MATEDOR

MAtrix, TEnsor, and Deep-learning Optimized Routines

INNOVATIVE COMPUTING LABORATORY

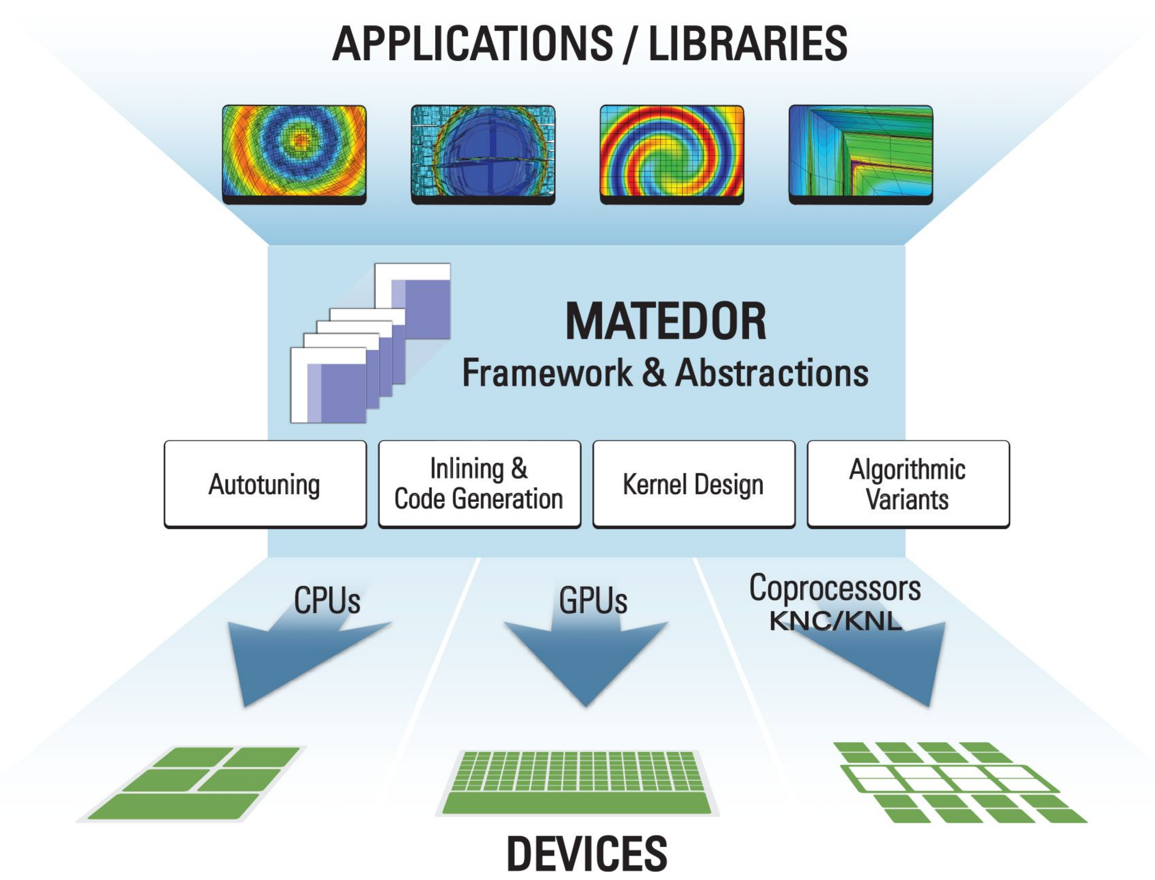
Ahmad Abdelfattah Jack Dongarra
Stanimire Tomov Ichitaro Yamazaki
UNIVERSITY OF TENNESSEE
Azzam Haidar NVIDIA

SCOPE

The MAtrix, TEnsor, and Deep-learning Optimized Routines (MATEDOR) project develops software technologies and standard APIs, along with a sustainable and portable library, for large-scale computations that can be broken down into very small matrix or tensor computations. The main target of MATEDOR is to accelerate applications from important fields that fit this profile, including deep learning, data mining, astrophysics, image and signal processing, hydrodynamics, and more.

