SPECTR

Formal Supervisory Control and Coordination for Many-core Systems Resource Management

Amir M. Rahmani   Bryan Donyanavard   Tiago Mück   Kasra Moazzemi
Axel Jantsch   Onur Mutlu   Nikil Dutt

23rd ACM International Conference on Architectural Support for Programming Languages and Operating Systems
Williamsburg, VA, March 2018

Executive Summary

- Motivation:
  - *Formal supervisory control theory (SCT) can combine the strengths of classical control theory with heuristic approaches to efficiently meet changing runtime goals.*
  - *SCT enables hierarchical control and facilitates automatic synthesis of the high-level supervisory controller and its property verification.*
- Problem: Current resource management techniques do not offer 1) robustness, 2) formalism, 3) efficiency, 4) coordination, 5) scalability, and 6) autonomy all together.
- Goal: Address all six key challenges in heterogeneous multiprocessors (HMPs) resource management, in particular scalability and autonomy
- Our Proposal: SPECTR uses SCT techniques such as gain scheduling to allow autonomy for individual controllers, and modular decomposition of control problems to manage complexity.
- Evaluation:
  1. *We implement SPECTR on an Exynos platform containing ARM's big.LITTLE-based HMP*
  2. *SPECTR can manage multiple interacting resources (e.g., chip power and processing cores) in the presence of competing objectives (e.g., satisfying QoS vs. power capping)*
  3. *SPECTR achieves up to 8x and 6x better target QoS and power tracking over state-of-the-art, respectively (in our case study).*

SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

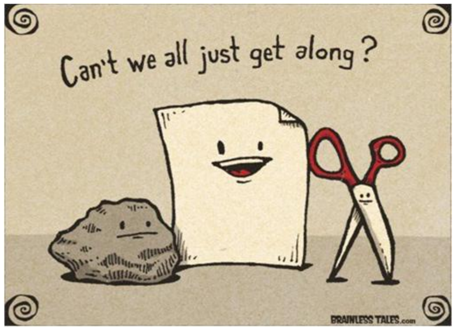
SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

Resource Management in Many-core Systems

- Several conflicting goals/constraints

- Multiple tunable knobs



- Ad hoc heuristics
  - Can be sub-optimal
  - No formal methodology
  - No guarantees

Challenges in Resource Management

Can we offer a systematic design flow for hierarchical control (Scalability)?

The Goal

	Methods	Robustness	Formalism	Efficiency	Coordination	Scalability	Autonomy
A	Machine learning		✓	✓	✓		
B	Estimation/Model based heuristics			✓	✓		
C	SISO Control Theory	✓	✓	✓		✷	
D	MIMO Control Theory	✓	✓	✓	✓		
E	Supervisory Control Theory [SPECTR]	✓	✓	✓	✓	✓	✓

Major on-chip resource management approaches and the key questions they address (∗ = partially addressed)

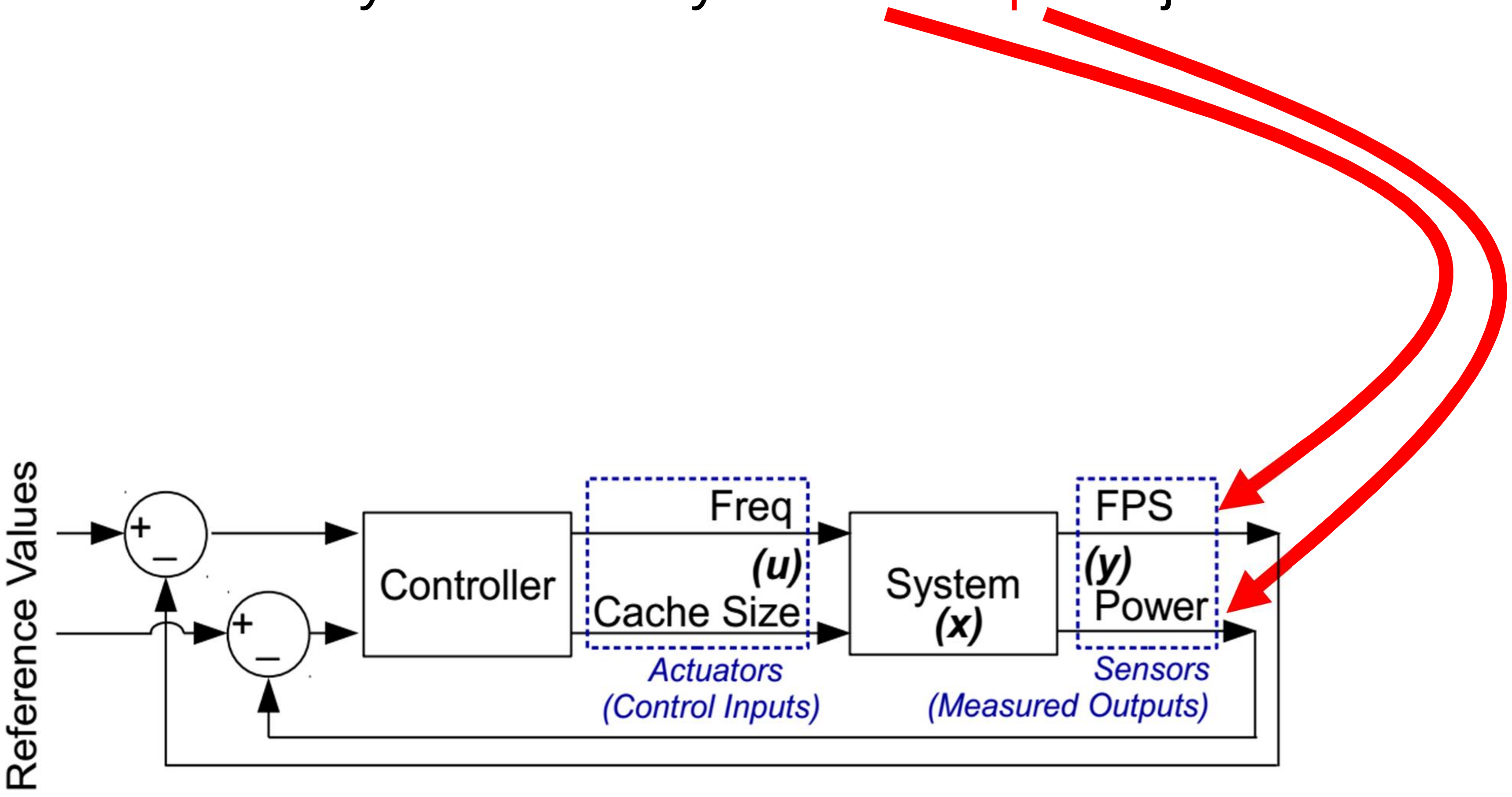
SPECTR Outline

- Motivation
- **MIMO Control Theory for Coordinated Management**
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

MIMO Control Theory for Coordination

Benefits:

- Simultaneously and robustly track multiple objectives



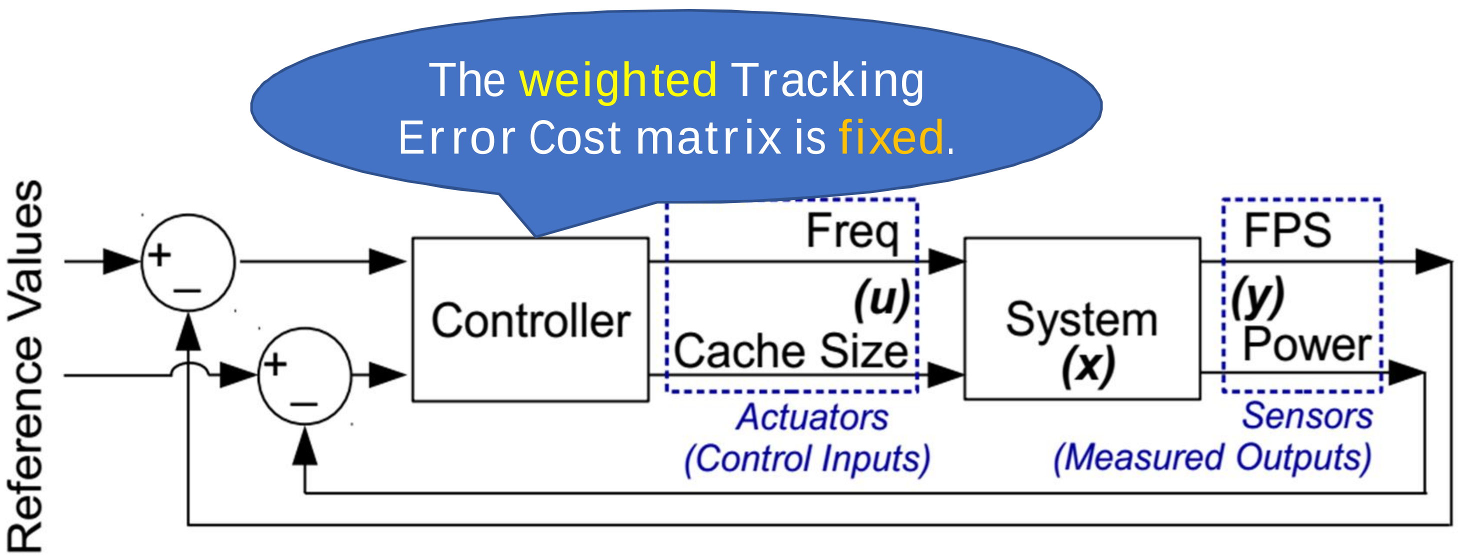
MIMO Control Theory for Coordination

Benefits:

- Simultaneously and robustly track multiple objectives

Shortcomings:

- The goal is fixed at design-time



FPS:Power <= 1:10

when Maximizing FPS under a Power cap

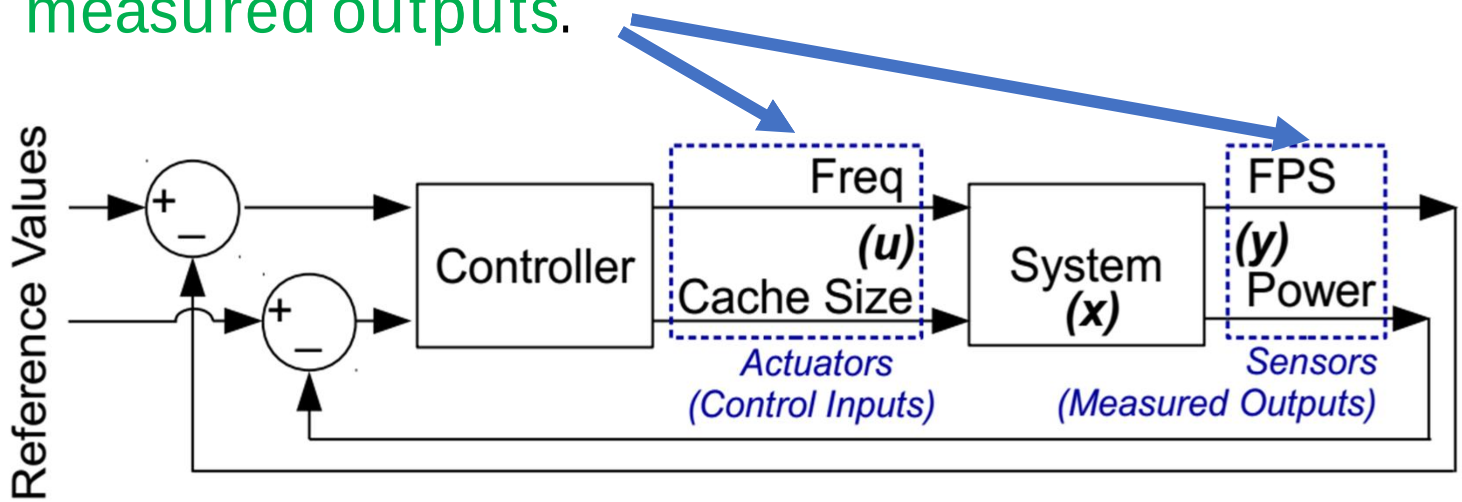
MIMO Control Theory for Coordination

Benefits:

- Simultaneously and robustly track multiple objectives

Shortcomings:

- The goal is fixed at design-time
- Does NOT scale when having several control inputs and measured outputs.

