Automatic Performance Tuning and Analysis of Sparse Triangular Solve

Richard Vuduc, Shoaib Kamil, Jen Hsu, Rajesh Nishtala

James W. Demmel, Katherine A. Yelick

June 22, 2002

Berkeley Benchmarking and OPtimization (BeBOP) Project

www.cs.berkeley.edu/~richie/bebop

Computer Science Division, U.C. Berkeley

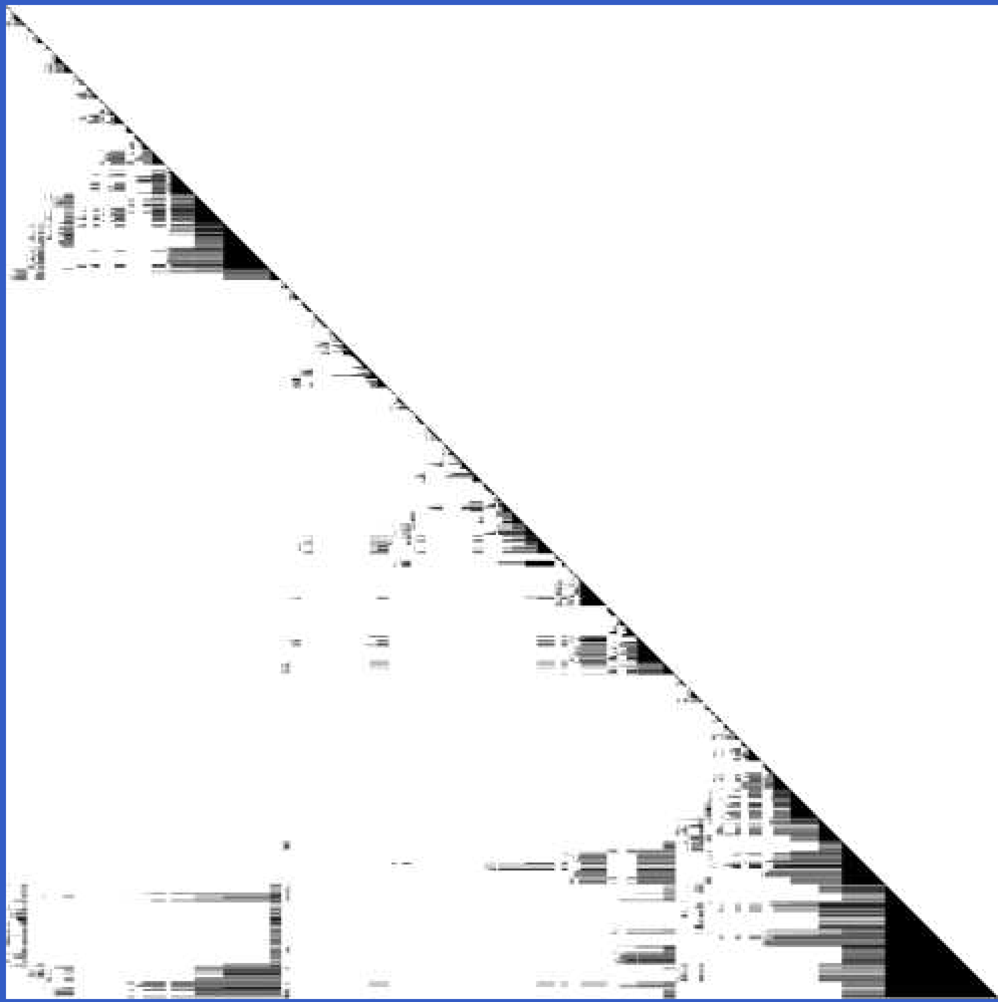
Context: High-Performance Libraries

- Application performance dominated by a few *computational kernels*
  - Solving PDEs (linear algebra ops)
  - Google (sparse matrix-vector multiply)
  - Multimedia (signal processing)
- Performance tuning today
  - Vendor-tuned standardized libraries (*e.g.*, BLAS)
  - User tunes by hand
- Automated tuning for dense linear algebra, FFTs, . . .
  - PHiPAC/ATLAS (dense linear algebra)
  - FFTW/SPIRAL/UHFFT (signal processing)

Problem Area: Sparse Matrix Kernels

- Performance issues in sparse linear algebra
  - High bandwidth requirements and poor instruction mix
  - Depends on architecture, kernel, and *matrix*
  - How to select data structures, algorithms? at run-time?
- Approach to automatic tuning: for each kernel,
  - *Identify* and *generate* a space of implementations
  - *Search* (models, experiments) to find the fastest one
- Early success: SPARSITY (Im & Yelick '99) for sparse matrix-vector multiply (SpM$\times$V)
- This talk: *Sparse triangular solve* (SpTS), arising in sparse Cholesky and LU factorization (uniprocessor)

Sparse Triangular Matrix Example



- `raefsky4` (structural problem) + SuperLU 2.0 + colmmd
- Dimension: 19779
- No. non-zeros: 12.6 M
- Dense trailing triangle:
  - dim=2268
  - 20% of total nnz

Idea: Sparse/Dense Partitioning

Partition the matrix into sparse (L_1, L_2) and dense (L_D) parts:

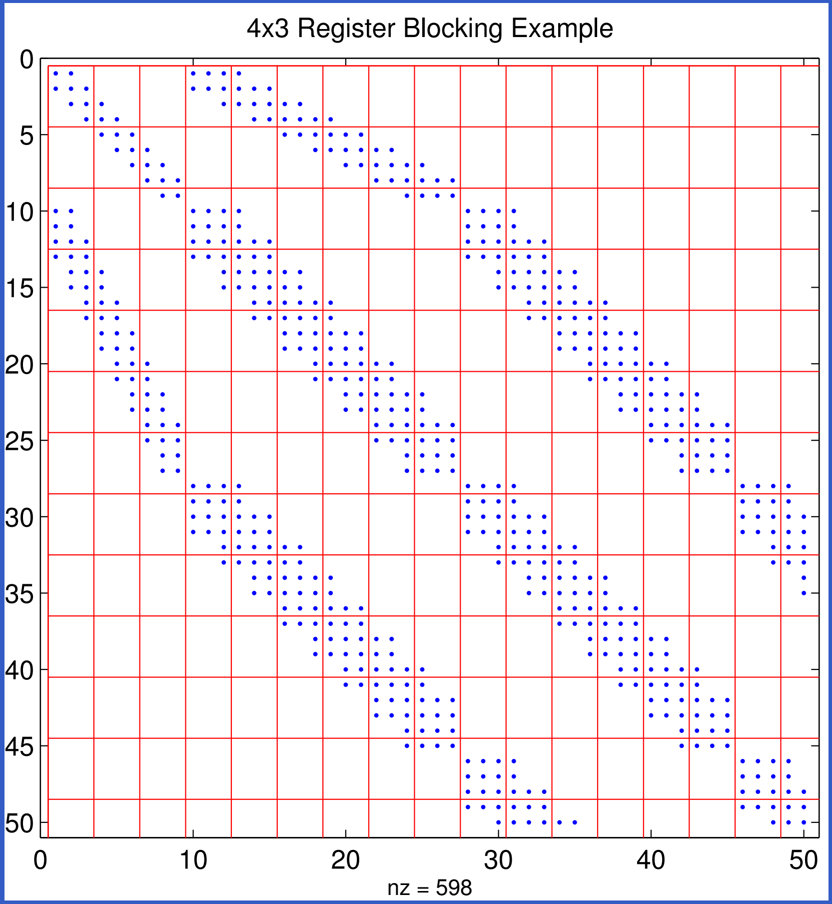
$$\begin{pmatrix} L_1 & \\ L_2 & L_D \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

Leads to 1 SpTS, 1 SpM$\times$V, and 1 Dense TS:

$$\begin{aligned} L_1 x_1 &= y_1 && (1) \\ \hat{y}_2 &= y_2 - L_2 x_1 && (2) \\ L_D x_2 &= \hat{y}_2 && (3) \end{aligned}$$

SPARSITY optimizations for (1)–(2); tuned BLAS for (3).

Register Blocking (SPARSITY)



- Store $r \times c$ dense blocks
- Multiply/solve block-by-block
- Fill in explicit zeros
- 1.3x–2.5x speedup on FEM matrices (SpM$\times$V)
- Reduced storage overhead over, *e.g.*, CSR
- Block ops are fully unrolled – improves register reuse
- Trade-off extra computation for efficiency

Tuning Parameter Selection

- Parameters: *switch point*, s, and *register block size*, b
- Off-line profiling
  - Benchmark routines on synthetic data
  - Only needed *once* per architecture
- At run-time (when matrix is known)
  - Determine or estimate matrix properties (*e.g.*, fill ratio, size of trailing triangle)
  - Combine with data collected off-line
  - Convert to new data structure
- In practice, total run-time cost to select and reorg: *e.g.*, 10–30 naïve solves on Itanium

