

Evaluation of sparse LU factorization and triangular solution on multicore architectures

X. Sherry Li

Lawrence Berkeley National Laboratory

ParLab, April 29, 2008

Acknowledgement: John Shalf, LBNL

Rich Vuduc , Georgia Tech

Sam Williams, UC Berkeley

Overview

- Chip multiprocessor (CMP) systems become de factor HPC building blocks
 - **better trade-offs between performance (parallelism) and energy efficiency**
 - **diverse CMP architectural designs: multicore, multithreading, ...**
- Testing machines in this study: all programmable in shared address space
 - **Intel Clovertown (homogeneous multicore)**
 - **Sun VictoriaFalls (hardware-threaded multicore, NUMA)**
 - **IBM Power 5 (conventional SMP node)**
- Questions
 - **programmability: Pthread, MPI**
 - **performance of existing code**
 - **where and how to improve performance**
- Findings may be applicable to other algorithms, such as ILU

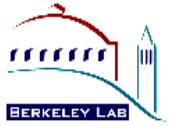
Architectural summary



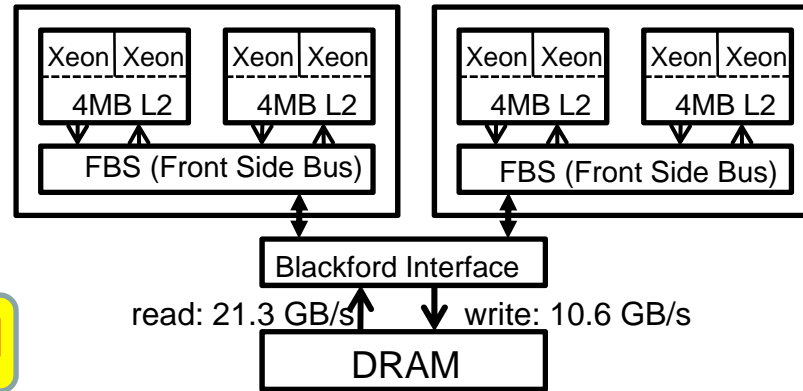
System	Intel Clovertown	Sun VictoriaFalls	IBM Power 5 (575)
Core type	superscalar (4)	multithreaded (8)	superscalar (4)
Clock (GHz)	2.3	1.4	1.9
L1 DCache	32 KB	8 KB	32 KB
# sockets	2	4	8
# cores/socket	4	8 (256 threads)	1
L2 cache	4 MB/2-cores (16 MB)	4 MB/socket (16 MB)	1.92 MB/core (32 MB L3\$/node)
DP Gflops	74.7	44.8	60.8
DRAM GB/s (read)	21.3	84	200
Byte-to-flop ratio	0.29	1.88	3.29
Socket power (Watts)	160 (max)	84 (max)	500 (measured)

