CLOUDERA

Streaming for the masses with FlinkSQL

Márton Balassi, Mátyás Őrhidi

About Us

Marton Balassi

@MartonBalassi, mbalassi@cloudera.com

- Apache Flink PMC member since 2014
- Spent 3 years at Cloudera as a Solutions Architect,
  worked with ~50 customers directly
- Currently leading the Streaming Analytics (Apache Flink)
  Engineering team at Cloudera

Matyas Orhidi

linkedin.com/in/matyasorhidi

- Joined Cloudera in 2016
  - Premier Support
  - Professional Services
  - Engineering
- Worked with strategic accounts
  - Deutsche Telekom, Deutsche Bank, Lufthansa, IQVIA
- Architect & Security SME
- Co-leading the CSA Engineering team with Marton

Agenda

The Power of Flink

- Noteworthy users
- Known use cases
- Basic Concepts
- Flink APIs

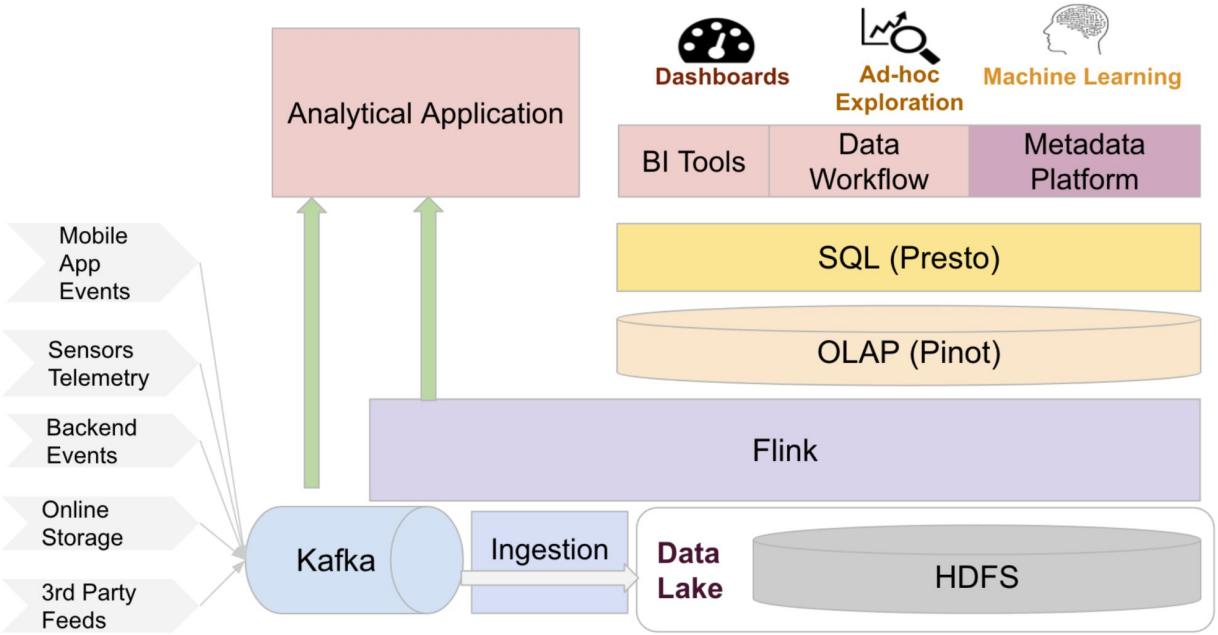
The Simplicity of SQL

- Traditional vs Streaming SQL
- Streams, Dynamic Tables & Continuous Queries
- Queries and Time

Users and use cases

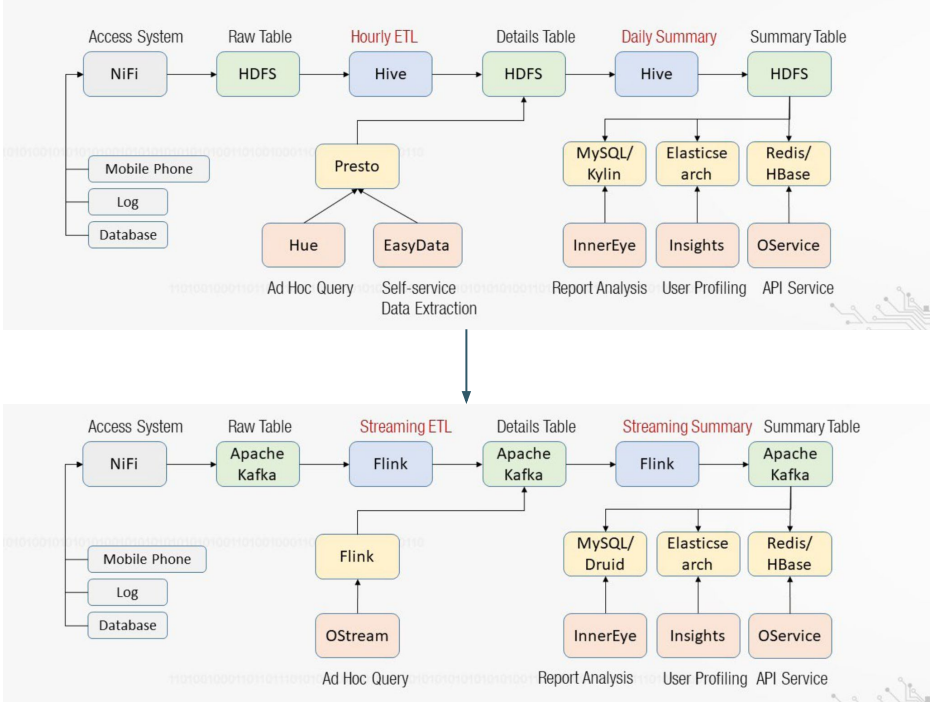
Powered by Flink

Uber’s data infrastructure



https://arxiv.org/pdf/2104.00087.pdf

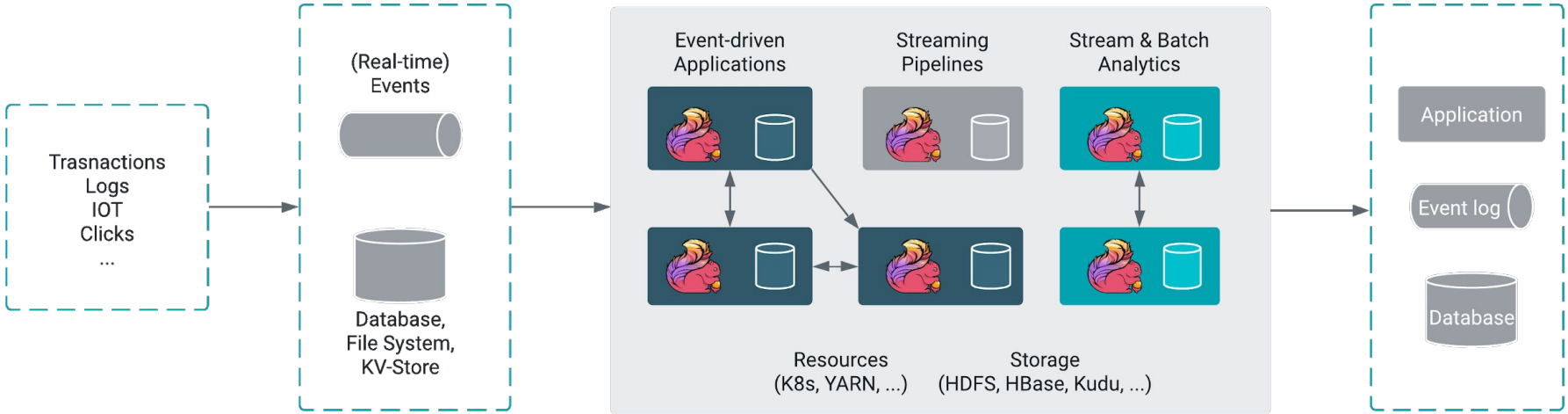
OPPO's Real Time Data WareHouse



https://resource.alibabacloud.com/whitepaper/real-time-is-the-future---apache-flink-best-practices-in-2020_1959 (page 96-100)

The Power of Flink

Flink is a Distributed Data Processing System



Consistency, Scale, Ecosystem

Flexible and expressive APIs

Guaranteed correctness

- Exactly-once state consistency
- Event-time semantics

In-memory processing at massive scale

- Runs on 100000s of cores
- Manages 100s TBs of state

Flexible deployments and large ecosystem

- Kubernetes, YARN, Docker, HDFS, Kafka, HBase, Kudu, S3, Kinesis...

Event-driven Applications

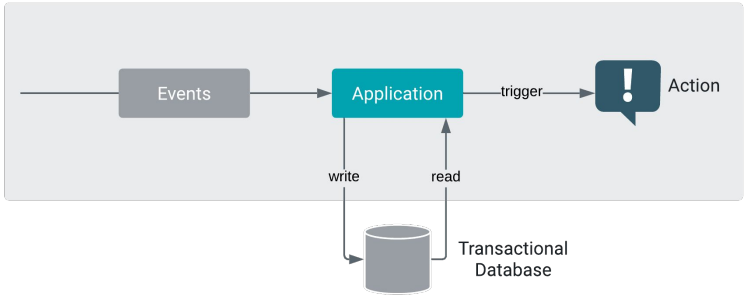
Traditional application design

- Compute & data tier architecture
- React to and process events
- State is stored in (remote) database

