Update on *t*-digest

Ted Dunning

Contact Information

Ted Dunning, PhD

Chief Application Architect, MapR Technologies

Board member, Apache Software Foundation

O'Reilly author

Email tdunning@mapr.com tdunning@apache.org

Twitter @ted_dunning

Who We Are

- MapR echnologies
  - We make a kick-ass platform for big data computing
  - Support many workloads including Hadoop / Spark / HPC / Other
  - Extended to allow streams and tables in basic platform
  - Free for academic research / training
- Apache Software Foundation
  - Culture hub for building open source communities
  - Shared values around openness for contribution as well as use
  - Many major projects are part of Apache
  - Even more minor ones!

Basic Outline

- Why we should measure distributions
- Basic Ideas
- How *t*-digest works
- Recent results
- Applications

Why Is This Practically Important

- The novice came to the master and says “something is broken”

Why Is This Practically Important

- The novice came to the master and says “something is broken”

- The master replied “What has changed?”

Why Is This Practically Important

- The novice came to the master and says “something is broken”
- The master replied “What has changed?”
- And the student was enlightened

Finding change is key

but what kind?

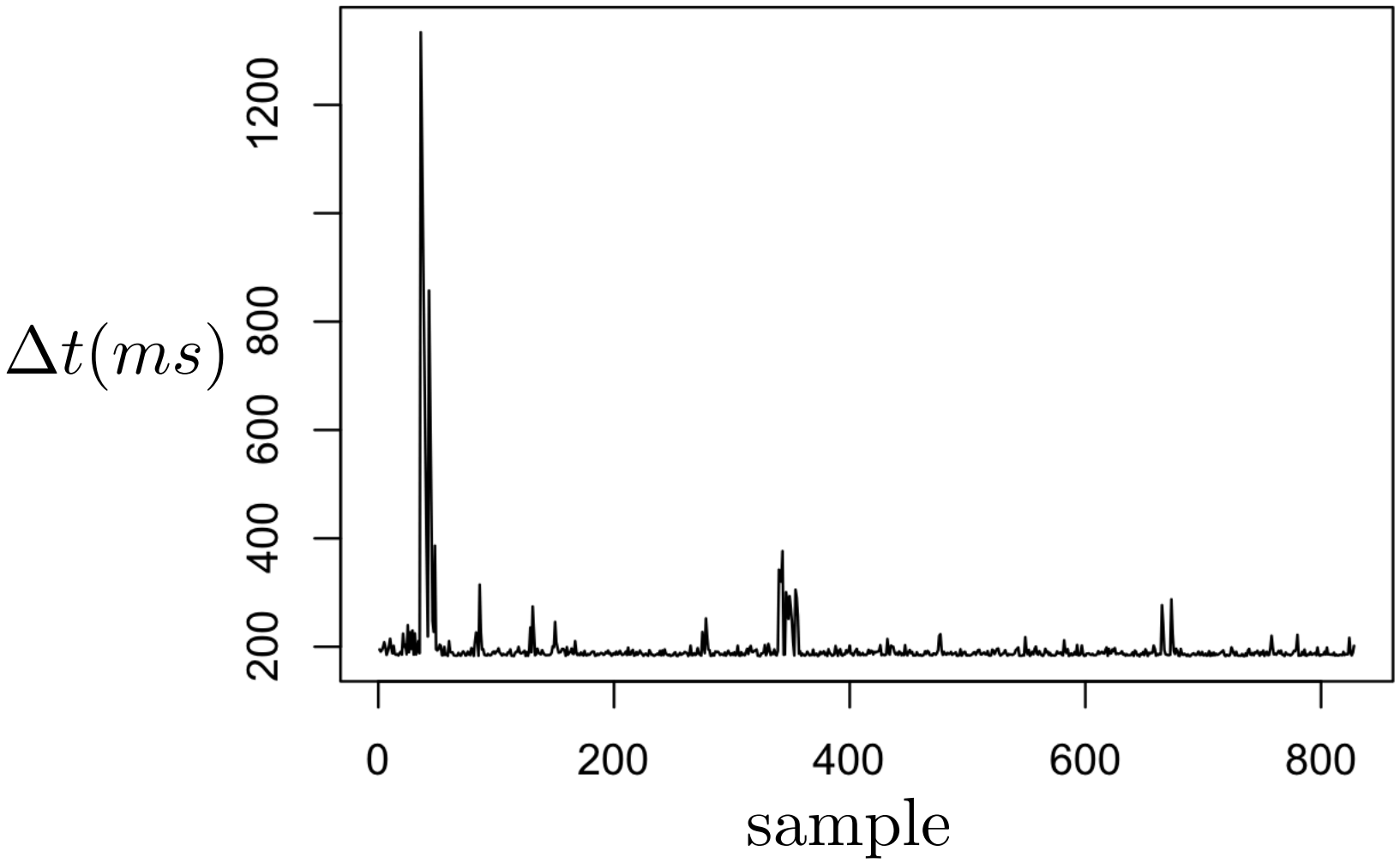
Last Night's Latencies

- These are ping latencies from my hotel
- Looks pretty good, right?
- But what about longer term?

```
> mean(y$t[i])
[1] 198.6047
> sd(y$t[i])
[1] 71.43965
```

```
208.302
198.571
185.099
191.258
201.392
214.738
197.389
187.749
201.693
186.762
185.296
186.390
183.960
188.060
190.763
```

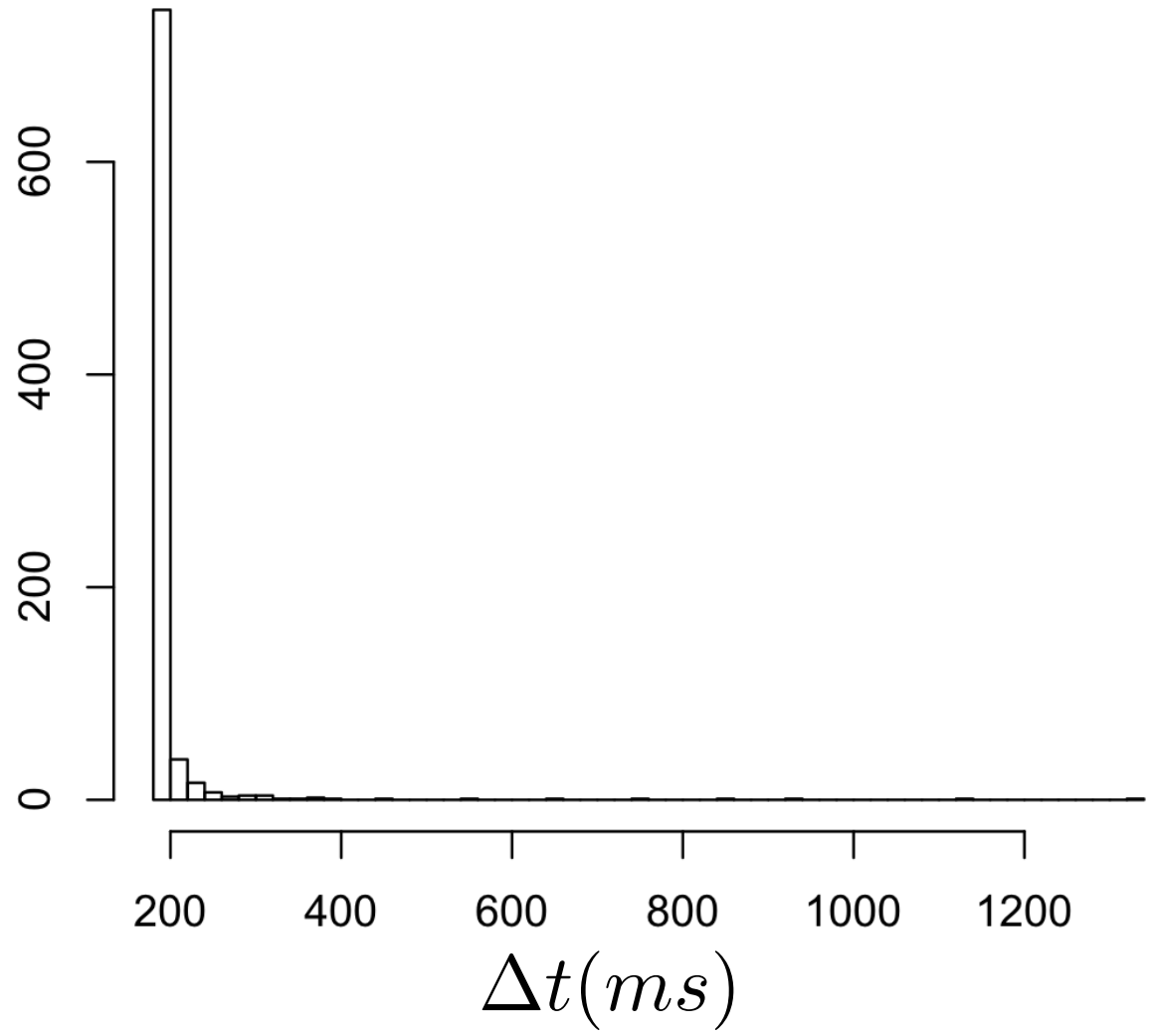
Not So Fast …



This is long-tailed land

This is long-tailed land

You have to know the distribution of values


0
200
400
600
200
400
600
800
1000
1200
$\Delta t(ms)$

A single number
is simply not enough

What We Really Need Here

- I want to be able to compute the distribution from any time period
- From any subset of measurements
- With lots of keys and filters
- And not a lot of space

- Basically, any OLAP kind of query

```
select distribution(x) from … where … group by y,z
```

Idea 0 – Pre-defined bins

- So let's assume we have bins
  - Upper, lower bound, constant width
- Get a measurement, pick a bin, increment count
- Works great if you know the data
  - And you have limited dynamic range (too many bins)
  - And the distribution is fixed
- Useful, but not general enough

Idea 1 – Exponential Bins

- Suppose we want relative accuracy in measurement space
- Latencies are positive and only matter within a few percent
  - 1.1 ms versus 1.0 ms
  - 1100 ms versus 1000 ms

- We can cheat by using floating point representations
  - Compute bin using magic
  - Count

