ClickHouse at Parcel Perform

Tuấn-Anh Nguyễn

April 12, 2025

Parcel Perform: A Mid-size Company

{Data,AI}-driven Delivery Experience Platform

- Integrated with more than 1100 carriers.
- 8 feature squads.
- ~100 services.
- Daily log volume: 2B records, 2TiB uncompressed.

ELK Stack: The "Tradition"

- Structured logs: loadbalancers (ALB/nginx), services (ELK).
- Monitoring & troubleshooting.
- Kibana for dashboards and ad-hoc queries.

ELK Stack: Problems (End of 2022)

- High disk space usage.
- Slow queries. Timeouts.
- Low concurrent users.
- Operations: complex, not well-documented.
    - Even with AWS-managed OpenSearch.

Observability Is an Analytics Problem

- {Elastic,Open}Search is not an analytics DB.
- We needed columnar storage, at the very least.

Analytics Problems Mean OLAP SQL Databases

- Redshift
- ClickHouse
- Pinot
- Druid

Requirements

- Low-latency for both writes and reads.
- Good integrations with visualization tools.
- Well-documented, straightforward architecture.
    - Day-2 operations.

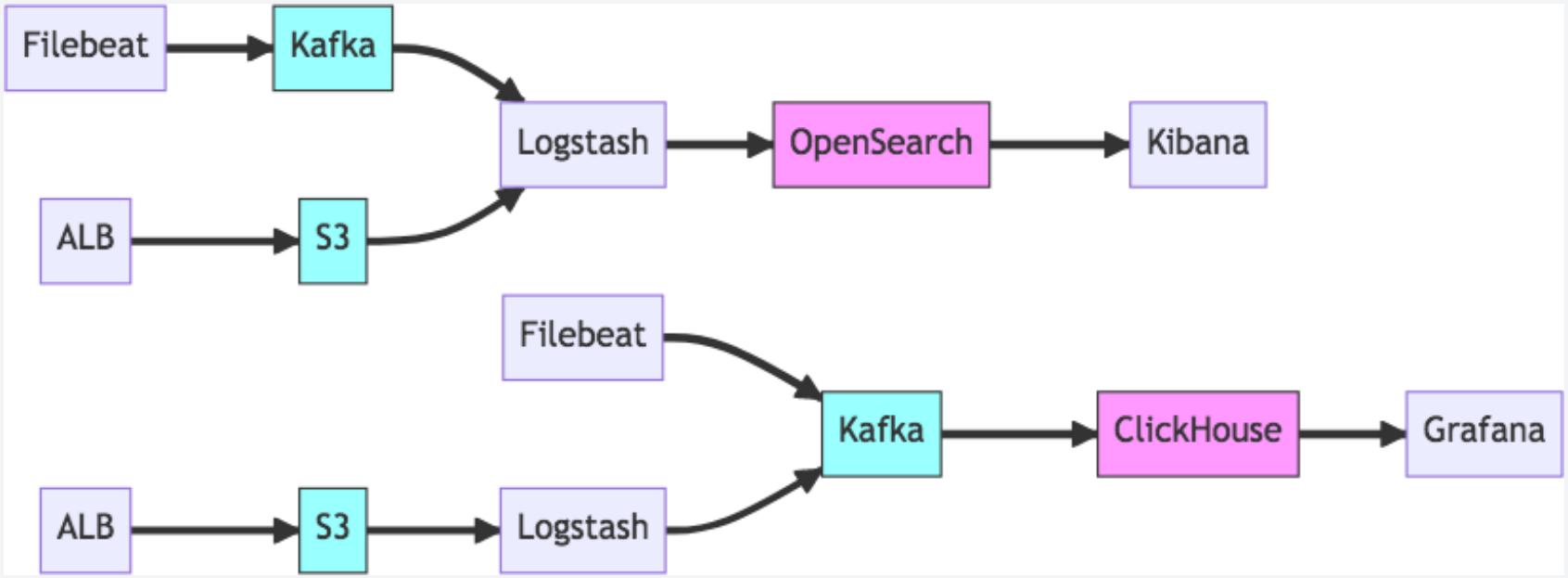
Evaluation of Options

- ~~Redshift~~
  - No streaming ingestion.
  - No time-based data retention.
  - No indexing support.
- ~~Pinot, Druid~~
  - Complex setups.
- ClickHouse
  - Cloudflare, Uber: It can scale up.
  - We: Can it scale down?

