Prometheus Storage Working Group

Author: Frederic Branczyk
Date: September 7th 2020

When?

Biweekly on Tuesdays 9 AM UTC.
https://meet.google.com/avj-kmna-hxj

Why are we here?

Prometheus' storage implementation has largely stayed unchanged architecturally since the TSDB rewrite that landed in Prometheus 2.0. It has served us well over the past 2.5 years, but within this time we also realized its shortcomings.

This working group intends to reflect on various parts of the storage. A non exhaustive high level list of things to be potentially revisit by this working group (it's likely that this list will grow over time, more suggestions welcome):

- **Compactions**. Compactions are not streaming, which causes a variety of problems in compactions but most importantly they are slow. Essentially indices of two blocks cannot be efficiently merged. Problem: Compaction takes longer than creation of block.
    - Notes:
        - Streaming merging of the inverted index is one thing, but IDs as they are become problematic. Potentially a global series ID could help (would bring its own set of challenges with it though).
            - Sha256
            - Different problem: Series in chunk has to be ordered (e.g in order of labels).
            - Idea: Ephemeral index - new layer of index for compaction time.
            - Chunk mergins is find O(1). Merge Sort.
        - Potentially not having no symbols for young blocks.
- **Metadata**. While series metadata exists, it is not persisted into TSDB, and this would involve invasive changes into the TSDB, but a long time standing feature we want.
- **Head memory**. While chunks of samples are flushed when reaching 120 samples and then memory mapped, the head block's index is still fully loaded in memory.

Frequently flushing efficiently mergeable data and memory mapping it could be a solution (potentially an extension to the compactions point).

It is unclear whether reflecting on these points is going to just iteratively improve the existing TSDB, but it is also possible that this might evolve into a next generation of the TSDB as a whole. It is also entirely possible that we will just research and learn and nothing will change. The point of this working group is researching, evaluating and potentially experimenting with improvements to known pain points.

Meeting notes

Make sure someone logs in with the Prometheus account to admit others to the meeting

Next

2022-05-03

- No agenda, no participants. See you next time

2022-04-19

- [Jesus/Ganesh/Dieter] News on *out of order*: We've published a design doc that is available for review!
    - We shared this news on the Github issue.
    - The design doc is here
- next-gen ideas for prometheus tsdb?
    - Multi-values to reduce index size/load
    - Not keeping all labels/index in memory. Maybe an mmapped trie, btree or just use sql

2022-04-05

- No attendance

2022-03-08

- Support RLE in TSDB is not going in as a GSoC project, we were late in submitting
- https://twitter.com/horovits/status/1500840190104903684 😲10x more write throughput, 3x more read throughput than Prometheus! @PrometheusIO
  Hear about the new research paper by @Slack engineering: A Pluggable Metrics Storage Engine for the Age of #Observability by @mansu @SlackEng on OpenObservability

2022-02-22

- PTAL https://github.com/prometheus/prometheus/pull/10275
    - Matthias will propose this (support RLE in TSDB) as a project for GSoC with Ganesh as co-mentor.
- Dieter, Jesus, and Ganesh, from Grafana Labs, are actively working on ingestion of out of order samples. We will float the design doc on what we are working on soon.
    - This is the issue to track: https://github.com/prometheus/prometheus/issues/8535

2022-02-08

- No agenda

2022-01-25

- No agenda

2022-01-11

- No agenda

2021-12-14

- Matthias: promtool analyse block with different encoding. Intermediate results.

2021-11-30

- Matthias: Prom-tool analyse block with different encoding. Overhead.
    - Report number of chunks that would be stored using RLE and DoubleDelta and how many would be unchanged. What would be the final result for disk space?
    - Obligatory reference to umbrella issue: tsdb: Investigate chunk compression. · Issue #5865 · prometheus/prometheus · GitHub
- Bartek: Query Pushdown PoC on Thanos side - PomQL related changes and benefits to Prometheus

2021-11-16

- Julien: (I will not be able to attend):
    - What can we write in the CHANGELOG for https://github.com/prometheus/prometheus/pull/9673?
    - “Optimized query by skipping unneeded sorting in TSDB”
- Bartek: Agent
- Bartek: xrate Design doc: Prometheus x-rate
- Darshan: https://github.com/prometheus/prometheus/pull/9626

