MAGMA Batched Computations: Approaches and Applications

Stan Tomov
Innovative Computing Laboratory
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville

Workshop on Batched, Reproducible, and Reduced Precision BLAS
Georgia Tech Computational Science and Engineering
Atlanta, GA
February 23—25, 2017

ICL UT INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Outline

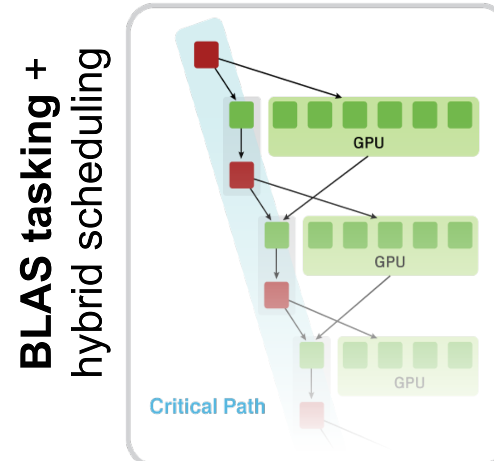
- **Motivation**
- **MAGMA Batched computations**
  - **Coverage**
  - **Interfaces**
  - **Applications**
  - **Design and optimizations for batched computations**
  - **Performance results**
- **Conclusions and future directions**

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Key Features of MAGMA 2.2

TASK-BASED ALGORITHMS

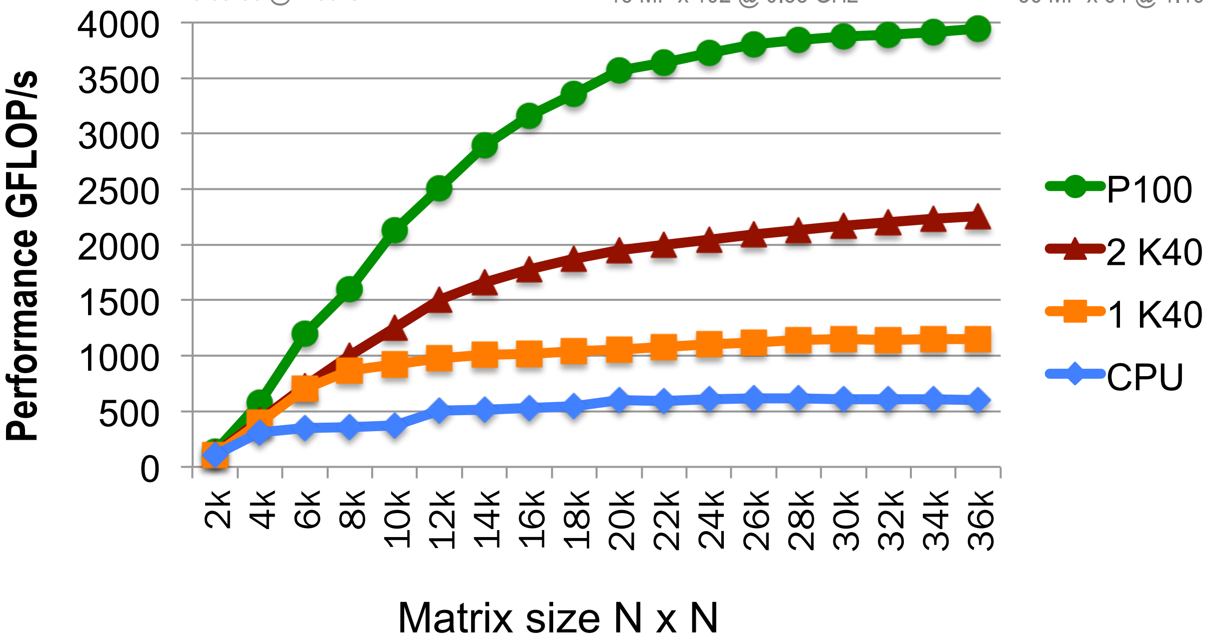
MAGMA uses task-based algorithms where the computation is split into tasks of varying granularity and their execution scheduled over the hardware components. Scheduling can be static or dynamic. In either case, small non-parallelizable tasks, often on the critical path, are scheduled on the CPU, and larger more parallelizable ones, often Level 3 BLAS, are scheduled on the GPUs.

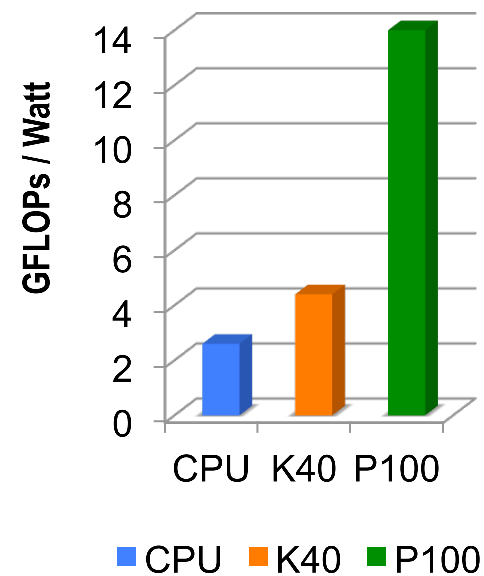


PERFORMANCE & ENERGY EFFICIENCY

MAGMA LU factorization in double precision arithmetic

CPU Intel Xeon E5-2650 v3 (Haswell) 2x10 cores @ 2.30 GHz
K40 NVIDIA K40 GPU 15 MP x 192 @ 0.88 GHz
P100 NVIDIA Pascal GPU 56 MP x 64 @ 1.19 GHz





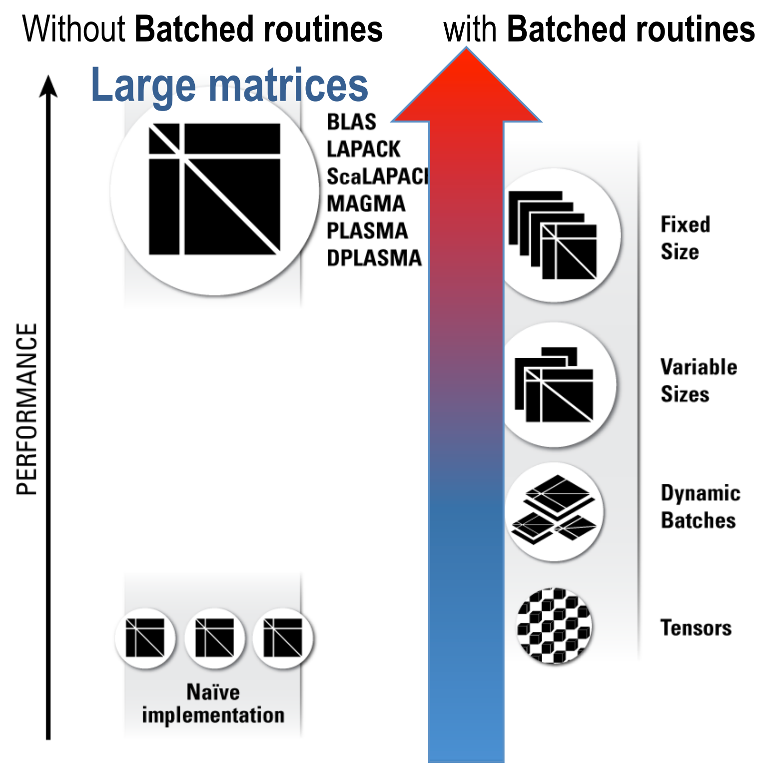
ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Linear Algebra on Small Matrices

Linear Algebra on small 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.



