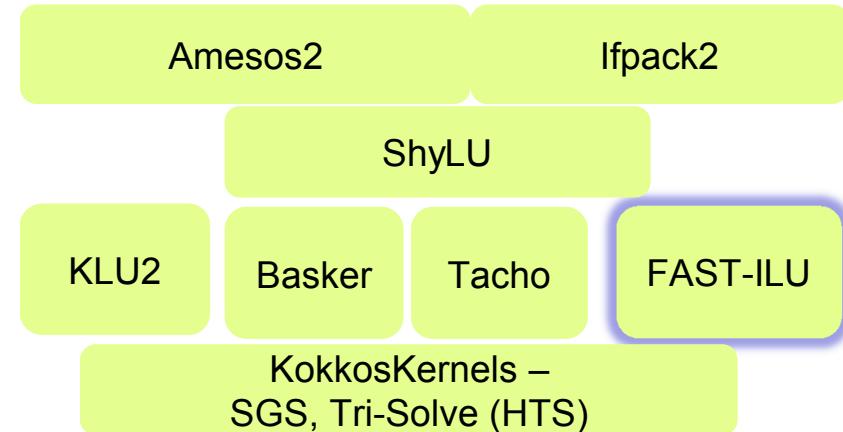
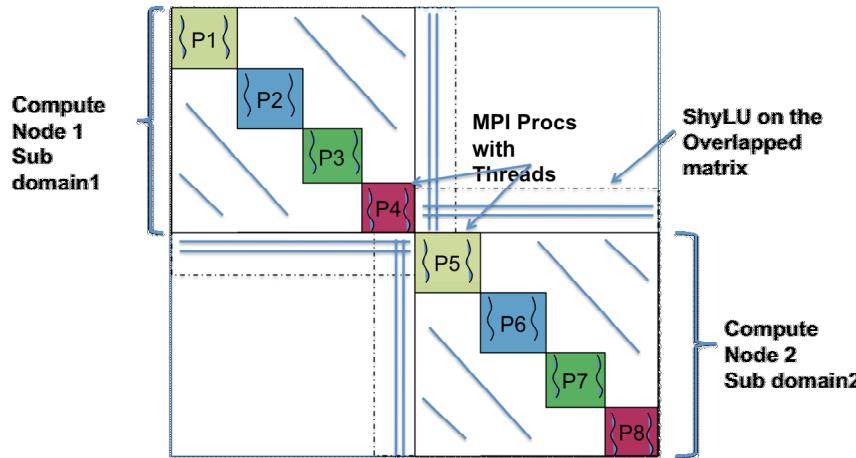

Basker : A Threaded Sparse LU factorization utilizing Hierarchical Parallelism and Data Layouts

Siva Rajamanickam
Joshua Booth, Heidi Thornquist

Sixth International Workshop on Accelerators and Hybrid Exascale Systems (IPDPS 2016)

ShyLU and Subdomain Solvers : Overview



- MPI+X based subdomain solvers
 - Decouple the notion of one MPI rank as one subdomain: Subdomains can span multiple MPI ranks each with its own subdomain solver using X or MPI+X
- **Subpackages of ShyLU:** Multiple Kokkos-based options for on-node parallelism
 - **Basker** : LU or ILU (t) factorization
 - **Tacho**: Incomplete Cholesky - IC (k) (See K.Kim talk in HIPS workshop later today)
 - **Fast-ILU**: Fast-ILU factorization for GPUs
- **KokkosKernels**: Coloring based Gauss-Seidel (see talk by M. Deveci), Triangular Solves
- Under active development. Jointly funded by ASC, ATDM, FASTMath, LDRD.

Themes for Architecture Aware Solvers and Kernels : Data layouts

- Specialized memory layouts
 - Architecture aware data layouts
 - Coalesced memory access
 - Padding
 - Array of Structures vs Structure of Arrays
 - Kokkos based abstractions (H. C. Edwards et al.)
- Two dimensional layouts for matrices
 - Allows using 2D algorithms for solvers and kernels
 - Bonus: Fewer synchronizations with 2D algorithms
 - Cons : Much more harder to design correctly
 - Better utilization of hierarchical memory like High Bandwidth Memory (HBM) in Intel Xeon Phi or NVRAM
- Hybrid layouts
 - Better for very heterogeneous problems

