Dataflow/Apache Beam

A Unified Model for Batch and Streaming Data Processing

Eugene Kirpichov, Google

STREAM 2016

Google Cloud Platform

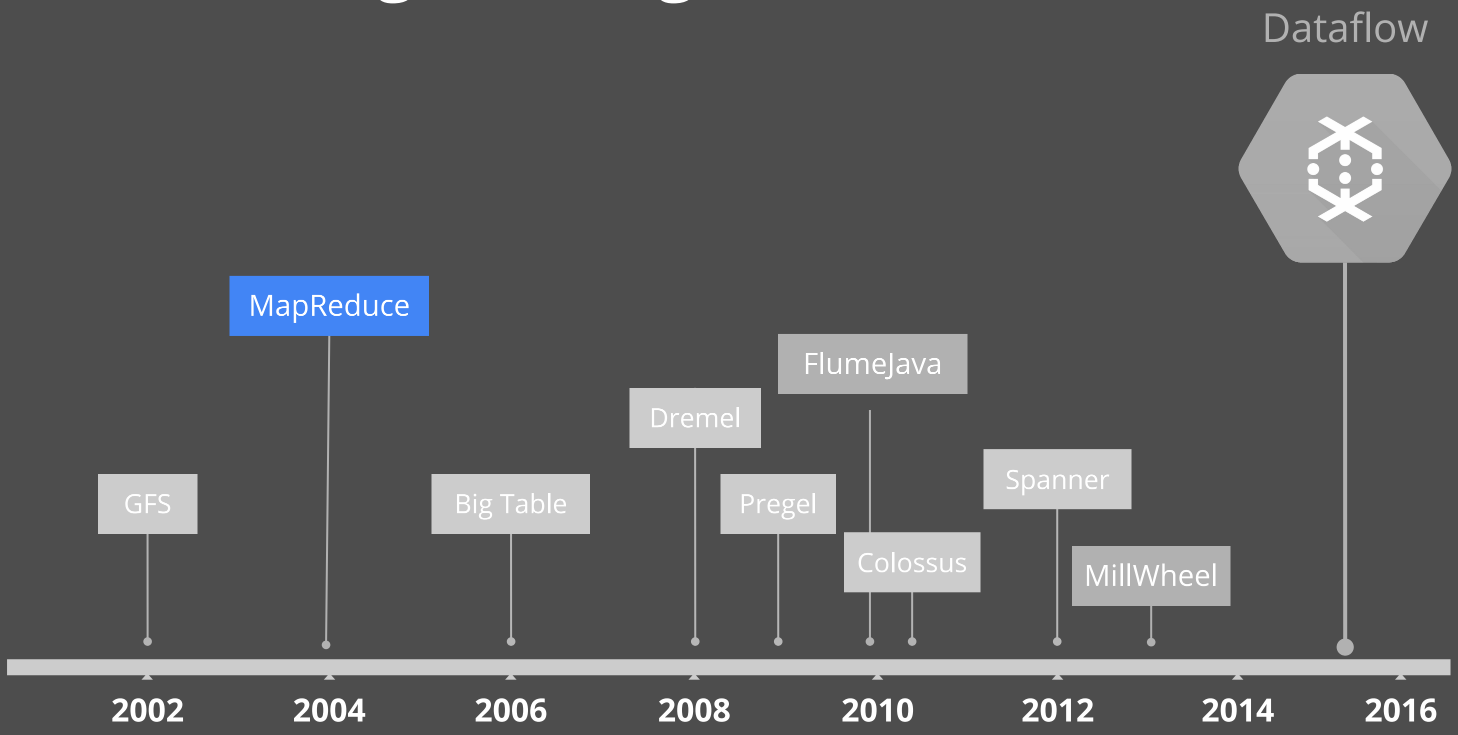
Agenda

1. Google's Data Processing Story
2. Philosophy of the Beam programming model