- Ganesh: Timescale contributing to TSDB - postings int64 - roaring bitmaps.
    - Making sure query is faster.
    - Quick fix for this
    - Thanos: Sharding blocks
    - Matthias: Dgraph implementation / roaring bitmaps used in Parca. New index version, with that implementation. It's using slightly modified postings. Not needed for Parca too much - not enough cardinality.
- Bartek: https://groups.google.com/g/prometheus-developers/c/FPe0LsTfo2E/m/yS7up2YzAwAJ

2021-11-02

- Dieter: Recent data, out of order
    - Me and Oleg
    - https://github.com/prometheus/prometheus/issues/8535
        - Only few 5 minutes out of order
    - Initially thought we can solve this too, but probably unlikely https://github.com/cortexproject/cortex/issues/2366
    - Seems like our (me+oleg) initial focus should be 5-10min OOO
    - Duplicated samples issue
    - Pretty common issue
    - More raw buffer idea / multiple parallel chunks
- Bartek: Agent status
    - Remote Write out of order compliance?
    - TODO: Explain persistence
    - Do we need to worry about data 10m?
- Bartek: Out-of-order old data upload
    - Why not build up remote writes to send for later?
- Bartek: Crazy idea - "pulled remote write"

2021-10-19

- Julien: In Prometheus, TSDB Select() is always sorted due to fanout storage.
    - https://github.com/prometheus/prometheus/blob/cda2dbbef67e2e98e059c4d25d840032d63badb7/cmd/prometheus/main.go#L452
    - https://github.com/prometheus/prometheus/blob/cda2dbbef67e2e98e059c4d25d840032d63badb7/storage/fanout.go#L92
    - We'd need to run benchmarks to assess this but I've spoken to Brian who says that this could create a 10% penalty in large histograms.
- Beorn: sparse histogram catch up
    - The work is currently embracing special properties of histograms. More generic tsdb changes might be possible in the future, but are not in scope right now.
    - Goal is to get histogram work to something usable, more generic applications later

    - Good PromCon feedback. Better storage reduction than expected. Partitioning by labels is relatively affordable in practice.
    - Good design progress allows thinking about a timeline: experimental query layer in 3 months?
    - Exemplars -> you wouldn't want one for each sparse bucket, the old bucketing scheme is actually a pretty good heuristic to define value ranges you care about. Exemplars are currently stored in ringbuffer in memory and contain their precise value anyway. No connection to the specific sparse histogram storage.
- Julien: Is there any public design doc / documentation about Parca's db, and/or a roadmap ?
    - Internal design doc exists for in-memory, might clean it up for public access
    - Next step could be discussing persistent storage
    - Profiles are currently tree structures, needs on-disk format discussion
- Dieter: series and chunk refs seem arbitrary/inconsistent
    - Proposal: create more specific types for refs
    - https://github.com/prometheus/prometheus/pull/9536/files

2021-10-05

- Dieter: Dev docs for TSDB
    - As a new comer had a bit of tough time understanding TSDB
    - There are blog posts but they get outdated
    - Would be a good idea to have developer docs in the main repo
    - Will open a PR for some initial docs for this soon

2021-09-07

- No agenda

2021-09-07 (Special, US friendly timing: 5pm CET)

- Prometheus Agent and way forward cc @Robert, @Srikrishna
    - Notes from Julien:
        - I am not able to attend, but here are my notes:
            - For me, a flag --agent is enough
            - I'd avoid duplicating main because it would be hard to keep 2 main's in sync (and probably a pain for contributors)
            - We can split main in multiple files
            - tsdb/agent seems a good location for that new wal, not to confuse it with the other wal (evenhttps://github.com/prometheus/prometheus/pull/9536/files if format is the same now)
            - Okay for me to have dedicated WAL implementation
    - Everyone in the meeting +1 with Julien's notes
    - Ganesh

        - I don’t see any blockers other than unit tests for missing parts, only needs maintainer reviews at this point
        - Splitting of main file could be a follow up work, but definitely needed
    - @Srikrishna will add unit tests to @rfratto’s PR
- Bryan: I was called for something
    - Ganesh: Different chunk encodings! But we need more people in the call to discuss that, so, moved to the next call :)

2021-08-24

- metalmatze interested in working on #5865 (tsdb: Investigate chunk compression)
    - My guess: Update/fork the design doc linked in the issue and start a proper investigation?
    - It seems Björn had some interest in this recently, is the issue up-to-date?
    - Proposal: Write CLI (promtool sub command) that analyzes all chunks in blocks and tries to better encode them. Then returns statistics about the process. Basically a first step of compaction compression?
- Julien: Storing exemplars in TSDB: how ? https://github.com/prometheus/prometheus/issues/9228
    - Seems OK to go with this, since they are now persisted on restart.

