Introduction to Apache Beam

Some of the slides (the really nice ones!) were contributed by Frances Perry & Tyler Akidau, April 2016

About me

Software engineer @PayPal, working on streaming data processing.

PMC member, committer @ApacheBeam, Spark runner lead.

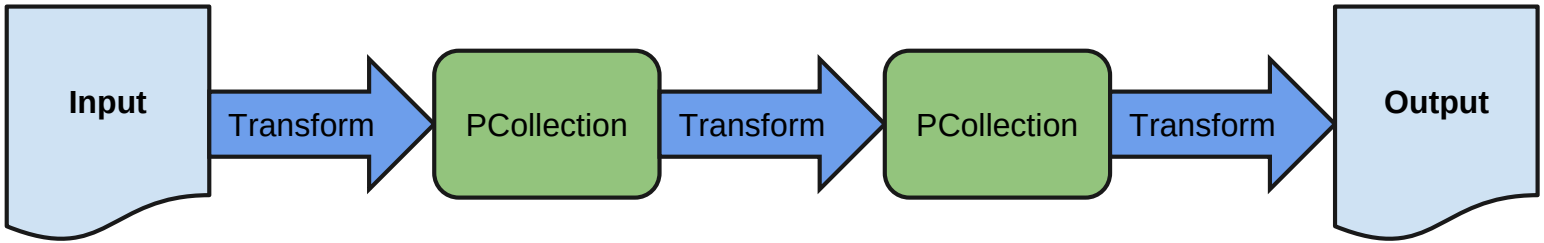
“A unified programming model for batch and streaming data processing, that can be executed on various processing engines”

What's in the box?

- SDKs for writing Beam pipelines -- Java and Python
- The Beam Model: **What** / **Where** / **When** / **How**
- Runners for existing distributed processing backends
  - Apache Apex
  - Apache Flink
  - Apache Spark
  - Google Cloud Dataflow
  - Direct (in-process) runner for testing

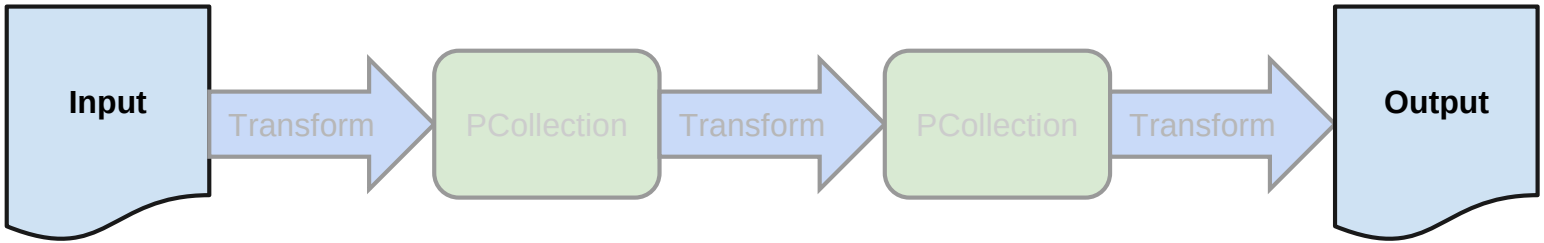
The Beam Pipeline

The Beam pipeline: overview



I/O

The Beam pipeline: IOs



Pipeline IOs: bounded source

```
TextIO.Read.Bound readText = TextIO.Read.from(“path/to/input.txt”);
```

```
TextIO.Write.Bound writeText = TextIO.Write.to(“path/to/output”);
```

```
pipeline
    .apply("ReadLines", readText)
    .apply(“CountWords”, new CountWords())
    .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
    .apply(“WriteFormatted”, writeText);
```

*Some code snippets were shortened or elided for clarity.

Pipeline IOs: unbounded source

```
KafkaIO.Read<Integer, String> readKafka =
KafkaIO.<Integer, String>read().withTopic(“my_input_topic”)...
```

```
KafkaIO.Write<Integer, String> writeKafka =
KafkaIO.<Integer, String>write().withTopic(“my_output_topic”)...
```

```
pipeline
    .apply("ReadLines", readKafka.values())
    .apply(“CountWords”, new CountWords())
    .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
    .apply(“WriteFormatted”, writeKafka.values());
```

*Some code snippets were shortened or elided for clarity.

