Abstract
Interactive services such as Web search, recommendations, games, and finance must respond quickly to satisfy customers. Achieving this goal requires optimizing tail (e.g., 99th+\ percentile) latency. Unfortunately, interactive services often show variability in service demand, with the tail latency being defined by a small number of long requests. Providers can keep the tail latency under a target by giving extra resources to long requests. This is challenging because (1) service demand is unknown when requests arrive; (2) blindly giving extra resources to all requests quickly oversubscribes resources; and (3) giving extra resources to the numerous short requests does not improve tail latency.

In this dissertation, we propose judicious allocation of processor cores in multicore servers to interactive service requests for optimizing tail latency. In the first part of the dissertation, we introduce Few-to-Many (FM) incremental parallelization for current multicore servers. FM dynamically increases parallelism to utilize idle cores and reduce tail latency. FM uses request service demand profiles and total number of cores in an offline phase to compute a policy, represented as an interval table, which specifies when and how much software parallelism to add. At runtime, FM adds parallelism to use extra cores as specified by the interval table indexed by dynamic system load and request execution time progress. The longer a request executes, the more parallelism (and higher number of cores) FM allocates to it. We evaluate FM in Lucene, an open-source enterprise search engine, and in Bing, a commercial Web search engine. FM improves the 99th percentile response time up to 32% in Lucene and up to 26% in Bing, compared to prior state-of-the-art parallelization. Compared to running requests sequentially in Bing, FM improves tail latency by a factor of two.

In the second part, we demonstrate how to tradeoff tail latency and energy consumption on emerging heterogeneous processors with fast cores and
energy-efficient slow cores on the same chip. We introduce Adaptive Slow-to-Fast (AS2F) scheduling algorithm, which migrates requests from slow to fast cores to give powerful cores to long requests. We use control theory to design threshold-based migration policies that manage latency and energy efficiency trade-offs for heterogeneous processors with any number of core types. We demonstrate several configurations that (1) improve energy-efficiency while meeting a target tail latency, (2) optimize average and tail latency, or (3) optimize energy only. AS2F can judiciously exploit slower cores to reduce energy by up to a factor of 2.1 while meeting the tail latency target, compared to only optimizing latency. This configuration consumes only 20 more energy than an oracular policy that has perfect knowledge of request lengths and arrival rate.

Overall, our experience and results show that FM and AS2F can be powerful techniques for managing tail latency and energy, smartly utilizing hardware resources, and exploiting emerging heterogeneous processors.