Algorithmic time, energy, and power on candidate HPC compute building blocks

Jee Choi, Marat Dukhan, Xing Liu, Richard Vuduc
School of Computational Science and Engineering
Georgia Institute of Technology
Atlanta, Georgia, 30332-0765, USA
{jee,mdukhan3,xing.liu,richie}@gatech.edu

***Abstract*—We conducted a microbenchmarking study of the time, energy, and power of computation and memory access on several existing platforms. These platforms represent candidate compute-node building blocks of future high-performance computing systems. Our analysis uses the "energy roofline" model, developed in prior work, which we extend in two ways. First, we improve the model's accuracy by accounting for power caps, basic memory hierarchy access costs, and measurement of random memory access patterns. Secondly, we empirically evaluate server-, mini-, and mobile-class platforms that span a range of compute and power characteristics. Our study includes a dozen such platforms, including x86 (both conventional and Xeon Phi), ARM, GPU, and hybrid (AMD APU and other SoC) processors. These data and our model analytically characterize the range of algorithmic regimes where we might prefer one building block to others. It suggests critical values of arithmetic intensity around which some systems may switch from being more to less time- and energy-efficient than others; it further suggests how, with respect to intensity, operations should be throttled to meet a power cap. We hope our methods can help make debates about the relative merits of these and other systems more quantitative, analytical, and insightful.**

***Index Terms*—energy; power; algorithms; system balance; performance modeling**

I. Introduction

We consider the problem of estimating how much time, energy, and power an abstract algorithm may require on a given machine. Our approach starts with an abstract cost model grounded in first principles of algorithm design. The model's utility derives from the way it facilitates quick and precise reasoning about *potential* time-efficiency, energy-efficiency, and power-efficiency. This paper applies the model to analyze candidate compute-node building blocks being considered for emerging and future HPC systems, which include high-end server and GPU platforms as well as low-end, low-power mobile platforms.

Importantly, beyond specific findings and data, we emphasize the methodological aspects of this paper. In particular, architects may find our high-level approach to be a useful additional way to assess systems across computations; our analysis technique aims to provide more insight than a collection of blackbox benchmarks provides but without having to know too much detail about specific computations. Similarly, we hope algorithm designers may find ways to reason about algorithmic techniques for managing energy and power, and tradeoffs (if any) against time.

A. Demonstration

As a quick preview, 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 the desktop-class NVIDIA GTX Titan against the "Arndale GPU," the on-chip GPU component of the Samsung Exynos 5 mobile processor. The GTX Titan has a peak performance of 5 trillion floating-point operations per second (5 Tflop/s) in single-precision and 250 Watts thermal design power (TDP) for the whole card; the Arndale GPU has a 72 Gflop/s peak and its standard developer board uses less than 10 Watts. These specifications suggest GTX Titan is better, based on its considerably higher flop/s per Watt. Yet, there are also active efforts to build systems from close equivalents to the latter.[1] Which is "correct?"

While a natural response is, "it depends," our model offers a more precise analysis, which fig. 1 summarizes. It compares time-efficiency (performance, or operations per unit time), energy-efficiency (operations per unit energy), and power (energy per unit time) of the two platforms. The y-axis measures this performance on a normalized scale. (The power and energy costs include the entire board, including memory and on-board peripherals, but *excluding* any host system.) The x-axis abstracts away a possible computation by its operational intensity, or the ratio of computation-to-communication (flop:Byte ratio). Decreasing values of intensity indicate increasing memory bandwidth boundedness. The dots are measured values from a synthetic microbenchmark (§IV); the dashed lines indicate our model's predictions. The model and measurements correspond well.

While the GTX Titan is much faster, in energy-efficiency the Arndale GPU compares well to it over a range of intensities: the two systems match in flops per Joule (flop/J) for intensities as high as 4 flop:Byte. For reference, a large sparse matrix-vector multiply is roughly 0.25–0.5 flop:Byte in single-precision and a large fast Fourier transform (FFT) is 2–4 flop:Byte [2]. And even at more compute-bound intensities, the Arndale is within a factor of two of the GTX Titan in energy-efficiency despite its much lower peak. It therefore appears to be an attractive candidate.

[1] E.g., The Mont Blanc Project: http://www.montblanc-project.eu/ [1].

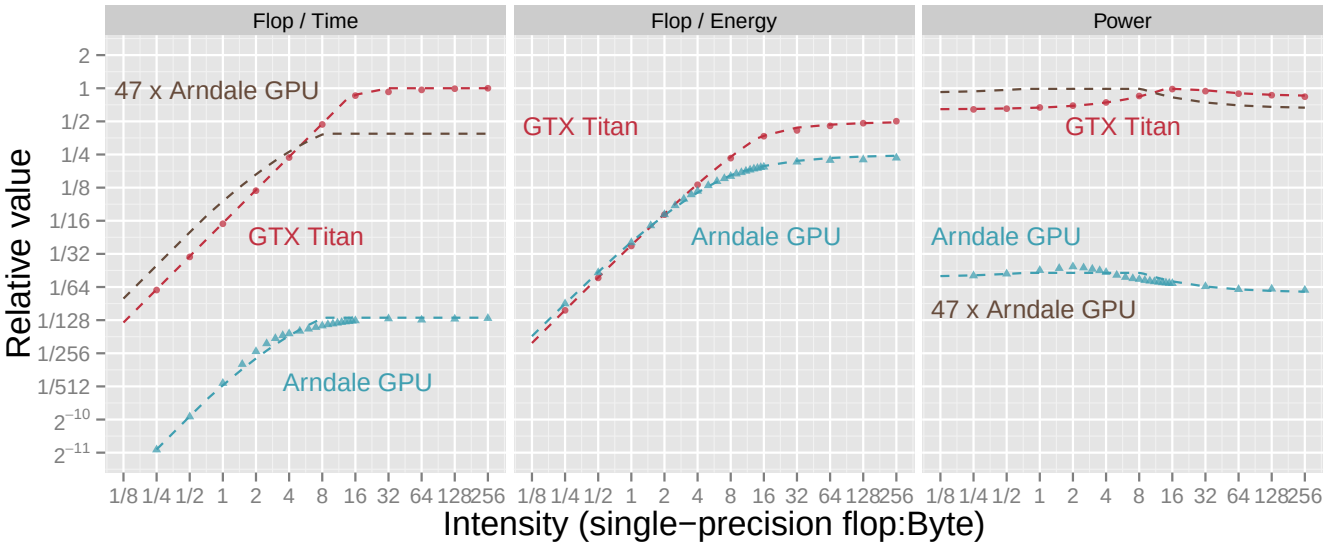


Fig. 1: Comparison of the time-efficiency (performance), energy-efficiency, and power required by a mobile GPU (from an "Arndale" Samsung Exynos 5 developer board) versus high-end gaming-grade desktop GPU (NVIDIA GTX Titan), over a range of synthetic computations with varying computational intensities (flop:Byte). Combining 47 of the mobile GPUs to match on peak power can lead to a system that outperforms the desktop GPU by up to 1.6× for relatively bandwidth-bound codes (flop:Byte less than 4), but at the cost of sacrificing peak performance (less than ½) for compute-bound codes.

