

Statement of Research Interests

Brian T. Gold

Carnegie Mellon University

My research focus is computer architecture and its broader impact on computer systems, including compilers, operating systems, programming languages, and I/O architectures. In particular, my research goals address the following questions:

- How do we ensure computing systems are **reliable** and **available** as errors in computation occur with increasing frequency and device fabrication itself becomes less reliable?
- How do we improve **power and thermal modeling** of increasingly complex software and hardware systems?
- How do we increase **programmability** for future, highly concurrent and heterogeneous architectures?

My dissertation research developed reliable and available servers capable of running unmodified shared-memory applications. In collaboration with my colleagues at CMU, I proposed the TRUSS (Total Reliability Using Scalable Servers) architecture [3], which allows designers to separately address processor, memory, and I/O reliability. My thesis work shows that redundant processor pairs can be distributed across chips without requiring lockstep operation or incurring significant performance overhead [2].

In addition to my dissertation work, I have recently developed fast and flexible models of power and temperature for data center architectures [4], and I led a study of fine grain parallelism in database operators running on a highly concurrent network processor [1]. These projects are in the early stages with significant opportunity for future research contributions. In the rest of this statement, I summarize my prior contributions and present future research plans.

Dissertation Research: Reliable, Available Shared-Memory Servers

The gigascale integration trend in semiconductor technology produces circuits with significant vulnerability to transient error (such as that caused by cosmic radiation) and permanent failure (such as that from device wearout). Conventional approaches to redundant instruction execution require drastic modifications in hardware and/or software to detect and recover from errors. In contrast, my research enables scalable, commodity shared-memory servers to benefit from the reliability and availability of redundant execution while requiring no changes to application software and little additional hardware.

My work proposes solutions to the following critical challenges:

- **Unbounded error-detection latency and limited comparison bandwidth.** Large on-chip caches allow errors to hide for millions of clock cycles, precluding existing detection mechanisms that wait for errors to reach the chip boundary (e.g., Tandem NonStop). Furthermore, limited off-chip bandwidth prohibits the explicit checking of all instruction results. We proposed Fingerprinting [5], a hardware mechanism that summarizes a processor's execution history into a hash-based signature. Although aliasing prevents perfect error coverage, we show that even a simple hash design can provide sufficient coverage for future designs.
- **Non-deterministic operation.** Historically, dual-modular redundant (DMR) architectures have relied on lockstep operation, but modern high-performance processors contain sufficient non-determinism (e.g., arbitration across clock domains) to make lockstep cost-prohibitive. We observe that in the common case, replica processors will execute the same dynamic instruction stream; only races for shared-memory locations can cause execution divergence. Leveraging this observation, we proposed the Reunion execution model [6], a formal framework that provides redundant execution without requiring lockstep. Reunion preserves the existing memory system interface, coherence protocols, and memory consistency models.

My thesis work proposes an implementation of the Reunion model that supports multi-chip redundancy, where cores are paired on separate chips to provide high availability in the presence of common-mode failures (e.g., faults affecting a shared bus). The principle challenge of multi-chip redundancy is that the processor must buffer unchecked instructions past the conventional retirement point in the pipeline. My thesis shows that

existing mechanisms for deep speculation—hardware transactional memory, memory-ordering speculation, large instruction-window processors, etc.—provide sufficient buffering for redundant execution in the Reunion model [2].

- **Vulnerability to chip failure.** Conventional lockstep-based redundancy (e.g., Tandem NonStop) could rely on replicas holding identical state on chip; thus, if one chip fails catastrophically, the other is an identical replacement. In the Reunion execution model, replica cores access the memory system independently, leaving the system vulnerable to chip failure anytime a unique value is passed to one replica but not the other. In the final component of my thesis research, I show that simple extensions to the coherence protocol can close this window of vulnerability. Using model checking, I formally verified these extensions in the context of an existing, state-of-the-art coherence protocol (publication in preparation).

Future Directions. Taken together, my thesis contributions provide future commodity architectures with support for high reliability and availability with little custom hardware. The principle limitation of this work is that *all* instructions are duplicated, so while few design modifications are required, there is an inherent 2X cost in area and power. My future research plans in this area are motivated by the observation that not all operations require full duplication, even for business-critical enterprise applications.

Recent work has shown that less than 5% of all latch and flip-flop errors go undetected at the program level and yet change program outcome (e.g., silent data corruption). In addition, enterprise applications often contain extensive recovery features for errors that are detected; for example, recovery logs in database systems and retry mechanisms in most networking applications. I intend to examine common enterprise application domains (e.g., database and web servers) for ways to improve fault detection for program paths vulnerable to silent data corruption, thereby mitigating the large overheads of full-time duplication.

