


Tail Latency

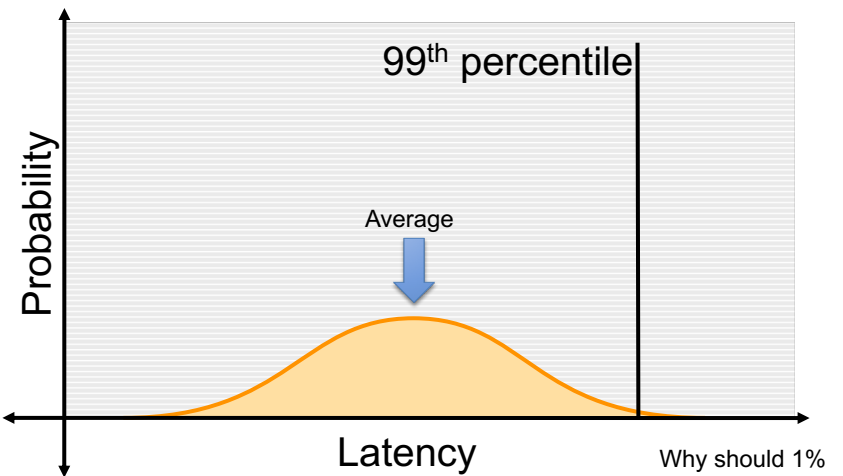
15-719 / 18-709

George Amvrosiadis
Greg Ganger
Majd Sakr

Feb 13, 2019 15-719/18-709 Adv. Cloud Computing 1



A hypothetical service time (latency) distribution



Probability

99th percentile

Average

Latency

Why should 1% slow ops matter?

Feb 13, 2019 15-719/18-709 Adv. Cloud Computing 2

Service times vary for many reasons

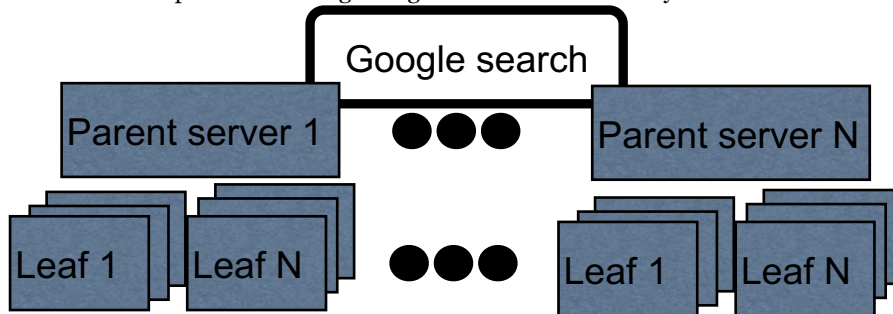
- Interference within shared infrastructure
 - Bobtail paper More VMs than cores in “cheap” EC2 instances, so easy to be allocated a VM sharing with too many compute-bound VMs
- Lots of other causes too
 - Background/maintenance activities
 - garbage collection, log compaction, virus scanning, backup, etc.
 - HPC calls this “OS jitter”
 - Dynamic and “static” hardware variations (e.g., power caps, disk heads)
 - Complex queuing and caching policies

Service time variation is a big deal in cloud services

- Very slow responses make for angry users
 - better to have them all be a little slow, than to have some very slow
 - e.g., below 100ms is plenty fast for humans
- Big jobs often wait for the last task to finish
 - so, runtime is the max task time rather than the average
 - think map-reduce, for example
 - again, better to have them all be a little slower, than to have few very slow

Tail latency & fan-out

- Matters to apps with large fan-outs of leaf tasks
 - Such fan-out parallelizes work
 - to lower latency perhaps
 - But, if app must wait until all leaves reply...
 - Example from reading: Google search works this way

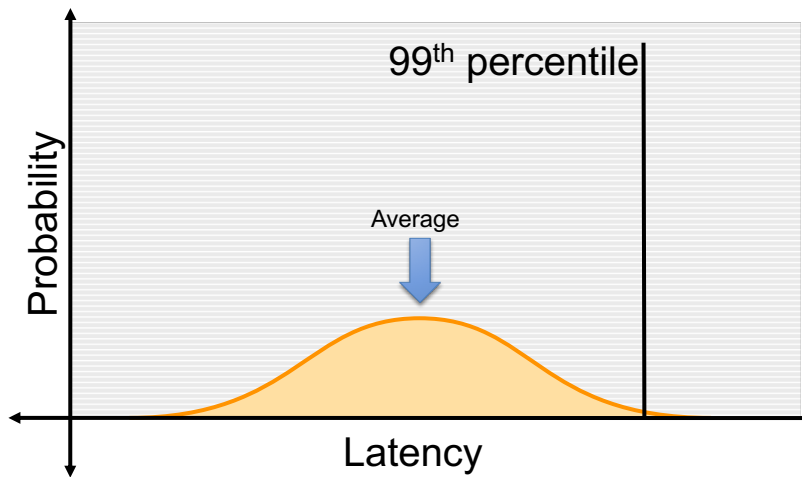


Feb 13, 2019

15-719/18-709 Adv. Cloud Computing

5

Why? Recall this graph from earlier



Feb 13, 2019

15-719/18-709 Adv. Cloud Computing

6

Tail latency & fan-out

- Consider a system with system-wide fan-out
 - Assume 100 leaf nodes
 - Assume each leaf node has 99%tile latency of 1 second
- What is the probability of any single user request taking more than than second?
 - $1 -$ probability of all 100 leaf accesses < 1 sec
 - $\sim 63\%$

