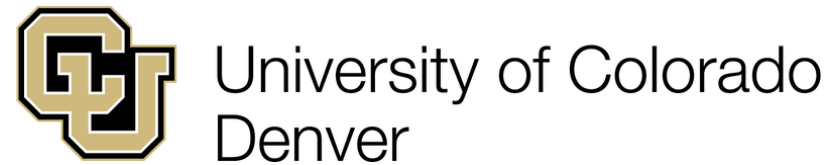


Numerical Libraries for Dense Linear Algebra

Innovative Computing Laboratory's Libraries

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

Academic Collaborations



Funding / Licensing



National Science Foundation
WHERE DISCOVERIES BEGIN



U.S. DEPARTMENT OF
ENERGY

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

-- Innovative Computing Laboratory
-- Electrical Engineering and Computer Science Department
-- University of Tennessee
-- (C) Copyright 2012-2015

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the University of Tennessee, Knoxville nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.


THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Industry Collaborations

INNOVATIVE
COMPUTING LABORATORY

ICL NVIDIA CUDA Center of Excellence

The University of Tennessee's Innovative Computing Laboratory has been recognized as a CUDA Center of Excellence since 2009.



NVIDIA's CUDA Center of Excellence (CCOE) program recognizes, rewards, and fosters collaboration with institutions at the forefront of massively parallel manycore computing research. The Innovative Computing Laboratory (ICL) is part of a select group of labs given the CCOE designation. UT's CCOE focuses on the development of numerical linear algebra libraries for CUDA-based hybrid architectures. ICL's work on the Matrix Algebra on GPU and Multicore Architectures (MAGMA) project further enables and expands our CUDA-based software library efforts, especially in the area of high-performance scientific computing.



<https://research.nvidia.com/content/cuda-centers-excellence>


THE UNIVERSITY of TENNESSEE **UT**
NEW HORIZONS IN SUPERCOMPUTING

INNOVATIVE
COMPUTING LABORATORY


ICL Intel Parallel Computing Center

The University of Tennessee's Innovative Computing Laboratory is now an Intel Parallel Computing Center.

The Intel Parallel Computing Center (IPCC) program is composed of universities, institutions, and labs that are leaders in their field, focusing on modernizing applications to increase parallelism and scalability through optimizations that leverage cores, caches, threads, and vector capabilities of microprocessors and coprocessors.



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. In collaboration with Intel's MKL team, the IPCC at ICL will modernize the popular LAPACK and ScaLAPACK libraries to run efficiently on current and future manycore architectures, and will disseminate the developments through the open source MAGMA MIC library.



<https://software.intel.com/ipcc>

THE UNIVERSITY of TENNESSEE **UT**
NEW HORIZONS IN SUPERCOMPUTING



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 Accelerators

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

PaRSEC

dynamic
runtime
schedulers

scheduling
(multicore)

scheduling
(distributed memory)

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

High Performance Linpack Benchmark
(dense)

High Performance Conjugate Gradient
(sparse)

HPC Challenge
(composite)

Performance API

icl.utk.edu/research

HPL

HPCG

HPCC

PAPI

**benchmarks
and performance tools**


Dense Linear Algebra Problems

- linear systems of equations $AX = B$
- linear least squares $\min \| B - AX \|_2$
- singular value decomposition (SVD) $A = U\Sigma V^T$
- eigenvalue value problems (EVP) $Ax = \lambda x$

