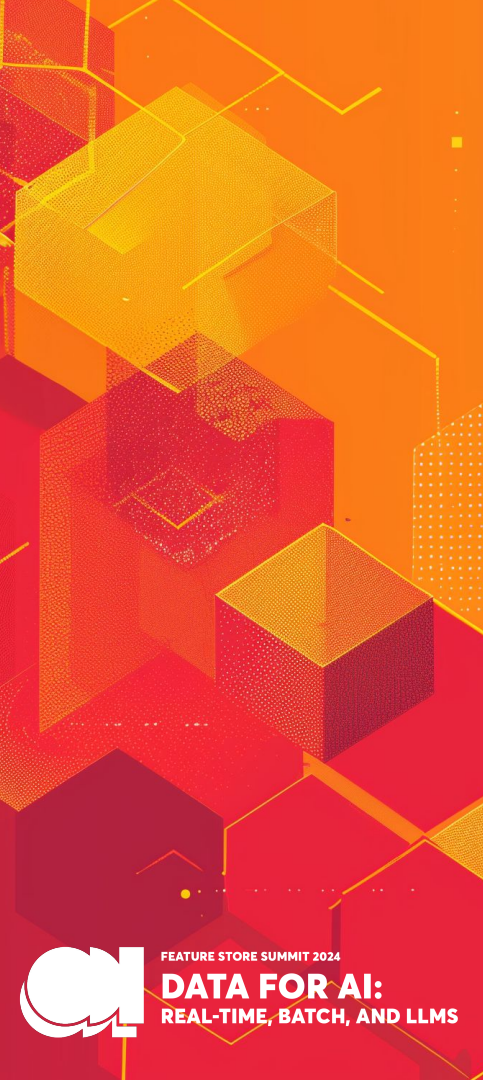


Chronon, Airbnb's open source feature engineering framework

Pengyu Hou, Software Engineer, Airbnb

Organized by HOPSWORKS

Agenda

- Why Chronon?
- What is Chronon?
- Point in time correct
- Life before Chronon
- Architecture
- Examples
- Takeaways + Call to action

Why Chronon?

Why Chronon?

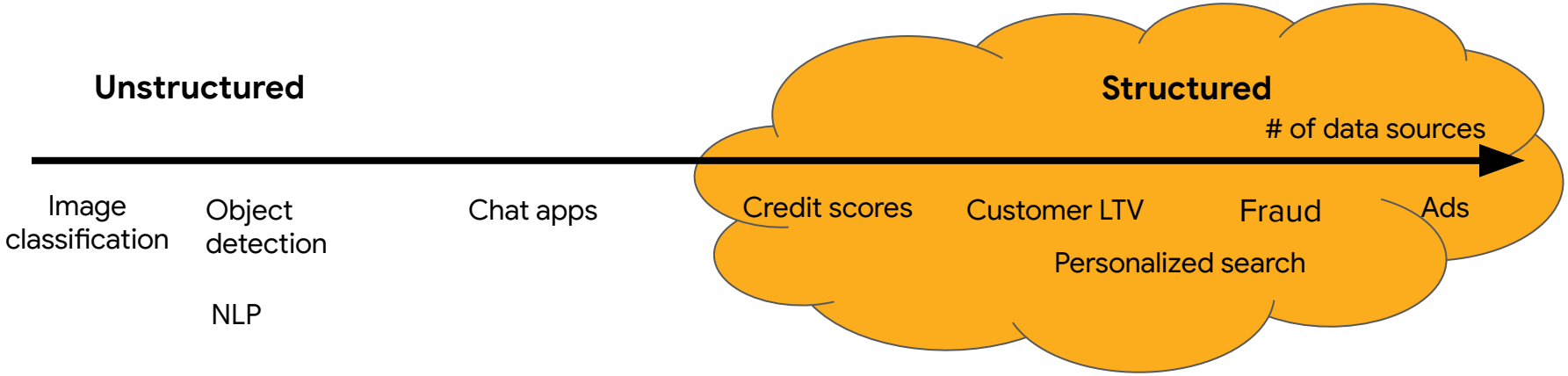
What are some common challenges?

- Feature definition in multiple places
- Online offline inconsistency
- Slow backfills ⇒ slow iteration speed
- Repetitive glue code for the pipelines
- Duplicate work across multiple orgs

What is Chronon?

Chronon is an open-source platform enabling ML practitioners to efficiently develop, deploy, manage, and monitor ML data pipelines.

