

# JustRunIt: Experiment-Based Management with Xen

Wei Zheng<sup>1</sup>

Ricardo Bianchini<sup>1</sup>

Yoshio Turner<sup>2</sup>

J. Renato Santos<sup>2</sup>

G. Janakiraman<sup>3</sup>

<sup>1</sup>Rutgers University

<sup>2</sup>HP Labs

<sup>3</sup>Skytap

# Data Center Management

- **Challenging**
  - **Variety of tasks**: Resource allocation, software/hardware upgrades, application and OS configuration...
  - **Complex**: Affect performance, availability, energy consumption
  - **Getting worse**: larger scale data centers hosting more diverse and dynamic workloads
- **Automation might save us**
  - **Using analytical models**: insight into system behavior, fast exploration of large parameter spaces
- **Modeling has some important drawbacks**
  - Expensive to develop, often relies on simplifying assumptions, may require re-calibration and re-validation as systems evolve

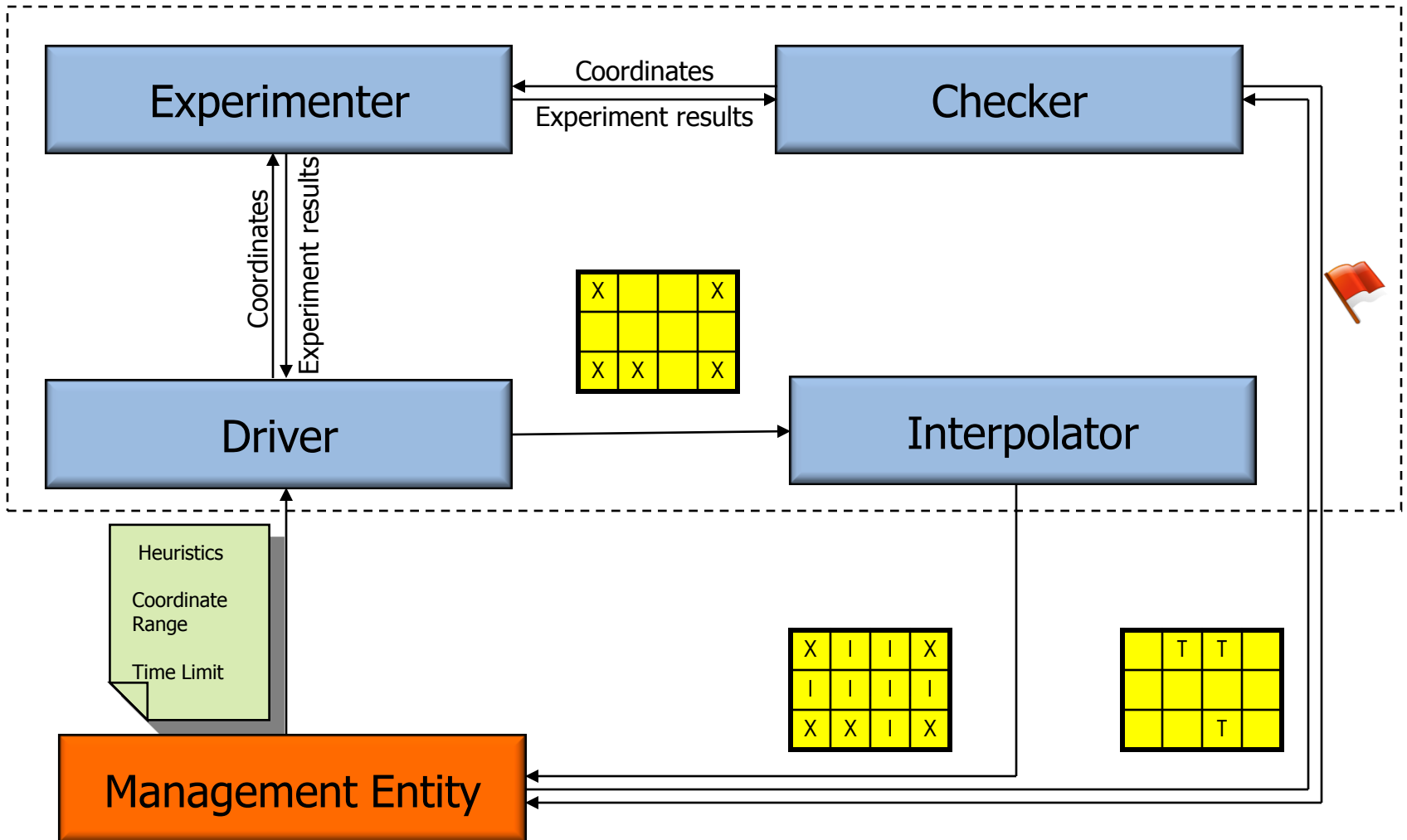
# Our Approach:

## Experiment-Based Management

- Experiments may be a better approach than modeling for many management tasks
  - Low cost: time and energy consumed by a few machines
  - No simplifying assumptions
  - No need for calibration/validation
- Use of VMs in production facilitates experimentation

**JustRunIt** – Infrastructure for experiment-based management of virtualized data centers hosting multiple services

# JustRunIt Architecture



# Experimenter



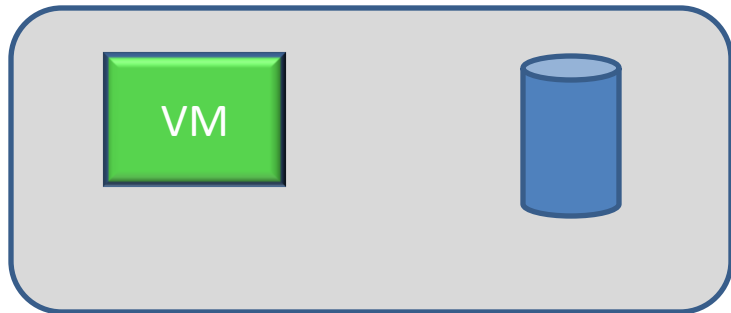
- 
- Step 1: Clone subset of production system to a sandbox

# Experimenter



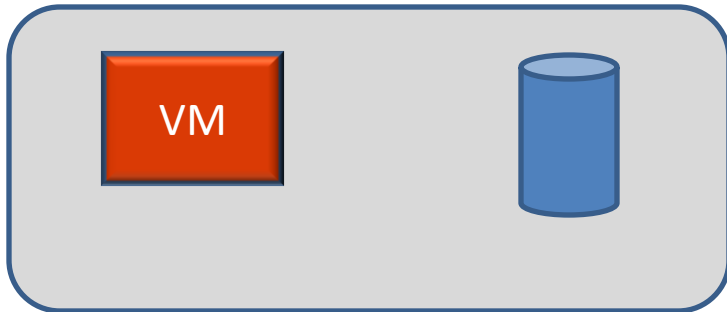
- Step 1: Clone subset of production system to a sandbox
  - VM cloning: Modify Xen live migration to resume original VM instead of destroying it
  - Storage cloning: LVM copy-on-write snapshot for sandbox VM

# Experimenter

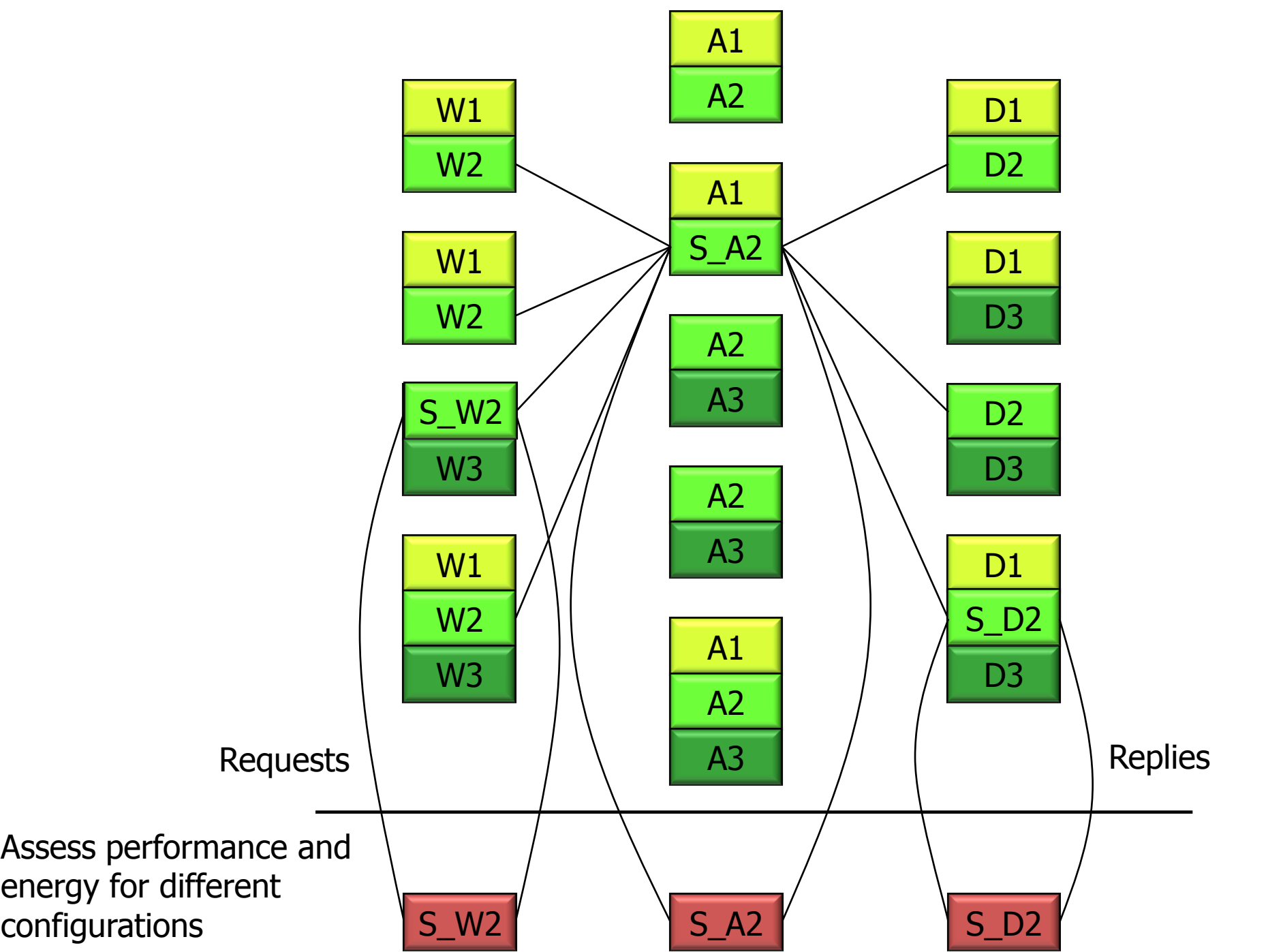


- Step 1: Clone subset of production system to a sandbox
  - VM cloning: Modify Xen live migration to resume original VM instead of destroying it
  - Storage cloning: LVM copy-on-write snapshot for sandbox VM
  - L2/L3 network address translation: implemented in driver domain netback driver to prevent network address conflict

# Experimenter



- Step 1: Clone subset of production system to a sandbox
  - VM cloning: Modify Xen live migration to resume original VM instead of destroying it
  - Storage cloning: LVM copy-on-write snapshot for sandbox VM
  - L2/L3 network address translation: implemented in driver domain netback driver to prevent network address conflict
  - Apply configuration changes : e.g., resource allocation
  - Resume execution



Requests

Replies

Assess performance and energy for different configurations

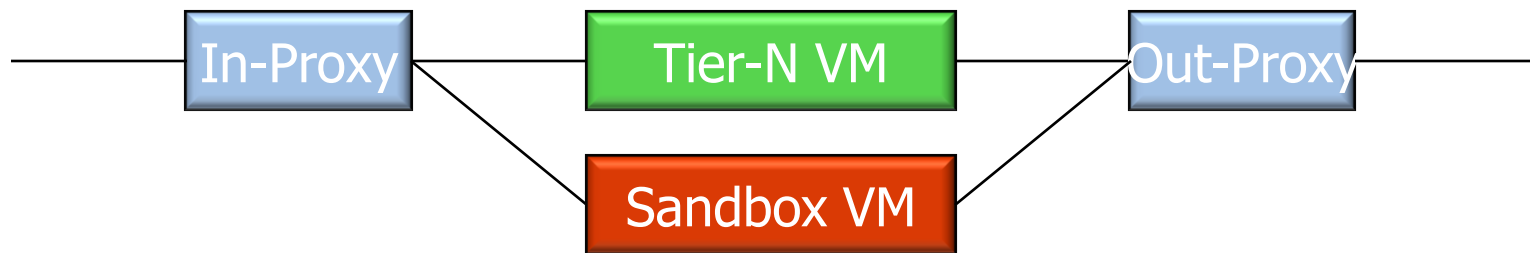
S\_W2

S\_A2

S\_D2

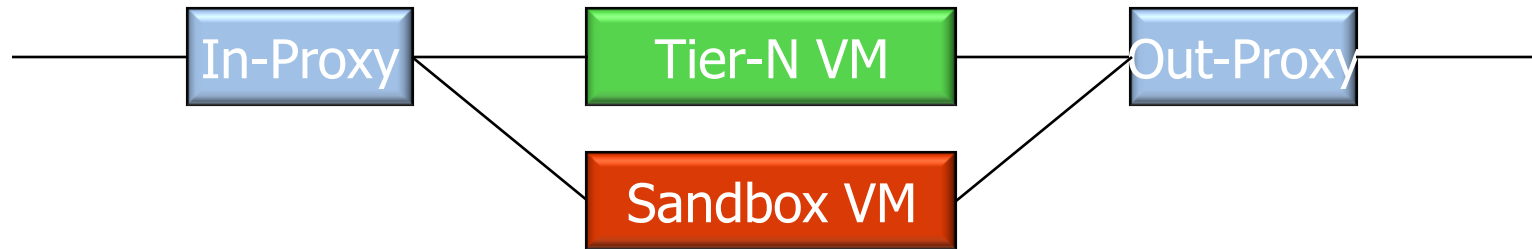
# Experimenter

- Step 2: Workload replay using network proxies



- Proxies filter requests/replies from the sandbox VM
- Emulates the timing and functional behavior of preceding and following service tiers
  - Application protocol level requests/replies (e.g. HTTP)
- In-proxy replays with fixed delay
  - Delay needed if sandbox is faster than production system

# Proxies



Req0	t0	Resp0	t0'	Resp0	ts0'
Req1	t1	Resp1	t1'	Resp1	ts1'
Req2	t2	Resp2	t2'	Resp2	ts2'
Req3	t3	Resp3	t3'	Resp3	ts3'

Req0	t00	Resp0	t00'
Req1	t01	Resp1	t01'
Req2	t02	Resp2	t02'
Req3	t03	Resp3	t03'