- dense (square, rectangular)
- band

Data Types

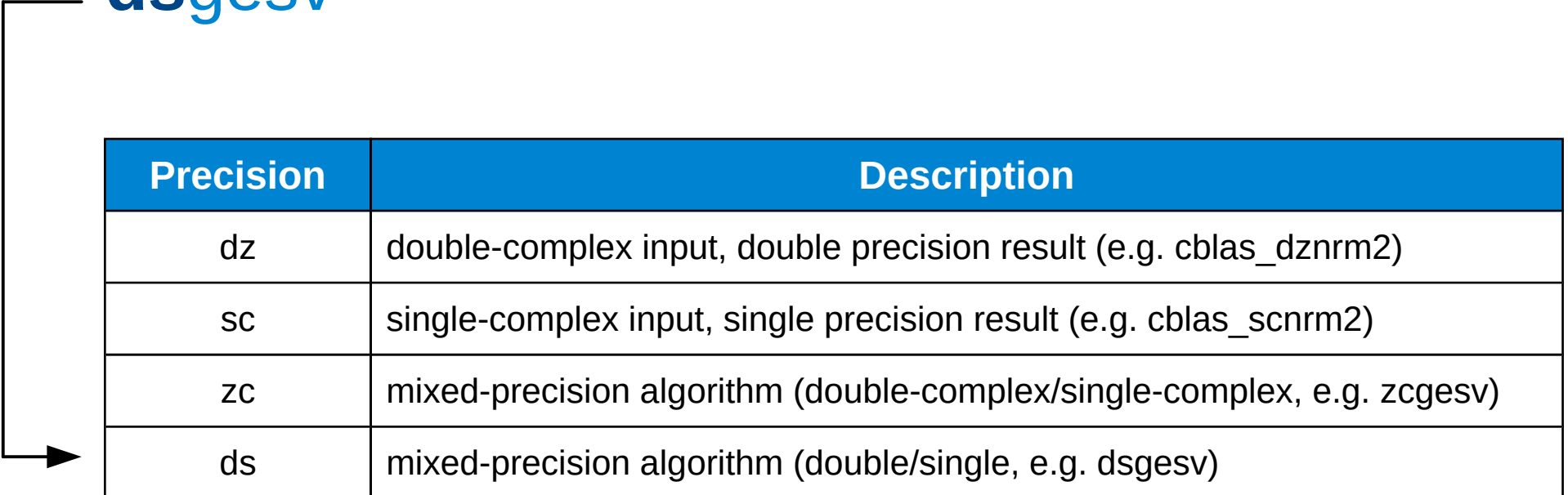
dgesv



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

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

Computational Routines

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

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

Auxiliary Routines

Name	Description
<code>_geadd</code>	add two matrices
<code>_laset</code>	set entries to a constant
<code>_lacpy</code>	copy a matrix
<code>_lascal</code>	scale a matrix
<code>_lange</code>	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 Matrices

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

LAPACK

LAPACKE

BLAS

CBLAS

<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

ATLAS

<http://math-atlas.sourceforge.net>

OpenBLAS

<http://www.openblas.net>

<https://github.com/xianyi/OpenBLAS>

Intel Stack for Numerical Linear Algebra

<https://software.intel.com/sites/campaigns/nest/>

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

royalty free

intel
Software

TAKE FLIGHT
LET YOUR CODE SOAR

Get Community Licensing >

WELCOME TO THE AVIARY.
Learn more here about your feathered friends.

NVIDIA Stack

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



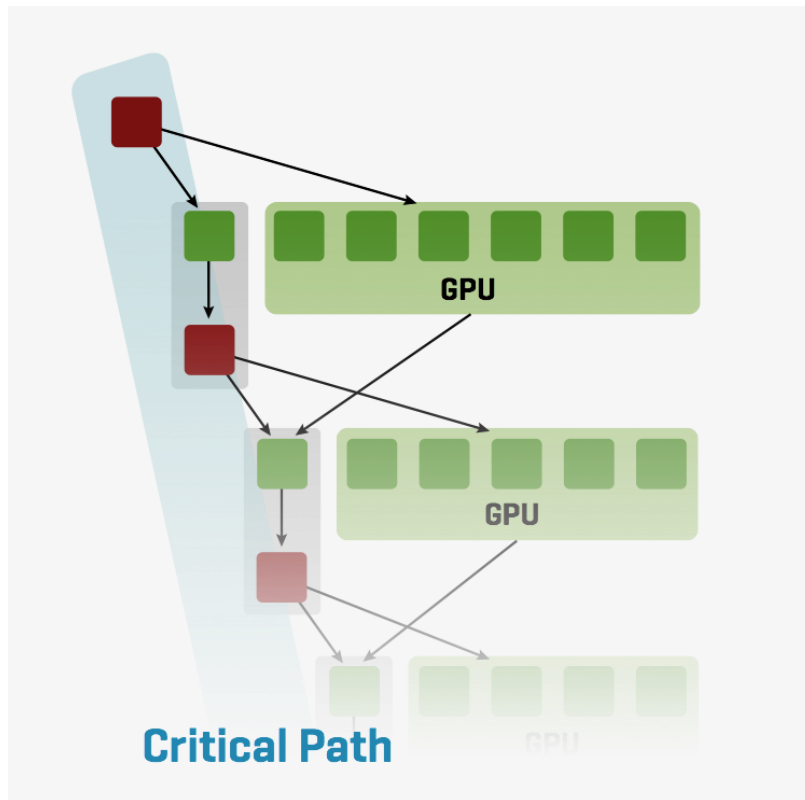
<http://icl.cs.utk.edu/magma/software/>



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

- cuBLAS – single GPU BLAS
- cuBLAS-XT – multi-GPU “OOO” 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's GPU Center of Excellence Program recognizes universities expanding the frontier of massively parallel computing using CUDA.



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.



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

FEATURES AND SUPPORT

- ▶ **MAGMA 2.2** FOR **CUDA**
- ▶ **cMAGMA 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

The screenshot displays three overlapping browser windows from the URL `icl.cs.utk.edu`. The top window shows the main MAGMA header. The middle window shows the 'MAGMA User Forum' page with a sidebar menu containing: Home, Overview, News, Downloads, Publications, People, Partners, Documentation, and User Forum. The main content area includes a 'User discussion' section with a search bar, a 'NEWTOPIC*' button, and several announcement and topic entries. The bottom window shows the 'MAGMA 2.1.0' documentation page, titled 'Matrix Algebra for GPU and Multicore Architectures'. It features a navigation menu with 'Main Page', 'Related Pages', 'Routines', and 'Files'. The 'Routines' section is expanded, listing items like 'MAGMA Users' Guide', 'Collaborators', 'Installing MAGMA', 'Running tests', 'Example', and 'Overview'. The 'Files' section is also expanded, showing a table of routines.

