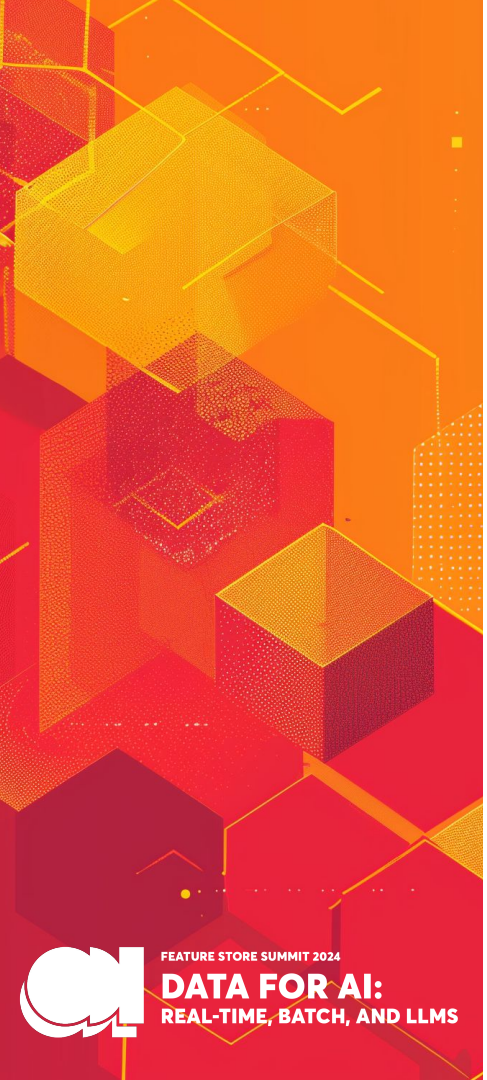


Cassandra Immutable Store

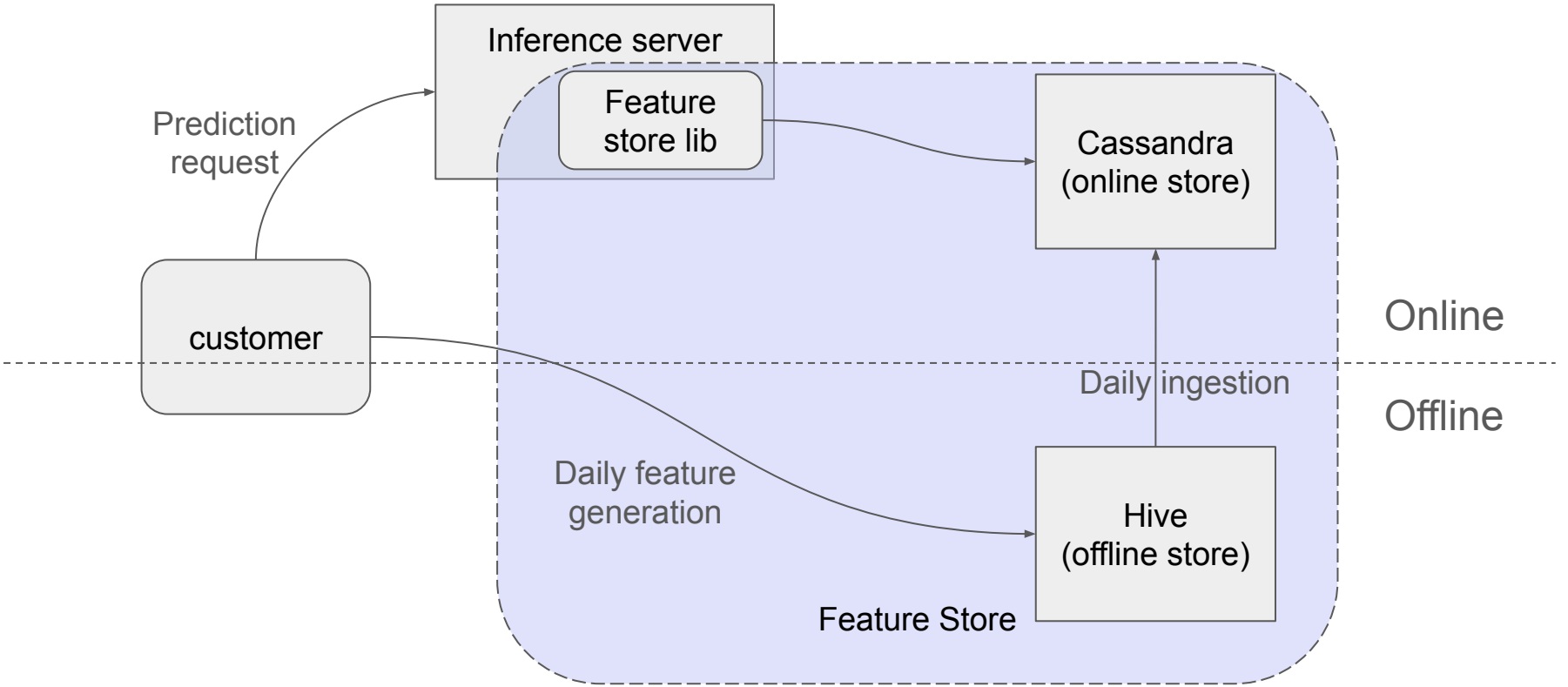
Chunhao Zhang, Sr Software Engineer, Uber