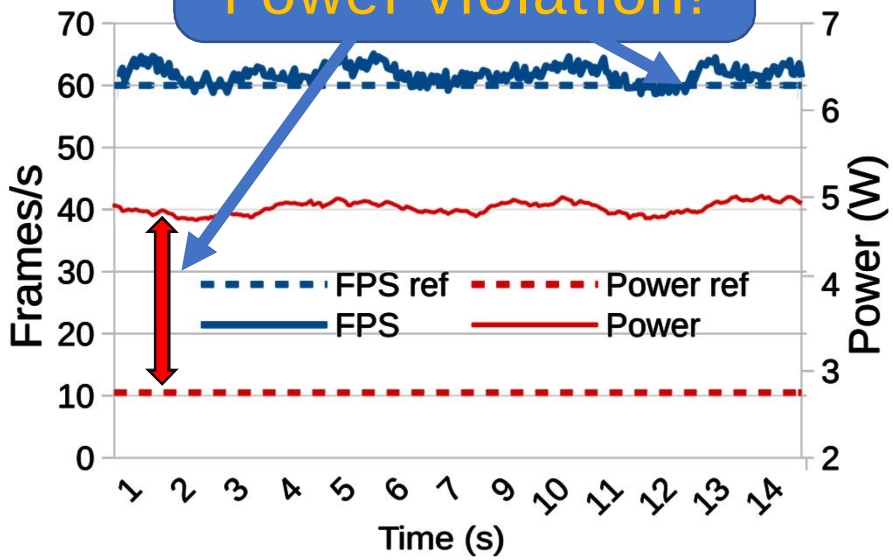


SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

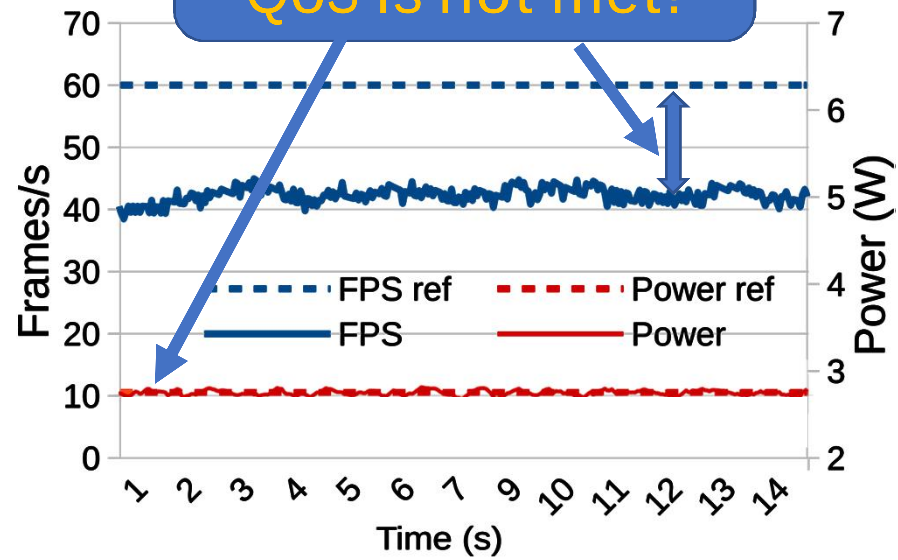
The Autonomy Challenge: An Example

QoS is met.
Power violation!



A MIMO controller designed with higher priority on QoS over power

Power is capped.
QoS is not met!



A MIMO controller designed with higher priority on power over QoS

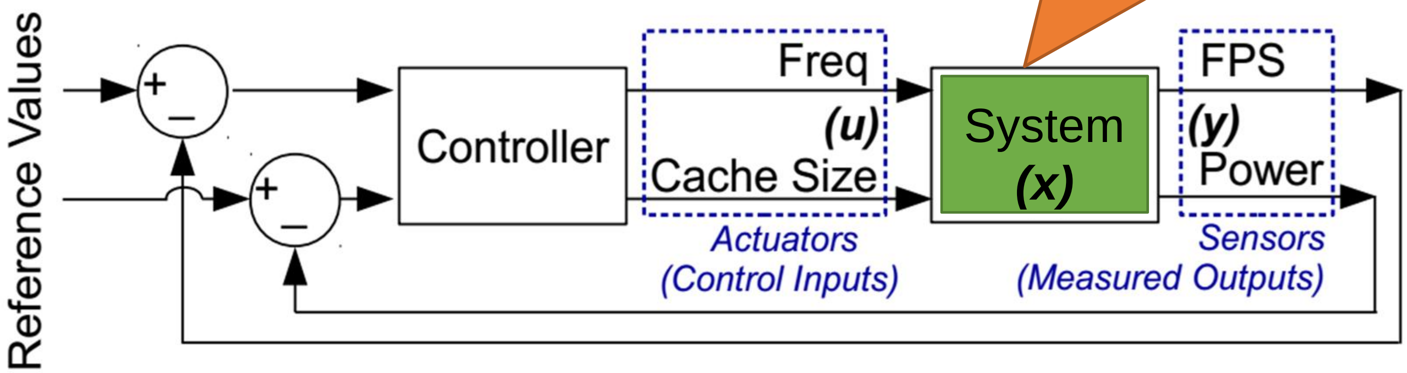
What if the goal changes at runtime?

We need the ability to switch modes at runtime

The Scalability Challenge: Example 1

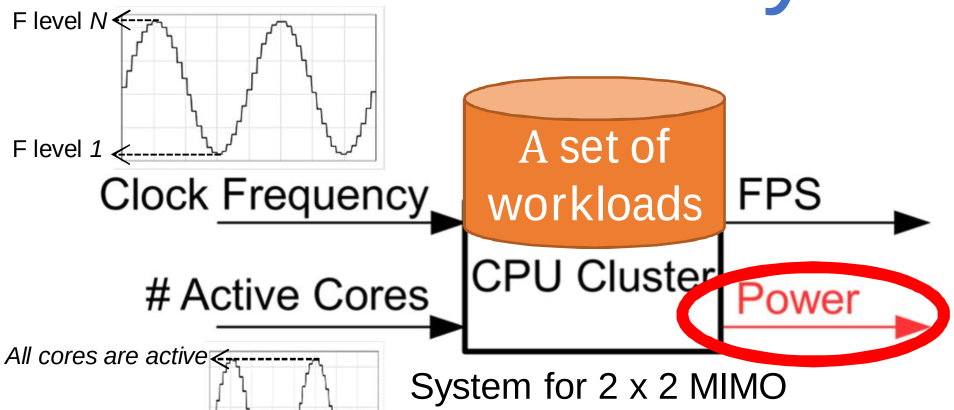
$$
\begin{aligned}
x(t+1) &= A \times x(t) + B \times u(t) \\
y(t) &= C \times x(t) + D \times u(t)
\end{aligned}
$$

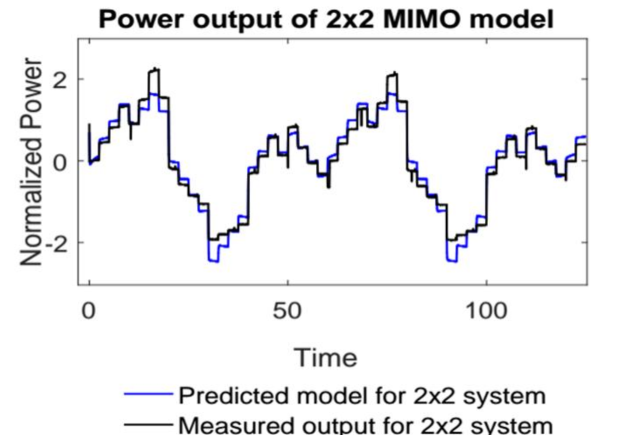
Black-box Identification of System Dynamics



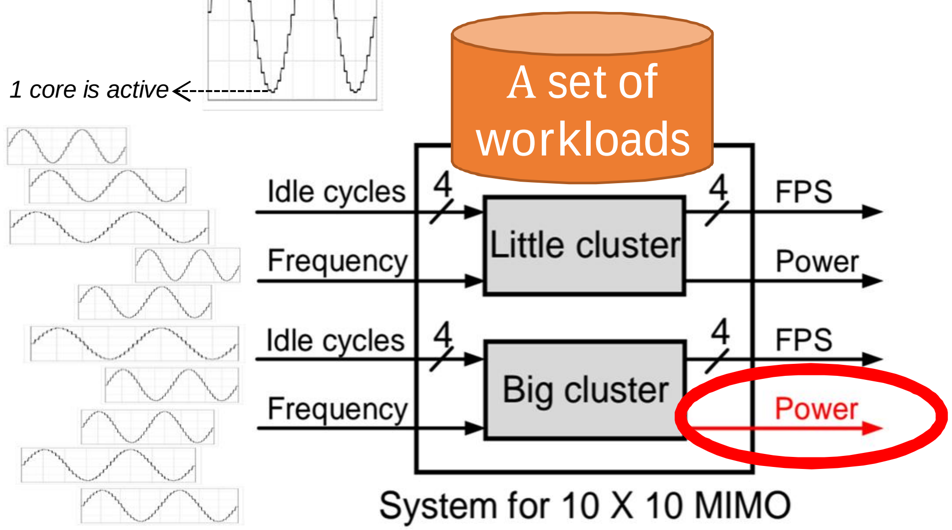
What if the # control inputs and measured outputs is large?

