Apache Flink® Training

Intro

Apache Flink® Training

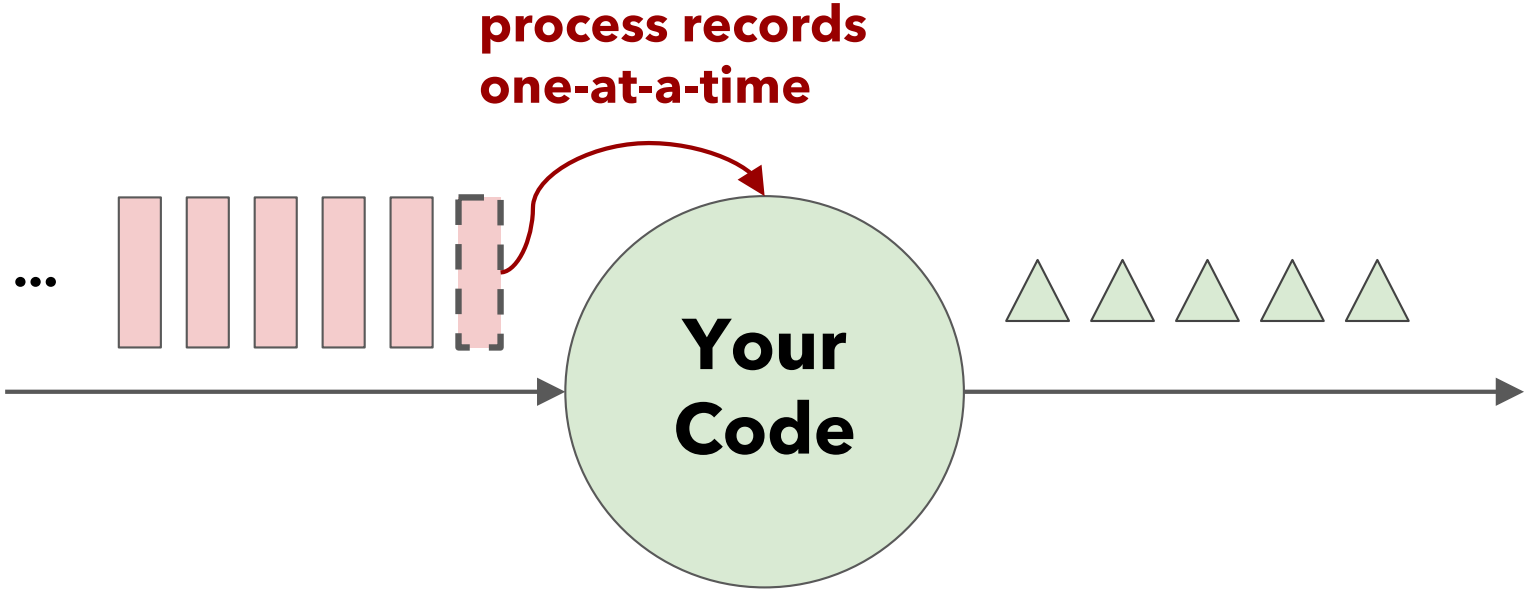
dataArtisans

Flink v1.3 – 20.06.2017

Where we're going today

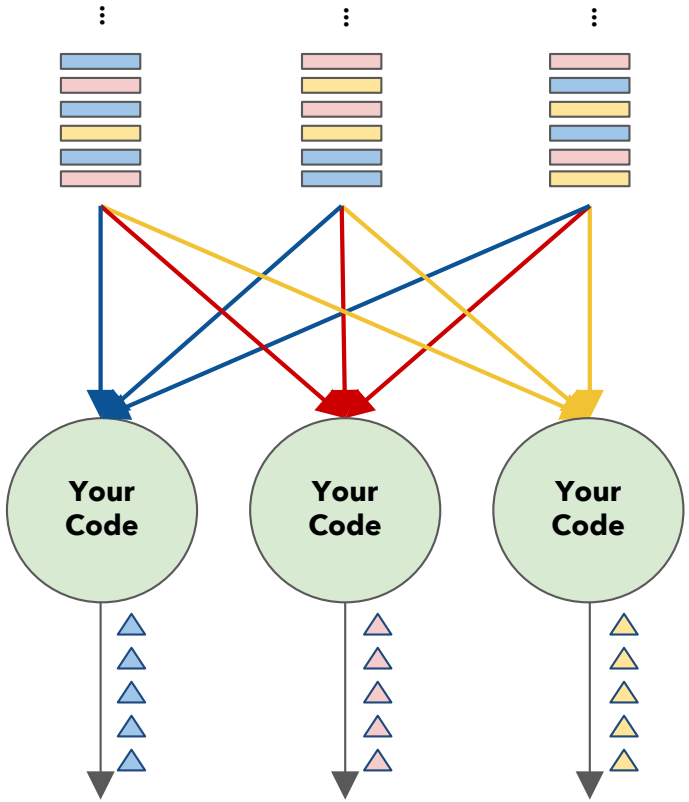
- Stateful stream processing as a paradigm for continuous data
- Apache Flink is a sophisticated and battle-tested stateful stream processor with a comprehensive set of features
- Efficiency, management, and operational issues for state are taken very seriously

Stream Processing



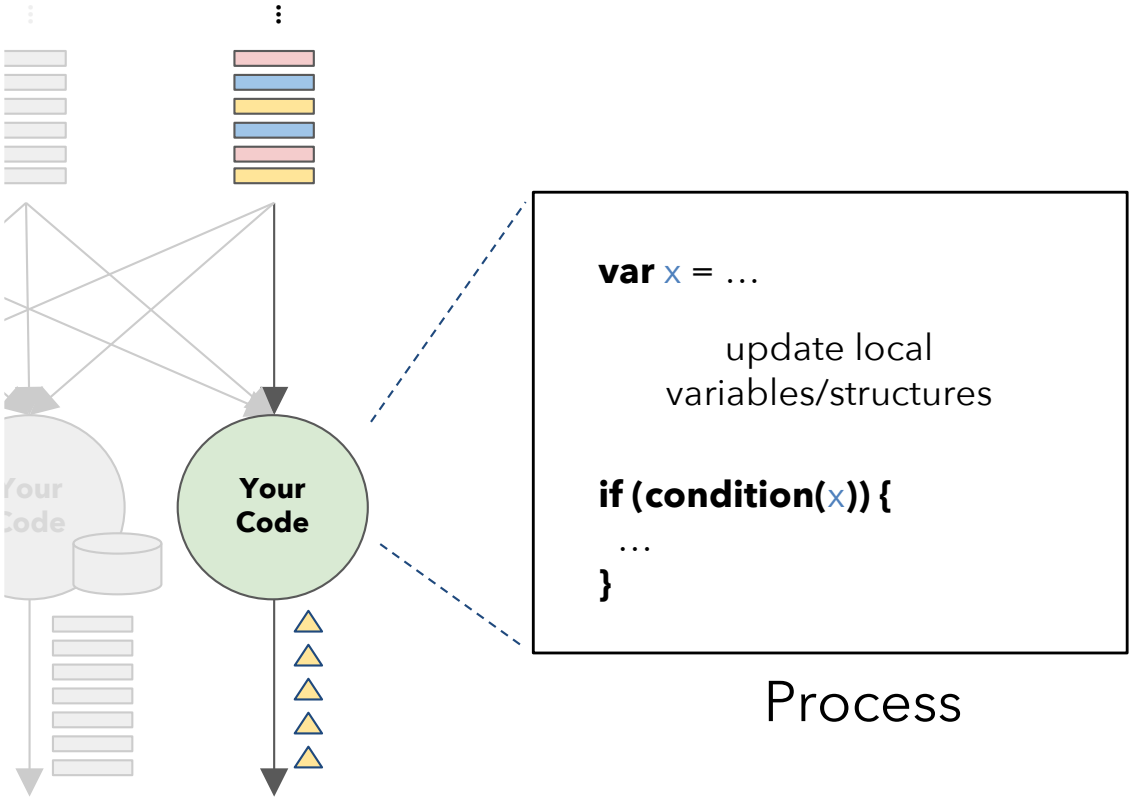
Long running computation, on an endless stream of input

Distributed Stream Processing

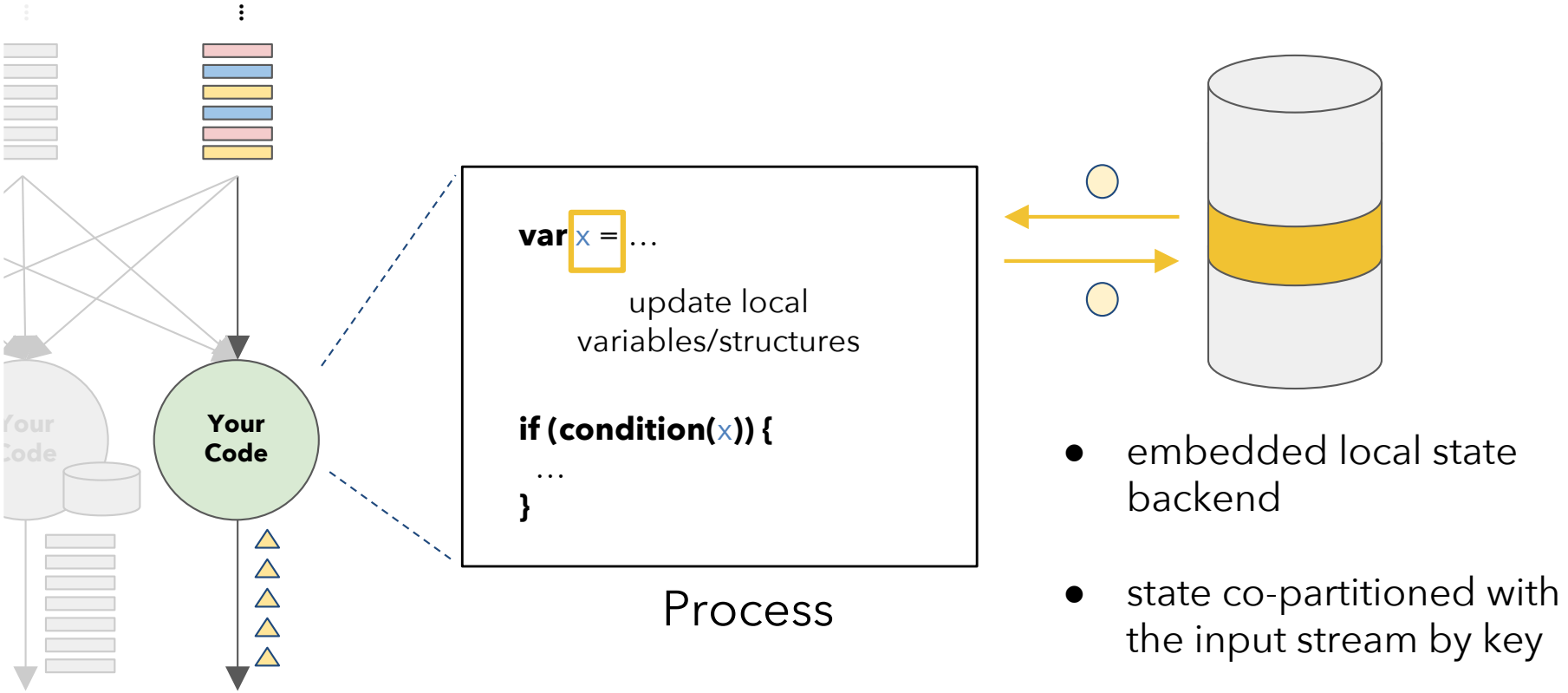


- partitions input streams by some key in the data
- distributes computation across multiple instances
- each instance is responsible for some key range

Stateful Stream Processing

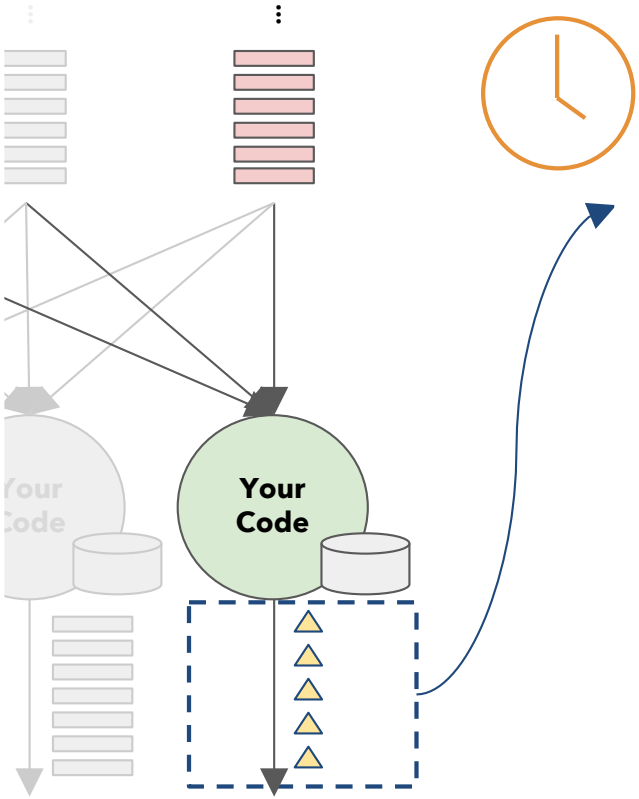


Stateful Stream Processing



- embedded local state backend
- state co-partitioned with the input stream by key

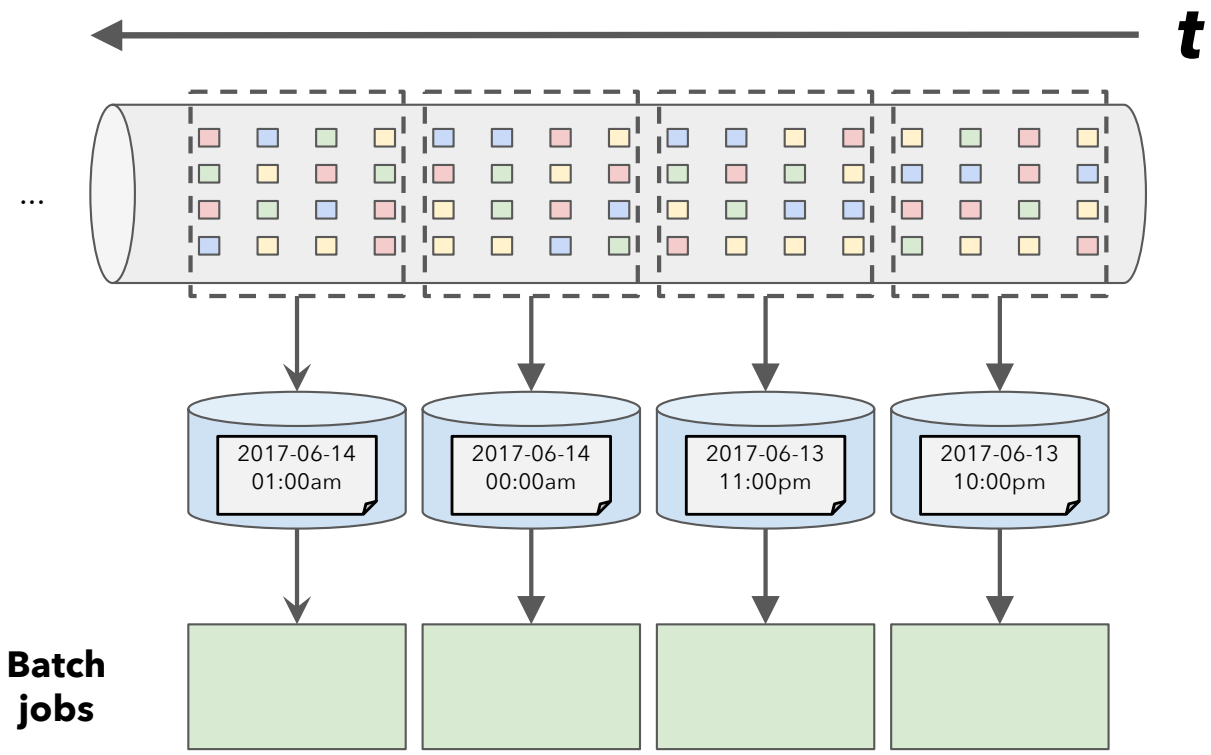
About time ...



When should results be emitted?

