Prometheus Deep Dive

Monitoring. At scale.

Richard Hartmann & Frederic Branczyk
@TwitchiH & @fredbrancz

2018-12-12

Who are we?

- Richard ”RichiH” Hartmann
  - Swiss army chainsaw at SpaceNet
  - Project lead for building one of the most modern datacenters in Europe
  - Debian Developer
  - FOSDEM, DebConf, DENOGx, PromCon staff
  - Prometheus team member
- Frederic Branczyk
  - Red Hat (previously CoreOS)
  - All things Prometheus / Kubernetes
  - Kubernetes SIG-Instrumentation lead
  - Prometheus team member

Show of hands

- Who has heard of Prometheus?
- Who is considering to use Prometheus?
- Who is POCing Prometheus?
- Who uses Prometheus in production?

Prometheus 101

- Inspired by Google's Borgmon
- Time series database
- int64 timestamp, float64 value
- Ecosystem of instrumentation & exporters
- Not for events
  - Logging
  - Tracing (more on that later)
  - etc.
- Dashboarding via Grafana

Main selling points

- Highly dynamic, built-in service discovery
- No hierarchical model, n-dimensional label set
- PromQL, one language for ALL the things: processing, graphing, alerting, export...
- Simple operation
- Highly efficient

Cloudy with a chance of buzzwords

- So it's built with highly dynamic environments in mind
- It's the second project to ever join CNCF and the de facto standard in cloud-native monitoring
- Kubelets, sidecars, microservices, ALL the cloud-native
- But it's a monolithic application
- ...why?

Resilience, resilience, and also resilience

- What do you need for operations?
  - Power and cooling
  - Network connectivity
  - Observability, a.k.a. Monitoring
- The rest you can fix

Three main features

- Storage backend
  - Caveat: Prometheus 2.0 comes with storage v3
- Staleness handling
- Remote read & write API is now stable-ish
- Links to in-depth talks about these features are at the end

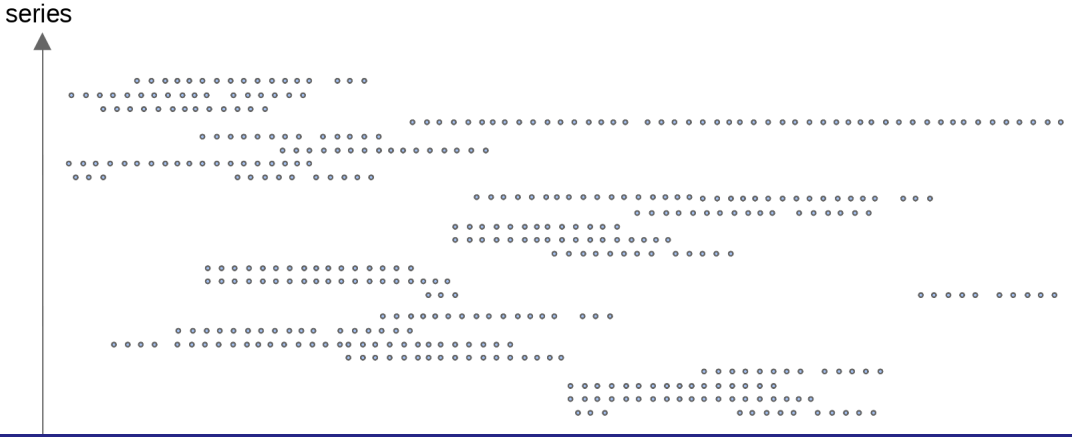
Prometheus 1.x

- We used to have one file per time series
- ..and one common index for all of time
- Relatively easy to implement
- Pretty efficient
- Why change?

