Sequence Features at Coinbase

Joseph McAllister, Senior ML Platform Engineer, Coinbase

Organized by HOPSWORKS

Agenda

- Overview of Coinbase's ML use-cases and Feature Platform Scale
- Streaming Pipelines and Sequence Features
- Ongoing Work

ML Feature Platform Applications at Coinbase

Predictive ML

- Fraud Detection
- Recommendation Systems
- Customer Support
- Compliance
- ...

Non-ML

- Decision Engines
- Reverse ETL & Online Data Serving
- Batch Data Pipelines

Feature Platform Scale

- ~100 Feature Platform internal users
  - Mix of MLE's, Analysts, Data Scientists, Software Engineers
- >30 use-cases
- >9k features
- >300 streaming workloads
  - >2x increase in last 12 months

Sequence Features


```
{
 "event": "Sign In"
 "timestamp": "2024-12-06 17:09:12"
 "metadata": {
   "device": "ios"
 }
}
```

```
{
 "event": "Sell"
 "timestamp": "2024-12-06 01:10:42"
 "metadata": {
   "currency": "ETH"
   "amount": 1.4
 }
}
```

```
{
 "event": "Buy"
 "timestamp": "2024-12-06 17:10:19"
 "metadata": {
   "currency": "BTC"
   "amount": 0.12
 }
}
```



```
{
 "event": "Sign In"
 "timestamp": "2024-12-07 14:09:56"
 "metadata": {
   "device": "android"
 }
}
```

```
{
 "event": "Buy"
 "timestamp": "2024-12-07 14:30:87"
 "metadata": {
   "currency": "ETH"
   "amount": 2.8
 }
}
```

```
{
 "event": "Crypto Send"
 "timestamp": "2024-12-07 14:32:51"
 "metadata": {
   "currency": "ETH"
   "amount": 2.8,
   "chain": "BASE"
 }
}
```

Motivation for Sequence Features

- Intuition
  - Richer than rolling window aggregates
  - Directly incorporate temporal nature
- Modeling
  - Generate generic user and asset embeddings (recsys)
  - Incorporate sequence features alongside traditional features (fraud detection)

Designing Sequence Features

- **Event selection**: <100 event types per sequence
- **Sequence length**: up to 1000 most recent events at read-time
- **Lookback window**: highly dependent on use-case, ranges from hours to lifetime
- **Freshness**: ranges from daily to sub-second
- **Offline read-time latency**: <2 hours, faster is better for iteration speed
- **Online read-time latency**: <100ms p99 latency

User Experience

Registering data sources

Python

```python
class SignIn(CoinbaseEvent):
    name = "sign_in"
    version = 1
    schema = StructType(
        [
            StructField("device", StringType(), True),
             ...
        ]
    )

class Buy(CoinbaseEvent):
    name = "buy"
    version = 1
    schema = ...
```

User Experience

Selecting Events

```python
sign_in_buys_sequence_ds = get_stream_source([SignIn, Buy], name="sign_in_buys_sequence")
```

- 1 LOC
  - Union multiple topics
  - Merge schemas
- Abstract away concept of backfills and offline event tables
- Encourage experimentation and rapid iteration

- Schema
  - user_id: StringType
  - event_name: StringType
  - timestamp: TimestampType
  - metadata: StructType

