# Ordering Sparse Matrices for Cache-Based Systems

***Rupak Biswas***[*] ***Xiaoye Li***[†]
***Parry Husbands***[‡] ***Leonid Oliker***[§]

The Conjugate Gradient (CG) algorithm is the oldest and best-known Krylov subspace method used to solve sparse linear systems. Most of the floating-point operations within each CG iteration is spent performing sparse matrix-vector multiplication (SPMV). We examine how various ordering and partitioning strategies affect the performance of CG and SPMV when different programming paradigms are used on current commercial cache-based computers. However, a multithreaded implementation on the cacheless Cray MTA demonstrates high efficiency and scalability without any special ordering or partitioning.

Numerical calculations of realistic problems usually require solving a large set of non-linear partial differential equations (PDEs) over a finite region. Unstructured meshes are often used to discretize applications involving complex geometries or those with dynamically moving boundaries. They also facilitate dynamic grid refinement and coarsening to efficiently resolve evolving physical features such as shocks, vortices, and shear layers. The CG algorithm is perhaps the best-known iterative technique to solve such sparse linear systems that are symmetric and positive definite. Within each iteration of CG, SPMV is usually the most expensive operation.

The numerical solution of such complex problems can be extremely time consuming, a fact driving the development of increasingly powerful parallel machines. However, modern computer architectures, based on deep memory hierarchies, show acceptable performance only if users care about the proper distribution and place-

[*] Mail Stop T27A-1, NASA Ames Research Center, Moffett Field, CA 94035 (`rbiswas@nas.nasa.gov`).

[†] One Cyclotron Rd, MS:50F, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (`xsli@lbl.gov`).

[‡] One Cyclotron Rd, MS:50F, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (`PJRHusbands@lbl.gov`).

[§] One Cyclotron Rd, MS:50B-2239, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (`loliker@lbl.gov`).

ment of their data. Single-processor performance on such cache-based systems depends on the exploitation of locality, and parallel performance degrades significantly if inadequate partitioning of data causes excessive communication and/or data migration.

In this work, we examine how various ordering and partitioning strategies affect the performance of CG and SPMV on different cache-based computers. In particular, we use the reverse Cuthill-McKee (RCM) [1] and the self-avoiding walks (SAW) [2] ordering strategies, and the METIS [3] partitioner. Parallel implementations are presented using MPI and shared-memory compiler directives on three state-of-the-art parallel supercomputers: a Cray T3E, an SGI Origin2000, and an IBM SP. Results show that ordering significantly affects overall performance, that cache reuse can be more important than reducing communication, and that it is possible to achieve message passing performance using shared-memory constructs through careful data ordering and distribution. However, a multithreaded implementation on the cacheless Cray MTA does not require any special ordering or partitioning to obtain high efficiency and scalability.

The Cuthill-McKee enumeration algorithm is based on ideas from graph theory. Starting from a vertex of minimal degree, levels of increasing distance from that vertex are first constructed. The enumeration is then performed level-by-level with increasing vertex degree (within each level). RCM is the most popular variation of this method, where the level construction is restarted from a vertex of minimal degree in the final level.

SAW, on the other hand, is similar to space-filling curves, but directed towards unstructured meshes. A SAW over a triangular mesh is an enumeration of the triangles such that two consecutive triangles in the SAW share an edge or a vertex. SAWs are amenable to hierarchical coarsening and refinement, i.e. they have to be rebuilt only in regions where mesh adaptation occurs, and can therefore be easily parallelized. Unlike RCM, SAW does not specifically enumerate vertices. This implies a higher construction cost for SAWs, but has the advantage that several different vertex enumerations can be derived from a given SAW.

METIS, which uses a multilevel algorithm, is currently the most popular graph partitioning library. It reduces the size of the graph by collapsing vertices and edges using a heavy edge matching scheme, applies a greedy graph growing algorithm to partition the coarsest graph, and then uncoarsens it back to the original by using a combination of boundary greedy and Kernighan-Lin refinement. It is important to note that partitioners strive to balance the computational workload among processors while reducing interprocessor communication; improving cache performance is not a typical objective.

An explicit message-passing implementation using MPI and the Aztec library was tested on the T3E at NERSC. Results on a $661,054^2$ matrix with 25,753,034 nonzeros for key kernel routines show that SAW is always about two times faster than RCM and METIS. However, all three strategies demonstrate good scalability (more than 75% efficiency) up to the 64 processors that were used for these experiments. Investigations reveal that servicing the cache misses is extremely expensive and typically requires more than 90% of the total execution time. SAW has the fewest cache misses while METIS minimizes the average communication volume, indicating that improving cache reuse may be more important than reducing interprocessor communication for sparse matrix computations.

A shared-memory version using SGI's native pragma directives was implemented on the Origin2000 at NASA Ames. Particular attention was paid to the distributed-memory nature of the machine by performing an appropriate initial data distribution. Little performance difference between RCM and SAW is observed since both ordering strategies reduce the number of secondary cache misses and the non-local memory references of the processors. A comparison with the MPI version running on the Origin2000 revealed similar runtimes, even though the programming methodologies are significantly different. This shows that it is possible to achieve message-passing performance using shared-memory constructs, through careful data ordering and distribution.

The MTA has a radically different architecture, and uses multithreading instead of data caches to hide memory latency. Compared to standard cache-based systems, programmability is significantly enhanced as the user need not control data placement. Performance primarily depends on having a large number of concurrent computation threads. The multithreaded implementation of CG only requires a few compiler directives. Results using 60 threads per processor show that CG and SPMV achieve excellent scalability (more than 90% efficiency) without any ordering. This indicates that there is enough thread and instruction level parallelism in CG to tolerate the relatively high overhead of memory access.

Realistic scientific applications require preconditioning algorithms, such as ILU, in order for CG to reach convergence. We report on the effects of preconditioning for various ordering strategies and architectural platforms. We also examine the effects of first partitioning the sparse matrix using METIS, and then performing RCM or SAW orderings on each subdomain. Finally, results from a hybrid (MPI and OpenMP) version of the code running on the new IBM RS/6000 SP machine at SDSC are presented.

[1] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," *Proc. ACM National Conference*, 1969, 157–192.

[2] G. Heber, R. Biswas and G.R. Gao, "Self-avoiding walks over adaptive unstructured grids," *Concurrency: Practice and Experience*, 12 (2000) 85–109.

[3] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Scientific and Statistical Computing* 20 (1998) 359–392.