Early Work on Modeling Computational Sprinting

Nathaniel Morris and Christopher Stewart
The Ohio State University

Robert Birke and Lydia Chen
IBM Research Zürich

Jaimie Kelley
Denison University

CCS CONCEPTS

• **Computer systems organization** → **Performance of systems**; *Modeling techniques*;

KEYWORDS

Computational Sprinting, Resource Management, Prediction Accuracy, CPU Throttling, Simulation, Queuing Models

ACM Reference Format:
Nathaniel Morris and Christopher Stewart, Robert Birke and Lydia Chen, and Jaimie Kelley. 2017. Early Work on Modeling Computational Sprinting. In *Proceedings of SoCC '17, Santa Clara, CA, USA, September 24–27, 2017,* 1 pages.
https://doi.org/10.1145/3127479.3132691

Ever tightening power caps constrain the sustained processing speed of modern processors. With computational sprinting, processors reserve a small power budget that can be used to increase processing speed for short bursts. Computational sprinting speeds up query executions that would otherwise yield slow response time. Common mechanisms used for sprinting include DVFS, core scaling, CPU throttling and application-specific accelerators.

It is challenging to set good sprinting policies. Policies based on human intuition often consider a small portion of possible settings and perform poorly when workloads or hardware change. In early work, we have started to explore a model-driven approach that sets sprinting policies based on their expected response time. To be precise, we propose a class of performance models that accept the following types of input: arrival rate, sustained processing rate, sprint rate and sprinting budget. Our models characterize response time and sprinting frequency.

A key component of early success with modeling computational sprinting has be the use of random decision forests. We use this graphical learning method to map marginal sprint rate (i.e the rate for a fully sprinted execution), workload, and sprint policies to effective sprint rate (i.e the rate that amortizes the dynamic runtime factors)— changing a difficult non-separable problem into distinct solvable parts. Specifically, the effect sprint rate can be fed into classic queuing models and simulators to predict response time.

We have already evaluated the accuracy of our approach across five workloads on a dedicated machine using DVFS for sprinting. Figure 1 plots relative error for each workload. The curves for all of the workloads are close in shape. Jacobi, Stream, NN, Leukocyte, and BFS had median error below 5%. Across all workloads, 75^{th} percentile error was below 10%.

SoCC '17, September 24–27, 2017, Santa Clara, CA, USA

ACM ISBN 978-1-4503-5028-0/17/09.
https://doi.org/10.1145/3127479.3132691

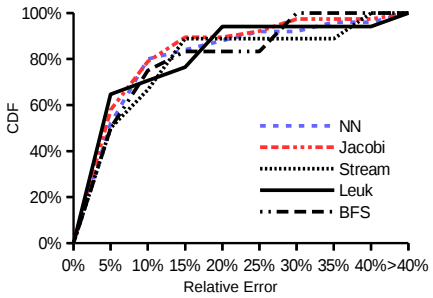

Fig. 1: CDFs of prediction error across workloads.

We observed similar results on other architectures. Query execution kernels, whether constrained by compute, memory or synchronization, did not affect our approach's accuracy. This is because our workload profiler accurately calibrates our approach for each kernel. Leukocyte has strong workload execution phases. Some phases are more amenable to computational sprinting than others. Nonetheless, our approach translates between marginal and effective sprint rate actively, using only workload-specific profiles.

Related Work: Queuing models can predict response time for server systems [6]. However, these models assume queuing delay and service time are independent. Interdependent queuing and service times lead to complicated and fragile models [2, 5]. When the focus is throughput, rather than response time, tree based offline profiling techniques work well and nearly optimally [3, 4, 8]. If profiles change during online execution, agent-based game theoretic approaches can provide optimal throughput-oriented sprinting policies [1, 7]. *Our work targets the very challenging problem of understanding the impact of sprinting on response time.*

Acknowledgements: This work is funded in part by NSF grants CAREER CNS-1350941 and CNS-1320071.

REFERENCES

[1] Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee. 2016. The Computational Sprinting Game. In *ASPLOS*. 561–575.
[2] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytiä. 2015. Reducing Latency via Redundant Requests: Exact Analysis. In *Sigmetrics*. 347–360.
[3] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Yuxiong He, and S. Elnikety. 2015. Measuring and Managing Answer Quality for Online Data-Intensive Services. In *IEEE ICAC*.
[4] Nikita Mishra, John Lafferty, and Henry Hoffmann. 2017. ESP: A Machine Learning Approach to Predicting Application Interference. In *International Conference on Autonomic Computing*.
[5] Nathaniel Morris, Siva Meenakshi Renganathan, Christopher Stewart, Robert Birke, and Lydia Chen. 2016. Sprint Ability: How Well Does Your Software Exploit Bursts in Processing Capacity?. In *International Conference on Autonomic Computing*.
[6] Christopher Stewart, Aniket Chakrabarti, and Rean Griffith. 2013. Zoolander: Efficiently Meeting Very Strict, Low-Latency SLOs. In *IEEE International Conference on Autonomic Computing*.
[7] Seyed Majid Zahedi, Songchun Fan, Matthew Faw, Elijah Cole, and Benjamin C Lee. 2017. Computational Sprinting: Architecture, Dynamics, and Strategies. *ACM Trans. on Computer Systems* 34, 4 (2017), 12.
[8] Huazhe Zhang and Henry Hoffmann. 2016. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. In *ASPLOS*. 545–559.