ClickHouse Setup

- ClickHouse Cloud didn't exist.
  - We used Altinity k8s operator.
- Kafka engines & materialized views.
- Schema: JSON blob.
  - Well-known generic columns are extracted.
  - Mainly around HTTP requests.
- Streaming pre-aggregation.
- Tailscale for remote access.

Log Pipelines



Schema: Do the Simplest Thing Possible

```
message String CODEC(ZSTD(3)),

service LowCardinality(String) DEFAULT JSONExtractString(message, 'service'),
log_timestamp DateTime DEFAULT parseDateTimeBestEffortOrZero(JSONExtractString(message, 'log
request String DEFAULT JSONExtractString(message, 'request'),
status LowCardinality(UInt16) DEFAULT JSONExtractUInt(message, 'status'),
request_time UInt32 DEFAULT JSONExtractUInt(message, 'request_time'),
body_bytes_sent UInt32 DEFAULT JSONExtractUInt(message, 'body_bytes_sent'),
body_bytes_response UInt32 DEFAULT JSONExtractUInt(message, 'body_bytes_response'),
server_addr String DEFAULT JSONExtractString(message, 'server_addr'),
remote_addr String DEFAULT JSONExtractString(message, 'remote_addr'),
http_user_agent LowCardinality(String) DEFAULT JSONExtractString(message, 'http_user_agent')
http_referer LowCardinality(String) DEFAULT JSONExtractString(message, 'http_referer'),
http_forward LowCardinality(String) DEFAULT JSONExtractString(message, 'http_forward'),
```

Experimentation Results

- OpenSearch: 86 GiB/day.
- ClickHouse: 8.5 GiB/day.
- Query latency: Too low to notice.
- Setup was smooth.

Migration from ELK to ClickHouse & Grafana

- Docs & training.
- Show & tell.
- Ongoing support.
- It's still called log.elk to this day. Naming is hard.
- It's the K that sold the ELK stack.

Next: AWS Cost Analyses

- We were using a lot of services.
- AWS Cost Explorer: inflexible, opaque, not very fast.
- Cost & Usage Reports provide parquet files!
  - Scheduled exports.
  - Detailed data dictionary.
- ClickHouse vs. DuckDB (after ingestion)
  - DuckDB was 2-3 times faster.
  - ClickHouse won in data accessibility.

Cost Analysis: Examples

- Extra CloudWatch cost caused by RDS Proxy's logging.
- Long-term trends of spot instance prices.
- IPv4 cost estimates before the new pricing scheme hit.
- LLM providers' charges and inference profiles.
- Cost attribution to squads, services, usage categories.

Data Retention: Tiered Storage with S3

- ClickHouse can treat S3 as disks.
- Initially: Moving partitions manually.
- Currently: TTL rules.

Growing Need to Evolve the Schemas

- Domain-specific log fields.
- Adding columns to existing log records.
- Straightforward, but time consuming.

Other Use Cases

- ASG events through EventBridge.
- Autoscaling metrics for Keda.
- Flink operations analyses.

Neat Stuff

- Indexes & projections.
- Querying parquet on S3.
- Sourcing dimension tables from other DBs.
- Aggregation combinators.
- System tables.
- S3 disks.
- FROM before SELECT.
- Host-based authentication. (Hello Tailscale!)

Things to Look out for

- Engineer-centric
  - A little strange dialect of SQL.
  - A lot of tuning knobs.
- The mini-batch nature of materialized views.
  - Especially on window queries.
- Merges and storages monitoring.
  - Don't let it merge off S3 storage: live data and TTL merges
  - Too many parts otherwise.

Next Step: More Sources & Flexible Schemas

- New true JSON type.
- OpenTelemetry. (+ custom semantic conventions).
- k8s, GitLab, Jira...

Upcoming Business Analytics Use Cases

- Replacing Redshift with ClickHouse.
- Ad-hoc parquet analysis.
- Querying CDC, from both Kafka topics and S3 archives.
- Customer-facing analytics: ClickHouse Cloud?
  - Better reliability guarantee.
  - Elasticity.