Suffix	Example	Description
none	<code>magma_dgetrf</code>	hybrid CPU/GPU routine where the matrix is initially in CPU host memory.
<code>_m</code>	<code>magma_dgetrf_m</code>	hybrid CPU/multiple-GPU routine where the matrix is initially in CPU host memory.
<code>_gpu</code>	<code>magma_dgetrf_gpu</code>	hybrid CPU/GPU routine where the matrix is initially in GPU device memory.
<code>_mgpu</code>	<code>magma_dgetrf_mgpu</code>	hybrid CPU/multiple-GPU routine where the matrix is distributed across multiple GPUs' device memories.

In general, MAGMA follows LAPACK's naming conventions. The base name of each routine has a one letter precision (occasionally two letters), two letter matrix type, and usually a 2-3 letter routine name. For example, DGETRF is D (double-precision), GE (general matrix), TRF (triangular factorization).

Precision	Description
s	single real precision (float)
d	double real precision (double)
c	single-complex precision (magmaFloatComplex)
z	double-complex precision (magmaDoubleComplex)
sc	single-complex input with single precision result (e.g., <code>scnrm2</code>)
dz	double-complex input with double precision result (e.g., <code>dznm2</code>)

Generated by [doxygen](#) 1.8.11

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
<code>magma_malloc_cpu</code>	allocate CPU memory
<code>magma_malloc_pinned</code>	allocate pinned CPU memory
<code>magma_malloc</code>	allocate GPU memory
<code>magma_free_cpu</code>	free CPU memory
<code>magma_free_pinned</code>	free pinned CPU memory
<code>magma_free</code>	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_bccyclic(m, n, h_A, lda, d_lA, ldda,  
ngpu, nb, queues);
```

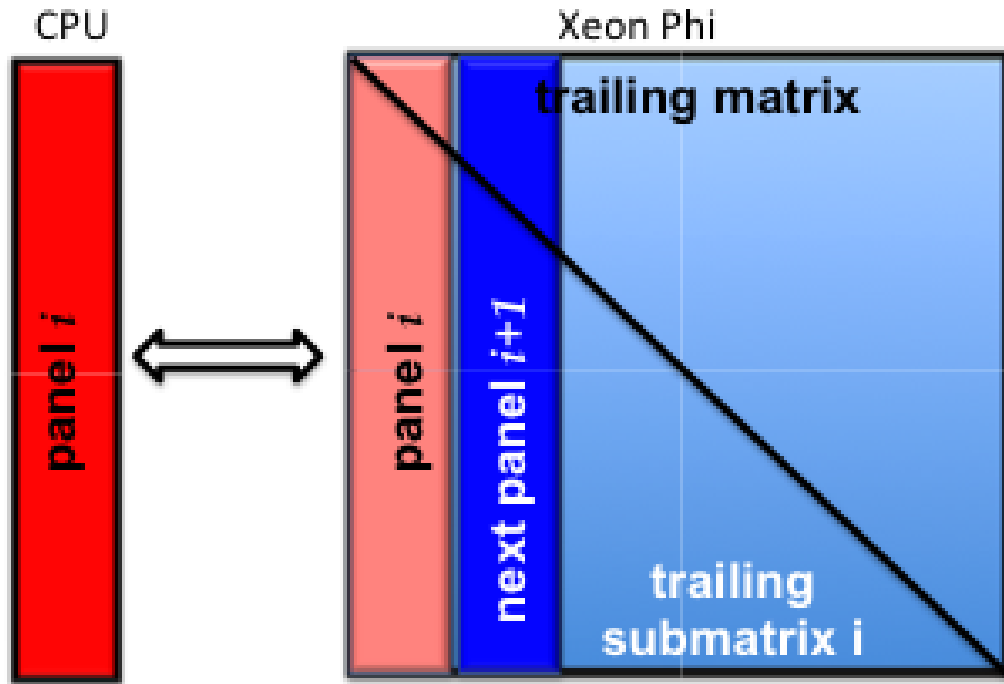
```
magma_zgetrf_mgpu(ngpu, M, N, d_lA, ldda, ipiv, &info);
```

```
magma_zgetmatrix_1D_col_bccyclic(m, n, d_lA, ldda, h_A, lda,  
ngpu, nb, queues);
```

