High Performance Computing

Homework #6

Due: Tuesday March 31 2015 by 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Mon, Mar 30 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)	

Performance Analysis

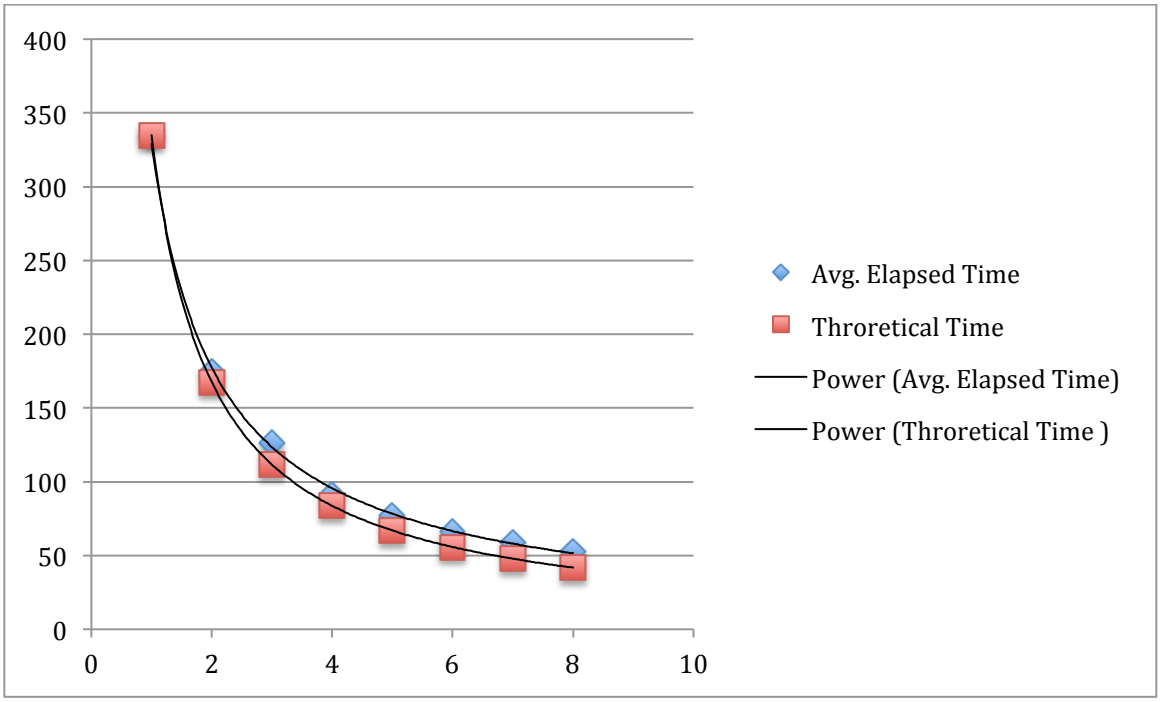
I defined a custom struct to store the number being checked and four bool variables to indicate its four properties. And beside that, I also defined a class which groups four static functions to check the four properties of the num. In the main function, the for loop to check all the numbers were wrapped inside a OpenMP block in order to use multi-thread to speed up the process.

Document the statistics collated from your experiments conducted in the table below. Delete the first row with fictitious data included just to illustrate an example.

#Threads	Timings from 5 different runs (Separated by commas)			Average Elapsed Time	95% CI
	User Time	Elapsed Time	%CPU		
1	315.85, 313.67, 314.80, 313.50, 315. 42	342.31, 326.42, 339.53, 323.67, 342.07	97%, 97%, 98%, 97%, 97%	334.8	7.695630533
2	324.39, 324.53, 327.26, 327.26, 326.70	173.79, 175.32, 175.65, 175.65, 173.47	186%, 185%, 186%, 186, 186%	174.776	0.904683747
3	346.53, 373.02, 319.70, 318.00, 318.31	128.91, 153.56, 116.93, 116.73, 116.08	268%, 242%, 273%, 272%, 274%	126.442	13.70893626
4	325.66, 326.65, 321.27, 321.56, 319.98	91.92, 92.22, 90.54, 90.58, 90.13	354%, 354%, 354%, 355%, 355%,	91.078	0.7919202
5	331.99, 332.37, 333.31, 327.11, 327.64	77.08, 76.91, 79.34, 76.45, 76.74	430%, 432%, 420%, 427%, 426%	77.304	0.990691561
6	333.51, 334.16, 333.57, 335.20, 334.11	66.04, 66.56, 66.03, 66.53, 66.22	505%, 502%, 505%, 503%, 504%	66.276	0.219303949
7	335.11, 334.22, 335.50, 335.46, 334.13	58.96, 58.76, 58.69, 58.80, 59.19	568%, 568%, 571%, 570%, 564%	58.88	0.1702399

8	334.69, 334.83, 334.75, 334.40, 334.92	52.68, 53.08, 53.04, 52.62, 52.85	635%, 630%, 631%, 635%, 633%	52.854	0.176181132

Using the above data plot a chart (using Excel) with number of threads on X-axis and theoretical expected runtime (computed as *Single Threaded Runtime ÷ Number Of Threads*), observed average elapsed time on Y-axis replacing chart below. Copy-paste the chart you have generated replacing the chart shown below (the chart shown below is fictitious and your chart will look different. **However, ensure you include axis titles, legend, and data-point labels in your chart!**).



Using the above chart develop a report (**at least 10 sentences are needed to earn 10 points!**) discussing the following performance aspects (use as much space as needed):

- Discuss the changes needed to the program to accomplish multi-threading
- Discuss the change in user time versus elapsed time
- Are the observed average runtime and expected theoretical runtime the same?
- Do they follow a similar trend?
- What could plausibly explain any difference between the theoretical and observed behaviors of your program?

In order to use multi-thread to solve the problem, we need to manfully divide the data into different blocks so that each thread can work on that part of data. As the thread number increase, the elapsed time decreases, the elapsed time is about single thread time * / (num of thread). Both of the theoretical time and avg. elapsed time follow a similar trend.