# Homework 3 CS 5220

Lara Backer, Greg Granito, Sam Tung

November 10, 2015

## 1 Introduction

# 2 Base OpenMP Code

### 2.1 Profiling

A look into performance of the code was conducted by using Intel's VTUNE on Totient. Due to some technical issues with the cluster, we could not run the "advanced-hotspots" for profiling.

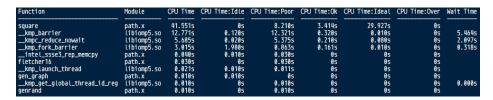


Figure 1: Most time consuming functions in the base code.

From looking at VTUNE, it appears that the most time spent in the program was done on the square function. This is not surprising, as there are plenty of nested for loops in the code, which undoubtedly takes a while to run through.

### 2.2 Scaling

# 3 OpenMPI

#### 3.1 Profiling

After attempting to implement OpenMPI, we once again look into performance of the code through profiling.

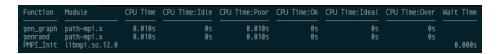


Figure 2: Most time consuming functions in the OpenMPI implementation.

## 3.2 Scaling

# 4 Processor Offloading

# 5 Additional Changes

For the future, there are many different things we can try. As evidenced in the past, proper usage of compiler flags should be able to provide some speedups for the process. Additionally, another option we plan to try is blocking. On a conceptual level, for each block, the fastest paths would be calculated given each potential entry and exit point, which would be the ghost cells around each block. From there, one aggregated faster path through would be calculated from the combination of smaller paths.

## 6 Overview

# References