Assignment# 1

CSE 530, Fall 2024

Note:

- Please include your name and PSUID on the first page.
- Submit all files on Canvas.
- The assignment must be submitted on Canvas before the due date (midnight).
- No single line answers are accepted in the submission.
- Refer to the syllabus for late submission policies.
- No kind of collaboration is allowed unless specifically mentioned in the assignment.
- All source materials must be cited. The University Academic Code of Conduct will be strictly enforced.*
- All queries related to Assignment should have a subject line **CSE530: Assignment#1**

DO NOT USE CHATGPT OR ANY OTHER SIMILAR TOOLS FOR THE COMPLETION OF THIS ASSIGNMENT!! ANY UTILIZATION OF SUCH RESOURCES TO COMPLETE THE ASSIGNMENT WOULD BE CONSIDERED AS AN ACT OF PLAGIARISM.

Goal: Analyze the cache system behavior for various memory-intensive kernels (working on 2D tensor).

Students are expected to develop the following memory kernels (based on example given below) and analyze the performance for different configurations of the cache architectures. You will need to summarize the observations and provide insights to any trends that you observe.

- **Scatter Operation**
  - Choose input tensor size of 128x128 and 150x150
  - Generate 1k indices, 1k data samples. Scatter those samples in the above matrices.
- **Gather Operation**
  - Choose input tensor size of 128x128 and 150x150
  - Generate 1k indices randomly, and gather the elements in those indices
- **Convolution Operation (not correlation operation)**
  - Choose any one of the following kernel sizes: [3x3, 5x5, 7x7] with a stride of 1. You can choose random values/Gaussion kernels, etc.
  - Input square matrix size: 128x128 and 150x150[reduce the size if you are facing any hardware issue, but please keep the first matrix dimension power of 2 and the second not, e.g. 64x64 and 80x80]
  - You are free to include, padding, and other modifications that make your job easier.

Cache system configuration knobs for different cache architectures:

- Cache hierarchy depth - It is used for creating multi-level cache hierarchy. [You can vary them from L1 to L3; 3 depths]
- Cache blocks - It is per cache and can be modified to have different cache size. [Choose either L1 or L2, and vary the number of blocks in those level]
- Cache associativity - To play with associativity of each cache level [Choose either L1 or L2, vary from 2-way to 8-way].

Note:

- Our kernel input size is 128x128 and assuming 'int' (4 bytes) as 2D matrix data type, we will get almost 128*128*4 = 65536 bytes ≈ 64 KB.
- If you keep your L3's cache size more than 46KB, you are likely not to get many misses. Please be aware of this fact and design your cache sizes appropriately to get interesting plots and observations.

Example:

Modern system has a hierarchical memory system. Initially input tensors are stored in the main memory (i.e., biggest, and slowest memory agent) and transferred to cache (i.e., smallest, and fastest memory agent) as per workload requirement. An example of such behavior is shown in Figure 1.

