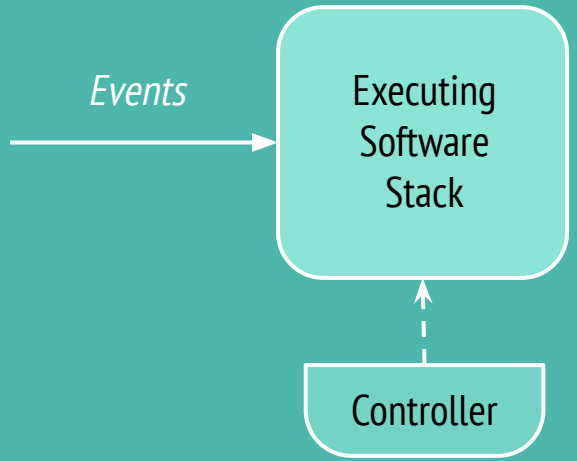
Rethinking Energy-Aware Performance Control for Event-Driven Systems

Yara Awad, Han Dong, Sanskriti Sharma, Jonathan Appavoo, Sanjay Arora

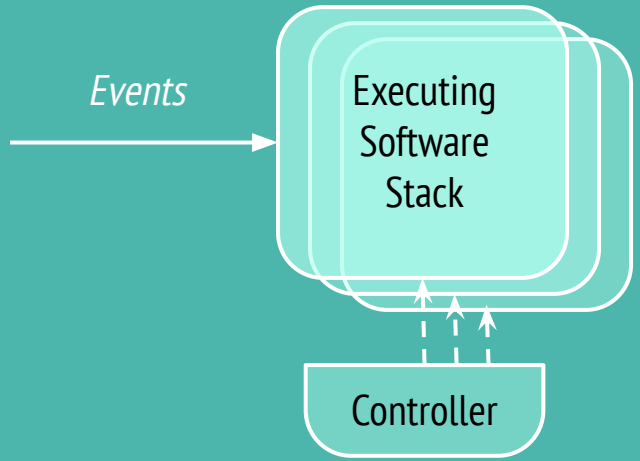
The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous ... ...



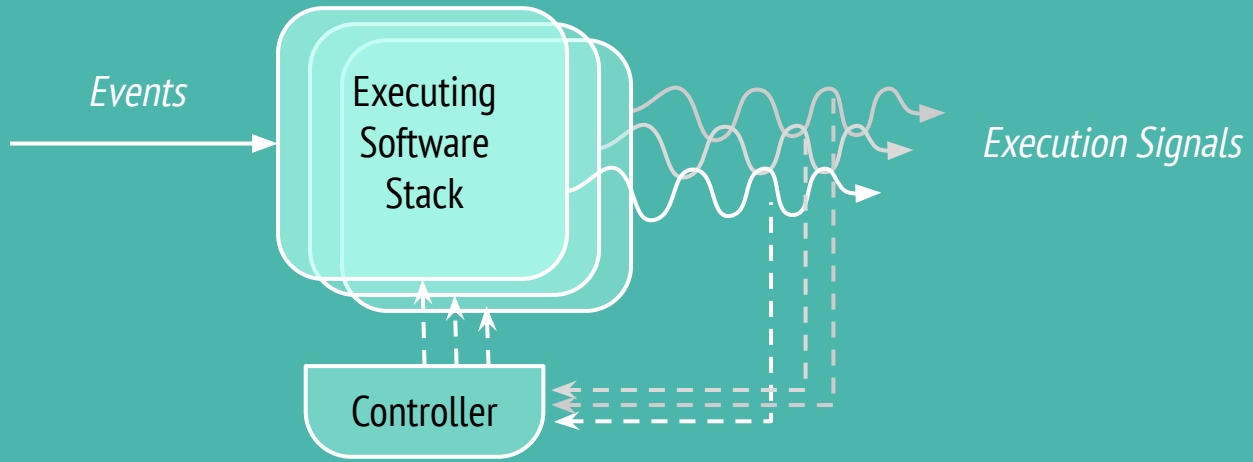
The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous ***Software Agnostic*** ...



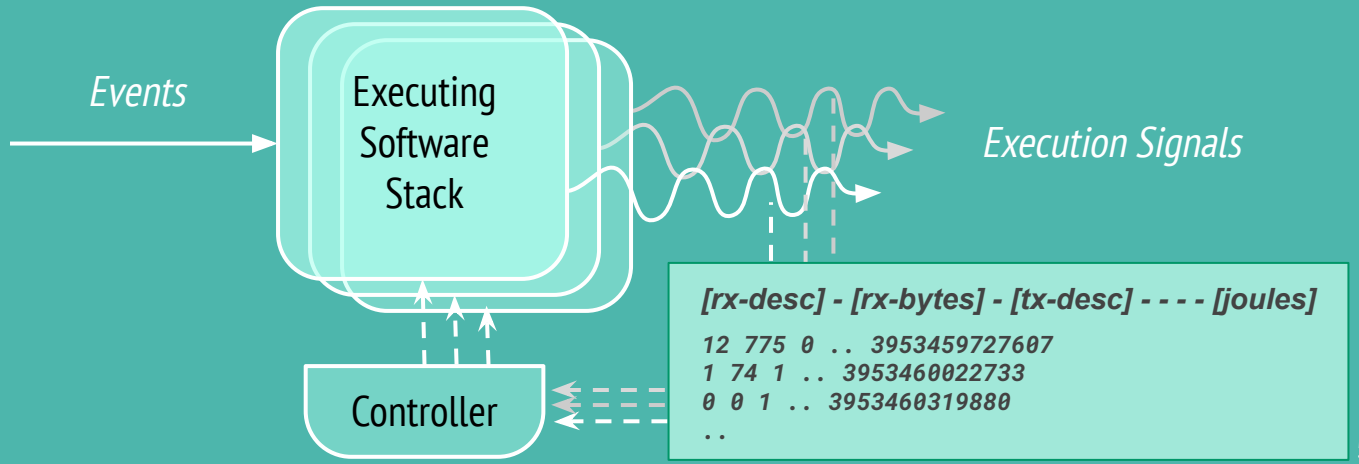
The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous *Software Agnostic* ***Feedback-Based***



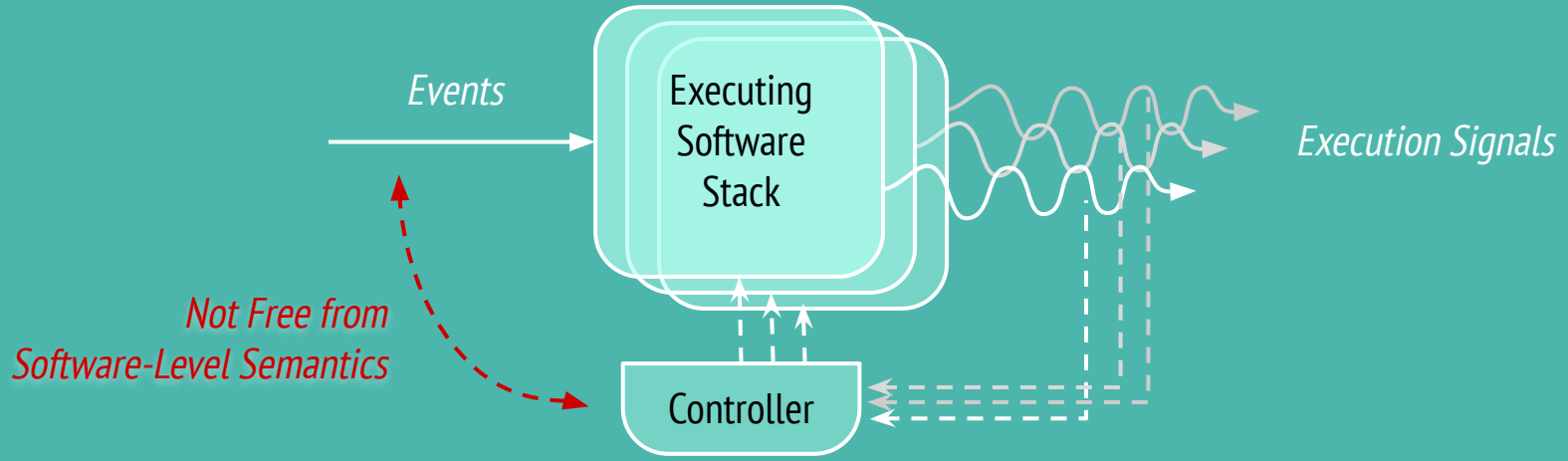
The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous *Software Agnostic* ***Feedback-Based***



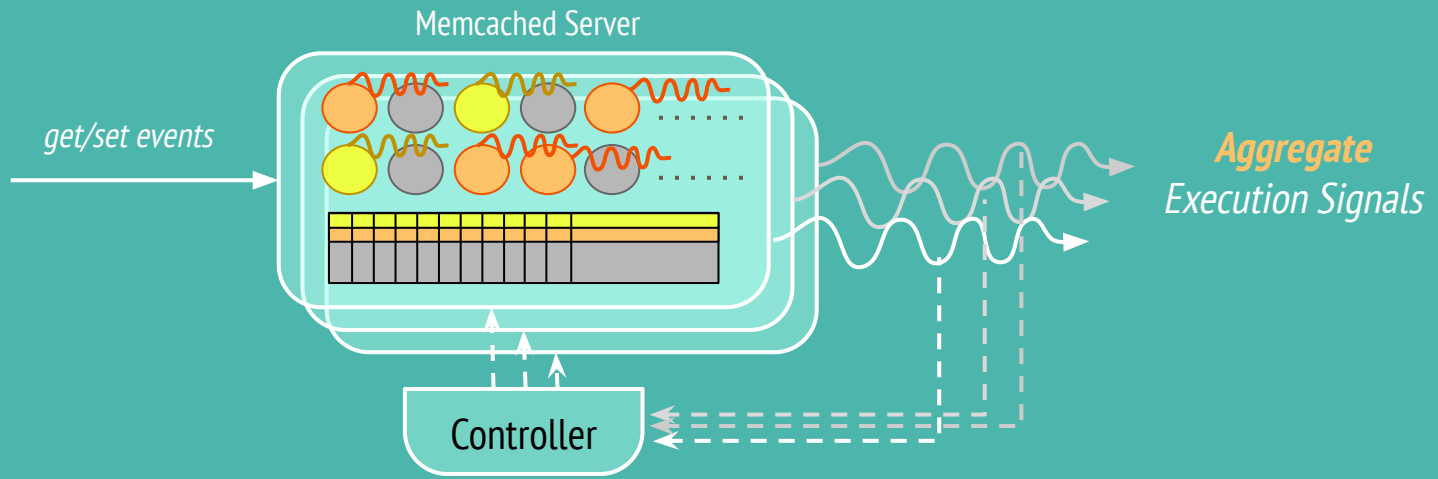
The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous *Software Agnostic* ***Feedback-Based***



The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

Exogenous *Software Agnostic* *Feedback-Based*



Could be the key toward enabling self-optimization in the face of dynamic event scenarios.

Exogenous *Software Agnostic* *Feedback-Based*

Goal: ***energy-aware, performance-based self-optimization in the face of dynamic network I/O.***

Exogenous *Software Agnostic* *Feedback-Based*

“The Case for Energy Proportional Computing”

L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," in Computer, vol. 40, no. 12, pp. 33-37, Dec. 2007, doi: 10.1109/MC.2007.443.

Energy proportional behavior: when the power draw of a computational unit is proportional to its utilization.

“The Case for Energy Proportional Computing”

L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," in Computer, vol. 40, no. 12, pp. 33-37, Dec. 2007, doi: 10.1109/MC.2007.443.

*“We loosely define **utilization** as a measure of the application performance— **such as requests per second on a Web server**—normalized to the performance at peak load levels.”*

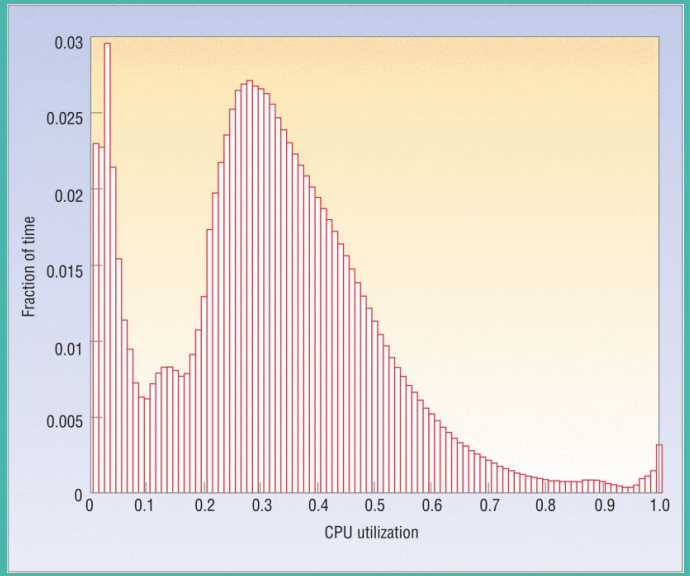


*Figure 1. **Average CPU utilization of more than 5,000 servers during a six-month period**. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.*

“The Case for Energy Proportional Computing”

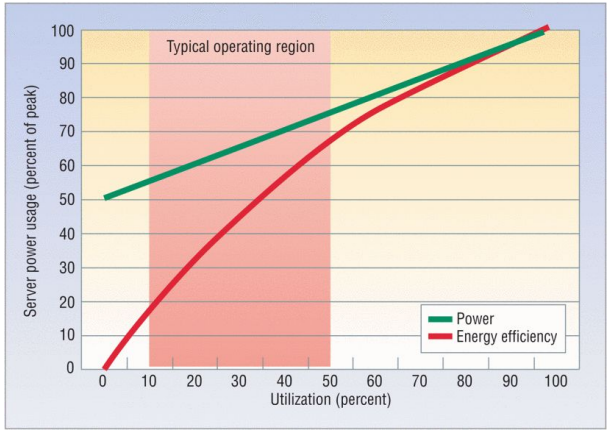


*Figure 2.Server power usage and energy efficiency at varying utilization levels, from idle to peak performance.**Even an energy-efficient server still consumes about half its full power when doing virtually no work.***

*“To derive power efficiency, we simply divide utilization by its corresponding power value. **We see that peak energy efficiency occurs at peak utilization and drops quickly as utilization decreases.**”*

Power draw should match utilization to achieve energy proportionality.