3. Apache Beam project

1

Google's
Data Processing Story

Data Processing @ Google



MapReduce: SELECT + GROUP BY

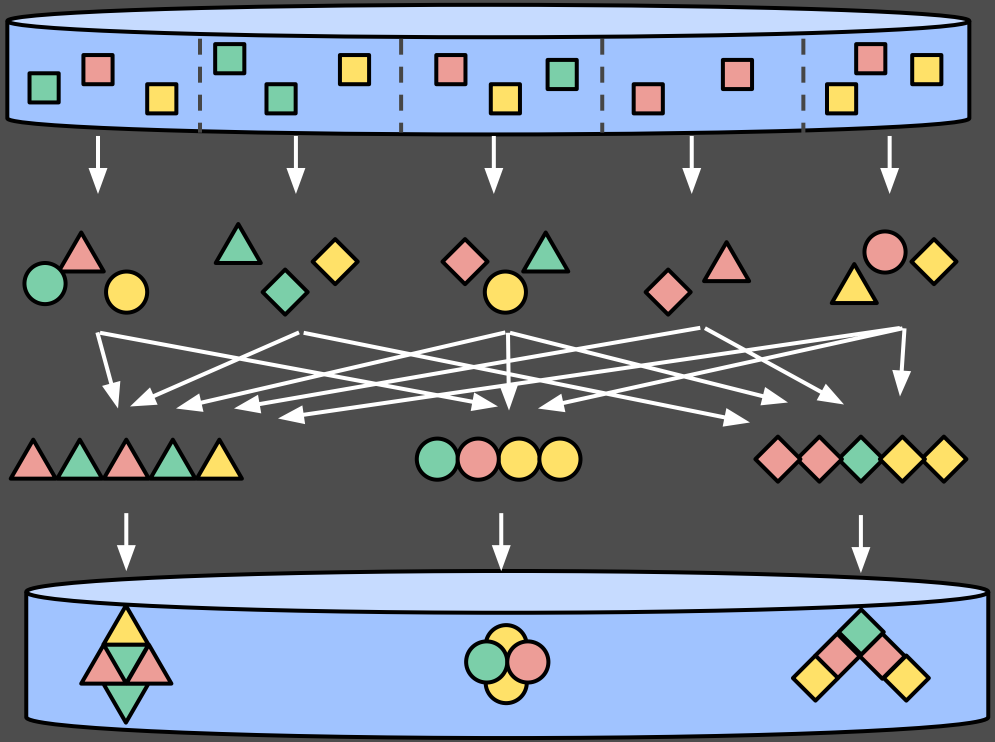
(distributed input dataset)

Map (SELECT)

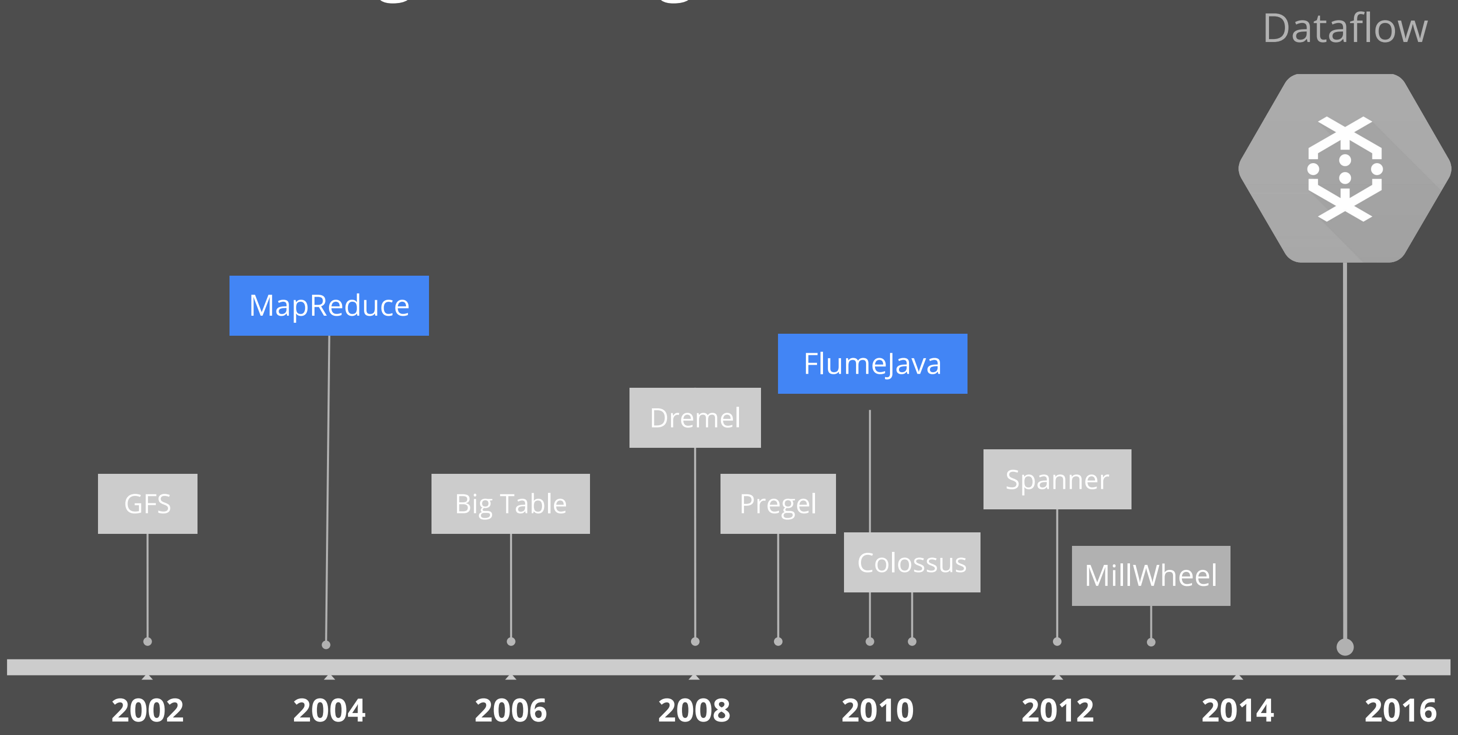
Shuffle (GROUP BY)

