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)?**
   Ruijia Wang: ruijiw
   Mengyuan Huang: mengyh2

2. **What assistance did you receive on this project? Include anyone or anything *except* your partner, the course staff, and the printed textbook.**
   Stackoverflow.com, TA office hour.

3. **a) How long did the project take?**
   Phase A: 1 day
   Phase B: 2.5 days
   b) Which parts were most difficult?
   Debug and Version5.
   c) How could the project be better?
   We should divide our work by functionality rather than divide work by versions. That might save us more time.

4. **(OPTIONAL) What "above and beyond" projects did you implement? What was interesting or difficult about them? Describe in detail how you implemented them.**

5. **a) How did you test your program? What parts did you test in isolation and how?**
   We test each method in our version classes separately. To test findCorners, we use toString method in the Rectangle class to print out the result and compare the rectangle with correct output. As for the calculateQuery method, we test it by calculating population according to different arguments and user input. Then we compare the results with sample output.

   c) What smaller inputs did you create so that you could check your answers?
   We use the first ten lines in the CenPop2010.txt as our own sample input.

   d) What boundary cases did you consider?
   We test the situation when the query rectangle is as large as U.S. rectangle, and we also test the boundary case when the query is only one grid.

6. **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.

a) Describe your experimental setup:

1) Your machine characteristics

MS Windows 7, Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz

2) How you collected timing information

We create new class to collect time of different experiments. In order to have more accurate and stable result, we use average time for each experiment. We throw away several runs at the beginning to exclude the JVM warmup effect.

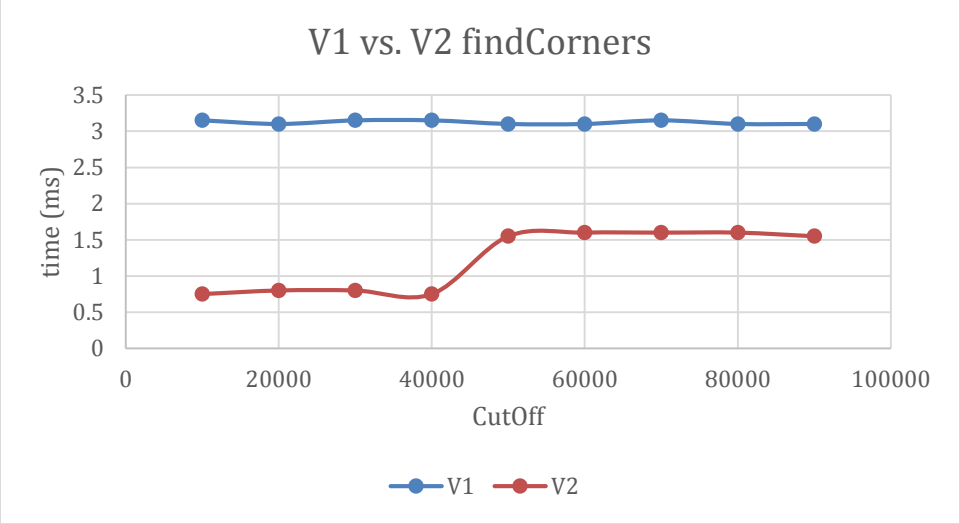
3) Any details that would be needed to replicate your experiments

We use the same column and row in the run configurations argument and user input for each experiment pairs so that time is the only variable to have effect on the experiment.

