Abstract

The world of Big Data involves an ever increasing field of players, from storage systems to processing engines and distributed programming models. Much as SQL stands as a lingua franca for declarative data analysis, Apache Beam aims to provide a standard for expressing both batch and streaming data processing pipelines in a variety of languages across a variety of platforms and engines.

In this talk, we will show how Beam gives users the flexibility to choose the best environment for their needs and read data from any storage system; allows any Big Data API to execute in multiple environments; allows any processing engines to support multiple domain-specific user communities; and allows any storage system to read/write process data at massive scale. In a way, Apache Beam is a glue that connects the Big Data ecosystem together; it enables “anything to run anywhere”.

Apache Beam: Integrating the Big Data Ecosystem Up, Down, and Sideways

Davor Bonaci
PMC Chair, Apache Beam
Software Engineer, Google

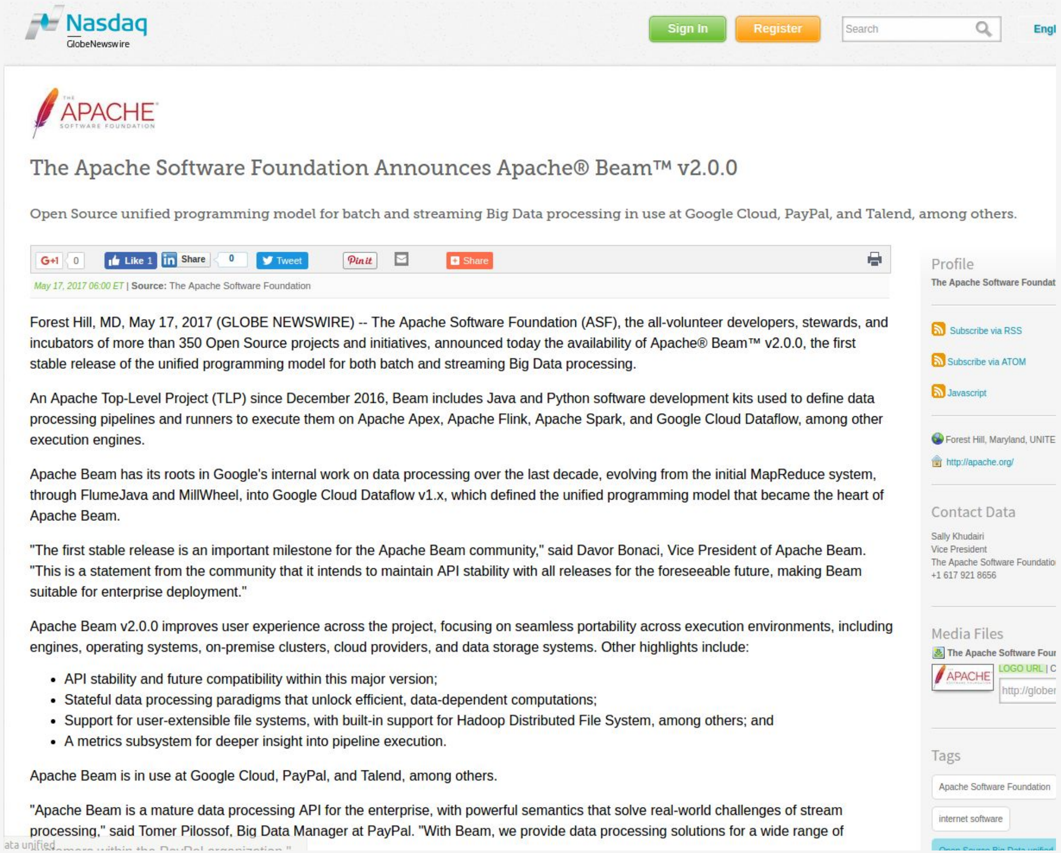
Jean-Baptiste Onofré
PMC Member, Apache Beam
Software Architect, Talend

Apache Beam: Open Source data processing APIs

- Expresses data-parallel batch and streaming algorithms using one unified API
- Cleanly separates data processing logic from runtime requirements
- Supports execution on multiple distributed processing runtime environments

Apache Beam is
a ***unified*** programming model
designed to provide
efficient and ***portable***
data processing pipelines

Announcing the first stable release



Nasdaq GlobeNewswire Sign In Register Search Engl

The Apache Software Foundation Announces Apache® Beam™ v2.0.0

Open Source unified programming model for batch and streaming Big Data processing in use at Google Cloud, PayPal, and Talend, among others.

May 17, 2017 06:00 ET | **Source:** The Apache Software Foundation

Forest Hill, MD, May 17, 2017 (GLOBE NEWSWIRE) -- The Apache Software Foundation (ASF), the all-volunteer developers, stewards, and incubators of more than 350 Open Source projects and initiatives, announced today the availability of Apache® Beam™ v2.0.0, the first stable release of the unified programming model for both batch and streaming Big Data processing.

An Apache Top-Level Project (TLP) since December 2016, Beam includes Java and Python software development kits used to define data processing pipelines and runners to execute them on Apache Apex, Apache Flink, Apache Spark, and Google Cloud Dataflow, among other execution engines.

Apache Beam has its roots in Google's internal work on data processing over the last decade, evolving from the initial MapReduce system, through FlumeJava and MillWheel, into Google Cloud Dataflow v1.x, which defined the unified programming model that became the heart of Apache Beam.

"The first stable release is an important milestone for the Apache Beam community," said Davor Bonaci, Vice President of Apache Beam. "This is a statement from the community that it intends to maintain API stability with all releases for the foreseeable future, making Beam suitable for enterprise deployment."

Apache Beam v2.0.0 improves user experience across the project, focusing on seamless portability across execution environments, including engines, operating systems, on-premise clusters, cloud providers, and data storage systems. Other highlights include:

- API stability and future compatibility within this major version;
- Stateful data processing paradigms that unlock efficient, data-dependent computations;
- Support for user-extensible file systems, with built-in support for Hadoop Distributed File System, among others; and
- A metrics subsystem for deeper insight into pipeline execution.

Apache Beam is in use at Google Cloud, PayPal, and Talend, among others.

"Apache Beam is a mature data processing API for the enterprise, with powerful semantics that solve real-world challenges of stream processing," said Tomer Pilossof, Big Data Manager at PayPal. "With Beam, we provide data processing solutions for a wide range of

Profile
The Apache Software Foundat

Subscribe via RSS
Subscribe via ATOM
Javascript

Forest Hill, Maryland, UNITE
http://apache.org/

Contact Data
Sally Khudairi
Vice President
The Apache Software Foundatio
+1 617 921 8656

Media Files
The Apache Software Four
LOGO URL | C
http://glober

Tags
Apache Software Foundation
internet software

Apache Beam at this conference

- *Using Apache Beam for Batch, Streaming, and Everything in Between*
    - Dan Halperin @ 10:15 am
- *Apache Beam: Integrating the Big Data Ecosystem Up, Down, and Sideways*
    - Davor Bonaci, and Jean-Baptiste Onofré @ 11:15 am
- *Concrete Big Data Use Cases Implemented with Apache Beam*
    - Jean-Baptiste Onofré @ 12:15 pm
- *Nexmark, a Unified Framework to Evaluate Big Data Processing Systems*
    - Ismaël Mejía, and Etienne Chauchot @ 2:30 pm

Apache Beam at this conference

- *Apache Beam Birds of a Feather*
  - Wednesday, 6:30 pm - 7:30 pm

- *Apache Beam Hacking Time*
  - Time: all-day Thursday
  - 2nd floor, collaboration area
  - (depending on interest)

APACHE:
BIG_DATA
NORTH_AMERICA

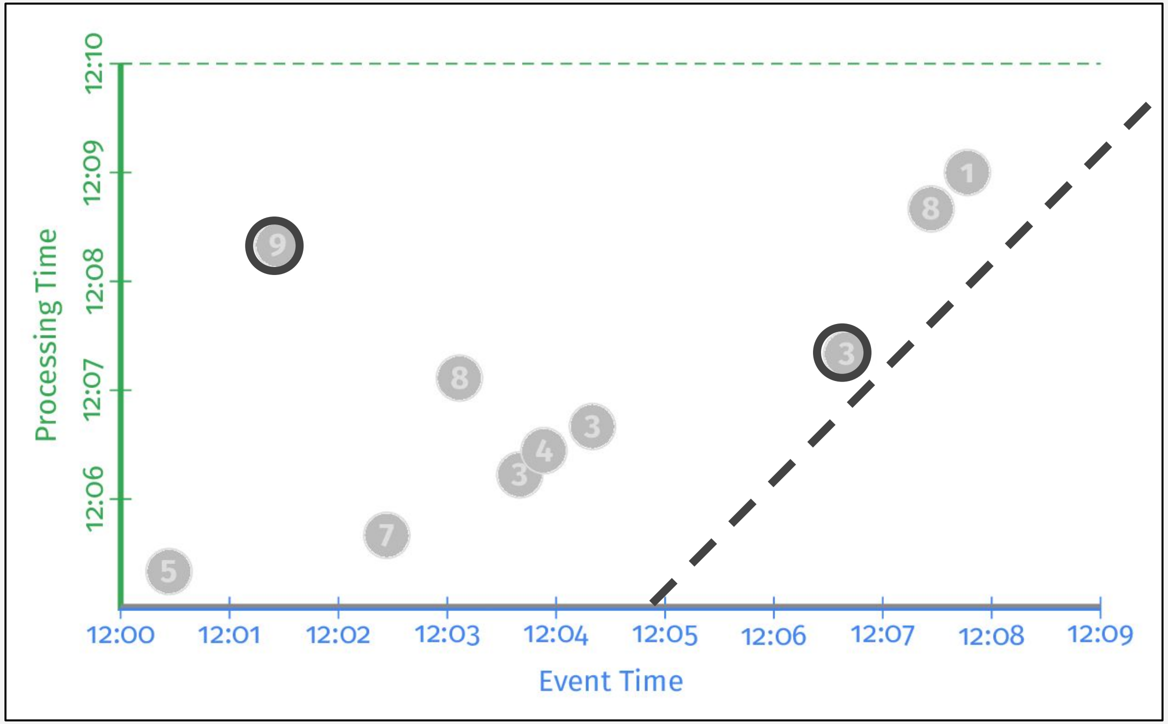
Agenda

1. Expressing data-parallel pipelines with the Beam model
2. The Beam vision for ***portability***
3. Parallel and portable pipelines in practice
4. ***Extensibility*** to integrate the entire Big Data ecosystem

Expressing data-parallel pipelines with the Beam model

A unified model for batch and stream data processing

Processing time vs. event time



The Beam Model: asking the right questions

What results are calculated?

Where in event time are results calculated?

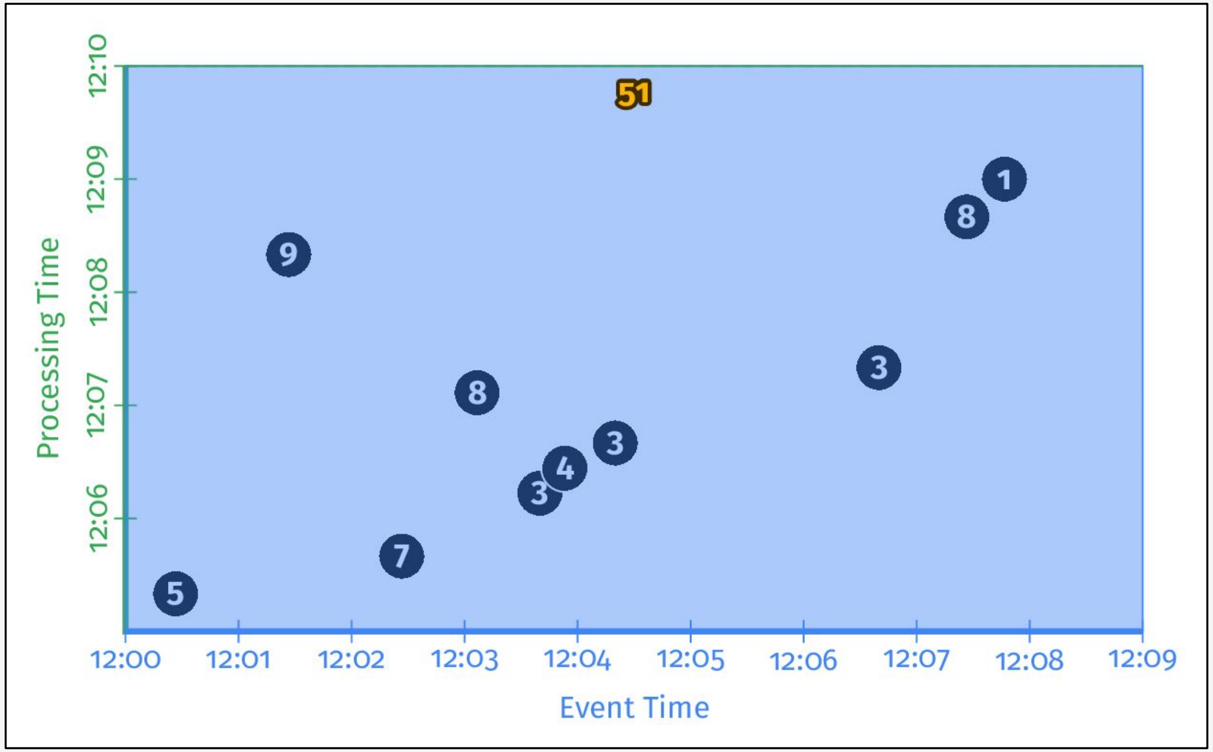
When in processing time are results materialized?

How do refinements of results relate?

