

MAGMA 1.0 – LAPACK for GPUs

Stan Tomov

ICL Lunch Talk
January 07, 2011

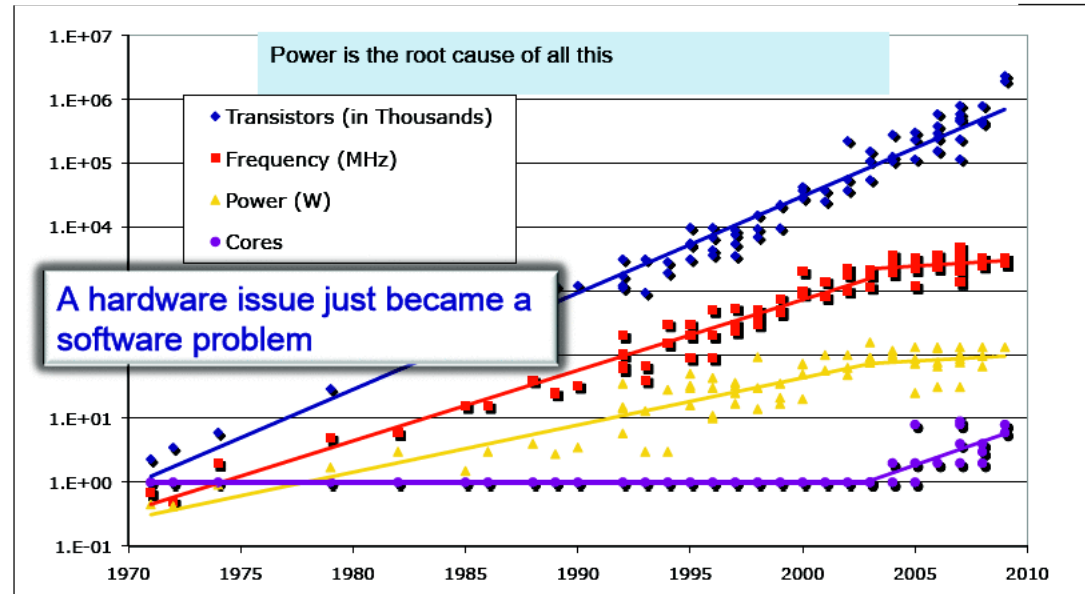


Outline

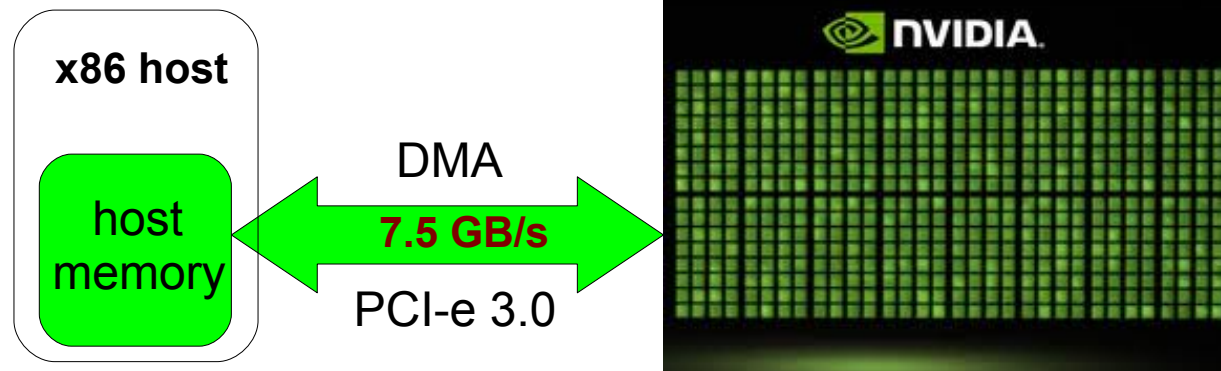
- **Project motivation**
 - ♦ Enable new architectures for efficient LA
 - ♦ Challenges of emerging architectures
- **MAGMA 1.0**
 - ♦ Outline
 - ♦ Results
- **Current & future work directions**
- **Conclusions**

Hardware Trends

- Power consumption and the move towards multicore
- Hybrid architectures
- GPU
- Hybrid GPU-based systems
 - ◆ CPU and GPU to get integrated (NVIDIA to make ARM CPU cores alongside GPUs)



Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanović
Slide from Kathy Yelick



Challenges to software

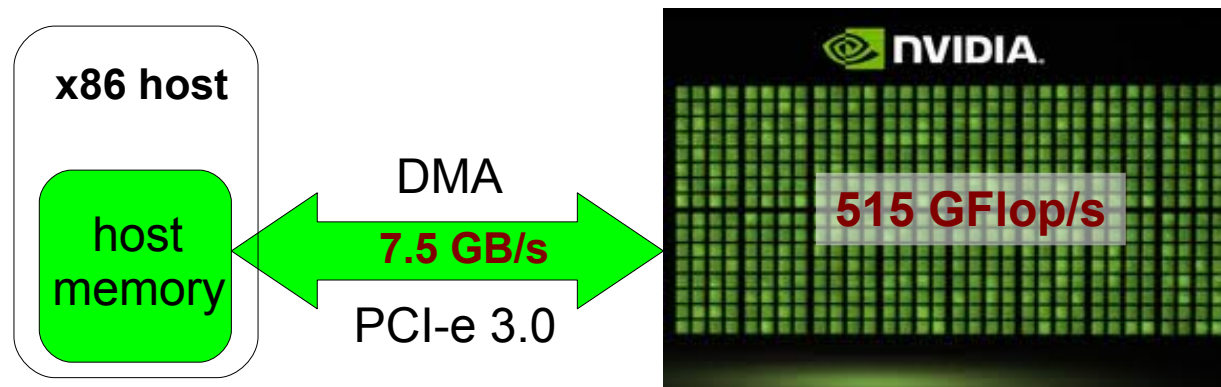
- Increase in parallelism

Tesla C2050 (Fermi): 448 CUDA cores @1.15 GHz
SP peak is 1075 GFlop/s, DP peak is 515 Gflop/s

- Increase in communication cost [vs computation]

Processor speed improves ~59% / year
memory bandwidth by only 23%

- Heterogeneity



Matrix Algebra on GPU and Multicore Architectures (MAGMA)

- **MAGMA**: a new generation linear algebra (LA) libraries to achieve the fastest possible time to an accurate solution on hybrid/heterogeneous architectures
Homepage: <http://icl.cs.utk.edu/magma/>
- **MAGMA & LAPACK**
 - **MAGMA** uses LAPACK (on the CPUs) and extends its functionality to hybrid systems (GPU support);
 - **MAGMA** is designed to be similar to LAPACK in functionality, data storage and interface
 - **MAGMA** leverages years of experience in developing open source LA software packages like LAPACK, ScaLAPACK, BLAS, ATLAS, and PLASMA
- **Support**
 - NSF, DOE
 - Microsoft, MathWorks
 - NVIDIA [**CUDA Center of Excellence at UTK** on the development of **Linear Algebra Libraries for CUDA-based Hybrid Architectures**]
- **MAGMA developers/collaborators**
 - University of Tennessee, **Knoxville**; University of California, **Berkeley**; University of Colorado, **Denver**
 - INRIA Bordeaux - Sud Ouest, France; University of Coimbra, Portugal

MAGMA 1.0

- 32 routines are developed (next)
 - ♦ Every routines is in 4 precisions (s/c/d/z, denoted by X)
 - ♦ There are 3 mixed precision routines (zc and ds, denoted by XX)
 - ♦ These are **hybrid algorithms**, extending the sequential **LAPACK algorithms** for hybrid systems (multicore + GPUs)
- Support is for single CUDA-enabled NVIDIA GPU, either Tesla or Fermi
- MAGMA BLAS
 - ♦ A subset of GPU BLAS, highly optimized for Tesla and Fermi GPUs

MAGMA 1.0

One-sided factorizations

1. Xgetrf	LU factorization; CPU interface
2. Xgetrf_gpu	LU factorization; GPU interface
3. Xgetrf_mc	LU factorization on multicore (no GPUs)
4. Xpotrf	Cholesky factorization; CPU interface
5. Xpotrf_gpu	Cholesky factorization; GPU interface
6. Xpotrf_mc	Cholesky factorization on multicore (no GPUs)
7. Xgeqrf	QR factorization; CPU interface
8. Xgeqrf_gpu	QR factorization; GPU interface; with T matrices stored
9. Xgeqrf2_gpu	QR factorization; GPU interface; without T matrices
10. Xgeqrf_mc	QR factorization on multicore (no GPUs)
11. Xgeqrf2	QR factorization; CPU interface
12. Xgeqlf	QL factorization; CPU interface
13. Xgelqf	LQ factorization; CPU interface

