A Solver That Learns

Tom Elken

Member of Technical Staff: Math Libraries

Silicon Graphics, Inc.

41st Cray User Group
Conference
Minneapolis, Minnesota

Overview

- *Sparse linear solvers at SGI*
- *Matrix ordering options*
- *New ordering options*
- *Performance results*
- *Other Enhancements to Sparse Solver*
- *Ideas for the Future*

- (much of this was joint work with Cheng Liao, SGI)

Sparse linear solvers at SGI

SCSL / complib

- PSLDLT: symmetric
- PSLDU: structurally symmetric

On Developer's Toolbox:

- sparse iterative solver

LIBSCI (PVP)

- iterative, symmetric, structurally symmetric and general solvers

Sparse linear solvers at SGI

sgi

- Introduced ~1995
- First offered to ISVs as static libraries to enhance applications
- Sometimes offered 5–10x advantage
- Added to complib and then SCSL
- Now, public domain versions exist & they are an industry standard
- Need to keep enhancing them

What is sparse-matrix ordering?

- In solving Mx=b, first factor M into LDL^T.
  L_{ij} entries can become non-zero when M_{ij} are zero.
- Solve by solving Lz=b; Dy=z; $L^T x=y$.
- Fill (non-zeroes) can be up to 50X in L compared to M.
- Factoring a permuted matrix PMP^T can dramatically reduce the work to factorize M. A permutation is a re-ordering.

PSLDLT/PSLDU ordering options

0. No re-ordering

1. Approximate Minimum Fill (AMF)

- fairly fast ordering, moderate quality

2. Nested Dissection Hybrid

- Ed Rothberg & Bruce Hendrickson original authors
- AKA: Extreme ordering or BEND
- more ordering time, better quality

Algorithmic ideas

- Finding the best ordering of rows is an NP-complete (very difficult) problem
- Currently, matrix ordering is a serial computation
- Very different quality of orders can be produced using different “starting points”
- ==> Do more orders using embarrassing parallelism & use the best
- Internally referred to as Extreme2 ordering

New ordering options

3. Multiple Nested Dissection orders

- default is OMP_NUM_THREADS orders
- repeatable quality

4. Multiple ND orders using feedback file information

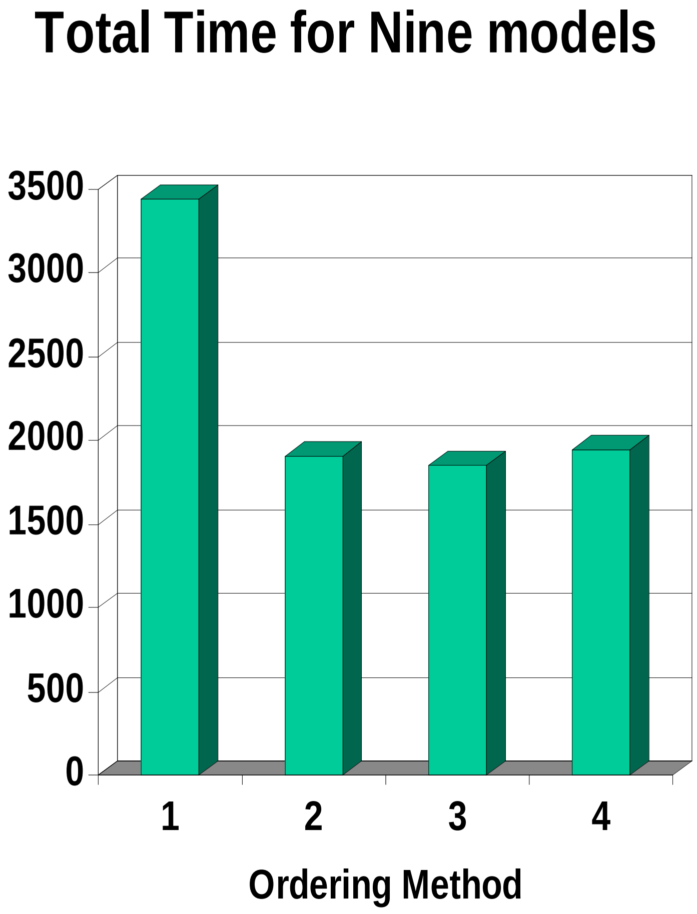
- default is 2 x OMP_NUM_THREADS orders
- file is at most 5KB, up to 200 records
- feedback file is binary
- A solver that learns

Other Changes to SGI Sparse Solvers

- New default for ordering option
  - Extreme ordering (Method 2) is now the default
- Out-of-core solver option
  - Was in recent SCSL version, but now is documented
  - Single-processor only
  - Striped file system useful
  - Simple interface and performs well

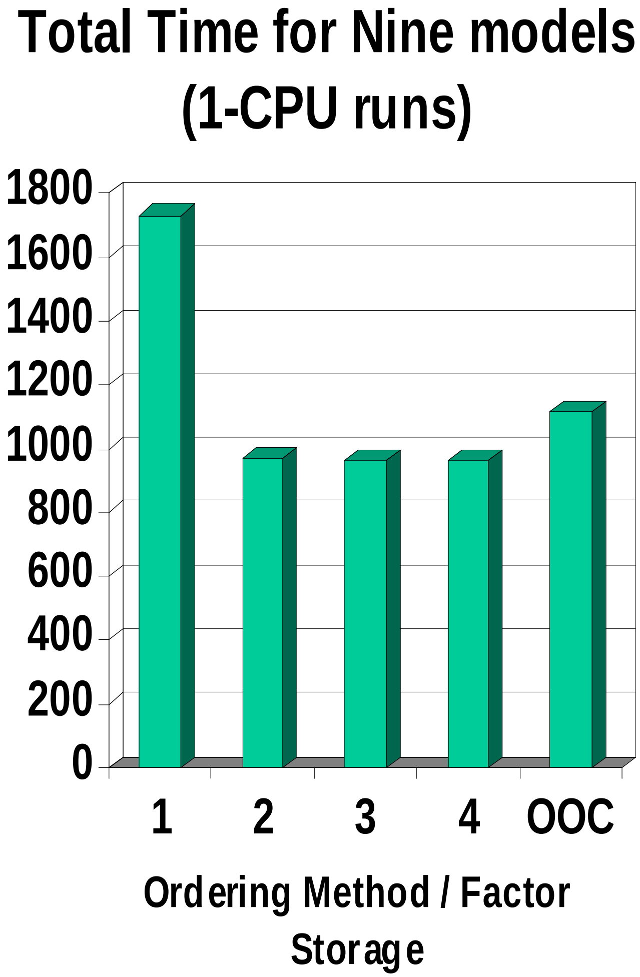
Choosing a default method

- Should default be best for which size model?
- Decided to optimize for medium or larger problems (at least 5000 equations)
- Extreme2 (3) about 3% faster than Extreme, but is new tech., so we use Method 2 as the new default.



Out-of-core (OOC) Option

- Performance 15% slower here than extreme (Method 2) ordering in-core...
- but faster than AMF (1)
- This used 4-way striping on file system -- 140 MB/s on some reads
- Allowed 128MB in-core for factor storage



New Sparse Solver Environment Variables

- SPARSE_NUM_ORDERS can be used to change the number of orderings performed from the default of 2*OMP_NUM_THREADS; Examples:
  - 'setenv SPARSE_NUM_ORDERS 100' to get a best-case ordering information into the feedback file
  - 'setenv SPARSE_NUM_ORDERS 1' when you have run a number of matrix orderings already (with method=4)

- SPARSE_FEEDBACK_FILE can be set to the path and file name where the feedback information will be kept. The default feedback file used otherwise is $HOME/.sparseFeedback.

Comparison of ordering methods

- Structural model with ~36K degrees of freedom
- Shows extra ordering time for new options
- Time reflects 1 preprocess, 4 factorizations & 2 solves
- “4 with history” ==> ‘setenv SPARSE_NUM_ORDERS 1’ since already have good info in SPARSE_FEEDBACK_FILE