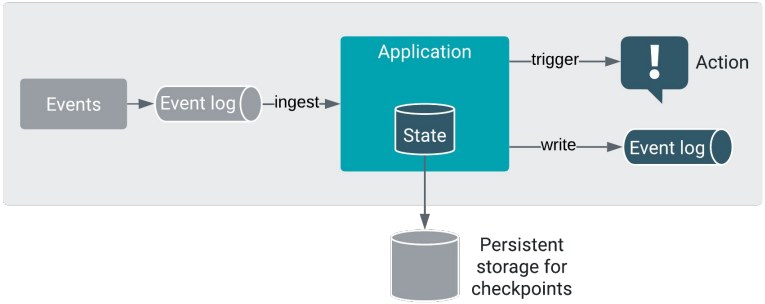
Event-driven application

- State is maintained locally
- Guaranteed consistency by periodic state checkpoints
- Tight coupling of logic and data
- Highly scalable design

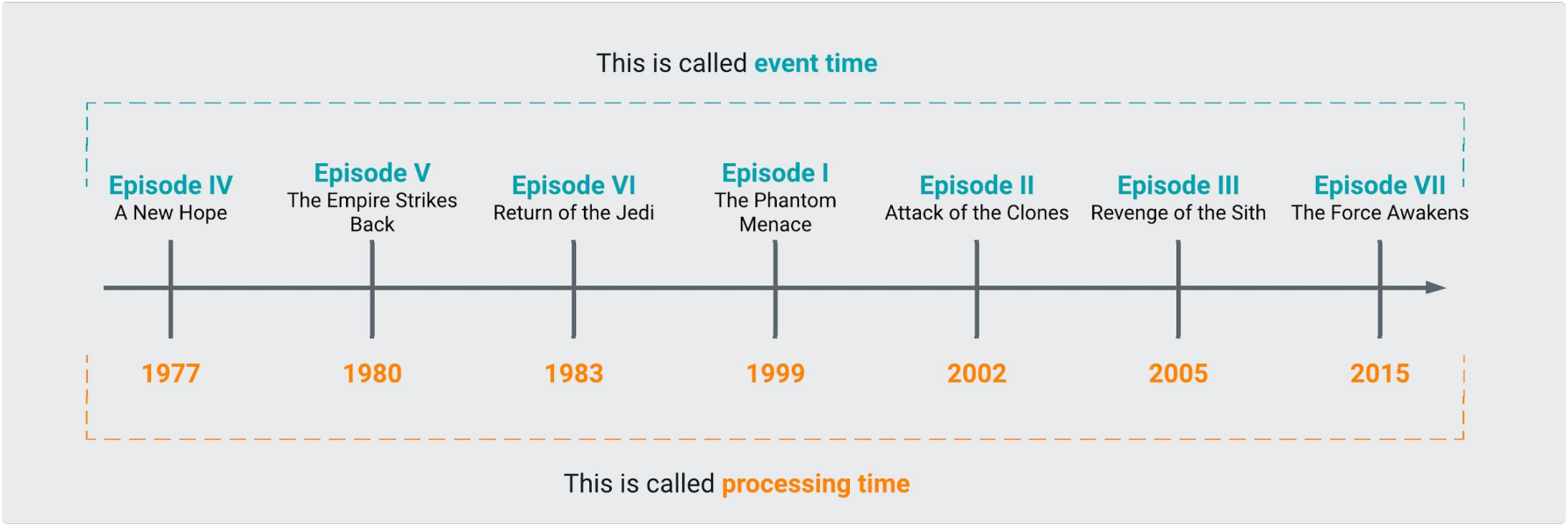
Transactional Application



Event-driven Application

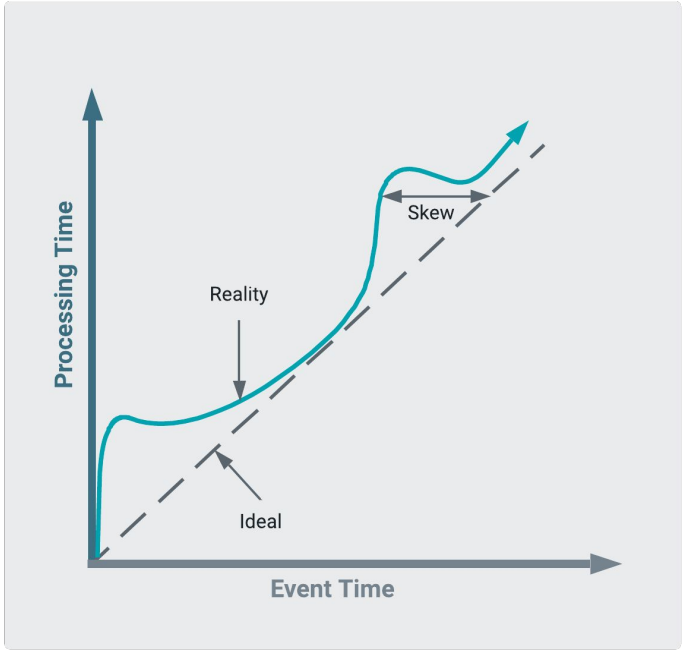


Event-Time and Processing-Time



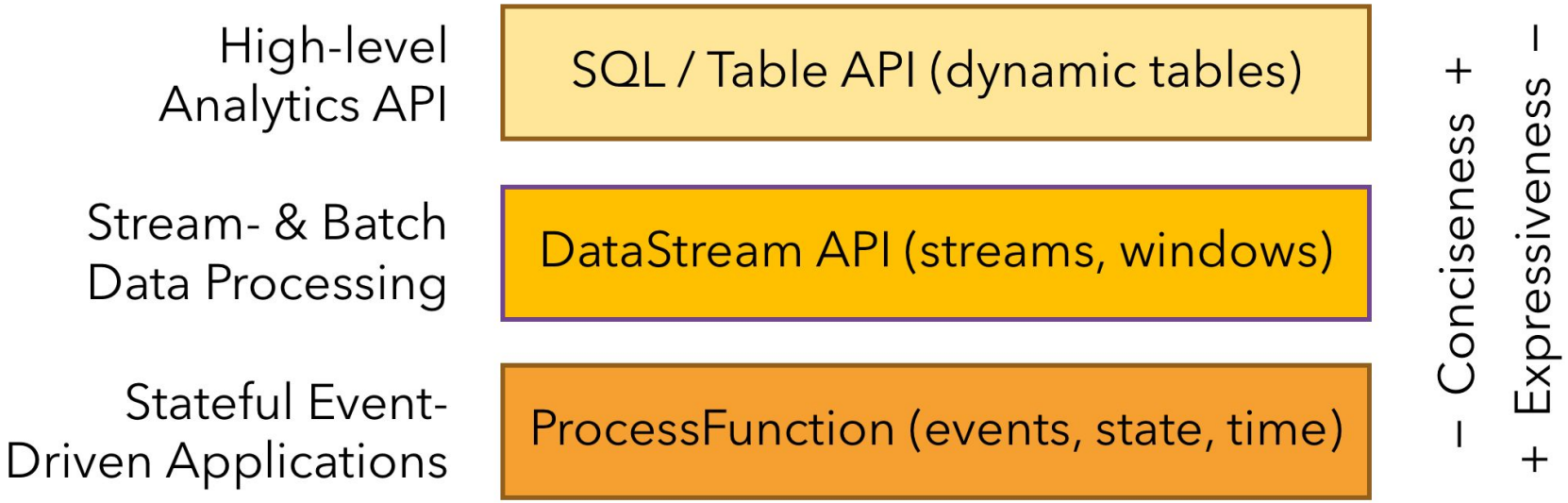
What is Event-Time?

- A record is processed based on an embedded timestamp.
- The “current” time is determined by watermarks
  - A watermark is a special record with a timestamp w
  - Denotes that no more records with a time t <= w will arrive
- Properties of event-time processing
  - Results are deterministic
  - Same semantics when processing recorded and live data
  - Can trade result latency for result completeness



Flink APIs

Layered APIs



SQL & Table API

Unified APIs for streaming data and data at rest

- Run the same query on batch and streaming data
- ANSI SQL: No stream-specific syntax or semantics!
- Many common stream analytics use cases supported

Count clicks per user and session (defined by 30 min. gap of inactivity).

```
SELECT
  userId,
  COUNT(*) AS cnt
  SESSION_START(clicktime, INTERVAL '30' MINUTE)
FROM clicks
GROUP BY
  SESSION(clicktime, INTERVAL '30' MINUTE),
  userId
```

DataStream API

```
// a stream of website clicks
DataStream<Click> clicks = ...

DataStream<Tuple2<String, Long>> result = clicks
  // project clicks to userId and add a 1 for counting
  .map(
    // define function by implementing the MapFunction interface.
    new MapFunction<Click, Tuple2<String, Long>>() {
      @Override
      public Tuple2<String, Long> map(Click click) {
        return Tuple2.of(click.userId, 1L);
      }
    })
  // key by userId (field 0)
  .keyBy(0)
  // define session window with 30 minute gap
  .window(EventTimeSessionWindows.withGap(Time.minutes(30L)))
  // count clicks per session. Define function as lambda function.
  .reduce((a, b) -> Tuple2.of(a.f0, a.f1 + b.f1));
```

Count clicks per user and session (defined by 30 min. gap of inactivity).
Same as the previous SQL query.

The Simplicity of SQL

The Simplicity of SQL

Implementing native Flink applications is challenging

- In-depth knowledge of **streaming concepts**
- Knowledge of **distributed data processing**
- **Java/Scala** experience

Writing Flink applications with SQL is easy

- Everybody knows SQL
- It is the **most widely used language for data analysis**
- SQL queries are optimized and efficiently executed
- Same syntax and semantics for batch & stream processing

SQL and Table API

- Apache Flink features two relational APIs
- Standard compliant **SQL** API
  - Uses Industry-standard SQL parser (Apache Calcite)
- Language-integrated **Table API** for Java, Scala, and Python
  - Composeses queries from relational operators such as selection, filter, and join
- Easily switch between all Flink APIs

Key Features

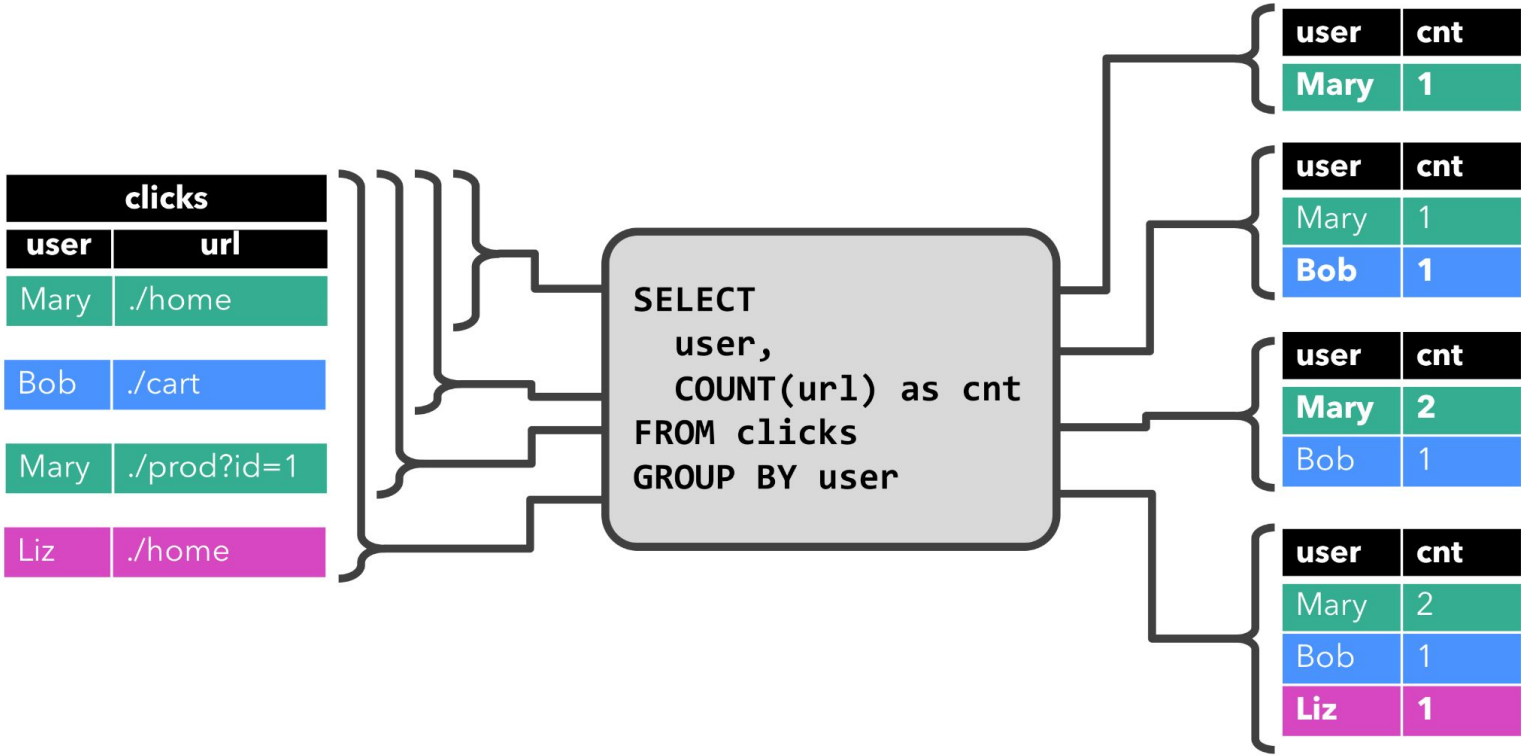
Basic Statements

- CREATE TABLE, DATABASE, VIEW, FUNCTION
- DROP TABLE, DATABASE, VIEW, FUNCTION
- ALTER TABLE, DATABASE, FUNCTION
- SELECT (Queries)
- INSERT INTO
- DESCRIBE, EXPLAIN
- SHOW CATALOGS, DATABASES,TABLES..
- USE
- LOAD, UNLOAD MODULE
- SET, RESET

Queries

- SELECT .. FROM .. WHERE
- JOINs
- Group Aggregation
- Over Aggregation
- Deduplication
- Full TPCS-DS support (*batch*)
- JOIN (interval, temporal, lookup) (*streaming*)
- Window Aggregation (*streaming*)
- Pattern Recognition (*streaming*)

