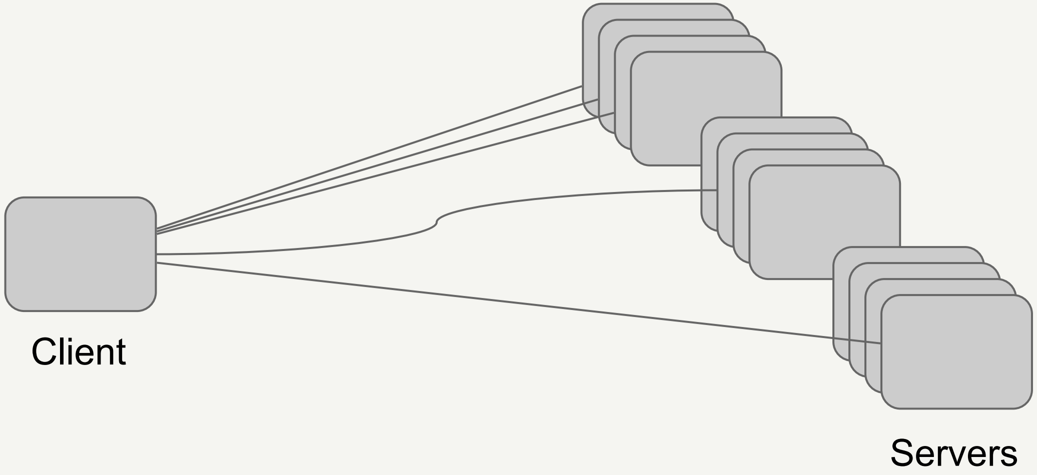
Predictive Load balancing: Unfair but Faster & more Robust

An interactive version (with in browser demos) of this presentation is available at:
https://storage.googleapis.com/strangeloop2017/slide0.html

Steve Gury
@stevegury
Sept. 30th 2017

NETFLIX

What is load-balancing?



Problem: I have a set of servers, but **which one** do I talk to?

Demo 1

Why do we care?

Is there an impact on the **performance** of the system?

Demo 2

Load-balancing implies “balancing”

Multiple strategies:

- Ensuring that over time, each server receives the same number of requests

Load-balancing implies “balancing”

Multiple strategies:

- Ensuring that over time, each server receives the same number of requests
- Ensuring that at any time each server processes the same number of requests

Load-balancing implies “balancing”

Multiple strategies:

- Ensuring that over time, each server receives the same number of requests
- Ensuring that at any time each server processes the same number of requests
- Minimizing some utility function at the client side (e.g. latency)

Trouble #1

Not all servers are always the same

Demo 3

Trouble #2

Thundering herd: we don’t want all clients to target a specific server

Demo 4

Trouble #3

We want to be resilient in presence of outliers

Demo 5

Trouble #4

A large cluster of servers **dilute** local states kept by clients

Demo 6

Load-Balancing Matrix

	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗

Load-Balancing Matrix

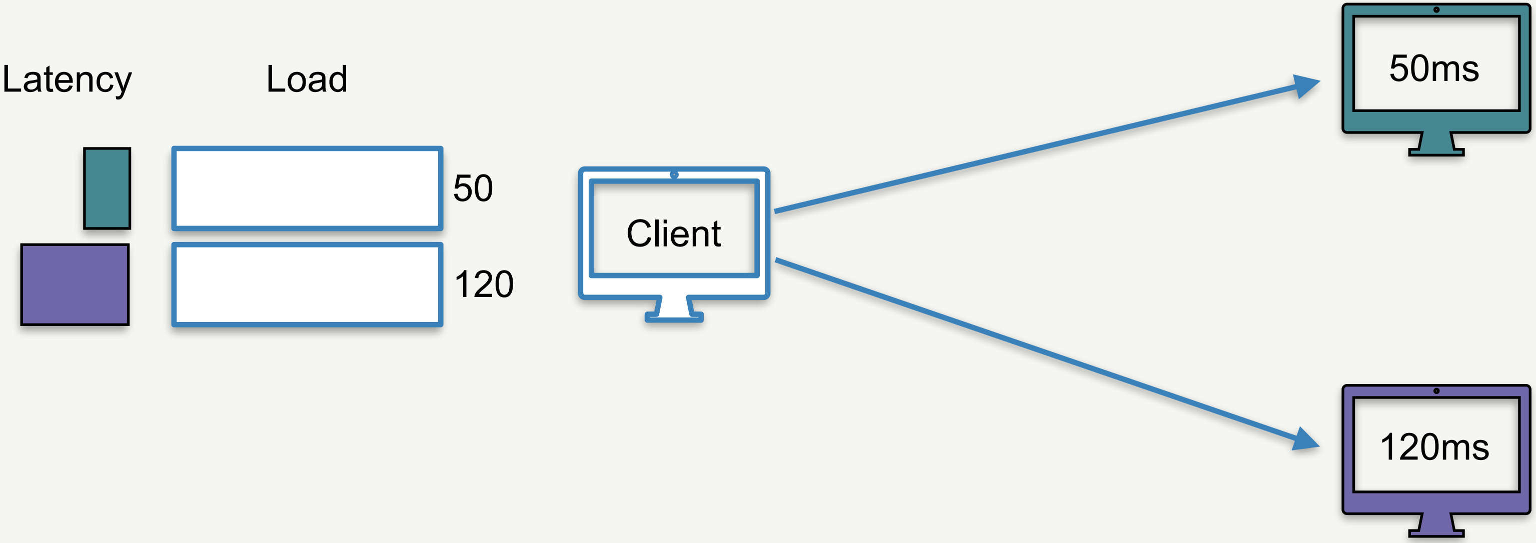
	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗
?	✓	✓	✓	✓

