BeBOP:
Berkeley Benchmarking and Optimization

Automatic Performance Tuning of Numerical Kernels

James Demmel
EECS and Math
UC Berkeley

Katherine Yelick
EECS
UC Berkeley

Support from DOE SciDAC, NSF, Intel

Performance Tuning Participants

- Faculty
  - Michael Jordan, William Kahan, Zhaojun Bai (UCD)
- Researchers
  - Mark Adams (SNL), David Bailey (LBL), Parry Husbands (LBL), Xiaoye Li (LBL), Lenny Oliker (LBL)
- PhD Students
  - ***Rich Vuduc***, Yozo Hida, Geoff Pike
- Undergrads
  - Brian Gaeke , Jen Hsu, Shoaib Kamil, Suh Kang, Hyun Kim, Gina Lee, Jaeseop Lee, Michael de Lorimier, Jin Moon, Randy Shoopman, Brandon Thompson

Outline

- Motivation, History, Related work
- Tuning Sparse Matrix Operations
- Results on Sun Ultra 1/170
- Recent results on P4
- Recent results on Itanium
- Some (non SciDAC) Target Applications
  - SUGAR – a MEMS CAD system
  - Information Retrieval
- Future Work

Motivation
History
Related Work

Conventional Performance Tuning

- Motivation: performance of many applications dominated by a few kernels
- Vendor or user hand tunes kernels
- Drawbacks:
  - Very time consuming and tedious work
  - Even with intimate knowledge of architecture and compiler, performance hard to predict
  - Growing list of kernels to tune
    - Example: New BLAS Standard
  - Must be redone for every architecture, compiler
  - Compiler technology often lags architecture
  - Not just a compiler problem:
    - Best algorithm may depend on input, so some tuning at run-time.
    - Not all algorithms semantically or mathematically equivalent

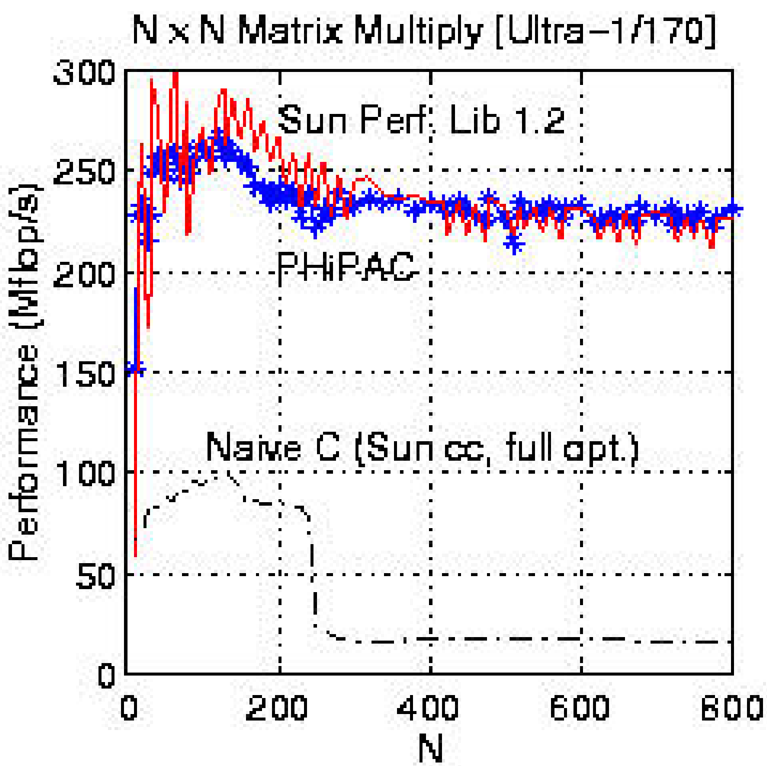
Automatic Performance Tuning

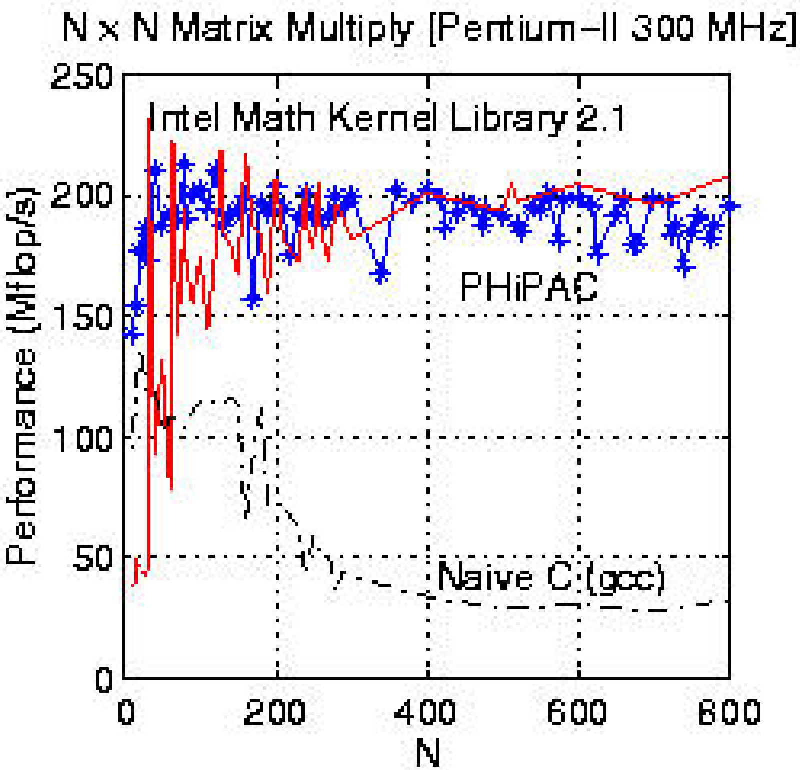
- Approach: for each kernel
  1. Identify and generate a space of algorithms
  2. Search for the fastest one, by running them
- What is a space of algorithms?
  - Depending on kernel and input, may vary
    - instruction mix and order
    - memory access patterns
    - data structures
    - mathematical formulation
- When do we search?
  - Once per kernel and architecture
  - At compile time
  - At run time
  - All of the above

Some Automatic Tuning Projects

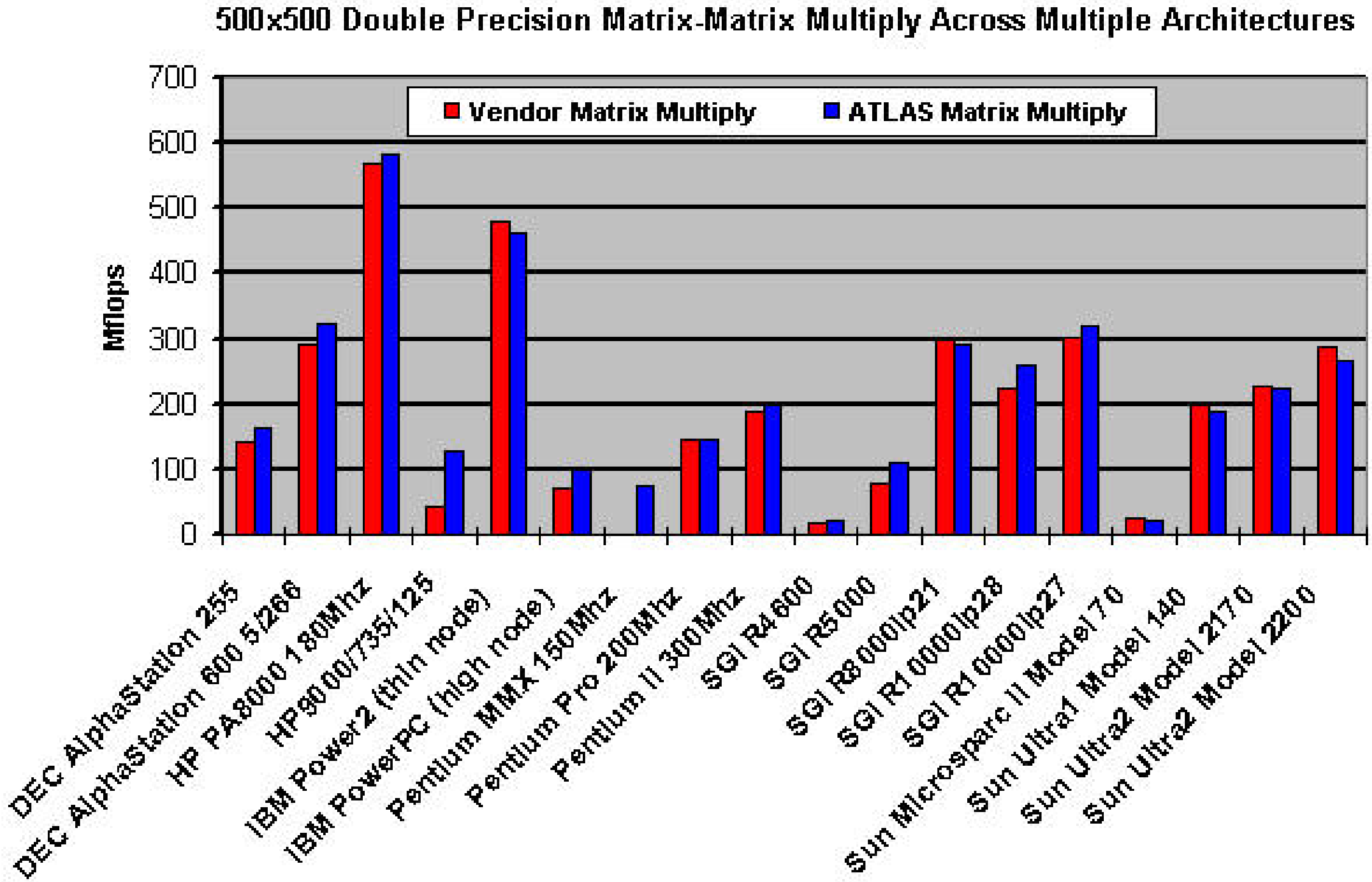
- PHIPAC (www.icsi.berkeley.edu/~bilmes/phipac) (Bilmes,Asanovic,Vuduc,Demmel)
- ATLAS (www.netlib.org/atlas) (Dongarra, Whaley; in Matlab)
- XBLAS (www.nersc.gov/~xiaoye/XBLAS) (Demmel, X. Li)
- Sparsity (www.cs.berkeley.edu/~yelick/sparsity) (Yelick, Im)
- FFTs and Signal Processing
  - FFTW (www.fftw.org)
    - Won 1999 Wilkinson Prize for Numerical Software
  - SPIRAL (www.ece.cmu.edu/~spiral)
    - Extensions to other transforms, DSPs
  - UHFFT
    - Extensions to higher dimension, parallelism
- Special session at ICCS 2001
  - Organized by Yelick and Demmel
  - www.ucalgary.ca/iccs
  - Proceedings available
  - Pointers to other automatic tuning projects at
    - www.cs.berkeley.edu/~yelick/iccs-tune

Tuning pays off – PHIPAC (Bilmes, Asanovic, Vuduc, Demmel)





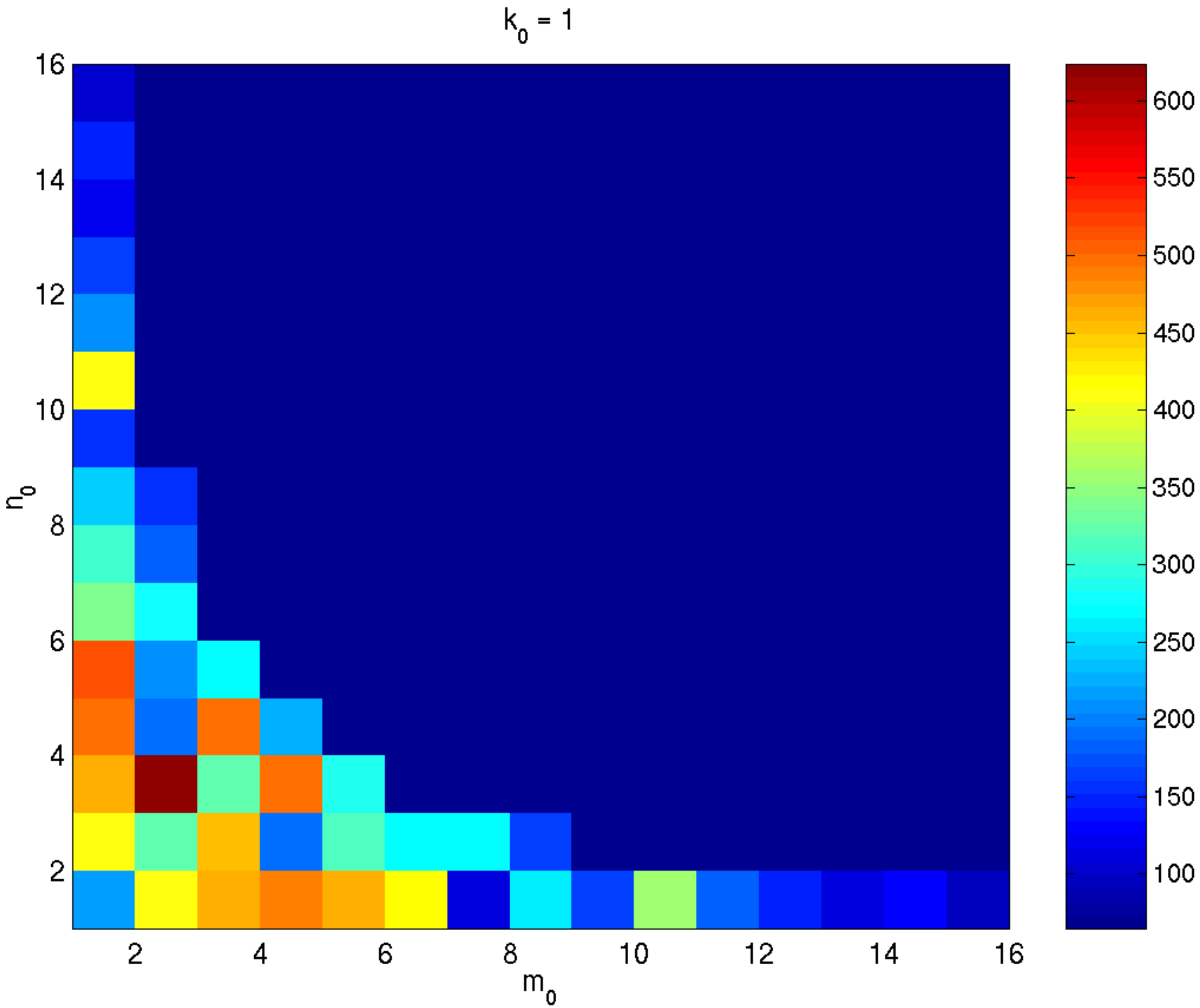
Tuning pays off – ATLAS (Dongarra, Whaley)



Extends applicability of PHIPAC

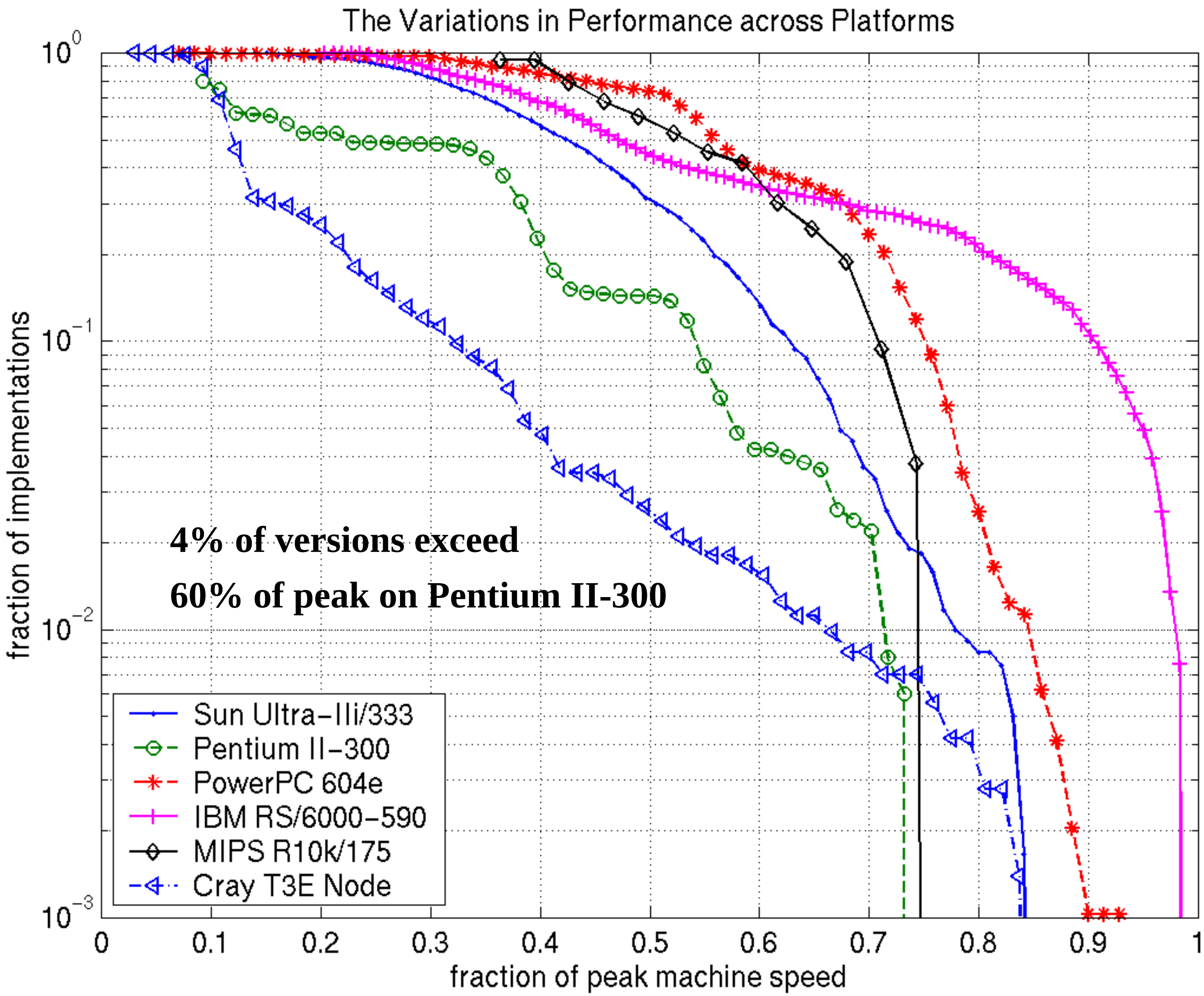
Incorporated in Matlab (with rest of LAPACK)

Search for optimal register tile sizes on Sun Ultra 10

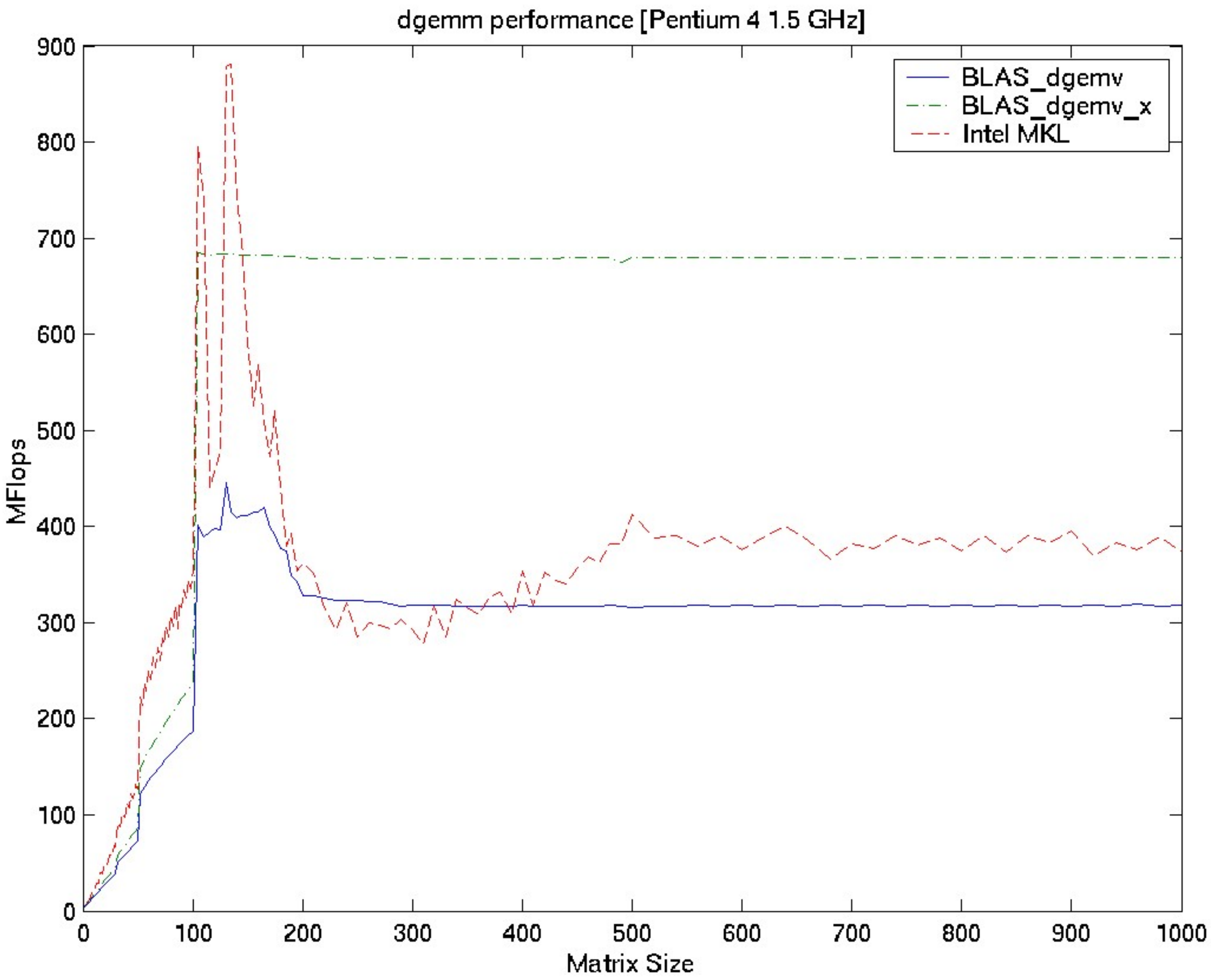


16 registers, but 2-by-3 tile size fastest

Search for Optimal L0 block size in dense matmul



High precision dense mat-vec multiply (XBLAS)



High Precision Algorithms (XBLAS)

- Double-double (High precision word represented as pair of doubles)
  - Many variations on these algorithms; we currently use Bailey's
- Exploiting Extra-wide Registers
  - Suppose s(1) , … , s(n) have f-bit fractions, SUM has F>f bit fraction
  - Consider following algorithm for $S = \Sigma_{i=1,n}\, s(i)$
    - Sort so that $|s(1)| \geq |s(2)| \geq \cdots \geq |s(n)|$
    - SUM = 0, for i = 1 to n SUM = SUM + s(i), end for, sum = SUM
  - Theorem (D., Hida) Suppose F<2f (less than double precision)
    - If $n \leq 2^{F-f} + 1$, then error $\leq$ 1.5 ulps
    - If $n = 2^{F-f} + 2$, then error $\leq 2^{2f-F}$ ulps (can be >> 1)
    - If $n \geq 2^{F-f} + 3$, then error can be arbitrary ($S \neq 0$ but sum = 0)
  - Examples
    - s(i) double (f=53), SUM double extended (F=64)
      - accurate if $n \leq 2^{11} + 1 = 2049$
    - Dot product of single precision x(i) and y(i)
      - s(i) = x(i)*y(i) (f=2*24=48), SUM double extended (F=64) $\Rightarrow$
      - accurate if $n \leq 2^{16} + 1 = 65537$

Tuning Sparse Matrix Computations

Tuning Sparse matrix-vector multiply

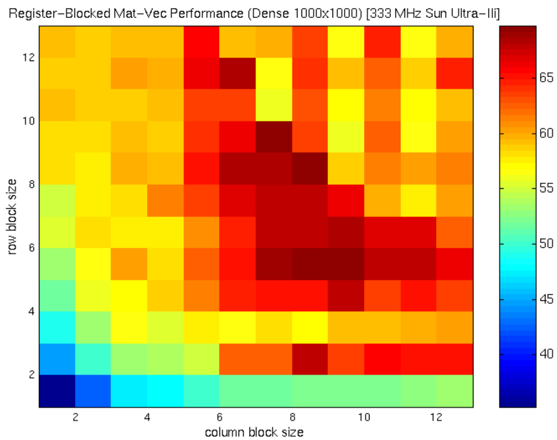
- Sparsity
  - Optimizes y = A*x for a particular sparse A
- Im and Yelick
- Algorithm space
  - Different code organization, instruction mixes
  - Different register blockings (change data structure and fill of A)
  - Different cache blocking
  - Different number of columns of x
  - Different matrix orderings
- Software and papers available
  - www.cs.berkeley.edu/~yelick/sparsity

How Sparsity tunes y = A*x

- Register Blocking
  - Store matrix as dense r x c blocks
  - Precompute performance in **Mflops** of dense A*x for various register block sizes r x c
  - Given A, sample it to estimate **Fill** if A blocked for varying r x c
  - Choose r x c to minimize estimated running time **Fill/Mflops**
    - Store explicit zeros in dense r x c blocks, unroll
- Cache Blocking
  - Useful when source vector x enormous
  - Store matrix as sparse 2^k x 2^l blocks
  - Search over 2^k x 2^l cache blocks to find fastest

Register-Blocked Performance of SPMV on Dense Matrices (up to 12x12)

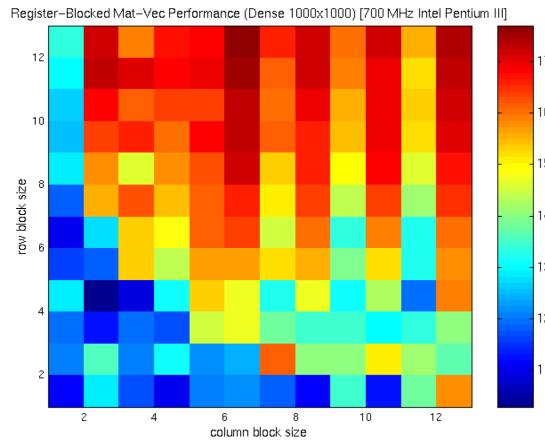
333 MHz Sun Ultra IIi



70 Mflops

35 Mflops

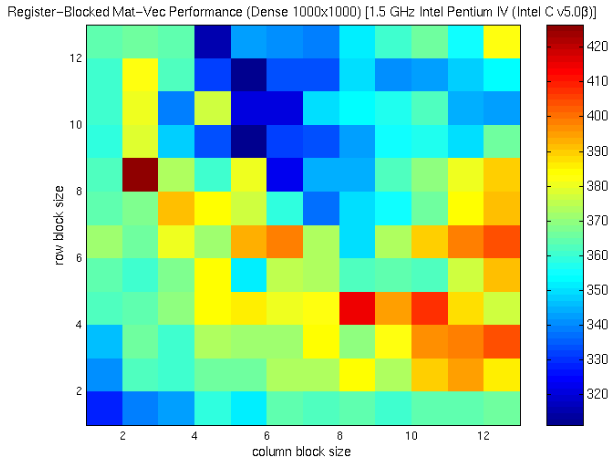
800 MHz Pentium III



175 Mflops

105 Mflops

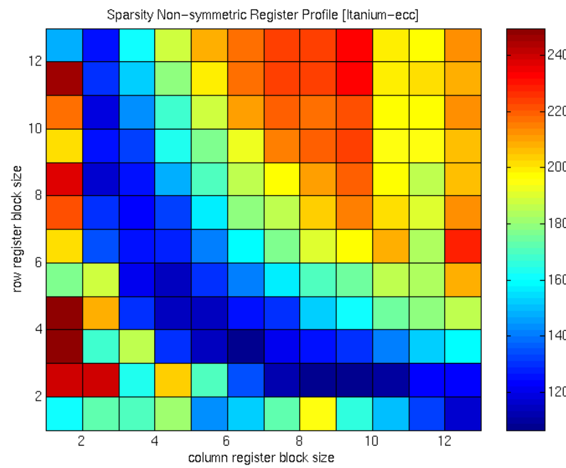
1.5 GHz Pentium 4



425 Mflops

310 Mflops

800 MHz Itanium



250 Mflops

110 Mflops

Which other sparse operations can we tune?