Themes for Architecture Aware Solvers and Kernels : Fine-grained Synchronization

- Synchronizations are expensive
 - 1D algorithms for factorizations and solvers, such as ND based solvers have a huge synchronization bottleneck for the final separator
 - Impossible to do efficiently in certain architectures designed for massive data parallelism (GPUs)
 - This is true only for global synchronizations, fork/join style model.
- Fine grained synchronizations
 - Between handful of threads (teams of threads)
 - Point to Point Synchronizations instead of global synchronizations
 - Park et al (ISC14) showed this for triangular solve
 - Thread parallel reductions wherever possible
 - Atomics are cheap
 - Only when used judiciously

Themes for Architecture Aware Solvers and Kernels : Task Parallelism

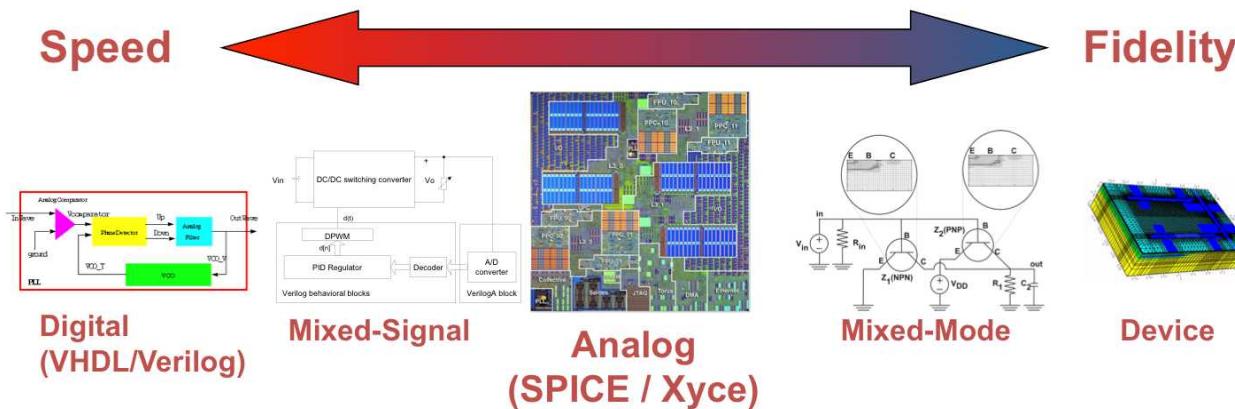
- Statically Scheduled Tasks
 - Determine the static scheduling of tasks based on a task graph
 - Eliminate unnecessary synchronizations
 - Tasks scheduled in the same thread do not need to synchronize
 - Find transitive relationships to reduce synchronization even further
 - Jongsoo Park et al
- Dynamically scheduled tasks
 - Use a tasking model that allows fine grained synchronizations
 - Requires support for futures
 - Not the fork-join model where the parent forks a set of tasks and blocks till they finish
 - Kokkos Tasking API
 - Joint work with Carter Edwards, Stephen Olivier, Kyungjoo Kim, Jon Berry, George Stelle
 - See K. Kim's talk in HIPS for a comparison with this style of codes

Themes for Architecture Aware Solvers and Kernels : Asynchronous Algorithms

- System Level Algorithms
 - Communication Avoiding Methods (s-step methods)
 - Not truly asynchronous but can be done asynchronously as well.
 - Multiple authors from early 1980s
 - Pipelined Krylov Methods
 - Recently Ghysels, W. Vanroose et al. and others
- Node Level Algorithms
 - Finegrained Asynchronous iterative ILU factorizations
 - An iterative algorithm to compute ILU factorization (Chow et al)
 - Asynchronous in the updates
 - Finegrained Asynchronous iterative Triangular solves
 - Jacobi iterations for the triangular solve.

Why Transistor-Level Circuit Simulation?

- Provides tradeoff between fidelity and speed/problem size
 - Xyce enables full system parallel simulation for large integrated circuits
- Essential simulation approach used to verify electrical designs
 - SPICE is the defacto industry standard (PSpice, HSPICE, etc.)
 - Xyce supports NW-specific device development

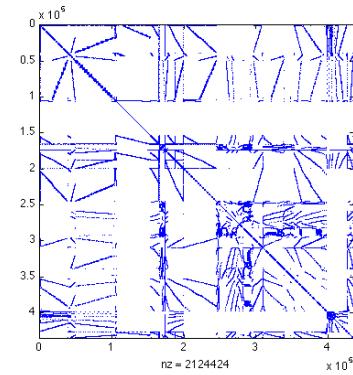
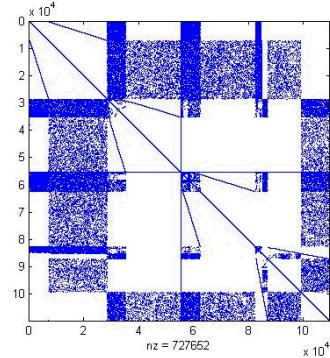


