# PARALLEL IMPLEMENTATION OF CONE BEAM TOMOGRAPHY

David A. Reimann[1,2], Vipin Chaudhary[3], Michael J. Flynn[1], and Ishwar K. Sethi[2]
[1] Department of Diagnostic Radiology, Henry Ford Health System
[2] Department of Computer Science, Wayne State University
[3] Department of Electrical and Computer Engineering, Wayne State University
Detroit, Michigan 48202

Abstract — *Three dimensional computed tomography is a computationally intensive procedure, requiring large amounts of RAM and processing power. Parallel methods for two dimensional computed tomography have been studied, but not extended or applied to 3D cone beam tomography. A discussion on how the 3D cone beam tomography problem can be implemented in parallel using MPI is given. We show an improvement in performance from 58.2% to 81.8% processor utilization in a heterogeneous cluster of workstations by load balancing. A 96.8% efficiency was seen using a 2 processor SMP.*

## INTRODUCTION

Tomography Problem

Tomographic reconstruction from projections using computed tomography (CT) is the noninvasive measure of structure from external measurements. With 3D CT, a set of 2D planar projections of the object are efficiently acquired at various positions around the object. In x-ray transmission CT, the x-rays are emitted from a point source in a conical shaped beam and collected with a 2D detector. Cone beam microtomography systems are limited primarily by the CCD camera transfer rate and available x-ray flux with acquisition times less than 1 hour [1]. New diagnostic medical imaging systems can acquire cone beam data in under 10 seconds. Previous results indicate reconstruction times of 12 hours for a $512 \times 512 \times 128$ volume [1]. Our goal is to reduce the disparity between acquisition and reconstruction times.

Cone Beam Reconstruction Algorithm

The reconstruction for the cone beam geometry has been investigated by numerous groups. The most efficient algorithm in use is the one developed by Feldkamp [2]. The analytic Feldkamp algorithm can be summarized as follows:

A. Weight projection data

$$P'_\theta(u,v) = \frac{d_s}{\sqrt{d_s^2 + v^2 + u^2}} P_\theta(u,v) \qquad (1)$$

B. Convolve weighted projection data

$$P^*_\theta(u,v) = P'_\theta(u,v) * h(u) \qquad (2)$$

C. Backproject $P^*_\theta(u,v)$

$$\mu(x,y,z) = \int_0^{2\pi} \frac{d_s^2}{(d_s - y)^2} P^*_\theta\left(\frac{d_s x}{d_s - y}, \frac{d_s z}{d_s - y}\right) d\theta \qquad (3)$$

It is assumed that the projection of the object at angle $\theta$, $P_\theta()$, is indexed by detector coordinates $u$ and $v$. The center of rotation is the $z$ axis. The distance from the x-ray focal spot to the rotation axis is $d_s$. It is assumed that a modified geometry with the detector containing the $z$ axis is used ($d_d = d_s$). By scaling the projection pixel sizes, the vertical axis of the detector can be moved to the $z$ axis. By subsequently scaling the geometry, the pixel sizes at the $z$ axis can be made 1. These scalings simplify the reconstruction algorithm.

The problem is discretized by turning the integral into a summation and making the volume and projections discrete. The reconstructed volume $\mu$, is indexed by physical coordinates $x$, $y$, and $z$ and contains $N_x \times N_y \times N_z$ voxels. The projections are $N_u \times N_v$ pixels.

Computational Analysis

The required number of floating point additions and multiplications is

$$F = 7N_uN_v + N_\theta[N_v(N_u(18\log_2(2N_u)) + 3(N_yN_x(9 + 17N_z))]. \qquad (4)$$

As the number of voxels increases, the number of angular views must also increase to maintain the same peripheral resolution. This is an important factor in larger reconstructions. For the case of $N = N_x = N_y = N_z = N_u = N_v$, the number of projections $N_\theta$ should be $\frac{\pi}{2}N$, and thus the complexity of the equation 4 is $O(N^4)$. To reconstruct a $512 \times 512 \times 512$ image volume from 800 projections with $512 \times 512$ pixels each requires roughly $1.87 \times 10^{12}$ floating point operations. Furthermore, 512 MB of RAM is required to keep the entire reconstruction volume resident in memory assuming the use of 32 bit voxels.

