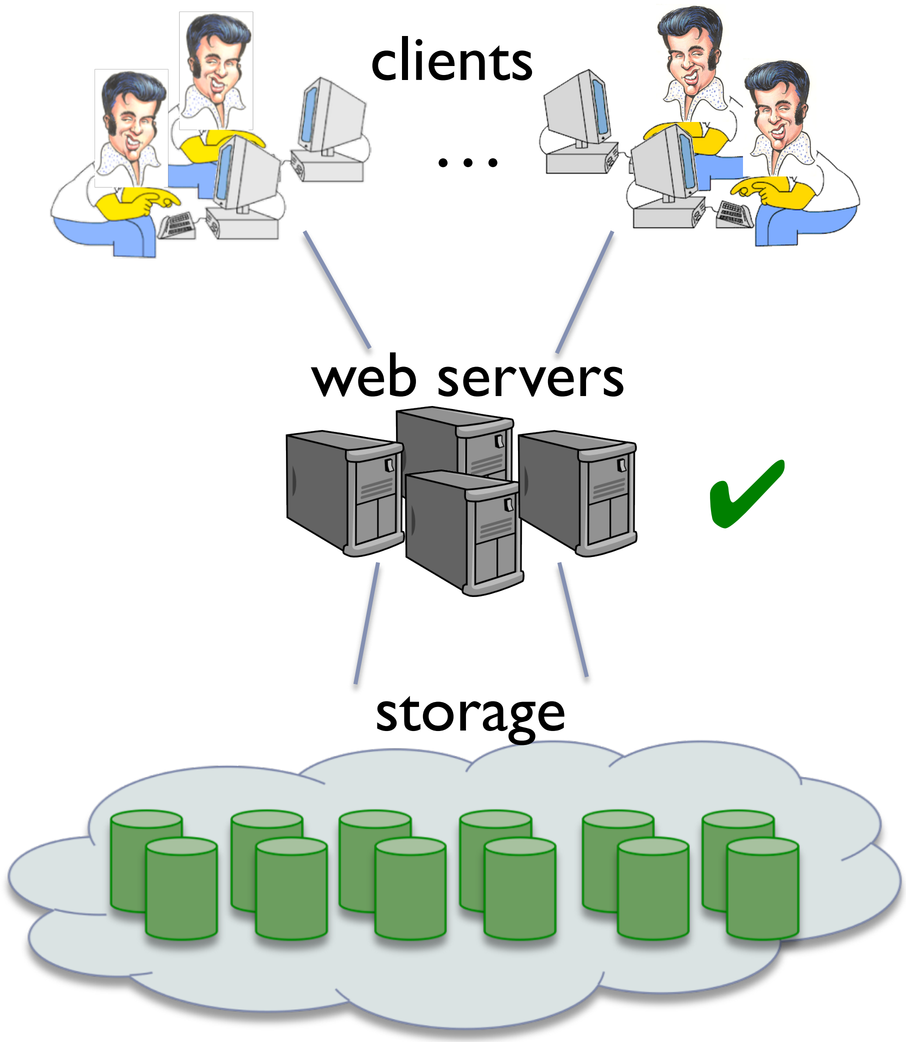
The SCADS Director:
Scaling a Distributed Storage System Under Stringent Performance Requirements

Beth Trushkowsky, Peter Bodík, Armando Fox,
Michael J. Franklin, Michael I. Jordan, David A. Patterson

