Profiling a Warehouse-Scale Computer

Datacenters are quickly becoming the platform of choice for modern applications. To understand how datacenter software utilizes the hardware and to improve future server processor performance, the authors profiled more than 20,000 Google machines over a three-year period, as they served the requests of billions of users.

Svilen Kanev
Juan Pablo Darago
Kim Hazelwood
Parthasarathy Ranganathan
Tipp Moseley
Google

Gu-Yeon Wei
David Brooks
Harvard University

The growth of cloud-based and cloud-supported computing in the past decade has created new challenges for the computer architectures that power large Internet services. The hardware platforms behind the cloud, referred to as *warehouse-scale computers* (WSCs), emphasize system design for Internet-scale services over thousands of computing nodes for performance and cost efficiency at scale. At such a scale, understanding performance characteristics becomes critical—even small improvements in performance or utilization can translate into immense cost savings for datacenter operators.

Historically, WSCs were created to solve computing problems significantly larger in size than those fitting on a single server, and their software stacks have evolved accordingly. They typically consist of a large set of distributed, multitiered services, each of which exposes a relatively narrow set of APIs and communicates to others exclusively through remote procedure calls (RPCs).

Although such a software architecture is easier to test, deploy, maintain, and iterate on, the large number of services also produces performance characteristics that are nontrivial to isolate and measure. Benchmarking efforts,[1,2] although extremely valuable for tuning individual services, often fail to capture the large-scale effects of their interaction, and therefore paint an incomplete picture. To compensate for this, we performed the first (to the best of our knowledge) longitudinal profiling study of a live production WSC. Our performance measurements span tens of thousands of machines, running thousands of different applications, which in turn serve the requests of billions of users over several years. We highlight important patterns and insights for computer architects, some significantly different from common wisdom for optimizing SPEC-like or open-source scale-out workloads.

We collected performance data from live workloads using Google Wide Profiling.[3] GWP is based on the premise of low-overhead random sampling of both machines within the datacenter and execution time within a machine. It has been unobtrusively sampling the cycle distribution across Google's server fleet for nearly a decade, which makes it a perfect vehicle for large-scale longitudinal performance studies, and lets us draw insights from many core-years' worth of measurements. We also extended GWP's infrastructure to collect more specific processor performance counters that go beyond attributing cycles to code regions. This lets us analyze more subtle interactions of warehouse-scale applications with the underlying hardware and ask microarchitecture specific questions.

Overall, our performance profiles suggest several interesting directions for future microarchitectural exploration: design of more general-purpose cores with additional threads to address broad workload diversity, with specific accelerators for "datacenter tax" components and increased emphasis on the memory hierarchy, including optimizations to trade off bandwidth for latency, as well as more advanced instruction-cache optimizations.

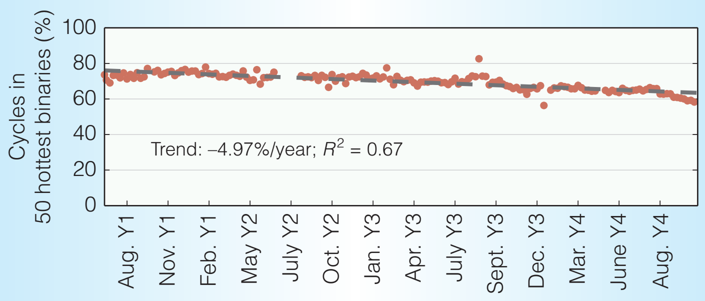

Figure 1. Workloads are getting more diverse. The fraction of cycles spent in the top 50 hottest binaries is steadily decreasing over the years.

Workload Diversity and Datacenter Tax

The most apparent outcome of this study is the diversity of workloads in a modern WSC. Although WSCs were initially created with a "killer application" in mind (be it Web search, social networking, or e-commerce), the model of "the datacenter is the computer" has since grown, and current datacenters handle a rapidly increasing pool of applications.

To confirm this point, we performed a longitudinal study of applications running in Google's WSCs over more than three years. We measured the distribution of CPU cycles across applications and found that no single application dominates execution time globally—the hottest one accounts for only approximately 10 percent of cycles. This distribution quickly becomes tail-heavy—it takes 50 different applications to build up to 60 percent of cycles, and thousands for the remaining 40 percent. Intra-application hotspots are not prevalent, either: profiles for single applications are predominantly flat. For example, a typical Web search binary covers 80 percent of cycles in more than 350 functions.

