Prometheus 2020 Dev Summit 2

2020-08-07, 14:00 - 18:00 UTC, Online

Last Meetings

- 2017
- 2018
- 2019.05
- 2019.11
- 2020.07

Attendees

- Richard “RichiH” Hartmann
- Ben Kochie, partial
- Chris Marchbanks, partial
- Brian Brazil
- Julius Volz
- Julien Pivotto
- Bartek
- Matthias Rampke
- Björn Rabenstein
- Frederic Branczyk
- Tom Wilkie, partial
- Callum, partial
- Goutham. partial

Editorial note: We captured less of the actual discussions in this document than we used to. Incentivizing people to write notes online is harder than if we’re all in the same room.

Meeting Notes

Frederic: Transaction based remote write

- All samples/series from a given scrape would be contained in a single remote write request
- Brian: 2 major issues
    - Some RW endpoints cannot deal with large payload sizes
        - Tom: Not our problem, though?

        - Brian: Yeah, but general performance issue if there's shard imbalance
        - Matthias: Option to limit the request size, you no longer get a complete transaction but would have some control
    - Sharding
        - Chris: If sharding becomes an issue, we can improve the algorithm
        - Tom: As it stands upsharding should handle issues
        - Matthias: Start without fragmentation?
        - Julius: What about max_samples_per_send
            - Remove it in same release vs deprecate it for a couple releases then remove it

<Callum, Goutham join 14:32>
<Tom leaves 14:32>

- CONSENSUS:
    - We want transactional remote write that, so that a given commit to the TSDB is not split across the requests that remote write sends.

RichiH: Extending PromQL

- RichiH: From bug scrub: PromQL: add let/where expression · Issue #6146 · prometheus/prometheus
    - Users want more features; but we need to be careful not to just add everything
    - Opinionated language based on collective decades of operational experience means we can and should uphold best practices.
    - Deeper statistical analysis
- Output modifiers
    - Matthias: I can see use cases for modifying output, sorting
    - RichiH: Same, I want this. Can see other output modifiers, but mainly care about label sorting
    - RichiH: Want to be able to have a standardized interface to get sorted stuff out of Prometheus proper, not rely on UI
    - Julien: Have issues with sorting in Elastic, didn't work in Grafana, would like to have fallback in the datasource
    - Brian: Would this boil down to embedding Lua for advanced use cases?
    - RichiH: Let's write down use cases we want to support and then deduce language requirements; should be similar to PromQL proper.
    - Brian: This is Pandora's box, consensus poorly defined
    - MR: Feel limited when using the Prometheus UI directly. Grafana Explore is almost there. Want to limit output to, say 200, but 200 what? Do not think we need a generic computation engine, just specific commands. E.g. "show me the outliers" as my browser crashes with 10k items
    - RichiH: We can support this in Grafana and we should. But I want Prometheus to be used. Not just the ugly thing with a different long-term storage, a different UI, a

different anything. I want people to use Prometheus proper. We need to enable people to get their job done.
    - <General agreement>
    - Brian: What are we talking about specifically?
    - MR: Sorting
        - RichiH: And pagination
    - Brian: Offsets are n^2, would prefer to send everything and let the other end handle it
    - MR/RichiH: What if the other end is a browser with a human?

<15:15 Ben Kochie & Chris Marchbanks left>

- CONSENSUS:
    - In the query APIs as an EXPERIMENTAL post-PromQL step
        - We want to support nested sorting ascending/descending of time series based on
            - label values
                - UTF-8 Default string sort
                - Numeric sorting based on trying to parse the whole label value as a float, so le/quantile works
            - sample values for the /query endpoint
        - We will explore how to do similar for sample values output of query_range
        - We want to support limiting the number of time series returned for query/query_range (similar to SQL LIMIT)
    - If it becomes stable, sort() and sort_desc() will be deprecated in 3.x in favor of The New Thing

<15:45-16:00 Break>

<16:05 Chris Marchbanks rejoined>

<16:14 Goutham left>

RichiH: “xrate”?