MAGMA 1.0

Linear solvers

14. Xgetrs_gpu	Work precision; using LU factorization; GPU interface
15. Xpotrs_gpu	Work precision; using Cholesky factorization; GPU interface
16. Xgels_gpu	Work precision LS; GPU interface
17. XXgetrs_gpu	Mixed precision iterative refinement solver; Using LU factorization; GPU interface
18. XXpotrs_gpu	Mixed precision iterative refinement solver; Using Cholesky factorization; GPU interface
19. XXgeqrsv_gpu	Mixed precision iterative refinement solver; Using QR on square matrix; GPU interface

MAGMA 1.0

Two-sided factorizations

20. Xgehrd	Reduction to upper Hessenberg form; with T matrices stored; CPU interface
21. Xgehrd2	Reduction to upper Hessenberg form; Without the T matrices stored; CPU interface
22. Xhetrd	Reduction to tridiagonal form; CPU interface
23. Xgebrd	Reduction to bidiagonal form; CPU interface

MAGMA 1.0

Generating/applying orthogonal matrices

24. Xungqr	Generates Q with orthogonal columns as the product of elementary reflectors (from Xgeqrf); CPU interface
25. Xungqr_gpu	Generates Q with orthogonal columns as the product of elementary reflectors (from Xgeqrf_gpu); GPU interface
26. Xunmtr	Multiplication with the orthogonal matrix, product of elementary reflectors from Xhetrd; CPU interface
27. Xunmqr	Multiplication with orthogonal matrix, product of elementary reflectors from Xgeqrf; CPU interface
28. Xunmqr_gpu	Multiplication with orthogonal matrix, product of elementary reflectors from Xgeqrf_gpu; GPU interface
29. Xunghr	Generates Q with orthogonal columns as the product of elementary reflectors (from Xgehrd); CPU interface

MAGMA 1.0

Eigen/singular-value solvers

30. Xgeev	Solves the non-symmetric eigenvalue problem; CPU interface
31. Xheevd	Solves the Hermitian eigenvalue problem; Uses devide and conquer; CPU interface
32. Xgesvd	SVD; CPU interface

- Currently, these routines have GPU-acceleration for the
 - two-sided factorizations used and the
 - Orthogonal transformation related to them
(matrix generation/application from slide 9)

MAGMA BLAS

Level 2 BLAS

1. Xgemv_tesla	General matrix-vector product for Tesla
2. Xgemv_fermi	General matrix-vector product for Fermi
3. Xsymv_tesla	Symmetric matrix-vector product for Tesla
4. Xsymv_fermi	Symmetric matrix-vector product for Fermi

MAGMA BLAS

Level 3 BLAS

5. Xgemm_tesla	General matrix-matrix product for Tesla
6. Xgemm_fermi	General matrix-matrix product for Fermi
7. Xtrsm_tesla	Solves a triangular matrix problem on Tesla
8. Xtrsm_fermi	Solves a triangular matrix problem on Fermi
9. Xsyrk_tesla	Symmetric rank k update for Tesla
10. Xsyr2k_tesla	Symmetric rank 2k update for Tesla

- Our GEMMs for Fermi are used in CUBLAS 3.2

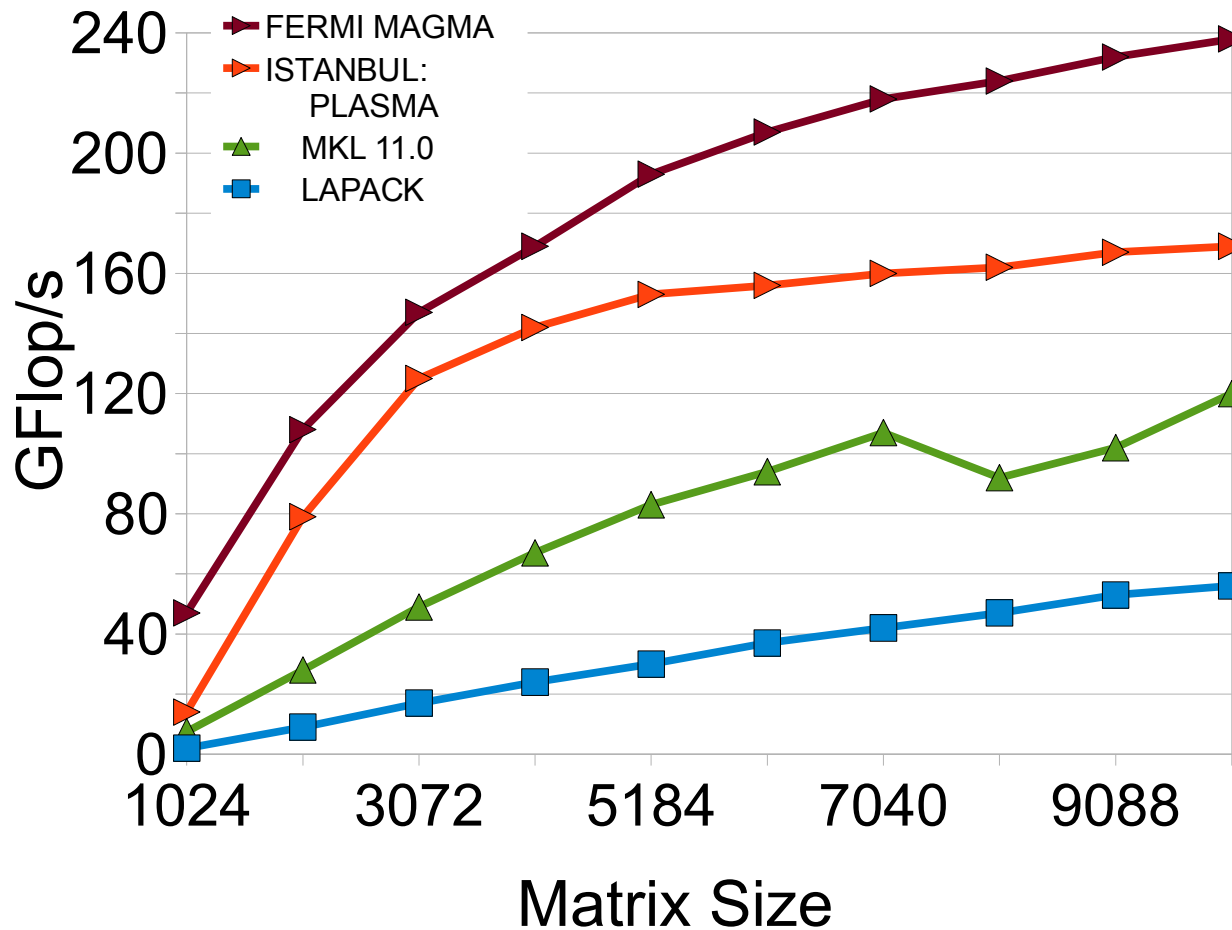
MAGMA BLAS

Other routines

11. Xswap	LU factorization; CPU interface
12. Xlacpy	LU factorization; GPU interface
13. Xlange	LU factorization on multicore (no GPUs)
14. Xlanhe	Cholesky factorization; CPU interface
15. Xtranspose	Cholesky factorization; GPU interface
16. Xinplace_transpose	Cholesky factorization on multicore (no GPUs)
17. Xpermute	QR factorization; CPU interface
18. Xauxiliary	QR factorization; GPU interface; with T matrices stored

