Uber's Risk Knowledge Platform

Jean Cai, Senior Software Engineer
Kaavya Srinivasan, Software Engineer
Christopher Settles, Machine Learning Engineer

Uber

FEATURE STORE SUMMIT 2023

Organized by HOPSWORKS

Agenda

- Platform Overview
    - Motivation
    - Architecture
    - Feature Flow
- uFlow Feature Store
    - Batch Feature Computation
    - Near Real Time Feature Computation
    - Aggregator Computation
    - Real Time Feature Computation & Fetching
- uGraph Feature Store
    - Why graph?
        - Motivation
    - OLTP and OLAP graphs
        - Ingestion framework
    - Feature Computation through Cypher
- Machine Learning for Risk Assessment
    - Realtime predictions
    - Batch Feature Backfilling
    - Streaming Feature Backfilling
    - Powerful modeling

Platform Overview

Motivation

Scalable, self-service Feature Engineering Platform for defining, computing, and monitoring features for predictive decisioning.

Payments Fraud

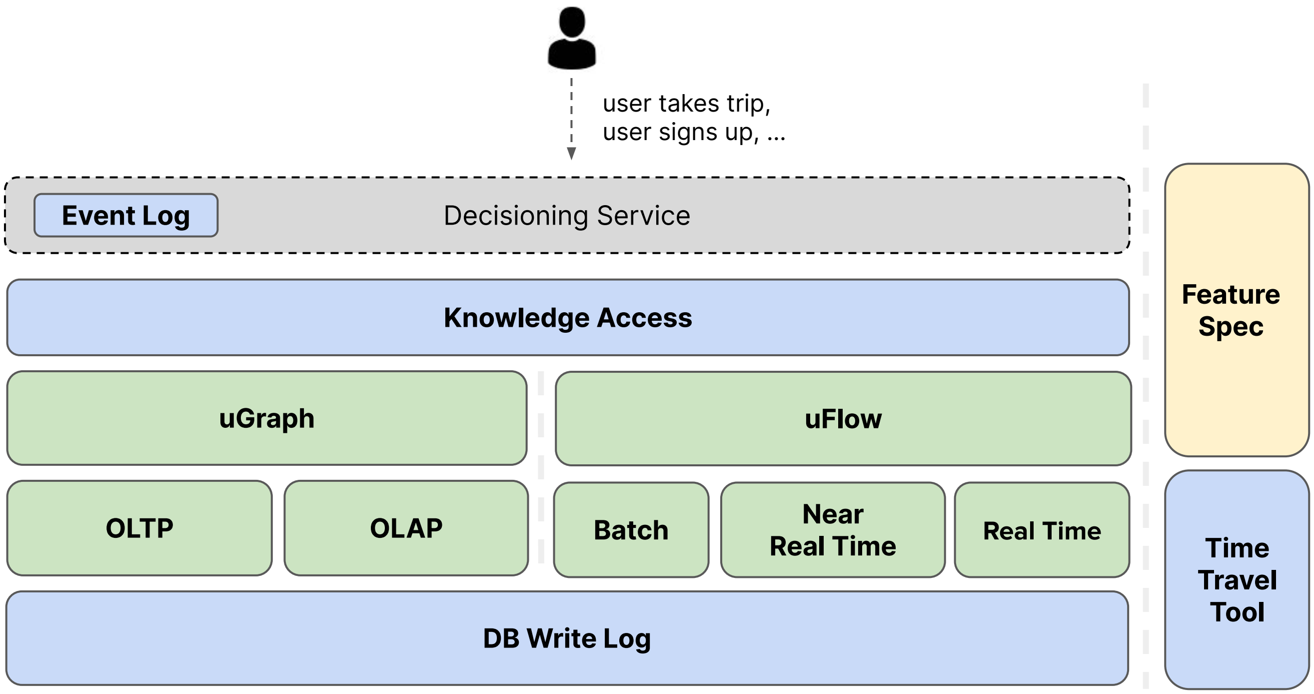
Promotions Abuse

Account Takeover

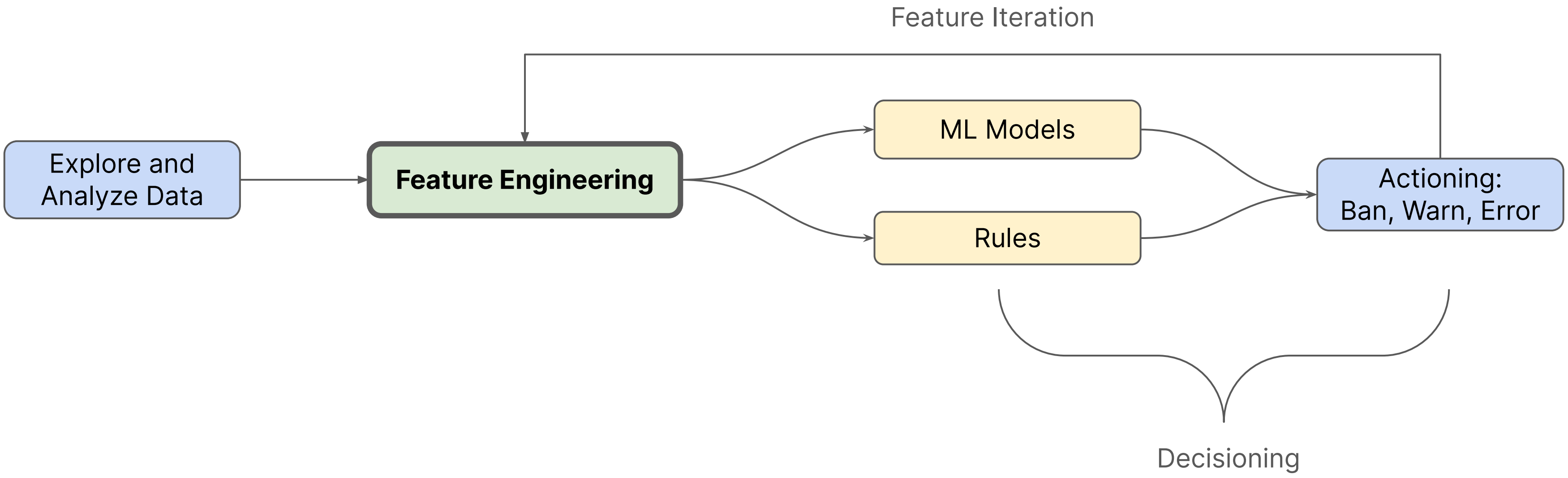
Misconduct

2015 2023

Architecture



E2E Feature Engineering Flow



uFlow Feature Store

Overview

Compute

Batch
Hive
Spark

Near Real Time
Kafka
Flink

Realtime
RPCs
@prediction time

Feature Store
Cassandra

Consolidate

Self-serve

Scale

ML-ready

Manage

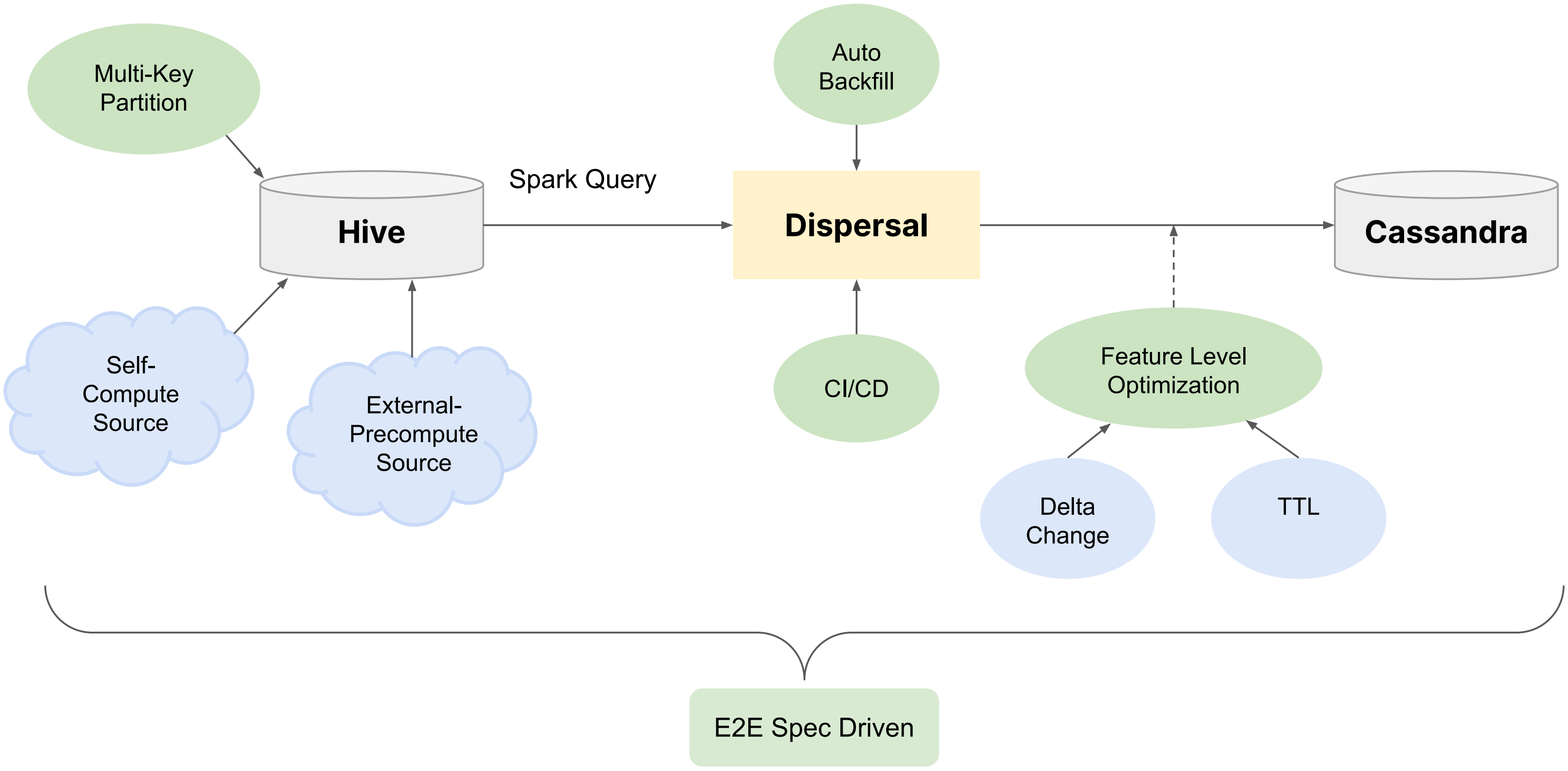
Feature Catalog
- Definitions
- Metadata

Feature Quality
- Correctness
- Distribution
- Completeness
- Freshness
- Anomaly Detection