ML Applications



- Most of the data is available at once: full image
- Features are *automatically* extracted from *few (often one)* data stream:
  - words from a text
  - pixels from an image

- Data arrives steadily as user interacts with the platform
- Features extracted from *many* event streams:
  - logins
  - clicks
  - page views, etc
- **Iterative *manual* feature engineering**

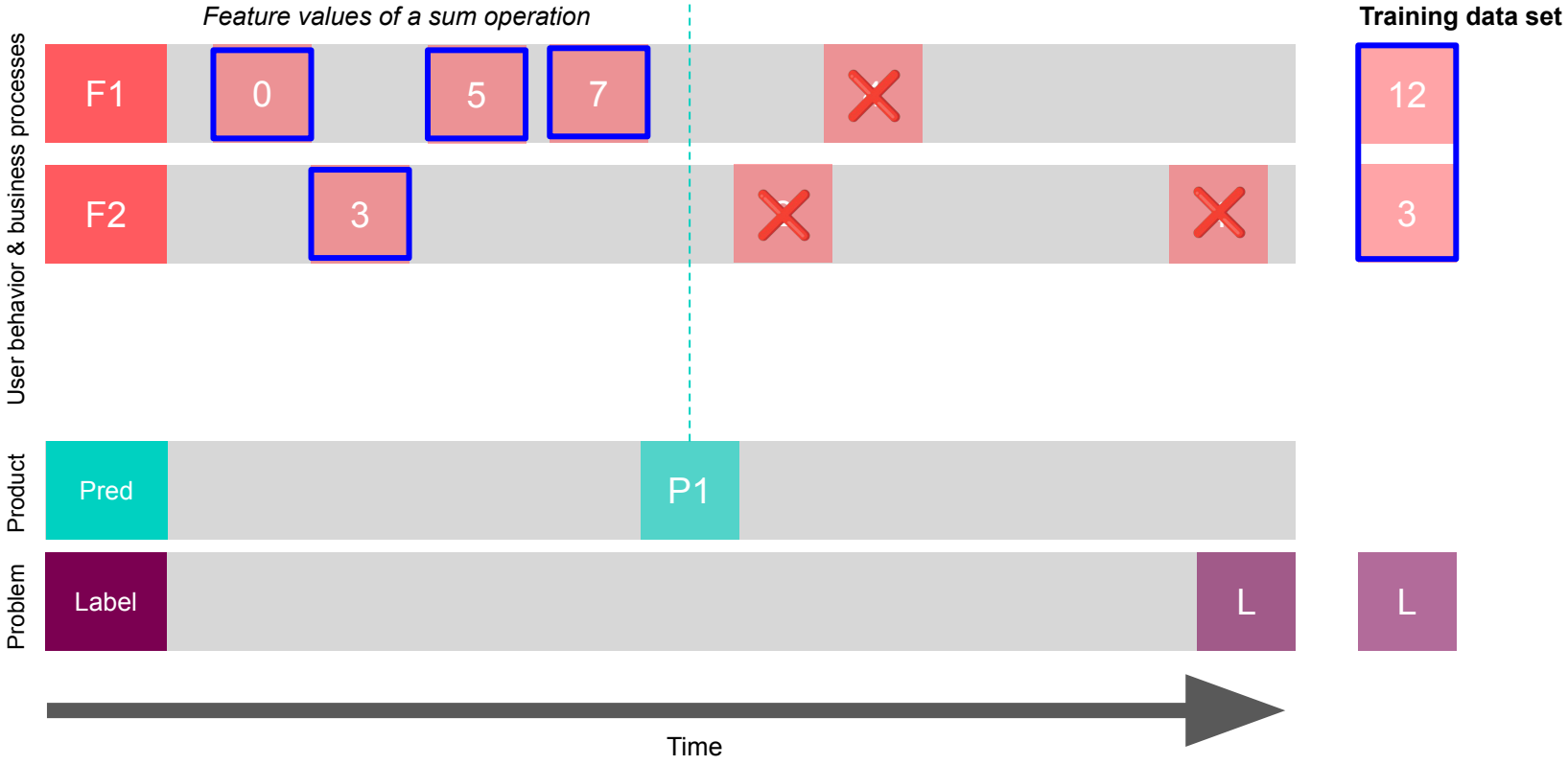
Journey to open source

Journey to open source

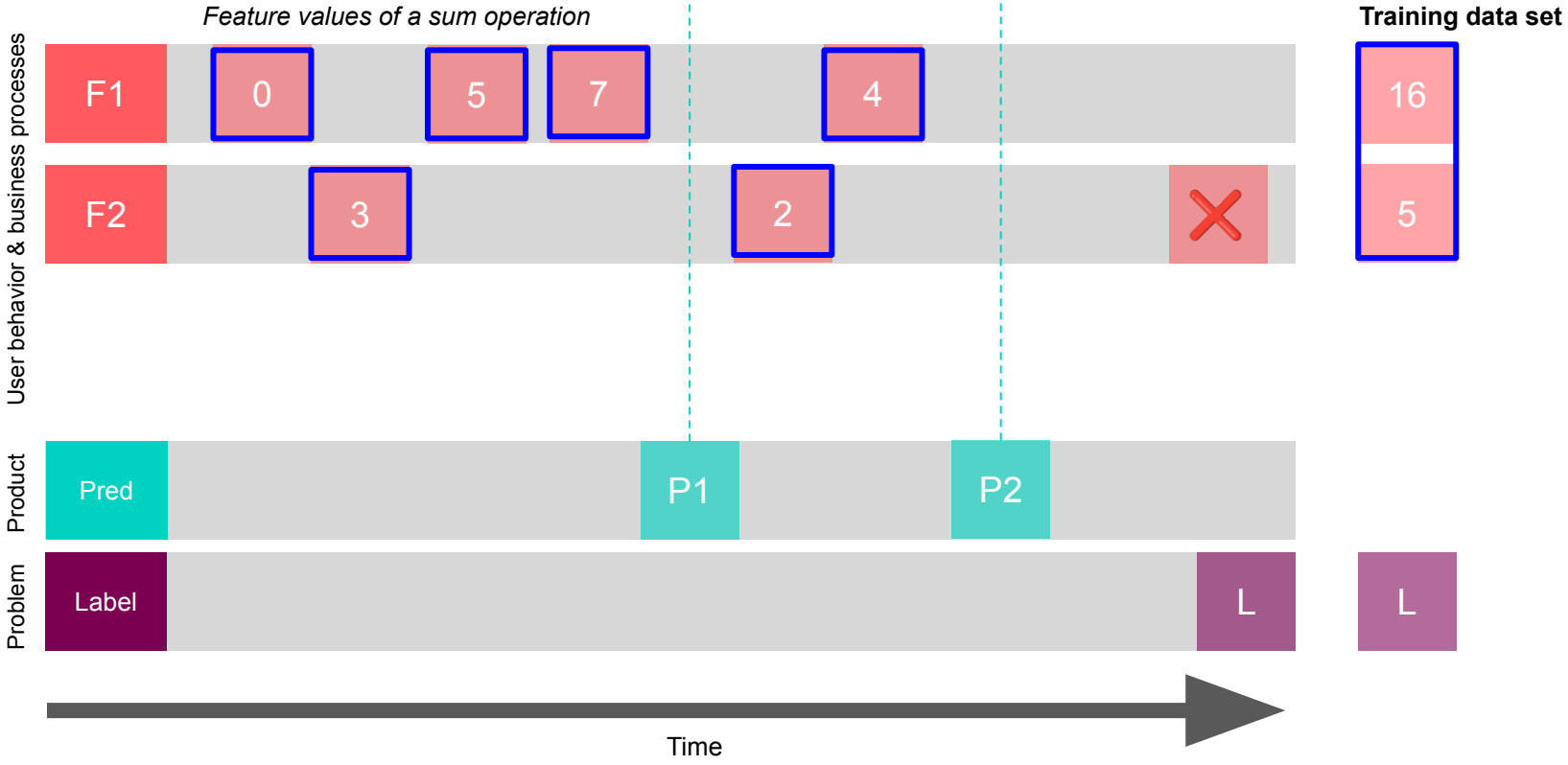
- 2017-2022: iterated over four internal versions
- 2022: private beta partnered with Stripe
  - Fully adopted within Stripe
  - They are also in this summit
- 2024: announced open source
  - Stripe is the co-maintainer

What is Point in Time Correct? (PITC)?

Point in time correct feature values



Point in time correct feature values



Point in time correct matters

What are some common symptom?

- Works well in local
- Performs poorly in prod

Point in time correct matters

What are some common symptom?

