How Amazon uses better metrics for improved website performance

Jim Roskind (he/him)  
VP, Distinguished Engineer  
Amazon.com

Frank Stone (he/him)  
Principal Solutions Architect  
Amazon Web Services
Agenda

Good vs. bad latency goals

Percentile latency goals: problematic!

Trimmed Mean: a better metric

Histograms

How to measure your latency in 4 easy steps
Why is Jim Roskind talking about metrics?

Cofounded Infoseek

1994

Implemented Python Profiler (used for 20 years)

Visualization is key

Worked at Google on making Chrome go faster

2008

Designed/implemented metrics infrastructure

Proposed, architected, lead development of QUIC protocol (basis of HTTP/3)

Critical: experiments/metrics

Amazon VP/ Distinguished Engineer

2016

Helped to drive latency down via “better metrics”
Context: faster is better

Customers want responses ASAP

End user latency

…Surprise question:
What is a “good goal” to lead an organization?
What should good latency mean?

Indistinguishable from instantaneous

...speed-of-light limits...
...and variance, since computers and networks get busy

200–500ms would be excellent

Changes of +/- 10-40ms can have large $ results
Switching to a “good metric” saved 30–40+% in 2020
Using a “good metric” identified 500ms regressions in H1 2021

Customers should “expect” good latencies
Define: “expect” good latencies

Expectation == “average value”

Amazon needs great averages

Amazon needs to avoid long (slow) tails in distributions

Why not use average?

“Average” is dominated by any looong tail!

Average 10ms with 10 minutes → bad result
dominated by 10 minutes!

Answer: end user variance contains too much noise!
What are bad goals?
What are “understats”?

EXAMPLE

“99% of Amazon’s pages served to 99% of Amazon’s customers in under 1 second.”

“Understats” pick a threshold (1 second?), measure %

Advantage: Just one metric to watch!

Understats are “fencepost goals”

Organizations don’t do well with fenceposts!?!
"Fenceposts" define only good vs. bad

**Good side**
- Under 1 second
- Stay on this side

**Bad side**
- Over 1 second
- Any time is equal
India has terrible page-load latency

India has ~400ms round trip time (RTT)
   Connections cost 3+ round trips
   Server gets request ~1.2 seconds after user clicks

Does the “1-second understat” goal help India?

Engineers shouldn’t work on this goal!!!
   It is impossible to improve on the “1-second understat goal”
   Why bother trying?
Fencepost goals are problematic

If you can’t better a fencepost, it isn’t worth trying

If you better a fencepost, there is no reason to do more

In fact...

You can often get worse, if you have a reason to do so

“Can I land my feature? It won’t change the [fencepost!] latency stats!”
Why were P50 and P90 used as Amazon goals?

They assume (?) all distributions are normal

- P50 helps us drive down the “mode”
  (median? average?)
- P90 helps us drive down the tail

😊 That’s what we thought !?!
What gets missed by P50 and P90 goals

- Fast/good samples
- Slow/bad samples
Why are P50 and P90 “bad latency goals?”

Both percentile-markers are fenceposts

Organizations don’t do well with fenceposts
“Show me the data”
“Show me the data!”
"When a measure becomes a target, it ceases to be a good measure."

Goodhart's law
Generalization by Marilyn Strathern
What are good vs. bad goals?

“Understats” are bad goals
- They are “fencepost goals” = bad

P50 and P90 are bad goals for latency
- They actually HARM latency
- They encourage latency to get worse!?!?

TM99 (trimmed mean at 99%) is a good goal for latency
- Single metric, that drives an organization

Histograms help us reach and surpass a goal
What is Trimmed Mean?
Define “Trimmed Mean”

TM99 means: discard slowest 1%, then average

TM 99.9 means: discard slowest 0.1%, then average

[Subtle details...for statisticians]

Technical definition: “trim at both ends”… but...
Don’t bother to trim at low end, it won’t matter

“Winsorizing”: Don’t just discard over 99%
…pretend samples above 99% were exactly P99
…and then average
“Trimmed mean” is NOT “discard above N”

Trimmed mean is NOT: “discard above 10 seconds”

Such a “discard policy” is VERY bad
Trimmed Mean “is” the expected value

Trimmed Mean discards almost nothing

It only discards “noise” in the tail

Unlike P90, TM99.x watches the worst 10%
TM99.x: constant organizational pressure

A single-goal organizational metric: easy to watch

Credit (and blame) for (almost) all latency changes

Customer obsessive as it watches (almost) everything

If TM99 is smooth over time, watch TM99.9

If TM99.9 is smooth, watch TM 99.99
Do goals really drive organizations?

