[ExaMath] Can algorithms inform architecture?

Richard Vuduc, Georgia Institute of Technology

April 30, 2013

A position based on the ideas of Kenneth Czechowski, Jee Whan Choi, and Aparna Chandramowlishwaran (MIT).

richie@cc.gatech.edu
vuduc.org | hpcgarage.org

In 1985, an iconclastic computer architect, W. Daniel Hillis, won the Association of Computing Machinery's Doctoral Dissertation Award for his work on the Connection Machine.[1] No stranger to controversy, he entitled his concluding chapter,

> "New computer architectures and their relationship to physics, or, **why computer science is no good.**"

[1] W. Daniel Hillis. *The Connection Machine*. PhD thesis, Massachusetts Institute of Technology, 1985

Hillis's critique was that parallel algorithm design at the time was too abstract, ignoring fundamental physical limitations of real machines. Left unchecked, there would be two unfortunate consequences. First, algorithms would tend to perform poorly on real machines. Secondly, architects would never learn how to build better machines, having no real insight into what algorithms needed.

Arguably, these consequences persist. On "cooperative design" of exascale systems, it is largely computer scientists, and not the applied mathematicians responsible for shepherding (parallel) algorithms, who run the show. Perhaps it is better this way. Or, perhaps it is time for applied mathematicians to *lead*. Indeed, that is our position, and we say so speaking primarily as computer scientists.

Considering Hillis's critique, exascale hardware analysis suggests we try to account for the *energy* and *power* of a computation.[2] Indeed, we are studying how to do so.[3] Our starting point is simple: just as flops and communication have costs in time, we may ascribe to them costs in energy.[4] We seek models parsimonious enough to use, but rich enough to make "useful" predictions.

[2] Peter Kogge et al. Exascale Computing Study: Technology challenges in acheiving exascale systems, September 2008. URL http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf

[3] Jee Choi, Dan Bedard, Rob Fowler, and Richard Vuduc. A roofline model of energy. In *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013. http://vuduc.org/pubs/choi2013-archline-ipdps.pdf

[4] By proxy, we can then reason about the average power of a computation through Power = Energy $\times$ Time.

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. The NVIDIA GTX Titan can achieve 5 TFLOP/s peak in single-precision within 250 Watts thermal design power (TDP) envelope. The GPU of a mobile Arndale system, based on the Samsung Exynos 5 system-on-chip (SoC) processor, delivers 72 GFLOP/s at 10 Watts TDP. The Titan achieves 20 GFLOP/s per Watt vs. 7.2 for the Arndale, yet, there are active efforts to build supercomputers from the latter.[5] Which is "correct?"

[5] The Mont Blanc Project, for instance: http://www.montblanc-project.eu/

Of course, "it depends"—but the right algorithm analysis may offer a more precise and pointed answer. Figure 1 shows one such analysis. "Algorithms" appear abstractly by their intrinsic computational intensity (FLOP : Byte), along the x-axis. The y-axes compare

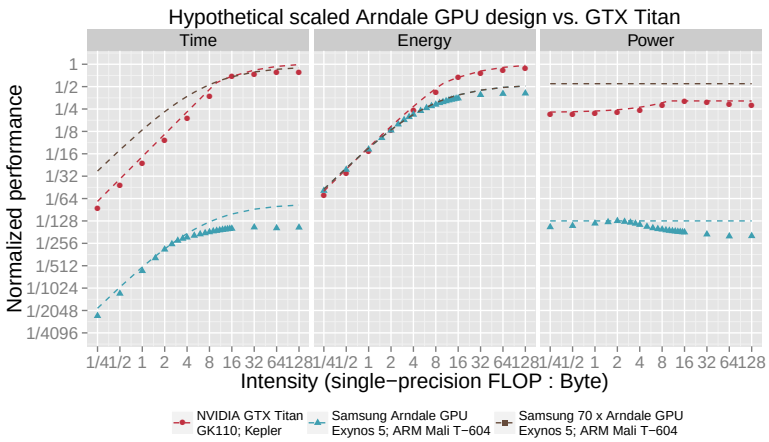

Figure 1: Comparison of the time-efficiency (performance), energy-efficiency, and power required by a mobile GPU (from an "Arndale" Samsung Exynos 5 developer board) versus high-end gaming-grade desktop GPU (NVIDIA GTX Titan), over a range of synthetic computations with varying computational intensities (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×) increase in power when the computation is sufficiently memory bound. Otherwise, the desktop GPU is a better building block.

time-efficiency (flops per unit time), energy-efficiency (flops per unit energy), and power (energy per unit time) of the two platforms. The dots are microbenchmark measurements and the dashed lines our model's prediction—limited, but with a good correspondence.

Most interesingly, this analysis reveals that the platforms *match* in energy-efficiency for intensities as high as 4 FLOP:Byte;[6] at higher intensities, the weaker Arndale is within 2× the GTX Titan in energy-efficiency despite having $1/70^{\text{th}}$ the peak performance. The dashed brown line is a hypothetical prediction, based on the model, of 70 Arndales stitched together: it could *match* the GTX Titan on peak for compute-bound codes while delivering up to 3–4× Titan's performance for memory bandwidth-bound codes. The price is a doubling of power, as well as interconnect costs, which this analysis ignores. Nevertheless, it quantifies the potential and at the very least suggests a precise and analytical way to compare these as building blocks, with algorithmic properties—like intensity—driving the analysis.

[6] A sparse matrix-vector multiply is roughly 0.25–0.5 FLOP:Byte in single-precision and a fast Fourier transform 2–4 FLOP:Byte.

THE GOAL OF CO-DESIGN should be to say, in broad but also quantitative terms, how macroscopic changes to an architecture might affect the execution time, scalability, accuracy, and power-efficiency of a computation; and, conversely, identify what classes of computation might best match a given architecture. The preceding demonstration maintains the algorithm abstractly. But many trade-off analyses become possible with the right model of cost. We have shown elsewhere, for instance, the notion of *algorithmic power scaling*: given different classes of algorithms, which scale better when more power is injected into the system?[7] It is respect to these high-level questions that we believe applied mathematicians, who actively engage in parallel algorithm design, might offer the most interesting answers.

[7] Kenneth Czechowski and Richard Vuduc. A theoretical framework for algorithm-architecture co-design. In *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013. `http://vuduc.org/pubs/czechowski2013-codesign-ipdps.pdf`

References

Jee Choi, Dan Bedard, Rob Fowler, and Richard Vuduc. A roofline model of energy. In *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013. http://vuduc.org/pubs/choi2013-archline-ipdps.pdf.

Kenneth Czechowski and Richard Vuduc. A theoretical framework for algorithm-architecture co-design. In *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013. http://vuduc.org/pubs/czechowski2013-codesign-ipdps.pdf.

W. Daniel Hillis. *The Connection Machine*. PhD thesis, Massachusetts Institute of Technology, 1985.

Peter Kogge et al. Exascale Computing Study: Technology challenges in acheiving exascale systems, September 2008. URL http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf.