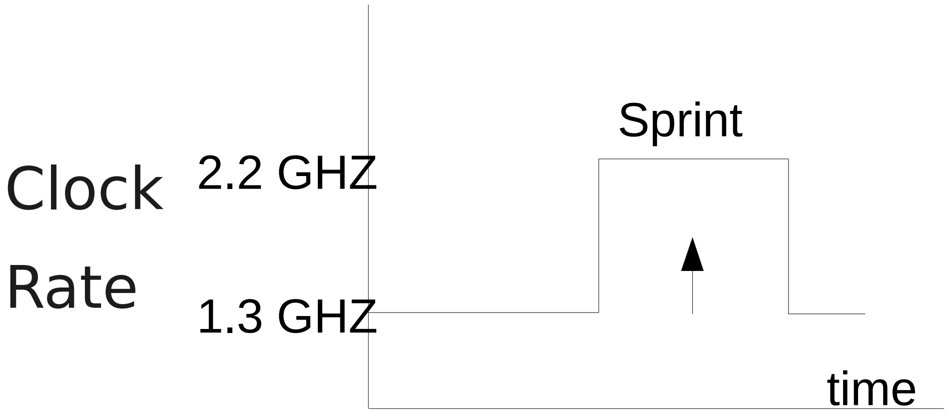
Model-Driven Computational Sprinting

Nathaniel Morris, Christopher Stewart, Lydia Chen,
Robert Birke, Jaimie Kelley

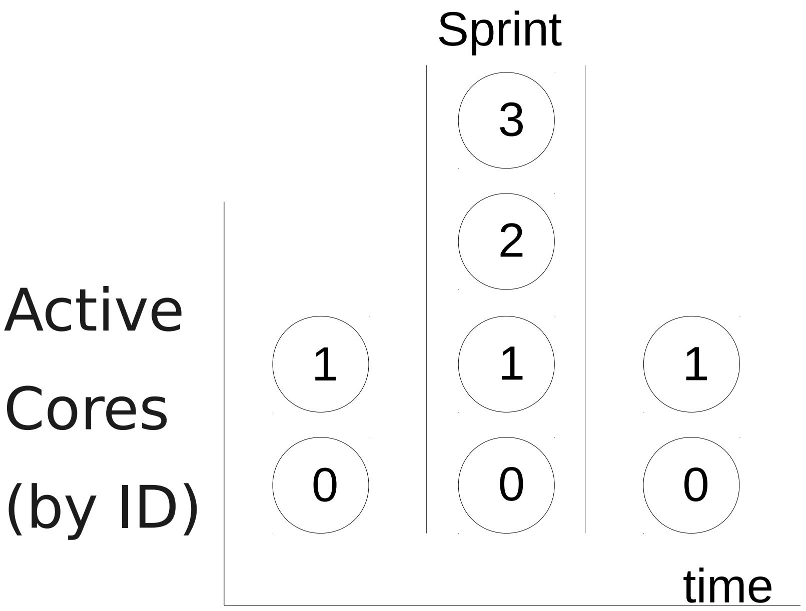
Computational Sprinting

[Raghavan, 2012]: Processor improves application responsiveness by temporarily exceeding its sustainable thermal budget

(1) DVFS



(2) Core Scaling



Computational Sprinting cont.

Sprinting budget constrains total time in sprint mode

- For example, 6 minutes per 1 hour (AWS Burstable)

Budget defined by scarce resources

- Thermal capacitance (Raghavan, 2012)
- Energy (Zheng,2015;Fan,2016)
- Reserve CPU cycles in Co-located Contexts (AWS)