Simulation Challenges

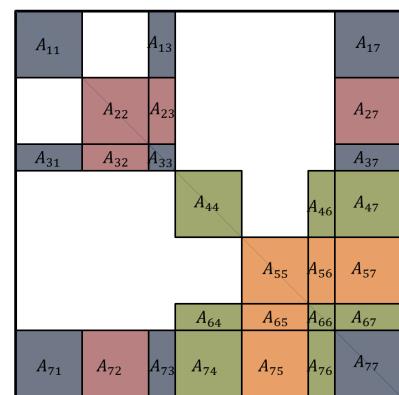
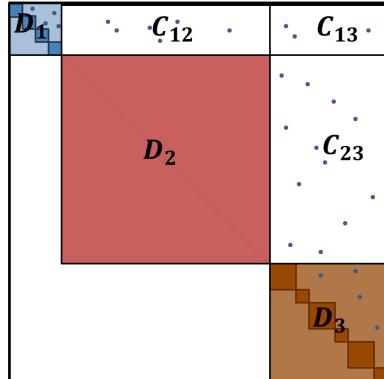
Analog simulation models network(s) of devices coupled via
Kirchoff's current and voltage laws

$$f(x(t)) + \frac{dq(x(t))}{dt} = b(t)$$

- Network Connectivity
 - Hierarchical structure rather than spatial topology
 - Densely connected nodes: $O(n)$
- Badly Scaled DAEs
 - Compact models designed by engineers, not numerical analysts!
 - Steady-state (DCOP) matrices are often ill-conditioned
- Non-Symmetric Matrices
- Load Balancing vs. Matrix Partitioning
 - Balancing cost of loading Jacobian values unrelated to matrix partitioning for solves
- **Strong scaling and robustness is the key challenge!**