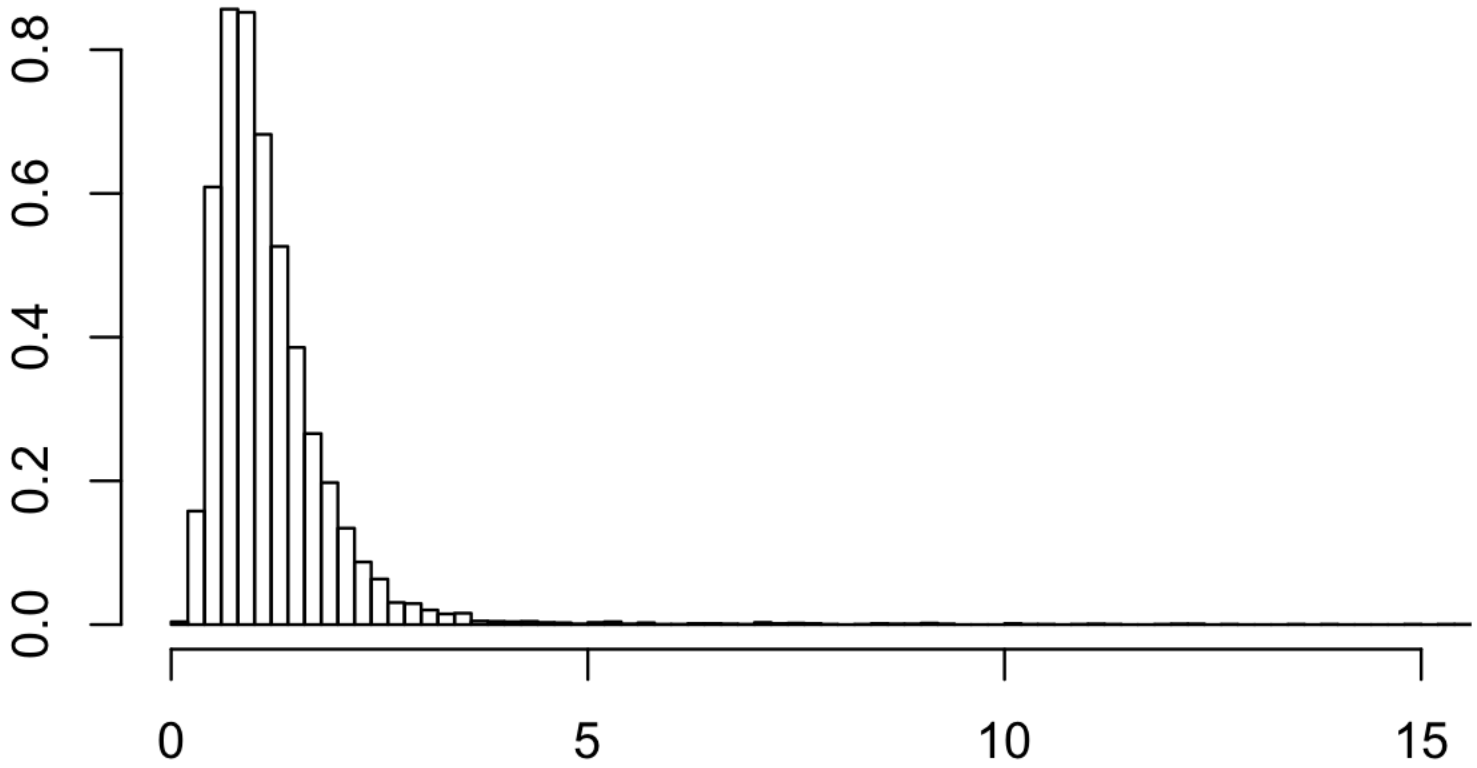
FloatHistogram

- Assume all measurements are in the range $[x_{\mathtt{min}}, x_{\mathtt{max}}]$
- Divide this range into power of 2 sub-ranges
- Sub-divide each sub-range evenly with 2^k steps
  - $k = 3$ is typical
- Relative error is bounded in measurement space

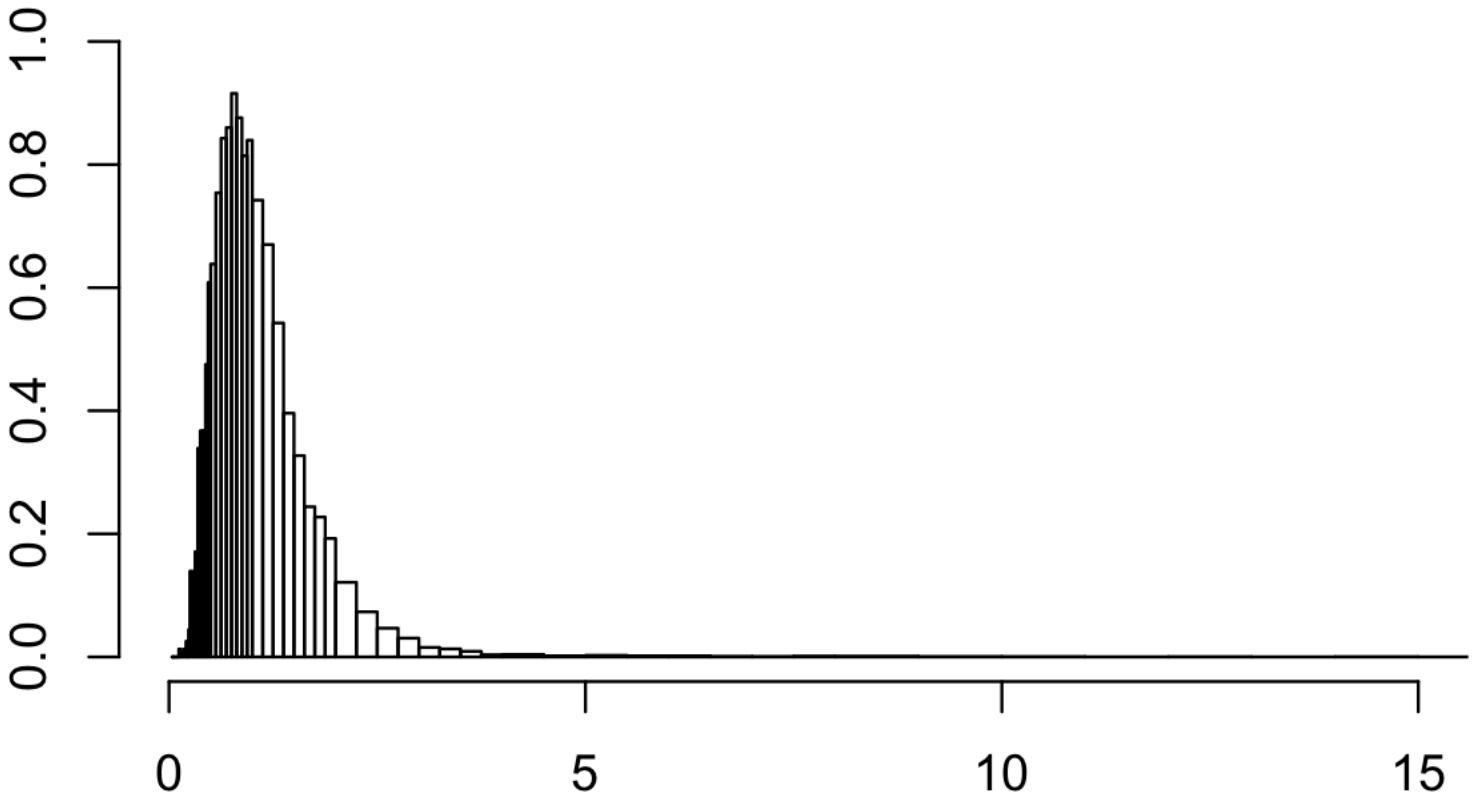
FloatHistogram

- Assume all measurements are in the range $[x_{\mathtt{min}}, x_{\mathtt{max}}]$
- Divide this range into power of 2 sub-ranges
- Sub-divide each sub-range evenly with 2^k steps
  - $k = 3$ is typical
- Relative error is bounded in measurement space
- Bin index can be computed using FP representation!

Fixed Size Bins



Approximate Exponential Bins



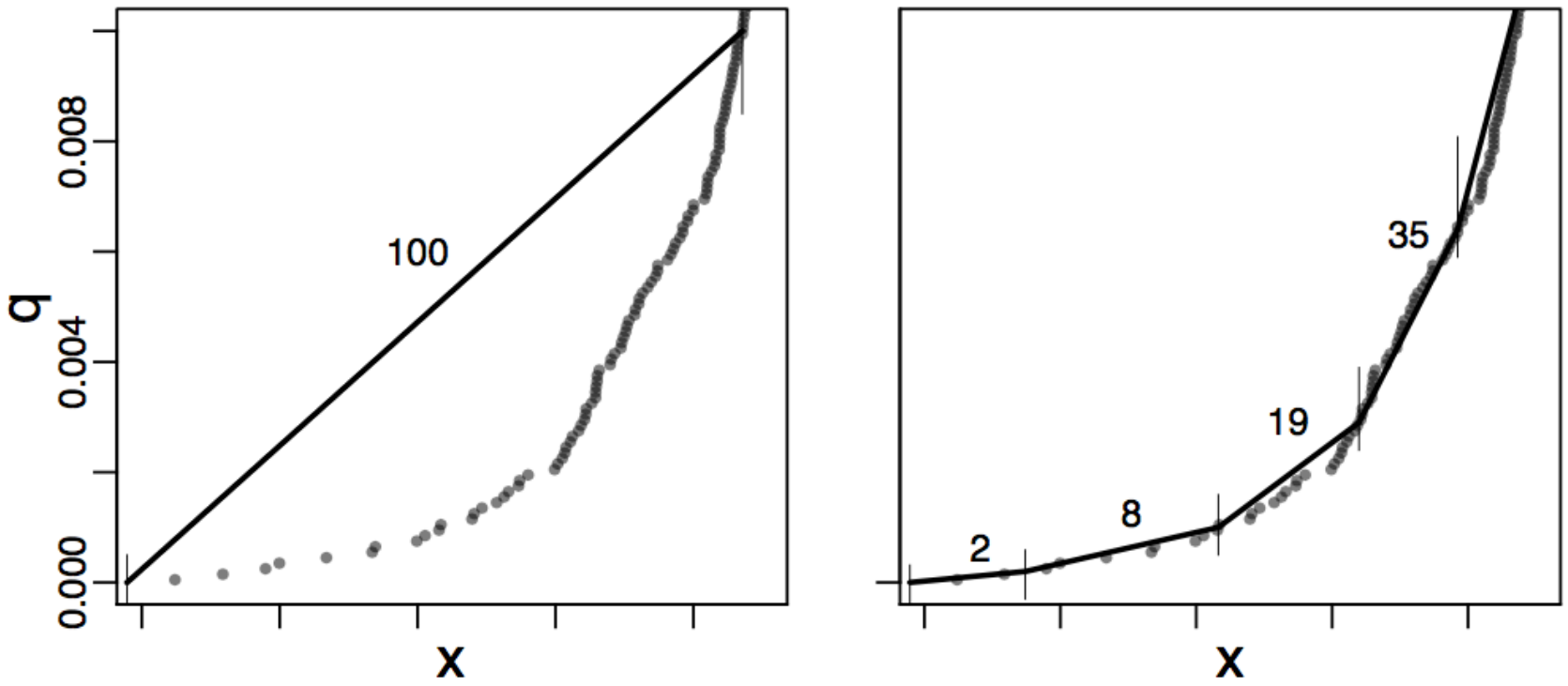
Non-linear bins are better (sometimes)