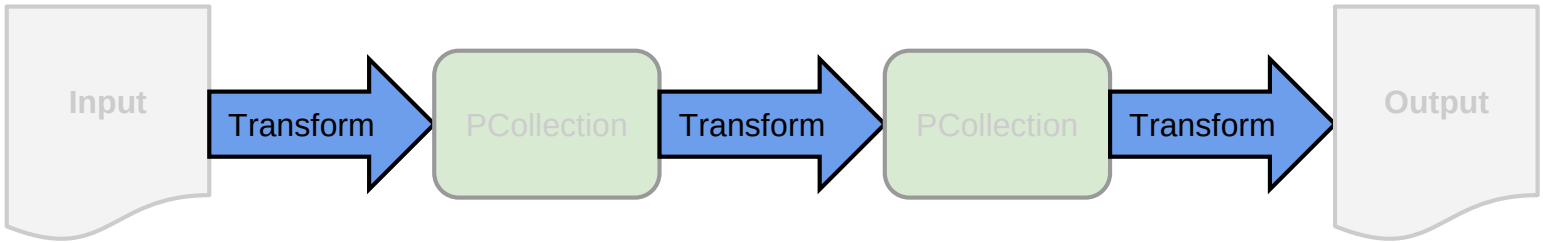
Supported IOs (April 2017)

- HDFS
- HBase
- JDBC
- MongoDB
- Elasticsearch
- Kafka
- Kinesis
- JMS
- MQTT
- Google - GCS, BigQuery, BigTable, Datastore

Most of this was done in little over a year, thanks to the Beam community!

Transformations

The Beam pipeline: transfomations



SDK core primitives

SDK transformations are mostly built on top of the following core primitives:

- ParDo - executing the user's DoFn function ~ map/flatmap.
- GroupByKey - grouping by key **and window**.
- Window.into - applying a window to a PCollection.
- Flatten.pCollections - union one or more PCollections into a single PCollection.

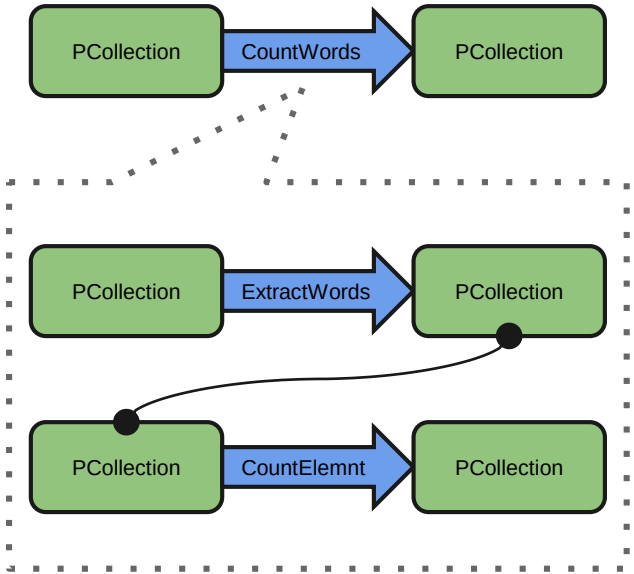
CountWords

```
pipeline
      .apply("ReadLines", TextIO.Read.from(options.getInputFile()))
      .apply(“CountWords”, new CountWords())
      .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
      .apply("WriteCounts", TextIO.Write.to(options.getOutput()));
```

*Some code snippets were shortened or elided for clarity.

