Cache for Cash

Speeding up production with Kafka and Mysql binlog

Barak Luzon, Scale engineer
Infrastructure R&D
Who am I?

Barak Luzon

barak-luzon
barak.luzon
Production Scale

+3.0B Page views /day
500K http Requests /second
+1.5B Monthly Unique Users
100+ TB /a day
100B Protobuf Messages /day
Recommendation lifecycle

Frontend Recommender

Backstage

Backend Processing

Stop/Restrict campaigns
Change configurations

Read:
Campaigns + restrictions
Configuration
Recommended items etc'

Create:
Campaigns restrictions

MySQL
Taboola Data Pipe

- Frontend Data Center
- Frontend
  - Recommender
  - AI Personalization
- Backend
  - Deep Learning Backend
  - Backend Processing
  - Backend - IL

- Kafka
- Frontend-DBs
- Main-DB

- REST API
- MySQL replication
3 Constraints of a Successful Production
#1 - Fast Response time
WASHINGTON – Less than 24 hours before the U.S. completed its withdrawal from Afghanistan, the Taliban stopped a bus headed for the Kabul airport and forced all the passengers off, saying the bus might be rigged with explosives and that it had two possible suicide attackers on board, according to the account of a U.S. citizen who was on the bus.

The U.S. citizen, whose name NBC News is withholding for security reasons, was on the bus with his six daughters Sunday when Taliban fighters stopped it at the Panjsher Pumping Station just outside the airport, two people familiar with the account said. The Taliban told everyone to get off.

The U.S. citizen and his daughters hid in a nearby drainage ditch until the Taliban gave them the all-clear.

"The Taliban were absolutely instrumental," said a senior congressional aide familiar with the account. "Without pulling that bus over, there could have been an attack at the airport that could have killed people, including Americans."
Production speed up

Local machine/In memory

FE DC

Memcached

By key Cache

Bulk Cache

Backend DC

Main-DB

FE-DB

FE-DB

FE-DB

FE-DB
Production speed up

Local machine/In memory

FE DC

Memcached

By key Cache

Bulk Cache

Backend DC

Main-DB

MySQL

FE-DB

MySQL

FE-DB

MySQL

FE-DB

MySQL

FE-DB
Production speed up

By key Cache

Bulk Cache

Memcached

FE-DB

FE-DB

FE-DB

FE-DB

Main-DB

Local machine/In memory

FE DC

Backend DC
Production speed up

- By key Cache
- Bulk Cache

- Local machine/In memory

- FE DC

- Backend DC

- Memcached

- MySQL FE-DB
- MySQL FE-DB
- MySQL FE-DB
- MySQL FE-DB

- MySQL Main-DB
Traditional cache layers
#2 - Fresh Data = higher yield
Cache read pattern

1M cache reads per second
Cache update pattern

13k cache updates per second
Cache TTL tradeoff

- Short TTL == high Load
- Long TTL == stale data
#3 - Lower load = Lower cost
LESS IS MORE
Optimal solution

- Detecting changes on MySQL tables
- Distribute the changes globally
- “Inject” the change to all relevant caches
- Overhead load << short TTL load
Detecting changes on Mysql tables
Multiple sources of data changes

Batch processing layer

MySQL

Backstage

Backend Services
Large amount of changes - overall

Main-DB events

Over 1M events per minute
Mysql Binlog

https://github.com/shyiko/mysql-binlog-connector-java
Taboola Binlog Service

- Parallel processing
- At least once delivery (GTID tracking)
- Easy filtering, configurable
- Better control
- Better visibility

Event type count by table
Desired solution

- Detecting changes on MySQL tables
- Distribute the changes globally
- “Inject” the change to all relevant services cache
Use Kafka for data distribution
** only on selected tables.
Taboola Data Pipe - Before

AI Personalization

Deep Learning Backend

Backend Processing

kafka

Frontend Recommender

Frontend-DBs

Main-DB

MySQL replication

REST API

Frontend Data Center

Backend - IL

Backstage

MySQL replication
Taboola Data Pipe - After

Frontend DBs → kafka → Backend Processing

Frontend Recommender

MySql replication

Main-DB

Backend - IL
Desired solution

- Detecting changes on MySql tables
- Distribute the changes globally
  - “Inject” the change to all relevant services cache
Inject Data to Caches
Inject services with fresh data

- Use embedded kafka consumers in the service
- Inject to cache directly, skip the db fetch
- All data must have a timestamp
- increase/remove cache TTL
Desired solution

- Detecting changes on MySql tables
- Distribute the changes globally
- “Inject” the change to all relevant services cache
Recap

- Stream changes from mysql main using binlog reader
- Send relevant changes to Kafka
- Consume changed information in all services
- Inject fresh data directly to cache
Show me some graphs!
Blazing fast

Total data delay p50 - from transaction begin to apply on the cache: Between 2-3 seconds
Load reduction
Frontend service

- #data changes << cache updates
- Less serialization of same data over and over
Load reduction frontend service - cache updates

Cache injection disabled
Load reduction frontend service - machine load

Cache injection disabled
Load reduction Mysql

Total QPS by DC

Binlog service was down
Thank you!

https://www.linkedin.com/in/barak-luzon/
https://discover.taboola.com/taboola-infrastructure-engineering/