Parallel iterative cone beam CT image reconstruction on a PC cluster

Xiang Li\textsuperscript{a}, Jun Ni\textsuperscript{c,d} and Ge Wang\textsuperscript{a,b}

\textsuperscript{a}Department of Biomedical Engineering, The University of Iowa, Iowa City, IA 52242, USA
\textsuperscript{b}Department of Radiology, The University of Iowa, Iowa City, IA 52242, USA
\textsuperscript{c}Department of Computer Science, The University of Iowa, Iowa City, IA 52242, USA
\textsuperscript{d}Research Services, ITS, The University of Iowa, Iowa City, IA 52242, USA

E-mail: \{xiang-li, jun-ni, ge-wang\}@uiowa.edu

Received 30 June 2004
Revised 30 July 2004
Accepted 7 August 2004

Abstract. The iterative reconstruction (IR) algorithms are able to generate CT images with higher quality compared with the filtered back-projection method when the projection data is noisy or the radiation dose is low. IR can also be used when the data is incomplete. The major disadvantage of the IR is its high demand on computation and slow reconstruction. To improve the time performance of the IR we parallelized four representative iterative algorithms: EM, SART and their ordered subset (OS) versions on a Linux PC cluster. A micro-forward-back-projector was implemented for increased parallelization. Parameters are cached at the ray level during forward-projection to further reduce time used for back-projection. The speed-up and efficiency factors of our parallel implementation are reported.

1. Introduction

The iterative reconstruction (IR) technique is a branch of methods other than the filtered back-projection (FBP) for the computed tomography (CT). We use IR to include algorithms from both the statistical reconstruction (SR) and the algebraic reconstruction technique (ART) because they all compute the final image through a loop of steps. There are many IR algorithms available. Representative among them are the maximum likelihood (ML) expectation maximization (EM) [1–3] and the simultaneous algebraic reconstruction technique (SART) [4–6]. With ML-EM the image is obtained iteratively as an optimal estimate that maximizes the likelihood of the detection of the actual measured photons of a statistical modeling of the imaging system. The EM method can also be deterministically interpreted as the process of minimizing the I-divergence error between the estimated and measured projection data in nonnegative space [7]. With SART the algorithm repeatedly minimizes the mean square error between the estimated and measured projections in real space. Both EM and SART, together with other IR algorithms, are superior to the FBP method in terms of better image quality (contrast and resolution) under noisy projections [8], which is especially prominent in the positron emission tomography (PET) and single photon emission computed tomography (SPECT) where the noise level in the projection is high. IR is capable of modeling the PET and SPECT systems more precisely by compensation for effects of attenuation, scatter and distance dependent detector response (DDDR) [9–13]. IR can also be...
used with incomplete projection data and for metal artifacts deduction where FBP fails [14]. Modern commercial PET and SPECT scanners have begun to include IR as a standard algorithm in the software package [15].

However, the major disadvantage of the IR is its high demand on computation. For a large set of data from 3D volume scan it might take several hundreds hours for IR to run on a single computer, preventing it from real time clinical applications. The reason is that besides updating the image a single iteration of the IR needs one forward-projection and one back-projection whereas the FBP only needs one back-projection. As an example for EM often at least 30 iterations are needed. One approach to improving the time performance of the IR is to accelerate the algorithm by applying the ordered-subset (OS) concepts [16,17] to the projection data during iteration in which the projections are divided into smaller-sized groups then the IR algorithm is applied to every subset in sequence. For example, the OS-EM has reduced the reconstruction time by an order of magnitude than the EM [16]. Other efforts to speed up the IR algorithm have focused on optimizing the forward and back projection schemes, including the application of the texture rendering hardware to increase the projection calculation speed [18,19].

Another approach has been focused on the parallelization of the computation. As a matter of fact, parallel computing can also quicken the back-projection part of the FBP method. Due to the inherent nature of data parallelization in IR algorithms, parallel implementation applies to all IR algorithms. Various parallelization schemes have been proposed and they can be categorized into two types in terms of the hardware used in the system. The first type utilized a centralized parallel system such as VLSI architecture computers, vector computers, large-scale parallel computers, and shared memory multi-processor computers [20–26]. The second type of parallel computing systems employed a network of general-purpose computers and the cluster of PCs was connected by a fast local area network (LAN). As the computing technology advances dramatically many recent parallel implementations were based on the second type [27–30]. Clusters have been built on either WinNT or Unix/Linux platform and could be a hybrid of SIMD computers and MIMD system. For example [29], and [31] used a cluster of computers each with four processors. Carefully designed load-balancing and inter-process communication algorithms have also been used to optimize the parallel system’s performance [32]. With the development of the distributed computing technology, client-server and peer-to-peer network models have been used to decentralize the image reconstruction system [27,29,33]. Most of these works parallelized EM and OS-EM algorithms for PET data.

