NSF-WSCS Statement of Interest

1 Introduction

Today's latency-sensitive cloud applications[1] play a critical role; in many cases, fleets of servers are dedicated to running a single instance of these applications [18, 15, 7, 20, 21]. As global data center energy use continues to rise [8, 19, 4, 17], it is critical to find ways to meet the challenging requirements of these applications while reducing their energy use. A recent study has shown that the environmental footprint of hardware manufacturing is also increasing concurrently [8], which stresses the importance of being able to extract more value out of existing software and hardware. Motivated by the rise of this software-hardware dedication, our work presents a generic approach to optimize the efficiency of network applications by taming and controlling its performance and energy trade-offs in an automatic way [9]. We achieve this through the use of a sample efficient machine learning technique, Bayesian optimization [6], to exploit two hardware mechanisms that externally control *interrupt coalescing* and *processor energy settings* and demonstrate improvements over existing mechanisms.

The key insight with which we base our Bayesian optimization approach is that being able to externally control *interrupt coalescing* helps stabilizes application latency periods such it becomes easier to control performance-energy trade-offs and magnify its benefits with *processor energy settings*. We arrived at this insight through a performance and energy study of network applications. From the data collected in our study, we uncover that a software stack has a characteristic performance-energy efficiency profile that is a function of two hardware mechanisms: 1) a NIC's interrupt delay setting (*ITR*) [12, 16, 10] to control coalescing behavior and 2) the processor's Dynamic Voltage Frequency Scaling (*DVFS*) [2, 1] to control its frequency and energy setting. We illustrate the controller's use in exploiting the stable mean demand curve of a publicly available key-value trace [13] to save up to 60% in server energy. Further, we demonstrate the generality of our approach, finding savings up to 36% on applications different from our study (Tailbench [14]) and on radically different hardware platforms released almost a decade apart (i.e. Intel E5-2640-released Q1'12 and Ampere ARMv8 released Q4'21).

2 Energy Study

We present an example result of our energy study of how ITR, DVFS, and OS logic and paths interact together to impact performance and energy trade-offs in Memcached [5].

2.1 Per-Interrupt Log Collection

For the study, we built a per-interrupt logging framework, `intlog` (Acesss to our data and logging scripts can be found at https://github.com/sesa/intlog), in Linux's network device driver. We collect the following data in the NIC's interrupt handler code: received and transmitted bytes, descriptors, sleep state statistics, and current timestamp via `rdtsc` instruction. Additionally, per-core Intel performance monitoring counters (PMCs) capture hardware statistics every millisecond, including instructions, cycles, and last-level cache misses. We use standard Running Average Power Limit (RAPL) hardware registers to read per-package energy values [11]

[1]Examples include kev-value stores, search, and image and speech recognition. The execution of such applications must often meet a specific performance target expressed as a Service Level Agreement (SLA). A common SLA is a 99% tail latency requirement – Eg. 99% of all requests must be completed within some time limit.

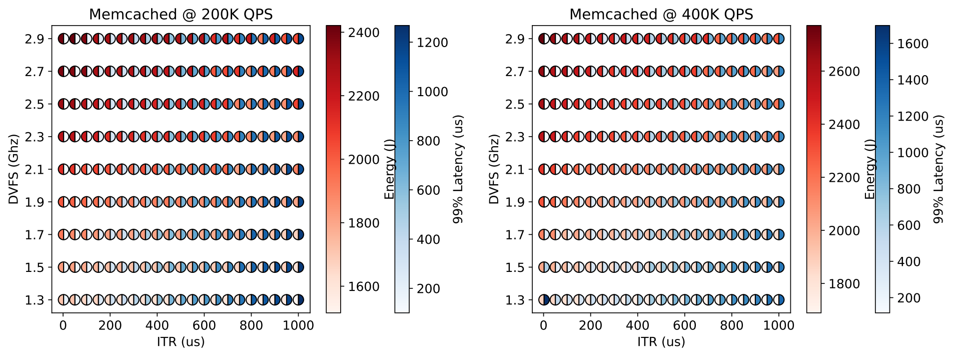
Figure 1: Illustrates the change in energy and 99% latency as different ITR, DVFS pairs are explored for Linux memcached at three different loads of 200K, 400K QPS.

2.2 Revealing ITR and DVFS Performance-Energy Trade-offs

To help build intuition of the impact of specific ITR and DVFS settings on the performance and energy of network applications, we present an example in fig. 1 featuring a Linux memcached server with a load of 200K and 400K requests-per-second (QPS). Using colored gradients, the figure visually represents the effects of each ITR-delay, and DVFS pair; each data point is divided in half, providing insights into their respective impacts on 99% latency and energy. In fig. 1, one can see the trend that as DVFS decreases from 2.9 GHz: the energy gradient becomes lighter, indicating a more pronounced impact on reducing energy use. Simultaneously, increasing ITR horizontally has an immediate effect on increasing measured 99% latency, evident in the darkening of colored gradients. Further, we observe as ITR increases, this induces additional queuing which enables efficient request batching, thereby facilitating additional energy savings. These observations indicate that one can build a controller that toggles combinations of ITR and DVFS to select different operating points and move within this energy space.