## PARALLEL ALGORITHMS FOR CT

In filtered backprojection, the majority of the computation is in the backprojection step. 98% of



0190-3918/96 $5.00 © 1996 IEEE

the operations are involved in backprojection when reconstructing a $512^3$ volume from 800 $512^2$ projections. This suggests effort invested in backprojection will be most fruitful. However, the large data sizes warrant careful algorithm selection to minimize communications and RAM requirements.

Previous Work

Several dedicated hardware implementations have been proposed. It is doubtful if any of these systems are currently in use, either because of newer hardware solutions, or the hardware is not yet fully integrated. Furthermore, none of these systems was developed for cone beam tomography. Another drawback to a dedicated hardware system is the relatively low market demand and the lack of scalability of such a system. However, a better understanding of the interplay between communications, processing power, and memory should be applicable to hardware implementations.

Another active area of research has been in the area of employing commercially available parallel systems to the reconstruction problem. Parallel efficiencies of greater than 88% have been reported [3]. These systems require a large investment compared with general purpose workstations.

None of this prior literature dealt specifically with large volume tomographic reconstruction from cone beam projections. In the following section the voxel driven approach for cone beam CT is further described. We argue that this method can minimize communications on some architectures.

Voxel Driven Parallel Algorithms

Four forms of parallelism were defined by Nowinski [4]: pixel parallelism, projection parallelism, ray parallelism, and operation parallelism. Pixel parallelism uses the fact that all pixels are independent of others. Projection parallelism uses independence among projections. Ray parallelism notes that rays can be backprojected independently. Operation parallelism can be used when low level operations such as multiplications and additions can be computed in parallel.

Pixel parallelism can be extended to voxel parallelism. A voxel driven approach may be taken where the volume is distributed over several PE's and each view is sent to the voxel PE's as shown in Figure 1. Each PE gets every projection, but only a small subset of the reconstructed voxels. The total memory required for this implementation is approximately equal to the total number of voxels. One advantage of this method is that the data is acquired in a serial fashion and backprojecting could be done in concert with acquisition. This type of parallelism was found to be superior [5].

The problem has a high degree of independent operations, and should be well suited to parallel implementations. The major nontrivial aspects of the problem is the data sizes needed. An acquisition can be as large as 800 views of $512^2$ 16 bit integer images. These projectional images are used to create a volumetric data set on the order of $512 \times 512 \times 512$ 32 bit floating point voxels. The use of 32 bit floating point voxels is required to provide sufficient accuracy in the result. The memory requirement to store the entire $512 \times 512 \times 512$ 32 bit volume is 512 MB, which is at the upper limit of available memory on most tightly coupled parallel systems currently available. However, dividing this memory among 16 workstations requires only 32 MB per workstation.

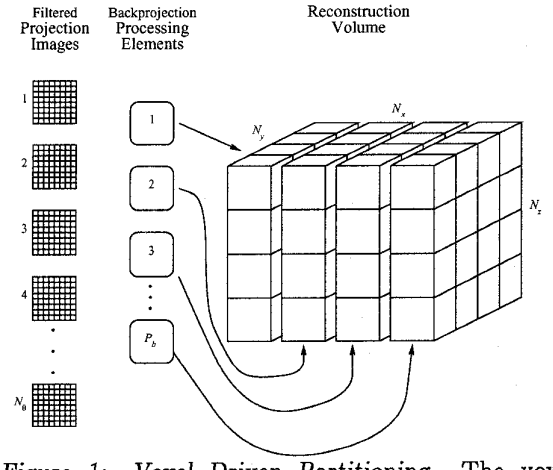


*Figure 1: Voxel Driven Partitioning.* The voxel driven data partitioning associated with the cone beam reconstruction problem is shown. Each processing element reconstructs a disjoint subset of the total reconstruction volume.

