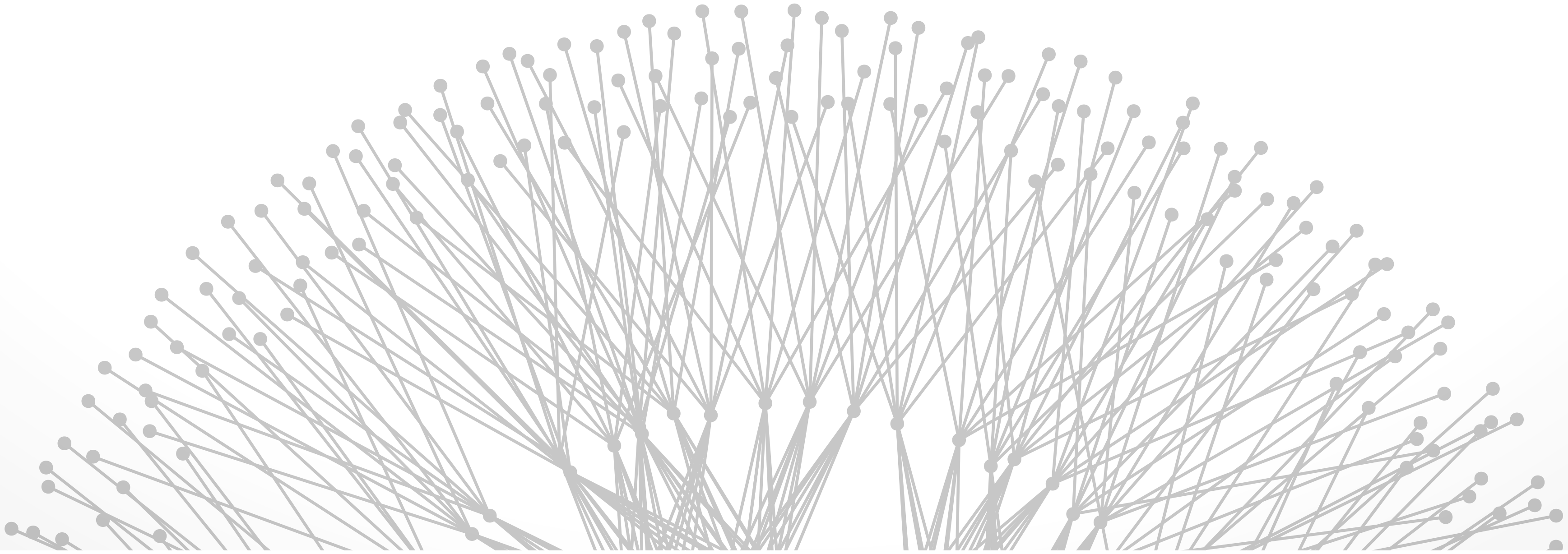
