High Performance Computing

Homework #5

Due: Tuesday March 17 2015 by 11:59 PM (Noon)

Email-based help Cutoff: 5:00 PM on Mon, March 16 2015

Name: Yan Yu

Experimental Platform

The experiments documented in this report were conducted on the following platform:

Component	*Details*
CPU Model	Intel(R) Xeon(R)
CPU/Core Speed	2.67GHz
Main Memory (RAM) size	24725392kB
Operating system used	Linux 2.6.32-279.14.1.e16.x86_64
Interconnect type & speed (if applicable)	
Was machine dedicated to task (yes/no)	Yes
Name and version of C compiler (if used)	Icc 4.9.0
Name and version of Java compiler (if used)	Javac 1.7.0_13
Name and version of other non-standard software tools & components (if used)	

Java Runtime Observations

Document the statistics collated from your experiments conducted as instructed for Part 1in the table below. In Table 1 report just mean and 95% CI values (not the five raw timing values) for the Java version of matrix multiplication.

MATRIX_SIZE	Samples	Java Execution Time (*Avg*±*CI* sec)	Peak Memory (KB)
500	5	2.758±0.0236	118289
750	5	8.592±0.124	128587.5
1000	5	19.434±0.532	152496.7
1500	5	1:14.93±1.213	210714.4
2000	5	4:13.94±3.977	306586.8
2500	5	6:56.59±9.87	421963.6
3000	5	12:39.875±61.73	626591

Table 1: Execution timings for simple matrix multiplication implemented in Java. The average execution times and 95% confidence intervals (CI) have been computed from five independent runs of the test program MatMul.java.

C++ Runtime Observations

Document the statistics collated from your experiments conducted as instructed for Part 1in the table below. In Table 2 report just mean and 95% CI values (not the five raw timing values) for the C/C++ version of matrix multiplication.

MATRIX_SIZE	Samples	C++ Execution Time (*Avg±CI* sec)	Peak Memory (KB)
500	5	0.254 ± 0.05381	16539
750	5	1.516 ± 0.00712	31304.4
1000	5	2.516 ± 0.5096	51810
1500	5	10.358 ± 0.5274	110408.6
2000	5	22.402 ± 1.058	192431
2500	5	49.497 ± 3.414	297892
3000	5	1:15.987 ± 1.056	426791

Table 2: Execution timings for simple matrix multiplication implemented in C. The average execution times and 95% confidence intervals (CI) have been computed from five independent runs of the test program MatMul.c.

Using the data in the column titled `Execution Time` plot the graph comparing the Java and C/C++ version of matrix multiplication using Excel and copy-paste the chart into this report replacing the chart already shown. You may use the chart below as a graphical template for developing your chart.

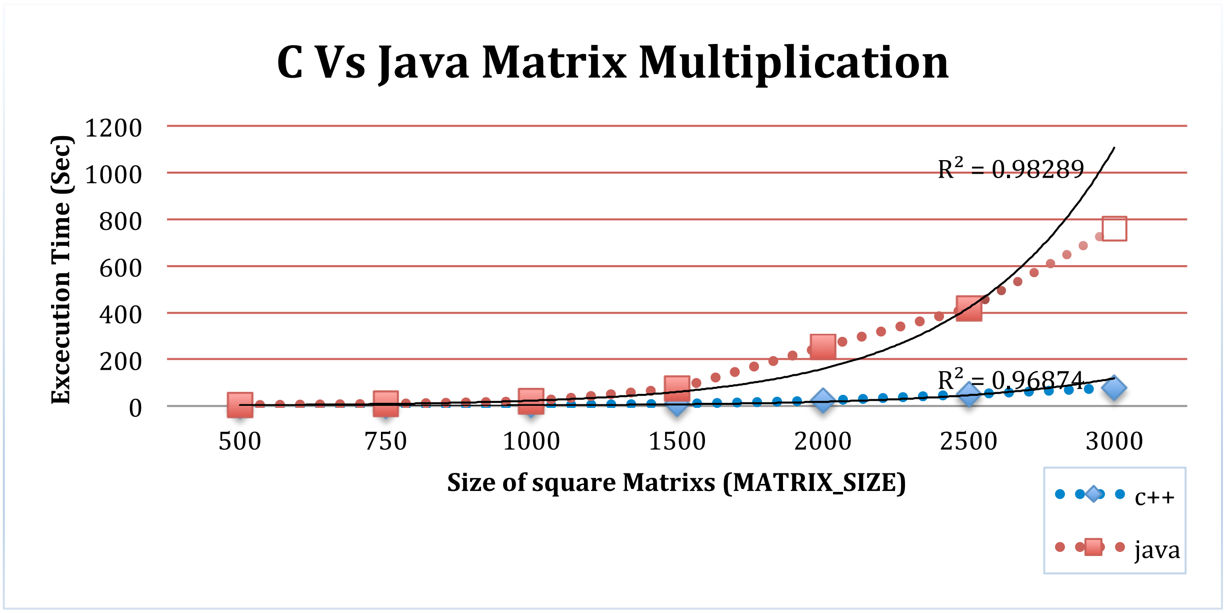

Figure 1: Plot comparing execution timings of matrix multiplication implemented in Java and C++

Using the data from the above chart indicate the following information for the C++ and Java runtimes:

Time complexity for Java version: (This is the equation for the trend line)	$O(1.2496e^{0.958n})$
Time complexity for C++ version: (This is the equation for the trend line)	$O(0.1591e^{0.9386n})$

Inferences & Discussions

Now, using the data from tables along with the graphs document your inferences contrasting matrix multiplication implemented in C++ and Java using the evidence gathered through the experiments (use as much space as you need). Compare and contrast on the performance trends and memory usage in the two languages. Compare and contrast source code differences between the source codes in the two languages In addition comment on which programming language would be more suitable for energy-efficient computing for mobile devices (smartphones, tablets, laptops) and discuss their advantages. Here is a classic one that I get almost every year (so this year you cannot use it in your discussions, sorry) – "*Energy efficient devices let students watch way more NetFlix/Hulu while lurking on Facebook/Twitter and hurt their grades! Consequently Python the slowest of the lot would be best for Miami students who belong to the Slytherin and are excellent at Parseltongue (or as we call it Python) even though Dr. Rao tries to convert students to the good side in the hopes of helping them get better jobs*." However, you are welcome to be sarcastic, but please enclose sarcastic comments between `<sarcasm>` `</sarcasm>` tags.

As seen from the tables, Java version is way slower than C++ version of matrices multiplication, which is about 90% slower. And also the memory usage of Java version is also larger than C++ version. The reason is because when compiling the program, C++ code is directly compiled to the native machine code, and Java is compiled into virtual machine code which will be executed by the Java virtual machine, and then Java virtual machine will handle the translation between the virtual machine code and actual machine code, which results in the performance decrease. Another reason is that the compiler for C++ also optimized the code so that the program can effectively utilize the hardware architecture to boost the performance, but Java compiler didn't do this level of optimization, it is the job of virtual machine.