EE194 Final Project Ideas

Riley Wood

March 30, 2016

Possible Research Questions

1. Efficiency of Legacy and Modern Concurrent Programming Languages

   - With multiprocessor computers becoming the norm, it has been necessary for programming languages to adopt concurrency constructs. Languages like C/C++ generally use locks to protect against data races and deadlock. However, locks in these languages suffer from needing to be globally scoped; abstracting away a lock in a library function means the programmer risks deadlocking the program. New languages like Rust address these shortcomings with different constructs. Specifically, Rust introduces the concept of ownership of data: data can only be owned and thus mutated in one place at a time, thereby avoiding data races and deadlock, but also allowing for abstraction. The question is, do these new constructs slow down execution at all? I would like to run concurrent benchmarks implemented in C and Rust through SNIPER and look for any significant differences in performance, in an effort to discover what if any differences in performance arise because of Rust's new concurrency constructs.

2. When To Switch From Ring Cache to Mesh Network Cache?

   - Multicore systems often organize their L2 caches in a ring structure. This means coherence messages only need to hop at most $\frac{N}{2}$ times to reach their destination, where N is the number of cores. At a certain point, though, the number of cores grows the cache coherence overhead too much, and it is preferable to use a mesh network. I would be interested in seeing at what point a mesh network becomes the preferable architecture: how many CPUs under which workloads with what other combinations of sytem parameters make a ring no longer viable?

3. Comparing Cache Coherence Protocols

   - I would like to see how various cache coherence protocols perform relative to one another across several workloads. I would look at protocols like MESI, MSI, MOSI, MOESI, MERSI, MESIF, write-once, and others. I would compare them based on metrics such as IPC, average memory miss penalty, average memory access time, misses per 1K instructions, and other memory-focused metrics.