How to Tame your VM:

an Automated Control System for Virtualized Services

Akkarit Sangpetch
asangpet@andrew.cmu.edu

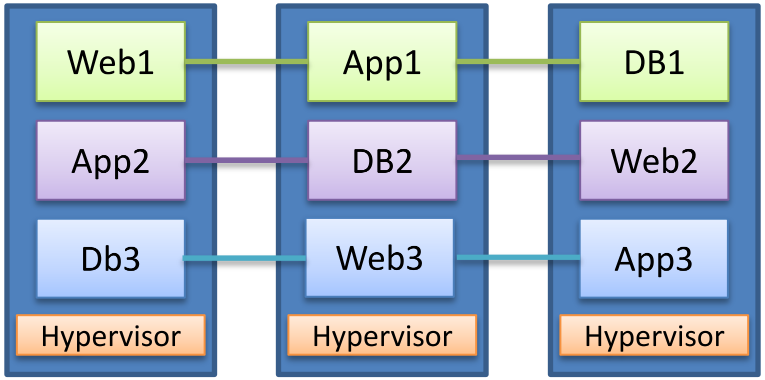
Andrew Turner
andrewtu@andrew.cmu.edu

Hyong Kim
kim@ece.cmu.edu

Department of Electrical and Computer Engineering
Carnegie Mellon University

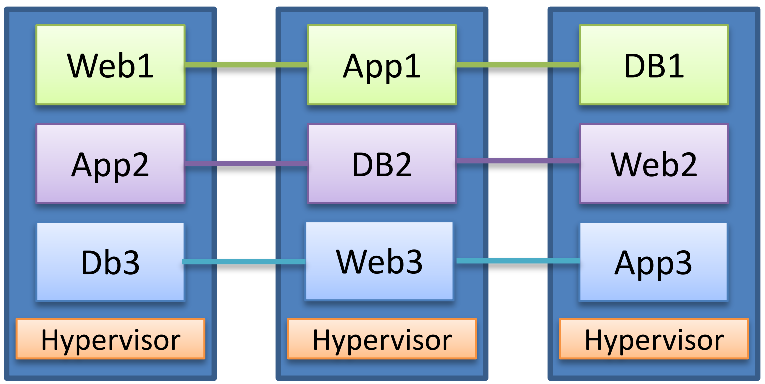
LISA 2010 - November 11, 2010

Virtualized Services



- Virtualized Infrastructure (vSphere,Hyper-V,ESX,KVM)
  - Easy to deploy
  - Easy to migrate
  - Easy to re-use virtual machines
- Multi-tier services are configured and deployed as virtual machines

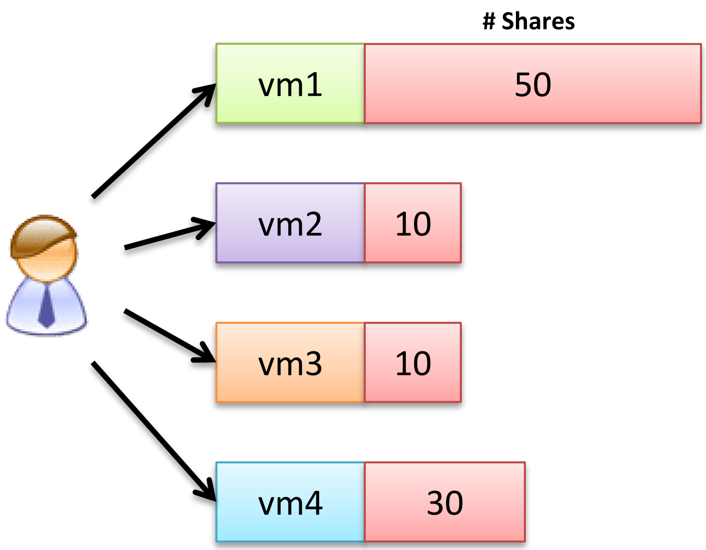
Problem: What about performance?

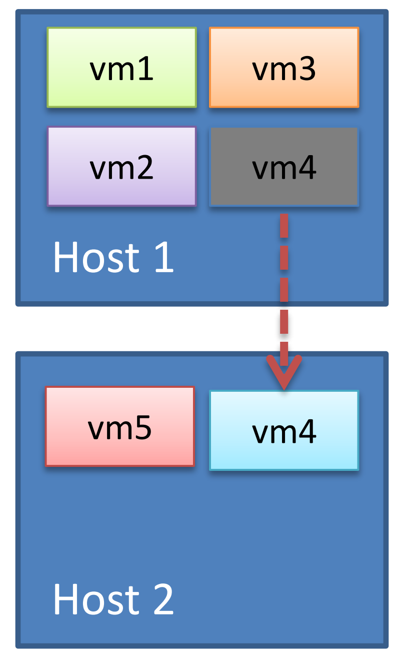


- VMs share physical resource (CPU time / Disk / Network / Memory bandwidth)
- Need to maintain service quality
  - Response time / Latency
- Existing infrastructure provides mismatch interface between the service's performance goal (latency) and configurable parameters (resource shares)
  - Admins need to fully understand the services before able to tune the system

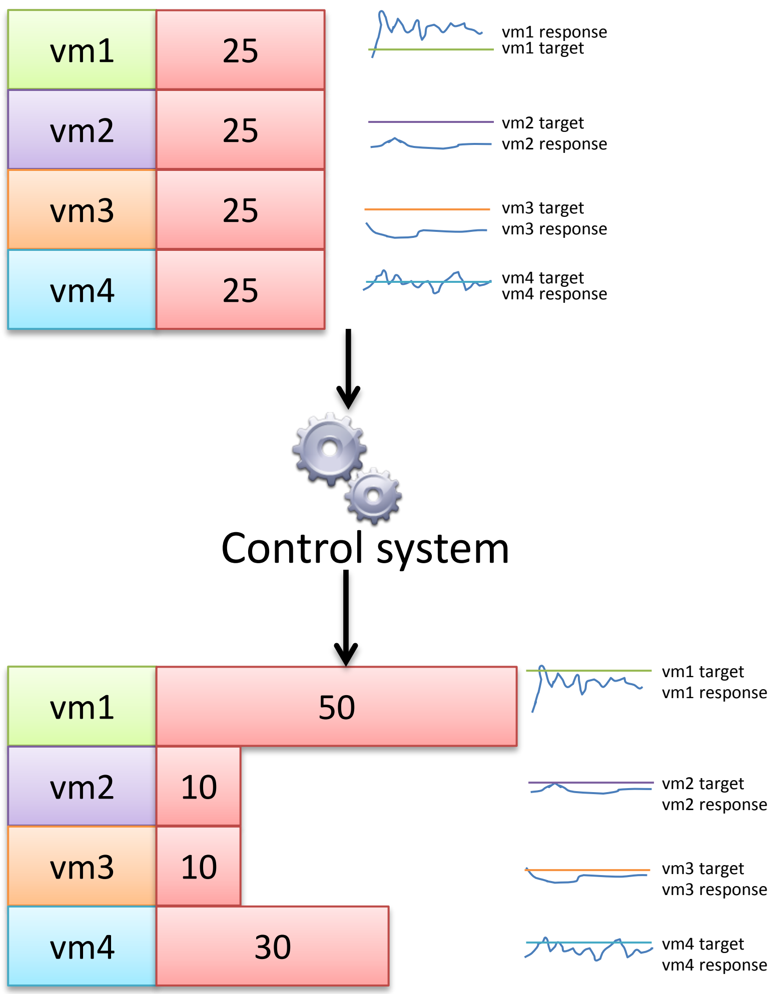
Existing Solutions – Managing Service Performance

- Resource Provisioning
  - Admins define resource shares / reservations / limits
  - The “correct” values depends on hardware / applications / workloads
- Load Balancing
  - Migrate VMs to balance host utilization
  - Free up capacity for each host (low utilization ~ better performance)



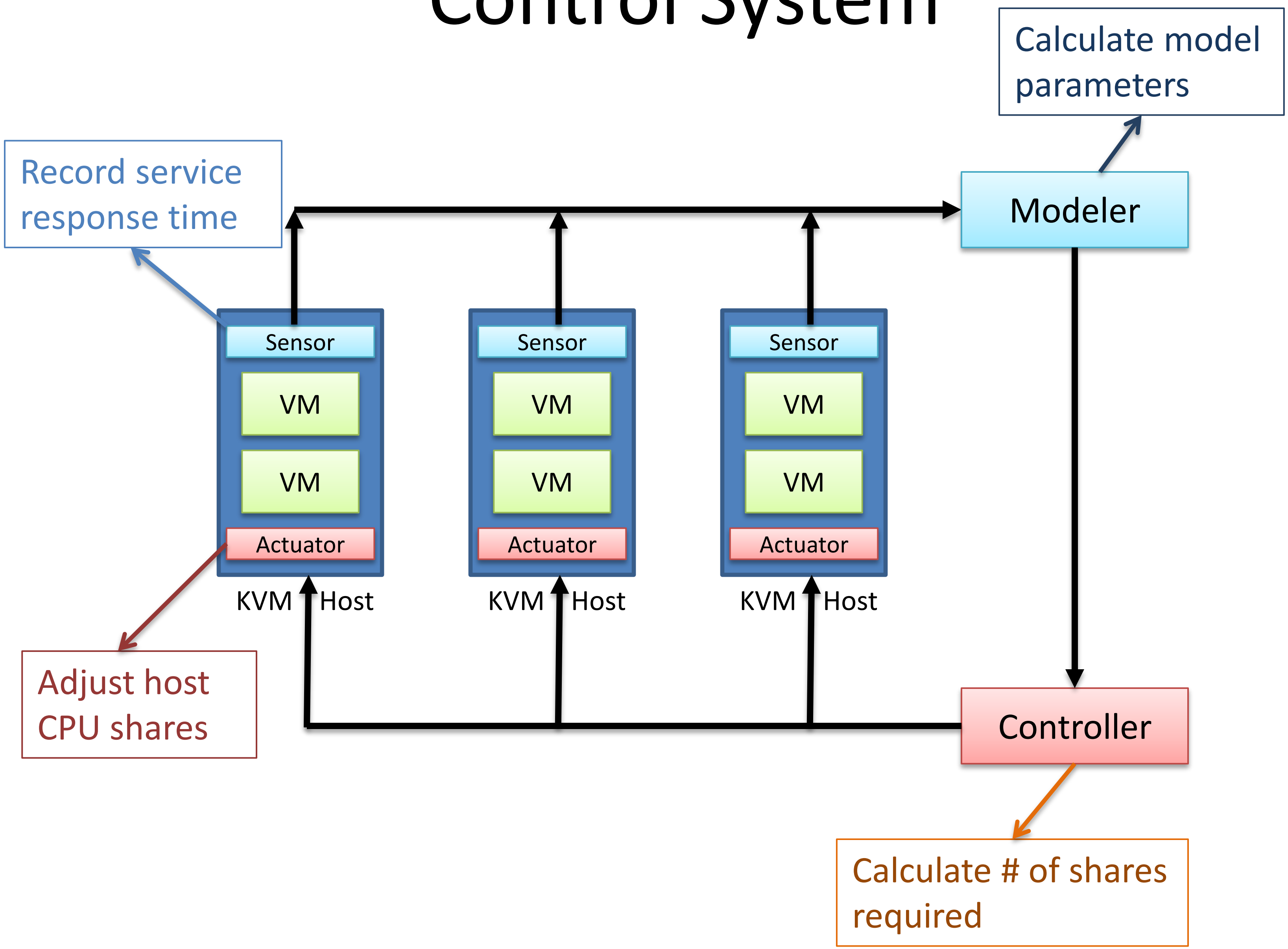


Automated Resource Control

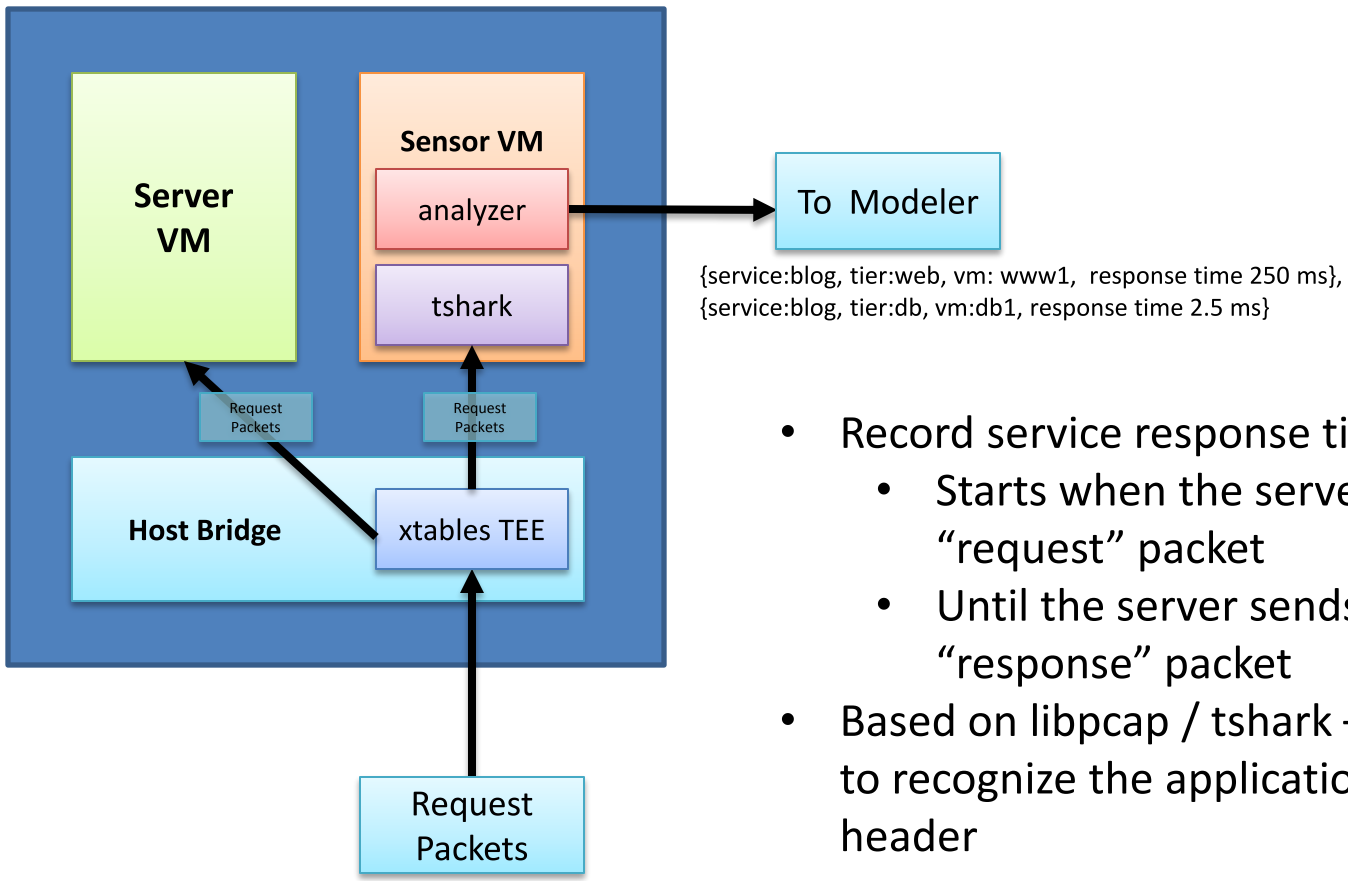


- Dynamically adjust resource shares during run-time
- Input = required service response time
  - I want to serve pages from vm1 in less than 4 seconds
- The system calculates number of shares required to maintain service level
  - Adapt to changes in workload
  - Admins do not have to guess the number of shares

Control System

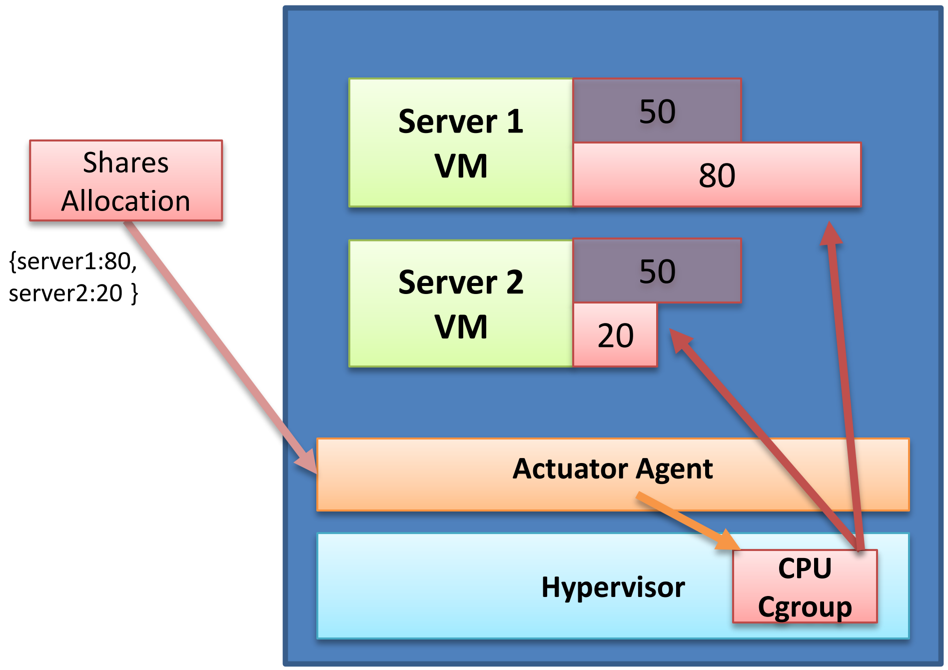


Sensor unit



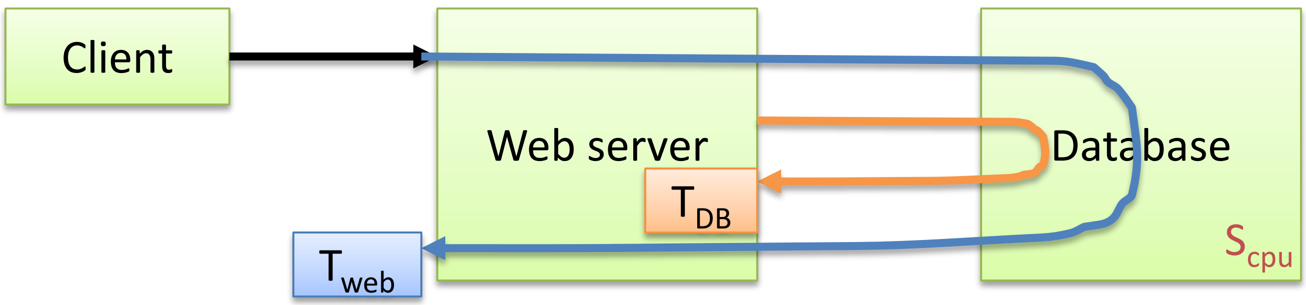
- Record service response time
  - Starts when the server sees "request" packet
  - Until the server sends "response" packet
- Based on libpcap / tshark – need to recognize the application's header

Actuator Unit



- Adjust number of shares based on controller's allocation
- We use Linux/KVM as our hypervisor
  - Cgroup cpu.shares – control CFS scheduling shares
  - Similar mechanism exists on other platform (Xen weight / ESX shares)