Performance Bounds

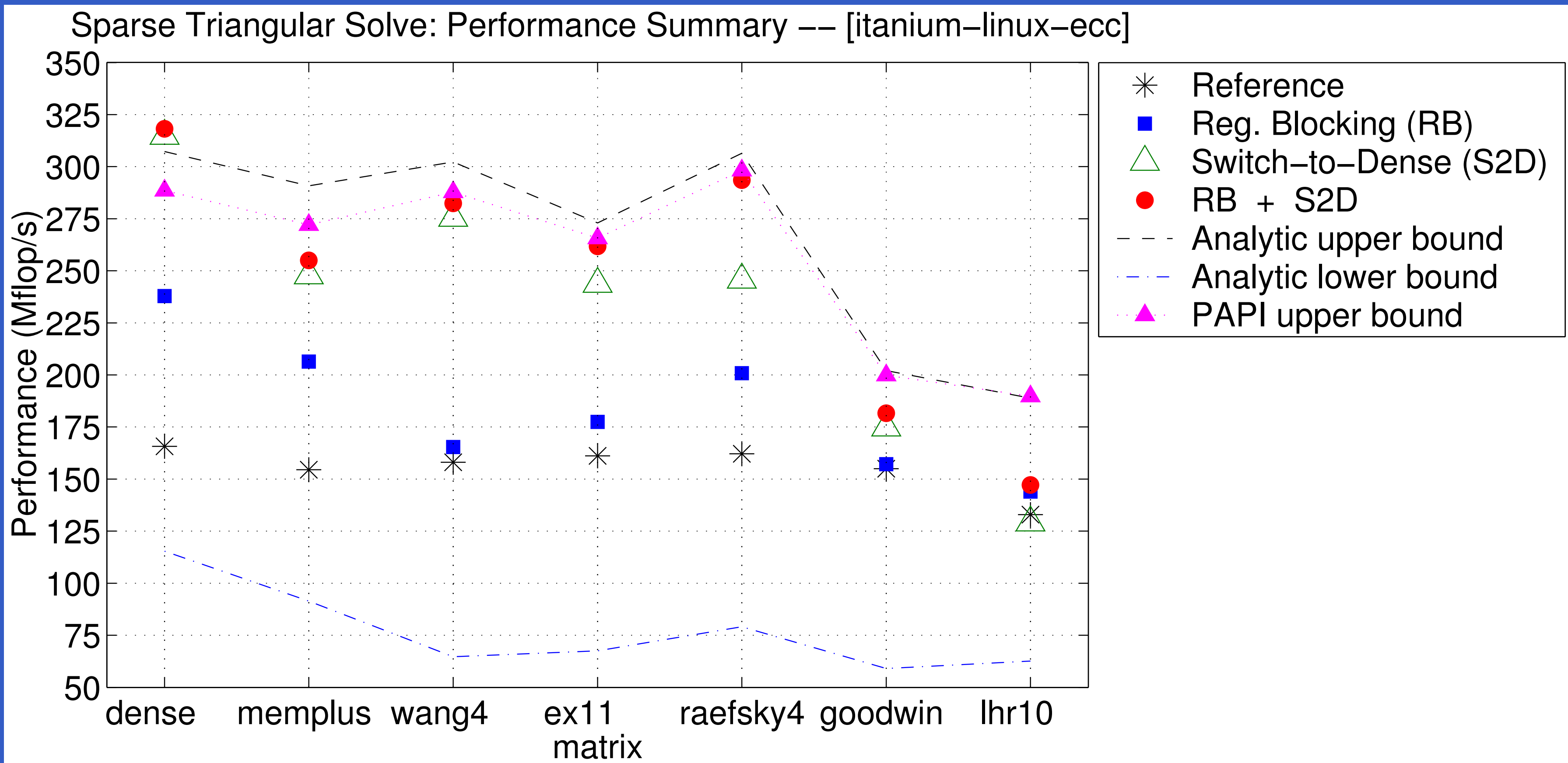
- *Upper-bounds* on performance (Mflop/s)?
- Flops: 2 * (number of non-zeros) - (dimension)
- Full latency cost model of execution time:

$$T(s,b) = \sum_{i=1}^{\kappa-1} H_i(s,b)\alpha_i + M_\kappa(s,b)\alpha_{\text{mem}} \tag{4}$$

- Lower bound on misses: ignore conflict misses on vectors

$$M_{\text{lower}}^{(i)}(s,b) = \frac{1}{l_i}\left[k f_b\left(1+\frac{1}{\gamma b^2}\right) + \frac{1}{\gamma}\left(\left\lceil\frac{n}{b}\right\rceil + 1\right) + \left(2n + \frac{(n-s)\left((n-s)+1\right)}{2}\right)\right] \tag{5}$$

Performance Results: Intel Itanium



Conclusions and Directions

- Limits of “low-level” tuning are near
  - Can we approach bandwidth limits?
  - Other kernels? $A^T Ax$, $A^k x$, RAR^T
  - Other structures? multiple vectors, symmetry, reordering
- Interface from/to libraries and applications?
  - Leverage existing generators (*e.g.*, Bernoulli)
  - Hybrid on-line, off-line optimizations
- SpTS-specific future work
  - symbolic structure; other fill-reducing orderings
  - refinements to switch point selection
  - Incomplete Cholesky and LU preconditioners

Related Work (1/R)

- Automatic tuning systems
  - PHiPAC [BACD97], ATLAS [WPD01], SPARSITY [Im00]
  - FFTW [FJ98], SPIRAL [PSVM01], UHFFT [MMJ00]
  - MPI collective ops (Vadhiyar, *et al.* [VFD01])
- Code generation
  - FLAME [GGHvdG01]
  - Sparse compilers (Bik [BW99], Bernoulli [Sto97])
  - Generic programming (Blitz++ [Vel98], MTL [SL98], GMCL [Neu98], . . .)
- Sparse performance modeling
  - Temam and Jalby [TJ92]
  - White and Sadayappan [WS97]
  - Navarro [NGLPJ96], Heras [HPDR99], Fraguela [FDZ99]

Related Work (2/R)

- Compilers (analysis and models); run-time selection
  - CROPS (UCSD/Carter, Ferrante, *et al.*)
  - TUNE (Chatterjee, *et al.*)
  - Iterative compilation (O'Boyle, *et al.*, 1998)
  - Broadway (Guyer and Lin, '99)
  - Brewer ('95); ADAPT (Voss, 2000)
- Interfaces: Sparse BLAS; PSBLAS; PETSc
- Sparse triangular solve
  - SuperLU/MUMPS/SPOOLES/UMFPACK/PSPASES…
  - Approximation: Alvarado ('93); Raghavan ('98)
  - Scalability: Rothberg ('92;'95); Gupta ('95); Li, Coleman ('88)

—End—

Tuning Parameter Selection

- First, select *switch point*, s; at run-time:
  - Assume matrix in CSR format on input
  - Scan bottom row from diag. until two consecutive zeros found
  - Fill vs. efficiency trade-off
- Then, select *register block size*, b
  - Maximize, over all b,

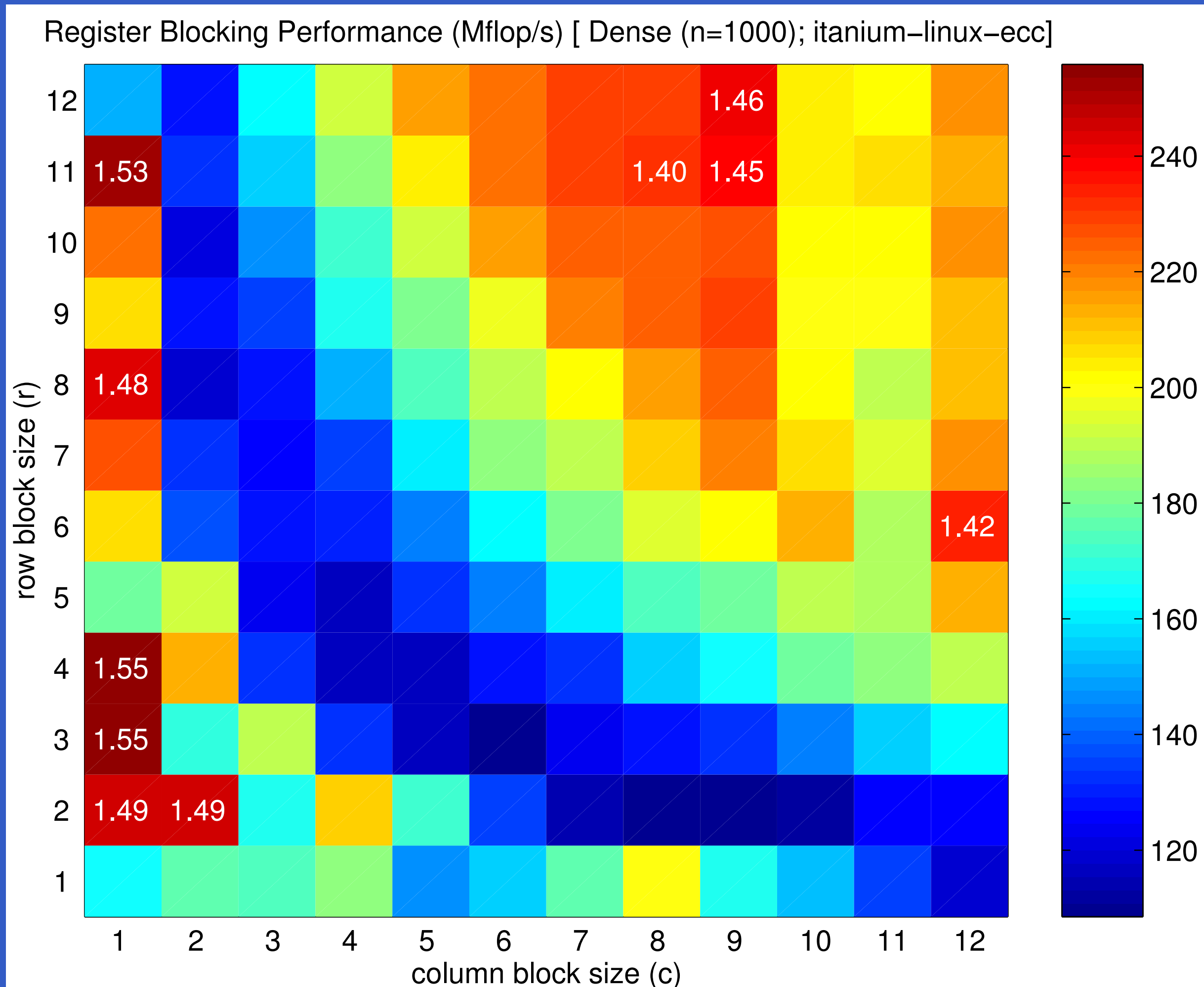
$$\frac{\text{Dense matrix in sparse format performance (Mflop/s) at } b}{\text{Estimated fill ratio at } b} \tag{6}$$

