Use Apache Beam to build Machine Learning Feature System at Affirm

Hao Xu

BEAM SUMMIT
NYC 2023

Use Apache Beam To Build Machine Learning Feature System At Affirm

- Hao Xu

01

ABOUT ME

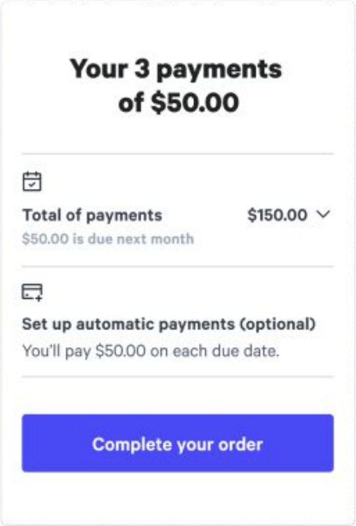
Earnest -> Fast -> Affirm -> JP Morgan & Chase

TABLE OF CONTENTS

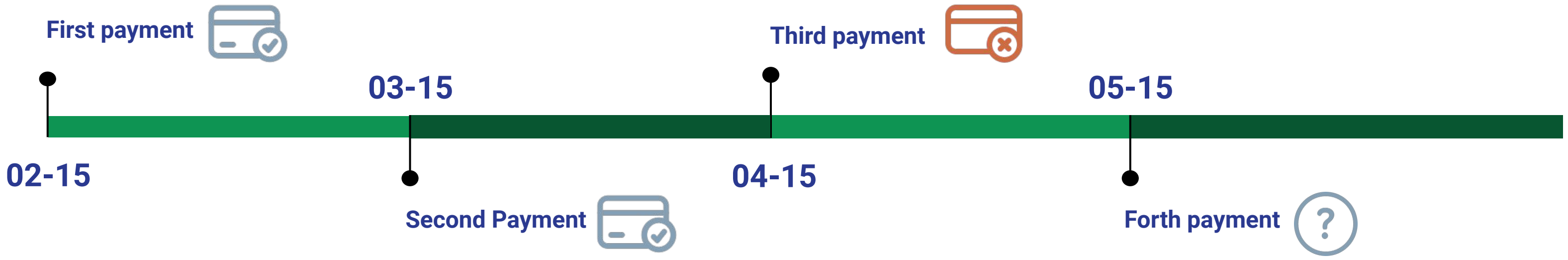
Background

- BNPL
- Machine learning feature store
- Streaming and Batch Compute Platform

The Story of BNPL



The Story of BNPL

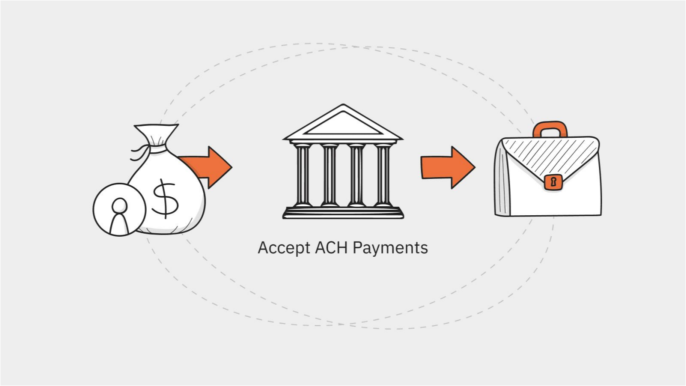


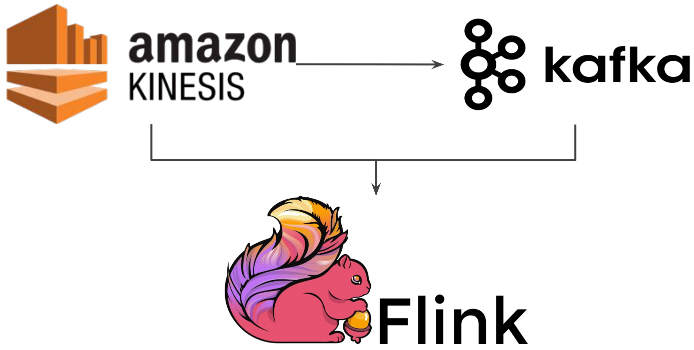
If a user failed the third payment, is it likely that they will also fail the fourth one?

