Analysis of Performance Evaluation of Parallel Katsevich Algorithm for 3-D CT Image Reconstruction

Jun Ni, Junjun Deng, Hengyong Yu, Tao He, and Ge Wang
Medical Imaging High Performance Computing Lab.
Department of Radiology, The University of Iowa, Iowa City, IA 52242
{jun-ni, junjun-deng, hengyong-yu, tao-he, ge-wang}@uiowa.edu

Abstract

The first theoretically exact spiral cone-beam CT reconstruction algorithm developed was by Katsevich [1-2]. Recently, Yu et al. [3-4] implemented the algorithm numerically. Although the method is very promising, the computation is very intensive. It requires huge amount of computer time. Recently, people [5-6] began to parallelize the algorithm for achieving high performance computation. This paper presents an analysis of data decomposition and data communication in the parallel Katsevich algorithm [5] and develops an analysis expression to evaluate the performance of the algorithm parallelism. The results based on the analytical model and numerical benchmarks compared in a fare agreement. The analytical model provides a great tool to evaluate high performance computing benchmarks in the parallel Katsevich algorithms.

1. Introduction

X-ray computed tomography (CT) is an important medical-imaging modality where projection data are used to reconstruct a cross-sectional or volumetric image of a patient. Spiral cone-beam CT is one of promising technologies for 3D CT. In a spiral cone-beam CT system, a data acquisition system consisting of an X-ray tube and a multi-row detector bank rotates while the patient is moved into a scanner gantry [7]. Relative to the patient, the X-ray source scans along a helix, and generates cone beam X-rays through the object. The attenuated X-ray signals are then recorded on the detectors placed on the other side of the patient.

Although the mechanism of spiral cone beam CT seems simplistic, the cone-beam divergence and the longitudinal truncation of projection data make the exact image reconstruction far from trivial. A landmark algorithm, contributed by Feldkamp et al [8] allowed approximate reconstruction from cone-beam data collected along a circular trajectory. Wang et al. [9] primarily developed a generalized approximate the Feldkamp algorithm which is excellent in terms of efficiency and parallelism. A breakthrough was made in 2002 when Katsevich derived a filtered backprojection (FBP) algorithm that is similar to the Feldkamp algorithm but perform reconstruct images exactly [1-2]. Late, the Katsevich algorithm was numerically implemented by Yu et al. [3-4]. The computation of Katsevich algorithm requires significant amount of computer time, especially for large dataset. From the computing perspective, people began to parallelize the Katsevich algorithm by using multiprocessor systems [5-6].

A parallel computing machine can be a single Symmetric Multi-Processing (SMP) system with multiple built-in processors sharing a common memory; or a cluster of locally-connected computer processors with distributed memories; or a cluster comprising multiple workstations linked by a LAN network. In analysis of parallel computing, a processor participating in a computational process is called a processing element (PE). An overall computational task is typically partitioned into multiple sub-tasks, and the associated data is sent to different PEs through an interconnection (with an internal switch) or a networked connection (with an external switch). After the sub-tasks are completed, the results are assembled by a master PE to obtain the final result. The partition is also called parallel decomposition. Parallel decomposition algorithms influence the final performance of computing. The parallel computing technology has been successfully used in several medical applications involving image reconstruction. Many parallel algorithms have been developed. For example, Raman [10] developed a parallel Filtered-Backprojection (FBP) algorithm and implemented it on Intel Paragon system with 16 processors and the Connection Machine (CM5) system with 32 processors. The performance of their parallel FBP programs was compromised by a large communication overhead, giving a speed-up of about 4 on Paragon and 1.36 on CM5, respectively. In the early 1990s, some parallel Expectation-Maximization (EM) algorithms were proposed [11-12]. The parallel implementation was directly based on the conventional
EM algorithm with various domain partition techniques [13-14]. Ordered subset techniques were also further used to speed up the iterative reconstruction [15]. Recently, Johnson and Sofer investigated various parallelisms in image reconstruction [16]. An OSC (Order-Subset Convex)-based parallel statistical cone-beam x-ray CT algorithm was proposed based on a shared memory [17]. This algorithm employed two parallelization techniques: (1) processing all the projections within one subset in parallel (OSC-ang), and (2) dividing the whole volume into various parts and reconstructing them in parallel (OSC-vol). Both the techniques rely on re-projection/back-projection operations heavily. The second parallelization strategy is suitable for distributed memory systems. It was also found that the optimal choice of the OSC-ang and OSC-vol specifics depended on the dataset size [17].

