Matthew Turner

CSCE435-500

Timothy Davis

# Project 1 Report

**Task 1**

This task was simply implementing the naïve matrix multiply algorithm described in Problem 2.4 in the book.

**Task 2**

This task was to break down the matrix into chunks which was made easy by the SUBMATRIX routine provided. These blocks were then iterated over using three for loops similar to the method of naïve matrix multiplication and then Task 1’s function was called to perform the block matrix multiplications at the innermost level of the loops. This method performs better than Task 1 because of cache locality.

**Task 3**

This task was to take Task 2 and add an OpenMP #pragma statement which would parallelize the loops and take advantage of the 20 cores each ada node has available to it. The trick here was determining which variables are private (indices I, J, K) and shared (matrices A, B, C).

**Performance / Output**

The fastest method overall was method 3, the parallel block matrix multiplication algorithm. This algorithm performed the multiplication in .393 seconds at 43.7 Gflops for a block size of 64 (peak performance point). This gave a speedup of 11.48 over the naïve sequential multiplication algorithm and a speedup of 13.49 over the block sequential multiplication algorithm.

Output given on next page.

