**CSE 332 Project 3 Write up**

\* Note: The last 3 questions require you to write code, collect data, and produce graphs of your results together with relatively long answers. Do not wait until the last minute to start this write up!

1. **Who is in your group (Your name, UW NetID)?**

Reggie Jones 1027248

Tristan Riddell …

1. **What assistance did you receive on this project? Include anyone or anything *except* your partner, the course staff, and the printed textbook.**

Java Docs

Stack overflow

Fork Join Framework notes by Dan Grossman

1. **a) How long did the project take?**

-Reggie:

-Tristan:

**b) Which parts were most difficult?**

**c) How could the project be better?**

1. **(OPTIONAL) What "above and beyond" projects did you implement? What was interesting or difficult about them? Describe in detail how you implemented them.**
2. **a) How did you test your program? What parts did you test in isolation and how?**

**c) What smaller inputs did you create so that you could check your answers?**

**d) What boundary cases did you consider?**

1. **For finding the corners of the United States and for the first grid-building step, you implemented parallel algorithms using Java's ForkJoin Framework. The code should have a sequential cut-off that can be varied. Perform experiments to determine the optimal value of this sequential cut-off.**

**1) Sequential vs. parallel versions of corner finding. Looking at V1 and V2, vary the cutoff for V2.**

**2) Cut-off in the grid-building step. Looking at V3 and V4, vary grid-building cutoff for V4.**

**3) Cut-off in the grid-merging step. Looking at V3 and V4, vary grid-merging cutoff for V4.**

\* You don't have to worry about finding the optimal combination of cut-offs.

**1.**

**a.** Machine characteristics: With each new sequential cut off value, runs 1000 warm up tests followed by 5000 real tests. The time for that cut off is the average of the 5000 tests. The code simply calls ParallelCorners, our v1 and v2 parallel class. When sequential cut off is equal to data size, the code is strictly sequential.

I collected timing information using System.currentTimeMillis().

No details needed to replicate experiment, merely rerun V2SequentialTest.java.

**c.** I expected to see time efficiency peak with a lower cut off, around the 1000 to 10000 range. However, according to my data, time is nearly constant as long as there are at least two threads. For any cut off that calls multiple threads, the time is about ~1.3. But when the code is run FULLY sequential, the time is over 5 milliseconds which is almost four times as much as even 1000 less than the data size. This did not entirely match my expectations. Although I believe the reason it turned out this way is the data set is too small to really see a difference with a changing cut off. I imagine if we performed the same test on a data file with 100 million data points, we would find an optimal cut off much easier.

I believe that with a data set this small, the efficiency effectively maxes at 2 processors. Which means as long as there are two threads, the performance will be good. This would explain why time is constant for all cut off, with an enormous outlier when it was run fully sequentially.

**a) Describe your experimental setup:**

**1) Your machine characteristics**

**2) How you collected timing information**

**3) Any details that would be needed to replicate your experiments**

**b) Experimental Results: Place your graph for experiment 1), 2) and 3).**

Clearly label which line is for which version in each of your plot.

1) Cutoff vs. Runtime for V1 and V2

2) Cutoff vs. Runtime for V3 and V4: Grid building

3) Cutoff vs. Runtime for V3 and V4: Grid merging

**c) Interpretation of Experimental Results**

Note that if the sequential cut-off is high enough to eliminate all parallelism, then you should see

performance close to the sequential algorithms, but evaluate this claim empirically

(and then answer the question - is this what you see?). For each of the experiments 1), 2) and 3),

answer the following questions.

**1) What did you expect about the result and why?**

**2) Did your result agree with your expectation?**

**3) If the result did not match with your expectation, why do you think it happened?**

**4) Draw a conclusion from the experimental result.**

1. **Compare the performance of V4 to V5 as the size of the grid changes.**

Clearly label which line is for V4 and which line is for V5 in your plot.

**a)** **Experimental Results: Place your graph for Grid size vs. Runtime for V4 and V5 here.**

**b) Intuitively, which version is better for small grids and which version is better for large grids?**

**c) Does the experimental data validate your hypothesis in b)? If the result did not match with your**

**expectation, why do you think it happened?**

1. **Compare the performance of V1 to V3 and V2 to V4 as the number of queries changes. That is, how many queries are necessary before the pre-processing is worth it?**

**Clearly label which line is for which version in each of your plot.** **Note you should time the actual code answering the query, not including the time for entering the query.**

**a) Experimental Results (Place your graph here).**

1) Number of Query vs. Runtime for V1 and V3

2) Number of Query vs. Runtime for V2 and V4

**b) Interpretation of Experimental Results**

For each of the experiments 1) and 2), answer the following questions:

**1) What did you expect about the result and why?**

**2) Did your result agree with your expectation?**

**3) If the result did not match with your expectation, why do you think it happened?**

**c) According to your experiment, how many queries are necessary before the pre-processing**

**is worth it?**

1. **If you worked with a partner:**

**a) Describe the process you used for developing and testing your code. If you divided it, describe**

**that. If you did everything together, describe the actual process used (eg. how long you talked**

**about what, what order you wrote and tested, and how long it took).**

**b) Describe each group member's contributions/responsibilities in the project.**

Reggie: -v1

-refactoring v1 to work with GUI

-cmd line arg processing & user query input processing

Tristan: -

**c) Describe at least one good thing and one bad thing about the process of working together.**

One good thing is we are both pretty rad so it was easy to work together. One bad thing is impact of our busy schedules / workloads from other loads usually don’t sync up so we are generally working on it at different times.

**Appendix**

Place anything else that you want to add here.