KubeCon NA; Salt Lake City; 13.10.2024

Callum Styan, Senior SWE at Grafana Labs
Bartek Płotka, Senior SWE at Google

Unlocking Cost Savings & New Possibilities:
Your Guide to Prometheus Remote Write 2.0

Prometheus

cstyan
bwplotka

Remote Storage

Remote storage #10

Closed **juliusv** opened this issue on Jan 4, 2013 · 170 comments

juliusv commented on Jan 4, 2013 Member

Prometheus needs to be able to interface with a remote and scalable data store for long-term storage/retrieval.

64

https://github.com/prometheus/prometheus/issues/10

cstyan
bwplotka

Remote Storage

Remote storage #10

Closed **juliusv** opened this issue on Jan 4, 2013 · 170 comments

juliusv commented on Jan 4, 2013 Member •••

Prometheus needs to be able to interface with a remote and scalable data store for long-term storage/retrieval.

👍 64

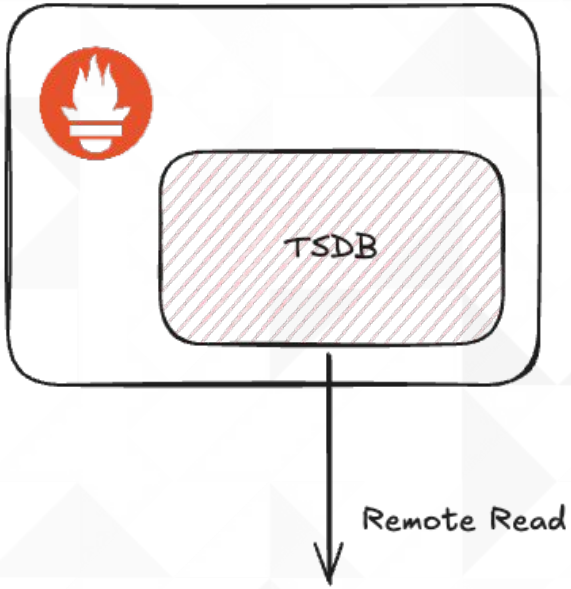
Remote storage integrations

Prometheus's local storage is limited to a single node's scalability and durability. Instead of trying to solve clustered storage in Prometheus itself, Prometheus offers a set of interfaces that allow integrating with remote storage systems.

cstyan
bwplotka

https://prometheus.io/docs/prometheus/latest/storage/#remote-storage-integrations

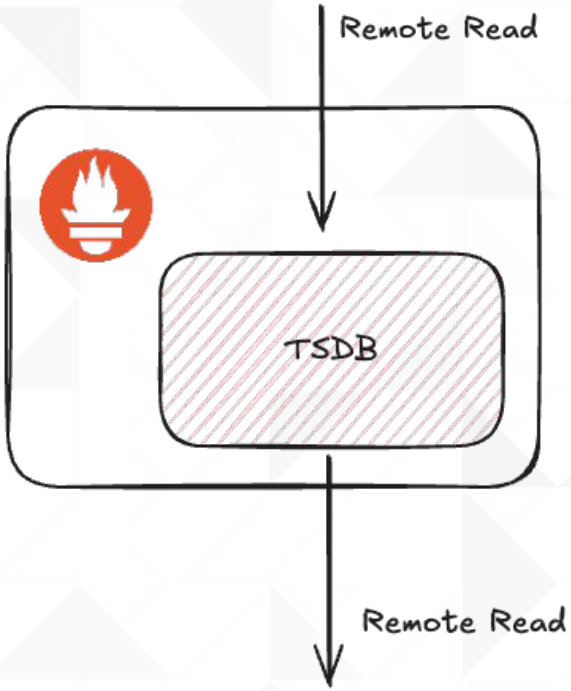
Remote Storage



"Give me samples for {job="my"} between t-100 and t"

cstyan
bwplotka

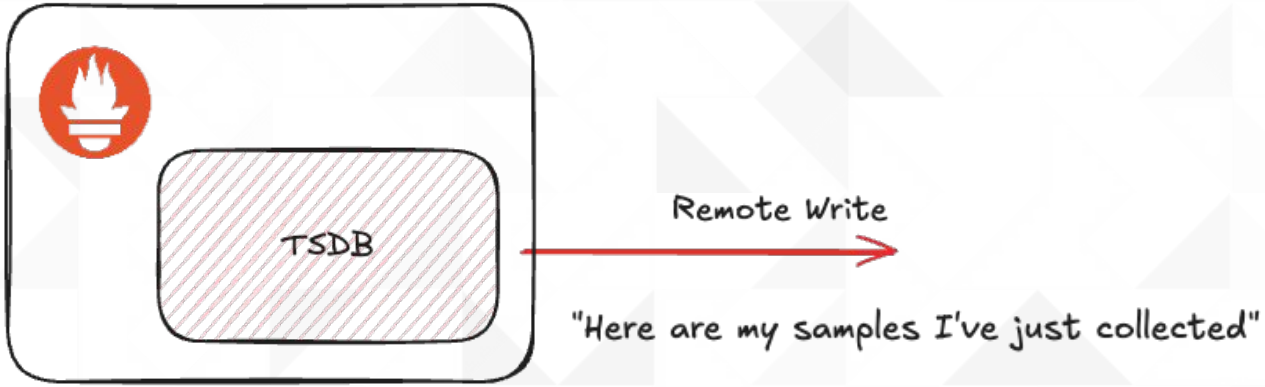
Remote Storage



"Give me samples for {job="my"} between t-100 and t"

