Munich, 9th August 2018

Thanos

Global, durable Prometheus monitoring

Fabian Reinartz

fabxc

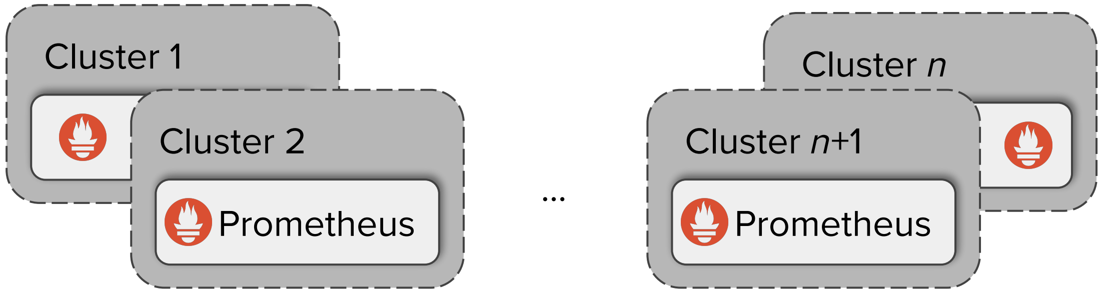
Bartek Plotka

Bwplotka Bplotka

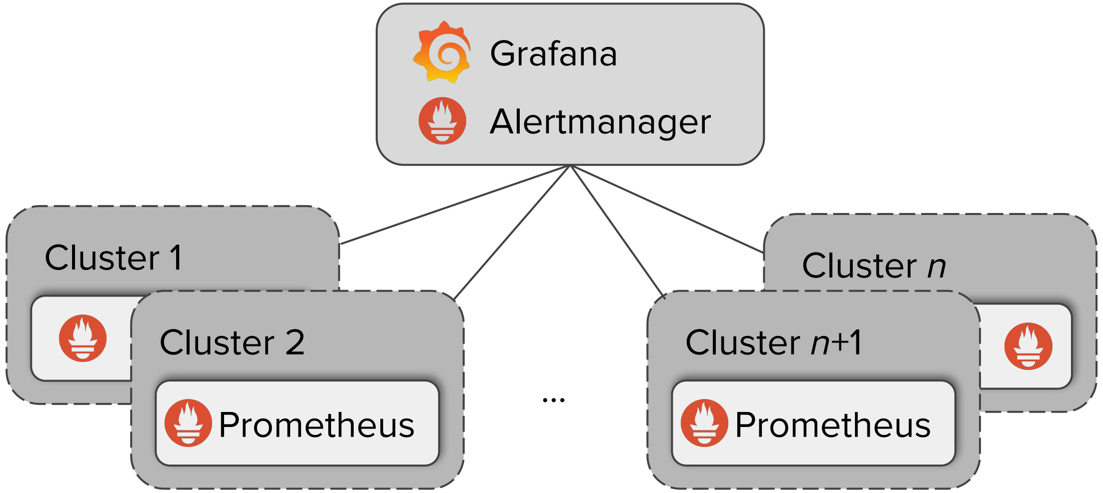
Prometheus 2.X

- Reliable operational model
- Powerful query language
- Scraping capabilities beyond the casual usage
- Local metric storage

Prometheus at Scale

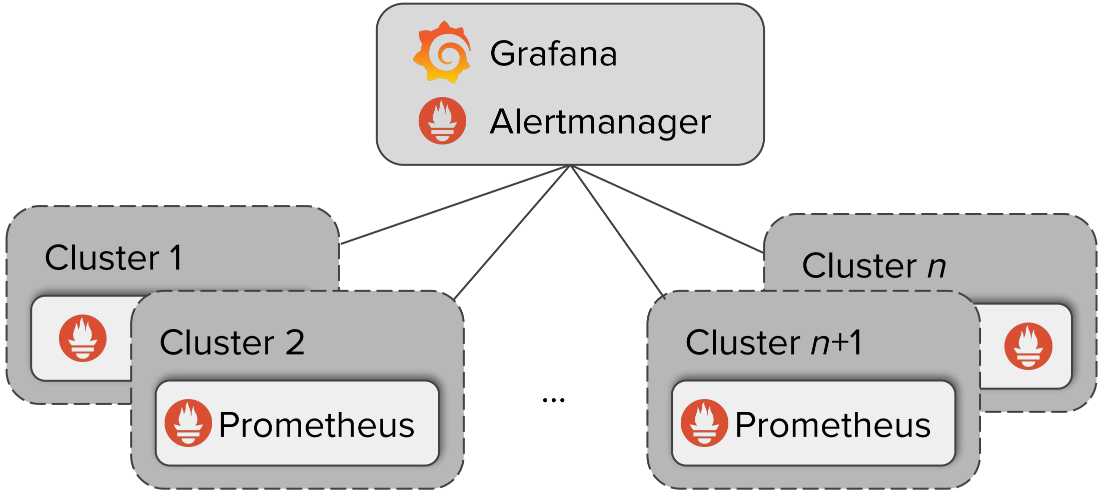


Problem: Global View



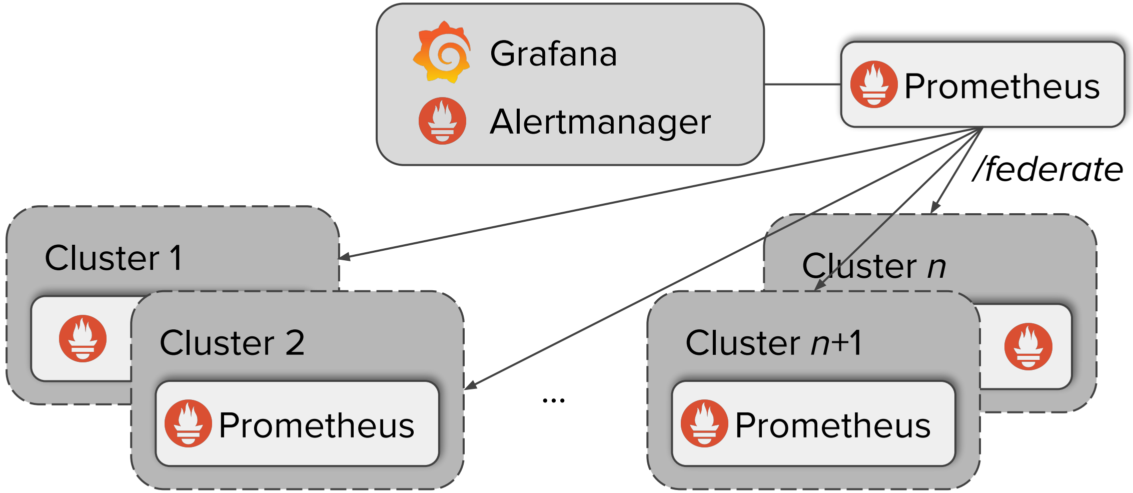
Problem: Global View

sum(rate(go_memstats_alloc_bytes_total[1m])) by (env, cluster, job) **?**

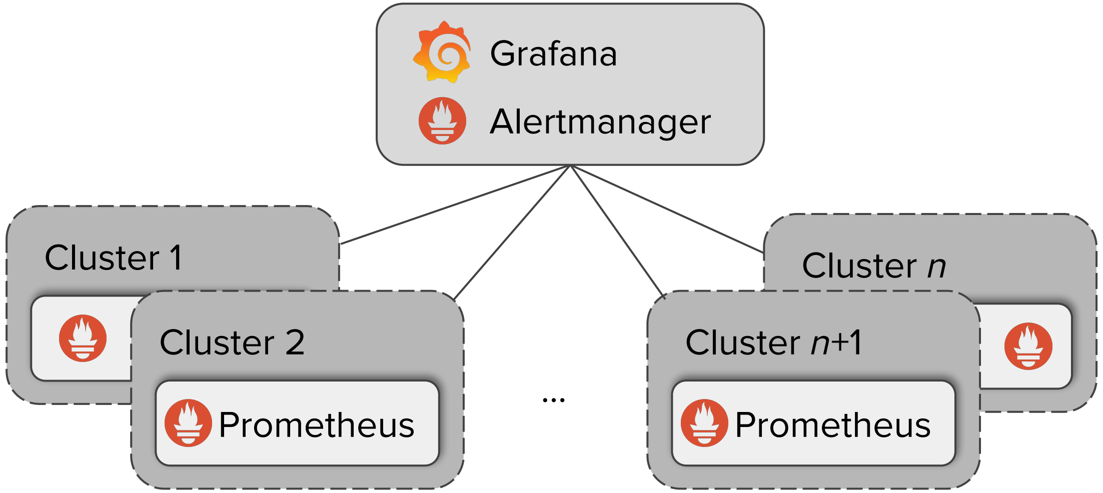


Problem: Global View

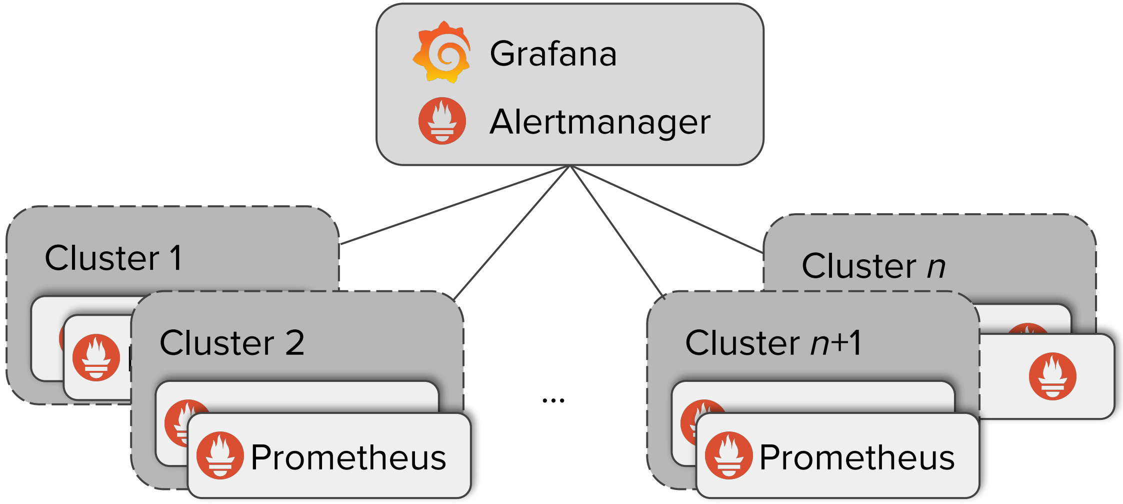
sum(go_memstats_alloc_bytes_total::rate1m) by (env, cluster, job) ✓