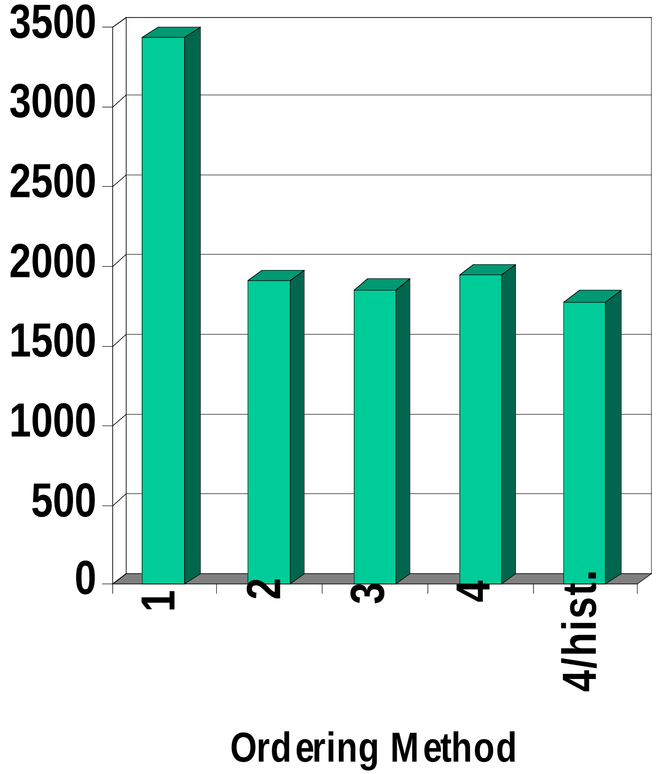
Ordering method

Matrix	BCSSTK01		nnzAA':1906964, n:35991			R10K/250
		Total time				
CPUs	1	2	3	4	4 w/ hist	4h v. 2
1	71	60	50.1	47.7	45.7	1.31
2	44.6	34.5	29	28.9	28.9	1.19
4	26.8	23.8	22.3	21.2	19.7	1.21
		Ordering time				
CPUs	1	2	3	4	4 w/ hist	
1	1.4	1.5	1.9	2.8	1.9	
2	1.5	1.6	2.2	3	1.9	
4	1.4	1.7	3.2	3.9	2.4	

Using the learning

- After doing a 4, 2 and 1 thread solves (history of 14 orderings), repeat using 'setenv SPARSE_NUM_ORDERS 1'
- Small advantage on these 9 matrices, more results shown soon.

Total Time for Nine models (1, 2 & 4 CPU runs)



Performance Results

Overview

- Applications already integrating new solver:
  - Abaqus
  - CPLEX
- PSLDLT performance
  - Case study
  - Large / dense model scalability
- Comparison to a public domain solver -- SPOOLES

ABAQUS: first to use

Uses a default of 36 orderings

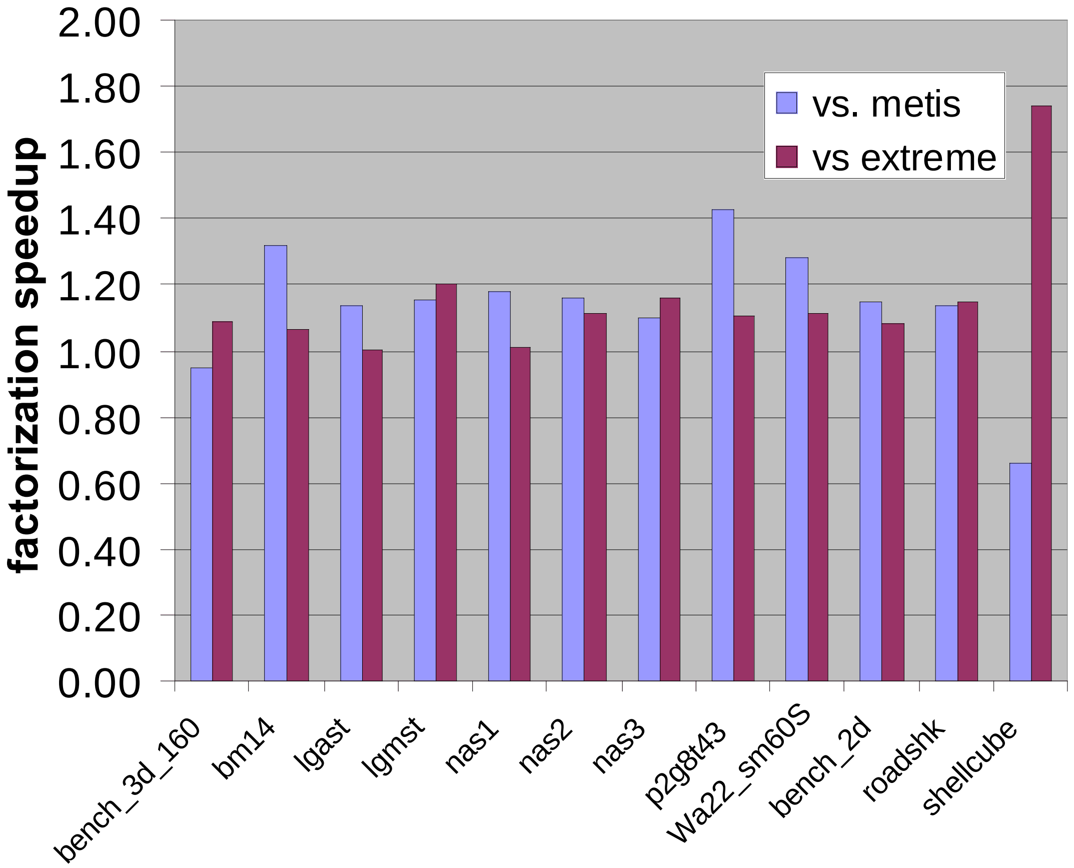
- Can be changed (BEST_HIGH env var)
- Best seed can be output by setting DUMP_EXTREME_INFO
- Seed can be input by SEED env var