Streaming vs Traditional SQL



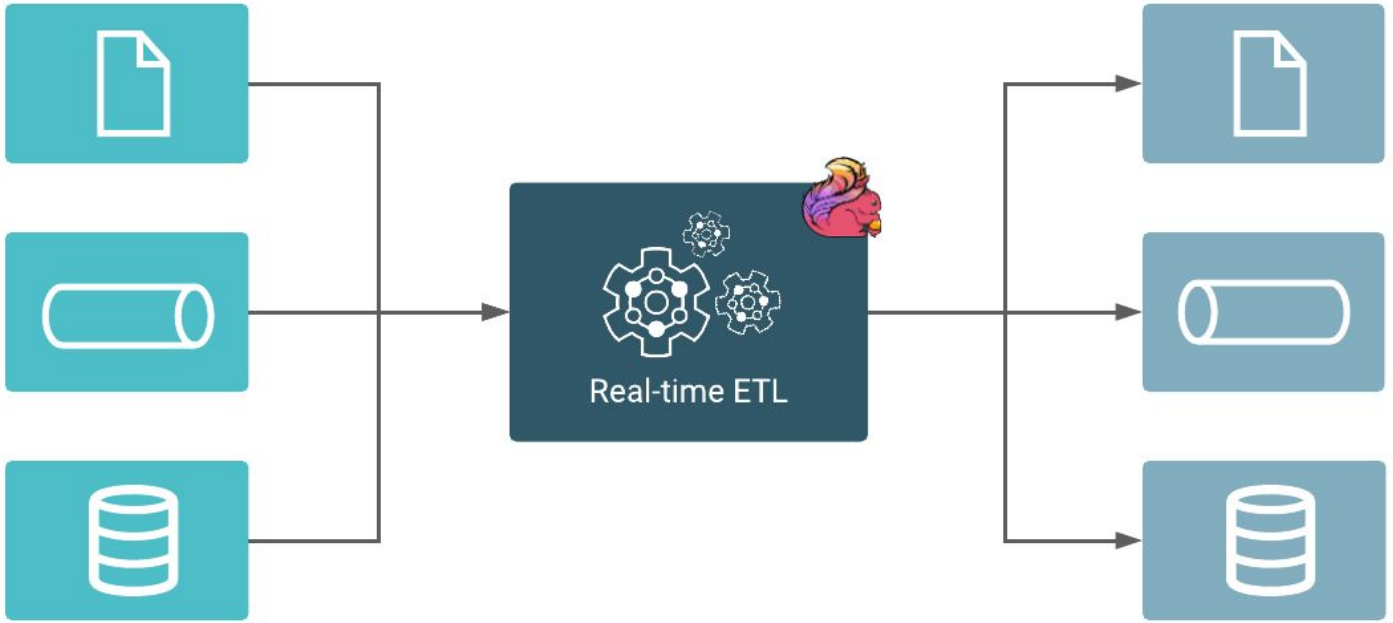
Dynamic Tables and Continuous Queries

Dynamic tables are the core concept of Flink's Table API and SQL support

- A stream is converted into a dynamic table
- A continuous query is evaluated on the dynamic table yielding a new dynamic table
- The resulting dynamic table is converted back into a stream