FAST 2011

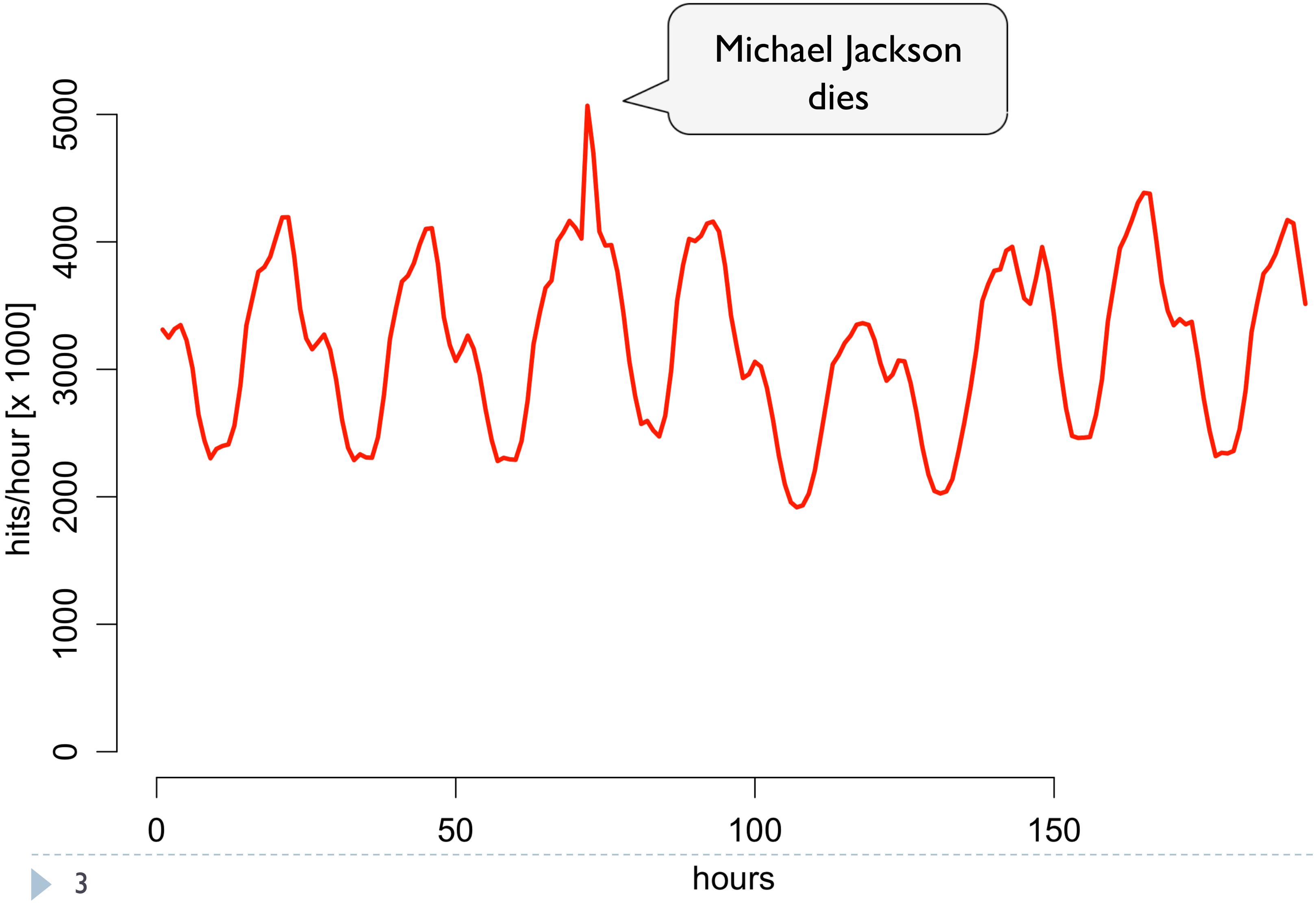
elasticity for interactive web apps



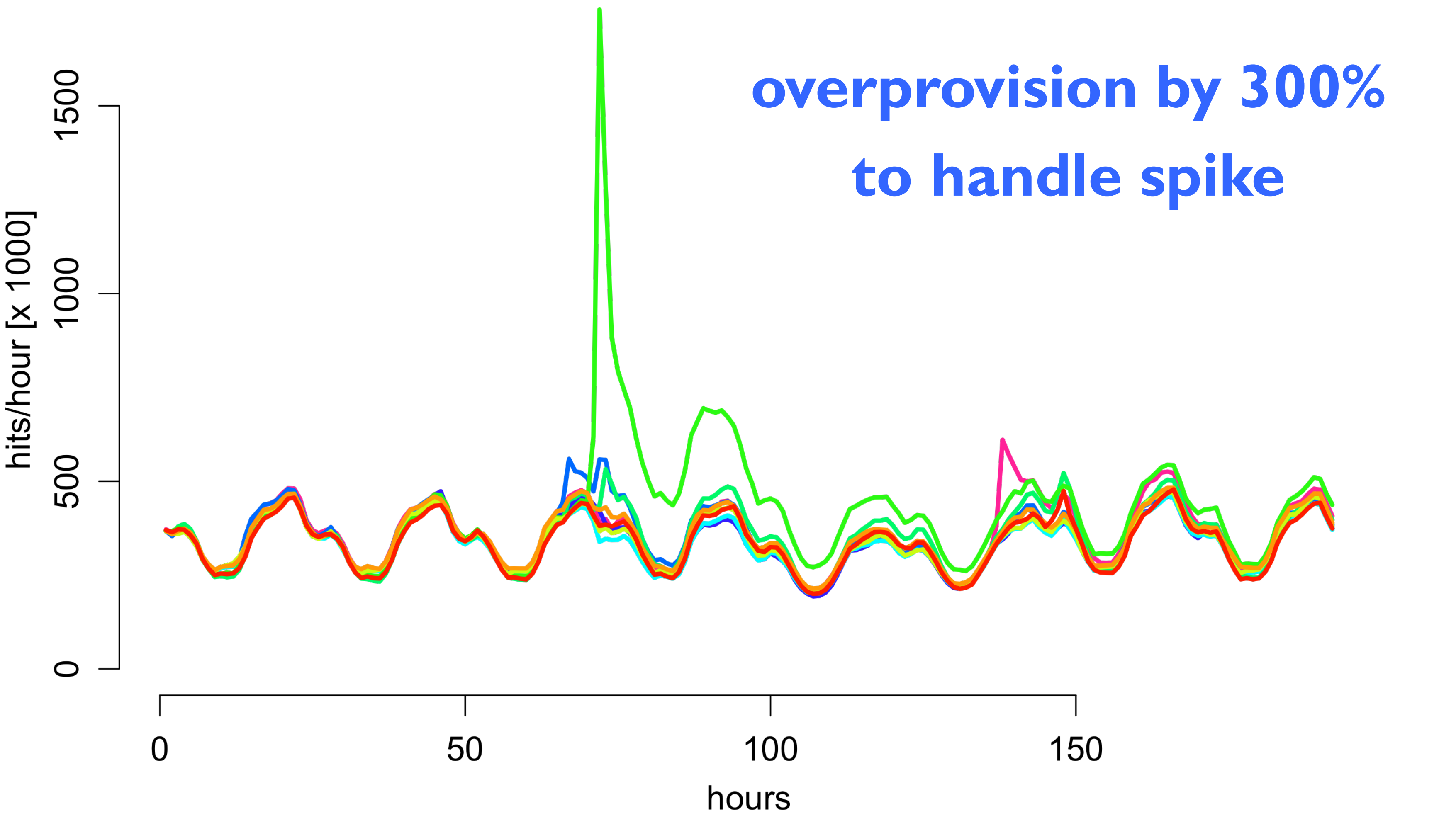
Interactivity Service-Level-Objective:
Over any 1-minute interval, 99% of requests are satisfied in less than 100ms

Targeted systems features:
- horizontally scalable
- API for data movement
- backend for interactive apps

wikipedia workload trace - June 2009



overprovisioning storage system



(assuming data stored on ten servers)

contributions

- Cloud computing is mechanism for storage elasticity
  - Scale up when needed
  - Scale down to save money
- We address the scaling policy
  - Challenges of latency-based scaling
  - Model-based approach for elasticity to deal with stringent SLO
  - Fine-grained workload monitoring aids in scaling up and down
  - Show elasticity for both a hotspot and a diurnal workload pattern

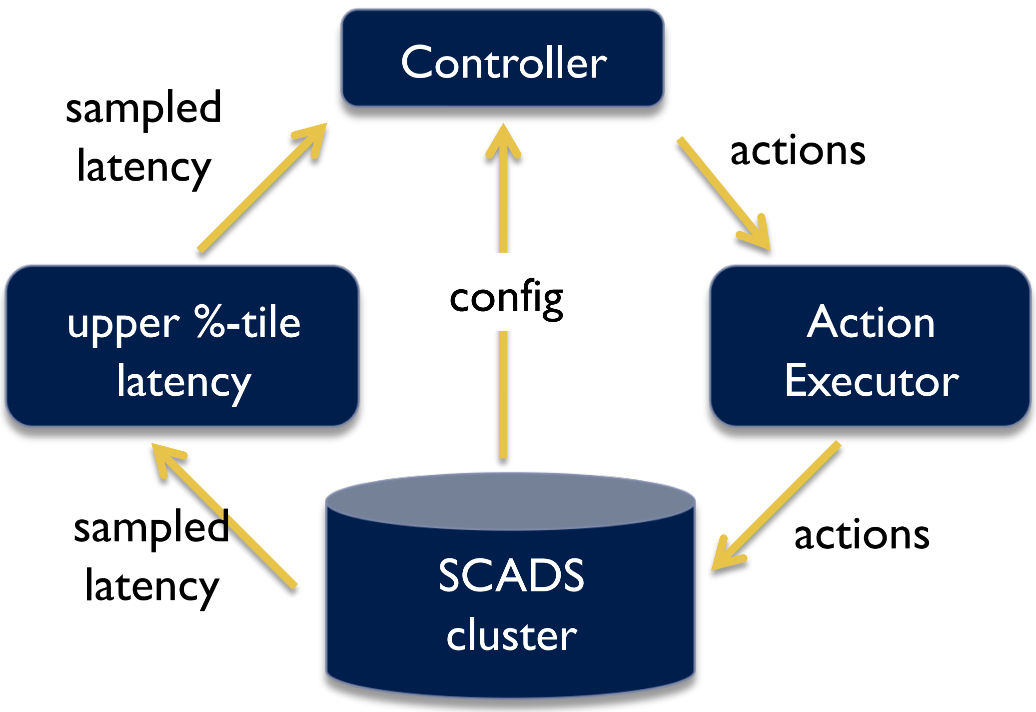
SCADS key/value store

- Features
  - Partitioning (until some minimum data size)
  - Replication
  - Add/remove servers

