Accelerating Direct Linear Solvers with Algorithmic and Hardware Advances

Xiaoye Sherry Li

xsli@lbl.gov

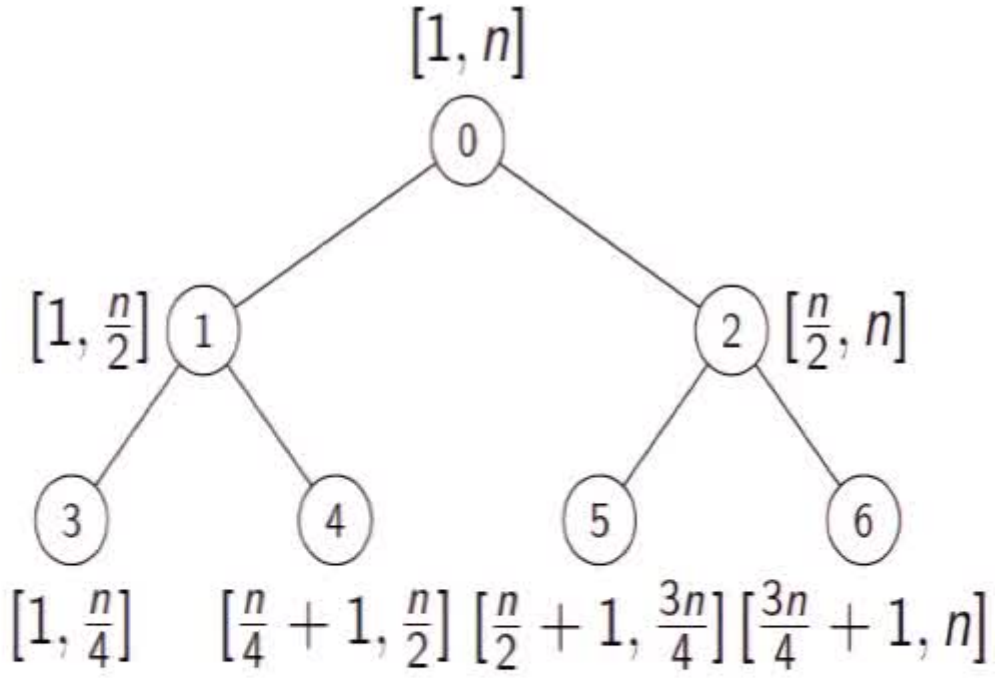
Lawrence Berkeley National Laboratory

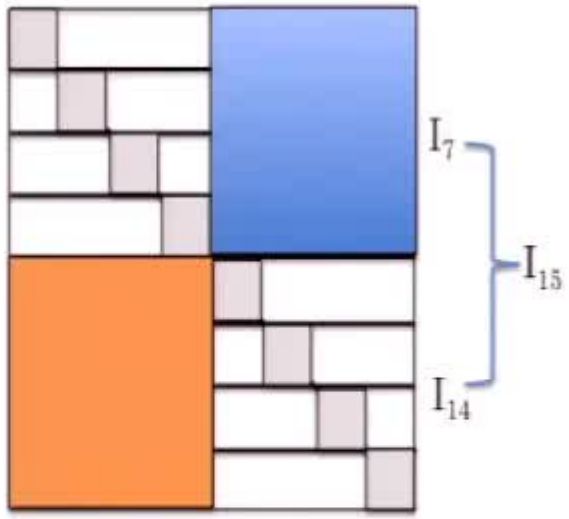
SIAM Conference on Applied Linear Algebra, Oct. 26-30, 2015

1. Cluster tree, block cluster tree

Cluster tree $T_{\mathcal{I}}$ defines hierarchical partitioning of the index set $[1, n]$.