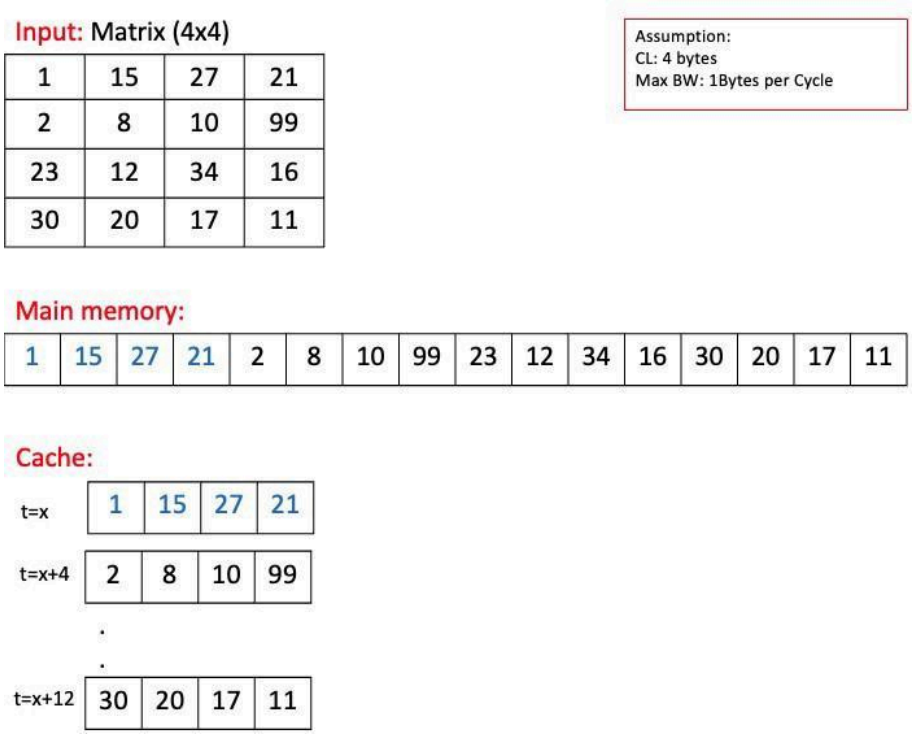

Figure 1: Matrix representation in memory system. (CL: cacheline size.)

Listing 1: Row wise memory copy kernel

```
//Here src[n][n] is input
tensor  //of size=nxn passed by
user int dst[n][n] = {};
parsec_roi_begin(); //Pintool will start recording memory
access for (auto row = 0; row != n; row++)
{
      for (auto col = 0; col != n; col++)
      {
            dst[row][col] = src[row][col];
      }
}
parsec_roi_end(); //Pintool will stop recording memory
access
```

Simulator Details:

Using Intel PIN and python-based cache simulator:

Students can use the intel pin tool to generate traces for various memory-intensive kernels (working on 2D tensor) and use the generated traces with the high-level cache simulator to analyze the performance of kernels on various configurations of the cache system.

Tools/Software packages required:

Intel Pin tool [1]
Cache Simulator [2]

Options:

1. *Use your own machine:*
    a. Build PIN tool on your own IA-32, Intel(R) 64 and Intel(R) Many Integrated Core architectures (basically any x86 arch) system. PIN tool supports Linux*, windows*, macOS* operating systems. (uses gcc with some basic libraries)
    b. Write C++ kernels for gather, scatter and convolution operations.
    c. Compile your kernels and run PIN simulator with the built binaries
    d. Clone the cache simulator tool and use any of your existing python environments to run the simulator.
    e. Repeat the above process for different cache parameters and report your observations and results.
2. *Use W135 machines*:
    a. Access W135 systems using GlobalProtectVPN. (Refer 'how to access W135 systems' section)
    b. PIN tool binaries are pre-built and available in '/home/other/CSE530-FA2022/pin-3.18-98332-gaebd7b1e6-gcc-linux'
    c. Write C++ kernels for gather, scatter and convolution operations.
    d. Compile your kernels and run PIN simulator with the built binaries
    e. Clone the cache simulator tool and use any of your existing python environments to run the simulator.
    f. Repeat the above process for different cache parameters and report your observations and results.

OPTION 1:

Steps to set up the environment:

1. wget https://software.intel.com/sites/landingpage/pintool/downloads/pin-3.18-98332-gaebd7b1e6-gcc-linux.tar.gz -P /tmp/
2. mkdir ~/cse530_labfiles
3. cd ~/cse530_labfiles
4. tar -zxvf /tmp/pin-3.18-98332-gaebd7b1e6-gcc-linux.tar.gz
5. git clone https://github.com/abhishekk06/CachePerformanceOnMatMul.git
6. cp CachePerformanceOnMatMul/pintool_script/pinatrace.cpp ~/cse530_labfiles/pin-3.18-98332-gaebd7b1e6-gcc-linux/source/tools/ManualExamples/
7. cd ~/cse530_labfiles/pin-3.18-98332-gaebd7b1e6-gcc-linux/source/tools/
8. make all [**should take a couple of mins**]
9. export PIN_ROOT=~/cse530_labfiles/pin-3.18-98332-gaebd7b1e6-gcc-linux [**set to full path**]
10. echo $PIN_ROOT
11. cd ~/cse530_labfiles/CachePerformanceOnMatMul
12. rm –r bin [it might fail if 'bin' folder is not there, and that's okay]
13. mkdir bin
14. rm –r traces [it might fail if 'traces' folder is not there, and that's okay]
15. mkdir traces

