Troubleshooting & Operations

Nico Kruber, Solutions Architect & Apache Flink committer
Alexander Fedulov, Solutions Architect

Agenda

Morning

- 10:00am - 11:30am: Getting Started
  - Flink's Distributed Architecture "Recap"
  - Getting started with the hands-on exercises
- **Coffee Break**
- 11:45am - 1:15pm: Event-Time & Latency
  - Metrics & Monitoring
  - Watermarking
  - Latency

Agenda

Afternoon

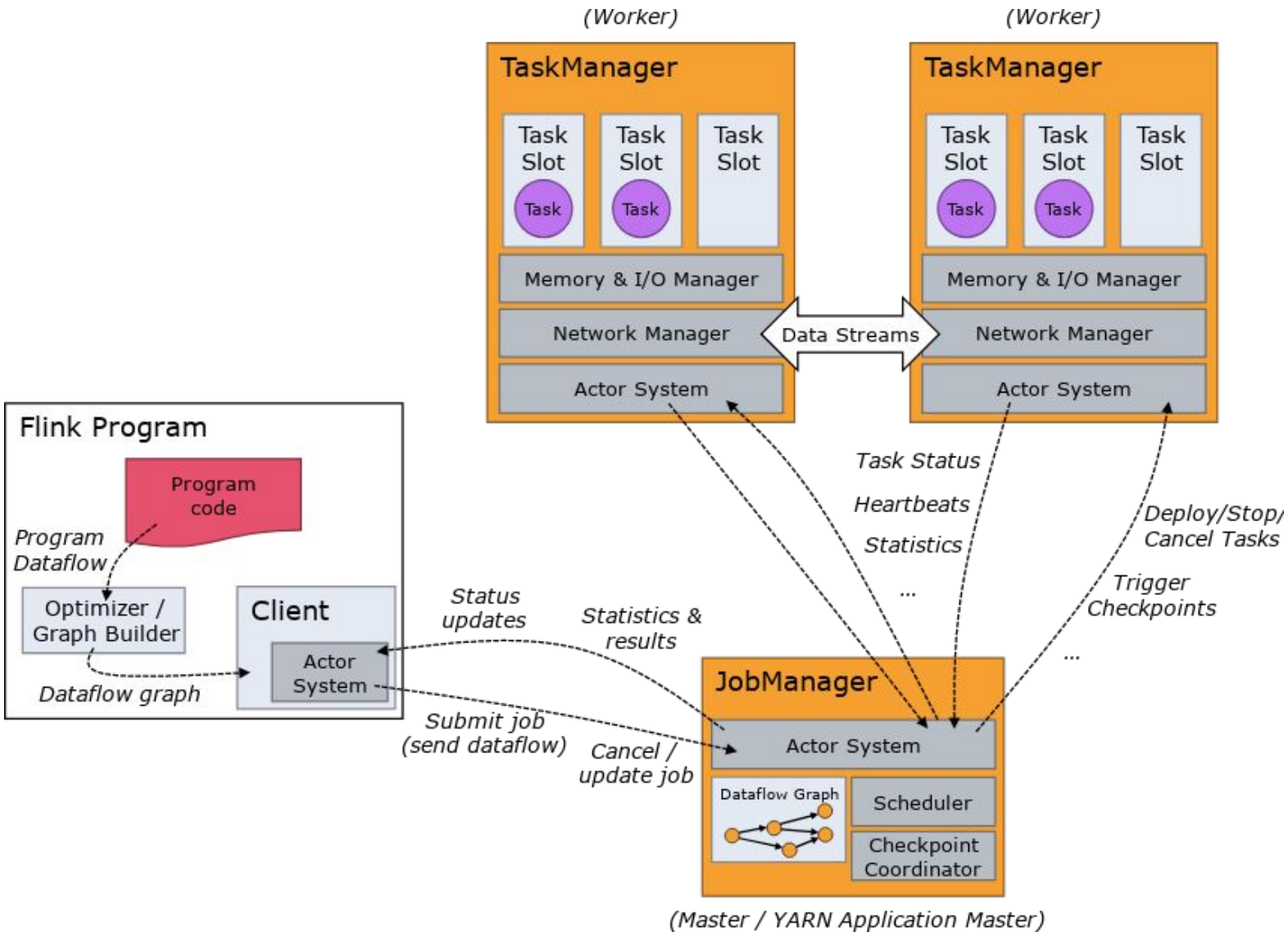
- 2:15pm - 3:45pm: Performance Tuning
  - Serialization
  - Throughput Optimization
- **Coffee Break**
- 4:00pm - 5:30pm: State, Memory, & Serialization
  - State & State Backends
  - Object-Reuse & Serialization (2)