Still not general enough

Idea 2 – Fully Adaptive Bins

- First intuition – in general, we want accuracy in terms of percentile
- Second intuition – we want better accuracy at extreme quantiles
  - 50%-ile versus 50.1%-ile?
  - What does 0.1% error even mean for 99.99th percentile

- We need bins with small counts near the edges



First 1% of data shown.
Left graph has 100 x 100 sample bins.
Right graph has ~130bins, variable size

The Basic *t*-digest

- Take a bunch of data
- Sort it
- Group into bins
  - But make the bins be smaller at the beginning and end

- Remember the centroid and count of each bin

- That's ***a*** *t*-digest

But Wait, You Need a Bit More

- Take a bunch of new data, old *t*-digest
- Sort the data and the old bins together
- Group into bins
  - Note that existing bins have bigger weights
  - So they might survive … or might clump

- Remember the centroid and count of each new bin

- That's ***an updated*** *t*-digest

Oh … and Merging

- Take a bunch of old *t*-digests
- Sort the bins
- Group into mega-bins
  - Respect the size constraint

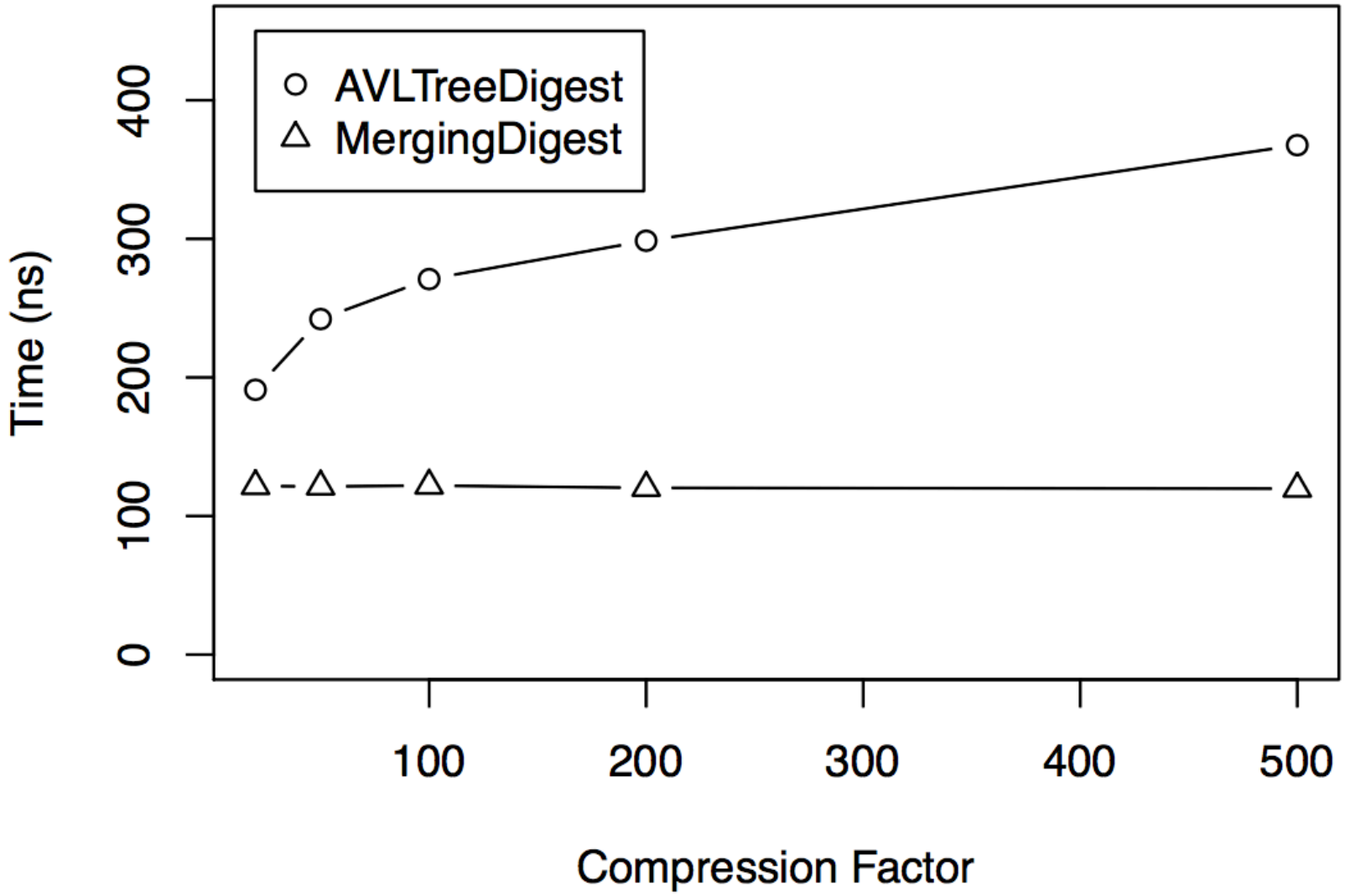
- Remember the centroid and count of each new bin

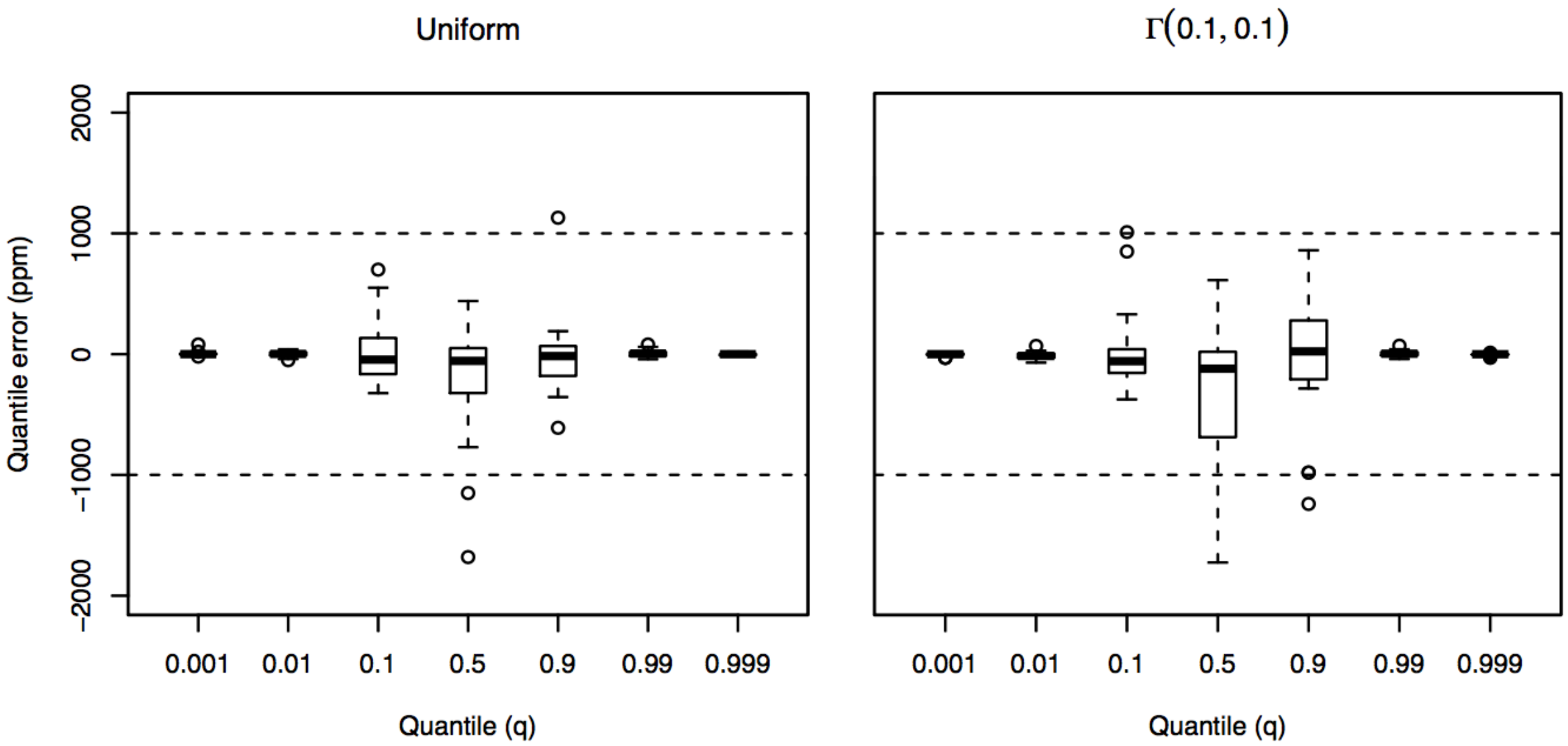
- That's ***a merged*** *t*-digest

Adaptive non-linear bins
are good and general

And can be grouped
and regrouped

Results





Status

- Current release
  - Small accuracy bugs in corner cases
  - Best overall is still AVLTreeDigest

Status

- Current release (3.x)
  - Small accuracy bugs in corner cases
  - Best overall is still AVLTreeDigest

- Upcoming release (4.0)
  - Better accuracy in pathological cases
  - Strictly bounded size
  - No dynamic allocation (with MergingDigest)
  - Good speed (100ns for MergingDigest, 5ns for FloatHistogram)
  - Real Soon Now

Example Application

- The data:
  - ~ 1 million machines
  - Even more services
  - Each producing thousands of measurements per second
- Store *t*-digest for each 5 minute period for each measurement

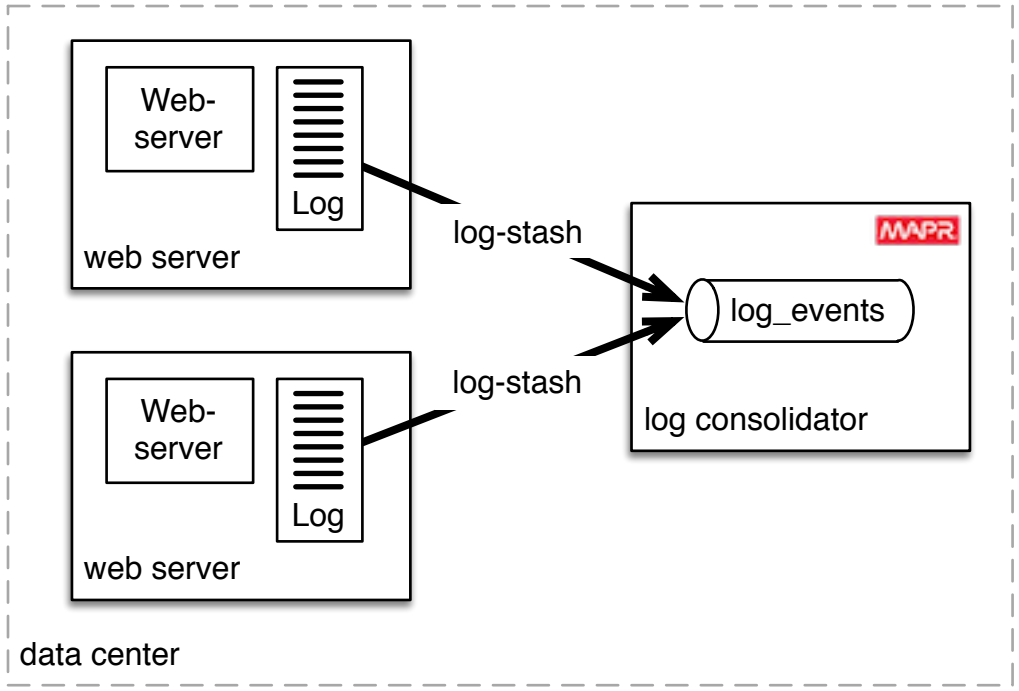
- Want to query any combination of keys, produce t-digest result

  “what was the distribution of launch times yesterday?”

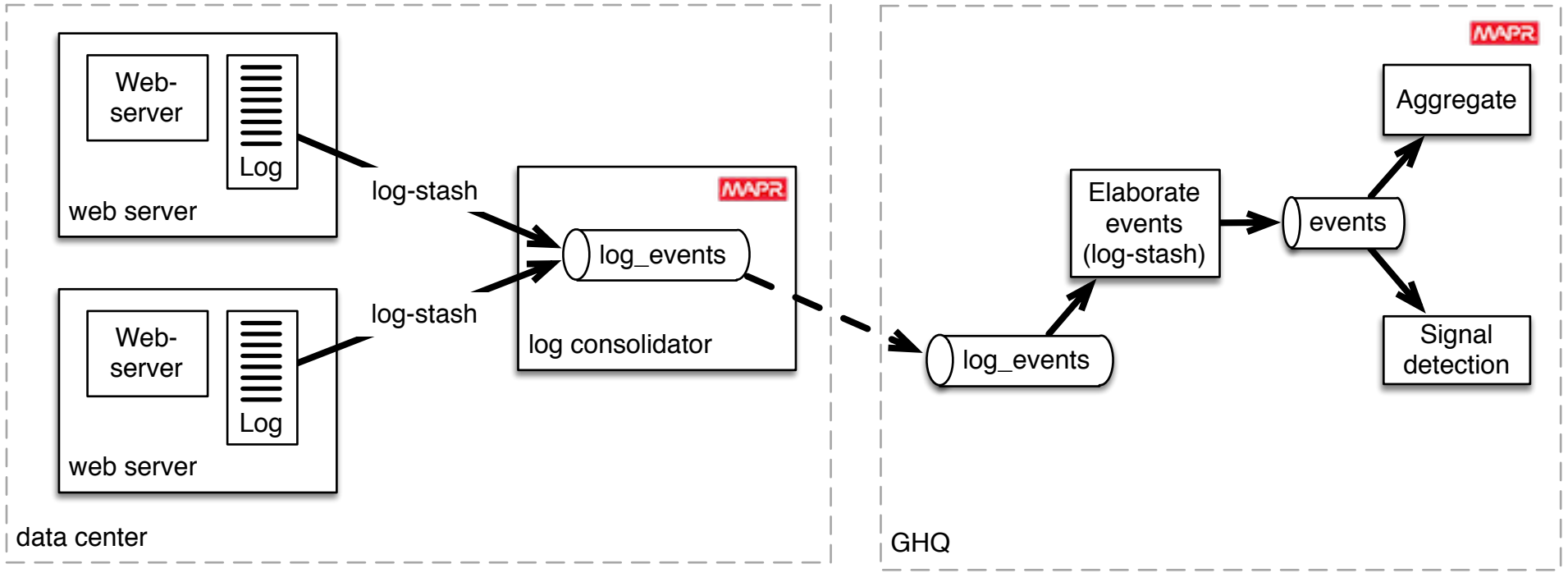
  “what about last month?”

  “in Europe versus in North America versus in Asia?”

