How NOT to Measure Latency

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

The “Oh S@%#!” talk

Gil Tene, CTO & co-Founder, Azul Systems

@giltene

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 pulling the wool up from over their eyes...

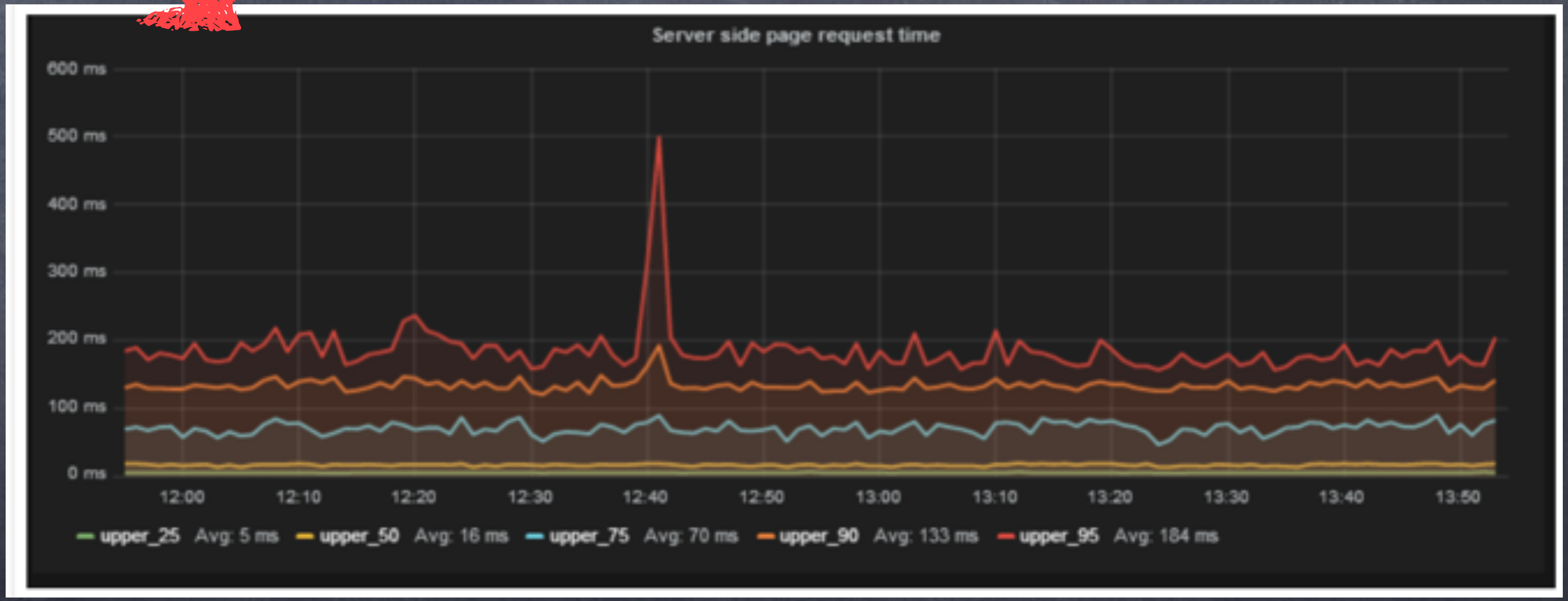
* working on real-world trash compaction issues, circa 2004

Latency Behavior

- Latency: The time it took one operation to happen
- Each operation occurrence has its own latency
- What we care about is how latency behaves
- Behavior is a lot more than "the common case was X"

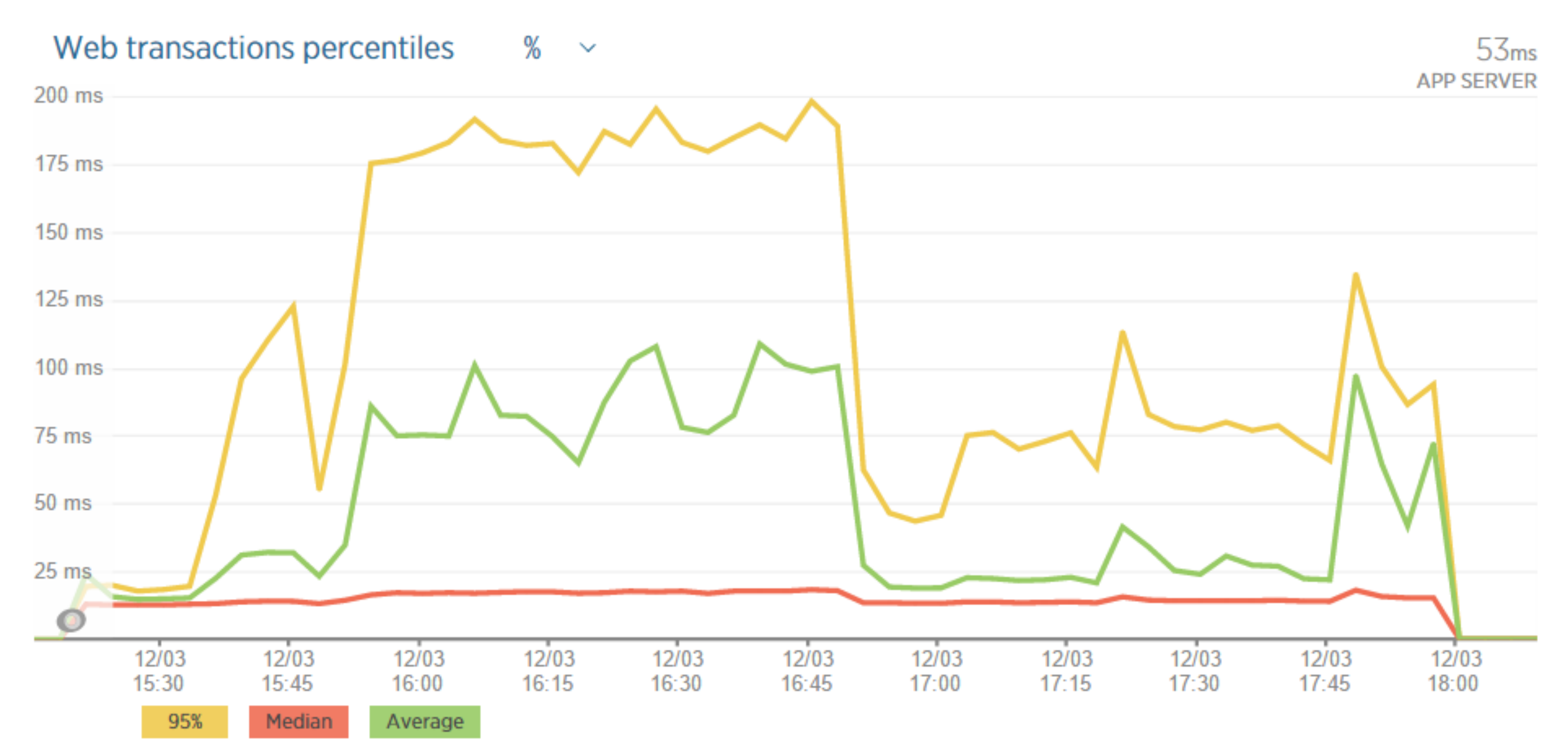
We like to look at pretty charts…

95%’lie

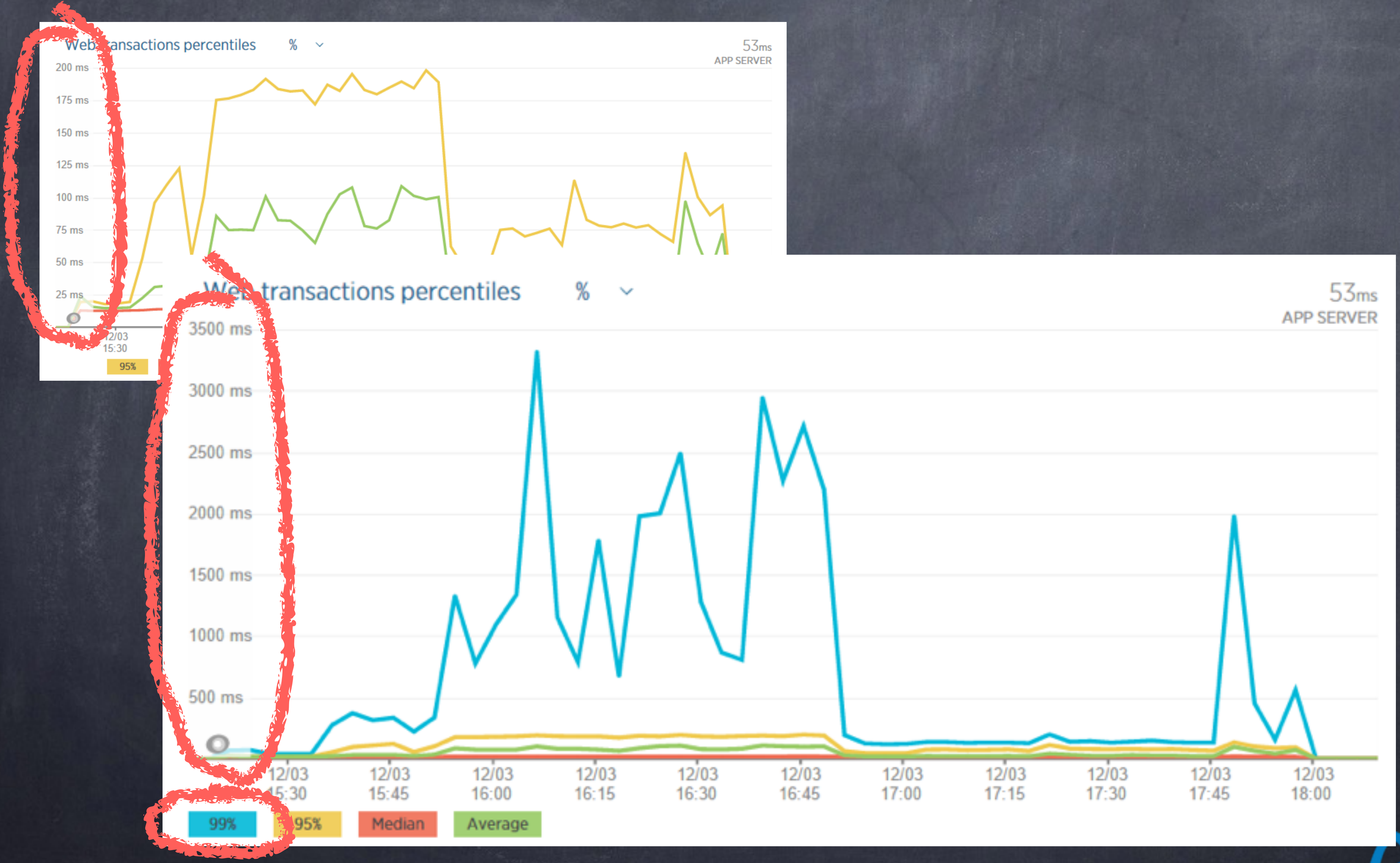


The “We only want to show good things” chart

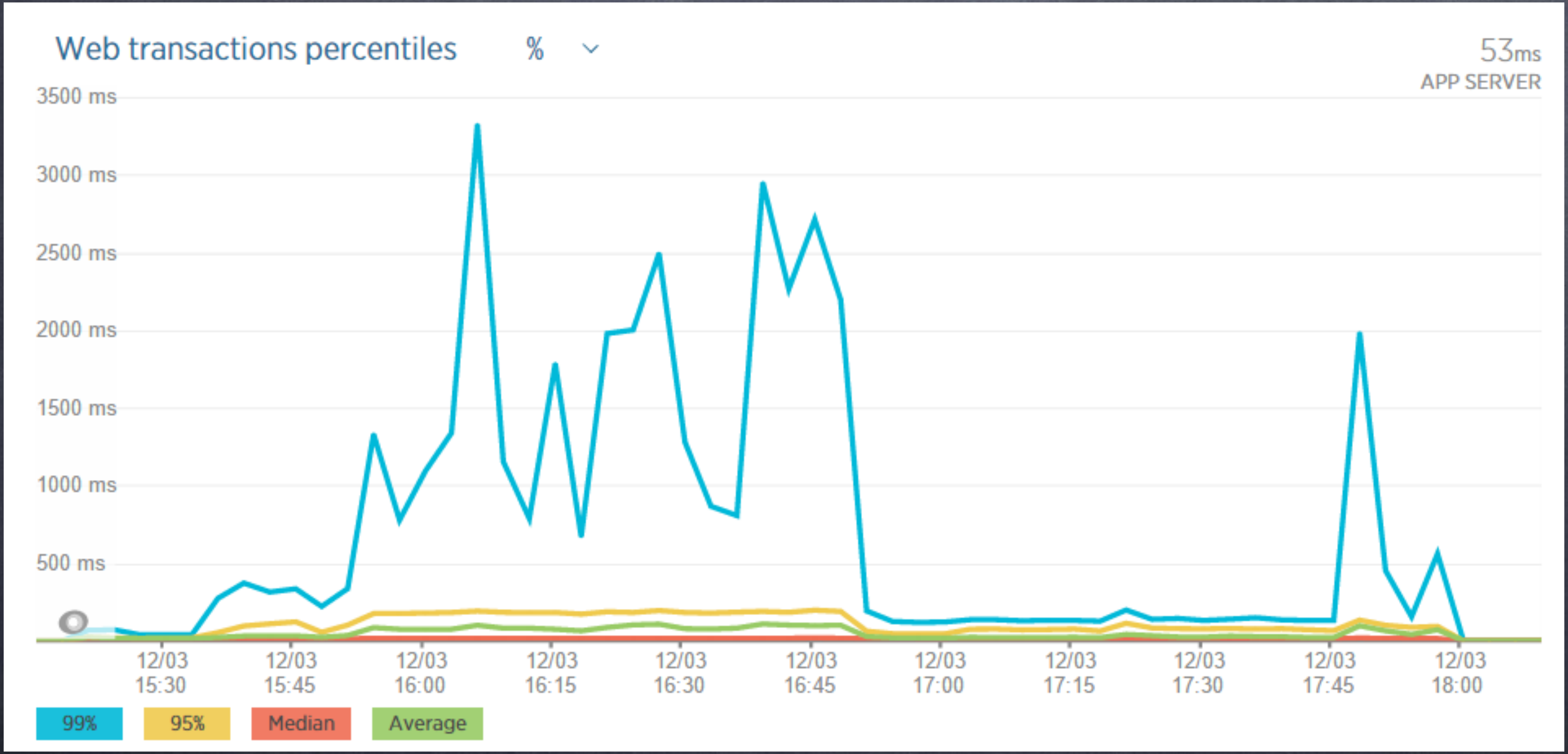
A real world, real time example



A real world, real time example

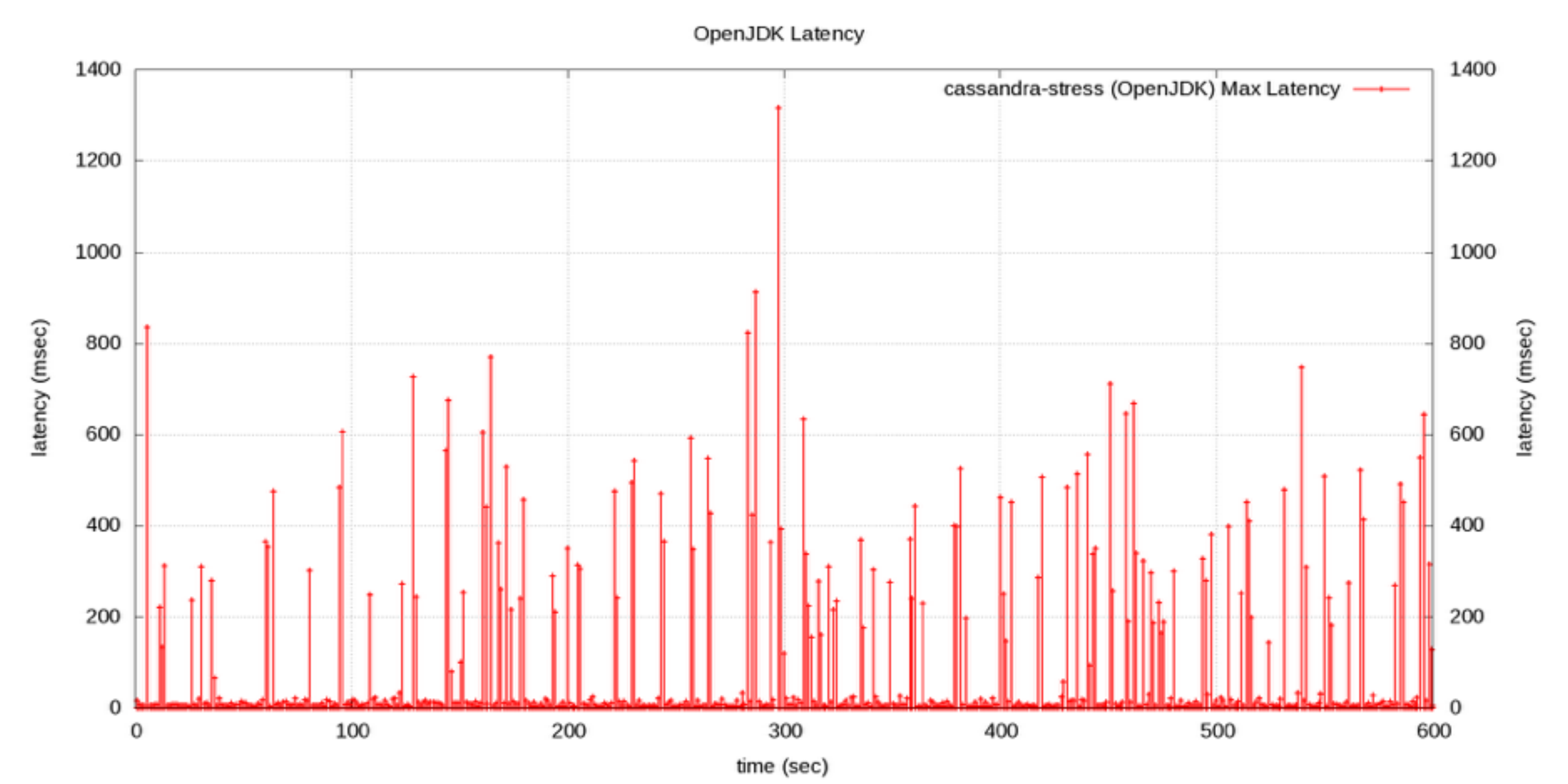


A real world, real time example



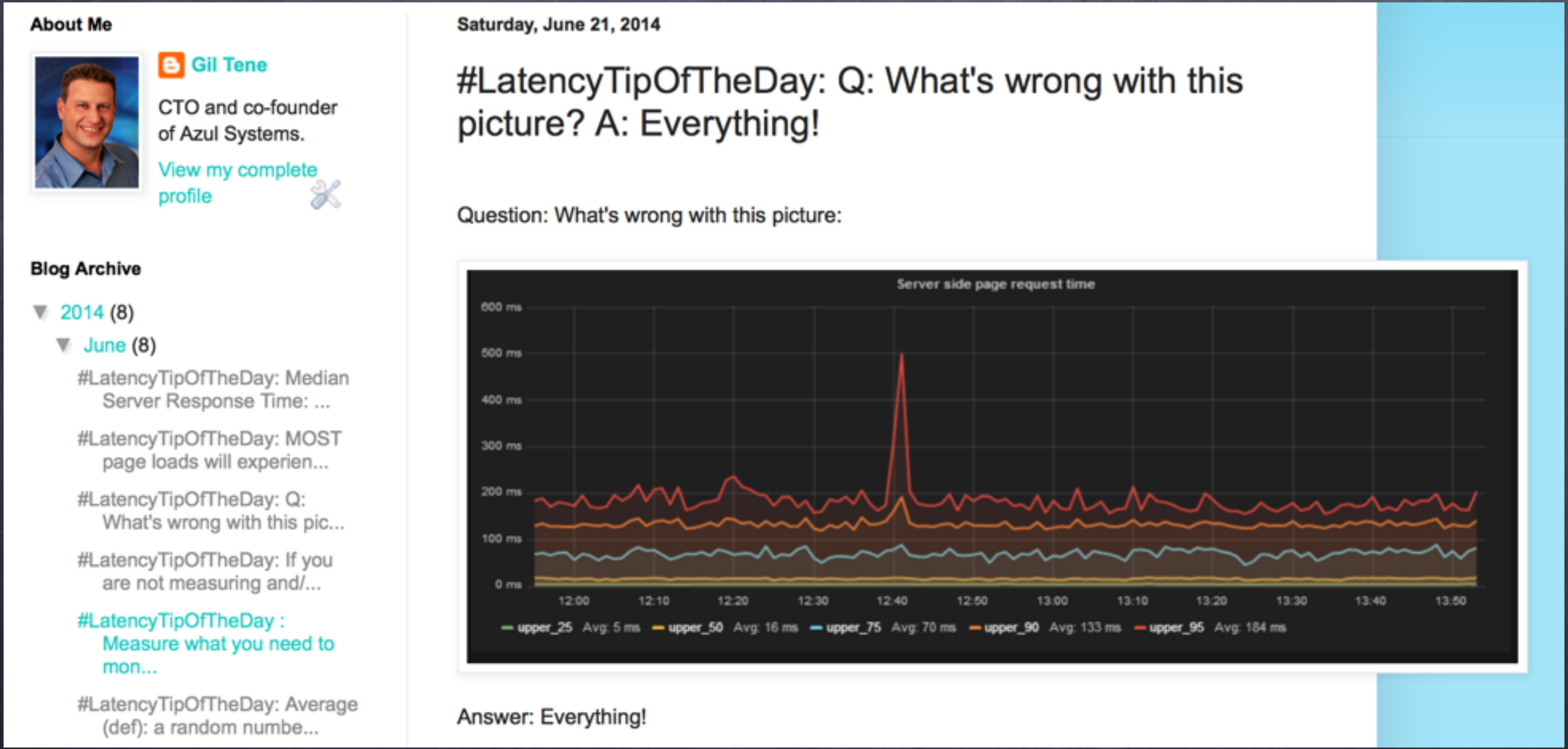
So this is a better picture. Right?

Why do we tend to avoid plotting Max latency?



Because no other %'ile will be visible on the same chart…

I like to rant about latency…



About Me

Gil Tene

CTO and co-founder of Azul Systems.

View my complete profile

Blog Archive

▼ 2014 (8)

▼ June (8)

#LatencyTipOfTheDay: Median Server Response Time: ...

#LatencyTipOfTheDay: MOST page loads will experien...

#LatencyTipOfTheDay: Q: What's wrong with this pic...

#LatencyTipOfTheDay: If you are not measuring and/...

#LatencyTipOfTheDay : Measure what you need to mon...

#LatencyTipOfTheDay: Average (def): a random numbe...

Saturday, June 21, 2014

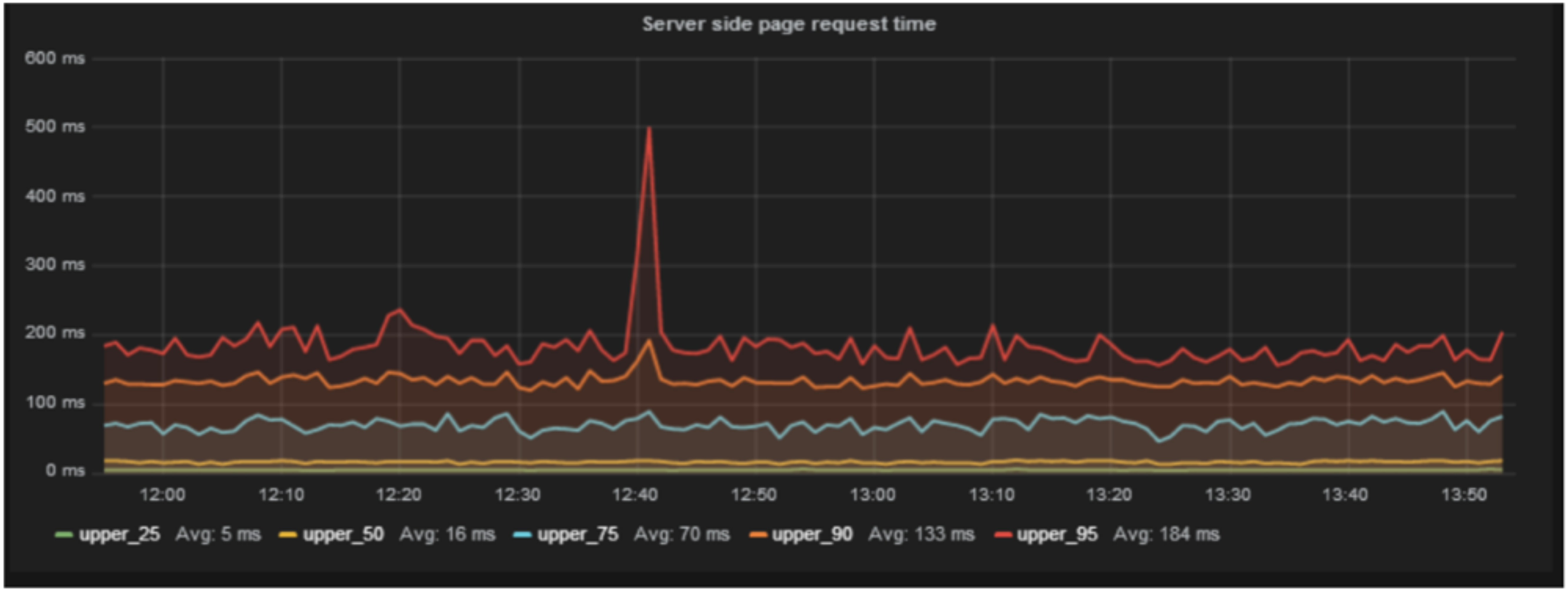
#LatencyTipOfTheDay: Q: What's wrong with this picture? A: Everything!

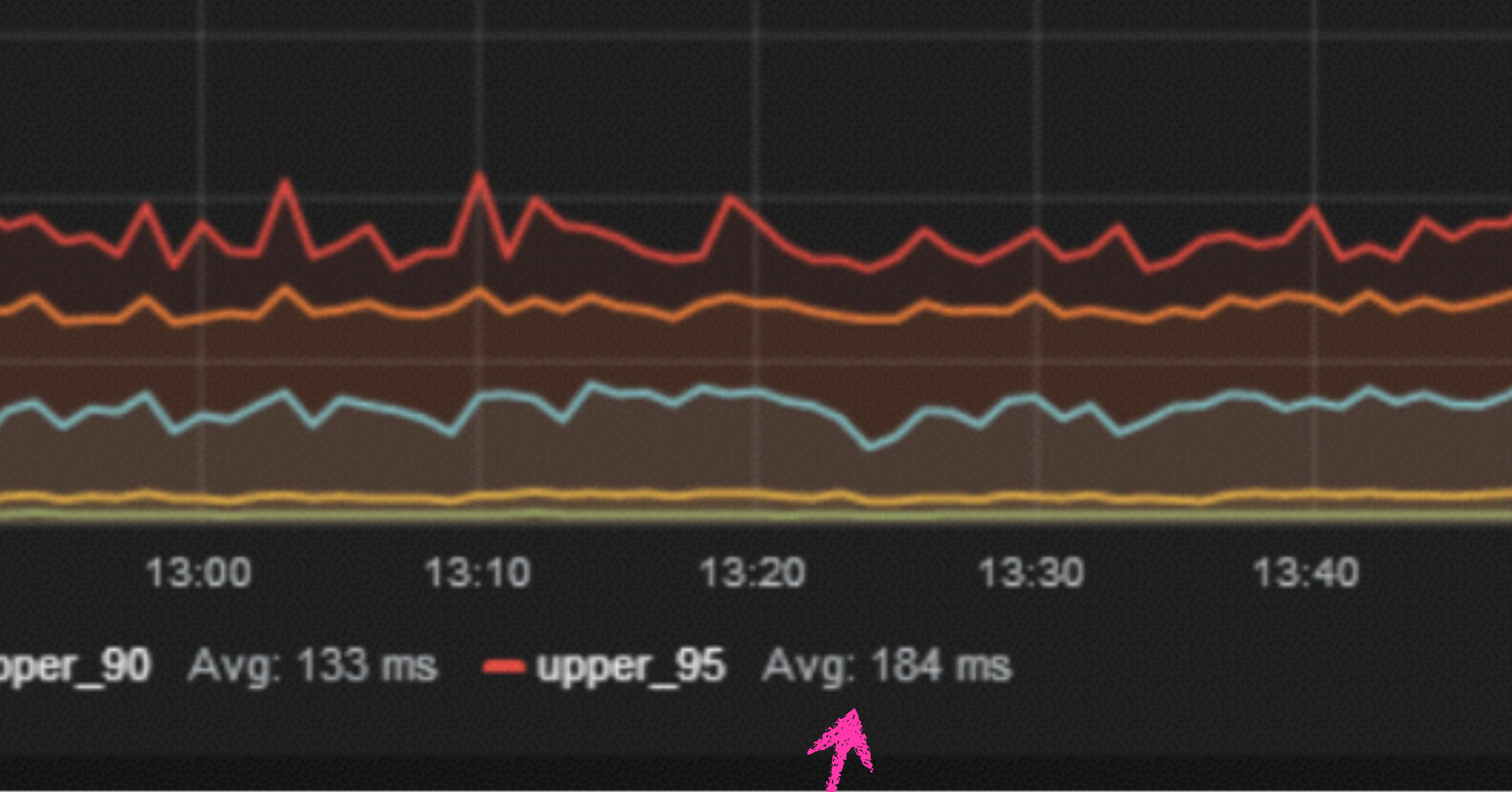
Question: What's wrong with this picture:

Answer: Everything!

#LatencyTipOfTheDay:

If you are not measuring and/or plotting Max, what are you hiding (from)?


Server side page request time
600 ms
500 ms
400 ms
300 ms
200 ms
100 ms
0 ms
12:00
12:10
12:20
12:30
12:40
12:50
13:00
13:10
13:20
13:30
13:40
13:50
upper_25 Avg: 5 ms
upper_50 Avg: 16 ms
upper_75 Avg: 70 ms
upper_90 Avg: 133 ms
upper_95 Avg: 184 ms



What (TF) does the Average of the 95%'lie mean?

What (TF) does the Average of the 95%'lie mean?

- Lets do the same with 100%'ile; Suppose we a set of 100%'ile values for each minute:

  [1, 0, 3, 1, 601, 4, 2, 8, 0, 3, 3, 1, 1, 0, 2]

  "The average 100%'ile over the past 15 minutes was 42"

- Same nonsense applies to any other %'lie

#LatencyTipOfTheDay:

You can't average percentiles.
Period.

Percentiles Matter

Is the 99%'lie "rare"?

99%'lie: a good indicator, right?

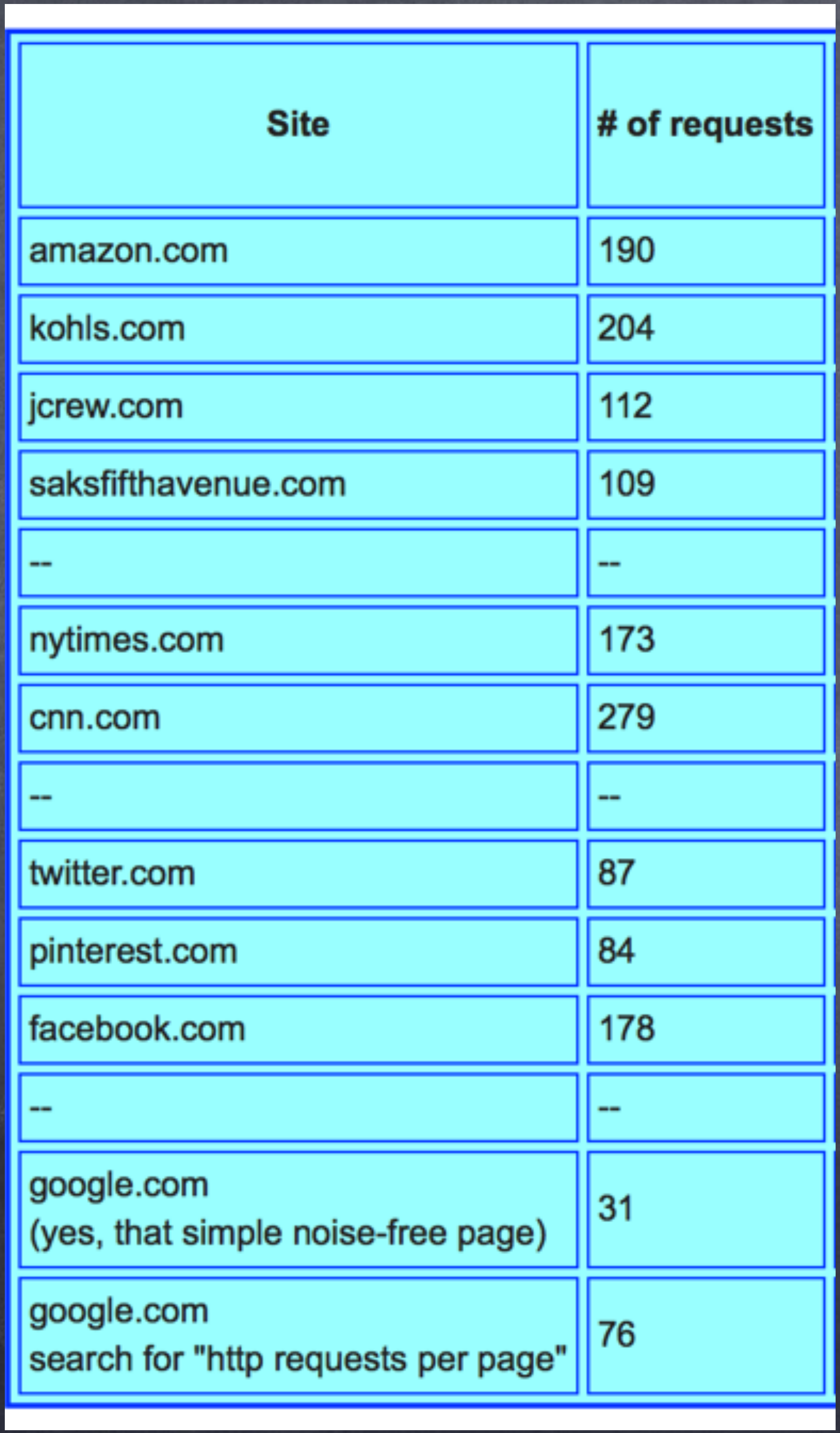
What are the chances of a single web page view experiencing >99%'lie latency of:

- A single search engine node?

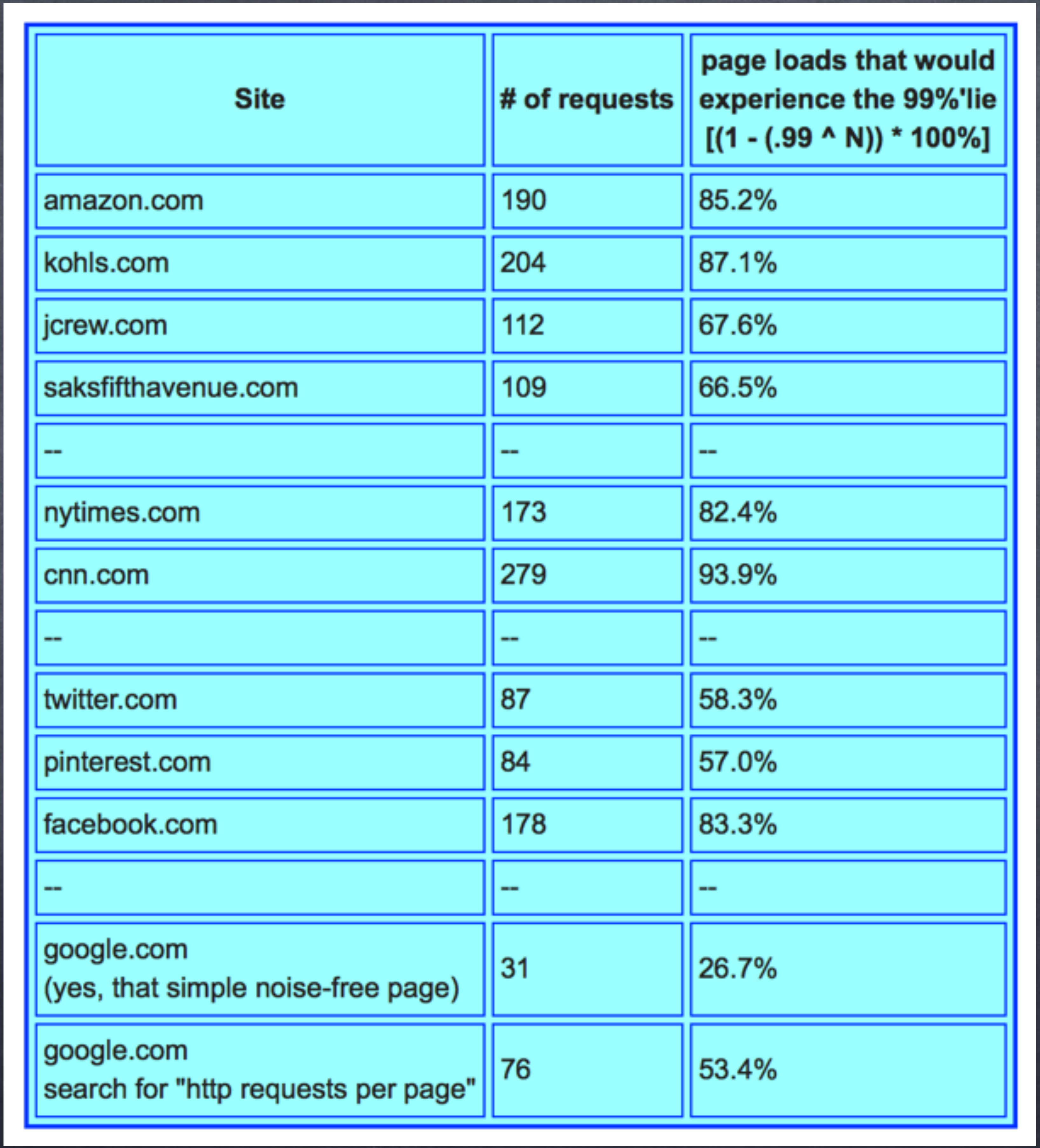
- A single Key/Value store node?

- A single Database node?

- A single CDN request?



Site	# of requests
amazon.com	190
kohls.com	204
jcrew.com	112
saksfifthavenue.com	109
--	--
nytimes.com	173
cnn.com	279
--	--
twitter.com	87
pinterest.com	84
facebook.com	178
--	--
google.com (yes, that simple noise-free page)	31
google.com search for "http requests per page"	76



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%

#LatencyTipOfTheDay:

MOST page loads will experience the 99%'lie server response

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

The 95%'lie   or   the 99.9%'lie

Gauging user experience

Example: If 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 of http requests?

Answer: ~0.003%

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

Answer: ~18%

Gauging user experience

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

- What http response percentile will be experienced by the 95%'ile of users?

Answer: ~99.97%

- What http response percentile will be experienced by the 99%'ile of users

Answer: ~99.995%

#LatencyTipOfTheDay:

Median Server Response Time:
The number that 99.9999999999%
of page views can be worse than

Why don’t we have response time or latency stats with multiple 9s in them???

Why don’t we have response time or latency stats with multiple 9s in them???

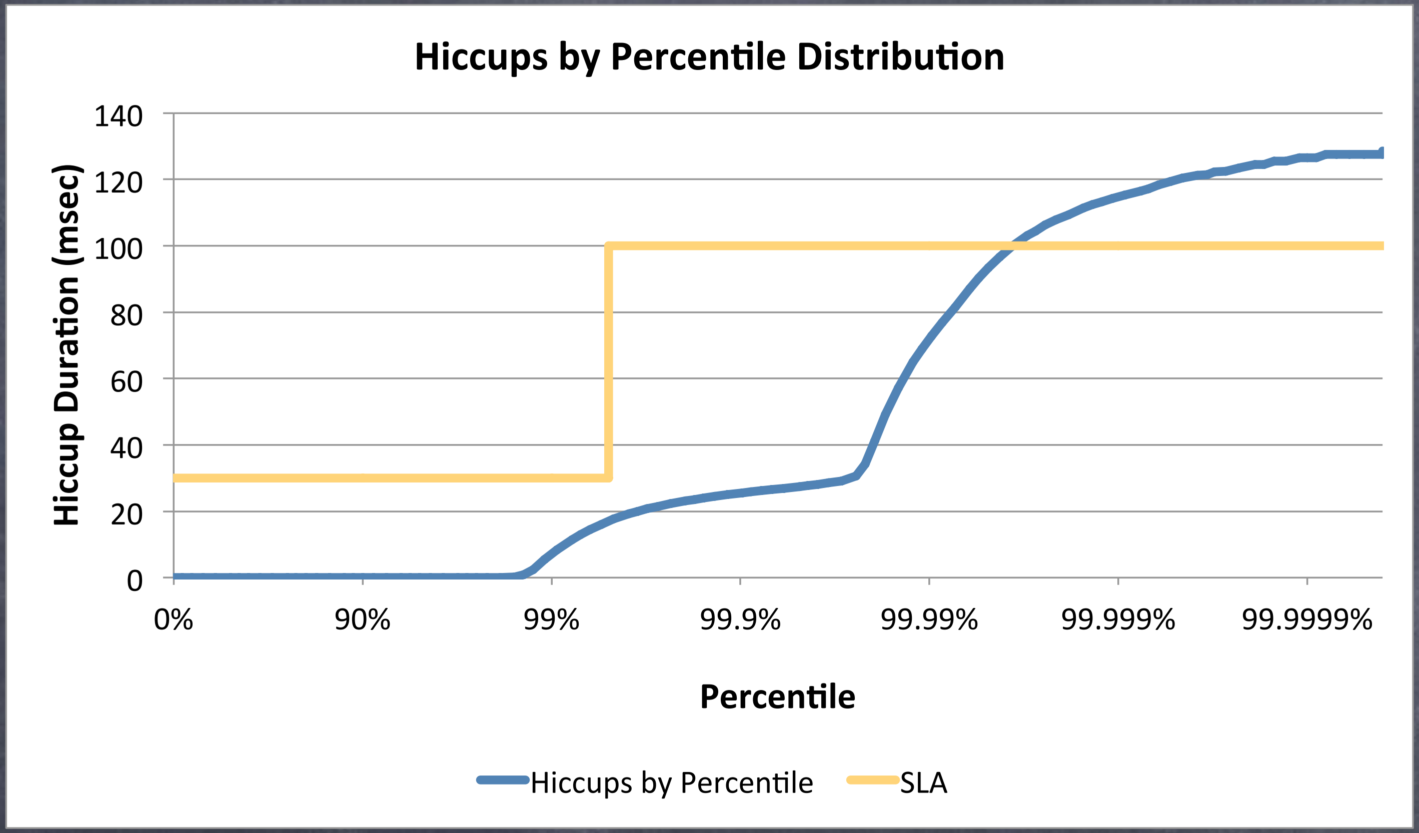
You can’t average percentiles…

And you also can’t get an hour’s 99.999%’lie out of lots of 10 second interval 99%’lie reports…

Why don’t we have response
time or latency stats with
multiple 9s in them???

Check out HdrHistogram

You can’t average percentiles…



It lets you have nice things….

And you also can’t get an hour’s
99.999%’lie out of lots
of 10 second interval 99%’lie reports…

AZUL
SYSTEMS

Latency "wishful thinking"

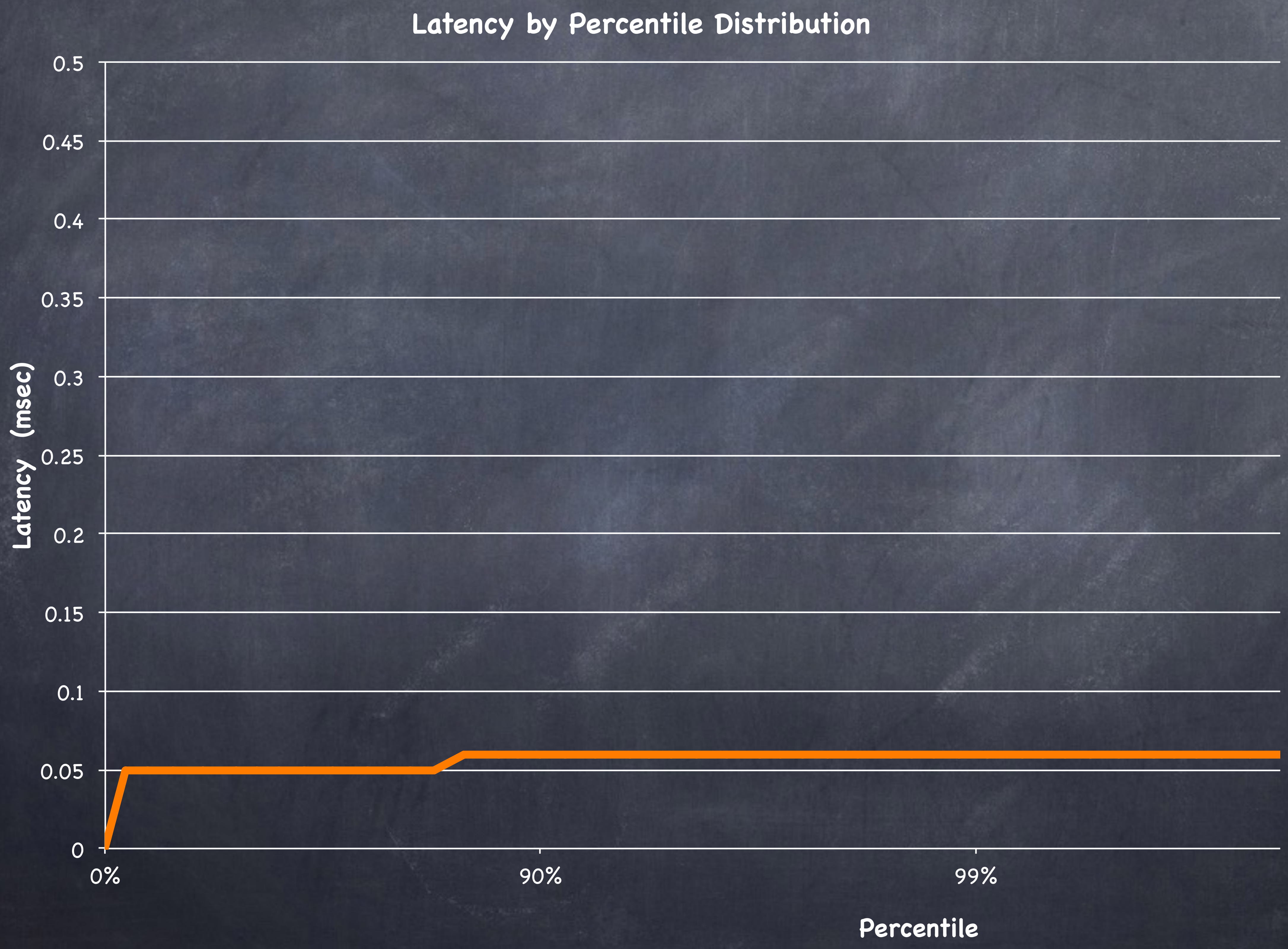
- We know how to compute averages & std. deviation, etc.

- Wouldn't it be nice if latency had a normal distribution?

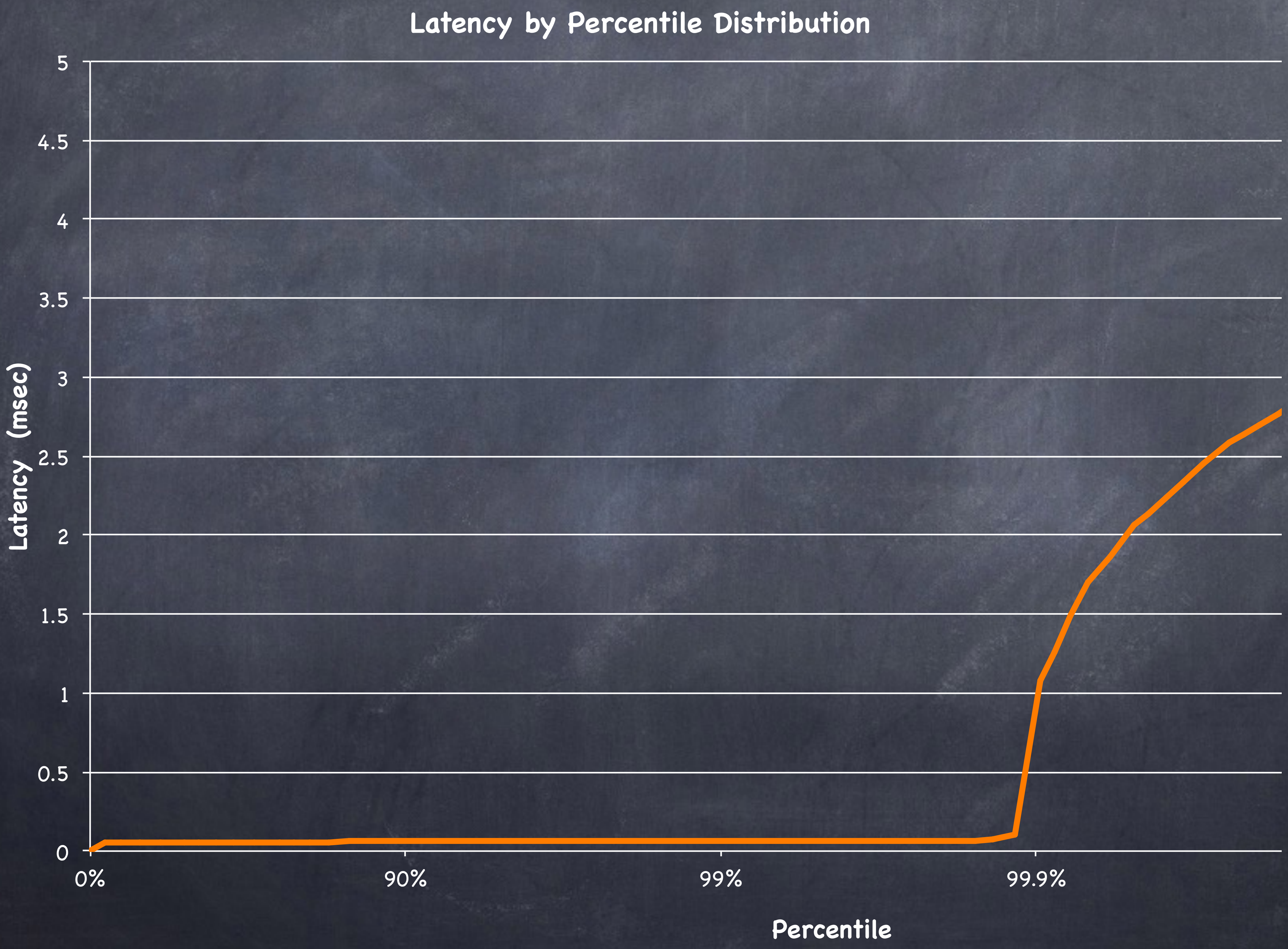
- The average, 90%'lie, 99%'lie, std. deviation, etc. can give us a "feel" for the rest of the distribution, right?

- If 99% of the stuff behaves well, how bad can the rest be, really?

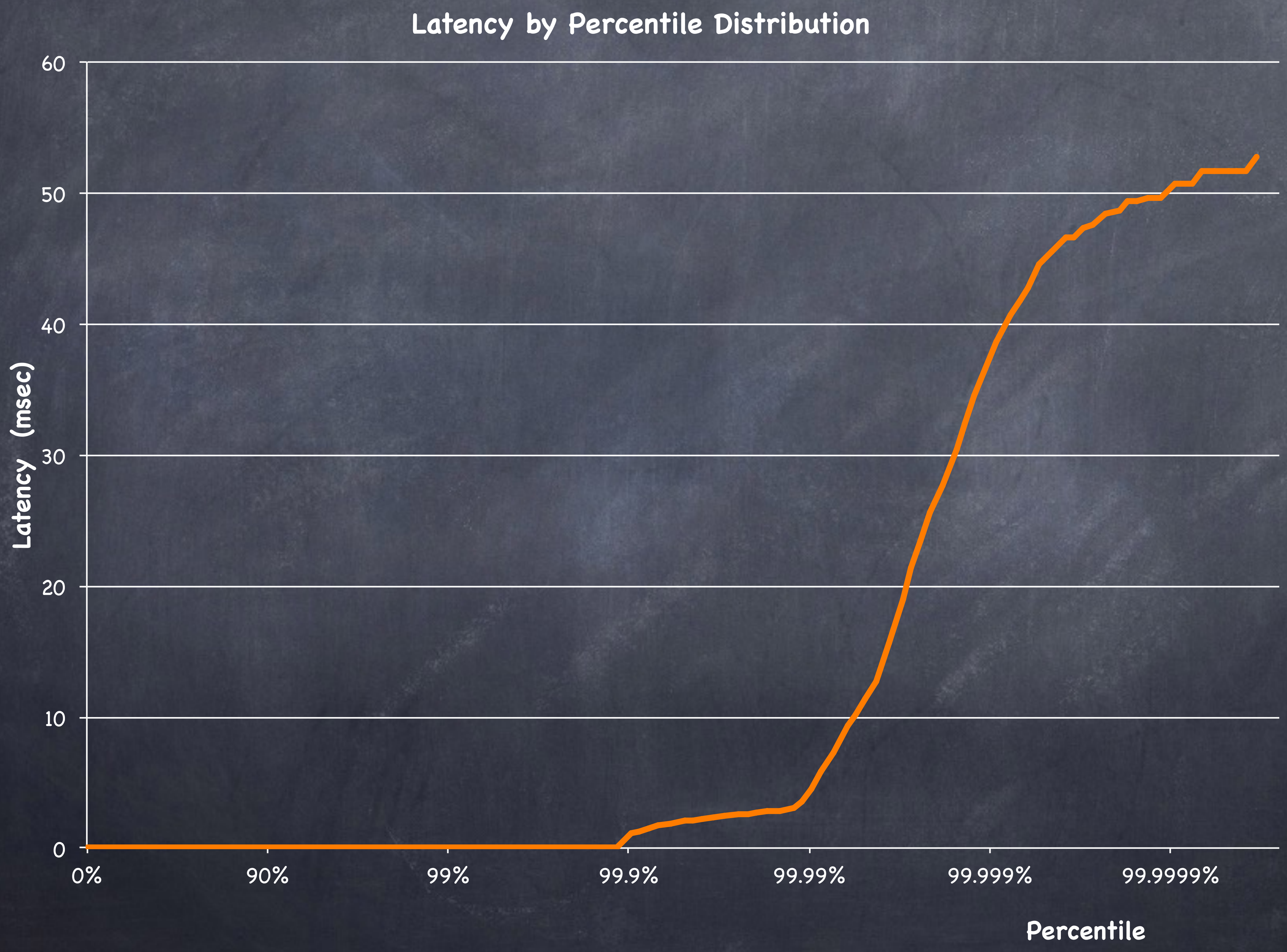
The real world: latency distribution



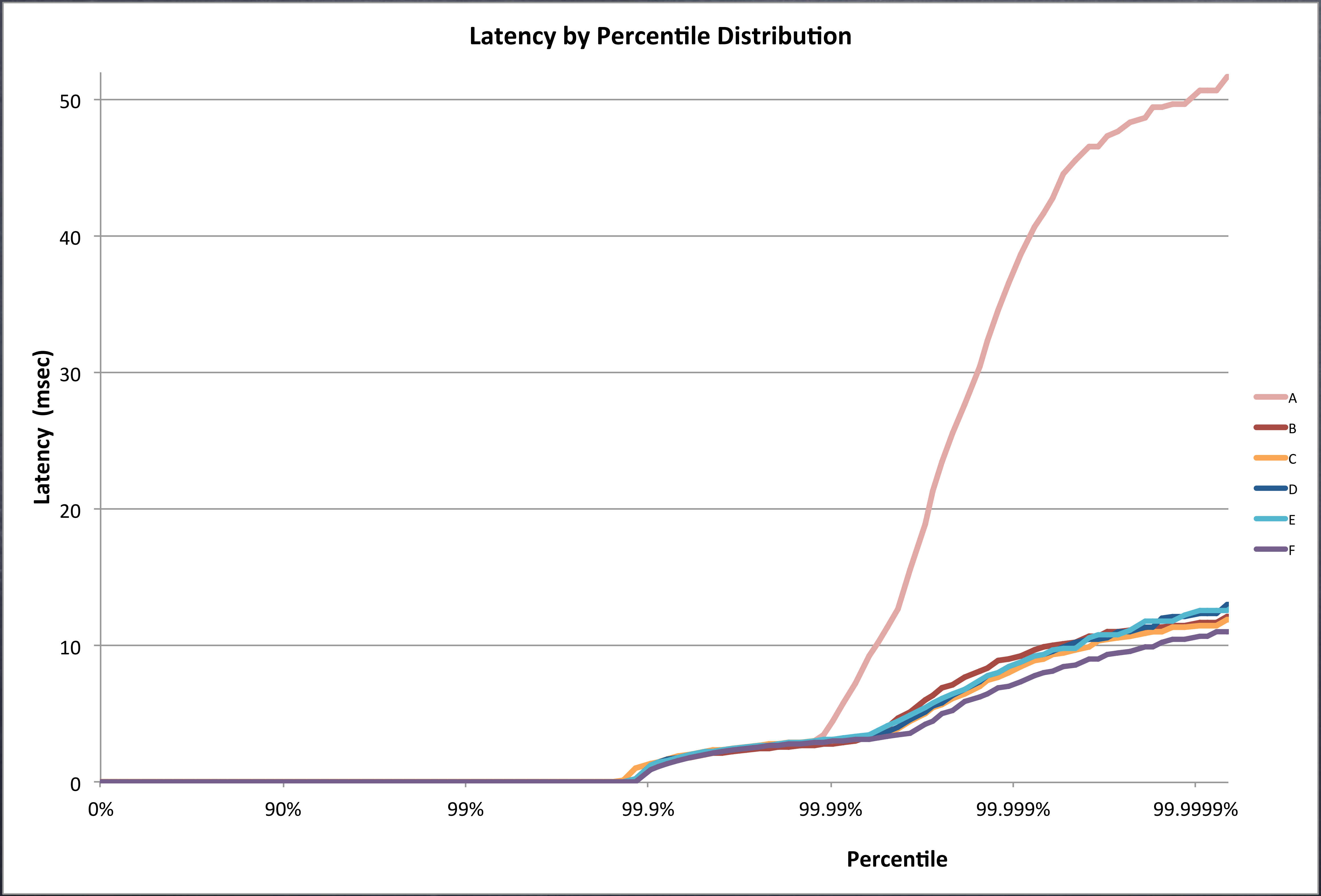
The real world: latency distribution



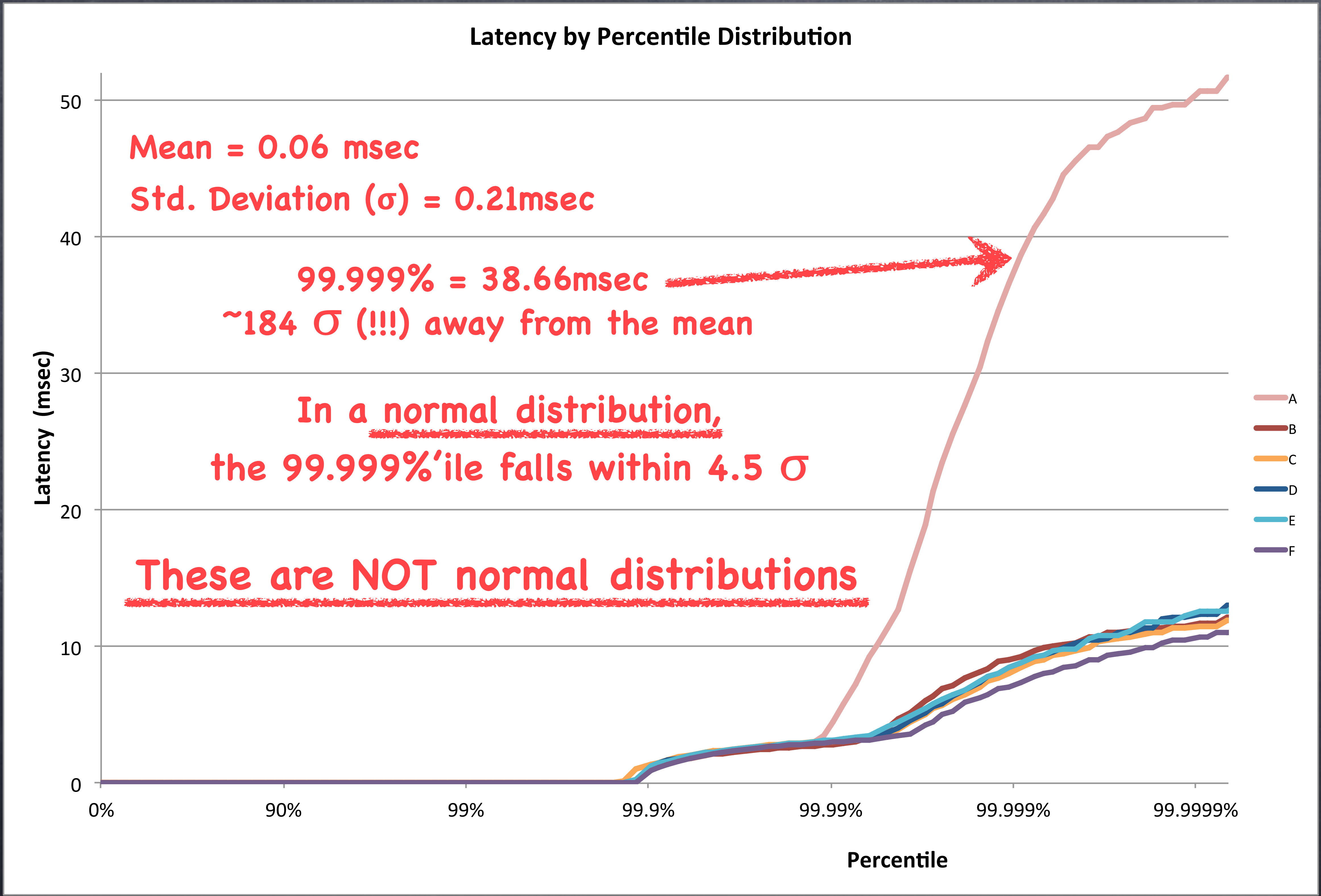
The real world: latency distribution



Dispelling standard deviation



Dispelling standard deviation



Mean = 0.06 msec

Std. Deviation (σ) = 0.21msec

99.999% = 38.66msec

~184 σ (!!!) away from the mean

In a normal distribution,

the 99.999%'ile falls within 4.5 σ

These are NOT normal distributions

The coordinated omission problem

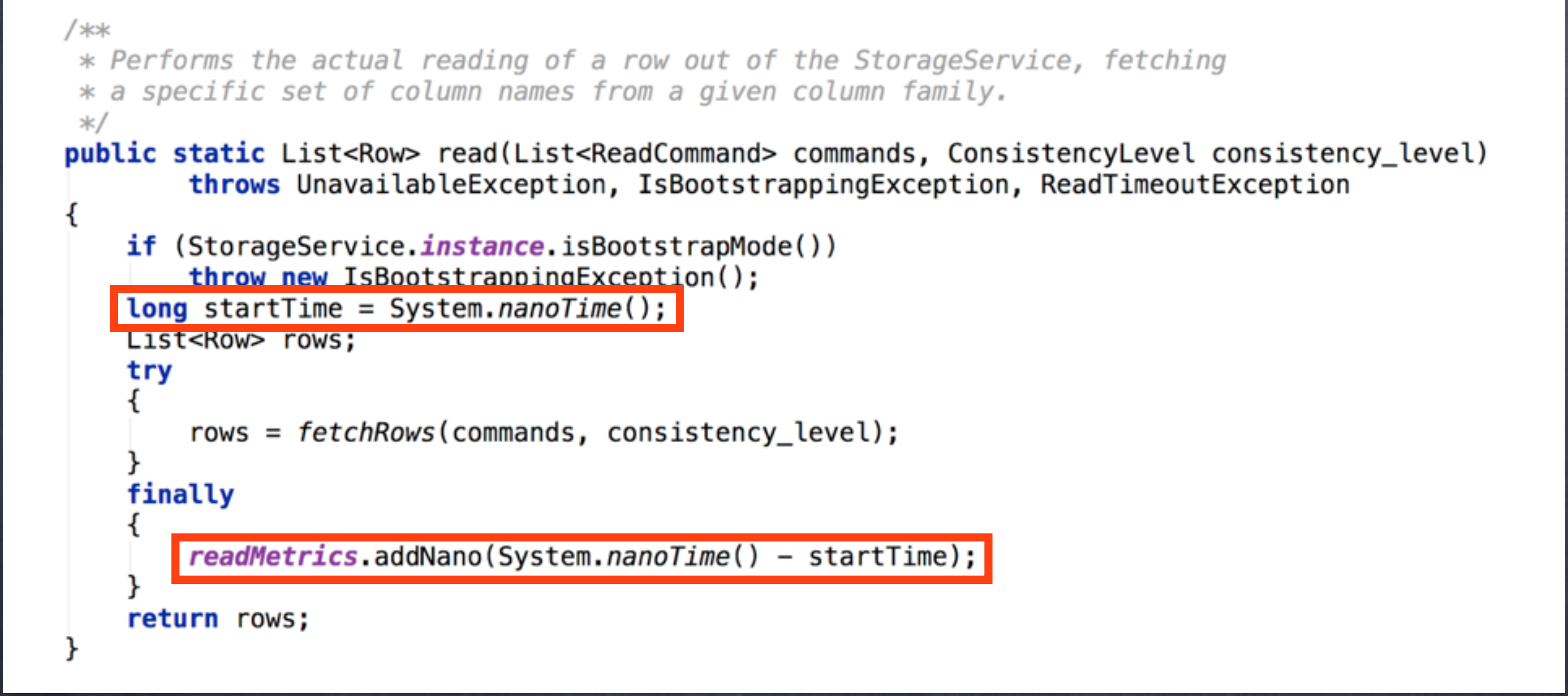
An accidental conspiracy...

The *lie* in the 99%'lies

The coordinated omission problem

- Common Example A (load testing):
  - each "client" issues requests at a certain rate
  - measure/log response time for each request
- So what's wrong with that?
  - works **only** if ALL responses fit within interval
  - implicit "automatic back off" coordination

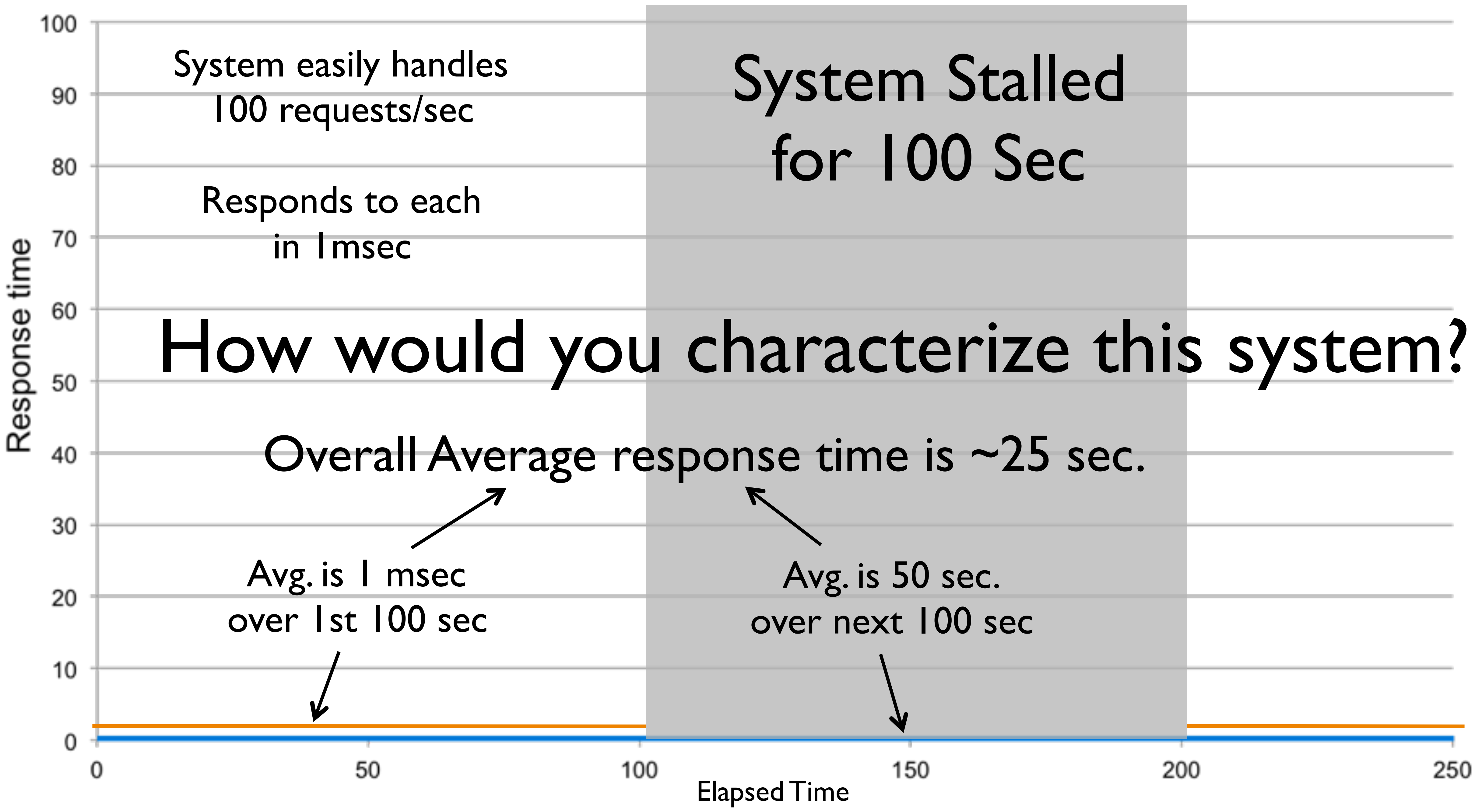
Common Example B:
Coordinated Omission in Monitoring Code



```java
/**
 * Performs the actual reading of a row out of the StorageService, fetching
 * a specific set of column names from a given column family.
 */
public static List<Row> read(List<ReadCommand> commands, ConsistencyLevel consistency_level)
        throws UnavailableException, IsBootstrappingException, ReadTimeoutException
{
    if (StorageService.instance.isBootstrapMode())
        throw new IsBootstrappingException();
    long startTime = System.nanoTime();
    List<Row> rows;
    try
    {
        rows = fetchRows(commands, consistency_level);
    }
    finally
    {
        readMetrics.addNano(System.nanoTime() - startTime);
    }
    return rows;
}
```

- Long operations only get measured once
- delays outside of timing window do not get measured at all

How bad can this get?



System easily handles
100 requests/sec

Responds to each
in 1msec

System Stalled
for 100 Sec

How would you characterize this system?

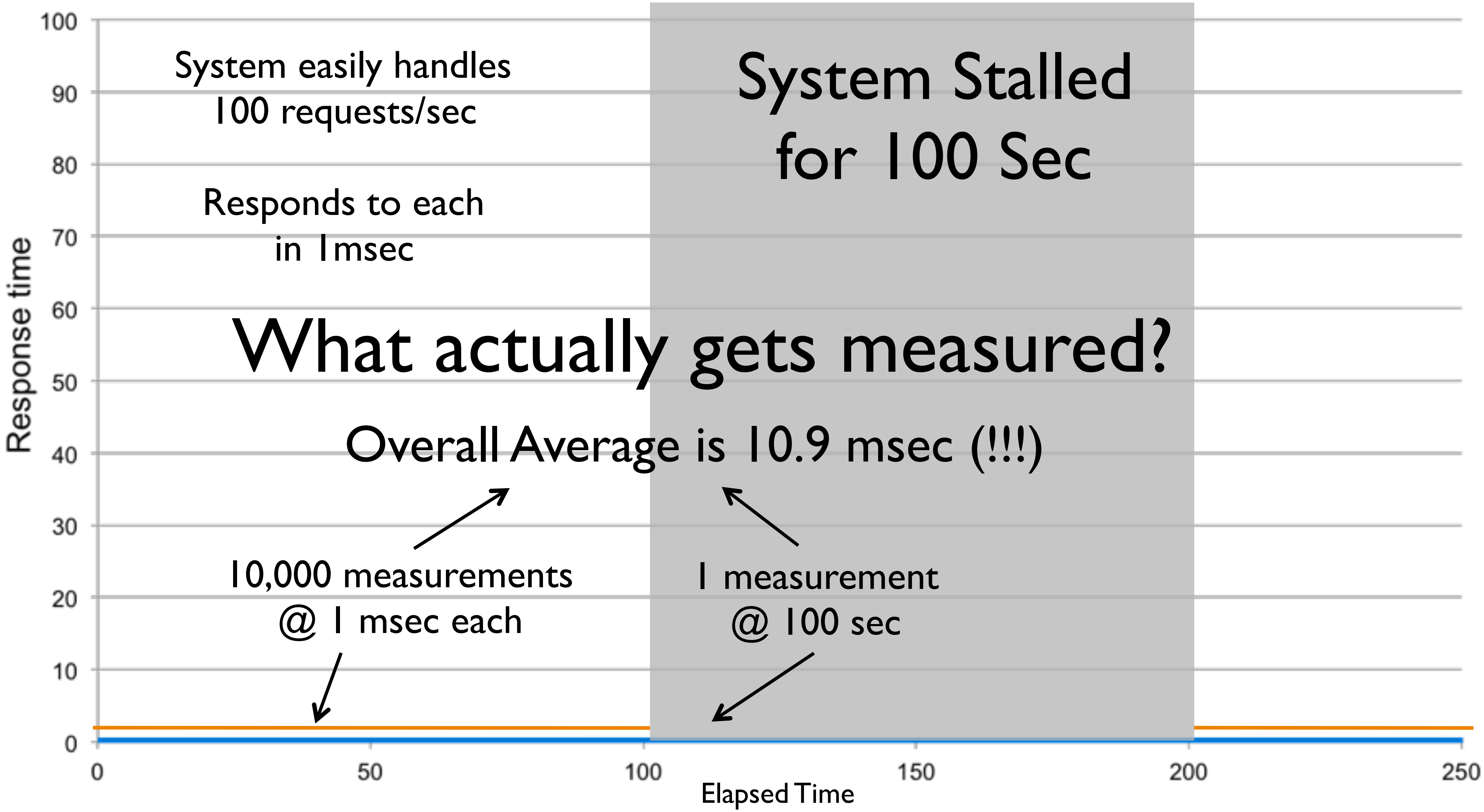
Overall Average response time is ~25 sec.

Avg. is 1 msec
over 1st 100 sec

Avg. is 50 sec.
over next 100 sec

~50%'ile is 1 msec ~75%'ile is 50 sec 99.99%'ile is ~100sec

Measurement in practice



System easily handles
100 requests/sec

Responds to each
in 1msec

System Stalled
for 100 Sec

What actually gets measured?

Overall Average is 10.9 msec (!!!)

10,000 measurements
@ 1 msec each

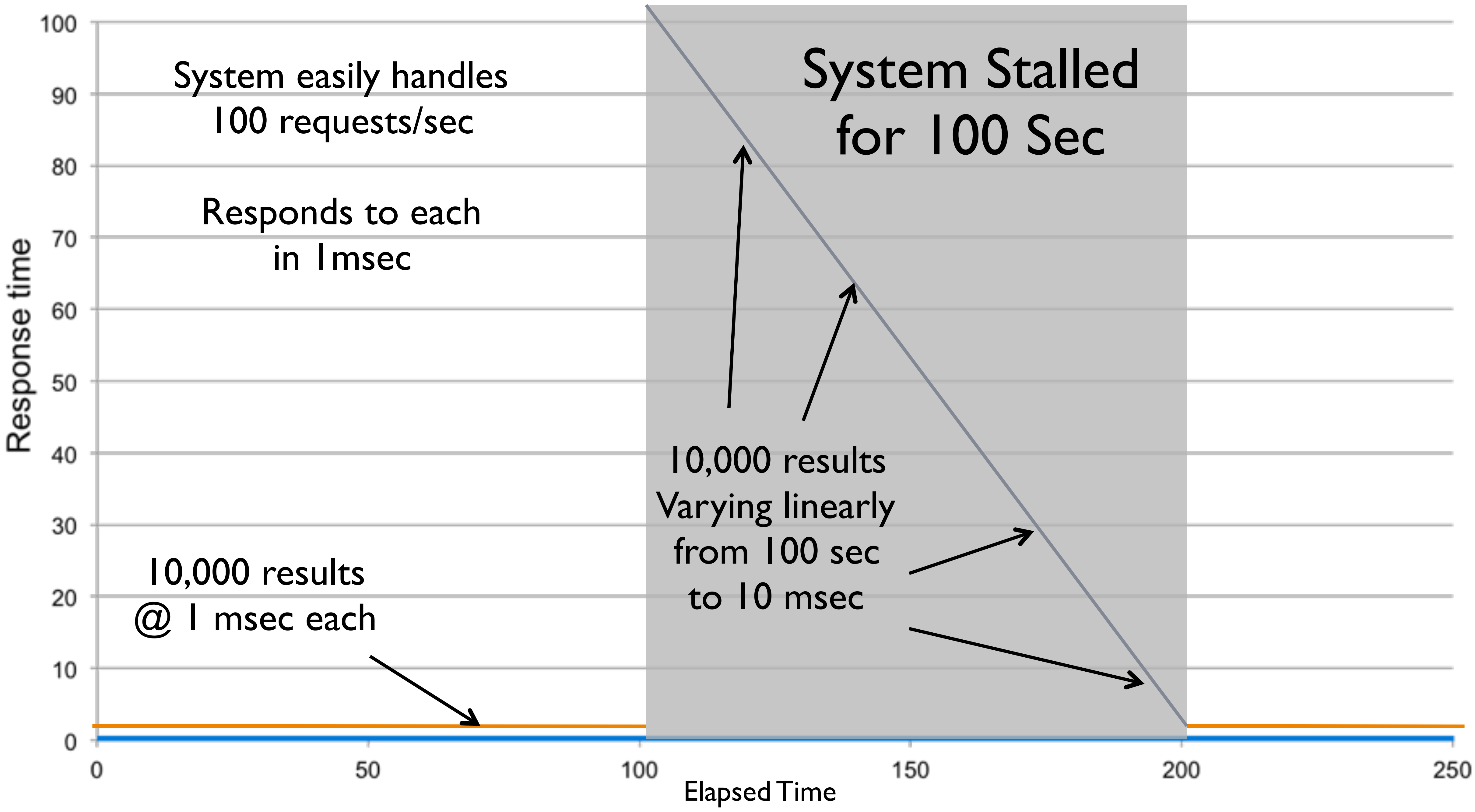
1 measurement
@ 100 sec

50%'ile is 1 msec

75%'lie is 1 msec
(should be ~50sec)

99.99%'lie is 1 msec
(should be ~100 sec)

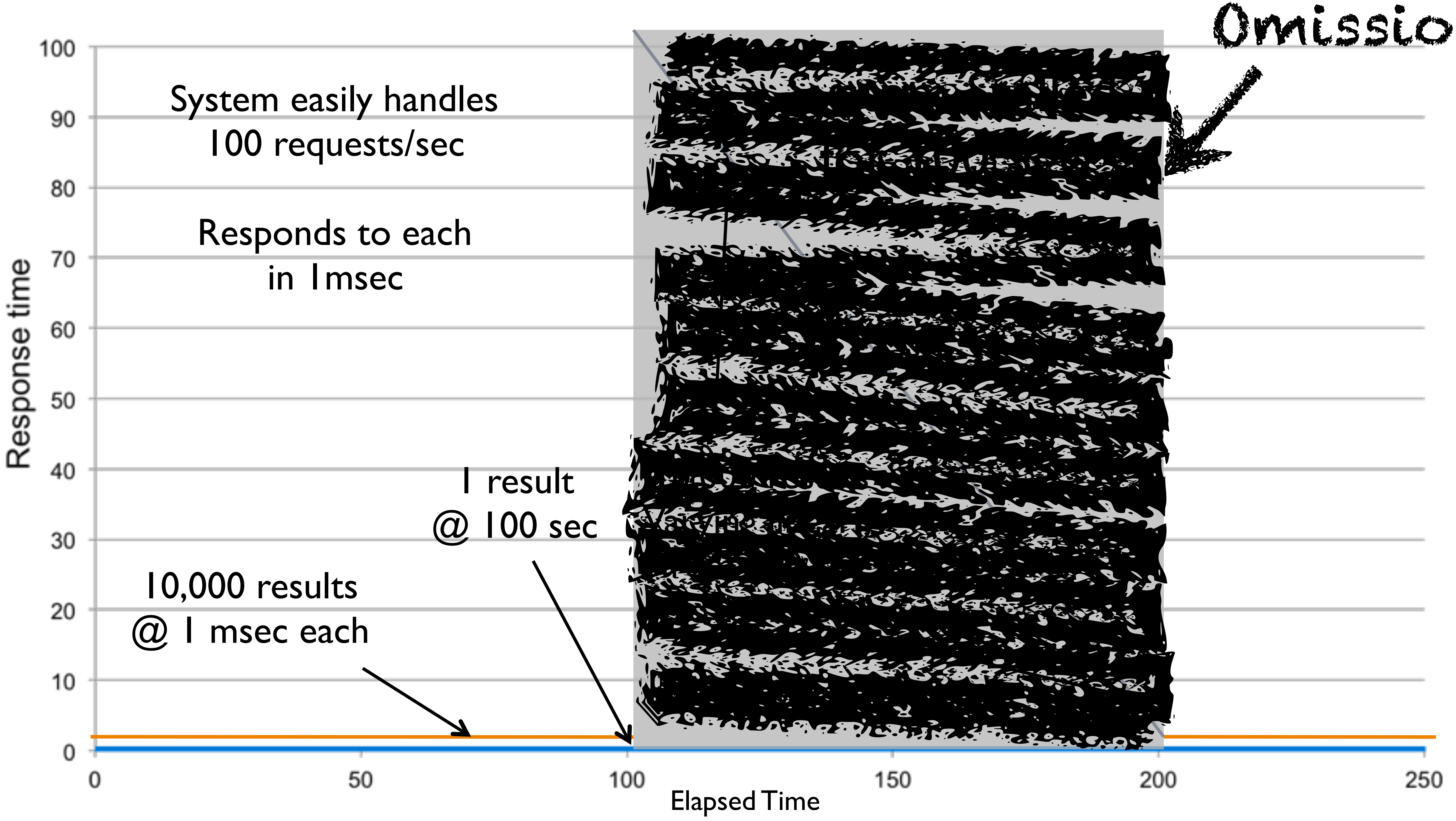
Proper measurement



~50%'ile is 1 msec ~75%'ile is 50 sec 99.99%'ile is ~100sec

Proper measurement

Coordinated Omission



[redacted]

System easily handles 100 requests/sec

Responds to each in 1msec

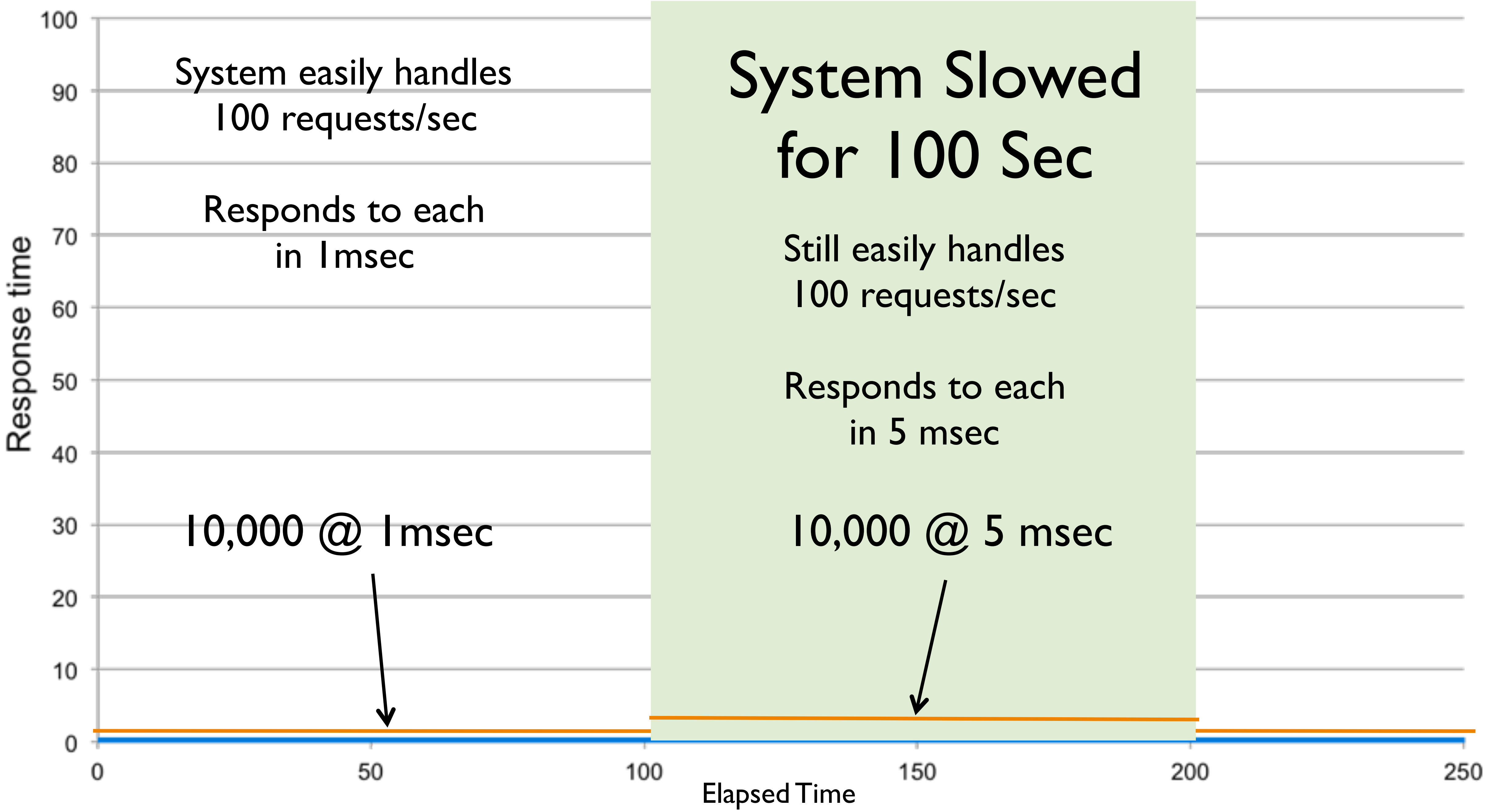
1 result @ 100 sec

10,000 results @ 1 msec each

Elapsed Time

~50%'ile is 1 msec ~75%'ile is ~~50 sec~~ 1 msec 99.99%'ile is ~~100sec~~ 1 msec

“Better” can look “Worse”



System easily handles 100 requests/sec

Responds to each in 1msec

10,000 @ 1msec

System Slowed for 100 Sec

Still easily handles 100 requests/sec

Responds to each in 5 msec

10,000 @ 5 msec

50%‘ile is 1 msec

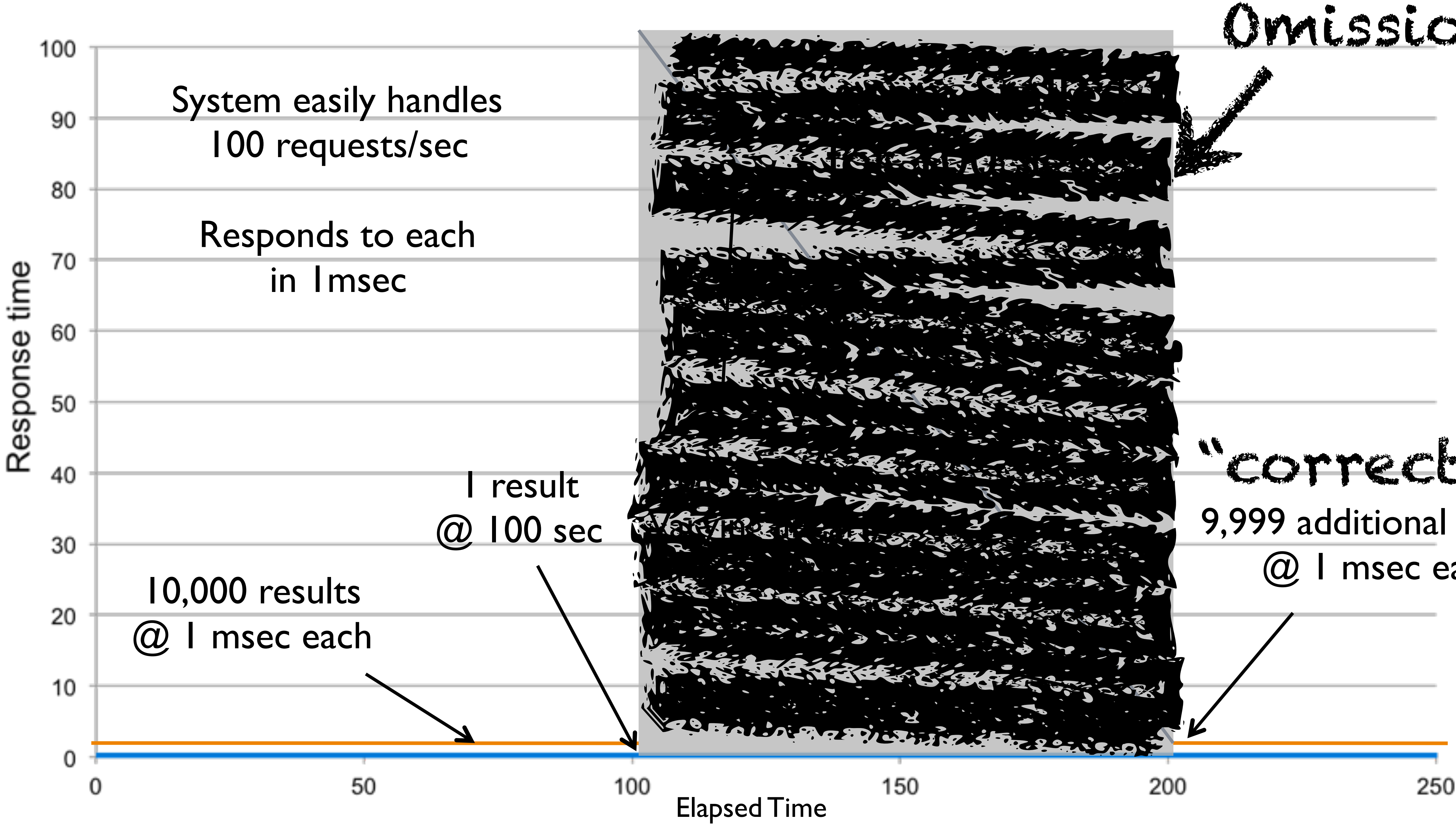
75%‘lie is 2.5msec
(stalled shows 1 msec)

99.99%‘lie is ~5msec
(stalled shows 1 msec)

“Correction”: “Cheating Twice”

Coordinated

Omission



System easily handles 100 requests/sec

Responds to each in 1msec

1 result @ 100 sec

10,000 results @ 1 msec each

“correction”

9,999 additional results @ 1 msec each

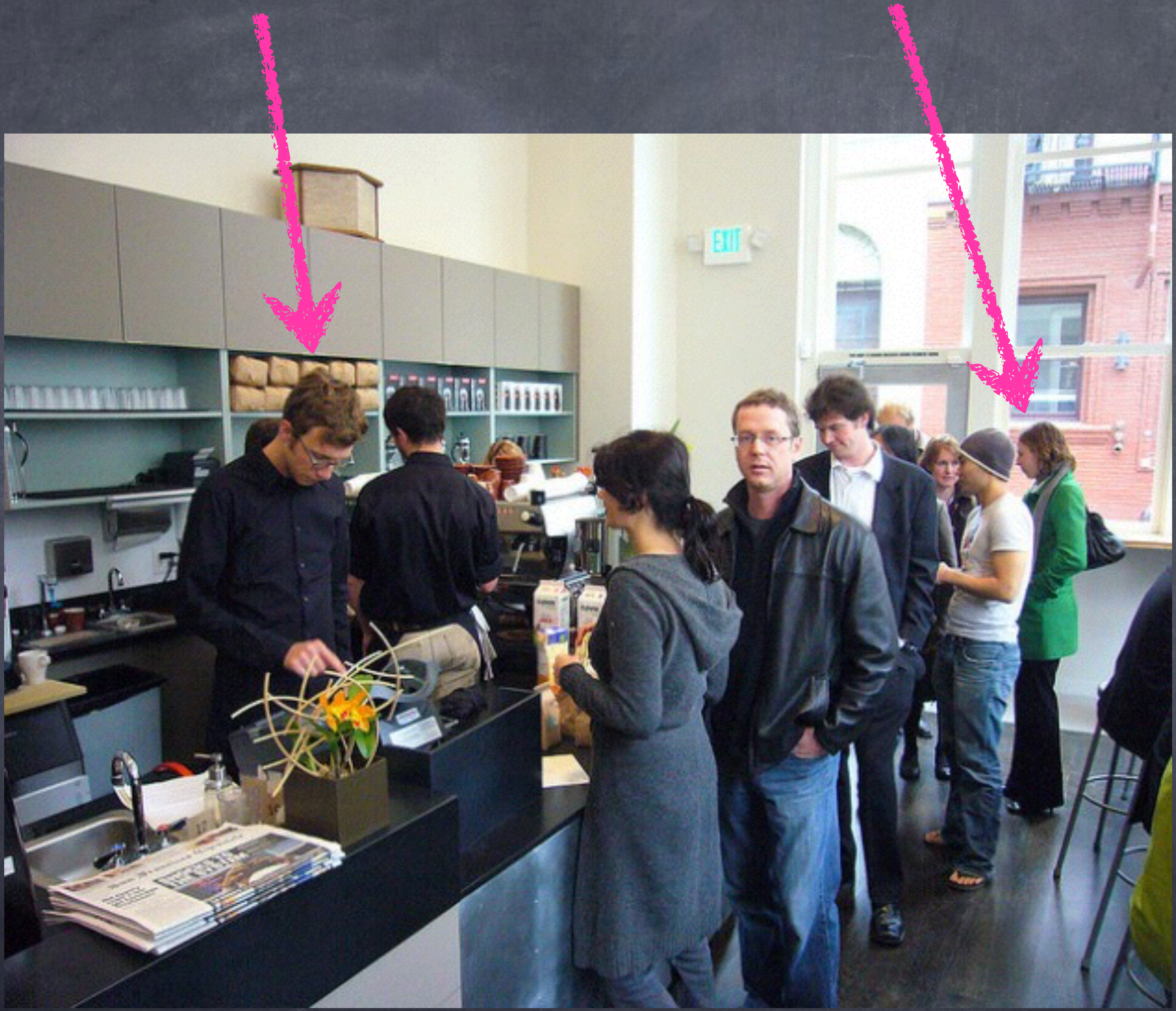
~50%‘ile is 1 msec

~75%‘ile is ~~50 sec~~ 1 msec

99.994%‘ile is ~~~100sec~~ 1 msec

Response Time vs. Service Time

Service Time vs. Response Time



Coordinated Omission

Usually

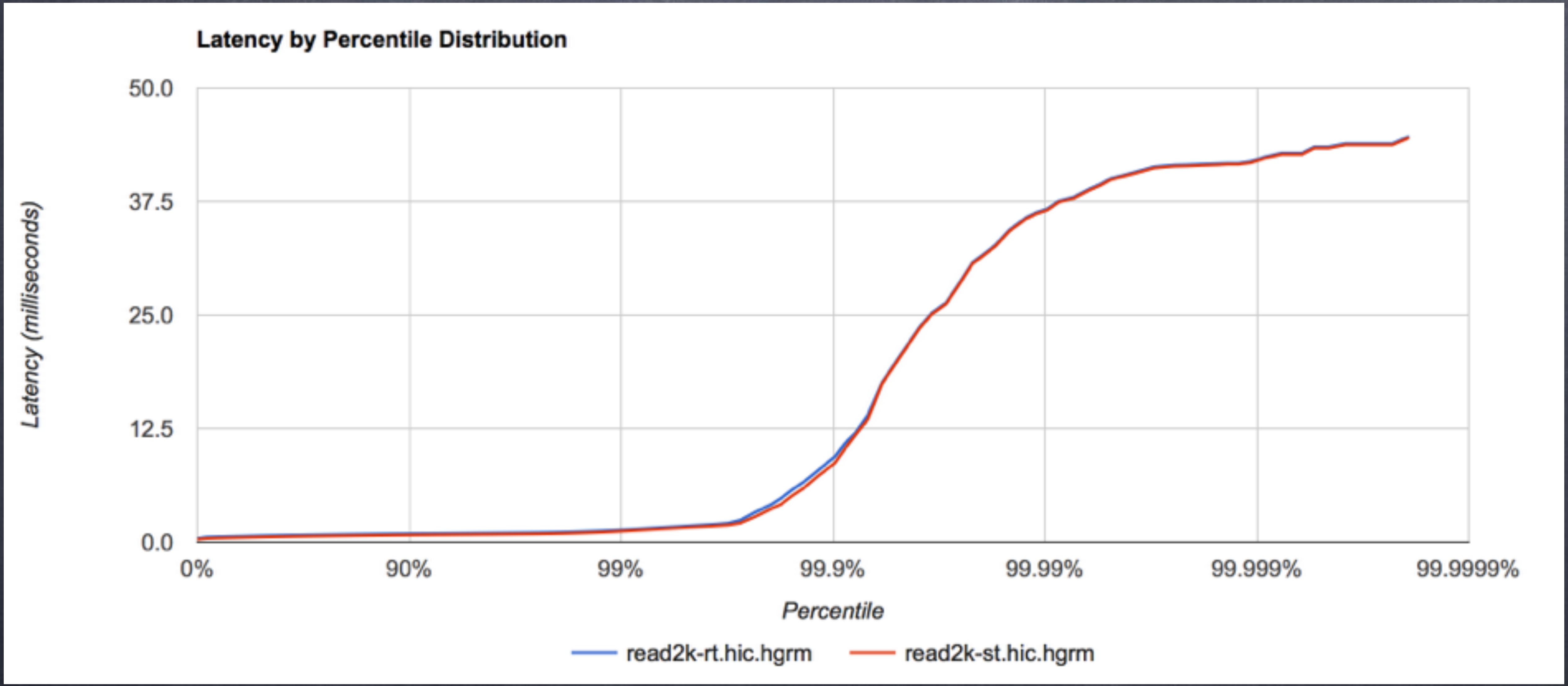
makes something that you ***think*** is a

Response Time metric

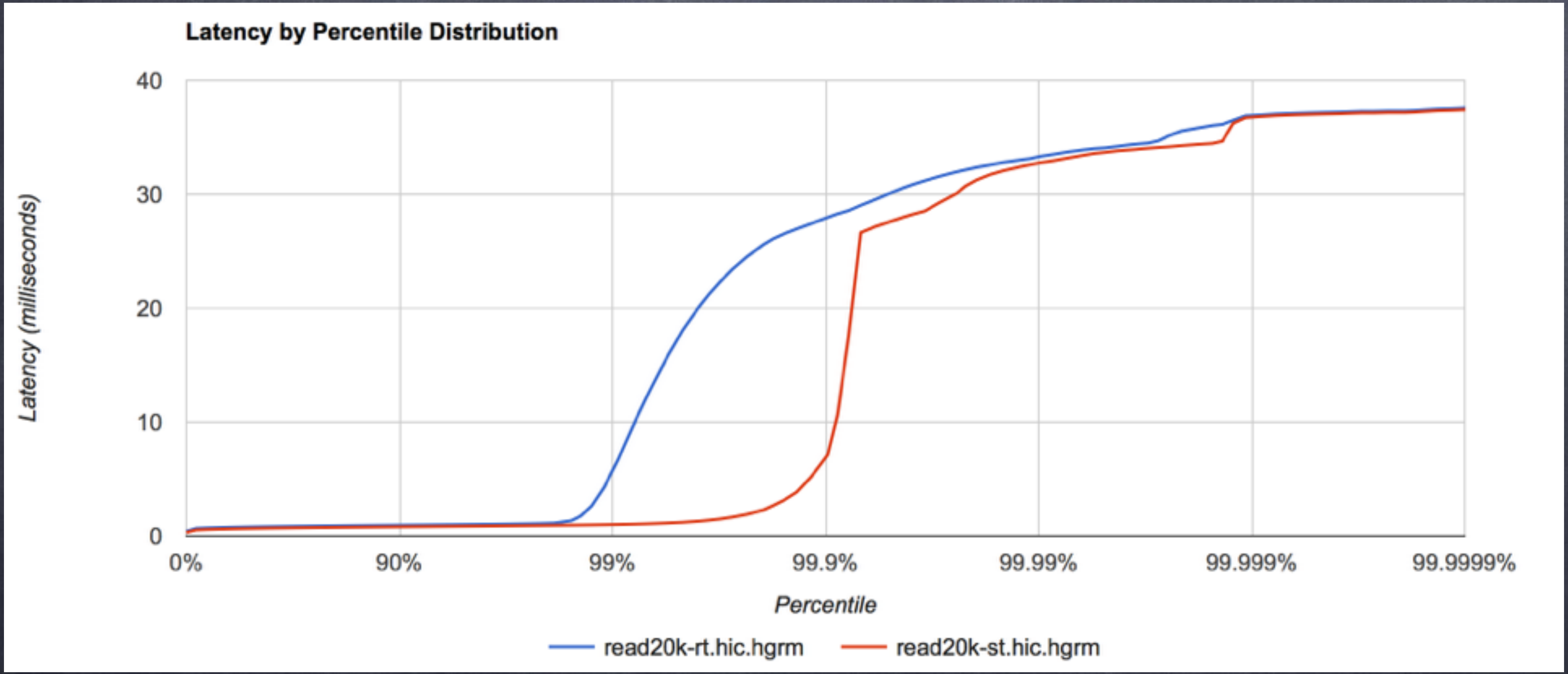
only represent

the ***Service Time*** component

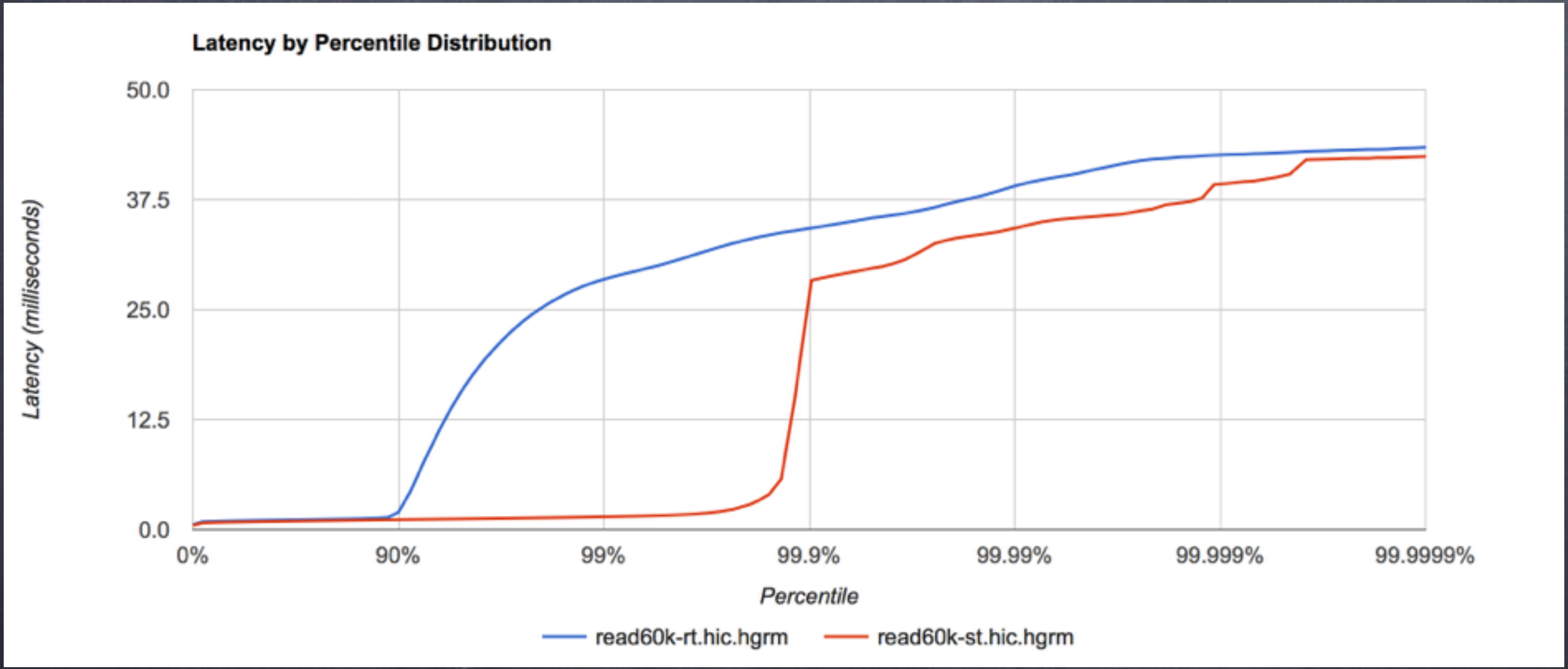
Response Time vs. Service Time @2K/sec



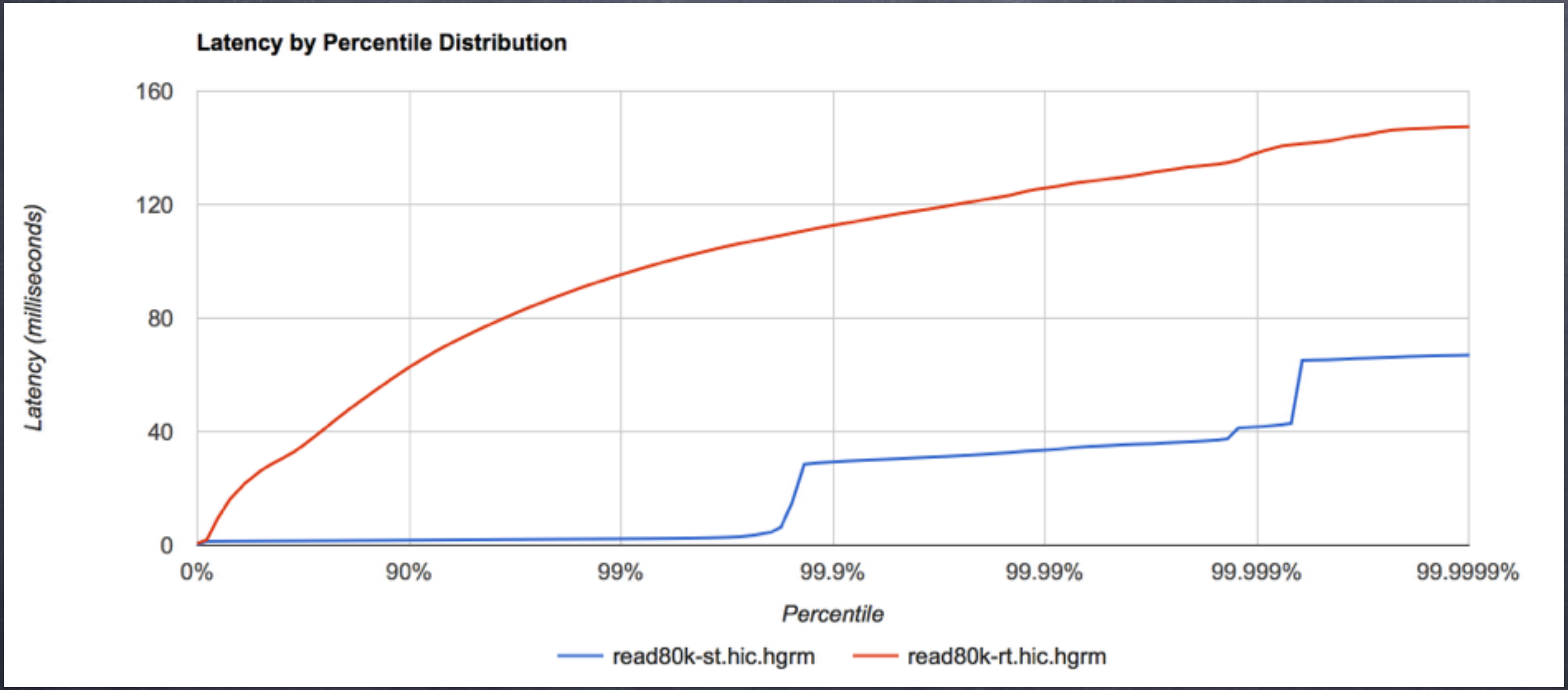
Response Time vs. Service Time @20K/sec



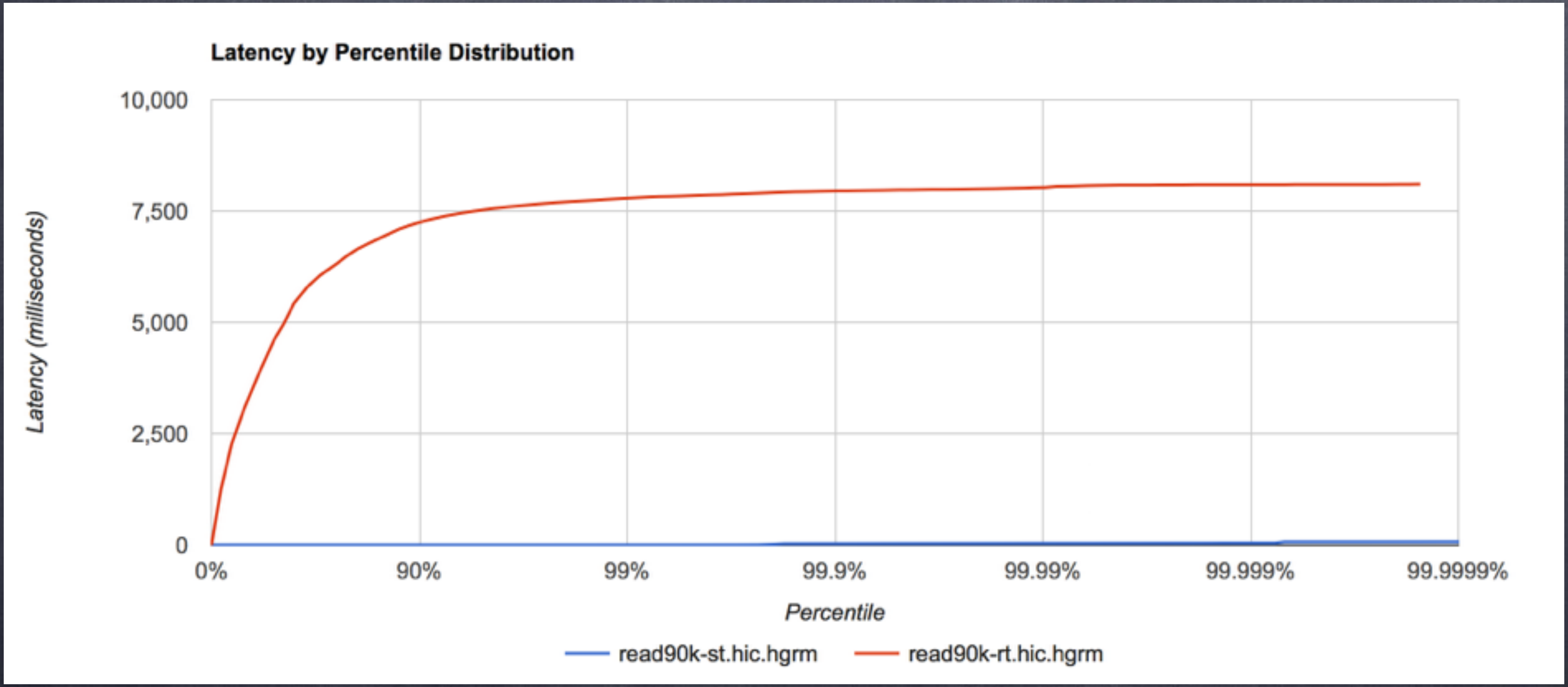
Response Time vs. Service Time @60K/sec



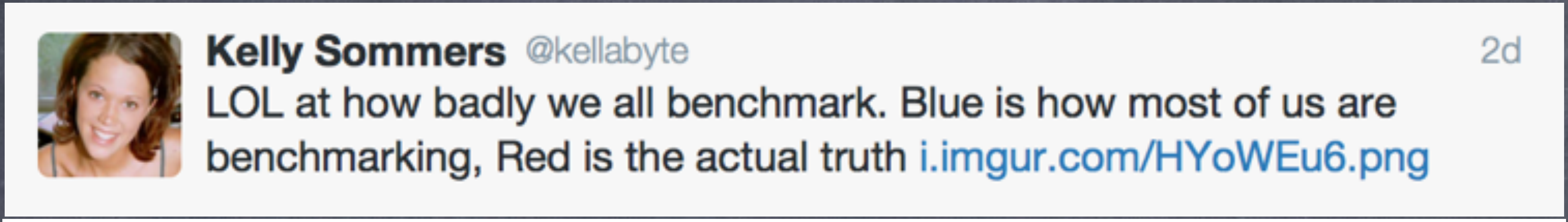
Response Time vs. Service Time @80K/sec



Response Time vs. Service Time @90K/sec

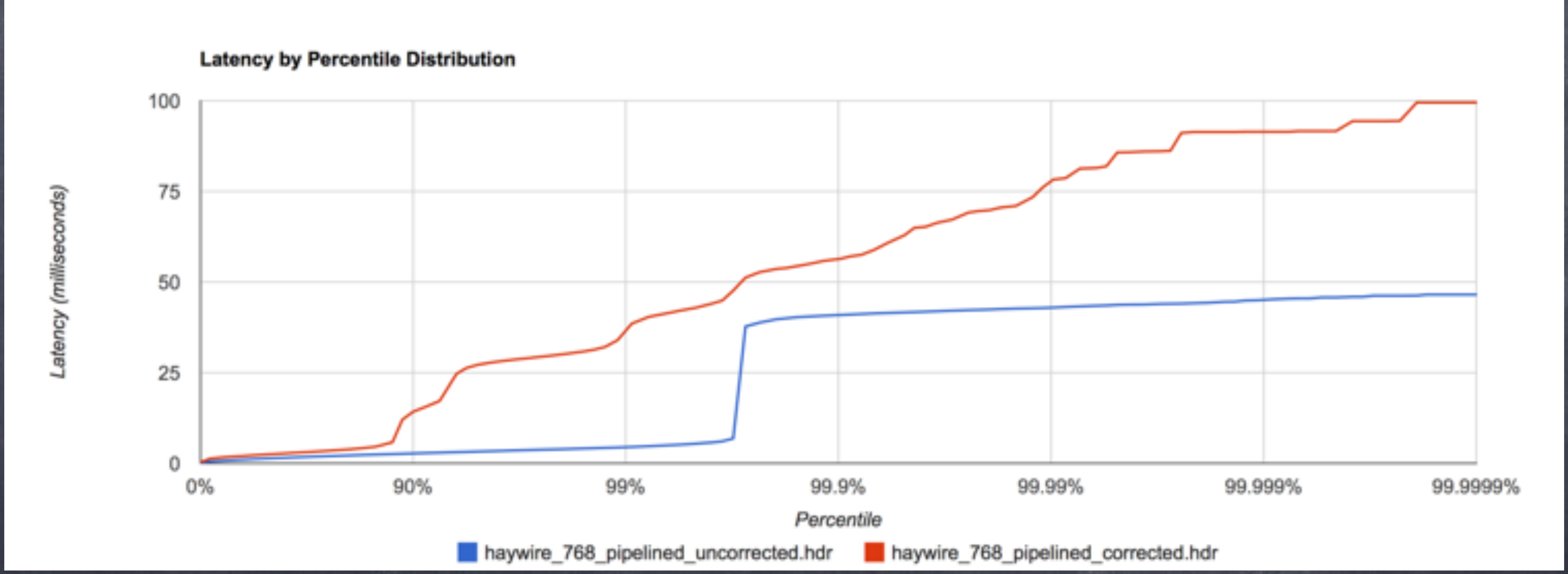


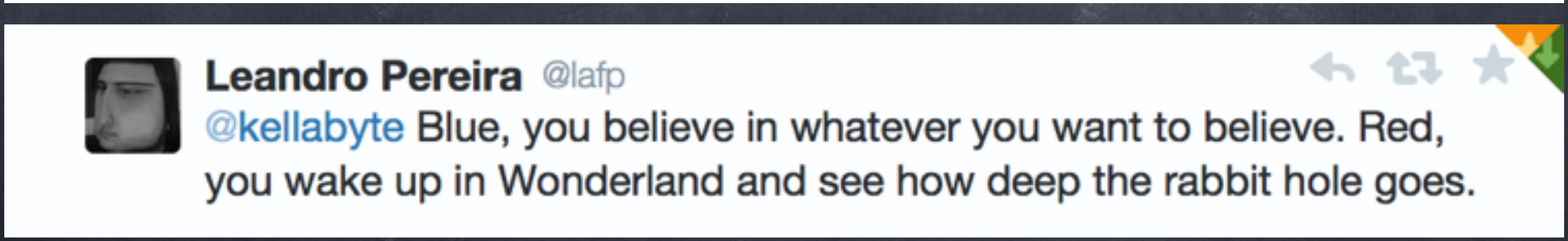
How “real” people react



Kelly Sommers @kellabyte 2d

LOL at how badly we all benchmark. Blue is how most of us are benchmarking, Red is the actual truth i.imgur.com/HYoWEu6.png

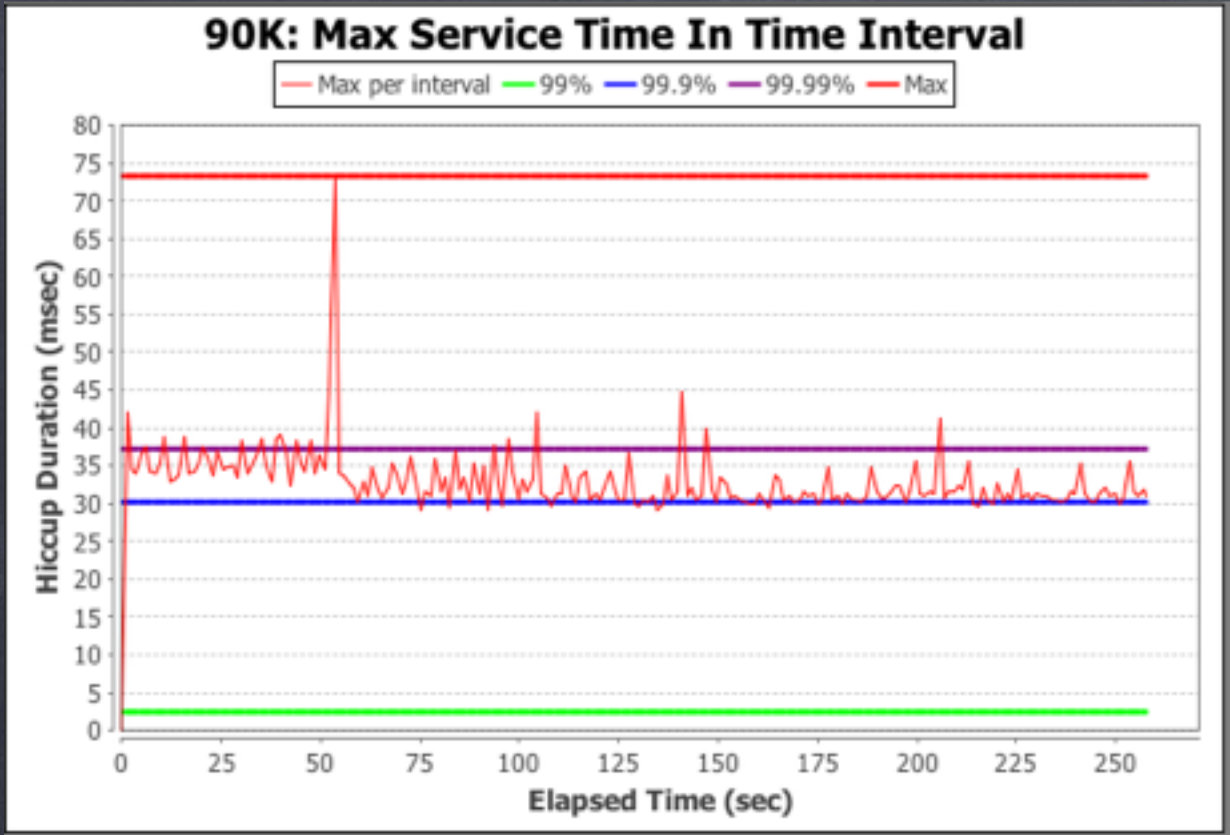


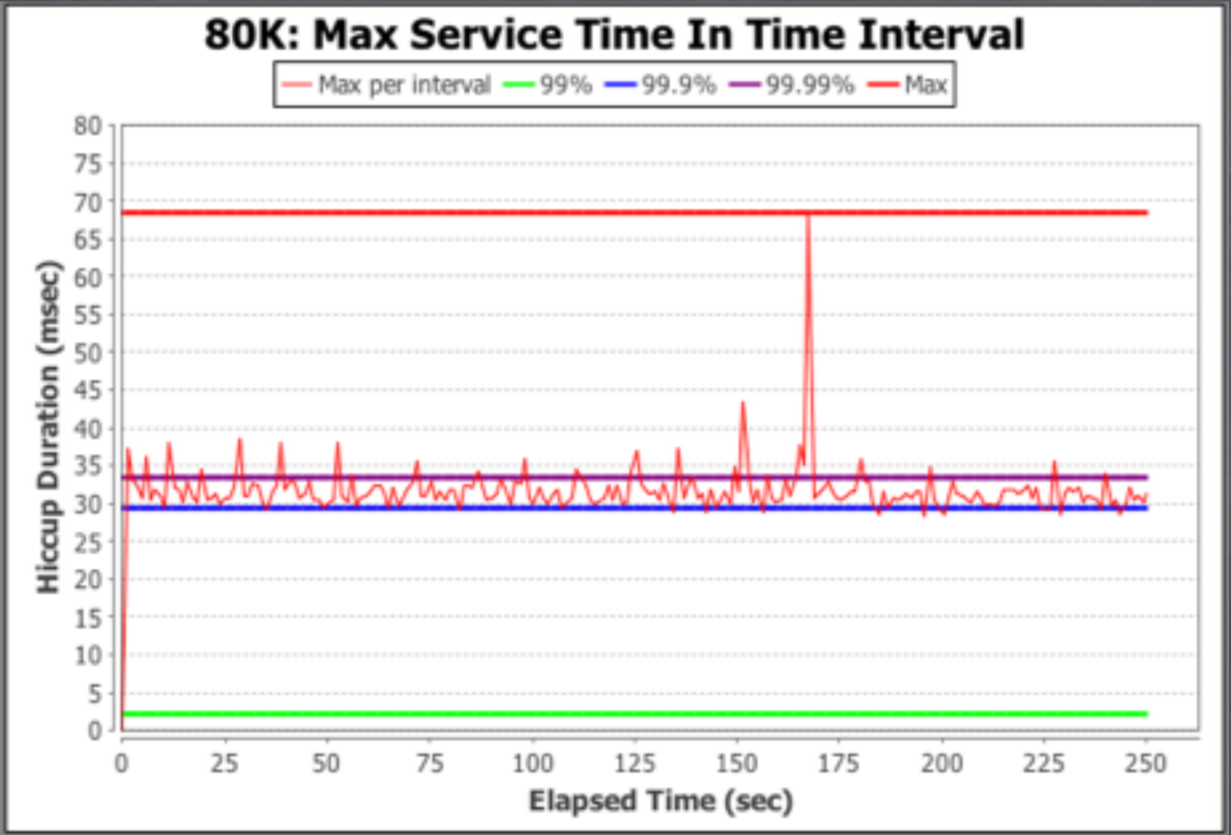


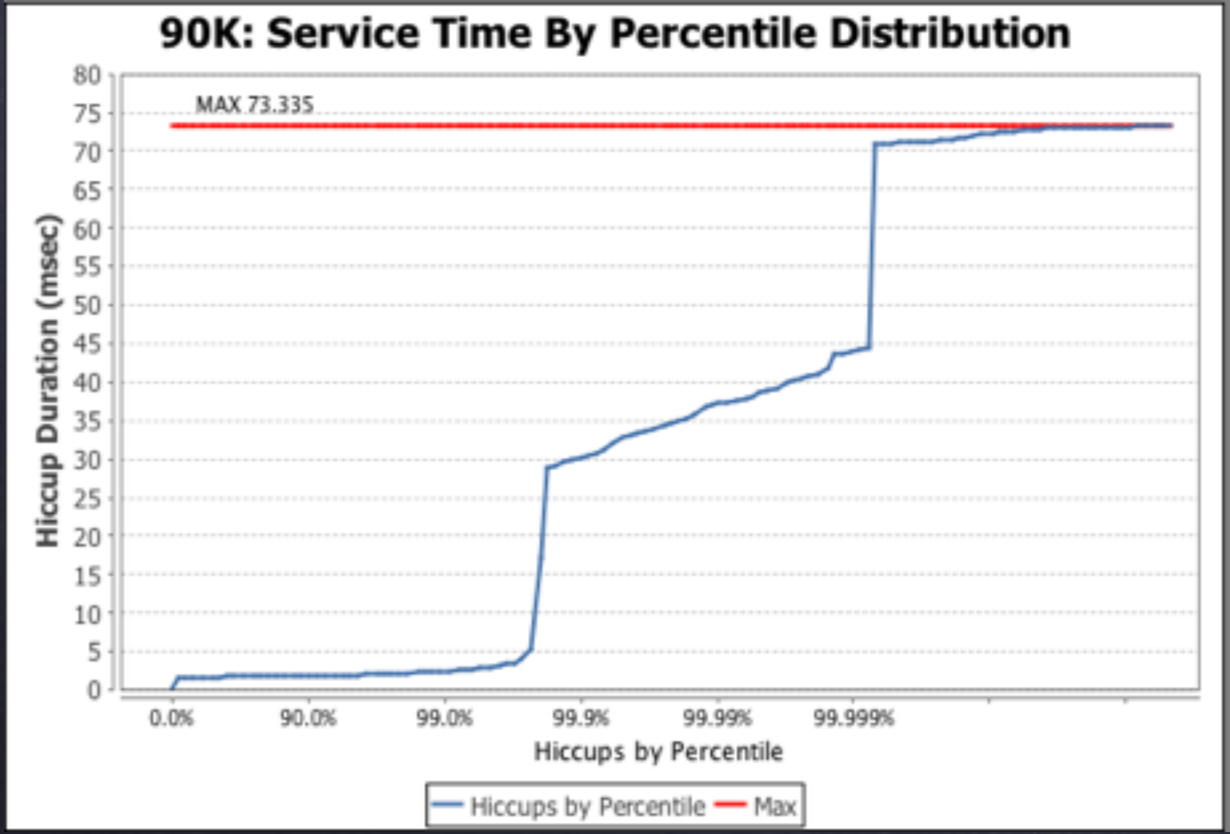
Leandro Pereira @lafp

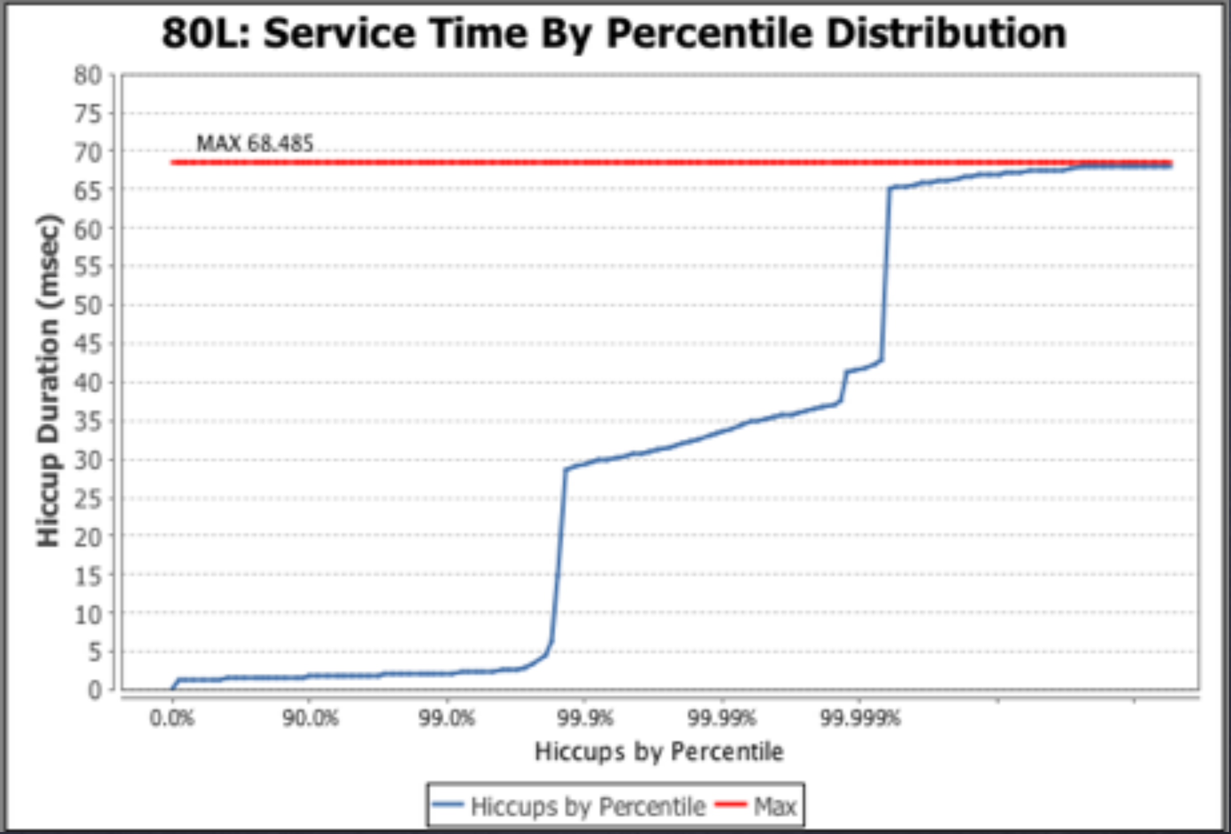
@kellabyte Blue, you believe in whatever you want to believe. Red, you wake up in Wonderland and see how deep the rabbit hole goes.

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

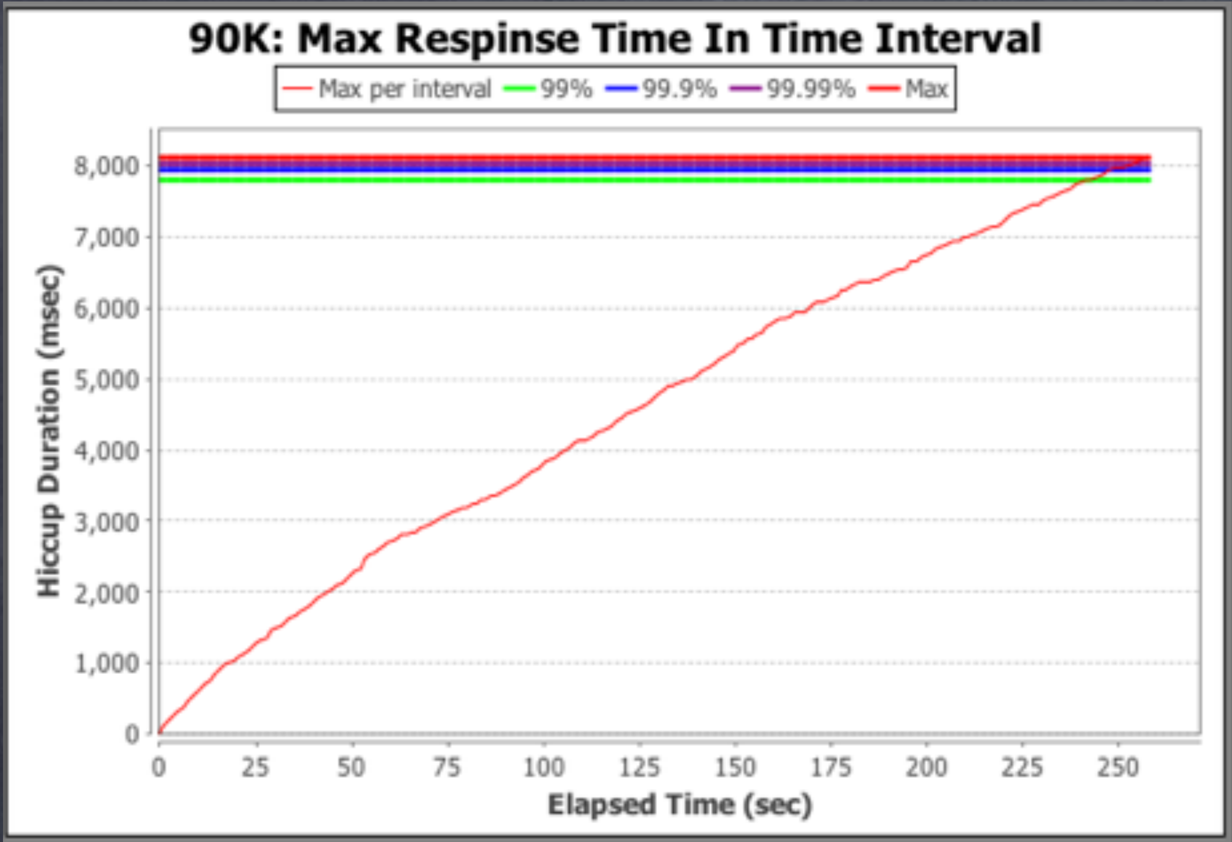


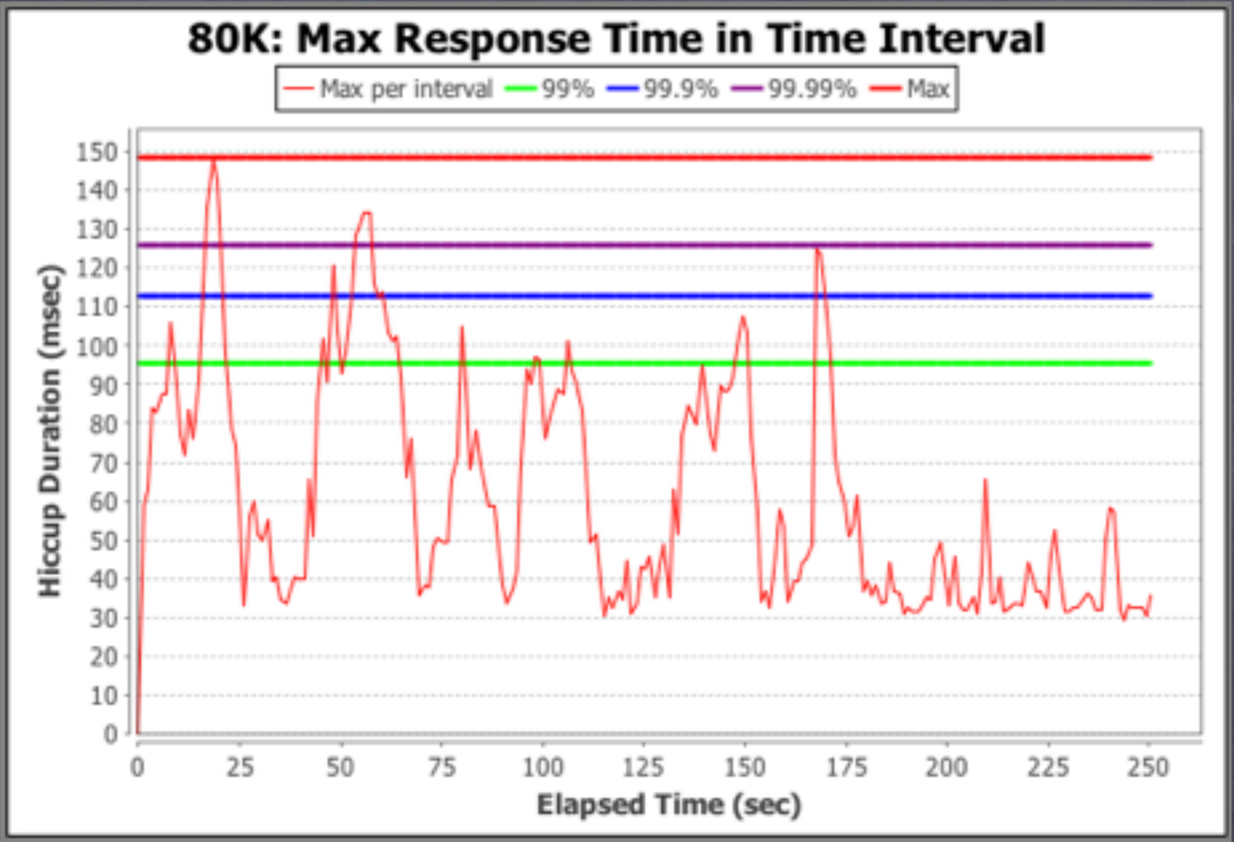


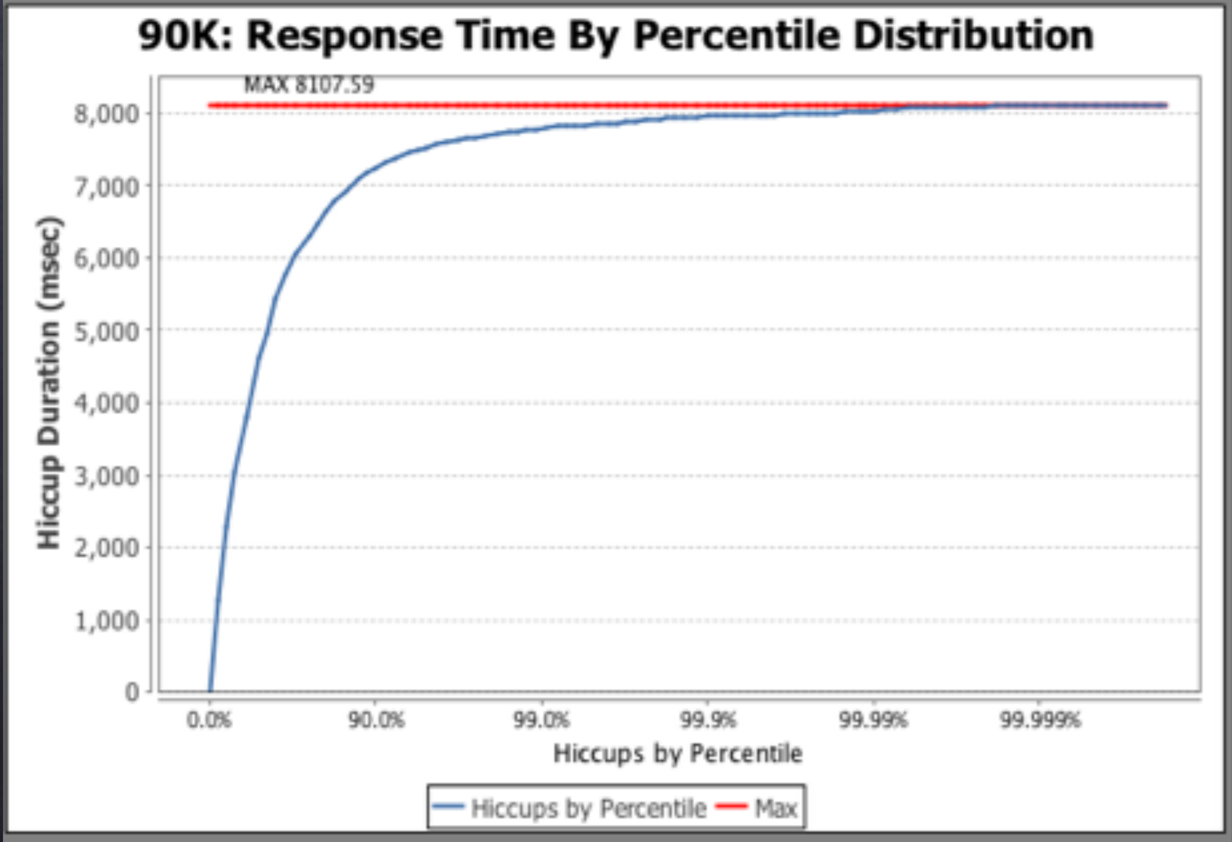


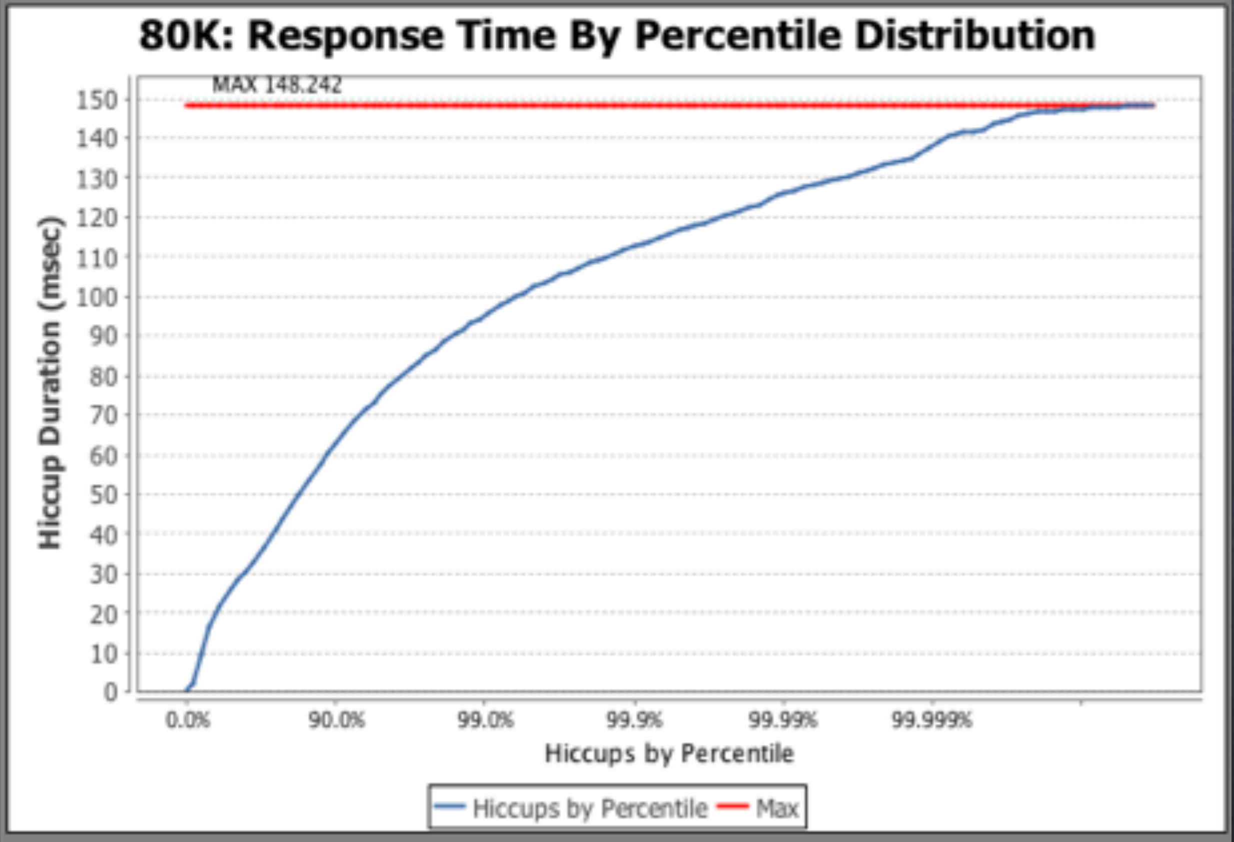


Response Time, 90K/s vs 80K/s

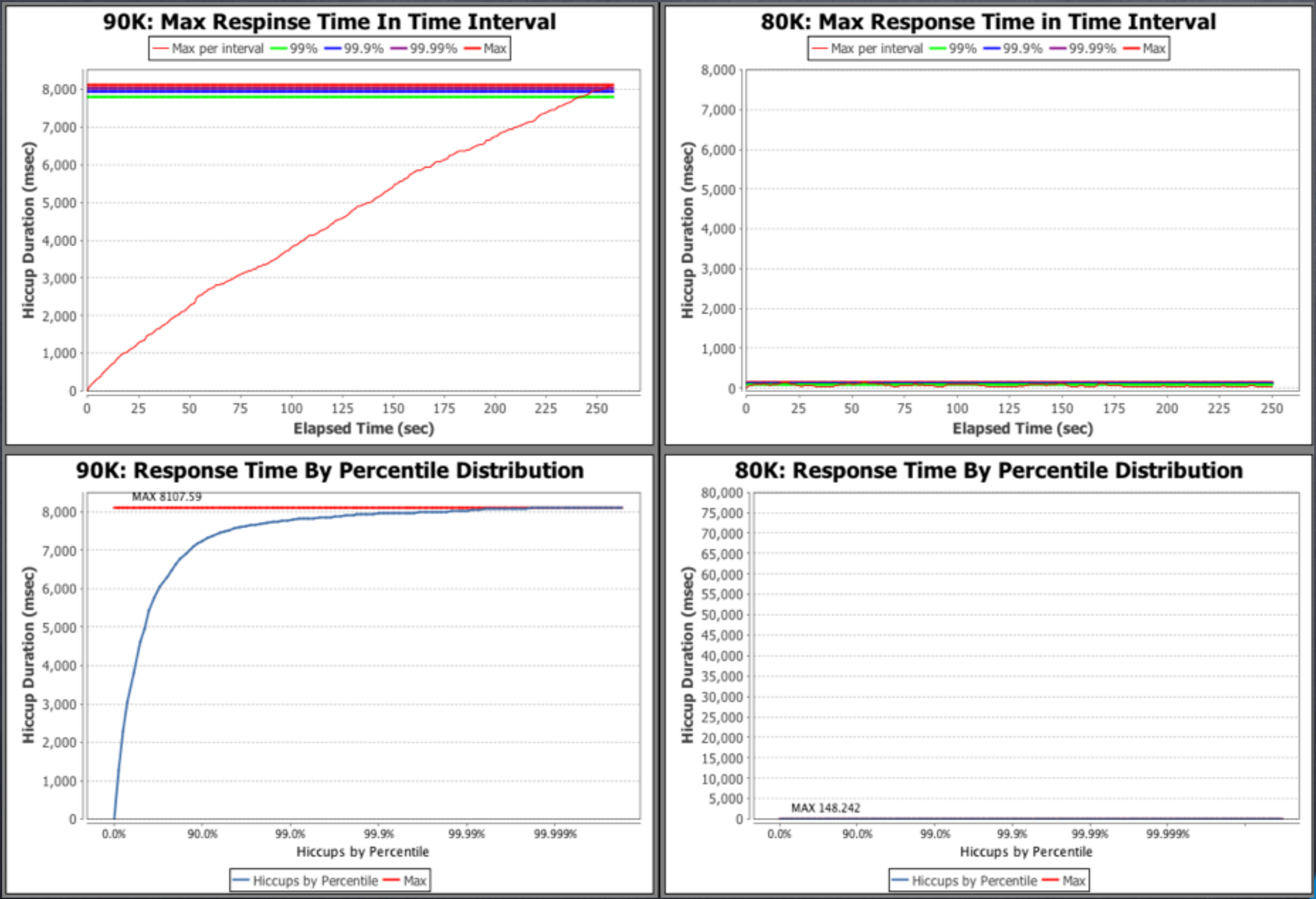

90K: Max Respinse Time In Time Interval


80K: Max Response Time in Time Interval


90K: Response Time By Percentile Distribution


80K: Response Time By Percentile Distribution

Response Time, 90K/s vs 80K/s (to scale)



90K: Max Respinse Time In Time Interval

80K: Max Response Time in Time Interval

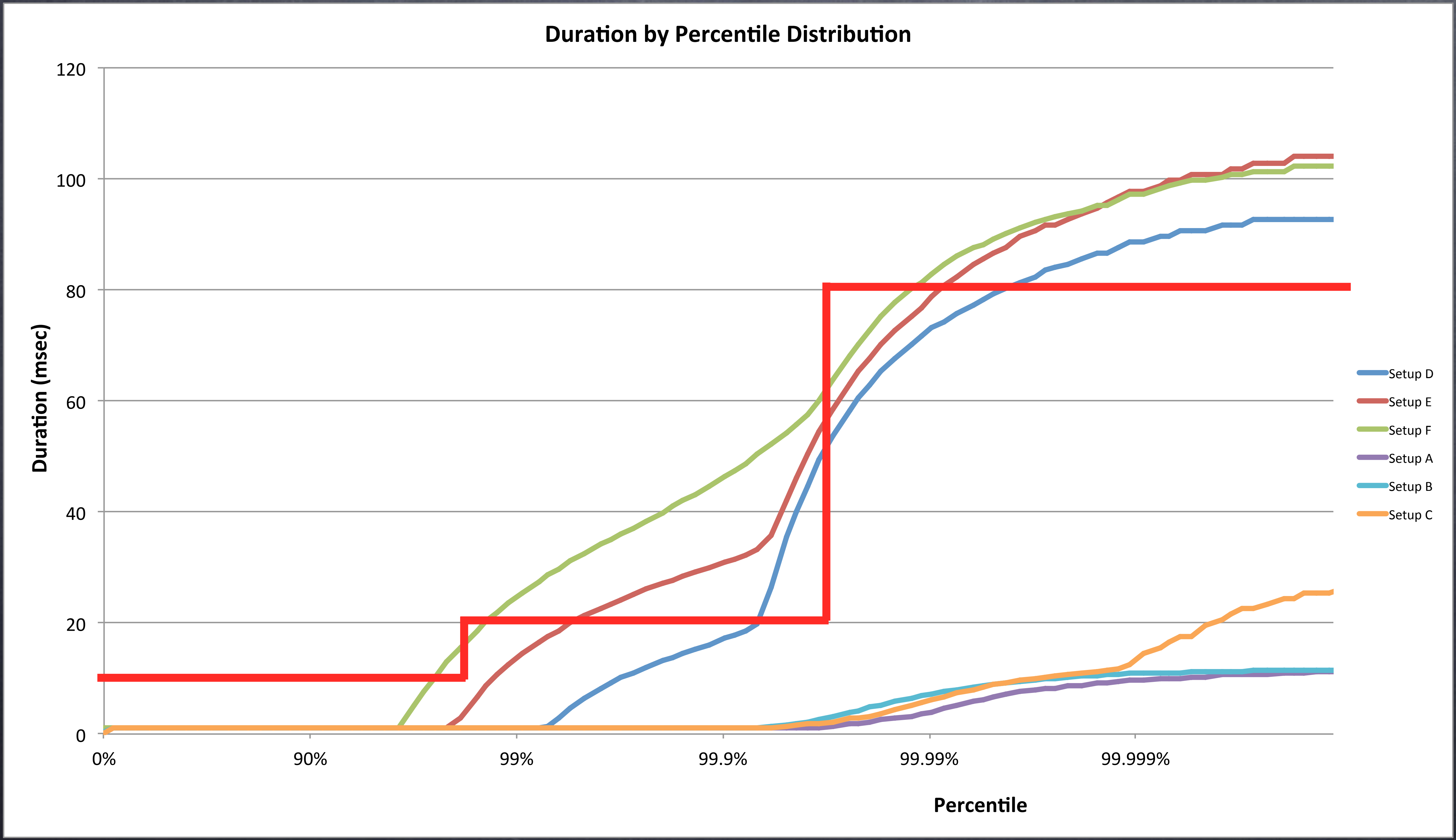
90K: Response Time By Percentile Distribution

80K: Response Time By Percentile Distribution

Latency doesn't live in a vacuum

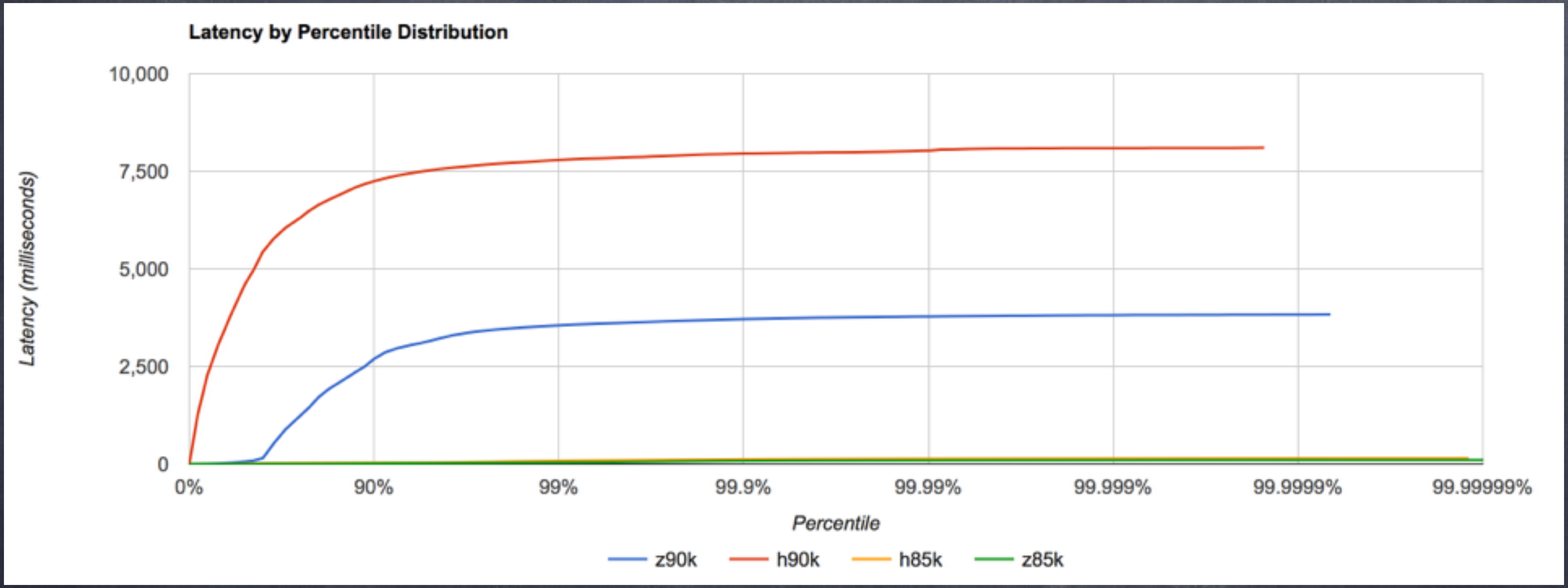
Sustainable Throughput: The throughput achieved while safely maintaining service levels

Comparing behavior under different throughputs and/or configurations



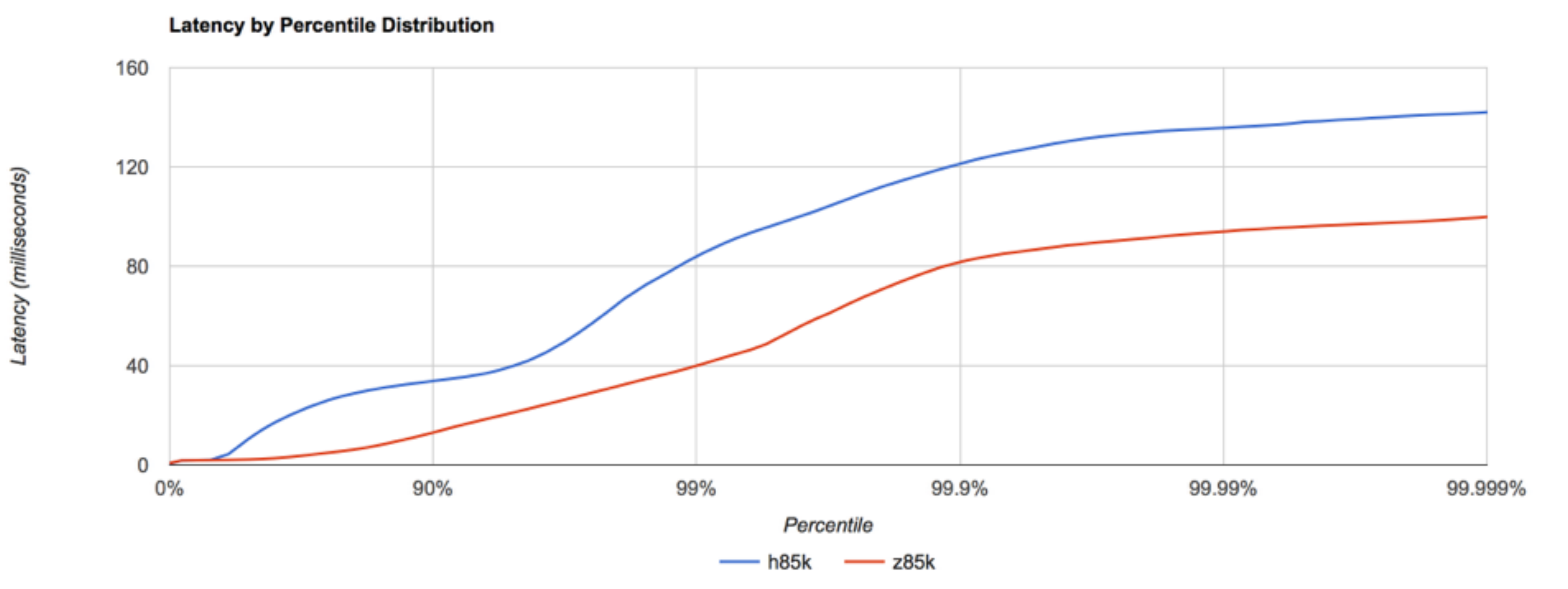
Comparing response time or latency behaviors

System A @90K/s & 85K/s vs. System B @90K/s & 85K/s



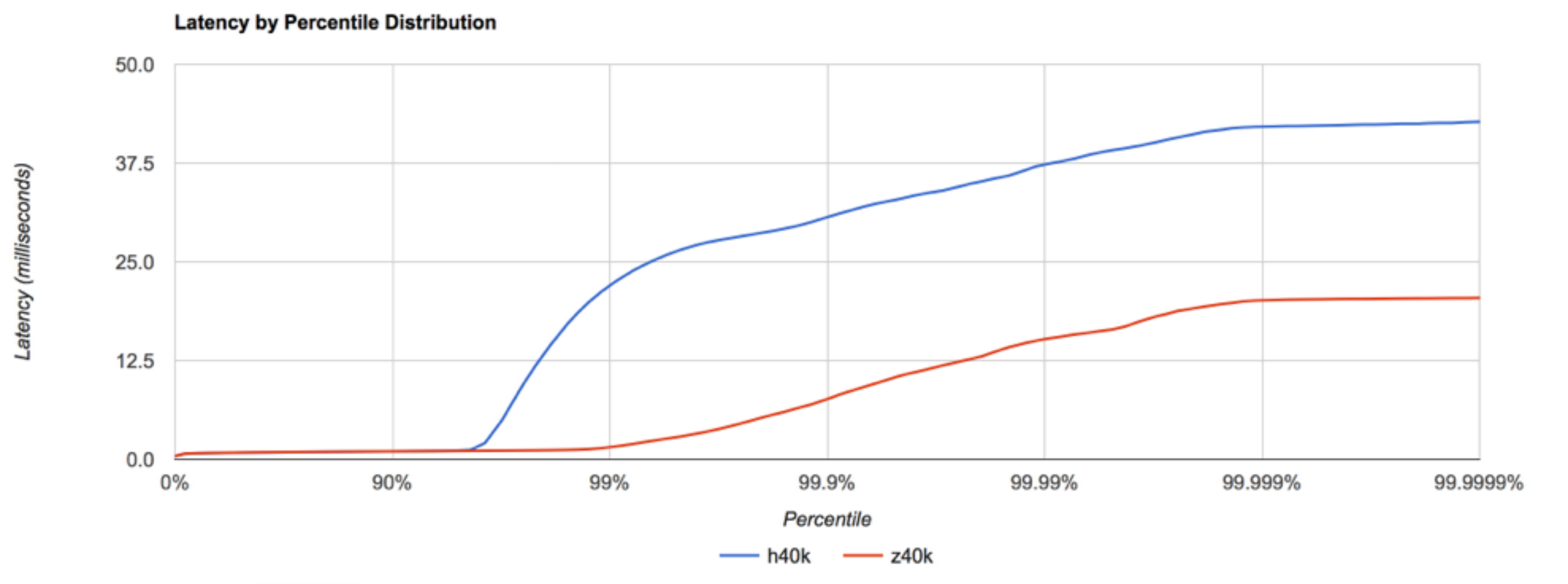
Wrong Place to Look:
They both “suck” at >85K/sec

System A 85K/s vs. System B 85K/s



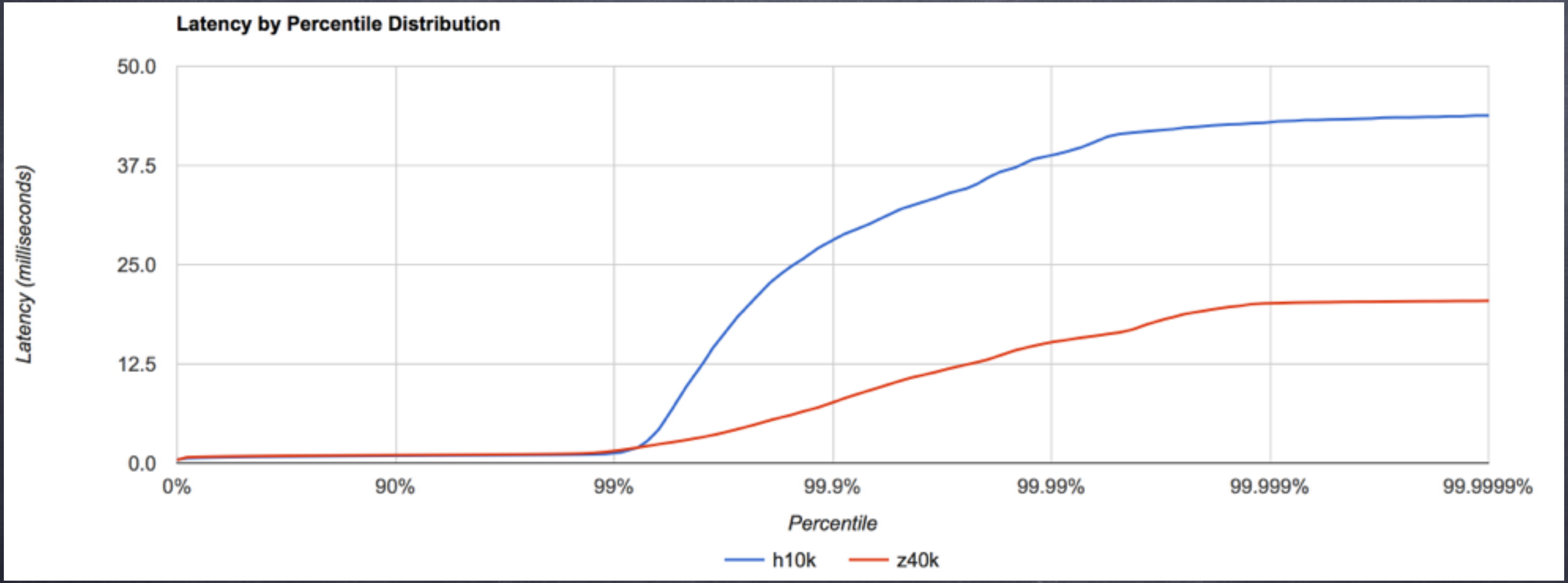
Looks good, but still
the wrong place to look

System A @40K/s vs. System B @40K/s



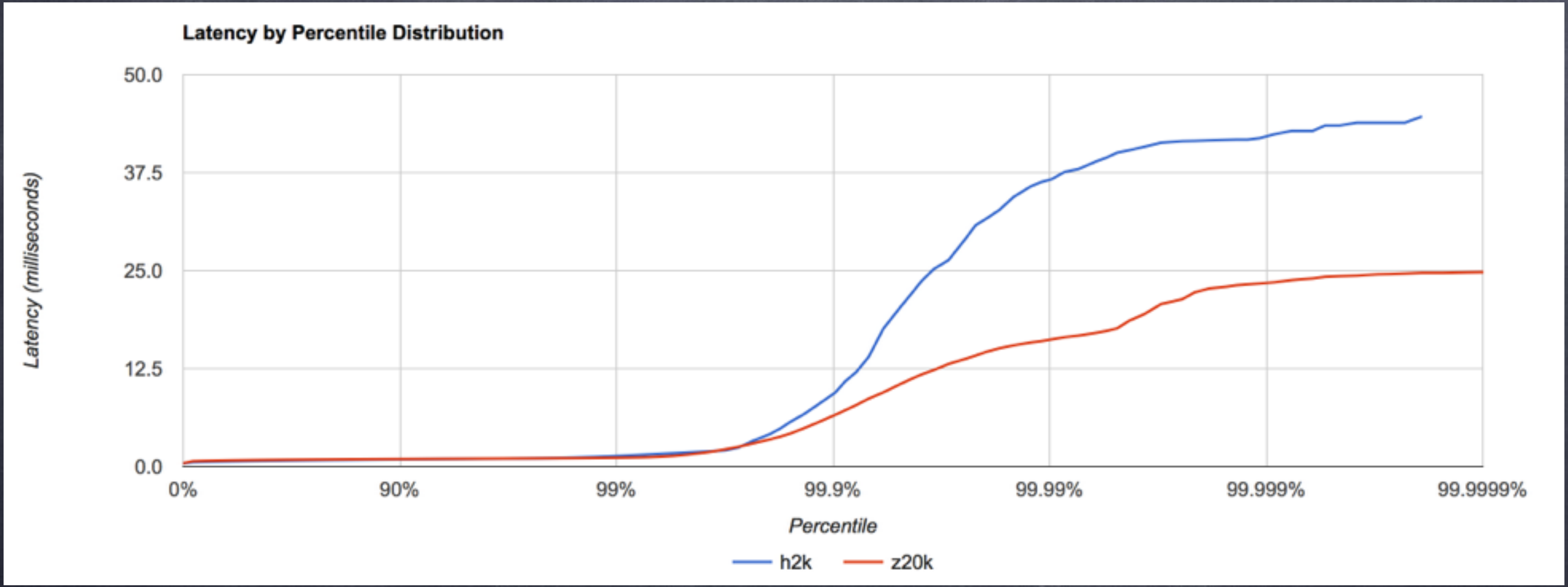
More interesting…
What can we do with this?

System A @10K/s vs. System B @40K/s



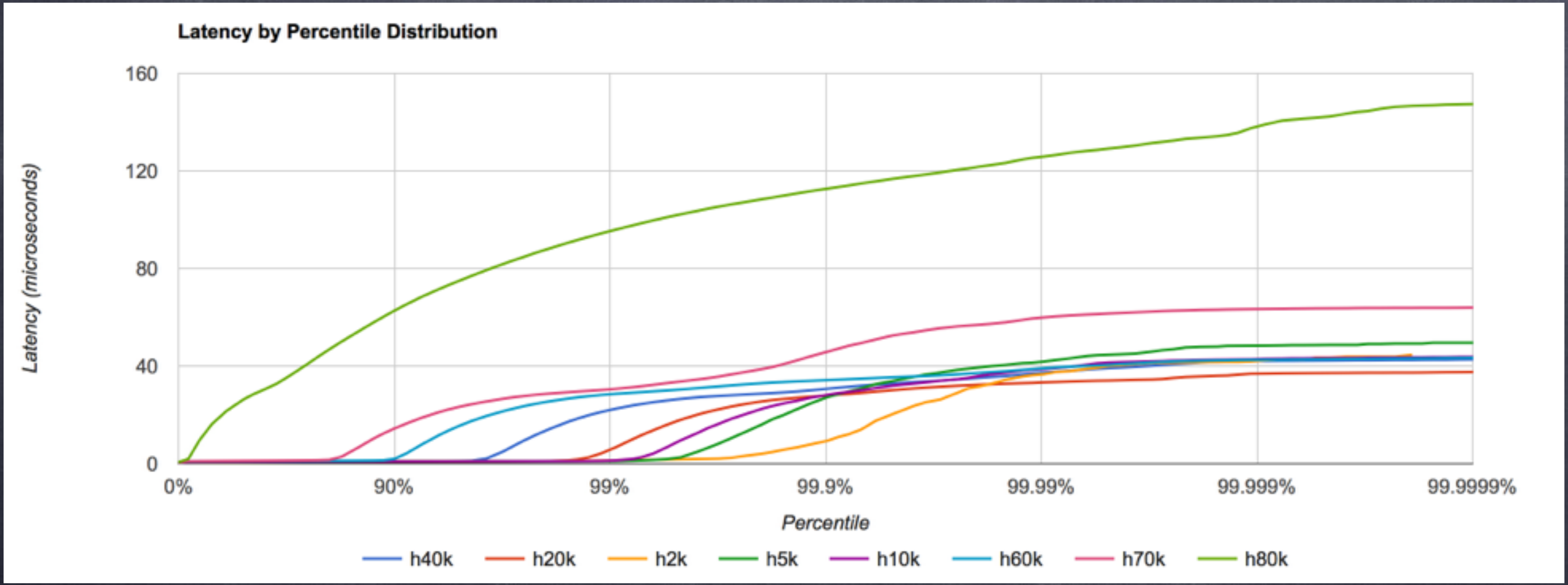
E.g. if “99%’ile < 5msec” was a goal:
System B delivers similar 99%’ile and superior 99.9%’ile+ while carrying 4x the throughput

System A @2K/s vs. System B @20K/s

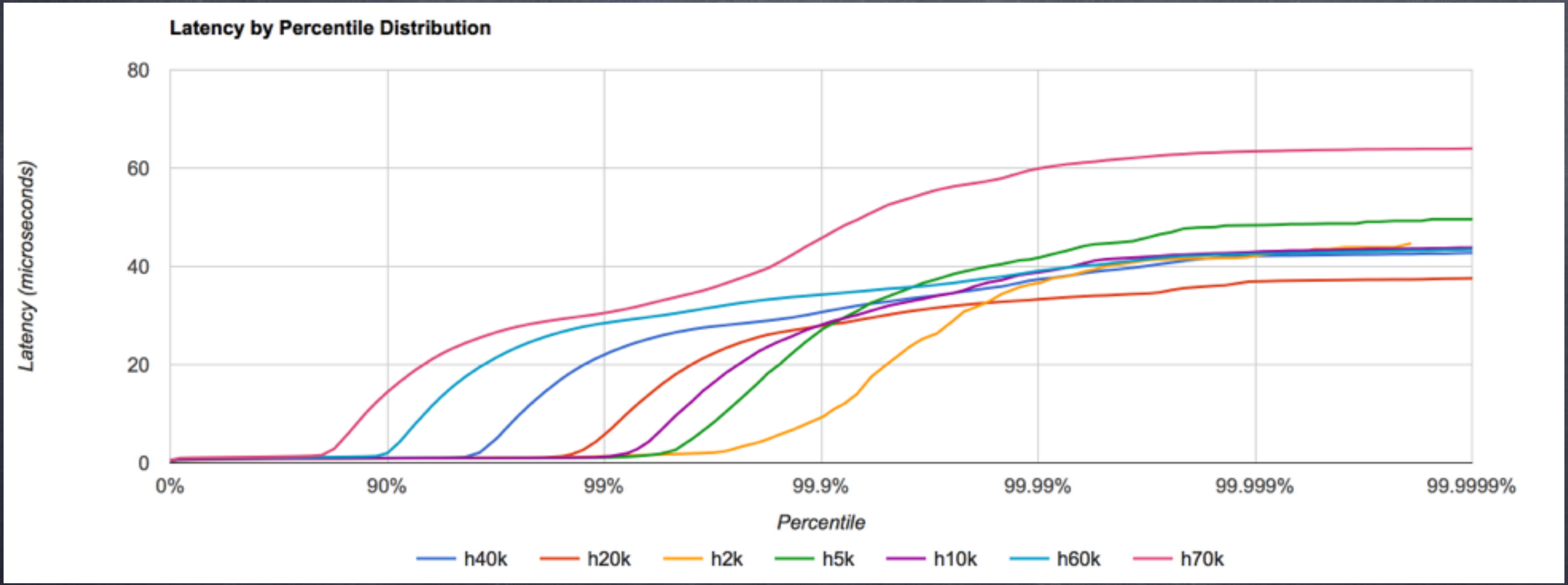


E.g. if “99.9%’ile < 10msec” was a goal:
System B delivers similar 99%’ile and 99.9%’ile
while carrying 10x the throughput