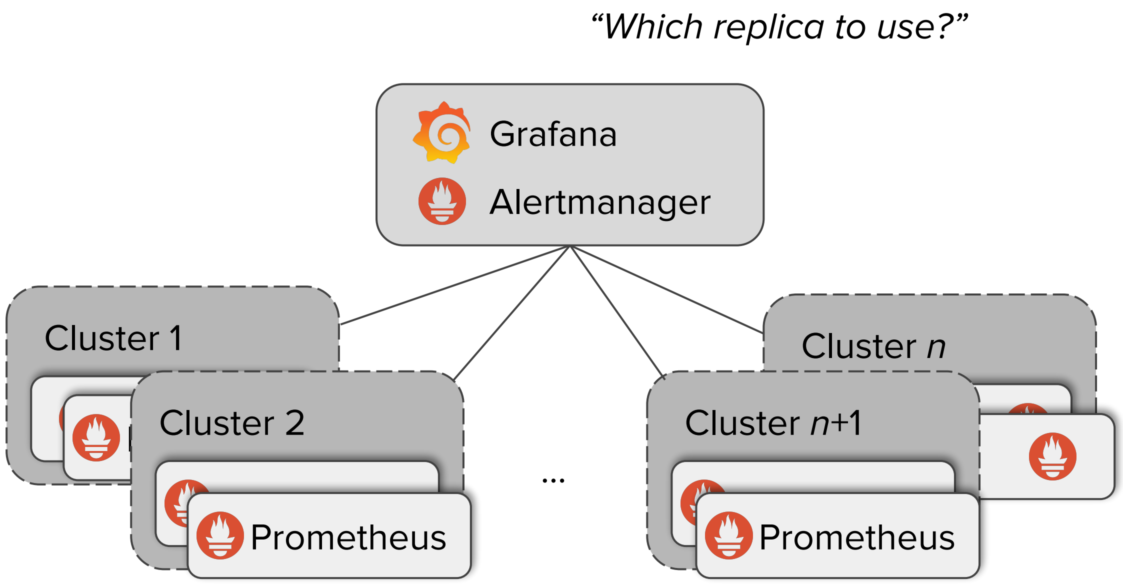
Problem: High Availability



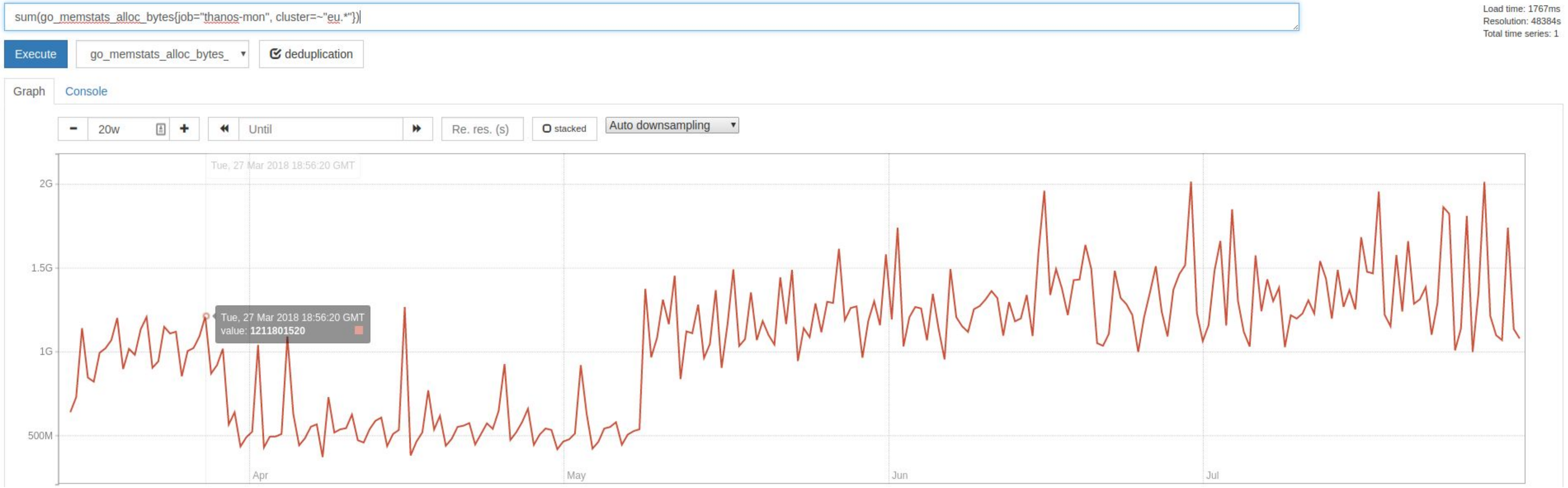
Problem: High Availability



Problem: High Availability



Problem: Metric retention

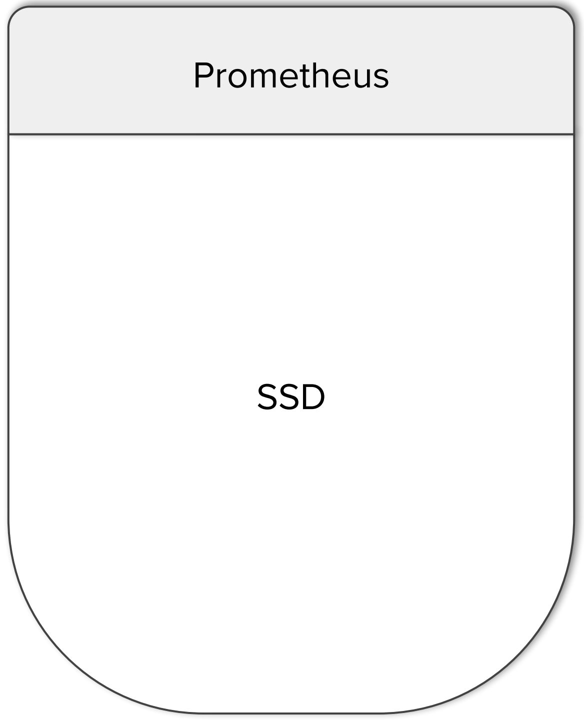


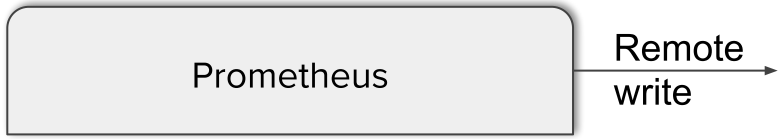
```
sum(go_memstats_alloc_bytes{job="thanos-mon", cluster=~"eu.*"})
```

Load time: 1767ms
Resolution: 48384s
Total time series: 1

Tue, 27 Mar 2018 18:56:20 GMT
value: **1211801520**

Problem: Metric retention

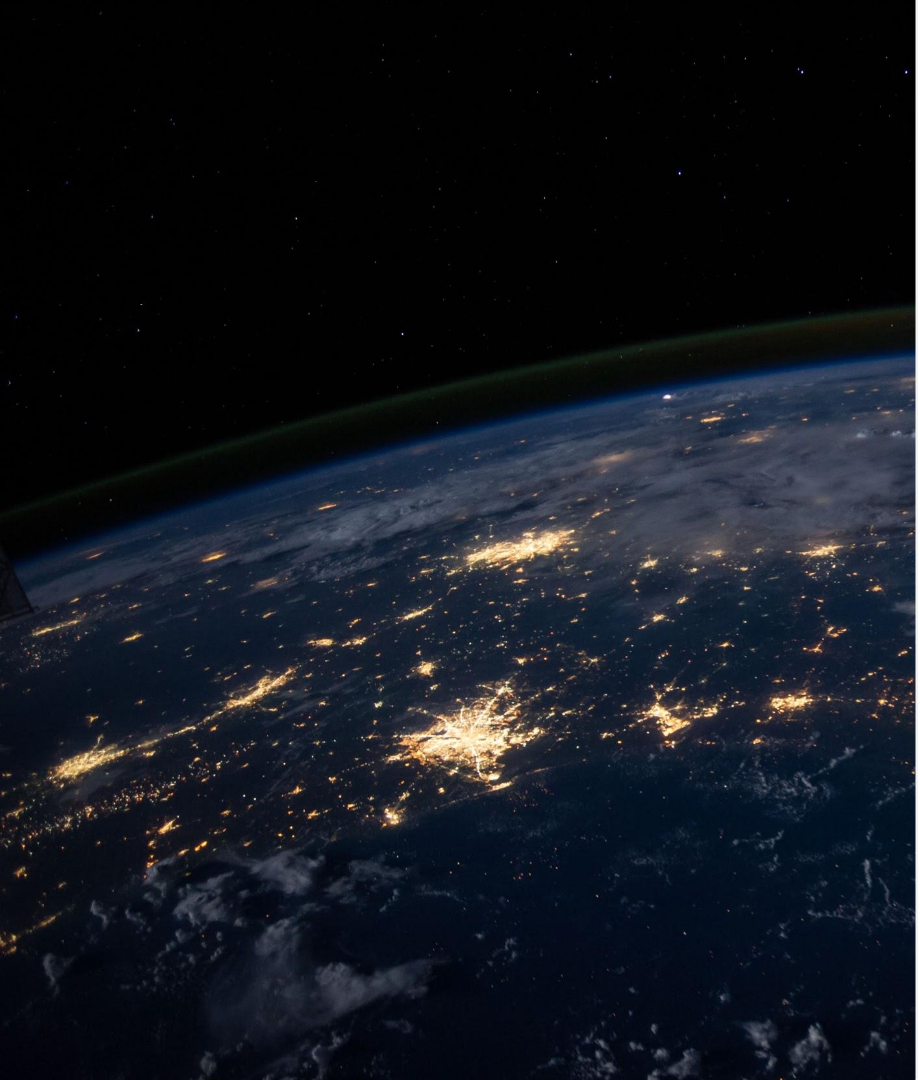




Thanos

Goals

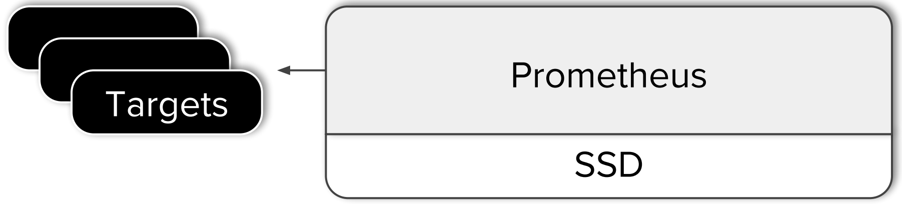
- Have a global view
- Have a HA in place
- Increase retention



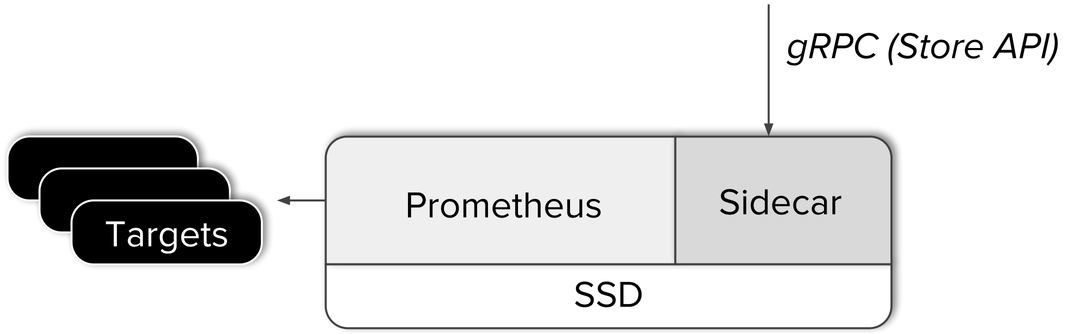
Global View

See everything from a single place!

