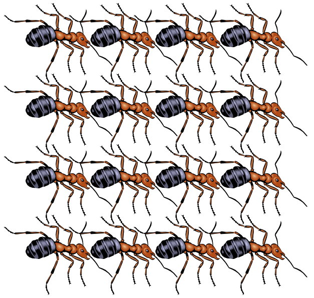
The SEEC Framework and Runtime System



Henry Hoffmann

MIT CSAIL

http://heartbeats.csail.mit.edu/

High Performance Embedded Computing Workshop

September 21-22, 2011

This work was funded by the U.S. Government under the DARPA UHPC program. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

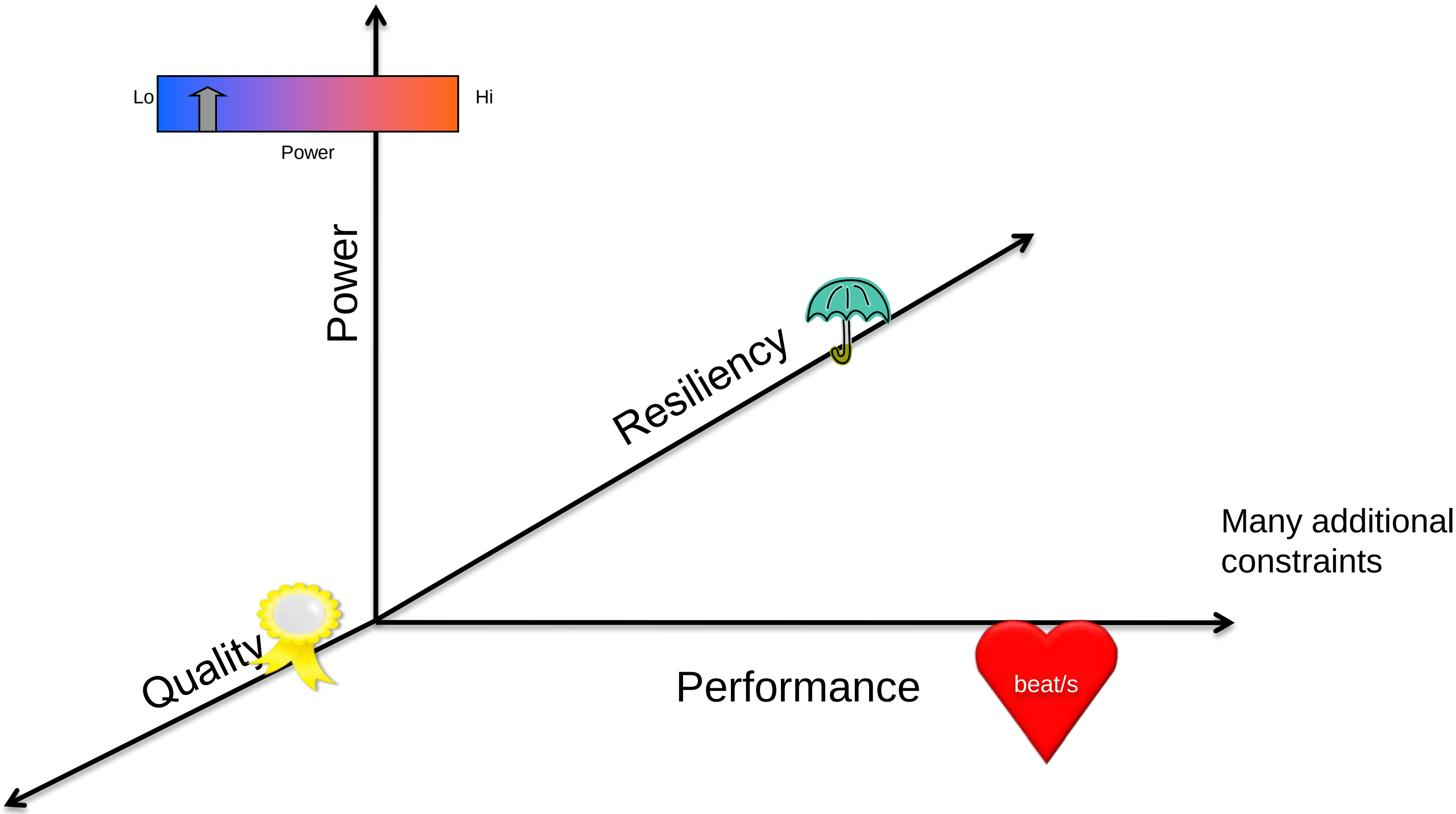
In the beginning…*

Application programmers had one goal:

Performance

*The beginning, in this case, refers to the beginning of my career (1999)

But Modern Systems Have Increased the Burden on Application Programmers

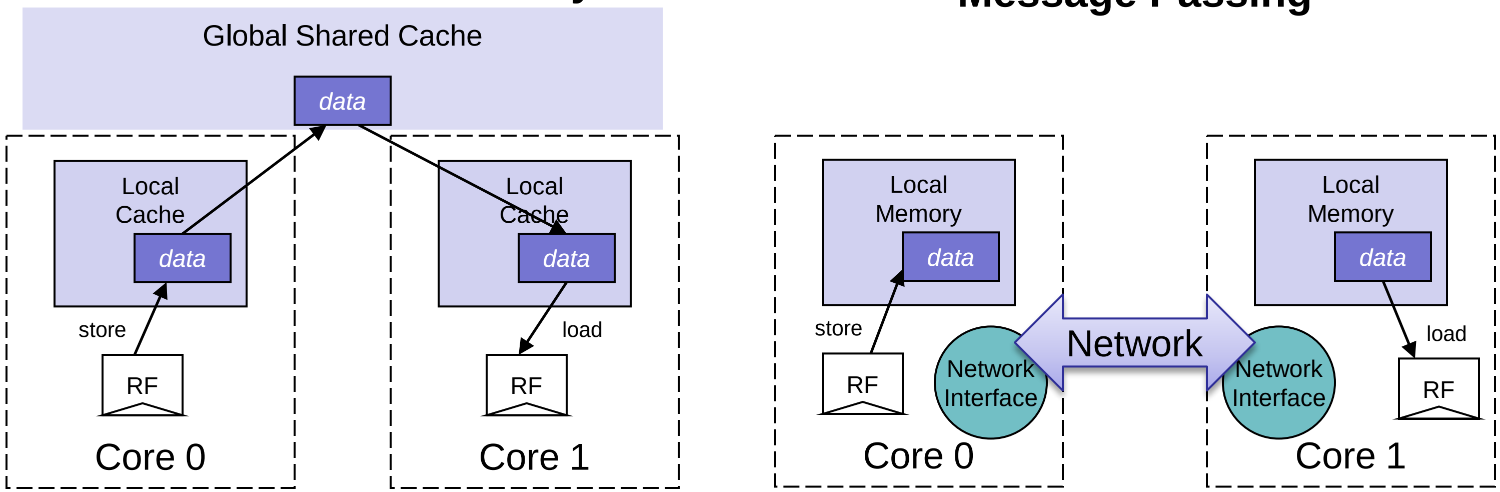


Even worse, constraints can change dynamically

E.g. power cap, workload fluctuation, core failure

Most Programming Models Designed for Performance

Coherent Shared Memory | **Message Passing**

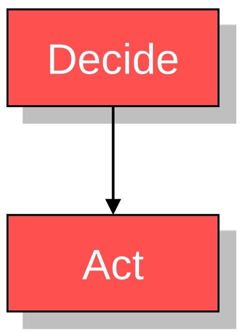


	Coherent Shared Memory	Message Passing
Concurrency	Multi-threaded	Multi-process
Communication	Through Memory	Through Network
Coordination	Locks	Messages
Control	Procedural	Procedural