In this paper we parallelized four representative IR algorithms: EM, OS-EM, SART, and OS-SART on a Linux PC cluster. The projection data were generated from a simulation of the spiral cone-beam X-ray CT scan. A micro-forward-back-projector was implemented for increased parallelization. Parameters are cached at the ray level during forward-projection to further reduce time used for back-projection. In next section we briefly reviewed IR algorithms and introduced the parallelization scheme. The performance of our parallel implementation was reported in the third section. We concluded the paper with discussion in the last section.

2. Methods

2.1. Review of EM and SART algorithms and their ordered-subset versions

Compared with the analytic method it is easy to adapt the IR algorithms to various scan geometries and EM, OS-EM, and SART for cone-beam geometry have been reported. In our simulation of the cone-beam X-ray scan, the volume is sampled with a cube of voxels of equal size. (Figure 1) Each voxel...
has a constant attenuation factor $x_j$, $j = 1, \ldots, N$, where $N$ is the total number of voxels. Estimated projection $\hat{b}_i$, $i = 1, \ldots, M$ is the line integral along the $i$-th ray path and $b_i$, $i = 1, \ldots, M$ is the measured projection, where $M$ is the total number of rays. Let $x$ and $b$ represent the corresponding $N$-dimensional and $M$-dimensional column vector respectively; $A = (a_{ij})$ be the matrix mapping $x$ to $b$, where $a_{ij}$ measures the contribution of $x_j$ to $b_i$. We have the following linear system:

$$Ax = b$$

The CT image reconstruction problem is to find $x$. The EM algorithm is formulated as follows:

$$x_j^{(k+1)} = x_j^{(k)} + \left( \frac{M}{\sum_{i=1}^{M} a_{ij}} \left( \frac{\sum_{i=1}^{M} a_{ij} (b_i - \hat{b}_i)}{\sum_{j=1}^{N} a_{ij}} \right) \right), j = 1, \ldots, N. \quad (2)$$

The SART formula is expressed as:

$$x_j^{(k+1)} = x_j^{(k)} + \left( \frac{M}{\sum_{i=1}^{M} a_{ij}} \left( \frac{\sum_{i=1}^{M} a_{ij} (b_i - \hat{b}_i)}{\sum_{j=1}^{N} a_{ij}} \right) \right), j = 1, \ldots, N. \quad (3)$$

In the ordered-subset (OS) version of IR the projection data are divided into smaller sets $S_t$, $t = 1, 2, \ldots, L$, where $L$ is the total number of subsets. The IR algorithms are applied to each individual
Fig. 2. Use of ordered subsets for iterative reconstruction.

```
For each_iteration Do
  For each_projection Do
    forward-projection
    get-comparison
    back-projection
    update-image
  End
End
```

(a)

```
For each_iteration Do
  For each_subset Do
    For each_projection Do
      forward-projection
      get-comparison
      back-projection
      update-image
    End
  End
End
```

(b)

Fig. 3. The pseudo-code for IR (a) and its OS version (b).

subset. The intermediate reconstructed image is used as the input to the next subset reconstruction. A single iteration is defined as a loop of calculations that pass through all subsets. Figure 2 gives a schematic illustration of the ordered-subset IR. The formulation of OS-EM and OS-SART can be found in [16,34]. In this paper the ordering of subsets is simply sequential according to the original projection view’s order.

2.2. Parallel implementation of the IR algorithm

In this paper we parallelized four representative IR algorithms: EM, OS-EM, SART, and OS-SART. Other IR algorithms can be parallelized in the same manner. These algorithms can be described by the pseudo-code as Fig. 3.