- Total cost to select and reorg.: *e.g.*, 10–30 naïve solves on Itanium

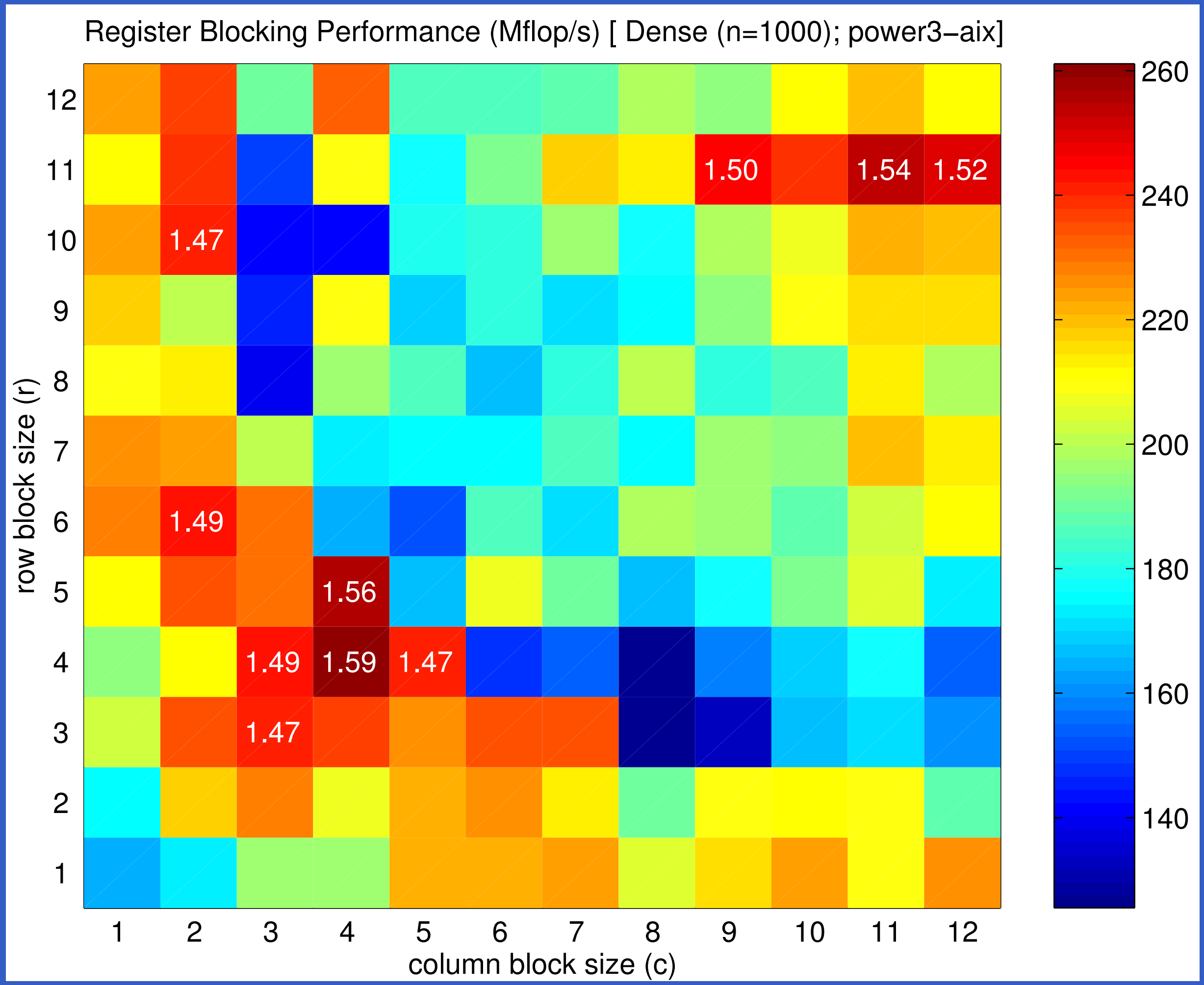
Matrix Benchmark Suite

Name	Application Area	Dim.	Nnz in L	Dense Trailing Triangle Dim.	Dense Trailing Triangle Density	Dense Trailing Triangle % Total Nnz
dense	Dense matrix	1000	500k	1000	100.0%	100.0%
memplus	Circuit simulation	17758	2.0M	1978	97.7%	96.8%
wang4	Device simulation	26068	15.1M	2810	95.0%	24.8%
ex11	Fluid flow	16614	9.8M	2207	88.0%	22.0%
raefsky4	Structural mechanics	19779	12.6M	2268	100.0%	20.4%
goodwin	Fluid mechanics	7320	1.0M	456	65.9%	6.97%
lhr10	Chemical processes	10672	369k	104	96.3%	1.43%

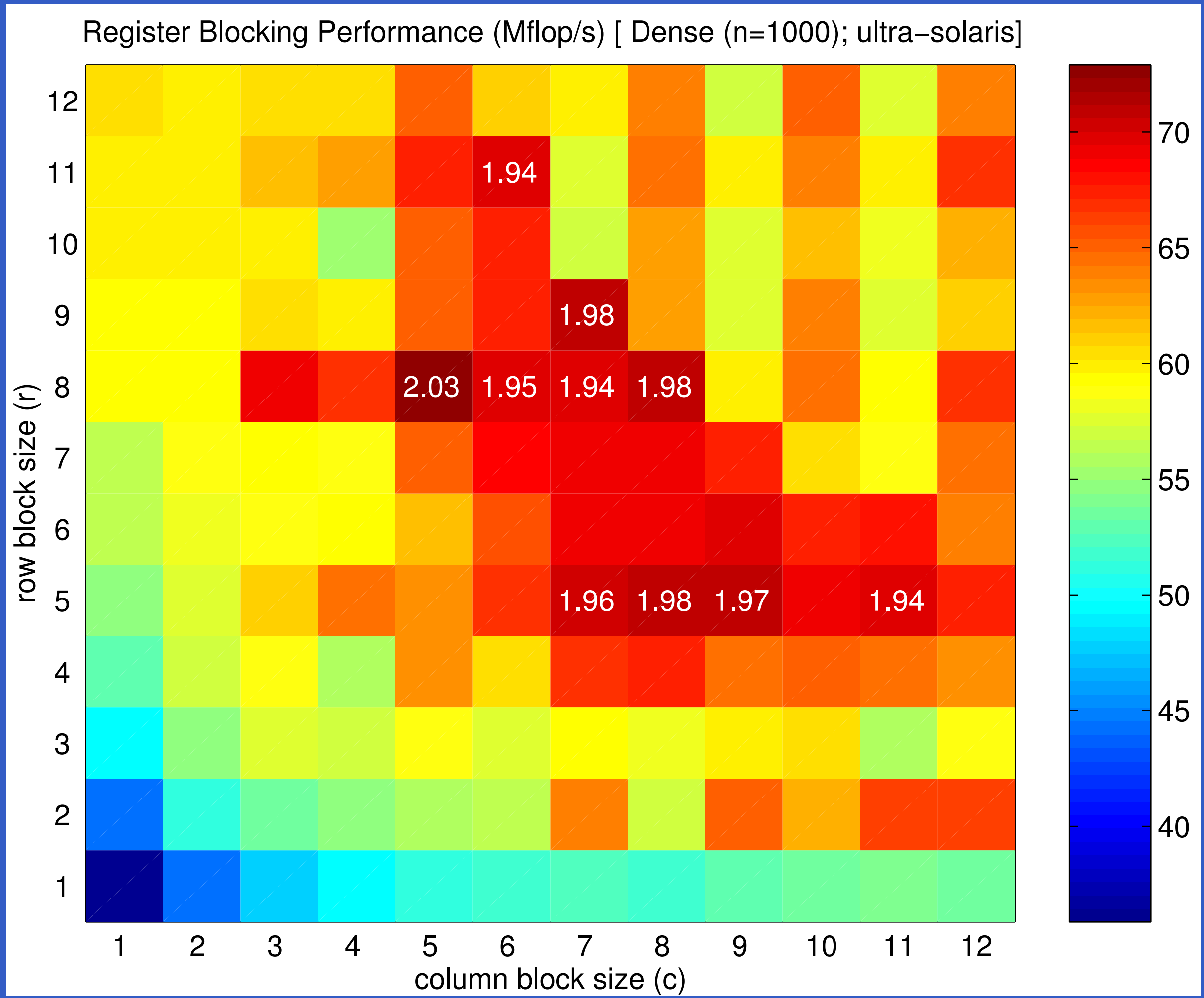
Register Profile (Intel Itanium)



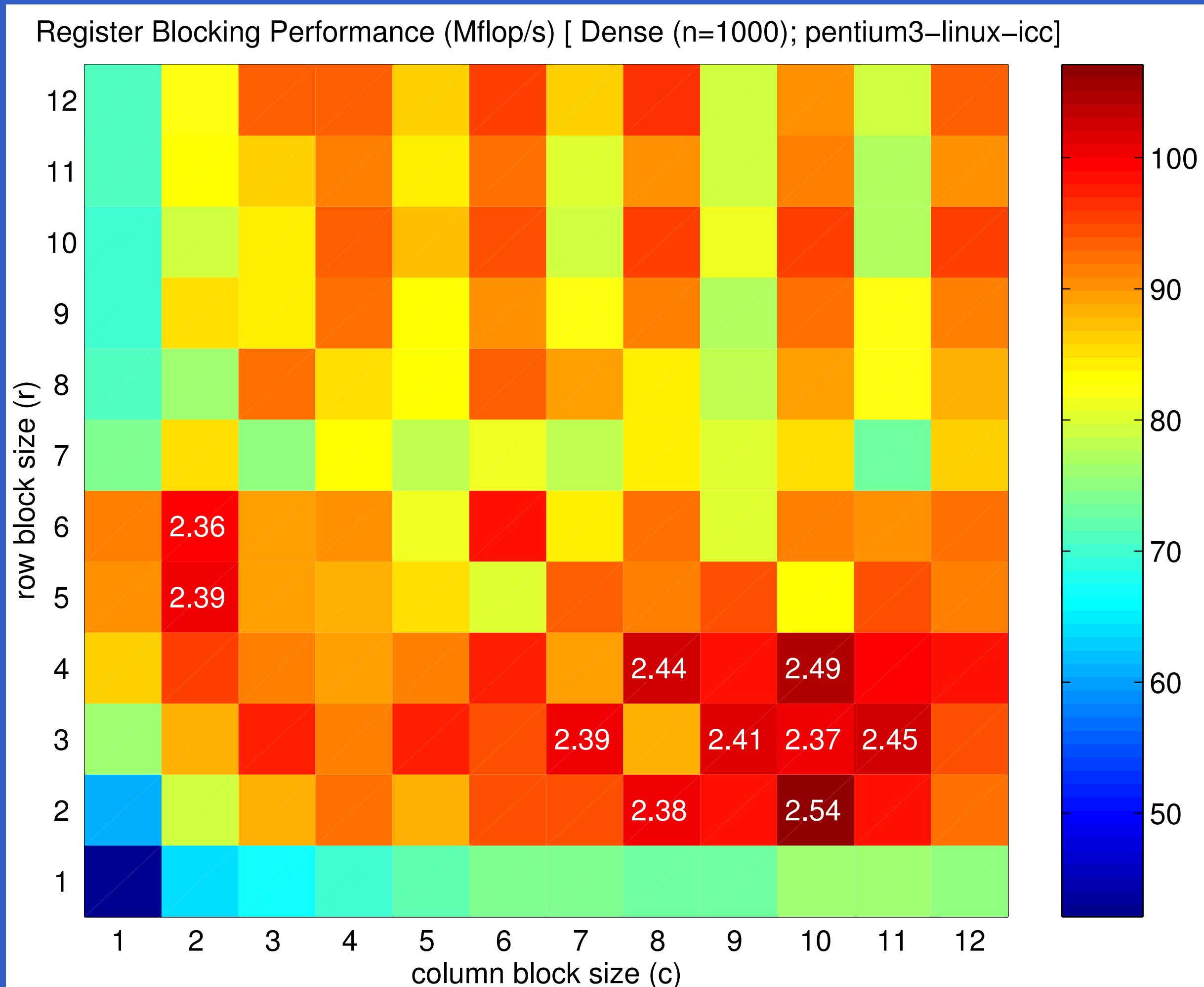
Register Profile (IBM Power3)