Procedural control insufficient to meet the needs of modern systems

SEEC Replaces Procedural Control with Self-Aware Control

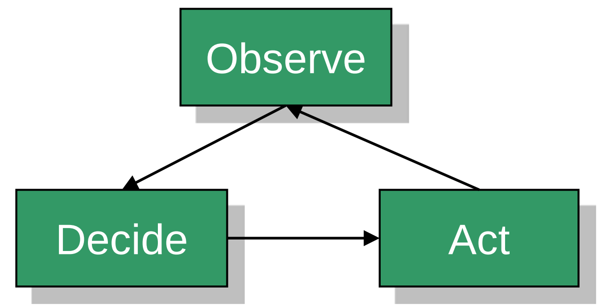
Procedural Control



- Run in open loop
- Assumptions made at design time
- Based on guesses about future

- Application optimized for system
- No flexibility to adapt to changes

Self-Aware Control



- Run in closed loop
- Understand user goals
- Monitor the environment

+System optimizes for application
+Flexibly adapt behavior

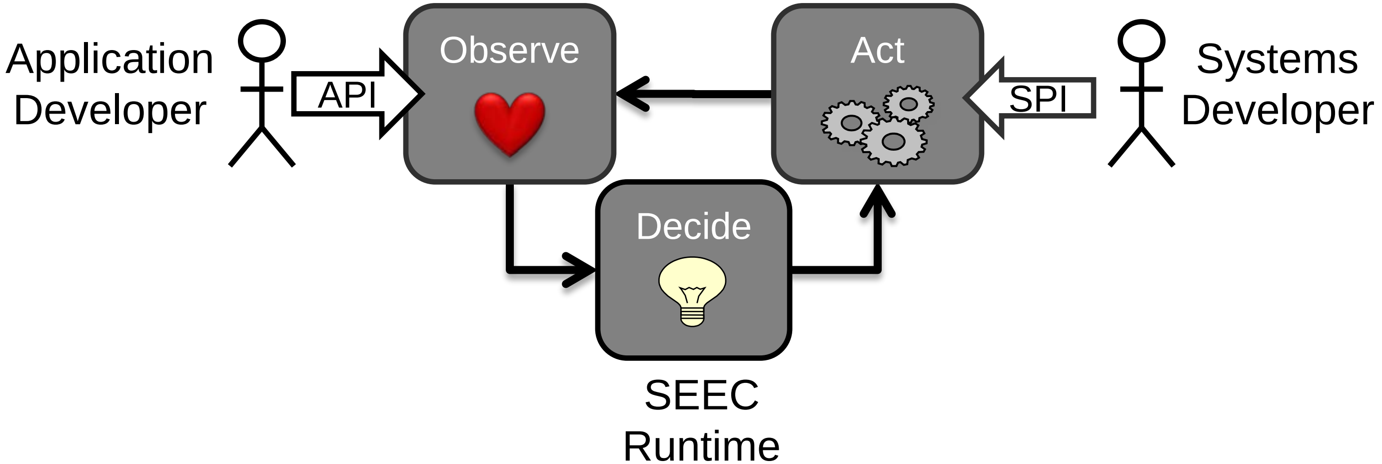
The self-aware model allows the system to solve constrained optimization problems dynamically

Outline

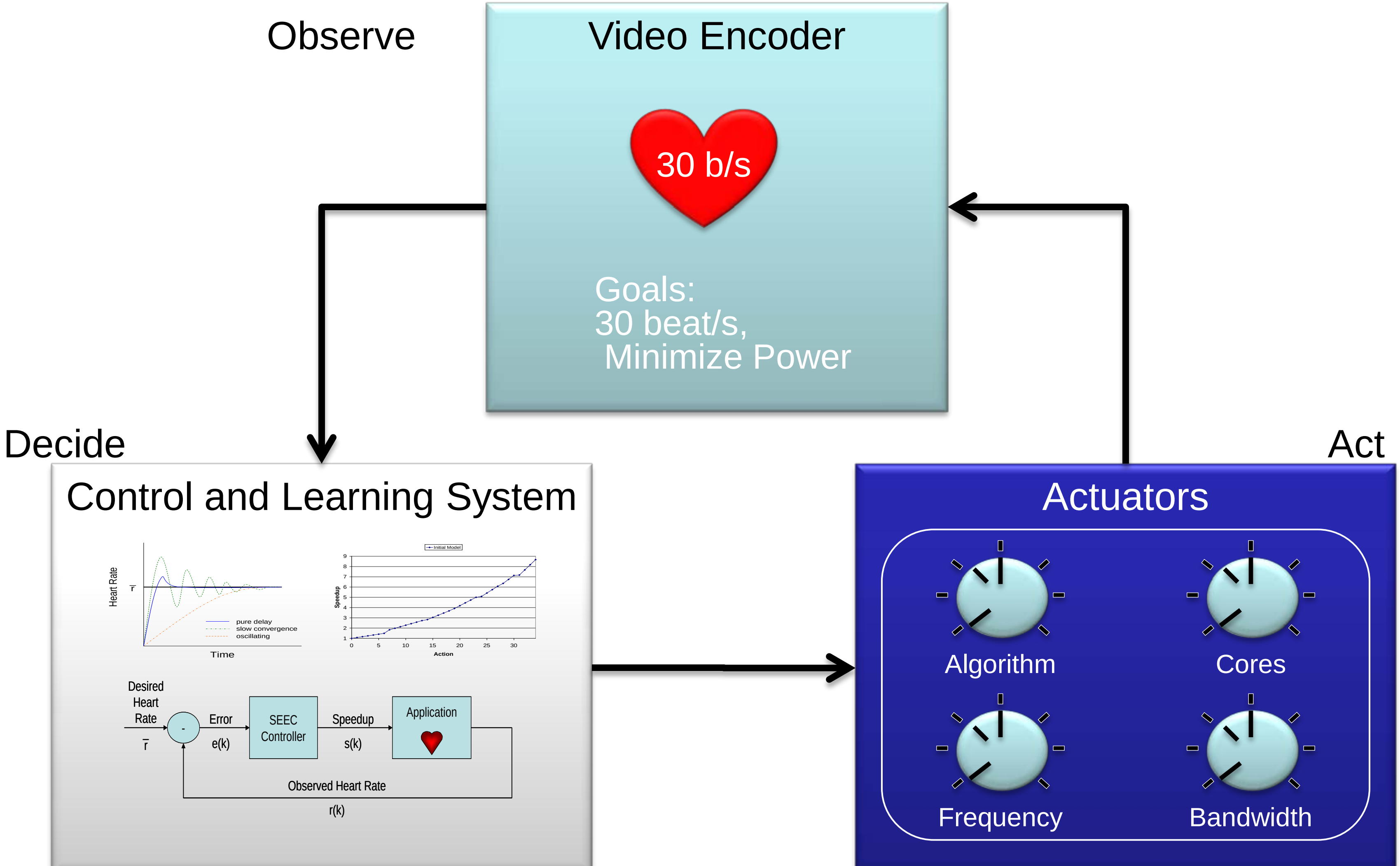
- Introduction/Motivation
- The SEEC Model and Implementation
- Experimental Validation
- Conclusions

The SElf-awarE Computing (SEEC) Model

- Goal:
  Reduce programmer burden by continuously optimizing online
- Key Features:
  1. ***Decoupled Approach***:
     - Applications explicitly state goals and progress
     - System software and hardware state available actions
     - The SEEC runtime system dynamically selects actions to maintain goals
  2. ***General and Extensible***:
     - New applications can be supported without training
     - New actions can be added without redesign and reimplementation