- Works well in local
- Performs poorly in prod

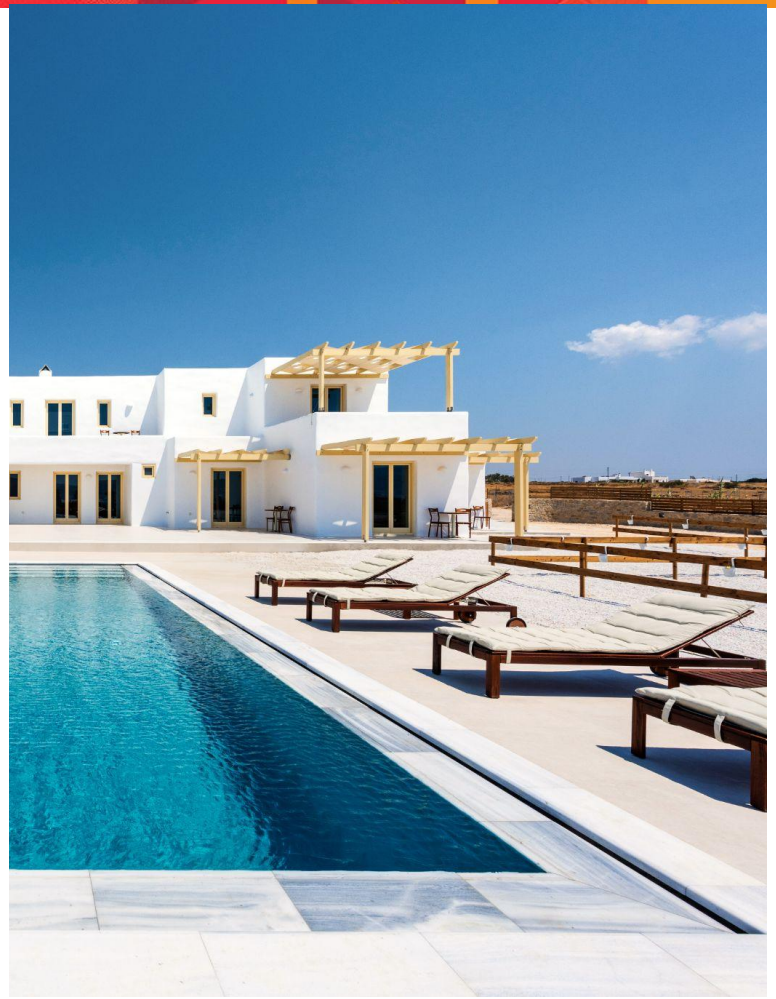
Data leakage example:

A house price prediction model can be based on:

- Rooms, location, size, age, etc

But not:

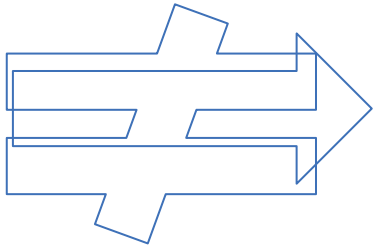
- Offers received
- Buyer's info



What was life like before Chronon?