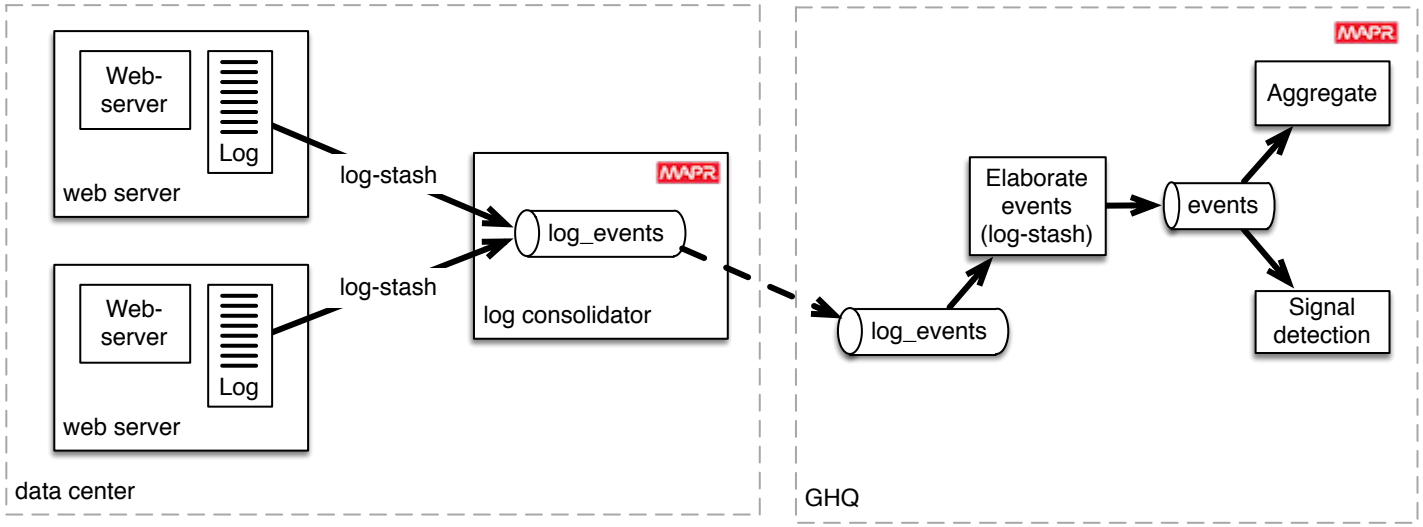
Collect Data



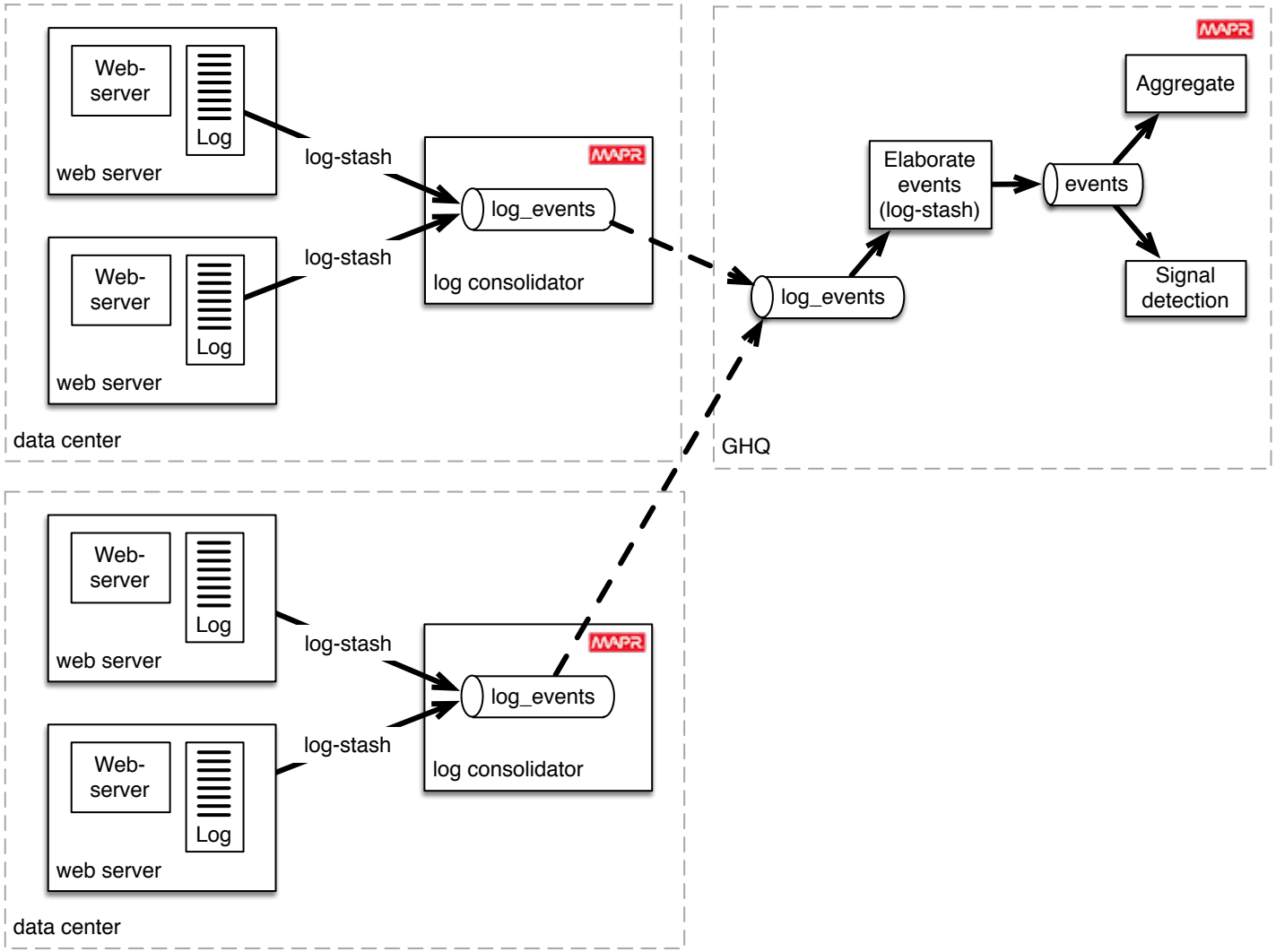
And Transport to Global Analytics



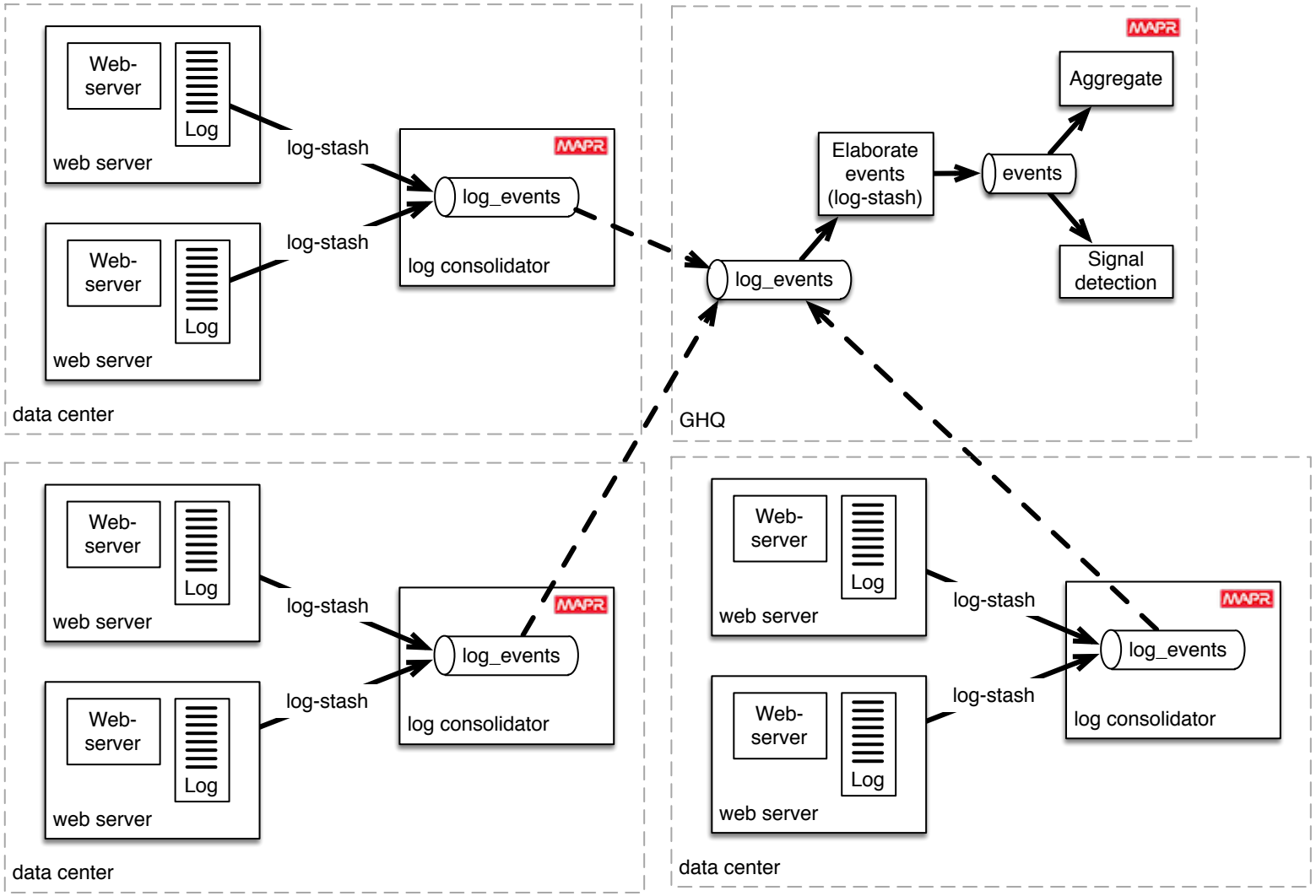
With Many Sources



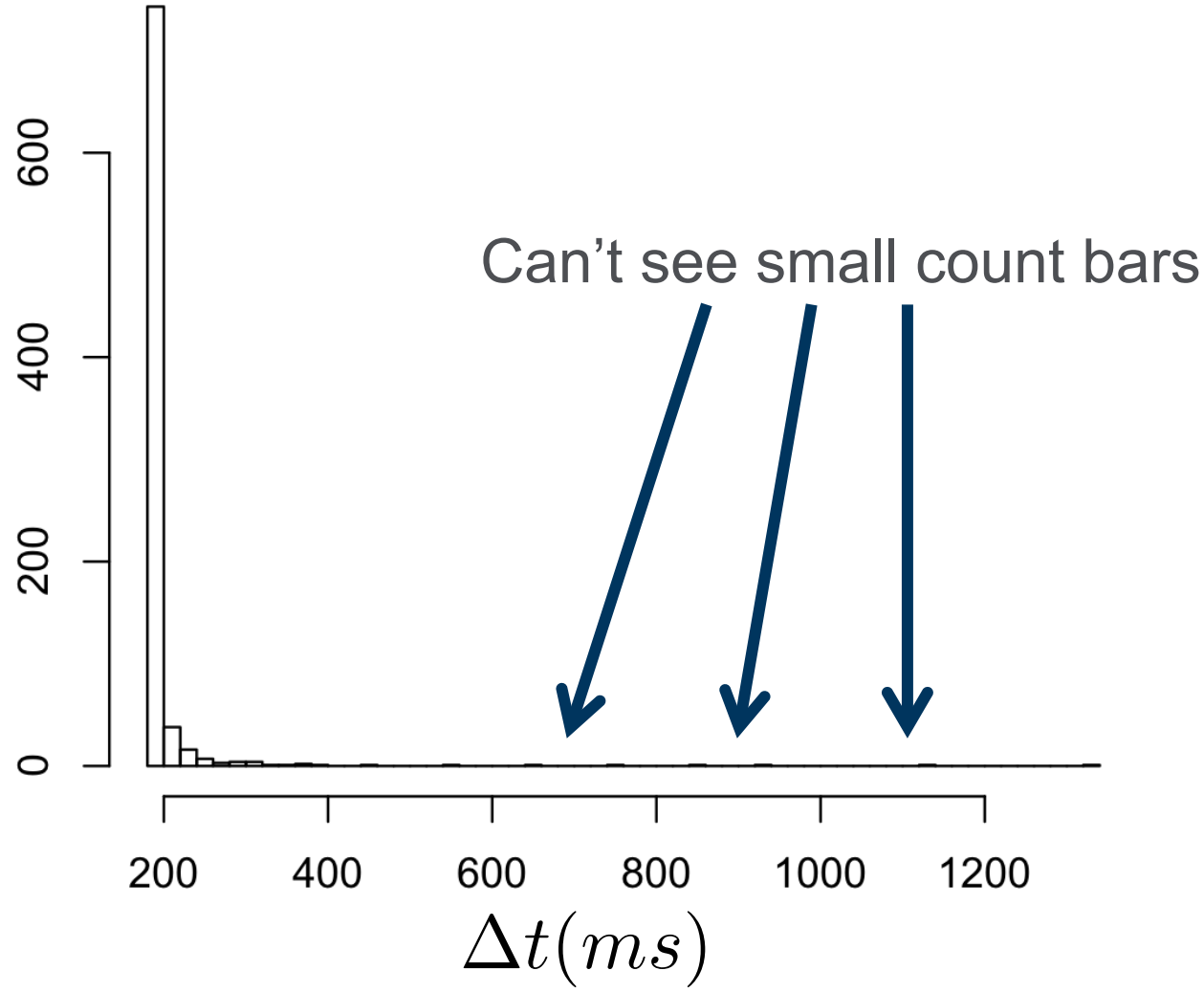
With Many Sources



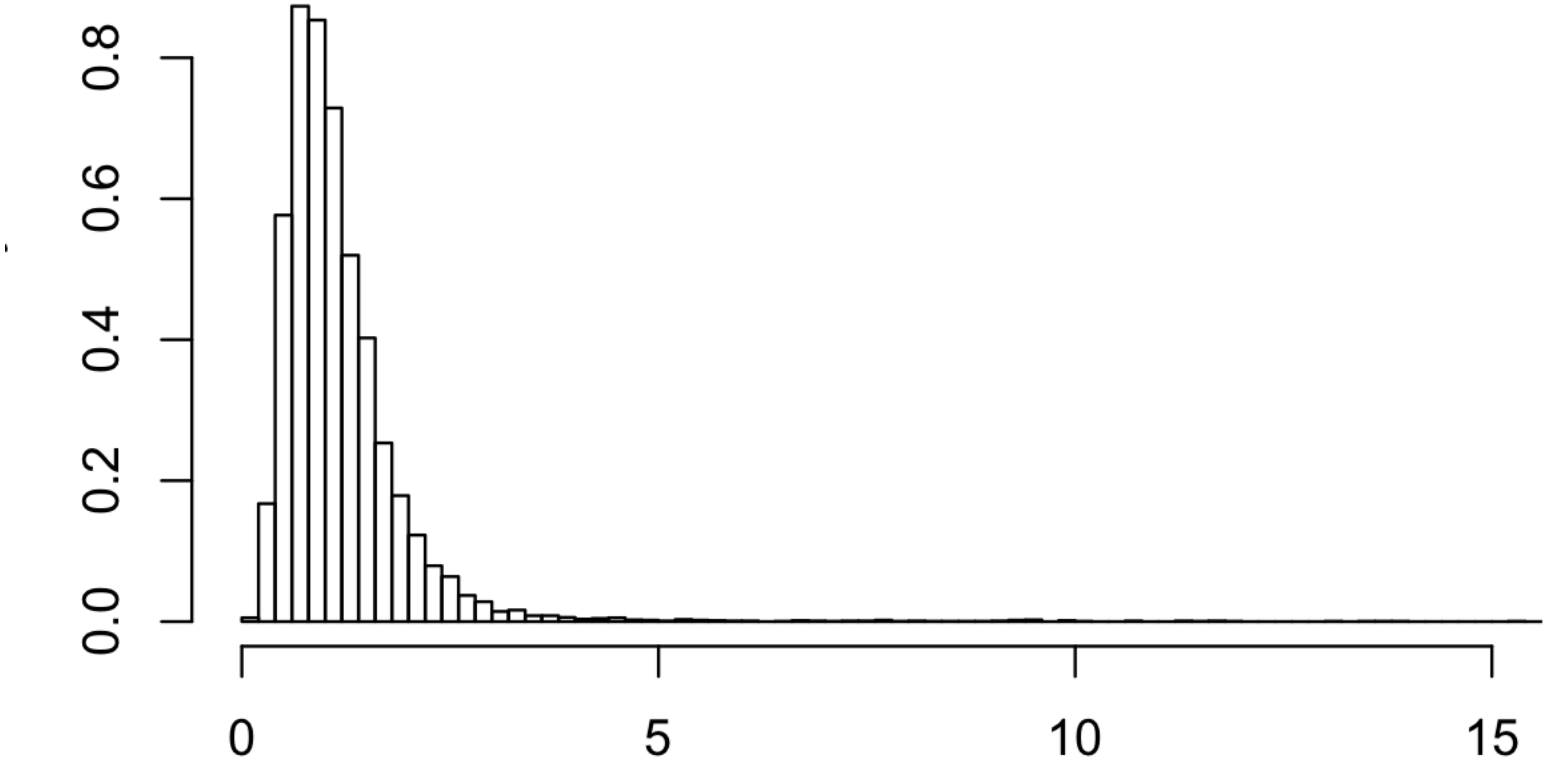
With Many Sources



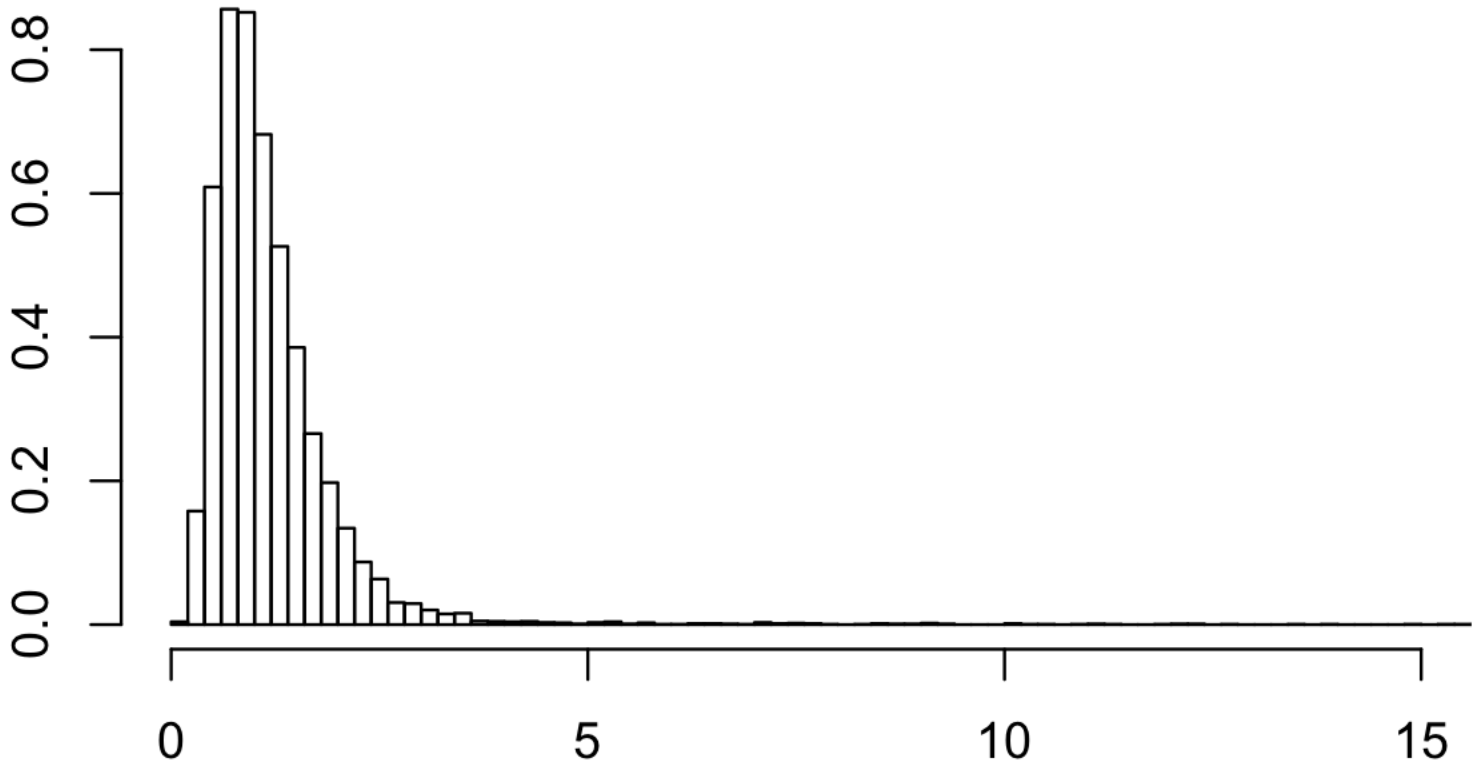
What about visualization?


Can’t see small count bars
0
200
400
600
200
400
600
800
1000
1200
$\Delta t(ms)$

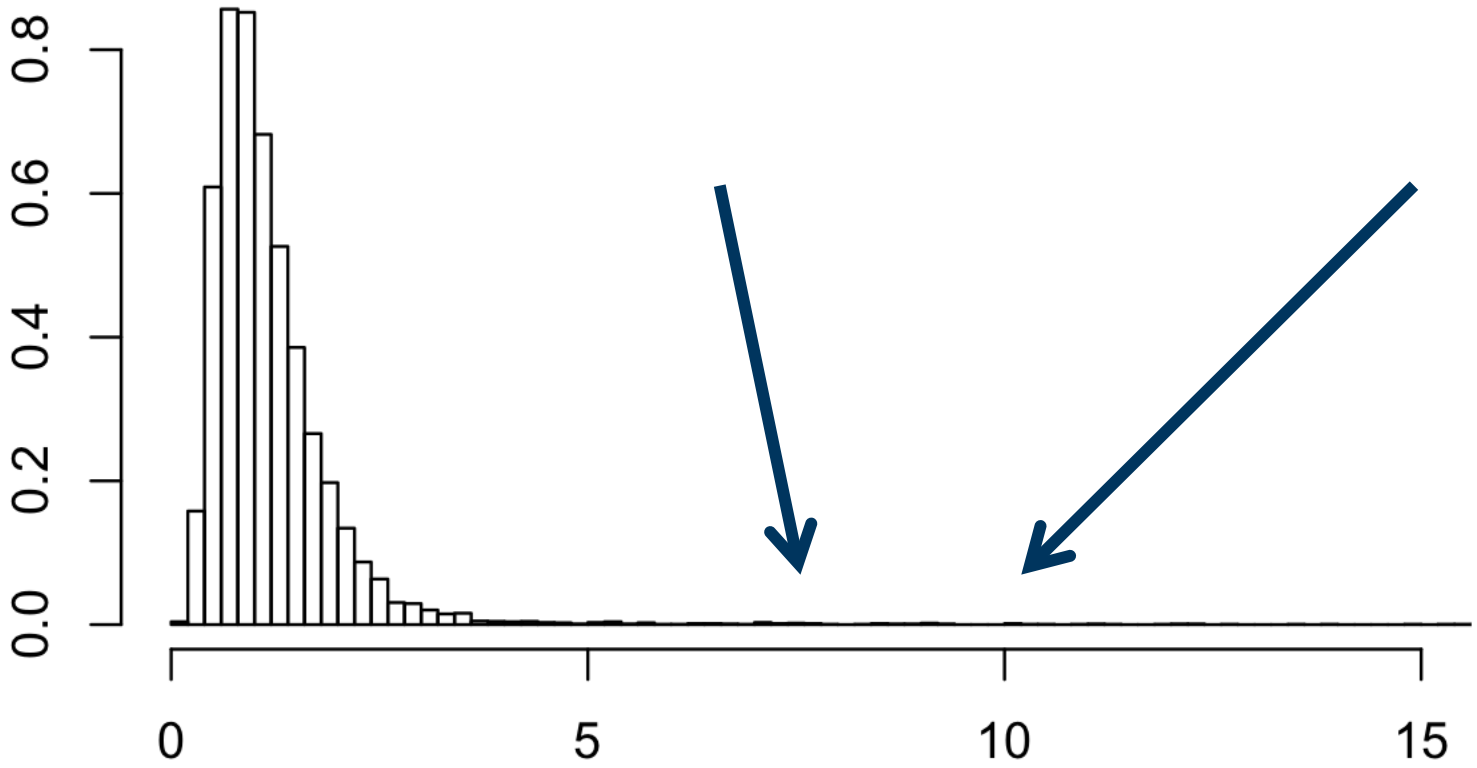
Good Results



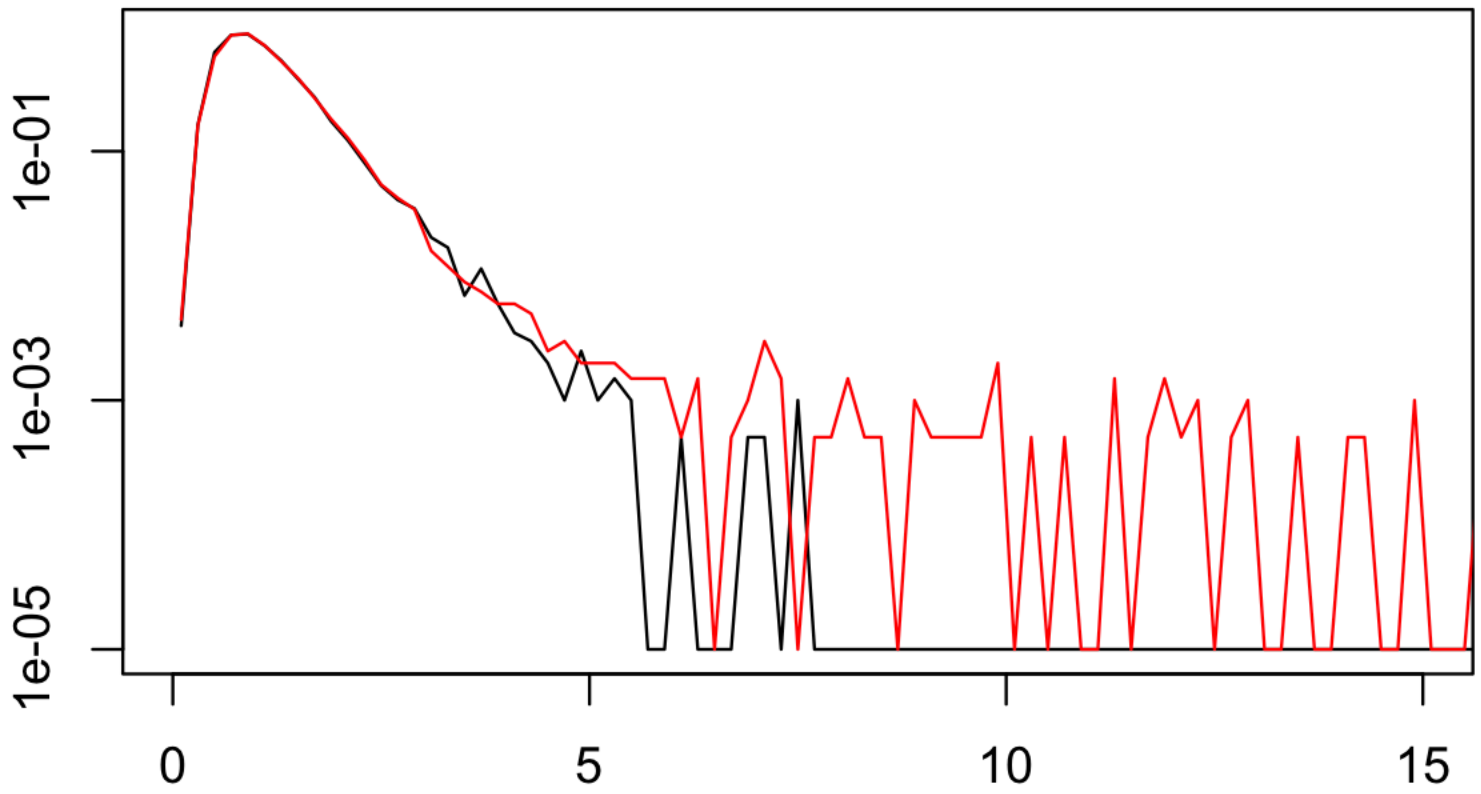
Bad Results – 1% of measurements are 3x bigger