```
magma_queue_destroy(queue);
```

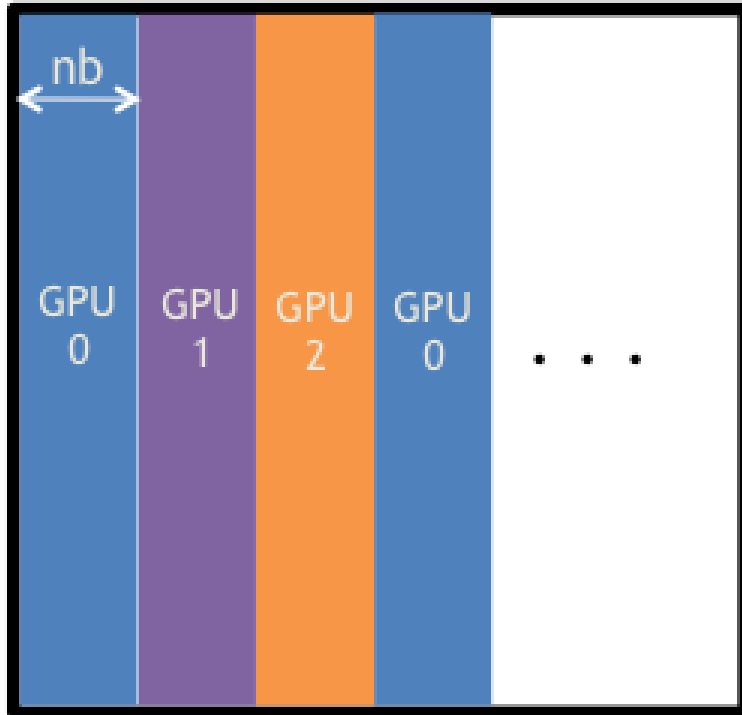
```
magma_finalize();
```

MAGMA Factorization Algorithms



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

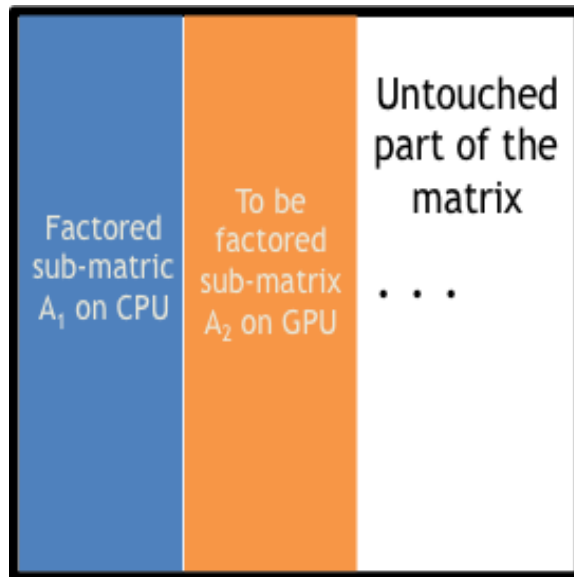