Sustainable and Performance-Portable Software

MATEDOR is a high-performance numerical library for batched linear algebra subroutines autotuned for modern processor architectures and system designs. The MATEDOR library includes LAPACK-compliant routines that target many small dense problems, tensor, and application-specific operations, e.g., deep-learning. These routines are constructed as much as possible out of calls to batch BLAS routines and their look-alikes required in sparse computation context.



Standard Interface for Batch Routines

Working closely with interested application communities, we define modular and language-agnostic interfaces that can be implemented to work seamlessly with compilers and to be optimized using techniques such as code generation and inlining. This enables application developers, compilers, and runtime systems to express computational workloads as one or more calls to standard batch routines, and would allow the entire linear algebra (LA) community to collectively attack a wide range of small matrix or tensor problems. Success in such an effort requires innovations in interface design, computational and numerical optimization, as well as packaging and deployment on the user side to trigger final stages of tuning at the moment of execution.

Standard C Interface:

1. Similar to the MAGMA library [3][4] and cuBLAS.
2. Pointer-to-pointer (P2P) interface. Problems are not necessarily equidistant from each other.
3. Separate APIs for fixed and variable size batch problems.

```
void dgemm_batched(  trans_t transA, trans_t transB, int_t m, int_t n, int_t k,
                     double alpha, double const * const * dA_array, int_t ldda,
                                   double const * const * dB_array, int_t lddb,
                     double beta,  double **               dC_array, int_t lddc,
                     int_t batch, queue_t queue );
```

```
void dgemm_vbatched(  trans_t transA, trans_t transB, int_t* m, int_t* n, int_t* k,
                      double alpha, double const * const * dA_array, int_t* ldda,
                                    double const * const * dB_array, int_t* lddb,
                      double beta,  double **               dC_array, int_t* lddc,
                      int_t batch, queue_t queue );
```

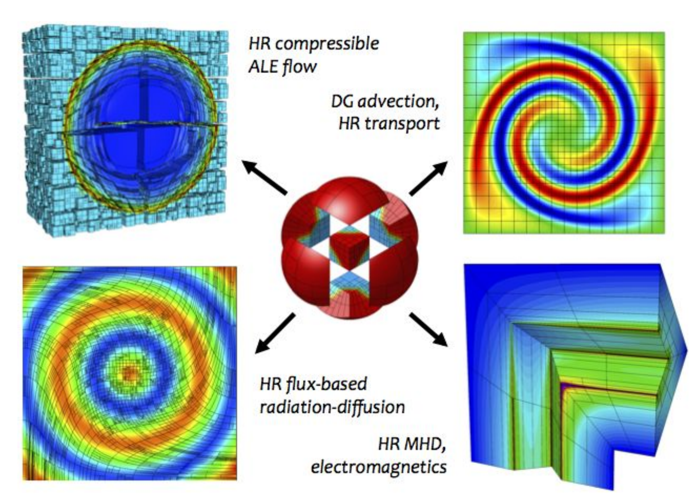
Standard C++ Interface:

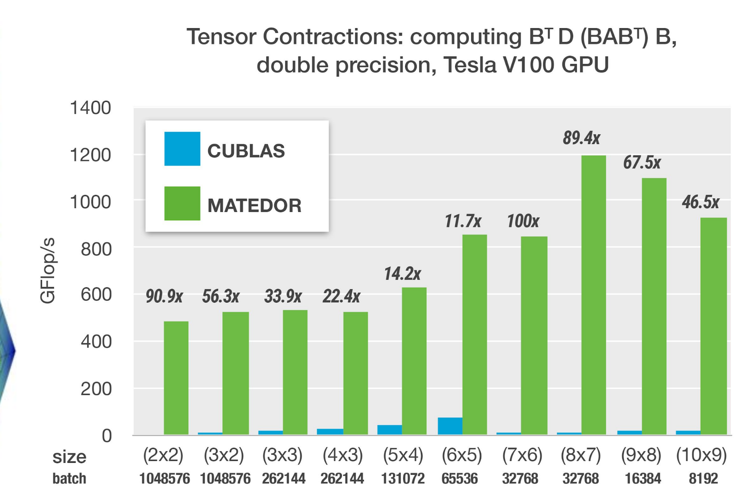
1. Uses `std::vector` C++ containers [5]
2. Significantly more flexible
3. A unified API that supports both fixed-size and variable size problems
4. Easily extendable to support groups and stride-based workloads
5. Simplified naming scheme through namespaces and overloading

```
namespace blas{
namespace batch{
void gemm(
     vector<blas::Op> const &transA, vector<blas::Op> const &transB,
     vector<int64_t>  const &m,
     vector<int64_t>  const &n,
     vector<int64_t>  const &k,
     vector<double >  const &alpha,
     vector<double*>  const &Aarray, vector<int64_t>  const &ldda,
     vector<double*>  const &Barray, vector<int64_t>  const &lddb,
     vector<double >  const &beta,
     vector<double*>  const &Carray, vector<int64_t>  const &lddc,
     const size_t batch,             vector<int64_t>         &info );
}  // namespace batch
}  // namespace blas
```

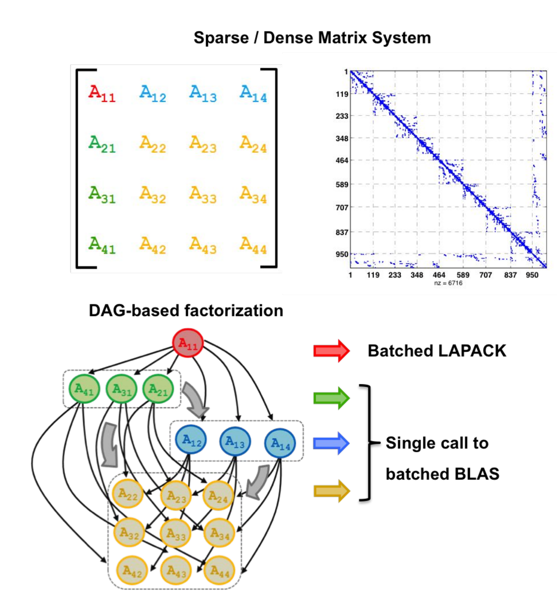
Applications

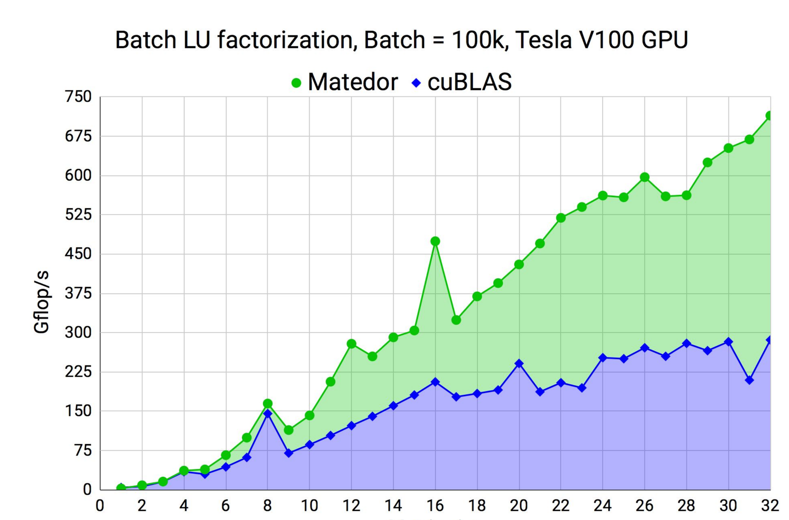
1. Tensor Contractions in High-order FEM and Applications [1]

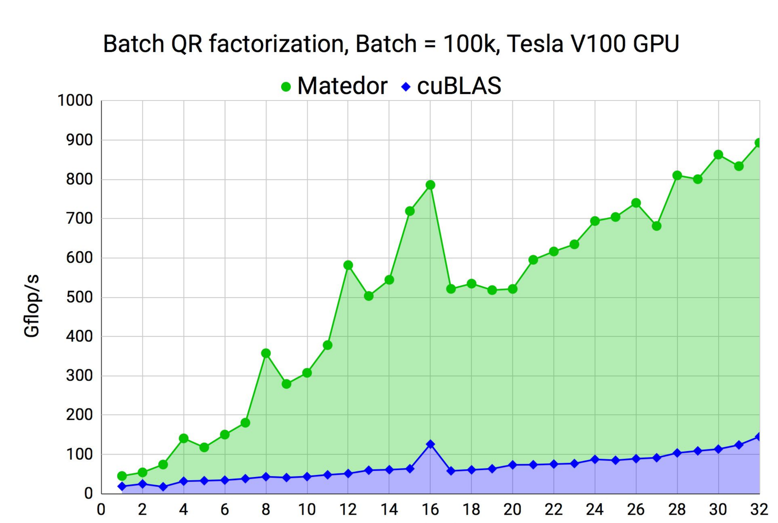


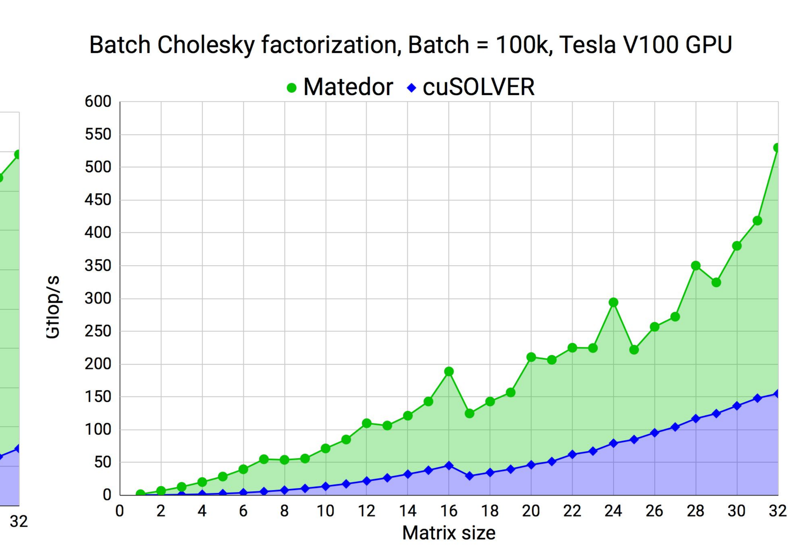


2. Dense/sparse System Solvers & Preconditioners [3][4]

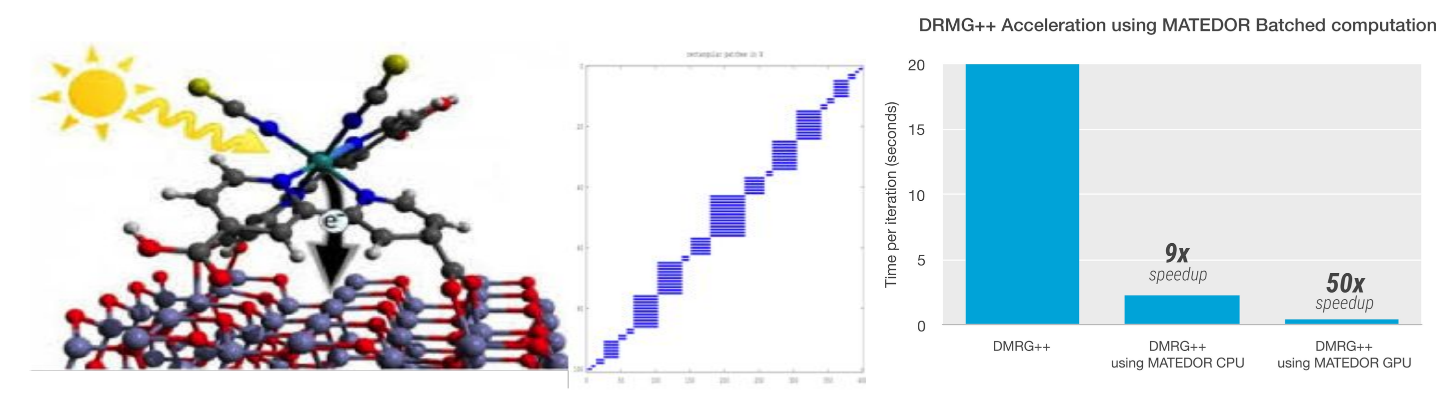