From these data, what is the best-case scenario for an Arndale GPU-based "supercomputer," assembled from enough Arndale GPUs to match the GTX Titan in peak *power*? Matching on power may require up to 47 Arndale GPUs, yielding the hypothetical system shown by a dashed brown line. This system would have less than half of the GTX Titan's peak, but would also have an aggregate memory bandwidth that is up to 1.6× higher for intensities up to about 4 flop:Byte, which could include, for instance, a large multidimensional FFT. However, this best-case ignores the significant costs of an interconnection network, or further potential improvements to the Arndale GPU system by better integration. As such, the 47 Arndale GPUs are more likely to improve upon GTX Titan only marginally or not at all across the full range of intensities. Regardless of one's interpretation, this type of analysis offers an analytical way to compare these as building blocks.

B. Summary of contributions and limitations

Overall, we claim two main contributions.

The first is our abstract model. It extends our previously proposed *energy roofline* model [3], adding several important components. These include explicit modeling of a power cap and modeling of cache energy costs. The power cap is especially significant, as it implies a way to *predict* power-throttling requirements. That is, if we wish to keep average power below some threshold, the model predicts by how much flops and memory operations should slow down.

Secondly, we compare the model to measurements on 12 platforms. These include x86 (both conventional and Xeon Phi), several flavors of ARM, desktop and mobile GPUs, and accelerated processing units (or APUs, such as those from AMD APU and other SoCs manufacturers). These experiments validate the basic form of the model. Furthermore, they yield empirical estimates of the effective energy required to perform flops and to move data. Breaking down the energy costs to these different components allows us to consider energy-efficiency at different intensity points, allowing more flexibility and analytical precision than simply dividing, say, peak performance by TDP. Moreover, these basic estimated constants may in and of themselves be useful reference values.

The main limitation of our work is that, to permit parsimonious analysis, we have kept our model simple, with many extensions left as future work. To mitigate this limitation, we have publicly released all of the code and documentation of our experimental setup as part of publication of this work.[2] This code includes finely tuned microbenchmarks in an array of programming environments, including assembly, SIMD intrinsics, CUDA, and OpenCL.

II. Related Work

This paper is a natural follow-up to our original work [3], [4] where we derive and partially validate an earlier model inspired by the roofline-style analysis [2], [5], [6]. It also contains a more extensive literature review than the one we include below; we refer interested readers thereto.

Since that survey, Kestor et al. have developed a microbenchmarking methodology similar to our own [7]. In

[2] http://hpcgarage.org/archline

particular, they focus on measuring energy costs due to the memory hierarchy and communication. Our modeling approach differs, with power caps being its most notable distinction.

In this paper, we extend our targets to low-power systems, such as those based on ARM and APUs. We consider both CPU and GPU architectures and use our model to test their viability for high performance computing. There are numerous other studies of low-power systems for HPC that include ARM, Atom, and Freescale processors, including the active Mont Blanc project [1] and for numerical and graph problems [8]–[10]. These studies focus on specific systems or workloads. By contrast, our paper tries to abstract the computation away to reason about many possible workloads simultaneously and analytically.

There is an additional trend to investigate special-purpose hardware [11]–[13], including approximate computing [14]. These are promising, forward-looking approaches that break the usual computing paradigms. Our study and its conclusions are limited to general-purpose off-the-shelf building blocks, and therefore addresses nearer-term design exploration and experiments.

We use PowerMon 2 to measure power [15]. However, there are several other options, including PowerPack, a hardware-software "kit" for power and energy profiling [16]; and hardware counter-based measurement methods available on some Intel and NVIDIA systems [17], [18].

Other efforts to model power dissipation include the CACTI, McPAT, and GPUWATTCH modeling tools [19]–[21]. The underlying models derive from device-level estimates of power dissipation for caches and processor cores. The tools enable design-space exploration by quantifying the cost of new features and materials over different process technology generations. However, CACTI and McPAT are only validated against other simulators or against a breakdown of maximum thermal design power (TDP) published by the vendors. GPUWATTCH validates against real measurements, but only on NVIDIA GPUs. Therefore, an interesting question may be to what extent these tools corroborate, complement, or contradict our experimental data and modeling approach.

III. A First-Principles Model

Our model of time, energy, and power assumes the abstract von Neumann architecture of fig. 2. The system comprises a processor attached to a fast memory of finite capacity (Z words), which is then attached to an infinite slow memory. The fast memory is effectively a last-level cache and may be generalized in the presence of a memory hierarchy. An abstract algorithm running on this machine executes $W = W(n)$ flops and transfers $Q = Q(n; Z)$ bytes of data between slow and fast memory, given an input of size n. (In what follows, we suppress the arguments n and Z unless needed explicitly.[3]) Below, we describe how we estimate time, energy, and power

[3]If flops are not the natural unit of work, one could imagine substituting, for instance, "comparisons" for sorting or "edges traversed" in a graph traversal computation.

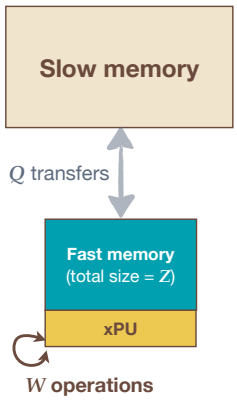


Fig. 2: A simple von Neumann architecture with a two-level memory hierarchy. In our first analysis, suppose that an algorithm performs W arithmetic operations and Q memory operations, or "mops," between slow and fast memories. The figure is taken from our prior work [3], [4].

for the abstract algorithm running on this abstract machine. The model derives partly from our earlier work [3], [4]; here, we highlight which features of this paper's model are new.

a) Cost model: Let the abstract machine be described by four fundamental time and energy costs: the time per flop, τ_{flop}; the time per byte, τ_{mem}; the energy per flop, ϵ_{flop}; and the energy per byte, ϵ_{mem}. Time and energy have units of, for instance, seconds (s) and Joules (J), respectively. In our abstract model, we do not take τ_{flop} and τ_{mem} to be latency costs; rather, we will use throughput values based on peak flop/s and peak memory bandwidth, respectively. That is, these costs are optimistic.

They also imply *power* costs, in units of energy per unit time. These are the peak power per flop, $\pi_{\text{flop}} \equiv \epsilon_{\text{flop}}/\tau_{\text{flop}}$, and the peak power per byte, $\pi_{\text{mem}} \equiv \epsilon_{\text{mem}}/\tau_{\text{mem}}$.

In addition, our abstract machine will require a minimum amount of *constant power*, π_1. This power is what the machine requires independent of what operations are executing. In contrast to other notions of "static power" in the literature, constant power in our model may *include* the power of other system components and peripherals, taken together.

b) Modeling execution energy: To estimate an algorithm's energy cost, we tally the total energy to execute all flops, to move the full volume of data, and to run given the cost of constant power. That is, the total energy E is

$$E = E(W, Q) \equiv W\epsilon_{\text{flop}} + Q\epsilon_{\text{mem}} + \pi_1 T(W, Q), \qquad (1)$$

where $T(W, Q)$ is the total running time of the computation, estimated below. The basic form of eq. (1) is identical to our earlier energy model [3], [4].