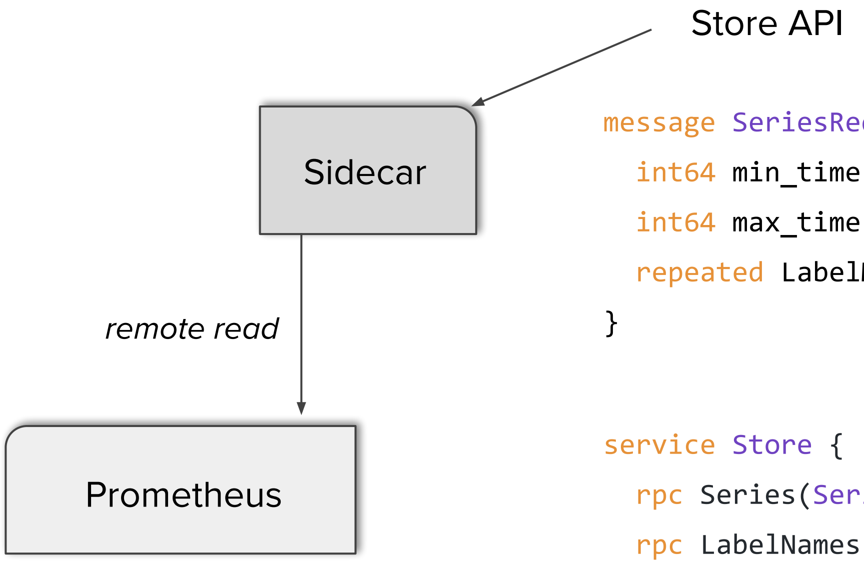
Prometheus



Sidecar



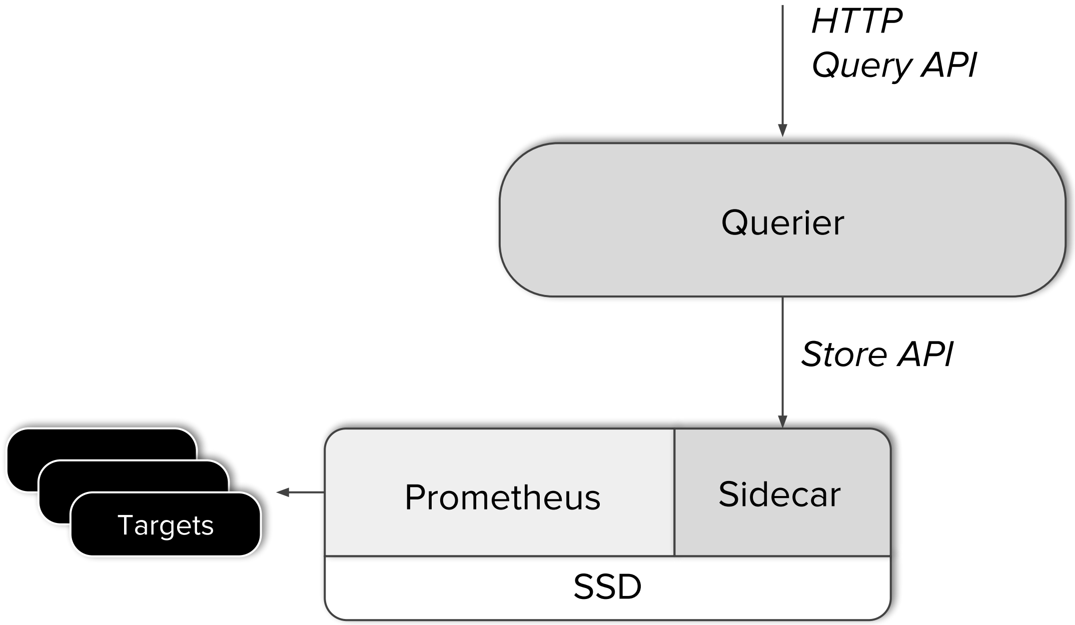
Store API



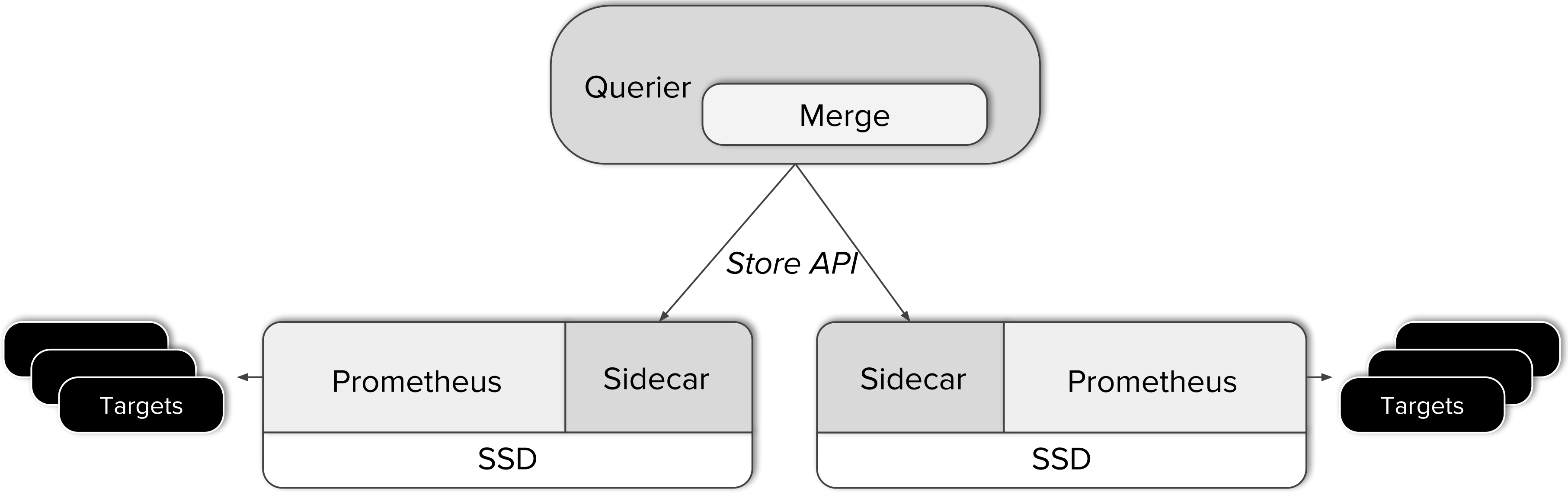
```
message SeriesRequest {
  int64 min_time = 1;
  int64 max_time = 2;
  repeated LabelMatcher matchers = 3;
}


service Store {
  rpc Series(SeriesRequest) returns (stream SeriesResponse);
  rpc LabelNames(LabelNamesRequest) returns (LabelNamesResponse);
  rpc LabelValues(LabelValuesRequest) returns (LabelValuesResponse);
}
```

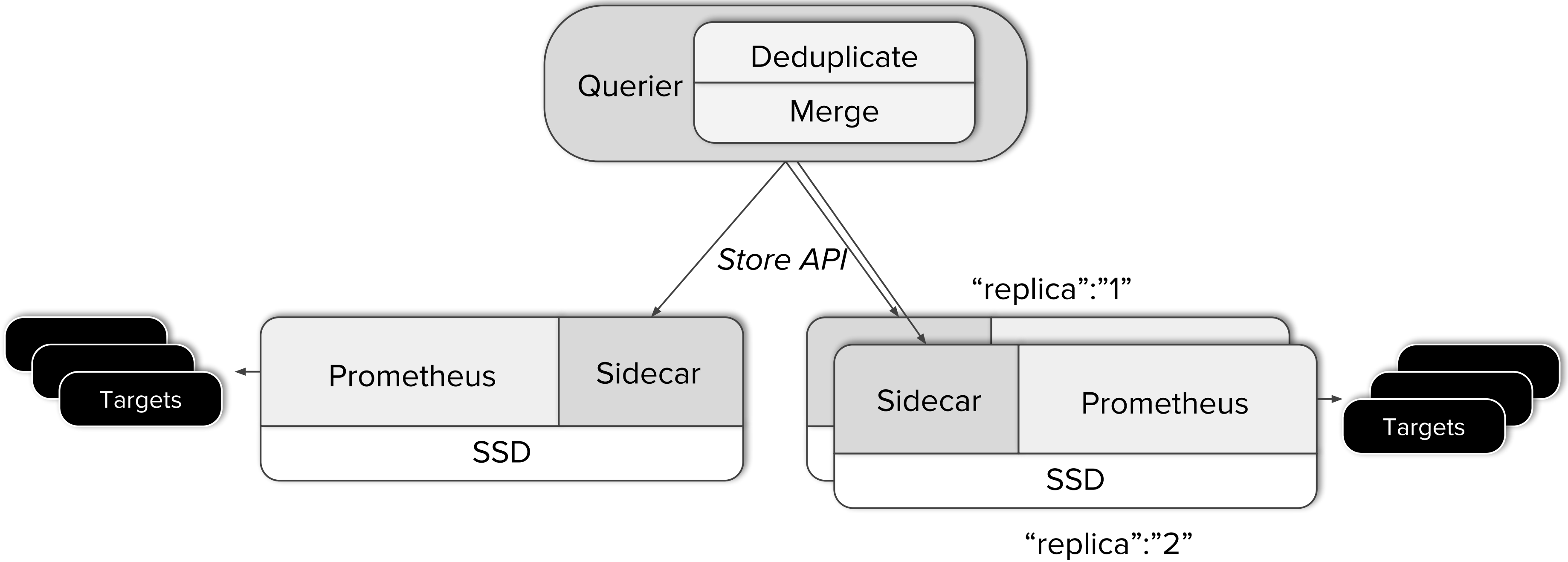
Querier


HTTP
Query API
Querier
Store API
Prometheus
Sidecar
SSD
Targets

Global View


Querier
Merge
Store API
Targets
Prometheus
Sidecar
SSD
Sidecar
Prometheus
SSD
Targets

