Modeling Cone-Beam Tomographic Reconstruction Using LogSMP: An Extended LogP Model for Clusters of SMPs

David A. Reimann[1], Vipin Chaudhary[2], and Ishwar K. Sethi[3]

[1] Department of Mathematics, Albion College, Albion, Michigan 49224
[2] Department of Electrical and Computer Engineering, Wayne State University, Detroit, Michigan 48202
[3] Department of Computer Science, Wayne State University, Detroit, Michigan 48202

Abstract. The tomographic reconstruction for cone-beam geometries is a computationally intensive task requiring large memory and computational power to investigate interesting objects. The analysis of its parallel implementation on widely available clusters of SMPs requires an extension of the original LogP model to account for the various communication channels, called LogSMP. The LogSMP model is used in analyzing this algorithm, which predicts speedup on a cluster of 4 SMPs using 10 Mbps, 100 Mbps, and ATM networks. We detail the measurement of the LogSMP parameters and assess the applicability of LogSMP modeling to the cone-beam tomography problem. This methodology can be applied to similar problems involving clusters of SMPs.

1 Introduction

Tomographic reconstruction from projections using computed tomography (CT) is the non-invasive measure of structure from external measurements. The information obtained describes both internal and external shapes and material densities. This is particularly useful when one cannot make internal measurements on the object of study for a variety of reasons. These reasons might be cost, no known non-invasive technique, or no physical means to make internal measurements.

With cone-beam CT, a set of two-dimensional (2D) planar projections, consisting of $N_u \times N_v$ pixels, are acquired at equal angles around the object. These projections are filtered and backprojected into a volume of $N_x \times N_y \times N_z$ voxels. Let N_θ be the number of projections acquired. Cone-beam tomography systems are useful in assessing microstructure of biomedical and industrial objects. Tomography applications continue to grow into areas such as reverse engineering, quality control, rapid prototyping, paleontology, geology, and nondestructive testing. Cone-beam tomography systems offer greater scanner efficiency and image quality, but require much more computing [1]. To improve reconstruction time, a parallel algorithm was developed using the MPI library for communications [2,3]. The parallel algorithm is based on the serial algorithm by Feldkamp [4]. We have optimized this algorithm for a cluster of SMPs.

This processing requires roughly $O(N^4)$ time to reconstruct an image volume of N^3 voxels. For large image volumes, the reconstruction time on a serial computer far exceeds the acquisition time. This is of particular importance as the desire to resolve more detail in larger fields-of-view has demanded increased image sizes. Objects are routinely reconstructed into image volumes of 512^3 voxels and a strong desire to use 768^3, 1024^3, and larger volumes in the future.

The original LogP model [5] was extended to clusters of SMPs, called LogSMP. The LogSMP model is used in analyzing the parallel cone-beam reconstruction algorithm to predict speedup on a cluster of 4 SMPs using 10 Mbps, 100 Mbps, and 155 Mbps ATM networks. We detail the measurement of the LogSMP parameters and assess the applicability of LogSMP modeling to the cone-beam tomography problem.

2 LogSMP Model

The LogP model characterizes a homogeneous parallel architecture with bulk parameters and assumes no specific topology [5]. This model contains 4 parameters: the communications latency L, the processor overhead required for sending or receiving messages o, the bandwidth/minimum gap between successive messages g, and the number of processors P.

When using a cluster of SMPs, one must account for the differences in intra- and inter- SMP communications. The LogSMP model accounts for this additional communication channel. For this discussion, assume there are q SMPs and let SMP$_1$ contain the root node. Let P_i be the number of processors on the ith SMP, such that $\sum_{i=1}^{q} P_i = P$. Since P_i processors are on the ith node, their intra-communication is governed by the LogP parameters for that channel, namely g_i, L_i, and o_i. The SMPs can communicate with each other as dictated by the LogP parameters for the inter-communications channel, namely g_0, L_0, and o_0. In the degenerate case where there is only one processor on the ith SMP, the parameters g_i, L_i, and o_i are of no consequence. The g values are based on the fastest processor in a heterogeneous environment. This is shown in Fig. 1.

3 Parameter Measurements

Modeling performance involves assessing several timings when using the LogSMP models. The network speed relative to processor speed is a vital part of the modeling. The parameter g, in instructions/byte, measures the number of instructions which can be processed in the time to send one byte of information between two processors. Processor speed and communications speed can be used to derive g. In addition, network latency and overhead must be measured.