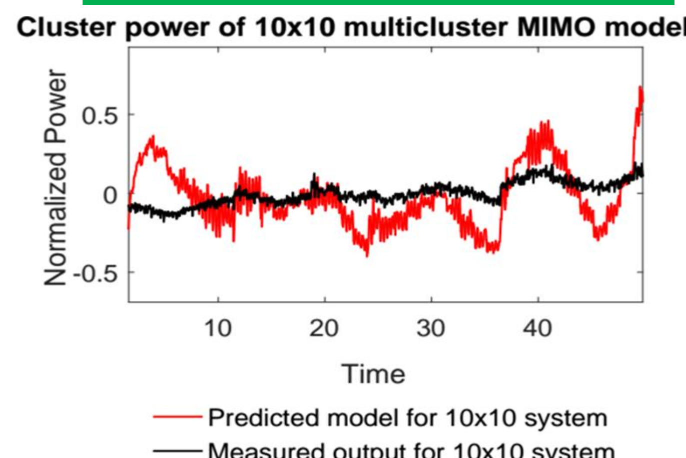
The Scalability Challenge: Example 1



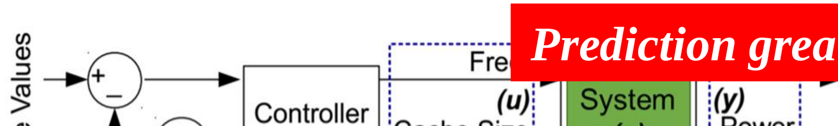


Prediction is very accurate





Prediction greatly diverges from reality

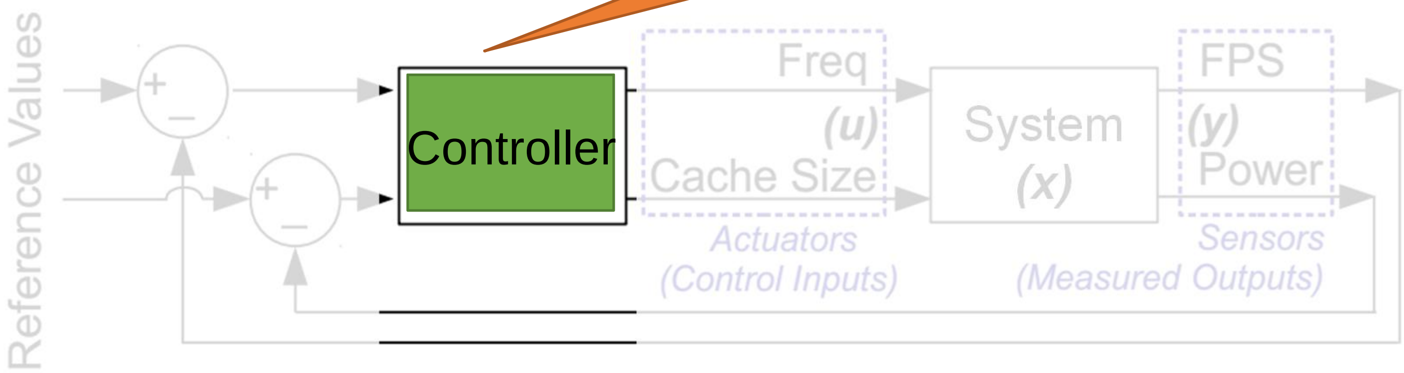


We need to limit the system size

The Scalability Challenge: Example 2

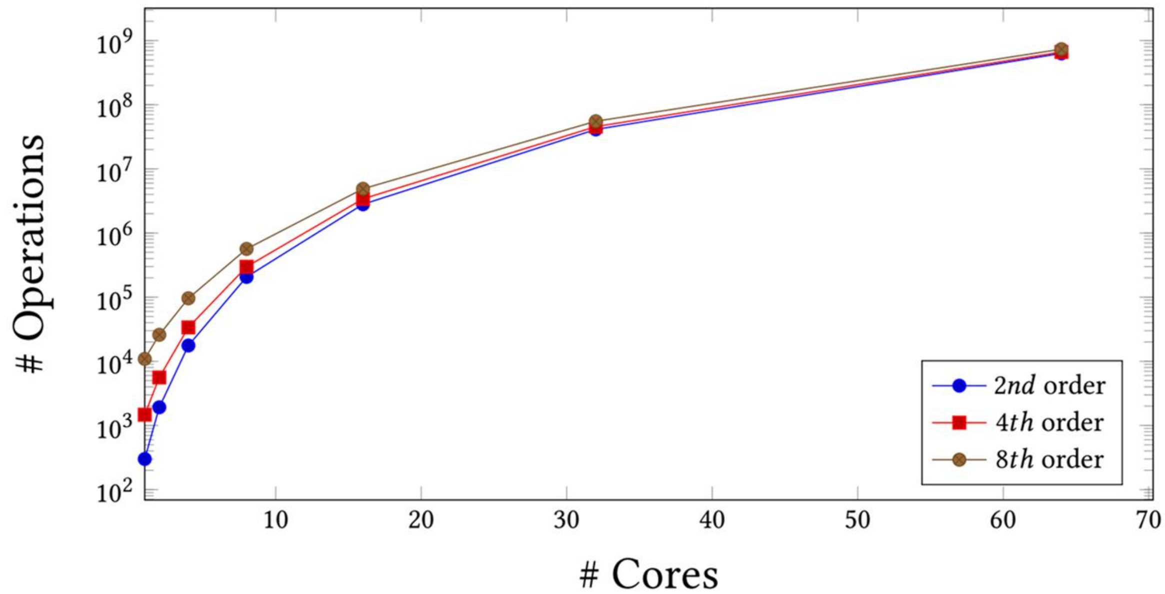
$$x(t+1) = A \times x(t) + B \times u(t)$$
$$y(t) = C \times x(t) + D \times u(t)$$

Controller Design Complexity

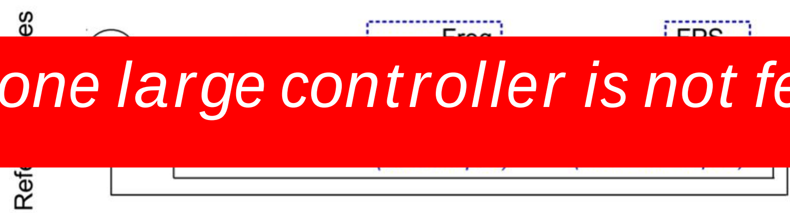


What if the # control inputs and measured outputs is large?

The Scalability Challenge: Example 2



How many operations are executed in each control epoch for a single large MIMO controlling N cores?



Using one large controller is not feasible!

SPECTR Outline

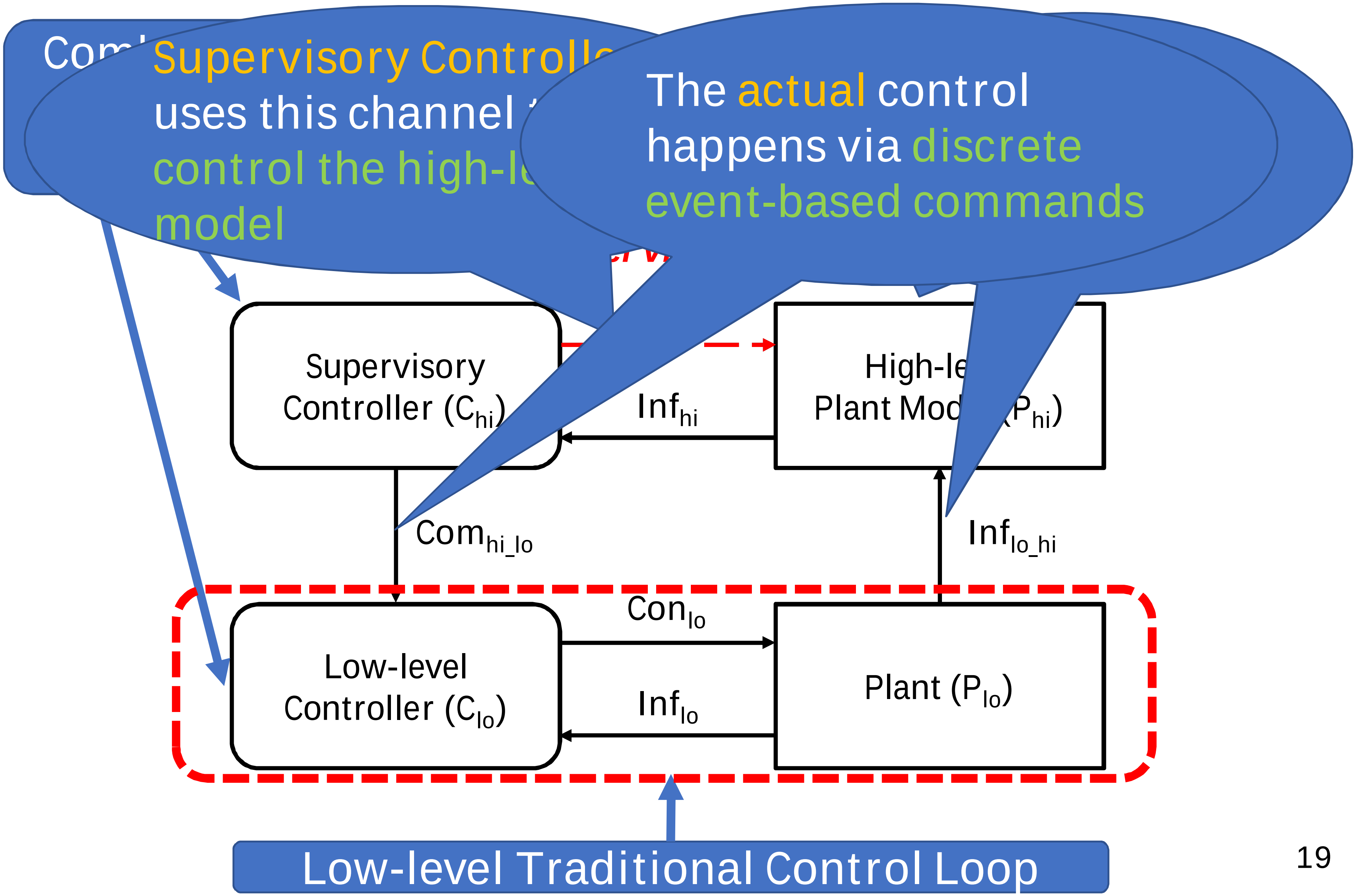
- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

SPECTR

Using Supervisory Control Theory we...

- Provide autonomy via adaptation in response to changes in policy
  - Compute control parameters for different policies offline
- Provide scalability via decomposition of system into multiple subsystems organized in a hierarchy
  - Supervisor provides high-level management

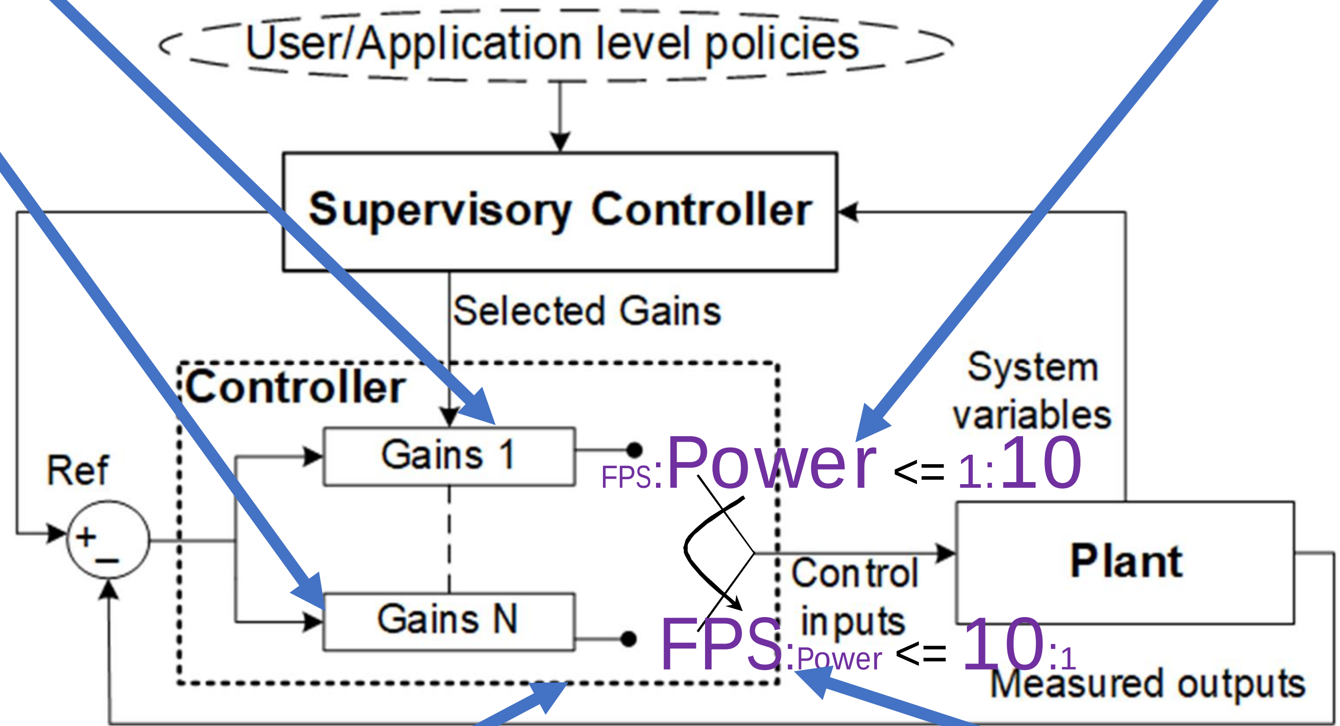
Scalability via Supervisory Control



Autonomy via Supervisory Control

Control parameters pre-designed to prioritize one measured output over the other(s)