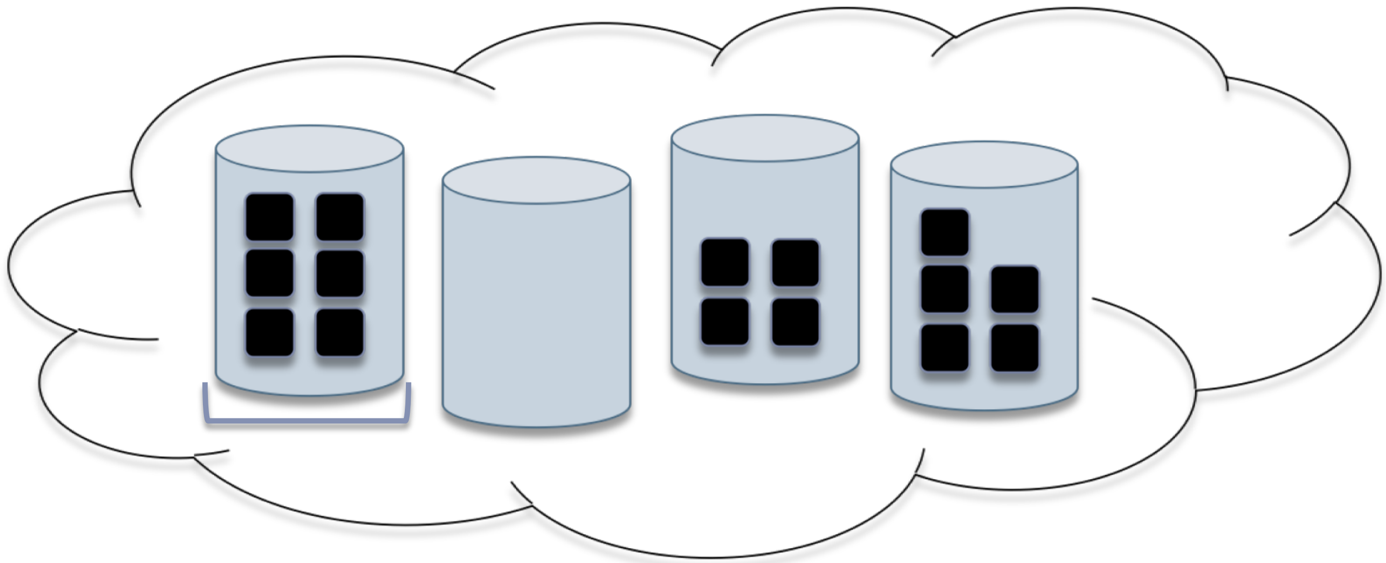
- Properties
  - Range-based partitioning
  - Data maintained in memory for performance
  - Eventually consistent

(see *SCADS: Scale-independent storage for social computing applications*, CIDR'09)

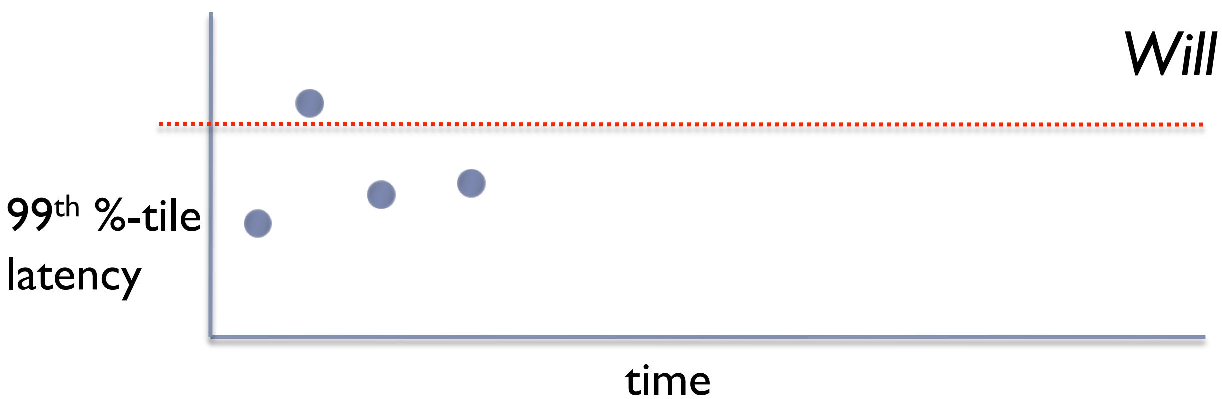
classical closed-loop control for elasticity?