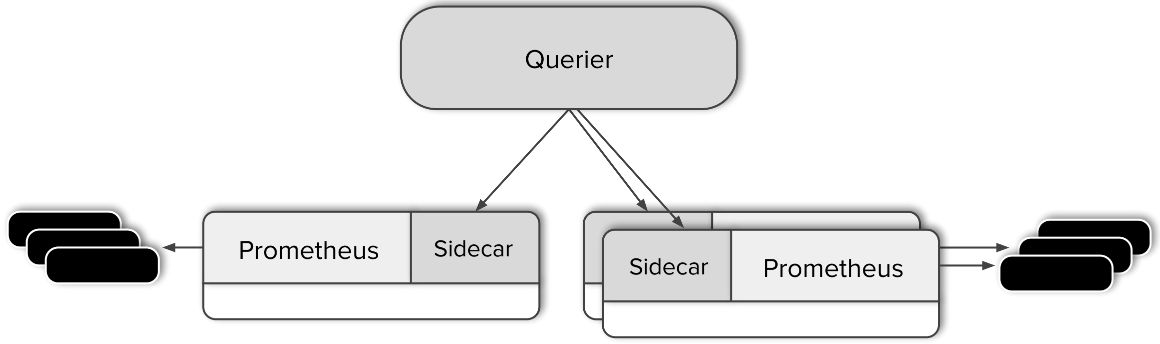
Global View + Availability



Thanos

Goals

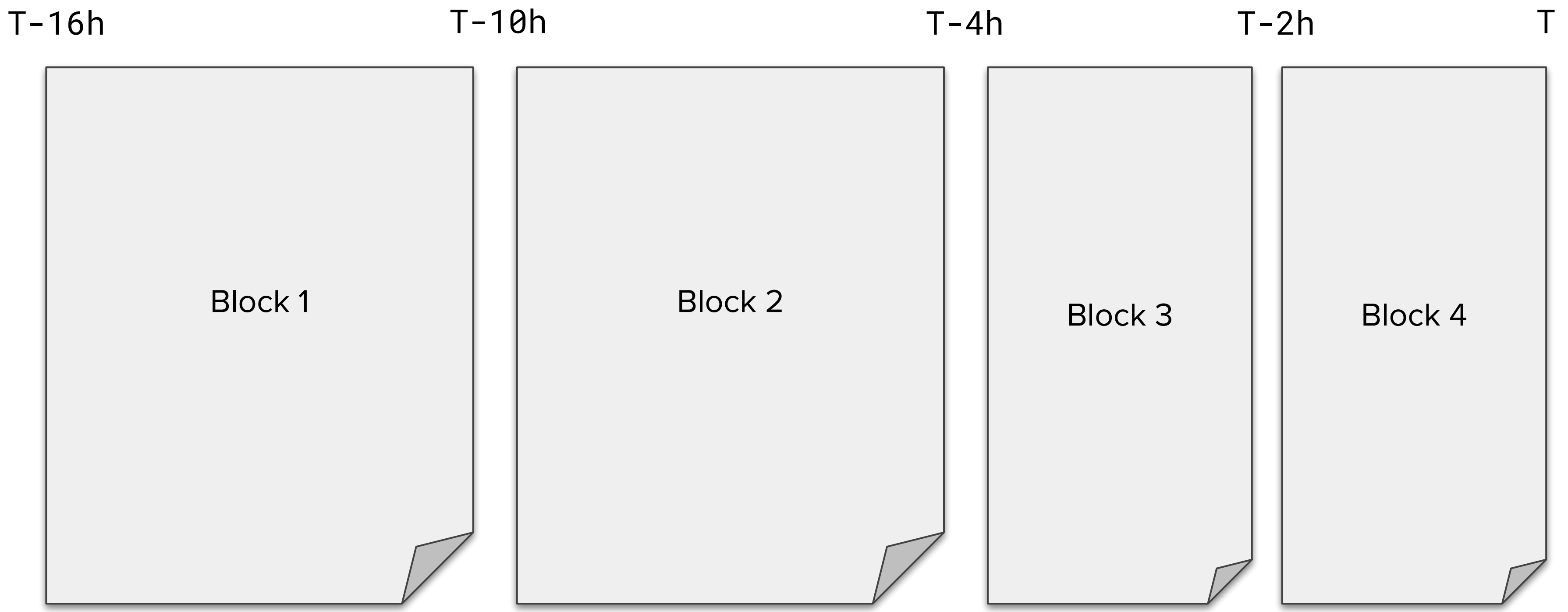
- Have a global view ✓
- Have a HA in place ✓



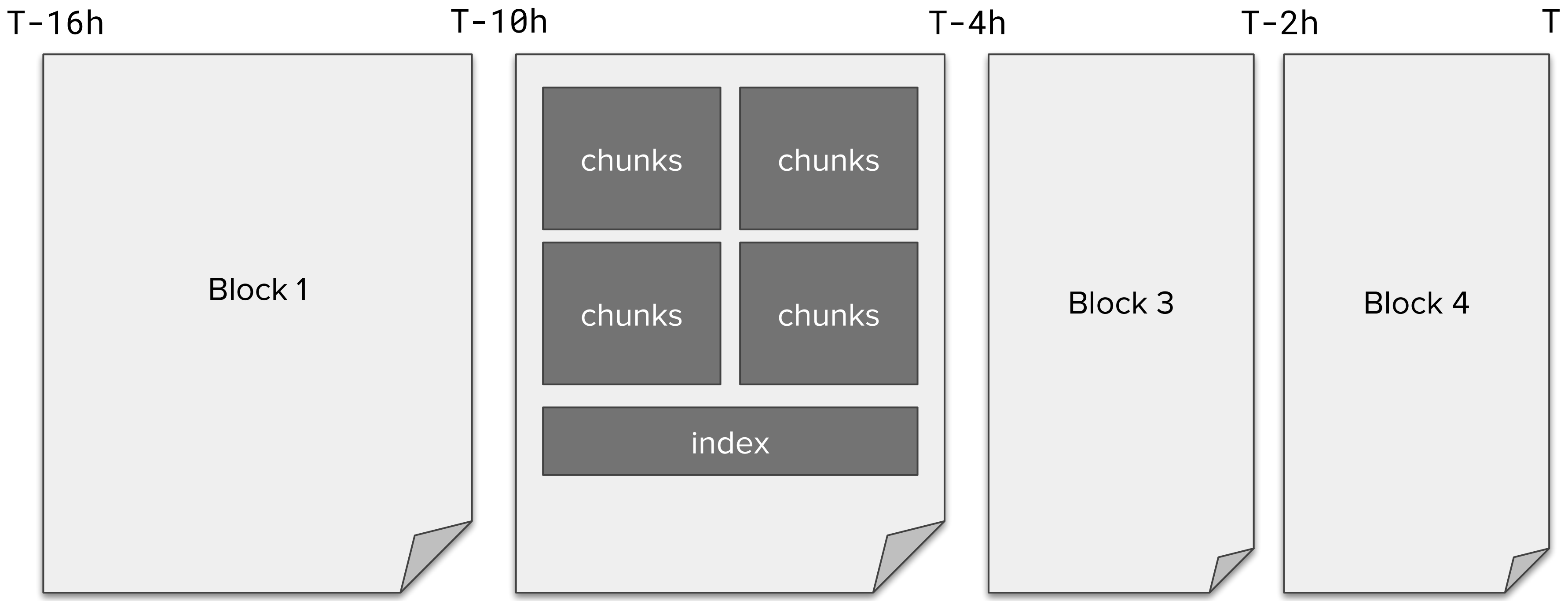
Historical Metrics

What exactly happened X months ago?

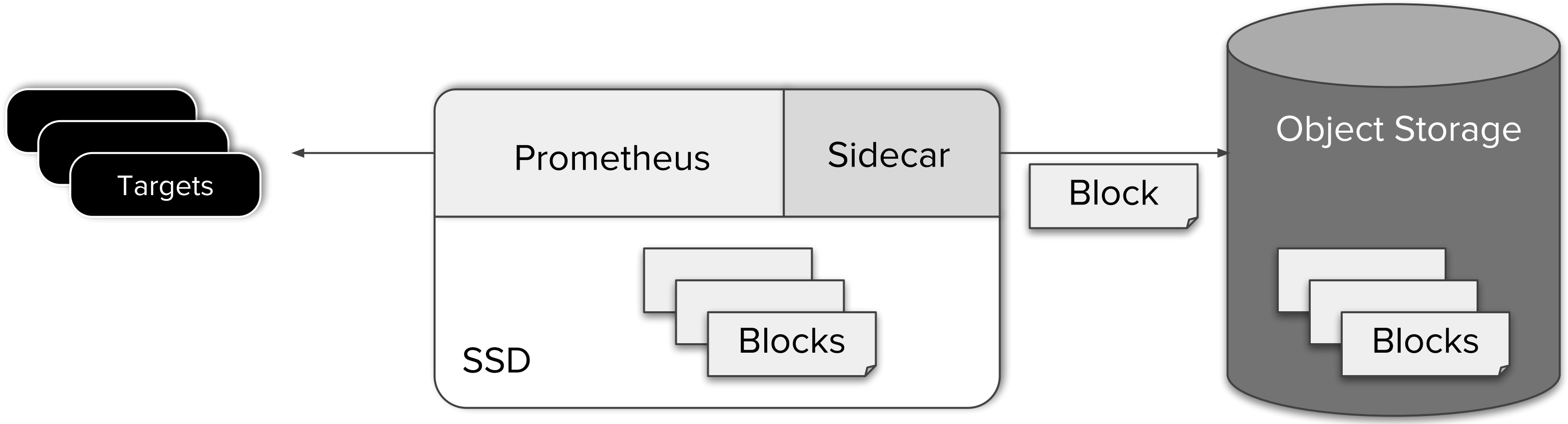
TSDB Layout



TSDB Layout

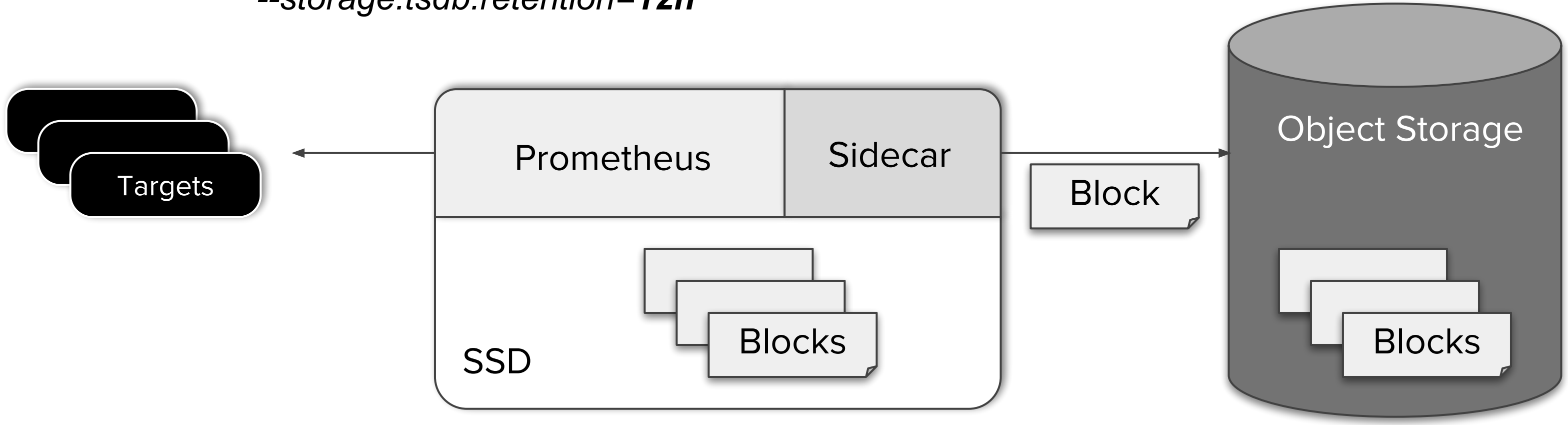

T-16h
T-10h
T-4h
T-2h
T
Block 1
chunks
chunks
chunks
chunks
index
Block 3
Block 4

