Apache Flink™ Training

System Overview

Tzu-Li (Gordon) Tai

tzulitai@apache.org

@tzulitai

Flink.tw

Apache Flink Taiwan User Group

Sept 2016 @ HadoopCon

00 Who am I?

- 戴資力（Gordon）
- Apache Flink Committer
- Co-organizer of Apache Flink Taiwan User Group
- Software Engineer @ VMFive
- Java, Scala
- Enjoy developing distributed computing systems

00 This session will be about ...

- An understanding of Apache Flink
- Flink’s streaming-first philosophy
- What exactly is a “streaming dataflow engine”?

Apache Flink

an open-source platform for distributed stream and batch data processing

- Apache Top-Level Project since Jan. 2015
- **Streaming Dataflow Engine** at its core
  - Low latency
  - High Throughput
  - Stateful
  - Distributed

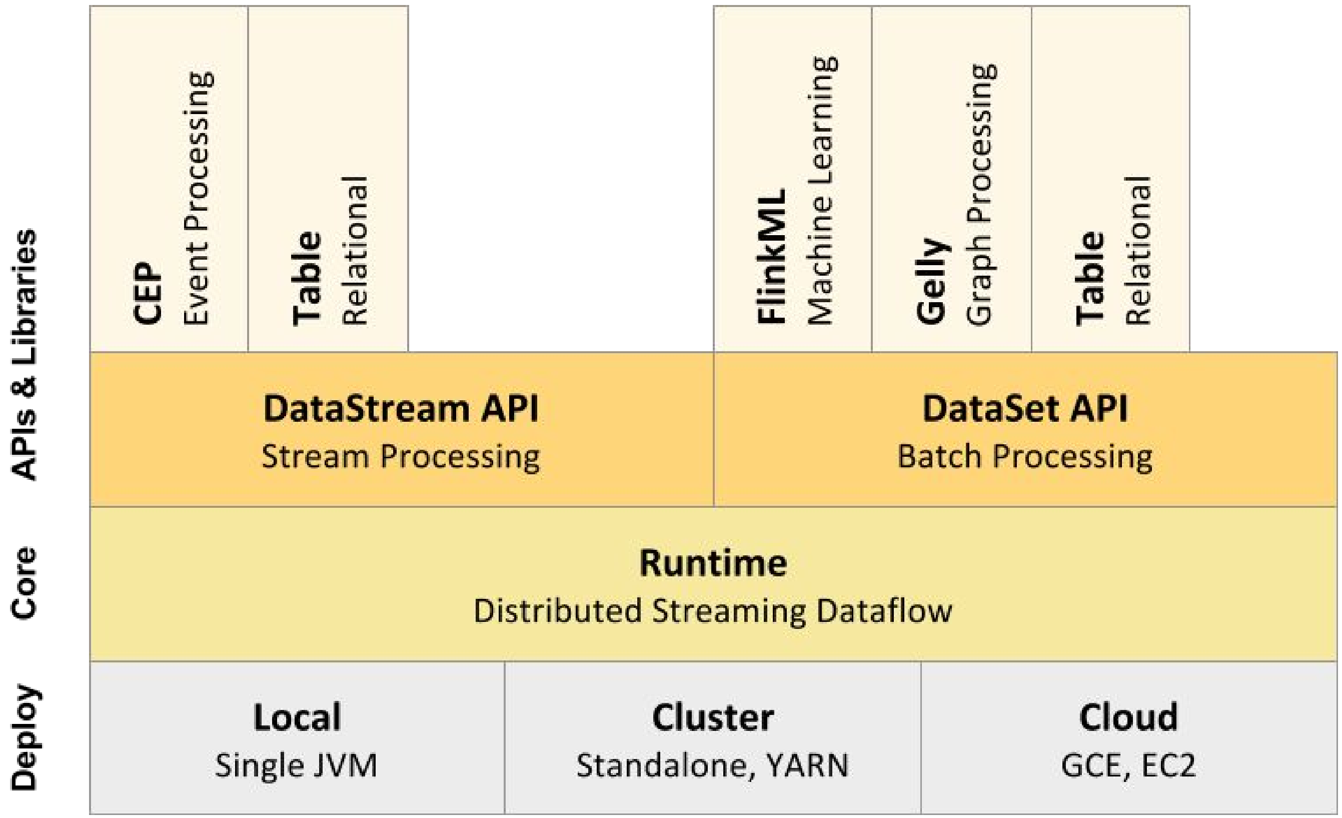
THE APACHE® SOFTWARE FOUNDATION

Apache Flink

an open-source platform for distributed stream and batch data processing

- ~100 active contributors for Flink 1.1.x release
- Used **in production** by:
  - **Alibaba** - realtime search ranking optimization
  - **Uber** - ride request fulfillment marketplace
  - **Netflix** - Stream Processing as a Service (SPaaS)
  - **Kings Gaming** - realtime data science dashboard
  - ...

01 Flink Components Stack



APIs & Libraries

CEP Event Processing

Table Relational

FlinkML Machine Learning

Gelly Graph Processing

Table Relational

DataStream API Stream Processing

DataSet API Batch Processing

Core

Runtime Distributed Streaming Dataflow

Deploy

Local Single JVM

Cluster Standalone, YARN

Cloud GCE, EC2

02 Scala Collection-like API

```
case class Word (word: String, count: Int)
```

DataSet API

```
val lines: DataSet[String] = env.readTextFile(...)

lines.flatMap(_.split(“ ”)).map(word => Word(word,1))
     .groupBy(“word”).sum(“count”)
     .print()
```

