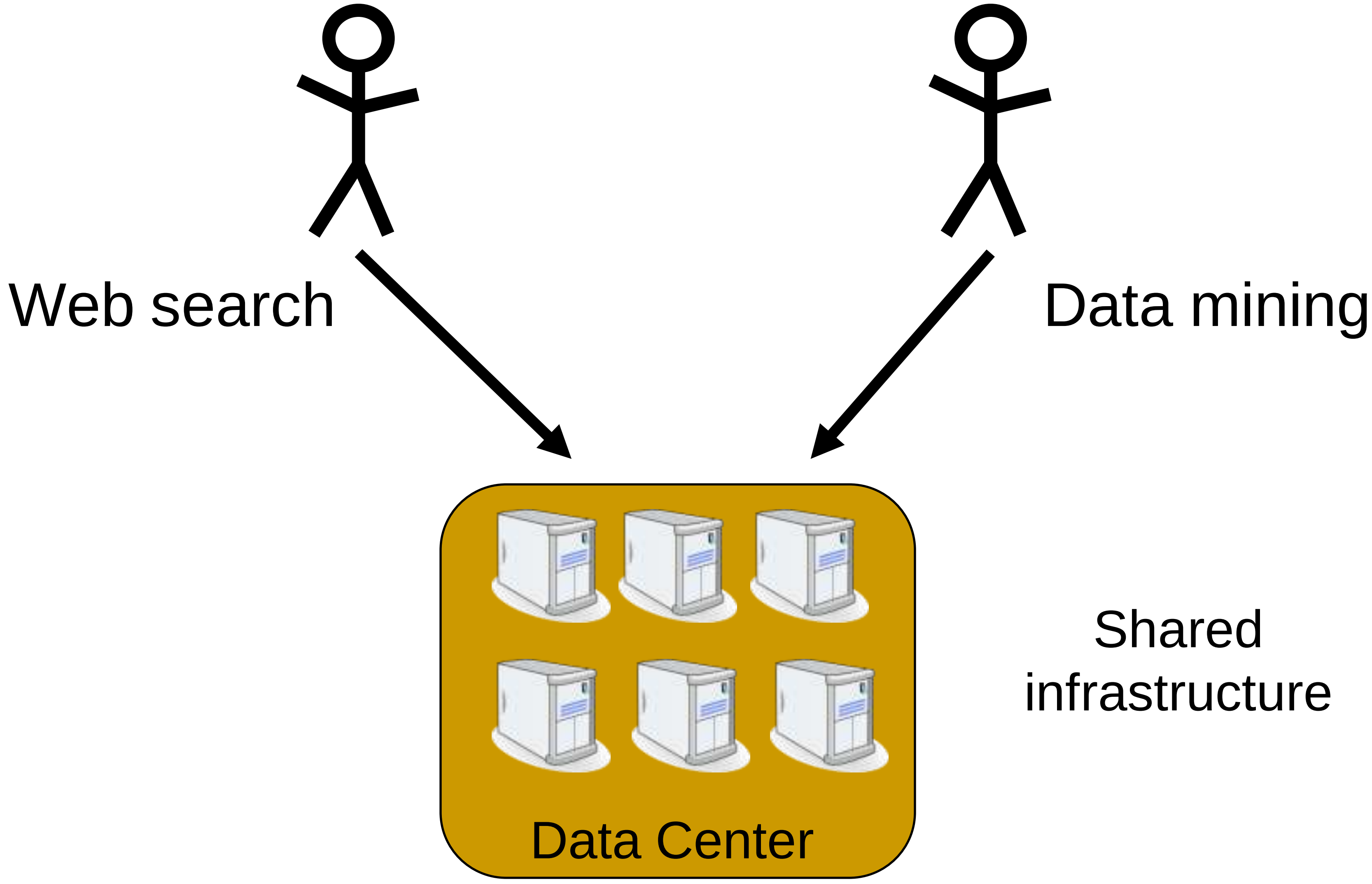
AutoControl: Automated Control of MultipleVirtualized Resources

Pradeep Padala, Karen Hou, Xiaoyun Zhu*, Mustfa Uysal[†], Zhikui Wang[†], Sharad Singhal[†], Arif Merchant[†], Kang G. Shin

University of Michigan, VMware* and HP Labs[†]

