A Framework for Network-Centric ML-Systems Datasets

Yara Awad[*] Boston University [*]*co-first*
Han Dong[*] Boston University [*]*co-first*
Sanskriti Sharma Boston University
Sanjay Arora Red Hat, Inc.
Jonathan Appavoo Boston University

Abstract

The availability of high-fidelity systems datasets is key in designing machine learning (ML) approaches for solving systems problems. This paper is part of an effort to capture, understand, and model the energy and performance behavior of network-driven computation in a data-centric way. We present a multi-terabyte systems-level dataset and a general and extensible implementation of the data logging framework we used to produce it. Our dataset is parameterized by an exhaustive sweep of different software and hardware controls and captures system behavior in a fine-grain, low-level temporal signal that is agnostic to software semantics. We believe that these attributes enable defining ML tasks for systems problems without extensive knowledge of the underlying system.

1 Introduction

Modern hardware components and software stacks expose a large number of parameters that govern internal system operations and interactions. There is a lot of work on defining heuristics to control these parameters [6, 30, 50, 51, 20, 39]. In recent years, there has been an explosion in using ML-based techniques [71, 15] to uncover more subtle system heuristics for resource management [25, 5, 8, 12, 13, 21, 29, 34, 50, 51, 53, 57, 65, 77], hardware and system configuration [2, 7, 14, 21, 25, 43, 48, 68, 69, 72, 74, 76, 78], high-performance computing [44, 51, 34, 75, 2, 48, 60], and data-center-scale applications [12, 13, 74, 7, 69, 78, 67, 17, 16, 47, 70, 18].

Though ML is a natural solution for domains like image, video, and audio processing, the complexity of computer systems often requires extensive expertise to map systems problems to ML tasks. To our knowledge, there have been few publicly available systems datasets for ML [36, 37, 4, 11, 16]. *A goal of this work is to provide a representation of computer system behavior as a log-based data object amenable for use by different ML techniques.* We define a log as a multivariate time-series. Each log entry captures low-level, software agnostic metrics that reflect the behavior of the system at a moment of network interaction. Over the course of network-bound operation, a log becomes a rich data object that reflects local and global system behavior.

We present a parameterized and software-agnostic logging framework (figure 1*A*) that exposes how controlled changes can impact network-driven behavior. The framework can be parameterized by *1)* software for which execution is to be captured, *2)* metrics to be logged, and *3)* hardware settings. With this framework, we can identify a set of control parameters to study and exhaustively sweep through their values to generate comprehensive datasets of parameterized execution logs.

We used our framework to gather a dataset that captures the energy and performance behavior of different network servers. In the following sections, we describe our dataset and discuss the attributes of our logging framework. *Our goal is for this framework to be a template that enables gathering systems datasets, inviting a broader community of researchers to define and explore ML solutions for systems regardless of specialized domain knowledge.*

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

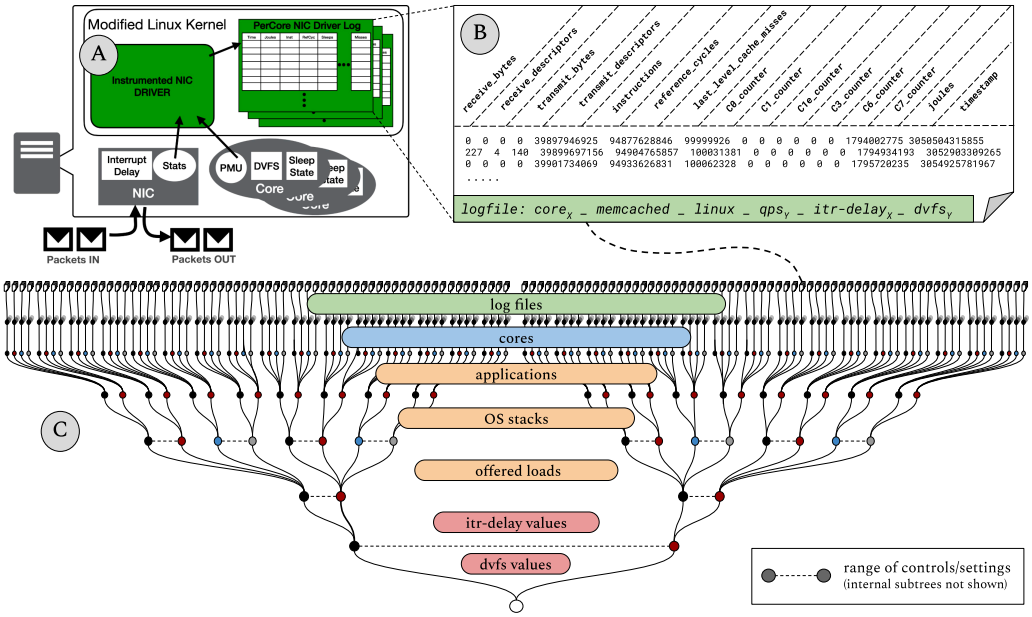


Figure 1: (A) Data logging framework. (B) Per-core log file. (C) An exhaustive parameter sweep.

2 Dataset

Our dataset *(A.1)* is motivated by interest in developing a network-centric and software-agnostic ML controller that can improve system energy-use while maintaining desired performance. For such a controller to be viable, it must *1)* be exposed to software-agnostic controls that have a net effect on energy and performance and *2)* understand system behavior across different modes of operation in order to affect desired change using available controls. Below, we describe our log-level representation of system behavior, two hardware-level controls that we chose to explore, and the different network and software configurations we used to capture a range of system behaviors. The dataset consists of per-core logs for every parameterized run of our exhaustive sweep of controls.

2.1 Log Data

Figure 1 illustrates an individual log file (1*B*) within a tree of experiments (1*C*) that produced our dataset. A timestamped log entry is generated every time the network interacts with the system and resides in a log file parameterized by the core for which logging was triggered. Every entry captures hardware metrics that are largely agnostic to the overarching software: number of instructions, cache misses, energy-use *(A.2.1)*, received and transmitted bytes, etc. *A full log is analogous to a video that describes overall system behavior, composed of frames that each individually describe execution at a single instant of the video.*

2.2 Hardware Controls

We explored two generic hardware controls whose values can be easily adjusted: processor dynamic-voltage-frequency-scaling (DVFS [19, 3]) and NIC interrupt-delay (ITR-delay) [33, 49]. DVFS is used to control core speed [27, 42, 61, 23, 46, 10, 35, 48, 45, 40, 26, 66, 41, 24, 22], and ITR-delay is used to control the rate at which a NIC fires interrupts signalling new work [73, 56, 54, 9, 22]. While there are many other controls to choose from, we use these two because of their simplicity and because preliminary logs revealed their clear impact on system behavior. We explored up to 340 unique combinations of ITR-delay and DVFS, producing a multi-terabyte dataset of log files parameterized by these two hardware controls and the different software and network configurations used. We will publish our dataset and logging framework at `https://github.com/SESA/intlog`.

OS	Application	Network Loads	Client-Server Loop	CPU/App	CPU/OS
Linux, EbbRT	NetPIPE	64B, 8KB, 64KB, 512KB	Closed	Low	High
	NodeJS	N/A	Closed	High	Low
	Memcached	200K, 400K, 600K QPS	Open	Low	High
	Memcached-Silo	50K, 100K, 200K QPS	Open	High	Low

Table 1: Operating system (OS), application, and network configurations. Network loads reflect mean values: our framework's benchmark tool generates distributions of offered loads (requests-per-second *(QPS)* or packet load sizes *(KB)*).

2.3 Software and Network Configurations

We ran our experiments with a matrix of configurations (*Table 1*) that populate our dataset with a versatile list of interactions between network demand and software processing efficiency:

Applications We divide the space of applications into the cross product of closed-versus-open-loop and high-versus-low CPU demand. Closed-loop performance, *throughput*, depends on the ability of server and client components to conduct synchronous packet exchange. Open-loop performance, *tail-latency*, depends on the ability of a server to satisfy a service-level-agreement (SLA, e.g. 500μs 99% tail-latency) for client-generated loads [63].

Operating Systems Our framework is designed to integrate naturally with different operating systems (OSes) that present varying local and global behavior worth capturing. Our integration with Linux and EbbRT [62] introduced 400 lines of new code. Including EbbRT as an alternative dataset parameter to Linux captures the performance and energy behaviors of a specialized library OS. To minimize general-purpose system noise in our Linux experiments, we built and used a set of application-specific high-performance Linux appliances [64, 28] *(A.3)*.

3 Attributes of a Data Logging Framework

Our framework adopts three main approaches:

1. **Software agnostic implementation**, to work effectively with no change to software.
2. **Fine-grain epoch-based logging**, to expose local events that explain global behavior.
3. **Noise-aware logging**, to produce logs that require minimal cleanup.

3.1 Software Agnostic Implementation

Our framework collects data transparently across different network applications and OSes [1]. We place its logic as far from the software stack and as close to the underlying hardware as possible. We implement logging in the NIC device driver - the first piece of logic to run after packets are received and the last to run before they are transmitted. This further enables seamless logging of network metrics to provide a more comprehensive context around network-induced system behavior *(A.2.3)*.

3.2 Fine-Grain Epoch-Based Logging

Per-Interrupt Logging An interrupt is the hardware event used by the NIC to signal to software that a network packet has been successfully sent or received. We exploit the lowest level of software that runs in response to NIC interrupts in order to log our metrics. This ensures that we capture log entries that reflect critical moments of system interaction with the network. While this approach can be supplemented with periodic sampling, we believe that it sufficiently captures highly salient events of network-driven execution (see figure 2).

[1] We integrated our platform with Apache Flink, an application streaming platform. We found our framework easy to integrate in this new domain without application or runtime changes.

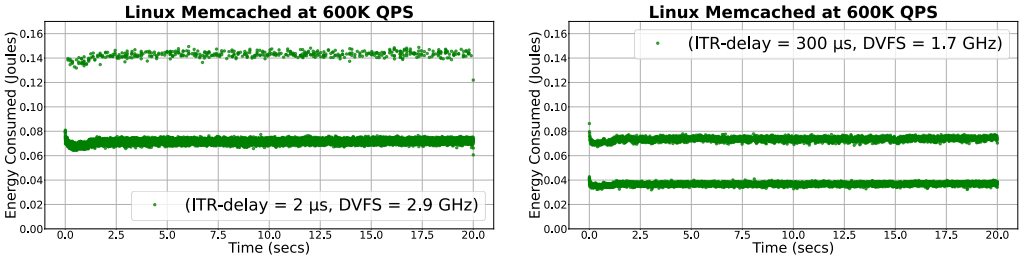


Figure 2: Per-interrupt measurements of a Memcached server processing 600K requests-per-second for 20 seconds. Interrupt-based logging defines epochs across timestamped log entries that reflect real values of elapsed time regardless of experimentally-induced frequency changes *(A.2.2)*. The logs that produced these plots are parameterized by two different ITR-delay and DVFS pairs that yield best *(right)* and worst *(left)* energy-use. All other log parameters are identical. The banding of energy-measurements reflects bimodal system behavior that was captured by our framework's interrupt-level logging. This banding reflects local system events which partition execution into idle (bottom band) and busy (top band) moments and explain net system energy-use.

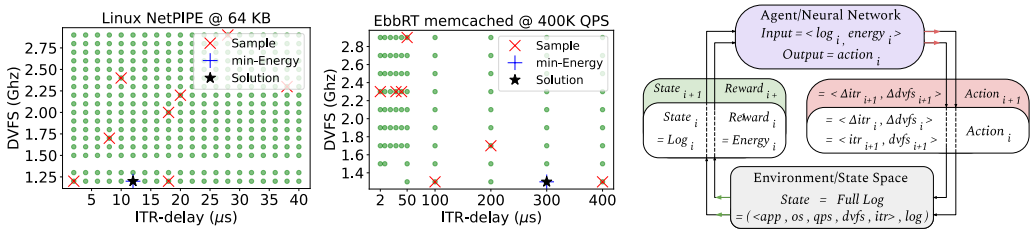
3.3 Noise-Aware Logging

We ensure that *logs reflect relevant system behavior by minimizing general system noise* in our experiments. For example, we maintain core affinity through application/core pinning and NIC-queue/core mapping to avoid run-time scheduling and memory allocation/de-allocation perturbations.

We ensure that *data logging has minimal impact on the system behavior that we want to capture*. For example, we use x86_64 non-temporal store instructions to read our log data. They are optimized for write-once-read-later data access, minimizing cache-coherency effects and the eviction of performance-critical application cache-lines.

4 Concluding Statements

Using our dataset, we posed a systems optimization problem as an ML task: when and to what values should we set our two hardware controls to yield optimal energy-use and performance. We demonstrated that a simple Bayesian optimization algorithm [1, 18, 52, 55] can effectively exploit structure in our dataset and converge to an optimal ITR-delay and DVFS setting for each of our permutations of software and network configurations (figures 3a and 3b). This result motivates us to pursue more complex ML tasks, such as developing a software and network agnostic reinforcement learning (RL) agent that can dynamically adjust ITR-delay and DVFS controls (figure 3c).



(a) (b) (c)

Figure 3: Current *(a, b)* and future *(c)* work on solving an energy and performance problems using ML.

We hope that our work can provide a motivational and extensible approach for capturing representative system-level datasets and using them to address issues of high priority, from datacenter energy efficiency to the general global minimization of computational resource footprint.

References

[1] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 469–482, Boston, MA, March 2017. USENIX Association.

[2] Jason Ansel, Maciej Pacula, Yee Lok Wong, Cy Chan, Marek Olszewski, Una-May O'Reilly, and Saman Amarasinghe. Siblingrivalry: Online autotuning through local competitions. In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, CASES '12, page 91–100, New York, NY, USA, 2012. Association for Computing Machinery.

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

[4] Atikoglu, Berk and Xu, Yuehai and Frachtenberg, Eitan and Jiang, Song and Paleczny, Mike. Workload Analysis of a Large-scale Key-value Store. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 53–64, New York, NY, USA, 2012. ACM.

[5] Ramazan Bitirgen, Engin Ipek, and Jose F. Martinez. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*, pages 318–329, 2008.

[6] Aaron Carroll and Gernot Heiser. Mobile multicores: Use them or waste them. *SIGOPS Oper. Syst. Rev.*, 48(1):44–48, may 2014.

[7] Chi-Ou Chen, Ye-Qi Zhuo, Chao-Chun Yeh, Che-Min Lin, and Shih-Wei Liao. Machine learning-based configuration parameter tuning on hadoop system. In *2015 IEEE International Congress on Big Data*, pages 386–392, 2015.

[8] Jian Chen and Lizy Kurian John. Predictive coordination of multiple on-chip resources for chip multiprocessors. In *Proceedings of the International Conference on Supercomputing*, ICS '11, page 192–201, New York, NY, USA, 2011. Association for Computing Machinery.

[9] C. Chou, L. N. Bhuyan, and D. Wong. Î¼dpm: Dynamic power management for the microsecond era. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 120–132, Los Alamitos, CA, USA, feb 2019. IEEE Computer Society.

[10] Ryan Cochran, Can Hankendi, Ayse K. Coskun, and Sherief Reda. Pack & Cap: Adaptive DVFS and Thread Packing under Power Caps. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, page 175–185, New York, NY, USA, 2011. Association for Computing Machinery.

[11] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 153–167, New York, NY, USA, 2017. Association for Computing Machinery.

[12] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, page 77–88, New York, NY, USA, 2013. Association for Computing Machinery.

[13] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-efficient and qos-aware cluster management. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, page 127–144, New York, NY, USA, 2014. Association for Computing Machinery.

[14] Zhaoxia Deng, Lunkai Zhang, Nikita Mishra, Henry Hoffmann, and Frederic T. Chong. Memory cocktail therapy: A general learning-based framework to optimize dynamic tradeoffs in nvms. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 232–244, New York, NY, USA, 2017. Association for Computing Machinery.

[15] Yi Ding, Nikita Mishra, and Henry Hoffmann. Generative and multi-phase learning for computer systems optimization. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, page 39–52, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Yi Ding, Ahsan Pervaiz, Michael Carbin, and Henry Hoffmann. Generalizable and interpretable learning for configuration extrapolation. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, page 728–740, New York, NY, USA, 2021. Association for Computing Machinery.

[17] Yi Ding, Avinash Rao, Hyebin Song, Rebecca Willett, and Henry Hoffmann. Nurd: Negative-unlabeled learning for online datacenter straggler prediction, 2022.

[18] Yi Ding, Alex Renda, Ahsan Pervaiz, Michael Carbin, and Henry Hoffmann. Cello: Efficient computer systems optimization with predictive early termination and censored regression, 2022.

[19] 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`.

[20] Han Dong, Sanjay Arora, Yara Awad, Tommy Unger, Orran Krieger, and Jonathan Appavoo. Slowing down for performance and energy: An os-centric study in network driven workloads. https://arxiv.org/abs/2112.07010, 2021.

[21] Christophe Dubach, Timothy M. Jones, Edwin V. Bonilla, and Michael F.P. O'Boyle. A predictive model for dynamic microarchitectural adaptivity control. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 485–496, 2010.

[22] Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy conservation policies for web servers. In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS'03, page 8, USA, 2003. USENIX Association.

[23] Krisztián Flautner, Steve Reinhardt, and Trevor Mudge. Automatic performance setting for dynamic voltage scaling. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, page 260–271, New York, NY, USA, 2001. Association for Computing Machinery.

[24] Vincent W. Freeh, Tyler K. Bletsch, and Freeman L. Rawson. Scaling and packing on a chip multiprocessor. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–8, 2007.

[25] Archana Ganapathi, Kaushik Datta, Armando Fox, and David Patterson. A case for machine learning to optimize multicore performance. In *Proceedings of the First USENIX Conference on Hot Topics in Parallelism*, HotPar'09, page 1, USA, 2009. USENIX Association.

[26] Rong Ge, Xizhou Feng, Wu-chun Feng, and Kirk W. Cameron. Cpu miser: A performance-directed, run-time system for power-aware clusters. In *2007 International Conference on Parallel Processing (ICPP 2007)*, pages 18–18, 2007.

[27] 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.

[28] Han Dong, Jonathan Appavoo. A Tutorial on Building Custom Linux Appliances. `https://www.usenix.org/publications/loginonline/building-linux-appliances`.

[29] Henry Hoffmann. Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 198–214, New York, NY, USA, 2015. Association for Computing Machinery.

[30] C. Imes, D. K. Kim, M. Maggio, and H. Hoffmann. Poet: a portable approach to minimizing energy under soft real-time constraints. In *2015 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 75–86, Los Alamitos, CA, USA, apr 2015. IEEE Computer Society.

[31] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume. `https://www.intel.com/content/dam/www/public/us/en/documents/manuals/`.

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

[33] 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`.

[34] Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana. Self-optimizing memory controllers: A reinforcement learning approach. *SIGARCH Comput. Archit. News*, 36(3):39–50, jun 2008.

[35] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, page 347–358, USA, 2006. IEEE Computer Society.

[36] Joseph L. Hellerstein. Google Cluster Data . `https://ai.googleblog.com/2010/01/google-cluster-data.html`.

[37] 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.

[38] David H. K. Kim, Connor Imes, and Henry Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *Proceedings of the 2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, CPSNA '15, page 78–85, USA, 2015. IEEE Computer Society.

[39] David H.K. Kim, Connor Imes, and Henry Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, pages 78–85, 2015.

[40] Wonyoung Kim, Meeta S. Gupta, Gu-Yeon Wei, and David Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134, 2008.

[41] Masaaki Kondo, Hiroshi Sasaki, and Hiroshi Nakamura. Improving fairness, throughput and energy-efficiency on a chip multiprocessor through dvfs. *SIGARCH Comput. Archit. News*, 35(1):31–38, March 2007.

[42] Etienne Le Sueur and Gernot Heiser. Slow down or sleep, that is the question. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, page 16, USA, 2011. USENIX Association.

[43] Benjamin C. Lee and David M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, page 185–194, New York, NY, USA, 2006. Association for Computing Machinery.

[44] Benjamin C. Lee, David M. Brooks, Bronis R. de Supinski, Martin Schulz, Karan Singh, and Sally A. McKee. Methods of inference and learning for performance modeling of parallel applications. In *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '07, page 249–258, New York, NY, USA, 2007. Association for Computing Machinery.

[45] Jungseob Lee and Nam Sung Kim. Optimizing throughput of power- and thermal-constrained multicore processors using dvfs and per-core power-gating. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, page 47–50, New York, NY, USA, 2009. Association for Computing Machinery.

[46] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server-level power control. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 4–4, 2007.

[47] Chi Li, Shu Wang, Henry Hoffmann, and Shan Lu. Statically inferring performance properties of software configurations. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery.

[48] J. Li and J.F. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, pages 77–87, 2006.

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

[50] Nikita Mishra, Connor Imes, John D. Lafferty, and Henry Hoffmann. Caloree: Learning control for predictable latency and low energy. *SIGPLAN Not.*, 53(2):184–198, mar 2018.

[51] Nikita Mishra, Huazhe Zhang, John D. Lafferty, and Henry Hoffmann. A probabilistic graphical model-based approach for minimizing energy under performance constraints. *SIGPLAN Not.*, 50(4):267–281, mar 2015.

[52] Luigi Nardi, Artur Souza, David Koeplinger, and Kunle Olukotun. Hypermapper: a practical design space exploration framework. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 425–426, 2019.

[53] Adam J. Oliner, Anand P. Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. Association for Computing Machinery.

[54] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, and Hari Balakrishnan. Shenango: Achieving high cpu efficiency for latency-sensitive datacenter workloads. In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, NSDI'19, page 361–377, USA, 2019. USENIX Association.

[55] Tirthak Patel and Devesh Tiwari. Clite: Efficient and qos-aware co-location of multiple latency-critical jobs for warehouse scale computers. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 193–206, 2020.

[56] Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. Arrakis: The operating system is the control plane. *ACM Trans. Comput. Syst.*, 33(4), November 2015.

[57] Paula Petrica, Adam M. Izraelevitz, David H. Albonesi, and Christine A. Shoemaker. Flicker: A dynamically adaptive architecture for power limited multicore systems. *SIGARCH Comput. Archit. News*, 41(3):13–23, jun 2013.

[58] 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.

[59] Xiang (Jenny) Ren, Kirk Rodrigues, Luyuan Chen, Camilo Vega, Michael Stumm, and Ding Yuan. An analysis of performance evolution of linux's core operations. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 554–569, New York, NY, USA, 2019. Association for Computing Machinery.

[60] Rohan Basu Roy, Tirthak Patel, Vijay Gadepally, and Devesh Tiwari. Bliss: Auto-tuning complex applications using a pool of diverse lightweight learning models. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, page 1280–1295, New York, NY, USA, 2021. Association for Computing Machinery.

[61] Hiroshi Sasaki, Satoshi Imamura, and Koji Inoue. Coordinated power-performance optimization in manycores. In *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, PACT '13, page 51–62. IEEE Press, 2013.

[62] Dan Schatzberg, James Cadden, Han Dong, Orran Krieger, and Jonathan Appavoo. Ebbrt: A framework for building per-application library operating systems. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 671–688, GA, 2016. USENIX Association.

[63] Bianca Schroeder, Adam Wierman, and Mor Harchol-Balter. Open versus closed: A cautionary tale. In *Proceedings of the 3rd Conference on Networked Systems Design and Implementation - Volume 3*, NSDI'06, page 18, USA, 2006. USENIX Association.

[64] Michael W. Shaffer. A linux appliance construction set. In *14th Systems Administration Conference (LISA 2000)*, New Orleans, LA, December 2000. USENIX Association.

[65] David C. Snowdon, Etienne Le Sueur, Stefan M. Petters, and Gernot Heiser. Koala: A platform for os-level power management. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys '09, page 289–302, New York, NY, USA, 2009. Association for Computing Machinery.

[66] V. Spiliopoulos, S. Kaxiras, and G. Keramidas. Green governors: A framework for continuously adaptive dvfs. In *Proceedings of the 2011 International Green Computing Conference and Workshops*, IGCC '11, page 1–8, USA, 2011. IEEE Computer Society.

[67] Gerald Tesauro. Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30, 2007.

[68] Erik Tomusk, Christophe Dubach, and Michael O'boyle. Four metrics to evaluate heterogeneous multicores. *ACM Trans. Archit. Code Optim.*, 12(4), nov 2015.

[69] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 1009–1024, New York, NY, USA, 2017. Association for Computing Machinery.

[70] Shu Wang, Chi Li, Henry Hoffmann, Shan Lu, William Sentosa, and Achmad Imam Kistijantoro. *Understanding and Auto-Adjusting Performance-Sensitive Configurations*, page 154–168. Association for Computing Machinery, New York, NY, USA, 2018.

[71] Nan Wu and Yuan Xie. A survey of machine learning for computer architecture and systems. *ACM Computing Surveys*, 55(3):1–39, apr 2023.

[72] Weidan Wu and Benjamin C. Lee. Inferred models for dynamic and sparse hardware-software spaces. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 413–424, 2012.

[73] Kenichi Yasukata, Michio Honda, Douglas Santry, and Lars Eggert. StackMap: Low-Latency networking with the OS stack and dedicated NICs. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 43–56, Denver, CO, June 2016. USENIX Association.

[74] Nezih Yigitbasi, Theodore L. Willke, Guangdeng Liao, and Dick Epema. Towards machine learning-based auto-tuning of mapreduce. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 11–20, 2013.

[75] Huazhe Zhang and Henry Hoffmann. Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, page 545–559, New York, NY, USA, 2016. Association for Computing Machinery.

[76] Yanqi Zhou, Henry Hoffmann, and David Wentzlaff. Cash: Supporting iaas customers with a sub-core configurable architecture. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 682–694, 2016.

[77] Yuhao Zhu and Vijay Janapa Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 13–24, 2013.

[78] Yuqing Zhu, Jianxun Liu, Mengying Guo, Yungang Bao, Wenlong Ma, Zhuoyue Liu, Kunpeng Song, and Yingchun Yang. Bestconfig: Tapping the performance potential of systems via automatic configuration tuning. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, page 338–350, New York, NY, USA, 2017. Association for Computing Machinery.

A Appendix

A.1 Dataset Details

1. Our dataset of logs resides in four folders labeled by our chosen network applications: `mcd`, `mcdsilo`, `node`, and `netpipe`. Each folder contains two sets of log files, one set of Linux logs and one set of EbbRT logs. Each log file contains raw timestamped log entries, and each entry lists: *log_identifier, receive_bytes, receive_descriptors, transmit_bytes, transmit_descriptors, instructions, reference_cycles, last-level_cache_misses, C1_counter, C3_counter, C6_counter, C7_counter, joules, rdtsc_timestamp*.
2. Each log file name is parameterized by the control values of the experiment that created it: `*.i_Core_ITR_DVFS_RAPL`, where `*` is the OS/application configuration, `i` is the experimental run number, `Core` is the processor core that the data is collected from, ITR is the interrupt-delay value used, `DVFS` is the frequency value that the processor core is set to, and `RAPL` is the RAPL power-limiting value that the package is set to.
3. Our dataset also contains non-log-based output files that were collected by our framework. One such file is created by the different client workload generators and shows the overall performance of the server. Others are per-experiment timestamp files that each contain two timestamps, the beginning and end of the experiment. We use these files to clean up logs and discard noisy entries generated during the initialization and clean-up of a server.

A.2 Framework Implementation Details

1. Energy is read from the MSR_PKG_ENERGY_STATUS register. While we validated results for per-application energy savings with respect to global rack energy numbers, we chose to use MSR_PKG_ENERGY_STATUS instead, because it allows reading at a finer granularity of one millisecond. In contrast, the per-second granularity of rack-level measurements could complicate the ability to attribute detailed energy-use to specific system events.
2. The `rdtsc` instruction [31] exists on every core and simplifies multi-core data logging by avoiding the use of global clock sources and synchronization. We invoke `rdtsc` to count the number of CPU ticks since the processor was reset. We then divide this value by the processor frequency imposed by the DVFS setting of our experiment to timestamp logs at a fixed-rate.
3. Typically, log data can be mapped to userspace memory. However, it is then the application's responsibility to handle storage, violating software agnostic properties. Instead, TCP protocols allow sending specific packet request data over the network to a storage server as an approach for log retrieval. In contrast to implementing custom network protocols for log data retrieval at the level of the OS, our implementation in the NIC device driver results in a more efficient process of transmission by avoiding the need to run the entire OS network stack. Another property of our NIC-level data logging is the use of fixed mapping between network queues and physical cores to create pre-allocated, per-core data structures sized appropriately for a target application and its offered load. This pre-allocation eliminates run-time OS overheads of dynamically allocating and freeing memory.

A.3 OS Stacks Studied

1. We integrated our logging framework into the NIC device drivers of two OSes, Linux and EbbRT [62]. Linux and EbbRT kernel images are setup similarly by configuring IA-32 architectural MSRs, processor specific MSRs (Tables 35-2 and 35-18 in [32]), and NIC features - disabling direct-cache injection, enabling receive-side scaling (RSS), and enabling hardware checksum offloading. We match the numbers of NIC transmit and receive descriptors and write-back thresholds for packet transmissions. We disable hyperthreads and TurboBoost on all processors. Prior work enables TurboBoost [9, 38, 27, 58], but energy-use anomalies have been reported upon use with different sleep states [42].
2. To reduce system noise in Linux logs, we built a set of application-specific Linux appliances that run a custom high-performance 5.5.17 kernel. We followed suggestions from prior work on reducing Linux core-operation costs [59] to create a modified high-performance

configuration of the 5.5.17 kernel. Linux appliances are constructed to run a RAM-based filesystem and contain only a small set of system libraries and kernel modules needed to run their constituent applications [28]. During Linux experiments, we pin applications to physical cores, avoiding system scheduling overheads and noise in the logs. We also disable Linux *irqbalance* in order to map packet receive interrupts to their respective cores. This ensures the affinity of log entries to the core for which logging has been triggered by a NIC interrupt.

3. We used EbbRT's library OS framework to construct custom binaries for the applications we consider. EbbRT consists of specialized, multi-core library OS components that are optimized to aggressively use per-core memory and fine-grain locking. These include the 82599 NIC driver, a custom TCP/IP stack, virtual and physical memory allocators that exploit large pages and pinned memory to avoid page-faults, and generic I/O buffers that enable zero-copy application data processing. It originally only ran within a virtualized environment through its custom `virtio` device driver. We ported it to run baremetal by developing a device driver for the Intel 82599 family of NICs. This driver interfaces with a multi-core TCP/IP stack and programs the NIC with per-core queues and interrupts, maintaining the affinity of TCP connections to their respective cores. Our logging framework was prototyped on this driver.