Towards a High-Performance Tensor Algebra Package for Accelerators
Marc Baboulin, Veselin Dobrev, Jack Dongarra, Christopher Earl, Joel Falcou, Azzam Haidar, Ian Karlin, Tzanio Kolev, Ian Masliah, Stanimire Tomov

To cite this version:
Towards a High-Performance Tensor Algebra Package for Accelerators

Abstract
Numerous important applications, e.g., high-order FEM simulations, can be expressed through tensors. Examples are computation of FE matrices and SpMV products expressed as generalized tensor contractions. Contractions by the first index can often be represented as tensor index reordering plus gemm, which is a key factor to achieve high-performance. We present ongoing work on the design of a high-performance package in MAGMA for Tensor algebra that includes techniques to organize tensor contractions, data storage, and parameterization related to batched execution of large numbers of small tensor contractions. We apply auto-tuning and code generation techniques to provide an architecture-aware, user-friendly interface.

Motivation
Numerous important applications can be expressed through tensors:
- High-order FEM simulations
- Signal Processing
- Numerical Linear Algebra
- Numerical Analysis
- Deep Learning
- Graph Analysis
- Neuroscience and more
- Data Mining

The goal is to design a:
- High-performance package for Tensor algebra
- Built-in architecture awareness (GPU, Xeon Phi, multicores)
- User-friendly interface

Example cases

Numerical linear algebra:
- A 4-dimensional tensor contraction
- rank-k update on matrices in tile format (k can be small, e.g., sub-vector/warp size)
- Must determine (in software) if possible to do through batched GEMM kernels

Lagrangian Hydrodynamics in the BLAST code:
On semi-discrete level our method can be written as:
- Momentum Conservation:
- Energy Conservation:
- Equation of Motion:

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


Tensor operations in high-order FEM
Consider the FE mass matrix \( \mathbf{M}_f \), for an element/zone \( E \) with weight \( \mu \), as a 2-dimensional tensor:
\[
(M_{ij})_{E} = \sum_{n} \mu_{n} (m_{n} v_{n}(i) v_{n}(j) / h_{n}(i) / h_{n}(j)),
\]
where
- \( i,j \) are the FE degrees of freedom (dofs)
- \( n \) is the number of quadrature points
- \( \{v_{n},h_{n}\} \) are the FE basis functions on the reference element
- \( \{m_{n}\} \) is the determinant of the element transformation
- \( \{i_{n}\} \) and \( \{j_{n}\} \) are the points and weights of the quadrature rule

Take the \( \mathbf{a} x \mathbf{w} \) matrix \( \mathbf{B}_a \), and \( (\mathbf{d}_n w_n) = \mathbf{M}_f w_n(i) / h_{n}(i) / h_{n}(j) \).
Then,
\[
(M_{ij})_{E} = \sum_{n=1}^{N_{b}} B_{a}(i) B_{w}(j), \quad \text{or omitting the E subscript}
\]
\[
M = B^T DB
\]
Using FE of order \( p \), we have \( \mathbf{ad} = \mathbf{Op}(d) \) and \( \mathbf{aw} = \mathbf{Op}(w) \)

Summary of kernels needed:
- Assembly of \( \mathbf{M} \) referred as equations (1) & (2) below
- Evaluations of \( M \) times \( \mathbf{v} \), referred as equations (3) & (4) below

Approach and Results
User-friendly interface
To provide various interfaces, including one using C++11.

Code generation
C++11 features will be used as much as possible. Additional needs will be handled by defining a domain-specific embedded language (DSEL). This technique is used in C++ to take advantage of DSL features while using the optimizations provided by a standard compiler: it will handle the generation of versions (index reordering, next) to be empirically evaluated and be part of the autotuning framework.

Index reordering/reshape
If we store tensors as column-wise 1D arrays,
\[
\mathbf{M}(i,j) = \mathbf{M}_f(i,j), \quad \mathbf{a}(i) = \mathbf{a}(i), \quad \mathbf{w}(j) = \mathbf{w}(j)
\]
\( i,j \) can be interpreted as a 4th order tensor, a \( a x w \) matrix, or a vector of size \( \mathbf{aw} \) without changing the storage. We can define
\[
\text{Reassembly}(T)_{i,j} = \text{Reshape}(T)_{i-1,j-1} = \sum_{k=1}^{p} T_{i-k,j-k} + \text{offset}
\]
\( \text{Reshape} \) as the contraction of a 4th order tensor, \( \mathbf{aw} x \mathbf{M} \), or a vector of size \( \mathbf{aw} \), with an offset.

Conclusions and Future directions
- High-performance package on Tensor Algebra has the potential for high-impact on a number of important applications
- Multidisciplinary effort
- Current results show promising performance, where various components will be leveraged from autotuning MAGMA Batched linear algebra kernels, and BLAST from LLNL
- This is an ongoing work