Static-scheduling and hybrid-programming in SuperLU_DIST on multicore cluster systems

Ichitaro Yamazaki
University of Tennessee, Knoxville

Xiaoye Sherry Li
Lawrence Berkeley National Laboratory

MS49: Sparse Linear Solvers on Many-core Architectures
SIAM PP: Savannah, 02/17/2012

SuperLU_DIST: direct solver for general sparse linear systems on a distributed memory system

- first release in 1999
  - each compute node with 1+ cores and UMA.
- capable of factorizing matrices with millions of unknowns from real applications.
- used in large-scale simulations: iterative/hybrid solvers

▷ quantum mechanics [SC'01]: low-order uncoupled systems
▷ fusion energy (M3D-C^1, PPPL): 2D slices of 3D torus

PDSLin: hybrid linear solver
▷ domain decomposition
▷ inetrior subdomain solver
▷ studied for accelerator modeling (Omega3P, SLAC)

Our testbeds at NERSC

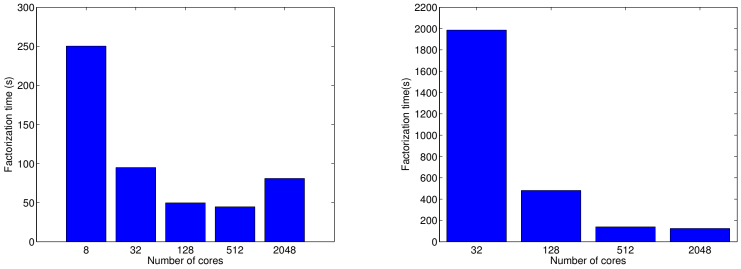
Cray-XE6 (Hopper):

- 6,384 nodes (peak 1.28Pflop/s), No. 8 on TOP500.
- two 12-core AMD MagnyCours (two six-core Bulldozer) 2.1GHz processors + 32GB of memory per node.
- Cray Gemini Network in a 3D torus.

IBM iDatPlex (Caver)

- 1,202 nodes (peak 0.11Pflops/sec), max. 64 nodes for a parallel job.
- two quad-core Intel Nehalem 2.7GHz processors + ~20GB of memory per node.
- 4x QDR InfiniBand in a 2D mesh.

SuperLU_DIST version 2.5 (released Nov. 2010) on Cray-XE6



(a) accelerator (sym), $n = 2.7M$, fill-ratio$= 12$ (b) DNA (unsym), $n = 445K$, fill-ratio$= 609$

- SuperLU_DIST often does not scale to thousands of cores.
- Why, and can we improve its performance?

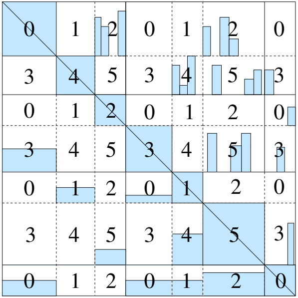
Outline:

- Introduction to SuperLU_DIST
- Static scheduling scheme
  - exploit more parallelism and reduce idle time.
  - obtain speedups of upto 2.6 on upto 2048 cores.
- Hybrid programming paradigm
  - reduce memory overhead and obtain similar parallel efficiency.
  - utilize more cores per node and reduce factorization time.

SuperLU_DIST: steps to solution

Compute factorization in three-stages:

1. Matrix preprocessing:
   - static pivoting/scaling/permutation to improve numerical stability and to preseve sparsity
2. Symbolic factorization:
   - computation of e-tree/structure of LU and static comm./comp. schedulings
   - supernodal (6-50 cols) for efficient dense block operations
3. Numerical factorization:
   - fan-out (right-looking, outer-product)
   - 2D cyclic MPI grid

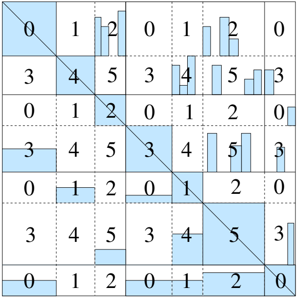


Compute solution with forward and backward substitutions.

SuperLU_DIST: steps to solution

Compute factorization in three-stages:

1. Matrix preprocessing:
   - static pivoting/scaling/permutation to improve numerical stability and to preseve sparsity
2. Symbolic factorization:
   - computation of e-tree/structure of LU and static comm./comp. scheduling
   - supernodal (6-50 cols) for efficient dense block operations
3. Numerical factorization: ← dominate
   - fan-out (right-looking, outer-product)
   - 2D cyclic MPI grid



Compute solution with forward and backward substitutions.

SuperLU_DIST: numerical factorization

fan-out (right-looking) factorization

for $j = 1, 2, \ldots, n_s$

  panel factorization (column and row)

    factor $A_{j,j}$ and

      isend to $P_C(k)$ and $P_R(k)$

    wait for $A_{j,j}$ and

      factor $A_{(j+1):n_s,j}$ and send to $P_R(:)$

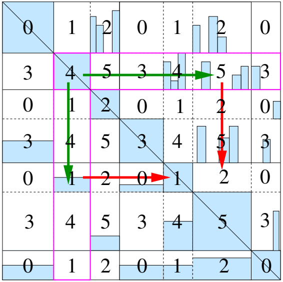
    wait for $A_{j,j}$ and

      factor $A_{j,(j+1):n_s}$ and send to $P_C(:)$

  trailing matrix update

    update $A_{(j+1):n_s,(j+1):n_s}$

end for

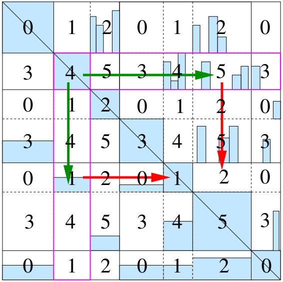


- high parallelism and good load-balance for trailing matrix updates, where most of computation time is spent.

SuperLU_DIST: numerical factorization

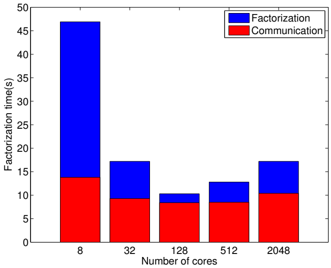
fan-out (right-looking) factorization

for $j = 1, 2, \ldots, n_s$
    panel factorization (column and row)
        factor $A_{j,j}$ and
          isend to $P_C(k)$ and $P_R(k)$
        wait for $A_{j,j}$ and
          factor $A_{(j+1):n_s,j}$ and send to $P_R(:)$
        wait for $A_{j,j}$ and
          factor $A_{j,(j+1):n_s}$ and send to $P_C(:)$
    trailing matrix update
        update $A_{(j+1):n_s,(j+1):n_s}$
end for

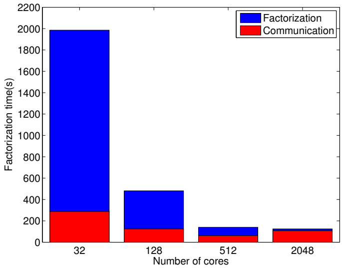


- sequential flow and limited parallelism at panel factorization
- poor scheduling causes processors being idle.

SuperLU_DIST version 2.5 (released Nov. 2010) on Cray-XE6



(c) accelerator (sym), $n = 2.7$M, fill-ratio$= 12$



(d) DNA, $n = 445$K, fill-ratio$= 609$

- synchronization dominates on a large number of cores (e.g., up to 96% of factorization time).

Outline:

- Introduction to SuperLU_DIST
- **Static scheduling scheme**
  - exploit more parallelism and reduce idle time.
  - obtain speedups of upto 2.6 on upto 2048 cores.
- Hybrid programming paradigm
  - reduce memory overhead and obtain similar parallel efficiency.
  - utilize more cores per node and reduce factorization time.