□ Sources: John Shalf, Sam Williams

Architectural diagram

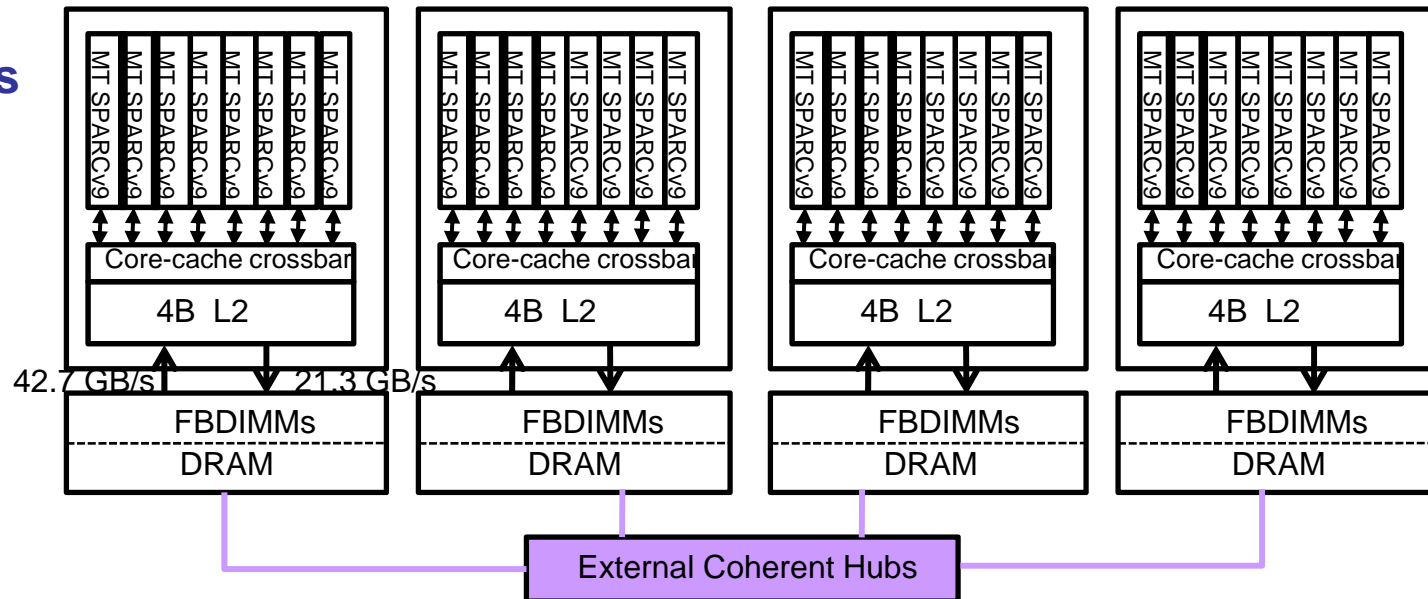


- Intel Colvertown
 - 2 sockets, 8 cores



Write bandwidth is half of Read

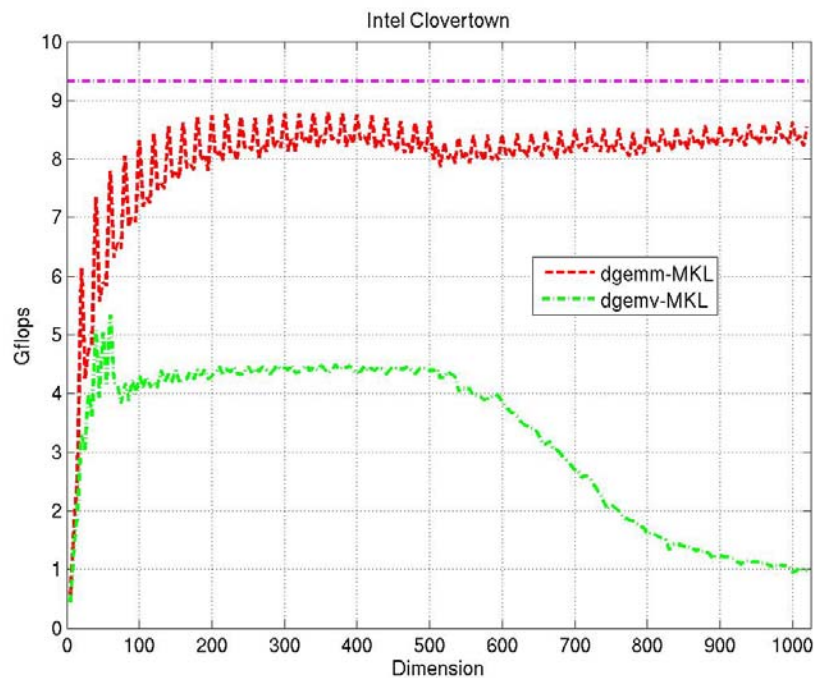
- Sun VictoriaFalls: Quad-chip Niagara2 (NUMA)
 - 32 cores
 - 256 threads



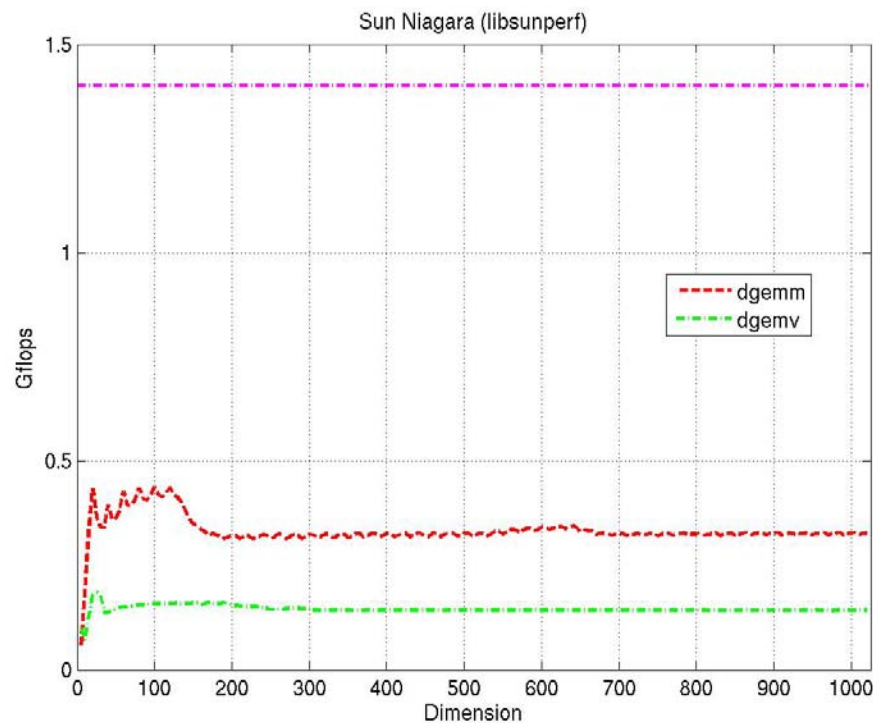
Single-core, single threaded BLAS



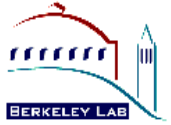
- Clovertown
 - Intel MKL



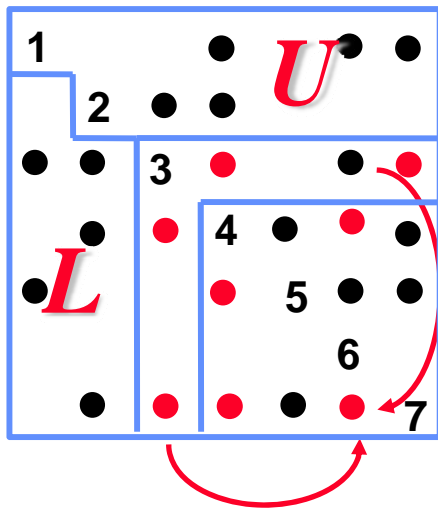
- VictoriaFalls
 - Sun Performance Library
 - Can't use 8 hw threads !!



Sparse GE (LU factorization)



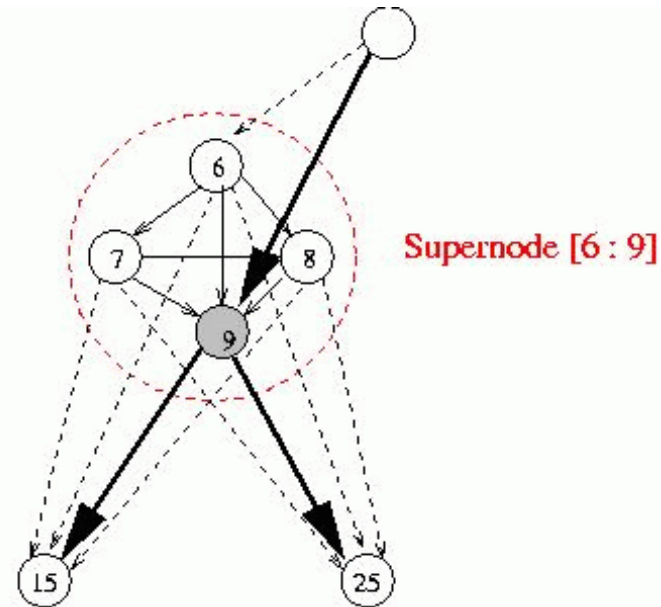
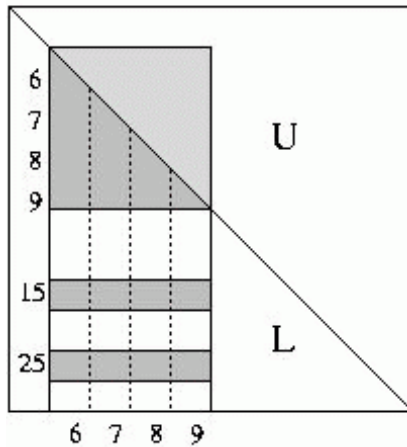
Scalar version : 3 nested loop



```
for i = 1 to n
  for j = i+1 to n
    A(j,i) = A(j,i) / A(i,i)
  for k = i+1 to n s.t. A(i,k) != 0
    for j = i+1 to n s.t. A(j,i) != 0
      A(j,k) = A(j,k) - A(j,i) * A(i,k)
```

Typical fill-ratio: 10x for 2D problems, 30-50x for 3D problems

Supernode: dense blocks in $\{L \setminus U\}$



- Good for high performance
 - Enable use of Level 3 BLAS
 - Reduce inefficient indirect addressing (scatter/gather)
 - Reduce time of the graph algorithms by traversing a coarser graph

Major stages

1. Order equations & variables to preserve sparsity.
 - NP-hard, use heuristics
2. Symbolic factorization.
 - Identify supernodes, set up data structures and allocate memory for L & U.
3. Numerical factorization – usually dominates total time.
 - How to pivot?
4. Triangular solutions – usually less than 5% total time.

SuperLU_MT

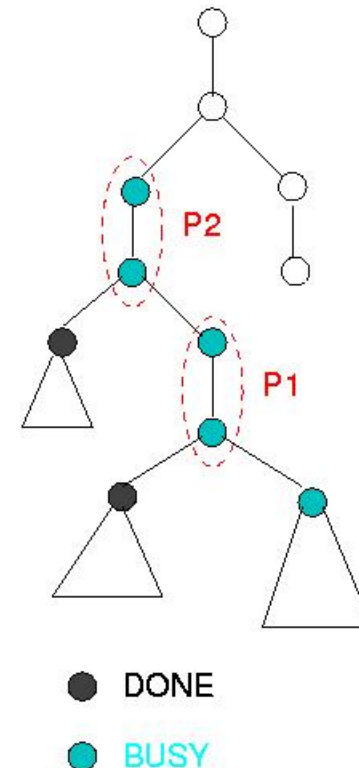
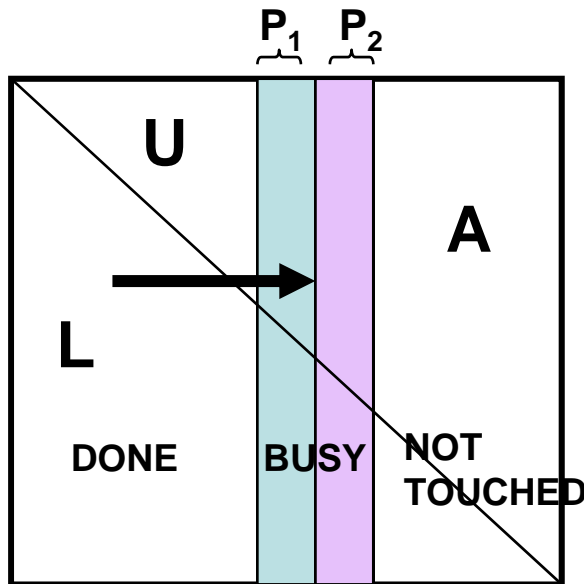
1. Sparsity ordering
2. Factorization
 - Partial pivoting
 - Symbolic fact.
 - Num. fact. (BLAS 2.5)
3. Solve

SuperLU_DIST

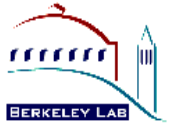
1. Static pivoting
2. Sparsity ordering
3. Symbolic fact.
4. Numerical fact. (BLAS 3)
5. Solve

SuperLU_MT [Li/Demmel/Gilbert]

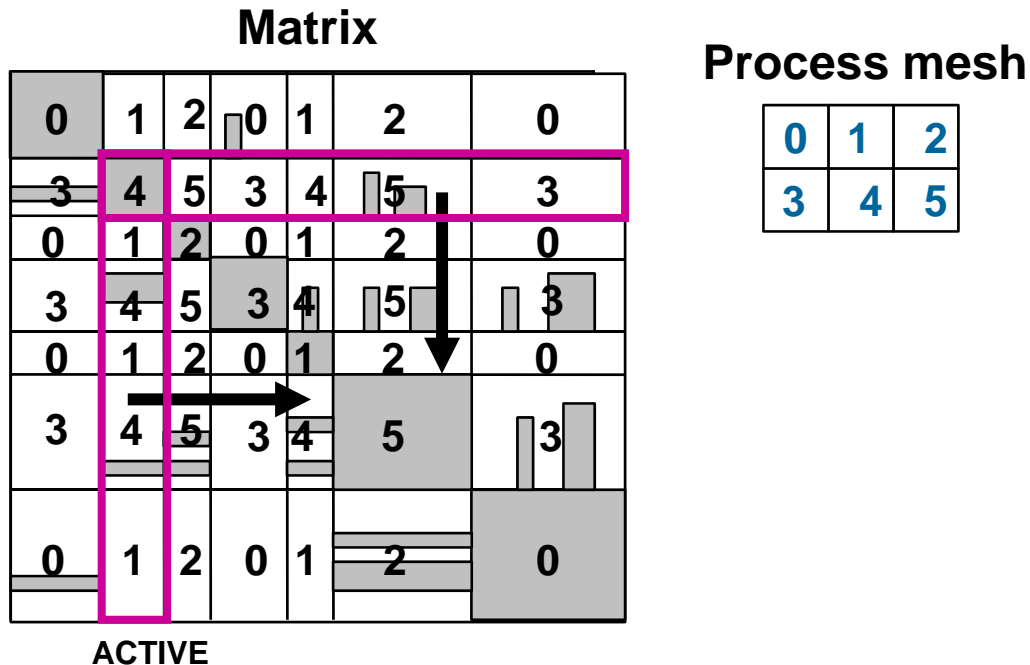
- Pthread or OpenMP
- **Left looking** – relatively more READs than WRITEs
- Use shared task queue to schedule ready columns in the elimination tree (bottom up)
- Over 12x speedup on conventional 16-CPU SMPs (1999)



SuperLU_DIST [Li/Demmel/Grigori]