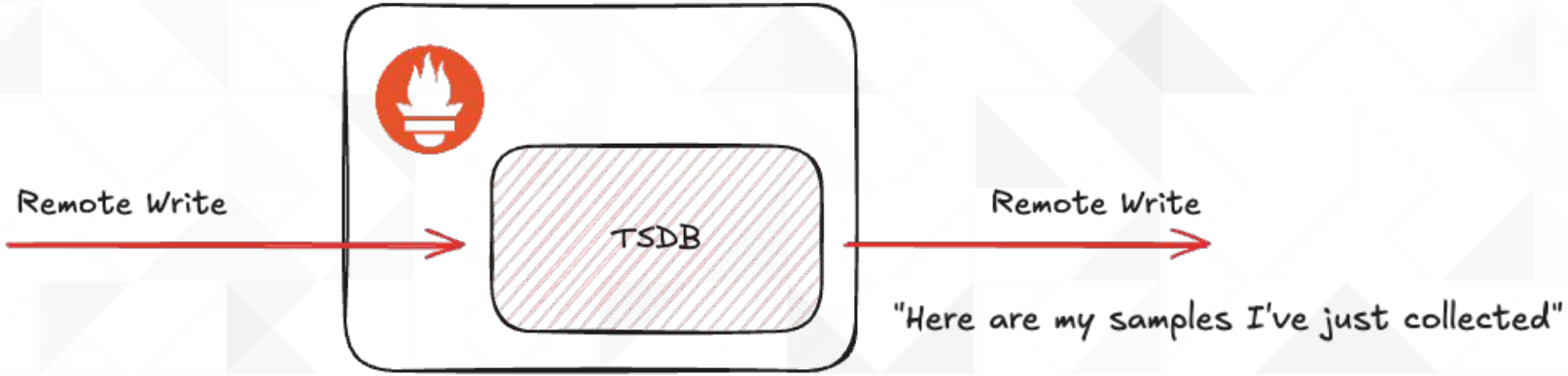
cstyan
bwplotka

Remote Storage



cstyan
bwplotka

Remote Storage



cstyan
bwplotka

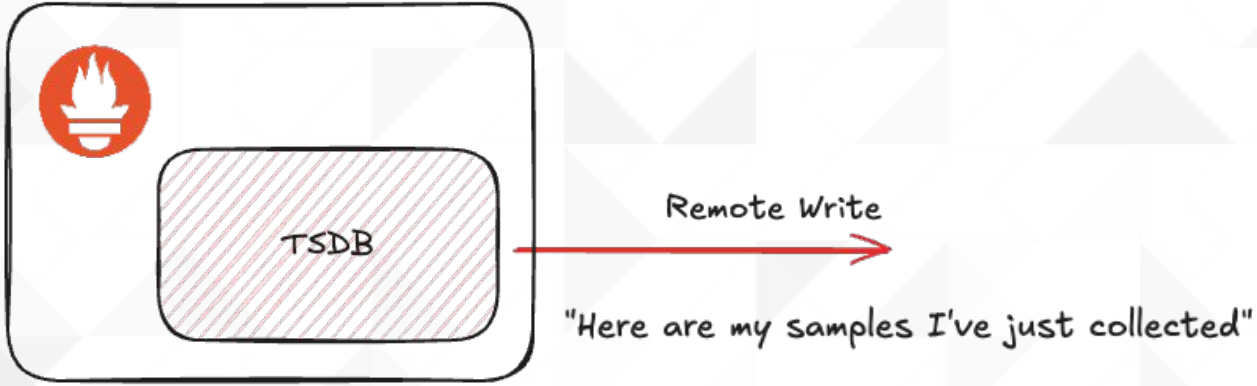
Remote Storage

PROMETHEUS REMOTE-WRITE SPECIFICATION

https://prometheus.io/docs/specs/remote_write_spec_2_0/

- Version: 2.0-rc.3
- Status: **Experimental** 🎉🎉🎉🎉
- Date: May 2024

The Remote-Write specification, in general, is intended to document the standard for how Prometheus and Prometheus Remote-Write compatible senders send data to Prometheus or Prometheus Remote-Write compatible receivers.



cstyan
bwplotka

Who are we?

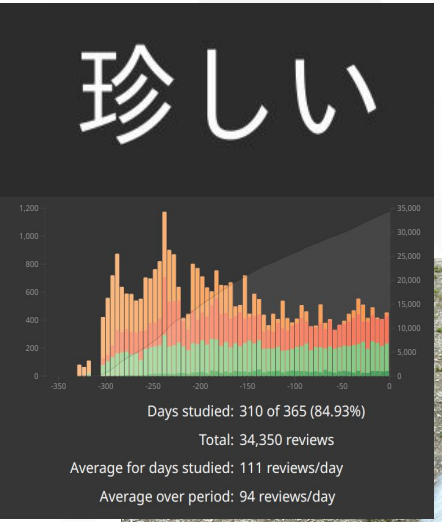
Callum Styan

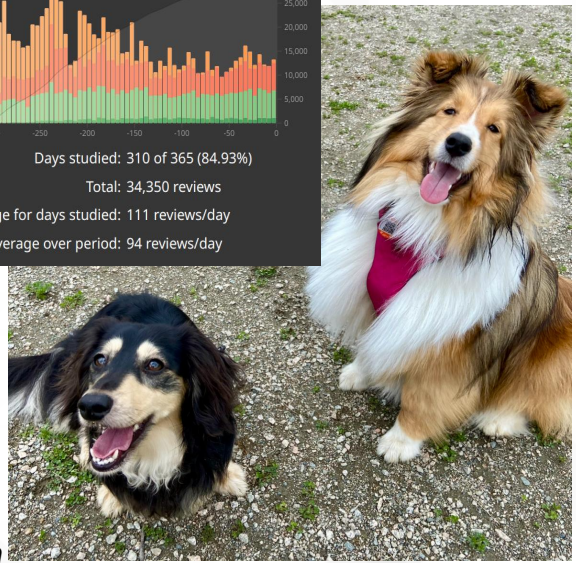
Senior Software Engineer @ Grafana Labs

- Engineer on Grafana Loki; aggregated logs storage system based on Prometheus' Label Model
- Prometheus Team Member since November 2019, maintainer of Remote Write
- Involved in protocol level work since university, reverse engineered Android ADB

cstyan
bwplotka

Who are we?





Callum Styan

Senior Software Engineer @ Grafana Labs

- Engineer on Grafana Loki; aggregated logs storage system based on Prometheus' Label Model
- Prometheus Team Member since November 2019, maintainer of Remote Write
- Involved in protocol level work since university, reverse engineered Android ADB
- Spend most of my time with my two dogs and studying Japanese

cstyan
bwplotka

Who are we?

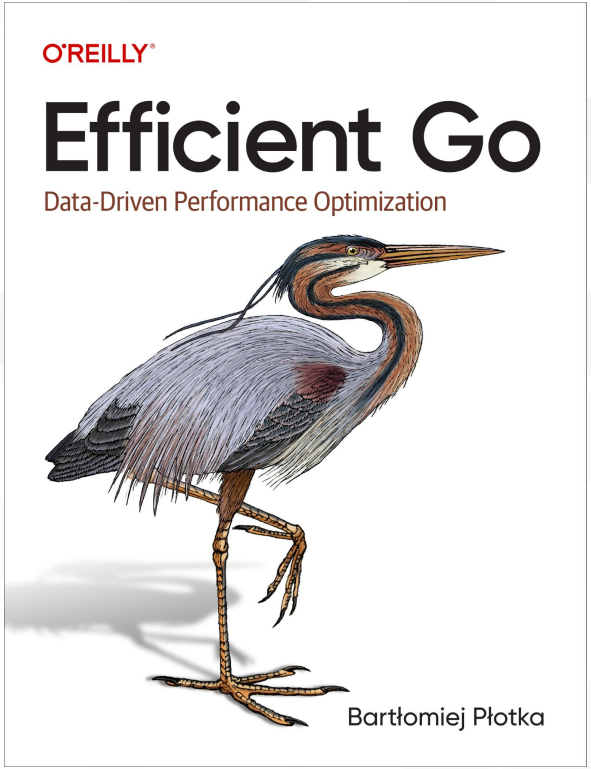
Bartłomiej Płotka

Senior Software Engineer @ Google

- *Tech Lead for Google Cloud Managed Service for Prometheus*
- *OSS Maintainer e.g. Prometheus, client_golang, Thanos and more (mostly Go libs/projects)*
- *Tech Lead for CNCF TAG Observability*

cstyan
bwplotka

Who are we?