Reduce (SELECT)

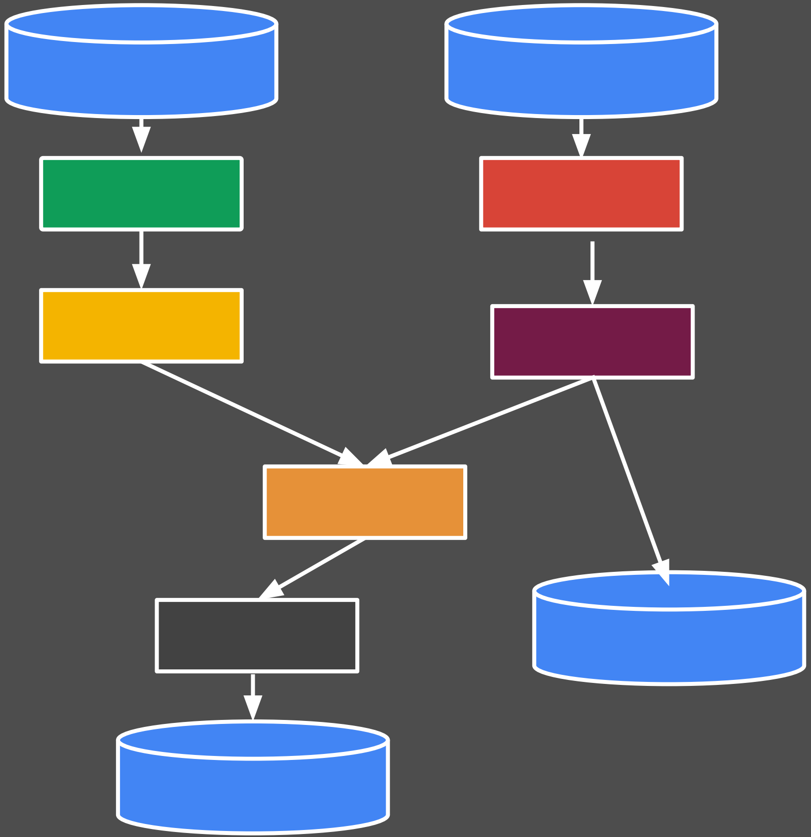
(distributed output dataset)



Data Processing @ Google



FlumeJava Pipelines

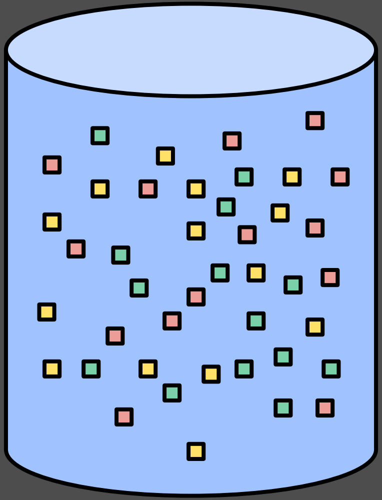


- A Pipeline represents a graph of data processing transformations
- PCollections flow through the pipeline
- Optimized and executed as a unit for efficiency