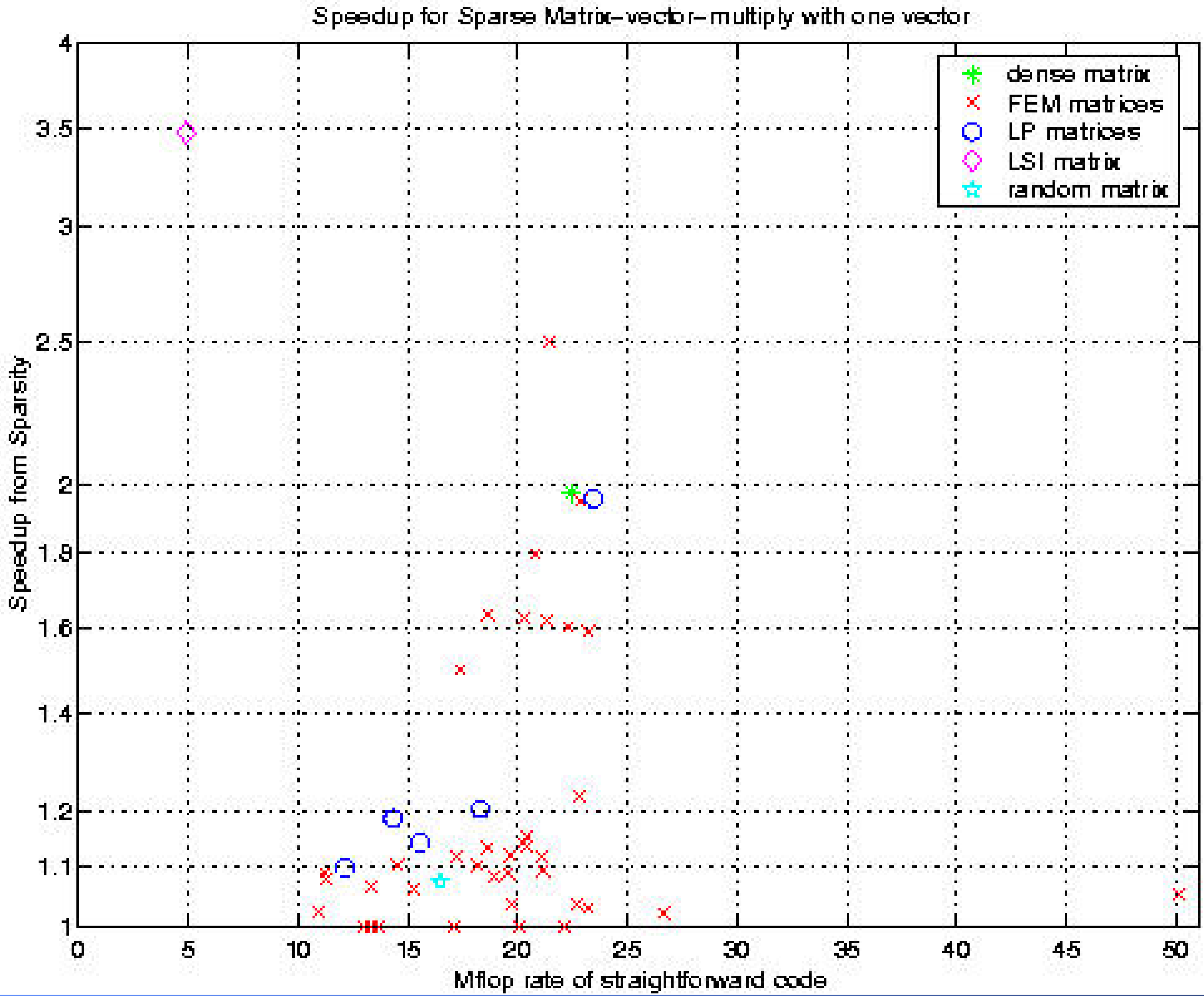
- General matrix-vector multiply $A*x$
  - Possibly many vectors x
- Symmetric matrix-vector multiply $A*x$
- Solve a triangular system of equations $T^{-1}*x$
- $y = A^T*A*x$
  - Kernel of Information Retrieval via LSI (SVD)
  - Same number of memory references as $A*x$
    - $y = \sum_i (A(i,:))^T * (A(i,:) * x)$
- Future work
  - A^2*x, A^k*x
    - Kernel of Information Retrieval used by Google
    - Includes Jacobi, SOR, …
    - Changes calling algorithm
  - A^T*M*A
    - Matrix triple product
    - Used in multigrid solver
  - What does SciDAC need?

Test Matrices

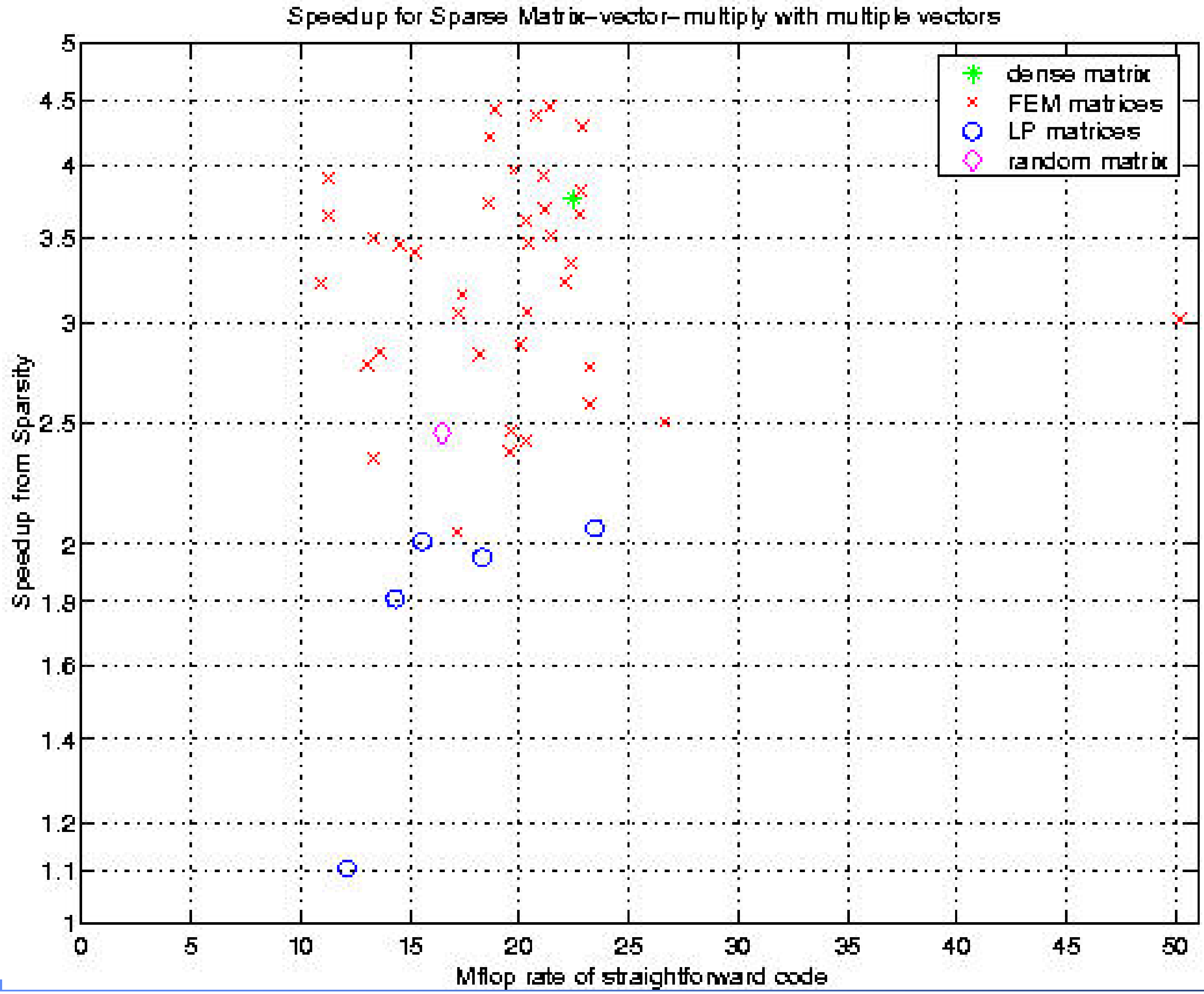
- General Sparse Matrices
  - Up to n=76K, nnz = 3.21M
  - From many application areas
  - 1 – Dense
  - 2 to 17 - FEM
  - 18 to 39 - assorted
  - 41 to 44 – linear programming
  - 45 - LSI
- Symmetric Matrices
  - Subset of General Matrices
  - 1 – Dense
  - 2 to 8 - FEM
  - 9 to 13 - assorted
- Lower Triangular Matrices
  - Obtained by running SuperLU on subset of General Sparse Matrices
  - 1 – Dense
  - 2 – 13 – FEM
- Details on test matrices at end of talk

Results on Sun Ultra 1/170

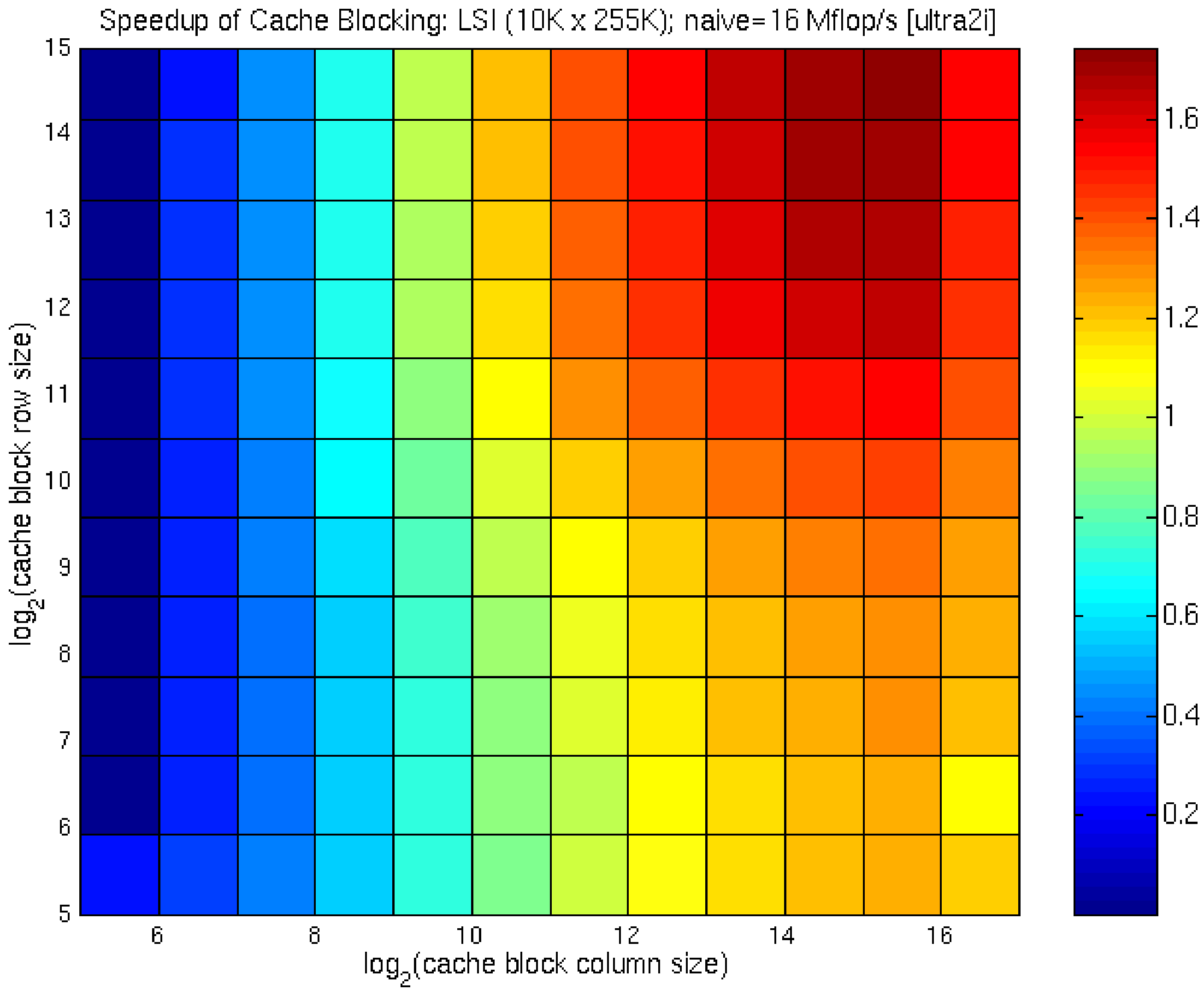
Speedups on SPMV from Sparsity on Sun Ultra 1/170 – 1 RHS



Speedups on SPMV from Sparsity on Sun Ultra 1/170 – 9 RHS

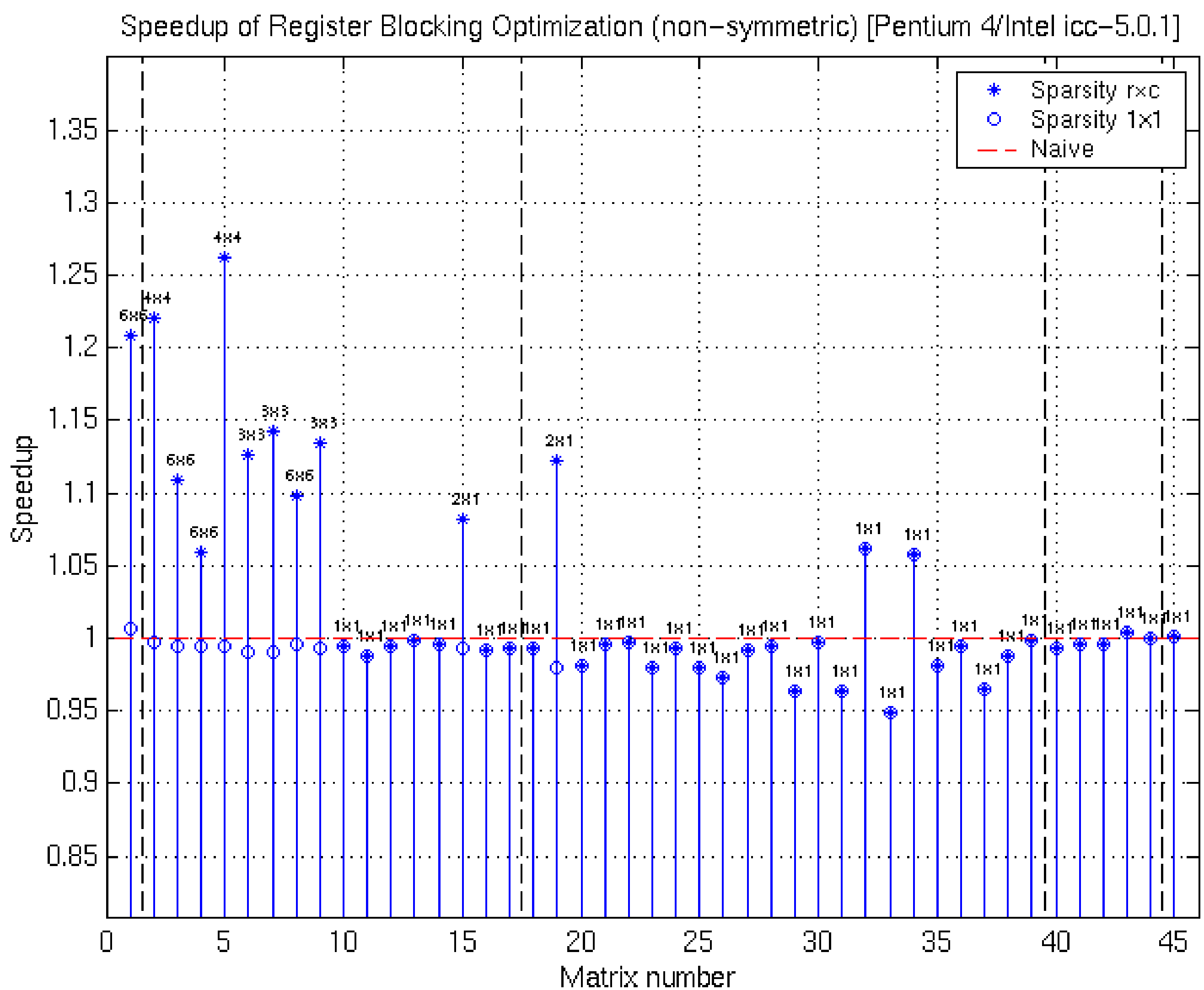


Speed up from Cache Blocking on LSI matrix on Sun Ultra

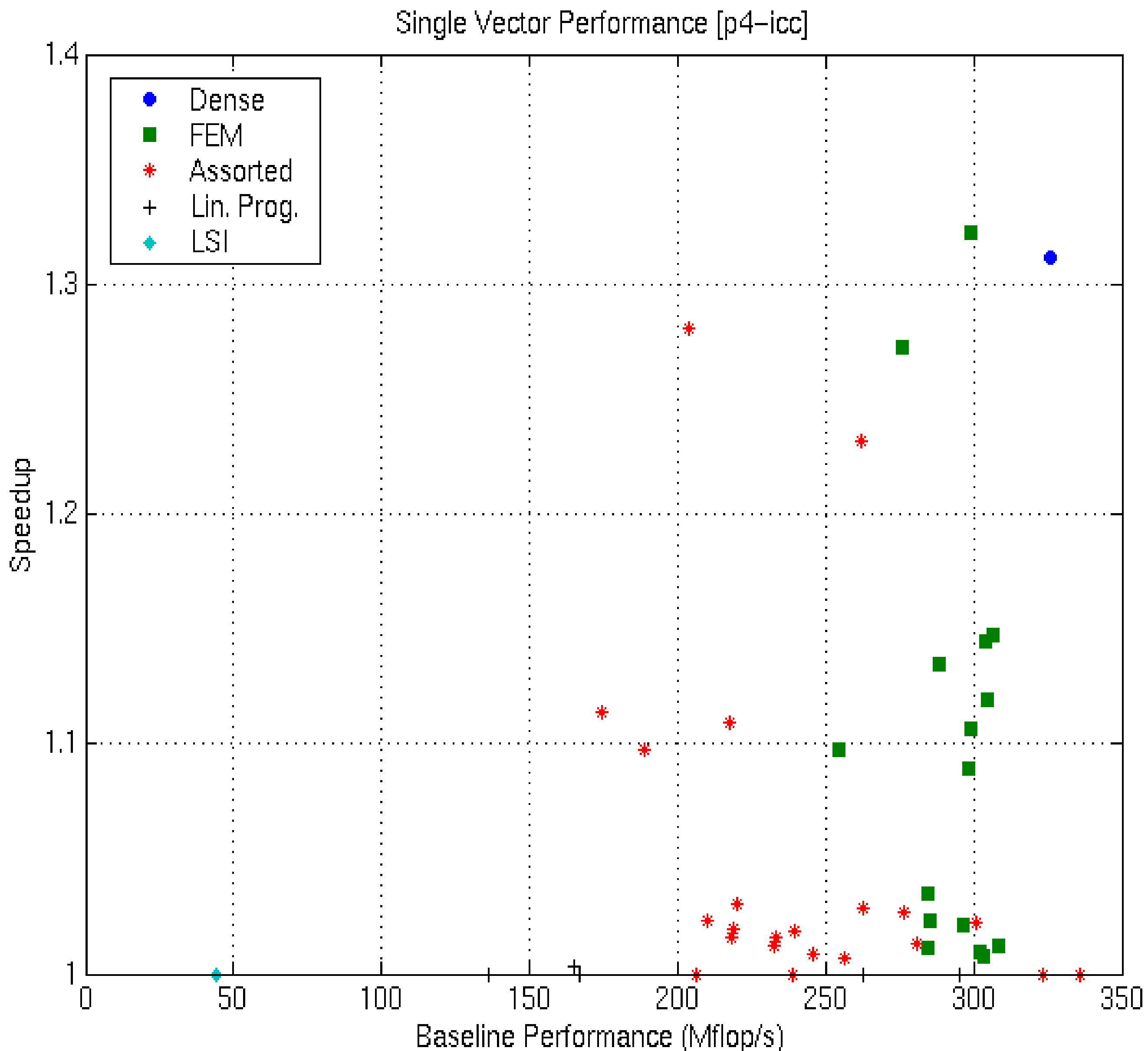


Recent Results
on P4 using icc and gcc

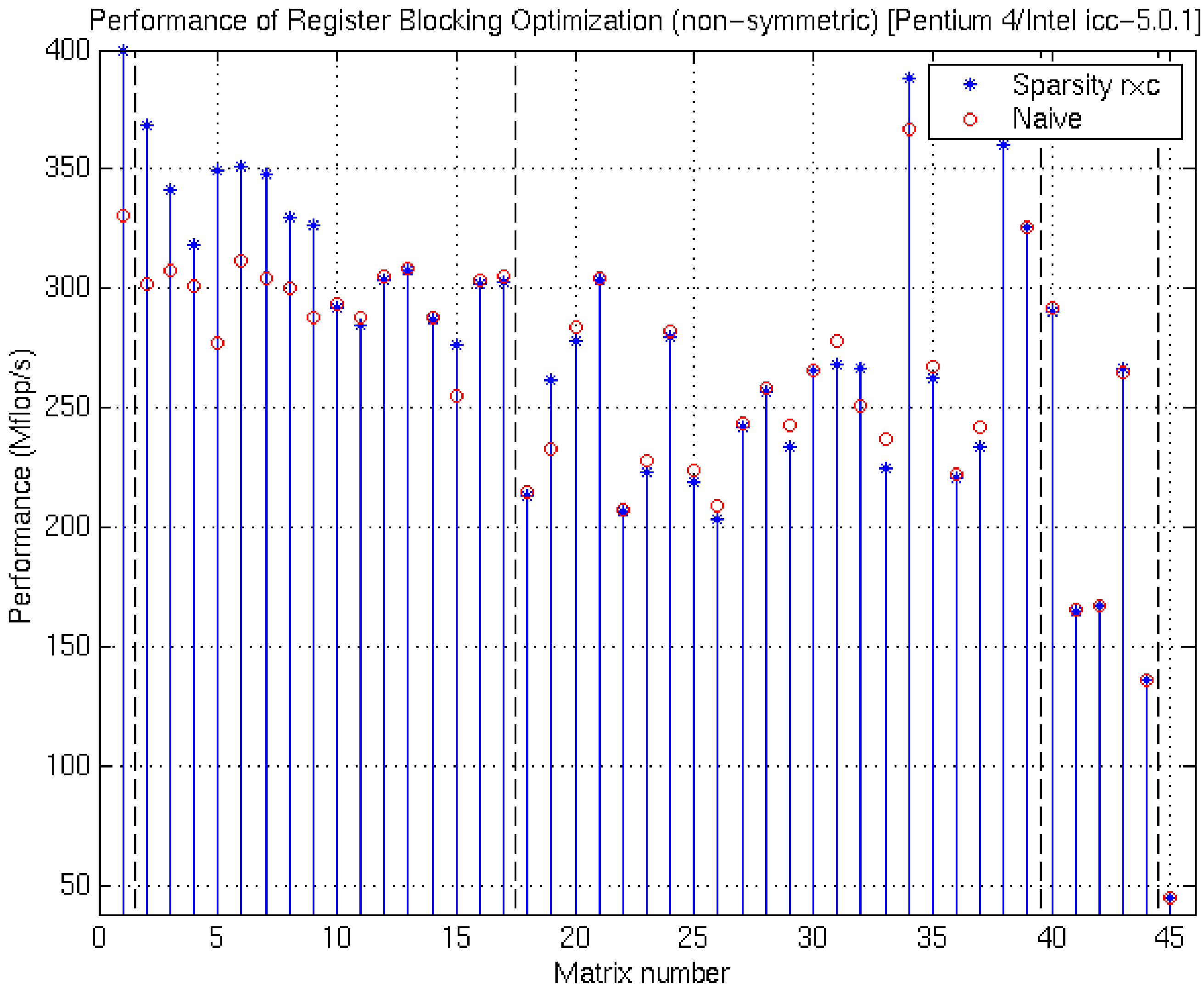
Speedup of SPMV from Sparsity on P4/icc-5.0.1



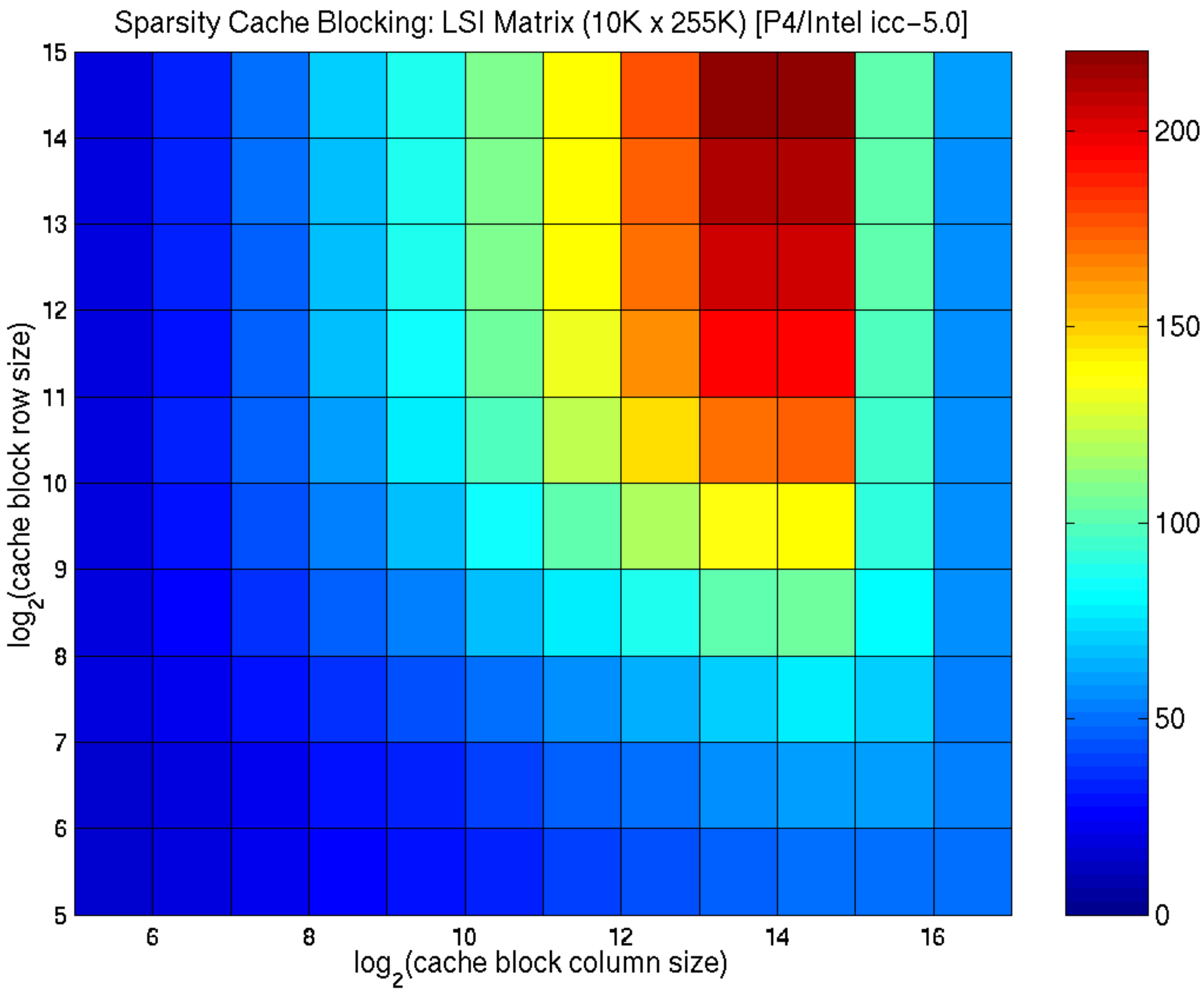
Single vector speedups on P4 by matrix type – best r x c