Steps to compile kernel, generate traces and run the cache simulator:

16. g++ -Wall src/matmul_ijkalgo.cpp -o bin/matmul_ijkalgo.o **[change this to your kernel file and make appropriate changes below]**
17. python utils/random_matrix_generator.py --n 128 --dump input_matrix.in --sparsity 50 [**make sure required libraries are installed in your python environment**]
18. $PIN_ROOT/pin -t $PIN_ROOT/source/tools/ManualExamples/obj-intel64/pinatrace.so -- bin/matmul_ijkalgo.o --input_file input_matrix.in
19. mv pinatrace.out traces/ijk_traces.out
20. source run_simulator.sh ~/cse530_labfiles/CachePerformanceOnMatMul/traces [**give absolute path to traces directory**]

OPTION 2:

Steps to set up PIN tool and Cache simulator:

1. git clone https://github.com/abhishekk06/CachePerformanceOnMatMul.git
2. export PIN_ROOT=/home/other/CSE530-FA2022/pin-3.18-98332-gaebd7b1e6-gcc-linux
3. echo $PIN_ROOT
4. cd ~/CachePerformanceOnMatMul

Steps to compile kernel, generate traces and run the cache simulator:

5. rm –r bin [it might fail if 'bin' folder is not there, and that's okay]
6. mkdir bin
7. rm –r traces [it might fail if 'traces' folder is not there, and that's okay]
8. mkdir traces
9. g++ -Wall src/matmul_ijkalgo.cpp -o bin/matmul_ijkalgo.o **[change this to your kernel file and make appropriate changes below]**

10. source /home/other/CSE530-FA2022/condarc
11. python utils/random_matrix_generator.py --n 128 --dump input_matrix.in --sparsity 50
12. $PIN_ROOT/pin -t $PIN_ROOT/source/tools/ManualExamples/obj-intel64/pinatrace.so -- bin/matmul_ijkalgo.o --input_file input_matrix.in
13. mv pinatrace.out traces/ijk_traces.out
14. source run_simulator.sh ~/CachePerformanceOnMatMul/traces [**give absolute path to traces directory**]

To change the cache system configuration, the YAML file in Simulator/config - config_simple_multilevel needs to be edited. With the help of Intel PIN tool, students will be able to capture memory access (reads/writes separately) within the ROI (Region of Interest) which can later be passed to cache simulator to analyze the performance of different cache architectures.

References

[1] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun and A. Karunanidhi, *Pinpointing Representative Portions of Large Intel ® Itanium ® Programs with Dynamic Instrumentation* 37th International Symposium on Microarchitecture (MICRO-37'04), 2004, pp. 81-92, doi: 10.1109/MICRO.2004.28.

[2] CacheSimulator, https://github.com/abhishekk06/CachePerformanceOnMatMul

[3] GEM5 building, https://www.gem5.org/documentation/general_docs/building

Submission Expectations:

Please note that in addition to sharing the kernel code, students are also expected to capture the following in their solution book:-

- Number of read/write issued as per the generated traces
- Cache architecture block diagram
- Simulator stats related to cache and reasoning behind the observed behavior

Report:

- Maximum of 4 pages! Failure to adhere to this limit will be penalized!
- We expect around 3 graphs per kernel (one for varying the cache hierarchy, one for varying the number of blocks, one for varying the associativity), giving up to 9 graphs in total. It is encouraged to include more graphs if you are within the page limit.
- Please make sure your graphs are legible and easy to read. We expect proper legend, axis, etc.
- You are free to plot numbers in both absolute scale and relative scale (showing percentage improvements)
- Make sure to include your observations from the graphs and explain possible reasoning behind those trends.

Format:

- Submit a **PDF report** with all your observations with graphs or tables with all the logs/scripts as a zip/tar file.

How to access W135 systems:

Most of the assignments can be done on your local computers. However, you can use the Penn State compute resources (W135).

