Time Series in Prometheus

Fabian Reinartz – Engineer, SoundCloud Ltd.

prometheus.io

Prometheus

OVERVIEW DOCUMENTATION COMMUNITY BLOG GITHUB

Prometheus

An open-source service monitoring system and time series database.

Get Started

Data model

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

View details »

Query language

A flexible query language allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

View details »

Visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, a GUI-based dashboard builder, and a console template language.

View details »

Storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

View details »

Operation

Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

View details »

Client libraries

Client libraries allow easy instrumentation of services. Currently, Go, Java, and Ruby are supported. Custom libraries are easy to implement.

View details »

Alerting

Alerts are defined based on Prometheus's flexible query language and maintain dimensional information. An alertmanager handles notifications and silencing.

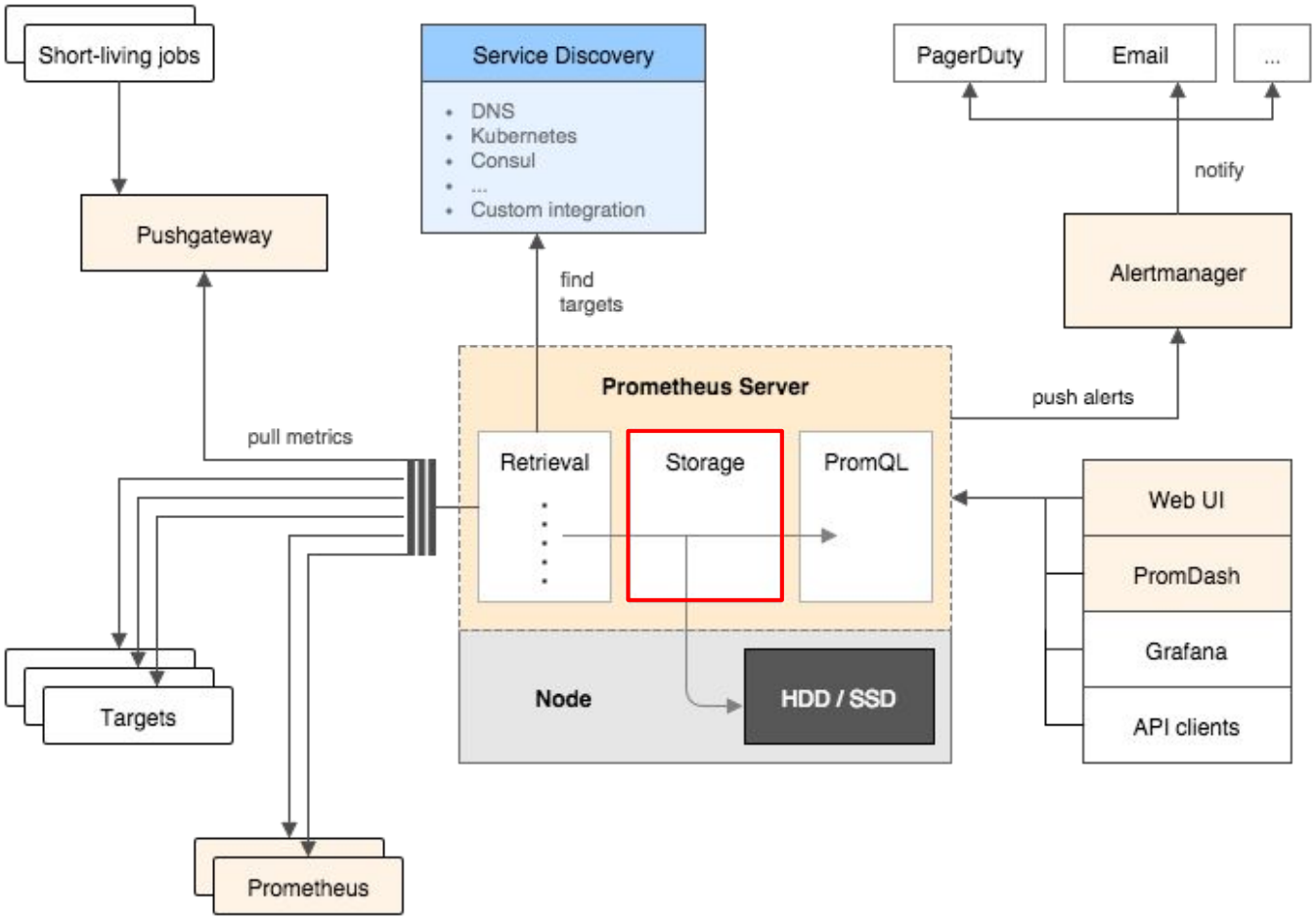
View details »

Exporters

Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

View details »

Brought to you by SOUNDCLOUD


Short-living jobs
Pushgateway
Service Discovery
DNS
Kubernetes
Consul
...
Custom integration
find
targets
PagerDuty
Email
...
notify
Alertmanager
Prometheus Server
Retrieval
Storage
PromQL
Node
HDD / SSD
pull metrics
push alerts
Web UI
PromDash
Grafana
API clients
Targets
Prometheus

Prometheus Metrics

Metric name	Labels	Timestamp	Sample Value
...			
http_requests_total	{status="200",method="GET"}	@1434317560938	94355
http_requests_total	{status="200",method="GET"}	@1434317561287	94934
http_requests_total	{status="200",method="GET"}	@1434317562344	96483
http_requests_total	{status="404",method="GET"}	@1434317560938	38473
http_requests_total	{status="404",method="GET"}	@1434317561249	38544
http_requests_total	{status="404",method="GET"}	@1434317562588	38663
http_requests_total	{status="200",method="POST"}	@1434317560885	4748
http_requests_total	{status="200",method="POST"}	@1434317561483	4795
http_requests_total	{status="200",method="POST"}	@1434317562589	4833
http_requests_total	{status="404",method="POST"}	@1434317560939	122
...			