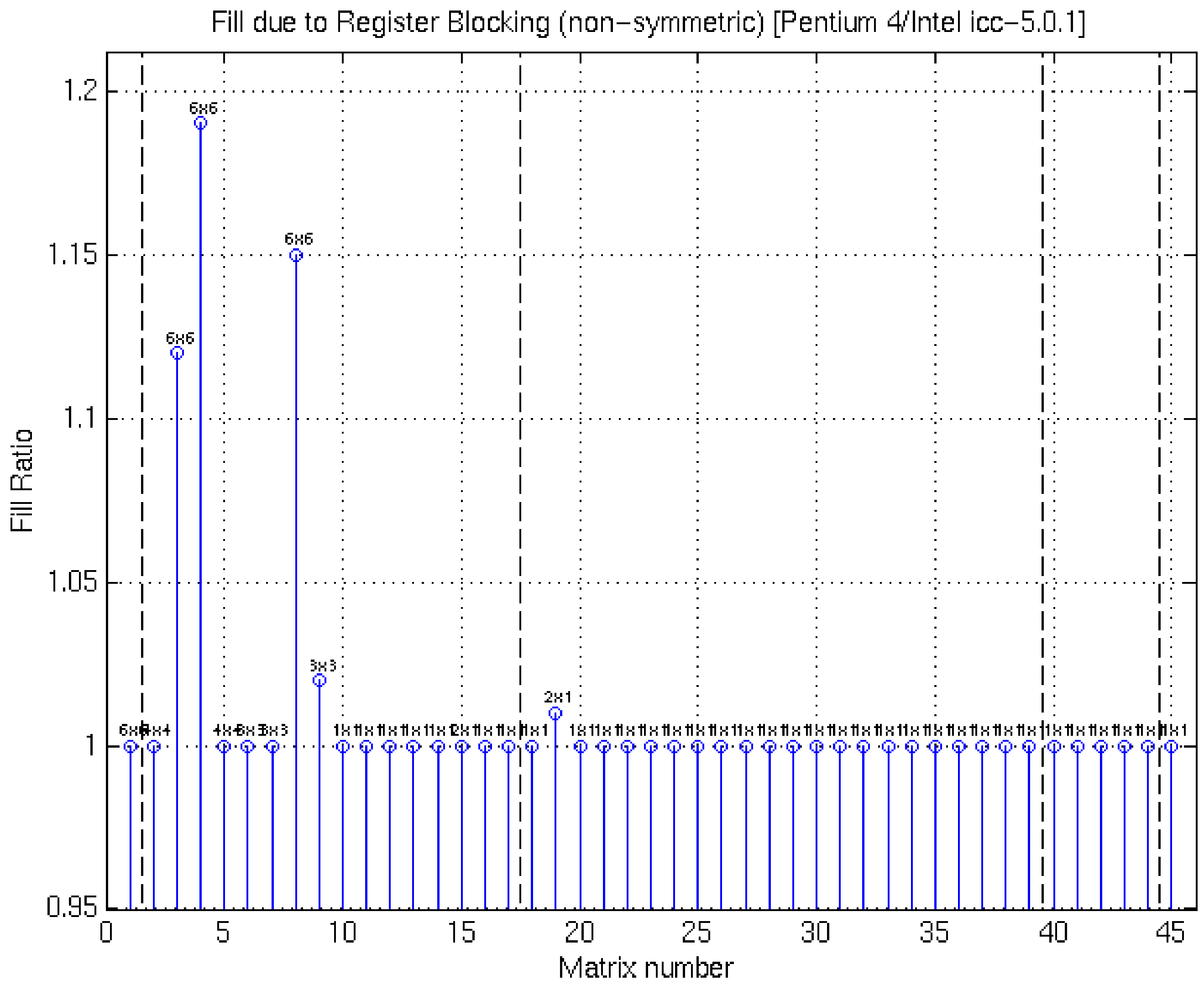
Performance of SPMV from Sparsity on P4/icc-5.0.1



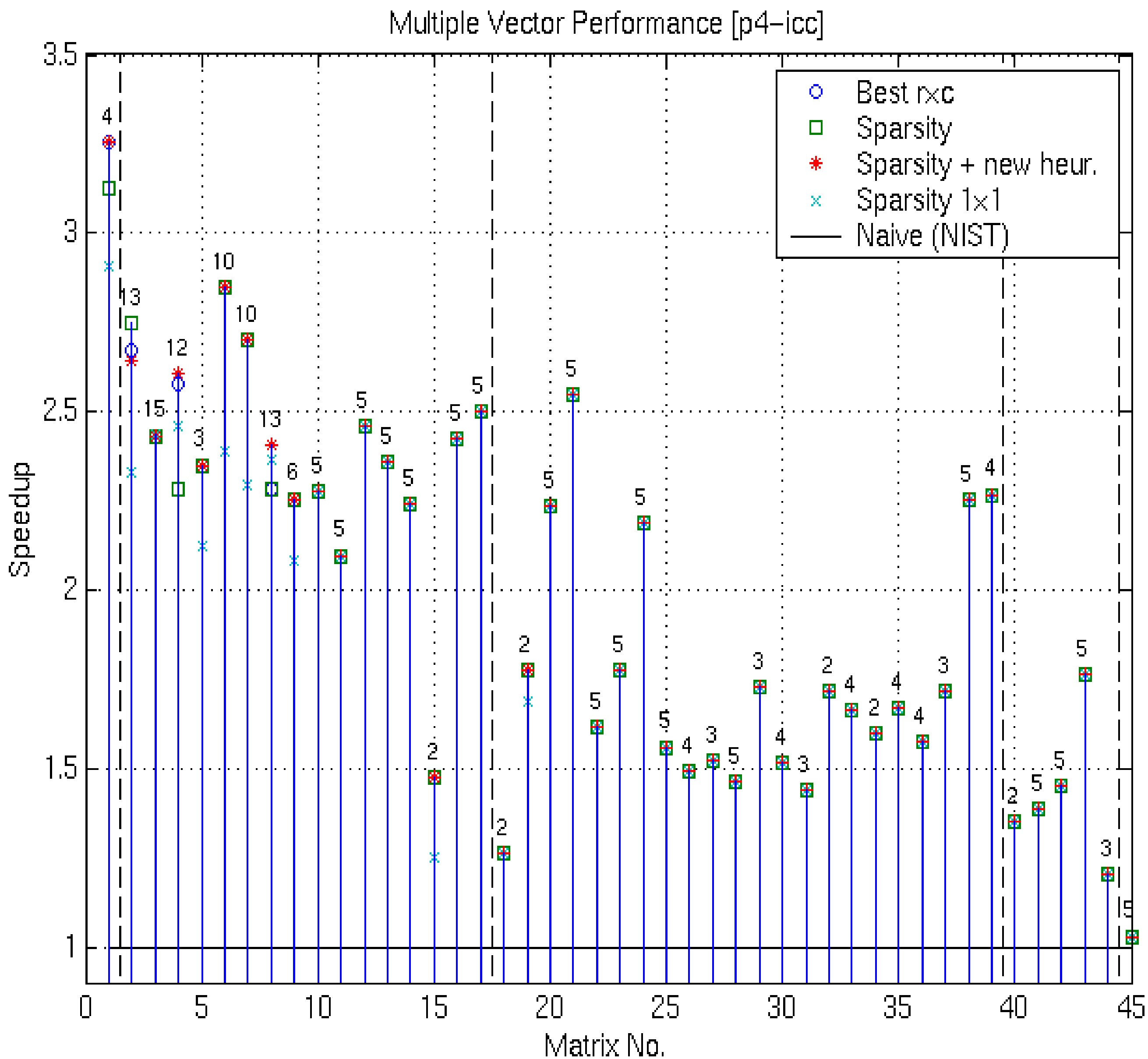
Sparsity cache blocking results on P4 for LSI



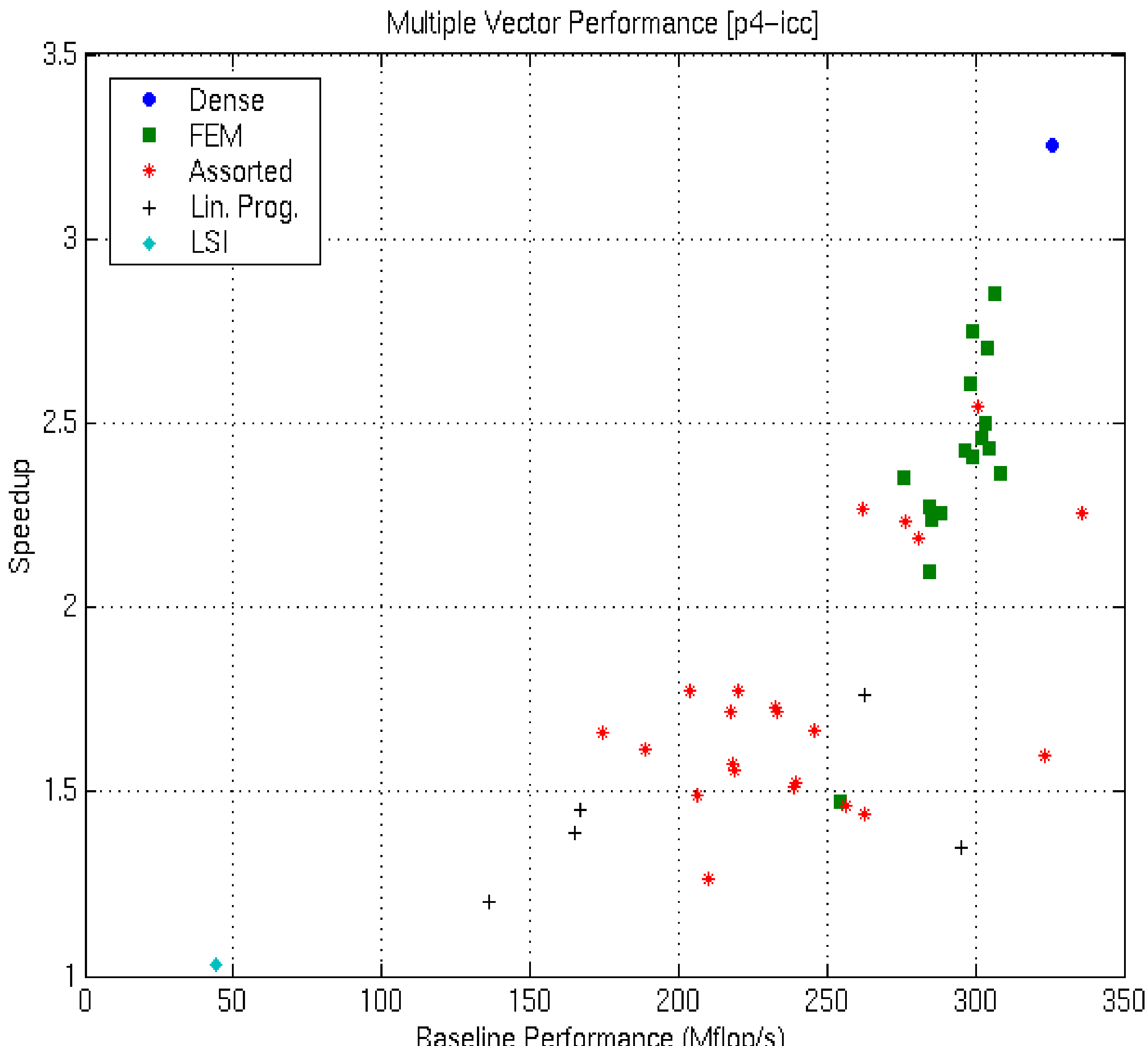
Fill for SPMV from Sparsity on P4/icc-5.0.1



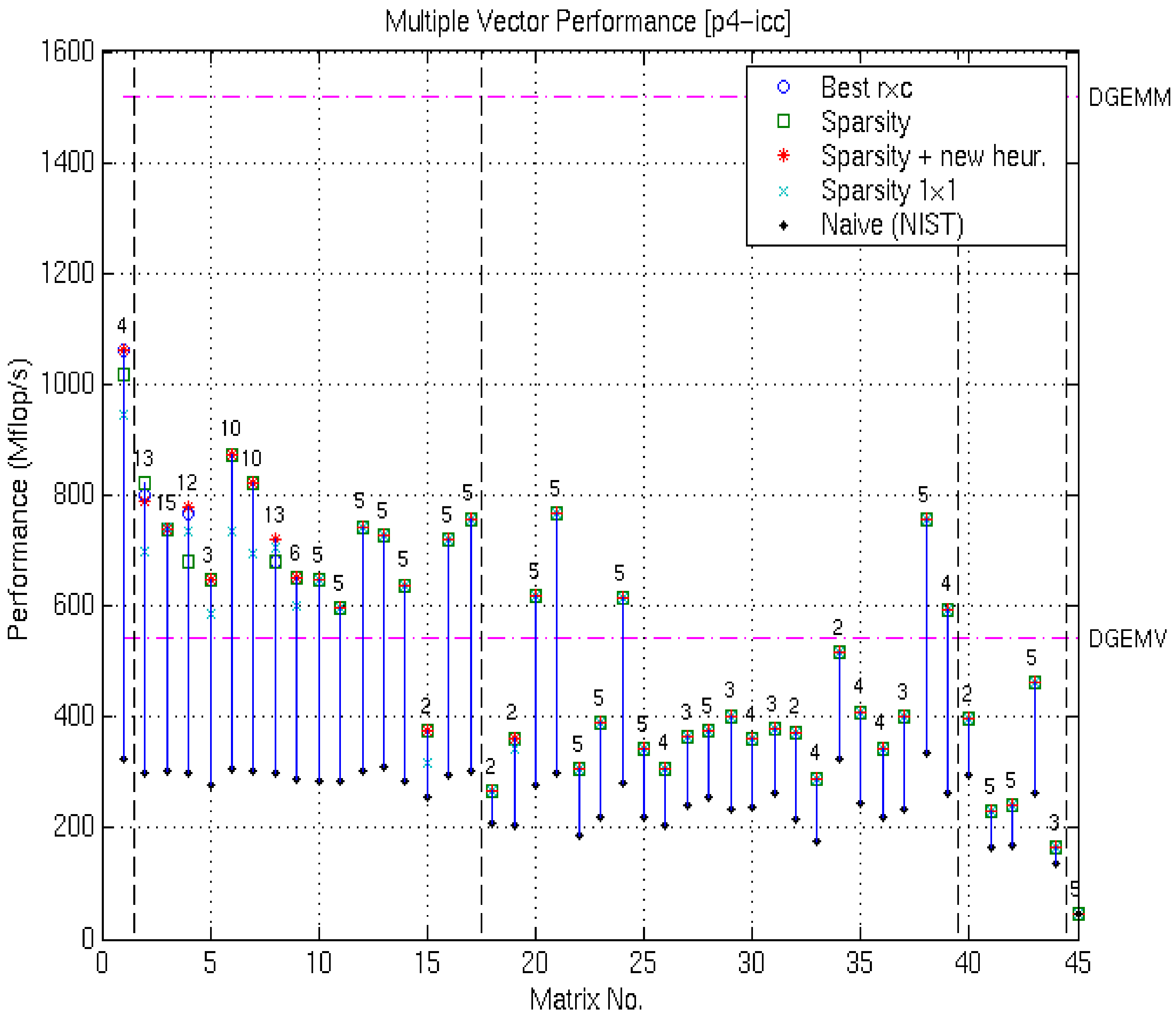
Multiple vector speedups on P4



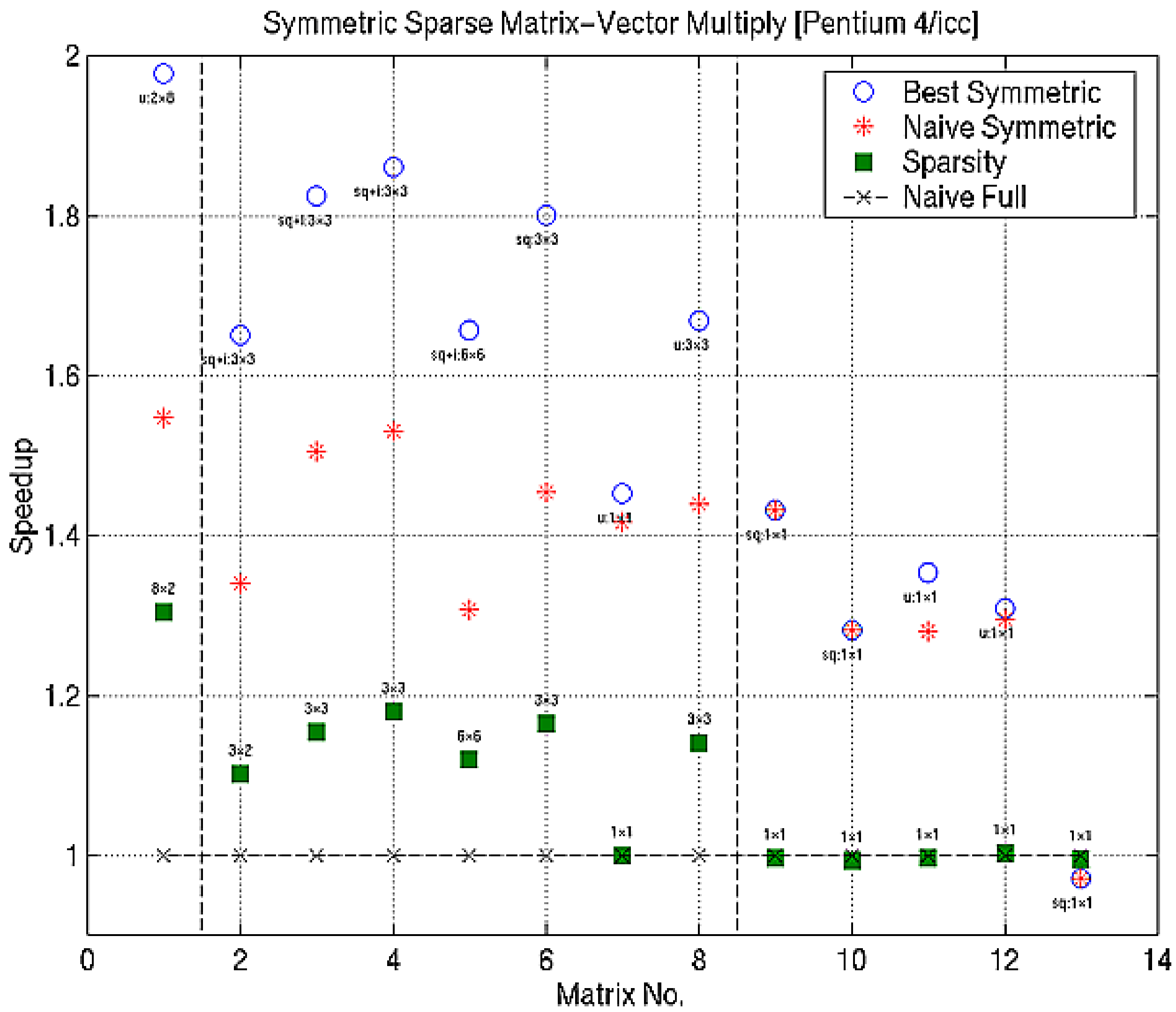
Multiple vector speedups on P4 – by matrix type



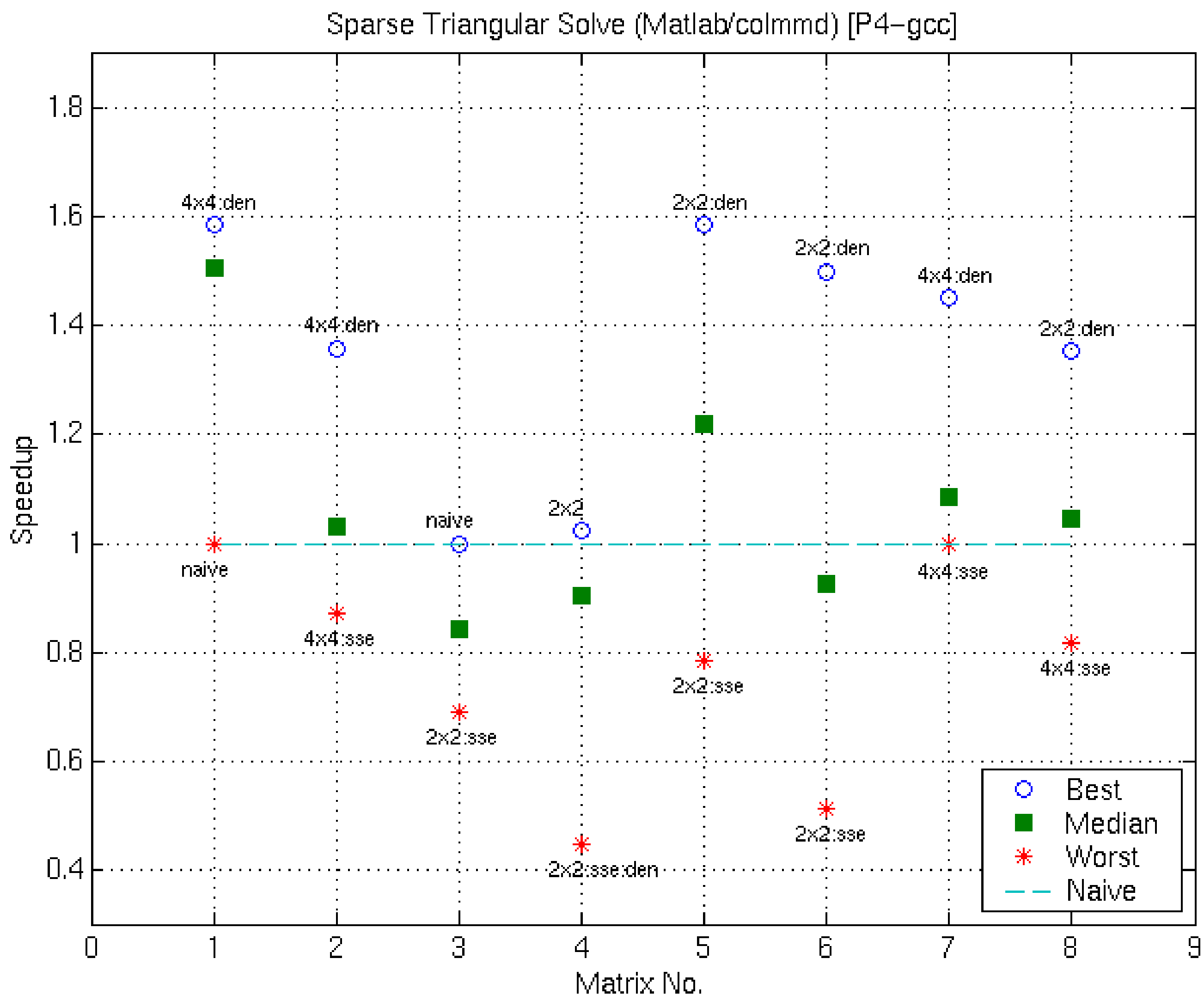
Multiple Vector Performance on P4



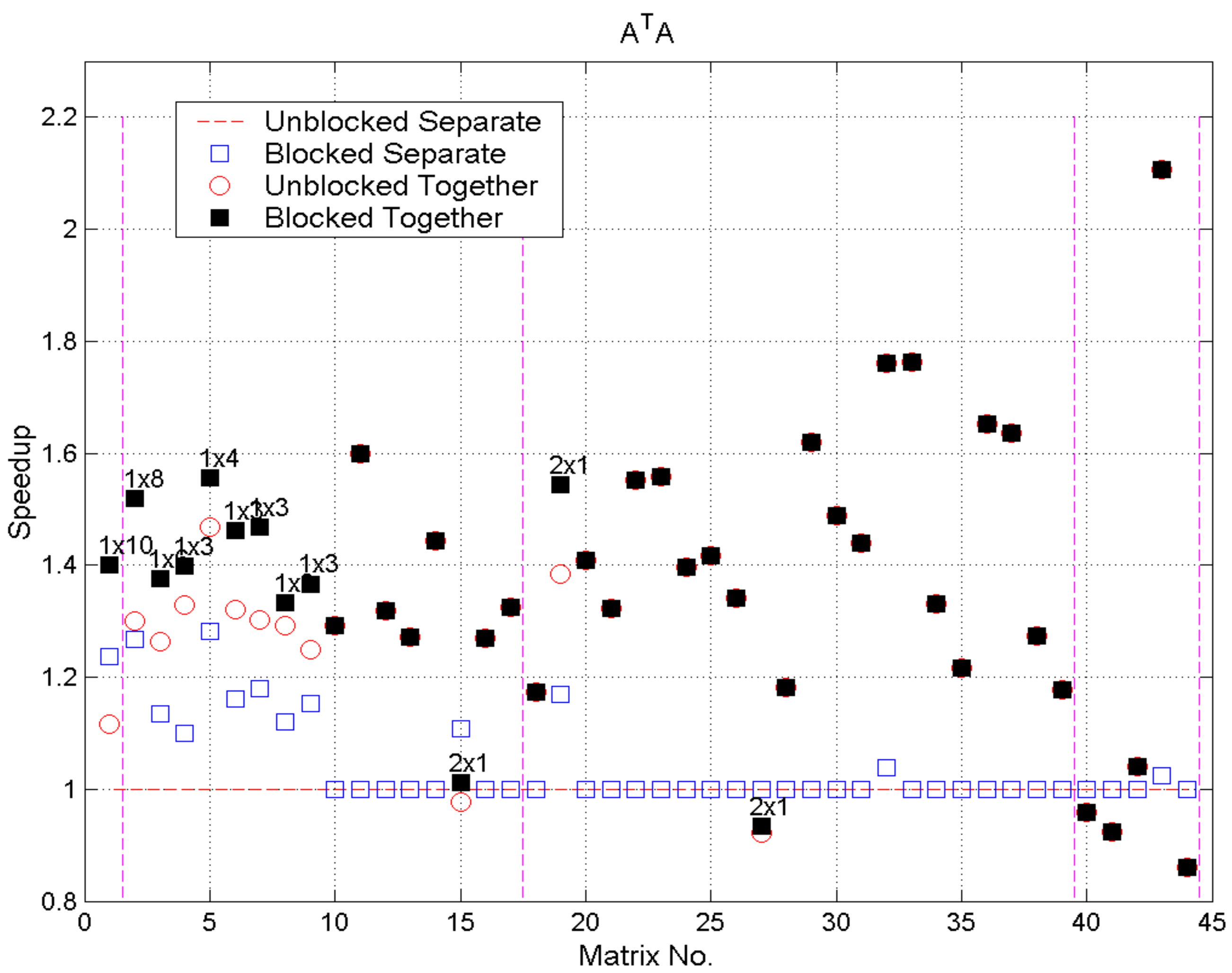
Symmetric Sparse Matrix-Vector Multiply on P4 (vs naïve full = 1)



Sparse Triangular Solve (Matlab's colmmd ordering) on P4

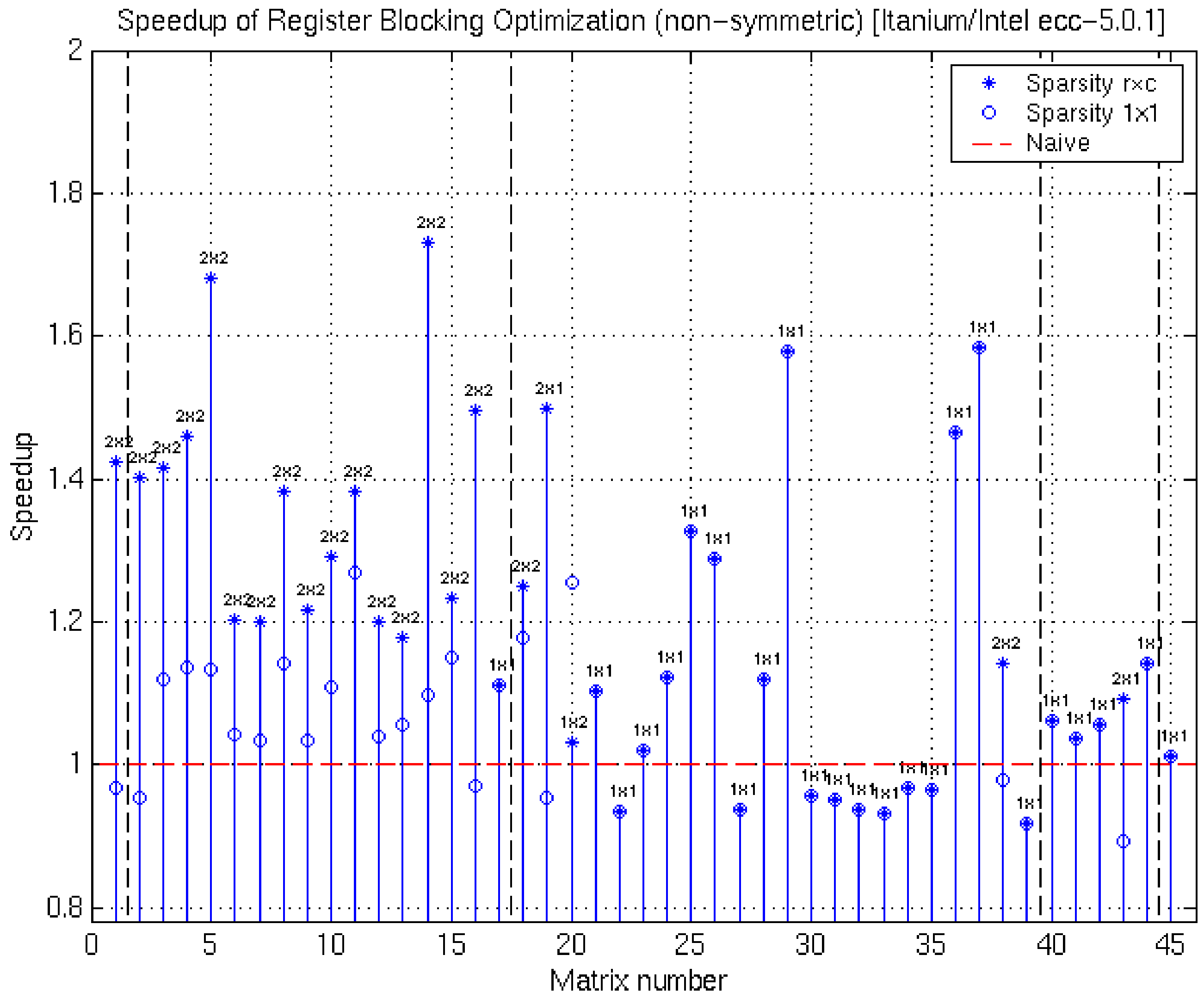


A^T*A on P4 (Accesses A only once)