Data saving

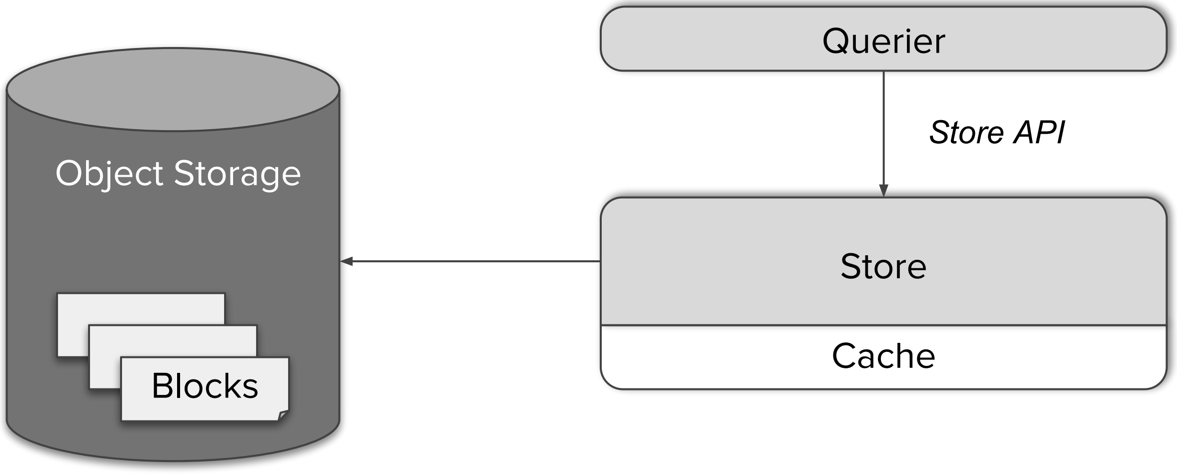

Targets
Prometheus
Sidecar
SSD
Blocks
Block
Object Storage
Blocks

Data saving

*--storage.tsdb.max-block-duration=**2h***
*--storage.tsdb.retention=**12h***



Store Gateway

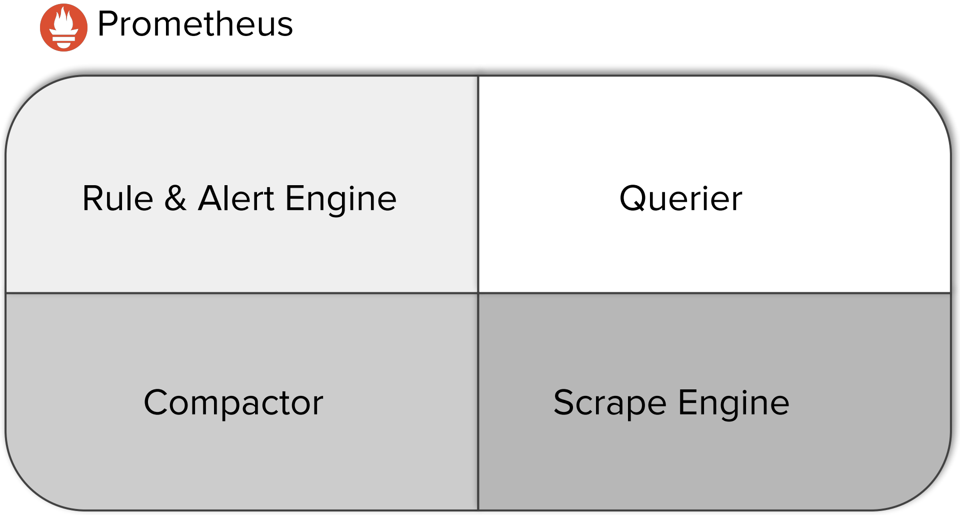


Thanos

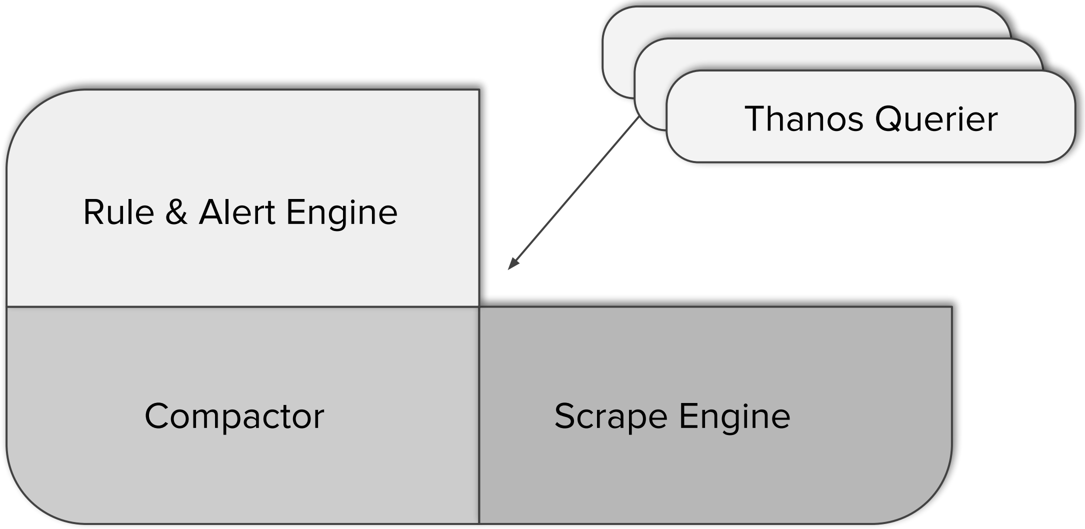
Goals

- Have a global view ✓
- Have a HA in place ✓
- Increase retention ✓

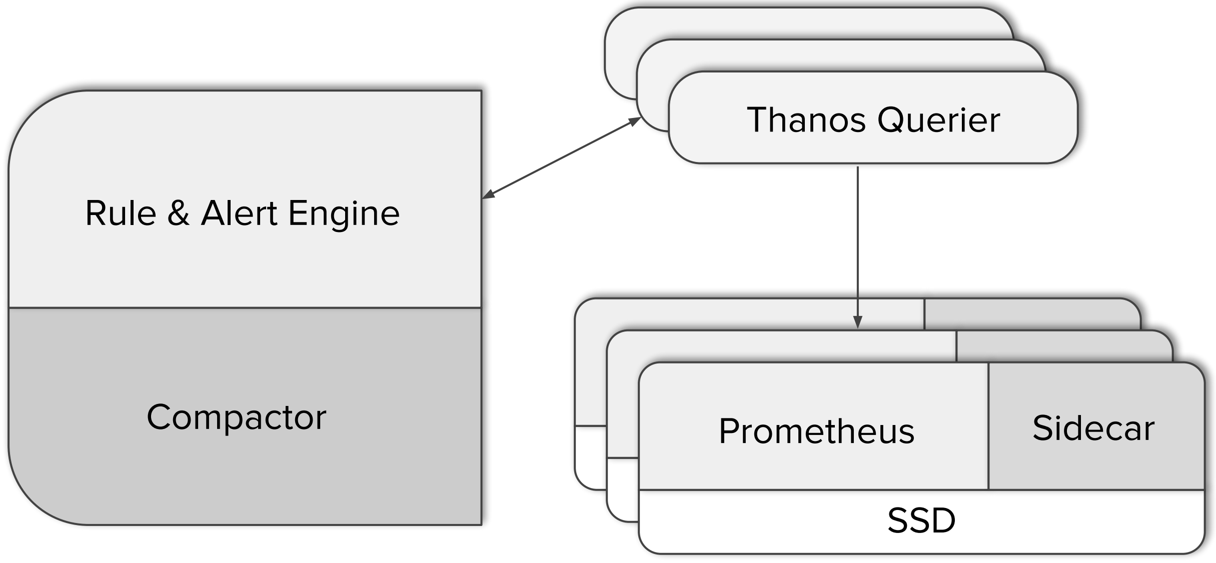
Prometheus


Prometheus
Rule & Alert Engine
Querier
Compactor
Scrape Engine

Thanos

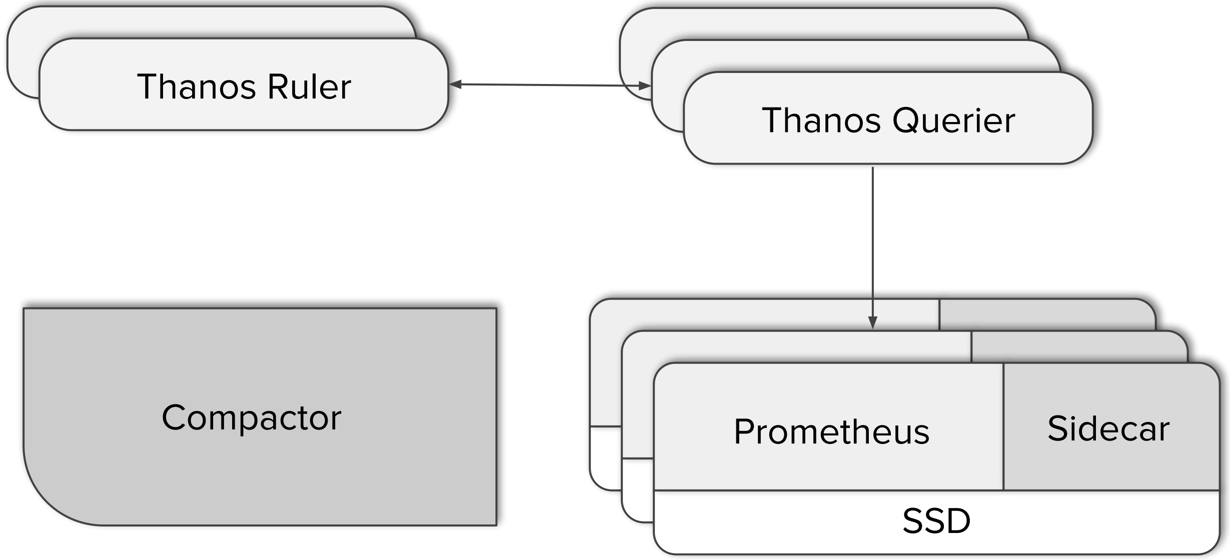


Thanos

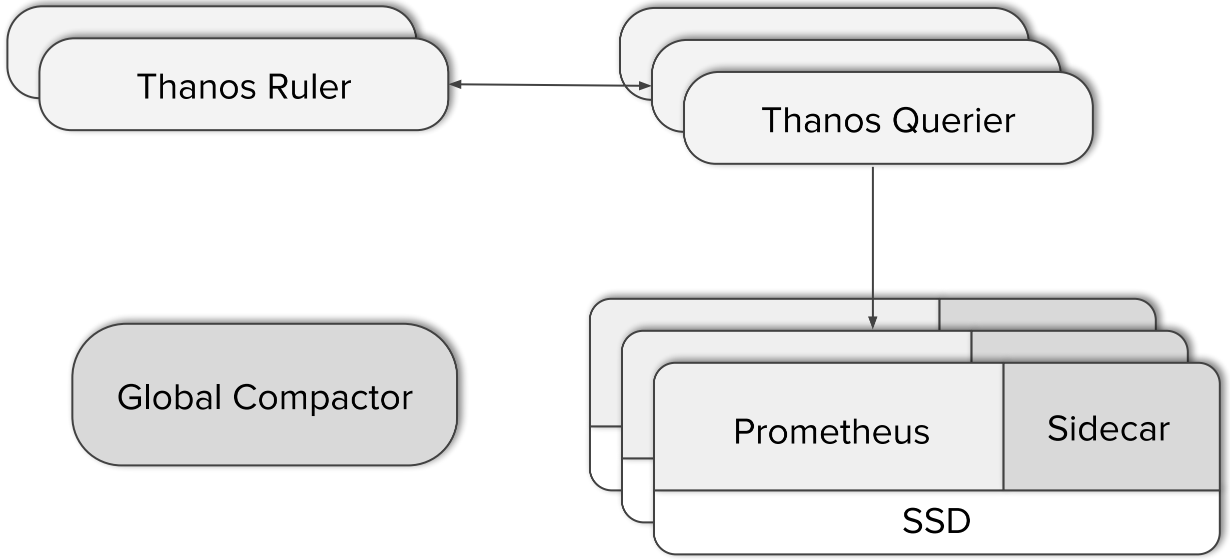


Rule & Alert Engine
Compactor
Thanos Querier
Prometheus
Sidecar
SSD

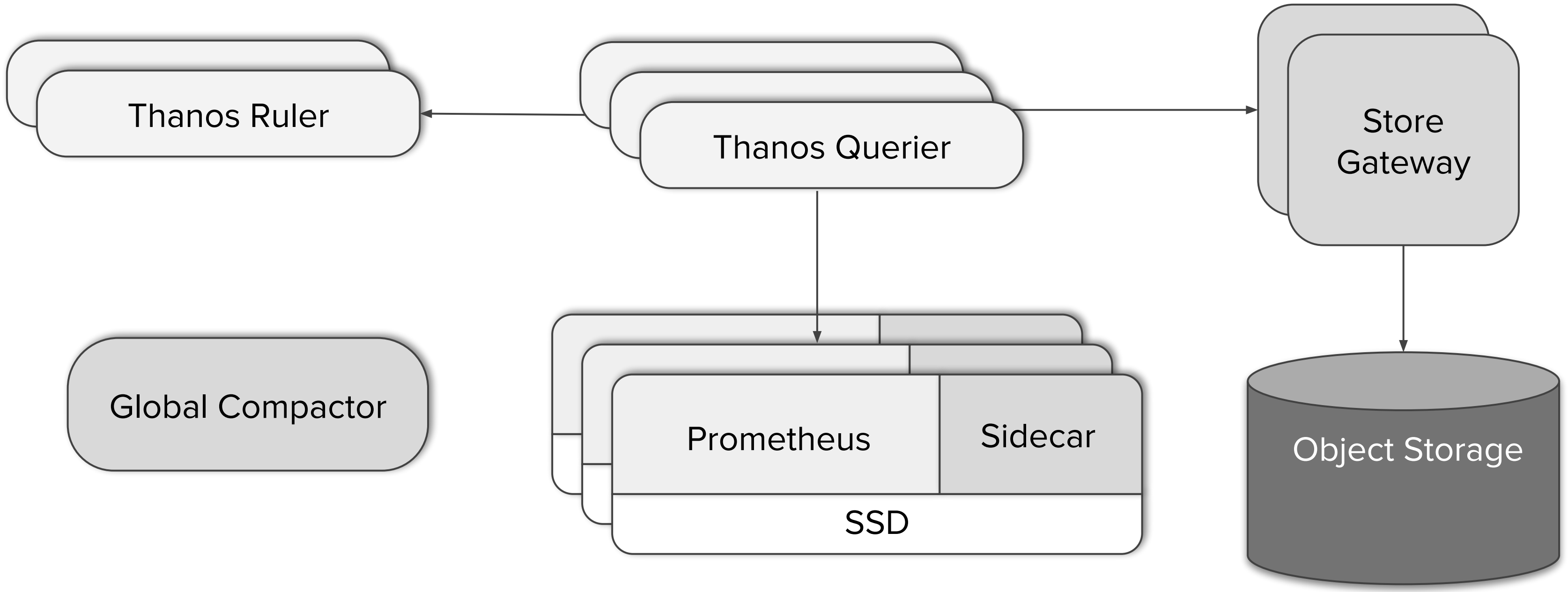
Thanos


Thanos Ruler
Thanos Querier
Compactor
Prometheus
Sidecar
SSD

Thanos

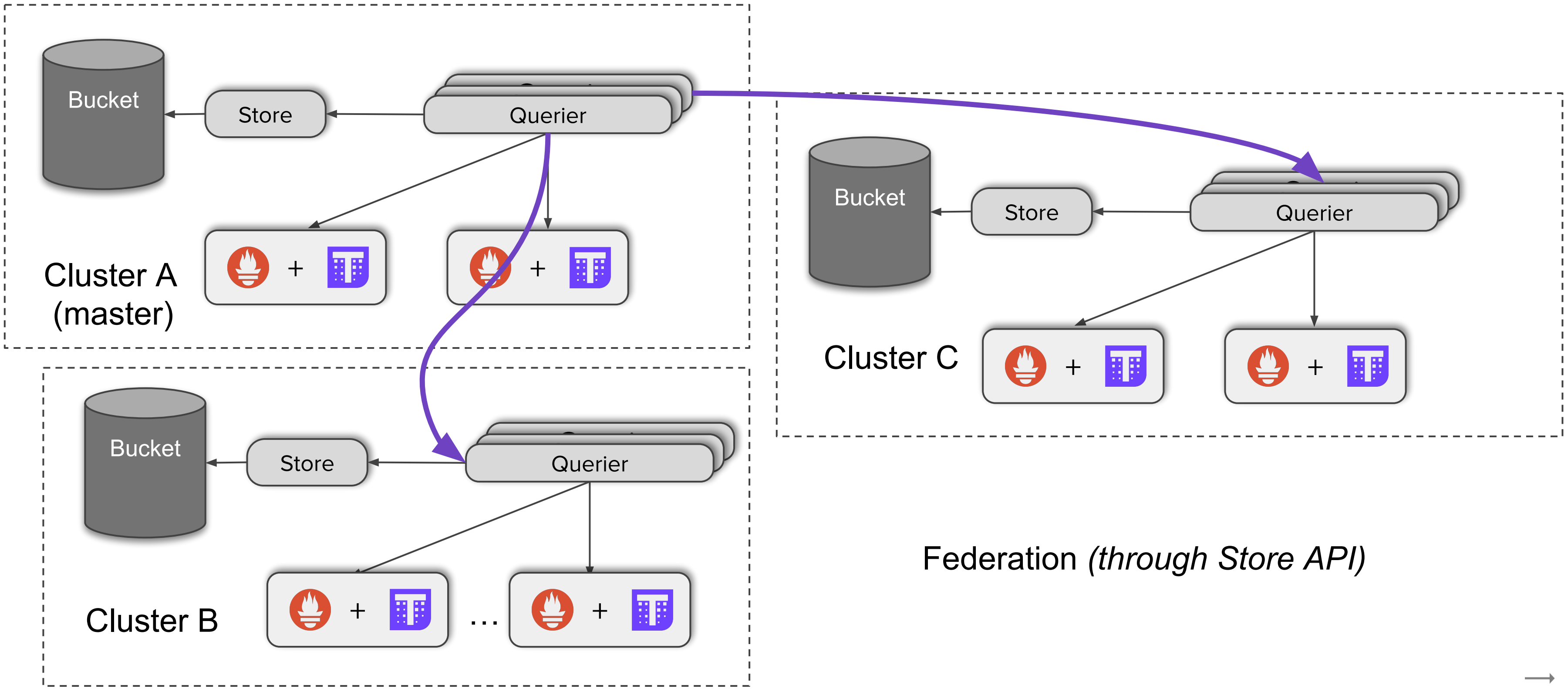

Thanos Ruler
Thanos Querier
Global Compactor
Prometheus
Sidecar
SSD

Thanos


Thanos Ruler
Thanos Querier
Store Gateway
Global Compactor
Prometheus
Sidecar
SSD
Object Storage

