

# TSDB: 1 year in

Ganesh Vernekar  
November, 2019



# About me



## Ganesh Vernekar

Software Engineer, Grafana Labs

Prometheus GSoC'18 student

Prometheus Member, TSDB Maintainer

Graduated this year from IIT Hyderabad

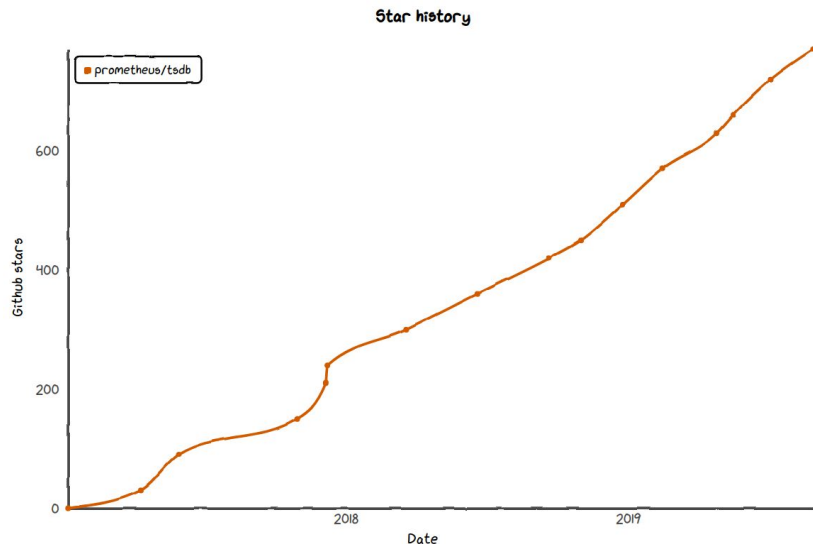
 [@\\_codesome](#)  [ganesh@grafana.com](mailto:ganesh@grafana.com)

# What is TSDB (Time Series DataBase)?

- Storage engine of Prometheus 2.x
- Independent repo in the past [prometheus/tsdb](https://github.com/prometheus/tsdb)
- Now a part of Prometheus repo, inside [tsdb](https://github.com/prometheus/tsdb) directory

# Statistics

- 500+ commits since Prometheus 2.0 release
- 60+ contributors
- 771 stars before archiving



# Some selected features/enhancements

# Backfilling

- Issue before: cannot have overlapping blocks
- Vertical query merging and compaction in Feb. 2019

## Vertical query merging and compaction #370

**Merged** gouthamve merged 46 commits into `prometheus:master` from `codesome:vertical-query-merge-and-compact`

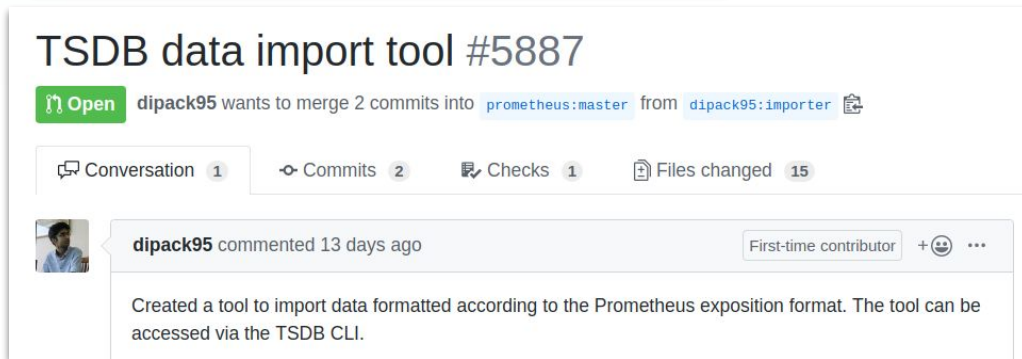
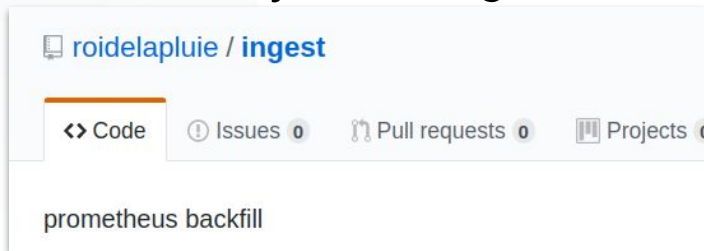
Conversation 125 Commits 46 Checks 1 Files changed 13

