Lab 05: Memory Performance Analysis and First OpenMP

Assigned: 2019-02-19 08:00:00

Due: 2019-02-21 23:59:00

Instructions:

- Written portions of this assignment are submitted via Canvas. Unless specified otherwise, the written portion of the assignment is to be completed using LaTeX. All derivations, images, graphs, and tables are to be included in this document. Handwritten solutions will receive zero credit.
- Code portions of this assignment are submitted via `code.vt.edu`. Source code must be in the private repository, to which the CMDA 3634 instructors must have access.

Deliverables: For this assignment, you are to submit the following:

1. (Canvas) `<pid>_Lab_05.pdf`: A PDF file, rendered by `pdflatex` (the file generated by Overleaf is sufficient) containing the answers to the questions requiring written answers. Use the template provided in the project repository.
2. (`code.vt.edu`) The source files required to compile and run your solutions to the lab and the tex and image files for your report, in the appropriate directories.

Collaboration: This assignment is to be completed by yourself, however, you make seek assistance from your classmates. In your submission you must indicate from whom you received assistance.

Honor Code: By submitting this assignment, you acknowledge that you have adhered to the Virginia Tech Honor Code and attest to the following:

> I have neither given nor received unauthorized assistance on this assignment. The work I am presenting is ultimately my own.

Resources

- Memory hierarchy:
  - `https://en.wikipedia.org/wiki/Memory_hierarchy`
- Cachegrind:
  - `http://valgrind.org/docs/manual/cg-manual.html`
- Basics of prefetching:
  - `https://en.wikipedia.org/wiki/Cache_prefetching`
- OpenMP:
  - General tutorial `https://computing.llnl.gov/tutorials/openMP/`
  - Timing `https://gcc.gnu.org/onlinedocs/gcc-4.5.0/libgomp/omp_005fget_005fwtime.html`

Tasks

In this lab, you will investigate the performance of multiple implementations of a matrix-vector multiplication, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A}$ and $\boldsymbol{x}$ are filled with data already, and $\boldsymbol{b}$ is overwritten with the output. The fundamental structures used are VectorND (same as previous lab) and Matrix (has a data array stored in row-major order, has a number of rows, and has a number of columns).

Warning: I have indicated where you should be running commands in a terminal with the > character. This character is **not** part of the command!

1. **Setup** your coding environment.
   (a) Pull the lab materials from the upstream repository.
   (b) Examine the history to see what new files have been added by the instructors.
   (c) Push your local branch to the origin repository to add the new changes.
   (d) For the second part of this lab, you will be using OpenMP, which requires the `libomp-dev` package.

   ```
   > sudo apt install libomp-dev
   ```

2. **Experiment** with performance. Be sure to use git to commit your code regularly.
   (a) Use the program `matvecTiming.c` to time multiple runs of matrix-vector multiplication implemented with two different access orderings:
   - row-oriented access: looping first over the rows, then over the columns
   - column-oriented access: looping first over the columns, then over the rows

   We have provided a bash shell script to help you run the various experimental cases. You can use either of the following commands to execute the script.

   ```
   > ./time_matvec.sh
   ```

   ```
   > bash time_matvec.sh
   ```

   (b) For testing cache performance, we have separated the two versions of matrix-vector multiplication in `matvecTiming.c` into two programs: `matvecColOrient.c` and `matvecRowOrient.c`. We can now use cachegrind to examine the cache performance of each implementation. Compile these programs (using the provided make rules) and run the experiment scripts (`cache_row_orient.sh` and `cache_col_orient.sh`) to generate data to study the cache performance.

(c) Make sure you have 2 or more processors enabled in your virtual machine. If you don't, you may need to shut down your virtual machine, then go into settings (`settings->system->processors`) to add another. **Warning:** If you have an older laptop, this may cause some performance issues. If it does, we can revert the VM back to 1 available processor. If you have issues and cannot complete this step, speak to the instructor or GTA to obtain instructions for an alternative mechanism for completing this step.

(d) Copy `matvecRowOrient.c` to a new file named `matvecOMP.c`. Be sure to tell git to track this file (add it, then commit it). Make the following changes:

- Include `omp.h` in the program
- Change the clock timing to use the `omp_get_wtime()` function. The return value of this function is of type `double` and is in seconds, not clock cycles.
- Change the line that previously called `matvec_row_oriented` so that it now calls the function `matvec_row_oriented_omp`
- Modify the function `matvec_row_oriented_omp` in Matrix.c to have an openmp for-loop that breaks up the matrix into subsets made up of groups of rows
- Build the program using the `matVecOMP` make rule and run the multithreaded timing test script. **Note:** The `-fopenmp` flag has already been included in all the make rules so that they won't break when you add the openmp include statements.