These findings are the result of an ongoing diversification trend. We demonstrate this in Figure 1, which plots the fraction of CPU cycles spent in the 50 hottest binaries for each week of the study. At the earliest periods we examined, 50 applications were enough to account for 80 percent of execution time, but three years later, the same number (not necessarily the same binaries) cover less than 60 percent of cycles. On average, the coverage of the top 50 binaries has been decreasing by 5 percentage points per year over a period of more than three years.

From a software engineer's perspective, the absence of immediately apparent hotspots, both on the application and function levels, implies that although there is value in optimizing hotspots on a per-application basis, the engineering costs associated with optimizing flat profiles are not always justified. This has driven Google to increasingly invest effort in automated, compiler-driven feedback-directed optimization.[4]

Nevertheless, targeting the right common building blocks across applications can have a significantly larger impact in the datacenter. Datacenter-wide profiling lets us identify these interapplication building blocks. We call the shared routines that comprise them the *datacenter tax,* after the necessary cost of performing computation that does not fit on a single machine.

We identify six components of this tax and estimate their contributions to all cycles in our WSCs. Figure 2 shows the results of this characterization over 11 months—"tax cycles" consistently comprise 22 to 27 percent of all observed execution. In a world of a growing number of applications, optimizing such interapplication common building blocks can lead to significant performance gains, more so than hunting for hotspots in individual binaries.

We included the following components in the tax classification: protocol buffer management, RPCs, hashing, compression, memory allocation, and data movement. When selecting which interapplication building blocks to classify as tax, we opted for generally mature low-level routines that are also relatively small and self-contained. The amount of code that

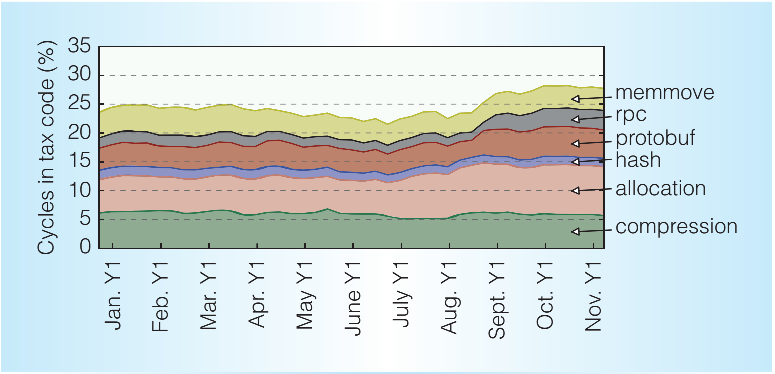

Figure 2. Datacenter tax components account for 22 to 27 percent of warehouse-scale computer (WSC) cycles. The fraction of tax cycles has shown relatively little variation over the duration of our study.

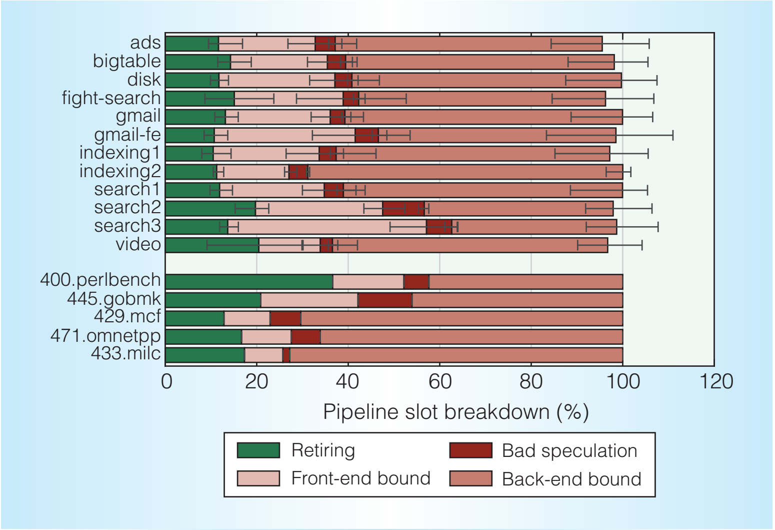

Figure 3. Top-level bottleneck breakdown. WSC workloads exhibit a combination of low retirement rates and high front-end boundedness.

we categorize as tax is in the tens of thousands of lines. On the contrary, if we were to account for the same amount of cycles on a per-application basis, we would need to cover at least three orders of magnitude more code lines. Thus, optimizing tax can have a very high reward-per-engineering-effort ratio.