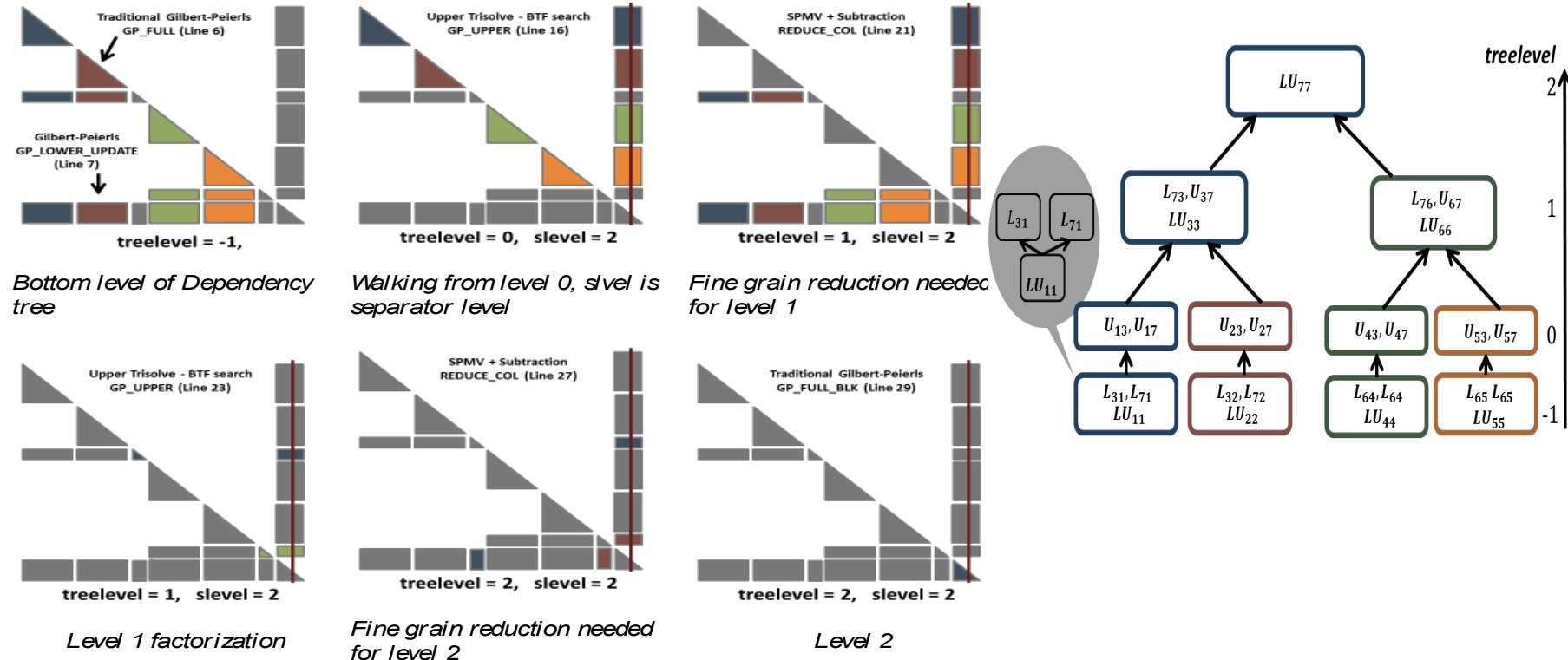


ShyLU/Basker : LU factorization



- Basker: Sparse (I)LU factorization
 - Block Triangular form (BTF) based LU factorization, Nested-Dissection on large BTF components
 - 2D layout of coarse and fine grained blocks
 - Previous work: X. Li et al, Rothberg & Gupta
 - Data-Parallel, Kokkos based implementation
 - Fine-grained parallel algorithm with P2P synchronizations
 - Parallel version of Gilbert-Pereils' algorithm (or KLU)
 - Left-looking 2D algorithm requires careful synchronization between the threads
 - All reduce operations between threads to avoid atomic updates

