Some Lessons on Dynamic Power Management of Data Centers

Mor Harchol-Balter
Computer Science Dept
Carnegie Mellon University

Anshul Gandhi (CMU)
Mike Kozuch (Intel)
Timothy Zhu (CMU)

Power is Expensive

Annual U.S. data center energy consumption

||

100 Billion kWh or 7.4 Billion dollars

||

 Electricity for over 9 million U.S. homes 

||

As much CO_2 as all of Argentina

All these are expected to increase > 50% over next 5 years.

[energystar.gov]

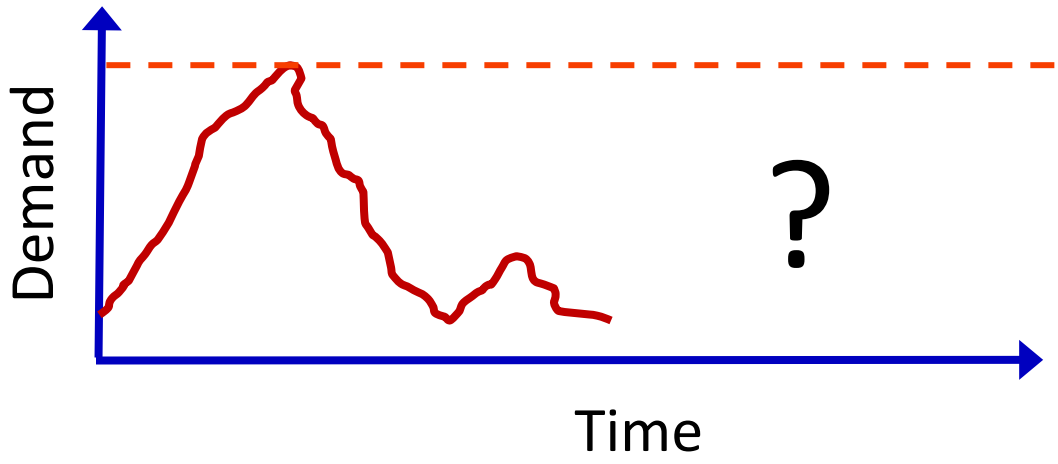
Much power is wasted

Servers only busy 5-30% time on average, but they're left ON, wasting power.
[Barroso, Hölzle '07], [Gartner '10]

Setup time 260s 200W (+more!)

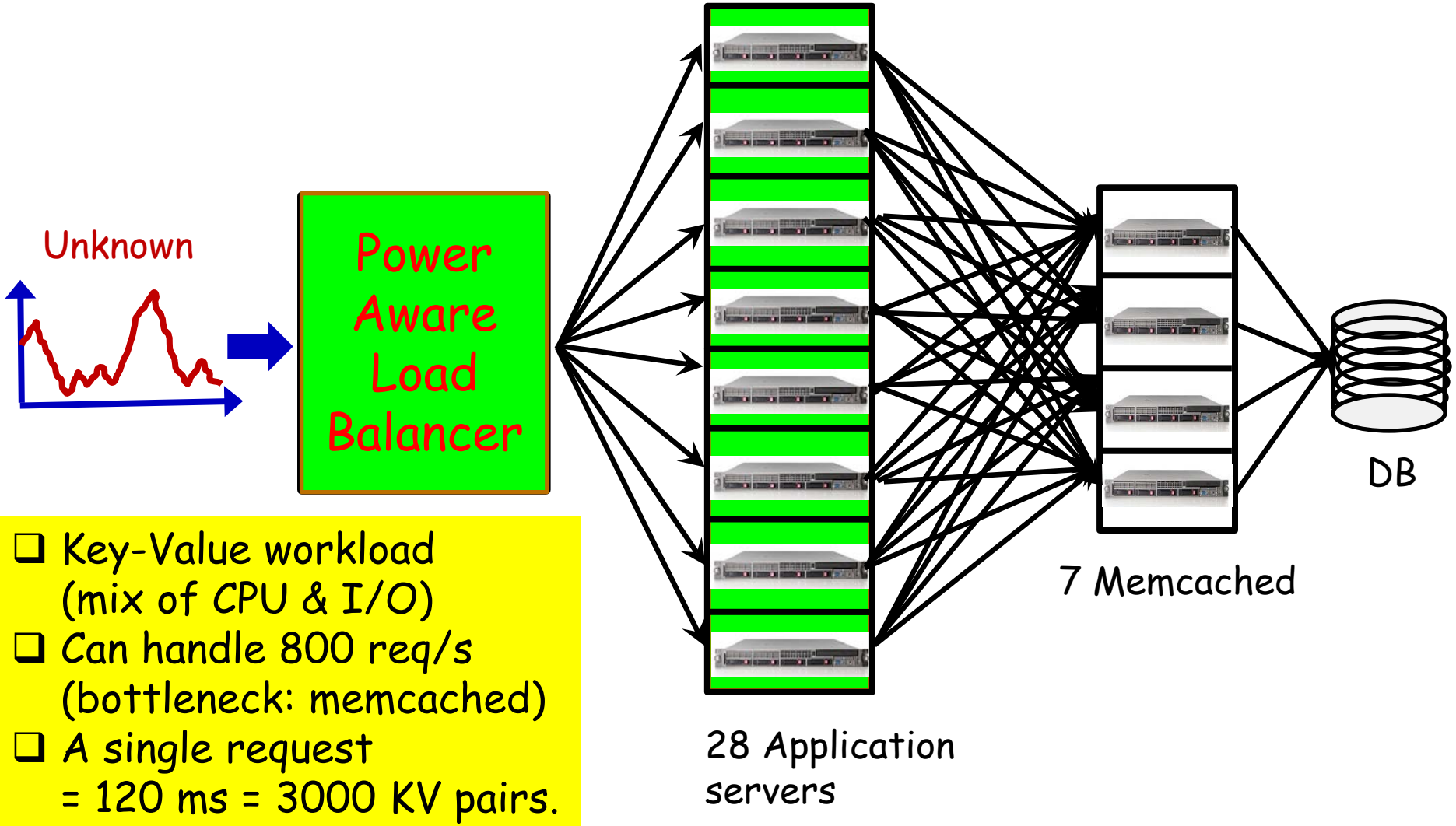
- BUSY server: 200 Watts
- IDLE server: 140 Watts
- OFF server: 0 Watts

