Cassandra in Response Time Sensitive Environments

Gil Tene, CTO & co-Founder, Azul Systems
@giltene

AZUL SYSTEMS

About me: Gil Tene

- co-founder, CTO @Azul Systems
- Have been working on "think different" GC approaches since 2002
- A Long history building Virtual & Physical Machines, Operating Systems, Enterprise apps, etc...
- I also depress people by demonstrating how terribly wrong their latency measurements are...

* working on real-world trash compaction issues, circa 2004

AZUL SYSTEMS

Azul Systems

- We build Java Virtual Machines
- Powering mission-critical Java applications for Global 2000+
- Deep expertise with latency-sensitive applications
  - from human sensitivity to application responsiveness (seconds to fractions of a second)
  - to low latency trading systems (fractions of a msec)
- Cassandra is one of our common deployment scenarios

Zing Overview

AZUL SYSTEMS

Zing 

- A JVM for Linux/x86 servers
- Delivers a continuously responsive execution platform
- ELIMINATES Garbage Collection as a concern for enterprise applications
- Very wide operating range:
  - Used in everything from low latency to huge in-memory apps
  - 1GB to 1TB Heaps. 10MB/sec to 20GB/sec allocation rates.
- Combats Execution inconsistencies of all types
  - Not just GC: Anything that makes a JVM glitch or slow down
  - "Not just Fast. Always Fast."

AZUL SYSTEMS

What is Zing good for?

- If you have a server-based Java application
- And you are running on Linux (x86)
- And you use using more than ~300MB of memory
- Then Zing will likely deliver superior behavior metrics

AZUL SYSTEMS

Where Zing shines

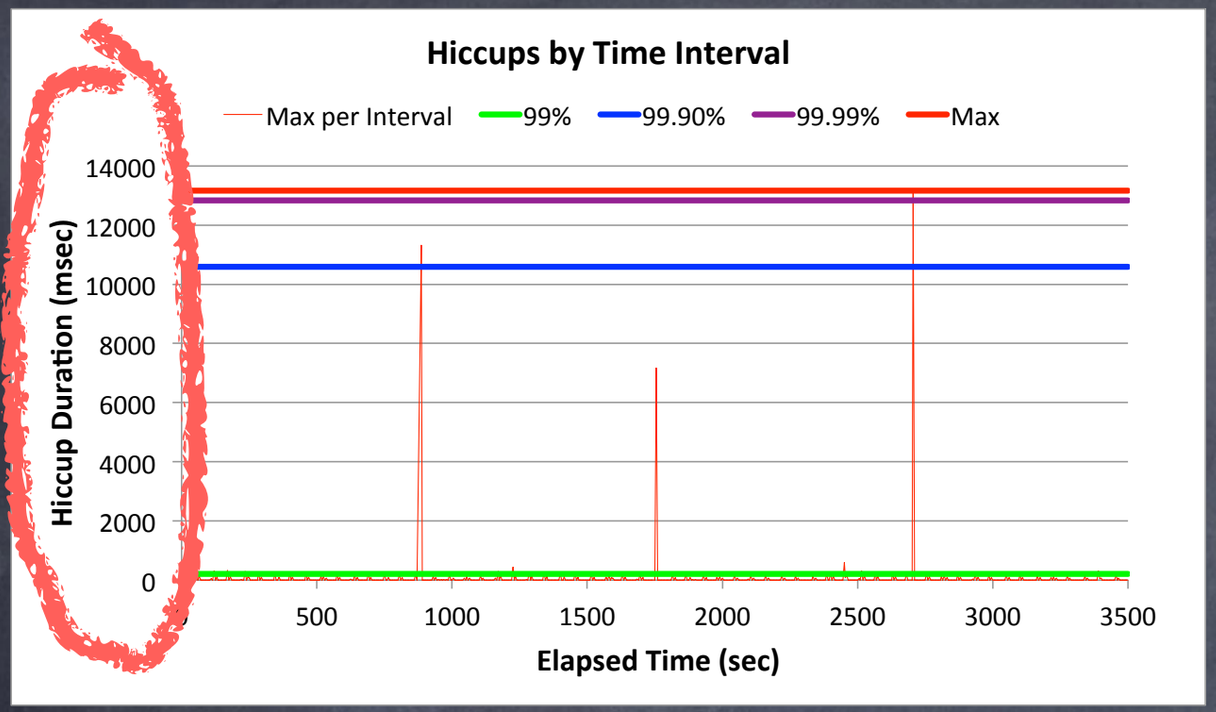
- Low latency
  - Eliminate behavior blips down to the sub-millisecond-units level
- Machine-to-machine "stuff"
  - Support higher *sustainable* throughput (the one that meets SLAs)
- Human response times
  - Eliminate user-annoying response time blips. Multi-second and even fraction-of-a-second blips will be completely gone.
  - Support larger memory JVMs *if needed* (e.g. larger virtual user counts, or larger cache, in-memory state, or consolidating multiple instances)
- "Large" data and in-memory analytics
  - Make batch stuff "business real time". Gain super-efficiencies.

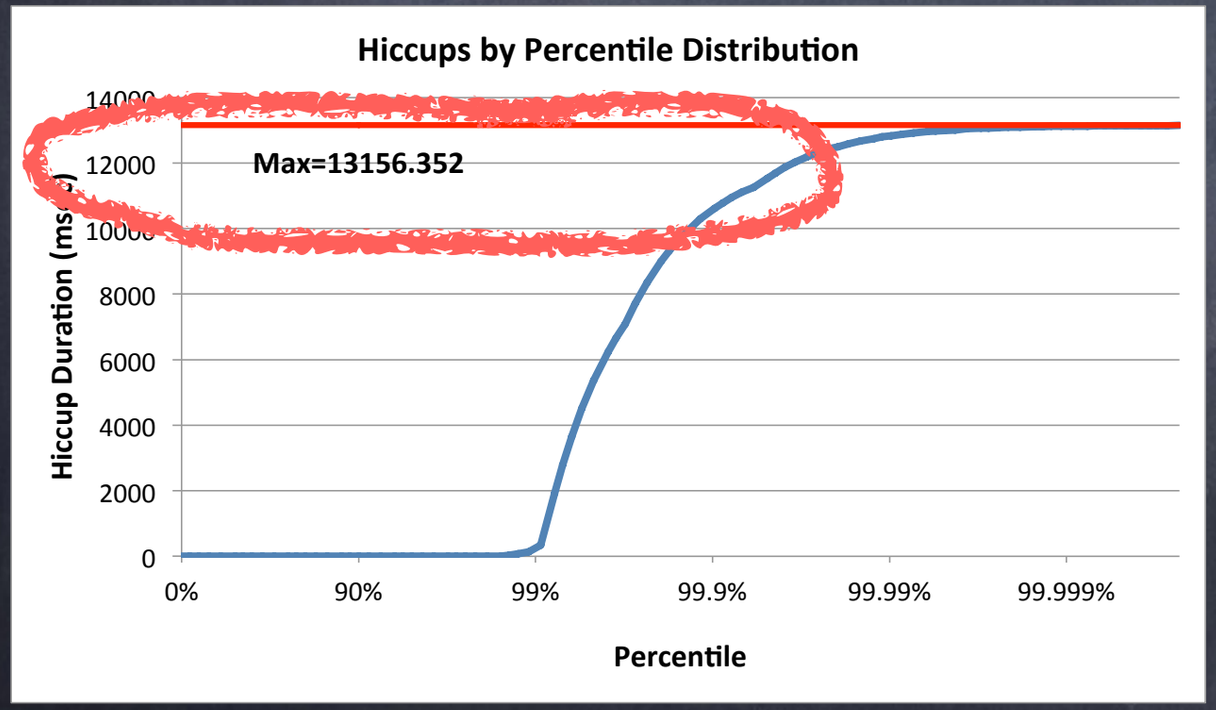
AZUL SYSTEMS

Why Zing?

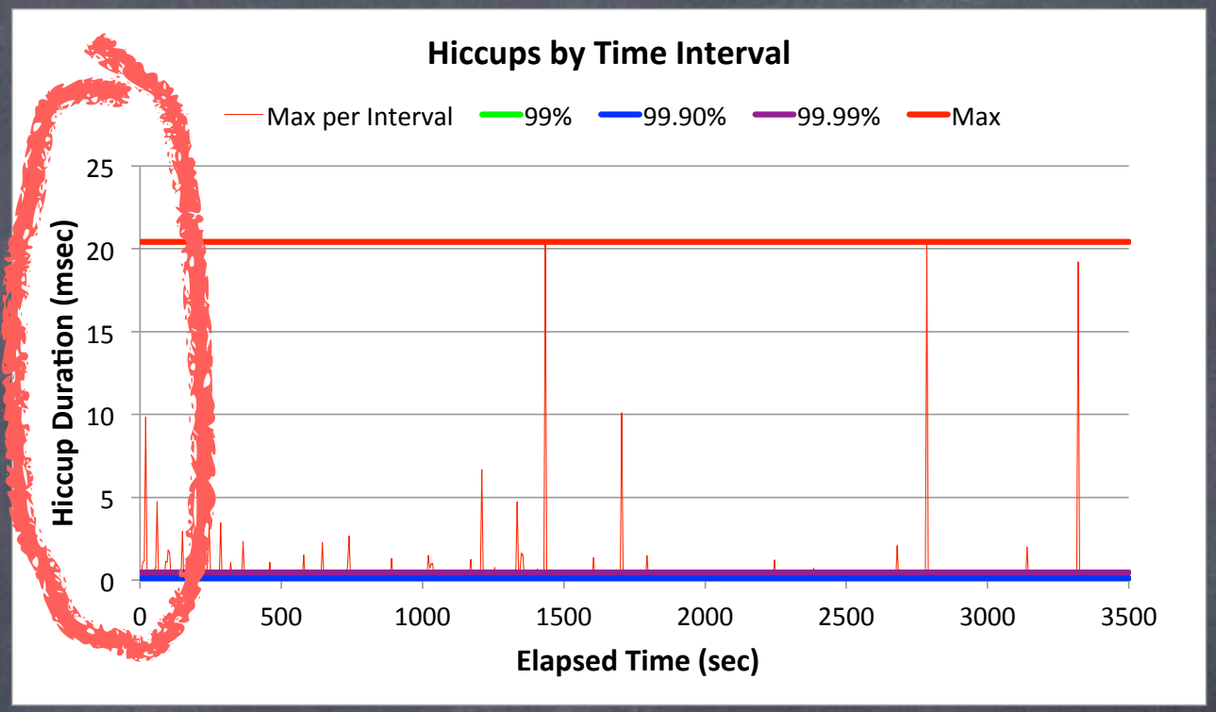
AZUL SYSTEMS

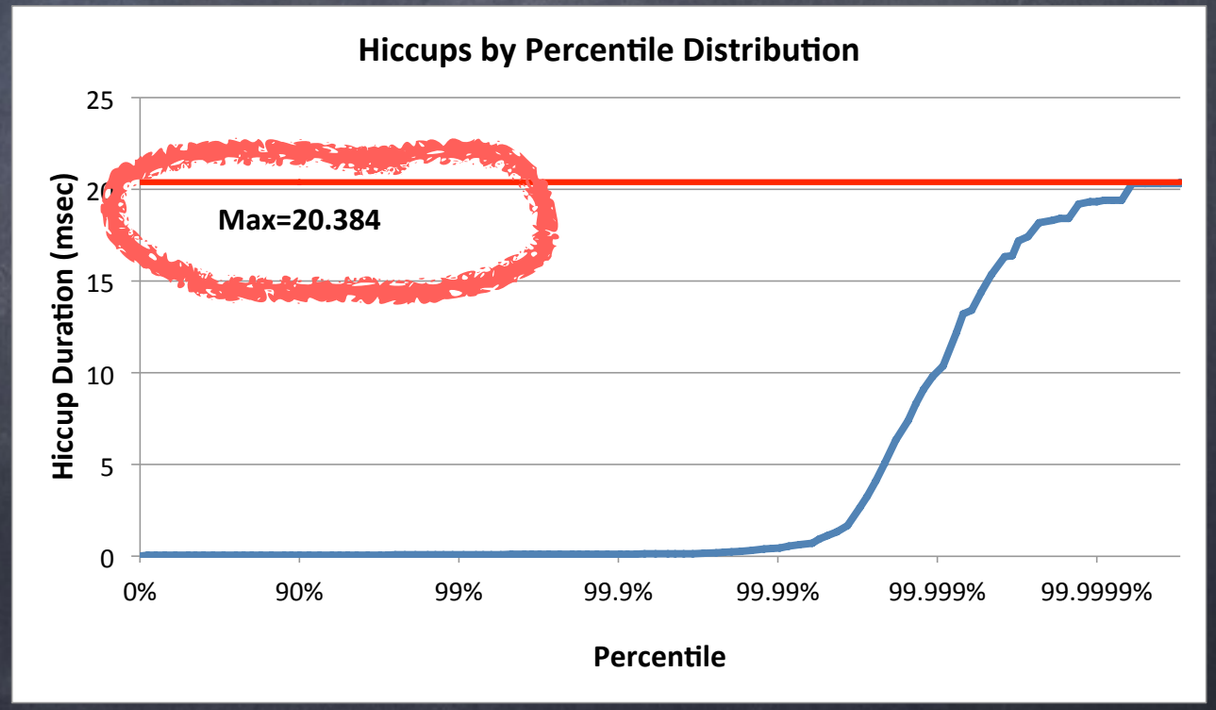
Oracle HotSpot CMS, 1GB in an 8GB heap





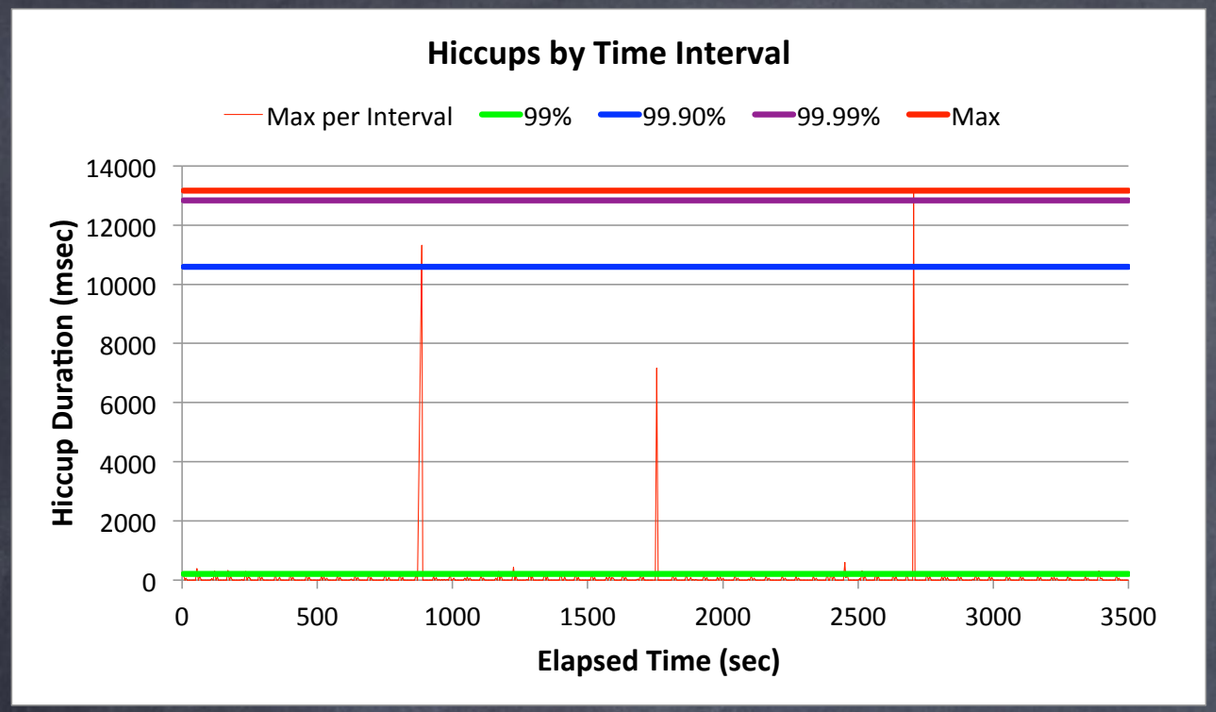
Zing 5, 1GB in an 8GB heap

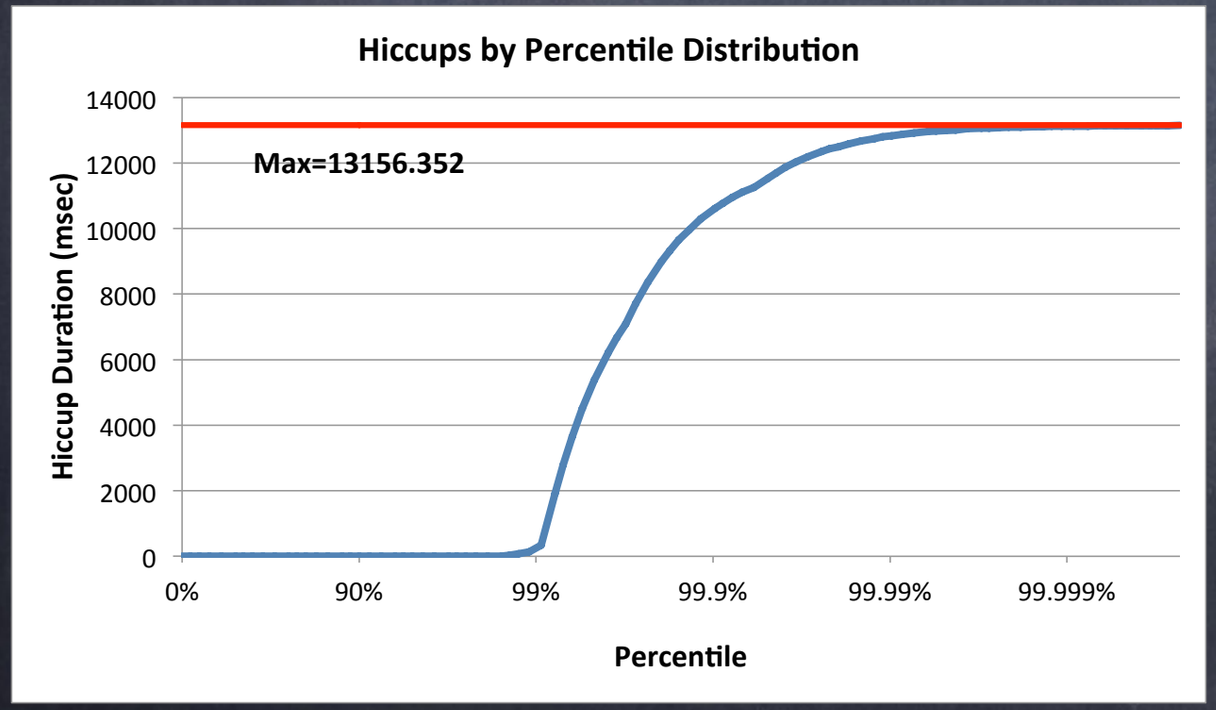




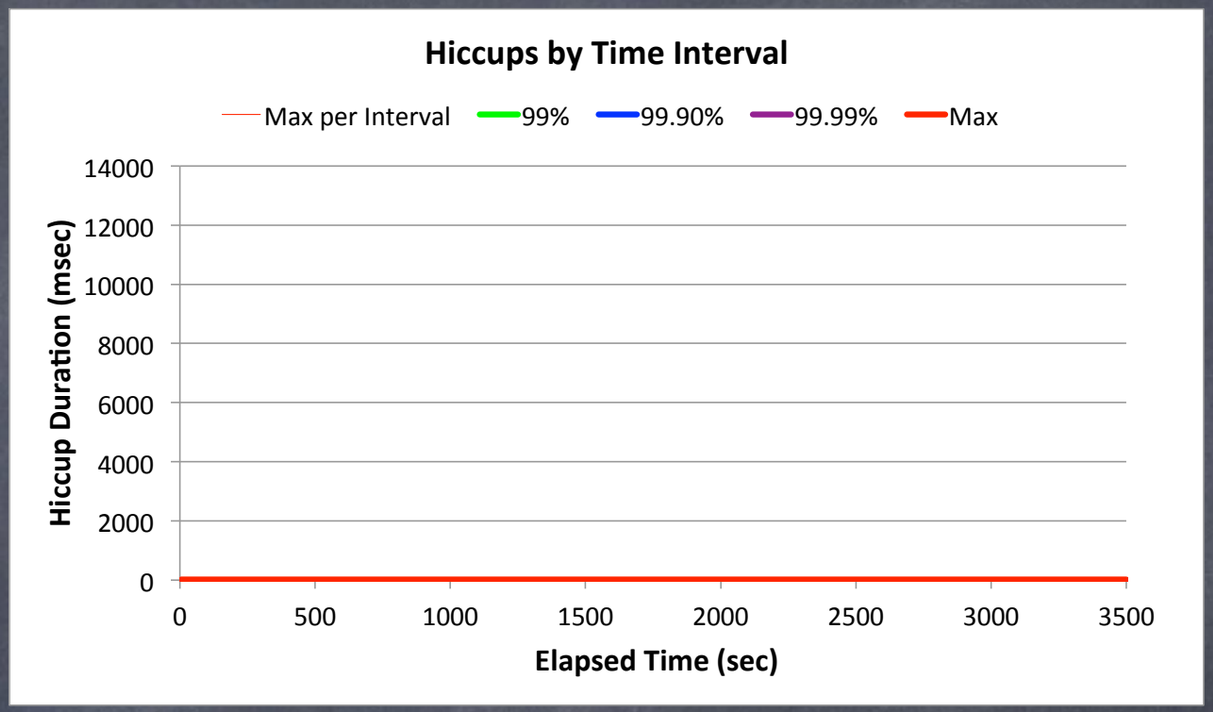
AZUL SYSTEMS

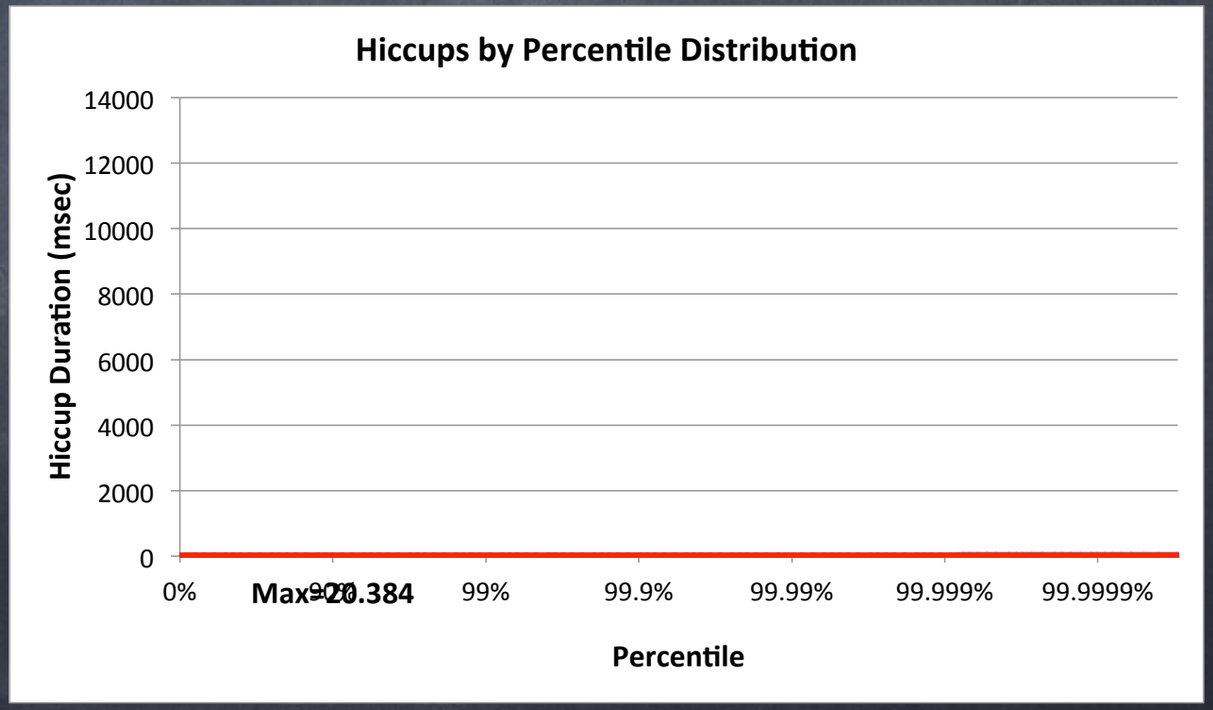
Oracle HotSpot CMS, 1GB in an 8GB heap





Zing 5, 1GB in an 8GB heap





Drawn to scale

AZUL SYSTEMS

Sustainable Throughput:
The throughput achieved while safely maintaining service levels

AZUL SYSTEMS

Percentiles Matter

AZUL SYSTEMS

Is the 99%'ile "rare"?

AZUL SYSTEMS

Cumulative probability…

What are the chances of a single web page view experiencing the 99%'ile latency of:

- A single search engine node?

- A single Key/Value store node?

- A single Database node?

- A single CDN request?

AZUL SYSTEMS

Site	# of requests	page loads that would experience the 99%'lie [(1 - (.99 ^ N)) * 100%]
amazon.com	190	85.2%
kohls.com	204	87.1%
jcrew.com	112	67.6%
saksfifthavenue.com	109	66.5%
--	--	--
nytimes.com	173	82.4%
cnn.com	279	93.9%
--	--	--
twitter.com	87	58.3%
pinterest.com	84	57.0%
facebook.com	178	83.3%
--	--	--
google.com (yes, that simple noise-free page)	31	26.7%
google.com search for "http requests per page"	76	53.4%

AZUL SYSTEMS

Which HTTP response time metric is more "representative" of user experience?

The 95%'lie or the 99.9%'lie

AZUL SYSTEMS

Gauging user experience

Example: A typical user session involves 5 page loads, averaging 40 resources per page.

- How many of our users will NOT experience something worse than the 95%'lie?

Answer: ~0.003%

- How may of our users will experience at least one response that is longer than the 99.9%'lie?

Answer: ~18%

AZUL SYSTEMS

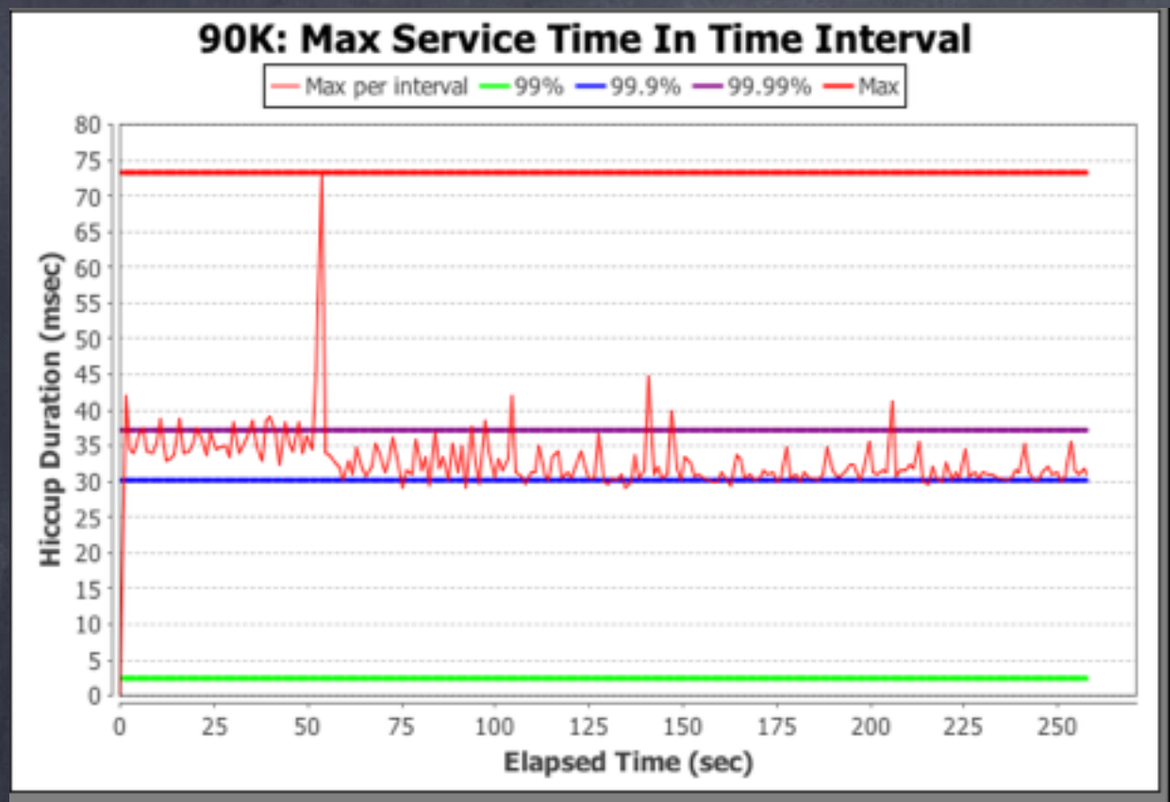
Response Time vs. Service Time

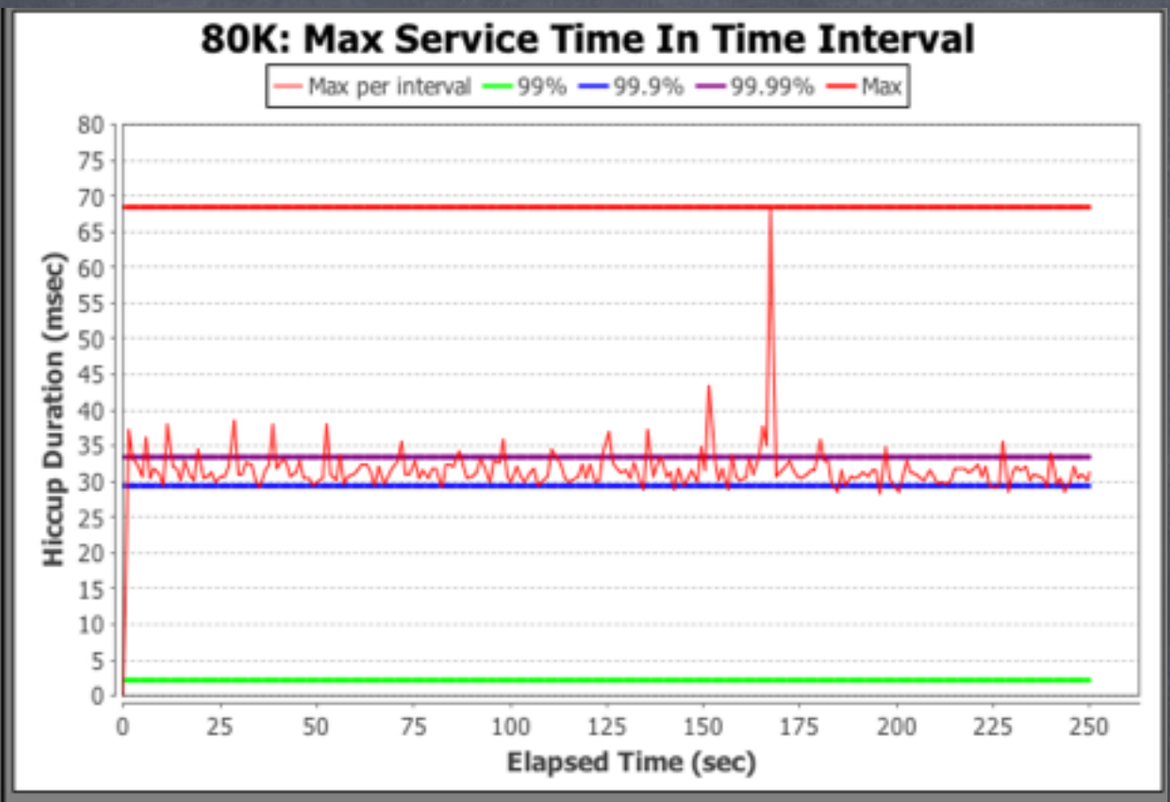
AZUL SYSTEMS

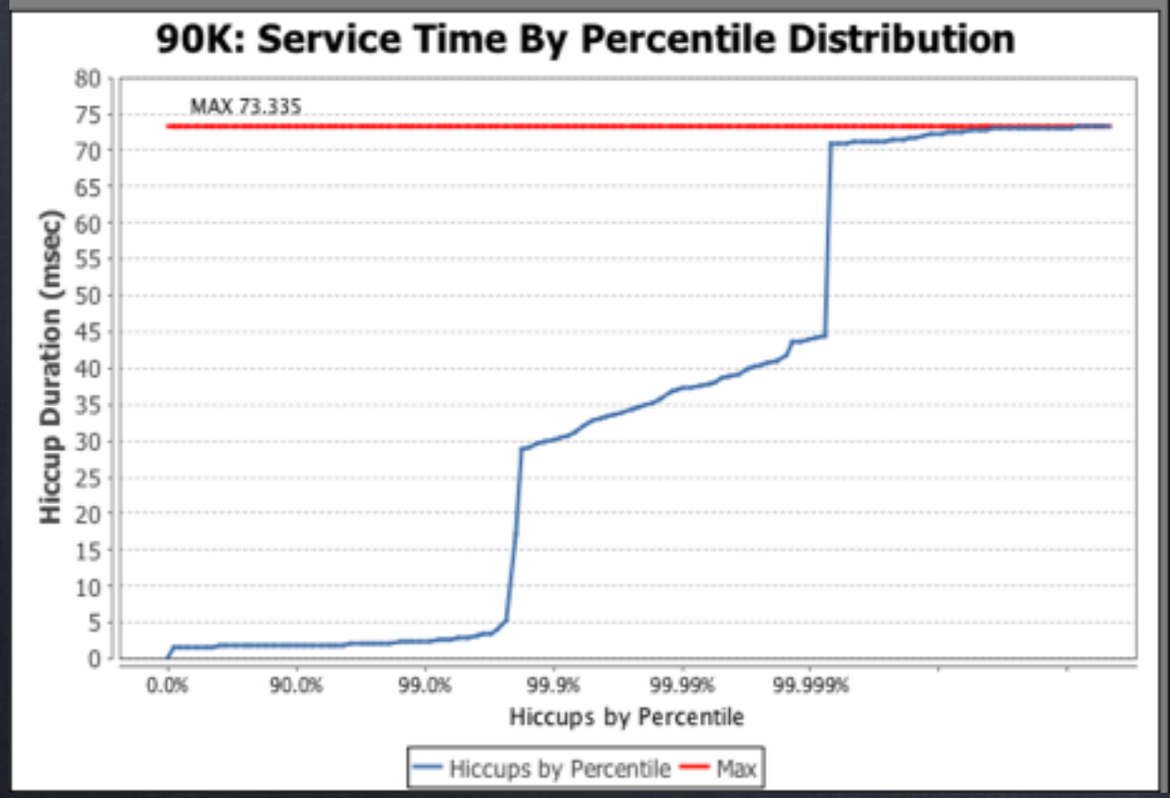
Service Time vs. Response Time

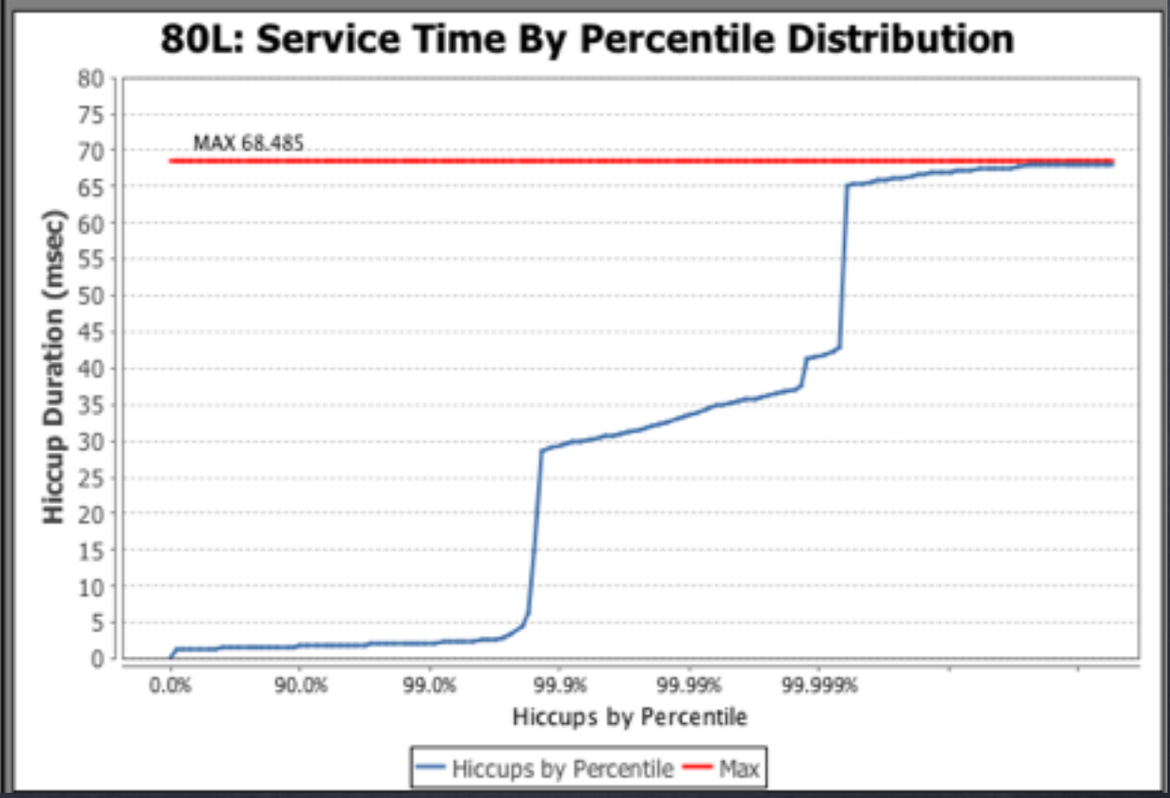
AZUL SYSTEMS

Service Time, 90K/s vs 80K/s



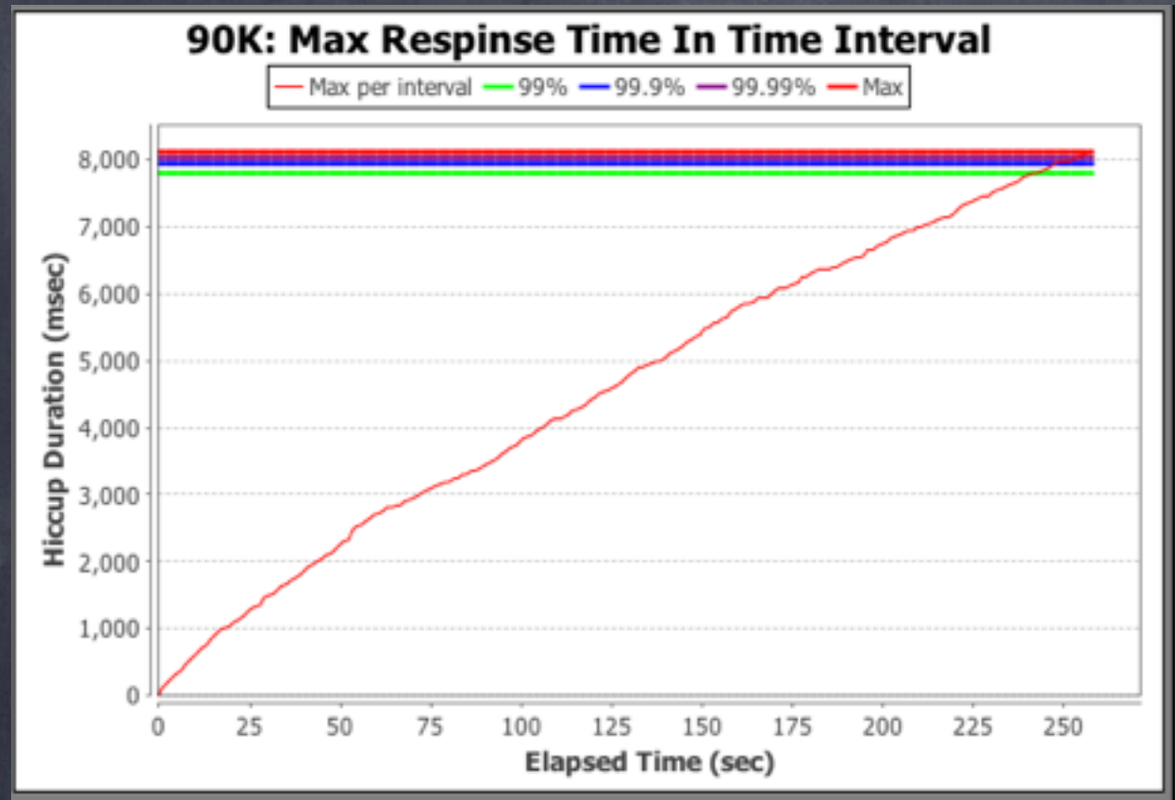


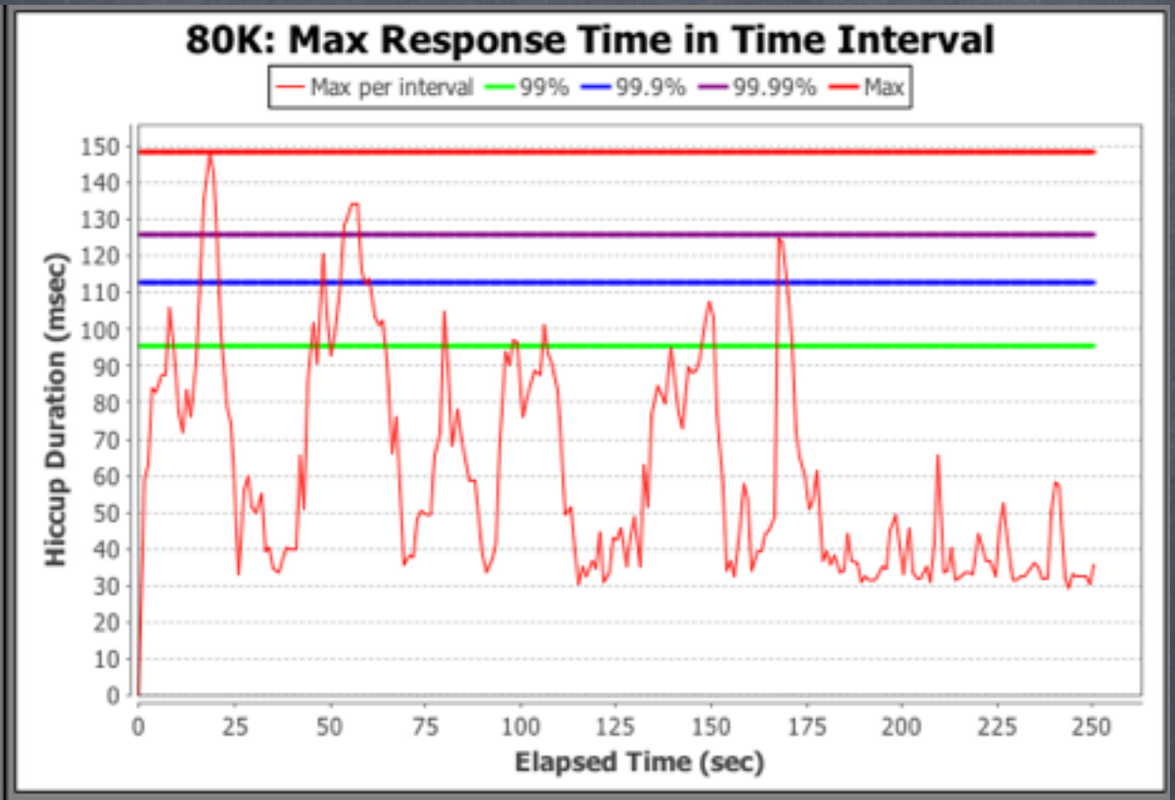


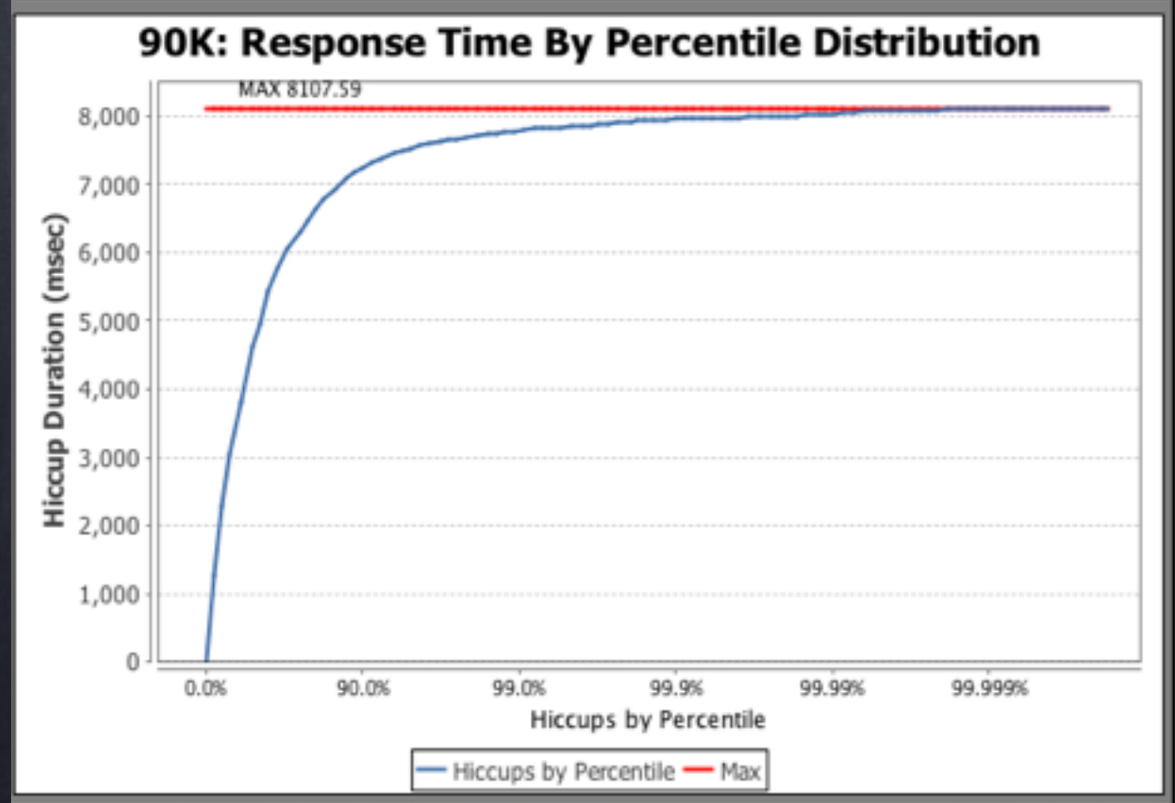


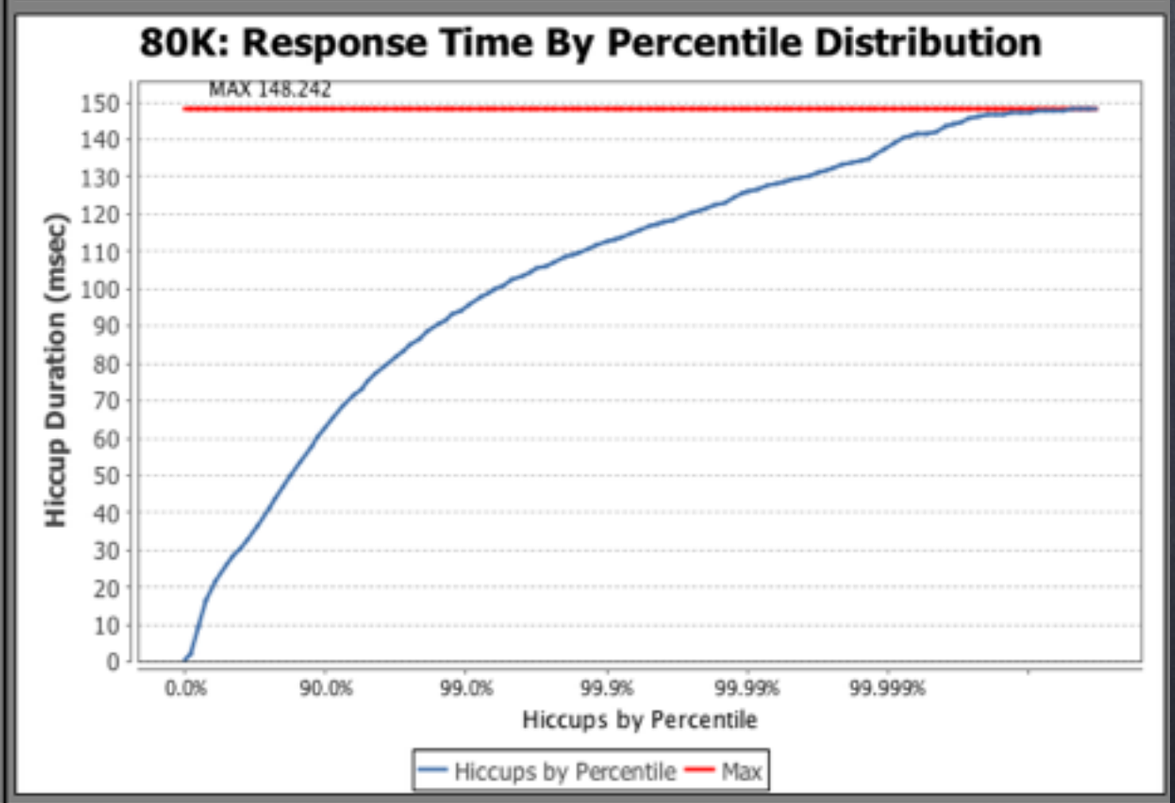
AZUL SYSTEMS

Response Time, 90K/s vs 80K/s



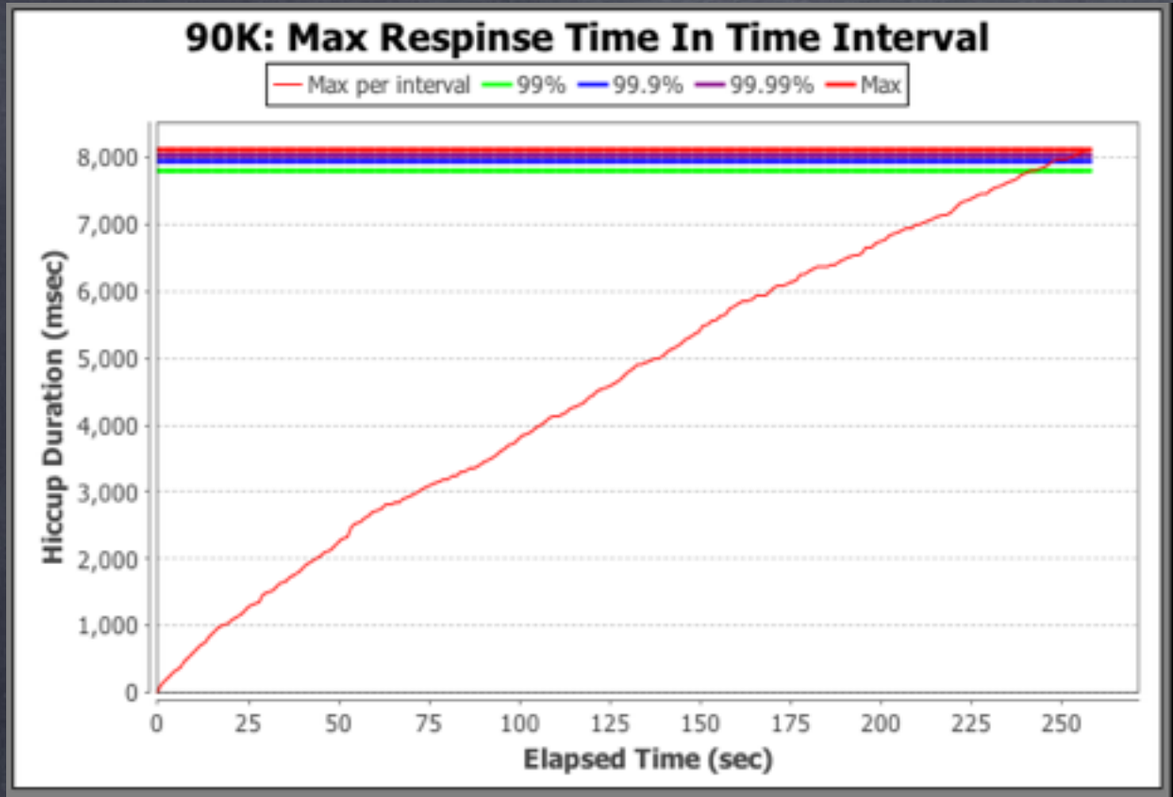


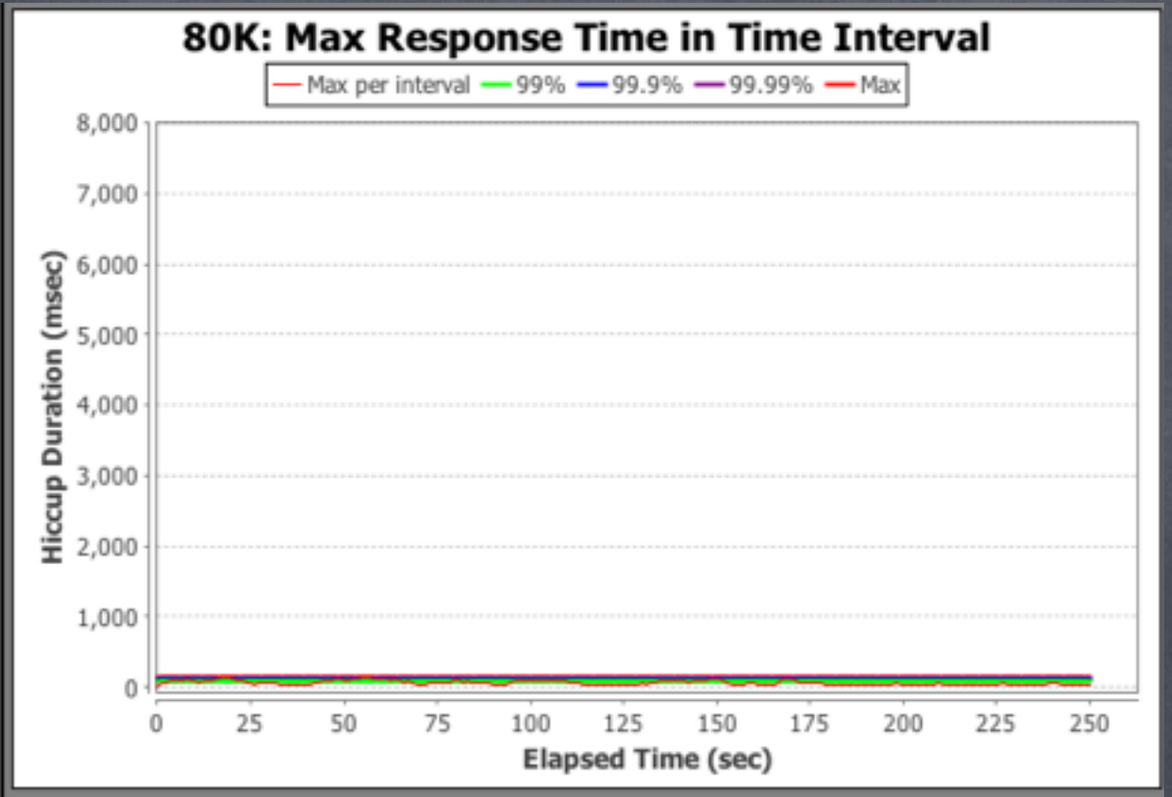


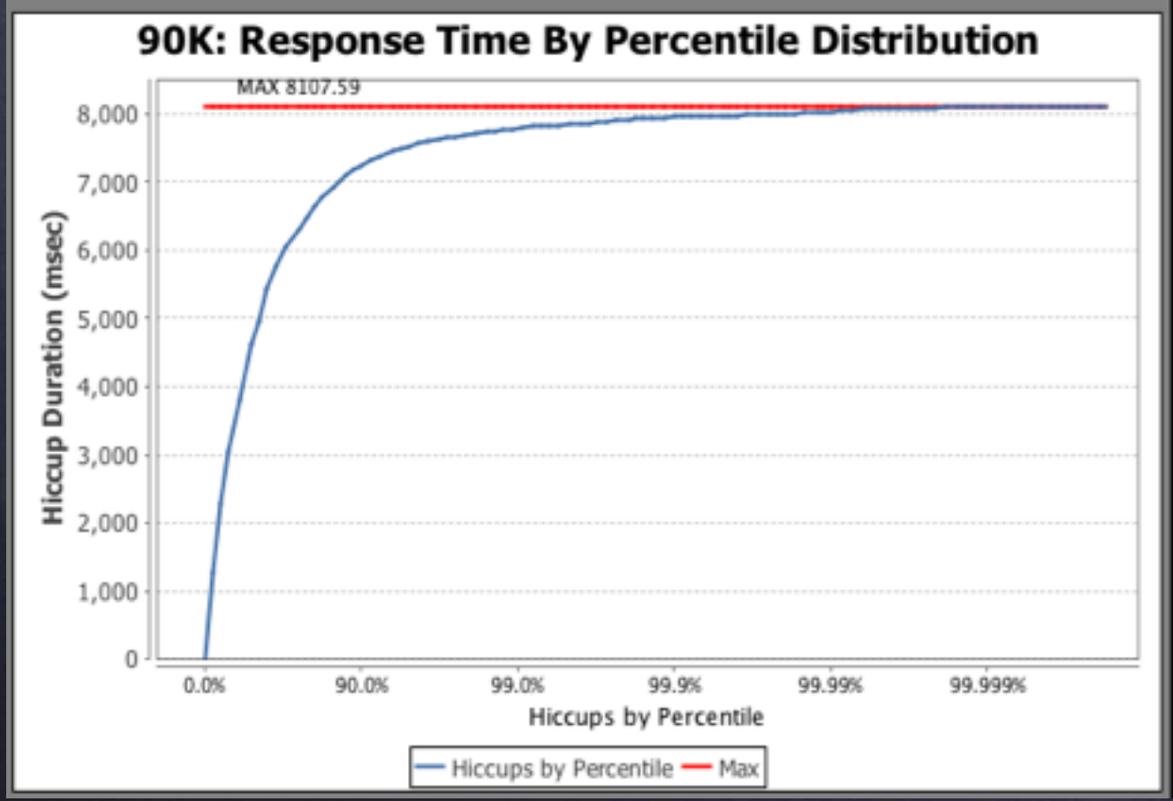


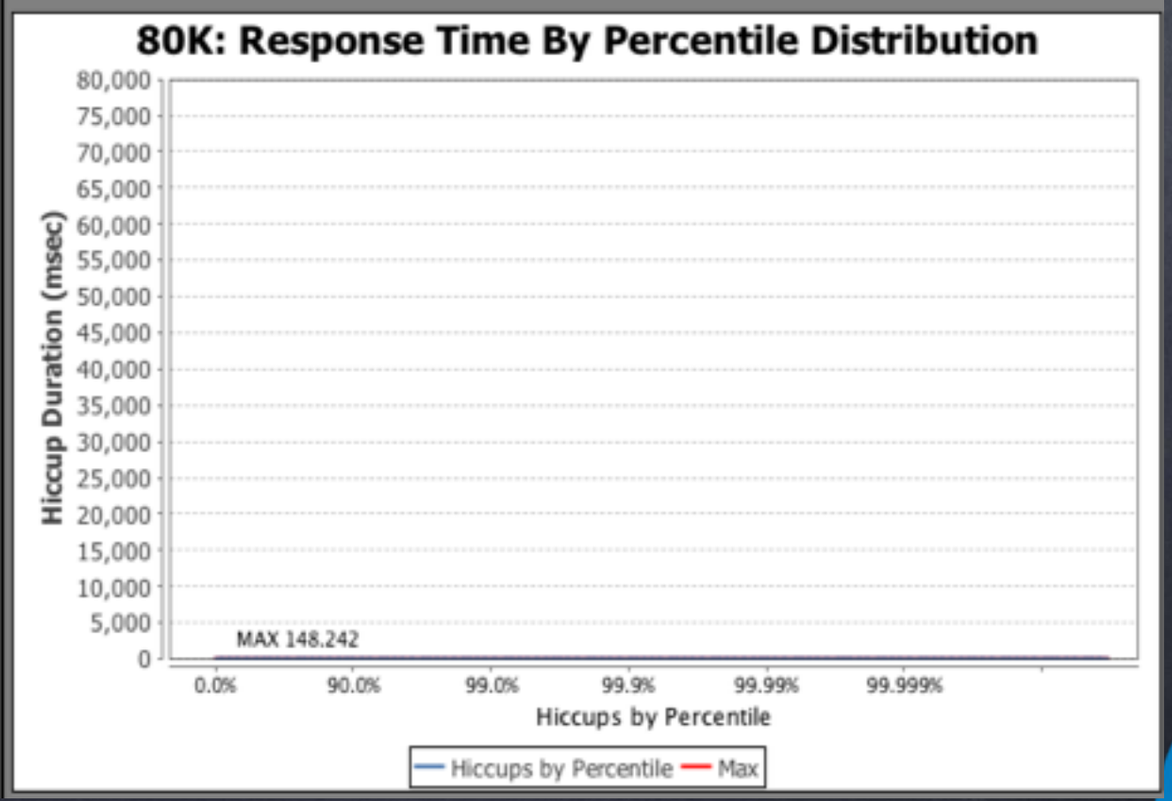
AZUL SYSTEMS

Response Time, 90K/s vs 80K/s : Boom!









AZUL SYSTEMS

"coordinator as savior" latency myth

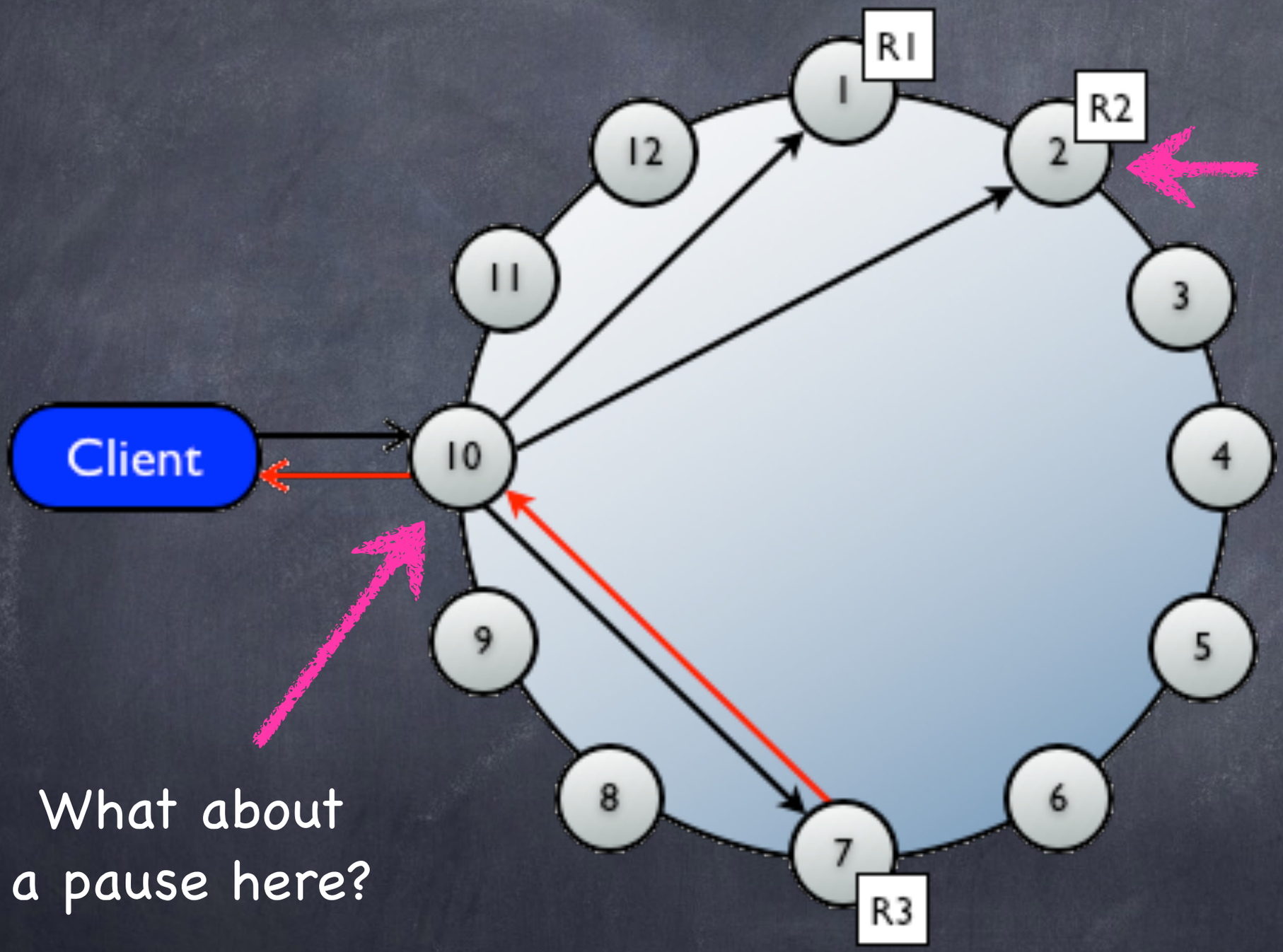
"But with Cassandra's Coordinator and Quorum Consistency levels..."

Theory: If one node pauses, other nodes are not likely to pause at the same time

... so a quorum will be reached without observing any one node's pause

AZUL SYSTEMS

Anatomy of a quorum read…



A pause here won't be noticed by client…

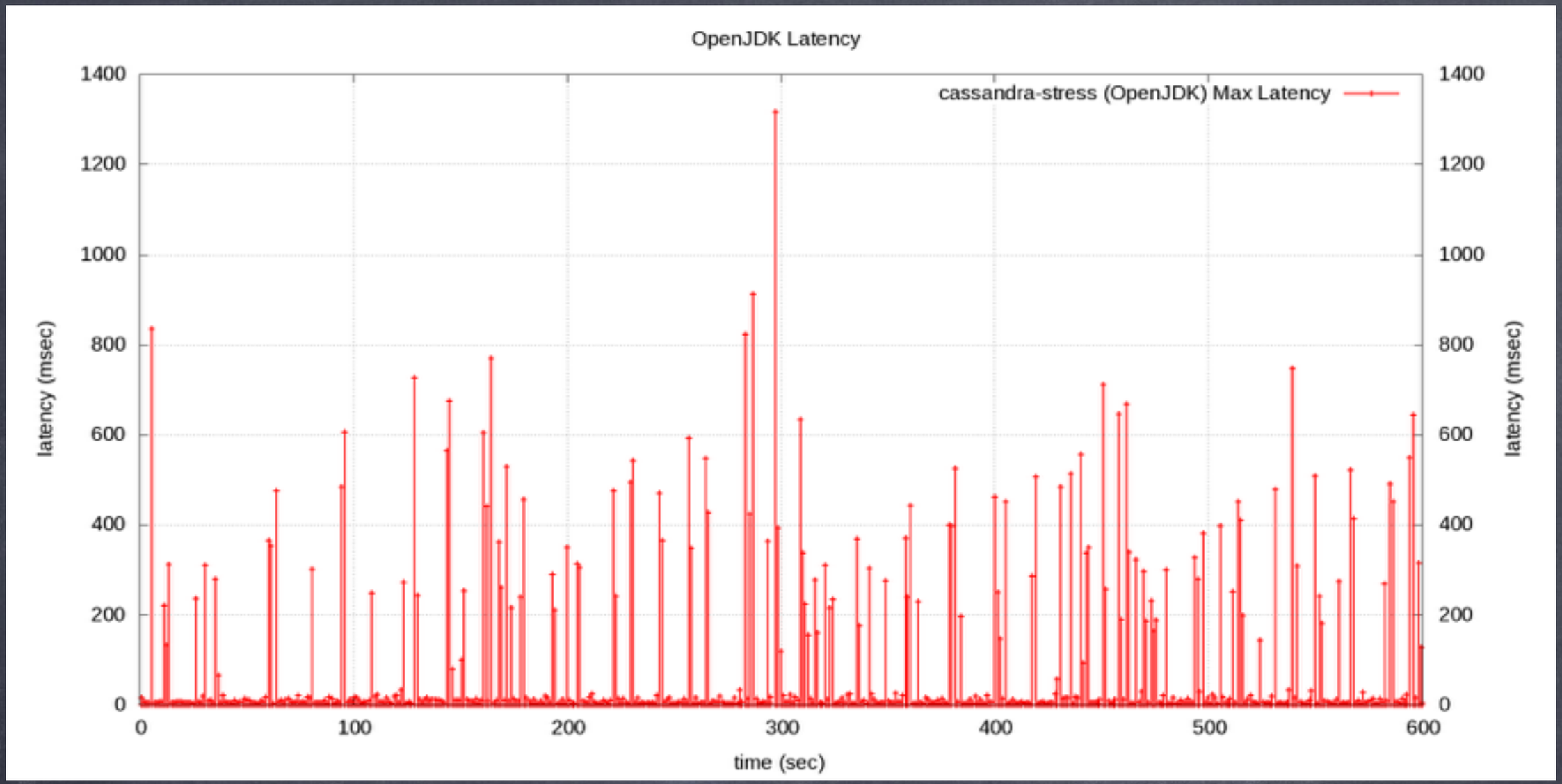
What about a pause here?

And since every node is also a coordinator…

AZUL SYSTEMS

Cassandra behavior on Zing

AZUL SYSTEMS

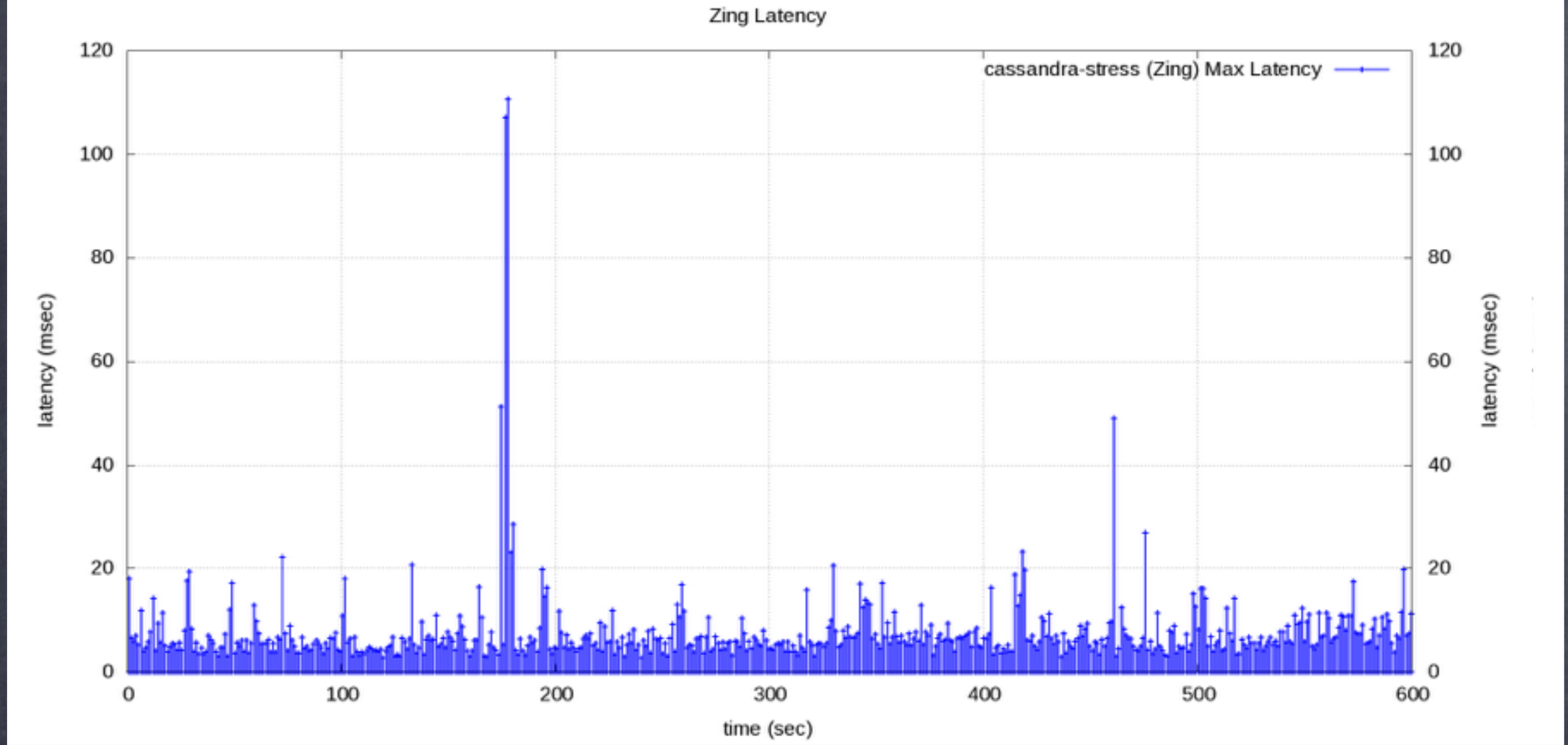



```
op rate                   : 40001
partition rate            : 26996
row rate                  : 26996
latency mean              : 30.6 (0.7)
latency median            : 0.5 (0.5)
latency 95th percentile   : 244.4 (1.1)
latency 99th percentile   : 537.4 (2.0)
latency 99.9th percentile : 1052.2 (8.4)
latency max               : 1314.9 (1312.8)
```

Response Time

Service time

AZUL SYSTEMS



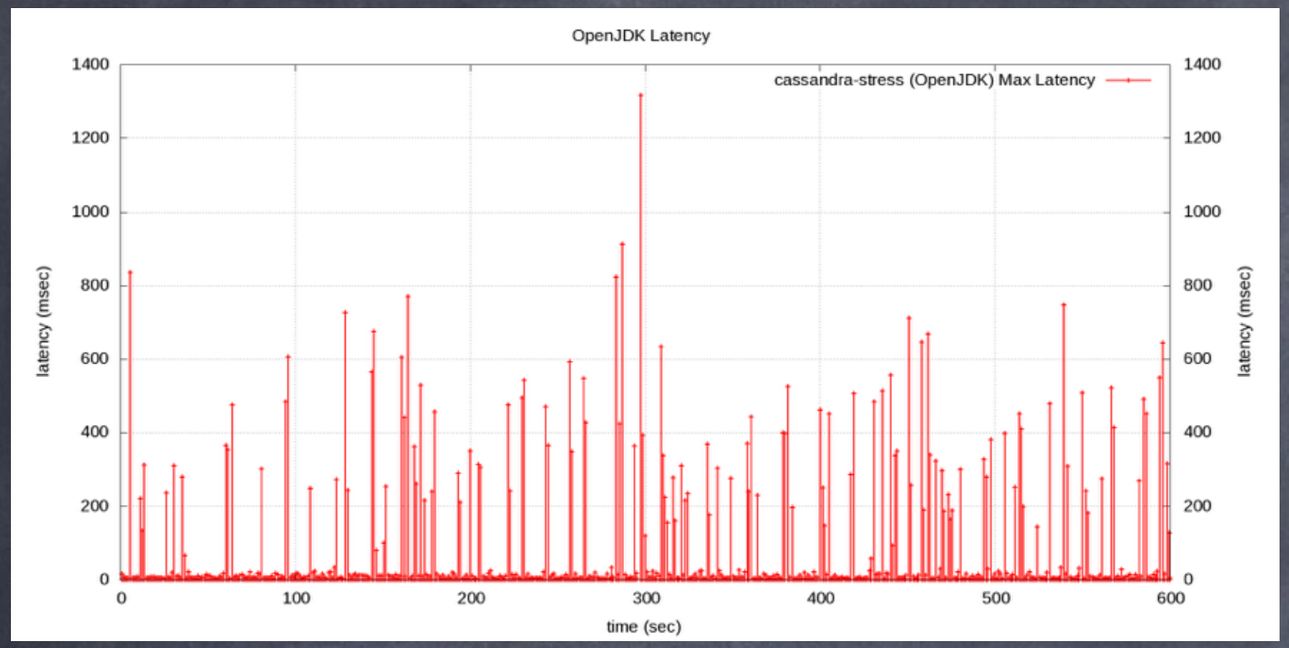

```
op rate                   : 40001
partition rate            : 26961
row rate                  : 26961
latency mean              : 0.6 (0.5)
latency median            : 0.5 (0.5)
latency 95th percentile   : 1.0 (0.9)
latency 99th percentile   : 2.7 (1.9)
latency 99.9th percentile : 13.3 (3.8)
latency max               : 110.6 (28.2)
```

Response Time

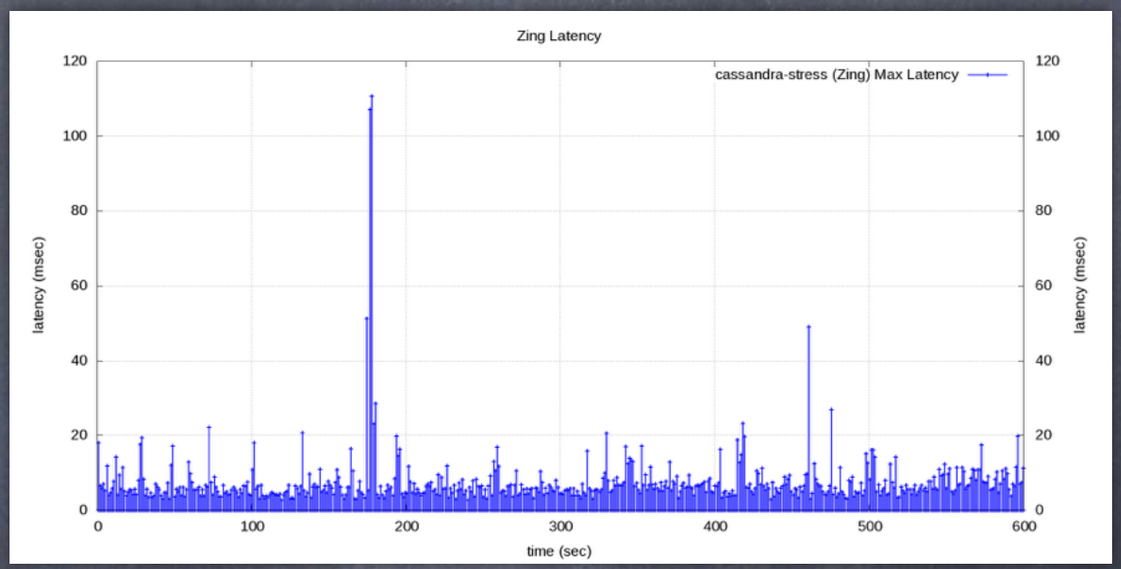
Service time

AZUL SYSTEMS

OpenJDK: 200-1400 msec stalls



Zing




```
op rate                   : 40001
partition rate            : 26996
row rate                  : 26996
latency mean              : 30.6 (0.7)
latency median            : 0.5 (0.5)
latency 95th percentile   : 244.4 (1.1)
latency 99th percentile   : 537.4 (2.0)
latency 99.9th percentile : 1052.2 (8.4)
latency max               : 1314.9 (1312.8)
```

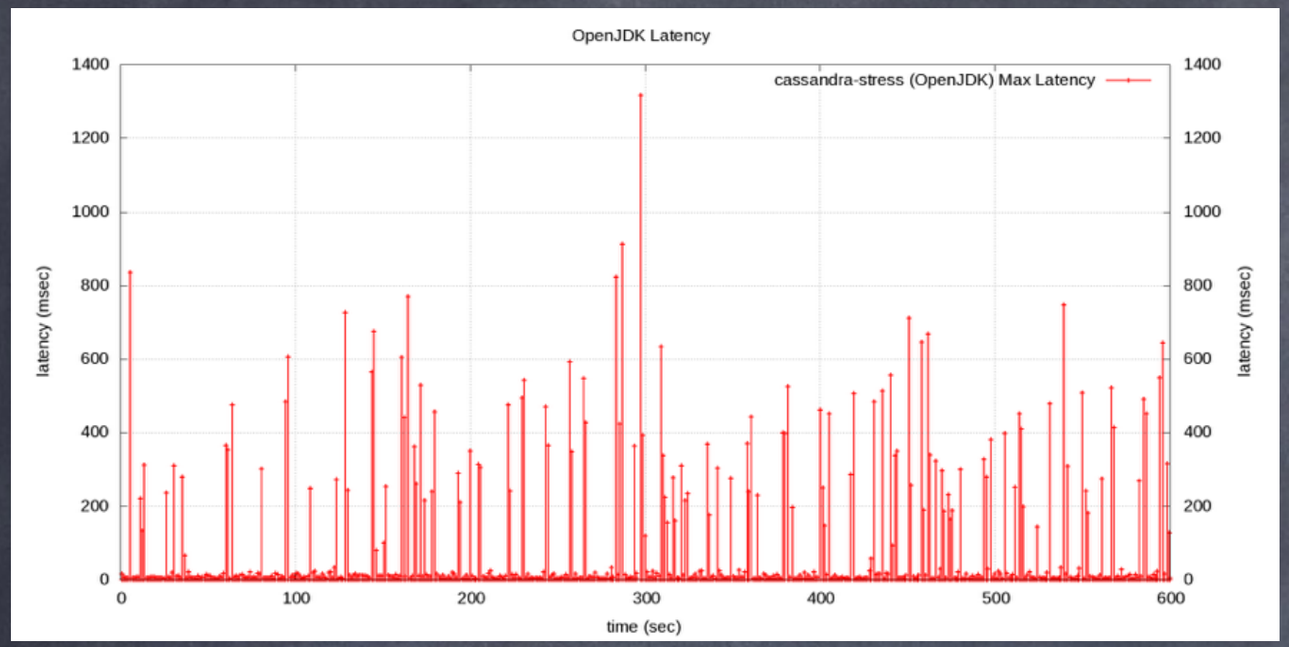
Response Time Service time

```
op rate                   : 40001
partition rate            : 26961
row rate                  : 26961
latency mean              : 0.6 (0.5)
latency median            : 0.5 (0.5)
latency 95th percentile   : 1.0 (0.9)
latency 99th percentile   : 2.7 (1.9)
latency 99.9th percentile : 13.3 (3.8)
latency max               : 110.6 (28.2)
```

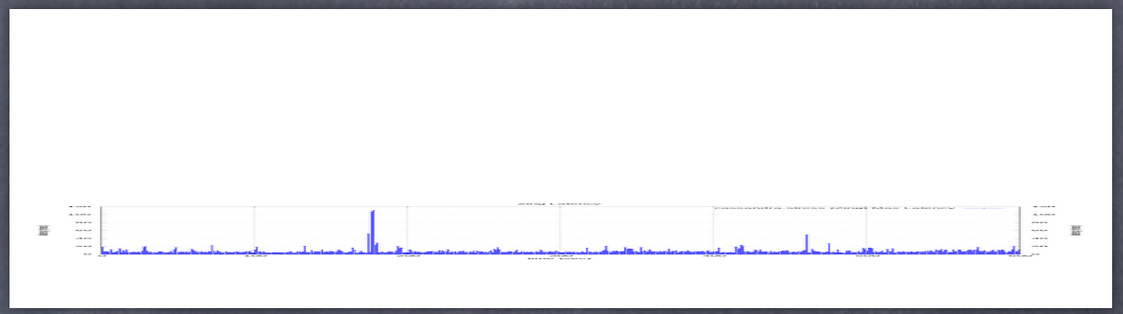
Response Time Service time

AZUL SYSTEMS

OpenJDK: 200-1400 msec stalls



Zing (drawn to scale)



```
op rate                   : 40001
partition rate            : 26996
row rate                  : 26996
latency mean              : 30.6 (0.7)
latency median            : 0.5 (0.5)
latency 95th percentile   : 244.4 (1.1)
latency 99th percentile   : 537.4 (2.0)
latency 99.9th percentile : 1052.2 (8.4)
latency max               : 1314.9 (1312.8)
```

Response Time Service time

```
op rate                   : 40001
partition rate            : 26961
row rate                  : 26961
latency mean              : 0.6 (0.5)
latency median            : 0.5 (0.5)
latency 95th percentile   : 1.0 (0.9)
latency 99th percentile   : 2.7 (1.9)
latency 99.9th percentile : 13.3 (3.8)
latency max               : 110.6 (28.2)
```

Response Time Service time

AZUL SYSTEMS

What if we focused on "already low latency" setups?

"I know really bad GC pauses may happen once in a while, but I'm interested in the common behavior between those…"

AZUL SYSTEMS

A set of pure read experiments…

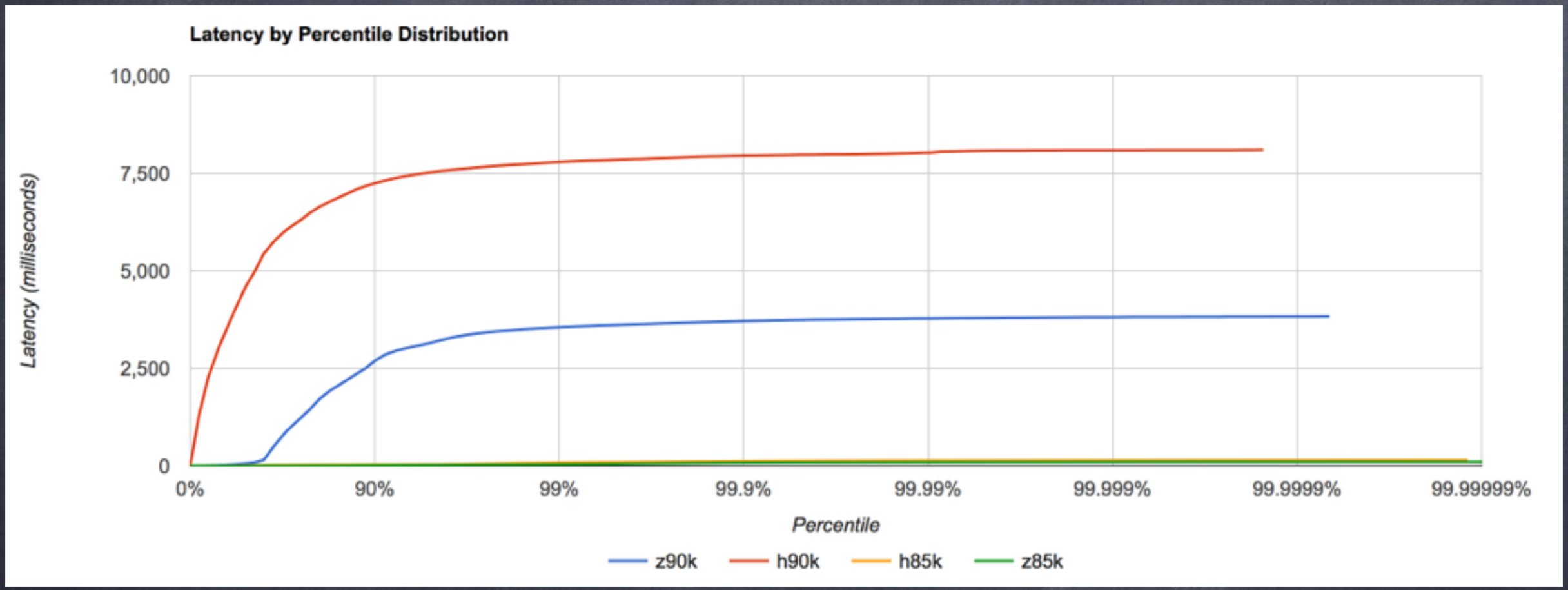
aimed at highly repeatable results

(focused on frequent blips, not the hard to reliably repeat huge pauses)

* Same AWS r3.8xlarge instance (underutilized)
** single node cluster, pre-primed with 5M entries
*** stressed via (enhanced) cassandra-stress, pure read test

AZUL SYSTEMS

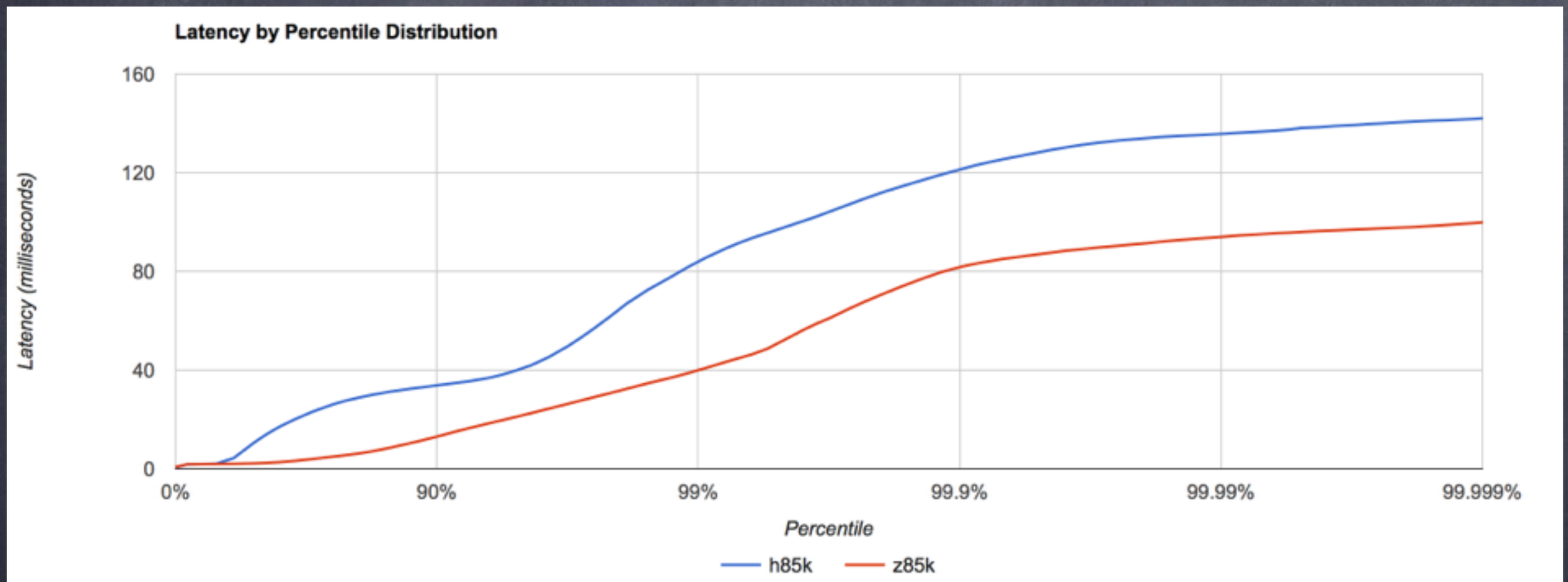
HotSpot @90K/s & 85K/s vs. Zing @90K/s & 85K/s



Wrong Place to Look: They both “suck” at >85K/sec

AZUL SYSTEMS

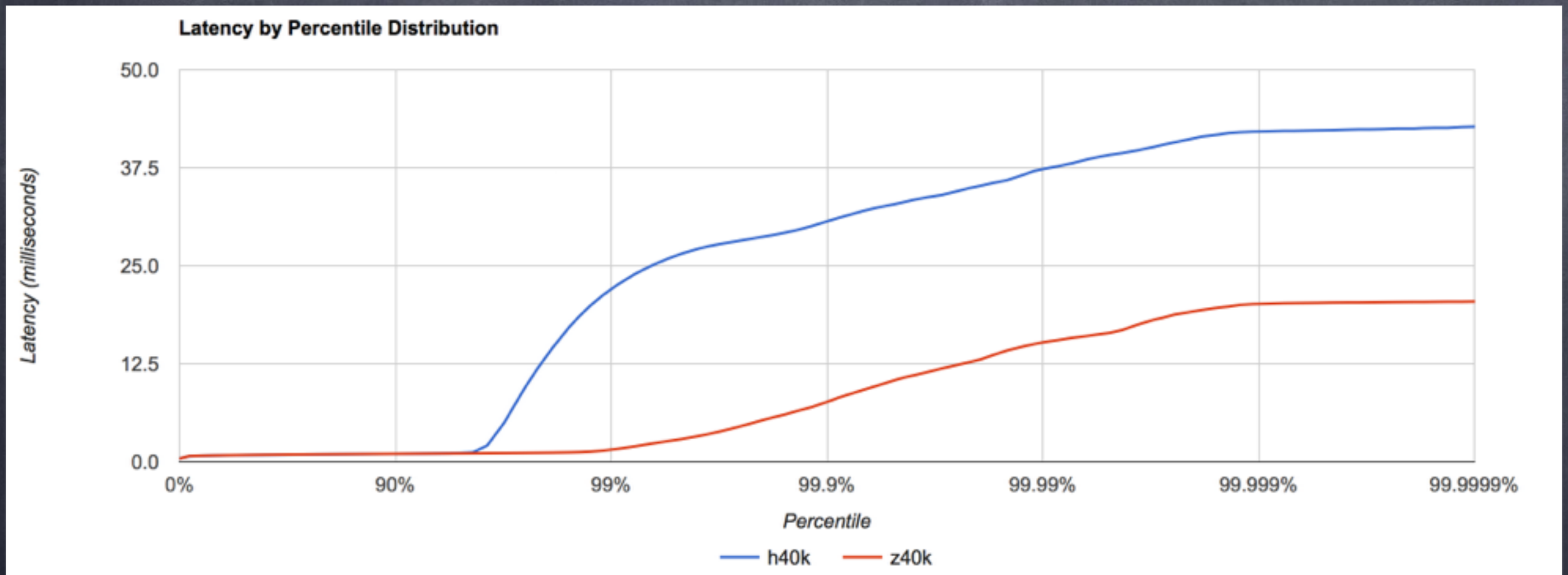
HotSpot 85K/s vs. Zing 85K/s



Looks good, but still
the wrong place to look

AZUL SYSTEMS

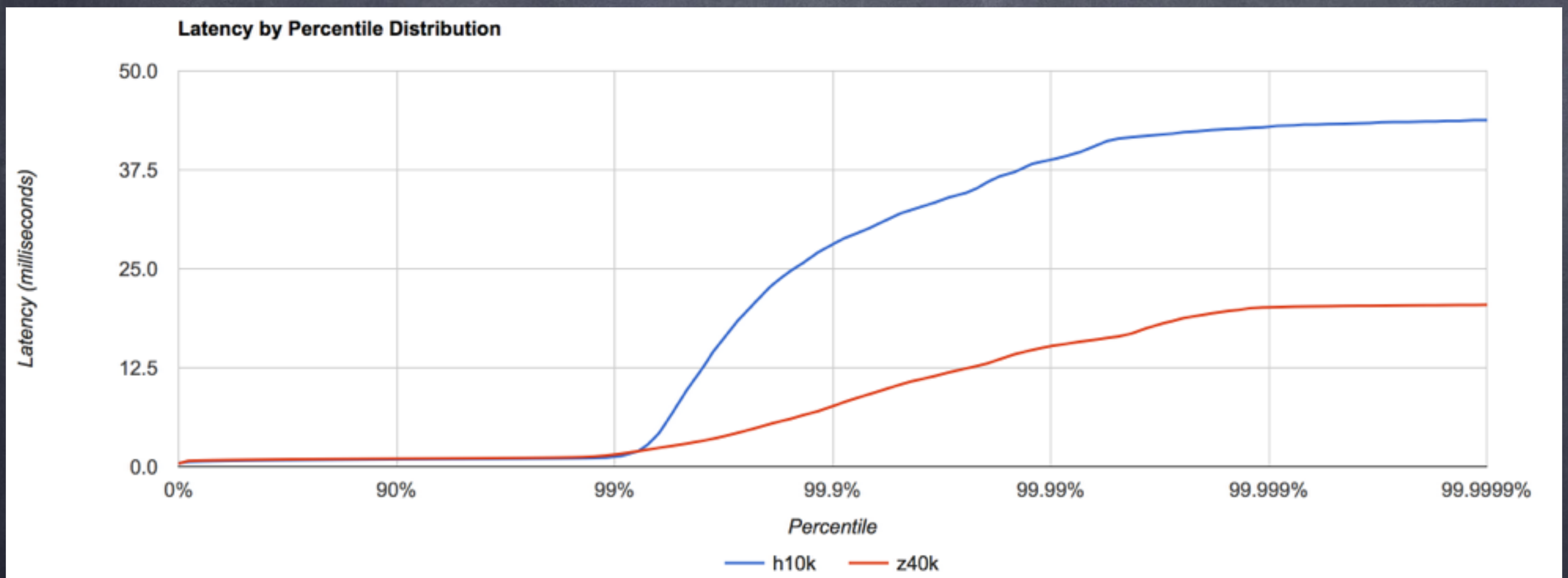
HotSpot @40K/s vs. Zing @40K/s



More interesting…
What can we do with this?

AZUL SYSTEMS

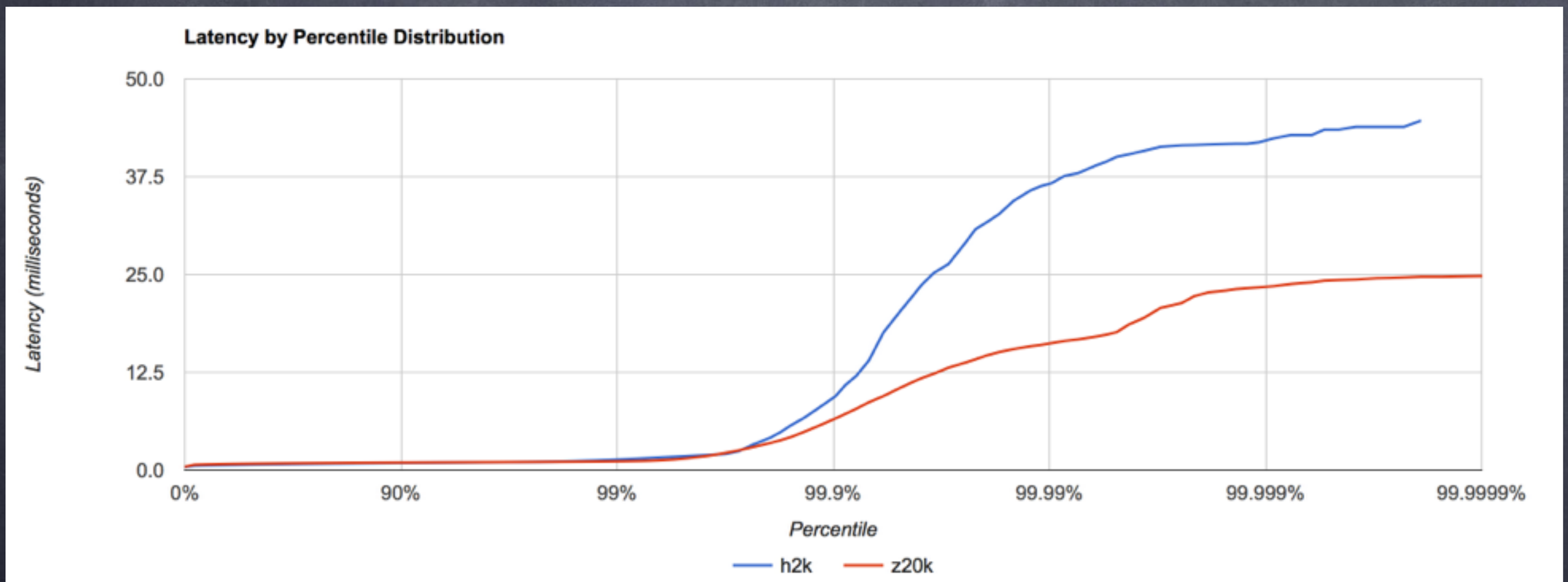
HotSpot @10K/s vs. Zing @40K/s



E.g. if “99%’ile < 5msec” was a goal:
Zing delivers similar 99%’ile and superior 99.9%’ile+
while carrying 4x the throughput

AZUL SYSTEMS

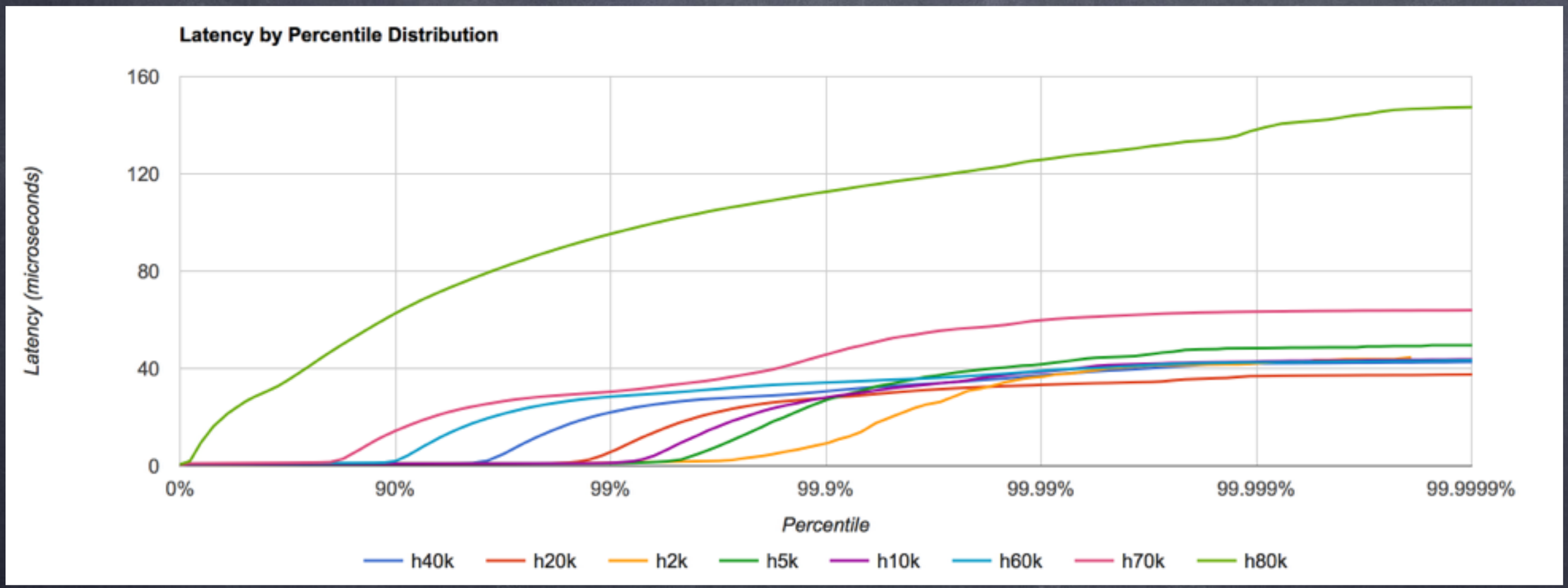
HotSpot @2K/s vs. Zing @20K/s



E.g. if "99.9%'ile < 10msec" was a goal:
Zing delivers similar 99%'ile and 99.9%'ile
while carrying 10x the throughput

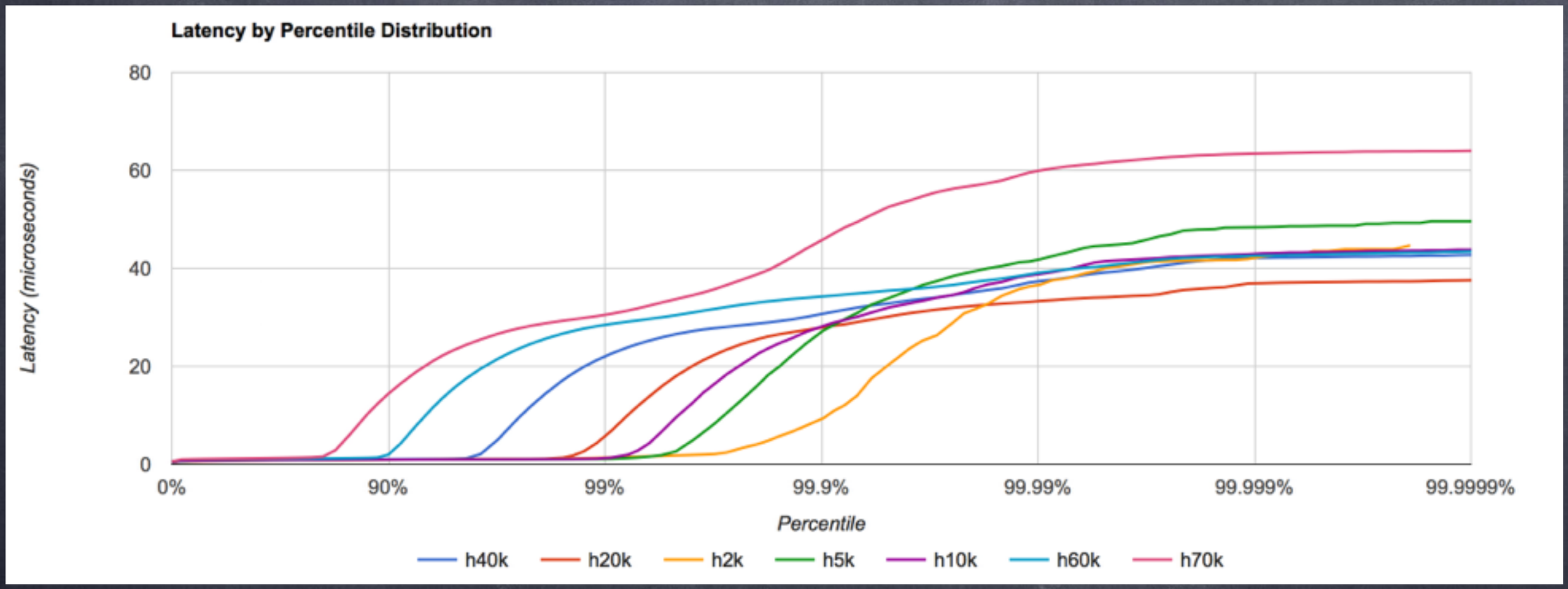
AZUL SYSTEMS

HotSpot @2k thru 80k



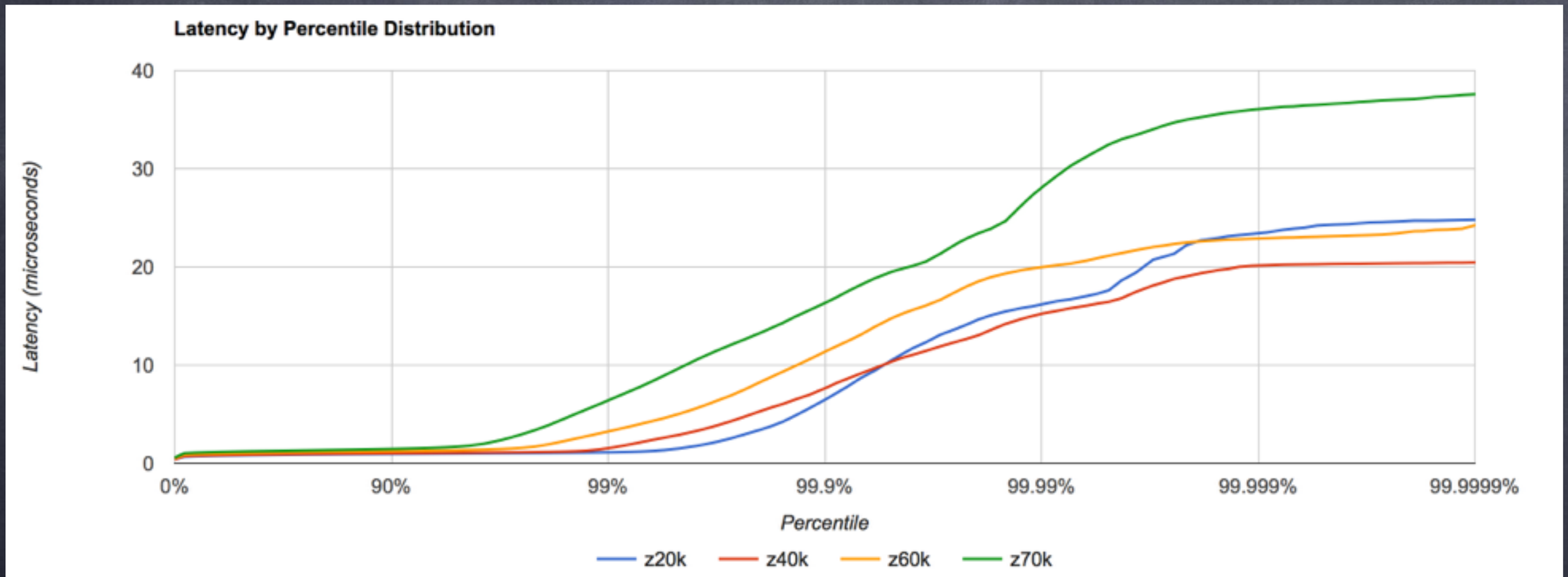
AZUL SYSTEMS

HotSpot @2k thru 70k



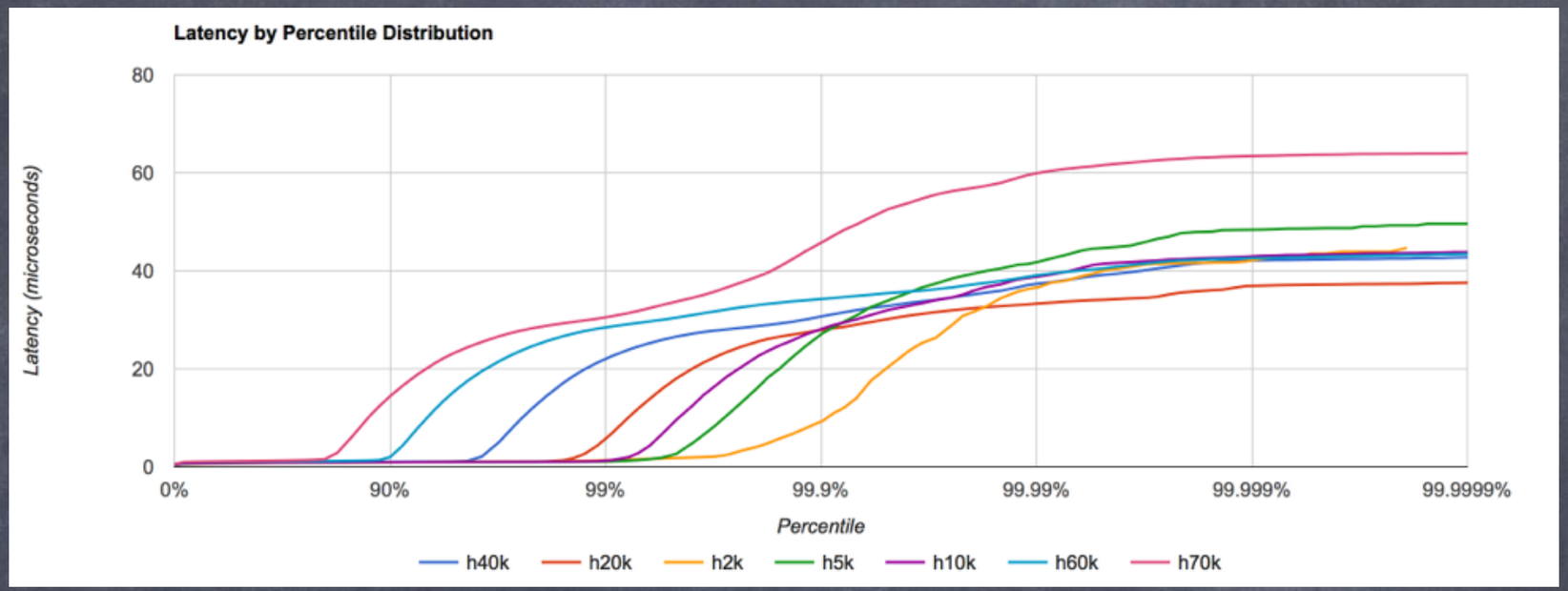
AZUL SYSTEMS

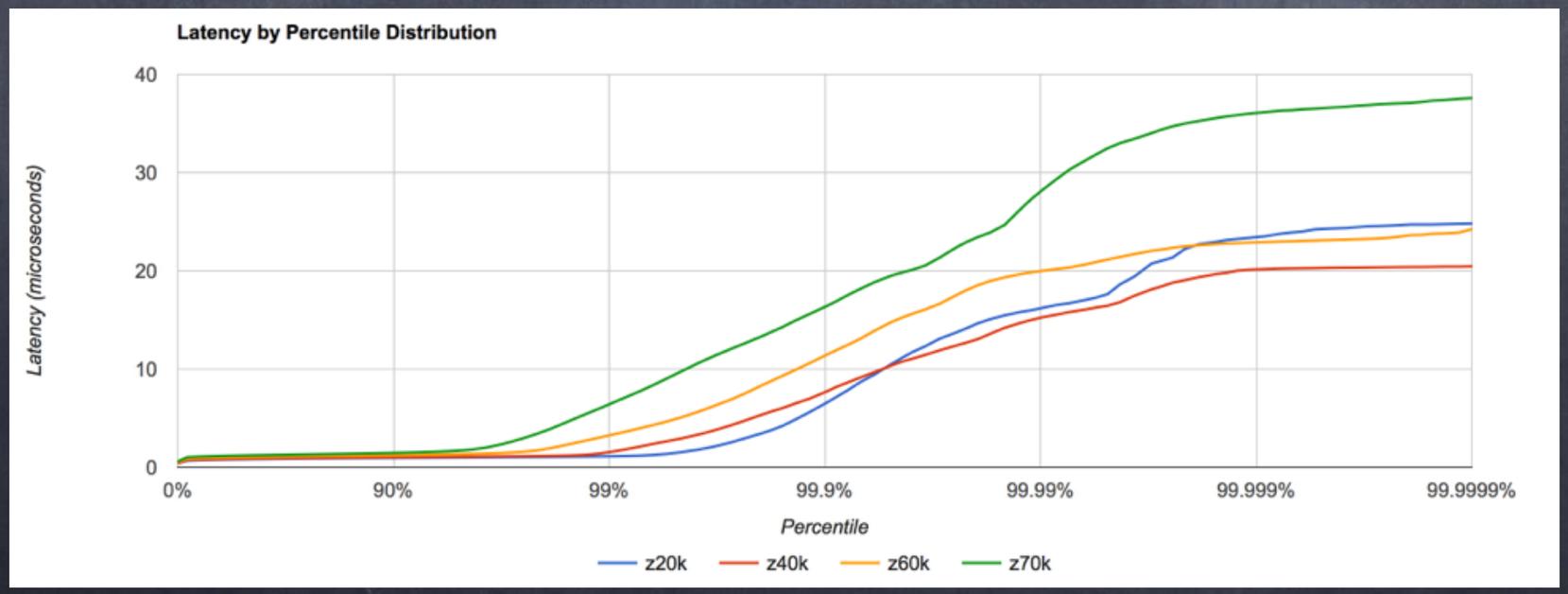
Zing @20k thru 70k



AZUL SYSTEMS

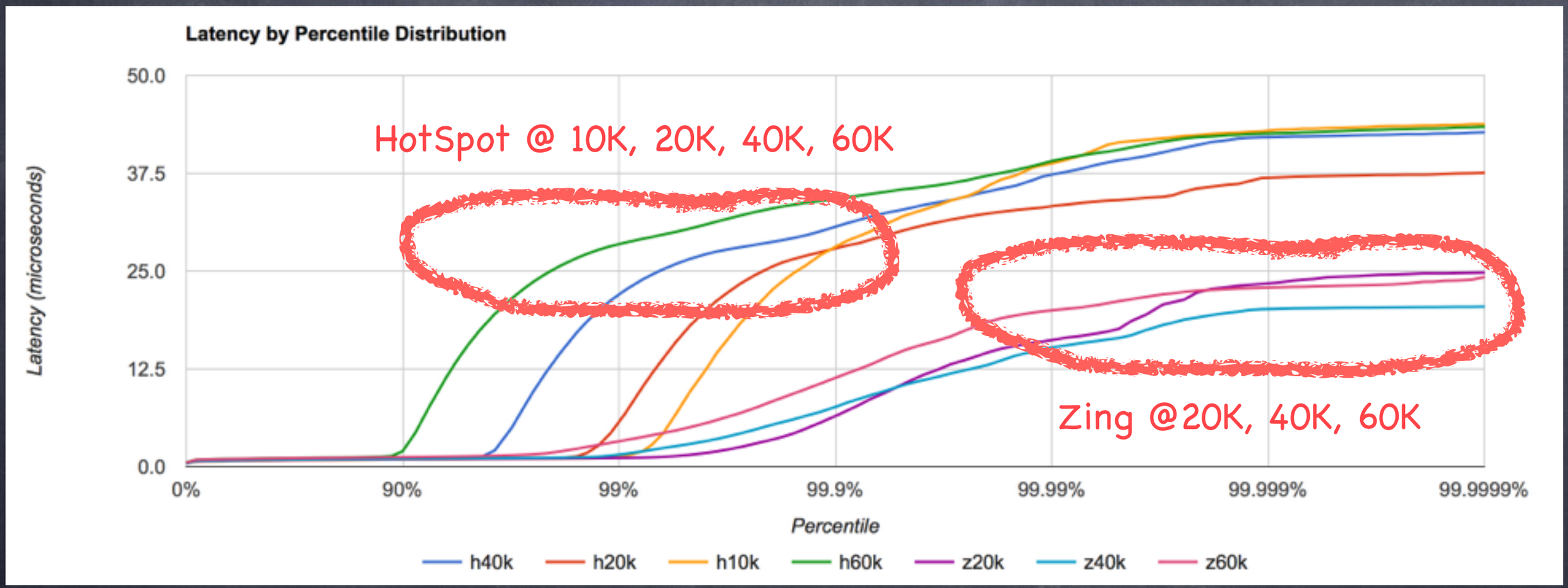
Zing & HotSpot @2k thru 70k





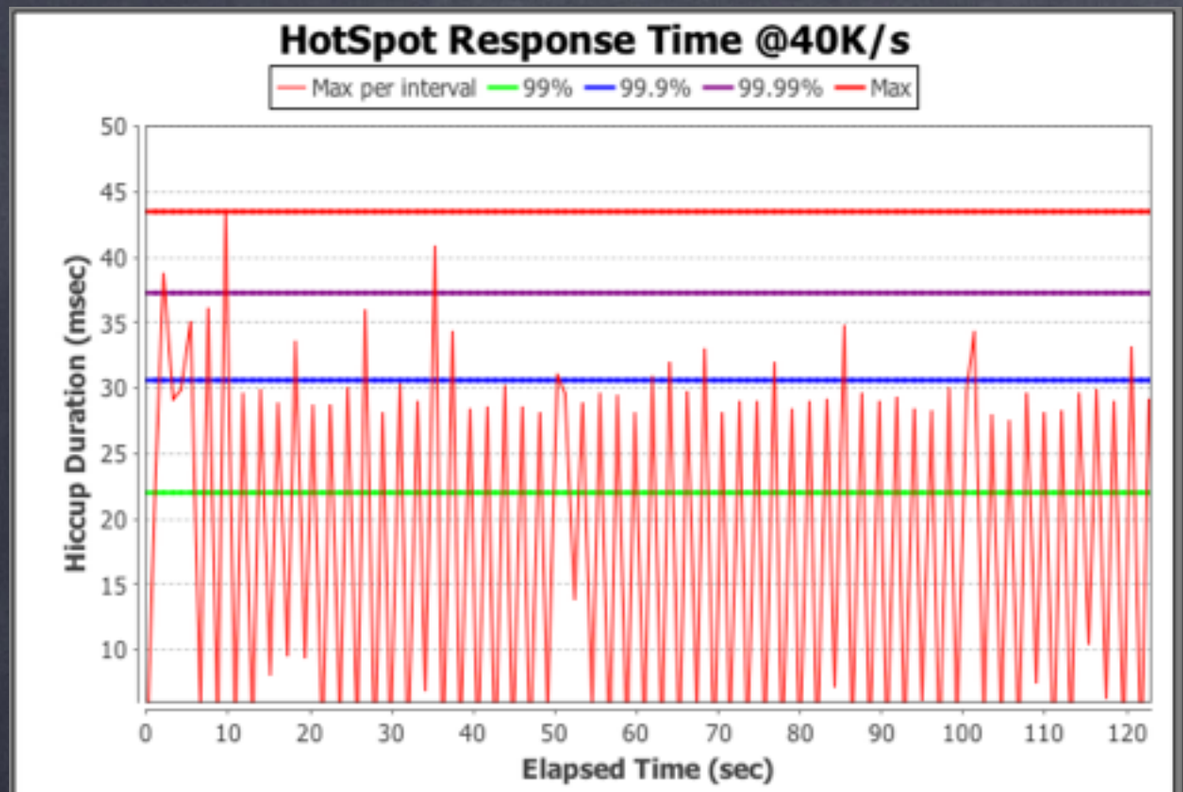
AZUL SYSTEMS

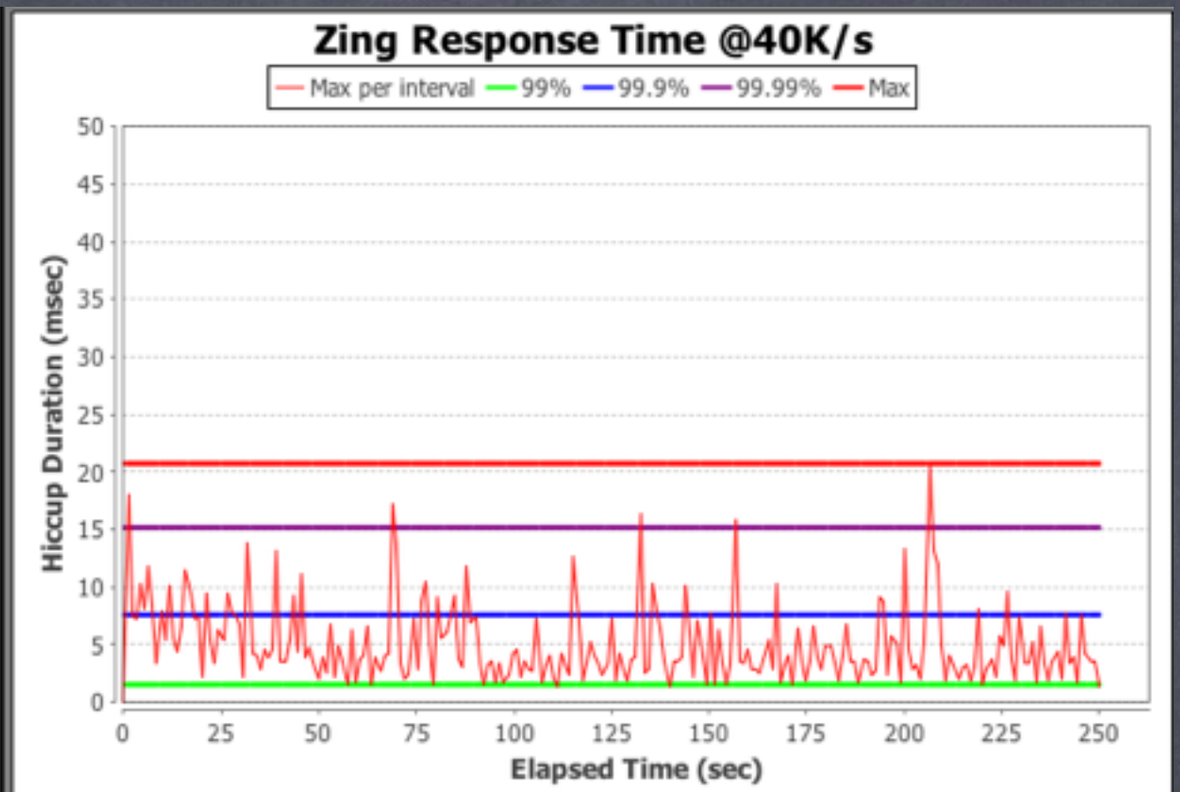
Zing & HotSpot, 10K/s thru 60K/s

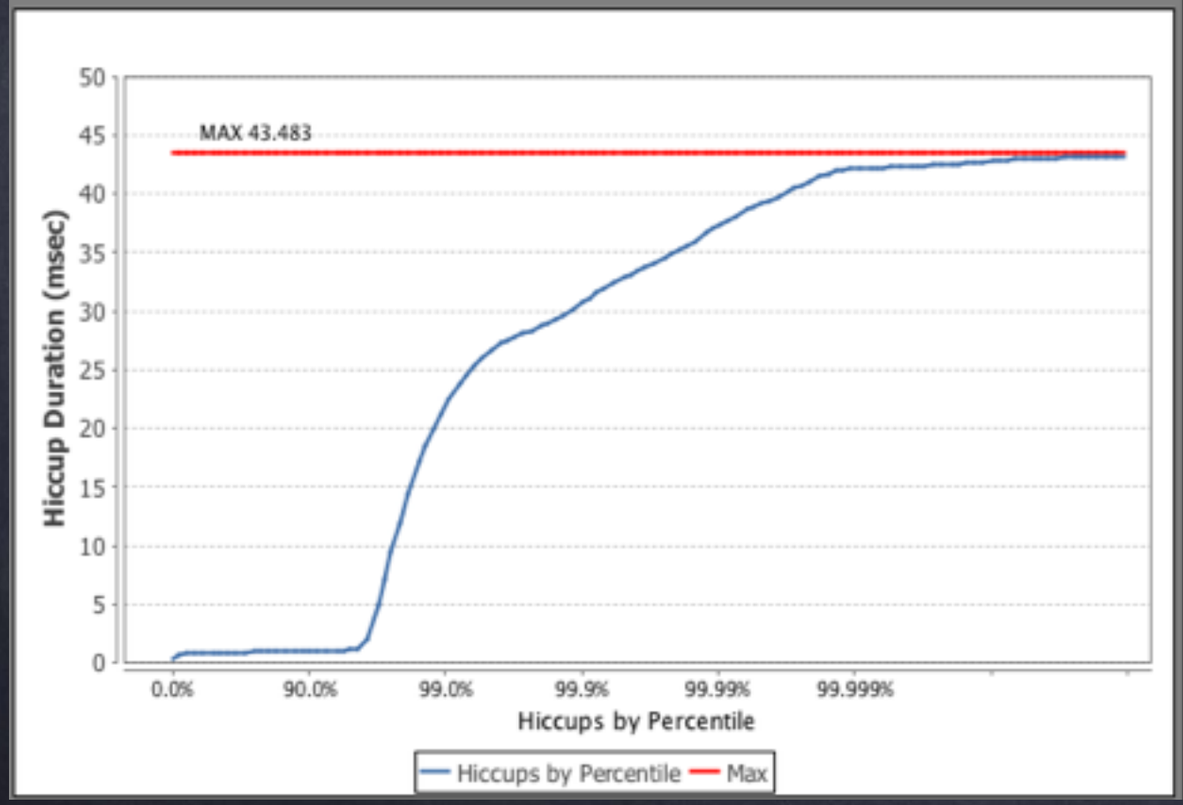


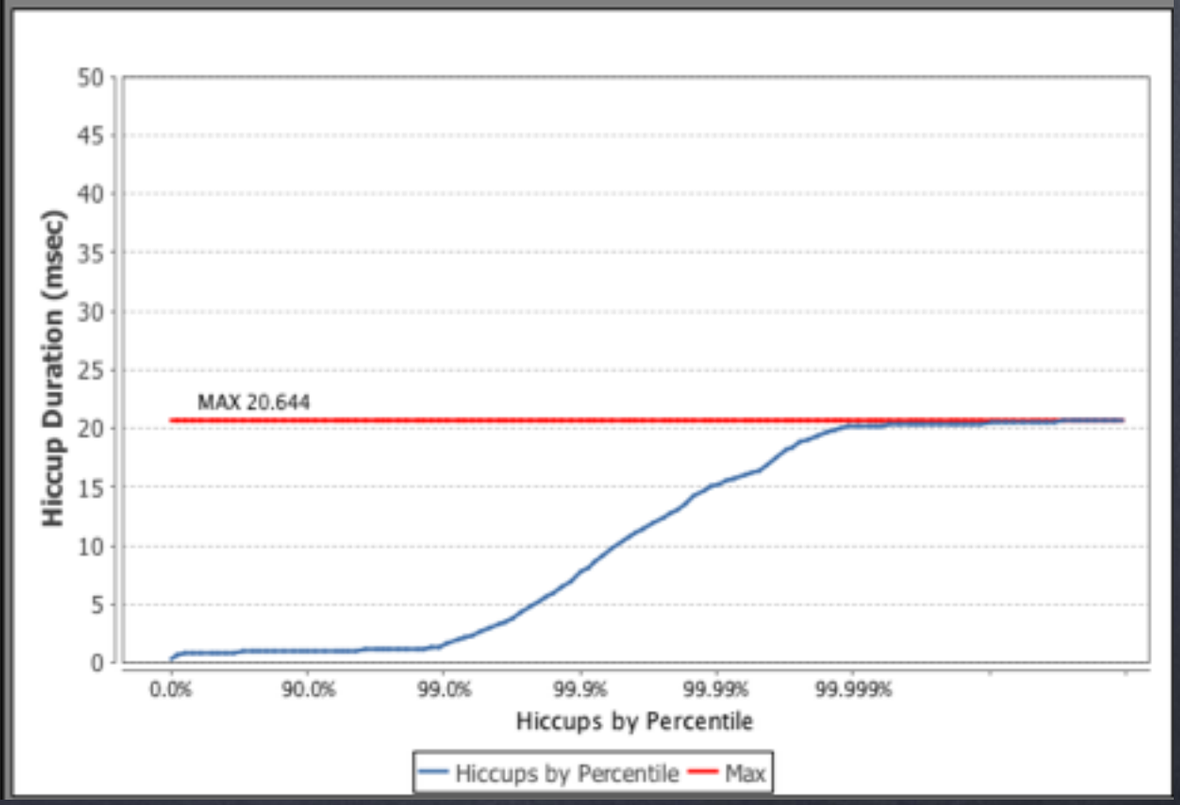
Lots of conclusions can be drawn from the above…
E.g. Zing delivers a consistent 100x reduction in the rate of occurrence of >20msec response times

AZUL SYSTEMS

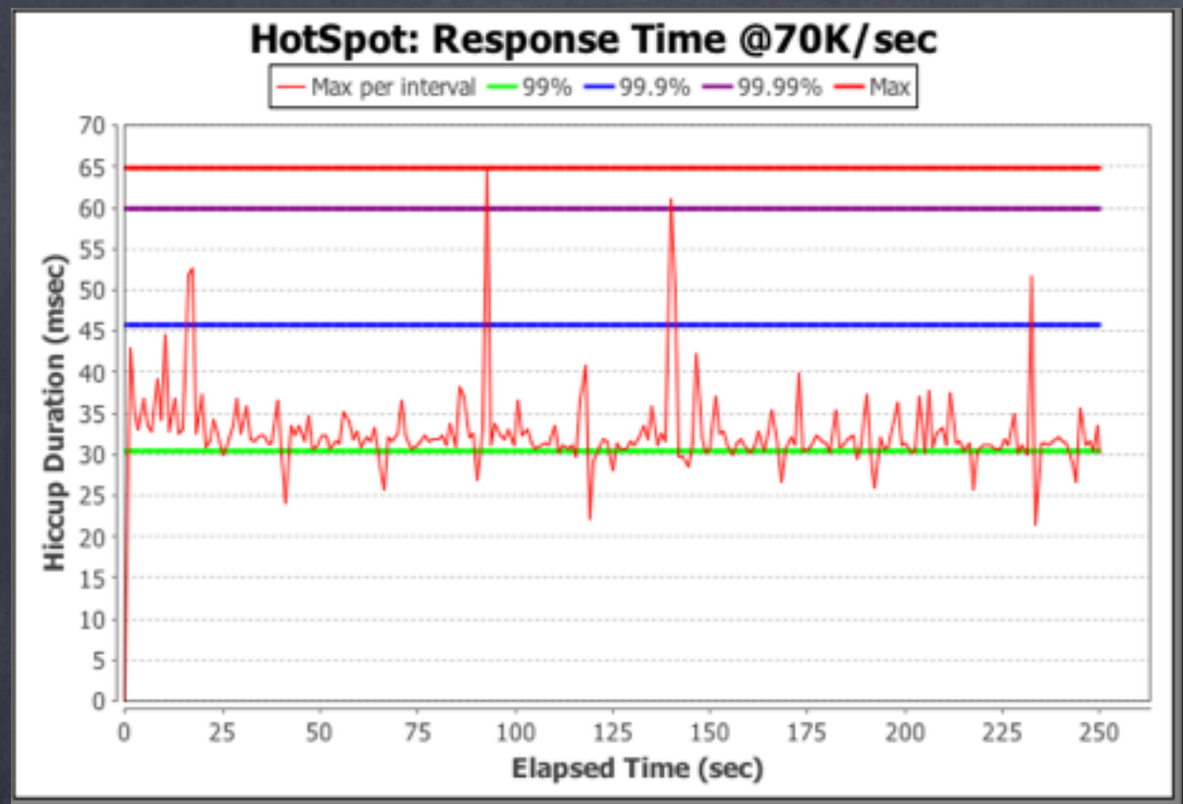


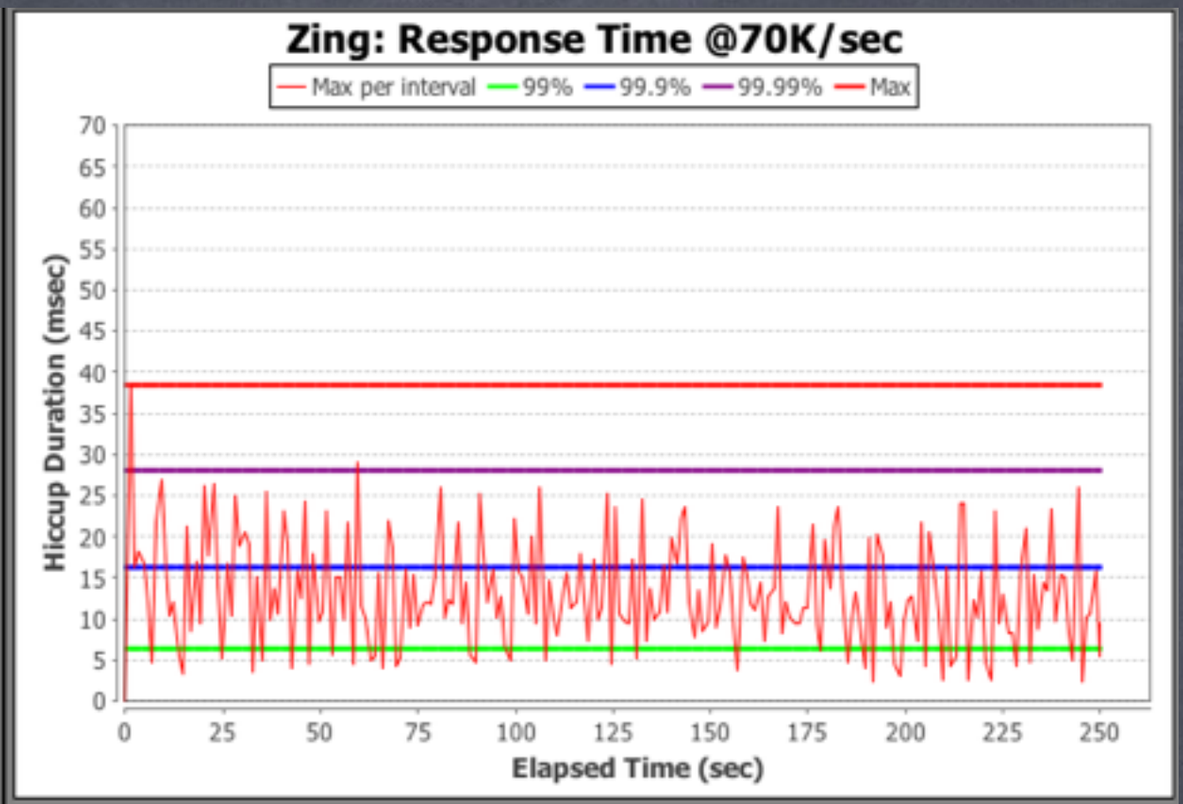


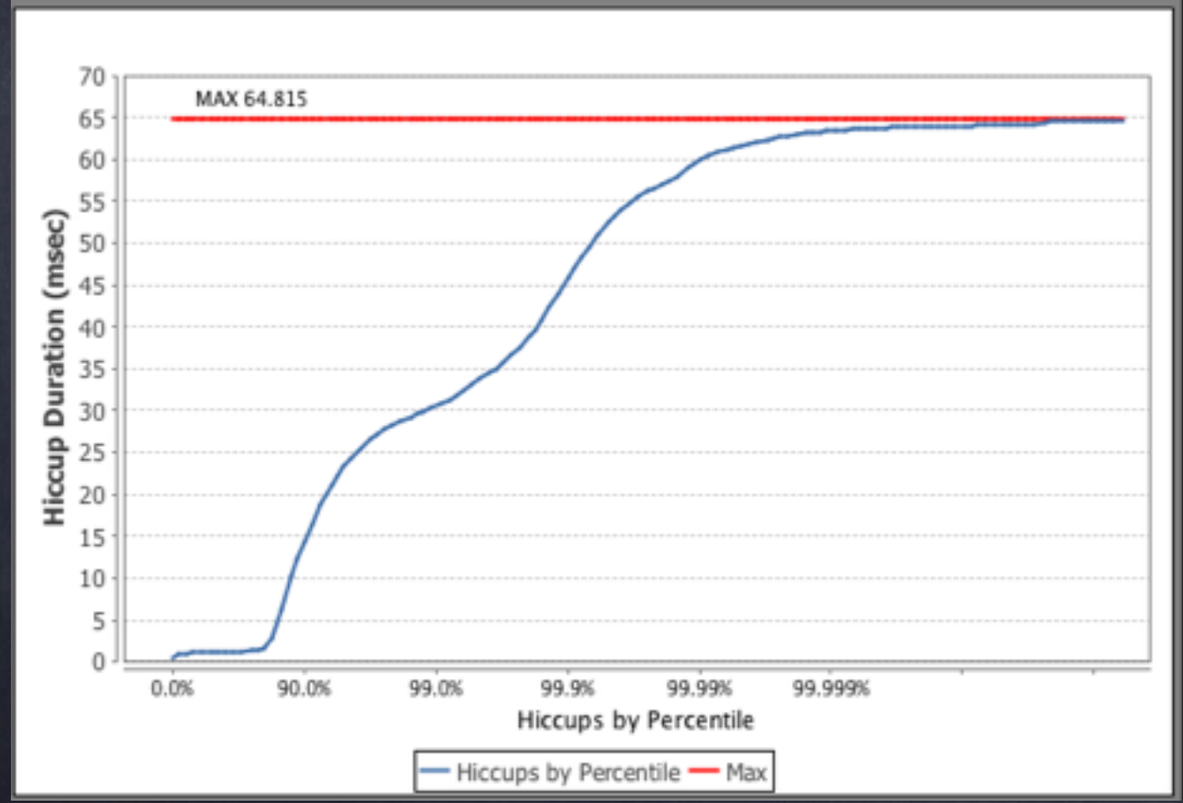


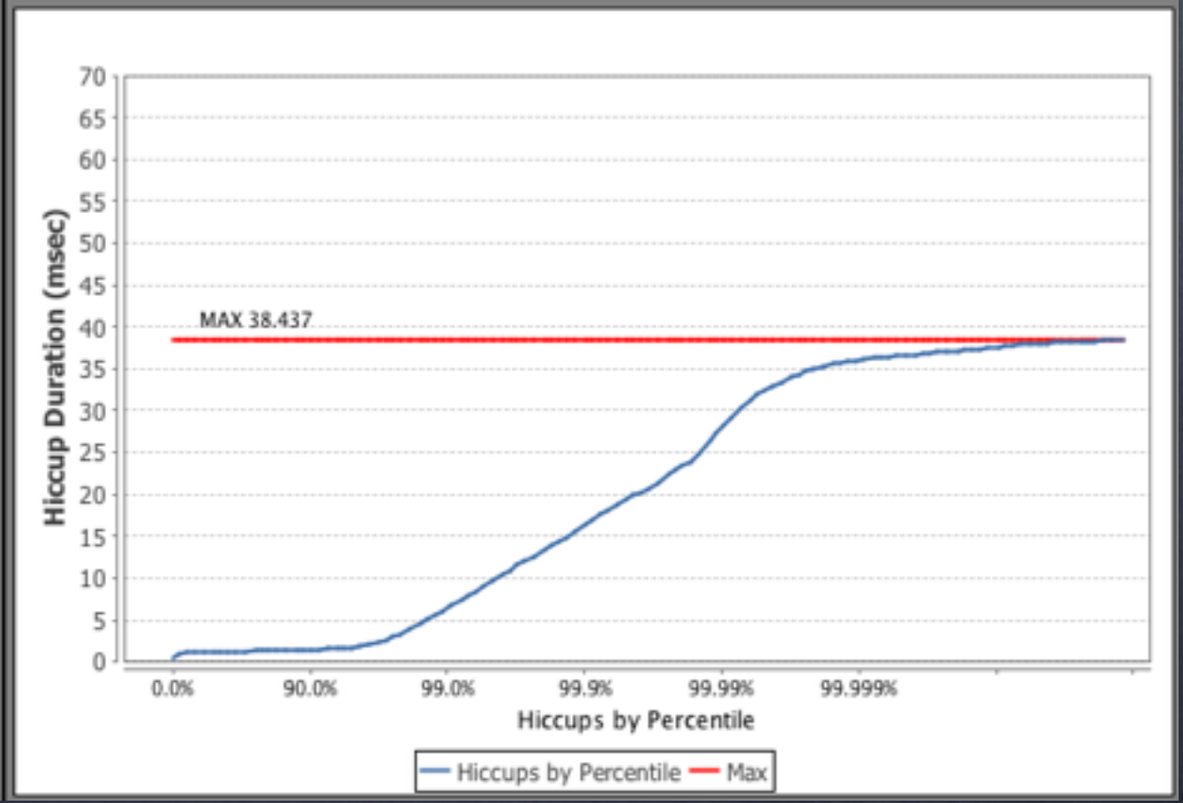


AZUL SYSTEMS

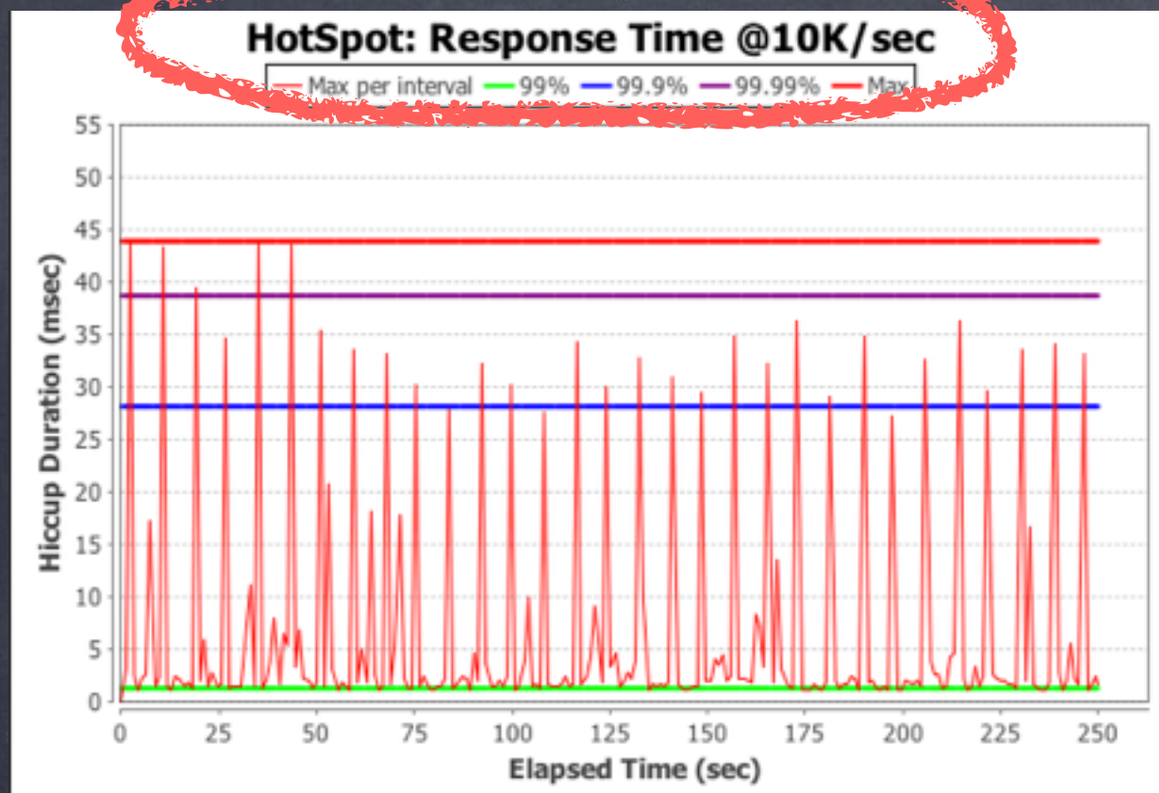


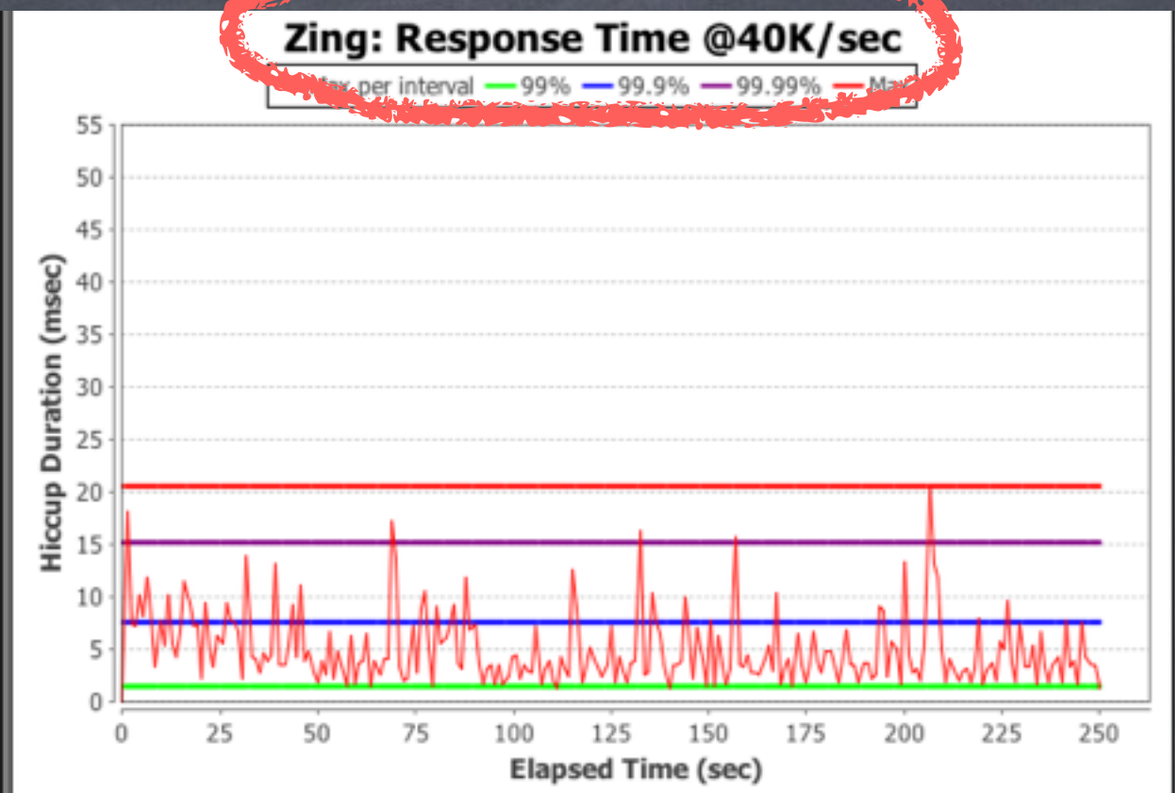


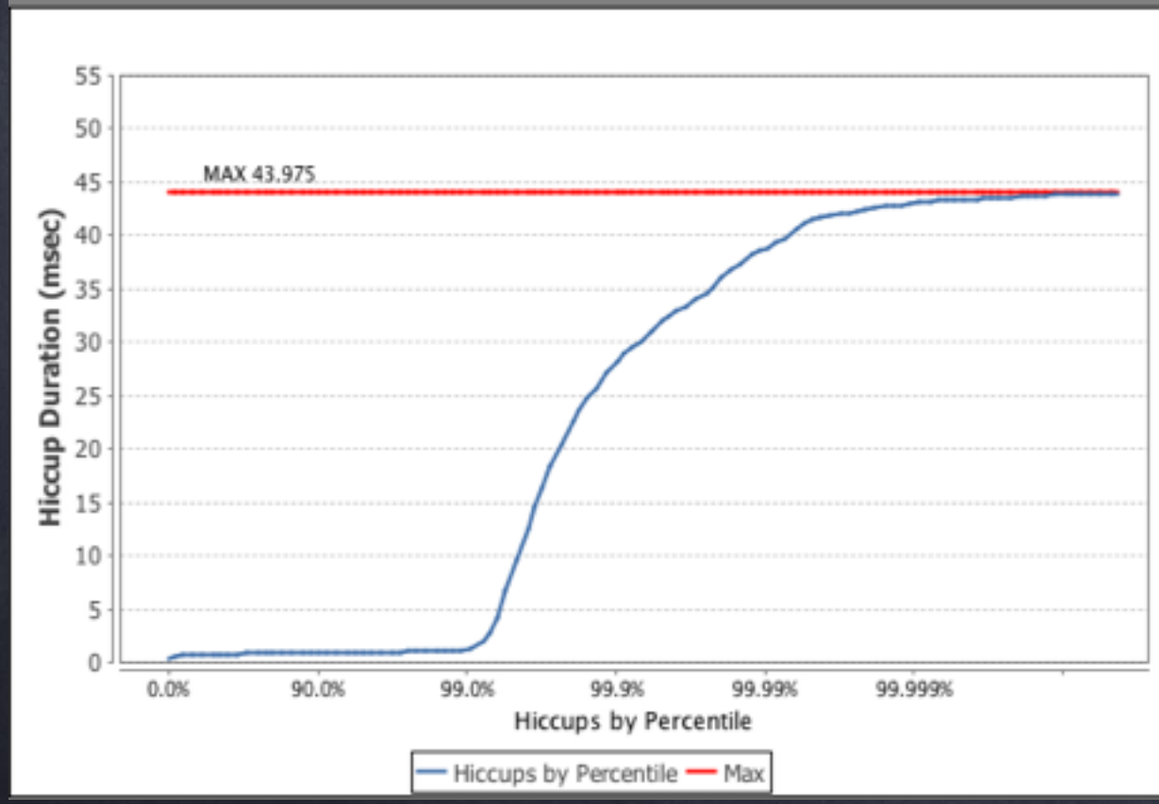


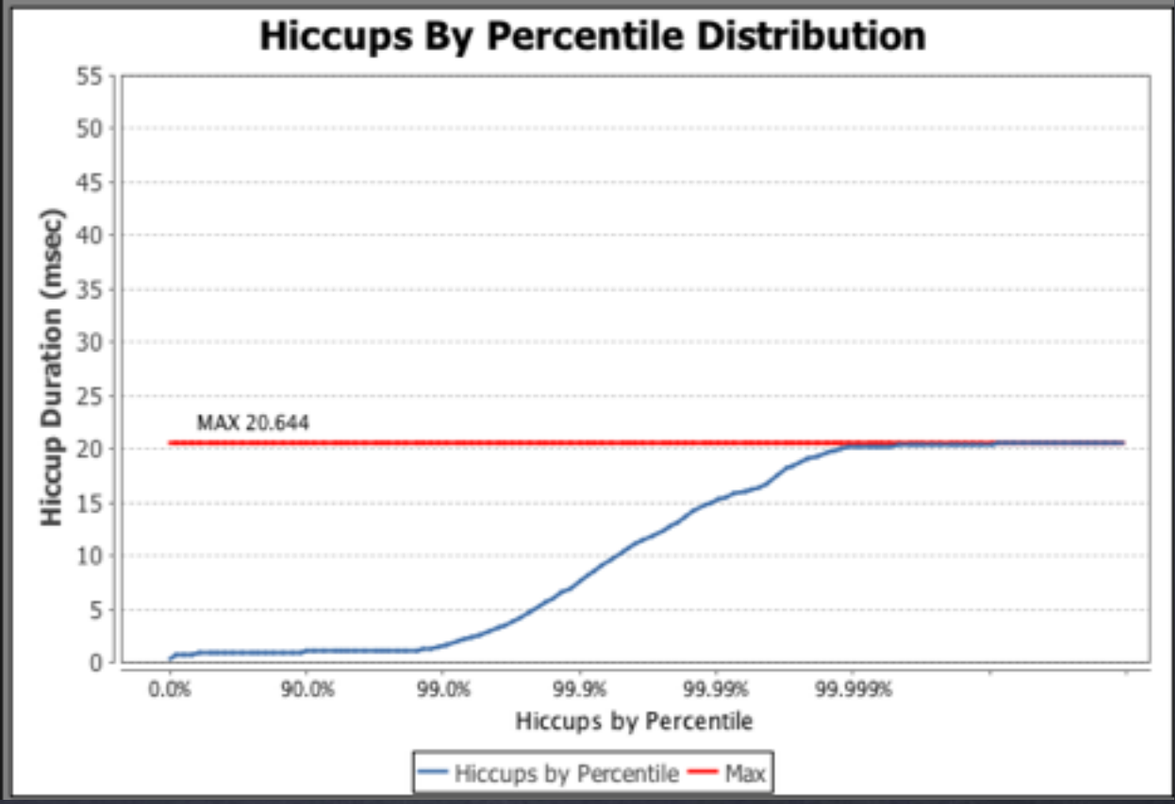


AZUL SYSTEMS

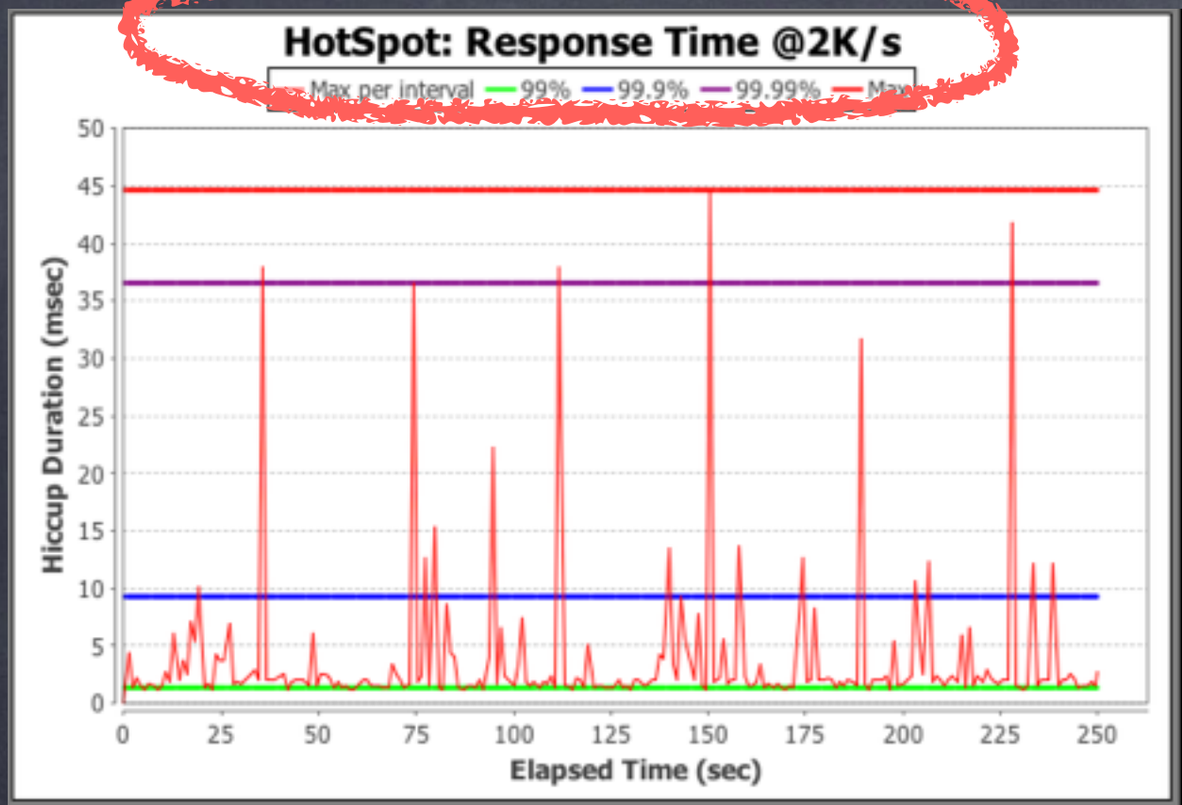


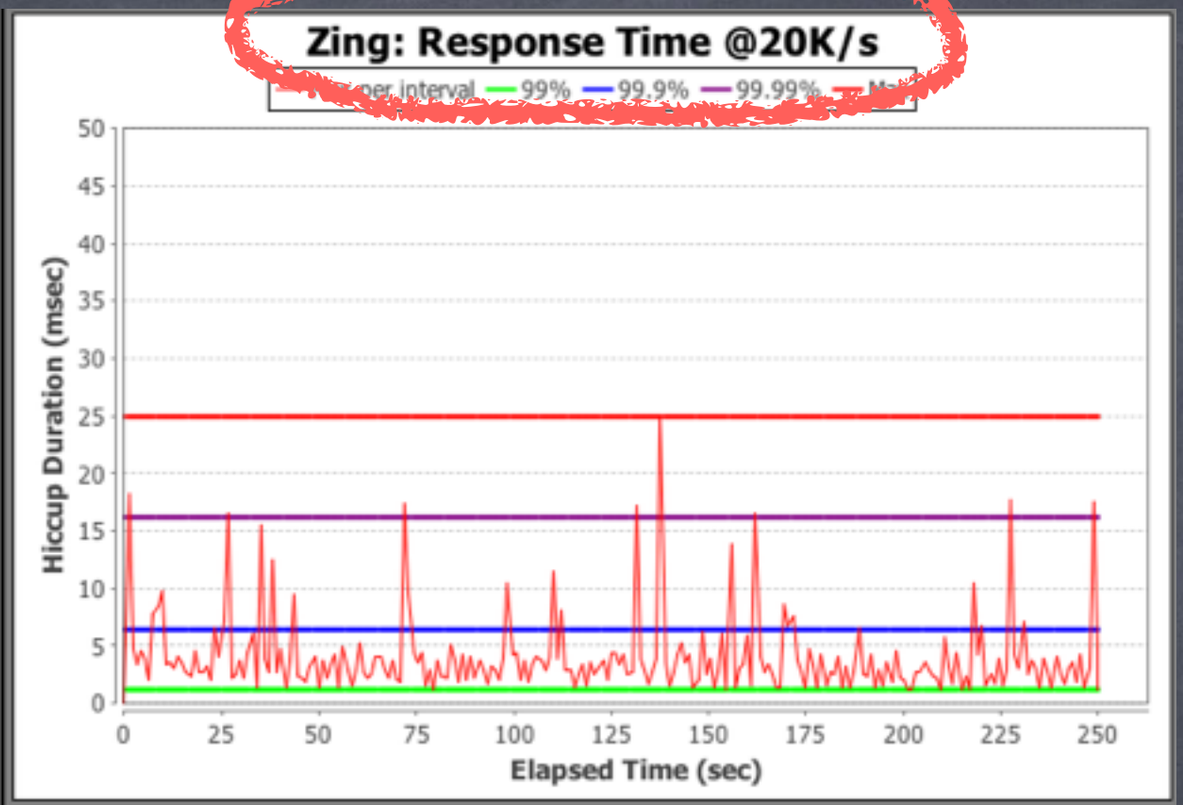


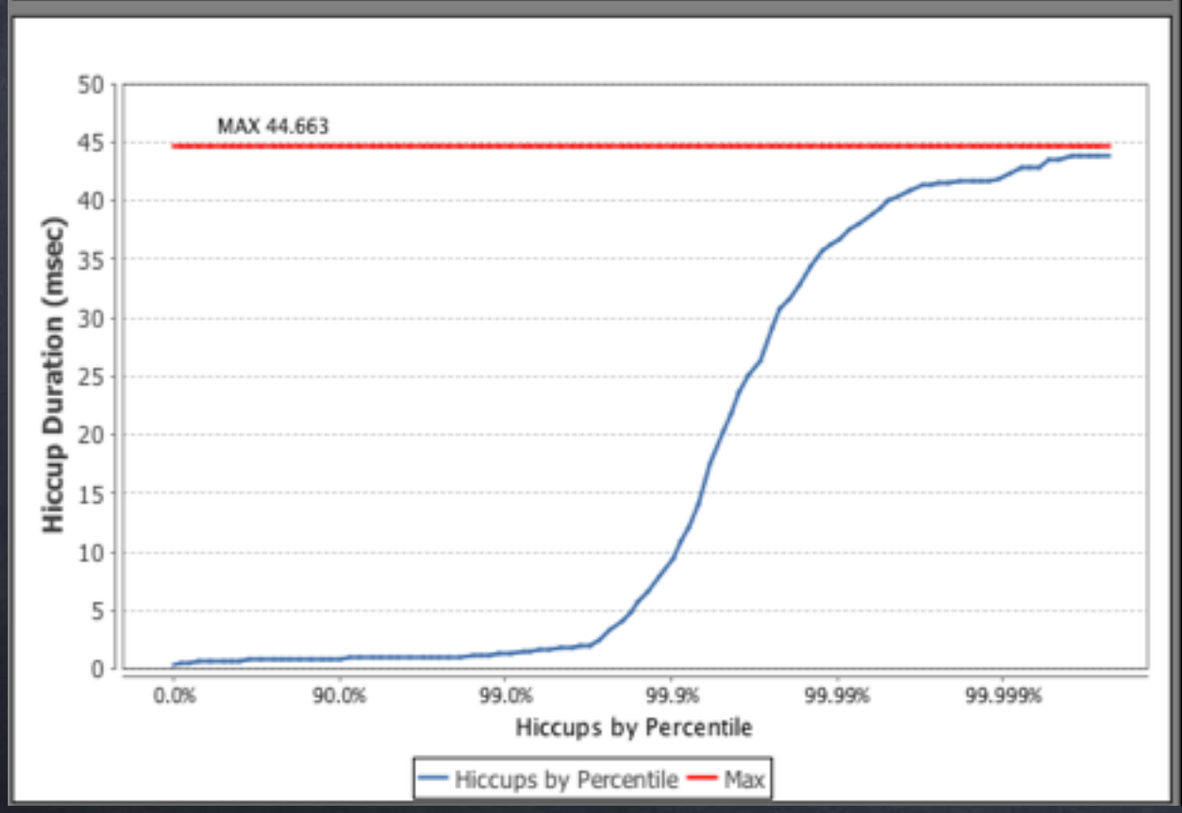


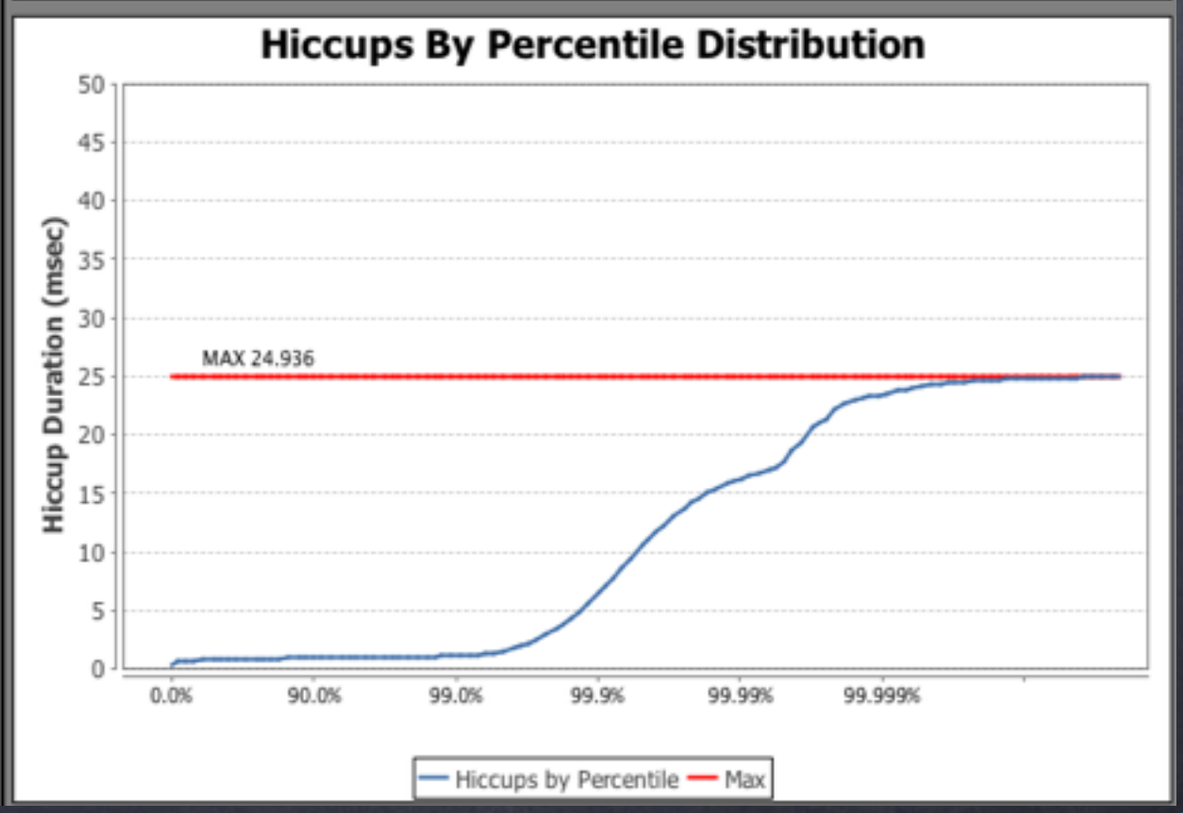


AZUL SYSTEMS



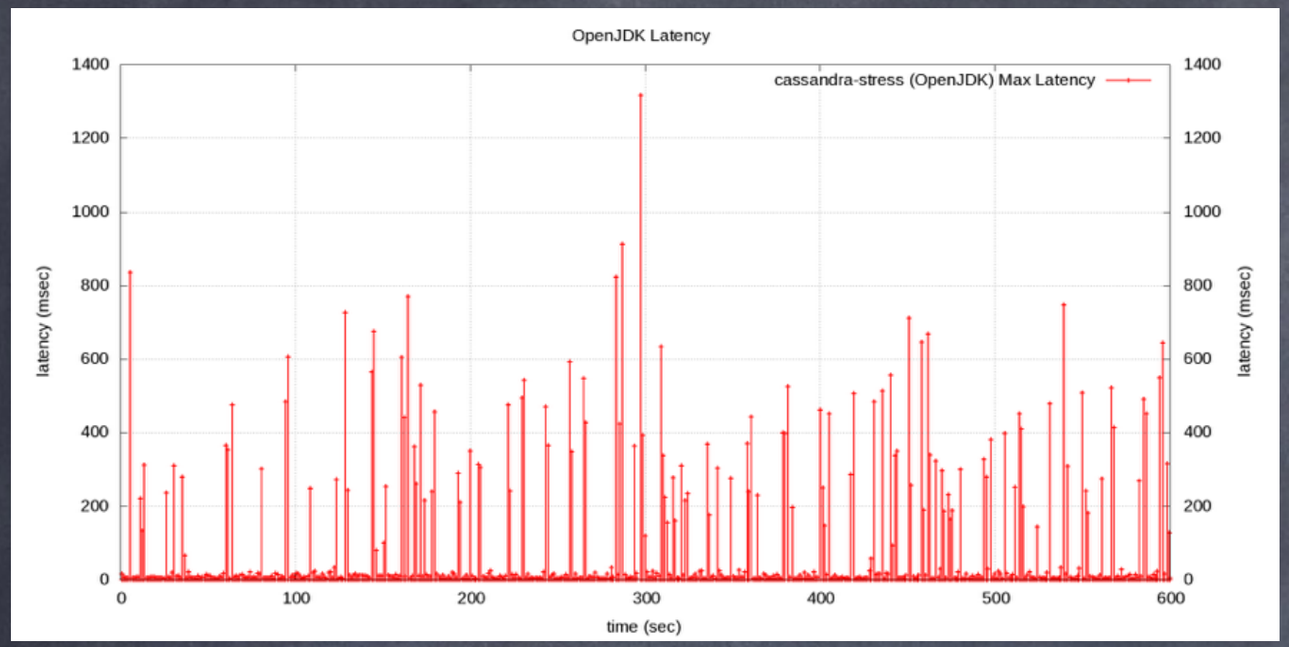




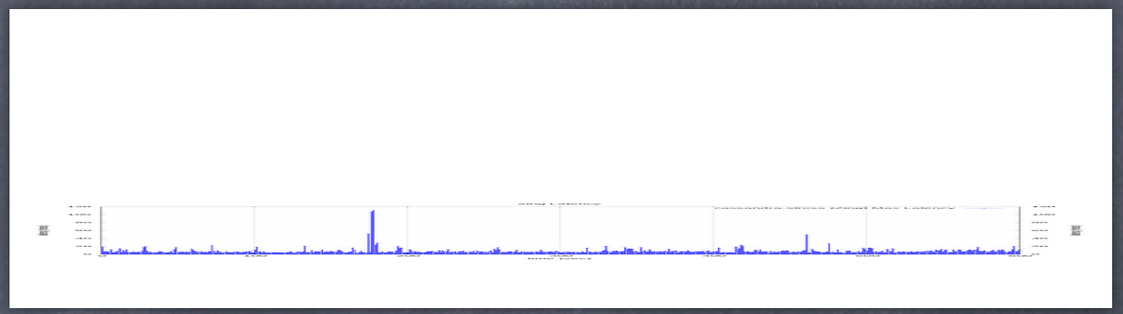


AZUL SYSTEMS

OpenJDK: 200-1400 msec stalls



Zing (drawn to scale)



```
op rate                   : 40001
partition rate            : 26996
row rate                  : 26996
