SAND2015-9187C

Exceptional service in the national interest

Sandia National Laboratories

HTS: A Multithreaded Direct Sparse Triangular Solver Combining Level Scheduling and Recursive Blocking

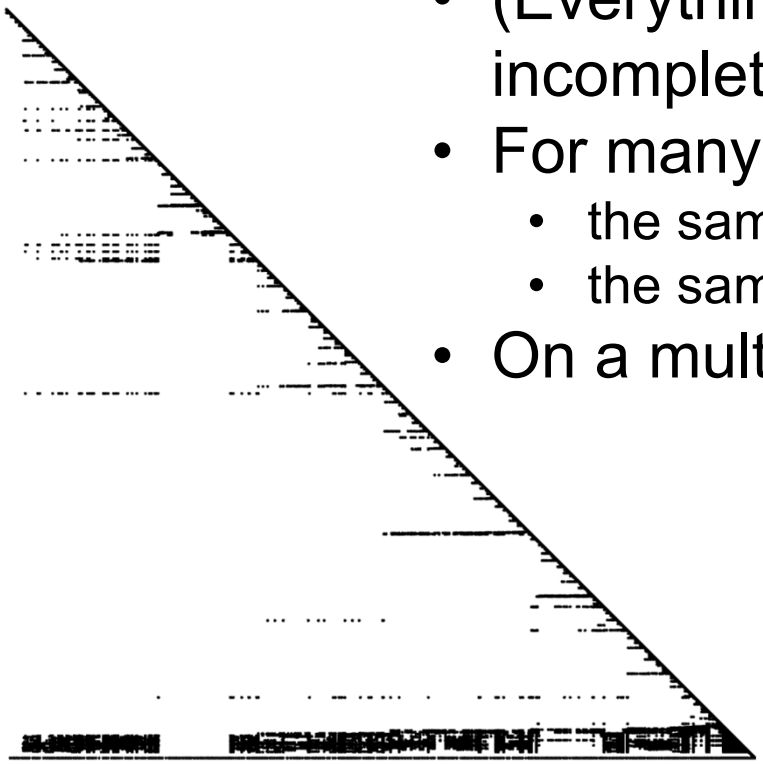
Andrew M. Bradley

Thanks: E. Boman, J. Booth, C. Dohrmann, S. Hammond, W. Held, M. Heroux, R. Hoekstra, K. Kim, S. Olivier, A. Prokopenko, S. Rajamanickam

SAND2015-XXXX

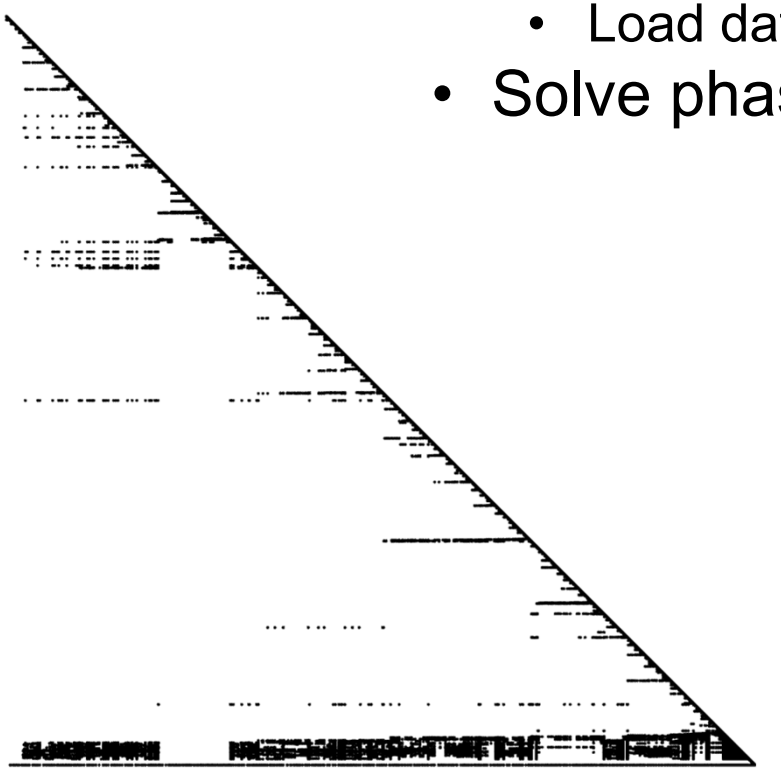
Problem Statement

- Solve R P T Q x = b
  - upper or lower sparse triangular matrix T
  - row scaling R
  - permutations P, Q
  - solution and RHS x, b
- (Everything that is needed for LDL, LU, incomplete factorizations, etc.)
- For many sequential RHS with
  - the same T
  - the same nonzero pattern pat(T)
- On a multi/many-core node

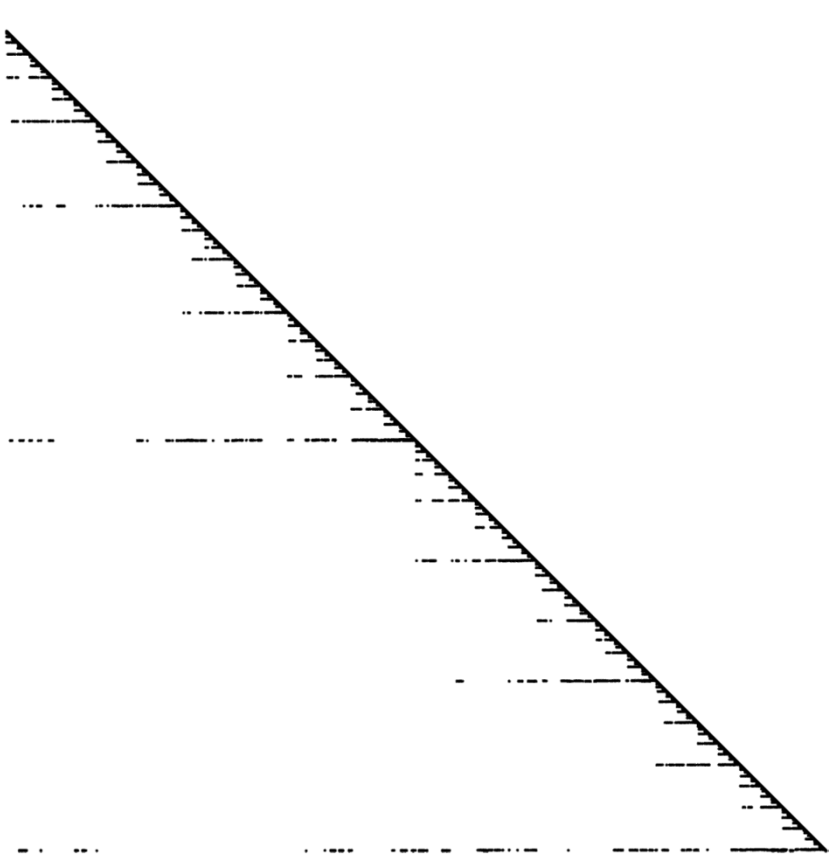


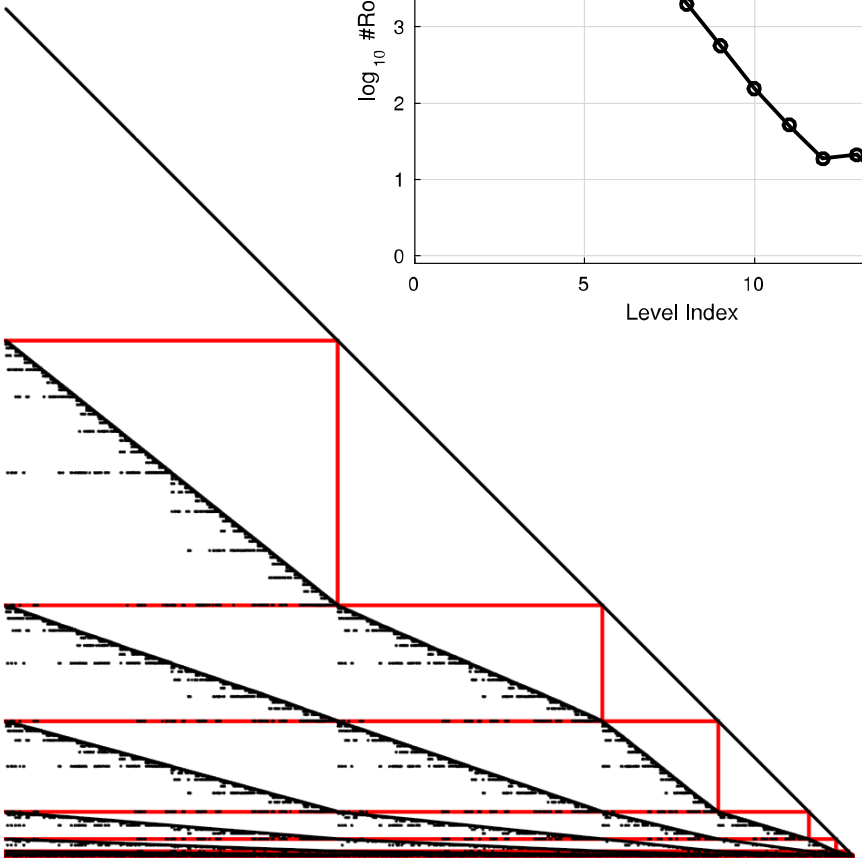
Solution Approach

- Direct
  - (Not iterative)
- Symbolic phase
  - Find parallelism in pat(T), the graph of T
- Numerical phase
  - Load data structures with numbers
- Solve phase