Has the user failed to make a loan payment, and if so, have we identified the issue? Should we approve another loan for them?

Payment flow

The payment data was processed in batches, resulting in a delay of a couple of days. Utilizing stream data can help prevent such delays in the future.





Feature Store

Figure 1. A feature store is the interface between feature engineering and model development.

Pain Points

Pain Points

Development Velocity

Slow backfilling of stream features.
Excessive code required to define a feature.

Variety

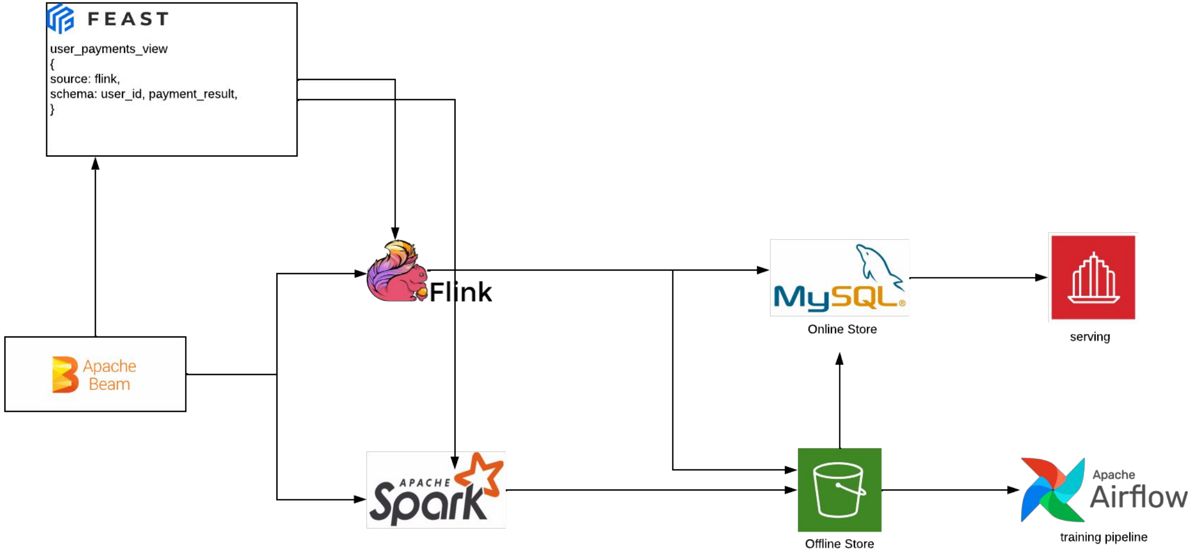
Inability to join two streams from Kinesis together, which is typically required for stateful processing.

Visibility

Lack of registry to quickly lookup data sources, features and metadata.

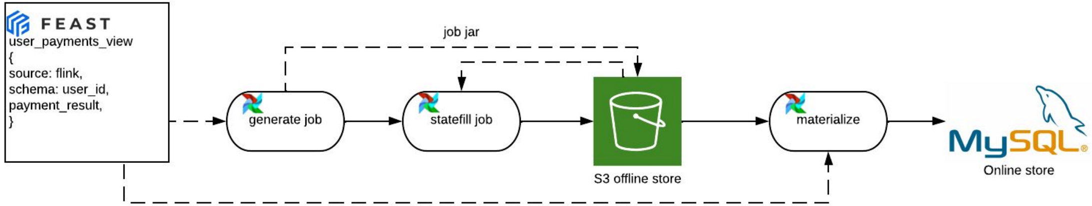
Solution

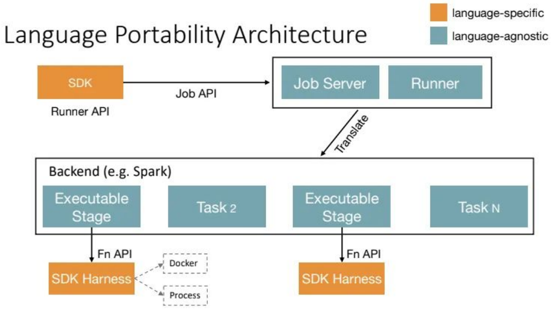
MLFS Architecture



Complex of Backfilling

Backfilling is the process to backfill a feature data to the historical point in time






```
spec:
  containers:
    - name: spark-kubernetes-executor
      volumeMounts:
        - name: beam-data
          mountPath: /opt/apache/beam/
  initContainers:
  - name: init-beam
    image: apache/beam_python3.7_sdk
    command:
    - cp
    - /opt/apache/beam/boot
    - /init-container/data/boot
    volumeMounts:
    - name: beam-data
      mountPath: /init-container/data
  volumes:
  - name: beam-data
    emptyDir: {}
```