Power and Temperature Modeling

Power and temperature are first-order design constraints for modern microprocessors. Although researchers have developed accurate models for power consumption and temperature, these require complex and often time-consuming simulations. Making system-wide decisions—such as thread or I/O scheduling—to improve power and thermal efficiency requires simpler models that capture the dominant effects of proposed techniques without costly simulation. I have taken two initial steps in this area by developing queuing models that incorporate power and temperature as performance metrics.

First, I developed a simple power model for a server that enters a low-power state whenever its work queue is empty [4]. The model treats a server as an M/G/1 queue, where the transition time into and out of an idle state is an ‘exceptional first service’ in queuing-theory terminology. In collaboration with researchers at the University of Michigan, we applied this model to traces collected from enterprise data centers and showed that trends in system design, such as solid-state disks and low-power DRAM modes, enable data centers to reduce power consumption by 70% over current state-of-the-art techniques [4]. I plan to extend this work to investigate coordinated power management across multi-cores (an M/G/k server) and approximations for non-Poisson arrival processes.

Second, I have developed queue-based models of server temperature that can be solved in closed form (for M/M/1 servers) or numerically with matrix-analytic methods (for M/G/1). Eventually, this work-in-progress will enable researchers to treat temperature as a performance metric like throughput or response time. My initial models can already answer questions such as “What is the average temperature for a given workload?” and “What fraction of time is temperature over 45C?” My future research plan includes continued development, validation, and application of these models. In particular, I plan to examine thermal-aware load balancing and efficient task assignment in data centers.

Programming Models for Emerging Architectures

Architects of future multi-core chips will rely on increasingly heterogeneous core designs, both in terms of performance capabilities (e.g., out-of-order vs. in-order) and execution models (e.g., conventional ISA vs. graphics-specific pipeline vs. reconfigurable fabric). Heterogeneous architectures require software that is aware of hardware

resources, which leads to a classic dilemma: who will write software for non-existent hardware, and why should we build new hardware for which no software exists?

Fortunately, existing network processors—chips designed specifically for packet processing in high-end routers—make extensive use of heterogeneous multi-core architectures. For example, the Intel IXP-series platform features 8 to 16 multi-threaded cores for common packet-processing operations and a single conventional core for rare or out-of-band tasks.

In joint work with database researchers at CMU, I led a study [1] that showed common database operators (hash joins, index scans, etc.) were 10X more power efficient on a network processor as compared to a high-performance out-of-order core. Building a complete database, however, requires more than highly parallel operations. Query optimizers, storage managers (e.g., locking and buffer-pool management), and file systems may benefit more from high single-thread performance in a conventional processor. I plan to extend the framework I developed to support a complete database system and study placement and scheduling algorithms on future heterogeneous multi-cores. Although databases are an important workload class on their own, I expect this work to provide insight into mapping other general-purpose computing problems.

References

- [1] **Brian T. Gold**, Anastasia Ailamaki, Larry Huston, and Babak Falsafi. Accelerating Database Operators Using a Network Processor. In *Proceedings of 1st International Workshop on Data Management on New Hardware (DaMoN)*, June 2005.
- [2] **Brian T. Gold**, Babak Falsafi, James C. Hoe, and Ken Mai. Redac: Distributed, Asynchronous Redundancy in Shared-Memory Servers. *In Submission*, November 2008. Also available as *Computer Architecture Lab at Carnegie Mellon (CALCM) Technical Report 2008-002*.
- [3] **Brian T. Gold**, Jangwoo Kim, Jared C. Smolens, Eric S. Chung, Vasileios Liaskovitis, Eriko Nurvitadhi, Babak Falsafi, James C. Hoe, and Andreas G. Nowatzky. TRUSS: A Reliable, Scalable Server Architecture. In *IEEE Micro Special Issue on Reliability-Aware Microarchitectures*, November-December 2005.
- [4] David Meisner, **Brian T. Gold**, and Thomas F. Wensich. PowerNap: Eliminating Server Idle Power. To Appear in *14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, March 2009.
- [5] Jared C. Smolens, **Brian T. Gold**, Jangwoo Kim, Babak Falsafi, James C. Hoe, and Andreas G. Nowatzky. Fingerprinting: Bounding Soft-Error Detection Latency and Bandwidth. In *Proceedings of 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI)*, October 2004. Also in *IEEE Micro 'Top Picks' from 2004*.
- [6] Jared C. Smolens, **Brian T. Gold**, Babak Falsafi, and James C. Hoe. Reunion: Complexity-Effective Multicore Redundancy. In *Proceedings of 39th International Symposium on Microarchitecture (MICRO-39)*, December 2006.