Evolving Prometheus for the Cloud Native World

Brian Brazil
Founder

Robust **Perception**

Who am I?

Engineer passionate about running software reliably in production.

- Core developer of Prometheus
- Studied Computer Science in Trinity College Dublin.
- Google SRE for 7 years, working on high-scale reliable systems.
- Contributor to many open source projects, including Ansible, Python, Aurora and Zookeeper.
- Founder of Robust Perception, provider of commercial support and consulting for Prometheus.

Robust **Perception**

What am I going to talk about?

How did we get to where we are?

What is Prometheus?

How has Prometheus changed with Cloud Native?

Historical Monitoring

A lot of what we do today for monitoring is based on tools and techniques that were awesome decades ago.

Machines and services were cared for by artisan sysadmins, with loving individual attention.

Special cases were the norm.

The Old World

Tools like Nagios came from a world where machines are pets, and services tend to live on one machine.

They come from a world where even slight deviance would be immediately jumped upon by heroic engineers in a NOC. Systems were fed with human blood.

We need a new perspective in a cloud native environment.

What is Different Now?

It's no longer one service on one machine that will live there for years.

Services are dynamically assigned to machines, and can be moved around on an hourly basis.

Microservices rather than monoliths mean more services created more often.

More dynamic, more churn, more to monitor.

What is Prometheus

Prometheus is a metrics-based monitoring system.

It tracks overall statistics over time, not individual events.

It has a Time Series DataBase (TSDB) at its core.

Powerful Data Model and Query Language

All metrics have arbitrary multi-dimensional labels.

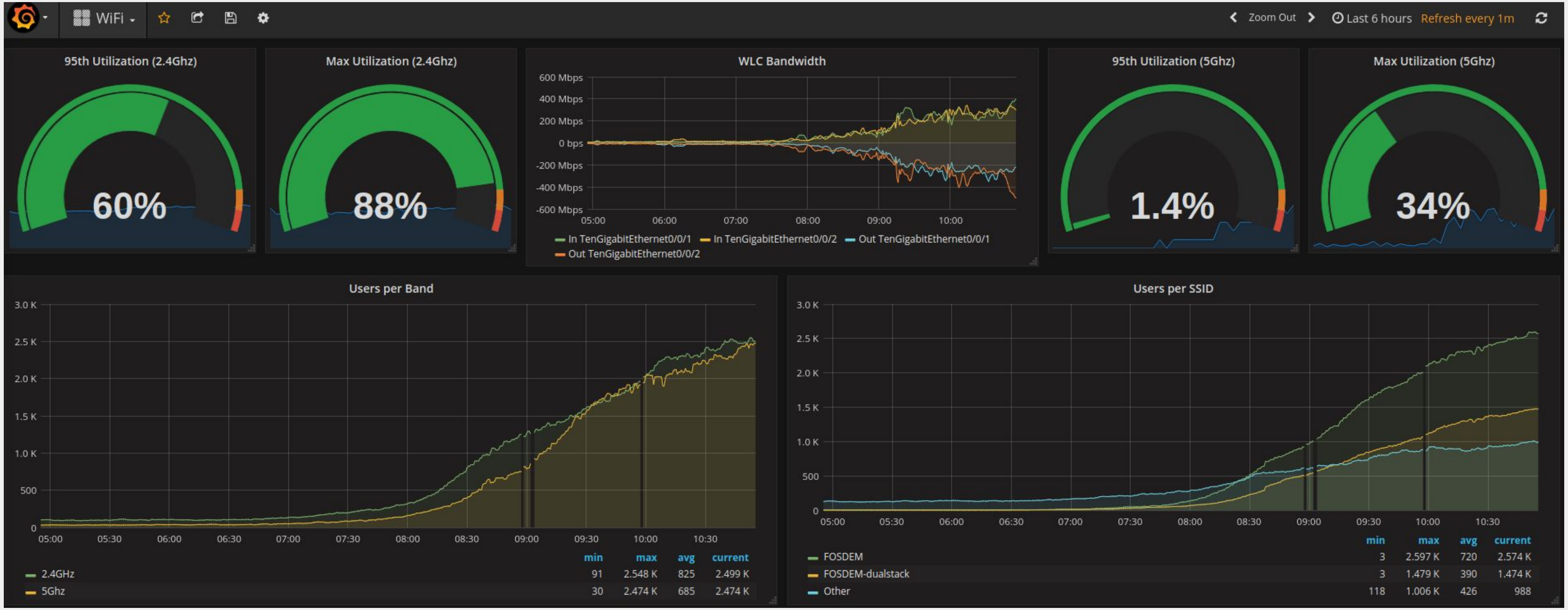
Supports any double value with millisecond resolution timestamps.

Can multiply, add, aggregate, join, predict, take quantiles across many metrics in the same query. Can evaluate right now, and graph back in time.

Can alert on any query.

Robust **Perception**

FOSDEM Wifi Metrics from Prometheus



	min	max	avg	current
2.4GHz	91	2.548 K	825	2.499 K
5Ghz	30	2.474 K	685	2.474 K

	min	max	avg	current
FOSDEM	3	2.597 K	720	2.574 K
FOSDEM-dualstack	3	1.479 K	390	1.474 K
Other	118	1.006 K	426	988

Robust Perception

Reliability is Key

Core Prometheus server is a single binary.

Each Prometheus server is independent, it only relies on local SSD.

No clustering or attempts to backfill "missing" data when scrapes fail. Such approaches are difficult/impossible to get right, and often cause the type of outages you're trying to prevent.

Option for remote storage for long term storage.

Prometheus and the Cloud

Dynamic environments mean that new application instances continuously appear and disappear.

Service Discovery can automatically detect these changes, and monitor all the current instances.

Even better as Prometheus is pull-based, we can tell the difference between an instance being down and an instance being turned off on purpose!

Heterogeneity

Not all Cloud VMs are equal.

Noisy neighbours mean different application instance have different performance.

Alerting on individual instance latency would be spammy.

But PromQL can aggregate latency across instances, allowing you to alert on overall end-user visible latency rather than outliers.

Symptoms rather than Causes

With far more complex environments with many moving parts, alerting on everything that might cause a problem is not tractable.

Even trying to enumerate everything that could go wrong is nigh on impossible.

Ultimately you care about user experience metrics, such as RED.

An alert on a symptom of high latency at your frontends will cover vast swathes of potential failure modes. From there, use dashboards to drill down through your architecture.

How has Prometheus Changed?

Prometheus started out with a basic TSDB, little in the way of service discovery and a much more primitive PromQL than we have today.

Over time all of these have evolved.

Prometheus 2.0 brings improvements in two areas.

A new TSDB is far more efficient.

New staleness handling better supports instances disappearing.

v1: The Beginning

For the first 2 years of its life, Prometheus had a basic implementation.

All time series data and label metadata was stored in LevelDB.

If Prometheus was shutdown, data was lost.

Ingestion topped out around 50k samples/s.

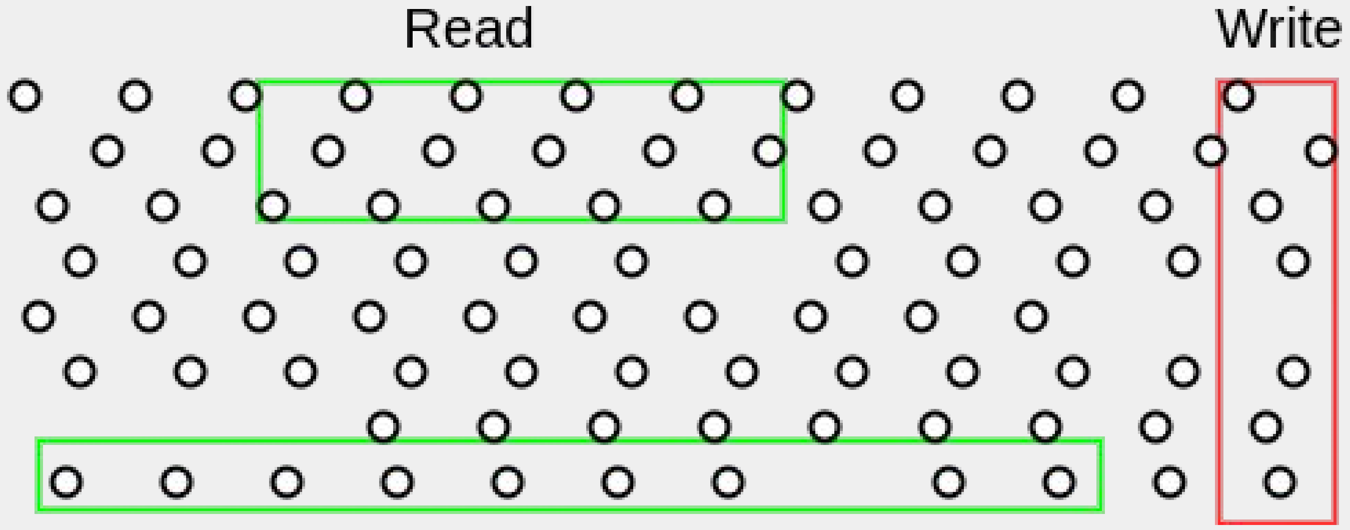
Enough for 500 machines with 10s scrape interval and 1k metrics each.

Robust **Perception**

Why Metrics TSDBs are Hard

Writes are vertical, reads are horizontal.

Write buffering is essential to getting good performance.



Robust **Perception**

v2: Improvements

v2 was written by Beorn, and addressed some of the shortcomings of v1.

It was released in Prometheus 0.9.0 in January 2015.

Time series data moved to a file per time series. Writes spread out over ~6 hours.

