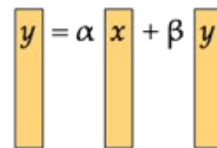


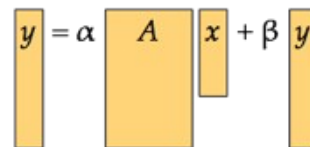
Dense Numerical Linear Algebra with MAGMA

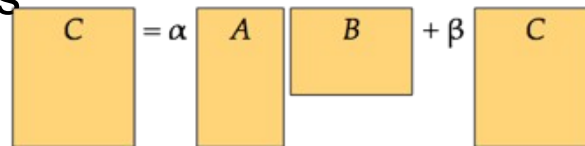
Piotr Luszczek

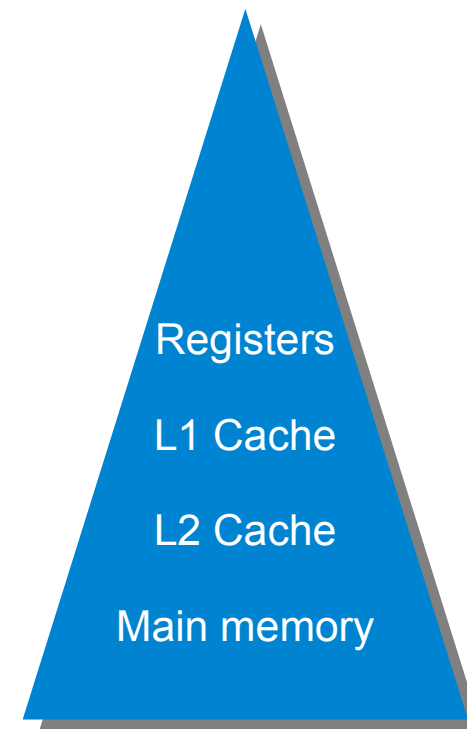
Basic Linear Algebra Subroutines

- Level 1 BLAS - vector operations
 - $O(n)$ data and flops
 - Memory bound:
 - $O(1)$ flops per memory access
 - ~ 2 Gflop/s on Skylake (not per core)
- Level 2 BLAS - matrix-vector operations
 - $O(n^2)$ data and flops
 - Memory bound:
 - $O(1)$ flops per memory access
 - ~ 4 Gflop/s on Skylake (not per core)
- Level 3 BLAS - matrix-matrix operations
 - $O(n^2)$ data, $O(n^3)$ flops
 - Surface-to-volume effect
 - Compute bound:
 - $O(n)$ flops per memory access
 - As high as 80 Gflop/s per core on Skylake

$$y = \alpha x + \beta y$$


$$y = \alpha A x + \beta y$$


$$C = \alpha A B + \beta C$$




Innovative Computing Laboratory's Libraries

- MAGMA
 - Matrix Algebra for GPUs and Multicore Architectures
- PLASMA
 - Parallel Linear Algebra Software for Multicore Architectures
- SLATE
 - Scalable Linear Algebra Targeting Exascale

Academic Collaborations



University of Colorado
Denver



Funding / Licensing



National Science Foundation

فہم لہو تجو تح بن تجو لا مم تجو بن بد بنین لا تح تجو مم تجو خ ے



**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.

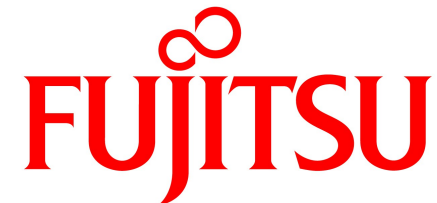


The objective of the Innovative Computing Laboratory's IPCC is the development and optimization of numerical linear algebra libraries and technologies for applications, while tackling current challenges in heterogeneous Intel® Xeon Phi™ coprocessor-based high-performance computing. In collaboration with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

www.netlib.org

LAPACK

ScaLAPACK

BLAS

CBLAS

LAPACKe

icl.utk.edu/research

PLASMA

dense linear algebra
(multicore)

MAGMA

dense linear algebra
(accelerators)

SLATE

dense linear algebra
(distributed memory + accelerators)

**new software
for multicore
and accelerators**

Software Projects - Runtimes

www.netlib.org

LAPACK

ScaLAPACK

BLAS

CBLAS

LAPACKe

icl.utk.edu/research

PLASMA

MAGMA

SLATE

QUARK

scheduling
(multicore)

PaRSEC

scheduling
(distributed memory)

dynamic
runtime
schedulers

Software Projects – New Runtimes

netlib.org

LAPACK

ScaLAPACK

LAPACKE

CBLAS

BLAS

icl.utk.edu/research

PLASMA

MAGMA

SLATE

OpenMP

scheduling
(multicore)

PaRSEC

scheduling
(distributed memory)

dynamic
runtime
schedulers

LAPACK++

<https://bitbucket.org/icl/lapackpp>

BLAS++

<https://bitbucket.org/icl/blaspp>

Software Projects - HPC

www.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**

BLAS Naming Scheme and Level 1 Examples

- One or two letter data type
 - l = integer (e.g., index)
 - s = single (float)
 - d = double
 - c = single-complex
 - z = double-complex
- Two letter matrix type (BLAS 2, 3)
 - ge = general nonsymmetric
 - sy = symmetric ($A = A^T$)
 - he = complex Hermitian ($A = A^H$)
 - tr = triangular (L or U)
 - Also banded and packed formats
- Two or more letter function, e.g.
 - mv = matrix-vector product
 - mm = matrix-matrix product
- BLAS 1 examples
 - sdot result = $x^T y$ (single)
 - ddot result = $x^T y$ (double)
 - cdotc result = $x^H y$ (single-complex)
 - cdotu result = $x^T y$ (single-complex)
 - zdotc result = $x^H y$ (double/complex)
 - zdotu result = $x^T y$ (double/complex)
 - _axpy $y = \alpha x + y$
 - _scal $y = \alpha y$
 - _copy $y = x$
 - _swap $x \leftrightarrow y$
 - _nrm2 $\|x\|_2$
 - _asum $\sum |x|$
 - i_amax $\arg \max |x|$
 - _rot Apply Given's rotation