Sprinting policy = mechanism + budget + trigger

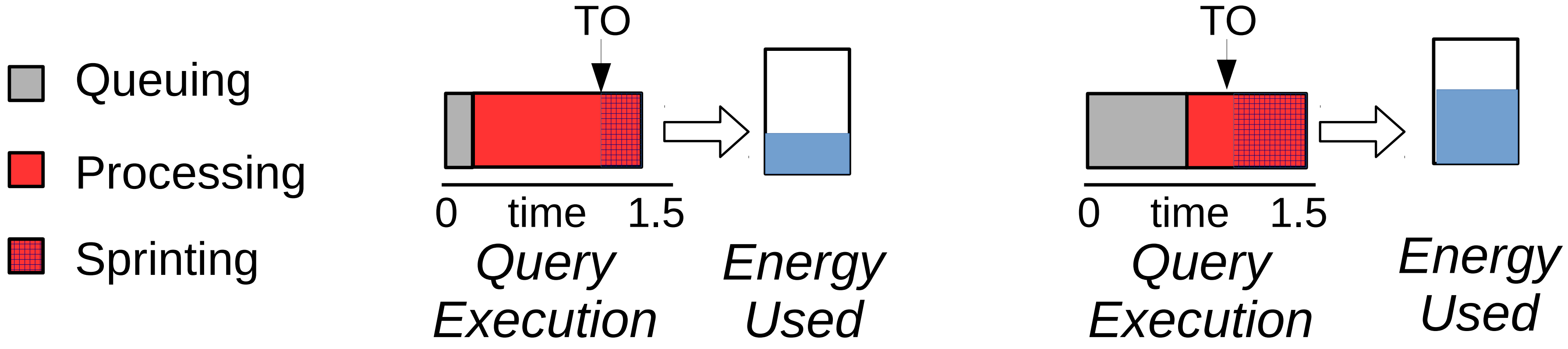
- SLO-driven services use timeouts to trigger sprinting
- [Haque, 2012; Hsu, 2015]

Sprinting Example

Example: SLO → Complete 99% of queries in 2 seconds

Example Policy: Execute at 1.3 GHZ. Time out after 1.5 seconds, set DVFS to 2.2 GHZ until (1) query completes or (2) 50 J budget is exhausted

- Root causes: (1) Slow execution (2) Long queuing delay



Sprinting Policies Are Hard to Set

With sprinting, dynamic runtime factors determine query execution time

- e.g., queue length, speedup from sprinting, remaining budget

How to set timeout policies and budgets?

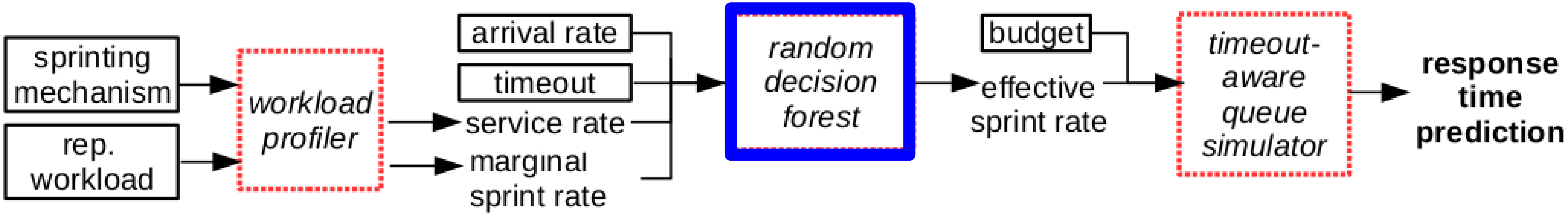
- State of practice: Same sprinting policy for all workloads [AWS Burstable]
- State of art: Target slower than expected query executions [Hsu, 2016], Target high utilization [Haque, 2015]
- These approaches are heuristic driven; Could perform poorly & sensitive to parameter settings

Model-Driven Computational Sprinting

Model-Driven Computational Sprinting predicts expected response time and uses the predictions to compare policies and discover high performance settings

Our approach combines:

- First-principles modeling to capture sprinting fundamentals
- Machine learning to accurately characterize the effects of runtime factors on response time



Outline

- Introduction
- First Principles for Sprinting
- Effective Sprint Rate
- Model Evaluation & Model-Driven Management

