MAGMA:
Evolution and Revolution

Stan Tomov
Innovative Computing Laboratory
University of Tennessee, Knoxville

ICL Lunch Talk Seminar
July 2, 2021

2021

MAGMA 2.6

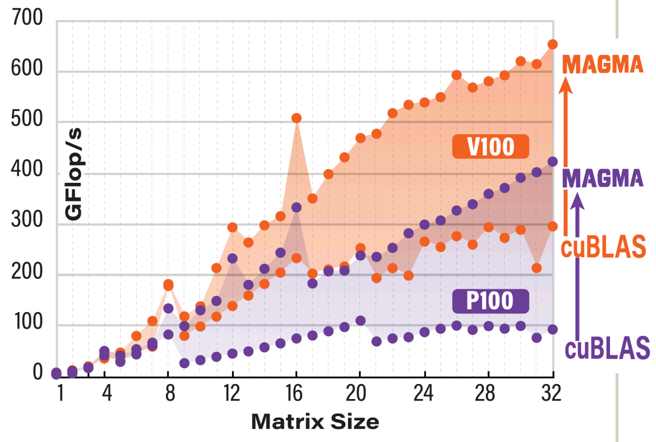
INNOVATIVE COMPUTING LABORATORY

MAGMA

- Shared memory systems
- BLAS/LAPACK on GPUs
- Hybrid CPU-GPU Algorithms
  - Linear system solvers (+ mixed precision)
  - Eigenvalue problem solvers
- Batched LA
  - BLAS-3 (fixed/variable), LU, QR, Cholesky
- Sparse LA
  - Solvers: BiCG, BiCGSTAB, GMRES
  - Preconditioners: ILU, Jacobi,
  - SPMV, SPMM (CSR, ELL, … etc.)

PERFORMANCE OF BATCHED LU

in double-precision arithmetic on 1 million matrices



Matrix Algebra on GPU and Multicore Architectures

PERFORMANCE & ENERGY EFFICIENCY

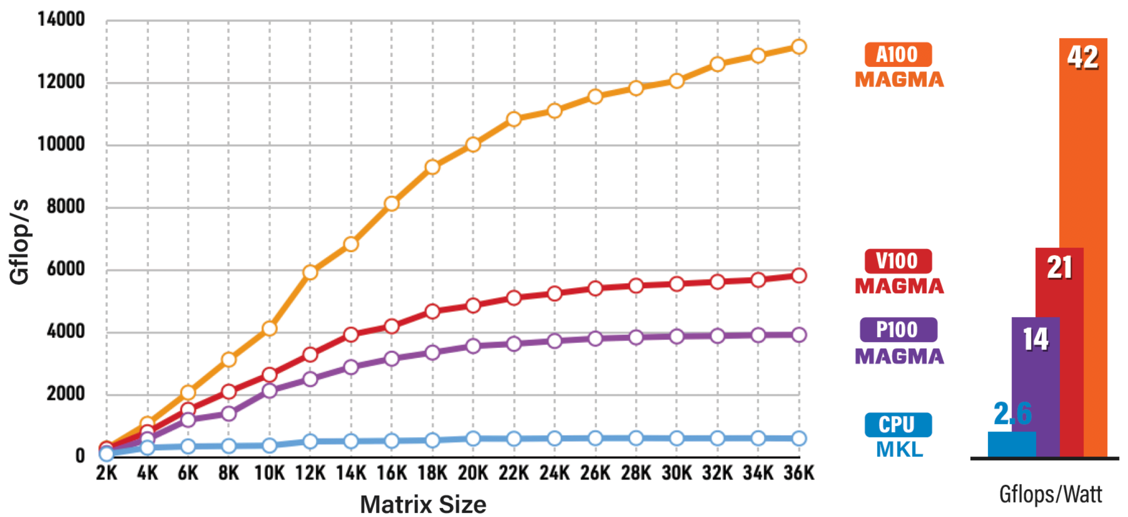
MAGMA LU factorization in double-precision arithmetic

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

P100 NVIDIA Pascal GPU 56 SMs x 64 @ 1.19 GHz

V100 NVIDIA Volta GPU 80 SMs x 64 @ 1.37 GHz

A100 NVIDIA Ampere GPU 108 SMs x 64 @ 1.41 GHz



- Support CUDA and ROCM/HIP
- https://icl.utk.edu/magma

IN COLLABORATION WITH: Berkeley University of California; University of Colorado Denver; Inria; KAUST

SPONSORED BY: U.S. Department of Defense; U.S. Department of Energy; ECP; National Science Foundation

WITH SUPPORT FROM: AMD; Intel; MathWorks; NVIDIA

GPUs: excelling in graphics rendering

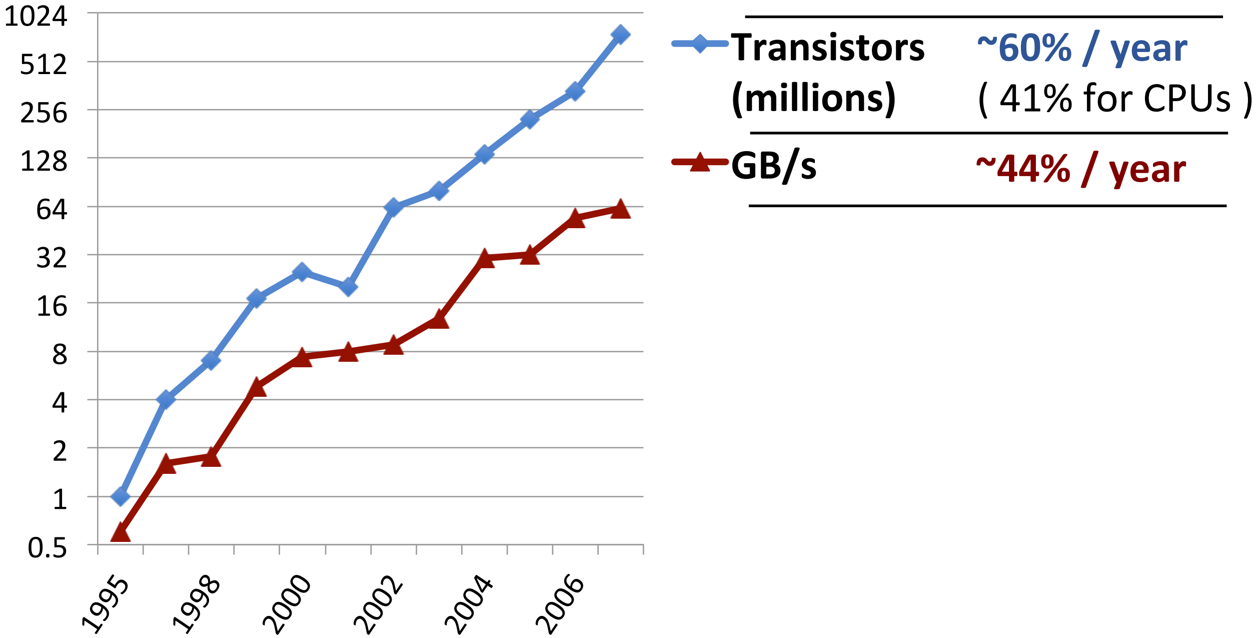
Scene model → streams of data → Graphics pipelined computation → Final image

Repeated fast over and over: e.g. TV refresh rate is 30 fps; limit is 60 fps

This type of computation:

- Requires **enormous computational power**
- Allows for **high parallelism**
- Needs **high bandwidth**

GPU Evolution from 1999 to 2007 (GeForce1 to GeForce8 Series)

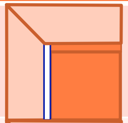
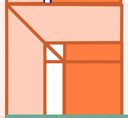
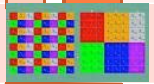
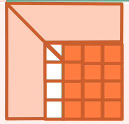


- Some applications were exploring GPUs use
- Some LA was developed but was "slow"
  - **No memory hierarchy for data reuse**
  - E.g., even a dot product would be computed using the graphics pipeline with multiple passes through the data to reduce it to single number

2007 — 2021

CUDA

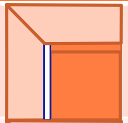
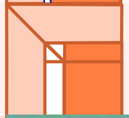
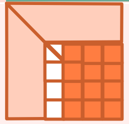
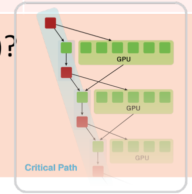
MAGMA 2.6

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
LAPACK for GPUs (00's)?		

Year	GB/s	SP (Gflop/s)	CUDA GPUs
2007	62	416	GeForce 8800 GTS (Tesla)
2008	70	470	GeForce 9800 GTX (Tesla)

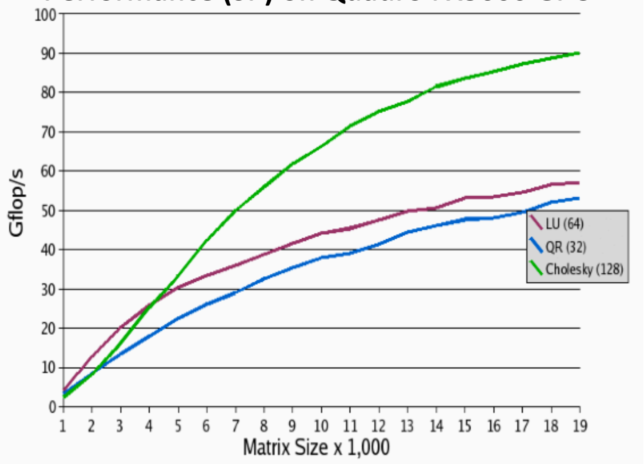
- CUDA first appears in 2007
- CUDA GPUs have **memory hierarchy**
  - Enables development of more efficient LA
  - BLAS can be efficiently implemented
  - More applications start to explore GPU use

07 CUDA — 08 Hybrid algorithms — 2021 **MAGMA 2.6**

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
LAPACK for GPUs (00's)? Hybrid algorithms (CPU + **GPU**)		

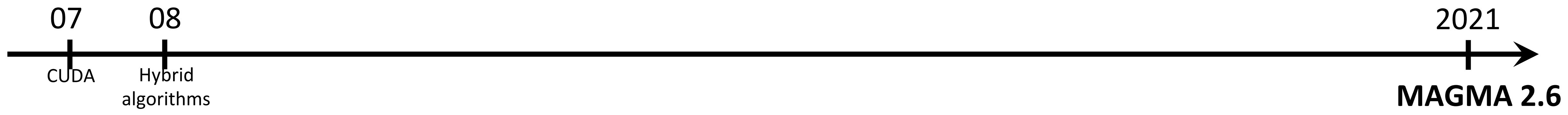
Year	GB/s	SP (Gflop/s)	CUDA GPUs
2007	62	416	GeForce 8800 GTS (Tesla)
2008	70	470	GeForce 9800 GTX (Tesla)

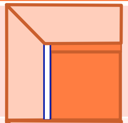
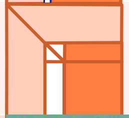
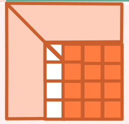
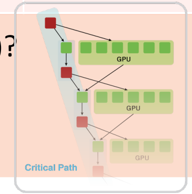
Performance (SP) on Quadro FX5600 GPU



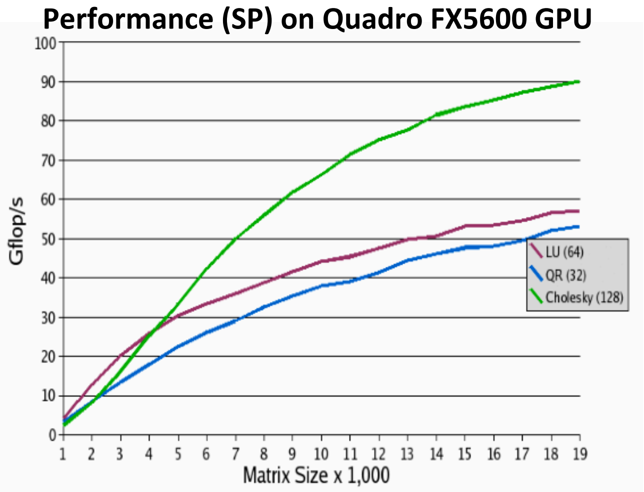
[1] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also at PARA2008 and LAPACK Working Note 200), January 2008.

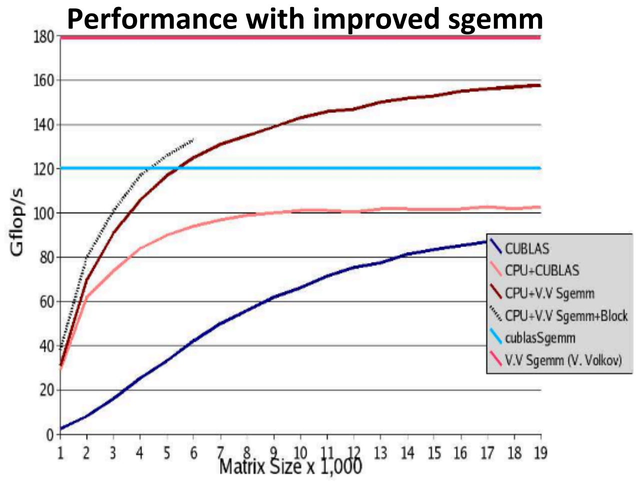
- 1st **hybrid CPU+GPU LA algorithms**
- LU, QR, and Cholesky
- LU to avoid pivoting (RBT)



Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
LAPACK for GPUs (00's)? Hybrid algorithms (CPU + **GPU**)		

Year	GB/s	SP (Gflop/s)	CUDA GPUs
2007	62	416	GeForce 8800 GTS (Tesla)
2008	70	470	GeForce 9800 GTX (Tesla)





[1] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also at PARA2008 and LAPACK Working Note 200), May 6, 2008.

