Flink and Beam:
Current State & Roadmap

Apache Flink®

Apache Beam
(incubating)

dataArtisans

Maximilian Michels
PMC Apache Flink and Apache Beam

FlinkForward 2016

Last year's talk:

Google Cloud Dataflow On Top of Apache Flink®

This year's talk:

Apache Flink® and **Apache Beam**: Current State & Roadmap

What happened?

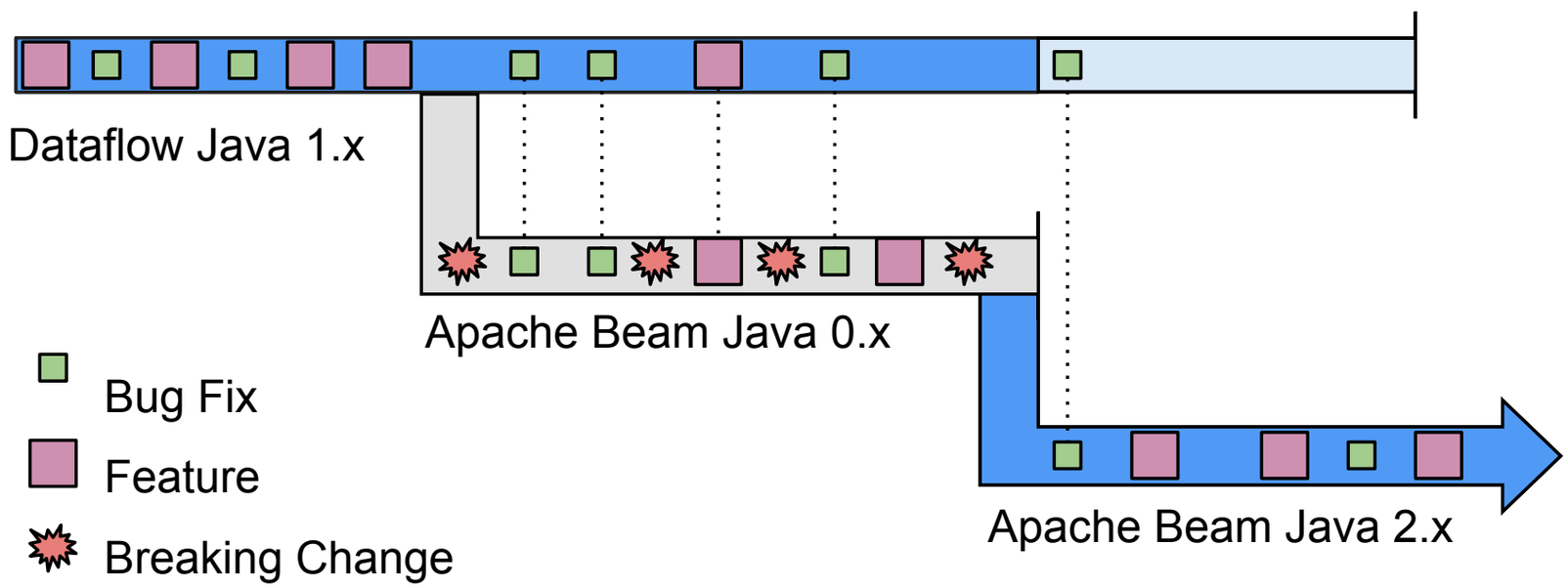
Two projects, one idea

- **Dec 2014** Google releases the Cloud Dataflow Java SDK
- **Feb 2016** Cloud Dataflow Java SDK becomes Apache Beam

Taken from the official Beam material

Apache Beam
(incubating)

- **Jan 2016** Google proposes project to the Apache incubator
- **Feb 2016** Project enters incubation
- **Jun 2016** Apache Beam 0.1.0-incubating released
- **Jul 2016** Apache Beam 0.2.0-incubating released