The paradigm of using multiple parallelization techniques is effective to reduce the communication cost during data transferring.

This paper highlights a parallel Katsevich algorithm developed by the authors [5]. It focuses on the data communication and analysis of performance. The analytical results are compared with numerical one to demonstrate the validity of the analytical expression for evaluating performance benchmarks. In the following sections, the Katsevich algorithm and numerical implementation are briefly outlined. The data parallelism and communication are summarized. An analytical expression to evaluate the performance speedup is derived. The results using the analytical expression and numerical experiments are compared and discussed. A conclusion is given followed by future work proposed.

As shown in Fig. 1, a helical scanning locus C in 3-D Euclidean space $R^3$ can be mathematically described as

\[ C = \{ y(y_1, y_2, y_3) \in R^3 \} \]

where

\[ y_1 = R \cos(s), \]

\[ y_2 = R \sin(s), \]

\[ y_3 = \frac{sh}{2\pi}, s \in R^3, \]

where $s$ is an angular parameter; $h(>0)$ and $R(>0)$ are the pitch and radius of the locus. In a practical CT system, a patient is moved through the gantry, while the X-ray source rotates around the patient. Relative to the patient's position, the locus of the X-ray source can be viewed as the helix set $C$. If $S^2$ be the unit sphere in $R^3$, then the cone-beam transform of $f$ can be expressed as

\[ f(x) = \frac{1}{2\pi^2} \int_{\omega} \frac{1}{\|x - y(s)\|^2} \int_{D} D_j(y(q), \Theta(s, x, y)) \left( \frac{dy}{\sin(\lambda)} \right) ds \]

where

\[ D_j(y, \beta) := \int \int f(y + \beta \alpha) d\alpha, \beta \in S^2 \]

The detailed expresses and notations can be found in [5].

### 2.2 Numerical Implementation

As illustrated in Fig. 1, a local coordinate system on planar detector is formed to numerically implement Katsevich’s formula [1]. The cone-beam projection data is measured using planar detector arrays parallel to $d_1$ and $d_2$ at a distance $D$ from $y(s)$. Katsevich algorithm can be numerically implemented by the following two steps: a filtration called Hilbert Filtering to calculate an intermediate function $\Psi(s, u, v)$, and Weighted Backprojection to compute $f$ function in a backprojection process. The major computation is consumed in the second step. The detailed expressions and numerical results can be found in [3-5].

### 3. Data parallelism and implementation

Since the accomplishment of a 3-D image reconstruction requires a great amount of time, the authors designed the algorithm’s parallelism using data decomposition.

As described above, the two major computation procedures are required to accomplish Katsevich algorithm for medical image reconstruction: filtration and back-projection. In filtration, the major computation time is used to calculate various numerical differentiations. The projection data from different views (say $n$ views) with corresponding angles can be distributed to multiple PEs and to be processed in parallel. The computation of the derivatives requires the projection data be partitioned in such a way that the data from several view angles be

---

**Figure 1 Coordinate systems and variables used for image reconstruction in the case of helical cone-beam CT.**

2. Katsevich Algorithm and its numerical implementation

2.1 Katsevich theorem
sent to the same PE for instructions. Therefore, how to decompose projection data be distributed to each PE is a key issue. In our design, each PE processor gains equal amount of projection data based on its computing capacity. Since the PC cluster is a homogenous system, the projection data are just partitioned evenly, as shown in the Fig. 2. If a system is heterogeneous, one can decompose projection data with a consideration of load balance.

The backprojection process, is a voxel-driven formulation. The reconstruction of each voxel $x$ can be independently performed, requiring the large amount of computation. Therefore, the volume is also partitioned over the PEs consistent to their processing capabilities. Each PE reconstructs corresponding voxels, as shown also in Fig. 2.

The overall parallel computing is processed in the following order. The projection data is first partitioned and distributed over all the PEs. After each PE receives its assigned projection data, it independently performs the filtering operation. Once each PE accomplishes its filtering operation, it sends the filtered data to all the PEs. Once each PE received all filtered data from other PEs, it independently performs intensive backprojection on partitioned voxels. Finally, the backprojected data are collected on a master PE which gives out the final reconstructed image.