V. Volkov and J. W. Demmel, "**Using GPUs to accelerate linear algebra routines**", Poster at PAR lab winter retreat, January 9, 2008.

V. Volkov and J. W. Demmel, "LU, QR, and Cholesky Factorizations using Vector Capabilities of GPUs", LAPACK Working Note 202, May 15, 2008.

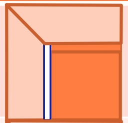
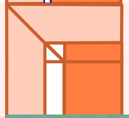
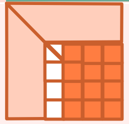
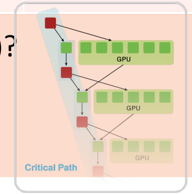
- 1st **hybrid CPU+GPU LA algorithms**
- LU, QR, and Cholesky
- LU to avoid pivoting (RBT)

- Very efficient sgemm (61% of peak) vs. CUBLAS 1.1 (37% of peak)
- Volkov and Demmel made the code code available and we used it to further accelerate LU, QR, and Cholesky

07 — CUDA

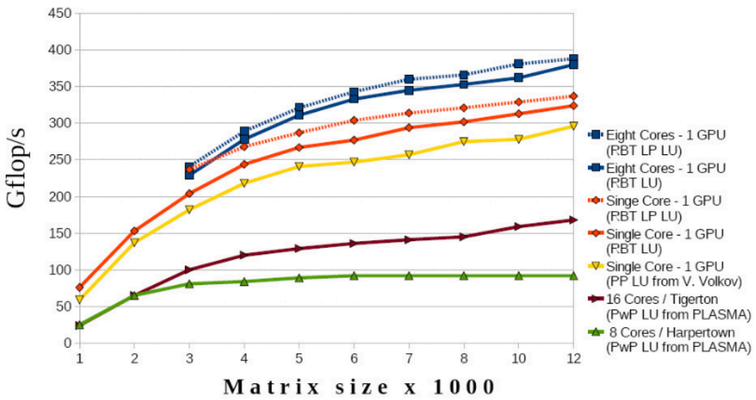
08 — Hybrid algorithms

2021 — **MAGMA 2.6**

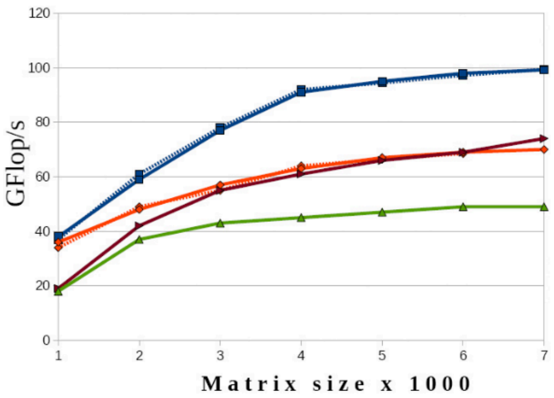
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
LAPACK for GPUs (00's)? Hybrid algorithms (CPU + **GPU**)		**Rely on** **- BLAS** tasking **- Hybrid scheduling**

Year	GB/s	SP (Gflop/s)	CUDA GPUs
2007	62	416	GeForce 8800 GTS (Tesla)
2008	70	470	GeForce 9800 GTX (Tesla)

LU and RBT LU performance (SP) on GTX 280 GPU



Performance in DP



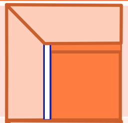
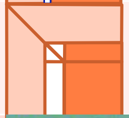
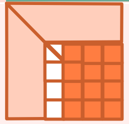
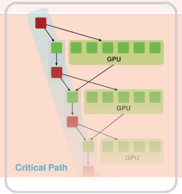
[1] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also at PARA2008 and LAPACK Working Note 200), May 6, 2008.

[2] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also LAPACK Working Note 200), October 2008.

- 1st **hybrid CPU+GPU LA algorithms**
- LU, QR, and Cholesky
- LU to avoid pivoting (RBT)

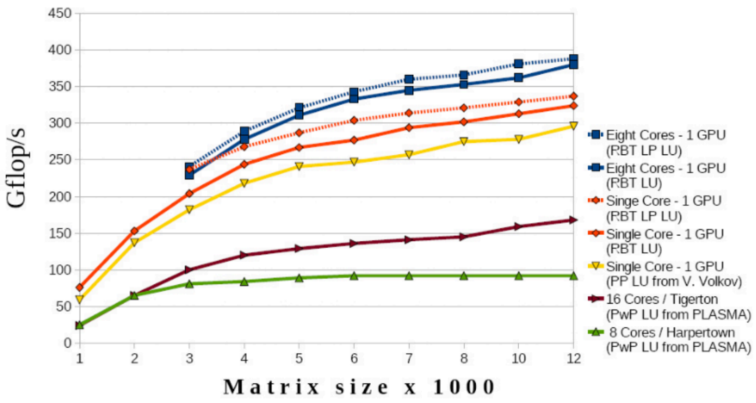
- **Look-ahead scheduling** to overlap CPU work with GPU
- Computing and performance in DP

07 CUDA — 08 Hybrid algorithms **MAGMA** — 2021 **MAGMA 2.6**

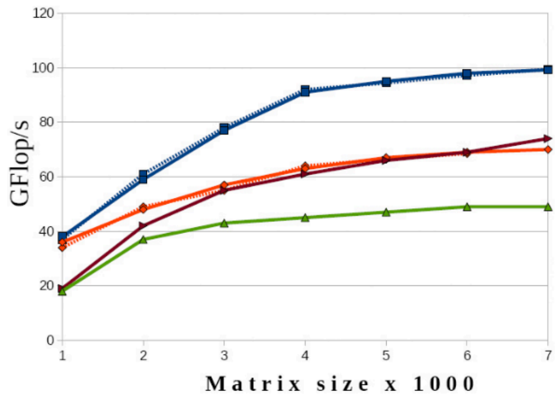
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
MAGMA (00's) Hybrid algorithms (CPU + **GPU**)		**Rely on - BLAS tasking - Hybrid scheduling**

Year	GB/s	SP (Gflop/s)	CUDA GPUs
2007	62	416	GeForce 8800 GTS (Tesla)
2008	70	470	GeForce 9800 GTX (Tesla)

LU and RBT LU performance (SP) on GTX 280 GPU



Performance in DP



[1] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also at PARA2008 and LAPACK Working Note 200), May 6, 2008.

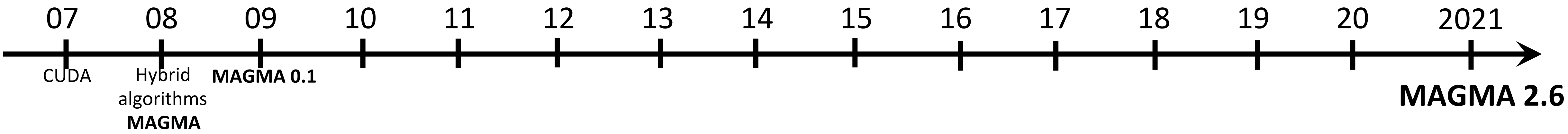
[2] Baboulin, M., J. Dongarra, and S. Tomov, "**Some Issues in Dense Linear Algebra for Multicore and Special Purpose Architectures,**" University of Tennessee Computer Science Technical Report, UT-CS-08-615 (also LAPACK Working Note 200), October 2008.

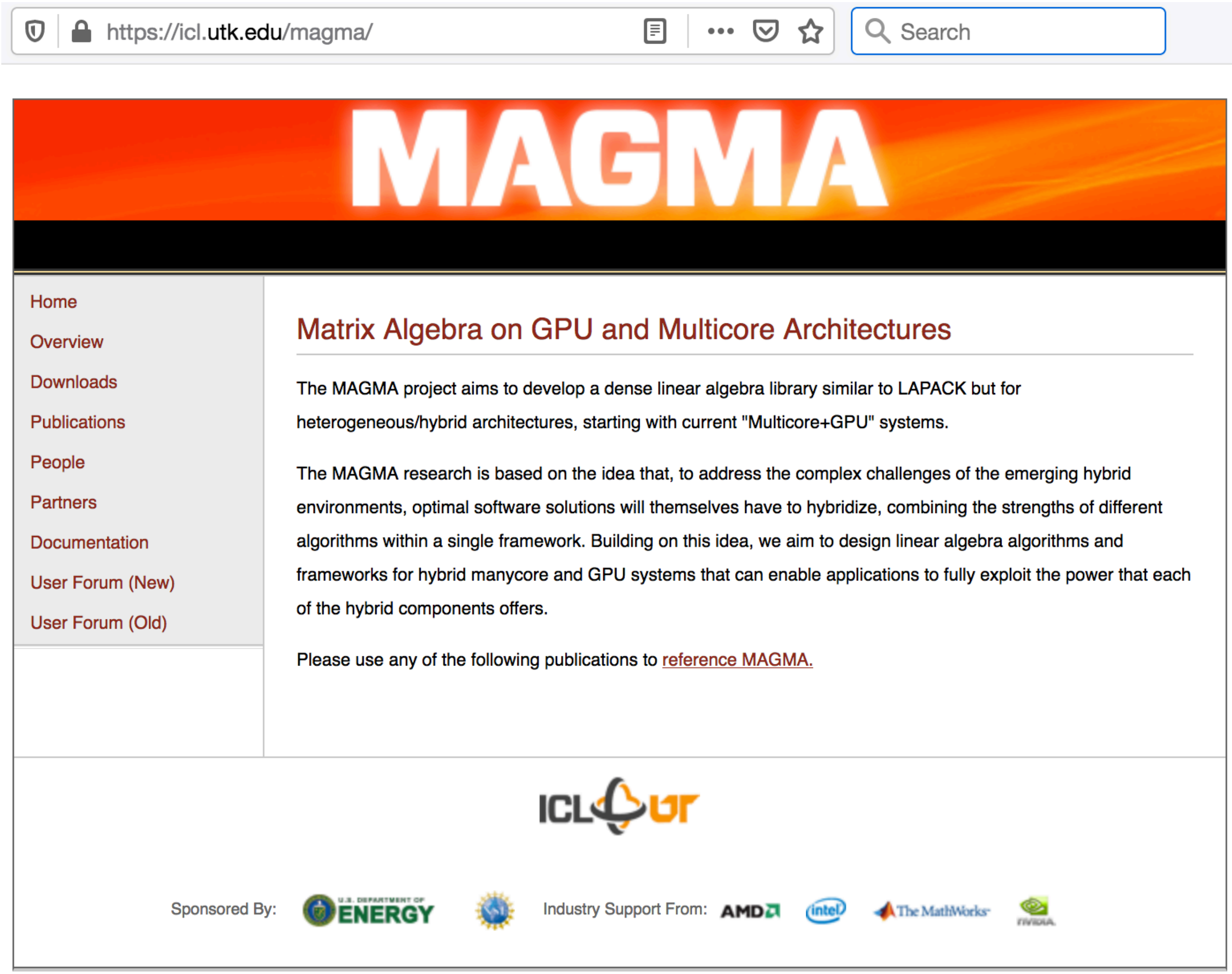
[3] Baboulin, M., J. Demmel, J. Dongarra, S. Tomov, and V. Volkov, **"Enhancing the Performance of Dense Linear Algebra Solvers on GPUs (in the MAGMA Project)"** , Austin, TX, The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC08), Poster, November 2008.

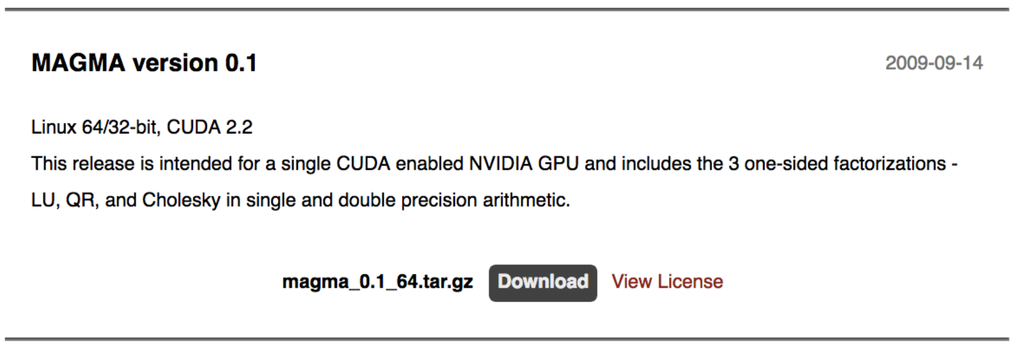
- 1st **hybrid CPU+GPU LA algorithms**
- LU, QR, and Cholesky
- LU to avoid pivoting (RBT)

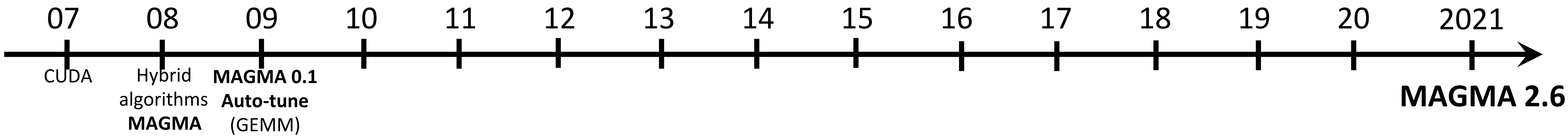
- **Look-ahead scheduling** to overlap CPU work with GPU
- Computing and performance in DP