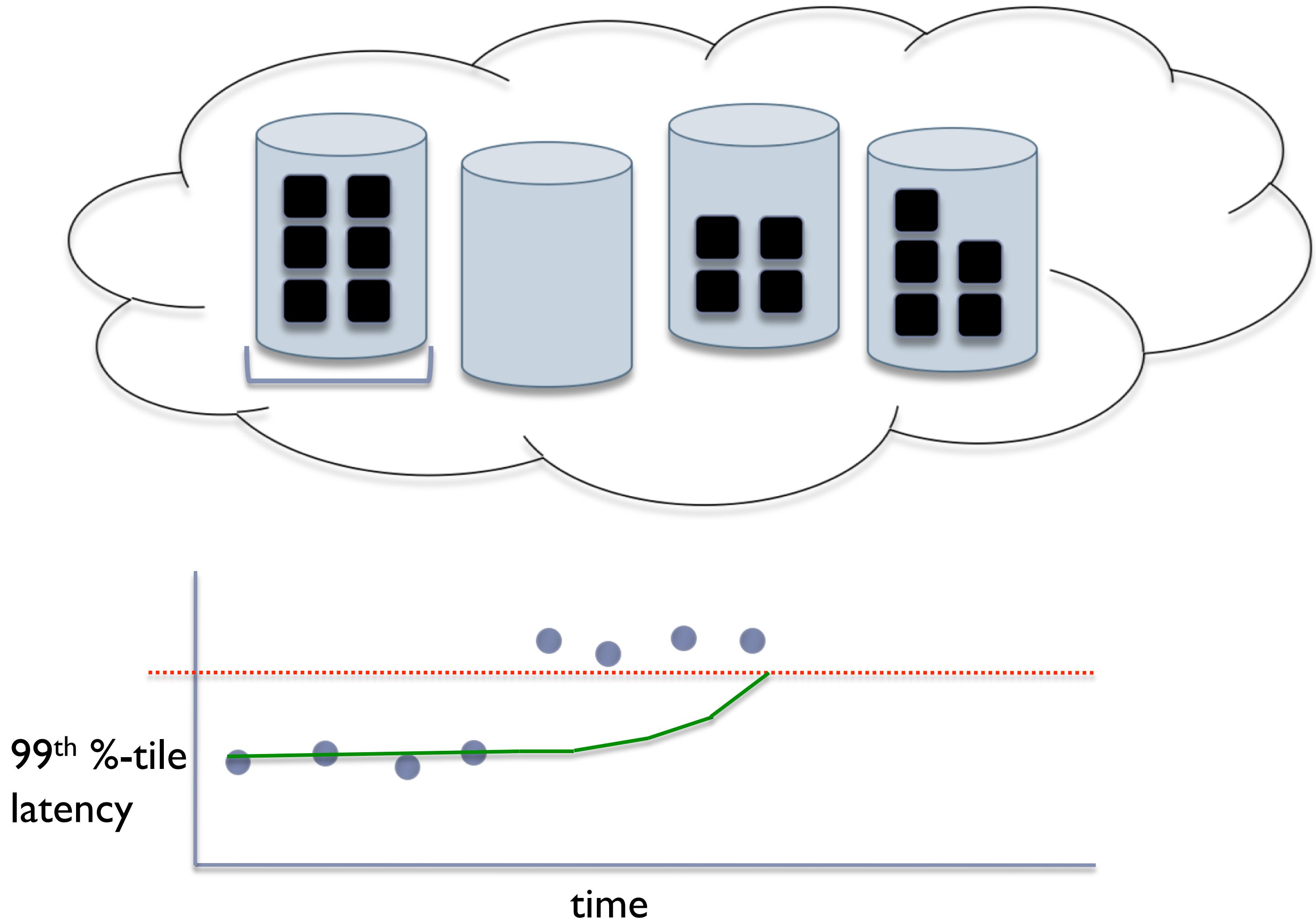
oscillations from a noisy signal



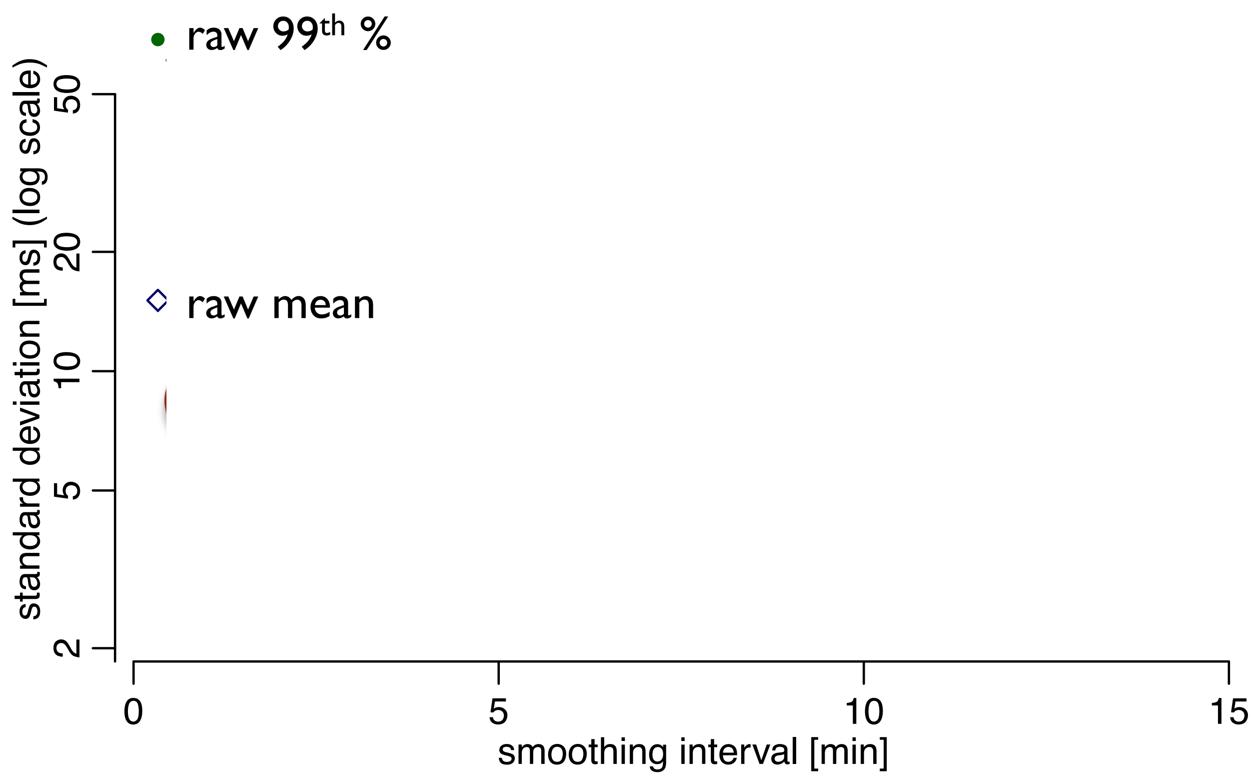
Noisy signal…
Will smoothing help?



too much smoothing masks spike



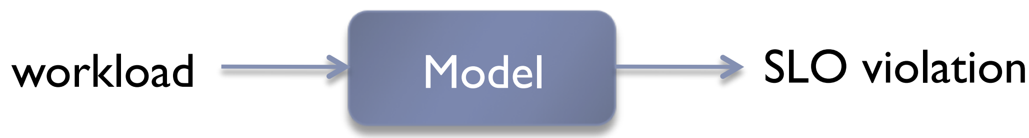
variation for smoothing intervals



(SCADS running on Amazon EC2)

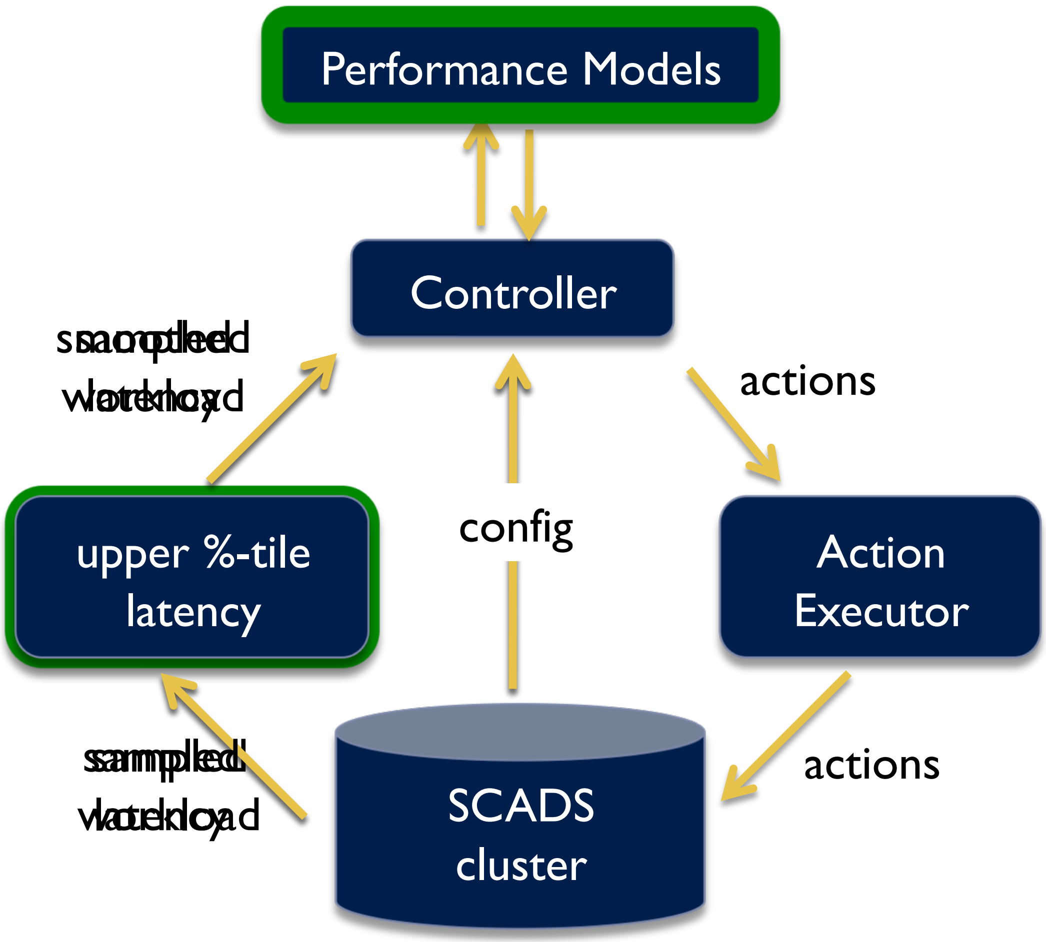
model-predictive control (MPC)

- MPC instead of classical closed-loop
  - Upper %-tile latency is a noisy signal
  - Use per-server workload as predictor of upper %-tile latency
  - Therefore need a model that predicts SLO violations based on observed workload