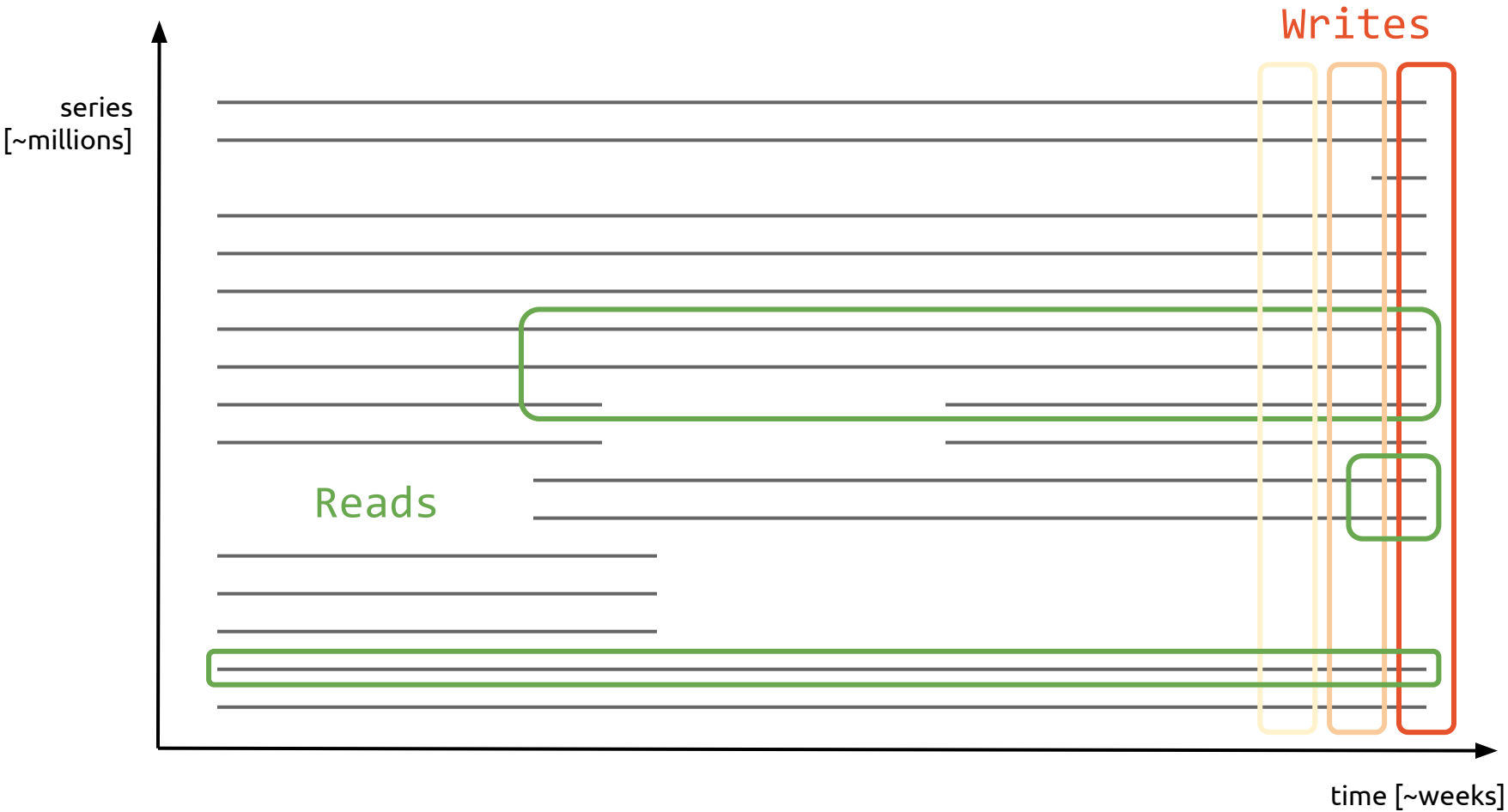
Requirements

- 1 million time series
- 10 second sample resolution
- 64bit timestamp + 64bit value

100,000 samples/sec

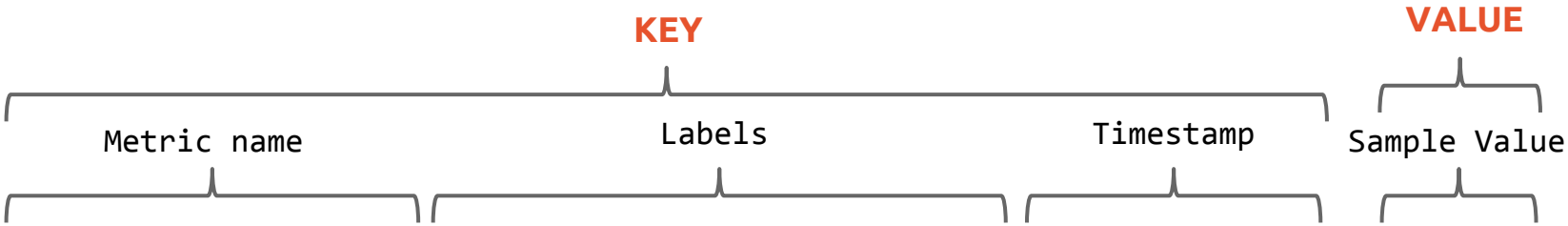
The Fundamental Problem

Orthogonal write and read patterns.