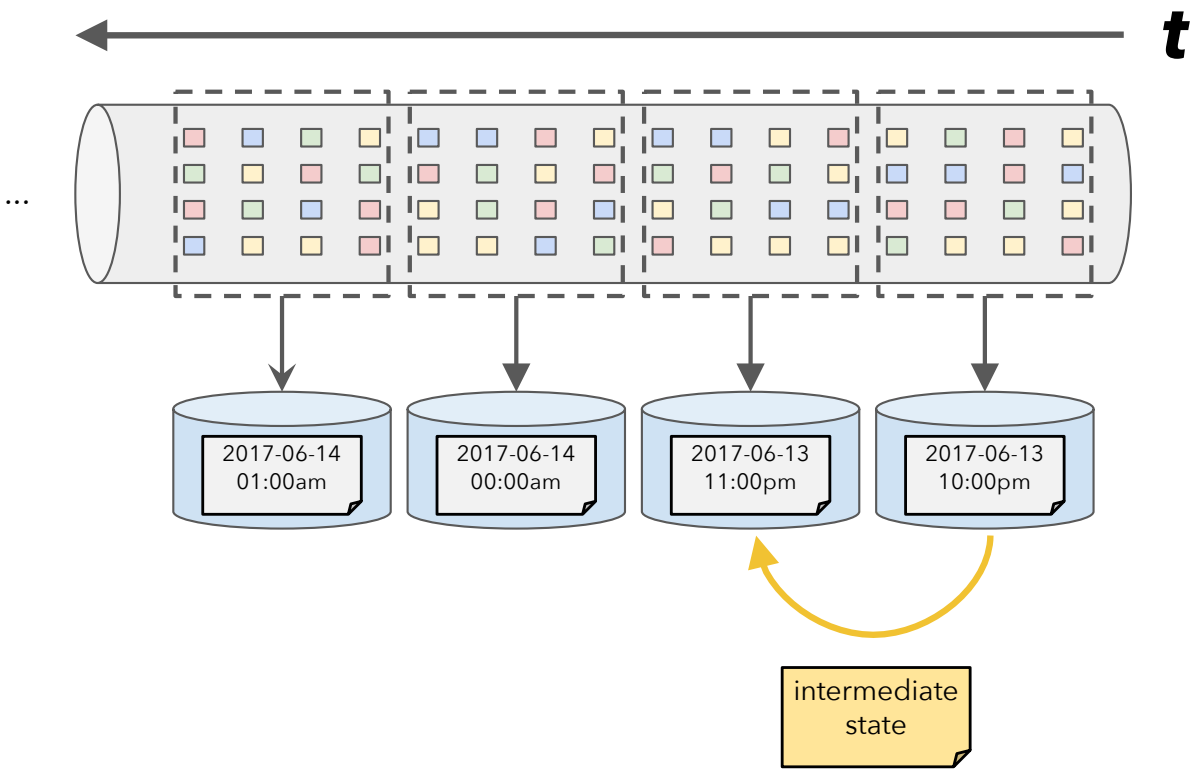
- Control for determining when the computation has fully processed all required events
- It's mostly about time. *e.g.* have I received all events for 3 - 4 pm?
- Did event *B* occur within 5 minutes of event *A*?
- **Wall clock** time is not correct. **Event-time awareness** is required.

Traditional batch processing



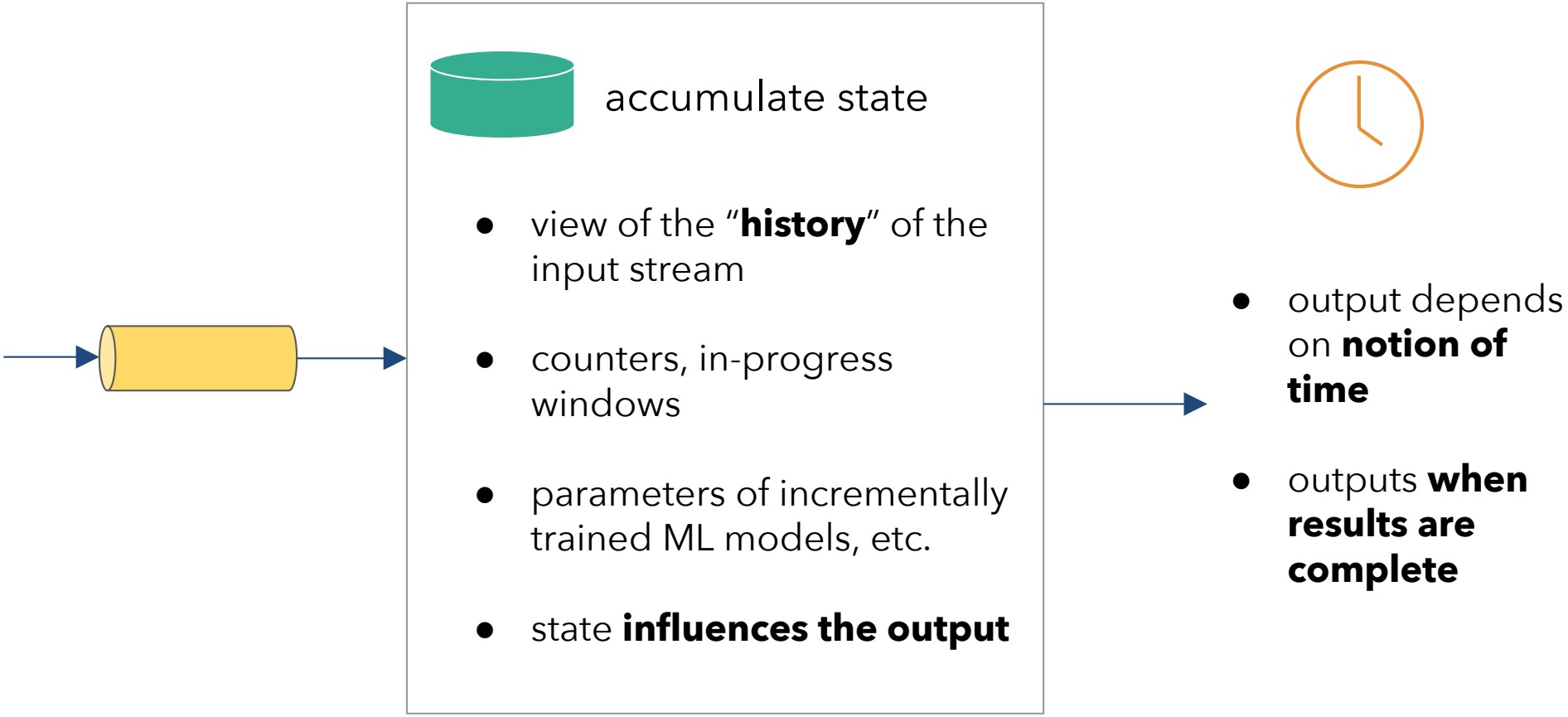
- Continuously ingesting data
- Time-bounded batch files
- Periodic batch jobs

Traditional batch processing (II)



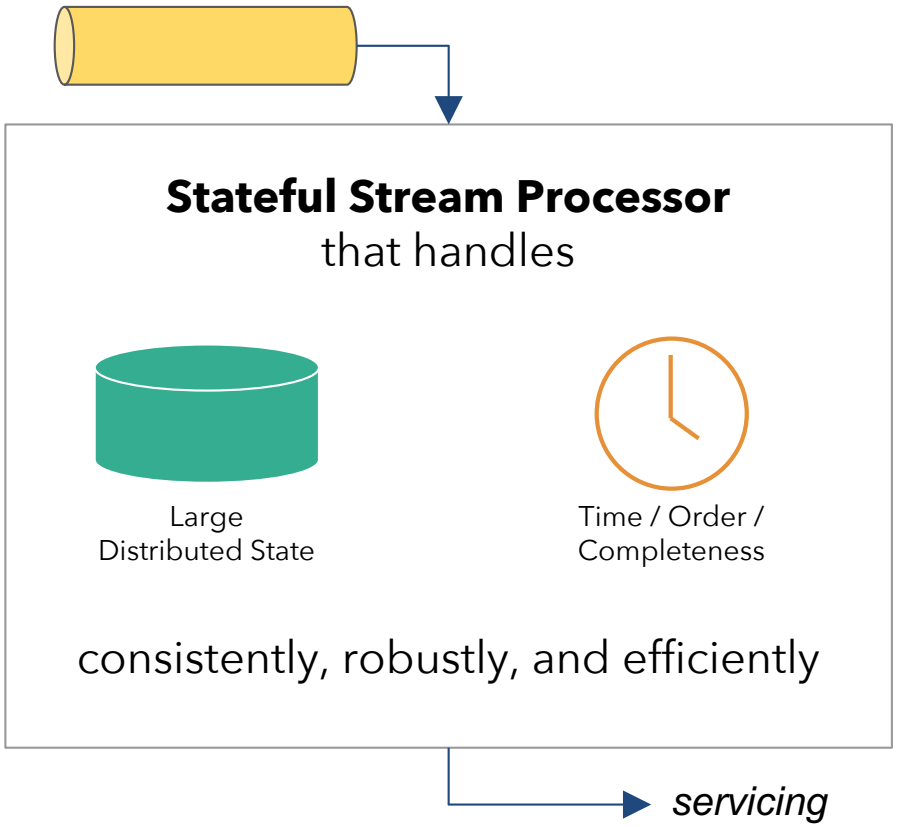
- Consider computing conversion metric **(# of A → B per hour)**
- What if the conversion crossed time boundaries?
  → carry intermediate results to next batch
- What if events come out of order?
  → ???

