metoda

Metoda's journey into ClickHouse

Metoda, Munich, 2022-07-27

metoda

about me

Milovan Zogovic

head of engineering at metoda (Munich)

- ~20 years of experience in IT
- ruby developer by passion
- handling data at scale

business Facts & Numbers

We are the expert for insight-driven growth.

Founded: **2012**

HQ: **Munich**

coverage: **global**

High-quality, global e-commerce **data & insights**

Smart technologie for Amazon (Ads) **process automation &** e-commerce **monitoring**

50+ industry experts & certified Amazon Ads team

Company values: **Drive Success Together**

Awards:


amazon spn certified Service Provider Network

amazon ads Advanced partner

sloggi · STABILO · STEINER · Tefal · Tetra · REMINGTON · BaByliss · BASTEI LÜBBE · BLACKROLL · BRAUN · CASIO · CFP · REVLON · Rowenta · Russell Hobbs · TONI&GUY · VARTA · deuter · DURABLE · emsa · ESCHENBACH · fischer · VAUDE · venta · VICTORINOX · WENKO · wolfcraft · ZTE · GEDORE · GROHE · hauck · HH · KRUPS · LURCH · Melitta

metoda

tech Facts & Numbers

- processing around 10TB of data per day
- running 50 production stacks
- in-house data management solutions

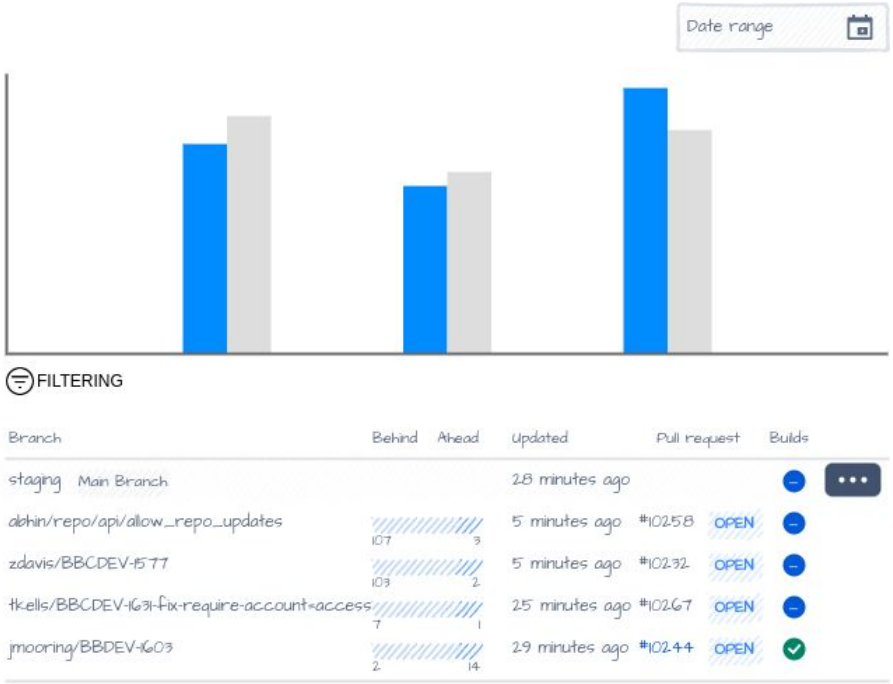
metoda

Why Clickhouse?

A "typical" app

- Custom date ranges
- Customizable chart
- Filtering by any column
- Sorting by any column
- Aggregated facts as columns

- Unlimited history
- User changes take effect immediately
- Dashboard loads fast



metoda

A "typical" implementation

- Using existing OLTP database (postgres in our case)
- Joining facts and dimensions
- Aggregating, filtering and sorting done at database level
- ...

it works for a while, but things starts to get slow

metoda

A "typical" next steps

- Understanding why is it getting slower
- Profiling and fine tuning database queries
- Scaling up the database server (vertically and/or horizontally)
- Partitioning the data
- Optimizing the data ingestion
- ...

metoda

Are we still using the right technology?

metoda

What are some of the alternatives?

we did some research

- AWS Athena - way to slow for our use case
- AWS Redshift - faster, but very expensive to scale
- Elasticsearch - really fast, but no joins
- ClickHouse - seems to fulfil the requirements

metoda

Benchmark

Setup

- ~10GB of data, 150M rows facts, 1M rows dimensions
- Single node setup (with 2GB RAM)
- Using top accounts (by number of facts)
- A random 30 day interval in past 6 months
- 10 concurrent users
- Requests every 2-3 seconds
- 5 minutes benchmark time

metoda

Benchmark

```
checks.........................: 100.00% ✓ 1121      ✗ 0
data_received..................: 958 kB  3.2 kB/s
data_sent......................: 1.4 MB  4.8 kB/s
http_req_blocked...............: avg=2.66ms   min=2.62µs   med=4.38µs   max=208.73ms p(90)=9.16µs   p(95)=11.48µs
http_req_connecting............: avg=801.46µs min=0s       med=0s       max=50.02ms  p(90)=0s       p(95)=0s
http_req_duration..............: avg=195.89ms min=107.73ms med=184.57ms max=795.31ms p(90)=253.33ms p(95)=285.7ms
  { expected_response:true }...: avg=195.89ms min=107.73ms med=184.57ms max=795.31ms p(90)=253.33ms p(95)=285.7ms
http_req_failed................: 0.00%   ✓ 0         ✗ 1121
http_req_receiving.............: avg=1.52ms   min=30.24µs  med=972.84µs max=14.9ms   p(90)=3.58ms   p(95)=5.3ms
http_req_sending...............: avg=35.32µs  min=14.09µs  med=27.26µs  max=157.39µs p(90)=61.82µs  p(95)=70.2µs
http_req_tls_handshaking.......: avg=978.6µs  min=0s       med=0s       max=67.86ms  p(90)=0s       p(95)=0s
http_req_waiting...............: avg=194.33ms min=103.78ms med=183.39ms max=795.23ms p(90)=251.83ms p(95)=285.62ms
http_reqs......................: 1121    3.711205/s
iteration_duration.............: avg=2.68s    min=2.12s    med=2.69s    max=3.66s    p(90)=3.08s    p(95)=3.14s
iterations.....................: 1121    3.711205/s
vus............................: 2       min=2       max=10
vus_max........................: 10      min=10      max=10
```

metoda

Benchmark

Outcome

Results

- Average response times of 200ms on complex queries
- Unfair tests (limited CPU, limited RAM)

Considerations

- Benchmarks ran against fully sorted immutable data
- Data changes will affect performance (versioning, merging, sorting)

metoda

Next steps

metoda

Getting the data in

- Fact data loaded less frequently (in batches)
  - data resides on S3, and loaded directly from there into clickhouse
- Dimension data loaded more frequently
  - source application streams the changes
  - changes buffered and flushed every X seconds
- Versioning via ReplacingMergeTree
  - handles both "update" and "delete" scenario
- View on top
  - hides away complexity from the outside world

metoda

Other considerations

- Data management
  - data catalog (what data is there, who produces it, and how do we access it)
  - dependency management (who consumes what data)
- Making data available outside clickhouse
  - ETL processes
  - AWS integration
- Managing production environment (cluster setup, scaling, backups, availability.. etc)

metoda

Thank you!