**codesome** commented on 4 Sep 2018 • edited ▾ Member ...

Fixes #90

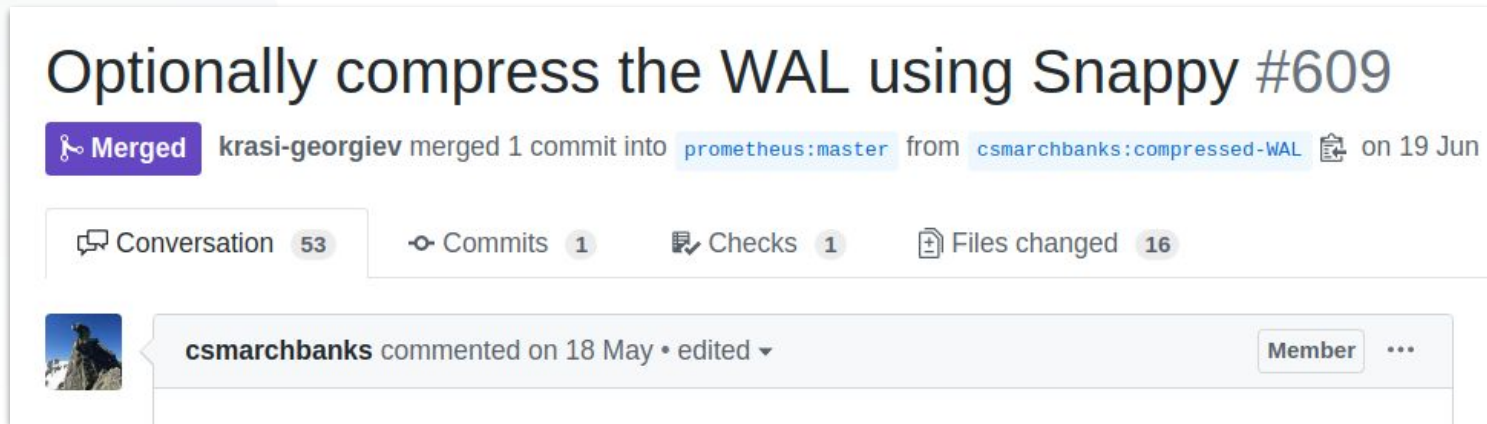
# Backfilling

- No recommended way to backfill yet
- Community is taking it forward



# WAL compression

- Optional snappy compression for WAL



- Can save up to 50% of WAL size without compromising on performance



# WAL read optimizations

## Improve WAL reading #440

**Merged** brian-brazil merged 4 commits into `master` from `wal-reading` on 14 Nov 2018

Conversation 5 Commits 4 Checks 1 Files changed 1



brian-brazil commented on 31 Oct 2018

Member ...



brian-brazil commented on 1 Nov 2018

Author Member ...

Found another useful optimisation.

Before:

```
1708.38user 17.95system 4:05.89elapsed 702%CPU (0avgtext+0avgdata 5887176maxresident)k
23526309inputs+96outputs (831major+2021538minor)pagefaults 0swaps
```

After:

```
261.94user 11.54system 1:01.40elapsed 445%CPU (0avgtext+0avgdata 3548828maxresident)k
23106453inputs+60outputs (5major+1092800minor)pagefaults 0swaps
```

So that's 6.5x less CPU, and 4x faster wall time.

# WAL read optimizations

## Decode WAL in Separate Goroutine #6230



Merged

codesome merged 2 commits into `master` from `async-wal-decode` 6 hours ago



Conversation 22



Commits 2



Checks 1



Files changed 2



csmarchbanks commented 12 days ago • edited ▼

Member



This PR moves the decoding of WAL records into a separate goroutine which passes the decoded results back using a channel. When looking at startup performance in a handful of Prometheus instances, startup time is reasonably evenly split between decoding records and processing the result so this has the potential to reduce WAL replay time by up to 50%.

Benchmarks show a reasonable performance improvement.

# WAL read optimizations



**csmarchbanks** commented 3 days ago • edited ▼

Author

Member



/benchmark cancel

Just restarted each Prometheus. Timing numbers are from the log line for starting to replay the WAL to the last segment loaded.

```
master:      7m59s
pr:          5m52s
improvement: 23.3%
```

# Allocation/memory optimization for compaction

## Use already open blocks while compacting. #441



**brian-brazil** merged 1 commit into `master` from `compact-ram` on 15 Nov 2018



Conversation 15



Commits 1



Checks 1



Files changed 3



**brian-brazil** commented on 1 Nov 2018

Member



This roughly halves the RAM requirements of compaction.