BLAS Naming Scheme and Levels 2 and 3 Examples

- One or two letter data type
 - l = integer (e.g., index)
 - s = single (float)
 - d = double
 - c = single-complex
 - z = double-complex
- Two letter matrix type (BLAS 2, 3)
 - ge = general non-symmetric
 - sy = symmetric ($A = A^T$)
 - he = complex Hermitian ($A = A^H$)
 - tr = triangular (L or U)
 - Also banded and packed formats
- Two or more letter function, e.g.
 - mv = matrix-vector product
 - mm = matrix-matrix product
- BLAS 2 examples
 - `_gemv` $y = Ax + y$, A general
 - `_symv` $y = Ax + y$, A symmetric
 - `_hemv` $y = Ax + y$, A Hermitian
 - `_ger` $C = xy^T + C$, C general
 - `_syr` $C = xx^T + C$, C symmetric
 - `_her` $C = xx^H + C$, C Hermitian
 - `_trmv` $x = Ax$, A triangular
 - `_trsvsolve` $Ax = b$, A triangular
- BLAS 3 examples
 - `_gemm` $C = AB + C$ all matrices general
 - `_symm` $C = AB + C$ A symmetric
 - `_hemm` $C = AB + C$ A Hermitian
 - `_syrk` $C = AA^T + C$, C symmetric
 - `_herk` $C = AA^H + C$, C Hermitian
 - `_trmm` $X = AX$, A triangular
 - `_trsm` solve $AX = B$, A triangular

Dense Linear Algebra Problems

- linear systems of equations
- linear least squares
- singular value decomposition (SVD)
- eigenvalue value problems (EVP)

$$AX = B$$

$$\min \|B - AX\|_2$$

$$A = U\Sigma V^T$$

$$Ax = \lambda x$$

- dense (square, rectangular)
- band

Data Types

dgesv

Example	Precision	Description	C type
zgesv	z	double precision complex	double _Complex
cgesv	c	single precision complex	float _Complex
dgesv	d	double precision real	double
sgesv	s	single precision real	float

Data Types – Mixed Precision

dsgesv

Precision	Description	Example
dz	double-complex input, double precision result	cblas_dznrm2
sc	single-complex input, single precision result	cblas_scnrm2
zc	mixed-precision algorithm (double-complex/single-complex)	zcgsv
ds	mixed-precision algorithm (double/single)	dsgsv

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
_lascl	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 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

LAPACK

BLAS

CBLAS

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

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

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

includes: LAPACK, BLAS, CBLAS – reference implementations

ATLAS

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

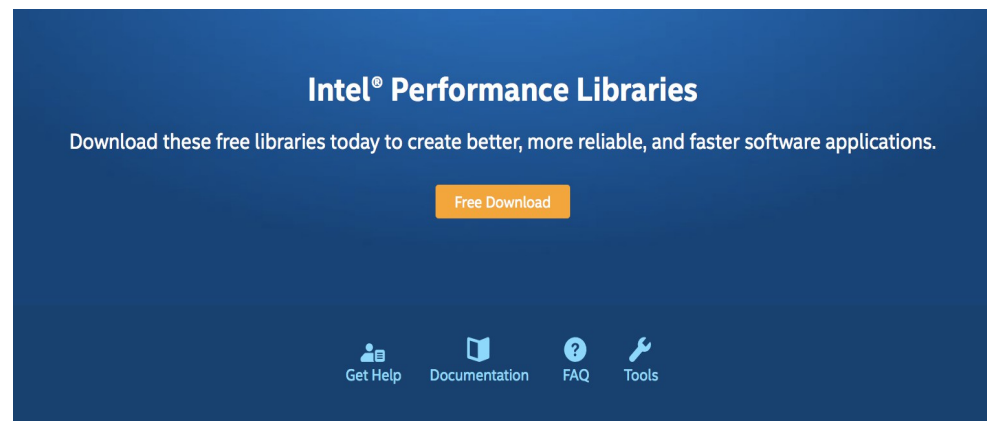
OpenBLAS

<http://www.openblas.net>

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

Intel Stack for Numerical Linear Algebra

- Intel Threading Building Blocks
- Intel Integrated Performance Primitives
- Intel Data Analytics Acceleration Library
- Intel Math Kernel Library
 - LAPACK
 - LAPACKe
 - BLAS
 - CBLAS
- Royalty free
 - Some sources available



Take advantage of powerful and award-winning performance libraries that optimize your code and shorten development time. These libraries are offered for free as part of Intel's mission to support innovation and impressive performance on Intel® architecture.



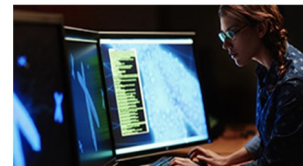
Intel® Math Kernel Library

This popular, fast math library for Intel® and other compatible processors features highly optimized, threaded, and vectorized functions to maximize performance on each processor family.



Intel® Integrated Performance Primitives

Gain a competitive performance advantage with this library that offers image, signal, compression, and cryptography functions for multiple operating systems and platforms.



Intel® Threading Building Blocks

Benefit from this widely used C++ library for shared-memory parallel programming and heterogeneous computing.

NVIDIA Software Stack



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

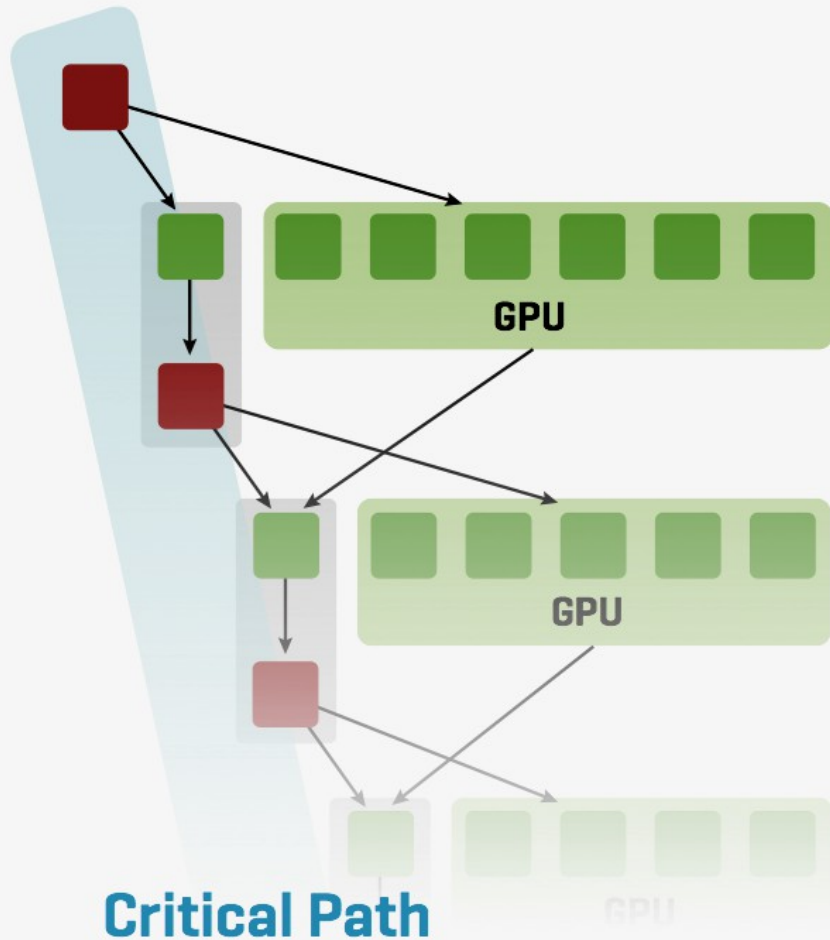
<http://icl.cs.utk.edu/magma/software/>



<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



GPU
CENTER OF
EXCELLENCE

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**
- ▶ **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

The screenshot displays the MAGMA website interface. At the top, the 'MAGMA Forum' header includes a search bar and the text 'User discussion for MAGMA library (Matrix Algebra on GPU and Multicore Architectures), http://icl.cs.utk.edu/magma/'. Below this, the 'MAGMA 2.1.0' banner is visible, followed by the tagline 'Matrix Algebra for GPU and Multicore Architectures'. The main content area is divided into two columns. The left column contains a 'User discussion' section with a 'NEWTOPIC*' button and a list of recent posts under 'ANNOUNCEMENTS' and 'TOPICS'. The right column features a 'Main Page' tab with a navigation menu (Main Page, Related Pages, Routines, Files) and a table of routines. The table has columns for 'Suffix', 'Example', and 'Description'. Below the table, a paragraph explains the naming conventions, and another table provides a mapping of 'Precision' to 'Description'.

Suffix	Example	Description
none	magma_dgetrf	hybrid CPU/GPU routine where the matrix is initially in CPU host memory.
_m	magma_dgetrf_m	hybrid CPU/multiple-GPU routine where the matrix is initially in CPU host memory.
_gpu	magma_dgetrf_gpu	hybrid CPU/GPU routine where the matrix is initially in GPU device memory.
_mgpu	magma_dgetrf_mgpu	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., scnrm2)
dz	double-complex input with double precision result (e.g., dznrm2)

Generated by [doxygen](#) 1.8.11

MAGMA Routines for LU Solver

Suffix	Example	Description
<empty>	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 LU Factorization

```
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);
```

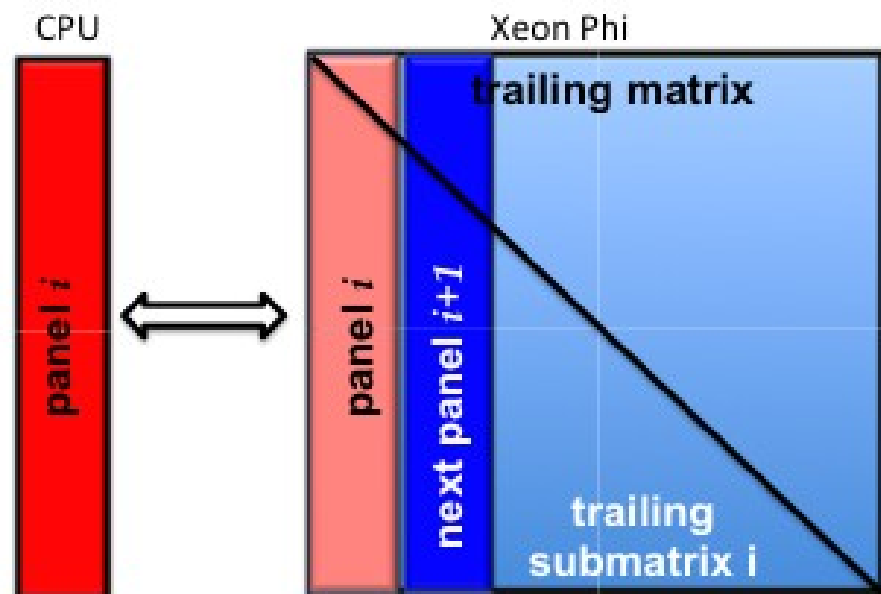
```
magma_zgetrf_mgpu(ngpu, M, N, d_lA, ldda, ipiv, &info);
```

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

```
magma_queue_destroy(queue);
```