We will sometimes consider another form of eq. (1), parameterized by *intensity*, $I \equiv W/Q$, and *energy balance*, $B_\epsilon \equiv \epsilon_{\text{mem}}/\epsilon_{\text{flop}}$. Both these quantities have units of flops per Byte, with I expressing an intrinsic property of the algorithm and B_ϵ expressing an intrinsic property of the machine with

respect to energy. From these definitions, eq. (1) becomes

$$E = E(W, I) = W\epsilon_{\text{flop}}\left(1 + \frac{B_\epsilon}{I}\right) + \pi_1 T\left(W, \frac{W}{I}\right). \quad (2)$$

Equation (2) clarifies that the total energy, relative to the minimum energy $W\epsilon_{\text{flop}}$ to execute the flops alone, increases with increasing energy balance, decreases with increasing intensity, and increases with relative increases in constant power or other time inefficiencies.

c) Modeling execution time: Executing W flops takes $W\tau_{\text{flop}}$ time and moving Q bytes takes $Q\tau_{\text{mem}}$ time. In the best case, we may maximally overlap flops and memory movement, in which case T will be the maximum of these two values. Indeed, this definition was exactly our previous model [3].

Here, we extend our model of T to include *power caps*. Our previous model sometimes over-predicted performance and average power [3]. For some GPU and low-power systems we consider in this paper, which go beyond the original work, ignoring a potential power cap can be severe (§V).

We model a power cap as follows. Suppose that on top of the π_1 constant power, the system has $\Delta\pi$ additional units of *usable power* to perform any operations. The parameter $\Delta\pi$ is now a new fundamental parameter of the system. Thus, an algorithm limited only by $\Delta\pi$ will require no more than $(W\epsilon_{\text{flop}} + Q\epsilon_{\text{mem}})/\Delta\pi$ time to execute. Then, the best-case execution time is

$$T = T(W, Q) \equiv \max\left(W\tau_{\text{flop}}, Q\tau_{\text{mem}}, \frac{W\epsilon_{\text{flop}} + Q\epsilon_{\text{mem}}}{\Delta\pi}\right). \quad (3)$$

That is, if there is enough usable power to run at peak operational or memory performance, we do so assuming maximal overlap; otherwise, we must throttle all operations, which the third term of $\max$ captures. We may also rewrite eq. (3) as,

$$T = T(W, I) = W\tau_{\text{flop}} \max\left\{1, \frac{B_\tau}{I}, \frac{\pi_{\text{flop}}}{\Delta\pi}\left(1 + \frac{B_\epsilon}{I}\right)\right\}, \quad (4)$$

where $B_\tau \equiv \tau_{\text{mem}}/\tau_{\text{flop}}$ is the *time balance* of the system. This value is more commonly referred to as the intrinsic flop-to-Byte ratio of the machine, and defines the intensity at which the time to execute flops and time to execute memory operations are equal.

d) Modeling power: Given models of E and T, we may model average instantaneous power as $\bar{P} \equiv E/T$.

There are many ways to expand this definition, one of which we present here. Let

$$B_\tau^+ \equiv B_\tau \max\left(1, \frac{\pi_{\text{mem}}}{\Delta\pi - \pi_{\text{flop}}}\right) \quad (5)$$

$$\text{and } B_\tau^- \equiv B_\tau \min\left(1, \frac{\Delta\pi - \pi_{\text{mem}}}{\pi_{\text{flop}}}\right). \quad (6)$$

Observe that $B_\tau^- \le B_\tau \le B_\tau^+$. When $\Delta\pi \ge \pi_{\text{flop}} + \pi_{\text{mem}}$, there is enough usable power to run flops and move data at their maximum rates, in which case $B_\tau^+ = B_\tau^- = B_\tau$. Otherwise, $[B_\tau^-, B_\tau^+]$ defines an interval containing B_τ. From these definitions, one can show that $\bar{P}$ becomes

$$\bar{P} = \bar{P}(I) = \pi_1 + \begin{cases} \pi_{\text{flop}} + \pi_{\text{mem}}\frac{B_\tau}{I} & \text{if } I \ge B_\tau^+ \\ \pi_{\text{flop}}\frac{I}{B_\tau} + \pi_{\text{mem}} & \text{if } I \le B_\tau^- \\ \Delta\pi & \text{otherwise} \end{cases}. \quad (7)$$

Equation (7) reflects what we might expect. As I increases beyond B_τ^+ toward infinity, $\bar{P}$ decreases toward flop-only power, π_{flop}. Similarly, as I decreases away from B_τ^- toward 0, $\bar{P}$ decreases toward memory-only power, π_{mem}. The peak power occurs when $B_\tau^- \le I \le B_\tau^+$. In particular, when there is enough usable power, meaning $\Delta\pi \ge \pi_{\text{flop}} + \pi_{\text{mem}}$, then $\bar{P}(I)$ peaks at the value, $\pi_1 + \pi_{\text{flop}} + \pi_{\text{mem}}$, at $I = B_\tau$; otherwise, the power cap dominates and $\bar{P}(I) = \pi_1 + \Delta\pi$ for all $B_\tau^- \le I \le B_\tau^+$.

IV. Experimental Setup

We benchmarked and assessed the nine systems shown in table I. For the four discrete coprocessors—NVIDIA GTX 580, 680, Titan, and Intel Phi—we consider just the card itself and ignore host power and host-to-coprocessor transfer costs. Additionally, three of the systems—labelled "NUC," "Arndale," and "APU"—have hybrid CPU+GPU processors. We considered their CPU and GPU components separately, i.e., running no or only a minimal load on the other component. As such, we claim to evaluate twelve "platforms."

For each platform, we wrote an architecture-specific hand-tuned microbenchmark that gets as close to the vendor's claimed peak as we could manage. These microbenchmarks measure "sustainable peak" flop/s, streaming bandwidth (from main memory, L1, and L2 caches where applicable), and random main memory access. The measured values appear parenthetically in columns 8–10 of table I, and should be compared against the theoretical peaks shown in columns 3–5. For cache bandwidth, we were not able to determine theoretical peaks on all platforms; therefore, we report only measured values in columns 11 and 12. Lastly, we fitted our model to the microbenchmark data. The model parameters appear as columns 6–13 of table I.

e) Intensity microbenchmark: The intensity microbenchmark varies intensity nearly continuously, by varying the number of floating point operations (single or double) on each word of data loaded from main memory. We hand-tuned these microbenchmarks for each platform. Examples of specific tuning techniques we used include unrolling, to eliminate non-flop and non-load/store overheads that might otherwise distort our energy estimates; use of fused-multiply adds where available; tuning the instruction selection and instruction mix, carefully considering pipeline and issue port conflicts; prefetching; and resorting to assembly where needed; to name a few. We test single- and double-precision operations separately; their energy costs appear as ϵ_{s} and ϵ_{d}, respectively.

f) Random access microbenchmark: Our random access microbenchmark implements pointer chasing, as might appear in a sparse matrix or other graph computation. It fetches data from random places in the memory rather than streaming the data, reporting sustainable accesses per unit time. By its

Column 1	2	3	4	5	6	7	8	9	10	11	12	13
		Vendor's claimed peak			Power (empirical)		Energy (and empirical throughput)					Random access
Platform	Processor	single Gflop/s	double Gflop/s	mem. bw. GB/s	π_1 Watts (idle)	$\Delta\pi$ Watts	ϵ_s pJ/flop (Gflop/s)	ϵ_d pJ/flop (Gflop/s)	ϵ_{mem} pJ/B (GB/s)	ϵ_{L1} pJ/B (GB/s)	ϵ_{L2} pJ/B (GB/s)	ϵ_{rand} nJ/access (Macc/s)
Desktop CPU "Nehalem"	Intel Core i7-950 (45 nm)	107	53.3	25.6	122 (79.9)	44.2	371 (99.4)	670 (49.7)	795 (19.1)	135 (201)	168 (120)	108 (149)
NUC CPU "Ivy Bridge"	Intel Core i3-3217U (22 nm)	57.6	28.8	25.6	16.5 (13.2)	7.37	14.7 (55.6)	24.3 (27.9)	418 (17.9)	8.75 (201)	14.3 (103)	54.6 (55.3)
NUC GPU	HD 4000	269	—	25.6	10.1 (13.2)*	17.7	76.1 (268)	—	837 (15.4)	—	—	—
APU CPU "Bobcat"	AMD E2-1800 (40 nm)	13.6	5.10	10.7	20.1 (11.8)	1.39	33.5 (13.4)	119 (5.05)	435 (3.32)	84.0 (25.8)	138 (11.6)	75.6 (8.03)
APU GPU "Zacate"	HD 7340	109	—	10.7	15.6 (11.8)	3.23	5.82 (104)	—	333 (8.70)	6.47 (46.0)	—	45.8 (115)
GTX 580 "Fermi"	NVIDIA GF100 (40 nm)	1580	198	192	122 (148)*	146	99.7 (1400)	213 (196)	513 (171)	149 (761)	257 (284)	112 (977)
GTX 680 "Kepler"	NVIDIA GK104 (28 nm)	3530	147	192	66.4 (100)*	145	43.2 (3030)	263 (147)	437 (158)	51 (1150)	195 (297)	184 (1420)
GTX Titan "Kepler"	NVIDIA GK110 (28 nm)	4990	1660	288	123 (72.9)	164	30.4 (4020)	93.9 (1600)	267 (239)	24.4 (1610)	195 (297)	48.0 (968)
Xeon Phi "KNC"	Intel 5110P (22 nm)	2020	1010	320	180 (90)	36.1	6.05 (2020)	12.4 (1010)	136 (181)	2.19 (2890)	8.65 (591)	5.11 (706)
PandaBoard ES "Cortex-A9"	TI OMAP 4460 (45 nm)	9.60	3.60	3.20	3.48 (2.74)	1.19	37.2 (9.47)	302 (3.02)	810 (1.28)	79.5 (18.4)	134 (4.12)	60.9 (12.1)
Arndale CPU "Cortex-A15"	Samsung Exynos 5 (32 nm)	27.2	6.80	12.8	5.50 (1.72)	2.01	107 (15.8)	275 (3.97)	386 (3.94)	76.3 (50.8)	248 (15.2)	138 (14.8)
Arndale GPU "Mali T-604"		72.0	—	12.8	1.28 (1.72)*	4.83	84.2 (33.0)	—	518 (8.39)	71.4 (33.4)	—	125 (33.6)

TABLE I: Platforms summary, with 9 systems and 12 distinct "platforms." We distinguish between manufacturer's peak (columns 3–5) and "sustained peak" using our microbenchmarks, shown parenthetically in columns 8–10. *Note 1*: In four cases, denoted by an asterisk ("*"), our fitted constant power is less than observed idle power under no load. *Note 2*: Some data are missing. In particular, double-precision support is not available on all platforms; additionally, deficiences in the current OpenCL driver prevented some microbenchmarks from running on the Intel HD 4000 GPU.

nature, it cannot fully use the memory interface width or the prefetching units. Therefore, we expect poor performance compared to the system's bandwidth.

g) Cache microbenchmarks: Our cache microbenchmarks assess the performance and energy cost of accessing the different levels of the cache.

On CPU systems, these can be either the pointer chasing or the intensity benchmark, depending on which gives better performance. We need only ensure the data set size is small enough to fit into the target cache level.

GPUs have different memory hierarchy designs, which requires platform-dependent coding and tuning. On NVIDIA Fermi GPUs, we assess L1, L2 caches; on Kepler systems, we test L2 and shared memory, since the L1 cache is no longer used to store data and all data reuse has to be done manually via the shared memory. On AMD HD 7340 and ARM Mali-T604, we use the software-managed scratchpad memory.

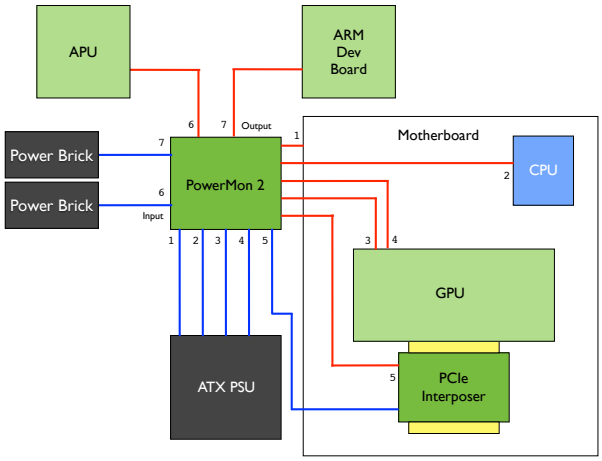


Fig. 3: Placement of the measurement probes, PowerMon 2 and our custom PCIe interposer

h) Power measurement infrastructure: Our power measurement setup appears in fig. 3. We use two tools to measure power. The first is Powermon 2, a fine-grained integrated power measurement device for measuring direct current (DC) voltage and current [15]. PowerMon 2 sits between any device and its DC source to intercept and measure the DC current and voltage. It samples at 1024 Hz per channel, with an aggregate frequency of up to 3072 Hz over 8 channels. It reports time-stamped measurements without the need for specialized software. It also fits in a 3.5-inch internal hard drive bay, which simplifies installation. The second measurement tool is a custom-made PCIe interposer that sits between the motherboard and the target PCIe device (i.e., GPU) to measure the power provided by the motherboard.

For the mobile systems, we measure system-level power, which includes CPU, GPU, DRAM, and peripherals. For CPU systems, we measure input both to the CPU and to the motherboard, which provides power to the DRAM modules. High-performance GPUs, such as the GTX Titan, require a more complicated setup. They draw power from multiple sources, including the motherboard via the PCIe connector and the 12 V 8-pin and 6-pin PCIe power connectors.

PowerMon 2 measures the current and voltage from each source at a regular interval and computes the instantaneous power by multiplying measured current and voltage. Assuming uniform samples, we compute the average power as the average of the instantaneous power over all samples. For systems that draw from multiple power sources, such as a GPU, we sum the average powers to get total power. Total energy is then the average power times the execution time.

V. Results and Discussion

We divide the analysis and discussion of our experiments into four parts. First, we compare the power-capped model of this paper with our prior model [3], [4], showing both qualitative and quantitative improvements (§V-A). Secondly, we explain our memory hierarchy measurement results (§V-B). Thirdly, we analyze the platforms in detail (§V-C), to show what one might conclude about their relative time-efficiency, energy-efficiency, and power characteristics. Lastly, we use the model to consider a variety of "what-if" scenarios, such as what we expect to happen under power throttling or power bounding (§V-D).

We focus on single-precision results, since full support for double is incomplete on several of our evaluation platforms. However, the interested reader can still find the main summary estimate of double-precision flop energy cost in table I.

A. Model fitting and accuracy

For each of the twelve platforms shown in table I, we ran our microbenchmark suite (§IV) at varying W and Q values, and measured total execution time and energy. These include runs in which the total data accessed only fits in a given level of the memory hierarchy. We then used (nonlinear) regression parameter fitting techniques to obtain statistically significant estimates of the values τ_{flop}, τ_{mem}, ϵ_{flop}, ϵ_{mem}, π_1, and $\Delta\pi$, as well as the corresponding parameters for each cache level where applicable.[4] The resulting set of parameters appears in columns 6–13 of table I.