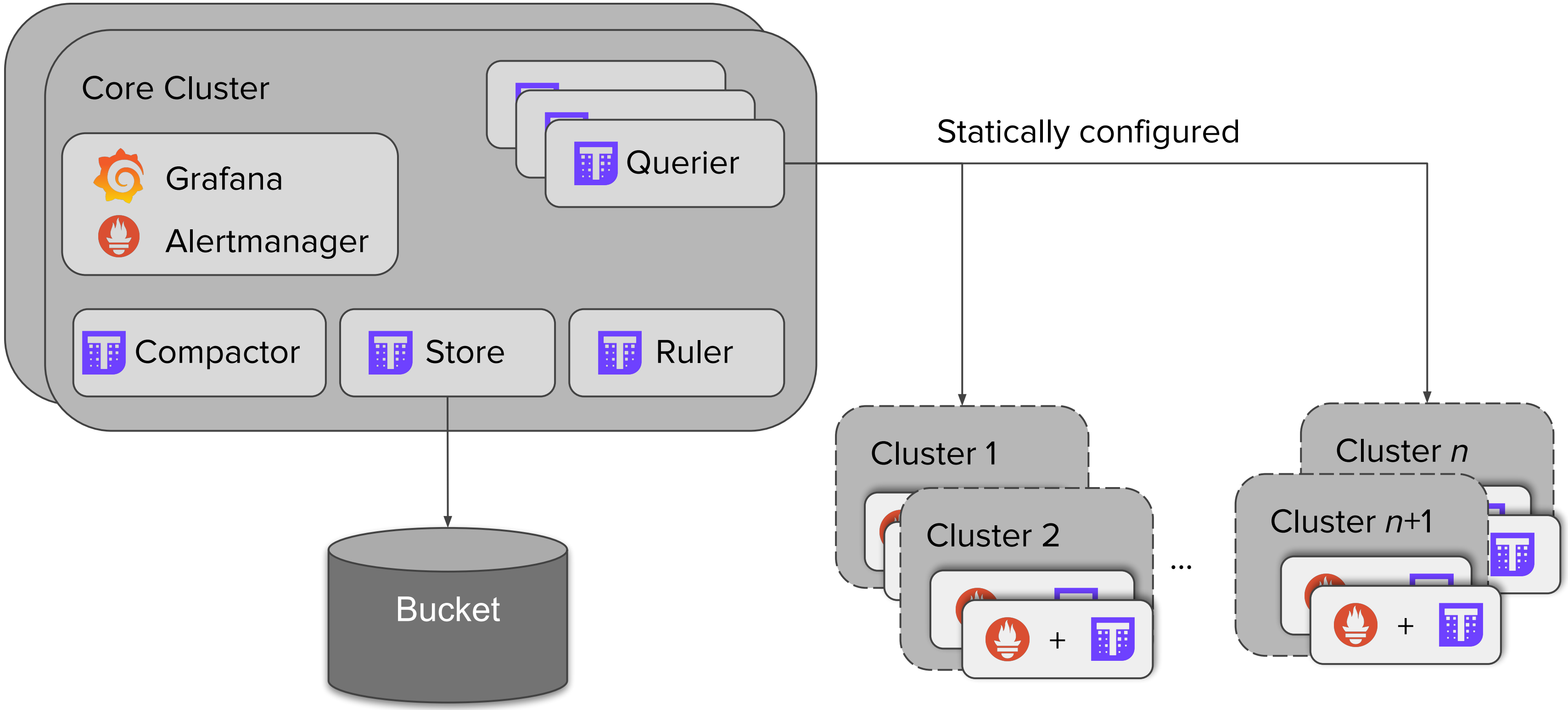
Deployment Models

Federation

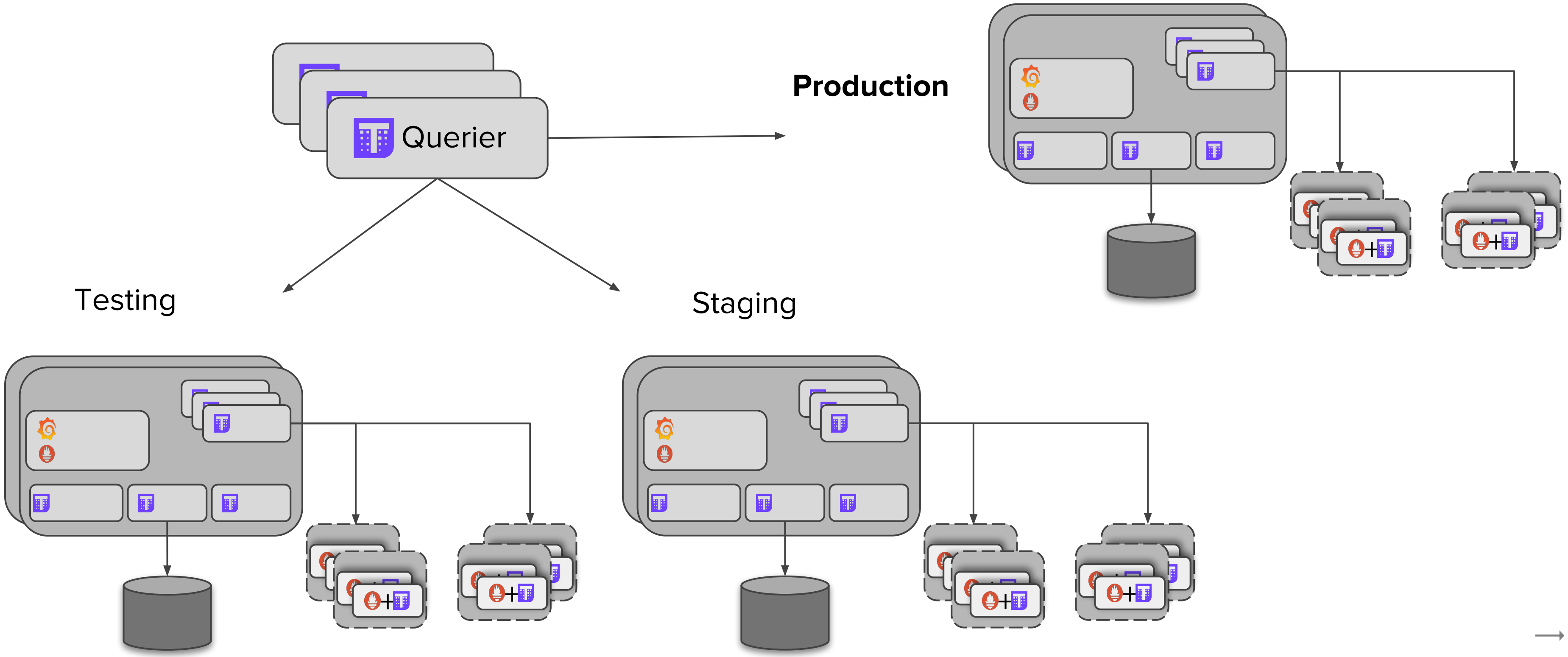


Federation *(through Store API)*

Example Deployment

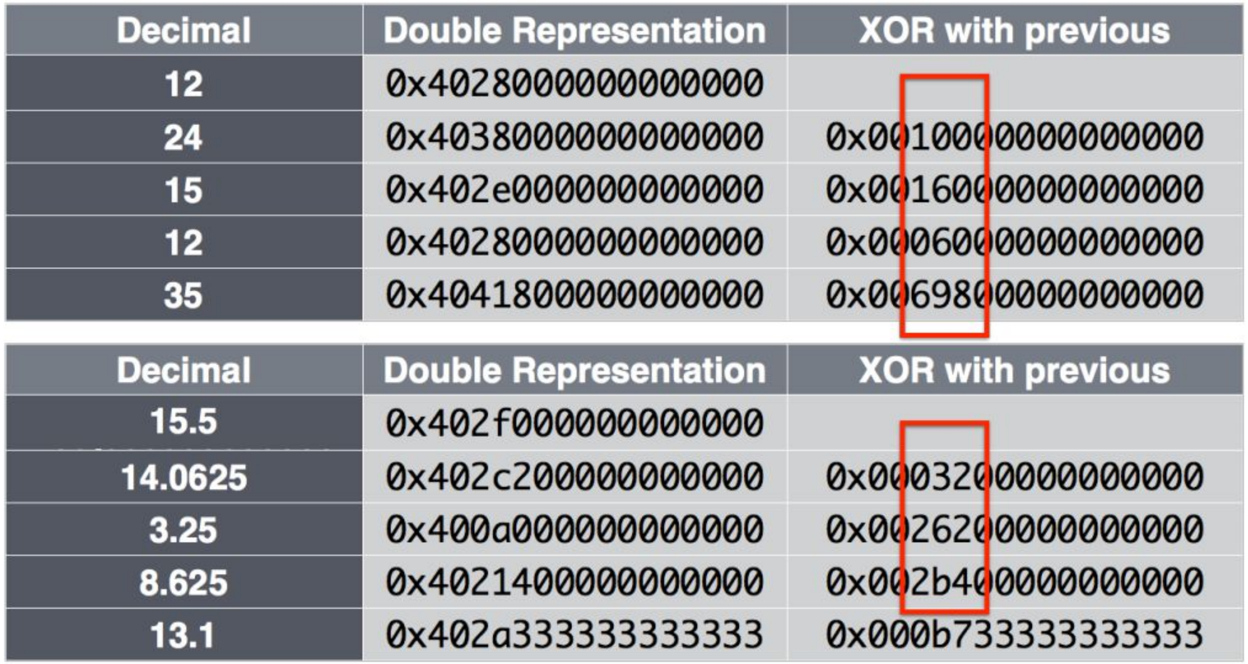


Example Global Deployment



Bonus: Downsampling

Downsampling



Decimal	Double Representation	XOR with previous
12	0x4028000000000000	
24	0x4038000000000000	0x0010000000000000
15	0x402e000000000000	0x0016000000000000
12	0x4028000000000000	0x0006000000000000
35	0x4041800000000000	0x0069800000000000

Decimal	Double Representation	XOR with previous
15.5	0x402f000000000000	
14.0625	0x402c200000000000	0x0003200000000000
3.25	0x400a000000000000	0x0026200000000000
8.625	0x4021400000000000	0x002b400000000000
13.1	0x402a333333333333	0x000b733333333333

Raw: 16 bytes/sample

Compressed: 1.07 bytes/sample

Downsampling

BUT...