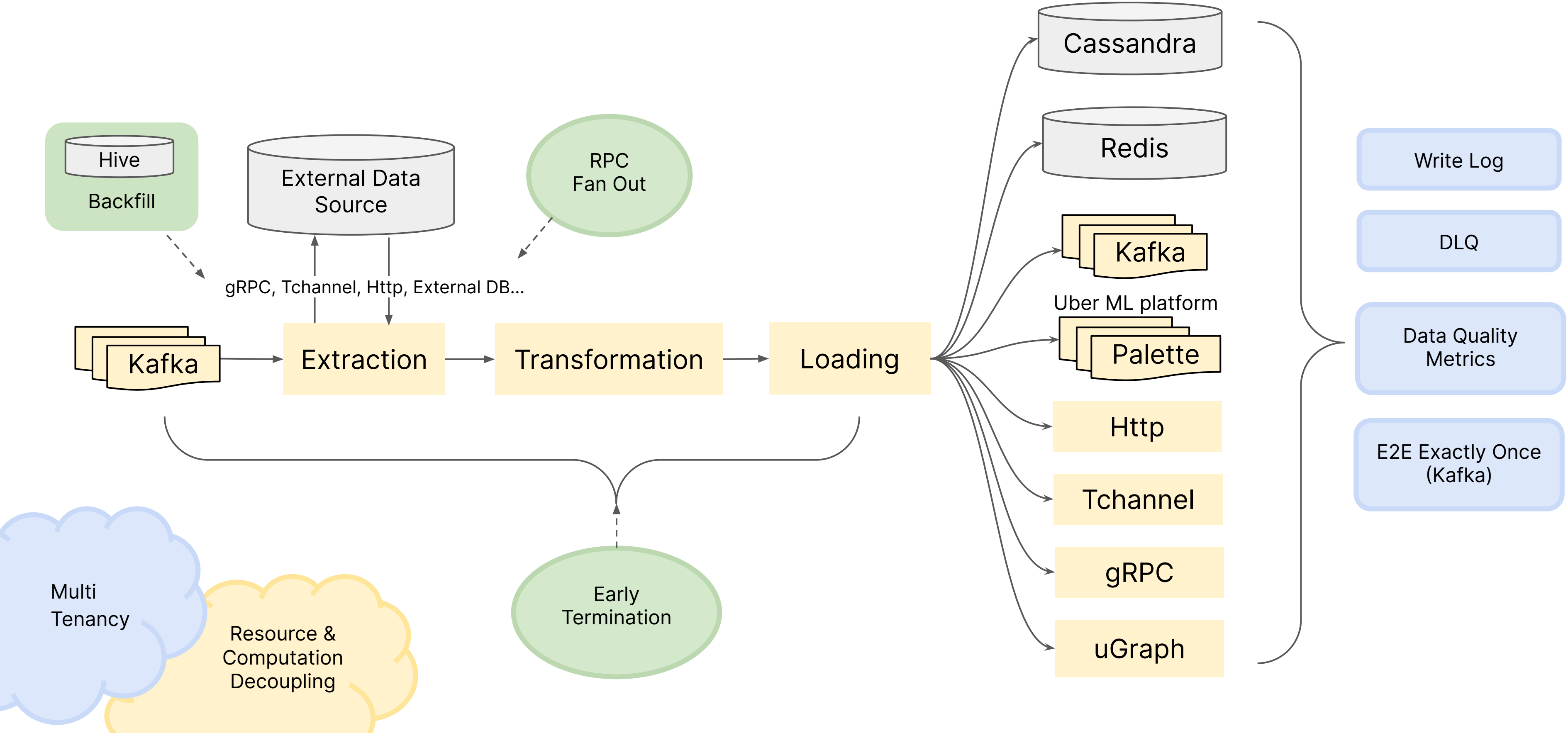
Feature Lineage
- Tracing Up & Down Stream
- Feature Correlation

