Present and Future Computing Requirements for SuperLU and TOORSES

X. Sherry Li

Scientific Computing Group, LBNL

NERSC ASCR Requirements for 2017
January 15, 2014
LBNL

1. Project Description (1 of 2)

PI: Sherry Li, LBNL

- Summarize your project(s) and its scientific objectives through 2017
  - SuperLU is a direct solver library for sparse linear systems
  - Most parallel one is SuperLU_DIST, MPI-only at present
  - Many users: 27,403 downloads in FY13
  - Included in Cray's LibSci, FEMLAB, HP's MathLib, IMSL, NAG, OptimaNumerics, Python (SciPy)
  - Objectives through 2017:
    - Strong scale to 10K nodes with hybrid programming
    - 50M-100M dof (presently 10M-20M)
- Solver's characteristics:
  - Numerically robust, no convergence issue
  - Requires large memory, unless use disk with I/O

1. Project Description (2 of 2)

PI: Sherry Li, LBNL

- Present focus is to restructure the algorithm and code to explore hierarchical parallelism through hybrid programming, including accelerators like GPU, MIC
- By 2017 we expect to
  - Flexible hybrid code: MPI + OpenMP + CUDA/OpenCL/xx
    - Preliminary results on Dirac show 3x faster than MPI-only using 8 CPU cores + 1 GPU
  - Capable of using any heterogeneous architectures
  - ABFT resilience with fault detection & recovery

2. Computational Strategies

- These codes are characterized by these algorithms:
  - Sparse LU, sparse triangular solve
  - Supernode partition, 2D block cyclic matrix/process distribution
  - Pre-pivoting via weighted maximum bipartite matching algorithm
  - Sparsity ordering with parallel graph partitioning: ParMETIS, PT-Scotch, Zoltan
  - Parallel symbolic factorization
- Our biggest computational challenges are:
  - Task & data dependency (esp. triangular solve)
  - Low arithmetic intensity
  - Pre-pivoting is the serial bottleneck
- We expect our computational approach and/or codes to change by 2017:
  - Alternative to pivoting: Random Butterfly Transformation (RBT)
  - Expose more data parallelism to utilize GPU, MIC, etc.

3. Current HPC Usage

- Machines currently using (NERSC or elsewhere)
  - NERSC: hopper, dirac, edison
- Hours used in 2012-2013 (list different facilities)
  - Developers: mostly NERSC, ~100K hours
  - Users: many other facilities
- Typical parallel concurrency and run time, number of runs per year
  - 100s – 1000s cores, minutes – couple of hours, thousands runs per year
- Data read/written per run
  - Developers: sequentially read a sparse matrix from a file, in a few minutes
  - Users: generate matrix on the fly
- Memory used per (node | core | globally)
  - Can use up all the memory
  - Serial pre-pivoting requires matrix A to fit on one node
- Necessary software, services or infrastructure
  - MPI, BLAS
- Data resources used (/scratch,HPSS, NERSC Global File System, etc.)
  - /scratch and HPSS are used to store test matrices (~5 GB one large matrix)

4. HPC Requirements for 2017

- Compute hours needed (in units of Hopper hours)
  - 500K
- Changes to parallel concurrency, run time, number of runs per year
  - 10-20x more concurrency, similar run time and number of runs per year
- Changes to data read/written
  - No, currently no plan to develop out-of-core capability
- Changes to memory needed per (core | node | globally)
  - May reduce per-node memory if found good pre-pivoting alternatives: parallel maximum matching, or RBT, . . .
- Changes to necessary software, services or infrastructure
  - No

5. Strategies for New Architectures (1 of 2)

- Does your software have CUDA/OpenCL directives; if yes, are they used, and if not, are there plans for this?
  - Started CUDA development. AMD is interested in developing OpenCL
- Does your software run in production now on Titan using the GPUs?
  - Expect soon
- Does your software have OpenMP directives now; if yes, are they used, and if not, are there plans for this?
  - Yes
- Does your software run in production now on Mira or Sequoia using threading?
  - Not yet
- Is porting to, and optimizing for, the Intel MIC architecture underway or planned?
  - Not yet, but have great interests

5. Strategies for New Architectures (2 of 2)

- Have there been or are there now other funded groups or researchers engaged to help with these activities?
  - Collaborating with Rich Vuduc's group of Georgia Tech for manycore developments and energy-aware algorithms
- What role should NERSC play in the transition to these architectures?
  - Help on-node performance models, performance tools for threads, GPUs
- What role should DOE and ASCR play in the transition to these architectures?
  - Funding for code development
- Other needs or considerations or comments on transition to manycore:
  - How to speed up non-BLAS-like operations: scattering, graph traversal

TOORSES (Towards Optimal Order Resilient Solvers at Extreme Scale)