Intel Xeon E5520
2 quad-core 2.27 GHz
16 GB memory



Provisioning for peak

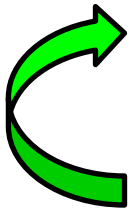
Implementation Testbed



- Key-Value workload (mix of CPU & I/O)
- Can handle 800 req/s (bottleneck: memcached)
- A single request = 120 ms = 3000 KV pairs.
- Goal: T_{95} < 500ms

Setup Cost is non-trivial

Setup time 260s 200W (+more!)

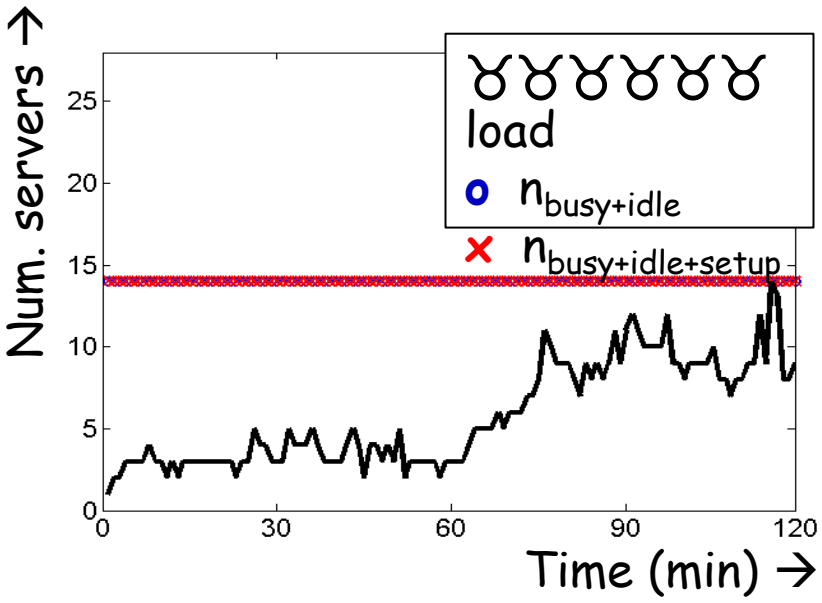


- BUSY server: 200 Watts