ABAQUS a good use of this technology

- one ordering is used hundreds of times

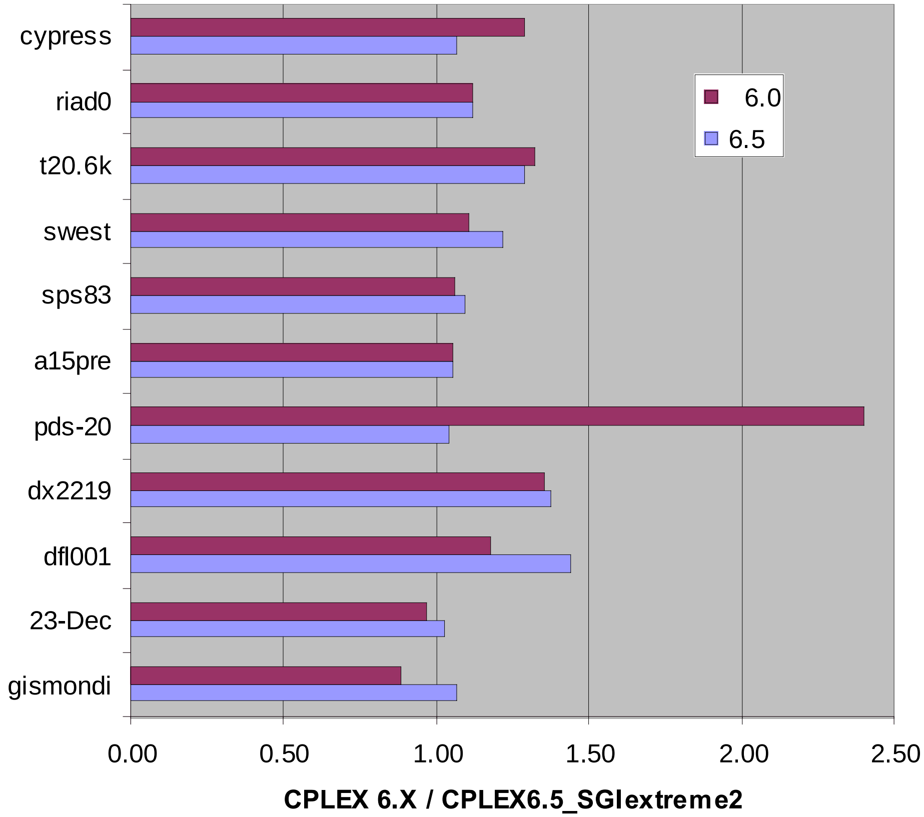
ABAQUS Performance

- 12 of 28 benchmarks shown here
- Extreme2 shows: 18% improvement over extreme; 10% improvement over METIS (public domain S/W)