- RichiH: Put it here because it’s the best showcase. Good arguments in favor, good arguments against. Users want it, but they trust us to get it right or not do it at all. Need to ensure implementation is up to our standards.
- Brian + Björn: explanation of use cases
    - Increments can be lost with back-to-back but non-overlapping intervals
    - Small intervals that only include one sample
    - increase() over integers produces non-integers
- Björn: Grafana is introducing special range variable to encode best practices
- Brian:

    - General problem, if you add an extra point for rate that is incorrect for x_over_time. How to square that circle semantically
    - This would replace rate, we don't need two rate functions
- MR: extend definition of range vector to include "additional samples"
- Björn: "mode switch" coercion for integer
- Explicit "wider" range vectors?
    - Brian: not user friendly, users would need to know exactly when to use them
- Björn: 3 different things we're talking about:
    - Integer operations on integer inputs return integers
    - Magically expanding range vector for range
    - Let rate() know the query_range step
- How much additional data to add?
    - Brian: Lookback delta, based on context (rate gets more data than x_over_time)
    - MR: keep it separate
    - Julien: with Brian on this
    - MR: is it a problem that the meaning of range vector is now context dependent?
    - Brian: not really if we're careful in how we define the semantics
- Björn: would need to run any proposal through testing of aggregation through a rolling restart. If we still come out at "no", we should make the whole thought process public as part of our documentation
    - RichiH: Maybe create a reference of closed major requests/issues/anti-patterns, explaining why they are a bad idea. If "no" here, then a good first candidate.

<16:32 Frederic left>

- CONSENSUS:
    - We want to rethink this from first principles. We will collect our research, thoughts, considerations, and results in "Prometheus is not feature complete" and discuss at a future dev summit.

<17:00 Julius left>

Cure53

- RichiH: Publish the report right now or wait until we changed code?
    - Brian: Only DoS issue worries me, and you can do that otherwise too
- <General agreement, but skipping call for consensus until we walked through the report>

<17:03 Bartek left>
<17:04 Goutham joined>

- PRM-02-002
    - Internet Explorer bug, can set headers
- CONSENSUS: We will work around it in the places which are easy

- PRM-02-005 TLS: Insecure TLS versions accepted (High)
    - https/tls_config.go allows users to configure TLS 1.0 & 1.1
    - MR: Think we can mandate it
    - Brian: Java 1.6 & Python 2.7 need this
    - RichiH: Agreed that Prometheus & pushgateway might be special, but all libraries and exporters can forbid this
    - Brian: I'm against different binaries doing different things, should be the same everywhere
        - RichiH: Explicitly disagree as Prometheus & pushgateway consume, rest exposes only
        - Brian: Don't think it's a good idea to let random exporter developers 2nd-guess security choices of users
    - Julien: We are adding something new, users can keep using what they use today without TLS if they can't support 1.2
    - RichiH: We can go further: Mandate TLS 1.3 or HTTP. Good security, or no security. No pretense.
    - MR: We're only adding it, we're not breaking anything existing
    - Julien: And we prohibit insecure cyphers
        - Brian: If someone has a good use case, we should support this
        - RichiH: No. They can carry a patchset and compile themselves
- CONSENSUS: We are not enough people to make such a sweeping decision
    - Possible positions were
        - Mandate HTTP or HTTPs with TLS 1.2 & 1.3
        - Mandate HTTP or HTTPs with TLS 1.3
        - Leave as-is, default to TLS 1.2 & 1.3, possible to configure lower

- PRM-02-001 Web: General HTTP security headers missing (Medium)
    - Brian: We set this previously in Prometheus 2.0, had to quickly revert due to breakage: https://github.com/prometheus/prometheus/pull/4258
    - X-Frame-Options: Frames have valid use cases, was even someone doing it at one of the Promcon talks. No mutable endpoints in default configuration, limiting clickjacking risk
    - X-Content-Type-Options: Should be safe, but check for Prometheus consoles having the right Content-type
    - X-XSS-Protection: Might be safe now that we support same-origin as a flag. Needs investigation. Cannot break API usage of Prometheus via browsers.
    - Strict-Transport-Security: Cannot be a default, includeSubDomains does not make sense - would lead to outages. May make sense as an option in tls_config.
- PProf:
    - We do not want to dig into the pprof removal unless a user comes up with a good reason to disable that.
- MR: Create on issue per PRM on GitHub

    - RichiH: Only after we release the report; will follow up on my team mail about release as I didn't get any replies yet
- CONSENSUS: Not enough people, moving to team mailing list

<18:06 End>