To assess the model fits, we evaluated both our prior "uncapped" model [4] and our new "capped" model, eqs. (1)–(7). We compared them against the measured values, as follows. First, we calculated the relative error, $(\text{model} - \text{measured})/\text{measured}$, at each intensity value. Given a platform and model, we regard the set of errors over all intensity values as the error distribution. For each platform, we compared our prior and new models by comparing their distributions.

Figure 4 summarizes these error distributions. (We have similar data for time and energy, omitted for space.) Each platform appears on the x-axis, and the y-axis measures relative error. Each observed error value appears as a dot; the boxplots reflect the median, 25%, and 75% quantiles of the distribution of those errors. Platforms are sorted in descending order of median uncapped model's relative error. That is,

[4]The precise fitting procedure follows our prior work [4] and is part of our publicly released source code: http://hpcgarage.org/archline

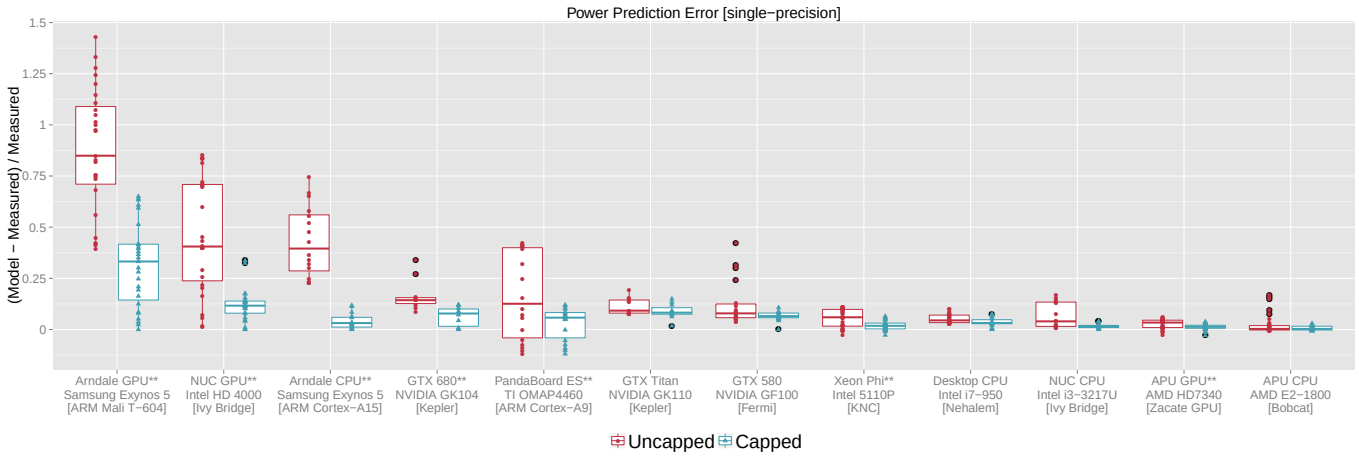


Fig. 4: Summary of modeling errors with respect to performance (FLOP/s). We compare the prediction errors of our prior model ("free" or uncapped; [3], [4]) against our new model ("capped"), which notably includes an explicit power cap. Qualitatively, the distribution of errors on all platforms improves, becoming either lower in median value or more tightly grouped. On platforms labeled by a double-asterisks ("**"), the free and capped distributions differ statistically (at $p < .05$) by the Kolmogorov-Smirnov test [22].

our original model does relatively better as we move from platforms on the left toward platforms on the right of fig. 4.

Qualitatively, fig. 4 shows that the new capped model tends to reduce the magnitude and spread of relative error compared to the previous model. The bias is to overpredict, i.e., most errors greater than zero. To facilitate a more rigorous quantitative comparison, we also performed the Kolmogorov-Smirnov nonparametric test of whether two empirical distributions differ, against a null hypothesis that the distributions come from the same underlying distribution. (The K-S test makes no assumptions about the distributions, such as normality, and so may be pessimistic.) Any platform for which the null hypothesis may be rejected (at a p-value less than 0.05) is marked with two asterisks ("**") in fig. 4. In this statistical sense, 7 of the 12 platforms–Arndale GPU, NUC GPU, Arndale CPU, GTX 680, PandaBoard ES, Xeon Phi, and APU GPU—the uncapped and capped error samples likely come from different distributions.

B. Interpreting memory hierarchy energy costs

Some care is needed to correctly interpret the memory hierarchy parameter estimates of table I. The key principle is that our energy cost estimates reflect *inclusive* costs. Alternatively, one should regard the energy cost of a memory hierarchy operation in our model as the *additional* energy required to complete *one additional* instance of that operation. The following examples clarify how this interpretation works.

The cost of loading a byte from DRAM (ϵ_{mem}) includes not only the costs of reading the byte from the DRAM cells and driving the wires, but also the energy spent by the memory controller as well as the cost of going through the memory hierarchy (e.g., the L1 and L2 caches). The rationale is that these costs are unavoidable whenever data moves between DRAM and registers. Note that we currently do not differentiate reads and writes, so consider ϵ_{mem} as the average of these costs. Also, in order to prevent the prefetcher from loading unused data, we have designed our microbenchmark to "direct" the prefetcher into prefetching only the data that will be used.

We define ϵ_{L1}, ϵ_{L2}, and ϵ_{rand} in a similar manner as ϵ_{mem}. The energy cost of loading data from the L2 cache (ϵ_{L2}) includes the energy consumed by reading data from the L2 memory cells as well as those consumed by reading from and writing data to the L1 cache as the data moves up the cache hierarchy. It also includes instruction overheads, as well as any other costs that might be involved such as the energy consumed by the cache coherency protocol. Naturally, we expect ϵ_{L1} to be smaller than ϵ_{L2} on the same system as they would most likely incur similar overheads, but fetching data from the L1 cache will not involve going through the L1 cache itself. This also serves as a way of sanity checking our model and as it can be seen in table I, $\epsilon_{\text{L1}} \leq \epsilon_{\text{L2}}$ for every system.

The energy cost of accessing memory at random locations (ϵ_{rand}) will include the cost of reading an entire cache line from the memory, as well as the usual overheads such as instruction, memory hierarchy, and associated protocols. As such, we expect this cost to be at least an order of magnitude higher than ϵ_{mem}, as table I reflects.

