FASTMath Direct Solver Technologies

FASTMath Team Members: Jim Demmel (UC Berkeley), Sherry Li, Francois-Henry Rouet, Pieter Ghyseles (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, , encapsulated in PSDLin 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.

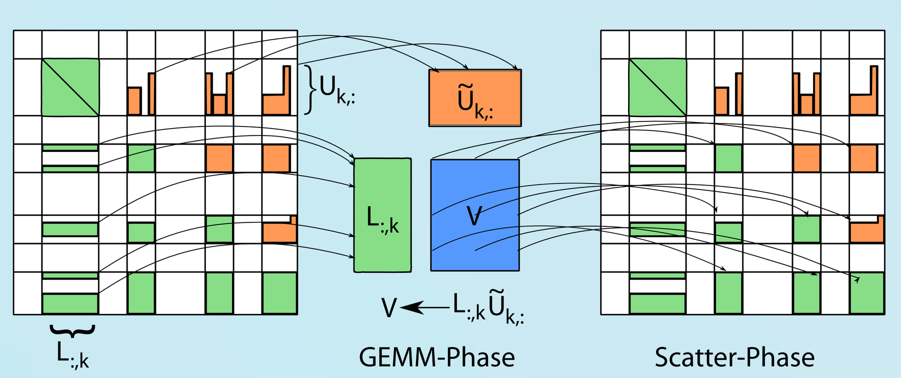
Multicore / GPU-aware SuperLU

Objectives

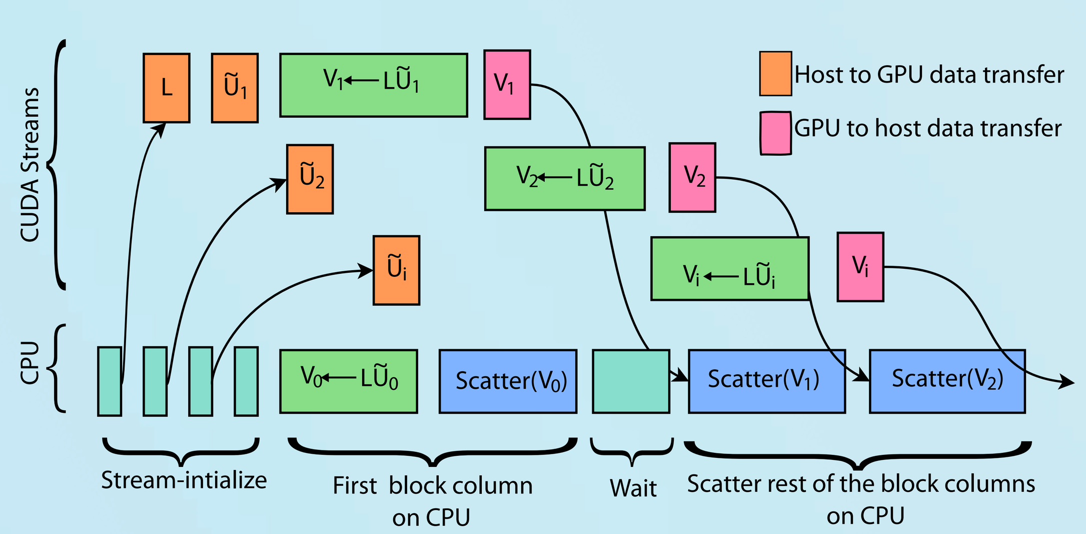
- Develop scalable sparse direct linear solvers to support simulations of numerically challenging problems, e.g., accelerator, fusion, quantum chemistry, and fluid mechanics.
- Efficient utilization of new hardware resources, especially heterogeneous nodes.

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.



① *Aggregate small blocks*
② *GEMM of large blocks*
③ *Scatter*



GPU acceleration:

Software pipelining to overlap GPU execution with CPU Scatter, data transfer.

Future Plans

- Intel Xeon Phi in progress.
- More OpenMP for the "other" part, e.g., triangular solve.
- Study on larger GPU cluster: Titan, Blue Waters.

References

- P. Sao, R. Vuduc, and X.S. Li, "A distributed CPU-GPU sparse direct solver", Proc. of Euro-Par 2014 Parallel Processing, August 25-29, Porto, Portugal.

