
# The Prometheus Time Series Database

*Björn “Beorn” Rabenstein, Production Engineer, SoundCloud Ltd.*

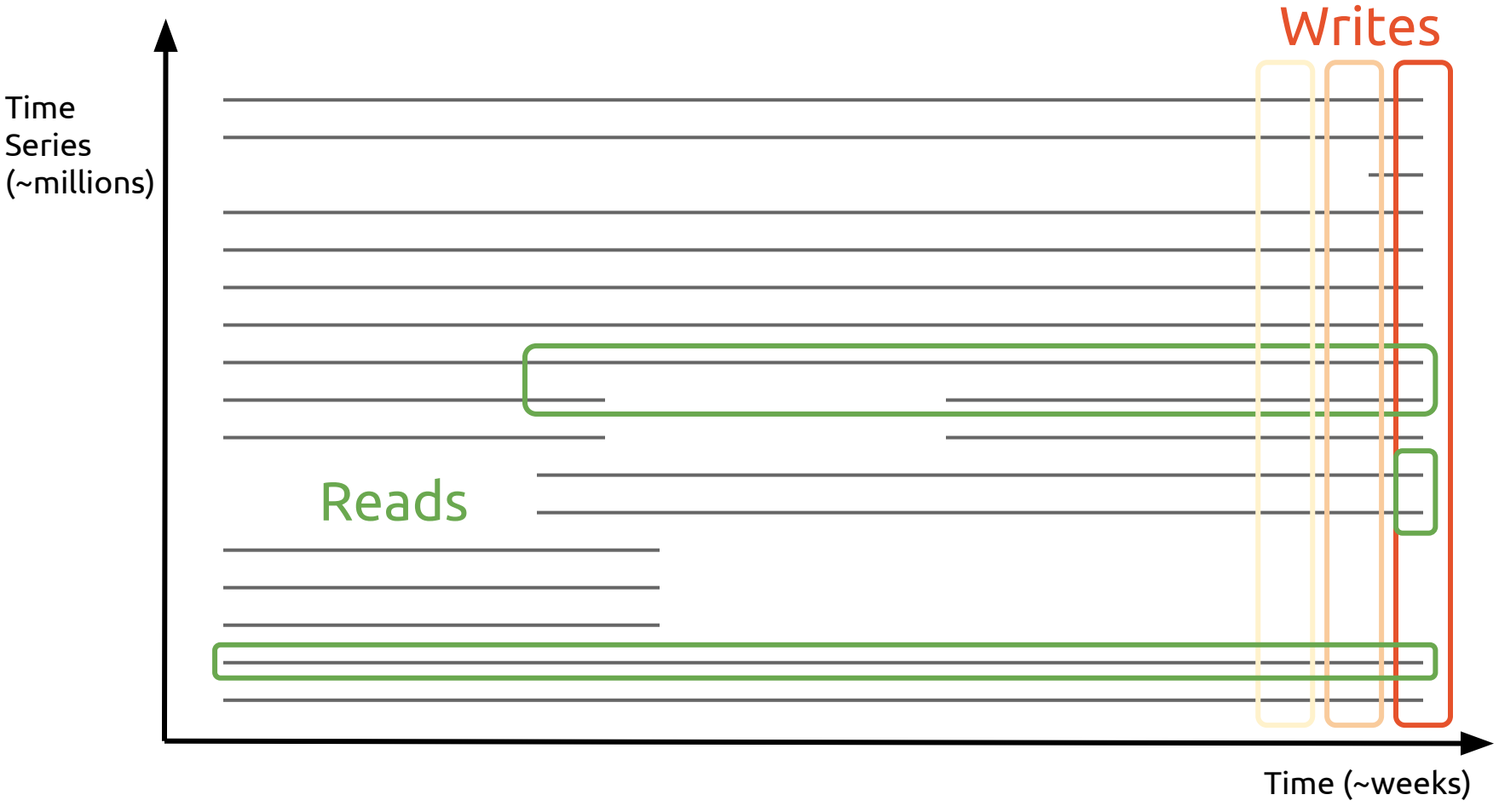
# This is about sample storage…

…not indexing (as needed for PromQL)