- First mention of **MAGMA** project

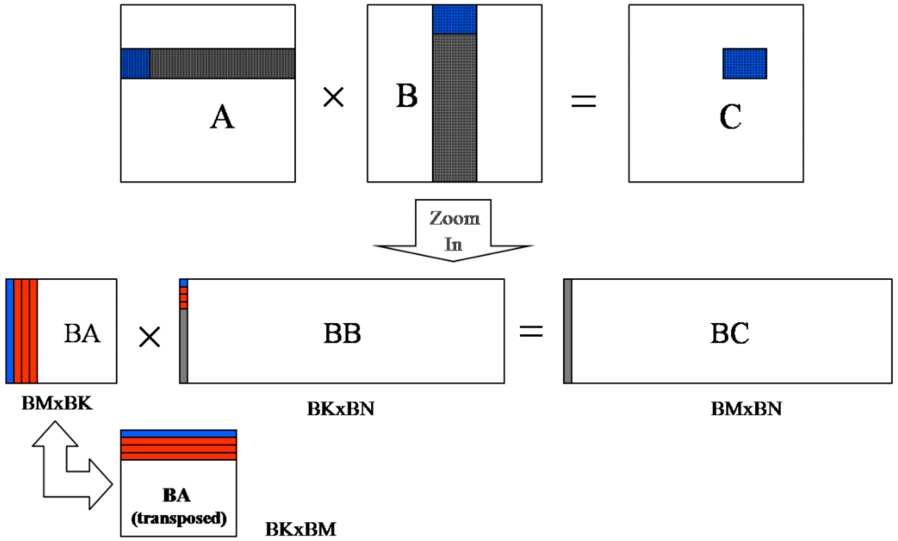

07
08
09
10
11
12
13
14
15
16
17
18
19
20
2021
CUDA
Hybrid algorithms
MAGMA
MAGMA 0.1
MAGMA 2.6


https://icl.utk.edu/magma/
Search
MAGMA
Home
Overview
Downloads
Publications
People
Partners
Documentation
User Forum (New)
User Forum (Old)
Matrix Algebra on GPU and Multicore Architectures
The MAGMA project aims to develop a dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures, starting with current "Multicore+GPU" systems.
The MAGMA research is based on the idea that, to address the complex challenges of the emerging hybrid environments, optimal software solutions will themselves have to hybridize, combining the strengths of different algorithms within a single framework. Building on this idea, we aim to design linear algebra algorithms and frameworks for hybrid manycore and GPU systems that can enable applications to fully exploit the power that each of the hybrid components offers.
Please use any of the following publications to reference MAGMA.
Sponsored By:
Industry Support From:


MAGMA version 0.1
2009-09-14
Linux 64/32-bit, CUDA 2.2
This release is intended for a single CUDA enabled NVIDIA GPU and includes the 3 one-sided factorizations - LU, QR, and Cholesky in single and double precision arithmetic.
magma_0.1_64.tar.gz
Download
View License



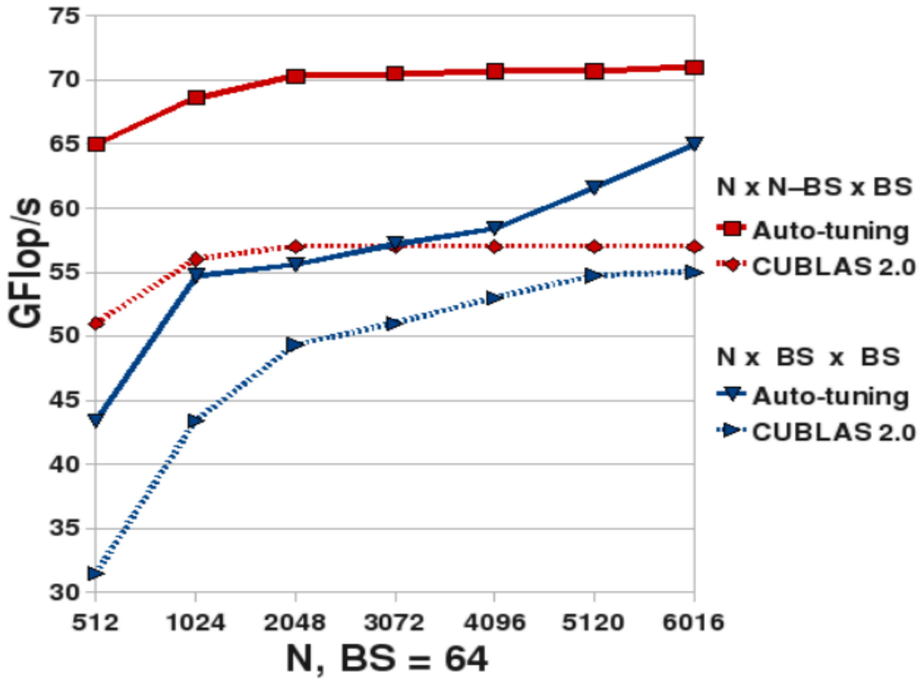
Algorithmic view of template code for GEMM

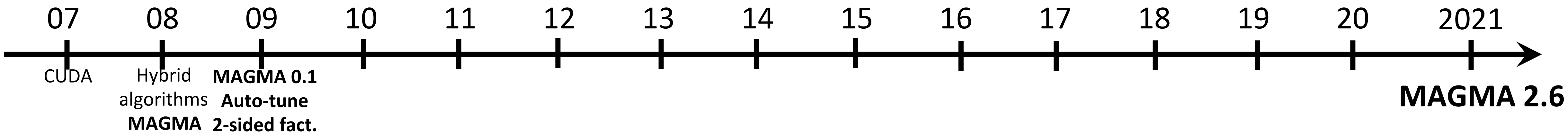


[1] Y. Li, J. Dongarra, and S. Tomov, "**A note on auto-tuning GEMM for GPUs,**" International Conference on computational Science (ICCS 2009), 2009

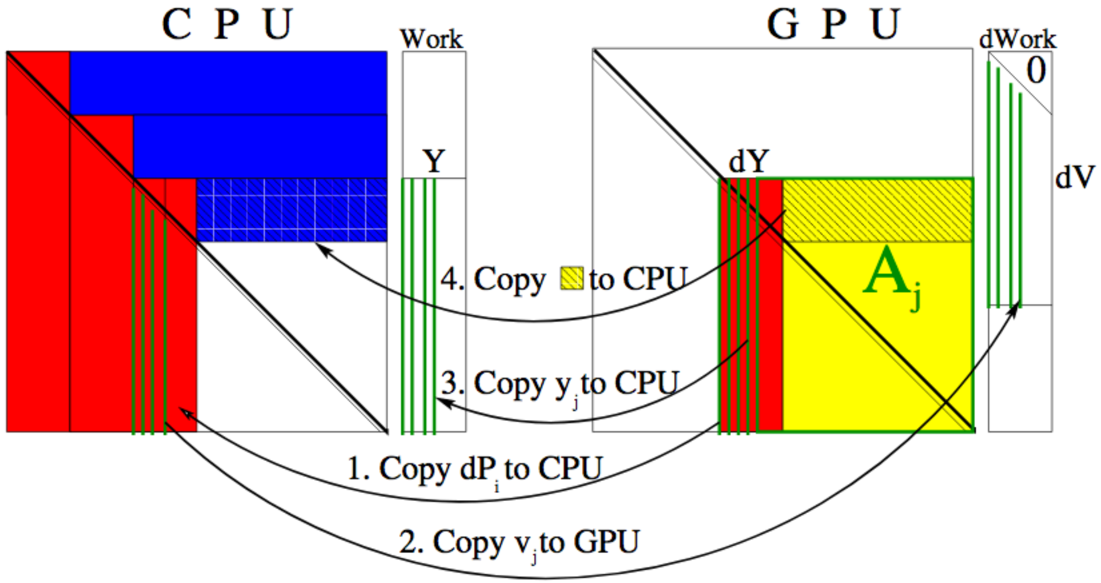
- GEMM performance is crucial for performance of LA (GEMM is about 12x faster than GEMV at the time)
- Parameterize and template code and use empirical **auto-tuning**
- **Obtained substantial, up to 27%, speedup compared to CUBLAS 2.0**

Auto-tuning rank-k update DGEMM (k=64)
(on GeForce GTX 280 GPU)



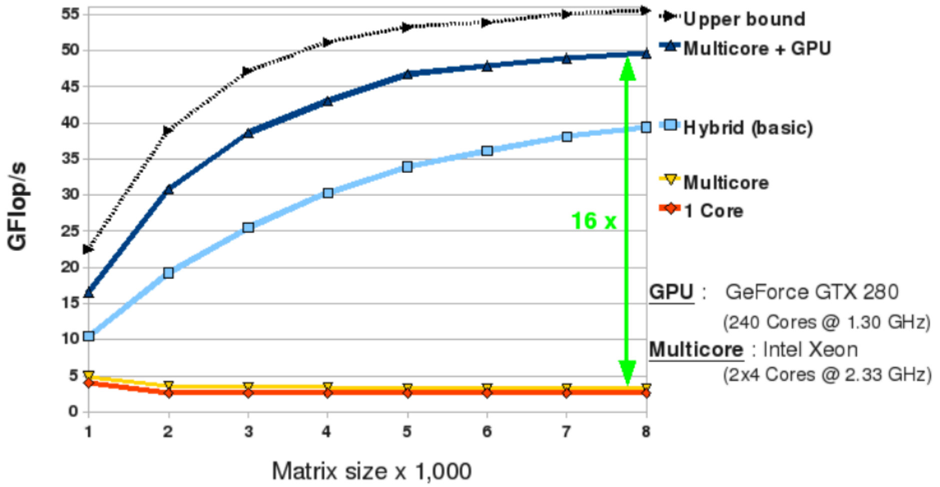


Hybrid algorithms for two-sided factorizations

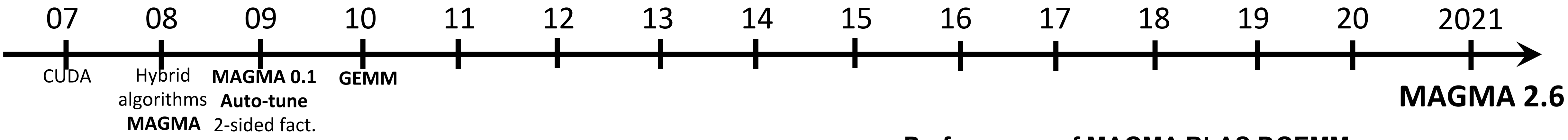


- Eigen-solvers need two-sided factorizations but are notoriously slow on CPUs due to need of Level 2 BLAS computations
- Develop Level 2 BLAS to benefit GPU's high-bandwidth
- **Developed hybrid CPU+GPU** two-sided factorizations (Hessenberg for general eigen-solvers, bidiagonal for SVD, and tridiagonal for symmetric eigen-solvers)

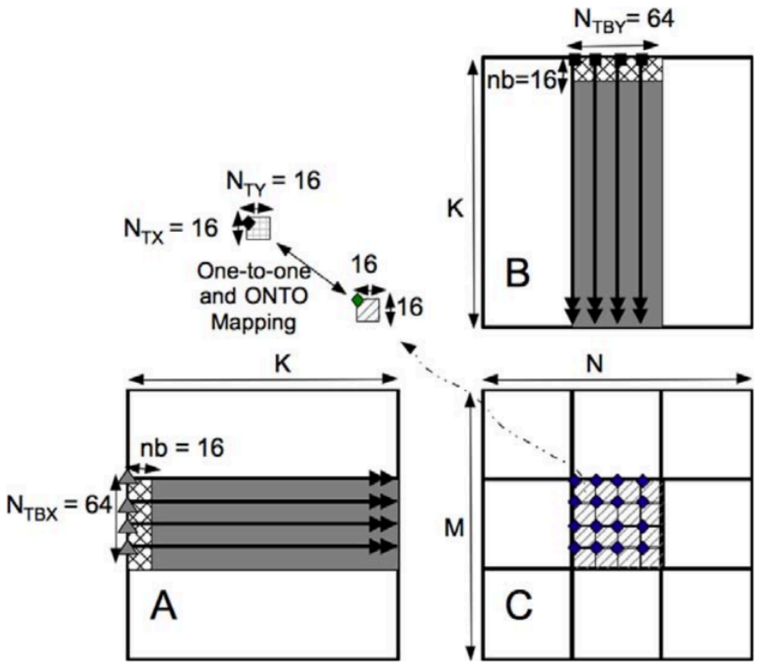
Performance in DP of Hessenberg reduction on GTX 280 GPU



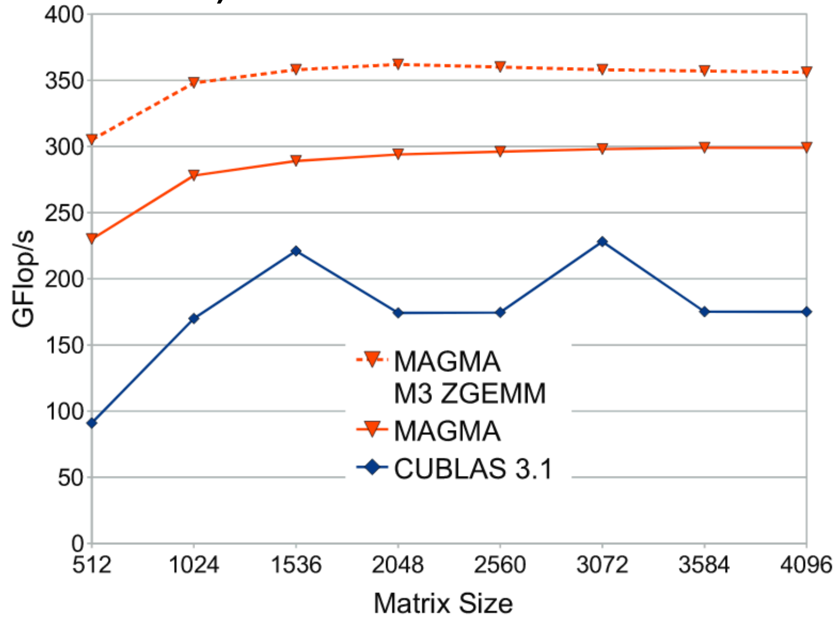
[1] Y. Li, J. Dongarra, and S. Tomov, "**A note on auto-tuning GEMM for GPUs,**" International Conference on computational Science (ICCS 2009), 2009.
[2] Tomov, S., and J. Dongarra, "**Accelerating the Reduction to Upper Hessenberg Form through Hybrid GPU-Based Computing**," University of Tennessee Computer Science Technical Report, UT-CS-09-642 (also LAPACK Working Note 219), May 2009.