- IDLE server: 140 Watts
- OFF server: 0 Watts

Setup time ~ 2500 × Job Size
~ 500 × SLA

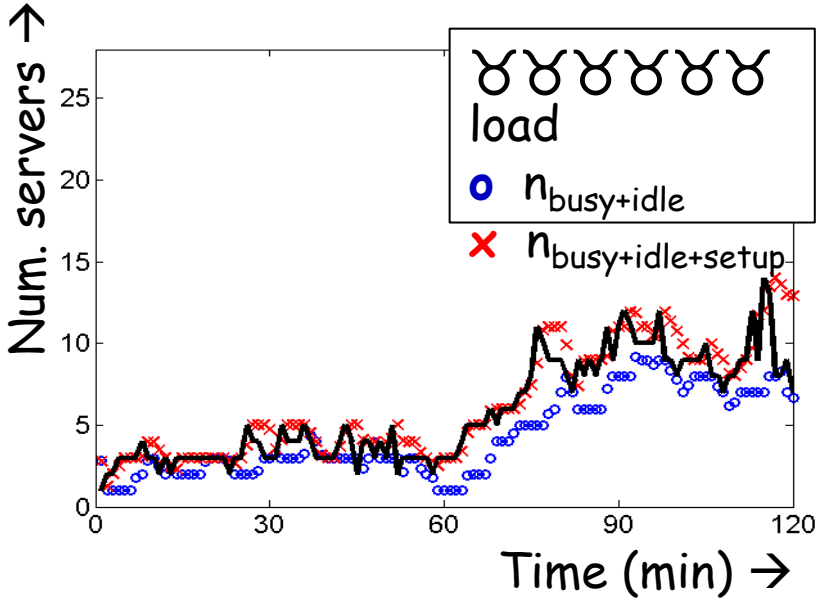
Lesson 1: Setup time can kill performance

AlwaysOn



T_{95}=291ms,
P_{avg}=2,323W

Reactive

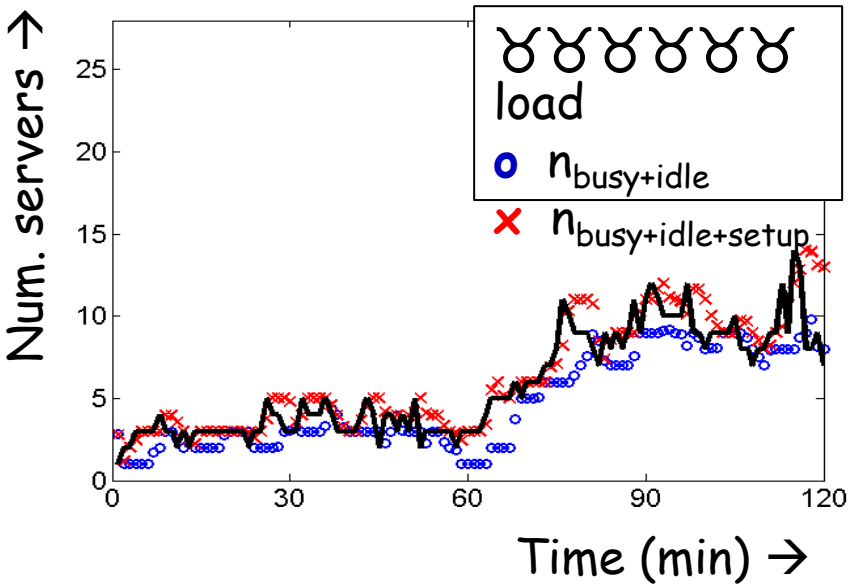


T_{95}=11,003ms,
P_{avg}=1,281W

Quick oscillations are deadly given setup time.

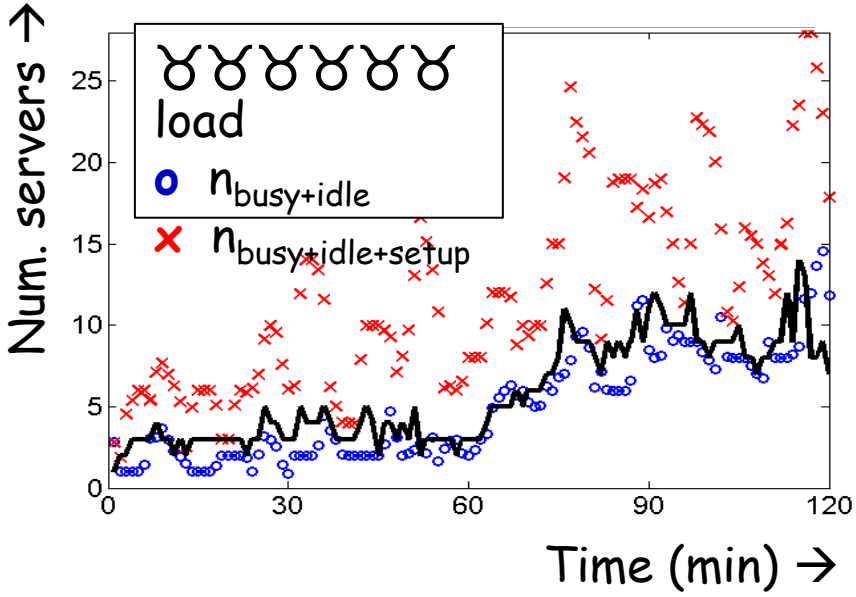
Lesson 1: Setup time can kill performance