Tracking power is 10x more important than tracking QoS



This SCT technique is called *Gain Scheduling*

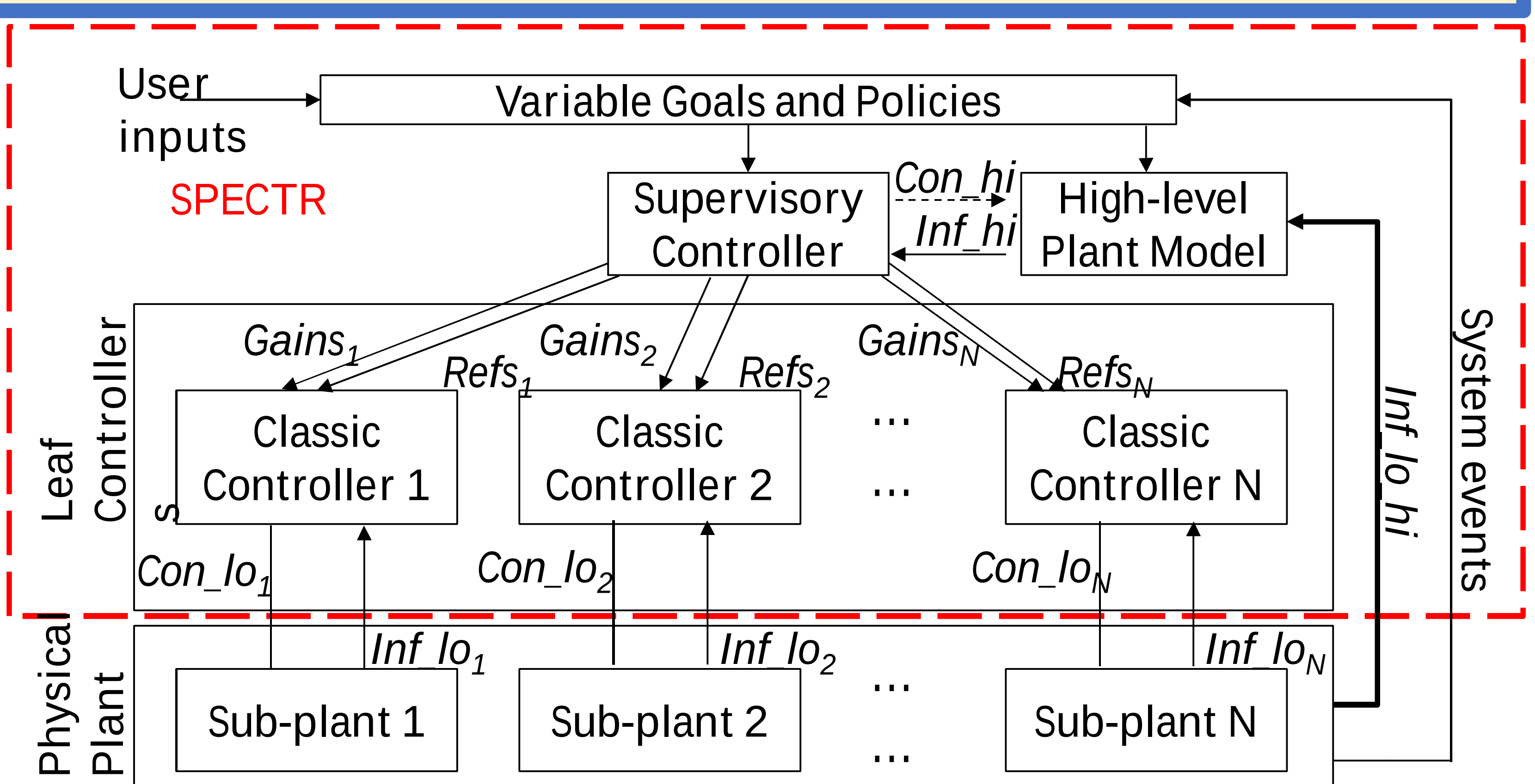
Tracking QoS is 10x more important than tracking power

SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- **Supervisory Control Theory (SCT) via SPECTR**
  - Scalability and Autonomy through SCT
  - **SPECTR Overview**
  - Case Study
  - Supervisor Synthesis Process
- Results
- Summary

SPECTR overview

Putting **hierarchical control** and **gain scheduling** together!

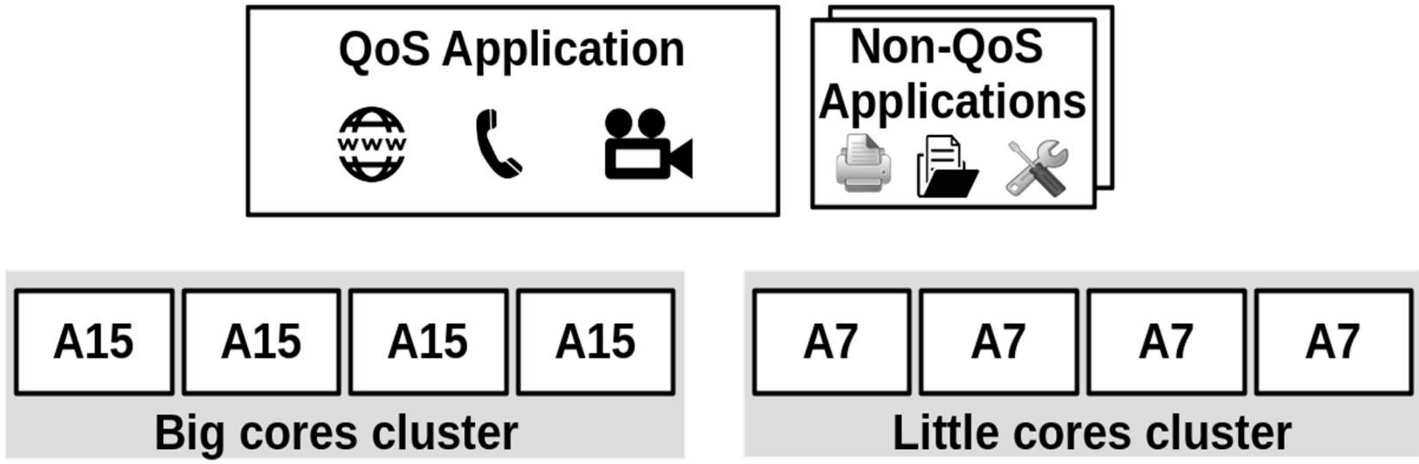


The supervisor **updates goals** and **allocates resources** at runtime

SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- **Supervisory Control Theory (SCT) via SPECTR**
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - **Case Study**
    - Supervisor Synthesis Process
- Results
- Summary

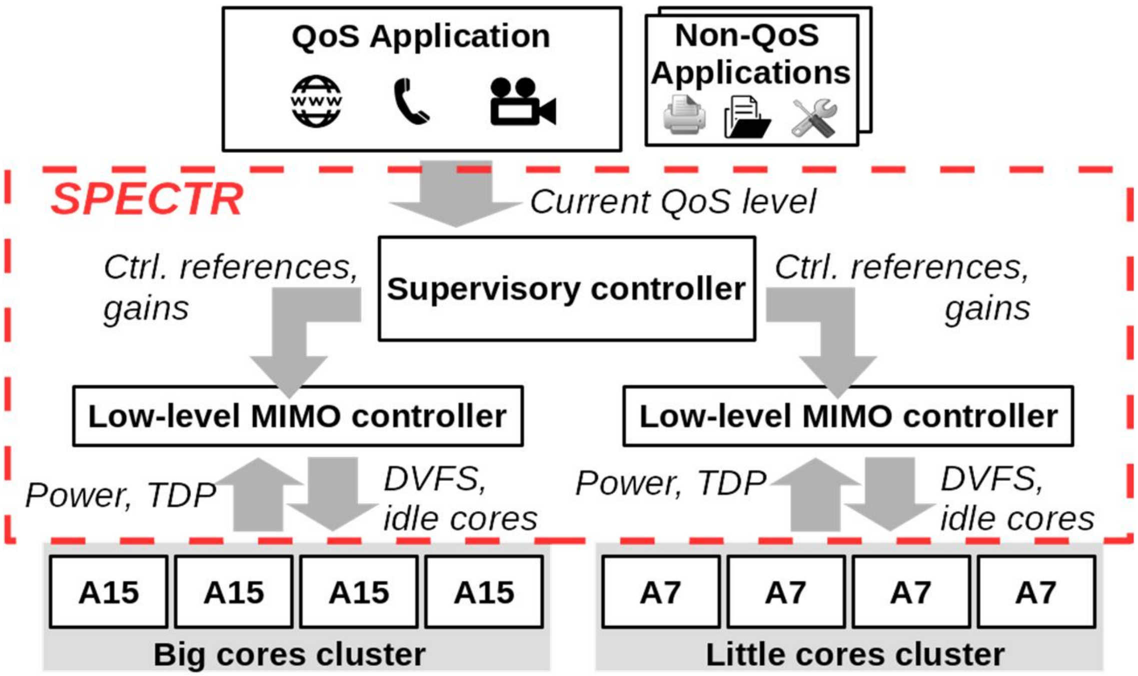
Case Study



ODROID-XU3 platform contains an Exynos 5422 Octa-core SoC

- 8-core big.Little HMP
- Two set of applications:
  - A foreground application with QoS requirements (e.g., FPS)
  - A number of background applications with no QoS requirements

Case Study

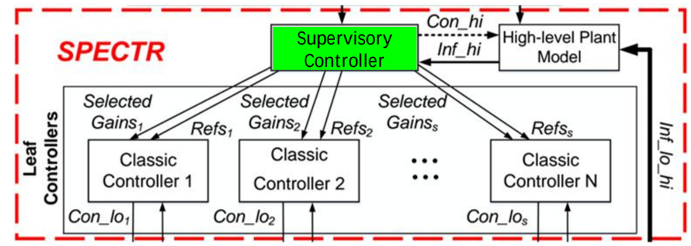


- Control knobs: per-cluster DVFS, number of idle cores
- System goals:
  - Meet the QoS requirement of the foreground application
  - Ensure the total system power always remains below the Thermal Design Power (TDP)
  - Minimize energy consumption

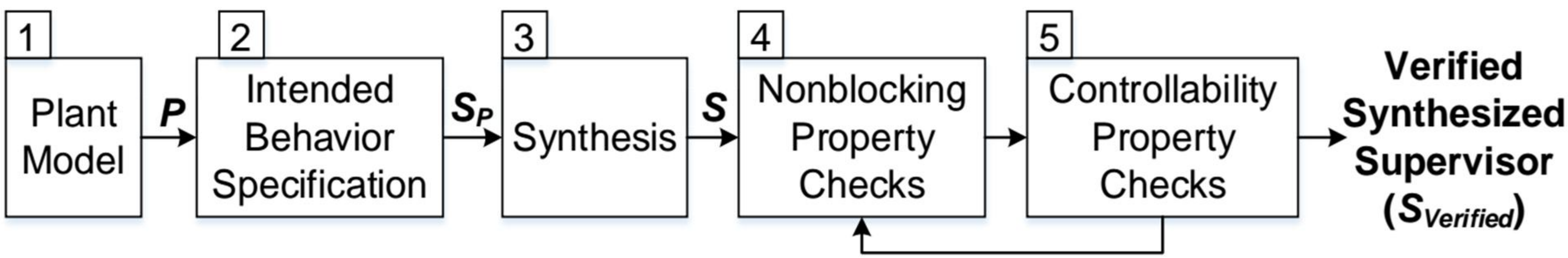
SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- **Supervisory Control Theory (SCT) via SPECTR**
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - **Supervisor Synthesis Process**
- Results
- Summary

Supervisor Synthesis Process



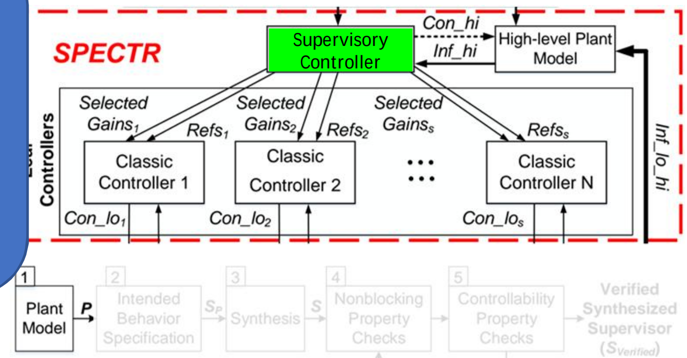
5 steps to design and verify a supervisor:

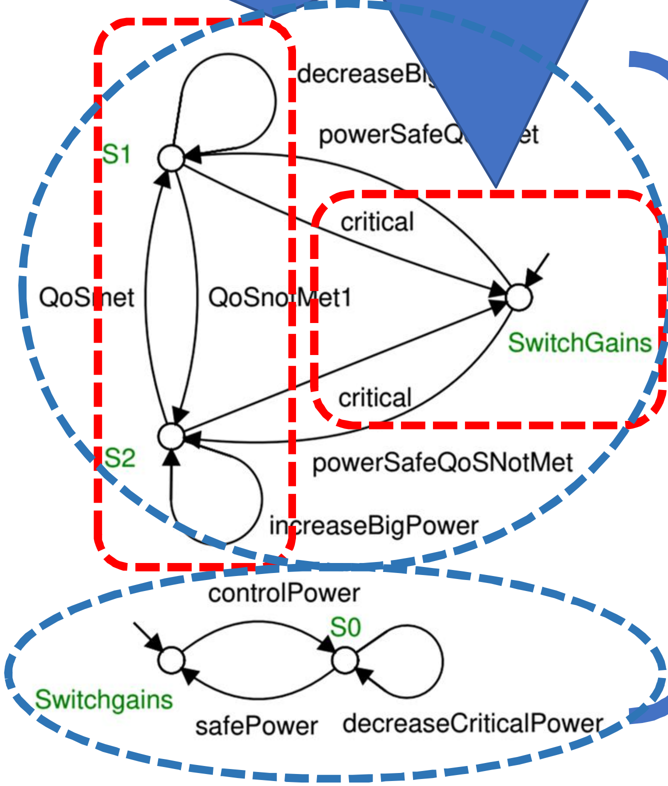


Step 1: Plant Model

Prio
refe
meet
an er

Power budget violation generates a *critical* event and results in gain switching towards the power-driven goal.