Flink's Distributed Architecture

Nico Kruber, Solutions Architect & Apache Flink committer

 |

Cluster Components



Job Definition

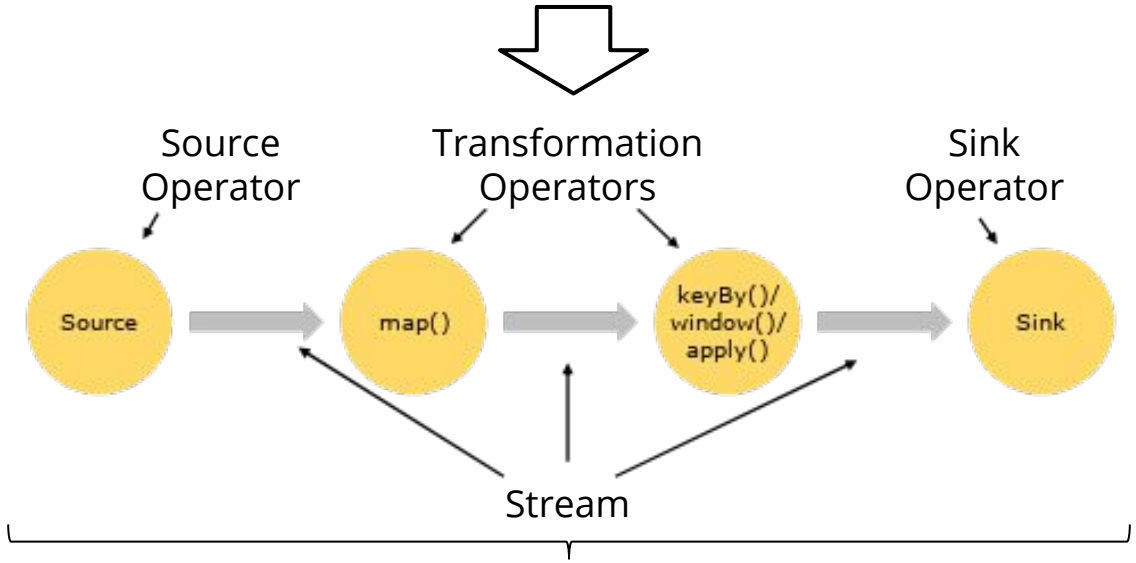
```
DataStream<String> lines = env.addSource(
  new FlinkKafkaConsumer<>(...));

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

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

stats.addSink(new BucketingSink<>(path));
```