The ideal way



long running computation

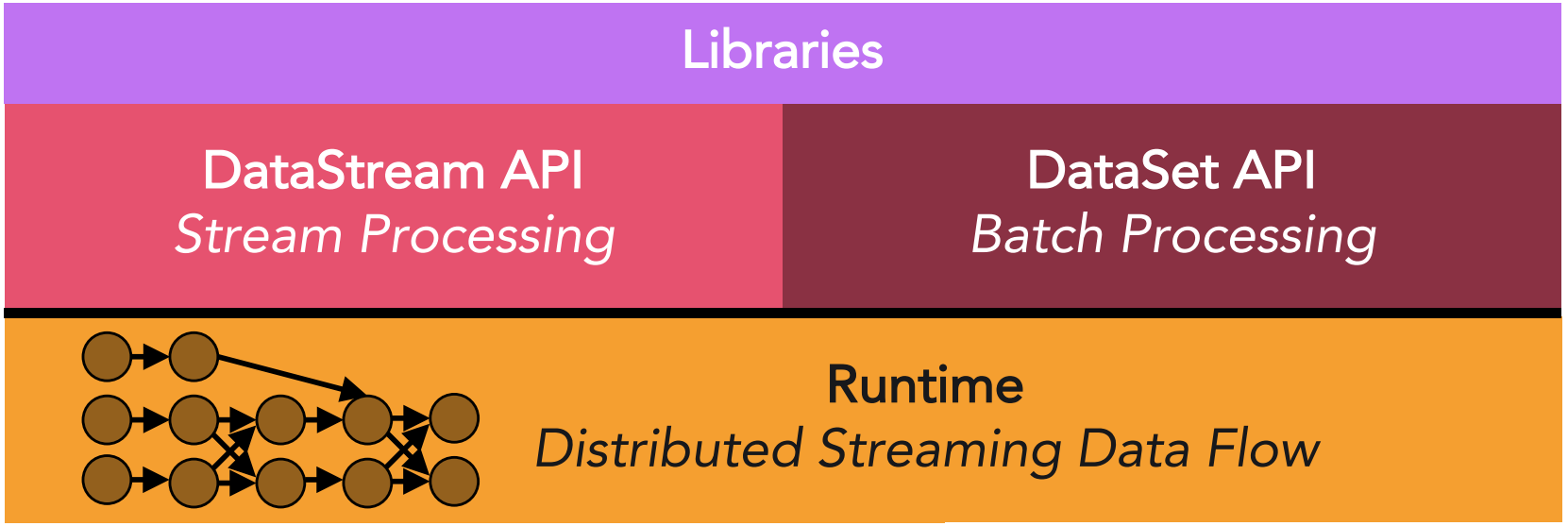
The ideal way (II)



- Stateful stream processing as a **new paradigm** to **continuously process continuous data**
- Produce **accurate** results
- Having results available in real-time (with low latency and high throughput) is a natural consequence of the model
- Process both real-time and historic data using exactly the same application

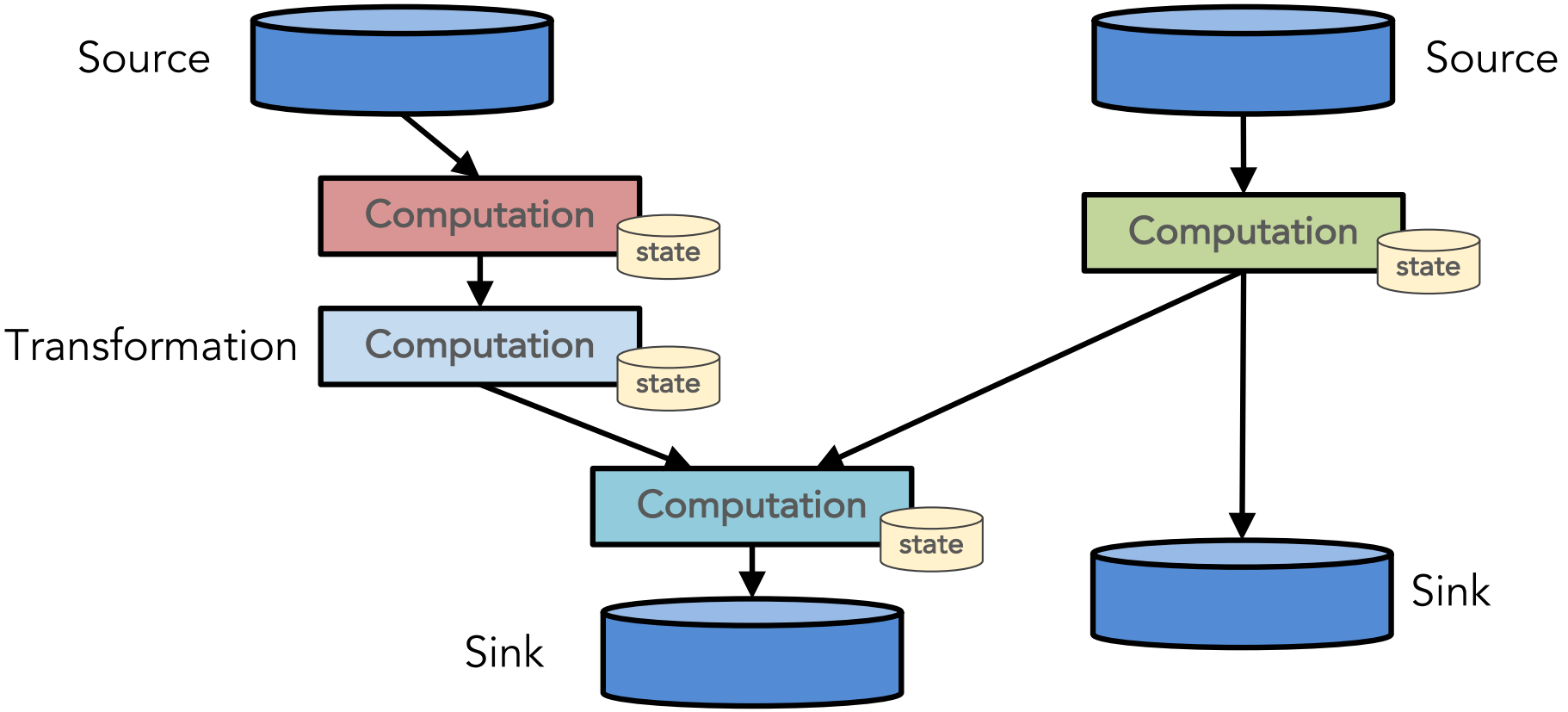
Flink APIs and Runtime

Apache Flink Stack



Streaming and **batch** as **first class citizens**.

Programming Model



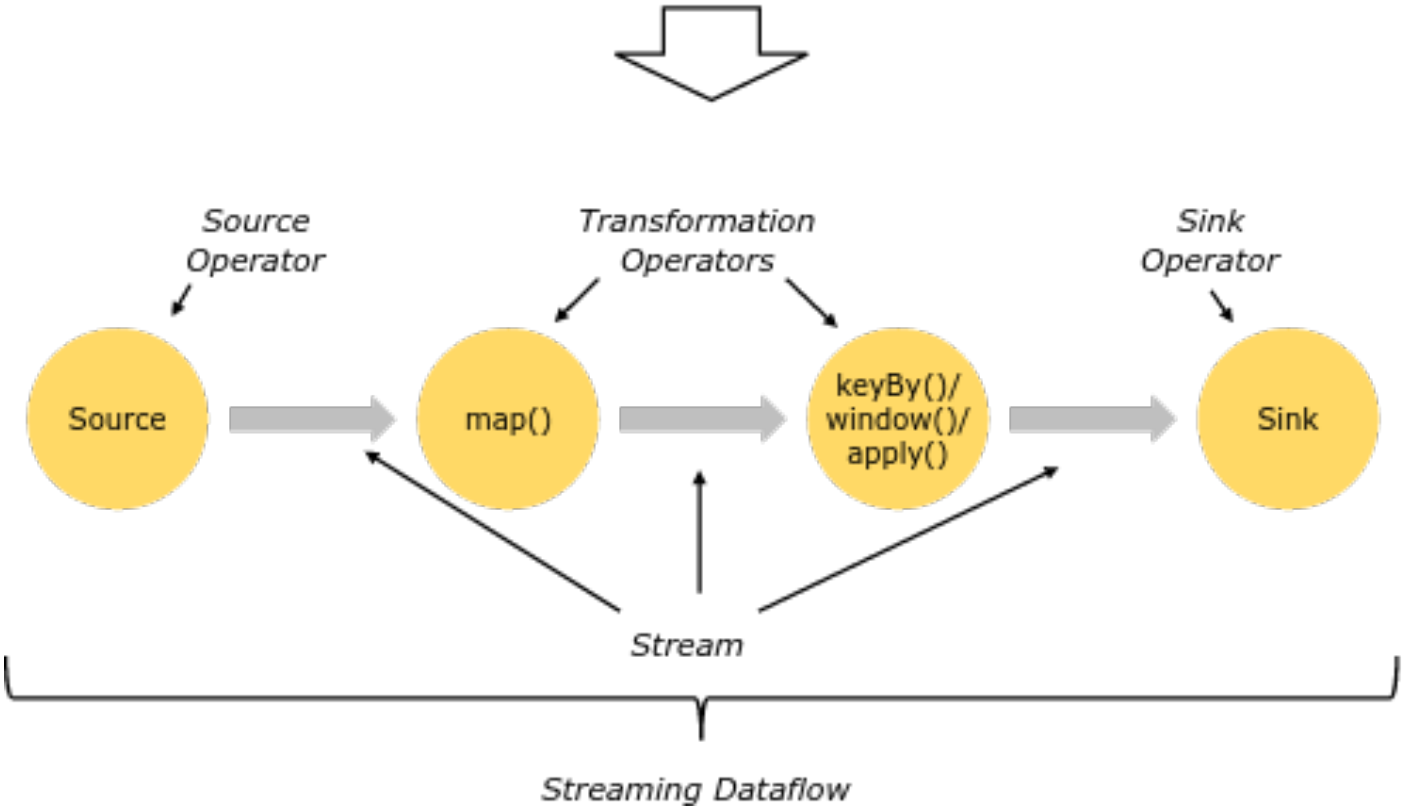
```
DataStream<String> lines = env.addSource(
                                new FlinkKafkaConsumer<>(…));

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));
```