The Beam Model: **What** is being computed?

```
PCollection<KV<String, Integer>> scores = input

.apply(Sum.integersPerKey());
```

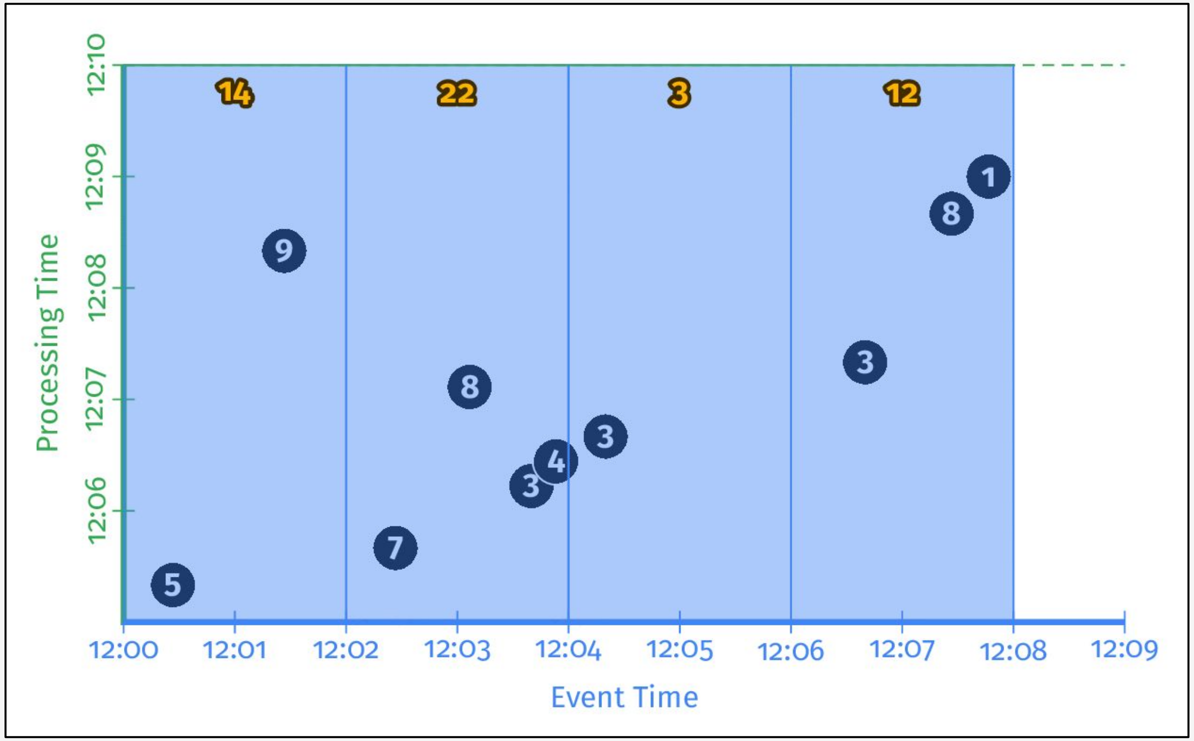
The Beam Model: **What** is being computed?



The Beam Model: **Where** in event time?

```
PCollection<KV<String, Integer>> scores = input

.apply(Window.into(FixedWindows.of(Duration.standardMinutes(2)))

.apply(Sum.integersPerKey());
```

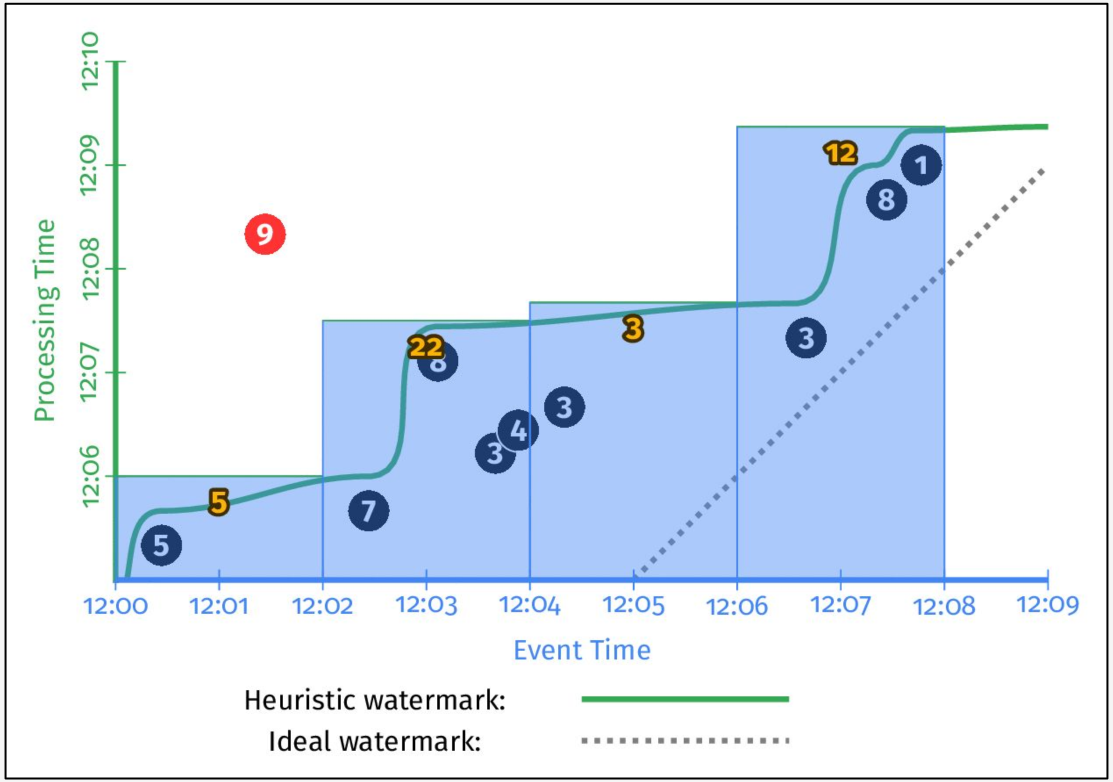
The Beam Model: **Where** in event time?



The Beam Model: **When** in processing time?

```
PCollection<KV<String, Integer>> scores = input
.apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
               .triggering(AtWatermark()))
.apply(Sum.integersPerKey());
```

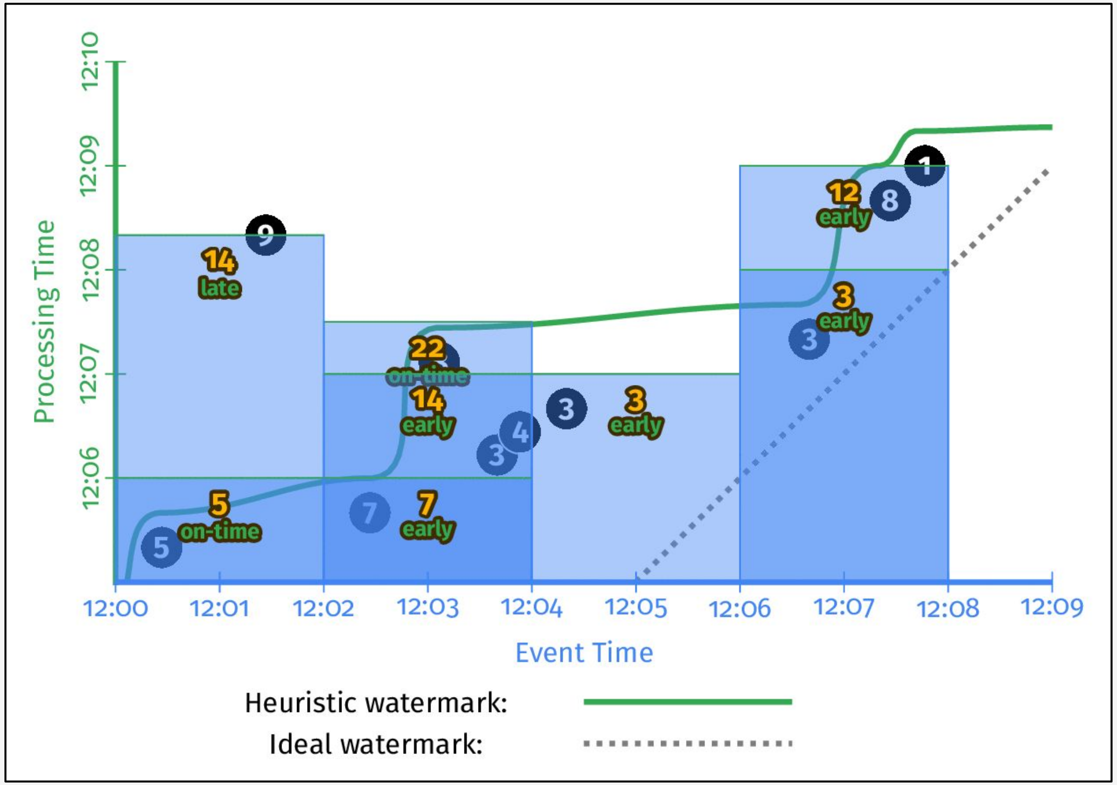
The Beam Model: **When** in processing time?



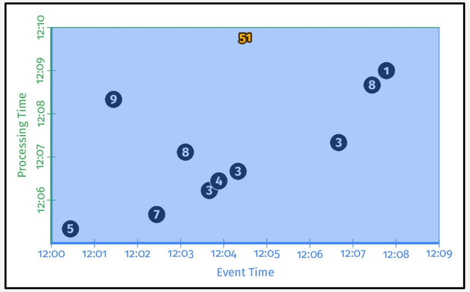
The Beam Model: **How** do refinements relate?

```
PCollection<KV<String, Integer>> scores = input

.apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
             .triggering(AtWatermark()

                 .withEarlyFirings(
                     AtPeriod(Duration.standardMinutes(1)))
                 .withLateFirings(AtCount(1)))
             .accumulatingFiredPanes())

    .apply(Sum.integersPerKey());
```

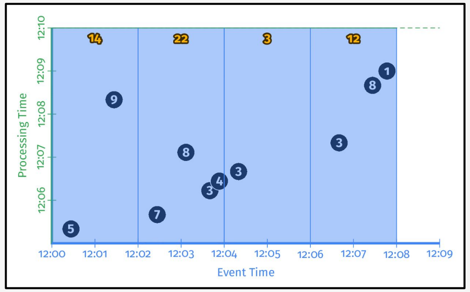
The Beam Model: **How** do refinements relate?



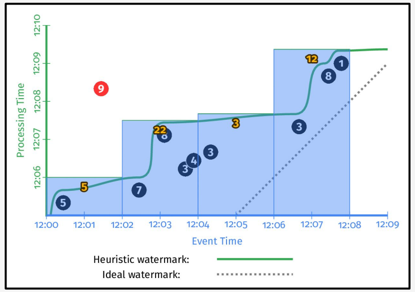
Customizing What Where When How



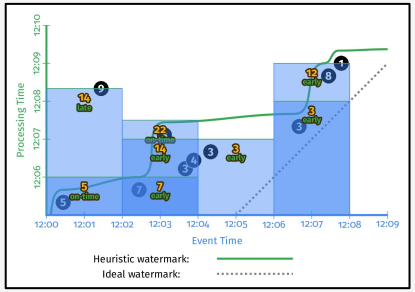
1
Classic Batch



2
Windowed Batch



3
Streaming

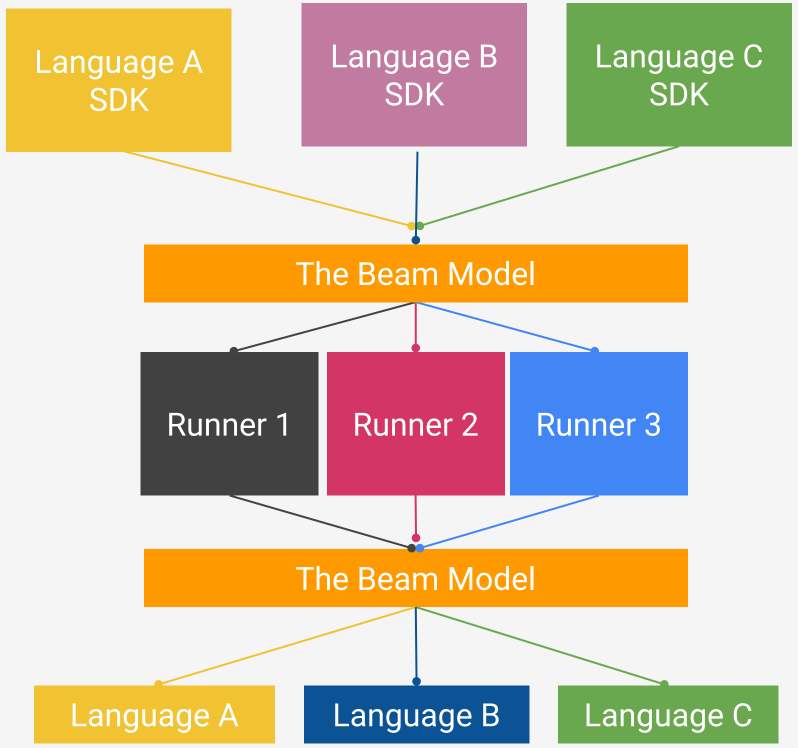


4
Streaming + Accumulation

The Beam vision for portability

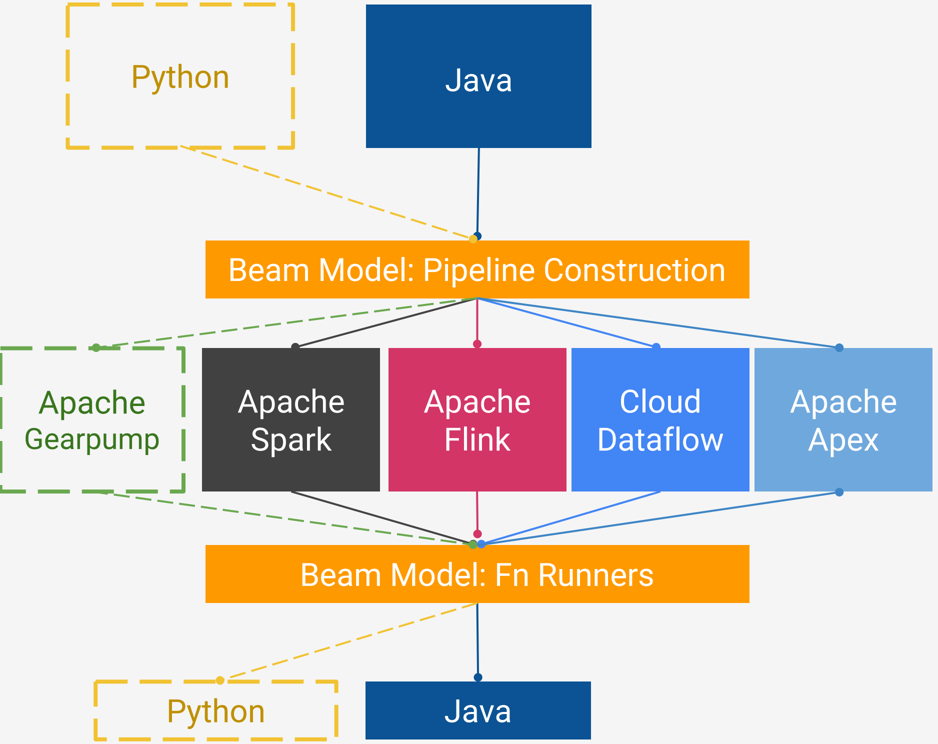
“Write once,
run anywhere”

