Bartek Płotka
Software Engineer

bartek@improbable.io
🔗 @bwplotka
“Improbable’s platform, SpatialOS, is designed to let anyone build massive simulations, running in the cloud: imagine Minecraft with thousands of players in the same space or researchers creating simulated cities to model the behaviour of millions. Its ultimate goal: to create totally immersive, persistent virtual worlds.”

- WIRED, May 2017
Prometheus
Agenda
Agenda

- Prometheus
- Thanos
Agenda

Prometheus

Thanos

Prometheus
Agenda

- Prometheus
- Thanos
- Open Source

@bwplotka
Prometheus

https://github.com/prometheus/prometheus
Prometheus

- Rule & Alert Engine
- Query Engine
- Compactor
- Scrape Engine
- Local storage

Services

Alertmanager

Grafana

HTTP Query API
every 15s

/metrics

Prometheus

@bwplotka
Single Prometheus

Cluster

Grafana
Alertmanager
Prometheus

Workloads you want to monitor
Single Prometheus

200 000 samples / sec = 1 CPU, 13 GB RAM = 3 mln series with 15s scrape interval.
High Availability

Cluster

- Grafana
- Alertmanager
- Prometheus

Workloads you want to monitor
Globally distributed clusters

Cluster 1
Cluster 2
Prometheus

Cluster n
Cluster n+1
Prometheus
Problem: Global View + HA

```go
go_memstats_alloc_bytes_total(...) [deduplicated]
```

![Diagram](diagram.png)

@bwplotka
Problem: Metric retention
Problem: Metric retention

- Prometheus
- SSD
- Remote write
Goals

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

See everything from a single place!
Prometheus

- Targets
- Prometheus
- SSD
Sidecar

Targets

Prometheus

Sidecar

SSD

gRPC (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

Store API

Prometheus

Sidecar

SSD

Targets
Global View

```
Querier

Merge

Store API

Prometheus | Sidecar
SSD

Sidecar | Prometheus
SSD

Targets
```

@bwplotka
Global View + Availability

- Prometheus
- Sidecar
- SSD
- Targets

Querier

Deduplicate
Merge

Store API

“replica”:“A”

“replica”:“B”
Thanos

Goals

- Have a global view ✓
- Have a HA in place ✓
Global View + Availability
Global View + Availability
Historical Metrics

What exactly happened $X$ months ago?
Prometheus

Prometheus

Rule & Alert Engine  Query Engine

Compactor  Scrape Engine

TSDB
TSDB Layout

Block 1

chunks

chunks

index

chunks

chunks

Block 2

Block 3

Block 4
Data saving

Prometheus

Sidecar

SSD

Blocks

--gcs.bucket=...

Object Storage

Blocks

Targets
Data saving

```bash
--storage.tsdb.max-block-duration=2h
--storage.tsdb.retention=12h
```
Store Gateway

Object Storage

Blocks

Store Gateway

Querier

Store API

Cache
Ruler
Compaction

Density matters
Compaction

Diagram showing the process of compaction in an object storage system. The diagram illustrates the movement of blocks from object storage to the compactor, which then stores the blocks on disk.
Downsampling
# Downsampling

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Double Representation</th>
<th>XOR with previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0x4028000000000000000</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0x4038000000000000000</td>
<td>0x00100000000000000</td>
</tr>
<tr>
<td>15</td>
<td>0x402e000000000000000</td>
<td>0x00160000000000000</td>
</tr>
<tr>
<td>12</td>
<td>0x4028000000000000000</td>
<td>0x00060000000000000</td>
</tr>
<tr>
<td>35</td>
<td>0x4041800000000000000</td>
<td>0x00698000000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Double Representation</th>
<th>XOR with previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.5</td>
<td>0x402f000000000000000</td>
<td></td>
</tr>
<tr>
<td>14.0625</td>
<td>0x402c200000000000000</td>
<td>0x00032000000000000</td>
</tr>
<tr>
<td>3.25</td>
<td>0x400a000000000000000</td>
<td>0x00262000000000000</td>
</tr>
<tr>
<td>8.625</td>
<td>0x4021400000000000000</td>
<td>0x002b4000000000000</td>
</tr>
<tr>
<td>13.1</td>
<td>0x402a333333333333333</td>
<td>0x000b7333333333333</td>
</tr>
</tbody>
</table>

Raw: 16 bytes/sample

Compressed: 1.07 bytes/sample
Decompressing one sample takes 10-40 nanoseconds.
Decompressing one sample takes 10-40 nanoseconds.

Decompressing 1000 series @ 30s scrape interval for 1 year data takes 10-40 seconds alone.

Plus your actual computation over all those samples, e.g. rate()
Downsampling

Block RAW

10x

Block @ 5m

12x

Block @ 1h
Goals

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

- Rule & Alert Engine
- Query Engine
- Compactor
- Scrape Engine
Thanos

Rule & Alert Engine

Compactor

Scrape Engine

Thanos Querier

@bwplotka
Thanos

Rule & Alert Engine

Compactor

Thanos Querier

Prometheus

Sidecar

SSD
Thanos

Thanos Ruler

Thanos Querier

Compactor

Prometheus

SSD

Sidecar
Thanos
Thanos

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

- Monitoring Cluster:
  - Grafana
  - Alertmanager

- Statically configured / global discovery

- Cluster 1
  - Querier
  - Ruler

- Bucket

- Cluster n
  - Querier
  - Ruler

- Cluster n+1
Open Source & Thanos
Open Source:

It is worth it!
Open source from the very beginning
Be flexible while staying focused
Stay focused: Injection API

<File_sd_config>

File-based service discovery provides a more generic way to configure static targets and serves as an interface to plug in custom service discovery mechanisms.

It reads a set of files containing a list of zero or more <static_config> s. Changes to all defined files are detected via disk watches and applied immediately. Files may be provided in YAML or JSON format. Only changes resulting in well-formed target groups are applied.

The JSON file must contain a list of static configs, using this format:

```
[
  {
    "targets": [ "<host>", ... ],
    "labels": {
      "<labelname>": "<labelvalue>", ...
    }
  },
  ...
]
```

Source: https://prometheus.io/docs/prometheus/latest/configuration/configuration/#<file_sd_config>
Less magic == better
Avoid magic: Gossip example

- Easy to misconfigure
- Hard to debug
- Difficult cross cluster setup
- Over-complicated for Thanos needs
Thanos

Goals

- Have a global view  ✓
- Have a HA in place  ✓
- Increase retention  ✓
- Join effort with community  ✓
Summary

- Start small with just single Prometheus.
- Extend your setup gradually.
  - Just global querier.
  - Federated global querier.
  - Object storage.
- Model your Thanos deployment.

Generally: Share early, keep project focused and simple.
...psst, join our slack workspace for more info!

github.com/improbable-eng/thanos

Credits:
- Percona blog post about Prometheus 2 perf
- Emojis designed by Freepik from Flaticon