DLA 2016 Survey

- Dominant matrices to solve

O(10)	18%
O(100)	37%
O(1000)	61%
O(1000)-by-O(10)	28%

- One or many at a time

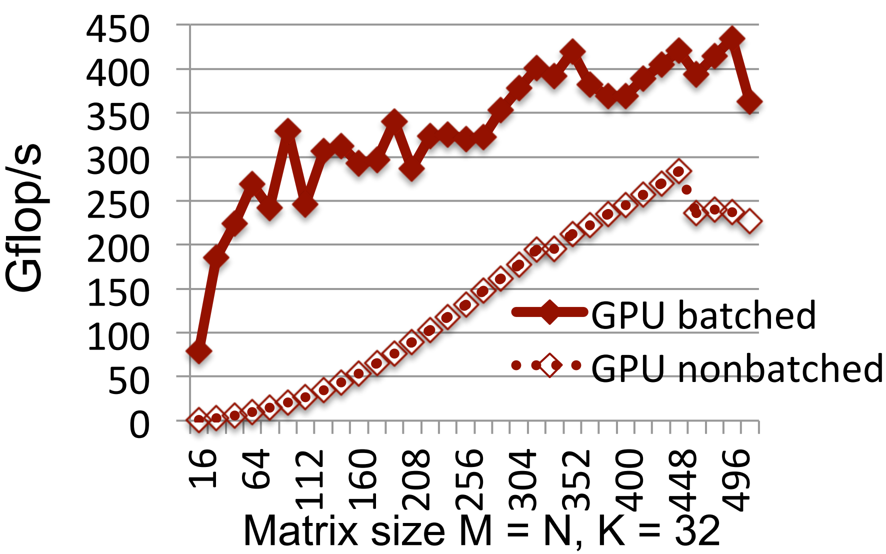
one	62%
many	38%

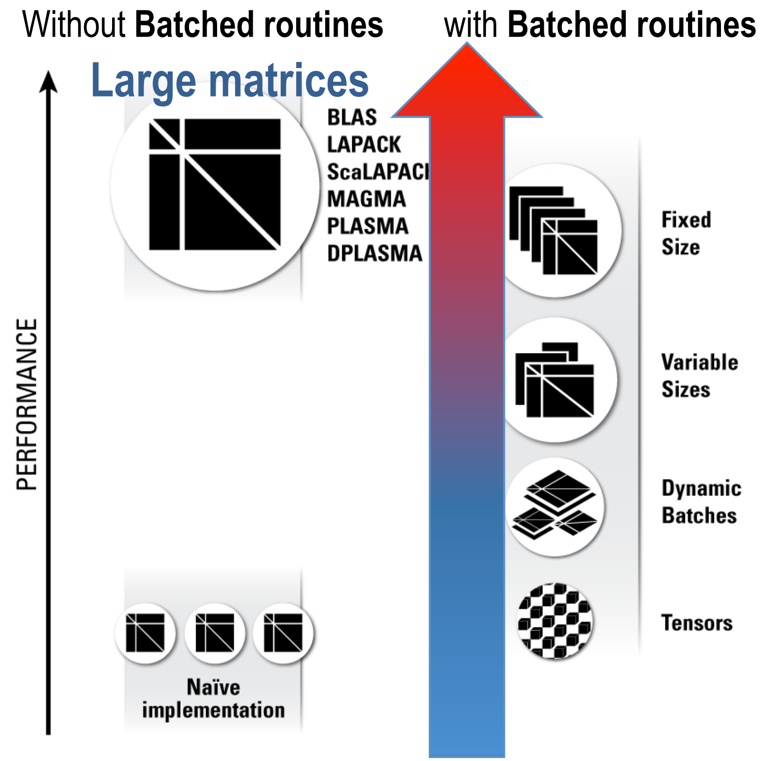
Linear Algebra on Small Matrices

Linear Algebra on small 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.

DGEMM (NN), batch_count = 500, 1 Tesla K40c GPU





ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Implementation on current hardware is becoming challenging

Memory hierarchies	Haswell E5-2650 v3	KNL 7250 DDR5|MCDRAM	ARM	K40c	P100
	10 cores	68 cores	4 cores	15 SM x 192 cores	56 SM x 64 cores
Registers	16/core AVX2	32/core AVX-512	32/core	256 KB/SM	256 KB/SM
L1 cache & GPU shared memory	32 KB/core	32 KB/core	32 KB/core	64 KB/SM	64 KB/SM
L2 cache	256 KB/core	1024 KB/2cores	2 MB	1.5 MB	4 MB
L3 cache	25 MB	0...16 GB	N/A	N/A	N/A
main memory	64 GB	384 |16 GB	4 GB	12 GB	16 GB
Main memory bandwidth	68 GB/s	115 | 421 GB/s	26 GB/s	288 GB/s	720 GB/s
PCI express gen3 x16	16 GB/s	16 GB/s	16 GB/s	16 GB/s	16 GB/s
Interconnect Cray Gemini	6 GB/s	6 GB/s	6 GB/s	6 GB/s	6 GB/s

Memory hierarchies for different type of architectures

Workshop on Batched, Reproducible, And Reduced Precision BLAS
Innovative Computing Laboratory
University of Tennessee
May 18th – 19th, 2016

http://bit.ly/Batch-BLAS-2016

Draft Reports
Batched BLAS Draft Reports:
https://www.dropbox.com/s/olocmipyxfvcaui/batched_api_03_30_2016.pdf?dl=0

Batched BLAS Poster:
https://www.dropbox.com/s/ddkym76fapddf5c/Batched%20BLAS%20Poster%2012.pdf?dl=0

Batched BLAS Slides:
https://www.dropbox.com/s/kz4fhcipz3e56ju/BatchedBLAS-1.pptx?dl=0

Webpage on ReproBLAS:
http://bebop.cs.berkeley.edu/reproblas/

Efficient Reproducible Floating Point Summation and BLAS:
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-229.pdf