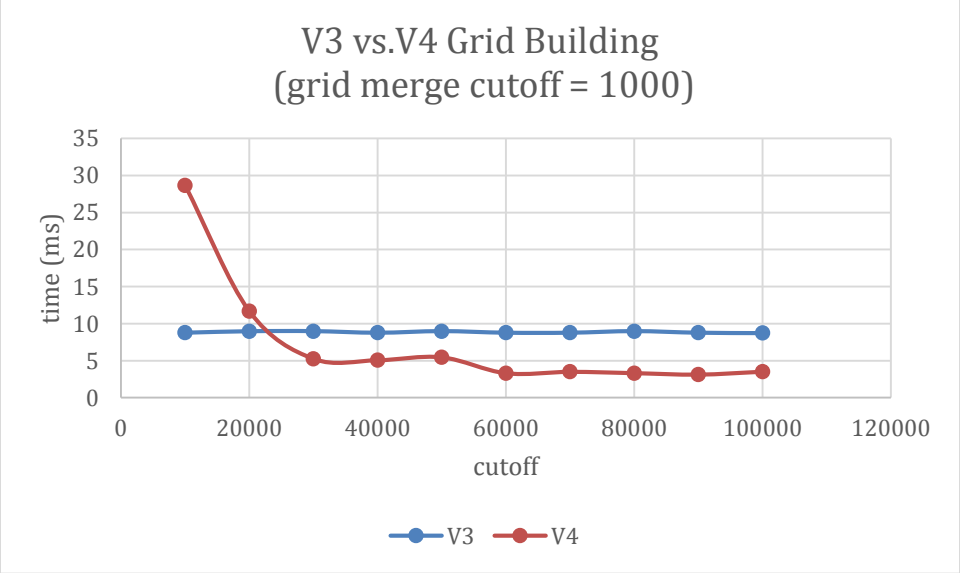
b) Experimental Results: Place your graph for experiment 1), 2) and 3).

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

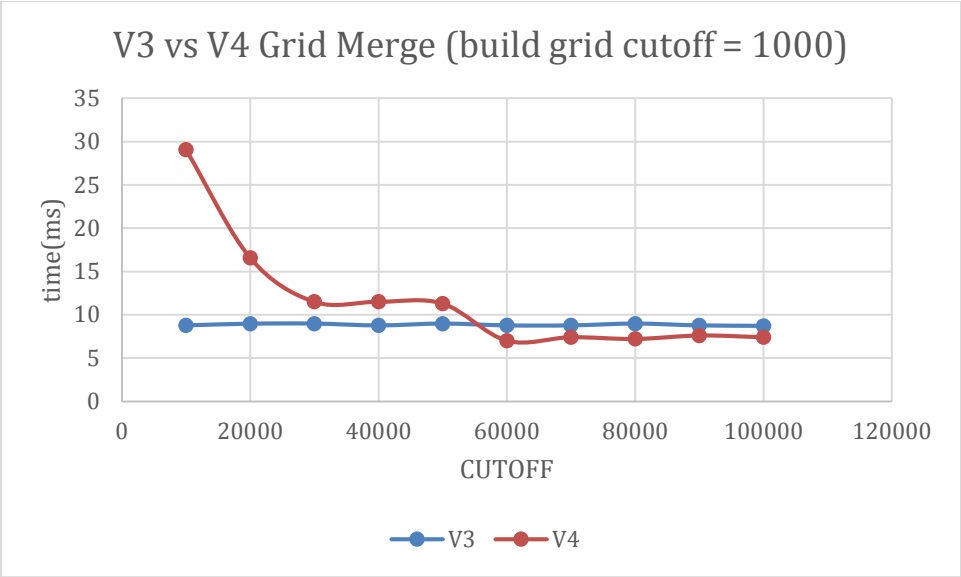
1) Cutoff vs. Runtime for V1 and V2



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?

Experiment 1: We expect the parallelism part has better performance than the sequential version, because more threads save more time.
Experiment 2: We expect Version4 would be quicker than Version3 on build grid because Version4 using parallelism.
Experiment 3: same as Experiment 2.

For all the experiments, we expect the result of parallelism is getting closer to sequential result when the cutoff is getting greater, because most part of parallelism version will process data sequentially when cutoff is big.

2) Did your result agree with your expectation?

