Procedia Computer Science

Volume 80, 2016, Pages 108–118

ICCS 2016. The International Conference on Computational Science

High-Performance Tensor Contractions for GPUs

A. Abdelfattah[1], M. Baboulin[2], V. Dobrev[3], J. Dongarra[1,4], C. Earl[3], J. Falcou[2], A. Haidar[1], I. Karlin[3], Tz. Kolev[3], I. Masliah[2], and S. Tomov[1]

[1] Innovative Computing Laboratory, University of Tennessee, Knoxville, TN, USA
[2] University of Paris-Sud, France
[3] Lawrence Livermore National Laboratory, Livermore, CA, USA
[4] University of Manchester, Manchester, UK

Abstract

We present a computational framework for high-performance tensor contractions on GPUs. High-performance is difficult to obtain using existing libraries, especially for many independent contractions where each contraction is very small, e.g., sub-vector/warp in size. However, using our framework to batch contractions plus application-specifics, we demonstrate close to peak performance results. In particular, to accelerate large scale tensor-formulated high-order finite element method (FEM) simulations, which is the main focus and motivation for this work, we represent contractions as tensor index reordering plus matrix-matrix multiplications (GEMMs). This is a key factor to achieve algorithmically many-fold acceleration (*vs.* not using it) due to possible reuse of data loaded in fast memory. In addition to using this context knowledge, we design tensor data-structures, tensor algebra interfaces, and new tensor contraction algorithms and implementations to achieve 90+% of a theoretically derived peak on GPUs. On a K40c GPU for contractions resulting in GEMMs on square matrices of size 8 for example, we are 2.8× faster than CUBLAS, and 8.5× faster than MKL on 16 cores of Intel Xeon E5-2670 (Sandy Bridge) 2.60GHz CPUs. Finally, we apply autotuning and code generation techniques to simplify tuning and provide an architecture-aware, user-friendly interface.

Keywords: Tensor contractions, Tensor HPC, GPU, Batched linear algebra, FEM, Applications

1 Introduction

The development of high-performance tensor algebra is important due to tensors' frequent use in physics and engineering, where tensors provide a foundational mathematical tool for brief, yet comprehensive, formulations and solutions of problems in areas such as elasticity, fluid mechanics, multi-physics, quantum chemistry, general relativity, and many others [10]. Advances in microprocessors and storage technologies have made it feasible to target higher dimension and accuracy computational approaches that model mutilinear relations, e.g., in recent areas of high interest such as various data analysis applications and machine learning; that can also be formulated through tensors. At the same time, to enable these applications to efficiently

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.302

use tensor computations on current hardware, and in particular GPUs, a number of research challenges must be addressed, including advances in the development of scalable, tensor-based algorithms, autotuning, and code generation techniques, that are targets of this paper, towards setting the foundations for a high-performance tensor algebra library for accelerators [3].

Tensors are multi-dimensional arrays that can be used to describe physical properties featuring multilinear relations. Well known mathematical objects like scalars, vectors, and matrices can be generalized to tensors that are of order zero, one, and two, respectively. Also, tensor transformations like flattening of a tensor to matrices or reshaping of matrices into tensors, can be used to link tensor computations to the developments in high-performance numerical linear algebra (LA). Therefore, similar to many applications, tensor computations can also significantly benefit from representing their computations in terms of BLAS, as well as from other dense LA algorithms and techniques for multicore and GPU architectures. While a comprehensive LAPACK-style initiative to tensors may be still far away [1], this work concentrates on the development of tensor contractions – a building block for tensor computations – through leveraging the current LA developments for GPU and multicore architectures in libraries like BLAS and MAGMA [20], and more specifically the MAGMA Batched computational framework [6, 8, 9].

Microprocessor and storage technology advances have significantly influenced the design of high-performance numerical algorithms and libraries over the years. While humans perceive well algorithms expressed in terms of scalar computations (tensors of order zero), advances towards vector machines in the 70's lead to the development of the LINPACK library to use vector operations (or 1^{st}-order tensors), which was redesigned for performance into LAPACK in the 80's to better use cache-based machines through matrix-matrix operations (2^{nd}-order tensors). Operations on higher-dimensional data were added in the 90's for distributed-memory systems, where ScaLAPACK was designed for 2D-block cyclic matrix distributions. For the emerging in the 00's multicore architectures, the PLASMA library introduced tiled algorithms and tiled data layouts (4^{th}-order tensors; see Figure 1). In the 2010's, the MAGMA libraries were designed for heterogeneous architectures, including current investigations on new data layouts, functionalities, batched computations, and possibly generalizations to tensors, in order to provide applications new functionalities to deal efficiently with multi-dimensional data.

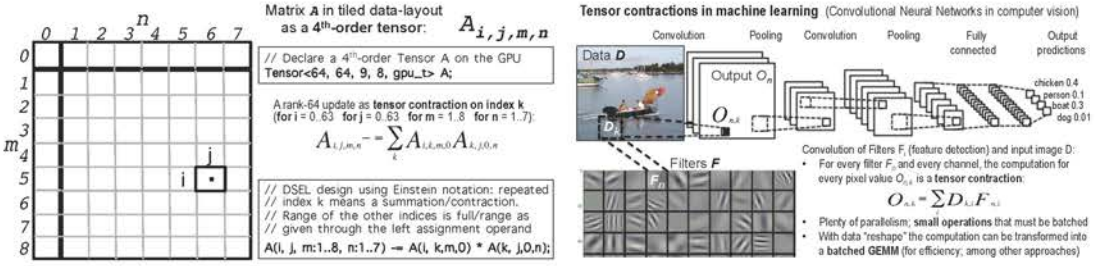

Figure 1: **Left**: Example of a 4^{th}-order tensor resulting from tile matrix layout used in dense LA, a tensor contraction, and a possible tensor contractions design using Einstein summation notation and a Domain Specific Embedded Language (or *DSEL*) . **Right**: Illustration of tensor contractions needed and viable approaches to solve them in machine learning.

Figure 1 Left illustrates the notion of tensor and tensor contraction in DLA, as well as one of our tensor contraction design using a *DSEL* . Figure 1 Right illustrates the need of tensor contractions in machine learning. The computational characteristics in this case are common