Energy Optimising Methodologies on Heterogeneous Data Centres

Rajiv Nishtala

Xavier Martorell (UPC, BSC)
Daniel Mossé (UPitt)

July 10th, 2017

Lectura for Doctor of Philosophy

Talk Organisation

- Motivation
- Thesis contributions
- REPP: Runtime Estimation of Performance and Power
- Hipster: Hybrid Task Manager for Latency-Critical Cloud Workloads
- Future work & Contributions recap

- U.S Data Centres power consumption ≈ Power of **2X** houses in NY

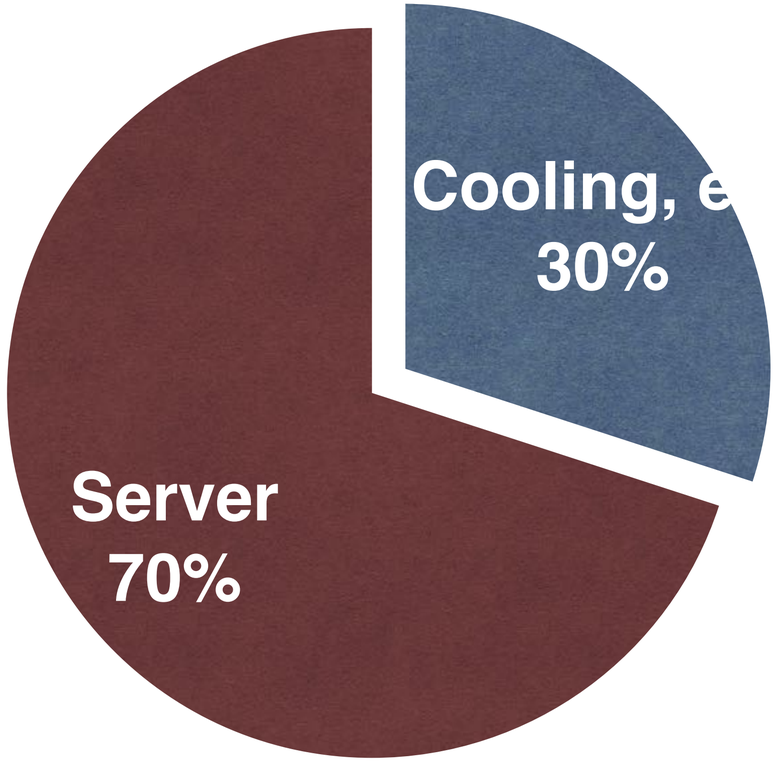
- U.S Data Centres power consumption ≈ Power of **2X** houses in NY
- U.S Data Centres pollution ≈ Pollution of UK

Problems for Data Centres

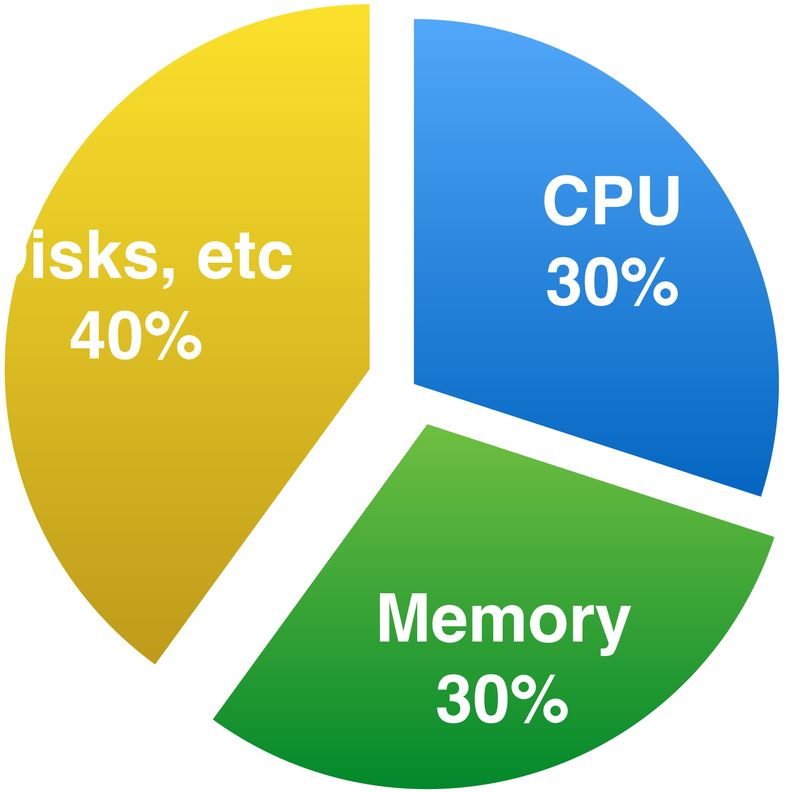
- Cost
  - Setting up a new data centres is expensive
  - Power is of limited availability
  - Improving power delivery systems is not feasible
- Scalability
  - New data centres cost a lot of money
  - More servers per data centre, costs more money

Where does all the energy go?

How do data centres utilise energy given?



Data Centre



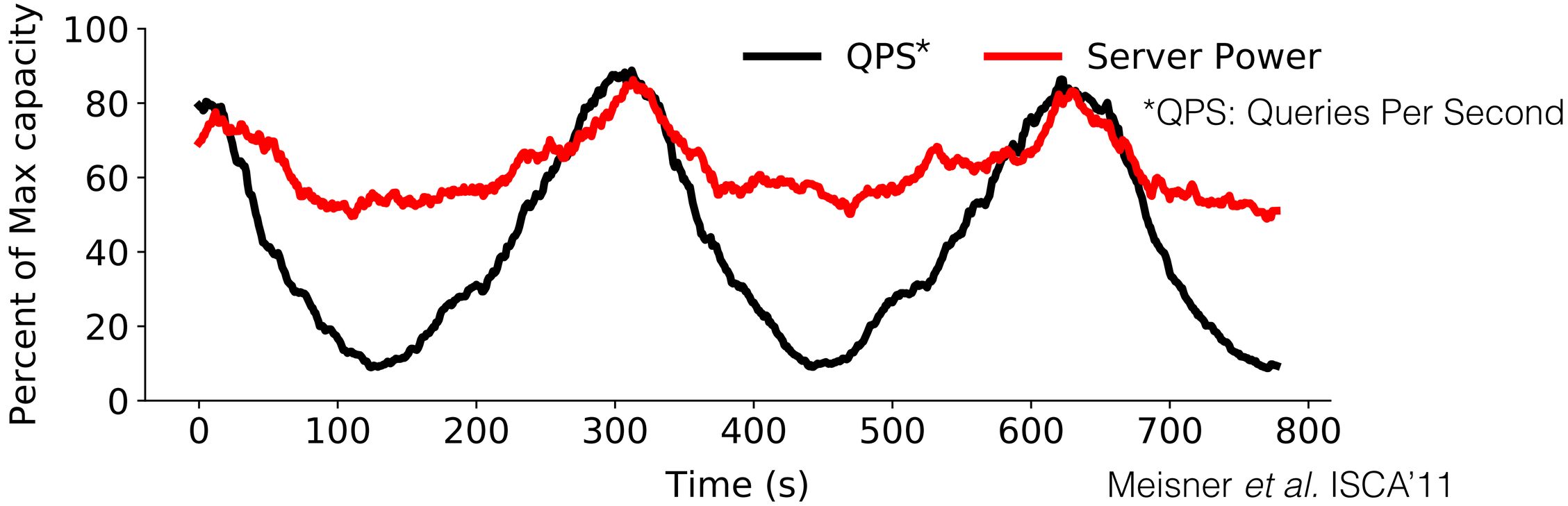
Server

Power consumption breakdown

CPU and **memory** are the most power consuming parts

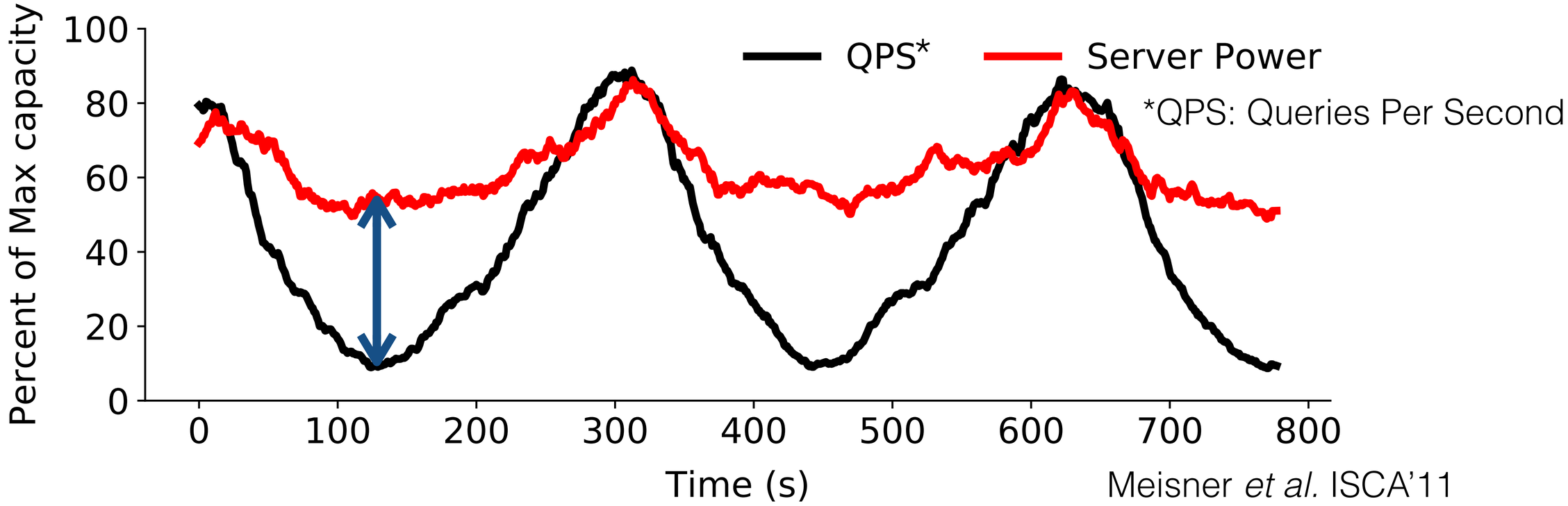
Lefurgy *et al.* computer'03, Kultursay *et al.* ISPASS'13

Typical Cloud Workload



Web-Search running on 2 big cores of ARM Juno R1 (big.LITTLE architecture)

Typical Cloud Workload



Web-Search running on 2 big cores of ARM Juno R1 (big.LITTLE architecture)

System power consumption is *not* proportional to utilisation

Challenges

Challenges

- Shared resource contention
- Data centre particularities
- Transition in data centres
- Schedule workloads

Obstacles

Cache, memory, network, etc

Different requirements!

Transition to hetero. arch.

Traditional scheduling mechanisms?

Applications have different requirements

- Latency-critical workloads
  - Examples: Memcached (Nishtala *et al.* USENIX'13), Web-Search (Kasture *et al.* MICRO'15), ML_Cluster (Lo *et al.* ISCA'16), etc.
  - Important metric: *Tail latency* (Suresh *et al.* NSDI'15)
- Batch workloads
  - Examples: Neural networks (Schmidhuber Technical report'14), numerical simulations, etc.
  - Important metric: *Mean throughput*

Talk Organisation

- Motivation
- THESIS CONTRIBUTIONS
- REPP: Runtime Estimation of Performance and Power
- Hipster: Hybrid Task Manager for Latency-Critical Cloud Workloads
- Future work & Contributions recap

Contributions of the Thesis

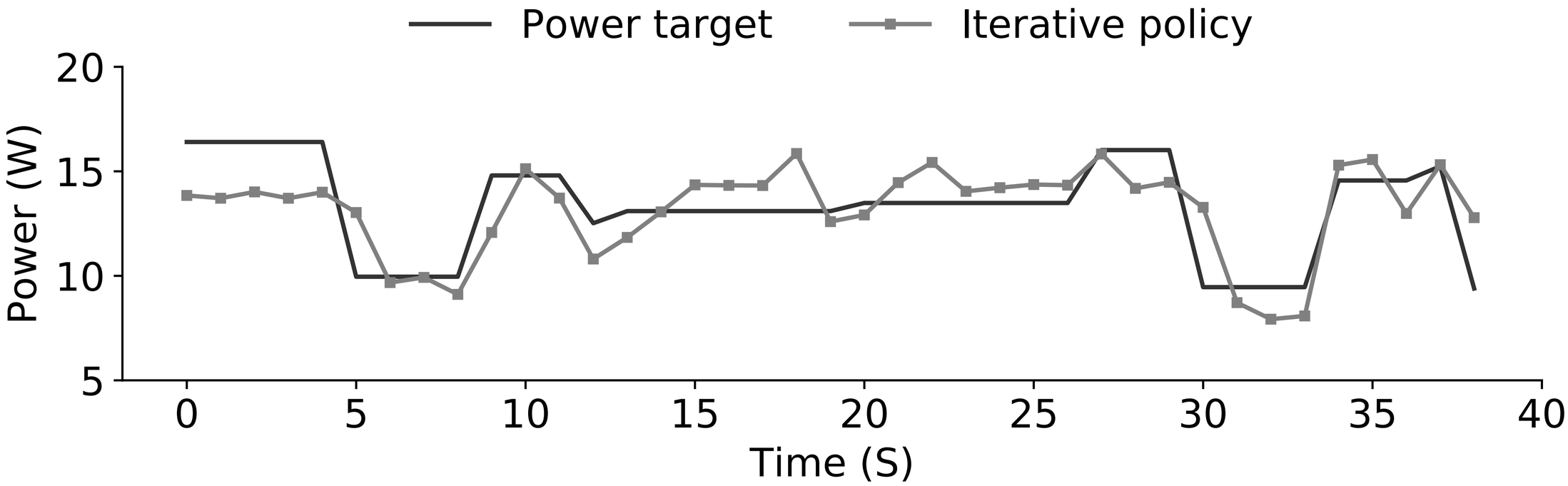
- Fast responsiveness to external constraints
- Improve energy proportionality
- Improve resource utilisation

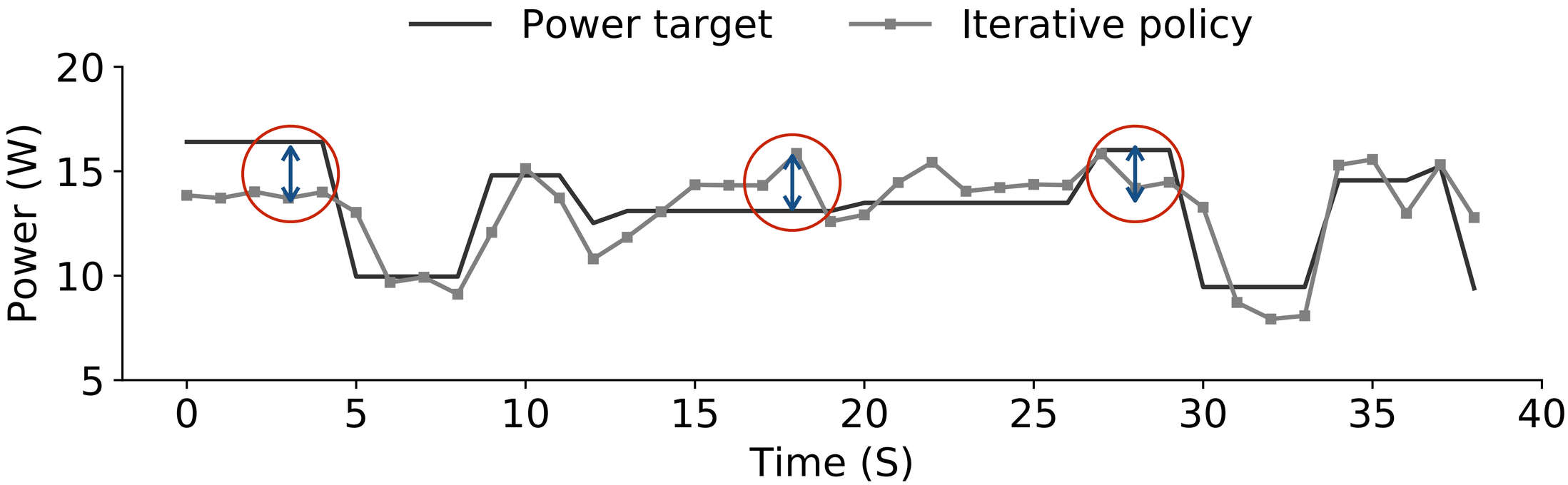
Challenges in Meeting External Constraints

- Performance/Energy efficiency are a *major* concern in data centres
- Server architecture heterogeneity is already present(!)
  (Mars *et al.* ISCA'13, Mars *et al.* CAL'11, Nathuji *et al.* ICAC'07)
- And some challenges arise
  - How to allocate heterogeneous apps in a dynamic server farm?
  - How to manage system performance and power?
- With challenge comes opportunity!

Fast Responsiveness to External Constraints

- Strict service level requirements for power and performance
  - Performance guarantees, power capping, and energy efficiency (Cochran *et al.* MICRO'11, Miftakhutdinov *et al.* MICRO'12)
- Two main variants:
  - Iterative approaches — (e.g., Intel RAPL (Rountree *et al.* IGSC'13))
    - May require iterations, lower accuracy, higher SLA violation
  - Reactive approaches — (e.g., PEPP (Su *et al.* MICRO'14))
    - Single step, higher accuracy, smaller SLA violation
    - Architecture specific counters (Miftakhutdinov *et al.* MICRO'12)

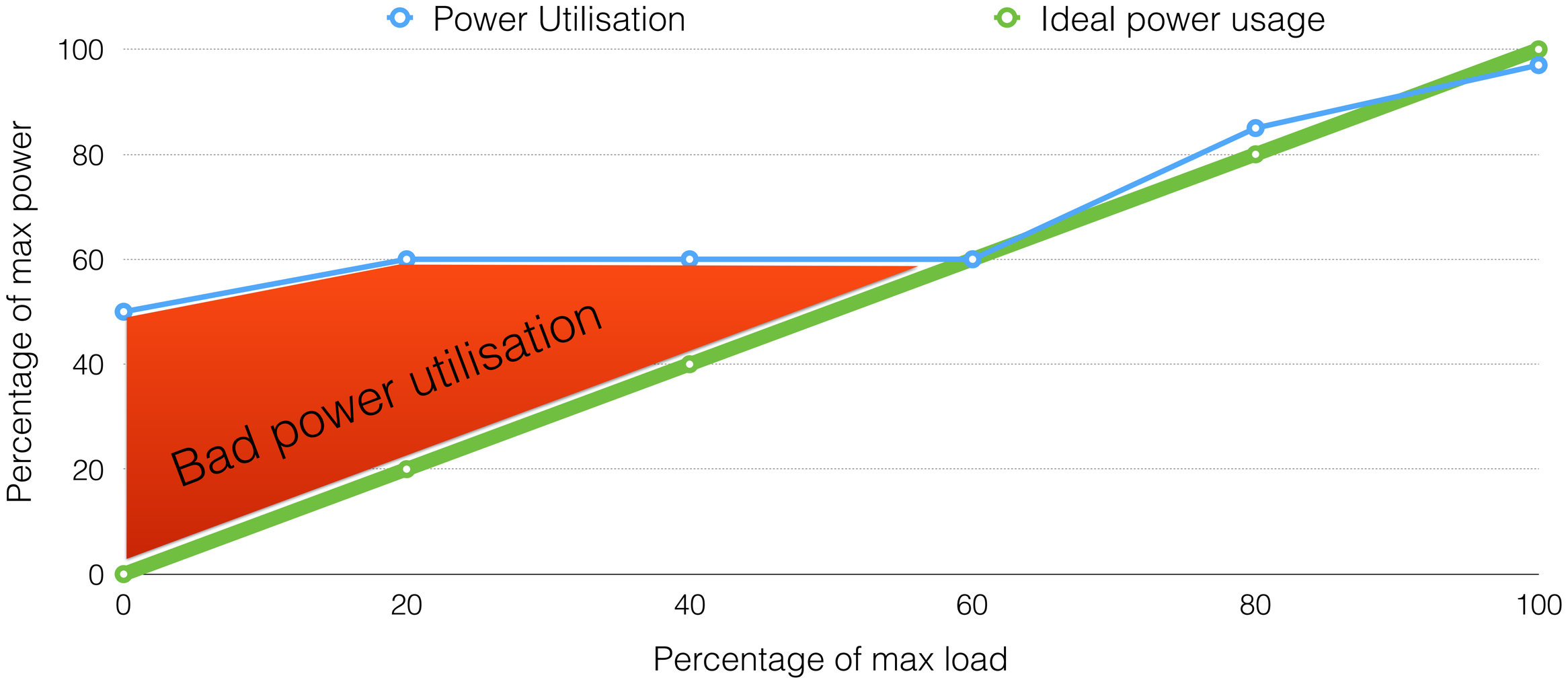

Power target
Iterative policy
Power (W)
20
15
10
5
0
5
10
15
20
25
30
35
40
Time (S)



Bad power capping mechanism

Adapts slowly and violates power SLA (Service Level Agreements)

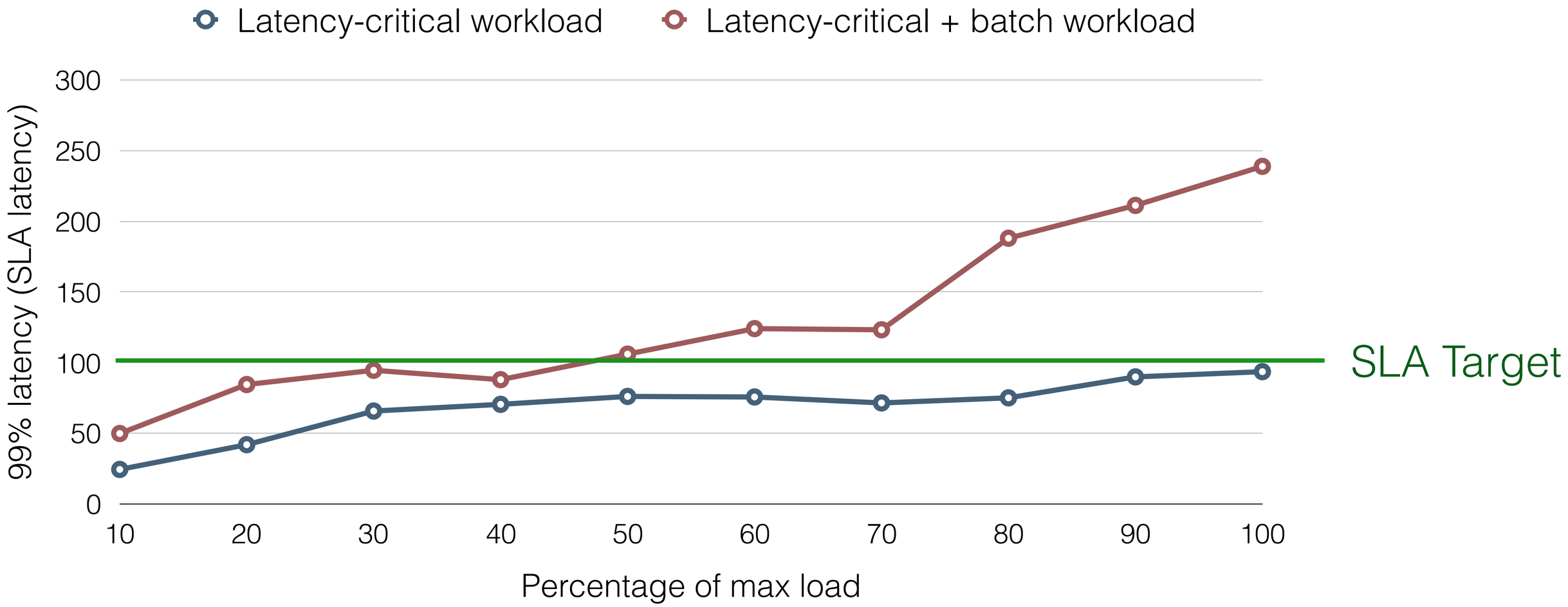
Improve Energy Proportionality



Improve Resource Utilisation

- **Idea**: Collocate latency-critical workloads with batch workloads in low utilisation periods

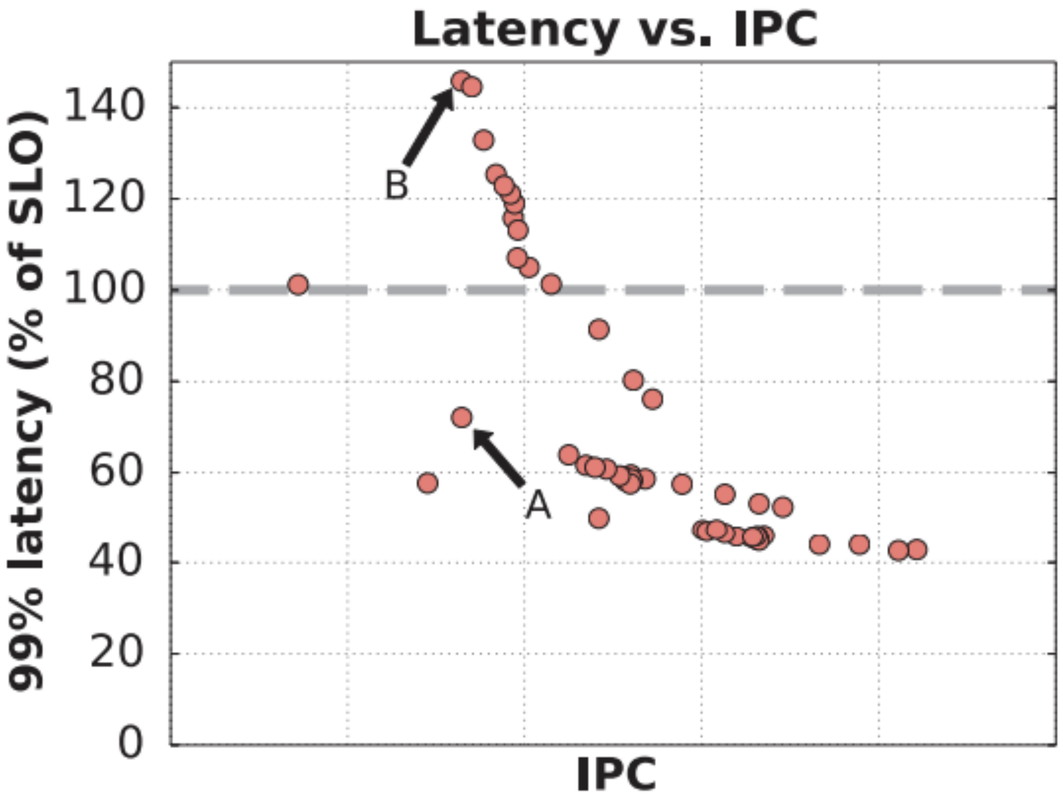
- Unfortunately, latency-critical workloads like to be alone
  - Our old friend, shared resource contention is back!
    (Lo *et al.* ISCA'16, Petrucci *et al.* HPCA'15, Kasture *et al.* MICRO'15)
  - Violate latency



Interactive workloads violate latency after 45% load when collocated

Traditional Scheduling Mechanisms

- Web-Search on Intel Xeon E5
- Traditional scheduling mechanisms
  - IPC/IPS based
  - Interactive workloads violate latency



Reproduced from Lo *et al.* TOCS'16

No correlation between IPC/IPS and Latency

Potential Solutions

- Data centre requirements suggest:
  1. Fast responsive to external constraints
  2. Improved energy proportionality
  3. Improved resource utilisation

- Approach the 1st requirement
  - **REPP**: An accurate power and performance estimation approach
- Solve the 2nd and 3rd requirements, in one shot
  - **Hipster**: Improves energy proportionality and resource utilisation

Talk Organisation

- Motivation
- Thesis contributions
- REPP: RUNTIME ESTIMATION OF PERFORMANCE AND POWER
- Hipster: Hybrid Task Manager for Latency-Critical Cloud Workloads
- Future work & Contribution recap

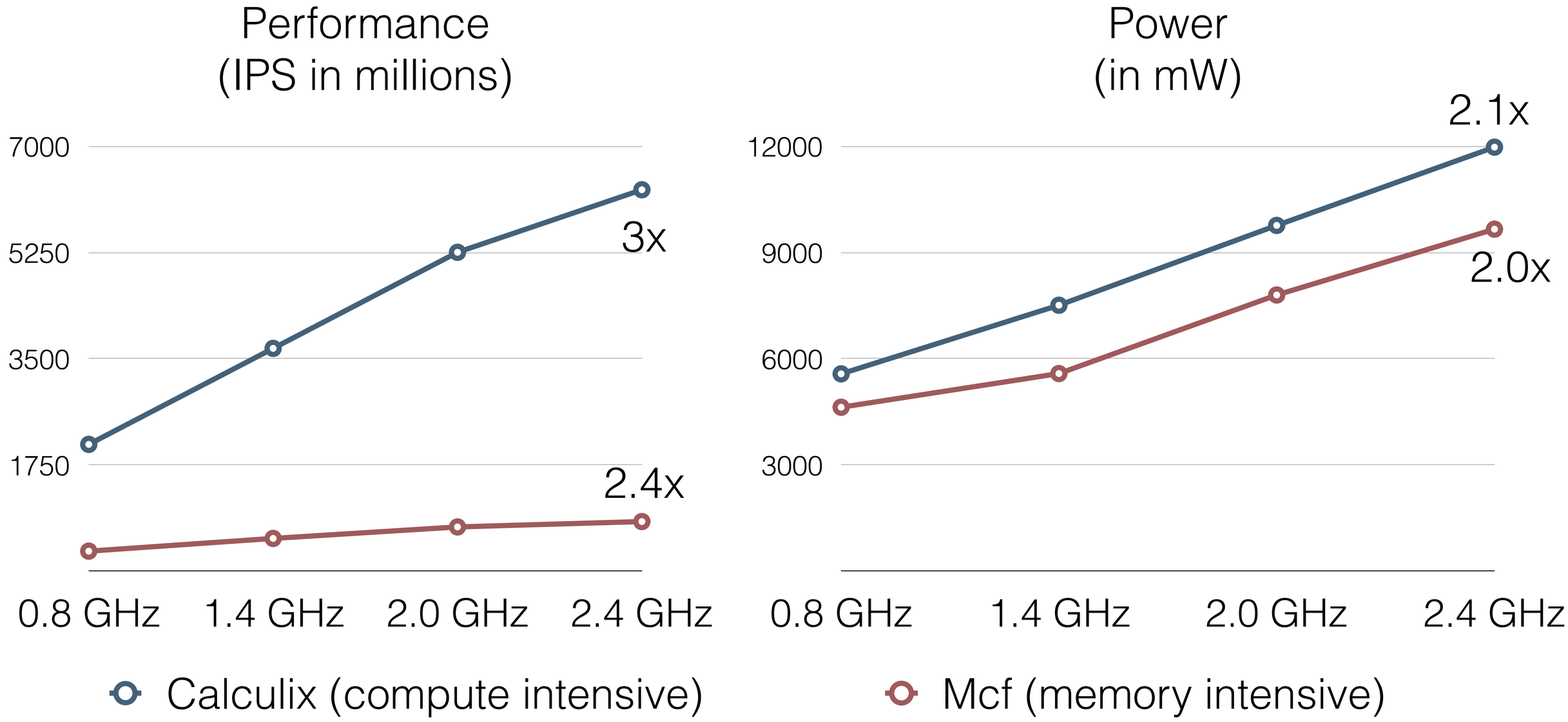
REPP

Runtime Estimation of Power and Performance

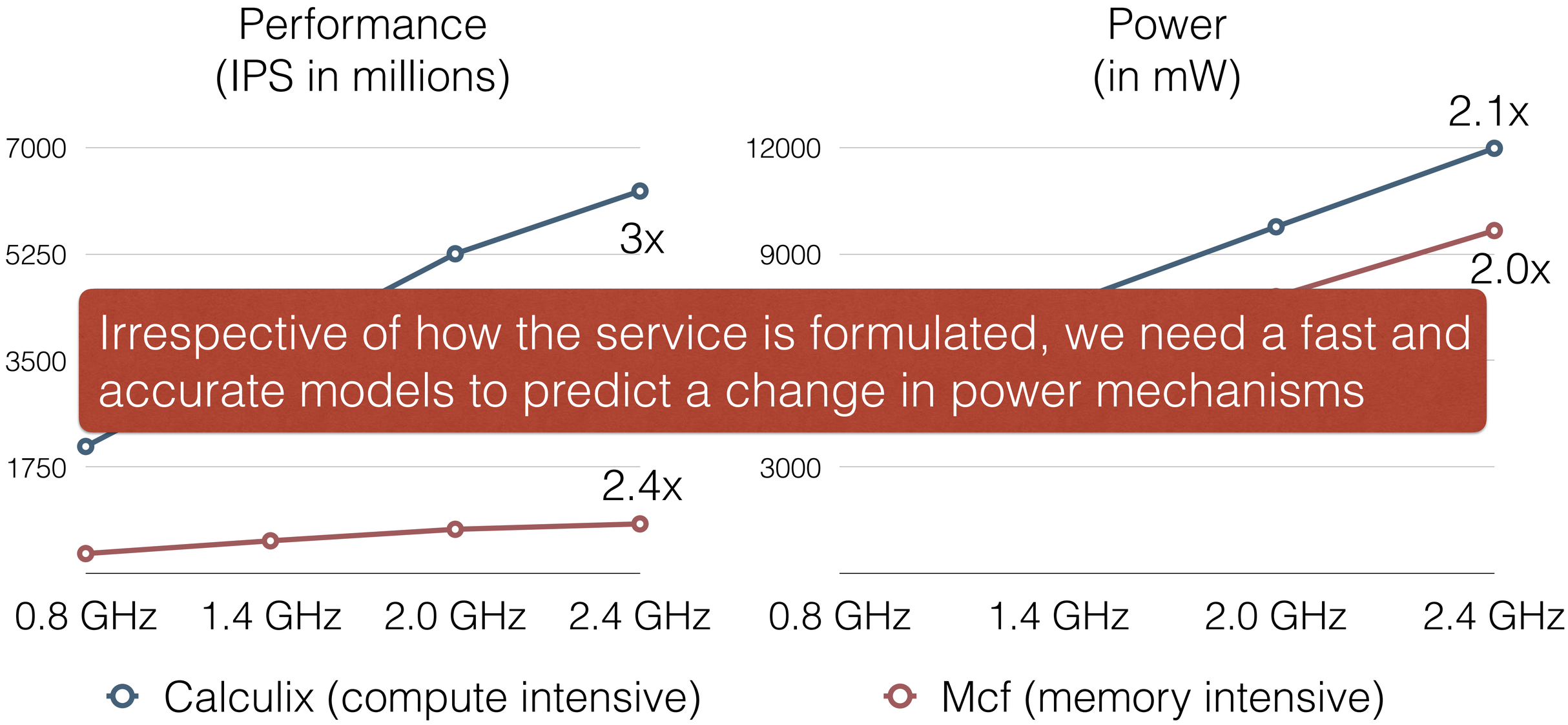
R. Nishtala *et al.*
1. "*A Methodology to Build and Predict Performance and Power in CMPs*",
International Conference on Parallel Processing (ICPPW) 2015
2. "*REPP-H: Runtime Estimation of Performance and Power for Heterogeneous Data Centers*",
International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) 2016
3. "*REPP-C: Runtime Estimation of Performance and Power with Workload Consolidation*",
International Green and Sustainable Computing (IGSC) 2016

Power and Performance Management

- **Requirement**: Meet power and performance target
  - Power management mechanisms
    - DVFS/P-States: Performance states
    - CI-States (Intel_powerclamp): Forced deep sleep state residency in a core
  - Hardware performance monitoring counters (PMCs)



How does performance and power change with P-State?



How does performance and power change with P-State?

REPP in a nutshell

REPP in a nutshell

- Accurate prediction model across P-States and CI-States

REPP in a nutshell

- Accurate prediction model across P-States and CI-States
- Uses basic counters available across architectures
  - Gather counters with high dynamic power consumption

REPP in a nutshell

- Accurate prediction model across P-States and CI-States
- Uses basic counters available across architectures
  - Gather counters with high dynamic power consumption
- Meets performance and power targets in multicore

REPP in a nutshell

- Accurate prediction model across P-States and CI-States
- Uses basic counters available across architectures
  - Gather counters with high dynamic power consumption
- Meets performance and power targets in multicore

- Single core model for each architecture
  - Power is measured in Watts (W)
  - Performance is measured in Instructions per Second (IPS)
  - Model coefficients are blind to number of apps

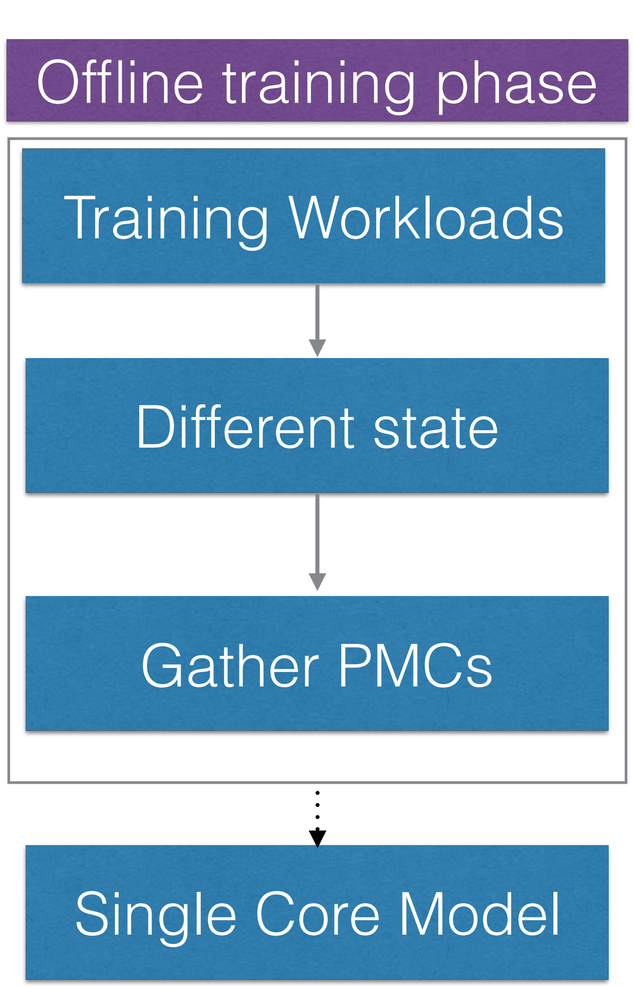
REPP's Methodology

Offline training phase

Online single core validation

Multicore modelling

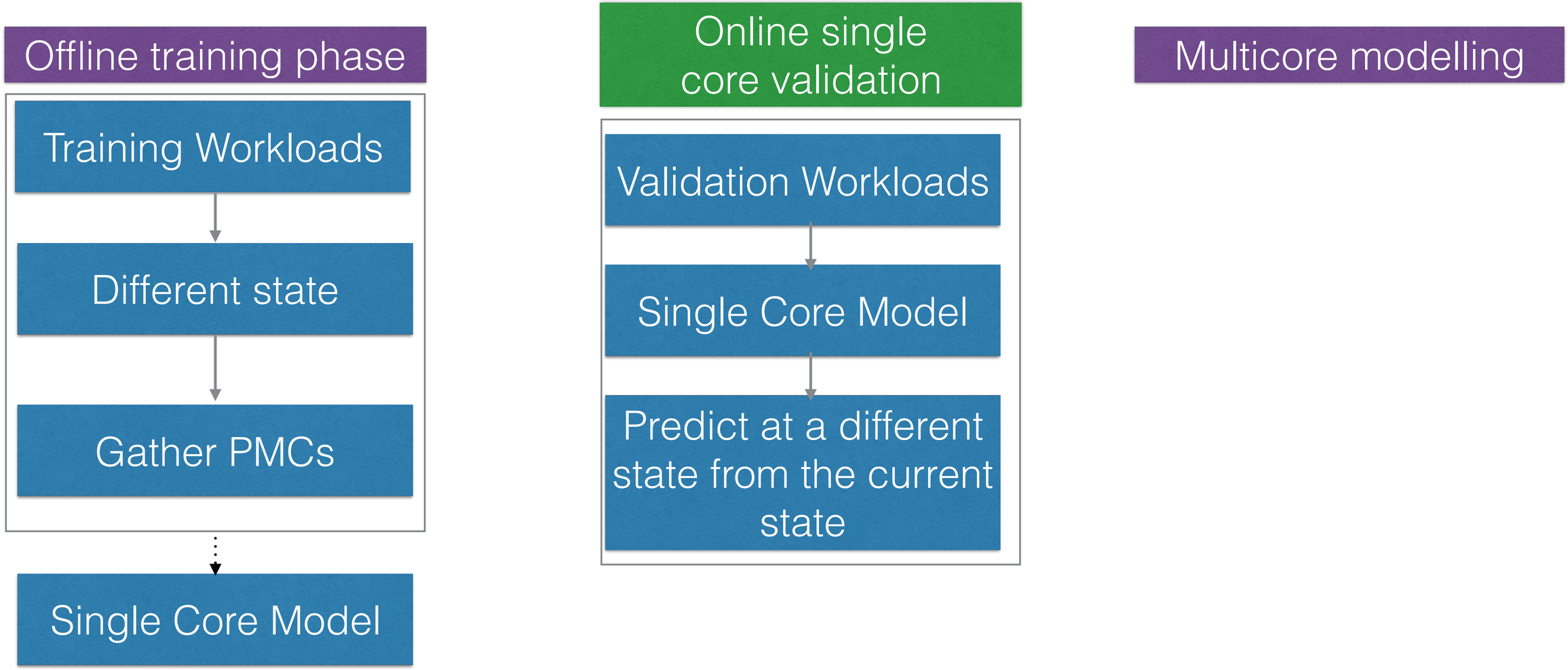
REPP's Methodology



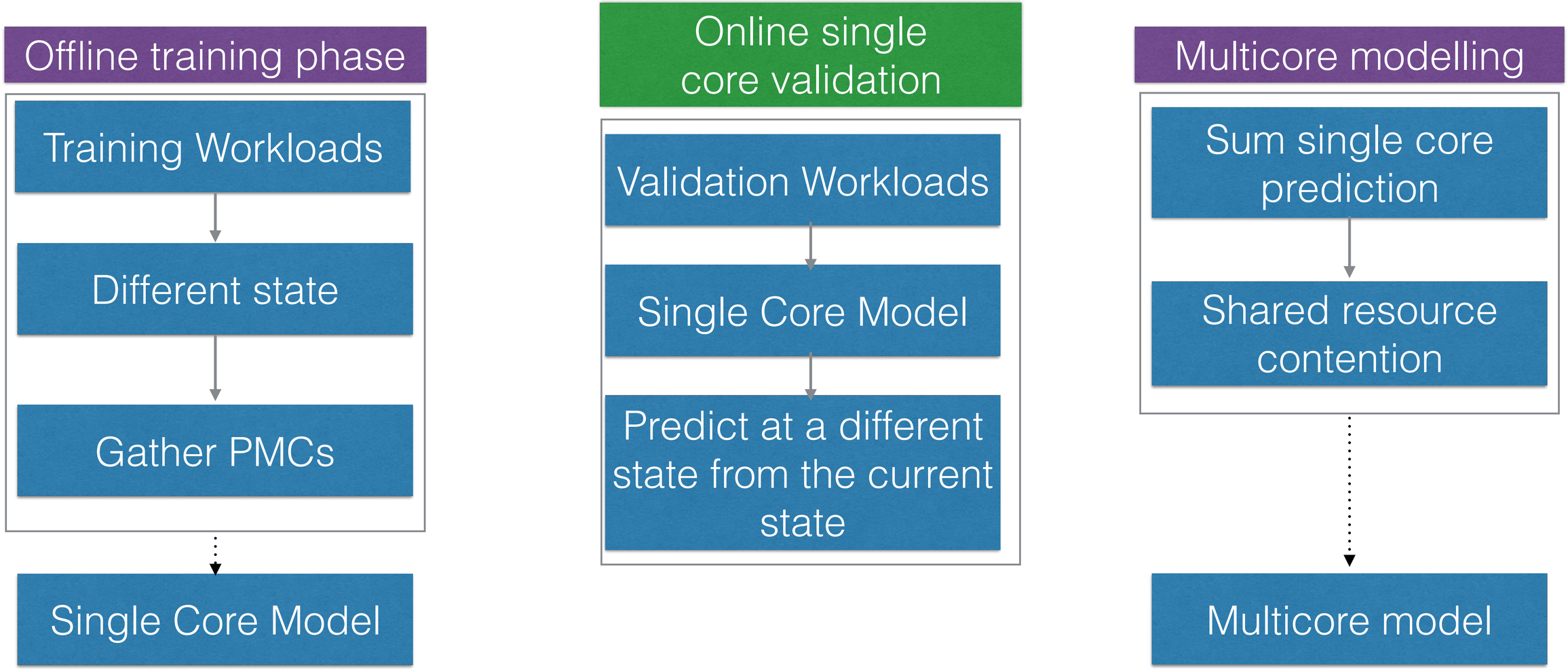
Online single core validation

Multicore modelling

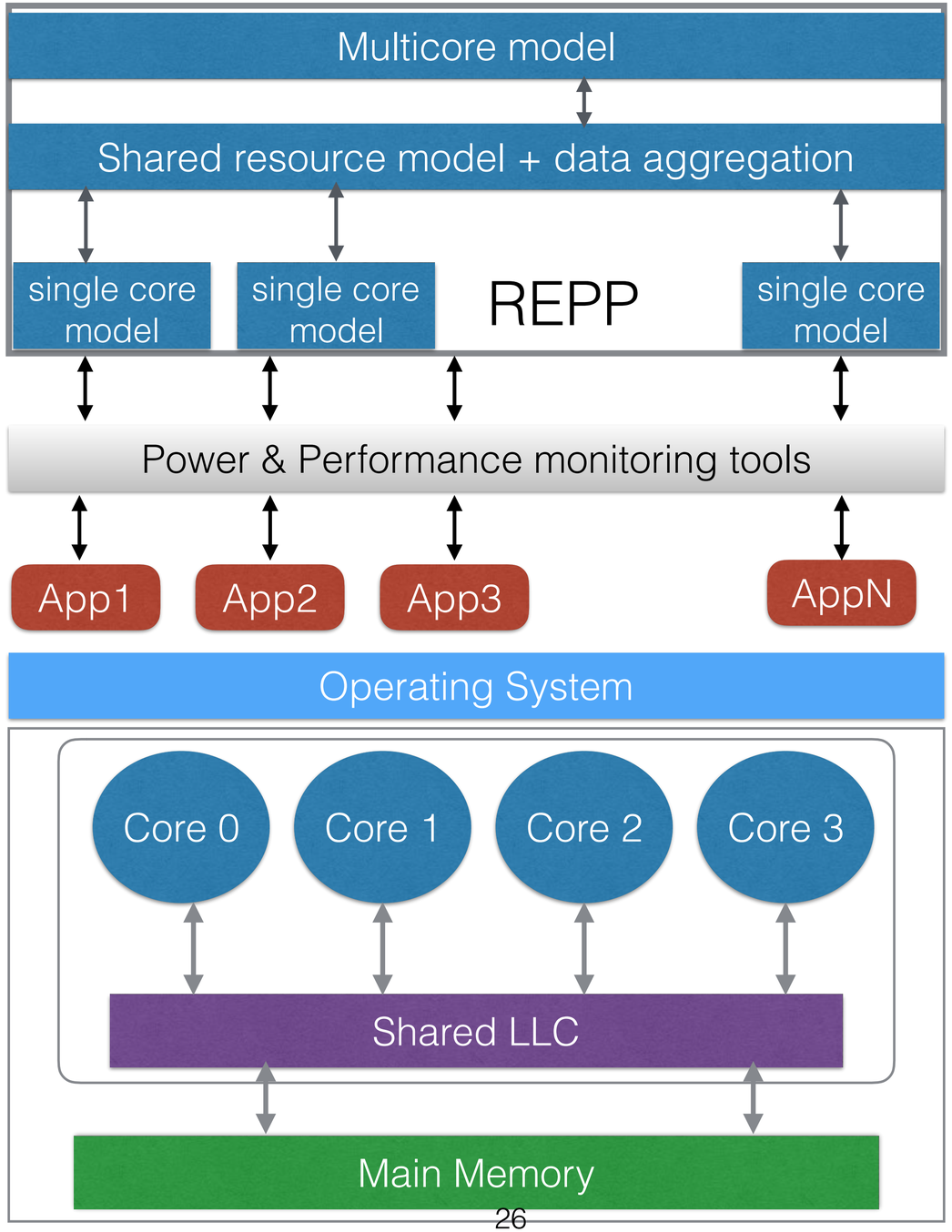
REPP's Methodology



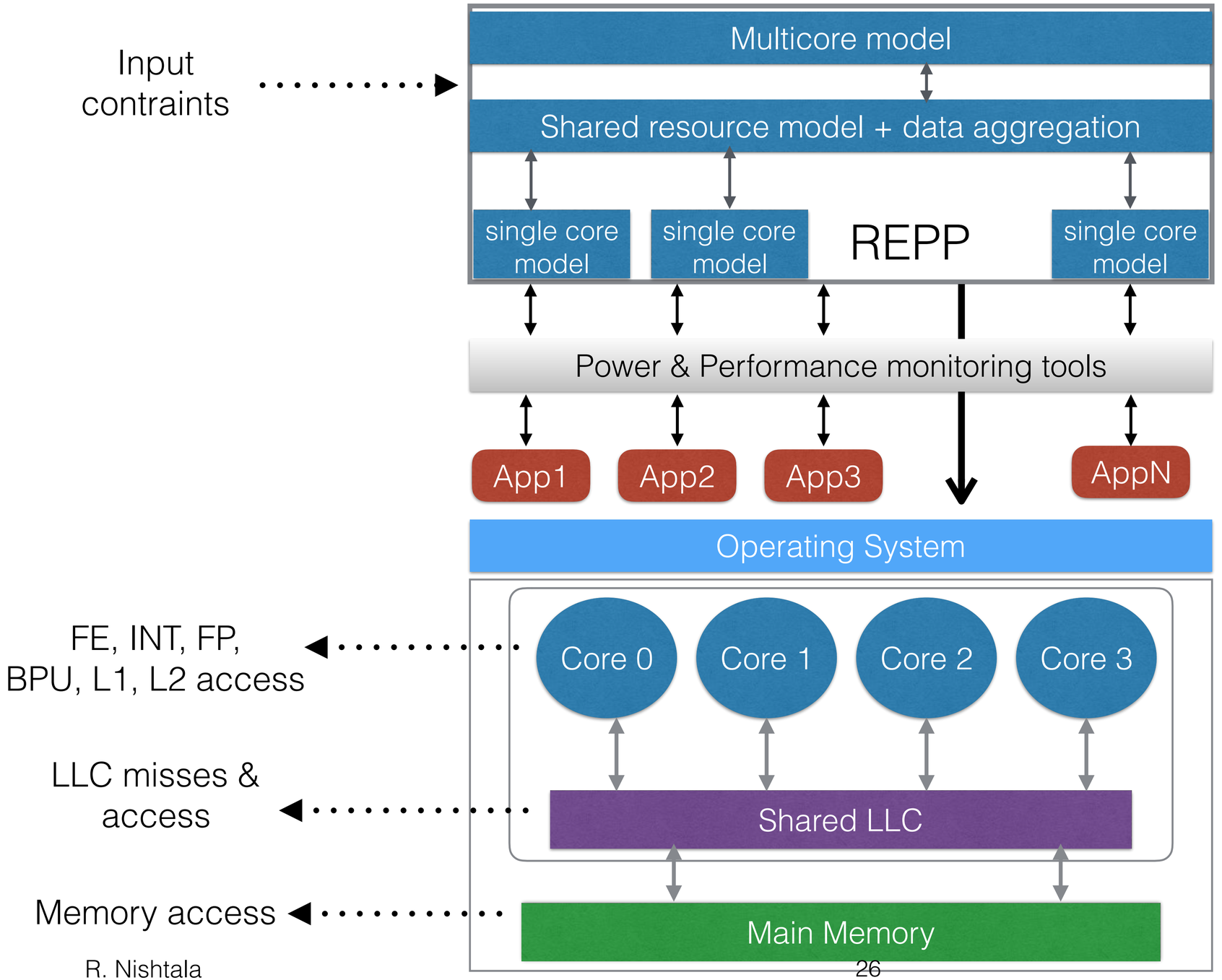
REPP's Methodology



REPP's Approach

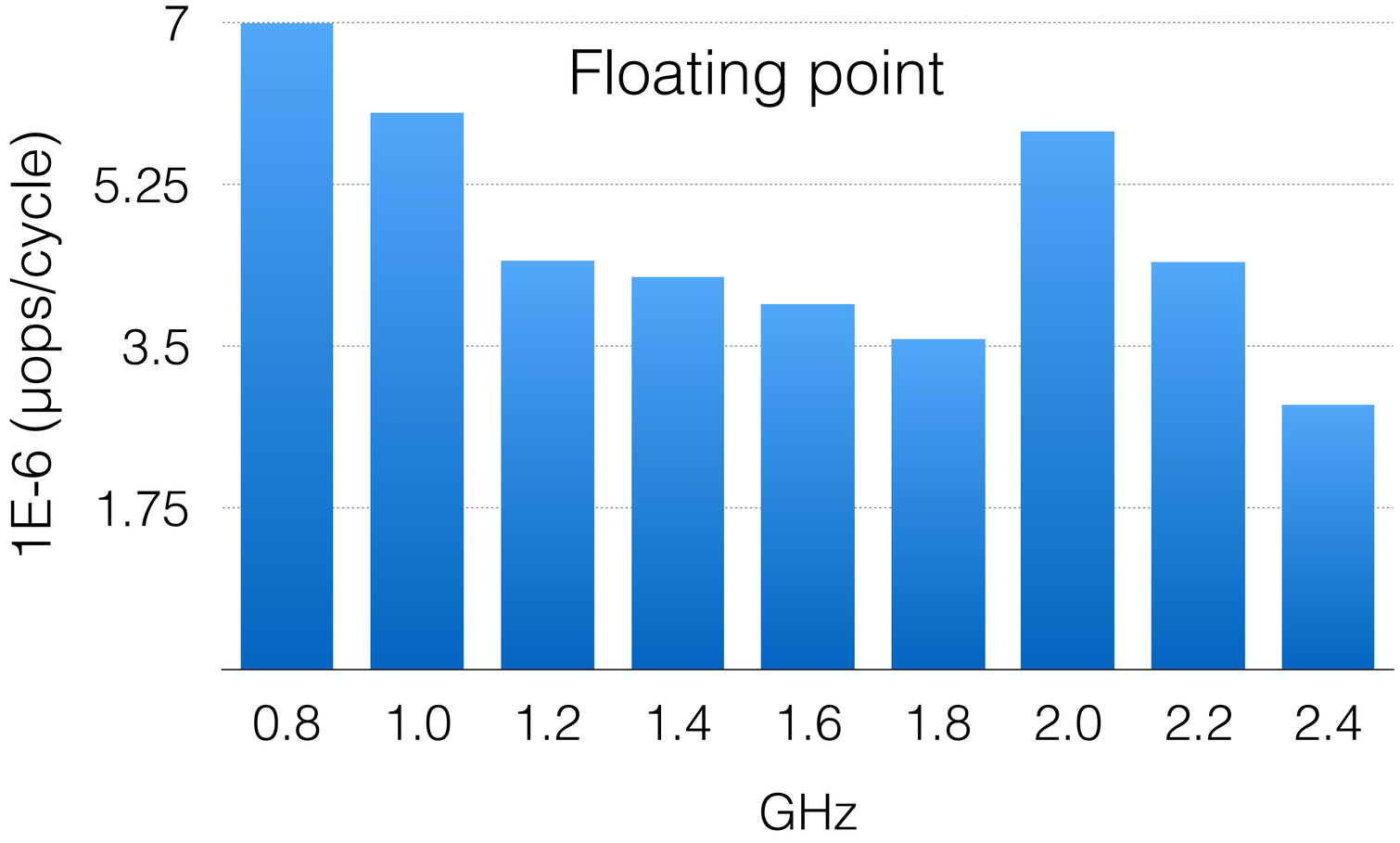


REPP's Approach



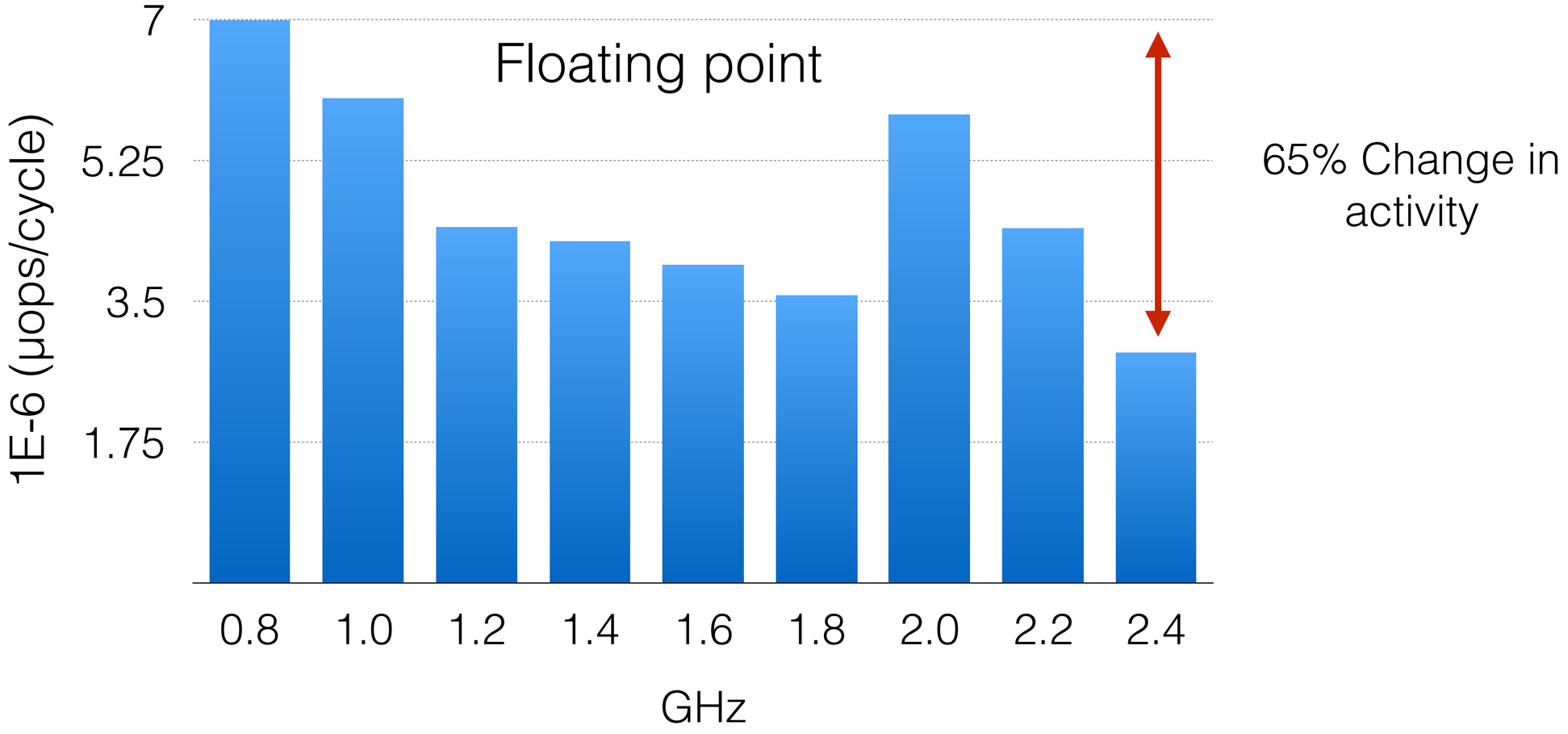
FE - Front End
INT - Integer
FP - Floating Point
BPU - Branch Predictor Unit
L1 - L1 cache
L2 - L2 cache
LLC - Last Level Cache

Components Activity for LBM* (SPEC)



* Memory intensive — floating point — benchmark

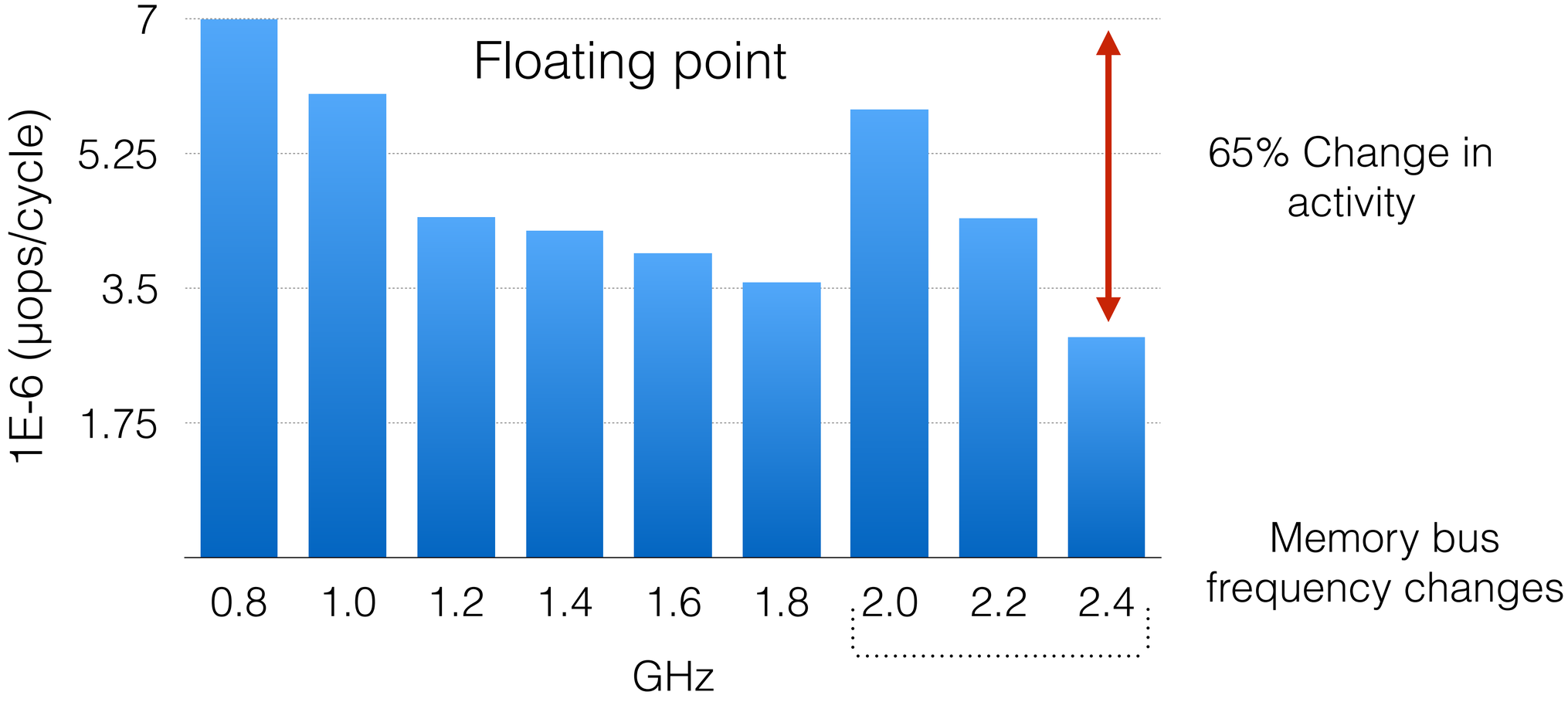
Components Activity for LBM* (SPEC)



Massive relative change in component activity, but absolute change is negligible

* Memory intensive — floating point — benchmark

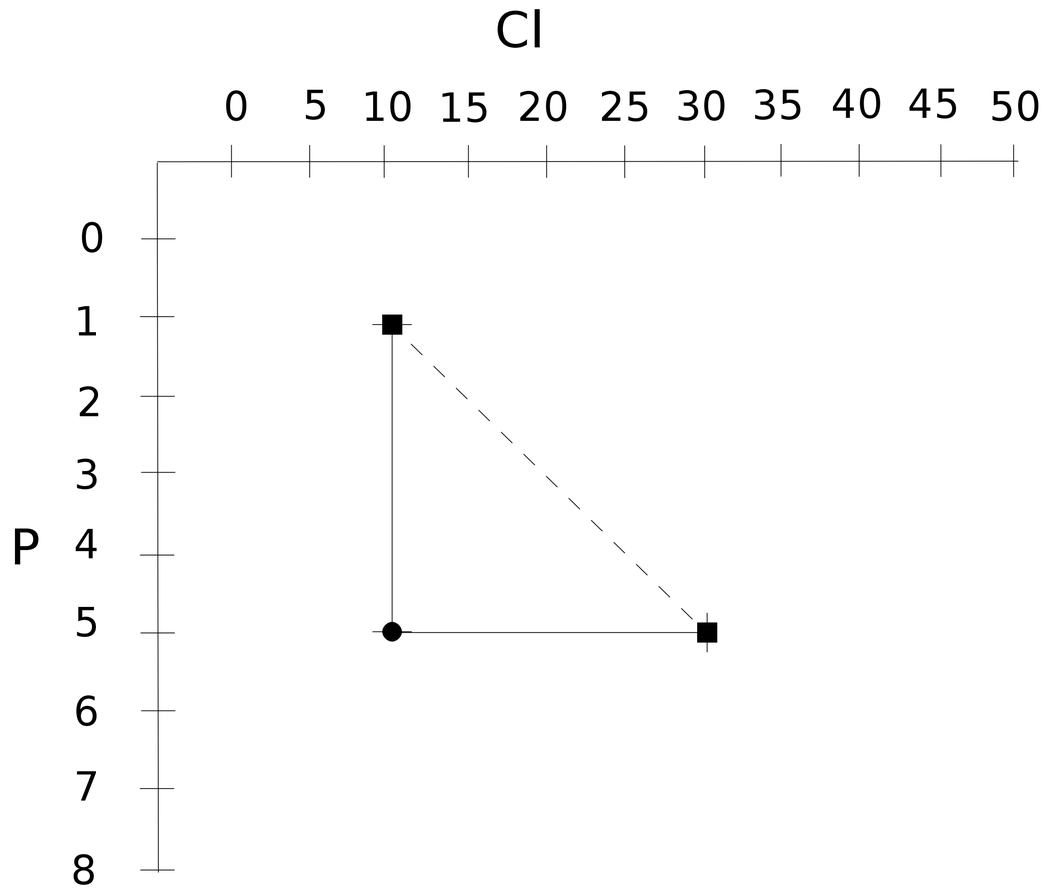
Components Activity for LBM* (SPEC)



Massive relative change in component activity, but absolute change is negligible

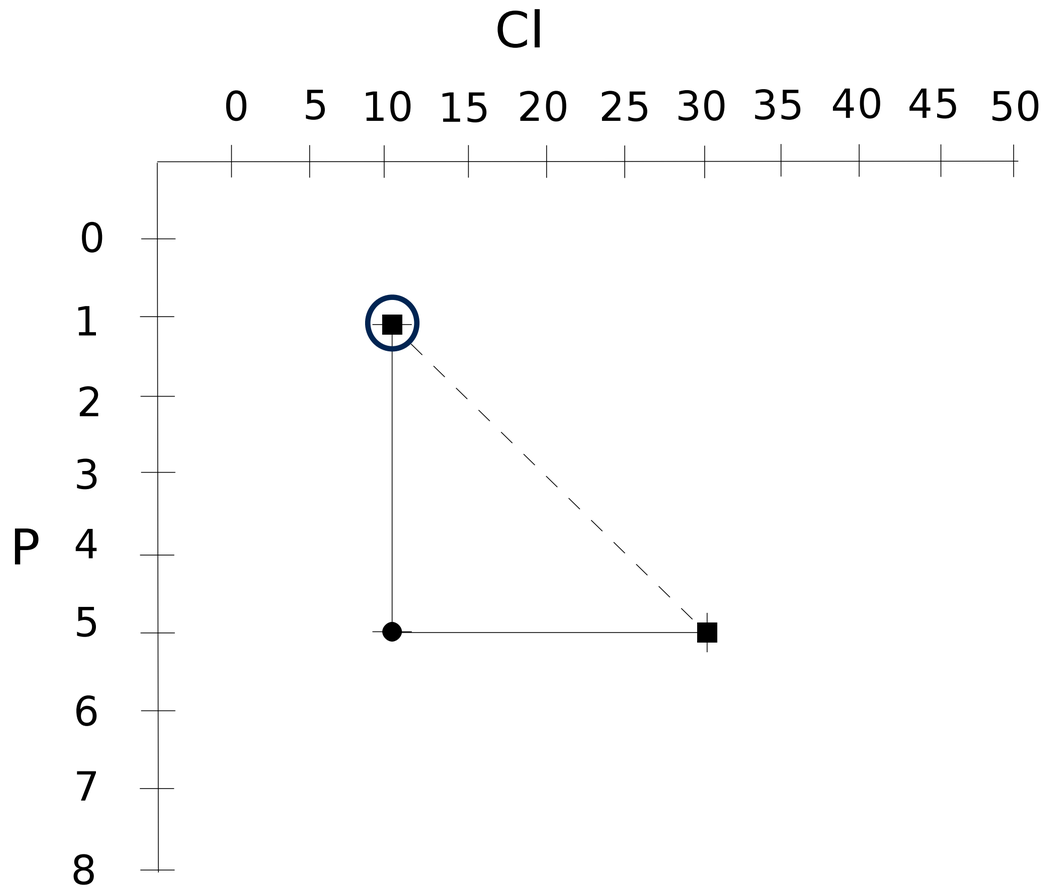
* Memory intensive — floating point — benchmark

Prediction Methodology



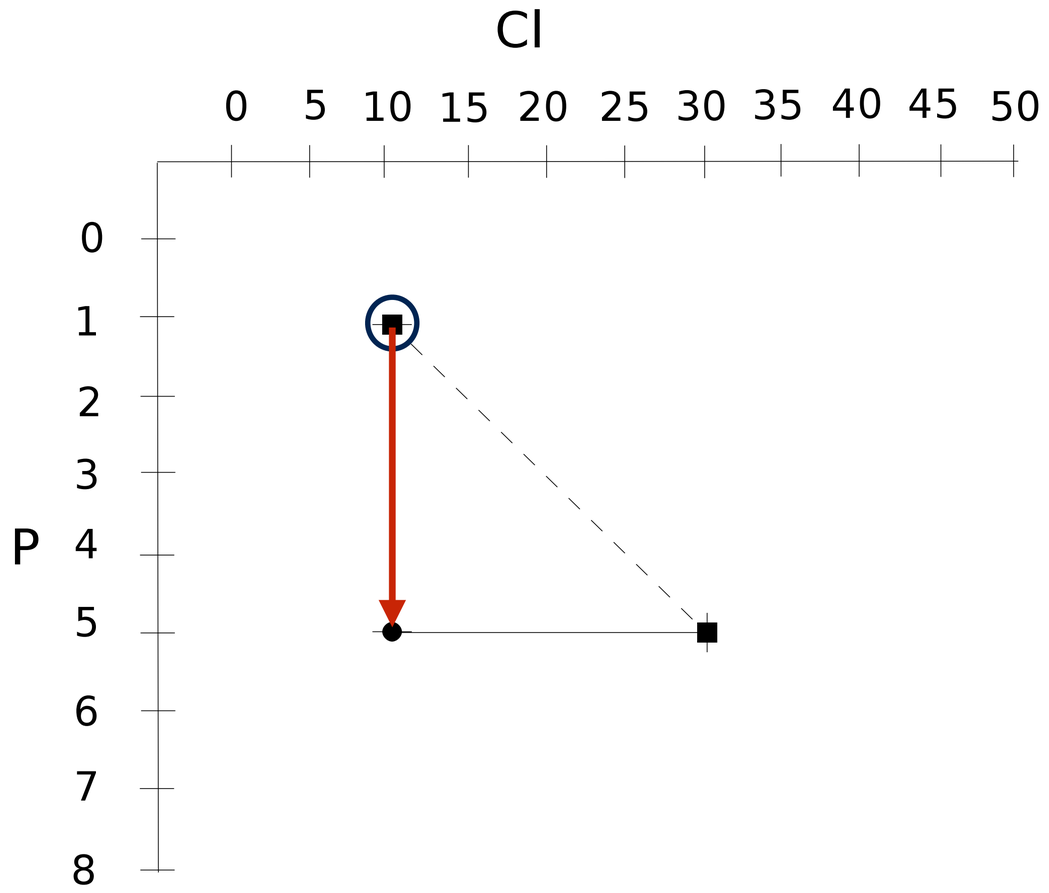
Prediction Methodology

- Predict power and performance at current configuration



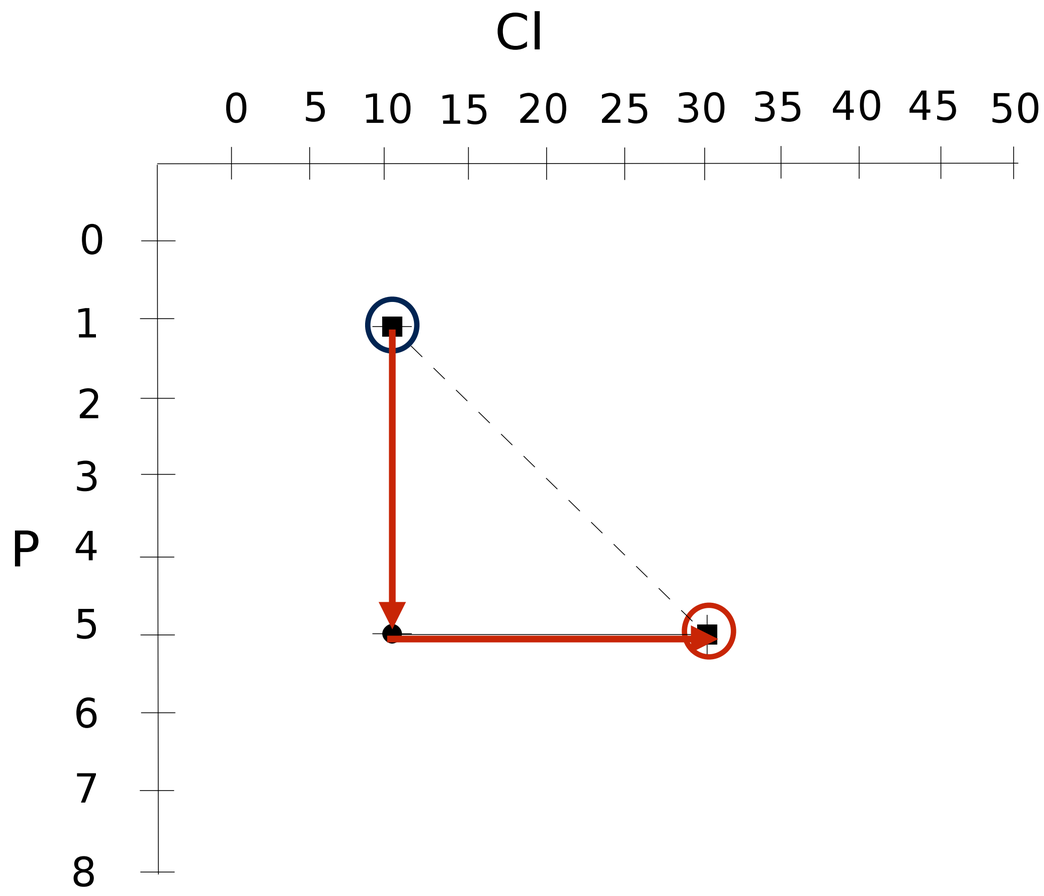
Prediction Methodology

- Predict power and performance at current configuration
- Fixed CI-State, predict across P-States



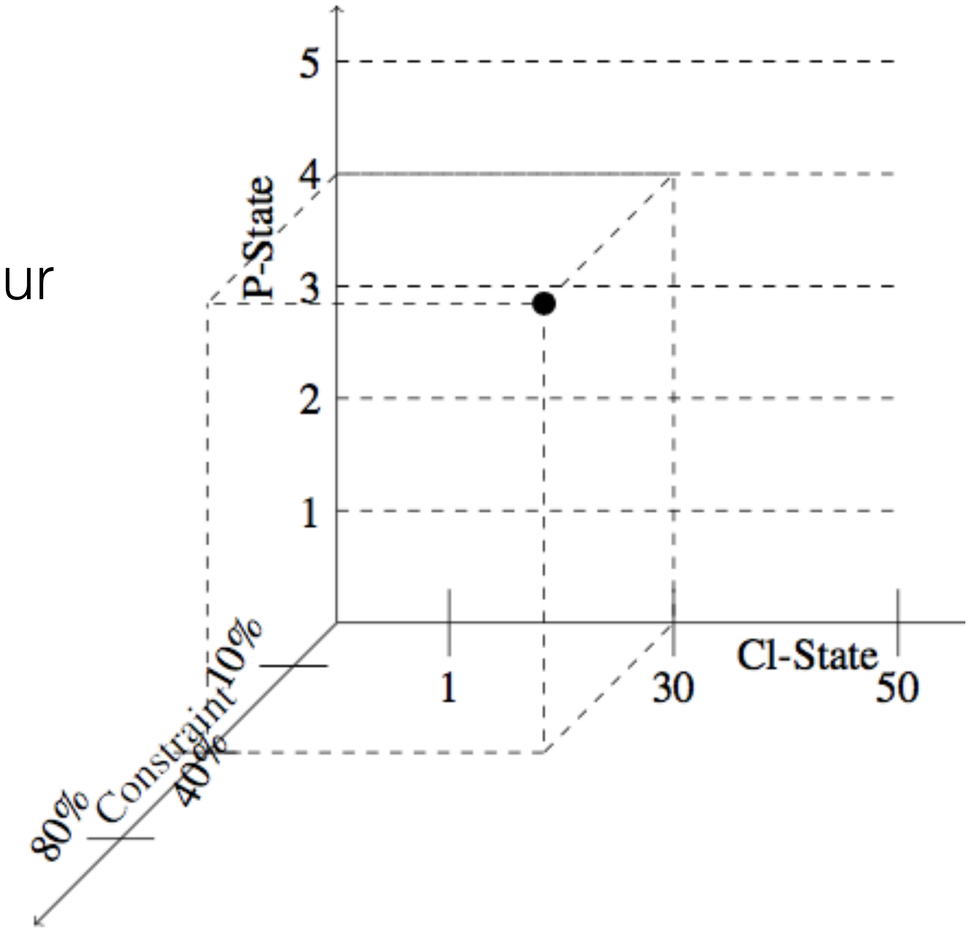
Prediction Methodology

- Predict power and performance at current configuration
- Fixed CI-State, predict across P-States
- Fixed P-State, predict across CI-State



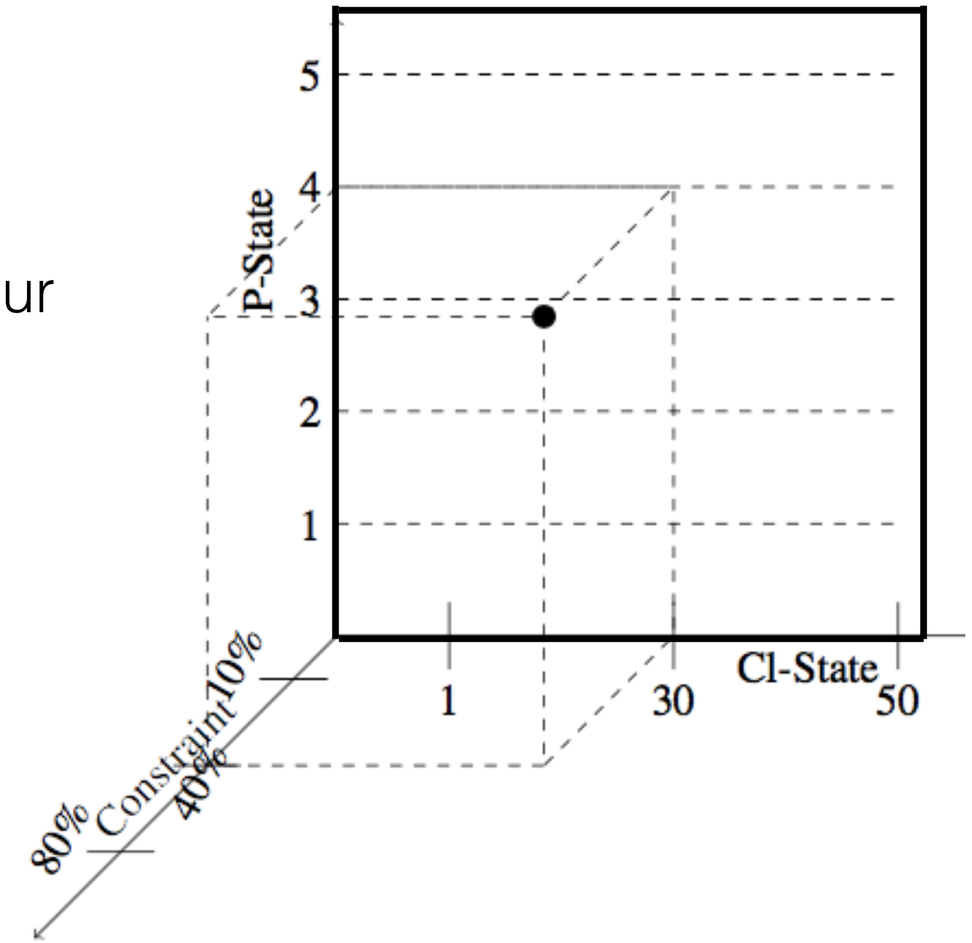
REPP Configuration Selector

- Repeat this process every 250 ms
  - Most SPEC workloads exhibit this behaviour