Sample: 64bit timestamp + 64bit floating point value

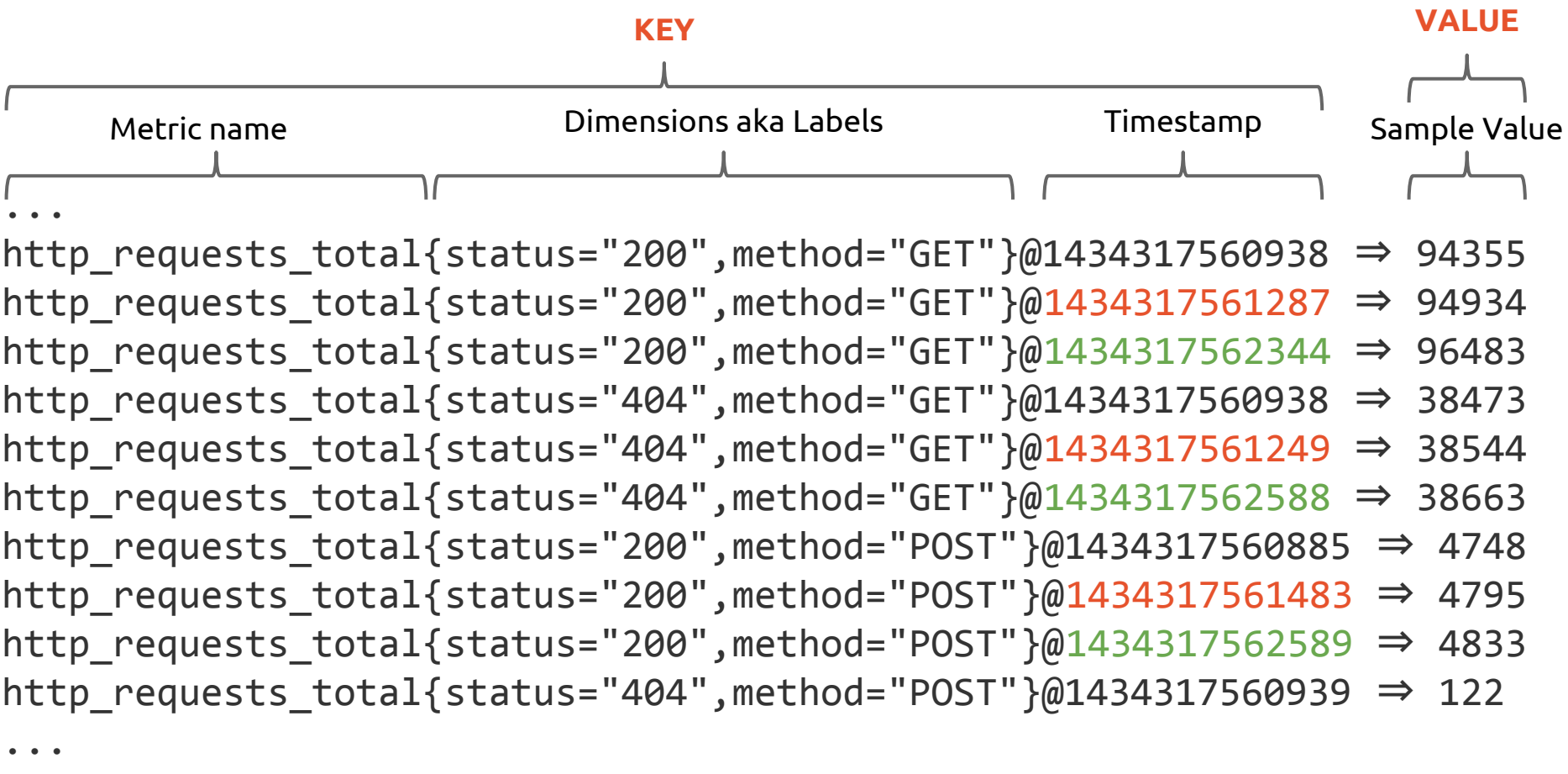
# The fundamental problem of TSDBs

Orthogonal write and read patterns.



# External storage needed

Key-Value store (with BigTable semantics) seems suitable.



# Google Cloud Bigtable Schema Design

https://cloud.google.com/bigtable/docs/schema-design-time-series

# Why is in-memory compression needed?

By Brocken Inaglory, CC BY-SA 3.0

## Gorilla vs. Prometheus

Gorilla: A Fast, Scalable, In-Memory Time Series Database, T. Pelkonen *et al.*

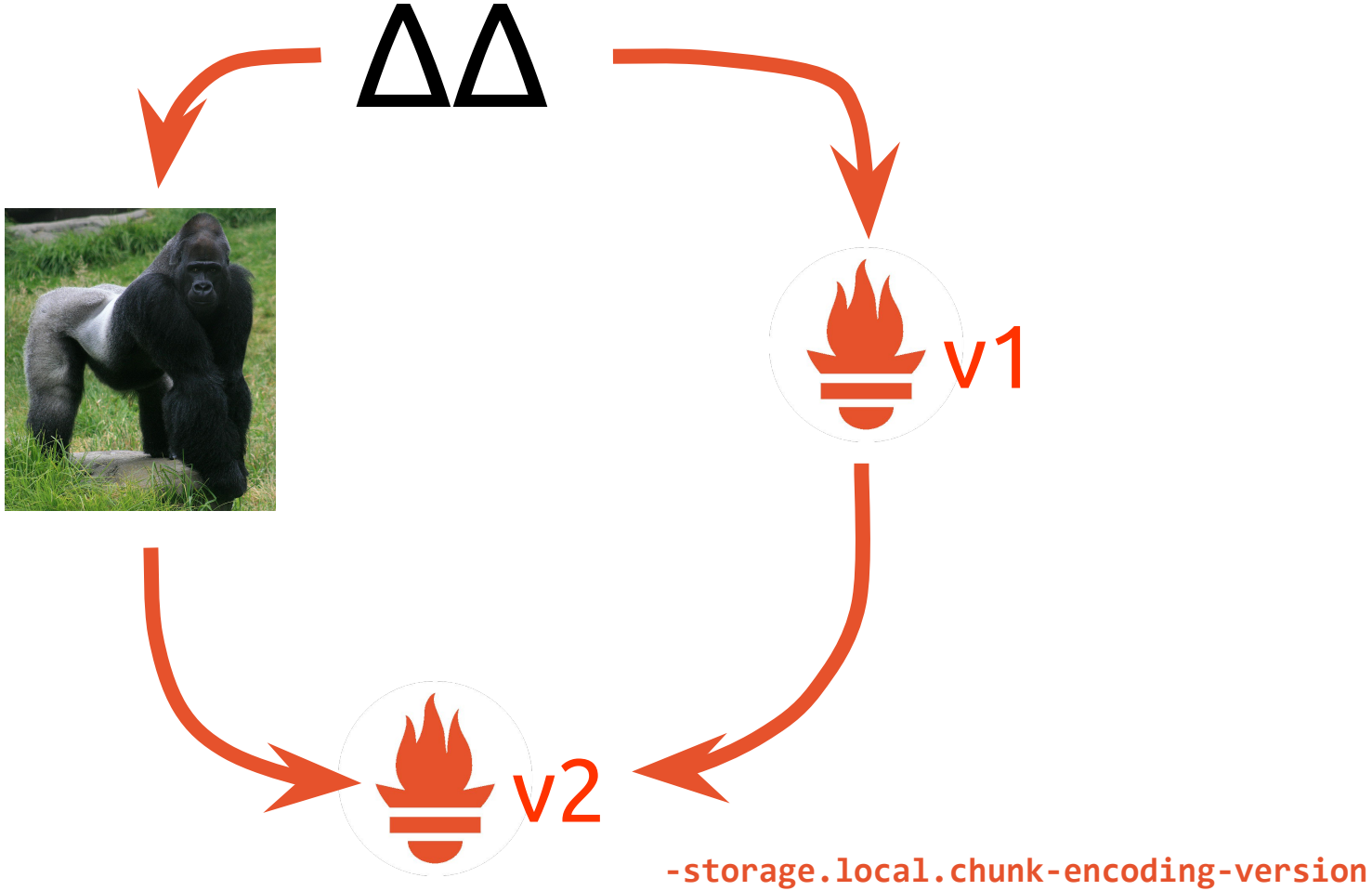
Proceedings of the VLDB Endowment
Volume 8 Issue 12, August 2015
Pages 1816-1827

1.37 bytes/sample

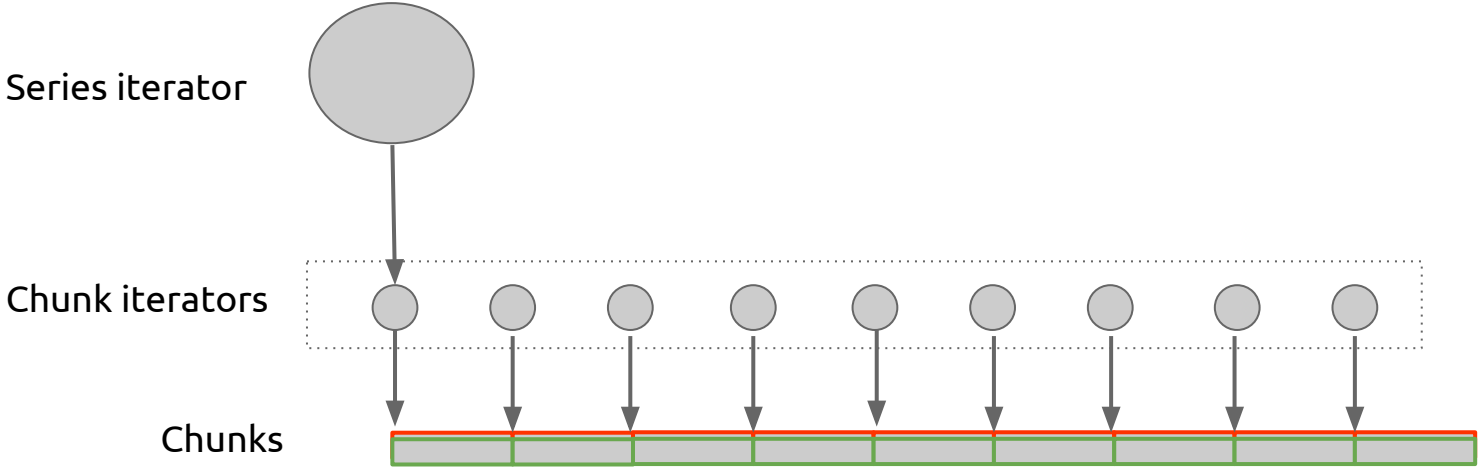
3.3 bytes/sample

- In-memory only
- 1s resolution
- Fixed-time blocks (2h)
- Not concerned with decoding