Feature LifeCycle
- Feature Creation
- Evaluation / Deprecation

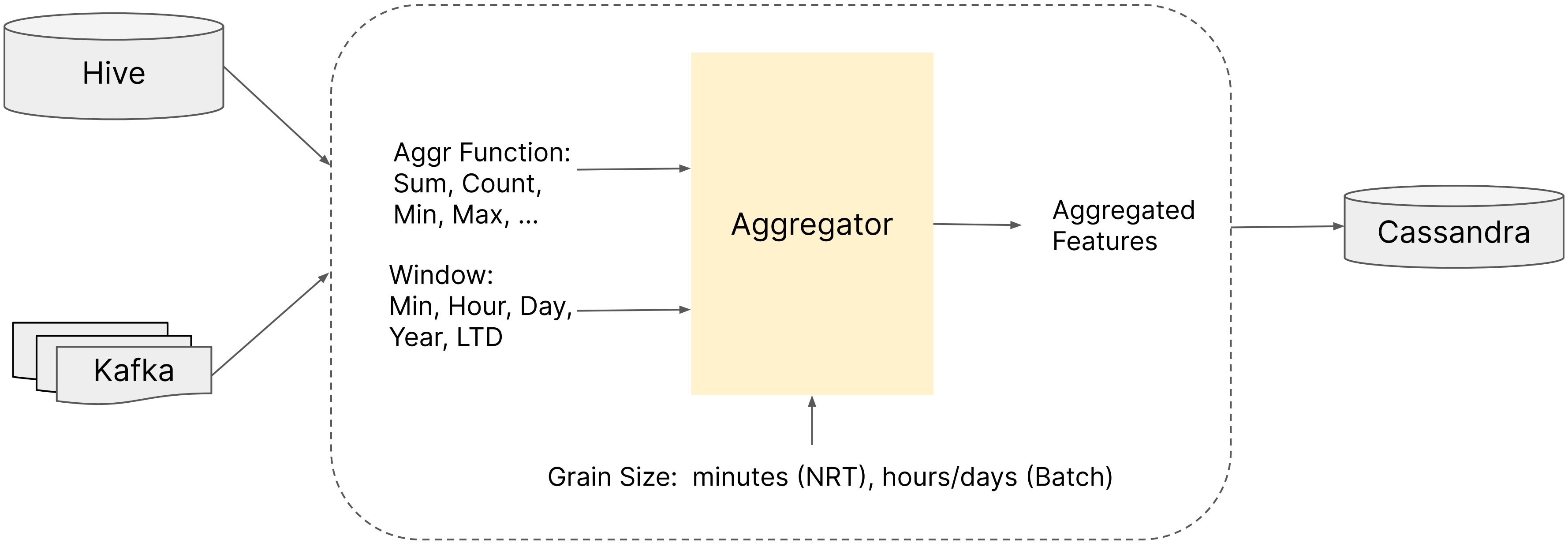
Batch Feature Computation



Near Real Time Computation

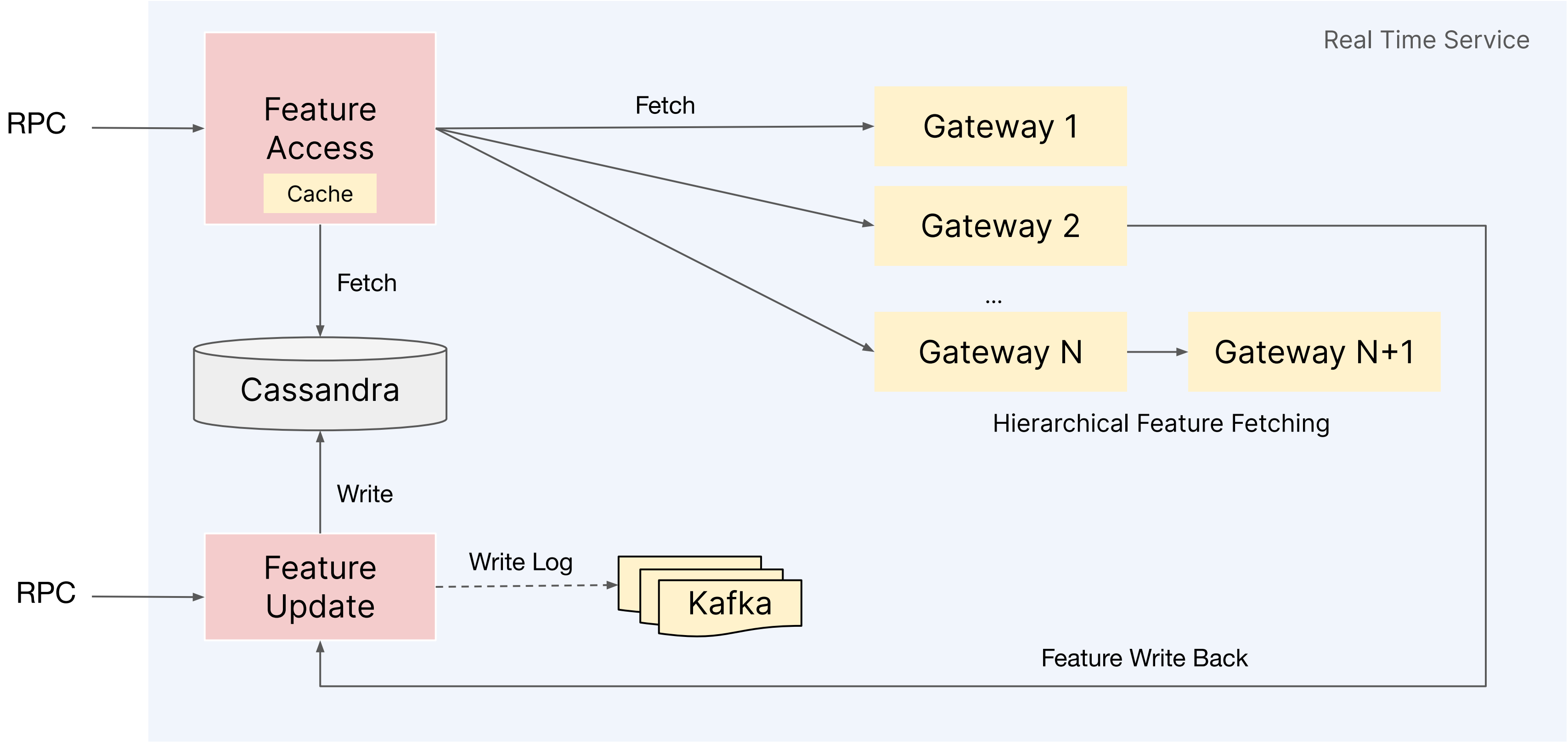


Aggregator Computation



*LTD= Lifetime to Date

Real Time Computation & Fetching



99.99% Availability

uGraph Platform

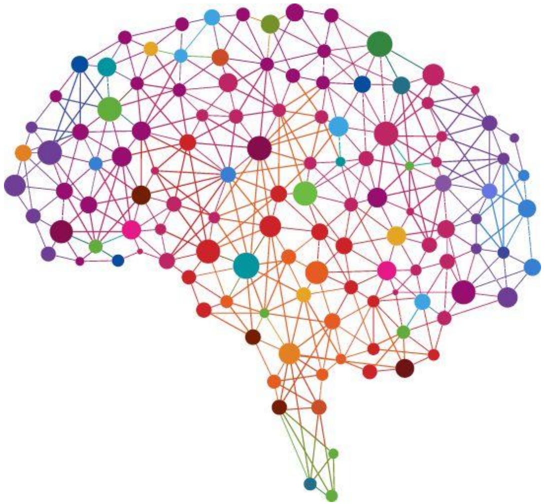
FEATURE STORE SUMMIT 2023

Knowledge Graphs - What and Why?

Knowledge graph is a knowledge base that uses a graph-structured data model to represent data

Why are knowledge graphs useful?

- Allows us to structure the unstructured data in the form of vertices and edges
- Entities (vertices and edges) have a definition and a context
- Entities can belong to disparate domains of knowledge connected through the underlying ontology/schema
- Easy to visualize, understand and query



"We are drowning in information and starving for knowledge" (Rutherford D. Rogers)

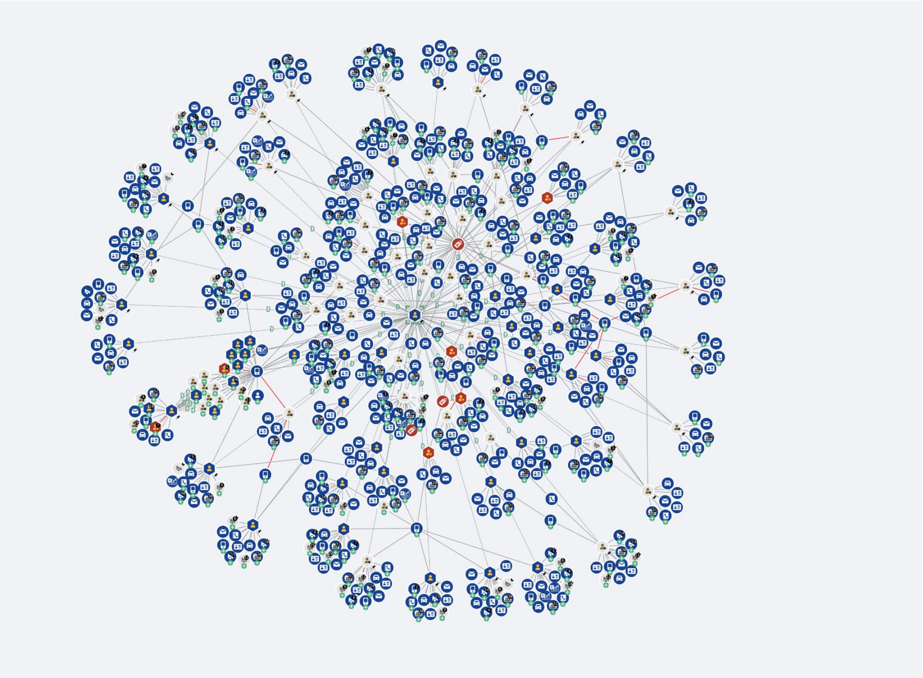
What is uGraph?