Principles of Sprinting

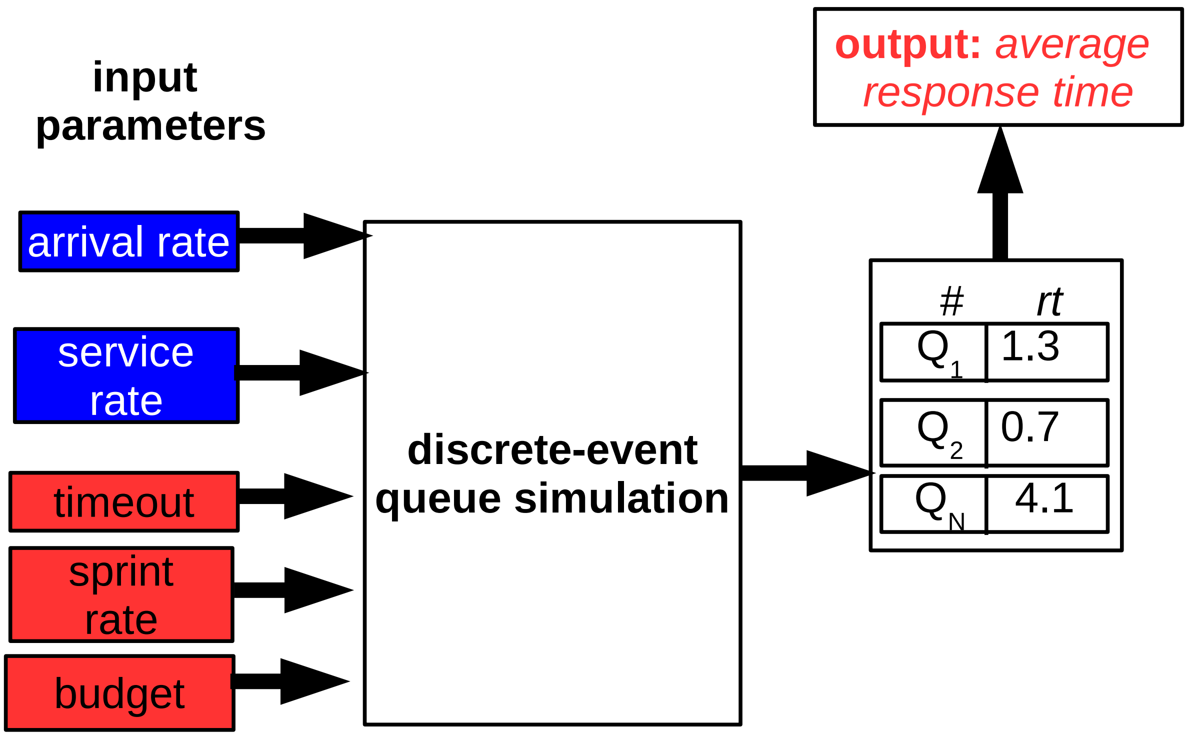
Discrete-event queuing simulator for sprinting

Traditional queuing

- Arrival & service rate

Sprinting accepts additional parameters

- Sprint rate & Timeout
- Budget



Principle: Compute resp. time for each job given queuing delay, processing time and timeout

Offline Workload Profiling

Profiling varies workload conditions and sprinting policies

The service rate (sustained processing time) and marginal sprint rate are calculated via profiling

Marginal sprint rate:

- Processing time when a entire query execution is sprinted offline

Outline

- Introduction
- First Principles for Sprinting
- Effective Sprint Rate
- Model Evaluation & Model-Driven Management

Runtime Factors Affect Sprinting

Offline profiling explains sprinting in isolation

System properties known only under live workload, i.e., at runtime, affect response time significantly

Why offline profiling is inaccurate?

Concurrency Paradox: A sprint that alters 1 query execution can affect response time for many queries

- The sprint reduces queuing backlog

Phase Paradox: For 1 query execution, sprinting can consistently yield less speedup under live workload

- Timeout triggers too late, missing execution phases amenable to sprinting mechanism (e.g., seq phase under core scaling)

From Marginal to Effective Sprint Rate

Naive insight: Learn F(wrkld, sprint policy) → resp. time

- Complicated function, lots of training

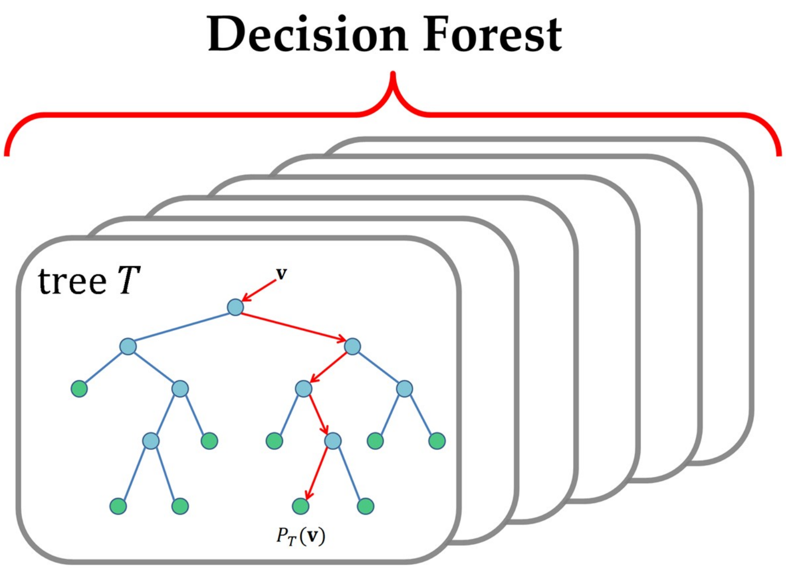
Our insight: Learn F(wrkld, sprint policy) → eff. sprint rate

- Then use first principles to get response time

Which machine learning approach?