Results – one sided factorizations

LU Factorization in double precision

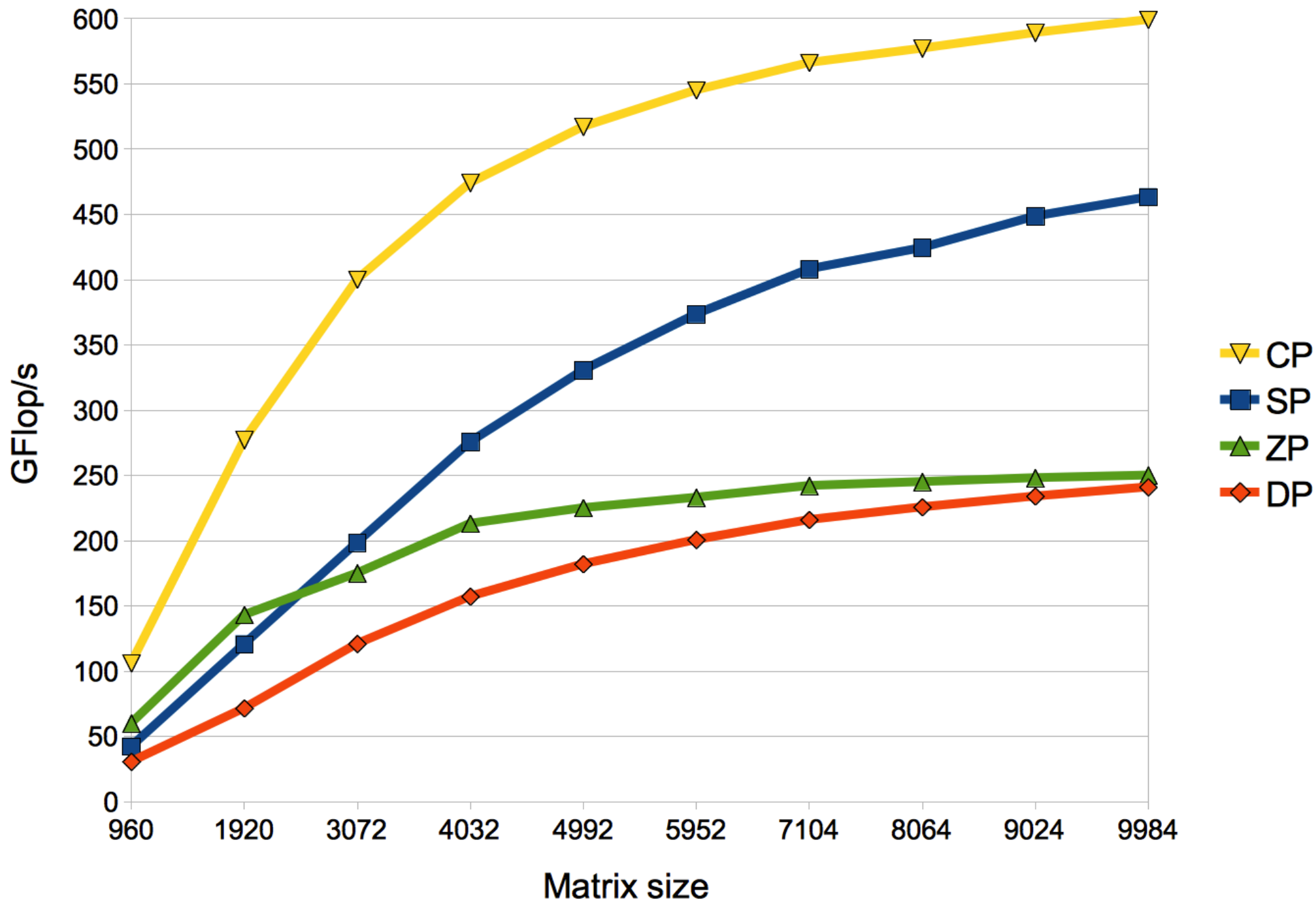


FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s

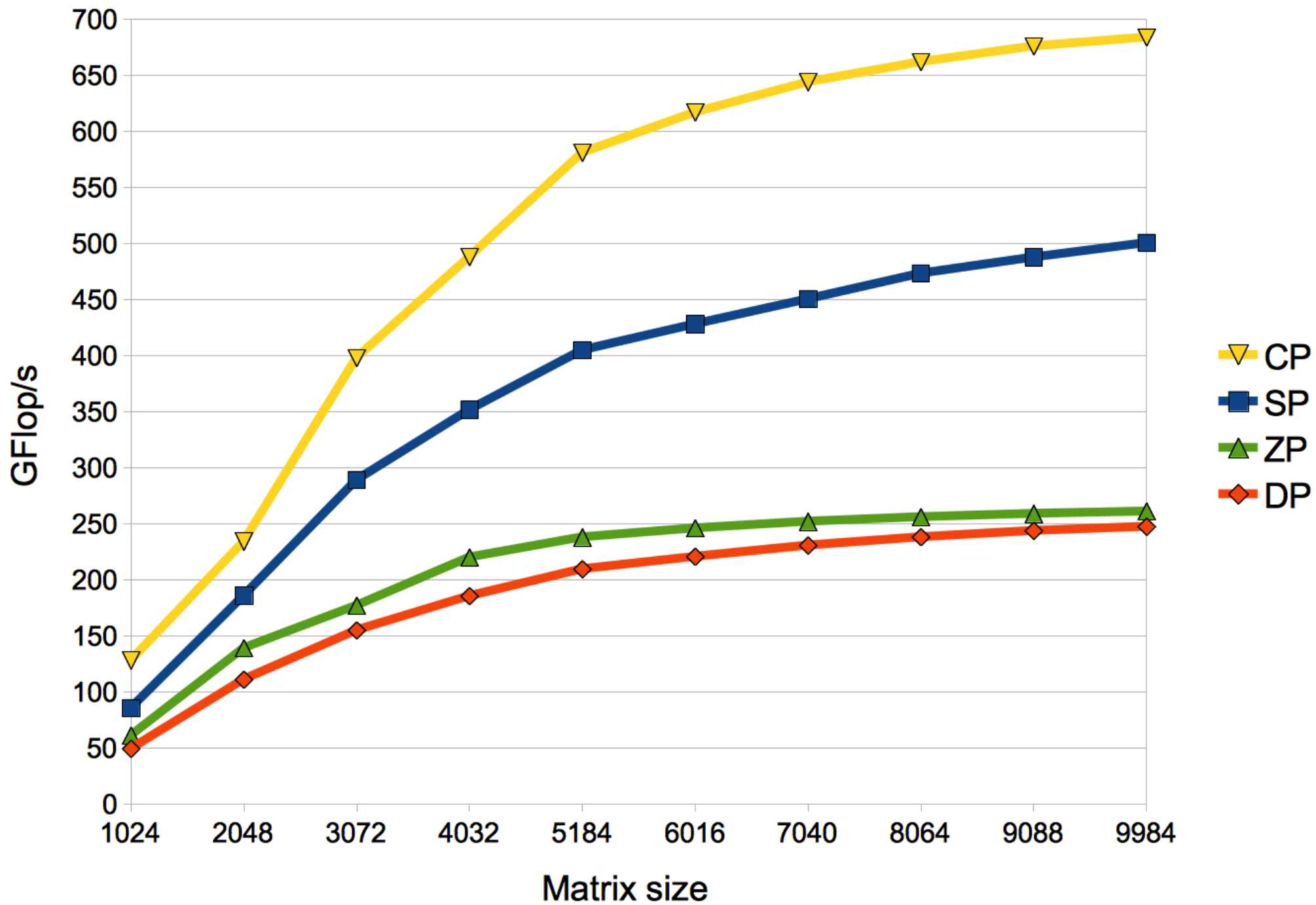
ISTANBUL AMD 8 socket 6 core (48 cores) @2.8GHz
SP/DP peak is 1075 / 538 GFlop/s

- Similar results for Cholesky & QR
- Fast solvers (several innovations)
 - in working precision, and
 - mixed-precision iter. refinement based on the one-sided factor.