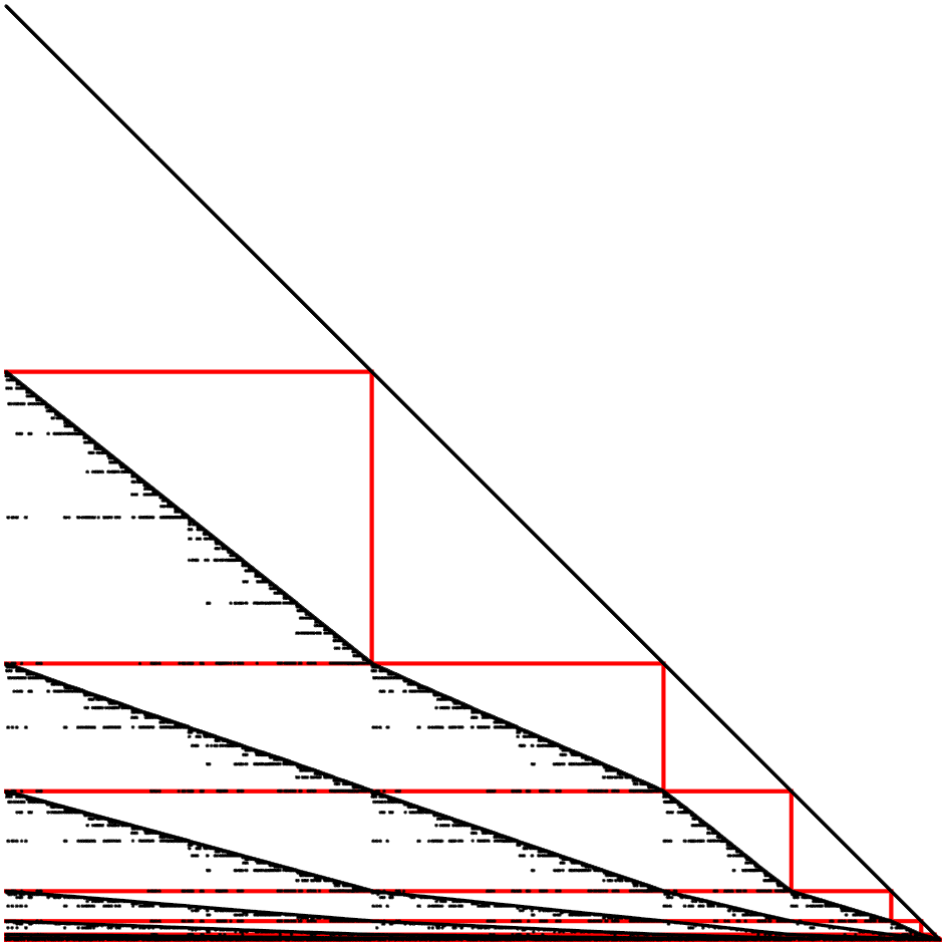
Algorithms: Level Scheduling

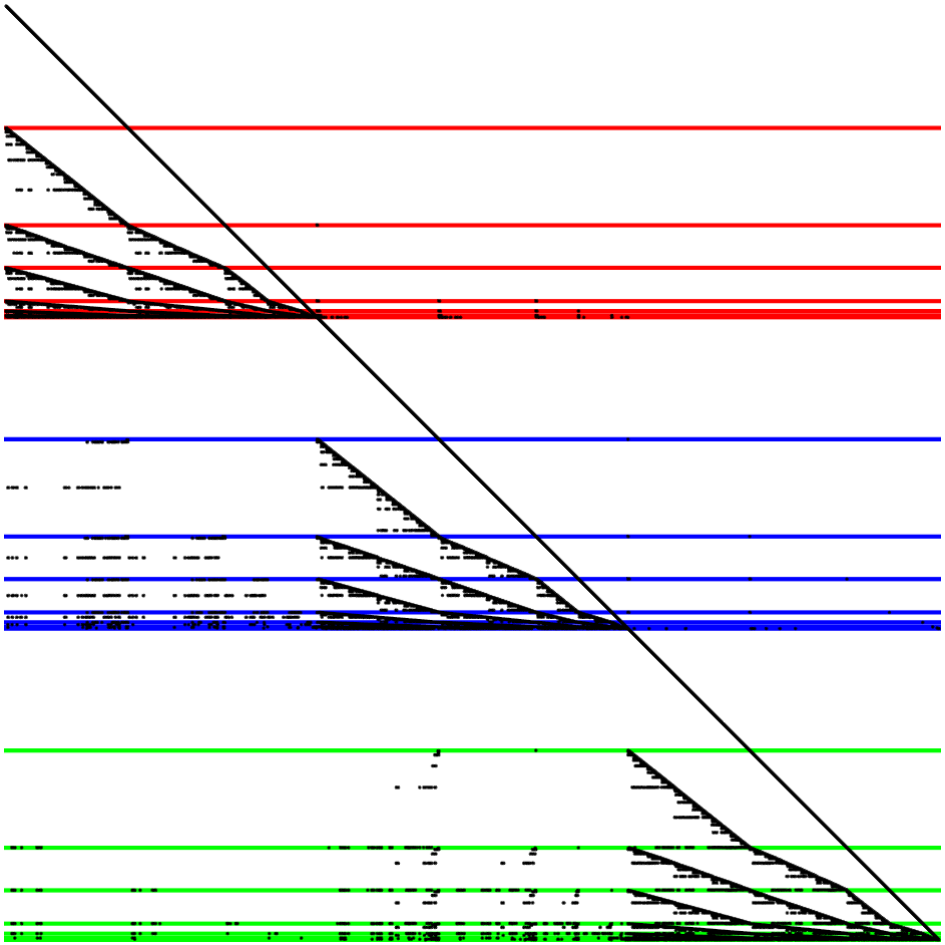




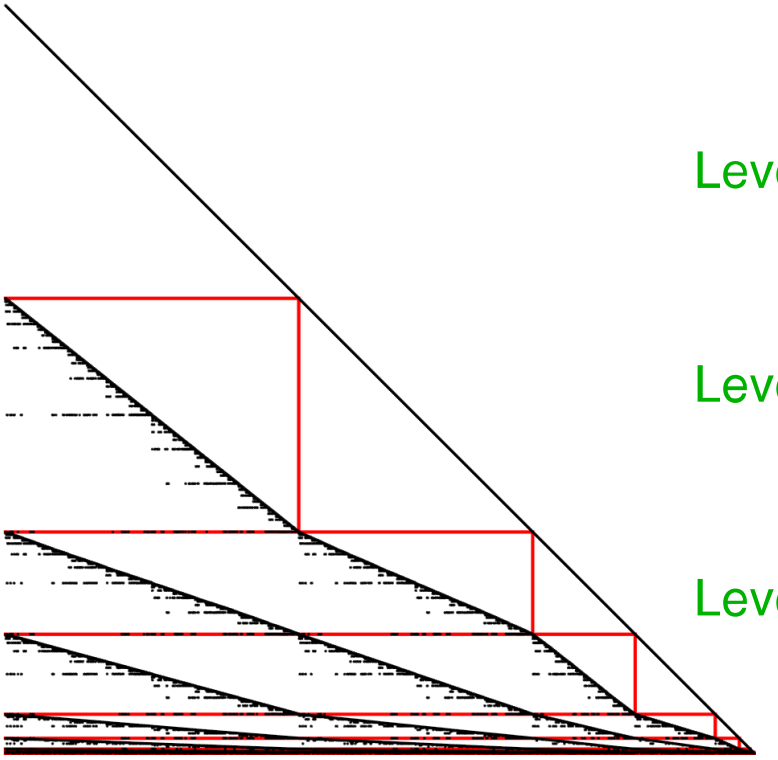
Algorithms: Level Scheduling

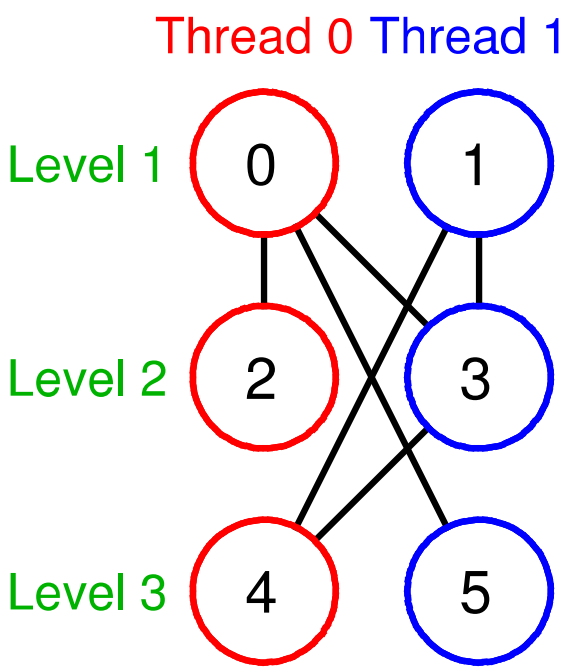
Sandia National Laboratories

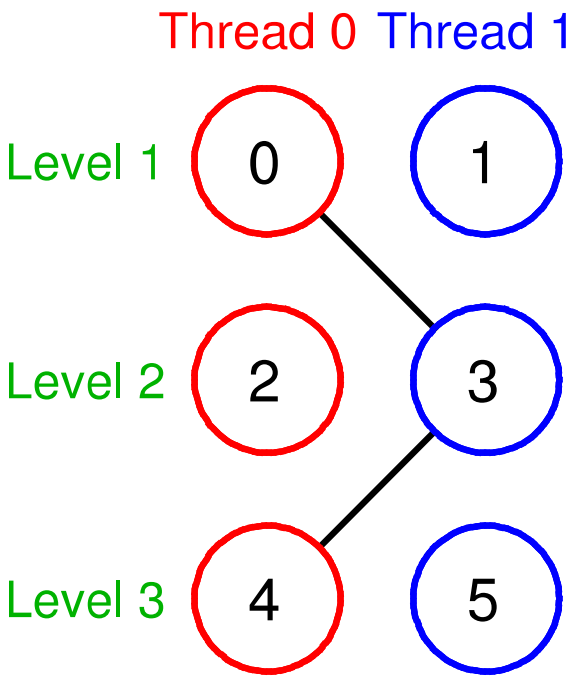




Algorithms: Pruned Point-to-Point

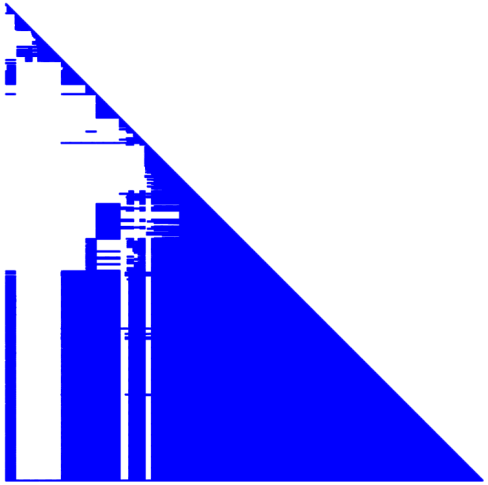


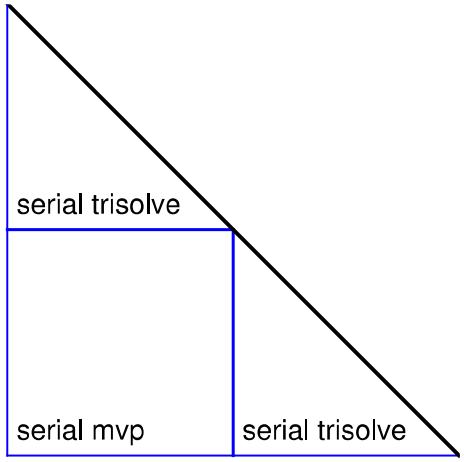


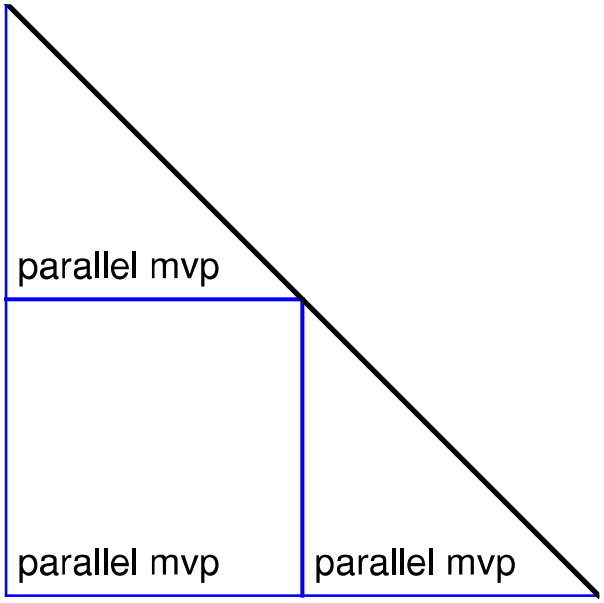


Park, J., M. Smelyanskiy, N. Sundaram, and P. Dubey., "Sparsifying synchronization for high-performance shared-memory sparse triangular solver." In *Supercomputing*, *pp.* 124-140. Springer International Publishing, 2014.