Algorithmic view of template code of GEMM for Fermi

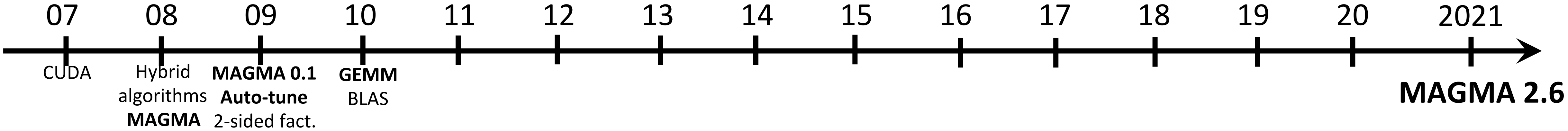


Performance of MAGMA BLAS DGEMM (and ZGEMM) vs. CUBLAS 3.1 on Fermi C2050 GPU

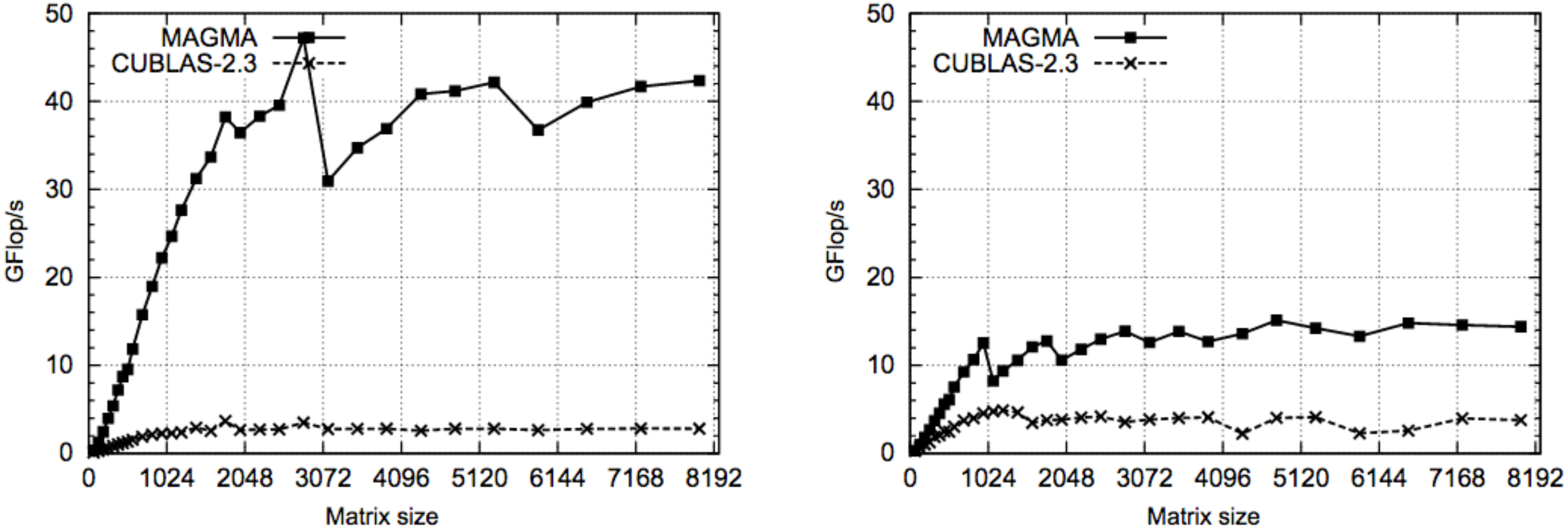


[1] R. Nath, S. Tomov, and J. Dongarra, "**An improved MAGMA GEMM for Fermi GPUs,**" International Journal of High Performance Computing Applications, vol. 24, no. 4, 2010.

- **Best GEMM performance** at the time (58% of peak in DP and 63% of peak in SP)
- **Added register blocking** to previous state-of-art
- Incorporated in CUBLAS
- Still used today
  - although its currently best implementations are in assembly



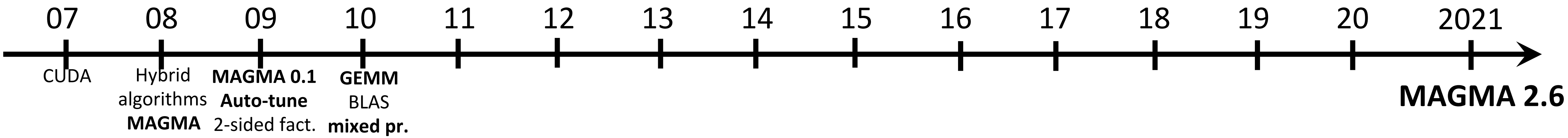
Performance of xSYMV on a GTX 280



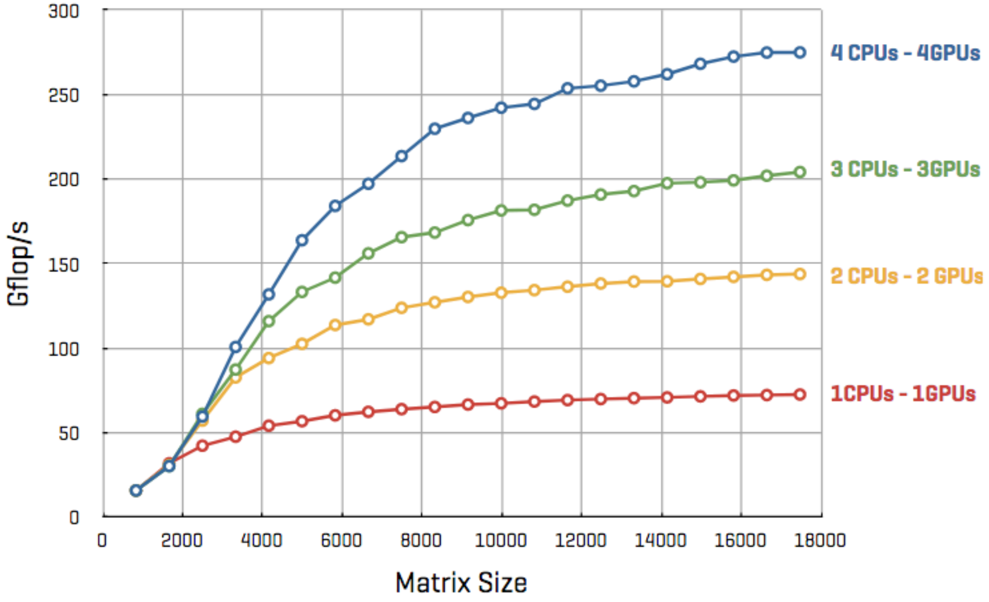
(a) Single Precision (b) Double Precision

- **Best GEMM performance** at the time
  (58% of peak in DP and 63% of peak in SP)
- **Added register blocking** to previous state-of-art
- Incorporated in CUBLAS
- More BLAS in MAGMA BLAS

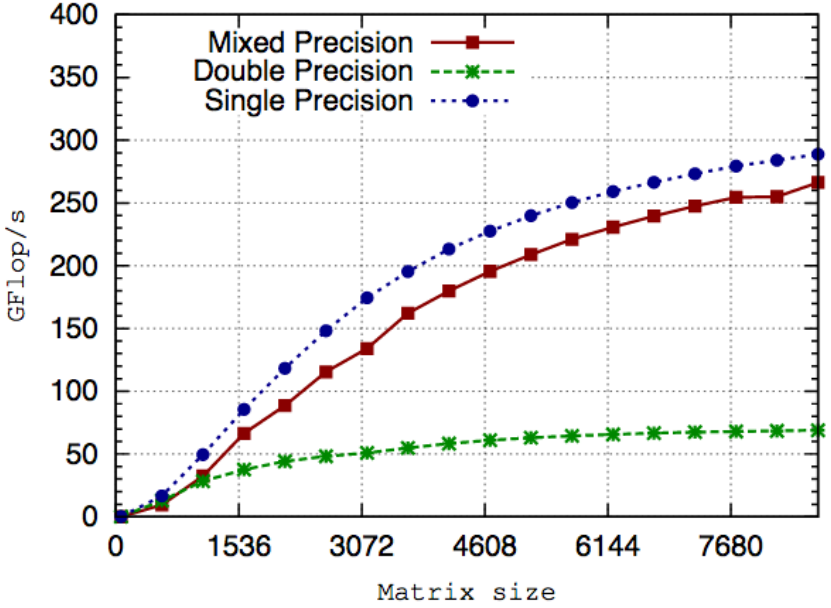
[1] R. Nath, S. Tomov, and J. Dongarra, "**An improved MAGMA GEMM for Fermi GPUs,**" International Journal of High Performance Computing Applications, vol. 24, no. 4, 2010.
[2] Nath, R., S. Tomov, and J. Dongarra, "**Blas for GPUs,**" Scientific Computing with Multicore and Accelerators, Boca Raton, Florida, CRC Press, 2010.



Scaling of hybrid Cholesky on multiple C1060 GPUs (double precision)



Mixed-precision LU solver on GTX280 GPU (double precision accuracy)



- **Best GEMM performance** at the time (58% of peak in DP and 63% of peak in SP)
- **Added register blocking** to previous state-of-art
- Incorporated in CUBLAS
- More BLAS in MAGMA BLAS
- **Multi-GPU**
- **Mixed-precision solvers**

[1] R. Nath, S. Tomov, and J. Dongarra, "**An improved MAGMA GEMM for Fermi GPUs,**" International Journal of High Performance Computing Applications, vol. 24, no. 4, 2010.
[2] Nath, R., S. Tomov, and J. Dongarra, "**Blas for GPUs,**" Scientific Computing with Multicore and Accelerators, Boca Raton, Florida, CRC Press, 2010.
[3] Tomov, S., R. Nath, H. Ltaeif, and J. Dongarra, "**Dense Linear Algebra Solvers for Multicore with GPU Accelerators**," IPDPS 2010 IEEE, Atlanta, GA, pp. 1-8, 2010.

07 08 09 10 11 12 13 14 15 16 17 18 19 20 2021

CUDA

Hybrid algorithms **MAGMA**

MAGMA 0.1 **Auto-tune** 2-sided fact.

GEMM BLAS **mixed pr.**

Tuning; Power **Scheduling** **MAGMA 1.0**

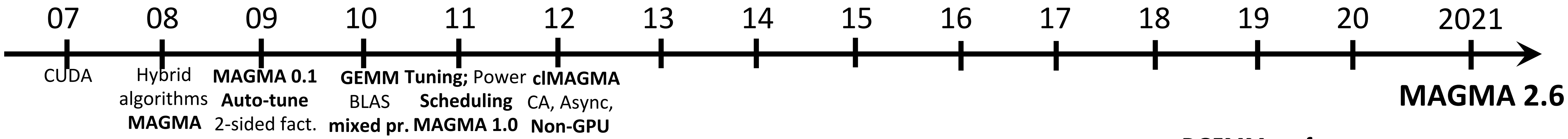
MAGMA 2.6

- **Auto-tuning GEMMs for Fermi** (J. Kurzak, S. Tomov, J. Dongarra, 2011)
- **Hybrid LAPACK algorithms** (M. Horton, S. Tomov, J. Dongarra, 2011)
- **Hybridization methodology** (Agullo, E., C. Augonnet, J. Dongarra, H. Ltaeif, R. Namyst, S. Thibault, and S. Tomov in 2011 GPU Computing Gems)

- **Power-aware computing on GPUs (**Kasichayanula, K., H. You, S. Moore, S. Tomov, H. Jagode, and M. Johnson, 2011)
- **Parallel Performance Measurement of Heterogeneous Parallel Systems with GPUs (**Malony, A. D., S. Biersdorff, S. Shende, H. Jagode, S. Tomov, G. Juckeland, R. Dietrich, D. Poole, and C. Lamb, ICPP'11)

- **Multi-core and multi-GPU algorithms** (F. Song, S. Tomov, J. Dongarra, 2011)
- **Performance portability with the DAGuE framework** (Bosilca, G., A. Bouteiller, T. Herault, P. Lemariner, N. Ohm Saengpatsa, S. Tomov, and J. Dongarra)

