Piotr Luszczek

Numerical Libraries for Dense Linear Algebra

Innovative Computing Laboratory's Libraries

- MAGMA
  - Matrix Algebra for GPUs and Multicore Architectures
- PLASMA
  - Parallel Linear Algebra Software for Multicore Architectures

Academic Collaborations

The University of Manchester

University of Colorado Denver

informatics mathematics

KAUST

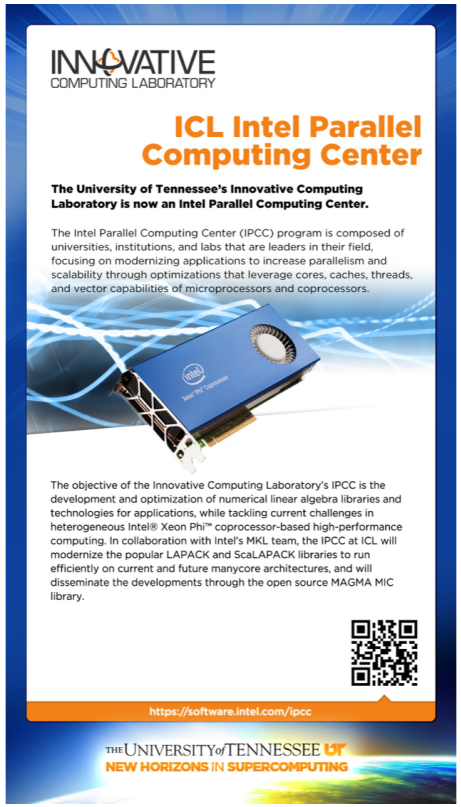
Funding / Licensing

National Science Foundation
WHERE DISCOVERIES BEGIN

U.S. DEPARTMENT OF ENERGY

Modified BSD License
en.wikipedia.org/wiki/BSD_licenses

Industry Collaborations



MathWorks®

AMD

Microsoft

FUJITSU

Software Projects - Legacy

netlib.org

LAPACK	dense linear algebra (serial)
ScaLAPACK	dense linear algebra (distributed memory)
BLAS	Basic Linear Algebra Subroutines
CBLAS	BLAS C API
LAPACKE	LAPACK C API

legacy software and reference implementations

Software Projects – Multicores and Accelertors

netlib.org

- LAPACK
- ScaLAPACK
- BLAS
- CBLAS
- LAPACKE

icl.utk.edu/research

- PLASMA — dense linear algebra (multicore)
- MAGMA — dense linear algebra (accelerators)
- DPLASMA — dense linear algebra (distributed memory + accelerators)

new software for multicore and accelerators

Software Projects - Runtimes

netlib.org

LAPACK

ScaLAPACK

BLAS

CBLAS

LAPACKE

icl.utk.edu/research

PLASMA

MAGMA

DPLASMA

QUARK

scheduling
(multicore)

PaRSEC

scheduling
(distributed memory)

dynamic runtime schedulers

Software Projects – New Runtimes

netlib.org

LAPACK

ScaLAPACK

BLAS

CBLAS

LAPACKE

icl.utk.edu/research

PLASMA

MAGMA

DPLASMA

OpenMP

scheduling
(multicore)

PaRSEC

scheduling
(distributed memory)

dynamic runtime schedulers

Software Projects - HPC

netlib.org

- LAPACK
- ScaLAPACK
- BLAS
- CBLAS
- LAPACKE

icl.utk.edu/research

High Performance Linpack Benchmark (dense)	HPL
High Performance Conjugate Gradient (sparse)	HPCG
HPC Challenge (composite)	HPCC
Performance API	PAPI

benchmarks and performance tools

Dense Linear Algebra Problems

- linear systems of equations $AX = B$
- linear least squares $min \parallel B - AX \parallel_2$
- singular value decomposition (SVD) $A = U\Sigma V^T$
- eigenvalue value problems (EVP) $Ax = \lambda x$

- dense (square, rectangular)
- band

Data Types

dgesv

Precision	Description	C type
z	double precision complex	double _Complex
c	single precision complex	float _Complex
d	double precision real	double
s	single precision real	float

Data Types – Mixed Precision

dsgesv

Precision	Description
dz	double-complex input, double precision result (e.g. cblas_dznrm2)
sc	single-complex input, single precision result (e.g. cblas_scnrm2)
zc	mixed-precision algorithm (double-complex/single-complex, e.g. zcgesv)
ds	mixed-precision algorithm (double/single, e.g. dsgesv)

Matrix Types

dgesv

Precision	Description
ge	general
sy	symmetric
he	Hermitian
po	positive definite
tr	triangular
or	orthogonal
un	unitary

Precision	Description
gb	general band
sb	symmetric band
hb	Hermitian band
pb	positive definite band

Driver Routines

Precision	Description	
_gesv	$AX = B$	A is general (nonsymmetric)
_posv	$AX = B$	A is symmetric/Hermitian positive definite
_sysv/_hesv	$AX = B$	A is symmetric/Hermitian indefinite
_gels	$AX = B$	A is rectangular

Precision	Description	
_geev	$Ax = \lambda x$	A is general
_syev/_heev	$Ax = \lambda x$	A is symmetric/Hermitian
syevd/heevd	$Ax = \lambda x$	A is symmetric/Hermitian, divide and conquer
sygvd/_hegvd	$Ax = \lambda Bx$	A is symmetric/Hermitian
_gesvd	$A = U\Sigma V^T$	A is general
_gesdd	$A = U\Sigma V^T$	A is general, divide and conquer

Computational Routines

Name	Description
_getrf, _potrf, _sytrf	triangular factorization (LU, Cholesky, LDL^T)
_getrs, _potrs, _sytrs	triangular solve (forward, backward substitution)
_getri, _potri, _sytri	triangular inverse

Name	Description
_geqrf, _gelqf	QR, LQ factorizations
_ormqr, _ormlq	multiply by Q (real)
_unmqr, _unmlq	multiply by Q (complex)
_orgqr, _orglq	generate Q (real)
_ungqr, _unglq	generate Q (complex)

Auxiliary Routines

Name	Description
_geadd	add two matrices
_laset	set entries to a constant
_lacpy	copy a matrix
_lascal	scale a matrix
_lange	compute a norm

LAPACK Working Notes

http://www.netlib.org/lapack/lawns/

lawn05 [pdf]

Provisional Contents

by C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, and D. Sorensen

ANL, MCS-TM-38, September 1988

lawn04 [pdf]

Guidelines for the Design of Symmetric Eigenroutines, SVD, and Iterative Refinement and Condition Estimation for Linear Systems

by J. Demmel, J. Du Croz, S. Hammarling, and D. Sorensen

ANL, MCS-TM-111, March 1988

lawn03 [pdf]

Computing Small Singular Values of Bidiagonal Matrices with Guaranteed High Relative Accuracy

by J. Demmel and W. Kahan

ANL, MCS-TM-110, February 1988

lawn02 [pdf]

Block Reduction of Matrices to Condensed Forms for Eigenvalue Computations

by J. Dongarra, S. Hammarling, and D. Sorensen

ANL, MCS-TM-99, September 1987

lawn01 [pdf]

Prospectus for the Development of a Linear Algebra Library for High-Performance Computers

by J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, and D. Sorensen

ANL, MCS-TM-97, September 1987

Last updated 2016-11-18 17:33:01 PST

lawn11 [pdf]

The Bidiagonal Singular Value Decomposition and Hamiltonian Mechanics

by P. Deift, J. Demmel, L.-C. Li, and C. Tomei ANL, MCS-TM-133, August 1989.

lawn10 [pdf]

Installing and Testing the Initial Release of LAPACK --Unix and Non-Unix Versions

by E. Anderson and J. Dongarra ANL, MCS-TM-130, May 1989.

lawn09 [pdf]

A Test Matrix Generation Suite

by J. Demmel and A. McKenney ANL, MCS-P69-0389, March 1989.

lawn08 [pdf]

On a Block Implementation of Hessenberg Multishift QR Iteration

by Z. Bai and J. Demmel

ANL, MCS-TM-127, January 1989.

lawn07 [pdf]

Computing Accurate Eigensystems of Scaled Diagonally Dominant Matrice

by J. Barlow and J. Demmel

ANL, MCS-TM-126, December 1988.

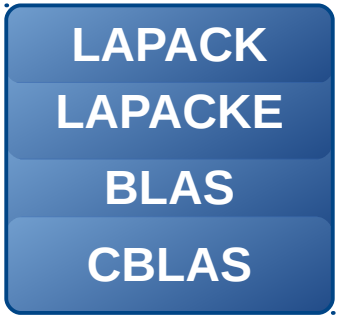
lawn06 [pdf]

Tools to Aid in the Analysis of Memory Access Patterns for FORTRAN Programs

by O. Brewer, J. Dongarra, and D. Sorensen

ANL, MCS-TM-120, June 1988

Open Source Stack



http://www.netlib.org/lapack/

http://www.netlib.org/lapack/lapack-3.6.1.tgz

https://github.com/Reference-LAPACK/lapack

includes: LAPACKE, BLAS, CBLAS – reference implementations

http://math-atlas.sourceforge.net

http://www.openblas.net

https://github.com/xianyi/OpenBLAS

Intel Stack for Numerical Linear Algebra

https://software.intel.com/sites/campaings/nest/

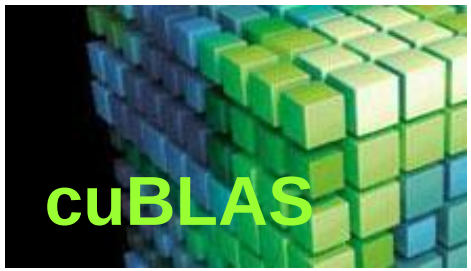
- Intel Threading Building Blocks
- Intel Integrated Performance Primitives
- Intel Data Analytics Acceleration Library
- Intel Math Kernel Library
  - LAPACK
  - LAPACKE
  - BLAS
  - CBLAS

royalty free

NVIDIA Stack

https://developer.nvidia.com/gpu-accelerated-libraries

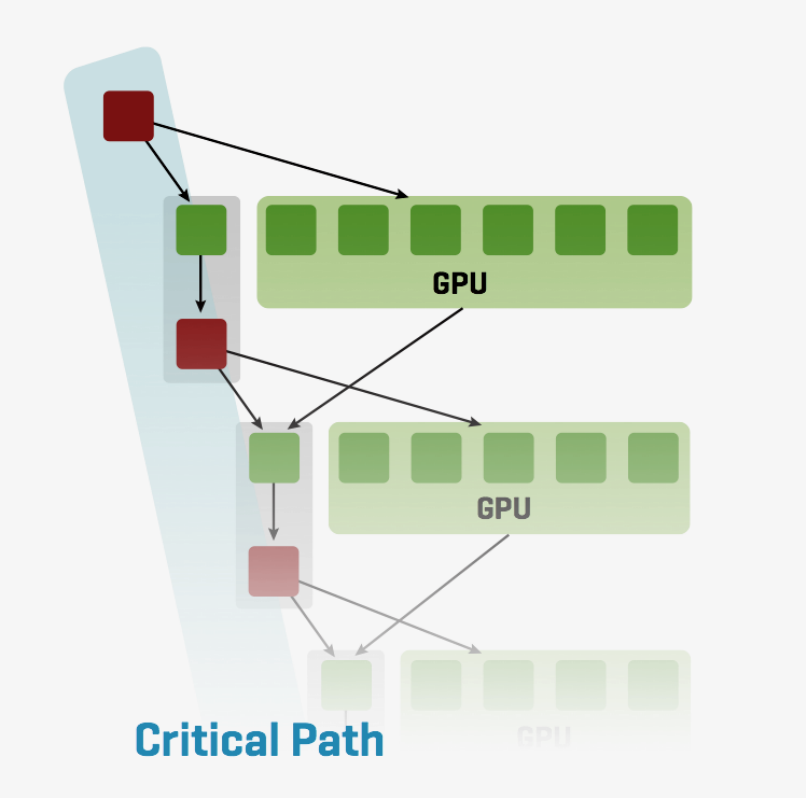
http://icl.cs.utk.edu/magma/software/



https://developer.nvidia.com/cuda-downloads

- cuBLAS – single GPU BLAS
- cuBLAS-XT – multi-GPU “OOC” BLAS
- NVBLAS – “auto acceleration” API for cuBLAS-XT

MAGMA



- dense linear algebra for accelerators
  - NVIDIA using CUDA
  - AMD using OpenCL
  - Intel Xeon Phi
- hybrid, CPU-GPU implementations
  - single-GPU
  - multi-GPU
  - OO-GPU-memory
- managing data transfers
- some batched routines
- some sparse solvers

MAGMA Overview

NVIDIA
GPU
CENTER OF
EXCELLENCE

NVIDIA's GPU Center of Excellence Program recognizes universities expanding the frontier of massively parallel computing using CUDA.

intel
Intel Parallel Computing Center

The objective of the Innovative Computing Laboratory's IPCC is the development and optimization of numerical linear algebra libraries and technologies for applications, while tackling current challenges in heterogeneous Intel® Xeon Phi™ coprocessor-based High Performance Computing.