Microarchitectural Diversity

Diversity in WSCs does not manifest itself only in the number of applications; we also see multiple bottlenecks on the microarchitectural level. To demonstrate the variety in bottlenecks, we extended the GWP infrastructure to collect performance-counter expressions and identify specific microarchitectural performance pain points. Each such dedicated collection touches on approximately 20,000 randomly selected Intel Ivy Bridge machines and samples execution on each machine for 1 second. We used Top-Down performance analysis to better understand bottleneck root causes.[5]

A major conclusion from this analysis is the prevalence of low retirement rates and, consecutively, the low instructions per cycle (IPC) that WSC workloads exhibit (see Figure 3)—almost two times lower than the SPECint geomean and close to that of the most memory-bound benchmarks in SPEC CPU2006. Although no single microarchitectural bottleneck is responsible for these high stall times, we identify various optimizations in the cache hierarchy, both instruction and data, that have the potential to reap substantial benefits.

Data Caches

On the data side, we found 50 to 60 percent of all core cycles stalled on data—a result consistent with the multi-gigabyte working sets of these applications, which do not fit in any level of the cache hierachy. Compared to more traditional high-performance computing workloads, these data stall cycles were not caused by bandwidth saturation. In fact, memory bandwidth utilization was habitually low; we measured a 95th percentile utilization of only 31 percent.

Similar results from benchmarking studies have inspired research in adopting wimpier cores in datacenters—if cores are mostly stalled on memory, there is little need to waste energy on wide out-of-order execution resources. Our measurements, however, suggest a more nuanced picture.

We show this in Figure 4 through the distribution of extracted instruction-level parallelism (ILP), or the number of simultaneously executing micro-ops at each cycle when some micro-ops are issued from the out-of-order scheduler to execution units. We see that 72 percent of execution cycles exhibit low ILP (one or two micro-ops per cycle on a six-wide Ivy Bridge core), consistent with the fact that the majority of cycles are spent waiting on caches. However, for the other 28 percent of cycles, three or more functional units are kept busy each cycle, which suggests a very large performance hit from moving to narrower cores.

One explanation consistent with such behavior is that WSC applications exhibit a fine-grained mix of dependent cache accesses and bursty computation. The bursts of computation can either be dependent on the cache references or independent and extractable as ILP. The difference between these two variants—whether intense computation-bound phases are on the critical path of execution—could be detrimental for the end performance of wimpier cores, and requires a dedicated simulation study.

Instruction Caches

The Top-Down analysis of execution cycles also suggests an unusually high fraction of stalls due to instruction caches (the majority of front-end bound slots in Figure 3). Many such stalls are so severe that they completely drain the deep buffers in a core's front end, causing full instruction starvation. We pinpointed this mainly to a large number of high-penalty L2 instruction misses: at 5 to 20 misses per kilo instruction (MPKI), they were 10 times the highest seen in SPEC CPU2006, and more than 1.5 times the highest observed in other scale-out workloads.[2]

The main reason for such high miss rates is simply the large code footprint of WSC applications. Binaries of hundreds of Mbytes are common and often without significant hotspots. Thus, instruction caches have to deal with large code working sets—lots of lukewarm instructions. This is especially problematic in the L2 cache, where instructions compete for cache capacity with the data stream, which typically also has a large working set. As a result, instructions are often fetched from the shared L3 cache, whose latency cannot always be hidden by front-end buffers.

This instruction cache problem is also one in the making. We demonstrate this by estimating the historical growth in instruction-cache working set sizes of datacenter binaries over a period of 30 months. (For details on the estimation methodology, see our extended paper.[6])

Actively developed binaries' working sets grow at alarming rates and outpace the growth of instruction-cache capacity. For example, Figure 5 shows one search application whose instruction footprint grows at

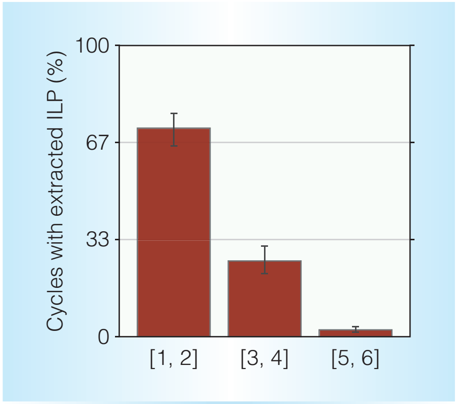

Figure 4. Extracted instruction-level parallelism (ILP) is bimodally distributed. Bursts of computation that use three or more execution ports on a six-wide machine account for 28 percent of all cycles.