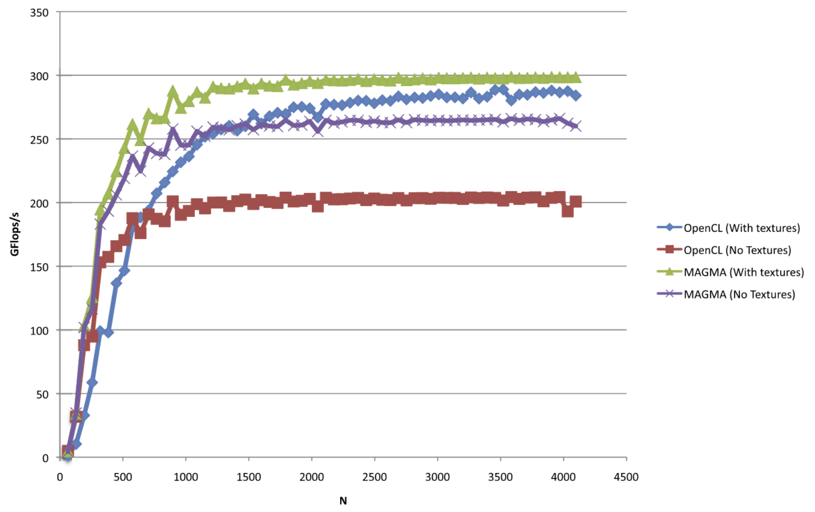
- Two major releases per year
  - **MAGMA 1.0** released in August, 2011



- **clMAGMA – OpenCL port of MAGMA used to add support to AMD GPUs**
- **clMAGMA 0.1 (April), clMAGMA 0.2 (May), clMAGMA 0.3 (June), clMAGMA 1.0 (December)**
  - LU, QR, and Cholesky factorization and solvers
  - Two-sided factorizations, eigen-solvers and SVD
  - Orthogonal transformation routines

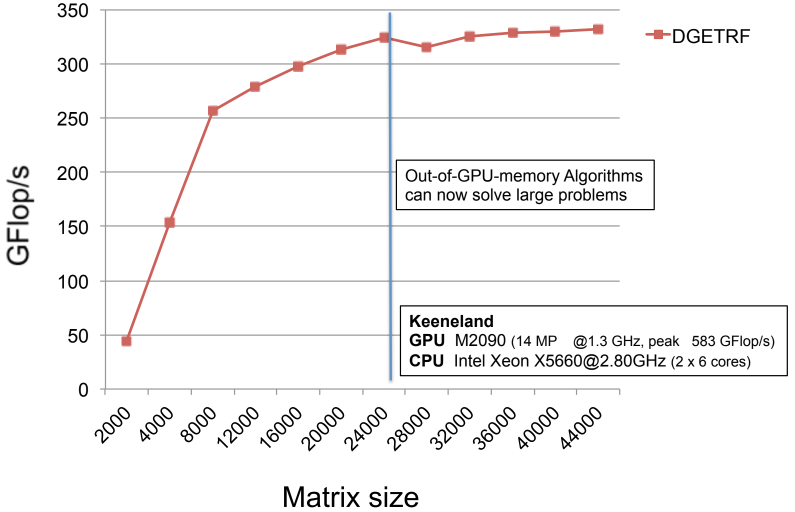
[1] Du, P., R. Weber, P. Luszczek, S. Tomov, G. D. Peterson, and J. Dongarra, **"From CUDA to OpenCL: Towards a Performance-portable Solution for Multi-platform GPU Programming,"** Parallel Computing, vol. 38, no. 8, pp. 391-407, August 2012.

DGEMM performance on Tesla C2050 under OpenCL and CUDA



- **Communication-avoiding algorithms** (Baboulin, M., S. Donfack, J. Dongarra, L. Grigori, A. Remi, and S. Tomov)
- **Asynchronous methods** (H. Anzt et al.)
- **Non-GPU resident algorithms** (I. Yamazaki et al.)

Solving large problems that do not fit in the GPU memory

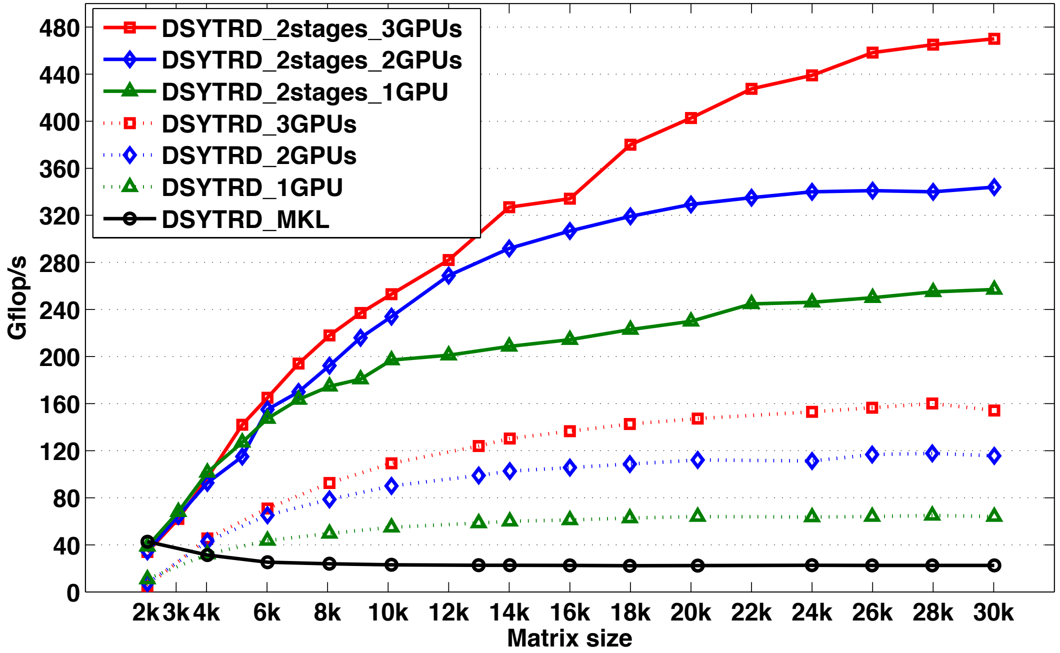


07 08 09 10 11 12 13 14 15 16 17 18 19 20 2021

CUDA | Hybrid algorithms **MAGMA** | **MAGMA 0.1** **Auto-tune** 2-sided fact. | **GEMM Tuning;** BLAS **mixed pr.** | Power **Scheduling** **MAGMA 1.0** | **clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**

MAGMA 2.6

Toward fast Eigensolver

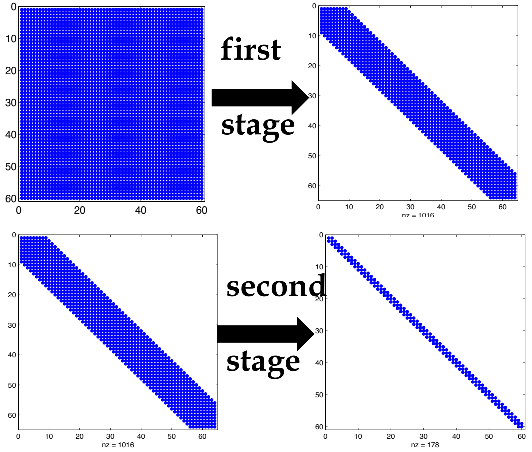


flops formula: **$n^3/3$*time**
Higher is faster

Keeneland system, using one node
3 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

✷ Characteristics

- **Stage 1:** BLAS-3, increasing computational intensity,
- **Stage 2:** BLAS-1.5, new cache friendly kernel,
- 4X/12X faster than standard approach,
- Bottelneck: if all Eigenvectors are required, it has 1 back transformation extra cost.



A. Haidar, S. Tomov, J. Dongarra, T. Schulthess, and R. Solca, ***"A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks"***, ICL Technical report, 03/2012 (in IJHPCA 2014)

07 08 09 10 11 12 13 14 15 16 17 18 19 20 2021

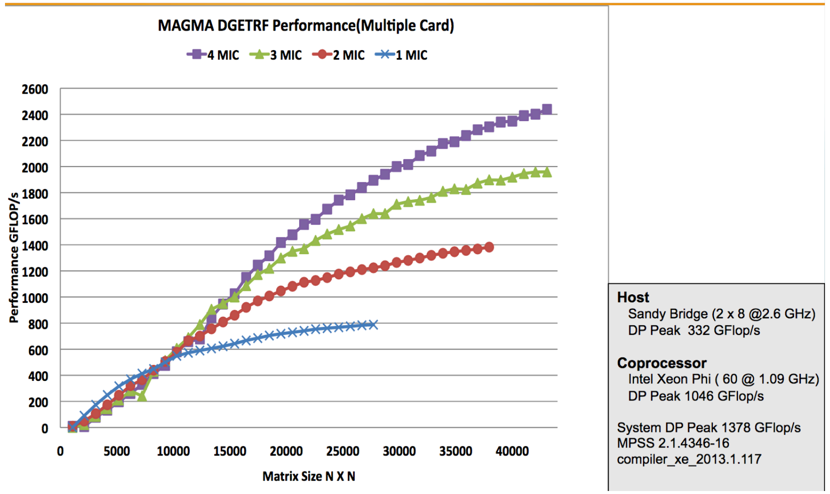
- 07: CUDA
- 08: Hybrid algorithms **MAGMA**
- 09: **MAGMA 0.1** **Auto-tune** 2-sided fact.
- 10: **GEMM Tuning;** BLAS **mixed pr.**
- 11: Power **Scheduling** **MAGMA 1.0**
- 12: **clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**
- 13: MAGMA **MIC**

MAGMA 2.6

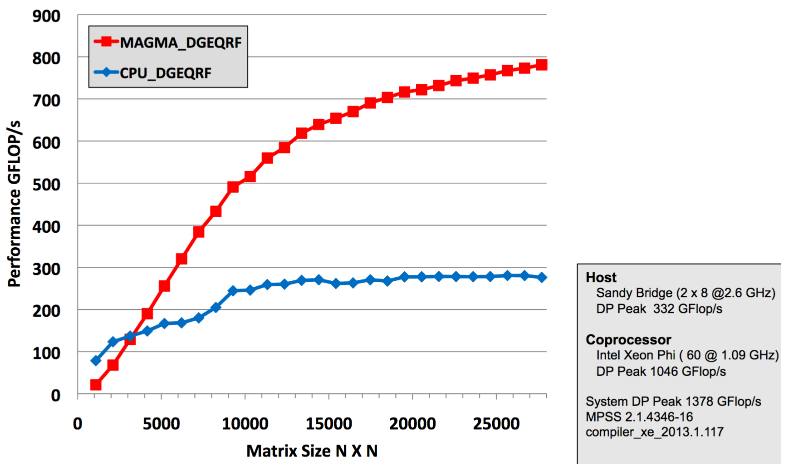
- MAGMA MIC – **MAGMA port providing support** for Intel Xeon Phi Coprocessors
- **MAGMA MIC 0.3 (November 2012), MAGMA MIC 1.0 (May 2013)**
  - LU, QR, and Cholesky factorization and solvers
  - Two-sided factorizations, eigen-solvers and SVD
  - Orthogonal transformation routines

[1] Dongarra, J., M. Gates, A. Haidar, Y. Jia, K. Kabir, P. Luszczek, and S. Tomov, "**Portable HPC Programming on Intel Many-Integrated-Core Hardware with MAGMA Port to Xeon Phi**," PPAM 2013, Warsaw, Poland, September 2013.

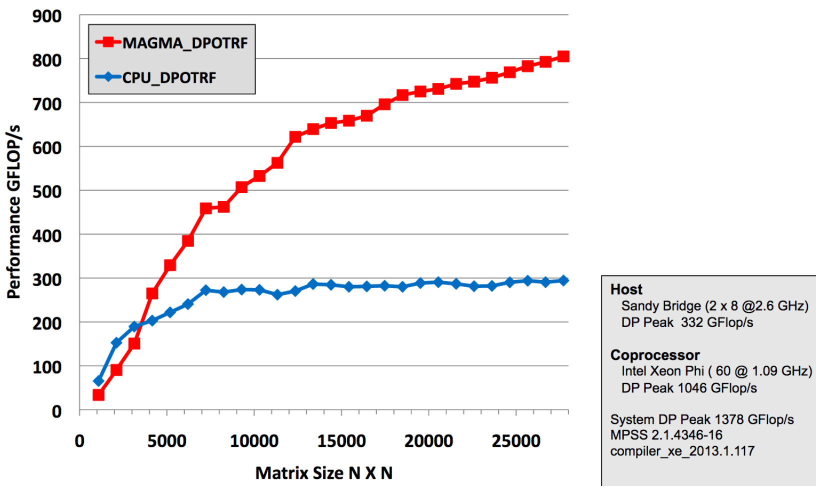
MAGMA MIC Scalability LU Factorization Performance in DP



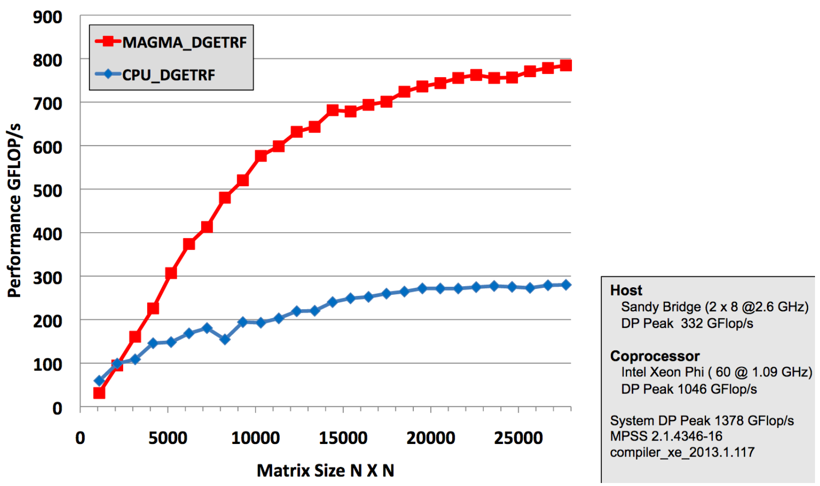
MAGMA MIC Performance (QR)