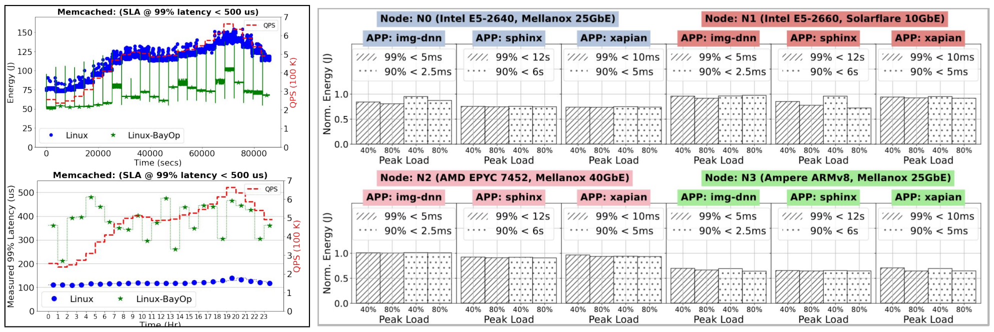
Figure 2: *BayOp* controller applied to various applications and loads.

3 Proof-of-Concept Controller

We built a prototype controller, *Linux-BayOp*, that periodically samples the energy and performance of a live system and uses Bayesian optimization to tune ITR, DVFS to minimize energy while adhering to application SLA objectives. In fig. 2, we applied it to a memcached server over 24 hour period and found it can dynamically adjust to changing load rates and save up to 60% energy over Linux's default policy while maintaing SLA objectives. We then demonstrate its generality across a set of benchmarks from Tailbench [14] while running on a diverse set of hardware nodes from CloudLab [3] and discovered that ITR, DVFS tuning can be agnostically applied in both software and hardware.

4 Future Work

Linux contains many dynamic policies that have their own objectives and are full of implicit and/or hidden heuristics. Our data-driven approach to creating energy efficiency system policies is continuing as part of the Boston Univerity-Red Hat Research Collabotary in two main projects. The first involves improving the energy efficiency of streaming-based application deployments such as Apache Flink where there is a often a dynamic feedback loop that adjusts input rate depending on processing capabilities of downstream worker tasks. In the second we are working on the Kubernetes Efficient Power Level Exporter (Kepler) project to utilize their exported metrics to create energy efficient schedulers.

References

[1] ARM. https://developer.arm.com/documentation/den0013/d/Power-Management.

[2] Dominik Brodowski, Nico Golde, Rafael J. Wysocki, Viresh Kumar. CPU frequency and voltage scaling code in the Linux(TM) kernel. https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt.

[3] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 1–14, July 2019.

[4] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, page 13–23, New York, NY, USA, 2007. Association for Computing Machinery.

[5] Brad Fitzpatrick. Distributed Caching with Memcached. *Linux Journal*, 2004(124):5, August 2004.

[6] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

[7] Akhil Guliani and Michael M. Swift. Per-application power delivery. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing, 2020.

[9] Han Dong. Tuning Linux kernel policies for energy efficiency with machine learning. https://research.redhat.com/blog/article/tuning-linux-kernel-policies-for-energy-efficiency-with-machine-learning/.

[10] Nathan Hanford, Vishal Ahuja, Matthew K. Farrens, Brian Tierney, and Dipak Ghosal. A survey of end-system optimizations for high-speed networks. *ACM Comput. Surv.*, 51(3), July 2018.

[11] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B:System Programming Guide, Part 2. https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-\vol-3b-part-2-manual.pdf.

[12] Intel. Tuning Throughput Performance for Intel® Ethernet Adapters. https://www.intel.com/content/www/us/en/support/articles/000005811/network-and-i-o/ethernet-products.html.

[13] Rashmi Vinayak Juncheng Yang, Yao Yue. A large scale analysis of hundreds of in-memory cache clusters at twitter. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, November 2020.

[14] Harshad Kasture and Daniel Sanchez. Tailbench: a benchmark suite and evaluation methodology for latency-critical applications. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–10, 2016.

[15] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving resource efficiency at scale. *SIGARCH Comput. Archit. News*, 43(3S):450–462, June 2015.

[16] Mellanox. https://community.mellanox.com/s/article/understanding-interrupt-moderation.

[17] Nicola Jones. How to stop data centres from gobbling up the world's electricity. https://www.nature.com/articles/d41586-018-06610-y.

[18] George Prekas, Mia Primorac, Adam Belay, Christos Kozyrakis, and Edouard Bugnion. Energy proportionality and workload consolidation for latency-critical applications. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, page 342–355, New York, NY, USA, 2015. Association for Computing Machinery.

[19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[20] Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, Ben Christensen, Alex Gartrell, Maxim Khutornenko, Sachin Kulkarni, Marcin Pawlowski, Tuomas Pelkonen, Andre Rodrigues, Rounak Tibrewal, Vaishnavi Venkatesan, and Peter Zhang. Twine: A unified cluster management system for shared infrastructure. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 787–803. USENIX Association, November 2020.

[21] Juncheng Yang, Yao Yue, and K. V. Rashmi. A large scale analysis of hundreds of in-memory cache clusters at twitter. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 191–208. USENIX Association, November 2020.