Source

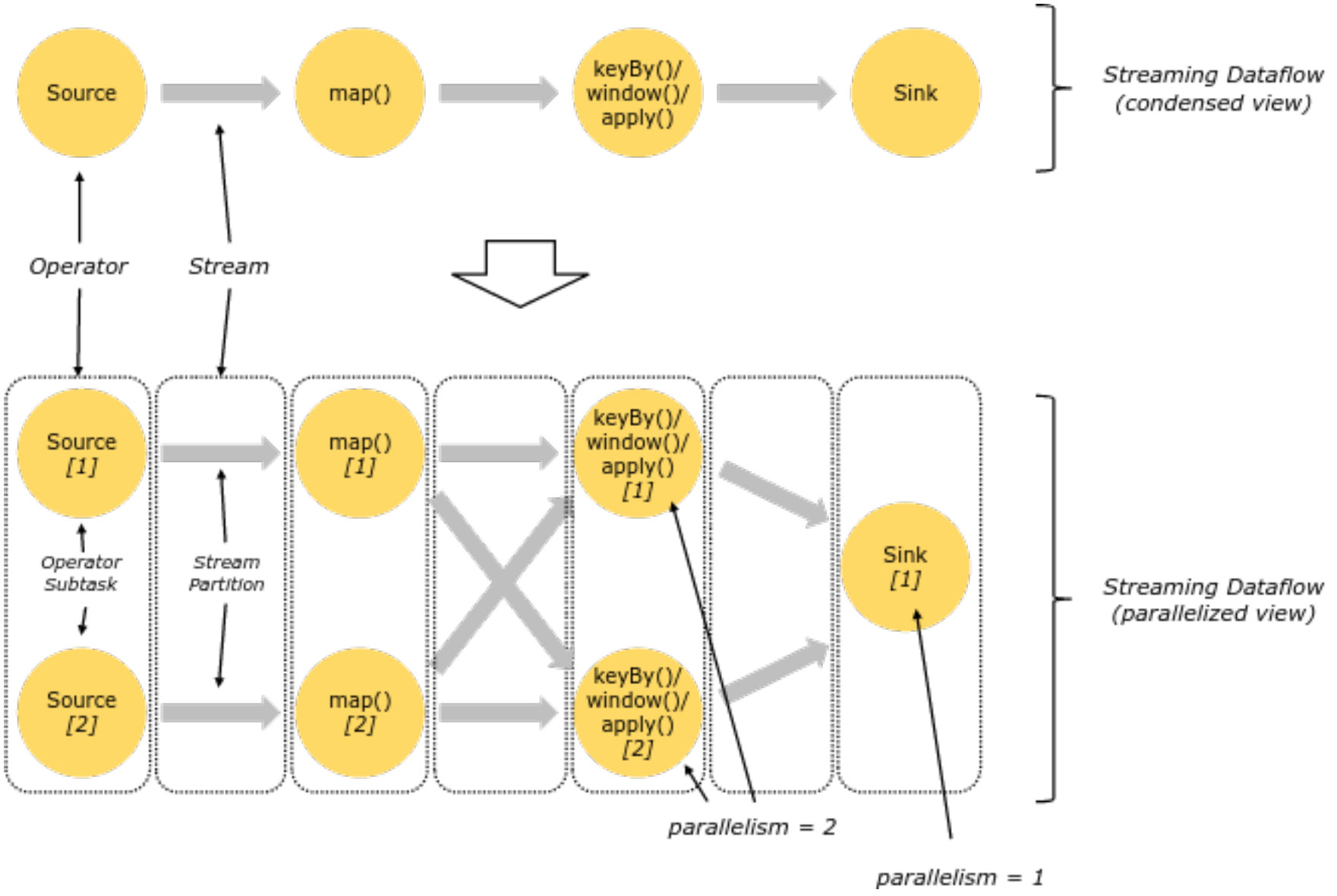
Transformation

Transformation

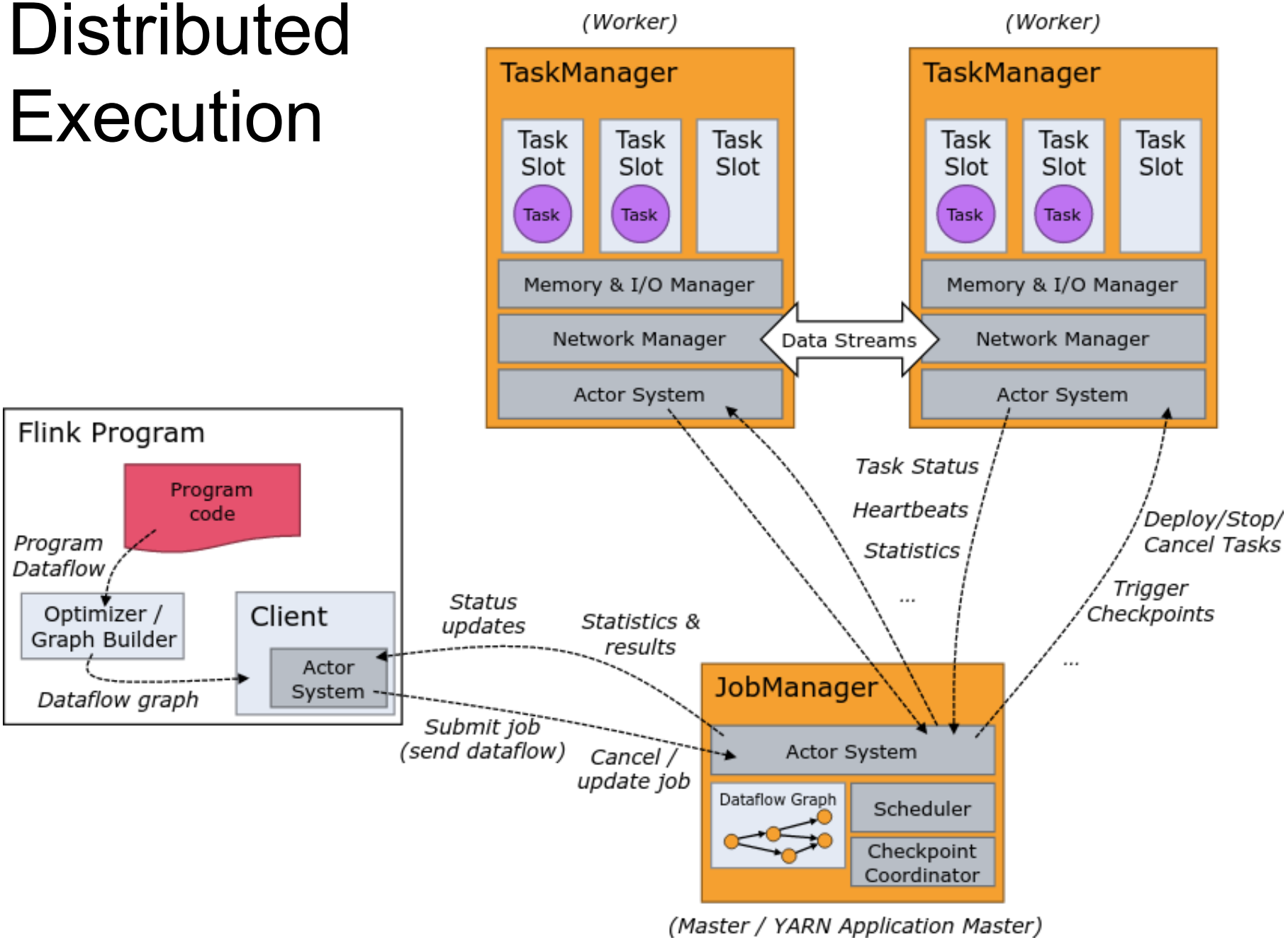
Sink



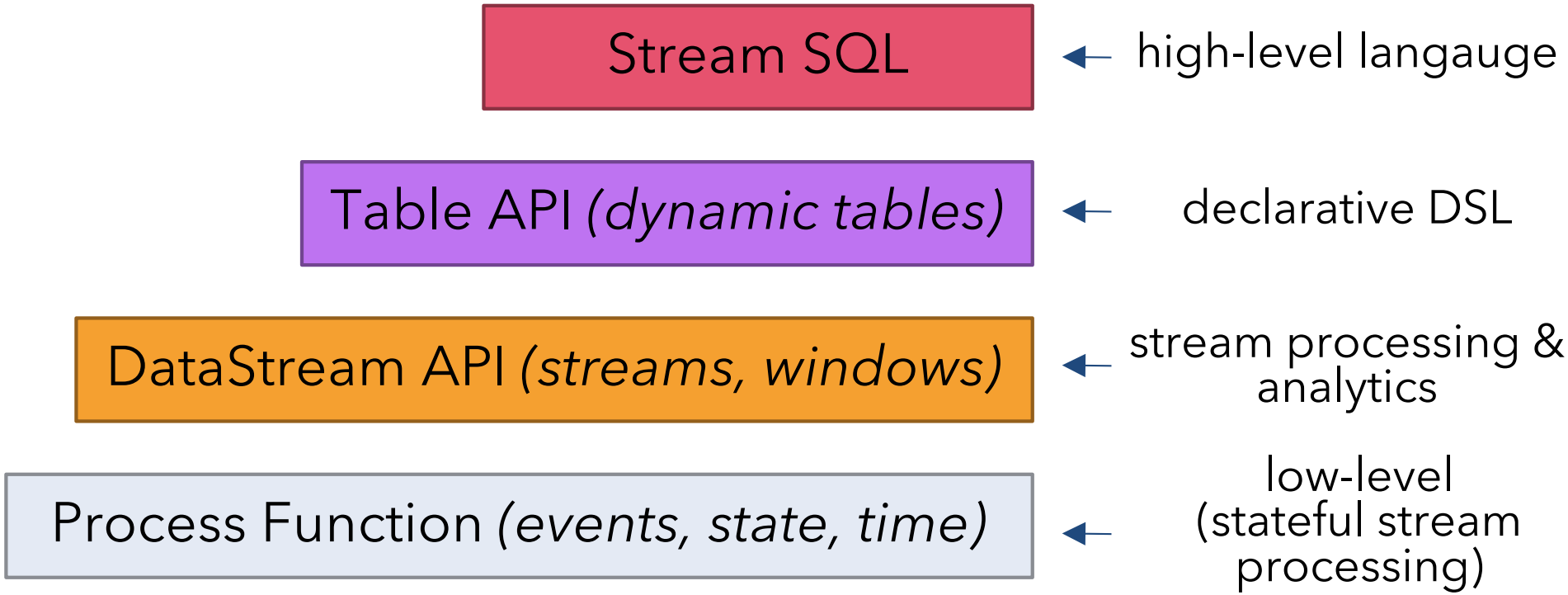
Parallelism



Distributed Execution



Levels of abstraction



Process Function

```
class MyFunction extends ProcessFunction[MyEvent, Result] {

    // declare state to use in the program
    lazy val state: ValueState[CountWithTimestamp] = getRuntimeContext().getState(…)

    def processElement(event: MyEvent, ctx: Context, out: Collector[Result]): Unit = {
        // work with event and state
        (event, state.value) match { … }

        out.collect(…) // emit events
        state.update(…) // modify state

        // schedule a timer callback
        ctx.timerService.registerEventTimeTimer(event.timestamp + 500)
    }

    def onTimer(timestamp: Long, ctx: OnTimerContext, out: Collector[Result]): Unit = {