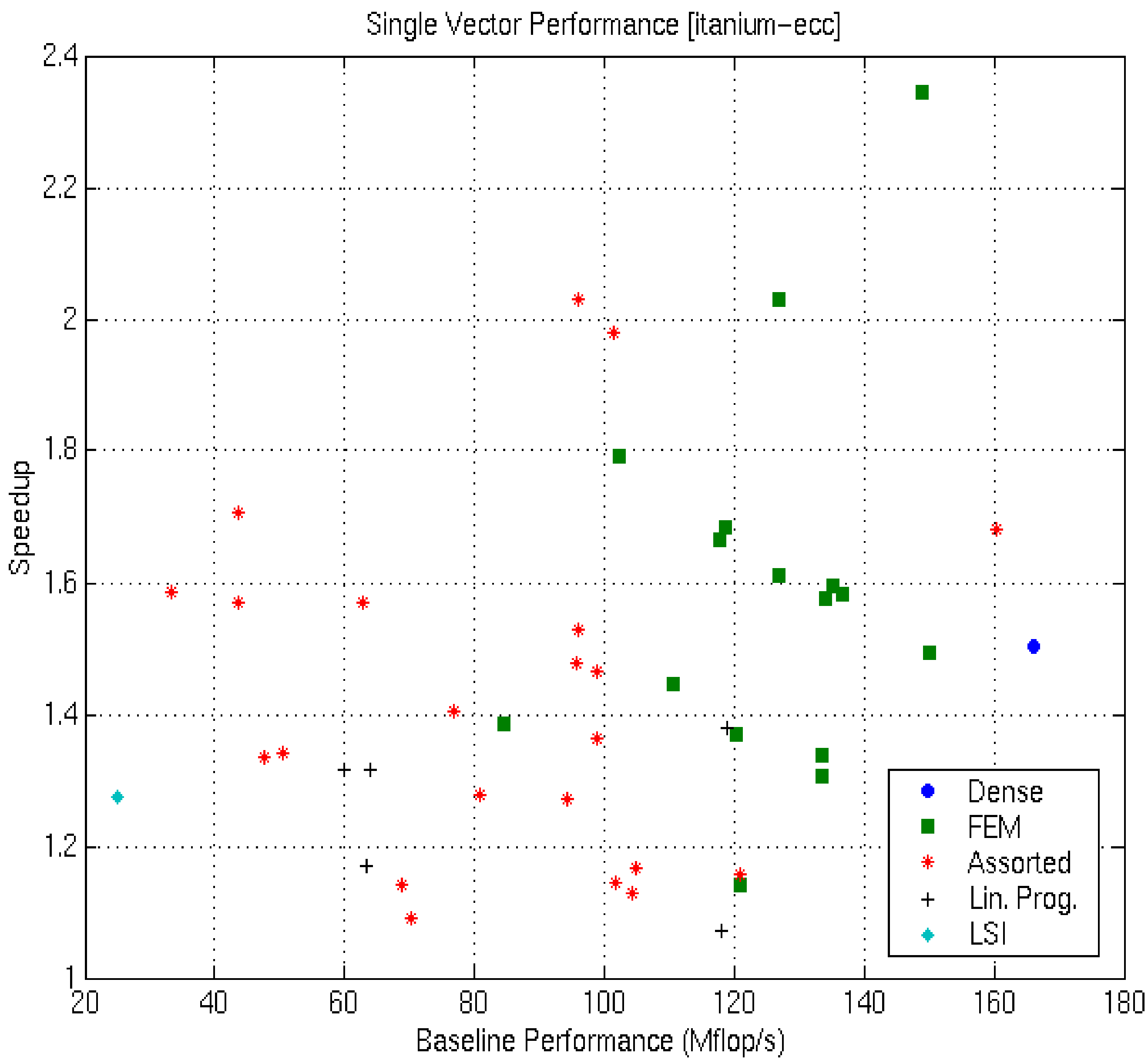


Preliminary Results on Itanium using ecc

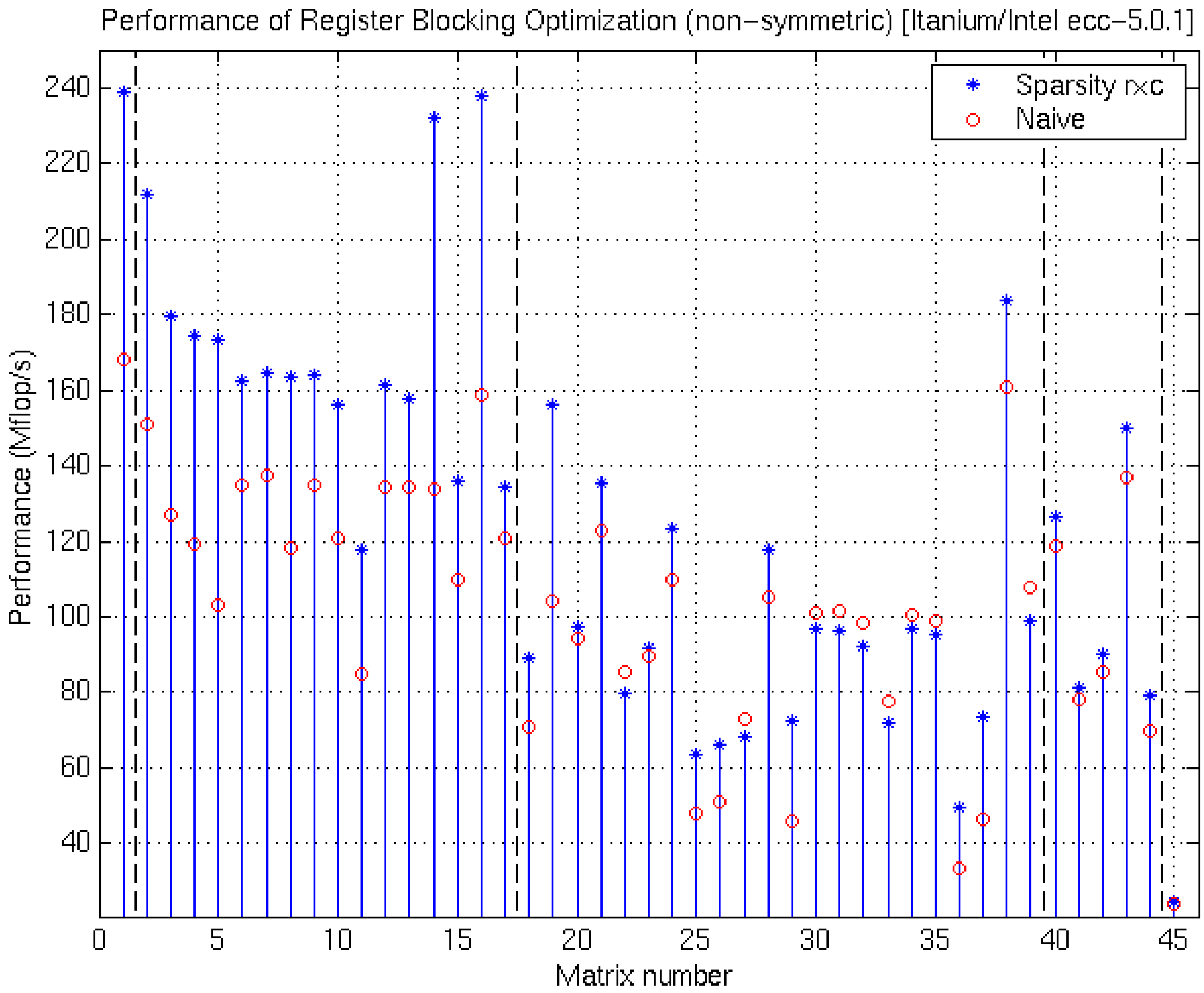
Speedup of SPMV from Sparsity on Itanium/ecc-5.0.1



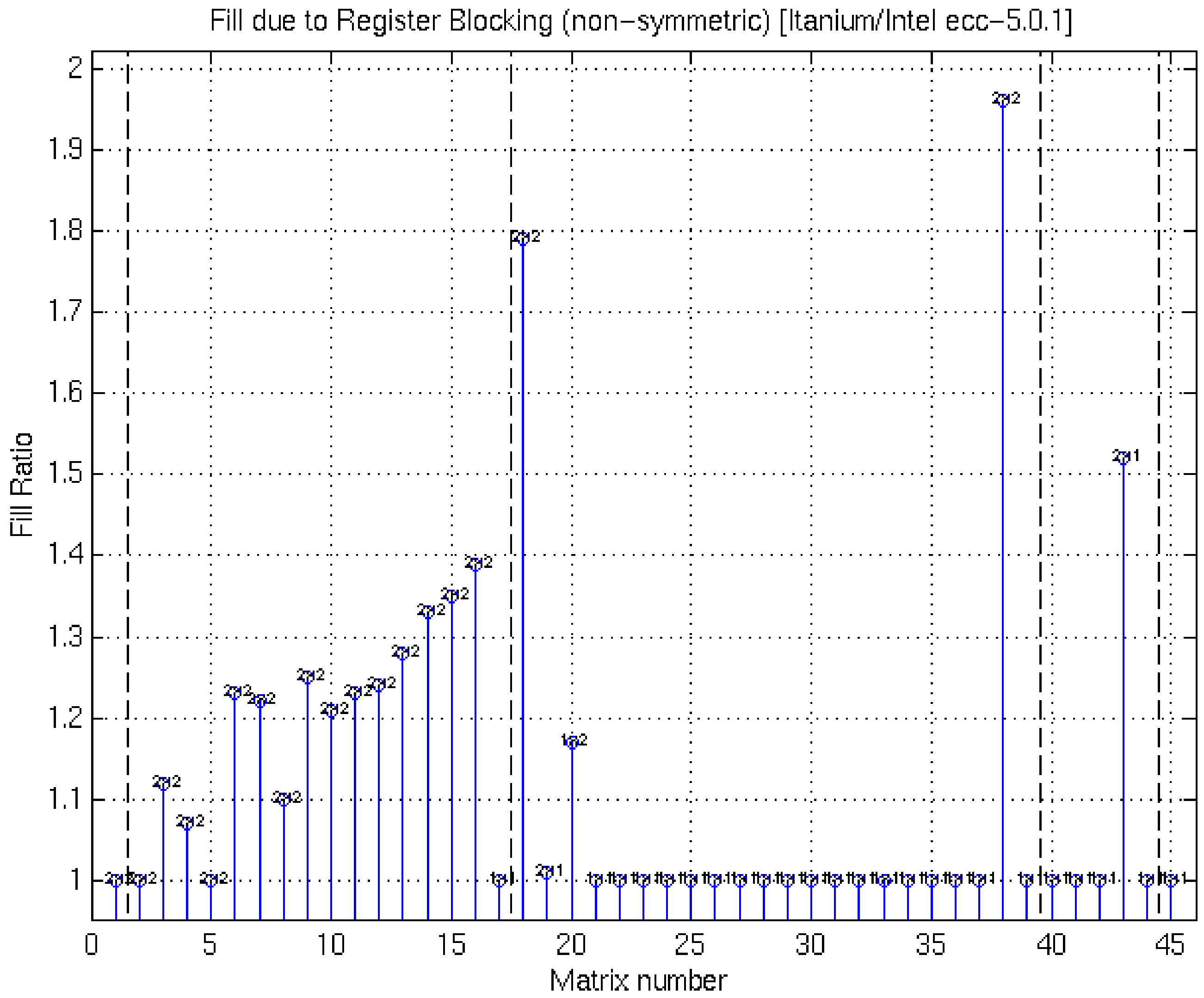
Single vector speedups on Itanium by matrix type



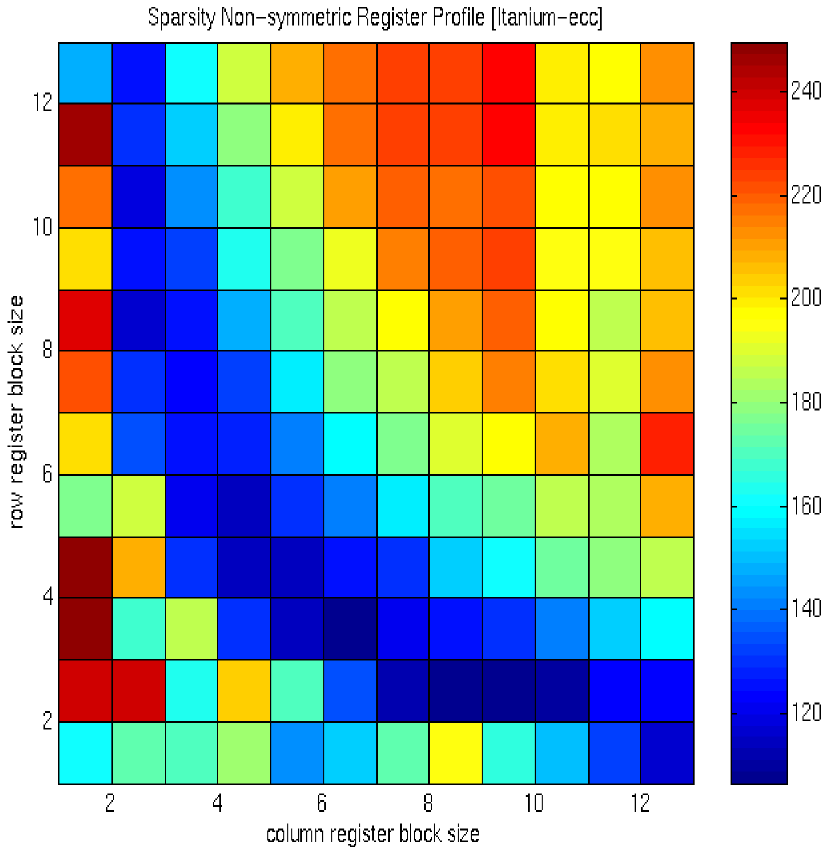
Raw Performance of SPMV from Sparsity on Itanium



Fill for SPMV from Sparsity on Itanium

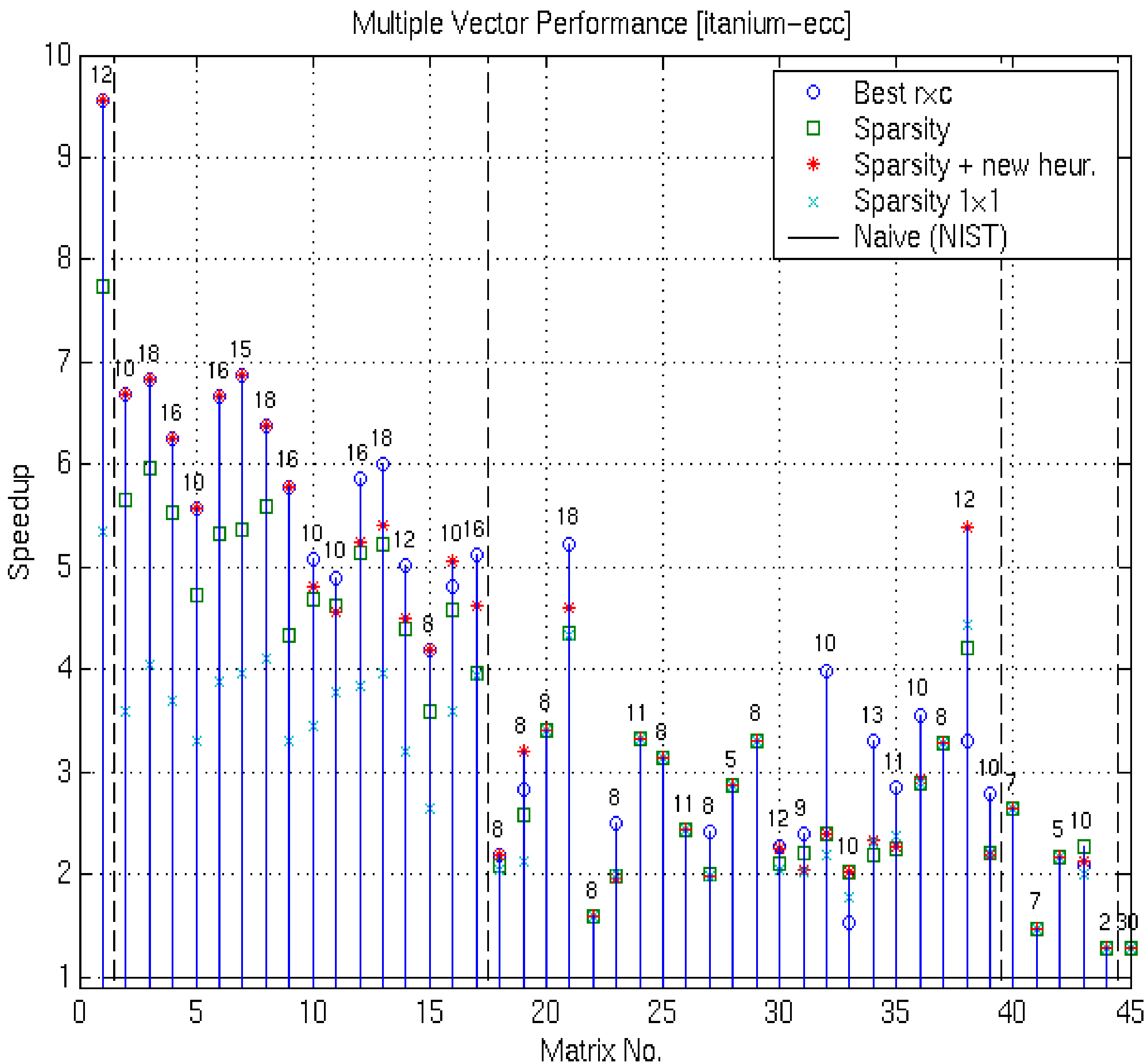


Improvements to register block size selection

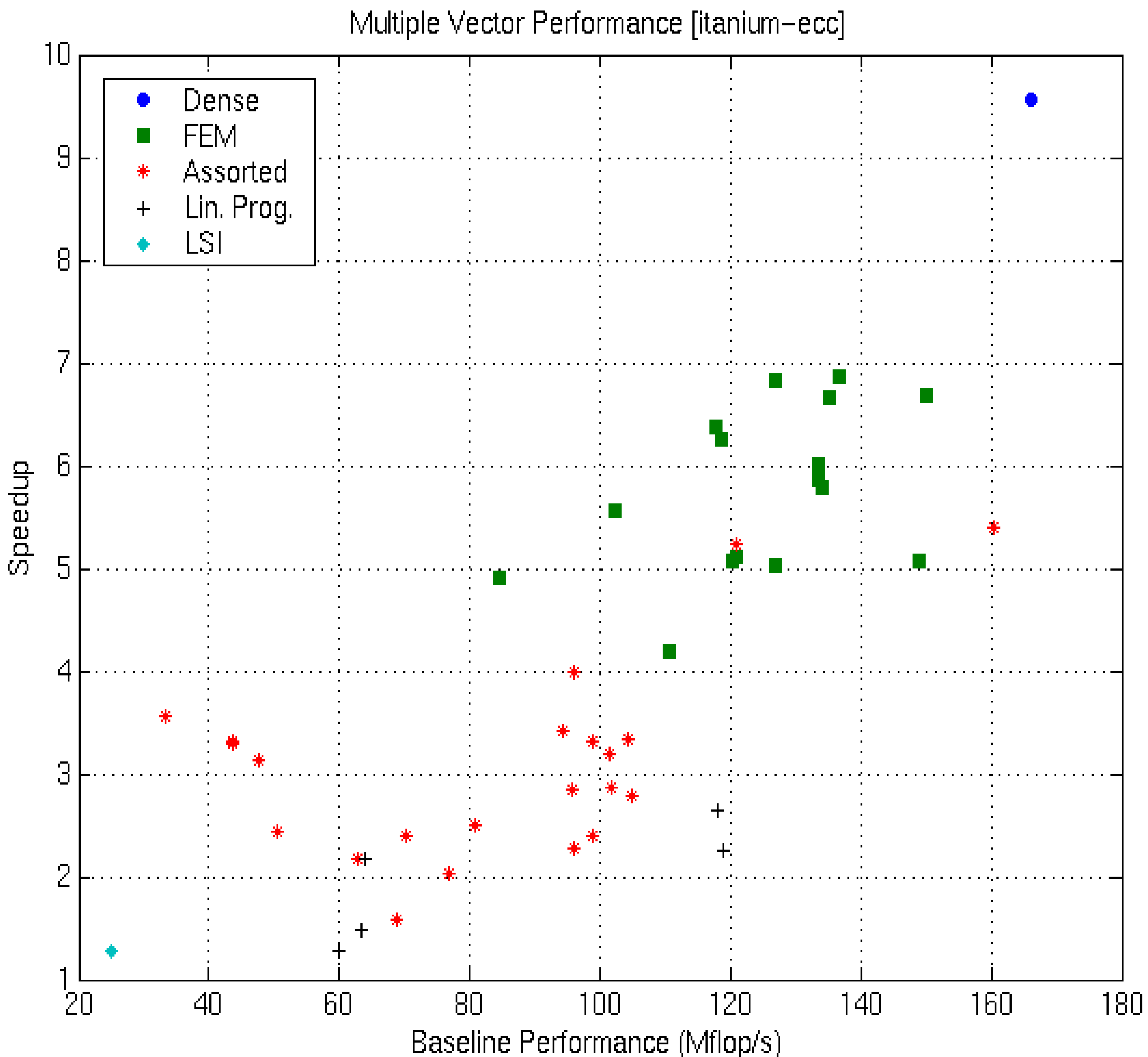


- Initial heuristic to determine best r x c block biased to diagonal of performance plot
- Didn't matter on Sun, does on P4 and Itanium since performance so "nondiagonally dominant"
- Matrix 8:
  - Chose 2x2 (164 Mflops)
  - Better: 3x1 (196 Mflops)
- Matrix 9:
  - Chose 2x2 (164 Mflops)
  - Better: 3x1 (213 Mflops)

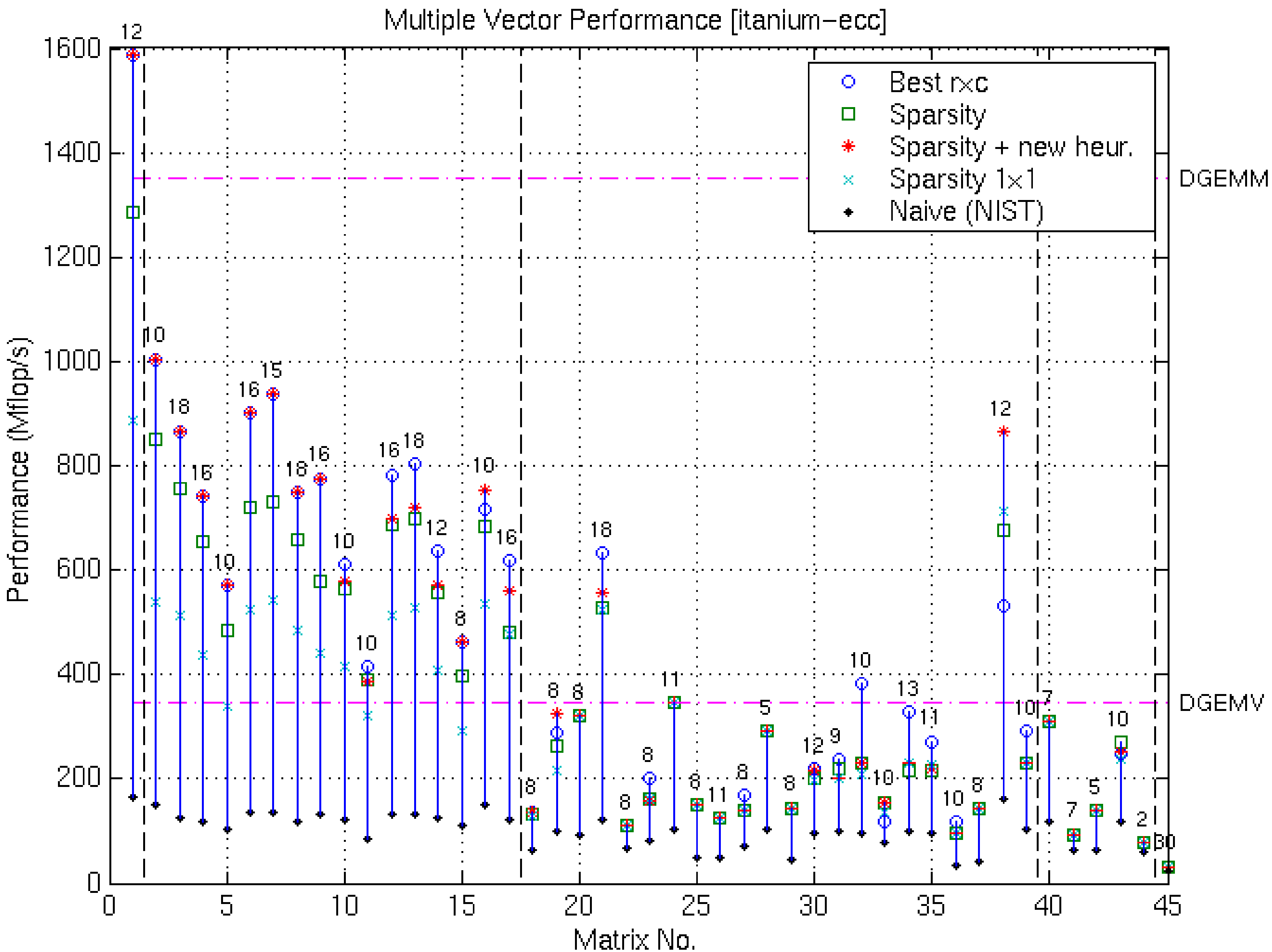
Multiple vector speedups on Itanium



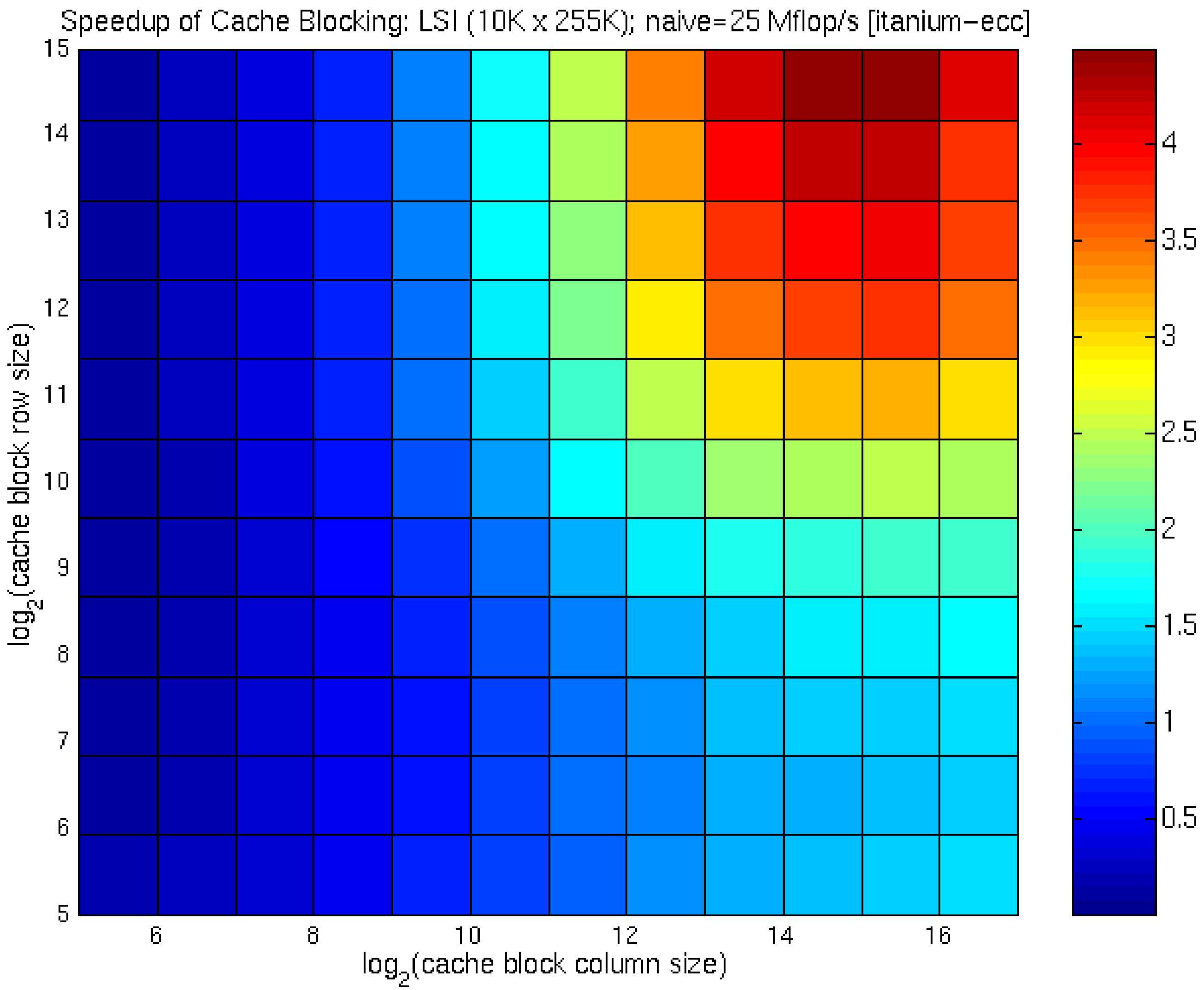
Multiple vector speedups on Itanium – by matrix type