AMD

Long-term collaboration and support on the development of clMAGMA, the OpenCL™ port of MAGMA.

FEATURES AND SUPPORT

- **MAGMA 2.2** FOR **CUDA**
- **clMAGMA 1.4** FOR **OpenCL**
- **MAGMA MIC 1.4** FOR **Intel Xeon Phi**

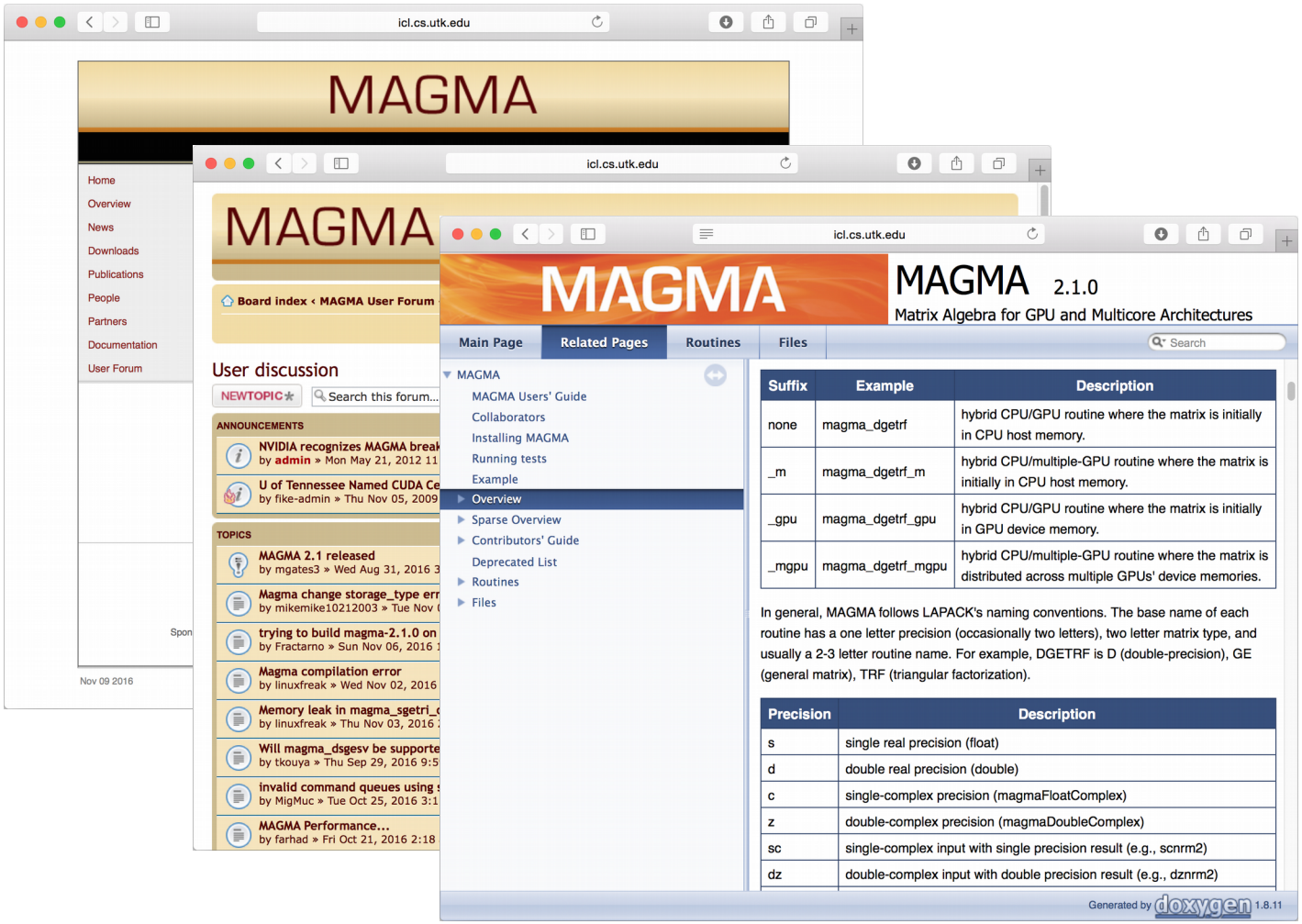
CUDA	OpenCL	Intel Xeon Phi	
●	●	●	Linear system solvers
●	●	●	Eigenvalue problem solvers
●	●		Auxiliary BLAS
●			Batched LA
●		●	Sparse LA
●	●	●	CPU Interface
●	●	●	GPU Interface
●	●	●	Multiple precision support
●			Non-GPU-resident factorizations
●	●	●	Multicore and multi-GPU support
●	●	●	LAPACK testing
●	●	●	Linux
●	●		Windows
●	●		Mac OS

MAGMA User Outreach

http://icl.cs.utk.edu/magma/

→ User Forum

→ Documentation



MAGMA Routines

Suffix	Example	Description
none	**magma_dgesv**	hybrid CPU/GPU routine – matrix in CPU memory
_m	**magma_dgesv_m**	hybrid CPU/multi-GPU routine – matrix in CPU memory
_gpu	**magma_dgesv_gpu**	hybrid CPU/GPU routine – matrix in GPU memory
_mgpu	**magma_dgesv_mgpu**	hybrid CPU/multi-GPU routine – matrix distributed across GPU memories

MAGMA Memory Allocation

Name	Description
magma_malloc_cpu	allocate CPU memory
magma_malloc_pinned	allocate pinned CPU memory
magma_malloc	allocate GPU memory
magma_free_cpu	free CPU memory
magma_free_pinned	free pinned CPU memory
magma_free	free GPU memory

MAGMA Data Transfers

Name	Description
setmatrix	send matrix to GPU
setvector	send vector to GPU
getmatrix	get matrix from GPU
getvector	get vector from GPU

Calling MAGMA

```
info = LAPACKE_dgetrf   (layout, m, n, hA,   lda, ipiv);

       // A in CPU memory
       magma_dgetrf     (        m, n, hA,   lda, ipiv, &info);


       // A in CPU memory
       magma_dgetrf_m   (ngpu,   m, n, hA,   lda, ipiv, &info);


       // A in GPU memory
       magma_dgetrf_gpu (        m, n, dA,   lda, ipiv, &info);


       // A in GPU memories
       magma_dgetrf_mgpu(ngpu,   m, n, dA[], lda, ipiv, &info);
```

MAGMA: initialize, compute, finalize

```
magma_init();
magma_dgetrf(m, n, hA, lda, ipiv, &info);
magma_finalize();
```

```
magma_init();
magma_dgetrf_m(ngpu, m, n, hA, lda, ipiv, &info);
magma_finalize();
```

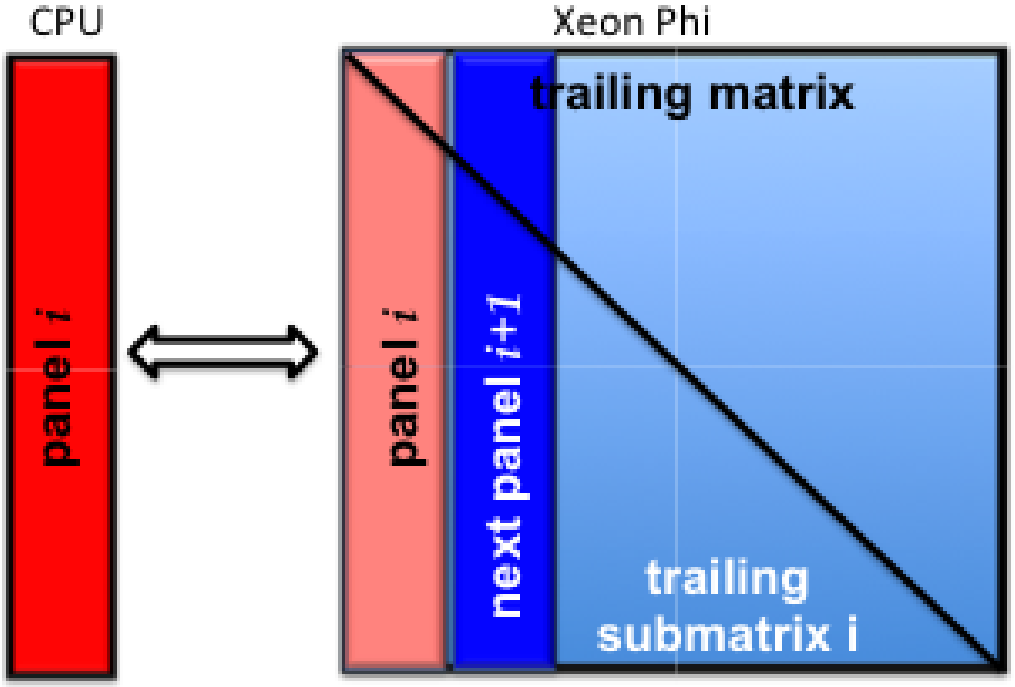
MAGMA Queues

```
magma_init();
magma_queue_create(device, &queue);
magma_zsetmatrix(m, n, h_A, lda, d_A, ldda, queue);

magma_zgetrf_gpu(m, n, d_A, ldda, ipiv, &info);

magma_zgetmatrix(m, n, d_A, ldda, h_A, lda, queue);
magma_queue_destroy(queue);
magma_finalize();
```

Multi-GPU Interface

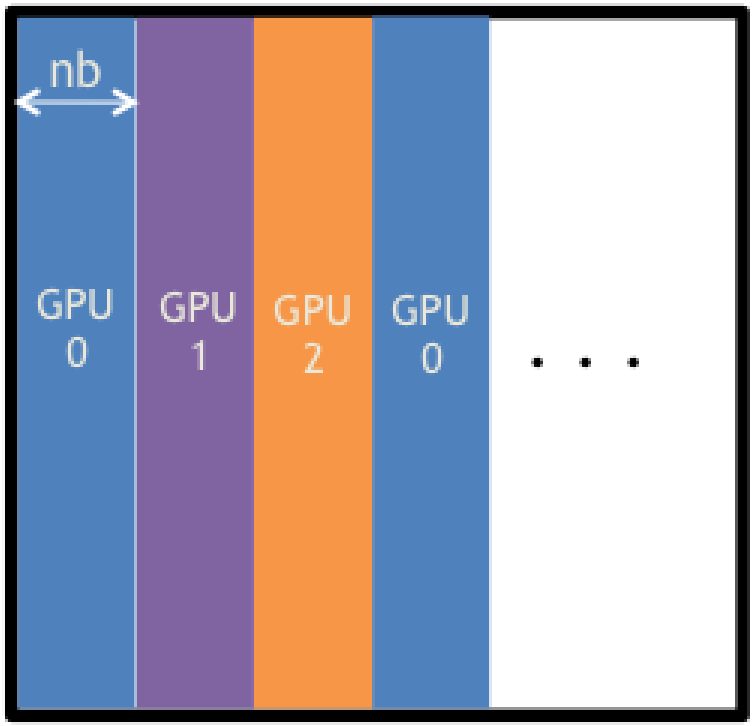
```
magma_init();
nb = magma_get_zgetrf_nb(m, n);
magma_queue_create(device, &queue);
magma_zsetmatrix_1D_col_bcyclic(m, n, h_A, lda, d_lA, ldda,
ngpu, nb, queues);

magma_zgetrf_mgpu(ngpu, M, N, d_lA, ldda, ipiv, &info);

magma_zgetmatrix_1D_col_bcyclic(m, n, d_lA, ldda, h_A, lda,
ngpu, nb, queues);
magma_queue_destroy(queue);
magma_finalize();
```