real-time and batch

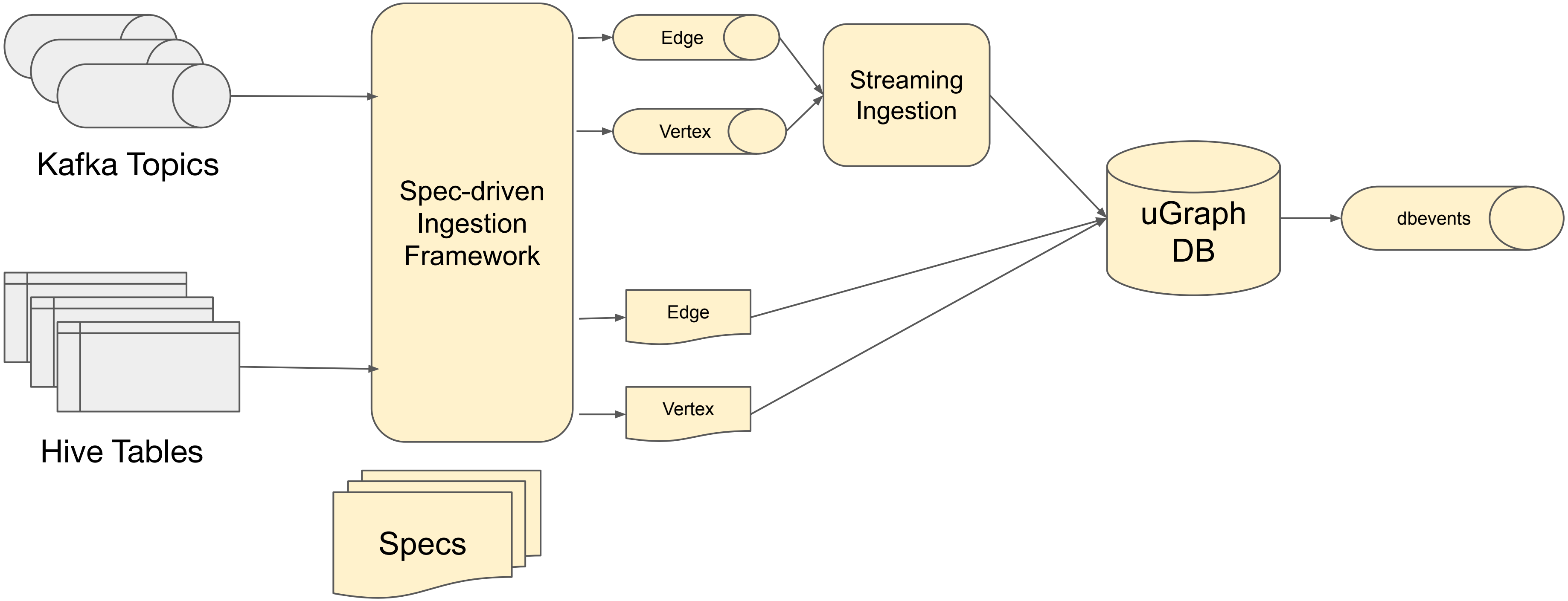
facts(name, email, trips taken, orders placed)

derive insights

Feature computation for ML



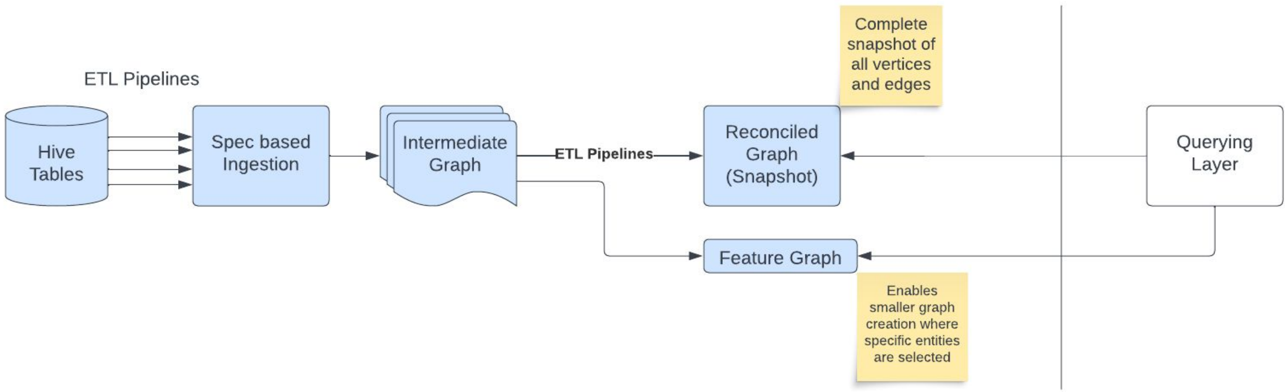
OLTP Ingestion Framework



OLAP Ingestion Framework

- Intermediate graphs are stored for recreation of features at a different time than current time.
- Also used for recreation of reconciled graph if something is amiss.

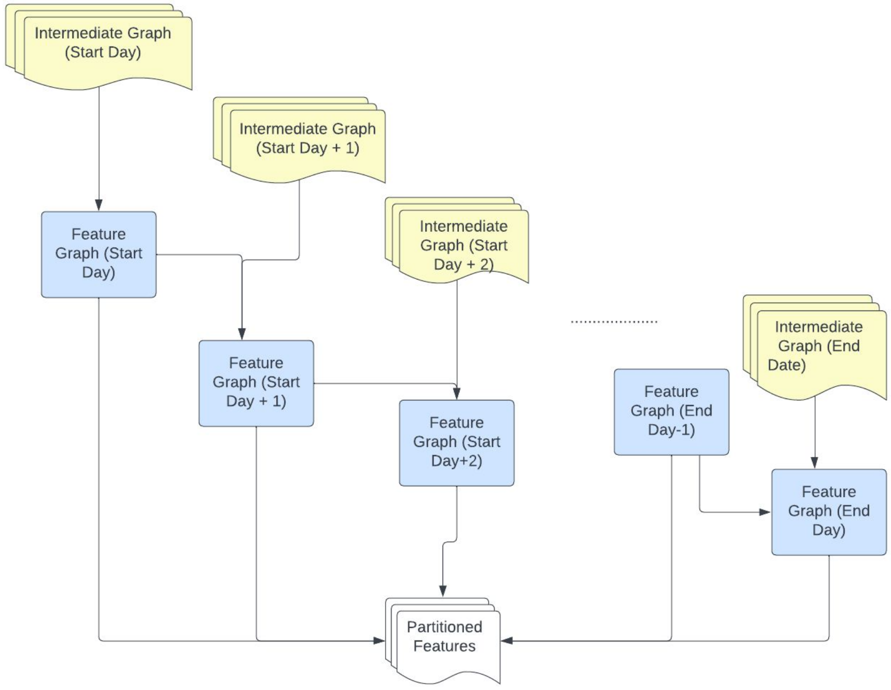
- Reconciled graph gives a wholistic view of entities in current time.
- Feature graph is a smaller graph with only interested entities for ML usecases, usually for past time.



