Quantitatively Assessing Performance Portability with Roofline
(the slides are available under “Presentation Materials”)
Date: January 23, 2019
Presented by: John Pennycook (Intel), Charlene Yang and Jack Deslippe (LBNL/NERSC)

Q: Do you have benchmark applications that do nothing important but have measurable tasks?

A: NERSC users a series of both micro-benchmarks and full applications for a number of purposes. Many of these can be found here: http://www.nersc.gov/research-and-development/benchmarking-and-workload-characterization/ - a number of these are appropriate candidates for a roofline analysis.

Q: How are these concepts (also the Roofline model) extended to include inter-node performance and how do you practically address this?

A: Both the PP metric and Roofline model are focused on single-node analysis, but we think it should be possible to extend either to include inter-node performance. One way to do this could be to perform the analysis for a single node in a large multi-node run. Another way could be to define a "platform" to represent a particular number of nodes, and look at efficiency relative to the peak/best-known performance for that number of nodes collectively. We have only used the metric for single-node analysis so far, but would be interested in collaborating with others to explore its applicability to multiple nodes.

Q: In practice, how to you measure empirical FLOPs and canonical FLOPs (with regard to properly accounts for divides)?

A: In this particular talk, we used SDE to count FLOPs on KNL and nvprof on V100, empirically. Canonical FLOPs is usually done by theoretical estimation based on the algorithm in the code, but in this talk, to be exact, we also used these tools to measure a modified version of GPP where divides are all artificially replaced by multiplies. Our point here is to encourage people to use tools and get empirical measurements on each architecture because theoretical estimation can be inaccurate and implementations of the same operation can change from one architecture to another, even from one arithmetic precision to another.

Q: In how far is it useful to talk about achieving ~100% Performance Portability when not using the optimal algorithm/setup (e.g., wrt to the strided access)

A: This is a good example of why it can be useful to look at both architectural and application efficiencies together. 100% architectural efficiency in a case with large stride shows that the application can't be made faster on the hardware without some form of algorithmic change, but the fact that faster algorithms are known would then be reflected in the application efficiency calculation. If one efficiency is 100% and the other isn't, that tells you where you need to focus to improve.

Q: Do you have a special treatment for zero operations when computing the performance efficiency? Example: on the GPU we are using batched DGEMM by padding all matrices => 90% zero operations but still very fast => very close to peak bandwidth and peak performance. On the CPU we don't have batched DGEMM => quite far away from roofline

A: I guess there's a difference between GFLOP/s and useful/effective GFLOP/s here. It's a good way to see how close you can get to peak FLOP/s, even with the not-so-useful FLOPs, but it's probably only fair when comparing with the CPU performance, to exclude these zero-operations-related GFLOP/s on the GPU. At the end of the day, we want to have an apples-to-apples comparison.

Q:When adding the bandwidths for L1/L2 caches to a roofline model for GPUs, should we use the cumulative bandwidth of all the L1/L2 caches on all the multiprocessors, or just for a single multiprocessor?

A: I have been using the cumulative bandwidths for L1/L2 caches across SMs, and it's probably the easiest. But you can definitely build a Roofline that's just for one multiprocessor, but in that case, you would need to make sure that all the bandwidths and data movement (when calculating arithmetic intensity) to be 'per-multiprocessor' instead of 'per-GPU'. Inconsistent measurements may render not-so-helpful or misleading Roofline.

Q: How would you account for different algorithms being used on different architectures?

A: Applications that achieve very high levels of performance portability are unlikely to use exactly the same algorithm on different architectures, and so I agree that

accounting for this is very important. When developing the metric, we tried to ensure that using different algorithms was compatible with our definitions of both architectural and application efficiency: using different algorithms with different arithmetic intensities on each platform will impact which performance bound (roof) is used to calculate architectural efficiency; and the "best-known" performance results used to calculate application efficiencies may well come from different algorithms on each platform.

Q: How might one verify best known performance and encourage its reporting?

A: Today, verifying best-known performance requires an extensive literature review and maintaining accurate logs of your own performance during development (in order to catch any regressions), which is far from ideal. Several benchmarks in the performance portability space (e.g. CloverLeaf, https://github.com/UK-MAC/CloverLeaf) have multiple implementations available, and this combined with efforts like Supercomputing's Reproducibility Initiative (https://sc18.supercomputing.org/submit/sc-reproducibility-initiative/) should make it easier to find and reproduce best-known performance numbers for commonly used benchmarks.

Ideally we would have some sort of central database for tracking the best-known performance of important problems across a wide set of different platforms. One day we might see something like this, but there's a long road ahead of us -- we need to refine our definition(s) and metric(s) to the point that the community thinks that they are useful and is happy to use them, and then we can start to think about projects like this.