MAGMA Factorization Algorithms



- panel factorization on multicore
- trailing submatrix update on GPU/Phi
- concurrent operation
- lookahead
- hidden communication

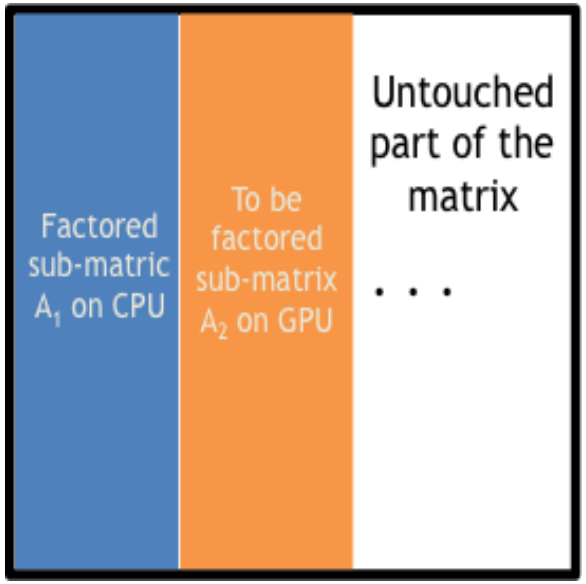
MAGMA Multi-GPU



- 1D block-cyclic distribution
- panel on CPUs
- updates on GPUs
- lookahead

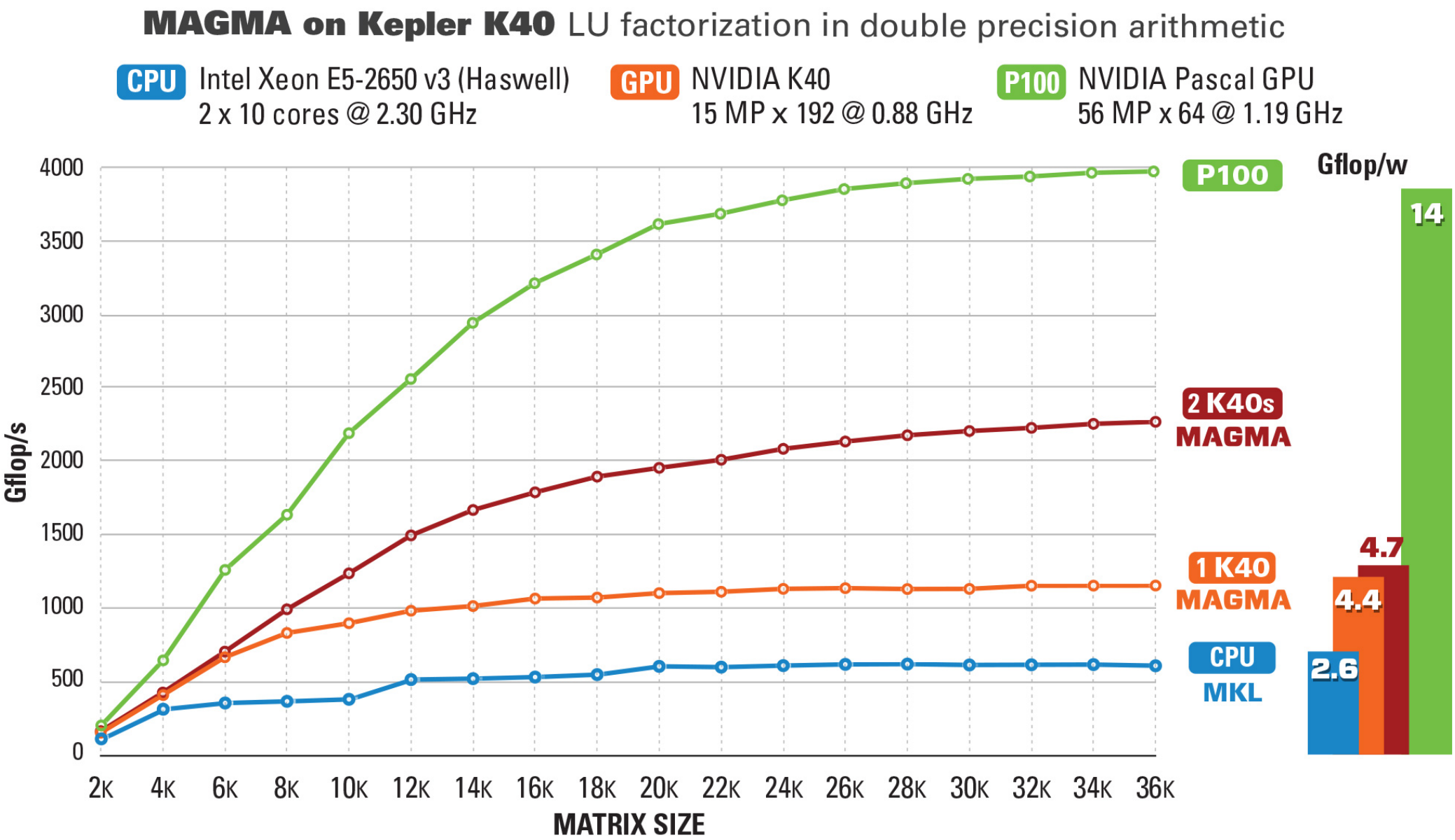
MAGMA External-Memory Computing

- perform left-looking factorization on a submatrix that fits in the GPU memory
- the rest of the matrix stays on the CPU