SuperLU_DIST: numerical factorization

fan-out (right-looking) factorization

for $j = 1, 2, \ldots, n_s$

panel factorization (column and row)

factor $A_{j,j}$ and

isend to $P_C(k)$ and $P_R(k)$

wait for $A_{j,j}$ and

factor $A_{(j+1):n_s,j}$ and send to $P_R(:)$

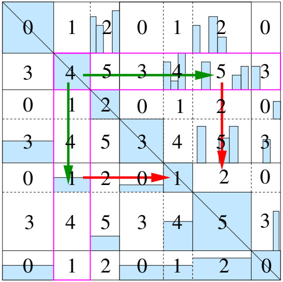
wait for $A_{j,j}$ and

factor $A_{j,(j+1):n_s}$ and send to $P_C(:)$

trailing matrix update

update $A_{(j+1):n_s,(j+1):n_s}$

end for



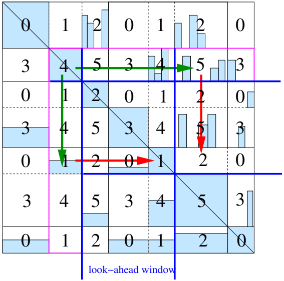
- synchronization dominates on a large number of cores.
  - processors in $P_C(k)$ and $P_R(k)$ wait for diagonal factorization.
  - all the processors wait for the panel factorization.
- due to sparsity, many panels won't be updated by remaining panels and are ready to be factorized.

Look-ahead in SuperLU_DIST with a fixed window size n_w

At each j-th step; factorize all "ready" panels in the window.

- reduce idle time of cores
- overlap comm. and comp.
- exploit more parallelism

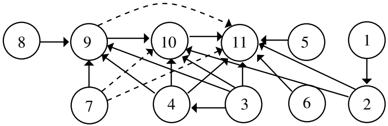
for $j = 1, 2, \ldots, n_s$
  look-ahead row factorization
  for $k = j+1, j+2, \ldots, j+n_w$ **do**
    if $U_{k,k}$ has arrived on $P_R(k)$ then
      factor $A_{k,(k+1):n_s}$ and isend to $P_C(:)$
  synchronizations
  wait for $U_{j,j}$ and factor $A_{j,j+1:n_s}$ if needed
  wait for $L_{:,j}$ and $U_{j,:}$
  look-ahead column factorization
  for $k = j+1, j+2, \ldots, j+n_w$ **do**
    update $A_{:,k}$
    if possible then
      factor $A_{k:n_s,k}$ and isend to $P_R(:)$
  trailing matrix update
  update remaining $A_{(j+n_w+1):n_s,(j+n_w+1):n_s}$
end for

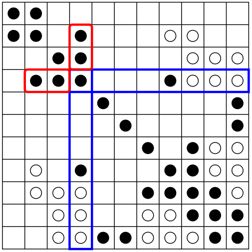


- factorization time is reduced only by 10%
- ready-for-factorize panels are not in window.
- performance depends on ordering of panels.

Keeping track of dependencies in SuperLU_DIST:

Directed acyclic graph (DAG)





- node for each panel factor. task
- edge (k, j) for dependency $k \to j$
- source/leaf = ready for factorization

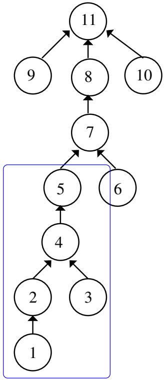
Unsymmetric case: symmetrically-pruned DAG

- identify smallest s_k such that $U(k, s_k)$ and $L(s_k, k)$ are the first symmetrically matched non-empty block for each k.
- add an edge (k, j) for all the non-empty $U(k, j)$ for $j \leq s_k$.
- add an edge (k, i) for all the non-empty $L(i, k)$ for $j \leq s_k$.

Symmetric case: elimination-tree (etree)

Symbolic factorization of SuperLU_DIST:

- order columns in postorder of etree
  (of $|A|^T + |A|$ for unsymmetric A)
  - larger supernodes
  - same structures/dependencies of LU
- setups data structures as supernodes are identified (postorder).
- uses same postordering during numerical factorization.
  - data locality
- limit number of ready panels in window
  - look-ahead only subtree



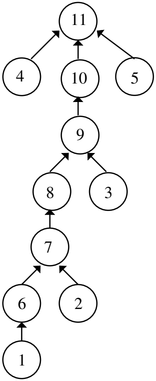
Static scheduling in SuperLU_DIST:

Symmetric case:

- keep track of dependencies using etree
- schedule tasks from leaves to root with higher-priority on the nodes further away from the root (bottom-up topological ordering)

Unsymmetric case:

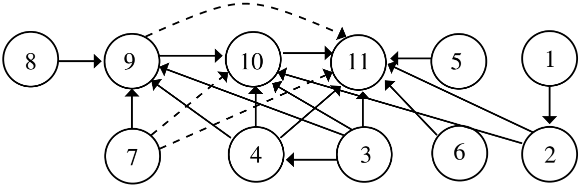
- use etree of $|A|^T + |A|$
- over-estimate dependencies for unsymmetric factorization



Static scheduling in SuperLU_DIST:

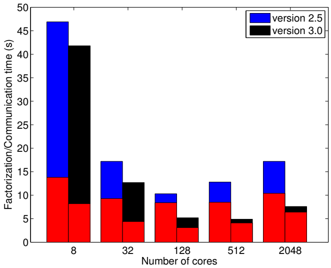
Unsymmetric case:

- symmetrically-pruned DAG to track both row and column dependencies.

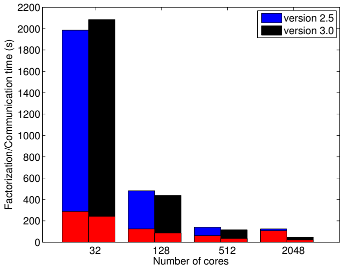


- "source" nodes represent ready-to-be-factorized columns.
- statically schedule tasks from sources to sinks with higher-priority on the nodes further away from a sink.

SuperLU_DIST version 2.5 and 3.0 on Cray-XE6



(e) accelerator (sym), $n = 2.7M$, fill-ratio= 12



(f) DNA (unsym), $n = 445K$, fill-ratio= 608

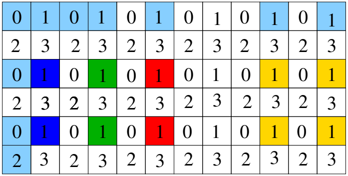
- idle time was significantly reduced (speedups of up to 2.6).
- To further improve the performance,
  - more sophisticated scheduling schemes
  - hybrid programming paradigms

Outline:

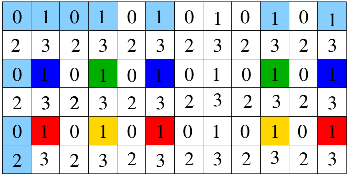
- Introduction to SuperLU_DIST
- Static scheduling scheme
  - exploit more parallelism and reduce idle time.
  - obtain speedups of upto 2.6 on upto 2048 cores.
- **Hybrid programming paradigm**
  - reduce memory overhead and obtain similar parallel efficiency.
  - utilize more cores per node and reduce factorization time.

Hybrid programming in SuperLU_DIST

- Computation is dominated by trailing matrix updates, where each MPI process updates independent supernodal blocks $\rightarrow$ use OpenMP threads to update these blocks



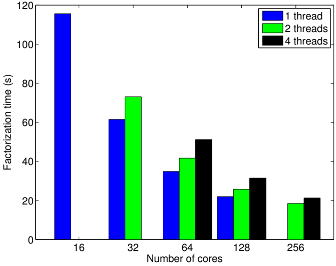
(g) 1D block if enough columns



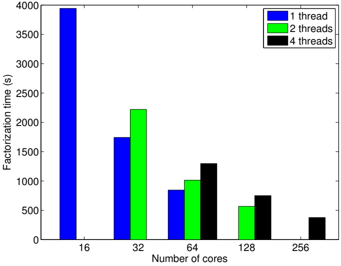
(h) 2D cyclic otherwise

Hybrid programming on top of SuperLU_DIST version 3.0

factorization times on 16 nodes of Cray-XE6.



(i) accelerator (sym), $n = 2.7$M, fill-ratio$= 12$



(j) DNA (unsym), $n = 445$K, fill-ratio$= 608$

- hybrid pradigm utilizes more cores on a node avoiding MPI overheads (memory/time).

Final remarks

- static scheduling reduced idle time with speedups of upto 2.6
  - available in version 3.0: `http://crd-legacy.lbl.gov/~xiaoye/SuperLU`.
- hybrid programming reduced MPI overheads, utilized more cores/node, and obtained speedups on same node count.
- more results are available in IPDPS'12 proceedings.

Thank you!!