Double-delta compression, 3.3B/sample.

Regular checkpoints of in-memory state.

v2: Additional Improvements

Over time, various other aspects were improved:

Basic heuristics were added to pick the most useful index.

Compression based on Facebook Gorilla, 1.3B/sample.

Memory optimisations cut down on resource usage.

Easier to configure memory usage.

v2: Outcome

Much more performant, the record is ingestion of 800k sample/s.

Not perfect though. That big a Prometheus takes 40-50m to checkpoint.

Doesn't deal well with churn, such as in highly dynamic environments. Limit of on the order of 10M time series across the retention period.

Write amplification is an issue due to GC of time series files.

LevelDB has corruption and crash issues now and then.

Where to go?

We need something that:

- Can deal with high churn
- Is more efficient at label indexing
- Avoids write amplification

Supporting backups would be nice too.

Robust **Perception**

v3: The New Kid on the Block

Prometheus 2.0 has a new TSDB, written by Fabian.

Data is split into blocks, each of which is 2 hours. Blocks are built up in memory, and then written out. Compacted later on into larger blocks.

Each block has an inverted index implemented using posting lists.

Data accessed via mmap.

A Write Ahead Log (WAL) handles crashes and restarts.

v3: Outcome

It is early days yet, but millions of samples ingested per second is certainly possible.

Read performance is also improved due to the inverted indexes.

Memory and CPU usage is already down ~3X, due to heavy micro-optimisation.

Disk writes down by ~100X.

Prometheus for the Cloud Native World

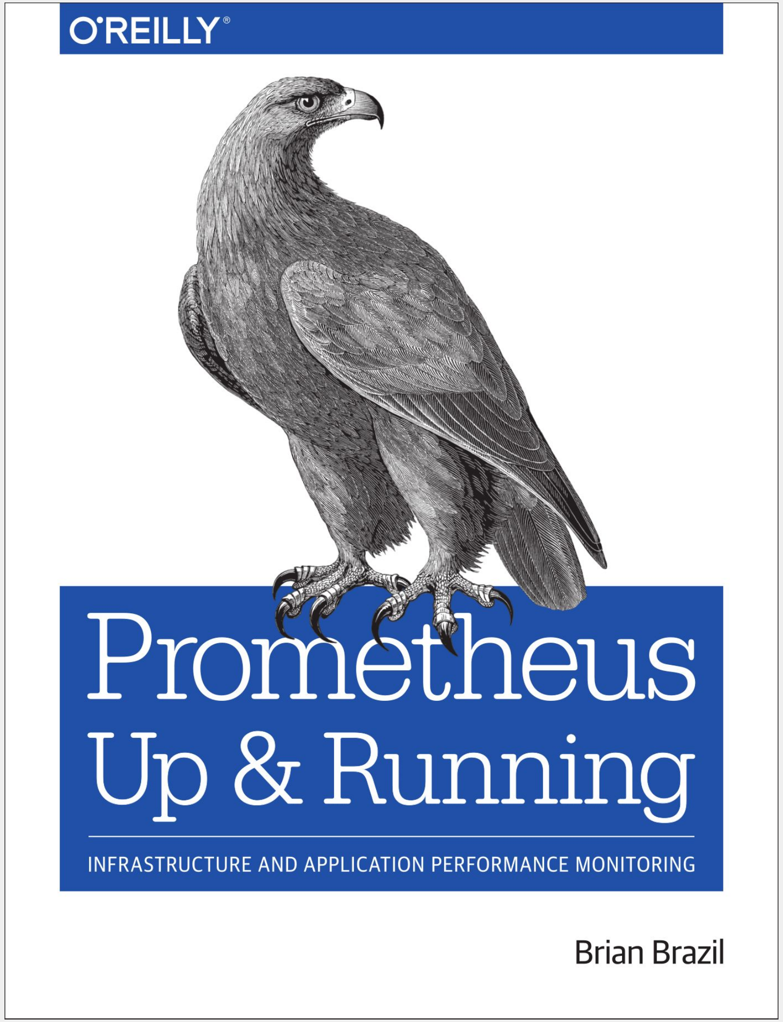
Prometheus 2.0 is based on years of monitoring experience.

The new TSDB greatly expands Prometheus's ability to deal with the dynamic and high churn cloud environments that are common today.

Service discovery knows what to monitor, and PromQL allows alerting on overall symptoms rather than individual causes.

Prometheus is a good choice for Cloud Native metrics monitoring, and has a community of thousands of companies!

Robust **Perception**


O'REILLY
Prometheus Up & Running
INFRASTRUCTURE AND APPLICATION PERFORMANCE MONITORING
Brian Brazil

Prometheus: The Book

Coming in 2018!

Robust Perception

Questions?

Prometheus: prometheus.io

Live Demo: demo.robustperception.io

Company Website: www.robustperception.io

Robust **Perception**