Use Cases

Build scalable, real-time ETL pipelines



Use Cases

Define and maintain materialized views

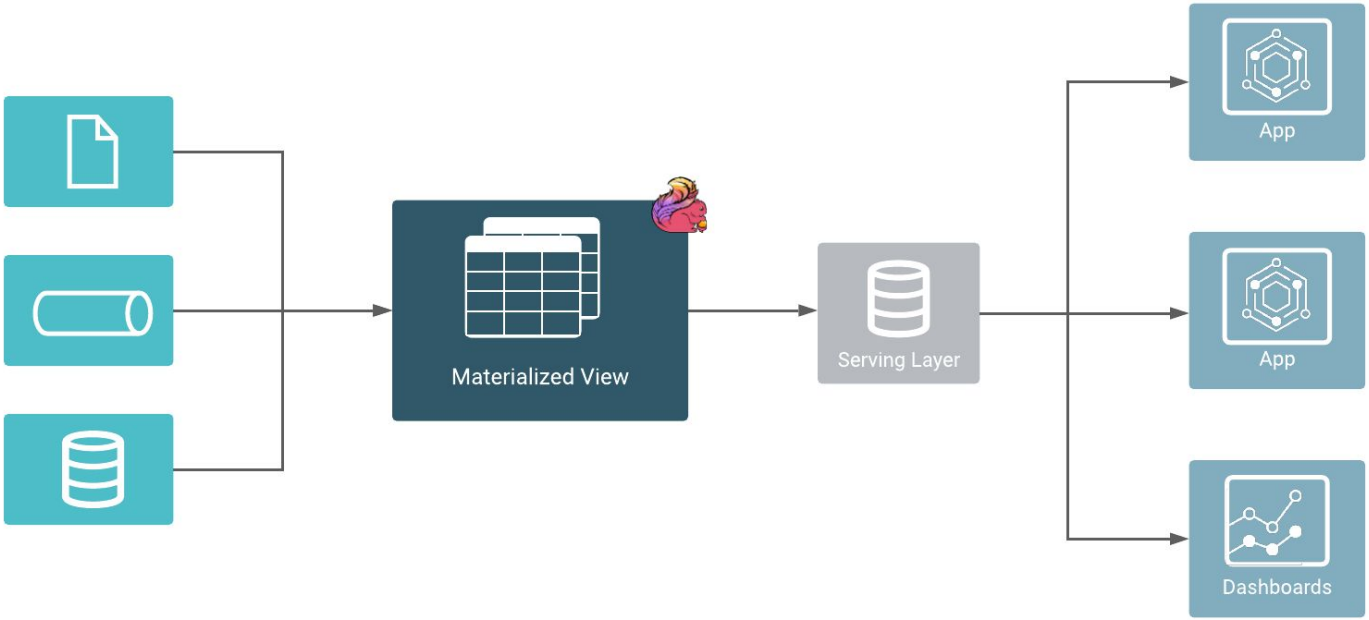


Table & SQL Connectors

Name	Source	Sink
Filesystem	Bounded and Unbounded Scan, Lookup	Streaming Sink, Batch Sink
Elasticsearch	Not supported	Streaming Sink, Batch Sink
Apache Kafka	Unbounded Scan	Streaming Sink, Batch Sink
Amazon Kinesis Data Streams	Unbounded Scan	Streaming Sink
JDBC	Bounded Scan, Lookup	Streaming Sink, Batch Sink
Apache HBase	Bounded Scan, Lookup	Streaming Sink, Batch Sink
Apache Hive	Unbounded Scan, Bounded Scan, Lookup	Streaming Sink, Batch Sink

Flink DDL

```
CREATE TABLE KafkaTable (
  user_id BIGINT,
  item_id BIGINT,
  behavior STRING,
  ts TIMESTAMP(3) METADATA FROM 'timestamp'
) WITH (
  'connector' = 'kafka',
  'topic' = 'user_behavior',
  'properties.bootstrap.servers' = 'localhost:9092',
  'format' = 'json'
)
```

Common Table Formats

Formats	Supported Connectors
CSV	Apache Kafka, Upsert Kafka, Amazon Kinesis Data Streams, Filesystem
JSON	Apache Kafka, Upsert Kafka, Amazon Kinesis Data Streams, Filesystem, Elasticsearch
Apache Avro	Apache Kafka, Upsert Kafka, Amazon Kinesis Data Streams, Filesystem
Apache Parquet	Filesystem
Apache ORC	Filesystem
Debezium, Maxwell, Canal CDC	Apache Kafka, Filesystem

Queries and Time

Typical Questions

"Join price info with most recent exchange rate"

"Emit an alert if 3 unsuccessful login attempts occurred within 2 minutes"

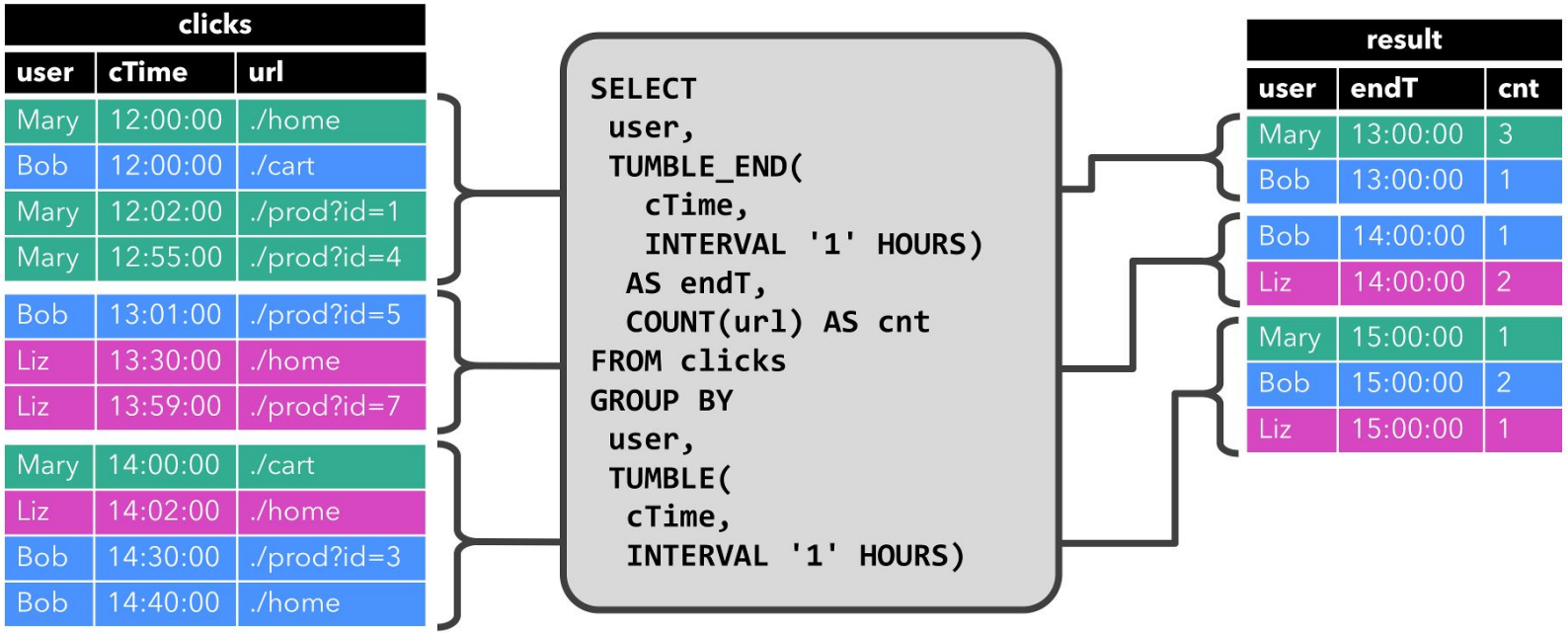
"Count the number of 403 requests per user over the duration of a session"