C. Constant power and power caps across platforms

There is a wide range of power behaviors across platforms, but also a narrow relative range within each platform. Refer to fig. 5, which shows the power of the twelve platforms from table I. It compares model (solid lines) to measurements (dots), and facilitates cross-platform comparisons across a full range of intensities (x-axis). The platforms are ordered from top-left to bottom-right in decreasing order of *peak energy-efficiency*, with the GTX Titan in the top-left at 16 Gflop/J

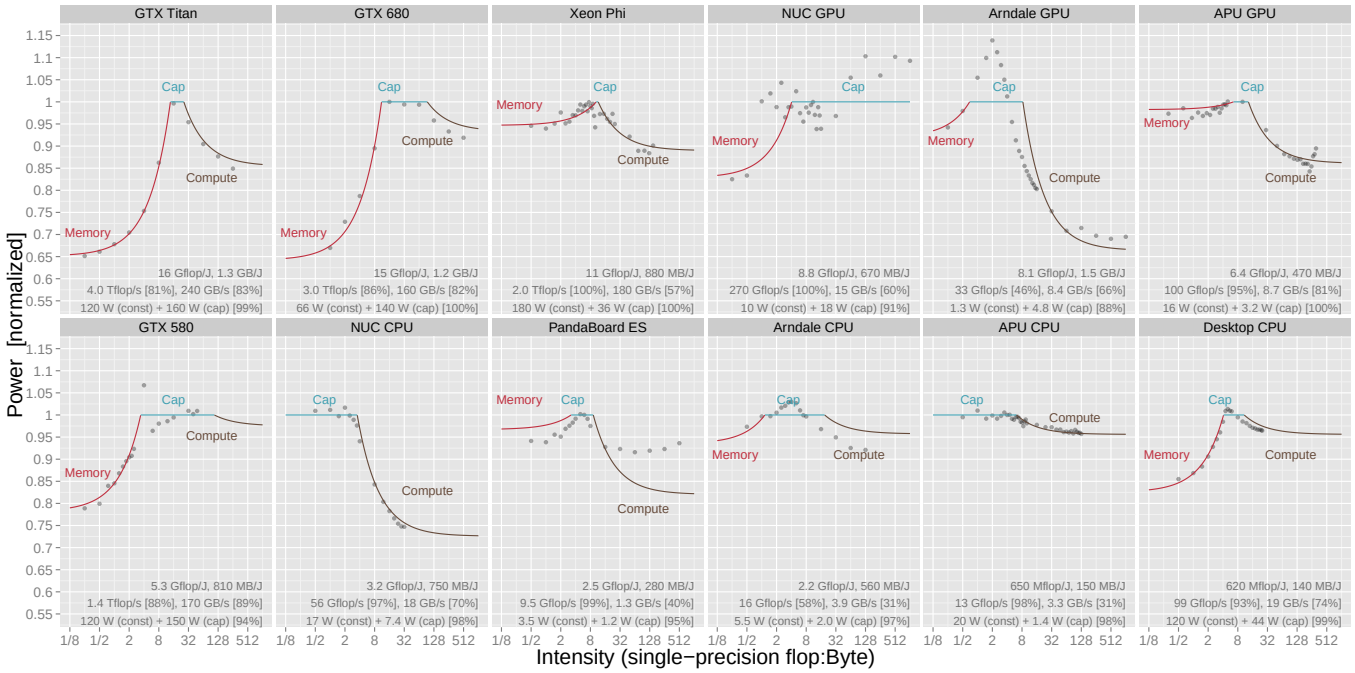


Fig. 5: Power (normalized). The y-axis is linear but the x-axis is logarithmic, base 2. The model appears as solid lines in up to three segments, indicating the three possible regimes: memory bandwidth-bound, power-bound due to capping, and compute-bound. Measurements appear as solid grey dots. Platforms appear from left-to-right and top-to-bottom in *decreasing* order of *peak energy-efficiency*, i.e., the most energy-efficient platform is the GTX Titan, whose peak is 16 Gflop/J.

and the Desktop CPU (Nehalem) at the bottom-right at just 620 Mflop/J.

Across platforms, power allocation between memory and processing differs. Flops on the GTX Titan consume more of the power budget than memory operations, while the Arndale GPU exhibits just the opposite behavior. However, there are no strongly discernable patterns in the architectural power allocations as energy-efficiency varies. The Arndale GPU, for example, has a peak energy-efficiency within a factor of two of the GTX Titan (8.1 Gflop/J vs. 16 Gflop/J), while putting much more of its power into the memory system. Consequently, its design yields on peak flop energy-efficiency to boost memory energy-efficiency, e.g., 1.5 Gflop/J on the Arndale GPU vs. 1.3 Gflop/J on GTX Titan.

But within a platform, the measurements vary only between the range of 0.65 to 1.15, or less than 2×. A narrow range means there is little room to reconfigure power to improve or adapt energy-efficiency to a computation. The main obstacle is the relatively large value of constant power, π_1, shown in table I. Indeed, the fraction of maximum power that π_1 consumes, or $\pi_1/(\pi_1 + \Delta\pi)$, is more than 50% for 7 of the 12 platforms in table I. As it happens, this fraction correlates with overall peak energy-efficiency, with a correlation coefficient of about -0.6 (not shown). Thus, driving down π_1 would be the key factor for improving overall system power reconfigurability.

Indeed, the mere fact of π_1 can invert our expectations. Consider a hypothetical workload that simply streams data from memory. How much energy does this computation use? Referring to table I, the Xeon Phi has the lowest ϵ_{mem} (136 pJ/B). It is lower than, for instance, the higher bandwidth GTX Titan (267 pJ/B) and lower-power Arndale GPU (518 pJ/B). However, we must also pay a constant energy charge of $\tau_{\text{mem}} \times \pi_1$, which adds 994 pJ/B to Xeon Phi, 515 pJ/B to GTX Titan, and just 153 pJ/B to the Arndale GPU. Then, the total energy per byte is 671 pJ/B for the Arndale GPU, 782 pJ/B for the GTX Titan, and 1.13 nJ/B to the Xeon Phi. This example underscores the critical role that π_1 plays.

The extent to which power capping, as we model it, affects the power characteristics of a given platform varies widely. Caps apply over a wider range of intensities on the hybrid NUC CPU+GPU and AMD APU platforms than they do on the GTX Titan and Xeon Phi. However, our approach to capping appears inaccurate on the NUC GPU and Arndale GPU. Although these mispredictions are always less than 15%, they raise questions about what mechanisms are operating. On the NUC GPU, as it happens measurement variability owes to OS interference.[5] However, on the Arndale GPU, the mismatch at mid-range intensities suggests we would need a different model of capping, perhaps one that *not* assume constant time and energy costs per operation. That is, even with a fixed clock frequency, there may be active energy-efficiency scaling with respect to processor and memory utilization.

[5]OpenCL drivers for NUC are available only when running Windows, which lacks easy user-level power management support.