Random Decision Forest combines multiple, deep decision trees

- **Deep → low bias**
- **Multiple → reduce variance**



The Ohio State University
ReRout Lab
Computer Systems Research That Reaches Out

Outline

- Introduction
- First Principles for Sprinting
- Effective Sprint Rate
- Model Evaluation & Model-Driven Management

Evaluation Setup

- Set up 7 services (2 Spark + 5 NAS) and tested multiple sprint policies
- Tested DVFS, Core-Scale, ec2-DVFS
- **Methodology:** Given arrival rate and sprinting policy, predict response time. Error is percent difference between prediction and observed response time

Goals:

1. Compare how well our modeling approach generalizes

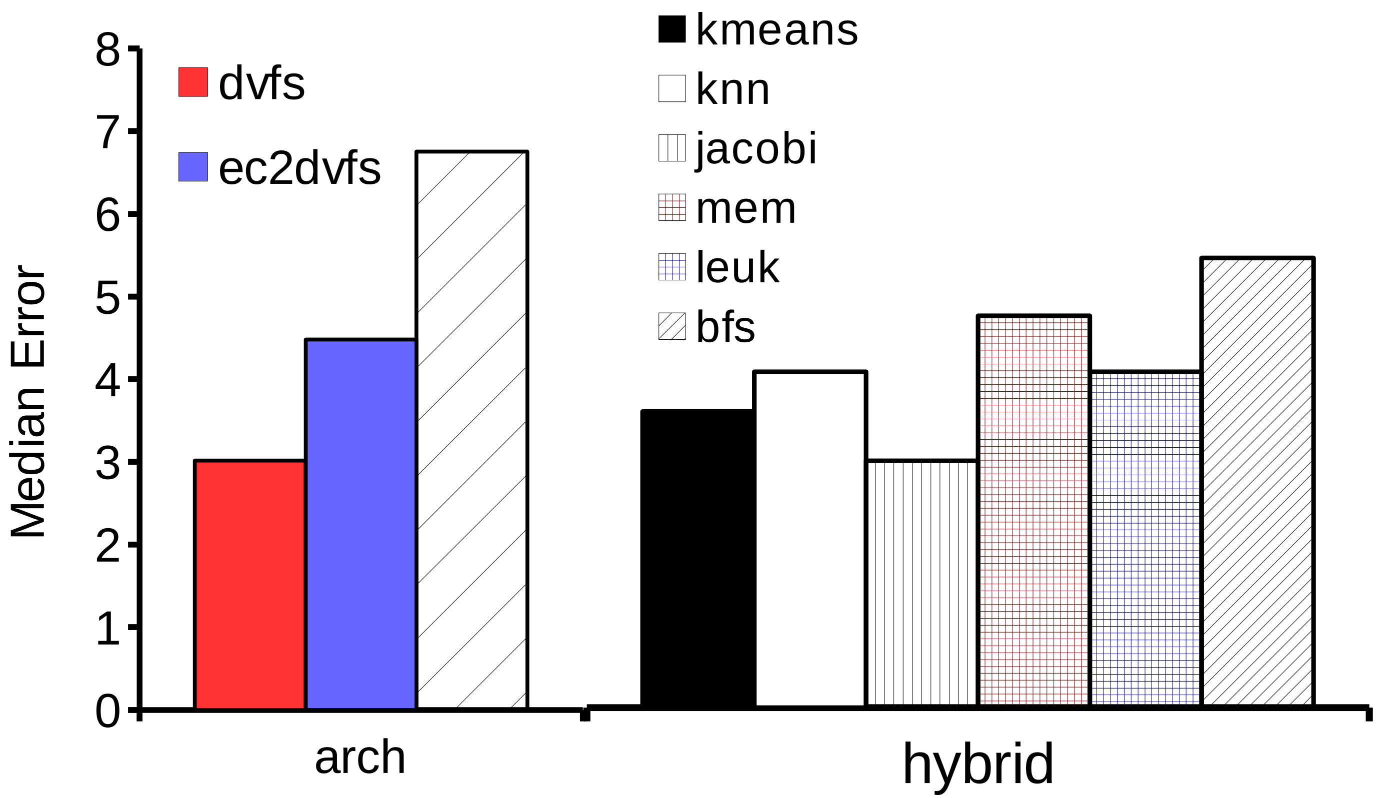
Do sprinting mechanisms affect accuracy? Workloads?