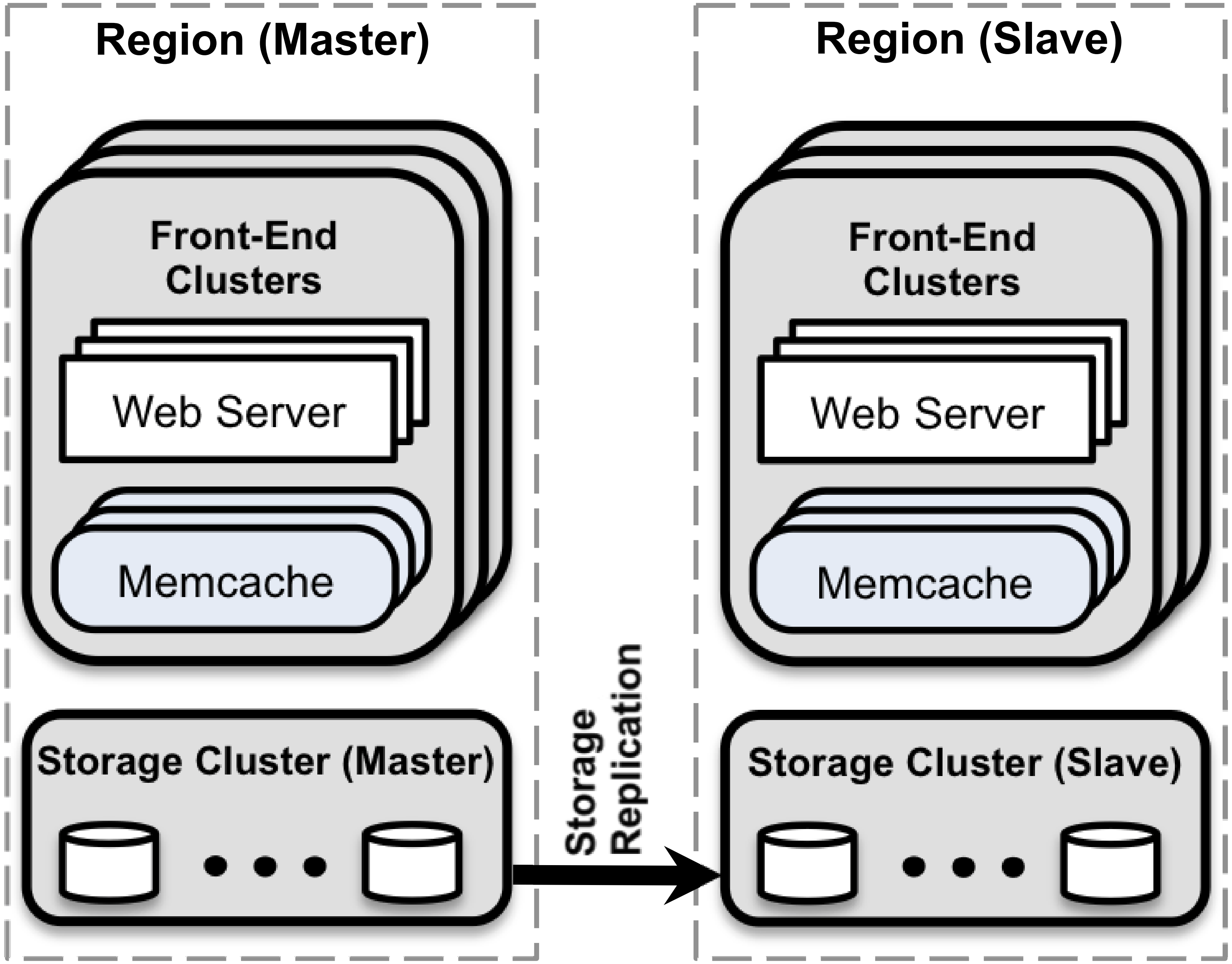
Cloud Services

Brighten Godfrey
CS 538 December 3 2013



Cloud service architecture



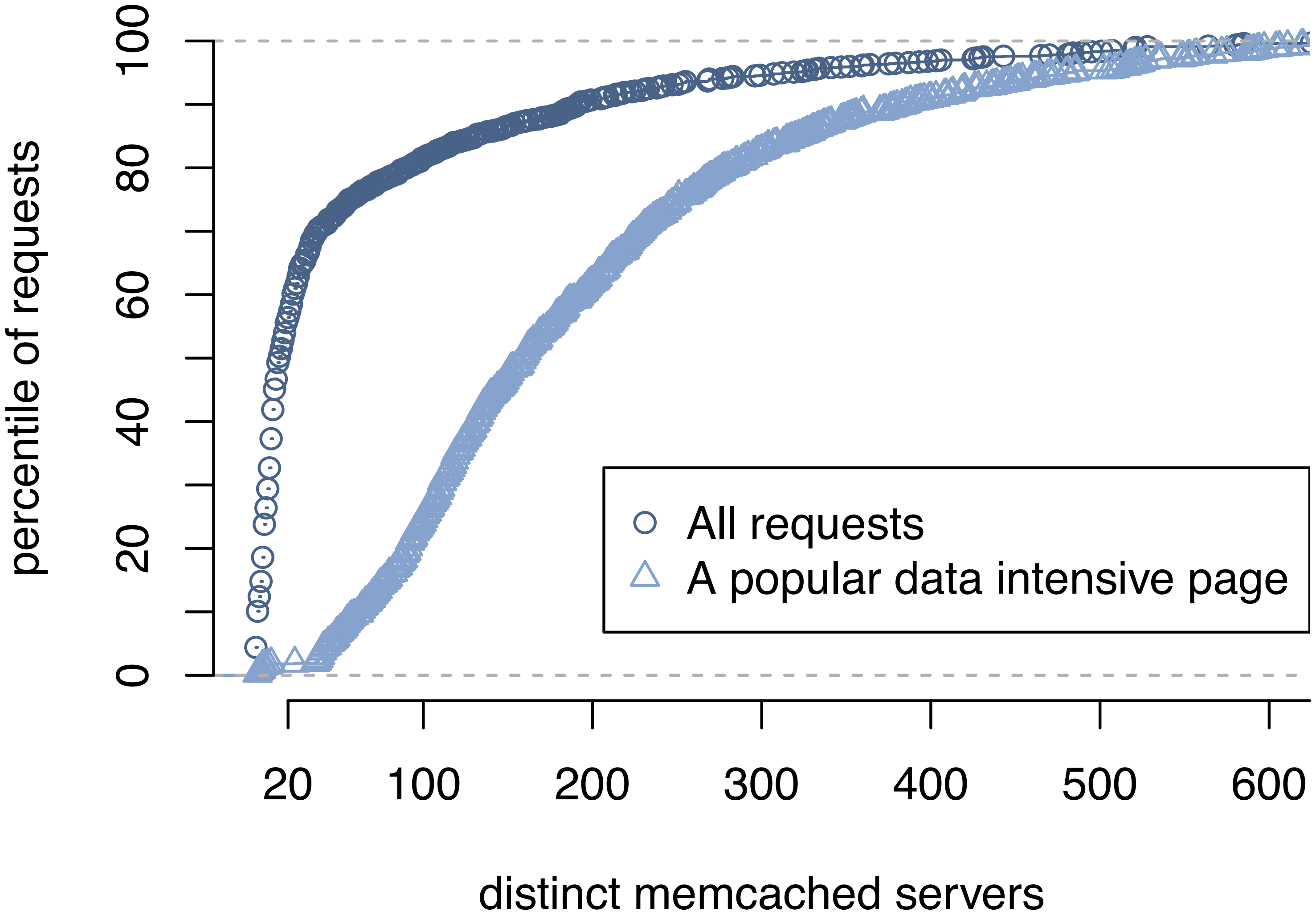
[from Nishtala et al., Scaling Memcache at Facebook, NSDI 2013]

Cloud service characteristics

O(billions) scale

Wide “fan-out”

- 100s of memcached servers per request
- Causes all-to-all traffic from web to memcached servers



(from web server for single web request)

[from Nishtala et al., Scaling Memcache at Facebook, NSDI 2013]

Cloud service characteristics

O(billions) scale

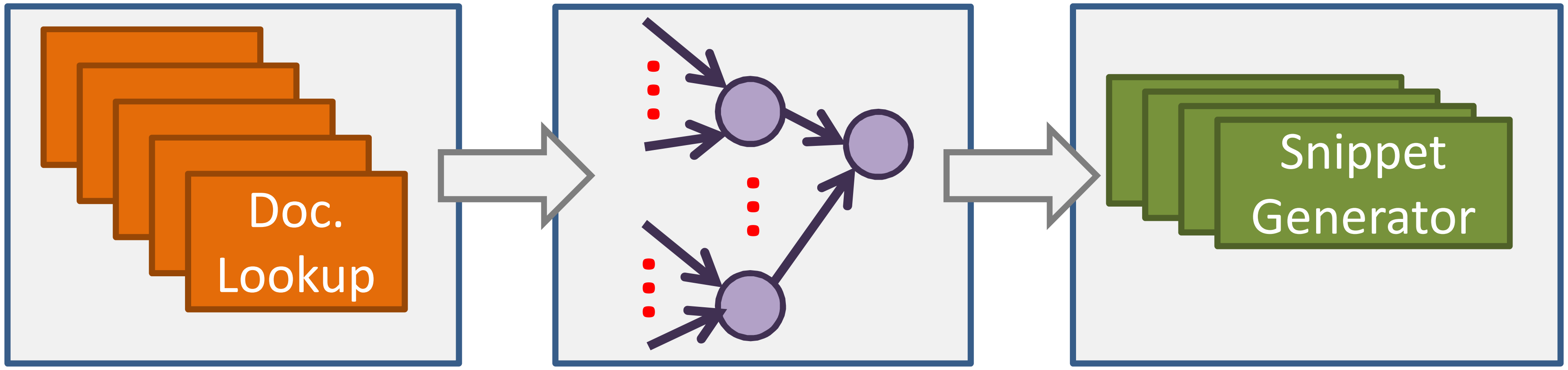
App workflows have wide "fan-out"

- 100s of memcached servers per request
- Causes all-to-all traffic from web to memcached servers

App workflows need multiple rounds per request

- Service tasks according to the DAG of dependencies
- Example of needing multiple rounds?

Simplified Bing workflow



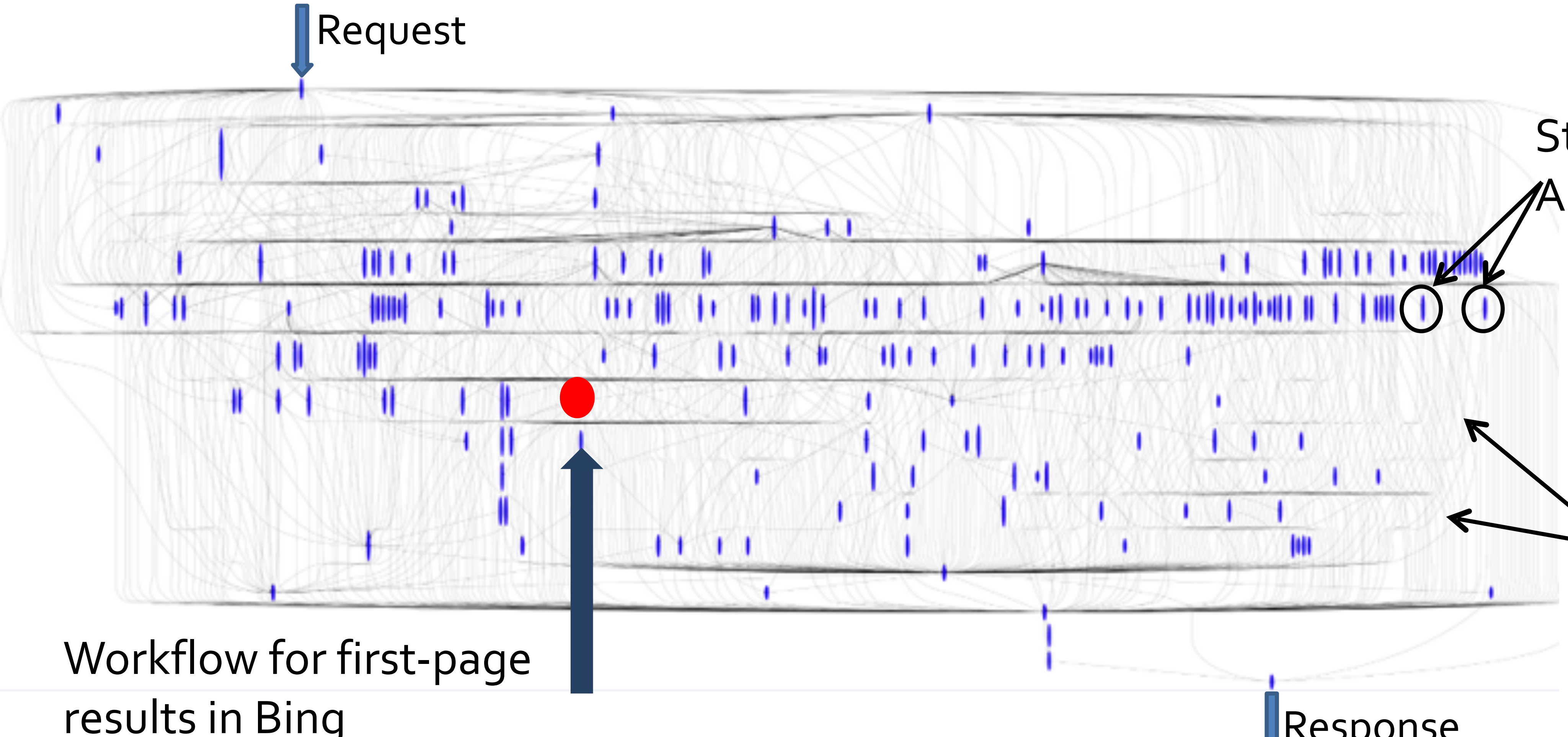
Looking up the index for a document

(network) Lag to collect responses

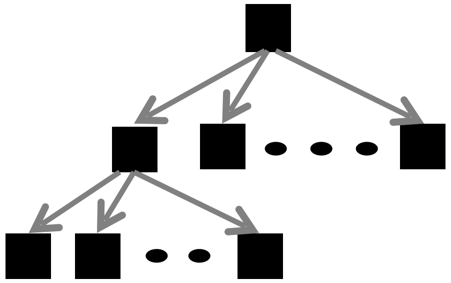
Snippet generation

[from Jalaparti, Bodik, Kandula, Menache, Rybalkin, Yan, SIGCOMM 2013]

Web services implemented as complex Workflows



Workflow for first-page results in Bing



Partition-aggregate

- Up to 150 stages, degree of 40, path lengths of 10 or more
- Each stage can be a complex workflow

Stochastic delays accumulate across stages

[Slide from Jalaparti, Bodik, Kandula, Menache, Rybalkin, Yan, SIGCOMM 2013]

Cloud service characteristics

O(billions) scale

Wide “fan-out”

- 100s of memcached servers per request
- Causes all-to-all traffic from web to memcached servers

Needs multiple rounds per request

- Service tasks according to the DAG of dependencies
- Example of needing multiple rounds?

Implications

- Need extreme performance
- Exceptional conditions become the common case

Key design goal: scale to billions

A cornucopia of systems optimizations

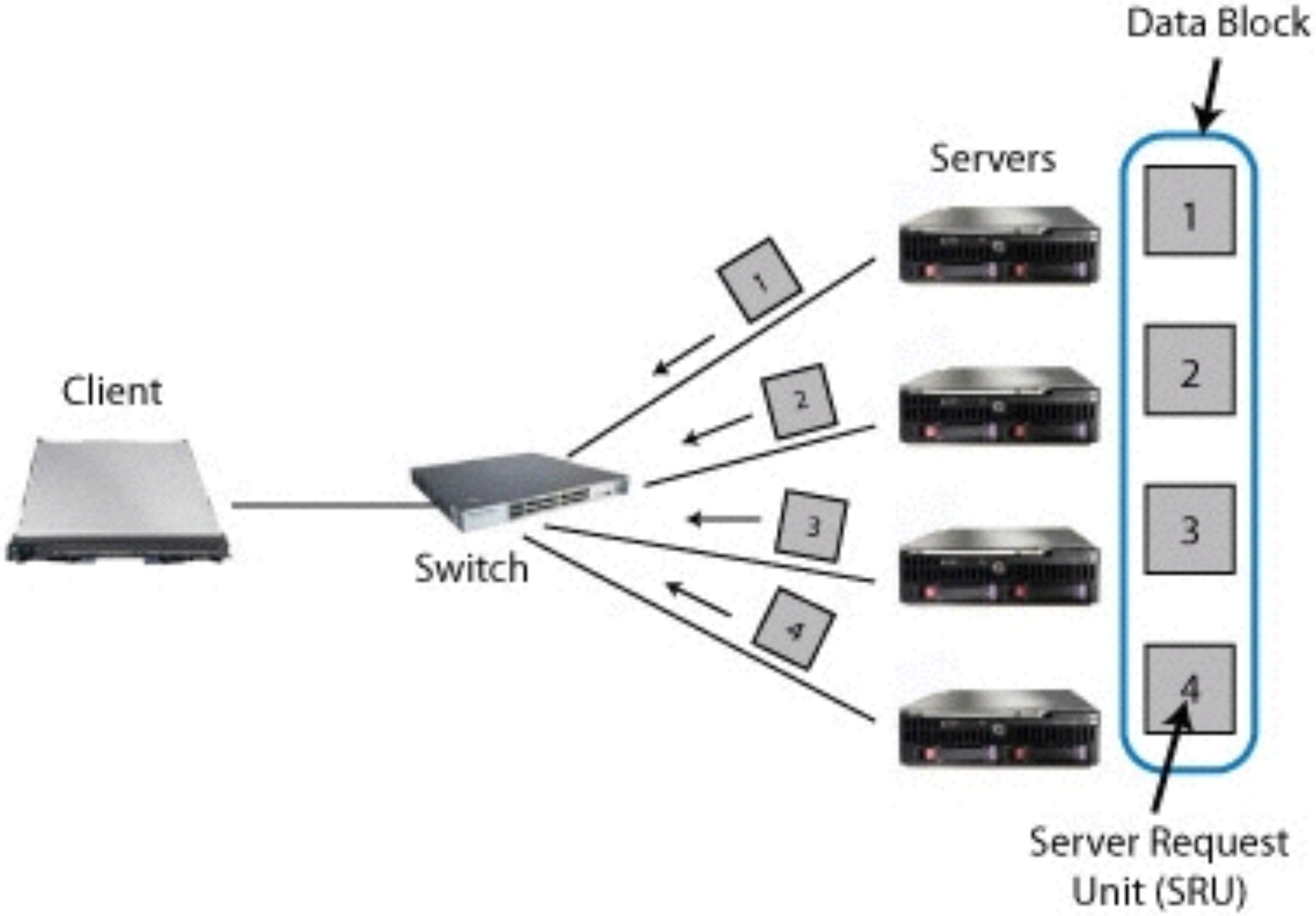
- Aggregate queries across threads, compression, batching requests in one packet, custom malloc, use UDP, client flow control to avoid incast, …
- One master region handles writes, others read-only

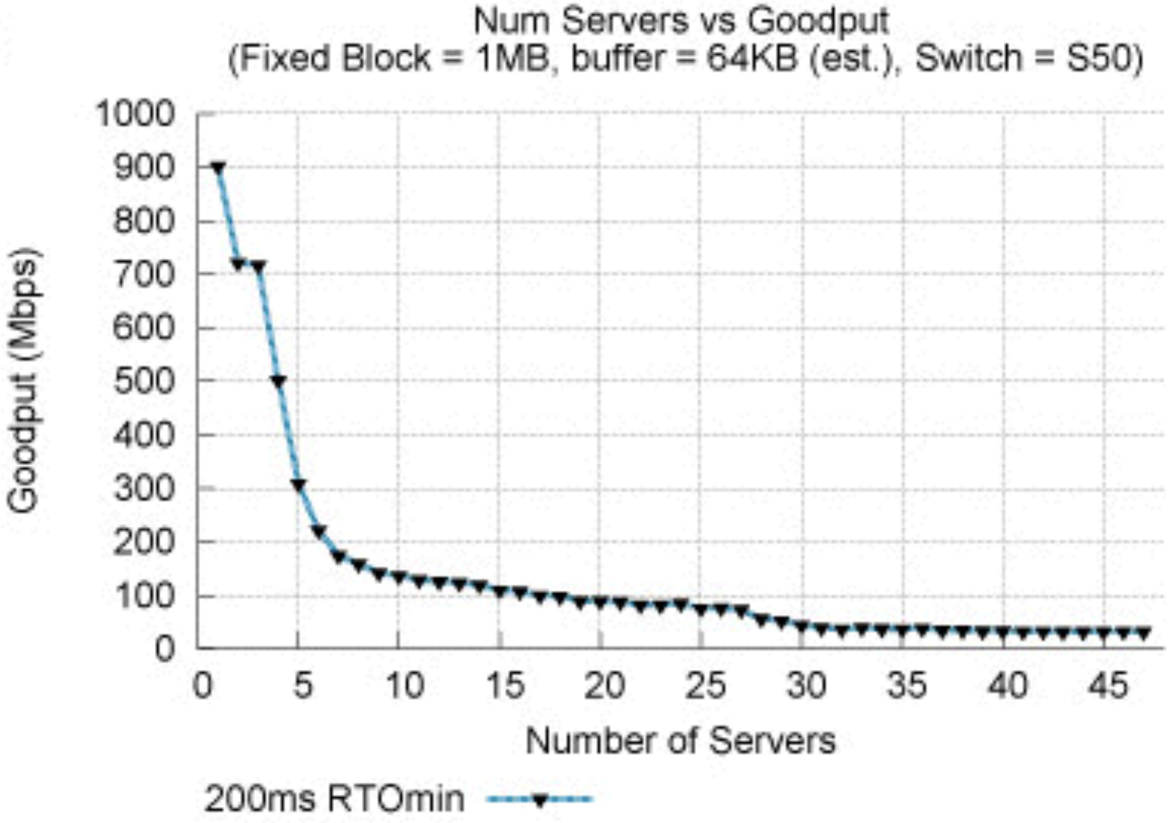
Keep memcache servers simple

- Only talk to web clients
- Web clients handle complexity (e.g., installing cached values, carrying tokens, error recovery)

Pr[stale] is tunable, not a correctness problem

Aside: what's TCP incast?





Figures from CMU PDL INCAST project:
http://www.pdl.cmu.edu/Incast/

Interesting observations

Warmup takes hours! (How did they handle this?)

- Bring up new cluster fast by moving content from already-warm memcache cluster
- memcached servers store cached values semi-persistently
  - in shared memory region
  - doesn't die when memcached process is killed or upgraded!

Intriguing questions

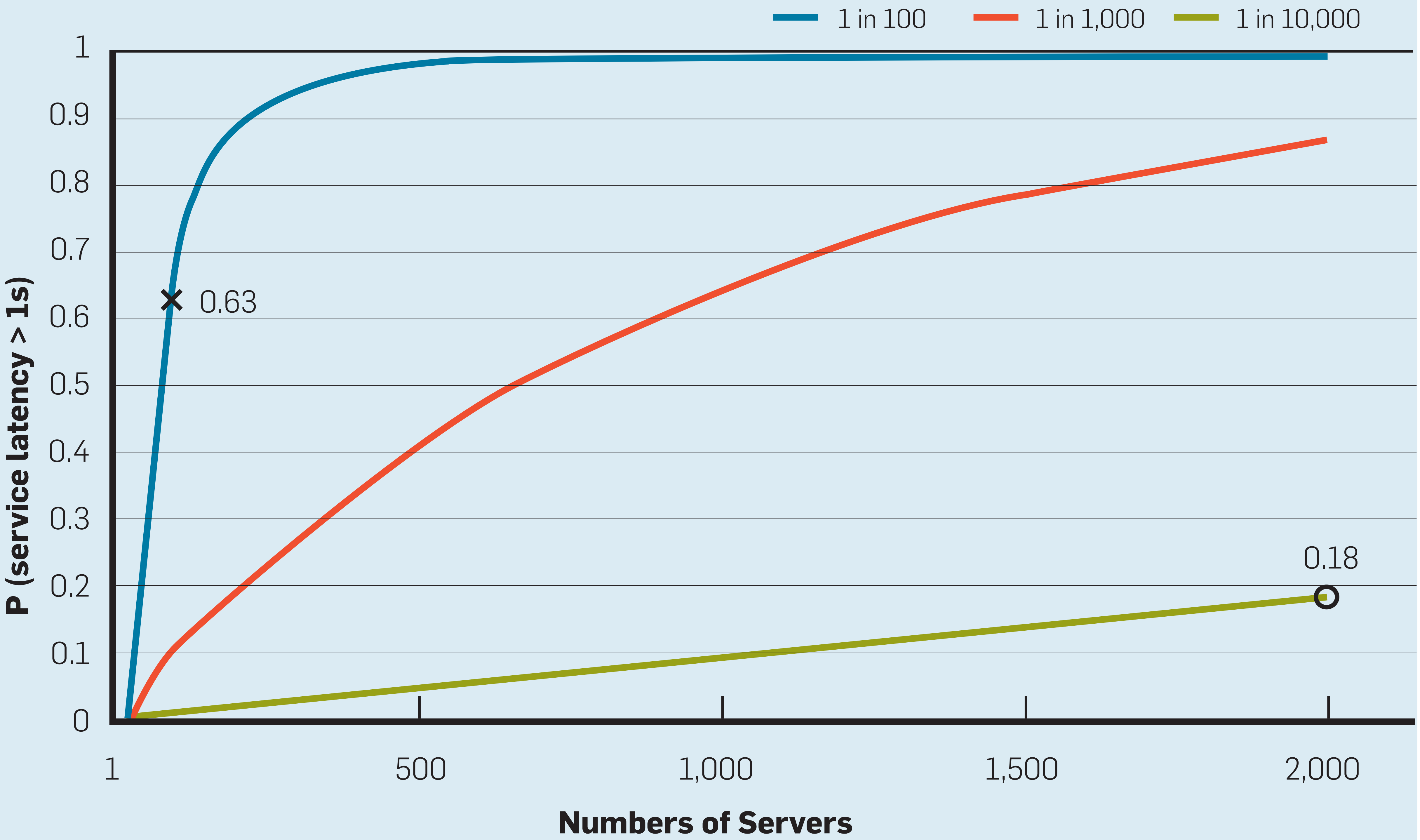
- What would happen if you shut off Facebook and turned it back on again?
- What if you shut off the Internet and turned it back on again?

“Tail at Scale” [Dean and Barroso]

Key problem identified?

