7 CONCLUSION

Power and energy consumption are first-class constraints today for computer system design and operations. One of the reasons for power (and energy) waste in computers is that they continue to consume power (and energy) even when they are idle. Barroso and Hölzle [22] proposed Energy Proportionality (EP) as a model for ideal system behavior. In this model, the ideal system should use power in proportion to utilization (performance or load served). The EP model has been very influential in encouraging system designers to improve energy efficiency of their systems.

The EP model for ideal behavior holds true for systems with fixed resources. However, modern computers have reconfigurable resources. Such reconfigurable systems may also exhibit super-proportional behavior. The advent of such super-proportional systems is a game changer. In these systems, one can get more performance for the power consumed (or equivalently, more work done for the energy consumed) at intermediate loads than what one could get in an "ideal" EP system. EP no longer characterizes the maximum energy efficiency that super-proportional systems can attain. In fact, aiming for EP behavior can be significantly suboptimal except at very low loads. In Chapter 2 we proposed a new ideal model, EOP (Energy Optimal Proportional), that characterizes maximum energy efficiency for all systems, both super-proportional and otherwise.

EOP is an ideal model for system designers. It describes a lower bound on the energy consumption (upper bound on the energy efficiency) that the given system can achieve. Any shortfall from this maximum efficiency at any operating load indicates potential for further improvement. System designers can use this characterization to both improve the upper bound as well as reduce shortfall, that is, make the Pareto frontier (Dynamic EO) closer to EOP, throughout the operating range.

For a given machine, system operators should aim to constrain the system to operate at or close to the Pareto frontier (Dynamic EO). They should use operating policies, or governors, that manage the system to satisfy different SLAs—to maximize energy efficiency (SLAee), to maximize performance within a given power budget (SLApower), to maximize power savings while meeting a given performance constraint (SLAperf), etc. We develop new governors that configure processor frequency and cache prefetching (Chapter 3) or cache capacity (Chapter 5).

We use a profiling-based approach to decide prefetch settings (enable/disable). To predict cache performance, we first sample the cache access stream and use a Bloom filter to determine the reuse distance distribution. The, our analytical models, described in Chapter 4, read the estimated reuse distribution to predict cache performance for other cache configurations that differ in the number of sets or ways. Our method is decoupled from the current cache configuration. The hardware requirement of our approach is less than that of shadow tags+way counters for large caches and is also more flexible in that it can also estimate performance for target cache configurations having different numbers of sets in addition to those with different associativities.

Being able to dynamically reconfigure (last-level) cache capacity opens up the possibility of "lowering" the Pareto frontier resulting in more energy-efficient system operation. In Chapter 5 we developed a governor that simultaneously configures processor frequency and cache capacity. Our experiments in that chapter only considered SLApower, but it can be easily extended to target the other SLAs as well. We showed that our analytical models that drive reconfiguration decisions work quite well compared to an idealistic/oracular approach.

In datacenters, energy is wasted by both the cooling and the IT (compute, networking, etc.) infrastructures. The PUE (Power Usage Effectiveness) metric [16] tracks excess

energy used by the cooling infrastructure. Through intense focus on the problem of reducing energy waste in the cooling infrastructures, modern datacenters have succeeded in having very low PUE values. This in turn makes energy waste in the IT infrastructure, particular servers, a major contributor to overall energy waste in modern datacenters. In Chapter 2 we proposed a new metric, Computational PUE (CPUE), that tracks excess energy used by the servers. Our new Iron Law of Energy decomposes CPUE into three factors—LUE (Load Usage Effectiveness), RUE (Resource Usage Effectiveness), and E_{min}. LUE is affected by inter-server load management decisions and demand-driven load fluctuations. RUE is affected by intra-server resource configurations. E_{min} is affected by system design choices. Making systems more energy efficient will require focused effort on improving each of these aspects. We hope that the Iron Law of Energy will help to drive and focus this effort.

Our work has the following limitations.

- **Single-server systems**: Our experiments in this work used single-server systems. We believe that the concepts and metrics that we introduced, such as EOP, CPUE, LUE, RUE, will continue to hold for multi-server systems as well. However, currently we do not have experimental data to quantitatively compare against the traditional EP model for such systems or how close to Dynamic EO the governors make the systems operate.

- **Long-term adaptations**: Our governors target relatively long-term adaptations of resources. The DVFS and prefetching governors (Chapters 2 and 3) work over hundreds of milliseconds while our cache governors (Chapters 4 and 5) work over seconds of execution time. Cache adaptations over short time intervals may not be attractive due to significant overheads in reconfiguration and subsequent warmup. A short time interval for our DVFS governor is suboptimal (**R(1)**, see Chapter 3).

- **Simulation study**: We use a real system (HS) for our DVFS and prefetching studies (Chapters 2 and 3). Real systems today support dynamic reconfiguration for only a small set of resources. On HS, dynamic reconfiguration of cache sizes is not supported. Some other recent systems [109] support cache space allocation to cores/applications, but are limited in the modes of reconfiguration (it is not clear if both the number of sets and ways can be changed). Moreover, hardware support for inspecting or sampling addresses of cache accesses is missing. Instrumentation-based frameworks [143] may not fully address the needs because usually they do not track OS kernel accesses to the cache. Thus, we use full-system simulation to study cache reconfigurations (Chapters 4 and 5). We believe that simulation is an indispensable tool for fully exploring this space. Simulators are flexible and provide insight but may be inaccurate with respect to real system implementations.

- **Prefetching models**: Our governors in Chapter 3 account for the effects of hardware prefetching and can handle prefetching reconfigurability in terms of enabling or disabling it. Further reconfigurability for prefetching, e.g., dynamically changing prefetch depths and strides, may need analytical modeling for prefetching impact as a function of configuration. It is unfortunate that real systems today do not support such reconfigurability. Our cache models in Chapter 4 and governors in Chapter 5 do not account for potential changes in the address stream due to prefetching reconfigurations. This limitation, however, is not unique to our cache models but affects almost all, if not all, current analytical models for cache performance.

- **Throughput metrics**: Our governors have focused on throughput-based metrics for performance, e.g., BIPS, transactions per second, etc. that may be required to also satisfy some constraints, e.g., in the case of SPECpower. Our governors currently do not focus on latency-based metrics, e.g., response time or tail latency

distributions that may be important for some applications, e.g., interactive online applications, high-frequency trading applications, etc.

Future research will focus on removing these limitations. Recent work [125, 171] has proposed techniques for fast adaptation, but doing so in a workload-agnostic manner and without changes to the underlying OS is hard. Cache reconfigurations will likely be at long-term intervals. Integrating cache performance models with prefetching models [48] seems to be an interesting direction for further exploration.

Current studies have not fully explored simultaneous reconfiguration of multiple different types of knobs within the same server. Our classification scheme in Chapter 6 highlights the potential for future work in the area. One of the challenges in doing such work is the lack of availability of many dynamically reconfigurable knobs in real systems. This makes full-system simulation a necessity for exploring this space. A number of modern systems support reconfiguring the number of logical cores and socket-level DVFS, but fewer systems support per-core DVFS. Only a handful of Xeon models support cache capacity allocation. Hardware support for inspecting cache access streams is absent. Hardware prefetching control is extremely limited (only enable/disable). Some other knobs such as QPI (Quick Path Interconnect) speed and memory speed can be configured only at system boot time. Making these and other knobs dynamically reconfigurable will greatly help OS schedulers to operate systems more efficiently.

A SPECPOWER POWER-PERFORMANCE

Figure A.1 shows the power-performance state space for SPECpower on HS for various configurations chosen statically and fixed throughout the entire run. Each subfigure shows particular combinations of number of cores (4 cores or 1 core) and memory frequency (1600 MHz or 1067 MHz) and all possible DVFS levels for each combination. We keep the default setting of prefetching enabled for all runs.

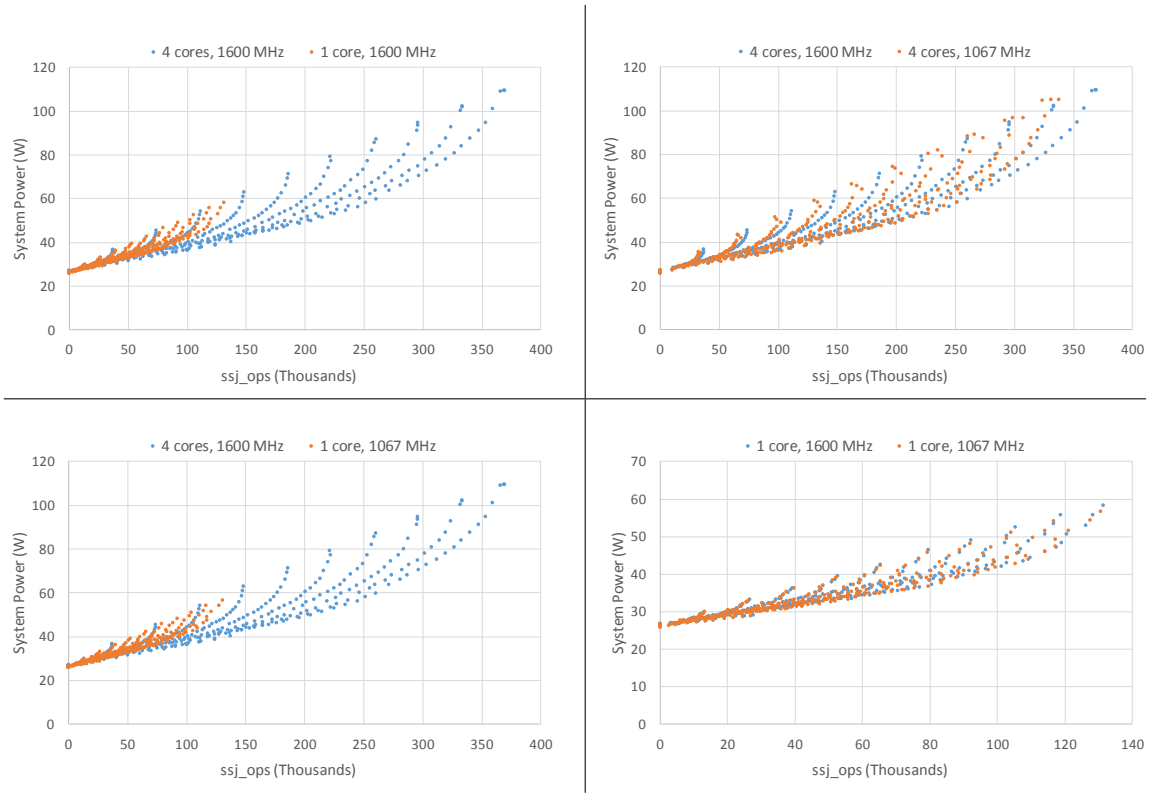

Figure A.1: SPECpower power-performance with different configurations.

Reducing memory frequency affects bandwidth, but has noticeable impact only when there are more active cores to generate memory traffic. Reducing the number of cores or memory frequency reduces the maximum load that can be served, but does not significantly lower the Pareto frontier at low loads.