The CountWords composite

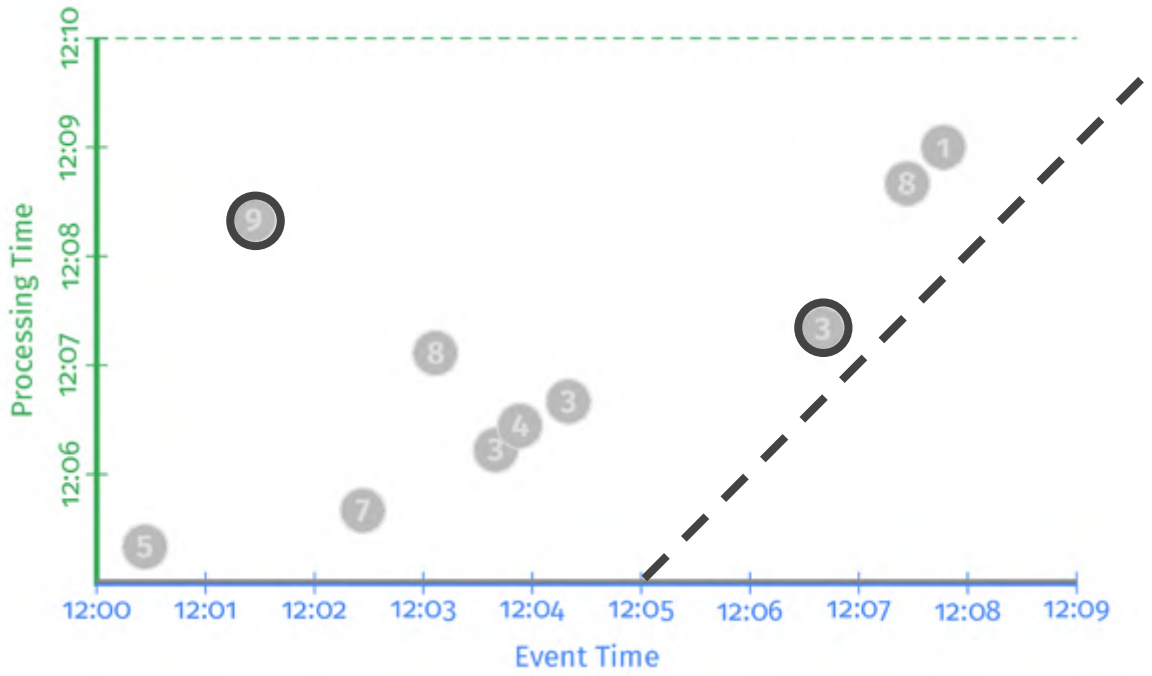
```
// Convert lines of text into individual words.
PCollection<String> words =
lines.apply(ParDo.of(new ExtractWordsFn()));

// Count the number of times each word occurs.
PCollection<KV<String, Long>> wordCounts =
words.apply(Count.<String>perElement());
```



*Some code snippets were shortened or elided for clarity.

The Beam Model

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?

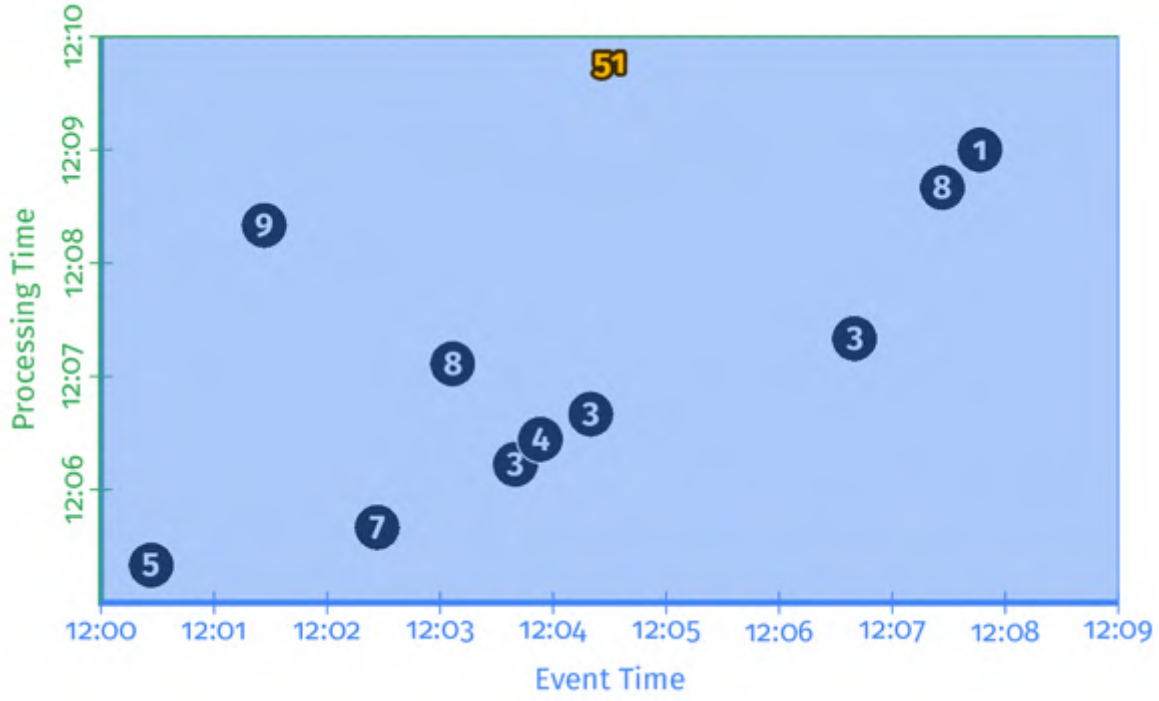
How do refinements of results relate?