FEATURE STORE SUMMIT 2023

Feature Computation Backfill for ML Training

- ML engineers often need to backfill features for point in time computation for model training.
- The feature computation backfill system enables us to do this seamlessly just through a spec



Graph Query Language (Cypher)

Cypher is supported as the graph query language. It is like SQL for graphs, and was inspired by SQL so it lets you focus on what data you want out of the graph

MATCH (u:User)-[hc:HAS_PHONE_NUMBER]->(p)<-[]-(u2)

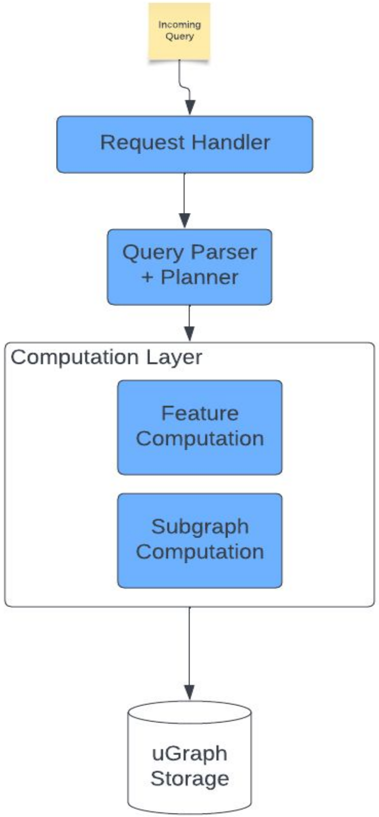
- Vertex with label and variable name
- Edge with label and variable name
- Vertex with only variable name (label is inferred)
- Edge without label or variable name (labels are inferred)

Graph Query Language (Cypher) Contd..

```
MATCH (u:User)-[hc:HAS_PHONE_NUMBER]->(t)<-[]-(u2)
WHERE u.uuid = ‘{{user_uuid}}’
WITH u.uuid as user_uuid,
u2.uuid as uuid
RETURN  user_uuid, count(DISTINCT uuid) AS
connected_users_share_same_phone_count
```

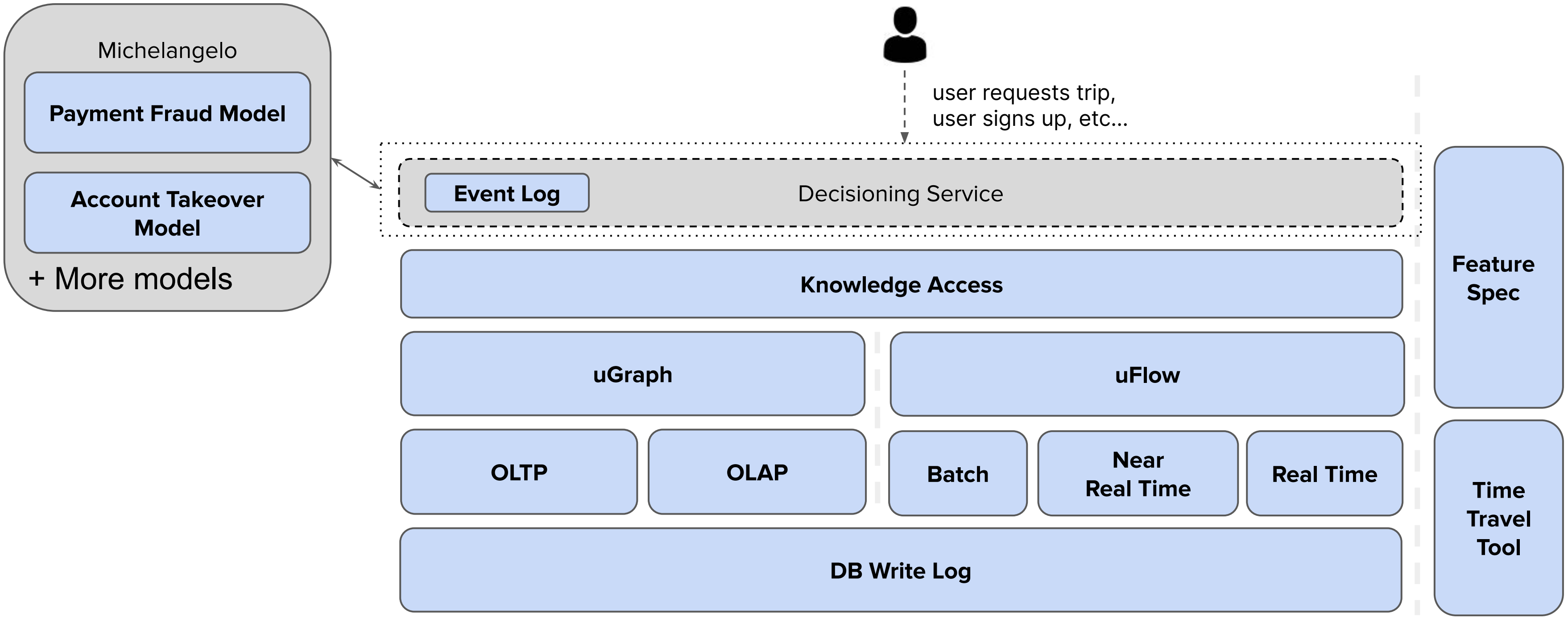
Querying Framework

- Requests are parsed and converted to a query plan for optimal execution.
- Only specific entities are fetched for feature computation(subgraph) and features are computed on top of the returned graph

