P. Hadjidoukas

Set 1 - Performance Measures

Issued: March 1, 2023

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:

$$PP\ [FLOP/s] = f\ [HZ = cycle/s] \times c\ [FLOP/cycle] \times v\ [-] \times n\ [-], \tag{1}$$

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.

The exact values for the above features can be found at the hardware specifications or given by the system administrators. For example, for the Euler II cluster[1] we can find the specifications of the Intel Xeon E5-2680v3 here: `https://ark.intel.com/products/81908/`.

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:

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

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`).

[1]`https://scicomp.ethz.ch/wiki/Euler`

Memories based on the DDR (Double Data Rate) technology, such as DDR-SDRAM, DDR2-SDRAM, and DDR3-SDRAM[2], transfer two data per clock cycle. As a result, they achieve 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 "1,866 MHz" device, even though the clock signal does not really work at 1.866 GHz.

Question 1: Performance Measures

a) According to the wiki of the Euler cluster, each of the two sockets on a node hosts an Intel Xeon E5-2680v3 12-core Haswell CPU capable of delivering 480 GFLOP/s each. In addition, the theoretical memory bandwidth for each of the two sockets can reach 68.3 GB/s. Try to justify the above numbers.

From `https://ark.intel.com/products/81908/` we find the following specifications for the Intel Xeon E5-2680v3:

- 12 cores
- 2.50 GHz clock frequency
- Supports AVX2 extended instruction set (256-bit wide FMA instructions)
- The Intel Haswell architecture can issue 2 FMA instructions per cycle

The Haswell architecture has two execution units for FMA instructions which yields a maximum of four flops per cycle (assuming you have a high instruction density to hide the 5 cycle latency of the FMA instructions). The 256-bit registers can hold either 4 double precision numbers or 8 single precision numbers. The total double precision peak performance is then calculated by (for a base frequency of 2.5 GHz)

$$\pi = 2.50 \times 10^9 \text{ cycle/s} \times 12 \text{ cores} \times 4 \text{ SIMD width} \times 4 \text{ flop/cycle} = 480.0 \text{ Gflop/s}. \quad (3)$$

Note that the processor frequency is allowed to vary and can go up to 3.3 GHz.

For the memory we find

- Supports DDR4 memory with frequencies 1600 MHz/1866 MHz/2133 MHz
- Maximum number of memory channels is 4
- 64-bit architecture with word size 64-bit

Given this data, we can compute the nominal peak bandwidth by (for the fastest possible memory)

$$\beta = 2.133 \times 10^9 \text{ cycle/s} \times 4 \text{ channel} \times 64 \text{ bit/channel/cycle} \times \frac{1}{8} \text{ byte/bit} = 68.3 \text{ Gbyte/s} \quad (4)$$

b) Try to find the nominal peak performance and the theoretical memory bandwidth for your laptop or desktop machine.

We consider a laptop with the following hardware specifications:

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

- Processor: Intel(R) Core(TM) i7-7567U CPU 3.50 GHz
- Memory: 2133 MHz LPDDR3

Based on the information available at: `https://ark.intel.com/products/97541`, the system is based on the Kaby Lake (`https://en.wikichip.org/wiki/intel/microarchitectures/kaby_lake` microarchitecture and the peak performance of the system is

$$\pi = 3.5 \times 10^9 \text{ cycle/s} \times 2 \text{ cores} \times 4 \text{ SIMD width} \times 2 \text{ flop/cycle} = 56.0 \text{ Gflop/s}. \qquad (5)$$

while the theoretical memory bandwidth is 34.1 Gbyte/s, half of the previous system (Eq. 4) due to the two memory channels.

c) Sometimes the nominal peak 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.

Write a small C/C++ benchmark code 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 your laptop or desktop system. Can you reach the theoretical peak performance reported before? Please explain your observations.

The source code for this exercise is in the `ex01_solution_code.zip` file.

In order to reach peak performance on the Kaby Lake architecture, we must minimize access to slow memory and ensure a high density of fused-multiply-add (FMA) instructions. The Kaby Lake architecture can issue 2 FMA instructions each cycle. Furthermore, to hide latency of the FMA instruction we must ensure that we issue at least 10 independent FMA instructions at each cycle.

We try to reach peak performance by performing the multiply-add operation

$$q = \alpha r + q$$

multiple times in a loop. The value α is a constant where q and r are values that change during the loop iterations. Given our current knowledge, we implement the balanced multiply-add operation above and hope that the compiler will recognize the pattern and generate FMA instructions in the assembly code. The code is compiled with `-O3 -march=core-avx2 -ffast-math -funroll-loops` optimization flags.

The code achieves 13.44 Gflop/s on average, which corresponds approximately to 25% of the nominal peak performance on two cores and 50% if we consider the single core. If we inspect the assembly code generated by GCC, we observe that the compiler does not recognize the FMA pattern and instead uses addition and multiply instructions (vaddsd and vmulsd (`https://docs.oracle.com/cd/E36784_01/html/E36859/gntbd.html`).