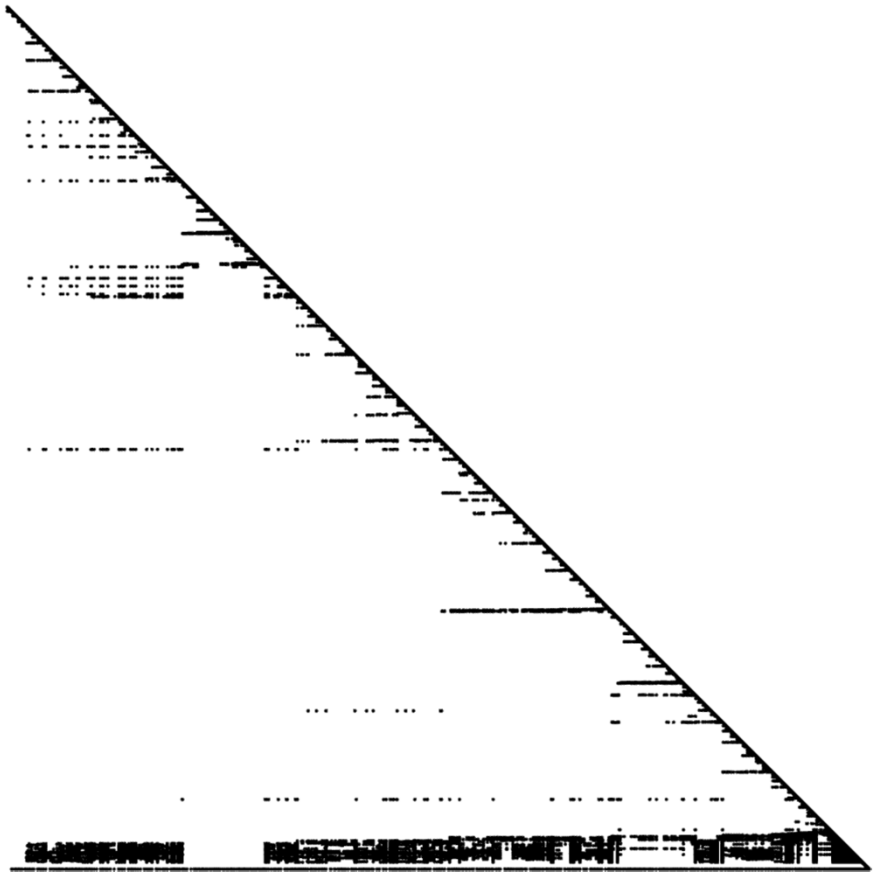
Algorithms: Recursive Blocking

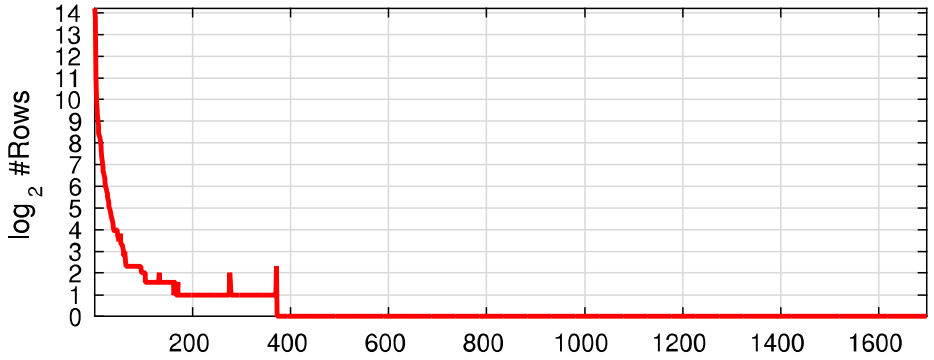


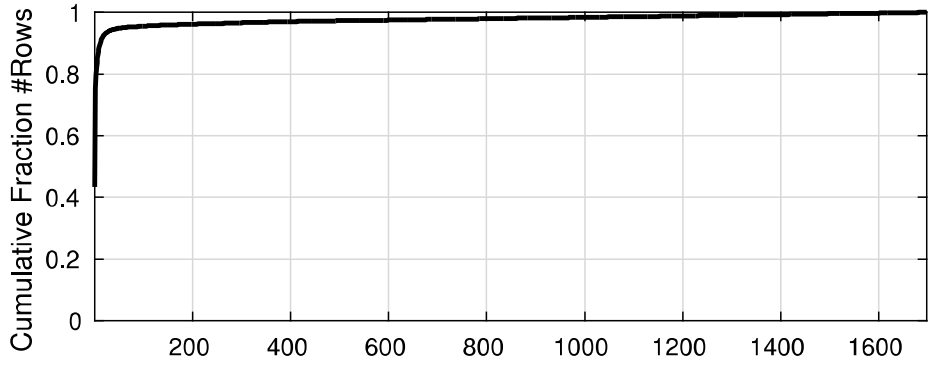


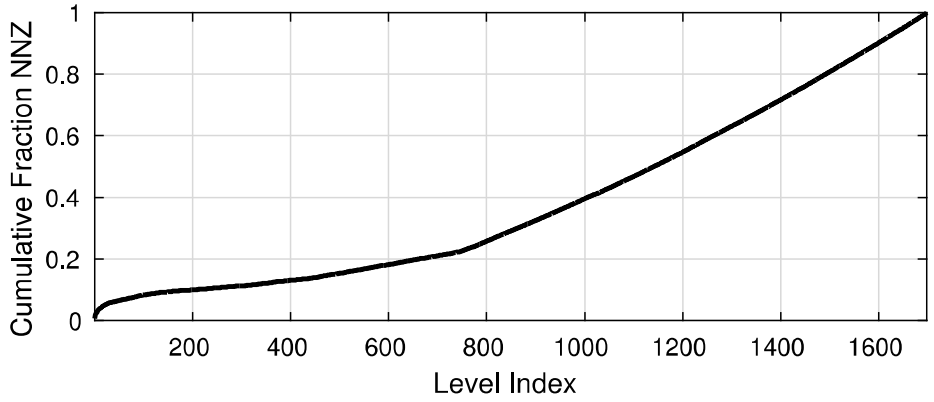


Algorithms: Hybrid

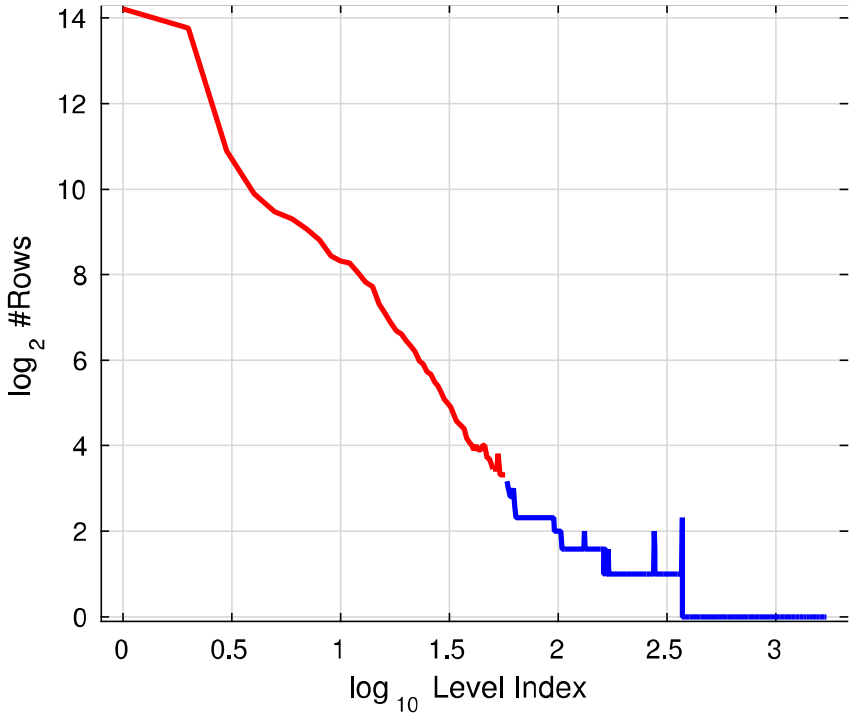


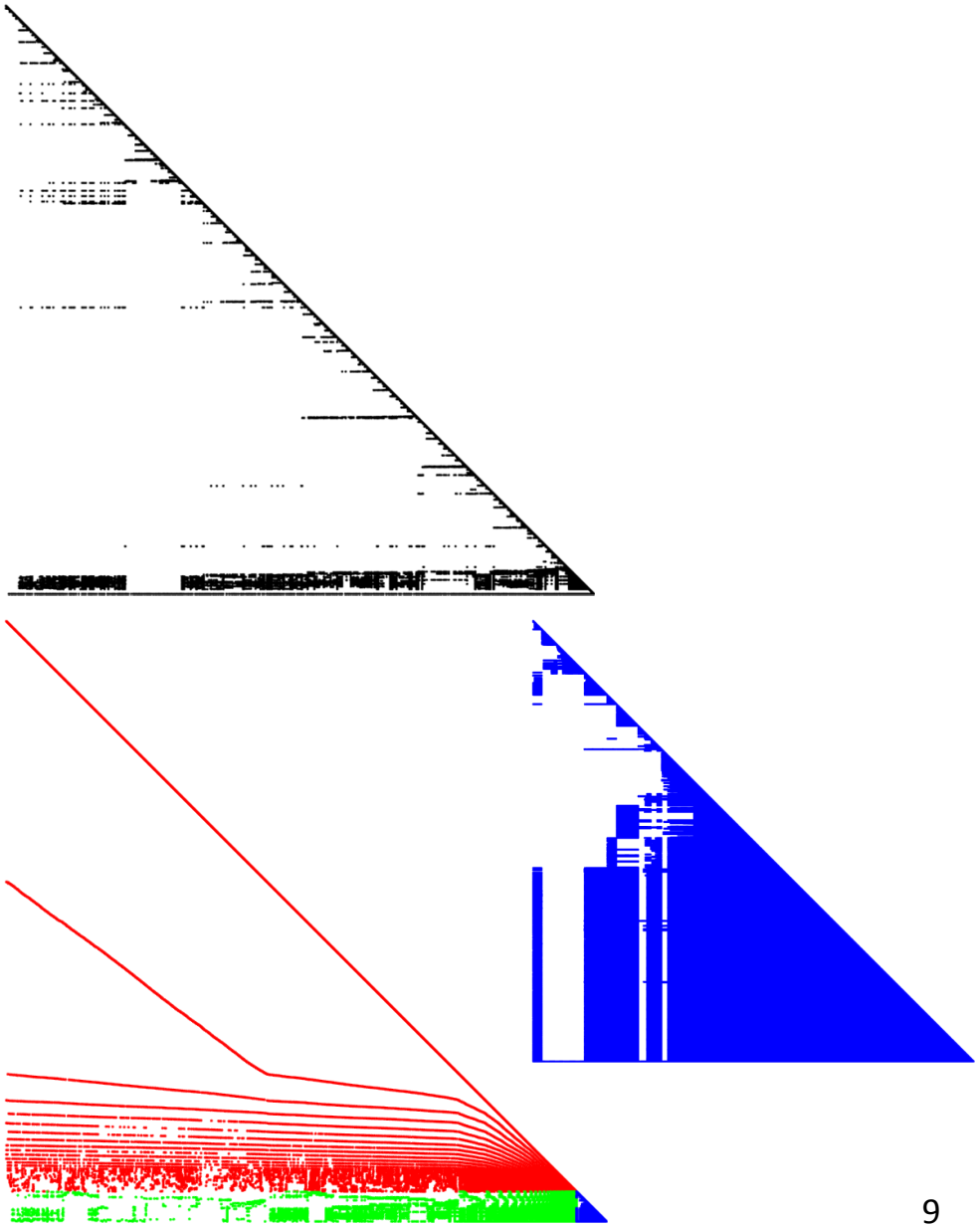




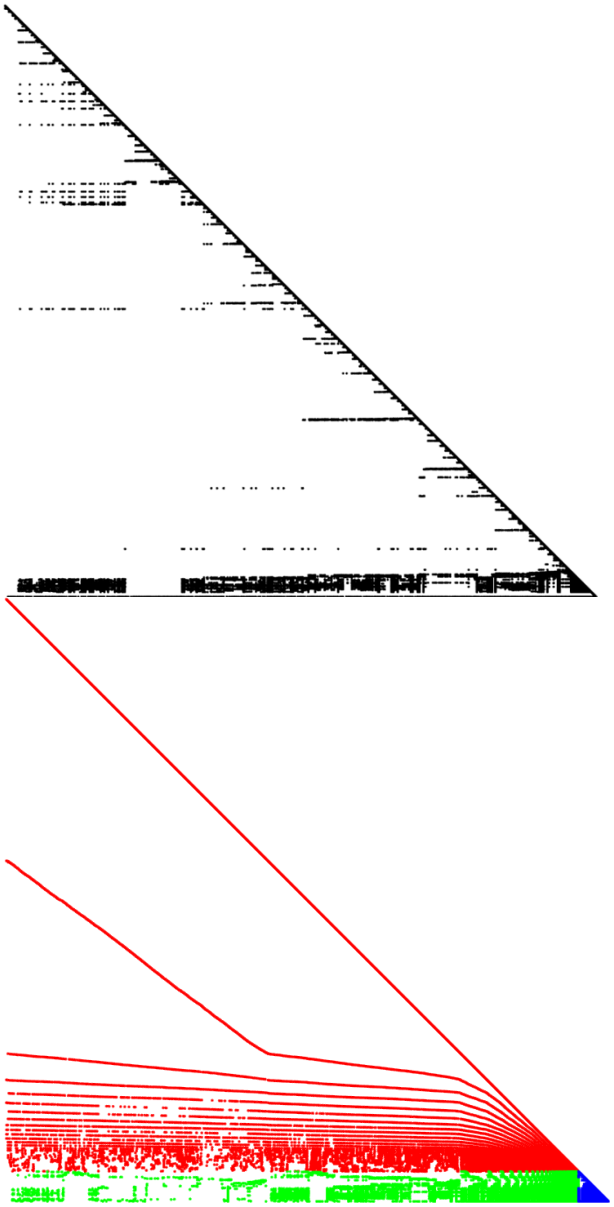


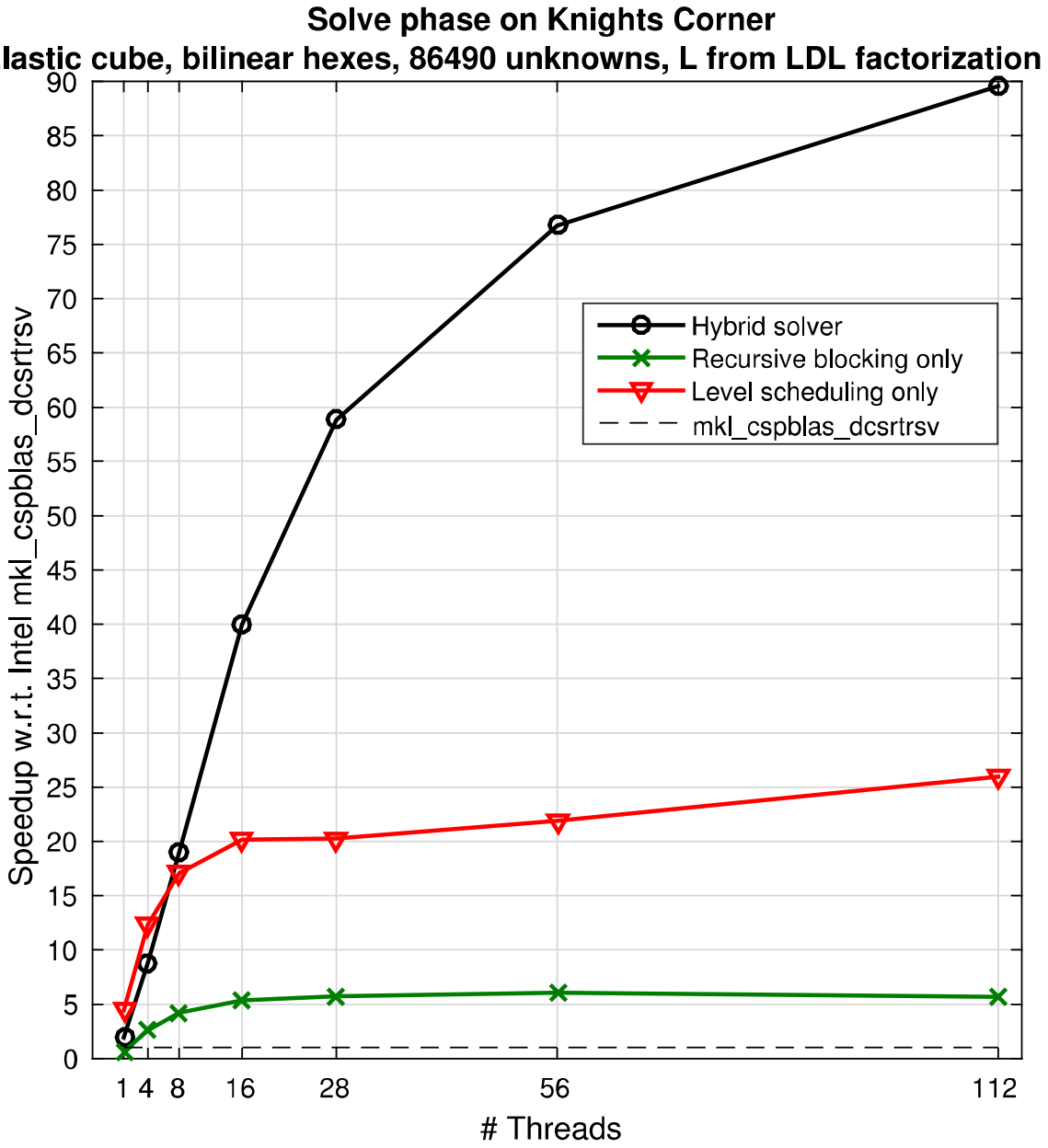
Algorithms: Hybrid





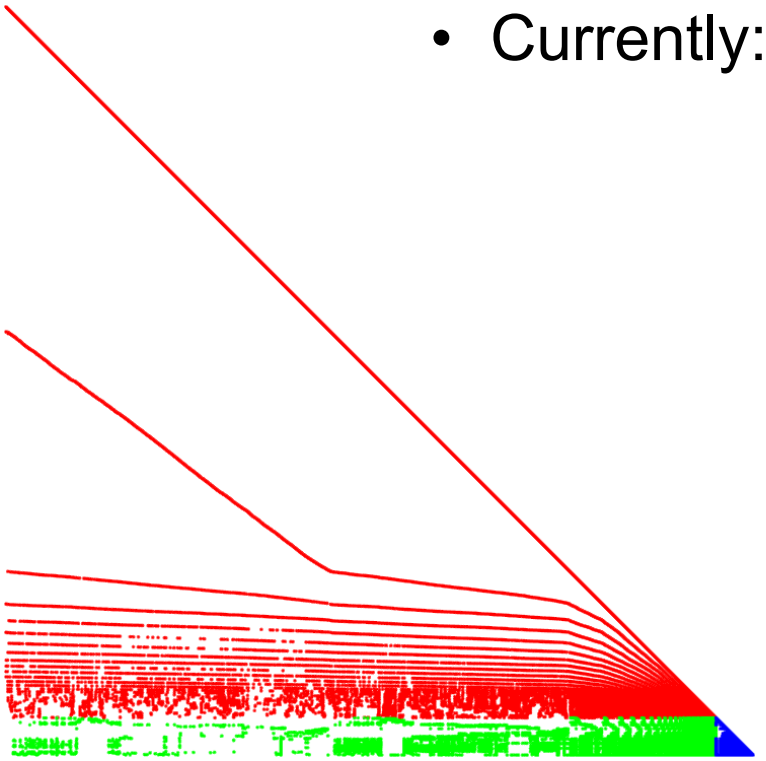
Algorithms: Hybrid

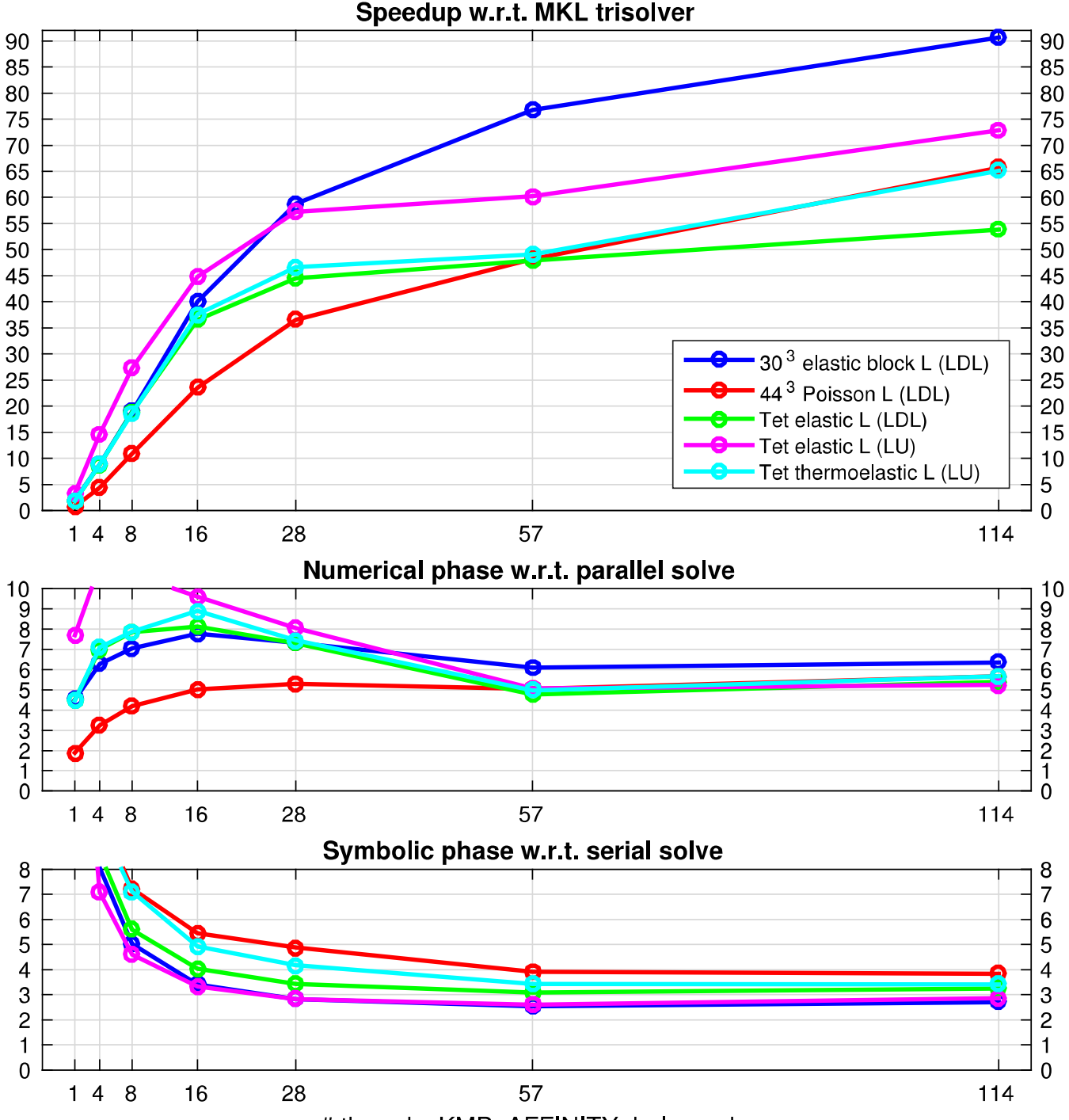




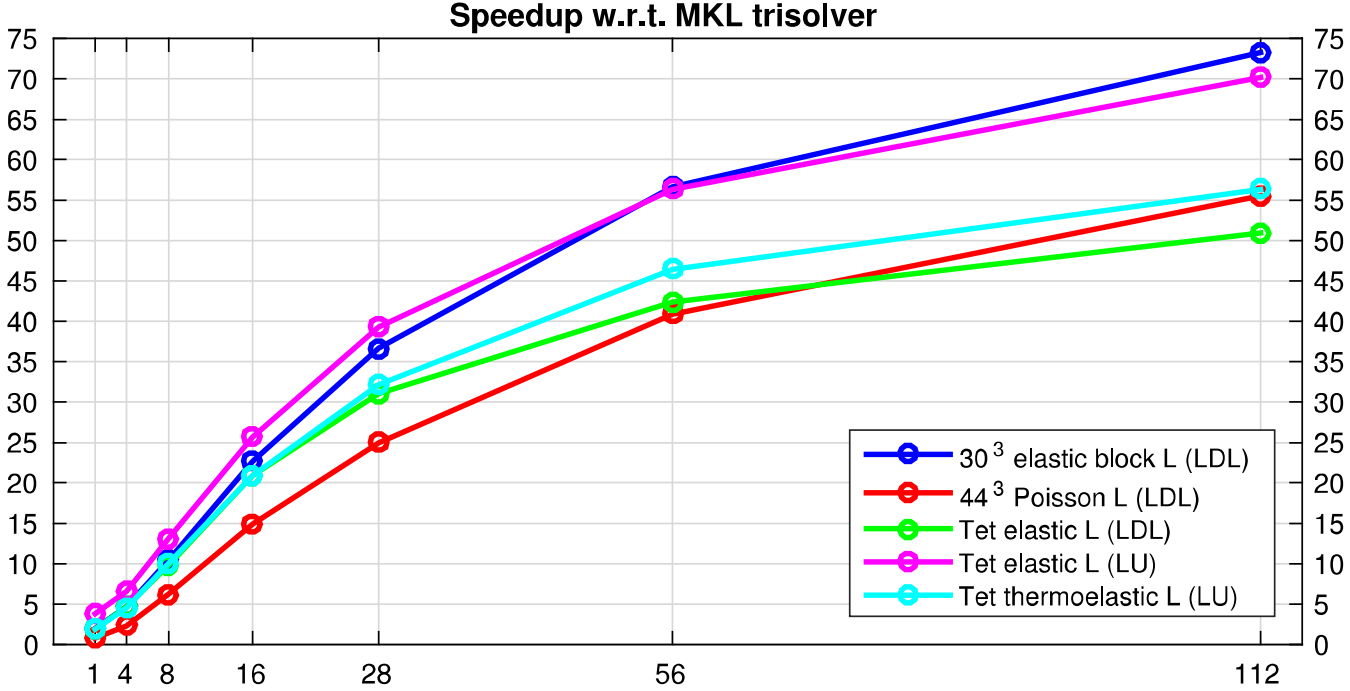
HTS

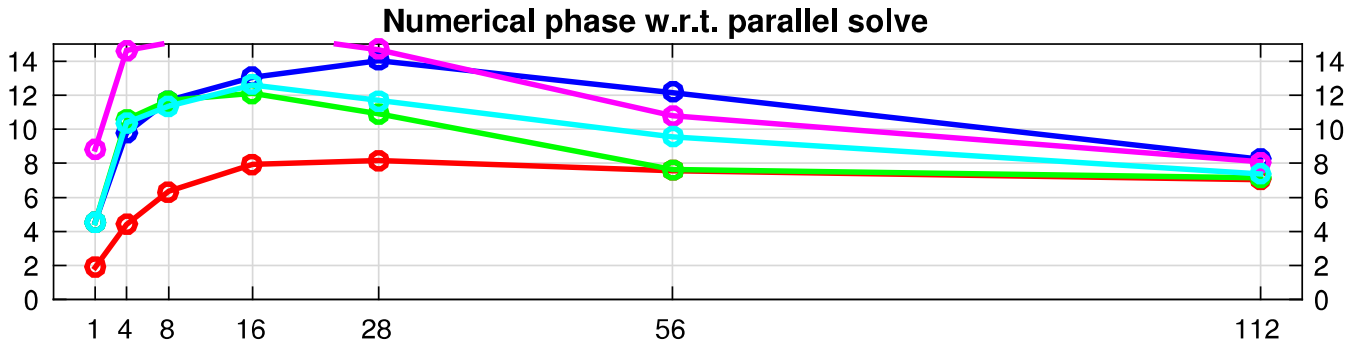
- Level schedule solve: Park et al.’s method
- Recursive blocking
  - Also with point-to-point
- Switching method
  - Currently: smoothed level size with threshold
- Currently: OpenMP and C++98