2021-08-10

- Skipped as no topics

2021-07-27

- Julien: Move vertical compaction out of experimental #9117
- Julien: Make TSDB hot-configurable #9115

2021-07-13

- Frederic: https://github.com/dgraph-io/sroar (since last time we talked about roaring bitmaps the missing part was an immutable on-disk format that could be memory-mappable)
- Ganesh: https://github.com/prometheus/prometheus/issues/8535#issuecomment-878381037 ingesting >1h old samples
- Nasty known bug #8221: Ganesh reproduced it in a unit test, now working on fix.
- Histogram is happening in the sparsehistogram branch .

2021-06-29

- Björn/Ganesh/Dieter: Working in the sparse histograms during Grafana hackathon: https://github.com/prometheus/prometheus/tree/sparsehistogram
-

2021-06-15

- Organizational [Frederic]: Move meeting 1h earlier?
- Julien: Does Prometheus compact the head if no samples are received? https://github.com/prometheus/prometheus/blame/0a8912433a457b54a871df75e72afc3e45a31d5c/tsdb/head.go#L1719
    - Looks from code that compact will produce an empty block: https://github.com/prometheus/prometheus/blob/f9e2dd069758f70957e5a447d8245865f35b78b0/tsdb/compact.go#L434
    - Q: Will the head compaction ever happen?
    - Prometheus' vanilla TSDB does not close the head block.
- Beorn: Appending histogram samples:
    - ConProf: https://github.com/conprof/tsdb/blob/master/chunkenc/byte.go#L129
- Agent mode: https://github.com/prometheus/prometheus/pull/8933

2021-05-18

- Jan, Damien: High series churn and memory consumption
    - OOMing because of churn during upgrade
    - Fredric: Replay WAL should be fixable and constant memory
    - Damien: Idea: Flushing stale chunks
    - Frederic: Makes sense, but not issue yet added
    - Jan: On normal upgraded without Prometheus restarted, continuous OOM
    - Frederic: Difficult situation, mechanism to offload the index.
    - Bartek: Flush in short block?
    - Frederic: Not sure if periodic helps
    - Bartek: Periodic no, but only failover flush only close to limits
    - Frederic. Sceptical, but not sure if this helps, we could add Flush to admin API and experiment
        - What would be nice: Index into external memory, mapping to memory.
    - Jan: Number of limits could be a trigger. Two mitigations might work but of course does not help.
        - How can we implement stale chunks?
        - Frederic: We already do this for housekeeping,
        - Julien: Backfill 1h in the past.
        - Bartek: That's a similar risk with backfill and without. We could mitigate those
        - Julien: Out of order?
        - Frederic: We can still fix it.
    - Frederic: Index in memory
        - Each series with of 1000 samples
        - Jan: Existing issue
    - Frederic: WAL Replay is import
    - Bartek: Can we have issues for this:
        - WAL Replay: https://github.com/prometheus/prometheus/issues/6934
        - Index in external memory, flush of WAL index

        - New structure
    - WAL chunks flush
    - Experimental WAL flush endpoint (postponing the issue until we sure we need this, index in ext mem will help more)

2021-05-04

- Not enough TSDB experts around to discuss complex data types and such, moving to next meeting

2021-04-20

- Frederic: WAL replay memory usage higher than normal use
    - Ganesh: Hasn't yet profiles WAL during start up yet, probably related to GC
    - Matthias: We need to fix conprof#37 and we'll profile automatically