For the experiments, a cluster of four Sun Enterprise SMPs, each running SunOS 5.6, was used. The first had six 250 MHz nodes and 1.5 GB total RAM. This machine was used for processors 1–6 in the experiments. The other three SMPs each had four 250 MHz processors with 0.5 GB total RAM on each SMP.

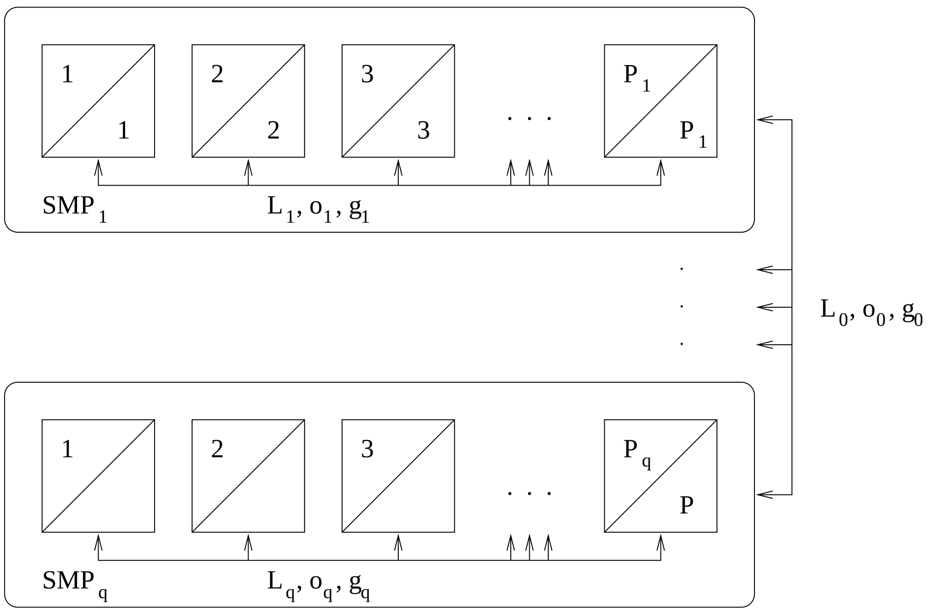

Fig. 1. In a cluster of q SMPs, the LogSMP parameters consists of the LogP parameters are required for intra- and inter- SMP communication. Each SMP can be considered a subgroup with a particular local root processor in addition to the overall root processor

Either a 10 Mbps Ethernet, 100 Mbps Ethernet, or a 155 Mbps ATM link was used for the communication. These architectures will be designated wsu10, wsu100, and wsuATM in this discussion. Version 1.1 of the MPICH implementation of MPI was used.

The processors performance was analyzed by measuring execution time of problems of size N and one processor and timing the backprojection steps. The times were then compared to the single processor model floating point operations. The processor speed was determined to be 45.1 million floating point operations per second using an independent benchmarking program. Results from this are consistent with other benchmarking programs.

Communications speed and overhead was measured using round trip point-to-point passing of messages having various power of 2 sizes from 0 to 4 MB in length. The MPI routines `MPI_Send` and `MPI_Recv` were used. For each message size, 1000 trials were measured to reduce the variance in the estimate.

The parameter g was determined by dividing the processor performance, measured in instructions per second, by the communications speed, measured in bytes per second. While the value g will in general be a function of the message size, the model presented uses the peak value. The resulting values are 0.587, 3.31, 4.59 and 39.2 cycles for the SMP, wsuATM, wsu100, and wsu10 respectively.

The latencies were 91.2, 89.1, 90.3, and 91.5 cycles for the SMP, ATM, 100 Mbps Ethernet, and 10 Mbps Ethernet respectively. In each case the processor overhead was assumed to be 0.1% of the communication time.

The overhead for starting a parallel MPI program was also determined. For each network, the influence on the number of nodes used in the computation was determined by running a parallel program which terminated immediately upon startup. A program consisting of `MPI_Init` followed immediately by `MPI_Finalize` was used. The time to execute this program was repeated ten times for 1 through 18 processors. The times from using 1 through 10 processors was fit to a line and used to predict startup times in the model. The time increases by about 0.5 seconds per processor for the wsu10, wsu100, and wsuATM networks. The variation and magnitude in startup time varies significantly once more than ten processors (more than 2 SMPs) are used.

Another significant factor in the total time required is due to reading raw projectional data and writing the final reconstructed image. The amount of data read is $2N_\theta N_u N_v$ bytes, or for the simplified problem of size N, the total amount is πN^3 bytes. For writing the final data, $2N^3$ bytes are written. The times for input and output were empirically measured and incorporated into the model.

4 LogSMP Model for the Parallel Cone-Beam Algorithm

A voxel driven approach is taken where the volume is distributed over the processors and each projection is sent to every processor. Each processor sees 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, which is just the product of the volume dimensions and the bytes required for each voxel (4), namely $4N_xN_yN_z$. The voxel data is not replicated on each processor, so individual memory requirements are not as demanding. Another advantage of this method is that the data is acquired in a serial fashion and processing could be done in concert with acquisition.

The parallel algorithm utilizes a master processor which does all I/O and essentially dictates the tasks performed by the other processors. While parallel I/O is sometimes available, the algorithm requires the final reconstructed data to be explicitly sent back to the master processor. The MPI library is initialized using `MPI_Init`. MPI assigns each processor in the communication group a unique id. The number of other processes in the group is also available. All processes are initially only part of the `MPI_COMM_WOLD` process group. An MPI communicator is then created for each SMP containing all processes on that SMP. Another communicator is created containing the root nodes of each SMP communicator group.

For the case of $N = N_x = N_y = N_z = N_u = N_v$, the number of projections N_θ should be $\frac{\pi}{2}N$, and thus the complexity of the problem is $O(N^4)$. N can also serve as an upper bound when N is the maximum of those parameters. Using this simple model it becomes easier to study the effects of N, P, and g on the theoretical speedup.

In addition to computation requirements, a major nontrivial aspect of the problem is the data size required. For example, an acquisition of 804 views of 512^2 images with 16 bit integer pixels is needed to create a 512^3 volumetric data set with 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 volume is 512 MB, which is currently feasible on SMPs, but even more so on a collection of SMPs.

The time required to perform a reconstruction can be expressed as

$$T = T_0 + T_\mathrm{i} + T_\mathrm{r} + T_\mathrm{f} + T_\mathrm{c} + T_\mathrm{b} + T_\mathrm{g} + T_\mathrm{w} \tag{1}$$

where T_0 is the MPI startup time, T_i is the initialization time, T_r is the time to read a projection from disk, T_f is the time to filter a projection, T_c is the time to communicate the filtered projection from the root processor to the others for backprojection, T_b is the time to backproject a filtered projection, T_g is the time required to gather the reconstructed volume to the root processor for subsequent scaling and writing. and T_w is the time to write the reconstructed volume to disk. In general, these times are functions of processor speed, network bandwidth and latency, and the number of processors used. T_0, T_r, and T_w are measured empirically as a function of P as described below.

The algorithm consists of repeated asynchronous sends of the filtered projections from the root node a single node in each SMP, and synchronous broadcasts occur from that node to all others in that SMP. The root node also accounts for the filtering time and reduces the number of voxels to backproject by the corresponding amount. In this algorithm times can be modeled as follows. $T_\mathrm{i} = 7N_uN_v$ is the initialization time and consists of memory allocation, and precomputation of values required for weighting the projectional data. $T_\mathrm{f} = N_\theta(N_uN_v(3 + 18\log_2(2N_u)))$ is the time required to filter a projection on the root processor. $T_\mathrm{b} = N_\theta(N_yN_x(13 + 12N_z))/P$ is the time required to backproject on P processors. $T_\mathrm{g} = (P_1 - 1)4N_y(N_xN_zg_1 + L_1)(1 + o_1)/P + (P - P_1)4N_y(N_xN_zg_0 + L_0)(1 + o_0)/P$ where the first expression is gathering on root SMP, and second expression is gathering from remote (non-root) SMPs.

$$T_\mathrm{c} = N_\theta(4(P_0-1)(N_uN_vg_0+L_0)(1+o_0)+\max_{i=1}^{P_0} 4(P_i-1)(N_uN_vg_i+L_i)(1+o_i)) \tag{2}$$

where the first expression corresponds to inter-SMP communication and the second corresponds to intra-SMP communication. Where it is assumed that P_i, L_i, o_i, and g_i are the LogP parameters for ith SMP and P_0, L_0, o_0, and g_0 are the LogP parameters corresponding to communications between SMPs.