Using W135 compute resources:

1. Install GlobalProtect VPN and 2-factor authentication if you have not already. https://softwarerequest.psu.edu/Home/AllReleases https://accounts.psu.edu/2fa
2. HOW TO remotely access the W135 machines are mentioned in the below link - https://www.eecs.psu.edu/cse-student-lab-access/index.aspx
3. The IT has timely maintenance shutdown, please check your emails frequently and plan your assignment promptly.
4. Each student is allotted with limited storage budget (1G), so use wisely!
5. You might find MobaXterm or Xquartz SSH clients useful.
6. You can create virtual environments because of lack of root privilege, so please create conda session for hassle-free usage.

Conda is also installed in '/home/other/CSE530-FA2022'. Scipy, numpy and terminaltables packages are installed in the conda base environment. You can find the bash environment settings in condarc. Use following command to source settings on bash:

source /home/other/CSE530-FA2022/condarc

FAQs:

Contact:

If your query is not CSE530-specific:
Please type your query in Google and see if there are answers present in websites like stackoverflow.
Otherwise:

1. Read this document.
2. If you do not find an answer: Please include the following details so we can find solutions to your issues faster.

Please mention what you have tried and what worked/did not work in your CANVAS email. If it is environment related include

- machine model name in cat /proc/cpuinfo,
- python version, and
- if you are using CSE lab resources-- conda version and your relevant bashrc settings

CANVAS:

1. **course link: https://psu.instructure.com/courses/2276017**
2. **Question:** I haven't used canvas before. I am not sure which function to communicate
3. **https://psu.instructure.com/conversations#filter=type=inbox** -> Click on "compose new message".
4. https://psu.instructure.com/courses/1741795/pages/4a-intro-to-communicationtools?module_item_id=20342415
5. https://vimeo.com/103432881?embedded=true&source=vimeo_logo&owner=9556738

Assignment-1 machine requirements:

1. CSE-135 machines will work for Assignment-1. If you are using a personal machine, make sure it is an x86-based machine.
2. I have a windows machine and I wanted to know if the assignment specifically requires Linux OS or can be done on Windows machine.
    a. If it is an Intel machine, you should be able to do it on your personal machine. Check link to identify processor architecture https://www.howtogeek.com/413942/how-to-see-whatcpu-is-in-your-pc-and-how-fast-it-is/ and https://www.howtogeek.com/706226/how-tocheck-if-your-mac-is-using-an-intel-or-apple-silicon-processor/
3. **Linux vs Windows:** These are not CSE530-specific. Please use Google search for these kinds of issues. This project was tested to work on Unix-like systems. The project may or may not work on other OSes. Students are free to try it, but support for such issues is limited.
4. What are the Windows alternatives for the Linux commands specified in the shown examples?
    a. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server2012-r2-and-2012/cc754340(v=ws.11)
5. Wget does not work on Windows. What do I do?

a. Simply download the file! It is a link, and *wget* downloads it!
b. https://www.google.com/search?q=wget+windows+command+line&oq=wget+windows+command+line&aqs=chrome..69i57.4479j0j7&sourceid=chrome&ie=UTF-8

6. PIN tool does not work on Windows/MAC/google colab/python notebook. What do I do?
   a. **This was not tested. If it does not work, use CSE 135 lab resources.**

Using CSE resources:

· Please read the "How to access W135 Machines" section in this document for details on how to access CSE resources. The link can also be found on the Syllabus page in Canvas.

Conda environment and dependencies:

1. If you are using CSE lab resources and you have limited space, you may find the Anaconda installation in **/home/other/CSE530-FA2022** helpful.
2. Scipy, numpy and terminaltables packages are installed in the conda **base** environment.
3. You can find the bash environment settings in condarc. Use following command to source settings on bash:

   source /home/other/CSE530-FA2022/condarc

Disk space issues on CSE lab resources:

1. You should be able to work within the 1GB limit since Conda is installed on a common path (see details above).
2. The command below will help in planning the 1GB space in your CSE grads folder.

   du –h ~
3. You could limit the space usage by cleaning and building just the source/tools/ManualExamples folder. (Do not clean the pin executable)
4. If you do not need any other changes to the PIN-instrumentation functions, PIN library (used in the example) is built in /home/other/CSE530-FA2022/pin-3.18-98332-gaebd7b1e6-gcclinux/source/tool/ManualExamples/obj-intel64.
5. In the worst case,
   a. If you are Ph.D./M.S. student, you will have research space in /home/<groupname>/<user-name> – please use this folder, so you do not have the 1GB limit in /home/grads/<user-name>
      1. You could contact your research group for more details on this.
      2. You could contact CSE helpdesk if details of your research group if you do not have a folder allocated for you yet.
   b. If you are an E.E. or undergrad or anyone without research space and you want to use CSE lab resources, please contact CSE helpdesk specifying the course number and request for disk space for one semester.