Beam Vision: mix and match SDKs and runtimes



- **The Beam Model:** the abstractions at the core of Apache Beam
- **Choice of SDK:** Users write their pipelines in a language that's familiar and integrated with their other tooling
- **Choice of Runners:** Users choose the right runtime for their current needs -- on-prem / cloud, open source / not, fully managed / not
- **Scalability for Developers:** Clean APIs allow developers to contribute modules independently

Beam Vision: as of May 2017



- Beam's Java SDK runs on multiple runtime environments, including:
    - Apache Apex
    - Apache Spark
    - Apache Flink
    - Google Cloud Dataflow
    - *[in development]* Apache Gearpump

- Cross-language infrastructure is in progress.
    - Beam's Python SDK currently runs on Direct runner & Google Cloud Dataflow

Example Beam Runners

Apache Spark

- Open-source cluster-computing framework
- Large ecosystem of APIs and tools
- Runs on premise or in the cloud

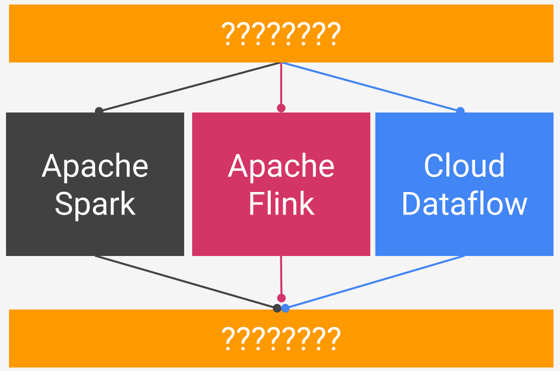
Apache Flink

- Open-source distributed data processing engine
- High-throughput and low-latency stream processing
- Runs on premise or in the cloud

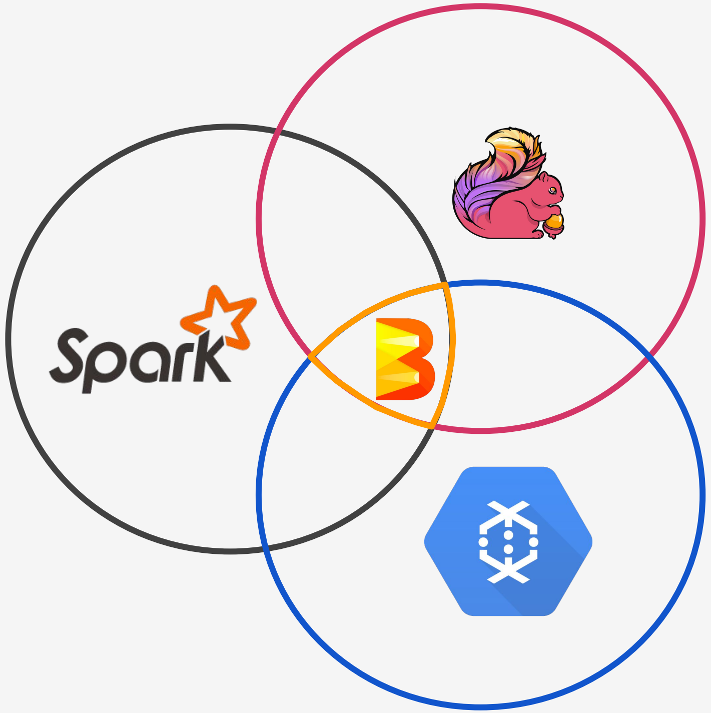
Google Cloud Dataflow

- Fully-managed service for batch and stream data processing
- Provides dynamic auto-scaling, monitoring tools, and tight integration with Google Cloud Platform

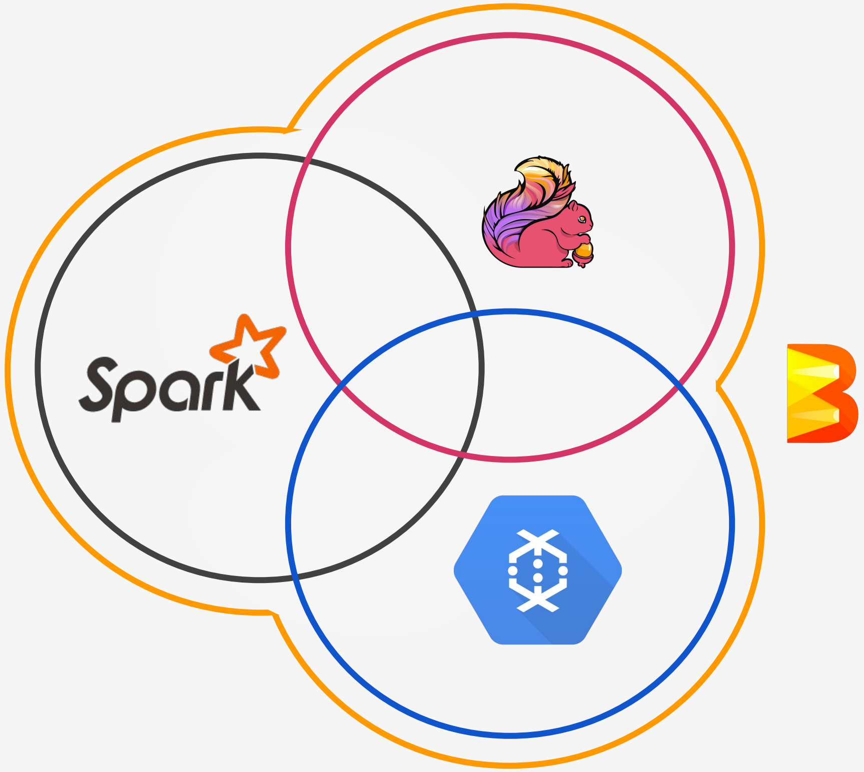
How do you build an abstraction layer?



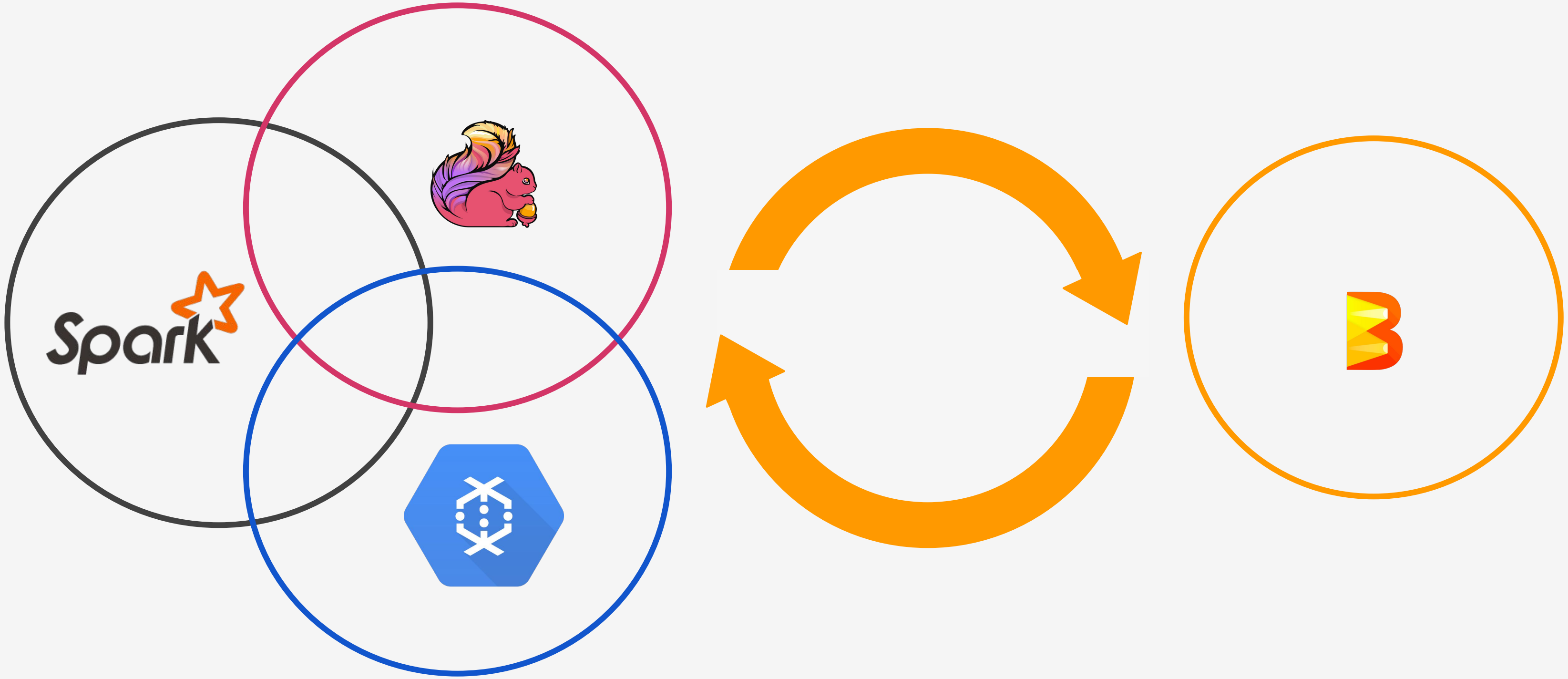
Beam: the intersection of runner functionality?



Beam: the union of runner functionality?



Beam: the future!



Categorizing Runner Capabilities

	Beam Model	Dataflow	Flink	Spark	Apex
ParDo	✓	✓	✓	✓	✓
GroupByKey	✓	✓	✓	~	✓
Flatten	✓	✓	✓	✓	✓
Combine	✓	✓	✓	✓	✓
Composite Transforms	✓	~	~	~	~
Side Inputs	✓	✓	✓	✓	✓
Source API	✓	✓	✓	✓	✓
Aggregators	~	~	~	~	✕
Keyed State	✕	✕	✕	✕	✕

	Beam Model	Dataflow	Flink	Spark	Apex
Configurable triggering	✓	✓	✓	✕	✓
Event-time triggers	✓	✓	✓	✕	✓
Processing-time triggers	✓	✓	✓	✓	✓
Count triggers	✓	✓	✓	✕	✓
[Meta]data driven triggers	✕	✕	✕	✕	✕
Composite triggers	✓	✓	✓	✕	✓
Allowed lateness	✓	✓	✓	✕	✓
Timers	✕	✕	✕	✕	✕

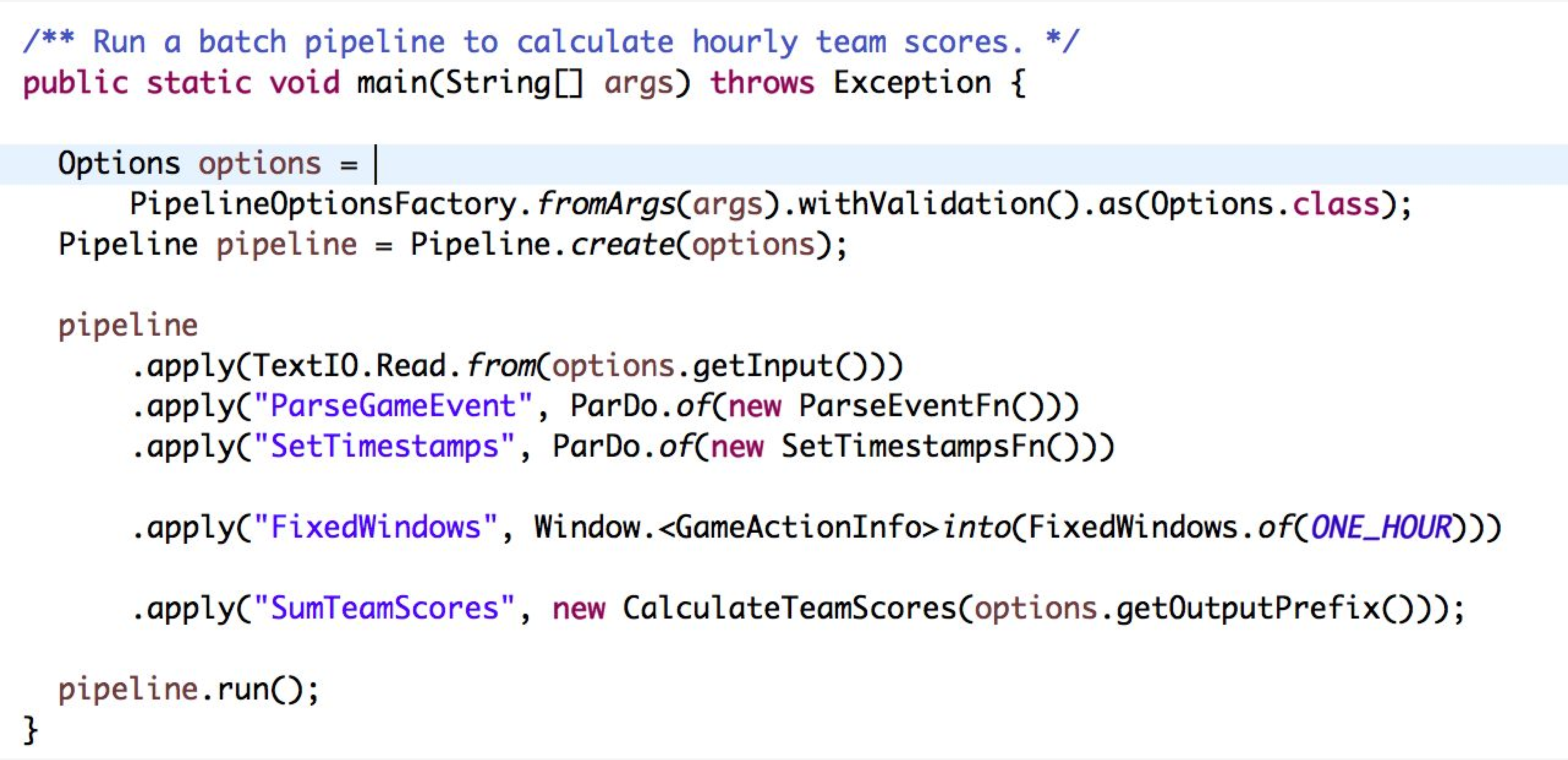
	Beam Model	Dataflow	Flink	Spark	Apex
Global windows	✓	✓	✓	✓	✓
Fixed windows	✓	✓	✓	✓	✓
Sliding windows	✓	✓	✓	✓	✓
Session windows	✓	✓	✓	✓	✓
Custom windows	✓	✓	✓	✓	✓
Custom merging windows	✓	✓	✓	✓	✓
Timestamp control	✓	✓	✓	✓	✓