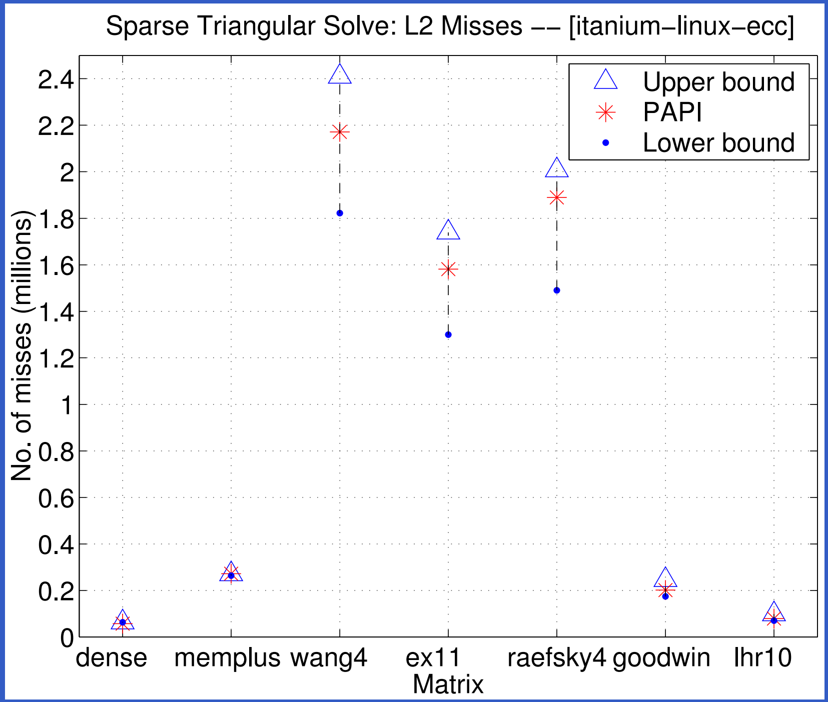
Register Profile (Sun Ultra 2i)

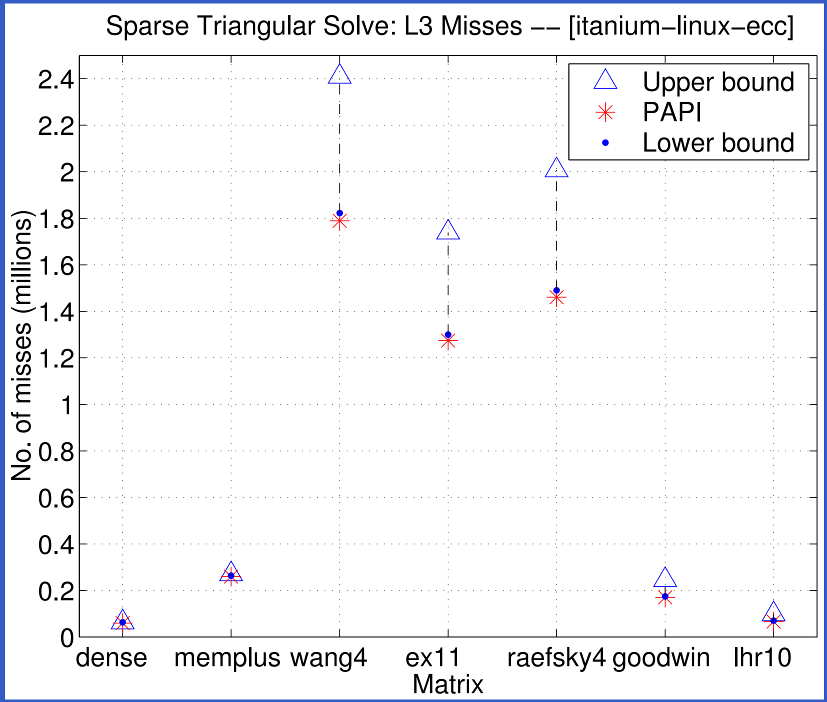


Register Profile (Intel Pentium III)

