Early Work on Modeling Computational Sprinting

Nathaniel Morris (morris.743@osu.edu), Christopher Stewart (cstewart@cse.ohio-state.edu), Robert Birke, Lydia Chen, and Jaimie Kelley

The Ohio State University, IBM Research Zürich, Denison University

Objectives

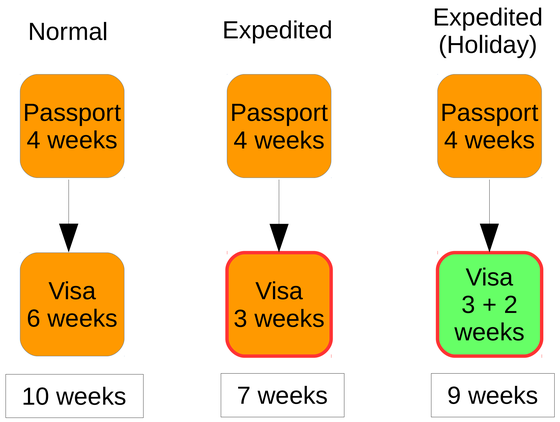
Model the time spent from start to finish for accelerated processes:

- Model the time spent for various types of acceleration.
- Model the time spent for more or less acceleration.
- Model the time spent for more or less frequent acceleration.

Find optimal policies.

Motivation

- A travel-visa is needed as soon as possible.
- The passport or travel-visa's processing time can be reduced by 50% but not both (highlighted in red).
- An additional 2 weeks are added if the travel-visa is processed during a holiday week (shaded green)?

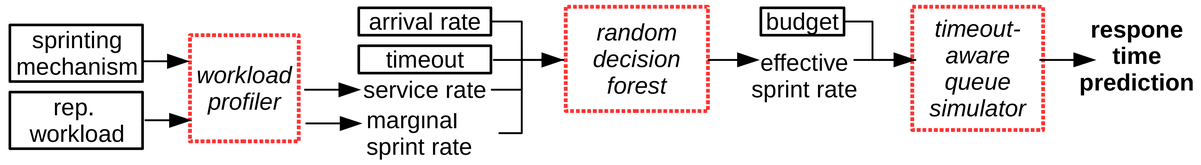


Introduction

- Power caps constrain the sustained processing speed of modern processors.
- With computational sprinting, processors reserve a small power budget to increase processing speed for short bursts.
- Sprinting mechanisms: DVFS, Core Scaling, CPU Throttling, and application-specific accelerators.
- It is challenging to set good sprinting policies.

Our Model-Driven Approach

This approach uses a set of rules to manage acceleration. These rules characterize the time spent from start to finish for accelerated processes. We use these rules to find configurations which lead to minimum finish time.



Design

Definitions:

Service rate: The inverse of mean processing time for queries that do not trigger sprinting.

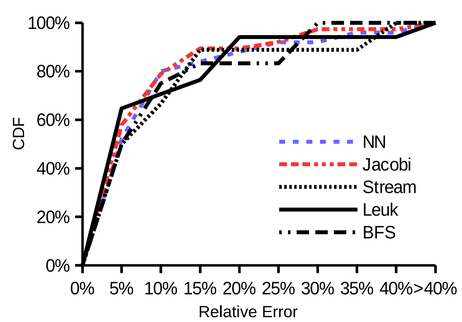
Marginal sprint rate: The inverse of mean processing time for queries that are sprinted for their whole execution.

Effective sprint rate: The sprint rate that would yield observed response time in our simulator.

Observed response time: Observed response time under the tested conditions and policies.

1. **Workload Profiler**
   - Profile the service rate and marginal sprint rate for a given workload.
2. **Random Decision Forest**
   - Map marginal sprint rate to the effective sprint rate.
3. **Timeout-aware simulator**
   - Predict response time based on system parameters and the effective sprint rate.

Result 1



Key Result

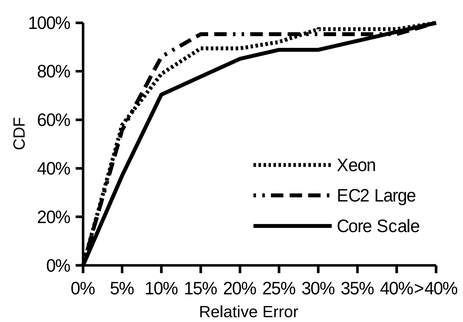
Across all workloads, 75% of the predictions contained less than 10% error.

Evaluation

For each workload and platform tested, we observe response time for all workload conditions and sprinting policies at cluster sampling centroids. This data provides the ground truth used to compute relative error.

- **Workloads:** Jacobi, NN, BFS, Stream, Leukocyte
- **Architectures:**
  1. **DVFS**—Sustained power cap:50-70 watts Burst power cap:90-110 watts
  2. **Core Scale**—Sustained speed:8 active cores Burst speed:16 active cores
  3. **EC2 DVFS**—Sustained speed:1.4 GHz Burst speed:2.0 GHz

Result 2

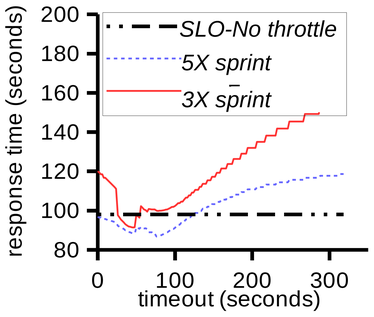


Key Result

Over 80% of the predictions on Xeon and EC2 contained less than 10% error.

Optimal Sprinting Policies

- Use our model to exhaustively search for a sprint policy that yields the lowest response time.
- **No throttle:** Our SLO is 1.15X response time observed with computational sprinting disabled on DVFS.
- **3X sprint:** Sprint rate was limited to 44.4 qph.
- **5X sprint:** Sprint rate was limited to 74 qph.



Conclusion

- Subtle changes in timeout settings (in either direction) can increase response time significantly.
- The key aspects to our model are
  1. profiling workload characteristics
  2. accounting for dynamic runtime factors via machine learning,
  3. capturing interdependence between service and waiting times via first-principles queue simulation.
- We validated our model across multiple sprinting platforms and diverse workload kernels and conditions.

Future Work

- Extend model to generalized tasks.
- Implement model in compute platform (Spark).

Acknowledgements

I would like to thank my advisor and the co-authors for providing constructive feedback. Also, I extend my gratitude to NSF for funding this project.