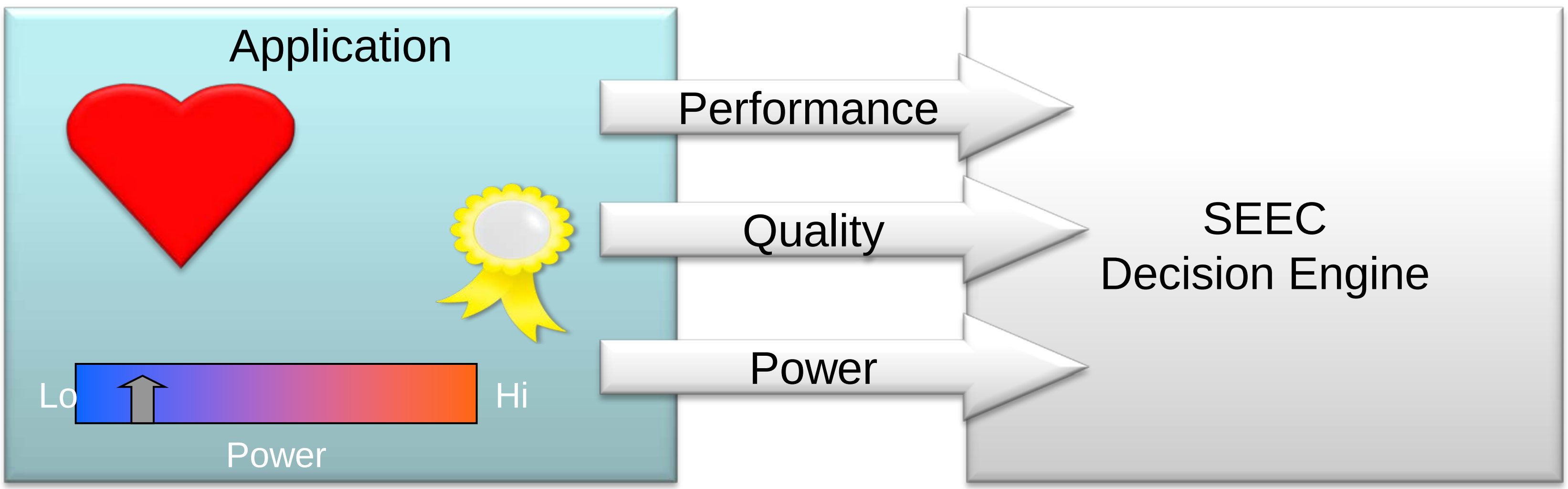
Example Self-Aware System Built from SEEC



Roles in SEEC's Decoupled Model

	Application Developer	Systems Developer	SEEC Runtime System
Observe	Express application goals and progress (e.g. frames/ second)		Read goals and performance
Decide			Determine how to adapt (e.g. How much to speed up the application)
Act		Provide a set of actions and a callback function (e.g. allocation of cores to process)	Initiate actions based on results of decision phase

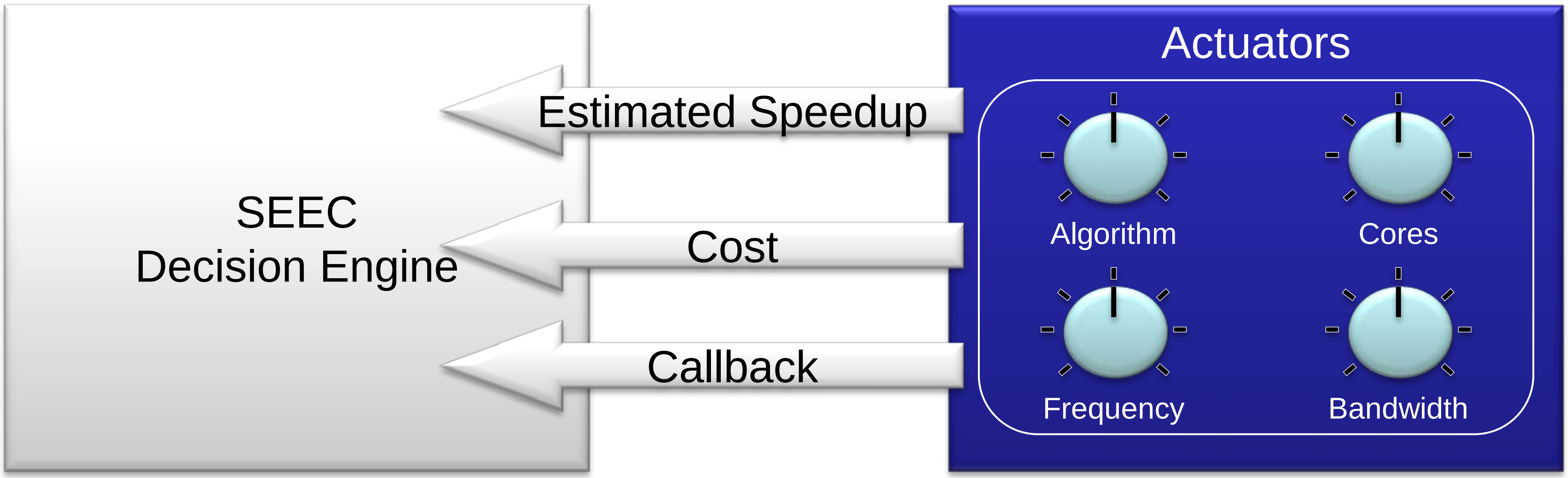
Registering Application Goals



- Performance
  - Goals: target heart rate and/or latency between tagged heartbeats
  - Progress: issue heartbeats at important intervals
- Quality
  - Goals: distortion (distance from application defined nominal value)
  - Progress: distortion over last heartbeat
- Power
  - Goals: target heart rate / Watt and/or target energy between tagged heartbeats
  - Progress: Power/energy over last heartbeat interval

Research to date focuses on meeting performance while minimizing power/maximizing quality

Registering System Actions

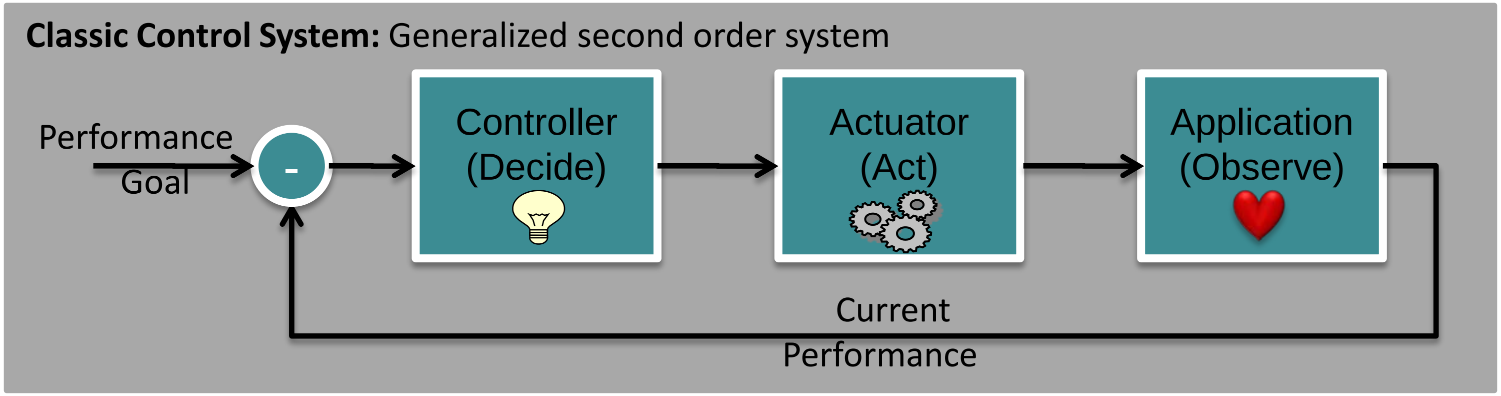


Each action has the following attributes:

- Estimated Speedup
  - Predicted benefit of taking an action
- Cost
  - Predicted downside of taking an action
  - Axis for cost (accuracy, power, etc.)
- RPC handle
  - A function that takes an id an implements the associated action
  - This is currently subject to change/redesign

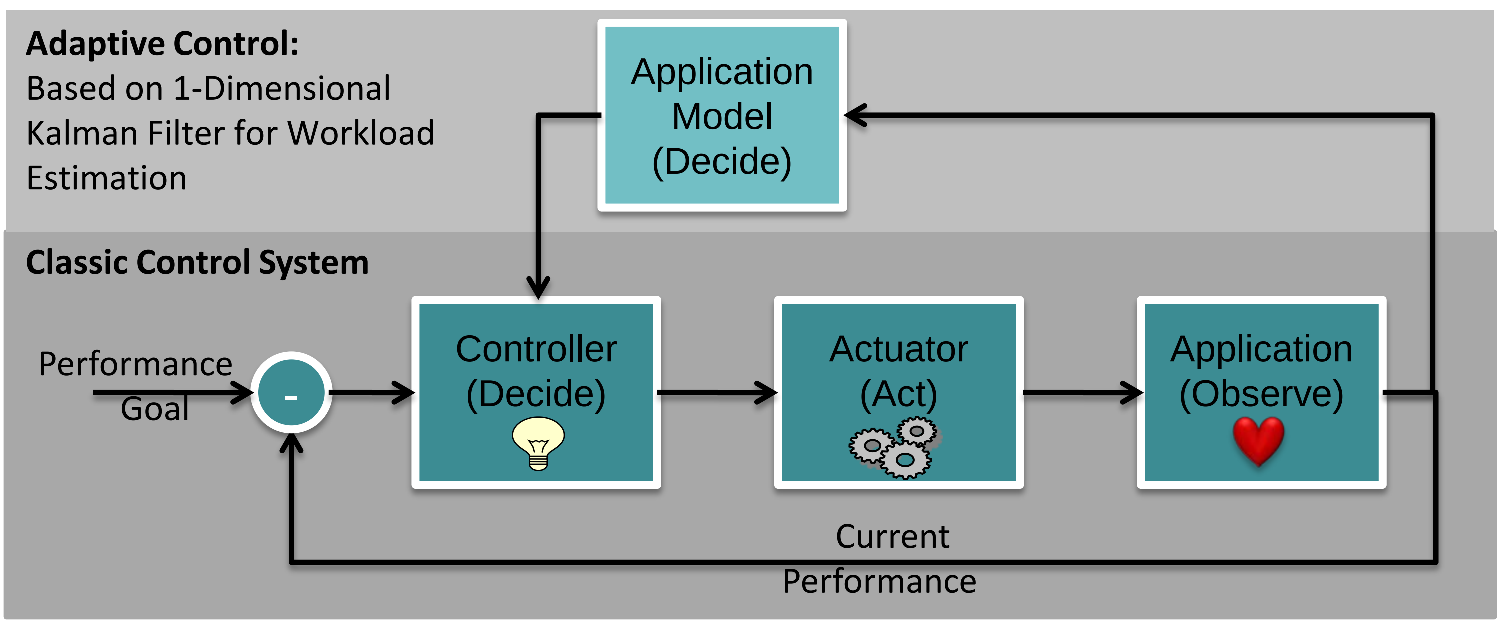
The SEEC Decision Engine
(A general, extensible approach)

- Pros: Simple, Analyzable, Works well for profiled applications
- Cons: Lack of generality for unseen applications



The SEEC Decision Engine
(A general, extensible approach)

- Pros: Adapts to unseen applications
- Cons: Assumes (relative) system models are correct
  Cannot support race-to-idle

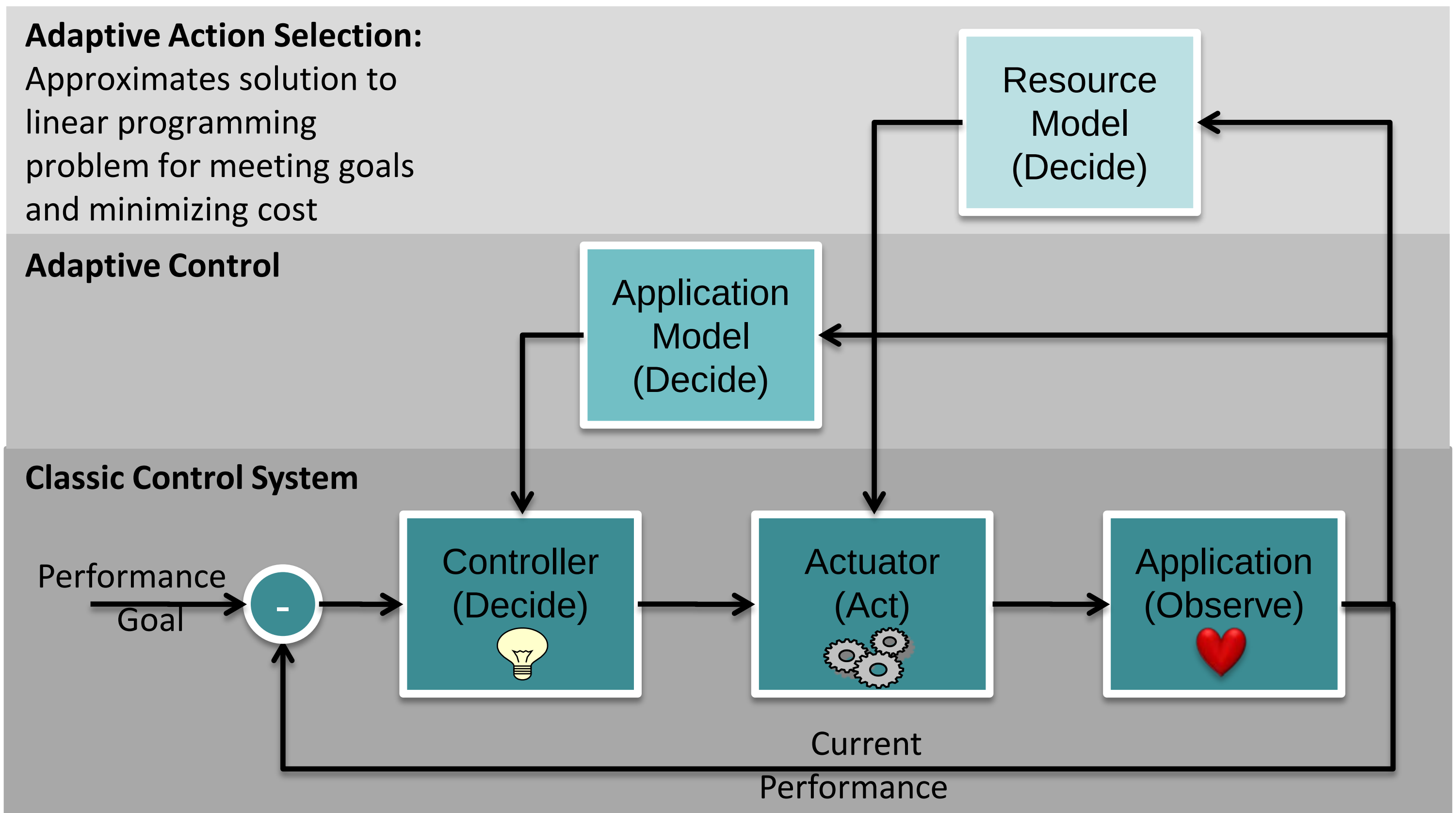


The SEEC Decision Engine

(A general, extensible approach)