4. Parallel Experiments and numerical results

4.1. System hardware and software

The parallel Katsevich algorithm was implemented on a Microway 64-bit AMD-based Opteron HPC cluster with 16 nodes with 16 nodes. Each node has two processors (PEs) and 4 GB memory. The system is located at Medical Imaging High Performance Computing Lab (MIHPC Lab) of the University of Iowa. The total system storage is 8 TB for archiving and retrieval of high-resolution images. The program is in C, compiled by the Portland C compiler with Message Passing Library (MPI). The program invokes MPI functions, such as sending, receiving, broadcasting, and collecting.

4.2. Data preparation

The parallel implementation of the Katsevich algorithm was evaluated by reconstructing the 3-D Shepp-Logan phantom [17]. The spiral cone-beam projection data was collected with a planar detector, as shown in Fig. 1. Different datasets (volumes of $128^3$, $256^3$, $384^3$ and $512^3$) were used to measure the performance (mainly speedup and efficiency) and to study the effects of data size, data transfer rate etc. The double precision format was used for expressing the projection data and image data.

4.3. Results of parallel computations

The measured computational time for volumes of $128^3$, $256^3$, $384^3$ and $512^3$ voxels, (Case I, Case II, Case III, and Case IV) respectively, are plotted in Fig. 3(a). The detailed data are available in [5]. The results show that the reconstruction time significantly decreases with the increment in the number of PEs. The benchmarks of a parallel algorithm are given in terms of a speed-up $S_p = T_o / T_p$ and a parallel efficiency $\eta = S_p / n_p$, where $n_p$ (or N) is the number of processors, $T_o$ is the total execution time when one processor is used, $T_p$ is the total parallel execution time when $N$ processors are used. The speed-up was calculated and plotted in Fig. 3(b) with the number of processors in each of the four datasets. The parallel efficiencies in the Cases I, II, III and IV with respect to the number of processors are plotted in Fig. 3(c). It is noticed that the efficiency curve for the first case stays below the ideal efficiency curve and decreases relatively rapidly, whereas the curves for the other cases descend slowly and are close to the ideal efficiency curve. All the X-axes represent the number of processors. The Y-axes of (a), (b), (c), and (d) are for computational time, speedup, efficiency, and ratio of the communication time to the computation time, respectively.
Figure 3. Comparisons of the performance parameters for the parallel Katsevich algorithm.

In addition, the efficiency curves for the latter cases show a common wavy pattern, in which the efficiency decreases first, then increases and finally decreases again. In the region 1, where the number of PE ranges from 1 to 5, the parallel efficiencies for these cases decrease. In the region 2, the efficiencies increase with increment in the number of PEs. The curves reach their peaks when the number of PEs is about 16. In the region 3, also called the post-peak performance region, the efficiencies decrease again as the number of PEs further increases. The appearance of the super-linear effect (the behavior in which the speed-up is greater than the ideal linear speed-up) is due to the fact that in the multiprocessor system the memory usage associated with each PE is less than that in the single processor system [18]. For example, during the backprojection process, each processor reconstructs a portion of the object, thus allocating only that portion of memory. In the case IV, to reconstruct an object into a volume of $512^3$, at least $512^3 \times 8$ bytes-1GB memory is needed for backprojection in a single-processor system. While in a multiprocessor system where $n$ ($n>1$) PEs are used, the memory associated with each processor is $1/n$ of the memory (1GB). The impact of memory on the computational ability of PEs is responsible for the super-linear speed-up. Such phenomena are more evident with a larger dataset. That is why it appears more prominently in the cases II through IV.

The communication time constitutes a smaller percentage of the total reconstruction time as the reconstruction volume becomes larger. Hence, the parallel algorithm will be computationally more efficient when a large dataset is dealt with for higher resolution reconstruction. The ratio between the communication and computation time corresponding to different numbers of processors is also plotted in Fig. 3(d). It shows that as the size of a dataset increases this ratio decreases, resulting in a higher performance.

4.4. Validity Comparison

Finally, to verify the correctness of the current parallel implementation, the selected slices of the reconstructed objects are compared with the corresponding slices of the 3-D Shepp-Logan phantom. An excellent agreement can be seen in Fig. 4.

5. Analytical study of parallel performance

The speedup of the parallel Katsevich algorithm can be analytically evaluated through a model. The model can be used to predict benchmark performance in terms of speedup and efficiency without conducting computations, and can be used to design a parallel CT reconstruction system. It has of great value to medical imaging reconstruction using parallel the Katsevich algorithm. A model can be derived based on the workflow of parallel data communications and various computations.