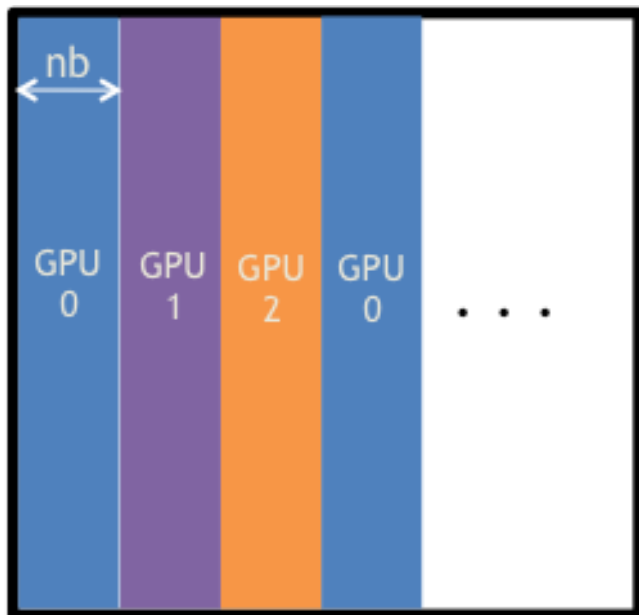
```
magma_finalize();
```

MAGMA Factorization Algorithms



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

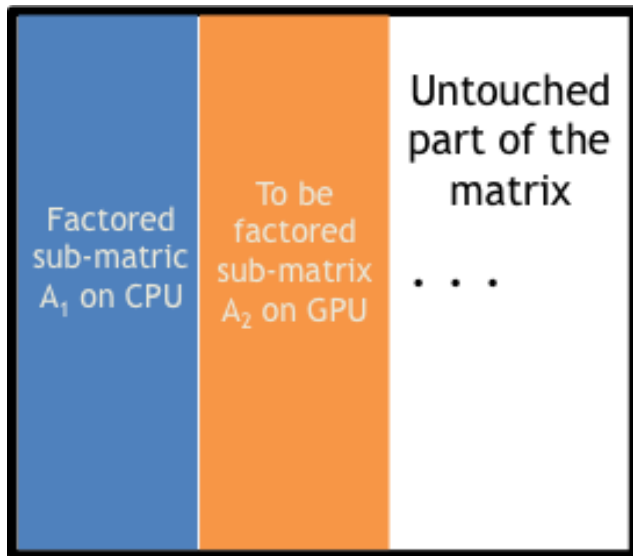
MAGMA Multi-GPU



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

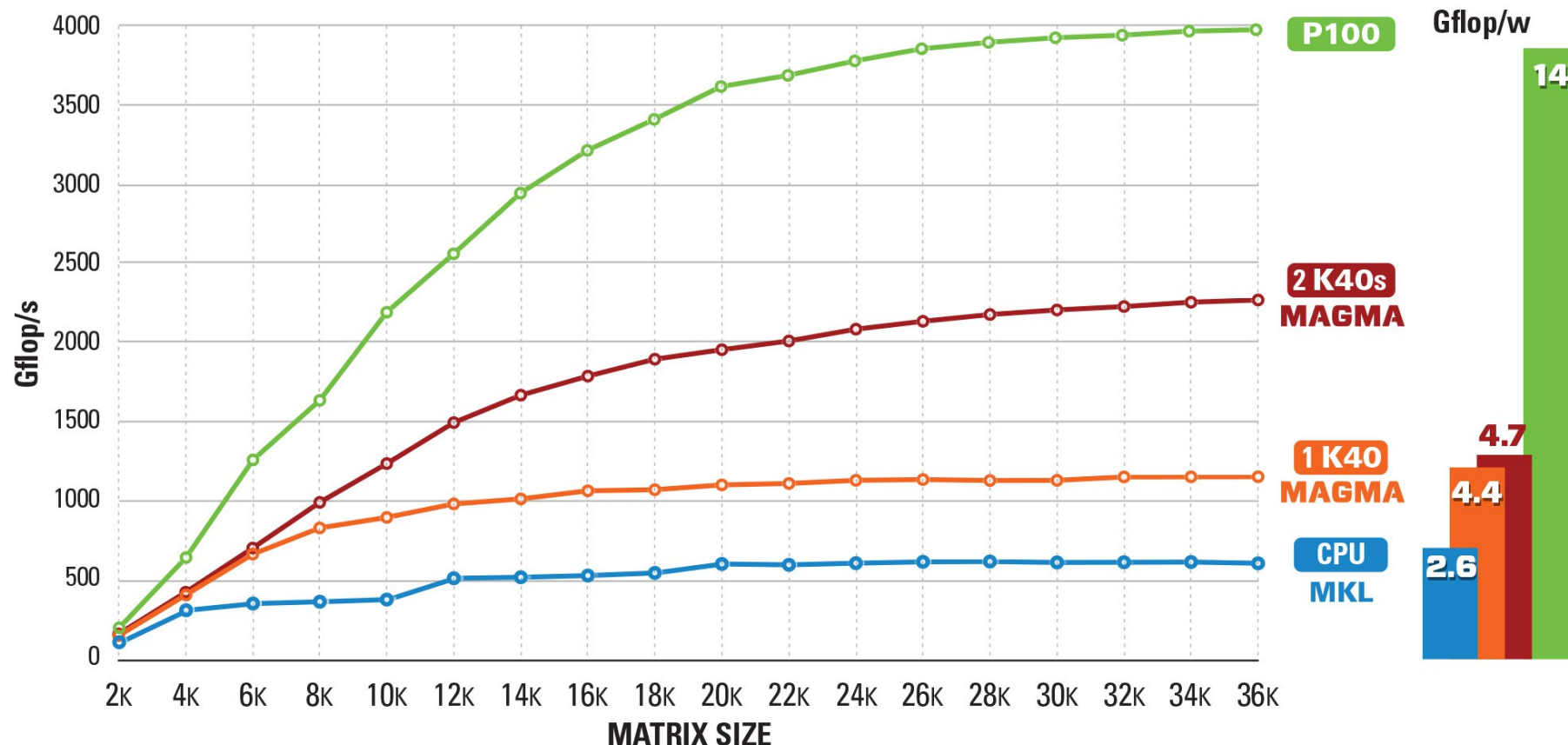
MAGMA Performance – LU Factorization

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

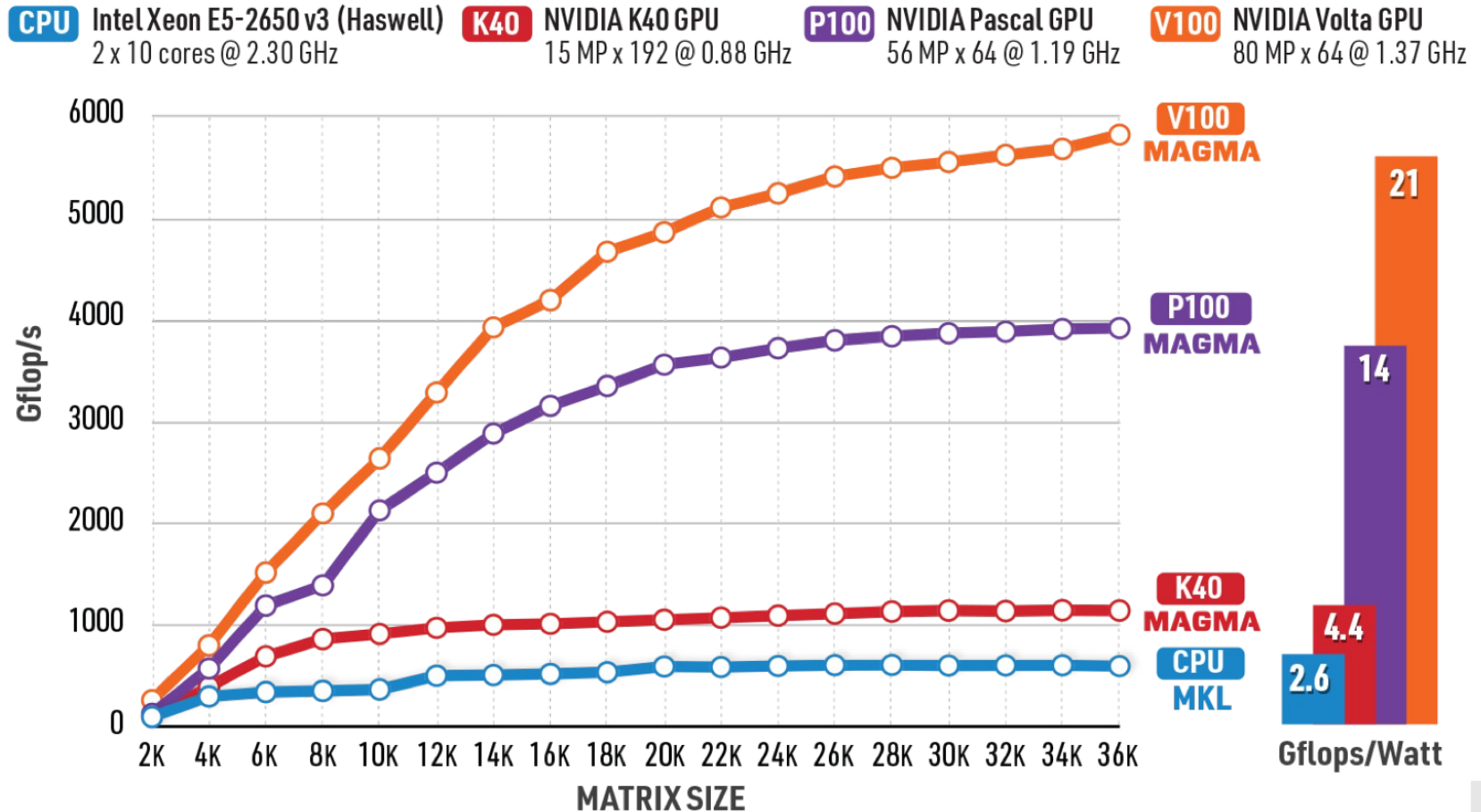
P100 NVIDIA Pascal GPU
56 MP x 64 @ 1.19 GHz