Multiple Vector Performance on Itanium



Speed up from Cache Blocking on LSI matrix on Itanium



Applications of Performance Tuning (non SciDAC)

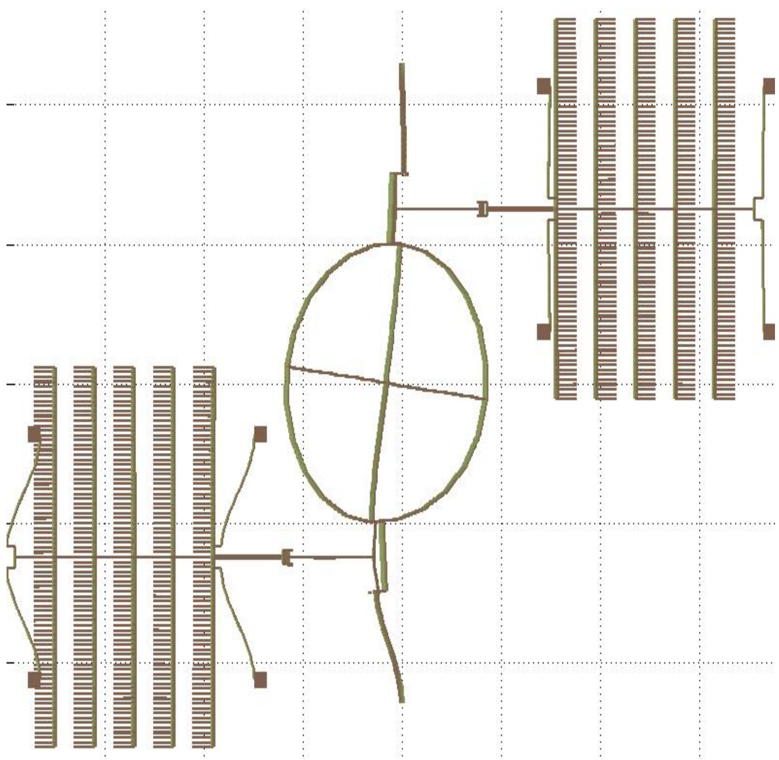
SUGAR – A CAD Tool for MEMS

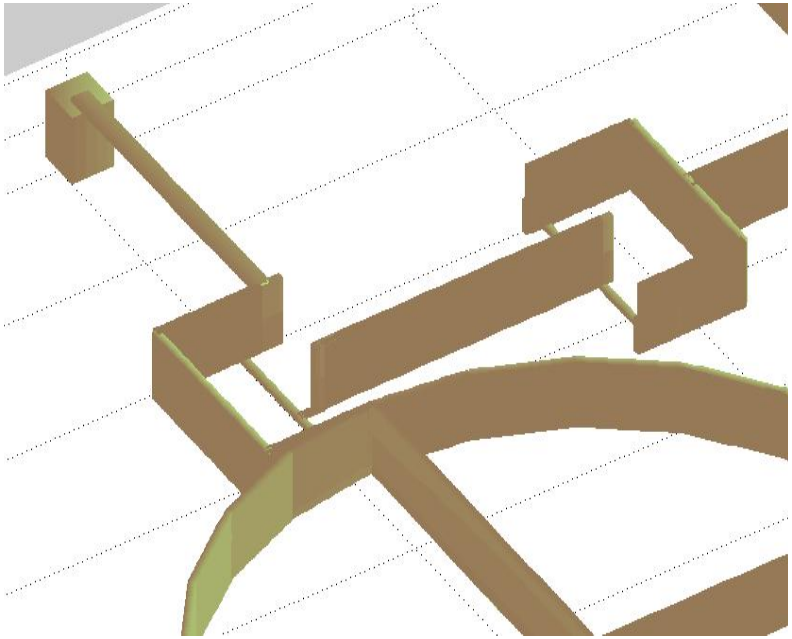
Applications to SUGAR – a tool for MEMS CAD

- Demmel, Bai, Pister, Govindjee, Agogino, Gu, …
- Input: description of MicroElectroMechanical System (as netlist)
- Output:
  - **DC, steady state, modal, transient analyses to assess behavior**
  - **CIF for fabrication**
- Simulation capabilities
  - **Beams and plates (linear, nonlinear, prestressed,…)**
  - **Electrostatic forces, circuits**
  - **Thermal expansion, Couette damping**
- Availability
  - **Matlab**
    - Publicly available
    - www-bsac.eecs.berkeley.edu/~cfm
    - 249 registered users, many unregistered
  - **Web service – M & MEMS**
    - Runs on Millennium
    - sugar.millennium.berkeley.edu
    - Now in use in EE 245 at UCB…96 users
- Lots of new features being added, including interface to measurements

Micromirror (Last, Pister)

- Laterally actuated torsionally suspended micromirror
- Over 10K dof, 100 line netlist (using subnets)
- DC and frequency analysis
- All algorithms reduce to previous kernels





Applications of Performance Tuning

Information Retrieval

Information Retrieval

- Jordan
- Collaboration with Intel team building probabilistic graphical models
- Better alternatives to LSI for document modeling and search
- Latent Dirichlet Allocation (LDA)
  - Model documents as union of themes, each with own word distribution
  - Maximum likelihood fit to find themes in set of documents, classify them
  - Computational bottleneck is solution of enormous linear systems
  - One of largest Millennium users
- Identifying influential documents
  - Given hyperlink patterns of documents, which are most influential?
  - Basis of Google (eigenvector of link matrix → sparse matrix vector multiply)
  - Applying Markov chain and perturbation theory to assess reliability
- Kernel ICA
  - Estimate set of sources s and mixing matrix A from samples x = A*s
  - New way to sample such that sources are as independent as possible
  - Again reduces to linear algebra kernels…

More on Kernel ICA

- Algorithm 1
  - nonlinear eigenvalue problem, reduces to a sequence of many
  - very large generalized spd eigenproblems $A - \lambda B$
    - Block structured, A dense, B block diagonal
    - Only smallest nonzero eigenvalue needed
  - Sparse eigensolver (currently ARPACK/eigs)
    - Use Incomplete Cholesky (IC) to get low rank approximzation to dense subblocks comprising A and B
    - Use Complete (=Diagonal) Pivoting but take only 20 << n steps
    - Cost is O(n)
      - Evaluating matrix entries (exponentials) could be bottleneck
      - Need fast, low precision exponential
- Algorithm 2
  - Like Algorithm 1, but find all eigenvalues/vectors of $A - \lambda B$
  - Use Holy Grail

Future Work

- SciDAC
  - Evaluate on SciDAC applications
  - Determine interfaces for integration into SciDAC applications
- Further exploit Itanium, other architectures
  - 128 (82-bit) floating point registers
    - 9 HW formats: 24/8(v), 24/15, 24/17, 53/11, 53/15, 53/17, 64/15, 64/17
    - Many fewer load/store instructions
  - fused multiply-add instruction
  - predicated instructions
  - rotating registers for software pipelining
  - prefetch instructions
  - three levels of cache
- Tune current and wider set of kernels
  - Improve heuristics, eg choice of r x c
- Further automate performance tuning (NSF)
  - Generation of algorithm space generators

Background on Test Matrices

Sparse Matrix Benchmark Suite (1/3)

#	Matrix Name	Problem Domain	Dimension	No. Non-zeros
1	dense	Dense matrix	1,000	1.00 M
2	raefsky3	Fluid structure interaction	21,200	1.49 M
3	inaccura	Accuracy problem	16,146	1.02 M
4	bcsstk35*	Stiff matrix automobile frame	30,237	1.45 M
5	venkat01	Flow simulation	62,424	1.72 M
6	crystk02*	FEM crystal free-vibration	13,965	969 k
7	crystk03*	FEM crystal free-vibration	24,696	1.75 M
8	nasasrb*	Shuttle rocket booster	54,870	2.68 M
9	3dtube*	3-D pressure tube	45,330	3.21 M
10	ct20stif*	CT20 engine block	52,329	2.70 M
11	bai	Airfoil eigenvalue calculation	23,560	484 k
12	raefsky4	Buckling problem	19,779	1.33 M
13	ex11	3-D steady flow problem	16,214	1.10 M
14	rdist1	Chemical process simulation	4,134	94.4 k
15	vavasis3	2-D PDE problem	41,092	1.68 M

Note: * indicates a symmetric matrix.

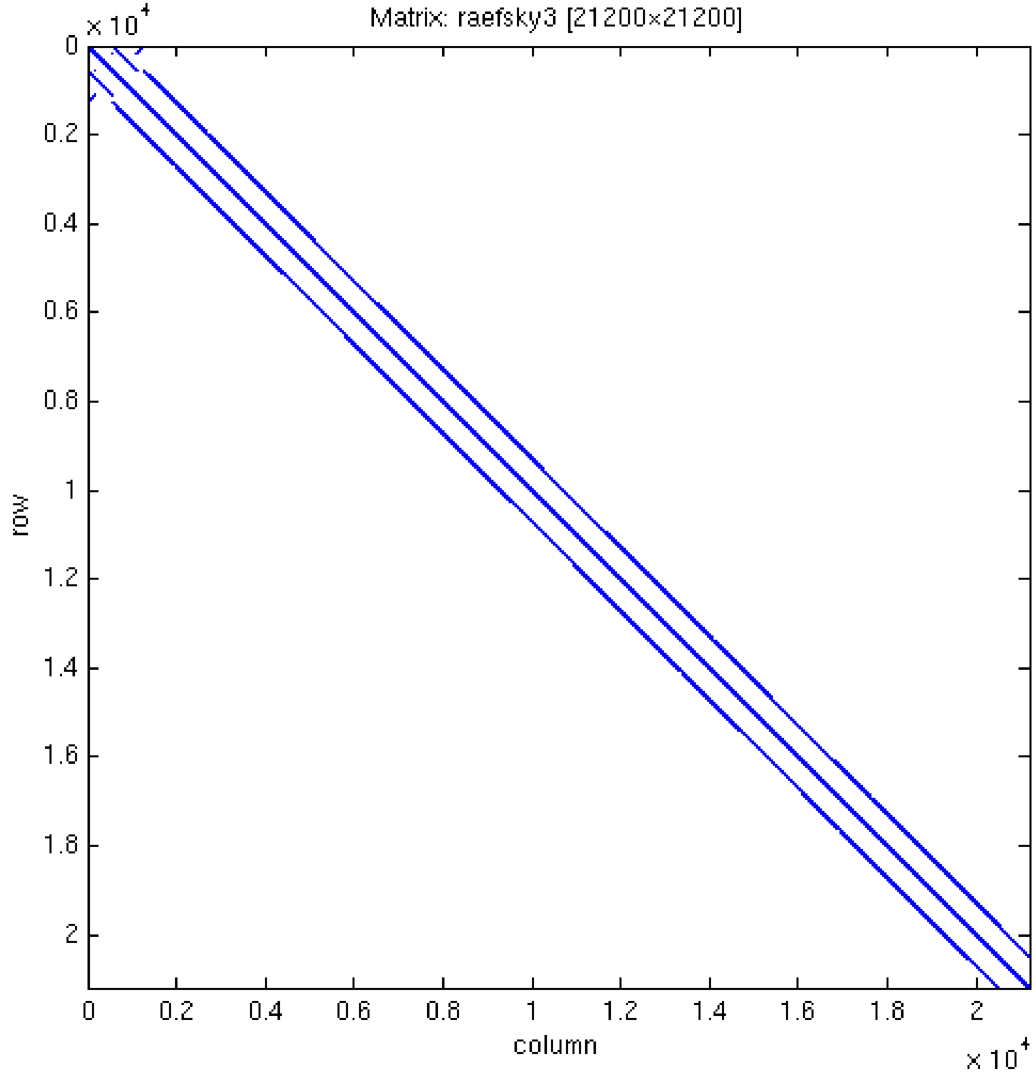
Sparse Matrix Benchmark Suite (2/3)

#	Matrix Name	Problem Domain	Dimension	No. Non-zeros
16	orani678	Economic modeling	2,529	90.2 k
17	rim	FEM fluid mechanics problem	22,560	1.01 M
18	memplus	Circuit simulation	17,758	126 k
19	gemat11	Power flow	4,929	33.1 k
20	lhr10	Chemical process simulation	10,672	233 k
21	goodwin*	Fluid mechanics problem	7,320	325 k
22	bayer02	Chemical process simulation	13,935	63.7 k
23	bayer10	Chemical process simulation	13,436	94.9 k
24	coater2	Simulation of coating flows	9,540	207 k
25	finan512*	Financial portfolio optimization	74,752	597 k
26	onetone2	Harmonic balance method	36,057	228 k
27	pwt*	Structural engineering	36,519	326 k
28	vibrobox*	Vibroacoustics	12,328	343 k
29	wang4	Semiconductor device simulation	26,068	177 k
30	lnsp3937	Fluid flow modeling	3,937	25.4 k

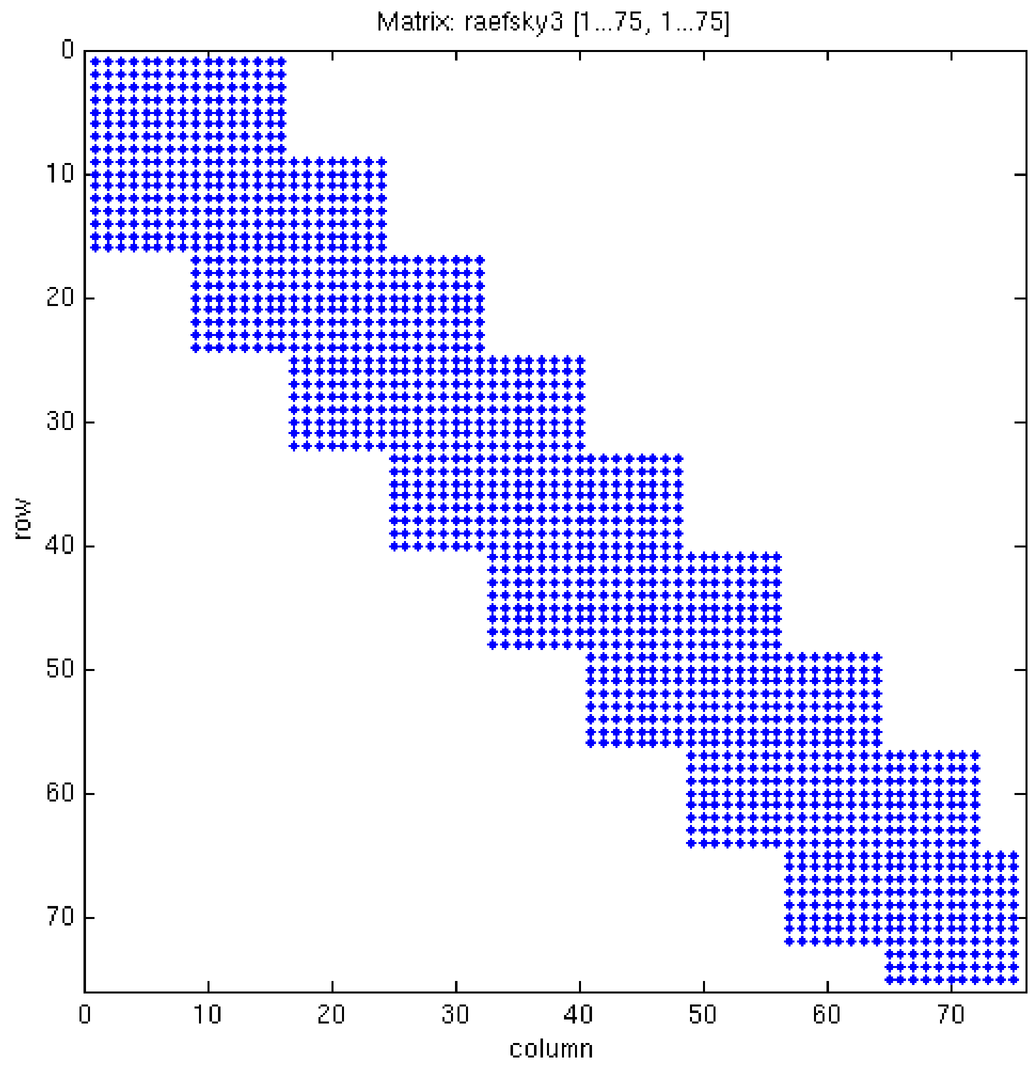
Sparse Matrix Benchmark Suite (3/3)

#	Matrix Name	Problem Domain	Dimensions	No. Non-zeros
31	lns3937	Fluid flow modeling	3,937	25.4 k
32	sherman5	Oil reservoir modeling	3,312	20.8 k
33	sherman3	Oil reservoir modeling	5,005	20.0 k
34	orsreg1	Oil reservoir modeling	2,205	14.1 k
35	saylr4	Oil reservoir modeling	3,564	22.3 k
36	shyy161	Viscous flow calculation	76,480	330 k
37	wang3	Semiconductor device simulation	26,064	177 k
38	mcfe	Astrophysics	765	24.4 k
39	jpwh991	Circuit physics problem	991	6,027
40	gupta1*	Linear programming	31,802	2.16 M
41	lpcreb	Linear programming	9,648 x 77,137	261 k
42	lpcred	Linear programming	8,926 x 73,948	247 k
43	lpfit2p	Linear programming	3,000 x 13,525	50.3 k
44	lpnug20	Linear programming	15,240 x 72,600	305 k
45	lsi	Latent semantic indexing	10 k x 255 k	3.7 M

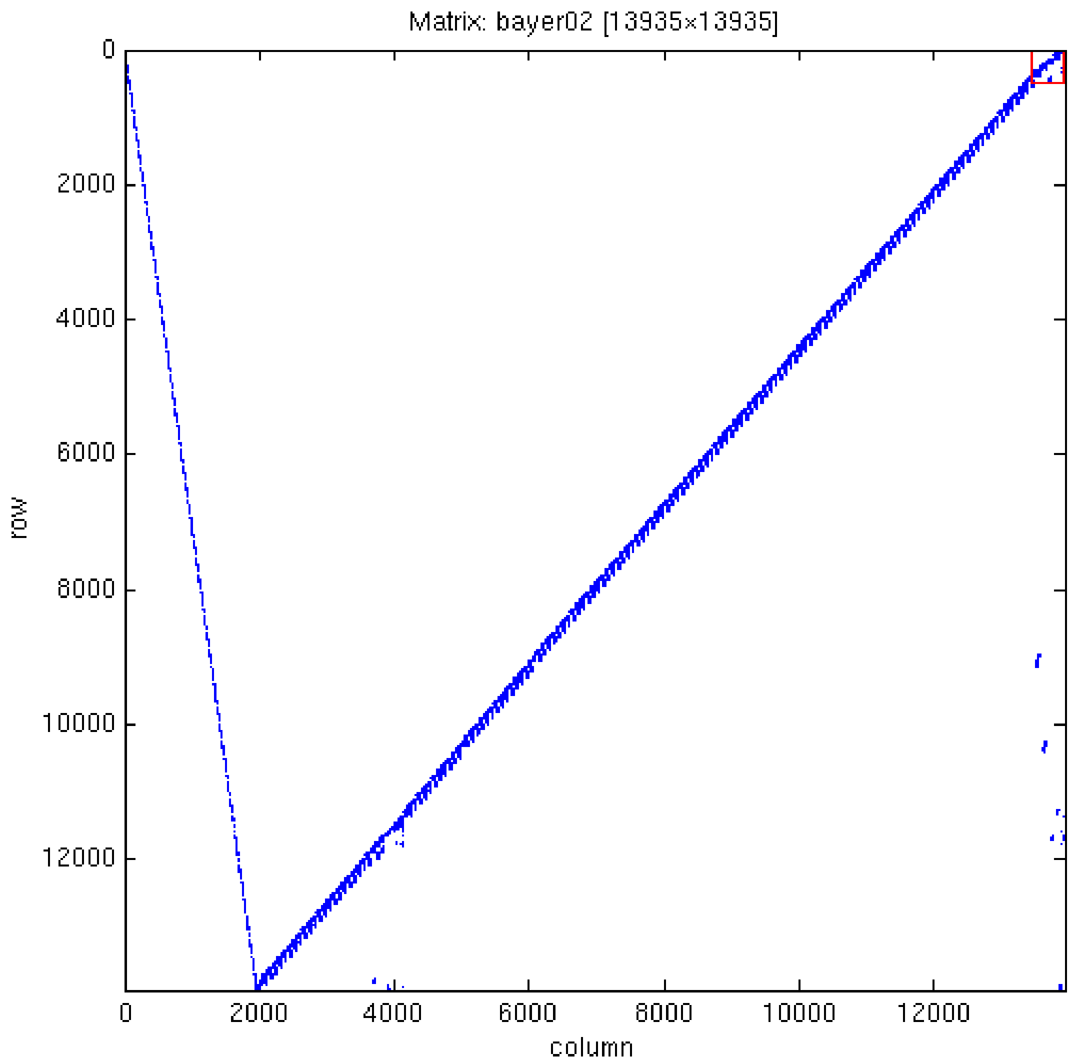
Matrix #2 - raefsky3 (FEM/Fluids)



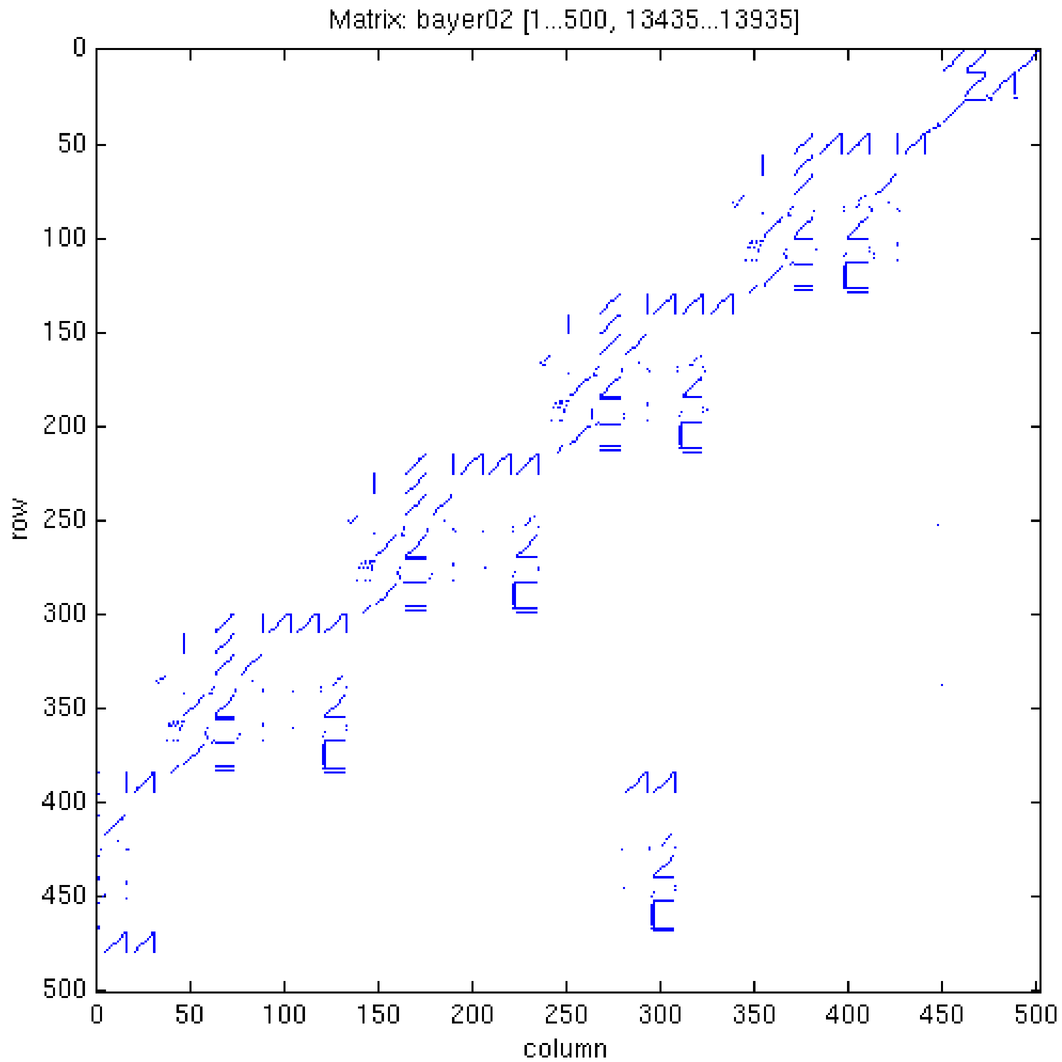
Matrix #2 (cont'd) – raefsky3 (FEM/Fluids)



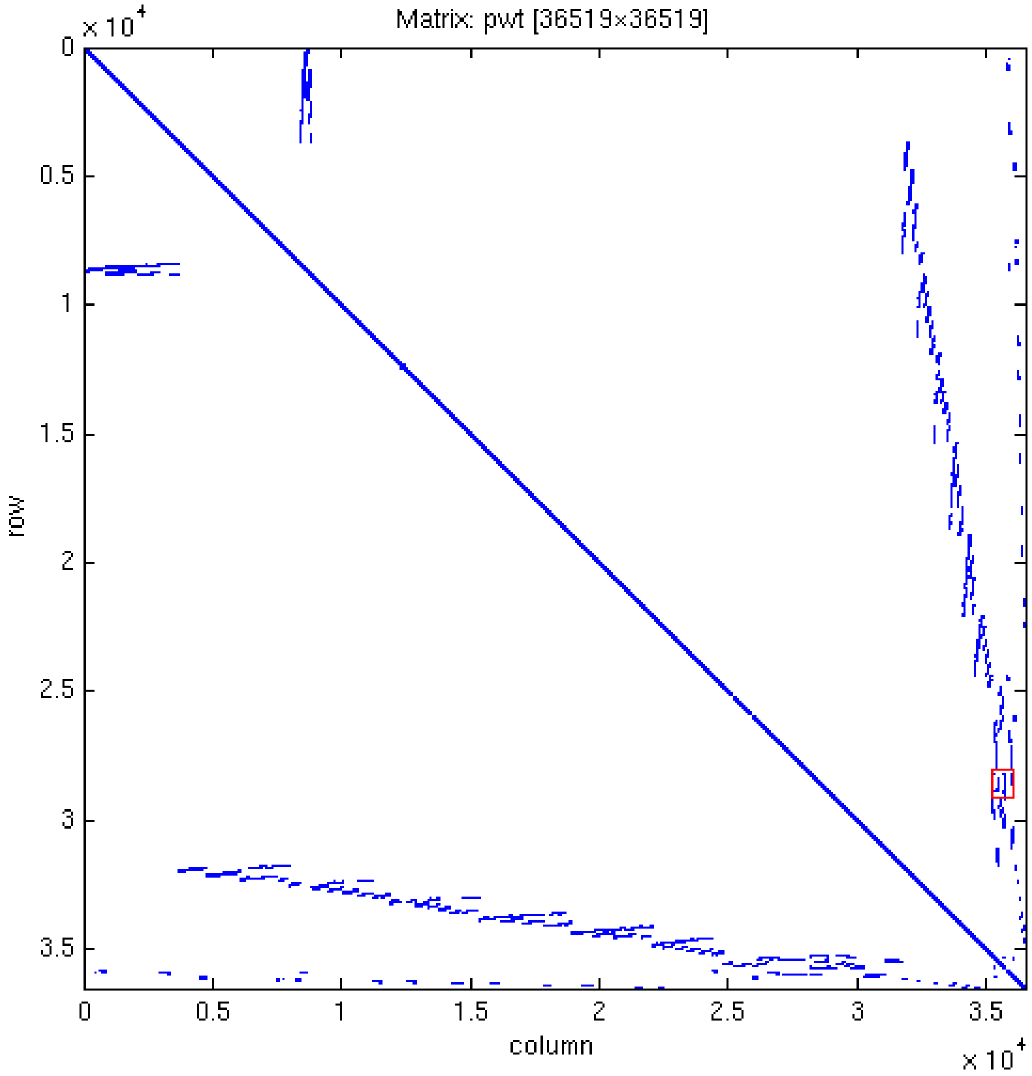
Matrix #22 - bayer02 (chemical process simulation)