Bartłomiej Płotka

Senior Software Engineer @ Google

- *Tech Lead for Google Cloud Managed Service for Prometheus*
- *OSS Maintainer e.g. Prometheus, client_golang, Thanos and more (mostly Go libs/projects)*
- *Tech Lead for CNCF TAG Observability*
- ***Efficient Go** book author: https://www.bwplotka.dev/book*

cstyan
bwplotka

Who are we?

Bartłomiej Płotka

Senior Software Engineer @ Google

- *Tech Lead for Google Cloud Managed Service for Prometheus*
- *OSS Maintainer e.g. Prometheus, client_golang, Thanos and more (mostly Go libs/projects)*
- *Tech Lead for CNCF TAG Observability*
- ***Efficient Go** book author: https://www.bwplotka.dev/book*
- *Motorcycling!*

cstyan
bwplotka

Why Remote Write 2.0

cstyan
bwplotka

8 Years Ago

Generic write via gRPC #1930

Merged juliusv merged 4 commits into `master` from `generic-write-grpc` on Aug 30, 2016

Conversation 48 | Commits 4 | Checks 0 | Files changed 101

juliusv commented on Aug 29, 2016 Member

This takes **@brian-brazil**'s latest gRPC generic write implementation and does a couple of cleanups / fixes on it:

- fold metric name into labels
- die on error when constructing the gRPC client connection
- remove example binary from git
- other cleanups

To add in the future: tests, compression.

https://github.com/prometheus/prometheus/pull/1930

cstyan
bwplotka

8 Years Ago

```
17
18  message Sample {
19    double value       = 1;
20    int64 timestamp_ms = 2;
21  }
22
23  message LabelPair {
24    string name  = 1;
25    string value = 2;
26  }
27
28  message TimeSeries {
29    repeated LabelPair labels = 1;
30    // Sorted by time, oldest sample first.
31    repeated Sample samples   = 2;
32  }
33
34  message WriteRequest {
35    repeated TimeSeries timeseries = 1;
36  }
```

Generic write via gRPC #1930

Merged juliusv merged 4 commits into `master` from `generic-write-grpc` on Aug 30, 2016

Conversation 48 | Commits 4 | Checks 0 | Files changed 101

juliusv commented on Aug 29, 2016 Member

This takes **@brian-brazil**'s latest gRPC generic write implementation and does a couple of cleanups / fixes on it:

- fold metric name into labels
- die on error when constructing the gRPC client connection
- remove example binary from git
- other cleanups

To add in the future: tests, compression.

https://github.com/prometheus/prometheus/pull/1930

cstyan
bwplotka

Adoption

Remote Endpoints and Storage

The remote write and remote read features of Prometheus allow transparently sending and receiving samples. This is primarily intended for long term storage. It is recommended that you perform careful evaluation of any solution in this space to confirm it can handle your data volumes.

- AppOptics: write
- AWS Timestream: read and write
- Azure Data Explorer: read and write
- Azure Event Hubs: write
- Chronix: write
- Cortex: read and write
- CrateDB: read and write
- Elasticsearch: write
- Gnocchi: write
- Google BigQuery: read and write
- Google Cloud Spanner: read and write
- Grafana Mimir: read and write
- Graphite: write
- GreptimeDB: read and write
- InfluxDB: read and write
- Instana: write
- IRONdb: read and write
- Kafka: write
- M3DB: read and write
- Mezmo: write
- New Relic: write
- OpenTSDB: write
- QuasarDB: read and write
- SignalFx: write
- Splunk: read and write
- Sysdig Monitor: write
- TiKV: read and write
- Thanos: read and write
- VictoriaMetrics: write
- Wavefront: write

Retroactive 1.0 Spec

- INTRODUCTION
- CONCEPTS
- PROMETHEUS SERVER
- VISUALIZATION
- INSTRUMENTING
- OPERATING
- ALERT MANAGER
- BEST PRACTICES

PROMETHEUS REMOTE-WRITE SPECIFICATION

- Version: 1.0
- Status: Published
- Date: April 2023

This document is intended to define and standardise the API, wire format, protocol and semantics of the existing, widely and organically adopted protocol, and not to propose anything new.

The remote write specification is intended to document the standard for how Prometheus and Prometheus remote-write-compatible agents send data to a Prometheus or Prometheus remote-write compatible receiver.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

- Introduction
  - Background
  - Glossary
- Definitions
  - Protocol
  - Backward and forward compatibility
  - Labels
  - Ordering
  - Retries & Backoff
  - Stale Markers
- Out of Scope
- Future Plans
- Related
  - Compatible Senders and Receivers
  - FAQ

https://prometheus.io/docs/specs/remote_write_spec/

cstyan
bwplotka

Stable for years

```
17
18  message Sample {
19    double value       = 1;
20    int64 timestamp_ms = 2;
21  }
22
23  message LabelPair {
24    string name  = 1;
25    string value = 2;
26  }
27
28  message TimeSeries {
29    repeated LabelPair labels = 1;
30    // Sorted by time, oldest sample first.
31    repeated Sample samples   = 2;
32  }
33
34  message WriteRequest {
35    repeated TimeSeries timeseries = 1;
36  }
```

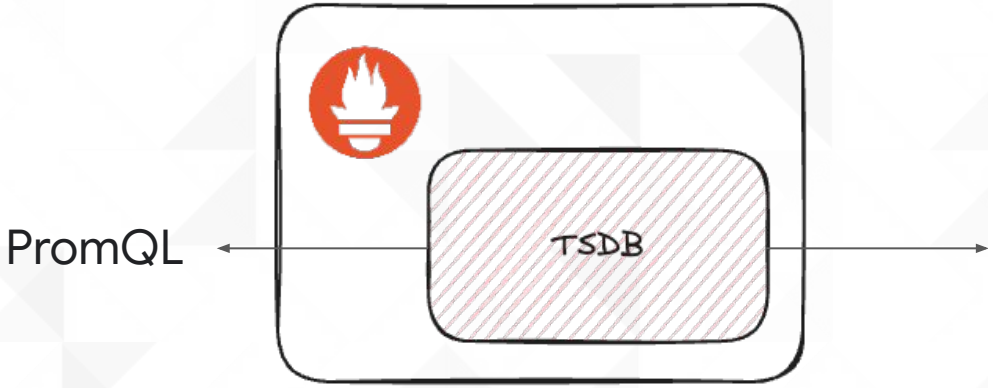
```
func Send(WriteRequest)

message WriteRequest {
  repeated TimeSeries timeseries = 1;
}

message TimeSeries {
  repeated Label labels   = 1;
  repeated Sample samples = 2;
}

message Label {
  string name  = 1;
  string value = 2;
}

message Sample {
  double value    = 1;
  int64 timestamp = 2;
}
```