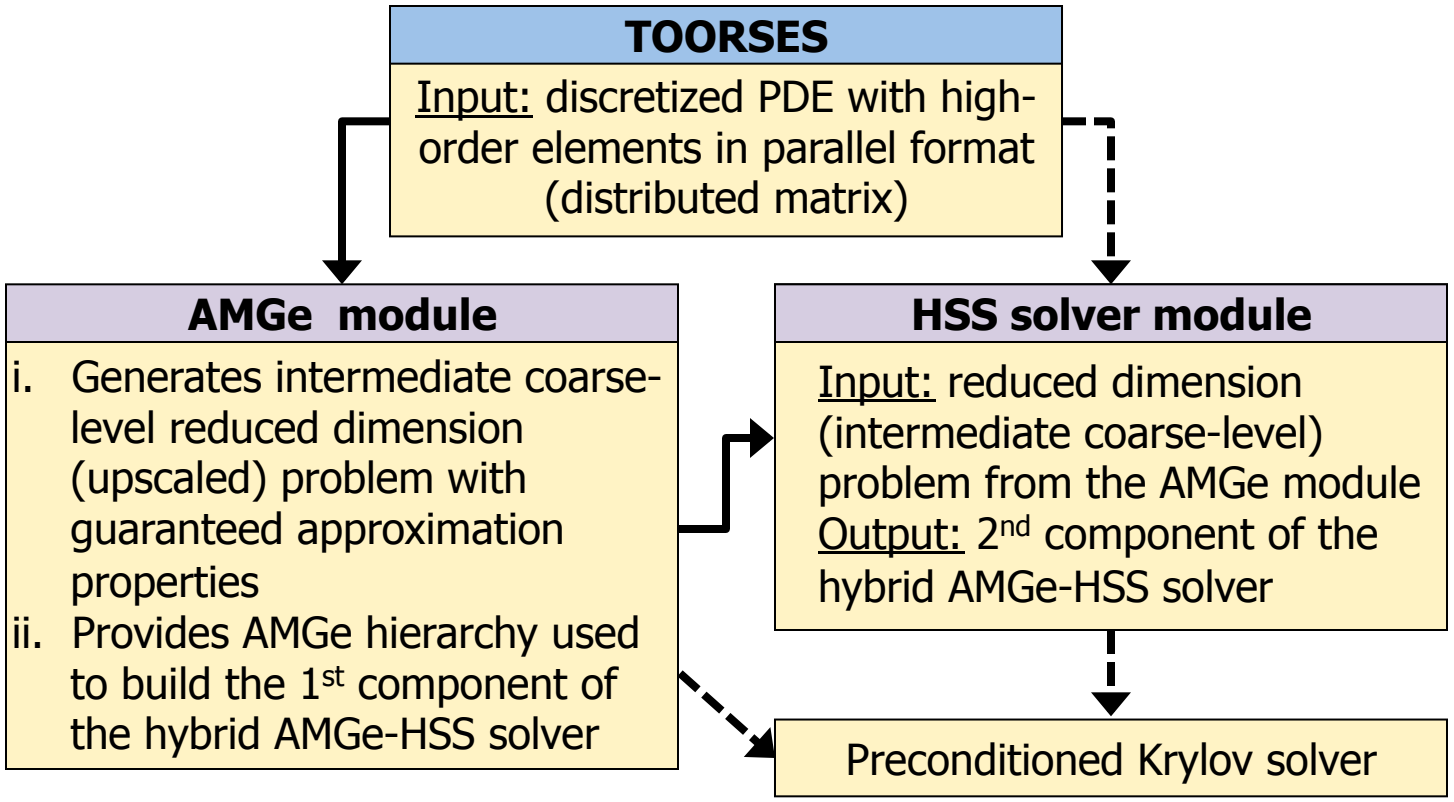
("RX-Solvers" in ASCR Applied Math)

PI: Sherry Li , LBNL, co-PI: Panayot Vassilevski, LLNL

- Goal: develop hybrid, optimal order resilient solvers and preconditioners for unstructured, broad classes of large PDEs.

- Two approaches combined:
  - Algebraic multigrid with higher-order elements
  - Coarse level uses low-rank sparse factorization with hierarchically semi-separable (HSS) matrix representation
- Both methods achieve optimal-order complexity for certain PDEs; both algorithms are organized in multilevel, hierarchical fashion
  - Matches well with extreme-scale hardware: hierarchical memory, NUMA nodes
  - ABFT can be built in a hierarchical manner
- Collaborate with DEGAS X-Stack to implement resilient algorithms
  - Hierarchical programming model, resilience through containment

TOORSES Solver Structure



Composable & flexible usage

1) AMG solver alone
2) HSS solver alone
3) Hybrid AMG-HSS solver
4) AMG preconditioned Krylov
5) HSS preconditioned Krylov
6) AMG+HSS preconditioned Krylov

TOORSES Algorithmic Challenges for Manycores

- AMGe module
  - Mesh partitioning for element agglomeration
    - Adapt graph partitioning packages
  - Triple sparse matrix product $P^T AP$ to build multigrid hierarchy
  - Restriction & prolongation between levels
- HSS solver module
  - Even smaller dense blocks than SuperLU
  - Load imbalance: numerical ranks differ between levels and among nodes within the same level of the separator and HSS trees
  - Adaptivity to handle rank resolution dynamically