Let set $m$ the total pixel size on a detector plane (in Fig. 5). The $m$ stands for partial amount of projection data on a detector plane. The $m$ value is determined by a CT scanner. If one collects $n$ perspective views of CT projection (see in Fig. 5), the total projection data size $P$ is equal to the product of $m$ and $n$ (i.e., $P=m*n$). The projection data unit is pixel. If one desires to generate a
volume of reconstruction object in voxels, the value can be denoted as \( V \).

Figure 4. Representative slices of reconstructed 256^3 volume. The top row shows the reconstructed slices of the 3D Shepp-Logan phantom, while the bottom reveals the differences between the reconstructed and original slices. The gray ranges are [1.00, 1.05] and [-0.05 0.05] for the reconstructed slices and the differences, respectively.

Figure 5. Image data size and decomposition

At the initial stage, there is a time need to prepare initiation of the program, input or load projection data, dynamic allocation of memory etc. on a master node. It can be expressed as

\[
T_{ini} = T_{sys} + T_{kpi} + T_{alloc} + \ldots
\]

(3)

Usually, this amount of time is negligibly small compare with other communication times and computation times.

Once the projection data are prepared, the master begins to partition the projection data and transfer the data to each computational node. There are \( N \) processors that participate a parallel computation task. If the total projection data \((n*m)\) is decomposed evenly on a homogeneous HPC system, each PE receives \((nm)/N+2m\), since the upper or lower layers need information of the neighboring ones for calculating derivatives [3-5]. Therefore, each PE needs average communication time as

\[
T_{comm} = \left\{ \begin{array}{l}
T_{latency} + \frac{(nm)}{N} + mT_{transfer} \\
(PE_i, i = 2, 3, 4, \ldots N - 1)
\end{array} \right.
\]

(4)

\[
T_{comm} = \left\{ \begin{array}{l}
T_{latency} + \frac{(nm)}{N} + 2mT_{transfer} \\
(PE_i, i = 1, N)
\end{array} \right.
\]

where \( T_{latency} \) is the time holding for prepare and initiate data transfer. It depends on network latency. For simplicity with lose important information, one can approximately use the following expression to estimate \( T_{comm} \). The first data communication time can be expressed as

\[
T_{comm} = T_{latency} + \frac{(nm)}{N} + mT_{transfer}
\]

(5)

This is a time required to accomplish data communications between the master node to the working nodes. The \( t_{transfer} \) is the time used to transfer a byte between two nodes (master and working nodes). It can be determined by the local or network bandwidth in unit of byte per second. For example, a Gagabit switch offers bandwidth (BW) 128M byte/s data transfer rate. Therefore typically, \( t_{transfer}=1/BW \) is about \( 10^{-8} \) (second).

If \( t_{filter} \) is the computational time required to filter a single projection data, the total computational time for filtration process on each processor can be approximately calculated as

\[
T_{filter} = \frac{(nm)}{N} t_{filter} (PE_i, i = 1, 2, 3, 4, \ldots N)
\]

(6)

Let's evaluate the second data communication that is needed to pass filtered data from each single processor to all other processors. If \( T_{comm} \) stands for the second part of data communication time used for all the nodes to gain filtered projection data, it can be estimated as

\[
T_{comm} = T_{latency} + \frac{(nm)}{N} t_{transfer}
\]

(7)

If \( t_{bp} \) stands for the time required accomplish a “backprojection” for a single voxel, the total backprojection computational time \( T_{comp} \) used for backprojection on the whole voxels can be estimated as

\[
T_{bp} = \frac{V}{k} t_{bp} = kP \frac{V}{N} t_{bp} = \frac{k(nm)}{N} t_{bp}
\]

(8)

where \( k \) is defined as the ratio of total image volume to total projection data, \( k=V/P \) (in unit of voxel/pixel). Finally, a communication is needed to transfer backprojected data to the master node for assembly the final reconstructed image. It can be calculated as

\[
T_{comm} = T_{latency} + V t_{transfer} = T_{latency} + kP t_{transfer}
\]

(9)

Therefore, the total parallel time using \( N \) processors can be evaluated using the following expression
\[ T_{\text{parallel}} = T_{\text{comm}} + T_{\text{filtration}} + T_{\text{comm}} + T_{\text{backprojection}} + T_{\text{output}} \]  \hspace{1cm} (10)

Since the total sequential computation time is
\[ T_{\text{sequential}} = T_{\text{init}} + P_{\text{filter}} + V_{\text{bp}} + T_{\text{output}} \]
\[ = T_{\text{init}} + (nm)f_{\text{filter}} + k(nm)t_{\text{tp}} + T_{\text{output}} \]
Thus, the speedup can be expressed as
\[ S_n = \frac{T_{\text{sequential}}}{T_{\text{parallel}}} = \frac{T_{\text{init}} + (nm)f_{\text{filter}} + k(nm) + T_{\text{output}}}{T_{\text{comm}} + T_{\text{瀚滤}} + T_{\text{瀚瀚}} + T_{\text{瀚瀚}}} \]
\[ a = \frac{T_{\text{瀚瀚}} + (nm) + (nm) + k(nm) + T_{\text{瀚瀚}}}{T_{\text{瀚瀚}} + (nm) + (nm) + k(nm) + T_{\text{瀚瀚}}} \]
\[ b = \frac{T_{\text{瀚瀚}} + T_{\text{瀚瀚}} + (nm) + T_{\text{瀚瀚}}}{T_{\text{瀚瀚}} + T_{\text{瀚瀚}} + (nm) + T_{\text{瀚瀚}}} \]
\[ c = \frac{T_{\text{瀚瀚}} + 3T_{\text{瀚瀚}} + (n+m) + T_{\text{瀚瀚}}}{T_{\text{瀚瀚}} + 3T_{\text{瀚瀚}} + (n+m) + T_{\text{瀚瀚}}} \]
\[ \frac{a}{c} \]
\[ \frac{b}{c} \]
\[ \frac{c}{c} \]
If one of the values of \( T_{\text{瀚瀚}} \), \( T_{\text{瀚瀚}} \) and \( T_{\text{瀚瀚}} \) are negligibly small, as usual, the above equation yields
\[ S_n = \frac{t_{\text{瀚瀚}} + k t_{\text{瀚瀚}}}{(2n+k) t_{\text{瀚瀚}} + 1 n t_{\text{瀚瀚}} + k n t_{\text{瀚瀚}}} \] \hspace{1cm} (13)

Usually, the \( n \) is very large and order magnitude of 1,000-10,000, the second term in the denominator is a very small value, which can be neglected. The above expression reduces to
\[ S_n = \frac{t_{\text{瀚瀚}} + k t_{\text{瀚瀚}}}{(2+k) t_{\text{瀚瀚}} + 1 n t_{\text{瀚瀚}} + k n t_{\text{瀚瀚}}} \] \hspace{1cm} (14)

Or
\[ S_n = \frac{t_{\text{瀚瀚}} + k t_{\text{瀚瀚}}}{(2+k) t_{\text{瀚瀚}} + 1 n t_{\text{瀚瀚}} + k n t_{\text{瀚瀚}}} \] \hspace{1cm} (15)

where
\[ \lambda = \frac{t_{\text{瀚瀚}}}{t_{\text{瀚瀚}} + k t_{\text{瀚瀚}}} \] \hspace{1cm} (16)

Equation (16) gives a precise model to evaluate the parallel performance of the algorithm. The value \( \lambda \) is a correlation factor that is determined by many independent variables. Through equation (16), one can discuss many influences on the speedup. The discussion is depicted as follow.

5.1 Influence by number of processors, \( N \)

It is very clear that the speedup is mainly determined by the number of processors, \( N \), in a linear relationship, corrected by the \( \lambda \) factor, if the \( k \) value is small, of if the byte transfer rate is small due to a fast bandwidth of intern-connection or networking. This explains why when \( N \) is small, the speedup appears linearly. When \( N \) increases, the speedup up slow down. If \( N \) becomes a large number, for given \( k \), filtering and backprojection algorithms, and bandwidth, the speedup reaches a limitation, i.e.
\[ S_n = \frac{t_{\text{瀚瀚}}}{k t_{\text{瀚瀚}}} \] \hspace{1cm} (17)