There's not much point in optimising compaction memory itself further, as loading in the new blocks would OOM at reload time. Reducing the memory used by the indexes would be the main hope.

This also makes compaction a little bit faster.

# Allocation/memory optimization for compaction

## 35% allocations

Reuse string buffer in stringTuples.Swap #654

**Merged** codesome merged 5 commits into `prometheus:master` from `codesome:string-tuples` on 10 Jul

Conversation 5 Commits 5 Checks 1 Files changed 1

codesome commented on 5 Jul

Member ...

## 19% allocations

Reuse byte buffer in WriteChunks and writeHash #653

**Merged** codesome merged 6 commits into `prometheus:master` from `codesome:write-chunks-buf` on 12 Jul

Conversation 29 Commits 6 Checks 1 Files changed 1

codesome commented on 5 Jul • edited

Member ...


# Allocation/memory optimization for compaction

6.5% allocations

## Breakdown generic writeOffsetTable #643

**Merged** codesome merged 5 commits into `prometheus:master` from `codesome:offset-table` on 10 Jul

Conversation 9 Commits 5 Checks 1 Files changed 1

 codesome commented on 2 Jul • edited Member ...

(Don't have the numbers)

## Re-use 'keys' in ReadOffsetTable #645

**Merged** codesome merged 2 commits into `prometheus:master` from `codesome:reuse-keys` on 9 Jul

Conversation 3 Commits 2 Checks 1 Files changed 1

 codesome commented on 2 Jul • edited Member ...

# Various optimizations for the queries

## Improvement on postings intersection #616

Merged

krasi-georgiev merged 4 commits into [prometheus:master](#) from [naivewong:patch-15](#) on 11 Jun

Conversation 11

Commits 4

Checks 1

Files changed 2



naivewong commented on 31 May

Contributor ...

benchmark	old ns/op	new ns/op	delta
BenchmarkIntersect/ManyPostings-8	18495452	649025	-96.49%
BenchmarkIntersect/LongPostings2-8	422	136	-67.77%
BenchmarkIntersect/LongPostings1-8	341	125	-63.34%
benchmark	old allocs	new allocs	delta
BenchmarkIntersect/ManyPostings-8	99999	1	-100.00%
BenchmarkIntersect/LongPostings1-8	4	2	-50.00%
BenchmarkIntersect/LongPostings2-8	4	2	-50.00%
benchmark	old bytes	new bytes	delta
BenchmarkIntersect/ManyPostings-8	4799963	32	-100.00%
BenchmarkIntersect/LongPostings2-8	216	98	-54.63%
BenchmarkIntersect/LongPostings1-8	208	96	-53.85%

