Parallel and Distributed Final Project

Daniel Muckerman

Abstract

This paper examines the matrix vector multiplication algorithm, and various parallel implementations of said algorithm. Matrix multiplication has numerous applications in mathematics, especially in the fields of applied mathematics, physics, and engineering.

Introduction

Matrix multiplication is an important central operation in many numerical algorithms (TODO: research numerical algorithms) and potentially time consuming, both of which have led it to becoming a well-studied problem in numerical computing. For this paper, I will be focusing on matrix vector multiplication, which is a subset of matrix multiplication.

Implementation

For tackling the problem of multiplying a square matrix of dimensions $n * n$ by a vector with length of n, we can determine the product as follows:

$$A = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix}, B = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \tag{1}$$

$$AB = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax & by & cz \\ px & qy & rz \\ ux & vy & wz \end{pmatrix} \tag{2}$$

And here's an example of a properly calculated resultant vector, given an $n * n$ matrix, and n length vector:

$$A = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \end{pmatrix}, \tag{3}$$

$$AB = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 5*1 & 0*0 & 0*0 & 0*2 \\ 0*1 & 1*0 & 0*0 & 0*2 \\ 0*1 & 0*0 & 2*0 & 0*2 \\ 0*1 & 0*0 & 0*0 & 1*2 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 0 \\ 2 \end{pmatrix} \quad (4)$$

There are numerous practical applications for matrix vector multiplication in linear algebra alone, where the problem arises quite often. By choosing appropriate values, A can be used to represent a variety of transformations like rotations, scaling, reflections and shears.

Results

Using a sequential implementation of the problem I wrote in C, I ran the multiplication on random matrices of steadily increasing sizes, and these are the results:

sequential table
sequential graph

And, for comparison, I implemented a parallelized version of the above sequential program, also in C, using pthreads for the multithreading. I ran it through the same gamut of tests, increasingly larger random matrices, and compared the results to the output of the sequential method, to ensure they matched. These are the results:

parallel table
parallel graph

insert hopefully intelligent, and somewhat witty commentary on the comparison here

Extensions?

i dunno

Conclusion

things happened, and there was a winner. probably. maybe not. haven't run the tests yet

Works Cited

- http://en.wikipedia.org/wiki/Matrix_multiplication
- Old Graphics Lectures