From Apache Flink®
1.3 to 1.4

Till Rohrmann
till@data-artisans.com
@stsffap

dataArtisans

dataArtisans

Original creators of Apache Flink®

dA PLATFORM

Providers of dA Platform 2, including open source Apache Flink + dA Application Manager

Overview

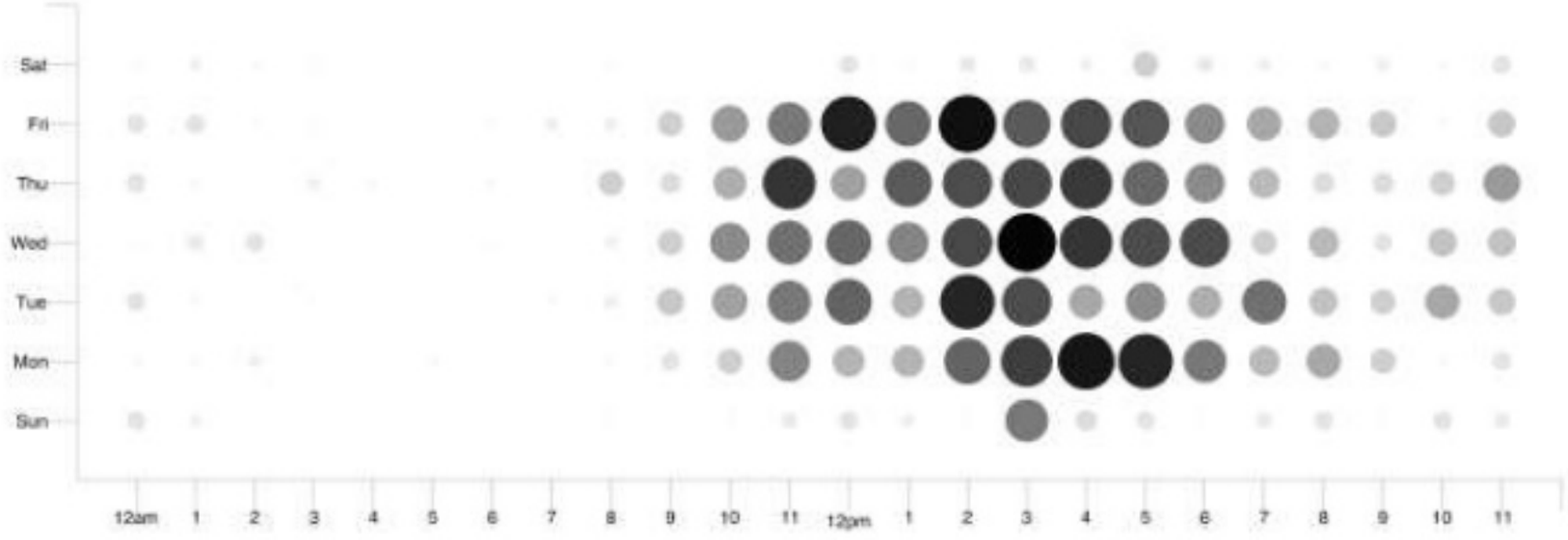
- Apache Flink 1.3 – Previously on Apache Flink
- Apache Flink 1.4 – What's happening now?
- Apache Flink 1.5+ – Next on Apache Flink

Apache Flink 1.3

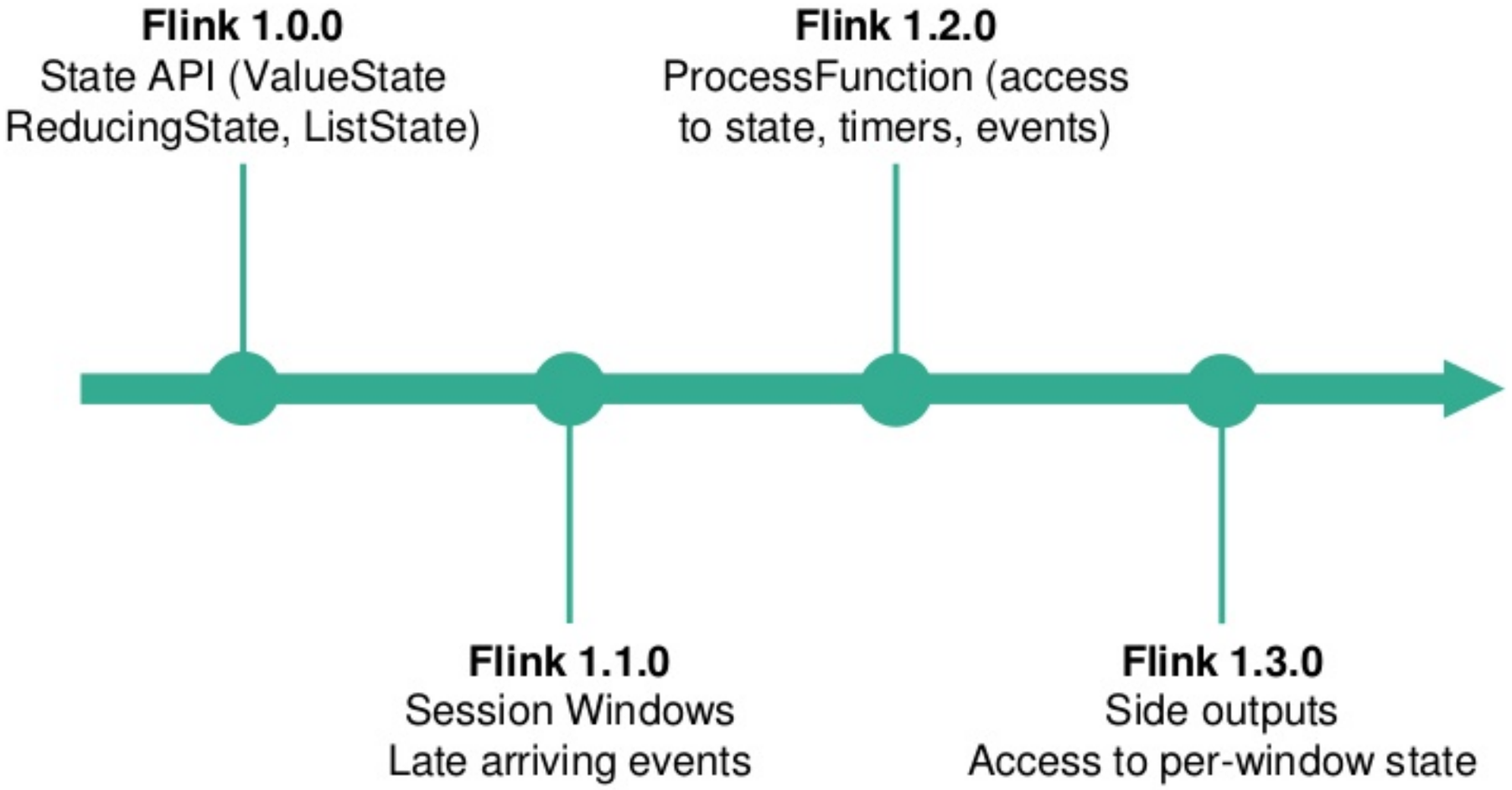
Previously on Apache Flink

Apache Flink 1.3 in Numbers

- 141 contributors (no deduplication)
- 1400 commits
- >= 680 resolved JIRA issues
- +261813 / -65646 LOC

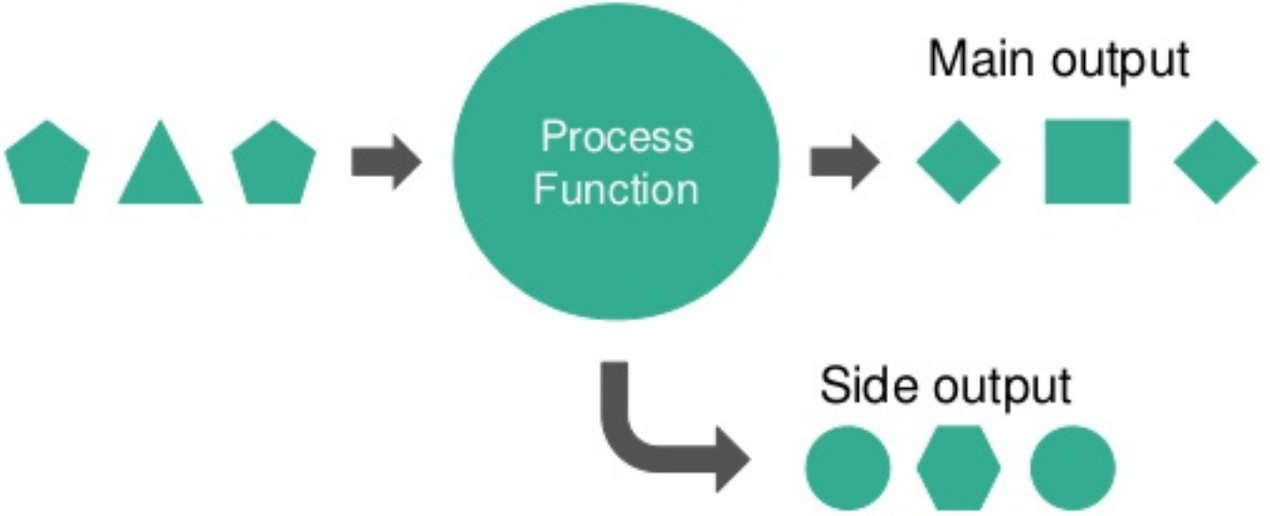


Evolution of Flink's API

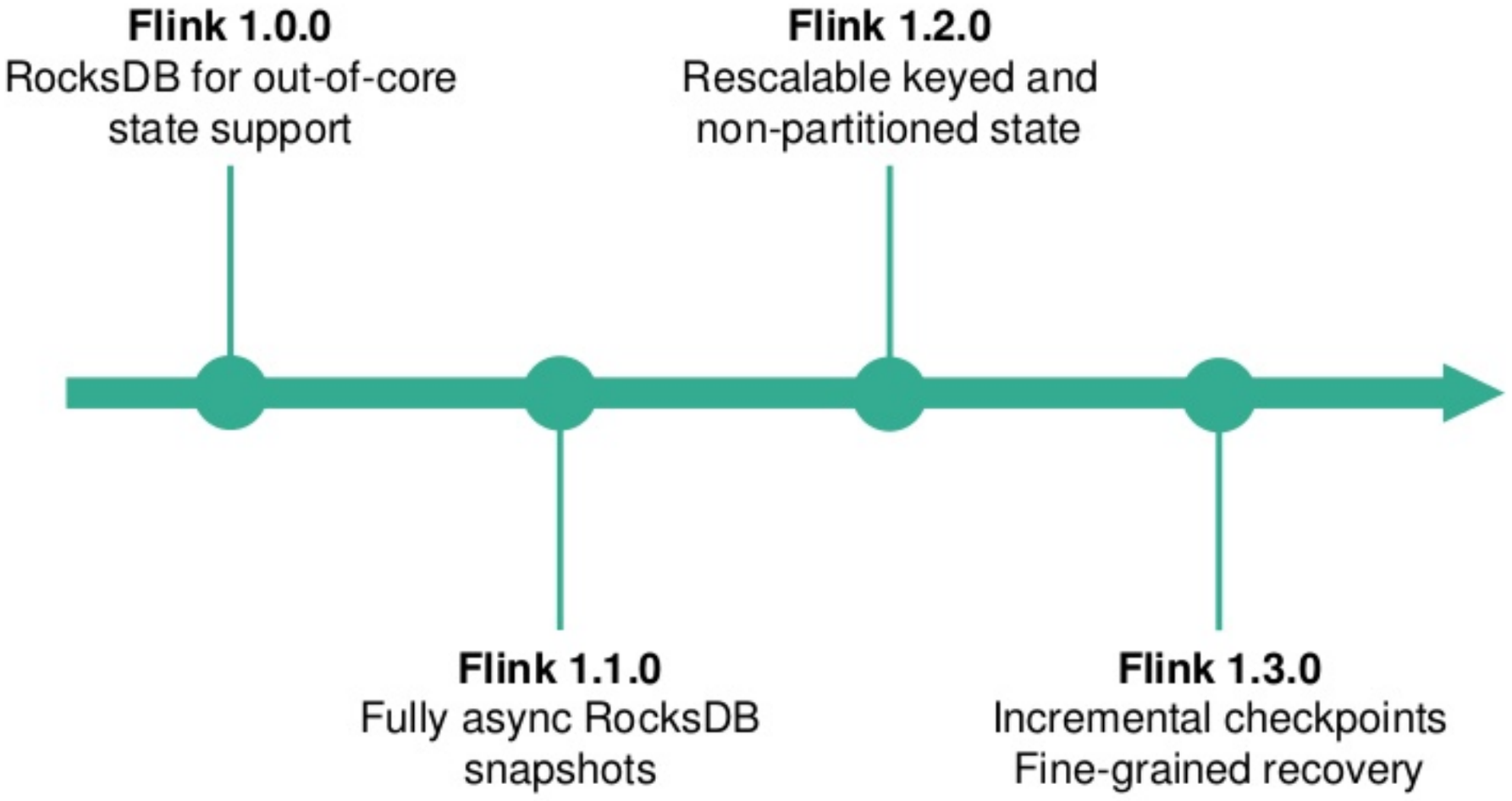


Side Outputs

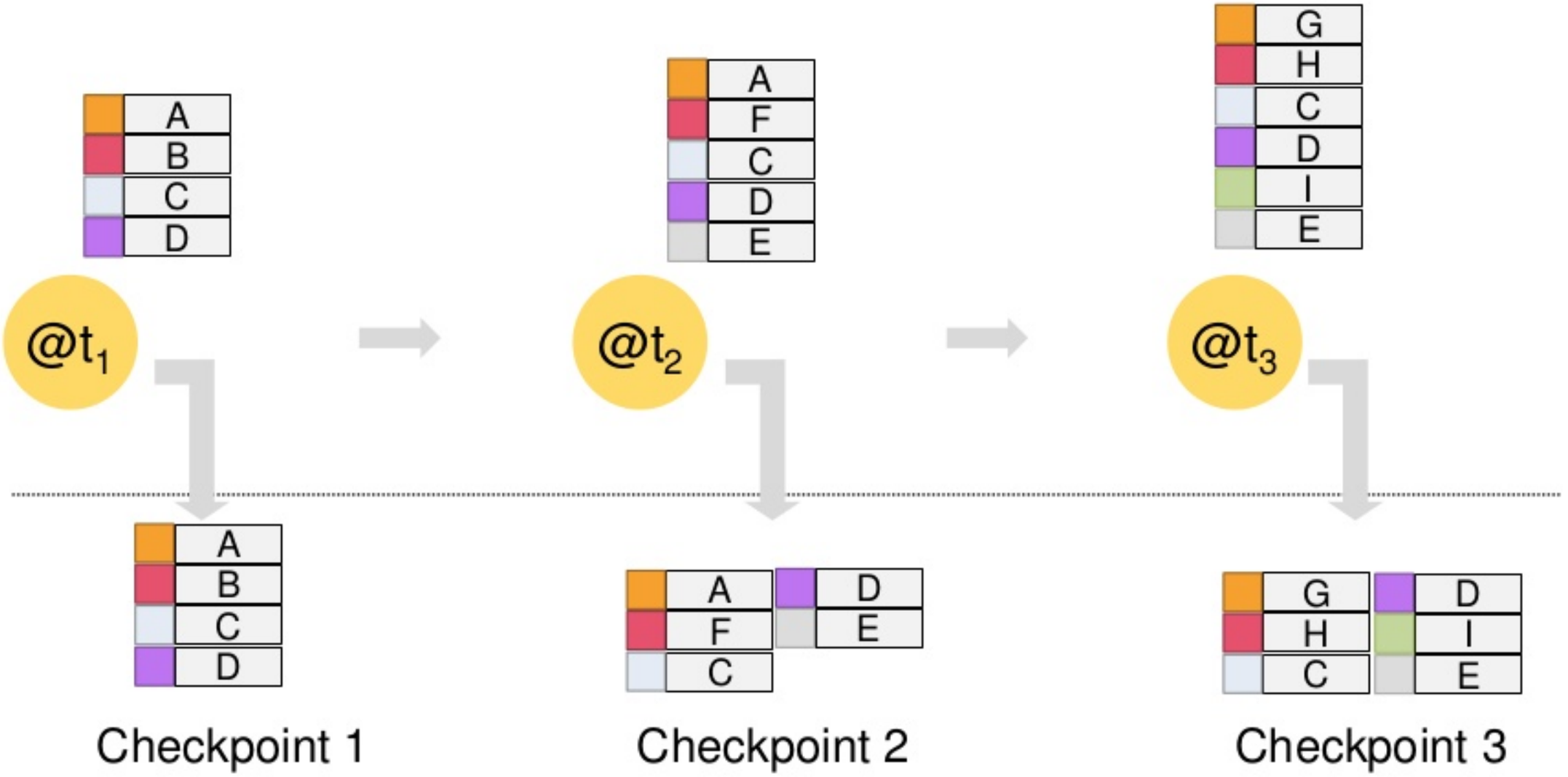
- Additional outputs for a stream
  - Late events
  - Corrupted input data
- More expressive APIs
- FLINK-4460



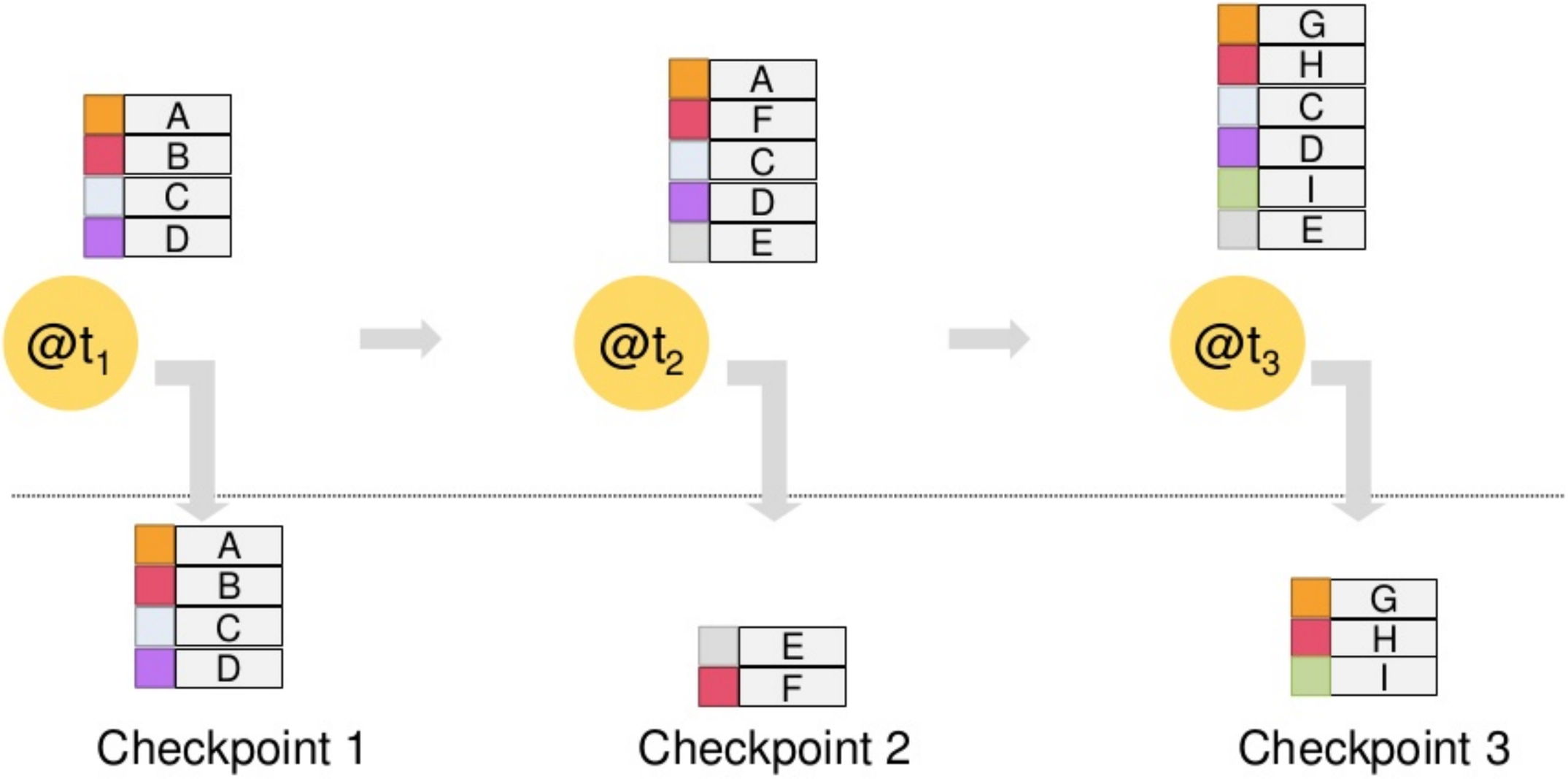
Evolution of Large State Handling



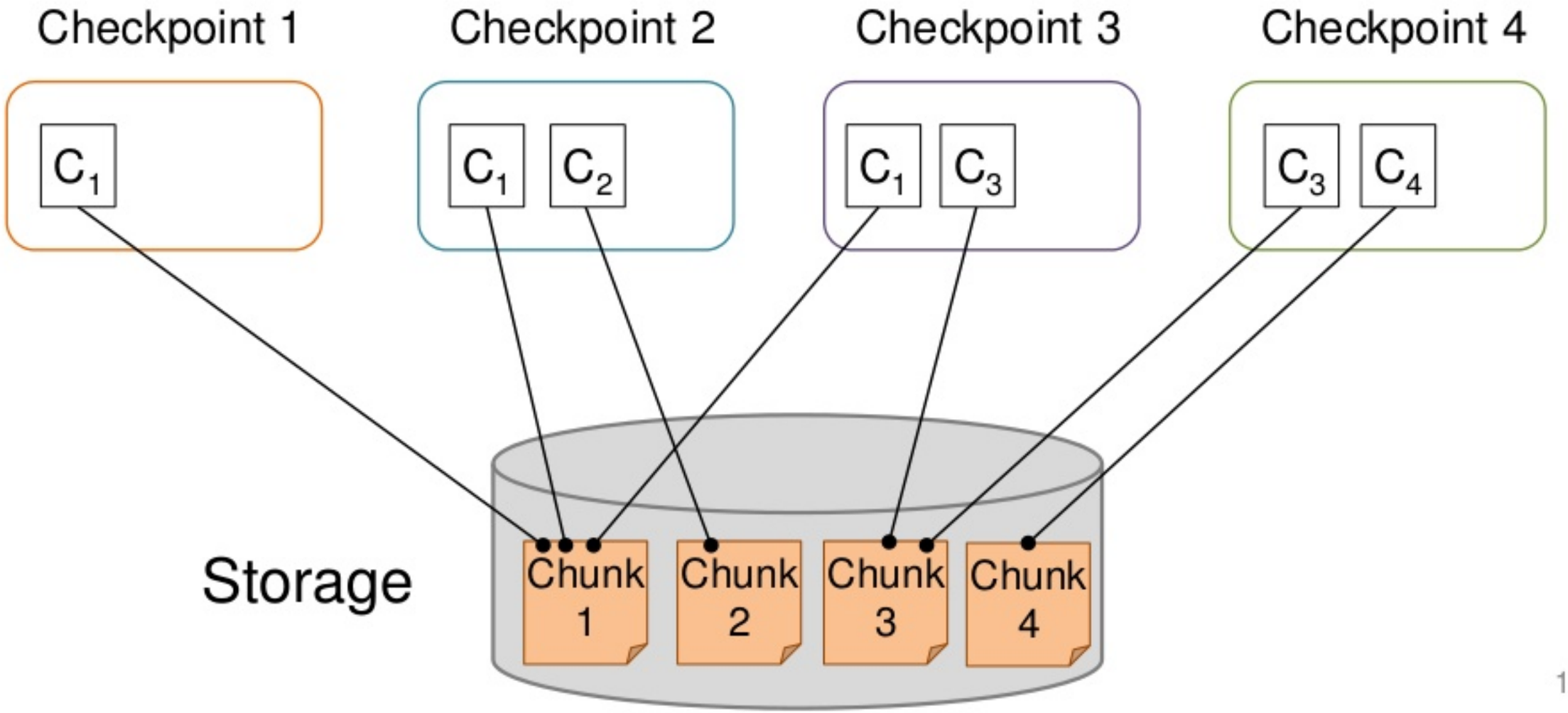
Full Checkpoints



Incremental Checkpoints



Incremental Checkpoints



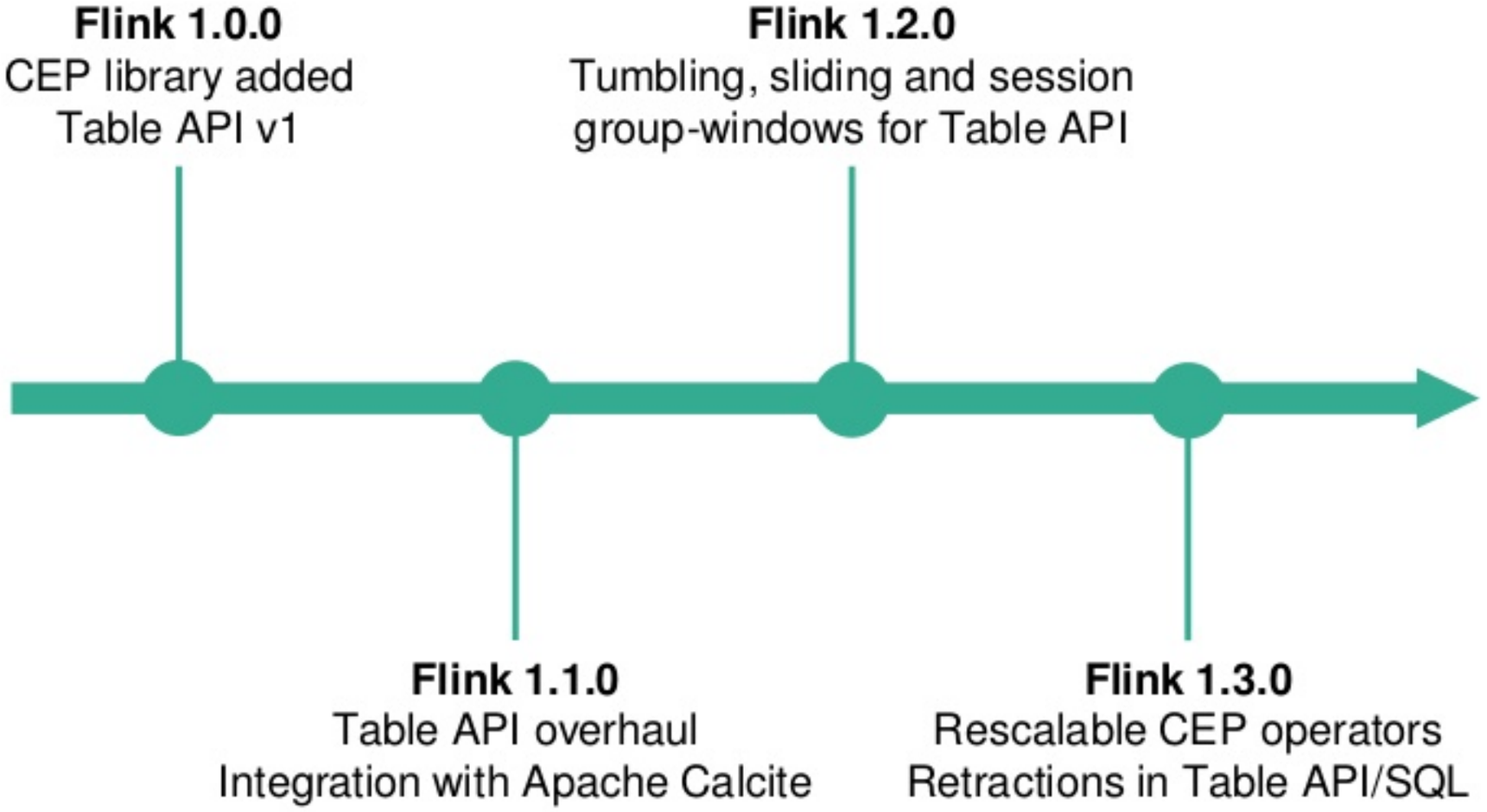
Incremental Checkpointing Contd.

- Currently supported for RocksDB state backend
- FLINK-5053
- Faster and smaller checkpoints

"*A Look at Flink's Internal Data Structures and Algorithms for Efficient Checkpointing*" by Stefan Richter, Tomorrow @ 12:20 pm Maschinenhaus

	Full checkpoint	Incremental checkpoint
Size	60 GB	1 – 30 GB
Time	180 s	3 – 30 s

Evolution of High Level APIs



Enriched CEP Language

- Support for quantifiers (+, *, ?)
  - FLINK-3318
- Iterative conditions
  - FLINK-6197
- Not operator
  - FLINK-3320

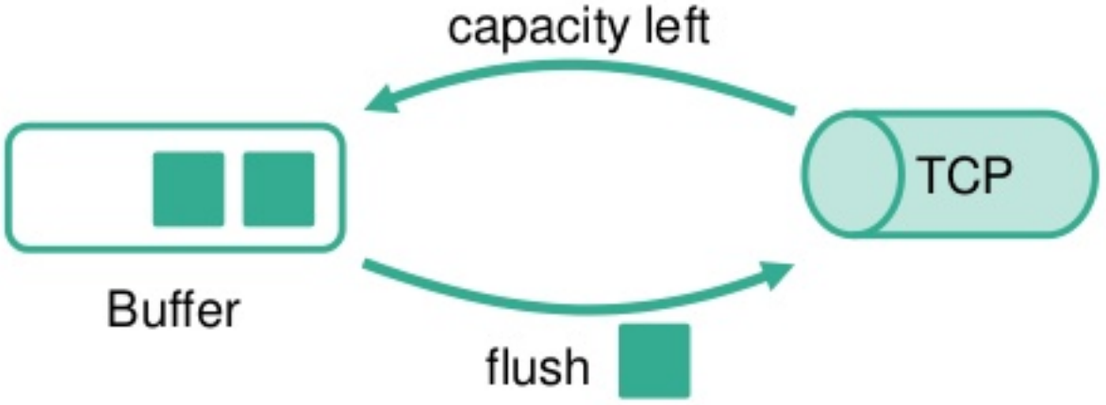
> "*Complex Event Processing With Flink: The State of FlinkCEP*" by Kostas Kloudas, Today @ 2:30 pm Maschinenhaus

Apache Flink 1.4

What’s Happening Now?

Event Driven I/O

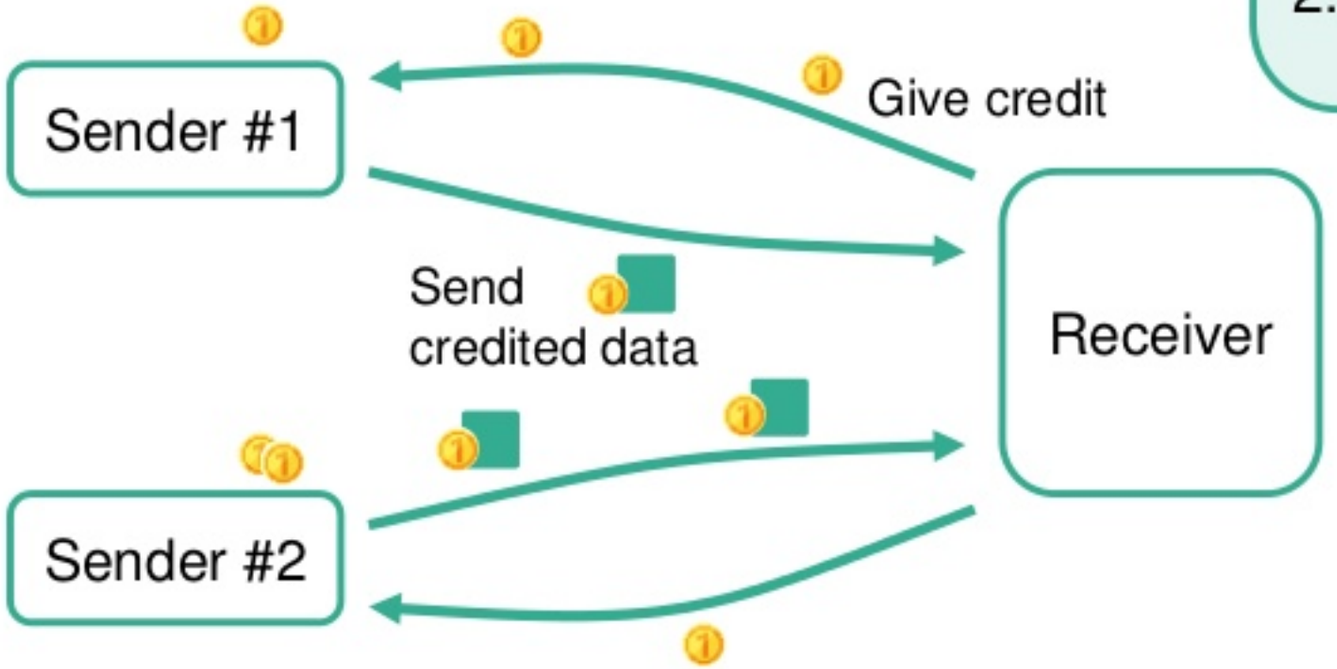
- Rework of Flink's network stack
- Event driven network I/O
  - Use full available capacity
  - Near perfect latency behaviour



Flow Control

- Flow control for TaskManager communication
  - Single channel no longer stalls other multiplexed channels
  - Fine-grained backpressure control
  - Improves checkpoint alignments

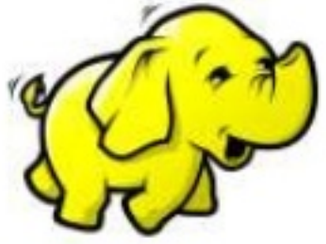
"*Building a Network Stack for Optimal Throughput / Low-Latency Trade-Offs*" by Nico Kruber, Today @ 2:00 pm Palais Atelier



New Deployment Model

- Rework of Flink's distributed architecture
- Ready for multitude of deployment scenarios
- Support for dynamic scaling

"*Flink in Containerland*" by Patrick Lucas, Tomorrow @ 3:20 pm Maschinenhaus



MESOS

Producing Exactly Once with Kafka 0.11

- Support for Kafka 0.11
- First Kafka producer with *exactly once processing* guarantees

"*Hit Me, Baby, Just One Time – Building End-to-End Exactly Once Applications With Flink*" by Piotr Nowojski, Today @ 3:20 pm Palais Atelier

Operational Robustness

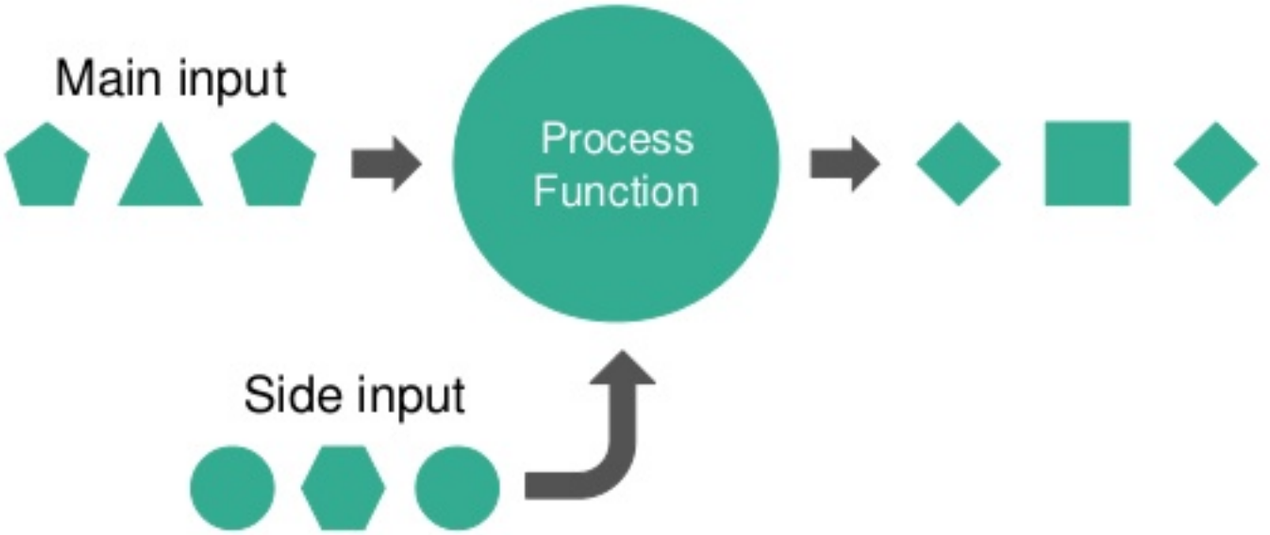
- Drop Java 7
- Support Scala 2.12
- Avoid dependency hell
  - Child first class loading
  - Relocation of dependencies
- De-Hadoopification

Apache Flink 1.5+

Next on Apache Flink

Side Inputs

- Additional input for operator
  - Join with static data set
  - Feeding of externally trained ML model
  - Window joins
- Flip-17 design document: https://goo.gl/W4yMEu