MAGMA Multi-GPU



- 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 A_2 to the GPU
- update A_2 using A_1
- factor the updated A_2
- return A_2 to the CPU

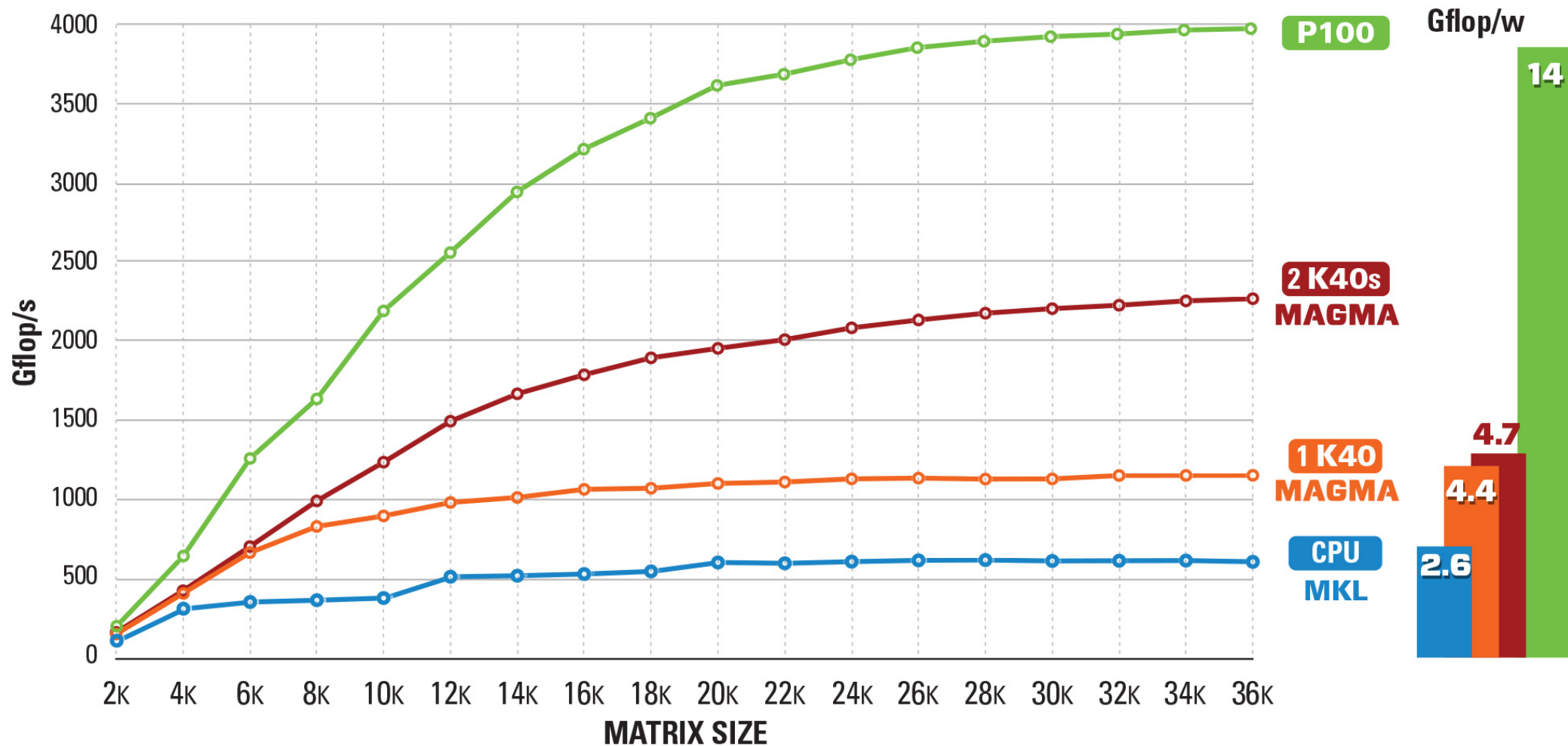