Predictive: MWA



T_{95}=7,740 ms, P_{avg}=1,276W

Predictive: LR

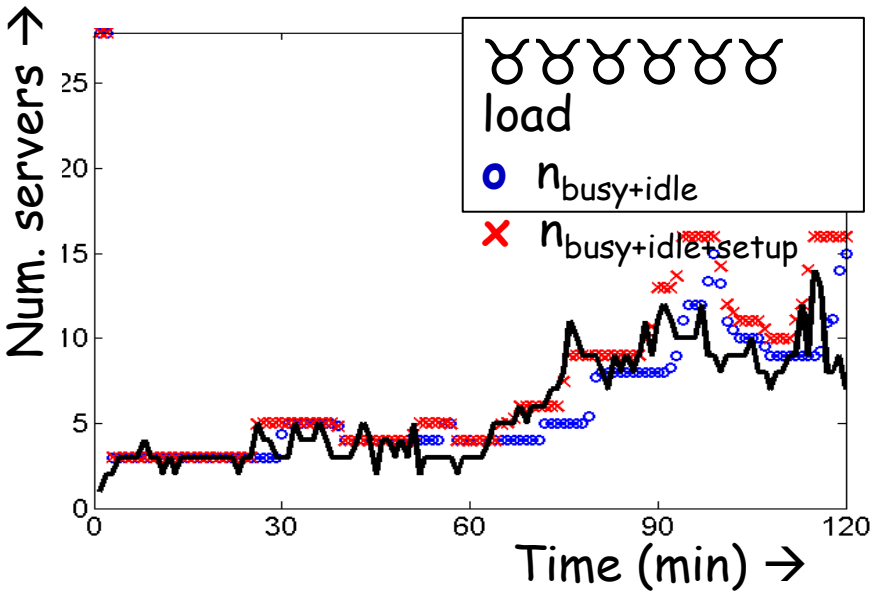


T_{95}=2,544ms, P_{avg}=2,161W

Hard to "predict" the unpredictable.

Soln 1: Better than prediction: just delay turning servers off.