- Demultiplexing to local disk
- 1ms resolution
- Fixed-size chunks (1kiB)
- Random accessibility and decoding


ΔΔ
v1
v2
-storage.local.chunk-encoding-version

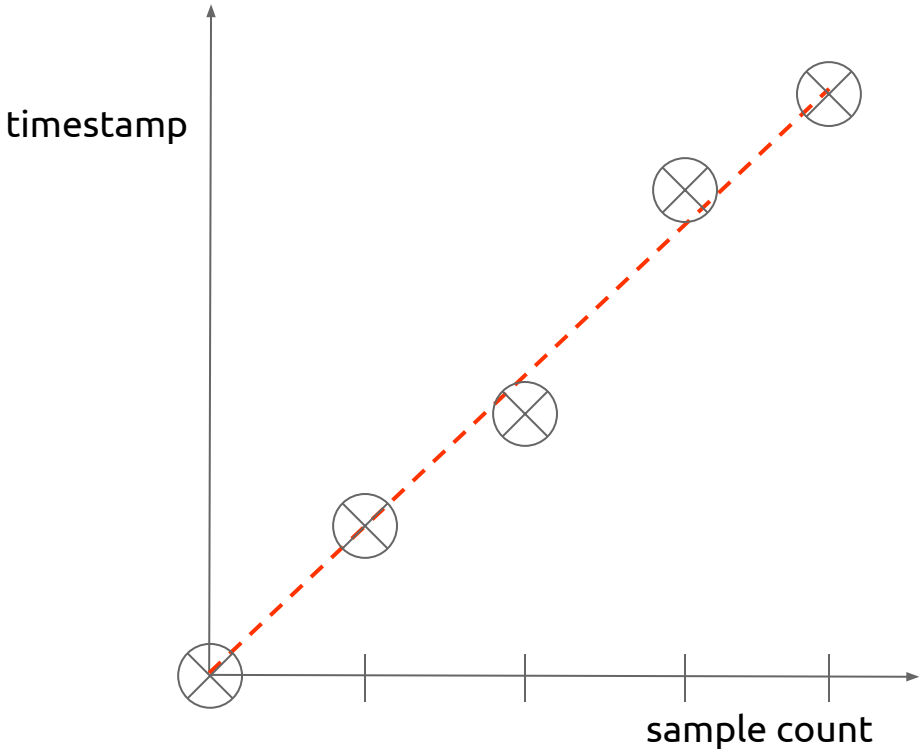
# Prometheus's chunked storage



# Timestamp compression

v1

"Pretty" regular sample intervals.




| | | |
|---|---|---|
| 0 → 1000s | | |
| 1 → 1015s | 15s | |
| 2 → 1029s | 29s | -1s |
| 3 → 1046s | 46s | 1s |
| 4 → 1060s | 60s | 0s |

store in fixed bit width (8, 16, 32) as required by sample set

| | | |
|---|---|---|
| 0 → 1000s | | |
| 1 → 1015s | 15s | |
| 2 → 1029s | 14s | -1s |
| 3 → 1046s | 17s | 3s |
| 4 → 1060s | 14s | -3s |

store with variable bit width (1, 9, 12, 16, 36) as required per sample

# Prometheus v2 timestamp encoding

Almost like Gorilla, with different bit buckets…

- If ΔΔt in [-32,31]: 10 + 6bit
- If ΔΔt in [-65536,65535]: 110 + 17bit
- If ΔΔt in [-4194304,4194303]: 111 + 23bit
- If a chunk doesn't get anything in 1h, we close it anyway.

BUT:

- If ΔΔt = 0: 0 + 7bit counting repetitions (–1)

# Value compression

Way more tricky...

64bit floating point numbers. Ugh...