Decide

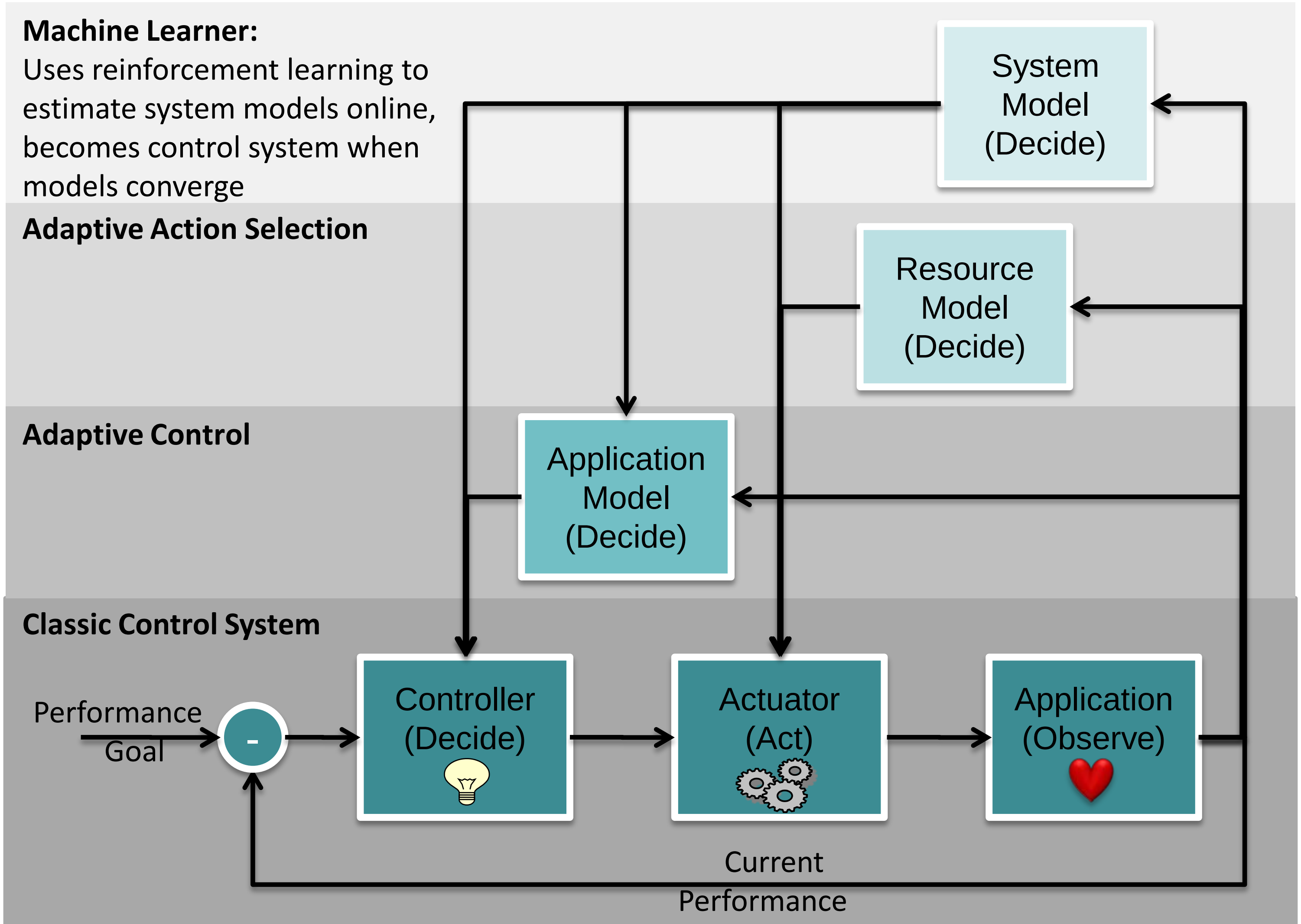
- Pros: Supports race-to-idle and proportional allocation
- Cons: May overprovision due to system model errors



The SEEC Decision Engine

(A general, extensible approach)

Decide



Outline

- Introduction/Motivation
- The SEEC Model and Implementation
- ➡ Experimental Validation
- Conclusions

Systems Built with SEEC

System	Actions	Tradeoff	Benchmarks
Dynamic Loop Perforation	Skip some loop iterations	Performance vs. Quality	7/13 PARSECs
Dynamic Knobs	Make static parameters dynamic	Performance vs. Quality	bodytrack, swaptions, x264, SWISH++
Core Scheduler	Assign N cores to application	Compute vs. Power	PARSEC
Clock Scaler	Change processor speed	Compute vs. Power	PARSEC
Bandwidth Allocator	Assign memory controllers to application	Memory vs. Power	STREAM, PARSEC
Power Manager	Combination of the three above	Performance vs. Power	PARSEC, STREAM, simple test apps (mergesort, binary search)
Learned Models	Power Manager with speedup and cost learned online	Performance vs. Power	PARSEC
Multi-App Control	Power Manager with multiple applications	Performance vs. Power and Quality for multiple applications	Combinations of PARSECs

Systems Built with SEEC

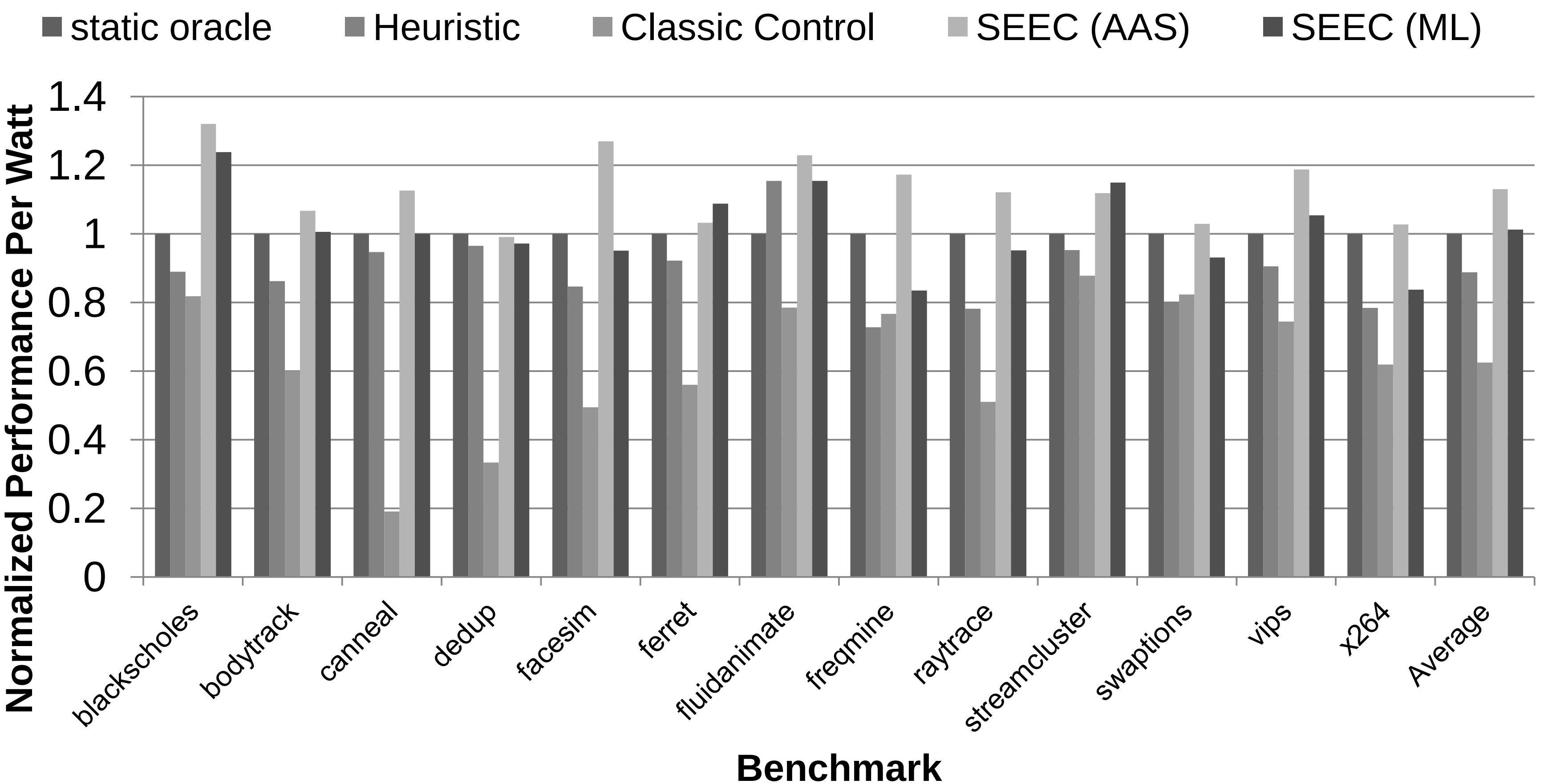
System	Actions	Tradeoff	Benchmarks
Dynamic Loop Perforation	Skip some loop iterations	Performance vs. Quality	7/13 PARSECs
Dynamic Knobs	Make static parameters dynamic	Performance vs. Quality	bodytrack, swaptions, x264, SWISH++
Core Scheduler	Assign N cores to application	Compute vs. Power	PARSEC
Clock Scaler	Change processor speed	Compute vs. Power	PARSEC
Bandwidth Allocator	Assign memory controllers to application	Memory vs. Power	STREAM
Power Manager	**Combination of the three above**	**Performance vs. Power**	**PARSEC, STREAM, extra test apps (mergesort, binary search)**
Learned Models	**Power Manager with speedup and cost learned online**	**Performance vs. Power**	**PARSEC**
Multi-App Control	**Power Manager with multiple applications**	**Performance vs. Power and Quality for multiple applications**	**Combinations of PARSECs**

Constrained Optimization: Managing Performance/Watt for PARSEC

Optimize performance/Watt on multiple machines

Application Goals	Maintain performance, minimize power
System Actions	Allocate cores Allocate clock speed Allocate memory bandwidth
Experiment	Execute on two machines (w/ different power profiles) Compare SEEC to several other approaches including a static oracle

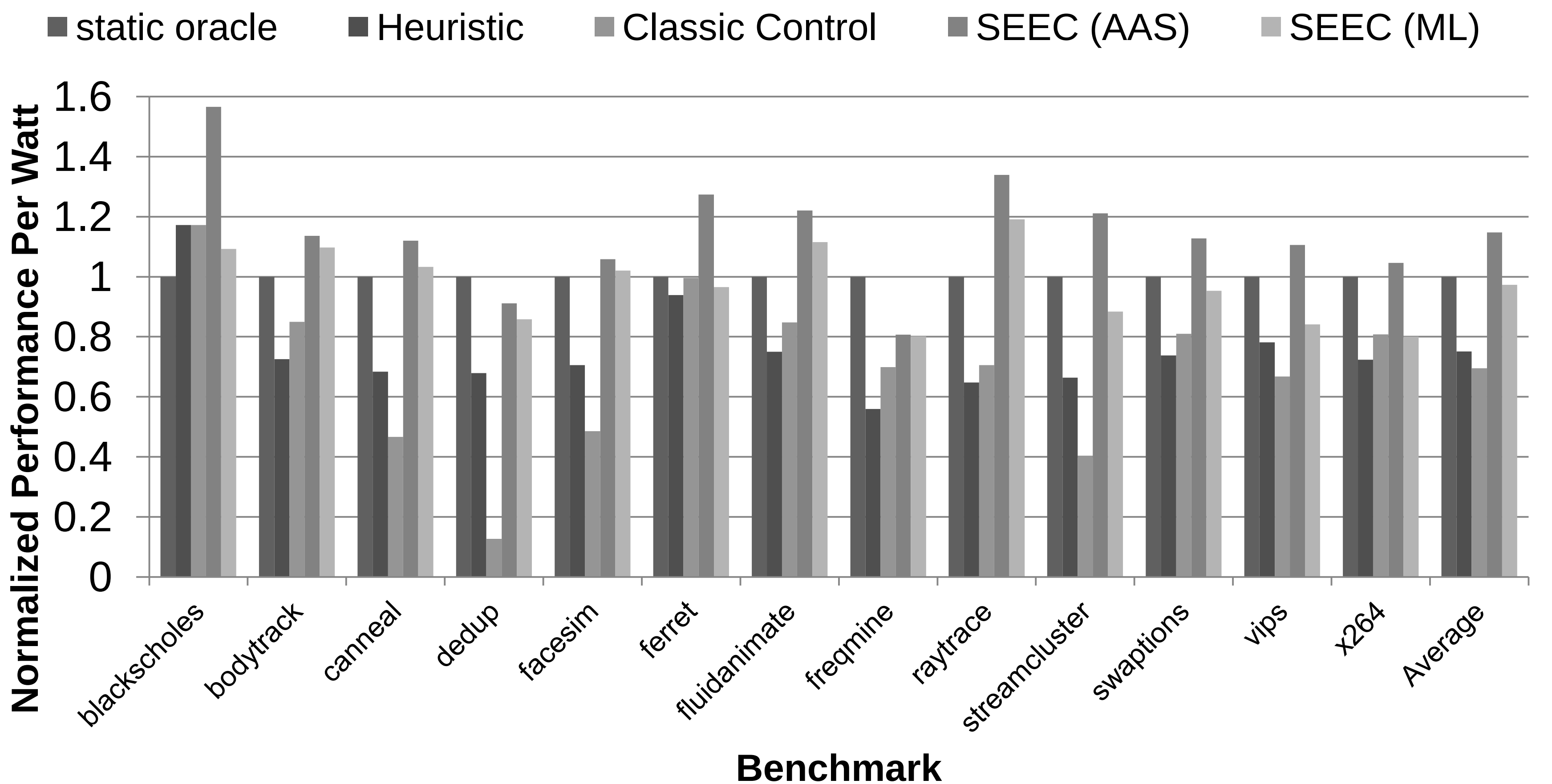
Performance/Watt for PARSEC
(On server with low idle power)



SEEC beats the static oracle by adjusting to phases within an application and recognizing when to race-to-idle

Performance/Watt for PARSEC

(On server with high idle power)



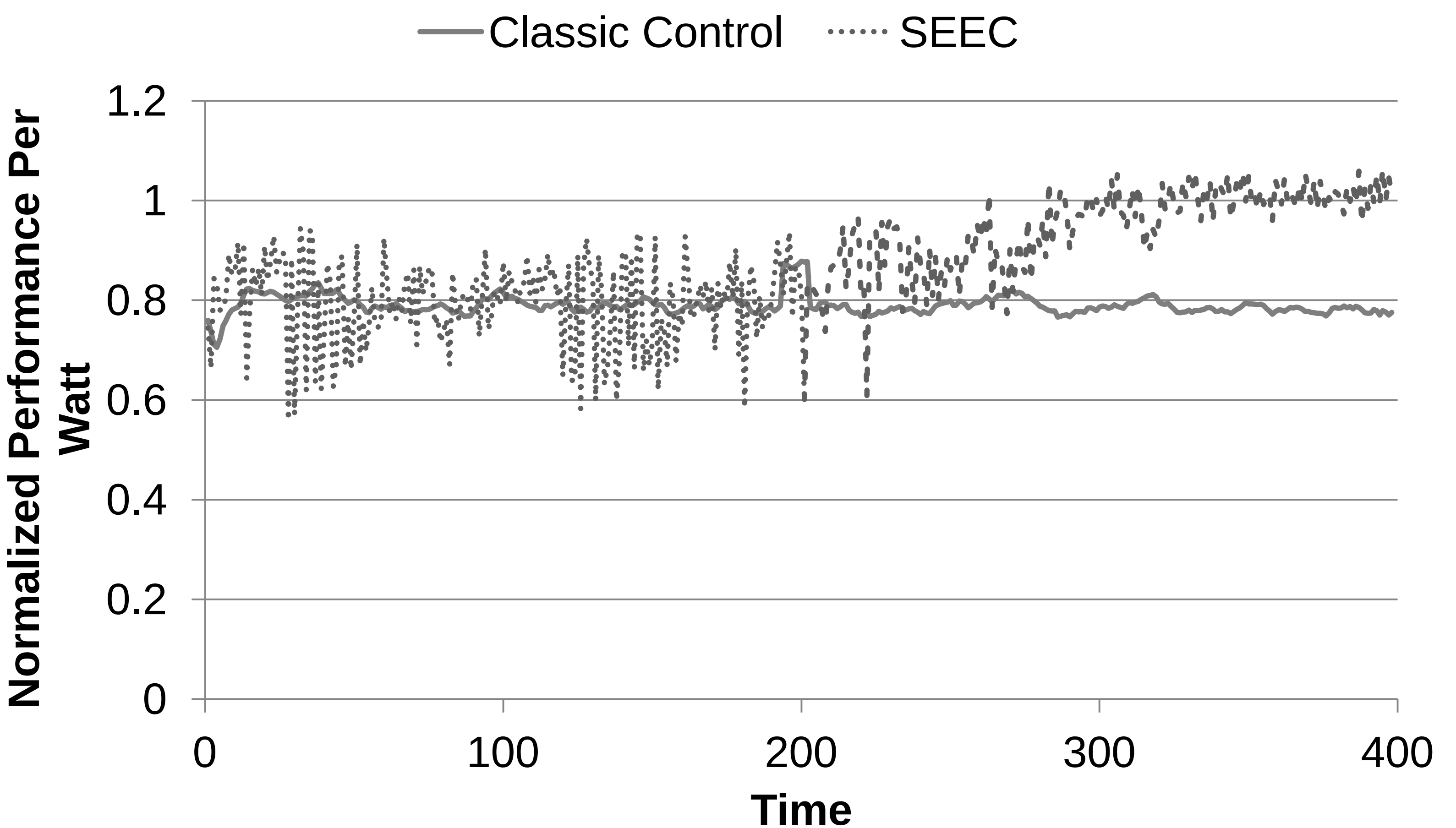
SEEC is able to beat the static oracle on a different machine without code changes

Learning Models Online

Adapt system behavior when initial models are wrong

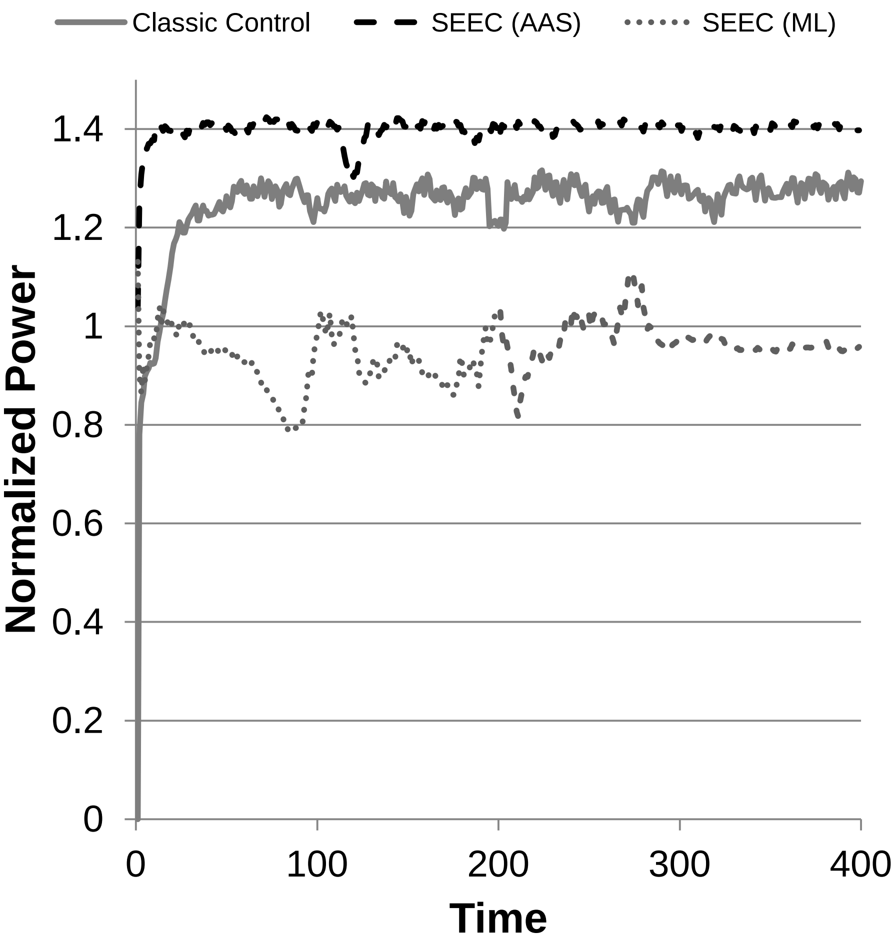
Application Goals	Minimize power consumption while meeting target performance
System Actions	Change cores, clock speed, and mem. bandwidth Initial models are incredibly optimistic (Assume linear speedup with any resource increase)
Experiment	Benchmark: STREAM Observe convergence time and performance/Watt for converged system

