A Framework for Distributed Tensor Computations

Martin D. Schatz∗ and Robert van de Geijn (Advisor)*
∗The University of Texas at Austin

1. Introduction

Branches of scientific computing, express data as a tensor which can be viewed as a multi-dimensional analog of a matrix. For instance, chemical methods heavily rely on the tensor contraction operation, which can be viewed as a generalization of matrix-matrix multiplication [1]. The algebra associated with tensors in these applications is referred to as multi-linear algebra, the multi-dimensional analog of linear algebra.

Problems in this area of research frequently require the use of distributed-memory computing architectures to compute the desired method or operation. One common approach to computing multi-linear operations acting on tensors on such architectures is to cast the operation in terms of linear operations acting on matrices and utilize high-performance linear algebra libraries to compute the result [2]. Unfortunately, the underlying linear algebra library may incur higher network communication costs than is necessary as some structure of the multi-linear objects is lost when cast to linear objects (the library considers the objects matrices, which inherently removes some structure of the original tensors). However, as we argue in this work, one can devise a notation which formalizes the data distributions of arbitrary-dimensional tensors mapped to arbitrary-dimensional processing grids as well as relate redistributions to efficient collective communications (analogous to what the theory underlying the Elemental library [3] does for linear algebra operations).

A significant benefit of this is that one can then systematically derive and analyze different algorithms for tensor operations, enabling the automatic generation of implementations for tensor operations. In this work, we introduce this notation and show its utility for the tensor contraction operation.

This document is organized as follows:

- Section 2 briefly describe the main challenge associated with computing tensor operations on distributed-memory architectures which this work addresses
- Section 3 briefly describes the solution proposed by this work
- Section 4 provides concluding remarks

2. Data Redistribution

As mentioned in Section 1, mapping multi-linear operations to linear operations can potentially incur a greater communication of data than is necessary due to some loss of structure of the objects. The practical effect of this is that a communication pattern cannot easily be identified and thus a general, likely less efficient, communication pattern must be utilized in its place. Consider the tensor contraction

\[ C_{ikmn} = A_{ijm} B_{knj}, \]

defined element-wise as
\[ e_{i_0i_1i_2i_3} = \sum_{\ell=0}^{J-1} A_{i_0i_2\ell} \cdot B_{i_1i_3\ell} \]

where \( J \) is the dimension or length of the index \( j \). This can be computed as a matrix-matrix multiplication by reordering data. Since this is being performed in a distributed-memory environment and we cannot, without a formal specification, assume a structured relationship between how elements are distributed in the tensor representation and how they are distributed in the matrix representation, a reordering of data could involve communication with every process in the processing grid (each process requiring a unique set of elements, which are all owned by a single process). The practical consequence is that we are forced to use a general, potentially more expensive, communication pattern, whereas with a formal specification, one could potentially identify that the redistribution required can be completed with a more specialized, likely more efficient, communication pattern. For example, this corresponds to utilizing an all-to-all communication with all processes when only a point-to-point communication within rows of processes is needed.

### 3. Proposed Solution

The notation underpinning Elemental describes how data is distributed among processes of the processing grid when logically viewed as a different shape. For instance, the notation describes how elements of a matrix are distributed on a two-dimensional, \( 2 \times 3 \) processing grid, or a one-dimensional, length 6 vector of process. Further, the theory relates redistributions between the various views to different collective communication types.

The work presented here describes an extension of this notation which allows for elements of arbitrary-dimensional tensors to be mapped to arbitrary-dimensional views of processing grids.

### 4. Conclusion

This work provides a formal notation for describing how elements of arbitrary-dimensional tensors can be distributed to arbitrary-dimensional processing grids and relates redistributions to different collective communication types. In addition, this work allows for the systematic derivation of algorithms, identification of required collective communications, and systematic analysis of different algorithms, allowing for, eventually, the automatic generation of implementations for tensor contraction operations.

### References