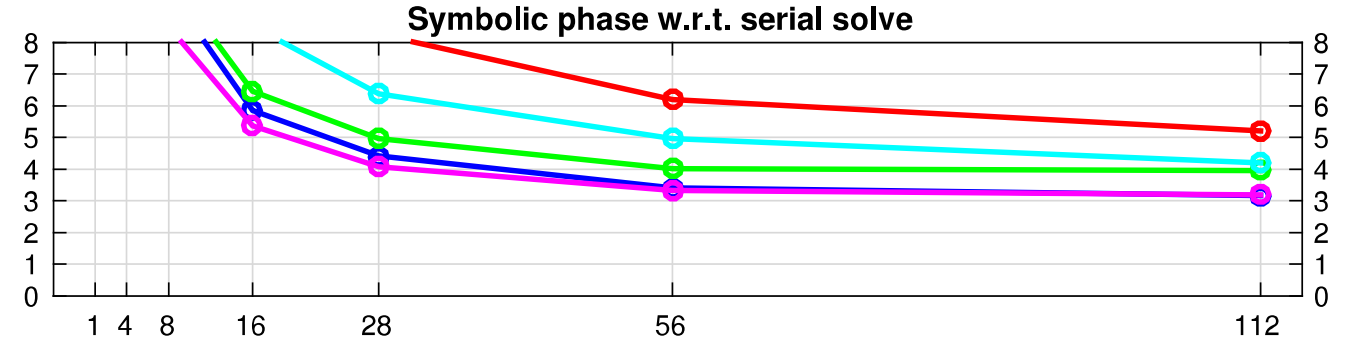




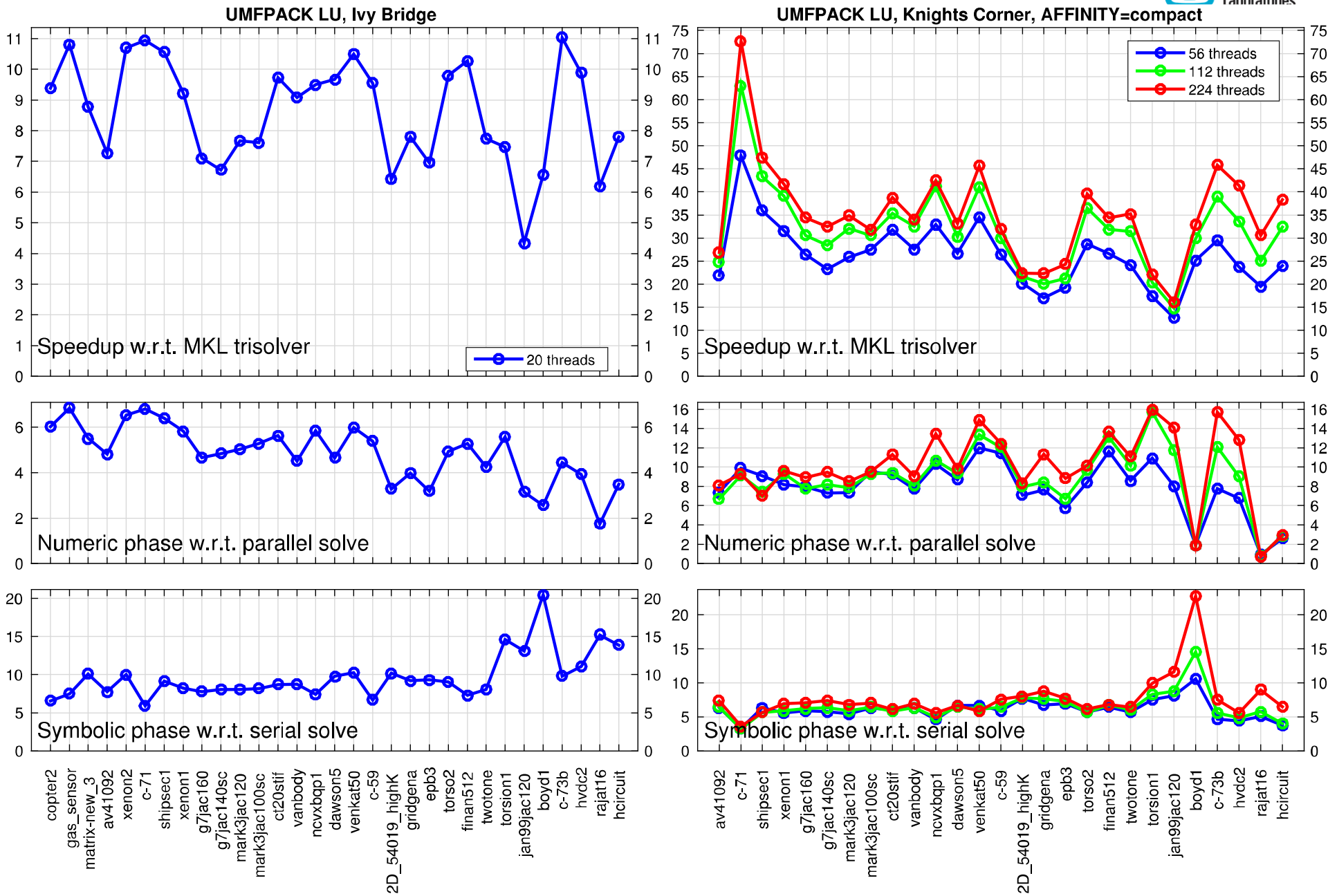
Sandia National Laboratories



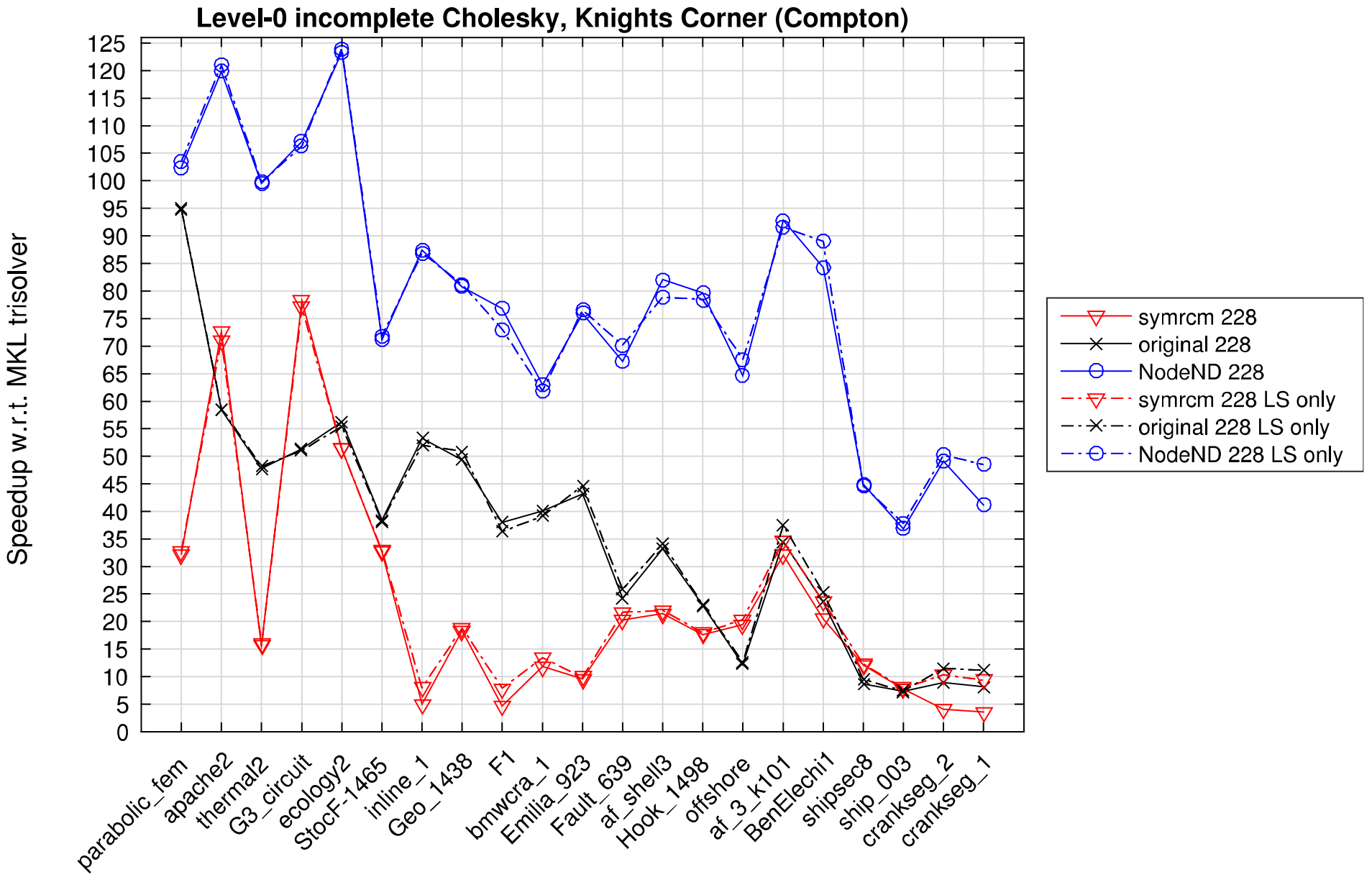


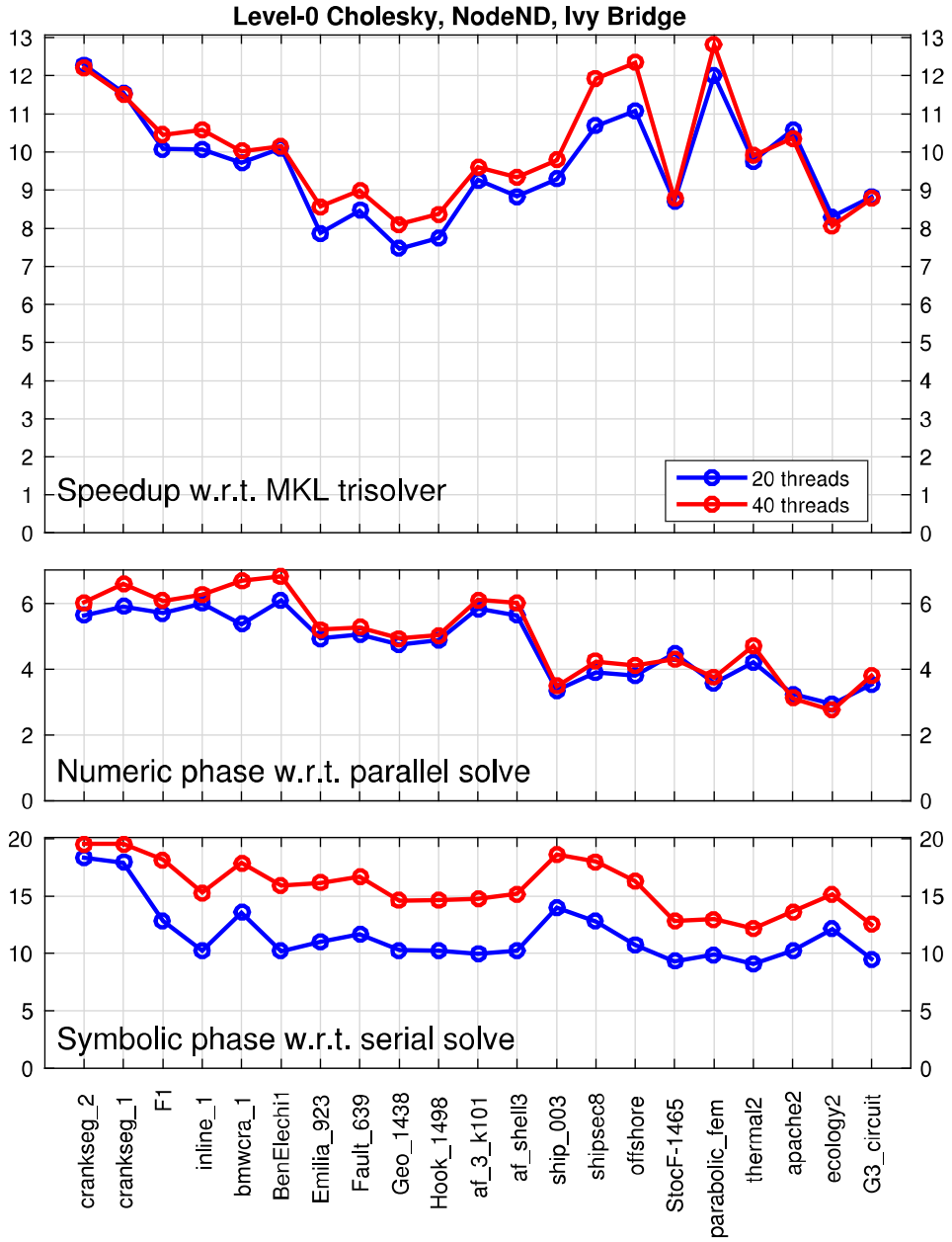


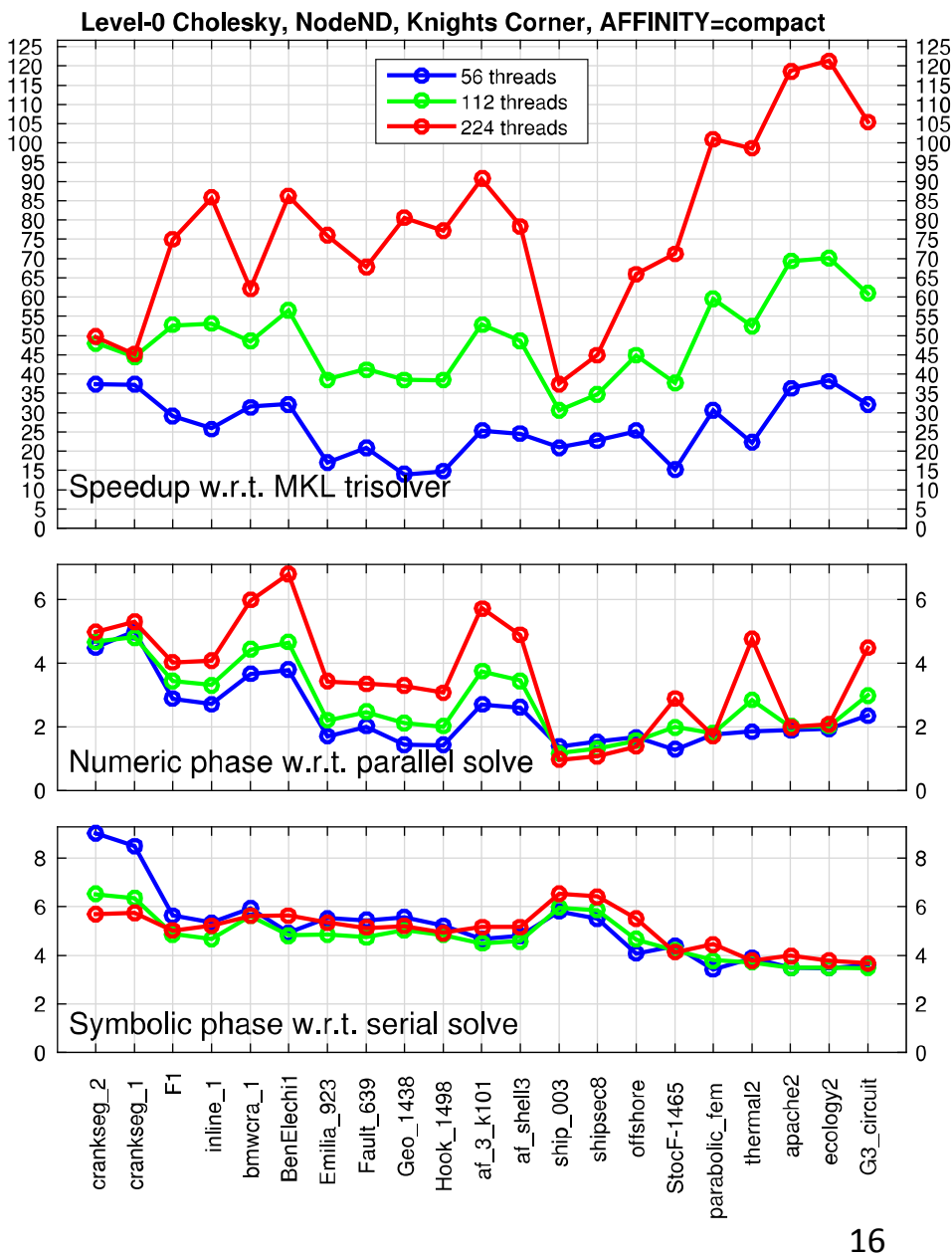
Sandia National Laboratories

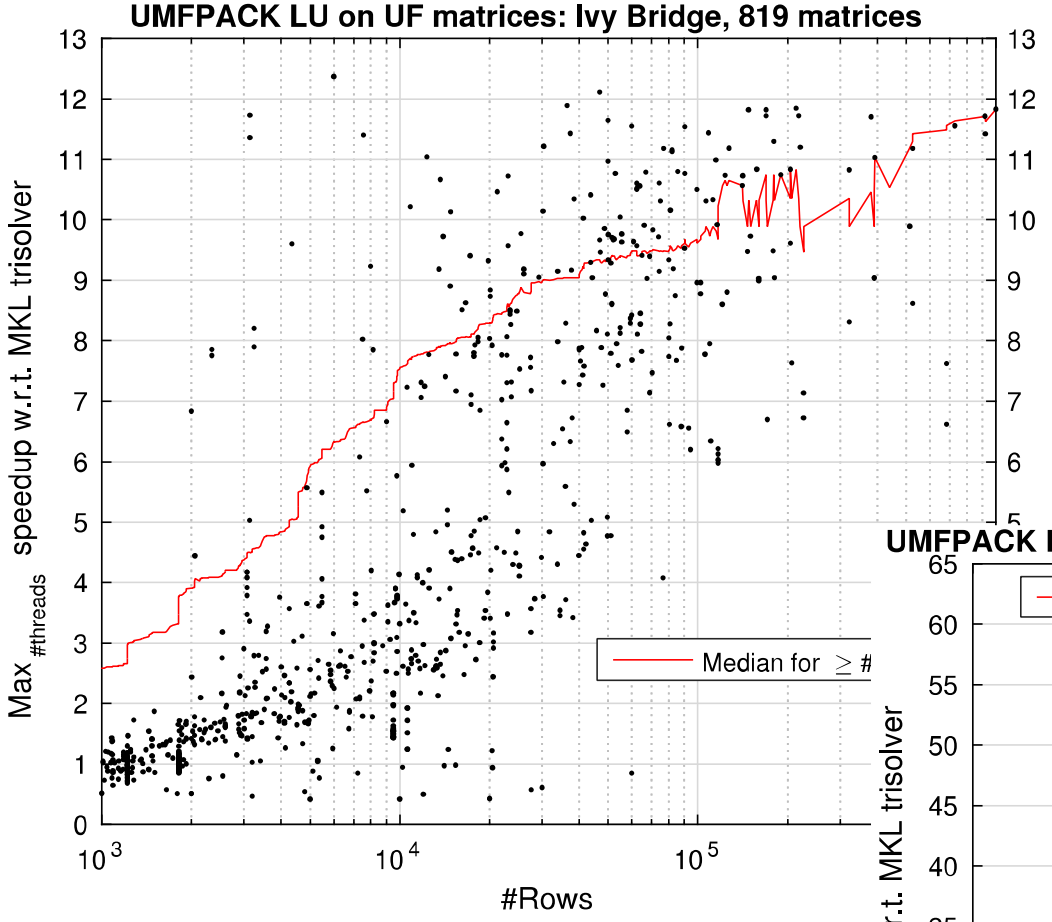

UMFPACK LU, Ivy Bridge
UMFPACK LU, Knights Corner, AFFINITY=compact
Speedup w.r.t. MKL trisolver
20 threads
56 threads
112 threads
224 threads
Numeric phase w.r.t. parallel solve
Symbolic phase w.r.t. serial solve
copter2
gas_sensor
matrix-new_3
av41092
xenon2
c-71
shipsec1
xenon1
g7jac160
g7jac140sc
mark3jac120
mark3jac100sc
ct20stif
vanbody
ncvxbqp1
dawson5
venkat50
c-59
2D_54019_highK
gridgena
epb3
torso2
finan512
twotone
torsion1
jan99jac120
boyd1
c-73b
hvdc2
rajat16
hcircuit

Ordering