Typical scenario in shared infrastructures



Application requirements

Web search

Fast searches

✓ Low response time

Data mining

Analyze large data

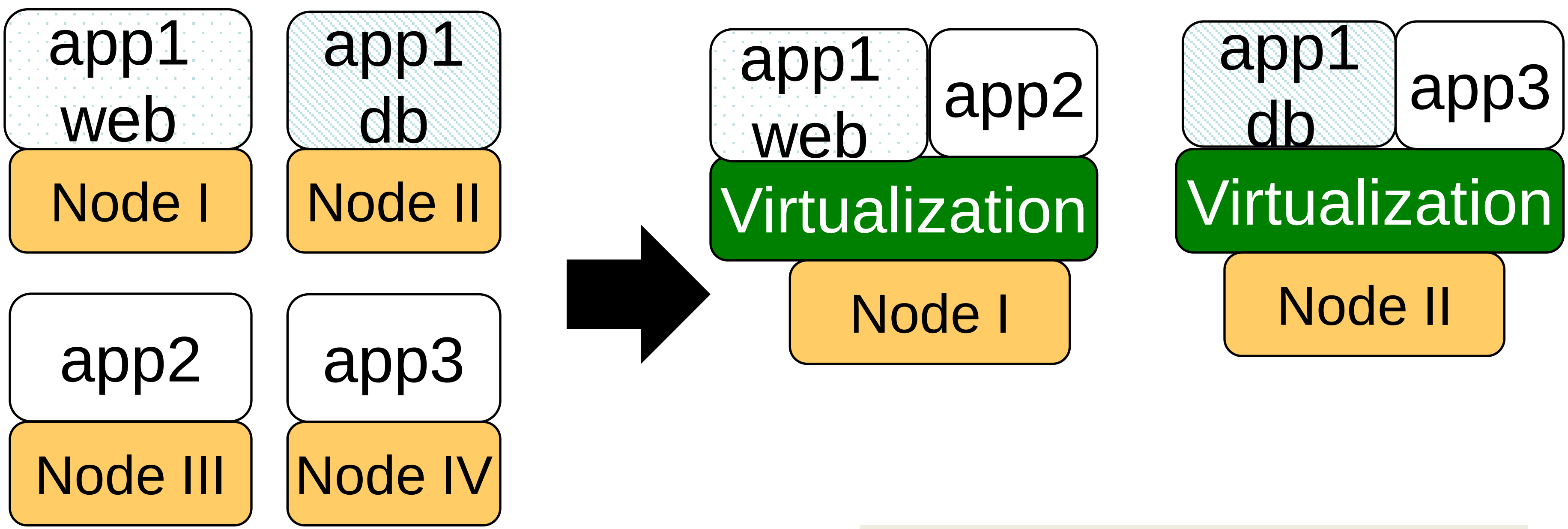
✓ High throughput

✓ QoS differentiation 3:1

Hosting applications

Physical partitioning

Virtual data center

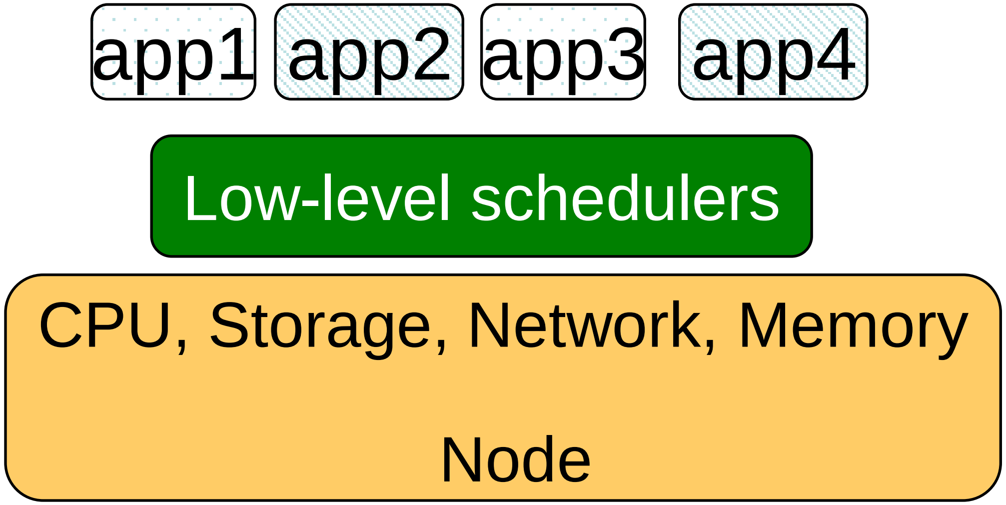


- × Wasteful
- × Difficult to manage

- ✓ Improved utilization
- ✓ Reduced costs
- ✓ High flexibility

Problem: How to allocate resources?

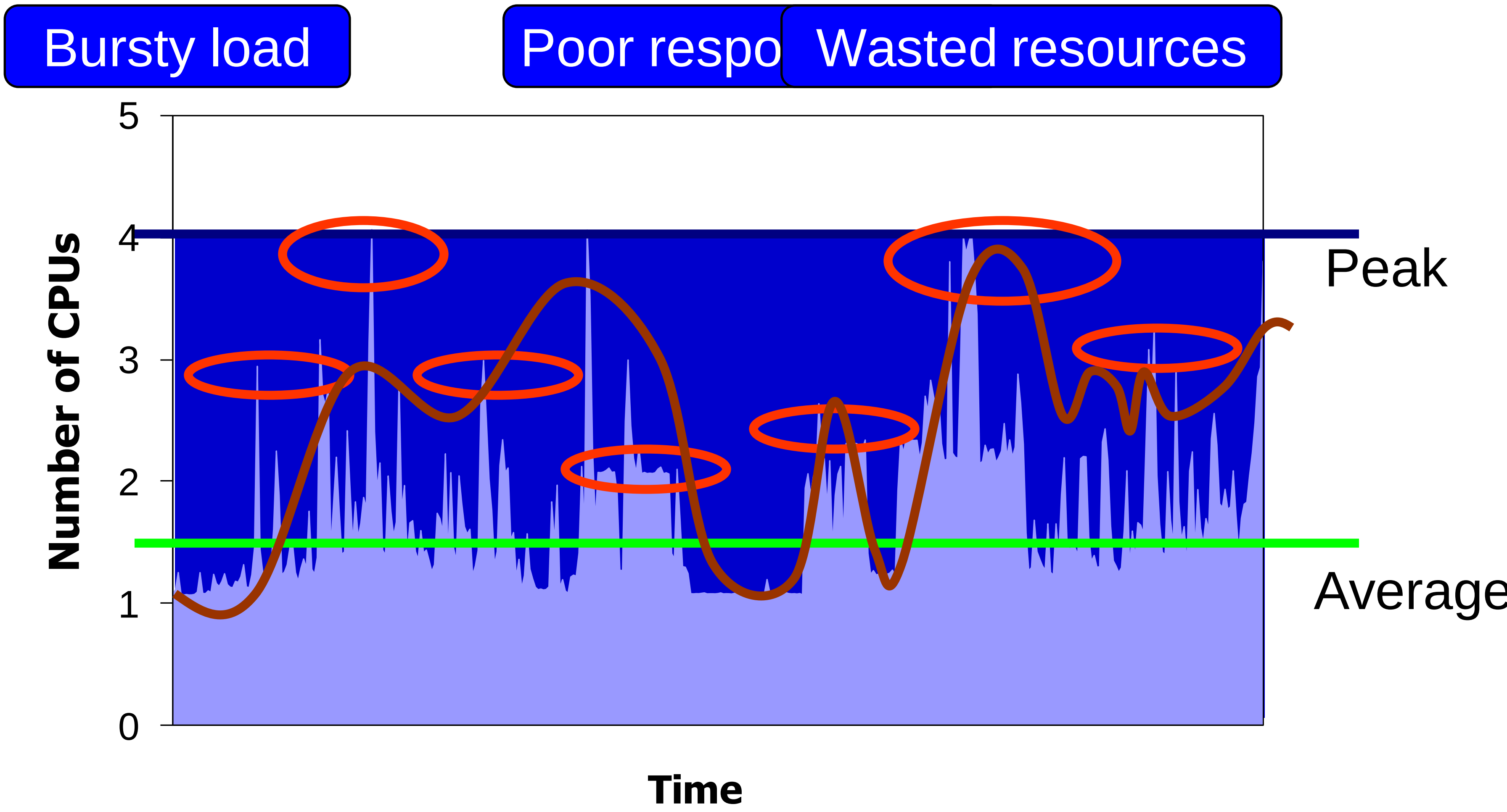
Approach I: Work-conserving mode



All applications can use as much resources as they require

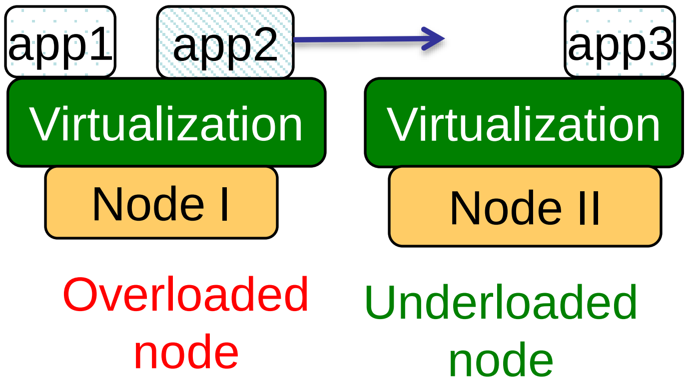
- Greedy applications cause SLO (service level objective) violations
- How to prioritize? – no differentiation
- How to use scheduler mechanisms to meet targets?

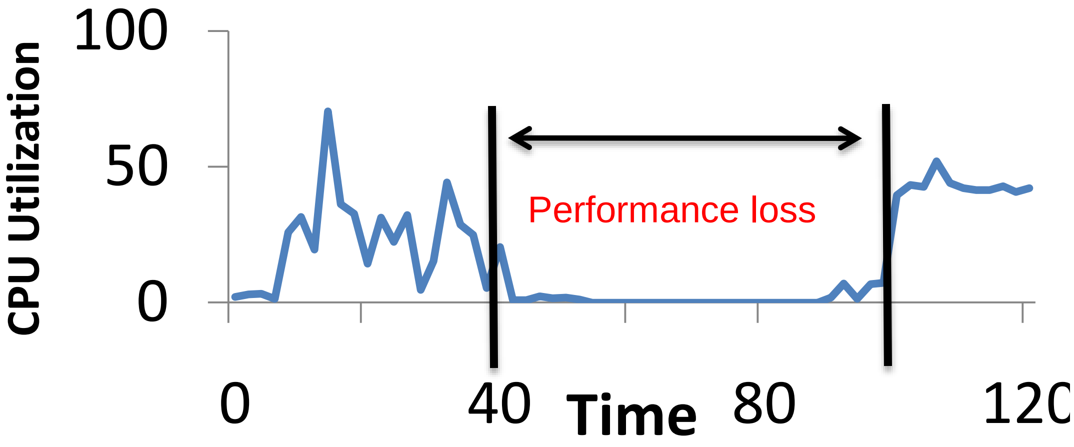
Approach II: Static allocation



Finding the right share is hard!