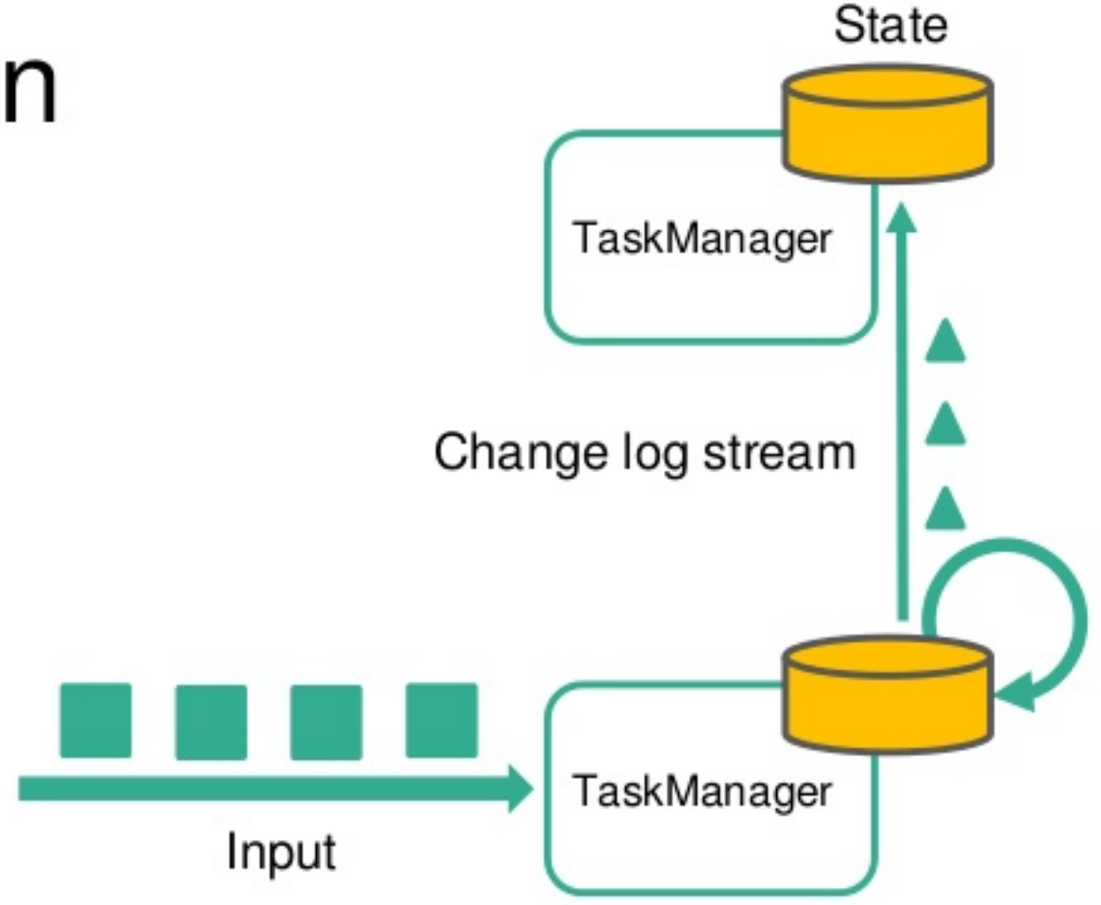
State Management & Evolution

- Eager state declaration
  - State type, serializer and name known at pre-flight time
  - Flip-22 design document: https://goo.gl/trFiSi
- Evolving existing state
  - Schema updates
  - Serializer upgrades

"Managing State in Apache Flink" by Tzu-Li Tai, Today @ 4:30 pm Kesselhaus

State Replication

- Replicate state between TaskManagers
  - Faster recovery in case of failures
  - High throughput queryable state



Programmatic Job Control

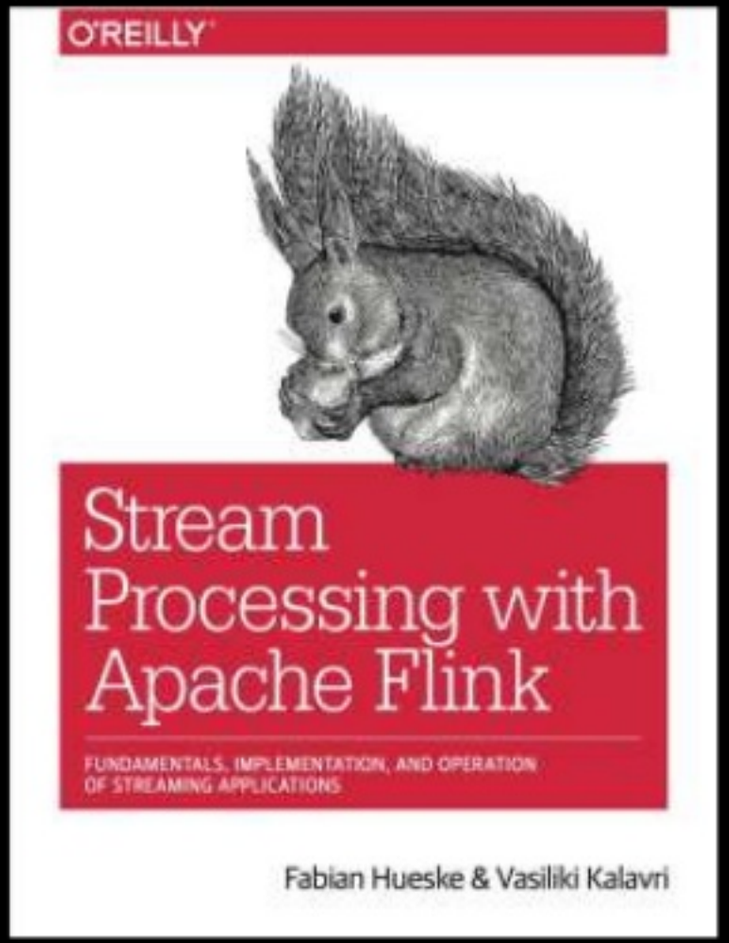
- Improve client to give better job control
- Run concurrent jobs from the same program
- Trigger savepoints programmatically
- Better testing facilities

JobClient & ClusterClient

```
StreamExecutionEnvironment env = ...;
// define program

JobClient jobClient = env.execute();

CompletableFuture<Acknowledge> savepointFuture = jobClient.takeSavepoint(savepointPath);

// wait for the savepoint completion
savepointFuture.get();

CompletableFuture<JobExecutionResult> resultFuture = jobClient.getResultFuture();

// cancel the job
jobClient.cancelJob();

// get the execution result --> should be canceled
JobExecutionResult result = resultFuture.get();

// get list of all still running jobs on the cluster
ClusterClient clusterClient = jobClient.getClusterClient();
CompletableFuture<List<JobInfo>> jobInfosFuture = clusterClient.getJobInfos();
List<JobInfo> jobInfos = jobInfosFuture.get();
```

TL;DL

- Apache Flink one of the most innovative open source stream processing platforms
- Stay tuned what's happening next ☺
- Visit the in depths talks to learn more about Flink's internals



Thank you!

@stsffap
@ApacheFlink
@dataArtisans

We are hiring!

data-artisans.com/careers