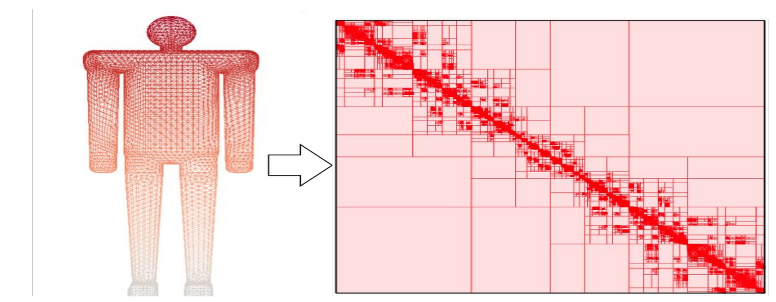




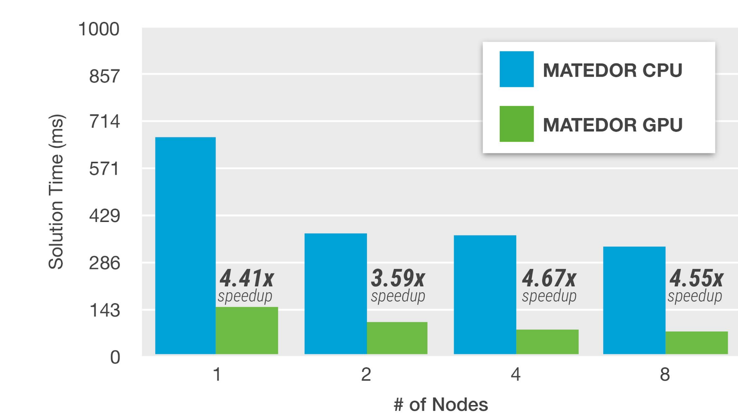
3. Density Matrix Renormalization Group (DMRG++)



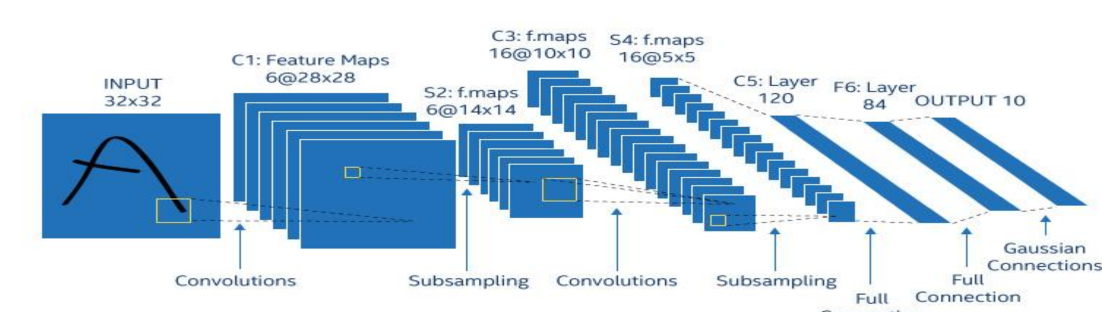
4. Distributed hierarchical linear solver [6]