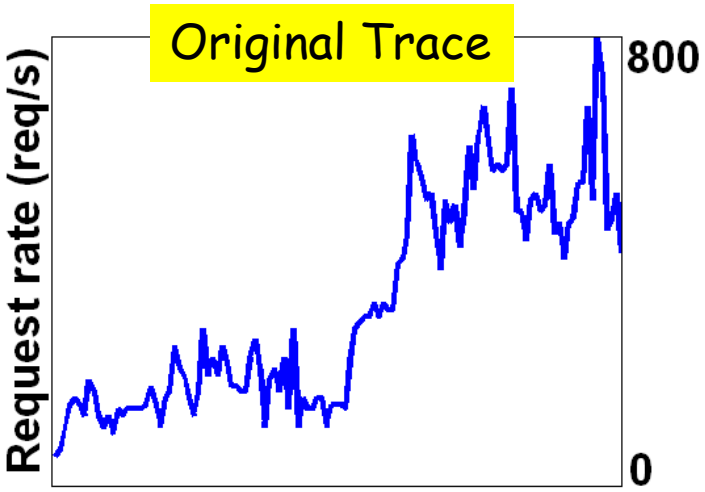
AutoScale

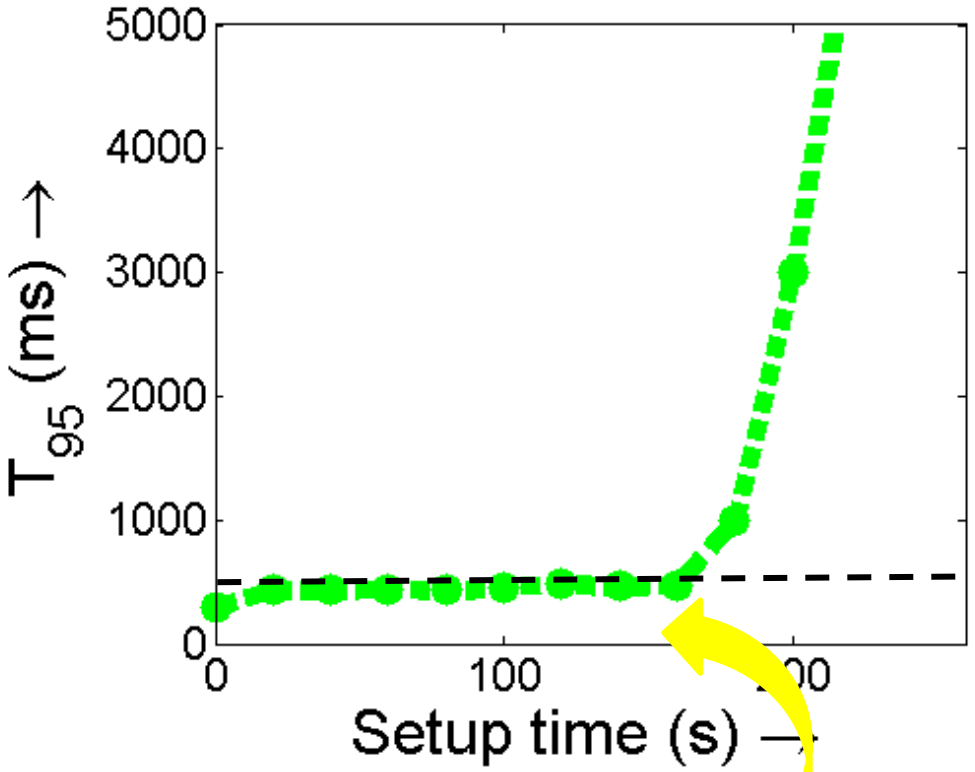


T_{95}=474ms, P_{avg}=1,387W

	T_{95}	P_{avg}
AlwaysOn	**291ms**	**2,323W**
Reactive	**11,003ms**	**1,281W**
Predictive MWA	**7,740ms**	**1,276W**
Predictive LR	**2,544ms**	**2,161W**
AutoScale	**491ms**	**1,297W**

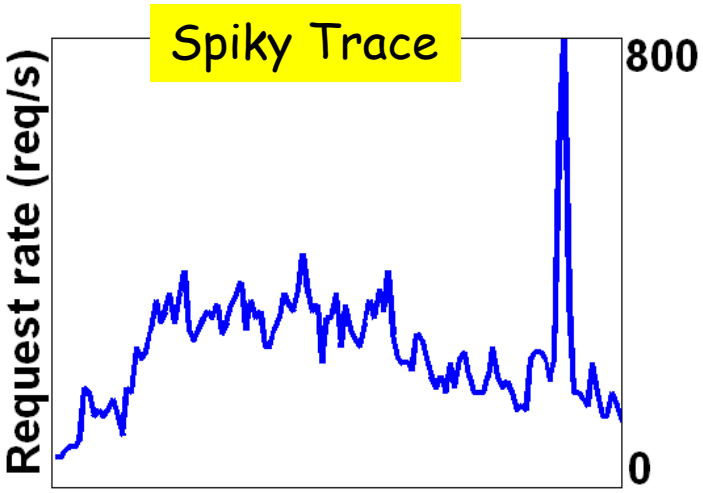
Lesson 2: When have bad spikes, even small setup is deadly

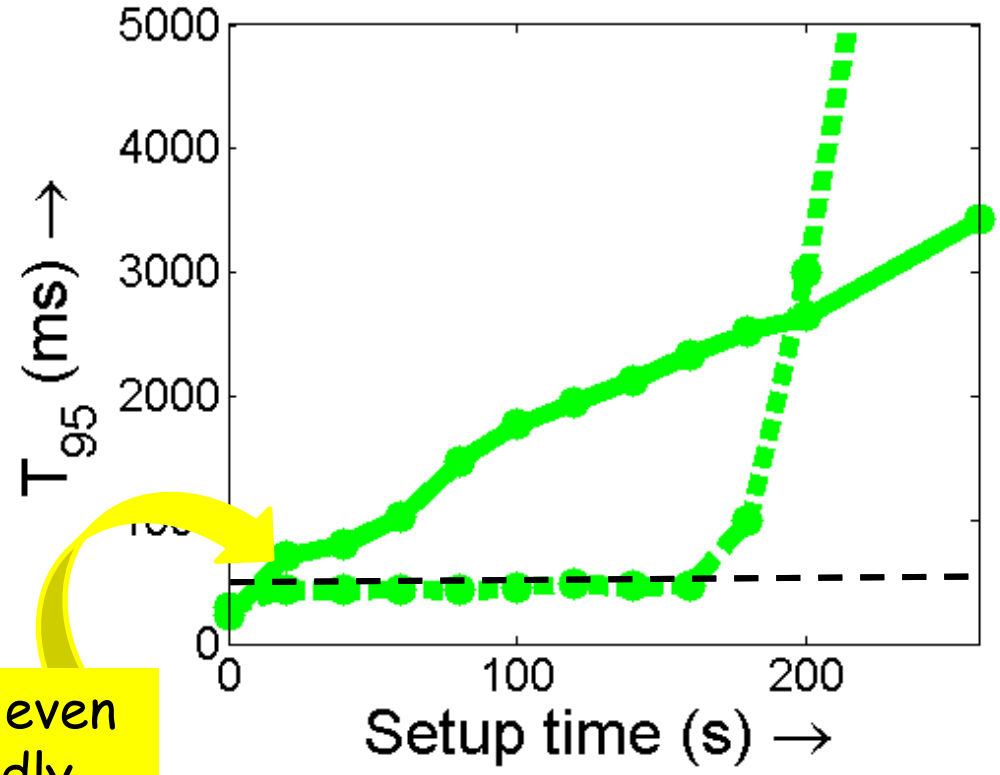




Normally, dropping the setup time helps a lot.

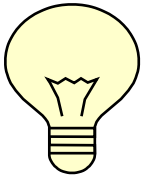
Lesson 2: When have bad spikes, even small setup is deadly





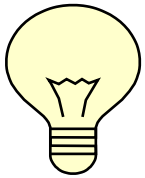
For spiky trace, even 20s setup is deadly.

Soln 2: Need zero setup time source

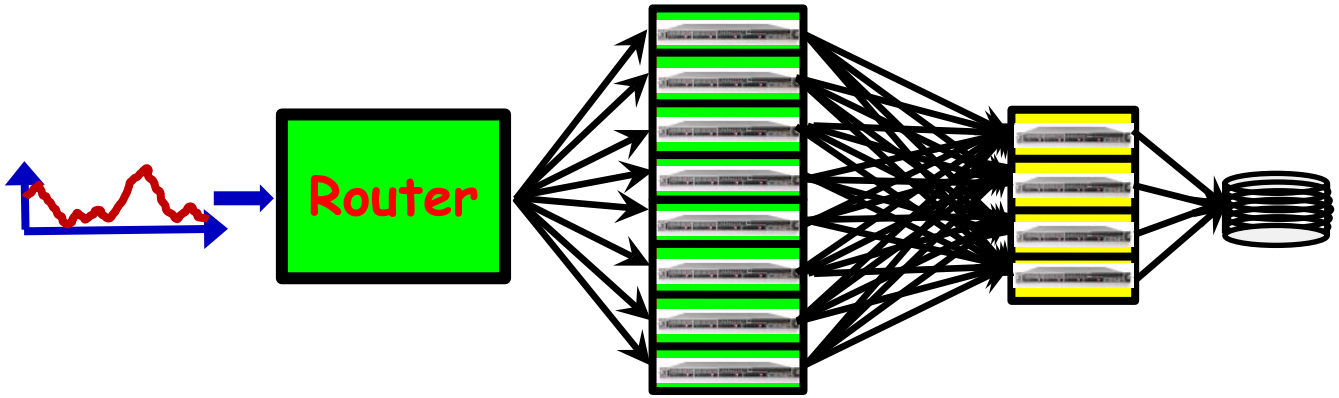


Extra App servers

But for spiky trace, need 100% extra capacity ☹



Temporarily use Memcache. (see our poster)



Lesson 3: Arrival rate is not enough

- Arrival rate might change
- Job sizes sometimes change
- Server speed might change

All dynamic provisioning algorithms in literature use only change in arrival rate.
What we really want to know is change in load.