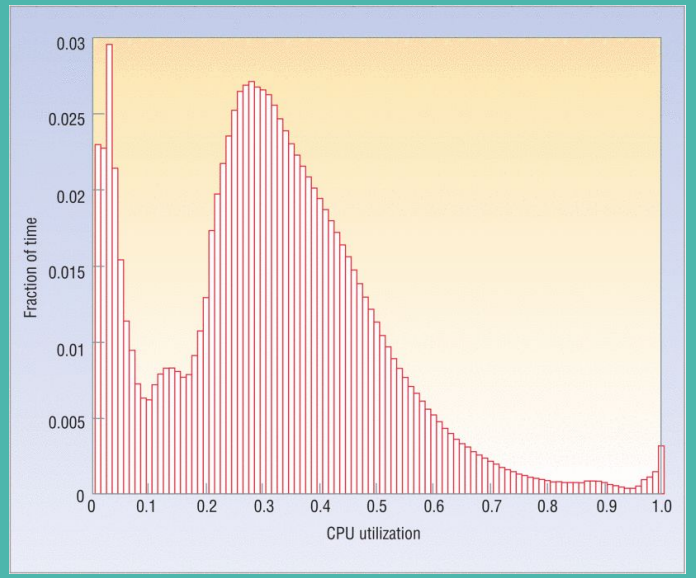
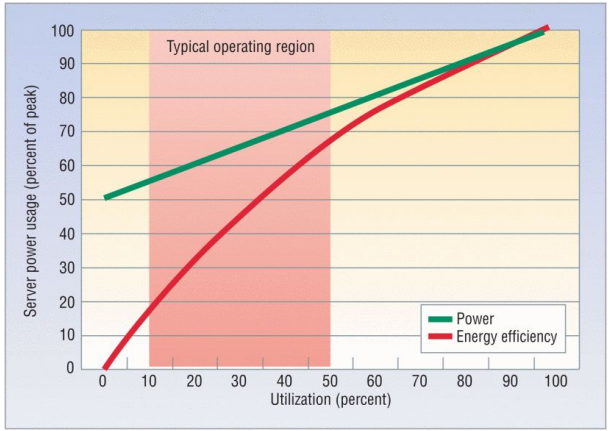
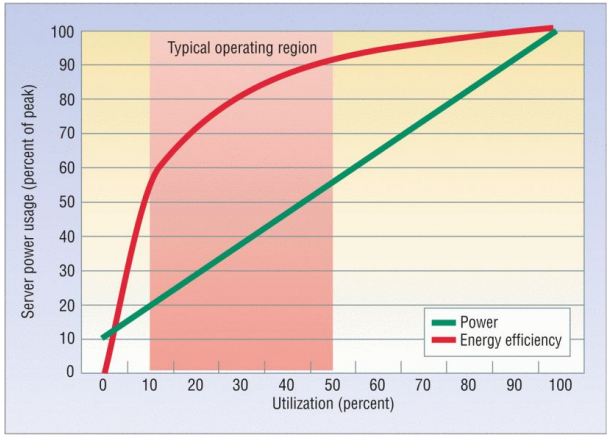


*Figure 1. **Average CPU utilization of more than 5,000 servers during a six-month period**. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.*

“The Case for Energy Proportional Computing”





Power draw should match utilization to achieve energy proportionality.

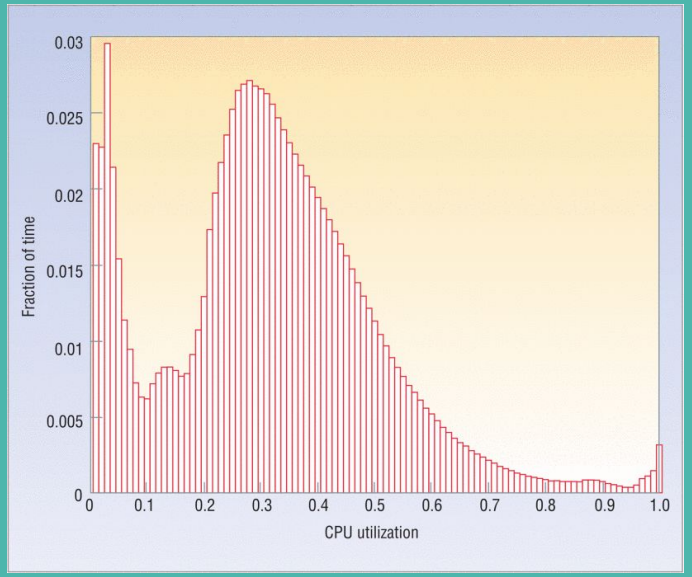


Figure 1. ***Average CPU utilization of more than 5,000 servers during a six-month period****. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.*

Power Management and *Computational Batching* for Performance Control

Two historically relevant mechanisms that dominate performance control solutions.

Systemic performance control exploits two main historically-relevant mechanisms.

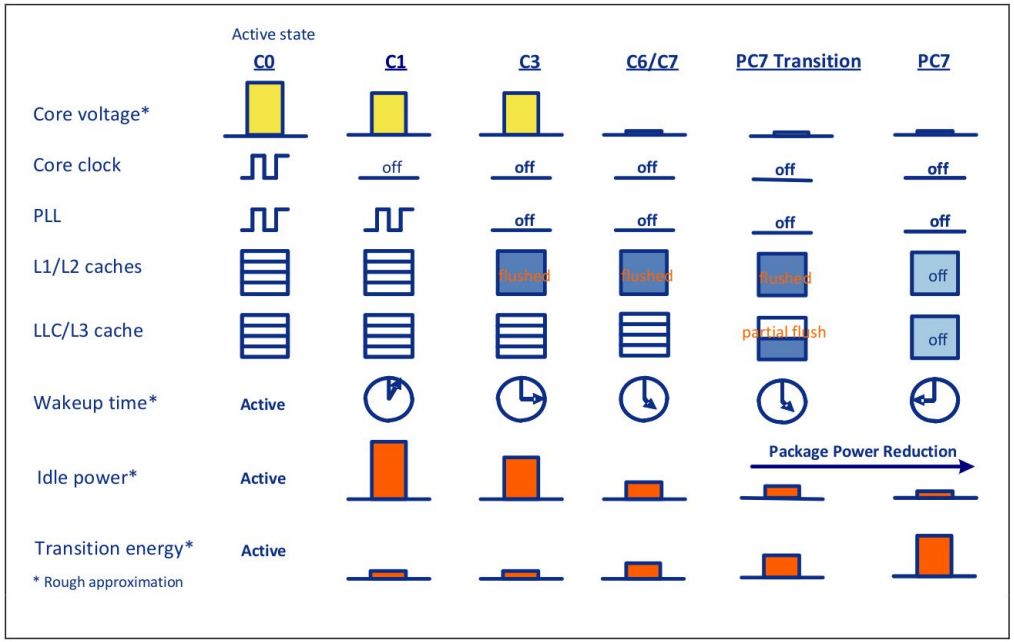
Power Management

Processor Frequency/Voltage
Idle State Entry/Exit

Computational Batching

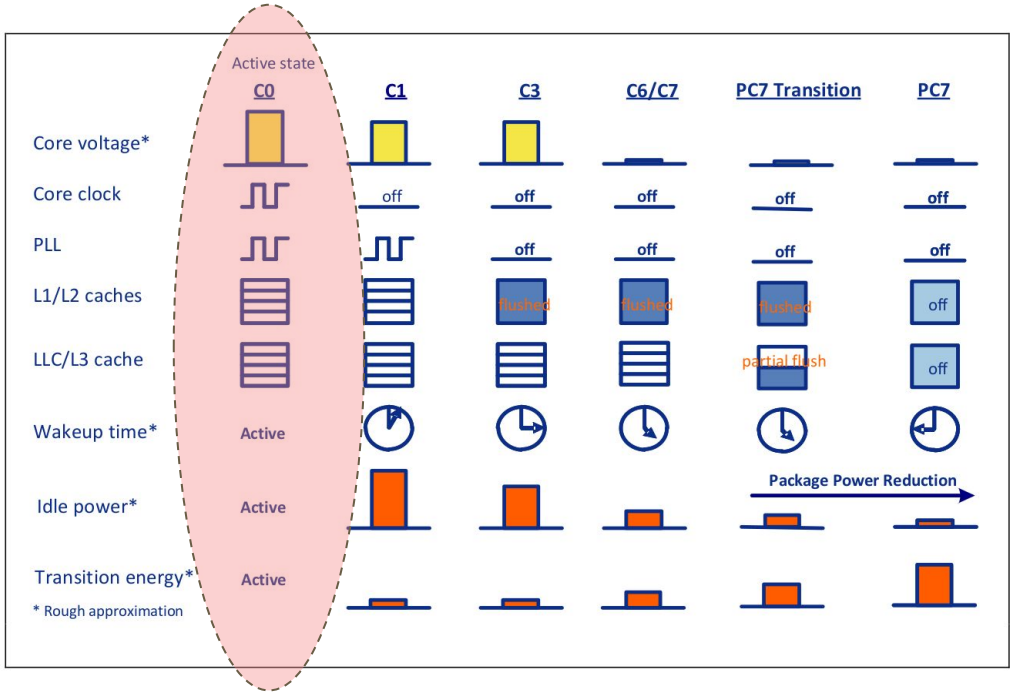
Power Management: Power-Saving and Sleep States

Sleep States (C-states)



Power Management: Power-Saving and Sleep States

Sleep States (C-states)



Power Management: Power-Saving and Sleep States

Power Saving States (P-states)

$$P_{CPU} = P_{dynamic} + P_{SC} + P_{leak}$$

$$\mathbf{P_{dynamic} \sim f V^2}$$

P_{SC} : short circuit power ~ f

P_{leak} : leakage power ~ V

System-level performance control exploits two main historically-relevant mechanisms.

Power Management

Processor Frequency/Voltage
Idle State Entry/Exit

Computational Batching

Batch-Aware Policies/Structures

System-level performance control exploits two main historically-relevant mechanisms.