Prometheus Alerts Graph Status Help

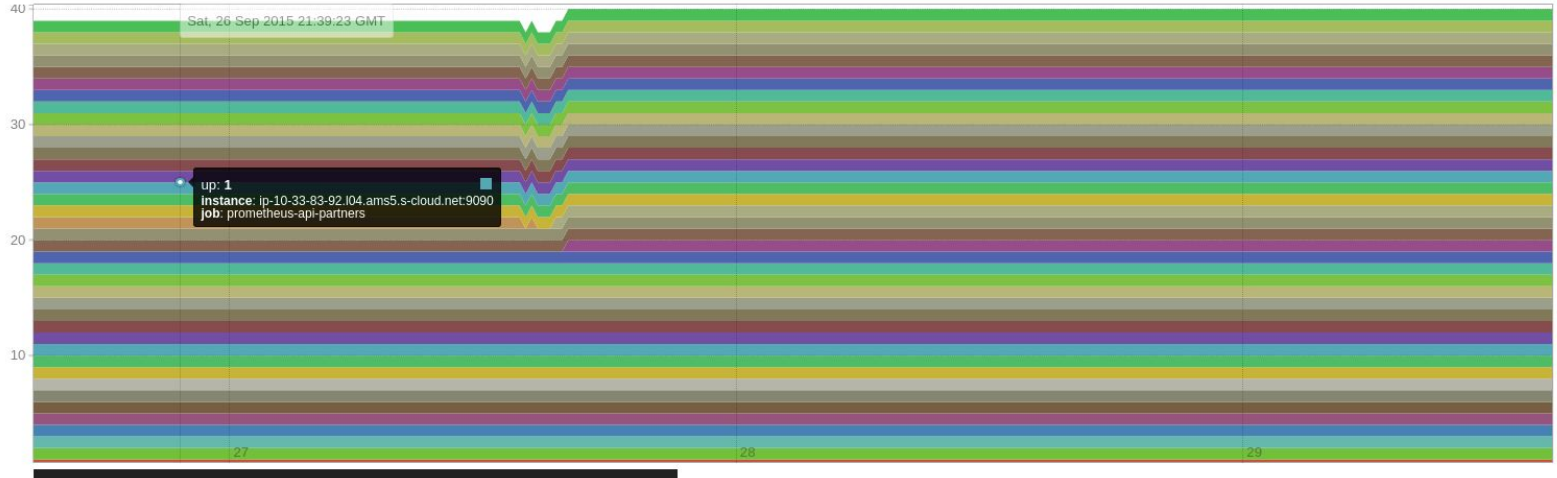
up

Load time: 219ms
Resolution: 1036s

Execute - insert metric at cursor -

Graph Console

3d Until Res. (s) stacked




- up{instance="localhost:9090",job="prometheus"}
- up{instance="ip-10-70-8-45.eu-west.s-cloud.net:9090",job="prometheus-creators"}
- up{instance="ip-10-70-46-59.eu-west.s-cloud.net:9090",job="prometheus-playback"}
- up{instance="ip-10-70-44-245.eu-west.s-cloud.net:9090",job="prometheus-iss-ea"}
- up{instance="ip-10-70-44-100.eu-west.s-cloud.net:9090",job="prometheus-metrics"}
- up{instance="ip-10-70-43-180.eu-west.s-cloud.net:9090",job="prometheus-stations"}
- up{instance="ip-10-70-40-37.eu-west.s-cloud.net:9090",job="prometheus-goku"}
- up{instance="ip-10-70-30-32.eu-west.s-cloud.net:9090",job="prometheus-content-ingestion"}
- up{instance="ip-10-70-30-229.eu-west.s-cloud.net:9090",job="prometheus-payments"}
- up{instance="ip-10-70-26-72.eu-west.s-cloud.net:9090",job="prometheus-discovery"}
- up{instance="ip-10-70-15-239.eu-west.s-cloud.net:9090",job="prometheus-revdev-staging"}
- up{instance="ip-10-70-10-173.eu-west.s-cloud.net:9090",job="prometheus-internal-tools"}

# Constant value time series

Prometheus v1/2

- Store value once (64bit float).
- Then store no values at all. The timestamp is enough.

→ 0bit/sample.

Gorilla

- Store first value (64bit float).
- Then store XOR between current and previous value (yields 0 for constant values).
- Store a single 0 bit.

→ 1bit/sample.

# The best case for a Prometheus v2 chunk

Constant metric value, perfectly regular scraping.

## 124,547 samples

(3w with 15s scrape interval)

0.066 **bits**/sample

Prometheus Alerts Graph Status Help

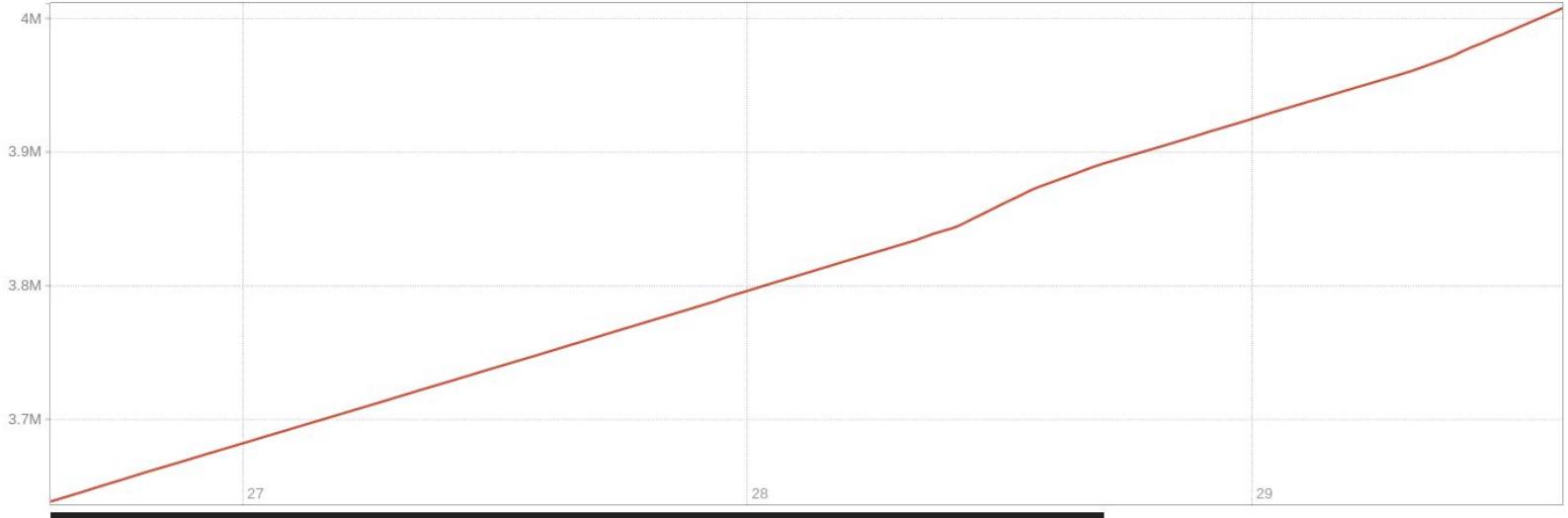
http_requests_total{code="200",handler="query_range",instance="ip-10-33-57-87.k11.ams5.s-cloud.net:9090",job="prometheus-tss",method="get"}