Feb 13, 2019

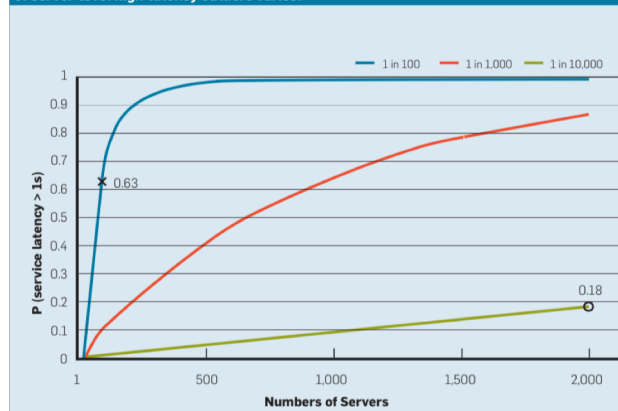
15-719/18-709 Adv. Cloud Computing

7

Bigger fan-outs suffer more

- What if only 1 in 1000 nodes see latency > 1 sec?
 - Fanout of 1000 sees $\sim 63\%$ slow requests
- At .01% slow ops fanout of 2000 sees $\sim 20\%$ slow requests

Probability of one-second service-level response time as the system scales and frequency of server-level high-latency outliers varies.



Dean, CACM13

Feb 13, 2019

15-719/18-709 Adv. Cloud Computing

8

One approach: reduce service time variation

- Great option, when it's possible
 - But, it's really difficult to do comprehensively
- Some approaches
 - Prioritize
 - do the stuff that is being waited for first (before background stuff)
 - do the stuff that is "falling behind" first
 - Manage background activities
 - synchronize schedulable maintenance stuff among machines
 - HPC deals with OS jitter this way (on global barrier sync)
 - do other background stuff when not busy with stuff

Alternate approach: "tail tolerance techniques"

- Design system assuming service time variation is inevitable
 - and do things to make tail latencies less problematic
- Some approaches
 - "hedged" requests (or "speculative" redundant requests)
 - ask more than one server to do the work (e.g., read replica, compute map)
 - take the first response to arrive and ignore the slow one
 - great for hiding infrequent slow responses, especially if 2nd request is delayed
 - "tied" requests (aggressive hedging)
 - ask more than one server immediately, but let them know you did
 - when one finishes (or starts), it "cancels" the other/redundant request
 - addresses infrequent slow responses faster with less redundant work
 - "micro-partitioning" migrate (replicate?) 5% of partitions on imbalance
 - "probation" elimination of node from datapath until its specs get better

Special app-specific “tail tolerance techniques”

- Some apps offer special opportunities or challenges
 - e.g., large information-retrieval (IR) apps like “fuzzy” search
- Positive ex.: an IR service can answer without all leaves responses
 - Why: a query displaying most possible answers is usually “good enough”
 - So, just return what is available within acceptable time limit
- Negative ex.: some queries can sometimes cause deterministic failure
 - Bugs in the system, perhaps, triggered by specific queries
 - Executing same query on all leaves causes “soft crash” outage latencies
 - “Canary requests” are one or two requests sent first to test the waters
 - If these crash, system can tolerate and this request is suppressed
 - Dean13 claims benefit of avoiding crashes worth extra round of latency

Next week

- Fault tolerance and more...

Bobtail Overview

- Paper examines RTTs in AWS EC2
 - Within a single Availability Zone (AZ) and across AZs in US East
 - Compares RTTs to 'dedicated' datacenters
- Finds that median (0.6ms/1ms) is similar
- But, finds that 99.9th percentile is ~ 2X worse
 - Good nodes: 99.9th percentile < 10ms
 - 40-50% of nodes within AZs are bad
 - Some AZs return no bad nodes...
 - Bad nodes are persistent

Feb 13, 2019

15-719/18-709 Adv. Cloud Computing

13

Root-cause analysis

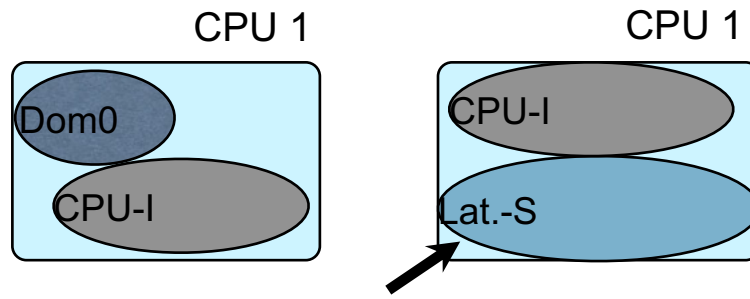
- High-tail latencies caused by...
 - Co-scheduling of latency/CPU-intensive jobs with more jobs than cores
 - Interaction with Xen (AWS hypervisor)
- Xen Details
 - Has 1 privileged VM (called *dom0*), typically pinned on 1-2 cores
 - Allows for multiple guest VMs, scheduled over remaining cores
 - Uses credit-based scheduling
 - Each VM given 30ms of credit
 - Drained in 10ms increments
 - VMs with remaining credit can be BOOSTED (run first when one VM yields)

Feb 13, 2019

15-719/18-709 Adv. Cloud Computing

14

When does problem manifest?



Latency sensitive job might have to wait an entire round to be woken up! (10ms or more)

Why doesn't BOOST help?

- When CPU-intensive jobs that take $< 100\%$ CPU can also enter BOOST queue
 - E.g., jobs that run for 28 ms in each round
- BOOST queue serviced in FIFO order