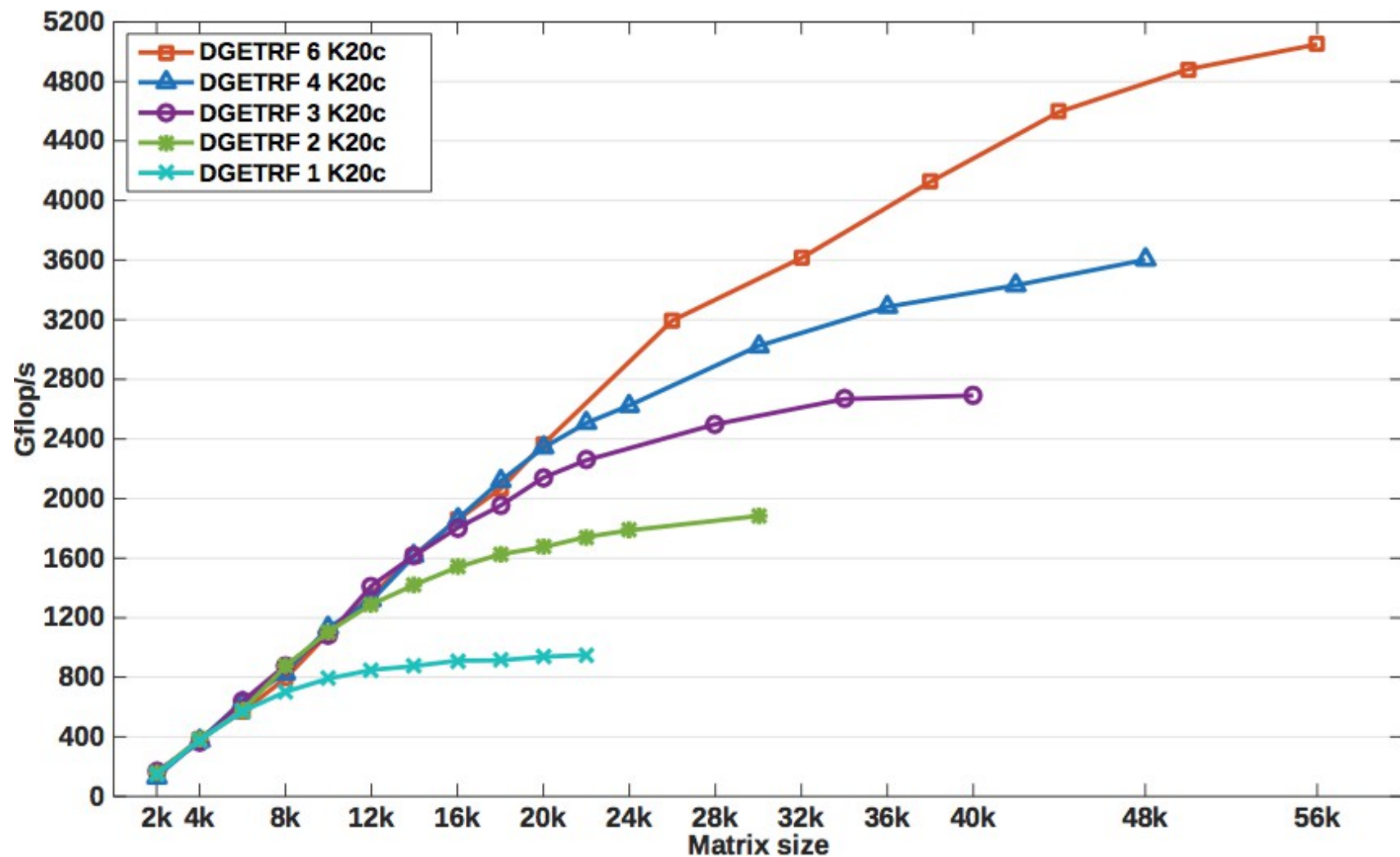
MAGMA Performance on NVIDIA Volta

PERFORMANCE & ENERGY EFFICIENCY

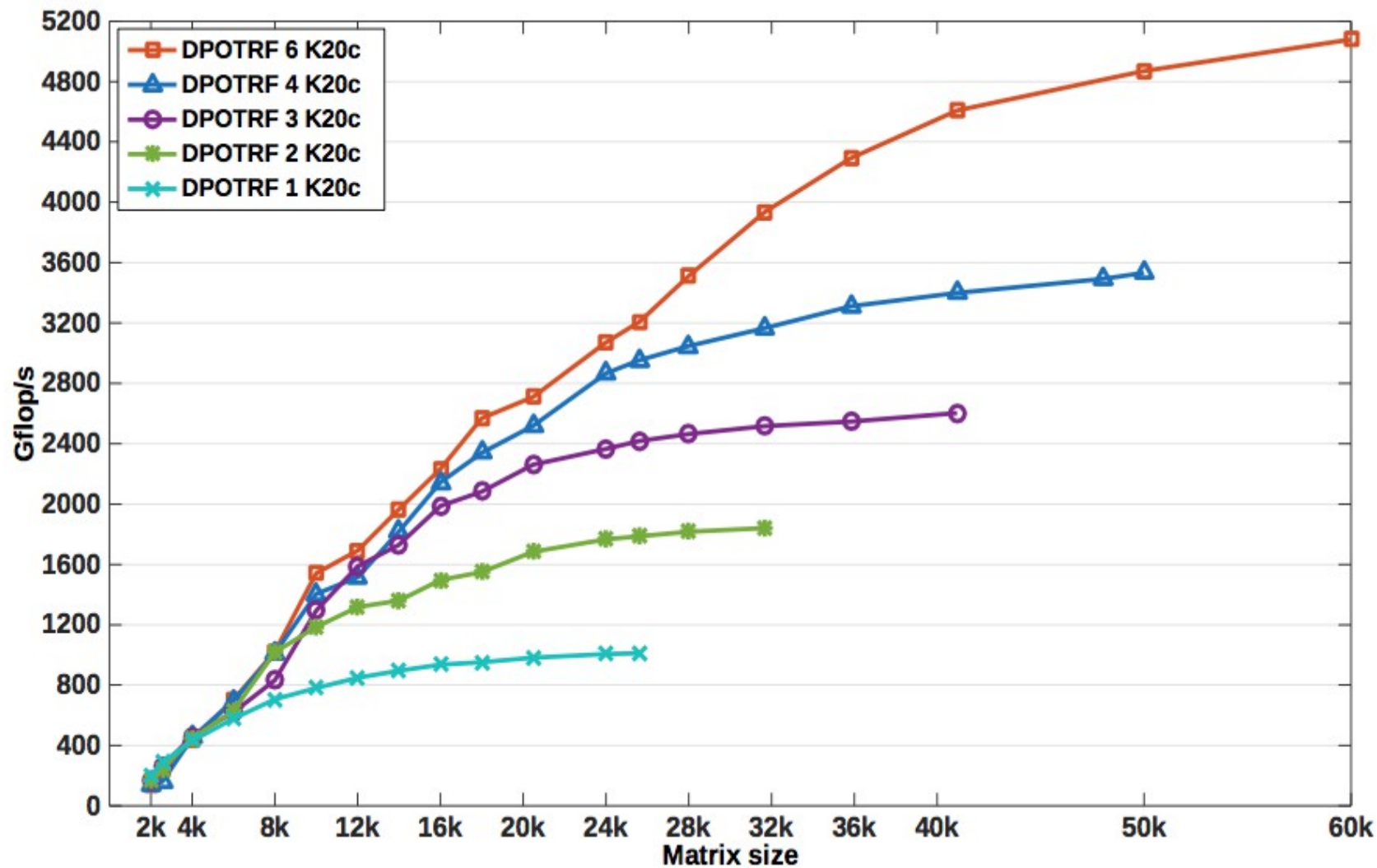
MAGMA LU factorization in double precision arithmetic