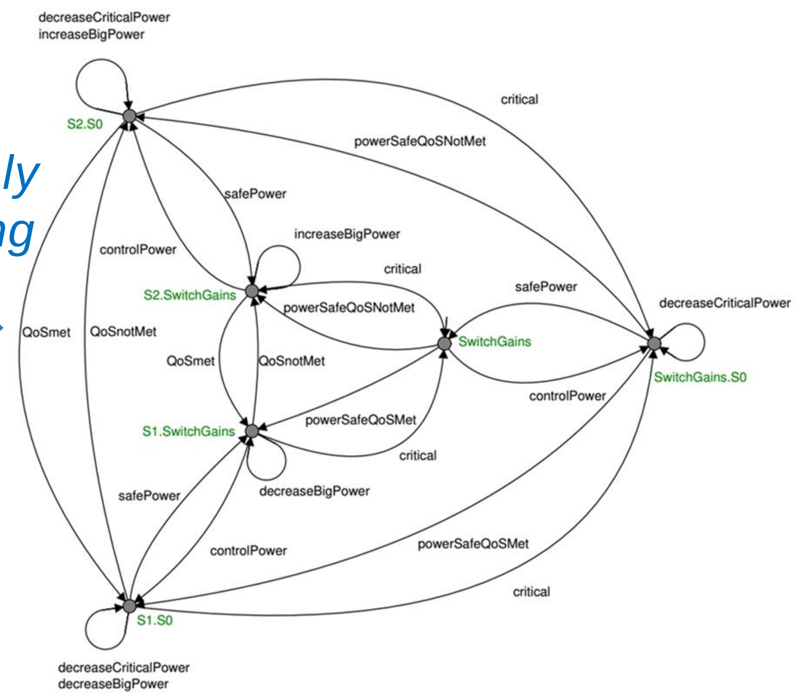




Manually modeled sub-plants

Multiple characteristics are automatically synthesized using SCT tools

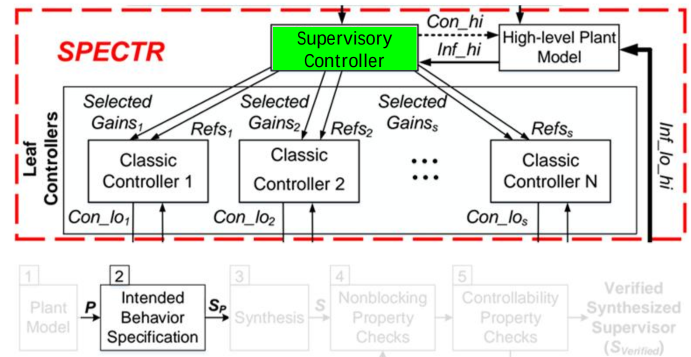
E.g. Supremica

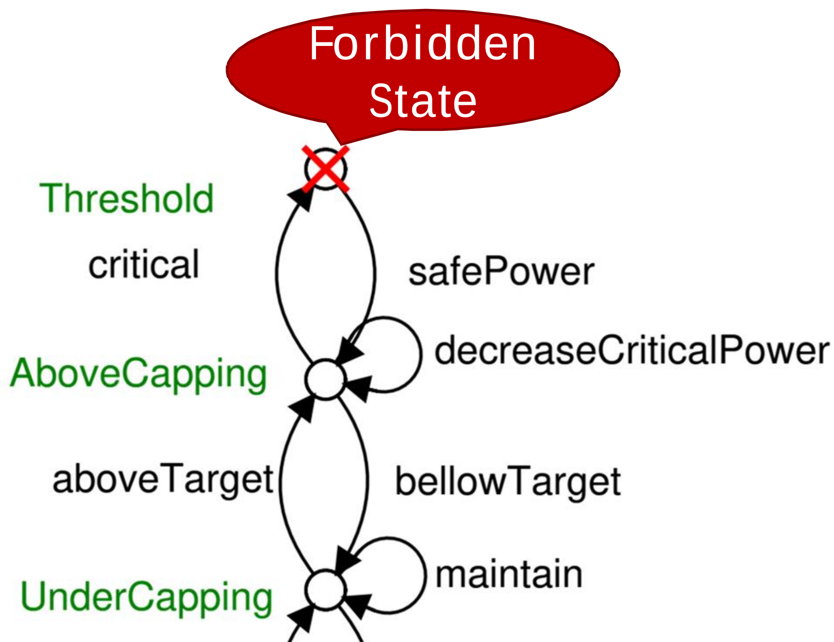


Synthesized plant

Step 2: Intended Behavior Specification

A **specification** defines the **accepted** and **forbidden** states via **restrictions** *on the behavior of the plant model*.



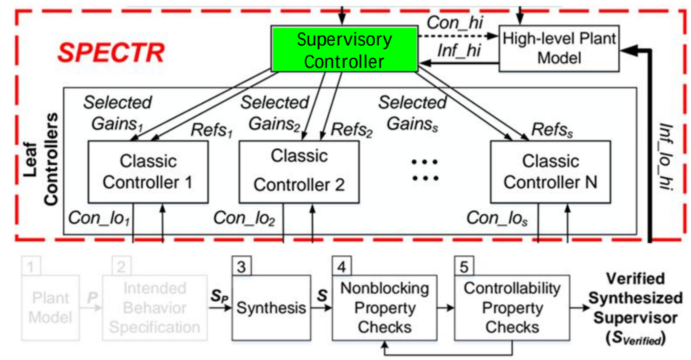


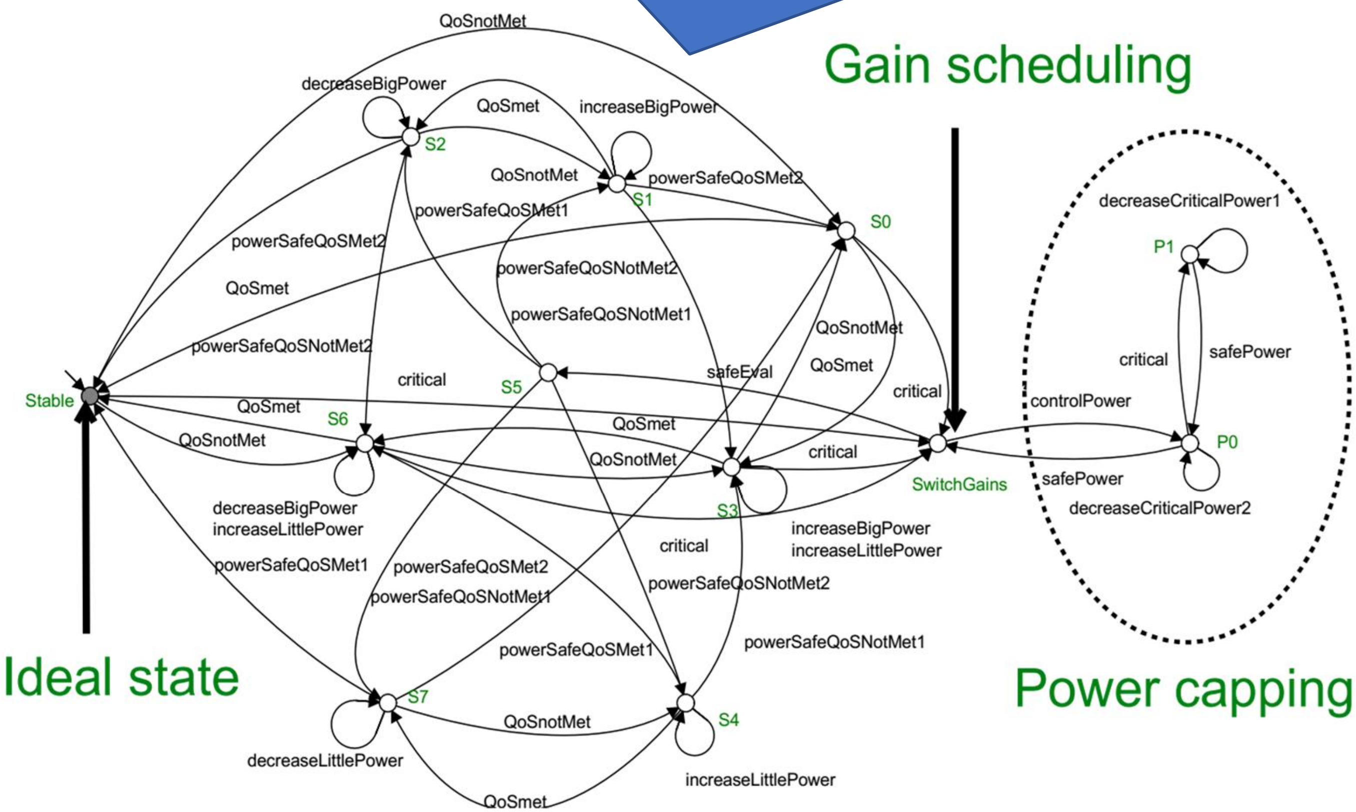
This example specification prevents exceeding the power budget for no more than three control intervals.

Note: The model in Step 1 has no limitations!
(e.g., on exceeding the power budget)

Steps 3-5: Synthesis and Verification

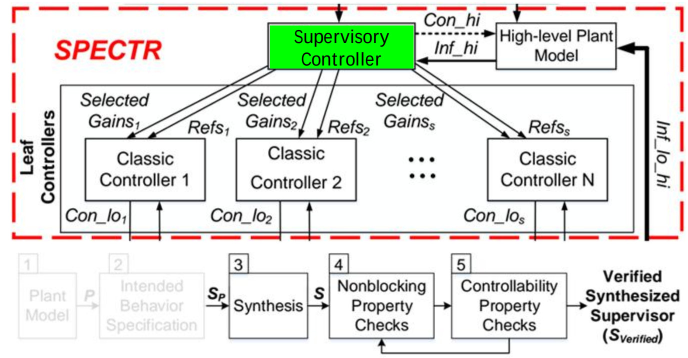
Automatically generated and verified using synchronous composition operations in Supremica SCT tool.



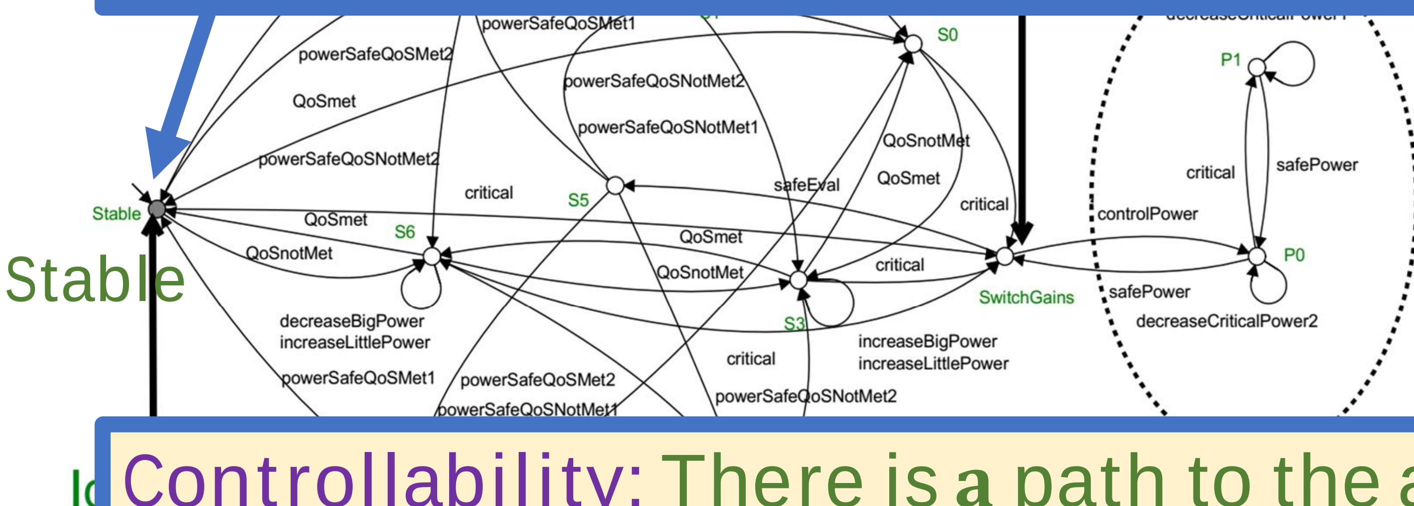


Steps 3-5: Synthesis and Verification

SCT tools (e.g., Supremica) also verify the non-blocking and controllability properties of the synthezied controller.



Non-blocking: Accepted states (e.g., ideal states) can always be reached.