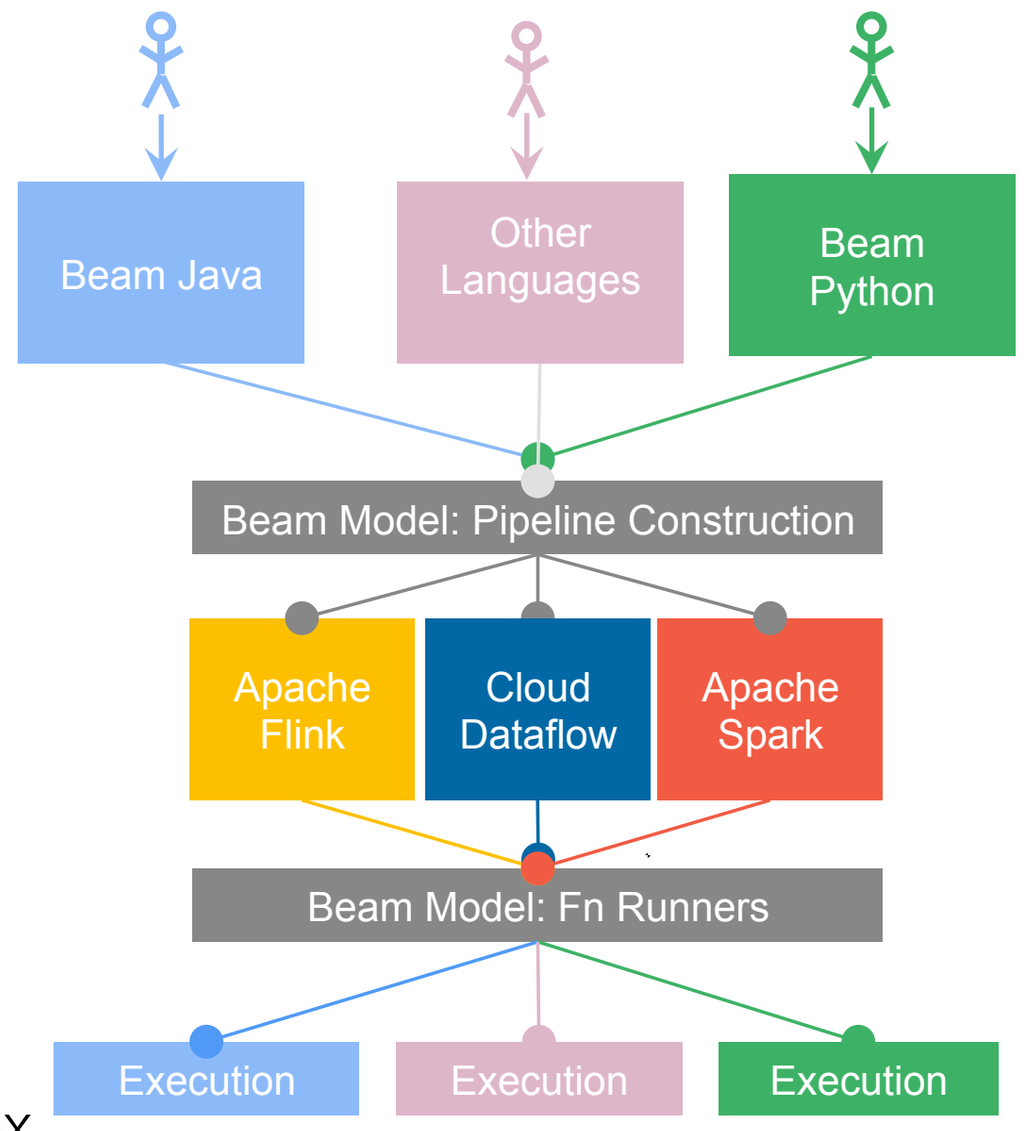


The Dataflow/Beam Model

- **2004** MapReduce
- **2010** Flume Java (Beam's API)
- **2013** MillWheel (Watermarks, exactly-once, Windows)
- **Sep 2015** Dataflow Model published at VLDB
- **Nov 2015** Apache Flink 0.10.0 DataStream API features Event Time in line with the Dataflow model

Beam Runners

- A **Runner** ports Beam code to a backend
- A Runner's concern is
  - a) translation
  - b) runtime execution
- Runners available:
  - Google Cloud Dataflow
  - Apache Flink
  - Apache Spark
- In Development: Gearpump, Apache Apex



Taken from the official Beam material

Flink Runner - Status Quo

- Stable: 0.2.0-incubating
  - Powered by Flink 1.0.3
- Development: 0.3.0-incubating-SNAPSHOT
  - Powered by Flink 1.1.2
- Changes coming in regularly, use the snapshot version to get the latest features

Evolution of the Flink Runner

dataArtisans

2014

Dec

- Google Cloud Dataflow SDK released

2015

Jan

- Initial commit and drafting

Mar

- Batch support without Windows

Dec

- Streaming support with Watermarks and Windows

THE APACHE SOFTWARE FOUNDATION

2016

Mar

- Contribution to Apache Beam

May

- Parallel and checkpointed sources in streaming
- Batch support with Windows and Side Inputs
- RunnableOnService for batch

Aug

- Side Inputs for streaming
- RunnableOnService for streaming

Batched vs Streamed Execution

- Batched Execution
  - Built on top of the DataSet API
  - Supports only bounded sources
  - Managed Memory
- Streamed Execution
  - Built on top of the DataStream API
  - Supports bounded and unbounded sources
  - Watermark support which leverages full Dataflow Model
  - Checkpointing
- When to pick which? If undecided, just let the Runner decide

The Flink Runner's Strength

- Backed by a runtime which is streaming and batch aware
- Embeds and reuses Beam logic whenever possible
  - Window and Triggering
  - Source API
  - State Internals / Checkpointing
- Whenever possible, translate primitive transforms
- Integration with Beam's RunnableOnService tests (batch/ streaming)

Capability Matrix

- Compares Runners with the reference model
  - What results are being calculated?
  - Where in event time?
  - When in processing time?
  - How do refinements of results relate?

For more information see
http://beam.incubator.apache.org/learn/runners/capability-matrix/

What

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark
ParDo	✓	✓	✓	✓
GroupByKey	✓	✓	✓	~
Flatten	✓	✓	✓	✓
Combine	✓	✓	✓	✓
Composite Transforms	✓	~	~	~
Side Inputs	✓	✓	✓	~
Source API	✓	✓	✓	✓
Aggregators	~	~	~	~
Keyed State	× (BEAM-25)	×	×	×

Where

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark
Global windows	✓	✓	✓	✓
Fixed windows	✓	✓	✓	~
Sliding windows	✓	✓	✓	~
Session windows	✓	✓	✓	×
Custom windows	✓	✓	✓	×
Custom merging windows	✓	✓	✓	×
Timestamp control	✓	✓	✓	×

When

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark
Configurable triggering	✓	✓	✓	×
Event-time triggers	✓	✓	✓	×
Processing-time triggers	✓	✓	✓	✓
Count triggers	✓	✓	✓	×
[Meta]data driven triggers	× (BEAM-101)	×	×	×
Composite triggers	✓	✓	✓	×
Allowed lateness	✓	✓	✓	×
Timers	× (BEAM-27)	×	×	×

How

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark
Discarding	✓	✓	✓	✓
Accumulating	✓	✓	✓	×
Accumulating & Retracting	× (BEAM-91)	×	×	×

So we’re good?

- Flink Runner currently the most advanced Runner which is backed by an open-source engine

- Does that mean the Flink Runner is perfect?

Roadmap Flink Runner

- Side Input streaming: size restrictions
- Integrate with new PipelineResult
- Dynamic Scaling
- Incremental Checkpointing
- Connectors (Kafka 8/9, Cassandra, Elasticsearch, RabbitMQ, Redis, NiFi, Kinesis)
- Libraries (Gelly, CEP, Storm Compatibility, ML)
- Performance testing
- Bug fixes

Apache Flink Features powering the Runner

- Streaming
  - Checkpointing / Savepoints
  - State backends
- Batch
  - Pipelined execution
  - Managed Memory
- High Availability
- Flink’s native sources / sinks

Demo

- Develop a Beam program with the Flink Runner
- Monitor applications using the web interface
- Handle failures by restoring from checkpointed state
- Create a Savepoint, stop the Beam program, resume execution from it

Thank you for your attention!