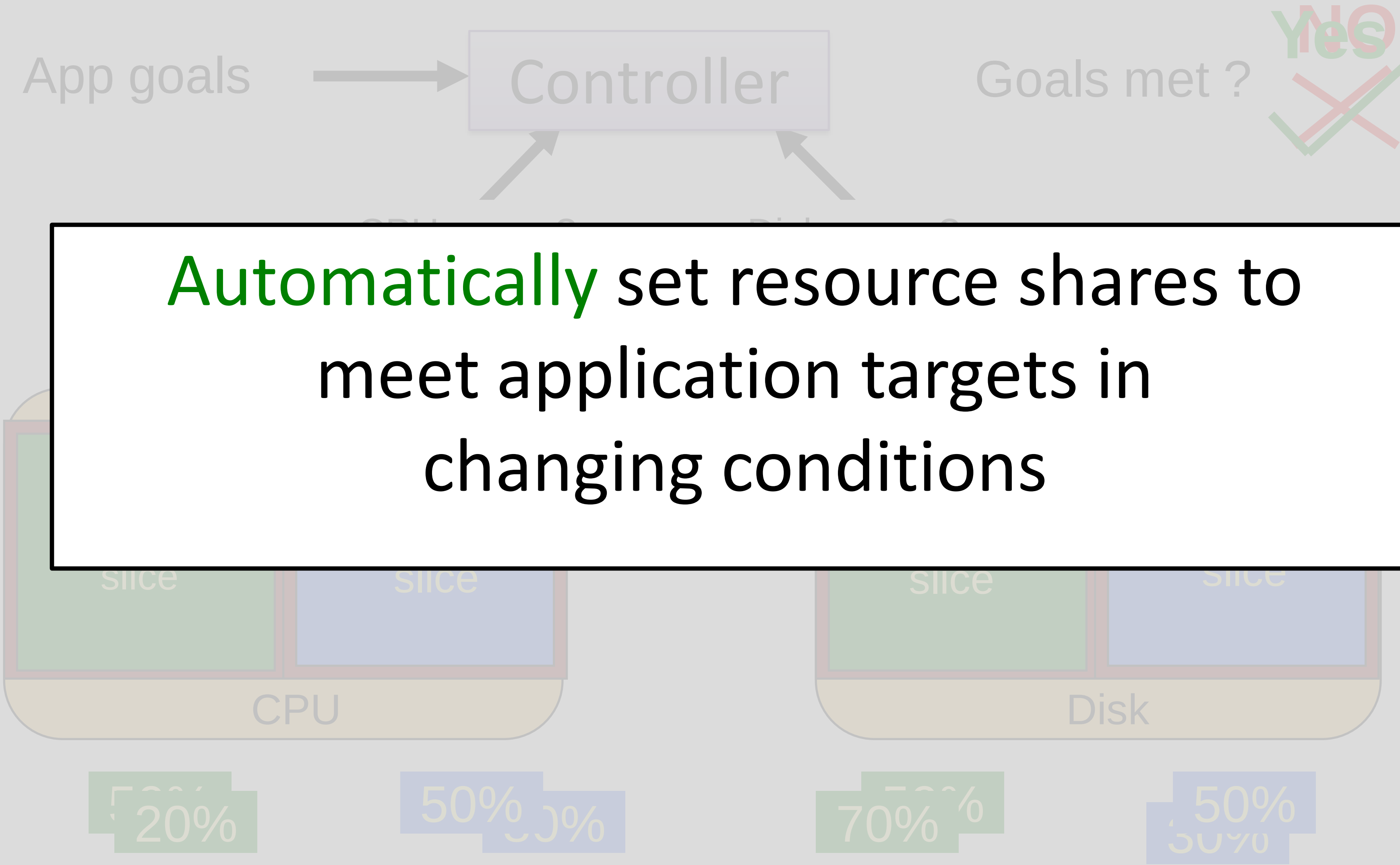
Approach III: Migration





- Good choice for long term overload
- Poor choice for bursty loads
- Adds overhead to an already overloaded node
- SLO violations while being migrated

Automated Control – an example



Automatically set resource shares to meet application targets in changing conditions

Previous work

- Distributed resource allocation
  - AronSIGMETRICS00, ChaseSOSP01, ShenOSDI02
  - Orthogonal to our approach
- Low-level schedulers
  - Credit, CFQ, SFQ, WaldspurgerOSDI02, GulatiTR07
  - Policy vs. Mechanism
- QoS mechanisms
  - ChandraIWQOS03, UrgaonkarICAC05
  - Developed for a single resource or application
- Control theory based
  - AbdelZaherTPDS02, HellersteinBook04,KarlssonIWQOS04
  - Applied in other scenarios

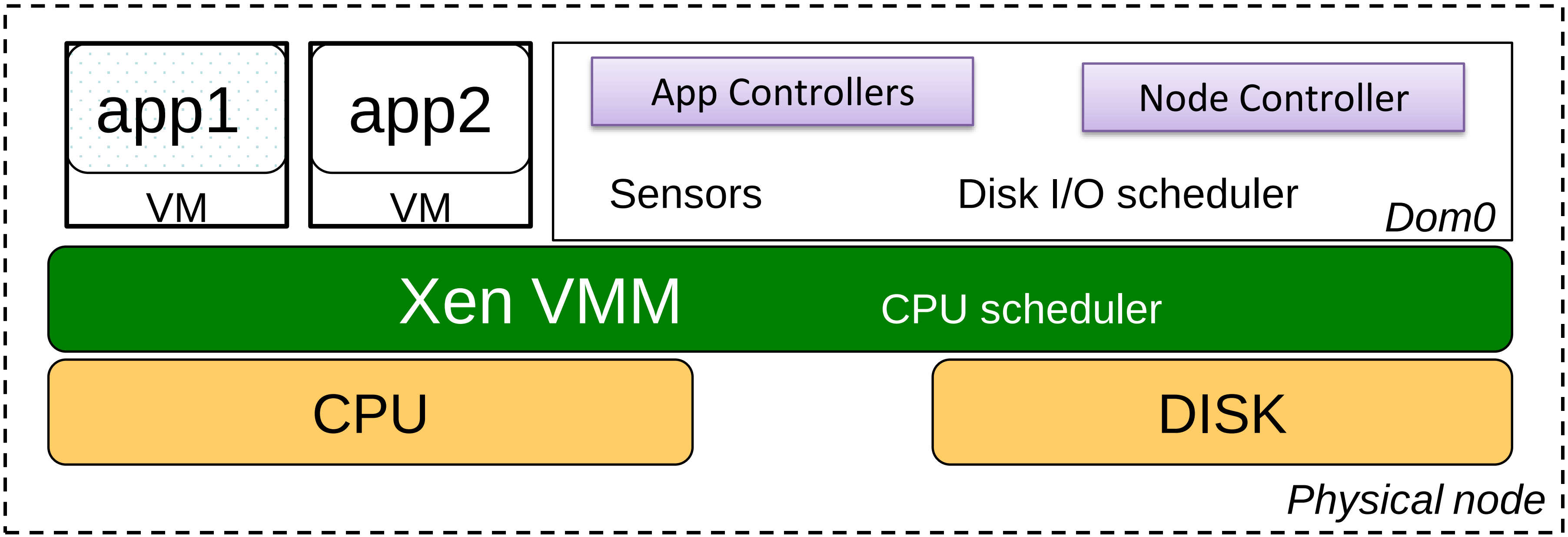
Outline

- *Motivation*
- *Background*
- *Idea*

- Modeling
- Controller Design
  - Application Controller
  - Node Controller

- Evaluation
  - Synthetic workloads
  - CPU and Disk bottlenecks

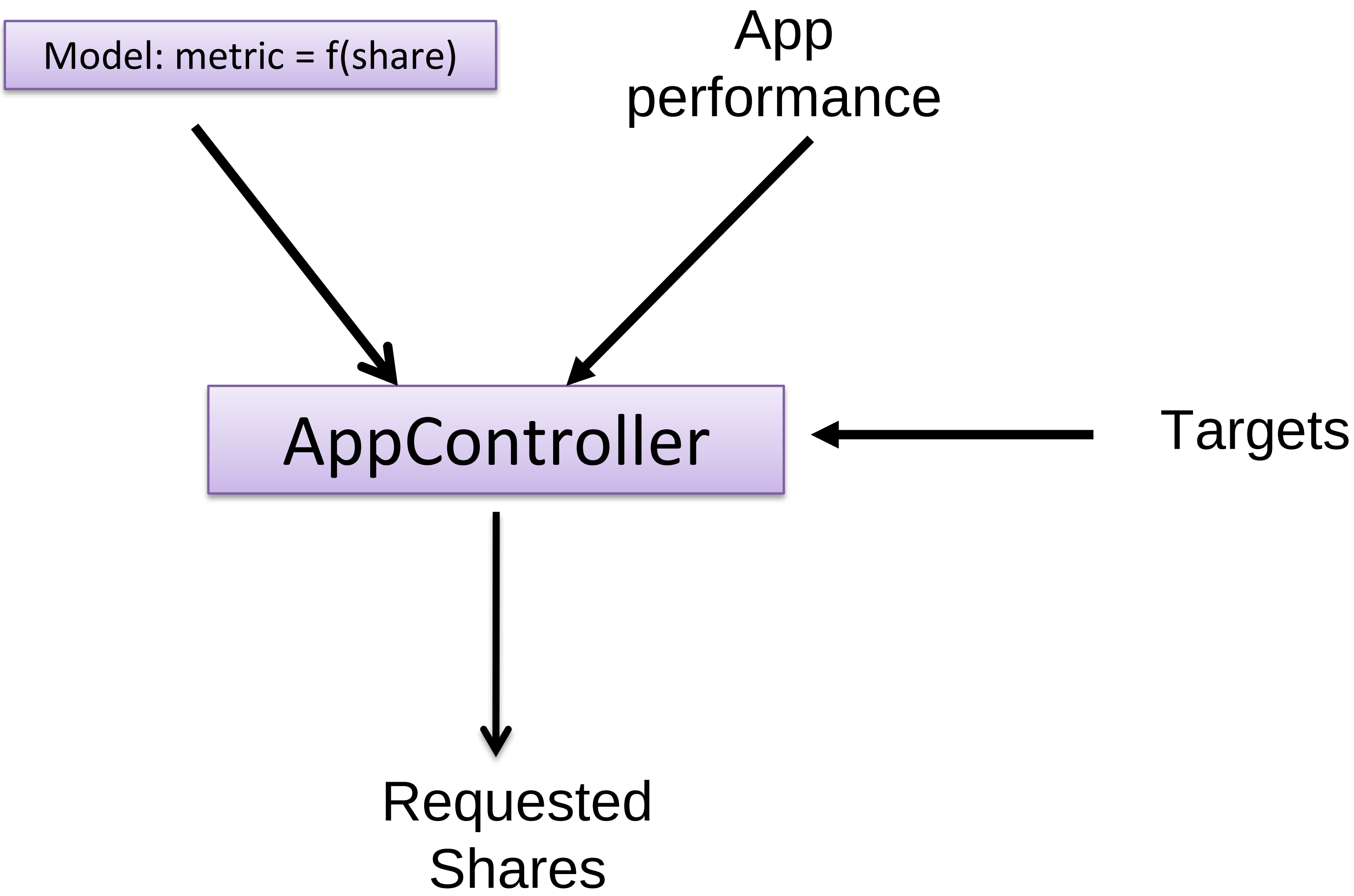
AutoControl system – 1,000ft view



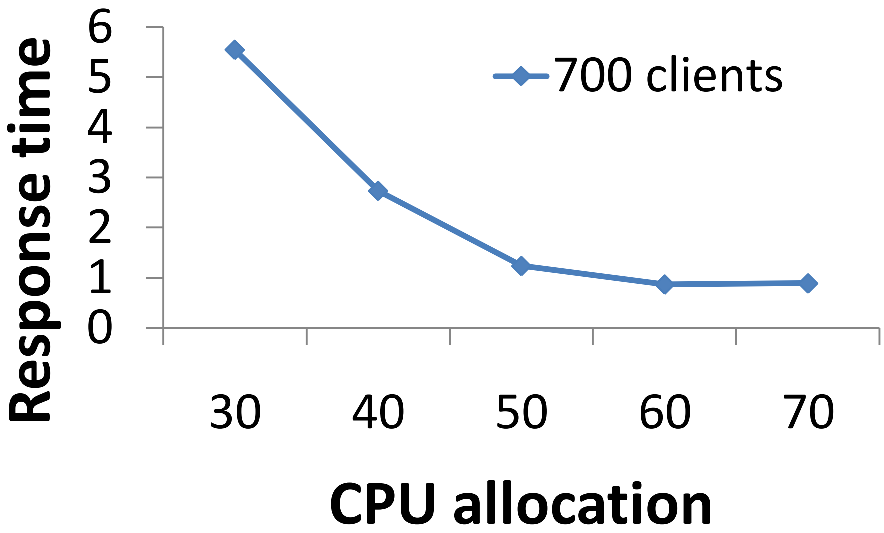
Every control interval

App Controller	Figures out the resource share required for a single app to meet its targets
Node Controller	Arbitrates among multiple applications All node controllers are independent
CPU and disk schedulers	Final shares are set

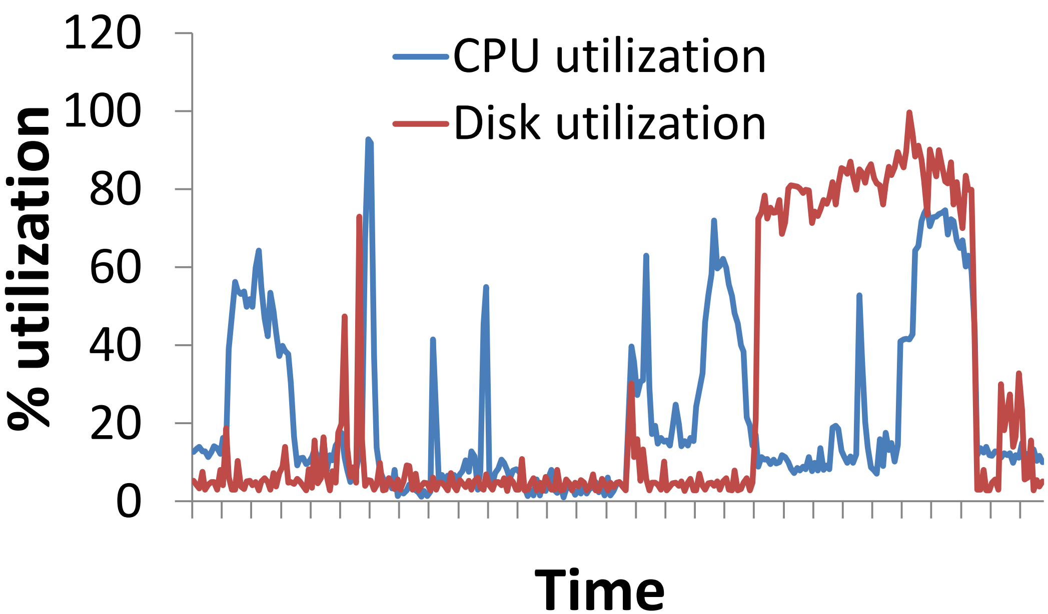
Application controller



Why is modeling hard?



Non linear relationships



Multiple resources

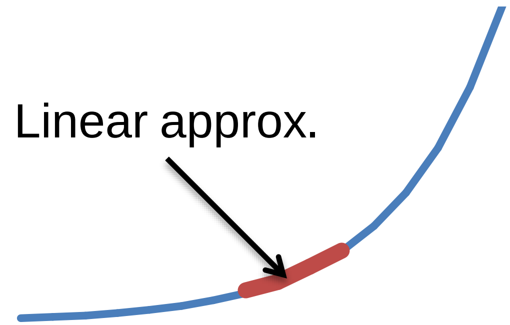
Bottleneck shifts

Multi-tier applications

Solution: Dynamic black box modeler

✓ Nonlinearity approximated using linear equations

✓ Multiple resources & multi-tier apps modeled with Multi Input Multi Output (MIMO) model



$$y_k = (a1)(y_{k-1}) + (b1\ b2)\begin{pmatrix} webcpu_{k-1} \\ webdsk_{k-1} \end{pmatrix} + (b3\ b4)\begin{pmatrix} dbcpu_{k-1} \\ dbdsk_{k-1} \end{pmatrix}$$

Current performance — Prev performance — Resource shares

First order

✓ Parameters (a1… b1 …) updated recursively (Recursive Least Squares RLS)

AppController internals
Control as optimization

Aggression factors

$$Cost = W\left\|y - y_{ref}\right\|^2 + Q\left\|u_k - u_{k-1}\right\|^2$$

Track target — Don't go wild

$$y = f(u_k)$$

$$y_{ref} = \text{target}$$

$$u_k = \begin{pmatrix} cpu \\ dsk \end{pmatrix}$$

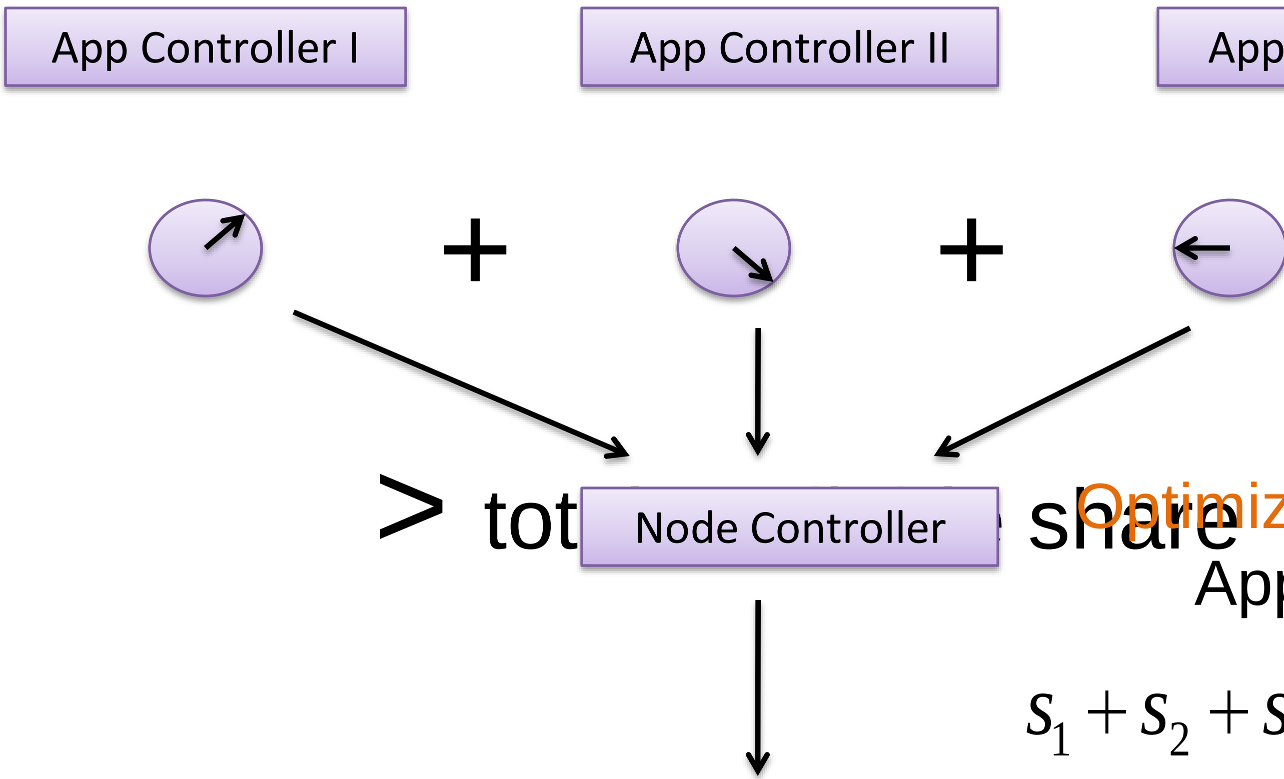
K[th] time interval

- ✓ Minimize cost
- ✓ Quadratic solvers to find u_k

Simplified Linear Quadratic Regulator formulation

Gory details: [CDC'07]

Control with consolidation



Resource shares

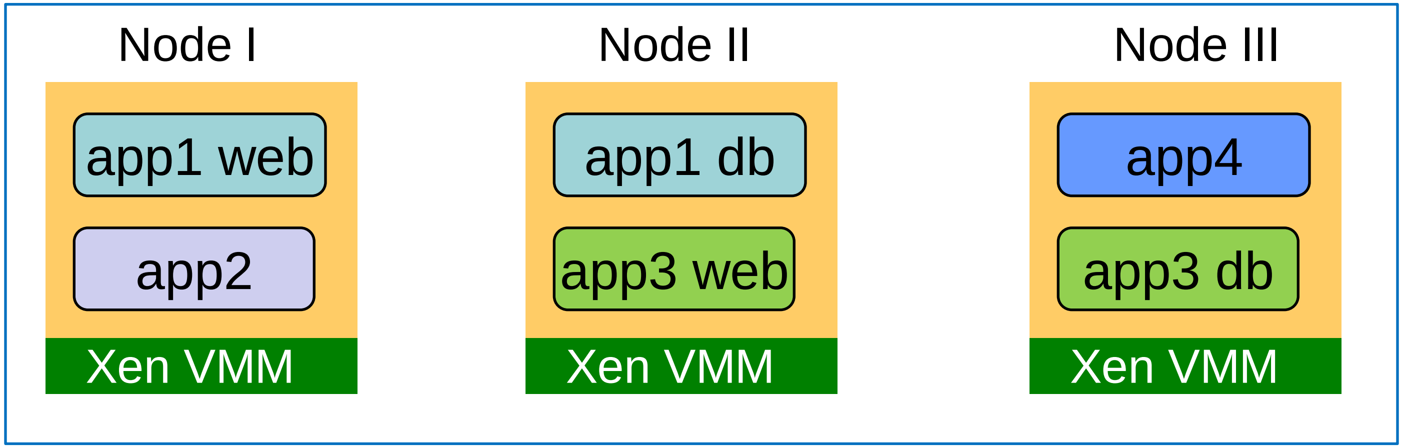
Optimization similar to AppController

$$S_1 + S_2 + S_3 + \ldots + S_n \leq S_{max}$$

Final share values

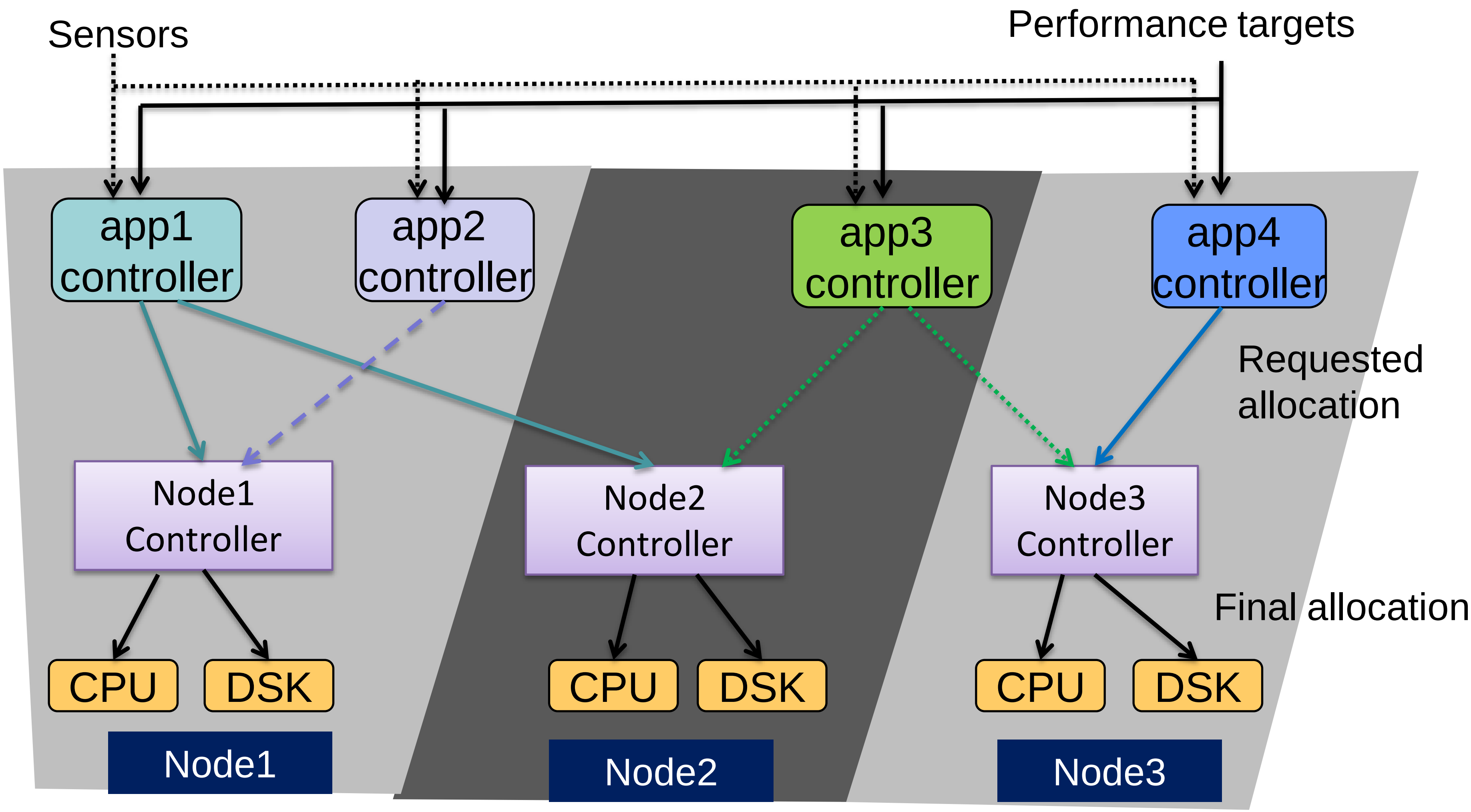
Control for data center scale

- Why not centralized controller? – variable explosion
- Why not combine app and node controllers?
  - Applications may span multiple nodes



- One AppController for application
- One NodeController per node

Distributed control



Experiments with 40 nodes on Emulab are successful

Outline

- *Motivation*
- *Background*
- *Idea*

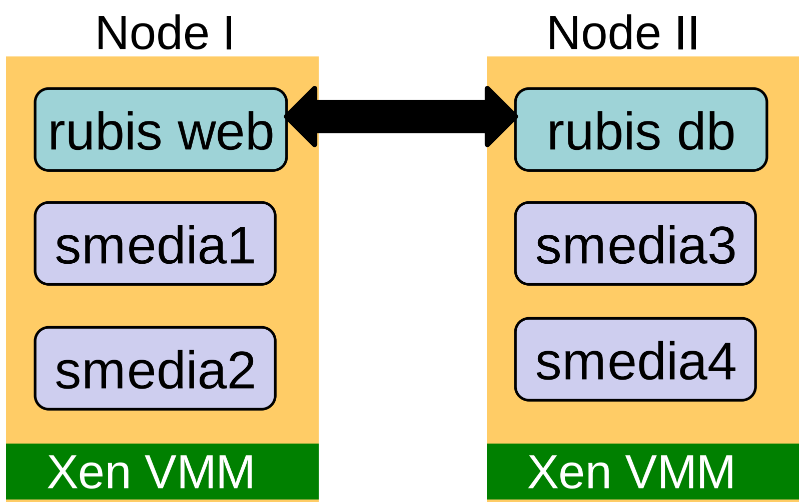
- Modeling
- Controller Design
  - Application Controller
  - Node Controller

- Evaluation
  - Synthetic workloads
  - CPU and Disk bottlenecks

Evaluation

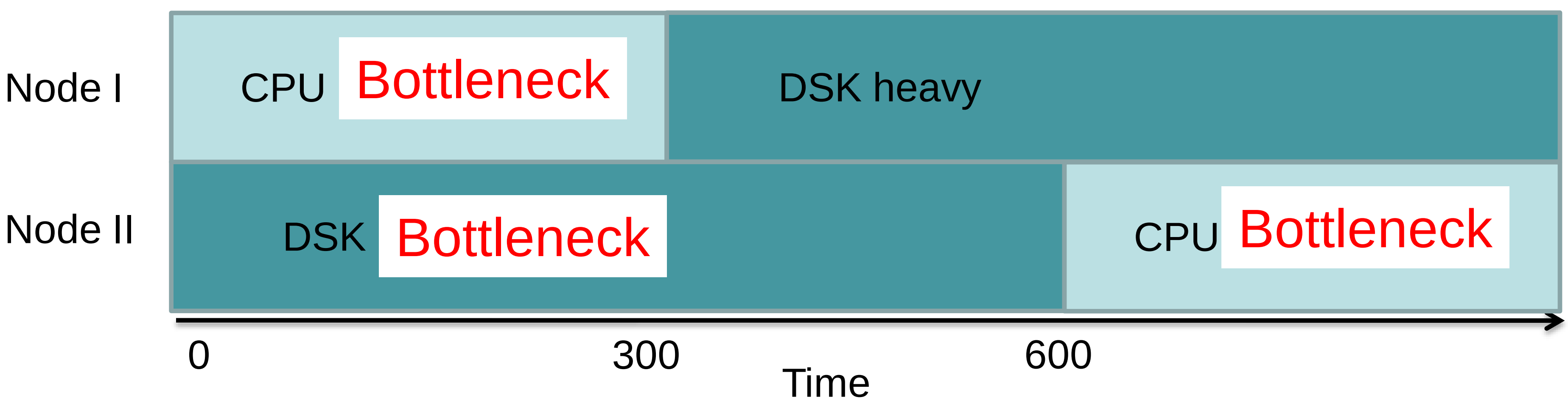
- Applications
  - RUBiS: eBay style auction benchmark
  - TPC-W: Transactional web e-Commerce benchmark
  - Smedia: Custom built secure media server
- Workloads
  - Synthetic
  - CPU and disk bottlenecks
- Evaluation questions
  - Can the controller meet targets?
  - Can it identify bottlenecks over time in different tiers and fix them?
  - Can it identify bottlenecks of different resources? (ex. CPU/Disk)

Experimental setup

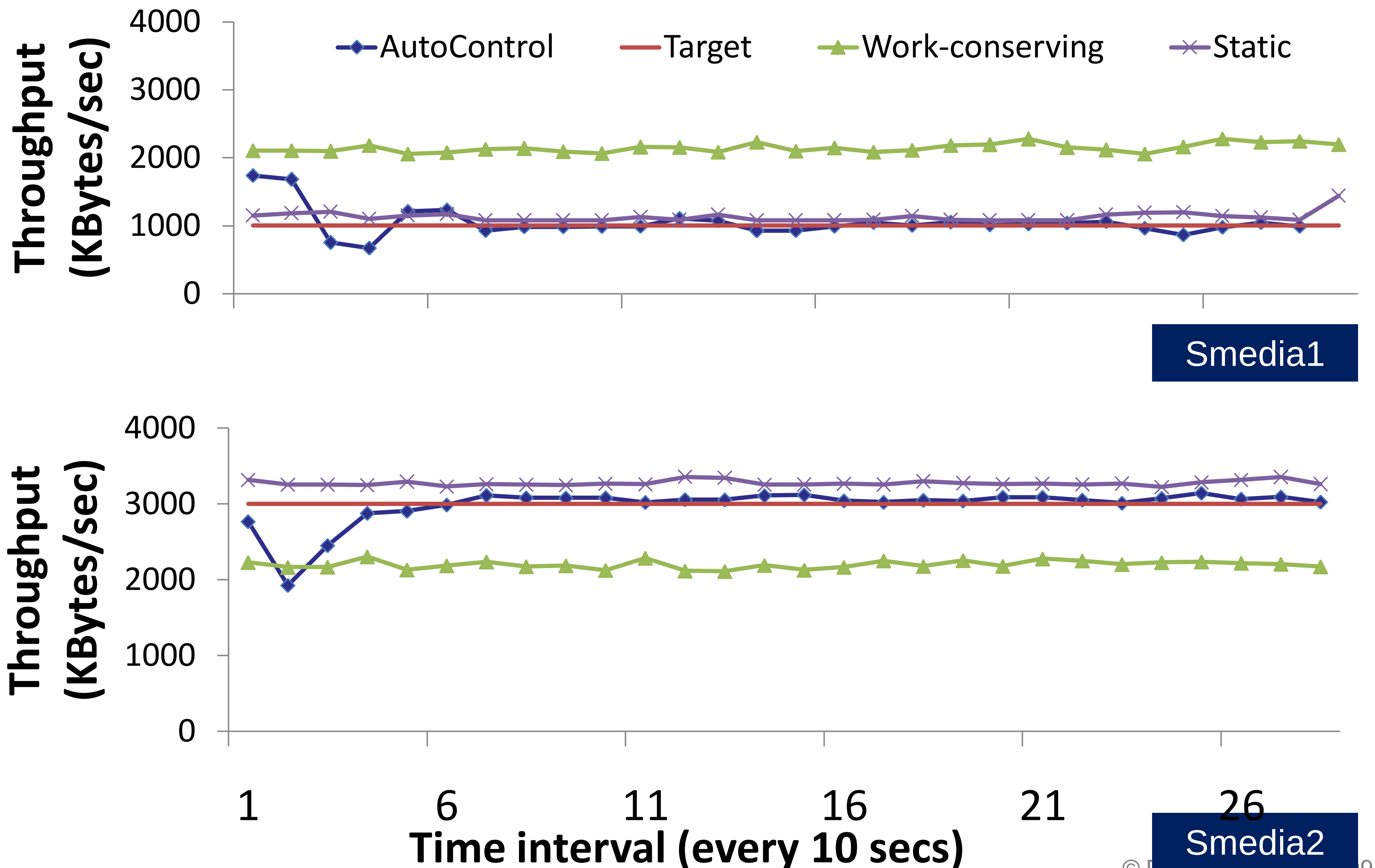


<u>Targets</u>

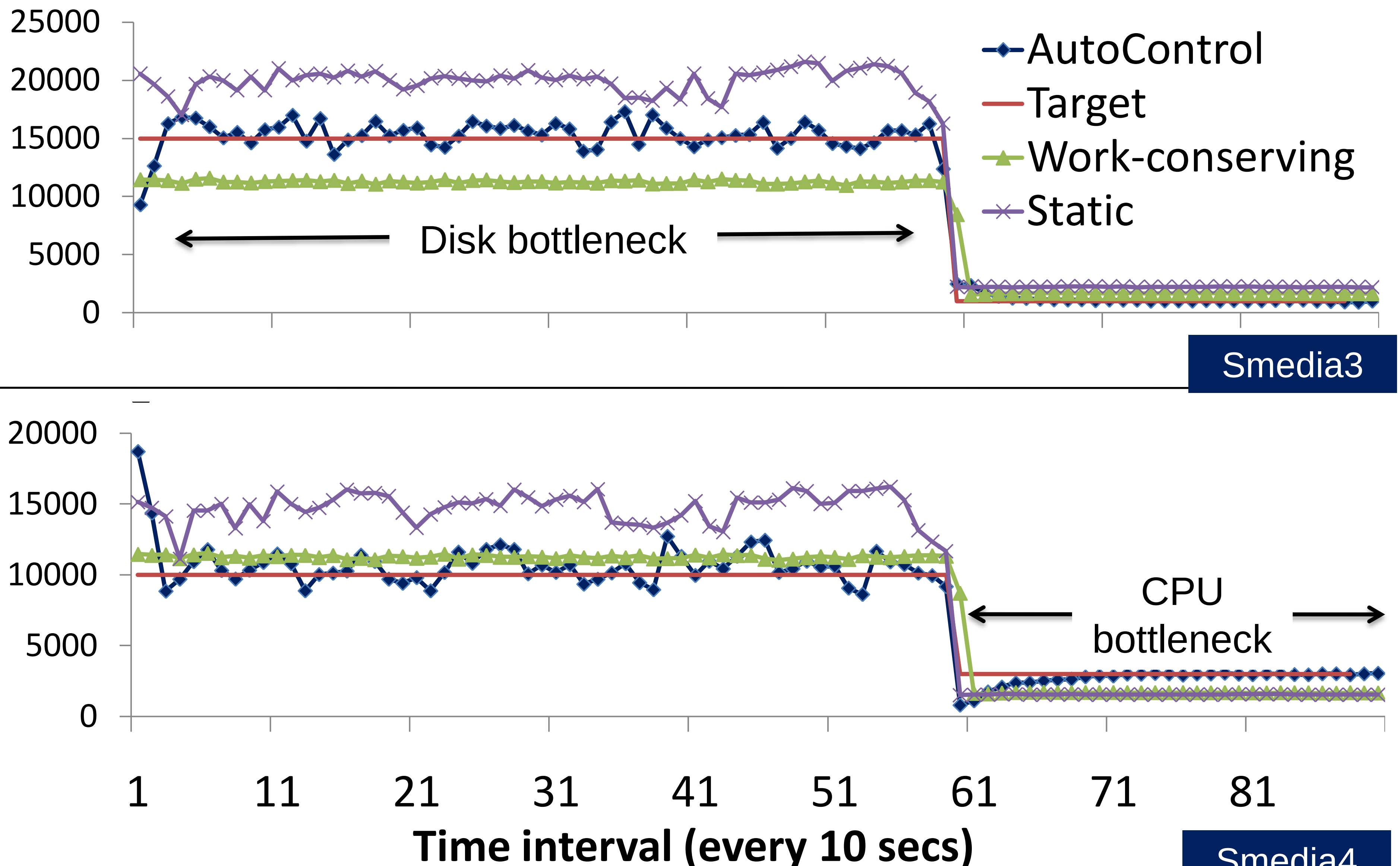
RUBiS – 100 req/sec
Smedia1 – 1000 Kbytes/sec
Smedia2 – 3000 Kbytes/sec
Smedia3 – 15000 Kbytes/sec
Smedia4 – 10000 Kbytes/sec