- Each node is associated with an interval $\mathcal{I}_\tau$.
- For children ν_1 and ν_2, parent $\mathcal{I}_\tau = \mathcal{I}_{\nu_1} \cup \mathcal{I}_{\nu_2}$, and $\mathcal{I}_{\nu_1} \cap \mathcal{I}_{\nu_2} = \emptyset$





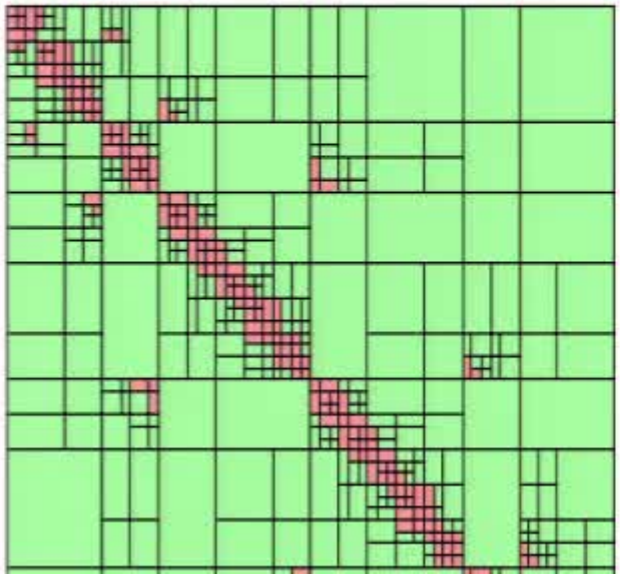
Block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$ defines partitioning of the index set $\mathcal{I} \times \mathcal{J}$, both

Families of $\mathcal{H}$ and $\mathcal{H}^2$ matrices

- Admissibile block (τ, σ): $\max\{\text{diam}(\tau), \text{diam}(\sigma)\} \leq \eta \, \text{dist}(\tau, \sigma)$
- Strong admissibility: blocks next to diagonal not compressed, only compress well separated blocks

- $\mathcal{H}$: split a node in a block cluster tree if its block is admissible
- $\mathcal{H}^2$: uniform $\mathcal{H}$ partitioning, with nested bases

[Börn/Grasedyck/Hackbusch]



Various data-sparse formats

Method	Hier. part.	Nested bases	Admissibility	Family
HODLR	yes	no	weak	$\mathcal{H}$
HSS/HBS	yes	yes	weak	$\mathcal{H}^2$
Barnes-Hut	yes	no	strong	$\mathcal{H}$
FMM	yes	yes	strong	$\mathcal{H}^2$
BLR	no	no	weak	

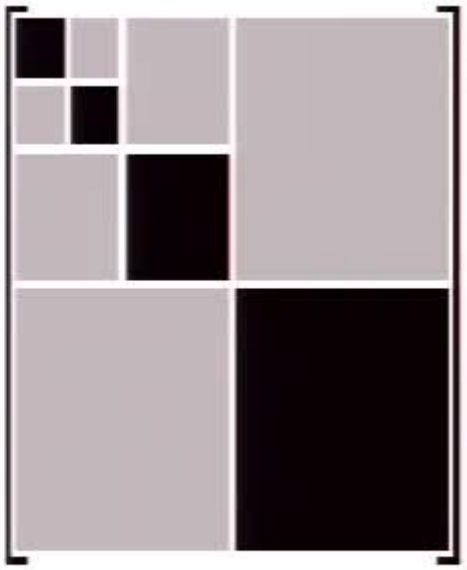
Practical comparisons ... two other talks

- Francois-Henry Rouet, MS39, Thursday, 11:45-12:10
  "A Comparison of Different Low-Rank Approximation Techniques"

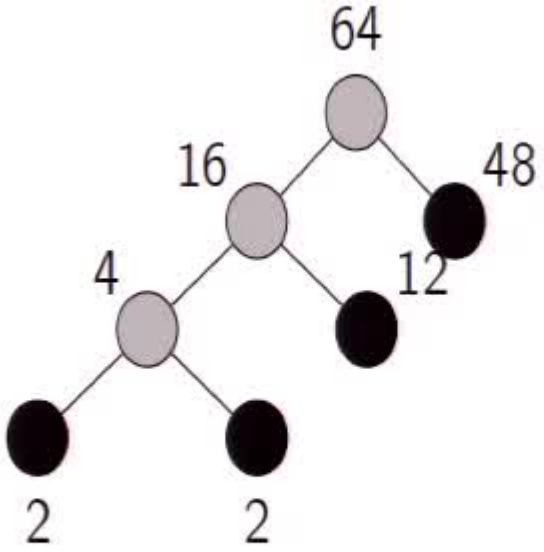
- Rio Yokota, MS45, Thursday, 3:30-3:55
  "Comparison of FMM and HSS at Large Scale"