	Beam Model	Dataflow	Flink	Spark	Apex
Discarding	✓	✓	✓	✓	✓
Accumulating	✓	✓	✓	✕	✓
Accumulating & Retracting	✕	✕	✕	✕	✕

**https://beam.apache.org/
documentation/runners/capability-matrix/**

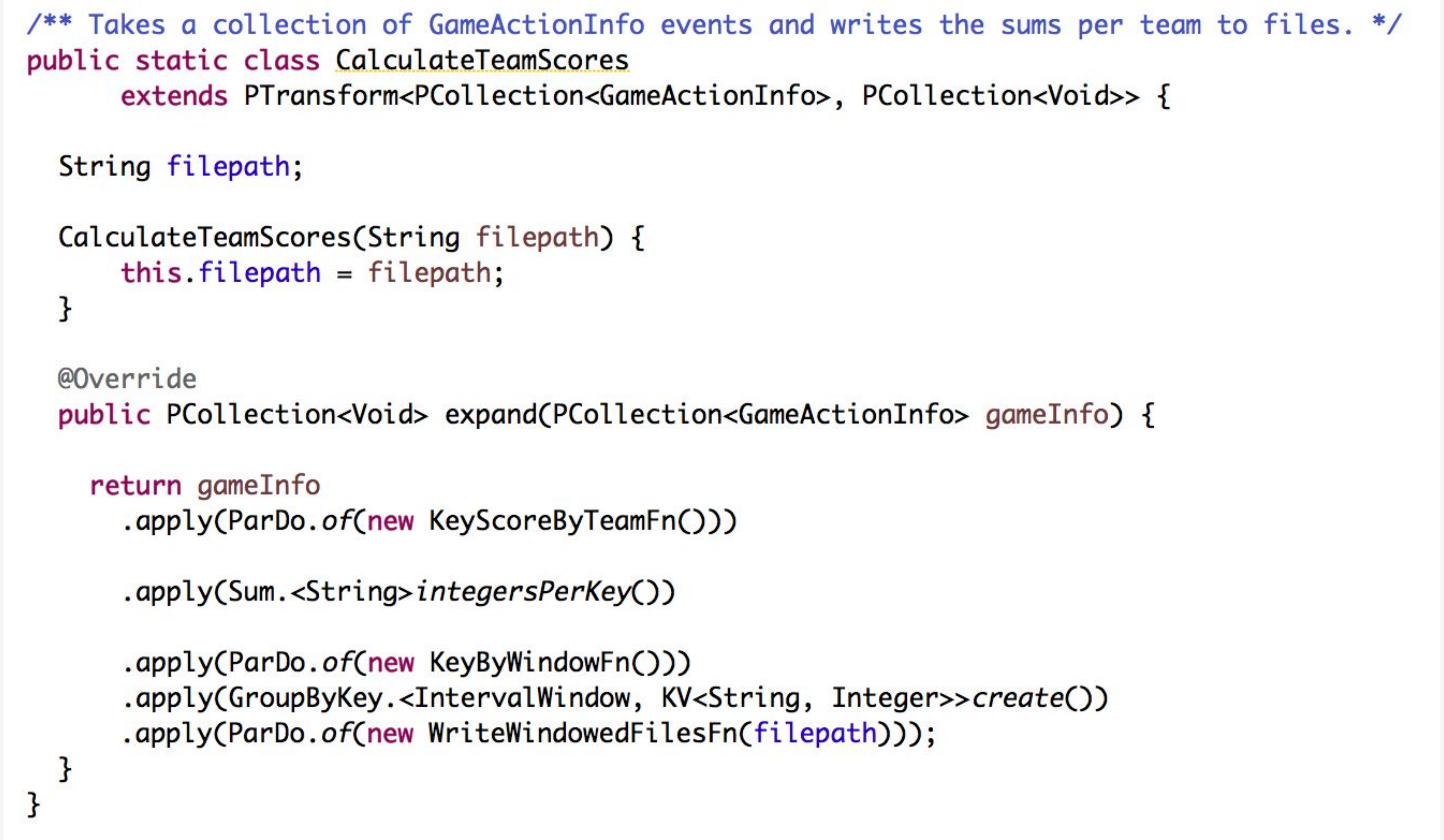
Parallel and portable pipelines in practice

A Use Case

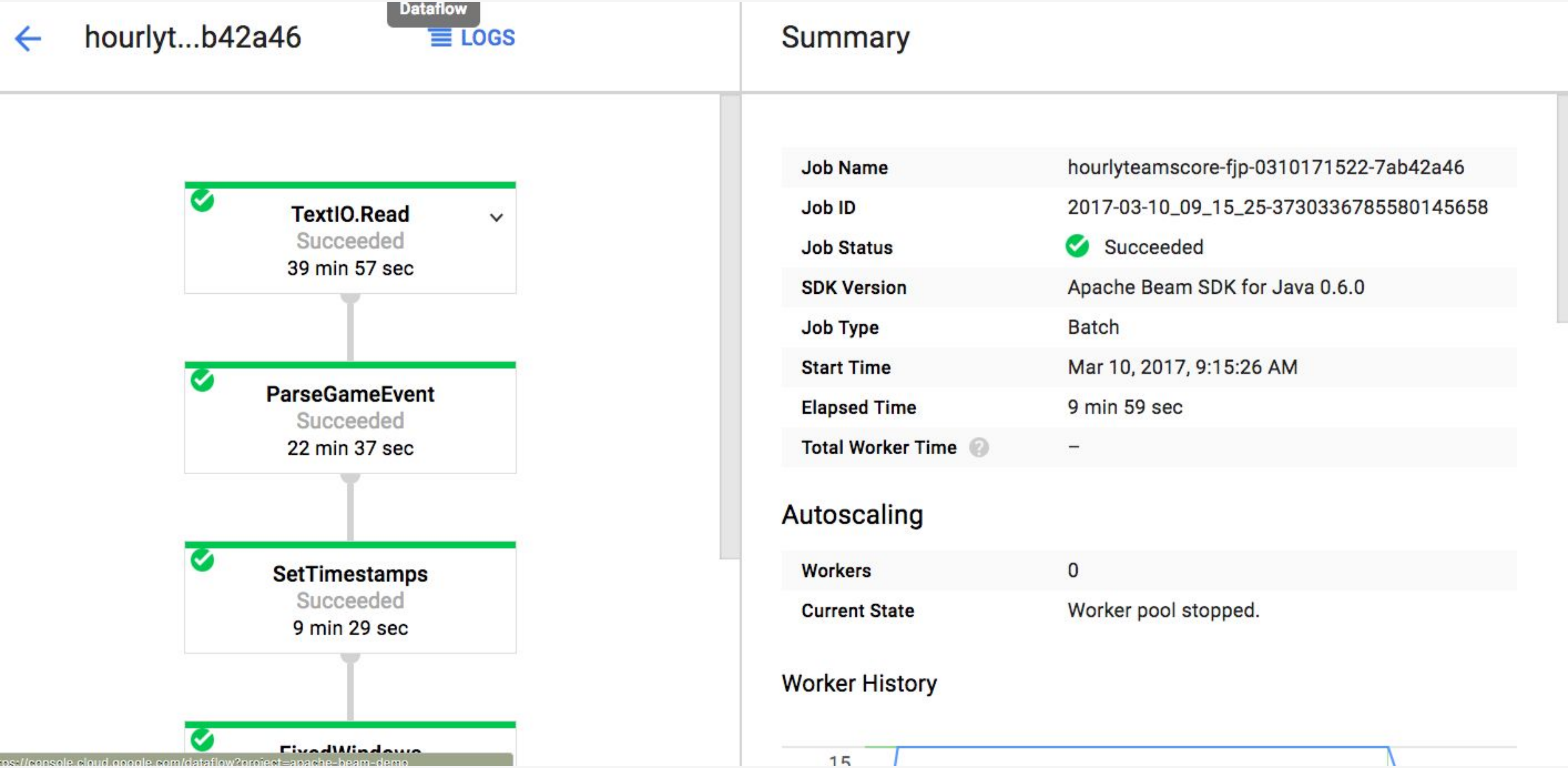


```java
/** Run a batch pipeline to calculate hourly team scores. */
public static void main(String[] args) throws Exception {

  Options options = 
      PipelineOptionsFactory.fromArgs(args).withValidation().as(Options.class);
  Pipeline pipeline = Pipeline.create(options);

  pipeline
      .apply(TextIO.Read.from(options.getInput()))
      .apply("ParseGameEvent", ParDo.of(new ParseEventFn()))
      .apply("SetTimestamps", ParDo.of(new SetTimestampsFn()))

      .apply("FixedWindows", Window.<GameActionInfo>into(FixedWindows.of(ONE_HOUR)))

      .apply("SumTeamScores", new CalculateTeamScores(options.getOutputPrefix()));

  pipeline.run();
}
```

```java
/** DoFn to parse raw log lines into structured GameActionInfos. */
static class ParseEventFn extends DoFn<String, GameActionInfo> {

  private static final Logger LOG = LoggerFactory.getLogger(ParseEventFn.class);
  private static final Counter numParseErrorsCounter = Metrics.counter(ParseEventFn.class,

  @ProcessElement
  public void processElement(ProcessContext c) {
    String[] components = c.element().split(",");
    try {
      String user = components[0].trim();
      String team = components[1].trim();
      Integer score = Integer.parseInt(components[2].trim());
      Long timestamp = Long.parseLong(components[3].trim());
      GameActionInfo gInfo = new GameActionInfo(user, team, score, timestamp);
      c.output(gInfo);
    } catch (ArrayIndexOutOfBoundsException | NumberFormatException e) {
      numParseErrorsCounter.inc();
      LOG.info("Parse error on " + c.element() + ", " + e.getMessage());
    }
  }
}
```



```
/** Takes a collection of GameActionInfo events and writes the sums per team to files. */
public static class CalculateTeamScores
      extends PTransform<PCollection<GameActionInfo>, PCollection<Void>> {

  String filepath;

  CalculateTeamScores(String filepath) {
      this.filepath = filepath;
  }

  @Override
  public PCollection<Void> expand(PCollection<GameActionInfo> gameInfo) {

    return gameInfo
      .apply(ParDo.of(new KeyScoreByTeamFn()))

      .apply(Sum.<String>integersPerKey())

      .apply(ParDo.of(new KeyByWindowFn()))
      .apply(GroupByKey.<IntervalWindow, KV<String, Integer>>create())
      .apply(ParDo.of(new WriteWindowedFilesFn(filepath)));
  }
}
```



