Sparse Direct Solvers and Preconditioners on Manycore Systems

Sherry Li, Pieter Ghysels, Francois-Henry Rouet (Lawrence Berkeley National Laboratory), Piyush Sao, Richard Vuduc (Georgia Tech)

We develop scalable sparse direct linear solvers and effective preconditioners for the most challenging linear systems on manycore parallel machines. Our focal efforts are the developments of two types of solvers: The first is a pure direct solver, encapsulated in SuperLU software. The second is the nearly-optimal preconditioners using the HSS low-rank approximate factorization of the dense submatrices, encapsulated in STRUMPACK software.

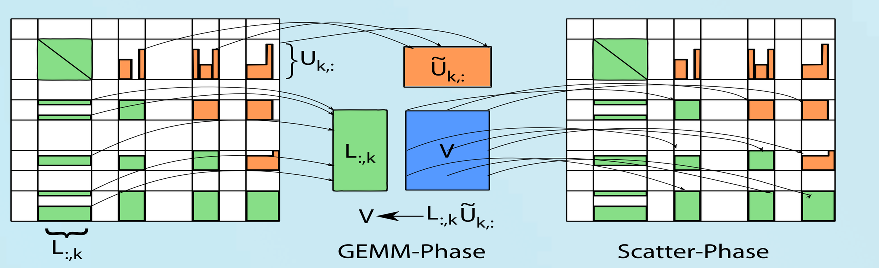
Direct Solver SuperLU: Multicore / GPU-aware

Challenges

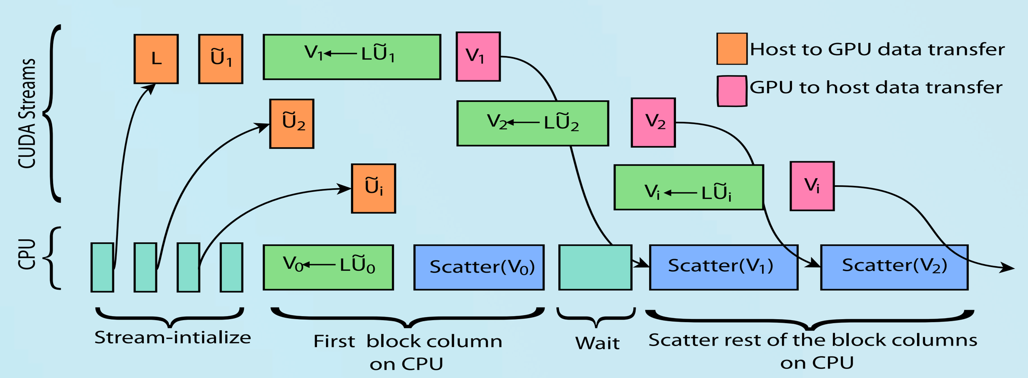
- Strong task/data dependencies given by DAG
- Irregular data access, scatter/gather
- Low Arithmetic Intensity in the beginning, higher AI later

Strategies on CPU + GPU

- CPU multithreading Scatter/Gather, GPU does data-parallel BLAS only.
- Overlap CPU & GPU activities to hide PCI transfer.
- Results: 100 nodes GPU clusters, 2.7x faster, 2-5x memory saving
- Programming: MPI + OpenMP + CUDA