2021-04-06

- Frederic: High cardinality (related) metrics chunk design
    - Histograms bucket not indexed. It might work there as buckets are related.
    - We are saving by storing timestamps multiple times.
    - It will be very invasive to chunk interfaces (it has to fetch series sets).
    - Secondary index? If the index is cache oblivious it might be not that bad.
    - Frederic use case, con prof. We stop treating it as log data, and instead decompose profiles and save samples as if they were series for metrics. High cardinality, but still bounded. Label is function name. Save only raw data of the sample, and keep metadata separate metadata. Kind of interning.
    - Bjorn: Metadata in TSDB? Frederic probably not. Probably ConProf will evolve to its own database.
    - Frederic: Profiles do not behave like logs but rather like metrics with extra metadata. We now understand usage patterns, so we can reflect. I don't think we have the same state for Exemplars.
    - Richi: Similar discussion were with Rob (M3db)
    - Bartek: Downsampling is the same on Thanos. It works on chunk iterating.
    - Bartek: Downsampling vs Profiles: Only few (5 aggregations vs all stacks on trace\
    -)
- Bartek: Exemplars in remote write
    - Open PR https://github.com/prometheus/prometheus/pull/8296
    - Callum is assigned. (:
    - Any blockers? Ganesh!! (:
- Bartek: Exemplars in Recording Rules
    - Bartek: Use Case: Aggregation recording rules being written to remote storage.
    - Bjorn: Query Layer for Exemplars?

    - Julien: Current API is not great. I went from dashboard to exploring grafana - examples was 1 thousands lower. Recording rule might be confusion. Maybe more selective exemplars.
    - Richi: I like the idea. Let's make it as stupid and quick as we can (experimental) for now. Tail based sampling will be more complicated. Logic on selecting. It might be more useful to have something simple but it will give us more opinions. Even if we randomly select one. We can gain experience, and our users, too.
    - Julien: Storing label from exemplar?
    - Priority - enable to ecosystem.
    - Julien: Federation etc
    - Richi: Alerting? Random exemplar? Add it as an annotation if experimental feature enabled. Query exemplar function - Jaeger trace.
    - Bartek: Sure, same as recording rule. Instead of dashboard configured with exemplar query, then alert annotation too
    - Julien: Another function in template.
- Frederic:

March 23rd

- Flaky test in Prometheus that shows up in conprof too: https://github.com/prometheus/prometheus/blob/main/tsdb/wal.go#L570-L587
    - Open PR https://github.com/prometheus/prometheus/pull/7655

March 9th

- Julien: (rare) bug up for grab (i think): https://github.com/prometheus/prometheus/issues/8318 affects cortex too
- Julien: Prometheus moved to go1.16, TSDB tested with 1.15/1.16
- Mathias: Update calendar
- Bartek: https://github.com/open-telemetry/wg-prometheus
- Julien: Prometheus Agent
- Bartek: Tailing traces
    - You can never know when traces are done
    - Probably 10m timeout for traces in Tempo
    - Single ServiceMonitor/PodMonitor for all observability signals!
- Bartek: Tempo: Index.
    - Goutham: APM capabilities. Good engineering only have same trace ID, request ID.
    - Tempo -> is not only a key value store. 1.0. Key value search features stable. How we store data.
    - How many traces are not linked to log lines.

Feb 23

- Bartek: Let's fix meet link (:

- Julien: Adding extra CI for tsdb / support multiple go versions?
    - It seems that thanos has moved to go 1.16 already ; tests still fail in Prometheus tsdb (I think it is harmless).
    - Should we have additional / dedicate CI jobs for tsdb to test it on multiple go versions ?
        - Note: we can limit that on tsdb/*.go changes
        - Goal would be to test the 2 supported go versions: latest and latest-1.
    - Which go version is cortex using?
- Julien: https://promcon.io -> CFP is open, any interesting TSDB topic/story to tell?
    - **Deadline: 5th of March**
- Pushgateway API vs OM vs remote write: (Database replication protocol. DO)
- Prometheus is OM compatible
- Benchmarks gRPC streaming! So much performant. Cooler Necessary remember series references. labels.
- OM iterating over it
- Out of order samples TSDB -> Longer to 2h TSDB
- Push forward in future?
-

Feb 9

- Julien:
    - https://github.com/prometheus/graphite_exporter/pull/145
    - Work in progress to turn a graphite database into a Prometheus TSDB
    - Currently a lot of Copy/Paste, I wish we could have public interfaces to implement instead.
- Frederic:
    -