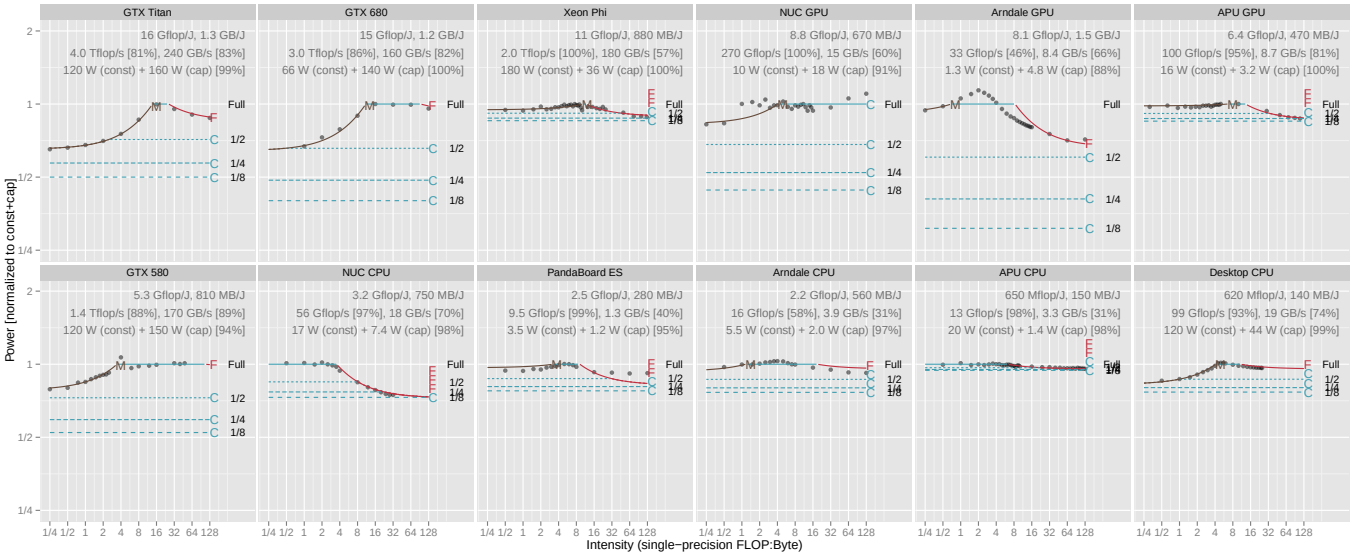


Fig. 6: Hypothetical power, performance, and energy-efficiency as the usable power cap ($\Delta\pi$) decreases. Note the log-log scales, base 2. Each curve represents a power cap setting: full refers to $\Delta\pi$ from table I and $1/k$ refers to a power cap setting of $\Delta\pi/k$, for $k \in \{2, 4, 8\}$. The curve color annotations, "F" for flop-bound (compute-bound), "C" for power cap-bound, and "M" for memory-bound.

D. Power throttling scenarios

Using the model, we may consider a variety of "what-if" scenarios related to power.

i) Power throttling: Suppose we lower $\Delta\pi$. What could the impact on maximum system power, performance, and energy-efficiency be, assuming all other model parameters, including π_1, remain equal?

Figure 6 shows power when the power cap is set to $\Delta\pi/k$, where $\Delta\pi$ is the original power cap (see table I) and $k \in [1, 8]$ is the reduction factor.

First, consider the extent to which reducing $\Delta\pi$ reduces overall system power. Figure 6 confirms that, owing to constant power $\pi_1 > 0$, reducing $\Delta\pi$ by k reduces overall power by less than k. It further shows that the Arndale GPU has the most potential to reduce system power by reducing $\Delta\pi$, whereas the Xeon Phi, APU CPU, and APU GPU platforms have the least. More node-level headroom—that is, low π_1 compared to $\Delta\pi$—may be very important, since it leaves more relative power for other power overheads, including the network and cooling.

Next, consider the extent to which reducing $\Delta\pi$ reduces performance. Figure 7a shows the variability across platforms and computational intensities. Highly memory-bound, low intensity computations on the GTX Titan degrade the least as $\Delta\pi$ decreases, since its design overprovisions power for compute. By contrast, for highly compute-bound computations, the NUC CPU degrades the least, since its design overprovisions power for memory. A similar observation holds for energy-efficiency, fig. 7b.

j) Power bounding: The preceding scenarios may have further implications for the idea of dynamic *power bounding*, which Rountree et al. have suggested will be a required mechanism of future systems [23].

Recall fig. 1 from §I, which compared GTX Titan and Arndale GPU building blocks. That analysis suggested that, as configured, an Arndale GPU building block would, even in the best case, offer only marginal improvements over GTX Titan, and would likely be worse.

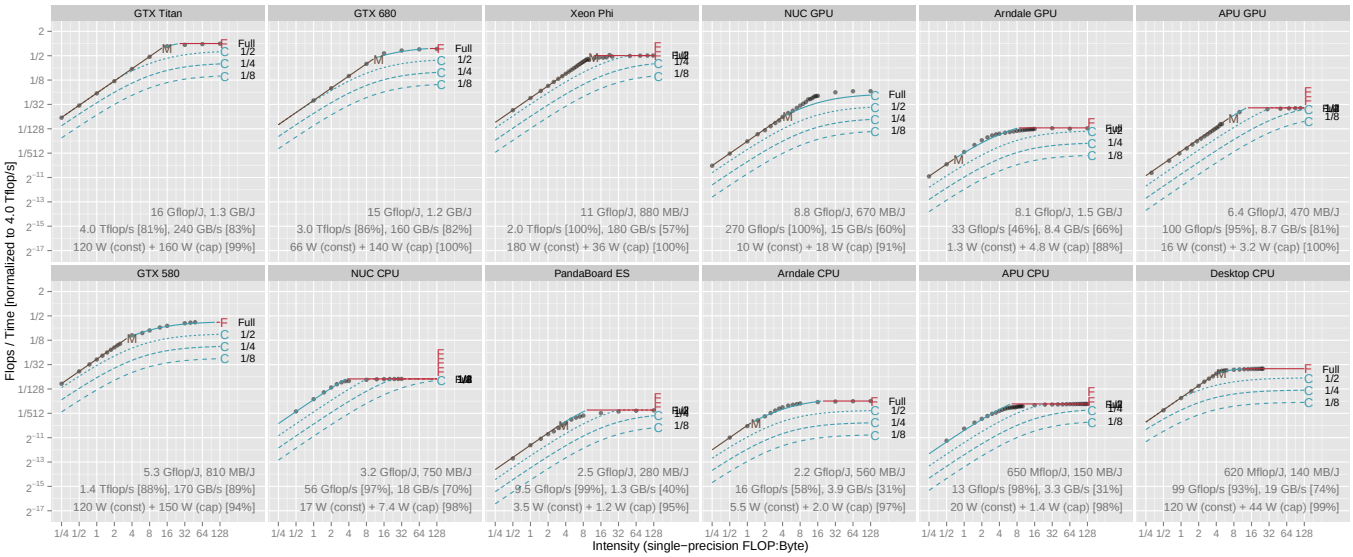
However, in a power bounding scenario, it might be necessary to reduce node power to a particular level. Suppose that, in a system based on GTX Titan nodes, it is necessary to reduce per-node power by half, to 140 Watts per node. This corresponds to a power cap setting of $\Delta\pi/8$ in fig. 6, which in turn will imply a performance of approximately $0.31\times$ at $I = 0.25$ relative to the default $\Delta\pi$. One can determine that, in the best case, assembling 23 Arndale GPUs will match 140 Watts but will be approximately $2.8\times$ faster at $I = 0.25$, which is better than the $1.6\times$ scenario from fig. 1. Essentially, a lower power grainsize, *combined* with a compute building block having a lower π_1, may lead to more graceful degradation under a system power bound.

VI. Conclusions and Future Directions

Our study adopts an algorithmic first-principles approach to the modeling and analysis of systems with respect to time, energy, and power. This approach can offer high-level analytical insights for current debates about the form of future HPC platforms. Our central example is the ability to consider—even in the model's relatively simple form—a variety of "what-if" scenarios, including what would happen if it became necessary to impose a power cap on any of the 12 evaluation platforms.