Performance Results: as a CPLEX add-on

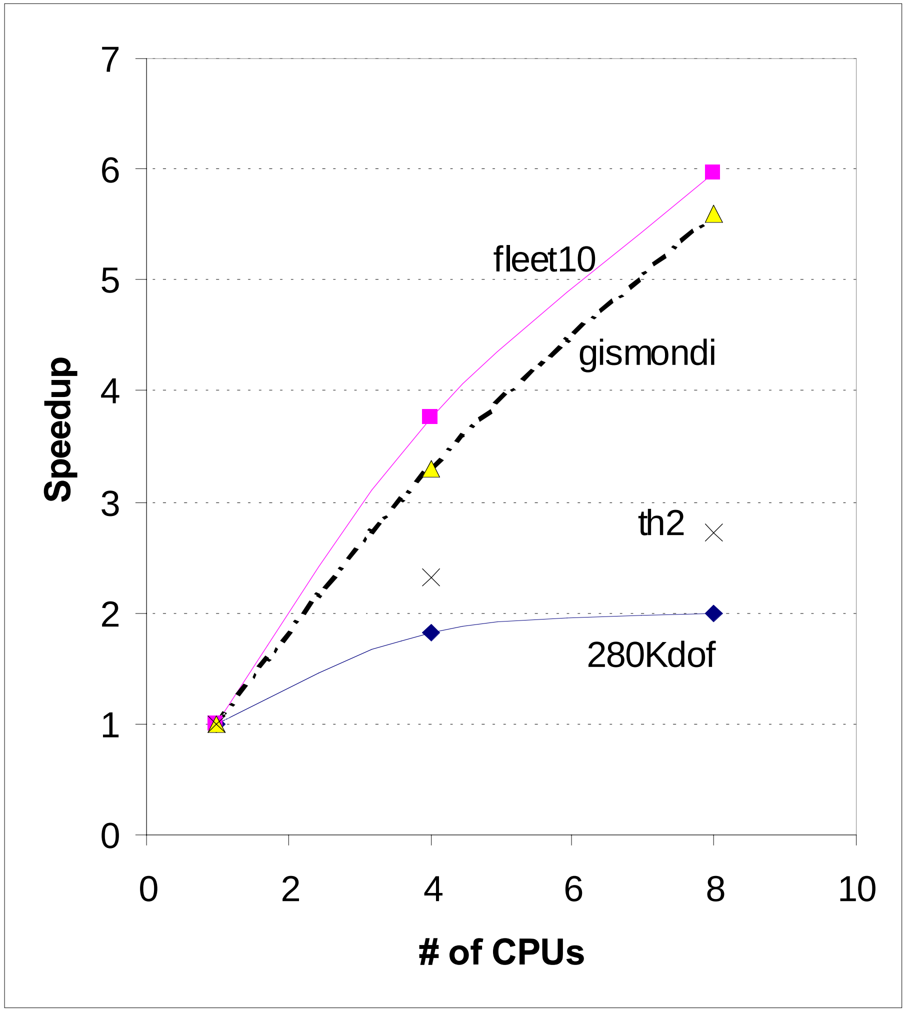
- ILOG provides ordering ‘hook’ so customer can link SGI extreme2 matrix ordering w/ CPLEX 6.5.
- Speedups vary from nothing to ~2x. Average 16% over default 6.5 on this set of (mostly high fill-in) models -- 25% over CPLEX 6.0
- Uses a ‘history’ of 50 orderings.