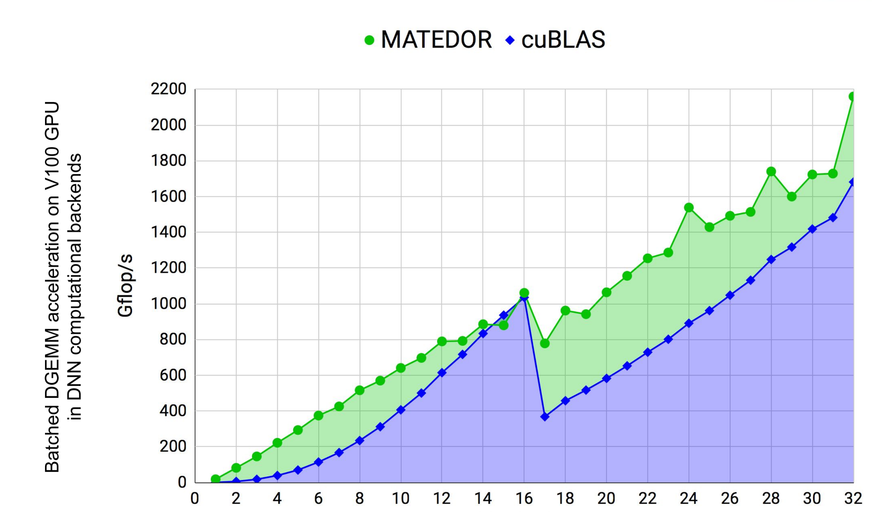


Hierarchical Linear Solver, 2 P100 GPUs per node



5. Deep Neural Networks and Data Analytics [1]





Resources

MATEDOR in the MAGMA Library	Standard C++ Interface	Community Engagement
https://icl.utk.edu/magma/	http://www.icl.utk.edu/publications/swan-004	http://icl.utk.edu/bblas/

Acknowledgements

This work is partially supported by NSF Grant No. OAC 1740250 and CSR 1514286, NVIDIA, and the Department of Energy under the Exascale Computing Project (17-SC-20-SC and LLNL subcontract under DOE contract DE-AC52-07NA27344).

References

1. A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra, "**Tensor Contractions using Optimized Batch GEMM Routines,**" March 26-29 2018, GPU Technology Conference (GTC), Poster, San Jose, CA. [Online]. Available: http://icl.cs.utk.edu/magma/software/
2. L. Ng, K. Wong, A. Haidar, S. Tomov, and J. Dongarra, "**MagmaDNN High-Performance Data Analytics for Manycore GPUs and CPUs**," December 2017, MagmaDNN, 2017 Summer Research Experiences for Undergraduate (REU), Knoxville, TN. [Online]. Available: http://icl.cs.utk.edu/magma/software/
3. A. Haidar, A. Abdelfattah, M. Zounon, S. Tomov, and J. Dongarra, "**A Guide For Achieving High Performance With Very Small Matrices On GPU: A case Study of Batched LU and Cholesky Factorizations**," IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99, pp. 1–1, 2017.
4. A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra, "**Batched one-sided factorizations of tiny matrices using GPUs: Challenges and countermeasures,**" Journal of Computational Science, 2018. (http://www.sciencedirect.com/science/article/pii/S1877750317311456)
5. Ahmad Abdelfattah, Konstantin Arturov, Cris Cecka, Jack Dongarra, Chip Freitag, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and PanruoWu. **"C++ API for Batch BLAS"**. Technical Report 4, ICL-UT-17-12, 2017.(http://www.icl.utk.edu/publications/swan-004)
6. I. Yamazaki, A. Abdelfattah, A. Ida, S. Ohshima. S. Tomov, R. Yokota, J. Dongarra, " **Performance of Hierarchical-matrix BiCGStab Solver on GPU clusters,**" 2018 IEEE International Parallel & Distributed Processing Symposium (IPDPS'18).

INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY OF TENNESSEE KNOXVILLE
NSF