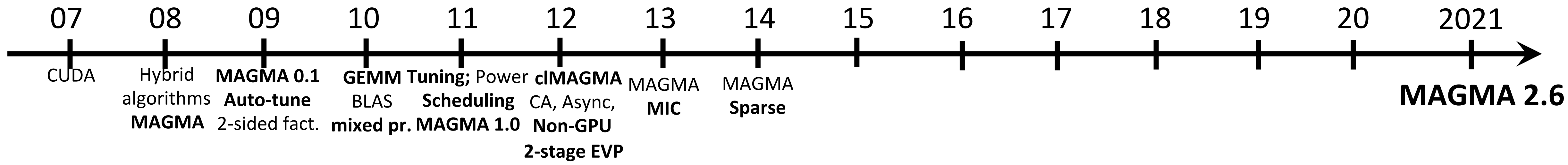


MAGMA MIC Performance (Cholesky)



MAGMA MIC Performance (LU)

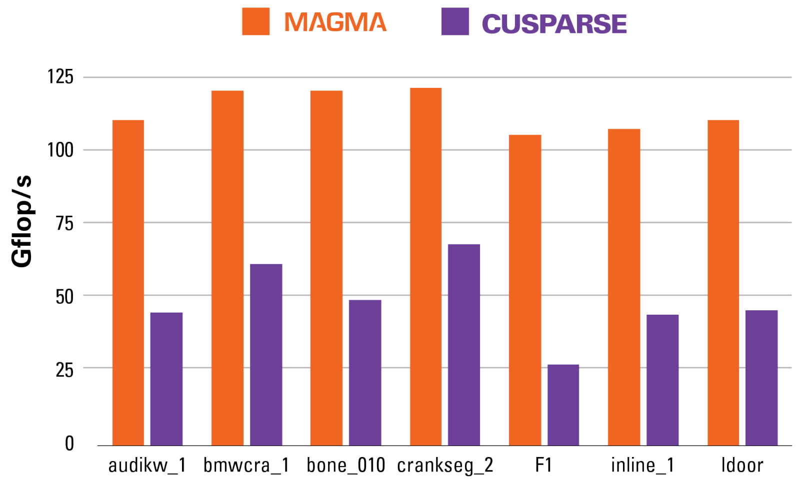




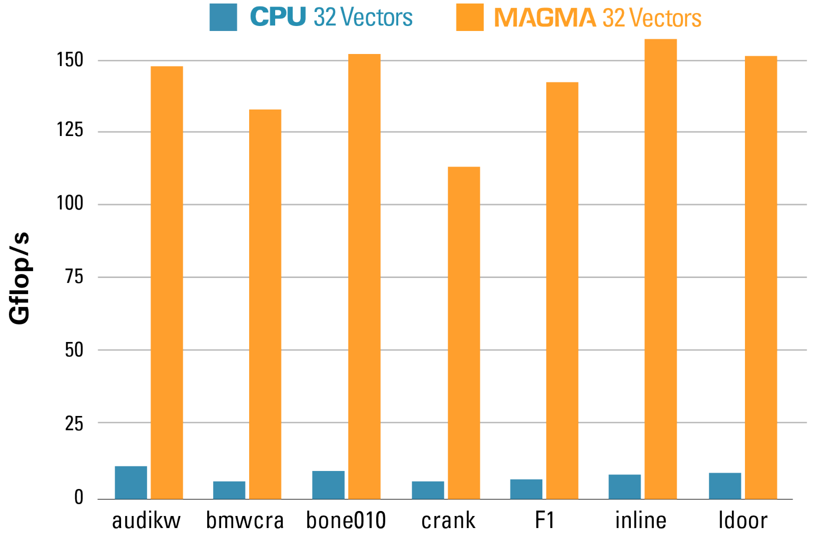
- MAGMA Sparse available since MAGMA 1.3 (November 2012)
- Extended continuously improved up to current MAGMA 1.5 release (September 2014)

- **Krylov Subspace Solvers (**H. Anzt, J. Dongarra, P. Luszczek, W. Sawyer, S. Tomov, I. Yamazaki)
- **CA Krylov methods (** I. Yamazaki et al.)
- **LOBPCG and Blocked SpMM** (H. Anzt et al.)
- **Mixed-precision orthogonalization and adaptive CA-GMRES** (I. Yamazaki et al.; best paper at VECPAR)
- **Self-adaptive multiprecision preconditioners** (H. Anzt et al.)
- **CA-GMRES with multiple GPUs** (I. Yamazaki et al.)
- **Sparse Matrix Vector Products for GPUs** (H. Anzt et al.)

Performance of SpMM with various matrices (x 32 vec.)



Overall speedup vs. LOBPCG from BLOPEX on CPUs



GPU K40 (all 16 cores) **CPU** 2 x 8-core Intel Sandy Bridge + GPU **BLOPEX LOBPCG:** uses CPU **MAGMA Sparse:** uses CPU

07 08 09 10 11 12 13 14 15 16 17 18 19 20 2021

07	08	09	10	11	12	13	14	15	2021
CUDA	Hybrid algorithms **MAGMA**	**MAGMA 0.1** **Auto-tune** 2-sided fact.	**GEMM Tuning;** BLAS **mixed pr.**	Power **Scheduling** **MAGMA 1.0**	**clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**	MAGMA **MIC**	MAGMA **Sparse**	MAGMA **Batched**	**MAGMA 2.6**

- MAGMA Batched available at least since MAGMA 1.5 (September 2014)
  - Continuously developed, extended, and improved
- **Batched BLAS**
- **Batched LAPACK**
- **Leading Batched BLAS and LAPACK standard development**
- **Batched LA is needed in many applications**
- **MAGMA provides most complete batched BLAS and LAPACK**
  - **Vendors have also started to provide some in their numerical libraries**

[1] Haidar, A., A. Abdelfattah, S. Tomov, and J. Dongarra, "**Batched Matrix Computations on Hardware Accelerators Based on GPUs**," SIAM LA15.

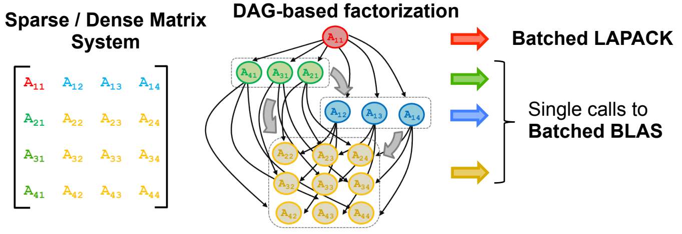
[2] Haidar, A., P. Luszczek, S. Tomov, and J. Dongarra, "**Batched Matrix Computations on Hardware Accelerators**," EuroMPI/Asia 2015.

[3] A Haidar, T Dong, P Luszczek, S Tomov, J Dongarra, **"Towards batched linear solvers on accelerated hardware platforms",** ACM SIGPLAN Notices 50 (8), 261-262, 2015.

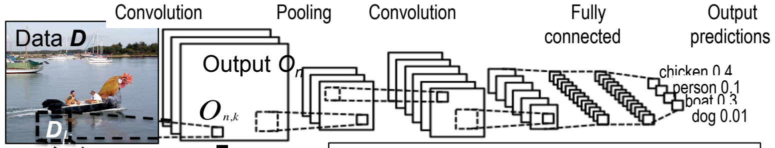
Data Analytics and associated with it Linear Algebra on small LA problems are needed in many applications:

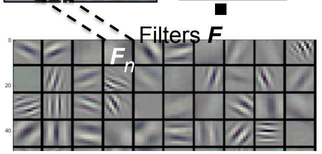
- Machine learning,
- Data mining,
- High-order FEM,
- Numerical LA,
- Graph analysis,
- Neuroscience,
- Astrophysics,
- Quantum chemistry,
- Multi-physics problems,
- Signal processing, etc.

Sparse/Dense solvers & preconditioners



Machine learning





Convolution of Filters F_i (feature detection) and input image D:

- For every filter F_n and every channel, the computation for every pixel value $O_{n,k}$ is a **tensor contraction**:

$$O_{n,k} = \sum_i D_{k,i} F_{n,i}$$

- Plenty of parallelism; **small operations** that must be batched
- With data "reshape" the computation can be transformed into a **batched GEMM** (for efficiency; among other approaches)

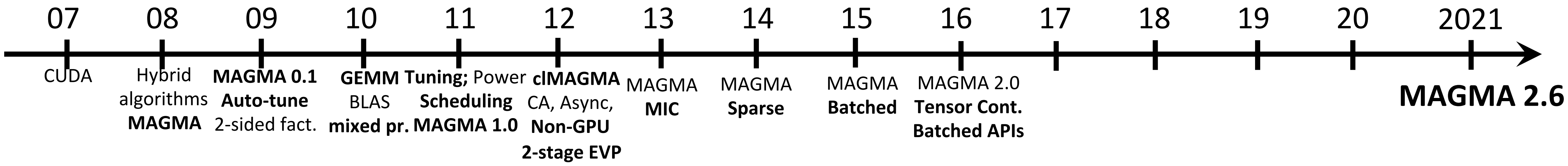
Applications using high-order FEM

- Matrix-free basis evaluation needs efficient tensor contractions,

$$C_{i1,i2,i3} = \sum_k A_{k,i1} B_{k,i2,i3}$$

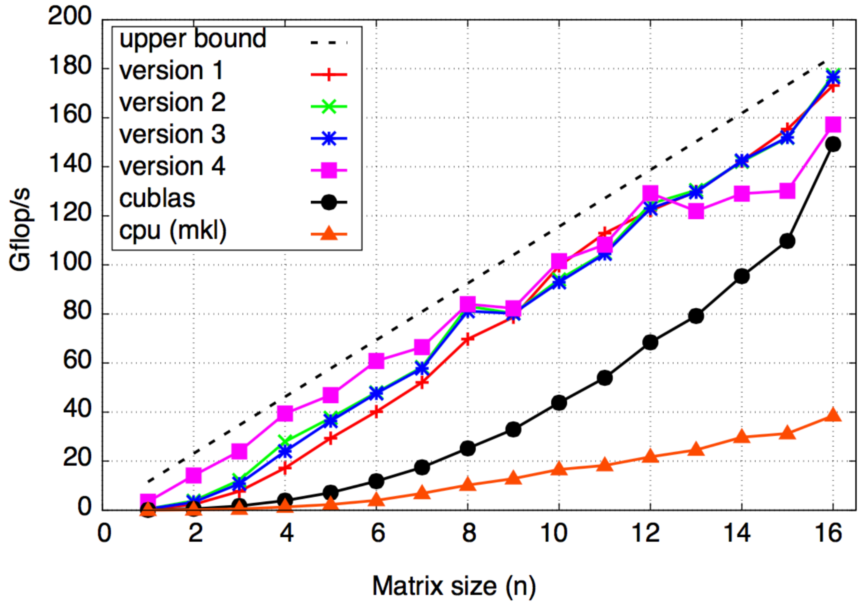
- **Within ECP CEED Project,** designed MAGMA batched methods to split the computation in many small high-intensity GEMMs, grouped together (batched) for efficient execution:

Batch_{ $C_{i3} = A^T B_{i3}$, for range of i3 }

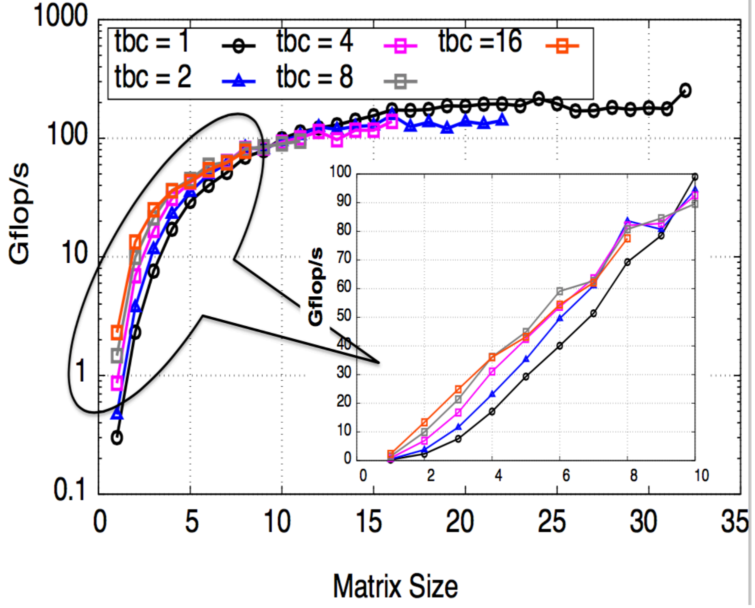


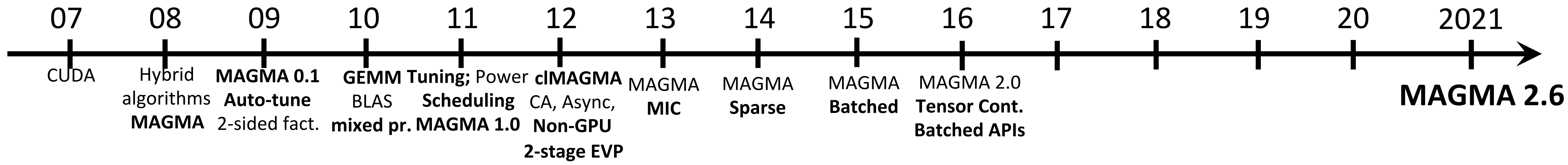
- **Performance, design, and autotuning of batched GEMM for GPUs** (A Abdelfattah et al.)
- **High-performance matrix-matrix multiplications of very small matrices** (I Masliah et al.)
- **High-performance tensor contractions for GPUs** (A Abdelfattah et al.)
- **Accelerating Tensor Contractions for High-Order FEM on CPUs, GPUs, and KNLs** (A. Haidar et al.)
- **Performance Tuning and Optimization Techniques of Fixed and Variable Size Batched Cholesky Factorization on GPUs** (A Abdelfattah et al.)
- **A proposed API for batched basic linear algebra subprograms** (J Dongarra et al.)