Making software robust

- Adaptive sampling machinery
  - Automatic handling unknown rank patterns: incrementally adjust sample size at any node when rank revealed is too large.
- Non-uniform clustering & partitioning



(c) Matrix structure.



(d) Weighted process mapping

Parallel weak scaling

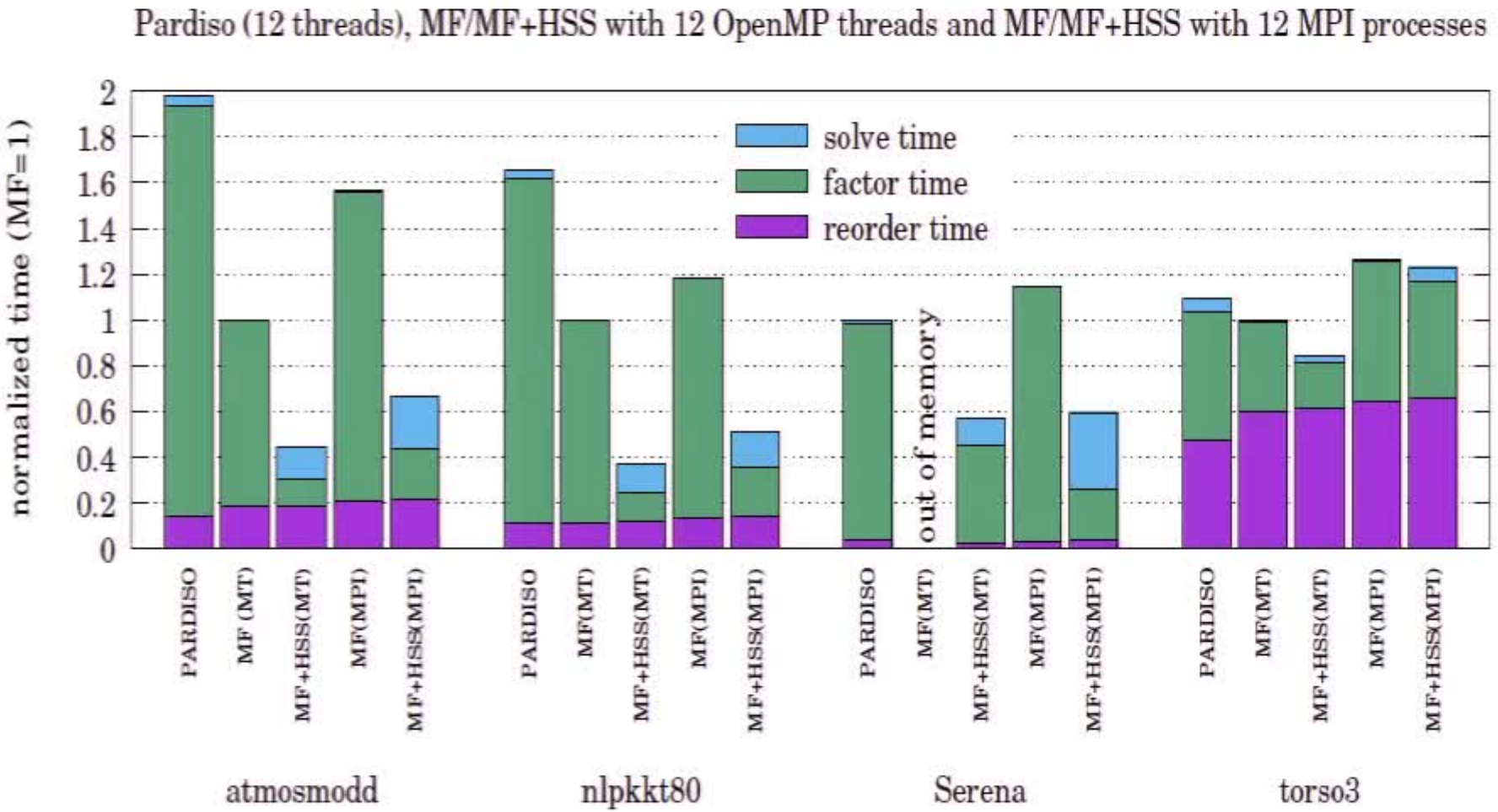
Root node of the multifrontal factorization of a discretized Helmholtz problem (frequency domain, PML boundary, 10Hz).