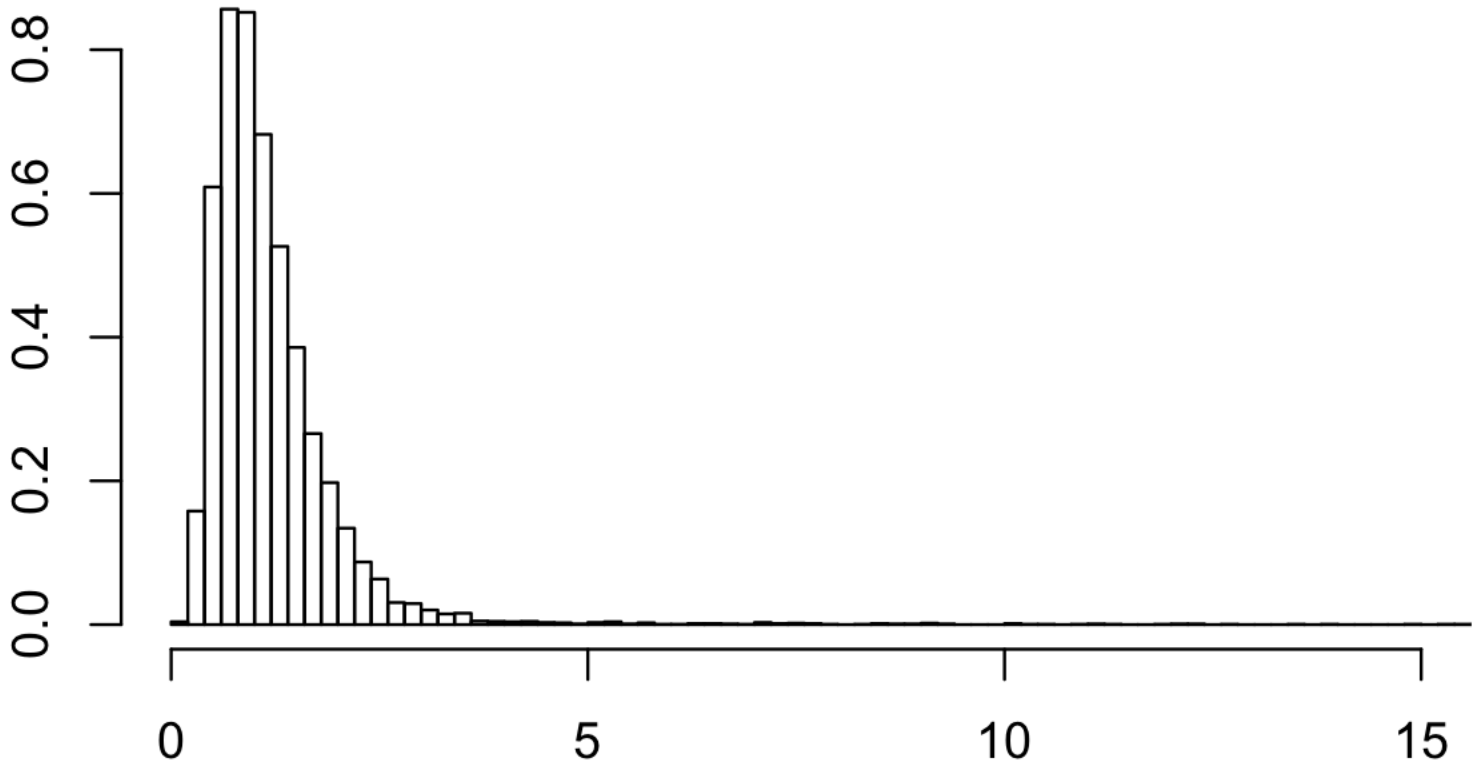
Bad Results – 1% of measurements are 3x bigger