Experiment 1: The result agrees with our expectation. But the graph didn't show the result of V1 and V2 getting closer when cutoff is getting greater.
Experiment 2: The result partly corresponds to our expectation. At first V4 is slower than V3, but when the cutoff increase, V4 is quicker than v3.
Experiment 3: The result partly corresponds to our expectation. At first V4 is slower than V3 and along the cutoff increase V4 is getting quicker.

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

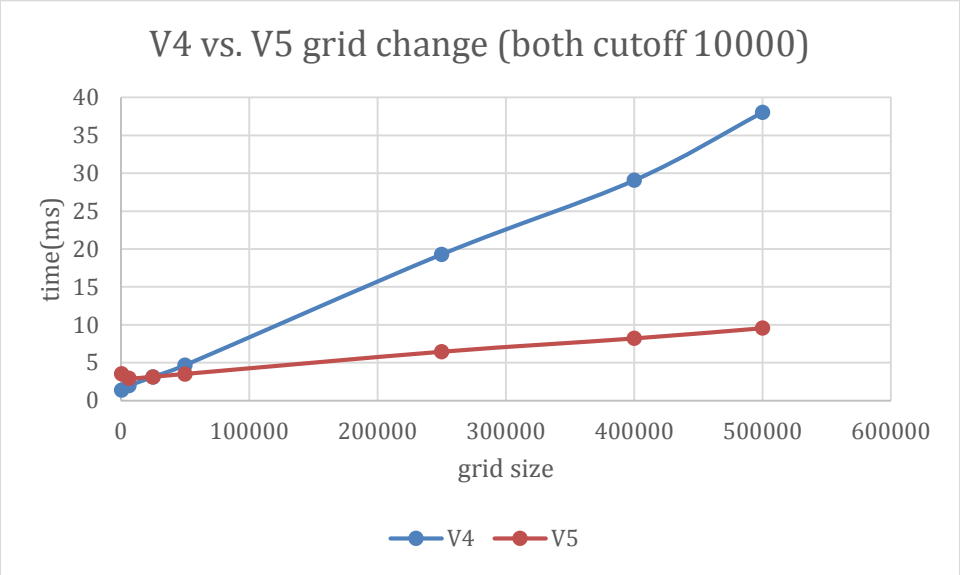
Experiment 2 & 3: For small n, recursive threads tend to cost more than doing sequential process.
Experiment 1: We probably need bigger range of cutoff to determine if the lines of V1 and V2 get closer when cutoff is greater.

4) Draw a conclusion from the experimental result.

From the above three experiments, we think that parallelism doesn't always have better performance than sequential process, because for small number of data there is a tradeoff to have more recursive threads. Also, we think when the cutoff is getting greater, the parallelism result tends to be closer to sequential result.

7. **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.



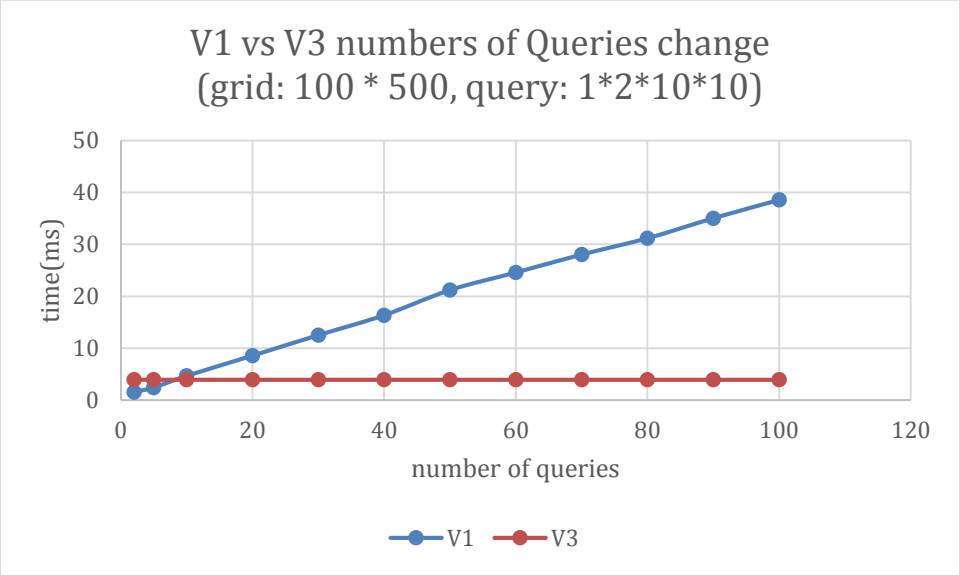
b) Intuitively, which version is better for small grids and which version is better for large grids?
We expected that with small grid size V4 has better performance than V5. When grid size keep increasing, V5 performance quicker than V4.

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?
Yes, the experimental data validate our hypothesis in b).

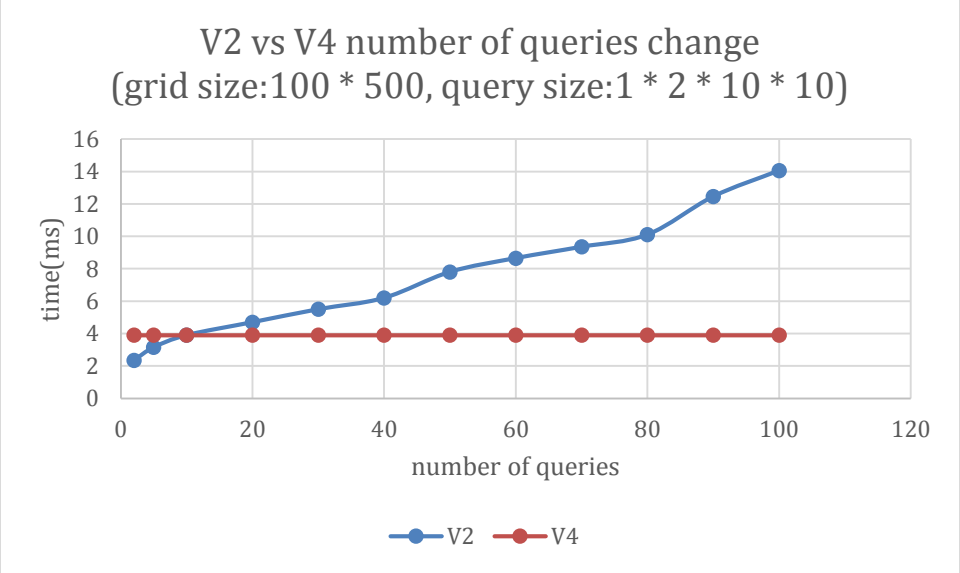
8. **Compare the performance of V1 to V3 and V2 to V4 as the number of queries changes. That is, how many queries are necessary before the pre-processing is worth it?**
Clearly label which line is for which version in each of your plot. Note you should time the actual code answering the query, not including 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?

We expect V3 and V4 have constant runtime no matter how many queries the program called, because it only take O(1) to calculate the query by using grid we build in V3 and V4.

2) Did your result agree with your expectation?

Yes, it agree with our 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?

We think around 10 queries are necessary before the preprocessing is worth it.

9. **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).

We divided the work by different versions. One of us did verison1 and 3 and most write-up work. The other person did verison2, 4 and 5. We write our test code separately. We developed our code by the order of versions.

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

Mengyuan Huang: Version1 and 3 and test code. Calculator.java. (abstract class for version structure) and the main method in PopulationQuery.java.

Ruijia Wang: Version2, 4 and 5 with test code. Common.java(for store common code in all the version). We did the readme.doc together.

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

Good thing: less workload for each person.

Bad thing: since some versions are based on previous ones. We cannot do our work concurrently.

Appendix

Place anything else that you want to add here.