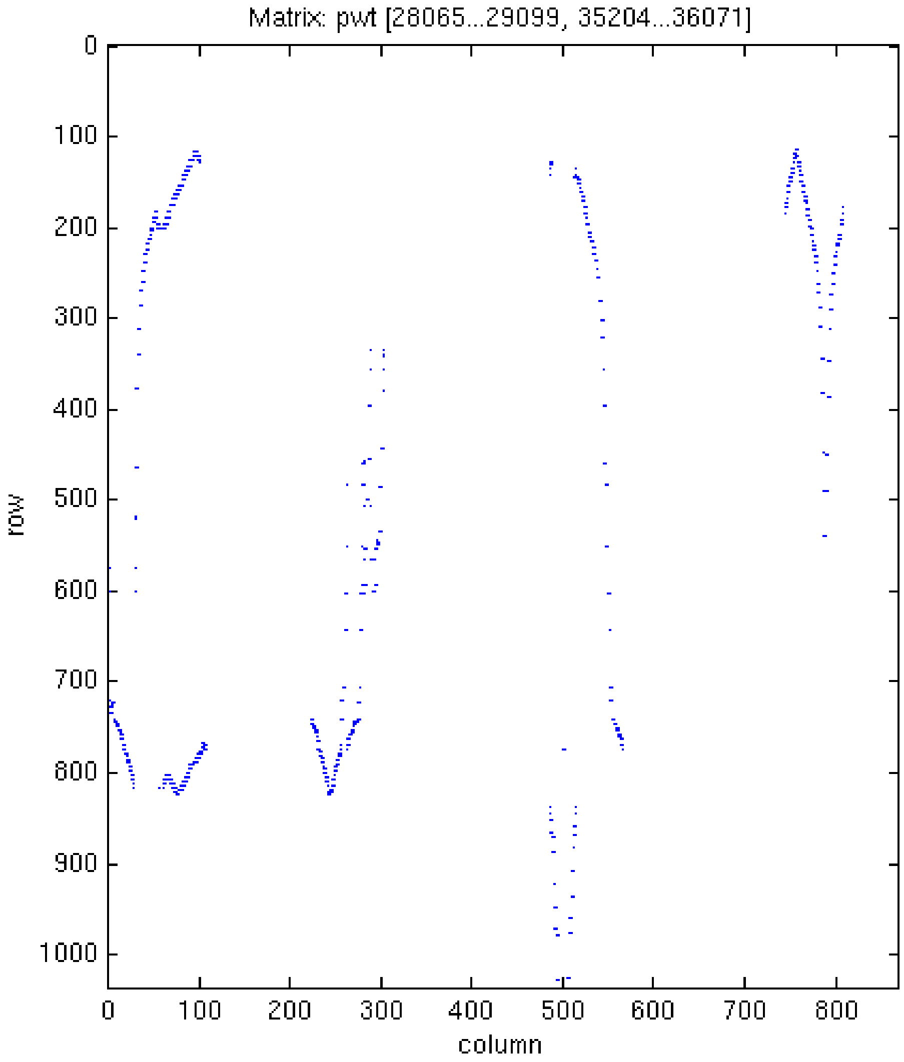
Matrix #22 (cont'd)- bayer02 (chemical process simulation)



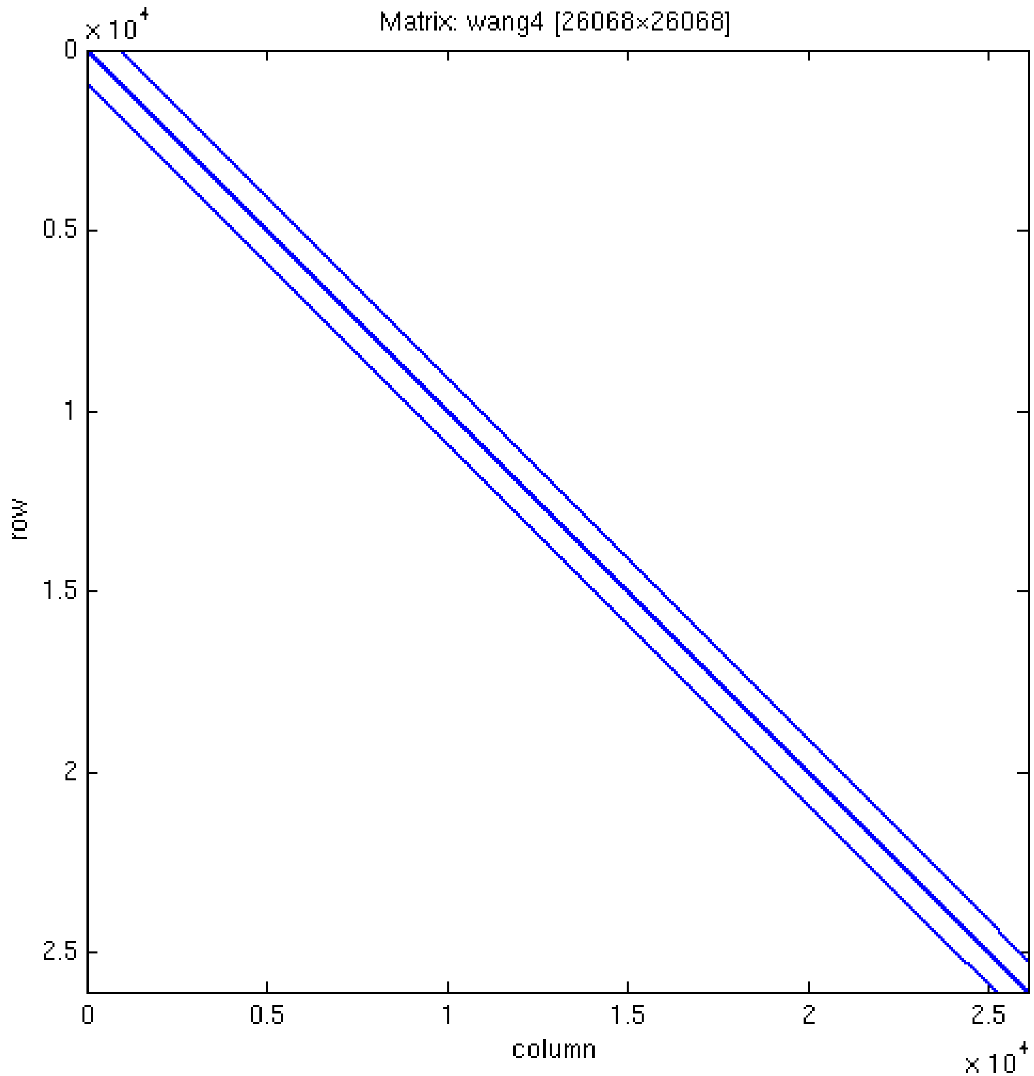
Matrix #27 - pwt (structural engineering)



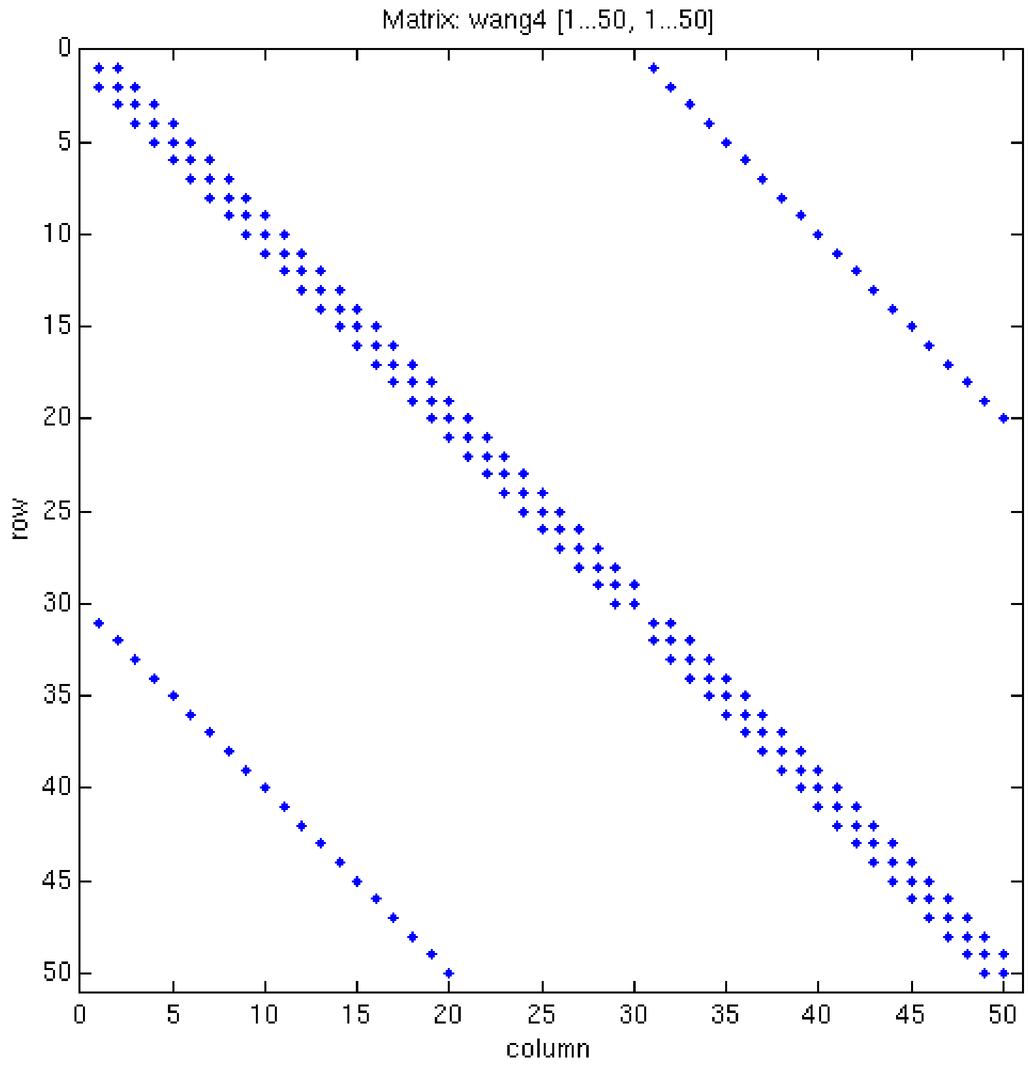
Matrix #27 (cont'd)- pwt (structural engineering)



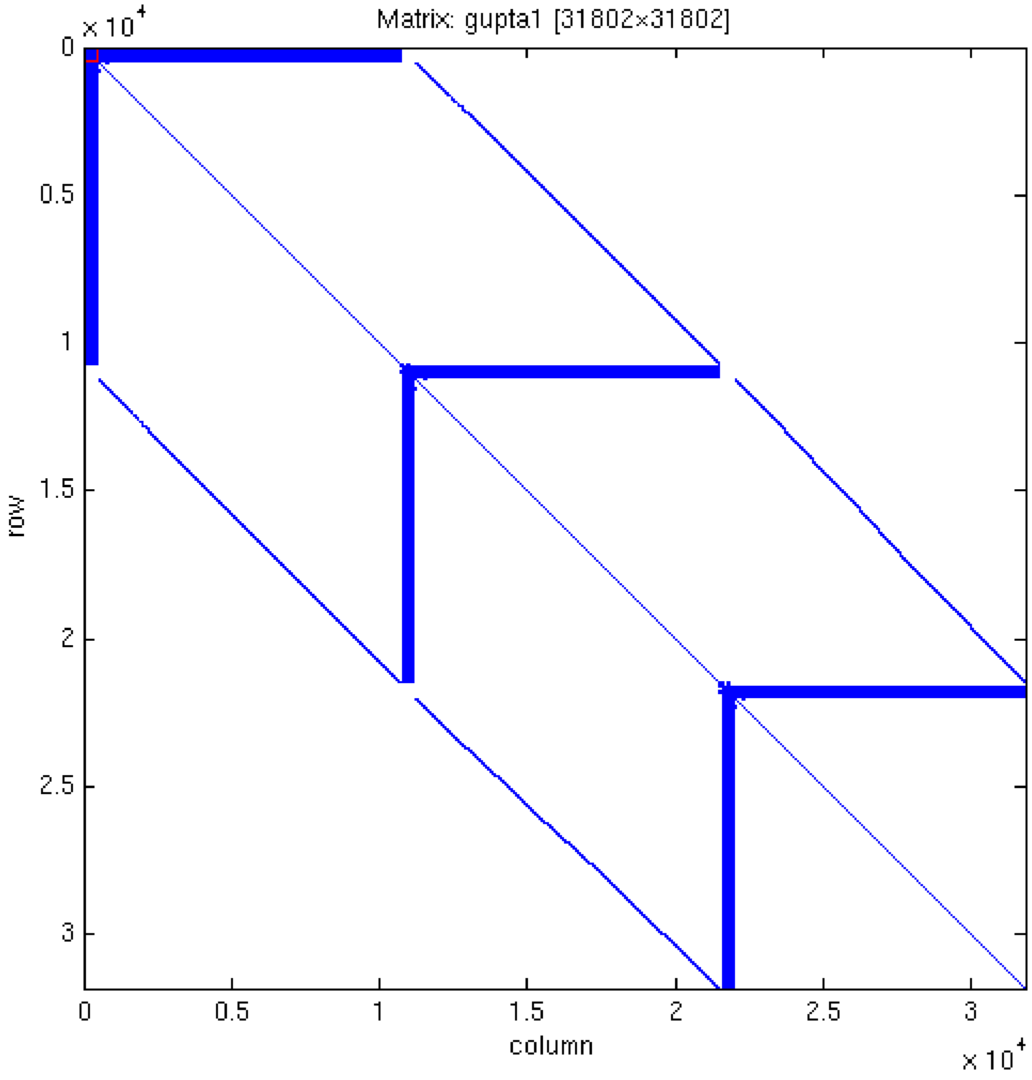
Matrix #29 – wang4 (semiconductor device simulation)



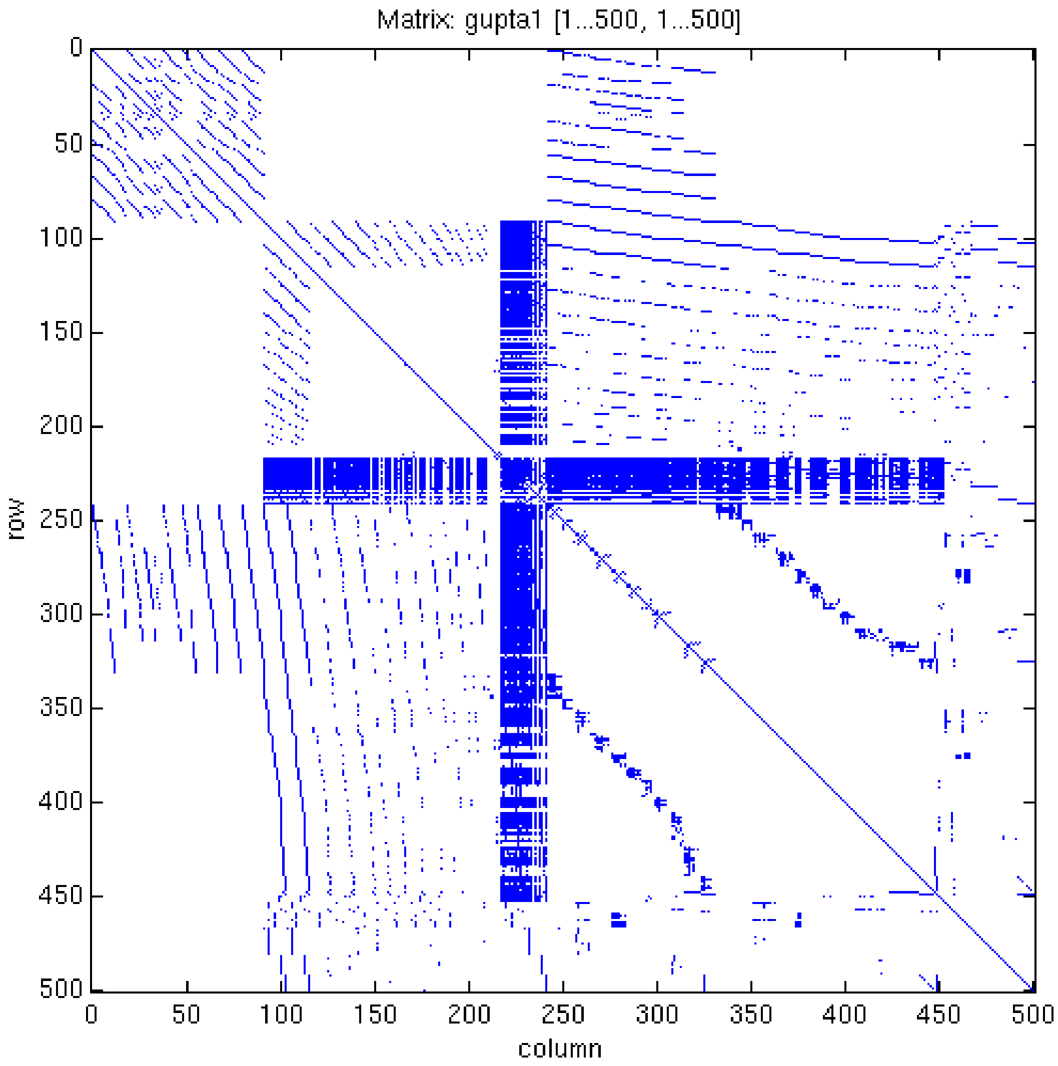
Matrix #29 (cont'd)-wang4 (seminconductor device sim.)



Matrix #40 – gupta1 (linear programming)



Matrix #40 (cont'd) – gupta1 (linear programming)



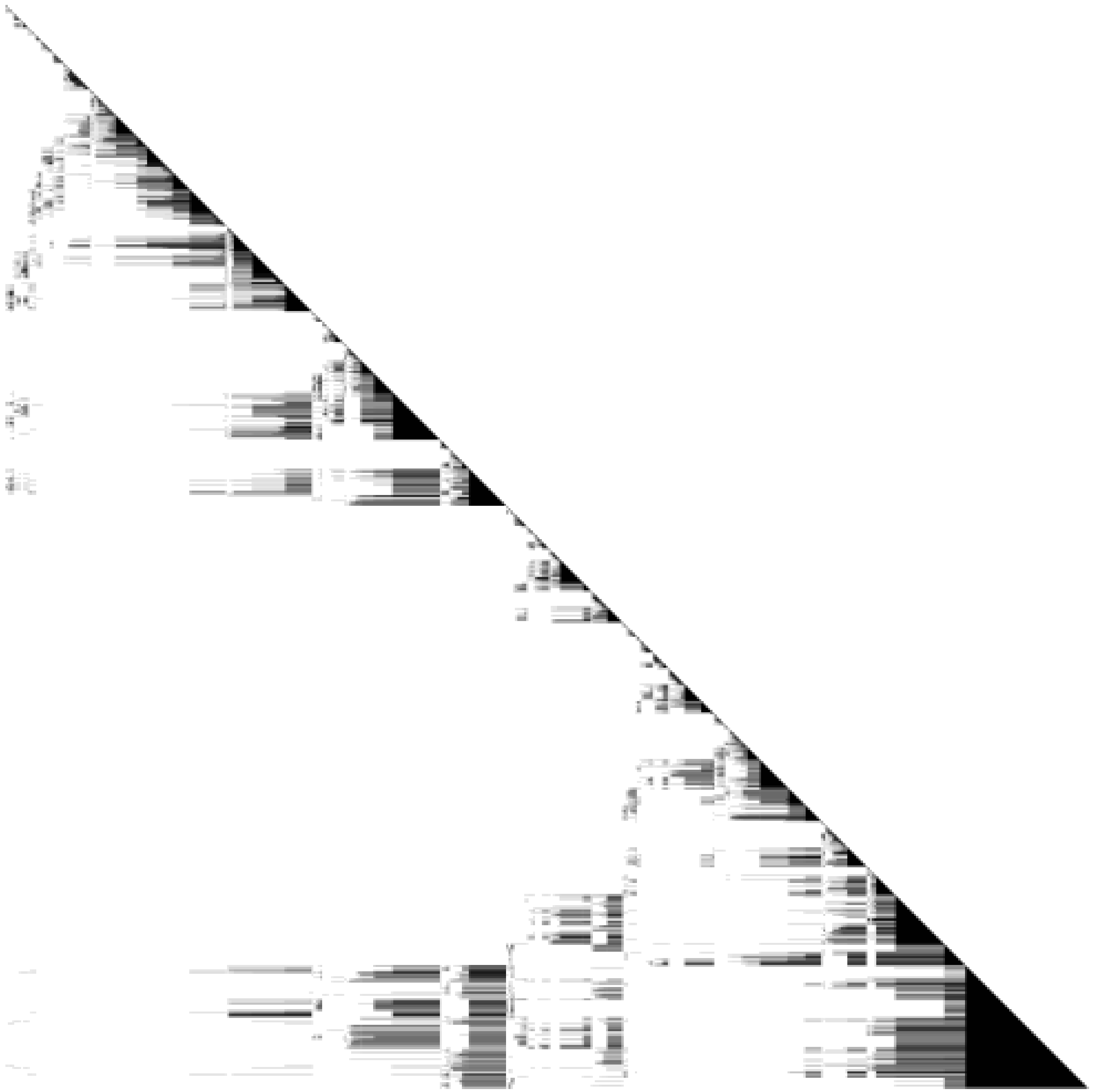
Symmetric Matrix Benchmark Suite

#	Matrix Name	Problem Domain	Dimension	No. Non-zeros
1	dense	Dense matrix	1,000	1.00 M
2	bcsstk35	Stiff matrix automobile frame	30,237	1.45 M
3	crystk02	FEM crystal free vibration	13,965	969 k
4	crystk03	FEM crystal free vibration	24,696	1.75 M
5	nasasrb	Shuttle rocket booster	54,870	2.68 M
6	3dtube	3-D pressure tube	45,330	3.21 M
7	ct20stif	CT20 engine block	52,329	2.70 M
8	gearbox	Aircraft flap actuator	153,746	9.08 M
9	cfd2	Pressure matrix	123,440	3.09 M
10	finan512	Financial portfolio optimization	74,752	596 k
11	pwt	Structural engineering	36,519	326 k
12	vibrobox	Vibroacoustic problem	12,328	343 k
13	gupta1	Linear programming	31,802	2.16 M

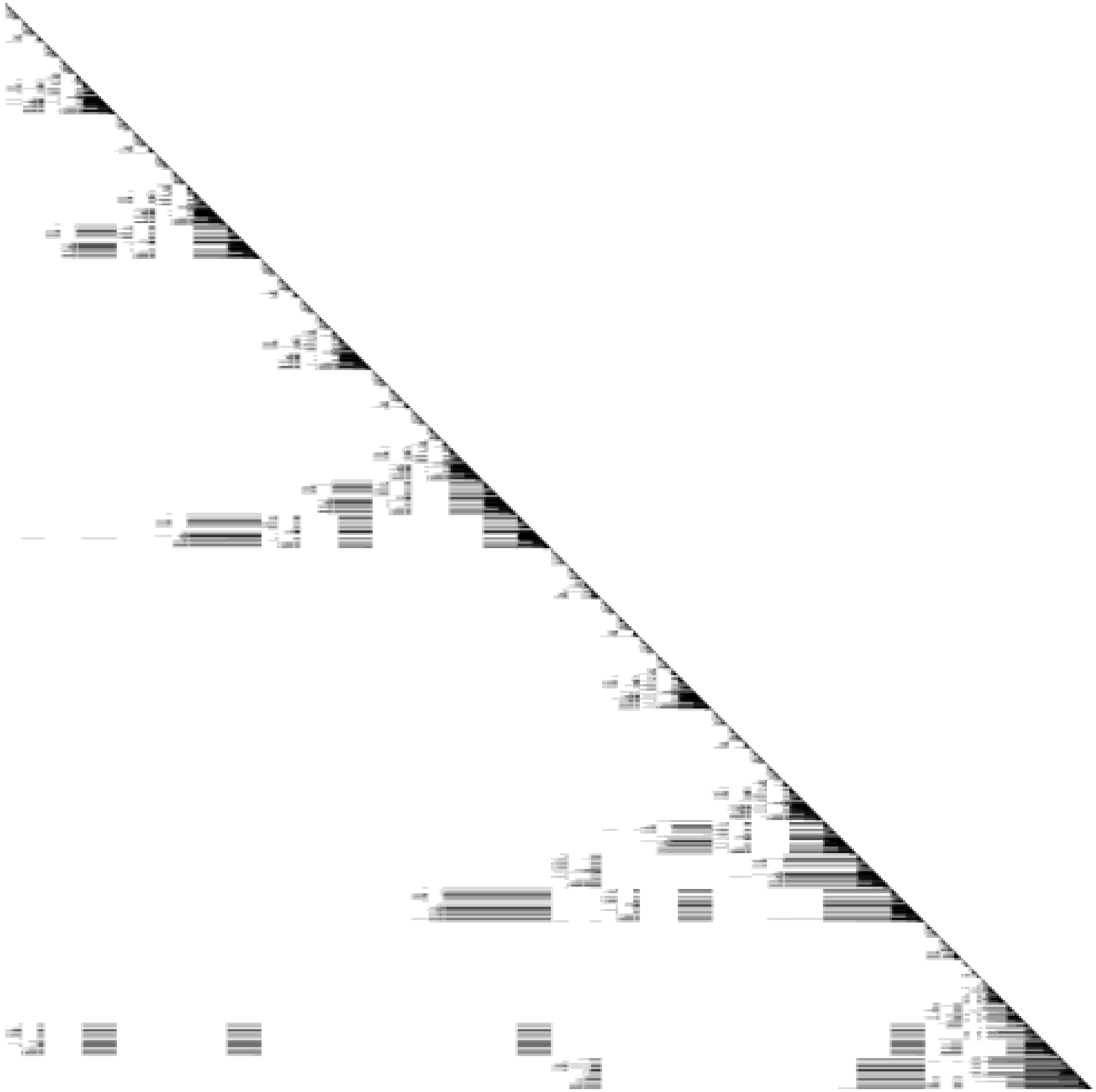
Lower Triangular Matrix Benchmark Suite

#	Matrix Name	Problem Domain	Dimension	No. of non-zeros
1	dense	Dense matrix	1,000	500 k
2	ex11	3-D Fluid Flow	16,214	9.8 M
3	goodwin	Fluid Mechanics, FEM	7,320	984 k
4	lhr10	Chemical process simulation	10,672	369 k
5	memplus	Memory circuit simulation	17,758	2.0 M
6	orani678	Finance	2,529	134 k
7	raefsky4	Structural modeling	19,779	12.6 M
8	wang4	Semiconductor device simulation, FEM	26,068	15.1 M