# Various optimizations for the queries



`{foo=~"bar|baz"}` => `{foo="bar"}` or `{foo="baz"}`

Make Grafana dashboard queries faster



# Various optimizations for the queries

## Reuse Chunk Iterator #642

**Merged** codesome merged 10 commits into `prometheus:master` from `codesome:xor-chunk-opt` on 9 Jul

Conversation 28

Commits 10

Checks 1

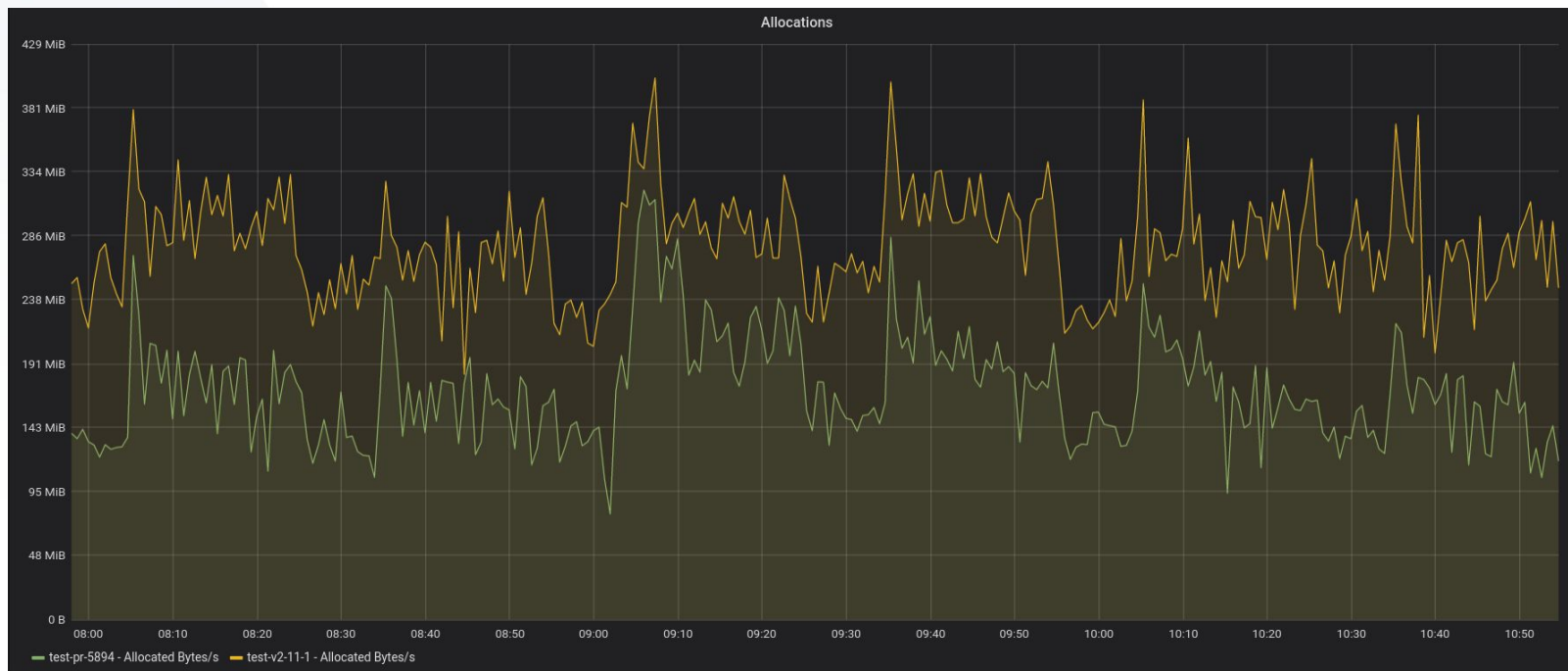
Files changed 11



**codesome** commented on 28 Jun • edited ▾

Member ...