DataStream API

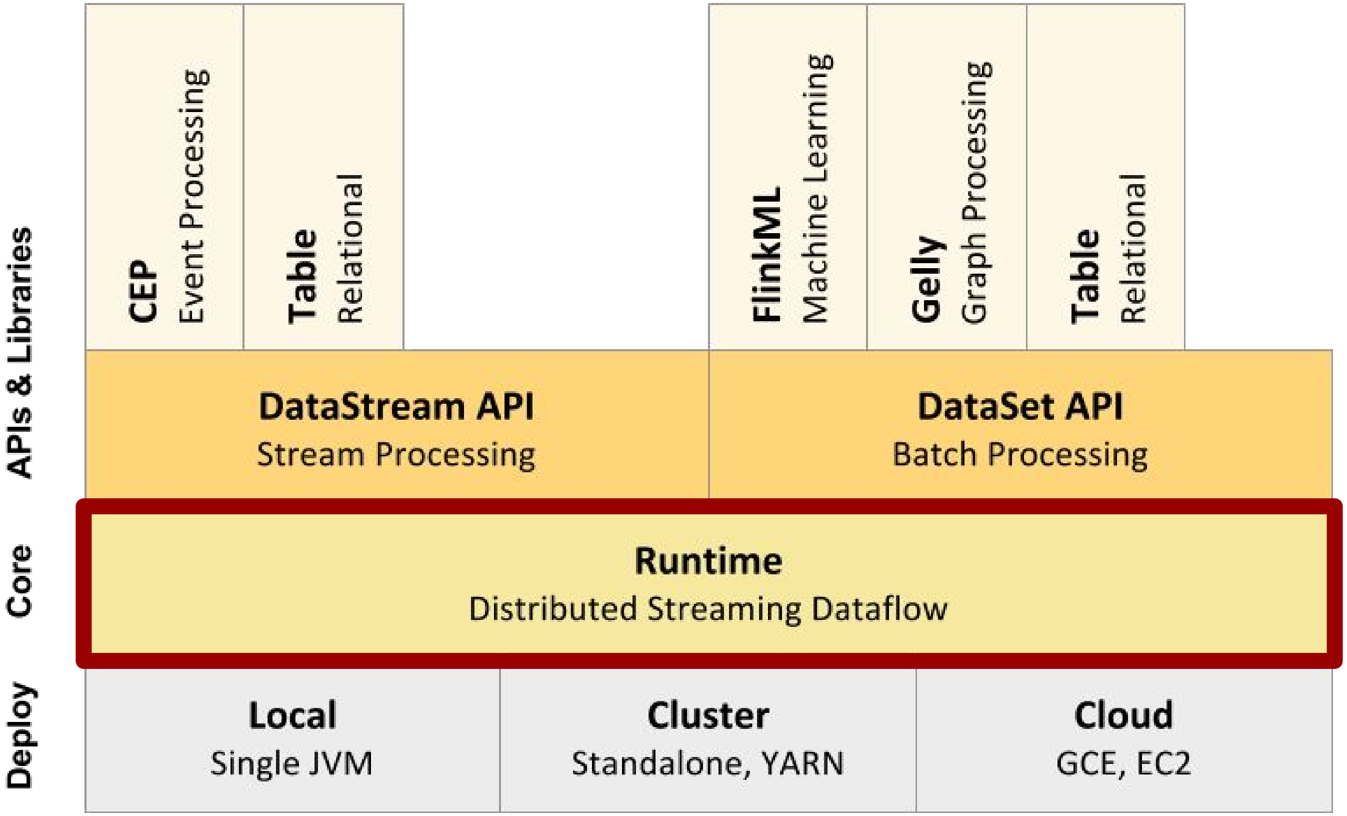
```
val lines: DataStream[String] = env.addSource(new KafkaSource(...))

lines.flatMap(_.split(“ ”)).map(word => Word(word,1))
     .keyBy(“word”).timeWindow(Time.seconds(5)).sum(“count”)
     .print()
```

02 Scala Collection-like API

```
.filter(...).flatmap(...).map(...).groupBy(...).reduce(...)
```

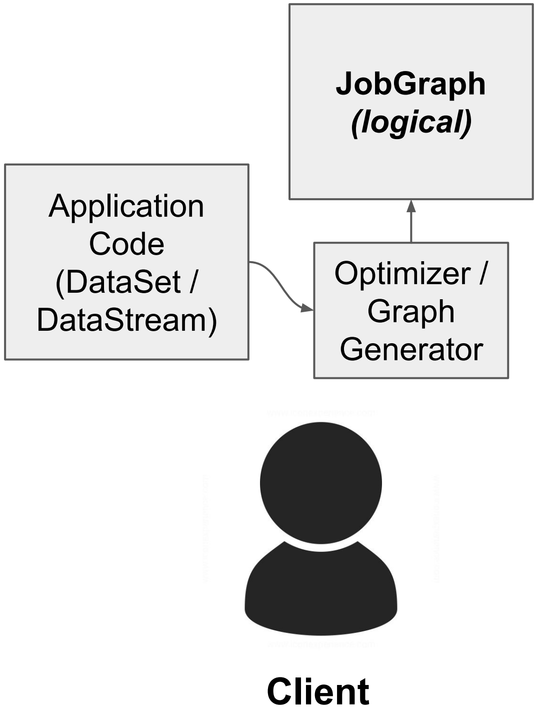
- Becoming the *de facto standard* for new generation API to express data pipelines
- Apache Spark, Apache Flink, Apache Beam ...

03 Focusing on the Engine ...



Lifetime of a Flink Job

04 Flink Job



JobGraph
(logical)

Optimizer / Graph Generator

Application Code (DataSet / DataStream)

Client

04 Flink Job

Application code:

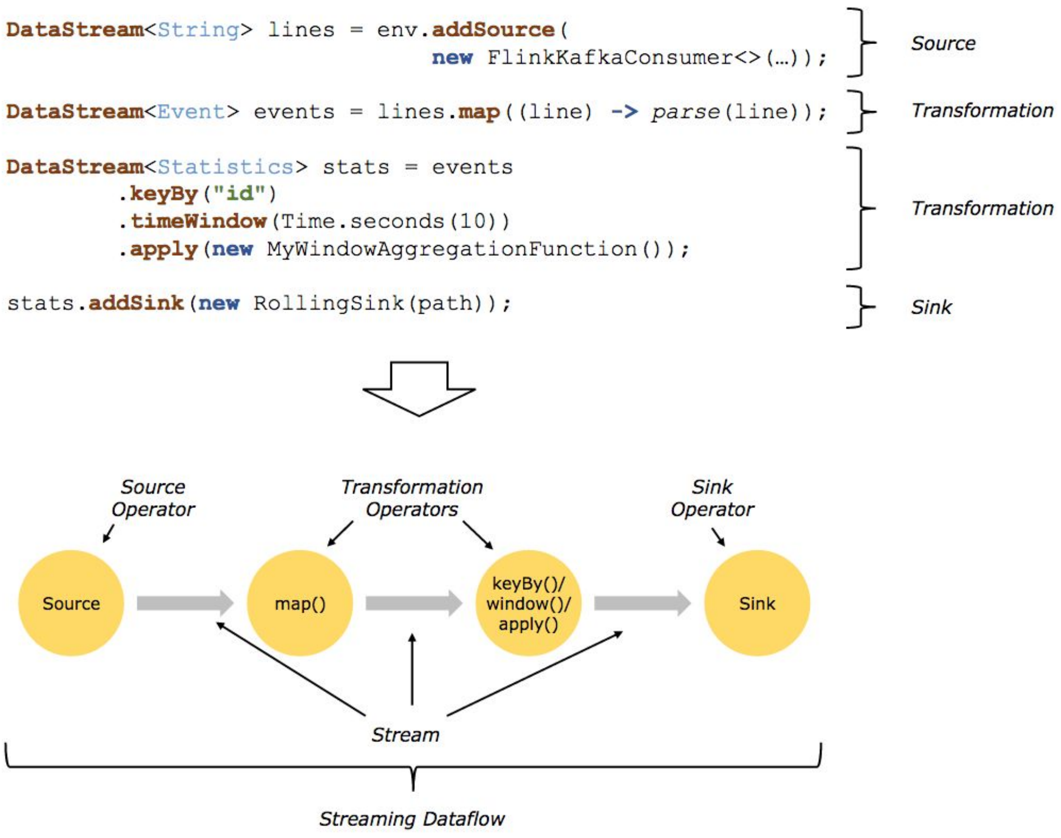
- Define sources
- Define transformations
- Define sinks