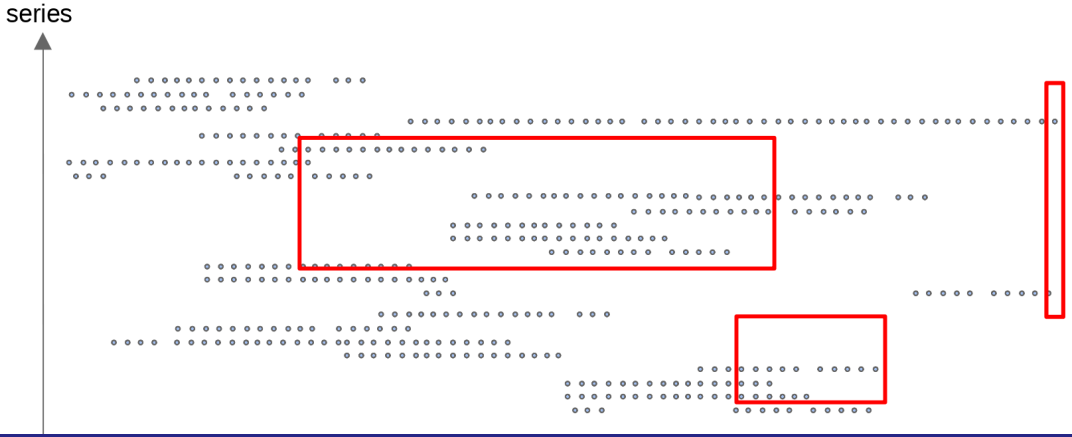
Churn

- Churn was becoming more and more of a problem
- There's a company with a 15 minute maximum lifetime for their containers
- If you have a lot of files which might contain data for any given time frame, you need to look at all of them

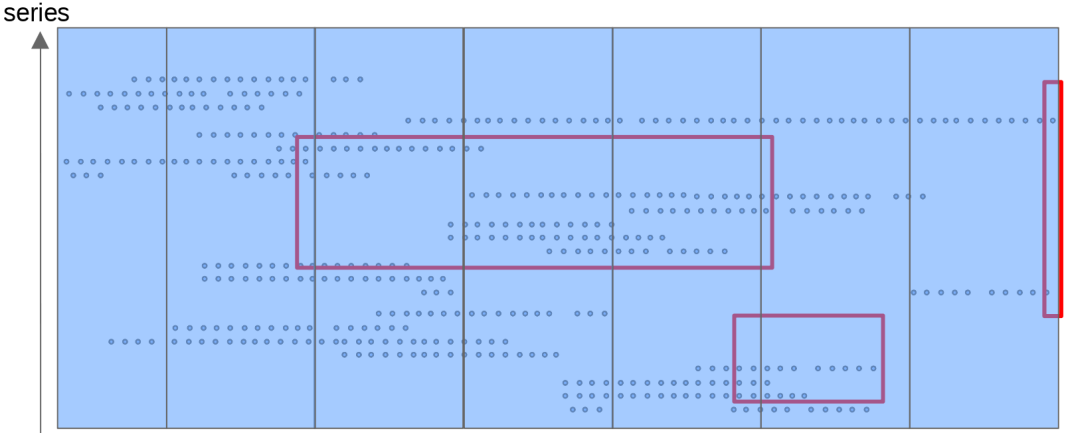
One file per series



Selection



Blocks



Test setup for 2.0 release

- Kubernetes cluster with dedicated Prometheus nodes
- 800 microservice instances and Kubernetes components
- 120k samples/sec
- 300k active time series
- Swap out 50% of all pods every 10 minutes

Results

- 15x reduction in memory usage
- 6x reduction in CPU usage
- 80-100x reduction in disk writes
- 5x reduction in on-disk size
- 4x reduction in query latency on expensive queries
- Want to reproduce? `https://github.com/prometheus/prombench`

Playing nicely with others

- We now have a stable-ish remote read/write API
- Twelve integrations for this API
- Ongoing work to send write-ahead-log over the wire to fill gaps

Security & quality

- CNCF sponsored external code review by Cure53
- Focussed on security, but this always means looking at stability as well
- Keep in mind that Prometheus willfully ignored most security considerations
- Encryption, authentication, and authorization currently need to be handled via reverse proxies
- We will be changing Prometheus to support security out-of-the-box

Release stability

- Every single release since 2.0.0 has had issues
- Some bugs and some human mistakes in the release process
- Always running latest is the cloud-native approach, but this is still not acceptable
- ..especially if every single version has its issues
- We put in more checks and balances to ensure cleaner releases going forward
- If there are bugs we can't get rid of, we go into feature moratorium
- 2.3.2 is the first fully stable release in the 2.x train