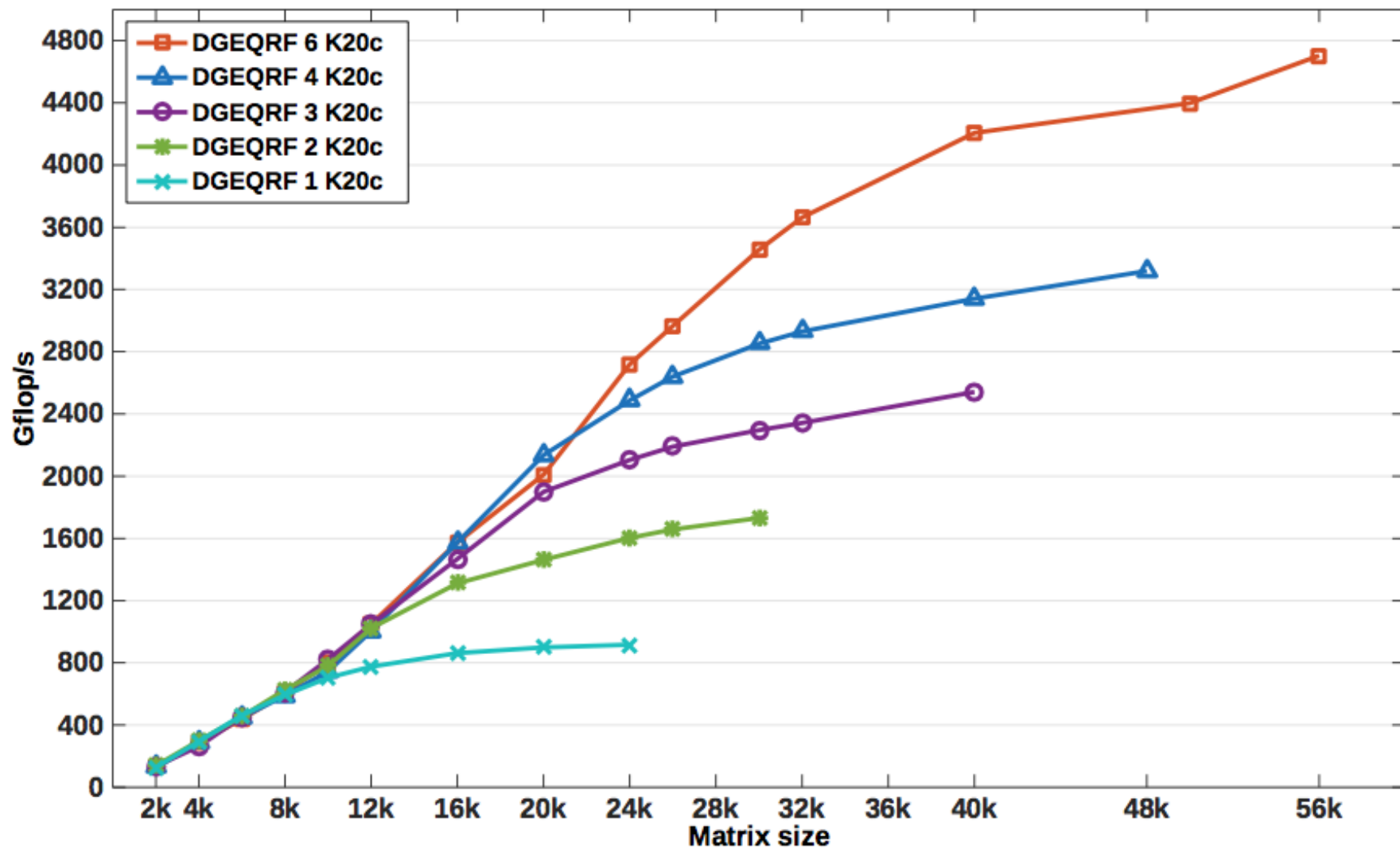
MAGMA Performance – DGETRF on Kepler K20c



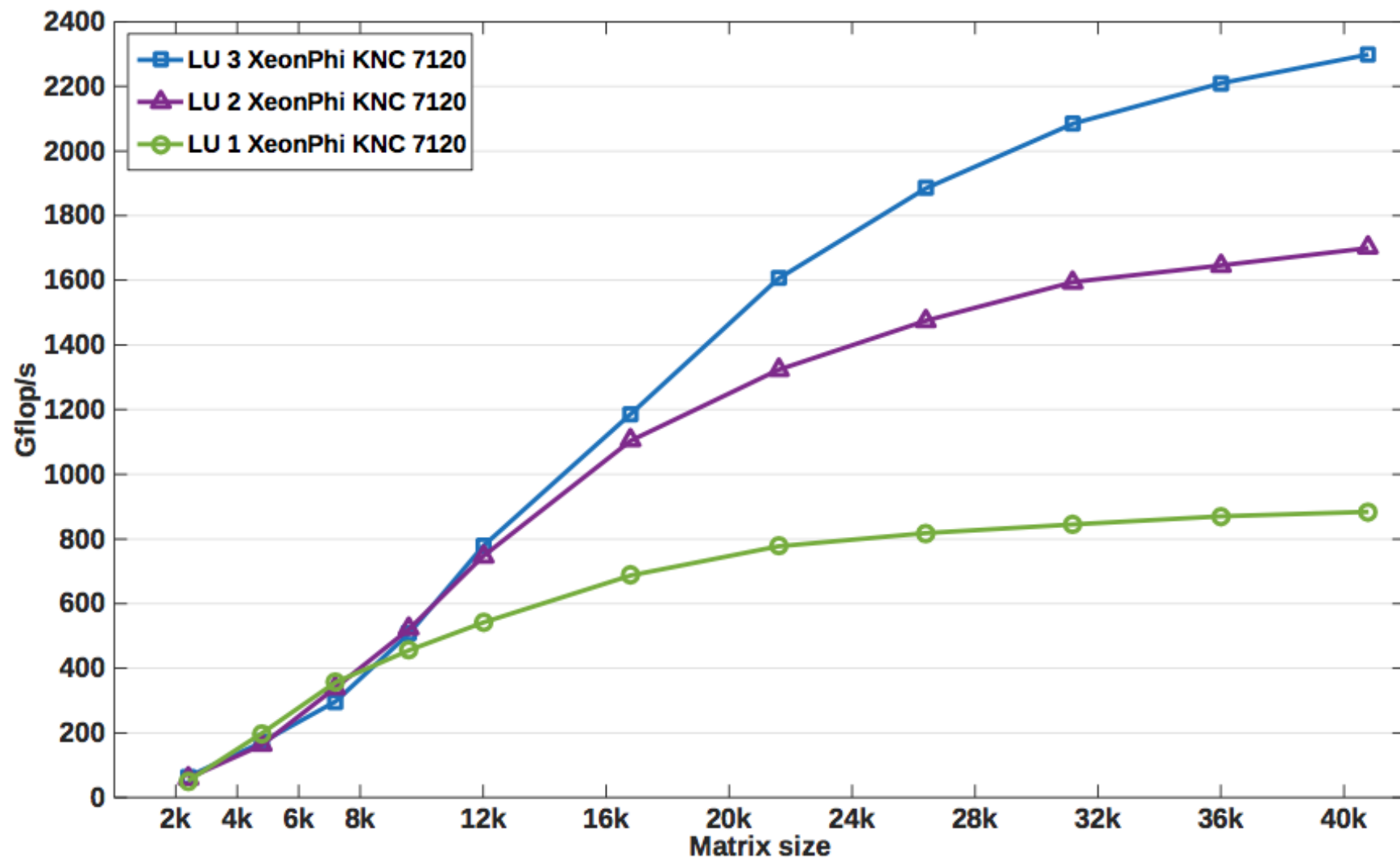
MAGMA Performance – DPOTRF on Kepler K20c