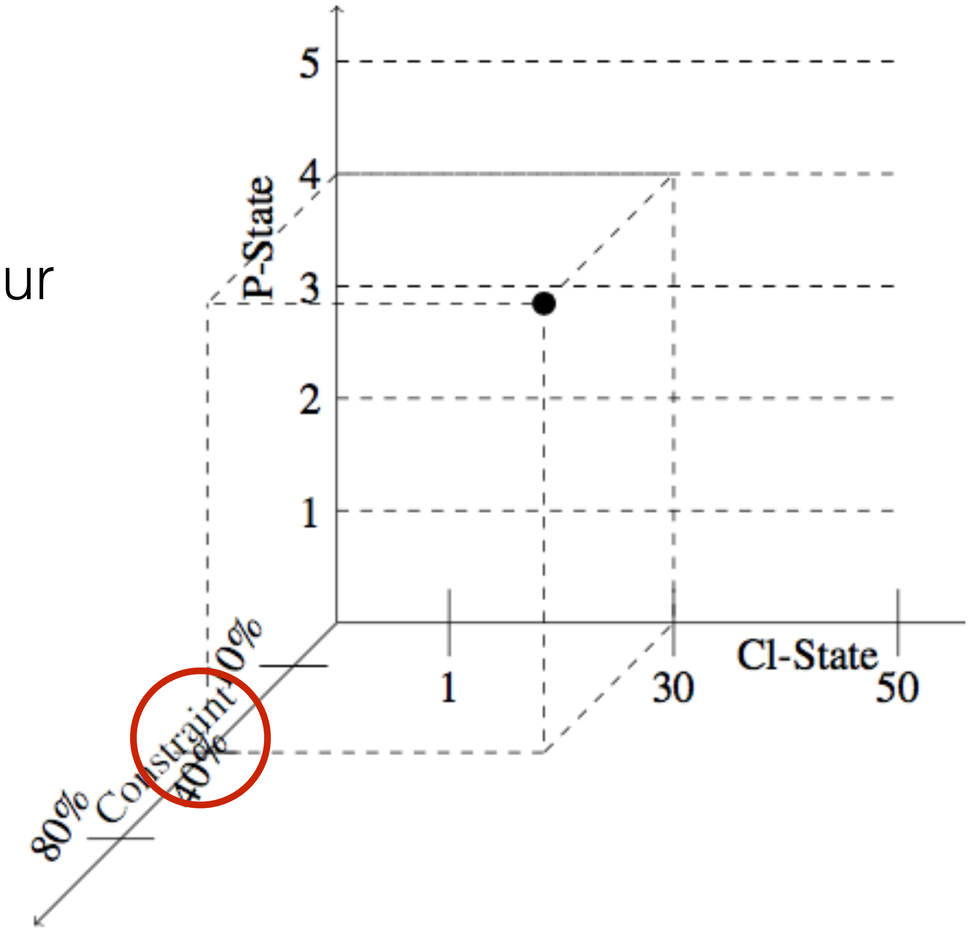
REPP Configuration Selector

- Repeat this process every 250 ms
  - Most SPEC workloads exhibit this behaviour



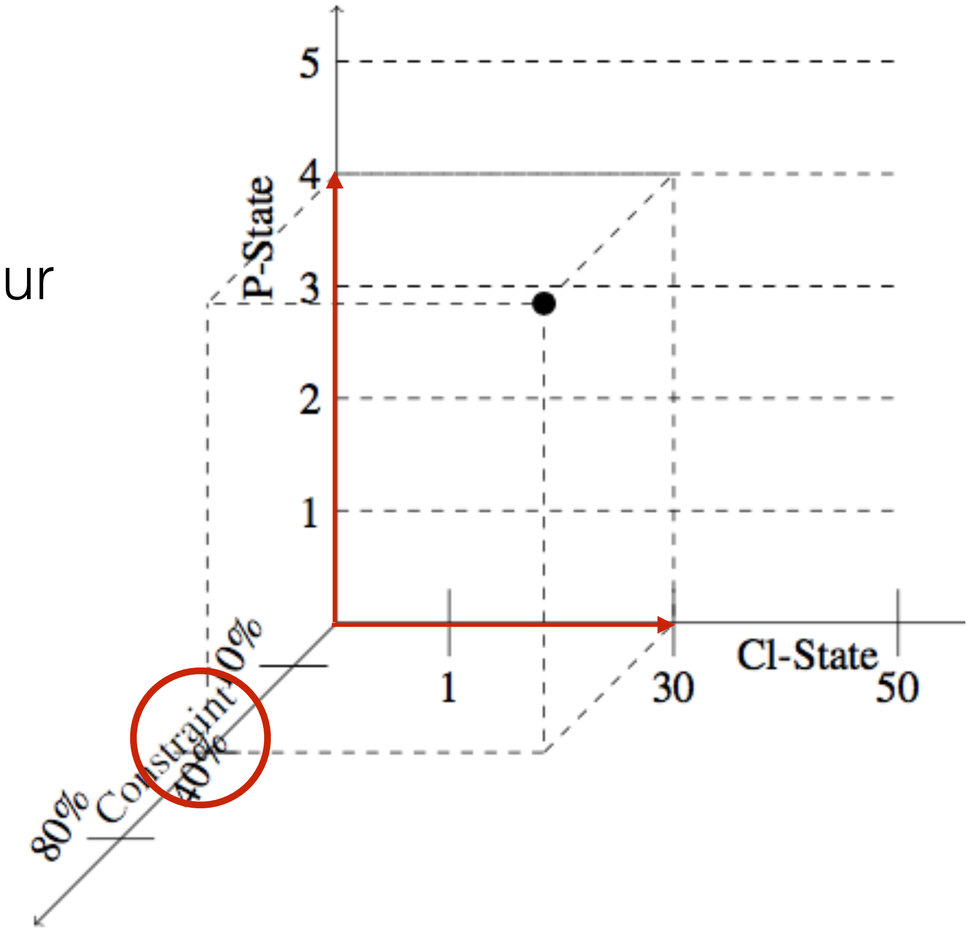
REPP Configuration Selector

- Repeat this process every 250 ms
  - Most SPEC workloads exhibit this behaviour



REPP Configuration Selector

- Repeat this process every 250 ms
  - Most SPEC workloads exhibit this behaviour



Architecture Specific Details

	Intel SandyBridge i7	AMD Opteron II	ARM Juno R1 (big.LITTLE)
Linux kernel	3.14.5	3.13.0	4.3.0
Hardware details	4 cores, 256kB (L2), 6MB (L3)	4 cores, 512kB (L2), 6MB (L3)	2 cortex A57 - 2MB (L2) 4 cortex A53 - 1MB (L2) No L3
Control knobs	P-States CI-States	P-States	P-States
Power measurement	Intel RAPL (DRAM+CPU+uncore)	WattsUP Pro (Socket power)	Power Registers (big+LITTLE+SoC)
Performance measurement	Perf (Linux)	Perf (Linux)	Perf (Linux)

Hyperthreading is disabled

REPP Evaluation

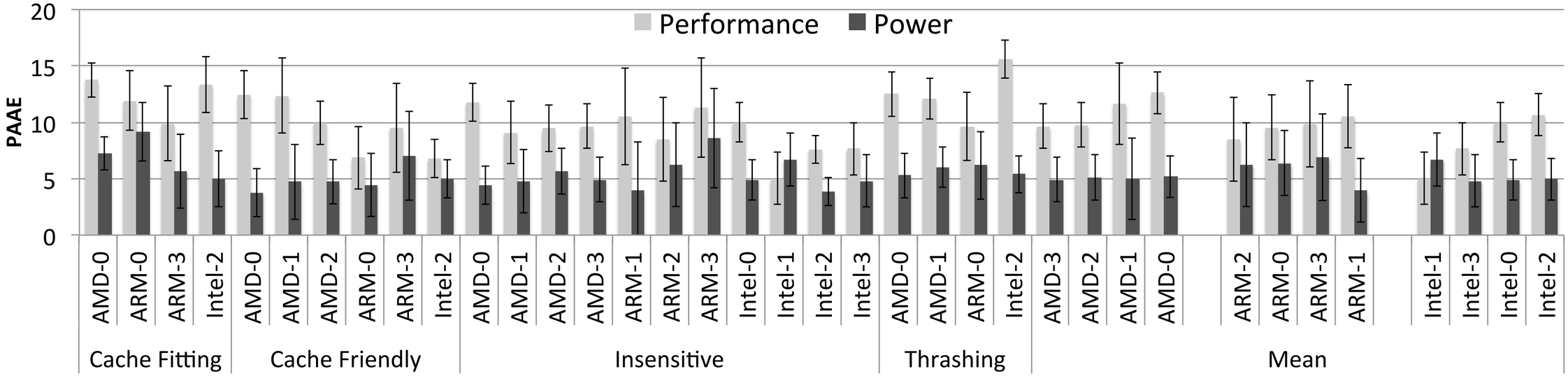
- Benchmark suites
  (Sequential workloads)
  - SPEC CPU2006
  - PARSEC 3.0
  - SPLASH-2x
  - NAS
- Categorise benchmarks
  (Sanchez *et al.* ISCA'11)
  - Insensitive (**N**)
  - Cache-Friendly (**F**)
  - Cache-Fitting (**T**)
  - Thrashing/Streaming (**S**)
- # apps = # cores
- Apps stats collected from *perf* per thread session every 250 ms

Model Assessment

- **Two metrics** to validate the models
  - **PAAE**: Percentage Absolute Average Error
    - Error between predicted values from actual values
    - Actual values are read from HW counters or power meter
  - **STDEV:** Standard Deviation
    - STDEV over PAAE shows the variability in PAAE
- Error is measured over the predicted performance or power at runtime

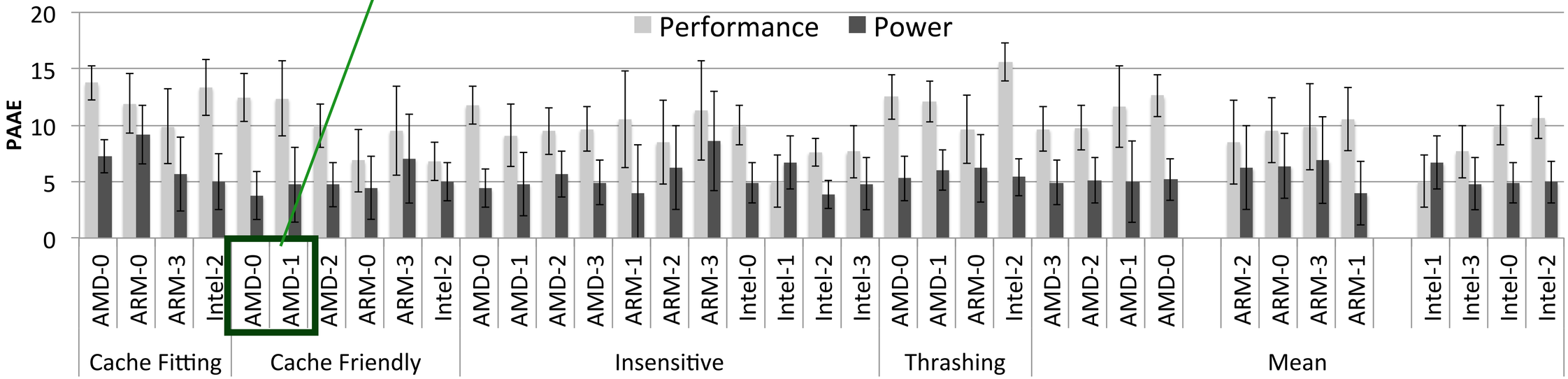
Single Core Evaluation

- Predictions over combinations of P-States and CI-States
- K-Means (T. Kanungo *et al.* TAPM'02) used to classify workloads



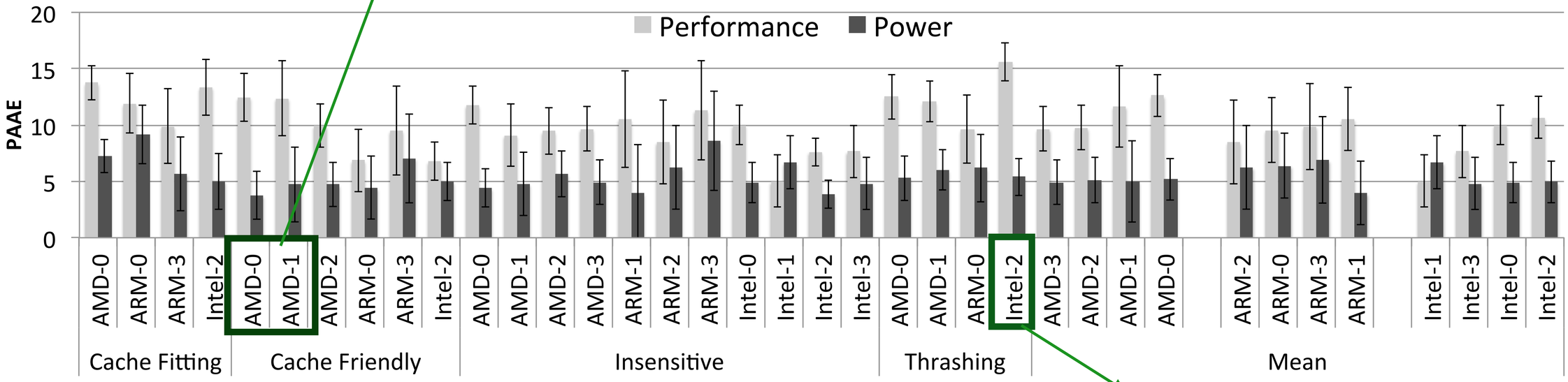
Single Core Evaluation

- Canneal and dedup have very high dynamic variability
  (D. Chasapis *et al.* TACO'15)
- Small execution time



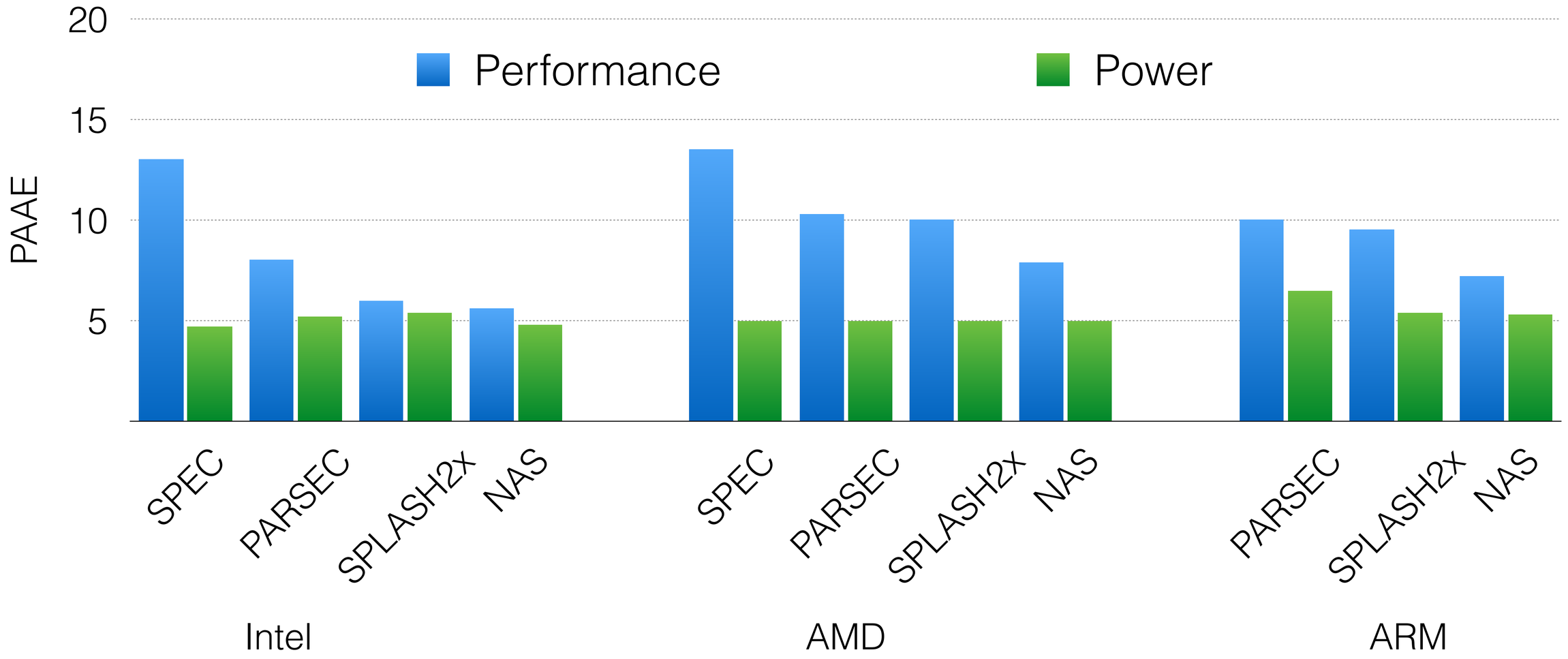
Single Core Evaluation

- Canneal and dedup have very high dynamic variability
  (D. Chasapis *et al.* TACO'15)
- Small execution time



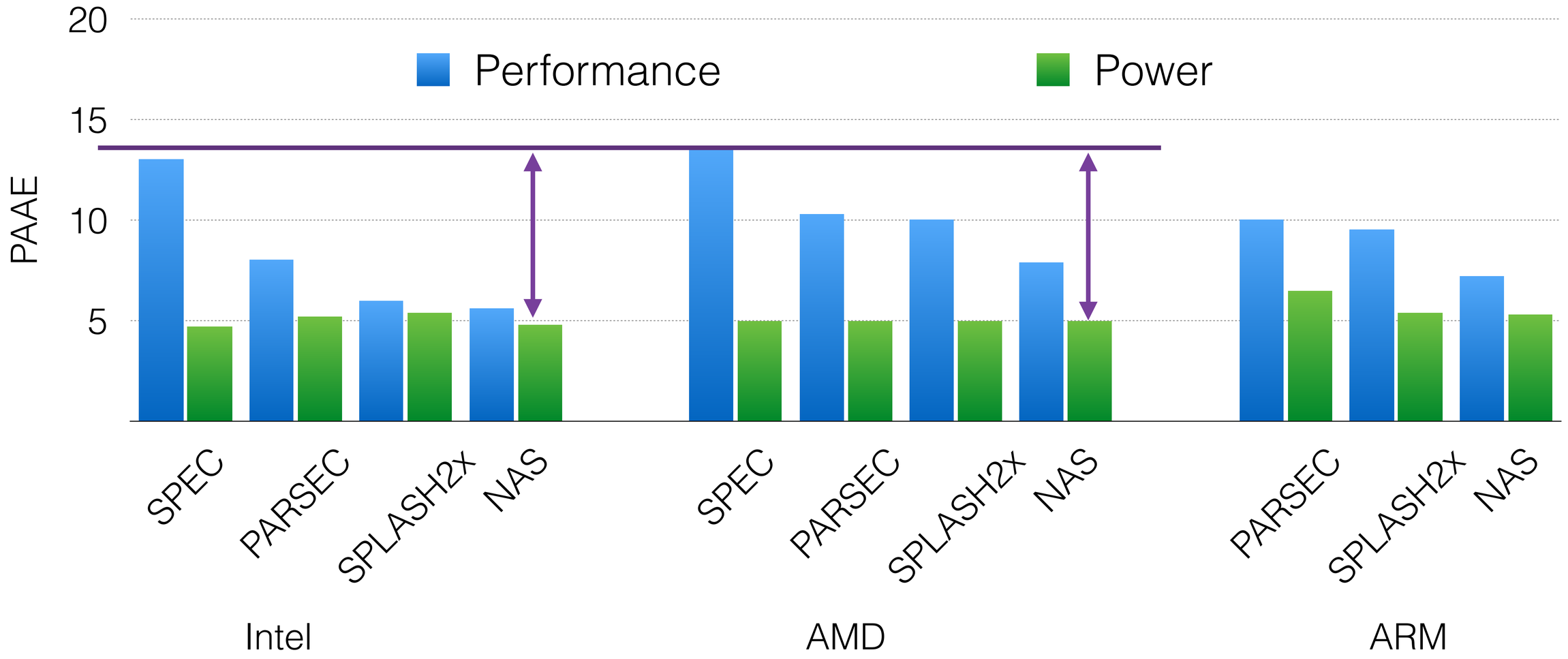
- Mcf has a very high miss rate
- Lbm, also thrashing, but lower error

Per-suite Evaluation



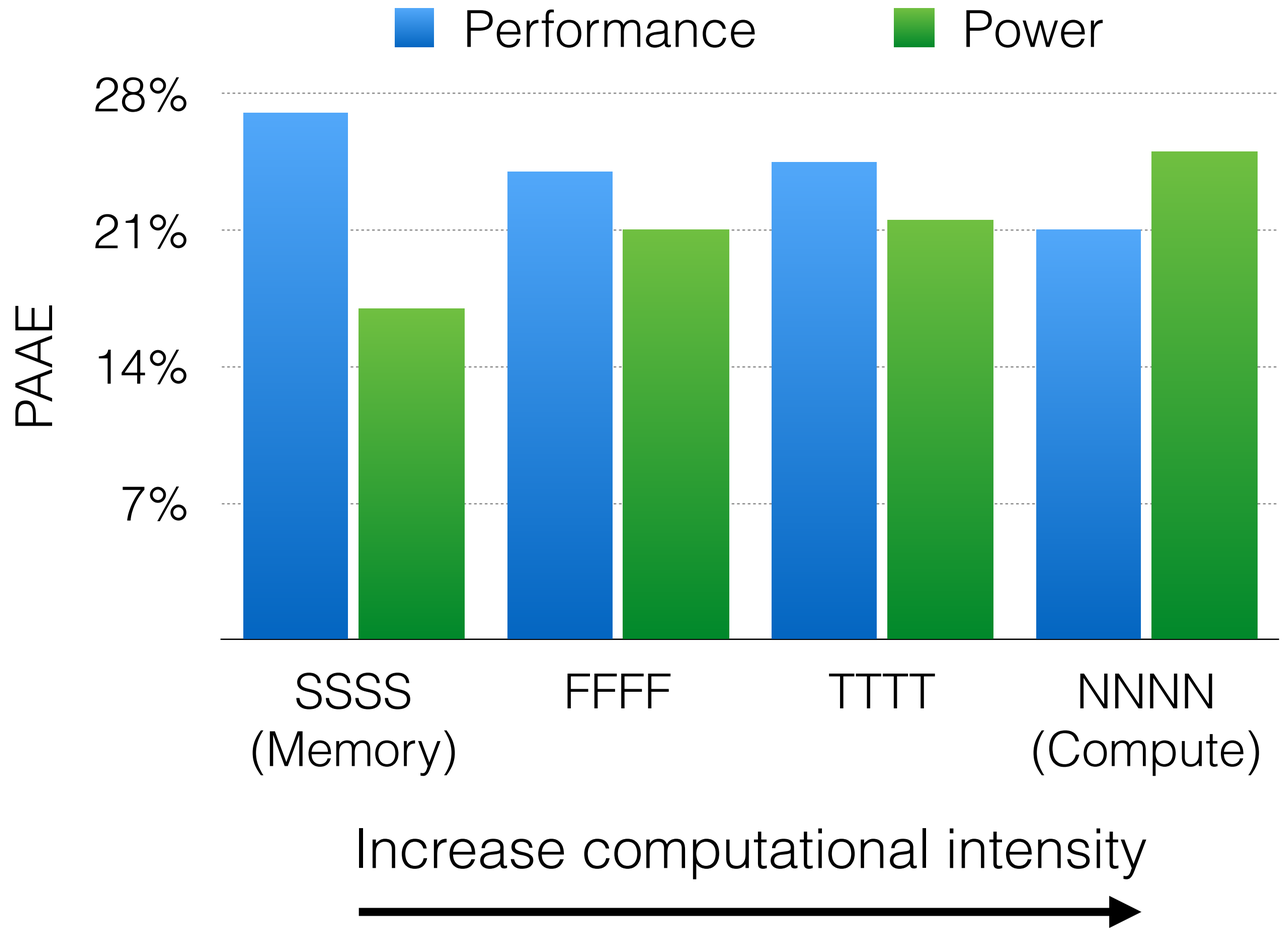
Average error less than 10% across architectures

Per-suite Evaluation



Average error less than 10% across architectures

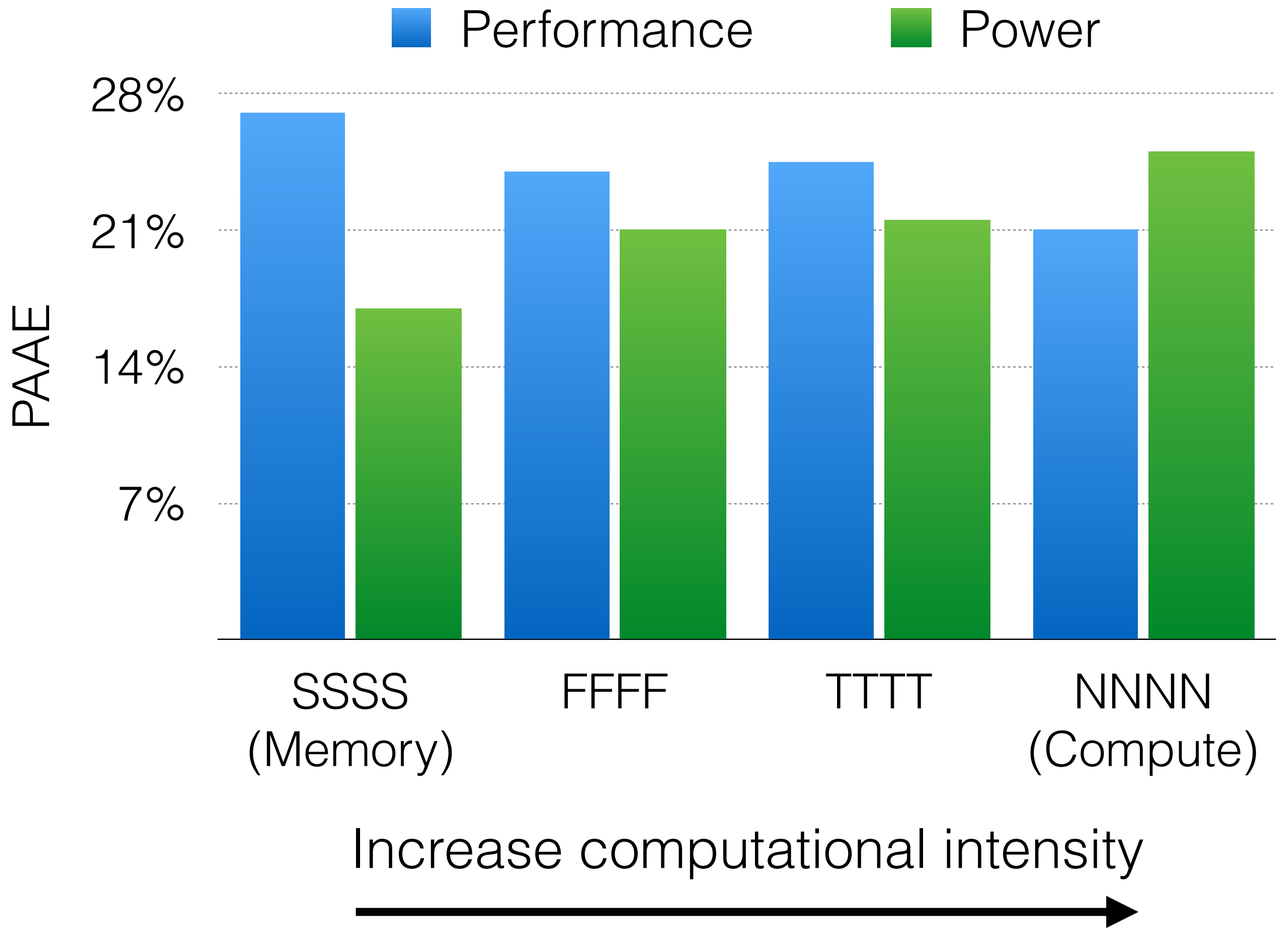
Prediction Error Ignoring Shared Resource Contention



Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

Prediction Error Ignoring Shared Resource Contention

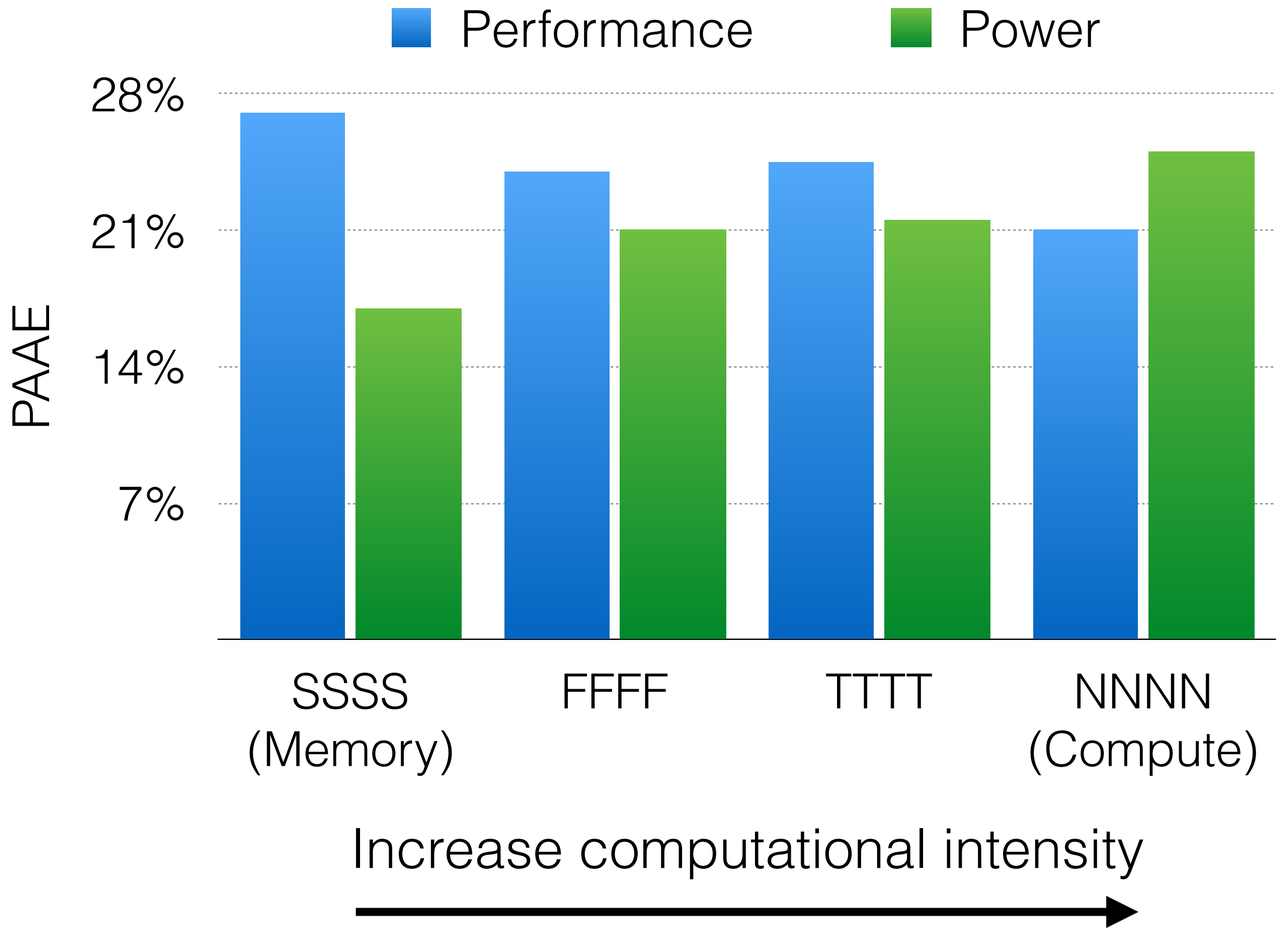
- Performance prediction
  - Error increases with memory intensity
  - As aggregation of single core results gives a lower activity in memory subsystem



Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

Prediction Error Ignoring Shared Resource Contention

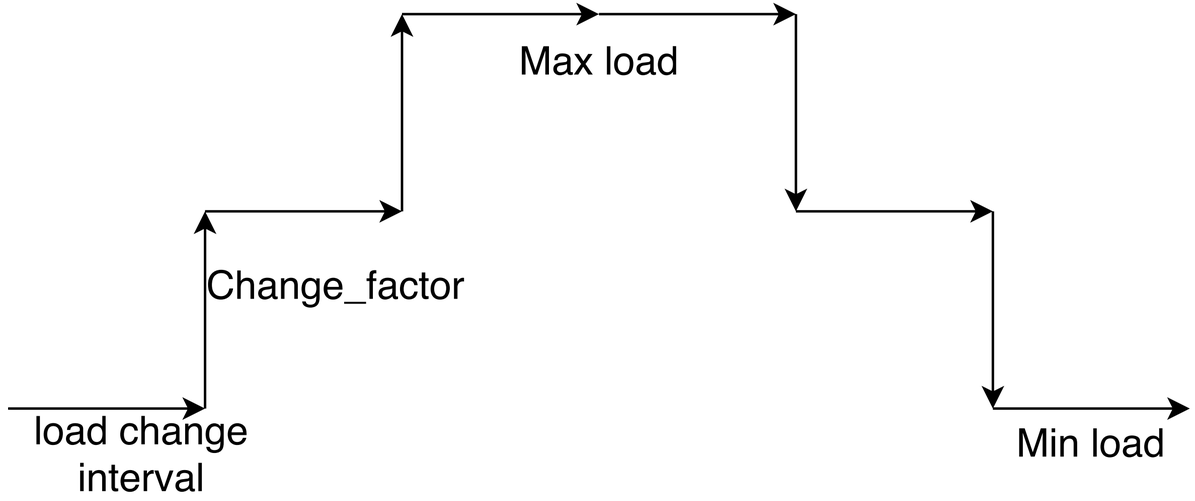
- Performance prediction
  - Error increases with memory intensity
  - As aggregation of single core results gives a lower activity in memory subsystem
- Power prediction
  - Error increase with compute intensity
  - As aggregation of single core results gives a higher activity in memory subsystem



Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

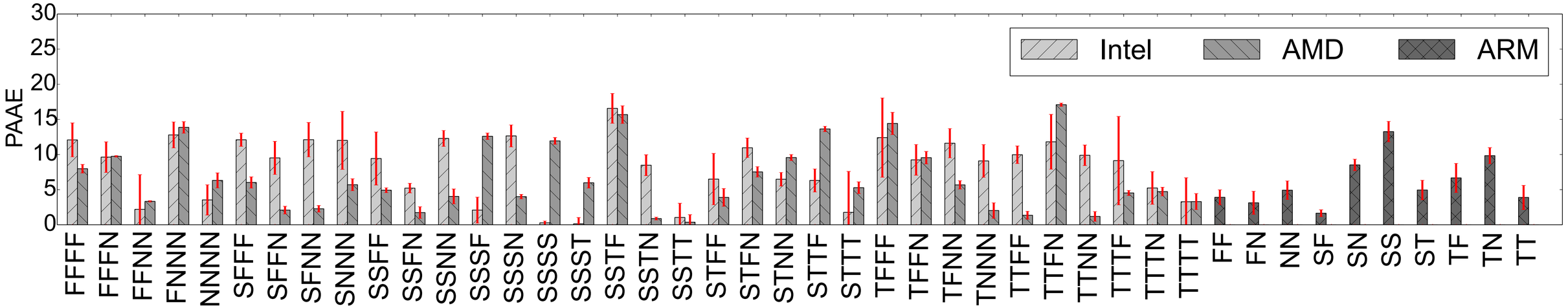
Power and Performance Targets

- We define two parameters
  - Load change interval: 1 s, 6 s, 9 s
  - Change_factor: 20%, 35%, 50%
  - Combination gives us 9 parameters
  - Random parameter



- Error can occur in two instances:
  - Falls short of performance target
  - Exceeds power requirement

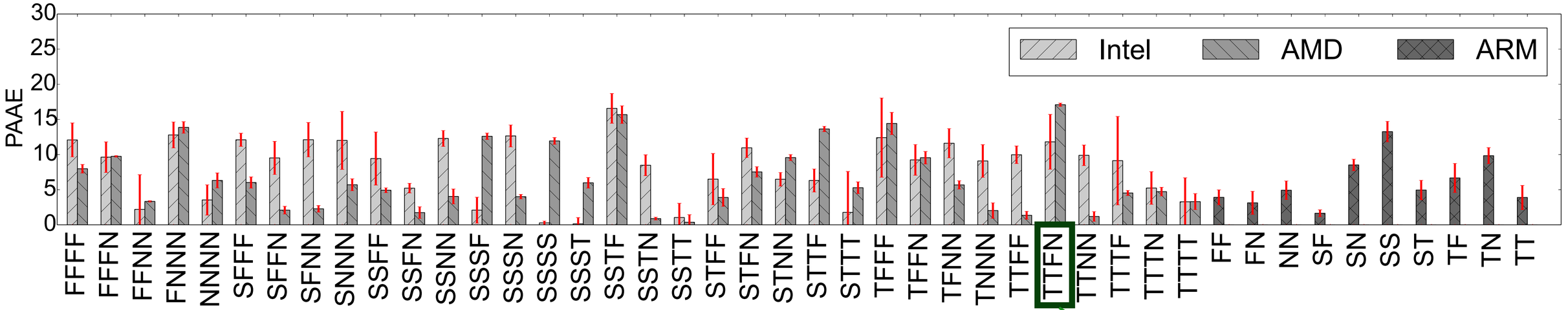
Multicore Power Evaluation



Average PAAE: Intel (8%), AMD (6%), ARM (6%)

Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

Multicore Power Evaluation

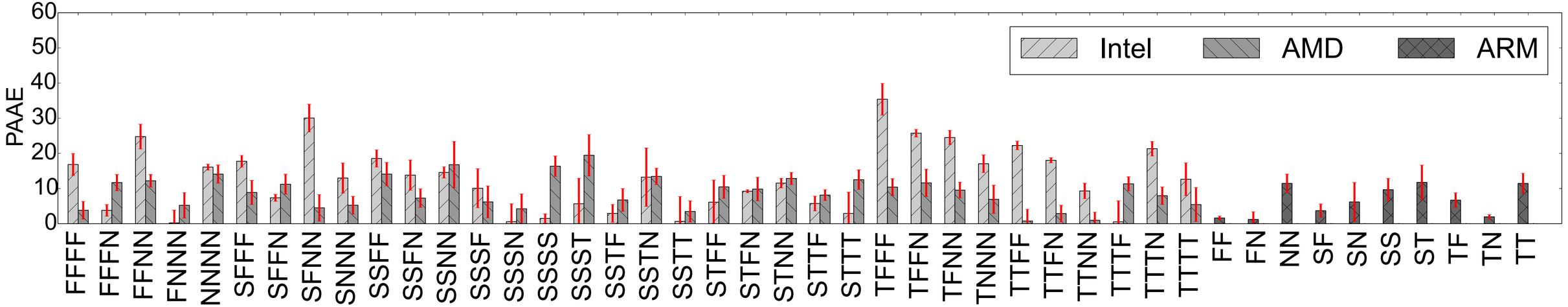


Radix is a part of the multi-programmed workloads

Average PAAE: Intel (8%), AMD (6%), ARM (6%)

Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

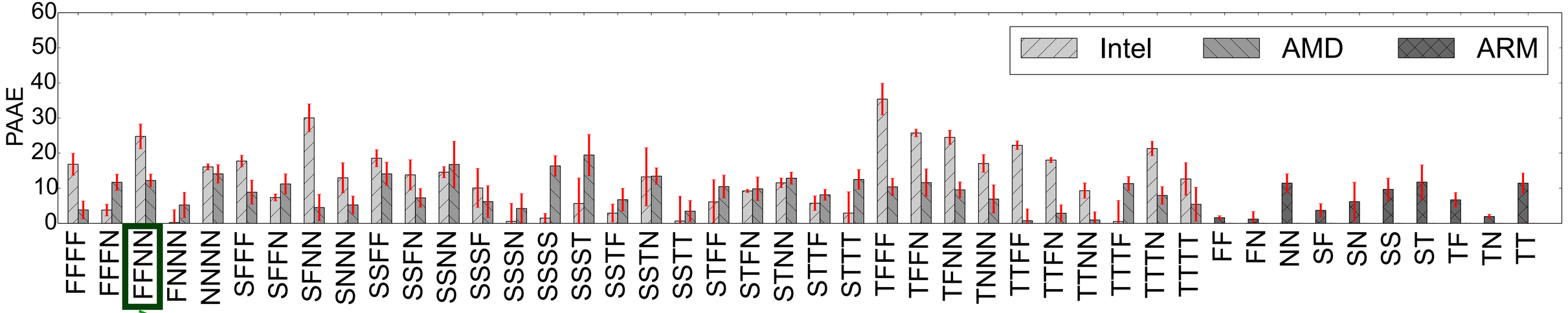
Multicore Performance Evaluation



Average PAAE: Intel (7%), AMD (9%), ARM (7%)

Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

Multicore Performance Evaluation

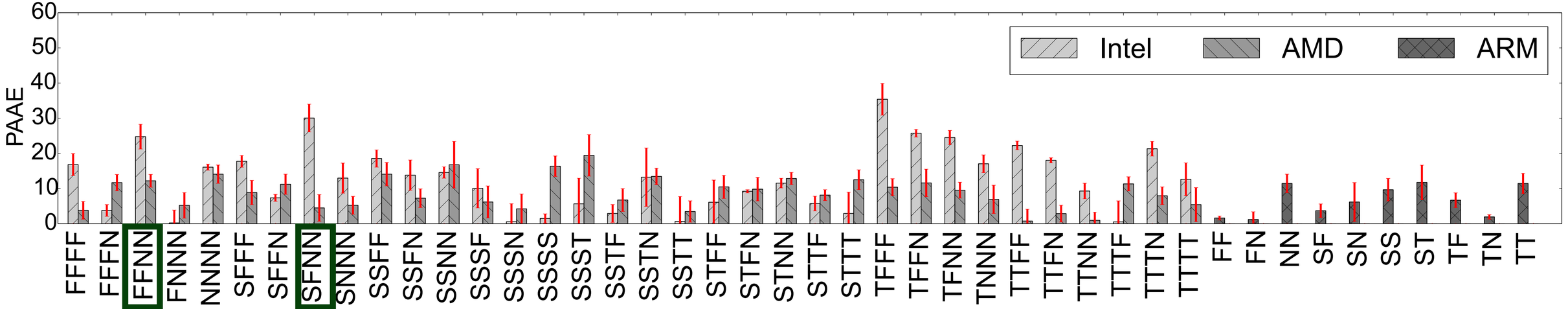


- ep.C high activity in BPU and FE

Average PAAE: Intel (7%), AMD (9%), ARM (7%)

Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

Multicore Performance Evaluation

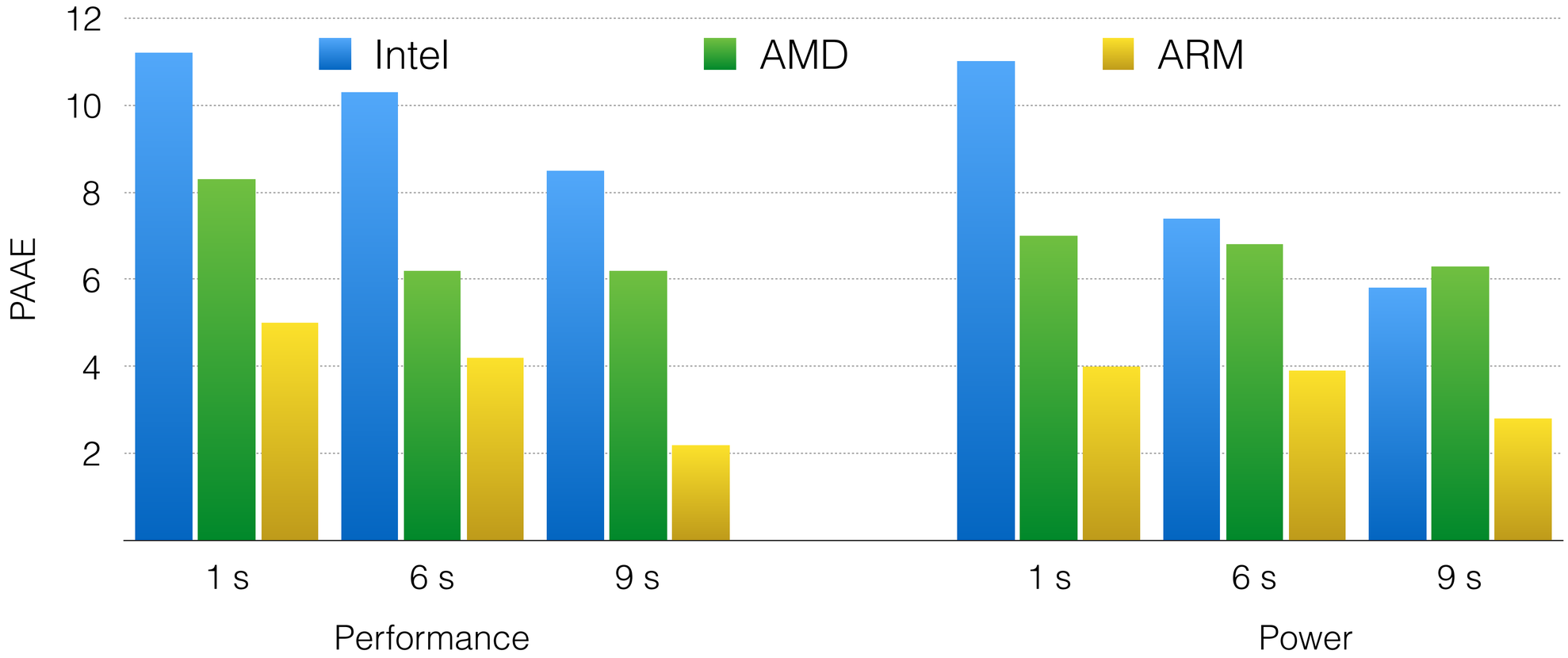


- ep.C high activity in BPU and FE

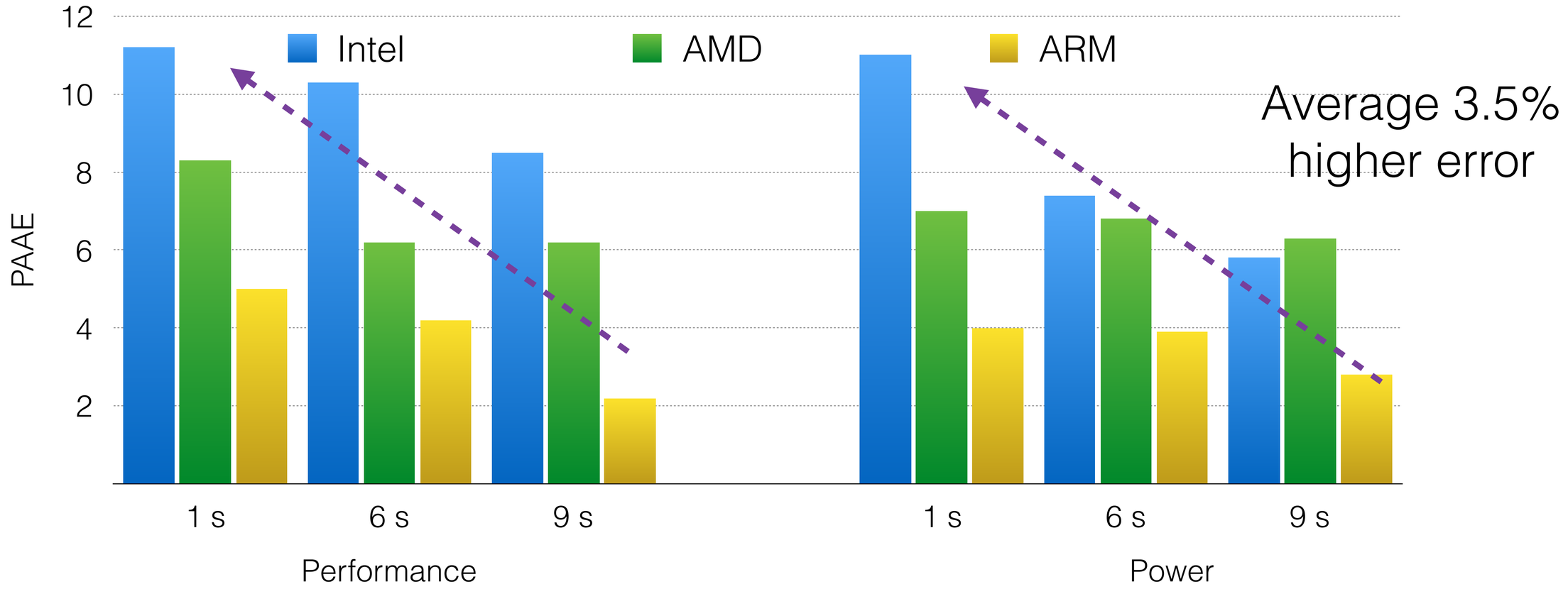
- lu_ncb, non-contiguous block of memory, higher activity in LLC

Average PAAE: Intel (7%), AMD (9%), ARM (7%)

Insensitive (N); Cache-Friendly (F); Cache-Fitting (T); Thrashing/Streaming (S)

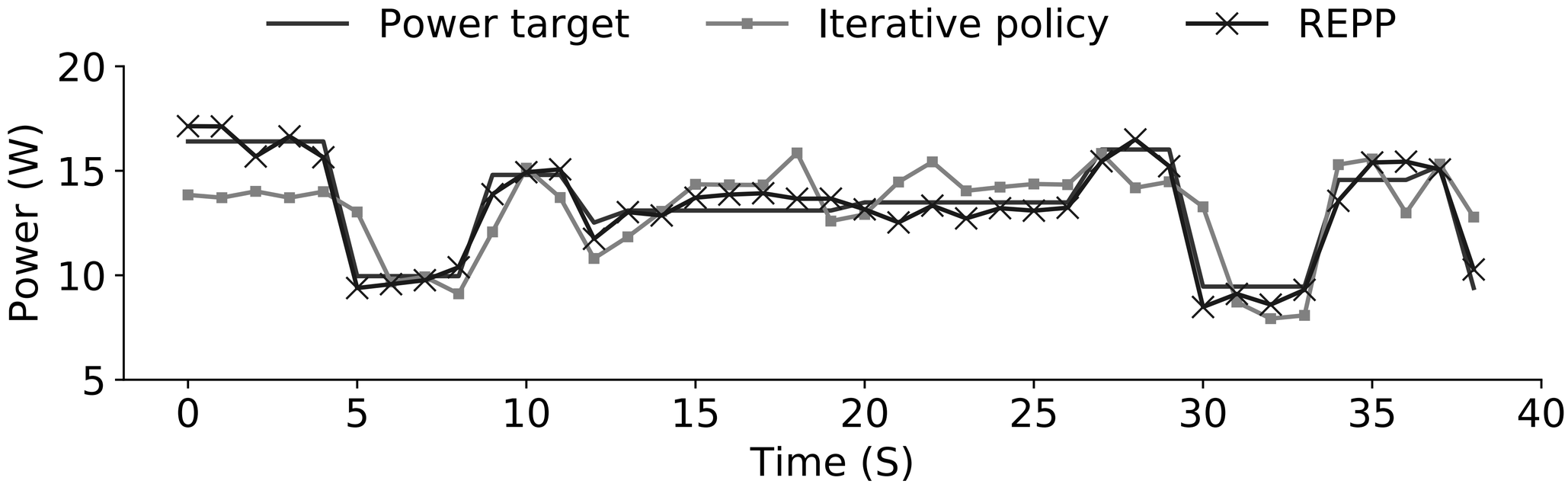


Evaluation across load change intervals



Evaluation across load change intervals

Rapid changes in load seldom occur in data centres



Responsiveness to power capping on Intel processors

Single step power capping mechanism

Meets power target 0.37s on average and 3.6X faster

REPP Conclusion

- REPP: Runtime Estimation of Performance and Power
  - Hardware knobs: P-States and CI-States
  - Architecture and application agnostic
- Prediction models using widely available counters/meters
- Evaluated on real platforms
  - Intel SandyBridge, AMD Opteron II, and ARM Juno R1
  - SPEC CPU2006, PARSEC 3.0, SPLASH-2x, and NAS
- Mean prediction accuracy greater than 93%

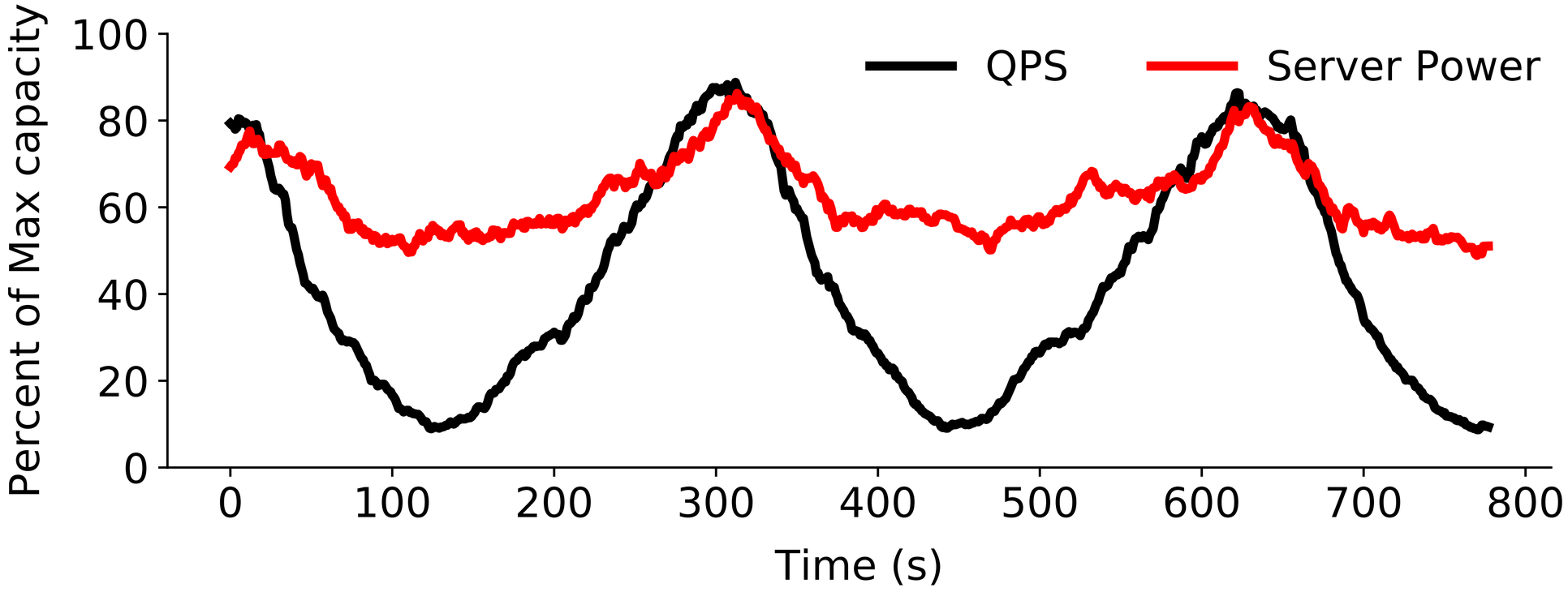
Talk Organisation

- Motivation
- Thesis contributions
- REPP: Runtime Estimation of Performance and Power
- HIPSTER: HYBRID TASK MANAGER FOR LATENCY-CRITICAL CLOUD WORKLOADS
- Future work & Contributions recap

Hipster

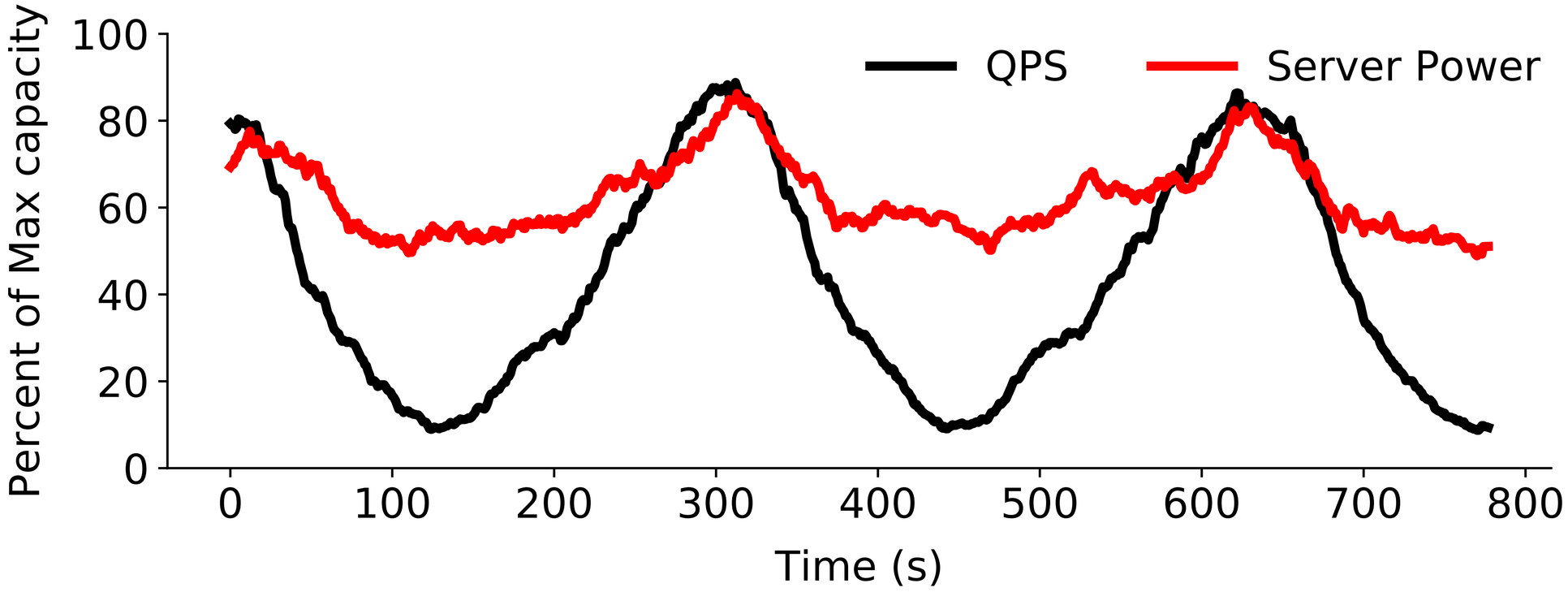
Hybrid Task Manager for Latency-Critical Cloud Workloads

R. Nishtala *et al.*
4. "***Hipster: Hybrid Task Manager for Latency-Critical Cloud Workloads***",
IEEE Symposium on High Performance Computer Architecture (HPCA) 2017 — *HiPEAC Paper Award 2017*
5. "***The Hipster Approach for Improving Cloud System Efficiency***",
ACM Transactions on Computer Systems (TOCS) 2017. Under review, submitted on Feb 24, 2017


Percent of Max capacity
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
Time (s)
QPS
Server Power

Prior Work*

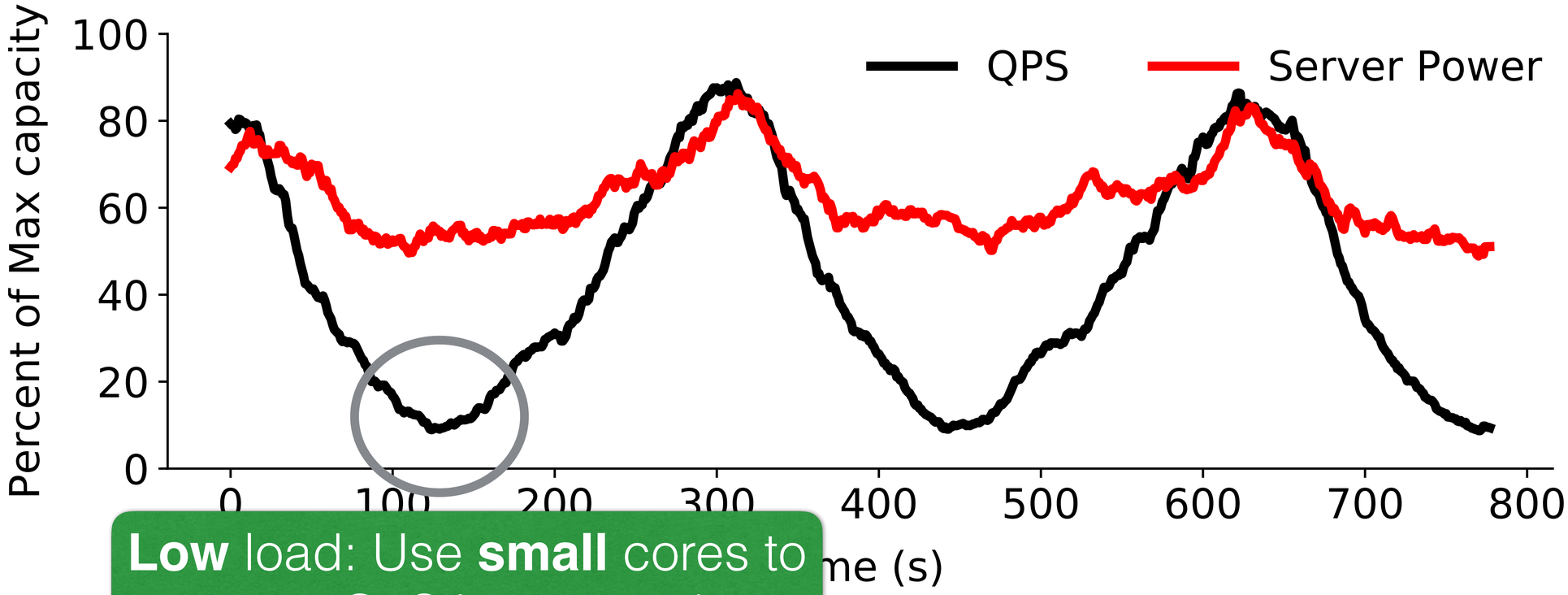
Explores heterogeneous (big.LITTLE) arch for improved resource efficiency



*Petrucci *et al.* HPCA'15

Prior Work*

Explores heterogeneous (big.LITTLE) arch for improved resource efficiency

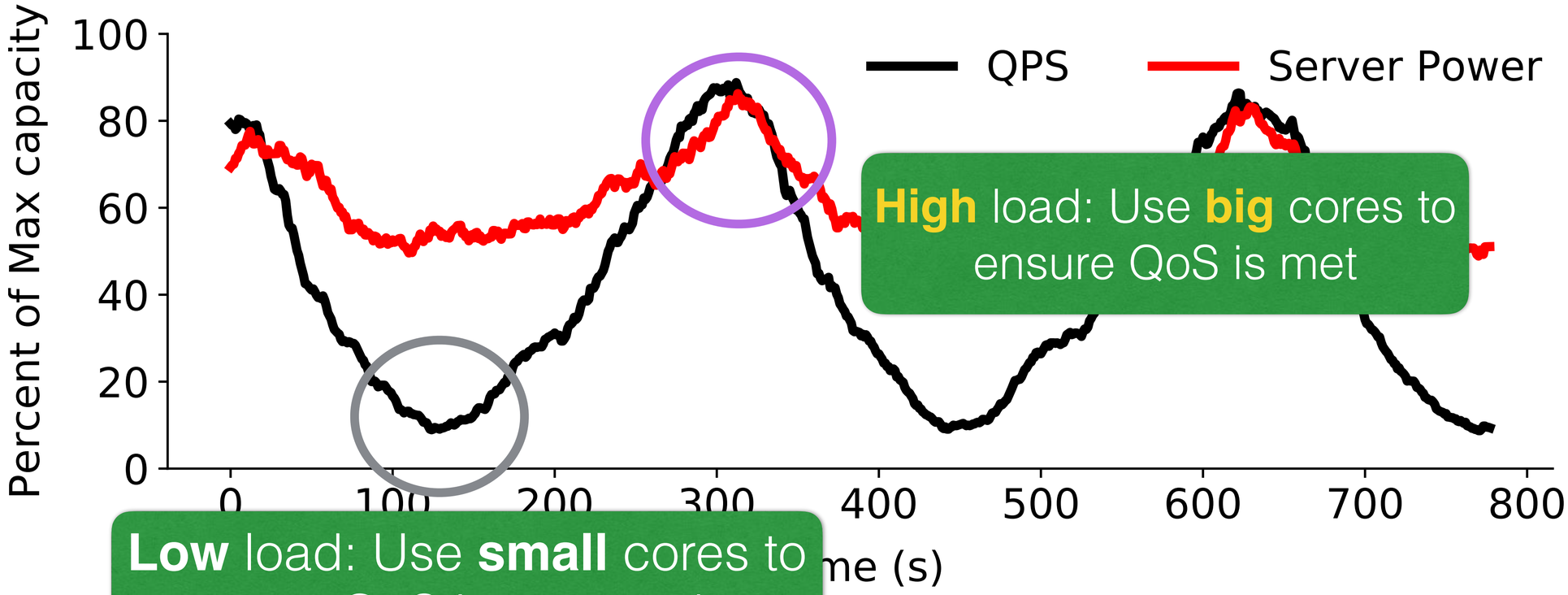


Low load: Use **small** cores to meet QoS just enough

*Petrucci *et al.* HPCA'15

Prior Work*

Explores heterogeneous (big.LITTLE) arch for improved resource efficiency



*Petrucci *et al.* HPCA'15

QoS vs. Resource Efficiency

QoS vs. Resource Efficiency

- Meeting QoS is important
  - You don't want to wait 1 s for Google to reply back!
    - End users notice if the QoS is violated significantly
  - Example of QoS = 99th percentile of all requests are on time

QoS vs. Resource Efficiency

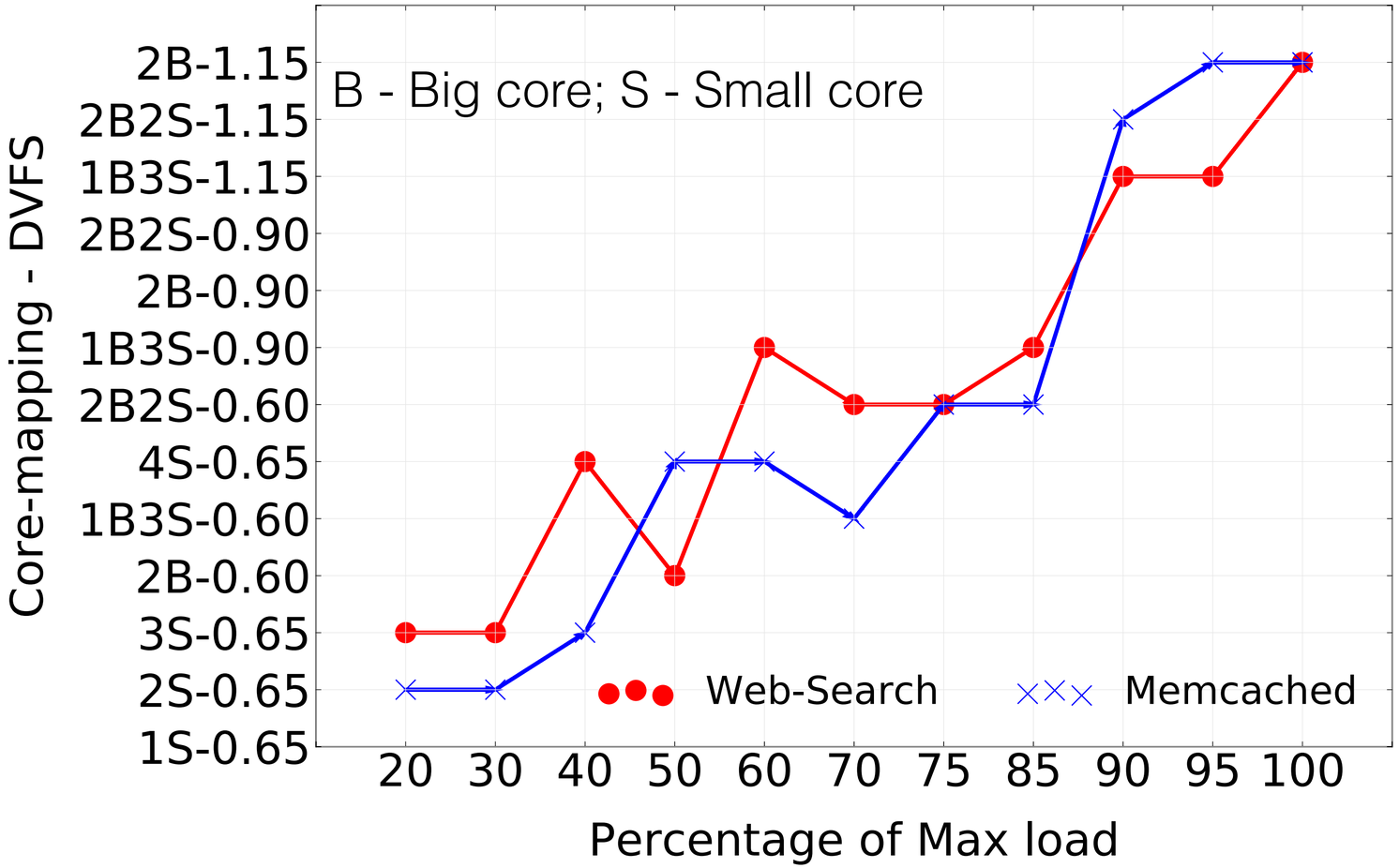
- Meeting QoS is important
  - You don't want to wait 1 s for Google to reply back!
    - End users notice if the QoS is violated significantly
  - Example of QoS = 99th percentile of all requests are on time
- But it is more efficient to meet **workload specific** requirements just enough to meet the QoS and minimise power or maximise utilisation

QoS vs. Resource Efficiency

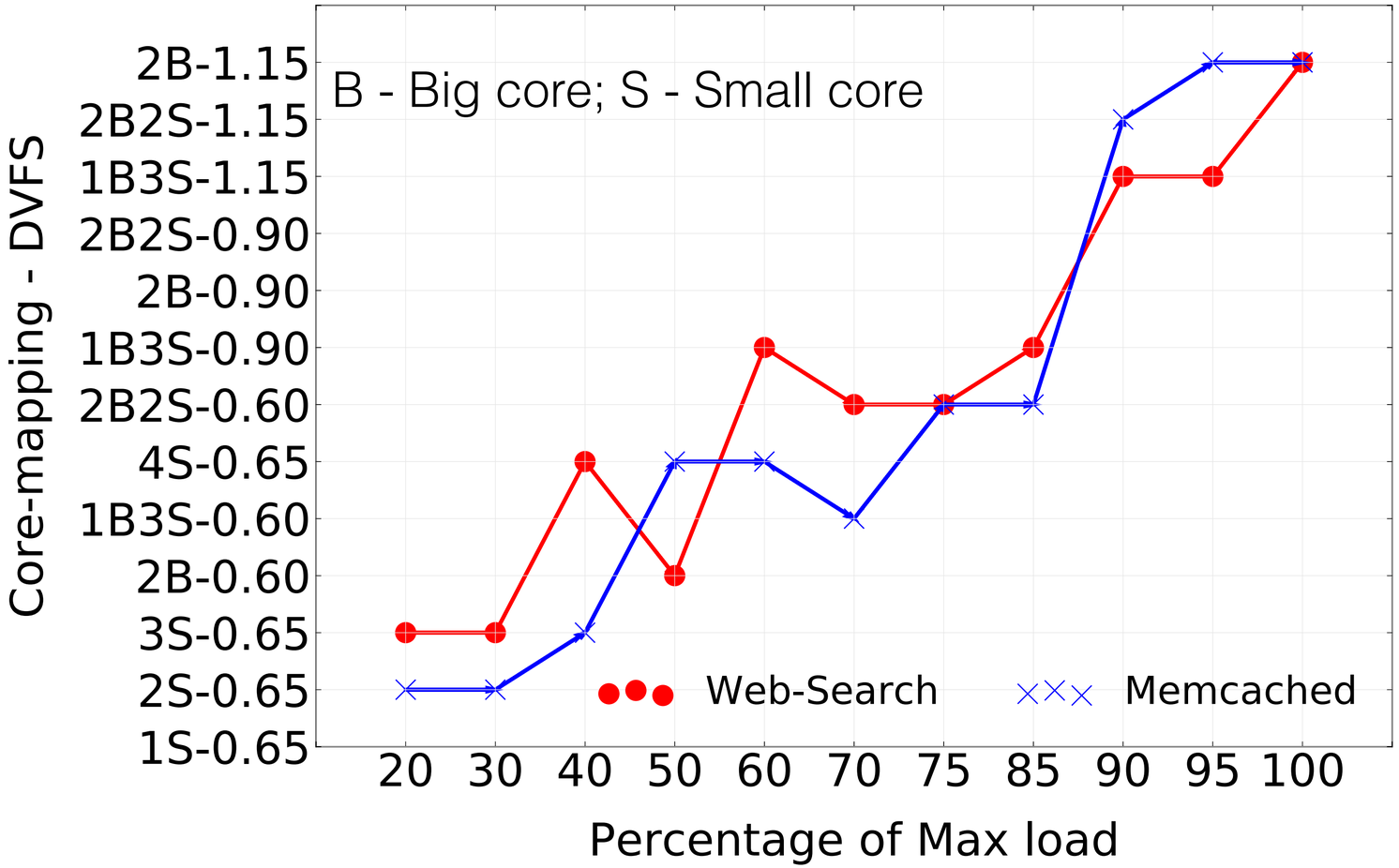
- Meeting QoS is important
  - You don't want to wait 1 s for Google to reply back!
    - End users notice if the QoS is violated significantly
  - Example of QoS = 99th percentile of all requests are on time
- But it is more efficient to meet **workload specific** requirements just enough to meet the QoS and minimise power or maximise utilisation

Characterise workload specific requirements to deliver adequate resources

Workload Specific Mapping

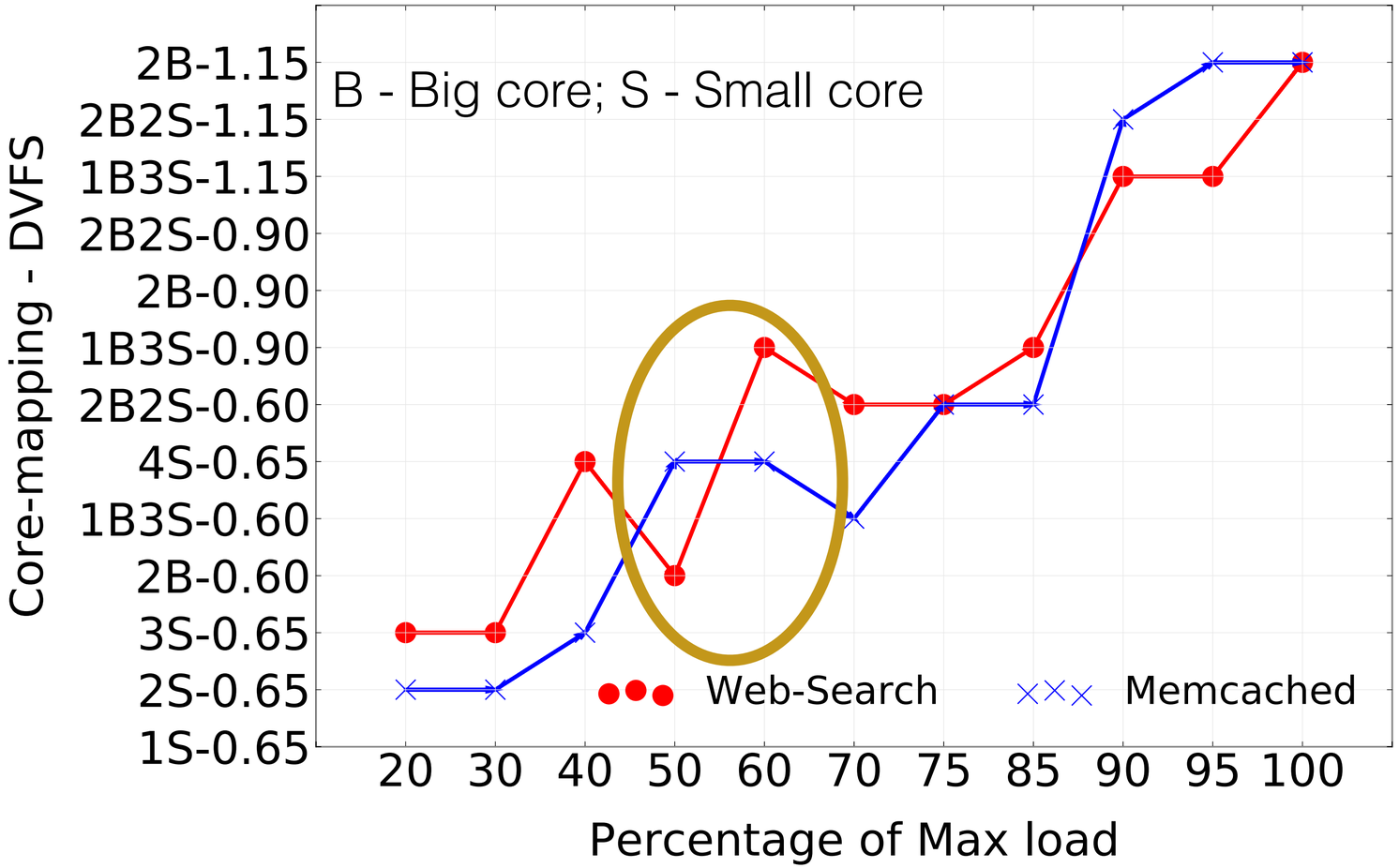


Workload Specific Mapping



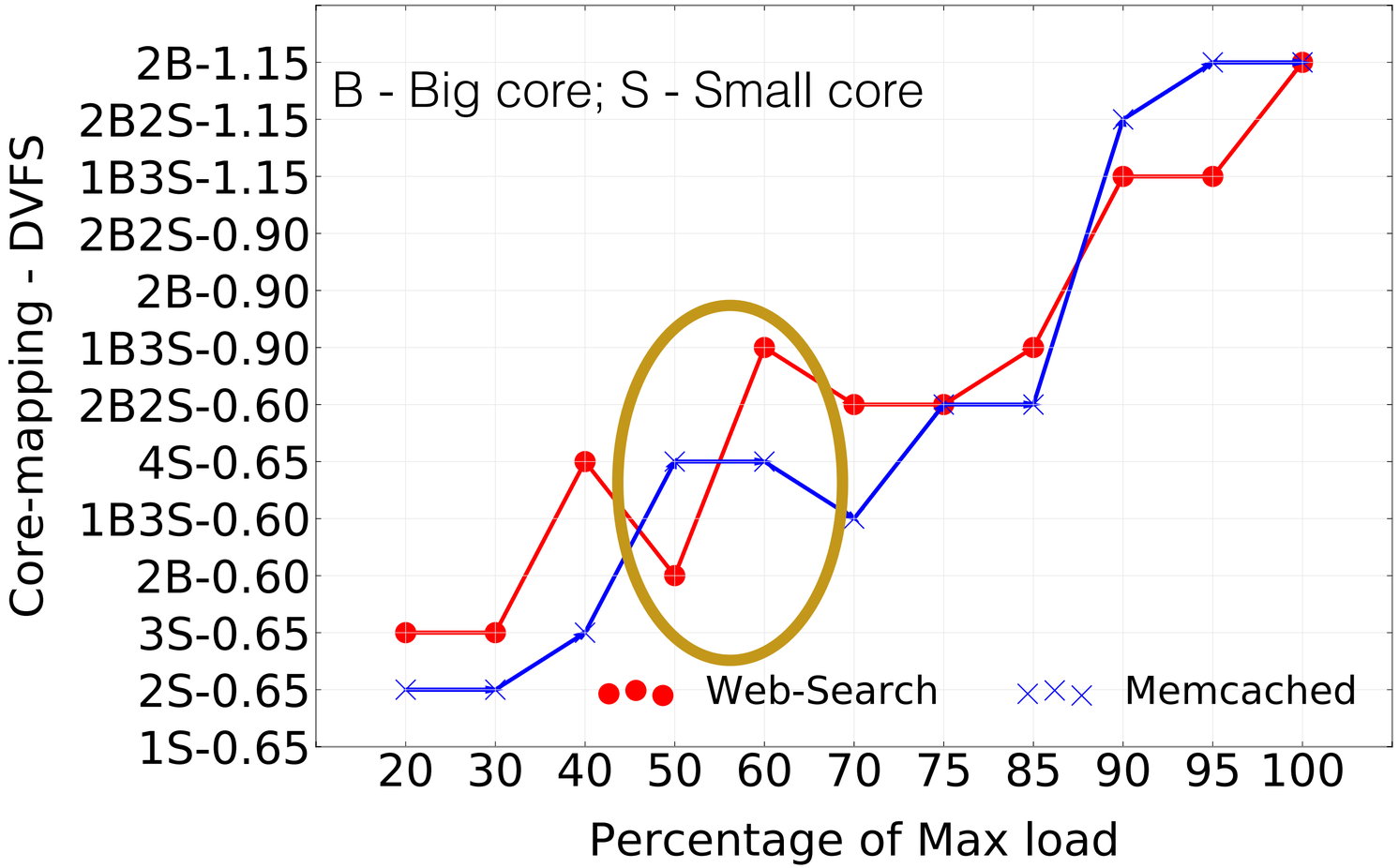
- Most energy efficient mapping that meets QoS

Workload Specific Mapping



- Most energy efficient mapping that meets QoS
- Applications need different mappings at same load

Workload Specific Mapping



- Most energy efficient mapping that meets QoS
- Applications need different mappings at same load

Application-tailored mappings are more energy efficient

Hipster in a nutshell

Hipster in a nutshell

- **Hipster:** Hybrid reinforcement learning mechanism
  - Application load used to guide system configuration
  - **Constraint:** Satisfies QoS requirements for interactive workload

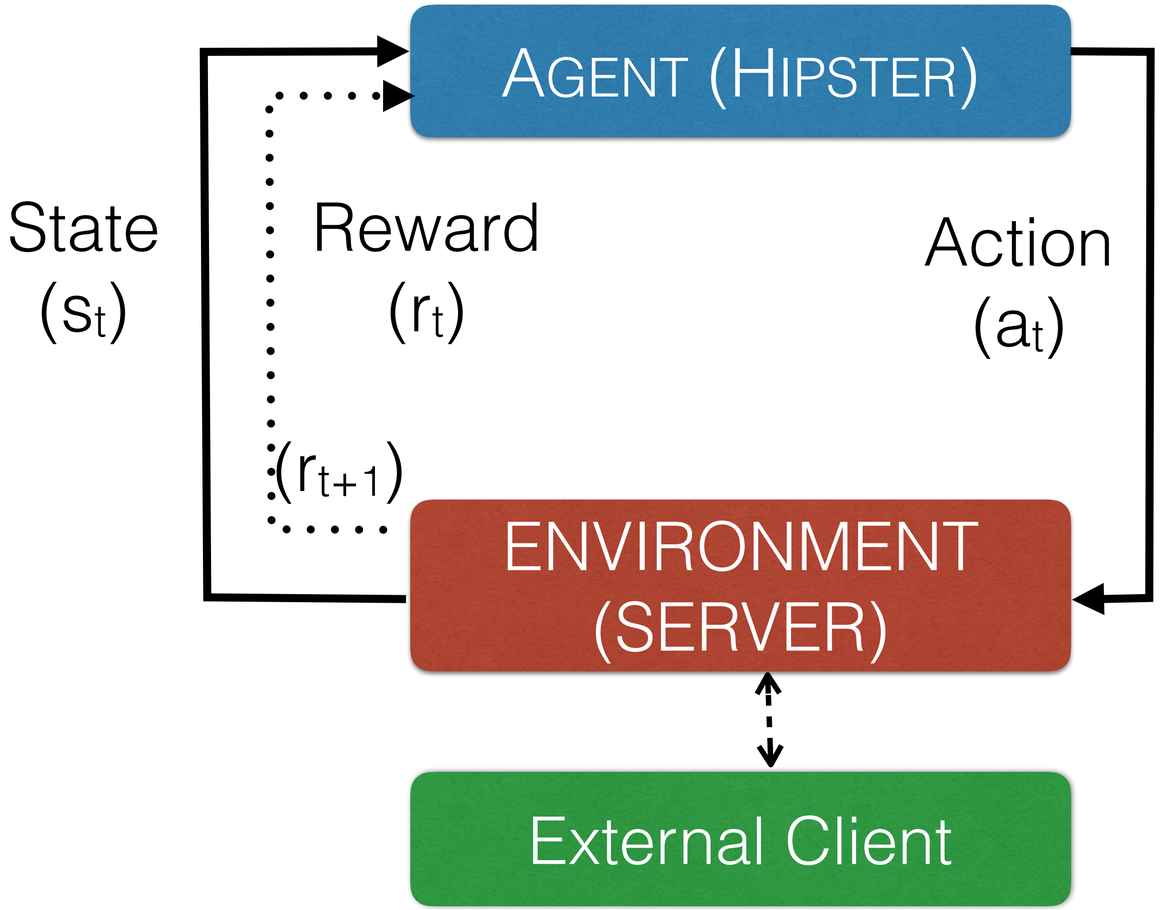
Hipster in a nutshell

- **Hipster:** Hybrid reinforcement learning mechanism
  - Application load used to guide system configuration
  - **Constraint:** Satisfies QoS requirements for interactive workload