- Reacting with MPC
  - Use model of the system to determine a sequence of actions to change state to meet constraint
  - Execute first steps, then re-evaluate

model-predictive control loop

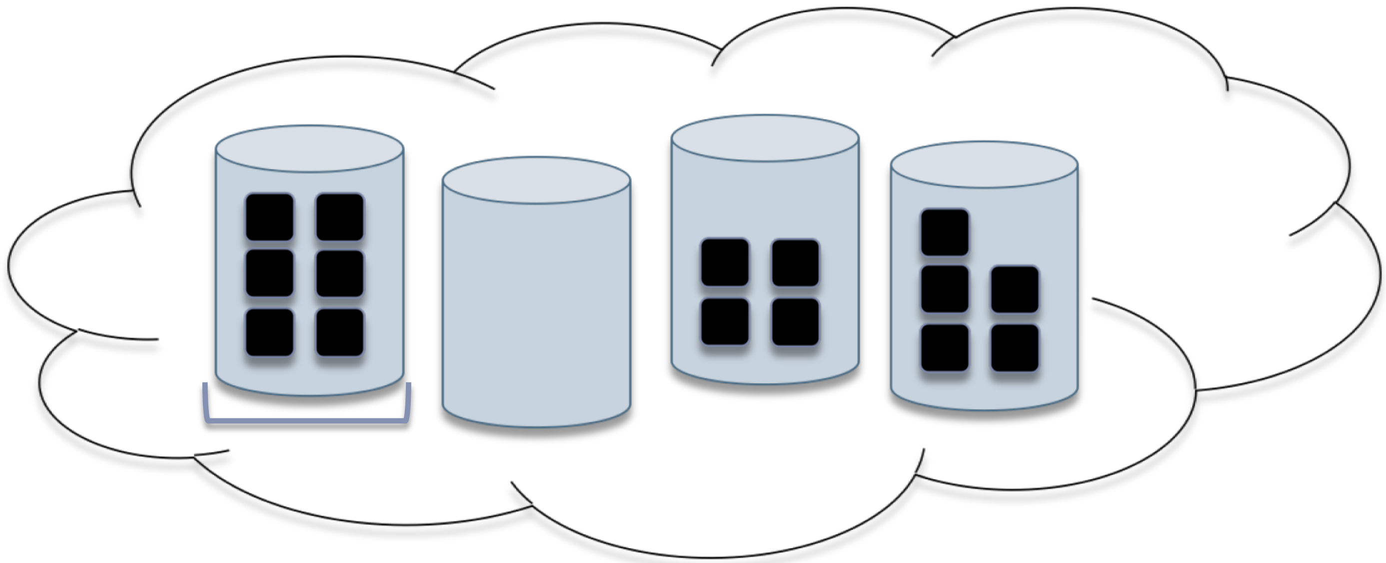


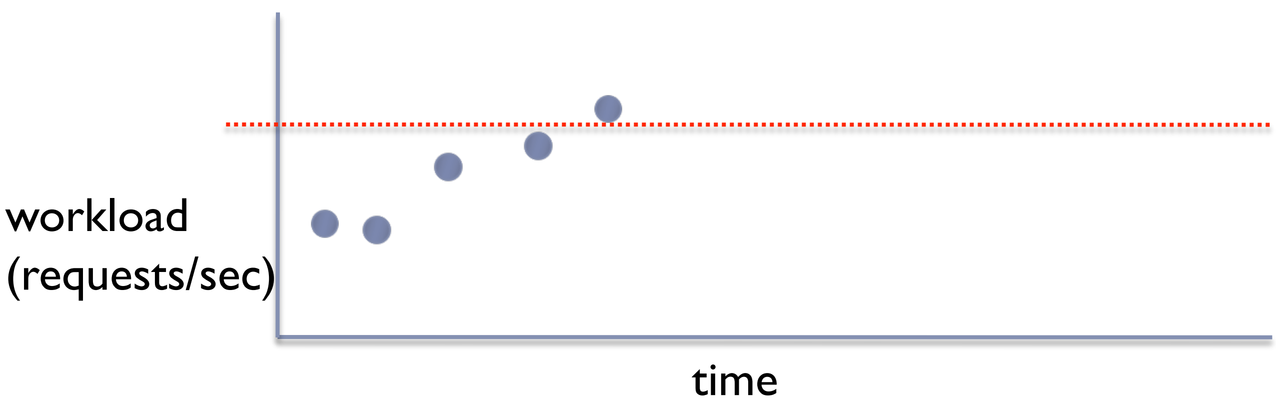
building a performance model

- Benchmark SCADS servers on Amazon's EC2

- Steady-state model
  - Single server capacity
  - Explore space of possible workload
  - Binary classifier: SLO violation or not

how much data to move?





finer-granularity workload monitoring

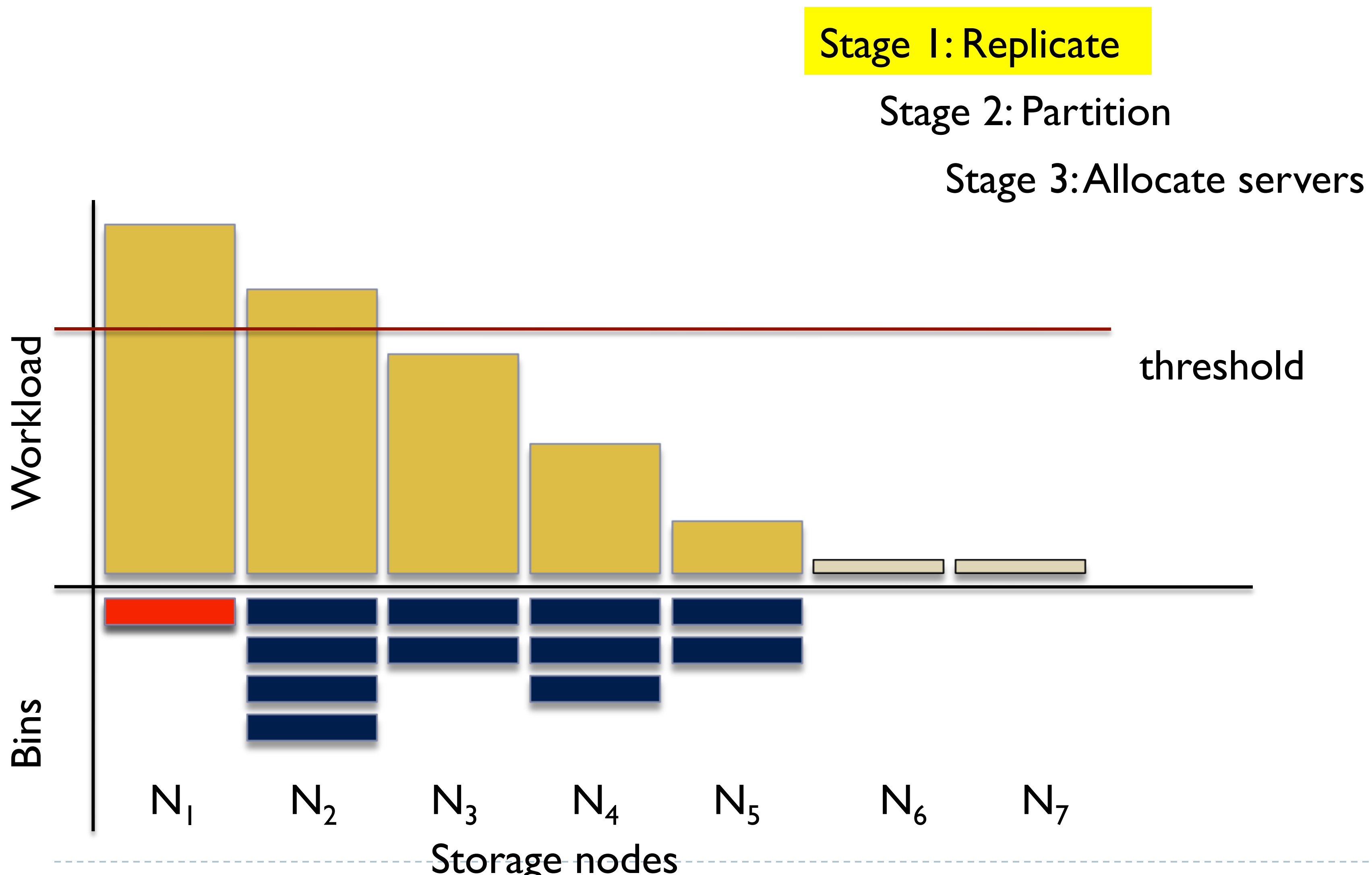
- Need fine-grained workload monitoring
  - Data movement especially impacts tail of latency distribution
  - Only move enough data to alleviate performance issues
  - Move data quickly
  - Better for scaling down later

