Power and Performance Modeling for High Performance Computing Algorithms

Jee Whan Choi (*Advisor*: Richard Vuduc)
Georgia Institute of Technology
jee@gatech.edu

1 Introduction

The overarching goal of this research is to provide an algorithm-centric approach to analyzing the relationship between time, energy and power. This work is aimed at algorithm designers and performance tuners so that they may be able to make decisions on how algorithms should be designed and tuned depending on whether the goal is to minimize time or to minimize energy on current and future systems.

2 Roofline model for energy

Firstly, we present a simple analytical cost model for energy and power. Assuming a simple von Neumannn architecture with a two-level memory hierarchy, this model predicts energy and power for algorithms using just a few simple hardware and algorithm-specific parameters, such as the number of floating point operations (FLOPs) and the energy cost of moving a unit of data (Joules). Using highly optimized microbenchmarks and a fine-grained power measurement tool, we show that though our model uses only a few simple parameters, it can nevertheless accurately predict energy and power on real systems.

We can summarize the results of our model and experiments by rooflines in time [1] (red solid lines) and *arch lines* in energy [2] (blue solid lines) as shown in figure 1.

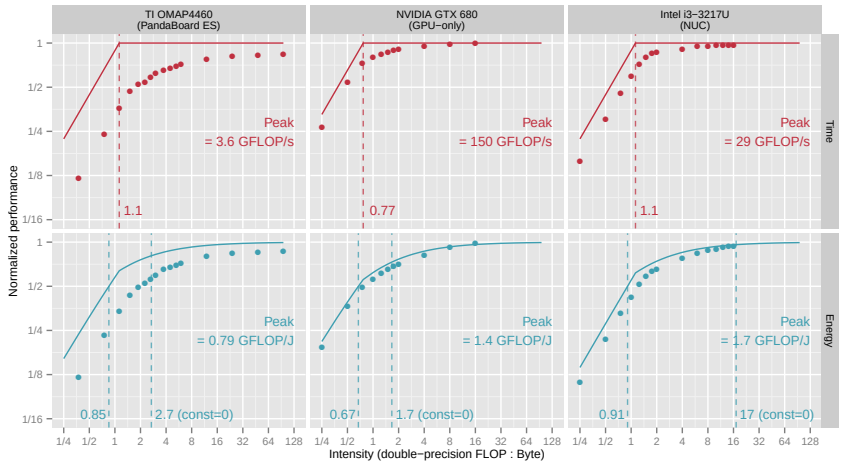

Figure 1: Measured time and energy for a synthetic benchmark corroborates the basic form of our model. We show the true energy-balance point as well as the energy-balance if $\pi_0 = 0$.

Just as there is a *balance* in time for each system (red vertical line), there is also a *balance* in energy, which our arch line in energy helps to demonstrate (left blue vertical line). The energy-balance point measures the ratio of useful compute operations and bytes per unit-energy. If an algorithm is compute-bound in energy, more energy is spent on compute operations than on memory operations, and vice versa when it is memory-bound in energy. This metric can be used to measure the *energy-efficiency* of an algorithm.

For all of our target systems, the time-balance point exceeds the $y = 1/2$ energy-balance point, which means that the time efficiency will tend to imply energy-efficiency. We believe this observation explains why *race-to-halt* can be such an effective energy-saving strategy in practice on today's systems.

If instead it were possible to drive constant power $\pi_0 \to 0$, then the situation could reverse. We show this scenario using the hypothetical energy-balance lines labeled "const=0." In all three cases, this results in the energy-balance exceeding the time-balance, and being compute bound in time no longer implies energy-efficiency. Although this scenario is artificial, with the current scaling trend for energy costs of transistors and interconnects, energy-balance is expected to grow faster than time-balance. Therefore, in the future, optimizing for energy may be the "nobler" goal as energy-efficiency will imply time-efficiency whereas the converse would not be true.