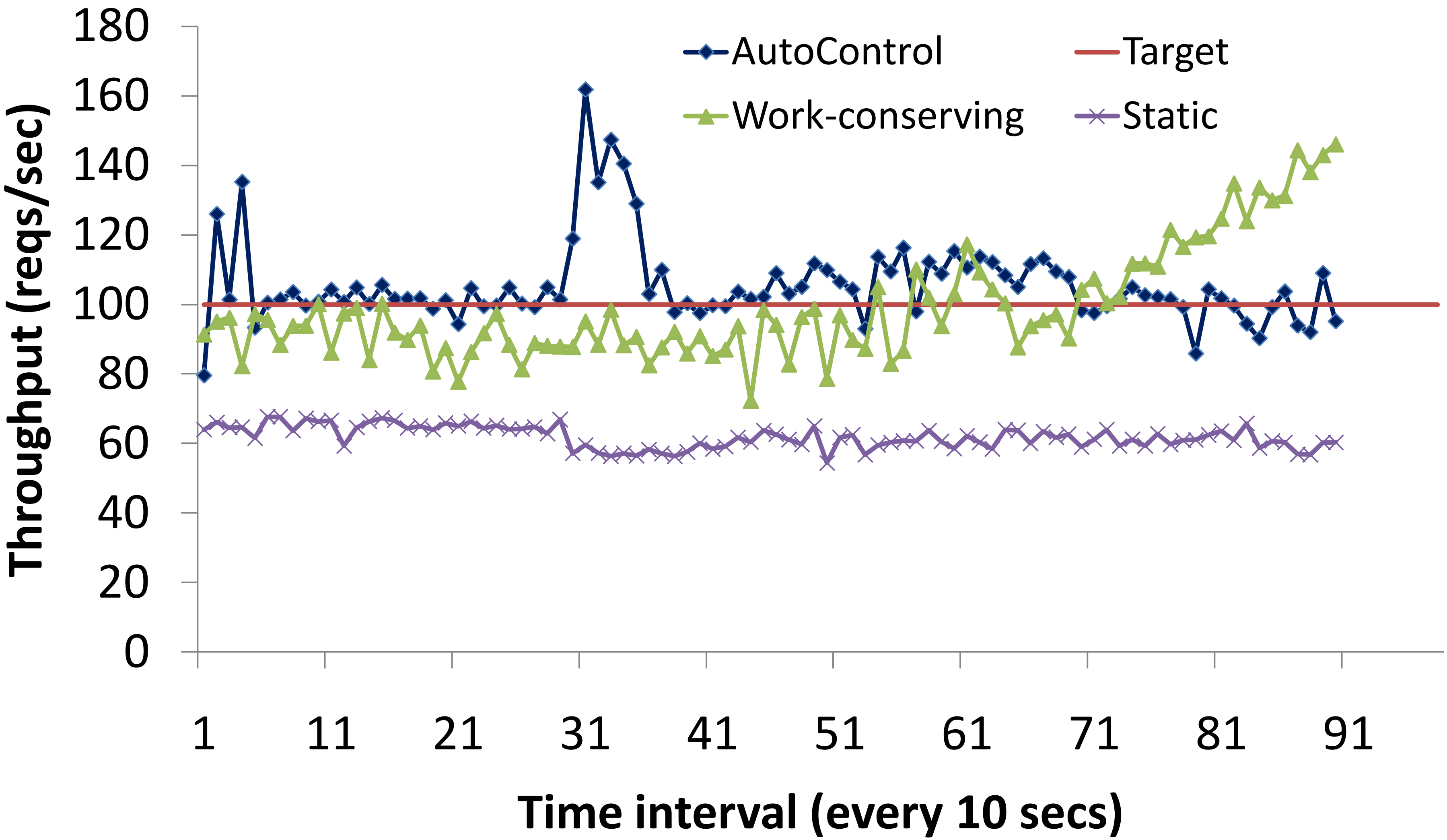
Node 1: CPU bottleneck



Node 2: DISK -> CPU bottleneck



Node 1 & 2: RUBiS performarnce



RUBiS

Experiment summary – average error

$$\text{Error} = \frac{\left\| y - y_{ref} \right\|}{yref} * 100$$

App	Work-conserving	Static	AutoControl
RUBiS	13.8%	38.3%	8.2%
Smedia1	100%	12.1%	4.3%
Smedia2	26.2%	9.6%	2.2%
Smedia3	44.3%	61.4%	10.1%
Smedia4	24.6%	47.5%	9.3%

AutoControl achieves <=10% error

Limitations and future work

- Modeling challenges
  - Non linear, fast changing workloads create problems
  - Combining white-box and black-box models
- Actuator and sensor behavior
  - Inaccuracies in measurements may lead to inaccurate models.
  - We are limited by what the scheduler can do
- Network and memory control
  - Earlier efforts with network control were unsuccessful
  - Preliminary memory control [IM'09 min-conference]
- Integrating with migration

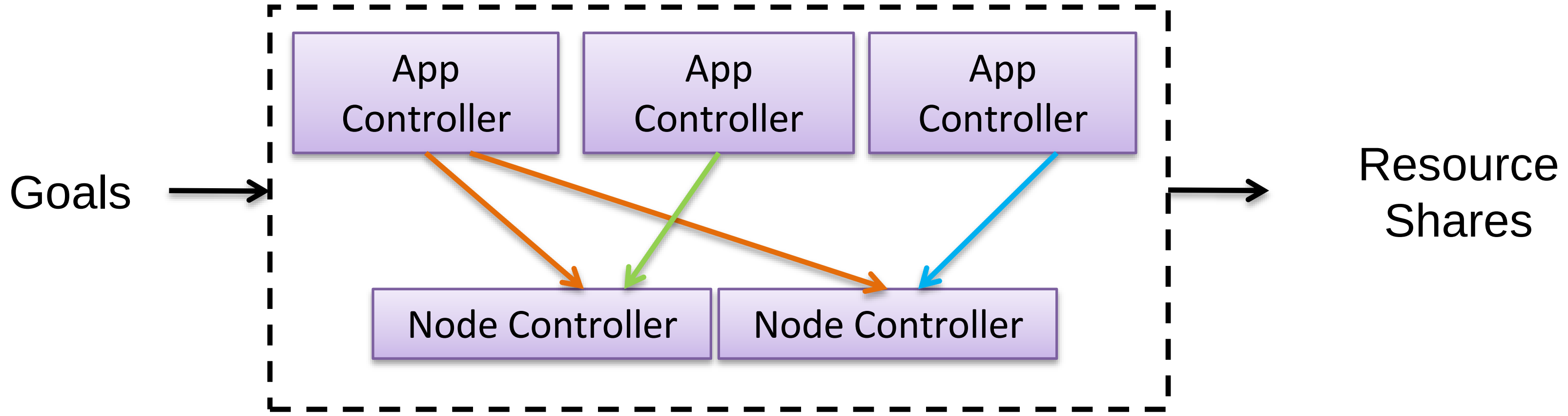
Summary

Automated Control of Multiple Virtualized Resources

✓Feedback control can be successfully applied to computer systems

- Dynamic black box modeler captures complex dynamics
- AppController can compute shares to meet targets for a single app
- NodeController arbitrates among competing apps

✓Distributed architecture that scales well



ppadala@umich.edu