Load time: 65ms
Resolution: 1036s

Execute - insert metric at cursor -

Graph Console

3d Until Res. (s) stacked




Add Graph

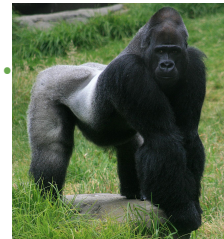
# Regularly increasing values


Prometheus v1

- Apply same double-delta encoding as for timestamps.
- Use integers (8, 16, 32 bit) internally if possible, otherwise float32. If 64bit are required, revert to storing values directly as float64.
- For values increasing with precisely the same slope, 0bit needed.

Gorilla

- As before: Store 1st value directly, then store XOR result of current value with previous value.
- Now encode it in a clever way referring to previous XOR value (similar to double-delta, but the two steps are XOR and *complicated*).



Prometheus Alerts Graph Status Help

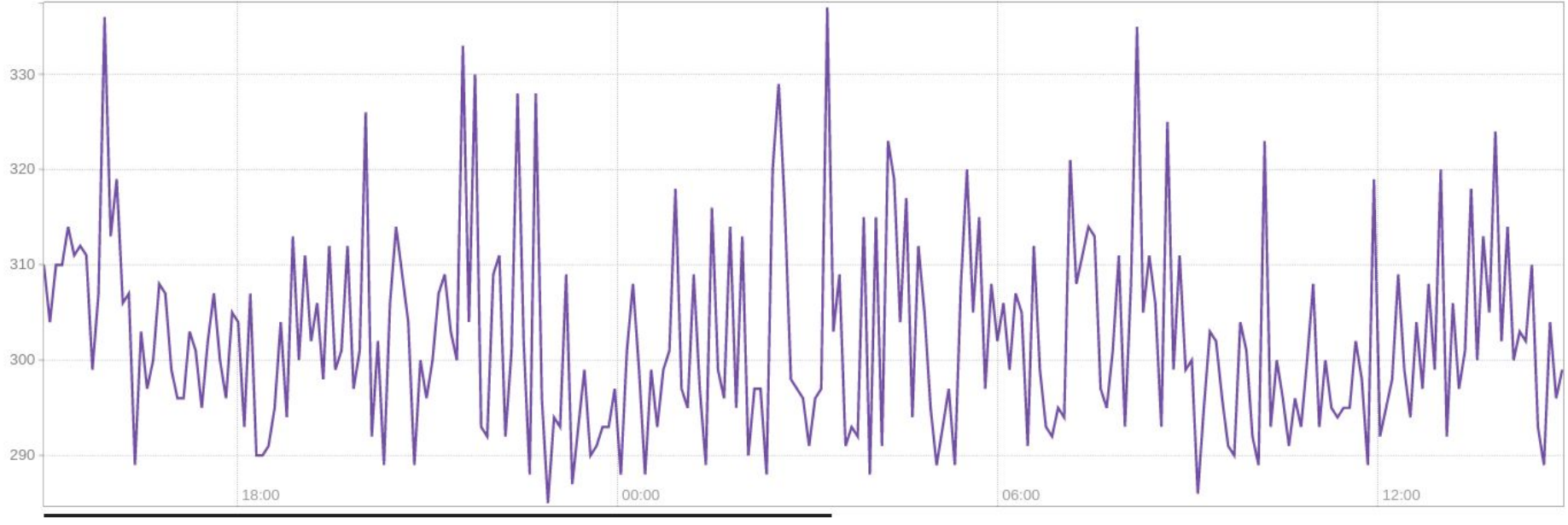
go_goroutines

Load time: 271ms
Resolution: 345s

Execute - insert metric at cursor -

Graph Console

- 1d + Until Res. (s) stacked



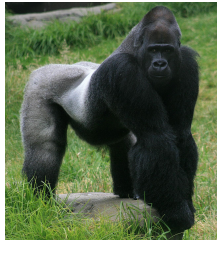
# More or less random values


Prometheus

- Double-delta encoding is tried, but fall-back to directly saving float64 values is likely.