① Aggregate small blocks
② GEMM of large blocks
③ Scatter

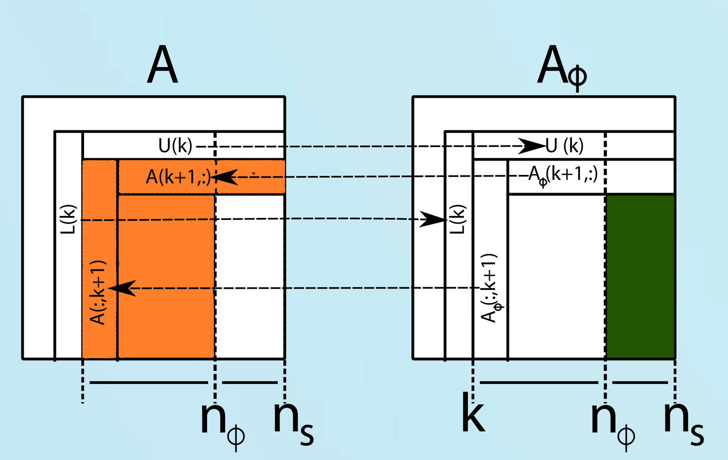


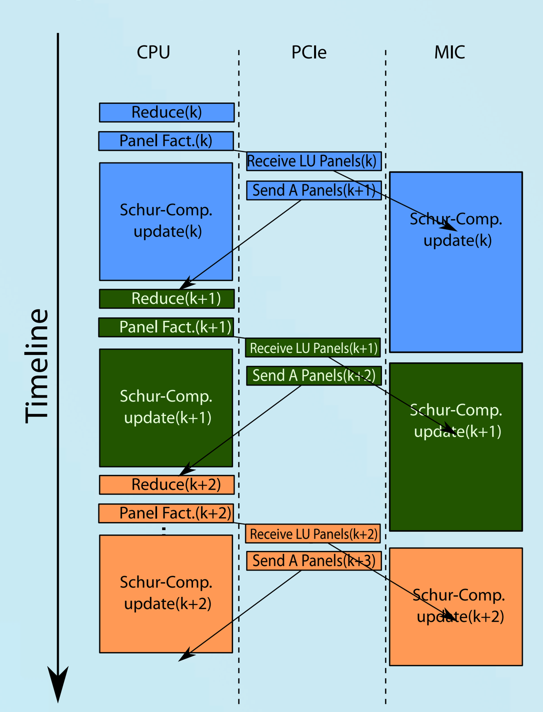
GPU acceleration:

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

Strategies on Intel Xeon Phi (MIC): offload mode

- Offload both GEMM and Gather/Scatter on MIC, take advantage of more powerful cores than GPU, higher memory BW on MIC.
- HALO algorithm – Highly Asynchronous Lazy Offload
  - Two partial sums of Schur-complement are maintained on CPU, MIC.
  - Reduce the to-be-factorized panel on CPU, absorbing MIC's panel.



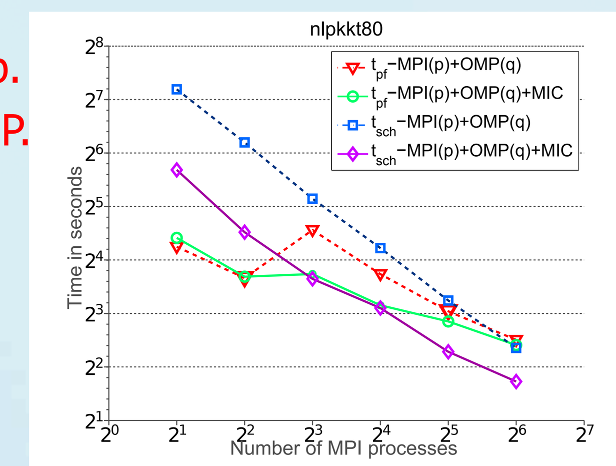


References

- 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.
- P. Sao, X. Liu, R. Vuduc, and X.S. Li, "A Sparse Direct Solver for Distributed Memory Xeon Phi-accelerated Systems", IPDPS 2015, May 25-29, 2015, Hyderabad, India.

SuperLU_DIST Performance on Intel Phi

- 1-node Sandy Bridge-EP: 2 sockets / 16 cores / 32 threads, 2 MICs
  - CPU only: OMP(p), MPI(p)+OMP(q)
  - Added MIC: OMP(p)+MIC (1), MPI(p)+OMP(q)+MIC(2)
- Results:
  - 2nd MIC gives additional 1.8x speedup.
  - 2.5x faster than CPU-only with OpenMP.
- Bottleneck: panel factorization.



RBT to Avoid Numerical Pivoting

Algorithms

- Use Randomized Butterfly Transformation as preprocessing to avoid expensive pivoting in sparse LU or LDLT.
- RBT is easily scalable, as opposed to numerical pivoting.
- RBT: A1 = $U^T AV$, where U and V are recursive butterfly matrices.
  A1 is guaranteed to be factorizable without 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 random diagonal $\frac{n}{2} \times \frac{n}{2}$ matrices,

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

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

Results:

- The increase of A1's factor 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.
- Parallel transformation (d = 1),
  - nlpkkt120: a matrix of dimension 3.5 M, 95 M nonzeros
  - 1 second @ 4 cores, 0.4 seconds @ 32 cores
- In the process of scalability 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.