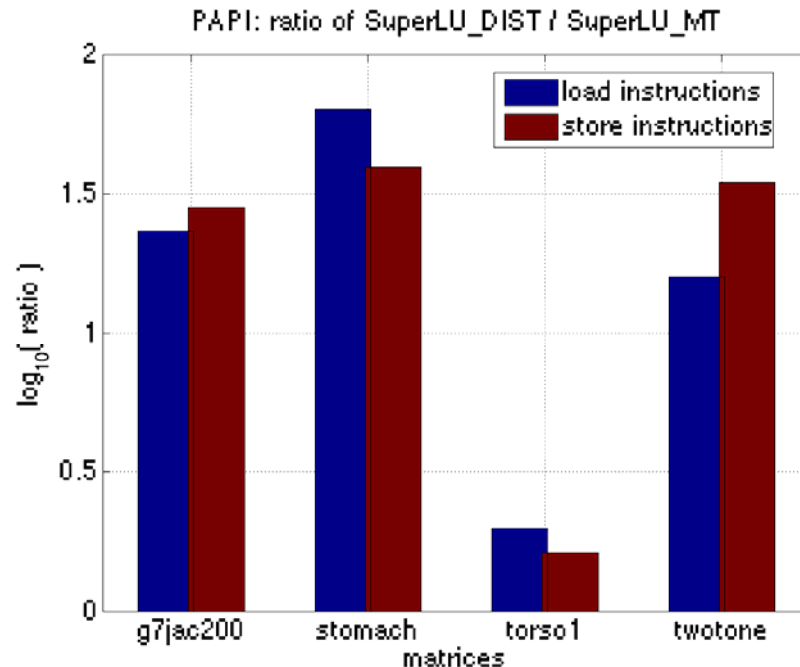


- MPI
- **Right looking** -- relatively more WRITES than READS
- 2D block cyclic layout
- One step look-ahead to overlap comm. & comp.
- Scales to 1000s processors



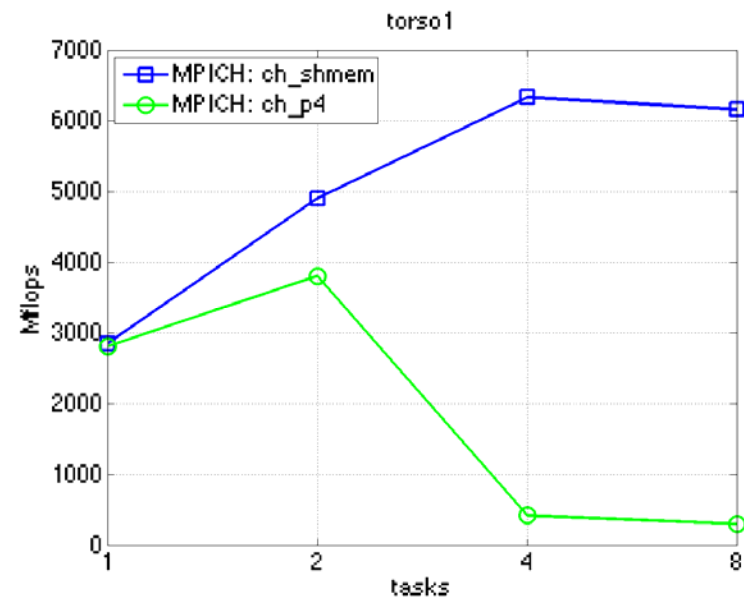
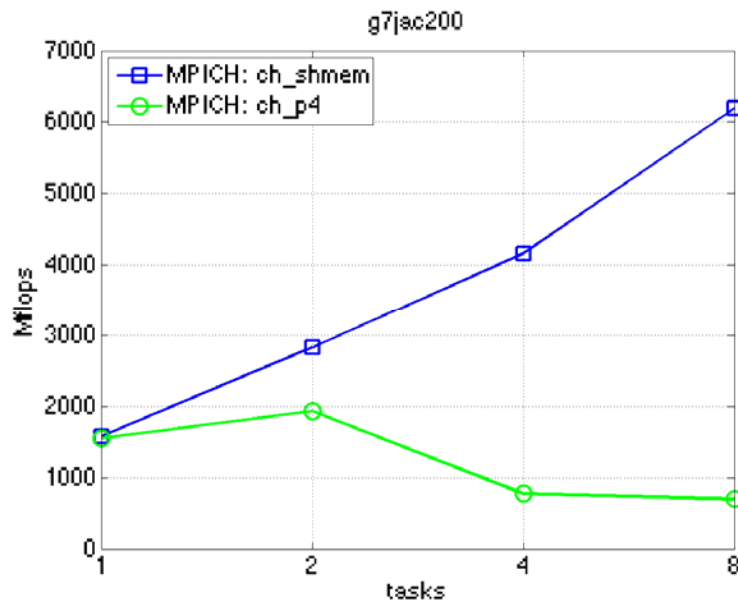
PAPI : load/store counters

- PAPI: Performance Application Programming Interface
 - **Portable interface to access hardware performance counters**
- right-looking (slu_dist) has over 30x more load or store instructions
- STORE is costly: cache coherence, lower bandwidth

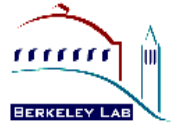


Clovertown – SuperLU_DIST

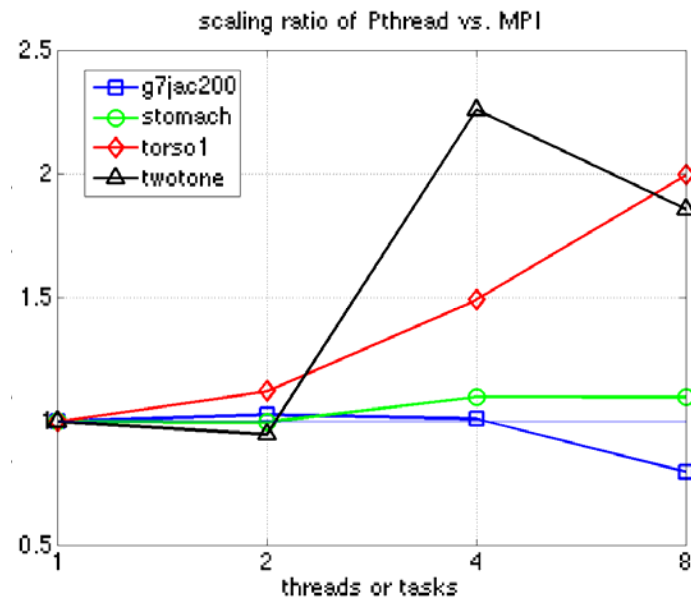
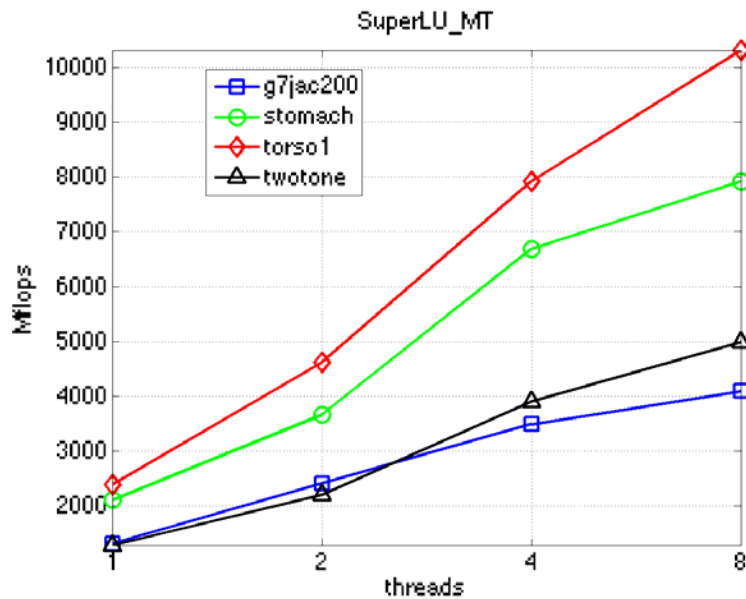
- MPICH can be configured one of two modes:
 - “ch_shmem” within socket
 - “ch_p4” across sockets
- MPICH needs hybrid mode (not yet available !!)



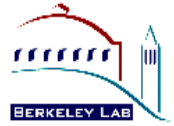
Clovertown – SuperLU_MT



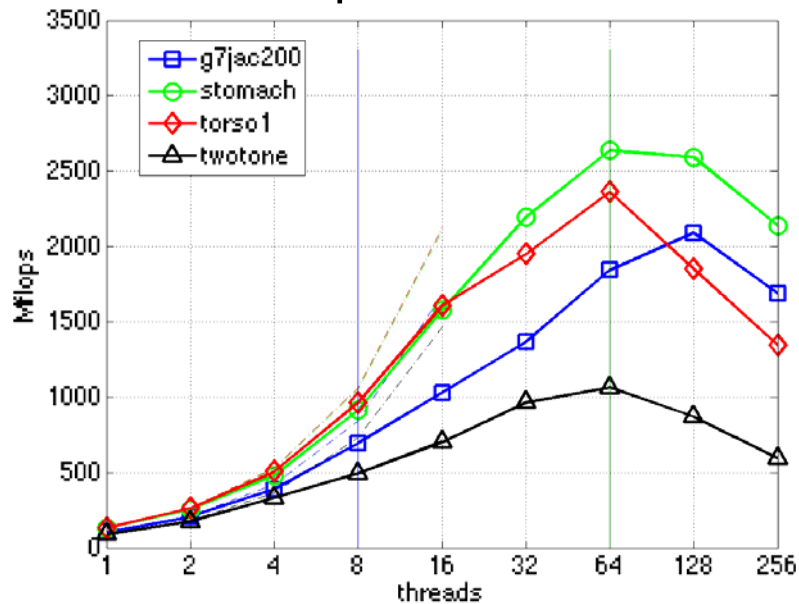
- Maximum speedup 4.3, smaller than conventional SMP
- Pthreads scale better