Latency based Load-Balancing

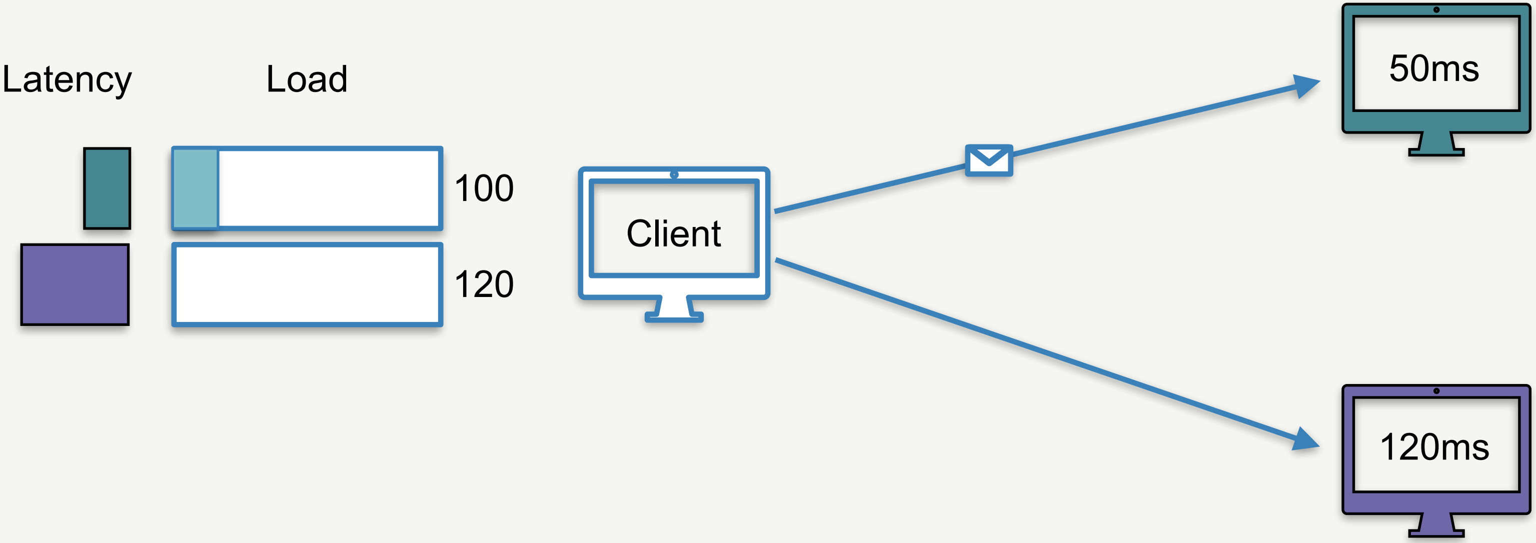
Idea: Use the latency of the server as a measure of the load

```
Load = Predicted Latency * (#requests + 1)
```