ShyLU/Basket : Steps in a Left looking factoization

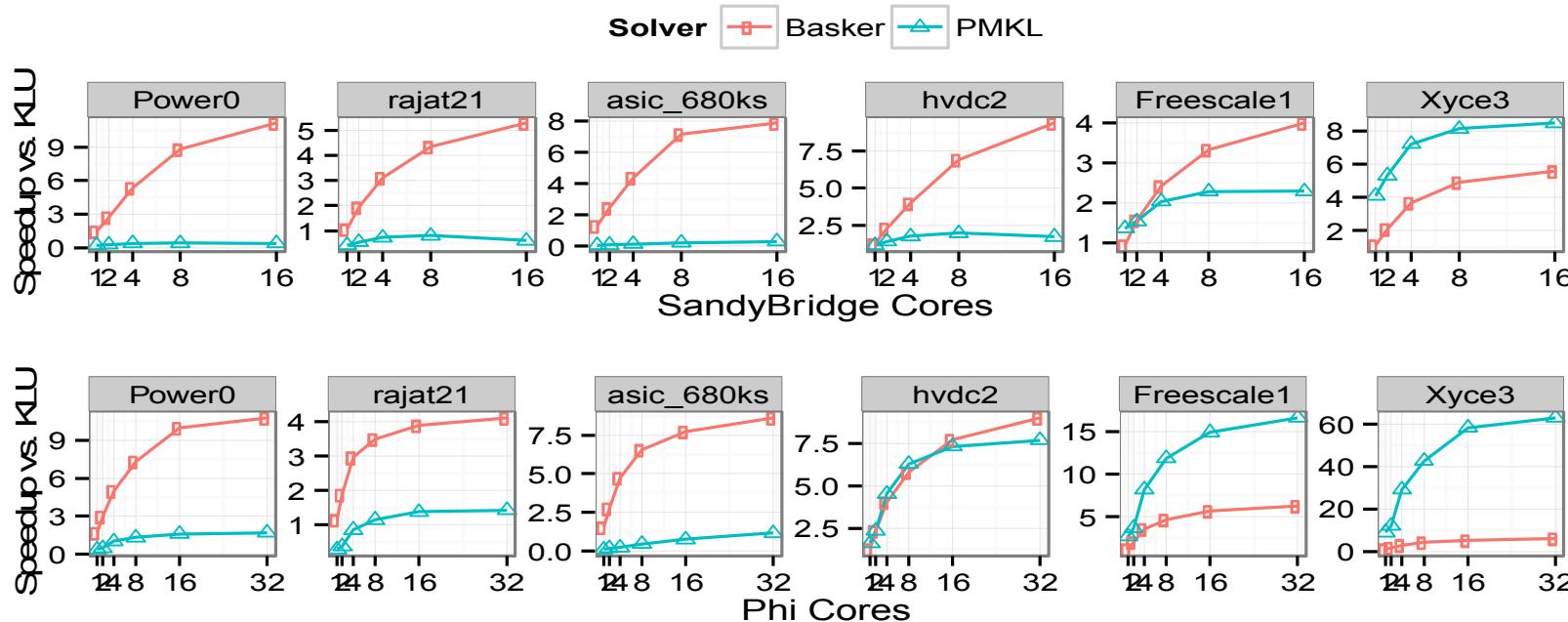


- Different Colors show different threads
- Grey means not active at any particular step
- Every left looking factorization for the final separator shown here involves four independent triangular solve, a mat-vec and updates (P2P communication), two independent triangular solves, a mat-vec and updates, and triangular solve. (Walking up the nested-dissection tree)

ShyLU/Basker : Performance Results

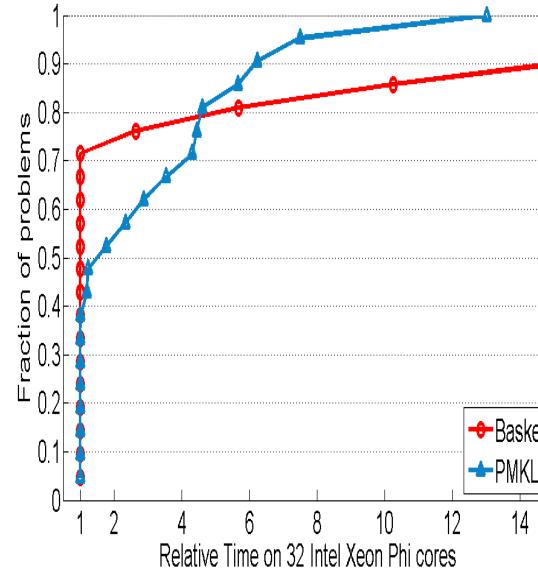
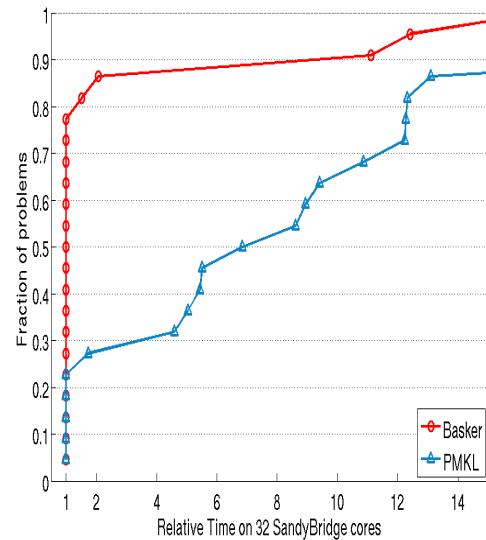
- A set of problems selected from UF Sparse Matrix Collection and Sandia's internal problem set
 - Representative problems with both high fill (fill-in density > 4.0) and low fill-in density
- OpenMP and Kokkos based implementation for CPU and Xeon Phi based architectures
- Testbed Cluster at Sandia
 - SandyBridge based two eight core Xeons (E5-2670), 24GB of DRAM
 - Intel Xeon Phi (KNC) co-processors with 61 cores with 16 GB main memory
- The number of non-zeros between the solvers are different due to the different ordering schemes used by the solvers
- Comparisons with KLU, SuperLU-MT and MKL-PARDISO

ShyLU/Basker : Performance Results



- Speedup 5.9x (CPU) and 7.4x (Xeon Phi) over KLU (Geom-Mean) and up to 53x (CPU) and 13x (Xeon Phi) over MKL
- Low-fill matrices Basker is consistently the better solver. High fill matrices MKL Pardiso is consistently the better solver

ShyLU/Basker : Performance Results



- Performance Profile for a matrix set with a high-fill and low-fill matrices shown (16 threads on CPUs /32 threads on Xeon Phi)
- Low-fill matrices Basker is consistently the better solver. High fill matrices MKL Pardiso is consistently the better solver

Conclusions

- Themes around Thread Scalable Subdomain solvers
 - Data Layouts
 - Fine-grained Synchronizations
 - Task Parallelism
 - Asynchronous Algorithms
- Basker LU factorization
 - Uses 2D layouts with hierarchy from block triangular form and nested dissection
 - Uses fine-grained synchronizations between teams of threads
 - Uses a static tasking mechanism with data-parallelism

Questions ?