Constant power, π_1, is a critical limiting factor. It accounts for more than 50% of observed power on 7 of the 12 evalu-

(a) Performance (Gflop/s)



(b) Energy-efficiency (Gflop/J)

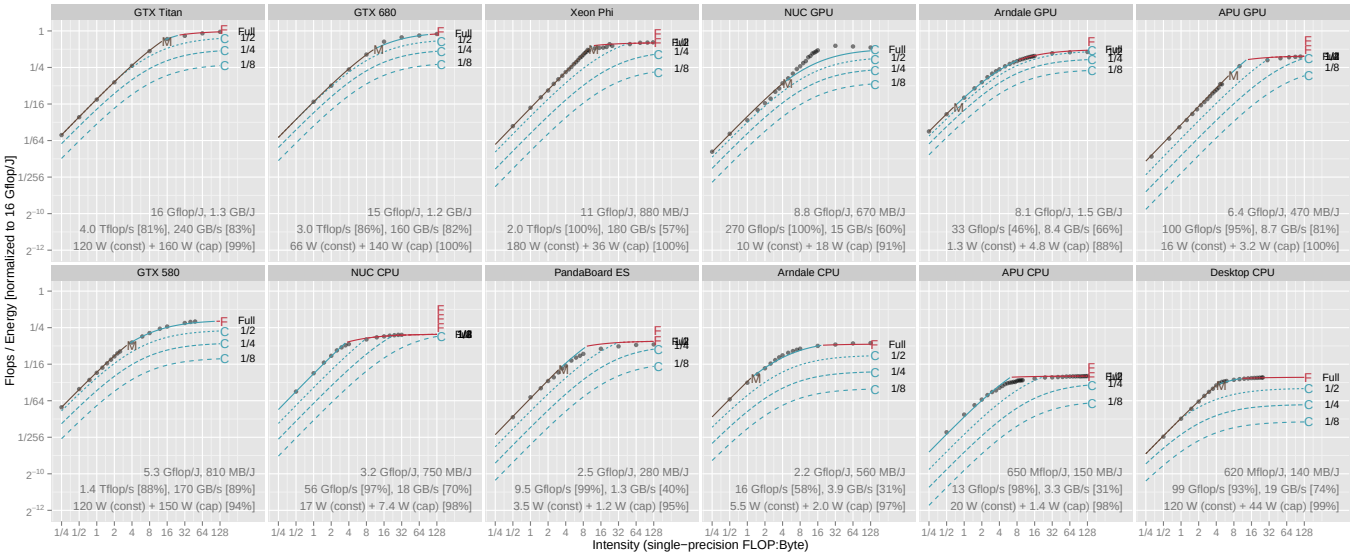


Fig. 7: Hypothetical performance and energy-efficiency as the usable power cap ($\Delta\pi$) decreases. This figure uses same plotting and notation conventions as fig. 6.

ation platforms. Its impact is to reduce the degree of power reconfigurability possible, and invert our expectations relative to the time and energy costs of primitive operations. These observations raise a natural question for device designers, architects, and system integrators: To what extent can π_1 be reduced, perhaps by more tightly integrating non-processor and non-memory components?

Beyond these observations, we hope the microbenchmarks and modeling methodology, as well as the parameters of table I, will prove useful to others. Indeed, table I is full of interesting data points, such as the fact that random memory access is on the Xeon Phi at least one order of magnitude less energy per access than any other platform, suggesting its utility on highly irregular data processing workloads.

The main limitations of this work are its many simplifying assumptions and its microbenchmark-only evaluation. Consequently, there may be a considerable gap between the best-case abstract analysis of this paper and actual applications. We are pursuing both more complex applications and more detailed models that can account for the observed errors (e.g., Arndale GPU) as part of our on-going work.

ACKNOWLEDGEMENTS

We thank Edmond Chow and Intel for providing access to Xeon Phi. Thanks also to Hyesoon Kim for the PCIe interposer. This work was supported in part by the National Science Foundation (NSF) under NSF CAREER award number 0953100; the U.S. Dept. of Energy (DOE), Office of Science, Advanced Scientific Computing Research under award DE-FC02-10ER26006/DE-SC0004915, and the Scientific Discovery through Advanced Computing (SciDAC) program under award DE-FC02-06ER25764 and DE-FG02-11ER26050/DE-SC0006925; and grants from the Defense Advanced Research Projects Agency (DARPA) Computer Science Study Group program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of NSF, DOE, or DARPA.

REFERENCES

[1] N. Rajovic, L. Vilanova, C. Villavieja, N. Puzovic, and A. Ramirez, "The low power architecture approach towards exascale computing," *Journal of Computational Science*, no. 0, pp. –, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750313000148

[2] 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

[3] J. Choi, D. Bedard, R. Fowler, and R. Vuduc, "A roofline model of energy," in *Proc. IEEE Int'l. Parallel and Distributed Processing Symp. (IPDPS)*, Boston, MA, USA, May 2013, this paper is a short peer-reviewed conference version of the following technical report: https://smartech.gatech.edu/xmlui/handle/1853/45737.

[4] J. W. Choi and R. Vuduc, "A roofline model of energy," Georgia Institute of Technology, School of Computational Science and Engineering, Atlanta, GA, USA, Tech. Rep. GT-CSE-12-01, December 2012. [Online]. Available: https://smartech.gatech.edu/xmlui/handle/1853/45737

[5] A. Ilic, F. Pratas, and L. Sousa, "Cache-aware roofline model: Upgrading the loft," *IEEE Computer Architecture Letters*, vol. 99, no. RapidPosts, p. 1, 2013.

[6] D. Antão, L. Taniça, A. Ilic, F. Pratas, P. Tomás, and L. Sousa, "Monitoring performance and power for application characterization with cache-aware roofline model," in *Proc. 10th Int'l. Conf. Parallel Processing and Applied Mathematics (PPAM)*, September 2013.

[7] G. Kestor, R. Gioiosa, D. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Workload Characterization (IISWC), 2013 IEEE International Symposium on*, Sept 2013, pp. 56–65.

[8] K. Fürlinger, C. Klausecker, and D. Kranzlmüller, "Towards energy efficient parallel computing on consumer electronic devices," in *Proceedings of the First international conference on Information and communication on technology for the fight against global warming*, ser. ICT-GLOW'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2035539.2035541

[9] V. Janapa Reddi, B. C. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: quantifying and mitigating the price of efficiency," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 314–325, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816002

[10] P. Stanley-Marbell and V. Cabezas, "Performance, power, and thermal analysis of low-power processors for scale-out systems," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, 2011, pp. 863–870.

[11] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips," in *Proceedings of the 37th annual international symposium on Computer architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 37–47. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815968

[12] A. Pedram, R. van de Geijn, and A. Gerstlauer, "Codesign tradeoffs for high-performance, low-power linear algebra architectures," *Computers, IEEE Transactions on*, vol. 61, no. 12, pp. 1724–1736, 2012.

[13] A. Pedram, S. Z. Gilani, N. S. Kim, R. A. van de Geijn, M. J. Schulte, and A. Gerstlauer, "A linear algebra core design for efficient level-3 blas," in *ASAP*, 2012, pp. 149–152.

[14] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," pp. 1–1, 2013.

[15] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, "Powermon: Fine-grained and integrated power monitoring for commodity computer systems," in *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, march 2010, pp. 479 –484.

[16] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 21, no. 5, pp. 658–671, May 2010.

[17] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, March-April 2012.

[18] NVIDIA, "NVML API Reference Manual," 2012.

[19] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, Dec 2007, pp. 3–14.

[20] S. Li, J.-H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, Dec 2009, pp. 469–480.

[21] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: Enabling energy optimizations in gpgpus," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 487–498. [Online]. Available: http://doi.acm.org.prx.library.gatech.edu/10.1145/2485922.2485964

[22] A. Kolmogorov, "Sulla determinazione empirica di una legge di distribuzione," *Giornale dell'Instituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.

[23] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, "Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound," *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pp. 947–953, May 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6270741