This phenomenon can be found in our experiments of image constructions on USA National TeraGrid, when we deployed 800 processors. When the \( N \) reaches to a value of 200 processors, the speedup reaches to the \( S_n \) which is about 300. In the current study, for the cases of reconstruction volumes \( V=128^3 \) and \( 256^3 \), the total projection data \( P=3501*300*40 \) (m=300*40, n=3501), while for the case of reconstruction volume is \( 512^3 \), the total projection data \( P=7001*500*70 \) (m=500*70, n=7001). Therefore, the ratios \( k \) are 0.049918, 0.39934, and 0.547827, respectively. Since the \( t_{\text{瀚瀚}} \) is proposal to the number of projection views, \( n \), The ratios of \( t_{\text{瀚瀚}} = t_{\text{瀚瀚}} |_{512^3} / t_{\text{瀚瀚}} |_{256^3} \propto (n |_{512^3} / n |_{256^3}) \]
\[ =7000/3500 = 2, \] and the value filtration \( t_{\text{瀚瀚}} \) comparing with \( t_{\text{瀚瀚}} \), is relatively small, the ratio of \( S_n \) in \( 312^3 \) case to the one in \( 256^3 \) can be estimated as the ratio of \( t_{\text{瀚瀚}} \), which is double by the factor of two. That can be exactly seen in Fig. 6. One can use the relation to predict the size effects of high performance results when dealing with large dataset.

5.2 Influence by bandwidth

The correction factor is determined by network bandwidth (in unit of bits/second), the ratios of image volume vs. the total projection data size, filtering time and backprojection time for each projection data and volume, respectively.

It is very interesting to see that the projection data described by \( m \) and \( n \) has no direct influence on the speedup. However, the speedup is influenced by the \( k \)
value (ratio of final image volume vs. size projection data), time for filerings and time for backup.

The $t_{\text{transfer}}$ is determined by the inverse of the bandwidth, i.e., $1/\text{bandwidth}$. Therefore, the bandwidth of network among processors is a very important factor which strongly influences the speedup. If the bandwidth is lower, the speedup will be decreased. If one has fast bandwidth switch or network, the performance will be improved. If the bandwidth increases, the $\lambda$ value decreases, and, the speedup increases. In a special parallel system, a bandwidth is more or less fixed. For example, in our study, we use NetGear Gaga switch and bandwidth is about $10^{-8}$-$10^{-7}$ second, if the network latency is large, it definitely influences the final parallel performance. That is why in the parallel experiments, the results are so varying for each test, even for an interconnection.

6. Discussions and Conclusions

Some concerns need to be addressed here. One may argue why not to start the 3-D backprojection as soon as some PEs finish their filtering task so as to avoid waiting for the others. Theoretically, it is feasible. However, there are three reasons that make it unnecessary or impractical. First, the time needed for the filtering constitutes only a small portion of the total reconstruction time. Second, in our homogeneous system, the computation load is balanced among PEs in the filtering step, which can be observed from the timeline, hence the PEs finish the filtering task at almost the same time. Third, the programming complexity would be increased if we had done that way. Nevertheless, it is admitted that, in a heterogeneous system where the computation load of PEs is imbalanced, one need consider starting backprojection asynchronously.

Another concern is that, after the filtration, it seems not economic in terms of data communication and memory storage to send a full copy of the filtered data to every involved PE. An alternative solution is to search for all the PI-segment and the affiliated projection data immediately after the filtration, gather and distribute only the necessary data, and finally do the backprojection. It could reduce the needed data storage and communication among PEs, but it would demand more computing resources to compute and store the endpoints of the PI-segments and so on. Therefore, the tradeoff between them should be optimized.

Both these concerns suggest that the parallel implementation of the Katsevich algorithm is not unique. Since this work is to demonstrate the feasibility and advantages utilizing the parallelism, more implementation options are not discussed here. It is also worth noting that the proposed parallel computing structure can definitely be adapted for many other CB-FBP algorithms, such as those described in [19-21].

In conclusion, the parallel Katsevich algorithm for 3-D CT has been designed and studied. Our algorithm distributes the projection data and image sub-volumes to multiple PEs consistent to their computing abilities. It is feasible to modify the partitioning scheme when PEs are not identical or more PEs are used. Future work includes studies on the trend of speed-up and efficiency curves when more PEs are used, and on the impact of enlarging the image volume on the speedup and efficiency of the parallel computing system.
8. Acknowledgement

The project is partially supported by USA National Health Institute (NIH/NIBIB) grants EB001685 and EB002667. The authors are grateful for the anonymous reviewers who made constructive comments. The authors thank Mr. Boyd Knosp from Academic Technology-Research Services, and Information Technology Services at the University of Iowa for his generous support. We also thank Ms. Diane Machatka, Manager in Facility Management at the University of Iowa for contributing legacy Gateway Desktop PCs to build a test-bed PC cluster.

9. Reference