Gorilla

- Same encoding as before. Truly random data could result in an overhead (more than 64bit per sample).



# Prometheus v2 value encoding

Picks the first that works from the following list:

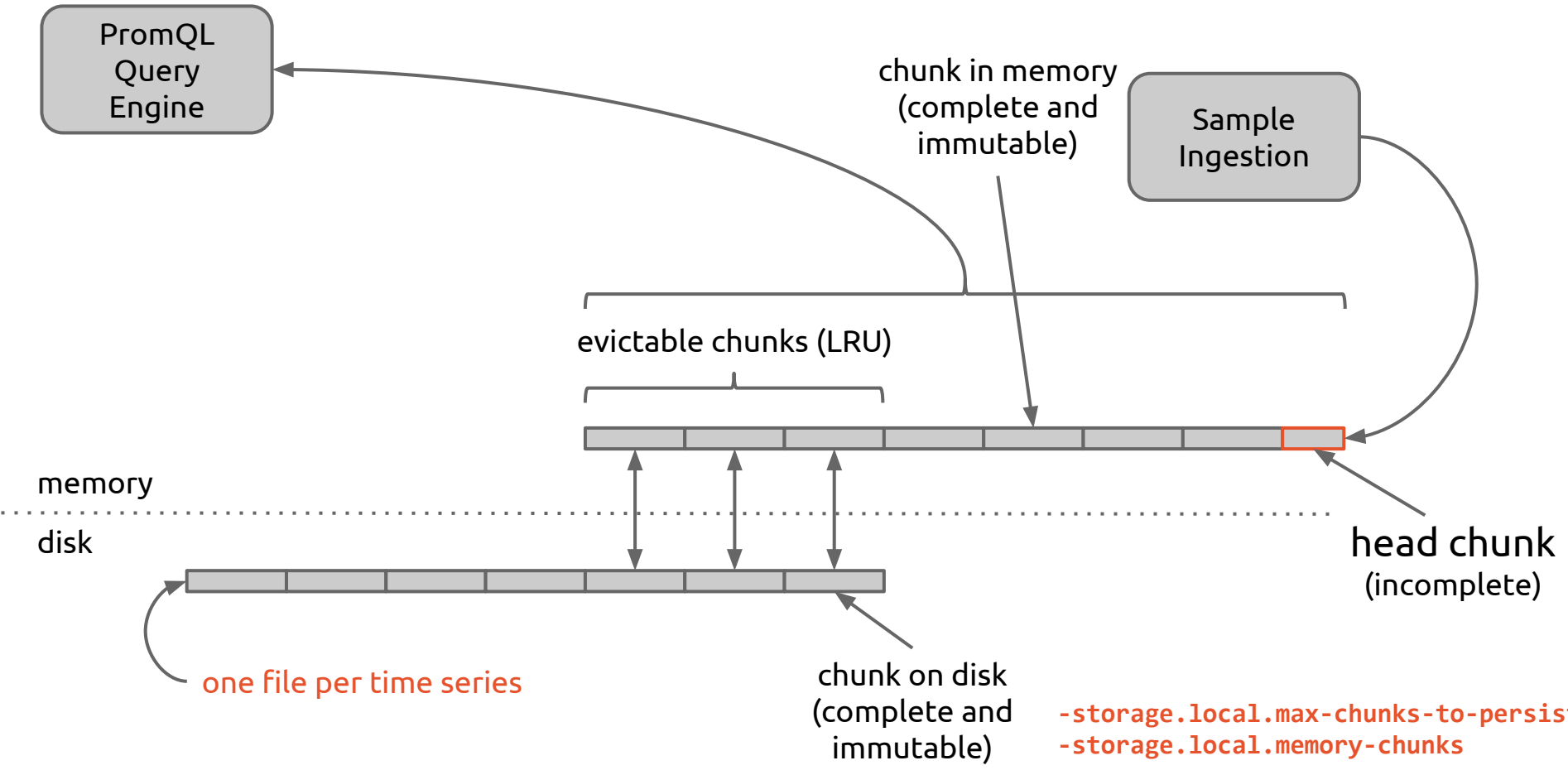
1. Zero encoding.
2. Integer double-delta encoding with 0/6/13/20/33 bit buckets
3. XOR float encoding (like Gorilla with minor tweaks)
4. Direct encoding (if XOR results in 64bit or more per value)

If you dare, check out `storage/local/varbit.go`.