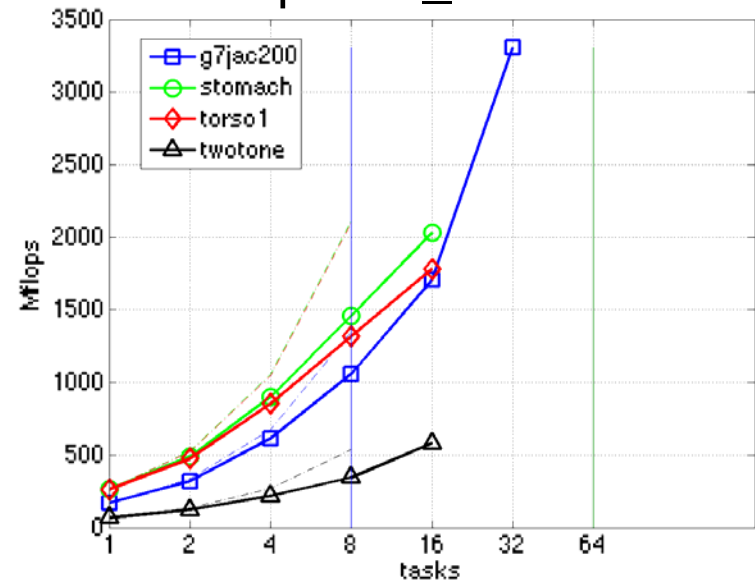
VictoriaFalls – multicore + multithread



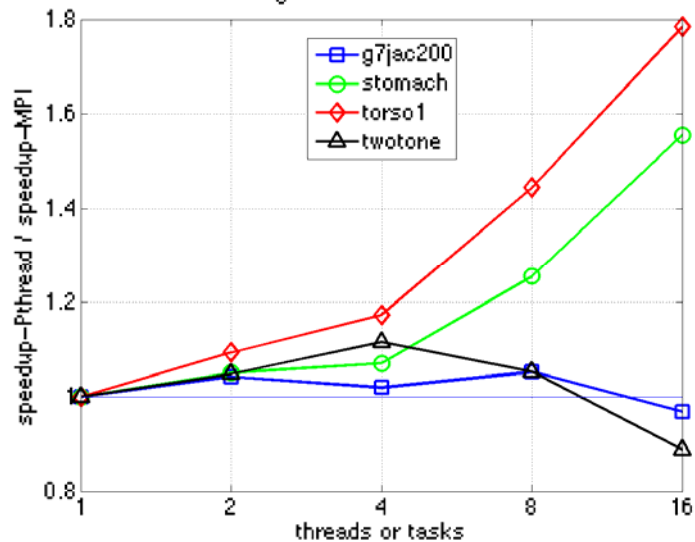
SuperLU_MT



SuperLU_DIST

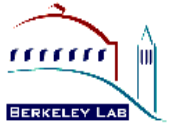


scaling ratio of Pthread vs. MPI

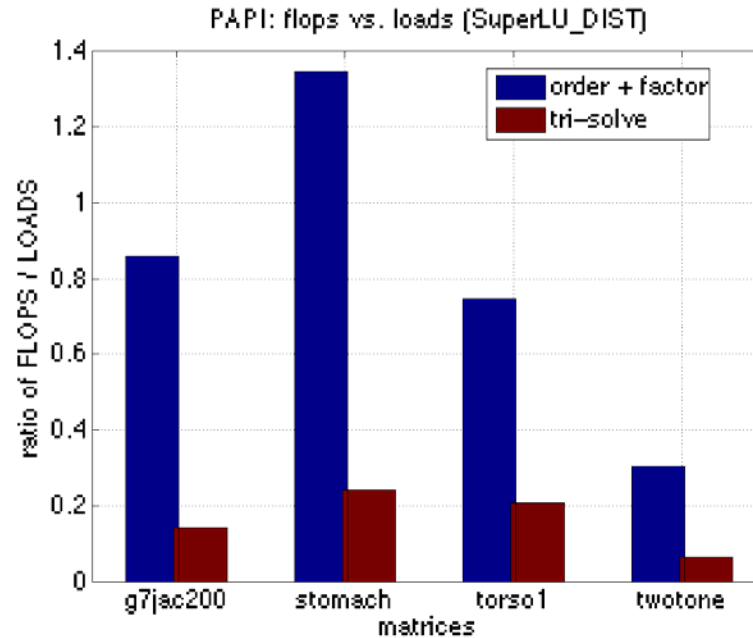


- Pthreads more robust, scale better
- MPICH crashes with large tasks
 - mismatch between coarse and fine grain models

