Parallel Sparse Triangular Solution with Partitioned Inverses and Prescheduled DAGs

Frederic T. Chong*
ftchong@lcs.mit.edu
MIT Laboratory for Computer Science

Sparse triangular solution offers a challenging irregular problem for parallel systems. The repeated solution of the system $Lx = b$, where L is a lower triangular factor of a sparse matrix, arises in numerous applications. A previous study [CSBS95] has shown that, if L is an incomplete factor, $Lx = b$ can be efficiently solved on a parallel system through substitution. This was accomplished by representing the computation as a directed acyclic graph (DAG) that was prescheduled by dominant sequence clustering (DSC) [GY92].

Unfortunately, when L is a complete factor, parallelism is extremely limited. This study uses the method of partitioned inverses [AS93] [APS93] to increase available parallelism over substitution, commonly by an order of magnitude. We also represent our partitioned inverses computation as a DAG and preschedule using DSC. Our preliminary results exhibit dramatically improved speedups on the CM5 and CM5E.

Partitioned inverses is an alternative method of sparse triangular solution that has been shown to greatly enhance parallelism [AS93] [APS93]. In general, the system $Lx = b$ can be solved through the computation $x = L^{-1}b$. Unfortunately, the calculation of L^{-1} can lead to *fill*, nonzeros that were not in the original sparse structure of L. Fill occurs because L^{-1} must have the structure of the transitive closure of L. Fill is undesirable because it increases computation and, more importantly, decreases parallelism.

To avoid fill, we can partition L, as shown in Figure 1, into subgraphs (called *column subgraphs*) that are transitively closed. We can then solve our system by calculating $x = L_1^{-1} L_0^{-1} b$.

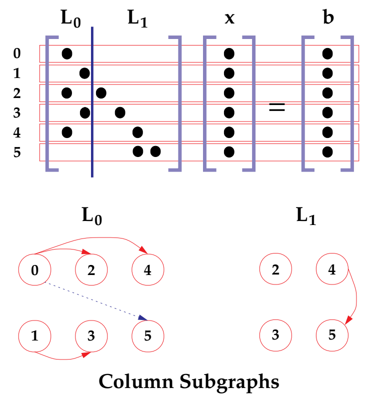

Figure 1: A partitioned L and its column subgraphs. We can avoid fill by partitioning L into L_0 and L_1 and then inverting the L_i's separately. The dashed arcs in L_0 no longer apply because the 4-to-5 dependency has been partitioned to L_1, breaking the original transitive closure.

*Fred Chong is advised by Anant Agarwal. This work was done in collaboration with Robert Schreiber, NASA Research Institute for Advanced Computer Science. **Support:** This work was supported in part by ARPA contract N00014-94-1-09885, by NSF Experimental Systems grant MIP-9012773, by an NSF Presidential Young Investigator Award to Prof. Anant Agarwal, by Project SCOUT (ARPA Contract MDA972-92-J-1032) and by NASA via Contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA). **Address:** Room 613, 545 Technology Square, Cambridge MA 02139. **Web:** http://www.ai.mit.edu/people/ftchong/

In general, $Lx = b$ may be solved by computing:

$$x = \prod_i L_i^{-1} b$$

where the L_i's are chosen to have transitively closed column subgraphs. We use the `rp2` algorithm given in [AS93] to choose our partitions. Note that `rp2` may also reorder L into other lower triangular matrices in an attempt to find better partitions.

We ran our experiments on several sparse matrices taken from the Harwell-Boeing [DGL92] benchmark set. We took lower triangular factors from five power grid matrices and two structures matrices. Our results demonstrate that, using partitioned inverses, speedups are possible where substitution would only produce slowdowns. Figure 2 shows speedups for our benchmarks on the CM5 from 4 to 128 processors. Speedups are relative to an optimized sequential, substitution-based code running on a single Cypress node of the CM5. From our previous study, we know that our DAG execution code runs at about 25 percent efficiency. Additionally, partitioned inverses does about 10 percent more work than substitution on these benchmarks. Any additional inefficiency is due to scheduling. Scheduling losses are due to either limitations in inherent parallelism or DSC's inability to model communications overhead.

A full paper on this research can be found in [CS95].

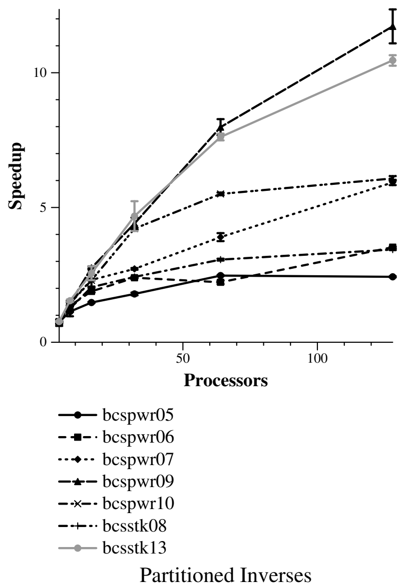

Figure 2: CM5 speedups.

References

[APS93] Fernando Alvarado, Alex Pothen, and Robert Schreiber. *Highly parallel sparse triangular solution*, pages 141–157. Springer-Verlag, New York, 1993. The IMA Volumes in Mathematics and Its Applications, Volume 56.

[AS93] Fernando Alvarado and Robert Schreiber. Optimal parallel solution of sparse triangular systems. *SIAM Journal of Scientific and Statistical Computation*, 14:446–460, 1993.

[CS95] Frederic T. Chong and Robert Schreiber. Parallel sparse triangular solution with partitioned inverses and prescheduled DAGs. In *1995 Workshop on Solving Irregular Problems on Distributed Memory Machines*, Santa Barbara, California, April 1995. To appear.

[CSBS95] Frederic T. Chong, Shamik D. Sharma, Eric A. Brewer, and Joel Saltz. Multiprocessor runtime support for irregular DAGs. In Rajiv K. Kalia and Priya Vashishta, editors, *Toward Teraflop Computing and New Grand Challenge Applications*. Nova Science Publishers, Inc., 1995. To appear.

[DGL92] Ian S. Duff, Roger G. Grimes, and John G. Lewis. User's guide for the Harwell-Boeing sparse matrix collection. Technical Report TR/PA/92/86, CERFACS, 42 Ave G. Coriolis, 31057 Toulouse Cedex, France, October 1992.

[GY92] Apostolos Gerasoulis and Tao Yang. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. *Journal of Parallel and Distributed Computing*, 16:276–291, 1992.