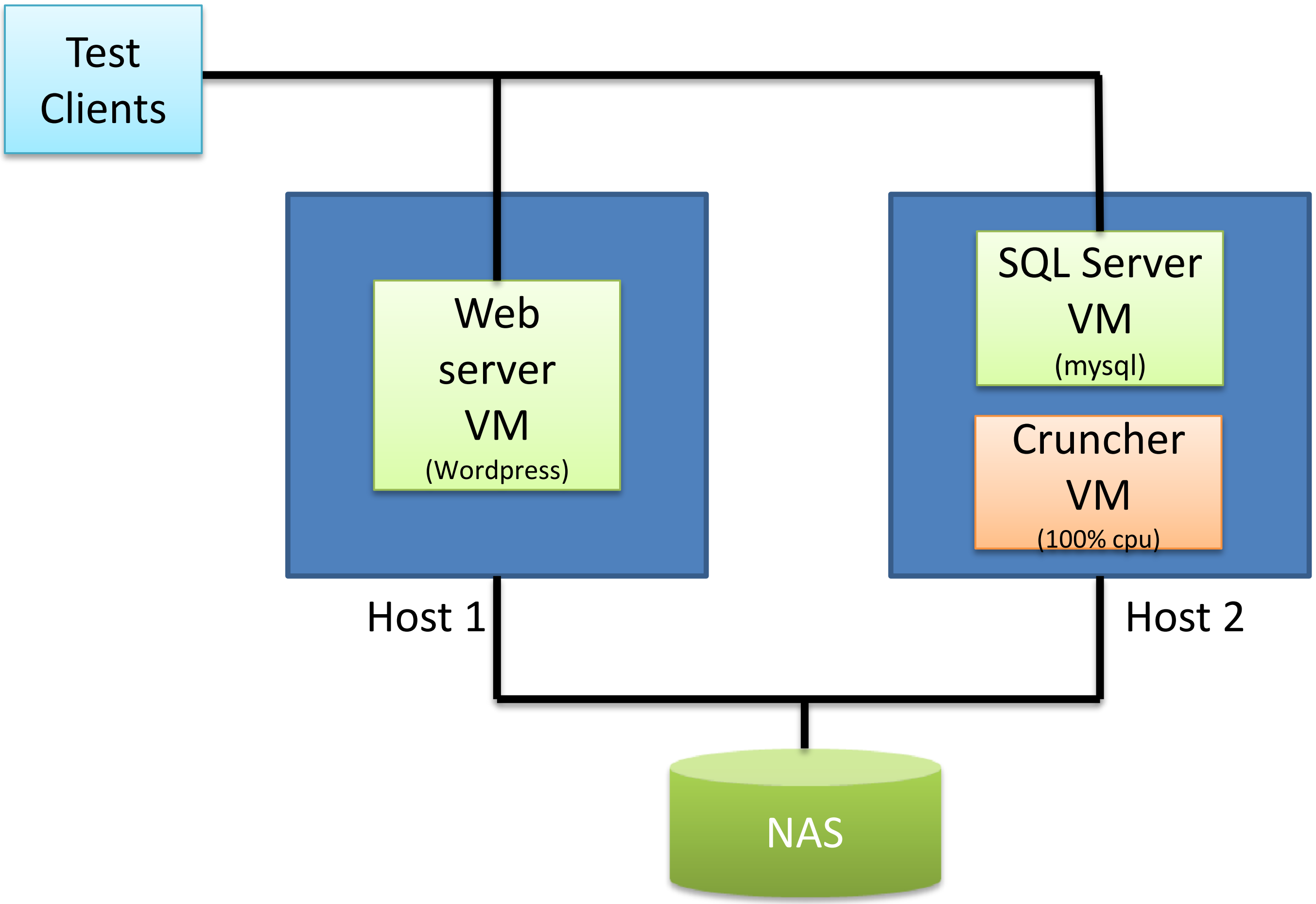
Modeler



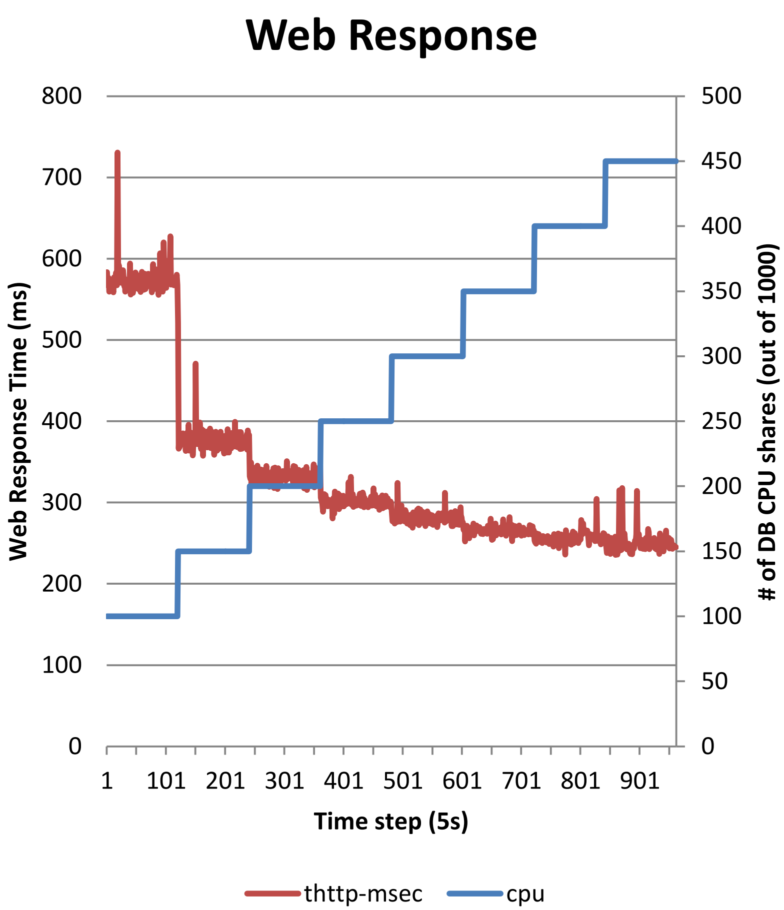
- The modeler calculates the expected service response time based on shares allocated to each VM
- We use an intuitive model for our 2-tier services in the experiment

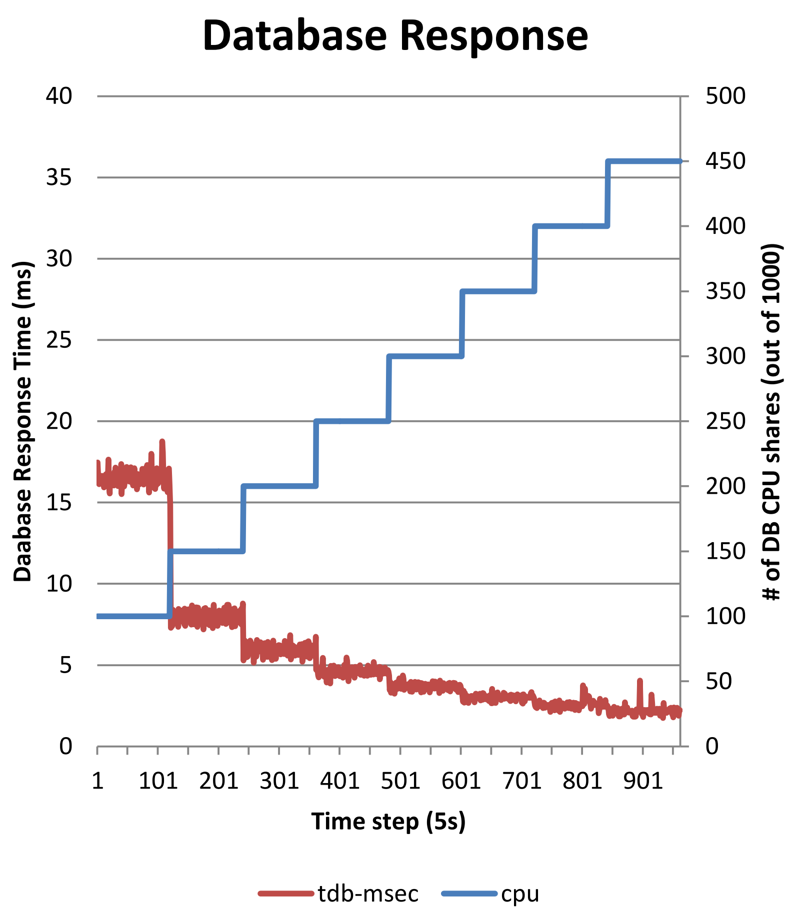
$$T_{web} = f\,(T_{DB})$$
$$T_{DB} = f\,(S_{cpu})$$