latency mean              : 30.6 (0.7)
latency median            : 0.5 (0.5)
latency 95th percentile   : 244.4 (1.1)
latency 99th percentile   : 537.4 (2.0)
latency 99.9th percentile : 1052.2 (8.4)
latency max               : 1314.9 (1312.8)
```

Response Time Service time

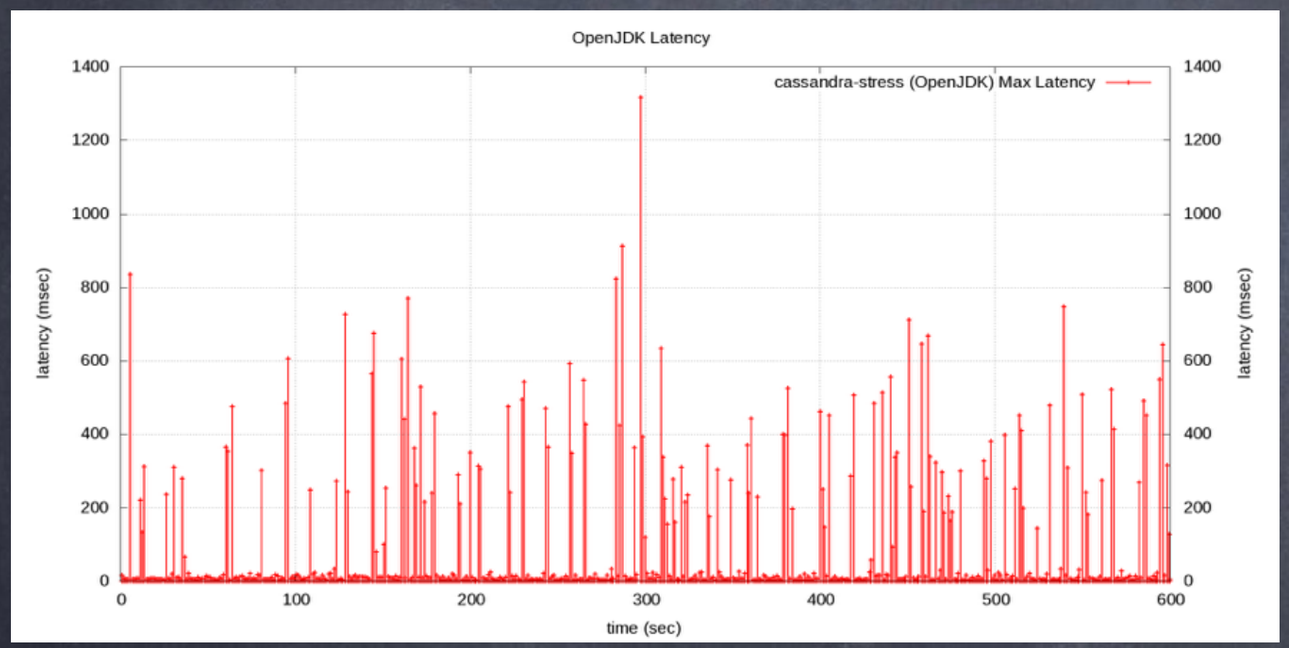
```
op rate                   : 40001
partition rate            : 26961
row rate                  : 26961
latency mean              : 0.6 (0.5)
latency median            : 0.5 (0.5)
latency 95th percentile   : 1.0 (0.9)
latency 99th percentile   : 2.7 (1.9)
latency 99.9th percentile : 13.3 (3.8)
latency max               : 110.6 (28.2)
```

Response Time Service time

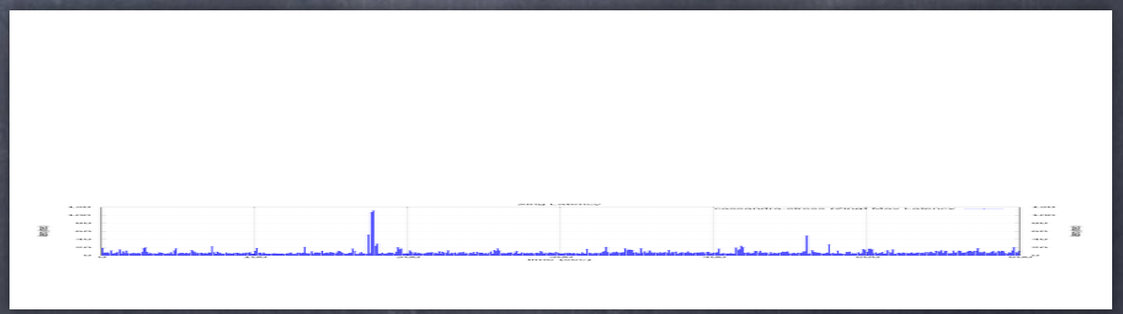
AZUL SYSTEMS

A simple visual summary

This is Cassandra on HotSpot



This is Cassandra on Zing



Any Questions?

AZUL SYSTEMS