Dataflow

hourlyt...b42a46

LOGS

TextIO.Read
Succeeded
39 min 57 sec

ParseGameEvent
Succeeded
22 min 37 sec

SetTimestamps
Succeeded
9 min 29 sec

FixedWindows

Summary

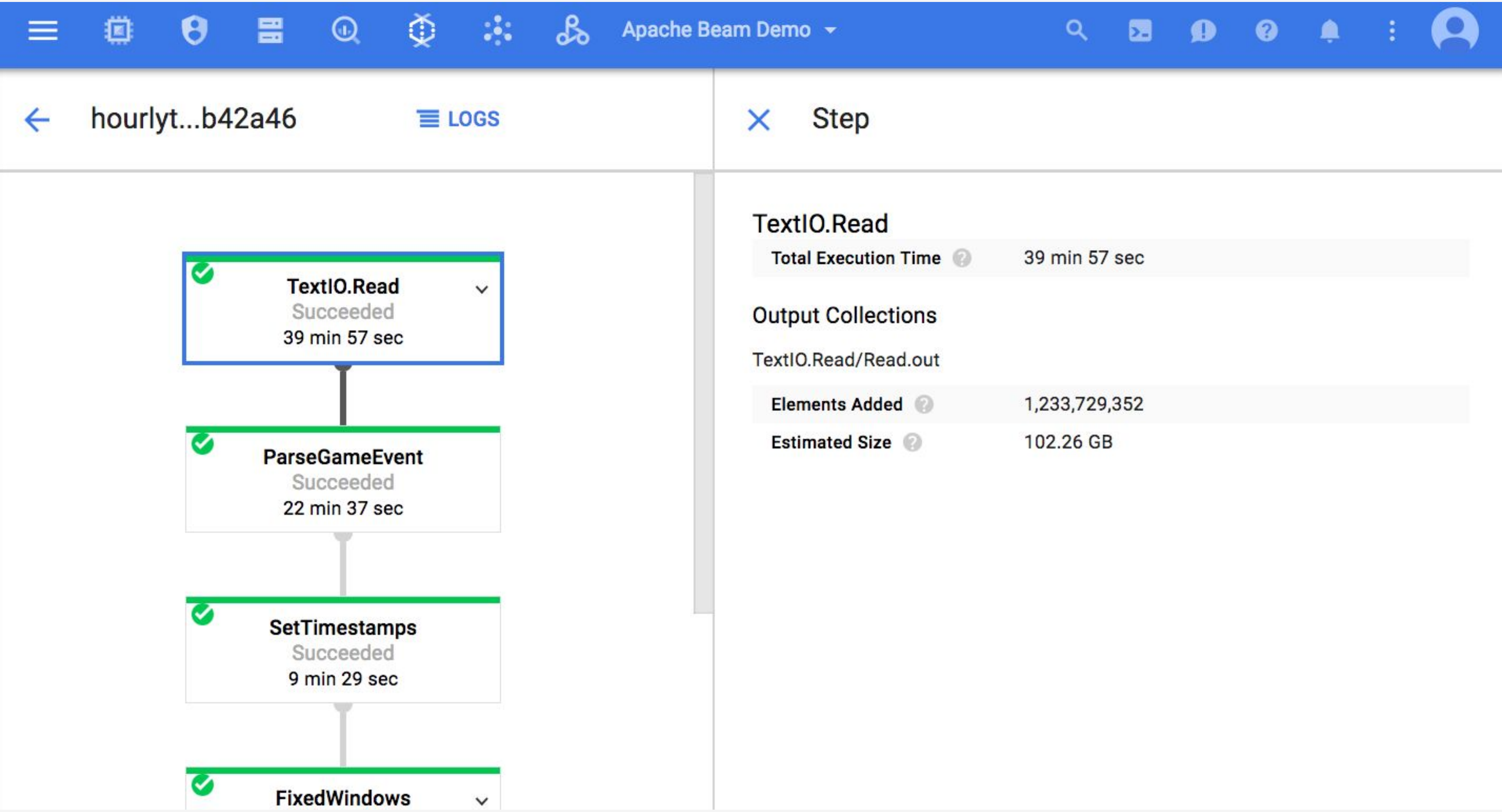
Job Name	hourlyteamscore-fjp-0310171522-7ab42a46
Job ID	2017-03-10_09_15_25-3730336785580145658
Job Status	✓ Succeeded
SDK Version	Apache Beam SDK for Java 0.6.0
Job Type	Batch
Start Time	Mar 10, 2017, 9:15:26 AM
Elapsed Time	9 min 59 sec
Total Worker Time	–

Autoscaling

Workers	0
Current State	Worker pool stopped.

Worker History

15



TextIO.Read
Succeeded
39 min 57 sec

ParseGameEvent
Succeeded
22 min 37 sec

SetTimestamps
Succeeded
9 min 29 sec

FixedWindows

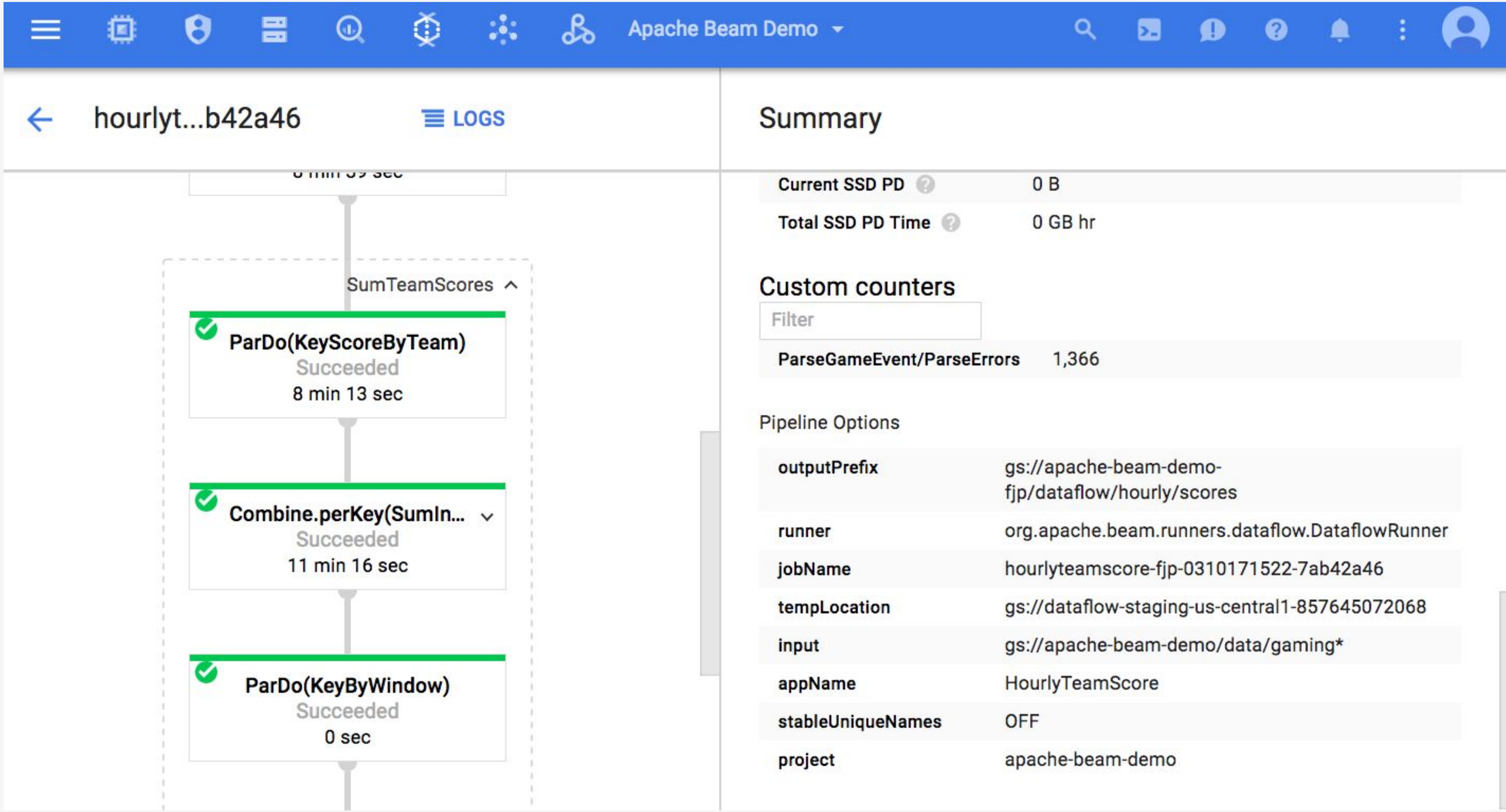
TextIO.Read

Total Execution Time	39 min 57 sec

Output Collections

TextIO.Read/Read.out

Elements Added	1,233,729,352
Estimated Size	102.26 GB



Summary

Current SSD PD	0 B
Total SSD PD Time	0 GB hr

Custom counters

ParseGameEvent/ParseErrors	1,366

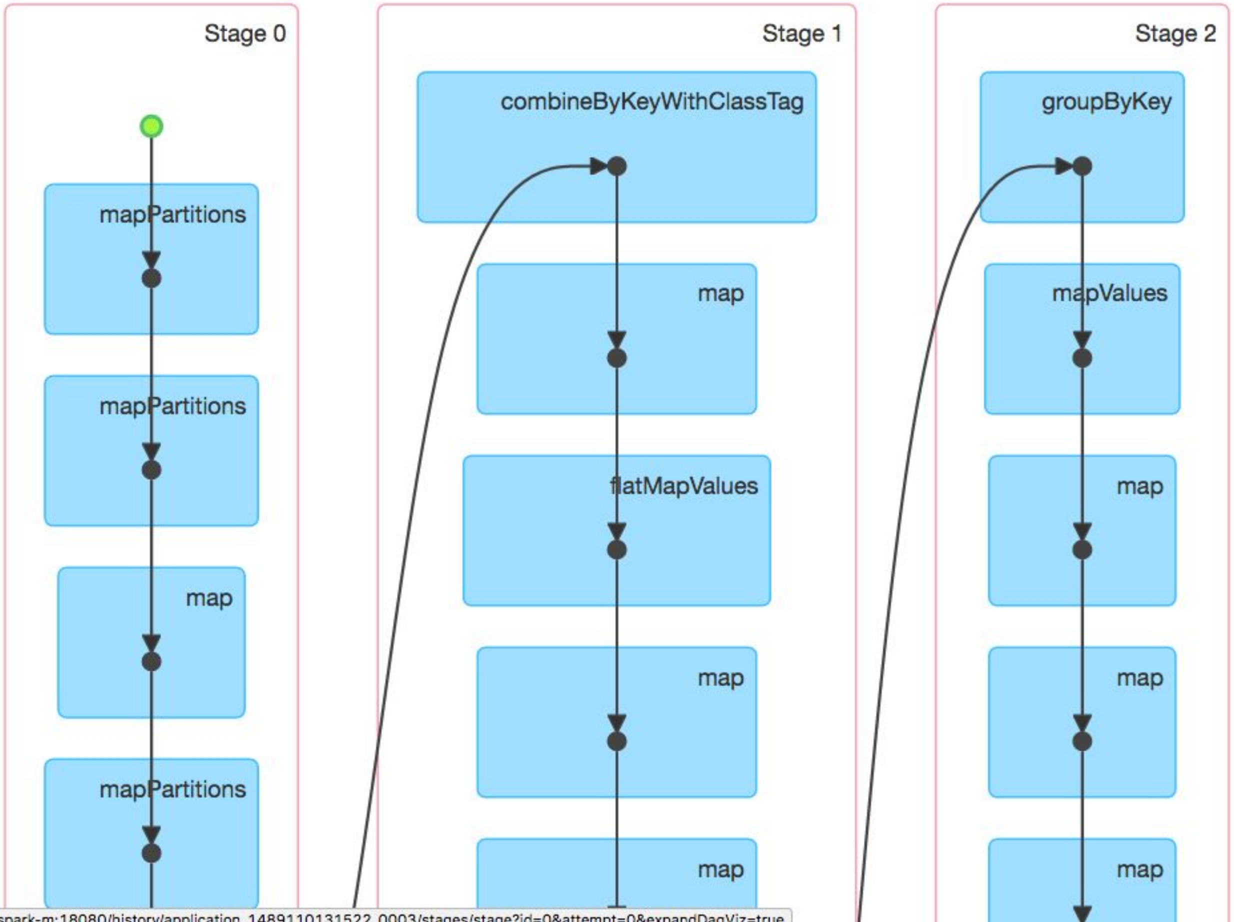
Pipeline Options

outputPrefix	gs://apache-beam-demo-fjp/dataflow/hourly/scores
runner	org.apache.beam.runners.dataflow.DataflowRunner
jobName	hourlyteamscore-fjp-0310171522-7ab42a46
tempLocation	gs://dataflow-staging-us-central1-857645072068
input	gs://apache-beam-demo/data/gaming*
appName	HourlyTeamScore
stableUniqueNames	OFF
project	apache-beam-demo

Completed Stages: 3

▸ Event Timeline

▾ DAG Visualization



gaming-spark-m:18080/history/application_1489110131522_0003/stages/stage?id=0&attempt=0&expandDagViz=true

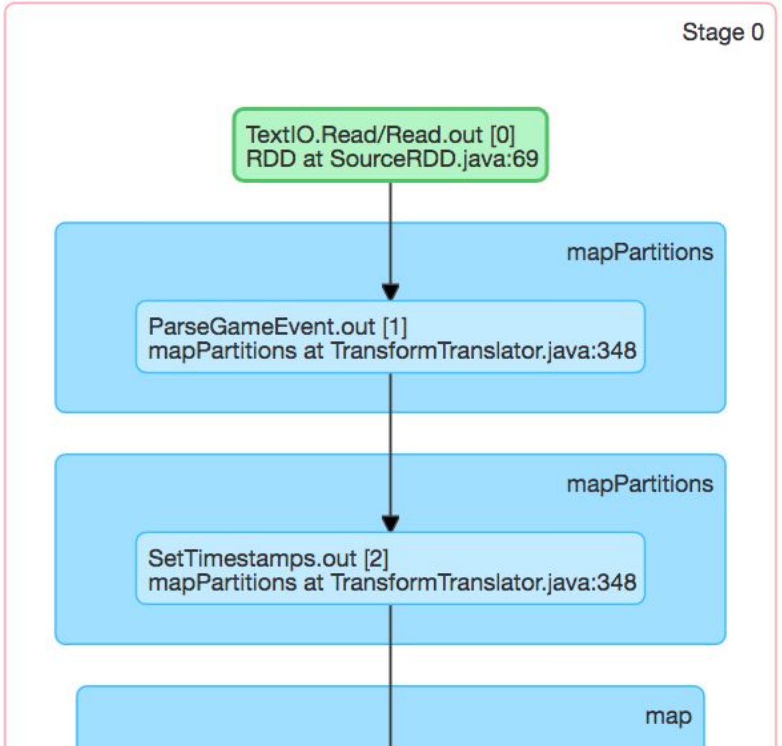
Details for Stage 0 (Attempt 0)

