

The Computational Sprinting Game

Songchun Fan*, Seyed Majid Zahedi*, Benjamin C. Lee

{songchun.fan, seyedmajid.zahedi, benjamin.c.lee}@duke.edu

[* Co-First Authors]



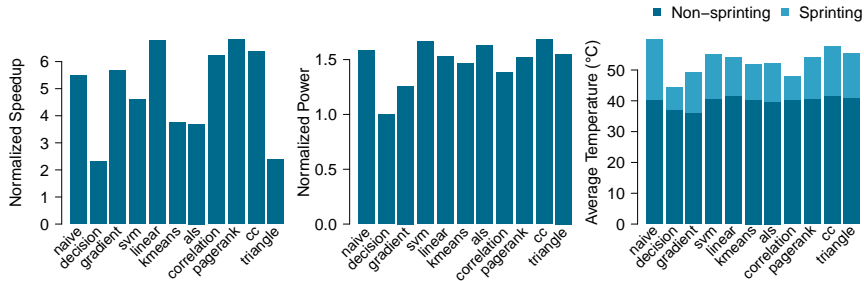
Computational Sprinting

- Supply extra power to enhance performance for short durations
- Activate more cores, boost voltage/frequency



Computational Sprinting

- Supply extra power to enhance performance for short durations
- Activate more cores, boost voltage/frequency



Sprinting Architecture

- Power for sprints supplied by shared rack
- Heat from sprints absorbed by thermal packages

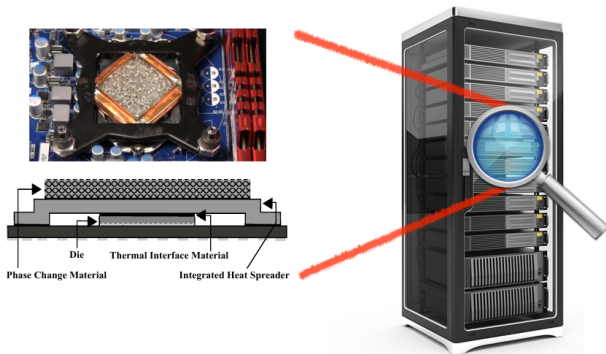
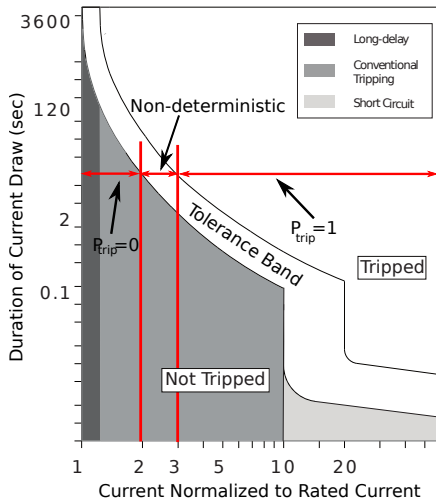


Fig. www.fortlax.se and Raghavan, Arun, et al. "Computational sprinting on a hardware/software testbed."

Power Emergencies



- Sprints may trip breaker
- Current \uparrow with sprinters
- Time \uparrow with sprint duration
- Risk \uparrow with current, time

Fig. Fu, Wang, and Lefurgy. "How much power oversubscription is safe and allowed in data centers."



Uninterruptible Power Supplies



- When sprints trip breaker, draw on batteries
- When sprints complete, recharge batteries

Example – Private Clouds

Google Apps



- Applications compute on servers that share power
- Processors sprint independently
- Processors sprint selfishly for performance

Sprinting Management

When should processors sprint?

- Phases with higher performance from sprints
- But sprints prohibited as chip cools

Which processors should sprint?

- Processors that benefit most from sprints
- But sprints prohibited as batteries recover



Management Desiderata

Individual Performance

- Sprints account for phase behavior
- Sprints now constrain future sprints

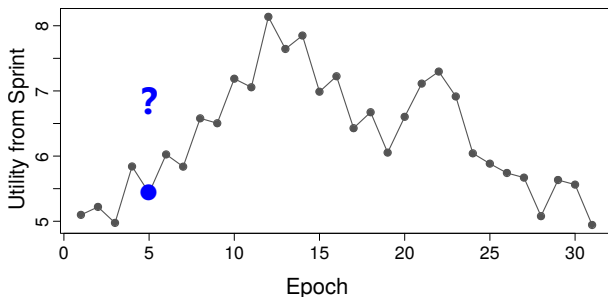
System Stability

- Sprints account for others' sprinting strategies
- Sprints risk power emergencies