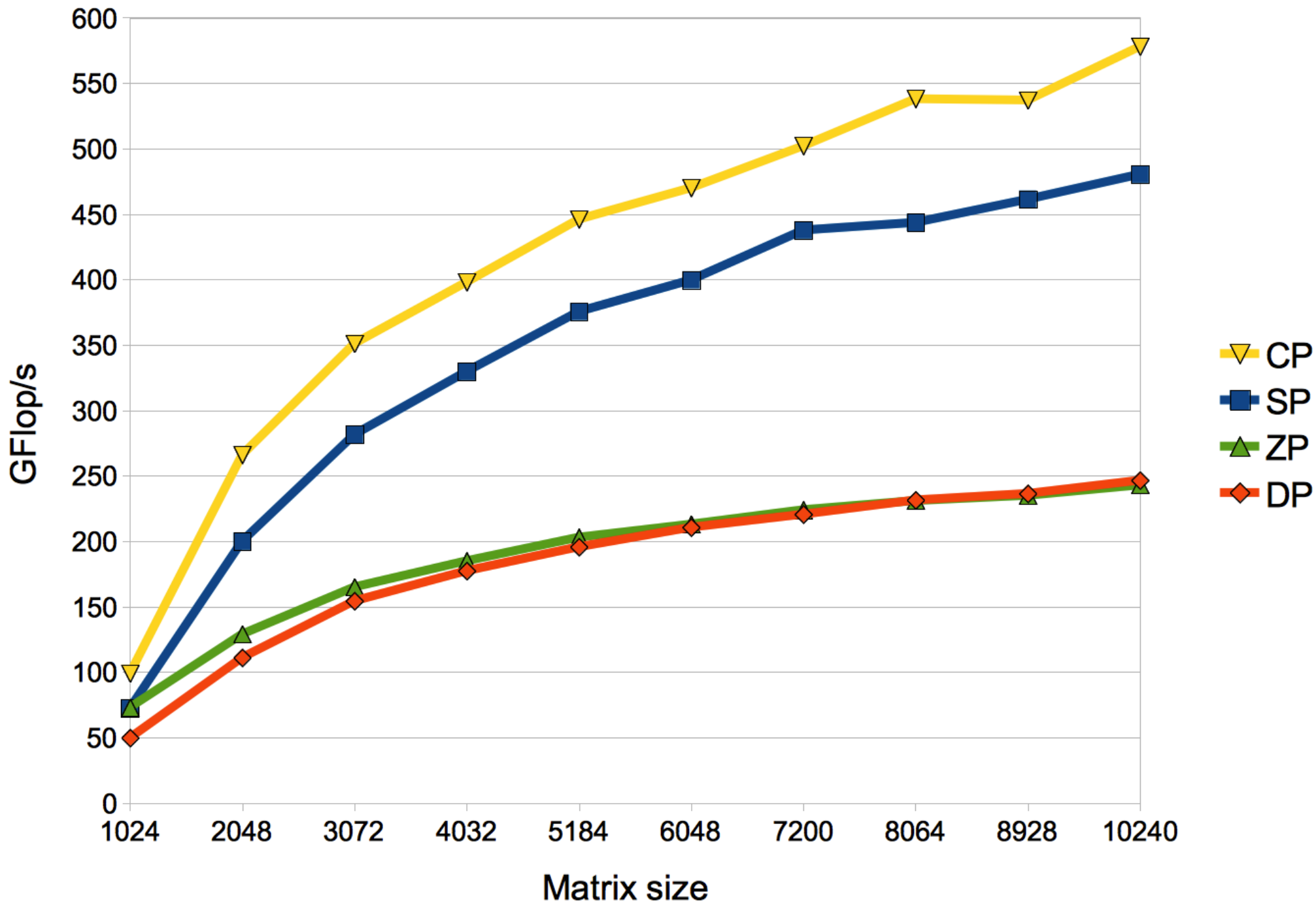
Performance of MAGMA LU on Fermi (C2050)



Performance of MAGMA QR on Fermi (C2050)

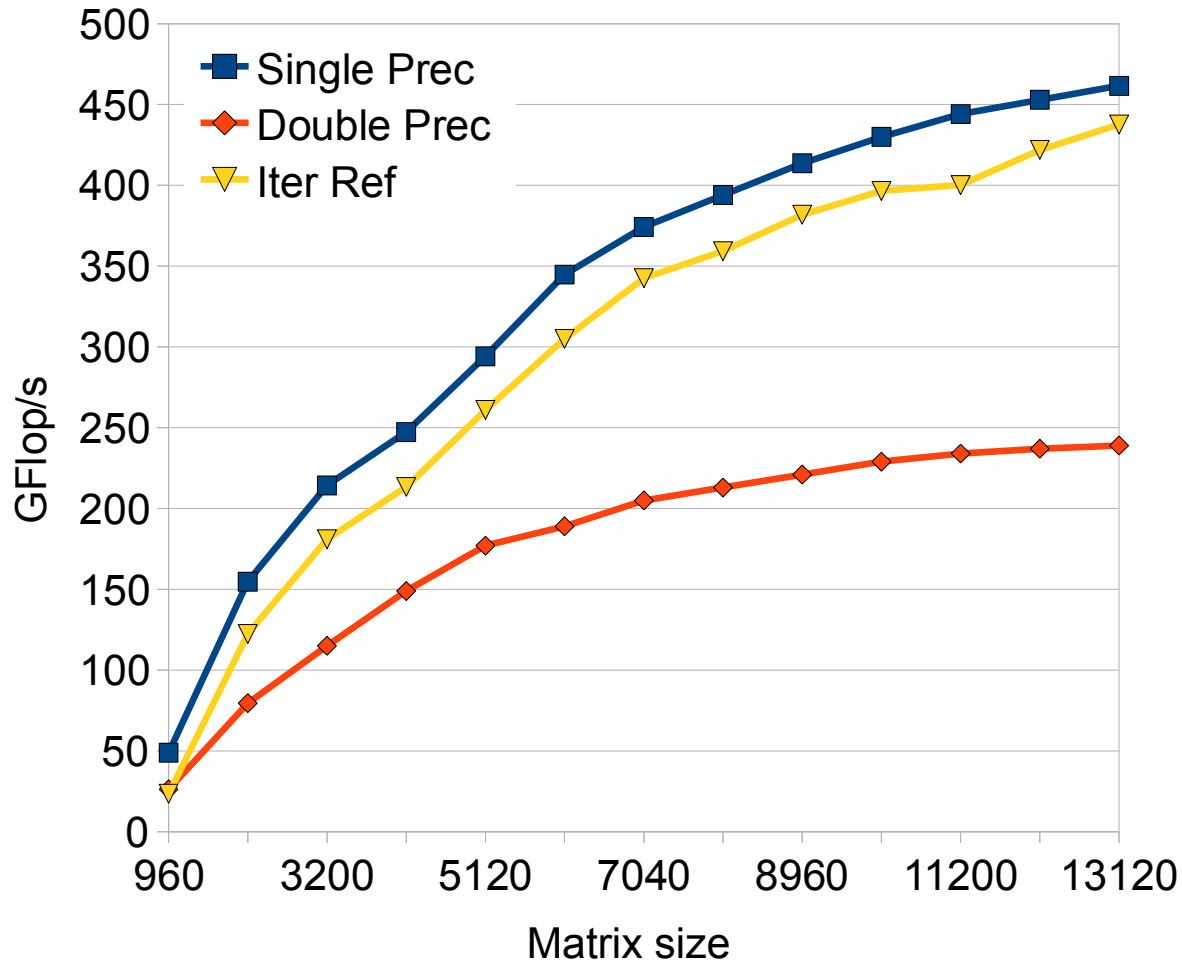


Performance of MAGMA Cholesky on Fermi (C2050)



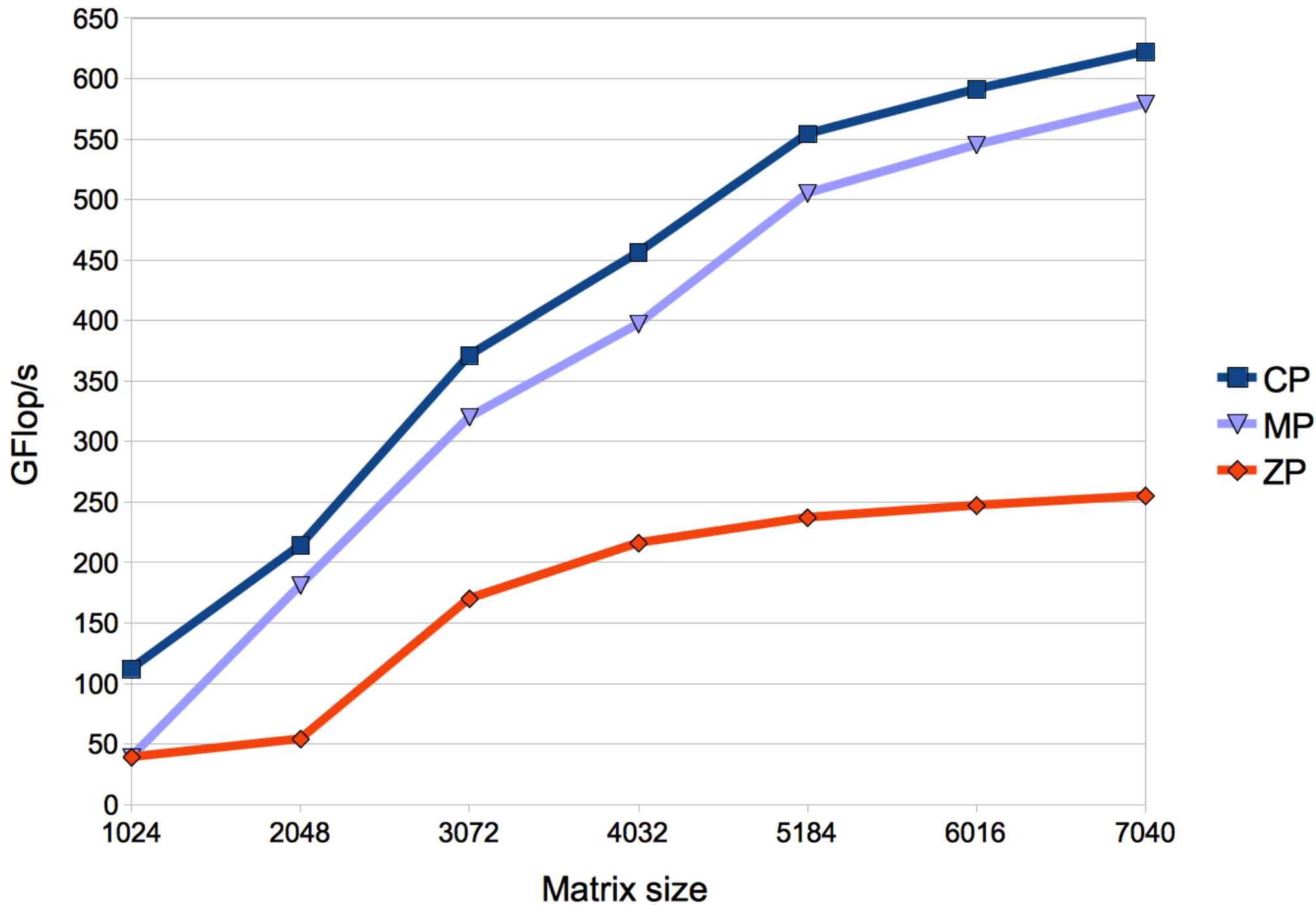
Results – linear solvers

MAGMA LU-based solvers on Fermi (C2050)



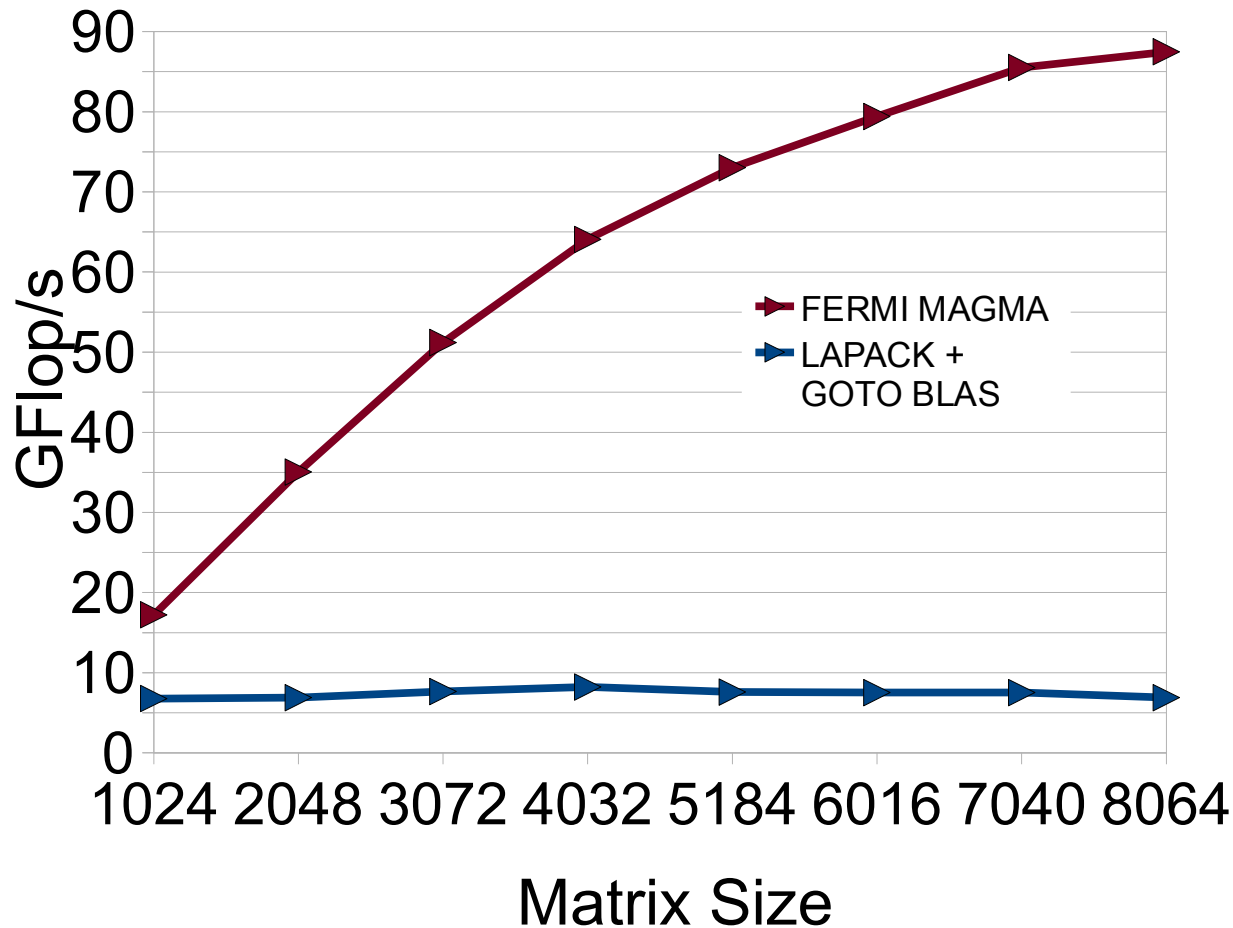
FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s

Performance of MAGMA QR LS Solvers on Fermi (C2050)



Results – two sided factorizations

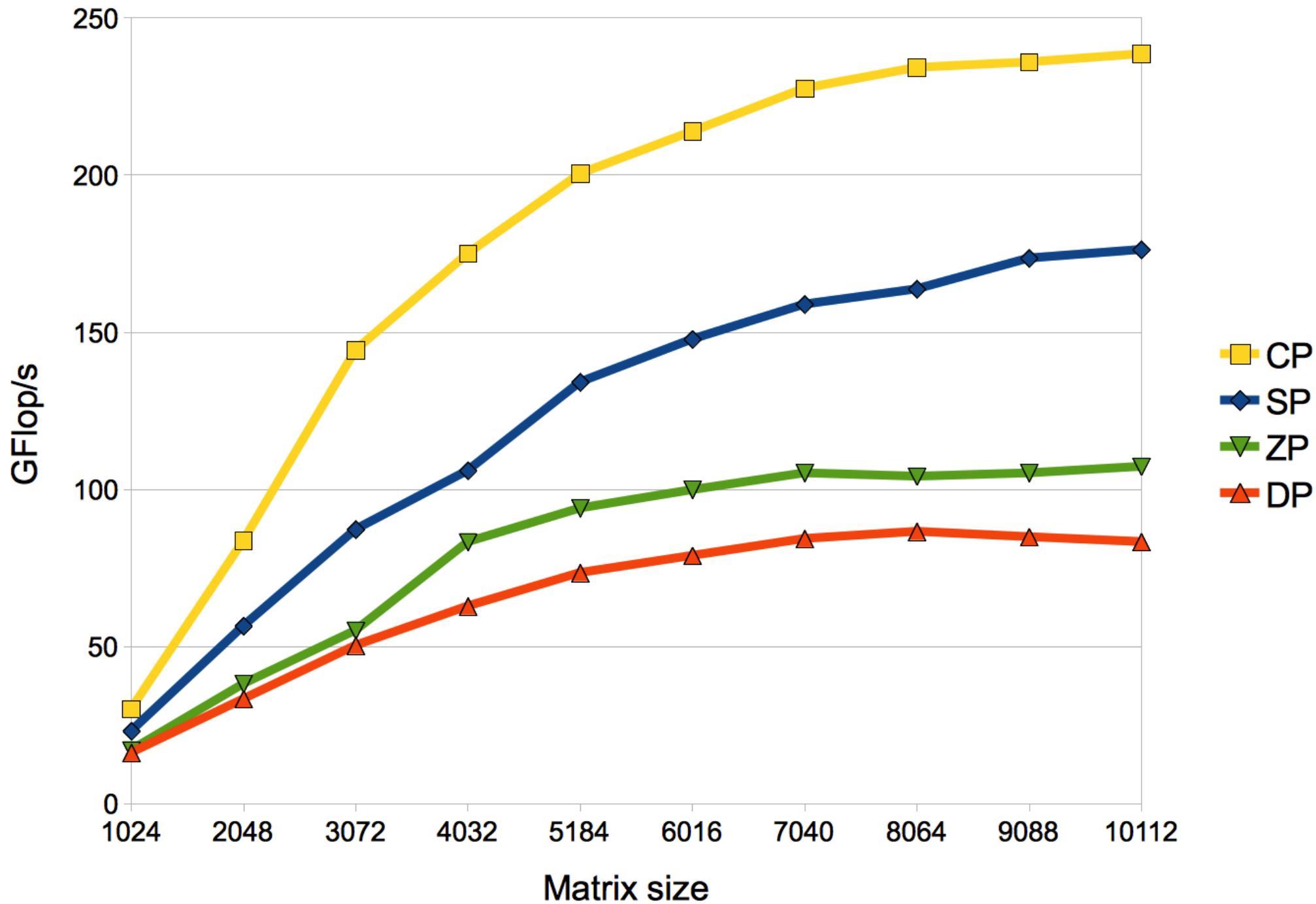
Hessenberg Factorization in double precision
[for the general eigenvalue problem]