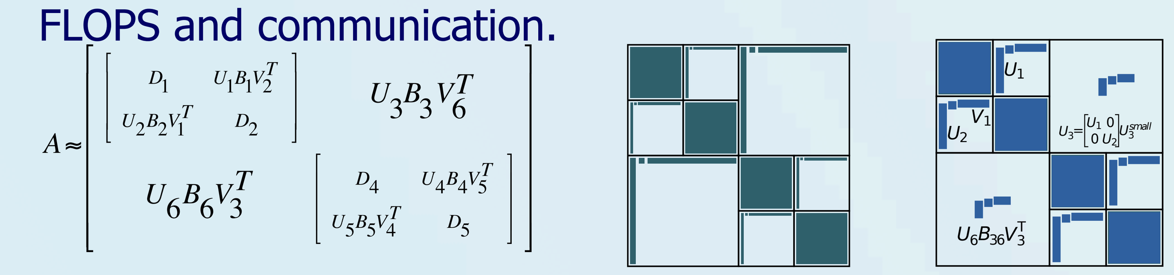
Parallel HSS Low Rank Factorization

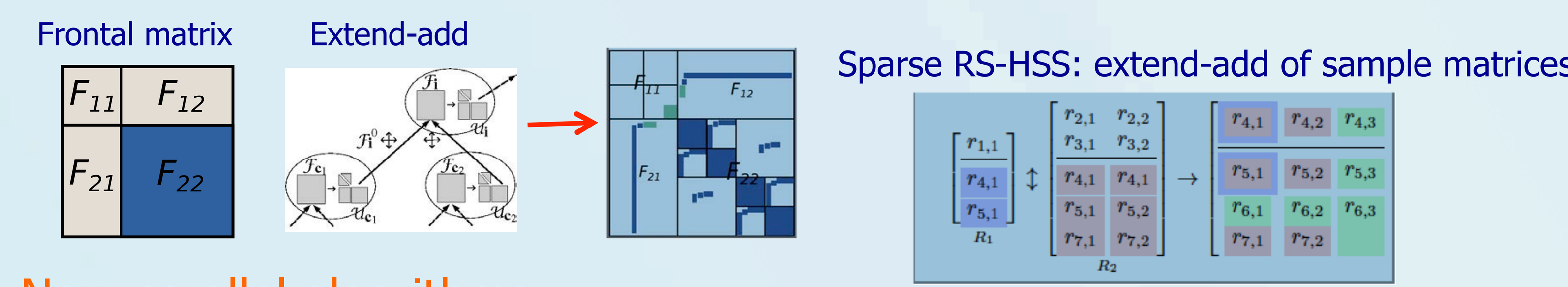
STRUMPACK Software (STRUctured Matrices PACKage)

- http://portal.nersc.gov/project/sparse/strumpack/
- Dense: distributed using MP; Sparse: shared-memory using OpenMP

Hierarchically semiseparable low-rank matrices

- As direct solvers for PDEs with smooth kernels, BEMs, integral equations, or preconditioners for general problems.
- Nested bases allows to achieve asymptotically lower complexity in both FLOPS and communication.

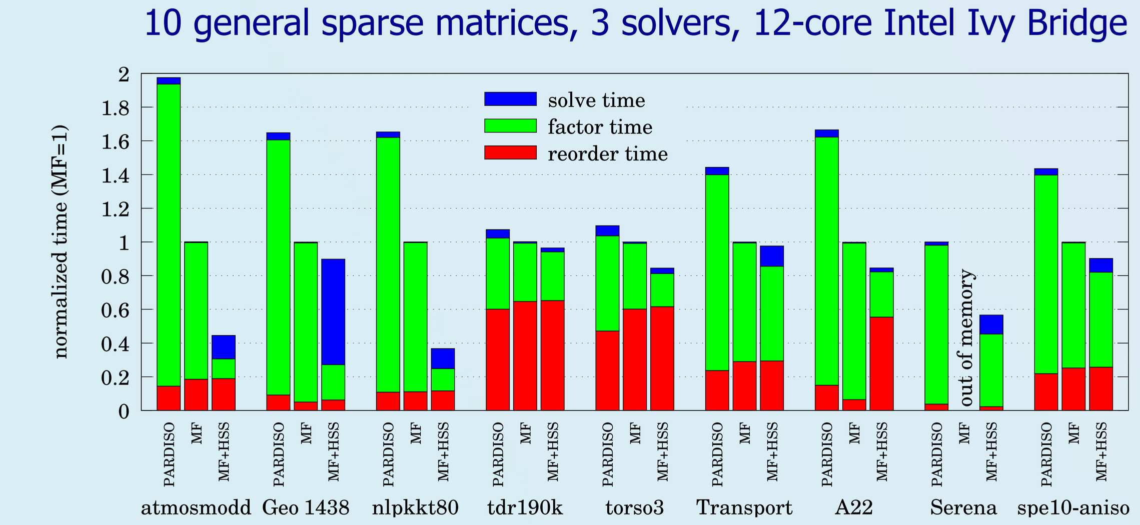




New parallel algorithms

- Randomized sampling in place of traditional RRQR for compression, simplifies extend-end in sparse MF, further reduces flops.
- Shared-memory: use OpenMP task pragma to schedule tree-based irregular parallelism.
- Distributed-memory: 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:
  - 1024 cores, RS-based HSS construction 6x faster than RRQR based.
  - Sparse RS-HSS-MF up to 7x faster than MF for model PDEs.

10 general sparse matrices, 3 solvers, 12-core Intel Ivy Bridge



References

- F.-H. Rouet, Xiaoye S. Li, P. Ghysels. A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. Submitted to ACM Transactions on Mathematical Software, 2014.
- P. Ghysels, X.S. Li, F.-H. Rouet, S. Williams, A. Napov An efficient multi-core implementation of a novel HSS-structured multifrontal solver using randomized sampling. Submitted to SIAM SISC Special Issue CSE 2015, 2015.

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

SciDAC Scientific Discovery through Advanced Computing