*P. Koumoutsakos, M. Troyer*
*ETH Zentrum, CLT F 12*
*CH-8092 Zürich*

Fall semester 2014

# Set 6 - Roofline Model and Performance Measures

Issued: October 31, 2014
Hand in: November 7, 2014, 8:00am

Understanding the characteristics of the platform on which performance tests are executed is of fundamental importance since it gives a context in which to read performance results. The primary objective of this exercise is to learn how to characterize computing hardware performance.

### Peak Performance and System Memory Bandwidth

The number of executed floating point operations (FLOP) is a measure used to characterize the costs of scientific software, e.g. computational fluid dynamics codes, finite element analysis programs, computational chemistry and computational biology packages.

The peak floating point performance, hereafter simply called peak performance, is a measure of the quantity of FLOP that a machine can execute in a given amount of time. Typically, the peak floating point performance (PP) in FLOP/s can be computed as in the following:

$$PP\ [FLOP/s] = f\ [HZ = cycle/s] \times c\ [FLOP/cycle] \times v\ [-] \times n\ [-] \qquad (1)$$

where $f$ is the core frequency in CPU cycles per second (given in Hz), $c$ the number of FLOP executed in each cycle, $v$ the SIMD width (in number of floats) and $n$ the number of cores.

Typical floating point performance values are reported in GFLOP/s or TFLOP/s. Floating point performance is also used to assess the performance of a given algorithm implementation (`http://en.wikipedia.org/wiki/FLOPS`).

The bandwidth of the system memory is a measure of the speed of data movement from the system to caches and vice-versa. As the problems considered here in fit on a single node, we are mostly interested in quantifying the DRAM bandwidth.

Given the specifications of the DRAM memory, the theoretical memory bandwidth (PB) in B/s can be computed as follows:

$$PB\ [B/s] = f_{DDR}\ [Hz = cycle/s] \times c\ [channel] \times w\ [bit/channel/cycle] \times 0.125\ [B/bit] \quad (2)$$

where $f_{DDR}$ is the DDR clock rate, $c$ the number of memory channels and $w$ the bits moved through a channel per cycle (typically 64 bits). Typical bandwidths are reported in MB/s, GB/s or TB/s (`http://en.wikipedia.org/wiki/Memory_bandwidth`).

Memories based on the DDR (Double Data Rate) technology, such as DDR-SDRAM, DDR2-SDRAM, and DDR3-SDRAM[1], transfer two data per clock cycle. As a result, they achieve

[1] `http://en.wikipedia.org/wiki/DDR3_SDRAM`

double the transfer rate compared to traditional memory technologies (such as the original SDRAM) running at the same clock rate. Because of that, DDR-based memories are usually labeled with double their real clock rate. For example, DDR3-1866 memories actually work at 933 MHz transferring two data per clock cycle, and thus are labeled as being a "1,333 MHz" device, even though the clock signal does not really work at 1.866 GHz.

## Question 1: Roofline Model

The roofline model[2] is a simple visual tool that can be used to understand performance on a multicore architecture.

The hardware architecture is abstracted to its memory bandwidth and its peak floating point performance.

For this exercise you will learn how to use the Roofline model for the solution of the diffusion equation with Finite Differences.

a) According to the wiki of the Euler cluster, each 24-core Ivy-Bridge compute node is capable of delivering 576 GFLOP/s. In addition, the theoretical memory bandwidth of each of the two NUMA nodes of a single compute node can reach 59.7 GB/s. Try to justify the above numbers.

b) Draw the roofline for the 24-core Ivy-Bridge node of Euler. Note that the bandwidth is found as the ratio of GFLOP/s performance to operational intensity.

c) Compute the number of FLOP, bytes of data read and written and the operational intensity for a single timestep of diffusion using explicit Euler (use your code from Exercise 3 as a reference):

$$\rho_{i,j}^{n+1} = \rho_{i,j}^{n} + D\delta t \frac{\rho_{i+1,j}^{n} + \rho_{i-1,j}^{n} + \rho_{i,j+1}^{n} + \rho_{i,j-1}^{n} - 4\rho_{i,j}^{n}}{\delta x^2} \tag{3}$$

d) Show the operational intensity on the plot. Is your implementation memory bound or compute bound? What is the maximum performance that your code can reach according to the model?

e) Measure the performance (in GFLOP/s) of your code from Exercise 3 or the code provided in the master solution. Put the results as points in your roofline plot. How efficient are the two codes compared to the expected value? What could be done to get closer to the theoretical peak performance?

## Question 2: Performance Measures

Sometimes the computation of the peaks of a platform is not sufficient as you might want a more realistic upper bound on what performance you can effectively reach with a program. The objective of this exercise is to get as high as possible with your current knowledge.

a) Write a C/C++ benchmark to measure the peak single precision performance of a given platform. Report the method you used to measure the peak performance. Try your code on a compute node of the Euler cluster. Can you reach the theoretical peak performance reported before? Please explain your observations.

[2] Williams et al, 2009: the original paper can be found on the course website.

b) Write a C/C++ benchmark to measure the peak memory bandwidth of a given platform. Report a short description of the method you use in the benchmark. Plot the measured bandwidth against the number of threads.

   Disable any OpenMP pragmas and run your program on a single core (i.e one thread) for different memory footprints (total size of your data set). Starting your measurements for 10KB of memory footprint. Plot using a log-log scale the bandwidth against the memory footprint. What do you observe in the plot?

# Summary

Summarize your answers, results and plots into a PDF document. Furthermore, elucidate the main structure of the code and report possible code details that are relevant in terms of accuracy or performance. Send the PDF document and source code to your assigned teaching assistant.