8y ago

1.0 today

cstyan
bwplotka

Closely following Prometheus storage data model

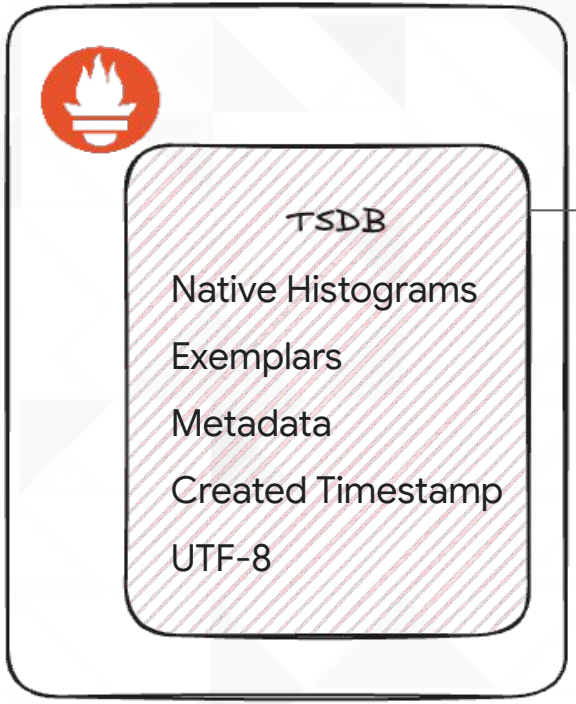



```
func Send(WriteRequest)

message WriteRequest {
  repeated TimeSeries timeseries = 1;
}

message TimeSeries {
  repeated Label labels   = 1;
  repeated Sample samples = 2;
}

message Label {
  string name  = 1;
  string value = 2;
}

message Sample {
  double value    = 1;
  int64 timestamp = 2;
}
```

1.0 today

cstyan
bwplotka

...but Prometheus storage evolves!



?

```
func Send(WriteRequest)

message WriteRequest {
  repeated TimeSeries timeseries = 1;
}

message TimeSeries {
  repeated Label labels   = 1;
  repeated Sample samples = 2;
}

message Label {
  string name  = 1;
  string value = 2;
}

message Sample {
  double value    = 1;
  int64 timestamp = 2;
}
```

1.0 today

cstyan
bwplotka

1.x Experiments

```
message WriteRequest {
  repeated prometheus.TimeSeries timeseries = 1 [(gogoproto.nullable) = false];
  // Cortex uses this field to determine the source of the write request.
  // We reserve it to avoid any compatibility issues.
  reserved  2;
  repeated prometheus.MetricMetadata metadata = 3 [(gogoproto.nullable) = false];
}

// TimeSeries represents samples and labels for a single time series.
message TimeSeries {
  // For a timeseries to be valid, and for the samples and exemplars
  // to be ingested by the remote system properly, the labels field is required.
  repeated Label labels         = 1 [(gogoproto.nullable) = false];
  repeated Sample samples       = 2 [(gogoproto.nullable) = false];
  repeated Exemplar exemplars   = 3 [(gogoproto.nullable) = false];
  repeated Histogram histograms = 4 [(gogoproto.nullable) = false];
}
```

Experimental 1.x proto today

```
func Send(WriteRequest)

message WriteRequest {
  repeated TimeSeries timeseries = 1;
}

message TimeSeries {
  repeated Label labels   = 1;
  repeated Sample samples = 2;
}

message Label {
  string name  = 1;
  string value = 2;
}

message Sample {
  double value    = 1;
  int64 timestamp = 2;
}
```

1.0 today

cstyan
bwplotka

So the 2.0 work started!

cstyan
bwplotka

2.0 Requirements

Prometheus RW requirements:

- *low complexity, light dependencies*
- *easy upgradability of existing clients*
- *easy scalability of receivers*
- *close to Prometheus storage model*

Development Process

- Originally a Grafana Labs Hackathon project by @cstyan
- Working group on CNCF slack #prometheus-prw2-dev
- Fortnightly meetings
- prometheus/proposal repo (“Why decisions were made”)
  - https://github.com/prometheus/proposals/pull/35
- Spec file in the prometheus/docs repo (“What has been decided”)
  - https://prometheus.io/docs/specs/remote_write_spec_2_0/

What 2.0 Enables?

Spec: Partial write statistics

- Problem:
  - Send 100 metrics via Remote Write, 99 are ingested ok, one has a problem (too old, duplicate, etc)
  - How do you know?
    - A single HTTP status code is too coarse
- Solution:
  - Headers to provide insight into partial writes

```
X-Prometheus-Remote-Write-Samples-Written: 99
X-Prometheus-Remote-Write-Histograms-Written: 12
X-Prometheus-Remote-Write-Exemplars-Written: 2
```