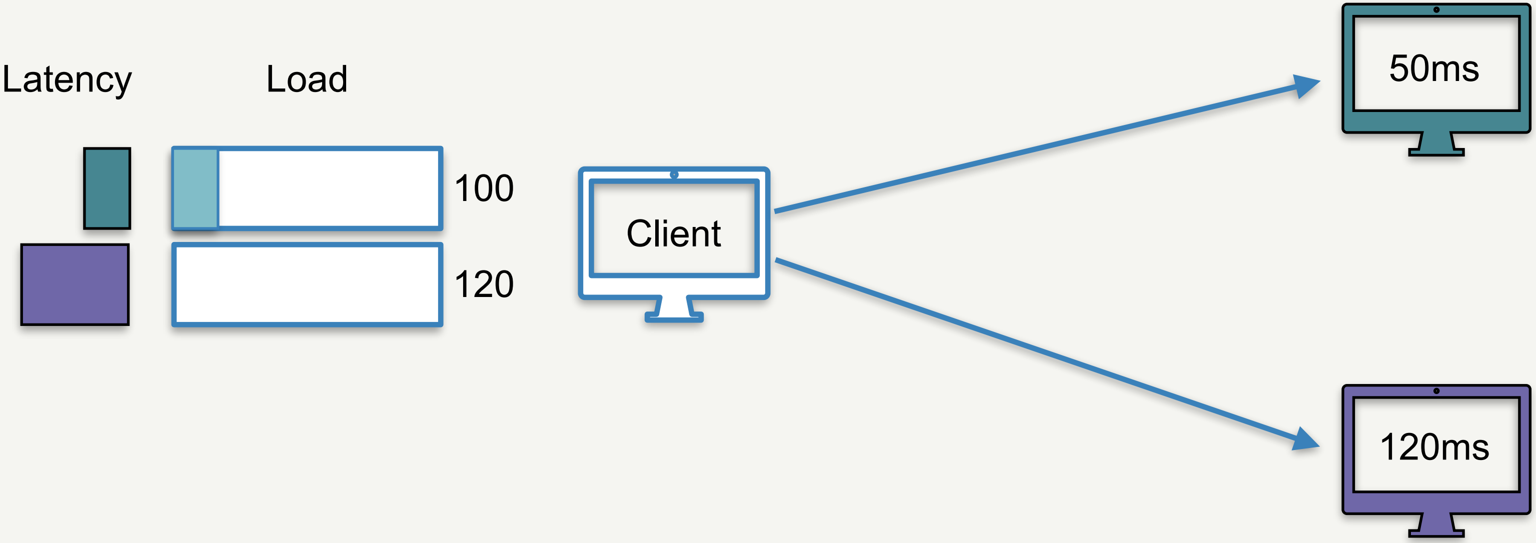
Latency based Load-Balancing



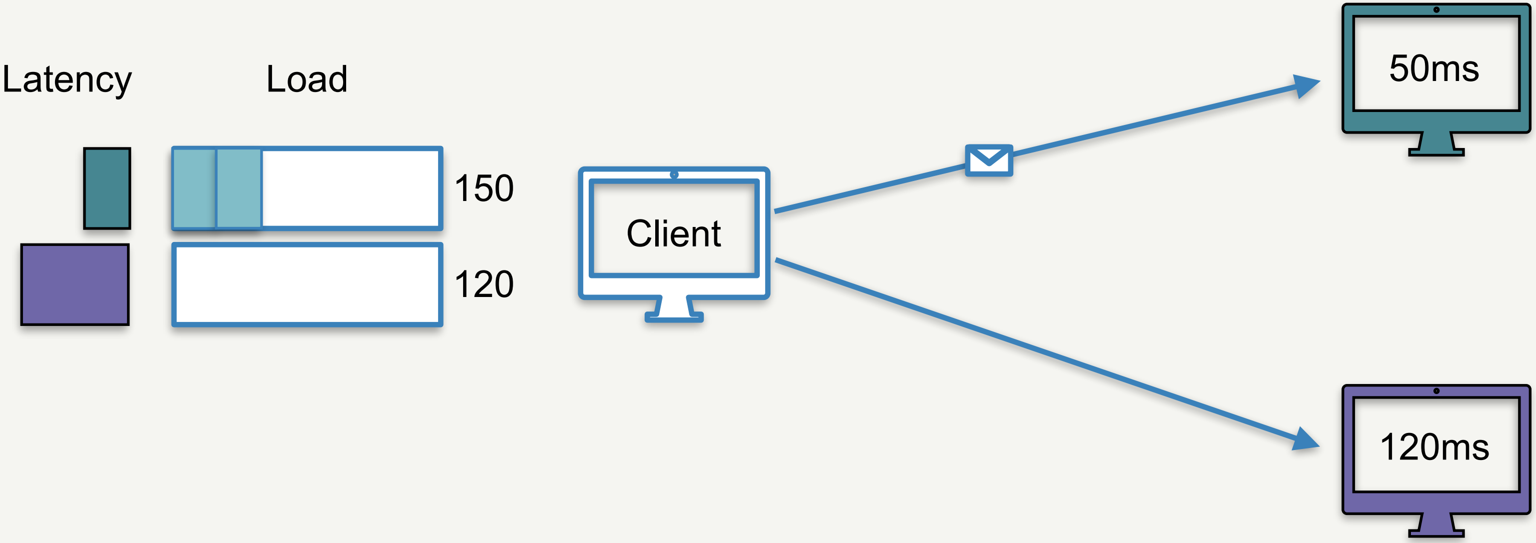
Latency based Load-Balancing



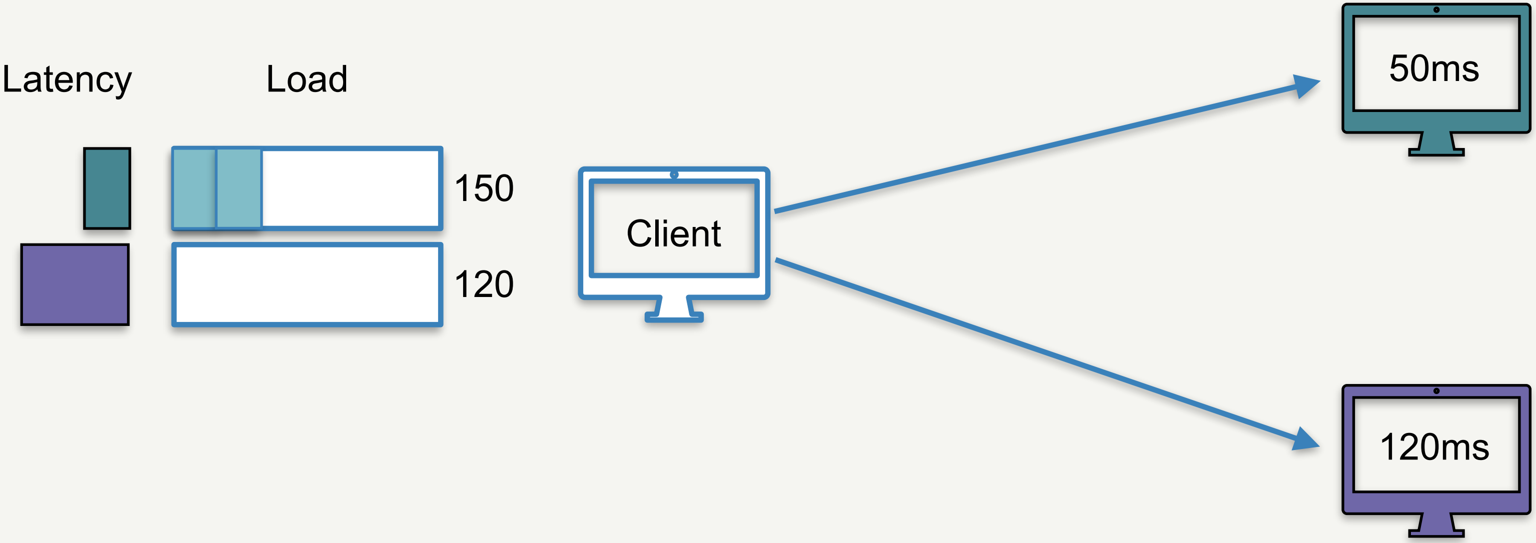
Latency based Load-Balancing



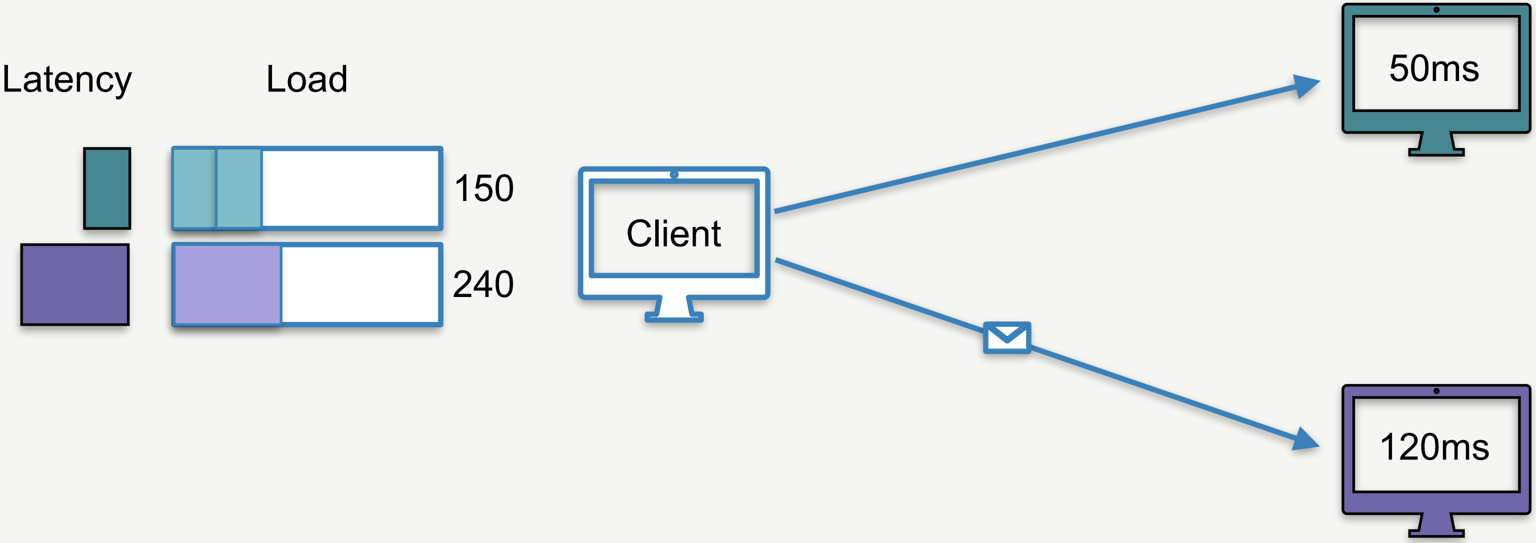
Latency based Load-Balancing



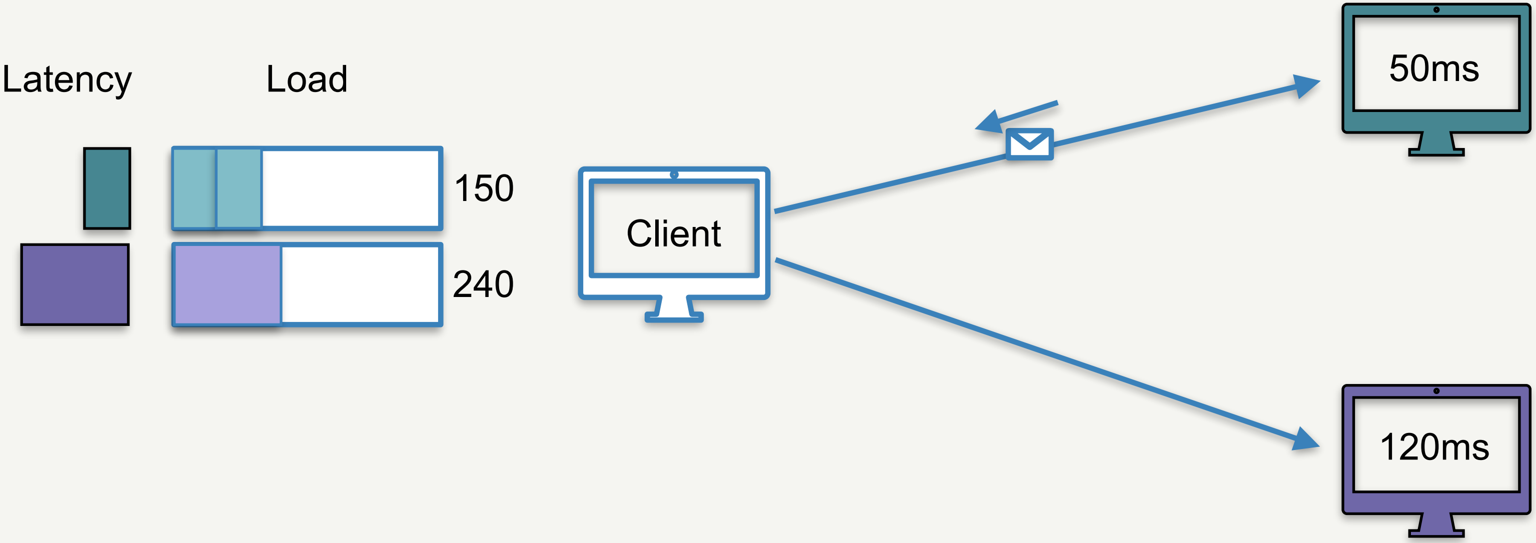
