You can do this assignment in a Group of at most 2 students. Due: Sep. 29, 2025

Marks [100]

P1. Constructing Roofline Model [30+20 marks]

In this problem you will construct appropriate roofline models for two benchmarks that help to explain performance of the two applications on your laptop/desktop. For the sake of simplicity, you can focus on single-threaded application running on a single core.

B1. Matrix-Matrix multiplication of B x C where B and C are matrices of size [4096 x 4096], with each element of the matrix being a double. You will consider four different versions of the program:

1. Simple-[i,j,k] loop order

```
for(i=0; i<N; i++)
   for(j=0; j<N; j++){
     A[i][j] = 0.0;
      for (k=0; k < N; k++)
        A[i][j] += B[i][k] * C[k][j];
   }
```

2. Simple-[k,i,j] loop order – similar to the above, but with permuted loop order
3. Tiled-[i,j,k] loop order

```
for(i=0; i<N; i=i+b)
   for(j=0; j<N; j=j+b){
      A[i][j] = 0.0;
      for (k=0; k < N; k=k+b)
         for(ii=i; ii<i+b; ii++)
            for(jj=j; jj < j+b; jj++)
               for(kk=k; kk < k+b; kk++)
                    A[ii][jj] += B[ii][kk] * C[kk][jj];
   }
```

4. Tiled-[k,i,j] loop order – similar to the above, but with permuted loop order

For each of these versions compute the performance (FLOPS) of the matrix multiply program and plot it in the roofline model graph. To compute the intensity of the program, you could use performance monitoring counters available in the architecture to measure the relevant parameters. Your roofline should include DRAM, L3, L2, and L1 bandwidth lines, so that the performance of simple and tiled versions can be explained.

B2. Breadth First Search program in the GAP benchmark suite (https://github.com/sbeamer/gapbs) on a large graph with 2^{25} vertices. In the initialization phase of the program, the graph is generated and associated data structure for the graph is created. The computation phase consists of computing the BFS value. Since BFS does not involve FP computation, the performance metric Traversed Edges Per Second (TEPS) will be used for throughput. Intensity should also be defined appropriately. Develop the roofline model for the computation phase of the BFS algorithm. Plot and explain the performance of the BFS algorithm on your laptop in this roofline model. Discuss if TEPS is a useful performance metric. Explore at least one optimization that you could perform to improve the BFS performance. Show the performance of the optimized version in the roofline model.

Write a report summarizing your findings. The following resources may be useful in constructing the roofline model.
https://crd.lbl.gov/divisions/amcr/computer-science-amcr/par/research/roofline/

P2. Obtaining CPI Stack for Programs using Hardware Performance Counters and Linear Regression [10+20+20 marks]

(a) In this exercise first you will obtain the IPC characteristics of two benchmark of programs (one from GAP benchmark suite and another from Rodinia or PARSEC) on your laptop/desktop. You will use the performance counter values to compute the IPC of (short) intervals of program execution and plot them for entire execution of the program (at least 10B instructions). Identify different (major) phases in program execution using IPC characteristics plot. Consider only the phases are coarse, at least 100M or more instructions.

(b) In the second part of this problem, you are required to obtain the CPI (Cycles Per Instruction) stack for programs using hardware performance monitoring counters. CPI stack divides the total CPI of an application into components that reflect the time spent in various events, such L1-I Cache misses, L1-D cache misses, L2/L3 Cache misses, I-TLB and D-TLB misses, branch misprediction, etc. (See [1] for details regarding CPI Stack). The counts of various miss events occuring during program execution can be obtained using hardware Performane Monitoring Counters (PMC) on modern architectures. (See e.g., [2] for details regarding PMCs available on Intel Skylake Processor). Performance counter values for running application can be obtained using performance monitoring tools such as PAPI [3] or `perf` [4].

You are required to develop a simple linear regression model (using simple linear terms) for CPI for each application. Since we are interested in the CPI stack where the contribution due to different miss events are supposed to be additive, the regression model should have only non-negative coefficients. The regression model (with additive terms) essentially gives the CPI stack. For each program-input pair, there would be a separate regression model which gives the CPI stack for that program.

To build the regression model, one requires a training data set consisting of at least a few hundred to may be a few thousands data points for a given program-input pair. One crude solution is to measure the PMC values over several intervals (could be fixed intervals of say 100M instructions/cycles or 10mSec of execution), so that for a single program we can have many data points. The interval size given above is only indicative. It (the interval size) should be chosen appropriately so that there are enough data points for single program-input pair and the intervals are sufficient large to give meaningful counter values for building the CPI stack. Note that the model would be somewhat approximate due to various reasons. Also, validating the models may be more involved (using simulators) and is not be a part of the exercise.

You are required to develop the CPI stack (or separate linear regression model) for two benchmark programs. In each case choose the appropriate input for the program. You can experiment with different interval sizes and different sets of features (miss events) in building your regression model. Report the RMSE, R^2, adjusted R^2 values, Residuals, F-statistic and p-value to assess the quality of the model generated of the model. Comparing the CPI stacks across the programs, you can report your observations for different programs.

For one of the programs, construct the CPI stack for any 3 distinct phases and discuss (compare and contrast) the composition of the CPI stacks for these phases.

Write a report summarizing your findings.

A few remarks/points to be noted are listed below:

(a) As the performance counters available on a processor differ across generation of processors, you can go through the processor manual to know the list of available counters specific to your processor generation.

(b) In order to read these performance counters, performance monitoring tools like perf and PAPI can used in Linux environment. Perf tool presents a simple command line interface and can be installed as a linux utility tool using the command
"apt-get install linux-tools-common linux-tools-generic linux-tools-`uname -r`"

install linux-tools corresponding to your exact Linux Kernel version. You can read more about perf at https://perf.wiki.kernel.org/index.php/Tutorial

(c) PAPI provides APIs that can be inserted in the source code to query the performance monitoring counters.
You can find more information about PAPI and installation at https://icl.utk.edu/papi/

(d) When you run the application on your laptop/desktop and gather performance counter values, make sure that no other processes (like browsers, video players, other C programs etc.) are running on your system/or on the specific core as they could influence the counter values. You can also pin your process to a specific core and run the above tools to read counters on that core. Also ensure you run all the experiments in the same system while collecting the performance counters

(e) You can choose applications which run for atleast 10 seconds so that there are at least 1,000 data points generated. Note that these numbers (interval size and the number of data points) are indicative. You can choose appropriate values, appropriate regions of interest in the program, but document these in your report.

(f) You should use as many miss events as required for the model. Note that some of the miss events may not figure as significant terms in the linear model for some benchmark-input pair . It is receommended that you measure at least 7 miss events for developing the model. In building the model you can consider only linear terms and no need to consider interaction terms (interaction of two events and their counter values).

(g) In building the regression model the feture values in the train data could be normalized so that individual feature values are always between 0 and 1.

(i) In building the model, you should take care to avoid over-fitting the data. You could use some part of the data (chosen randomly) to perform cross-validation.

(j) You can use available R-packages (or other open source software) for building the regression model.

[1] A Top-Down Approach to Architecting CPI Component Performance Counters Article in IEEE Micro · February 2007
[2] Intel® 64 and IA-32 Architectures Software Developer's Manual - Volume 3B: System Programming Guide, Part 2
[3]PAPI User's Guide. Available at:
http://icl.cs.utk.edu/projects/papi/files/documentation/PAPI_USER_GUIDE_23.htm
[4] Linux Kernel Profiling with perf. Available at:
https://perf.wiki.kernel.org/index.php/Tutorial