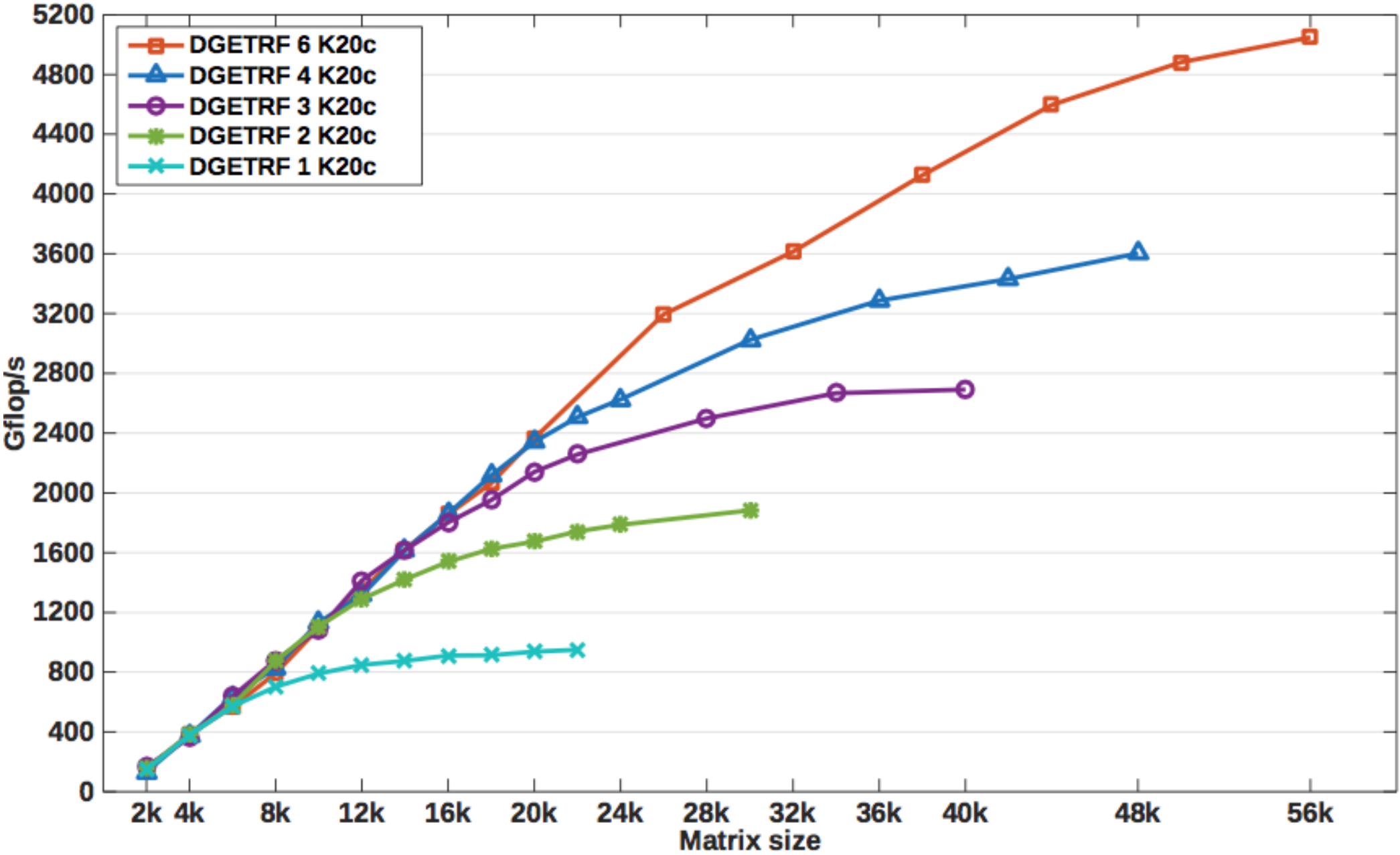


- copy A2 to the GPU
- update A2 using A1
- factor the updated A2
- return A2 to the CPU