Fixed release cycle

- Every six weeks, we mark a new RC
- Cycle is relative to previous RC, not previous release
- RC is published and iterated upon as long as there are issues
- Release handling cycles between team members to ensure several people are able to release

Quick is not quick enough

- Brian Brazil optimized PromQL
- 5x faster for time vector functions
- 100x reduction in garbage to collect

Problem statement

- Long-term storage is one of the last remaining major features left untackled
- Fundamentally, Prometheus operates as distinct data islands
- As there's no backfill, data dies along with its instance by default

Solutions

- Storage v3 supports backups efficiently and effectively
- Remote read-write allows you to integrate with a growing list of projects and products, e.g. Cortex
- On storage level, there are object storage backends for Prometheus, e.g. Thanos
- Remote API can now send WAL over the wire to fill gaps in data
- There are twelve different systems which are able to ingest Proemtheus data this way
- We deliberately do not endorse any particular approach or solution; this might change over time

Testing

- Unit tests for alerts
- Our goal is to allow end-to-end testing of not only Prometheus as software, but also of any individual deployment

ACID databases...

- **A**tomicity - since 1.x
- **C**onsistency - since 1.x
- **I**solation - will happen within 2.x
- **D**urability - since 2.0

Isolation

- Each append action gets a write ID (64 bit monotonic counter)
- Every sample's write ID is noted along with value and timestamp
- Any append action which has not yet been committed, or has been rolled back, is ignored at query time
- We keep write IDs in memory; if we restart or crash, the atomicity of the write ahead log will protect us

Humble aspirations

- When we say that we want to change how the world does monitoring, we mean it
- One of our most powerful features are labels
- Labels are encoded in our exposition format
- Some third-party projects and vendors have an issue with supporting a ”competing” project

What do?

- We are spinning out Prometheus' exposition format
- Face-to-face kick-off last August at Google London
- Independent CNCF member project, IETF RFC, test suite, etc
- We are writing code in Prometheus and the Python client library
- `https://github.com/OpenObservability/OpenMetrics`
- Prometheus 2.5 has experimental OpenMetrics support

Beyond metrics

- OpenMetrics supports more than just metrics
- Every single data point in a time series can point to one single event
- Especially useful if you emit one trace id per histogram bucket
- Some integrations already support this concept, e.g. OpenCensus
- Ingestors are free to discard this optional data, e.g. Prometheus

Bringing observability together

- OpenTracing already on board with this effort
- There will be an observability sidetrack
- Long-term goal is one common, modular, well-engineered standard under a new name

First committers to adopt, too many to list all

- Cloudflare
- CNCF at large
- GitLab
- Google
- Grafana
- InfluxData
- Kausal.co
- Oath.com / Yahoo / Verizon
- RobustPerception
- SpaceNet
- Uber

Generally speaking...

- Yes, we want to change the world
- Simple and resilient operation of Prometheus remains a core goal
- The path from raw data to reliable alerts is the single most important user contract we have
- More project and software integrations... and we're talking to hardware vendors as well
- Supporting tomorrow's 10x scale today

Relevant talks

- Storing 16 Bytes at Scale:
  `https://promcon.io/2017-munich/talks/staleness-in-prometheus-2-0/`
- Staleness and Isolation in Prometheus 2.0:
  `https://promcon.io/2017-munich/talks/staleness-in-prometheus-2-0/`
- Social aspects of change:
  `https://promcon.io/2017-munich/talks/social-aspects-of-change/`

Further reading

- Prometheus 2017 Dev Summit: `https://docs.google.com/document/d/1DaHFao0saZ3MDt9yuuxLaCQg8WGadO8s44i3cxSARcM/edit`
- Prometheus 2018 Dev Summit: `https://docs.google.com/document/d/1-C5PycocOZEVIPrmM1hn8fBelShqtqiAmFptoG4yK70/edit`
- OpenMetrics: `https://github.com/RichiH/OpenMetrics`
- This and other talks: `https://github.com/RichiH/talks/`

Thanks!

Thanks for listening!

Questions?