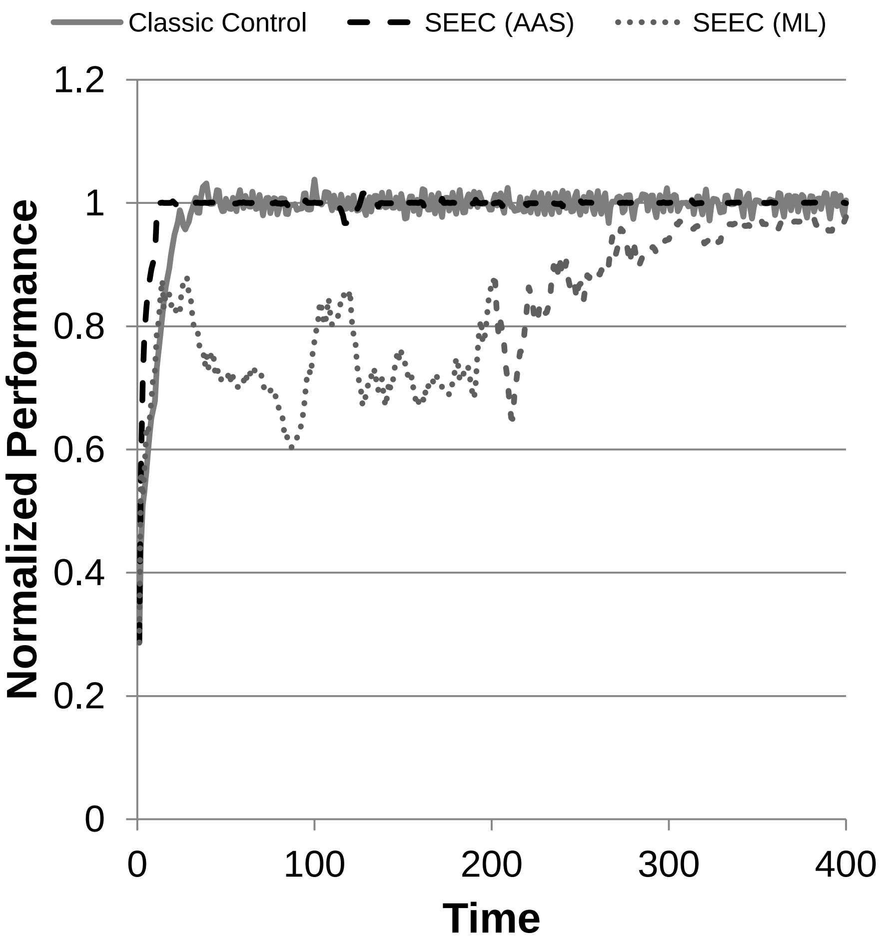
SEEC Can Learn Models Online



SEEC learns not to allocate too many compute resources to a memory bound application

When System Models Are Wrong
(Breakdown)





Adaptive Control achieves performance fastest, but wastes power.
ML reaches target performance slowest, but saves power

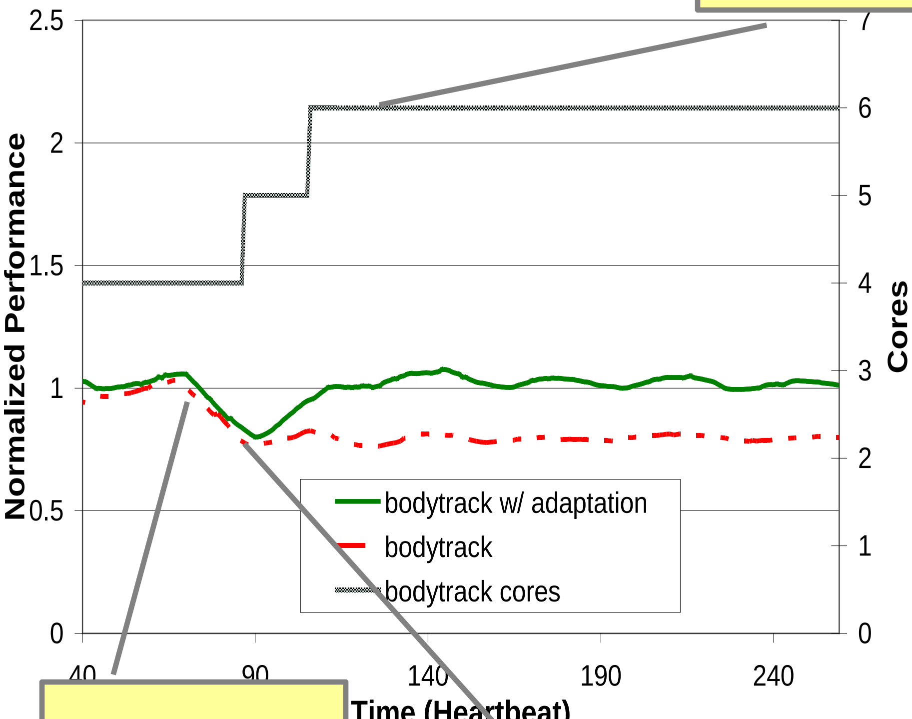
Managing Application and System Resources Concurrently

Manage multiple applications when clock frequency changes

Application Goals	**bodytrack:** maintain performance, minimize power **x264:** maintain performance, minimize quality loss
System Actions	Change core allocation to both applications Change x264's algorithms
Experiment	Maintain performance of both applications when clock frequency changes

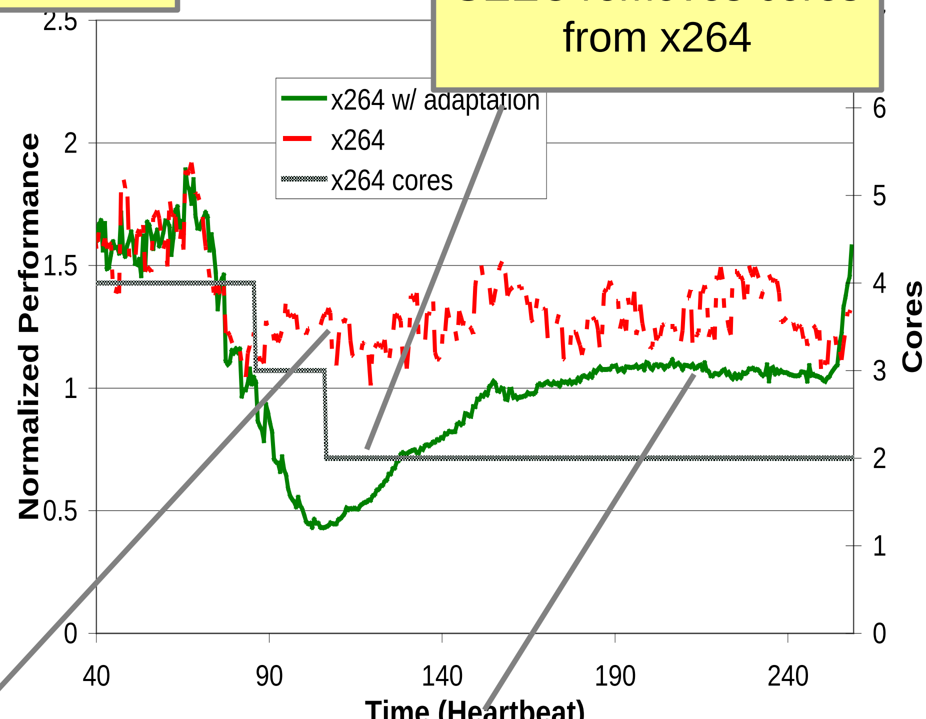
SEEC Management of Multiple Applications In Response to a Power Cap

bodytrack



SEEC allocates cores to bodytrack

x264



SEEC removes cores from x264

Clock drops 2.4-1.6GHz

w/o SEEC app **misses** goals

w/o SEEC app **exceeds** goals

SEEC adjusts algorithm to meet goals

Outline

- Introduction/Motivation
- The SEEC Framework
- Experimental Validation
- Conclusions

Conclusions

- SEEC is designed to help ease programmer burden
  - Solves resource allocation problems
  - Adapts to fluctuations in environment and application behavior
- SEEC has two distinguishing features
  - Decoupled Design
    - Incorporates goals and feedback directly from the application
    - Allows independent specification of adaptation
  - General and Extensible Decision Engine
    - Uses an adaptive second order control system to manage adaptation
- Demonstrated the benefits of SEEC in several experiments
  - Optimize performance per Watt for multiple benchmarks on multiple machines
  - Adapts algorithms and resource allocation as environment changes

Thanks

- DARPA's UHPC Program
- MIT, Politecnico di Milano, Freescale
- Martina Maggio, Marco D. Santambrogio, Jason E. Miller, Jonathan Eastep
- Stelios Sidiroglou, Sasa Misailovic, Michael Carbin
- Anant Agarwal, Martin Rinard, Alberto Leva, Jim Holt