k (3D mesh: k^3)	100	200	300	400	500
Matrix size ($=k^2$)	10,000	40,000	90,000	160,000	250,000
# Cores	64	256	1,024	4,096	8,192
Maximum rank	313	638	903	1289	1625
Compression time	2.0	13.0	30.6	60.8	133.6
Speed-up over ScaLAPACK	**1.8**	**4.0**	**5.4**	**4.8**	**3.9**
Flops ratio	**0.6**	**18.8**	**132.7**	**626.1**	**1716.7**

Load imbalance

Parallel performance of sparse MF-HSS solvers

Cray XC30, Edison at NERSC



- Compared to PARDISO in Intel MKL library (12 threads)

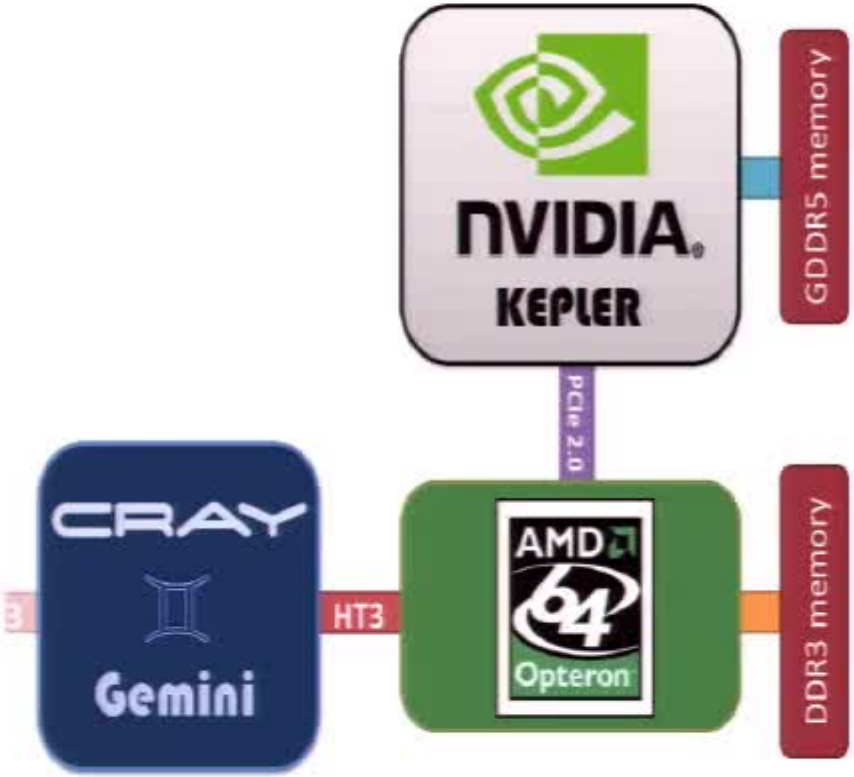
Evolution of parallel machines, programming

- Vector machines, program with vectorization directives
- Shared memory UMA, program with directives or explicit threading
- Distributed memory machines presented major challenges
  - Data distribution, locality
  - Program with explicit messages, e.g., MPI
- Recently, heterogeneous node architectures, more disruptive
  - NUMA, socket / core / vector unit; accelerator / co-processor (e.g., GPU)
  - Memory per core is small
  - Program with mixed MPI & threads & CUDA ...