PSLDLT: Scalability to 8 CPUs

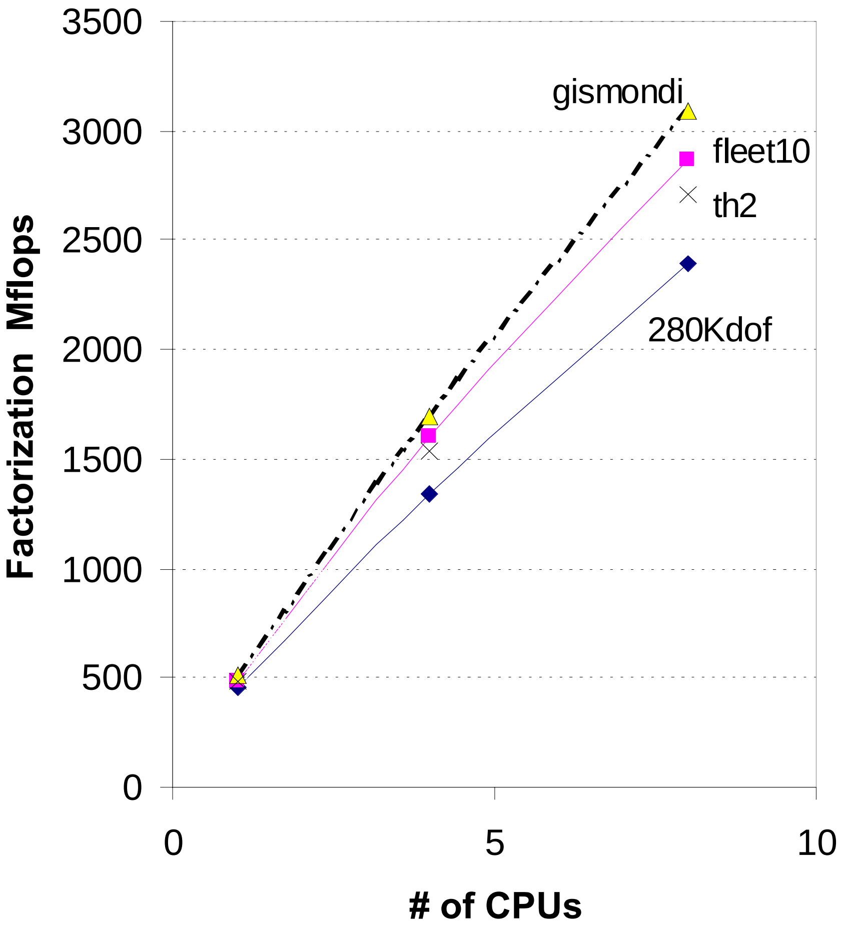
- Measured: Elapsed time for 1 preprocess, 2 factorizations, 2 solves.
- # f.p. ops to factor & preprocess time :

	Gflop	secs.
fleet10	383	27
gismondi	133	3
th2	34	18
280Kdof	18	15



Scalability: Factorization Mflops

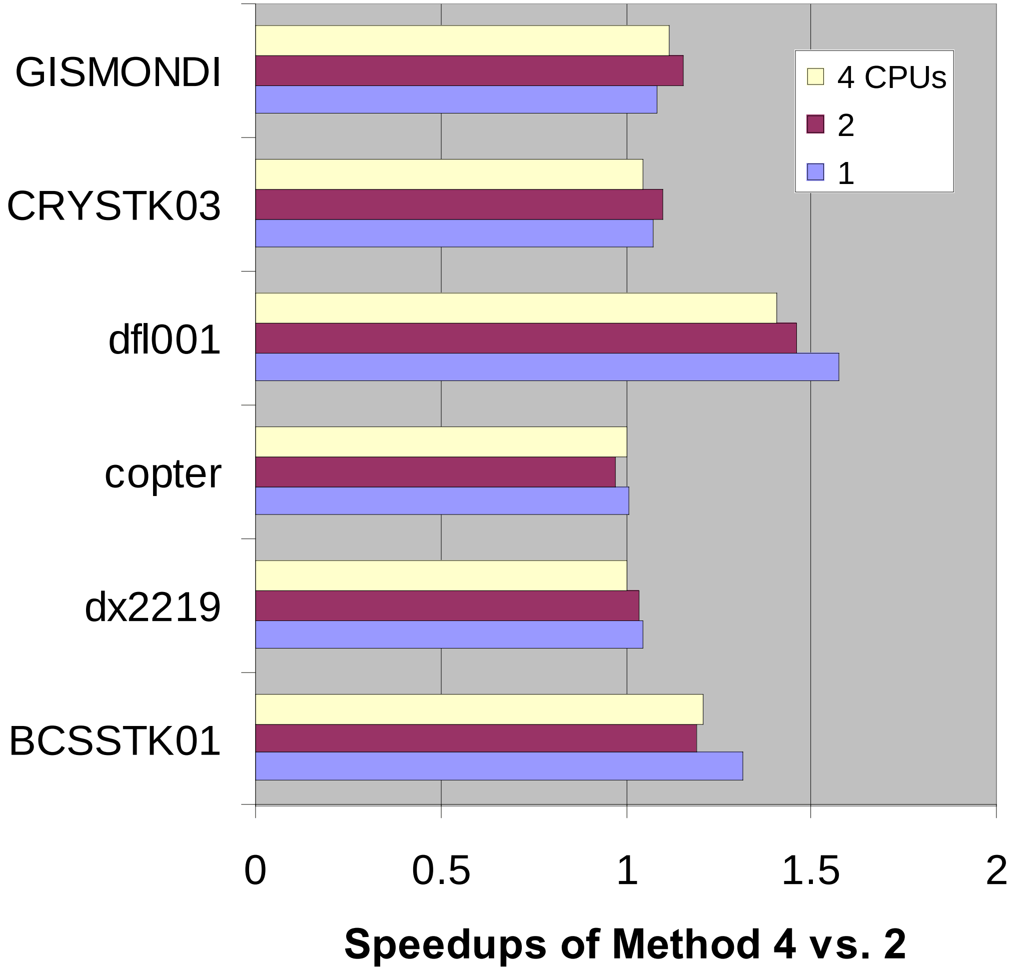
- Amdahl's law resp. for much of lack of scaling in previous chart
- Over 11 Gflops achieved on gismondi on 48 CPUs
- More can be done to improve memory placement
- These results used DSM_ROUND_ROBIN data placement