Errors

1. *source /home/other/CSE530-FA2022/condarc Illegal variable name.*
   a. **Solution**: The given condarc is for bash shell. If you need csh/other shells, you can convert those as needed.

2. **Error cache_simulator.py: No such file or directory**
    a. Solution: Run the run_simulator.sh script from the CachePerformanceOnMatMul folder.

What to submit?

1. **Question: Is there a specific format we should use to turn in our work? Like a ipynb file or just a document with answers and the results?**
    a. **Answer: Submit all your files as a zip, along with a PDF report with all the graphs, Export ipynb to a html / PDF. The important part is the PDF report with all the graphs. Without the PDF report, no points will be awarded.**
2. **Question: I know you want the results, but how much of the results? Should it just be everything after "Simulation Complete", (which contains instruction count, cycles, and tables generated of the caches)?**
    a. **Answer: Do NOT copy and paste the simulation output. Do attach it as a log. However, ONLY your observations along with graphs will get points. Try to get at least 3 points on each graph to talk about a trend.**
3. **Question: The assignment update file mentions writing C++ code for the kernels. Are we writing algorithms for the kernels or are we just writing YAML configurations based on different cache structures?**
    a. **Answer: Both. The C++ workloads (also called kernel, benchmark, or application) have distinctive characteristics as you will see when simulated on your cache system. Once you have written your kernels, start experimenting with different YAML configurations – vary the cache size and plot a performance metric against it. Similarly for all the other cache configuration knobs given to you.**
4. **The expectation is that the report should have graphs showing the trend of performance (clock cycles, miss rate, hit rate, #misses) with respect to each cache capacity, associativity and block size for all workloads considered.**
5. **Focus**: The assignment requires you to characterize the cache system – how it behaves with different workloads. **Just writing only the C++ application will not gain any points.**
6. References to cache performance studies in literature (if interested):
    a. Performance tradeoffs in cache design (https://ieeexplore.ieee.org/abstract/document/5239?casa_token=wk6TmNojr8AAAAA:sm69vRmipBPxV2WuZCN6eva655qAqv6pTu1Yx9iBUXFFULtqhfH5zDKe7BKGRMKPJgLevE8)
    b. Efficient trace-driven simulation methods for cache performance analysis (https://dl.acm.org/doi/abs/10.1145/98460.98497)
    c. The cache performance and optimizations of blocked algorithms (https://dl.acm.org/doi/pdf/10.1145/106974.106981)
7. **Kernels/workloads:**
    a. **Scatter and gather:** You may find the random matrix generator useful. Try to vary sparsity and understand how the cache system behaves with the sparsity parameter.

    python utils/random_matrix_generator.py --n 10 --dump input_matrix.in --sparsity 50

    b. Please read the *utils/random_matrix_generator.py* file. Open the csrA_input_matrix.in file and see the example outputs that get created. You are free to modify the script as

per your needs. You can read about CSR format here - https://en.wikipedia.org/wiki/Sparse_matrix#:~:text=The%20compressed%20sparse%20r ow%20(CSR,row%20indices%2C%20hence%20the%20name.

c. Question: I think in the example from the instructions we read files as a matrix instead of a CSR format. Should I take care of CSR or just use regular matrix as input?
    i. Answer: It is your preference. If the program you write requires it in CSR, dump & use CSR. If the program you write requires it in regular matrix format, dump & use regular format.

d. For scatter and gather, indexing vector is also generated randomly? and input vector is 1D array or 2D matrix? Is there any vector length limit?
    i. Answer: The assignment requires you to characterize the cache system – how it behaves with different workloads. So, characterize and see if these parameters matter

Grading

1. Since the assignment is open-ended, the evaluation will be relative (curved based on class performance. More insightful observations get more points.)