2. Contrast with alternative modeling approaches?

Accuracy? Cost to set up?

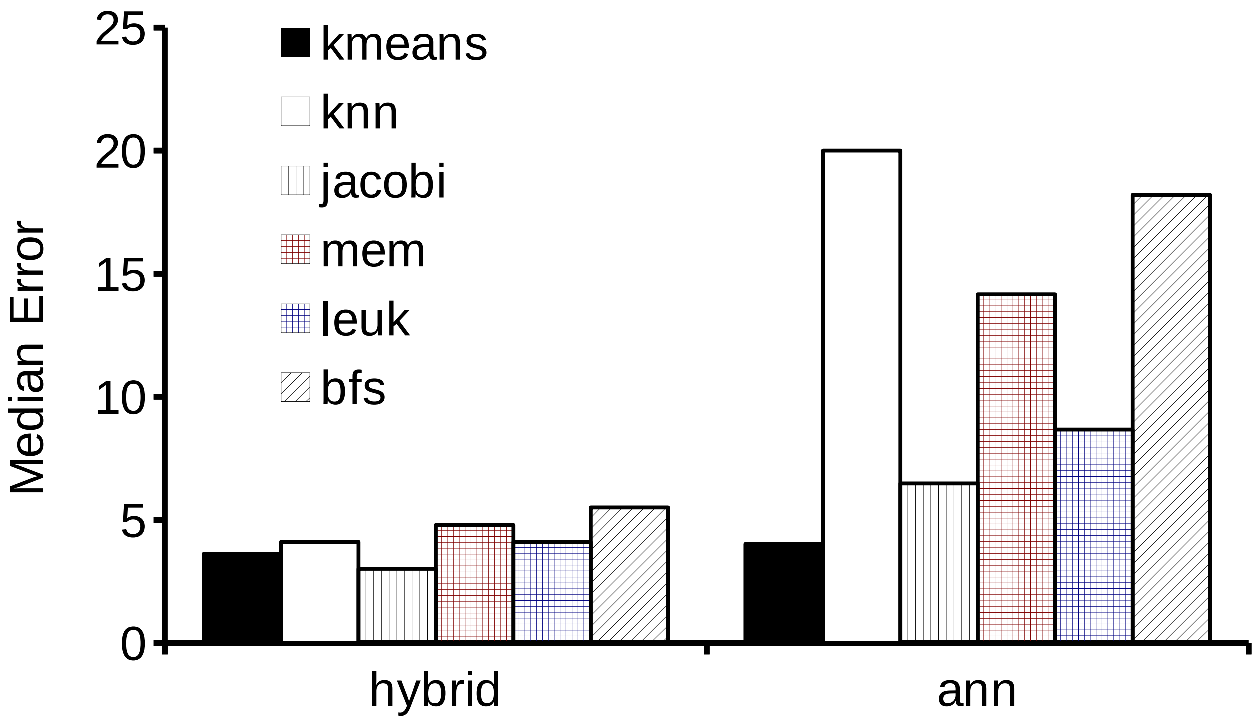
3. Does a model-driven approach help discover better sprinting policies?

Accuracy Across Mechanisms/Workloads



- **Our approach is 93-97% accurate across sprinting mechanisms and a wide variety of workloads.**