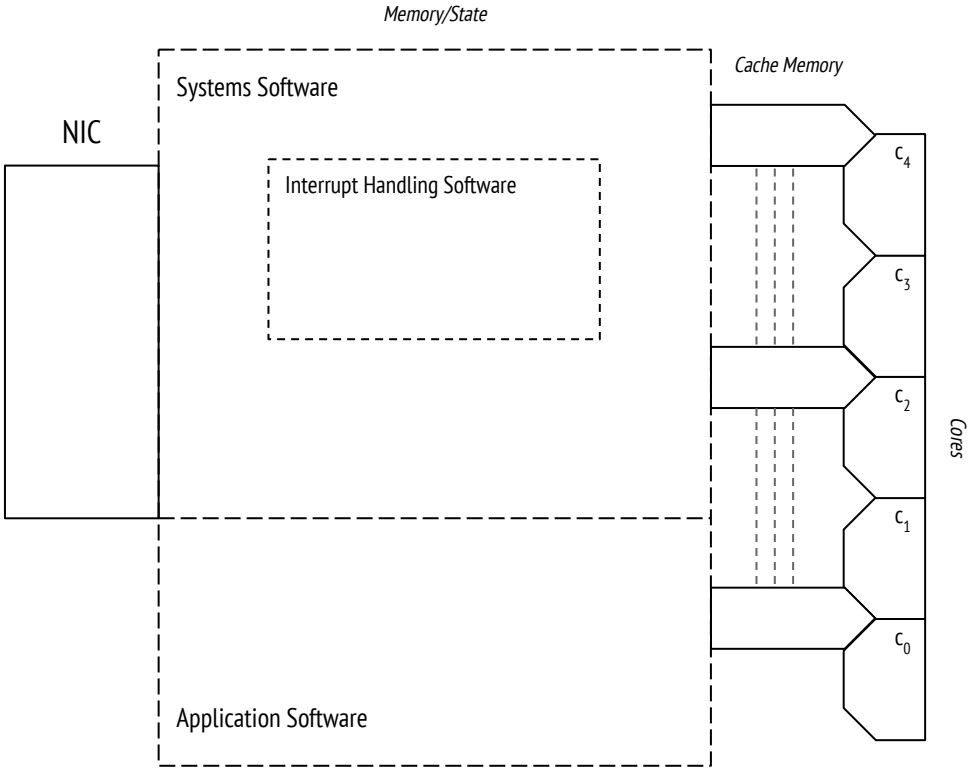
Power Management

Processor Frequency/Voltage
Idle State Entry/Exit

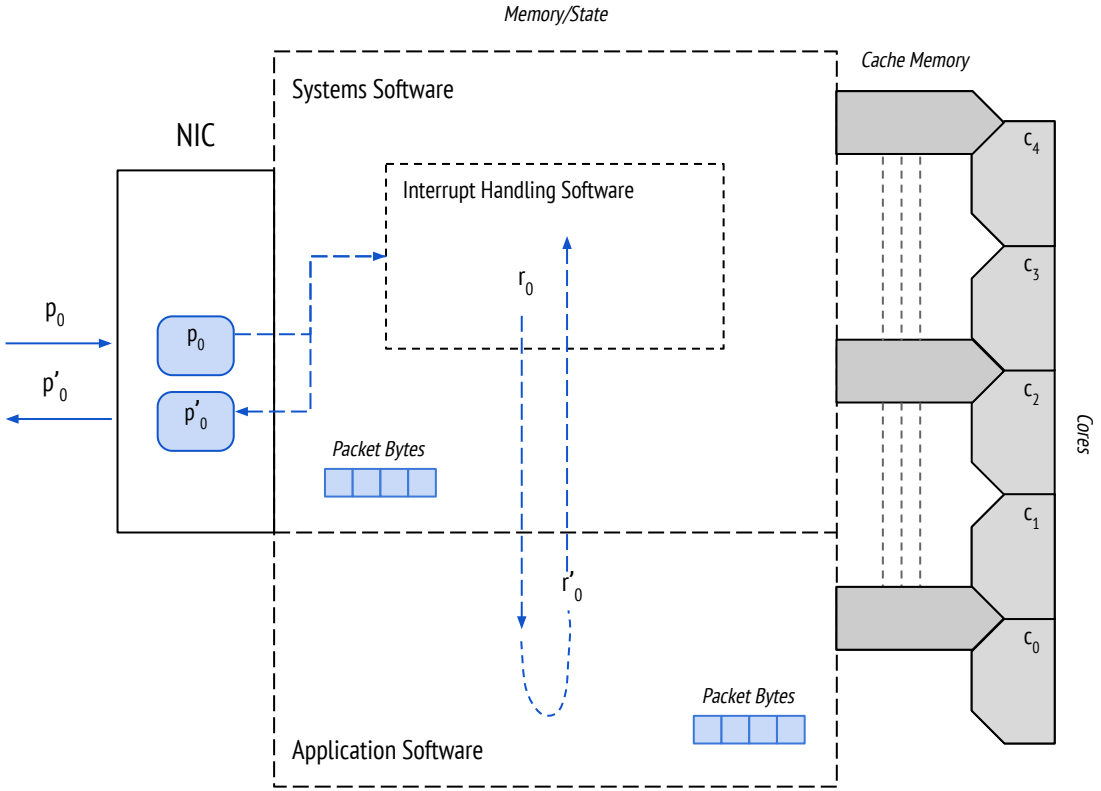
Computational Batching

Interrupt-Coalescing
Batch-Aware Policies/Structures

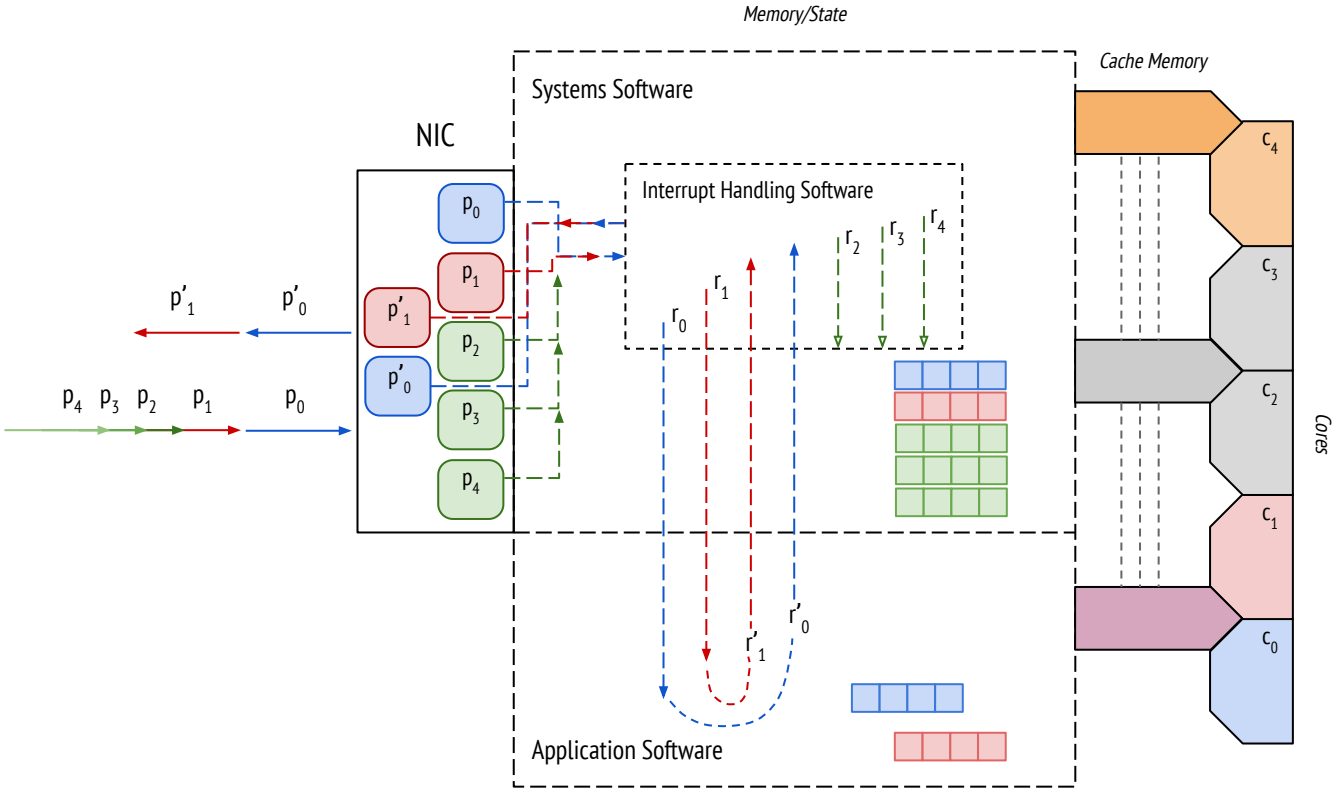
System Operation Under Network I/O: *An Illustration*