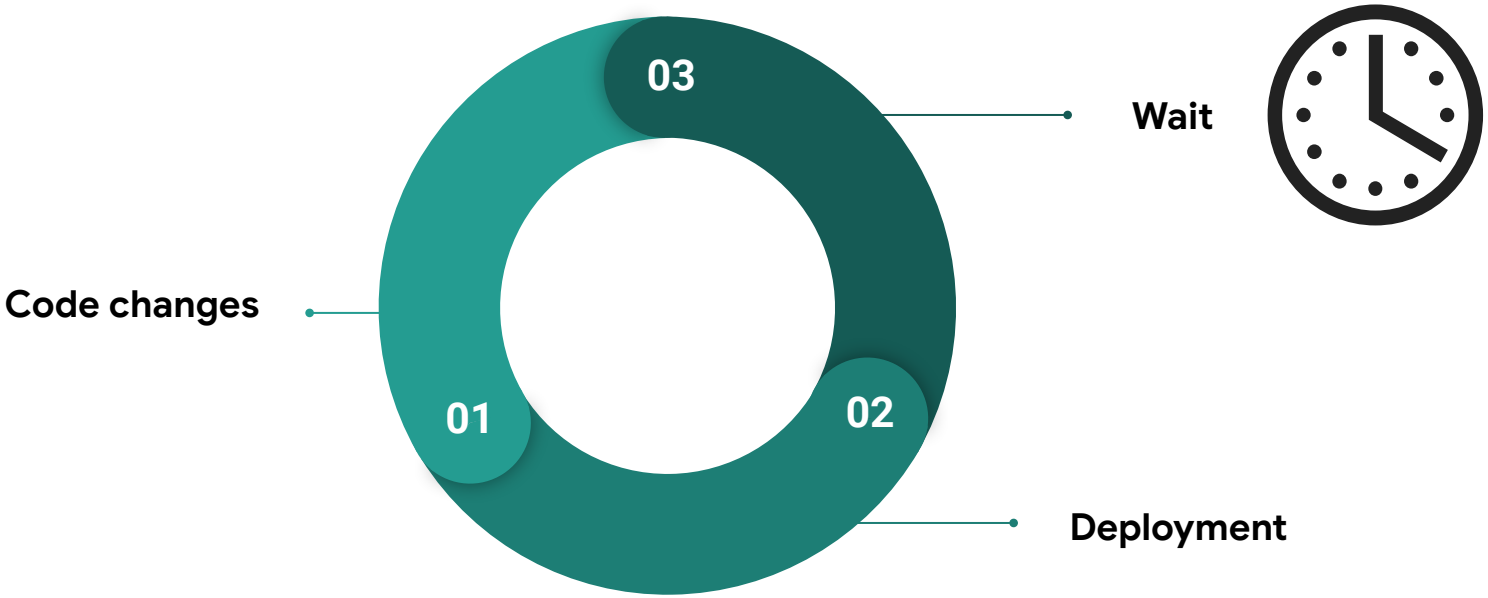
Replicate offline -> online

SQL



Program Language of the service

Log and wait



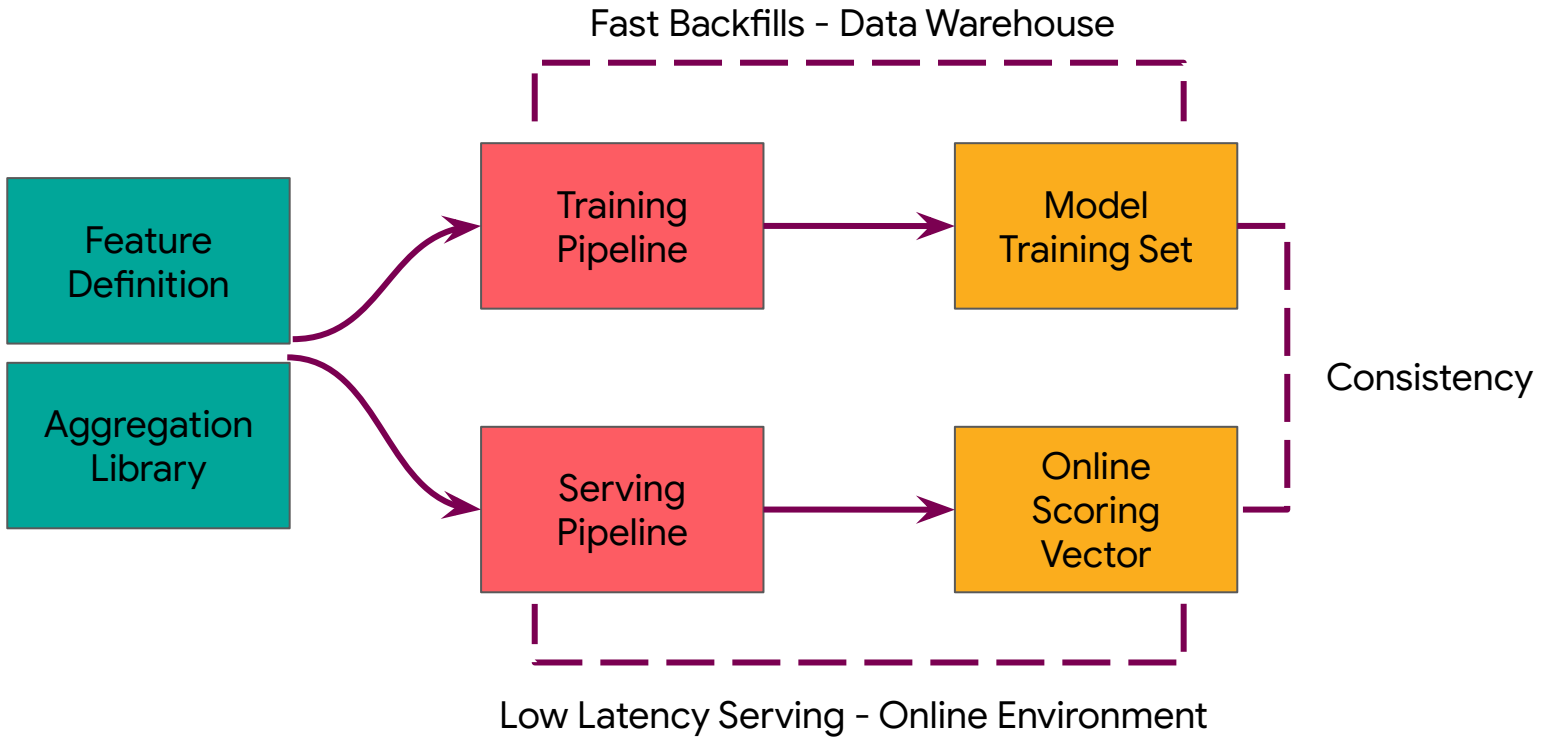
How can Chronon address consistency without waiting for months?