Hybrid Model vs ANN



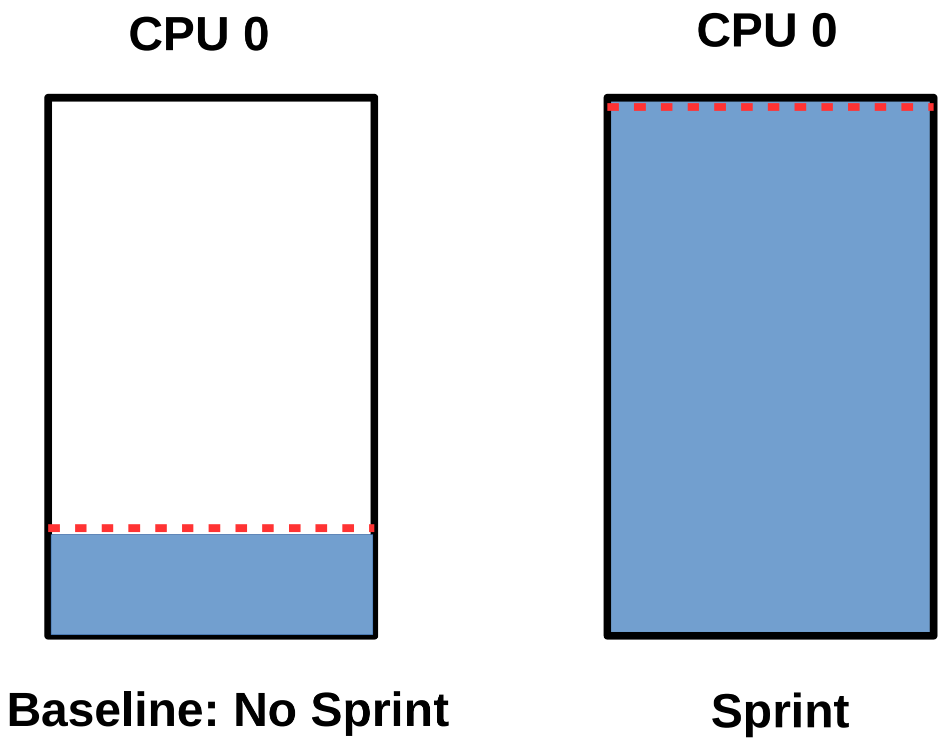
- **What if we just used machine learning? ANN – 5-layer Artificial Neural Network trained iteratively and tuned**
- **Our approach required 6x to 54x less data than ANN with comparable accuracy**

Model-Driven Management

CASE STUDY

Computational Sprinting & AWS Burstable Instances

- Service can access only a fraction of CPU resources during normal operation
- Service *sprints* (exclusive use of CPU) for 6 min/hour



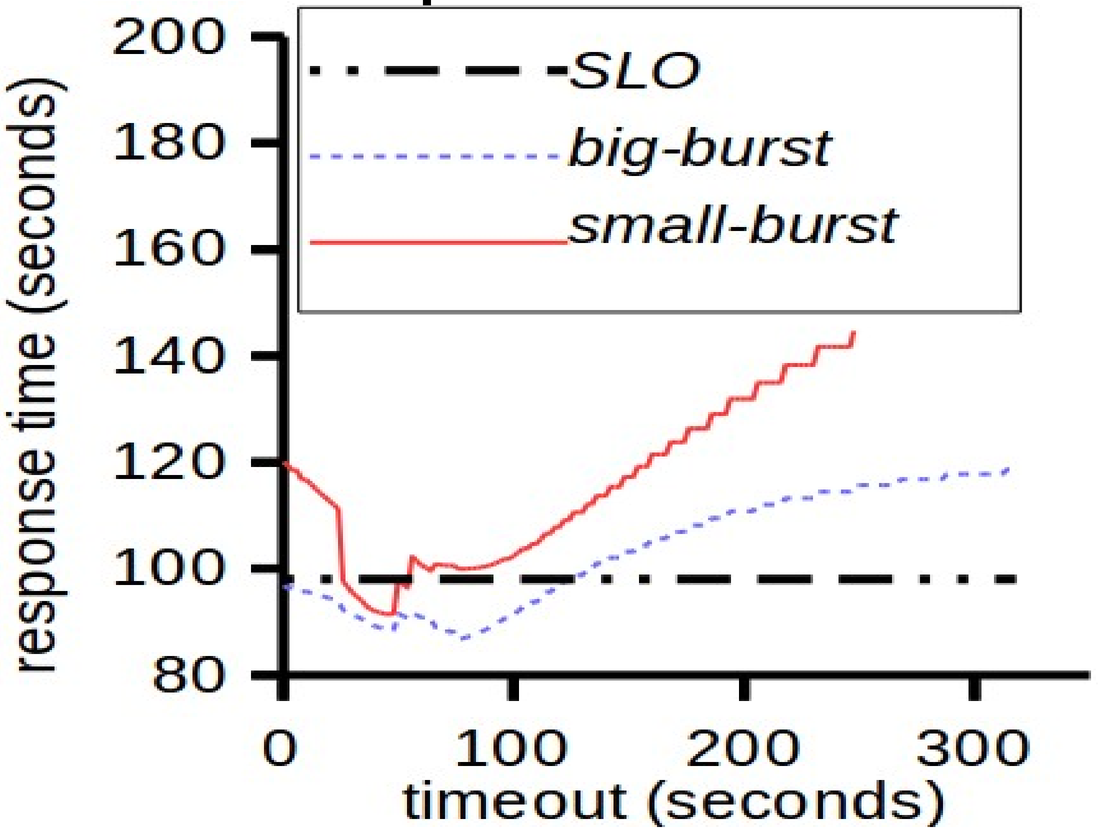
- **Implementations**
  - Big burst: 20% norm → 100% sprint
  - Small burst: 20% norm → 60% sprint