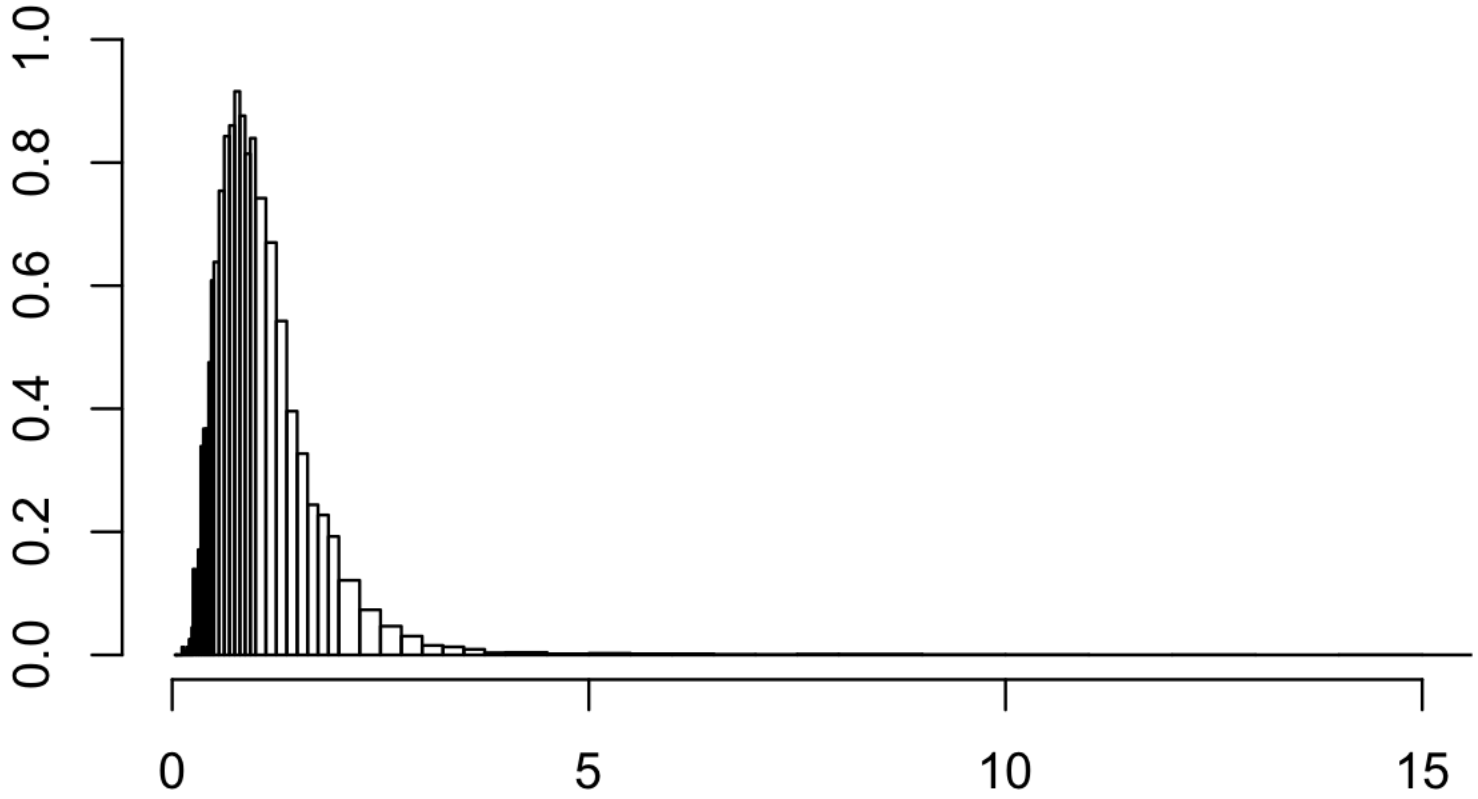
With Better Vertical Scaling



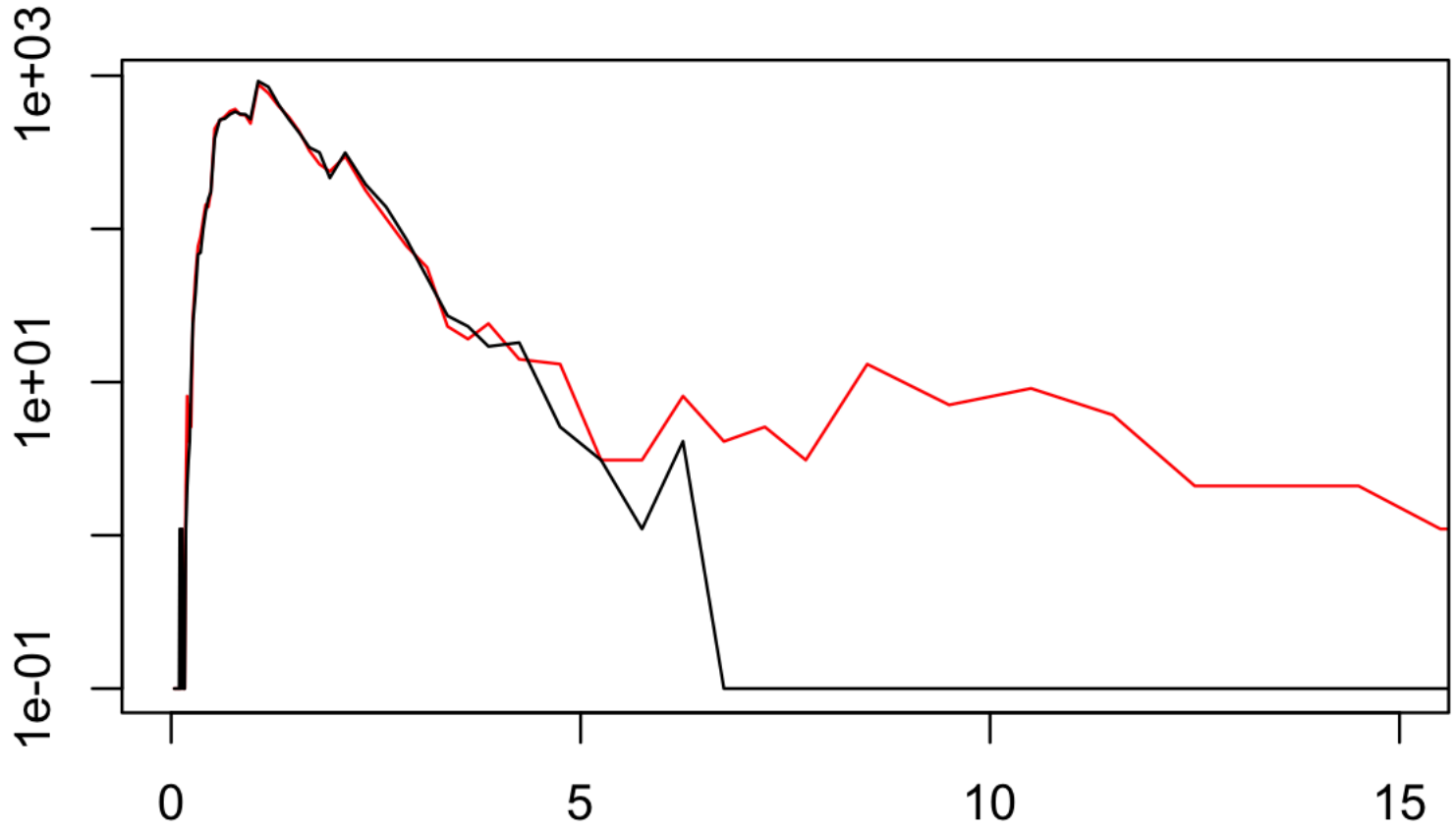
Uniform Bins



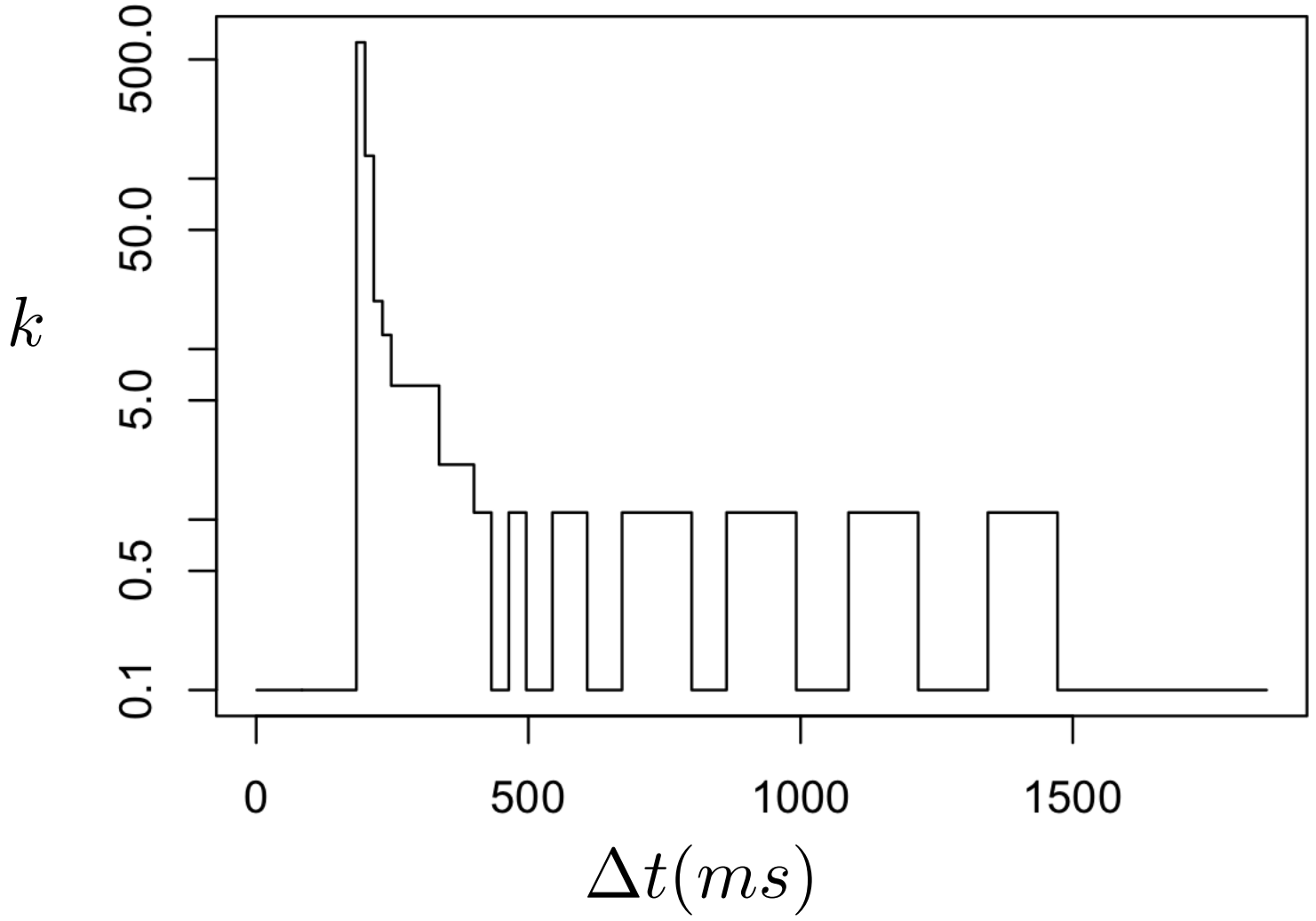
FloatHistogram Bins



With FloatHistogram



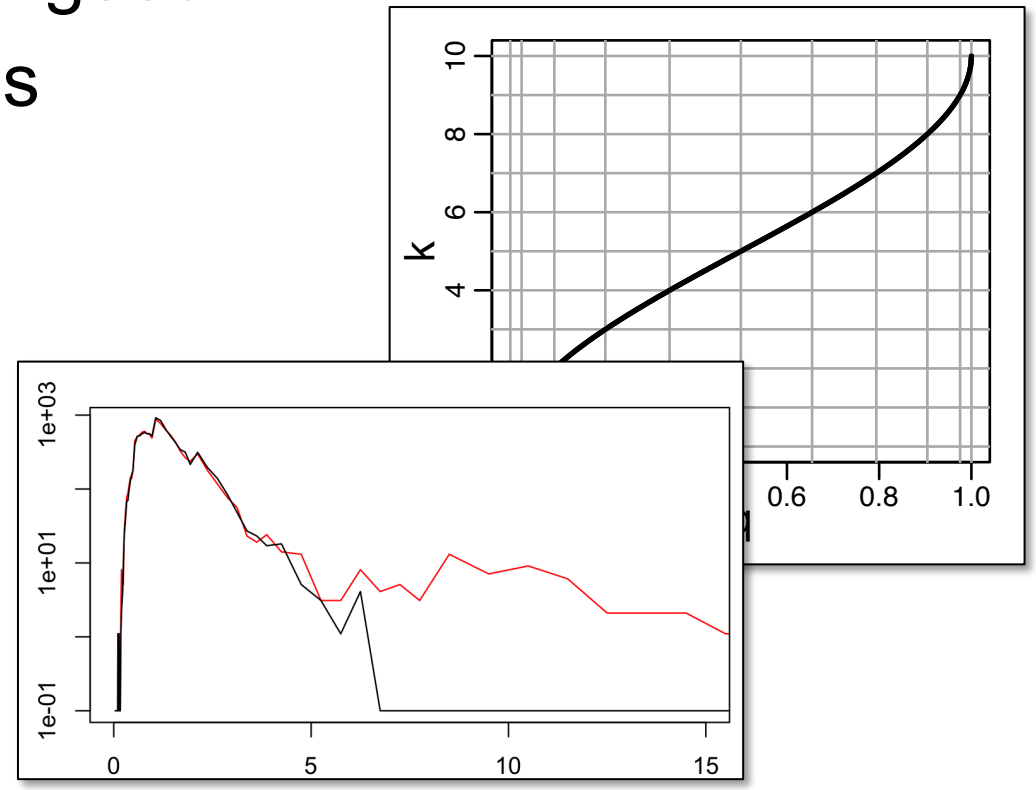
Original Ping Latency Data



Summary

- Single measurements insufficient, need distributions
- Uniform binned histograms not good
- FloatHistogram for some cases
- T-digest for general cases

- Upcoming release has super-fast and accurate versions
- Good visualization also key



Q & A

Contact Information

Ted Dunning, PhD

Chief Application Architect, MapR Technologies

Board member, Apache Software Foundation

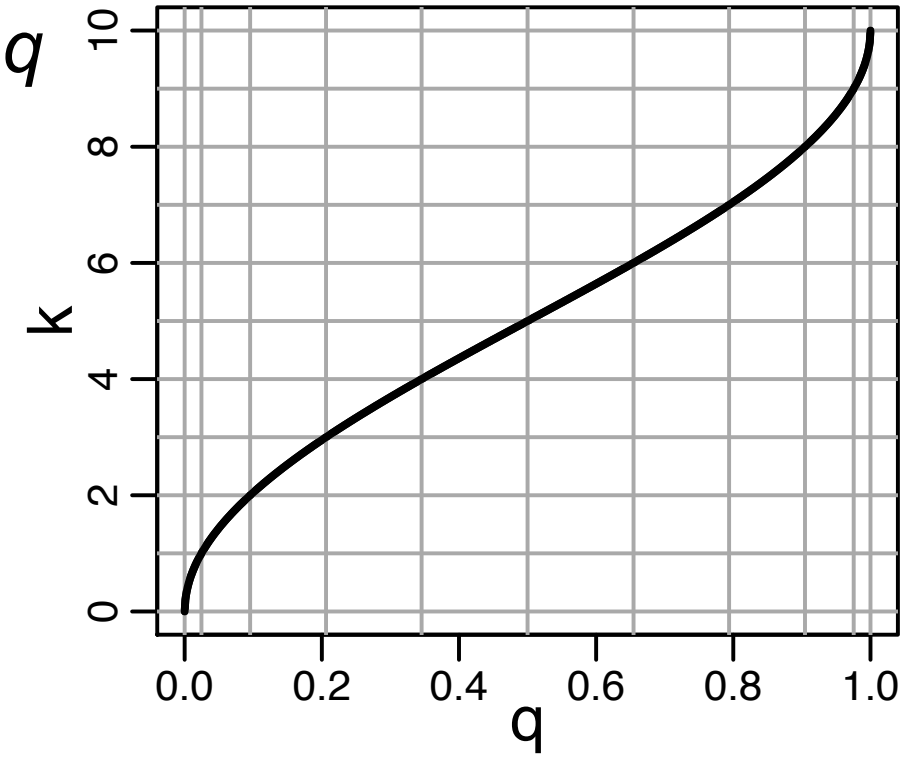
O'Reilly author

Email tdunning@mapr.com tdunning@apache.org

Twitter @ted_dunning

T-digest

- Or we can talk about small errors in *q*
- Accumulate samples, sort, merge
- Merge if *k*-size < 1
- Interpolate using centroids in *x*



- Very good near extremes, no dynamic allocation