Organized by HOPSWORKS

Aganda

- How Michelange used Cassandra
- Issue of using Cassandra
- Concept of Immutable Store
- Deep dive

How Michelangelo Used Cassandra


Inference server
Feature store lib
Prediction request
customer
Cassandra (online store)
Online
Daily ingestion
Offline
Daily feature generation
Hive (offline store)
Feature Store

Cassandra Schema

uuid	**timestamp**	feature_1	feature_2	…
xxx	1727215689000	1	abc	…

- New row of the same uuid will not override old rows
- Filter by timestamp to get latest row
- Longer TTL require more disk

Issue of Using Cassandra

Issue of Using Cassandra

Data Expired Outage

Write / Deletion out of sync

- Data deletion by TTL
- Data ingestion by periodical pipeline, can delay

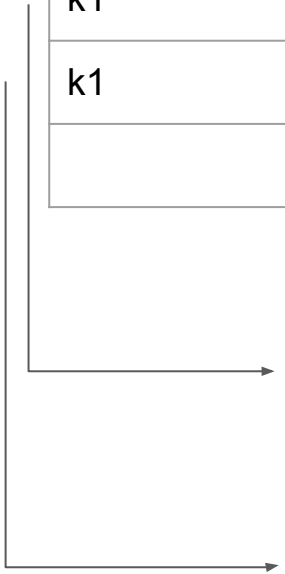
Concept of Immutable Store

Concept of Immutable Store

- No TTL
- Each ingestion ingests into a new table
- Ingester deletes the old tables

Deep Dive

key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	DELETED
k1	t1	run_2	2000		t1_v2	COMPLETED
k1	t1	run_3	3000		t1_v3	COMPLETED



uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	3000	2	abc	…

Data Ingester

- Create new version table and update schema table
- Read data from Hive and ingest into Cassandra
- Launch a new task to recycle old version tables
  - Old COMPLETED versions
  - Previous failures

key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	COMPLETED
k1	t1	run_2	2000		t1_v2	COMPLETED

uuid	**timestamp**	feature_1	feature_2	…
xxx	1000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	COMPLETED
k1	t1	run_2	2000		t1_v2	COMPLETED
k1	**t1**	**run_3**	**3000**		**t1_v3**	**PENDING**

uuid	**timestamp**	feature_1	feature_2	…
xxx	1000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	3000	2	abc	…

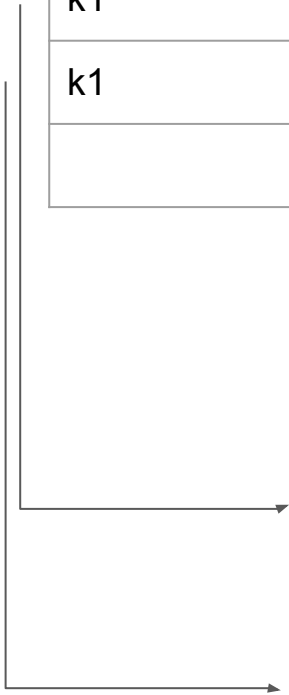
key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	COMPLETED
k1	t1	run_2	2000		t1_v2	COMPLETED
k1	t1	run_3	3000		t1_v3	**COMPLETED**

uuid	**timestamp**	feature_1	feature_2	…
xxx	1000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	3000	2	abc	…

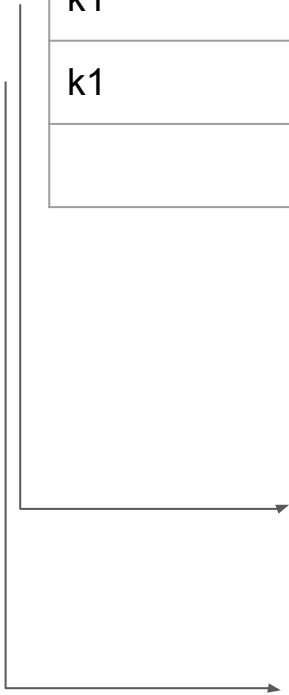
key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	**DELETING**
k1	t1	run_2	2000		t1_v2	COMPLETED
k1	t1	run_3	3000		t1_v3	COMPLETED



uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	3000	2	abc	…

key_space	**table**	**run_id**	**data_ts**	**create_ts**	version_name	status
k1	t1	run_1	1000		t1_v1	**DELETED**
k1	t1	run_2	2000		t1_v2	COMPLETED
k1	t1	run_3	3000		t1_v3	COMPLETED



uuid	**timestamp**	feature_1	feature_2	…
xxx	2000	1	abc	…

uuid	**timestamp**	feature_1	feature_2	…
xxx	3000	2	abc	…

Cassandra Client

- Maintain a map between table name and version table name
  - Prefer the COMPLETED version with largest data_ts
- Periodically query the metadata table to update the map
- Customer query the original table name
- Cassandra client modify the query to query version table

Thank you