Performance analysis for beginners

Dan Andresen, dan@ksu.edu, Kansas State University

1. Why do performance analysis? So we can interpret a paper's results and assertions correctly in the context of other related work.
   From [1]:

> Reproducing experiments is one of the main principles of the scientific method. It is well known that the performance of a computer program depends on the application, the input, the compiler, the runtime environment, the machine, and the measurement methodology [20, 43]. If a single one of these aspects of *experimental design* is not appropriately motivated and described, presented results can hardly be reproduced and may even be misleading or incorrect.
>
> The complexity and uniqueness of many supercomputers makes reproducibility a hard task. For example, it is practically impossible to recreate most hero-runs that utilize the world's largest machines because these machines are often unique and their software configurations changes regularly. We introduce the notion of *interpretability*, which is weaker than reproducibility. *We call an experiment interpretable if it provides enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.* In other words, interpretable experiments support sound conclusions and convey precise information among scientists. Obviously, every scientific paper should be interpretable; unfortunately, many are not.

2. So what we need?
   a. Experimental configuration – what did you run it on? OS & compiler versions, hardware configuration (RAM, processors, network speed, etc.)
      Also from [1]:

> **Our experimental setup** Each node of Piz Daint (Cray XC30) has an 8-core Intel Xeon E5-2670 CPU with 32 GiB DDR3-1600 RAM, an NVIDIA Tesla K20X with 6 GiB GDDR5 RAM, and uses Cray's Programming Environment version 5.1.29. Each node of Piz Dora (Cray XC40) has two 12-core Intel Xeon E5-2690 v3 CPUs with 64 GiB DDR4-1600 RAM and uses Cray's Programming Environment version 5.2.40. Both systems are interconnected by Cray's Aries interconnect in a Dragonfly topology. Pilatus has two 8-core Intel Xeon 5-2670 CPUs with 64 GiB DDR3-1600 RAM per node, is connected with an InfiniBand FDR fat tree topology, and runs MVAPICH2 version 1.9. All ping-pong results use two processes on different compute nodes. For HPL we chose different allocations for each experiment; all other experiments were repeated in the same allocation. Allocated nodes were chosen by the batch system (slurm 14.03.7). The filesystem configuration does not influence our results. All codes are compiled with gcc version 4.8.2 using -O3.

   b. What are you trying to show? Typically you are trying to show scalability, performance, and effectiveness compared to itself and standard algorithms in the field.
3. How do we show this?
   a. Run a variety of experiments. Repeat them enough (usually 3-10 times) to have a reasonable confidence level (or small enough standard deviation).
   b. Choose the experiments to illustrate the interesting and important sections of your results (this is not about your code, it's about your test results) – if your performance

graph is a straight line, you're probably not pushing the system enough. If it's a hockey stick, explain what's going on where it starts turning upwards.

4. What do you vary? Assuming you're on a cluster with, say, 32-core nodes and 128GB of RAM, and a million elements of data, you might measure:
    a. RAM usage – do different versions/datasets of your code use different amounts of RAM?
    b. CPU/# cores/threads – Maybe try 1, 2, 4, 8, 32 threads to see how it scales across a single machine. If you are using MPI, perhaps try 16x2 machines or 32x2. More if you have time. Does having more threads/core help?
    c. Network – what is your network utilization? Does a faster network help?
    d. Data sizes – Maybe try 1000, 10000, 100000, 1M data sizes - how does your system scale?
    e. Programming styles – how does the performance of pthreads, MPI, and OpenMP change each of the above results?
5. How do you show your results? Typically a graph or table. Make sure it's legible!

For an example of a short writeup, see http://people.cs.ksu.edu/~dan/papers/29.pdf .

[1] Torsten Hoefler and Roberto Belli. 2015. Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15). ACM, New York, NY, USA, Article 73, 12 pages. DOI: https://doi.org/10.1145/2807591.2807644

http://dl.acm.org/citation.cfm?id=2807644

So for Beocat:

1. Machine configuration – see https://support.beocat.ksu.edu/BeocatDocs/index.php/Compute_Nodes (and our network is 40GbE or 100GbE, depending on the machine)
2. Software versions – use the 'uname –a' command on a node, or see https://support.beocat.ksu.edu/BeocatDocs/index.php/Installed_software
3. RAM usage -