Latency based Load-Balancing



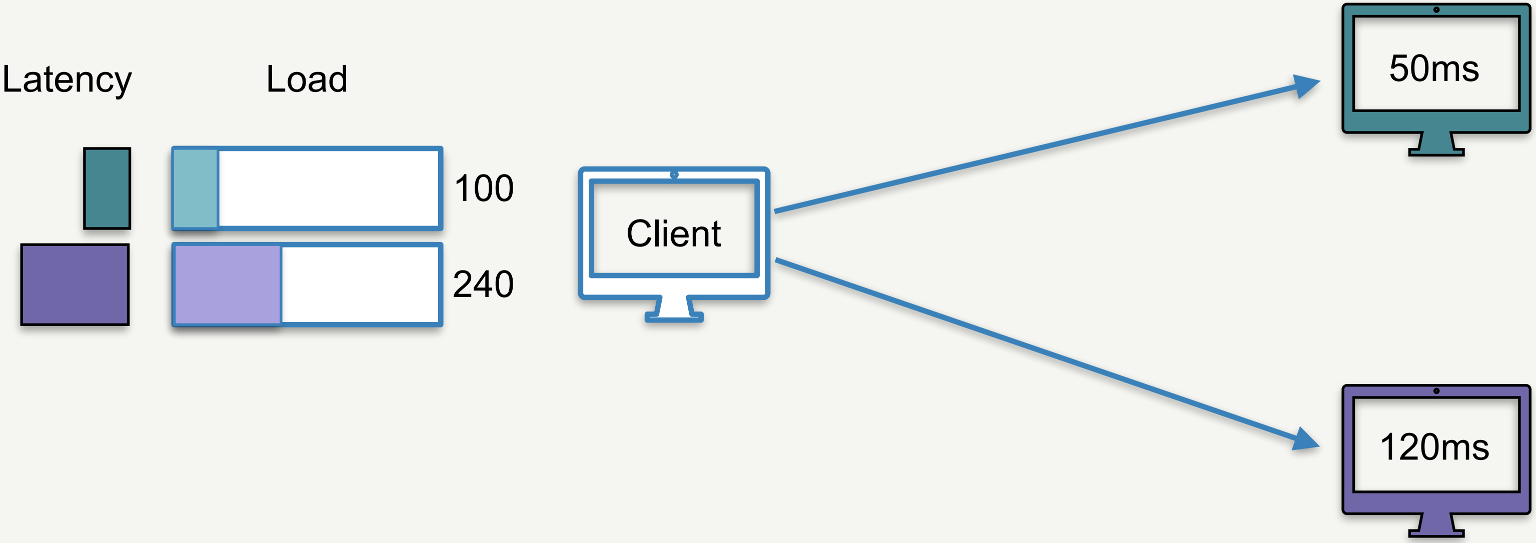
Latency based Load-Balancing



Latency based Load-Balancing



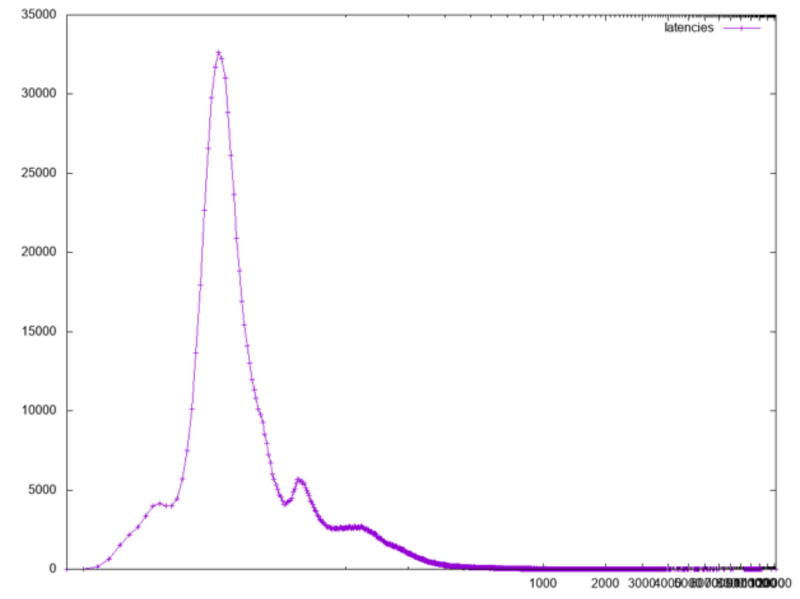
Latency based Load-Balancing



Load-Balancing Matrix

	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗
Predictive LoadBalancing	✓			

