How to determine the amount of FLOPs my computer is capable of

I would like to determine the theoretical number of FLOPs (Floating Point Operations) that my computer can do. Can someone please help me with this. (I would like to compare my computer to some supercomputers just to get an idea of the difference between them)

floating-point

edited Apr 10 '14 at 2:10

asked Apr 8 '14 at 5:35

Ol' Reliable
138 1 1 6

3 Answers

The theoretical peak FLOP/s is given by:

$$\text{Number of Cores} * \text{Average frequency} * \text{Operations per cycle}$$

The number of cores is easy. Average frequency should, in theory, factor in some amount of Turbo Boost (Intel) or Turbo Core (AMD), but the operating frequency is a good lower bound. The operations per cycle is architecture-dependent and can be hard to find (8 for SandyBridge and IvyBridge, see slide 26). It is the subject of this stack overflow question, which includes numbers for a bunch of modern architectures.

edited Apr 10 '14 at 5:59

answered Apr 8 '14 at 12:00

Max Hutchinson
2,045 6 22

Ok, I have 2 cores, Operating Frequency: 1.8 GHz, Intel Turbo Boost Technology: 3.00 Ghz, I can't find the operations per cycle, here is the website: ark.intel.com/products/75460/… thanks – Ol' Reliable Apr 10 '14 at 2:15

Haswell can do 16 DP / cycle. I just added a link in the answer body to an SO answer. – Max Hutchinson Apr 10 '14 at 6:00

Does this mean that my computer can do: 2 x 3,000,000,000 Hz x 16 = 96 Giga FLOPs? – Ol' Reliable Apr 10 '14 at 20:05

It means that it could do between 2*1.8 GHz*16 DP = 57.6 GFLOP/s and 96 GFLOP/s, depending on the actual average frequency. If you need to use a single number, 57.6 is the more fair one, IMO. – Max Hutchinson Apr 10 '14 at 22:42

1 FLOP rates are generally a poor measure of the 'goodness' of a processor. See scicomp.stackexchange.com/questions/114/… for example. You might want to think about the limiting costs of your task (e.g. compute bound vs memory bound vs disk bound) and focus on the relevant hardware (compute system, memory system, I/O). – Max Hutchinson Apr 13 '14 at 21:21

You will need to know the model and vendor of the CPUs in your machine. Once you have that, you can look up on the vendor's website (or maybe on Wikipedia) the clock rate, number of chips/sockets, number of cores per chip, number of floating point operations per cycle, and the vector width of those operations. Then, you simply multiply.

Take, for example, the Intel Xeon E5-2680 "Sandy Bridge" processors in Stampede where I work. The specs are:

- 2.7GHz
- 2 chips/node, 8 cores/chip
- 2 vector instructions/cycle
- 256-bit wide AVX instructions (4 simultaneous double-precision operands)

Multiplying those gives 345.6 GF/node or 2.2 PF for the un-accelerated part of the system.

We usually think in terms of double-precision (64-bit) operations, because that's the precision required for the vast majority of our users, but you can can redo the calculation in single-precision terms if you like. This usually only changes the last factor, say 8 SP Flops/instruction instead of 4 DP Flops/inst, but it can be wildly different from that. Older GPUs, for example, only did DP at about 1/8th the rate of SP. If you ever quote a number for your system, you should be explicit

about which you used if it's not double-precision because people will assume it was, otherwise.

Also, if your chip supports fused multiply-add (FMA) instructions, and it can do them at full rate, then most people count this as 2 floating-point operations though a hardware performance counter might count it as only one instruction.

Finally, you can also do this for any accelerators that might exist in your system (like a GPU or Xeon Phi) and add that performance to the CPU performance to get a theoretical total.

edited Apr 8 '14 at 13:32

answered Apr 8 '14 at 11:59

Bill Barth
8,453 1 10 28

It's not enough to know the CPU model, one needs to find out the actual operating frequencies – Aksakal Apr 10 '14 at 2:36

@Aksakal, for a theoretical analysis, it's probably OK to pick the nominal frequency. It's hard to know what frequency your chips will actually run at since that can depend on the workload and the quality of your air conditioning. – Bill Barth Apr 10 '14 at 2:57

I understand that you asked for the theoretical value, but as this is nearly always inaccessible by any real code, even LINPACK, you might want to just run (optimized) DGEMM for very large matrices. The reason that I prefer this method is that it exposes some of the shortcomings of certain processors that prevent them from achieving their theoretical peak flop value.

For example, NVIDIA GPUs currently do integer and floating-point operations on the same pipeline. This means that you can only achieve the theoretical peak flop/s if you do *no integer computation whatsoever*. As array indexing and any other form of data access requires integer arithmetic somewhere, no code can achieve the theoretical peak flop/s on an NVIDIA GPU. In most cases, one sees ~80% as the upper bound. For CPUs that issue integer and floating-point operations simultaneously, this is a non-issue.

On some GPU-like multicore processors like Intel Knights Corner and Blue Gene/Q, it is harder to achieve the peak flop/s than on traditional CPUs for similar pipeline issues (although both can achieve ~90% of peak in large DGEMM at least).

answered Apr 8 '14 at 18:06

Jeff
1,227 7 14