HARVEY MUDD COLLEGE Mathematics Clinic

Final Report for
Sandia National Laboratories

Parallelizing Intrepid Tensor Contractions Using Kokkos

April 2, 2015

Team Members
Brett Collins (Project Manager)
Alex Gruver
Ellen Hui
Tyler Marklyn

Advisor
Jeff Amelang

Liaison
H. Carter Edwards

Abstract

Your abstract should be a *brief* summary of the contents of your report. Don't go into excruciating detail here—there's plenty of room for that later.

If possible, limit your abstract to a single paragraph, as your abstract may be used in promotional materials for the Clinic.

Contents

List of Figures

List of Tables

Acknowledgments

Chapter 1

Sandia National Laboratories

1.1 Background

1.2 Problem

1.3 CPU vs GPU

1.4 Kokkos

Chapter 2

Intrepid

Intrepid is a C++ library developed as part of Sandia's Trilinos Project, providing algebraic operations over multi-dimensional arrays. Tensor contractions are one class of tools implemented by Intrepid, widely used in high-performance simulation software.

One particular type of tensor contraction, two-dimensional matrix multiplication, is known to show great speedup when implemented using CPU and especially GPU parallelism. For this reason among others, the team considered Intrepid tensor contractions to have good theoretical potential to derive significant performance gains from parallelism.

2.1 Tensor Contractions

2.2 Intrepid Contractions Overview

Intrepid provides nine types of tensor contraction, differing by input dimensionality, number of indices contracted away, and output dimensionality. It is most simple to classify Intrepid's tensor contractions by number of indices contracted away and output dimensionality.

Intrepid tensor contractions contract away one, two, or three indices. The output of a single contraction can be a scalar (zero-dimensional), a vector (one-dimensional), or a matrix (two-dimensional). Each combination of number of contraction indices and output dimension is handled by one Intrepid tensor contraction kernel.

In order to more easily discuss specific tensor contraction kernels in Intrepid, it helps to first understand the naming convention used for the kernel names. Each kernel's name contains two suffixes, where the first indi-

Kernel Name	Left Input	Right Input	Output	Contraction Indices
ContractDataDataScalar	1D	1D	Scalar	One
ContractDataDataVector	2D	2D	Scalar	Two
ContractDataDataTensor	3D	3D	Scalar	Three
ContractDataFieldScalar	2D	1D	Vector	One
ContractDataFieldVector	3D	2D	Vector	Two
ContractDataFieldTensor	4D	3D	Vector	Three
ContractFieldFieldScalar	2D	2D	Matrix	One
ContractFieldFieldVector	3D	3D	Matrix	Two
ContractFieldFieldTensor	4D	4D	Matrix	Three

Table 2.1 Summary of the nine Intrepid tensor contraction kernels

cates the dimensionality of the output and the second indicates the number of contraction indices.

String	Position	Meaning
DataData	First Suffix	Scalar Output
DataField	First Suffix	Vector Output
FieldField	First Suffix	Matrix Output
Scalar	Second Suffix	One Contraction Index
Vector	Second Suffix	Two Contraction Indices
Tensor	Second Suffix	Three Contraction Indices

Table 2.2 Intrepid tensor contraction suffixes

For instance, the first suffix `DataData` is used for kernels that produce scalar outputs, and the second suffix `Scalar` is used for kernels that contract away one dimension. Therefore, the Intrepid kernel `ContractDataDataScalar` produces scalar outputs and contracts away one dimension, and so by necessity the inputs for a single contraction must be vectors. All of the Intrepid tensor contraction kernel suffixes are summarized in Table 2.2. The nine tensor contractions in Intrepid are summarized in Table 2.1.

Note that the tensor contraction kernels in Intrepid each actually performs many contractions; for instance, `ContractDataDataScalar`, which performs contractions of two input vectors to a single output scalar (dot product), actually calculates an array of dot products. That is, the inputs are both arrays of vectors, and the output is an array of scalars. This is represented in the code using a dummy index, which we call the `Cell` index.

2.3 ContractDataDataScalar

`ContractDataDataScalar` is the simplest tensor contraction in Intrepid. The kernel takes two arrays of vectors and outputs an array of scalars. A snippet showing the simple serial implementation of `ContractDataDataScalar` can be seen in Figure 2.1.

```
for (int c = 0; c < numCells; ++c) {
    for (int qp = 0; qp < quadraturePoints; ++qp) {
        output[c] += leftInput[c][qp] * rightInput[c][qp];
    }
}
```

Figure 2.1 Code from serial `ContractDataDataScalar`

As shown in Figure 2.1, this kernel contracts away the `Quadrature Points` dimension, leaving only the `Cell` dimension

2.4 ContractDataDataVector

2.5 ContractDataDataTensor

2.6 ContractDataFieldScalar

2.7 ContractDataFieldVector

2.8 ContractDataFieldTensor

2.9 ContractFieldFieldScalar

2.10 ContractFieldFieldVector

2.11 ContractFieldFieldTensor

Chapter 3

Parallelism

3.1 Flat Parallelism

3.2 Reduction

3.3 Slicing

3.4 Tiling

Chapter 4

Experience with Kokkos

4.1 Performance

4.2 Snippets

Chapter 5

Performance