Source

Transformation

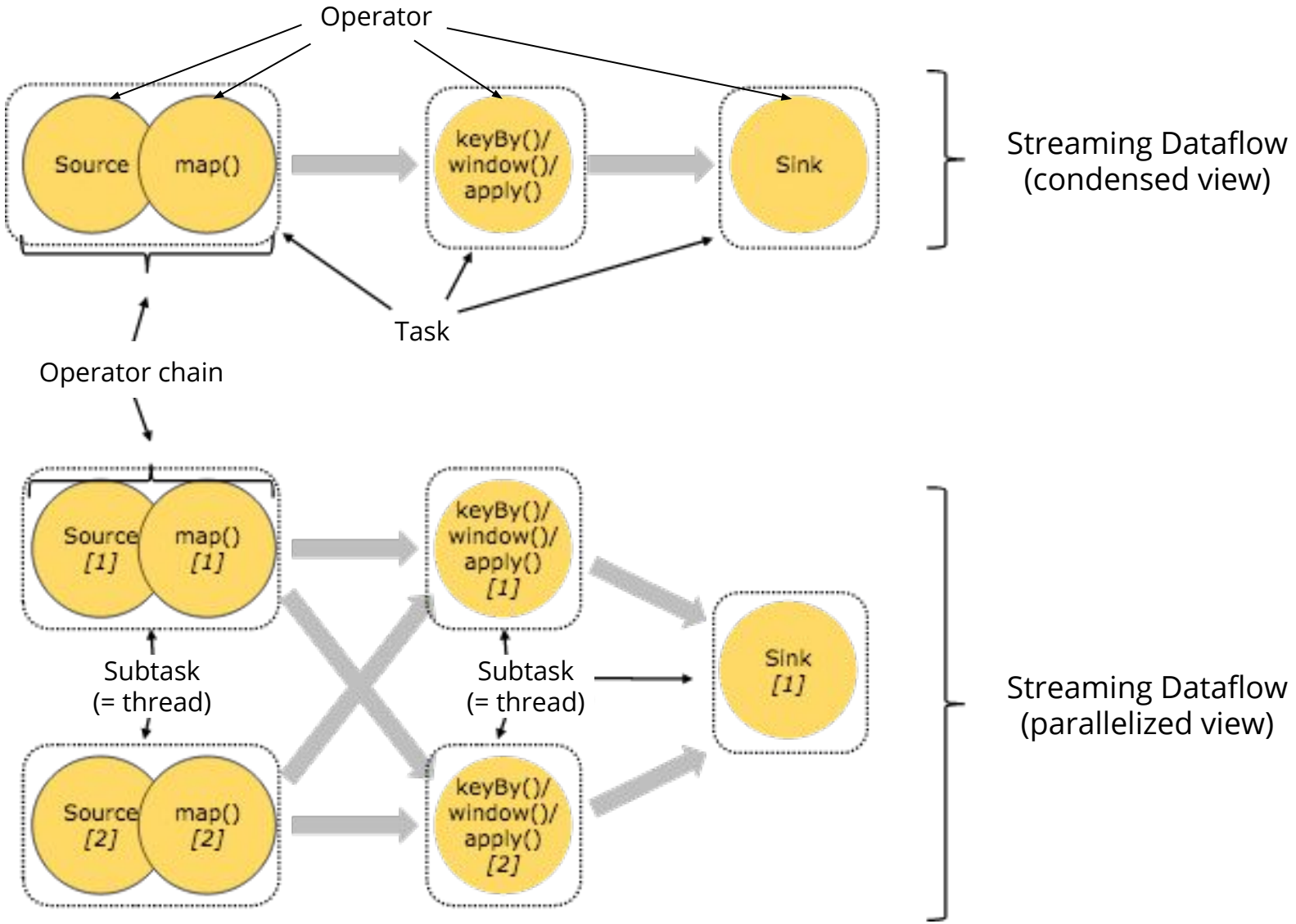
Transformation

Sink

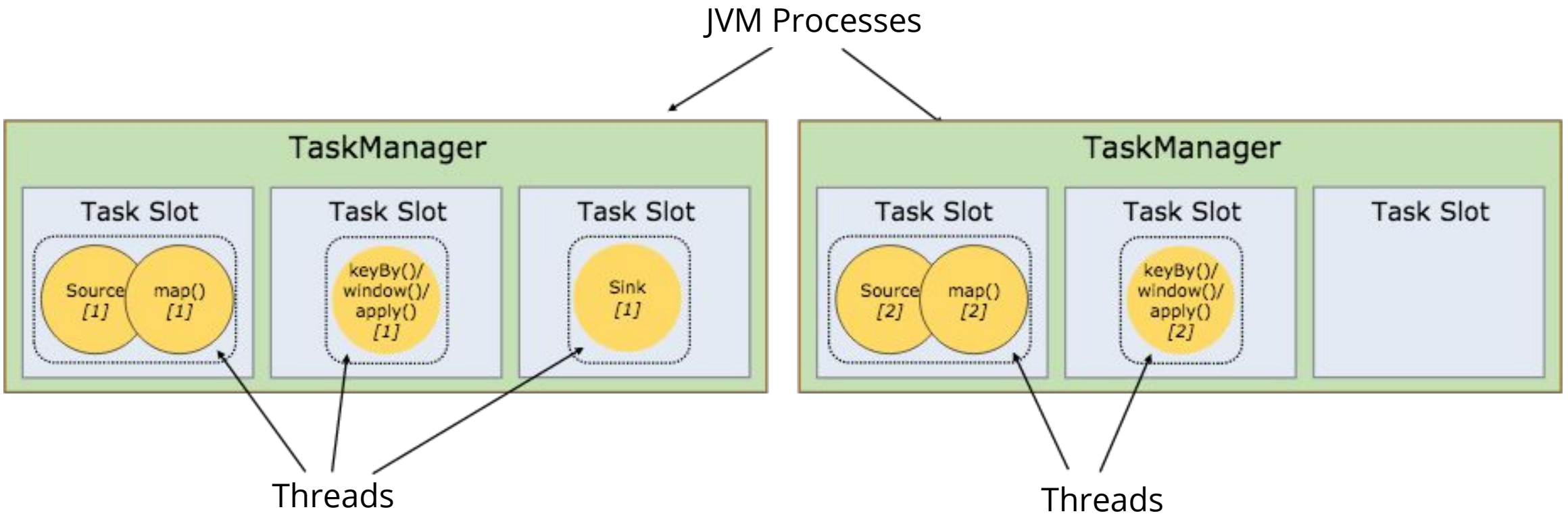


Streaming Dataflow

Job Components & Operator chaining

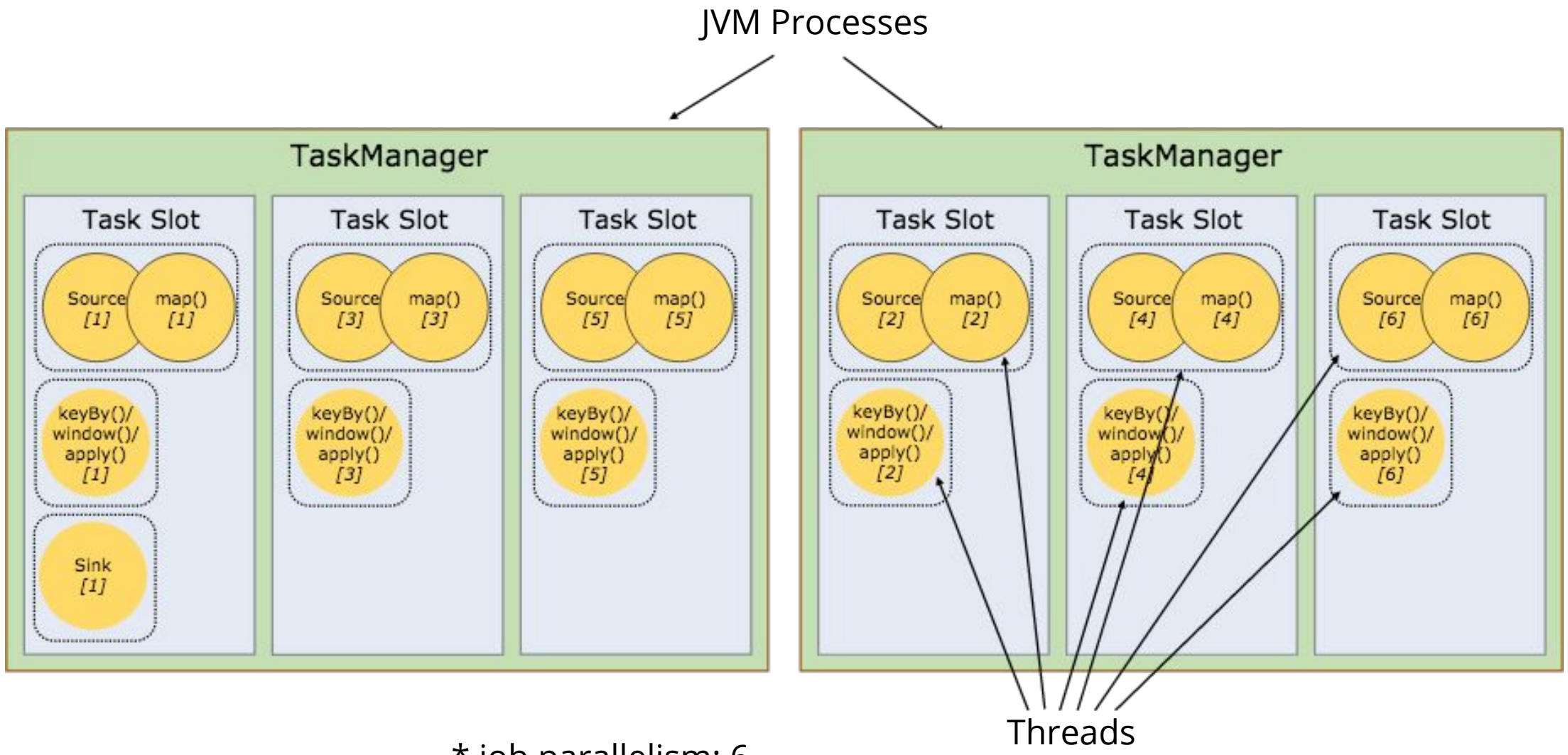


Task Deployment: Slots



* job parallelism: 2

Task Deployment: Slot Sharing



* job parallelism: 6

Fault Tolerance Guarantees

What happens if a worker goes down?

Flink supports different levels of guarantee for failure recovery:

Exactly once

- Each event affects the managed state exactly once.
- **Note**: This does *not* mean that events are processed exactly once!

At least once

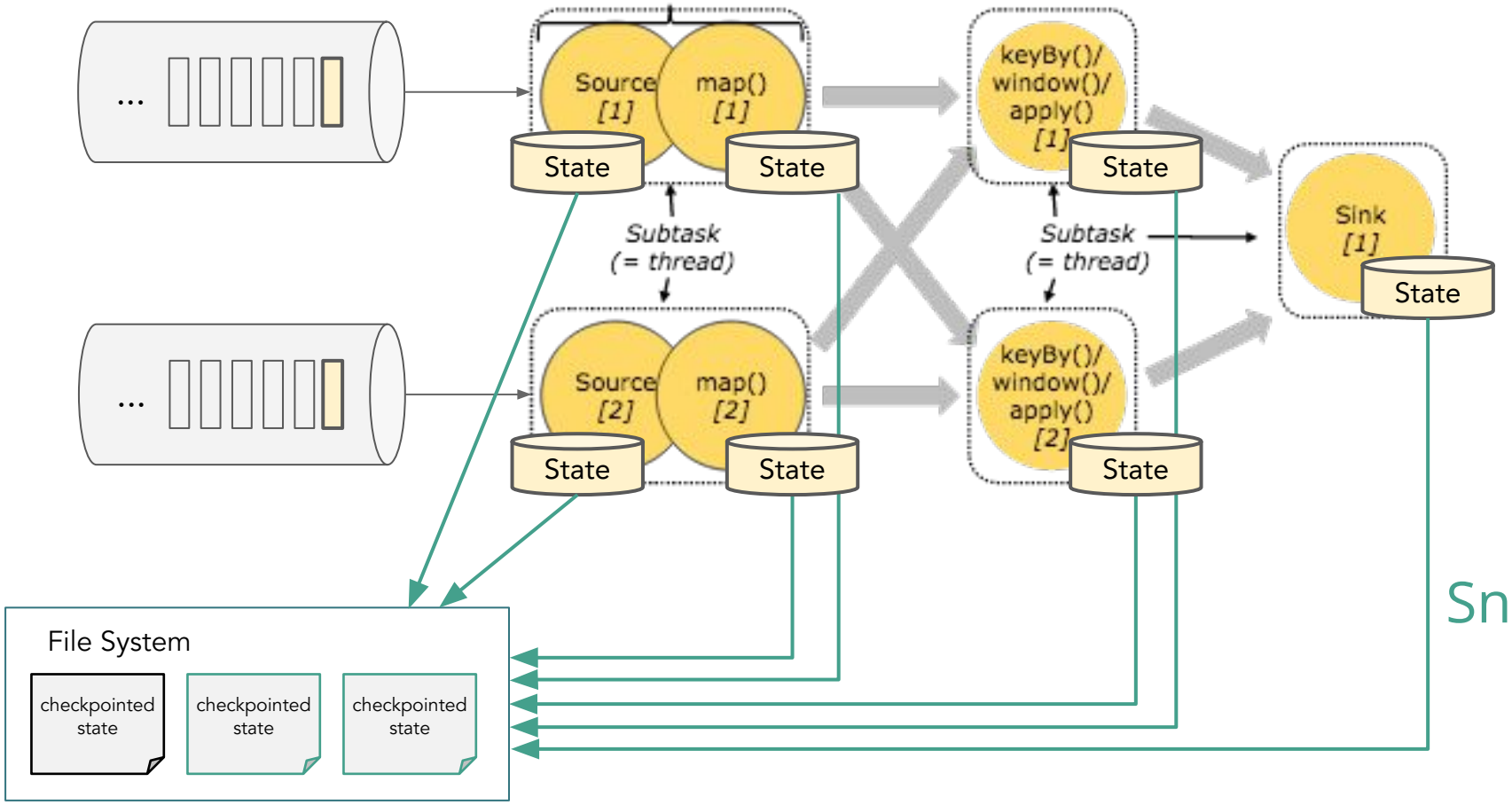
- Each event affects the declared state of a program at least once

Deactivated / None / At most once

- All state is lost in case of a failure

Job Lifecycle & Fault Tolerance

While running



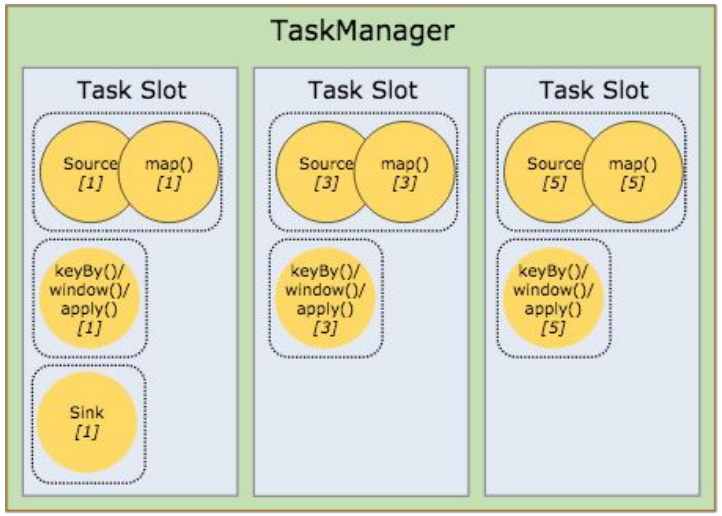
Snapshot*

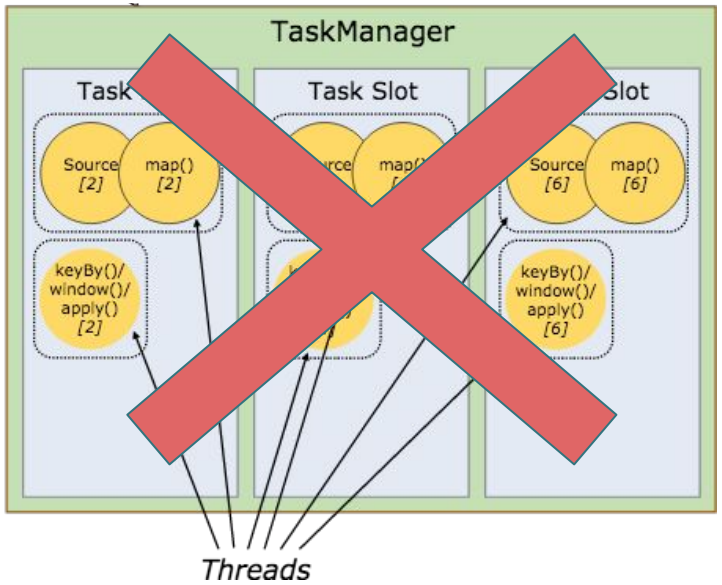
* Asynchronous Barrier Snapshotting
→ "checkpoint barriers" flow with data

