step and sent them to the root process. The root process gathered these information from all processes (including itself), made an appraisal of the balance of the work load, and adjusted the job division strategy accordingly. Since the assigned job in each process might be changed, additional data communication time might be introduced to provide each process necessary image (or projection) data for next iteration. Our existing load balancing algorithm was based on the assumption that all the computing nodes had same computing and communicating ability. We planed to include computing ability model for every individual node and extend our load balancing algorithm to the case of inhomogeneous cluster configurations.

IV. RESULTS

A. Reconstruction results

In the reconstruction study, the Shepp-Logan phantom was used as shown in Fig. 3. Monte Carlo simulated projection data was generated using SIMSET code. Two types of reconstruction were carried out: 64 × 64 × 64 reconstruction using 60 64 × 64 projections and 128 × 128 × 128 reconstruction using 120 128 × 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 after 10 iterations are shown in Fig. 3 as well. The projection data was reconstructed with the OSEM algorithm using 10 subsets each containing 12 projections in 128 × 128 × 128 case and 6 projections in 64 × 64 × 64 case. The reconstruction results should have been the same with different configurations of calculating processes, but in fact we found that there were some very small differences in the image value between such results. We believed that the difference resulted from the MPI data collective routines, where different order of floating point calculation might be used by MPI to achieve better parallelism when different configurations of processes participated in the calculation, and these errors would not influence the accuracy of the results.

B. ID/PS parallelization

The reconstruction task was parallelized using ID/PS method and executed on the experimental cluster with various number of calculating processes. Shown in Fig. 4(a) and Fig. 4(b) are the speedup factor and the corresponding parallel efficiency as a function of number of processes. On average, the parallel efficiency descends with the increase of number of processors due to 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_Allreduce routines, where communication in power-of-two number-of-processors cases is more efficient than in non-power-of-two cases. When making full use of the cluster (4 nodes, 8 CPUs), achieved speedup factors are 5.6211 in 64 × 64 × 64 case and 6.7091 in 128 × 128 × 128.
case, and corresponding parallel efficiencies are 70.27% and 83.86% respectively.

Fig. 3. Slices of the Shepp Logan phantom and reconstruction results.

C. Performance optimization

Fig. 5 shows the execution time of projection $T_{proj}$, communication $T_{comm}$, back-projection $T_{bkpj}$ and waiting $T_{wait}$ with 8 processes after 10 iterations both without and with optimization in $128 \times 128 \times 128$ reconstruction case. One can see from Fig.5(b) that communication time and waiting time are obviously reduced, and projection and back-projection time are more close to be equal benefiting from job redivision and load balancing.

Fig.6 shows the execution time of calculation, communication and waiting which is averaged over all processes. Shown cases are without optimization, with part optimization (no-blocking communication only) and with full optimization (both no-blocking communication and dynamic load balance). The total execution time $t_{total}$, speedup factor (SF) and parallel efficiency (PE) in the above cases are given in Table. I. It can be seen from Fig.6 that communication time is shortest with only the optimization of no-blocking communication, and is slightly increased when load balancing is also adopted. This is due to additional communication burden which is necessary for gathering the waiting time information and exchanging the image data. However, reduction of the total execution time and improvement of parallel efficiency is still achieved, which benefits from dramatically decreased waiting time, and the parallel efficiency is improved from 83.86% without optimization to 92.07% with full optimization.

V. CONCLUSION AND FUTURE WORK

In this work, we implemented a parallel OSEM reconstruction algorithm for full 3-D SPECT. Parallelization strategy was discussed and performance of the parallel algorithm was optimized through overlapping communication with computation and auto load balancing. Our future work includes developing more efficient global data exchanging algorithms.
and extend load balancing method to inhomogeneous cluster configurations.

REFERENCES