5 Results and Discussion

Data sets of size $N = 32, 64, 128, 256,$ and 512 were simulated and reconstructed using both systems described above. The number of processors varied from one to the maximum available, 18. Each run was repeated three times to average any deviations and to help identify any outliers. The time to complete the execution was measured using the UNIX time command. While time is precise to the nearest tenth of a second, that is sufficient precision when total times ranges from several minutes to several hours. Speedups were modeled for each value of N based on the measured LogSMP parameters.

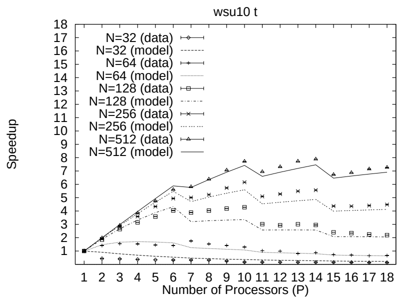

Fig. 2. 10 Mbps ethernet

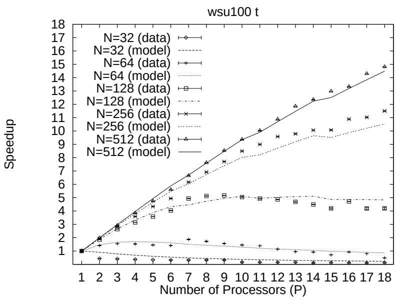

Fig. 3. 100 Mbps ethernet

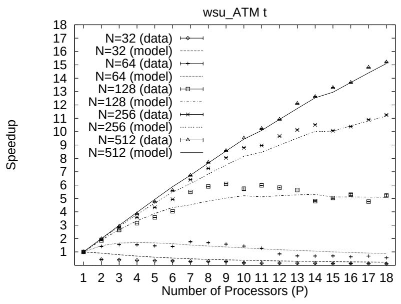

Fig. 4. 155 Mbps ATM

The cluster of SMPs was connected via 10 Mbps ethernet, 100 Mbps ethernet, and then ATM. For each case the reconstruction algorithm was timed and speedup as a function of the number of processors was computed. The speedups corresponding to the model are plotted along with the empirical data in Fig. 6–8. Recall that the first SMP has six processors and the other three each have four processors. Using 10 Mbps ethernet, a maximal speedup of 7.89 was achieved for a problem size of 512 and using 14 processors. Using 100 Mbps ethernet, a speedup of 14.8 was achieved for a problem size of 512 and using 18 processors. Using ATM, a speedup of 15.2 was achieved for a problem size of 512 and using 18 processors.

The speedup is nearly linear when $P < 6$, since $g \approx 1$. As P increases to force inter-SMP communication, a piecewise continuous behavior is observed and well modeled. By properly accounting for the parameters for each communications channel, an accurate prediction of speedup was obtained. We found the LogSMP model can take into account differences in communication costs and can be easily used to model performance when using clusters of SMPs. This methodology can be applied to similar problems involving large multi-dimensional data sets.

6 Acknowledgements

This research was supported in part by NIH grant AR-43183, NSF grants MIP-9309489, EIA-9729828, US Army Contract DAEA 32-93D004, and Ford Motor Company grants 96-136R and 96-628R.

References

1 David A. Reimann, Sean M. Hames, Michael J. Flynn, and David P. Fyhrie. A cone beam computed tomography system for true 3D imaging of specimens. *Applied Radiation and Isotopes*, 48(10–12):1433–1436, October–December 1997.

2 David A. Reimann, Vipin Chaudhary, Michael J. Flynn, and Ishwar K. Sethi. Parallel implementation of cone beam tomography. In A. Bojanczyk, editor, *Proceedings of the 1996 International Conference on Parallel Processing*, volume II, pages 170–173, Bloomingdale, Illinois, 12–16 August 1996.

3 David A. Reimann, Vipin Chaudhary, Michael J. Flynn, and Ishwar K. Sethi. Cone beam tomography using MPI on heterogeneous workstation clusters. In *Proceedings, Second MPI Developer's Conference*, pages 142–148, University of Notre Dame, Notre Dame, Indiana, 1–2 July 1996. IEEE Computer Society Press.

4 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.

5 David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a realistic model of parallel computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, San Diego, California, May 1993.