What Is Happening to Power, Performance, and Software?

SYSTEMATICALLY EXPLORING POWER, PERFORMANCE, AND ENERGY SHEDS NEW LIGHT ON THE CLASH OF TWO TRENDS THAT UNFOLDED OVER THE PAST DECADE: THE RISE OF PARALLEL PROCESSORS IN RESPONSE TO TECHNOLOGY CONSTRAINTS ON POWER, CLOCK SPEED, AND WIRE DELAY; AND THE RISE OF MANAGED HIGH-LEVEL, PORTABLE PROGRAMMING LANGUAGES.

Hadi Esmaeilzadeh
University of Washington

Ting Cao
Xi Yang
Stephen M. Blackburn
Australian National University

Kathryn S. McKinley
Microsoft Research

Quantitative performance analysis is the foundation for computer system design and innovation. In their classic paper, Emer and Clark noted that "a lack of detailed timing information impairs efforts to improve performance."[1] They pioneered the quantitative approach by characterizing instruction mix and cycles per instruction on timesharing workloads. Emer and Clark surprised expert readers by demonstrating a gap between the theoretical 1 million instructions per second (MIPS) peak of the VAX-11/780 and the 0.5 MIPS it delivered on real workloads. Hardware and software researchers in industry and academia now use and have extended this principled performance analysis methodology. Our research applies this quantitative approach to measured power and energy.

This work is timely because the past decade heralded the era of power-constrained hardware design. Hardware demands for energy efficiency intensified in large-scale systems, in which power began to dominate costs, and in mobile systems, which are constrained by battery life. Unfortunately, technology limits on power retard Dennard scaling[2] and prevent systems from using all transistors simultaneously (dark silicon).[3] These constraints are forcing architects seeking performance improvements and energy efficiency in smaller technologies to build parallel heterogeneous architectures. This hardware requires parallel software and exposes software to ongoing hardware upheaval. Unfortunately, most software today is not parallel, nor is it designed to modularly decompose onto a heterogeneous substrate.

Over this same decade, Moore's transistor bounty drove orthogonal and disruptive software changes with respect to how software is deployed, sold, and built. Software demands for correctness, complexity management, programmer productivity, time-to-market, reliability, security, and portability pushed developers away from low-level compiled ahead-of-time (native) programming languages. Developers increasingly choose high-level managed programming languages with a selection of safe pointer disciplines, garbage collection (automatic memory management), extensive standard libraries, and portability through dynamic just-in-time compilation. For example, modern web services combine managed languages, such as PHP on the server side and JavaScript on the client side. In markets as diverse as financial software and cell phone applications, Java and .NET are the dominant choice.

Published by the IEEE Computer Society

Related Work in Power Measurement, Power Modeling, and Methodology

Processor design literature is full of performance measurement and analysis. Despite power's growing importance, power measurements are still relatively rare.[1-3] Isci and Martonosi combine a clamp ammeter with performance counters for per-unit power estimation of the Intel Pentium 4 on SPEC CPU2000.[2] Fan et al. estimate whole-system power for large-scale data centers.[1] They find that even the most power-consuming workloads draw less than 60 percent of peak possible power consumption. We measure chip power and support their results by showing that thermal design power (TDP) does not predict measured chip power. Our work is the first to compare microarchitectures, technology generations, individual benchmarks, and workloads in the context of power and performance.

Power modeling is necessary to thoroughly explore architecture design.[4-6] Measurement complements simulation by providing validation. For example, some prior simulators used TDP, but our measurements show that this estimate is not accurate. As we look to the future, programmers will need to tune their applications for power and energy, and not just performance. Just as architectural event performance counters provide insight to applications, so will power and energy measurements.

Although the results show conclusively that managed and native workloads respond differently to architectural variations, perhaps this result should not be surprising.[7] Unfortunately, few architecture or operating systems publications with processor measurements or simulated designs use Java or any other managed workloads, even though the evaluation methodologies we use here for real processors and those for simulators are well developed.[7]

References

1. X. Fan, W.D. Weber, and L.A. Barroso, ''Power Provisioning for a Warehouse-sized Computer,'' *Proc. 34th Ann. Int'l Symp. Computer Architecture* (ISCA 07), ACM, 2007, pp. 13-23.
2. C. Isci and M. Martonosi, ''Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data'', *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (Micro 36), IEEE CS, 2003, pp. 93-104.
3. E. Le Sueur and G. Heiser, ''Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns,'' *Proc. Int'l Conf. Power-Aware Computing and Systems* (HotPower 10), USENIX Assoc., 2010, pp. 1-8.
4. O. Azizi et al., ''Energy-Performance Tradeoffs in Processor Architecture and Circuit Design: A Marginal Cost Analysis,'' *Proc. 37th Ann Int'l Symp. Computer Architecture* (ISCA 10), 2010, pp. 26-36.
5. Y. Li et al., ''CMP Design Space Exploration Subject to Physical Constraints,'' *Proc. 12th Int'l Symp. High-Performance Computer Architecture,* IEEE CS, 2006, pp. 17-28.
6. J.B. Brockman et al., ''McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures,'' *Proc. 42nd Ann. IEEE/ACM Int'l Symp. Microarchitecture (MICRO 42),* IEEE CS, 2009, pp. 469-480.
7. S.M. Blackburn et al., ''Wake Up and Smell the Coffee: Evaluation Methodologies for the 21st Century,'' *Comm. ACM,* vol. 51, no. 8, 2008, pp.83-89.

Exponential performance improvements in hardware hid many of the costs of high-level languages and helped create a virtuous cycle with ever more capable, reliable, and well performing software. This ecosystem is resulting in an explosion of developers, software, and devices that continue to change how we live and learn.

Unfortunately, a lack of detailed power measurements is impairing efforts to reduce energy consumption on modern software.

Examining power, performance, and energy

This work quantitatively examines power, performance, and energy during this period of disruptive software and hardware changes (2003 to 2011). Voluminous research explores performance and a growing body of work explores power (see the ''Related Work in Power Measurement, Power Modeling, and Methodology'' sidebar), but our work is the first to systematically measure the power, performance, and energy characteristics of software and hardware across a range of processors, technologies, and workloads. We execute 61 diverse sequential and parallel benchmarks written in three native languages and one managed language, all of which are widely used: C, C++, Fortran, and Java. We choose Java because it has mature virtual machine technology and substantial open-source benchmarks. We choose eight representative Intel IA32 processors from five technology generations (130 nm to 32 nm). Each processor has an isolated processor power supply on the motherboard. Each processor has a power supply with stable voltage, to which we attach a Hall effect sensor that measures current and, hence, processor power. We calibrate and validate our sensor data. We find that power consumption varies widely among benchmarks.

Findings

- Power consumption is highly application dependent and is poorly correlated to TDP.
- Energy-efficient architecture design is very sensitive to workload. Configurations in the native nonscalable Pareto frontier differ substantially from all the other workloads.
- Comparing one core to two, enabling a core is not consistently energy efficient.
- The Java Virtual Machine induces parallelism into the execution of single-threaded Java benchmarks.
- Simultaneous multithreading delivers substantial energy savings for recent hardware and in-order processors.
- The most recent processor in our study does not consistently increase energy consumption as its clock increases.
- The power/performance response to clock scaling of the native nonscalable workload differs from the other workloads.

Figure 1. We organize our discussion around these seven findings from an analysis of measured chip power, performance, and energy on 61 workloads and eight processors. The ASPLOS paper includes more findings and analysis.[4]

Furthermore, relative power, performance, and energy are not well predicted by core count, clock speed, or reported thermal design power (TDP).

We use controlled hardware configurations to explore the energy impact of hardware features, software parallelism, and workload. We perform historical and Pareto analyses that identify the most energy-efficient designs in our architecture configuration space. We made all of our data publicly available in the ACM Digital Library as a companion to our original ASPLOS 2011 paper.[4] Our data quantifies the extent, with precision and depth, of some known workload and hardware trends and some previously unobserved trends. This article is organized around just seven findings, listed in Figure 1, from our more comprehensive ASPLOS analysis. Two themes emerge from our analysis with respect to workload and architecture.

Workload

The power, performance, and energy trends of nonscalable native workloads differ substantially from native parallel and managed workloads. For example, the SPEC CPU2006 native benchmarks draw significantly less power than parallel benchmarks; and managed runtimes exploit parallelism even when executing single-threaded applications. Our results recommend that systems researchers include managed, native, sequential and parallel workloads when designing and evaluating energy-efficient systems.

Architecture

Hardware features such as clock scaling, gross microarchitecture, simultaneous multithreading (SMT), and chip multiprocessors (CMPs) each elicit a huge variety of power, performance, and energy responses. This variety and the difficulty of obtaining power measurements recommends exposing on-chip power meters and, when possible, structure-specific power meters for cores, caches, and other structures.

Modern processors include power-management techniques that monitor power sensors to minimize power usage and boost performance, but these sensors are not visible. Only in 2011, after our original paper, did Intel first expose energy counters in a production processor (Sandy Bridge).[5] Just as hardware event counters provide a quantitative grounding for performance innovations, future architectures should include power meters to drive innovation in the power-constrained computer systems era.

Measurement is the key to understanding and optimization.

Methodology

This section overviews essential elements of our experimental methodology. For a more detailed treatment, see our original paper.[4]

Software

We systematically explored workload selection because it is a critical component for analyzing power and performance. Native and managed applications embody different tradeoffs between performance, reliability, portability, and deployment. It is impossible to meaningfully separate language from workload. We offer no commentary on the virtue of language choice. We created the following four workloads from 61 benchmarks.

- *Native nonscalable benchmarks:* C, C++, and Fortran single-threaded compute-intensive benchmarks from SPEC CPU2006.

- *Native scalable benchmarks:* Multithreaded C and C++ benchmarks from PARSEC.
- *Java nonscalable benchmarks:* Single and multithreaded benchmarks that do not scale well from SPECjvm, DaCapo 06-10-MR2, DaCapo 9.12, and pjbb2005.
- *Java scalable benchmarks:* Multithreaded Java benchmarks from DaCapo 9.12 that scale in performance similarly to native scalable benchmarks on the i7 (45).

We execute the Java benchmarks on the Oracle HotSpot 1.6.0 Virtual Machine because it is a mature high-performance virtual machine. The virtual machine dynamically optimizes each benchmark on each architecture. We used best practices for virtual machine measurement of steady-state performance.[6] We compiled native nonscalable with icc at `-o3`. We used gcc at `-o3` for native scalable benchmarks because icc did not correctly compile all benchmarks. The icc compiler usually produces better performing code than the gcc compiler. We executed the same native binaries on all machines. All of the parallel native benchmarks scale up to eight hardware contexts. The Java scalable benchmarks are the subset of Java benchmarks that scale similarly to the native scalable benchmarks.

Hardware

Table 1 lists the eight Intel IA32 processors that cover four process technologies (130 nm, 65 nm, 45 nm, and 32 nm) and four microarchitectures (NetBurst, Core, Bonnell, and Nehalem). The release price and date give context regarding Intel's market placement. The Atoms and the Core 2Q (65) Kentsfield are extreme market points. These processors are only examples of many processors in each family. For example, Intel sells more than 60 Nehalems at 45 nm, ranging in price from around US$190 to more than US$3,700. We used these samples because they were sold at similar price points.

To explore the influence of architectural features, we selectively down-clocked (reduced the frequency of the CPU from its default) the processors, disabled the cores on the chip multiprocessors (CMP), disabled simultaneous multithreading (SMT), and disabled Turbo Boost using operating system boot time BIOS configuration.

Power, performance, and energy measurements

We measured on-chip power by isolating the direct current (DC) power supply to the processor on the motherboard. Prior work used a clamp ammeter, which can only measure the whole system alternating current (AC) supply.[7-9] We used Pololu's ACS714 current sensor board. The board is a carrier for Allegro's ±5 A ACS714 Hall effect-based linear current sensor. The sensor accepts a bidirectional current input with a magnitude up to 5 A. The output is an analog voltage (185 mV/A) centered at 2.5 V with a typical error of less than 1.5 percent. We place the sensors on the 12 V power line that supplies only the processor. We measured voltage and found it to be stable, varying less than 1 percent. We sent the values from the current sensor to the machine's USB port using a Sparkfun's Atmel AVR Stick, which is a simple data-logging device with a data-sampling rate of 50 Hz. We used a similar arrangement with a 30A Hall effect sensor for the high power i7 (45). We executed each benchmark, logged its power values, and then computed average power consumption.

After publishing the original paper, Intel made chip-level and core-level energy measurements available on Sandy Bridge processors.[5] Our methodology should slightly overstate chip power because it includes losses due to the motherboard's voltage regulator. Validating against the Sandy Bridge energy counter shows that our power measurements consistently measure about 5 percent more current.

We executed each benchmark multiple times. The aggregate 95 percent confidence intervals of execution time and power range from 0.7 to 4 percent. The measurement error in time and power for all processors and applications is low.

We compute arithmetic means over the four workloads, weighting each workload equally. To avoid biasing performance measurements to any one architecture, we compute a reference performance. We normalize

Table 1. Specifications for the eight experimental processors.

Processor	µArch	Processor	sSpec	Release date	Price (US$)	CMP SMT	LLC B
Pentium 4	NetBurst	Northwood	SL6WF	May 2003	—	1C2T	512 K
Core 2 Duo E6600	Core	Conroe	SL9S8	July 2006	$316	2C1T	4 M
Core 2 Quad Q6600	Core	Kentsfield	SL9UM	Jan. 2007	$851	4C1T	8 M
Core i7 920	Nehalem	Bloomfield	SLBCH	Nov. 2008	$284	4C2T	8 M
Atom 230	Bonnell	Diamondville	SLB6Z	June 2008	$29	1C2T	512 K
Core 2 Duo E7600	Core	Wolfdale	SLGTD	May 2009	$133	2C1T	3 M
Atom D510	Bonnell	Pineview	SLBLA	Dec. 2009	$63	2C2T	1 M
Core i5 670	Nehalem	Clarkdale	SLBLT	Jan. 2010	$284	2C2T	4 M

* µArch: microarchitecture; CMP: chip multiprocessor; SMT: simultaneous multithreading; LLC: last level cache; VID: processor's stock voltage; TDP: thermal design power; FSB: front-side bus; B/W: bandwidth.

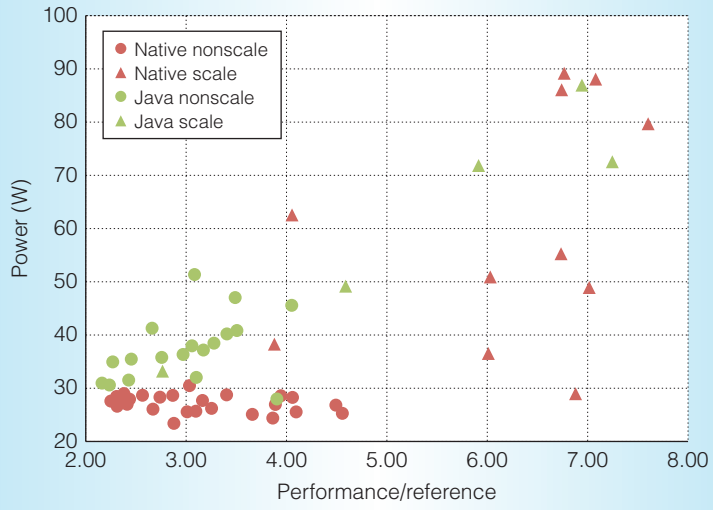

Figure 2. Power/performance distribution on the i7 (45). Each point represents one of the 61 benchmarks. Power consumption is highly variable among the benchmarks, spanning from 23 W to 89 W. The wide spectrum of power responses from different benchmarks points to power saving opportunities in software.

individual benchmark execution times to the average execution time when executed on four architectures: Pentium 4 (130), Core 2D (65), Atom (45), and i5 (32). These choices capture four microarchitectures and four technology generations. We also normalized energy to a reference, since energy = power × time. The reference energy is the average power on these four processors times the average execution time. Given a power and time measurement, we compute energy and then normalize it to the reference energy.

We measured 45 processor configurations (8 stock and 37 BIOS configured) and produced power and performance data for each benchmark and processor, as illustrated in Figure 2.

Perspective

The rest of this article organizes our analysis by the seven findings listed in Figure 1. (The ASPLOS paper contains additional analysis, results, and findings.) We begin with broad trends. We show that applications exhibit a large range of power and performance characteristics that are not well summarized by a single number. This section conducts a Pareto energy efficiency analysis for all the 45 nm processor configurations. Even with this modest exploration of architectural features, the results indicate that each workload prefers a different processor configuration for energy efficiency.

Power is application dependent

The nominal thermal design power (TDP) for a processor is the amount of power the chip may dissipate without exceeding the maximum transistor junction temperature. Table 1 lists the TDP for each processor. Because measuring real processor power is difficult and TDP is readily available, researchers often substitute TDP for real measured power. Figure 3 shows that this substitution is problematic. It plots on a logarithmic scale measured power for each benchmark on each stock processor as a function of TDP. TDP is marked with an ✗. Note that TDP is strictly

Clock (GHz)	nm	Trans (M)	Die (mm^2)	VID range (V)	TDP (W)	FSB (MHz)	B/W (Gbytes/s)	DRAM model
2.4	130	55	131	—	66	800	—	DDR-400
2.4	65	291	143	0.85–1.50	65	1,066	—	DDR2-800
2.4	65	582	286	0.85–1.50	105	1,066	—	DDR2-800
2.7	45	731	263	0.80–1.38	130	—	25.6	DDR3-1066
1.7	45	47	26	0.90–1.16	4	533	—	DDR2-800
3.1	45	228	82	0.85–1.36	65	1,066	—	DDR2-800
1.7	45	176	87	0.80–1.17	13	665	—	DDR2-800
3.4	32	382	81	0.65–1.40	73	—	21.0	DDR3-1333

higher than actual power, that the gap between peak measured power and TDP varies from processor to processor, and that TDP is up to a factor of four higher than measured power. The variation among benchmarks is highest on the i7 (45) and i5 (32), which likely reflects their advanced power management. For example on the i7 (45), measured power varies between 23 W for 471.omnetpp and 89 W for fluidanimate. The smallest variation between maximum and minimum is on the Atom (45) at 30 percent. This trend is not new. All the processors exhibit a range of application specific power variation. TDP loosely correlates with power consumption, but it does not provide a good estimate for maximum power consumption of individual processors, comparing among processors, or approximating benchmark-specific power consumption.

Finding: Power consumption is highly application dependent and is poorly correlated to TDP.

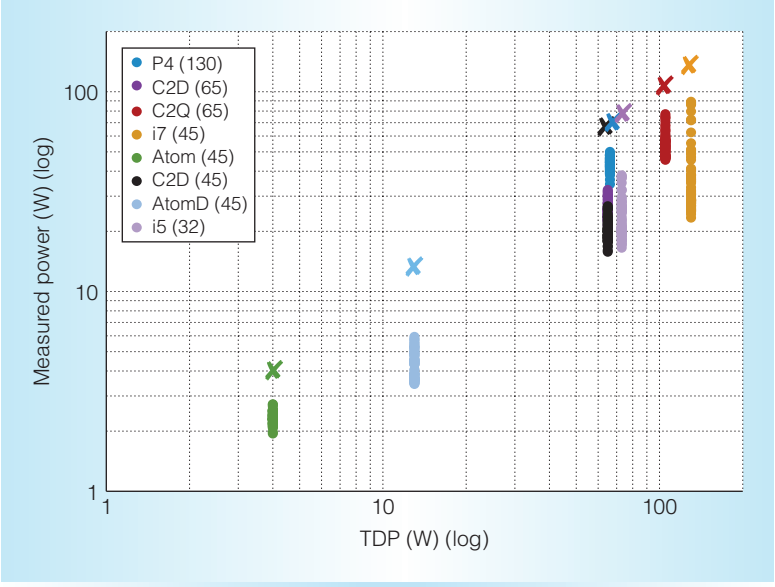

Figure 3. Measured power for each processor running 61 benchmarks. Each point represents measured power for one benchmark. An ✗ shows the reported TDP for each processor. Power is application dependent and does not strongly correlate with TDP.

Figure 2 plots power versus relative performance for each benchmark on the i7 (45) with eight hardware contexts, which is the most recent of the 45 nm processors. We measure this data for every processor configuration. Native and managed benchmarks are differentiated by color, whereas scalable and nonscalable benchmarks are differentiated by shape. Unsurprisingly, the scalable benchmarks (triangles) tend to perform the best and consume the most power. More unexpected is the range of performance and power characteristics of the nonscalable benchmarks. Power is not strongly correlated with performance across benchmarks. If the correlation were strong, the points would form a straight line. For example, the point on the bottom right of the figure achieves almost the best relative performance and lowest power.

Pareto analysis at 45 nm

The Pareto optimal frontier defines a set of choices that are most efficient in a tradeoff space. Prior research uses the Pareto frontier to explore power versus performance using

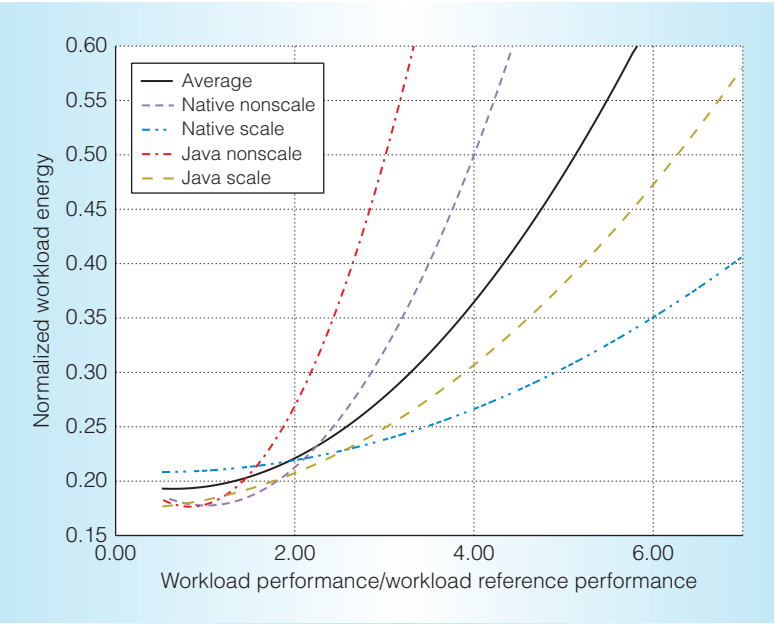

Figure 4. Energy/performance Pareto frontiers for 45 nm processors. The energy/performance Pareto frontiers are workload-dependent and significantly deviate from the average.

models to derive potential architectural designs on the frontier.[10] We present a Pareto frontier derived from measured performance and power. We hold the process technology constant by using the four 45 nm processors: Atom, Atom D, Core 2D, and i7. We expand the number of processor configurations from 4 to 29 by configuring the number of hardware contexts (CMP and SMT), clock scaling, and disabling or enabling Turbo Boost. The 25 nonstock configurations represent alternative design points. We then compute an energy and performance scatter plot (not shown here) for each hardware configuration, workload, and workload average. We next pick off the frontier—the points that are not dominated in performance or energy efficiency by any other point—and fit them with a polynomial curve. Figure 4 plots these polynomial curves for each workload and the average. The rightmost curve delivers the best performance for the least energy.

Each row of Figure 5 corresponds to one of the five curves in Figure 4. The check marks identify the Pareto-efficient configurations that define the bounding curve and include 15 of 29 processor configurations. Somewhat surprising is that none of the Atom D (45) configurations are Pareto-efficient. Notice:

- Native nonscalable shares only one choice with any other workload.
- Java scalable and the average share all the same choices.
- Only two of 11 choices for Java nonscalable and scalable workloads are common to both.
- Native nonscalable does not include the Atom (45) in its frontier.

Figure 5. Pareto-efficient processor configurations for each workload. Stock configurations are bold. Each ✓ indicates that the configuration is on the energy/performance Pareto-optimal curve. Native nonscalable has almost no overlap with any other workload.

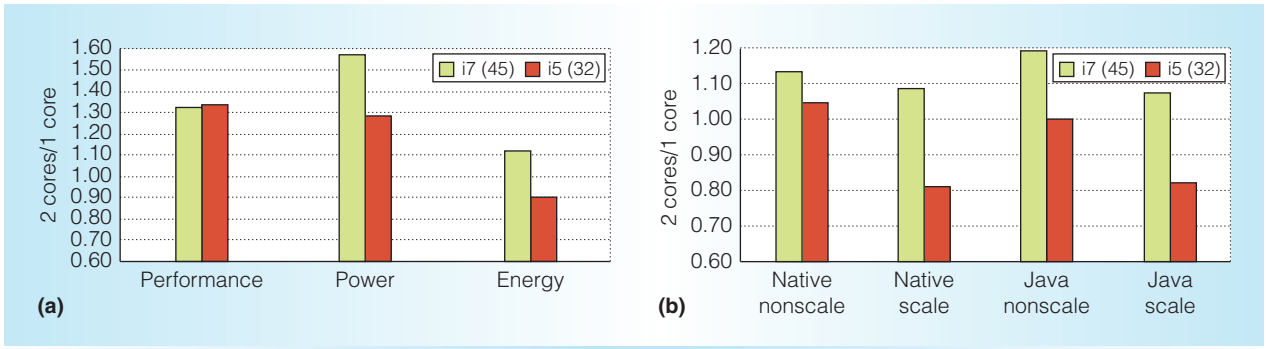

Figure 6. CMP: Comparing two cores to one core. Impact of doubling the number of cores on performance, power, and energy, averaged over all four workloads (a). Energy impact of doubling the number of cores for each workload (b). Doubling the cores is not consistently energy efficient among processors or workloads.

This last finding contradicts prior simulation work, which concluded that dual-issue in-order cores and dual-issue out-of-order cores are Pareto-optimal designs for native nonscalable workloads.[10] Instead, we find that all of the Pareto efficient points for the native nonscalable workload map to a quad-issue out-of-order i7 (45).

Figure 4 shows that each workload deviates substantially from the average. Even when the workloads share points, the points fall in different places on the curves because each workload exhibits a different energy and performance tradeoff. Compare the scalable and nonscalable benchmarks at 0.40 normalized energy on the *y*-axis. It is impressive how well these architectures effectively exploit software parallelism, pushing the curves to the right and increasing normalized performance from about 3 to 7 while holding energy constant. This measured behavior confirms prior model-based observations about software parallelism's role in extending the energy and performance curve to the right.[10]

Finding: Energy-efficient architecture design is very sensitive to workload. Configurations in the native nonscalable Pareto frontier differ substantially from all other workloads.

In summary, architects should use a variety of workloads and, in particular, should avoid only using native nonscalable workloads.

Feature analysis

Our original paper evaluates the energy effect of a range of hardware features: clock frequency, die shrink, memory hierarchy, hardware parallelism, and gross microarchitecture. This analysis yielded a large number of findings and insights. Reader and reviewer feedback yielded diverse opinions as to which findings were most surprising and interesting. To give a flavor of our analysis, this section presents our CMP, SMT, and clock scaling analysis.

Chip multiprocessors

Figure 6 shows the average power, performance, and energy effects of chip multiprocessors (CMPs) by comparing one core to two cores from the two most recent processors in our study. We disable Turbo Boost in all these analyses because it adjusts power dynamically based on the number of idle cores. We disable SMT here to maximally expose thread-level parallelism to the CMP hardware feature. Figure 6a compares relative power, performance, and energy as an average of the workloads. Figure 6b breaks down the energy efficiency as a function of the workload. While average energy is reduced by 9 percent when adding a core to the i5 (32), it is increased by 12 percent when adding a core to the i7 (45). Figure 6a shows that the source of this difference is that the i7 (45) experiences twice the power overhead for enabling a core as the i5 (32) while producing roughly the same performance improvement.

Finding: Comparing one core to two, enabling a core is not consistently energy efficient.

Figure 6b shows that native nonscalable and Java nonscalable benchmarks suffer the most energy overhead with the addition of

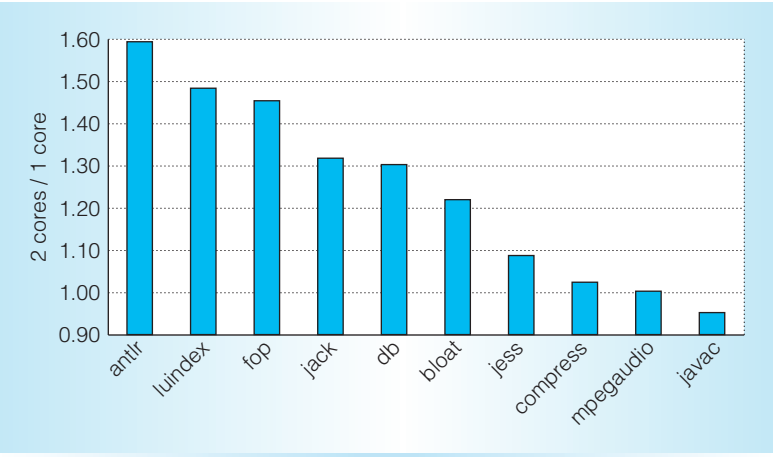

Figure 7. Scalability of single-threaded Java benchmarks. Counterintuitively, some single-threaded Java benchmarks scale well. They scale because the underlying JVM exploits parallelism during compilation, profiling, and garbage collection.

another core on the i7 (45). As expected, performance for native nonscalable benchmarks is unaffected. However, turning on an additional core for native nonscalable benchmarks leads to a power increase of 4 percent and 14 percent, respectively, for the i5 (32) and i7 (45), translating to energy overheads.

More interesting is that Java nonscalable benchmarks do not incur energy overhead when enabling another core on the i5 (32). In fact, we were surprised to find that the single-threaded Java nonscalable benchmarks run faster with two processors! Figure 7 shows the scalability of the single-threaded subset of Java nonscalable on the i7 (45), comparing one and two cores. Although the Java benchmarks are single threaded, the Java virtual machines (JVMs) on which they execute are not.

Finding: The JVM induces parallelism into the execution of single-threaded Java benchmarks.

Because virtual machine runtime services for managed languages—such as just-in-time (JIT) compilation, profiling, and garbage collection—are often concurrent and parallel, they provide substantial scope for parallelization, even within ostensibly sequential applications. We instrumented the HotSpot JVM and found that HotSpot JIT compilation and garbage collection are parallel. Detailed performance-counter measurements revealed that the garbage collector induced memory system improvements with more cores by reducing the collector's displacement effect on the application thread.

Simultaneous multithreading

Figure 8 shows the effect of disabling simultaneous multithreading (SMT) on the Pentium 4 (130), Atom (45), i5 (32), and i7 (45).[11] Each processor supports two-way SMT. SMT provides fine-grained parallelism to distinct threads in the processors' issue logic and threads share all processor components, such as execution units and caches. Singhal states that the small amount of logic exclusive to SMT consumes very little power.[12] Nonetheless, this logic is integrated, so SMT contributes a small amount to total power even when disabled. Therefore, our results slightly underestimate SMT's power cost. We use only one core to ensure that SMT is the sole opportunity for thread-level parallelism. Figure 8a shows that SMT's performance advantage is significant. On the i5 (32) and Atom (45), SMT improves average performance significantly, without much cost in power, leading to net energy savings.

Finding: SMT delivers substantial energy savings for recent hardware and for in-order processors.

Given that SMT was motivated by the challenge of filling issue slots and hiding latency in wide-issue superscalars, it may appear counter intuitive that performance on the dual-issue in-order Atom (45) should benefit so much more from SMT than the quad-issue i7 (45) and i5 (32) benefit. One explanation is that the in-order pipelined Atom (45) is more restricted in its capacity to fill issue slots. Compared to other processors in this study, the Atom (45) has much smaller caches. These features accentuate the need to hide latency and therefore the value of SMT. The performance improvements on the Pentium 4 (130) due to SMT are half to one-third of more-recent processors. Consequently, there is no net energy advantage. This result is not so surprising given that the Pentium 4 (130) was the first commercial SMT implementation.

Figure 8b shows that, as expected, native nonscalable benchmarks experience little

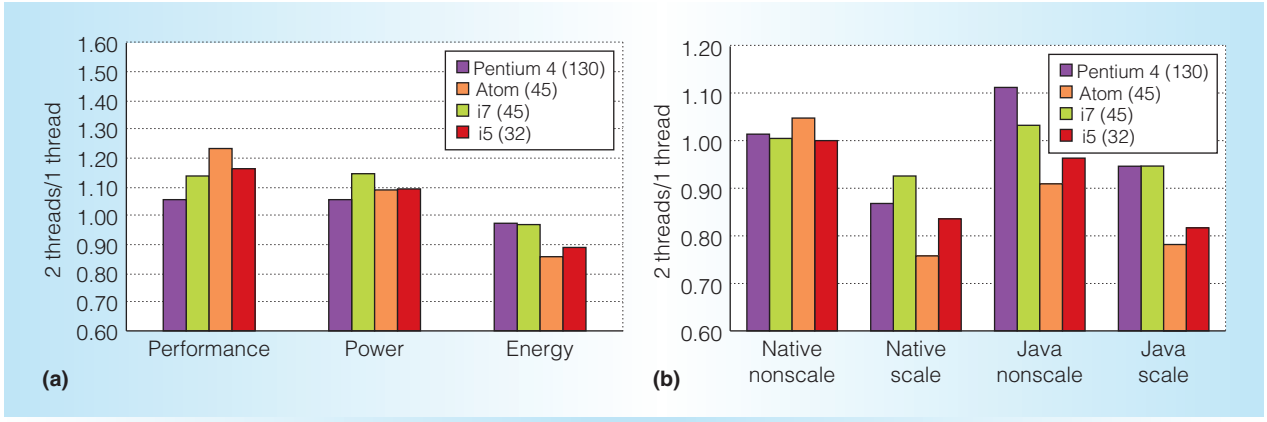

Figure 8. SMT: one core with and without SMT. Impact of enabling two-way SMT on a single core with respect to performance, power, and energy averaged over all four workloads (a). Energy impact of enabling two-way SMT on a single core for each workload (b). Enabling SMT delivers significant energy savings on the recent i5 (32) and the in-order Atom (45).

energy overhead due to enabling SMT. Whereas Figure 6b shows that enabling a core incurs a significant power and thus energy penalty. The scalable benchmarks of course benefit the most from SMT.

Compare Figures 6 and 8 to see SMT's effectiveness, which is impressive on recent processors as compared to CMP, particularly given its low die footprint. SMT provides about half the performance improvement compared to CMP, but incurs much lower power costs. These results on the modern processors show SMT in a much more favorable light than in a study of the energy efficiency of SMT and CMP using models.[13]

Clock scaling

We vary the processor clock on the i7 (45), Core 2D (45), and i5 (32) between their minimum and maximum settings. The range of clock speeds are 1.6 to 2.7 GHz for i7 (45), 1.6 to 3.1 GHz for Core 2D (45), and 1.2 to 3.5 GHz for i5 (32). Figures 9a and 9b express changes in power, performance, and energy with respect to doubling in clock frequency over the range of clock speeds to normalize and compare across architectures.

The three processors experience broadly similar performance increases of about 80 percent, but power differences vary substantially, from 70 percent to 180 percent. On the i7 (45) and Core 2D (45), the performance increases require disproportional power increases. Consequently, energy consumption increases by about 60 percent as the clock is doubled. In stark contrast, doubling the clock on the i5 (32) leads to a slight energy reduction.

Finding: The most recent processor in our study does not consistently increase energy consumption as its clock increases.

A number of factors may explain why the i5 (32) performs so much better at its highest clock rate. First, the i5 is a 32 nm processor, while the others are 45 nm and the voltage setting for each frequency setting is probably different. Second, the power-performance curve is nonlinear and these experiments may observe only the upper (steeper) portion of the curves for i7 (45) and Core 2D (45). Third, although the i5 (32) and i7 (45) share the same microarchitecture, the second generation i5 (32) likely incorporates energy improvements. Fourth, the i7 (45) is substantially larger than the other processors, with four cores and a larger cache.

Finding: The power/performance response to clock scaling of the native nonscalable workload differs from the other workloads.

Figure 9b shows that doubling the clock on the i5 (32) roughly maintains or improves energy consumption of all workloads, with the native nonscalable workload improving the most. For the i7 (45) and Core 2D (45), doubling the clock raises energy consumption. Figure 9d shows that the native nonscalable workload has different power and performance behavior compared to the

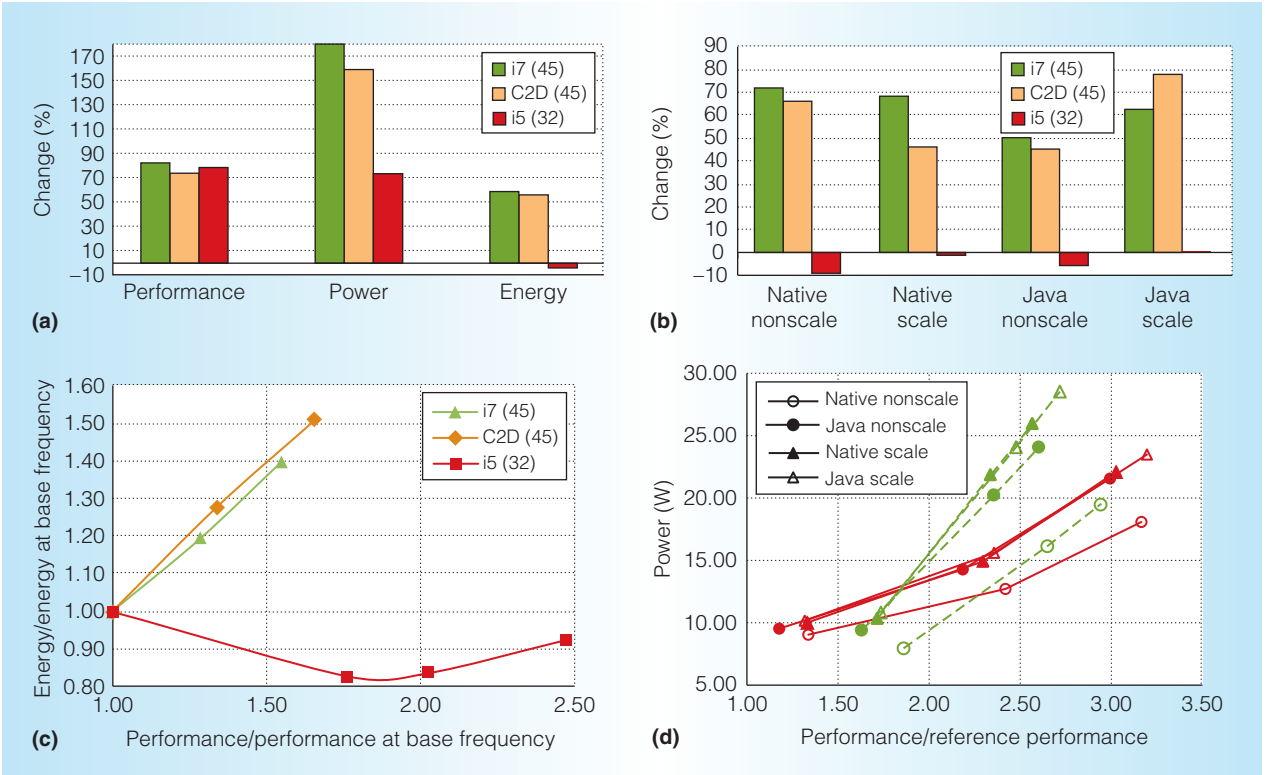

Figure 9. The impact of clock scaling in stock configurations. Power, performance, and energy impact of doubling clock averaged over all four workloads (a). Energy impact of doubling clock for each workload (b). Average energy performance curve; points are clock speeds (c). Absolute power (watts) and performance on the i7 (45) (dashed lines) and i5 (32) (solid lines) by workload; points are clock speeds (d). The i5 (32) does not consistently increase energy consumption as its clock increases. The power and performance response of the native nonscalable workloads to clock scaling differs from the other workloads.

other workloads, and this difference is largely independent of clock rate. Overall, benchmarks in the native nonscalable workload draw less power and power increases less steeply as a function of performance increases. The native nonscalable workload (SPEC CPU2006) is the most widely studied workload in the architecture literature, but it is an outlier. These results reinforce the importance of including scalable and managed workloads in energy evaluations.

Our experimentation and analysis yield a wide-ranging set of findings in this critical time period of hardware and software upheaval. These experiments and analyses recommend that manufacturers should expose on-chip power meters to the community, compiler and operating system developers should understand and optimize energy, researchers should use both managed and native workloads to quantitatively examine their innovations, and researchers should measure power and performance to understand and optimize power, performance, and energy. MICRO

Acknowledgments

We thank Bob Edwards, Daniel Frampton, Katherine Coons, Pradeep Dubey, Jungwoo Ha, Laurence Hellyer, Daniel Jiménez, Bert Maher, Norm Jouppi, David Patterson, and John Hennessy. This work is supported by Australian Research Council grant DP0666059 and National Science Foundation grant CSR-0917191.

References

1. J.S. Emer and D.W. Clark, ''A Characterization of Processor Performance in the VAX-11/780,'' *Proc. 11th Ann. Int'l Symp. Computer Architecture* (ISCA 84), ACM, 1984, pp. 301-310.

2. M. Bohr, ''A 30-Year Retrospective on Dennard's MOSFET Scaling Paper,'' *IEEE SSCS Newsletter,* 2007, pp. 11-13.
3. H. Esmaeilzadeh et al., ''Dark Silicon and the End of Multicore Scaling,'' *Proc. 38th Ann. Int'l Symp. Computer Architecture* (ISCA 11), ACM, 2011, pp. 365-376.
4. H. Esmaeilzadeh et al., ''Looking Back on the Language and Hardware Revolutions: Measured Power, Performance, and Scaling,'' *Proc. 16th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (Asplos 11), ACM, 2011, pp. 319-332.
5. S.M. Blackburn et al., ''Wake Up and Smell the Coffee: Evaluation Methodologies for the 21st Century,'' *Comm. ACM,* vol. 51, no. 8, 2008, pp. 83-89.
6. E. Le Sueur and G. Heiser, ''Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns,'' *Proc. Int'l Conf. Power-Aware Computing and Systems* (HotPower 10), USENIX Assoc., 2010, pp. 1-8.
7. C. Isci and M. Martonosi, ''Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data,'' *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (Micro 36), IEEE CS, 2003, pp. 93-104.
8. W.L. Bircher and L.K. John, ''Analysis of Dynamic Power Management on Multi-Core Processors,'' *Proc. 22nd Ann. Int'l Conf. Supercomputing* (ICS 08), ACM, 2008, pp. 327-338.
9. H. David et al., ''RAPL: Memory Power Estimation and Capping,'' *Proc. ACM/IEEE Int'l Symp. Low-Power Electronics and Design,* IEEE CS, 2010, pp. 189-194.
10. O. Azizi et al., ''Energy-Performance Tradeoffs in Processor Architecture and Circuit Design: A Marginal Cost Analysis,'' *Proc. 37th Ann Int'l Symp. Computer Architecture* (ISCA 10), 2010, pp. 26-36.
11. D.M. Tullsen, S.J. Eggers, and H.M. Levy, ''Simultaneous Multithreading: Maximizing On-chip Parallelism,'' *Proc. 22nd Ann. Int'l Symp. Computer Architecture,* 1995, pp. 392-403.
12. R. Singhal, ''Inside Intel Next Generation Nehalem Microarchitecture,'' Intel Developer Forum (IDF) presentation, Aug. 2008.
13. S.V. Adve et al., ''The Energy Efficiency of CMP vs. SMT for Multimedia Workloads,'' *Proc. 18th Ann. Int'l Conf. Supercomputing* (ICS 04), ACM, 2004, pp. 196-206.

Hadi Esmaeilzadeh is a PhD student in the Department of Computer Science and Engineering at the University of Washington. His research interests include power-efficient architectures, approximate general-purpose computing, mixed-signal architectures, machine learning, and compilers. Esmaeilzadeh has an MS in computer science from the University of Texas at Austin and an MS in electrical and computer engineering from the University of Tehran.

Ting Cao is a PhD student in the Department of Computer Science at the Australian National University. Her research interests include architecture and programming language implementation. Cao has an MS in computer science from National University of Defense Technology, China.

Xi Yang is a PhD student in the Department of Computer Science at the Australian National University. His research interests include architecture, operating systems, and runtime systems. Yang has an MPhil in computer science from the Australian National University.

Stephen M. Blackburn is a professor in the Research School of Computer Science at the Australian National University. His research interests include programming language implementation, architecture, and performance analysis. Blackburn has a PhD in computer science from the Australian National University. He is a Distinguished Scientist of the ACM.

Kathryn S. McKinley is a principal researcher at Microsoft Research and an endowed professor of computer science at the University of Texas at Austin. Her research interests span architecture, programming language implementation, reliability, and security. McKinley has a PhD in computer science from Rice University. She is a fellow of IEEE and the ACM.

Direct questions and comments about this article to Stephen M. Blackburn, School of Computer Science, Bldg. 108, Australian National University, Acton, ACT, 0200, Australia; Steve.Blackburn@anu.edu.au.