Model-Driven Management Cont.

Search for best sprinting policy

- Scan timeouts until the policy with lowest response time is found
- Try for a large and small budget
- The best timeout is different depending on budget and workload
- **Best policy improved response time by up to 1.4X**

Example with Jacobi Service



Model-Driven Management Cont.

Use hybrid model to search for best sprinting policy

Adrenaline: Sets timeout to the 85 th % percentile of non-sprinting response time [Hsu, HPCA, 2015]

Few-to-Many: Finds the largest timeout setting that exhausts budget (speeding up the slowest queries) [Haque, ASPLOS,2015]

	Response Time Improvement		
	Our Approach	Adrenaline	Few-to-Many
Big Burst	1	1.26	1.06
Small Burst	1	1.45	1.36

Conclusion

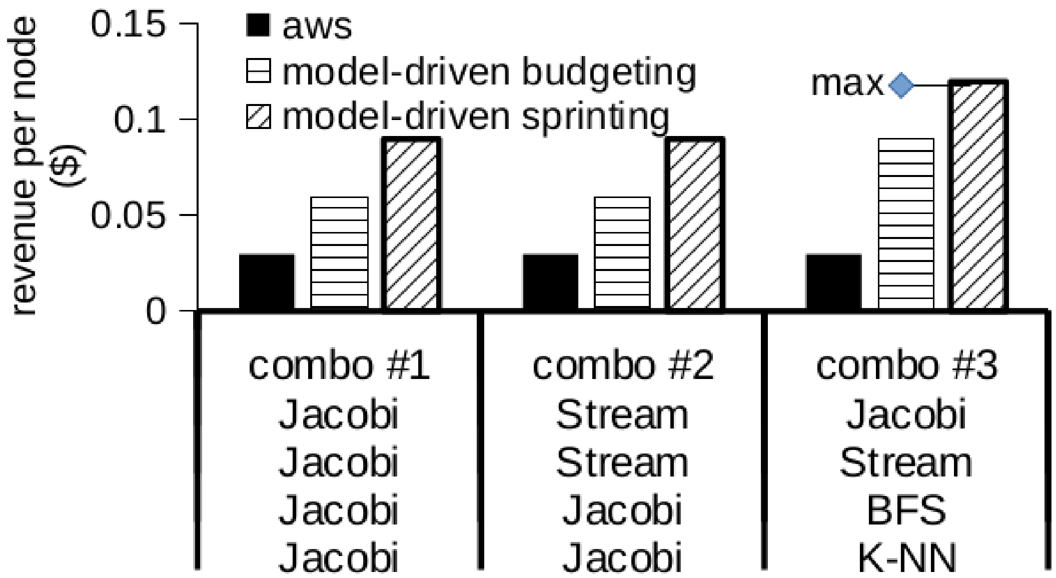
- Sprinting reduces SLO violations, but sprinting policies have complex effects on runtime execution and response time
- We combine machine learning and first principles to model response time quickly and accurately
- Our modeling approach introduces effective sprint rate, i.e., speedup given dynamic runtime conditions
- With our model, we discovered policies that outperformed state-of-the-art heuristics by 1.45X

Benefits of Good Sprinting Policies

Better sprinting policy allows for more colocated workloads

More workloads per node increases profit

- Profit increased by 1.6X



Budgeting shrinks budget but increases sprint rate

Our approach fixes the budget and selects a timeout

- Sprinting policies more efficient for all 3 combos