3. **Answer** the questions listed below. The following workflow uses Overleaf, but you are welcome to use a local tex installation if you prefer.

(a) In Overleaf, create a new empty project, and copy in `lab05_report.tex` and `lab05_rhewett_style.tex`.

(b) Answer the questions, and upload data files where necessary.

4. **Submit** your results.

(a) After you have completed this lab (which we'll continue in class on Wednesday), upload a PDF of your report to Canvas.

(b) Don't forget to clean up before submission (do not commit your cachegrind.###.out files).

(c) Push your source code and latex files to `code.vt.edu`. From anywhere in your local projects repository

```
> git push origin master
```

(d) Examine your assignment repository on `code.vt.edu` to be sure that all of your materials have been correctly submitted.

Questions

Answer the following questions using the latex template provided in the `lab05/report` directory of the assignment repository.

1. Answer the following questions about your laptop computer. You will need to refer to Internet resources for this information. You must cite your sources. Note, the mechanisms to find the answers to these questions are different for Windows vs macOS. Find this information for your laptop, not for the VM.
   - What brand, model, and version is your processor?
   - What size are the L1, L2, L3 (if it exists) caches?
   - How many bytes per cache line?
   - How big is your main memory?
2. In the command line on your virtual machine, you can check information about the processor by checking the contents of the /proc/cpuinfo file:

   ```
   > cat /proc/cpuinfo
   ```

   Copy this information into a table. Does this match your laptop's processor?
3. In your own words, describe the difference between the two calculations being timed in `matvecColOrient.c` and `matvecRowOrient.c`. Before running any code, can you make any predictions about their relative performance for different sized matrices?
4. We will analyze square matrices for this exercise, although the code allows for rectangular matrix studies. Say that you were to perform a matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^N$.
   (a) Write a general formula for how many bytes of data you expect this to use (similar to the $16N$ expression used in the random-order axpy example in class).
   (b) For your system, at what value for N do you expect to exceed L1 cache?
   (c) For your system, at what value for N do you expect to move out of lower level (L2 or L2+L3) cache?
5. Using the experiment script `cache_row_orient.sh`, record the results for `matvecRowOrient.c` in the table below, to analyze the cache performance for this data access pattern.

nRows	**nCols**	**time (s)**	L1 read miss %	LL read miss %	L1 write miss %	LL write miss %
10	10					
20	20					
40	40					
80	80					
160	160					
320	320					
640	640					
1280	1280					
2560	2560					
5120	5120					

6. Using the experiment scripts `cache_col_orient.sh`, record the results for `matvecColOrient.c` in the table below, to analyze the cache performance for this data access pattern.

nRows	**nCols**	**time (s)**	L1 read miss %	LL read miss %	L1 write miss %	LL write miss %
10	10					
20	20					
40	40					
80	80					
160	160					
320	320					
640	640					
1280	1280					
2560	2560					
5120	5120					

7. Using your favorite plotting software (Excel, Python Matplotlib, Plot.ly, Matlab, etc.) plot the timing results for each of the above tables. You should plot N on the x-axis and the run-time on the y-axis. Both axes should use a logarithmic scale.

   Be sure to add any files/code used to produce the plot in your `lab05/report` folder. Your plot must have a legend indicating which series corresponds to which experiment.

8. In the previous plot, where are the noticeable changes in the timing trends? At the point(s) where the timing trends change, what is causing this change? Support your argument with your memory size predictions and your cachegrind data. You may need to take more samples. Be sure to include any additional code/files used to produce those figures in your `lab05/report` folder.

9. How many processors does your virtual machine have? How do you know? Include a screenshot if it's useful.

10. In the table below, record your timing results for the serial matrix-vector multiply (with row-major access) and for the openmp matrix-vector multiply (with row-major access). Also, record the ratio of the times.

    Did you get as much speedup as you expected based on the number of processors? Why do you think this happened?

nRows	**nCols**	**time (s) without omp**	**time (s) with omp**	**serial time / omp time**
10	10			
20	20			
40	40			
80	80			
160	160			
320	320			
640	640			
1280	1280			
2560	2560			
5120	5120			

11. Other than the instructor or TAs, who did you receive assistance from on this assignment?