MAGMA Performance – LU Factorization

MAGMA on Kepler K40 LU factorization in double precision arithmetic

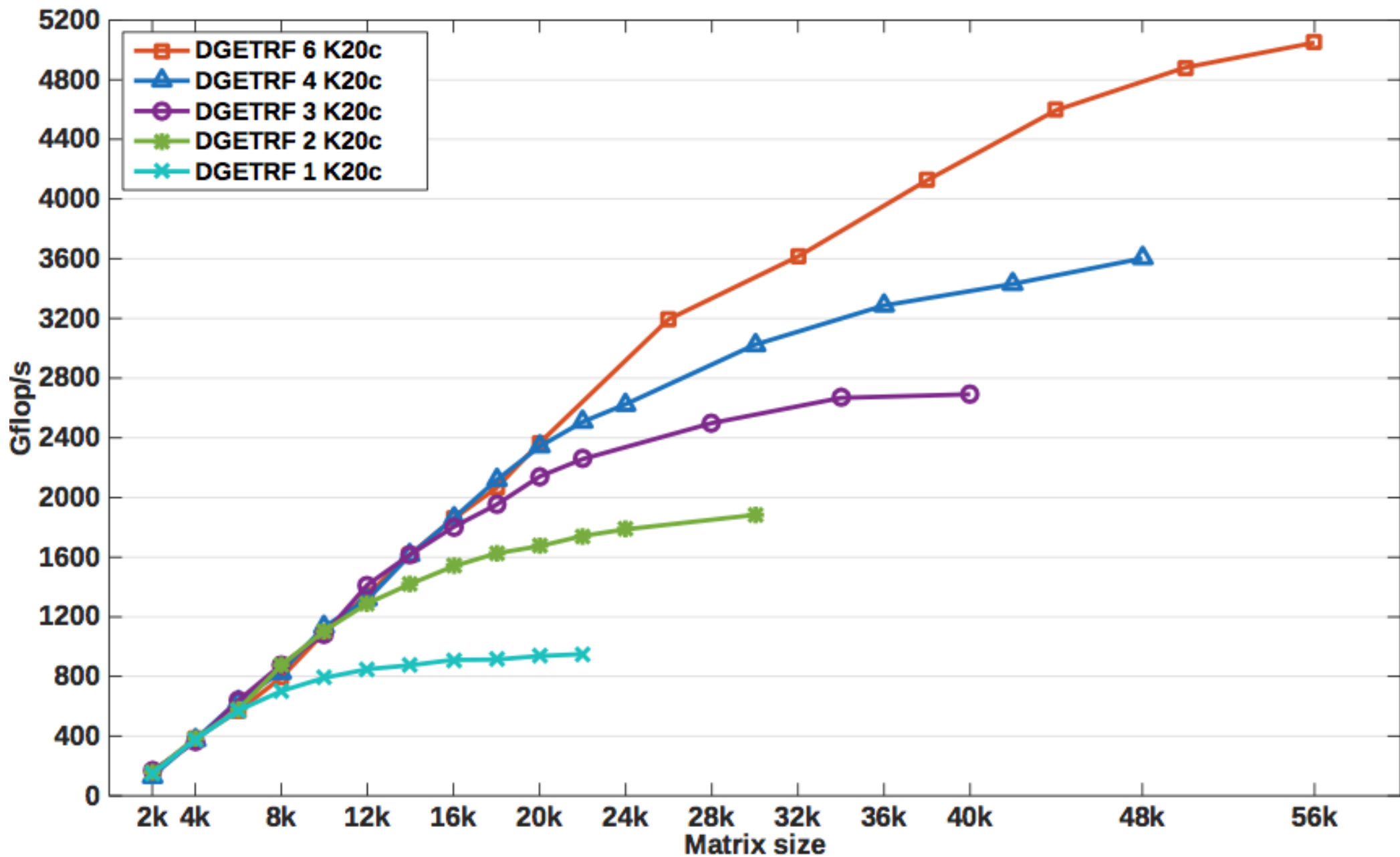
CPU Intel Xeon E5-2650 v3 (Haswell)
2 x 10 cores @ 2.30 GHz

GPU NVIDIA K40