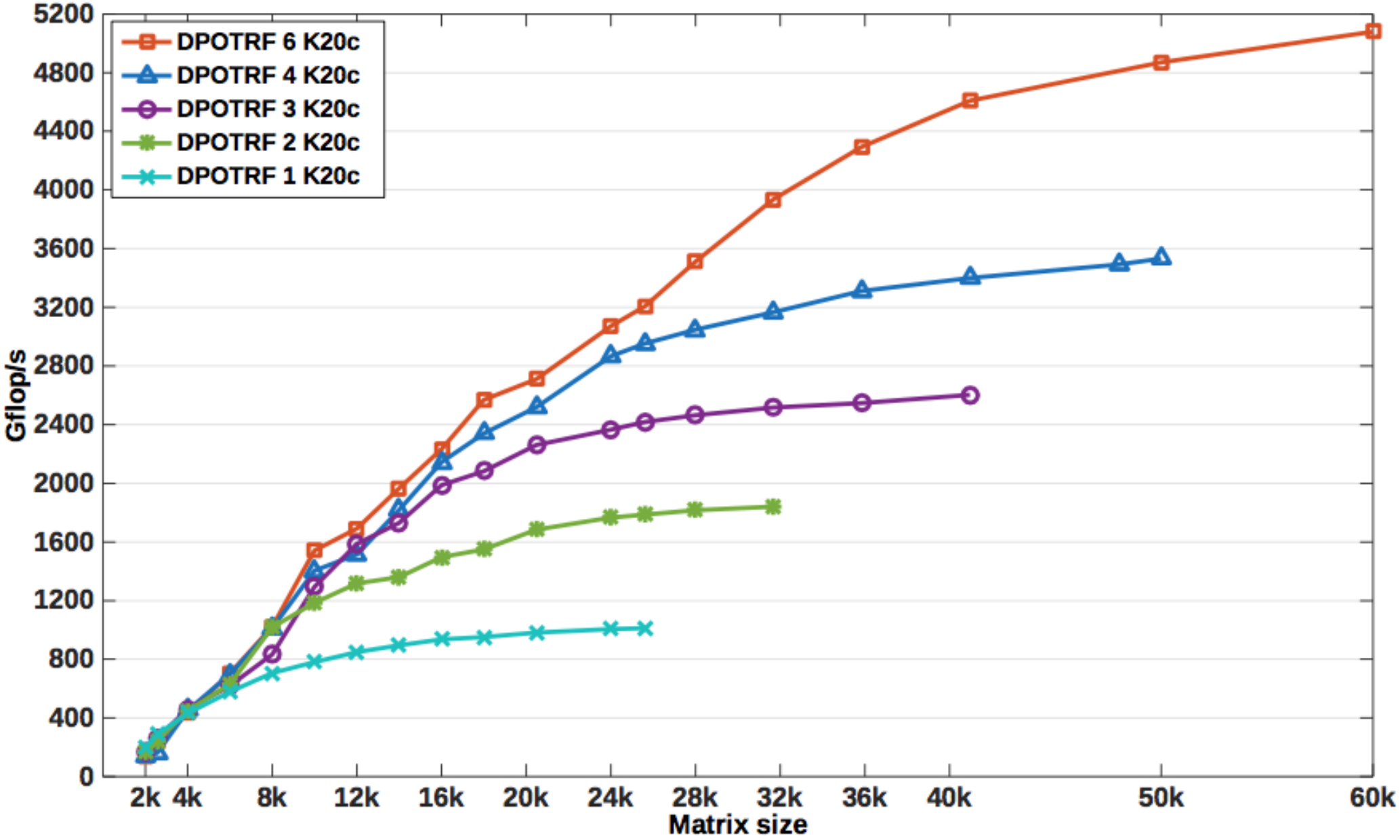
MAGMA Performance – LU Factorization



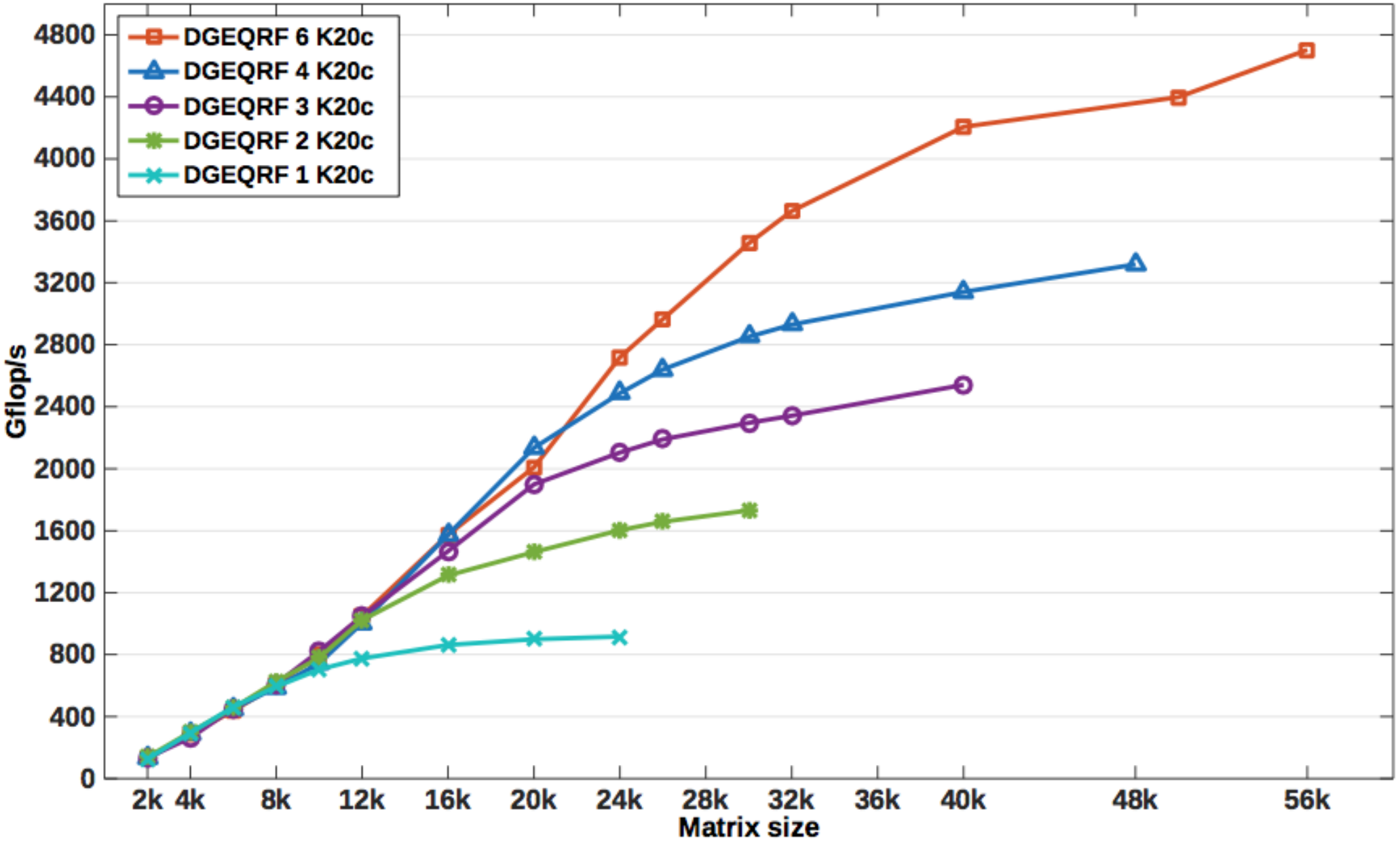
MAGMA Performance – DGETRF on K20c



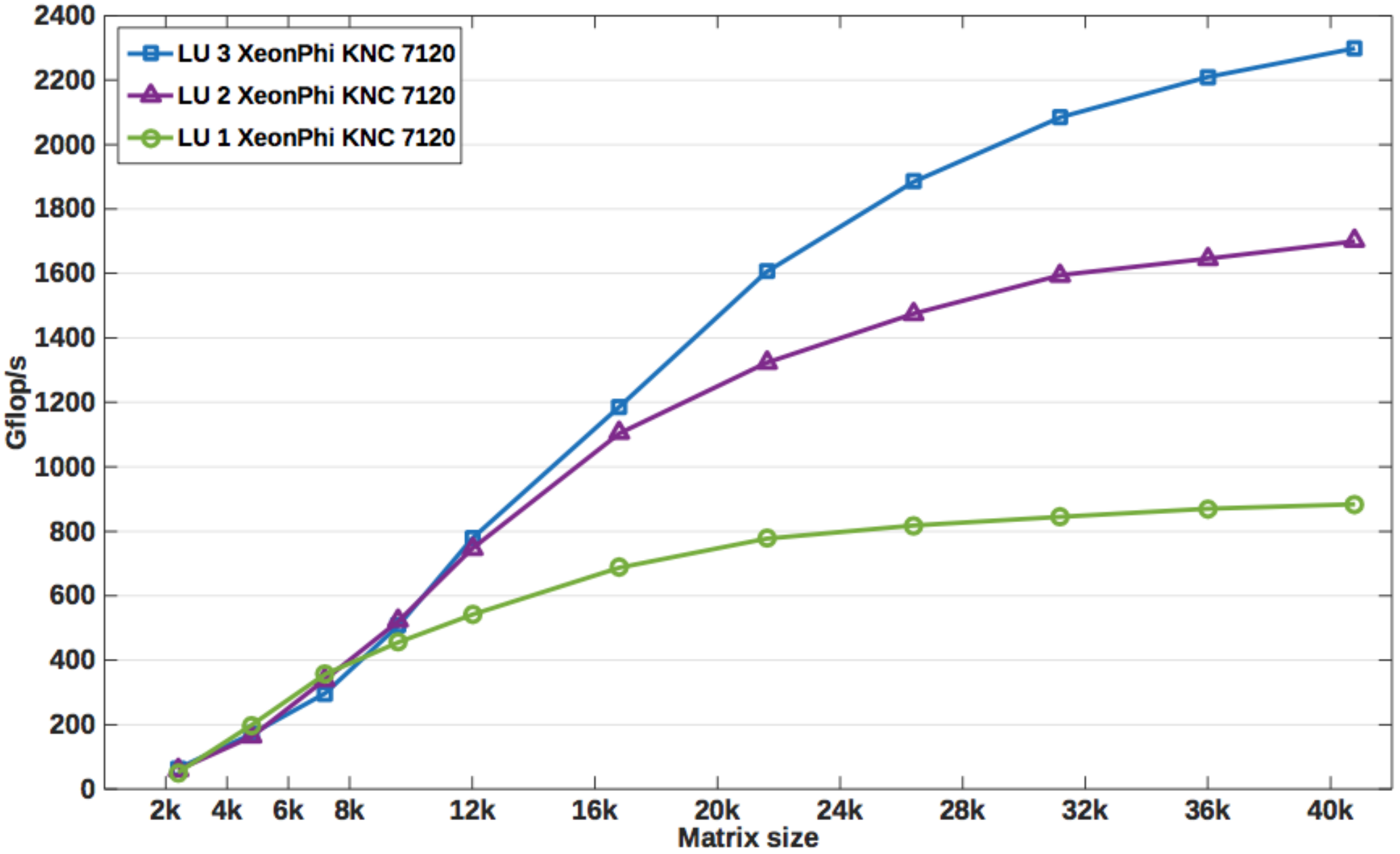