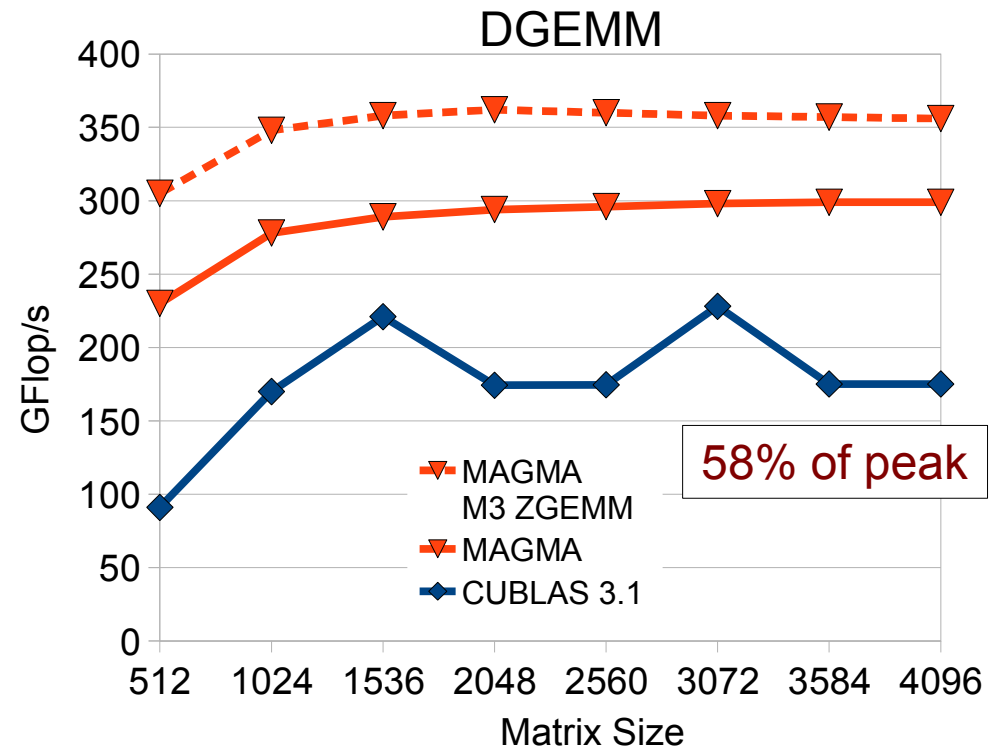
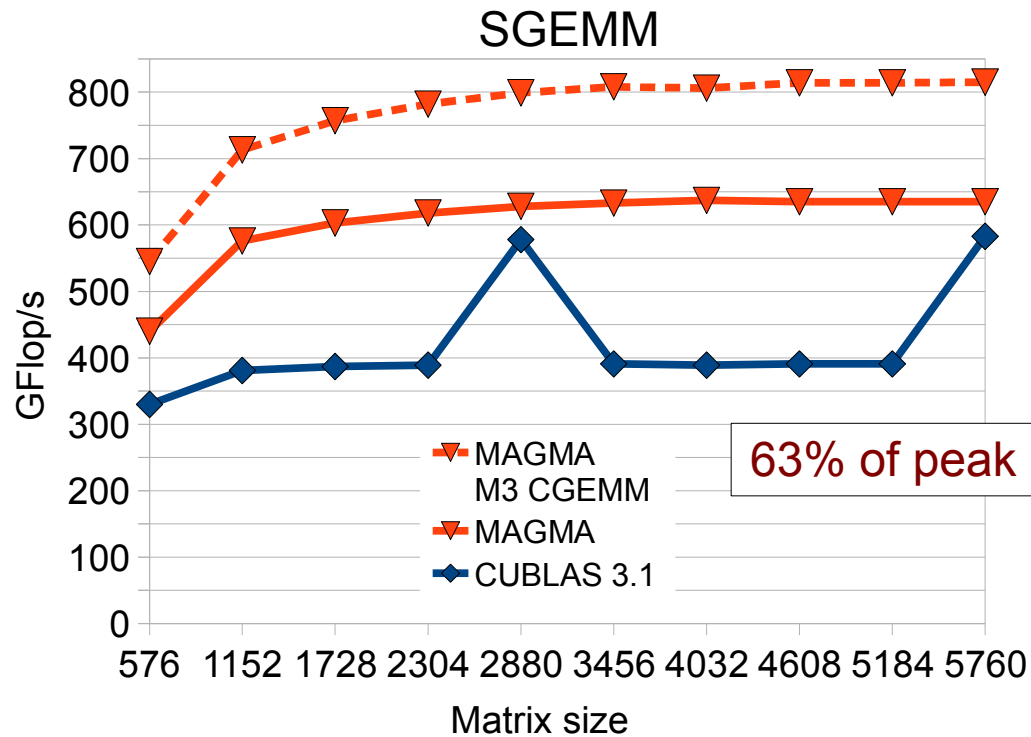
FERMI	Tesla C2050: 448 CUDA cores @ 1.15GHz SP/DP peak is 1030 / 515 Gflop/s [system cost ~ \$3,000]
ISTANBUL	AMD 8 socket 6 core (48 cores) @2.8GHz SP/DP peak is 1075 / 538 Gflop/s [system cost ~ \$30,000]

- Similar accelerations for the bidiagonal factorization [for SVD] & tridiagonal factorization [for the symmetric eigenvalue problem]
- Similar acceleration (exceeding 10x) compared to other top-of-the-line multicore systems (including Nehalem-based) and libraries (including MKL, ACML)

Performance of MAGMA Hessenberg on Fermi (C2050)