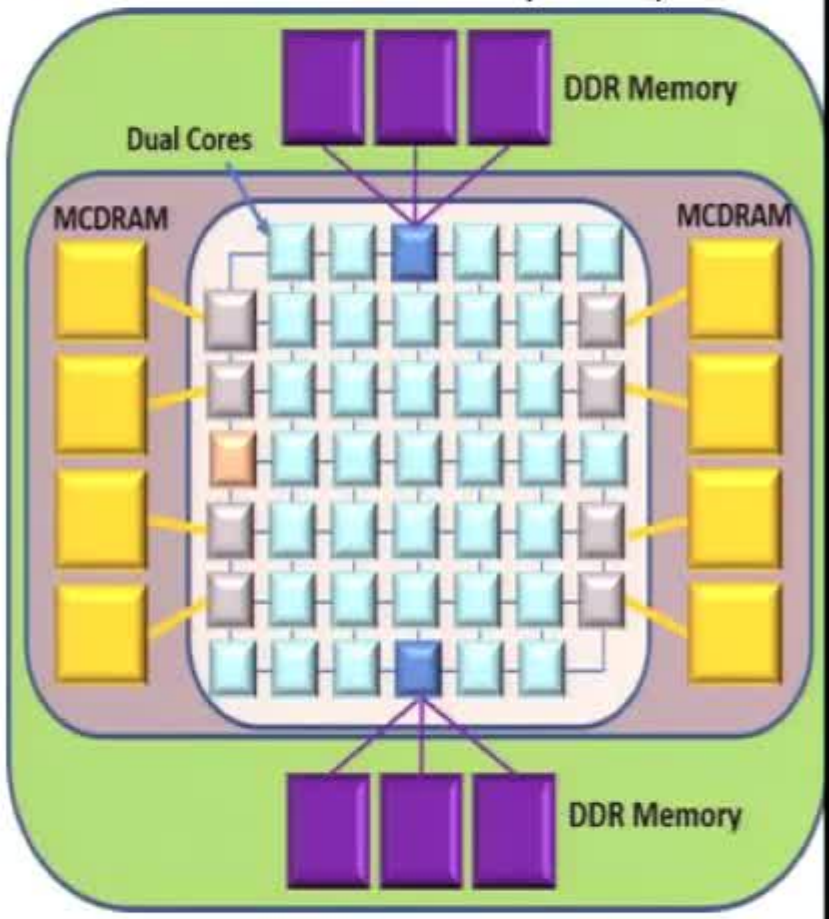
Mixing task parallelism and data parallelism.

Variety of node architectures

Titan at ORNL:
16-core AMD + K20X GPU



Intel Xeon Phi KNL (2016)