15 MP x 192 @ 0.88 GHz

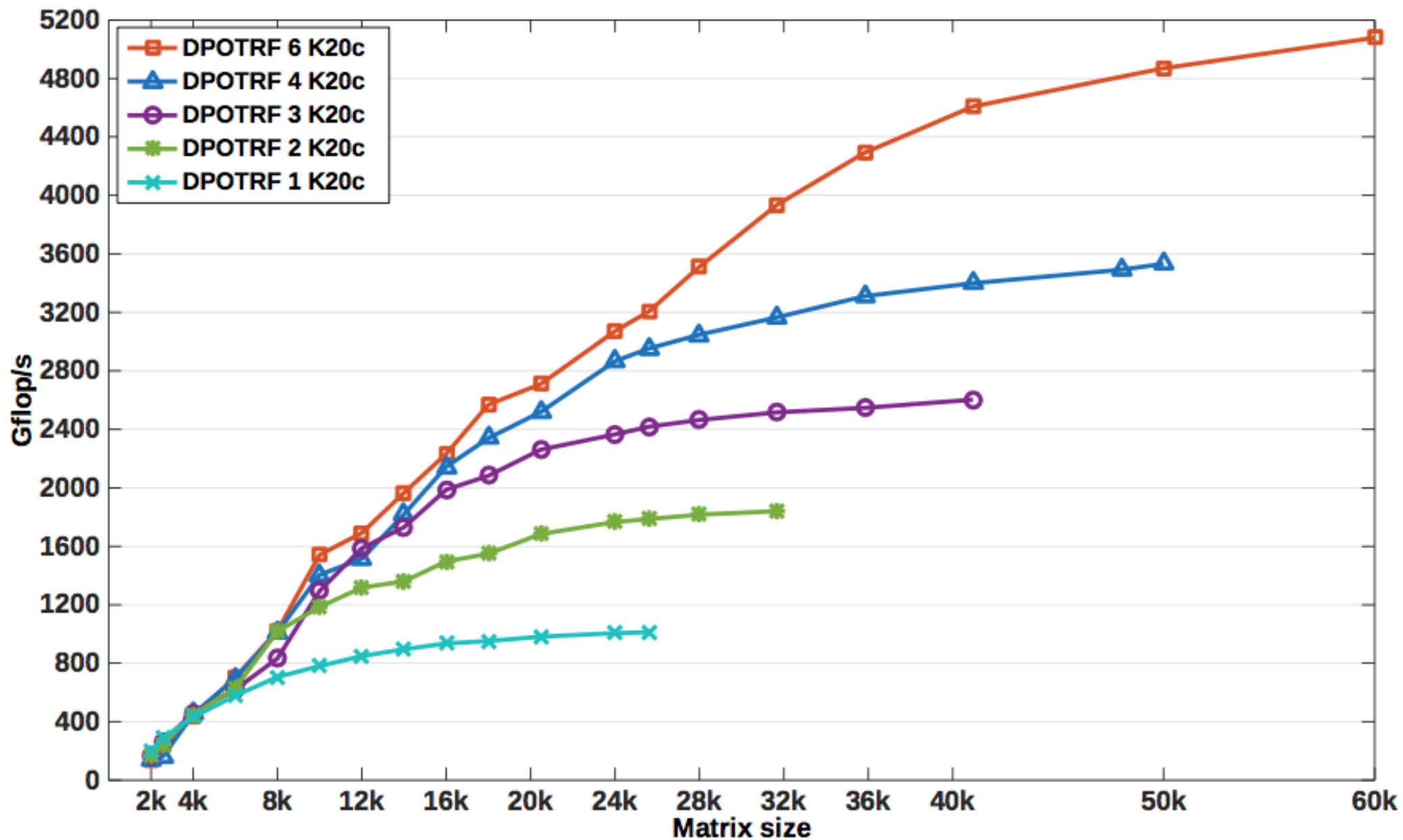
P100 NVIDIA Pascal GPU
56 MP x 64 @ 1.19 GHz



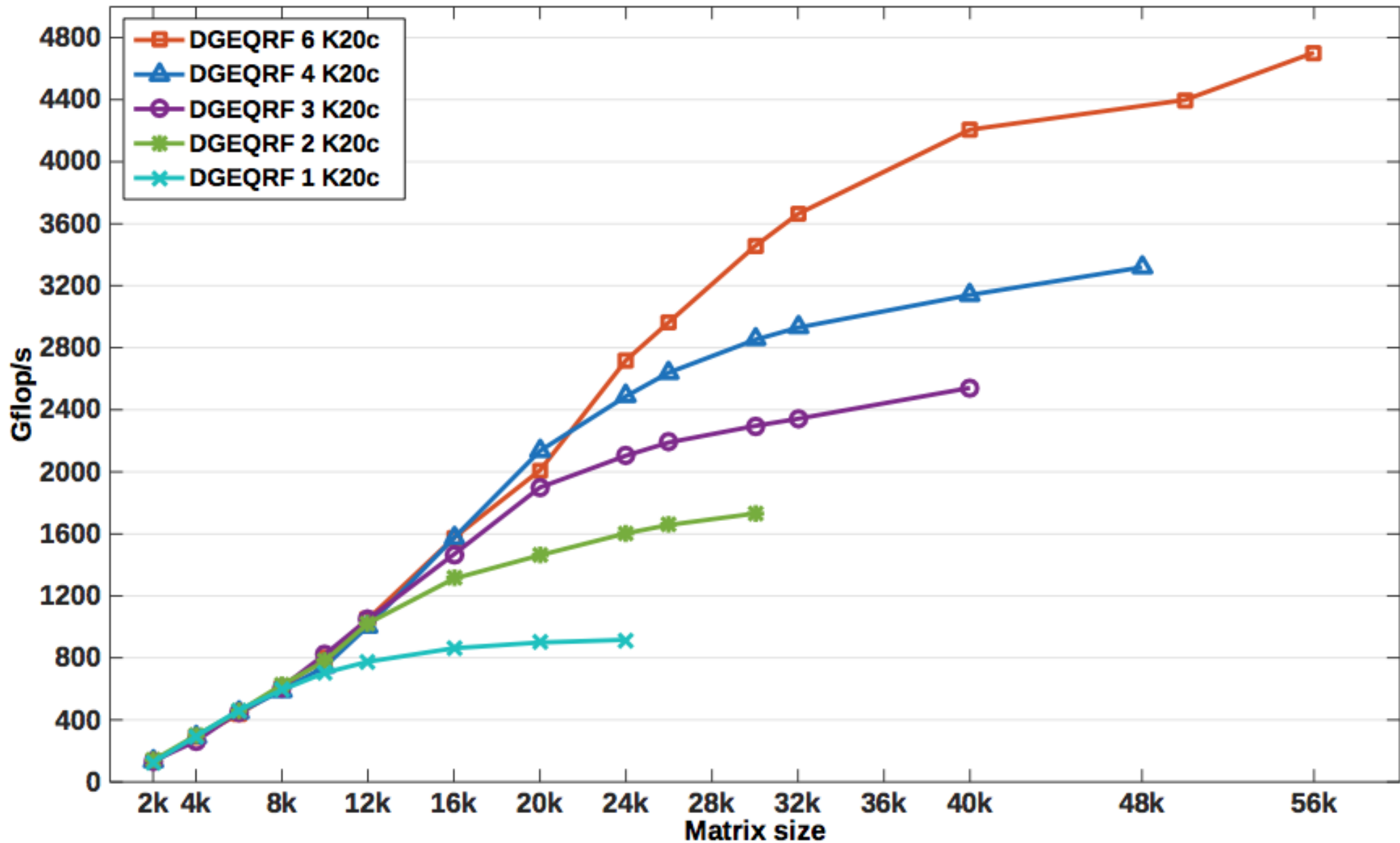
MAGMA Performance – DGETRF on K20c



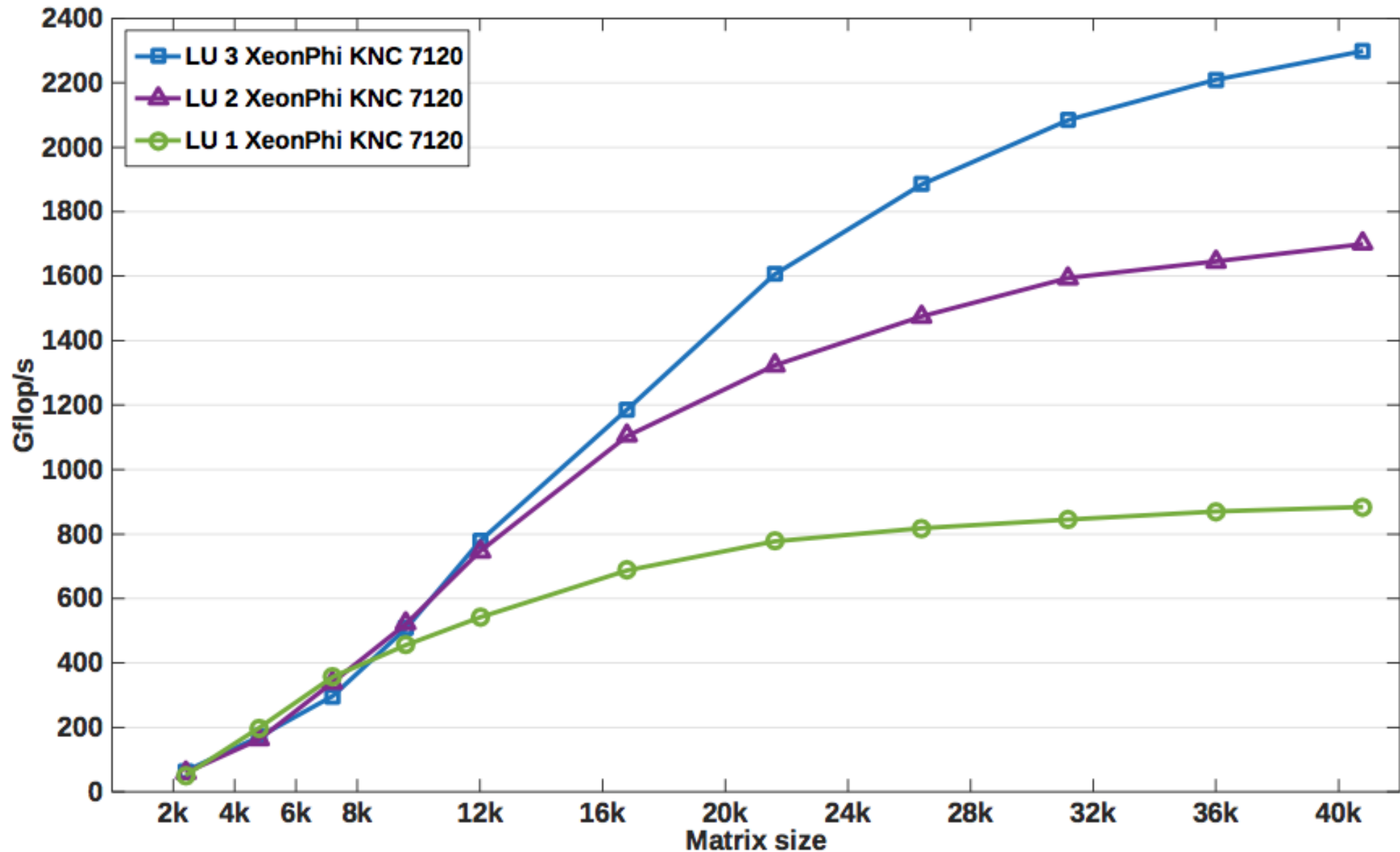