MAGMA Performance – DPOTRF on K20c



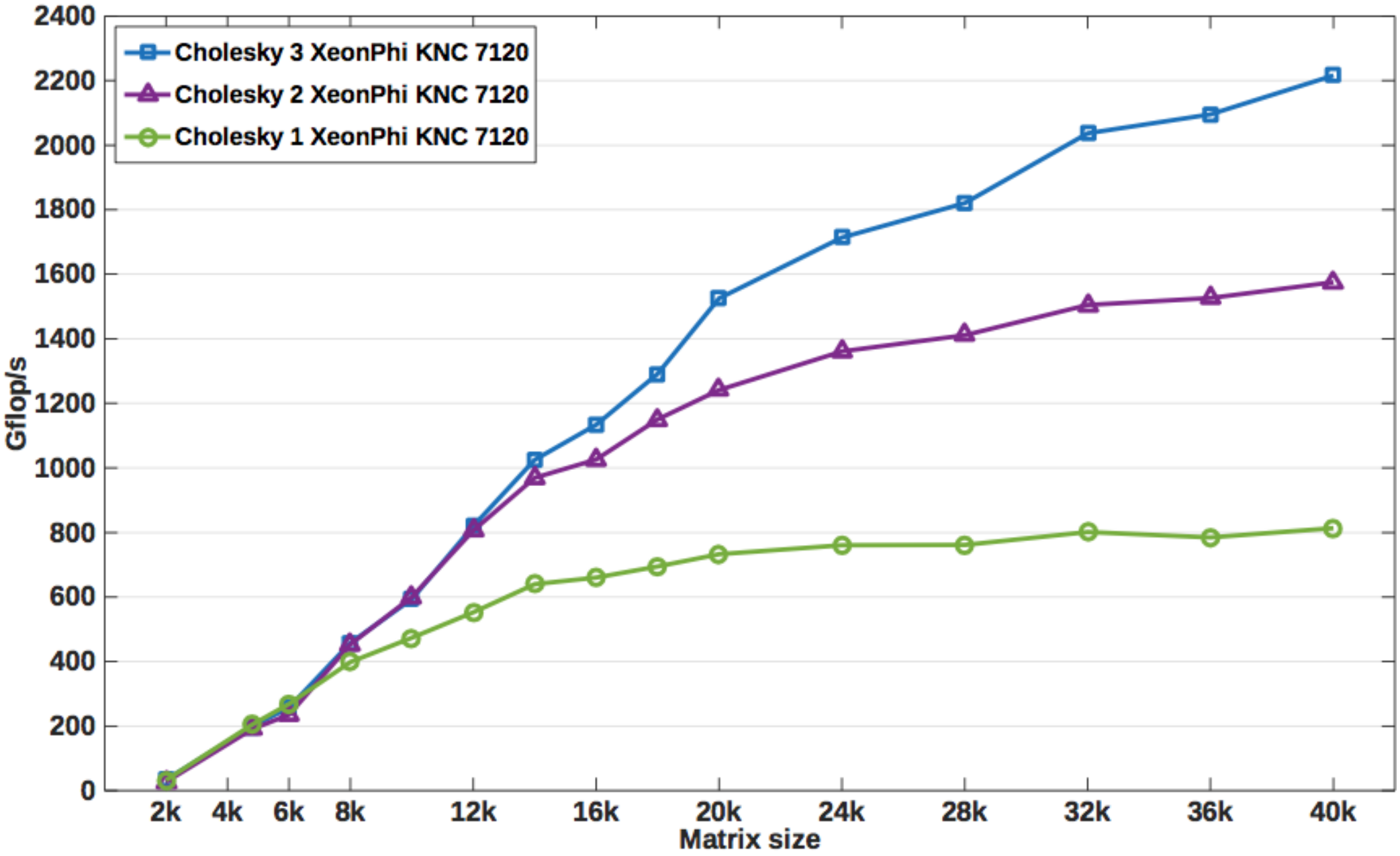
MAGMA Performance – DGEQRF on K20c



MAGMA Performance – DGETRF on KNC



MAGMA Performance – DPOTRF on KNC



MAGMA Performance – DGEQRF on KNC