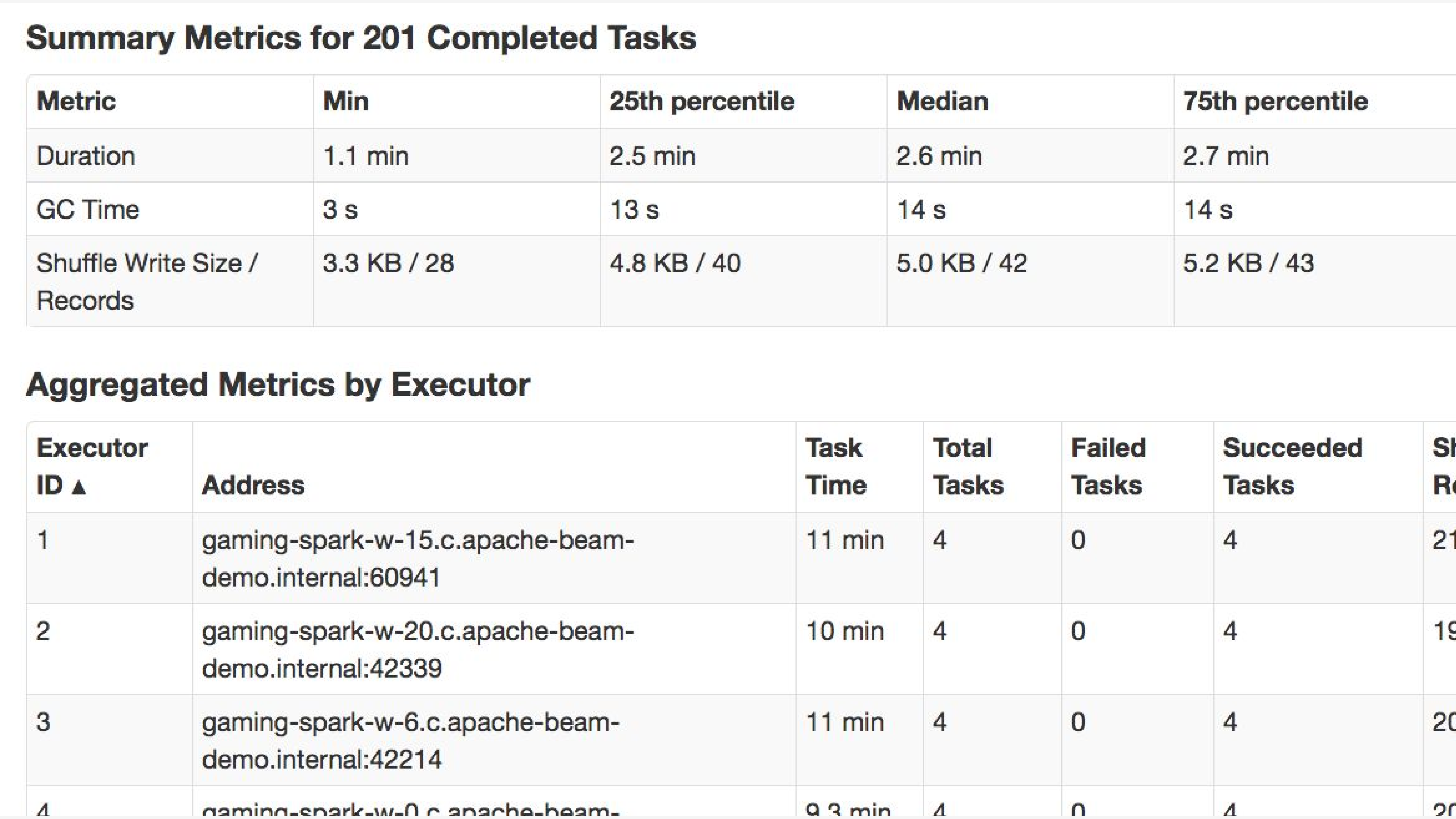
Total Time Across All Tasks: 8.6 h

Locality Level Summary: Process local: 201

Shuffle Write: 997.1 KB / 8374

▾ DAG Visualization



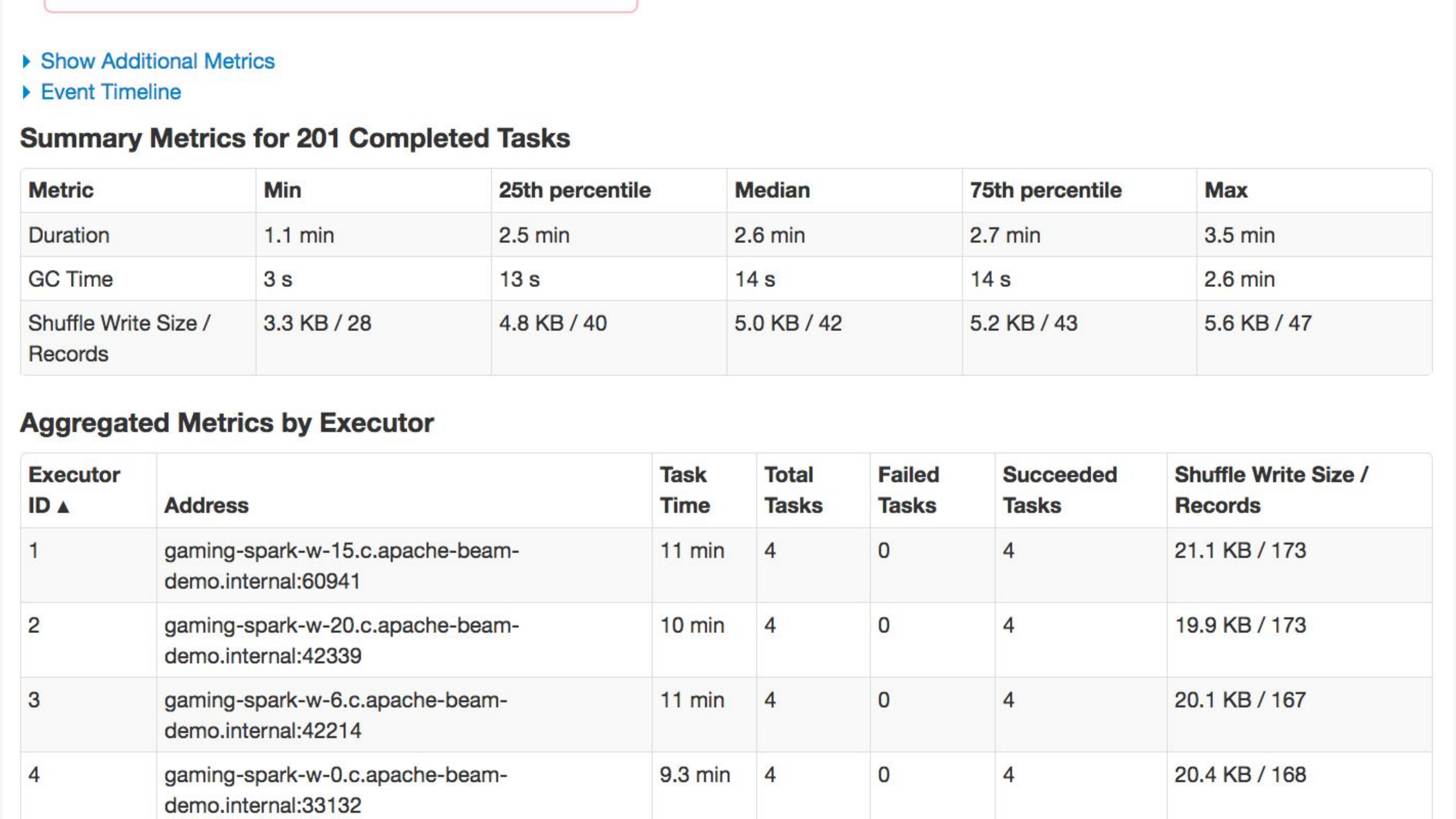


Summary Metrics for 201 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile
Duration	1.1 min	2.5 min	2.6 min	2.7 min
GC Time	3 s	13 s	14 s	14 s
Shuffle Write Size / Records	3.3 KB / 28	4.8 KB / 40	5.0 KB / 42	5.2 KB / 43

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Sh Re
1	gaming-spark-w-15.c.apache-beam-demo.internal:60941	11 min	4	0	4	21
2	gaming-spark-w-20.c.apache-beam-demo.internal:42339	10 min	4	0	4	19
3	gaming-spark-w-6.c.apache-beam-demo.internal:42214	11 min	4	0	4	20
4	gaming-spark-w-0.c.apache-beam-	9.3 min	4	0	4	20



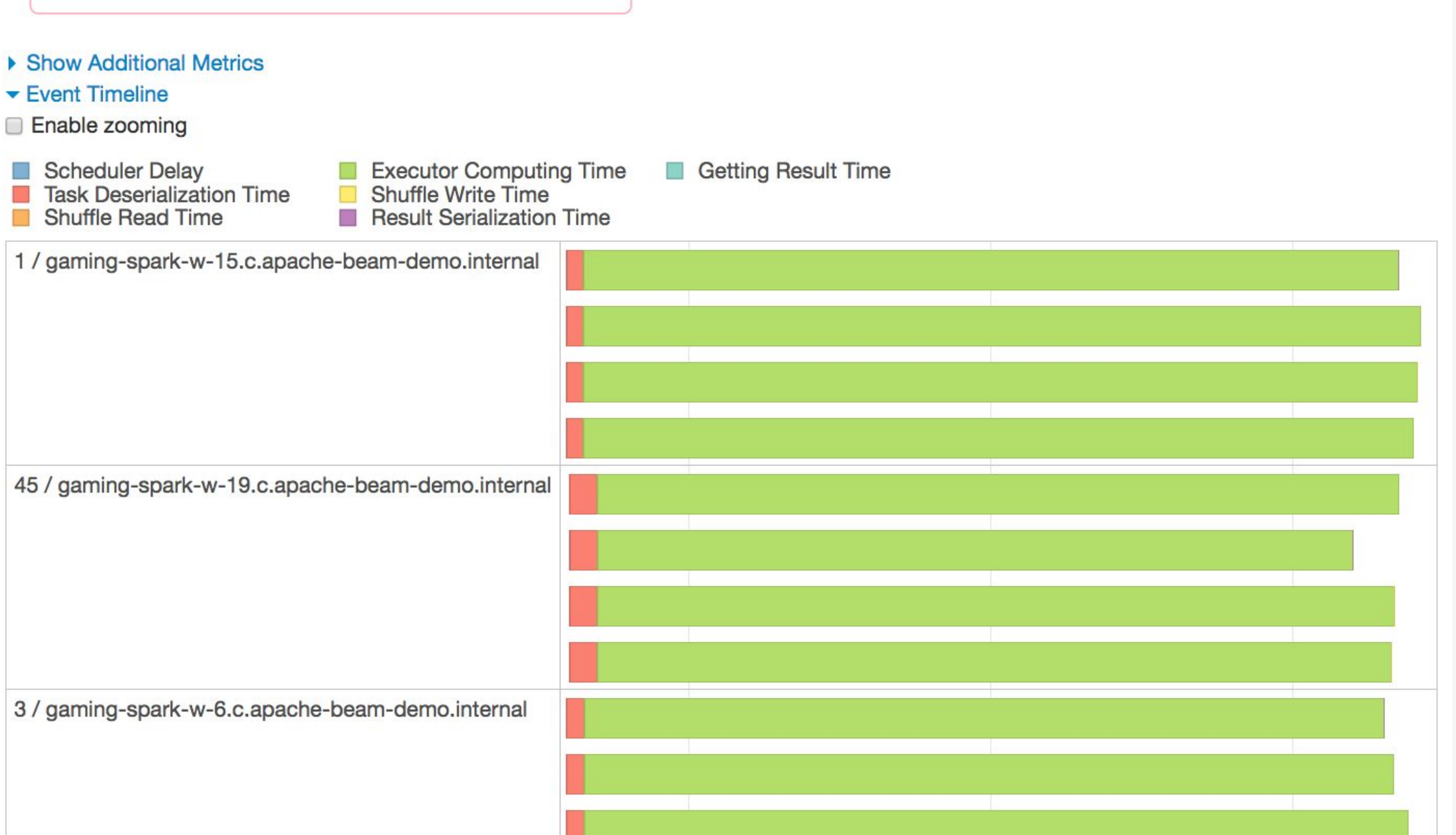
▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 201 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	1.1 min	2.5 min	2.6 min	2.7 min	3.5 min
GC Time	3 s	13 s	14 s	14 s	2.6 min
Shuffle Write Size / Records	3.3 KB / 28	4.8 KB / 40	5.0 KB / 42	5.2 KB / 43	5.6 KB / 47

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Shuffle Write Size / Records
1	gaming-spark-w-15.c.apache-beam-demo.internal:60941	11 min	4	0	4	21.1 KB / 173
2	gaming-spark-w-20.c.apache-beam-demo.internal:42339	10 min	4	0	4	19.9 KB / 173
3	gaming-spark-w-6.c.apache-beam-demo.internal:42214	11 min	4	0	4	20.1 KB / 167
4	gaming-spark-w-0.c.apache-beam-demo.internal:33132	9.3 min	4	0	4	20.4 KB / 168