Resource Shares Effect

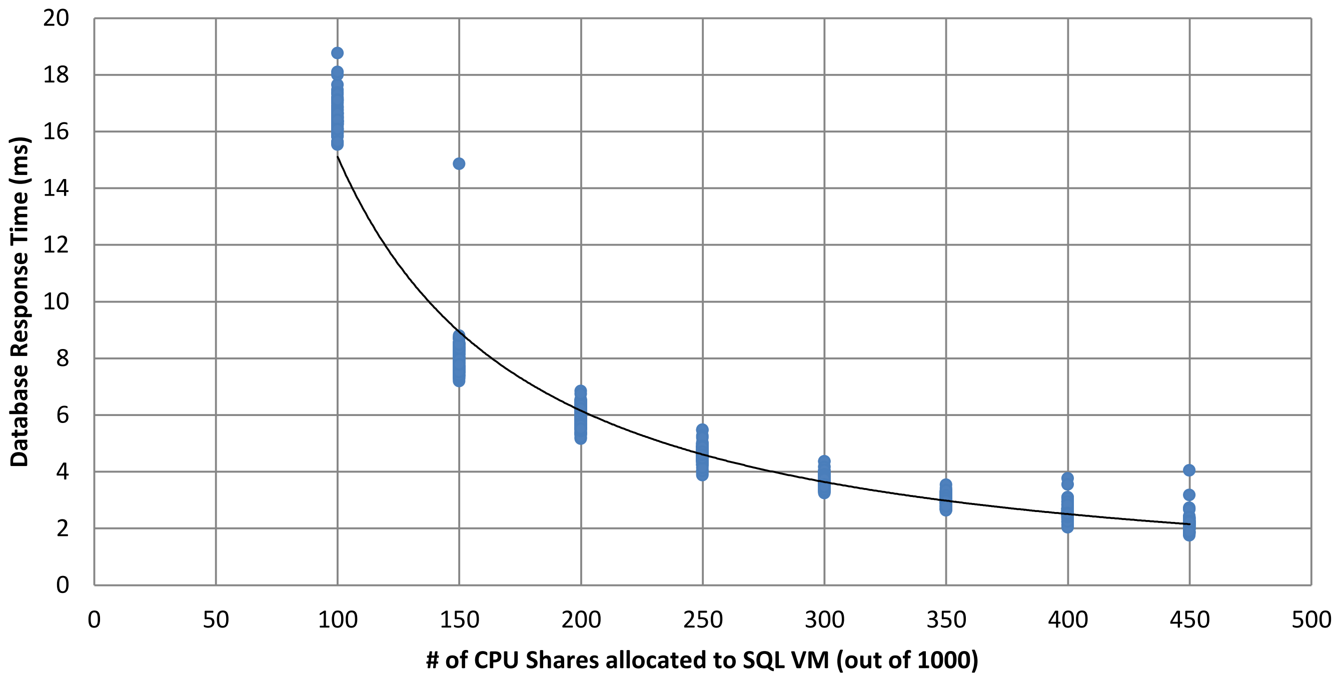


SQL Server CPU Allocation effect



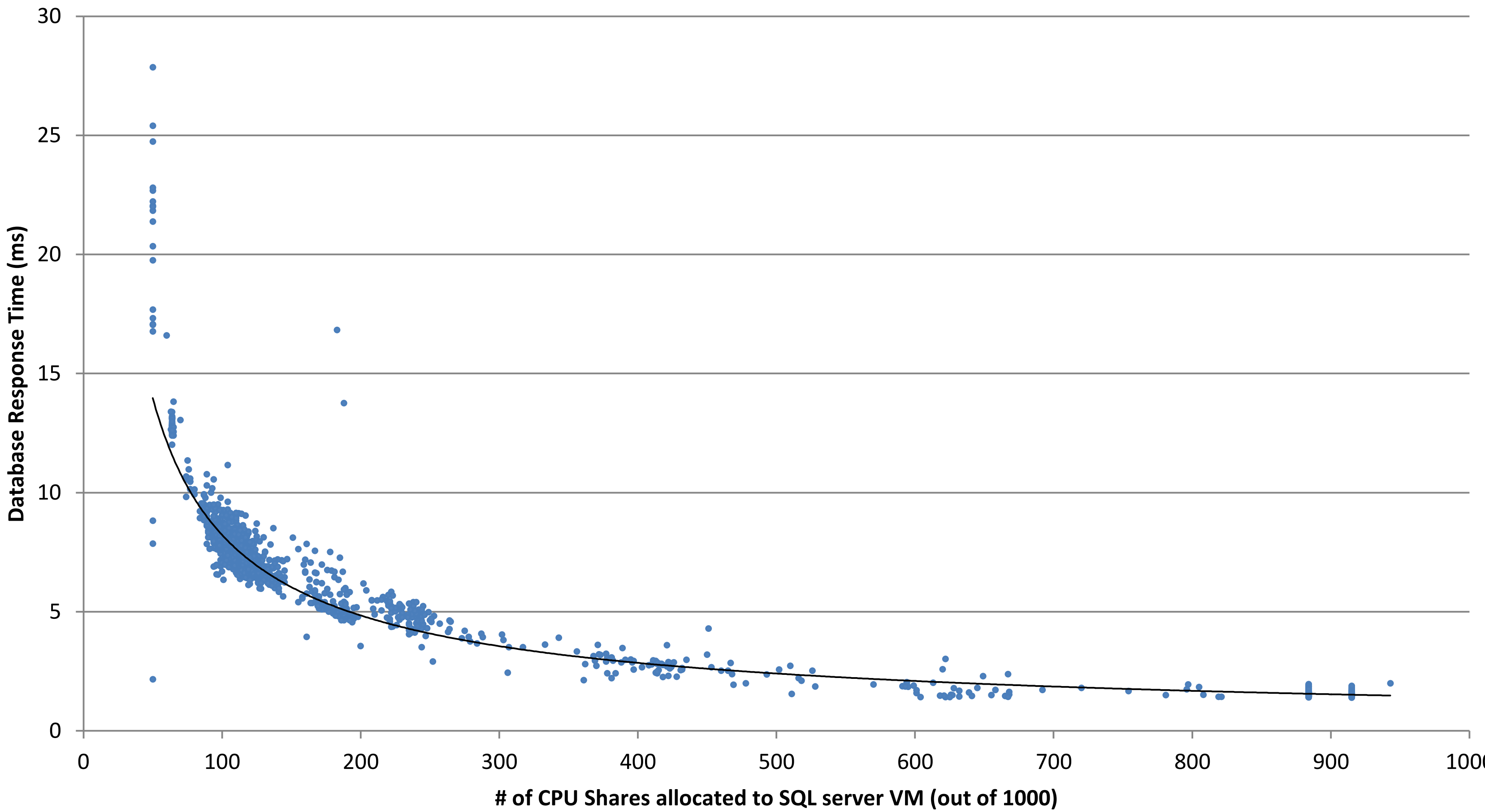