Reqn	tn	Respn	tn'	Respn	tsn'
------	----	-------	-----	-------	------

Reqn	t0n	Respn	t0n'
------	-----	-------	------



Throughput

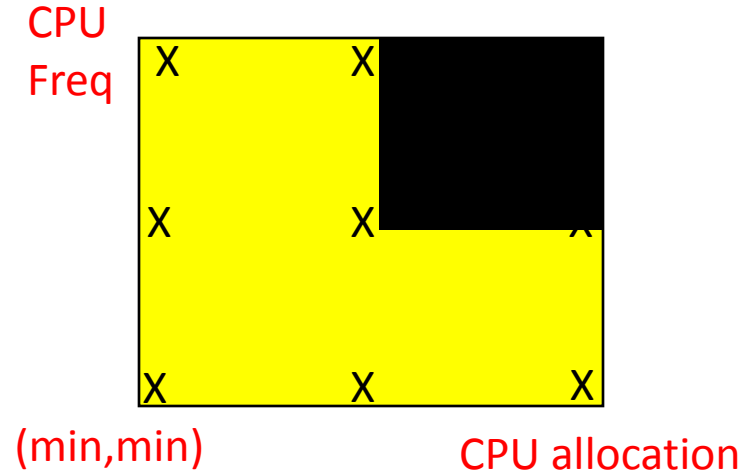
Online & Sandbox

Mean response time

Online & Sandbox

# Experiment Driver

- Fill in results matrix within a time limit



- Corners
- Midpoints (recursive)
- Heuristics
  - Cancel experiments if gain for a resource addition falls below a threshold
  - Cancel experiments for tiers that do not produce the largest gains from a resource addition

# Evaluation Overview

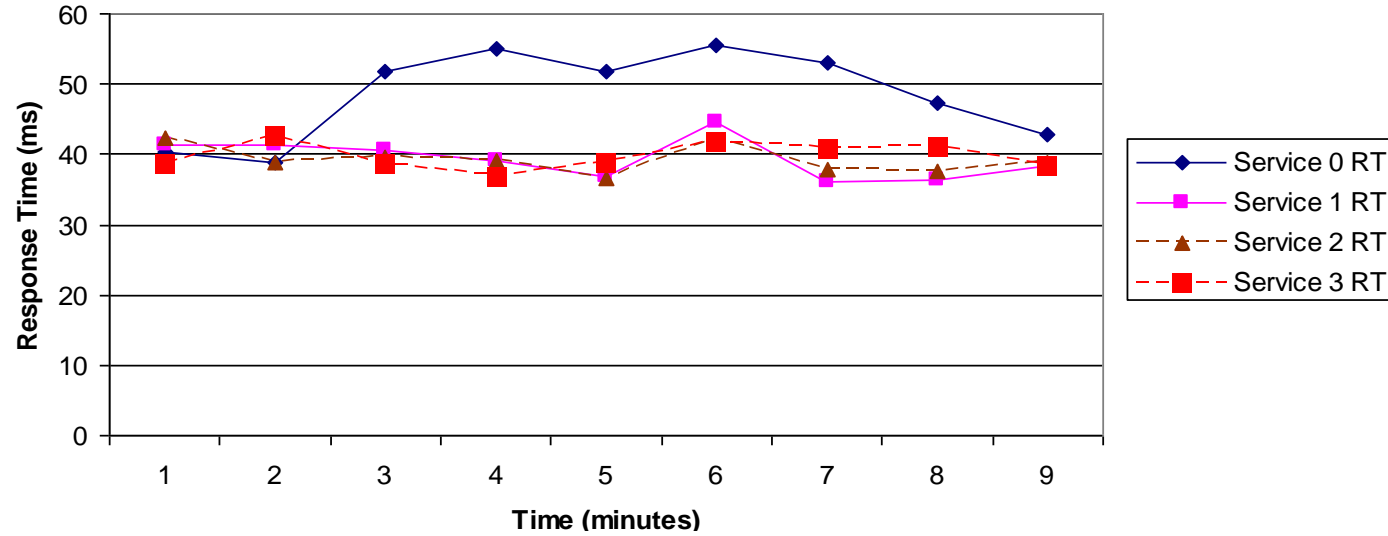
- Implementation
  - Xen (< 50 lines python, <250 lines of C in netback)
  - Proxies for HTTP, mod\_jk, MySQL (800-1500 LoC each)
    - Based on Tinyproxy codebase
    - Two services (auction and bookstore)
- Case studies
  - Resource management: CPU
  - Evaluation of hardware upgrade
- Proxy and Replay Overhead
  - Throughput: No impact
  - Response time: < 5 ms impact