The most time-consuming part of IR is the forward-projection and back-projection. The computation involved in steps of forward-projection, get-comparison, and back-projection is in a piecewise manner for all projection data. As soon as one piece of the projection data is calculated after ray-tracing along a specific path, the data can be compared with the corresponding measured projection and then be
Table 1
The time needed in seconds to reconstruct images in Fig. 5 by sequential IR algorithms and their parallel counterparts

<table>
<thead>
<tr>
<th></th>
<th>EM</th>
<th>P OS-EM</th>
<th>SART</th>
<th>P OS-SART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32 iterations</td>
<td>2 iterations</td>
<td>16 subsets</td>
<td>32 iterations</td>
</tr>
<tr>
<td>np = 1</td>
<td>1,420.28</td>
<td>120.79</td>
<td>1,428.18</td>
<td>121.11</td>
</tr>
<tr>
<td>np = 2</td>
<td>728.43</td>
<td>100.52</td>
<td>731.50</td>
<td>101.15</td>
</tr>
<tr>
<td>np = 4</td>
<td>379.89</td>
<td>98.67</td>
<td>381.96</td>
<td>99.13</td>
</tr>
<tr>
<td>np = 8</td>
<td>224.38</td>
<td>108.24</td>
<td>226.14</td>
<td>108.83</td>
</tr>
<tr>
<td>np = 16</td>
<td>162.70</td>
<td>140.66</td>
<td>163.20</td>
<td>140.45</td>
</tr>
<tr>
<td>Sequential</td>
<td>2,584</td>
<td>192</td>
<td>2,590</td>
<td>192</td>
</tr>
</tbody>
</table>

For each iteration Do
For each projection assigned to worker node Do
   micro-forward-back-projection
   update-image-on-master-node
   send-updated-image-to-all-worker-nodes
End

For each iteration Do
For each subset Do
   For each projection assigned to worker node Do
      micro-forward-back-projection
      update-image-on-master-node
   End
   send-updated-image-to-all-worker-nodes
End

(a) (b)

Fig. 4. The pseudo-code for parallel IR (a) and its OS version (b).

Back-projected. We took an object-oriented view of the abstract definition of a “ray” which allowed each ray to take care of its three actions (forward-projection, get-comparison, and back-projection) and three pieces of data (estimated projection, measured projection, and the comparison between the two). The ray was implemented as a micro-forward-back-projector and its implementation facilitated parallelization of the IR algorithm. Because the back-projection is along the same ray-path and using the same interpolation, we cached six parameters (three for ray path direction three for interpolation) during the forward-projection step. There are six direction parameters and eight interpolation weight factors in total. We find that more aggressive caching does not reduce computation time further because the cost for saving and retrieving more parameters overruns the time saved by caching. We divided projection data into groups and assigned each group to a process on the worker node. Figure 4 The worker node carried out computation of the micro-forward-back-projection. The master node gathered temporary images after back-projection and combined them to generate the updated image and then send it back to all worker nodes for next iteration of reconstruction. For OS version of IR algorithms projection data within each subset was divided and assigned among worker nodes. We used message passing model and the MPI protocol [35] for communication between master node and worker nodes.

2.3. Numerical simulation

A 3D Shepp-Logan phantom bounded by a $2.0^3$ volume of arbitrary unit was used as the object. The phantom was sampled by $128^3$ voxels of equal size. 96 projection views were measured along a spiral...
with one turn and a total translation of 1.0 along the axial direction. The source-origin distance was 5.0.
A planar detector with $128 \times 64$ cells was used which defined a cone-angle of $20^\circ$ and a fan angle of $30^\circ$.
Tri-linear interpolation was used in ray-tracing to obtain the value at sampling points from their eight neighboring voxels along the ray path. These neighboring points and associated interpolation weights were cached for a single ray so that during back-projection these parameters would not be computed again. All data were stored as 4-byte floating-point type.

The PC cluster has 14 nodes running Linux operating system with dual 1G Hz Pentium III processors and 1 GB memory. The processor cache size is 32 KB L1 and 256 KB L2. They are connected with gigabit/s switches. Figure 5 illustrates the topology of the cluster. The MPI implementation MPICH 1.2.5 [36,37] is installed as the message-passing interface among nodes. The sequential and parallel C code was compiled with gcc and mpicc, respectively.

3. Performance tests results

Figure 6 shows the reconstruction results from four parallel IR algorithms. The image is one slice of the 3D Shepp-Logan phantom at $z = -0.25$ where all ellipsoids intersect with the cross-section. Table 1 gives the time needed in seconds to reconstruct images in Fig. 6 by sequential IR algorithms and their parallel counterpart. Timing for sequential IR algorithms was taken using system clock. Timing for parallel IR algorithms was taken using MPI built-in function MPI\_Wtime(), which is often referred to as the “wall time”. The MPI function MPI\_Barrier() was called when timing started and ended to synchronize all processes. We can see that reconstruction of images with same visual appearance in parallel is a magnitude faster than in sequence for both EM and SART. For OS-EM and OS-SART the parallel implementation is approximately two times faster than their sequential counterparts. Note that the parallel reconstruction using only 1 process is still faster than the sequential reconstruction with a speed-up gain of 1.81. The reason is that the MPICH C compiler mpicc has internal optimization for multi-processor nodes.