Database Response vs CPU

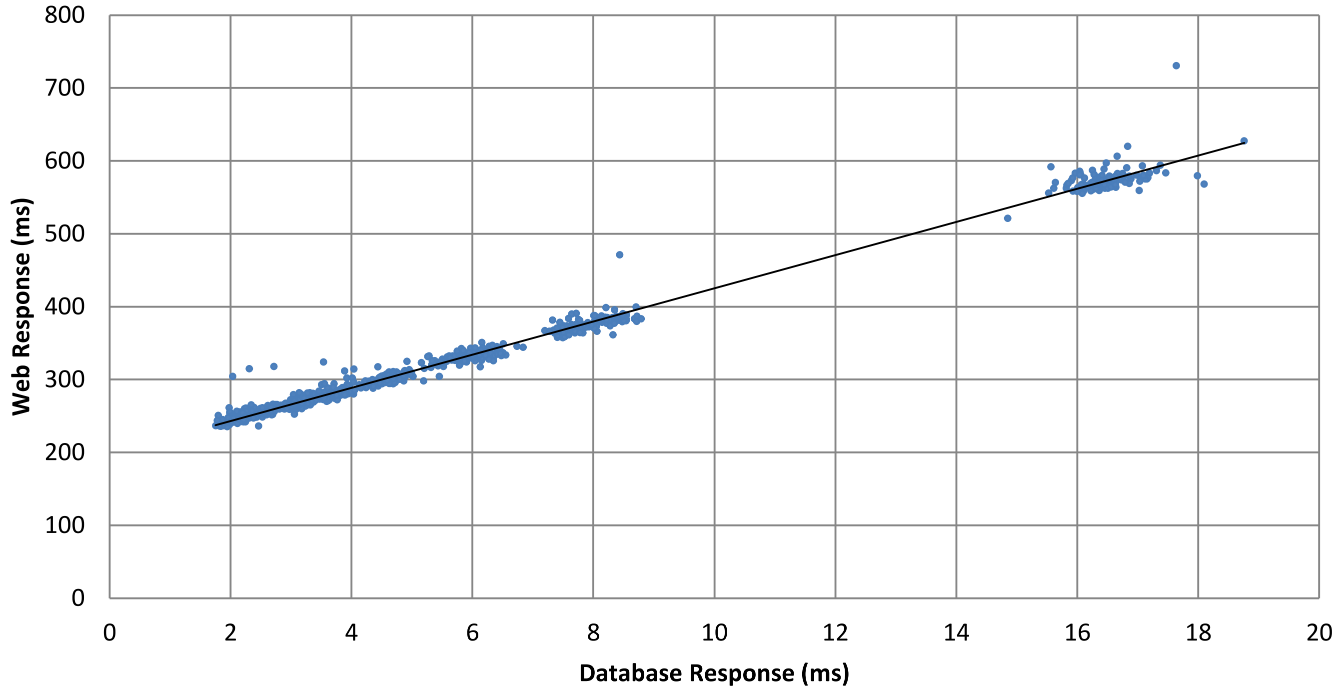


$$T_{DB} = a_0 (S_{cpu})^{b_0}$$

Database Response vs CPU (controlled)