One of our leaders asked:

“If we’ve only been using P50/P90 for a few years, how long will it take for TM99 goals to change things?”
Example: January vs. October 2020
Example: August 2019 through October 2020

August 2019
How do histograms help us?
What can be seen in a histogram

Chrome TCP Connect Latency on Windows (9 Billion samples)
It is not a bell curve: P50/90 don’t “tell all”
Odd modes become visible

Chrome TCP Connect Latency on Windows (9 Billion samples)

Cached connections 20%

P50=53ms

P90=306ms
Each surprise is an opportunity

Chrome TCP Connect Latency on Windows (9 Billion samples)

- P50 = 53ms
- P90 = 306ms
- SYN packet lost
- 3-second till retry
- 1-2% Total
Histograms show ALL the data

Every surprise is an opportunity

Better than watching P99

Outliers visible as “surprising modes”

Use log-scale for time axis
  Your entire dynamic range will be visible
Summary

- Trimmed Mean: a better goal
- Histograms: a better visualization
“That’s great Jim, but how do I apply this to my applications in AWS?”
Better metrics in four easy steps with CloudWatch

1. Collect your latency metrics
2. Measure with Trimmed Mean
3. Monitor progress: dashboards and alarms
4. Review histograms and optimize
Collect your latency metrics in CloudWatch

CloudWatch Real User Monitoring (RUM)
Collect client metrics

CloudWatch PutMetricData
Gather metrics from any source

CloudWatch Embedded Metric Format (EMF)
Capture metrics from logs
CloudWatch real-user monitoring (RUM)

Collect client-side metrics and data for your applications

- Register your application
- Add a provided snippet of code
- Capture the data
The CloudWatch PutMetricData API

Publish your own metrics to CloudWatch

Use the AWS Command Line Interface (CLI) or API

Publish individual or groups of values for your metrics

Include multiple metric dimensions

```
$ aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://metric.json
```
CloudWatch Embedded Metric Format (EMF)

Extract metrics from your log data directly into CloudWatch

Embed metrics with a standard JSON format in your log data

Preserve the detailed event context

Use Lambda functions to easily generate custom metrics

Capture embedded metrics and logs with CloudWatch Agent

```
{
  "_aws": {
  "Timestamp": 1574109732004,
  "CloudWatchMetrics": [
  {
    "Namespace": "lambda-function-metrics",
    "Dimensions": ["functionVersion"],
    "Metrics": [
      {
        "Name": "time",
        "Unit": "Milliseconds"
      }
    ]
  }
  ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```
Set and measure goals

CloudWatch metrics support Trimmed Mean as a statistic.

Use a period that includes many samples.

Use TM(XX%:XX%) to fine tune Trimmed Mean.
Trimmed mean: tips

1. Monitor your outliers:
   TM(99%:100%)

2. Remove anomalies at both ends:
   TM(1%:99%)

3. Use Trimmed Count to monitor sample size:
   TC(1%:99%)

4. Winsorized Mean is also an option:
   WM(1%:99%)
Monitor progress: dashboards and alarms

Use CloudWatch to monitor and track progress

Alarms

Dashboards

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Review histograms and optimize

Histograms in CloudWatch?

Yes, but there are a few steps:

- Determine the metric range and the increments you want to display
- Use multiple PR(XX:XX) labels to define each range/bar
- Select “Bar” for the visualization
Recap

Avoid “Fencepost” goals and leverage Trimmed Mean for latency metrics

Use histograms to find patterns in your metrics

CloudWatch supports Trimmed Mean

Leverage CloudWatch alarms, dashboards, and histograms
Resources:

CloudWatch Trimmed Mean

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Statistics-definitions.html

CloudWatch real user monitoring (RUM)


CloudWatch PutMetricsData API

https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_PutMetricData.html

CloudWatch embedded metrics format

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Embedded_Metric_Format.html

Thank you!

Please complete the session survey in the mobile app.