Tables 2 and 3 give the speed up and efficacy factors for each parallel algorithm. The speed-up is defined as:

$$\text{Speed}_\text{up (np)} = \frac{\text{Time}_\text{parallel (np = 1)}}{\text{Time}_\text{Parallel (np)}} \quad (3.1)$$

The efficiency is defined as:

$$\text{Efficiency} = \frac{\text{Speed}_\text{up (np)}}{\text{np}} \quad (3.2)$$
Table 2
The speed-up factors for each parallel IR algorithms

<table>
<thead>
<tr>
<th>np</th>
<th>EM_P</th>
<th>OS-EM_P</th>
<th>SART_P</th>
<th>OS-SART_P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.95</td>
<td>1.20</td>
<td>1.95</td>
<td>1.20</td>
</tr>
<tr>
<td>4</td>
<td>3.74</td>
<td>1.22</td>
<td>3.74</td>
<td>1.22</td>
</tr>
<tr>
<td>8</td>
<td>6.33</td>
<td>1.12</td>
<td>6.32</td>
<td>1.11</td>
</tr>
<tr>
<td>16</td>
<td>8.73</td>
<td>0.859</td>
<td>8.75</td>
<td>0.862</td>
</tr>
</tbody>
</table>

Table 3
The efficiency factors for each parallel IR algorithms

<table>
<thead>
<tr>
<th>np</th>
<th>EM_P</th>
<th>OS-EM_P</th>
<th>SART_P</th>
<th>OS-SART_P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.975</td>
<td>0.60</td>
<td>0.975</td>
<td>0.60</td>
</tr>
<tr>
<td>4</td>
<td>0.935</td>
<td>0.305</td>
<td>0.935</td>
<td>0.305</td>
</tr>
<tr>
<td>8</td>
<td>0.791</td>
<td>0.140</td>
<td>0.790</td>
<td>0.139</td>
</tr>
<tr>
<td>16</td>
<td>0.546</td>
<td>0.0538</td>
<td>0.547</td>
<td>0.0538</td>
</tr>
</tbody>
</table>

Fig. 6. The reconstructed image slice at position \( z = -0.25 \). (a) EM 32 iterations. (b) OS-EM 2 iterations 16 subsets. (c) SART 32 iterations. (d) OS-SART 2 iterations 16 subsets.

where np is the number of processes. The optimal linear speed-up is obtained if Efficiency = 1. If Efficiency < \( \frac{1}{np} \), the parallel implementation actually exhibits slow down. Please note that the EM_P and SART_P have almost identical speed-up and efficiency performance. The same thing happens to OS-EM_P and OS-SART_P. Therefore in Fig. 7 only the speed-up and efficiency for EM_P and OS-EM_P are drawn. From the figure we can see that the greatest speed-up is achieved for EM and OS-EM using 16 processes and 4 processes, respectively. The sub-linear speed-up for EM_P after nodes 8 is caused by the extra communication when passing from node 3 to node 4 via switch 2. (See Fig. 5) Due to the large load of data communication resulted from ordered subset technique, the parallel OS-EM that partitions data only in the projection domain is very inefficient in terms of processes used. Optimized data partition and optimization schemes are needed to improve the efficiency and to reconstruct large volume.

4. Discussion

With the ever-increasing computing power from the commodity desktop computers a PC cluster becomes a practical tool for both research and application purposes. In this paper we parallelized four representative IR algorithms: EM, OS-EM, SART, and OS-SART with spiral cone-beam scan simulation on a Linux PC cluster to demonstrate this capability. For ordered subset versions of EM and SART we used the simple sequential-order grouping of projection data. To achieve the same image reconstruction
quality the parallel EM and SART take less than three minutes while the normal sequential reconstructions takes more than 47 minutes. The speed-up factor is not so prominent, however, for the OS-EM and OS-SART. The reason is that traditional ordered subsets technique is inherently sequential within one iteration as every subset of projections is used in turn. A direct data-parallel partition scheme is very inefficient in this manner due to the large data integration and dispatching overhead occurred at the beginning and the end of each subset iteration. This suggests a future work of the redesign of a parallel algorithm for the ordered subsets.

Acknowledgements

We appreciate helpful suggestions from Dr. Shiying Zhao with the University of Iowa Radiology Department. We would like to thank Mr. James Hunsaker for the University of Iowa ITS cluster server management. This work is supported in part by an NIH grant (R01 DC03590).
References