This "balance gap" between time-balance and energy-balance also has an important impact on work-communication trade-off. That is, if it were possible to reduce communication by a factor of m in exchange for increasing computation (FLOPs) by a factor of f, how much energy we can save is bounded by this balance gap. We define $E_{1,1}$ as the energy consumed by our baseline algorithm and $E_{f,m}$ as the energy consumed by our "new" algorithm. Then, we can define

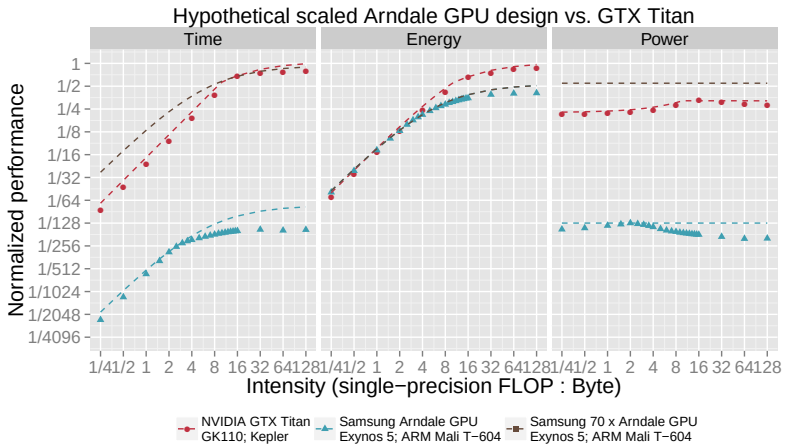

Figure 2: Comparison of the time-efficiency (performance), energy-efficiency, and power required by a mobile GPU (from an "Arndale" Samsung Exynos 5 developer board) versus high-end gaming-grade desktop GPU (NVIDIA GTX Titan), over a range of synthetic computations with varying computational intensities (FLOP:Byte). Our analysis suggests that combining 70 of the mobile GPUs can, relative to the desktop GPU, outperform in time, match in energy, at only a small ($2\times$) increase in power when the computation is sufficiently memory bound. Otherwise, the desktop GPU is a better building block.

"greenup" as $\Delta E = E_{1,1}/E_{f,m}$. Our research indicates that to see a greenup, the factor by which we can increase the number of FLOPs is bounded by the balance gap, B_ϵ/B_τ. Since bigger balance gap indicates more "slack" for increasing the number of FLOPs, the current trend of an increasing gap between time- and energy-balances implies that saving energy through work-communication trade-off will become "easier" in the future.

3 Candidate HPC building blocks

Secondly, we conduct a microbenchmarking study of the time, energy, and power of computation and memory access of several candidate compute-node building blocks of future high-performance computing systems. We include a dozen server-, desktop-, and mobile-class platforms that span a wide range of compute and power characteristics. Our modeling and analysis exercise permits a precise analytical characterization of abstract algorithmic regimes where one building block may be preferable to others.

We can summarize our results using a simple example. Suppose we wish to know whether overall time, energy, and power to compute would be better if the system building block is a high-end desktop GPU or a low-end, low-power, mobile GPU. Specifically, consider an NVIDIA GTX Titan, rated at 5 TFLOP/s peak in single precision and 250 Watts against an Arndale development board based on the Samsung Exynos 5 system-on-chip rated at 72 GFLOP/s and 10 Watts.

Figure 2 illustrates our analysis. While the GTX Titan is much faster in absolute performance, in energy-efficiency, Arndale compares well to it over a range of intensity values: the two systems match in FLOPs per Joule for intensities as high as 4 FLOP:Byte. And even at more compute-bound intensities, the Arndale is within a factor of two of the GTX Titan in energy-efficiency despite being about 70 times slower. That suggests that 70 Arndales together could *match* the GTX Titan on peak performance for compute-bound codes, while delivering up to 3–4$\times$ increase in memory bandwidth. This scenario is shown by the hypothetical dashed brown line.

References

[1] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009. [Online]. Available: http://doi.acm.org/10.1145/1498765.1498785

[2] J. Choi, R. Vuduc, R. Fowler, and D. Bendard, "A roofline model of energy," in *In Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS 13)*, 2013.

[3] J. Choi and R. Vuduc, "A roofline model of energy," Georgia Institute of Technology, Tech. Rep. GT-CSE-2012-01, December 2012.