MAGMA Performance – DPOTRF on K20c



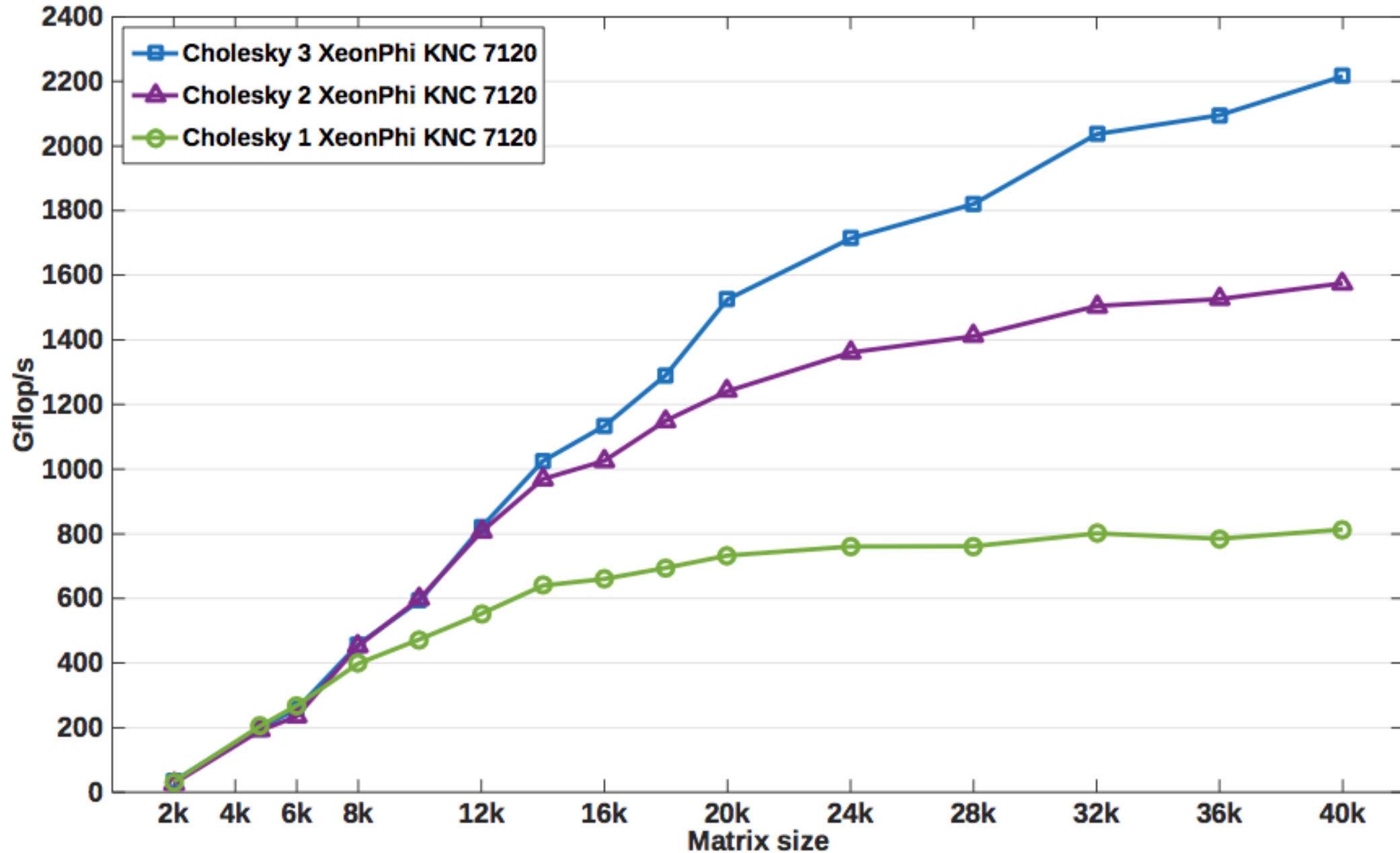
MAGMA Performance – DGEQRF on K20c



MAGMA Performance – DGETRF on KNC



MAGMA Performance – DPOTRF on KNC



MAGMA Performance – DGEQRF on KNC