more than 25 percent per year, eventually reaching four to five times the largest one seen in SPEC CPU2006 (400.perlbench). Using the measured working set for 400.perlbench as a calibration reference suggests that WSC footprints encroach 1 Mbyte. This is significantly larger than midlevel caches in current server processors (Intel: 256 Kbytes, AMD: 512 Kbytes, IBM: 512 Kbytes), which also have to be shared with typical multi-gigabyte data working sets.

Long-Term Implications

From a long-term perspective, the slowdown of general-purpose performance scaling will undoubtedly cause a stronger emphasis on performance monitoring and optimization at all levels of the stack. In this article, we showcase the GWP tool, which brings performance monitoring to a warehouse scale, and demonstrate the qualitatively new types of studies and optimization opportunities that it enables.

GWP-collected microarchitectural profiles suggest several directions to maintain the rate of performance improvement for general-purpose server CPUs. For example, the lack of a single dominant microarchitectural bottleneck, when combined with the large fraction of memory-latency-bound cycles, indicates that wider simultaneous multithreading can be efficient. Similarly, the high incidence of instruction misses and the growing instruction-cache working sets encourage more emphasis on reducing or protecting

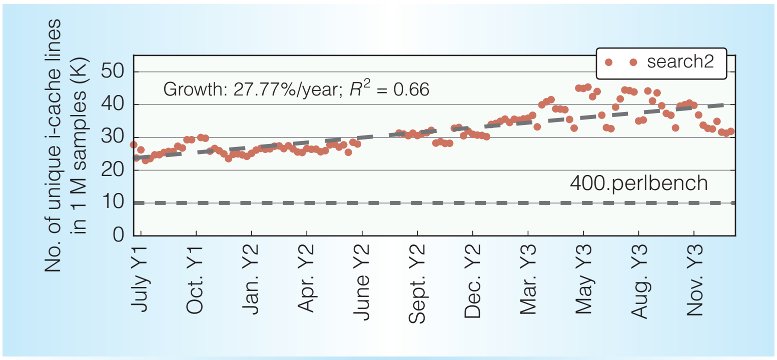

Figure 5. Estimated instruction-cache footprints are getting progressively larger over time for a search binary under active development. They have already exceeded both the SPEC-like working set sizes and the processors' midlevel cache sizes by several fold.

instruction-cache working sets—whether through techniques like instruction prefetching or instruction/data partitioning in shared midlevel caches. Some of our findings corroborate prior work on isolated benchmarking.[2] There is immense value in validating benchmarks with real usage patterns. Furthermore, benchmarking efforts often underestimate the severity of bottlenecks in production-grade codes (for example, instruction-cache pressure and flat execution profiles) and the engineering effort required to eliminate them.

In the longer term, the confluence of technology trends points steadily toward hardware specialization. Continued transistor-density increases, coupled with the end of Dennard scaling, result in the inability to power a whole chip at maximum performance—the problem known as *dark silicon*. This is old news in mobile processors, in which the majority of silicon area is dedicated to more efficient specialized blocks,[7] but server chips are still predominantly general purpose. Profiling efforts like the one we presented are the necessary first step in identifying the building blocks that will make up future accelerator-rich designs in the datacenter.

Our datacenter tax results suggest a qualitatively different approach to these future accelerator efforts. Datacenter hardware specialization so far has mainly focused on deep acceleration—identifying killer applications and optimizing their most costly kernels.[8] The diversification of workloads that we observed, coupled with the ubiquity of the datacenter tax suggest an alternative interapplication approach called *broad acceleration,* which speeds up shared lower-level routines. We expect significant academic and industrial interest in this approach, especially fueled by the growth of public clouds, which give orders of magnitude more developers access to warehouse-scale resources, and therefore breed even more workload diversity than what we have observed.

