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.

Function	Module	CPU Time	CPU Time:Idle	CPU Time:Poor	CPU Time:Ok	CPU Time:Ideal	CPU Time:Over	Wait Time
square	path.x	41.551s	0s	8.210s	3.414s	29.927s	0s	
__kmp_barrier	libiomp5.so	12.771s	0.120s	12.321s	0.320s	0.010s	0s	5.464s
__kmpc_reduce_nowait	libiomp5.so	5.685s	0.020s	5.375s	0.210s	0.080s	0s	2.097s
__kmp_fork_barrier	libiomp5.so	3.015s	1.980s	0.863s	0.161s	0.010s	0s	0.318s
__intel_ssse3_rep_memcpy	path.x	0.040s	0.010s	0.030s	0s	0s	0s	
fletcher16	path.x	0.030s	0s	0.030s	0s	0s	0s	
__kmp_launch_thread	libiomp5.so	0.021s	0.010s	0.011s	0s	0s	0s	
gen_graph	path.x	0.010s	0.010s	0s	0s	0s	0s	
__kmp_get_global_thread_id_reg	libiomp5.so	0.010s	0s	0.010s	0s	0s	0s	0.000s
genrand	path.x	0.010s	0s	0.010s	0s	0s	0s	

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.

Function	Module	CPU Time	CPU Time:Idle	CPU Time:Poor	CPU Time:Ok	CPU Time:Ideal	CPU Time:Over	Wait Time
gen_graph	path-mpi.x	0.010s	0s	0.010s	0s	0s	0s	
genrand	path-mpi.x	0.010s	0s	0.010s	0s	0s	0s	
PMPI_Init	libmpi.so.12.0							0.000s

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