- Two variants of Hipster:
  - **HipsterIn:** Interactive workload running solo
    - **Goal:** Optimised for energy efficiency
  - **HipsterCo:** Collocate interactive and batch workload
    - **Goal:** Optimised for improving resource utilisation

Reinforcement Learning

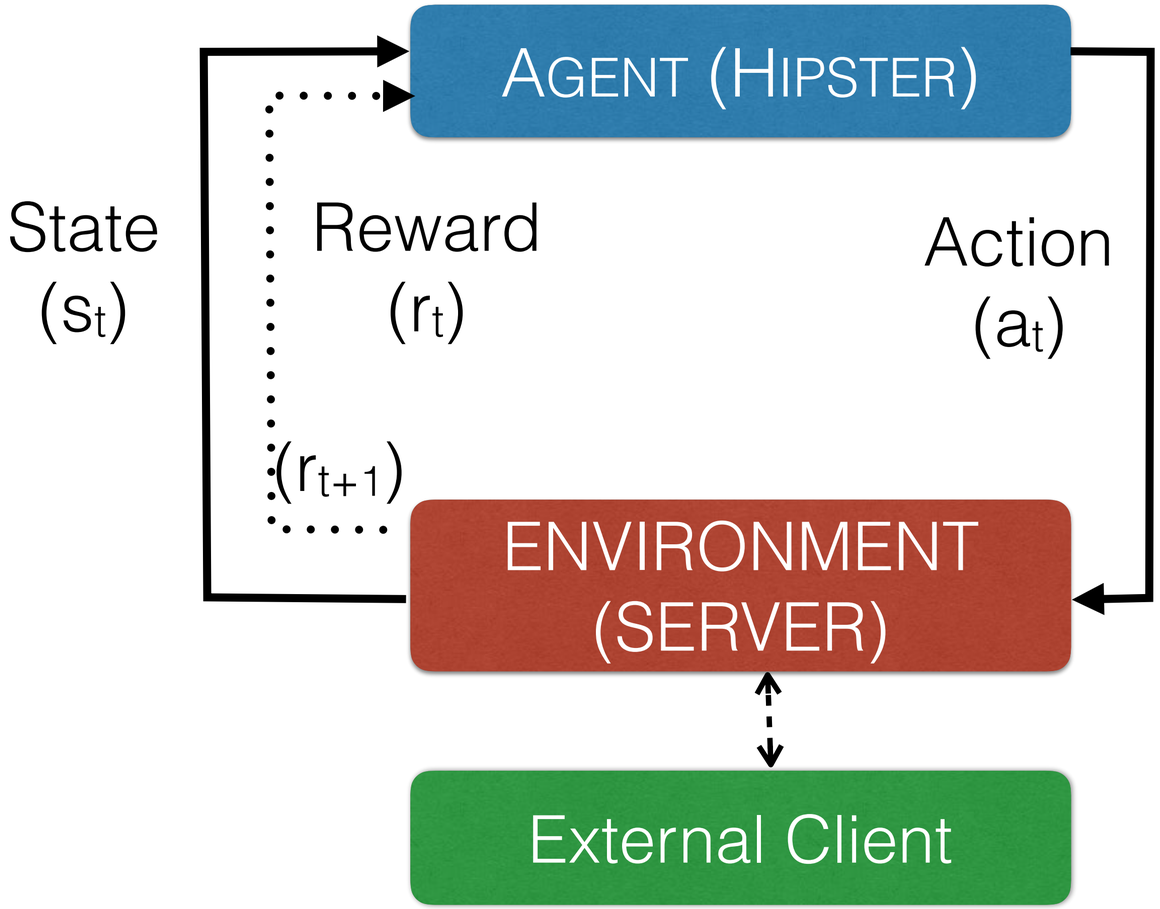
- **Agent:** Learner and decision maker
- **Environment:** Everything outside the agent
- **State:** Current state of the environment
- **Reward:** Based on an action, the agent receives a reward



The environment is formulated by Markov Decision Process (MDP)

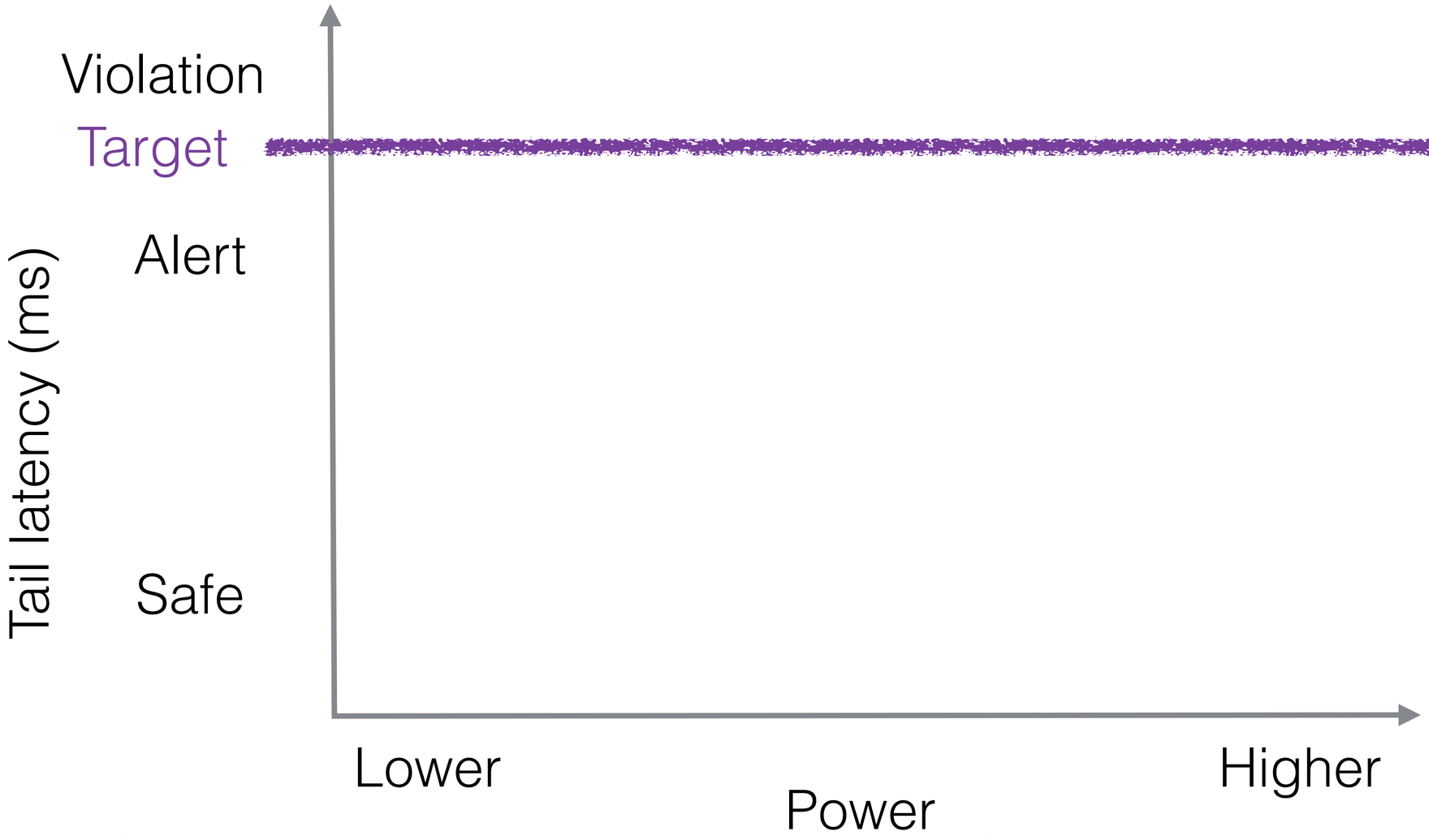
Hipster's MDP Formulation

- **State:** Load = # of requests / sec
- **Action:** Core mapping + DVFS state
- **Reward:** Balance of QoS guarantee, and power usage or resource utilisation

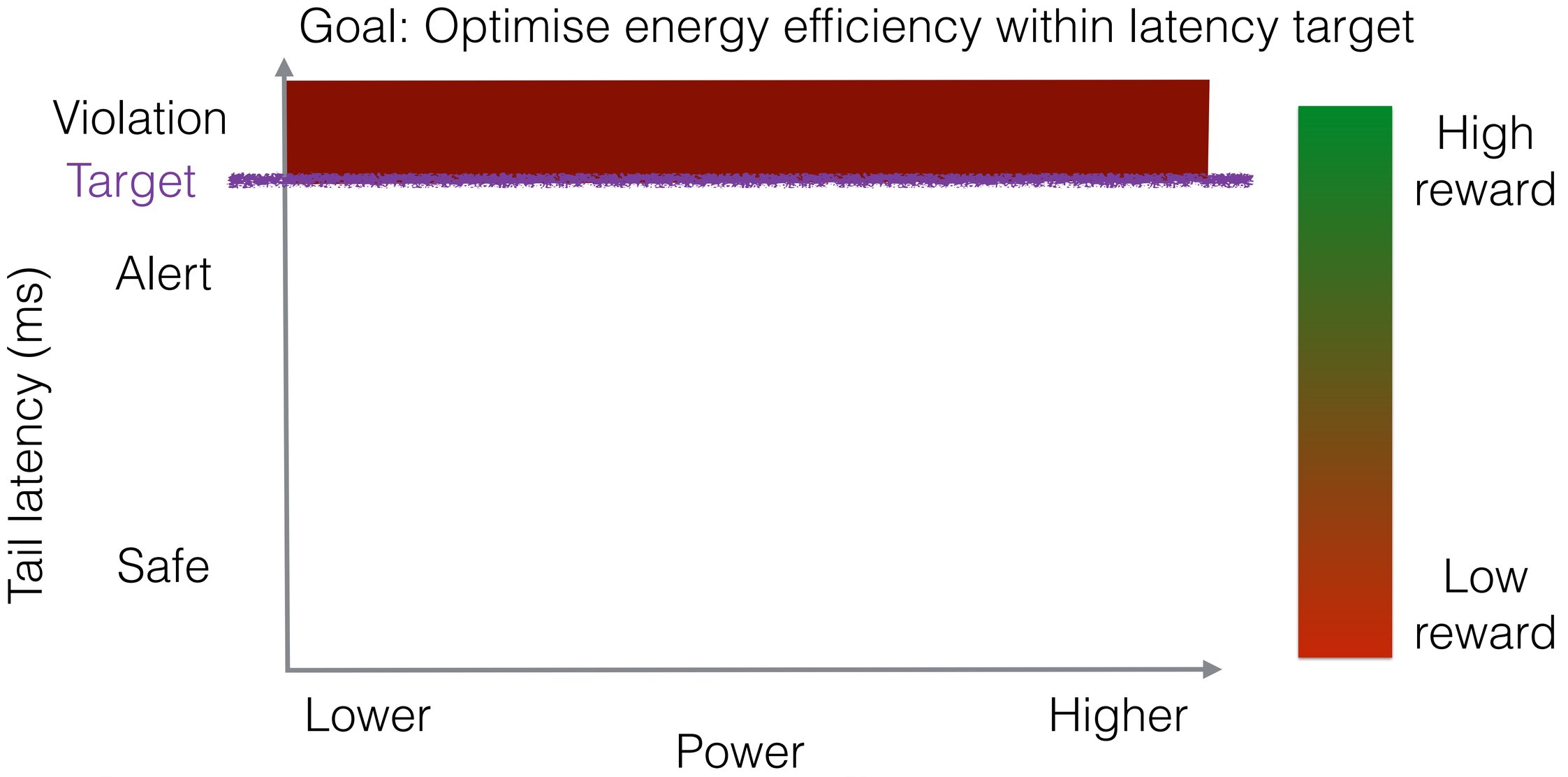


HipsterIn's Reward Mechanism

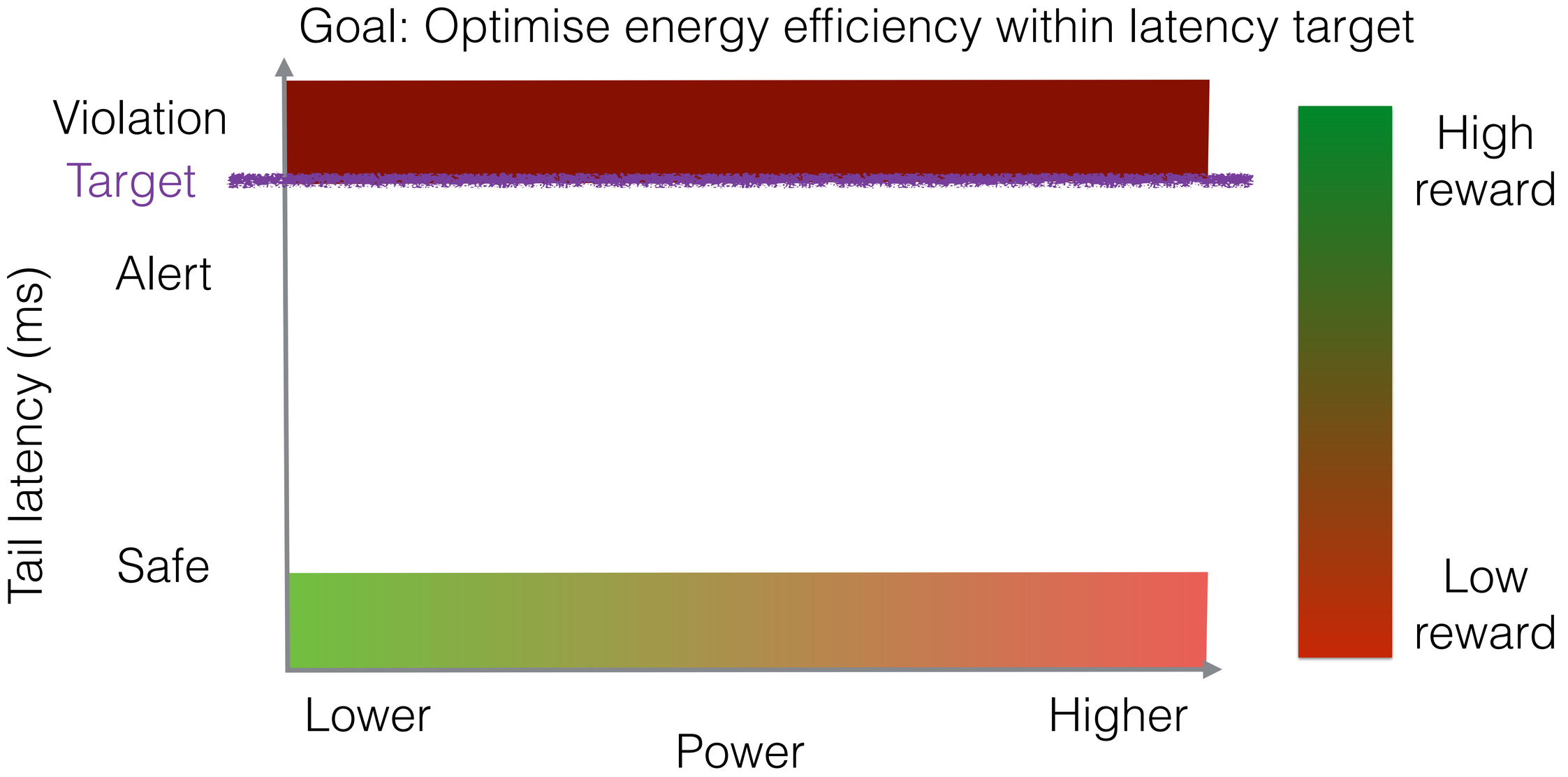
Goal: Optimise energy efficiency within latency target



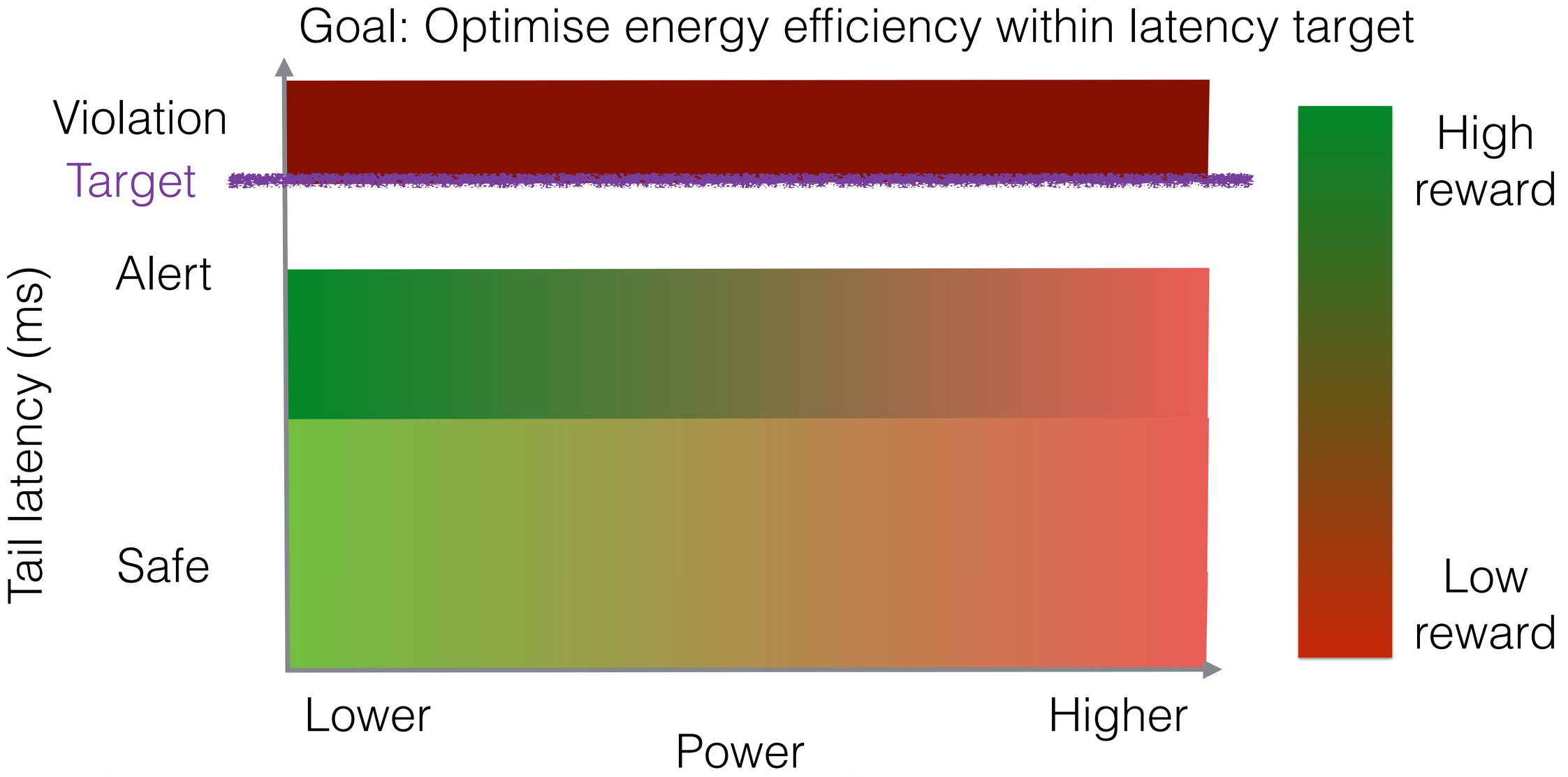
HipsterIn's Reward Mechanism



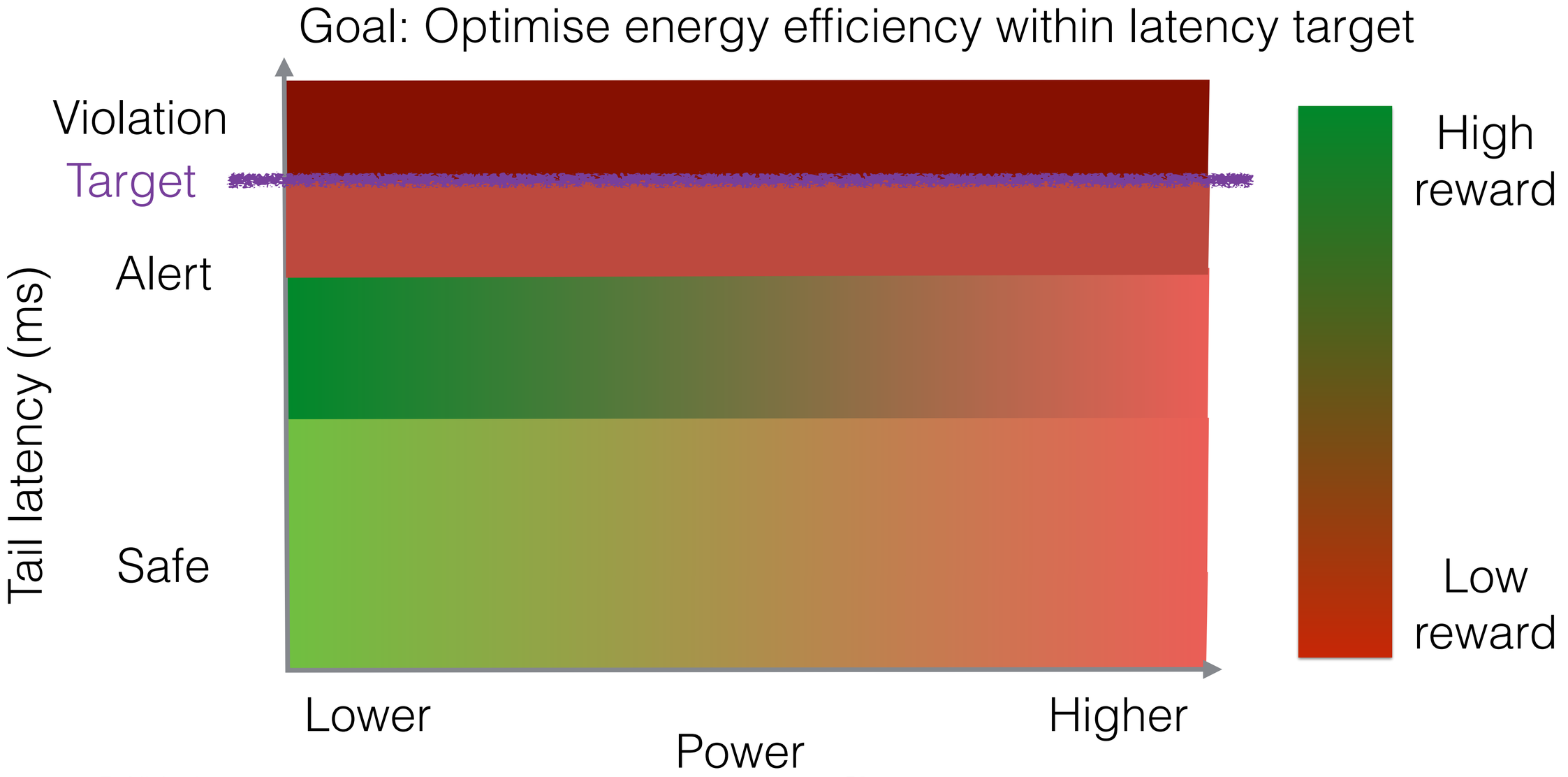
HipsterIn's Reward Mechanism



HipsterIn's Reward Mechanism

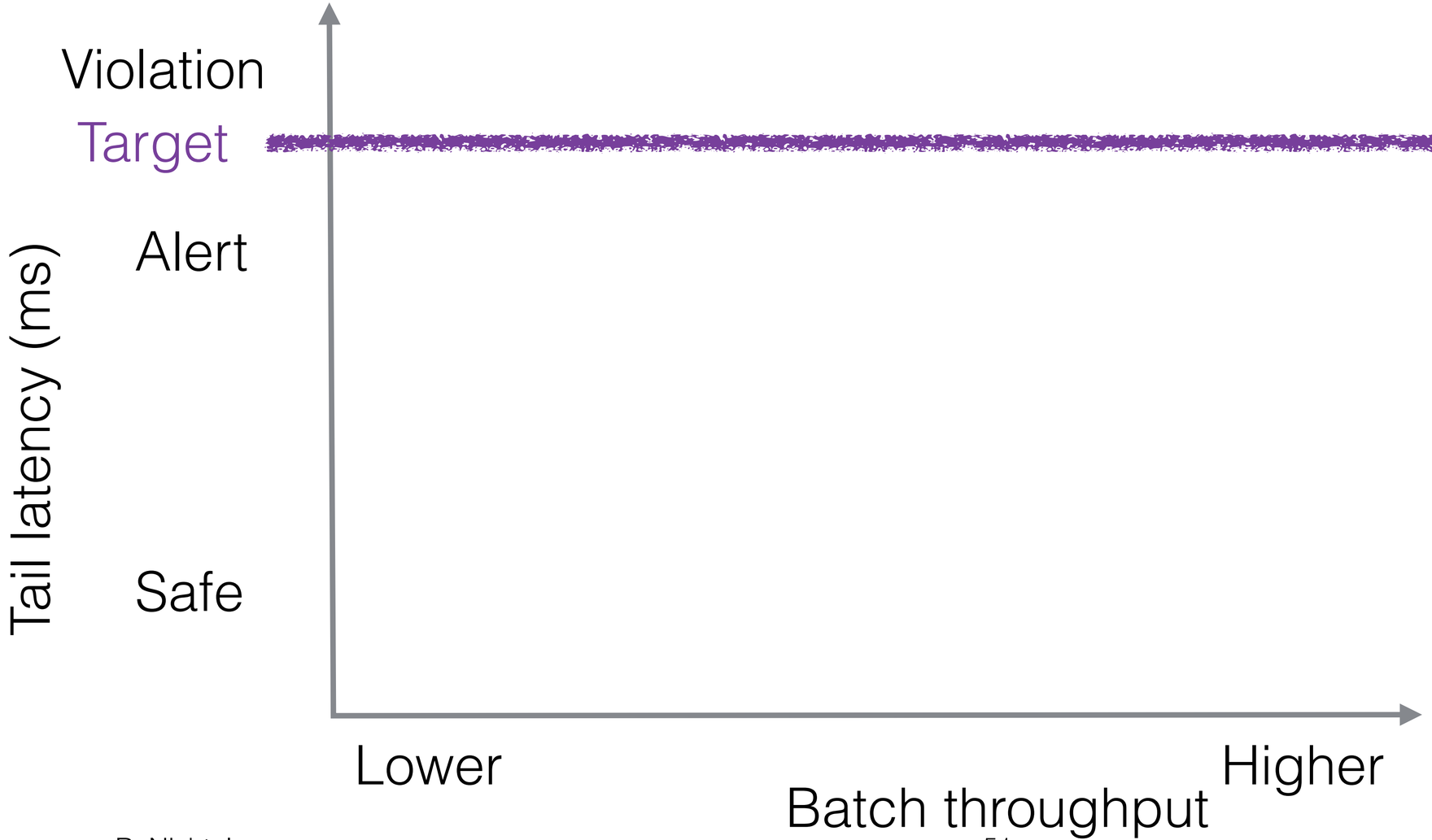


HipsterIn's Reward Mechanism

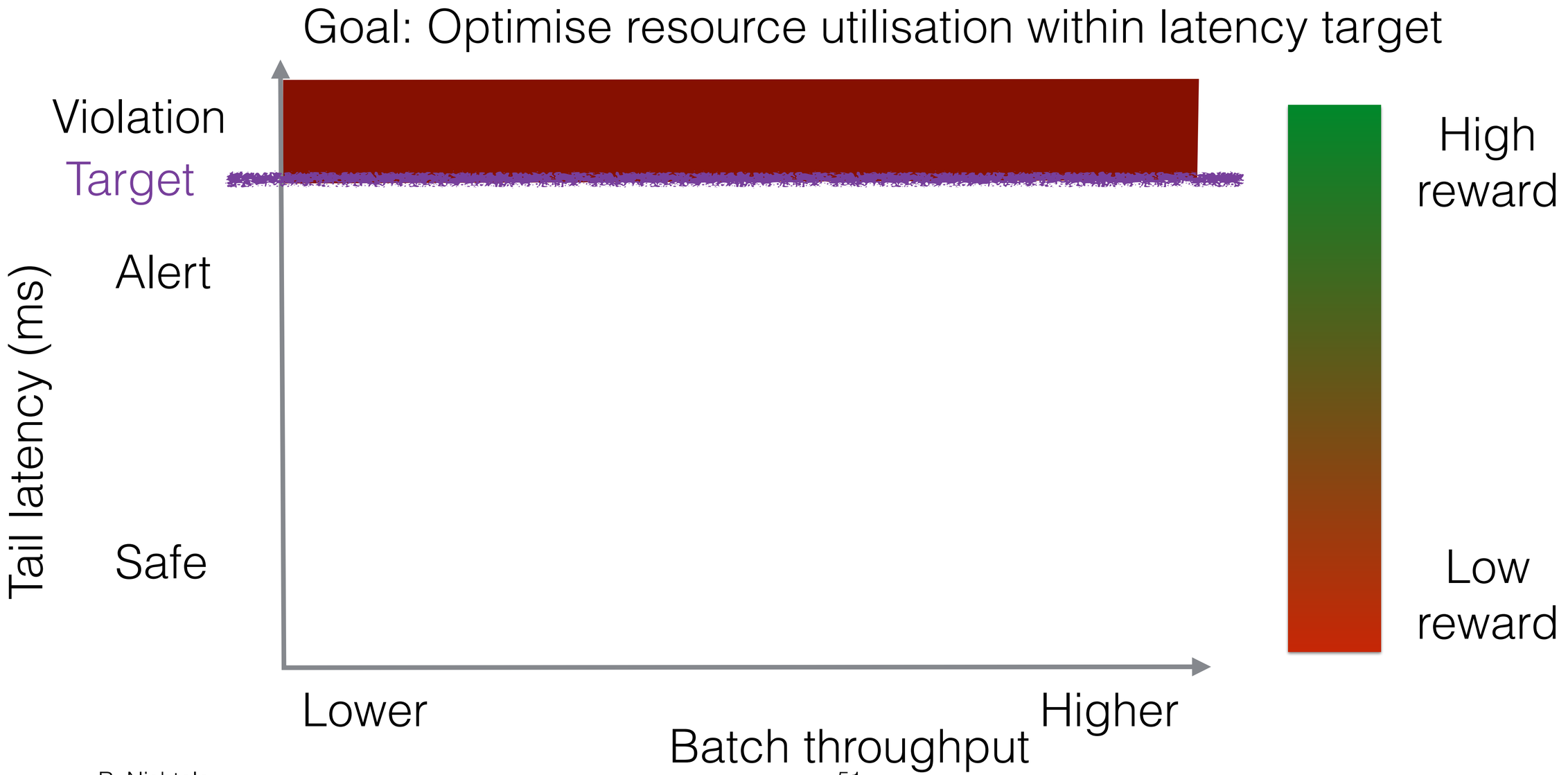


HipsterCo's Reward Mechanism

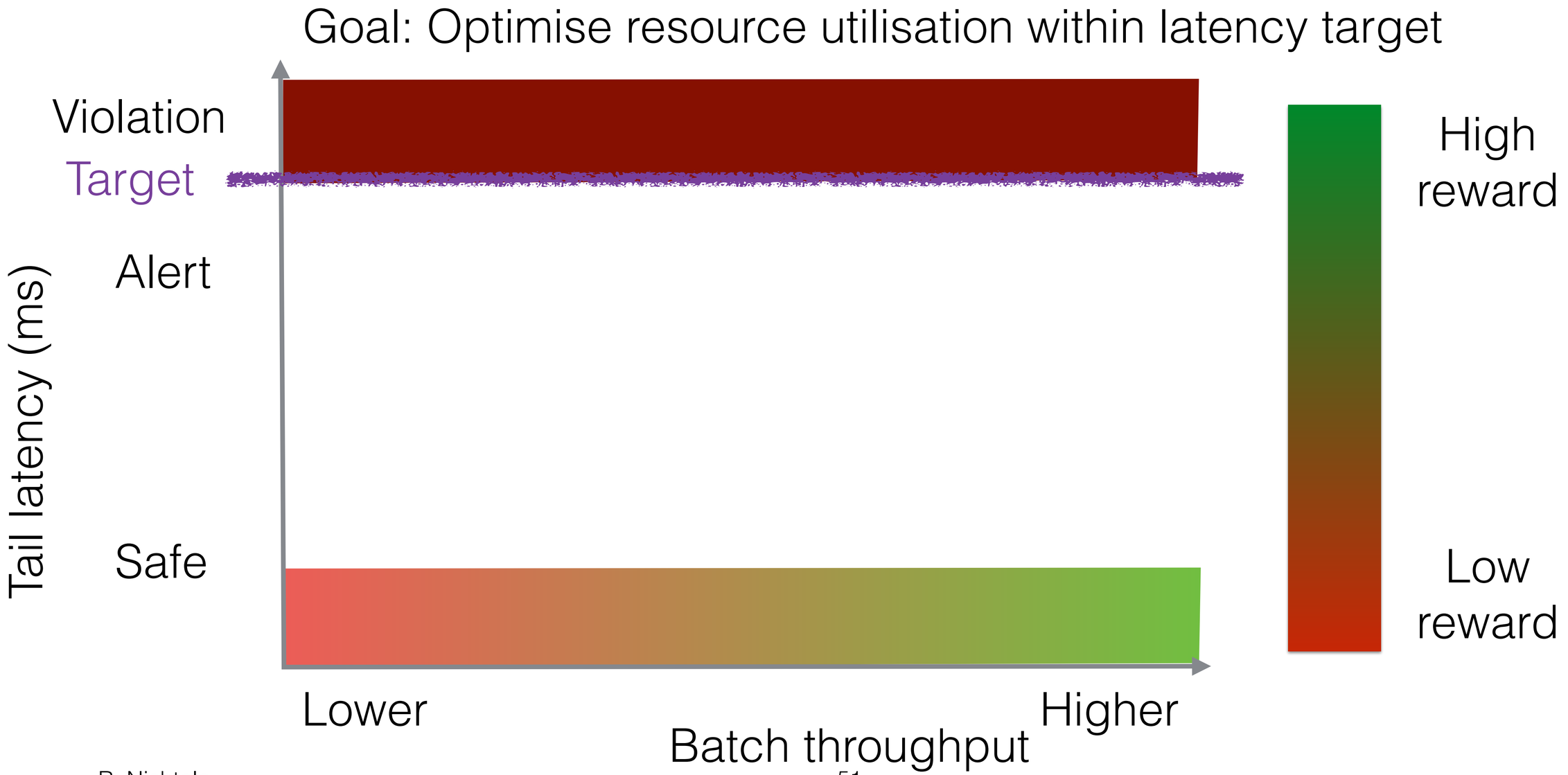
Goal: Optimise resource utilisation within latency target



