Naman Gangwani
nkg160030
CS 4301.001
February 25, 2018

Report for Pgm2

Problem Statement

Matrix multiplication can be quite resource intensive, especially when dealing with large matrices. Making use of multiple cores can help improve its performance. One way to specifically do this is by utilizing the OpenMP API for C. For this task, OpenMP directives will be useful in improving upon an existing matrix multiplication function.

Approach to Solution

The OpenMP API is the main tool utilized to assist in finding a solution to the problem. Additionally, SLURM (Simple Linux Utility for Resource Management) and access to a Linux server on a supercomputer through an SSH may be essential. Since it is apparent that the matrix multiplication is heavily reliant on loops, the decision to use a pragma directive (from the OpenMP API) that specialized on parallelizing loops was made. Additionally, privatizing the variables within the nested loops and making use of the reduction clause for OpenMP allowed for more optimizations of the matrix multiplication function.

Solution Description

In this task, the code written by Dr. Goodrum was modified merely with the OpenMP API, and any additional code added to it was mainly done on the matmul function along with comments added all throughout the C source file. Since Pgm1 also involved the usage of OpenMP, the *matmul.c* file compiled for Pgm1 was also compiled for this task, except a minor change was made where the *matmul* function declaration was made to be at the top of the source code while its definition was at the bottom. This was mainly done to improve readability.

When logged into a remotely hosted Linux server on a supercomputer through an SSH (with Stampede2 being the recommended remote host), it is important to have the *matmul.c* and *matmul.sh* files in the current directory. When they are there, assure that the current node is a login node, because the *sbatch* command can only work from login nodes. From there, executing the bash script with the SLURM-specific command *sbatch* should be done with the command *sbatch matmul.sh*. A file beginning with the prefix *matmul.o* followed by the batch job number should be produced in the current directory from executing this bash script, and the content of that file will display the statistics of the program that ran.

```
login1.stampede2(221)$ ls
matmul.c  matmul.sh
login1.stampede2(222)$ sbatch matmul.sh

------------------------------------------------------------
         Welcome to the Stampede 2 Supercomputer
------------------------------------------------------------

No reservation for this job
--> Verifying valid submit host (login1)...OK
--> Verifying valid jobname...OK
--> Enforcing max jobs per user...OK
--> Verifying availability of your home dir (/home1/05437/tg847367)...OK
--> Verifying availability of your work dir (/work/05437/tg847367/stampede2)...OK
--> Verifying availability of your scratch dir (/scratch/05437/tg847367)...OK
--> Verifying valid ssh keys...OK
--> Verifying access to desired queue (normal)...OK
--> Verifying job request is within current queue limits...OK
--> Checking available allocation (TG-CIE180001)...OK
Submitted batch job 832620
login1.stampede2(223)$ ls
a.out  matmul.c  matmul.e832620  matmul.o832620  matmul.sh
login1.stampede2(224)$ cat matmul.o832620
/home1/05437/tg847367
Sun Feb 25 17:47:44 CST 2018
L = 1000, M = 1000, N = 1000, elapsed = 2.6563, flops = 7.52928e+08
```

The execution of the bash script and the results after multiplying two 1000x1000 matrices on a login node in Stampede2.

With these statistics, it is made apparent that the optimizations made to the matrix multiplication function worked measurably, however, it did not work as well as the Bridges supercomputer from Pgm1. Still, there is a significant difference from using OpenMP on Stampede2 than regularly calling to this matrix multiplication function on a local machine. What would normally take roughly ten seconds to compute the result of multiplying two 1000 by 1000 sized matrices now takes approximately two and a half seconds, signifying great improvement and success in resolving the problem with a fourth of the time used.