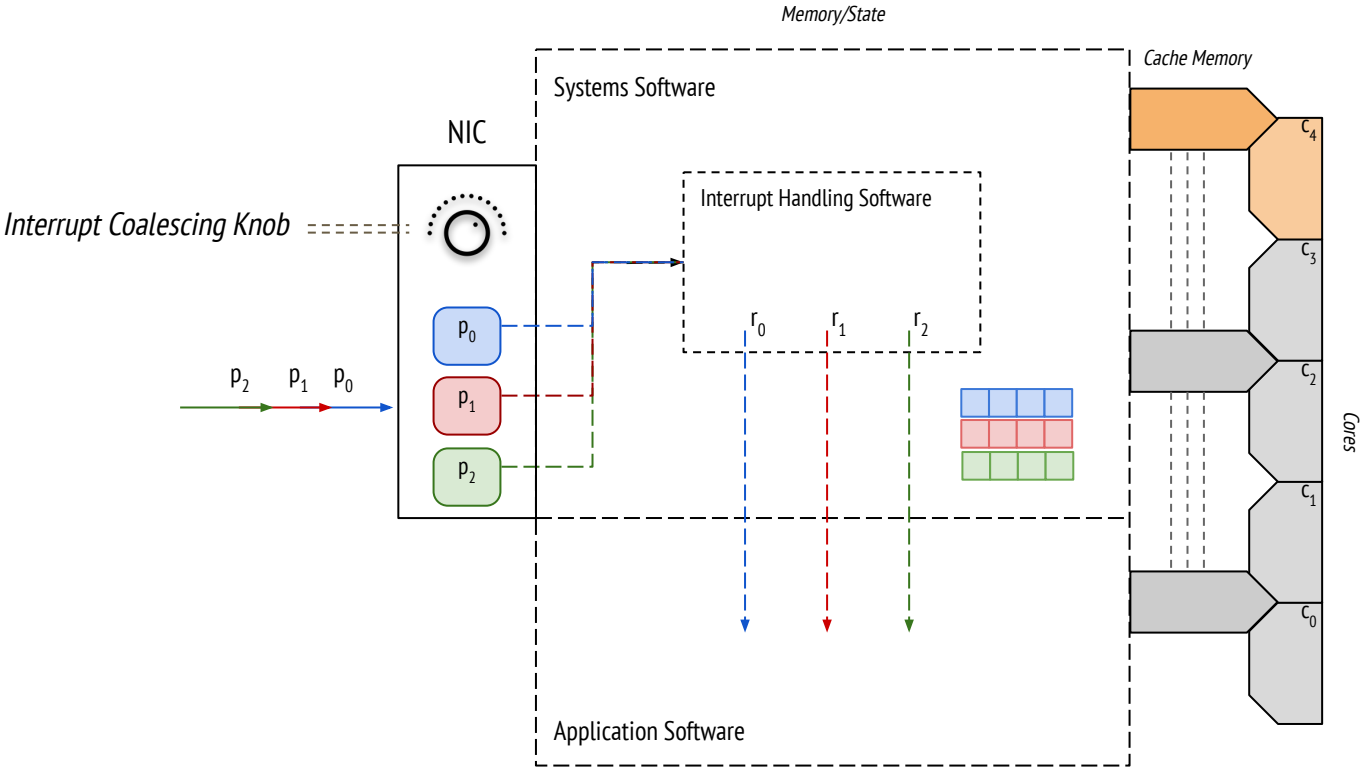
System Operation Under Network I/O



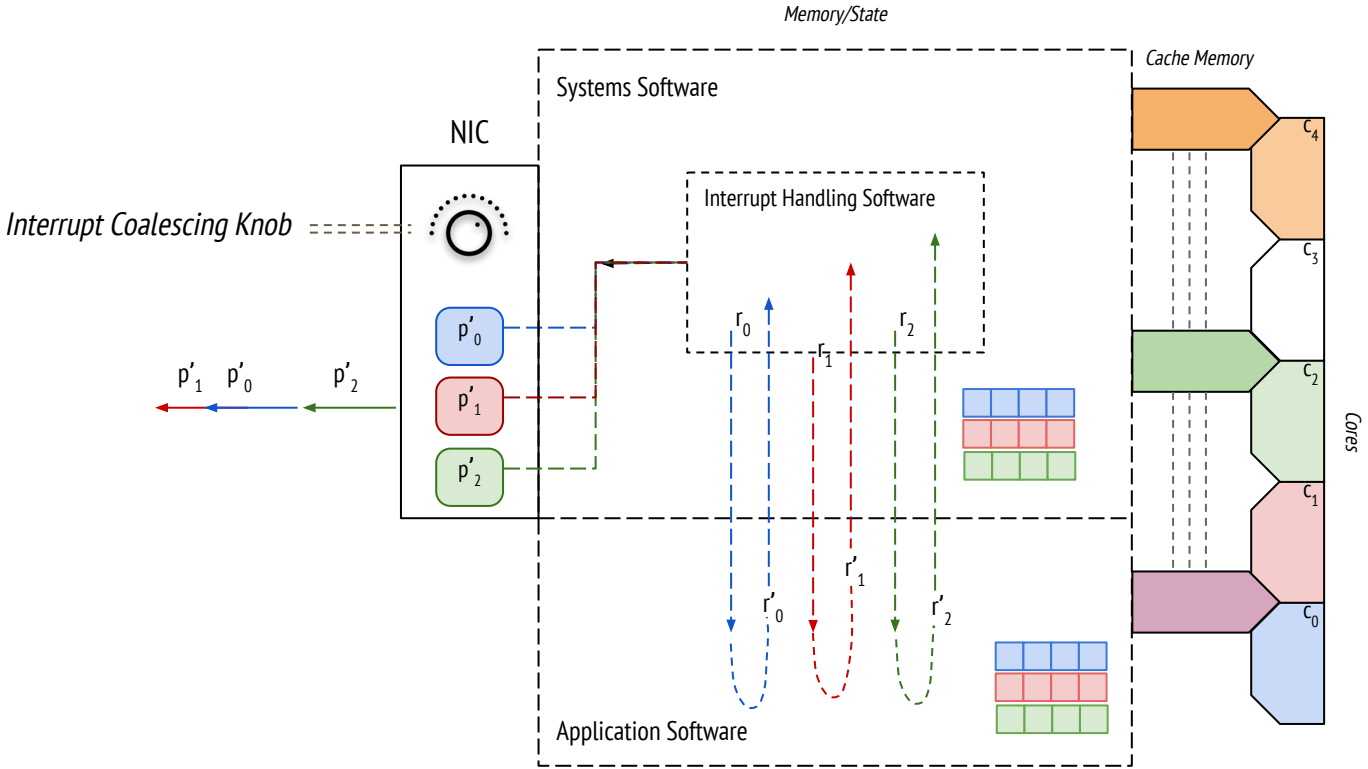
System Operation Under Network I/O



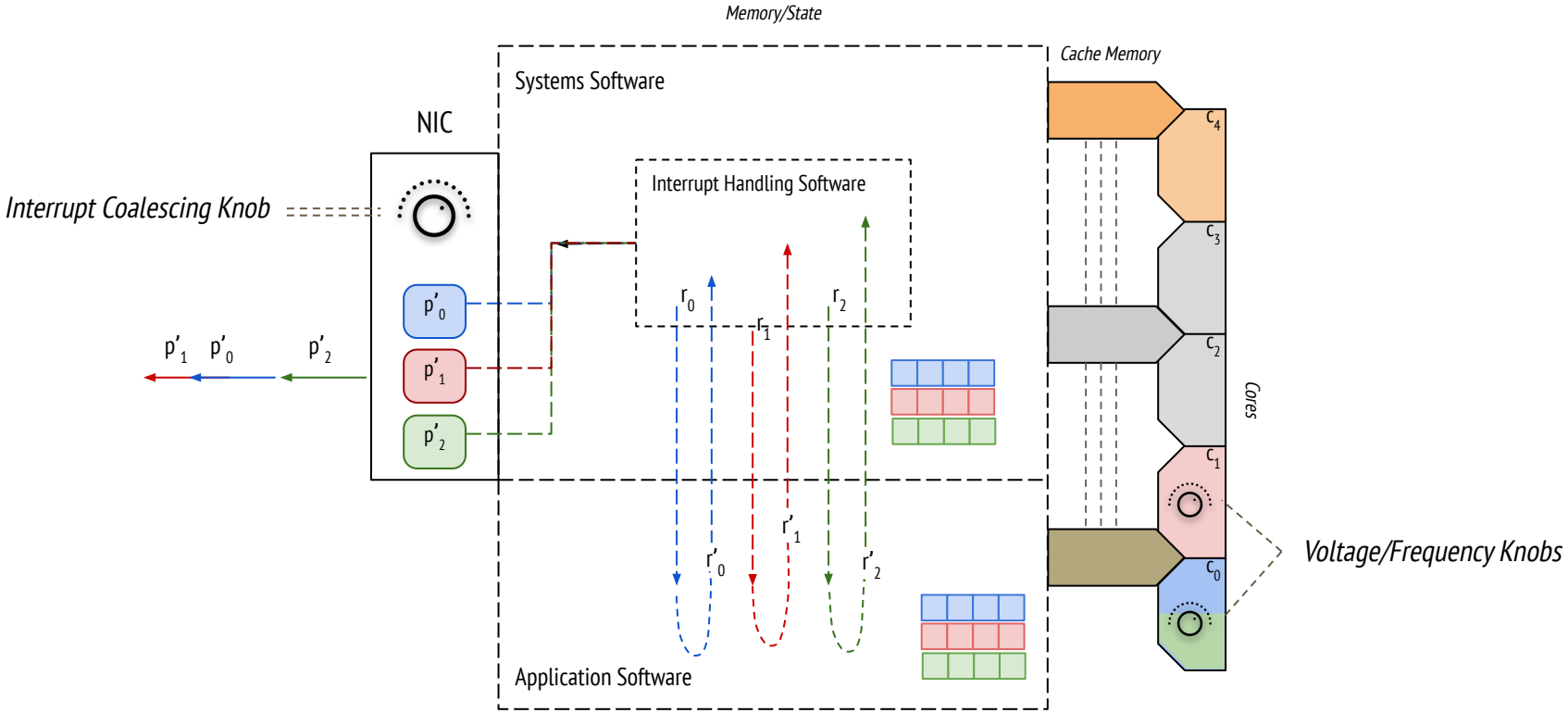
Interrupt Coalescing



Interrupt Coalescing



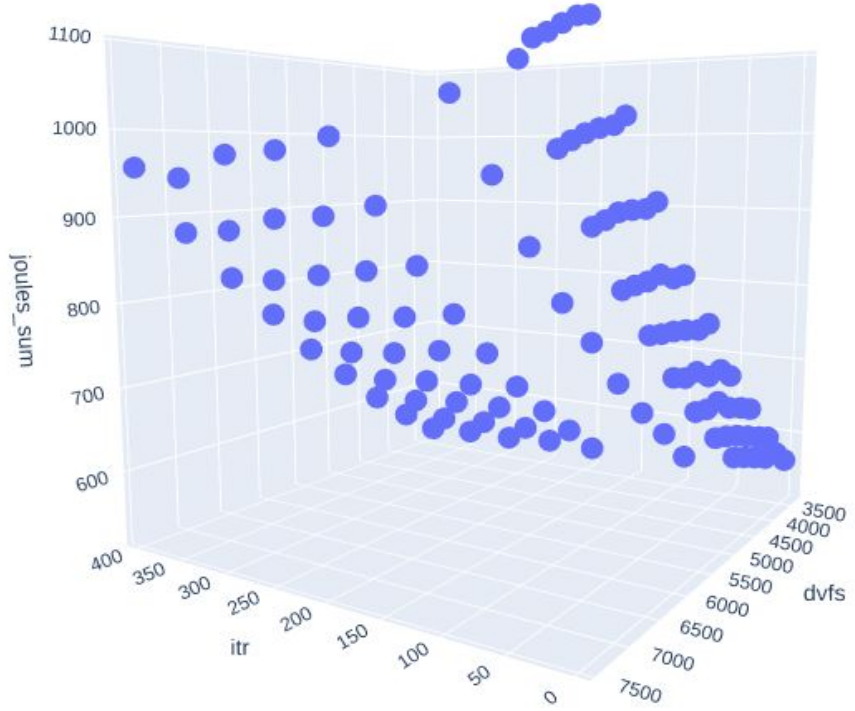
Interrupt Coalescing: *Interaction with Performance Control Mechanisms*



A Correlation:
CPU Voltage/Frequency and NIC Interrupt-Coalescing

When considered in tandem, they can magnify energy saving effects.

Measured Impact of Coupling Interrupt Coalescing and DVFS Control



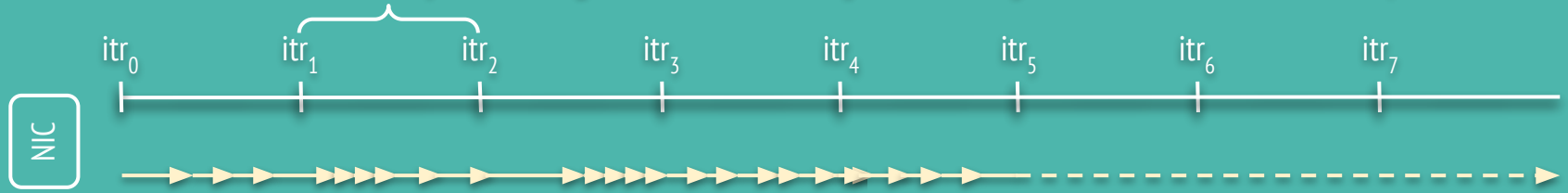
∃ Headroom for achieving *Lower Energy Consumption* with *Desired Performance* by Coupling Interrupt Coalescing and DVFS Control

https://arxiv.org/abs/2112.07010

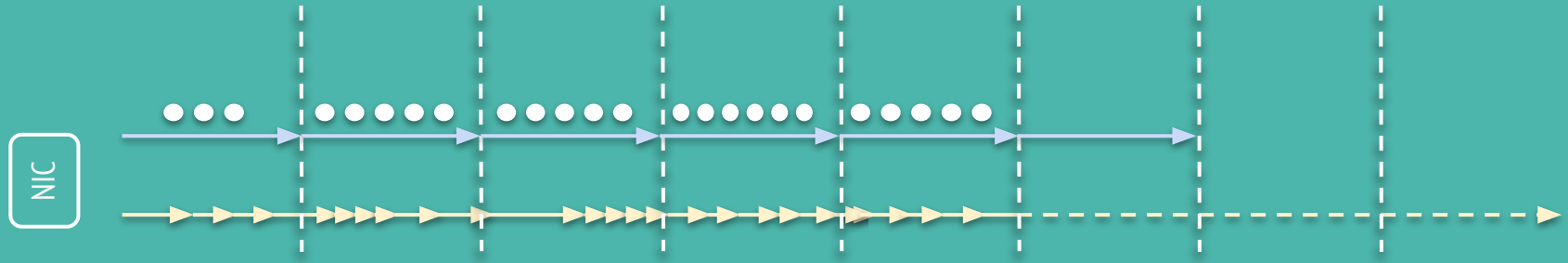
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

interrupt-coalescing: determines how long it takes to signal new work - i.e. *inter-interrupt rate*

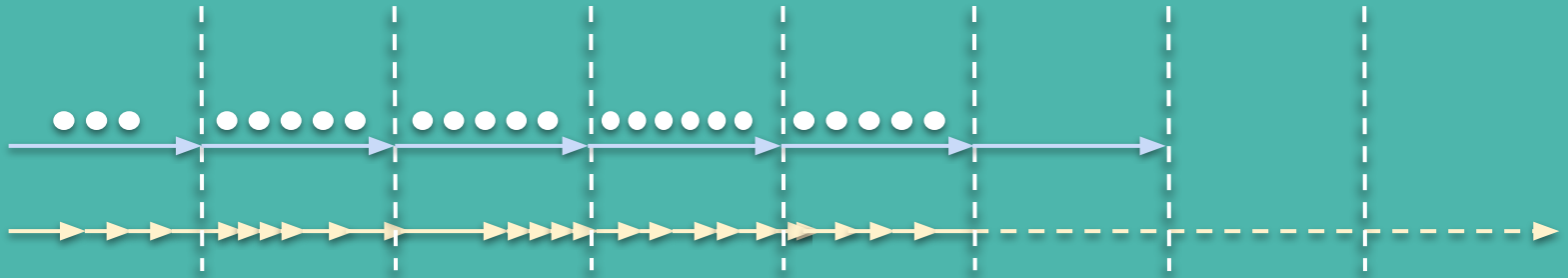


Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

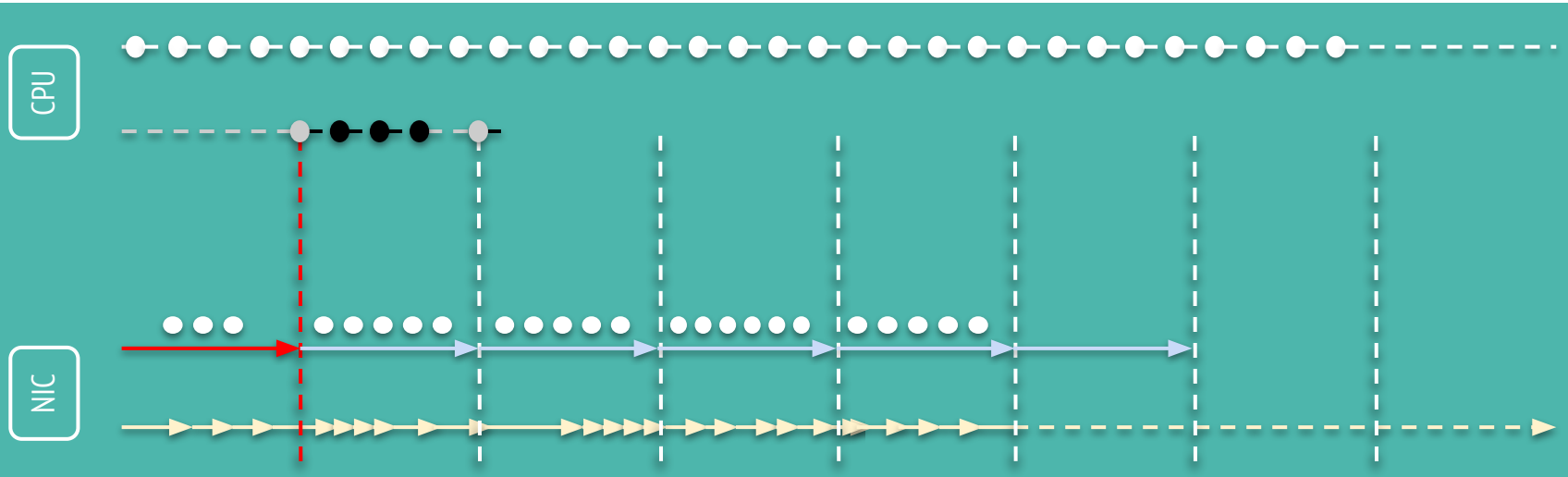


Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

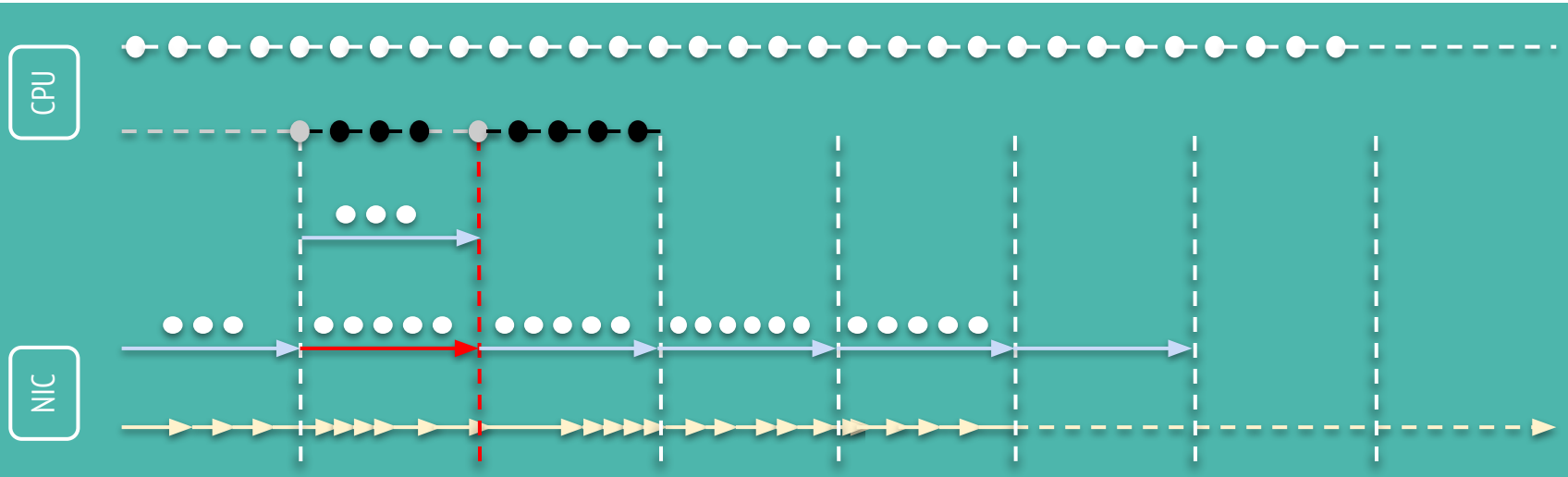
core voltage/frequency: determines how long it takes to complete a unit of computation - i.e. *intra-interrupt rate*