Controllability: There is a path to the accepted states from every other valid state.

SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- **Results**
- Summary

Evaluated resource manager configurations

- Compared SPECTR with three alternative resource managers
  - MM-Pow: 2x2 MIMOs (one per cluster) with gains optimized to track power
  - MM-Perf: 2x2 MIMOs (one per cluster) with gains optimized to track performance/QoS
  - FS: single system-wide 4x2 MIMO with gains optimized towards power

  } Fixed-Objective

- QoS applications:
  - PARSEC applications: x264, bodytrack, canneal, streamcluster
  - Data-intensive machine learning workloads: k-means, KNN, least squares, linear regression

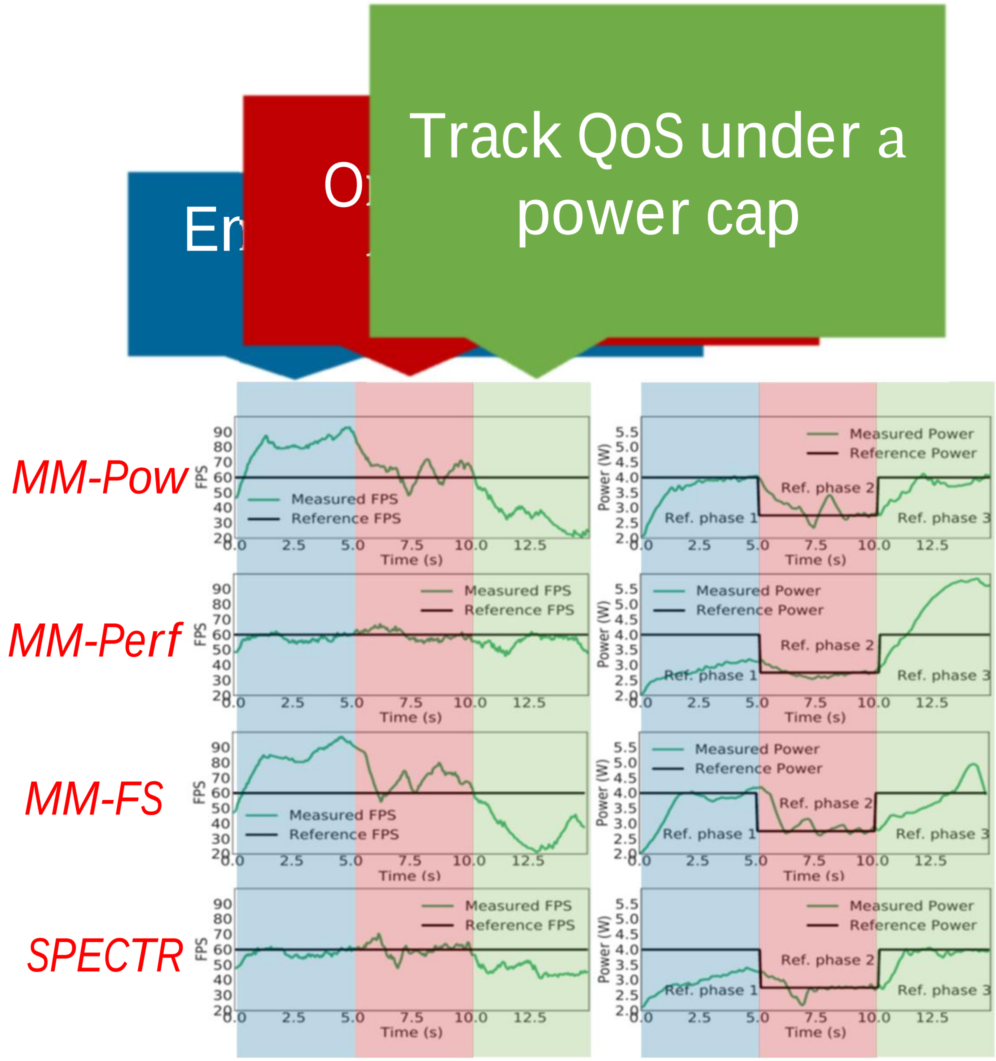
Experimental Results – Contoller Evaluation

Execution scenario with three phases (x264):

1. Phase **1** - Safe Phase:
   only the *QoS* application runs; power limited by TDP

2. Phase **2** - Emergency phase:
   power limit set to 1W below TDP to emulate a thermal emergency

3. Phase **3** - Workload disturbance phase:
   power limit restored to TDP, but now several background tasks start, interfering with the QoS application

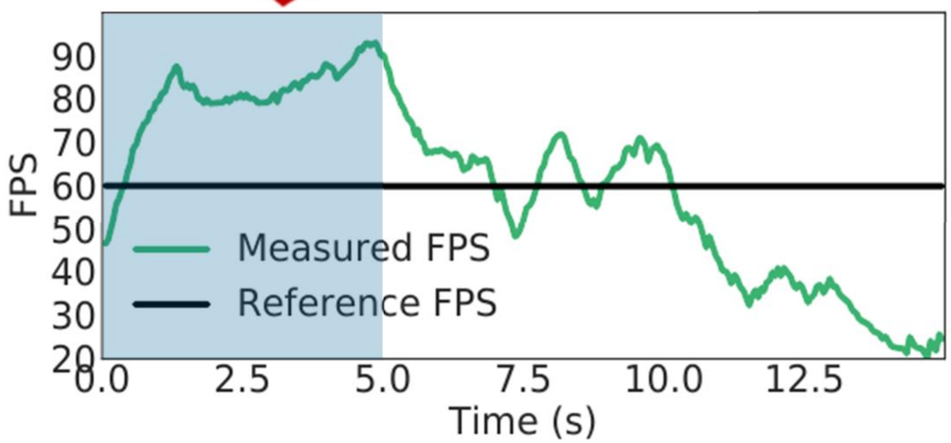


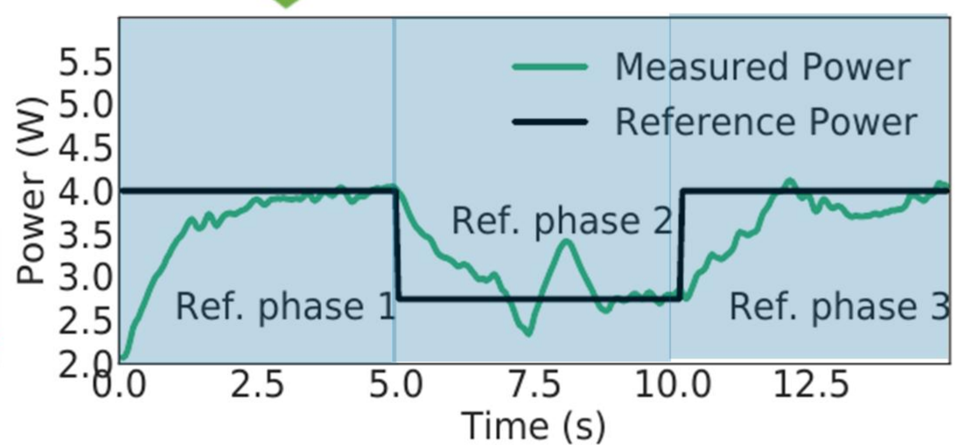
Experimental Results -- Contoller Evaluation

- QoS task: x264
- Controller: MM-Pow (power-oriented)
  - 2x2 MIMOs (one per cluster) with gains optimized to track power

Wasted performance

Under a power cap





~ 40% more then necessary FPS in Phase 1

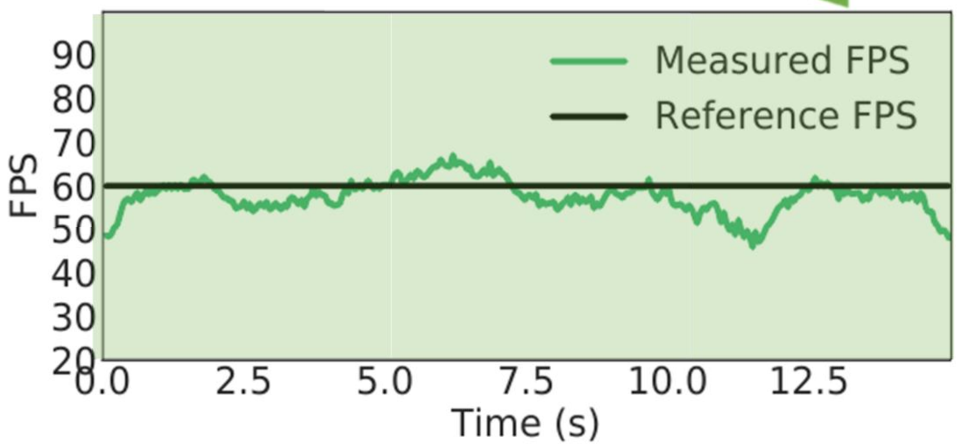
Wasting energy !

It works fine in Phase 2 and 3 by focusing on power capping!

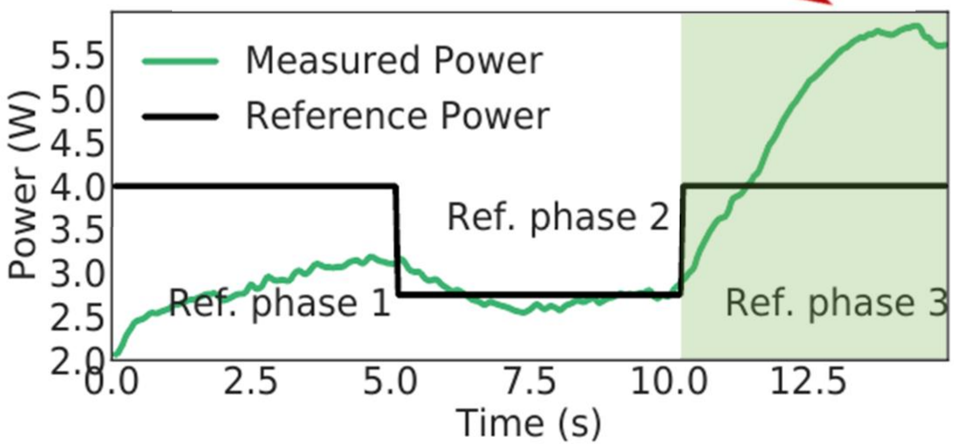
Experimental Results -- Contoller Evaluation

- QoS task: x264
- Controller: MM-Perf (performance-oriented)
  - 2x2 MIMOs (one per cluster) with gains optimized to track QoS

Tracking FPS



Exceeding power limit



Exceeds TDP by ~30% in Phase 3!

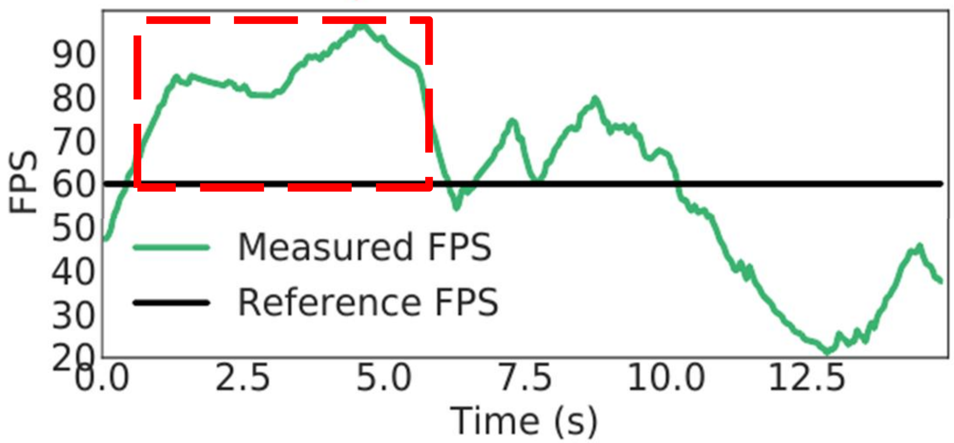
It works fine in Phase 1 and 2 by focusing on QoS tracking!