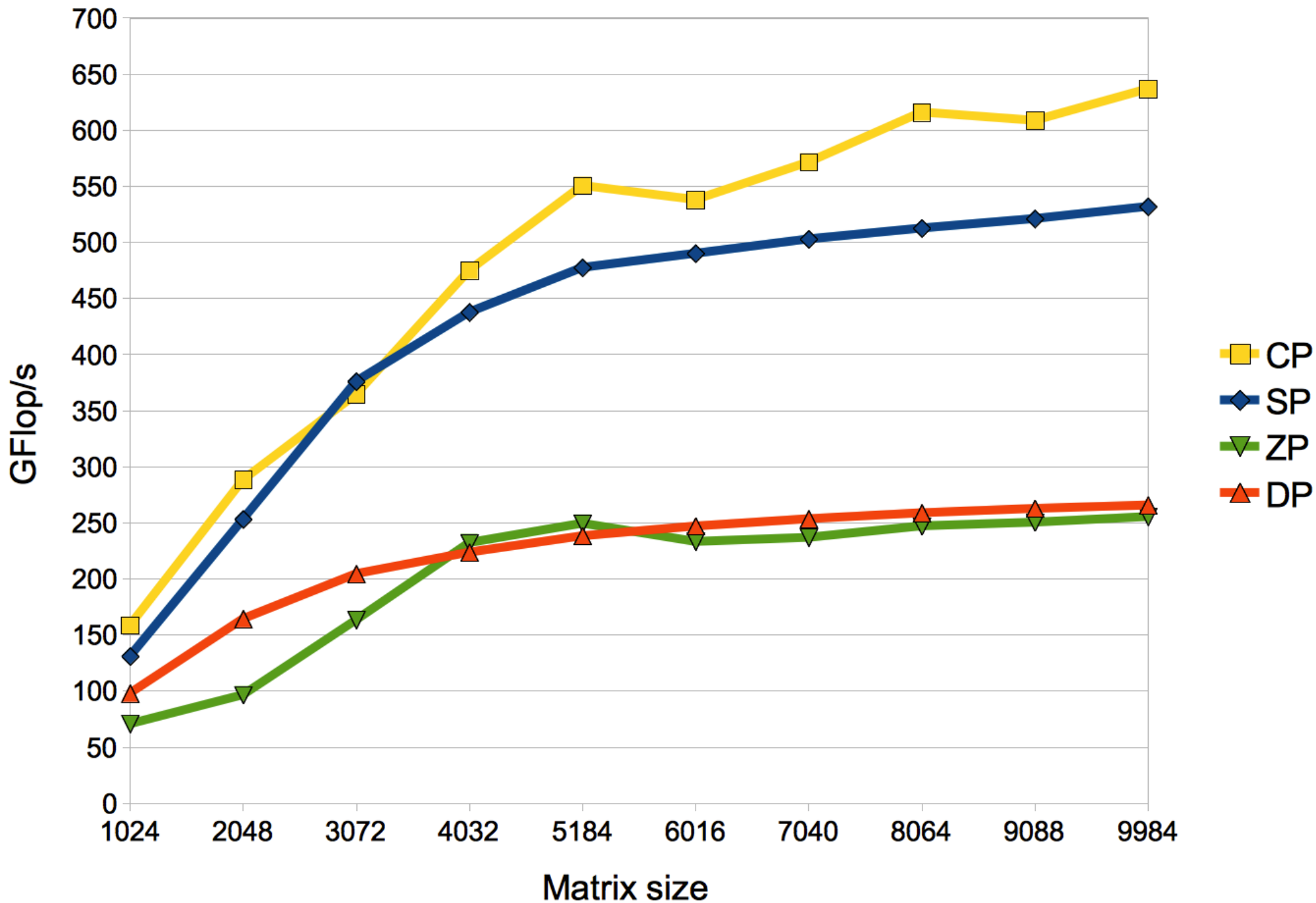


Results - BLAS

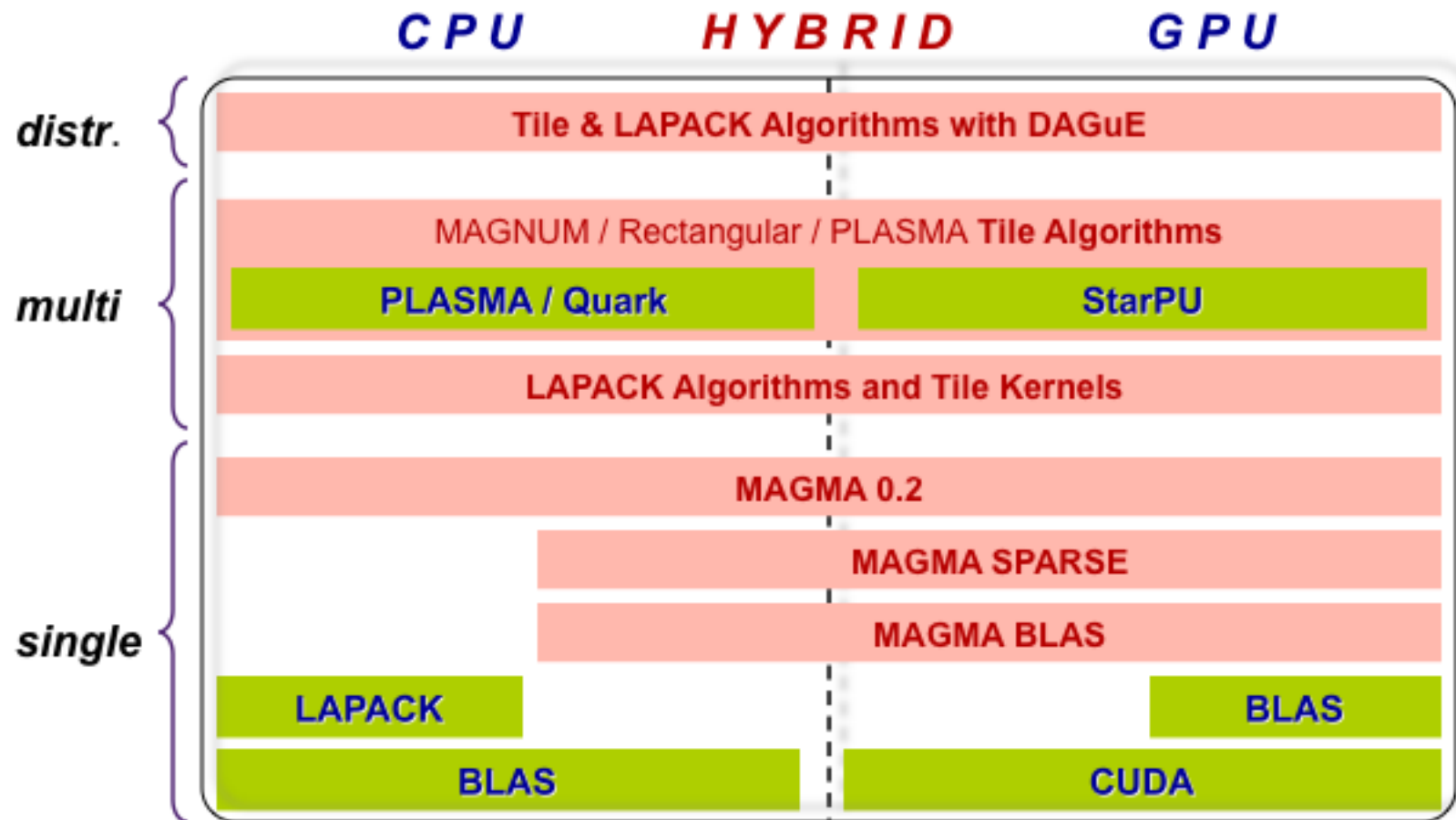


Tesla C2050 (Fermi): 448 CUDA cores @ 1.15GHz, theoretical SP peak is 1.03 Tflop/s, DP peak 515 GFlop/s

Performance of MAGMA xUNGQR on Fermi (C2050)