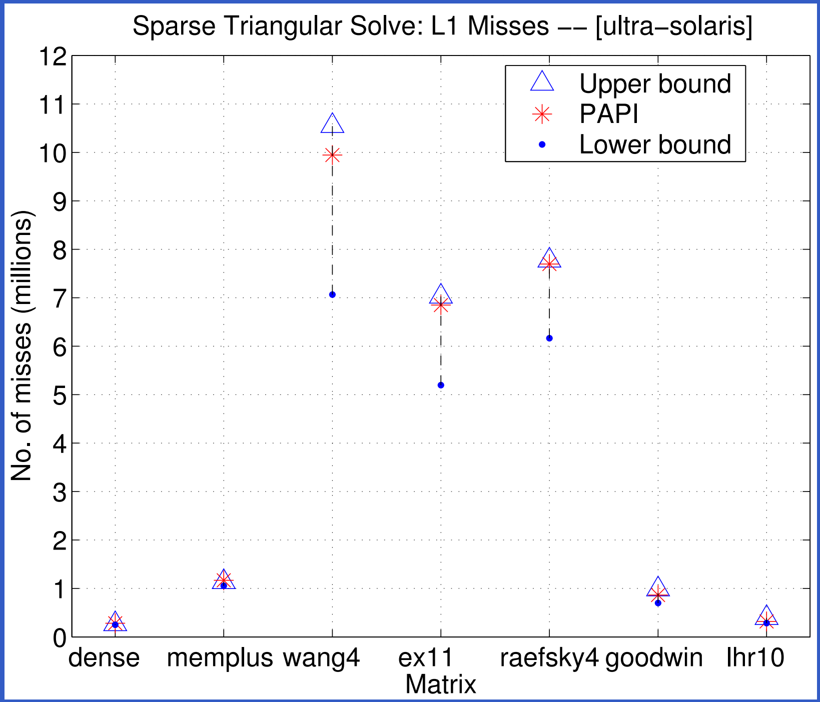


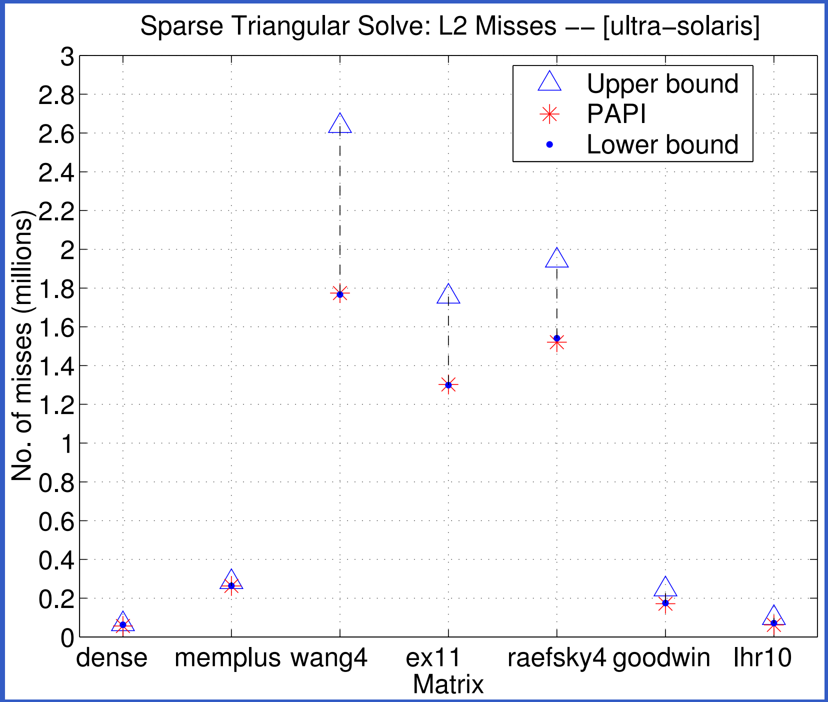
Miss Model Validation: Intel Itanium



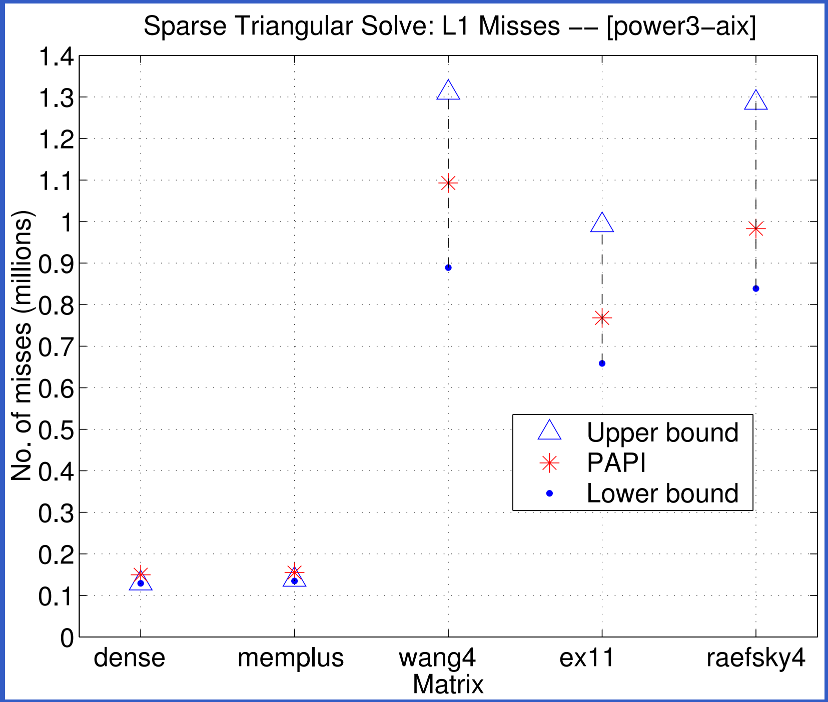


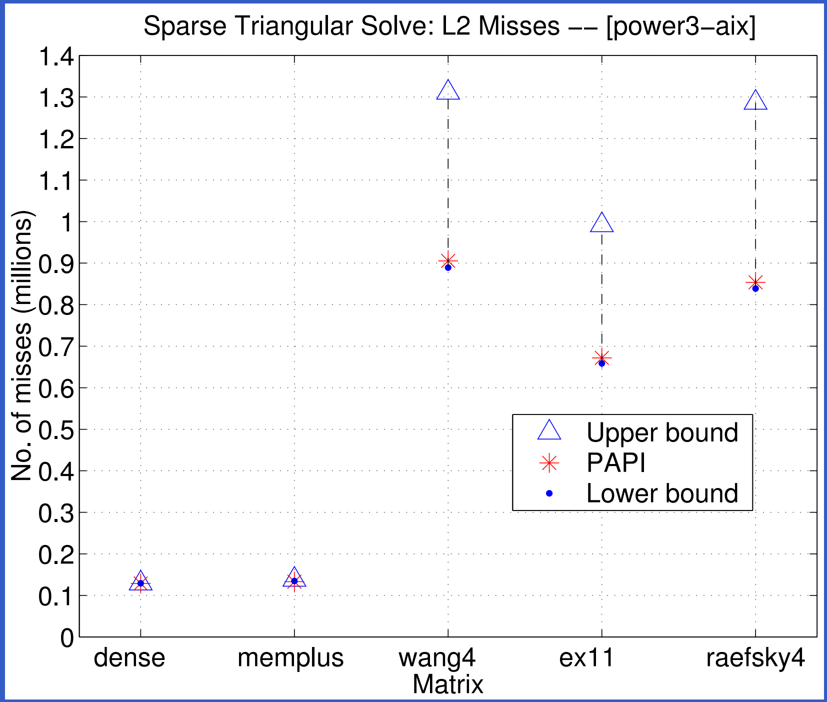
Miss Model Validation: Sun Ultra 2i



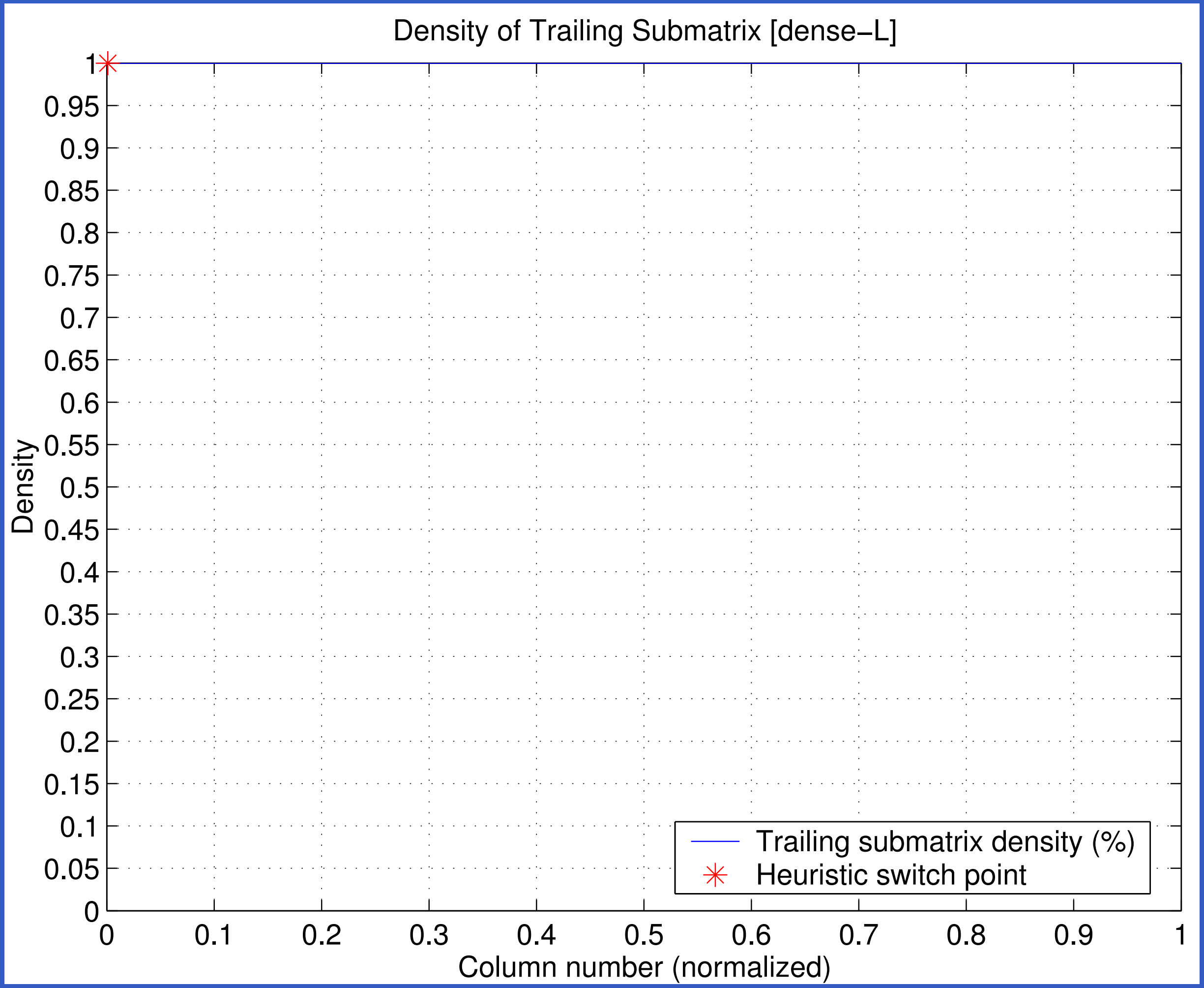


Miss Model Validation: IBM Power3

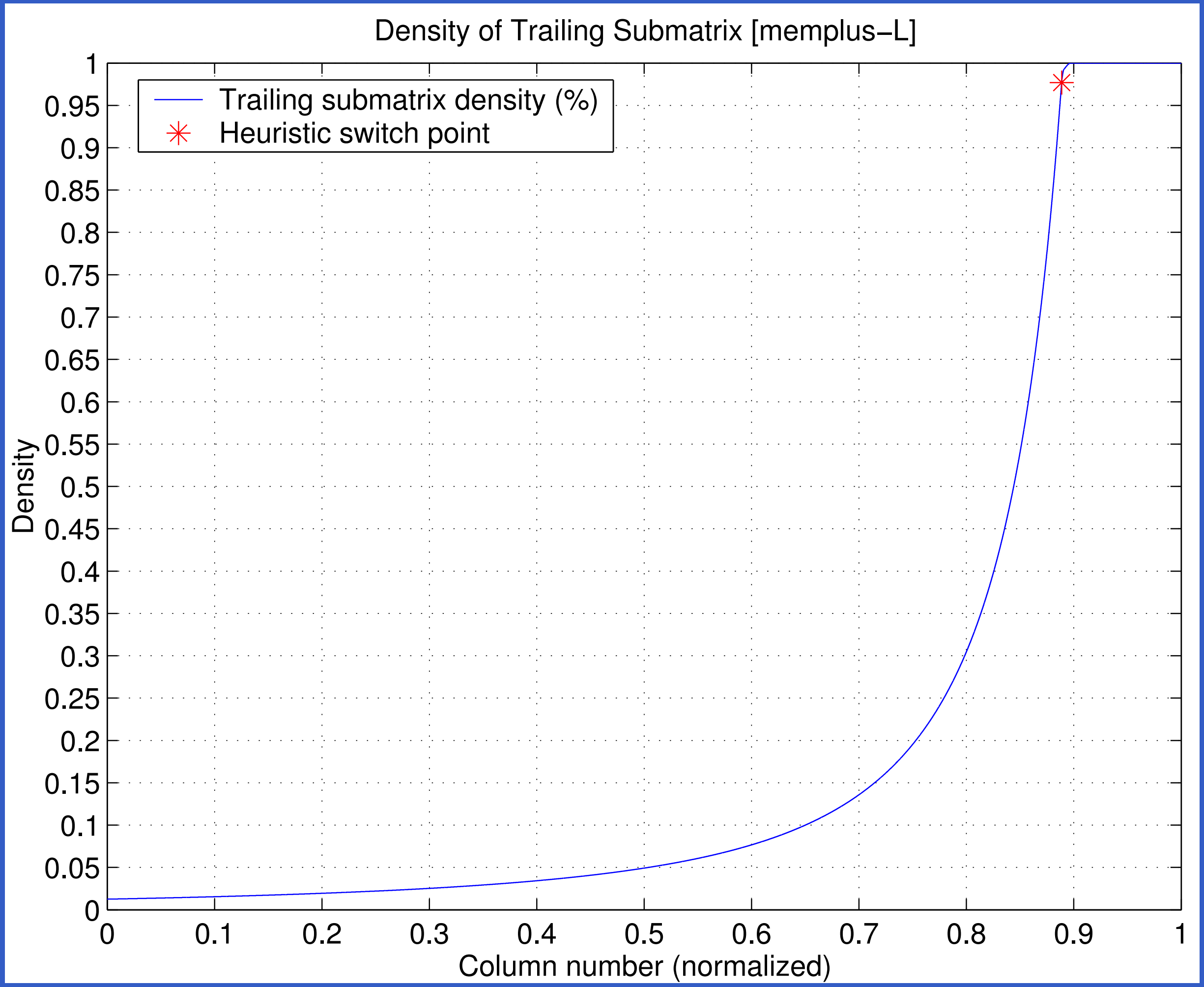




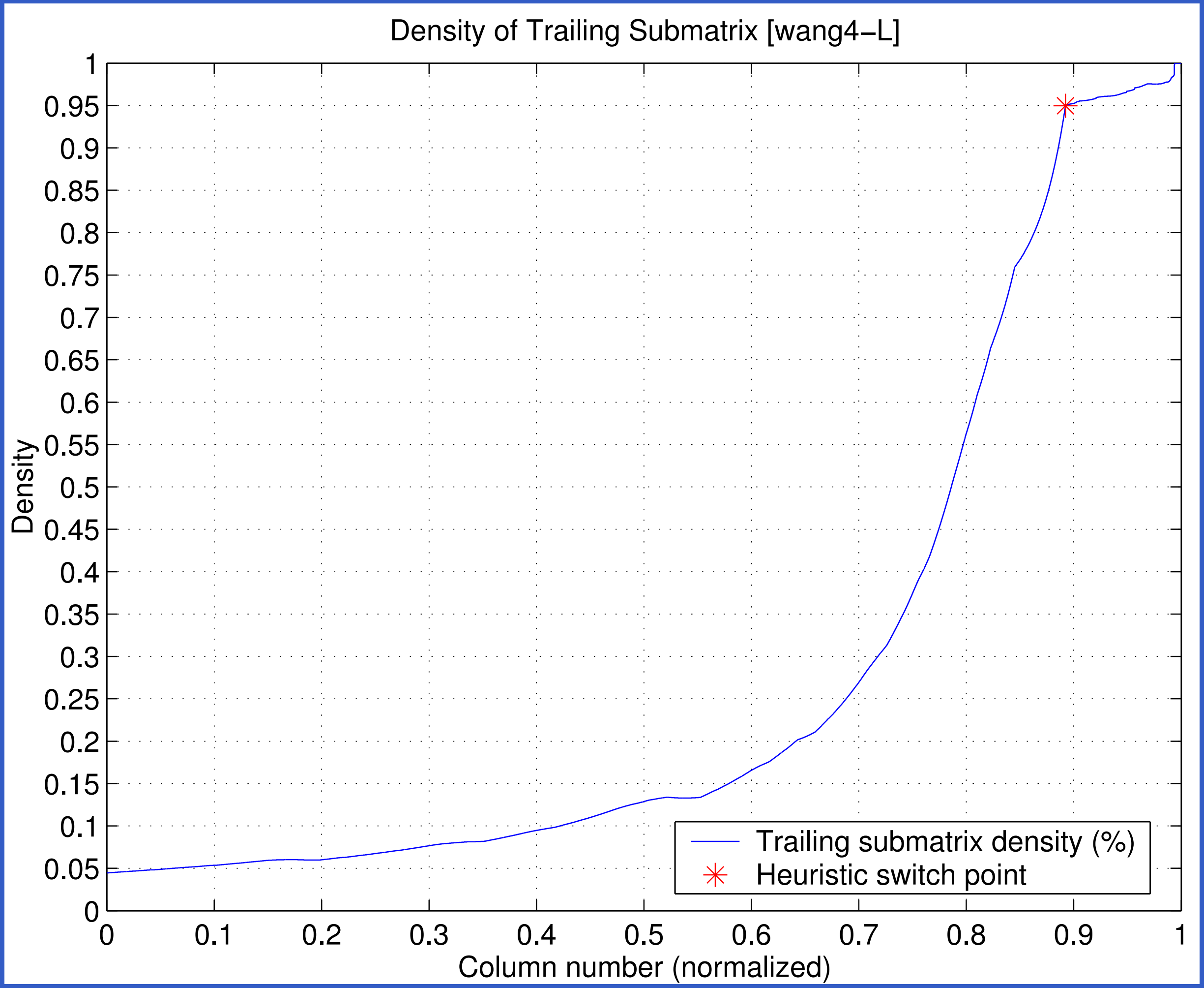
Dense Triangle Density: Dense Matrix



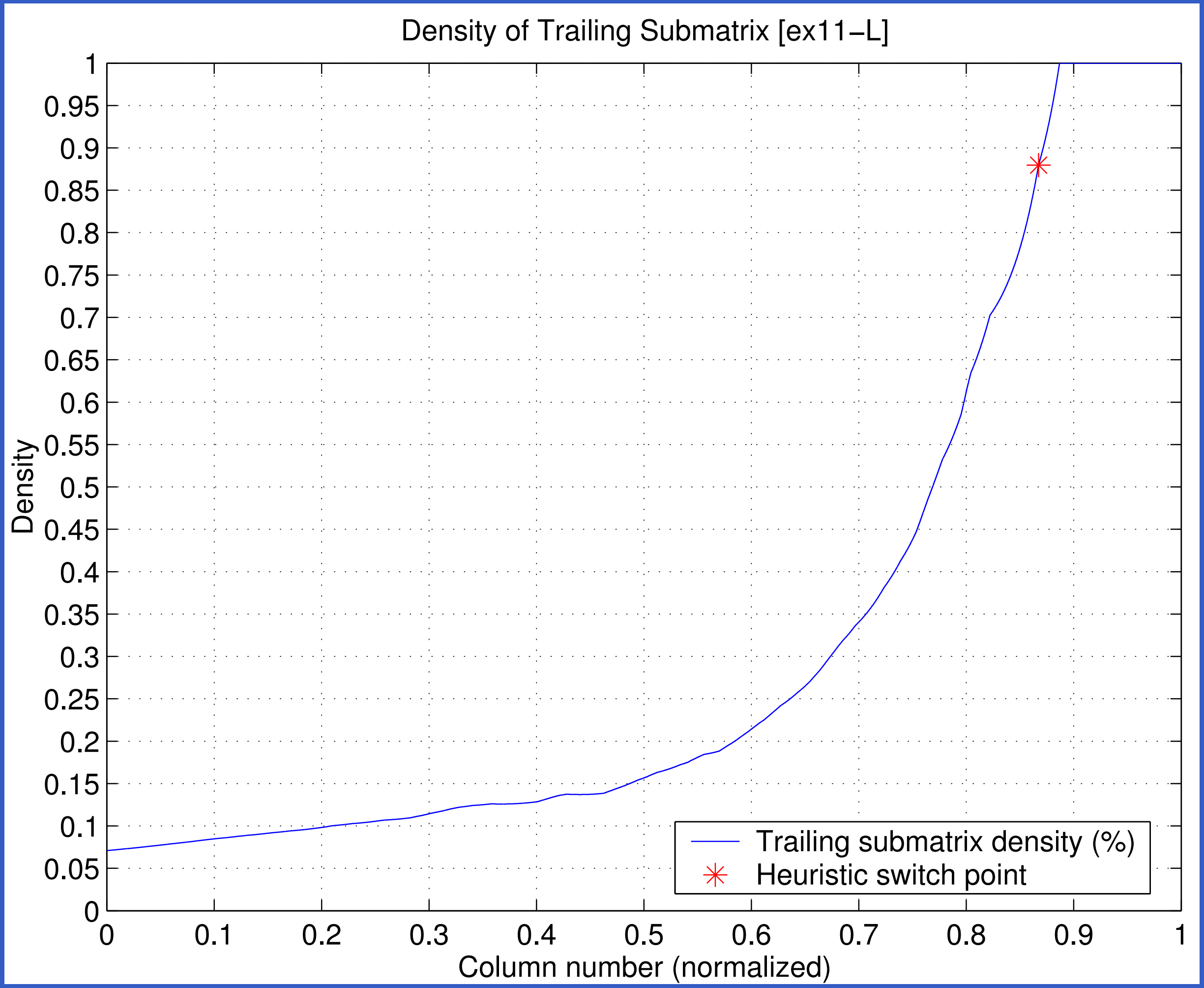
Dense Triangle Density: memplus



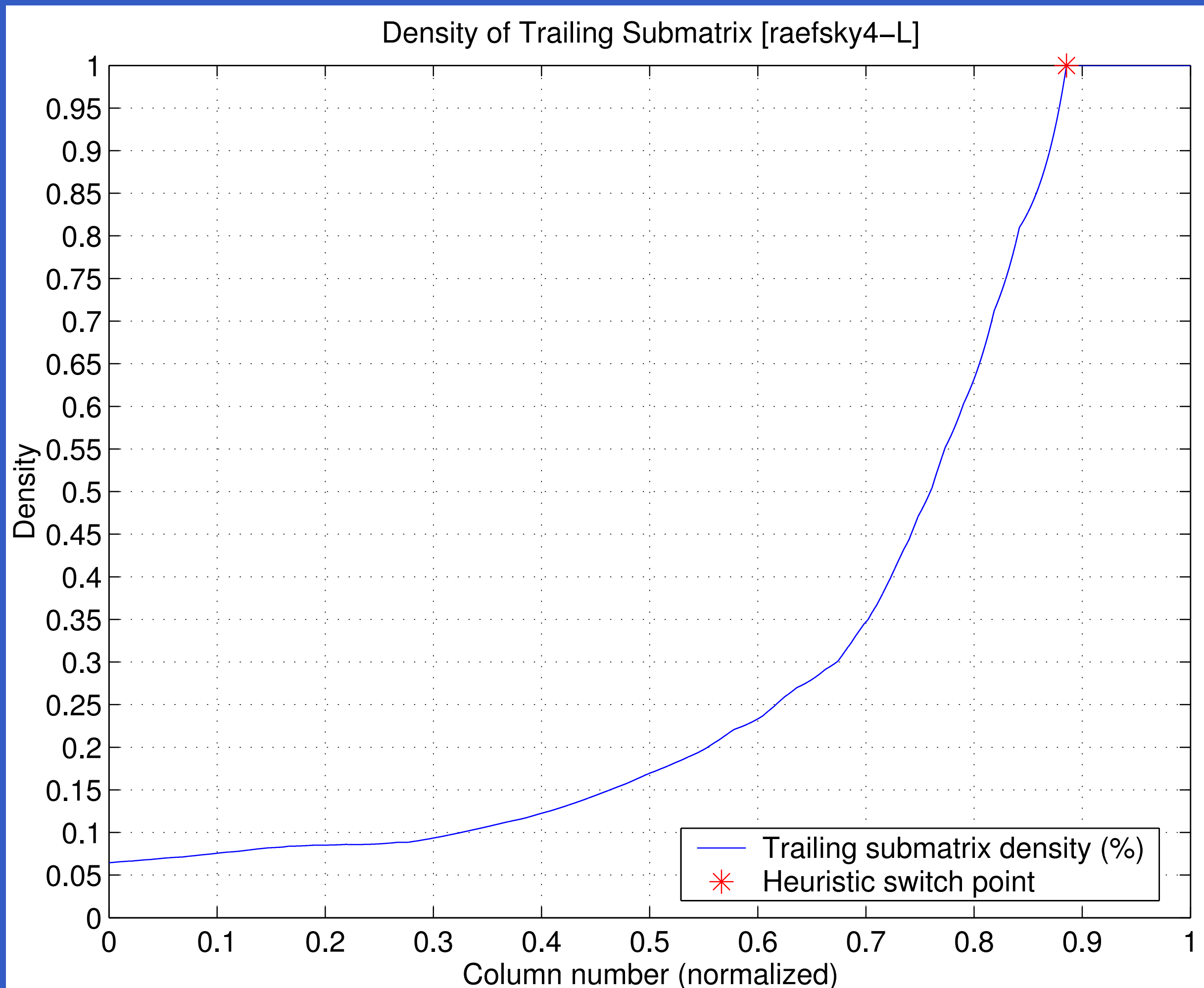
Dense Triangle Density: wang4



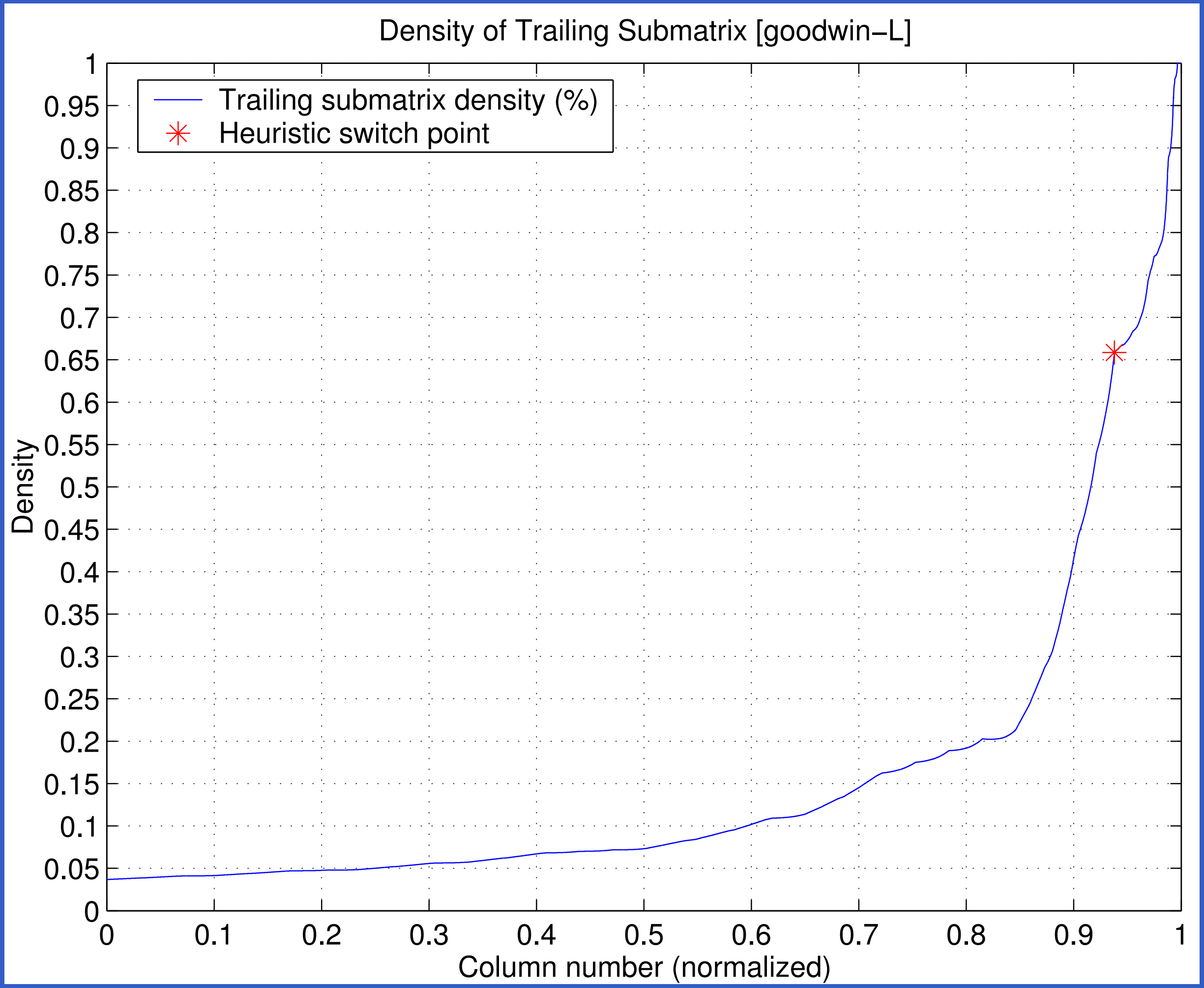
Dense Triangle Density: ex11



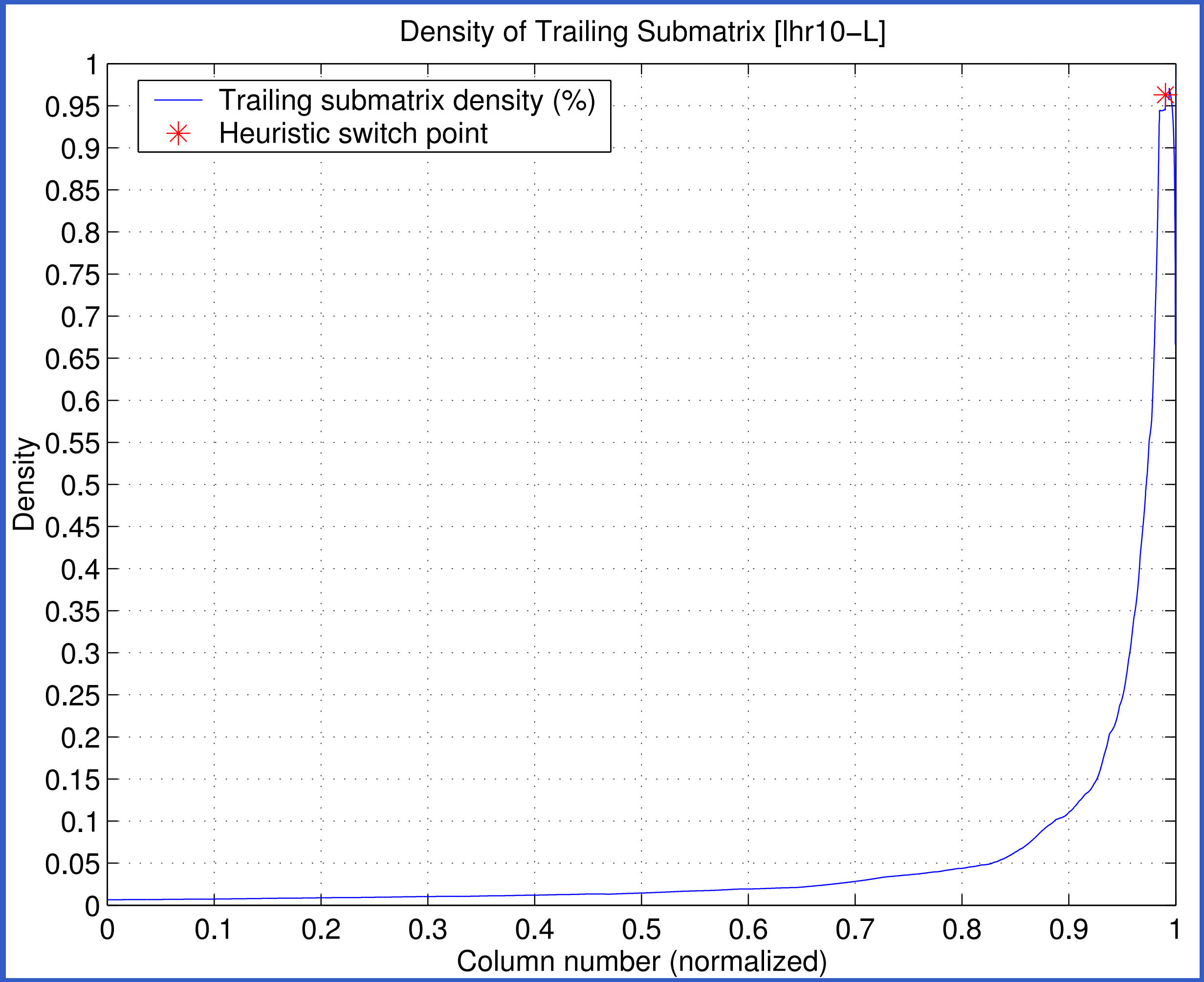
Dense Triangle Density: `raefsky4`



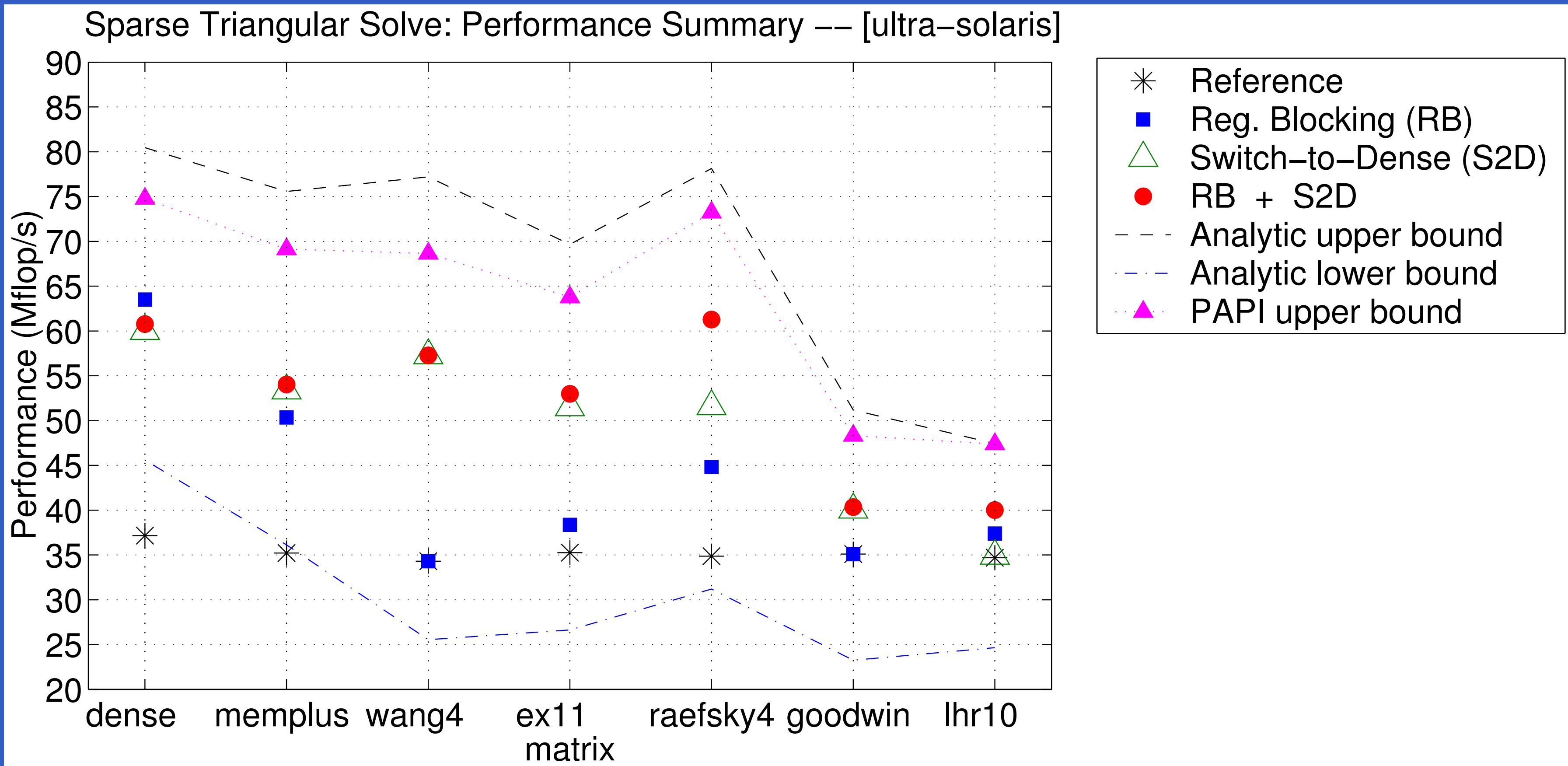
Dense Triangle Density: `goodwin`



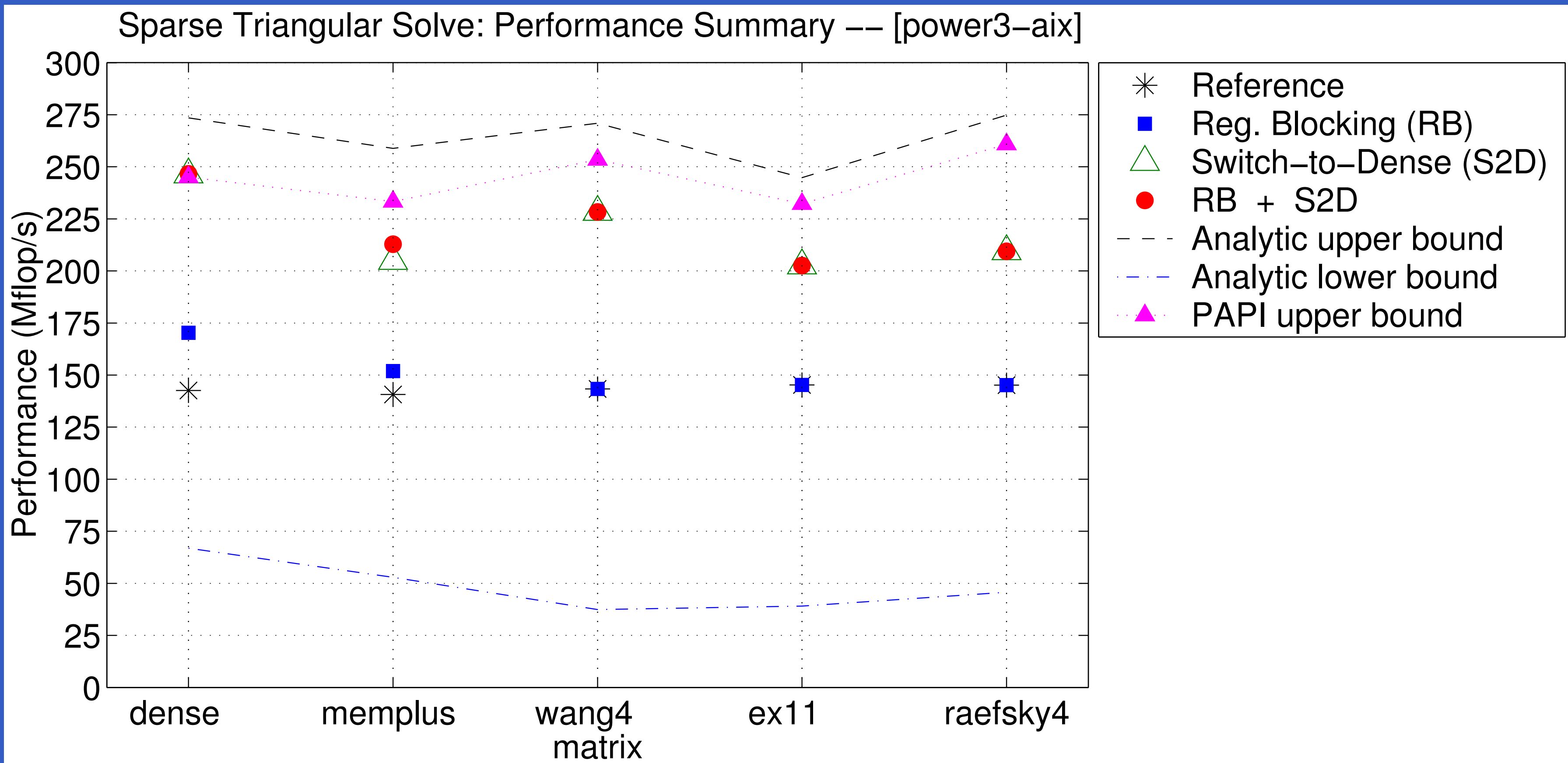
Dense Triangle Density: `lhr10`



Performance Results: Sun Ultra 2i



Performance Results: IBM Power3



References