        // handle callback when event-/processing- time instant is reached
    }
}
```

Data Stream API

```
val lines: DataStream[String] = env.addSource(
                                        new FlinkKafkaConsumer10<>(…))

val events: DataStream[Event] = lines.map((line) => parse(line))

val stats: DataStream[Statistic] = stream
    .keyBy("sensor")
    .timeWindow(Time.seconds(5))
    .sum(new MyAggregationFunction())

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

Table API & Stream SQL

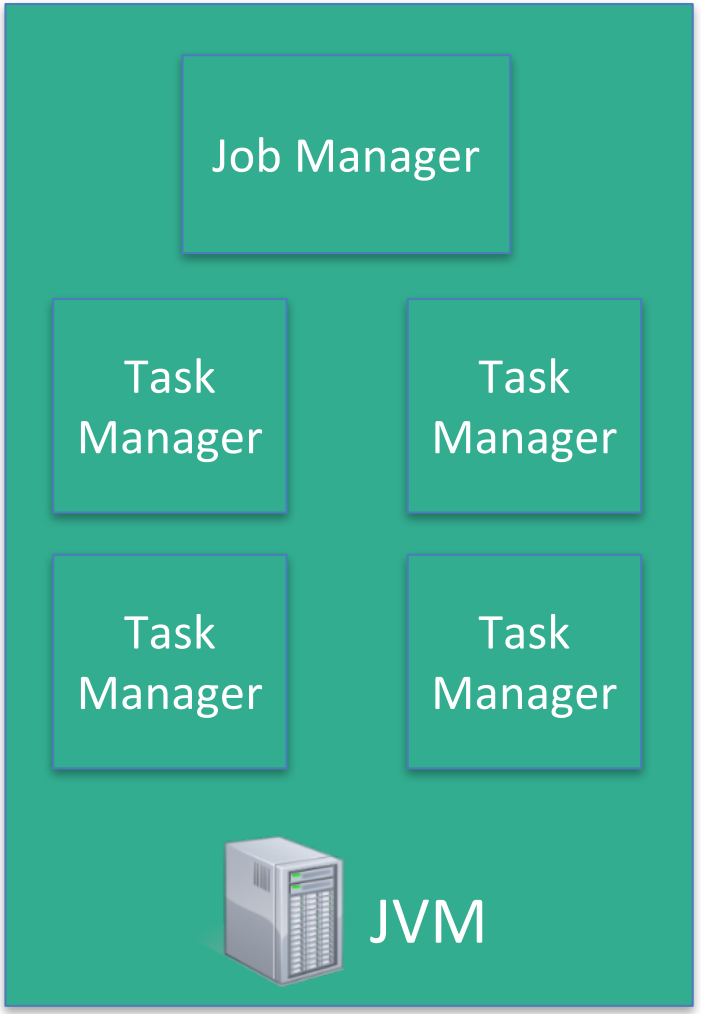
```
// Table API
val tapiResult: Table = tEnv.scan("sensors")    // scan sensors table
 .window(Tumble over 1.hour on 'rowtime as 'w) // define 1-hour window
 .groupBy('w, 'room)                           // group by window and room
 .select('room, 'w.end, 'temp.avg as 'avgTemp) // compute average temperature
```