HTTP vs Database response

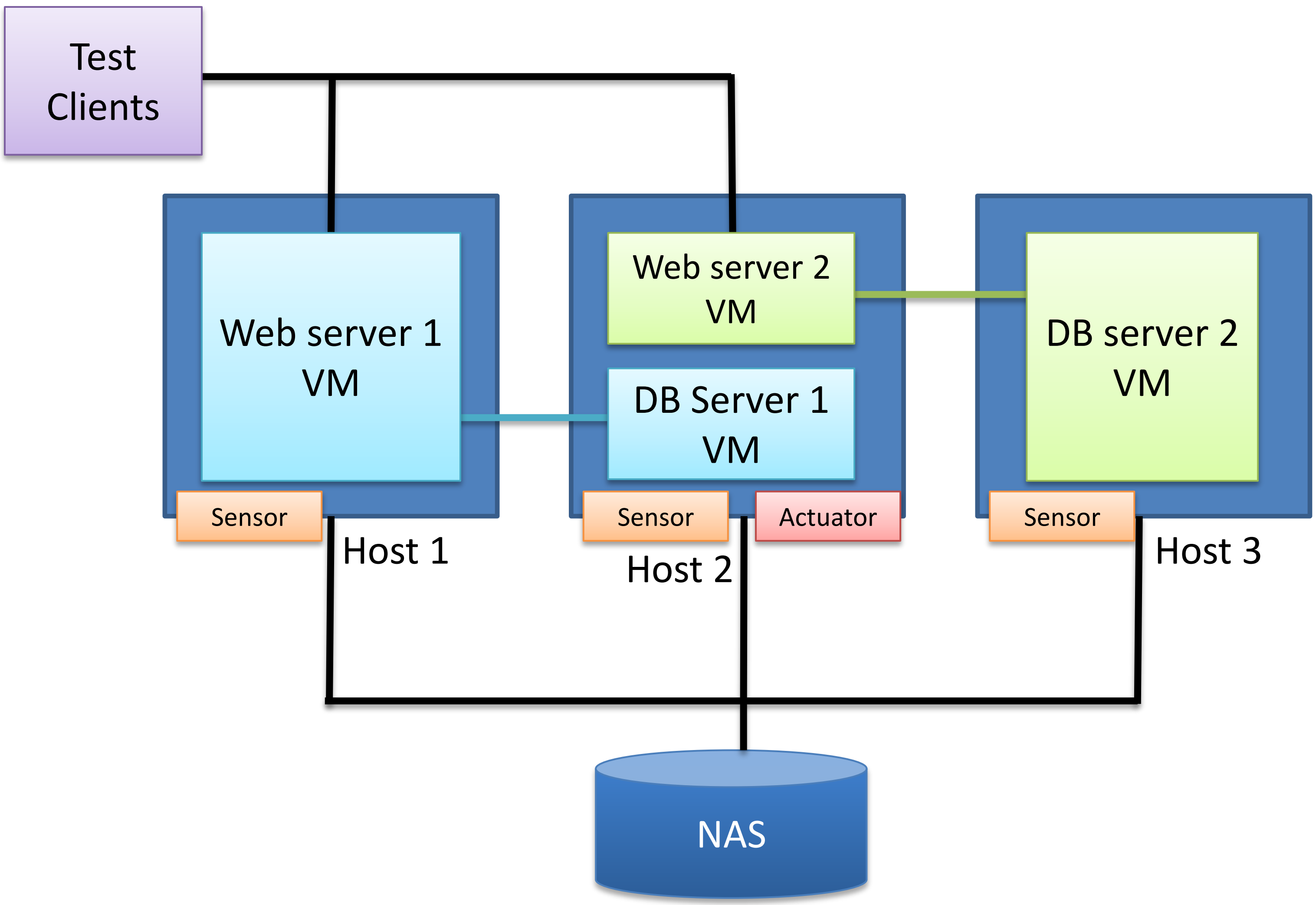


$$T_{web} = a_1 (T_{DB}) + b_1$$

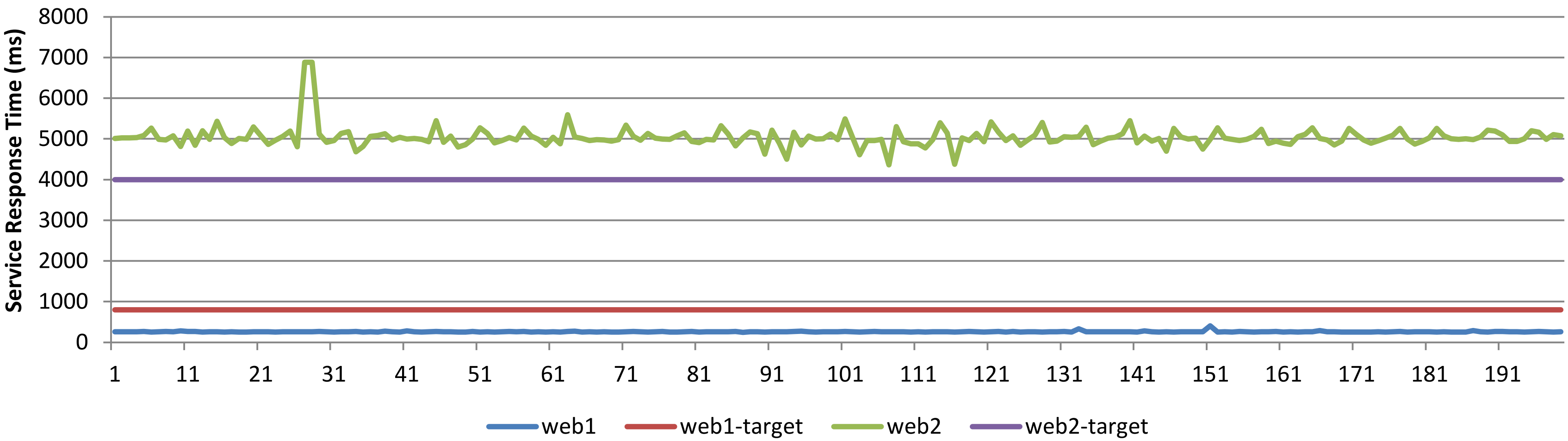
Controllers

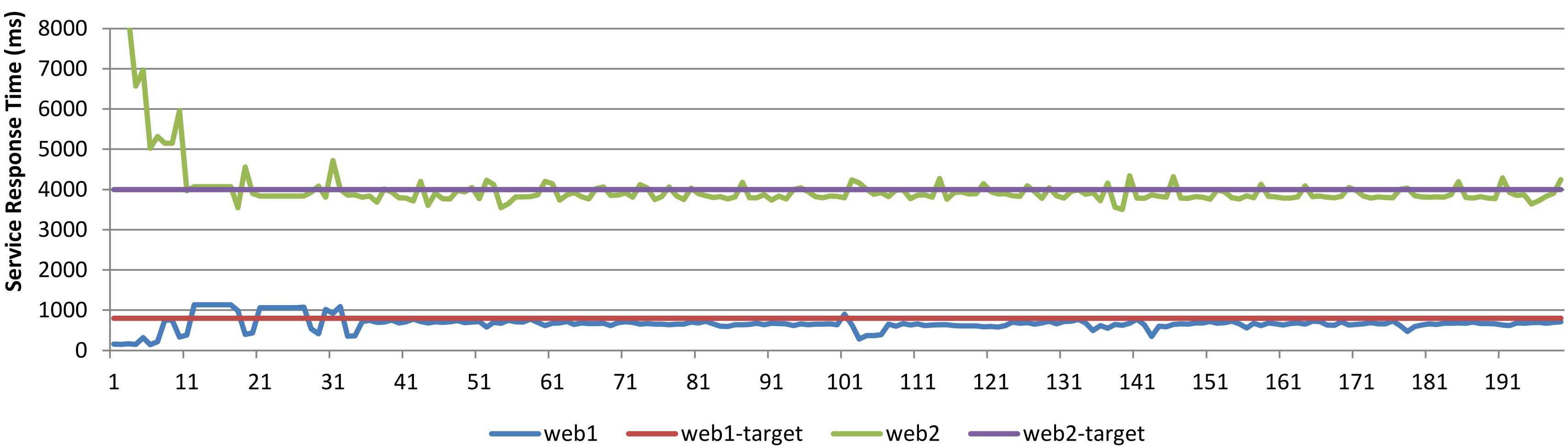
- Model uses readings to estimate the service parameters $<a_0,b_0,a_1,b_1>$
  - $T_{DB} = a_0 (S_{cpu})^{b_0}$
  - $T_{web} = a_1 (T_{DB}) + b_1$
- Controller finds the minimal S_{cpu} such that T_{web} < specified response time
  - Long-term control: uses moving average to find model parameters & shares
  - Short-term control: uses the last-period reading to find model parameters
    - Avoid excessive response time violation while waiting for the long-term model to settle