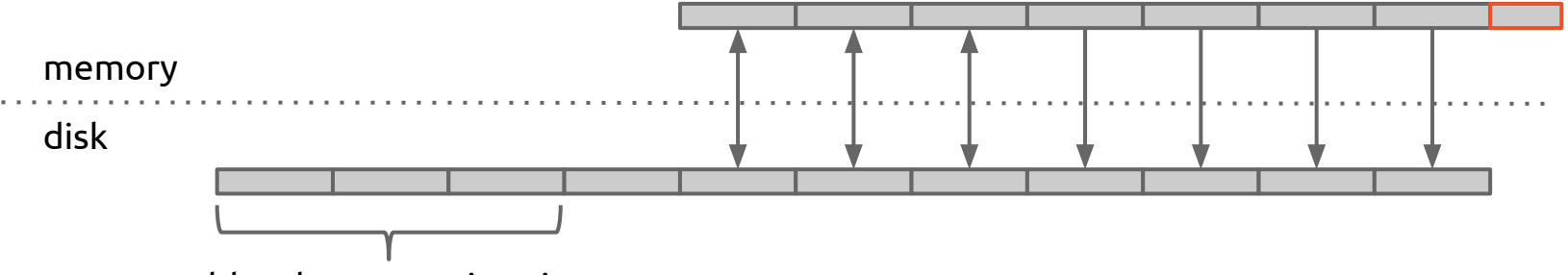
1.28 bytes/sample (typical SoundCloud server)

# Constant-size chunks.

1024 bytes.

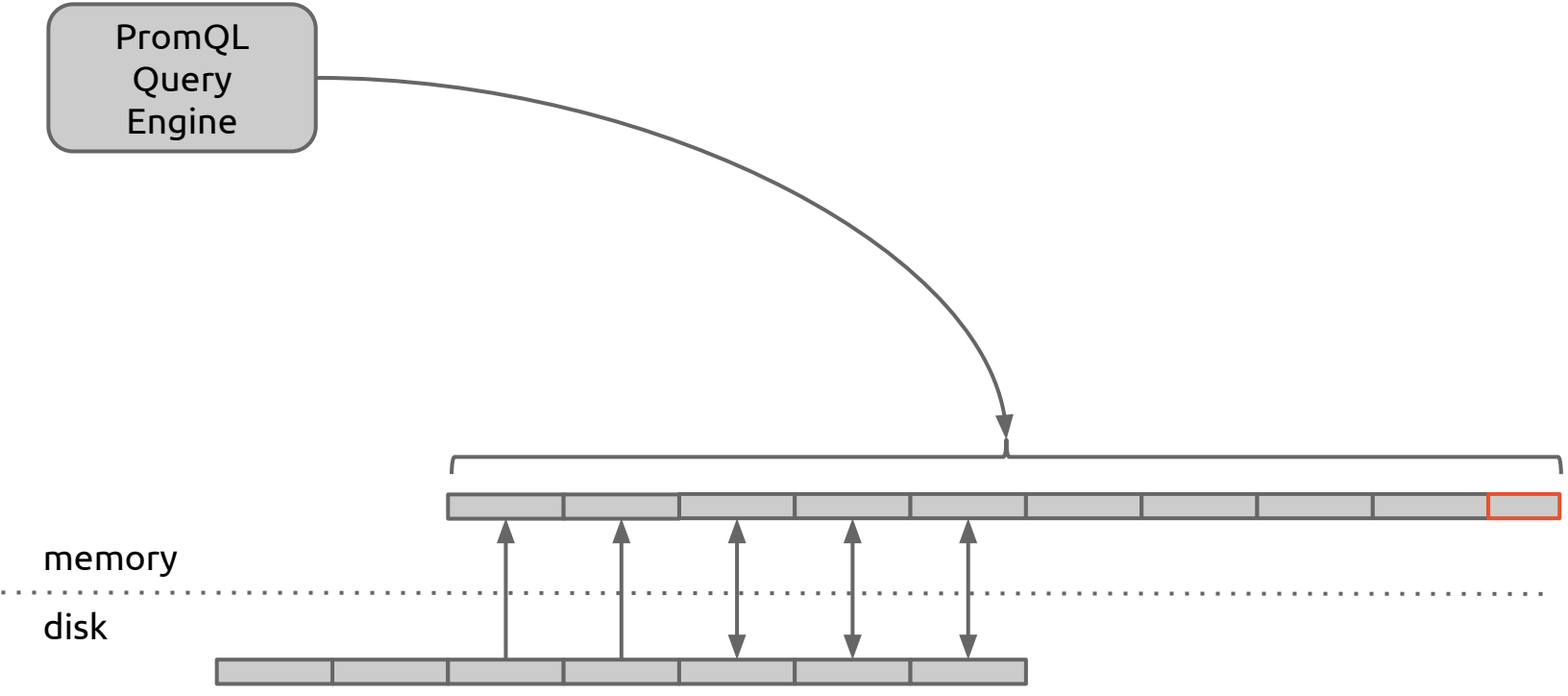


# Series maintenance.




```
-storage.local.retention
-storage.local.series-file-shrink-ratio
-storage.local.series-sync-strategy
```

# Chunk preloading.



# Checkpointing.

On shutdown and regularly to limit data loss in case of a crash.

`-storage.local.checkpoint-interval`
`-storage.local.checkpoint-dirty-series-limit`