optimizer

graph generator

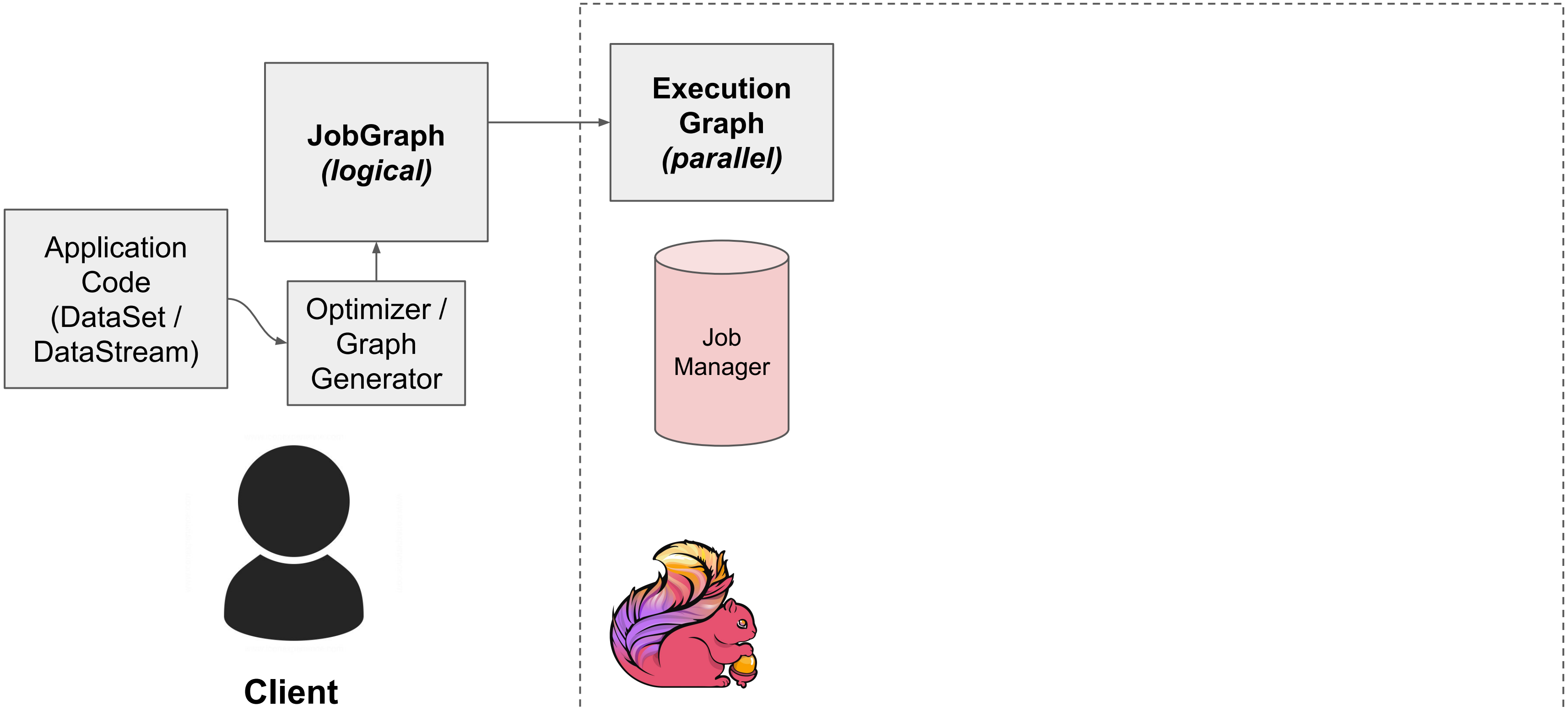
JobGraph

logical view of the dataflow pipeline



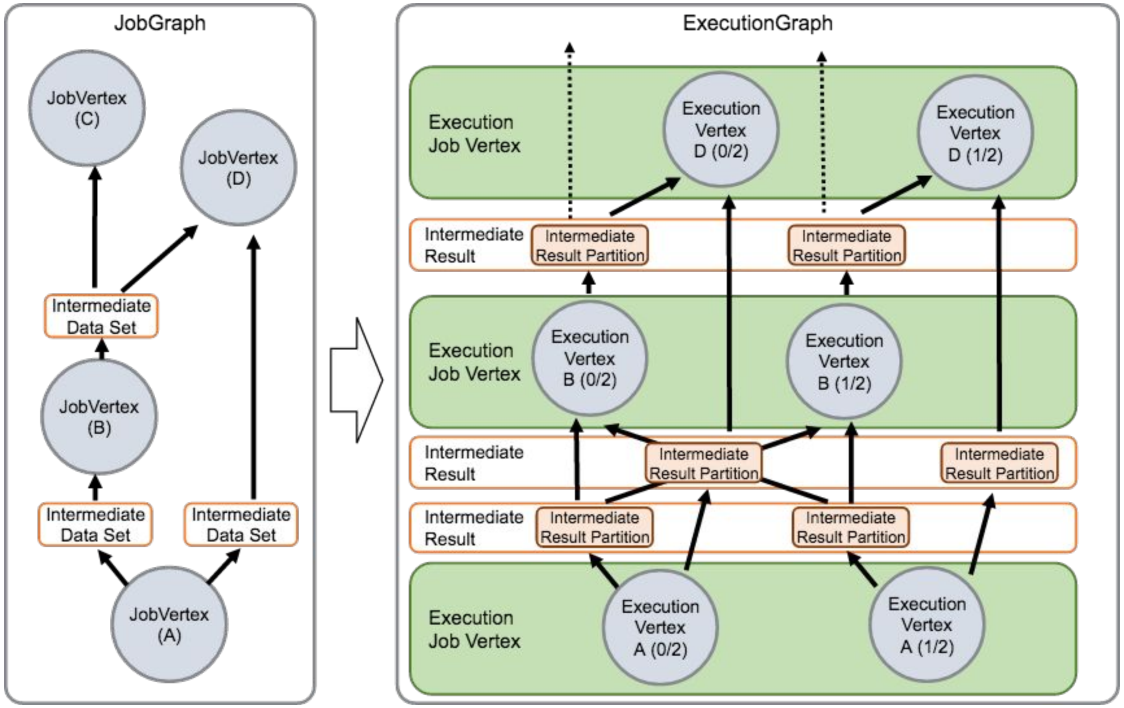
```
DataStream<String> lines = env.addSource(
                                   new FlinkKafkaConsumer<>(…));

DataStream<Event> events = lines.map((line) -> parse(line));

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))
        .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));
```