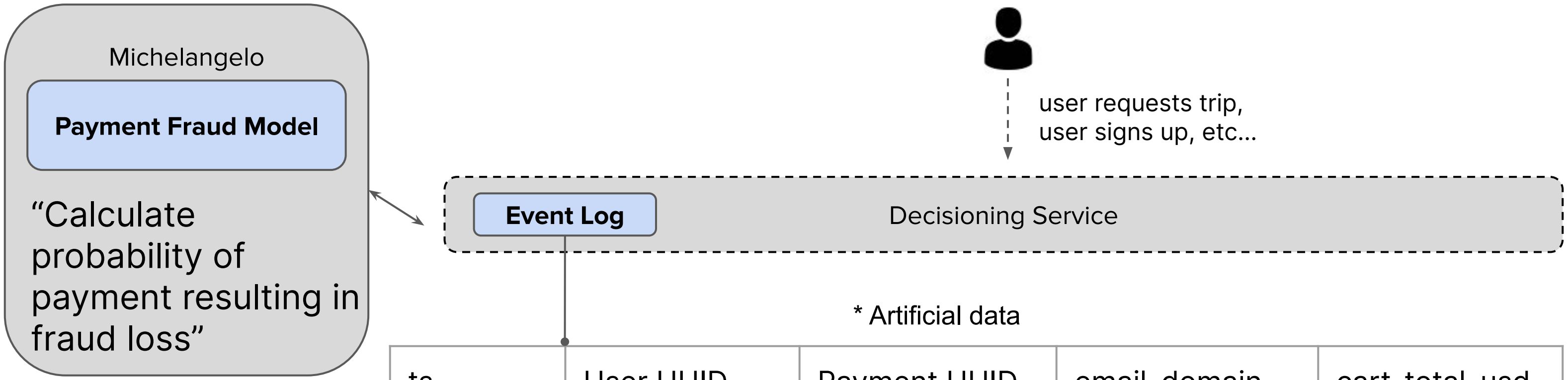


Machine Learning for Risk Assessment: Providing Real Time Predictions

Providing real time Predictive power - Architecture



Providing real time Predictive power - Training data



* Artificial data

ts	User UUID	Payment UUID	email_domain	cart_total_usd		label
27653	1234-5678	4532-7135	@good.com	$57.23	+	**Good**
31512	4321-8765	5612-8424	@bad.xyz	$323.76		**Fraud**

Basis Data

Providing real time Predictive power - Naive model training

ts	User UUID	Payment UUID	email_domain	cart_total_usd
27653	1234-5678	4532-7135	@good.com	$57.23
31512	4321-8765	5612-8424	@bad.xyz	$323.76

label
Good
Fraud

What might the model learn?

What can we do to make it better?

Enrich model with more features! Starting with batch...

Machine Learning for Risk Assessment: Batch Feature Backfill

Feature Backfilling - Basis data entities

ts	User UUID	Payment UUID	email_domain
27653	1234-567 8	4532-7135	@good.com
31512	4321-876 5	5612-8424	@bad.xyz

+

label
Good
Fraud

Primary entities:
The fields in a request that identify a primary actor involved in request

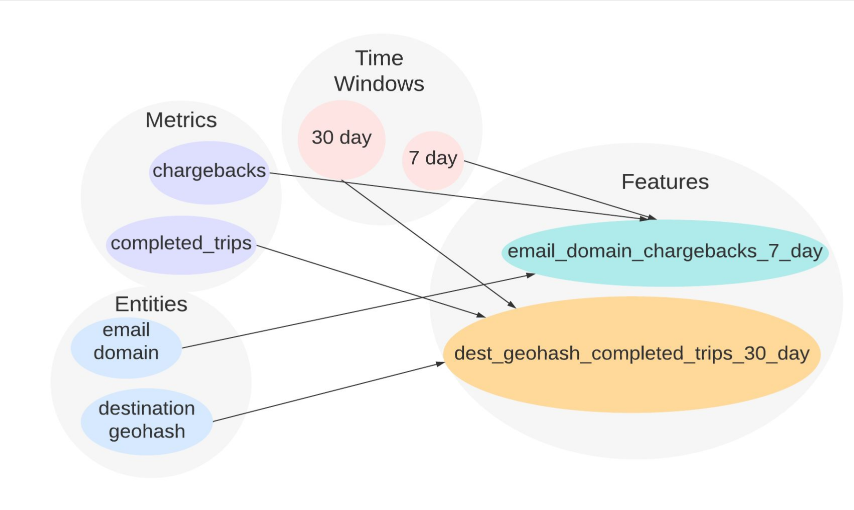
Secondary entities:
Fields in a request shared among multiple primary actors

Batch Feature Backfilling

ts	email_domain	Payment UUID
27653	@good.com	4532-7135
31512	@bad.xyz	5612-8424

Basis Table

* Artificial data



Entity Feature Tables

Expanded Training Set

Basis Table + $e \cdot m \cdot w$ = **1000s** of additional features

Computed Entity Features

Batch Feature Backfilling - Gaps

... but when do you need realtime feature engineering?

ts	user_trips_24h	user_last_known_phone_number	email_domain	users_linked_by_phone	banned_users_linked_by_phone
27657	7	867-5309	@new.tld	10	9

* Artificial data

Ask questions like...

1. count user trips in last 24 hours
2. user_last_used_phone_number
3. new entity values observed

   (graph)
4. count users linked by phone #
5. count banned users linked by phone #

Any feature engineering on primary entities

Problem types:

- Account Takeover
- New User Fraud
- Marketplace Abuse
- Safety

Machine Learning for Risk Assessment: Streaming Feature Backfill

FEATURE STORE
SUMMIT
2023

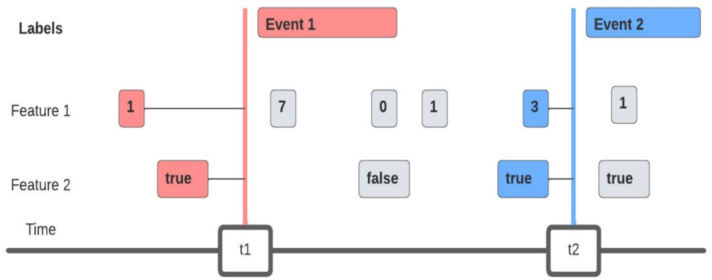
Streaming Feature Backfill - Attribute & Aggregation Feature

For **attribute** features, we can use **temporal join** to stitch their values to our training set.

date	ts	email_domain	email_domain_**batch**_trips_1d	email_domain_**nrt**_trips_1d
2023-09-01	27657	@new.tld	0	**20**