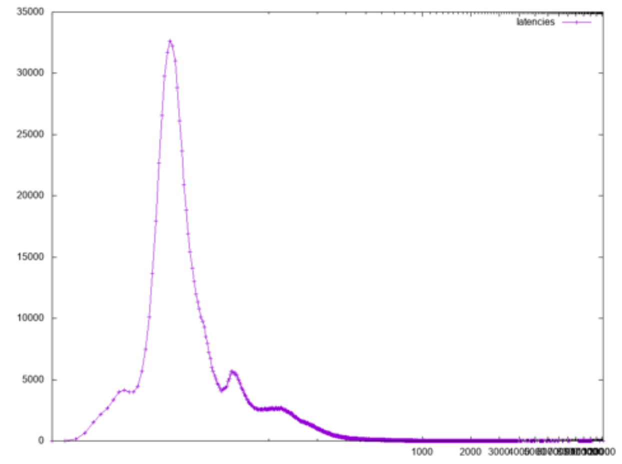
How to measure the latency?



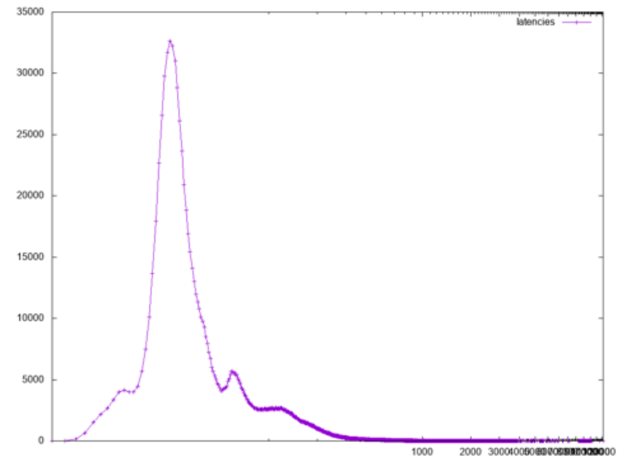
We need to extract one number from the latency distribution

How to measure the latency?

In steady state, most servers have the same latency characteristics

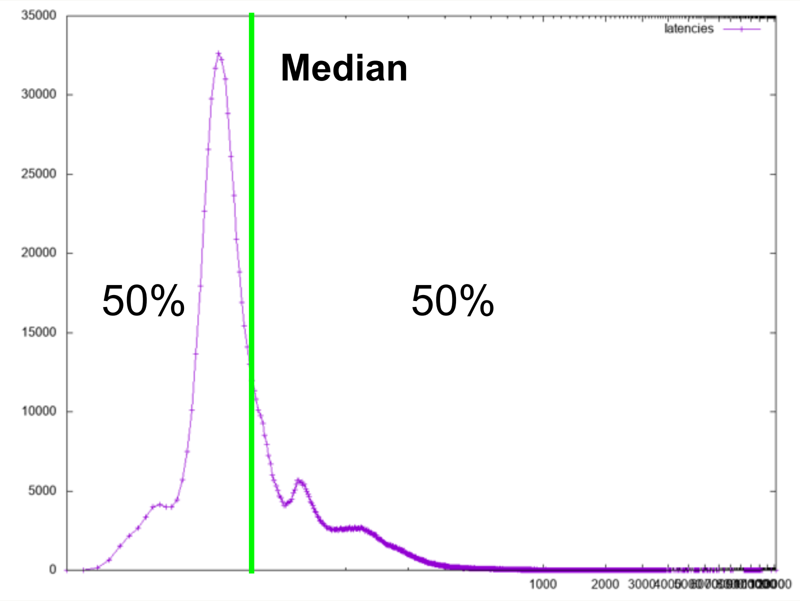


Server A



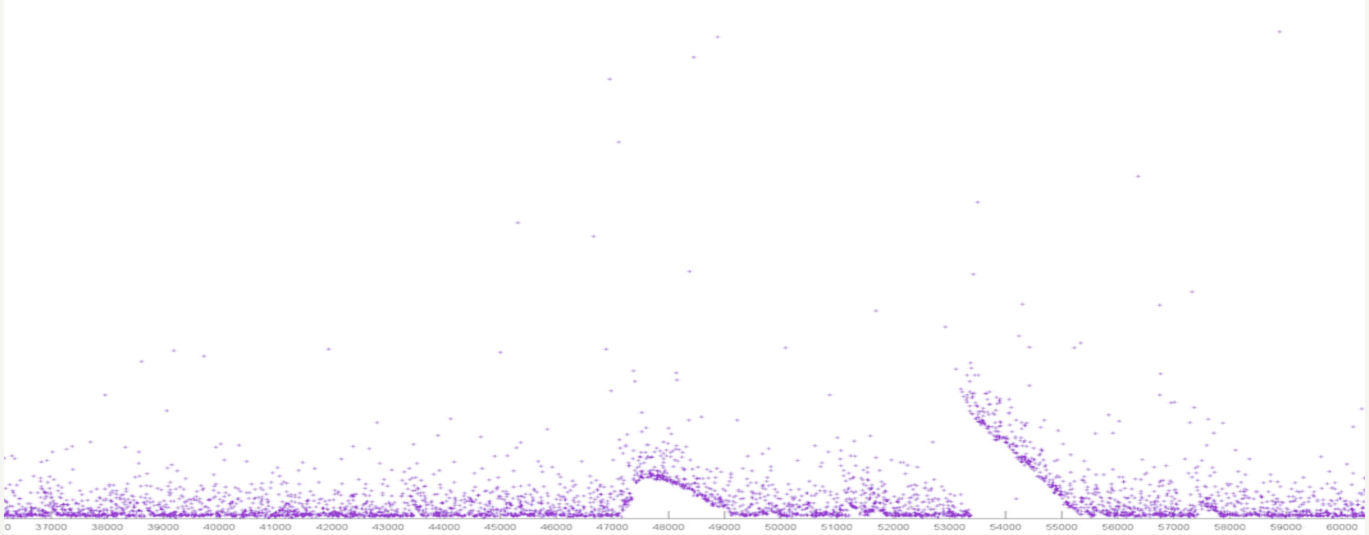
Server B

How to measure the latency?