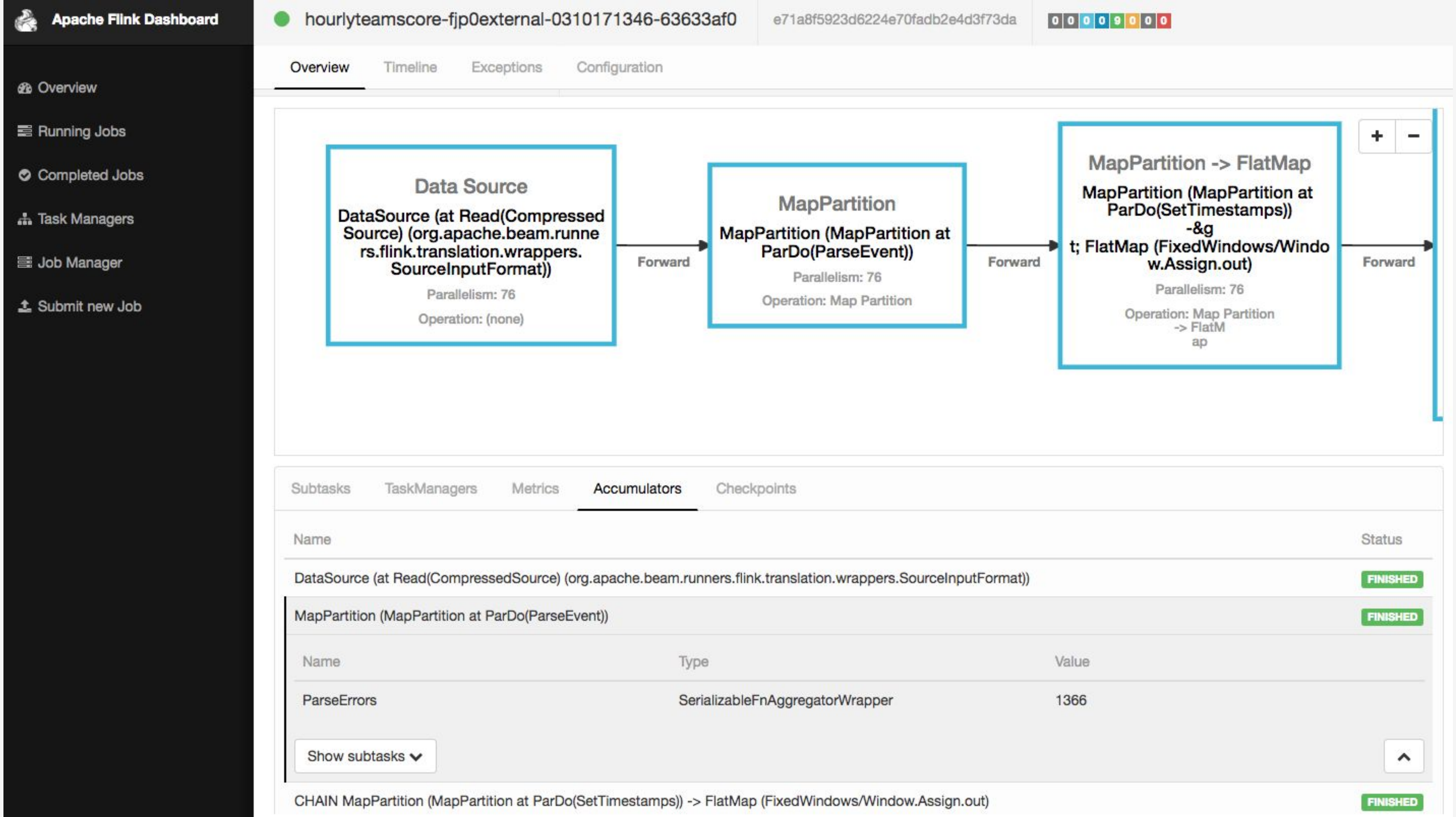
▸ Show Additional Metrics

▾ Event Timeline

☐ Enable zooming

🟦 Scheduler Delay	🟩 Executor Computing Time	🟦 Getting Result Time
🟥 Task Deserialization Time	🟨 Shuffle Write Time	
🟧 Shuffle Read Time	🟪 Result Serialization Time	

1 / gaming-spark-w-15.c.apache-beam-demo.internal
45 / gaming-spark-w-19.c.apache-beam-demo.internal
3 / gaming-spark-w-6.c.apache-beam-demo.internal



hourlyteamscore-fjp0external-0310171346-63633af0 e71a8f5923d6224e70fadb2e4d3f73da

Data Source
DataSource (at Read(CompressedSource) (org.apache.beam.runners.flink.translation.wrappers.SourceInputFormat))
Parallelism: 76
Operation: (none)

Forward

MapPartition
MapPartition (MapPartition at ParDo(ParseEvent))
Parallelism: 76
Operation: Map Partition

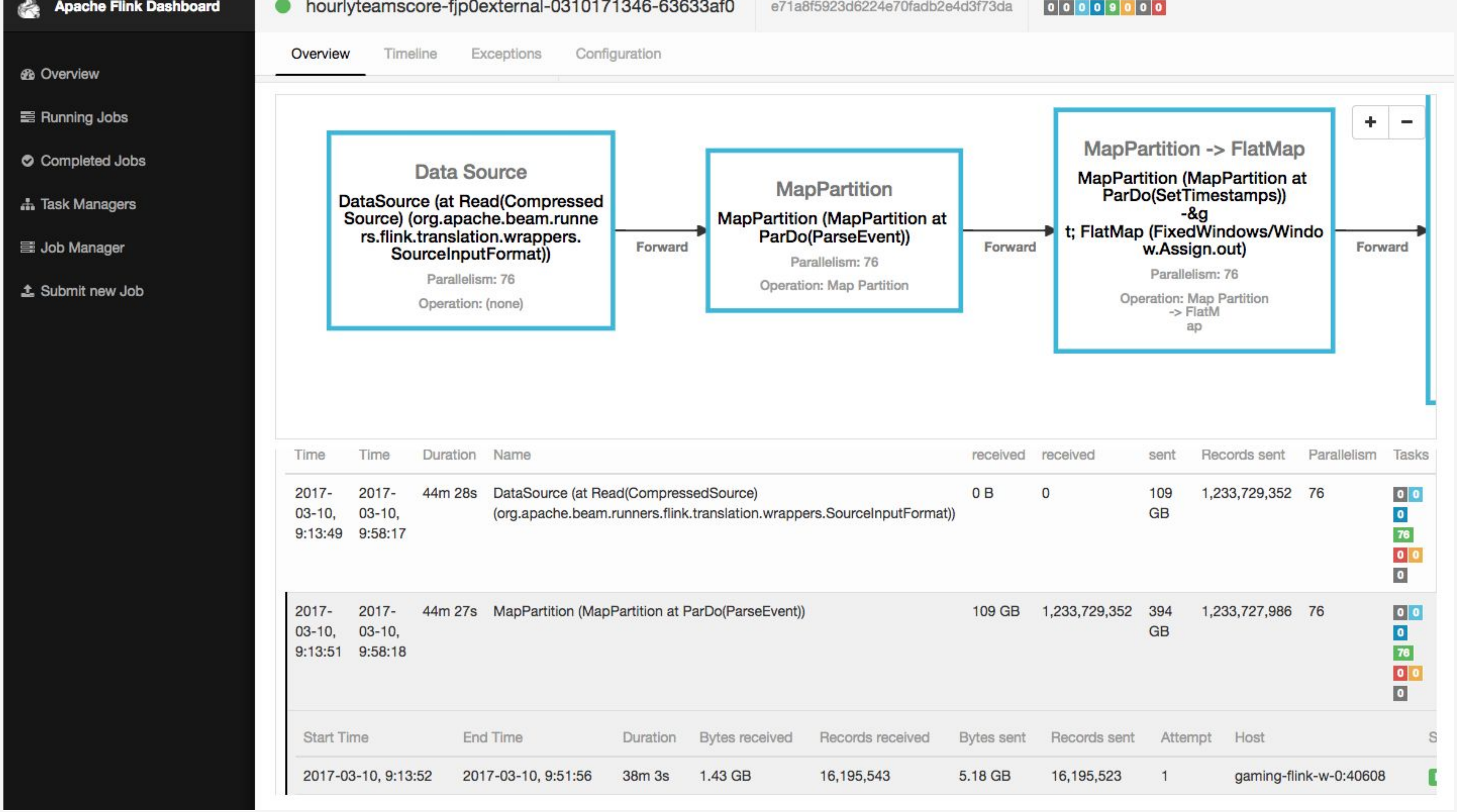
Forward

MapPartition -> FlatMap
MapPartition (MapPartition at ParDo(SetTimestamps)) -&gt; FlatMap (FixedWindows/Window.Assign.out)
Parallelism: 76
Operation: Map Partition -> FlatMap

Forward

Name	Status
DataSource (at Read(CompressedSource) (org.apache.beam.runners.flink.translation.wrappers.SourceInputFormat))	FINISHED
MapPartition (MapPartition at ParDo(ParseEvent))	FINISHED
CHAIN MapPartition (MapPartition at ParDo(SetTimestamps)) -> FlatMap (FixedWindows/Window.Assign.out)	FINISHED

Name	Type	Value
ParseErrors	SerializableFnAggregatorWrapper	1366



hourlyteamscore-fjp0external-0310171346-63633af0 e71a8f5923d6224e70fadb2e4d3f73da 0 0 0 0 9 0 0 0

Data Source
DataSource (at Read(CompressedSource) (org.apache.beam.runners.flink.translation.wrappers.SourceInputFormat))
Parallelism: 76
Operation: (none)

Forward

MapPartition
MapPartition (MapPartition at ParDo(ParseEvent))
Parallelism: 76
Operation: Map Partition

Forward

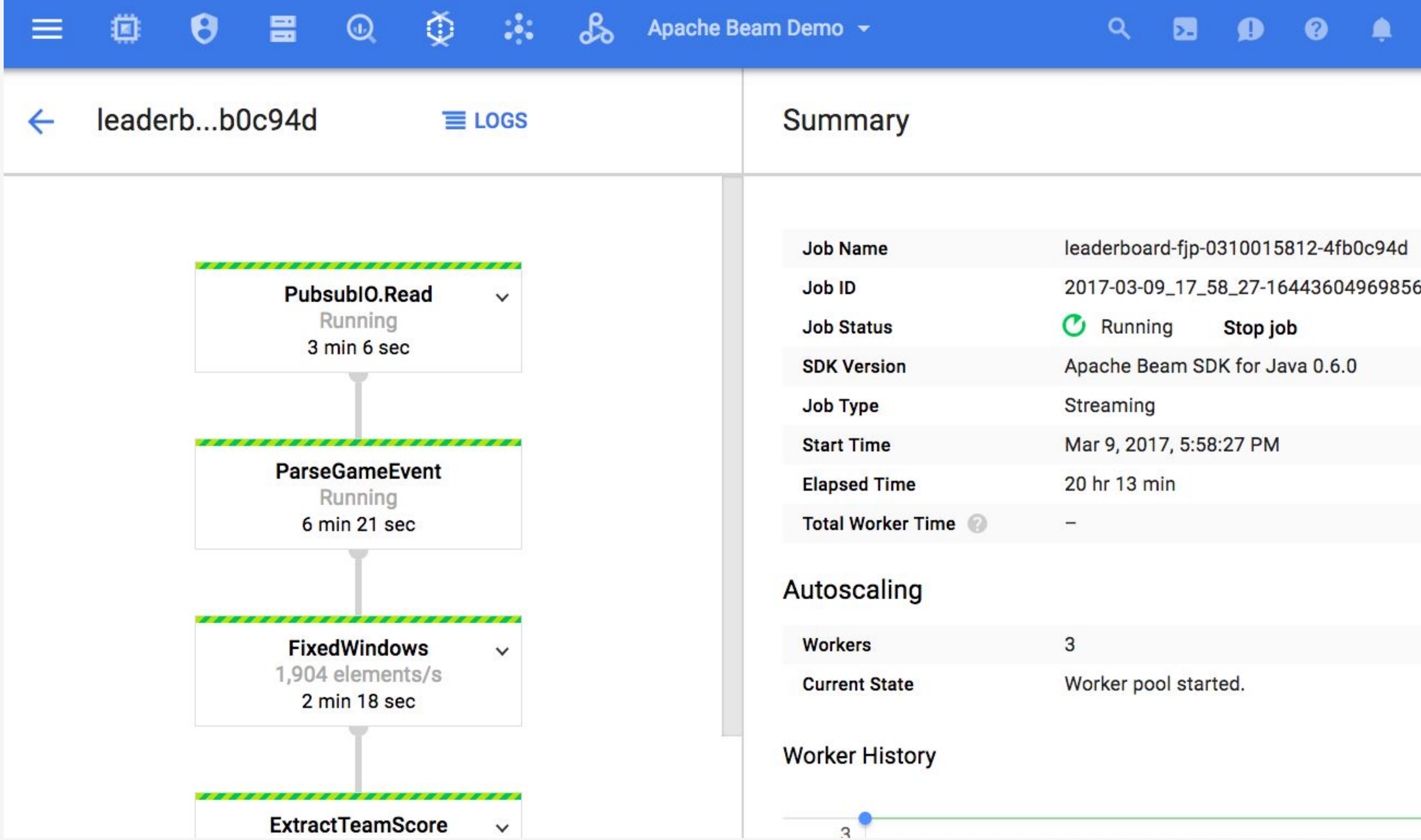
MapPartition -> FlatMap
MapPartition (MapPartition at ParDo(SetTimestamps)) -&g t; FlatMap (FixedWindows/Window.Assign.out)
Parallelism: 76
Operation: Map Partition -> FlatMap

Forward

Time	Time	Duration	Name	received	received	sent	Records sent	Parallelism	Tasks
2017-03-10, 9:13:49	2017-03-10, 9:58:17	44m 28s	DataSource (at Read(CompressedSource) (org.apache.beam.runners.flink.translation.wrappers.SourceInputFormat))	0 B	0	109 GB	1,233,729,352	76	0 0 0 76 0 0 0
2017-03-10, 9:13:51	2017-03-10, 9:58:18	44m 27s	MapPartition (MapPartition at ParDo(ParseEvent))	109 GB	1,233,729,352	394 GB	1,233,727,986	76	0 0 0 76 0 0 0

Start Time	End Time	Duration	Bytes received	Records received	Bytes sent	Records sent	Attempt	Host	S
2017-03-10, 9:13:52	2017-03-10, 9:51:56	38m 3s	1.43 GB	16,195,543	5.18 GB	16,195,523	1	gaming-flink-w-0:40608	

```
public static void main(String[] args) throws Exception {

  Options options =
      PipelineOptionsFactory.fromArgs(args).withValidation().as(Options.class);
  options.setStreaming(true);
  Pipeline pipeline = Pipeline.create(options);

  pipeline
      .apply(PubsubIO.<String>read()
          .timestampLabel(TIMESTAMP_ATTRIBUTE).topic(options.getTopic())
          .withCoder(StringUtf8Coder.of()))
      .apply("ParseGameEvent", ParDo.of(new ParseEventFn()))

      .apply("FixedWindows", Window.<GameActionInfo>into(FixedWindows.of(FIVE_MINUTES))
              .triggering(AfterWatermark.pastEndOfWindow()
                  .withEarlyFirings(AfterProcessingTime.pastFirstElementInPane()
                      .plusDelayOf(TWO_MINUTES))
                  .withLateFirings(AfterPane.elementCountAtLeast(1)))
          .withAllowedLateness(TEN_MINUTES)
          .accumulatingFiredPanes())

       .apply("ExtractTeamScore", new CalculateTeamScores(options.getOutputPrefix()));

  pipeline.run();
```



Summary

Job Name	leaderboard-fjp-0310015812-4fb0c94d
Job ID	2017-03-09_17_58_27-16443604969856
Job Status	Running
SDK Version	Apache Beam SDK for Java 0.6.0
Job Type	Streaming
Start Time	Mar 9, 2017, 5:58:27 PM
Elapsed Time	20 hr 13 min
Total Worker Time	–

Autoscaling

Workers	3
Current State	Worker pool started.

