CHALMERS
UNIVERSITY OF TECHNOLOGY

DAT400 Lab 3: Performance Modeling

DAT400 staff

September 2023

1 Objective

In this lab, you will learn how to characterize an application on a given computing platform using Roofline performance modelling tools.

2 Introduction

Performance models and tools are an integral part of the performance analysis and performance optimization process for users who seek higher performance and better utilization of the hardware. The Roofline performance model offers an intuitive and insightful way to compare application performance against machine capabilities, track progress towards optimality, and identify bottlenecks, inefficiencies, and limitations in software implementations and architecture designs. Its ability to extract key computational characteristics and abstract away the complexity of modern memory hierarchies has made Roofline-based analysis an increasingly popular tool in the HPC community.

The most standard Roofline model is as follows. It can be used to bound floating-point performance (GFLOP/s) as a function of machine peak performance, machine peak bandwidth, and arithmetic intensity of the application. The resultant curve (hollow purple) can be viewed as a performance envelope under which kernel or application performance exists, see Figure 1.

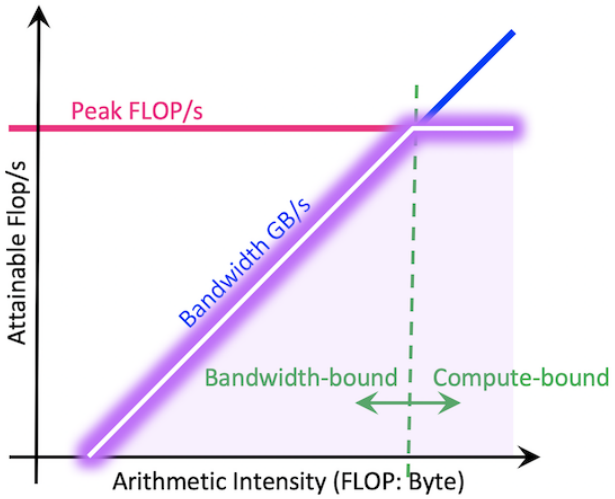

Figure 1: Roofline Model.

The ridge point on the Roofline is called the 'machine balance' point. Usually, if an application's arithmetic intensity is lower than this point, it is considered to be bandwidth bound, i.e. bound by

how fast the data can be moved through the memory system instead of how fast the calculations can be done on the CPU core or the GPU SMs. To optimize in this case, memory inefficiencies are usually good places to look at, such as the memory access pattern, data locality and cache reuse. On the other hand, if the application's arithmetic intensity is higher than machine balance, then the application is more likely to be limited by how fast the computation can be done. In this case, improving vectorization (to more efficiently utilize the vector units on each CPU core), or multi-threading (to utilize the multi or many cores better), can usually help.

2.1 Characterizing Applications on a Roofline Model

To characterize an application on a Roofline, three pieces of information need to be collected about the application;

1. Application execution time.
2. Total number of Flops performed.
3. Total number of Bytes read and written.

This can be for the entire application or only for a code region that is of interest.

2.2 Empirical Roofline Toolkit (ERT) for machine characterization

The Empirical Roofline Tool, ERT, automatically generates roofline data for a given computer. This includes the maximum bandwidth for the various levels of the memory hierarchy and the maximum gflop (Giga Flop) rate. This data is obtained using a variety of "micro-kernels". ERT runs a variety of micro-kernels and sweeps through a range of runtime configurations. These micro-kernels may be small and designed to just test a particular aspect of the system, but together they provide a more realistic set of estimations for the machine capability such as peak bandwidth on various cache levels, and peak GFLOP/s.

Instructions:

- Clone the ERT repository using
  `git clone https://bitbucket.org/berkeleylab/cs-roofline-toolkit.git`
- Please make sure that there are no heavy background processes or multiple users logged in. This can be checked by *htop* and *who* commands.
- Copy the config file config.lab3 in Canvas to folder cs-roofline-toolkit/Empirical_Roofline_Tool-1.1.0/Config
- Go to file Empirical_Roofline_Tool-1.1.0/Scripts/summary.py line 108, change 1.15 to 1.2, save and quit;
- Standing in Empirical_Roofline_Tool-1.1.0/, run ert: ./ert --verbose=2 --run ./Config/config.lab3

2.3 Arithmetic Intensity using Intel SDE

Arithmetic Intensity (AI) is the ratio of total floating-point operations (FLOPs) performed by a given code or code section, to the total data movement (Bytes) required to support those FLOPs. For large-scale applications, it is infeasible to estimate the FLOPs or bytes by hand and performance tools are recommended.

The Intel SDE tool offers dynamic instruction tracing capability. It can capture information such as the instructions executed, instruction length, instruction category and ISA extension grouping,

enabling accurate FLOPs estimation for a full application or a code region. Fore more details about Intel SDE, please refer to https://software.intel.com/content/www/us/en/develop/articles/intel-software-development-emulator.html.

Instructions:

- Download Intel SDE latest linux version from official website.
- Open a terminal and set OpenMP thread number: export OMP_NUM_THREADS=4
- Compile stream.c. Example:
  gcc -O -DNTIMES=2 -DSTREAM_ARRAY_SIZE=100000000 -fopenmp stream.c -o stream.100M
- Check your machine's CPU architecture family: cat /sys/devices/cpu/caps/pmu_name
- Run SDE. Example:
  ./sde64 -skx -i -global_region -omix YOUR_OUTPUT_FILE -- ../stream.100M
- Use the parser provided in the lab to parse the output:
  ./parse-sde.sh YOUR_OUTPUT_FILE

3 Tasks

3.1 ERT

(a) How many cache levels are there in your testing machine?

(b) According to your ERT running results, what is the peak bandwidth for each cache level and DRAM? What is the peak performance of the machine?

3.2 Calculating AI using SDE

Run `stream` benchmark with different problem sizes (e.g., 0.5M, 1M, 10M, 100M) and different number of threads. Answer the following questions for each scenario:

(a) What is the total number of Flops?

(b) What is the total number of Bytes?

(c) What is the execution time of the `stream` benchmark?

(d) Calculate AI.

(e) Calculate the performance (GFLOP/s).

4 Tasks

4.1 Performance Analysis Results

(a) Draw all the results in the ERT figure (e.g., by using gnuplot). Explain your observations. Below are a few guidelines.

- roofline.gnu will by default output a postscript (.ps) file. You may want to change this to for example a png file using *set terminal png*.

- You may need to change the ranges in x and/or y.
- You can save your data in a file called for example *ai.dat*, and then plot it in gnuplot using a command such as *plot 'ai.dat' w p ls 1*.

5 Report Submission

- Submit the report as a group of two. Write down the group number and CIDs of each partner (i.e., who participated in this lab).
- You will get a PASS only if all the tasks are properly addressed in the report.

6 Acknowledgment

This lab is inspired by NERSC roofline model documentation. [1]

[1] https://docs.nersc.gov/tools/performance/roofline/