## IMPLEMENTATIONS

Cluster of Workstations

A parallel implementation on a workstation cluster has several positive implementation aspects. Workstation clusters are common in industry, research, and universities and have been found useful in solving large problems. They are more cost effective than supercomputers both in terms of initial cost and maintenance. The availability of networked systems is generally high throughout the day and the availability and throughput on such a system can be accurately predicted. Since reconstruction is very deterministic, it is therefore a very good candidate for

job scheduling. Some hardware aspects of workstation clusters are better suited for the large tomography problem than dedicated parallel processors, such as higher cache/RAM and performance/cost ratios.

Symmetric Multiprocessors

Symmetric multiprocessors (SMP) have become very common in the past few years. One factor has been a higher performance/cost ratio, since all main memory and peripherals can be shared. Furthermore the communications time can be small compared with ethernet.

Message Passing Interface Library

To help solve the problem of portability of parallel programs, a group of computer vendors, software engineers, and parallel applications scientists developed the Message Passing Interface (MPI) Library. The MPI library, or simply MPI, was designed to take the useful feature of other parallel libraries and provide a common programming interface. The portability of this library allows excellent support for comparisons among various hardware architectures and between hardware architectures and theoretic models. The goal of MPI was to define precise semantics and bindings to allow third party implementation of the standard. The extensibility of the standard allows future improvements.

The current MPI specification has many useful features. It contains bindings for both C and FORTRAN. Support for standard and user defined data types is provided. It has support for synchronous and asynchronous communication. Point-to-point and collective communication on groups of processors can be easily done. Lastly, several portable implementations exist and are freely available (MPICH, LAM, etc.). Several vendors have provided specialized ports of MPI including Convex, Cray, SGI, NEC, and IBM.

3D CT using Implementation MPI

The Feldkamp algorithm was implemented using the freely available MPICH implementation of MPI. In addition to point-to-point send and receives, the reconstruction used asynchronous receives, collective broadcasts, derived data types, and synchronization barriers. The message passing algorithm for the voxel drive approach is pseudocoded as:

```
1   Initialize each PE
2     Read and Broadcast problem specifications
3     Partition memory
4     Allocate memory
5     Precomputation of weights
6   for each θ (N_θ views):
7     if (PE is ROOT)
8       Read Projection
9       Weight and Filter Projection
10    Broadcast Projection P*_θ(u, v)
11    Backproject Projection
12  Gather Reconstructed Voxels on ROOT PE
```

The important features of MPI used in the implementation are now discussed. `MPI_Bcast` was used to broadcast initial information from the root node, including volume allocation information. `MPI_Bcast` was used to send each filtered projection (32 bit pixels) to the backprojector processes. `MPI_Send` was used to send each completely reconstructed volume to the root node. `MPI_Irecv` was used to asynchronously send each completely reconstructed volume to the root node. Derived data types were used extensively. The gathering of the reconstructed volume of voxels and filtering of the projections is all done on the initial processor. This code was ported to a SMP and a workstation cluster.

## RESULTS

A volume of $256^3$ was reconstructed from 100 views of $256^2$ pixels each. A slab of memory is allocated to each processor as shown in Figure 1. The root node provides all the filtering of the projections and writing of the final result.

Cluster of Workstations

Computations were performed on a heterogeneous network of 6 Sun workstations. The workstations used were in the Sun 4 and Sparc 2 vintage. The problem was run on 1 workstation (Sun 4) with 80 MB of RAM as the serial benchmark. The problem was run again using all 6 workstations.

Because of the heterogeneous nature of the workstations, the measure of processor utilization was used. Processor utilization is defined as

$$\frac{\text{Total Computation Time}}{N_p \times \text{Wall Time}},$$

where $N_p$ is the number of processors used. The total computation time for the reconstruction problem can be easily computed as the total time for filtering and backprojecting.

A timing digram of the parallel implementation is shown in Figure 2. Note that the relative time spent in backprojection agrees well with the theory in equation 4 of 96%. An inefficiency due to load imbalance in the backprojection step was observed and reduced in two ways. First, by splitting the volume into slabs containing voxels proportional to predetermined computational speed, utilization was increased

from 58.2% to 71.7%. Second, by reducing the backprojection in the root process by the relative time of filtering and backprojection, an utilization of 81.8% was achieved.