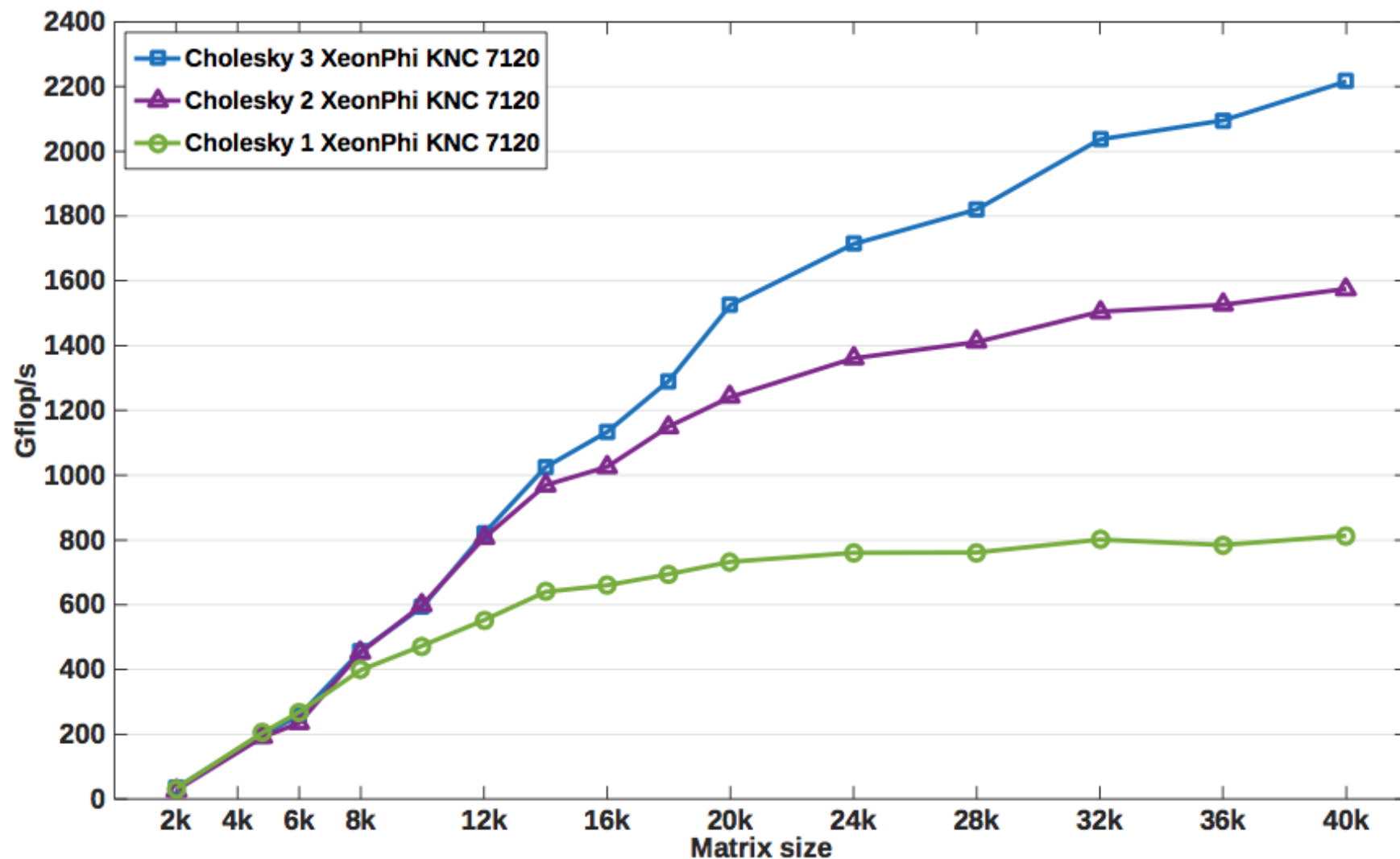
MAGMA Performance – DGEQRF on Kepler K20c



MAGMA Performance – DGETRF on Intel Xeon Phi KNC



MAGMA Performance – DPOTRF on Intel Xeon Phi KNC



MAGMA Performance – DGEQRF on Intel Xeon Phi KNC