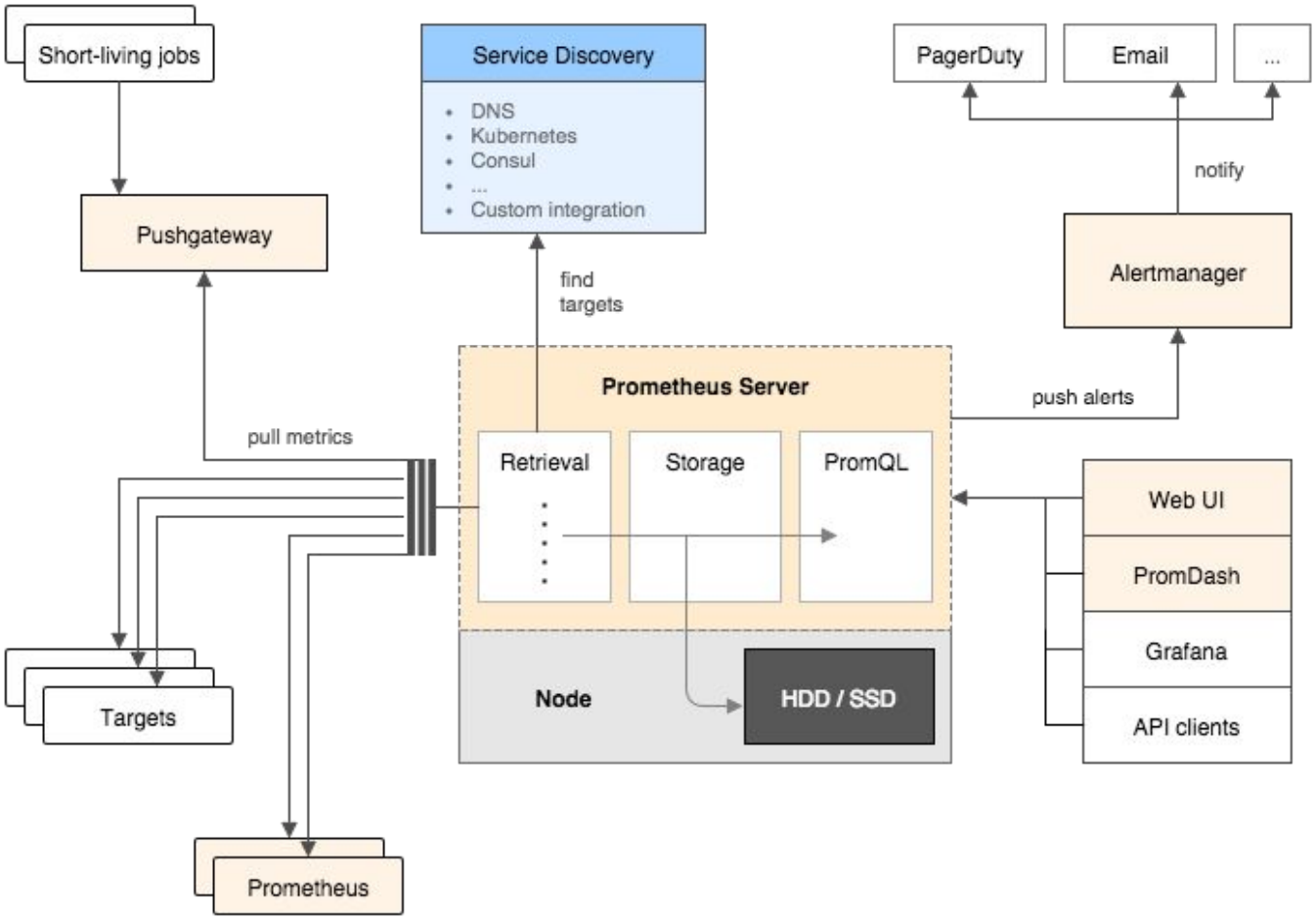


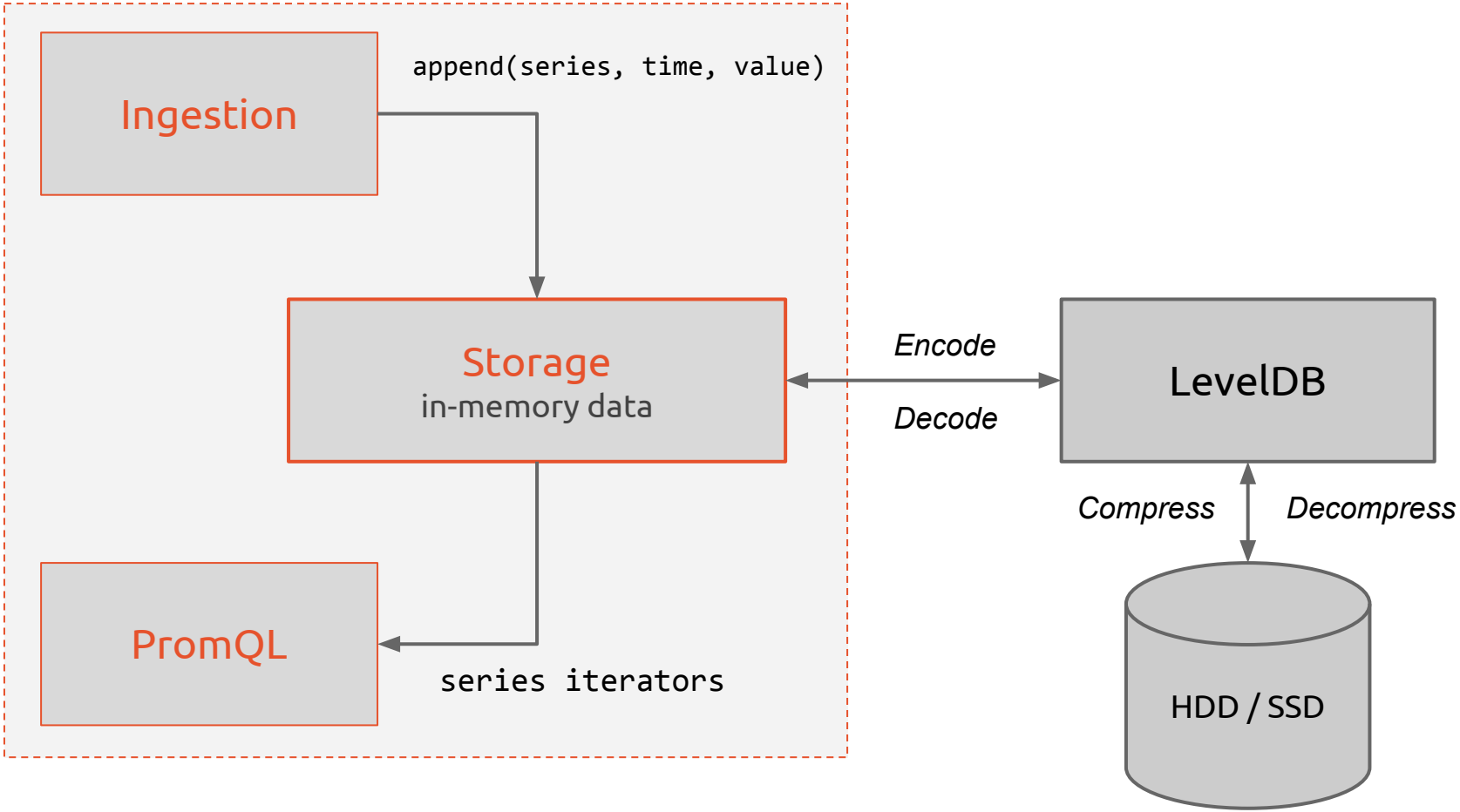
Prometheus Metrics

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




```
...
http_requests_total{status="200",method="GET"}  @1434317560938    94355
http_requests_total{status="200",method="GET"}  @1434317561287    94934
http_requests_total{status="200",method="GET"}  @1434317562344    96483
http_requests_total{status="404",method="GET"}  @1434317560938    38473
http_requests_total{status="404",method="GET"}  @1434317561249    38544
http_requests_total{status="404",method="GET"}  @1434317562588    38663
http_requests_total{status="200",method="POST"} @1434317560885    4748
http_requests_total{status="200",method="POST"} @1434317561483    4795
http_requests_total{status="200",method="POST"} @1434317562589    4833
http_requests_total{status="404",method="POST"} @1434317560939    122
...
```