System Evaluations

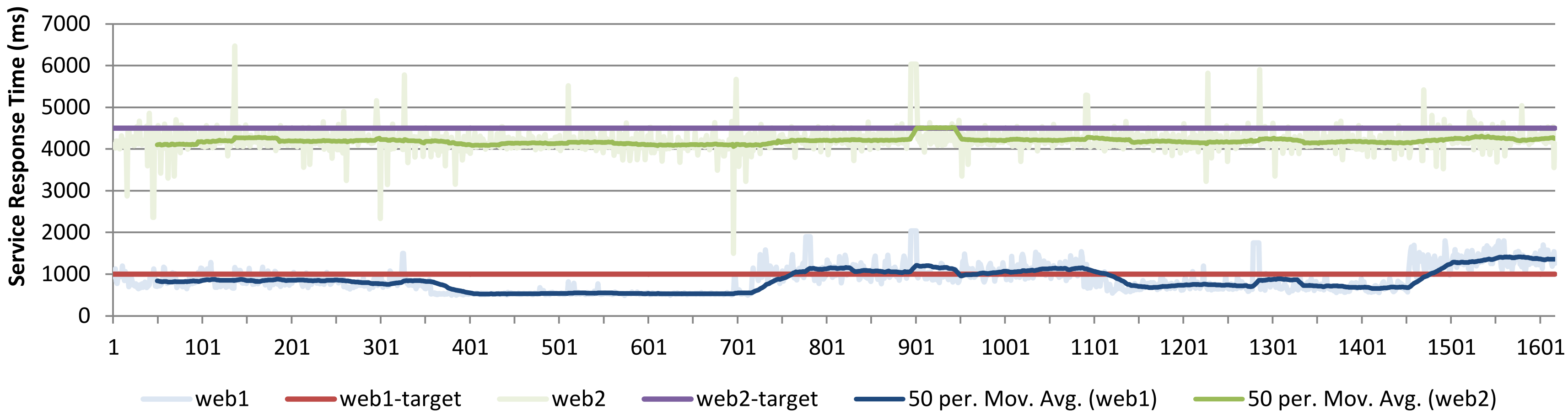


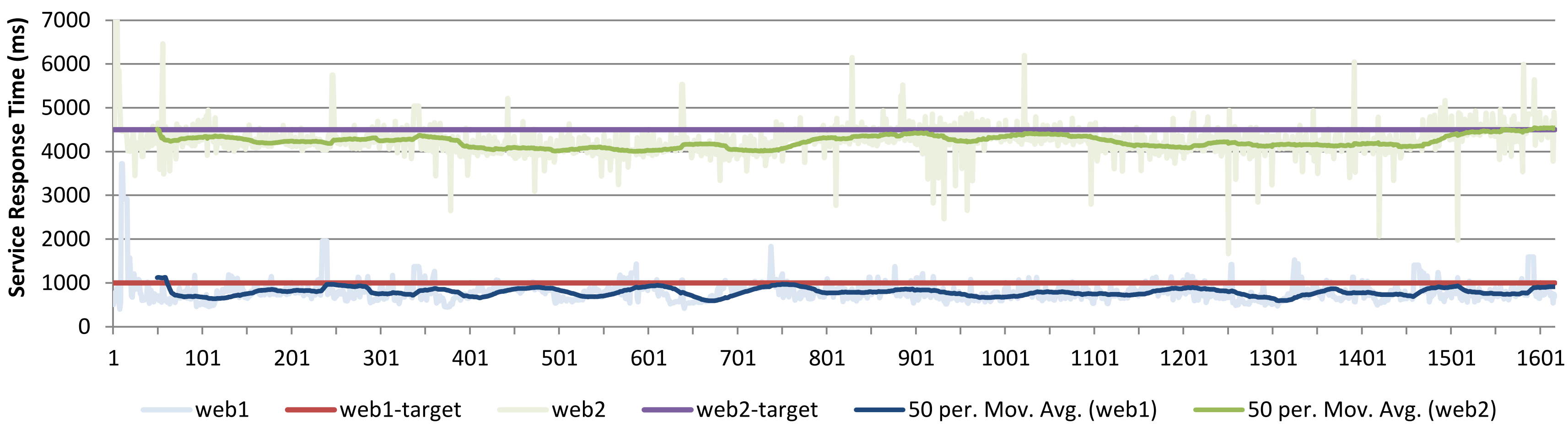
Static Workload





Dynamic Workload





Deviation from Target

	Without Control		With Control	
	Mean Deviation (ms)	Target Violation Period	Mean Deviation (ms)	Target Violation Period
Static Load				
Instance 1	540	0%	109 (-80%)	9%
Instance 2	1043	100%	282 (-73%)	26%
Dynamic Load				
Instance 1	276	7%	182 (-34%)	9%
Instance 2	354	27%	336 (-5%)	12%

Our control system helps track the target service response time

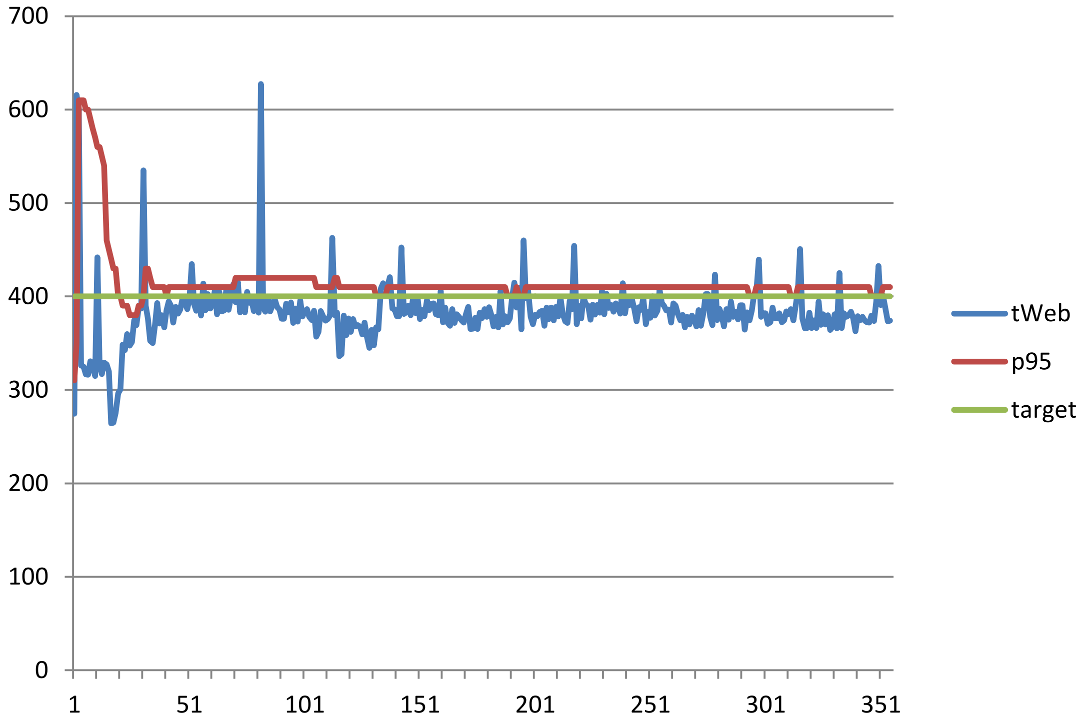
Further Improvements

- Enhance Service Model
  - Service dependencies
    - Cache / Proxy / Load balancer effects
  - Non-cpu resource (disk / network / memory)
- Robust Control system
  - Still susceptible to temporal system effects (garbage collections / cron jobs)
  - Need to determine feasibility of the target performance

Conclusions

- Automated Resource Control system
  - Maintain target system's response time
  - Allows admin to express allocation in term of performance target
- Managing service performance could be automate with sufficient domain knowledge
  - Need basic framework to describe performance model
  - Leave the details to the machines (parameter fitting / optimization / resource monitoring)

Sample Response & Share values



shares