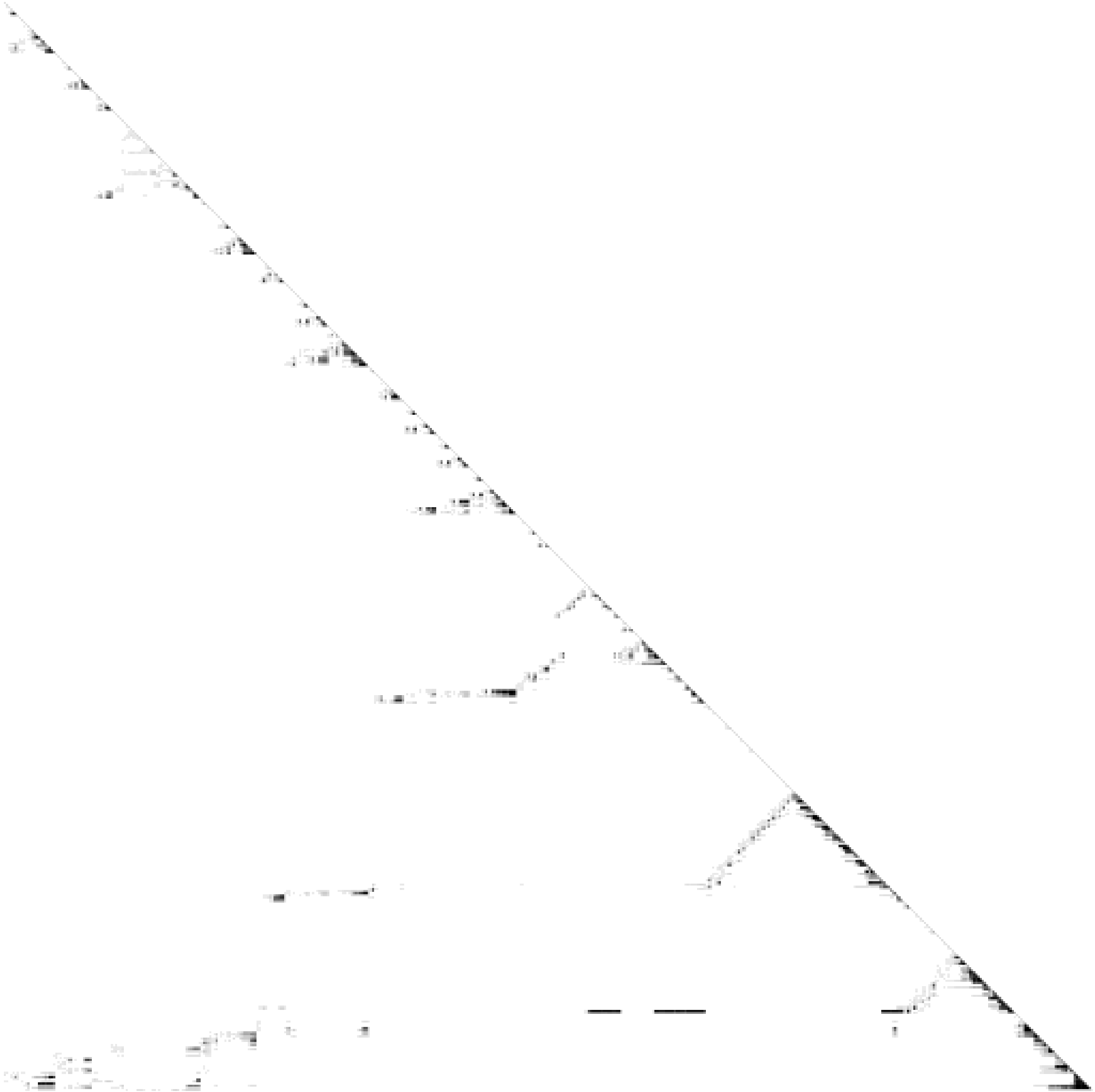
Lower triangular factor: Matrix #2 – ex11



Lower triangular factor: Matrix #3 - goodwin

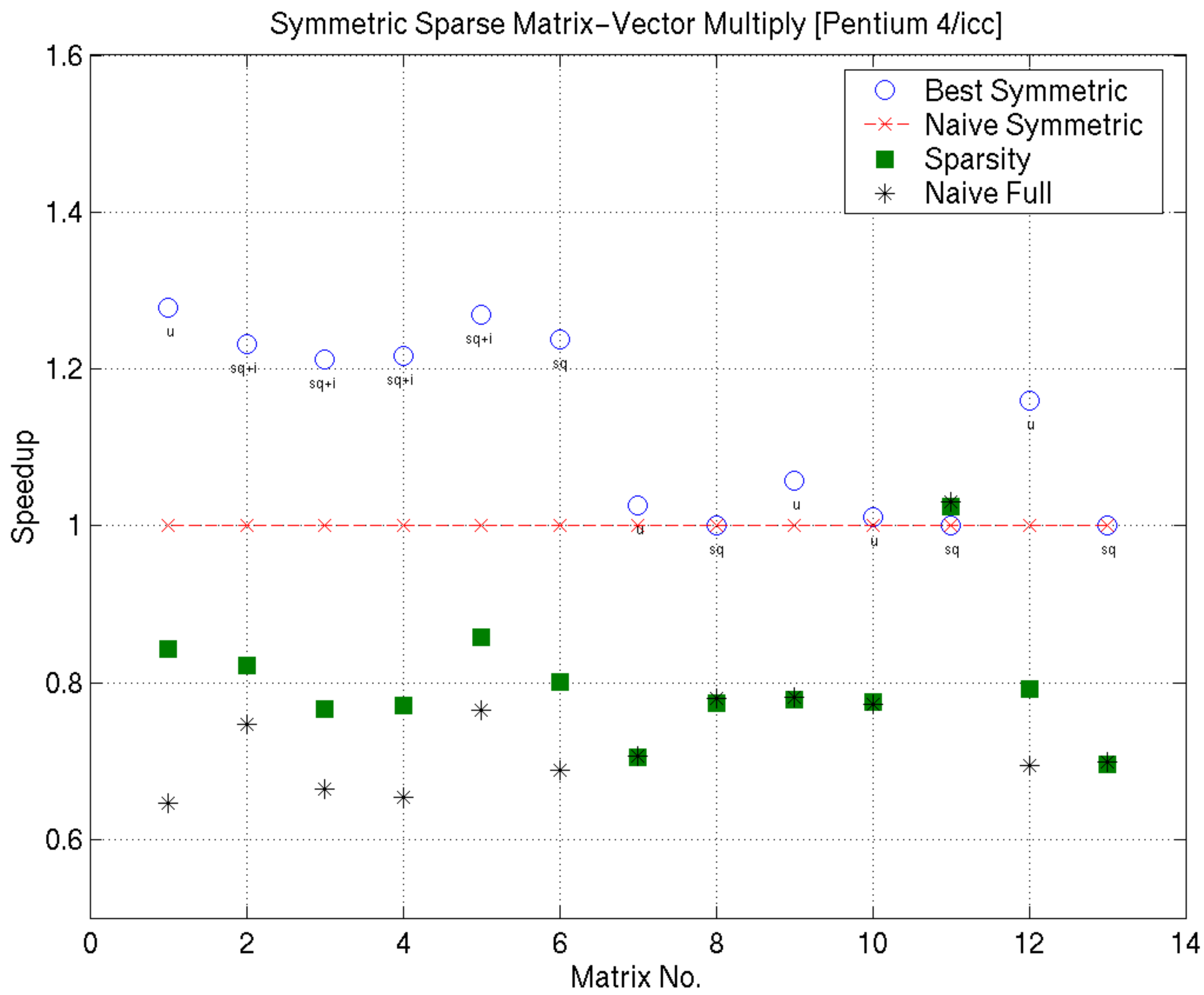


Lower triangular factor: Matrix #4 – lhr10

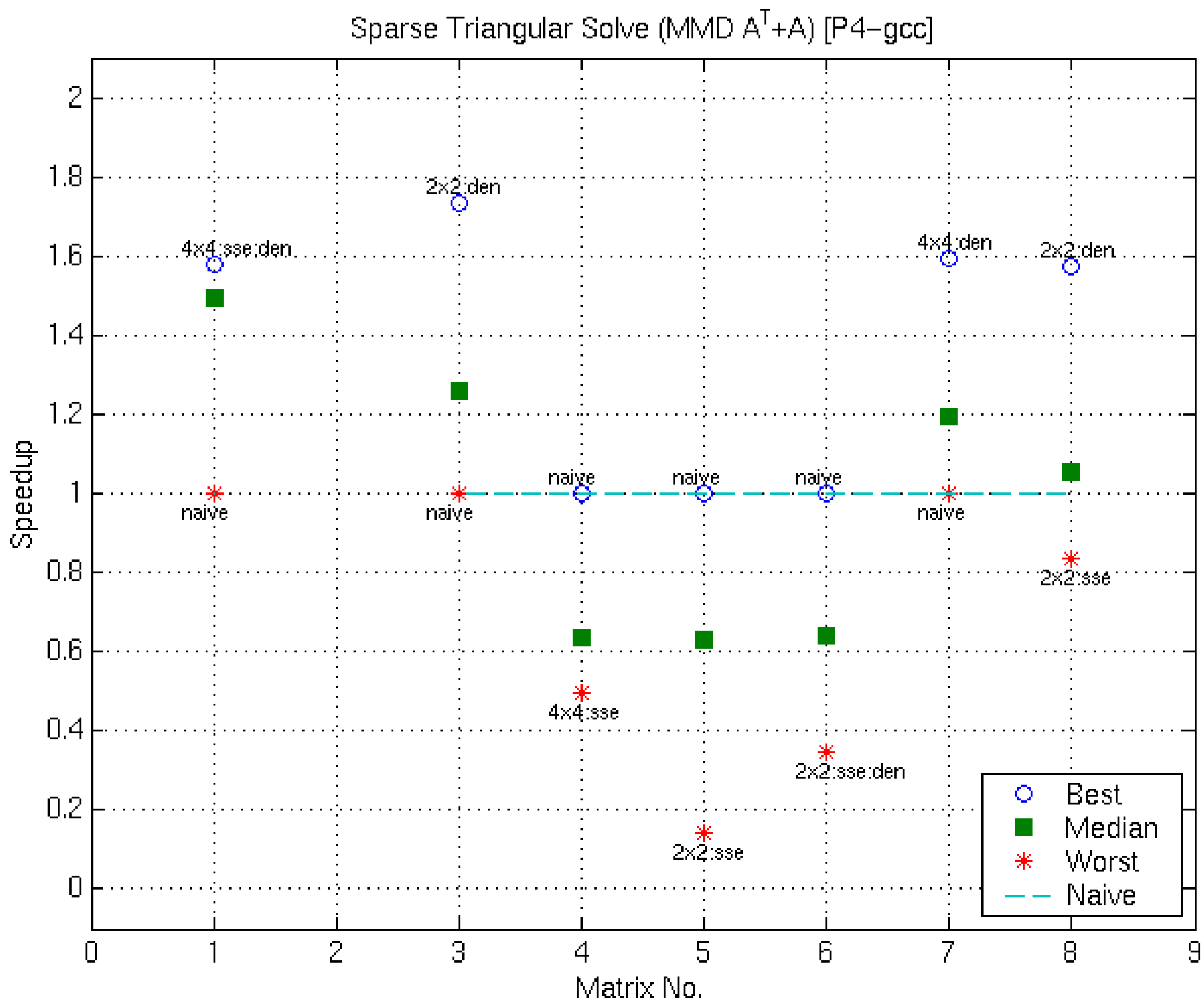


Extra Slides

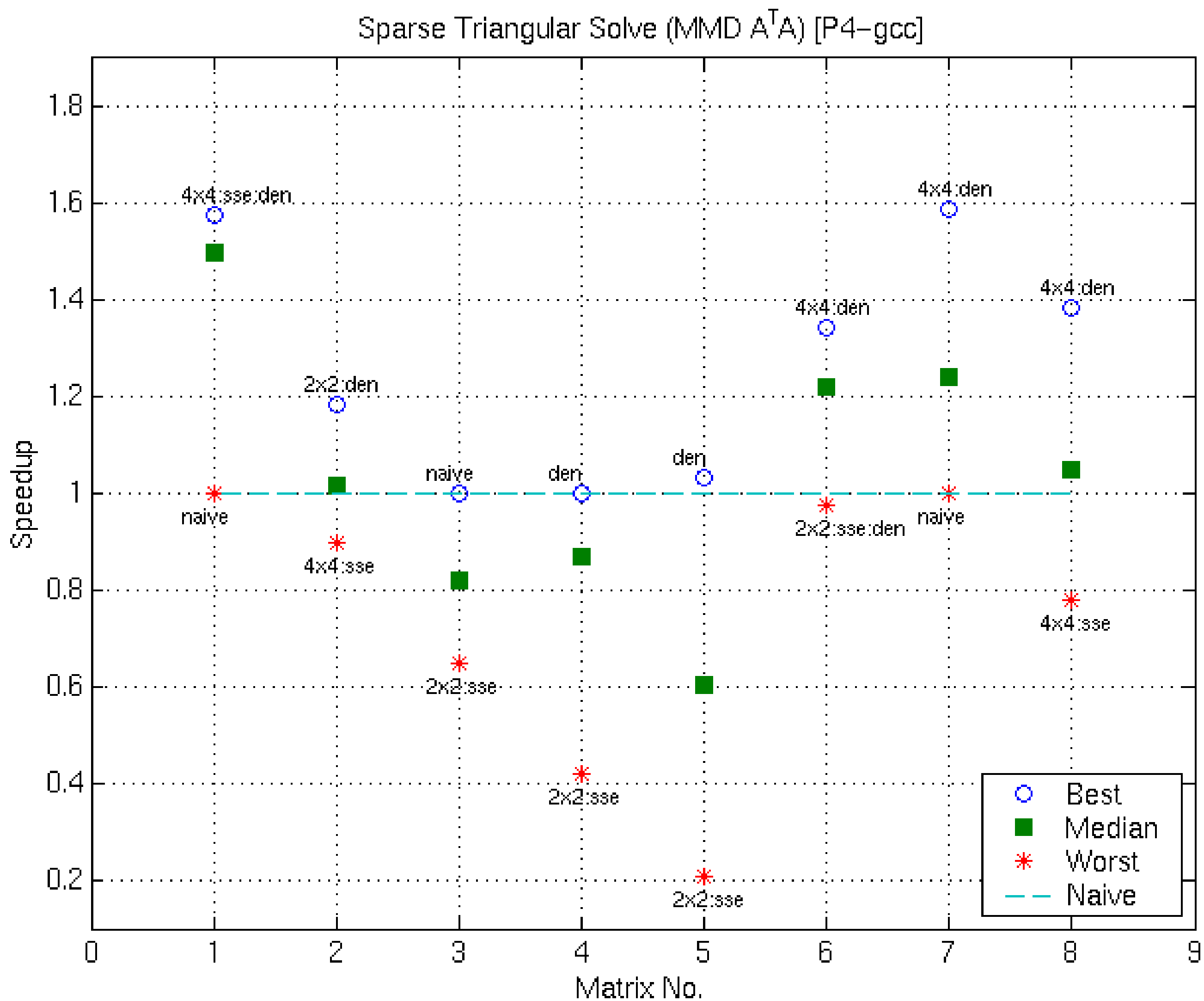
Symmetric Sparse Matrix-Vector Multiply on P4 (vs naïve symmetric = 1)



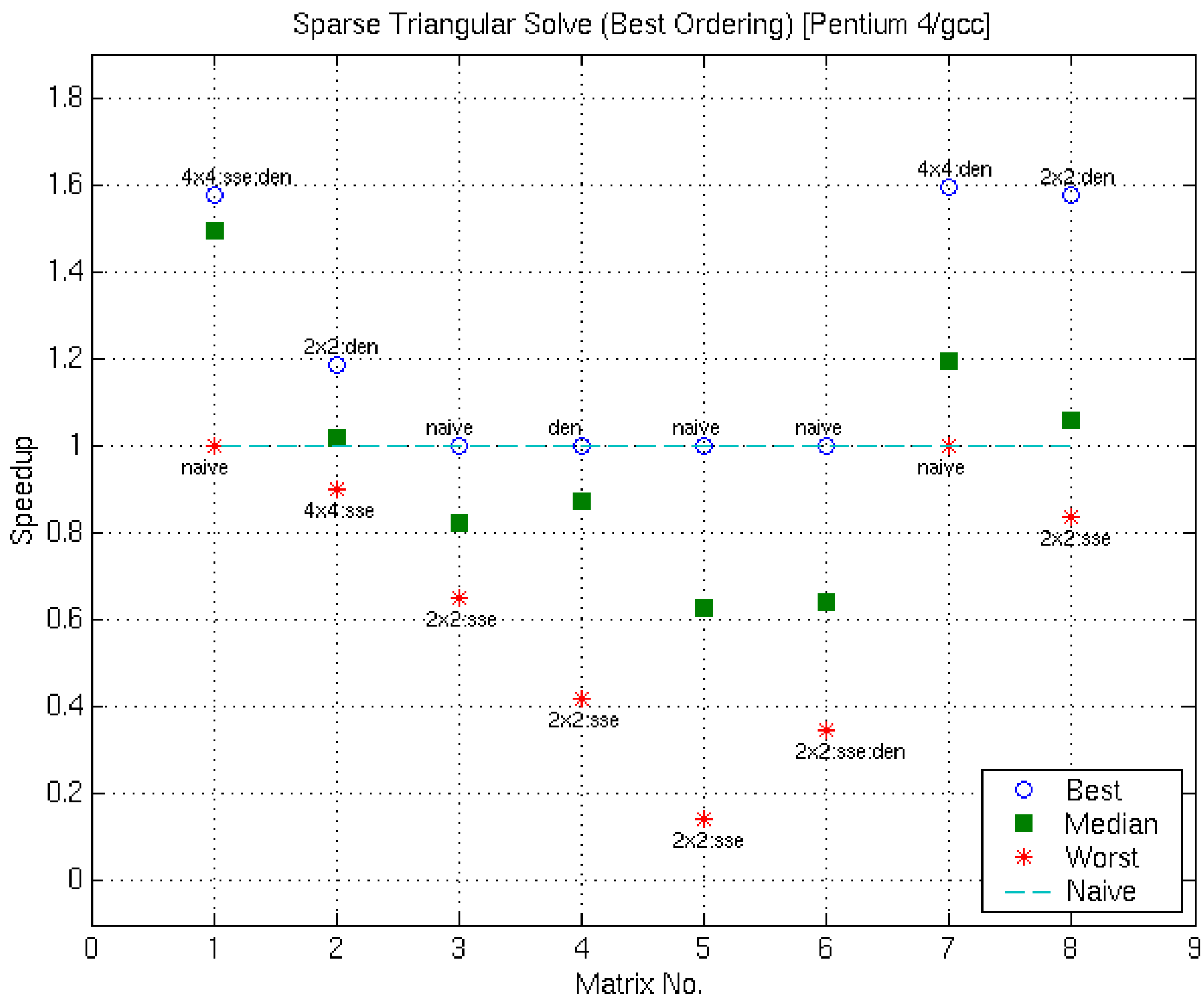
Sparse Triangular Solve (mmd on A^T+A ordering) on P4



Sparse Triangular Solve (mmd on A^T*A ordering) on P4

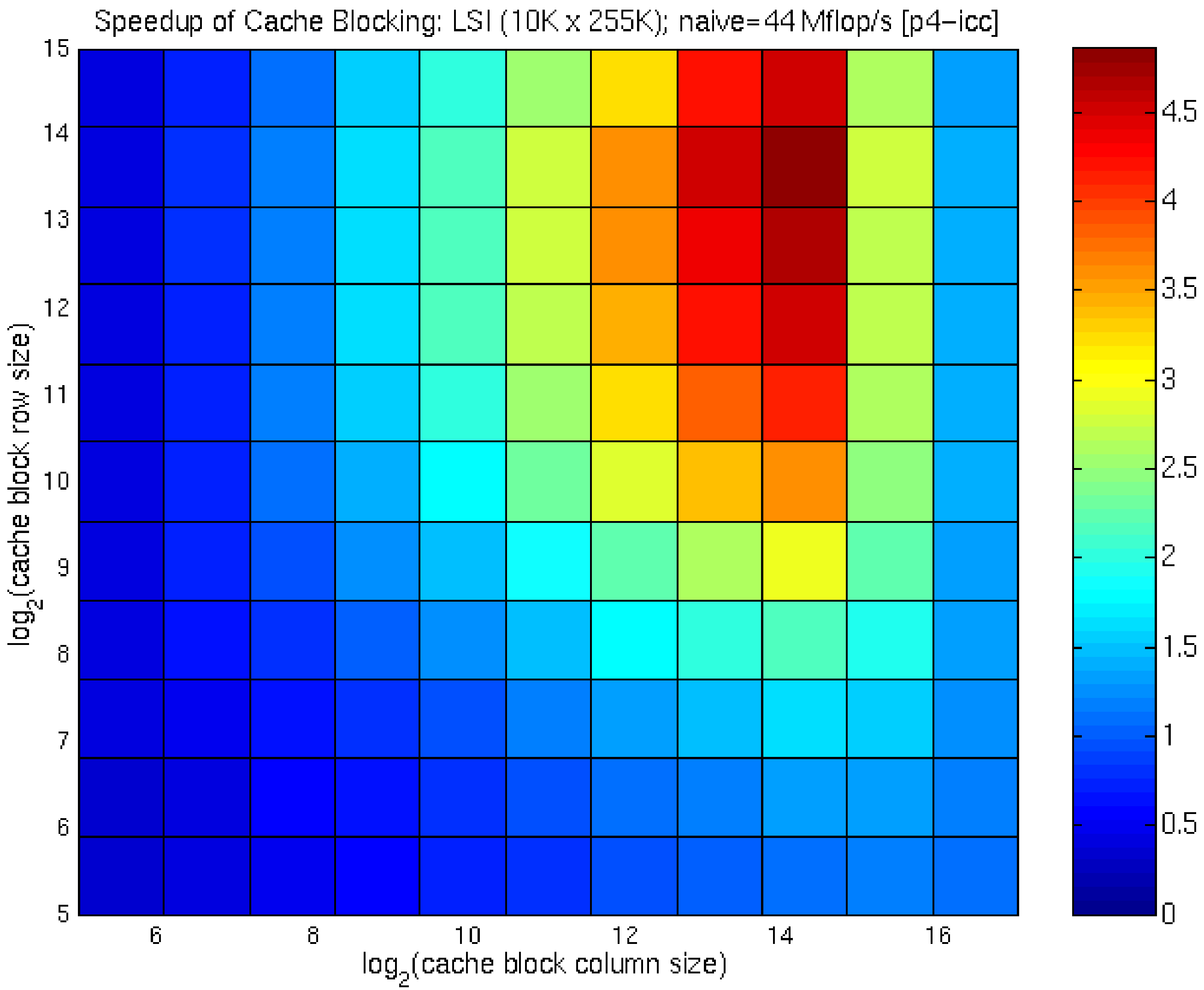


Sparse Triangular Solve (best of 3 orderings) on P4

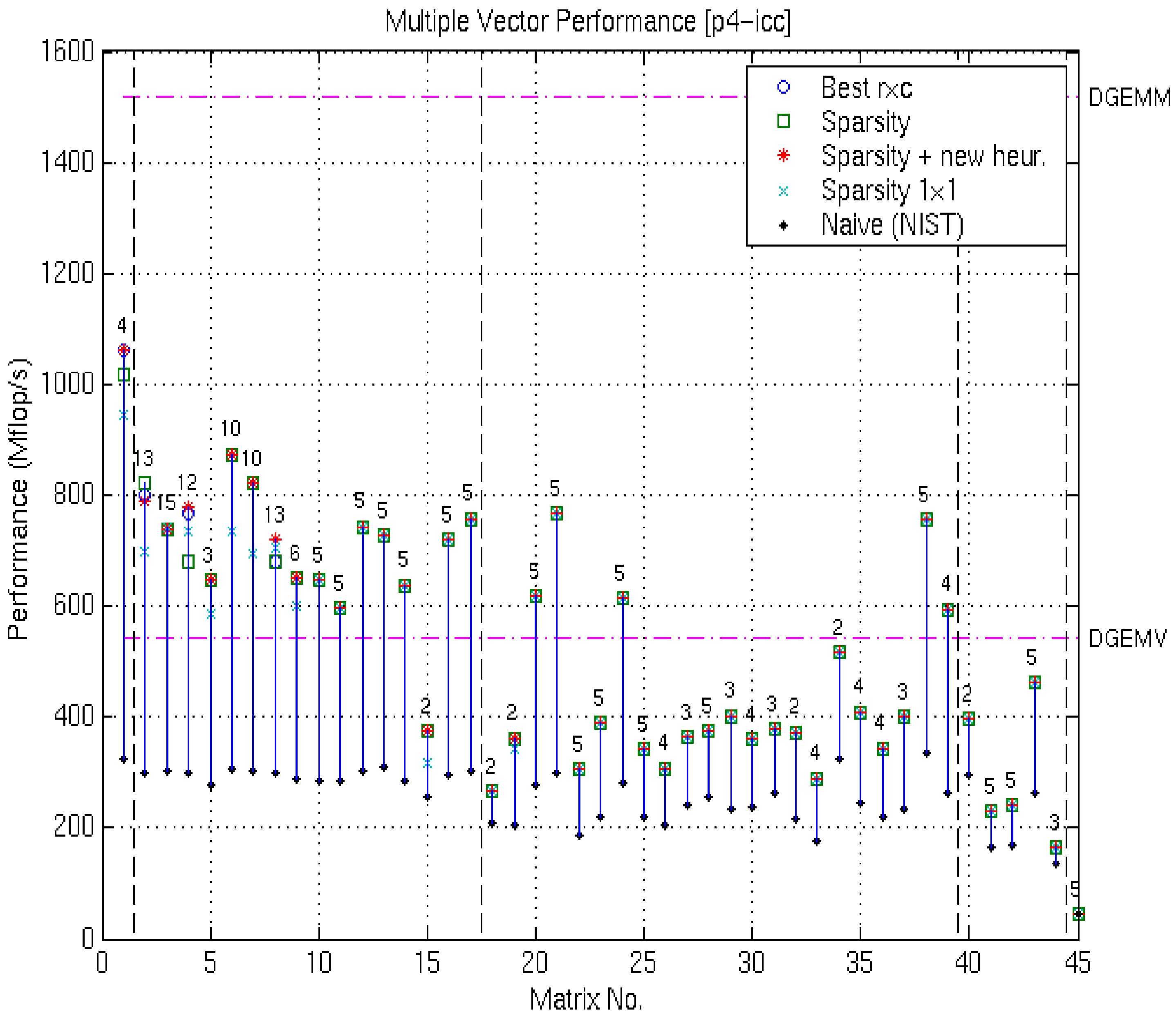


New slides from Rich

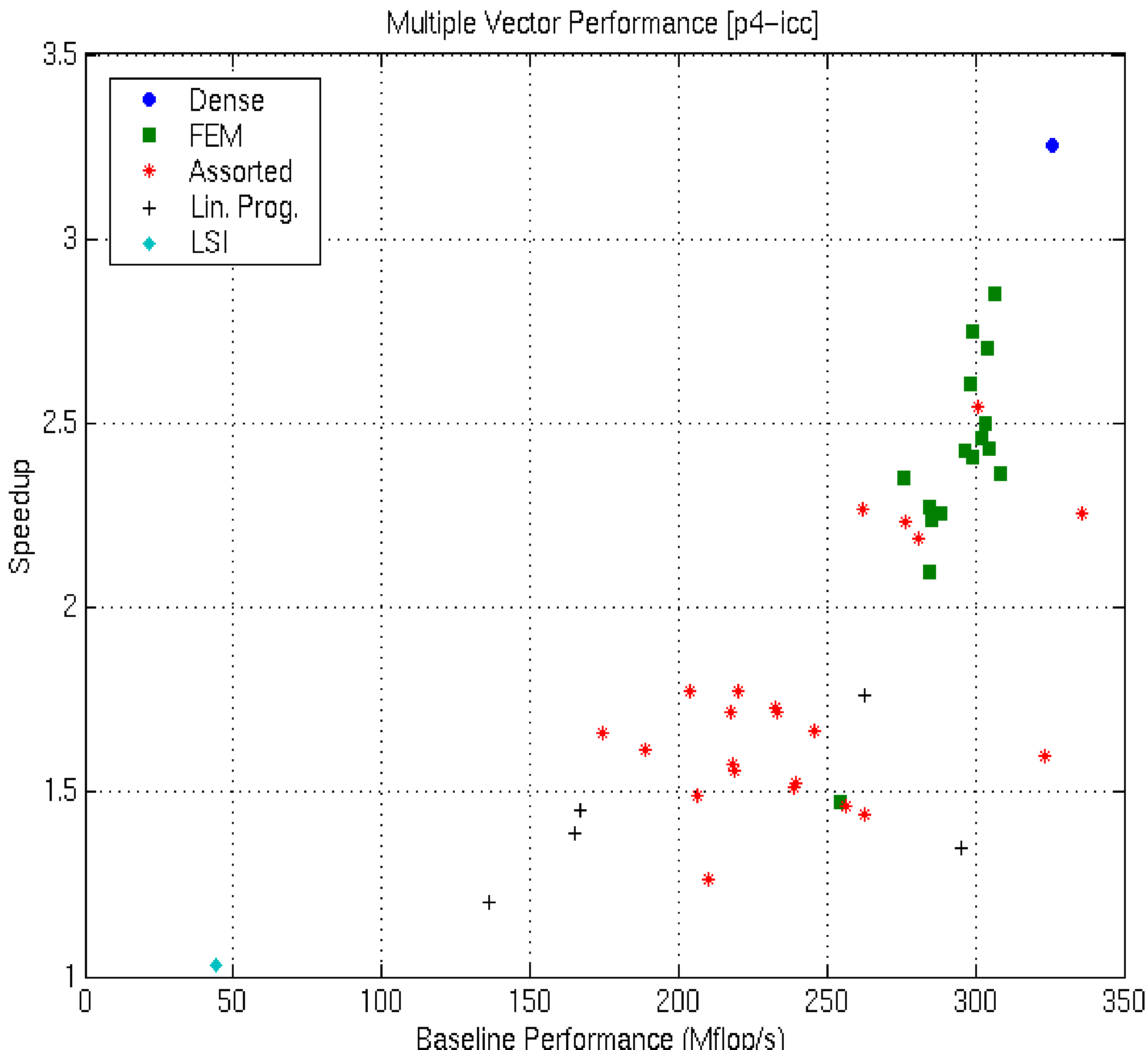
Speed up from Cache Blocking on LSI matrix on P4