04 Flink Job


JobGraph
(logical)
Execution
Graph
(parallel)
Application
Code
(DataSet /
DataStream)
Optimizer /
Graph
Generator
Job
Manager
Client

04 Flink Job

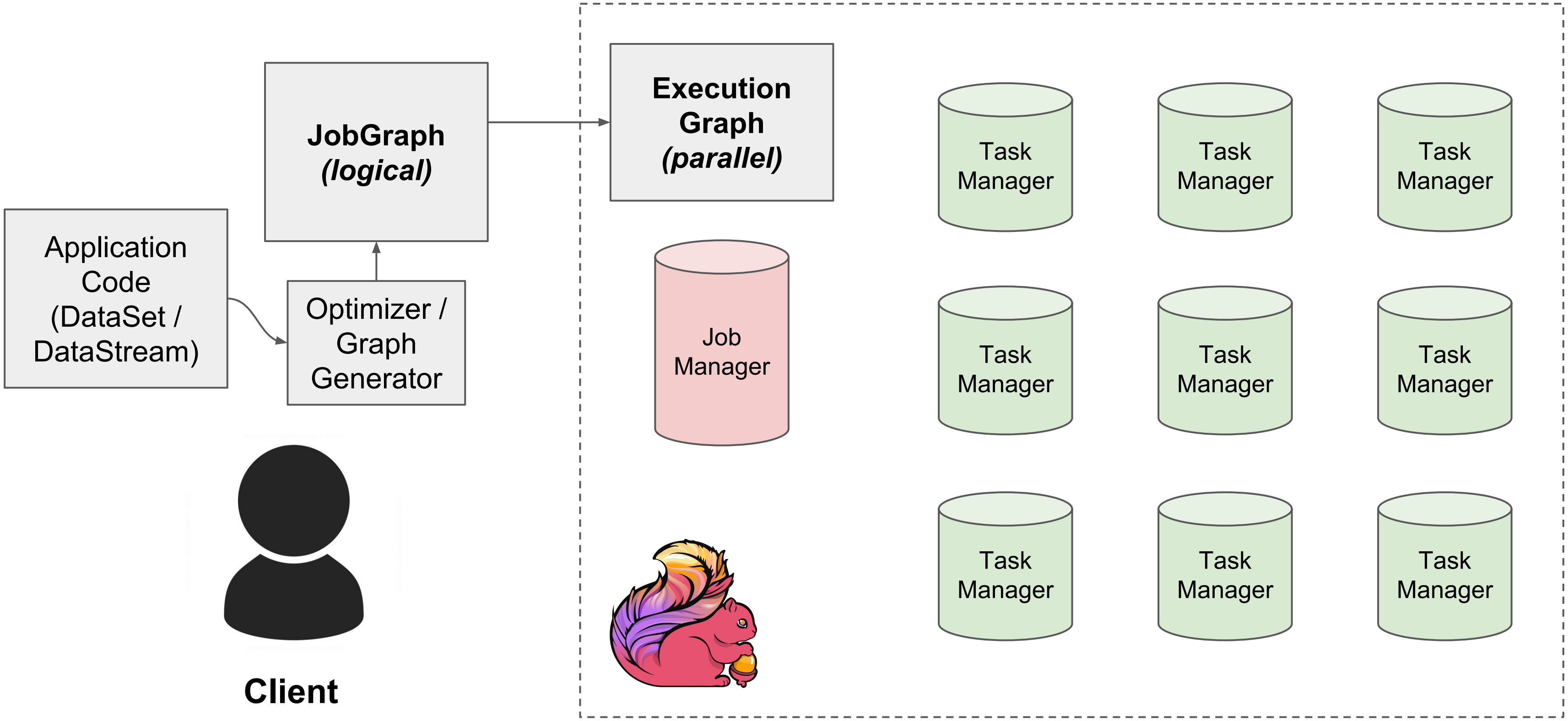
logical execution plan



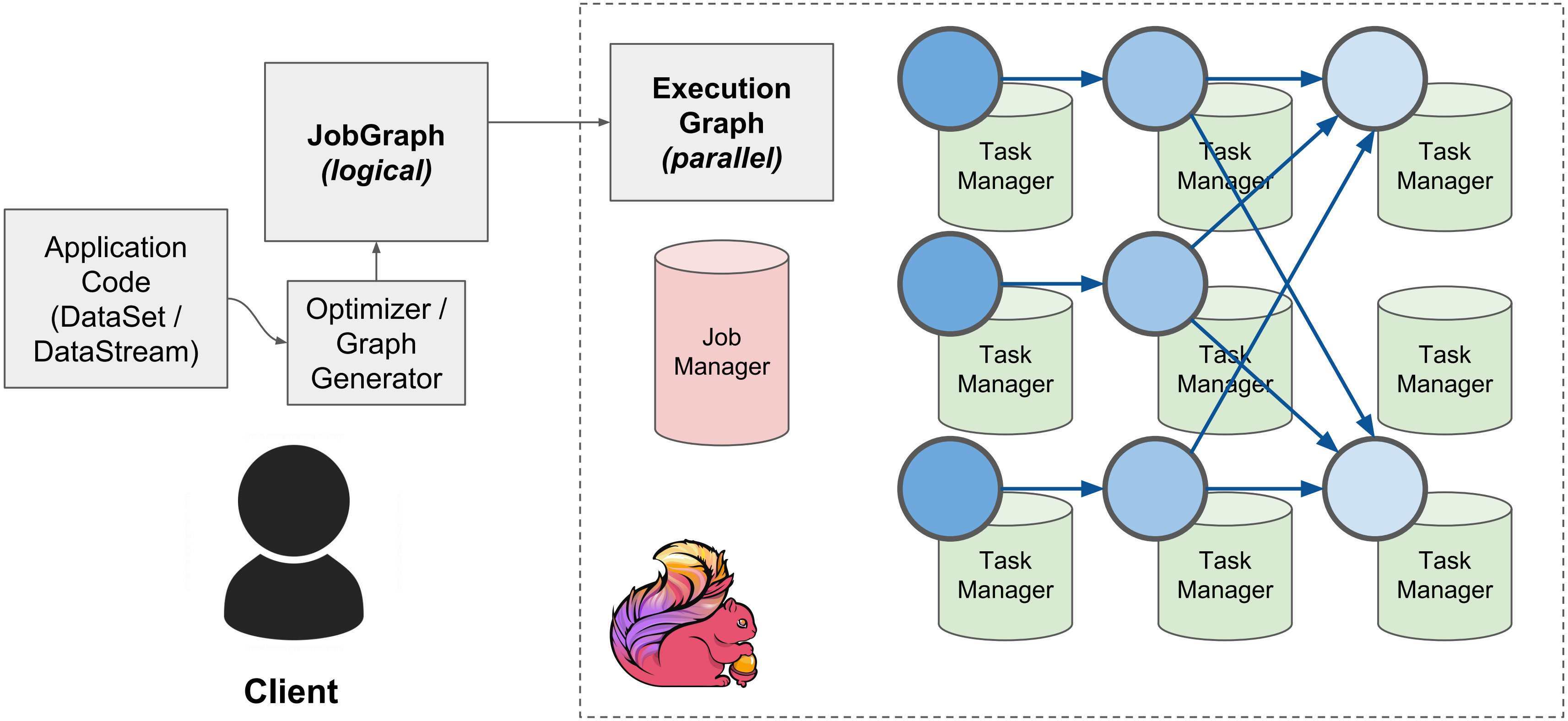
physical parallel execution plan

