Towards a High-Performance Tensor Algebra Package for Accelerators


Abstract: Numerous important applications, e.g., high-order FEM simulations, can be 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 number of small tensor contractions. We apply auto-tuning and code generation techniques to provide an architecture-aware, user-friendly interface.

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

Tensor operations in high-order FEM
Consider the FE mass matrix $M$ for an element $E$ with weight $p$, as a 2-dimensional tensor:

$$
(M_{E})_{ij} = \sum_{q=1}^{nq} m_{q} p_{i}(q) p_{j}(q) \langle \phi_{i}, \phi_{j} \rangle,
$$

where $nq$ is the number of FE degrees of freedom (dof), $m_{q}$ is the number of quadrature points, $p_{i}(q)$, $p_{j}(q)$, the reference element transformation, $\langle \phi_{i}, \phi_{j} \rangle$, are the points and weights of the quadrature rule.

Take the $n \times m$ matrix $B_{k} = p_{i}(k)B_{i}$, or omitting the $E$ subscript $M = B^{T}DB$.

Using FE of order $p$, we have $n = (p + 1)2^{d}$ and $m = (p + 1)2^{d}$, so $B$ is dense $(p + 1)2^{d} \times (p + 1)2^{d}$.

If the FE basis and the quadrature rule have tensor product structure, we can decompose dofs and quadrature point indices logical coordinate axes $i = (i_{1}, \ldots, i_{d}, j_{1}, \ldots, j_{d})$, $n = O(p_{1}, \ldots, p_{d})$, $m = O(p_{1}, \ld{...} ,p_{d})$.

Summary of kernels needed:
- Assambly of $M$, referred as equations (1) & (2) below
- Evaluations of $M$ times $V$, referred as equations (3) & (4) below

Example cases

Lagrangian Hydrodynamics in the BLAST code

- On semi-discrete level our Momentum conservation

\begin{equation}
\frac{dv}{dt} = -M_{v}F-v
\end{equation}

and Energy Conservation

\begin{equation}
\frac{dE}{dt} = \gamma
\end{equation}

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

User-friendly interface
Towards a high-performance package, including one using C++11.

Top level design to provide features of the mawadwe library (http://gthub.com/mawadwe/)

Code Generation
C++11 features will be used as much as possible. Additional needs will be handled through defining a domain specific language (DSL). DSL will handle 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 10 arrays, it can be interpreted as a 4th order tensor: a $n \times m \times r \times r$ matrix, or a vector of size $n \times m \times r \times r$, without changing the storage. We can define $reshape(V_{0}, \ldots, V_{r-1})$, or omitting the $E$ subscript $M = B^{T}DB$. 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., Constructions (3a) - (4f) can be implemented as tensor index-reordering plus gemm $A \rightarrow B^{T}$.

Batched LA
Tensor contractions are transformed through reshapes to batched LA operations, many of which available in MAGMA[11] (including LU, QR, Cholesky, GEMM, GEMV, TRSM, SYRK).

Autotuning
We are developing fixed-size gemm kernels for GPUs, Xeon Phi, and multicore (see Figure on Right) for a single core Intel Core i7 (or) through an autotuning framework. A number of generic versions are developed and parametrized for performance. The parameters are autotuned (empirically) to find “best” kernels for specific size.

Conclusions and Future directions
- High-performance package on Tensor Algebra has the potential for high-impact on a number of important applications
- Substantial efficiency
- 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


References: