Search at Uber over Apache Lucene

Sia (Egyptian God of Perception)
or
Search In Action

Yupeng Fu, Uber
About Me

- Yupeng Fu
- Search Platform (Sia)
- Real-time Data Platform
- yupeng9@github
- Apache Pinot Committer/PMC
- Alluxio PMC
Vision

A scalable, high-performance platform supporting flexible ranking for Uber’s search and discovery needs
Agenda

- Search Use Cases
- Architecture
  - Component Overview
  - Indexing
  - Serving
- Feature Highlights
Agenda

● Search Use Cases
● Architecture
  ○ Component Overview
  ○ Indexing
  ○ Serving
● Feature Highlights
Feed and Search are two main entries for consumers to discover Eats inventories including restaurants, dishes, grocery stores, etc.
Eats Feed: The Choice Problem

There are 1M+ restaurants on Eats. The average Eater has 1000+ restaurants/dishes to choose from when they open the app / website.

Variable intent
They don’t necessarily know what they want

Complex decision making
Selection (cuisines, restaurants, dishes), price, speed, reliability, dietary restrictions, party size and preferences.

Discovery is about helping Eaters decide what they want to eat or buy.
Pindrop: Build a magical pickup/dropoff experience for everyone

Request/Check out Flow

Home Screen
Location Editor
Product Selection
Pickup Setting

Pickup Refinement

Dispatching
En Route

Enroute flow

Rendezvous

Good evening, Yang

Home Screen
Location Editor
Product Selection
Pickup Setting

Pickup Refinement

Dispatching
En Route

Rendezvous

Anderson is on the way

Send a message

3M633AF2 Honda Civic - Silver
How do we determine the pickup location?

Two Steps:

1. Where is the Rider? (Request Location)
2. Where should the rider get picked up? (Rendezvous Location)

We serve 500+ locations per second across rides and eats
Search powered pickups

- Deterministic & limited set of locations
- Features be computed offline globally
- Sia supports ID based lookups
- Search for pre-generated Pickup Locations
- Rank based on request parameters and user preferences
Indigo: Real-time geospatial search for Ride’s match

- **RT-API | Edge Gateway**: All the core trip APIs for personal transport for riders, drivers...
- **Rider**
  - Eyeball Requests
  - Trip Request
  - Fulfillment
    - Trip lifecycle state management APIs and events
  - Upload Location
  - Plan Request
- **Driver**
  - Update Driver Location Index
  - Update Trip Lifecycle State
- **Indigo**
  - Keeps track of where vehicles are in real time and serves queries for nearby vehicles
- **Control Tower & Multileg Matching engine**
- **Nearby Cars**

**Flow**
1. Rider requests a trip.
2. Fulfillment handles trip lifecycle state management.
3. Plan Request is sent to the driver.
4. Driver sends a location update.
5. Upload Location is executed.
6. Offers are executed and nearby cars are matched.
7. Nearby cars are served queries for nearby vehicles.
Geosearch: Uber’s in-house location search engine

- Multi-billions queries globally per month
Agenda

- Search Use Cases
- **Architecture**
  - Component Overview
  - Indexing
  - Serving
- Feature Highlights
Challenges

● Large volume of serving traffic: tens of thousands QPS
● Geospatial data heavy
  ○ Discoverability
  ○ Deliverability
● Frequent updates
  ○ Driver/Rider location
  ○ Store open/close
  ○ Availability of dishes/items
● Complex ranking methods
  ○ Fulfillment match
  ○ Semantic search
● Integrate with Uber infra components
Evolution from Elasticsearch® over Lucene®

- Updates to index are costly:
  - Pushed at document granularity
  - Includes database like consistency guarantees, replication logs
  - Search nodes have to double as indexing nodes
- Very hard to rebuild index
- Schema changes difficult
- Obsolete data in index difficult to delete
- Document order in index cannot be specified
- Query and ranking functionality not tuned for Uber’s needs
- Difficult to integrate with Uber components
Agenda

- Search Use Cases
- Architecture
  - Component Overview
  - Indexing
  - Serving
- Feature Highlights
Sia Overview

- Powered by Lucene
- Inspired by Linkedin’s Galene
- Ingestion
  - Offline index building
  - Streaming index updates
- Serving
  - Search nodes
  - Aggregator
- Geo-sharding
  - Latitude-based
  - Hexagon-based
- Static ranking and early termination
- Integrates with Uber components
  - gRPC, Muttley
  - Rate limiting, circuit breaker
  - Apache Kafka®, ELK®, M3
  - Terrablob(ObjectStore), Hadoop
  - Schemaless, DocStore
  - Security/Compliance
Agenda

● Search Use Cases
● Architecture
  ○ Component Overview
  ○ Indexing
  ○ Serving
● Feature Highlights
Sia Overview - Indexing

- **Lambda Architecture** for indexing
- **Three layer Index**
  - **Base Index**: Immutable index of Lucene segments built periodically
  - **Snapshot Index**: Continuous snapshots of Lucene segments for node recovery, before getting merged into base eventually
  - **Live Index**
    - Support field level updates
    - In-memory implementation of lucene datastructures with lock-free updates
    - Updates are immediately available for querying (vs waiting for “commit” on ES)
The Base Index

Source of Truth (Online + Offline) -> Base Index Build (Spark) -> Base Index (Shard1), Base Index (Shard2), Base Index (ShardN)
The Base Index

- Normal Lucene index
- Mapped to memory for serving
- Offline index builds - no impact on search performance
- Clean reset
- Schema changes, versioning become easy
- Index building can be parallelized via mini shards
- Index merge from mini shards
- Can control order of documents in index
The Live Index

Sia Stream

Kafka -> Live Index

Base Index
The Live Index

- In-memory data structure
- Ingestion from Kafka
- Lock-free updates
- Updates at finest level of granularity
- Updates have almost no impact on query latencies
- Layered on top of Base Index (overrides Base Index)
- Tombstoning to support deletes
- Preserves document ordering in Base Index
- Supports addition of new documents
- Off-heap SkipList for better memory management and quick access
The Snapshot Index
The Snapshot Index

- Normal Lucene index
- Periodically persisted from Live Index
- More compacted form by merging live index
- Layered on top of Base Index (overrides Base Index)
- Layered below Live Index (overridden by Live Index)
- New documents carried over from Live Index
- Tombstones carried over from Live Index
- Preserves document ordering in Base Index
Agenda

- Search Use Cases
- Architecture
  - Component Overview
  - Indexing
  - Serving
- Feature Highlights
Basic Query Infrastructure

- Lucene queries are implemented recursively
- Only term queries (the leaves in query expressions) operate directly on index
- So we only need to change the term query implementation

So: \texttt{st:orderable}

Becomes: \texttt{st:orderable\_base OR st:orderable\_snapshot OR st:orderable\_live}

with consideration for tombstones and optimizations

- Therefore all Lucene and ElasticSearch queries remain supported
Serving Stack

- Aggregator + sharded search nodes
- gRPC API - same API at aggregator and search nodes
- Query understanding and query rewriting at aggregator
- Aggregator forwards rewritten query to search nodes
- Search nodes perform retrieval and at least one ranking pass
- Aggregator combines results from search nodes
- *Aggregator can be used as a library* - with in-process gRPC call (for Java only)
Agenda

- Search Use Cases
- Architecture
  - Component Overview
  - Indexing
  - Serving
- **Feature Highlights**
Geo-Sharding

- Latitude-based sharding
- Divide the world map into narrow stripes
- Buffer degree for over indexing the docs in the search radius
- Cover all timezones for serving
Geo-Sharding

- Latitude-based sharding challenges
  - Shard size skewness
  - Unnecessary scanning in retrieval

https://brilliantmaps.com/cities-transposed-latitude/
Geo-Sharding

- Hexagon-based sharding
- Tile the world map into hexagons
- Use higher-resolution hexagons for buffer
- Evenly sized shards via bin-packing solution
Time travel to earlier index version

Index corruption is hard to mitigate

- Rolling back to previous index took hours / days
- Building a new index takes hours / days

8+ hours to catch up
**Time travel:** keep all versions fresh with snapshot cluster

- Query Aggregator
- Sia Cluster (Current Version)
  - Previous Version (Disk)
- Live Updates
- Continuous Ingestion
- Indexes + Snapshots (HDFS / S3)
- Save Snapshot
- Load Snapshot
- Snapshot Cluster (Current Version)
  - (Previous Version)
Time travel: Mitigation flow

Sia Cluster
(Current Version)
Previous Version (Disk)

Query Aggregator

Live Updates
Continuous Ingestion

Indexes + Snapshots
(HDFS / S3)

Save Snapshot

Snapshot Cluster
(Current Version)
(Previous Version)

Load Snapshot

Time Travel (20min)
Time travel: Mitigation flow

Query Aggregator

Sia Cluster
Current Version (Disk)
(Previous Version)

Live Updates

Continuous Ingestion

Indexes + Snapshots
(HDFS / S3)

Save Snapshot

Snapshot Cluster
(Current Version)
(Previous Version)

Load Snapshot

Time Travel (20min)
Semantic Search

- Next-gen search architecture
- Incorporating semantic signals like context
- Deep integration with ML
- Better search quality

Query

“Hot drink” + attributes

Documents

Apple cider, Hot cocoa, Starbucks coffee

Documents

New York

Downtown conrad

Conrad New York
102 N End Ave, New York City

Conrad New York
151 W 54th St, New York City

Conrad
102 N End Ave, New York, NY

Conrads Famous Bakery
856 Utica Ave, Brooklyn NY

Conrad’s Pharmacy
333 Long Beach Rd, Island Park, NY

Set location on map
KNN in Sia

- Built on top of Lucene’s KNN
- Geo-sharded HNSW graphs at Uber scale
- Pre-filter with location conditions
- Filter Cache for latency improvement
Q&A

Apache®, Apache Kafka®, Apache Flink®, Apache Pinot®, Apache Hadoop® and their logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks. ElasticSearch® is registered trademark of Elastic Company.