FASTMath Team Members: Jim Demmel (UC Berkeley), Sherry Li, Francois-Henry Rouet, Pieter Ghyselens (Lawrence Berkeley National Laboratory)

We develop scalable sparse direct linear solvers and effective preconditioners for the most challenging linear systems, which are often too difficult for iterative methods. Our focal efforts are the developments of three types of linear solvers: The first is a pure direct solver, encapsulated in SuperLU_DIST software. The second is a Schur complement-based hybrid solver, superlinsp, encapsulated in PDSLin software. The third type is the nearly-optimal preconditioners using low-rank approximate factorization of the dense submatrices. We are also developing communication-avoiding linear algebra algorithms which have the potential to be used in the above sparse linear solvers.

Multicore / GPU-aware SuperLU

Objectives
- Develop scalable sparse direct solvers to support simulations of numerically challenging problems, e.g., accelerator, fusion, quantum chemistry, and fluid mechanics.
- Efficient utilization of new hardware resources, especially heterogeneous nodes.

Recent Accomplishments
- New hybrid programming code: MPI+OpenMP+CUDA, able to use all the CPUs and GPUs on manycore computers.
- New CPU multithreading and GPU
- Algorithmic changes:
  - Aggregate small BLAS operations into larger ones.
  - Multithreading Scatter/Gather operations.
  - Hide long-latency operations.
- Results: using 100 nodes GPU clusters, up to 2.7x faster, 2x-5x memory saving.
- New SuperLU_DIST 4.0 release, August 2014.

Parallel HSS Low Rank Factorization

Objectives
- Develop a new class of sparse approximate factorization algorithms that exploit the data-sparserness (low-rankness) in matrix off-diagonal blocks.
- As direct solvers for PDEs with smooth kernels, BEMs, integral equations, or preconditioners for general problems.
- Use Hierarchically Semiseparable (HSS) representation (nested bases), achieves asymptotically lower complexity in both FLOPS and communication.

Application Impact
- Use of PDSLin reduces time to solution and memory by factors of 20 and 5, respectively, in the ACE3P code for modeling accelerator cavities (ComPASS).
- SuperLU is used in the PEXSI method for electronic structure calculations (BES Scidac projects: AMIS and DGDFT), and the fusion simulation codes M3D-C1 and NIMROD (FES Scidac: CEMM).
- SuperLU has over 27,000 downloads in FY2013. It is used in commercial mathematical libraries, such as Cray's LibSci, FEMLAB, HP’s MathLib, IMSL, NAG, OptimaNumerics, and Python (SciPy).

Randomized Butterfly Transformations

Objectives
- Use RBT as preprocessing to avoid expensive pivoting in sparse LU or LDLT.
- RBT is easily scalable, as opposed to numerical pivoting.

Parallelization Impact
- RBT: B = U^T W U, where U and V are recursive butterfly matrices. B is guaranteed to be factorizable without pivoting.
- The increase of B's factors size is modest for many matrices.
- Tested 90 sparse matrices, compared to SuperLU (GE with partial pivoting): 37 have smaller factor size, 30 have increase <= 2x, 23 have increase > 2x. 69 have <= 2 digits loss of solution accuracy.
- In the process of parallelization study in SuperLU_DIST.

References

Communication-Avoiding Direct Methods

Objectives
- New, Stronger Communication Lower Bounds
  - Old: Latency_Lower_Bound = Bandwidth_Lower_Bound / Largest_Message_Size
  - Attainable for Matmul (yielding perfect strong scaling) but not LU
  - New: for LU (and other algorithms with similar dependencies): Latency*Bandwidth = O(n^2)
  - Can't minimize both simultaneously
  - 2.5D LU is optimal for all choices of latency and bandwidth
  - Similar results for dense TRSV, A^T x kernel for stencil-like matrices
- Generalizing Lower Bounds and Optimal Algorithm to More Programs
  - Old: Applied to direct linear algebra, i.e. programs expressible as 3-nested loop accessing 3 arrays, eg C(i,j,k), A(i,j), B(k,i)
  - New: Applies to *any* programs expressible as any number of nested loops, accessing any number of arrays, with any subscripts that are affine combinations of loop indices, eg C(i,j), 2i+j+3, j^2+j, m...)
  - Examples beyond linear algebra: Direct n-body algorithms (where 2 or more bodies interact), tensor contractions, Database join, ...

More Information: http://www.fastmath-scidac.org or contact Lori Diachin, LLNL, diachin2@llnl.gov 925-422-7130