```
|SELECT room, TUMBLE_END(rowtime, INTERVAL '1' HOUR), AVG(temp) AS avgTemp
|FROM sensors
|GROUP BY TUMBLE(rowtime, INTERVAL '1' HOUR), room
```

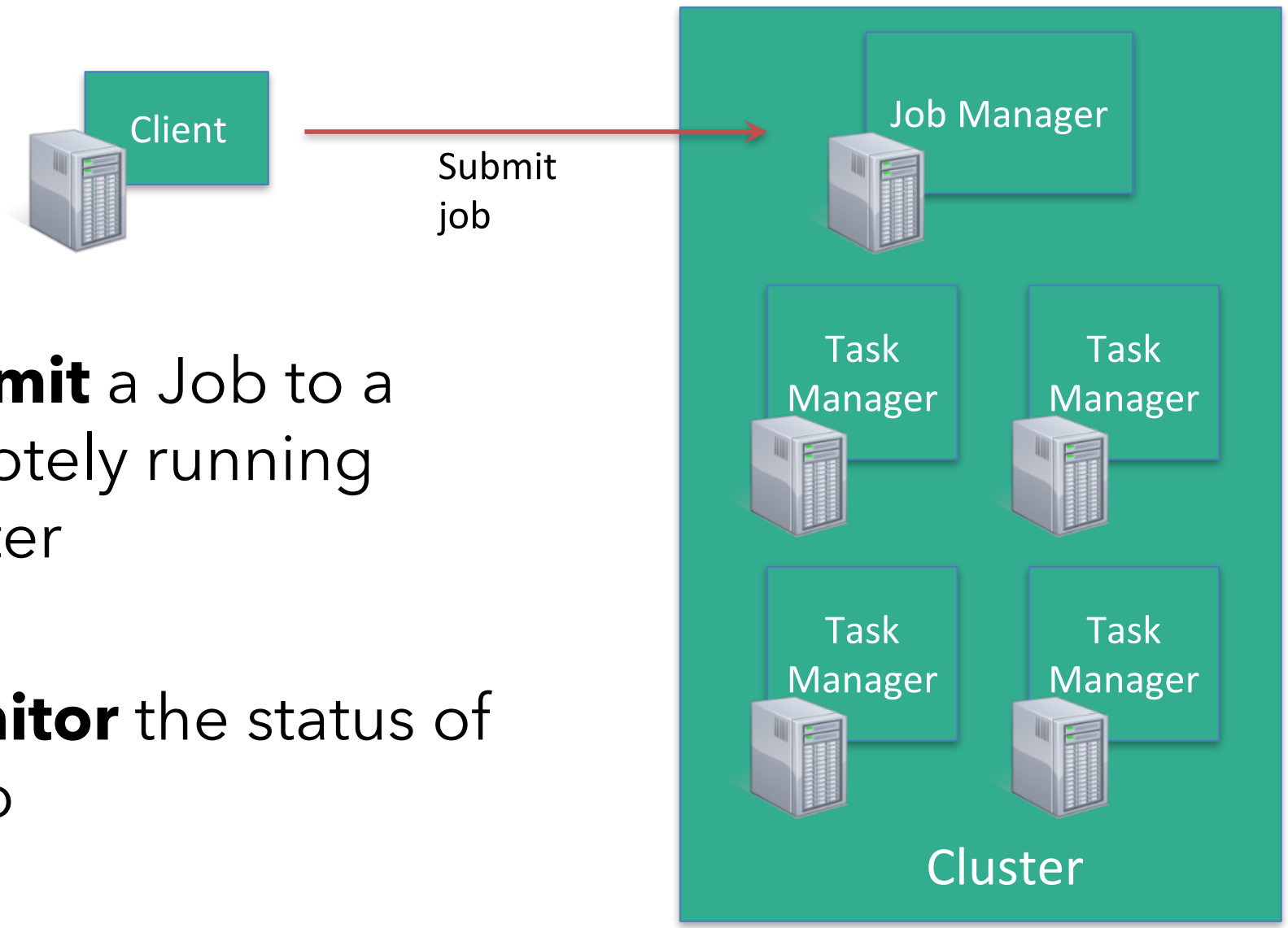
Deployment Options

Local Execution

- Starts local Flink cluster
- All processes run in the same JVM
- Behaves just like a regular Cluster
- Local cluster can be started in your IDE!
- Very useful for developing and debugging



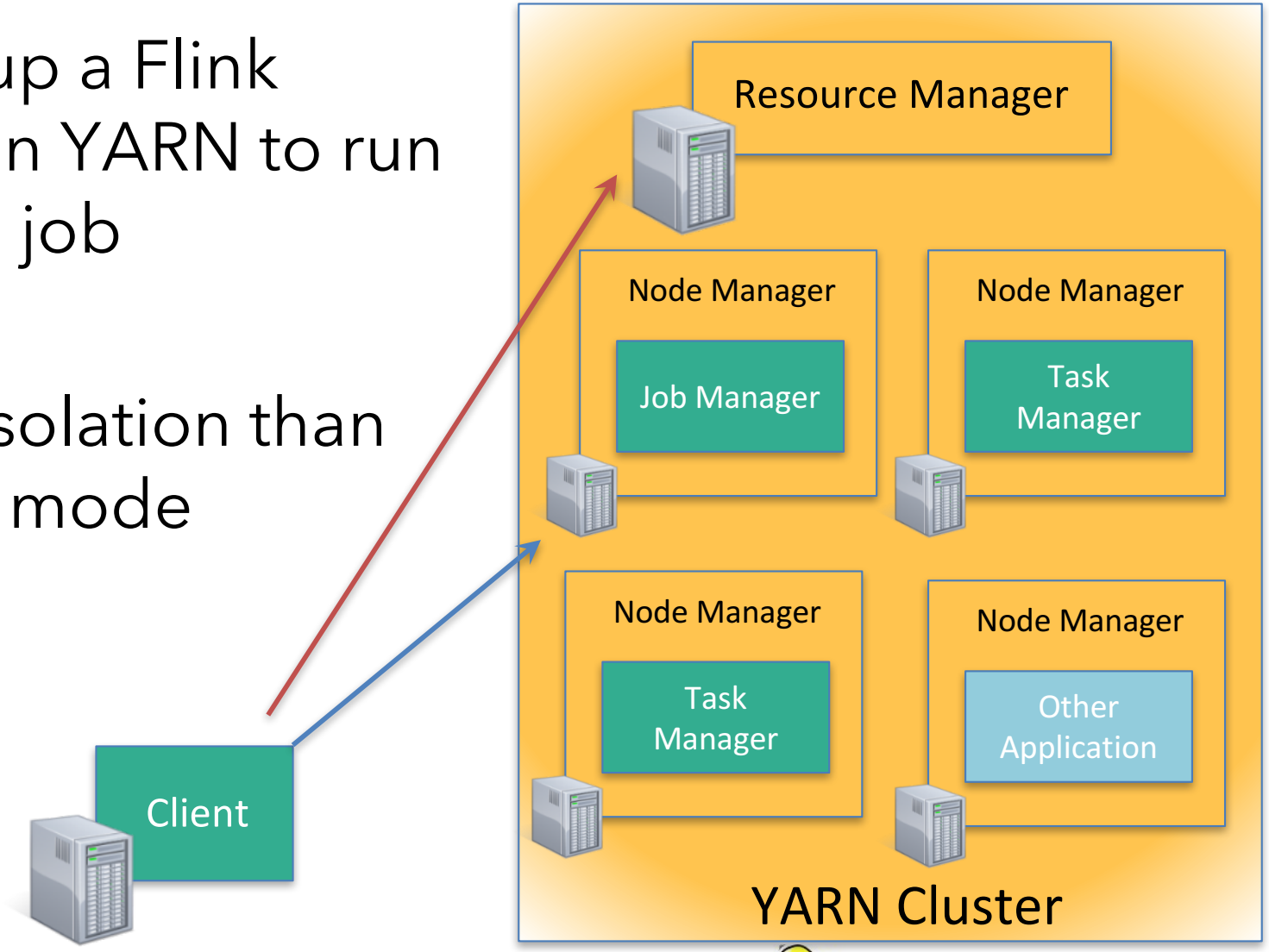
Remote Execution



- **Submit** a Job to a remotely running cluster
- **Monitor** the status of a job

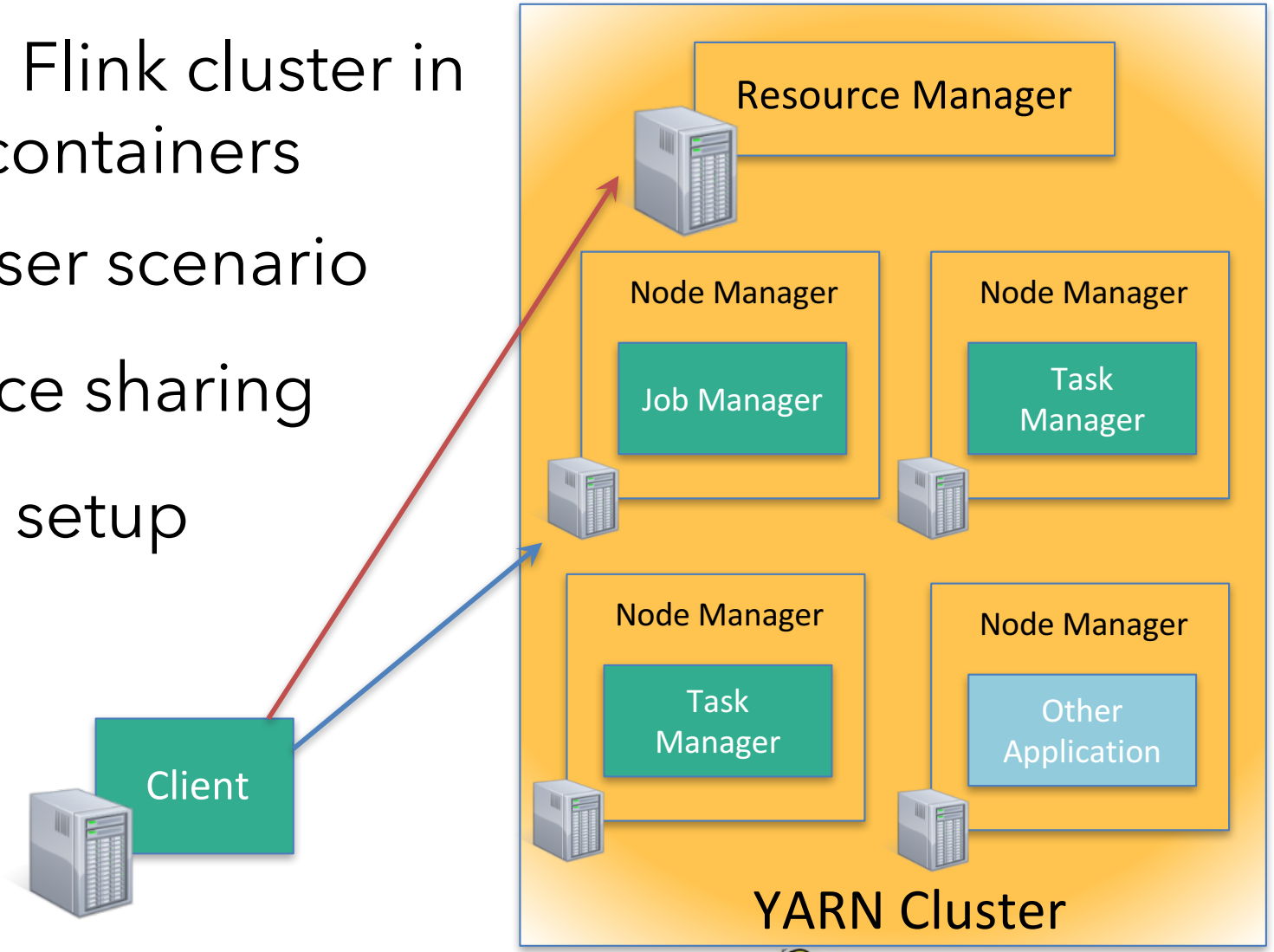
YARN Job Mode

- Brings up a Flink cluster in YARN to run a single job
- Better isolation than session mode



YARN Session Mode

- Starts a Flink cluster in YARN containers
- Multi-user scenario
- Resource sharing
- Easy to setup



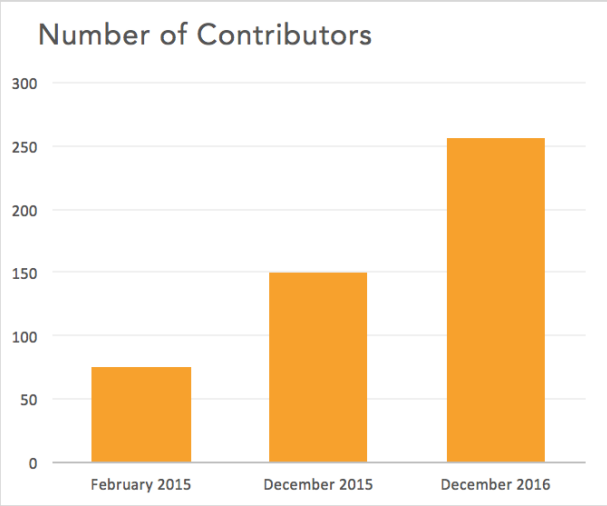
Other Deployment Options

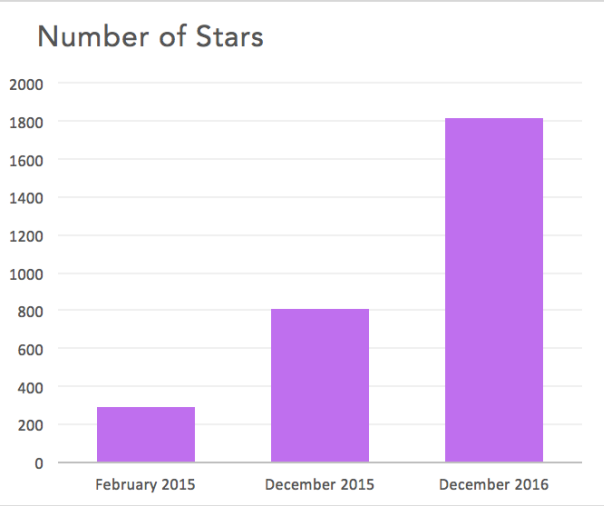
- Apache Mesos
  - Either with or without DC/OS
- Amazon Elastic MapReduce
  - Available in EMR 5.1.0
- Google Compute Engine
  - Available via bdutil
- Docker / Kubernetes

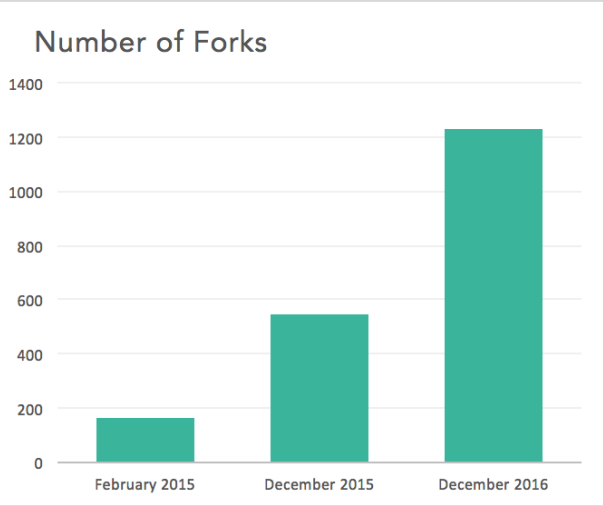
Flink in the real world

Flink community

Github







41 meetups
16,544 members

Powered by Flink

Zalando, one of the largest ecommerce companies in Europe, uses Flink for real-time business process monitoring.

King, the creators of Candy Crush Saga, uses Flink to provide data science teams with real-time analytics.