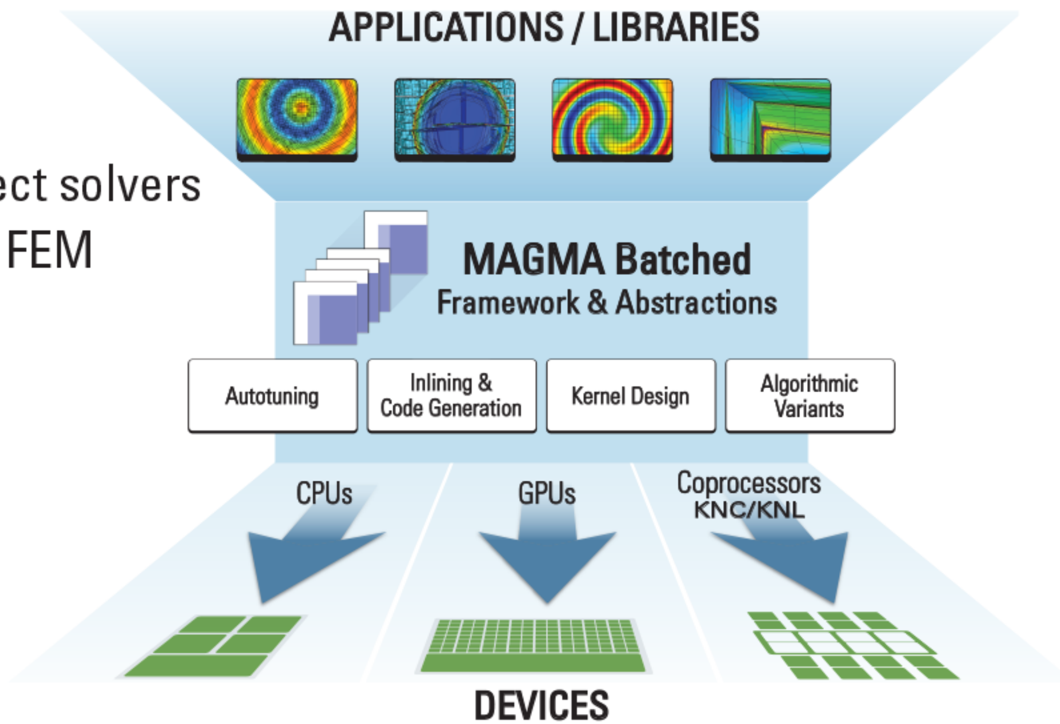
Batched routines released in MAGMA

MAGMA BATCHED

BATCHED FACTORIZATION OF A SET OF SMALL MATRICES IN PARALLEL

Numerous applications require factorization of many small matrices

- Deep learning
- Structural mechanics
- Astrophysics
- Sparse direct solvers
- High-order FEM simulations



ROUTINES

LU, QR, and Cholesky	✓
Solvers and matrix inversion	✓
All BLAS 3 (fixed + variable)	✓
SYMV, GEMV (fixed + variable)	✓

API for Batched BLAS in MAGMA

Batch of fixed-size problems:

```
extern "C" void
magmablas_dgemm_batched( magma_trans_t transA, magma_trans_t transB,
                         magma_int_t m, magma_int_t n, magma_int_t k,
                         double alpha,
                         double const * const * dA_array, magma_int_t ldda,
                         double const * const * dB_array, magma_int_t lddb,
                         double beta,
                         double **dC_array, magma_int_t lddc,
                         magma_int_t batchCount, magma_queue_t queue )
```

Batch of variable-size problems:

```
extern "C" void
magmablas_dgemm_vbatched(
    magma_trans_t transA, magma_trans_t transB,
    magma_int_t* m, magma_int_t* n, magma_int_t* k,
    double alpha,
    double const * const * dA_array, magma_int_t* ldda,
    double const * const * dB_array, magma_int_t* lddb,
    double beta,
    double **dC_array, magma_int_t* lddc,
    magma_int_t batchCount, magma_queue_t queue )
```

API for Batched LAPACK in MAGMA

Batch of fixed-size problems:

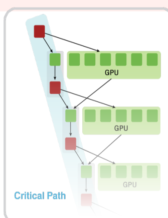
```
extern "C" magma_int_t
magma_zpotrf_batched(
    magma_uplo_t uplo, magma_int_t n,
    magmaDoubleComplex **dA_array, magma_int_t ldda,
    magma_int_t *info_array,  magma_int_t batchCount,
    magma_queue_t queue)
```

Batch of variable-size problems:

```
extern "C" magma_int_t
magma_zpotrf_vbatched(
    magma_uplo_t uplo, magma_int_t *n,
    magmaDoubleComplex **dA_array, magma_int_t *ldda,
    magma_int_t *info_array,  magma_int_t batchCount,
    magma_queue_t queue)
```

Batched BLAS Usage

Software/Algorithms follow hardware evolution in time	
LINPACK (70's) (Vector operations)	Level 1 **BLAS**
LAPACK (80's) (Blocking, cache friendly)	Level 3 **BLAS**
ScaLAPACK (90's) (Distributed Memory)	**PBLAS**
PLASMA (00's) New Algorithms (many-core friendly)	**BLAS** on tiles + DAG scheduling
MAGMA Hybrid Algorithms (heterogeneity friendly)	**BLAS** tasking + (CPU / GPU / Xeon Phi) hybrid scheduling



Algorithms expressed in terms of various BLAS calls

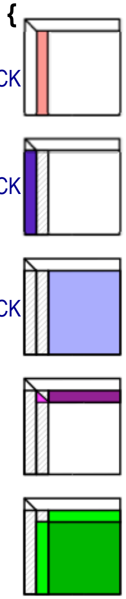
```
for(int i = 0; …) {
   // factor a panel
   // sequential LAPACK
   DGETRF2(…);

   // backward swap
   // sequential LAPACK
   DLASWP( … );

   // forward swap
   // sequential LAPACK
   DLASWP(…);

   // triangular solve
   // parallel BLAS
   DTRSM( … );

   // matrix multiply
   // Parallel BLAS
   DGEMM( … );
}
```



MAGMA BATCHED

- Batched routines can be developed efficiently using Batched BLAS
- Use and calling sequence of Batched BLAS is similar to BLAS

Applications – Tensor contractions

Numerous important applications:

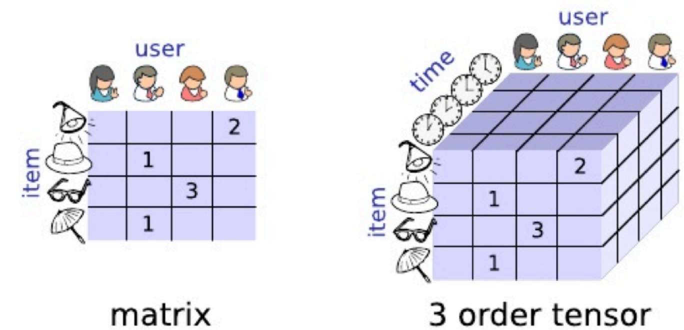
- **High-order FEM simulations (with LLNL)**
- Signal Processing, Numerical Linear Algebra, Numerical Analysis, Data Mining, Deep Learning, Graph Analysis, Neuroscience, and more

can be expressed through tensors.