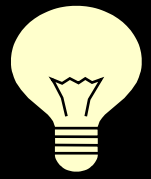
"Load" = (avg arrival rate) x (avg service reqt)

If load doubles, need to double # servers.

Lesson 3: Arrival rate is not enough

"Load" = (avg arrival rate) x (avg service reqt)

Q: But how do we know if load increased?

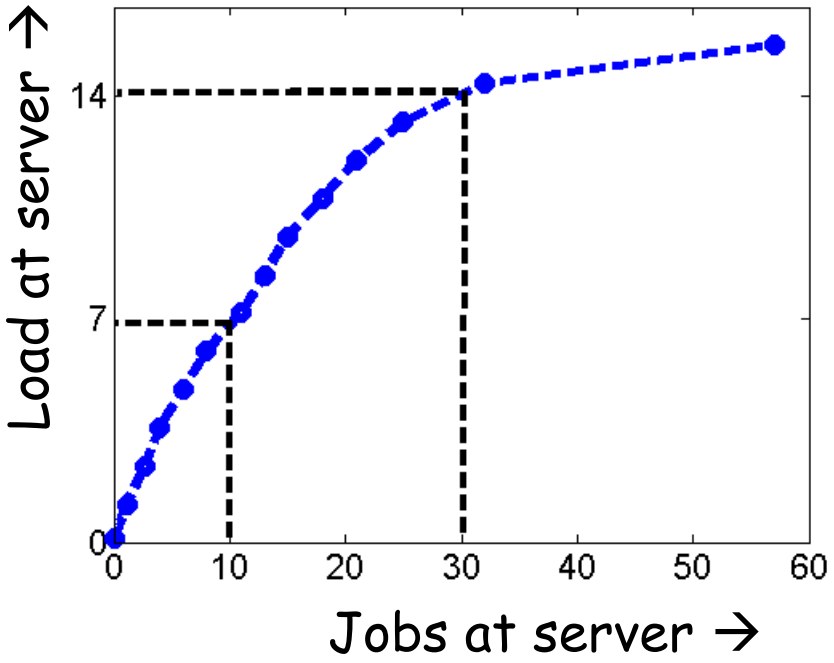


Look at increase in number of jobs.

Q: So if number doubles then load doubled?

A: Not exactly…

Soln 3: Translate number of jobs to load



Number of jobs at each server triples, but load only doubles.

This is the key idea in AutoScale++, which adapts capacity based on # jobs, not request rate.

Lesson 4: Economies of Scale

When scaling up, we always think we need more servers than we actually do …

Easy Example:

Suppose: Jobs require 1 sec on avg.
Arrival rate: 9 jobs/sec ➔ need 12 servers

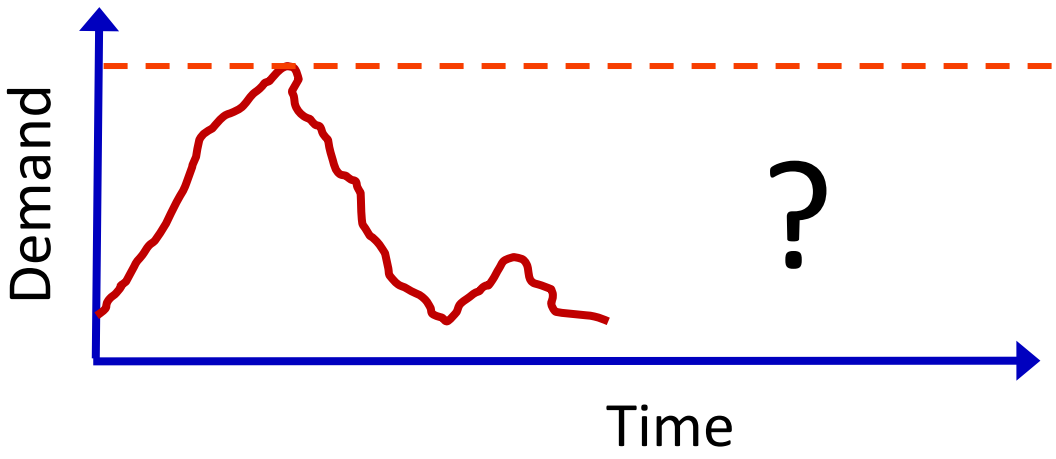
Question: 9×10^6 jobs/sec ➔ ? servers

"Square-root staffing"