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

```
PCollection<KV<String, Integer>> scores = input
.apply(Sum.integersPerKey());
```

*Some code snippets were shortened or elided for clarity.

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

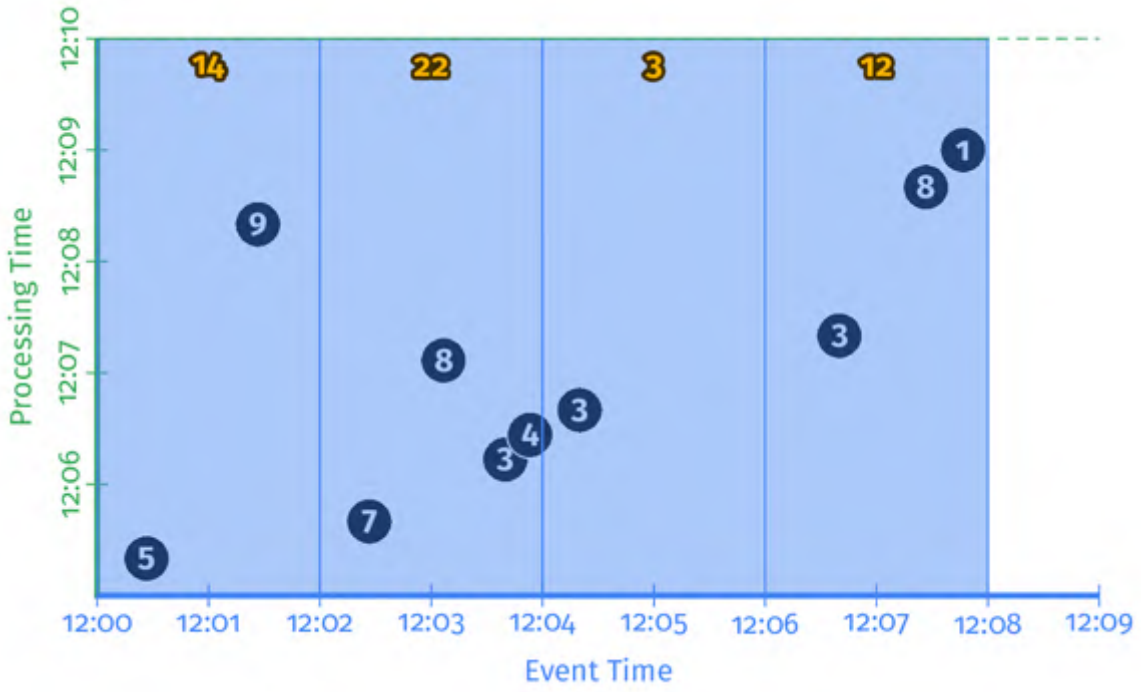


The Beam Model: **Where** in event time?

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

*Some code snippets were shortened or elided for clarity.