Downsampling

Decompressing one sample takes 10-40 nanoseconds

- Times 1000 series @ 30s scrape interval
- Times 1 year

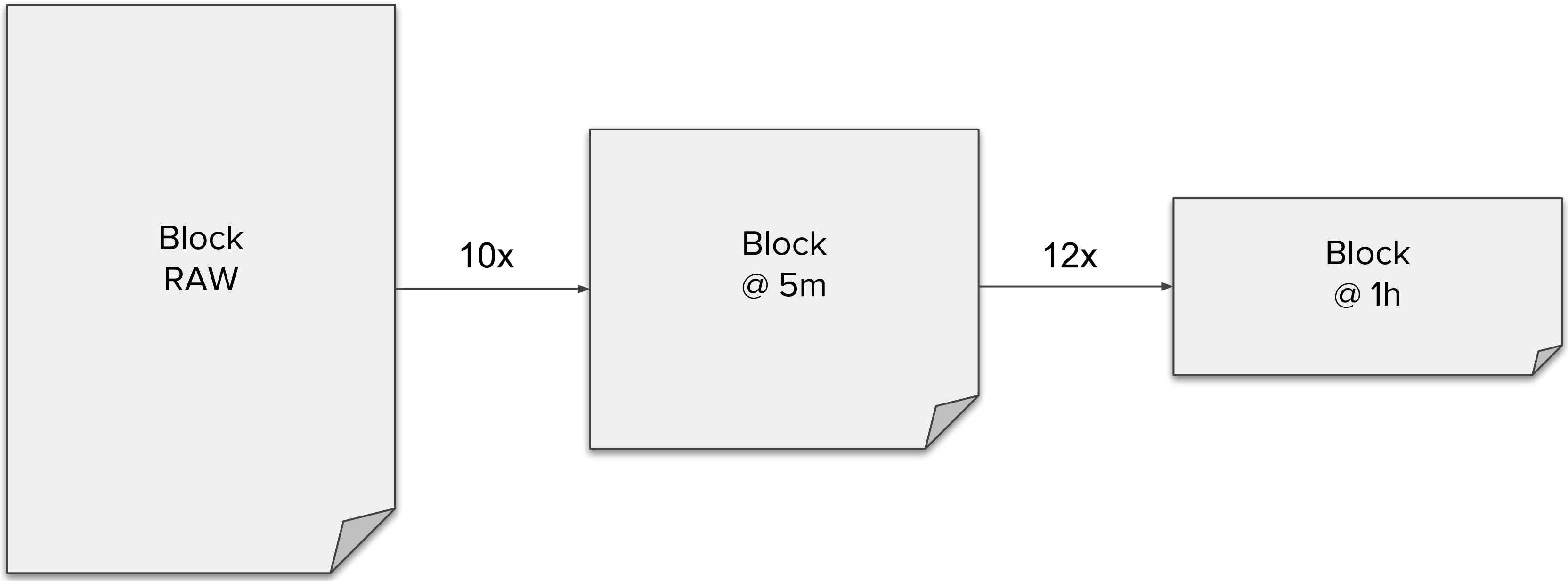
Downsampling

Decompressing one sample takes 10-40 nanoseconds

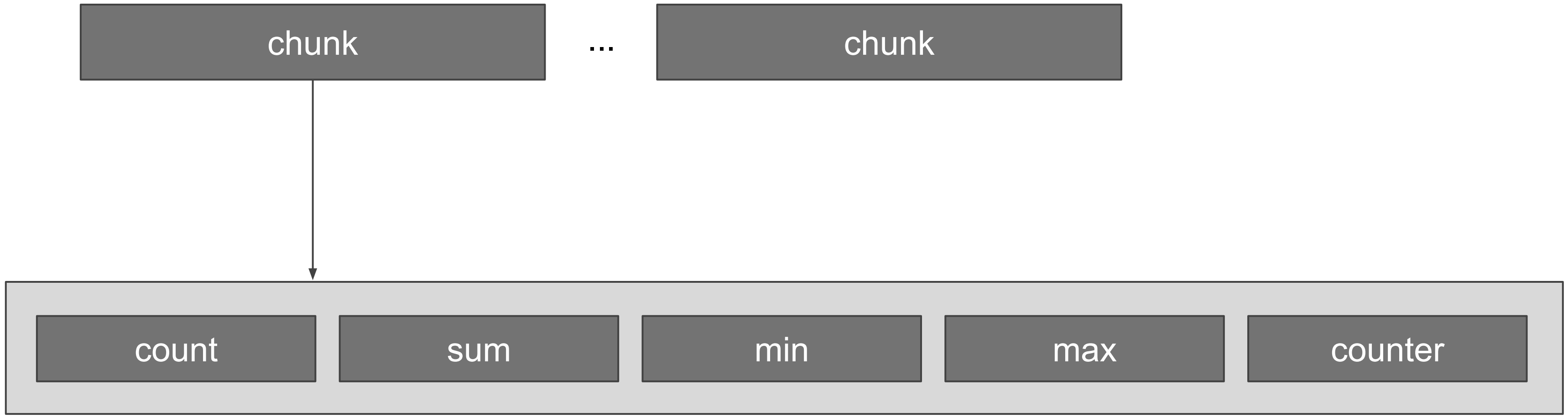
- Times 1000 series @ 30s scrape interval
- Times 1 year

- Over 1 billion samples, i.e. **10-40s** – for decoding alone
- Plus your actual computation over all those samples, e.g. `rate()`

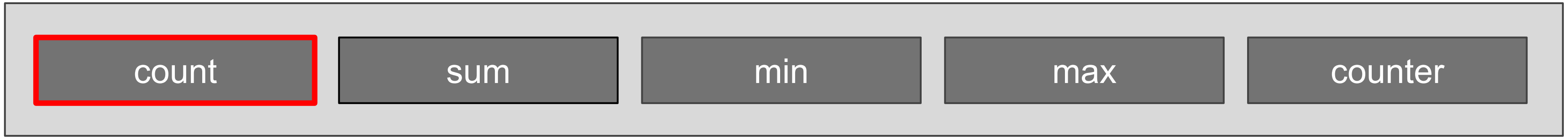
Downsampling



Downsampling

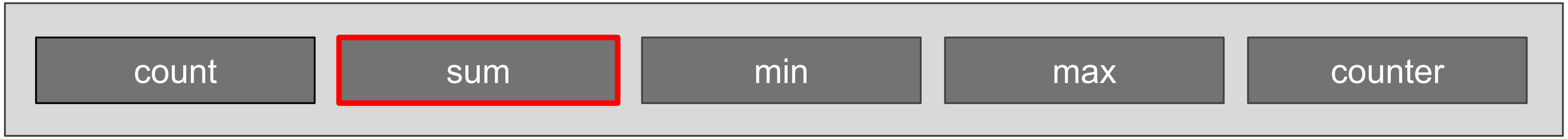


Downsampling



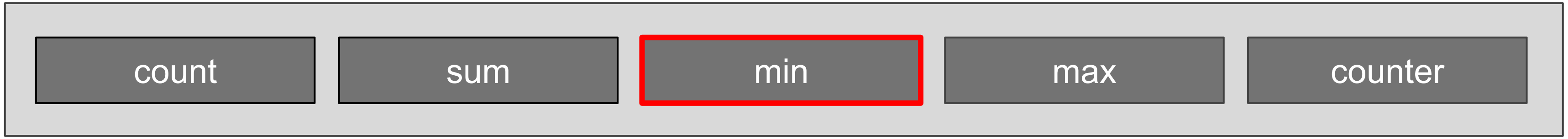
```
count_over_time(requests_total[1h])
```

Downsampling



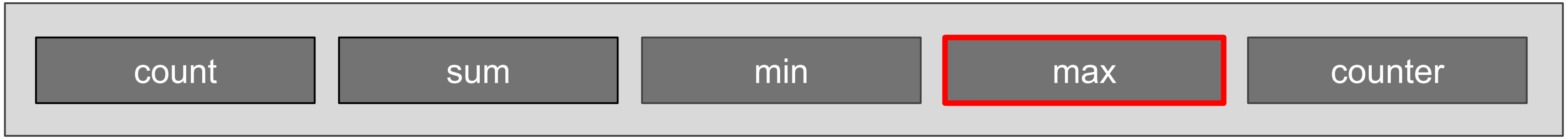
```
sum_over_time(requests_total[1h])
```

Downsampling



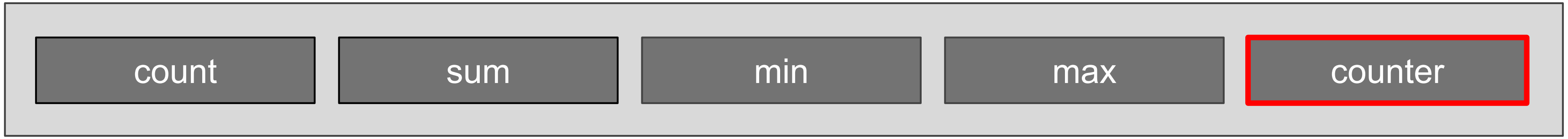
```
min(requests_total)
min_over_time(requests_total[1h])
```

Downsampling



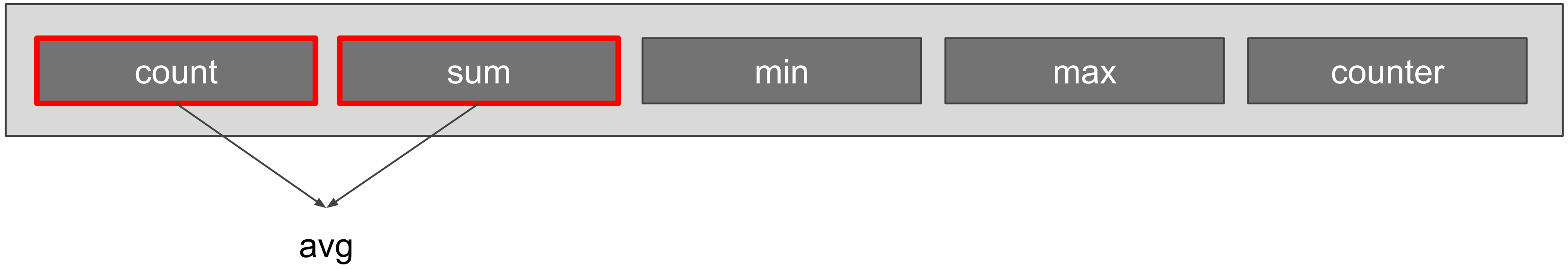
```
max(requests_total)
max_over_time(requests_total[1h])
```

Downsampling



```
rate(requests_total[1h])
increase(requests_total[1h])
```

Downsampling



```
requests_total
avg(requests_total)
...
*
```

Thanos

Goals

- Have a global view ✓
- Have a HA in place ✓
- Increase retention ✓

→

Any questions?

Fabian Reinartz

fabxc

Bartek Plotka

bwplotka *Bplotka*

github.com/improbable-eng/thanos