Chronon Way

- Single Python config file
- Same Scala compute engine for both envs
- Take care of the infra orchestration

Chronon Architecture

Feature Management System



Examples

Feature definition (in Python!)

```
v1 = GroupBy(
    source = source,
    keys = ["user_id"],
    aggregations = [Aggregation(
        name = "page_view_sum",
        operation = Operation.SUM,
        windows = [
            Window(length = 7, timeUnit = TimeUnit.DAYS),
            Window(length = 14, timeUnit = TimeUnit.DAYS),
            Window(length = 1, timeUnit = TimeUnit.MONTH),
            Window(length = 1, timeUnit = TimeUnit.YEAR),
        ],
        inputColumn = "page_view_count"
    )]
)
```

GroupBy: Aggregation over a single **Source**

Aggregation: Operation + Windows

Operation: functions i.e. SUM, AVG, LAST

Windows: for time-series data

Joins

- **Join**
  - What is the <aggregation statistic> for <key> from <time A> to <timestamp>?
- **Challenges**
  - Data skew e.g. bots
  - Computing with midnight accuracy can be faster

```
v1 = LeftOuterJoin(
    left=leftSource, # (user_id, timestmaps)
    rightParts=[
        JoinPart(
            group_by=page_views.v1,
            keyMapping={'user_id':'host_id'},
        ),
        JoinPart(
            group_by=profile_change.v2,
        ),
    ],
)
```

Fun fact

One user achieved:
250 lines of code → 1000 features

New Functionalities

Derivations

- **Why?**
  - Computing acceptance ratio = accept_sum / requests_sum
- **Challenges**
  - Uniform API - online & offline
  - Online Latency

```
v1 = Join(
    # it supports group by level as well
    left=...,
    right_parts=[...],
    derivations=[
        Derivation(
            name="acceptance_ratio",
            expression="accept_sum /
            requests_sum"
        ),
        ...
    ]
)
```

Real time chaining features

- **Why?**
  - Computing average price of ice-creams a user viewed in the last 14 days
  - Require transformations and aggregations along with denormalization
- **Challenges**
  - online & offline consistency
  - Online Latency

```
group_by = GroupBy(
    name="enriched_ice_cream_info",
    sources = join_source,
    keys = ["user"],
    aggregations = [
        # creates an aggregate of last 100 ice creams viewed in a 14 day window
        Aggregation(
            operation = Operation.LAST_K(100),
            input_column = "price",
            window = [Window(14, TimeUnit.DAYS)]
        )
    ],
    derivations = [
        Derivation(
            name = "avg_price_last_14d",
            expression = "aggregate (price_last100_14d, 0, (acc, x) -> acc + x) / size(price_last100_14d)"
        )
    ]
)
```

Key Takeaways

Key takeaways:

- Provides declarative language to define features once for both online and offline features.
- Lambda architecture to achieve online offline consistency and low latency serving
- Windowed operation supported
- Operations: currently 15 aggregation operation
    - Can be extended easily

Call to actions

Call to actions:

- Star the repo and fork it: https://github.com/airbnb/chronon
- Kick starter: report issues or ideas to git issues
- Discord: https://discord.gg/GbmGATNqqP

Additional Resources

Additional Resources:

- Chronon — A Declarative Feature Engineering Framework
- Chronon, Airbnb's ML Feature Platform, Is Now Open Source
- Shepherd: How Stripe adapted Chronon to scale ML feature development
- https://www.chronon.ai/
- https://github.com/airbnb/chronon/

Thank you