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

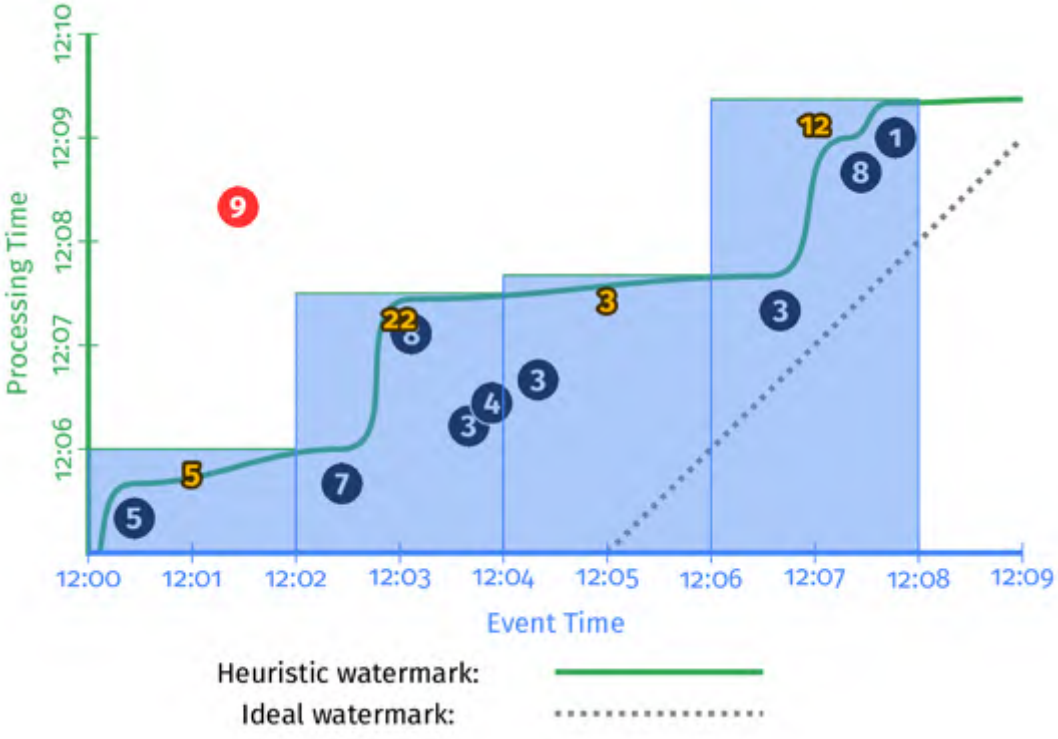


The Beam Model: **When** in processing time?

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

*Some code snippets were shortened or elided for clarity.

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

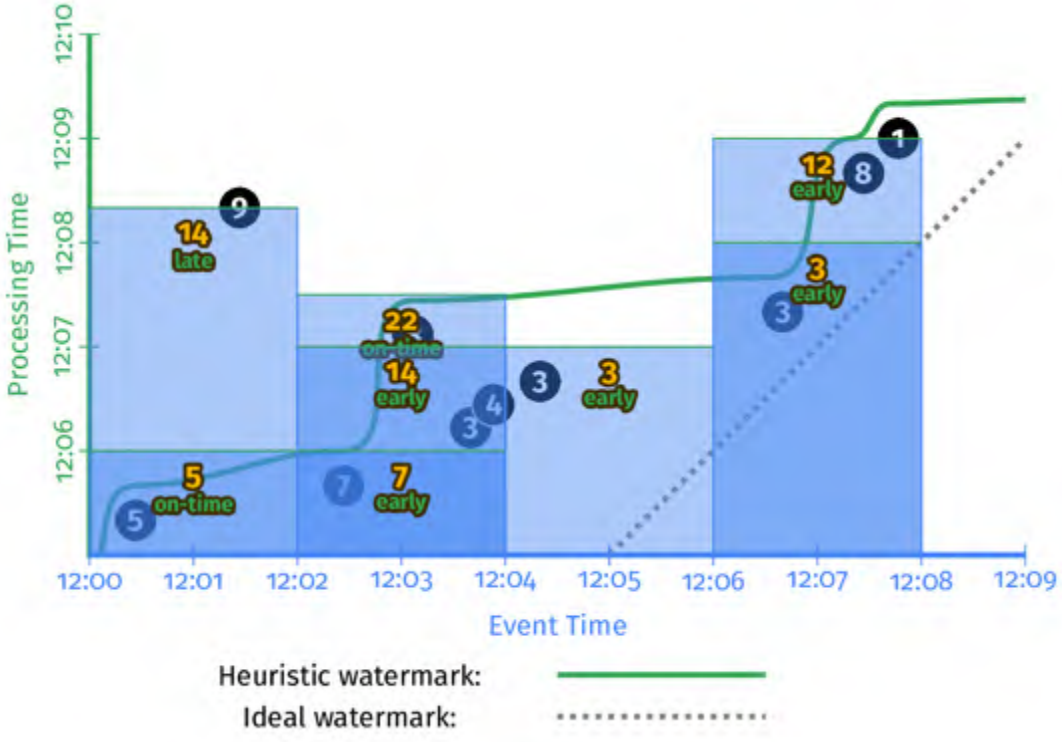


The Beam Model: How do refinements relate?