→ ***Intermediate Result Partitions*** define a data shipping channel between parallel tasks (not materialized data in memory or disk)

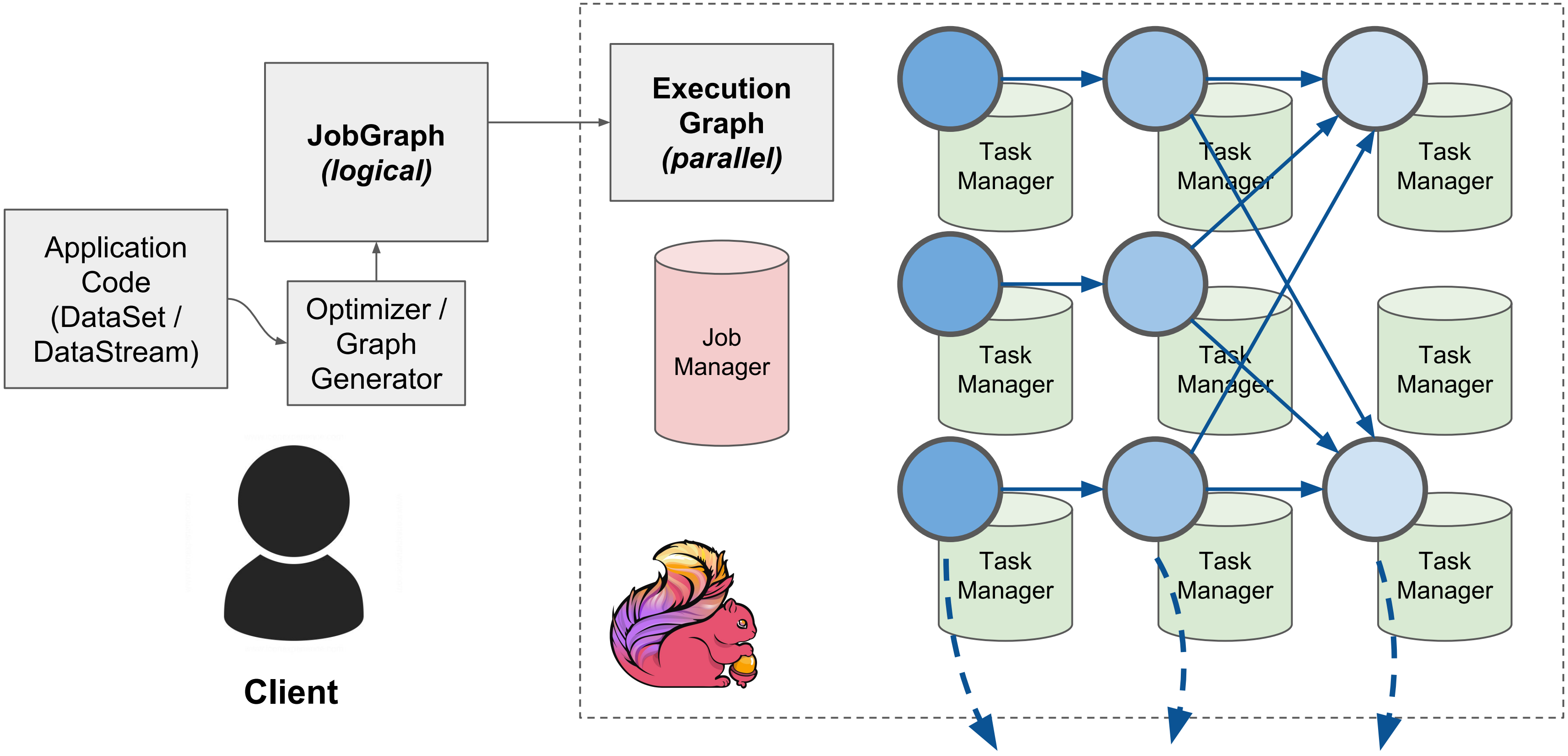
04 Flink Job


JobGraph
(logical)
Execution Graph
(parallel)
Application Code (DataSet / DataStream)
Optimizer / Graph Generator
Job Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Client