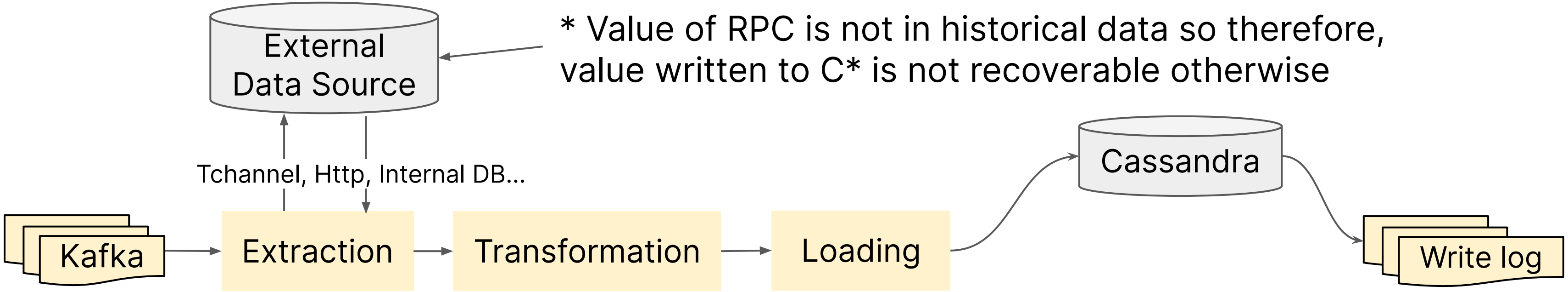
date	ts	**user_last_known_phone_number**
2023-09-01	27657	**867-5309**

NRT **aggregation** feature can help us fill in the gaps that batch computation would miss!



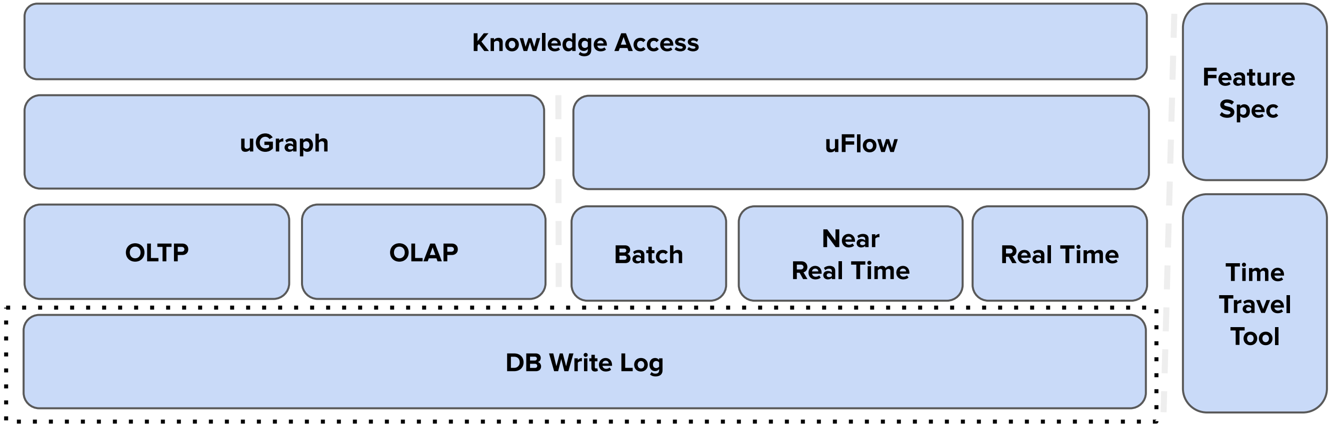
For both attribute and aggregation features, **data must exist in offline storage**.

Streaming Feature Backfill - Persisting NRT changes to DB write log



DB Write Log persists all changes to C* features

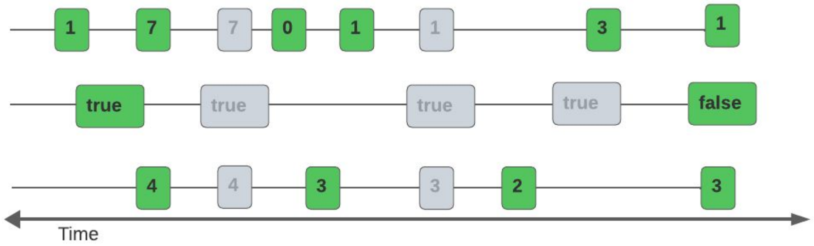
Write log is transformed into a change log



Streaming Feature Backfill - Transforming the write log into a change log

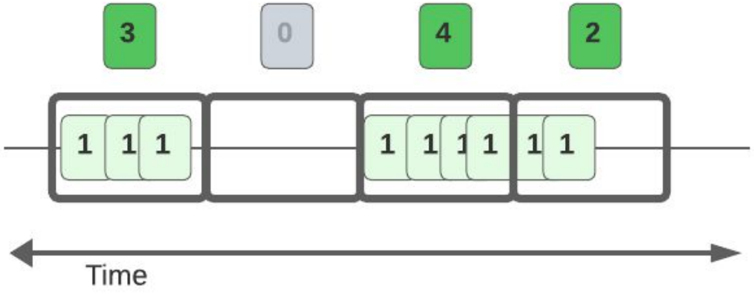
For streaming attribute features, only change data is needed

❖ Drop redundant writes



❖ Optimized Change Data Capture solution
  ➢ Much smaller HDFS storage costs

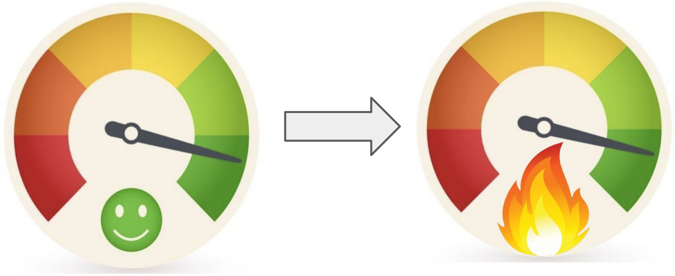
For streaming aggregation (velocity) features, all writes are needed.



Pre-aggregation:
- Reduce qps to C*
- Reduce change log storage
- When latency requirement can be relaxed

Streaming Feature Backfill - Using the Time Travel Tool

- ❖ Create Basis Table
- ❖ Specify join keys & feature groups to time travel
- ❖ Tools creates ML training data w/ batch & streaming features



ML with basis + batch features

ML with basis + near real time features

Supports:

- Time travel on DB change log (for RPC features)
- Time travel on hive ingested kafka data

End result:

Input: Basis Data → **Time Travel Tool** → Output: Basis Data | Batch Backfilled Feature Values | Near Real Time Backfilled Features

10s - 100s of features

1000s+ features

Thank you