PSLDLT Perf. Results

Extreme2 with feedback (method 4) vs. old extreme speedups

Speedups can vary by #CPUs based on how super-nodes split into panels



A Public Domain Alternative

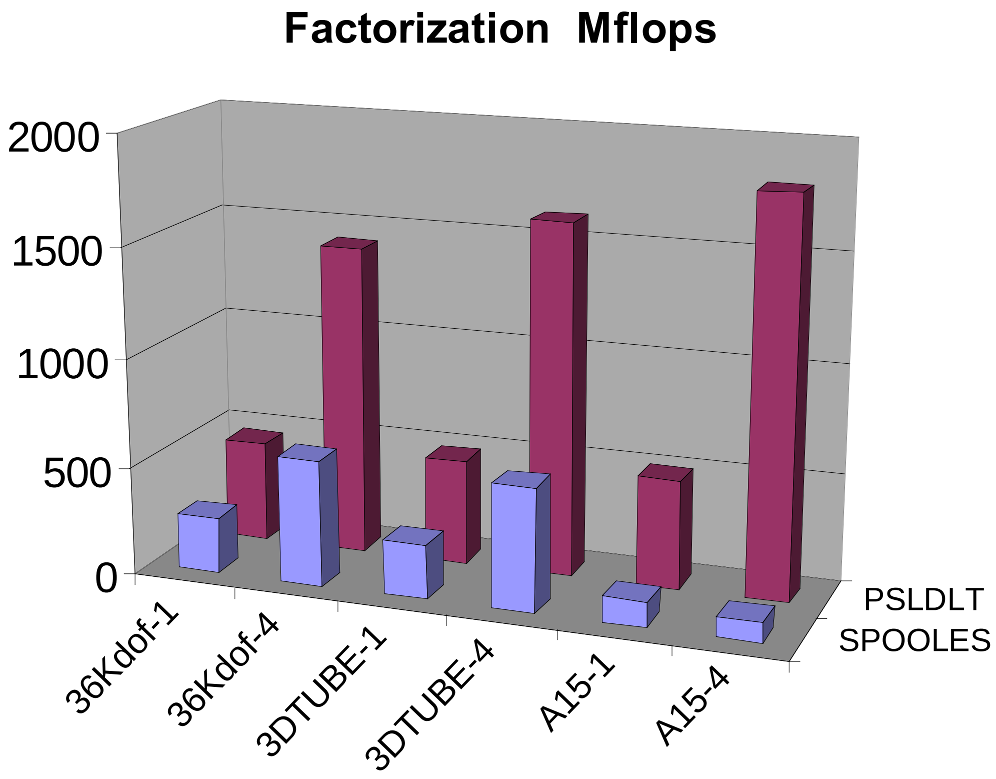
SPOOLES Library: Sparse Object-Oriented Linear Equation Solver

- As Object-Oriented as C allows
- Solves Real/Complex, Symmetric/Non-symm.
- With or without pivoting for stability
- Serial or Parallel (Pthreads or MPI)

Comes with various example programs -- the following results are from the LinSol MT wrapper object and driver