Such an approach is not without challenges that further research will need to address. For example, many of the routines in the datacenter tax are already well-optimized in software—a typical malloc() call takes only 20 to 30 CPU cycles on current-generation general-purpose processors—and potential accelerators must be tightly integrated with the surrounding code to improve on that. More broadly, calls to tax routines tend to be frequent, fast, and interspersed between other application code. Thus, accelerating them would require different techniques than those used in throughput-based specialized blocks, such as the ones used for image processing. It is up to the community to find them in the quest of ever more efficient computing at scale.

WSC workloads demonstrate significant diversity, both in terms of the applications themselves and within each individual one. Our measurements of Google's server fleet show a common microarchitectural signature for WSC applications—low IPC, large instruction footprints, bimodal ILP, and a preference for latency over bandwidth—which should influence future processors for the datacenter. Future designs should carefully balance wide, out-of-order cores with potentially wider simultaneous multithreading; optimize the cache hierarchy for decreasing and protecting instruction working sets; and potentially shift system balance away from memory bandwidth and toward latency.

Looking for longer-term performance optimizations, we identified ubiquitous low-level functions (the datacenter tax) that are shared across thousands of applications. Given the increasing popularity of specialized hardware, tax components are ideal candidates for specific accelerators in a future server system on a chip. MICRO

References

1. Perfkit benchmarker, 2015; http://perfkitbenchmarker.org.
2. M. Ferdman et al., "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," *Proc. 17th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 37–48.
3. G. Ren et al., "Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers," *IEEE Micro*, vol. 30, no. 4, 2010, pp. 65–79.
4. D. Chen, D.X. Li, and T. Moseley, "AutoFDO: Automatic Feedback-Directed Optimization for Warehouse-Scale Applications," *Proc. Int'l Symp. Code Generation and Optimization*, 2016, pp. 12–23.
5. A. Yasin, "A Top-Down Method for Performance Analysis and Counters Architecture," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software*, 2014, pp. 35–44.
6. S. Kanev et al., "Profiling a Warehouse-Scale Computer," *Proc. 42nd Ann. Int'l Symp. Computer Architecture*, 2015, pp. 158–169.
7. Y.S. Shao et al., "The Aladdin Approach to Accelerator Design and Modeling," *IEEE Micro*, vol. 35, no. 3, 2015, pp. 58–70.
8. A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *Proc. ACM/IEEE 41st Int'l Symp. Computer Architecture*, 2014, pp. 13–24.

Svilen Kanev is a PhD candidate at Harvard University and a tenured intern at Google, where he performed the research for this article. His research interests include performance analysis and modeling, datacenter hardware, and anything spanning the hardware-software boundary. Kanev received an MSc in computer science from Harvard University. Contact him at skanev@eecs.harvard.edu.

Juan Pablo Darago is a software engineer at Google Switzerland, where he works on infrastructure for Google Shopping. His research interests include optimization, high-performance computing, and computer architecture. Darago has an MSc in computer science from the University of Buenos Aires. Contact him at jpdarago@google.com.

Kim Hazelwood is a senior engineering manager in the Infrastructure Division at Facebook. Her research interests include datacenter-scale performance analysis. Hazelwood received a PhD in computer science from Harvard University. She performed the research for this article while she was a software engineer at Google. Contact her at kimhazelwood@fb.com.

Parthasarathy Ranganathan is a principal engineer at Google, where he is designing their next-generation systems. His research interests include systems architecture and management, power management, and energy efficiency for servers and datacenters. Ranganathan received a PhD in computer engineering from Rice University. He is a Fellow of IEEE and ACM. Contact him at parthas@google.com.

Tipp Moseley is a staff engineer at Google, where he works on datacenter-scale performance analysis. His research interests include compilers, operating systems, performance analysis, runtime systems, fault tolerance, and optimized lock-free data structures. Moseley received a PhD in computer science from the University of Colorado at Boulder. Contact him at tipp@google.com.

Gu-Yeon Wei is a Gordon McKay Professor of Electrical Engineering and Computer Science at Harvard University. His research interests include mixed-signal integrated circuits, computer architecture, and runtime software, looking for cross-layer opportunities to develop energy-efficient systems. Wei received a PhD in electrical engineering from Stanford University. Contact him at guyeon@eecs.harvard.edu.

David Brooks is the Haley Family Professor of Computer Science at Harvard University. His research interests include architectural and software approaches to address power, thermal, and reliability issues for embedded and high-performance computing systems. Brooks received a PhD in electrical engineering from Princeton University. Contact him at dbrooks@eecs.harvard.edu.