The goal is to design a:

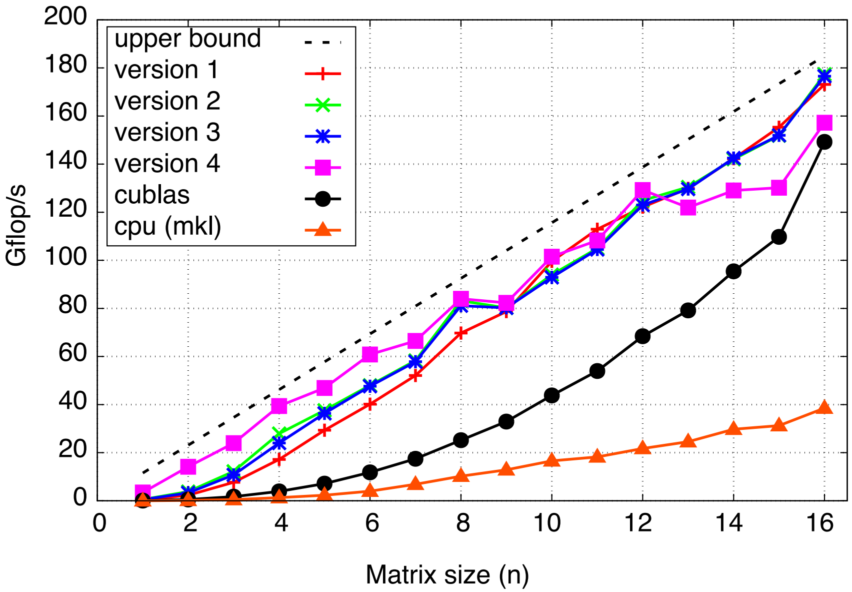
- High-performance package for Tensor algebra;
- Built-in architecture-awareness (GPU, Xeon Phi, multicore);
- User-friendly interface.

Example: **Relational Data**



Item	⇔ scalar	tensors	(0)
Items	⇔ vector		(1)
Relations of pairs	⇔ matrix		(2)
Relations of 3-tuple	⇔ 3-D array		(3)
…			
Relations of N-tuples	⇔ N-D array		(N)

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.



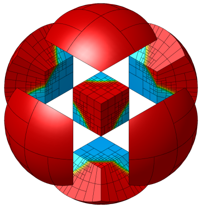
ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Applications – Tensor contractions

- **Domain: High-order (HO) Finite Element (FE) methods, spectral-element (SE)**

Lagrangian Hydrodynamics in the BLAST code[1]



On semi-discrete level our method can be written as

$$\textbf{Momentum Conservation:} \quad \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\mathbf{M}_{\mathbf{v}}^{-1}\mathbf{F}\cdot\mathbf{1}$$

$$\textbf{Energy Conservation:} \quad \frac{\mathrm{d}\mathbf{e}}{\mathrm{d}t} = \mathbf{M}_{\mathbf{e}}^{-1}\mathbf{F}^{\mathbf{T}}\cdot\mathbf{v}$$

$$\textbf{Equation of Motion:} \quad \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}$$

where $\mathbf{v}$, $\mathbf{e}$, and $\mathbf{x}$ are the unknown velocity, specific internal energy, and grid position, respectively; $\mathbf{M}_{\mathbf{v}}$ and $\mathbf{M}_{\mathbf{e}}$ are independent of time velocity and energy mass matrices; and $\mathbf{F}$ is the generalized corner force matrix depending on $(\mathbf{v}, \mathbf{e}, \mathbf{x})$ that needs to be evaluated at every time step.

[1] V. Dobrev, T.Kolev, R.Rieben. *High order curvilinear finite element methods for Lagrangian hydrodynamics*. SIAM J.Sci.Comp.34(5), B606–B641. (36 pages)

Need:

- **Tensor contractions for multicore CPUs, GPUs, and Xeon Phi** (very good results on all already published)
- **Batched solvers (LU/Cholesky) and eigensolvers**

Index reordering/reshape

If we store tensors as column-wise 1D arrays,

$$M^{nd_1\times nd_2\times nd_1\times nd_2}_{i_1,i_2,j_1,j_2} = M^{nd\times nd}_{i,j} = M^{nd^2}_{i+nd\,j} = M^{nd^2}_{i_1+nd_1i_2+nd(j_1+nd_1j_2)}$$

, *i.e.*, $\boldsymbol{M}$ can be interpreted as a 4th order tensor, a $\boldsymbol{nd}$ x $\boldsymbol{nd}$ matrix, or a vector of size $\boldsymbol{nd}^2$, without changing the storage. We can define

$$Reshape(T)^{m_1\times\cdots\times m_q}_{j_1,\cdots,j_q} = T^{n_1\times\cdots\times n_r}_{i_1,\cdots,i_r}$$

as long as $n_1 \ldots n_r = m_1 \ldots m_q$ and for every $i_{1..r}, j_{1..q}\; i_1 + n_1 i_2 + \ldots + n_1 n_2 \ldots n_{r-1} i_r = j_1 + m_1 j_2 + \ldots + m_1 m_2 \ldots m_{q-1} j_q$.

Contractions can be implemented as a sequence of pairwise contractions. There is enough complexity here to search for something better: code generation, index reordering, and autotuning will be used, *e.g.*, contractions (3a) - (4f) can be implemented as tensor index-reordering plus gemm $\boldsymbol{A}, \boldsymbol{B} \to \boldsymbol{A}^T\boldsymbol{B}$.

For example:

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

Can be written as

$$\mathrm{Reshape}(C)^{nd1\times(nd2nd3)} = A^T\,\mathrm{Reshape}(B)^{nq1\times(nd2nd3)}$$

Reference: **A. Abdelfattah, M. Baboulin, V. Dobrev, J. Dongarra, C. Earl, J. Falcou, A. Haidar, I. Karlin, Tz. Kolev, I. Masliah, S. Tomov,**
High-Performance Tensor Contractions for GPUs,
The International Conference on Computational Science (ICCS 2016), San Diego, CA, June 6—8, 2016.

Applications – Numerical LA

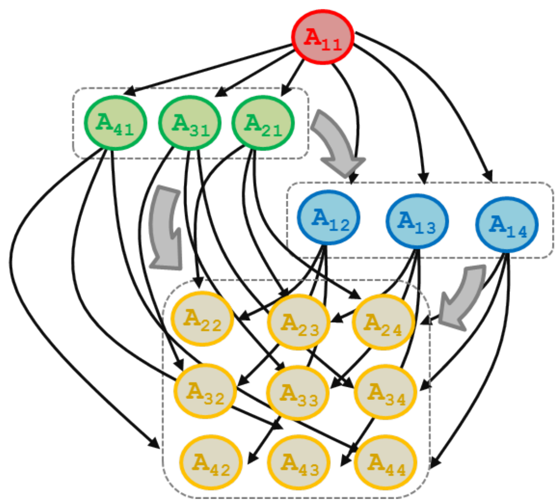
Need of **Batched** routines for **Numerical LA**

[e.g., sparse direct multifrontal methods, preconditioners for sparse iterative methods, tiled algorithms in dense linear algebra, etc.;]
[collaboration with Tim Davis at al., Texas A&M University]

Sparse / Dense Matrix System

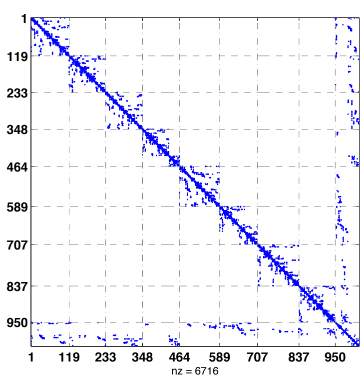
$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

DAG-based factorization



To capture main LA patterns needed in a numerical library for Batched LA

- **LU, QR,** or **Cholesky** on small diagonal matrices
- **TRSM**s, **QR**s, or **LU**s
- **TRSM**s, **TRMM**s
- **Updates (Schur complement) GEMM**s, **SYRK**s, **TRMM**s



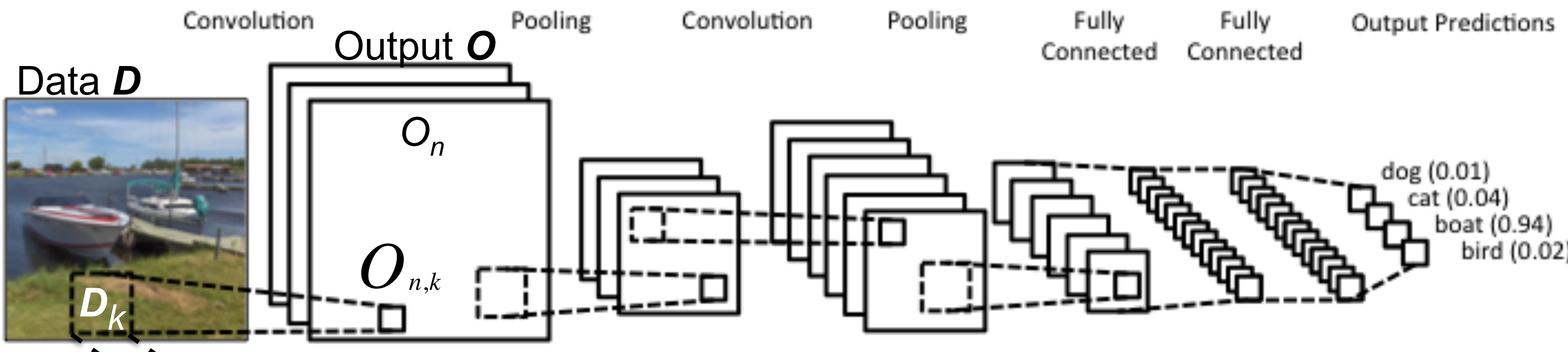
- Example matrix from Quantum chromodynamics
- Reordered and ready for sparse direct multifrontal solver
- Diagonal blocks can be handled in parallel through batched LU, QR, or Cholesky factorizations

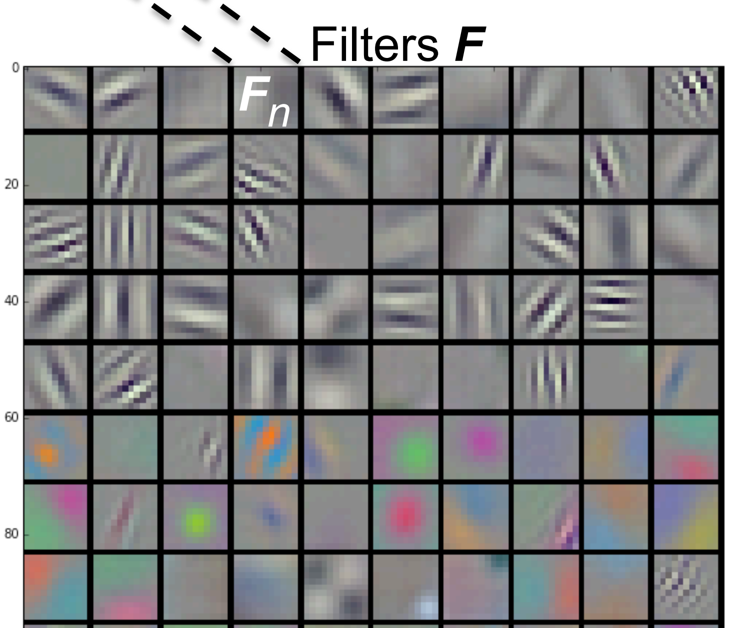
ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE Department of Electrical Engineering and Computer Science

Applications – Machine Learning

Need of **Batched and/or Tensor contraction** routines in **machine learning**

e.g., Convolutional Neural Networks (CNNs) used in computer vision
Key computation is convolution of Filter Fi (feature detector) and input image D (data):





Convolution operation:

- 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** (and hence, efficiently implemented; among other approaches)

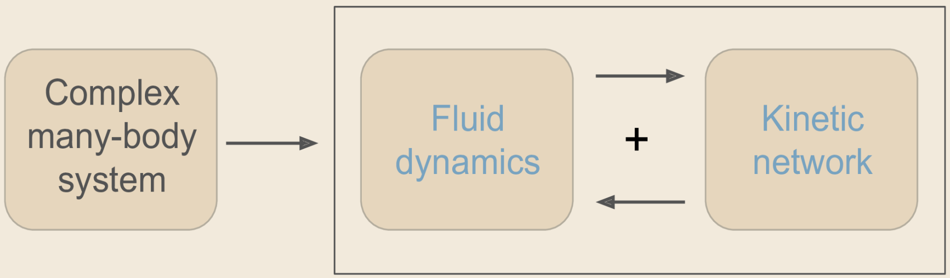
ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Applications – Multi-physics simulations

Fluid Dynamics Plus Kinetics Approximation

Many physical systems can be modeled by a fluid dynamics plus kinetics approximation.



Reference: A. Haidar, S. Tomov, A. Abdelfattah, M. Guidry, J. Billings, and J. Dongarra,
Optimisation Techniques Toward Accelerating Explicit Integration for Large Kinetic Networks.
International Conference on Parallel Processing, Philadelphia, PA, USA ICPP 2016.

Applications – Multi-physics simulations

Integrating Stiff Equations Numerically
(e.g., N coupled ODEs)

Explicit numerical integration:

To advance the solution from time t_n to t_{n+1}, only information already available at t_n is required.

Implicit numerical integration:

To advance the solution from time t_n to t_{n+1}, information at the new point t_{n+1} is required, implying an *iterative solution*.

Thus, for numerical integration

- Explicit methods are *inherently simple, but potentially unstable.*
- Implicit methods are *inherently complicated, but stable.*

Reference: A. Haidar, S. Tomov, A. Abdelfattah, M. Guidry, J. Billings, and J. Dongarra,
Optimisation Techniques Toward Accelerating Explicit Integration for Large Kinetic Networks.
International Conference on Parallel Processing, Philadelphia, PA, USA ICPP 2016.