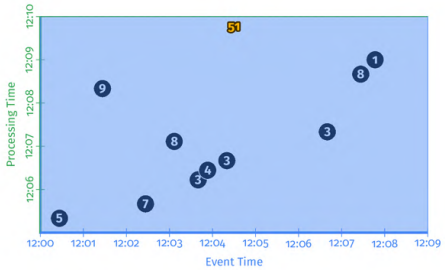
```
PCollection<KV<String, Integer>> scores = input
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
           .triggering(AtWatermark()
              .withEarlyFirings(AtPeriod(Duration.standardMinutes(1)))
              .withLateFirings(AtCount(1)))
           .accumulatingFiredPanes())
    .apply(Sum.integersPerKey());
```

*Some code snippets were shortened or elided for clarity.

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

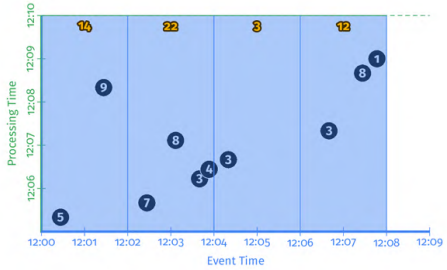


Customizing What Where When How



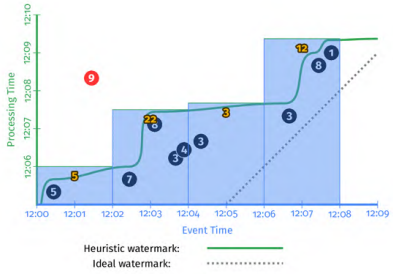
1

Classic Batch



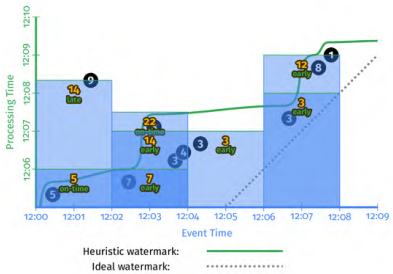
2

Windowed Batch



3

Streaming



4

Streaming + Accumulation

For more information see https://beam.apache.org/get-started/mobile-gaming-example/

The Beam streaming pipeline

```
pipeline
    .apply(KafkaIO.read().withTopic(“team_points_topic”))
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
           .triggering(AtWatermark()
              .withEarlyFirings(AtPeriod(Duration.standardMinutes(1)))
              .withLateFirings(AtCount(1))
           .accumulatingFiredPanes())
    .apply(Sum.integersPerKey())
    .apply(KafkaIO.write().withTopic(“team_points_topic”))
```

*Some code snippets were shortened or elided for clarity.

Portability

Direct runner

```
PipelineOptions options = PipelineOptionsFactory.create();
Pipeline pipeline = Pipeline.create(options);


pipeline
     .apply("ReadLines", TextIO.Read.from(options.getInputFile()))
     .apply(“CountWords”, new CountWords())
     .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
     .apply("WriteCounts", TextIO.Write.to(options.getOutput()));
```

*Some code snippets were shortened or elided for clarity.

Flink runner

```
FlinkPipelineOptions flinkPipelineOptions =
PipelineOptionsFactory.as(FlinkPipelineOptions.class);
flinkPipelineOptions.setRunner(FlinkRunner.class);
Pipeline pipeline = Pipeline.create(flinkPipelineOptions);

pipeline
     .apply("ReadLines", TextIO.Read.from(options.getInputFile()))
     .apply(“CountWords”, new CountWords())
     .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
     .apply("WriteCounts", TextIO.Write.to(options.getOutput()));
```

*Some code snippets were shortened or elided for clarity.

