**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) |  |

## Block Matrix Multiplication Observations

Document the statistics collated from your experiments conducted do measure the runtime of block matrix multiplication version of the program. In Table 1 report just mean and 95% CI values (not the five raw timing values) for the block matrix multiplication.

|  |  |  |  |
| --- | --- | --- | --- |
| **MATRIX\_SIZE** | **Samples** | **Block Execution Time**  **(*Avg*±*CI* sec)** | **Peak Memory (KB)** |
| 500 | 5 | 0.33 ± 0 | 17091.2 ± 6.10214006787783 |
| 750 | 5 | 0.94 ± 0.0313338727896824 | 31795.2 ± 6.10214006787783 |
| 1000 | 5 | 1.944 ± 0.0513807576744446 | 52480 ± 9.6483306079342 |
| 1500 | 5 | 6.246 ± 0.051025668708994 | 111318.4 ± 7.47356475264649 |
| 2000 | 5 | 14.02 ± 0.36161133504358 | 193584 ± 0 |
| 2500 | 5 | 27.742 ± 0.518927223333676 | 299276.8 ± 6.10214006787783 |
| 3000 | 5 | 46.634 ± 1.18197587762323 | 428444.8 ± 6.10214006787783 |

Table : Execution timings for block 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 BlockMatMul.cpp.

## Regular Matrix Multiplication Runtime Observations

Document the statistics collated from your experiments conducted to measure runtime of regular matrix multiplication (**Note: you can reuse data from previous part**). In Table 2 report just mean and 95% CI values (not the five raw timing values) for the C/C++ version of matrix multiplication implemented in MatMul.cpp.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 : 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.cpp.

Using the data in the column titled Execution Time plot the graph comparing the regular and block-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. However, ensure you include trendline for both impelemntations.

Figure : Plot comparing execution timings of regular and block matrix multiplication implemented in C++

Using the data from the above chart indicate the following information for the regular and block matrix multiplication runtimes:

|  |  |
| --- | --- |
| Time complexity for regular matrix multiplication:  (This is the equation for the trend line) | O(1E-09x2.777) |
|  |  |
| Time complexity for block matrix multiplication:  (This is the equation for the trend line) | O(1E-08x2.777) |

## Inferences

Now, using the data from tables along with the graphs document your inferences contrasting the two different forms of matrix multiplication implemented in C++. Compare and contrast on the performance trends and memory usage of the two versions. Discuss if the time complexity changed using the trend line equations.

|  |
| --- |
| As seen from the tables above, even through the trendline are about the same, the block matrices multiplication has about 40% performance increase.  The reason why block matrices multiplication is faster than regular matrices multiplication is that by using blocking, the program efficiently makes use of caching, so the accessing time to element of matrices is faster, which results in the performance boost for block matrices multiplication. |