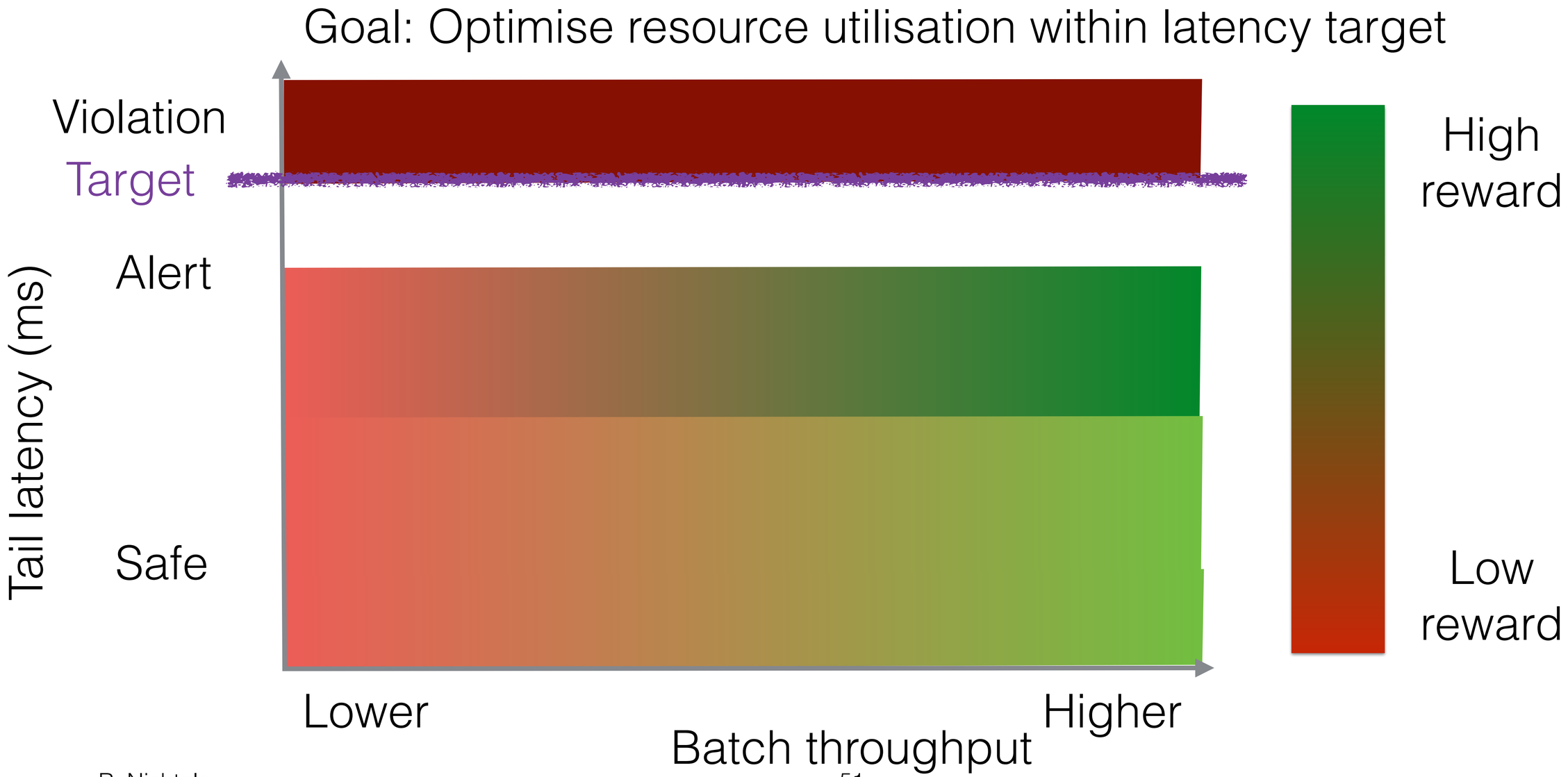
HipsterCo's Reward Mechanism



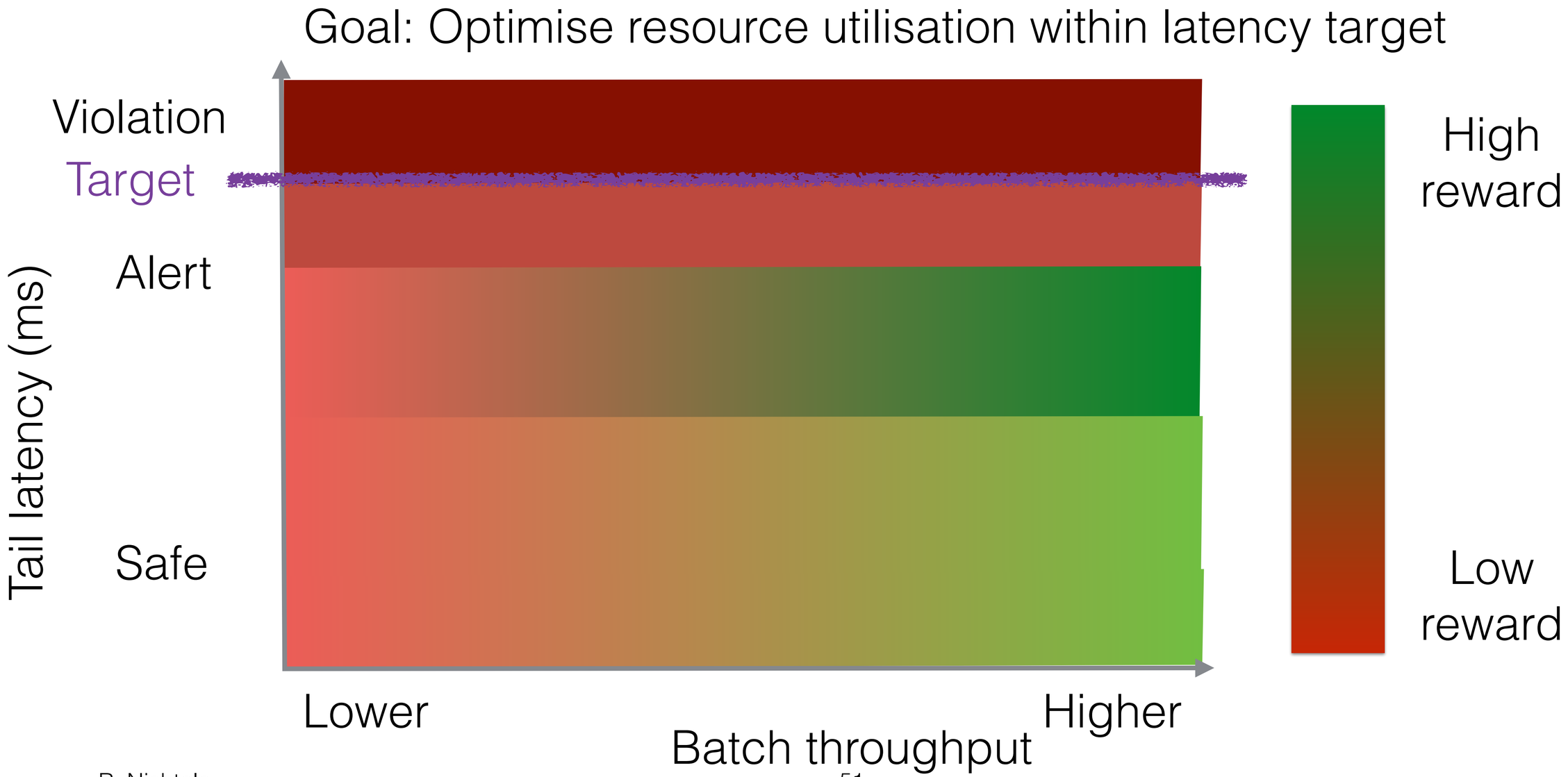
HipsterCo's Reward Mechanism



HipsterCo's Reward Mechanism

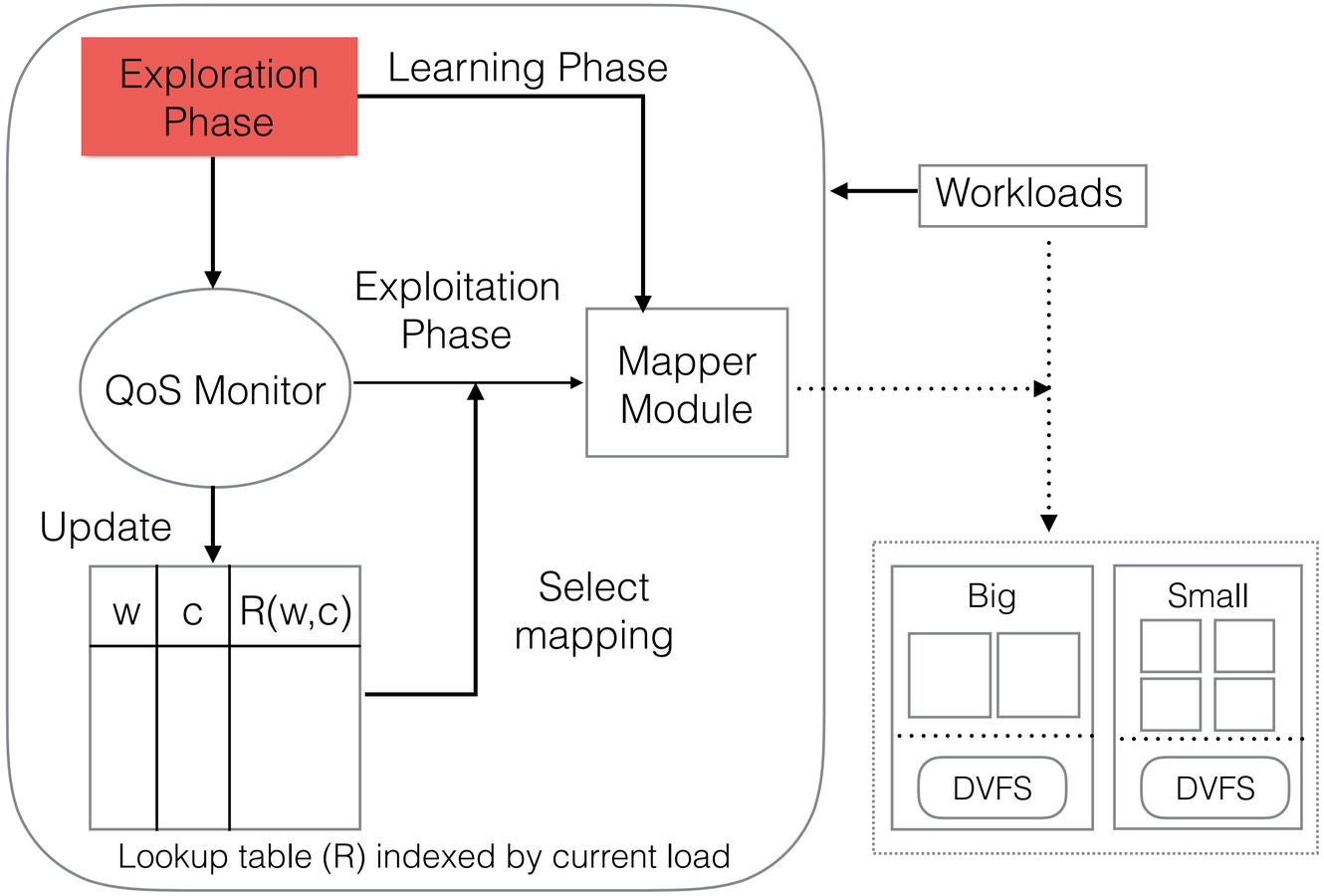


HipsterCo's Reward Mechanism



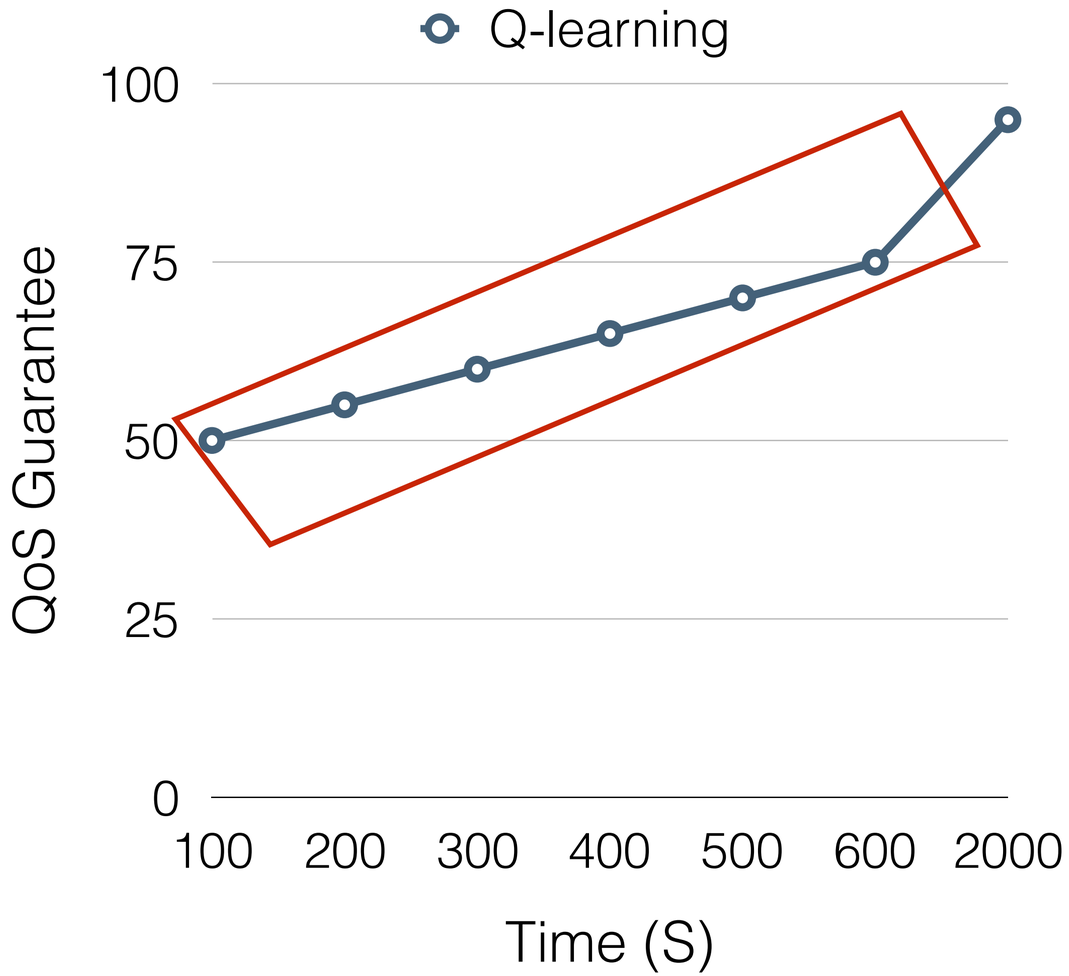
Traditional RL Approach

- **QoS Monitor:** App metric (load, latency, etc.,)
- **Exploration phase**: Random selection of actions
- **Exploitation phase:** Lookup table of Hipster's reward
- **Mapper module:** Task-to-core actuator



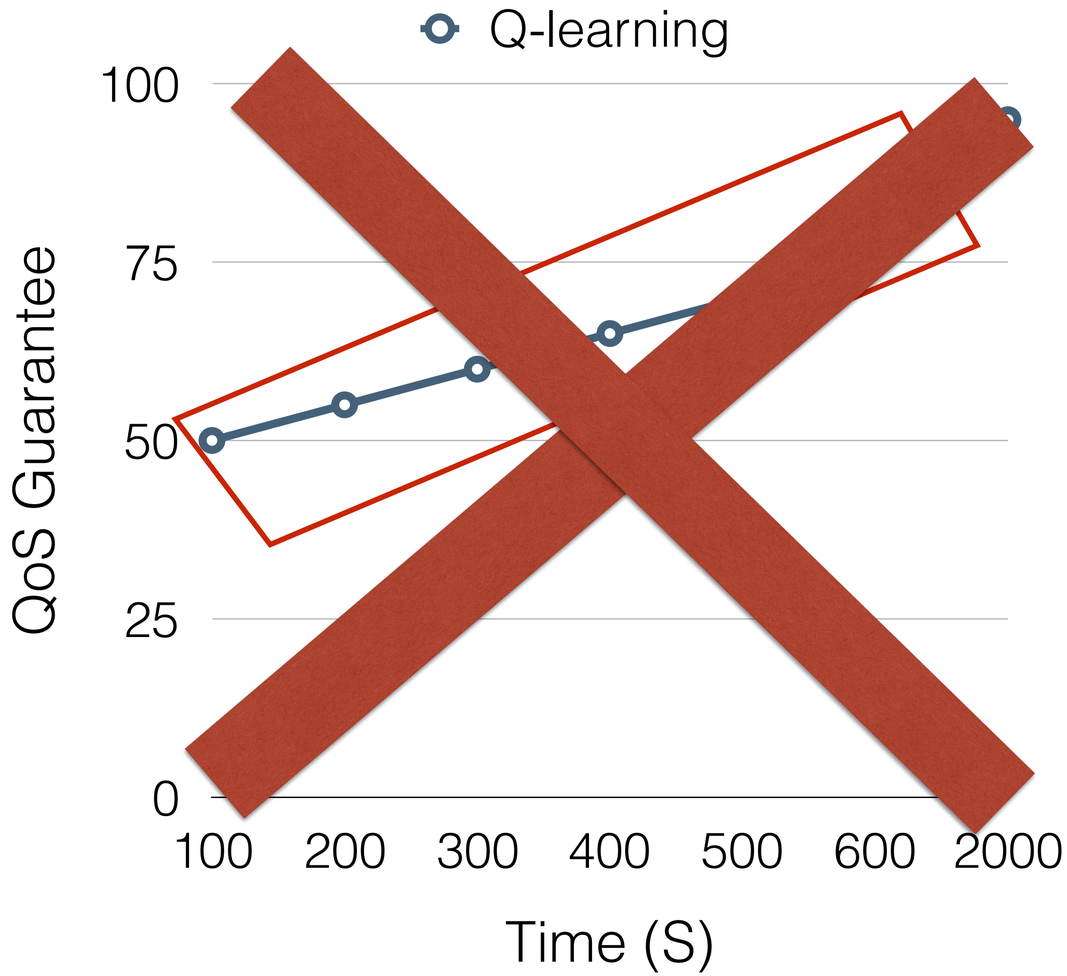
Lots of violations with pure RL

- Exploration-exploitation dilemma (**Q-learning**)
  - Exploration phase generates many random decisions leading to QoS violation
  - Exploration time increases with number of state-action pairs



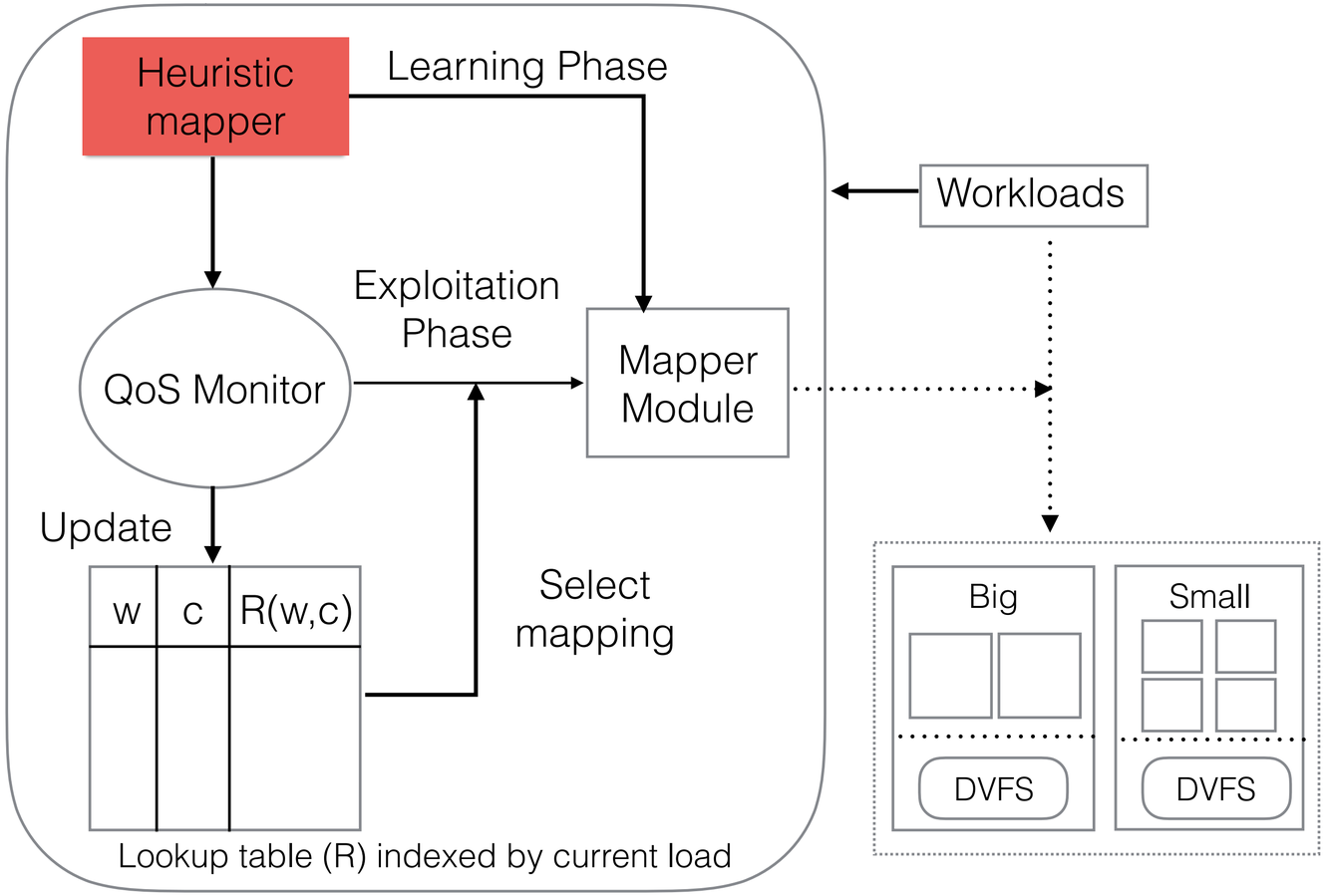
Lots of violations with pure RL

- Exploration-exploitation dilemma (**Q-learning**)
  - Exploration phase generates many random decisions leading to QoS violation
  - Exploration time increases with number of state-action pairs



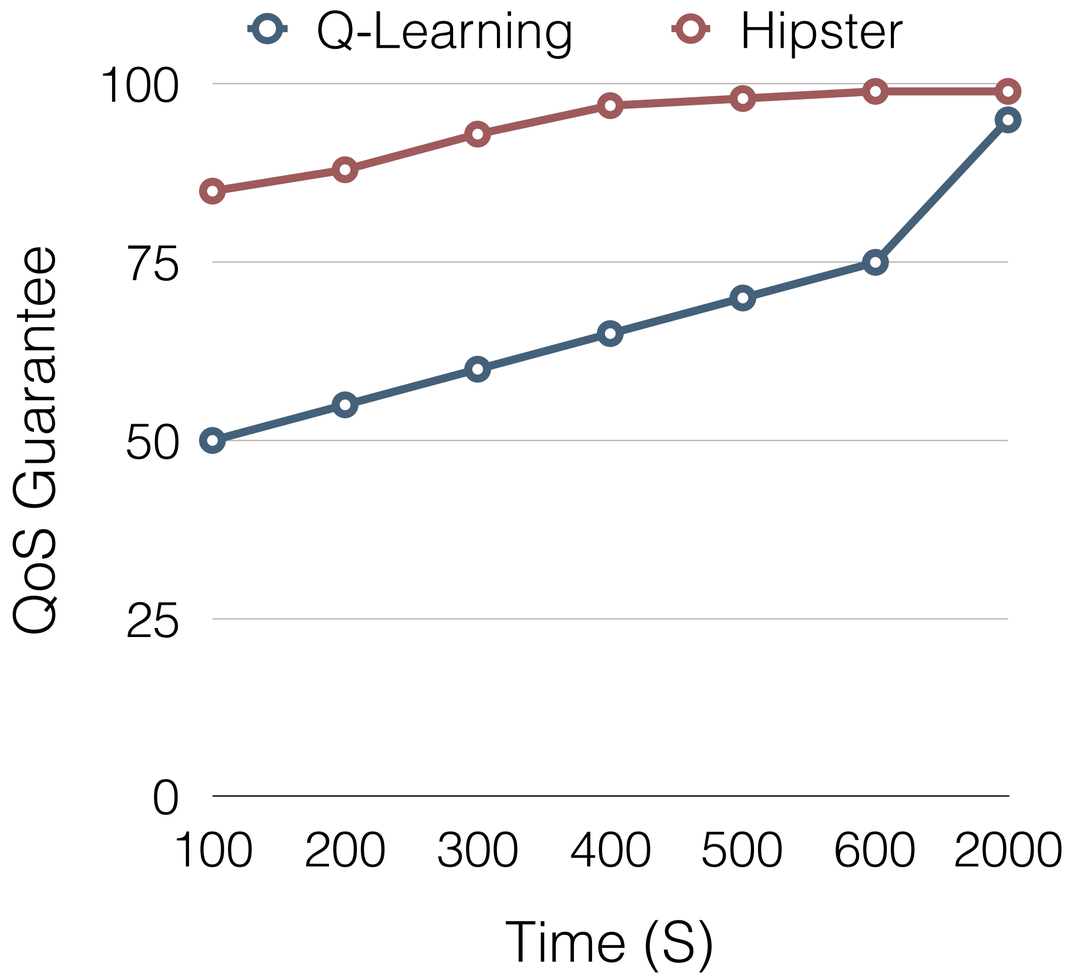
Hipster's Approach

- **QoS Monitor:** Application metric (load, latency, etc.,)
- **Heuristic mapper:** A static-oracle based on heuristics to speed up learning
- **Exploitation phase:** Lookup table of Hipster's reward
- **Mapper module:** Task-to-core mapping actuator

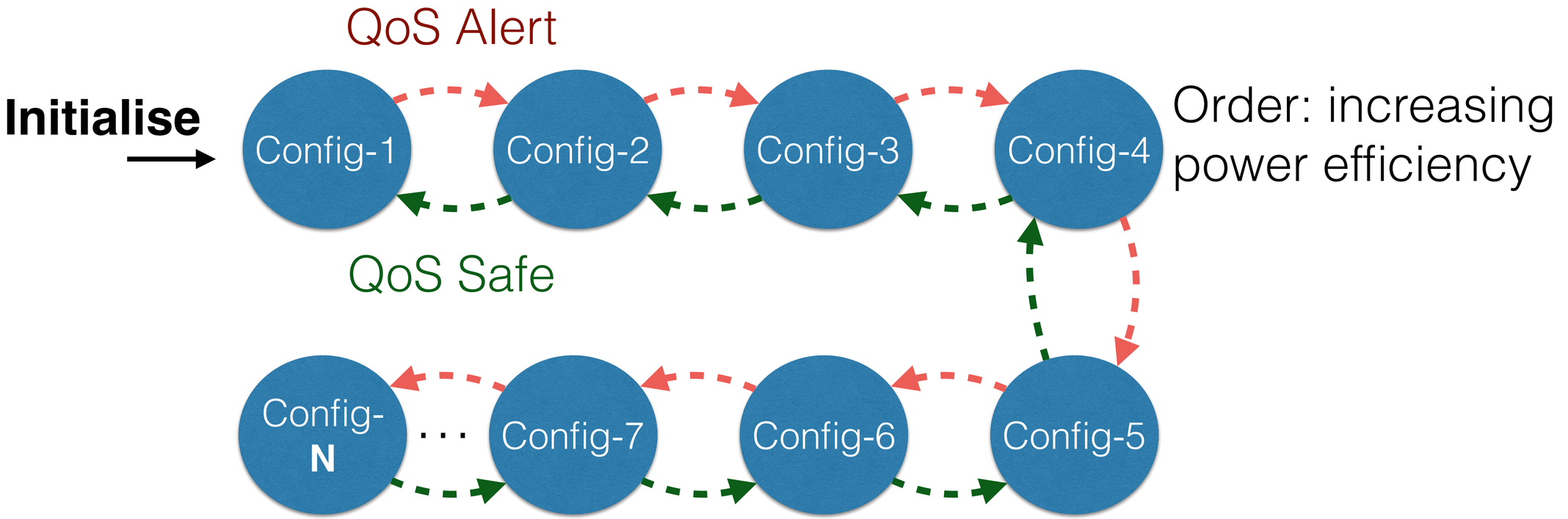


Hipster's Approach

- Improve learning phase with a heuristic
  - Reduces excessive violations
  - Speeds up learning phase
  - Not trying configurations that explicitly violate QoS



Previously proposed heuristic: Octopus-Man*

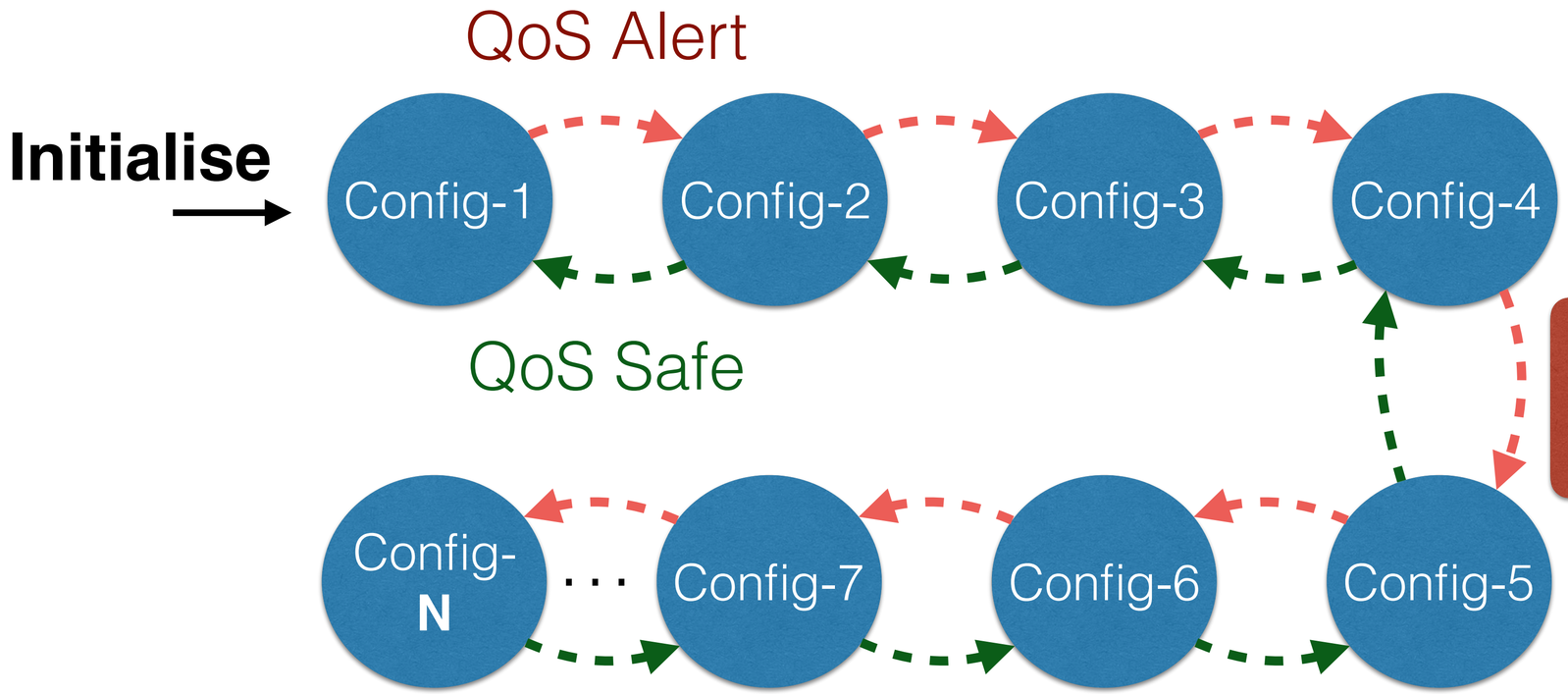


- QoS Alert: $QoS_{curr} > QoS_{target} * UP\_THR$
- QoS Safe: $QoS_{curr} < QoS_{target} * DOWN\_THR$

*Petrucci *et al.* HPCA'15

phdcomics.com/comics/archive.php?comicid=1906

Previously proposed heuristic: Octopus-Man*



Order: increasing power efficiency

Exclusive use of big or small cores at a fixed DVFS

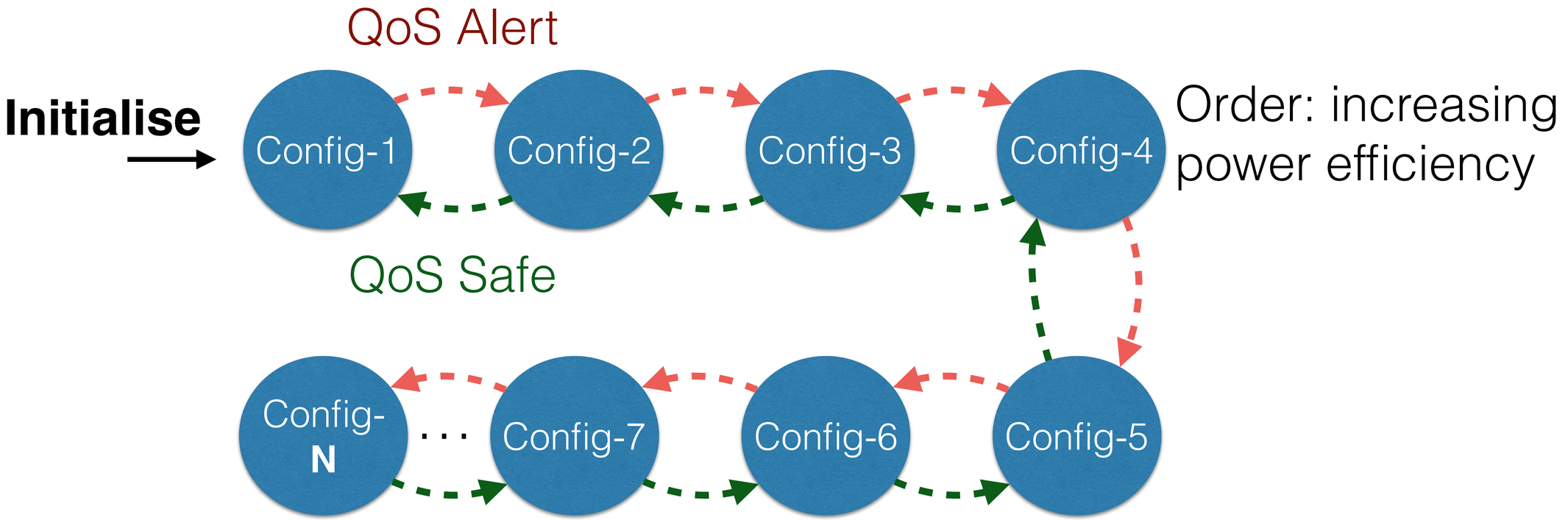
- QoS Alert: $QoS_{curr} > QoS_{target} * UP\_THR$
- QoS Safe: $QoS_{curr} < QoS_{target} * DOWN\_THR$

- **Pro:** Simplicity
- **Con:** reaction time ≈ # of config

*Petrucci *et al.* HPCA'15

phdcomics.com/comics/archive.php?comicid=1906

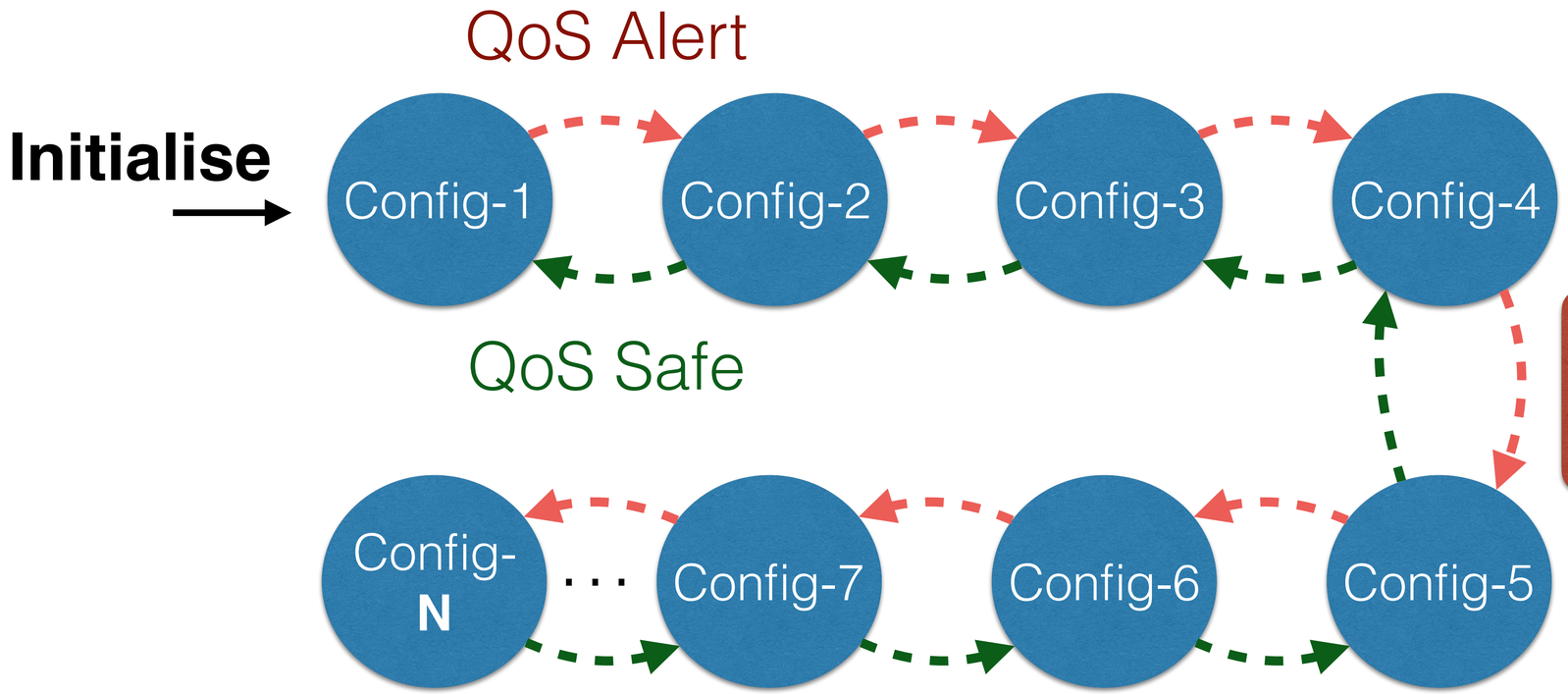
Hipster's Heuristic



- QoS Alert: $QoS_{curr} > QoS_{target} * UP\_THR$
- QoS Safe: $QoS_{curr} < QoS_{target} * DOWN\_THR$

phdcomics.com/comics/archive.php?comicid=1906

Hipster's Heuristic



Order: increasing power efficiency

All core and DVFS combinations

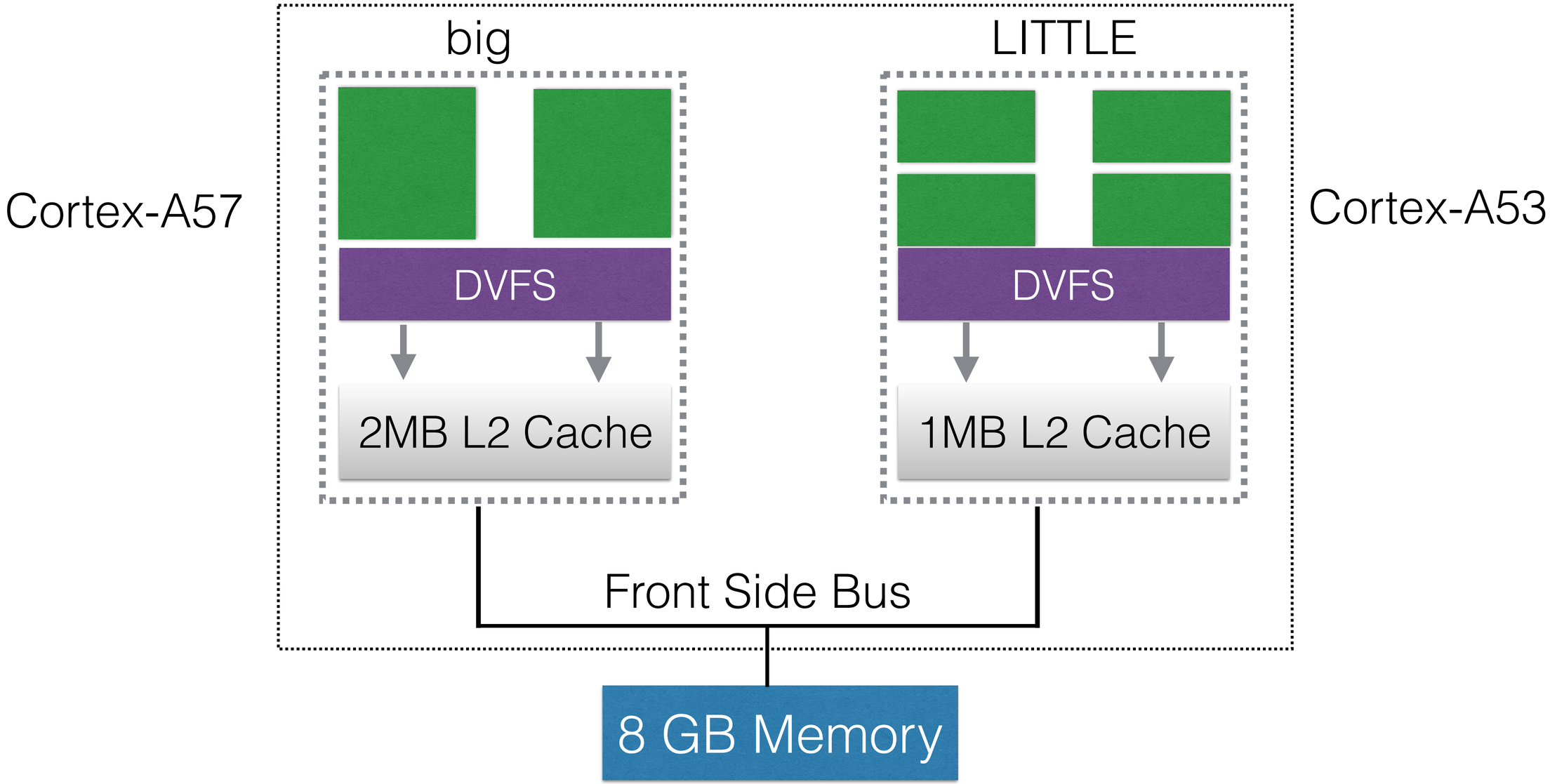
- QoS Alert: $QoS_{curr} > QoS_{target} * UP\_THR$
- QoS Safe: $QoS_{curr} < QoS_{target} * DOWN\_THR$

- **Pro:** Simplicity
- **Con:** reaction time ≈ # of config

Improves over Octopus-Man with fine-grained adaptations

phdcomics.com/comics/archive.php?comicid=1906

Experimental Setup: ARM Juno R1



Workloads

- Latency-critical workload
  - **Memcached**: In-memory key-value store for data caching
    - *QoS:*10 ms at the 95th percentile tail latency
  - **Web-Search** (Elasticsearch): Search engine
    - *QoS:* 500 ms at the 90th percentile tail latency
- Batch workloads: **SPEC** workloads
  - # of running instances = # of cores not used by interactive workload

Evaluation Metric

- **Two metrics** to quantify QoS of latency-critical workloads
  - **QoS guarantee:** Percentage of samples for which the measured QoS did not violate the target (100%-QoS violations%)
  - **QoS tardiness:** How intense a violation was ($QoS_{current}/QoS_{target}$)

- The performance of batch workloads is quantified as Instructions per Second

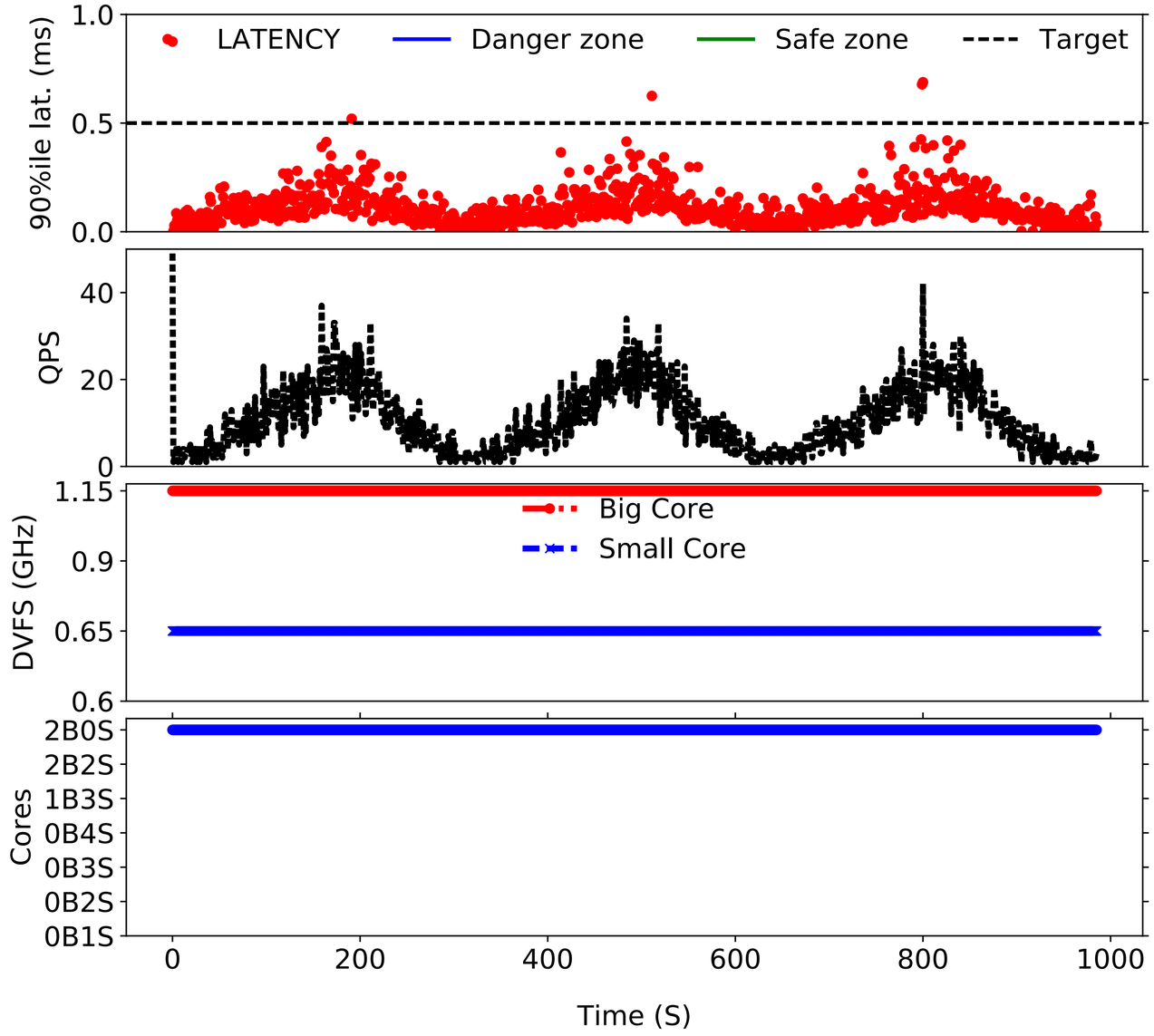
Baseline: **Static** all big cores (Web-Search)

QoS

Load

DVFS

Core combination



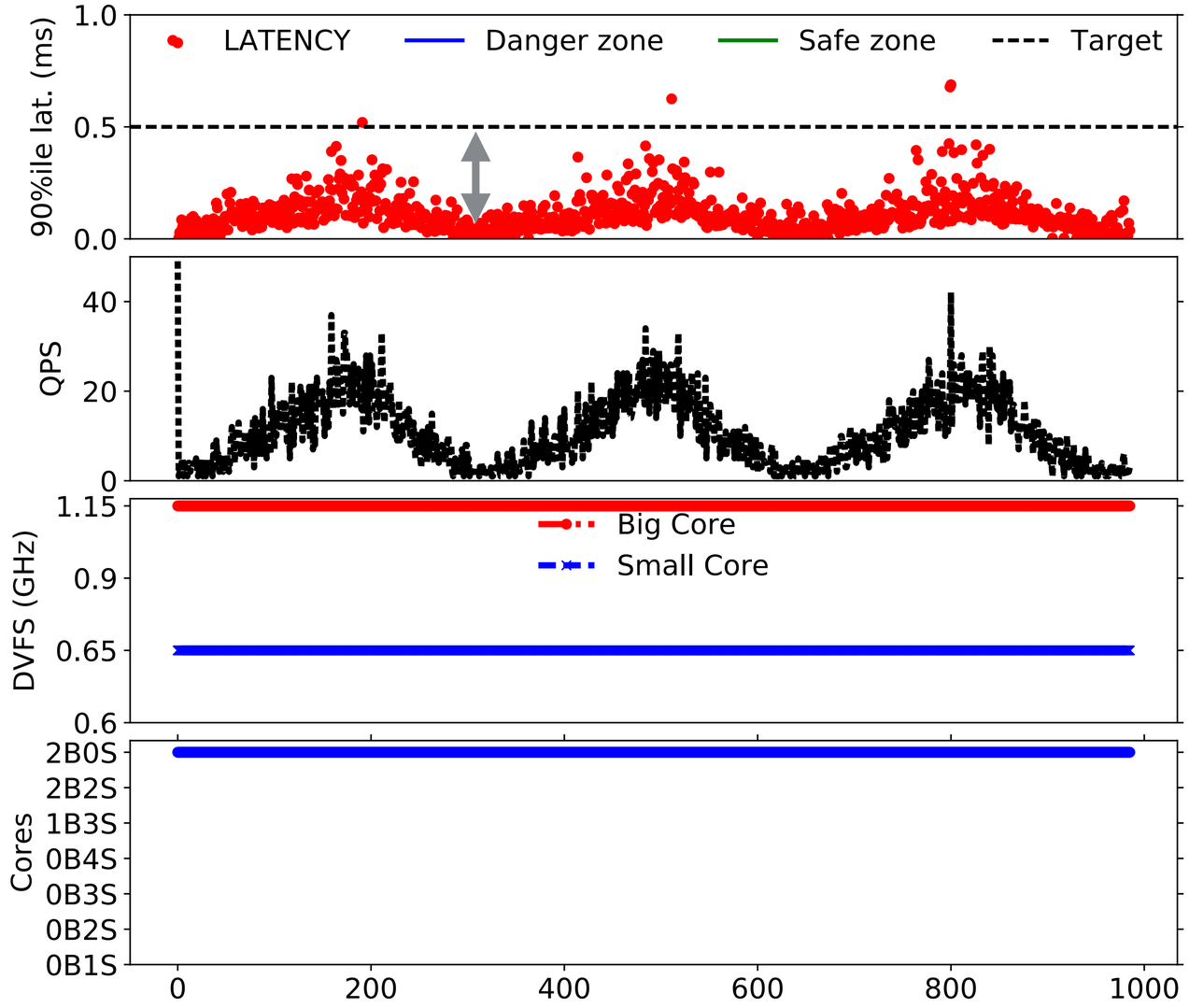
Baseline: **Static** all big cores (Web-Search)

QoS

Load

DVFS

Core combination



Latency slack!