Computing a Streaming Median gives us a stable latency prediction

Recency is important



We want to quickly update our estimation when the values are changing

Streaming Median Latency is what we measure as the load we sent to a server.

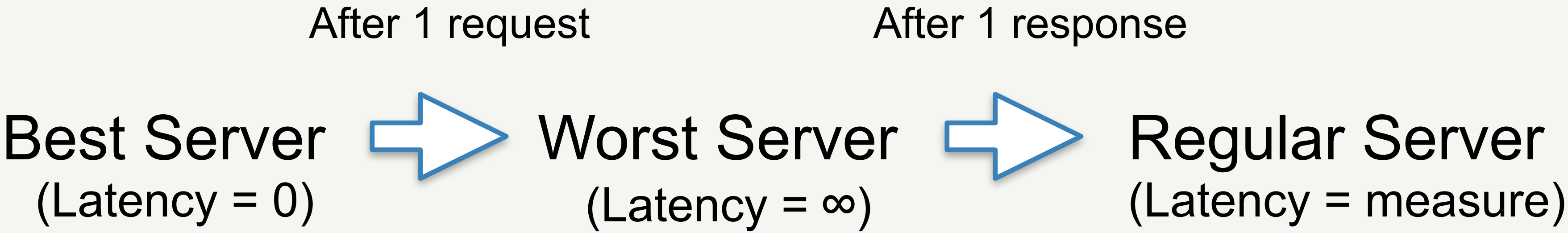
It is a stable value and represent recent data

Demo 7

Devil is in the details

- How to estimate latency of new servers (no data)?
- What if latency estimation is off?
- What about fast server erroring a lot?

How to estimate latency of new servers (no data)?



After 1 request

After 1 response

Best Server
(Latency = 0)

Worst Server
(Latency = ∞)

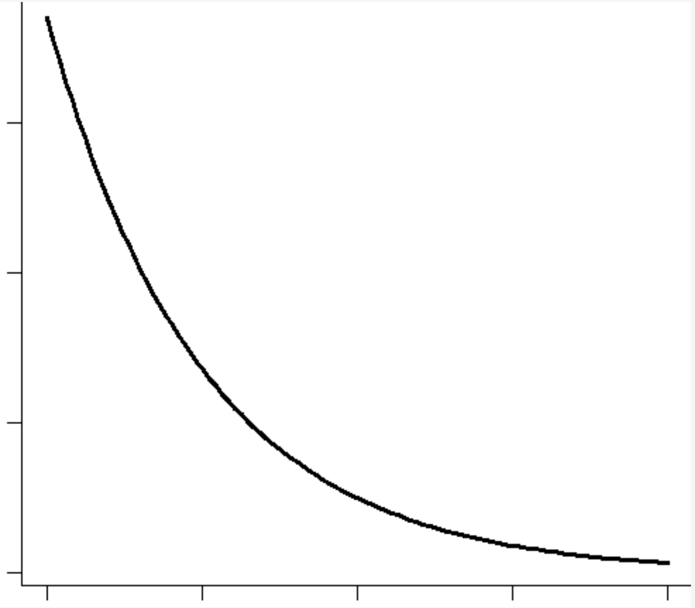
Regular Server
(Latency = measure)

Load-Balancing Matrix

	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗
Predictive LoadBalancing	✓	✓		

What if latency estimation is off?

If we don’t talk to a server for a long time, its latency prediction decays.

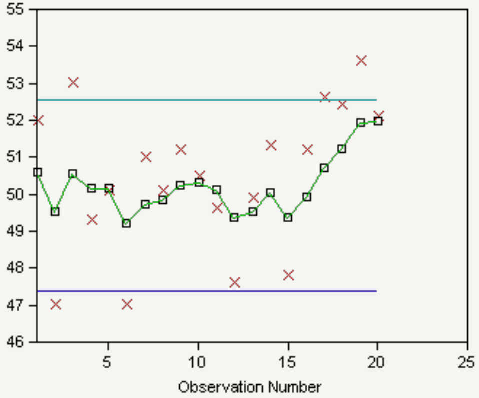


Fast server but Bad server

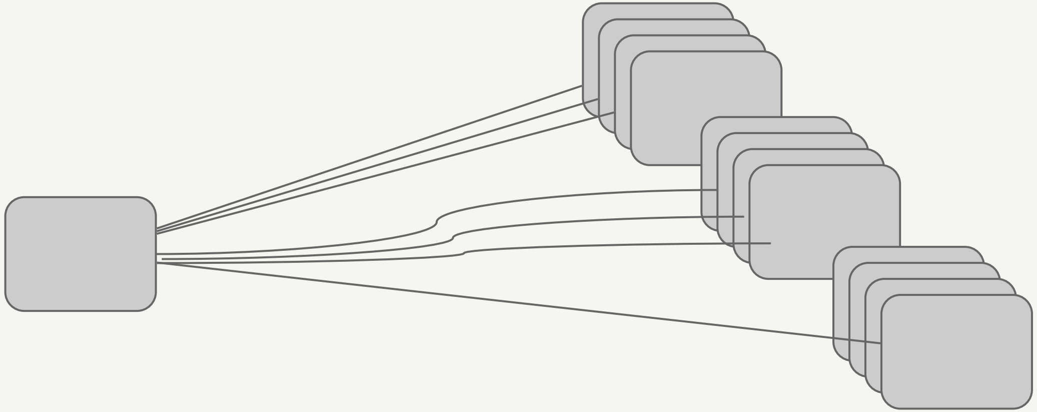
We only measure latency of successful responses

Compute a sliding success rate (EWMA)

Use this success rate to shape the traffic linearly
E.g. a server consistently sending 20% of errors will have its latency penalized by 20% (linear)