04 Flink Job

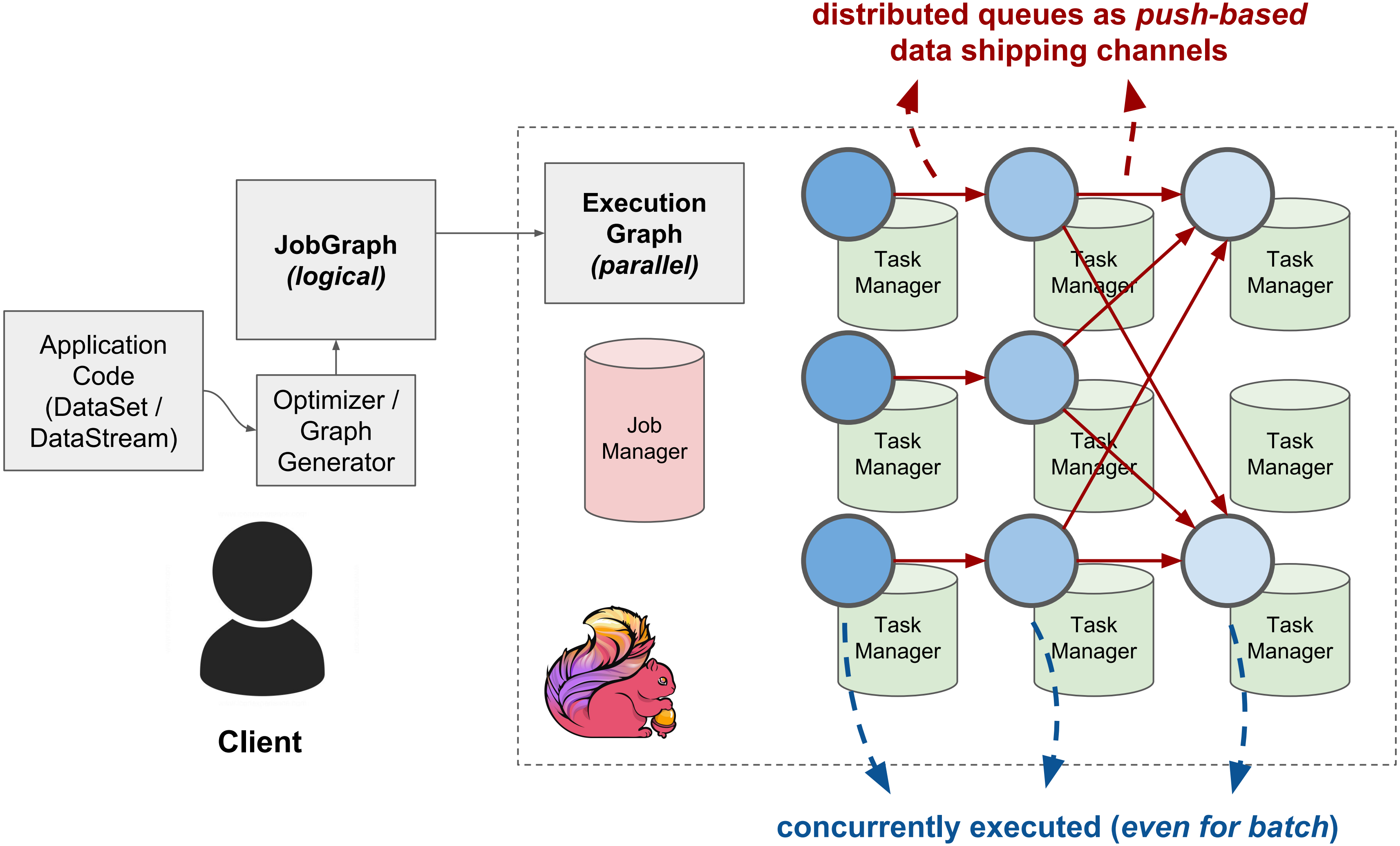

JobGraph
(logical)
Execution
Graph
(parallel)
Application
Code
(DataSet /
DataStream)
Optimizer /
Graph
Generator
Job
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Task
Manager
Client

04 Flink Job



concurrently executed (*even for batch*)

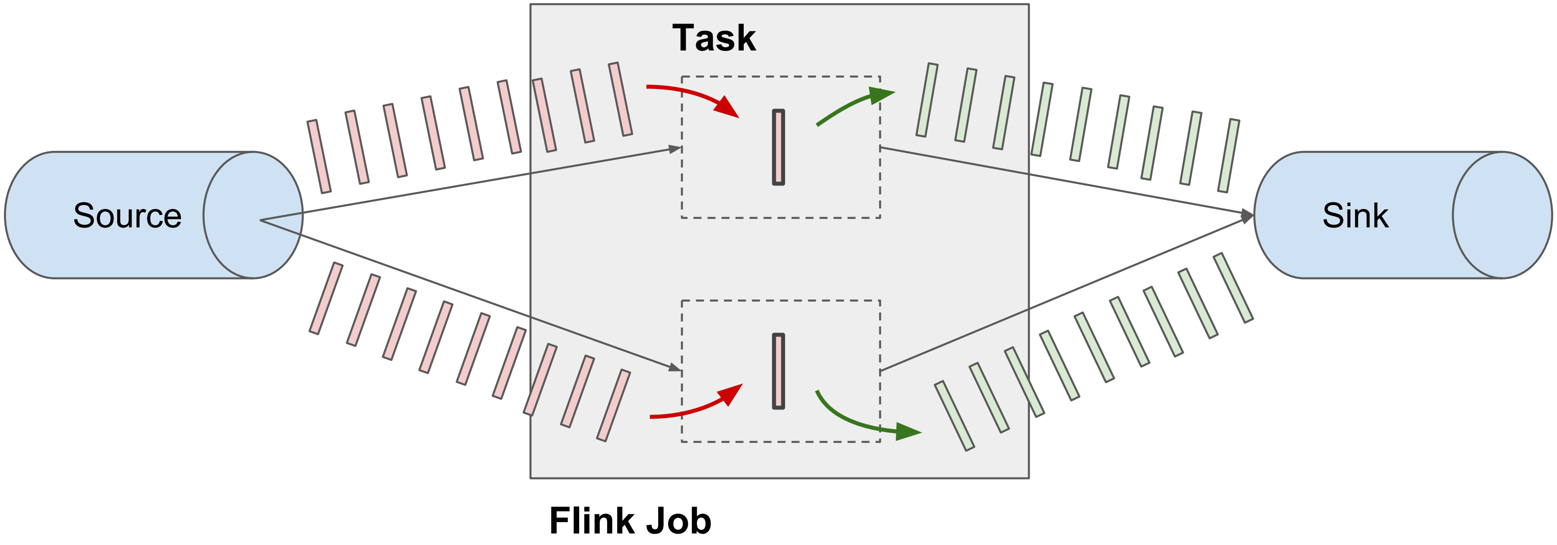
04 Flink Job


distributed queues as *push-based* data shipping channels
JobGraph (logical)
Execution Graph (parallel)
Application Code (DataSet / DataStream)
Optimizer / Graph Generator
Job Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Task Manager
Client
concurrently executed (*even for batch*)