Shows an opportunity to “just meet” QoS and reduce power

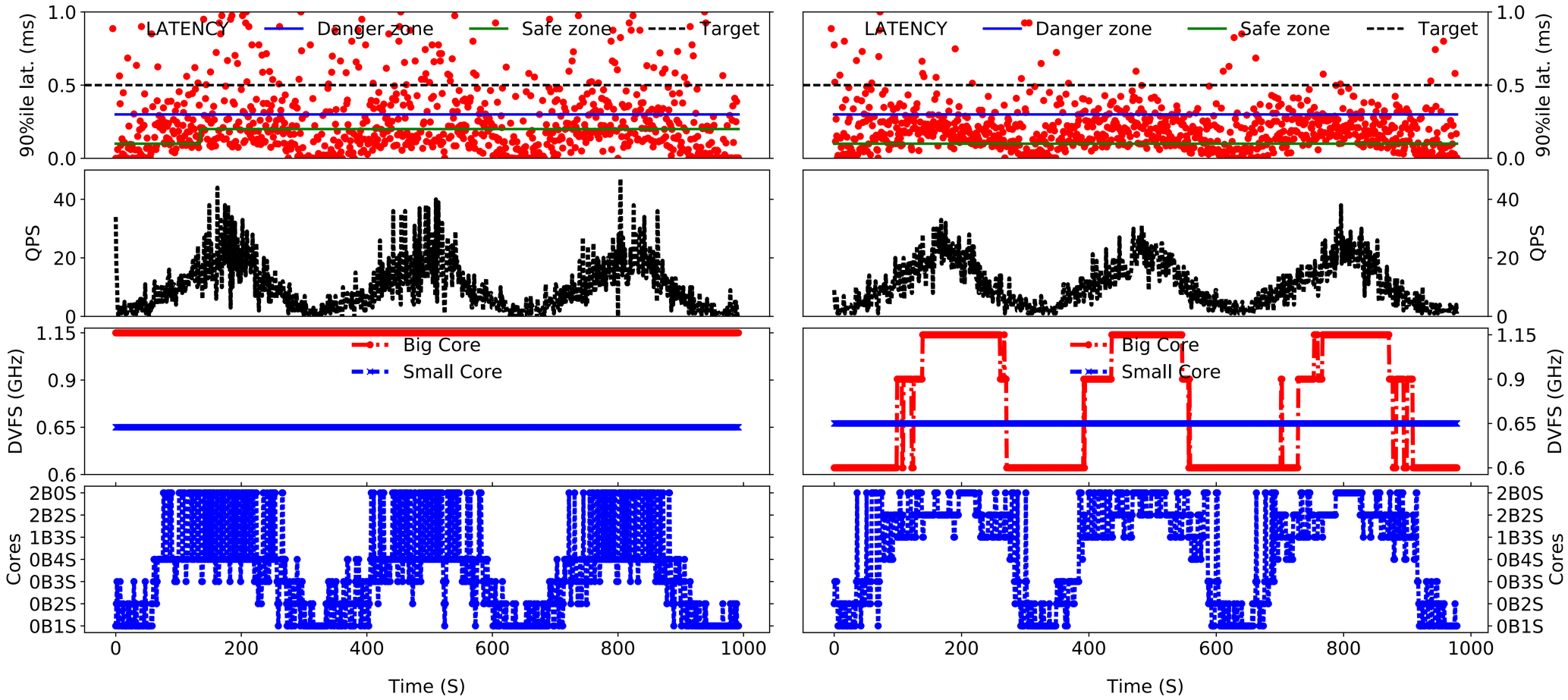
Octopus-Man vs Hipster Heuristic (Web-Search)

QoS

Load

DVFS

Core combination



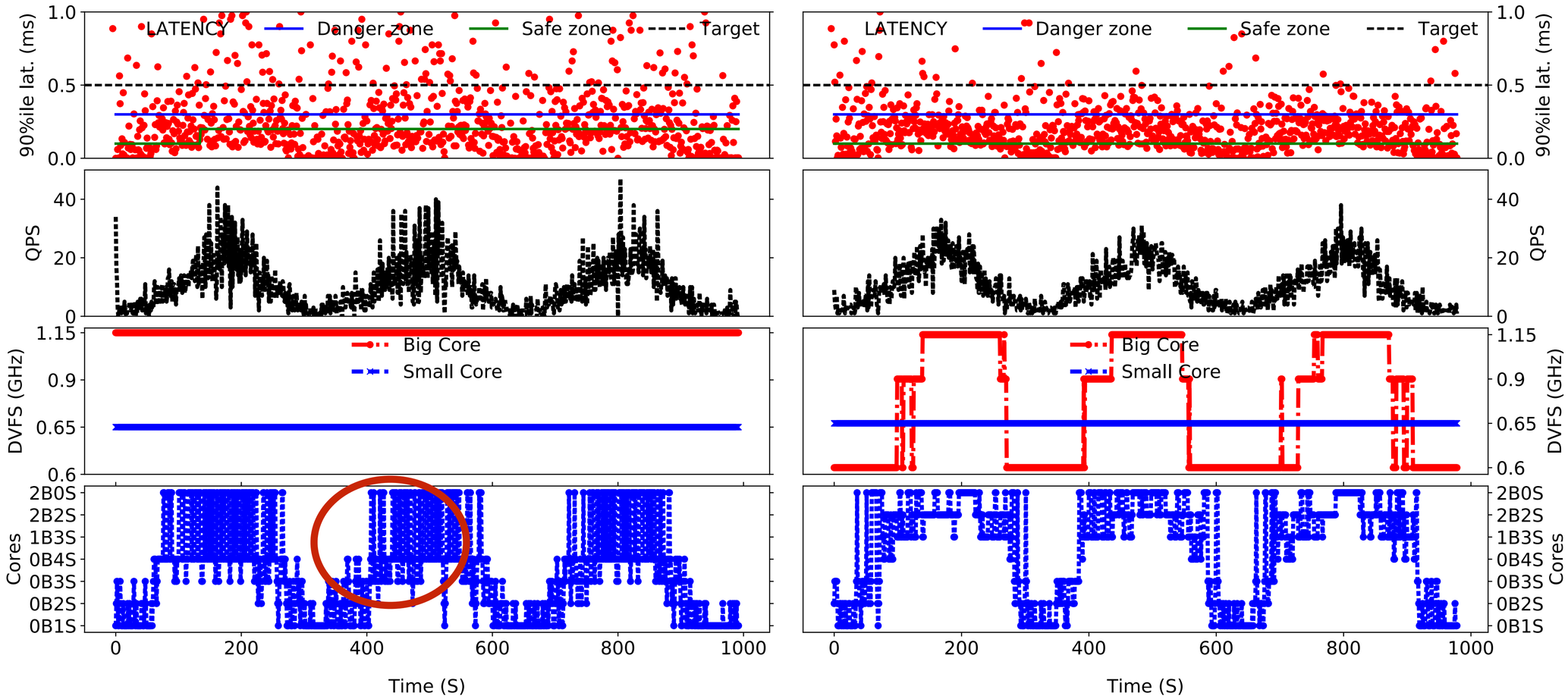
Octopus-Man vs Hipster Heuristic (Web-Search)

QoS

Load

DVFS

Core combination



Too many oscillations (across sockets) and a static DVFS!

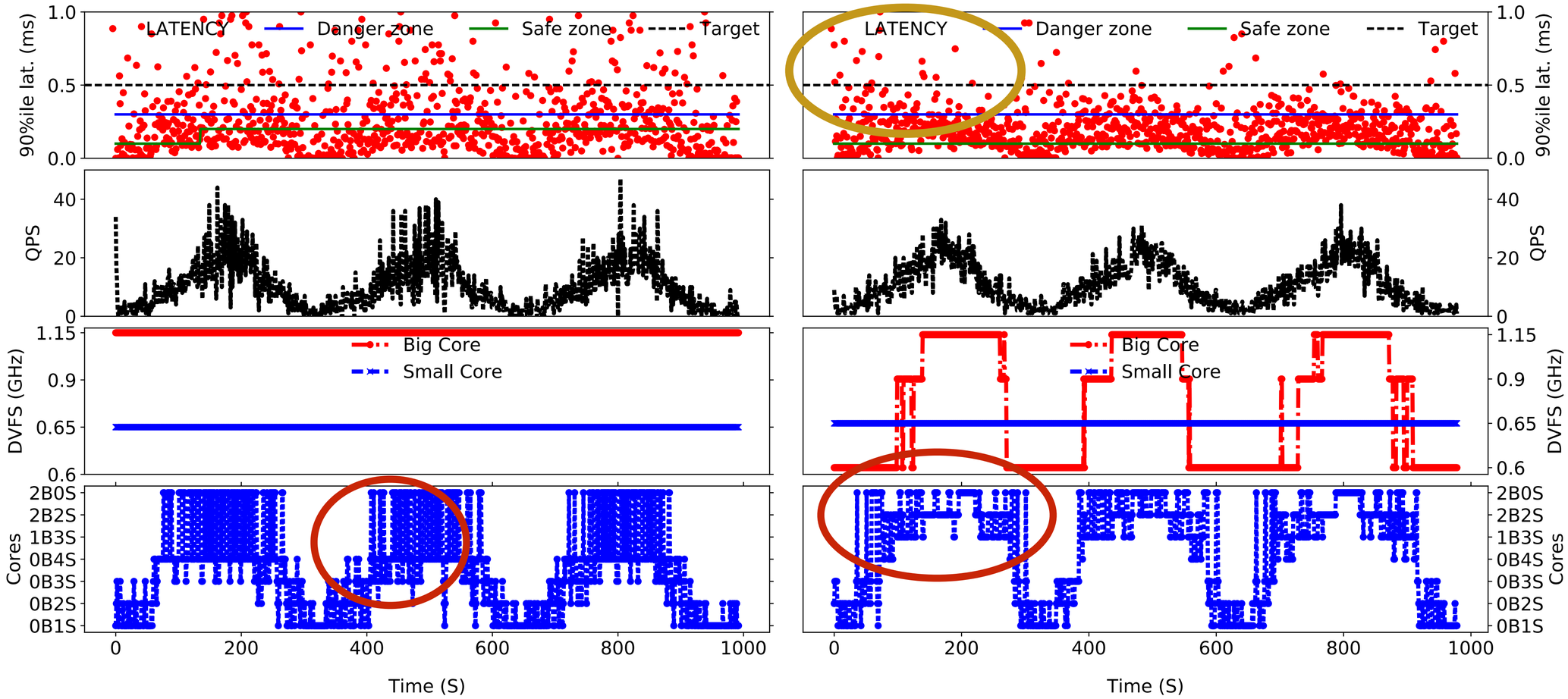
Octopus-Man vs Hipster Heuristic (Web-Search)

QoS

Load

DVFS

Core combination



Too many oscillations (across sockets) and a static DVFS!

Better, but still many violations!

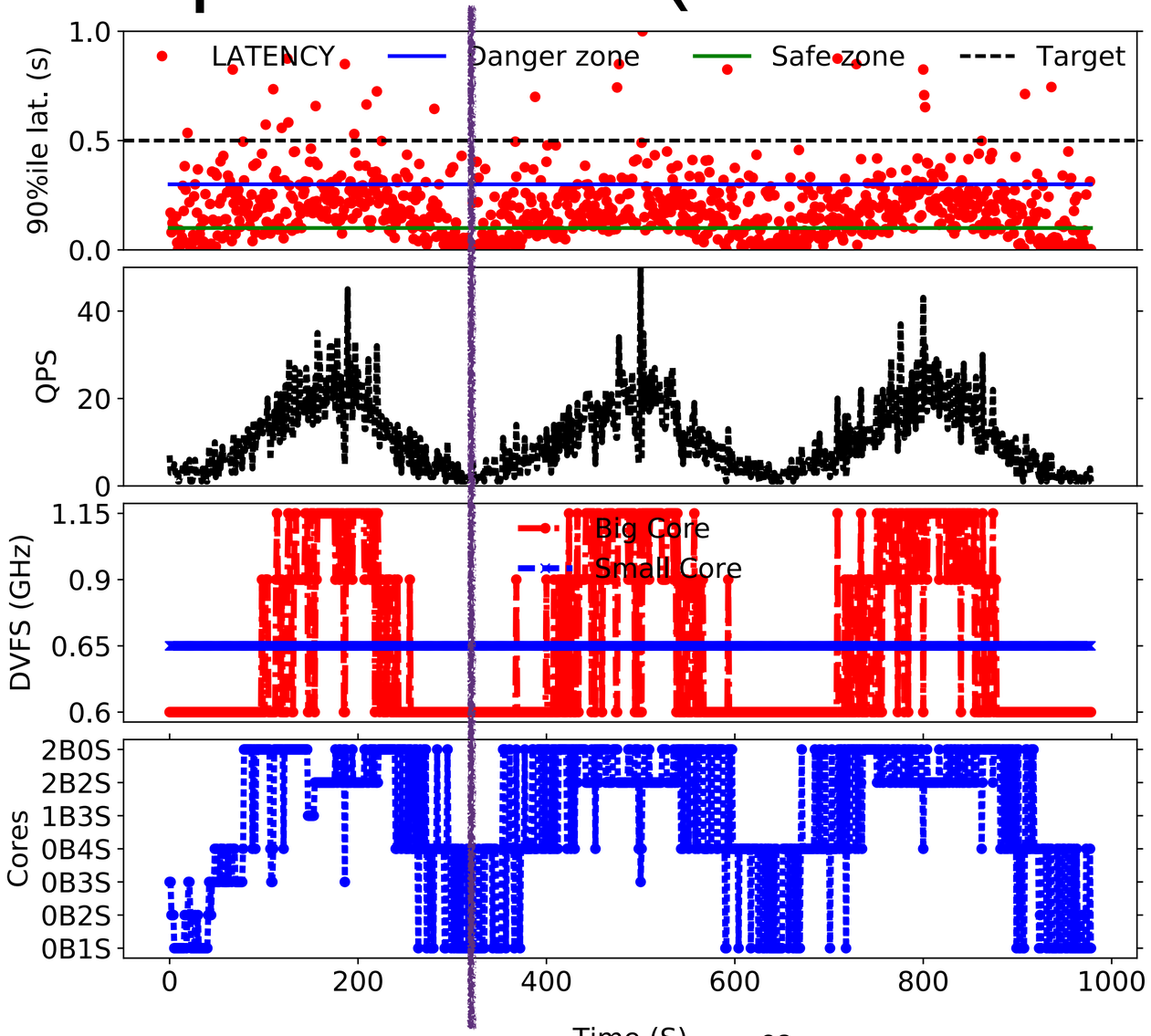
HipsterIn (Web-Search)

QoS

Load

DVFS

Core combination



Learning time: 300 s

DVFS oscillations are cheaper than core migrations
(Kasture *et al.* MICRO'15)

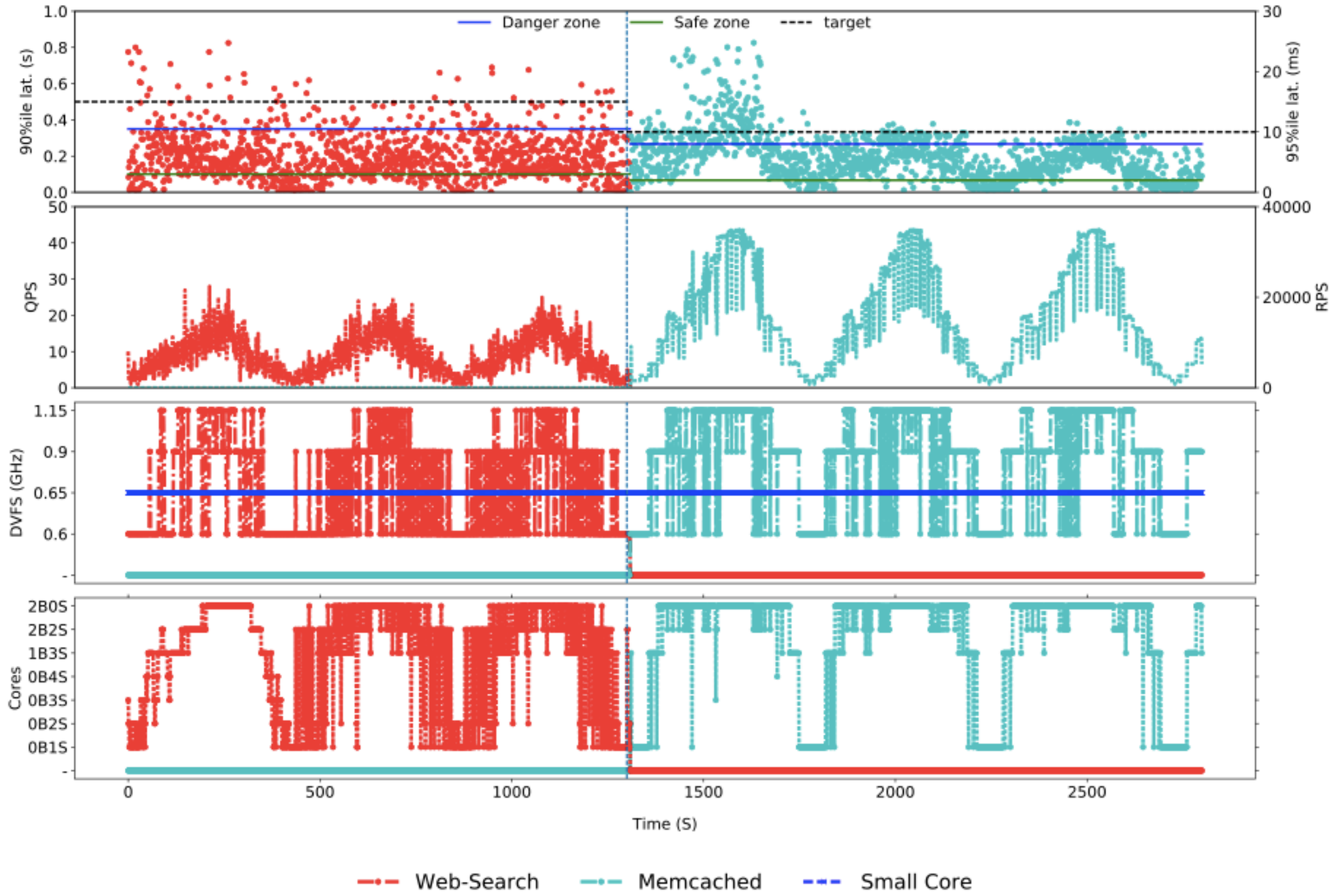
Web-Search to Memcached

QoS

Load

DVFS

Core combination



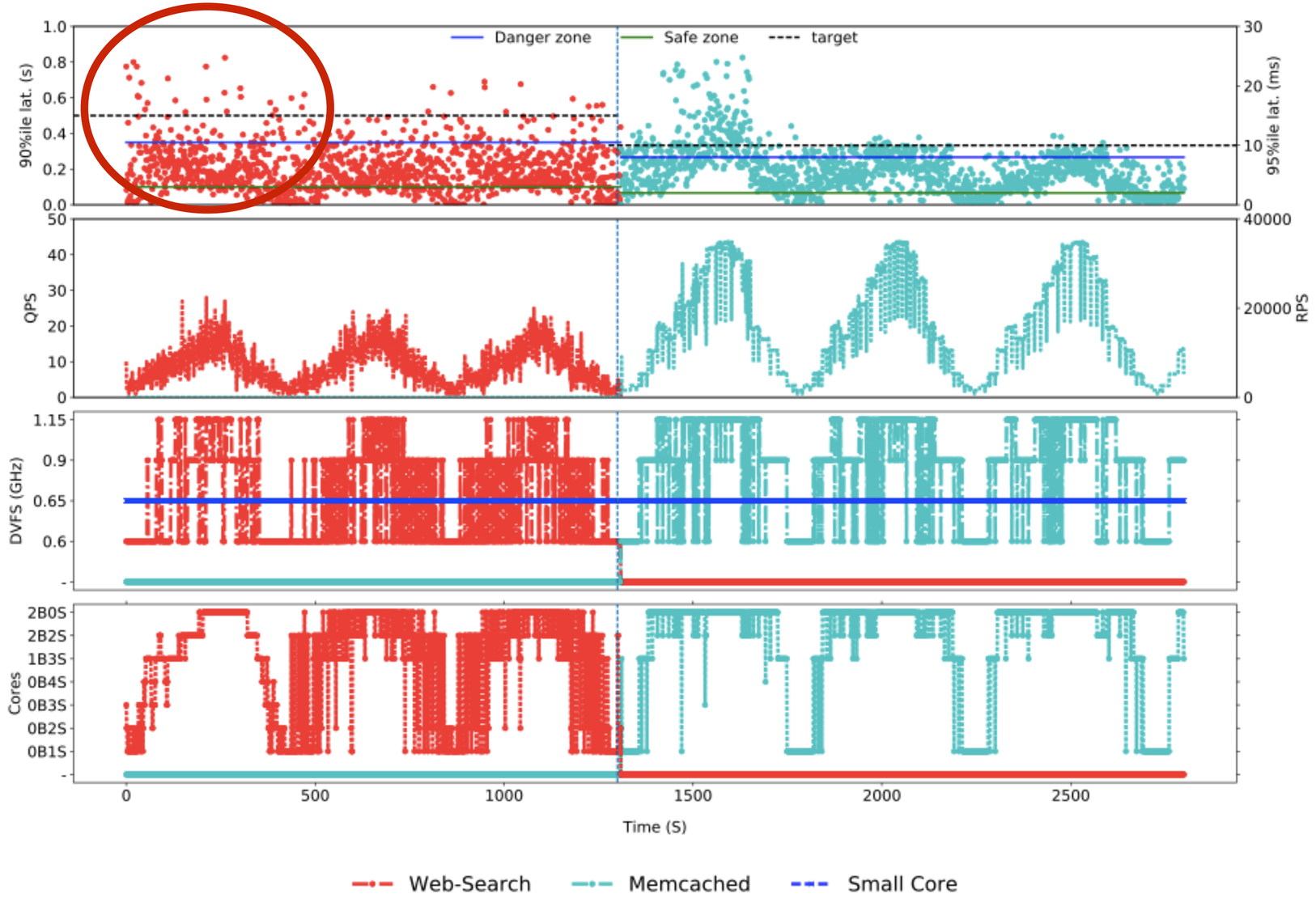
Web-Search to Memcached

QoS

Load

DVFS

Core combination



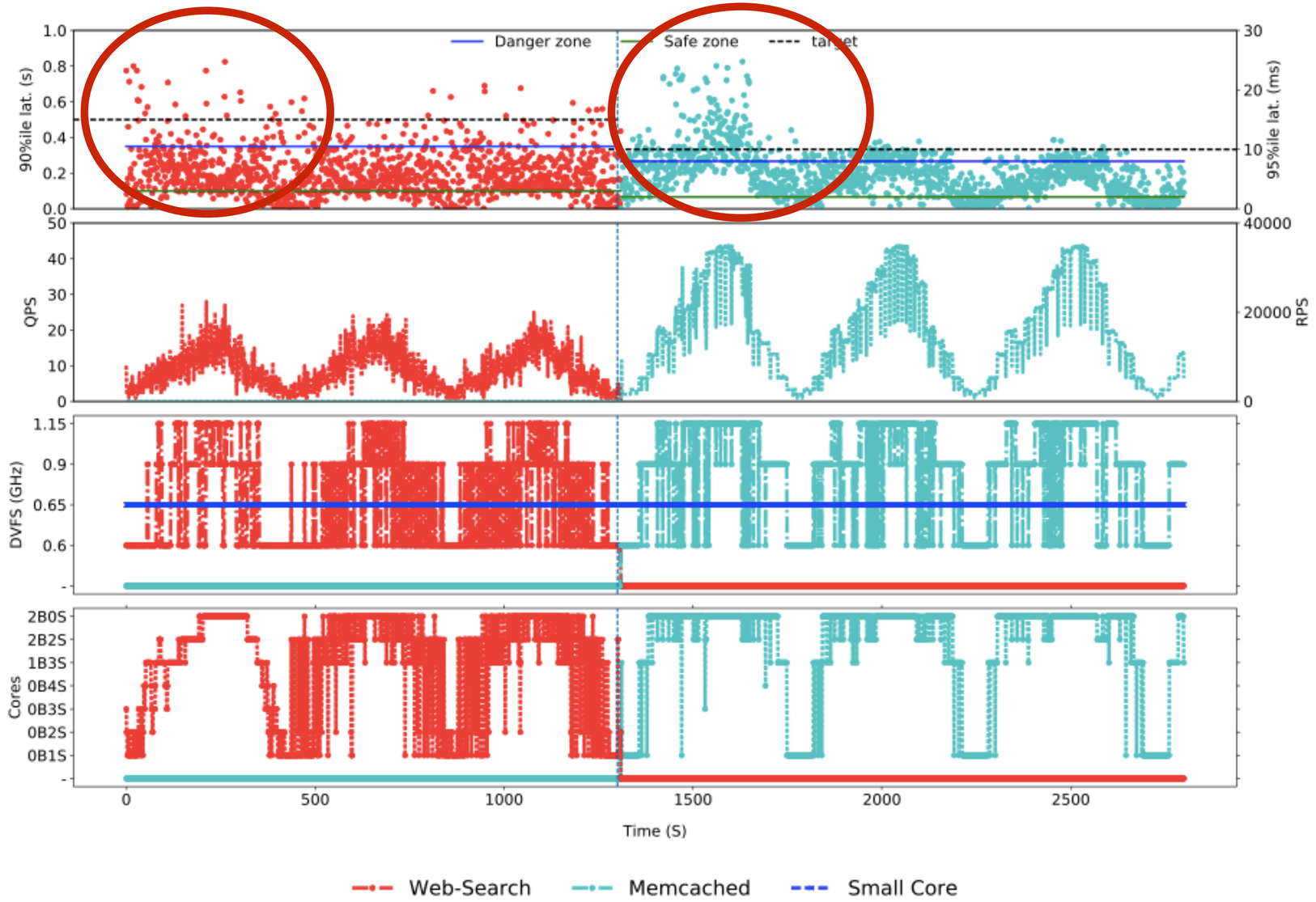
Web-Search to Memcached

QoS

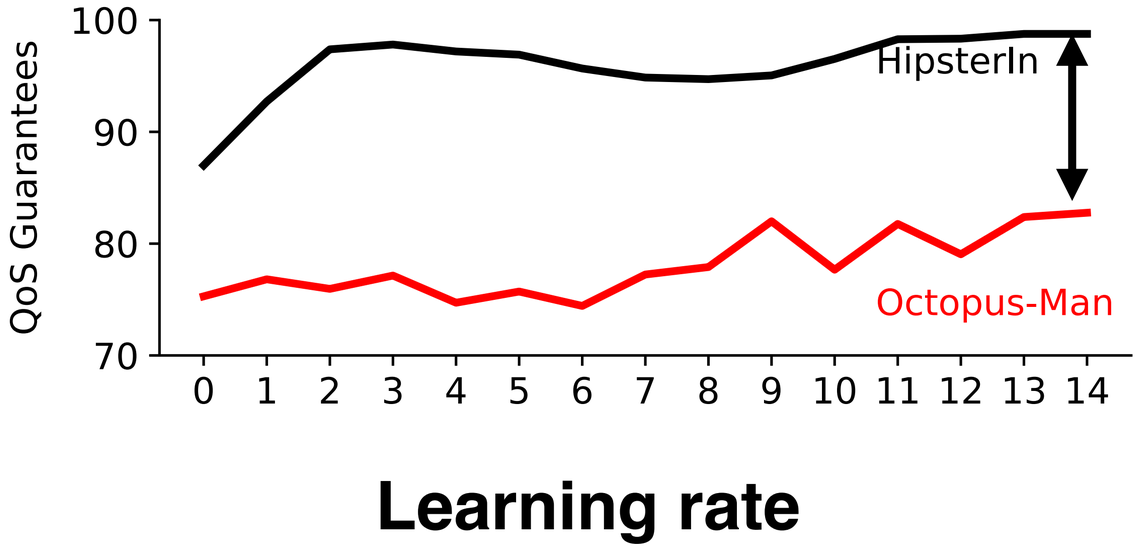
Load

DVFS

Core combination

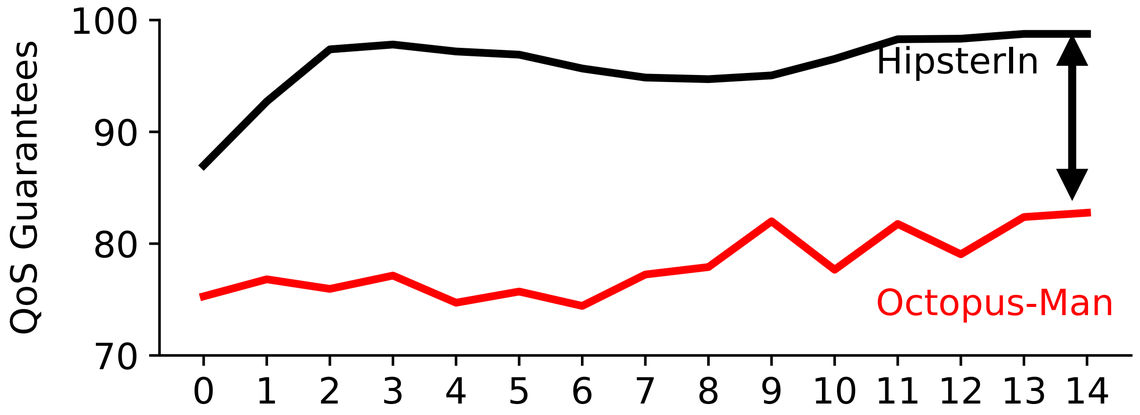


Evaluation

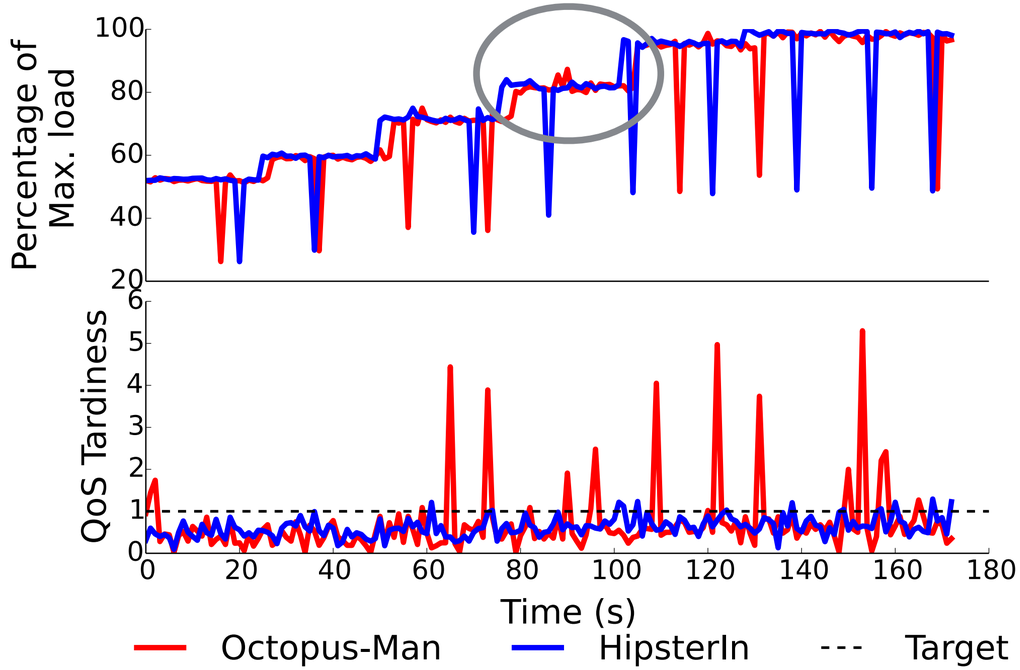


- Hipster**In** can learn quickly from past decisions
- Beats Octopus-Man by 20%

Evaluation



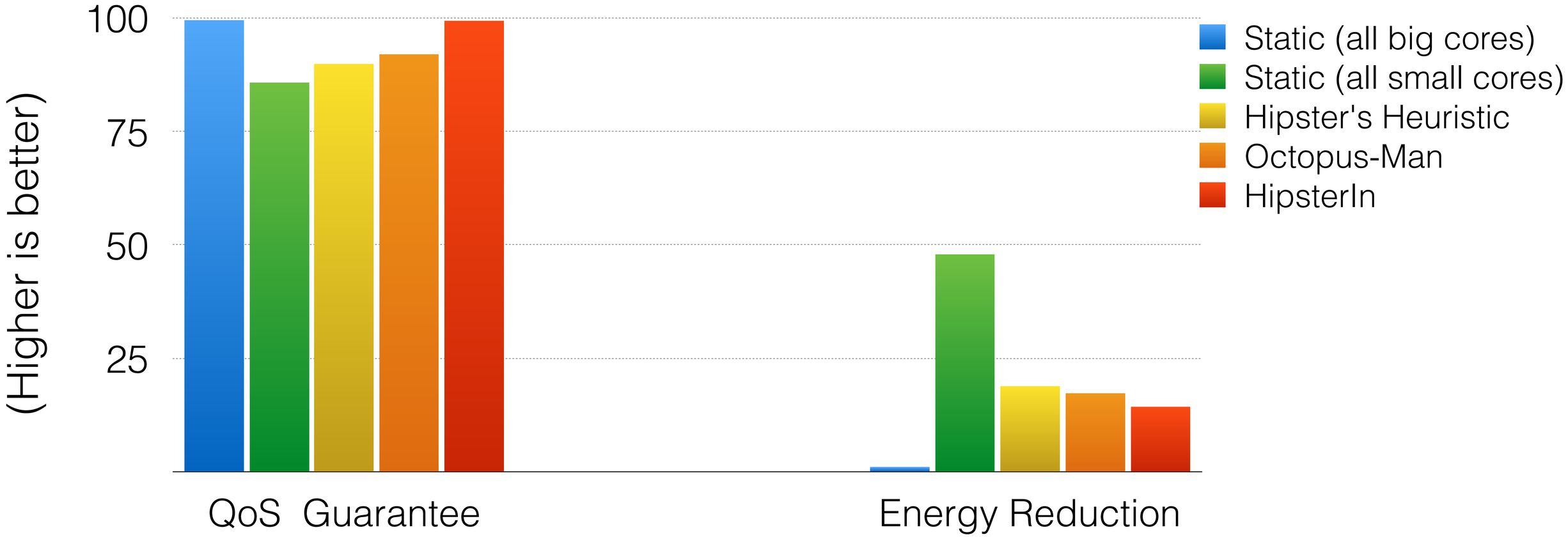
Learning rate



Responsiveness

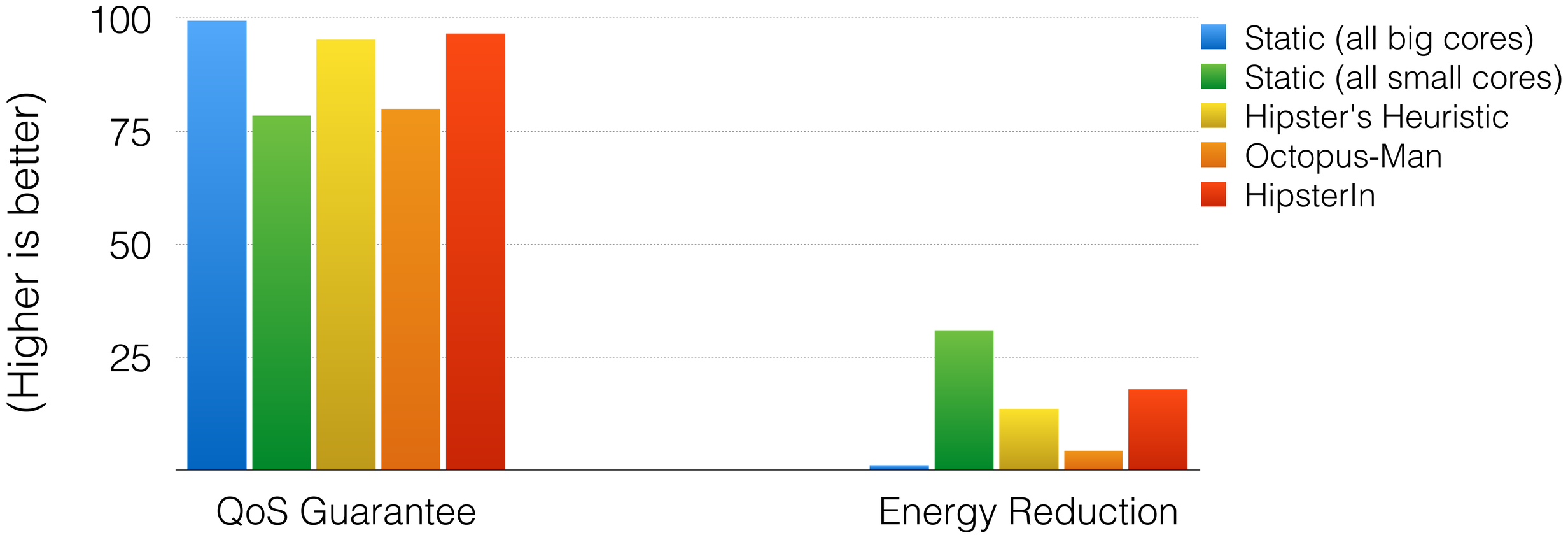
- Hipster**In** can learn quickly from past decisions
- Beats Octopus-Man by 20%
- Violation of Hipster**In** is 3.7X lower than Octopus-Man