Applications – Multi-physics simulations

Fundamental Sources of Stiffness

Negative populations

$$\frac{d\boxed{y_i}}{dt} = \boxed{F_i^{+} - F_i^{-}}$$

Macroscopic equilibration

$$= (f_1^{+} + f_2^{+} + \dots)_i - (f_1^{-} + f_2^{-} + \dots)_i$$

$$= (f_1^{+} - f_1^{-})_i + (f_2^{+} - f_2^{-})_i + \dots = \sum_j \boxed{(f_j^{+} - f_j^{-})_i}$$

Microscopic equilibration

Network of species y_i, e.g., $i = 1..150$

j denotes reactions, e.g., $j = 1..1604$

$(f_j)_i$ are fluxes between species

The key to stabilizing explicit integration is to understand the three basic sources of stiffness for a typical reaction network:

- *Negative populations,*
- *Macroscopic equilibration*
- *Microscopic equilibration.*

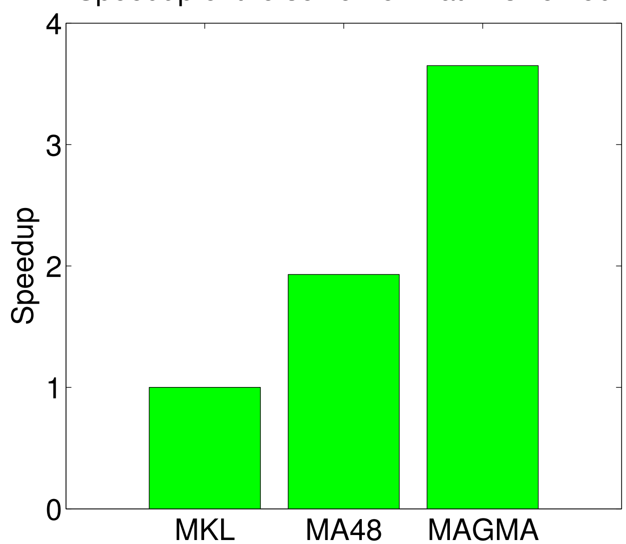
Reference: A. Haidar, S. Tomov, A. Abdelfattah, M. Guidry, J. Billings, and J. Dongarra, *Optimisation Techniques Toward Accelerating Explicit Integration for Large Kinetic Networks*. International Conference on Parallel Processing, Philadelphia, PA, USA ICPP 2016.

Applications – Multi-physics simulations

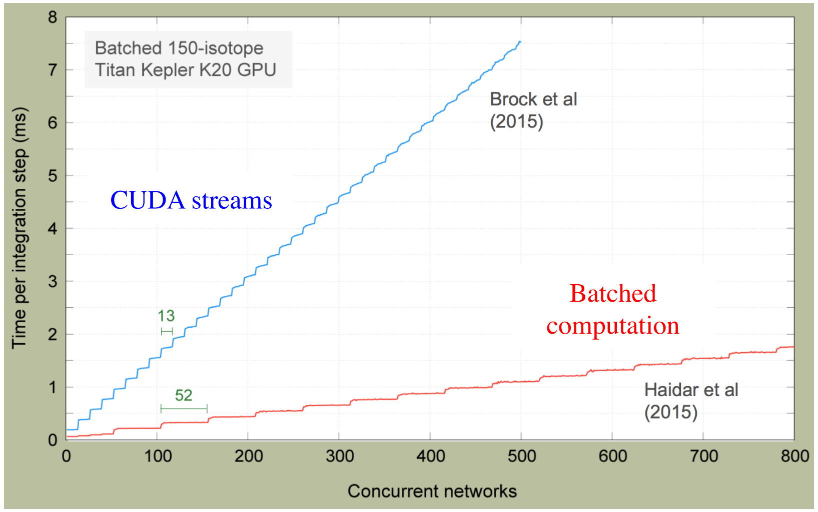
Multi-physics problems need **Batched LA on small problems**

Collaboration with ORNL and UTK physics department (Mike Guidry, Jay Billings, Ben Brock, Daniel Shyles, Andrew Belt)

- Many physical systems can be modeled by a fluid dynamics plus kinetic approximation
  e.g., in astrophysics, stiff equations must be integrated numerically:
  - **Implicitly**; standard approach, leading to need of batched solvers (e.g., as in XNet library)
  - **Explicitly**; a new way to stabilize them with Macro- plus Microscopic equilibration
    need batched tensor contractions of variable sizes



Additional acceleration achieved through MAGMA Batched



An additional **7x speedup**

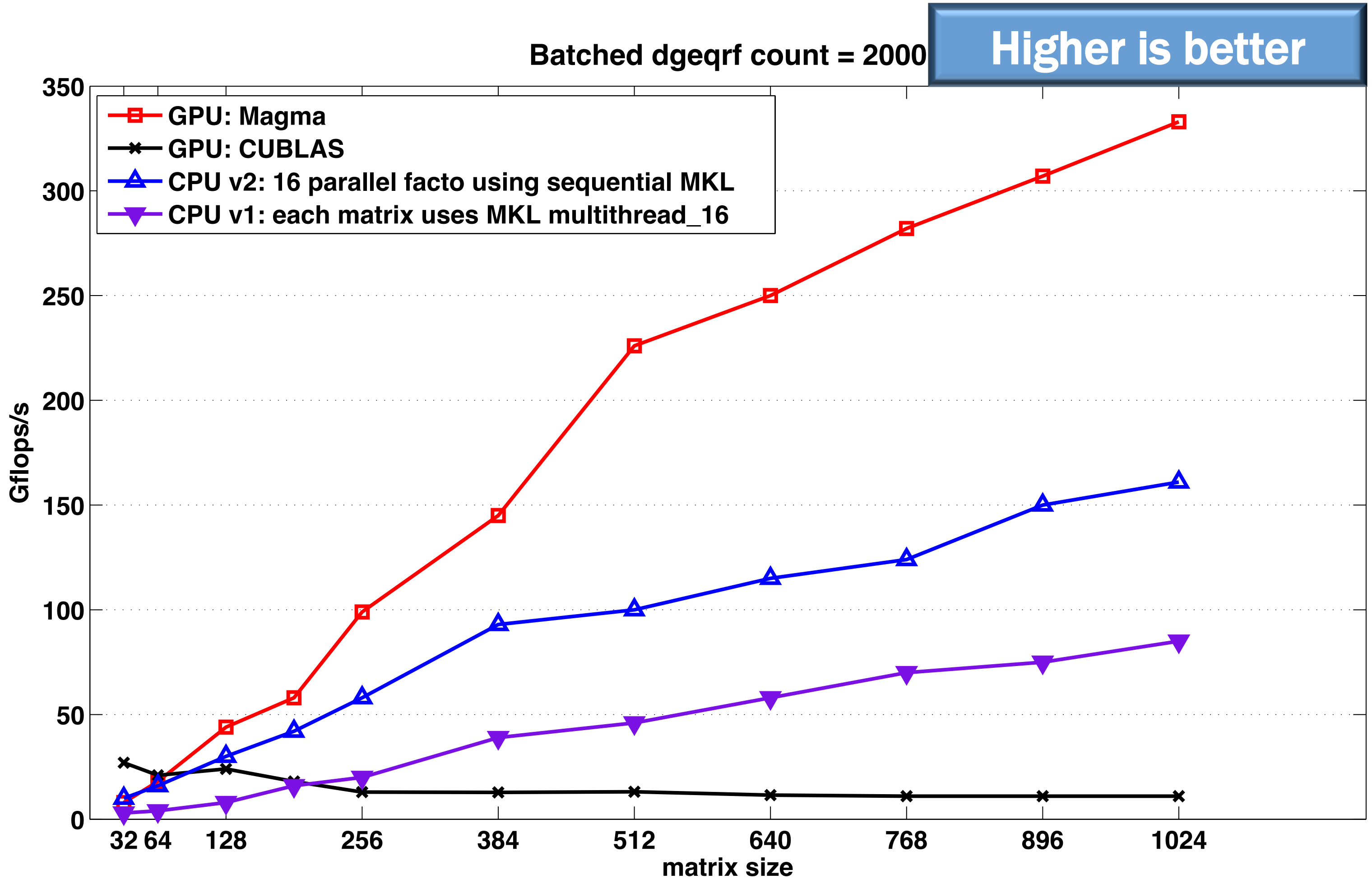
Reference: A. Haidar, S. Tomov, A. Abdelfattah, M. Guidry, J. Billings, and J. Dongarra,
Optimisation Techniques Toward Accelerating Explicit Integration for Large Kinetic Networks.
International Conference on Parallel Processing, Philadelphia, PA, USA ICPP 2016.