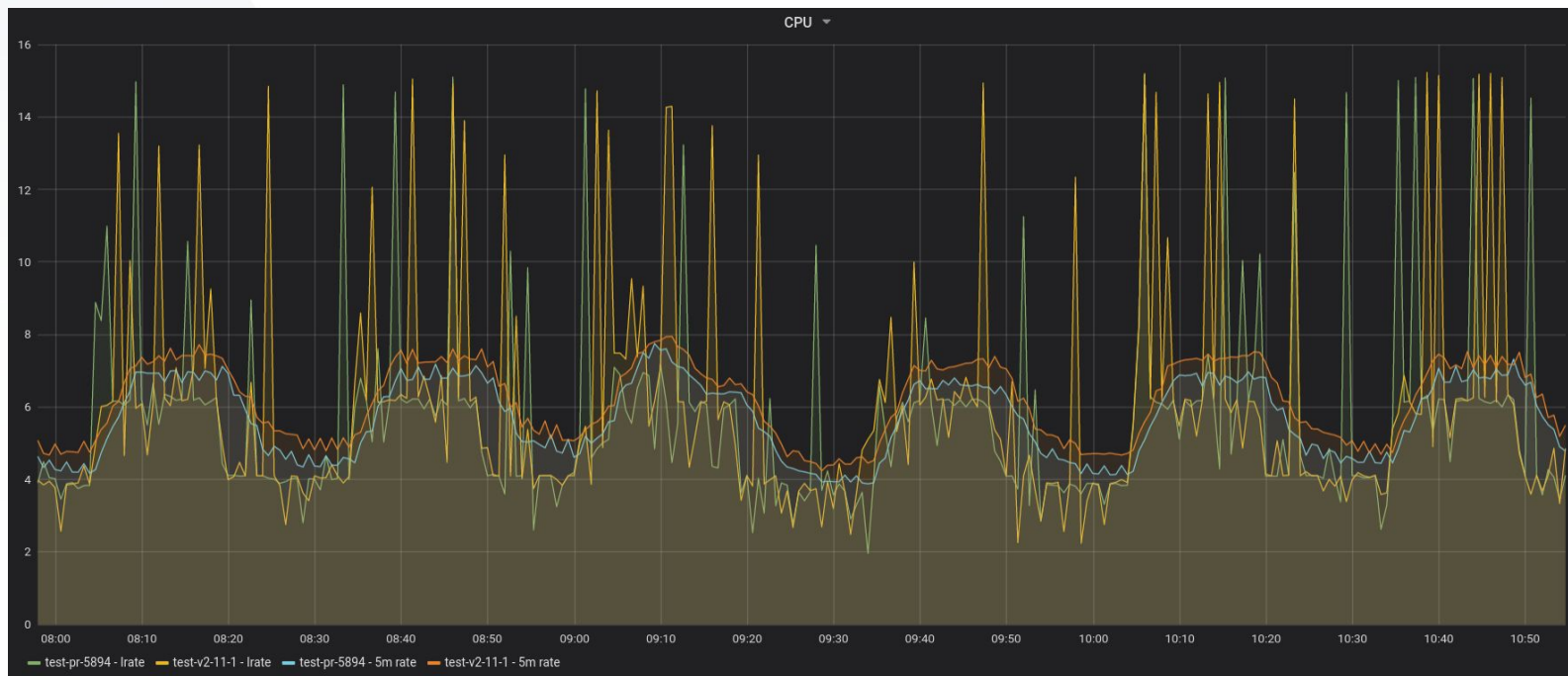
# Reuse Chunk Iterators



# Reuse Chunk Iterators

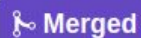


# Reuse Chunk Iterators



# Analyse churn

## Add a tsdbutil command to analyse churn etc. #478



Merged

brian-brazil merged 1 commit into `master` from `analyze` on 28 Dec 2018



Conversation 23



Commits 1



Checks 1



Files changed 2



brian-brazil commented on 13 Dec 2018

Member



This reports the cardinality of each label, the total number of label pairs, and how much series worth of time is "uncovered" by series data. Which is basically how much churn there is.

Still needs UI work, but the core is there.

# Read Only TSDB

- Interface/implementation for Read Only Mode TSDB
- Safe to use this interface with a live tsdb
- TSDB cli already uses it

# Upcoming features/enhancements

# Lifting index size limit

## tsdb: [feature request]: support index file size >64Gi



Open

davidquarles opened this issue on 19 Mar · 2 comments · May be fixed by #5884



davidquarles commented on 19 Mar



We're actually running into this on our largest cluster and it's brought thanos compaction to a standstill. I'm whittling away at cardinality, sharding our scrape targets where possible and backing down our retention policies. Hopefully all of the above will do the trick in the near-term, but it would be great to not have to worry about this in the future.

<https://github.com/prometheus/tsdb/blob/e7436e13f07511ee669876905f5147ae1713f5bd/index/index.go#L233>



# Lifting index size limit

- 32 bit postings currently (reference of series in index)
  - Index size is limited to 64 GiB
- Not difficult to hit the limit
  - 8M series with block spanning 9 days - 20 GiB

<https://grafana.com/blog/2019/10/31/lifting-the-index-size-limit-of-prometheus-with-postings-compression/>

# 64 bit postings

GSoC 2019 work by Alec Wang (GH: naivewong)

- 64 bit postings - practically unlimited index size
- Use prefix compression to store postings (48 bit prefix)
- Same performance and no increase in index size

011001	011010	101100	101101	101110	101111
--------	--------	--------	--------	--------	--------

0110	01	10	1011	00	01	10	11
------	----	----	------	----	----	----	----

<https://grafana.com/blog/2019/10/31/lifting-the-index-size-limit-of-prometheus-with-postings-compression/>

# Isolation

- TSDB has A C and D from ACID, but not I
  - Some progress in the past by Brian and Goutham
- Temporarily abandoned after that
- Plan is to add it this year
  - Rebase the work of Brian and Goutham :P

# Improved checkpointing of WAL

Potential speedup of WAL replay by a big factor

## Checkpoints with series chunks #6059

 **Draft** codesome wants to merge 7 commits into `prometheus:master` from `codesome:chunkpoint` 

 Conversation 8

 Commits 7

 Checks 1

 Files changed 8



**codesome** commented on 25 Sep

Member +  ...

I am currently exploring the idea of having the chunks of series written down into the checkpoint rather than the records and checkpointing regularly (every 10-15m).

This is similar to what I am doing in [cortexproject/cortex#1103](#), and I have seen some good WAL replay times with this and without much increase in avg disk writes.

# Thanks! Questions?



@\_codesome