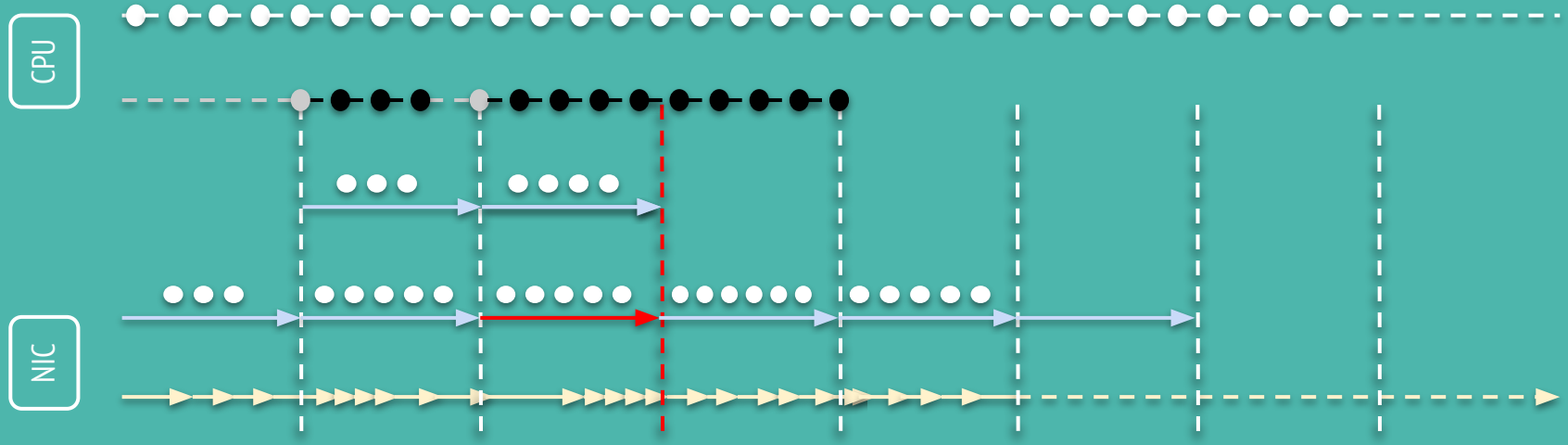
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



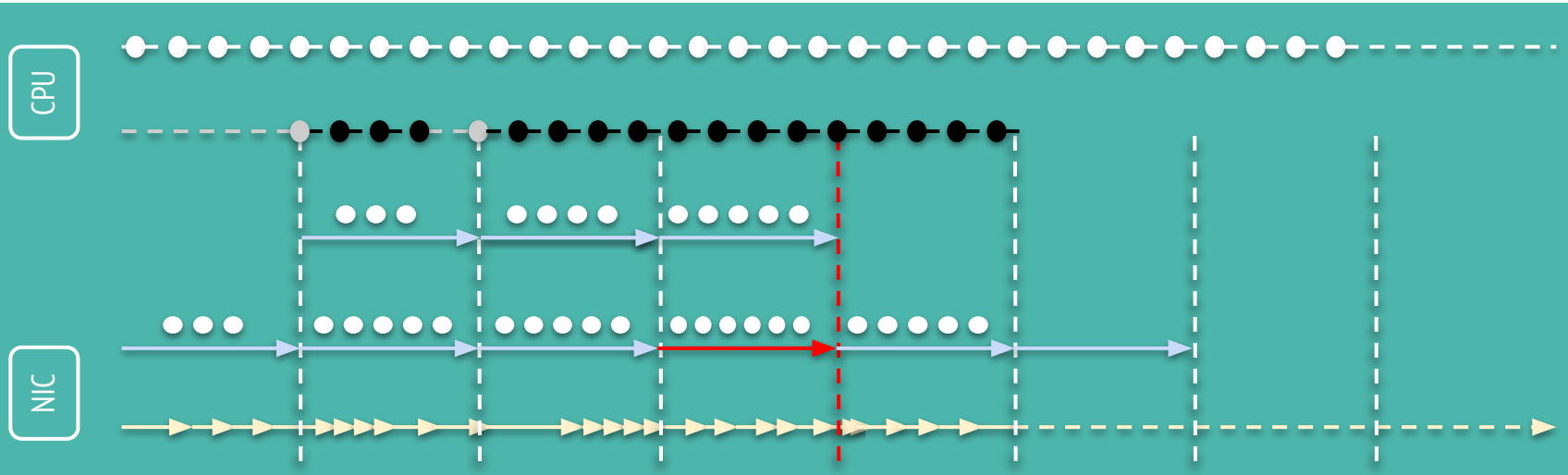
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



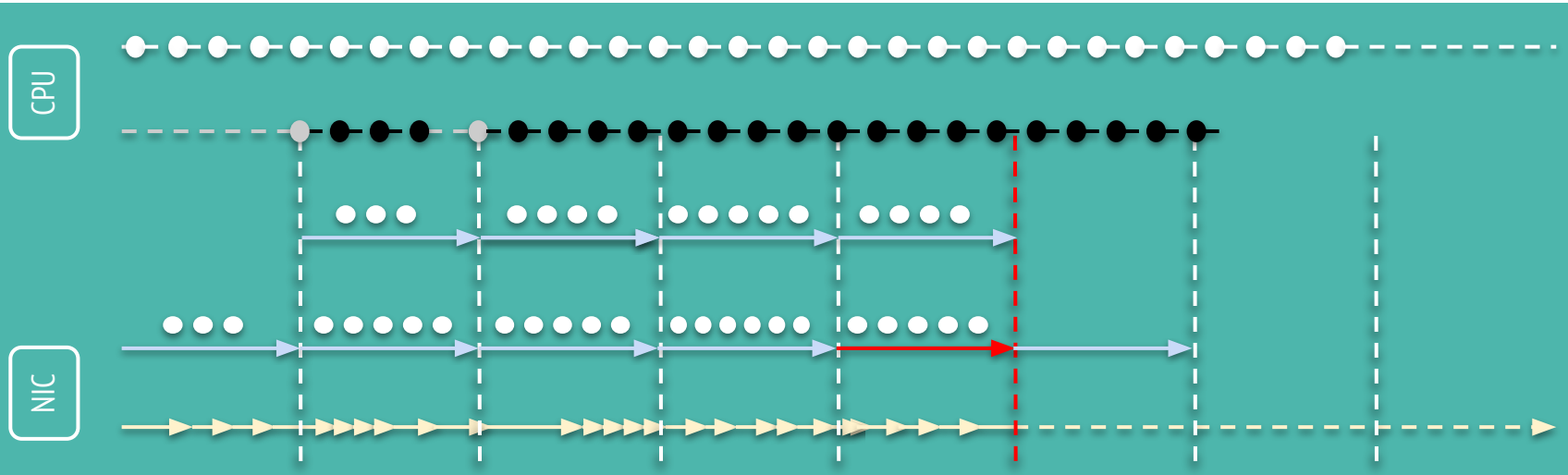
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



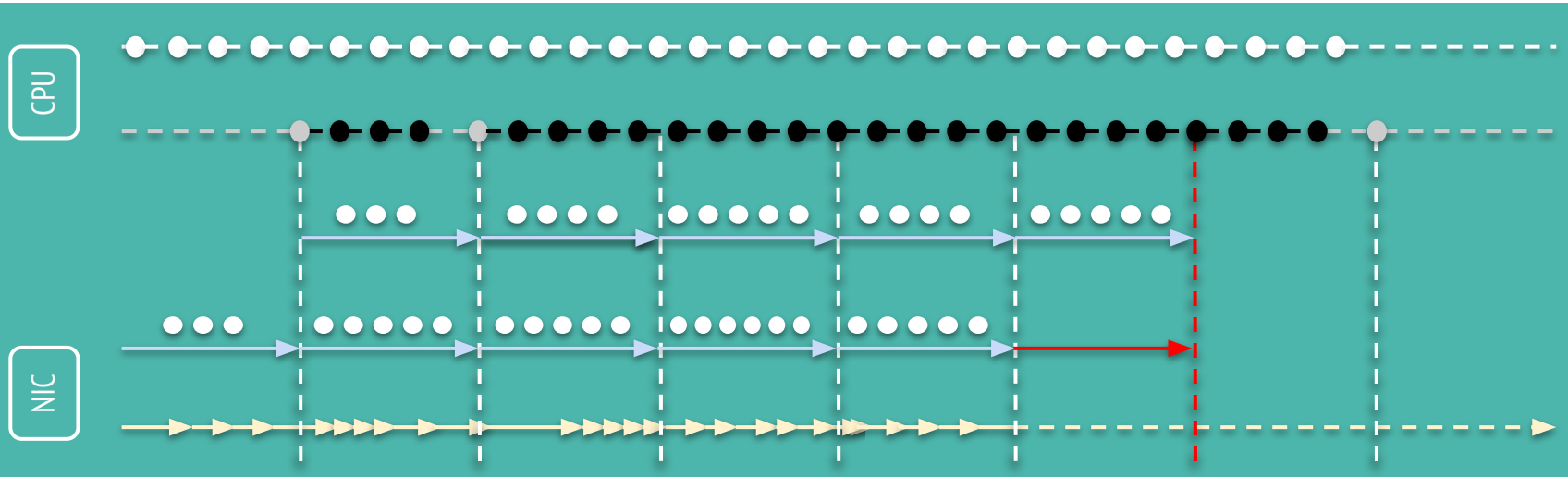
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



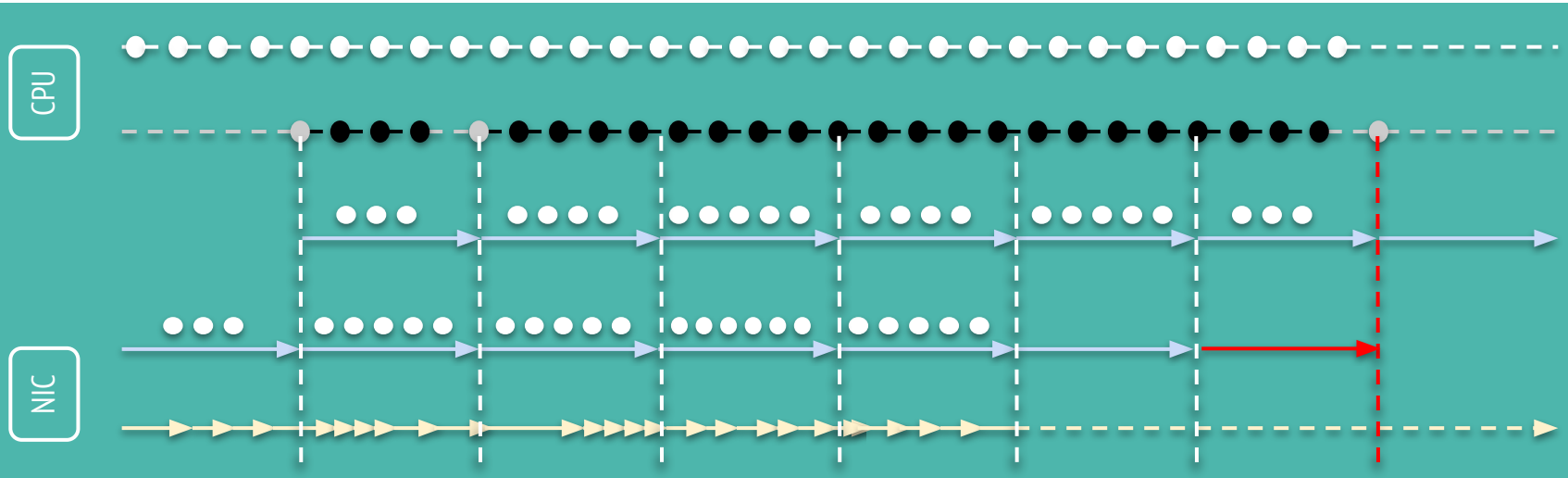
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



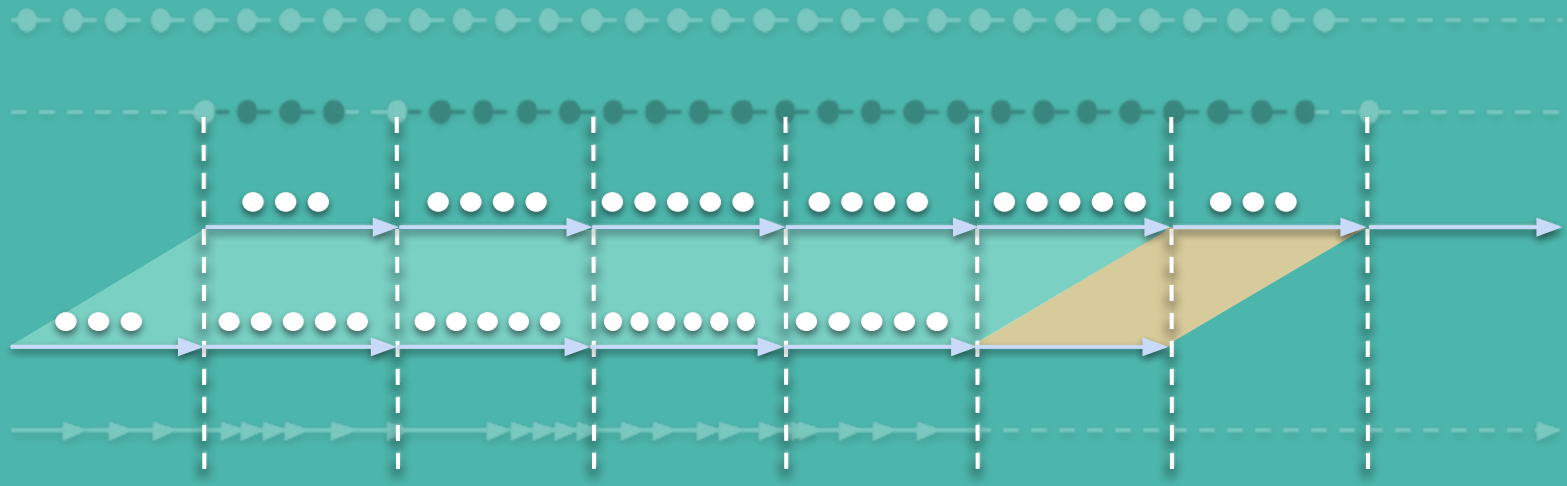
Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings



Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

We propose that rate matching is a missing system-level primitive that would synchronize the system's response to the world it is responding to.