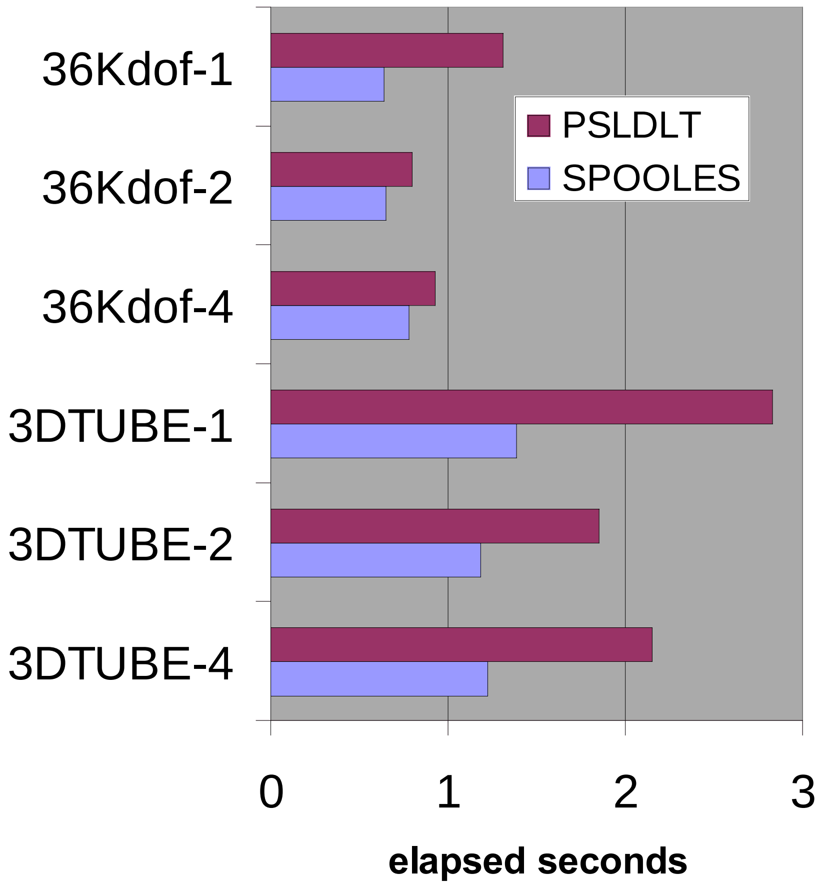
Factorization comparison

- PSLDLT faster on 1-CPU and better scalability
  - (in chart,
-1 ==> 1 CPU
-4 ==> 4 CPUs)
- A15 has a few large, dense supernodes – PSLDLT has been designed to handle
- Not a fair comparison:
  - PSLDLT has kernels (in C) hand-optimized for MIPS CPUs & large caches;
  - SPOOLES is more general; has pivoting option



Triangular Solve Comparison

- SPOOLES about twice as fast on the solve after the factorization
- Solve time is small % of total:
  - 1.5% (SPOOLES)
  - 5% (PSLDLT)
- We have some work to do



Preprocessing comparison

- Time: PSLDLT –Method 3 – does 1 ordering per thread & is generally faster; SPOOLES/LinSol uses 2 methods in serial
- Quality: PSLDLT (3) generally fewer factorization ops and can improve with more threads

		SPOOLES	PSLDLT	SPOOLES	PSLDLT
Matrix	**CPUs**	*elapsed seconds*		*factor ops (billions)*	
36Kdof	1	2.61	1.40	3.60	3.96
	2	2.61	1.60	3.60	3.96
	4	2.67	2.15	3.60	3.73
3DTUBE	1	6.95	3.18	13.60	12.10
	2	6.96	3.64	13.60	12.10
	4	6.95	4.33	13.60	12.00
TH2	1	47.52	12.41	38.90	35.80
	2	45.14	14.25	38.90	33.80
	4	47.41	17.08	38.90	30.50

Summary

- New default ordering option: Extreme (big speedups for larger/denser models)
- New matrix ordering option: Extreme2
  - Primarily useful when many factorizations will be done on one non-zero structure
- Out-of-core capabilities available (single-processor)

Possible futures for sparse solvers

- Tuning & algorithm improvements:
  - ordering and factorization scalability
  - triangular solve performance
- General sparse solver with pivoting?
- Port to IA-64 & IA-32? Linux or NT?
- Hybrid direct / iterative methods