Worker History

Getting Started with Apache Beam

Quickstarts

- Java SDK
- Python SDK

Example walkthroughs

- Word Count
- Mobile Gaming

Extensive documentation

Extensibility to integrate the entire Big Data ecosystem

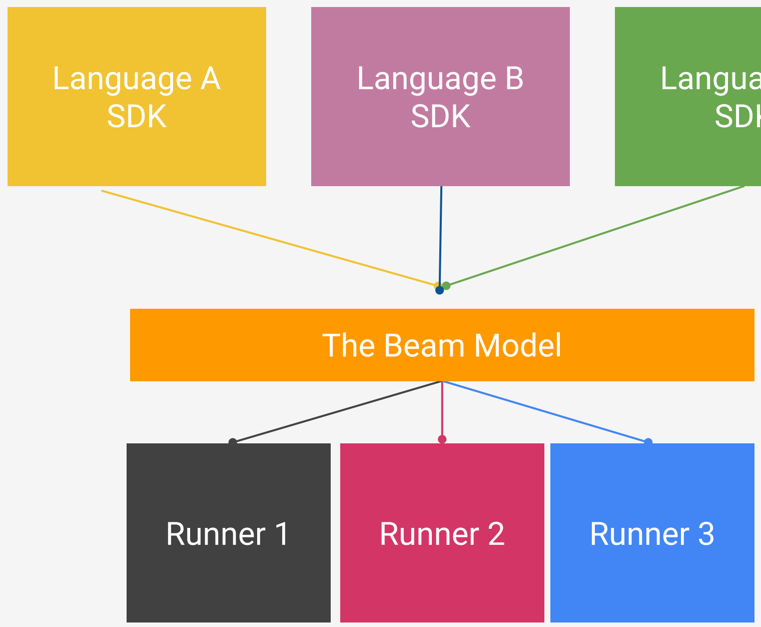
“Integrating Up, Down, and Sideways”

Extensibility points

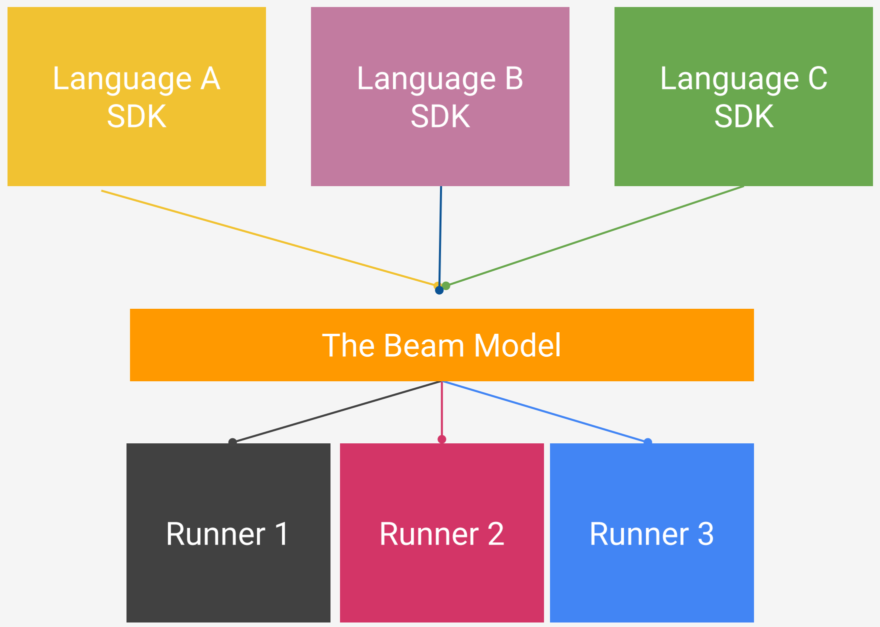
- Software Development Kits (SDKs)
- Runners

- Domain-specific extensions (DSLs)
- Libraries of transformations
- IOs
- File systems

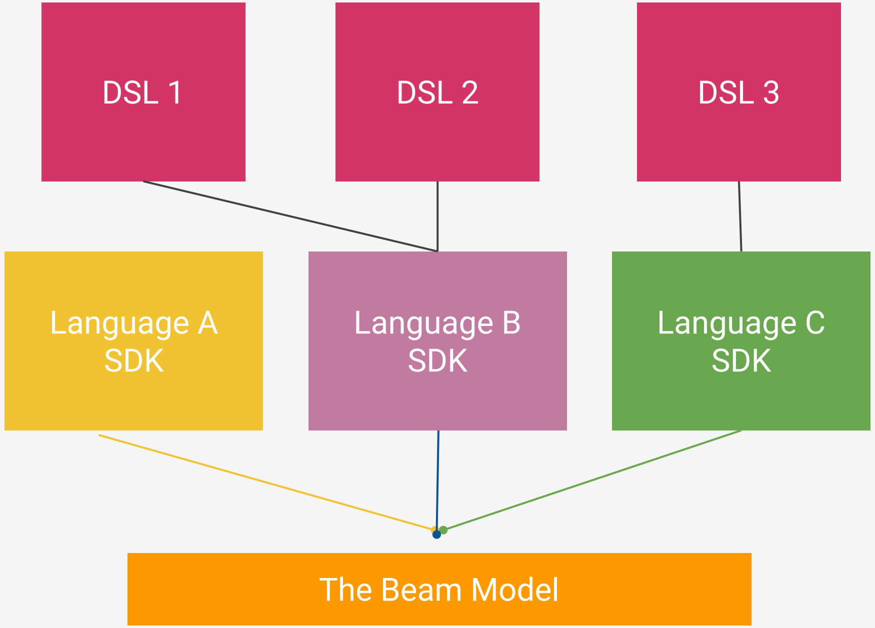
Software Development Kits (SDKs)



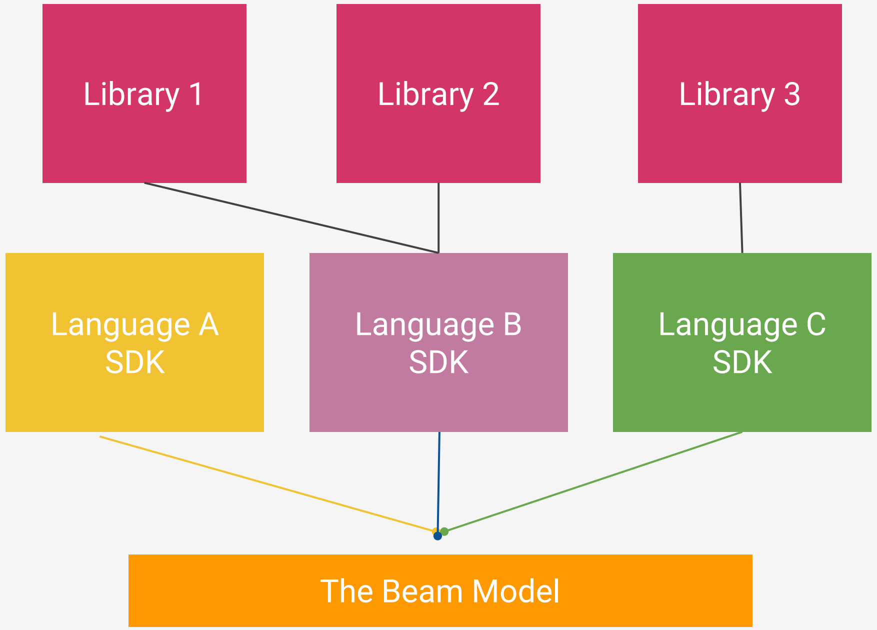
Runners


Language A SDK
Language B SDK
Language C SDK
The Beam Model
Runner 1
Runner 2
Runner 3

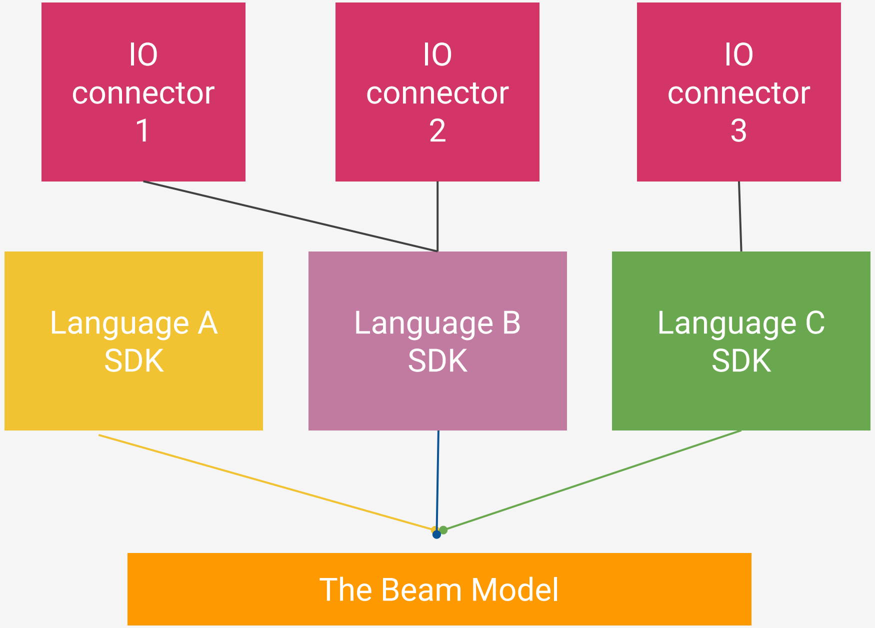
Domain-specific extensions (DSLs)



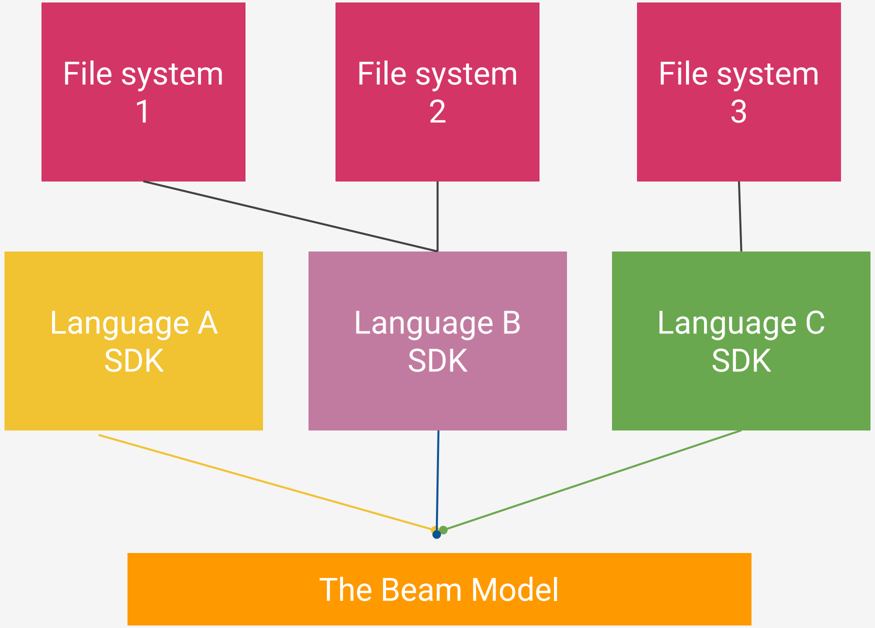
Libraries of transformations



IO connectors



File systems



Ecosystem integration

- I have an engine
  → write a Beam runner
- I want to extend Beam to new languages
  → write an SDK
- I want to adopt an SDK to a target audience
  → write a DSL
- I want a component can be a part of a bigger data-processing pipeline
  → write a library of transformations
- I have a data storage or messaging system
  → write an IO connector or a file system connector

Apache Beam is
a ***glue*** that integrates
the big data ecosystem

Learn more and get involved!

Attend a birds-of-a-feather session later today!

Apache Beam

https://beam.apache.org

Join the Beam mailing lists!

user-subscribe@beam.apache.org

dev-subscribe@beam.apache.org

Follow @ApacheBeam on Twitter

Apache Beam at this conference

- *Using Apache Beam for Batch, Streaming, and Everything in Between*
  - Dan Halperin @ 10:15 am
- *Apache Beam: Integrating the Big Data Ecosystem Up, Down, and Sideways*
  - Davor Bonaci, and Jean-Baptiste Onofré @ 11:15 am
- *Concrete Big Data Use Cases Implemented with Apache Beam*
  - Jean-Baptiste Onofré @ 12:15 pm
- *Nexmark, a Unified Framework to Evaluate Big Data Processing Systems*
  - Ismael Mejia, and Etienne Chauchot @ 2:30 pm

Apache Beam at this conference

- *Apache Beam Birds of a Feather*
  - Wednesday, 6:30 pm - 7:30 pm

- *Apache Beam Hacking Time*
  - Time: all-day Thursday
  - 2nd floor, collaboration area
  - (depending on interest)

APACHE:
BIG_DATA
NORTH_AMERICA