Jan 12

- Frederic (@brancz): Improvements in postings
    - Revisit: https://docs.google.com/document/d/1FX4wWfcYI4eE9CmdPHdfjecmp2jzyWKloKuqK5suxls/edit#heading=h.fhoqupzha3ie
        - Thinking about this more I (@brancz) think this was not properly representing roaring bitmaps, as it just replaced the underlying list with the bitmap, whereas the roaring bitmap shines in intersections/unions, which is the higher level interaction
            - Proposal: Benchmark on-disk index for unions/intersections/complement with:
                - Current implementation
                - Cache oblivious compressed (“perfect”) skip-list (probably need to experiment with various heights of

the skip-list, maybe even make it dynamic based on the length of the skip-list;)
        - Roaring bitmaps
    - Goutham: Are those streamable? Skip list of ranges?
- Goutham: Other Large blocks. Compactions should be better.
    - Splitting blocks to smaller blocks:
    - We already talking about this on Thanos: https://github.com/thanos-io/thanos/pull/3390
    - Sled https://github.com/spacejam/sled
- Frederic (@brancz): Simultaneous compactions by all Prometheus servers causing CPU/IO/Network saturation
    - https://cloud-native.slack.com/archives/CK5RSSC10/p1607677841014800
- Bartek: https://docs.google.com/document/d/1ajMPwVJYnedvQ1uJNW2GDBY6Q91rKAw3kgA5fyklRrg/edit#
    - Encoding samples after the eval on node? Spending CPU on compressing.
- Bartek: Debug view.
    - Cortex holds SLA: ~300k active series
    - https://github.com/prometheus/prometheus/issues/6668
    - https://github.com/prometheus/prometheus/pull/6890
    - SQL query trace
    - https://promlens.com/ ?
    - Frederic: Do we need to leak storage?
- Bartek:
    - Pprof traces profiles to have allocations
    - Manually!
    - Barrier?
    - Ongoing issue
    - Run query in process? In thread?

December 1st 2020

- Bartek: Can we record those meetings for US-based folks (e.g Ben)
    - Frederic: We could do both? Alternate and record?
    - **Action Item(bwplotka):** Set up recording
- Bartek: FHI Cuckoo filter (Use case: 30mln+ series blocks)
    - For based on trigrams for regexp
    - More important: [time buckets e.g day/hour -> postings]
    - https://docs.google.com/document/d/14yvcAC4NL1jvz6nnOBXArchoOhzJYkJm74vi3-ljCJ8/edit#
    - Benchmarks:
        - https://gist.github.com/bwplotka/cbcbbcd1802181b7785da11dcc0f5cfd
    - LabelNames churn? Random memory access of all potential series vs sequential for postings?
    - **Action Item(bwplotka):** Gather more data
        - Allocations vs Timings for different stages?

        - Download the big block and run locally (:
    - Frederic: If iterating posting is concerning maybe we can use skip list.
        - Might be helpful. Multiple matchers for the same list.
    - Frederic: Intersection structure:
        - Roaring bitmap
- Bartek: Pushdown for Queries (For Thanos/Cortex)
    - Initial proposal
- Bartek: We should be louder on the need to improve XOR compression, no one touched this yet (:
    - It's not easy.
    - Bjoern expressed interest. We can ping him, if he needs help (:
        - Rewrite chunks for generic purpose, framework for rewriting chunks.
        - Post optimization? Or at ingestion?
        - It would be nice at ingestions?
        - It will make compaction slower if we rewrite chunks at compaction time.
- Vertical rewrite with relabelling

November 17th 2020

- Marco will do some research on compaction and how to potentially parallelize parts of it, then bring research results to next meeting
- Compaction time spent in index compaction vs chunk compaction would be interesting to understand better where to focus optimization
- Identical series chunks possibly could be optimized in vertical compaction
- Still low hanging fruit available: https://github.com/prometheus/prometheus/issues/7261

November 3rd 2020

- Object Storage Streaming: https://github.com/thanos-io/thanos/issues/3389
    - We could stream chunks.
- "Fun" Statistics extension https://github.com/thanos-io/thanos/pull/3385
    - Avg lifetime of series was ~13h
    - Churn metrics
    - Cortex: Toggle to remove chunks smaller than 5 samples.
- Statistics for Prometheus?
    - Admin API -> It's only mutating, for security only.
    - Performance risk, we no longer keep most of the index in memory - could cause OOMs
- Size of blocks limitation: Proposal WIP https://github.com/thanos-io/thanos/pull/3390
    - Backfilling?
    - Prometheus: Size based retention kicks in.
        - Chunk half?
    - Partition
        - Metric name?
- Prometheus: Size based retention is sometimes not accurate.