- Monitor workload on small units of data (bins)
  - Move/copy bins between servers

summary of approach

- Fine-grained monitoring and performance model
  - Determine amount of data to move from overloaded server
  - Estimate how much "extra room" an underloaded server has
  - Know when safe to coalesce servers
- Replication for predictability and robustness
  - See paper and/or tonight's poster session

controller stages

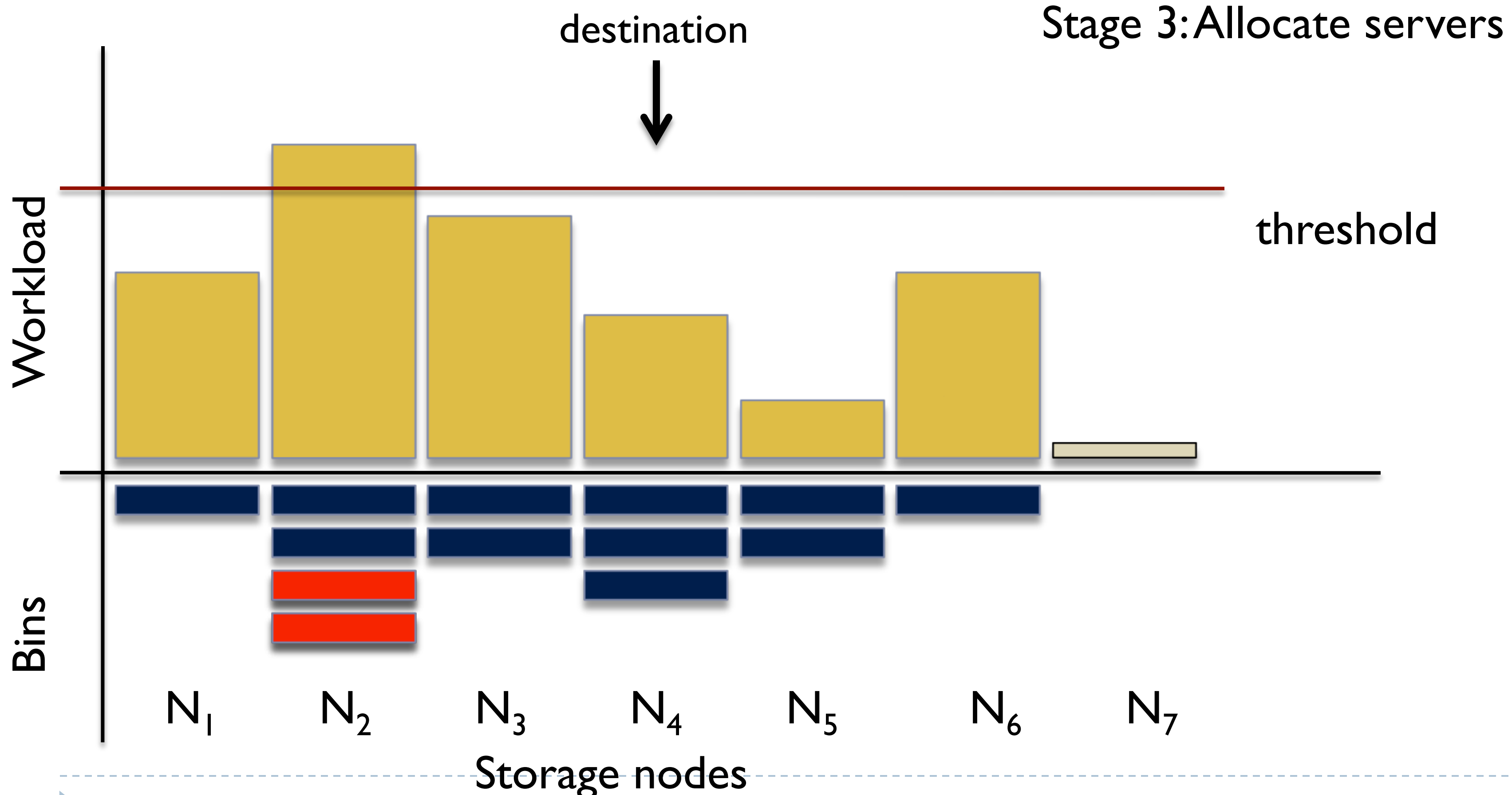


controller stages