Spark runner

```
SparkPipelineOptions sparkPipelineOptions =
PipelineOptionsFactory.as(SparkPipelineOptions.class);
sparkPipelineOptions.setRunner(SparkRunner.class);
Pipeline pipeline = Pipeline.create(sparkPipelineOptions);

pipeline
     .apply("ReadLines", TextIO.Read.from(options.getInputFile()))
     .apply(“CountWords”, new CountWords())
     .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
     .apply("WriteCounts", TextIO.Write.to(options.getOutput()));
```

*Some code snippets were shortened or elided for clarity.

Spark runner

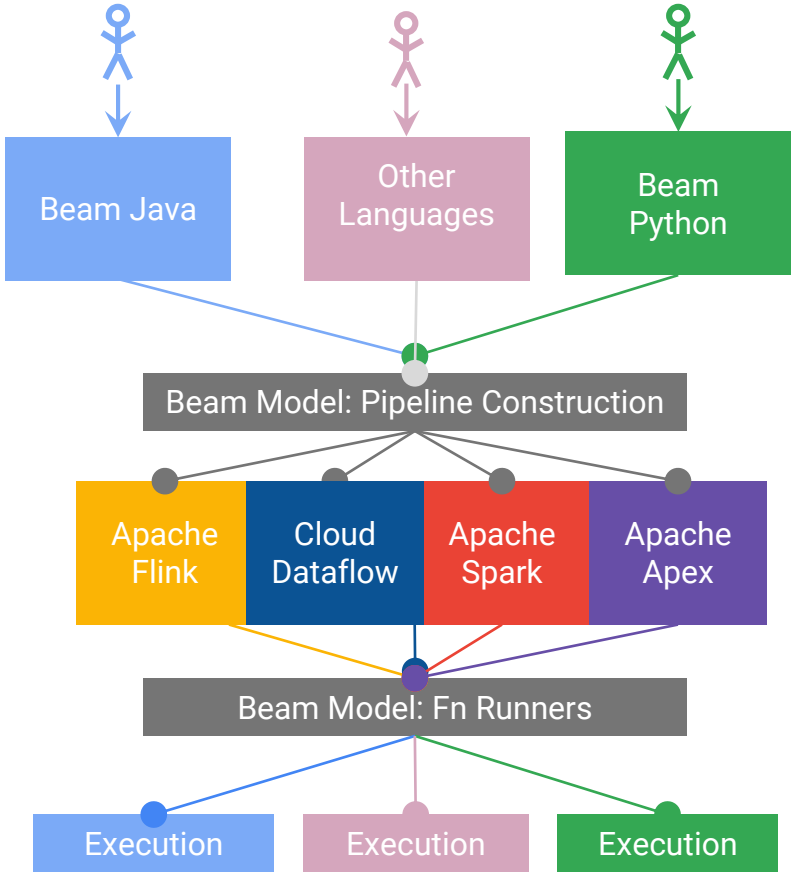
```
SparkPipelineOptions sparkPipelineOptions =
PipelineOptionsFactory.as(SparkPipelineOptions.class);
sparkPipelineOptions.setRunner(SparkRunner.class);
sparkPipelineOptions.setSparkMaster(“spark://IP:PORT”);
Pipeline pipeline = Pipeline.create(sparkPipelineOptions);
pipeline
     .apply("ReadLines", TextIO.Read.from(options.getInputFile()))
     .apply(“CountWords”, new CountWords())
     .apply(“FormatAsText”, MapElements.via(new FormatAsTextFn()))
     .apply("WriteCounts", TextIO.Write.to(options.getOutput()));
```

*Some code snippets were shortened or elided for clarity.

The Apache Beam Vision

The Apache Beam vision

1. **End users:** who want to write pipelines in a language that's familiar.
2. **SDK writers:** who want to make Beam concepts available in new languages.
3. **Runner writers:** who have a distributed processing environment and want to support Beam pipelines



Learn more!

Apache Beam

https://beam.apache.org

The World Beyond Batch 101 & 102

https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101
https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102

Join the mailing lists!

user-subscribe@beam.apache.org
dev-subscribe@beam.apache.org

Follow @ApacheBeam on Twitter

Thank you!