

# Statement of Research Interests

Michael Ferdman

<http://www.ece.cmu.edu/~mferdman/>

Cloud computing has emerged as the dominant platform for delivering scalable online services. Driven by the rapidly expanding global client base and the proliferation of online services, cloud operators are constantly building new datacenters and powering up more servers. However, with modern datacenters already occupying football-field sized warehouses and consuming megawatts of electrical power, the current datacenter expansion trends are unsustainable. My research interests target software and hardware co-design of high-performance, power-efficient, and compact servers to enable sustainable expansion of cloud computing to the global scale.

My work spans the fields of computer architecture and computer systems, with particular emphasis on the performance and efficiency of server architectures. Within this theme, my research objective is to understand the properties and interactions of server application software, operating systems, networks, and processor architecture, to drive the design of future systems. My goal is to advance server designs, improving per-server performance and energy efficiency to reduce the aggregate number of servers and electrical power needed by future datacenters.

## Graduate Research and Contributions

My research has focused on the performance, scalability, and efficiency of server systems. While most computing applications are centered around performing calculations, server applications are dominated by the movement of data. As a result, the memory hierarchy is the focal point of server systems. My graduate work has addressed the key challenges in the design of memory hierarchies for servers, eliminating fundamental performance bottlenecks and finding elegant solutions to long-standing design and scalability challenges.

### ***Proactive Instruction Caches: eliminating the instruction-access bottleneck in servers***

The performance of cloud and server systems is plagued by high instruction-access delays. Due to their large application, library, and OS instruction footprints, server systems experience frequent instruction-cache misses, losing performance and wasting resources while the processor remains idle, waiting for instructions to be retrieved from the lower levels of the memory hierarchy.

My work exploited the observation that instruction accesses can be predicted because applications execute code in long repetitive sequences [MICRO'08]. The mechanisms I proposed record access sequences, detect when a previously-recorded sequence repeats, and predict the upcoming instructions by replaying a recorded sequence. As part of this work, I identified performance-enhancing micro-architectural techniques that disrupt repetition in the memory system, paradoxically limiting the effectiveness of instruction prefetchers [MICRO'11]. My dissertation, a culmination of this work, proposes the Proactive Instruction Cache, a mechanism to leverage instruction access repetition to improve server performance and efficiency by eliminating the instruction-access bottleneck.

### ***Reactive NUCA: leveraging program behavior to improve server performance***

The advent of large on-chip caches created a challenge of how to effectively manage the cache capacity. Large working sets favor sharing cache capacity among processor cores. However, large shared caches require long slow wires at the circuit level, which degrade performance. Alternatively, cache capacity can be partitioned, enabling the use of fast short wires. However, independent partitions may cache the same data, diminishing performance due to reduced aggregate capacity. Because cache capacity and latency are both critical to server performance, complex hardware heuristics were proposed that try to balance between the two organizations.

Nikos Hardavellas and I observed that memory accesses belong to distinct classes (e.g., instructions, stack, shared heap), where each class is amenable to a different on-chip cache placement policy. Based on this observation, we proposed Reactive NUCA [ISCA'09], a distributed cache architecture to maximize effective use of the cache capacity. Reactive NUCA leverages the OS to accurately classify memory pages and perform near-optimal cache partitioning and block placement. Reactive NUCA avoids complex hardware and heuristics, achieving peak cache performance through a combination of simple software and hardware, effectively solving the long-standing cache organization challenge. This work was recognized by IEEE Micro as one of the *top computer architecture publications of 2009*.

### *Cuckoo Directories: power- and area-efficient cache coherence*

Multi-threaded server applications rely on fine-grained and fast hardware cache coherence. Because multiple on-chip caches may contain multiple copies of a datum, a coherence directory tracks all cached copies to enable invalidation of stale copies when the datum is updated in one of the caches. However, traditional coherence directory designs are inefficient, requiring either sparsely populated tables or content-addressable memories to keep track of the cached copies, resulting in either large or power-hungry hardware structures.

I proposed the Cuckoo Directory [HPCA'11], a power- and area-efficient scalable directory organization. The Cuckoo Directory borrows heavily from Cuckoo Hashing, a dense-storage software hashing technique. Rather than significantly over-provisioning storage capacity to avoid storage conflicts in a traditional lookup table, the Cuckoo Directory uses the Cuckoo Hashing algorithm to relocate conflicting entries within the directory to alternate non-conflicting locations. Leveraging the mathematically-robust properties of Cuckoo Hashing enables compact and power-efficient hardware coherence directories with predictable asymptotic behavior and without degenerate cases, improving over the state-of-the-art directory design without significantly increasing hardware complexity. This work was selected by the HPCA'11 program committee for presentation at the *best student papers* session.

### *Clearing the Clouds: optimally-efficient CPUs for scale-out server workloads*

In addition to my memory system work, I am a leading member of the architecture group in EuroCloud, a European Commission research project that joins industry and academia to investigate efficient server design. Server efficiency is at the forefront of all new datacenter designs, as it dictates the amount of computational resources that a datacenter will deliver for the existing space and power budget. I led an effort to characterize the execution behavior of a wide range of scale-out cloud server applications running on modern servers [ASPLOS'12].

Our analysis empirically showed that today's predominant processor architecture is inefficient for running cloud applications. Like traditional servers, cloud services are dominated by data movement rather than computation; as a result, at the micro-architectural level, cloud application performance is primarily sensitive to memory-access latency. However, modern processors devote most power and on-chip area to computation and high-bandwidth data access, in many cases at a cost to memory-access latency. Our analysis has enabled accurate models of cloud server performance, which form the basis of the EuroCloud design methodology for server processors [submitted to ISCA'12]. Additionally, we are publicly releasing the benchmarks, which are expected to inspire and guide further cloud application characterization, influencing the selection and the design of processors for future data centers.

## **Future Research Agenda**

The internet adoption rate is growing exponentially in most countries around the globe, while existing users are increasing their reliance on online services through a growing number of wired and mobile devices. Cloud services already run on millions of servers housed in multi-megawatt facilities around the globe. Google alone operates over a million servers that consume 260 megawatts, exceeding the output of many of the operating nuclear power plants; there are dozens of global-scale cloud operators today.

The vast amount of electrical power and datacenter space consumed by inefficient server designs renders the current datacenter expansion trends unsustainable. As a result, we are in an environment where innovation in space- and power-efficiency are actively sought after, offering short- and long-term opportunities to make considerable impact. Due to the vast opportunities that exist in improving server efficiency, research in this field has the potential to influence every aspect of future server systems. I plan to focus my work on innovation in server efficiency, with the goal of driving datacenter server architecture toward sustainable trends.

### *Short-Circuit Execution: skipping redundant request-to-request computation in server systems*

Repetitive execution of the same code is prevalent in all software; however, whereas traditional applications repeat execution over constantly changing intermediate results, datacenter workloads are dominated by redundant execution of the same operations with identical outcomes. Multitudes of clients initiate identical or similar requests, receiving nearly identical responses after the server executes practically the same functions for each request. To leverage this repetition, aggressive software caching strategies can be employed and request routing can be locality-optimized across the cloud. However, in many cases, high implementation complexity and performance cost of software caching lead to simplified implementations that re-compute the entire response for each request. Moreover, in cases where caching is effective, caching itself becomes a cloud service that consumes significant datacenter resources to repetitively serve identical requests with precisely the same responses.

The cost of detecting and leveraging execution redundancy in software is high and impractical. However, techniques that combine hardware and software can efficiently detect and avoid redundancy. Hardware mechanisms

can detect redundancy and short-circuit execution, skipping unnecessary computation by substituting a previously observed result. More complex mechanisms can use alternate fast-path code inserted by compilers and runtime systems to cooperate with the hardware in detecting and exploiting redundancy in server requests through partial specialization of functions, completing client requests with fewer instructions by reusing previously computed results when possible. Identifying, designing, and prototyping short-circuit execution mechanisms offers many opportunities for interdisciplinary research. I believe that exploiting execution redundancy enables large improvements in server performance and energy efficiency. I also believe that the low cost and low risk nature of these techniques is likely to lead to near-term influence on server designs.

### ***Specialization: offloading repetitive computation to efficient hardware***

Long-term sustainable expansion of datacenter performance requires a change of course, a careful re-design of server systems for performance efficiency. Today's servers are general-purpose machines, designed to achieve software flexibility and, to a large extent, built without consideration for space constraints and power efficiency. Practically all server components in use today (application software, operating systems, server hardware, and processors) are designed in isolation, in many cases with much broader goals than the server scope. On the other hand, embedded-device designs specialize software and hardware from the ground up, achieving phenomenal space and power efficiency. Taking example from the embedded devices, I believe that high-performance, power-efficient, and compact servers can be achieved through specialization. Moving work to specialized hardware will free the processors to perform unique and complex tasks, yielding significant performance improvements and simultaneously reducing server energy consumption by an order of magnitude.

Among the prevalent challenges standing in the way of server specialization is the lack of understanding of what operations to specialize. Unlike the embedded-device domain, where specialization can often be performed through intuitive understanding and automated profiling of the software, the selection of specialized server components must be forward looking; server hardware development cycles take several years, requiring specialized server components to be applicable to a wide range of future software. Beyond understanding what to specialize, a large array of equally complex questions must be answered to make specialization possible. How will software interface with specialized components? How must software development and debugging practices be adapted to handle systems with specialized components? As software evolves, how can specialized components be adapted without losing efficiency? Answers to these and many other questions require systems research and experimentation.

My vision is to approach the server specialization challenges by understanding the functionality of server systems as a whole, targeting the specific needs of cloud workloads by modifying all necessary parts of the system, from the application to the processor architecture. I believe that researchers are uniquely poised to lead this effort, as it requires undertaking large-scale cross-disciplinary research. Although many research questions must be answered to achieve this vision, there exists a clear and immediate need to take on these challenges to enable global scalability of future datacenters.

## **Referenced Publications**

[MICRO'08] **Temporal Instruction Fetch Streaming.** *Michael Ferdman, Thomas F. Wenisch, Anastasia Ailamaki, Babak Falsafi, Andreas Moshovos, In 41st Annual IEEE/ACM International Symposium on Microarchitecture, 2008.*

[MICRO'11] **Proactive Instruction Fetch.** *Michael Ferdman, Cansu Kaynak, Babak Falsafi, In 44th Annual IEEE/ACM International Symposium on Microarchitecture, 2011.*

[ISCA'09] **Reactive NUCA: near-optimal block placement and replication in distributed caches.** *Nikos Hardavellas, Michael Ferdman, Babak Falsafi, Anastasia Ailamaki, In 36th International Symposium on Computer Architecture, 2009.*

[HPCA'11] **Cuckoo Directory: A Scalable Directory for Many-Core Systems.** *Michael Ferdman, Pejman Lotfi-Kamran, Ken Balet, Babak Falsafi, In 17th IEEE International Symposium on High Performance Computer Architecture, 2011.*

[ASPLOS'12] **Clearing the Clouds: Study of Emerging Workloads on Modern Hardware.** *Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafae, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, Babak Falsafi, In 17th International Conference on Architectural Support for Programming Languages and Operating Systems, 2012.*