Short-living jobs
Pushgateway
Service Discovery
DNS
Kubernetes
Consul
...
Custom integration
find
targets
PagerDuty
Email
...
notify
Alertmanager
Prometheus Server
Retrieval
Storage
PromQL
Node
HDD / SSD
pull metrics
push alerts
Web UI
PromDash
Grafana
API clients
Targets
Prometheus


Ingestion
append(series, time, value)
Storage
in-memory data
Encode
Decode
LevelDB
Compress
Decompress
HDD / SSD
PromQL
series iterators

Prometheus Metrics

Metric name

Labels

```
http_requests_total{status="200",method="GET"}
```

Labels

```
{__name__="http_requests_total",status="200",method="GET"}
```

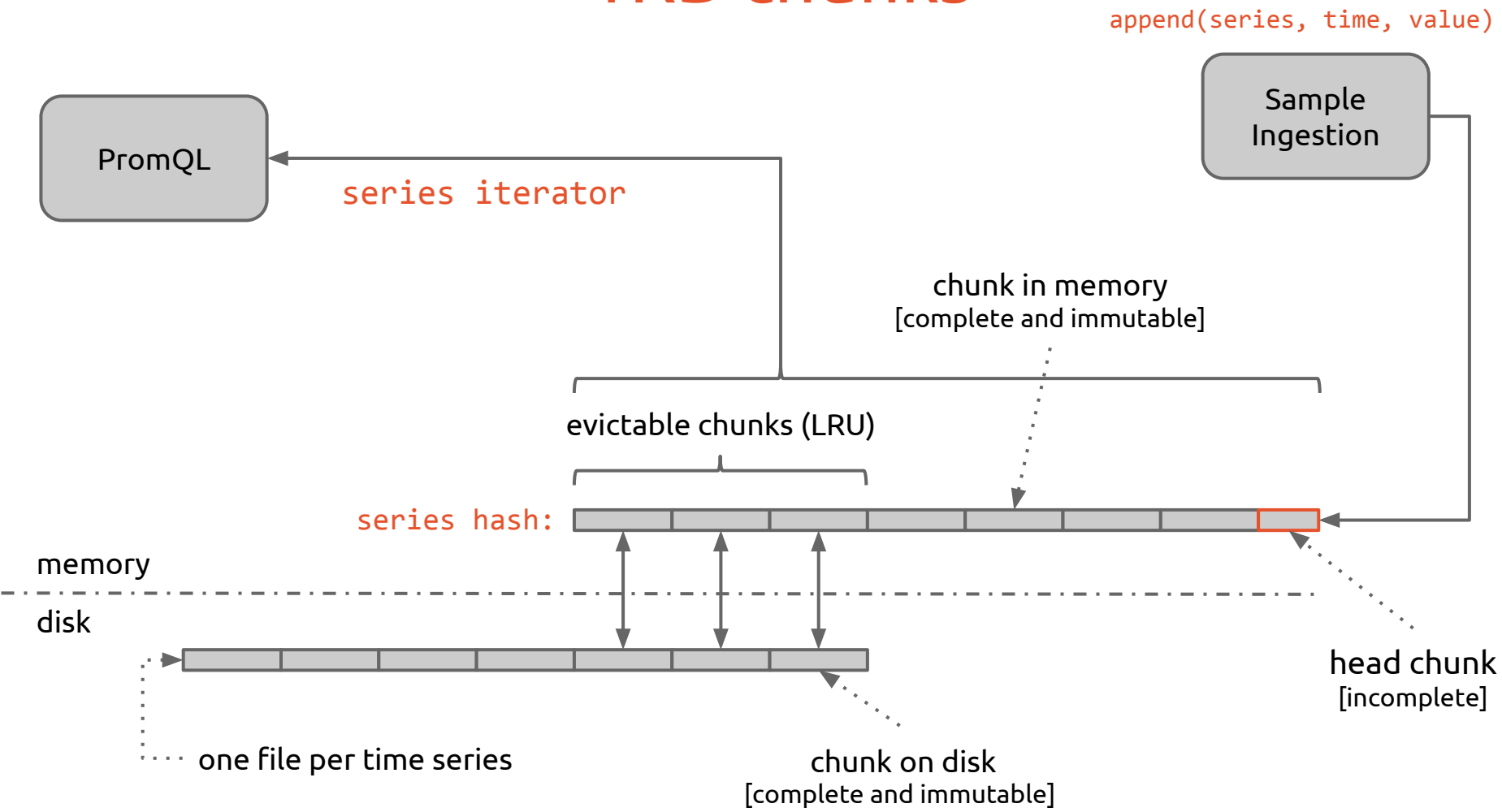
Prometheus Metrics

Learning the hard way

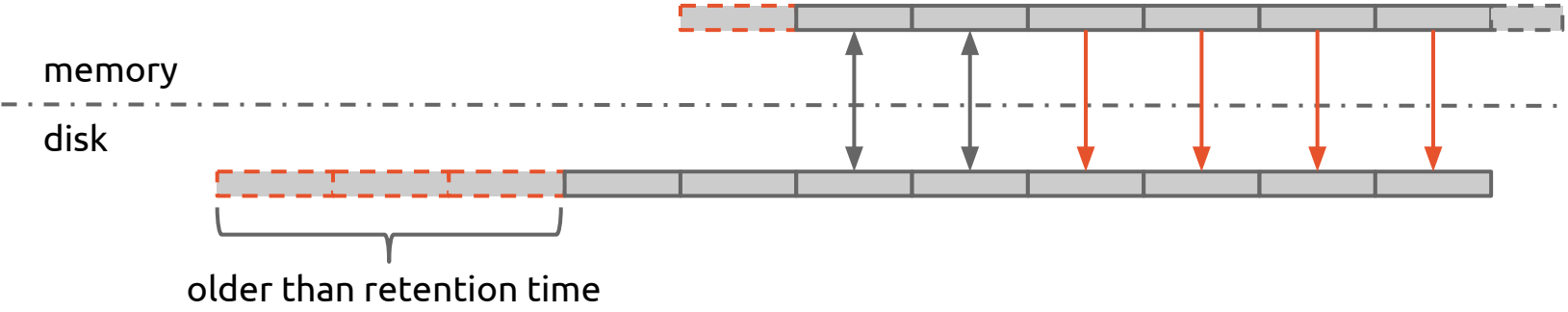
```
fnv(sort( __name__   = http_requests_total
          status     = 200
          method     = GET                  )
```

```
  fnv( __name__   = http_requests_total )
⊕ fnv( status     = 200 )
⊕ fnv( method     = GET )
```

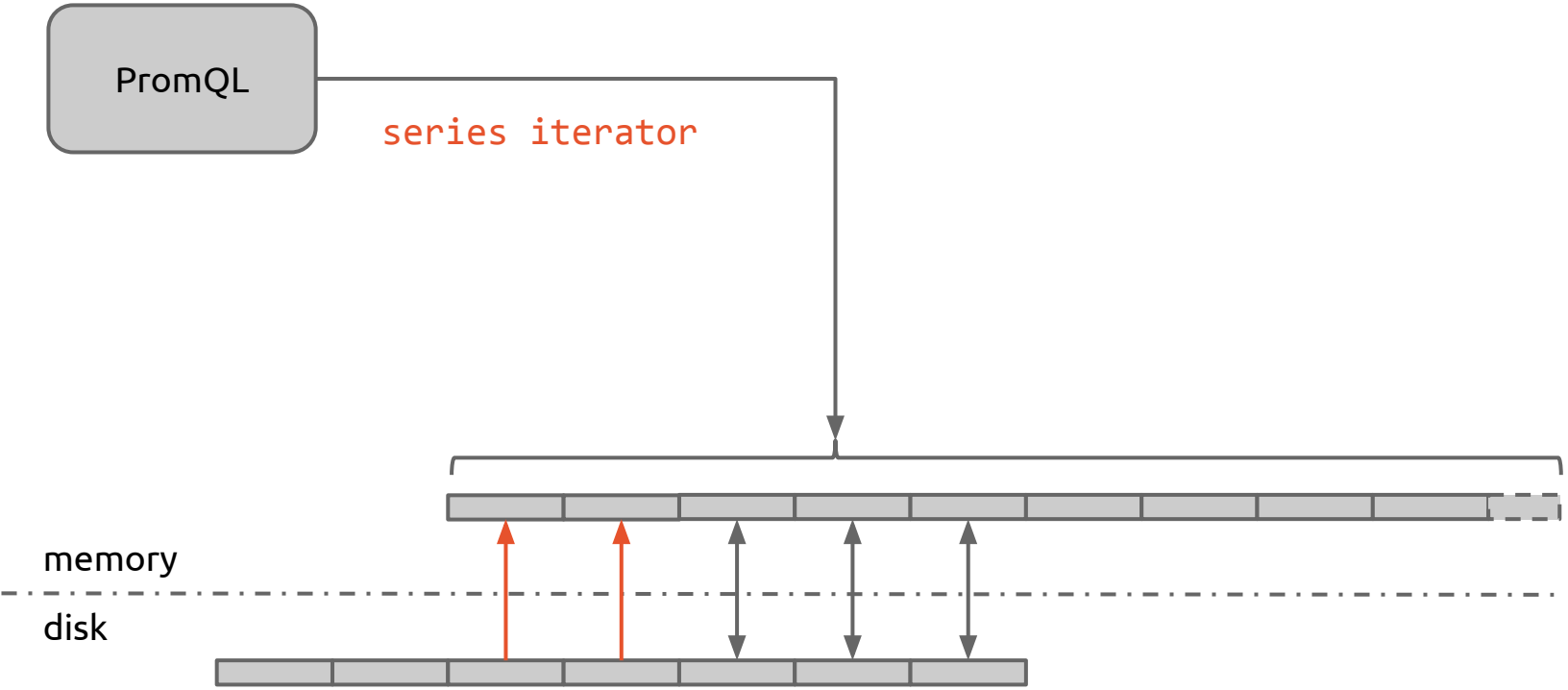
1KB chunks


append(series, time, value)
Sample Ingestion
PromQL
series iterator
chunk in memory
[complete and immutable]
evictable chunks (LRU)
series hash:
memory
disk
head chunk
[incomplete]
one file per time series
chunk on disk
[complete and immutable]

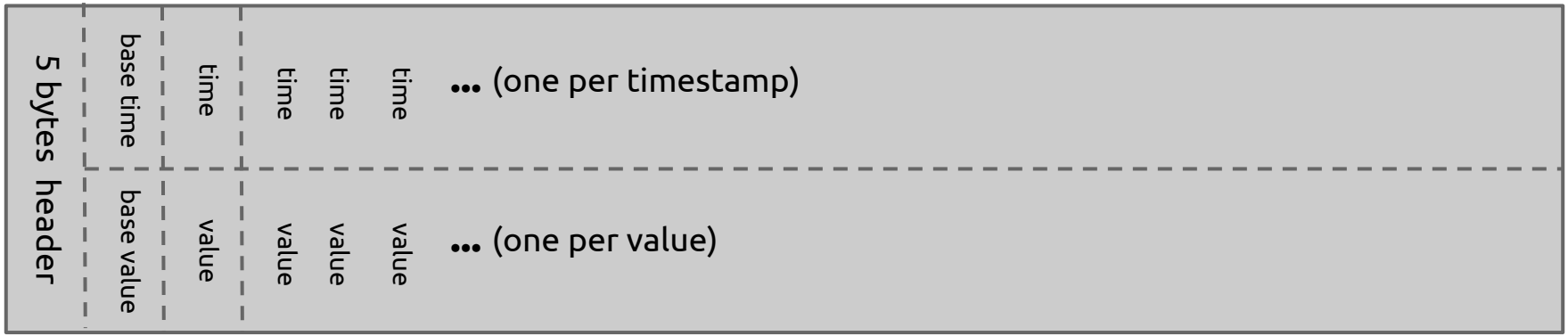
Series maintenance

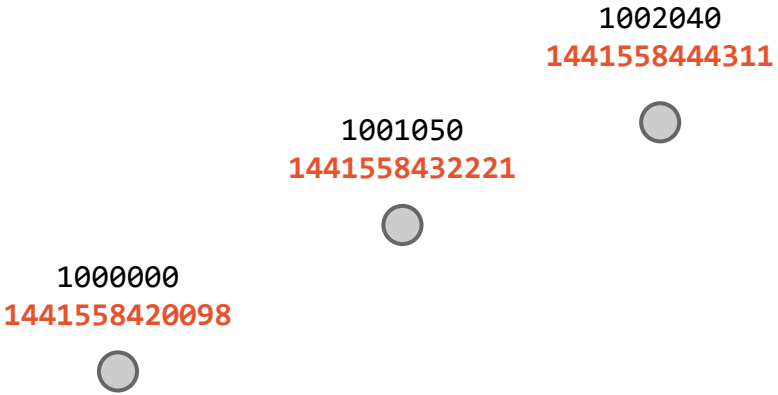


Chunk preloading



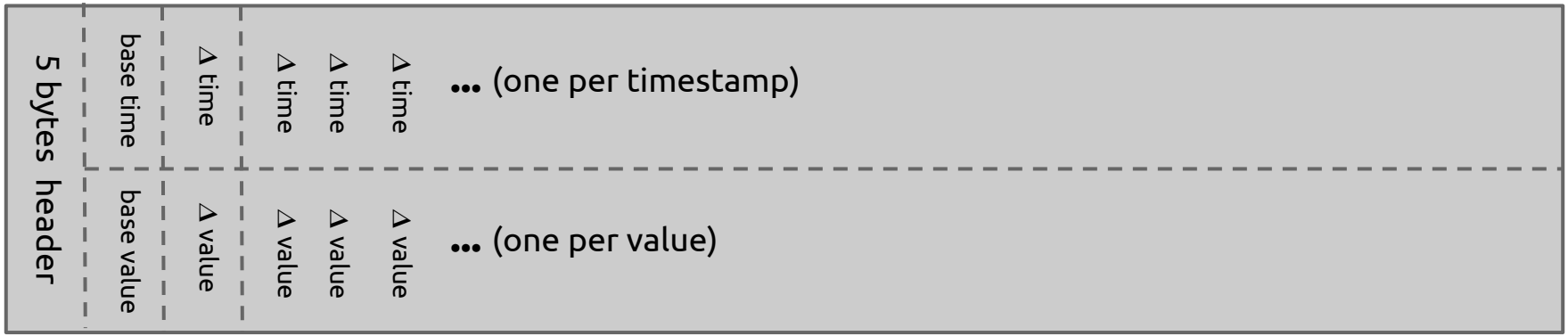
Anatomy of a chunk [v0]

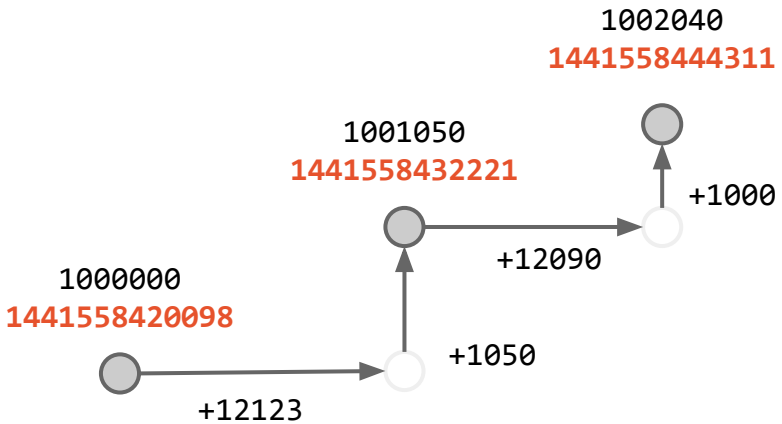




1000000	1441558420098
1001050	1441558432221
1002040	1441558444311

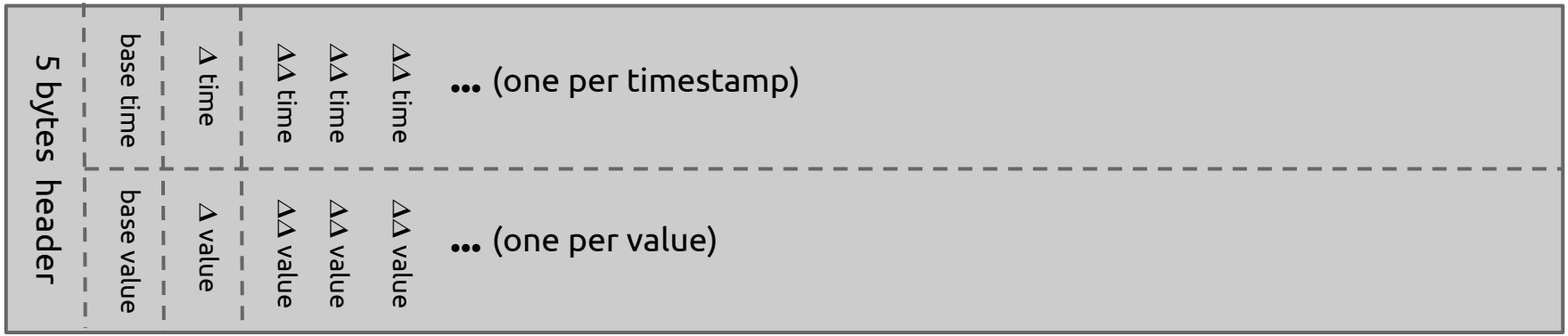
Anatomy of a chunk [v1]

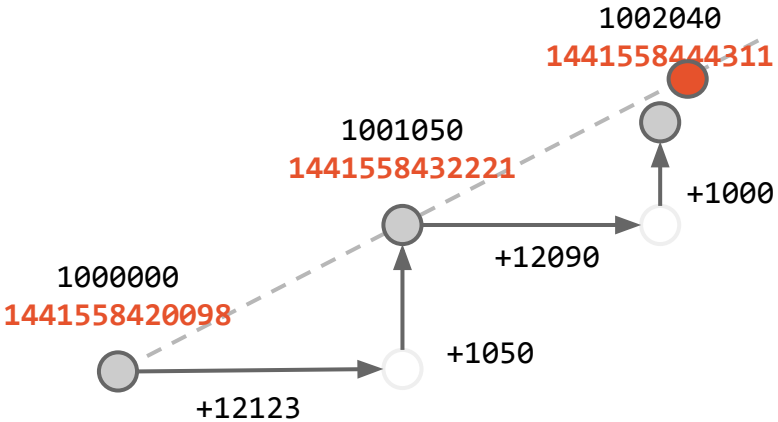




1000000	1441558420098
1050	12123
2040	24213
3100	36313
4250	48500

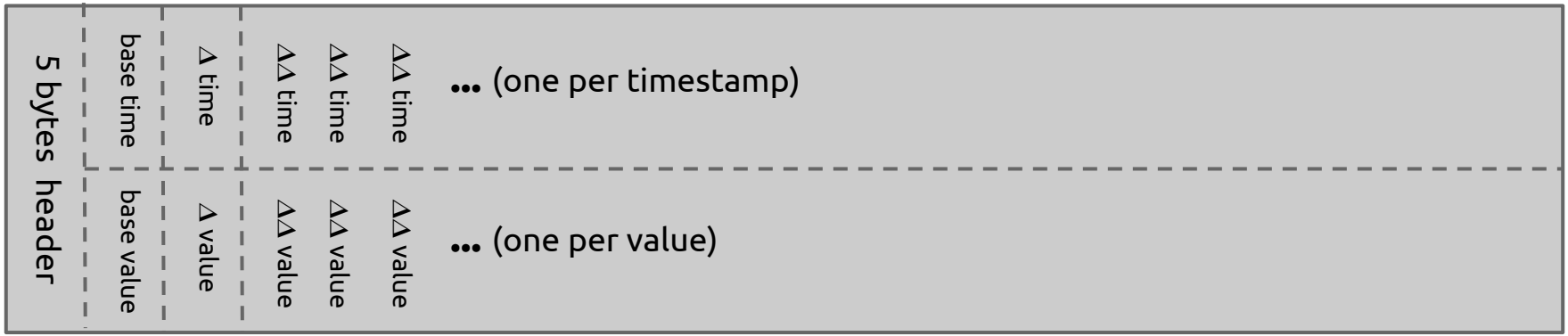
Anatomy of a chunk [v2]





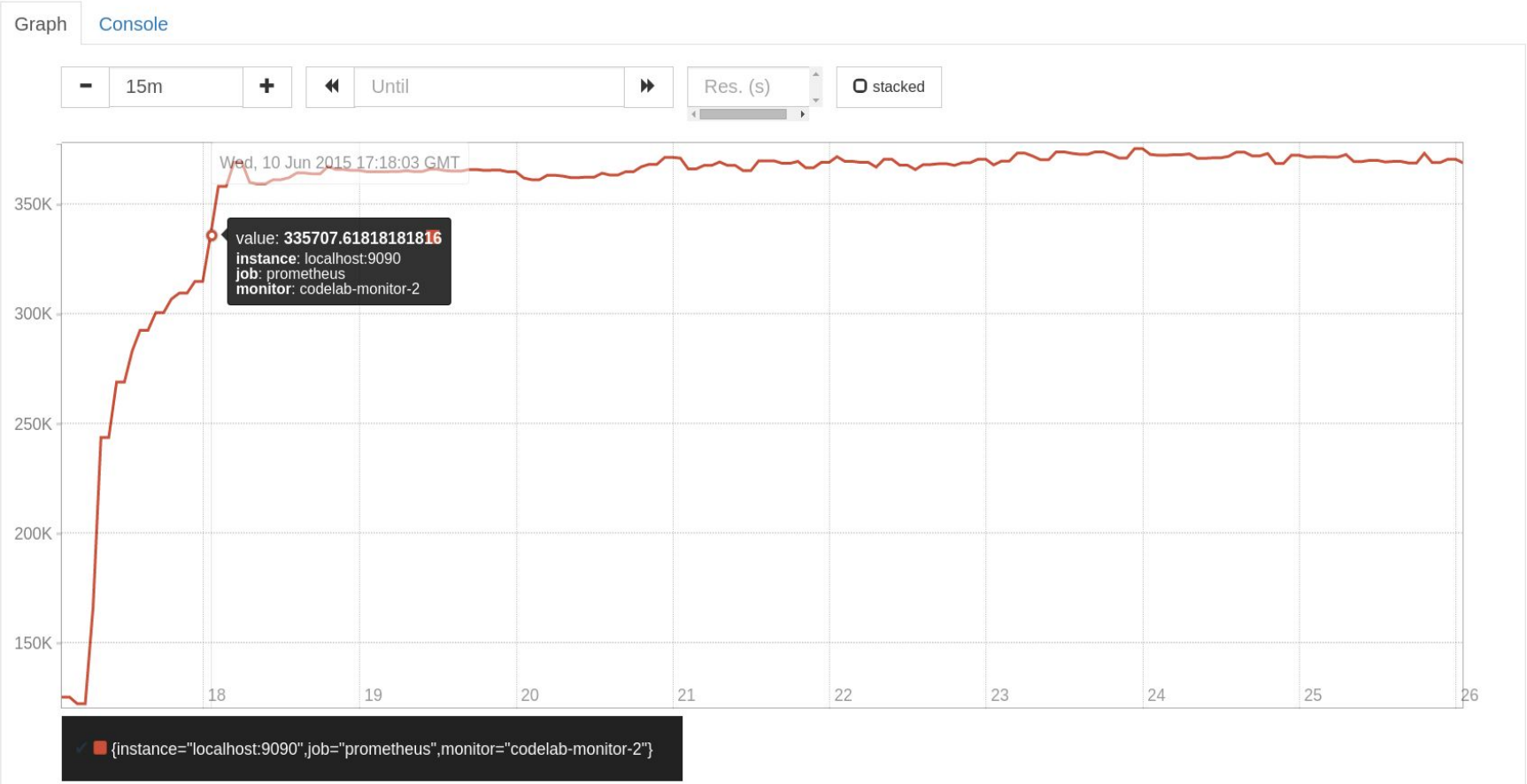
1000000	1441558420098
1050	12123
-60	-33
-50	-56
+50	-8

Anatomy of a chunk [v2]



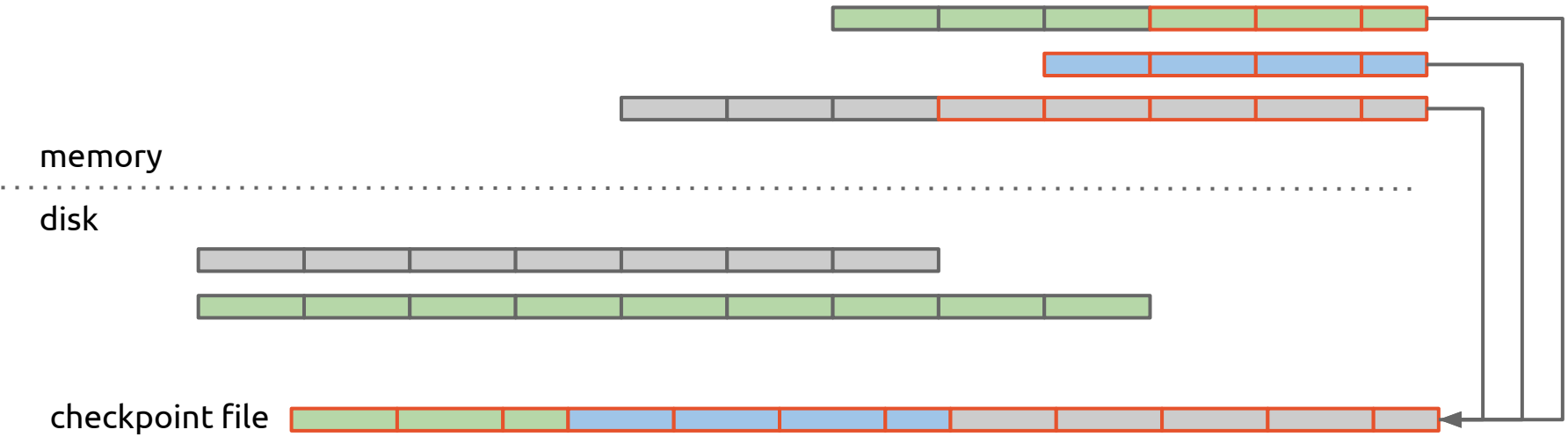

```
13:14 < nostrovsk> Hey guys, Looking for a sanity check here
13:15 < nostrovsk> 500 machines per server, each running node and jmx exporters, for 1 week is
                   only 30gb of data?
13:36 <@  bbrazil> what's your scrape rate and how heavy are those jmx exporters?
13:37 <@  bbrazil> doesn't sound implausible to me
13:42 <@  bbrazil> we're 25GB/two weeks with ~5k samples/s
13:45 <@   beorn7> Compression, it works... ;)
13:53 <     fish_> beorn7: nothing says better 'good job' than people coming to this channel
                   because they can't believe that things are soo good :)
```

```
rate(prometheus_local_storage_ingested_samples_total[1m])
```



Checkpointing

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



Prometheus Alerts Graph Status Help

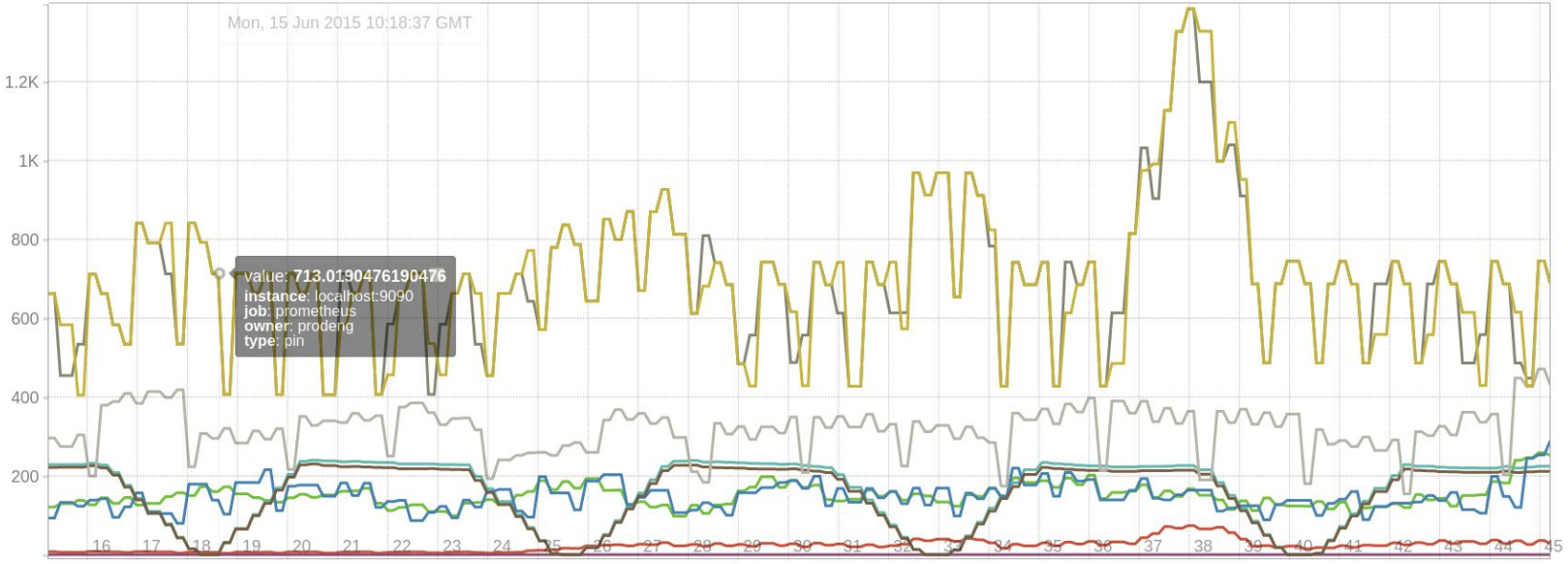
rate(prometheus_local_storage_chunk_ops_total[2m]) Execute

Load time: 145ms
Resolution: 7s

- Insert Metric at Cursor -

Graph Console

30m Until Res. (s) stacked



{instance="localhost:9090",job="prometheus",owner="prodeng",type="unpin"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="transcode"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="pin"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="persist"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="load"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="evict"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="drop"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="create"}
{instance="localhost:9090",job="prometheus",owner="prodeng",type="clone"}