Design and optimization strategies

- Multiple algorithmic versions/designs
  - Parallel swapping, panel blocking, recursion, left/right/top-looking, etc.
- Data Access Optimizations and Loop Transformation Techniques
- Register Data Reuse and Locality
- A Cache-based Approach
- A Shared Memory based Approach
- Instruction Mix
- TB-level Concurrency
- Template code based and autogeneration

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

MAGMA Batched Computations Comparison to CPUs



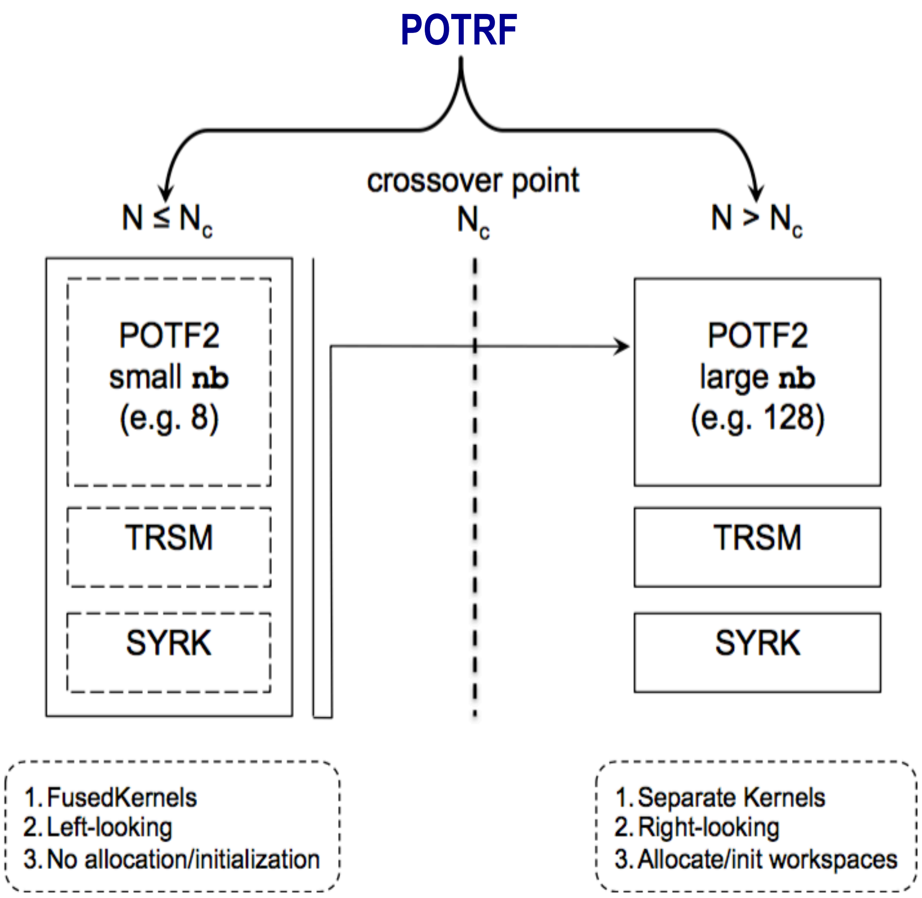
- 2x8-core Intel Xeon E5-2670 Sandy Bridge socket
- NVIDIA Kepler K40 GPU

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE Department of Electrical Engineering and Computer Science