Symmetric Multiprocessor

Computations were done on a DEC AlphaServer 2000 4/233. This machine is an SMP with 2 processors, and 256 MB of RAM. The parallel implementation with complete load balancing was used. The serial version ran in 684.2 seconds and the load balanced parallel version ran in 353.3 seconds. The speedup is therefore 1.94 and the efficiency is 96.8%. Processor utilization was 95.1%.

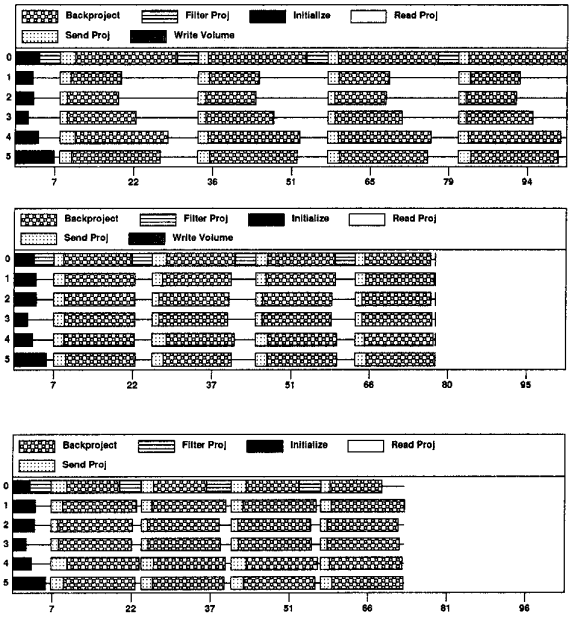


*Figure 2: Parallel Computation on Workstations.* Computation timelines are given for filtering and backprojecting four $256^2$ views into a $256^3$ volume. The top diagram shows the initial implementation without and load balancing. The middle diagram shows load balancing of the backprojection step. The bottom diagram shows load balancing of the backprojection and filtering steps.

## DISCUSSION

The reconstruction problem can be easily divided among processors. However, the large amount of communication remains an outstanding issue. Use of a broadcast bus for transmission of the projectional data could significantly reduce the communications time when many workstations are used. The use of broadcast channel has advantages in broadcasting a filtered projection to many backprojectors simultaneously, since communication time dominates when many workstations are used. Unfortunately, the broadcast mechanism in MPICH currently performs a serial send and does not take advantage of the broadcast potential of ethernet. A different broadcast scheme might also improve performance.

Further performance optimizations might be gained using asynchronous projection communications. A future goal is to fully develop a theoretic model of the voxel and ray driven backprojection algorithms. Comparison of theory with actual timings would give insight into architecture specific problems one may encounter on various systems.

## ACKNOWLEDGEMENTS

Partial support of this work was provided by NIH Grant RO1-AR42101.

## REFERENCES

[1] D. A. Reimann, M. J. Flynn, and S. M. Hames, A Flexible Laboratory System for 3D X-Ray Microtomography of 3–50 mm Specimens, in *3D Microscopy: Image Acquisition and Processing 1995*, Proceedings of the SPIE 2412, pages 186–195, San Jose, California, 5–10 February 1995.

[2] L. A. Feldkamp, L. C. Davis, and J. W. Kress, Practical cone-beam algorithm, Journal of the Optical Society of America A **1**, 612–19 (June 1984).

[3] C. Chen, S.-Y. Lee, and Z. Cho, A Parallel Implementation of 3-D CT Image Reconstruction on Hypercube Multiprocessor, IEEE Transactions on Nuclear Science **37**(3), 1333–1346 (June 1990).

[4] W. L. Nowiński, Parallel Implementation of the Convolution Method in Image Reconstruction, in *CONPAR 90 — VAPP IV*, edited by H. Burkhardt, volume 457 of *Lecture Notes in Computer Science*, pages 355–364, Springer-Verlag, 1990.

[5] M. S. Atkins, D. Murray, and R. Harrop, Use of Transputers in a 3-D Positron Emission Tomograph, IEEE Transactions on Medical Imaging **10**(3), 276–283 (September 1991).