Stage 1: Replicate

Stage 2: Partition

Stage 3: Allocate servers

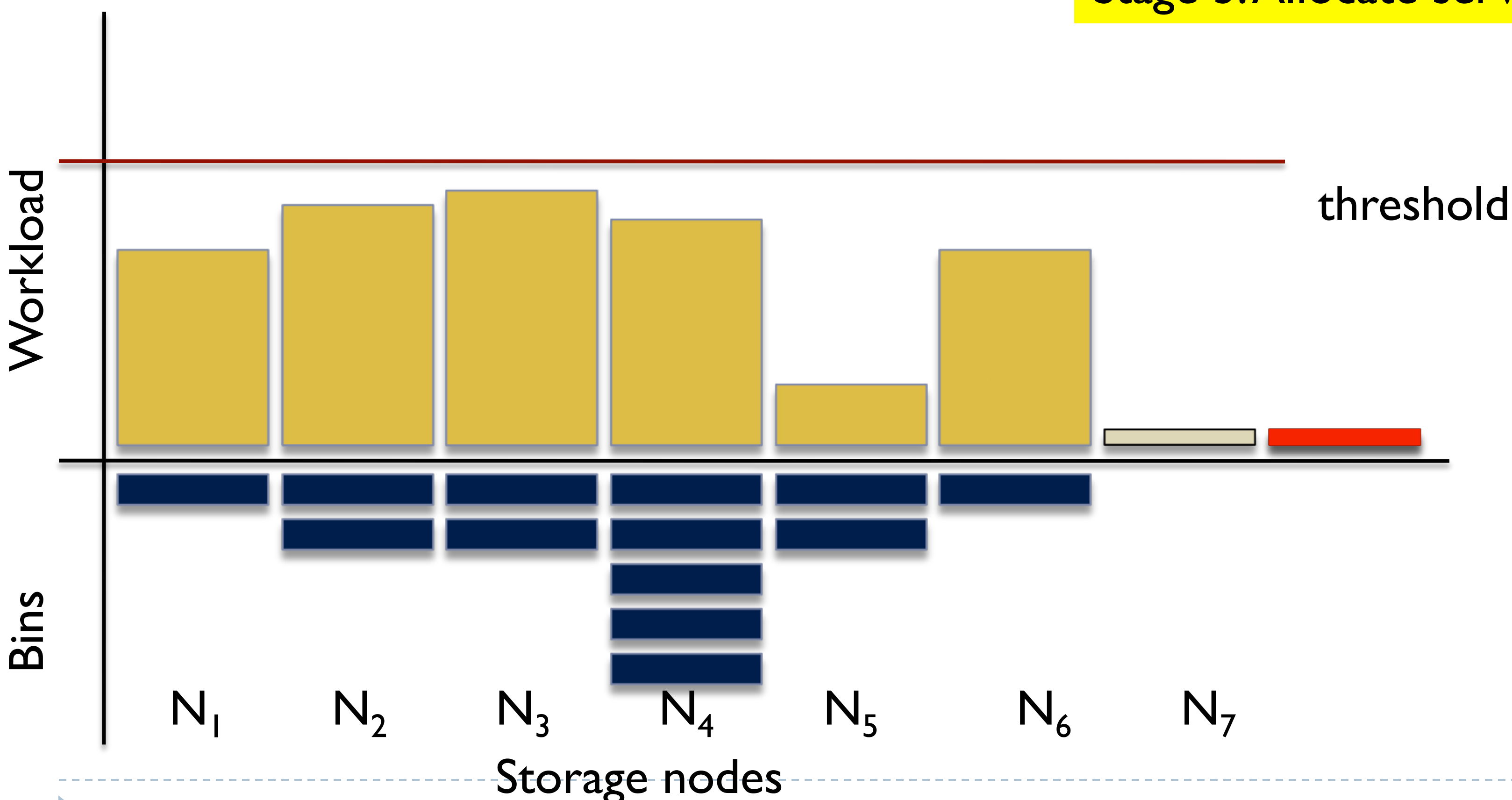


controller stages

Stage 1: Replicate

Stage 2: Partition

Stage 3: Allocate servers

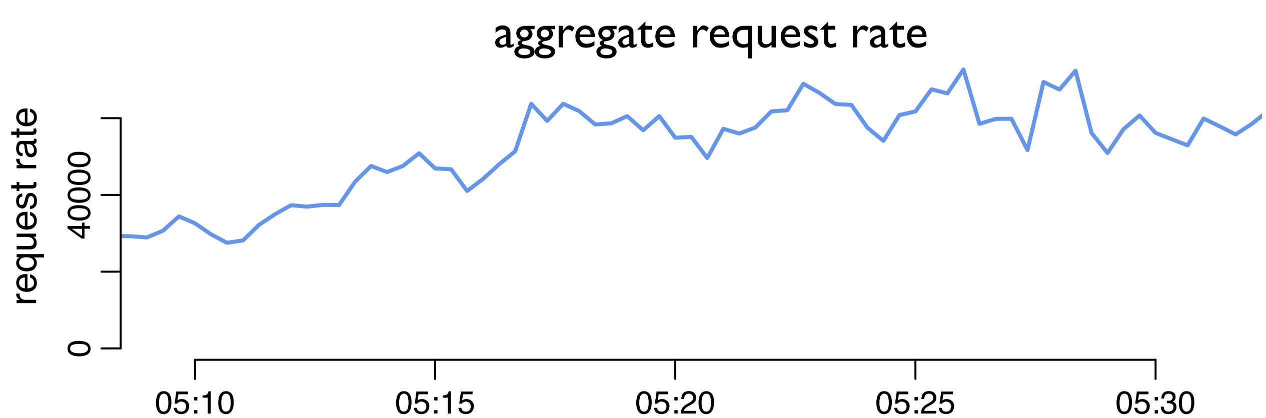


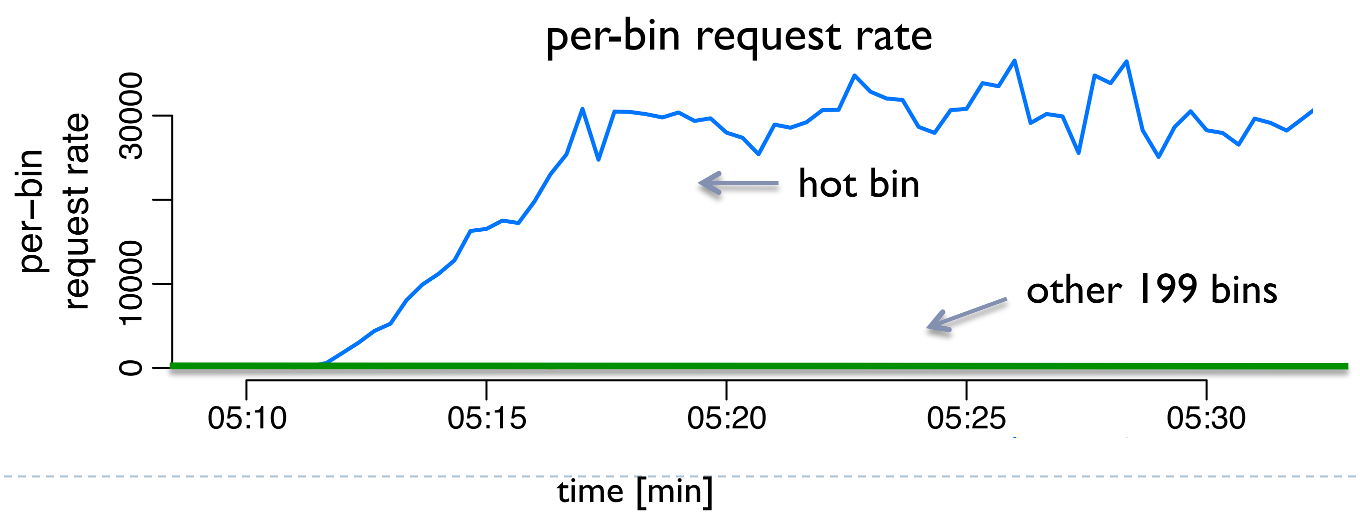
experimental results

- Experiment setup
  - Up to 20 SCADS servers run on m1.small instances on Amazon EC2
  - Server capacity: 800MB, due to in-memory restriction
  - 5-10 data bins per server
  - 100ms SLO on read latency

