Energy-efficient Dynamic Capacity Provisioning in Server Farms

VARUN GUPTA
Carnegie Mellon University

Partly based on joint work with:

Anshul Gandhi (CMU)
Mor Harchol-Balter (CMU)
Mike Kozuch (Intel Research)
The “provisioning for peak” problem

PROBLEM: Want to turn servers OFF/ON to match $\rho(t)$…

… and also minimize setup penalties!
First Attempt: ON/OFF policy

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
First Attempt: ON/OFF policy

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
First Attempt: ON/OFF policy

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
First Attempt: ON/OFF policy

WITH SETUP:  \( E[\text{Power}] = 51.9 \text{ kW} \)  \( E[\text{Response time}] = 13 \text{s} \)

NO SETUP:  \( E[\text{Power}] = 14.4 \text{ kW} \)  \( E[\text{Response time}] = 1 \text{s} \)

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
First Attempt: ON/OFF policy

Can we do better than ON/OFF? Add inertia while turning servers OFF

Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
Our Prescription: DELAYEDOFF

Turn OFF a server after idle for \( t_{\text{wait}} \)

\( t_{\text{wait}} \) independent of \( \rho(t) \)!

new arrival routed to the *most recently busy* (MRB) server

arriving job turns a server ON if all servers busy
$t_{\text{wait}}$ timers
$t_{wait}$ timers
$t_{wait}$ timers
THEOREM: For Poisson arrivals, as the load $\rho \to \infty$, the number of ON servers is concentrated around $\rho + \sqrt{\rho \log \rho}$.

--- Proof Intuition ---

Step 1: An equivalent system view

$k$ jobs run on the “first” $k$ servers

Step 2: Analysis of idle periods of $M/G/\infty$

Time from a $k \to (k-1)$ to the next $(k-1) \to k$ transition
Intuition for idle periods

1. \# jobs \approx \text{Normal with mean and variance } \rho

\[ \rho + c\sqrt{\rho \log \rho} \]

2. \[ P_r[\text{jobs } \geq \rho + c\sqrt{\rho \log \rho}] \propto \frac{1}{\sqrt{\rho}} e^{-\frac{c^2 \log \rho}{2}} \]
   \[ \propto \frac{1}{c^2 + 1} \frac{\rho}{2} \]

3. Events happen at rate \( \rho \)

4. Mean idle period of \( \rho + c\sqrt{\rho \log \rho} \) server \( \propto \frac{\rho}{\sqrt{\rho}} \frac{c^2 + 1}{2} \)
   \[ \propto \rho \frac{c^2 - 1}{2} \]

MRB \( \Rightarrow \rho + (1 \pm \epsilon)\sqrt{\rho \log \rho} \) servers for any constant \( t_{\text{wait}} \)!

In practice, we choose \( t_{\text{wait}} \) to amortize setup cost
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
Mean Job Size = 1s  
Setup delay = 200s  
Busy power = 240W

DELAYEDOFF:  
E[Power] = 18.9 kW  
E[Response time] = 1.002s

NO SETUP:  
E[Power] = 14.4 kW  
E[Response time] = 1s
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W
Mean Job Size = 1s
Setup delay = 200s
Busy power = 240W

DELAYEDOFF: E[Power] = 18.9 kW  E[Response time] = 1.002s
√ w/LOOKAHEAD: E[Power] = 15.8 kW  E[Response time] = 1.036s
1. Speed scaling algorithms
2. A simple proxy for MRB routing
3. Heterogeneous servers
4. Managing Virtual Machines in the Cloud
5. Wear-leveling/Performance tradeoffs
Q: Optimal speed to balance energy-performance?

A: your favorite metric = $F(E[\text{energy/job}], E[\text{response time}])$

$$E[\text{energy/job}] \xrightarrow{MRB} \frac{P(s)}{s}, \quad E[\text{response time}] \xrightarrow{MRB} \frac{1}{s}$$

$$s^* = \text{argmin}_s F \left( \frac{P(s)}{s}, \frac{1}{s} \right)$$
A simple proxy for MRB routing

MRB requires a lot of state updates 😞

Proxy policy
- Assign static ranks to servers
- Route a new arrival to *highest ranked idle* server

Almost the same performance as MRB
+ easier to implement than MRB
+ easy to extend
DELAYEDOFF for heterogeneous servers

Assign ranks to servers based on efficiency (1 = most efficient)

Route a new arrival to *highest ranked idle* server
Managing Virtual Machines in the Cloud

2GB RAM

2GB RAM

2GB RAM

1GB

1GB

1GB
Managing Virtual Machines in the Cloud

- Split physical servers into “virtual” servers
- Assign static ranks to virtual servers
- Route a new VM request to highest ranked idle virtual server
- Turn the physical server OFF after each virtual server has idled for $t_{\text{wait}}$
Problem: unpredictable demand and non-trivial setup costs

DELAYEDOFF: A new traffic-oblivious capacity scaling scheme

Extensions to real-world scenarios
THEOREM [MRB]: As the load $\rho \to \infty$, the number of ON servers is concentrated around $\rho + \sqrt{\rho \log \rho}$.

THEOREM [Round-Robin]: As the load $\rho \to \infty$, for constant job sizes, the number of ON servers is $\rho \left(1 + \frac{t_{wait}}{\text{job size}}\right)$.

Arrival rate $= \lambda$

avg. interarrival time

$N$ servers