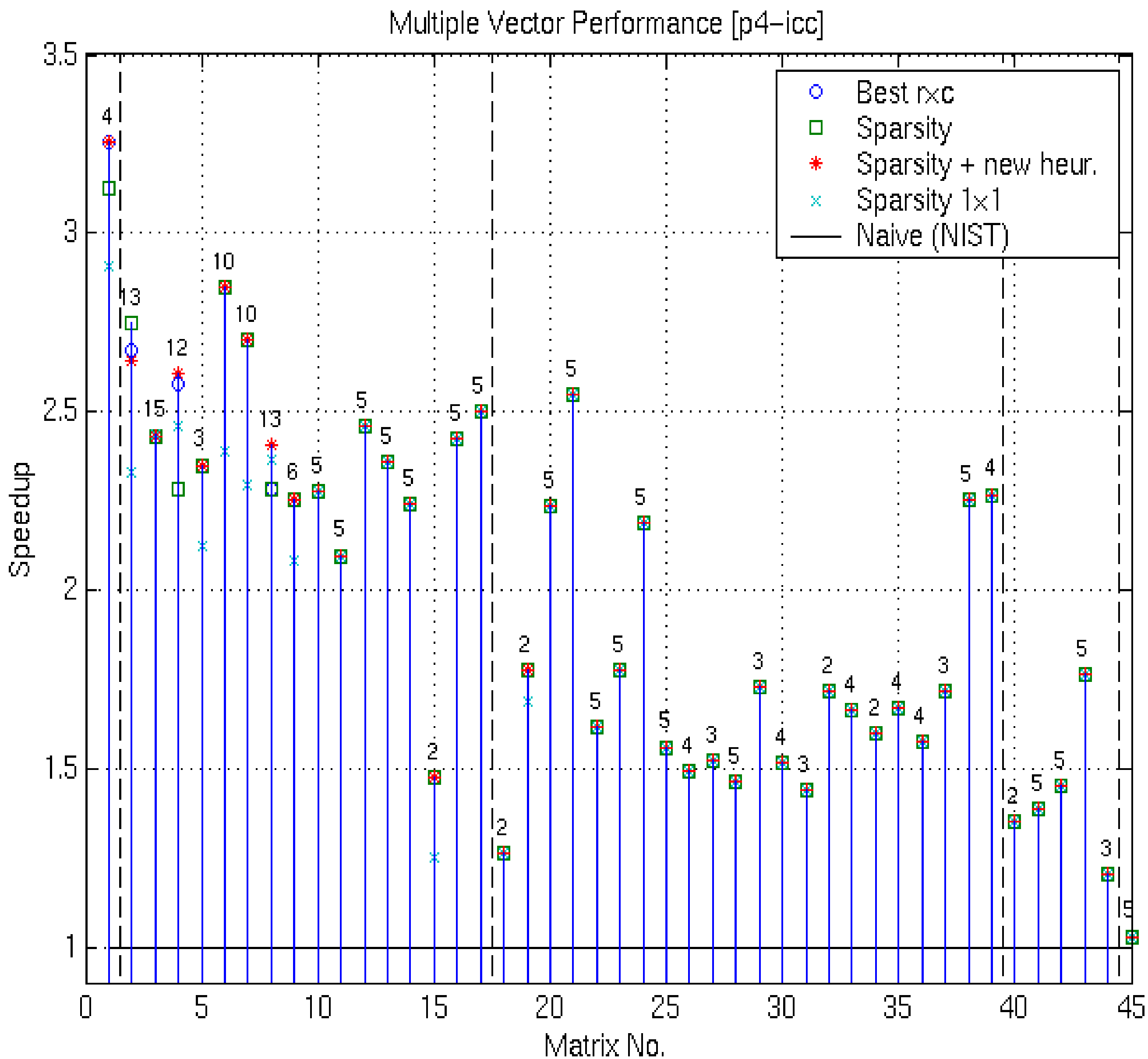
Multiple Vector Performance on P4



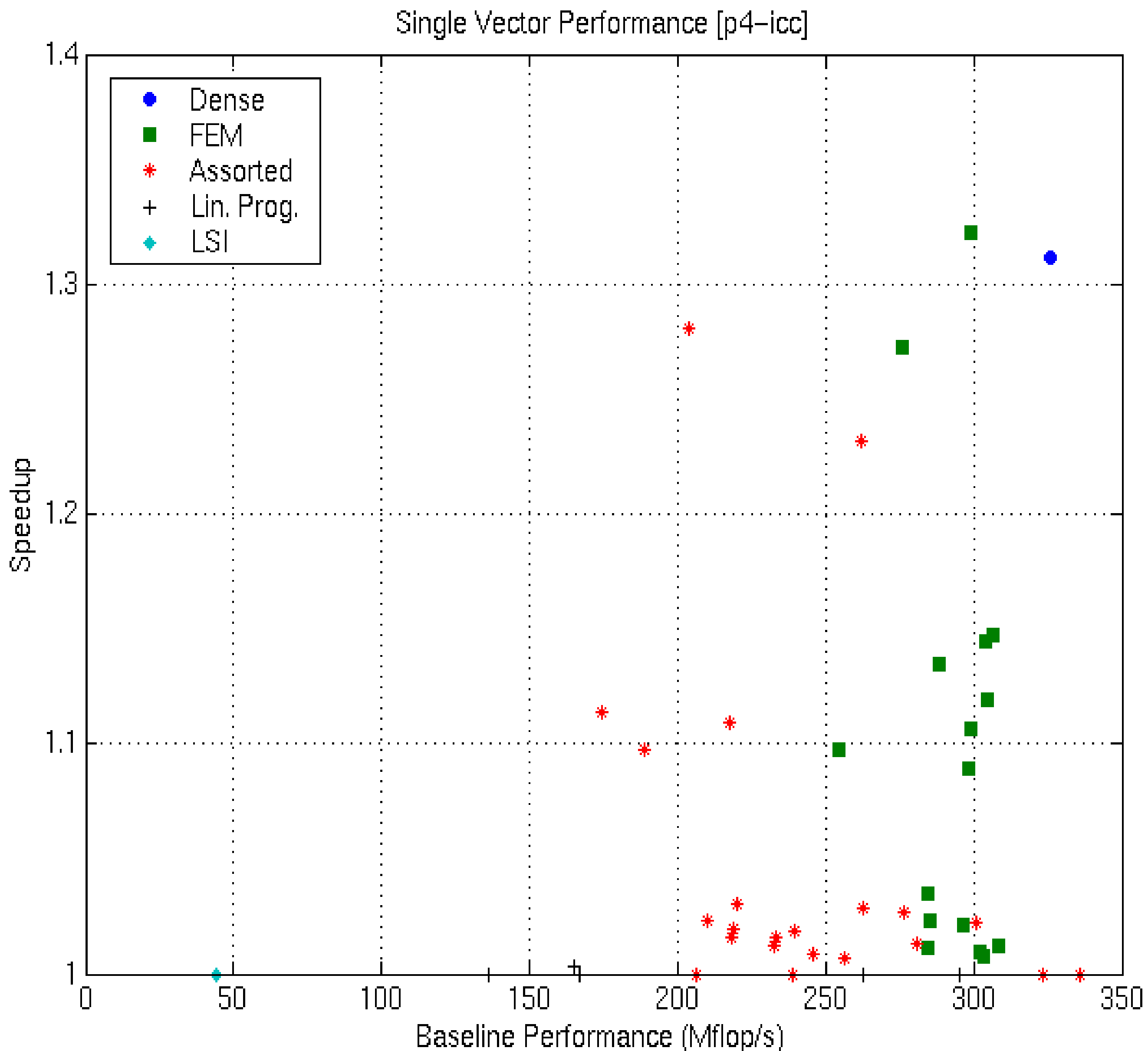
Multiple vector performance on P4 – by matrix type



Multiple vector speedups on P4



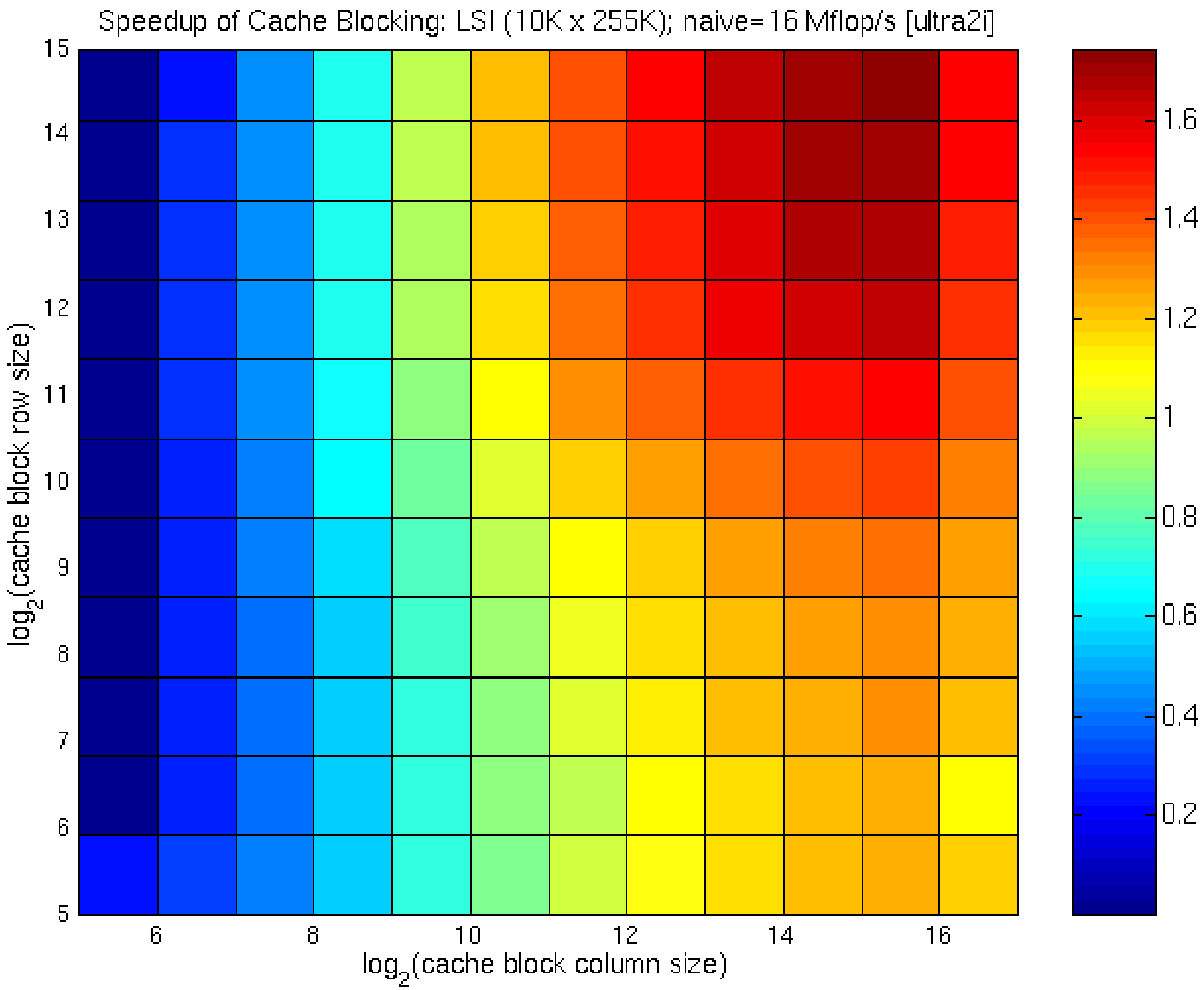
Single vector speedups on P4 by matrix type



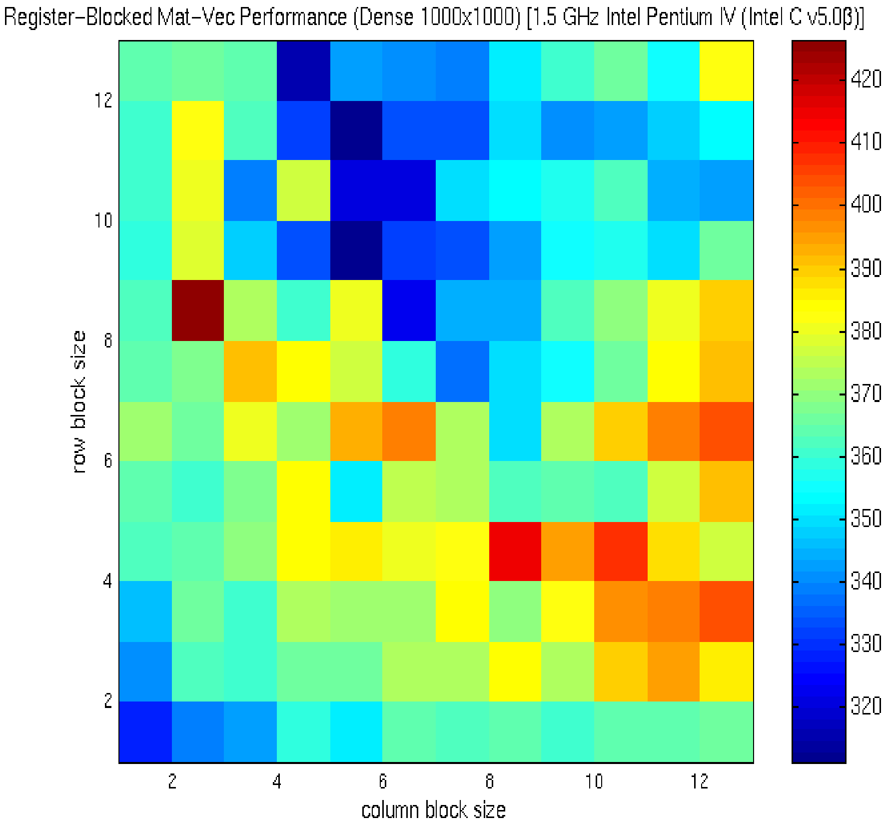
Performance Tuning

- Motivation: performance of many applications dominated by a few kernels
- MEMS CAD → Nonlinear ODEs → Nonlinear equations → Linear equations → Matrix multiply
  - Matrix-by-matrix or matrix-by-vector
  - Dense or Sparse
- Information retrieval by LSI → Compress term-document matrix → … → Sparse mat-vec multiply
- Information retrieval by LDA → Maximum likelihood estimation → … → Solve linear systems
- Many other examples (not all linear algebra)

Speed up from Cache Blocking on LSI matrix on Sun Ultra



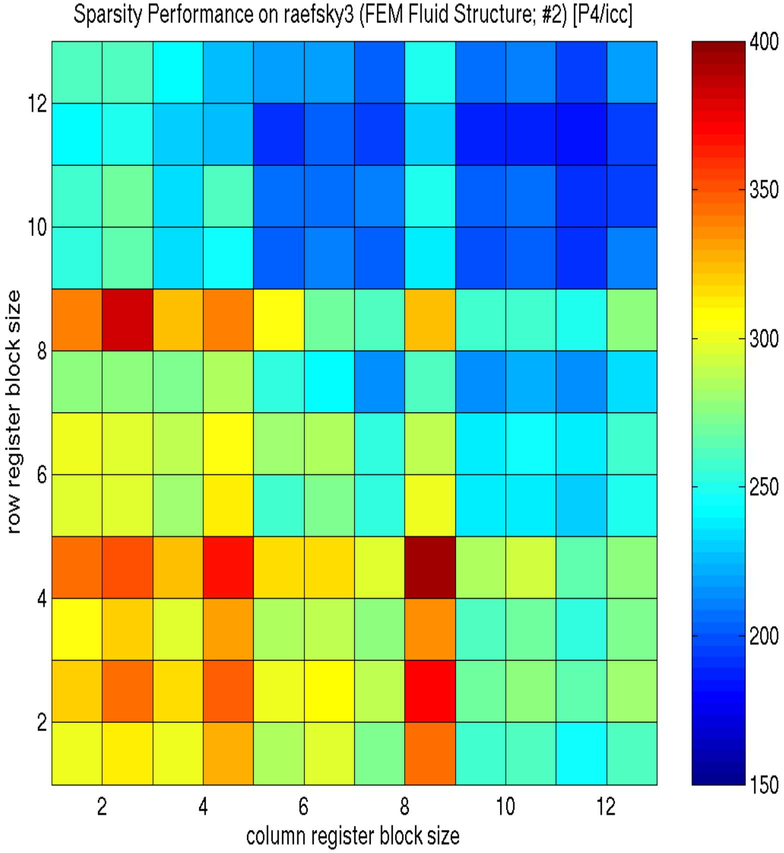
Possible Improvements



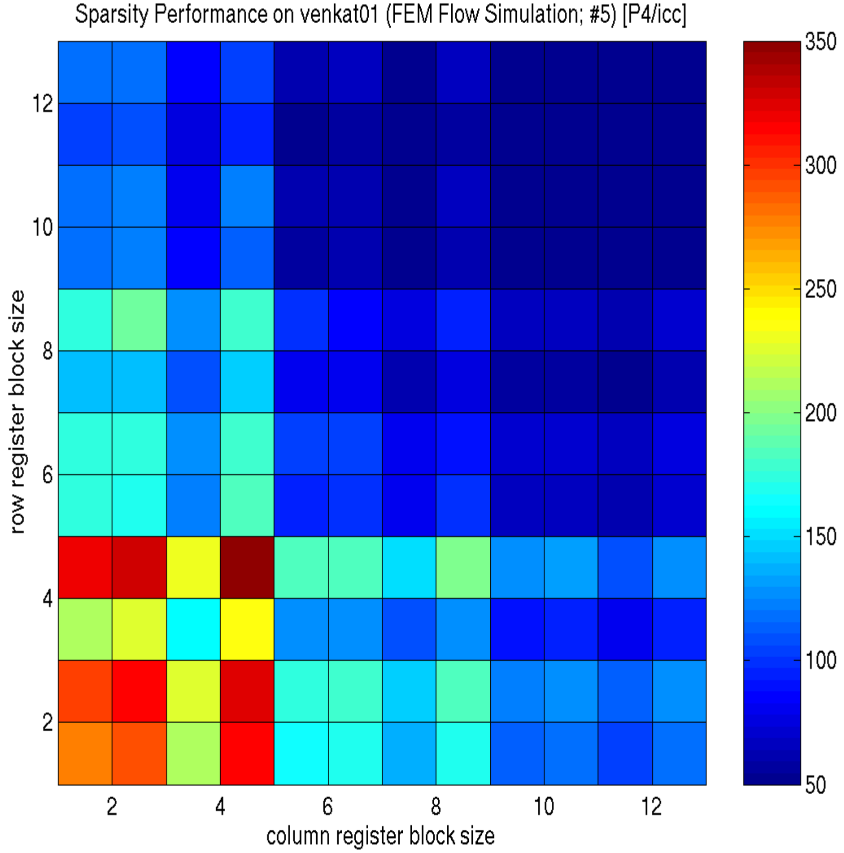
- Doesn't work as well as on Sun Ultra 1/170; Why?
- Current heuristic to determine best r x c block biased to diagonal of performance plot
- Didn't matter on Sun, does on P4 and Itanium since performance so "nondiagonally dominant"

Sparsity reg blocking results on P4 for FEM/fluids matrices

Matrix #2 (150 Mflops to 400 Mflops)



Matrix #5 (50 Mflops to 350 Mflops)



Possible collaborations with Intel

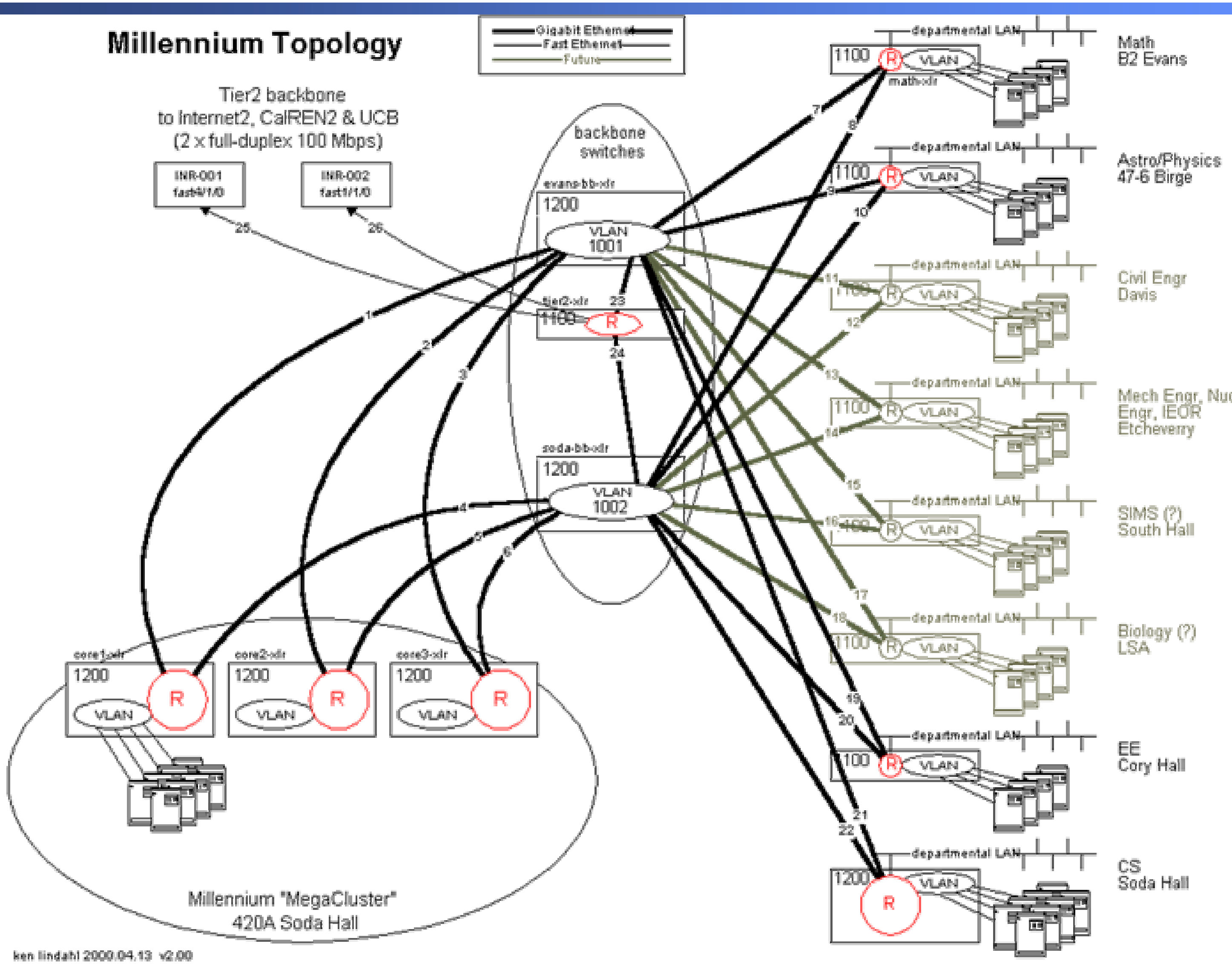
- Getting right tools
- Getting faster, less accurate transcendental functions
- Provide feedback on tools
- Provide tuned kernels, benchmarks, IR apps
- Provide system for tuning future kernels
  - To provide users
  - To evaluate architectural designs

Millennium

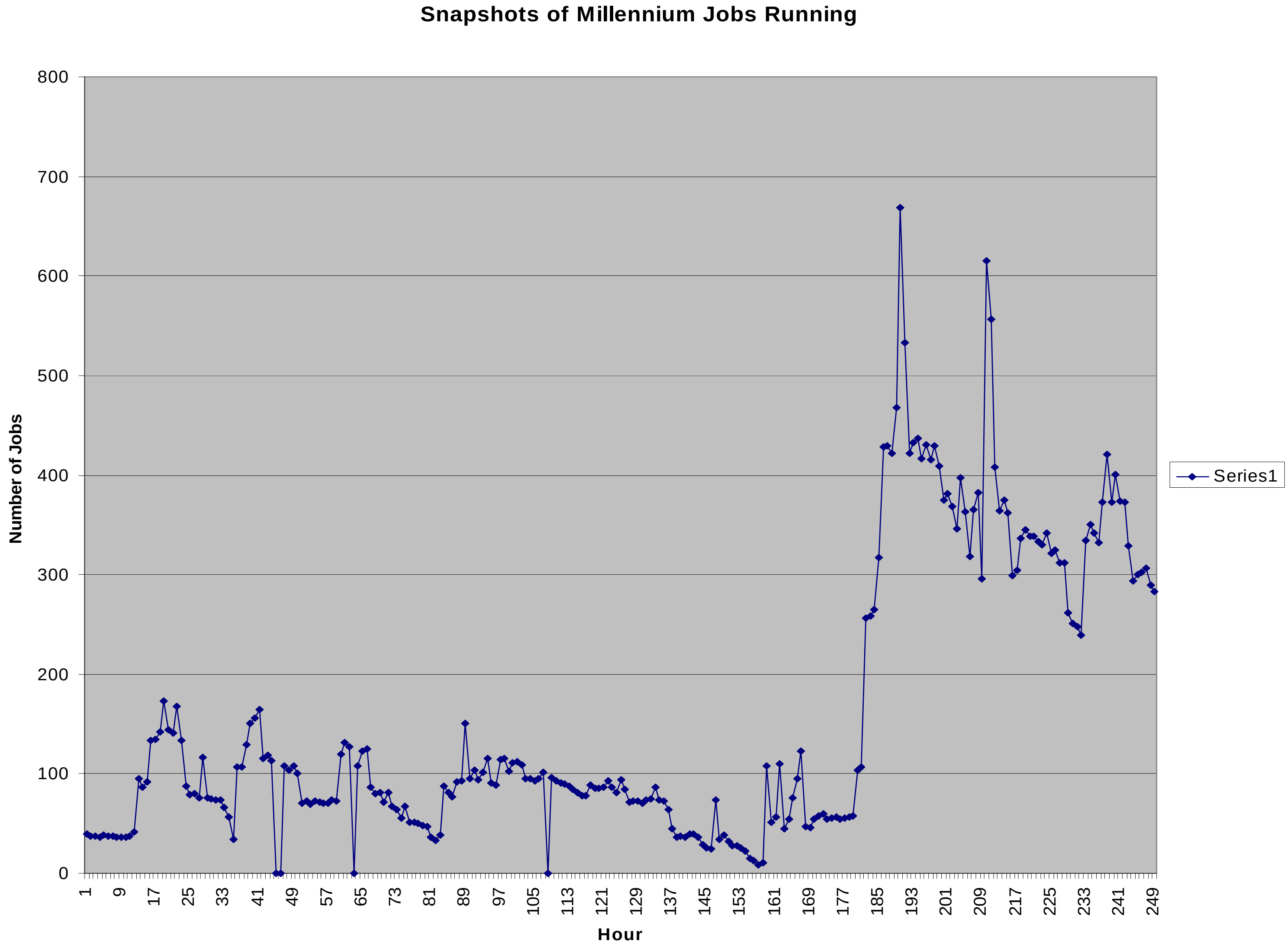
Millennium

- Cluster of clusters at UC Berkeley
  - 309 CPU cluster in Soda Hall
  - Smaller clusters across campus
- Made possible by Intel equipment grant
- Significant other support
  - NSF, Sun, Microsoft, Nortel, campus
- www.millennium.berkeley.edu

Millennium Topology



Millennium Usage Oct 1 – 11, 2001



100% utilization for last few days
About half the jobs are parallel

Usage highlights

- AMANDA
  - Antarctic Muon And Neutrino Detector Array
  - amanda.berkeley.edu
  - 128 scientists from 15 universities and institutes in the U.S. and Europe.
- TEMPEST
  - EUV lithography simulations via 3D electromagnetic scattering
  - cuervo.eecs.berkeley.edu/Volcano/
  - study the defect printability on multilayer masks
- Titanium
  - High performance Java dialect for scientific computing
  - www.cs.berkeley.edu/projects/titanium
  - Implementation of shared address space, and use of SSE2
- Digital Library Project
  - Large database of images
  - elib.cs.berkeley.edu/
  - Used to run spectral image segmentation algorithm for clustering, search on images

Usage highlights (continued)

- CS 267
  - Graduate class in parallel computing, 33 enrolled
  - www.cs.berkeley.edu/~dbindel/cs267ta
  - Homework
- Disaster Response
  - Help find people after Sept 11, set up immediately afterwards
  - safe.millennium.berkeley.edu
  - 48K reports in database, linked to other survivor databases
- MEMS CAD (MicroElectroMechanical Systems Computer Aided Design)
  - Tool to help design MEMS systems
  - Used this semester in EE 245, 93 enrolled
  - sugar.millennium.berkeley.edu
  - More later in talk
- Information Retrieval
  - Development of faster information retrieval algorithms
  - www.cs.berkeley.edu/~jordan
  - More later in talk
- Many applications are part of CITRIS