cstyan
bwplotka

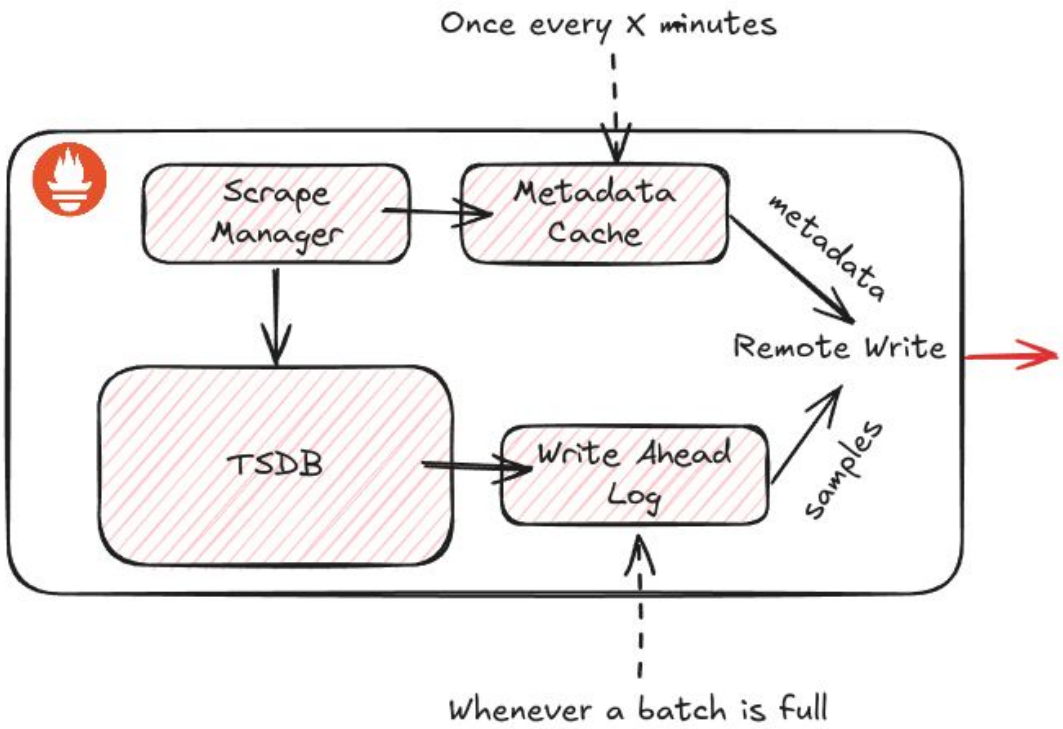
Spec: Native Histograms, CT, Exemplars, UTF-8

- PRW 2.0 Spec has native support for many of recently added features
  - Native Histograms
  - Created Timestamp
  - Exemplars
  - UTF-8 support
- Structure and Transactionality
- Separates Remote Read and Remote Write

```
// TimeSeries represents a single series.
message TimeSeries {
  repeated uint32 labels_refs = 1;

  repeated Sample samples = 2;
  repeated Histogram histograms = 3;
  repeated Exemplar exemplars = 4;

  Metadata metadata = 5;

  int64 created_timestamp = 6;
}
```

cstyan
bwplotka

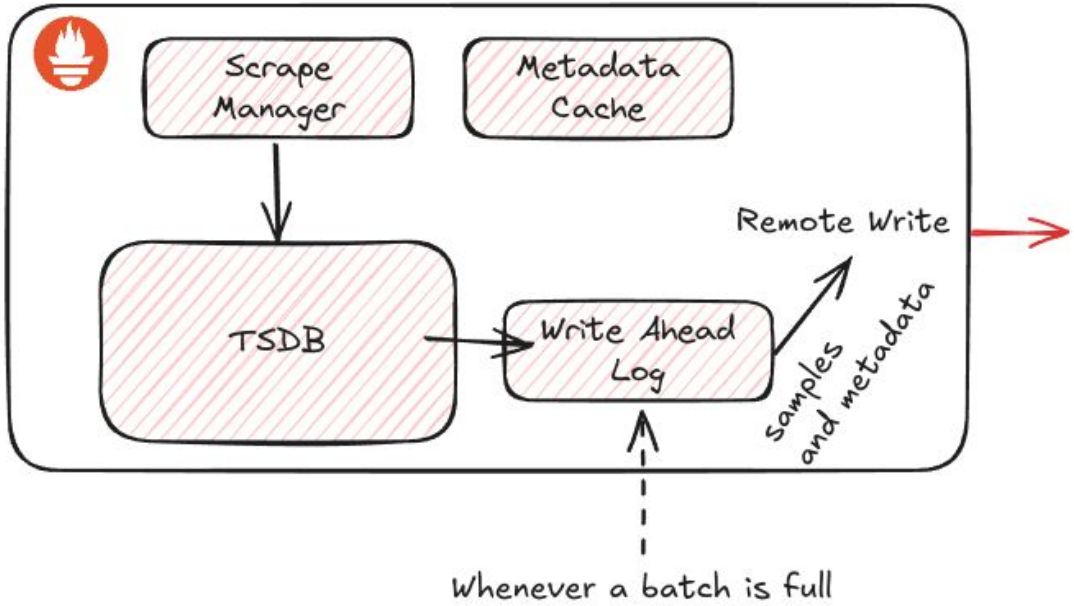
Spec: Metadata Always On




```
message WriteRequest {
  repeated prometheus.TimeSeries timeseries = 1;
  // …
  repeated prometheus.MetricMetadata metadata = 3;
}

message TimeSeries {
  repeated Label labels         = 1;
  repeated Sample samples       = 2;
  repeated Exemplar exemplars   = 3;
  repeated Histogram histograms = 4;
}
// …
```

cstyan
bwplotka

Spec: Metadata Always On




```
message Request {
  reserved 1 to 3;
  repeated string symbols = 4;
  repeated TimeSeries timeseries = 5;
}

// TimeSeries represents a single series.
message TimeSeries {
  repeated uint32 labels_refs = 1;
  repeated Sample samples = 2;
  repeated Histogram histograms = 3;
  repeated Exemplar exemplars = 4;
  Metadata metadata = 5;
  int64 created_timestamp = 6;
}
// …
```


cstyan
bwplotka

Spec: Metadata Always On

- Problem(s):
  - Metadata cache only updated upon scrape
  - Only per metric family name (unique "*__name__*" value)
- Solutions:
  - Metadata in the WAL (thanks: Paschalis Tsilias *et al*)
    - Could remove chunks of caching code
  - More repeated Data vs Stateful Implementation
    - Trade-off of more data balanced by string interning
  - Metadata now per series

Wait, more features == overhead, no?

Spec: String Interning

```
{"__name__"="kube_pvc_capacity_bytes", "cluster"="ops-us-east -0", "job"="foo", "node"="node_a"}
{"__name__"="kube_pvc_capacity_bytes", "cluster"="ops-us-east -0", "job"="foo", "node"="node_b"}
{"__name__"="kube_pvc_capacity_bytes", "cluster"="ops-us-east -0", "job"="bar", "node"="node_a"}
{"__name__"="kube_pvc_capacity_bytes", "cluster"="ops-us-east -0", "job"="bar", "node"="node_b"}
```

```
{symbols:["","__name__","kube_pvc_capacity_bytes","cluster","ops-us-east-
0","job","foo","node","node_a","bar","node_b"]}

{1,2,3,4,5,6,7,8}
{1,2,3,4,5,6,7,10}
{1,2,3,4,5,9,7,8}
{1,2,3,4,5,9,7,10}
```

cstyan
bwplotka

Spec - String Interning

Remote Write 1.0

```
message TimeSeries {
   repeated Label labels = 1;
   // …
}

message Label {
  string name  = 1;
  string value = 2;
}
// …
```