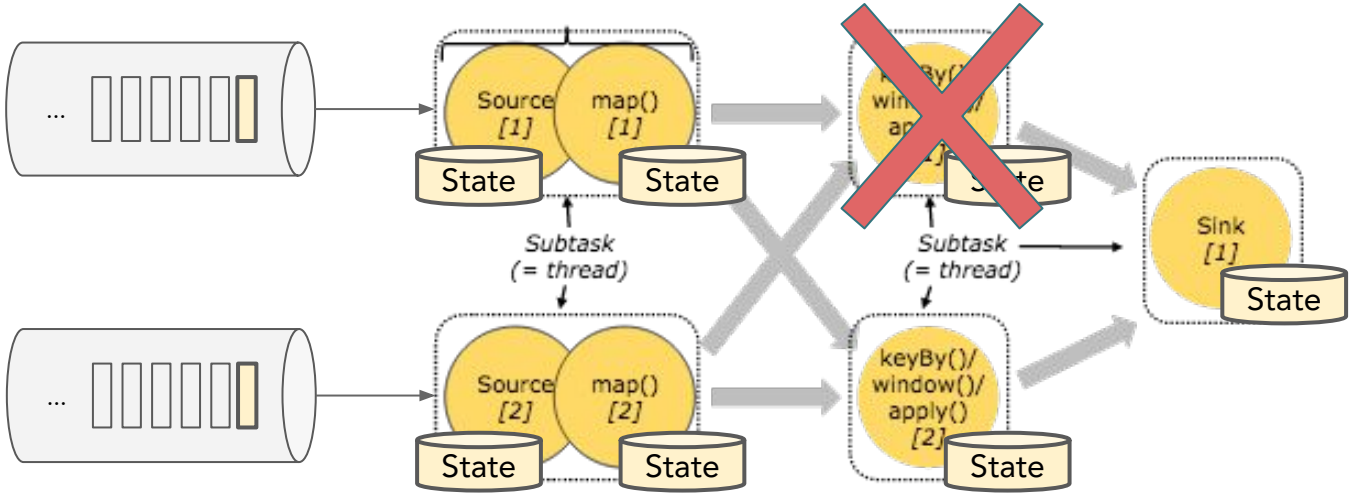
Job Lifecycle & Fault Tolerance

On Failure

- TaskManager fails
  ⇒ cancel job on all TMs,
  ⇒ restart
  (may require new TMs/slots)