Triangular solution



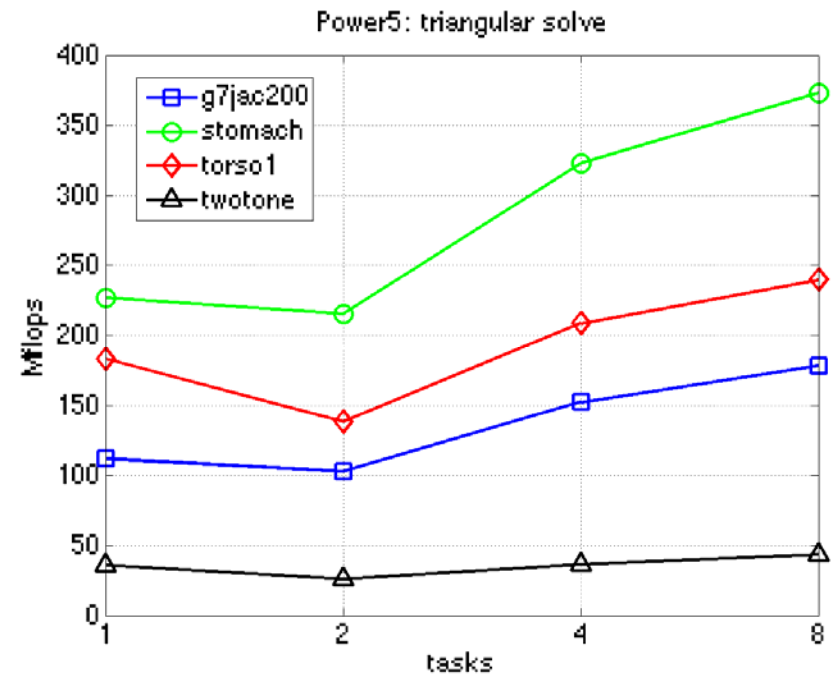
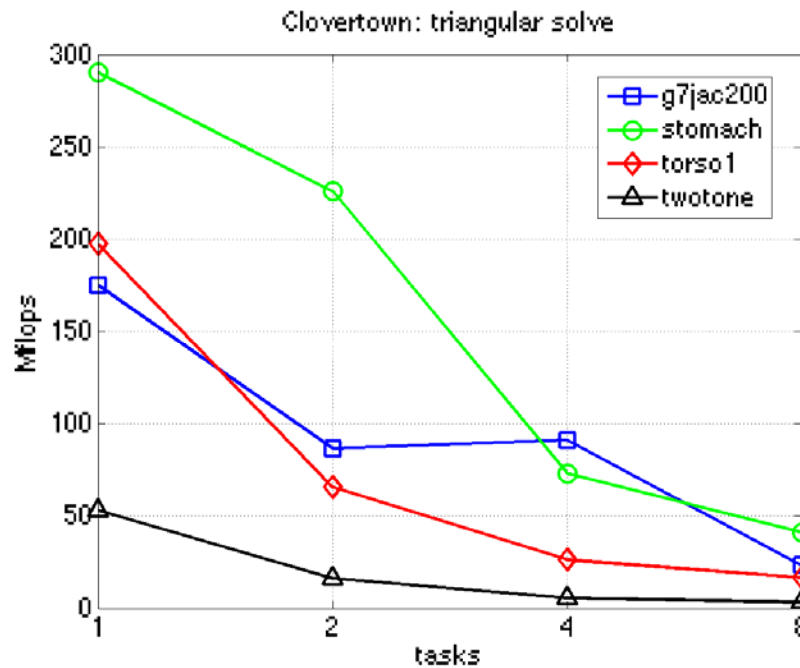
- PAPI counters of flops versus load instructions



Flops-to-load ratio

Triangular solution

- Clovertown: 8 cores
- IBM Power5: 8 cpus/node

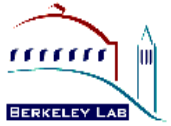


➤ Time of MPI_Reduce with 1, 2, 4, 8 tasks: 0.09, 0.50, 1.28, 2.52

Final remarks

- Results are preliminary, findings may be applicable to other algorithms, such as ILU preconditioner
 - **right-looking (maybe multifrontal) incurs more memory traffic**
- Hybrid algorithm, hybrid programming will be beneficial
 - **left-looking + right-looking**
 - **threading + MPI**
 - **require significant code rewriting**
- Need good runtime profiling tools to study multicore scaling
 - **how to calibrate memory and other contentions in the system?**

Test matrices



	apps	dim	nnz(A)	SLU_MT Fill	SLU_DIST Fill	Avg. S-node
g7jac200	Economic model	59,310	0.7 M	33.7 M	33.7 M	1.9
stomach	3D finite diff.	213,360	3.0 M	136.8 M	137.4 M	4.0
torso1	3D finite diff.	116,158	8.5 M	26.9 M	27.0 M	4.0
twotone	Nonlinear analog circuit	120,750	1.2 M	11.4 M	11.4 M	2.3

- SLU_MT “Symmetric Mode”
 - ordering on $A+A'$
 - pivot on main diagonal