Unified Transformation Interface

```
class UnifiedTransformer(Transformer[beam.PCollection, beam.PCollection]):

    @property
    def window(self) -> beam.WindowInto:
        return self._window

    @property
    def event_transform(self) -> beam.PTransform:
        return self._event_transform

    @property
    def aggregator(self) -> beam.PTransform:
        return self._aggregator

    def run(self, inputs: beam.PCollection) -> beam.PCollection:
        if self.feast_context.runner == Runner.flink:
            if self.window:
                inputs = inputs | self.window
            return (
                inputs
                | self.event_transform.with_output_types(Tuple)
                | self.aggregator.with_output_types(Tuple)
            )
        elif self.feast_context.runner == Runner.spark:
            return (
                inputs
                | self.event_transform.with_output_types(Tuple)
                | self.aggregator.with_output_types(Tuple)
            )
        else:
            raise ValueError("Unsupported runner: {}.".format(self.feast_context.runner))
```

Unified Transformation Interface

```python
@stream_feature_view(
    entities=[entity_registry['user_ari']],
    ttl=timedelta(days=0),
    schema=[
        Field(name="user_ari", dtype=String),
        Field(name="timestamp", dtype=UnixTimestamp),
        Field(name="latest_payment_fail", dtype=UnixTimestamp),
        Field(name="latest_payment_fail_ach_nsf", dtype=UnixTimestamp),
    ],
    online=True,
    source=user_payment_fails_stream_source,
    timestamp_field="timestamp",
    tags={},
    mode="flink",
)
def user_last_payment_fail(feast_context: FeastContext, inputs: PCollection) -> PCollection:
    transformer = UnifiedTransformer(
        feast_context=feast_context,
        aggregator=LatestFeatureAggregator(feast_context, 'timestamp'),
        event_transform=extract_payment_fail_data,
    )
    return transformer.run(inputs)
```

Outcome

Performance **boost**

80%

Backfilling time

Backfilling time improved by 80%

60%

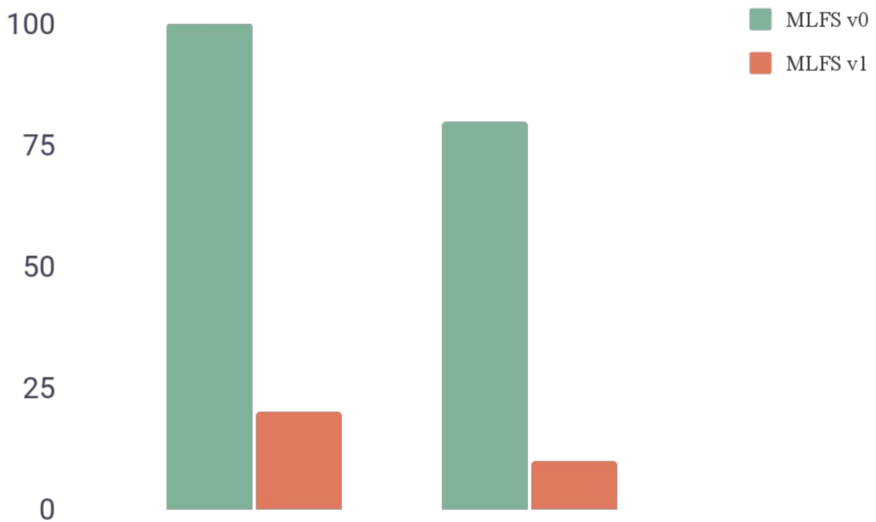
Code lines

Reduced 100+ lines to 20+ lines

40%

Registry

200+ data sources
100+ features



The time spent to backfill features for feature *time_since_user_checkout* and *tiem_since_user_last_payment_failure*

Future improvement

1. OOTB transformation interface
2. Transformation framework
3. Improvement on Beam Spark Runner