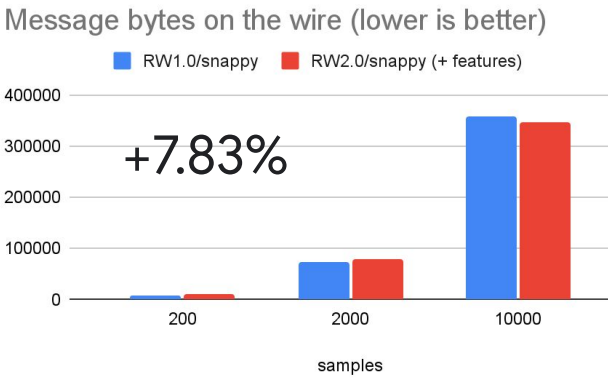
VS

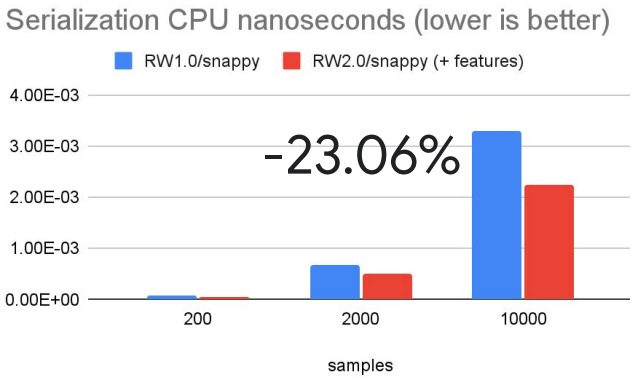
Remote Write 2.0

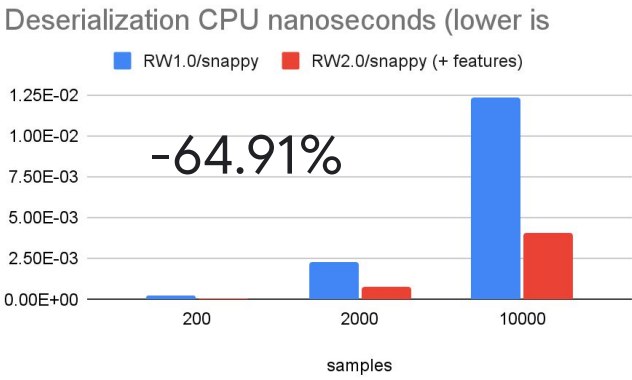
```
message TimeSeries {
   repeated string symbols = 1;
   // …
}

message Label {
  // labels_refs is a list of label
  // name-value pair references,
  // encoded as indicies to the
  // WriteRequest.symbols array.
  repeated uint32 labels_refs = 1;
  // …
}
// …
```

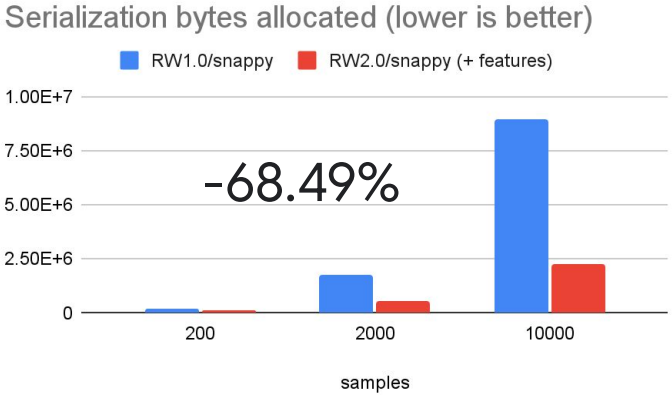
cstyan
bwplotka

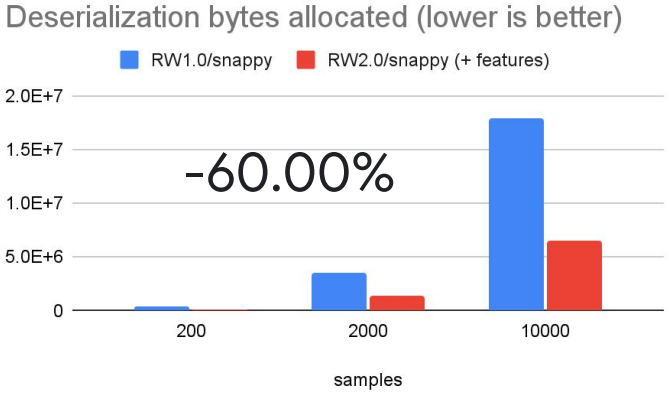
[1.0] vs [2.0 + features]