User Experience

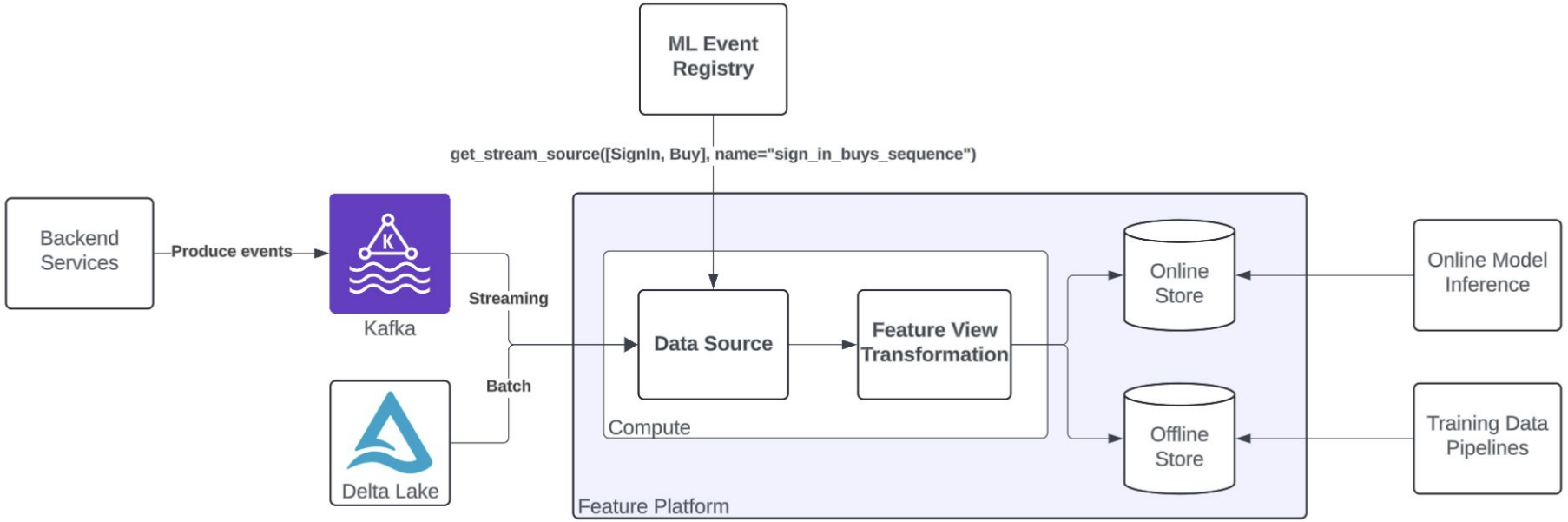
Derived Features

Python

```python
def add_day_tokens_to_sequence(unix_timestamps, events):
    """Return a list with a <DAY> token delineating days between each event"""
    current_day = None
    result = []
    for i in range(len(events)):
        day = datetime.fromtimestamp(int(unix_timestamps[i])).date()

        # Prepend sequence with <DAY>
        if current_day is None:
            result.append("<DAY>")
            current_day = day

        # Handle new or missing days
        while current_day < day:
            result.append("<DAY>")
            current_day += timedelta(days=1)

        result.append(events[i])

    return result
```

```python
@on_demand_feature_view(
    name="user_sequence_lifetime_stream_transformed",
    sources=[user_sequence_lifetime_stream],
    mode="python",
    schema=[
        Field("events_with_days", Array(String)),
    ],
)
def user_sequence_lifetime_stream_transformed(user_sequence_lifetime_stream):
    return {
        "events_with_days": add_day_tokens_to_sequence(
            user_sequence_lifetime_stream["event_times"],
            user_sequence_lifetime_stream["events"],
        ),
        ...
    }
```

Zooming Out


ML Event Registry
get_stream_source([SignIn, Buy], name="sign_in_buys_sequence")
Backend Services
Produce events
Kafka
Streaming
Batch
Delta Lake
Data Source
Feature View Transformation
Compute
Online Store
Offline Store
Feature Platform
Online Model Inference
Training Data Pipelines

Streaming Pipeline Optimizations

Stateless vs. Stateful Computation

- Traditional feature aggregations (sum, count, etc.) have an unbounded # of events per entity in the window ⇒
  - Stateful stream computation and compaction are needed to bound online read latency
- Sequence features (last(n)) have a strict bound on # of events per entity in the window ⇒
  - Aggregate at online read time with bounded read-time latency
  - *Stateless streaming model*
    - Fresher features (no streaming shuffle/aggregate)
    - Cheaper streaming compute and less tuning required

Ongoing Work

Ongoing Work

- Data Quality
  - Stronger contracts between data producers and consumers
  - Further observability improvements around feature drift and null rates
- Feature Lifecycle Management
  - Ease of use has led to many more feature pipelines
- Adoption of Databricks Real-Time Mode

Thanks!