# Case Study 1: Resource Management

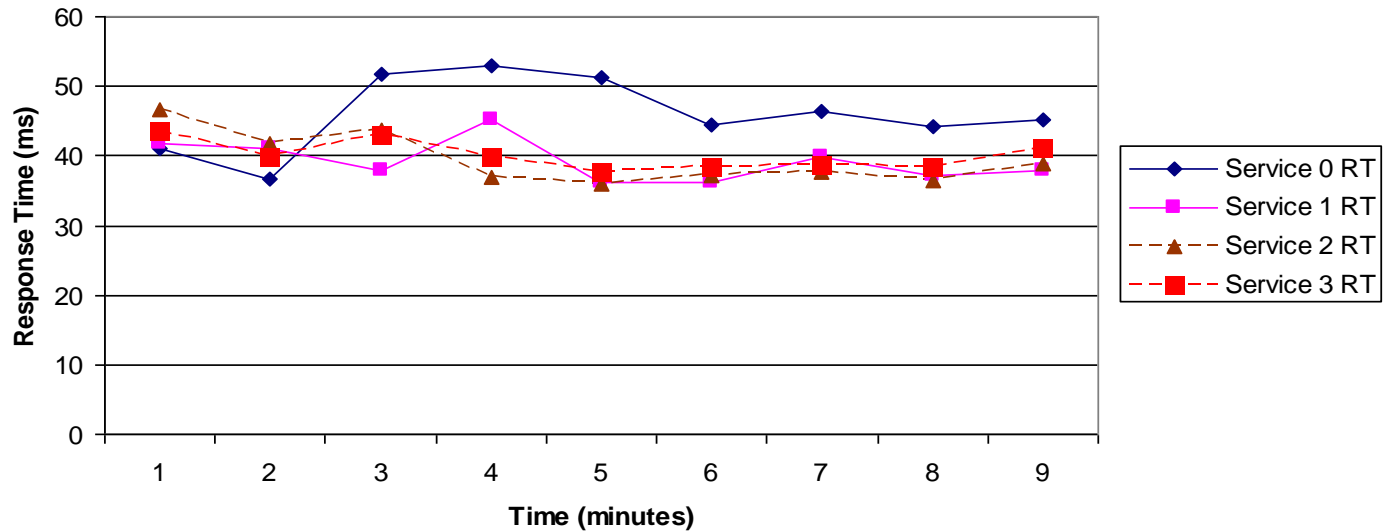
- Goal: satisfy  $< 50\text{ms}$  response time using minimum resource
- JustRunIt determines performance as function of resource allocation
- Management entity assigns resources using a bin packing algorithm (simulated annealing)
  - Minimize resource usage and VM migrations
- Compare experiment-based with a highly accurate model

# Case Study 1 Results

JustRunIt



Highly accurate modeling



# Case Study 2: Hardware Upgrades

- Sandbox on upgraded hardware
- Bin packing finds the number of new machines to accommodate the workload
- Example scenario: two services, each app server uses 90% of one CPU core on old hardware
  - New machine requires 72%
  - Example too small to show any consolidation benefits of upgrading

# Conclusions

- JustRunIt infrastructure can support multiple (automated) management tasks
  - Answers “what-if” questions realistically and transparently using actual experiments
- Possible directions include:
  - Tier interactions
  - Hypothetical workload mix
  - Validate administrator actions
  - Tradeoff between accuracy and experiment cost (resource usage and experiment time required)

# Related Work

- Modeling, simulation of data centers
  - Development and validation cost, simulation slow-down
- Scaled down emulation
  - DieCast [NSDI08] uses VMs and time dilation for emulation (JustRunIt targets subset of mgmt tasks, native execution, and minimal additional hardware)
- Sandboxing for managing data centers
  - Prior work from Rutgers (validating operator actions OSDI04 USENIX06, entire data center experiments EUROSYS07) and file server verification (Tan et al USENIX 05)
  - Very different infrastructure for JustRunIt (extrapolation, verification, application-transparency using VM cloning – practicality)