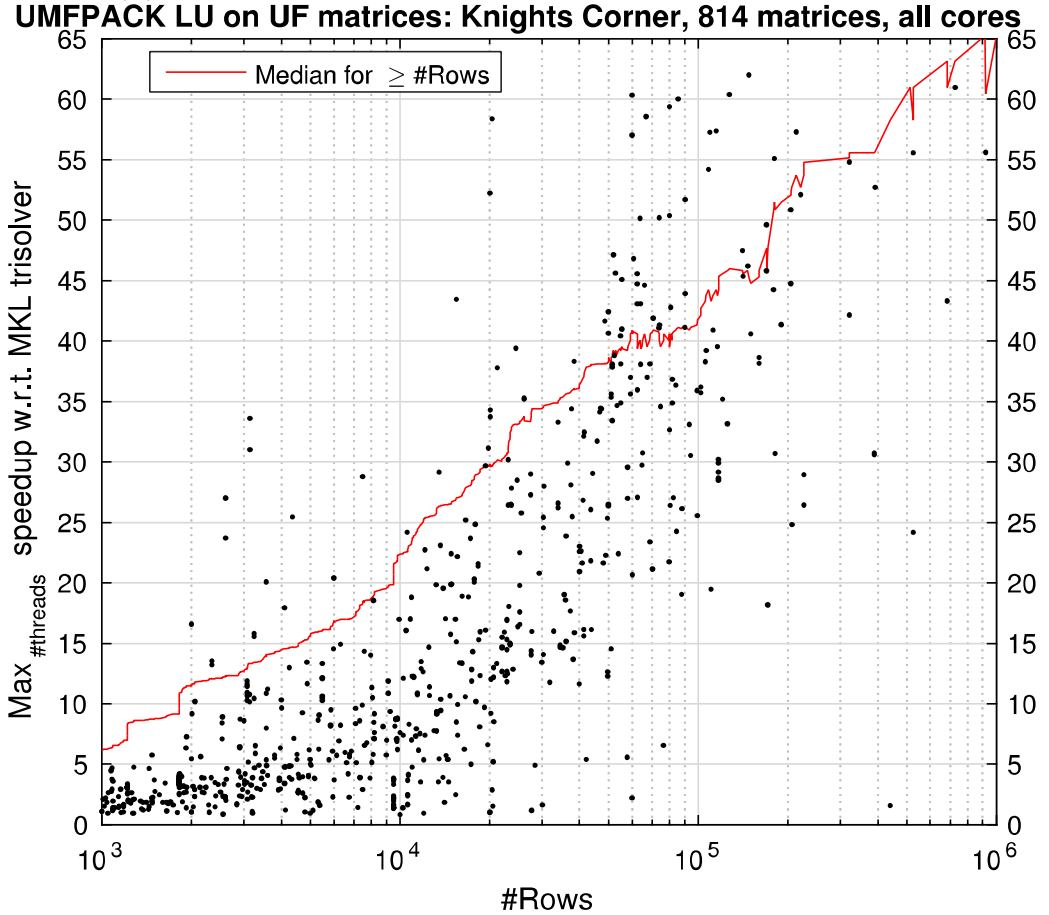

Level-0 incomplete Cholesky, Knights Corner (Compton)
Speedup w.r.t. MKL trisolver
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
parabolic_fem
apache2
thermal2
G3_circuit
ecology2
StocF-1465
inline_1
Geo_1438
F1
bmwcra_1
Emilia_923
Fault_639
af_shell3
Hook_1498
offshore
af_3_k101
BenElechi1
shipsec8
ship_003
crankseg_2
crankseg_1
symrcm 228
original 228
NodeND 228
symrcm 228 LS only
original 228 LS only
NodeND 228 LS only


Level-0 Cholesky, NodeND, Ivy Bridge
Speedup w.r.t. MKL trisolver
20 threads
40 threads
Numeric phase w.r.t. parallel solve
Symbolic phase w.r.t. serial solve
crankseg_2
crankseg_1
F1
inline_1
bmwcra_1
BenElechi1
Emilia_923
Fault_639
Geo_1438
Hook_1498
af_3_k101
af_shell3
ship_003
shipsec8
offshore
StocF-1465
parabolic_fem
thermal2
apache2
ecology2
G3_circuit


Level-0 Cholesky, NodeND, Knights Corner, AFFINITY=compact
56 threads
112 threads
224 threads
Speedup w.r.t. MKL trisolver
Numeric phase w.r.t. parallel solve
Symbolic phase w.r.t. serial solve
crankseg_2
crankseg_1
F1
inline_1
bmwcra_1
BenElechi1
Emilia_923
Fault_639
Geo_1438
Hook_1498
af_3_k101
af_shell3
ship_003
shipsec8
offshore
StocF-1465
parabolic_fem
thermal2
apache2
ecology2
G3_circuit



Sandia National Laboratories



Future Work

- Point-to-point level scheduling
  - Group rows into tasks to minimize #dependencies
  - Size tasks to reflect level of synchronization
- Hybrid
  - Switching method(s)
  - Does not have to be 3 blocks; could alternate
- HTS
  - Kokkos interface
  - Support a variety of input formats
  - Pure Kokkos implementation?
- Direct sparse methods on GPU?