- Exceptional conditions become the common case

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



[Dean and Barroso, CACM 2013]

“Tail at Scale” [Dean and Barroso]

Key problem identified:

- Exceptional conditions become the common case

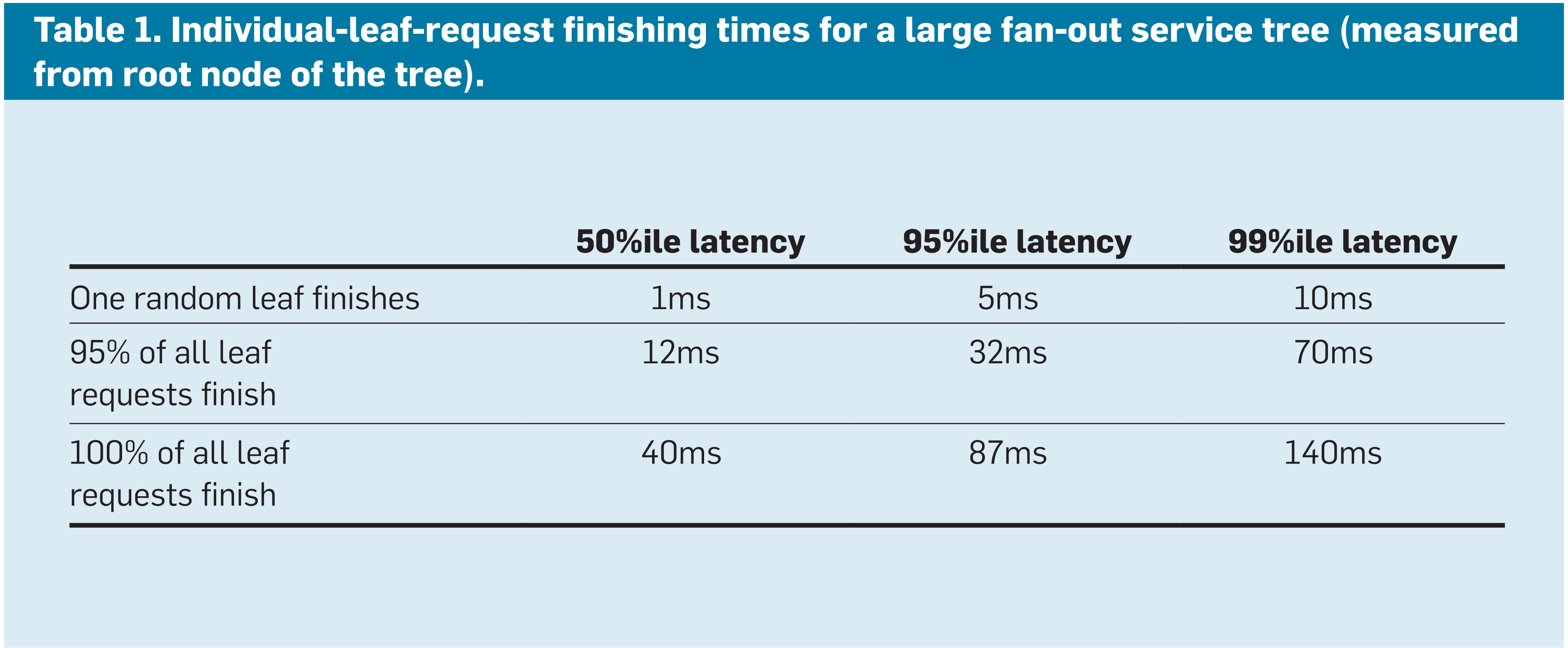


Table 1. Individual-leaf-request finishing times for a large fan-out service tree (measured from root node of the tree).

	50%ile latency	95%ile latency	99%ile latency
One random leaf finishes	1ms	5ms	10ms
95% of all leaf requests finish	12ms	32ms	70ms
100% of all leaf requests finish	40ms	87ms	140ms

[Dean and Barroso, CACM 2013]

Discussion question

How do these two papers approach replication?

- Google's "tail at scale"
- Facebook's scaled memcached

Discussion question

How do these two papers approach replication?

- Facebook's scaled memcached
  - Goal: scaling efficiently
  - Data in memory => minimize replicated data to maximize cache size
  - Replication is used to increase throughput

- Google's "tail at scale"
  - Goal: consistent performance
  - Data replicated for reliability, enabling …
  - …replicated requests ("hedged")
  - …replicated requests with cancellation ("tied")

Announcements

Assignment 2

- Solution key posted
- Feedback will be emailed to you this week

Thursday

- Composing SDNs [Monsanto et al., NSDI 2013]

Next Tuesday

- Presentations by those who can't make poster session
- Ping us to schedule
- Online students: we'll be in touch