Performance and Energy Control via Interrupt-Delay and DVFS Control

Interrupt Delay:

➔ NIC-level setting that determines the interrupt coalescing factor inherited by the system

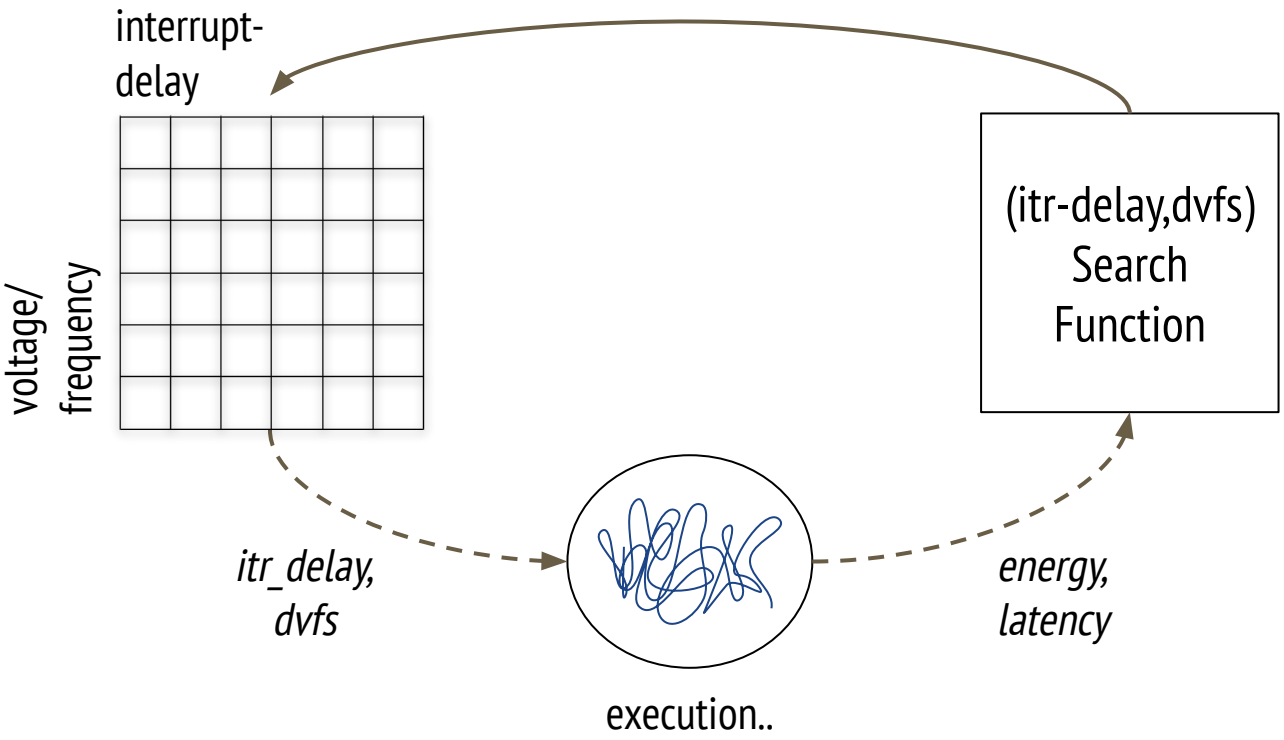
Dynamic Voltage/Frequency Scaling (DVFS):

➔ CPU-level mechanism that determines the frequency and voltage (i.e. power draw) of a processing unit

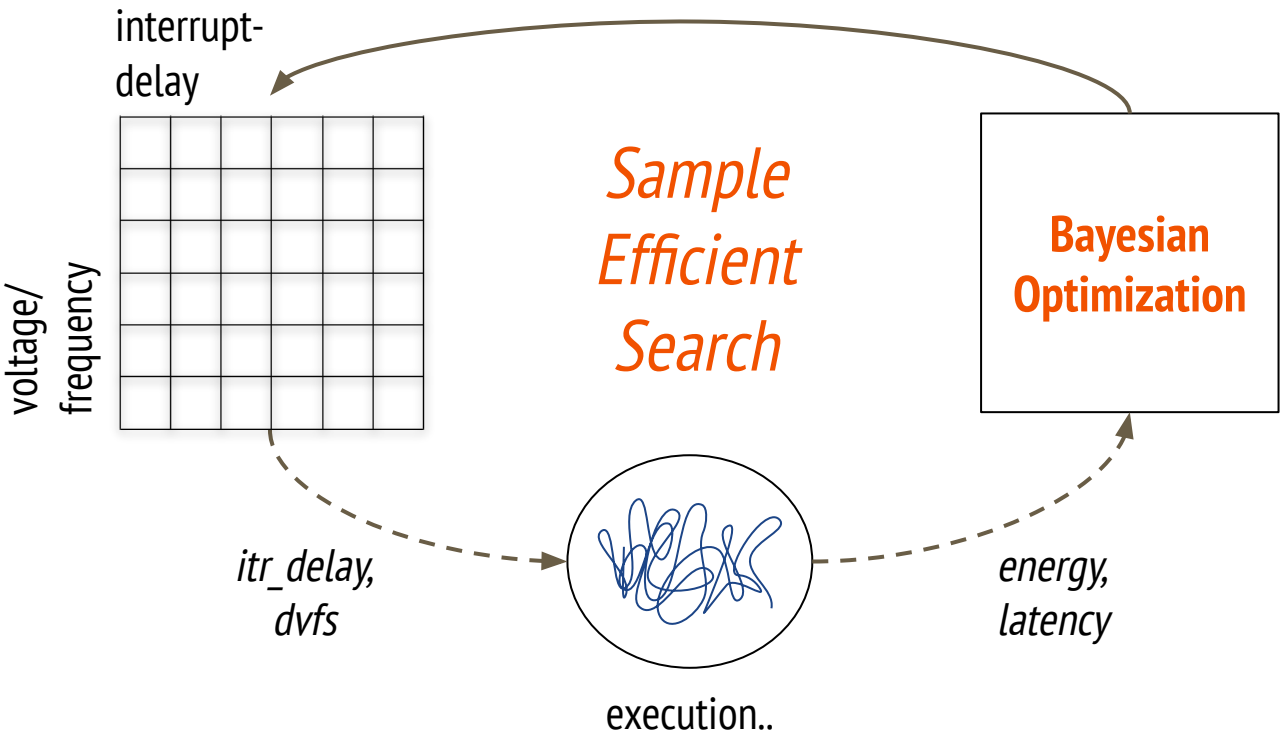
Implementation Target:

Bayesian Optimization Performance Controller

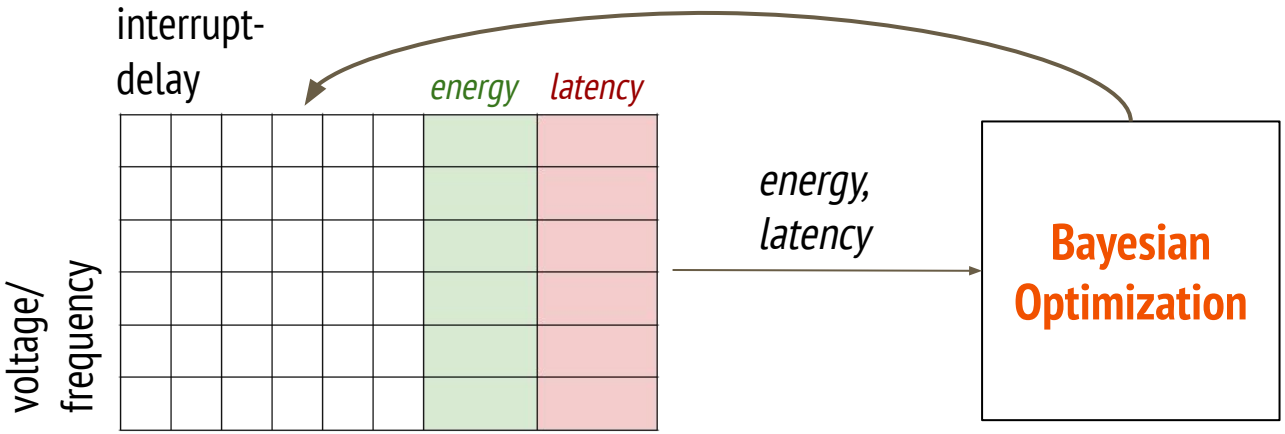
Bayesian Optimization Performance Controller: *Overview*



Bayesian Optimization Performance Controller: *Overview*

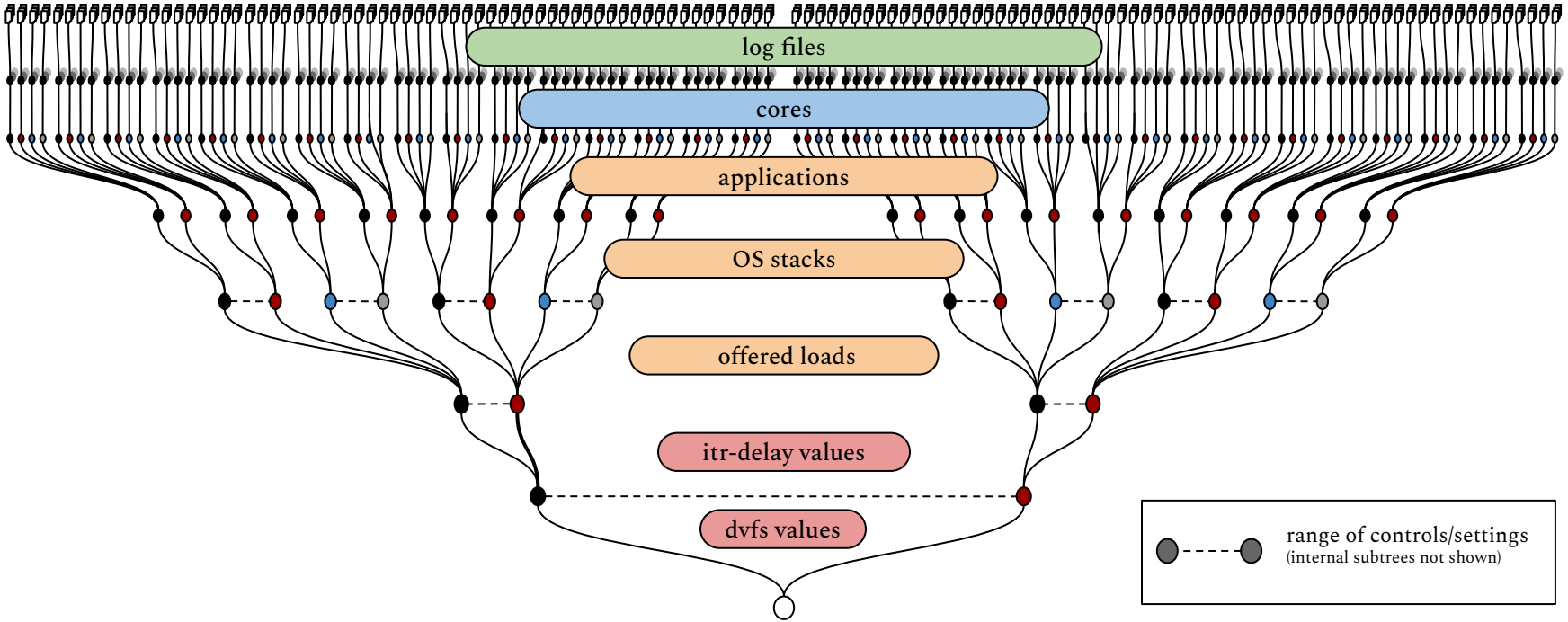


Bayesian Optimization Performance Controller: *Dataset/Environment*



We use a comprehensive dataset of **per-application** and **per-request-rate** *energy and performance measurements* under **different interrupt-delay and voltage/frequency** settings.