Event Time vs Processing Time

```
CREATE TABLE clicks (
    user VARCHAR,
    url  VARCHAR,
    cTime TIMESTAMP(3),
    WATERMARK FOR cTime AS cTime – INTERVAL '2' MINUTE)
```

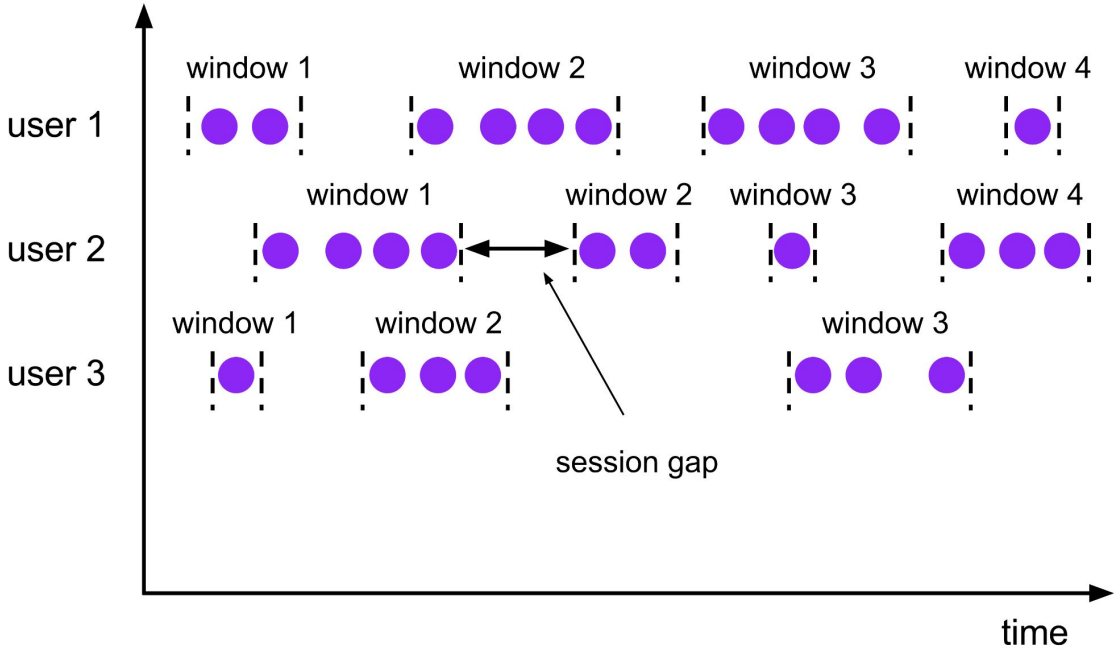
```
CREATE TABLE clicks (
    user VARCHAR,
    url VARCHAR,
    cTime AS PROCTIME())
```

TUMBLE Windows



SESSION Windows

count the number of 403 requests per user over the duration of a session



SESSION Windows in Flink SQL

Queries and Time

```
SELECT  userid,
  SESSION_START(log_time, INTERVAL '10' SECOND) AS sstart,
  SESSION_END(log_time, INTERVAL '10' SECOND) AS send,
  COUNT(request_line) AS request_cnt
FROM server_logs
WHERE status_code = '403'
GROUP BY
  userid,
  SESSION(log_time, INTERVAL '10' SECOND)
```

Summary

Summary

Flink is a powerful stream processor, gaining adoption

- Consistency at scale
- Wide range of users and use cases
- Foundation for real time data infrastructure

Writing Flink applications with SQL is easy

- Everybody knows SQL
- It is the most widely used language for data analysis
- SQL queries are optimized and efficiently executed
- Same syntax and semantics for batch & stream processing