Two Households, Both Alike in Dignity

Bartłomiej Płotka & Tom Wilkie
PromCon 2019
Started by Tom Wilkie and Julius Volz in June 2016
Joined CNCF sandbox Sept 2018
https://github.com/cortexproject/cortex

Started by Fabian Reinartz and Bartłomiej Płotka on Dec 2017
Joined CNCF sandbox in Aug 2019
https://thanos.io
When monitoring a global fleet with Prometheus, I need...

1. Global View

2. Multi-Replica Prometheus (HA)

3. Long Term Storage
#1 Global View
Queries over data from multiple Prometheus servers
Thanos: Fanout Queries

#1 Prometheus in each remote cluster has Thanos sidecar.

#2 Stateless Querier anywhere fanouts query to certain Prometheuses.

#3 Queries see all data.
Cortex: Centralised Data

1. Prometheus in separate clusters, remote writes metrics.
2. Scalable Cortex cluster stores metrics from multiple Prometheus servers.
3. Queries go to central cluster, cover all data.

us-west

us-east

push

eu-west
| #1 Global View | Data stays in Prometheus; Fanout query; | Centrally write data to a scalable Cortex cluster; query in one place. |
#2 Multi-Replica Prometheus (HA)

No gaps in the graphs caused by Prometheus server restarts
#1 Each Prometheus replica scraping the same targets has Thanos sidecar.

#2 Thanos Querier resolve gaps in query time.

#3 Queries only ever see a single version of each series.
Cortex: Resolve Gaps at Write Time

#1 Both Prometheus instances in each cluster remote-write metrics to Cortex.

#2 Cortex dedupes samples on ingestion, only storing data from a single Prometheus.

#3 Queries only ever see a single version of each series.
<table>
<thead>
<tr>
<th>#1 Global View</th>
<th>Data stays in Prometheus; Fanout query;</th>
<th>Centrally write data to a scalable Cortex cluster; query in one place.</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2 Multi-Replica Prometheus (HA)</td>
<td>Resolve gaps at query time; only renders single series</td>
<td>Resolve gaps at write time; only store single series.</td>
</tr>
</tbody>
</table>
#3 Long Term Storage

*Store data for long term analysis*
#1 Sidecar syncs TSDB blocks with Object Storage

#2 Thanos allows browsing uploaded blocks, compacting index and downsampling

#3 Queriers have access to both fresh and old data

Thanos: TSDB blocks in object store
#1 Samples from Prometheus are batched up into XOR Chunks in Cortex.

#2 Chunks are periodically flushed to an object store, and an inverted index over the chunks is written to a NOSQL database.

#3 Queries use the index in NOSQL to find relevant chunks, with heavy use of caches.
<table>
<thead>
<tr>
<th>#1 Global View</th>
<th>Data stays in Prometheus; Fanout query;</th>
<th>Centrally write data to a scalable Cortex cluster; query in one place.</th>
</tr>
</thead>
<tbody>
<tr>
<td>#2 Multi-Replica Prometheus (HA)</td>
<td>Resolve gaps at query time; only renders single series</td>
<td>Resolve gaps in write time; only store single series.</td>
</tr>
<tr>
<td>#3 Long Term Storage</td>
<td>TSDB blocks in object storage</td>
<td>NOSQL for index &amp; chunks in object storage</td>
</tr>
</tbody>
</table>
Future
Cortex query-frontend can be put in front of Thanos to accelerate queries using parallelisation and caching.
Cortex now embeds Thanos’s code to read & write blocks from object store for LTS, reduced dependencies and TCO.

https://github.com/cortexproject/cortex/pull/1695
Thanks!

Questions?

https://thanos.io
https://github.com/cortexproject/cortex