Advantages of a Streaming Dataflow Engine

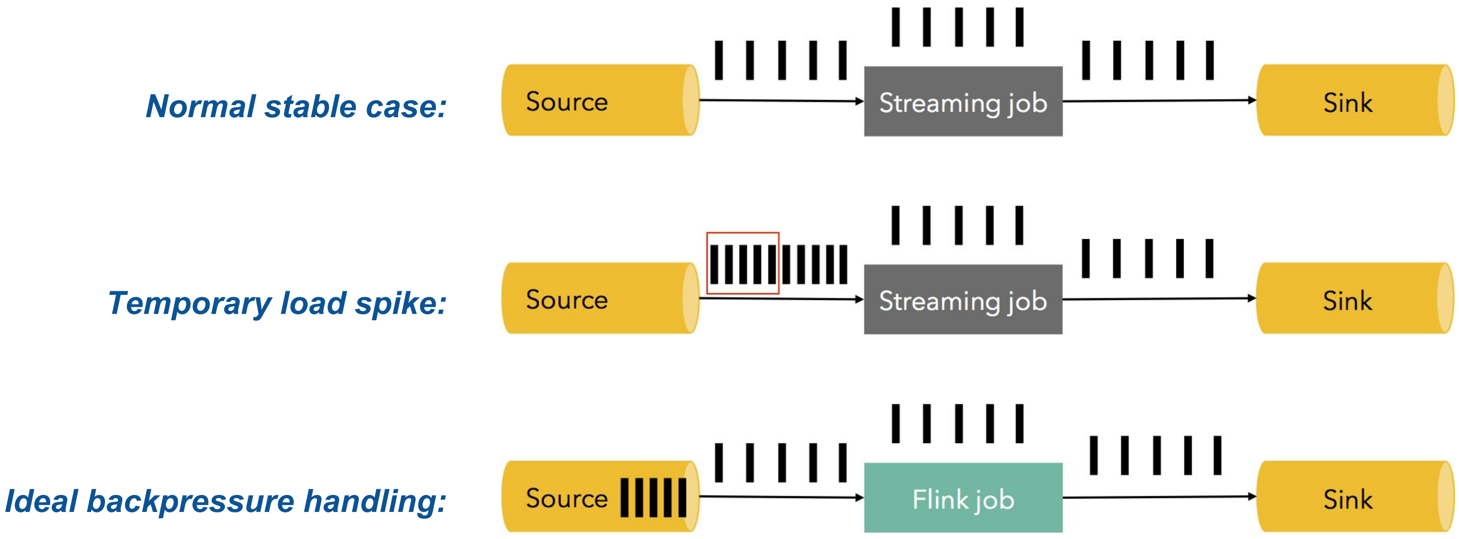
05 Streaming Dataflow Engine

- True one-at-a-time streaming
- Tasks are scheduled and executed concurrently
- Data transmission is *pushed* across *distributed queues*

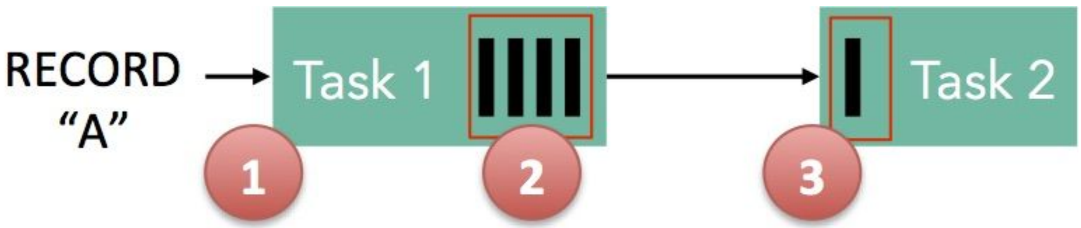


06 Backpressure Handling

- Receiving data at a higher rate than a system can process during a temporary load spike
    - ex. GC pauses at processing tasks
    - ex. data source natural load spike



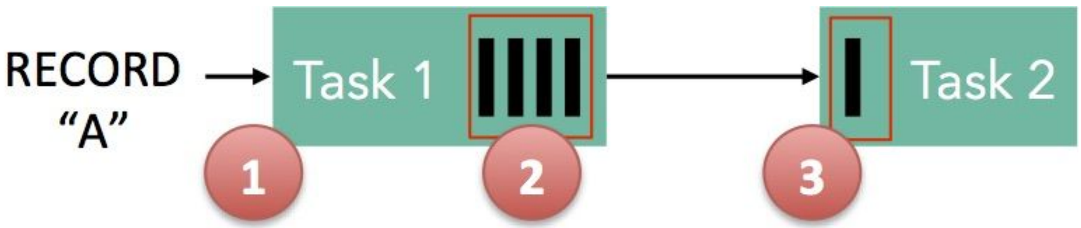
07 How Distributed Queues Work



1. Record “A” enters Task 1, and is processed
2. The record is serialized into an *output buffer* at Task 1
3. The buffer is shipped to Task 2’s *input buffer*

Observation: Buffers need to be available throughout the process *(think Java blocking queues used between threads)*

07 How Distributed Queues Work



1. Record "A" enters Task 1, and is processed

Taken from an output buffer pool

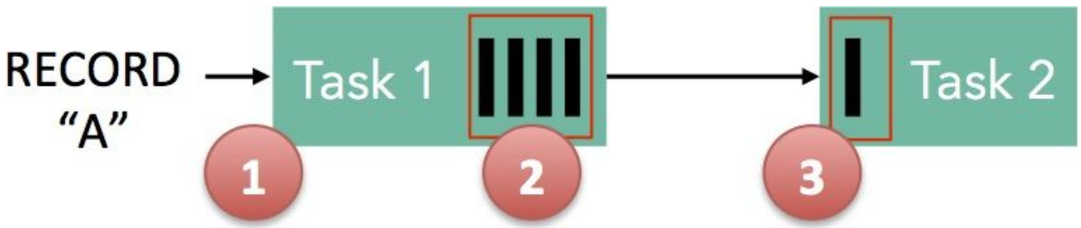
2. The record is serialized into an ***output buffer*** at Task 1

3. The buffer is shipped to Task 2's ***input buffer***

Taken from an input buffer pool

Observation: Buffers need to be available throughout the process *(think Java blocking queues used between threads)*

07 How Distributed Queues Work



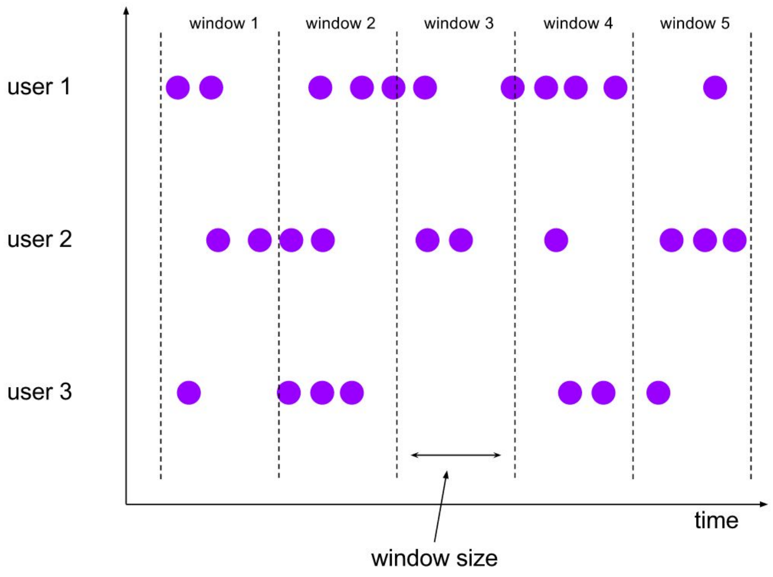
- Size of buffer pools defines the amount of buffering Flink can do in presence of different sender / receiver speeds
- **More memory** simply means Flink can **simply buffer away** transient, short-living **backpressure**
- **Lesser memory** simply means more **immediate backpressure response**