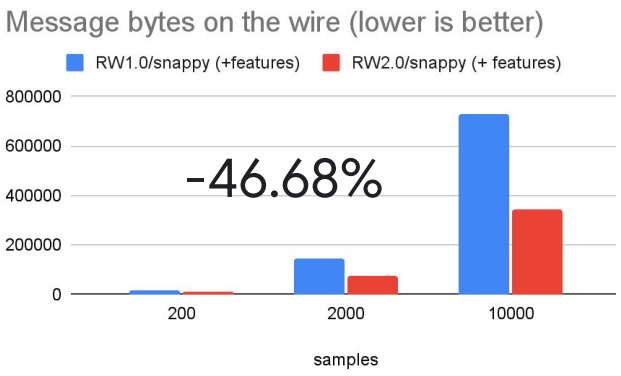


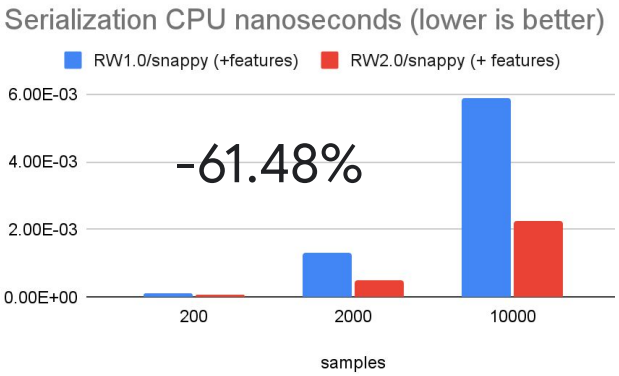


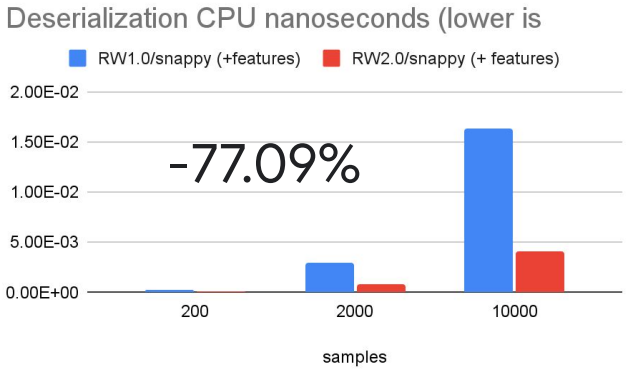


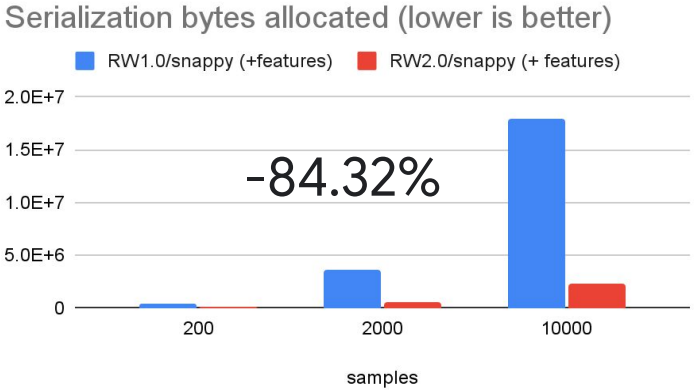
cstyan
bwplotka

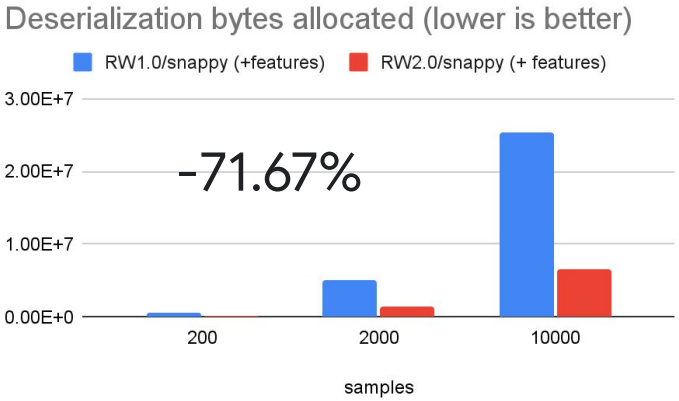
[1.0 + features] vs [2.0 + features]





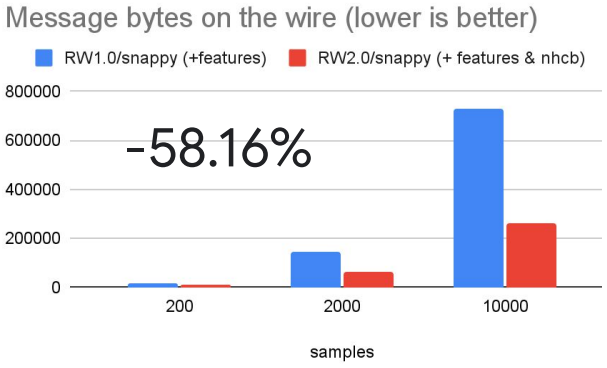


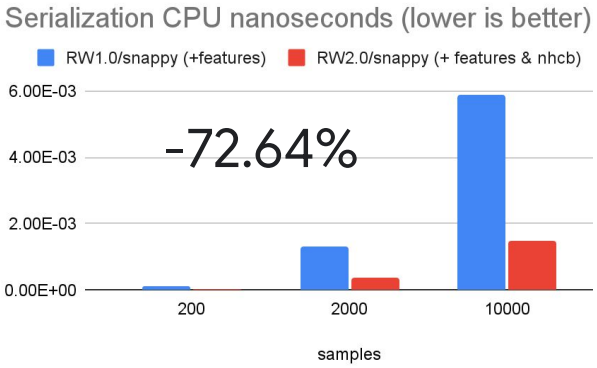


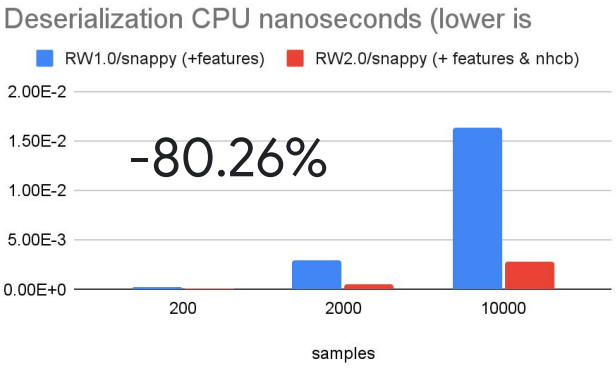


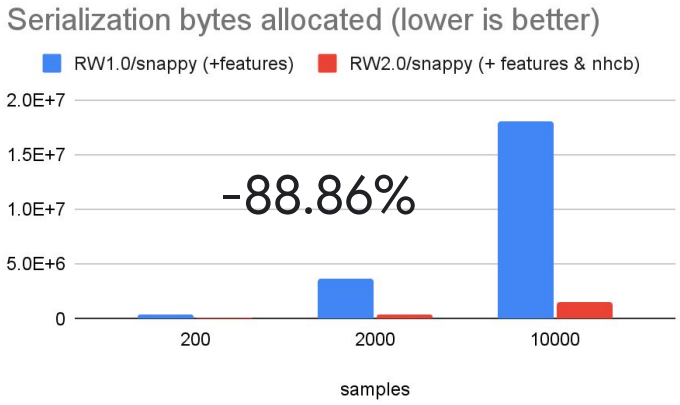
cstyan
bwplotka

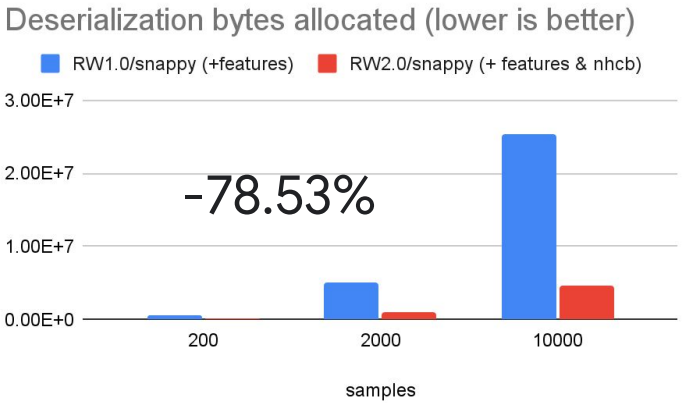
[1.0 + features] vs [2.0 + features & NHCB]











cstyan
bwplotka

How you can **use** or **adopt** it?

How to use it in Prometheus

- Enable feature flag to write Metadata records to WAL
  - `--enable-features=metadata-wal-records`
- Enable config for remote write:

```
remote_write:
  - url: http://remote1/push
    name: drop_expensive
    protobuf_message: io.prometheus.write.v2.Request
    write_relabel_configs:
      - source_labels: [__name__]
        regex: expensive.*
        action: drop
```

cstyan
bwplotka

How to adopt it

- Upgrade your client/receiver to the new proto, simple content negotiation and write statistics
  - *For Go: You can use our experimental client_golang client (and handler)*
- For testing and benchmarking you can use our sink.