Bayesian Optimization Performance Controller: *Dataset/Environment*

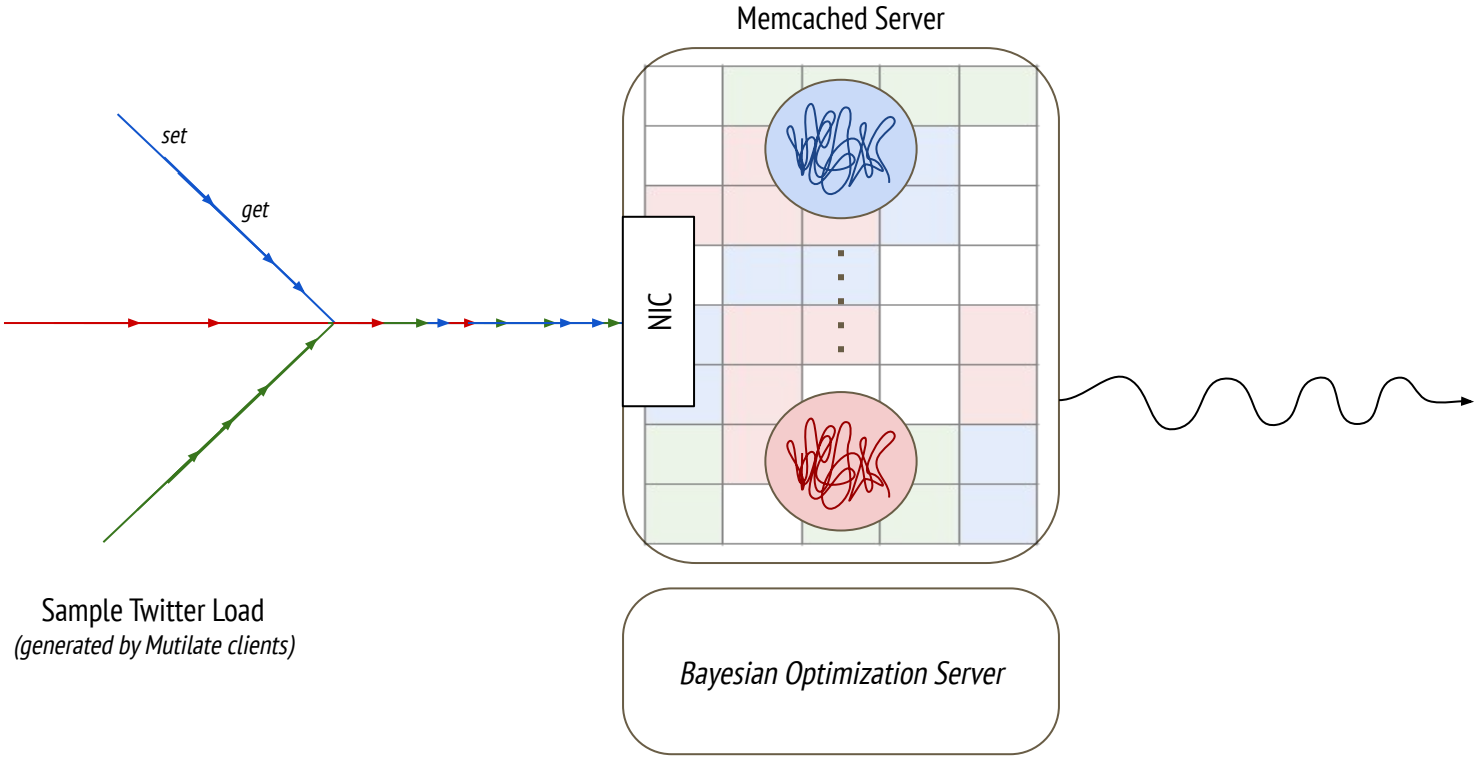


Exploiting Interrupt-Delay and DVFS Control in a Simulated Network Setting

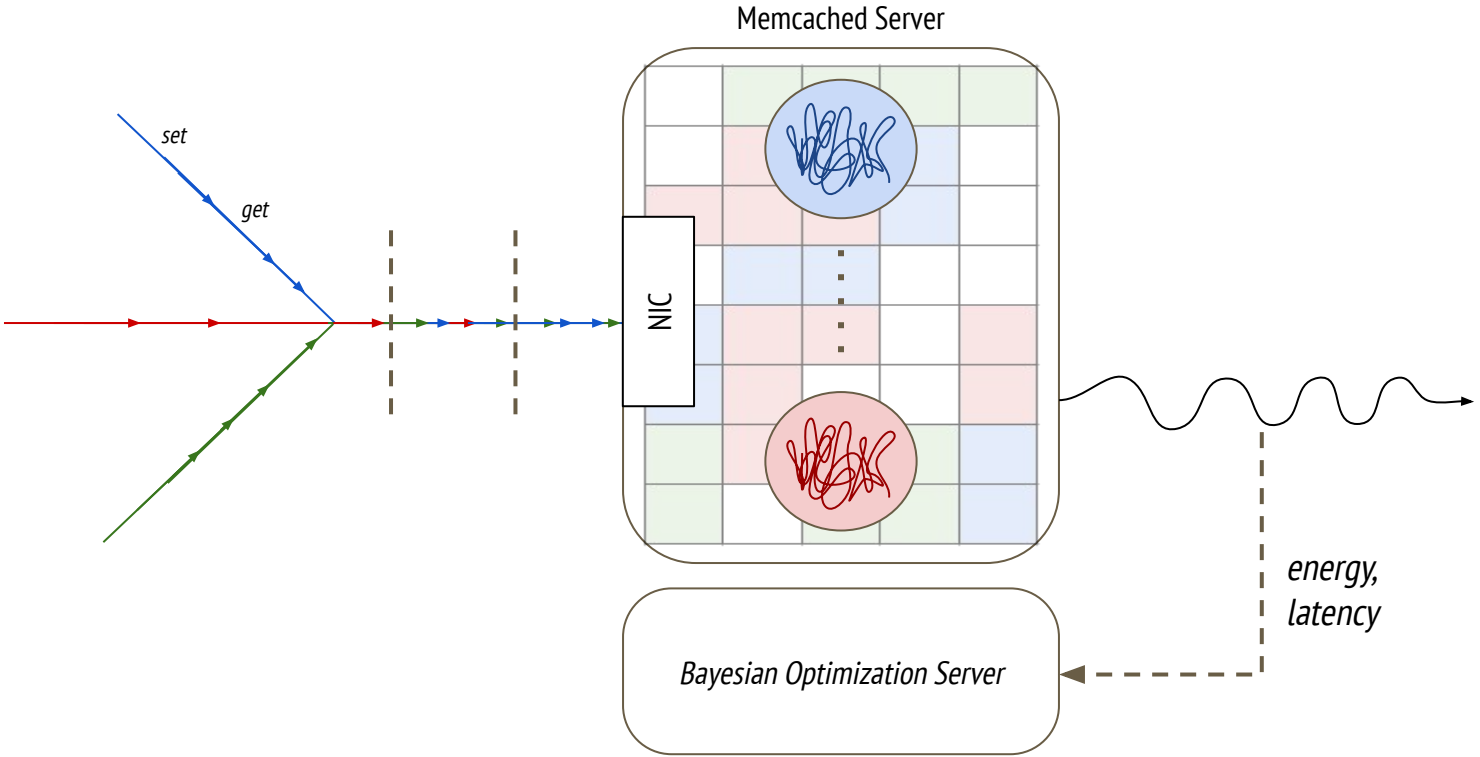
An Experiment

Simulating a Twitter-like network scenario for a Memcached-serving system.

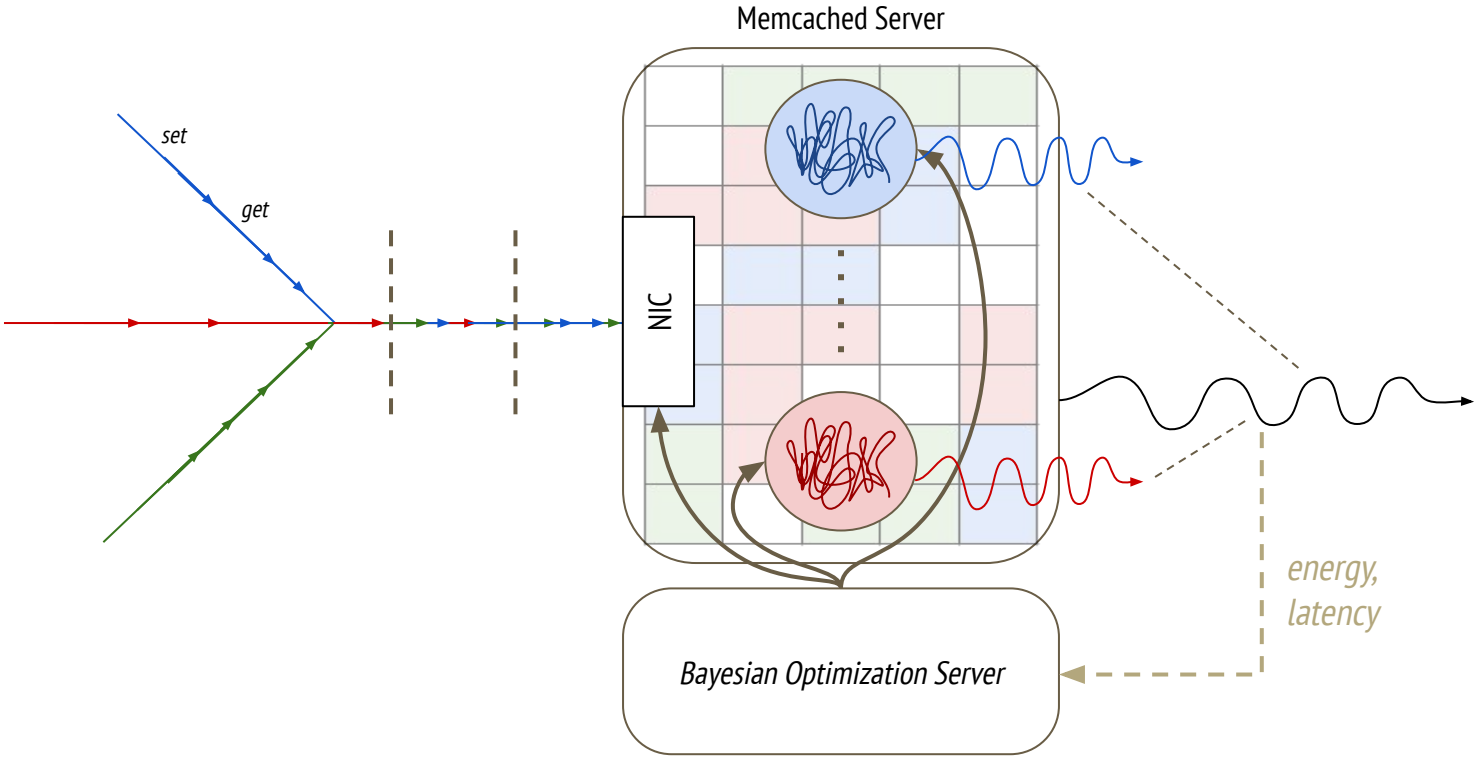
Simulating a Twitter-like Network Scenario for a Memcached-serving System



Simulating a Twitter-like Network Scenario for a Memcached-serving System



Simulating a Twitter-like Network Scenario for a Memcached-serving System

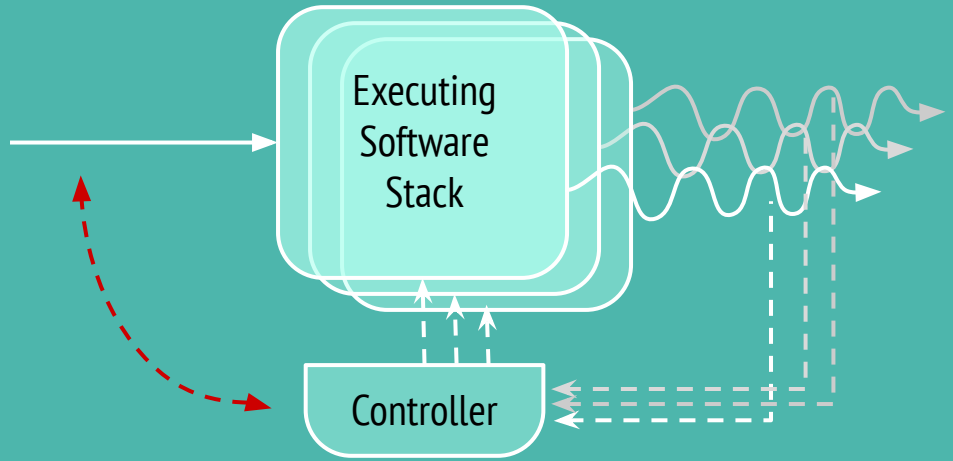


Implementation Target:

Rate-Aware Bayesian Optimization Performance Controller

The ability to affect system change - i.e. *control* - exogenously: not from within the system itself, but rather from a control space that is external to the system.

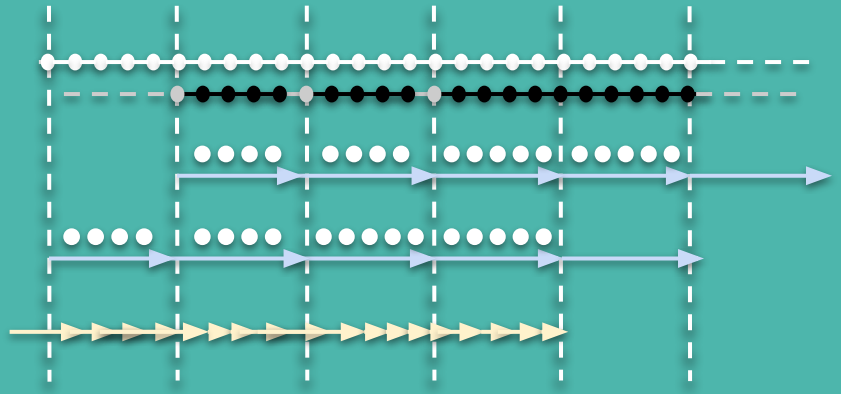
Exogenous *Software Agnostic* ***Feedback-Based***

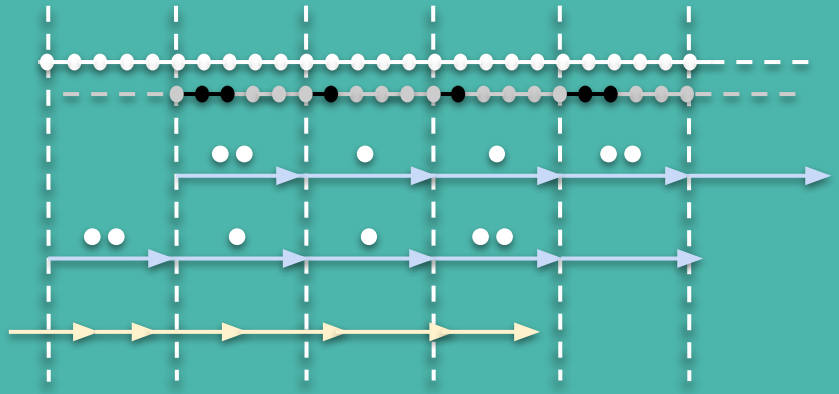


Implementation Target:

‘Rate-Aware’ Bayesian Optimization Performance Controller

Difficulty in Simulating and Modeling the Network





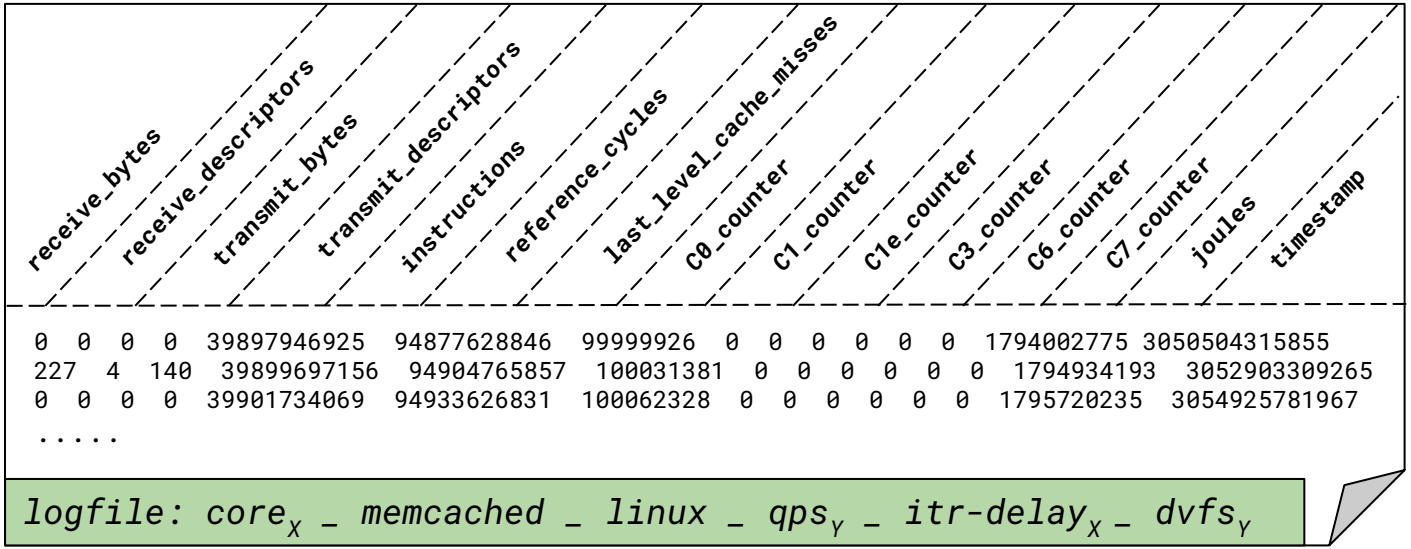
Instead: Modeling System/Network Interaction

Representation of System/Network Interaction

An *execution signal*:

➔ a representation of system behavior.

Execution Signal: Execution Log



Implementation Target:

Workload-Aware Reinforcement Learning Performance Controller

Potential Findings and Open Questions

Systemic performance control exploits two main historically-relevant mechanisms.

Power Management		Computational Batching	
Processor Frequency/Voltage	H/W	Interrupt-Coalescing	H/W
Idle State Entry/Exit	S/W	Batch-Driven Policies/Structures	S/W

System-Level Control of P-States (DVFS)

- e.g. Pegasus

System-Level Control of C-States

- e.g. PowerNap

System-Level Control of Interrupt Coalescing

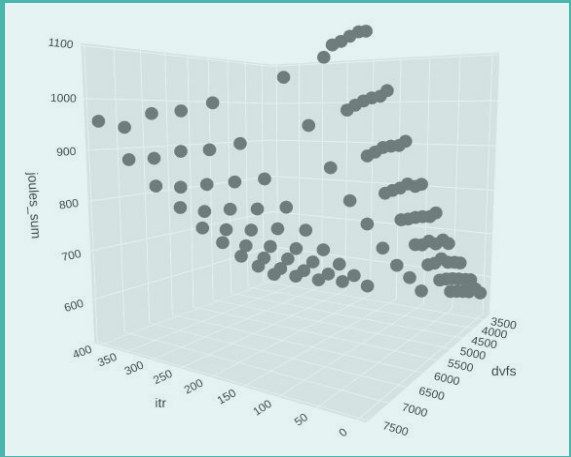
A large body of research presents solutions toward energy proportional computing.

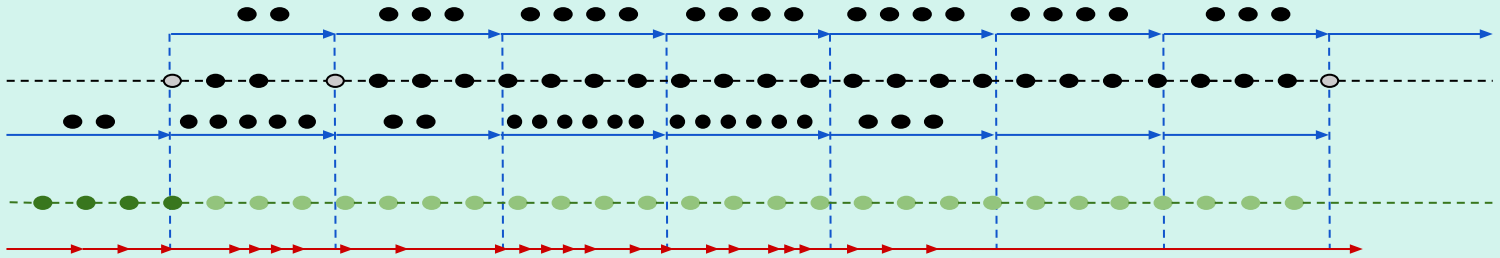
Hybrid Solutions Toward Energy Proportionality

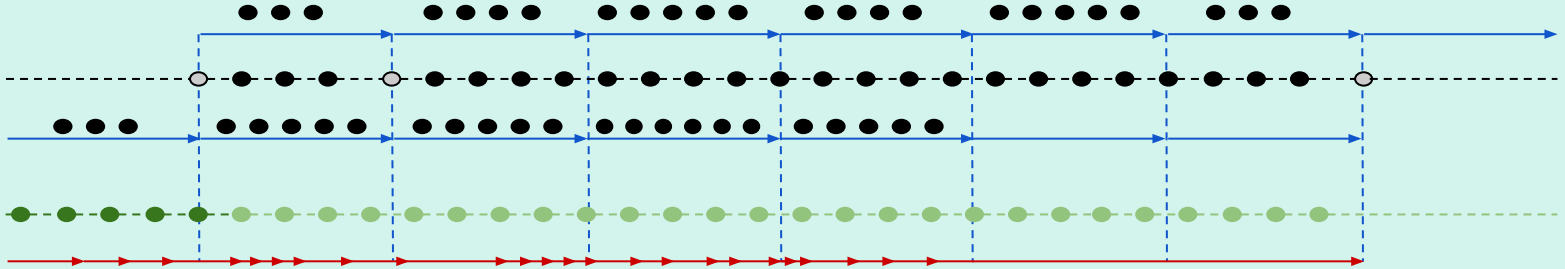
- e.g. SmoothOperator

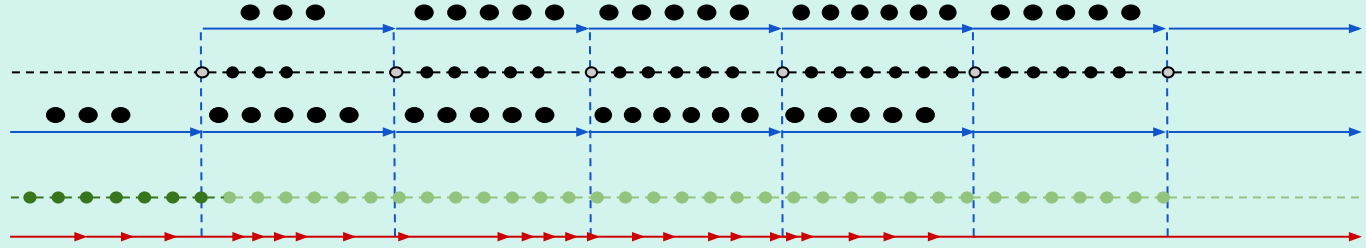
Control of P-States: Dynamic Voltage and Frequency Scaling (DVFS)

Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings

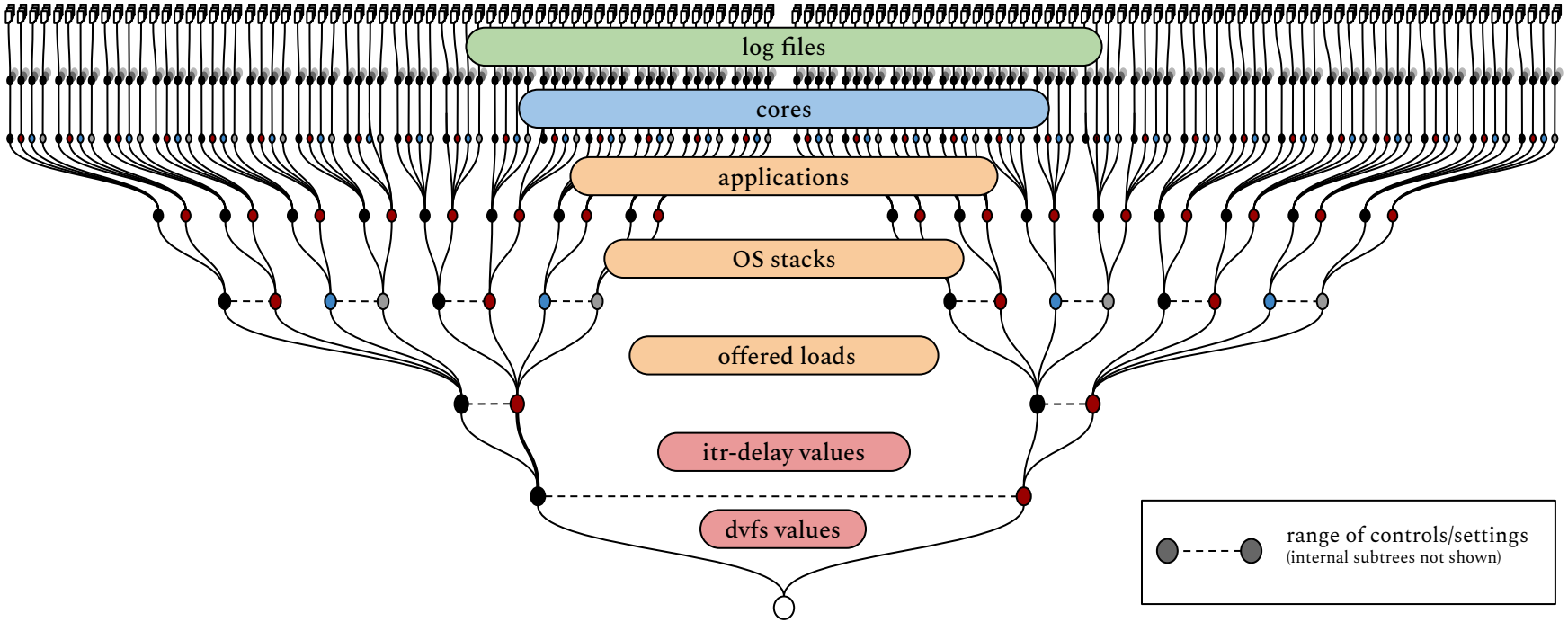






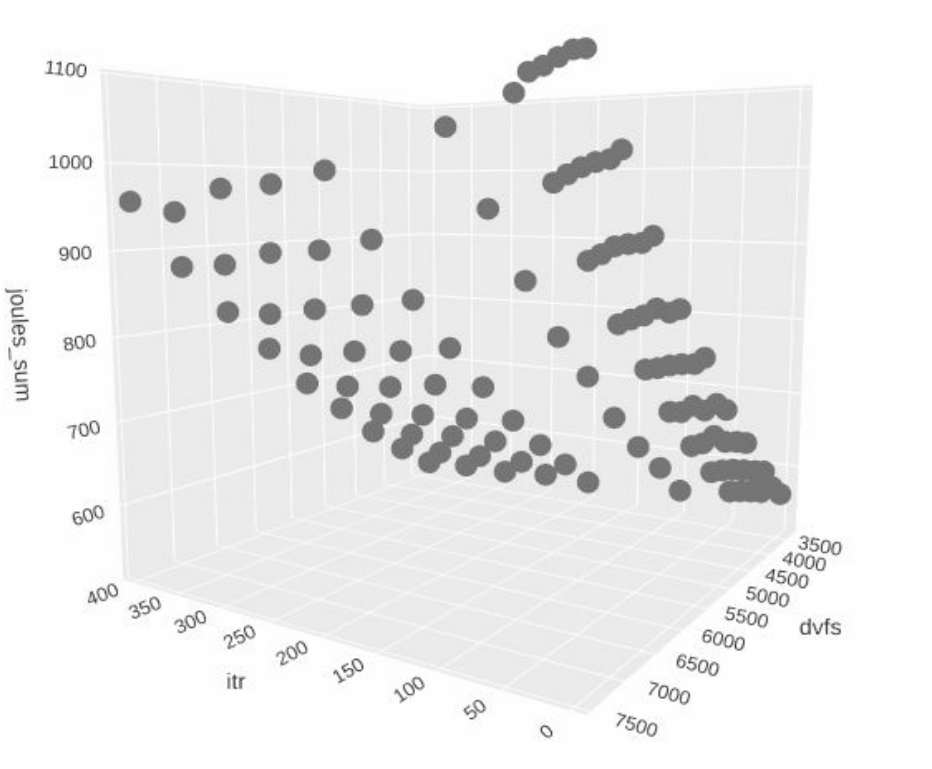


Dataset


log files
cores
applications
OS stacks
offered loads
itr-delay values
dvfs values
range of controls/settings
(internal subtrees not shown)

Execution State: NIC + Processor State

Compare: Impact of General-Purpose Linux Control



Reinforcement Learning: Performance Control ‘Game’

Reinforcement Learning

Per Network Workload, ∃ Relationship between Net Energy Consumption and Interrupt-Coalescing + DVFS Settings