[BACD97] J. Bilmes, K. Asanović, C.W. Chin, and J. Demmel. Optimizing matrix multiply using PHiPAC: a portable, high-performance, ANSI C coding methodology. In *Proceedings of the International Conference on Supercomputing*, Vienna, Austria, July 1997. ACM SIGARC. see `http://www.icsi.berkeley.edu/~bilmes/phipac`.

[BW99] Aart J. C. Bik and Harry A. G. Wijshoff. Automatic nonzero structure analysis. *SIAM Journal on Computing*, 28(5):1576–1587, 1999.

[FDZ99] Basilio B. Fraguela, Ramón Doallo, and Emilio L. Zapata. Memory hierarchy performance prediction for sparse blocked algorithms. *Parallel Processing Letters*, 9(3), March 1999.

[FJ98] Matteo Frigo and Stephen Johnson. FFTW: An adaptive software architecture for the FFT. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Seattle, Washington, May 1998.

[GGHvdG01] John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn. FLAME: Formal Linear Algebra Methods Environment. *ACM Transactions on Mathematical Software*, 27(4), December 2001.

[HPDR99] Dora Blanco Heras, Vicente Blanco Perez, Jose Carlos Cabaleiro Dominguez, and Francisco F. Rivera. Modeling and improving locality for irregular problems: sparse matrix-vector product on cache memories as a case study. In *HPCN Europe*, pages 201–210, 1999.

[Im00] Eun-Jin Im. *Optimizing the performance of sparse matrix-vector multiplication*. PhD thesis, University of California, Berkeley, May 2000.

[MMJ00] Dragan Mirkovic, Rishad Mahasoom, and Lennart Johnsson. An adaptive software library for fast fourier transforms. In *Proceedings of the International Conference on Supercomputing*, pages 215–224, Sante Fe, NM, May 2000.

[Neu98] T. Neubert. Anwendung von generativen Programmiertechniken am Beispiel der Matrixalgebra. Master's thesis, Technische Universität Chemnitz, 1998.

[NGLPJ96] J. J. Navarro, E. García, J. L. Larriba-Pey, and T. Juan. Algorithms for sparse matrix computations on high-performance workstations. In *Proceedings of the 10th ACM International Conference on Supercomputing*, pages 301–308, Philadelpha, PA, USA, May 1996.

[PSVM01] Markus Püschel, Bryan Singer, Manuela Veloso, and José M. F. Moura. Fast automatic generation of DSP algorithms. In *Proceedings of the International Conference on Computational Science*, volume 2073 of *LNCS*, pages 97–106, San Francisco, CA, May 2001. Springer.

[SL98] Jeremy G. Siek and Andrew Lumsdaine. A rational approach to portable high performance: the Basic Linear Algebra Instruction Set (BLAIS) and the Fixed Algorithm Size Template (fast) library. In *Proceedings of ECOOP*, 1998.

[Sto97] Paul Stodghill. *A Relational Approach to the Automatic Generation of Sequential Sparse Matrix Codes*. PhD thesis, Cornell University, August 1997.

[TJ92] O. Temam and W. Jalby. Characterizing the behavior of sparse algorithms on caches. In *Proceedings of Supercomputing '92*, 1992.

[Vel98] Todd Veldhuizen. Arrays in blitz++. In *Proceedings of ISCOPE*, volume 1505 of *LNCS*. Springer-Verlag, 1998.

[VFD01] Sathish S. Vadhiyar, Graham E. Fagg, and Jack J. Dongarra. Towards an accurate model for collective communications. In *Proceedings of the International Conference on Computational Science*, volume 2073 of *LNCS*, pages 41–50, San Francisco, CA, May 2001. Springer.

[WPD01] R. Clint Whaley, Antoine Petitet, and Jack Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1):3–25, 2001.

[WS97] James B. White and P. Sadayappan. On improving the performance of sparse matrix-vector multiplication. In *Proceedings of the International Conference on High-Performance Computing*, 1997.