Design and optimization strategies …

Overall design

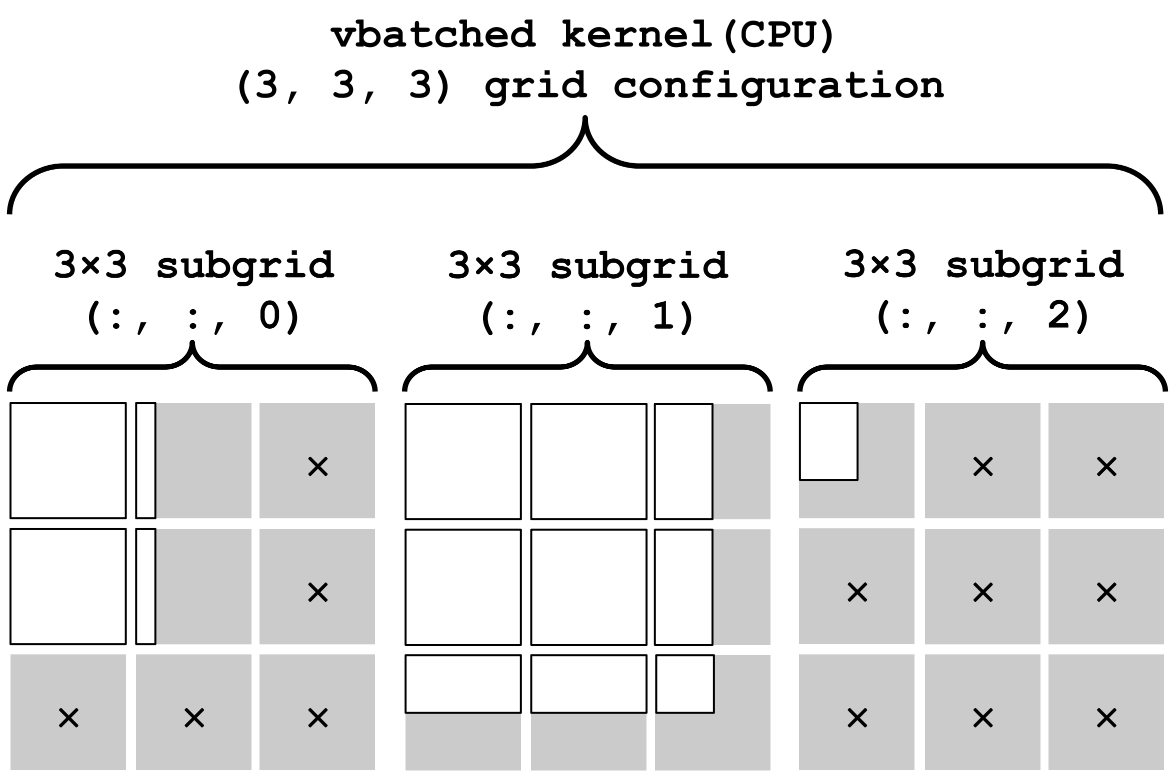


- Recursive multi-level blocking for the panels
- Data storage, e.g., standard vs. interleaved
- Kernel fusion and optimizations for data reuse
  - ***Loop-inclusive***
    (results in 1 GPU kernel)
  - ***Loop-exclusive***
    (outer loop launched from CPU)
- TB-level concurrency
  - For small matrices may need more than one matrix on a Thread Block (TB)
- Performance tuning
  - To handle complexity, **must be done through an autotuning framework**

Variable size techniques

Early Termination Mechanisms (ETMs) and scheduling

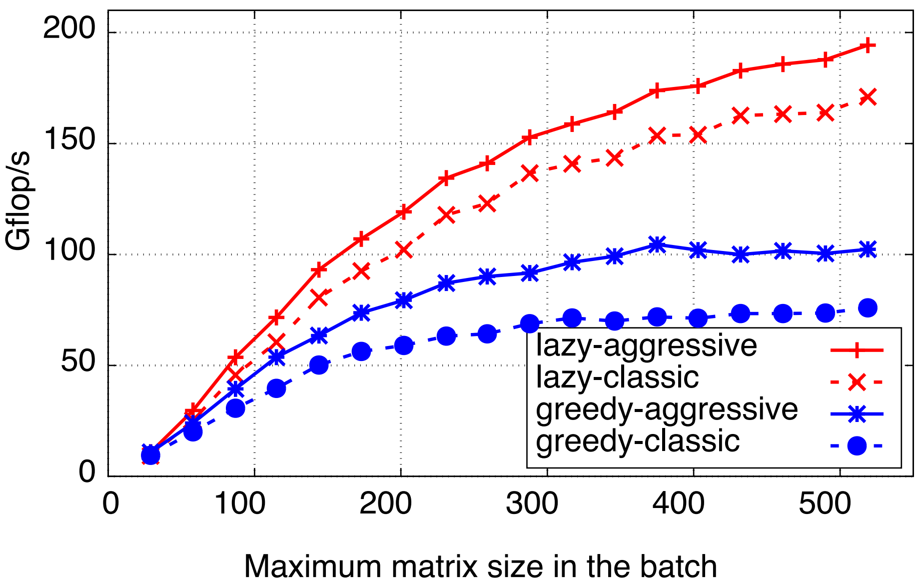
- Kernels are launched to accommodate the largest matrix
- ETMs terminate TBs that may not do work for smaller matrices
  - ***Classic*** vs**. *Aggressive*** (terminate entire TBs vs. TBs + individual threads)
  - ***Greedy*** vs. ***Lazy*** (all matrix factorizations start vs. delaying small ones)
- Used in GEMM, and consequently, TRSM and SYRK



Performance results (variable sizes)

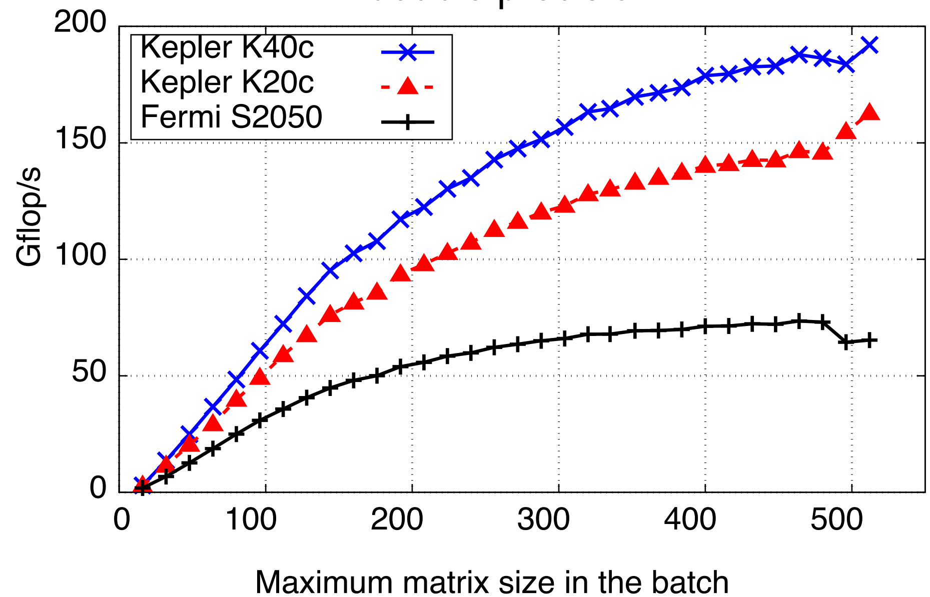
ETM and scheduling techniques

Tuning on K40c GPU, batchCount = 3,000, double precision



Performance on different GPUs

batchCount = 1,000, double precision



Paper also includes:

- Results with various matrix-size distributions (shown is Gaussian)
- Multicore CPU algorithms (using OpenMP) and optimization techniques
    - Padding, static and dynamic scheduling effects

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

MAGMA Batched Computations

Summary

- Batched computation can give a boost in performance for problem with very small sizes
- Traditional algorithmic design might not be the best direction
  - we need a new way of thinking
  - revisit and redesign algorithm to take advantage of the hardware specifics
- Performance modeling can help analyzing algorithm and their implementation, for example
  - An optimized GPU function cannot be efficient for all kind of computation, it depend on the context used for
  - Small computation are delicate and requires specific kernels (building block or fused).
  - Low level API is required to avoid overhead and context switching

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Future Directions

- Extended functionality
  - Variable sizes (work in progress)
  - Mixed-precision techniques
  - Sparse direct multifrontal solvers & preconditioners
  - Applications
- Further tuning
  - autotuning
- GPU-only algorithms and implementations
- MAGMA Embedded

Collaborators and Support

MAGMA team

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

PLASMA team

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

Collaborating partners

University of Tennessee, Knoxville
Lawrence Livermore National Laboratory,
Livermore, CA
University of California, Berkeley
University of Colorado, Denver
INRIA, France (StarPU team)
KAUST, Saudi Arabia

Umeå University

INRIA

Rutherford Appleton Laboratory

University of Manchester