Performance comparison of tensor contraction versions using batched C = αAB + βC on 100,000 square matrices of size n on a K40c GPU and 16 cores of Intel Xeon E5-2670, 2.60 GHz CPUs.



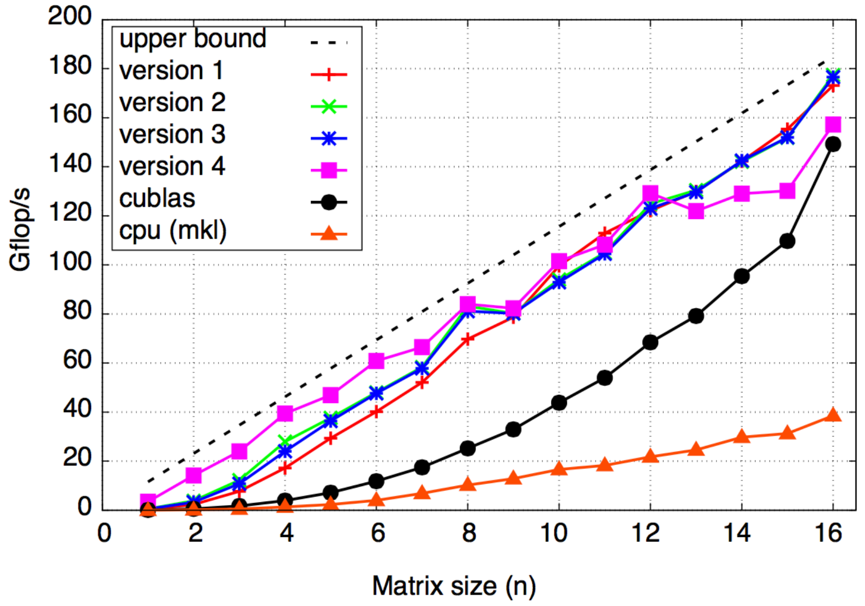
Effect of a Thread Block Concurrency (tbc) techniques where several DGEMMs are performed on one TB simultaneously



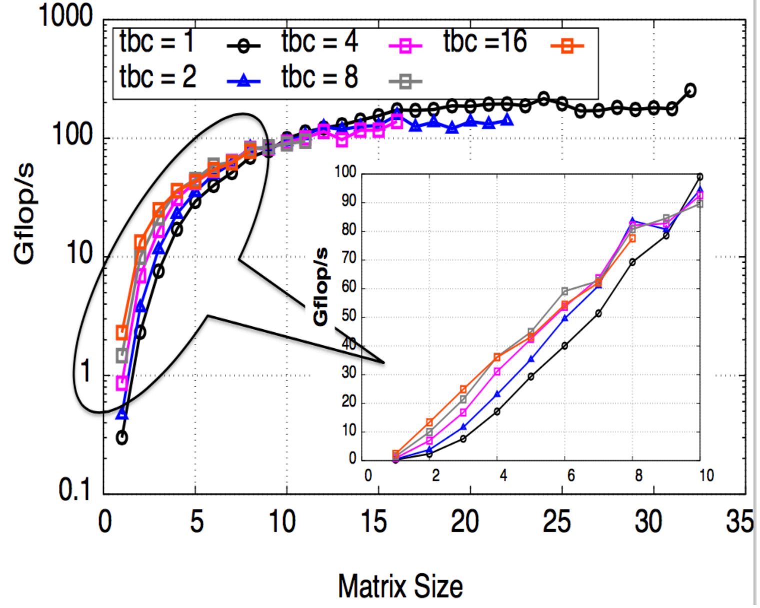


- **Performance, design, and autotuning of batched GEMM for GPUs** (A Abdelfattah et al.)
- **High-performance matrix-matrix multiplications of very small matrices** (I Masliah et al.)
- **High-performance tensor contractions for GPUs** (A Abdelfattah et al.)
- **Accelerating Tensor Contractions for High-Order FEM on CPUs, GPUs, and KNLs** (A. Haidar et al.)
- **Performance Tuning and Optimization Techniques of Fixed and Variable Size Batched Cholesky Factorization on GPUs** (A Abdelfattah et al.)
- **A proposed API for batched basic linear algebra subprograms** (J Dongarra et al.)

Performance comparison of tensor contraction versions using batched C = αAB + βC on 100,000 square matrices of size n on a K40c GPU and 16 cores of Intel Xeon E5-2670, 2.60 GHz CPUs.



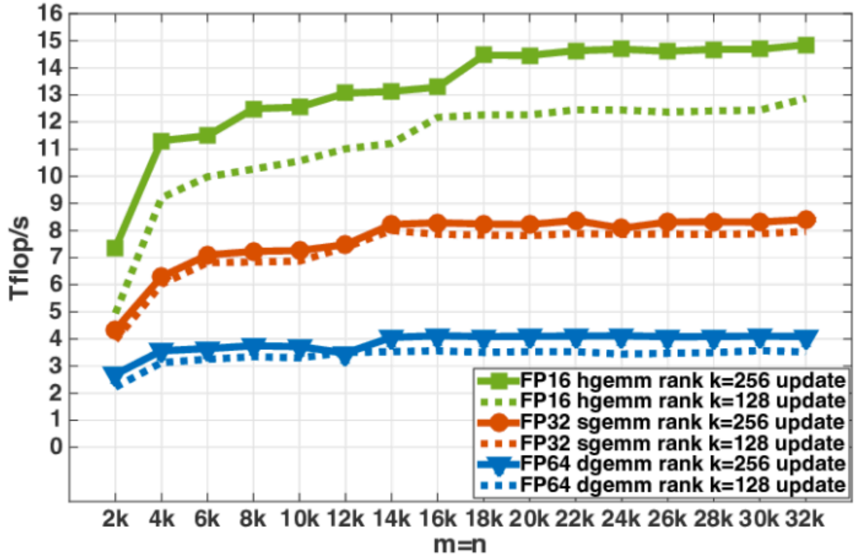
Effect of a Thread Block Concurrency (tbc) techniques where several DGEMMs are performed on one TB simultaneously



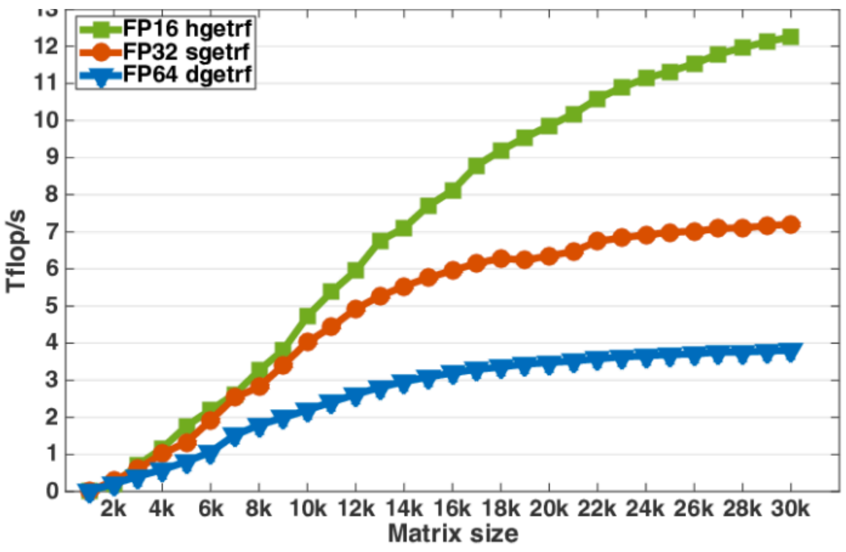
07	08	09	10	11	12	13	14	15	16	17	18	19	20	2021
CUDA	Hybrid algorithms **MAGMA**	**MAGMA 0.1** **Auto-tune** 2-sided fact.	**GEMM Tuning;** BLAS **mixed pr.**	Power **Scheduling** **MAGMA 1.0**	**clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**	MAGMA **MIC**	MAGMA **Sparse**	MAGMA **Batched**	MAGMA 2.0 **Tensor Cont.** **Batched APIs**	DNN **LDLT** **FP16 TC**				**MAGMA 2.6**

- MagmaDNN high-performance data analytics for manycore GPUs and CPUs
- ASGD training
- Sampling algorithms to update truncated SVD
- Out of memory algorithms (SVD and symmetric and indefinite problems)
- Symmetric Indefinite Solvers
- Convex optimizations
- Tensor contractions and other batched routines
- Mixed-precision iterative refinement using GPU Tensor Cores and FP16
    - "Investigating half precision arithmetic to accelerate dense linear system solvers", A Haidar, P Wu, S Tomov, J Dongarra, SC'17 ScalA Workshop, 2017.

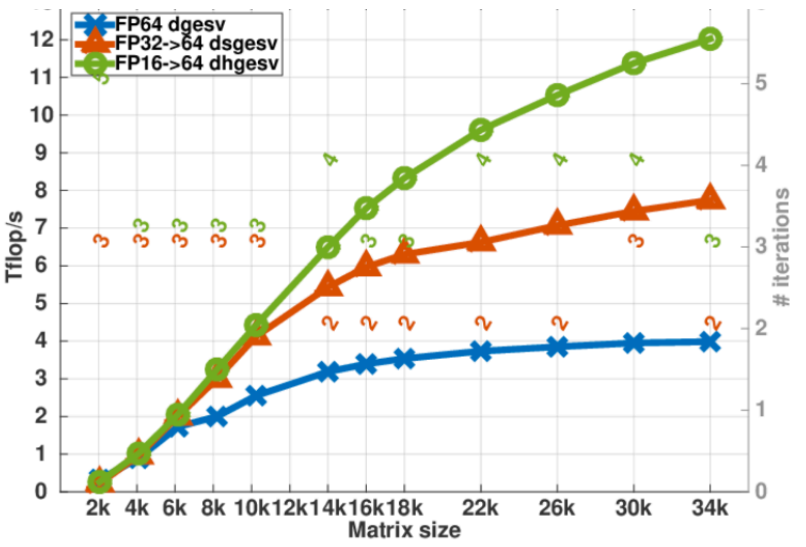
Rank-k (xGEMM) performance on V100



xGETRF performance on V100



Performance of Iter. Refinment Solver



07 — CUDA

08 — Hybrid algorithms **MAGMA**

09 — **MAGMA 0.1** **Auto-tune** 2-sided fact.

10 — **GEMM Tuning;** BLAS **mixed pr.**

11 — Power **Scheduling** **MAGMA 1.0**

12 — **clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**

13 — MAGMA **MIC**

14 — MAGMA **Sparse**

15 — MAGMA **Batched**

16 — MAGMA 2.0 **Tensor Cont.** **Batched APIs**

17 — DNN **LDLT** **FP16 TC**

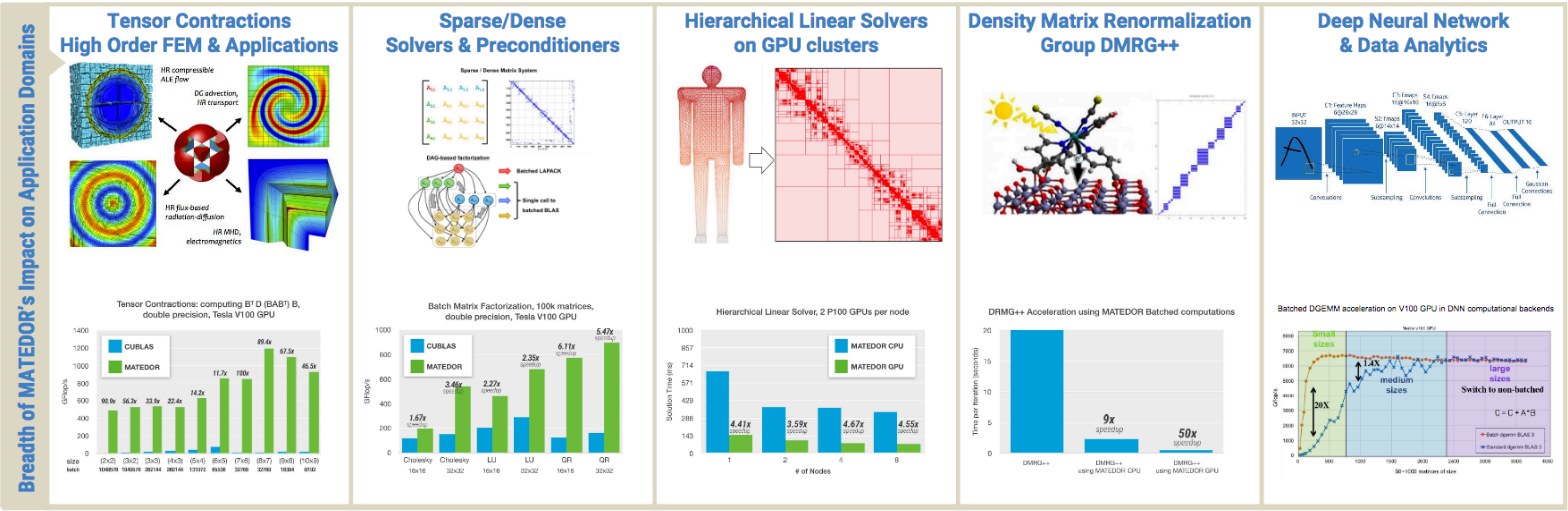
18 — MATEDOR **FP16 TC** **SVD**

19

20

2021 — **MAGMA 2.6**

- **MATEDOR – Matrix, Tensor, and Deep-learning Optimized Routines**
- **Tensor Contractions using Optimized Batch GEMM Routines**
- **Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers,** A Haidar, S Tomov, J Dongarra, NJ Higham, SC18.
- **Harnessing GPU's Tensor Cores Fast FP16 Arithmetic to Speedup Mixed-Precision Iterative Refinement Solvers and Achieve 74 Gflops/Watt on Nvidia V100** (Haidar et al., GTC18)
- **The design of fast and energy-efficient linear solvers: On the potential of half-precision arithmetic and iterative refinement techniques** (A Haidar et al., ICCS18)
- **Accelerating the SVD two stage bidiagonal reduction and divide and conquer using GPUs** (M. Gates et al., J. Parallel Computing 2018)