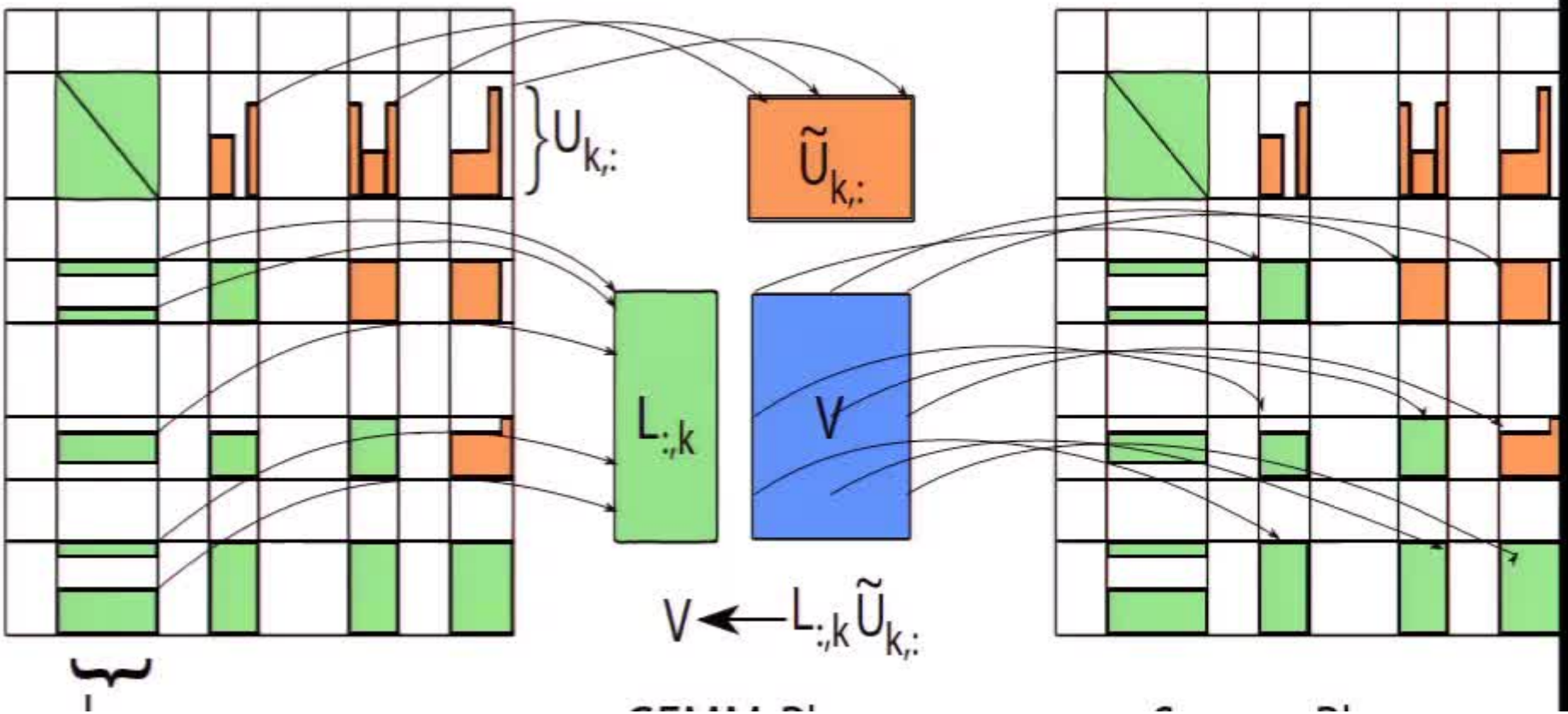
Programming:

- Separate CPU/GPU programs

- 72 cores
- 4 threads/core

Schur-complement udate

- Over 80% factorization time, ample parallelism
- Two operations: GEMM, Gather/Scatter



Design questions for accelerator / co-processor

- Only use accelerator, or use CPU as well?
  Accelerator memory small
  $\rightarrow$ best to use both (offload some computations to GPU)
- What to offload?
  Panel factorization not suitable for fine-grained data-parallel model
  $\rightarrow$ offload only Schur complement update
- Schur complement update: GEMM, and Gather/Scatter?
  - GEMM only compute intensive [Sao/Vuduc/L. 2014]
  - Both GEMM and Gather/Scatter indirect addressing, memory intensive [Sao/Liu/Vuduc/L. 2015]

Overlap activities on both CPU/GPU to hide transfer latency over PCIe bus (10-15 microseconds)

Further details ...

- Piyush Sao, MS48, Thursday, 4:00-4:25
  "A Sparse Direct Solver for Distributed Memory GPU and Xeon Phi Accelerated Systems"

Other sparse factorization GPU work:

- PARDISO: left-looking sparse LU, offload BLAS [Schenk/Christen/Burkhart, 2008]
- WSMP: multifrontal sparse Cholesky, offload BLAS [George et al., 2011]
- Multifrontal sparse Cholesky, offload frontal matrix computation [Krawezik and Poole 2009, Vuduc et al. 2011, Yu/Wang/Pierce 2011]
- Threading, offload large frontal matrix computation [Lucas et al., 2010]