    - e.g During compaction and writing WAL
- Reworked index design: https://docs.google.com/document/d/1zda4_ftTqr0eLiJA7JvADwhjZLhYA0dMrgilT2WPGfM/edit?usp=sharing

October 20th 2020

- Peter, moved from Oct 6th: Include chunk size into the index to allow for more efficient remote reading of chunks. Thanos (and Cortex) don't keep segment files locally, but read small parts of it on demand. Right now when Thanos loads a chunk, it always fetches 16K of bytes, because it doesn't know how big the chunk is, and assumes that 16K is enough for everyone. Some chunks can be bigger (bug in compactor, or when using ChunkWriter manually), and Thanos then needs to refetch such chunk. Index already contains chunk offsets (in "chunk ref"), so here I suggest to also encode chunk length into the index, to make fetching of individual chunks more efficient.
    - Proposed middle ground: Try next chunk - start of chunk.
    - **TODO: Document chunk being sorted. "Do what Prometheus does", "Everything is sorted unless specified otherwise"**
    - Problem: Migrating to block storage (chunk ordering). Might be expensive, but
- Peter, moved from Oct 6th: Optimize encoding of chunk-references. Chunk-reference is computed as "(segment file index) << 32 + (offset inside segment file)", and always stored in index as 64-bit value. Segment files are limited to 512 MB (2^29), so offsets are typically less than that value, and top 3 bits are always 0. For segment file index, 1TB block would only use 2000 segment files (512MB each), so normally they fit into 11 bits. Point is… encoding chunk ref as 64-bit value for chunk ref wastes space.
- Peter, moved from Oct 6th: Simple snappy-compression of the entire index reduces it from 324 453 939 to 110 977 558 bytes. We should be able to achieve similar reduction with more efficient encoding. (**This is just a general idea, and I was more curious to hear whether space optimization is something that we want for TSDB index**)
    - There is definitely compression possibilities
        - Let's experiment and based on the result decide

October 6th 2020

- Include chunk size into the index to allow for more efficient remote reading of chunks
    - Skipped, no author.
- Optimize encoding of chunk-references (it's 64-bit value, but mostly zeroes).
    - Skipped, no author.
- Simple snappy-compression of the entire index reduces it from 324 453 939 to 110 977 558 bytes. We should be able to achieve similar reduction with more efficient encoding.
    - Skipped, no author.
- Bartek: Compaction streaming more thoughts (comment)
    - Added notes to Compaction problem description above.

- Goutham: Priority shifted: Label / Series is not streaming and OOMing ingesters.
    - Brian: Could be https://github.com/prometheus/prometheus/issues/5547 ?
    - **TODO(Goutham): Iterators Labels, Series.**
- Goutham Related: Offset for series ID 32 bit -> Introducing 64bit posting.
    - **TODO: https://github.com/prometheus/prometheus/pull/5884**
- Bartek: Query cost estimation before eval.
    - Brian: Sample limit should cover that, everything is basically already streaming
    - Head might not ok with sample limit.
    - A lot of matchers.
    - Brian: Coalescing matchers for same labelname.
    - Bartek: 1 samples for 1000 series is different 1 series for 1000 samples.
    - **TODO: Real examples**
- Bartek: Max size of the block: https://github.com/thanos-io/thanos/issues/3068
    - General bigger blocks are more efficient?
    - Size based retention ~30d blocks. Size based retention.
    - Bucketing(!): Concat long time series in separate blocks!
- Beorn: Effect of timer jitter “proves” sub-optimal bit-bucketing.
    - Reason: Go bug
    - Brian: solution for now, is to lie about TS and pretend no jitter. (Short-medium term).
- Brian: WAL reading forward compatible as of next release (2.22)
- Next:
    - Frederic: Research head index memory offloading (Brian: 10-20% increase in lookup would be ok if we can offload to external memory)
    - Think about wish-list for index redesign

September 22nd 2020

- Kick off
    - Goutham: Compression? Exemplar storage? Histograms?
    - Brian: WAL might need field for target identifier (important for sharding of transactional remote write)
    - Beorn: Monarch paper: Shared timestamps per target?
        - Other timestamp encoding:
            - Maybe low hanging fruit: If double delta is not 0 it ends up being 16bits (byte alignment?)
    - Brian:
        - On disk layout was designed to be read into memory fast, but that’s not how we use it anymore
        - 64 bit offsets haven’t been merged yet.
        - https://github.com/prometheus/prometheus/issues/7261
        - B-tree for symbol table on disk? Could move things out of heap, but relatively small improvement
    - Next:

- (help wanted) Frequency analysis about potential double delta improvements
- Goutham: Will have a look at 64bit offset PR (de-prioritized)