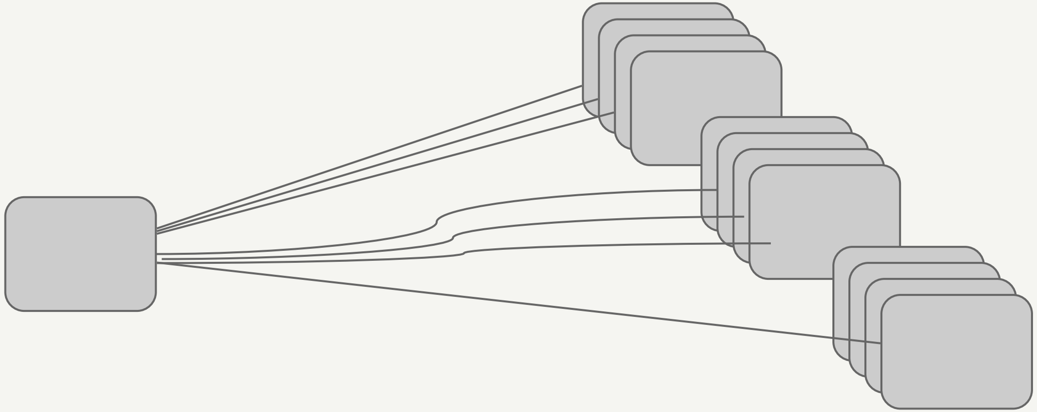


Keeping fresh statistics



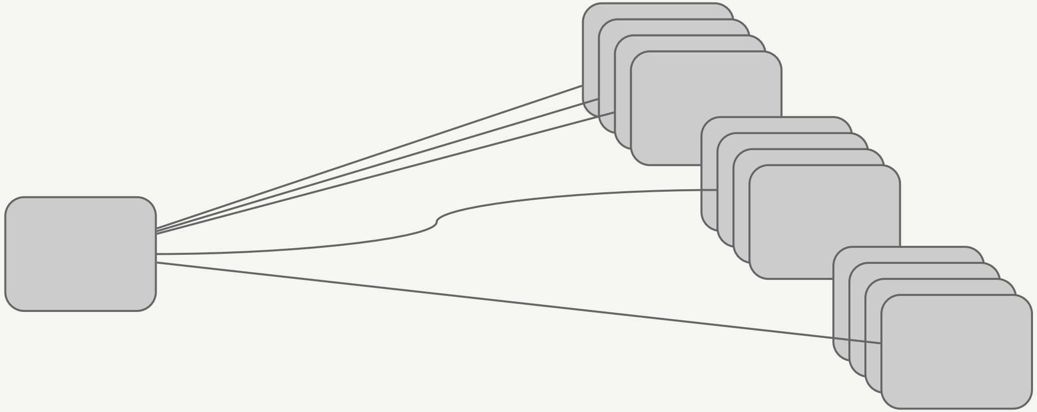
Too many connections = stale statistics

Keeping fresh statistics



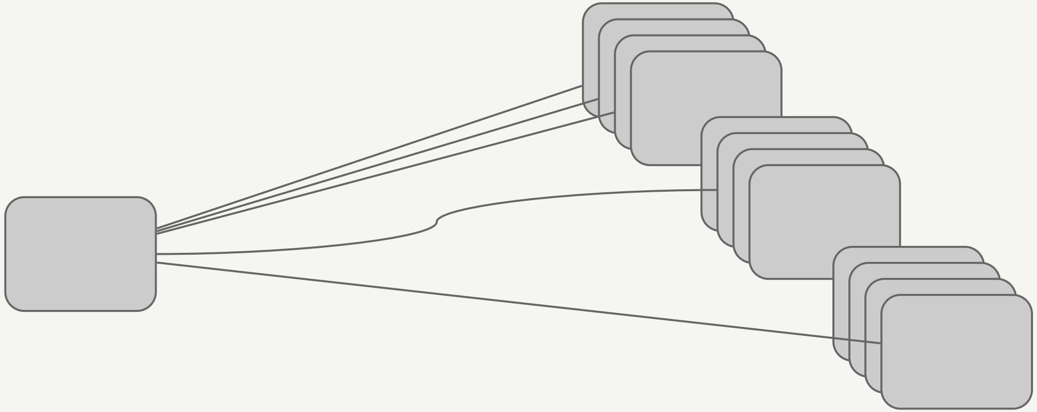
10k servers at 1k RPS = 1 requests every 10 sec / server

Keeping fresh statistics



Size the number of connection needed based on avg #outstandings

Keeping fresh statistics



Regularly evict the slowest server, and pick another one randomly

Load-Balancing Matrix

	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗
Predictive LoadBalancing	✓	✓		✓

All those statistics are fine but we need at least one slow response to realize that a server is slow.

How to react quickly

Agner Erlang Founding father of Queuing Theory.

Concept of Instantaneous Traffic: how much load have been offered at an instantaneous time.

Agner Erlang

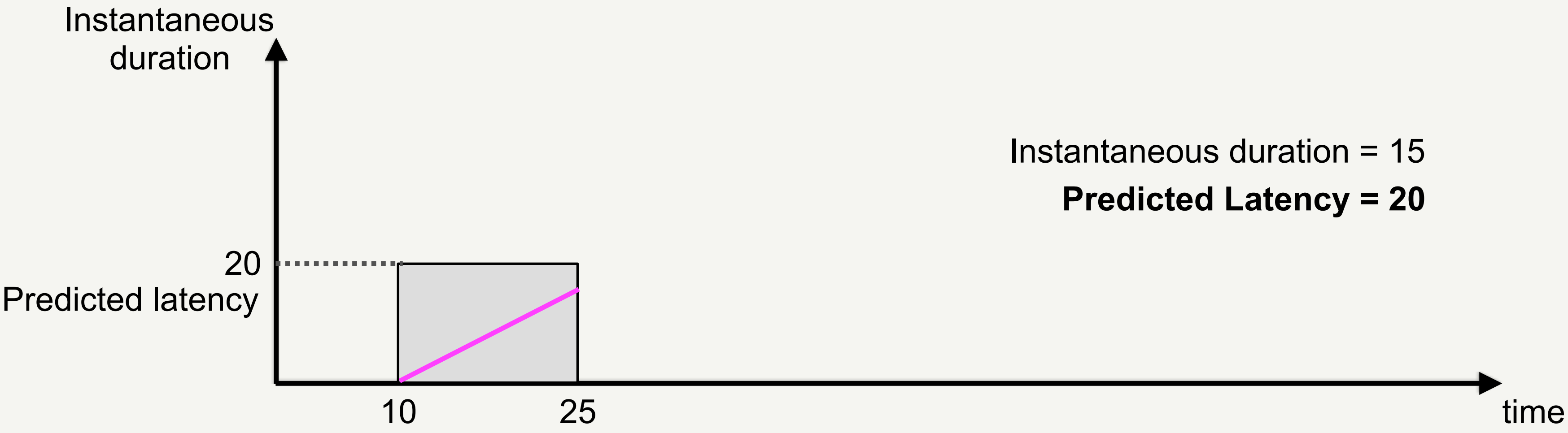
Instantaneous duration

Idea: Use the elapsed time (per request) as the value of the load offered to a server

