

A Parallel Tensor Product Implementation

(CSE260 Project Proposal)

Bryan Rasmussen *

10 October 2006

1 Introduction

The goal of this project is to write an efficient tensor product routine that takes advantage of parallel computation in an intelligent, scalable way. At a minimum, the routine must be able to compute the tensor products of two rank-4 tensors of reasonable dimensions.

Recall that a tensor is a generalization of the concept of a linear operator. When written with respect to a given reference frame (*i.e.*, a basis), tensors take on the form of multi-dimensional boxes of numbers. The dimensionality of the box is known as the *rank* of the tensor, and the number of points along each direction is called the *dimension*. Notationally, we represent these boxes using indices, so, for example, we write a rank-3 tensor with dimensions (3, 2, 5) as u_{ijk} , where $i = 1, 2, 3$; $j = 1, 2$; and $k = 1, 2, 3, 4, 5$.

In the scope of this project, we assume the existence of a standard basis, so we do not concern ourselves with how the representation transforms with a change in the basis. In particular, we do not distinguish between covariant and contravariant indices. Therefore, as far as this project is concerned, a tensor is simply a multi-dimensional array with a certain organizational structure.

A *tensor product* is the most general bilinear operation possible between two tensors. Considering tensors as multi-dimensional boxes, the tensor product is the pairwise product of all elements in the first tensor with all elements in the second, organized with a straightforward index inheritance. For example, the tensor product of two vectors (rank-1 tensors) is the fa-

*Los Alamos National Lab; bryanras@lanl.gov

miliar *outer product*,

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T = u_i v_j.$$

If $\mathbf{u} = (u_1 \ u_2 \ u_3)^T$ and $\mathbf{v} = (v_1 \ v_2)^T$, then

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \\ u_3 v_1 & u_3 v_2 \end{pmatrix}$$

In general, the tensor product of a rank- m tensor with dimensions (d_1, d_2, \dots, d_m) and a rank- n tensor with dimensions (e_1, e_2, \dots, e_n) is a rank $m+n$ tensor with dimensions $(d_1, d_2, \dots, d_m, e_1, e_2, \dots, e_n)$.

This computation requires at least $d_1 d_2 \cdots d_m e_1 e_2 \cdots e_n$ multiplications, plus the requisite memory allocation and storage. The tricky part will be the parallelization, particularly minimizing the inter-process communication. If, at the end of the time allotted, there is a working piece of code that computes tensor products of rank-4 tensors, has significant speedup, and scales relatively well, then the project will be a success.

2 Schedule

Unfortunately from a scheduling standpoint, this project is Boolean—the code will either work by the due date or it won't. It is possible to make some reasonable intermediate milestones, but we must always keep the final goal in mind. With that caveat, a preliminary schedule is as follows:

Week 1 Serial implementation of tensor product. This could be just some simple `Matlab` code, possibly even downloaded from the Internet.

Week 2 Working on parallelization.

Week 3 Preliminary parallel code. This will probably be inefficient.

Week 4 Somewhat final parallel code. There may actually be several versions that work better or worse depending on the types of tensors and architectures of the system.

Week 5 Benchmarks and speedup computations.