MAGMA Software Stack



Linux, Windows, Mac OS X | *C/C++, Fortran* | *Matlab, Python*

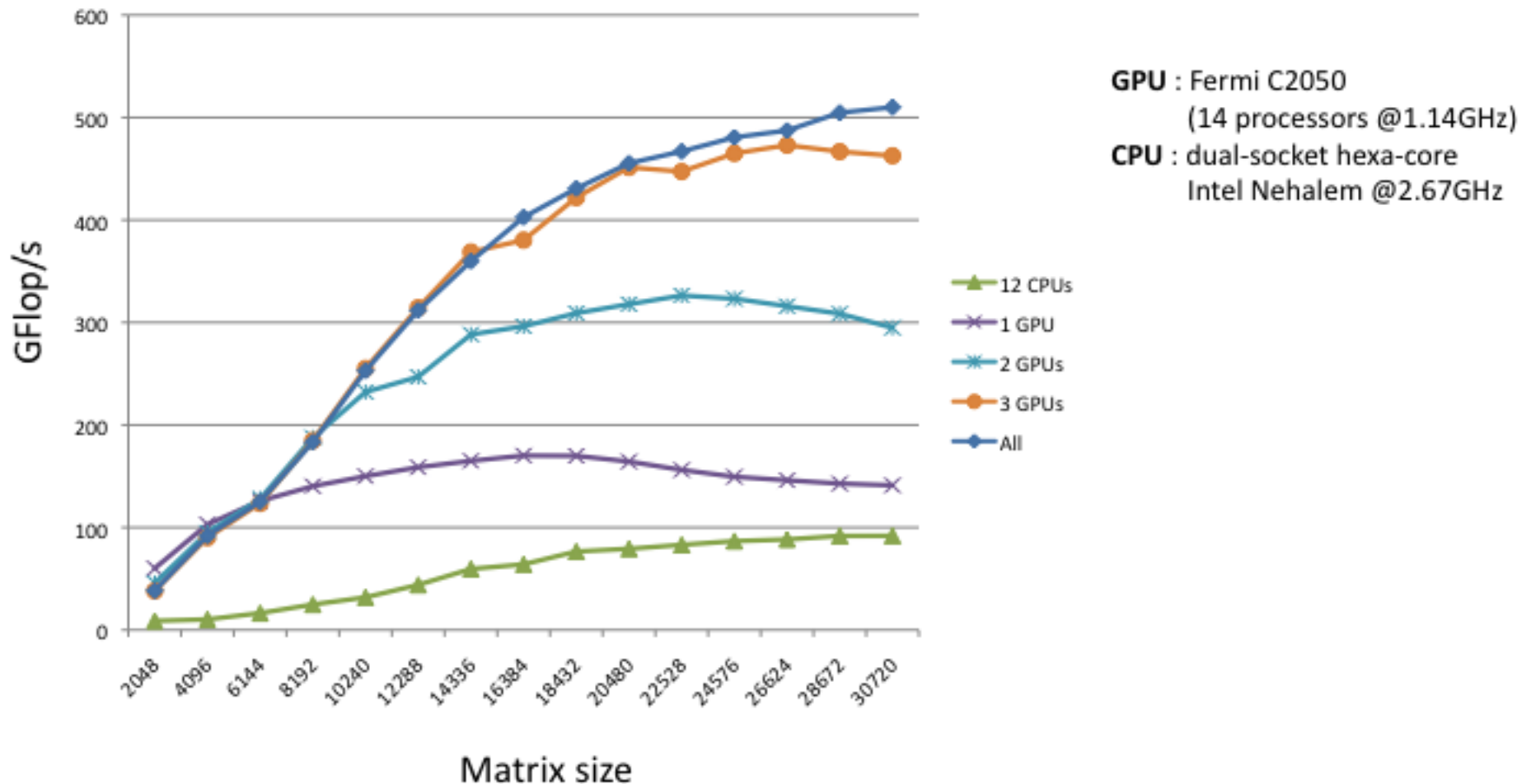
Current and future work

- Software engineering, improvements, and bug fixes
- MAGMA BLAS for Fermi
- Tuning
- Portability to other hardware
 - ♦ Through OpenCL and auto-tuning
- Algorithms for multiGPU and multicore
 - ♦ Various Hybrid Tile algorithms
 - ♦ Hybrid LAPACK algorithms
 - ♦ Distributed Hybrid Tile and LAPACK algorithms

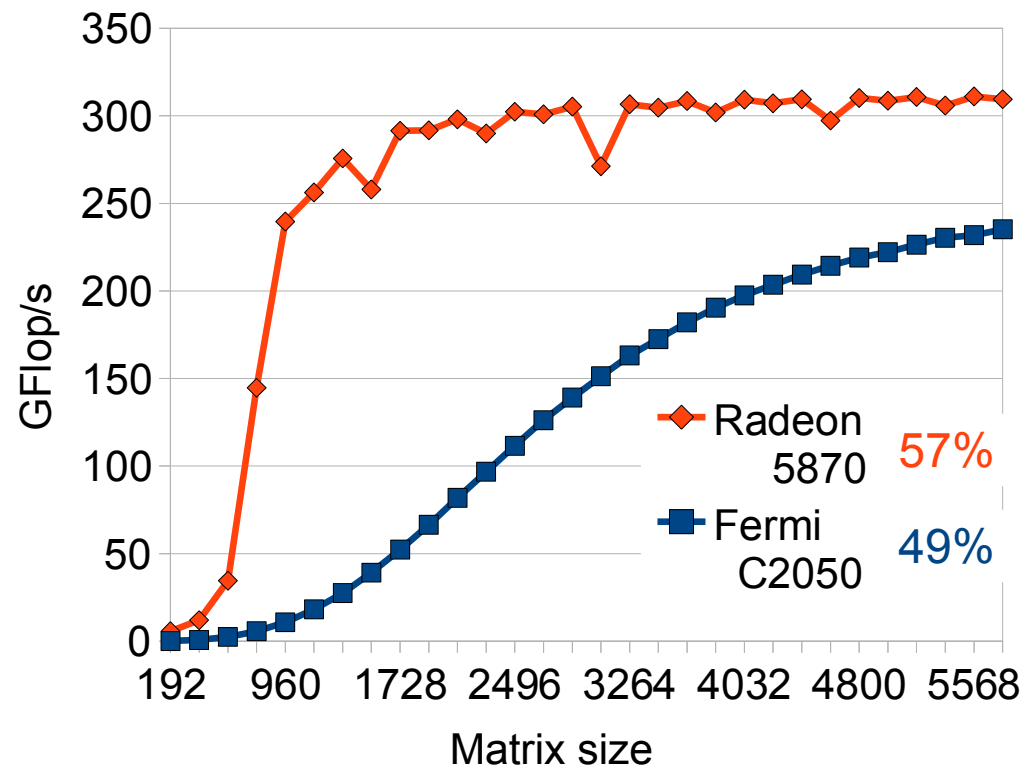
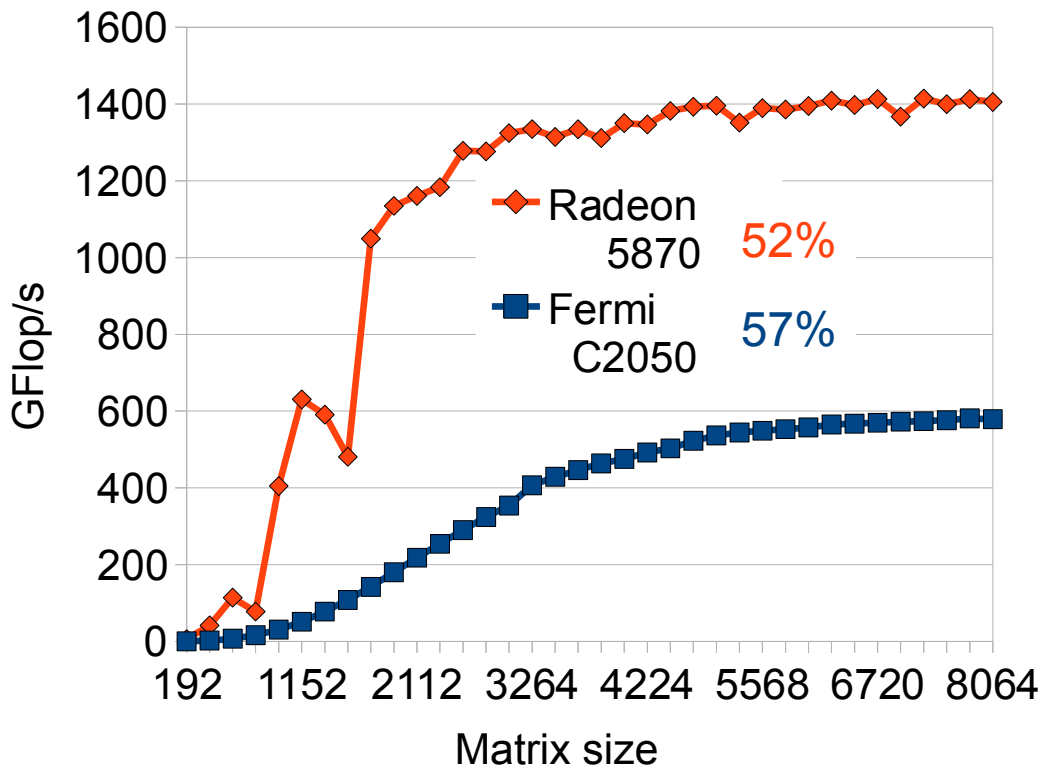
Magnum tile algorithms with StarPU

- We have finished the LU, QR, and Cholesky factorizations on a node (with Hatem, Mathieu and the StarPU team including Cedric and Emmanuel)

Performance of tile LU factorization in double precision arithmetic



Research on Portability Across Platforms through OpenCL



- Performance portability of OpenCL implementations – through **auto-tuning**
 - ◆ Collecting best kernel versions
 - ◆ Generating multiple kernel versions to explore the kernel parameter space
 - ◆ Find best performing kernel versions on particular architecture using empirical-based search enhanced with heuristic models

Conclusions

- ***Linear and eigenvalue solvers can be significantly accelerated on systems of multicore and GPU architectures***
- Many-core architectures with accelerators (e.g., GPUs) are the future of high performance scientific computing
- Challenge: Fundamental libraries will need to be redesigned/rewritten to take advantage of the emerging many-core architectures