Planned:

- For senders you're welcome to integrate with Prometheus our compliance test.
- For load generation you can use *avalanche* and *k6*.

cstyan
bwplotka

How to adopt it

Buf Schema Registry

Search ⌘ + K

bwplotka

buf.build / prometheus / prometheus

main

Docs | Code | SDKs | Commits | Deps

Aug 2, 2024

prometheus / io.prometheus.write.v2

Description

There is currently no description set for this package. More details about package documentation is available at our docs.

Index

Messages and enums

- { } BucketSpan
- { } Exemplar
- { } Histogram
- { } Metadata
- { } Request
- { } Sample
- { } TimeSeries

Messages and enums

{ } BucketSpan

io/prometheus/write/v2/types.proto

A BucketSpan defines a number of consecutive buckets with their offset. Logically, it would be more straightforward to include the bucket counts in the Span. However, the protobuf representation is more compact in the way the data is structured here (with all the buckets in a single array separate from the Spans).

```
message BucketSpan {
    sint32 offset = 1;
    uint32 length = 2;
}
```

{ } Exemplar

io/prometheus/write/v2/types.proto

Exemplar is an additional information attached to some series' samples. It is typically used to attach an example trace or request ID associated with the metric changes.

```
message Exemplar {
    // labels_refs is an optional list of label name-value pair references, encoded
    // as indices to the Request.symbols array. This list's len is always
    // a multiple of 2, and the underlying labels should be sorted lexicographically.
    // If the exemplar references a trace it should use the `trace_id` label name, as a best practice.
    repeated uint32 labels_refs = 1;
    // value represents an exact example value. This can be useful when the exemplar
    // is attached to a histogram, which only gives an estimated value through buckets.
    double value = 2;
    // timestamp represents the timestamp of the exemplar in ms.
    //
    // For Go, see github.com/prometheus/prometheus/model/timestamp/timestamp.go
    // for conversion from/to time.Time to Prometheus timestamp.
    int64 timestamp = 3;
}
```

cstyan
bwplotka

https://buf.build/prometheus/prometheus/docs/main:io.prometheus.write.v2

How to adopt it

```yaml
# buf.gen.yaml
version: v2

plugins:
- remote: buf.build/protocolbuffers/go:v1.31.0
  out: .
  opt:
  - Mio/prometheus/write/v2/types.proto=prometheus  /v2
  - Mtypes.proto=prometheus/v1
  - Mremote.proto=prometheus/v1

# vtproto for efficiency utilities like pooling etc.
# https://buf.build/community/planetscale-vtprotobuf?version=v0.6.0
- remote: buf.build/community/planetscale-vtprotobuf:v0.6.0
  out: .
  opt:
  - Mio/prometheus/write/v2/types.proto=prometheus/v2
  - Mtypes.proto=prometheus/v1
  - Mremote.proto=prometheus/v1
  - features=marshal+unmarshal+size+clone

inputs:
- module: buf.build/prometheus/prometheus:5b212ab78fb7460e831cf7ff2d83e385
  types:
  - "io.prometheus.write.v2.Request"
  - "prometheus.WriteRequest"
```

https://buf.build/prometheus/prometheus/docs/main:io.prometheus.write.v2

cstyan
bwplotka

Summary

Key takeaways

- 2.0 enables new features while maintaining efficiency aspect.
- Specify `io.prometheus.write.v2.Request` to use with Prometheus
- If you are owner of 3P project/tool – upgrade, test compatibility, and enjoy!
- Help us evolve remote storage story!
  - Give us feedback about 2.0.
  - Help us with Prometheus support of advanced 2.0 features.
  - Let's innovate e.g. more transactionality/structure, more compressions.

cstyan
bwplotka

Callum Styan, Senior SWE at Grafana
Bartek Płotka, Senior SWE at Google

Thank You! *Questions?*

Links:

- Remote Write 2.0 Spec
- Remote Write 2.0 Proposal
- Micro Benchmarks

Kudos to (hidden heroes)

- Alex, Nico, Paschalis
- Juraj, Arthur
- Original 1.0 authors! (Tom, Julius, and others)

cstyan
bwplotka

TBD

cstyan
bwplotka

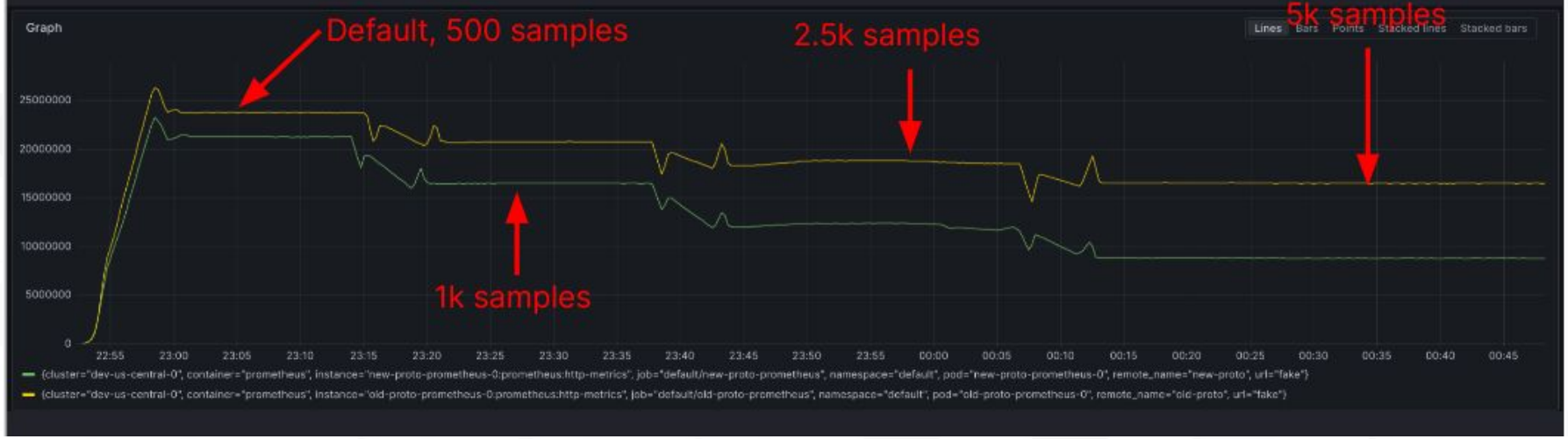
TBD

cstyan
bwplotka

Benchmarks

- 33% to 55% reduction in network egress used by Remote Write

- Yellow line is current prometheus remote write format, green is the new format



cstyan
bwplotka