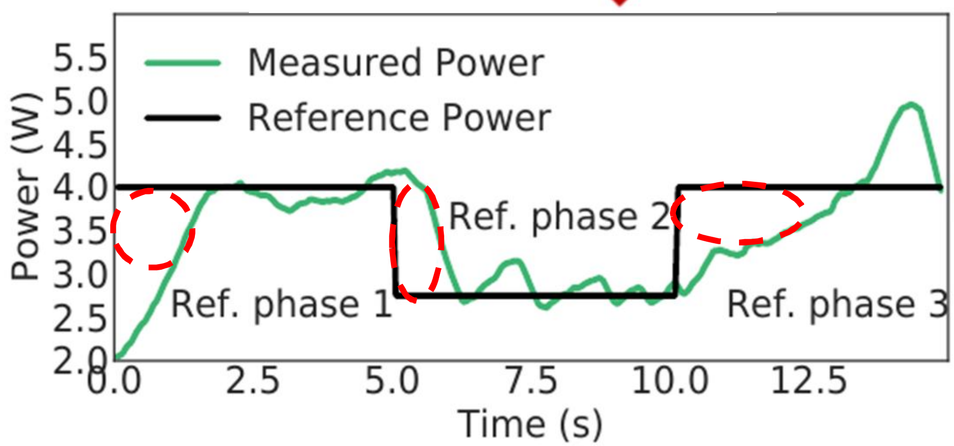
Experimental Results -- Contoller Evaluation

- QoS task: x264
- Controller: FS (large 4x2 power-oriented)
  - Single system-wide 4x2 MIMO with gains optimized towards power

Wasted performance

Sluggish response



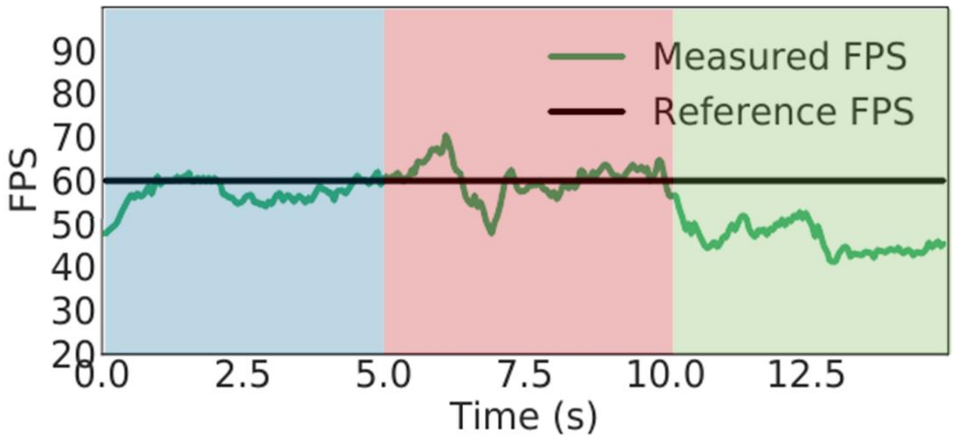


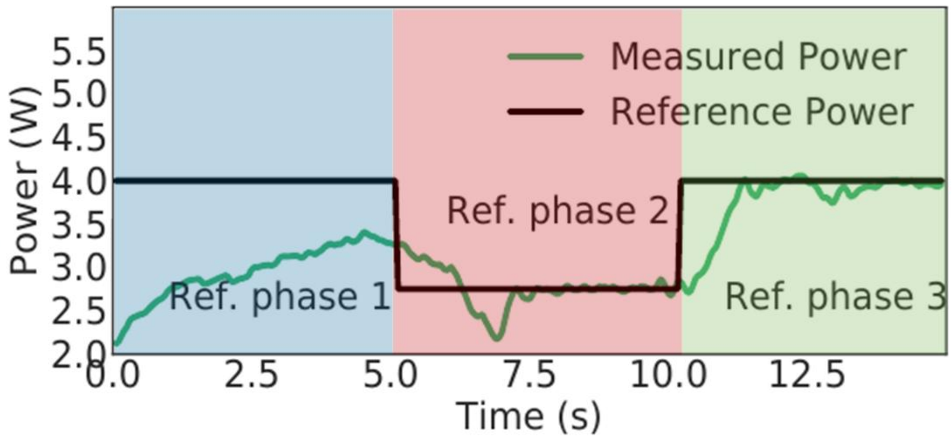
~ 40% more than necessary FPS in phase 1 (akin to MM-POW)

Longer settling time due to large MIMO controller.

Experimental Results -- Contoller Evaluation

- QoS task: x264
- Controller: SPECTR

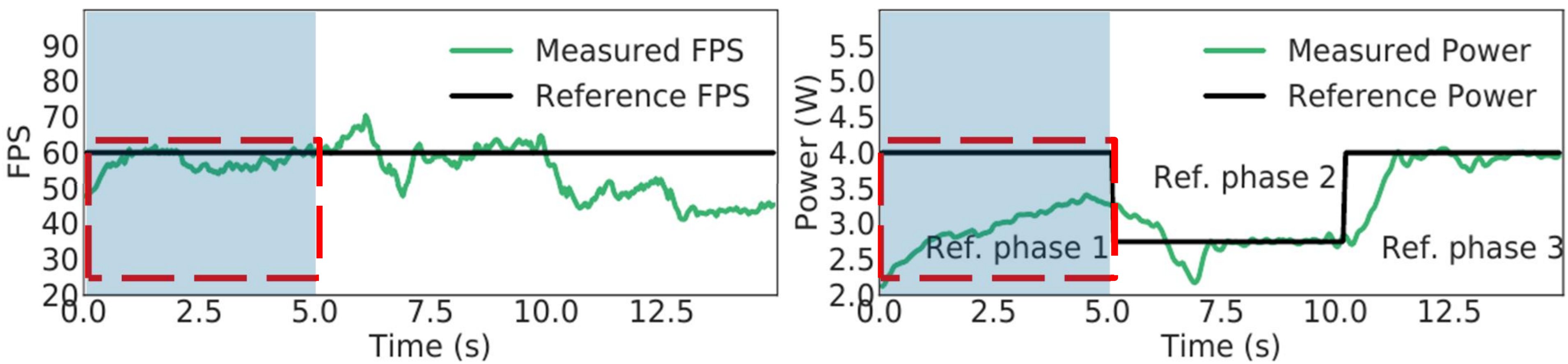




Let's Take a closer look at each phase

Experimental Results -- Contoller Evaluation

Safe Phase: QoS App only
SPECTR focuses on satisfying FPS with the minimum power



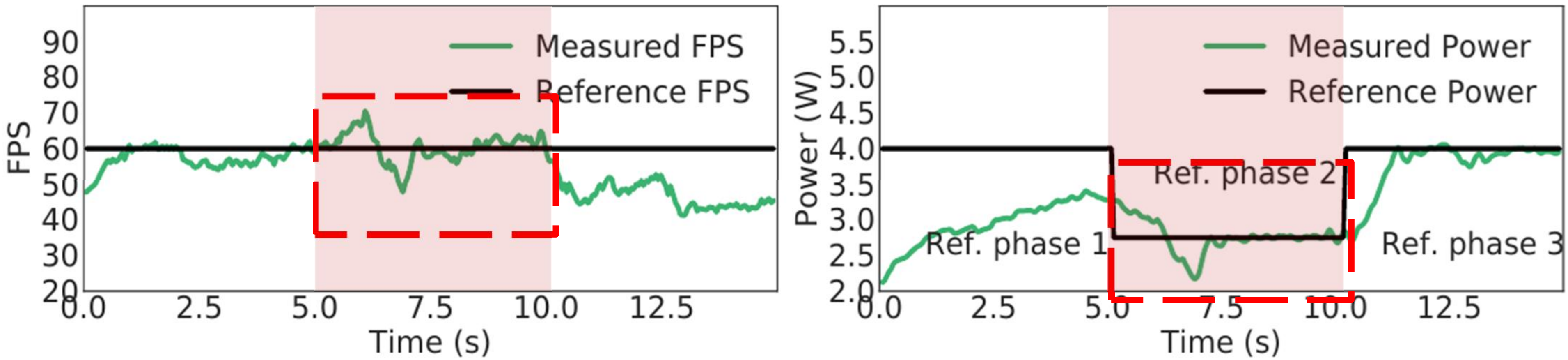
<5% FPS steady state error (minimal wasted perfomance)

Power below TDP

Experimental Results -- Contoller Evaluation

Emergency Phase: TDP reduced in response to thermal event

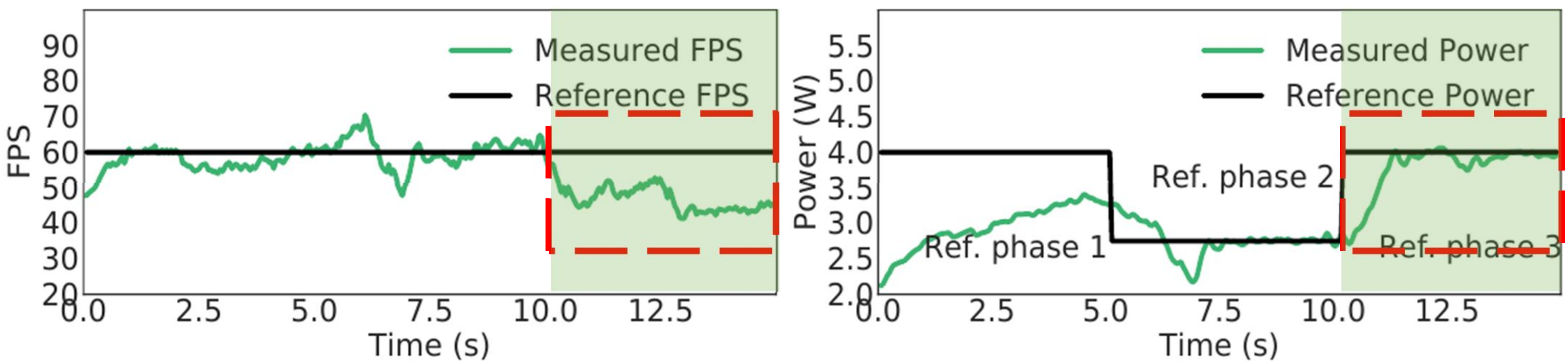
SPECTR satisfies the reference FPS and power



<5% FPS steady state error

Experimental Results -- Contoller Evaluation

Disturbance Phase: TDP returned to normal, background tasks added
SPECTR focuses on power capping



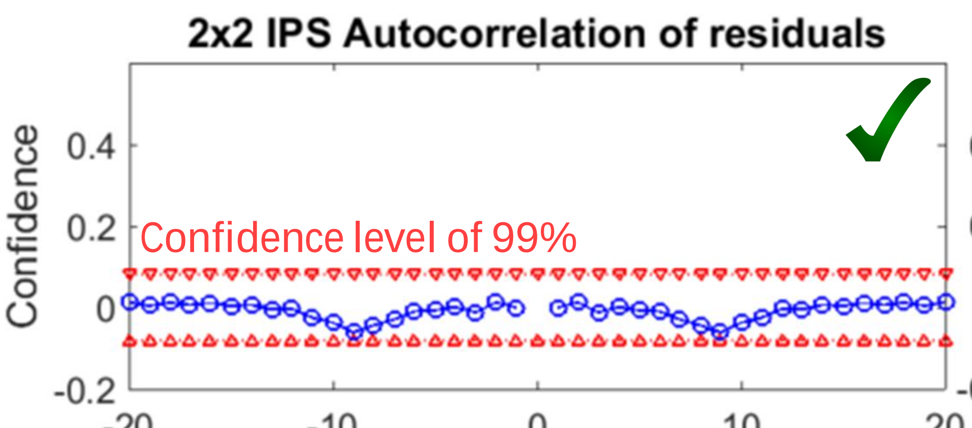
No TDP violations, but ~23% FPS steady state error

(impossible to track without violating TDP)

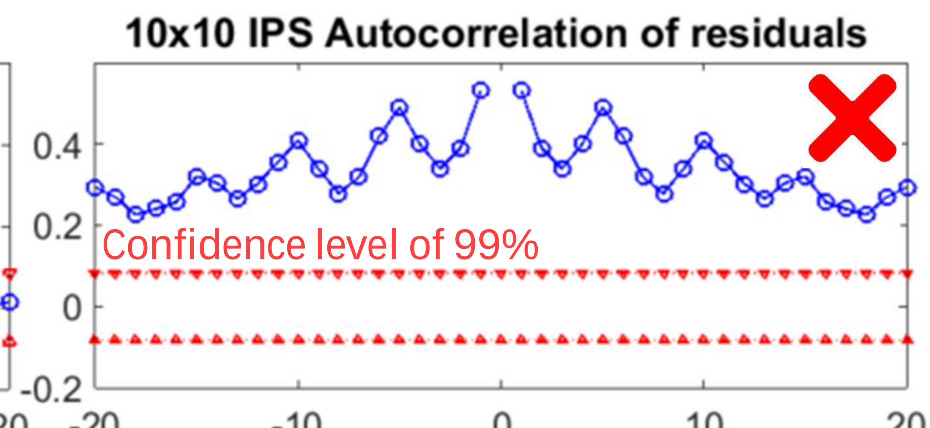
Experimental Results -- Scalability

- Accuracy of the identified system models of different sized MIMO controllers
- A model output within the confidence interval indicates that the deterministic component of the model output will be near the true output.