Alibaba, the world's largest retailer, built a Flink-based system (Blink) to optimize search rankings in real time.

Bouygues Telecom uses Flink for real-time event processing over billions of Kafka messages per day.

See more at flink.apache.org/poweredby.html

Largest job has > 20 operators, runs on > 5000 vCores in 1000-node cluster, processes millions of events per second

Complex jobs of > 30 operators running 24/7, processing 30 billion events daily, maintaining state of 100s of GB with exactly-once guarantees

30 Flink applications in production for more than one year. 10 billion events (2TB) processed daily

What is being built with Flink?

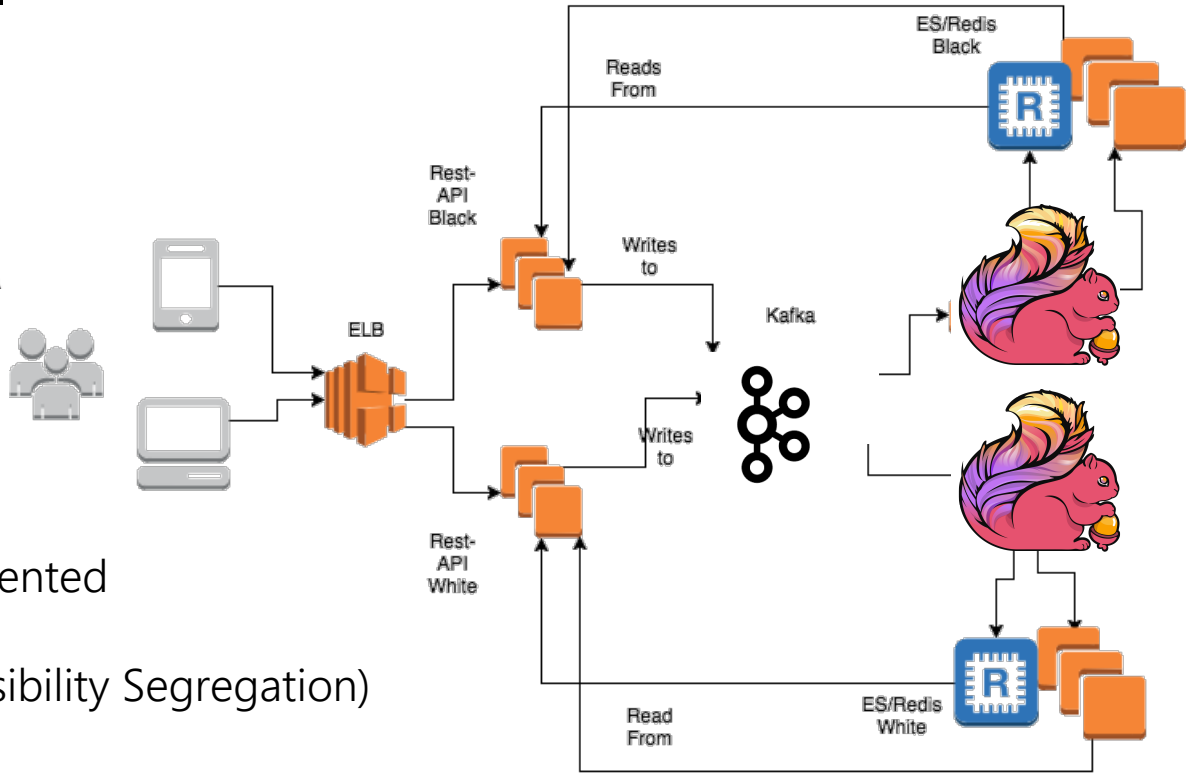
- First wave for streaming was lambda architecture
  - Aid batch systems to be more real-time
- Second wave was analytics (real time and lag-time)
  - Based on distributed collections, functions, and windows
- The next wave is much broader:

  A new architecture for event-driven applications

@

D_
TRB

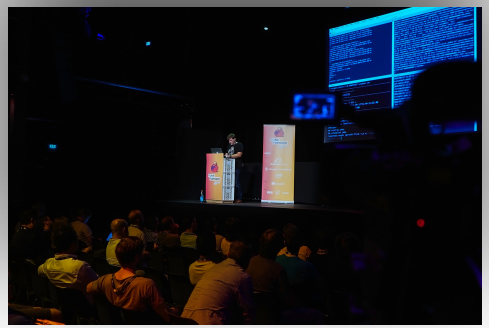
THE SOCIAL NETWORK
FOR PETROLHEADS



Complete social network implemented
using event sourcing and
CQRS (Command Query Responsibility Segregation)

Flink Forward 2016

Flink Forward 2017



San Francisco

- 10-11 April 2017
- The first Flink Forward event outside of Berlin
- Talks are online at sf.flink-forward.org/

Berlin

- 11-13 September 2017
- Over 350 attendees last year
- Registration opening soon!

http://dataartisans.github.io/flink-training/