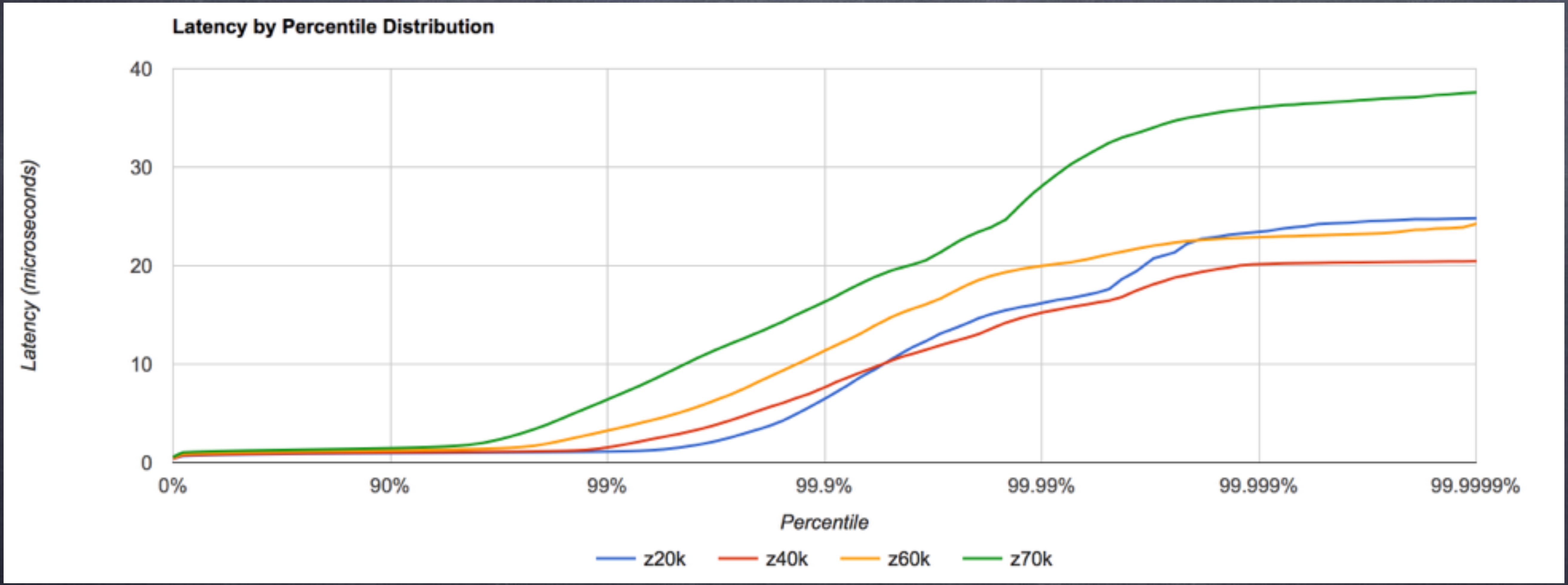
System A @2k thru 80k



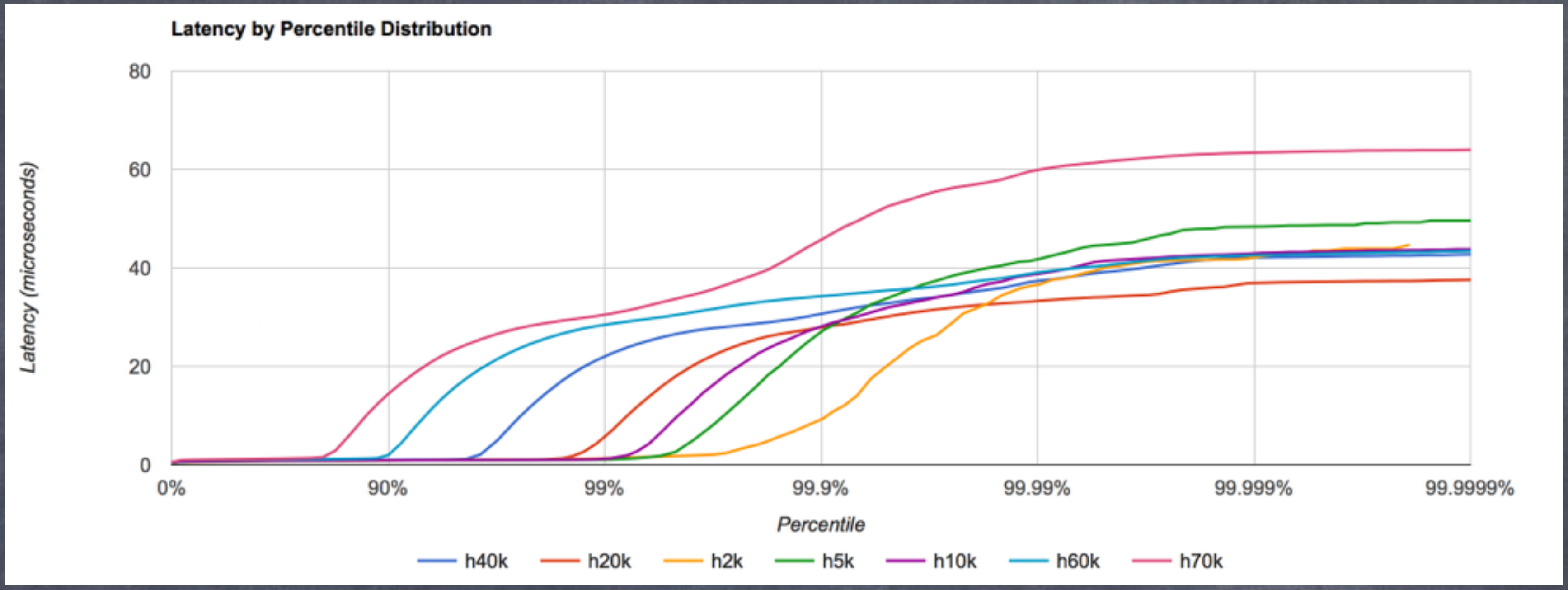
System A @2k thru 70k

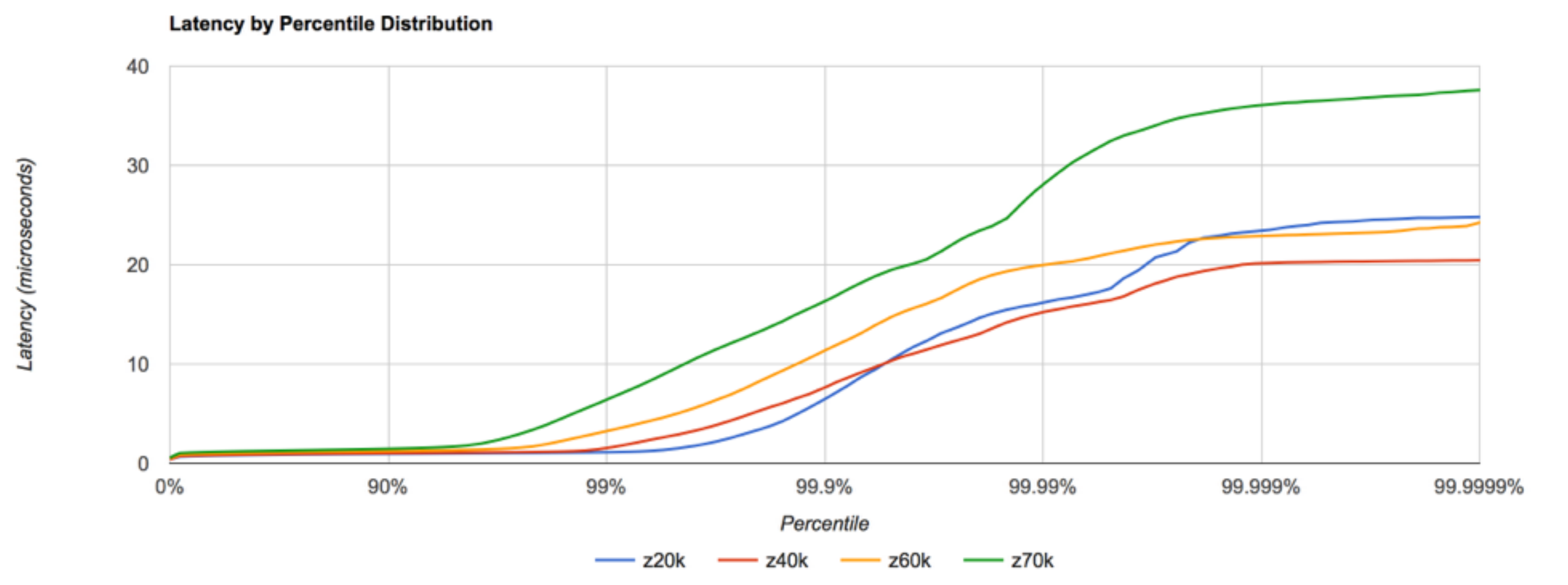


System B @20k thru 70k

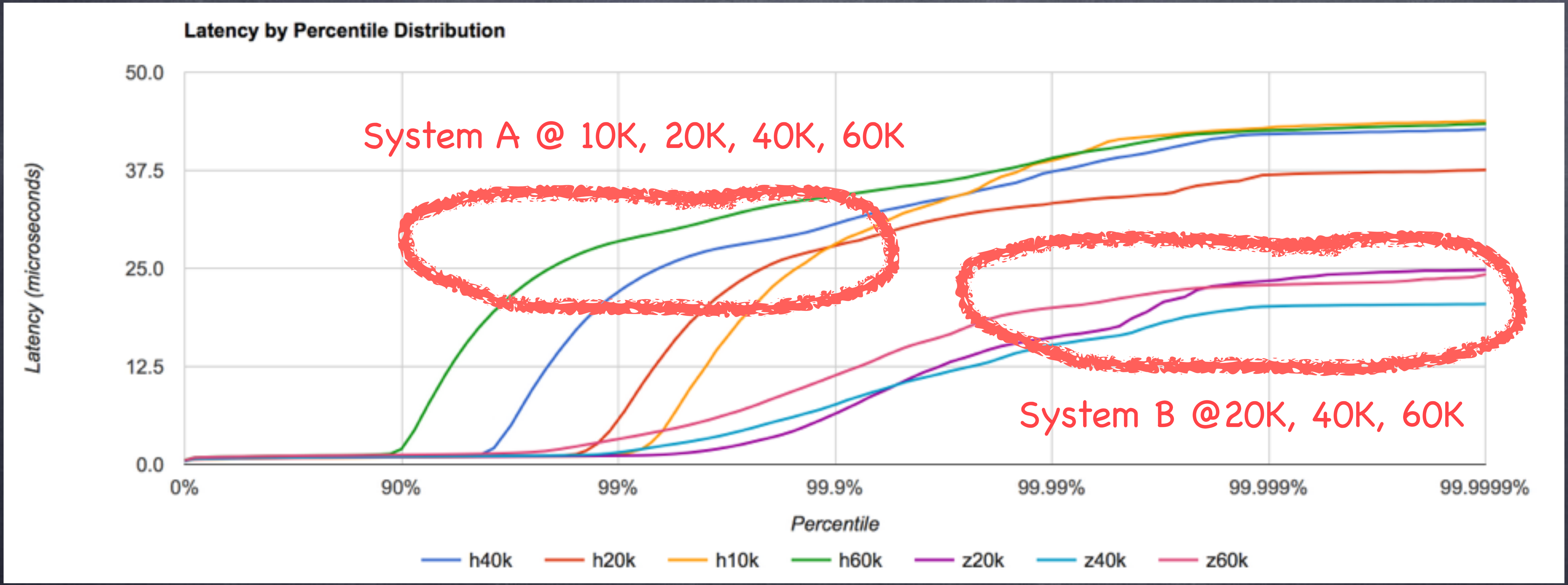


System A & System B @2k thru 70k



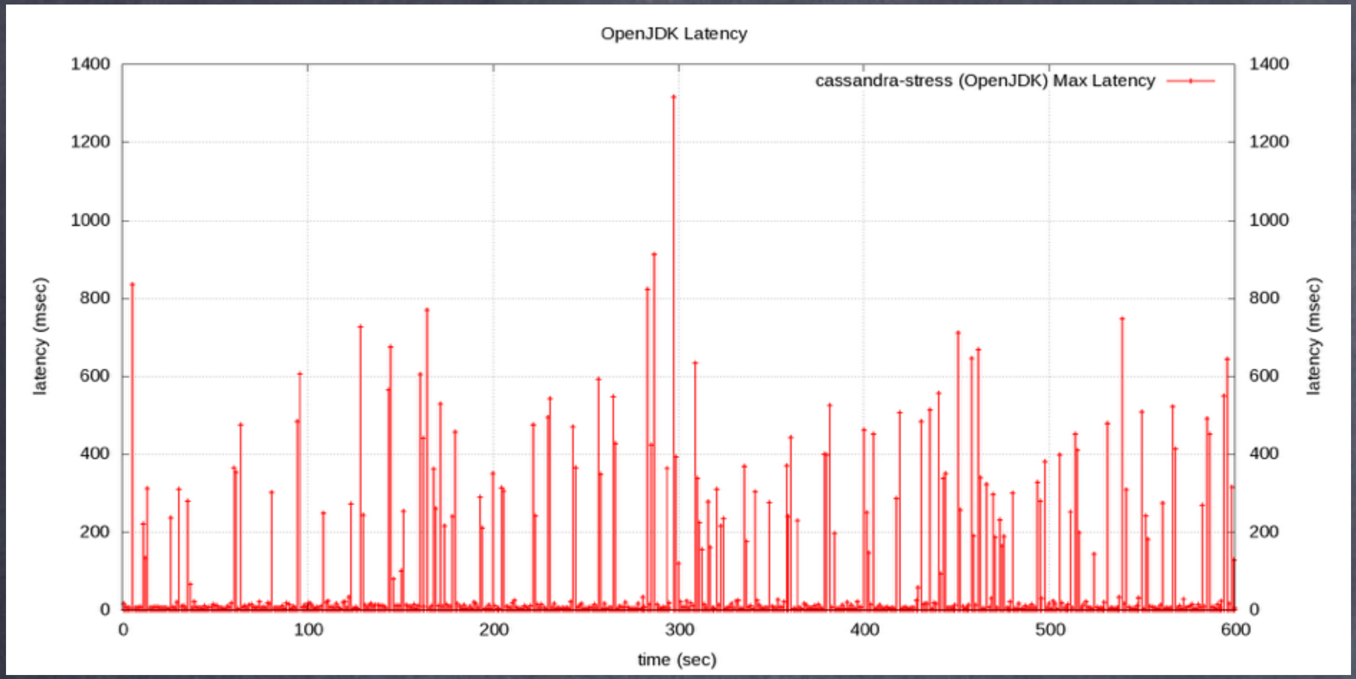


System A & System B 10K/s thru 60K/s

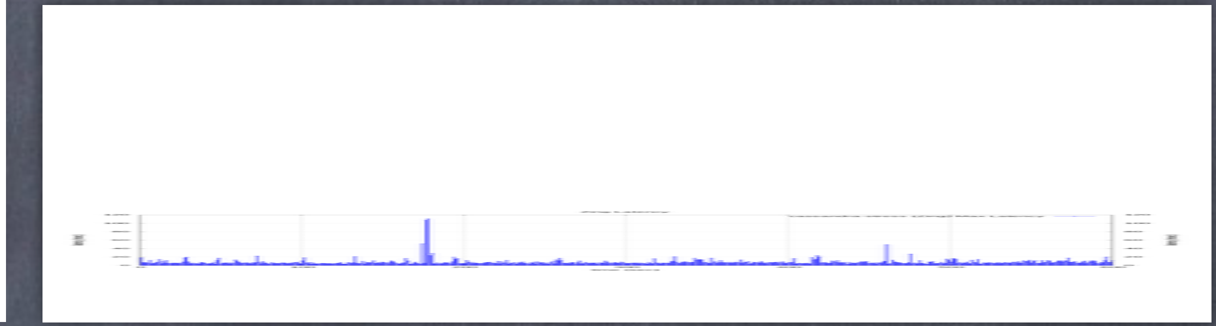


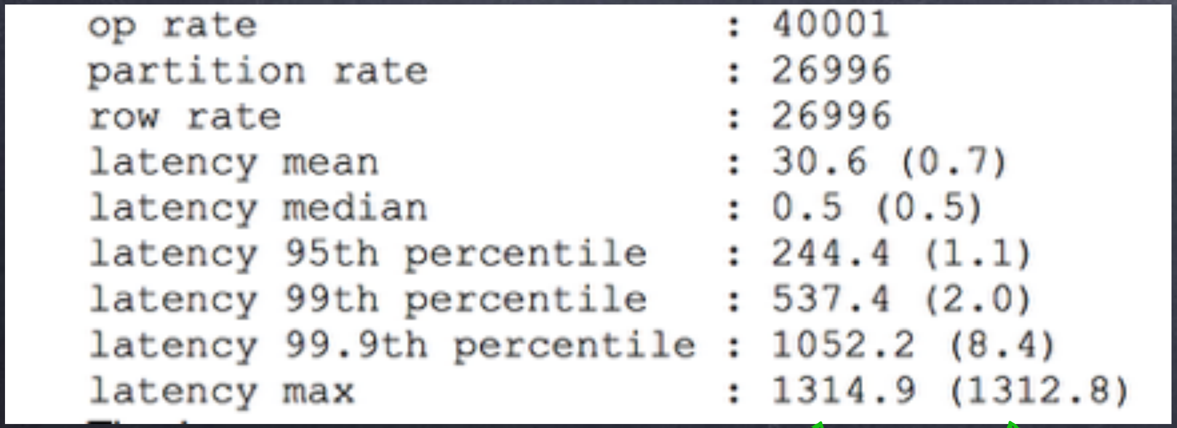
Lots of conclusions can be drawn from the above…
E.g. System B delivers a consistent 100x reduction in the rate of occurrence of >20msec response times

System A: 200-1400 msec stalls



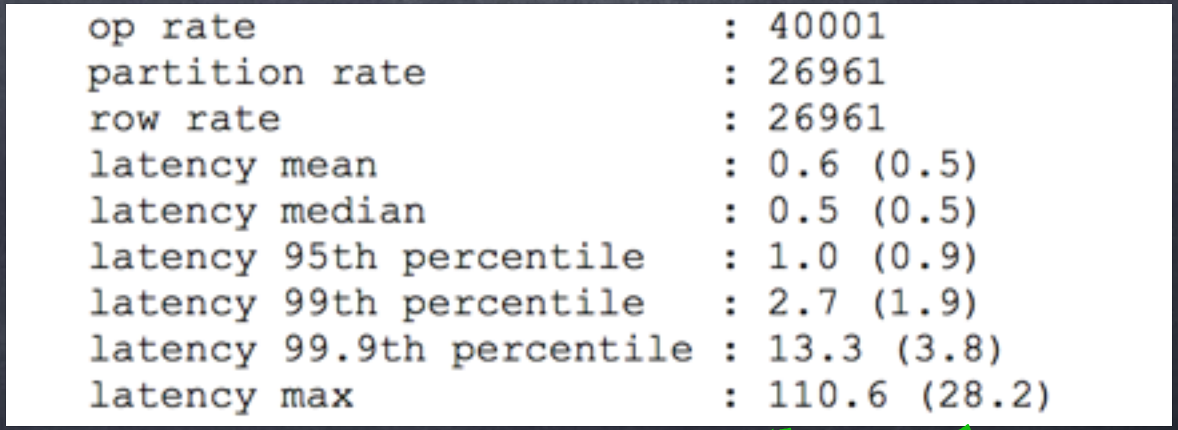
System B drawn to scale





```
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

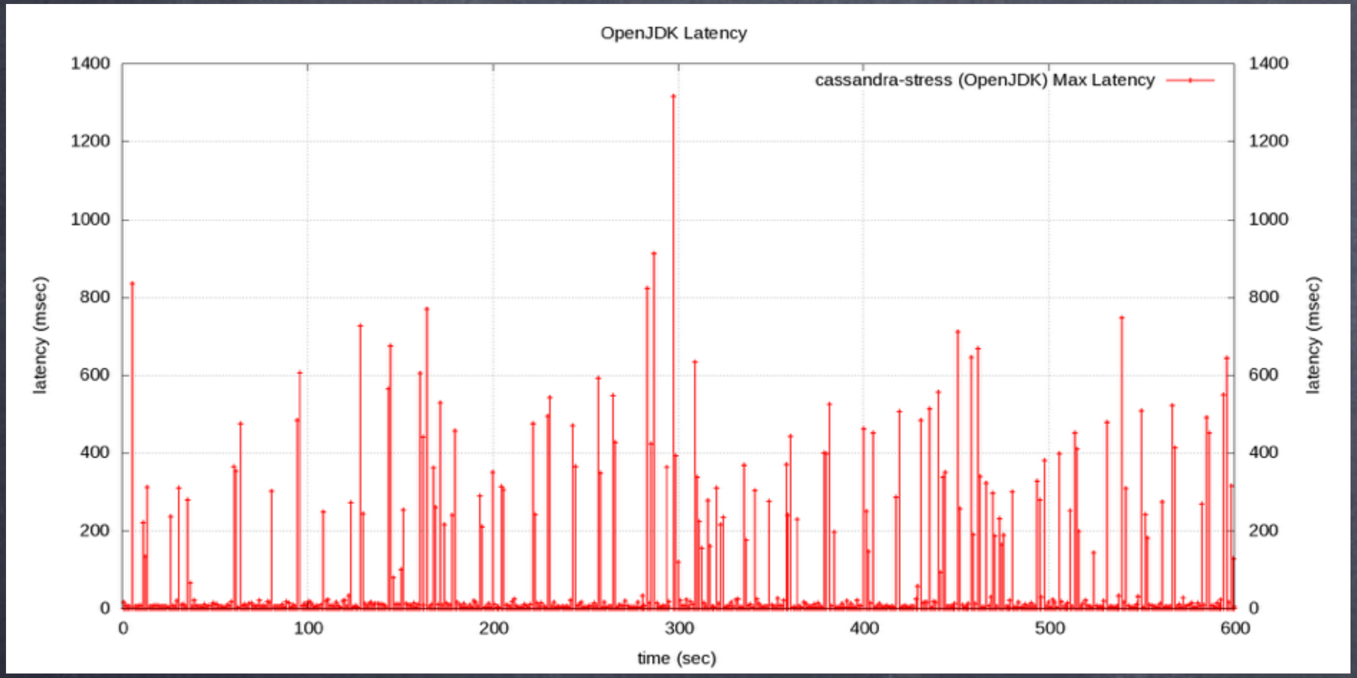


```
op rate                      : 40001
partition rate               : 26961
row rate                     : 26961
latency mean                 : 0.6 (0.5)
latency median               : 0.5 (0.5)
latency 95th percentile      : 1.0 (0.9)
latency 99th percentile      : 2.7 (1.9)
latency 99.9th percentile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

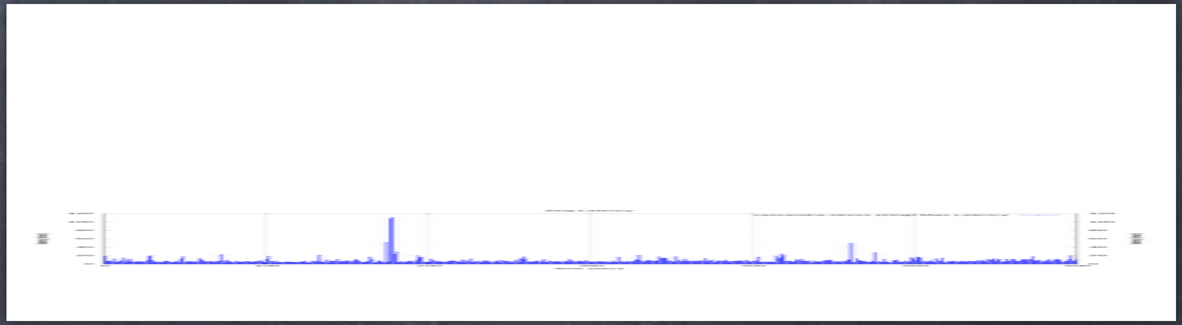
Response Time   Service time

A simple visual summary

This is Your Load on System A



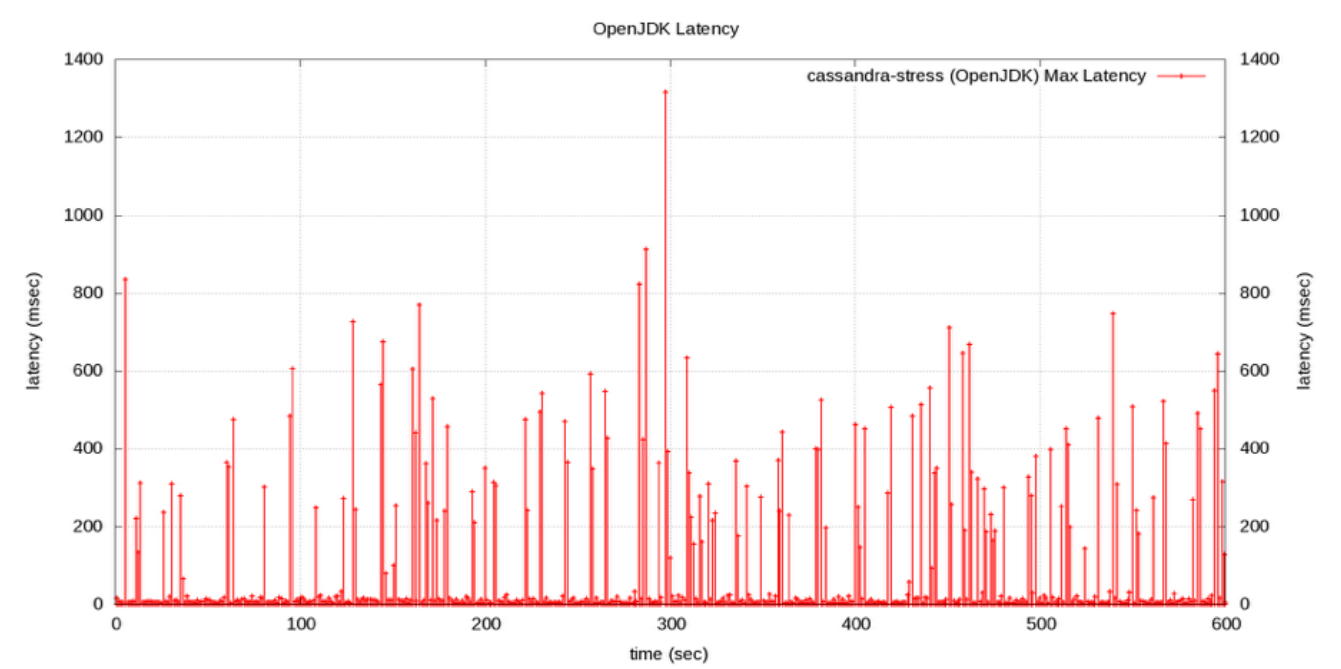
This is Your Load on System B



Any Questions?

Any Questions?

http://www.azulsystems.com


OpenJDK Latency
cassandra-stress (OpenJDK) Max Latency
latency (msec)
time (sec)