Within the confidence interval



Outside the confidence interval



Black-box system identification is not feasible for large and complex MIMO systems!

Other Results in the Paper

- A detailed Contoller Evaluation on:
  - PARSEC applications: bodytrack, canneal, streamcluster
  - Data-intensive machine learning workloads: k-means, KNN, least squares, linear regression

- Model Accuracy Analysis of different sized MIMO controllers:
  - 2x2 -> feasible and efficient
  - 4x2 -> feasible but sluggish
  - 10x10 -> not feasible

- Further discussion on:
  - Controller responsiveness (settling time)
  - Controller stability

SPECTR Outline

- Motivation
- MIMO Control Theory for Coordinated Management
- Unaddressed Challenges in Resource Management
  - Autonomy
  - Scalability
- Supervisory Control Theory (SCT) via SPECTR
  - Scalability and Autonomy through SCT
  - SPECTR Overview
  - Case Study
  - Supervisor Synthesis Process
- Results
- **Summary**

Summary

- Resource managers need to offer 1) robustness, 2) formalism, 3) efficiency, 4) coordination, 5) scalability, and 6) autonomy all together

- SPECTR offers them all!
  - SPECTR adapts to changing goals at rutime
  - SPECTR decomposes the control problems to manage its complexity

- SPECTR achieves up to 8x and 6x better target QoS and power tracking over state-of-the-art, respectively (in our case study)

- SPECTR is applicable to any resource type and objective as long as the management problem can be modeled using dynamical systems theory

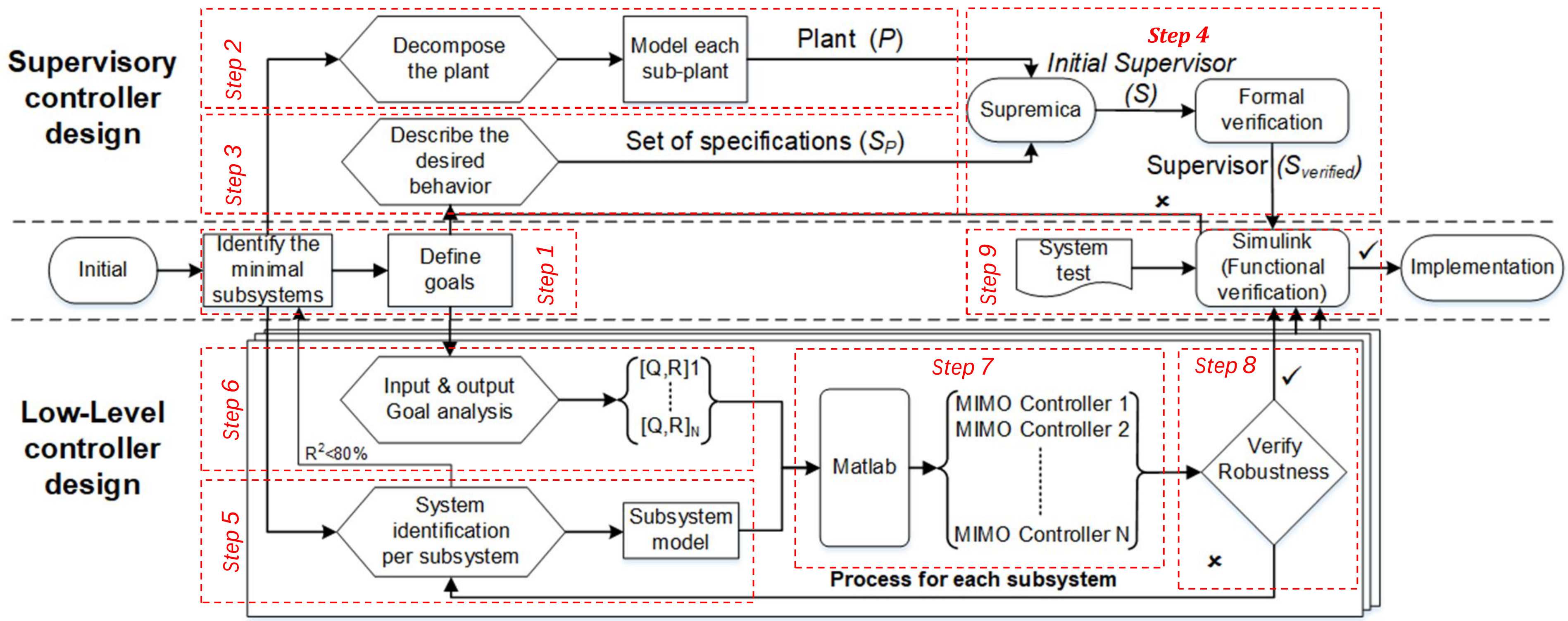
SPECTR

Formal Supervisory Control and Coordination for Many-core Systems Resource Management

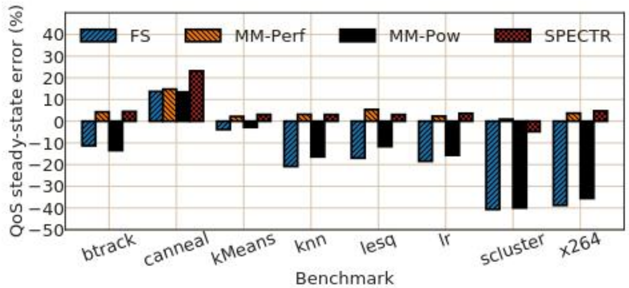
Amir M. Rahmani   Bryan Donyanavard   Tiago Mück   Kasra Moazzemi
Axel Jantsch   Onur Mutlu   Nikil Dutt

23rd ACM International Conference on Architectural Support for Programming Languages and Operating Systems
Williamsburg, VA, March 2018

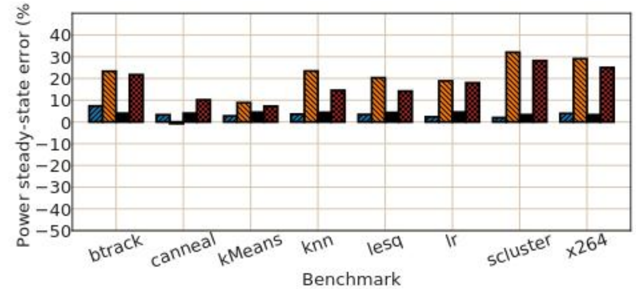
SPECTR Design Flow



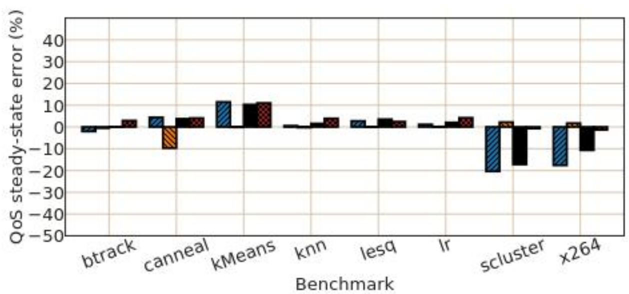
Steady-state Error for All Benchmarks



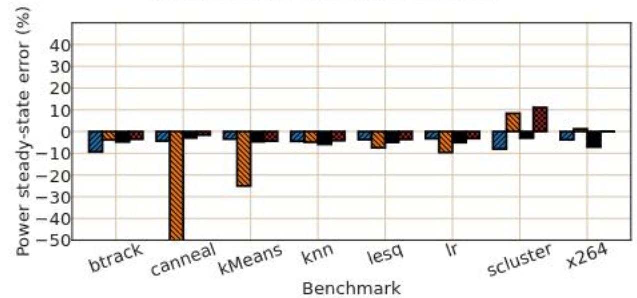
(a) QoS steady-state error in Phase 1.



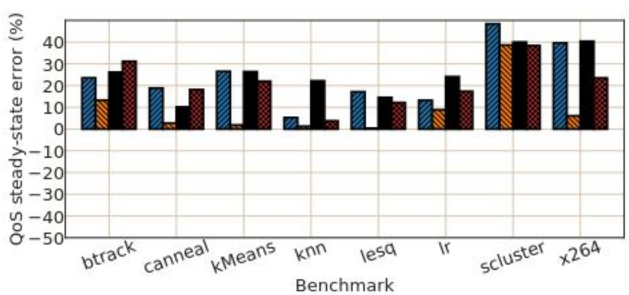
(b) Power steady-state error in Phase 1.



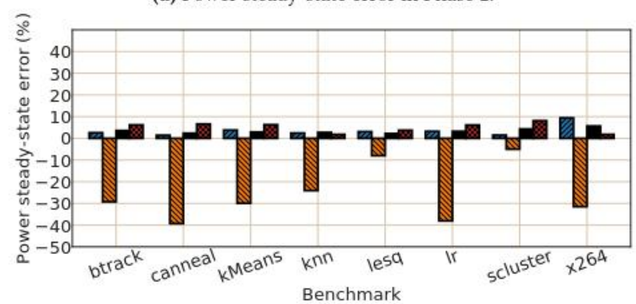
(c) QoS steady-state error in Phase 2.



(d) Power steady-state error in Phase 2.



(e) QoS steady-state error in Phase 3.



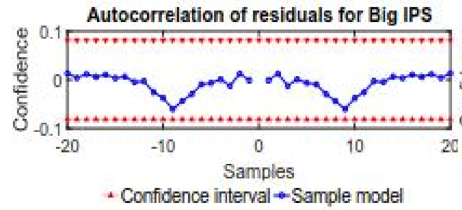
(f) Power steady-state error in Phase 3.

Steady-state error for all benchmarks, grouped by phase. A negative value indicates the amount of power/QoS exceeding the reference value (bad), a positive value indicates the amount of power saved (good) or QoS degradation (bad)

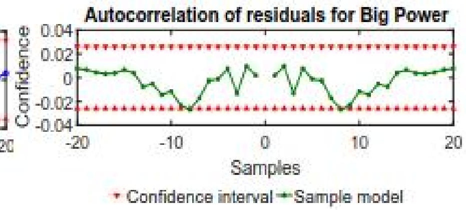
Model Accuracy

- Autocorrelation of residuals for identified system models of different sized MIMO controllers.

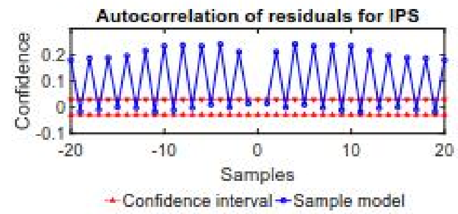
- We show a single performance and power output for each modeled system across multiple sample inputs.



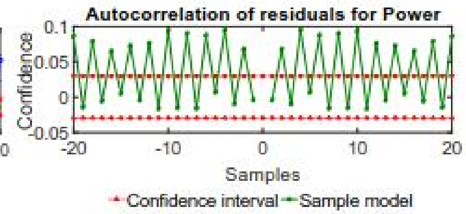
(a) 2×2 system model for the Big cluster controller of SPECTR, total IPS output.



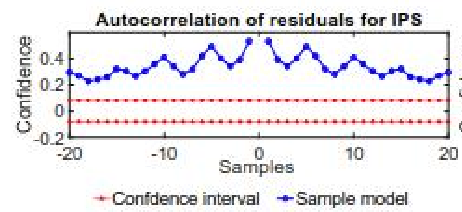
(b) 2×2 system model for the Big cluster controller of SPECTR, total power output.



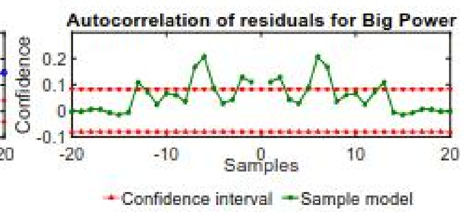
(c) 4×2 system model for the FS controller, total IPS output.



(d) 4×2 system model for the FS controller, total power output.



(e) 10×10 system model for a large-system controller (e.g., Fig. 4), single-core IPS output.



(f) 10×10 system model for a large-system controller (e.g., Fig. 4), Big cluster power output.