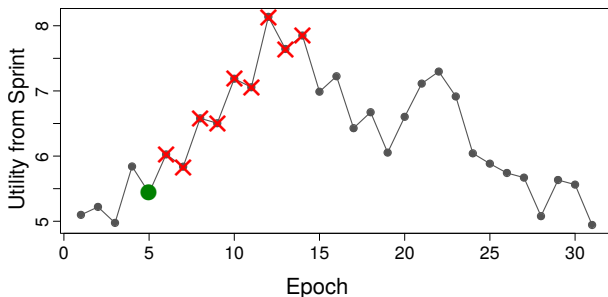
Sprinting Strategy

- Optimize sprints given constraints
- Sprint, wait $\Delta_{cooling}$ for chip cooling
- Sprint, wait $\Delta_{recovery}$ for rack recovery if breaker trips



Sprinting Strategy

- Optimize sprints given constraints
- Sprint, wait $\Delta_{cooling}$ for chip cooling
- Sprint, wait $\Delta_{recovery}$ for rack recovery if breaker trips



Game Theory

Study strategic agents

- Agents selfishly maximize individual utility

Optimize responses

- Response maximizes utility, given others' strategies

Find equilibrium

- State where all agents play their best responses



Sprinting Game

States

- Active – can sprint
- Cooling – cannot sprint, chip cooling
- Recovery – cannot sprint, batteries recharging

Actions

- Sprint or not, when active

Strategies

- Agent's state, app's phase, history, ...
- Others' strategies, utilities, and states, ...



Mean Field Equilibrium (MFE)

Challenges

- Large system with many agents
- Complex strategies and many competitors
- Intractable optimization for best response

Solution

- Abstract many agents with statistical distributions
- Optimize agents' strategies against expectations



Equilibrium Strategy

Agents maximize expected value of (not) sprinting

- Current state
- Utility from sprinting, u
- Probability of tripping, P_{trip}

Agents employ threshold strategy

- If active and $u \geq u_T$, then sprint



Find Equilibrium – Offline

- Initialize probability of breaker trip P_{trip}
- Given P_{trip} , optimize threshold strategy u_T
- Given u_T , estimate number of sprinters N
- Given N , update probability P'_{trip}
- Iterate if $P'_{trip} \neq P_{trip}$

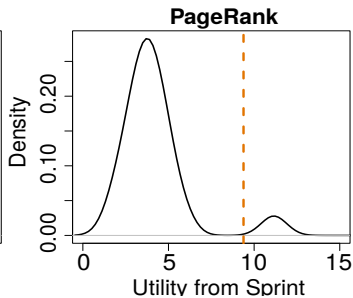
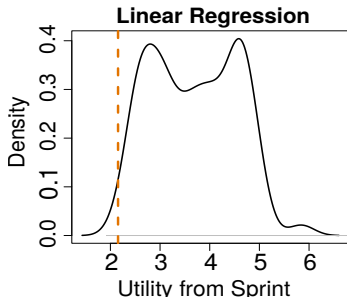


Execute Strategy – Online

If active and $u \geq u_T$, then sprint



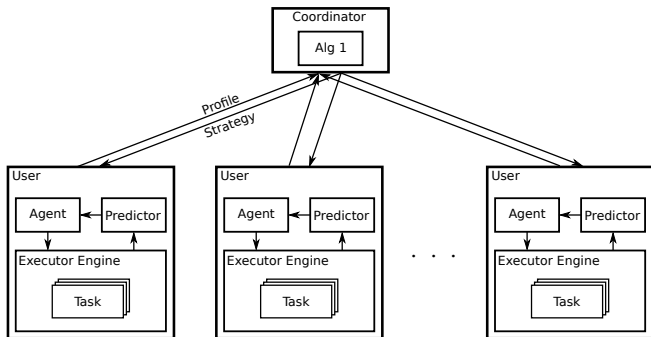
Sprinting Thresholds



- Thresholds are optimal and diverse
- Agents behave strategically to maximize performance



Management Architecture



- **Offline:** coordinator profiles utility, optimizes thresholds
- **Online:** predictors estimate sprint utility
- **Online:** agents apply threshold strategy
- **Online:** executor adapts computation



Experimental Methodology

Sprinting

- 3 cores @1.2GHz → 12 cores @ 2.7GHz

Workloads

- Apache Spark
- Spark engine dynamically schedules tasks on active cores

Performance Metric

- Tasks completed per second (TPS)