Answer: 9.003×10^6 servers

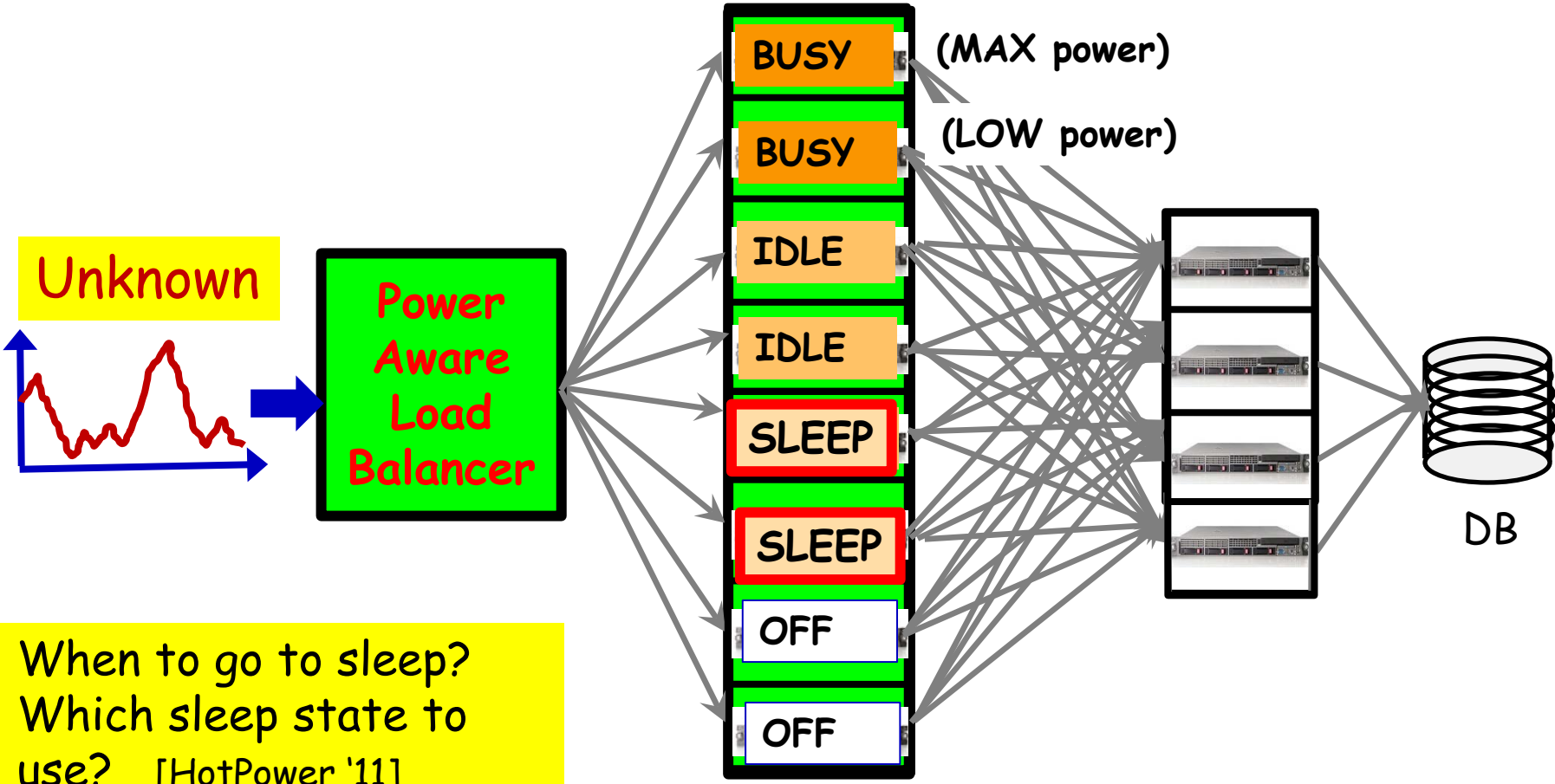
Larger server farms are more efficient.
They are also more tolerant to setup times.

Thank You



harchol@cs.cmu.edu

Future Work: Utilizing sleep states



1. When to go to sleep?
2. Which sleep state to use? [HotPower '11]
3. How many servers to sleep?
4. P-states also useful [Sigmetrics '09]