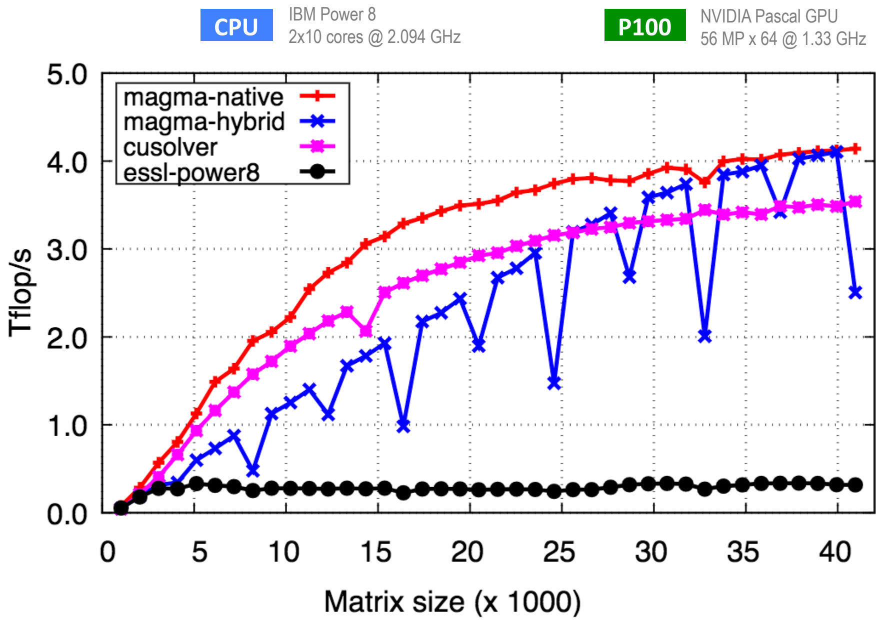
07	08	09	10	11	12	13	14	15	16	17	18	19	20	2021
CUDA	Hybrid algorithms **MAGMA**	**MAGMA 0.1** **Auto-tune** 2-sided fact.	**GEMM Tuning;** BLAS **mixed pr.**	Power **Scheduling** **MAGMA 1.0**	**clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**	MAGMA **MIC**	MAGMA **Sparse**	MAGMA **Batched**	MAGMA 2.0 **Tensor Cont.** **Batched APIs**	DNN **LDLT** **FP16 TC**	MATEDOR **FP16 TC** **SVD**	**FP16 TC** **DNN** **GPU-only**		**MAGMA 2.6**

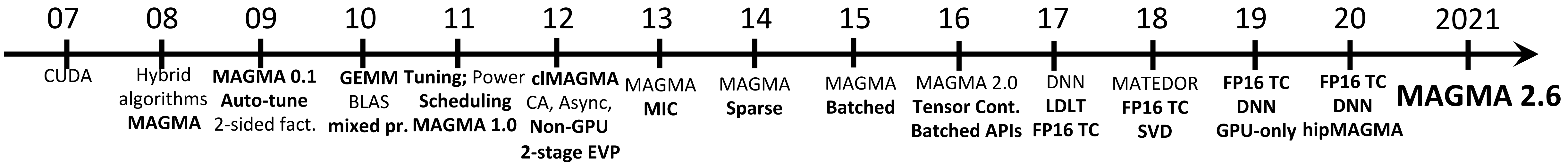
- **Towards Half-Precision Computation for Complex Matrices: A Case Study for Mixed Precision Solvers on GPUs** (A. Abdelfattah et al.)
- **Fast batched matrix multiplication for small sizes using half-precision arithmetic on GPUs** (A. Abdelfattah et al.)

- **MagmaDNN: towards high-performance data analytics and machine learning for data-driven scientific computing** (D Nichols et al., ISC19)
- **MagmaDNN 1.1 release**
  - Bug fixes and performance improvements;
  - Distributed training;
  - Hyperparameter optimization framework improvements;
  - Benchmarks using MagmaDNN;
  - Performance comparisons, accuracy validations, etc. (w\ TensorFlow, Theano, and PyTorch).

- **MAGMA 2.5 release**
  - Mixed-precision using Nvidia Tensor Cores
  - LU and Cholesky native (GPU-only)

MAGMA LU factorization in DP arithmetic



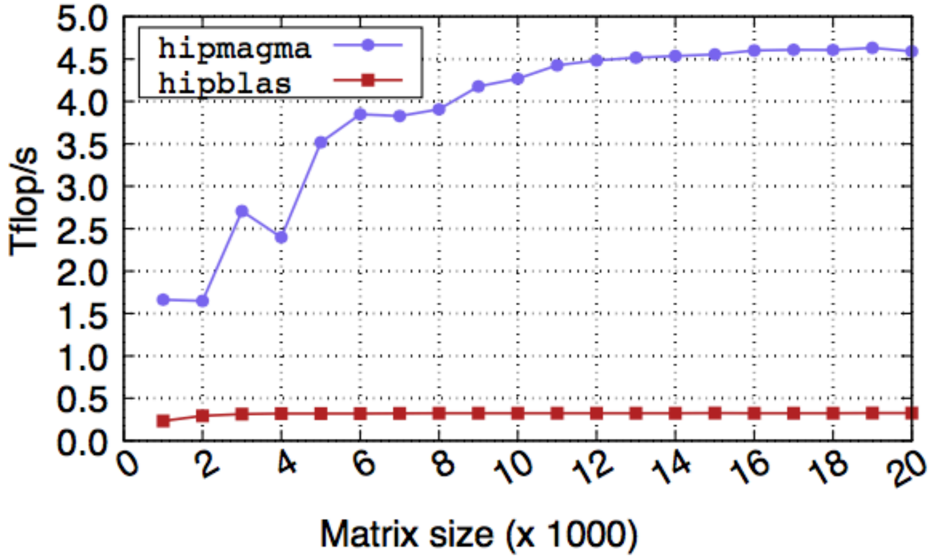


- **Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems** (A Haidar et al.)
- **Matrix multiplication on batches of small matrices in half and half-complex precisions** (A. Abdelfattah et al.)
- **Investigating the benefit of fp16-enabled mixed-precision solvers for symmetric positive definite matrices using gpus** (A. Abdelfattah et al.)

- **Integrating Deep Learning in Domain Sciences at Exascale** (R Archibald et al.; SMC20)
- **MagmaDNN 1.2 release**
  - oneDNN (MKL DNN) support including fully connected, convolutional, and pooling layers;
  - CMake build system;
  - Added NN examples including CNN, ResNet, AlexNet, LeNet, MNIST and CIFAR interactive, and VGG16;
  - Added examples for Tensor Math; C++ style formatter;
  - Modularized distributed optimizer;
  - CIFAR10, CIFAR100, and MNIST data loaders;
  - CUDA streams added;
  - Model summary prinout;
  - Spack package manager installation support.

- **Design, Optimization, and Benchmarking of Dense Linear Algebra Algorithms on AMD GPUs** (C Brown, A Abdelfattah, S Tomov, J Dongarra, HPEC'20)
- **hipMAGMA 1.0** (March 2020) and **hipMAGMA 2.0** (July 2020)
  - MAGMA BLAS and LAPACK
  - Batched BLAS and Batched LAPACK are ported to HIP
  - Working with AMD

DSYRK on the Mi50 GPU with ROCm 3.5



07	CUDA
08	Hybrid algorithms **MAGMA**
09	**MAGMA 0.1** **Auto-tune** 2-sided fact.
10	**GEMM Tuning;** BLAS **mixed pr.**
11	Power **Scheduling** **MAGMA 1.0**
12	**clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**
13	MAGMA **MIC**
14	MAGMA **Sparse**
15	MAGMA **Batched**
16	MAGMA 2.0 **Tensor Cont.** **Batched APIs**
17	DNN **LDLT** **FP16 TC**
18	MATEDOR **FP16 TC** **SVD**
19	**FP16 TC** **DNN** **GPU-only**
20	**FP16 TC** **DNN** **hipMAGMA**
2021	**MAGMA 2.6**

MAGMA Today

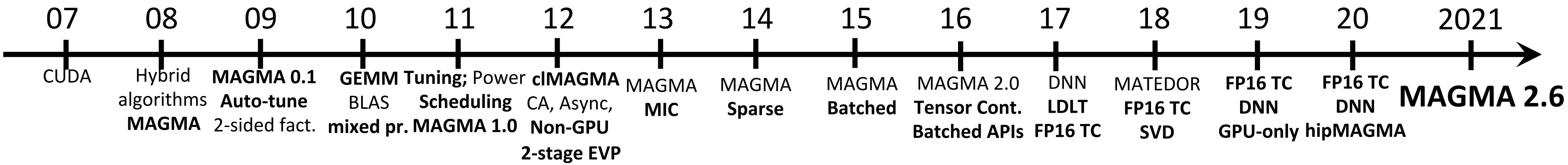
- **MAGMA has grown significantly in functionalities over the years**
- **Provided are around 3,000 routines (per architecture)**
- **Ports become challenge due to volume**
- **Still, there is functional and performance portability**
  - **Due to use of standards, e.g., BLAS for performance portability**
  - **Abstractions that keep base of code unchanged**
  - **Auto-source translation tools, when needed**
    - **The challenge are BLAS and low level kernels that may benefit architecture-specific tuning**
  - **Must add some auto-tuning framework to help performance portability**

for architectures in
{ CPUs + Nvidia GPUs (CUDA),
CPUs + AMD GPUs (HIP & OpenCL),
CPUs + Intel Xeon Phis,
manycore (native: GPU or KNL/CPU),
embedded systems, combinations, and
software stack, e.g., since CUDA x}

for precisions in
{ s, d, c, z,
half-precision (FP16),
mixed, … }

for interfaces
{ heterogeneous CPU/GPU, native, … }

- **LAPACK**
- **BLAS**
- **Batched LAPACK**
- **Batched BLAS**
- **Sparse**
- **Tensors**
- **MAGMA-DNN**
- **Templates**
- **…**



MAGMA 2.6 Release (July 1, 2021)

- **Added HIP support for AMD GPUs (former hipMAGMA) as part of MAGMA**
- **The CUDA and HIP functionalities are in sync**
- **MAGMA Sparse added for AMD GPUs with this release**
- **Support for makefile, cmake, or spack installation**
- **Added inertia computational routines for GPUs**
- **Performance improvements for AMD GPUs**
- **Performance improvement for magma_Xgesv_batched for small sizes**
- **Added Bunch-Kaufman GPU-only sover using BLAS calls (magma_zhetrs_gpu)**
- **Added include/magma_config.h file storing the configuration for a particular magma installation (CUDA vs. HIP, etc.)**
- **Added expert interfaces for magma_Xgetrf_gpu and magma_Xpotrf_gpu. These interfaces allow the user to specify the factorization mode; hybrid (CPU+GPU) vs. native (GPU only), as well as the blocking size (nb)**
- **Added tuning for small size LU, QR, and Cholesky factorizations**

07 CUDA

08 Hybrid algorithms **MAGMA**

09 **MAGMA 0.1** **Auto-tune** 2-sided fact.

10 **GEMM Tuning;** BLAS **mixed pr.**

11 Power **Scheduling** **MAGMA 1.0**

12 **clMAGMA** CA, Async, **Non-GPU** **2-stage EVP**

13 MAGMA **MIC**

14 MAGMA **Sparse**

15 MAGMA **Batched**

16 MAGMA 2.0 **Tensor Cont.** **Batched APIs**

17 DNN **LDLT** **FP16 TC**

18 MATEDOR **FP16 TC** **SVD**

19 **FP16 TC** **DNN** **GPU-only**

20 **FP16 TC** **DNN** **hipMAGMA**

2021 **MAGMA 2.6**

Next

- Mixed-precision solvers, e.g., to harness TC hardware for fast FP16
- Batched LA, e.g., SVD, solvers
- Performance optimizations
  - Tuning for new architectures
  - Kernels/BLAS
  - More GPU-only factorizations
  - Optimize multi-GPU algorithms
- User requests, e.g., inertia computations, LDLT factorizations
- Tensor contractions, e.g., for high-order FEM solvers
- Data analytics & AI
  - Kernels (GEMMs, SVD, batched routines, convolutions, FFT, FFT variants, etc.)
  - DNN framework
  - Users calling MAGMA through PyTorch, etc.
- Porting to other programming models
  - Intel GPUs (using MKL oneAPI & DPC++)