Simulation Method

- R-based simulator using traces of Spark computation



Management Policies

Greedy

- Sprint if neither cooling nor recovering

Exponential Back-off

- Sprint if neither cooling nor recovering
- Wait randomly for $U[0, 2^k]$ epochs after k^{th} trip

Cooperative Threshold

- Enforce globally optimized threshold

Equilibrium Threshold

- Announce decentralized, strategic threshold



Case for Equilibria

	Equilibrium	Cooperative
Performance		+
Stability	+	

- Cooperative (+): maximize global performance
- Equilibrium (+): remove incentives to deviate







Case for Equilibria

	Equilibrium	Cooperative
Performance		+
Stability	+	-

- Cooperative (+): maximize global performance
- Equilibrium (+): remove incentives to deviate
- Cooperative (-): enforce strategies globally



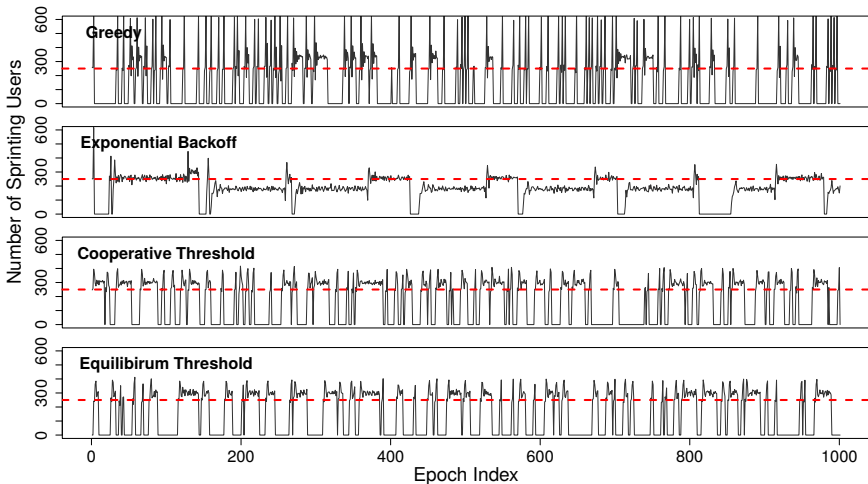
Case for Equilibria

	Equilibrium	Cooperative
Performance		
Stability		

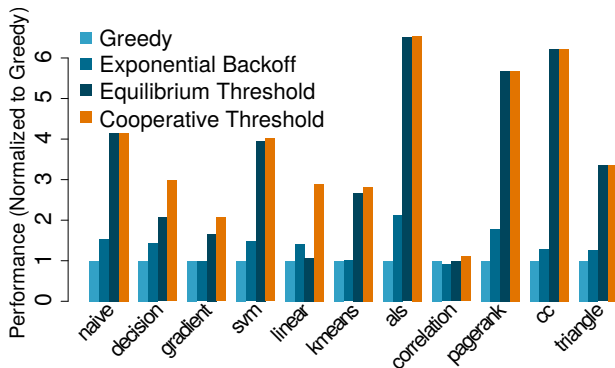
- Cooperative (+): maximize global performance
- Equilibrium (+): remove incentives to deviate
- Cooperative (-): enforce strategies globally
- Equilibrium (+): maximize individual performance



Sprinting Behavior



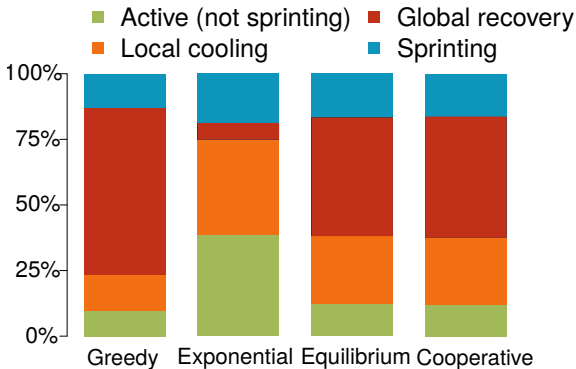
Sprinting Performance



- Greedy – aggressive, incurs emergencies
- Exponential – conservative, untimely sprints
- Equilibrium – strategic, produces equilibrium
- Cooperative – optimal, requires enforcement



Game States



- Greedy – time in recovery
- Exponential – untimely sprints
- Equilibrium – timely sprints
- Cooperative – timely sprints



Conclusion

Management with game theory

- Agents sprint according to threshold – inexpensive
- Agents have no incentives to deviate – stable
- Agents optimize response – high performance

Future directions

- Use game theory to manage scarce resources
- E.g., big/small processors, accelerators



Thank you

Questions?