Hipster**In** — Memcached



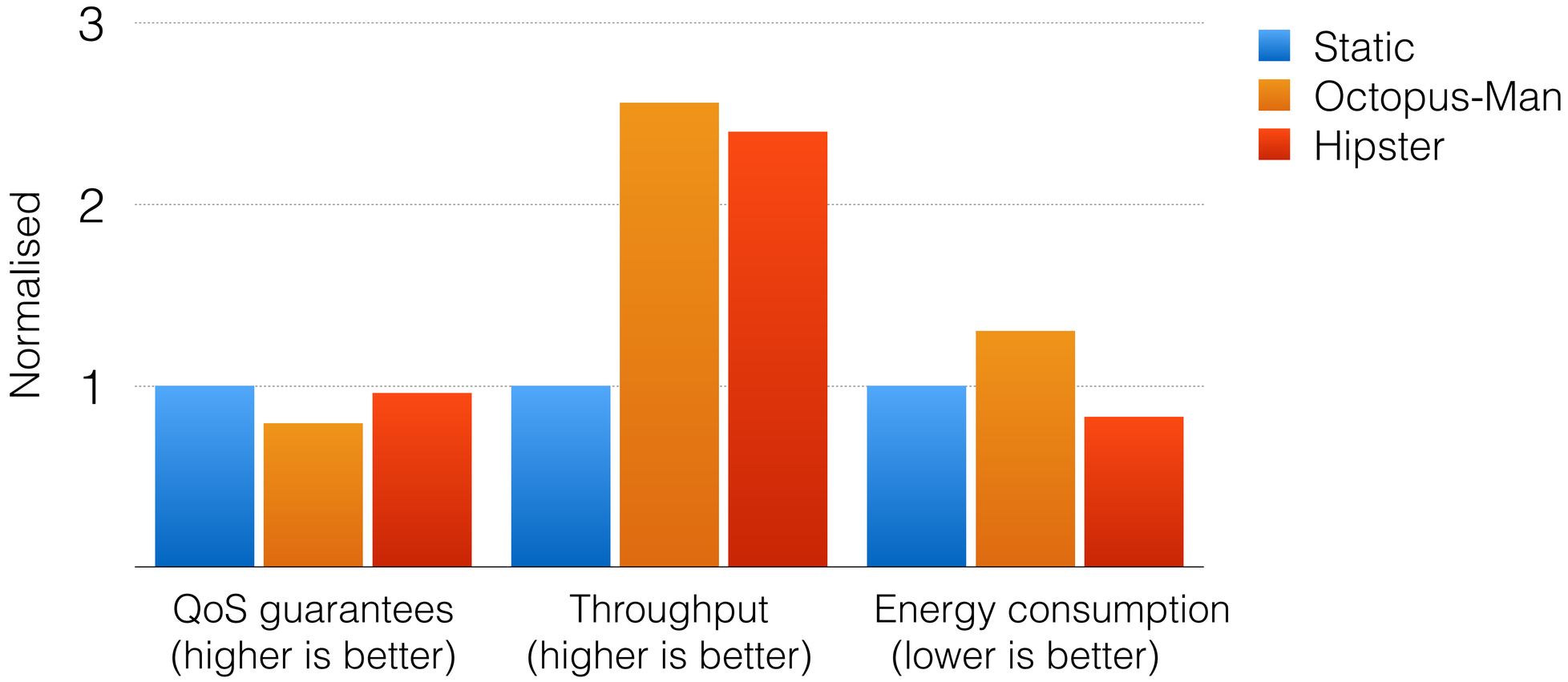
Beats Octopus-Man in meeting QoS and reduces energy

Hipster**In** — Web-Search



Improves QoS and reduces energy consumption

HipsterCo Summary



Better QoS guarantee + 230% more utilisation + Energy savings

Hipster Conclusion

- Hipster: Hybrid approach for heterogeneous workloads
- Improve energy efficiency and resource utilisation
- Evaluation on real heterogeneous platform (ARM Juno R1)
  - Web-Search and Memcached
- Energy improvement of up to 13% over all big cores
- Better QoS guarantee + 230% more utilisation + Energy savings

Talk Organisation

- Motivation
- Thesis contributions
- REPP: Runtime Estimation of Performance and Power
- Hipster: Hybrid Task Manager for Latency-Critical Cloud Workloads
- FUTURE WORK & CONTRIBUTIONS RECAP

Future work

- Explore multithreaded workloads for REPP
  - May use CPU utilisation as a good modelling parameter
- Satisfy thread level QoS guarantee for interactive workloads*
- Multi-node data centre type architecture
  - Could be a master-slave scheduler to deploy workloads
- Scale Hipster on a larger infrastructure like Cavium ThunderX*
  - Use (convolutional?) neural networks to scale and improve prediction accuracy

*work-in-progress

Contributions Recap

Contributions Recap

- **REPP**: Runtime Estimation of Performance and Power
  (R. Nishtala *et al.* ICPPW'15, SBAC-PAD'16, IGSC'16)
  - An accurate power and performance prediction approach
  - Architecture and application agnostic
  - Predicts with an accuracy greater than 93%

Contributions Recap

- **REPP**: Runtime Estimation of Performance and Power
  (R. Nishtala *et al.* ICPPW'15, SBAC-PAD'16, IGSC'16)
  - An accurate power and performance prediction approach
  - Architecture and application agnostic
  - Predicts with an accuracy greater than 93%

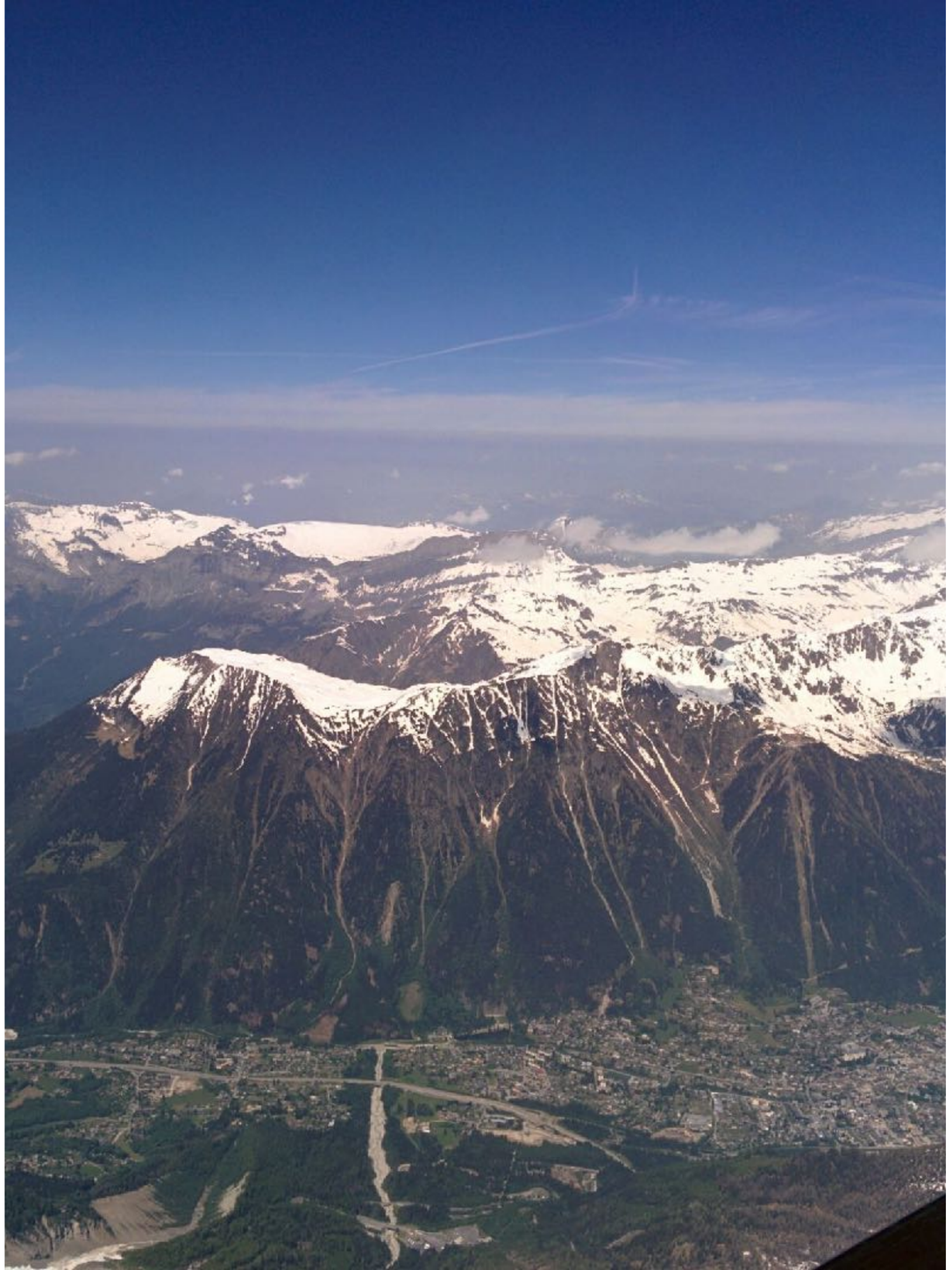
- **Hipster**: Hybrid Task Manager for Latency-Critical Clouds Workloads
  (R. Nishtala *et al.* HPCA'17, *ACM TOCS'17**)
  - Enables power savings and high utilisation while meeting QoS
  - Achieves 99% QoS with 13% energy savings and 230% more utilisation

*under review

Acknowledgement

- Thesis advisors: Xavier Martorell and Daniel Mossé
- Funding: FP7 Mont-Blanc 2, CAP 2015, MPEXPAR 2014
- Research group: Paul Carpenter, M. Gonzalez and V. Petrucci
- Family: Nishtala's
- Aleksandra Dokurno
- Friends

Thanks!



View from the *Vertical Kilometer* in Chamonix!

R. Nishtala

Backup Slides

Publications

1. **R. Nishtala**, M. González, X. Martorell, "*A Methodology to Build Models and Predict Performance-Power in CMPs*", ICPPW 2015
2. **R. Nishtala**, X. Martorell, V. Petrucci, D. Mossé, "*REPP-H: Runtime Estimation of Performance and Power on Heterogeneous Data Centers*", SBAC-PAD 2016
3. **R. Nishtala**, X. Martorell, "*REPP-C: Runtime Estimation of Performance and Power with Workload Consolidation in CMPs*", IGSC 2016
4. **R. Nishtala**, P. Carpenter, V. Petrucci, X. Martorell, "*Hipster: A Hybrid Task Manager for Latency-Critical Cloud Workloads*", HPCA 2017
5. **R. Nishtala**, P. Carpenter, V. Petrucci, X. Martorell, "*The Hipster approach for Improving Cloud System Efficiency*", ACM TOCS 2017 (under review, submitted in Feb-2017)
6. **R. Nishtala**, D. Mossé, V. Petrucci, "*Energy-aware thread co-location in Heterogeneous Multicore Processors*", EMSOFT 2013 (Not a part of this thesis)

REPP

Hardware counters

#	Component	Modeled Components
1	Front End	L1_ITLB, L1_ICACHE, FETCH_UNIT, PREDECODE, LSD, PREDECODE,uCODE ROM, SPT, uOP BUFFER,RAT, ROB
2	Integer	Integer arithmetic unit
3	Floating point	Floating point arithmetic unit
4	Branch Predictor	BPU and branch execution
5	L1 Cache	LD/ST execution, MOB, L1, L1 DTLB, L2 DTLB
6	L2 Cache	L2
7	Last Level Cache	LLC
8	Memory	Memory and Front Side Bus (FSB)

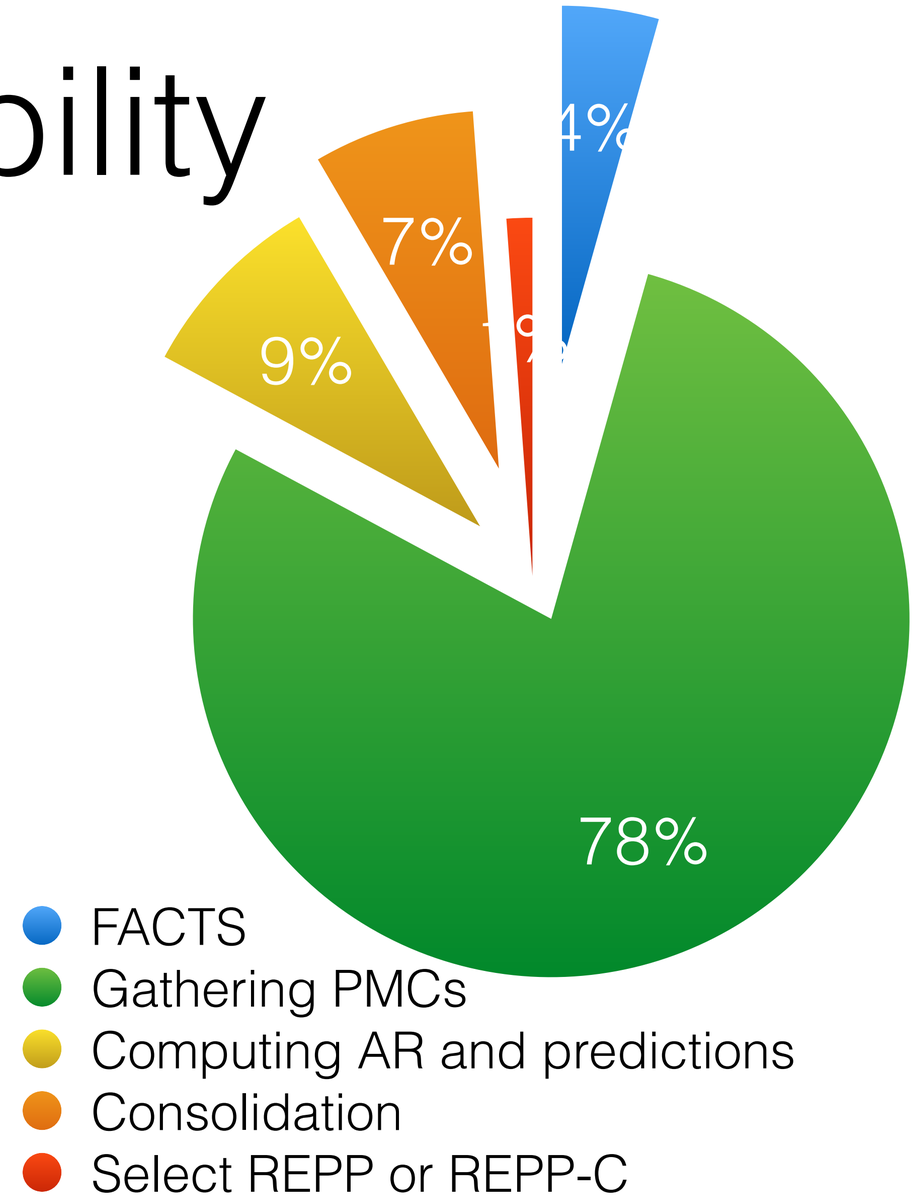
Activity in these components (uops/cycle) = Activity Ratio (AR)

Modelling

- **Power Modelling** is based on PMCs is a well researched are
  - Previous works (Bertran *et al.* ICS'10, Bellosa *et al.* SIGOPS EU'00, Srinivasan *et al.* SIGOPS'11, Isci *et al.* HPCA'06) refer to power prediction at the current HW configuration
- **Performance Modelling** is based on 3 major contraints
  - Inherent ILP of the execution flow
  - Number of stall cycles in the cache hierarchy
  - The number and latency of functional units such as FP and INT
- The relationship between these constraints makes it difficult to predict performance

Scalability

- Invoked every 250 ms
- RDTSC instruction to determine computational cost
- DVFS state changes are in order of microseconds
- FACTS: a scheduling algorithm when consolidation cores



Related Work

	Power and Performance modelling							
	Rountree *et al.* IPDPSW'15	**Das *et al.*** WEED'09	**Sasaki *et al.*** PACT'13	**Cochran *et al.*** MICRO'11	**Su *et al.*** MICRO'14	**Spiliopoulos *et al.*** IGCC'11	**Singh *et al.*** SIGARCH'09	**REPP**
PMCs	No, RAPL	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic Power Management	DVFS	Forced Idleness	DVFS & number of cores	DVFS	DVFS	DVFS	DVFS	DVFS, Cl-States, and Consolidation
Application / Architecture Specific	No / **Yes**	No / **Yes**	No / **Yes**	No / No	No / **Yes**	No / **Yes**	No / **Yes**	No / No

Hipster

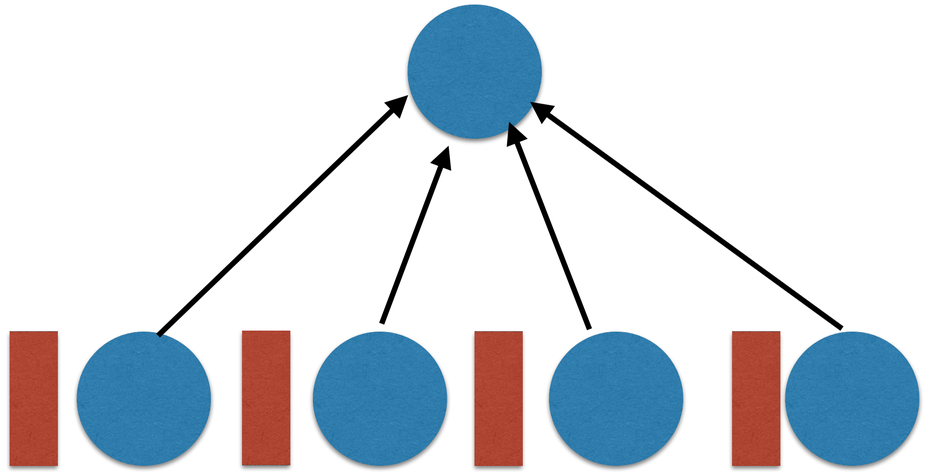
Tail Latency

- Requests fan in bursts
  - Large requests, 1000 s of servers

Network must meet deadlines and handle fan-in bursts

Tail Latency

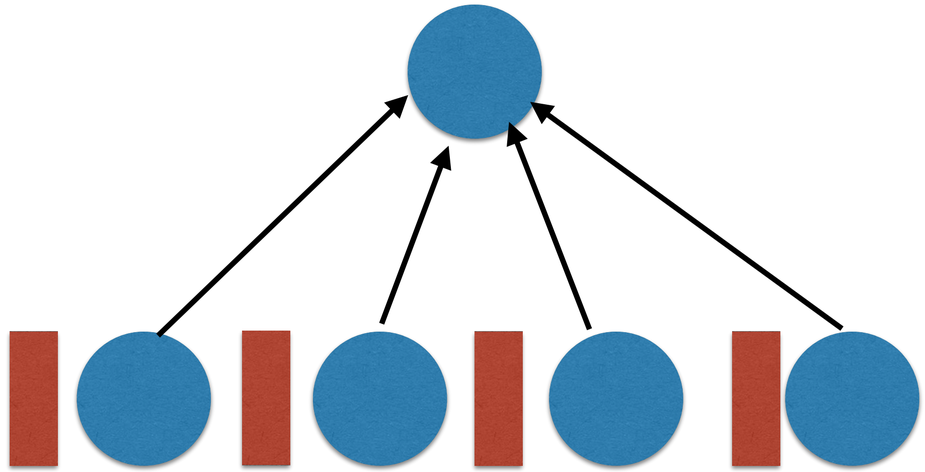
- Requests fan in bursts
  - Large requests, 1000 s of servers



Network must meet deadlines and handle fan-in bursts

Tail Latency

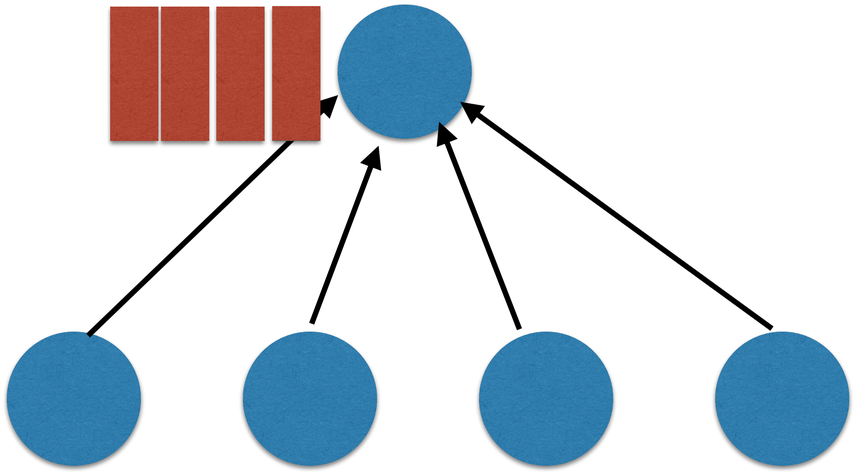
- Requests fan in bursts
  - Large requests, 1000 s of servers
    - Tree like data structure
    - Fan in-bursts -> Long Tail Latency



Network must meet deadlines and handle fan-in bursts

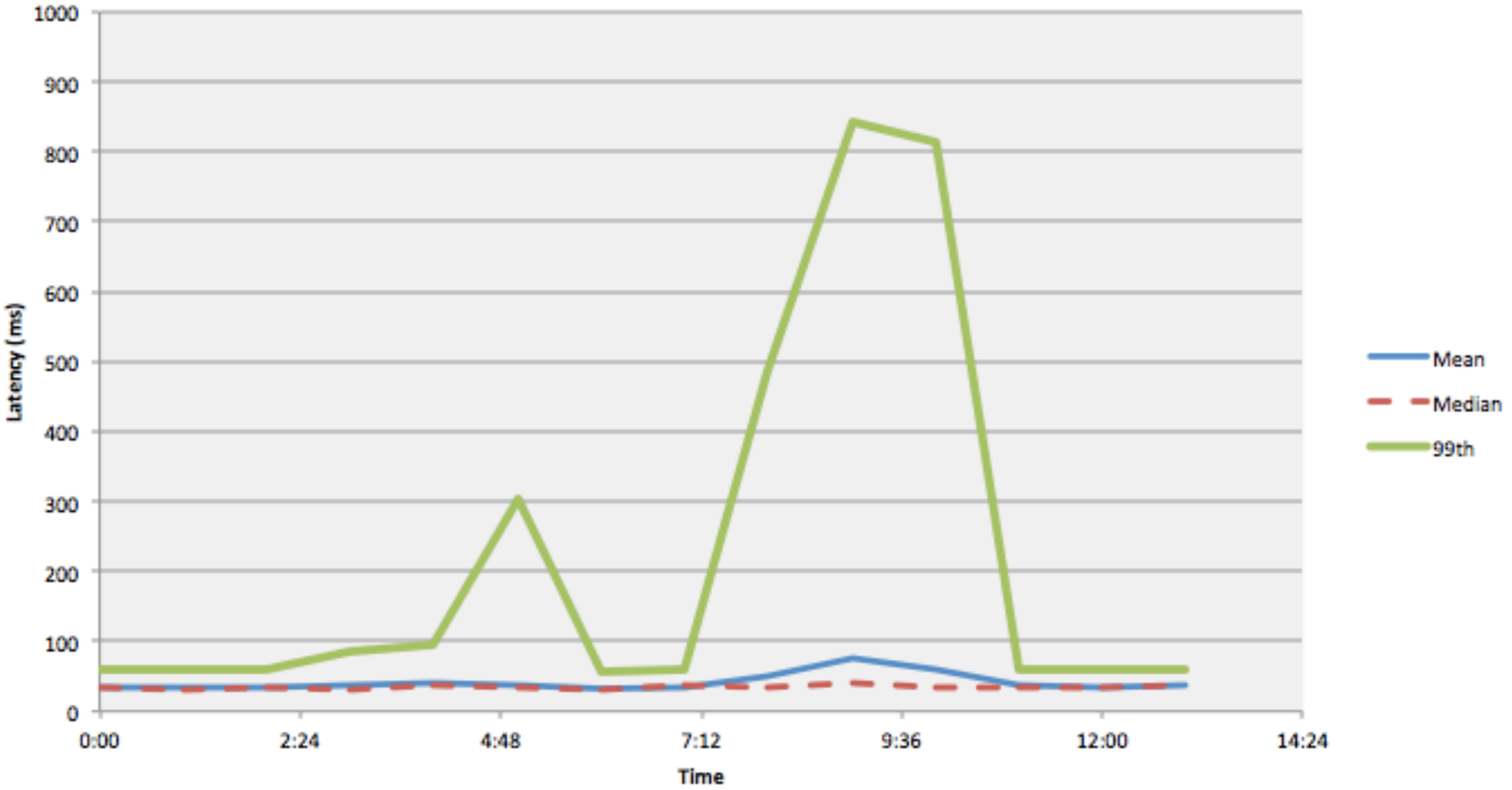
Tail Latency

- Requests fan in bursts
  - Large requests, 1000 s of servers
    - Tree like data structure
    - Fan in-bursts -> Long Tail Latency



Network must meet deadlines and handle fan-in bursts

99th percentile latency



Hipster

- RL problem solved by Hipster is a Markov Decision Process **(MDP)**
  - In an MDP, the decision making process to select best course of action over time
- A MDP is a tuple of $\langle w_n, c_n, p_n, \lambda_n, \gamma^n \rangle$ for a discrete time interval n
  - w_n— Current state
  - c_n — Current action from a finite set
  - p_n — Unknown probability distribution
  - λ_n — Reward function based on w_{n+1} selected with a random probability of p_n
  - γ^n — Discount factor between [0,1]

The problem is to maximise the total discounted reward $\sum_{n=0}^{\infty} \gamma^n \lambda_n$

Hipster's MDP

- A MDP is a tuple of $\langle w_n, c_n, p_n, \lambda_n, \gamma^n \rangle$ for a discrete time interval n
  - w_n — Current load of latency-critical workload in t_n to t_{n-1}
    - Hipster quantises the load into buckets
  - c_n — Configuration (core combination+DVFS) in t_n to t_{n+1}
  - p_n — Unknown probability distribution
  - λ_n — At t_{n+1}, the reward is computed for HipsterIn/HipsterCo
  - γ^n — Fixed discount factor

Markov Decision Process

State (w)	Action (c)	Reward(R (w, c))
W_1	C_1	0
W_1	C_2	25
W_1	C_3	28
W_2	C_1	20
W_2	C_2	35
W_2	C_3	50

- RL is a type of unsupervised machine learning
- Solves MDP using a **lookup** table
  - State (w) — Quantised load
  - Action (c) — core mapping + DVFS
  - Reward (R(w,c)) -- Reward for a given configuration and state

Markov Decision Process

State (w)	Action (c)	Reward(R (w, c))
W_1	C_1	0
W_1	C_2	25
W_1	C_3	28
W_2	C_1	20
W_2	C_2	35
W_2	C_3	50

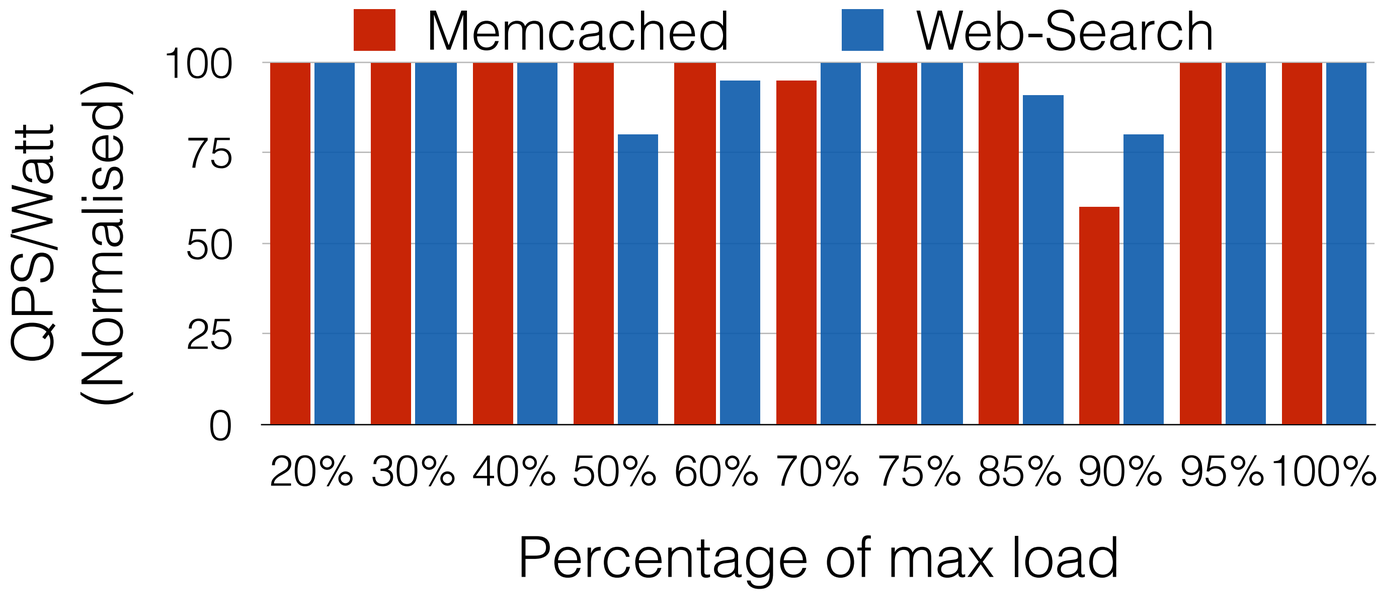
- RL is a type of unsupervised machine learning
- Solves MDP using a **lookup** table
  - State (w) — Quantised load
  - Action (c) — core mapping + DVFS
  - Reward (R(w,c)) -- Reward for a given configuration and state

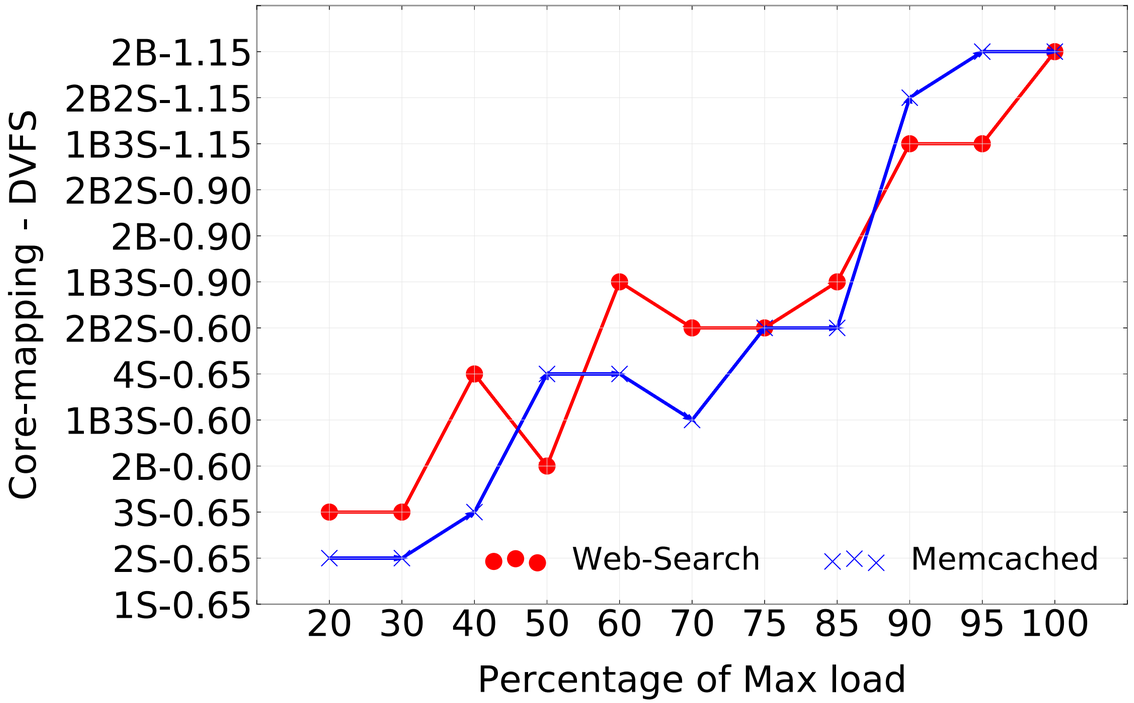
Markov Decision Process

State (w)	Action (c)	Reward(R (w, c))
W_1	C_1	0
W_1	C_2	25
W_1	C_3	28
W_2	C_1	20
W_2	C_2	35
W_2	C_3	50

- RL is a type of unsupervised machine learning
- Solves MDP using a **lookup** table
  - State (w) — Quantised load
  - Action (c) — core mapping + DVFS
  - Reward (R(w,c)) -- Reward for a given configuration and state

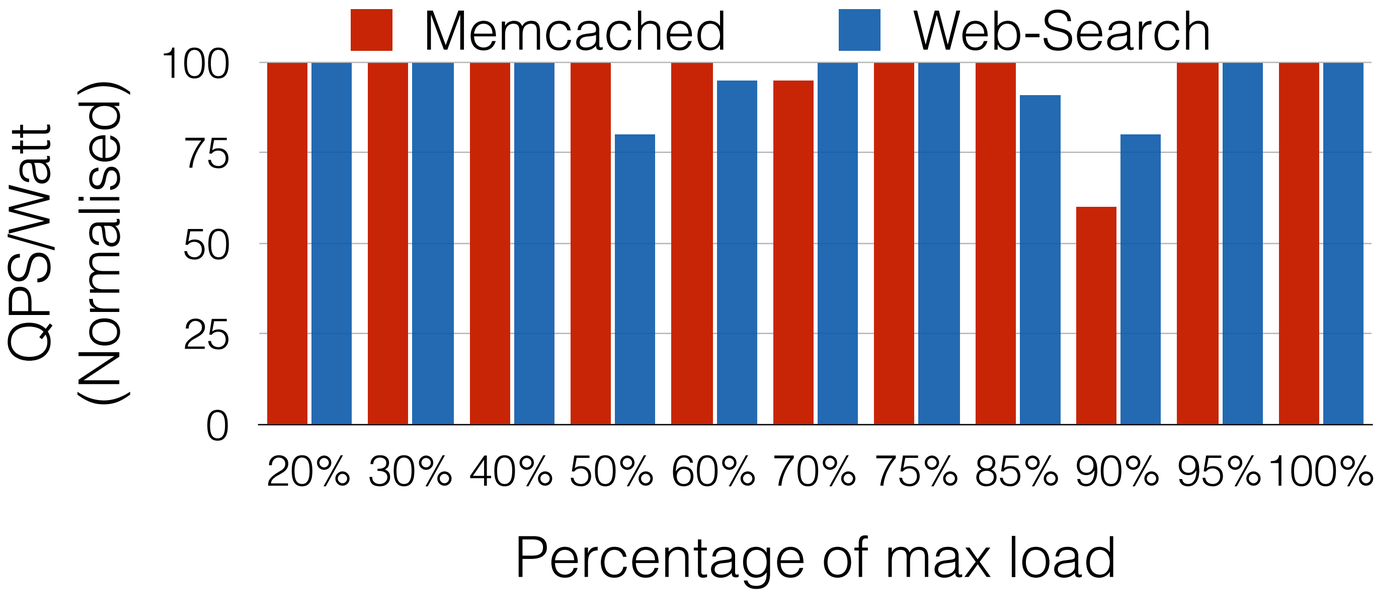
Swapped mappings

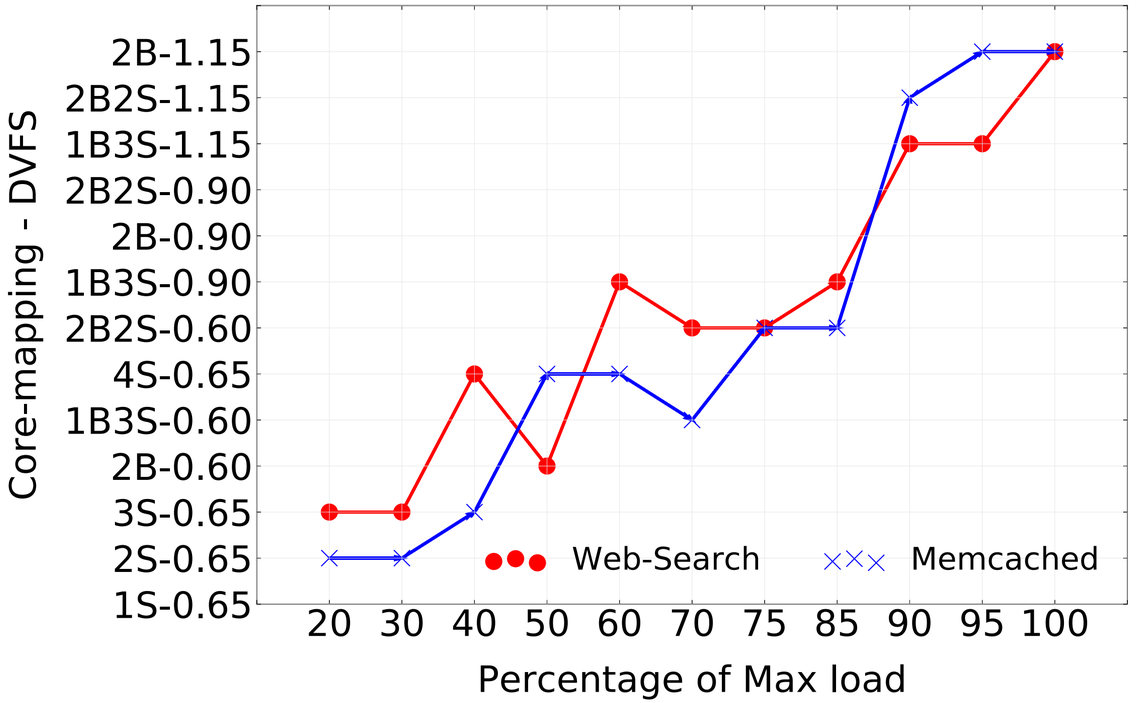




What happens when we use core mapping decisions learned from Memcached to Web-Search and vice-versa?

Swapped mappings

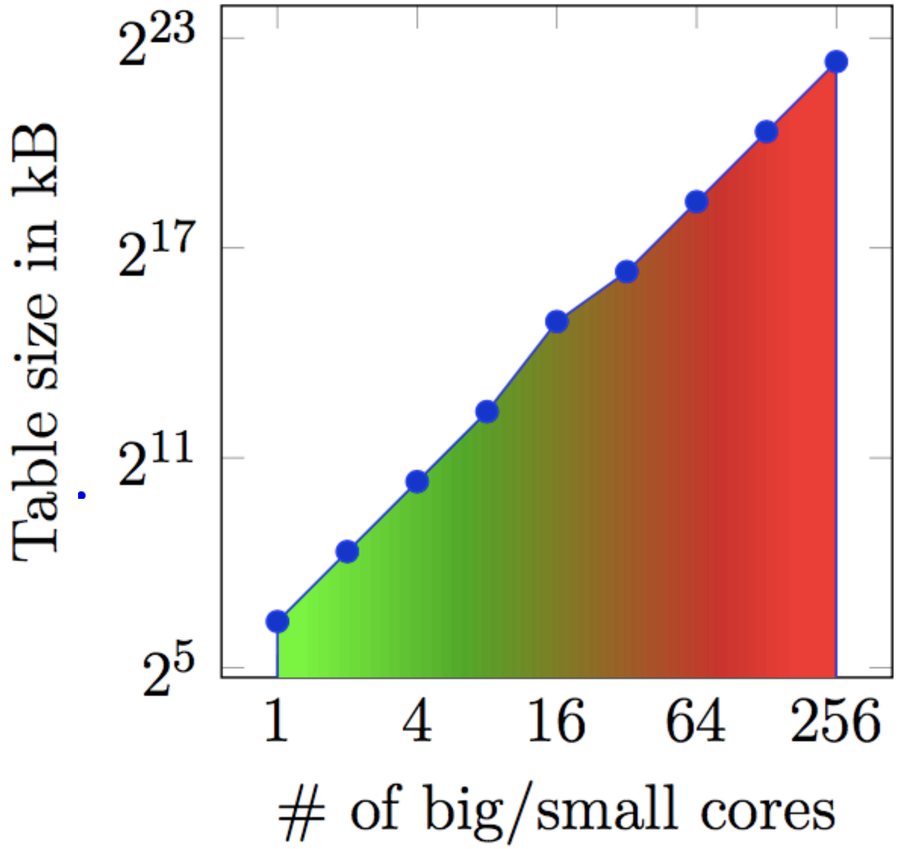




What happens when we use core mapping decisions learned from Memcached to Web-Search and vice-versa?

Application-tailored mappings are more energy efficient

Hipster's Scalability



Lookup table size (log scale) with same number of big and small cores using 10 different DVFS settings, and bucket size of 100.