CS546 Parallel and Distributed Processing

Program Assignment 4

Submission:
Due by 11:59pm of 11/05/2019
Total points 100 - Late penalty: 10% penalty for each day late
Please upload your assignment on Blackboard with the following name:
CS546_SectionNumber_LastName_FirstName_PA4.pdf.
Please do NOT email your assignment to the instructor and/or TA!

NOTE: for all the below assignment use the example from the lecture resource https://github.com/beiwang2003/codas_perftools.

When an application encounters some performance issues, we need to identify the segment of the code that causes the problem and resolve and optimize what really matters. Profilers are the tool used in practice to identify the source of performance issues. In this assignment, we'll use Linux perf tools to profile a simple C++ application. Perf is a Linux performance analyzing tool available since the Linux 2.6 version. The perf profiler uses hardware counters to profile the performance of a running application. The results of perf are precise and fast because it does not do instrumentation on the code.

Installation
First, if it is not already done, you need to install the perf tools on your computer. On Ubuntu, you can just use apt-get to install it:

```
sudo apt-get install linux-tools
```

On the other systems, just use your favorite package manager to install the perf tools.

Exercise 1: (15 points)
Please list all available CPU counters with the *perf list* command.

Exercise 2: (35 points)
Please use the *perf stat* command instrument and summarize selected CPU counters:
1 Compile the code: (the naive approach of performing matrix-matrix multiplication with i,j,k order)

```
g++ -g -fno-omit-frame-pointer -O3 -DNAIVE matmul_2D.cpp -o mm_naive.out
```

2 Run perf stat:

```
perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-
stores ./mm_naive.out 500
```

3 Record the numbers for each event.

4 Compile the code: (the interchange approach of performing matrix-matrix multiplication with i,k,j order)

```
g++ -g -fno-omit-frame-pointer -O3 -DINTERCHANGE matmul_2D.cpp -o mm_interchange.out
```

5 Run perf stat:

```
perf stat -e cpu-cycles,instructions,L1-dcache-loads,L1-dcache-load-misses,L1-dcache-
stores ./mm_interchange.out 500
```

6 Compare the numbers for both cases.

The number of CPU cycles should much lower for interchange version, please list the CPU counts for each version, analyze the reason based on the code and CPU counters you get.

Exercise 3: (35 points)
What will happen if the matrix dimension is changed to 1000? Please analyze the performance difference between Naive and Interchange.

Exercise 4: (15 points)
Please find the hotspot using perf.

```
perf record -g ./mm_interchange.out 500
```

Then, to see the list of the costliest functions, you just have to use perf report:

```
Perf report
```

Please display a list of the costliest functions ordered by cost.

Grading:

Report: 100 points **(pdf only)**

- Readability, use of English, organization
- Clarity of plots / figures / screenshot
- Performance evaluation and analysis of results

If you have any problems about setting up perf on your own machine, please try to use Fusion. You can search the key words "Your Fusion system account is created" in your email inbox to get your user name and password. And you can check the document about Fusion from http://fusion2.cs.iit.edu/# . Please contact the Fusion team at support-fusion-group@iit.edu if you have any questions.

Note: We encourage collaboration between you and your classmates. Discuss various approaches and techniques to better understand the questions. However, we do NOT allow copying solutions or code. This is considered as cheating and falls under IIT code of honor. Penalties will be enforced. Please make sure you write your own solutions. GOOD LUCK!