Assignment# 1

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 (11 p.m.).
- No single line answers are accepted in the submission.
- Late submission will not be accepted unless prior approval (24hr before the deadline) is taken from the Instructor or TA on the ground of medical/personal emergency or academic deadlines.
- 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 queries**

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:

- Column wise tensor copy
- Scatter Operation
- Gather Operation
- Transpose

Cache system configuration knobs for different cache architectures:

- Cache hierarchy depth - It is used for creating multi-level cache hierarchy
- Cache size/blocks - It is per cache and can be modified to have different cache size
- Cache associativity - To play with associativity of each cache level

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

- Input tensor size and the number of read/write issued as per the generated traces
- Number of indices used (for scatter/gather operation)
- Cache architecture block diagram
- Simulator stats related to cache and reasoning behind the observed behavior

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.

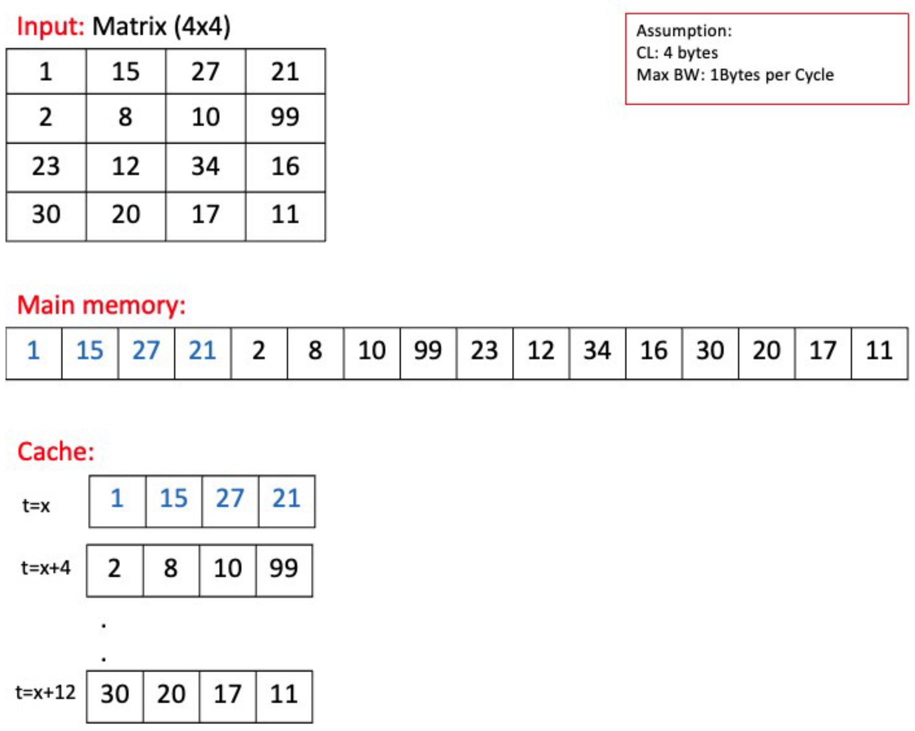

Figure 1: Matrix representation in memory system. (CL: cacheline size.)

Listing 1: Row wise memory copy kernel

```
//Here src[n][n] is input tensor
//of size=nxn passed by user
int dst[n][n] = {};
start_roi(); //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];
      }
}
end_roi(); //Pintool will stop recording memory access
```

Simulator Details:

Using Intel PIN and python-based cache simulator:

The student 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]

Steps to set up the environment: (See python notebook/PDF for the same steps below)

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 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
9. env PIN_ROOT=/cse530_labfiles/pin-3.18-98332-gaebd7b1e6-gcc-linux/
10. echo $PIN_ROOT
11. cd cse530_labfiles/CachePerformanceOnMatMul
12. rm –r bin && mkdir bin
13. rm –r traces && mkdir traces

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

14. cd cse530_labfiles/CachePerformanceOnMatMul
15. rm –r bin && mkdir bin
16. rm –r traces && mkdir traces
17. g++ -Wall src/matmul_ijkalgo.cpp -o bin/matmul_ijkalgo.o
18. python utils/random_matrix_generator.py --n 10 --dump input_matrix.in --sparsity 50
19. $PIN_ROOT/pin -t $PIN_ROOT/source/tools/ManualExamples/obj-intel64/pinatrace.so -- bin/matmul_ijkalgo.o --input_file input_matrix.in
20. mv pinatrace.out traces/ijk_traces.out
21. source run_simulator.sh cse530_labfiles/CachePerformanceOnMatMul/traces

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