Instantaneous duration = 0

Instantaneous duration

Idea: Use the elapsed time (per request) as the value of the load offered to a server

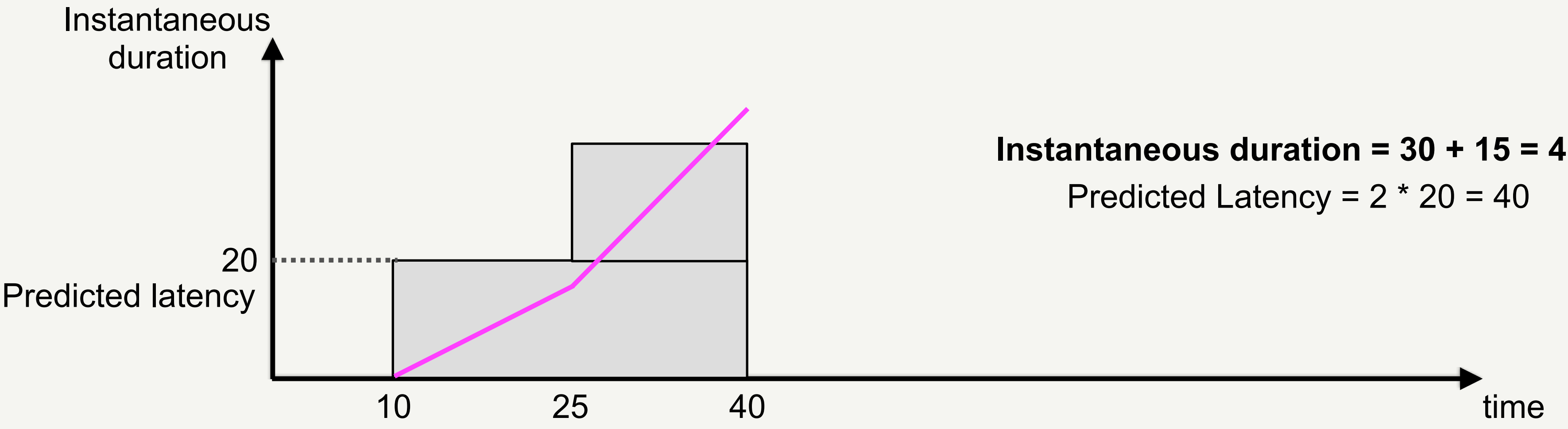


Instantaneous duration = 15

Predicted Latency = 20

Instantaneous duration

Idea: Use the elapsed time (per request) as the value of the load offered to a server

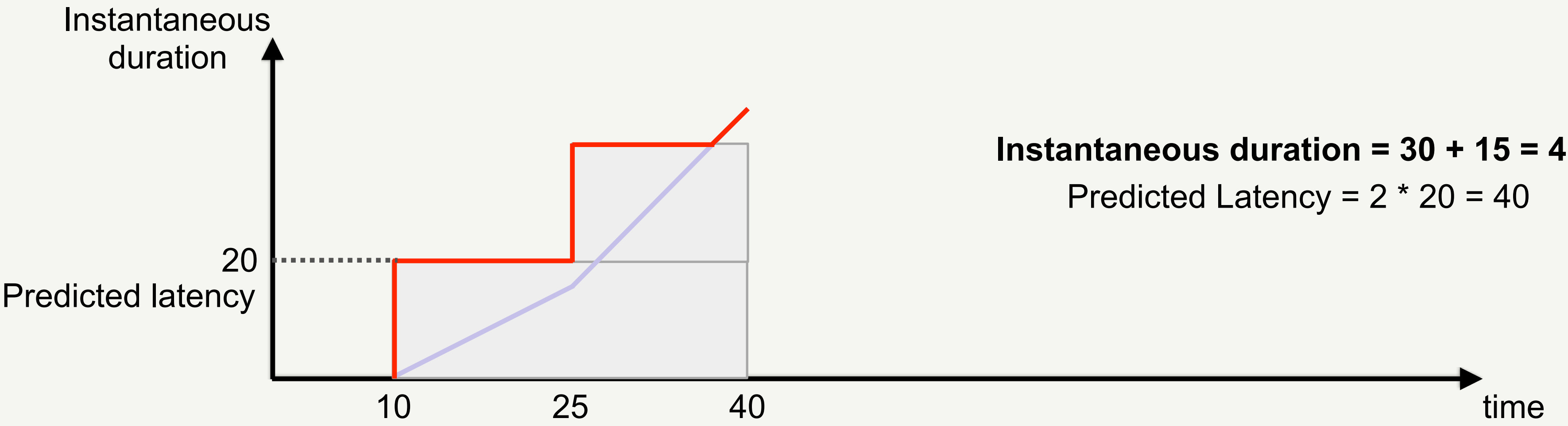


Instantaneous duration = 30 + 15 = 45

Predicted Latency = 2 * 20 = 40

Instantaneous duration

Load = Max(predicted latency, instantaneous duration)



Instantaneous duration = 30 + 15 = 45

Predicted Latency = 2 * 20 = 40

Instantaneous duration

- Great for sudden event (GC pause, network partition)
- Need an average of one median latency to detect a dead server.

Load-Balancing Matrix

	Uneven servers	Thundering Herd	Outliers	Large cluster
Random	✗	✓	✗	✓
Round Robin	✗	✓	✗	✗
Least Loaded	✓	✗	✓	✗
Predictive LoadBalancing	✓	✓	✓	✓

Demo 8

Nothing is perfect

- Latency not always a proxy for resources consumption

Nothing is perfect

- Latency not always a proxy for resources consumption
- Slow warmup of cold servers

Nothing is perfect

- Latency not always a proxy for resources consumption
- Slow warmup of cold servers
- Defeat canary analysis

Nothing is perfect

- Latency not always a proxy for resources consumption
- Slow warmup of cold servers
- Defeat canary analysis
- Doesn’t cope well with errors disguised as successes

Nothing is perfect

- Latency not always a proxy for resources consumption
- Slow warmup of cold servers
- Defeat canary analysis
- Doesn’t cope well with errors disguised as successes
- Request distribution may be temporarily uneven

Conclusion

- The only algorithm with 4 ✓
- Still experimental
- YMMV

Thank you.

@stevegury

NETFLIX