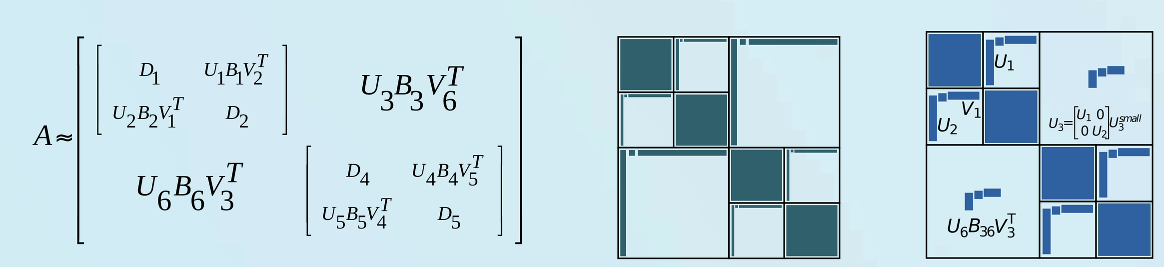
Application Impact

- 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).

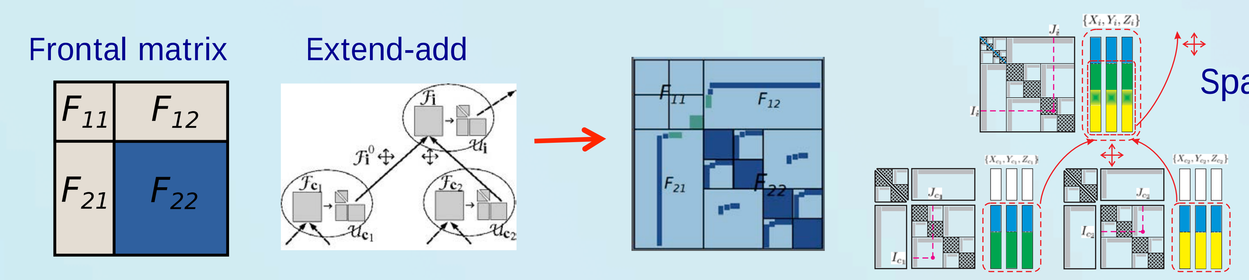
Parallel HSS Low Rank Factorization

Objectives

- Develop a new class of sparse approximate factorization algorithms that exploit the data-sparseness (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.



Dense HSS



Sparse Multifrontal HSS

Recent Accomplishments

- Randomized sampling in place of traditional RRQR for low-rank compression, which simplies extend-end in sparse MF, further reduces operation complexity.
- Shared-memory parallelizaion: use OpenMP task pragma to schedule tree-based irregular parallelism.
- Distributed-memory parallelization: use MPI, BLACS, and PBLAS; arrange processes in a tree structure with nested subgroups; use proportional mapping of e-tree and HSS tree nodes to balance workload.
- Results: using 1024 cores, RS-based HSS construction is 5x faster than RRQR.
- New parallel StruMPACK software to be released in Fall 2014.

Future Plans

- Partitioning/ordering to expose better low rankness ("admissible condition").
- Exploit fine-grained parallelism.
- Analyze communication lower bound.

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.

Butterfly matrix of size $n \times n$: $B^{<n>} = \frac{1}{\sqrt{2}}\begin{bmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{bmatrix}$, R_0 and R_1 are random diagonal $\frac{n}{2} \times \frac{n}{2}$ matrices,

Recursive Butterfly matrix is a product of butterfly matrices, $n = 2^d$:

$$W^{<n,d>} = \begin{bmatrix} B_1^{<n/2^{d-1}>} & & \\ & \ddots & \\ & & B_{2^{d-1}}^{<n/2^{d-1}>} \end{bmatrix} \cdot W^{<n,d-1>}, \text{ with } W^{<n,1>} = B^{<n>}$$

Recent Accomplishments

- RBT: $B = U^T AV$, 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

- M. Baboulin, X.S. Li and F.-H. Rouet, "Using Random Butterfly Transformations to avoid pivoting in sparse direct methods", VECPAR 2014 Conference, June 30 – July 3, 2014.

Communication-Avoiding Direct Methods

- 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 = Ω (n^2)
    - Can't minimize both simultaneously
    - 2.5D LU is optimal for all choices of latency and bandwidth
    - Similar results for dense TRSV, $A^k x$ kernel for stencils-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), A(i,k), B(k,j)
  - 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,2*i+3*j,k-2*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