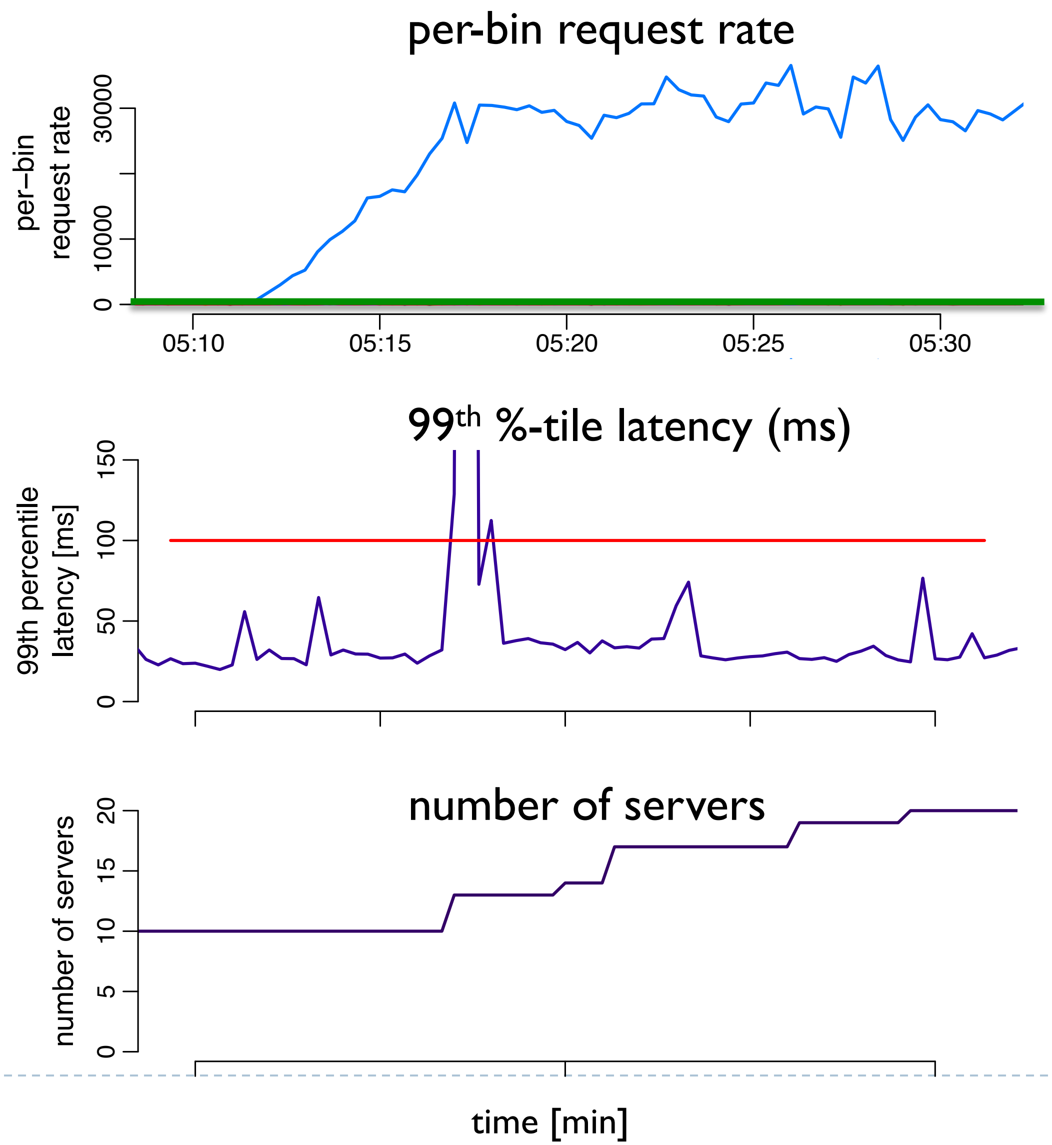
- Workload profiles
  - Hotspot
    - 100% workload increase in five minutes on a single data item
    - Based on spike experienced by CNN.com on 9/11
  - Diurnal
    - Workload increases during the day, decreases at night
    - Replayed trace at 12x speedup

extra workload directed to single data item



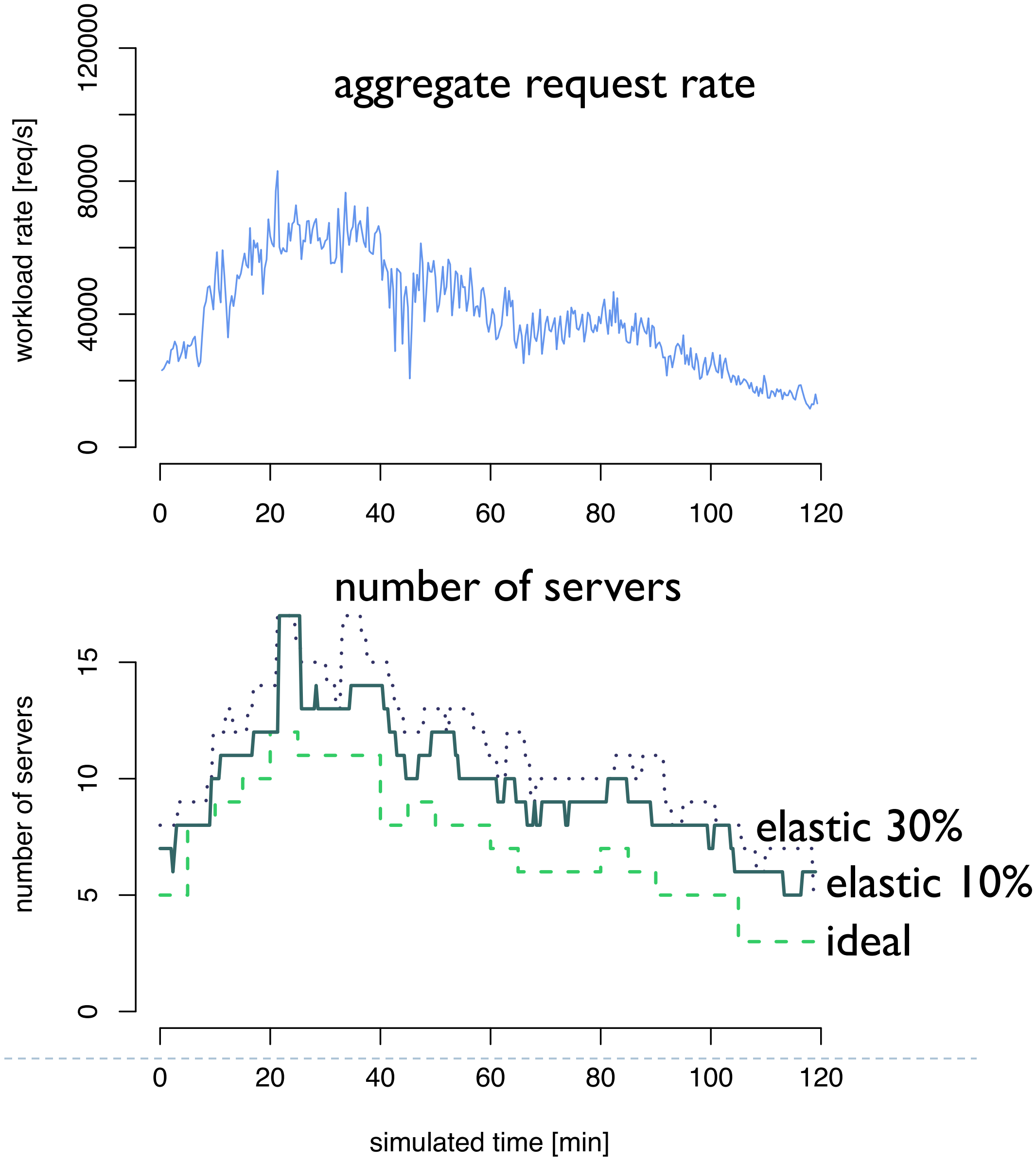


replicating hot data



scaling up and down

- Number of servers
  - two experiments close to "ideal"
- Over-provisioning tradeoff
  - Amplify workload by 10%, 30%
- Savings
  - Known peak: 16%
  - 30% headroom: 41%



cost-risk tradeoff

- Over-provisioning
  - Allows more time before violation occurs
  - Cost-risk tradeoff
- Comparing over-provisioning for diurnal experiment
  - Recall SLO parameters: *threshold*, *percentile*, *interval*
  - Over-provisioning factor of 30% vs 10%

Interval	**Max percentile achieved**	
	30%	**10%**
5 min	99.5	99
1 min	99	95
20 sec	95	90

conclusion

- Elasticity for storage servers possible by leveraging cloud computing
- Upper percentile too noisy
  - Model-based approach to build control framework for elasticity subject to stringent performance SLO
- Finer-grained workload monitoring
  - Minimize impact of data movement on performance
  - Quickly responding to workload fluctuations
- Evaluated on EC2 with hotspot and diurnal workloads

increasing replication