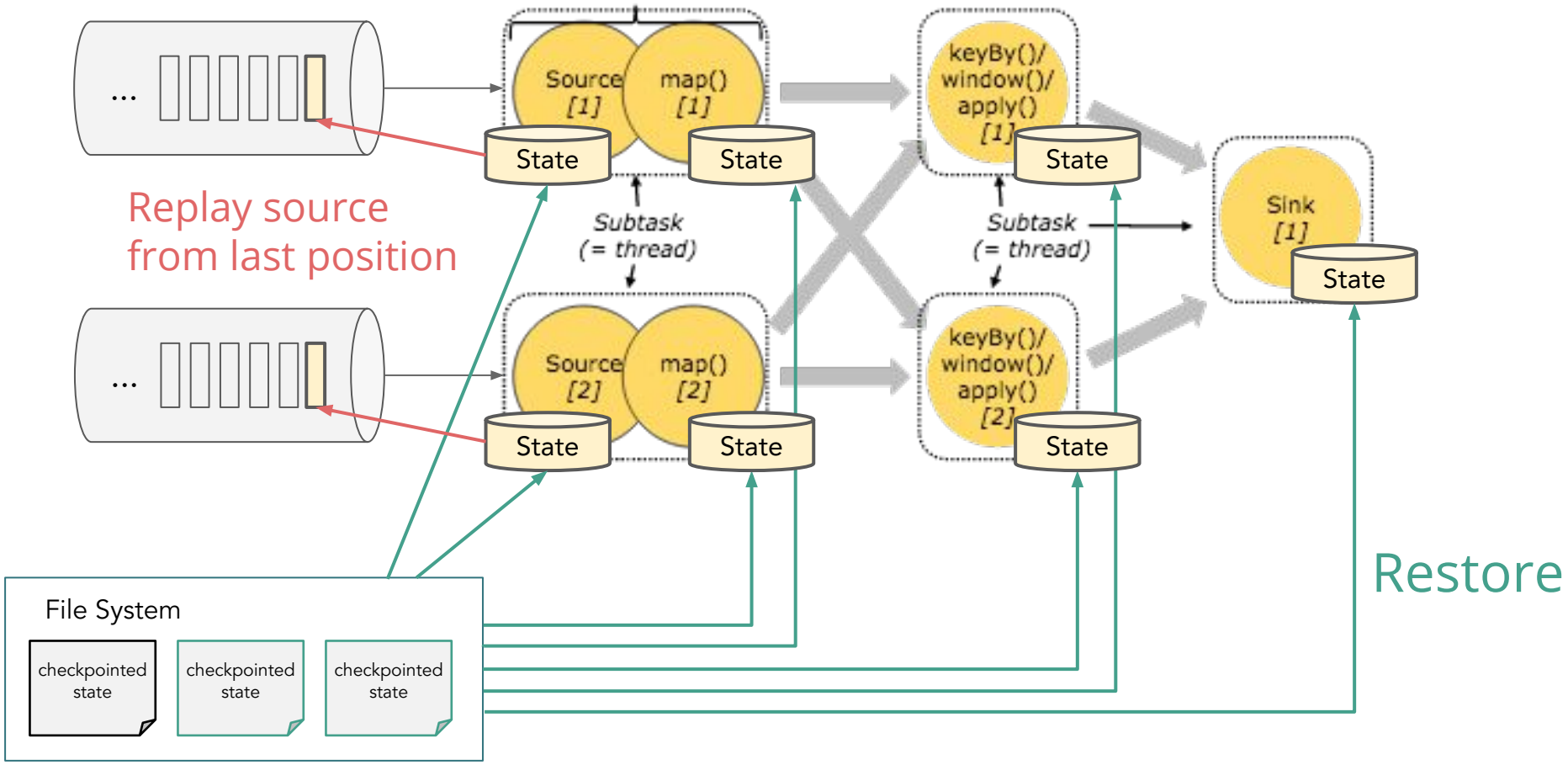


- Operator code fails
  ⇒ cancel job on all TMs,
  ⇒ restart

 |

Job Lifecycle & Fault Tolerance

After restart



 |

Exercises

Exercises

Getting Started

1. `Preparations`
   a. `git clone https://github.com/ververica/flink-training-troubleshooting`
   b. **`git pull origin master`**
   c. Import project into IDE
2. Run the `TroubledStreamingJob` locally by executing `TroubledStreamingJobRunner` under `test/`

   Note: The job is meant to restart periodically.
3. Run `TroubledStreamingJob` on Ververica Platform (see next slide)

Exercises

Getting Started

1. Login into Ververica Platform 2.0 (pre-release!)
   - http://<your-training-ip>/ (Credentials provided)
2. Upload JAR
   - `mvn clean package`
   - Upload `target/flink-training-troubleshooting-0.1.jar` via ApplicationManager's Artifacts tab
3. Verify "Troubled Flink Job" Deployment in Ververica Platform
   - Upgrade Strategy: **STATELESS**, Restore Strategy: **NONE**, Parallelism: **4**
   - Jobmanager/Taskmanager CPU: **1/1 (defaults)**
   - Jobmanager/Taskmanager Memory: **2G/2G (defaults)**
4. Adapt Jar URI in Deployment in Application Manager if you change the name
5. Start "Troubled Flink Job" Deployment in Ververica Platform

Getting Started Walk-Through

Exercises

Debugging Frequent Job Failures

Exercise 1
As we just saw, `TroubledStreamingJob` is restarting frequently. Locate the problem and deploy a fix for it to Ververica Platform.

Note: Classes/Methods annotated with **`@DoNotTouchThis`** should not be modified during the training.

nico@ververica.com www.ververica.com @VervericaData