08 Flexible Windows

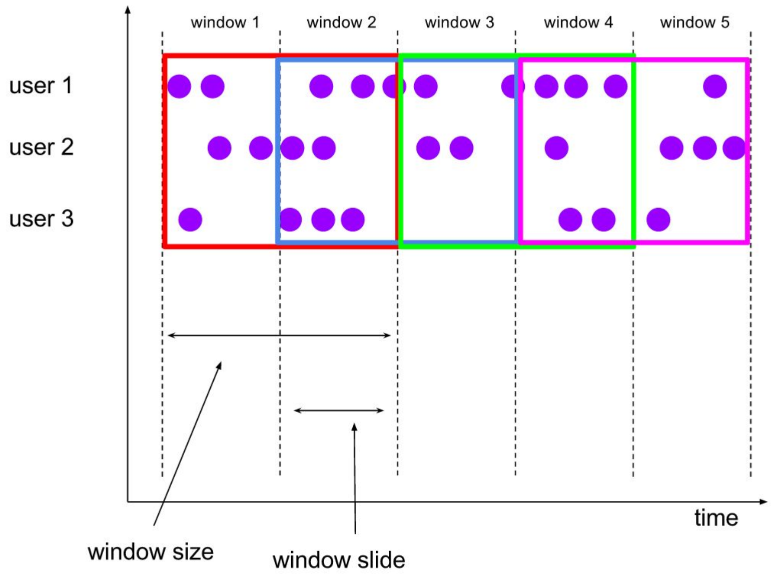
- Due to one-at-a-time processing, Flink has very powerful built-in windowing (certainly among the best in the current streaming framework solutions)
  - Time-driven: *Tumbling window, Sliding window*
  - Data-driven: *Count window, Session window*

09 Time Windows

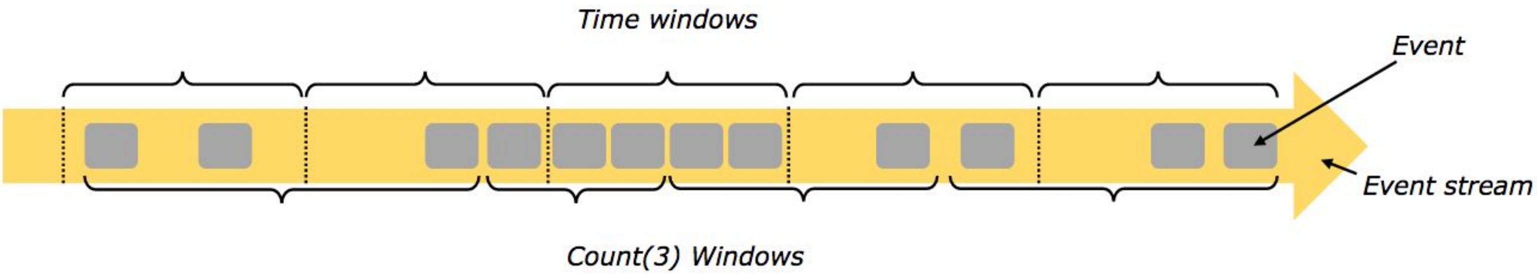
Tumbling Time Window



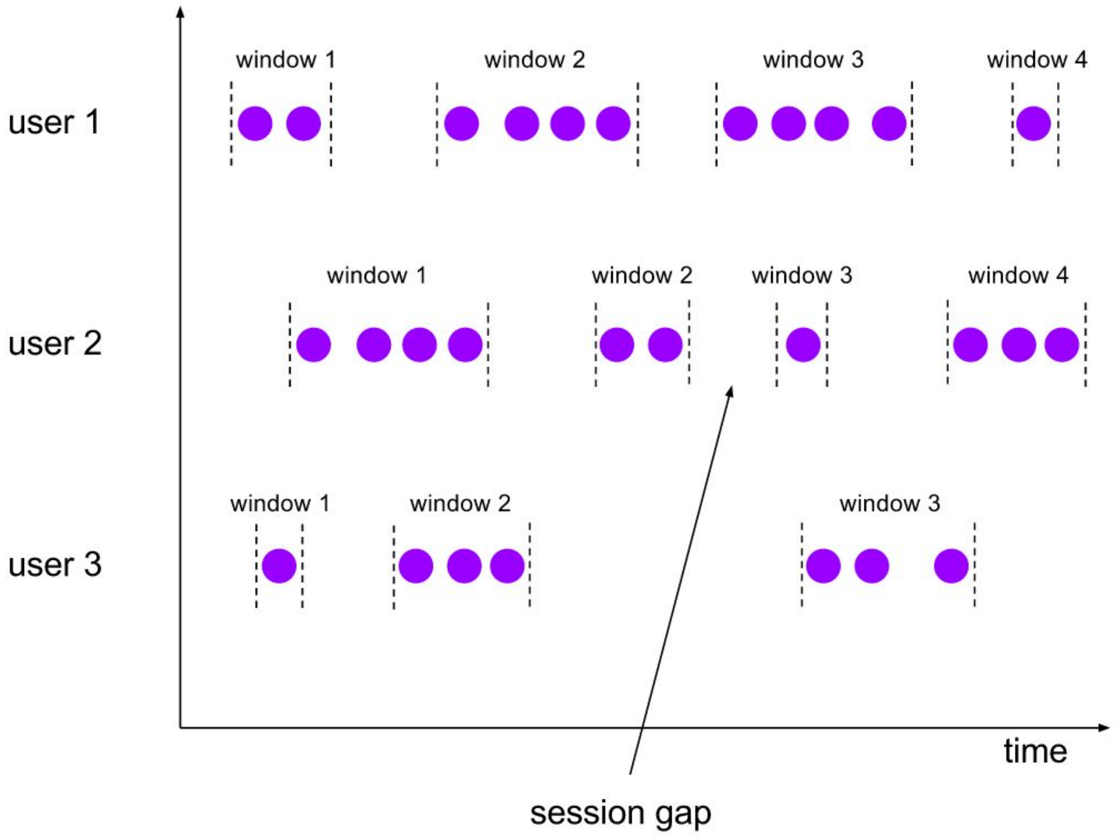
Sliding Time Window



10 Count-Triggered Windows



11 Session Windows



Prepare for Hands-On!

12 Clone & Import Project

→ https://github.com/flink-taiwan/hadoopcon2016-training