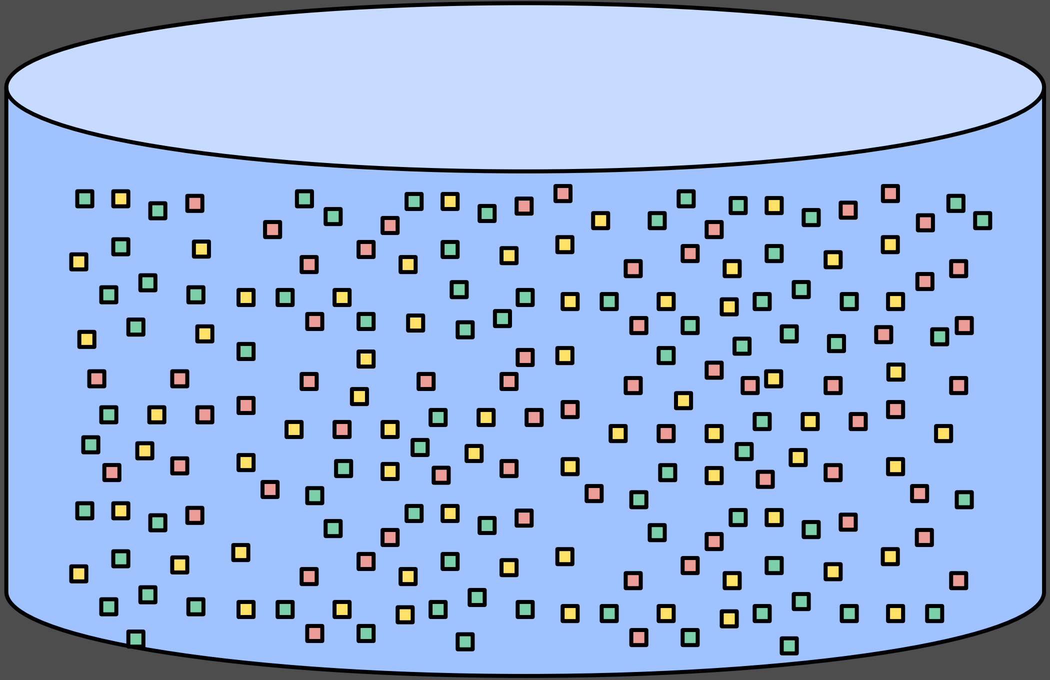
Example: Computing mean temperature

```
// Collection of raw events
PCollection<SensorEvent> raw = ...;
// Element-wise extract location/temperature pairs
PCollection<KV<String, Double>> input =
    raw.apply(ParDo.of(new ParseFn()))
// Composite transformation containing an aggregation
PCollection<KV<String, Double>> output = input
  .apply(Mean.<Double>perKey());
// Write output
output.apply(BigtableIO.Write.to(...));
```

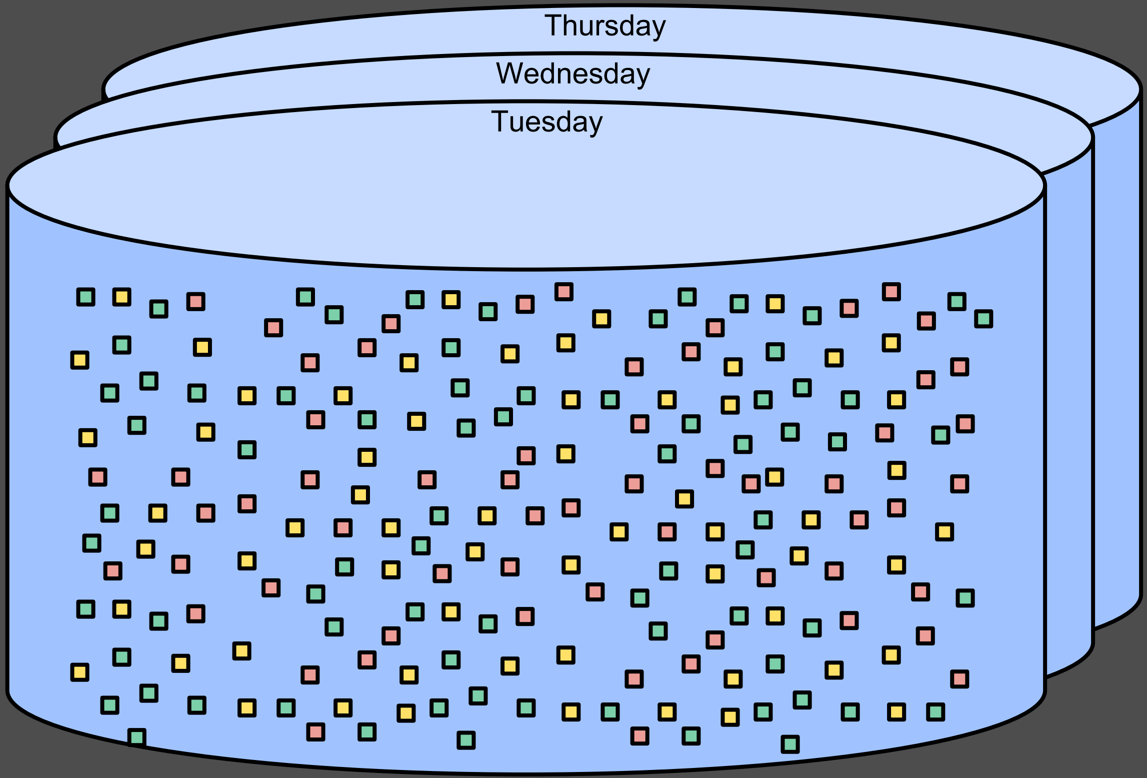
So, people used FJ to process data...



...big data...



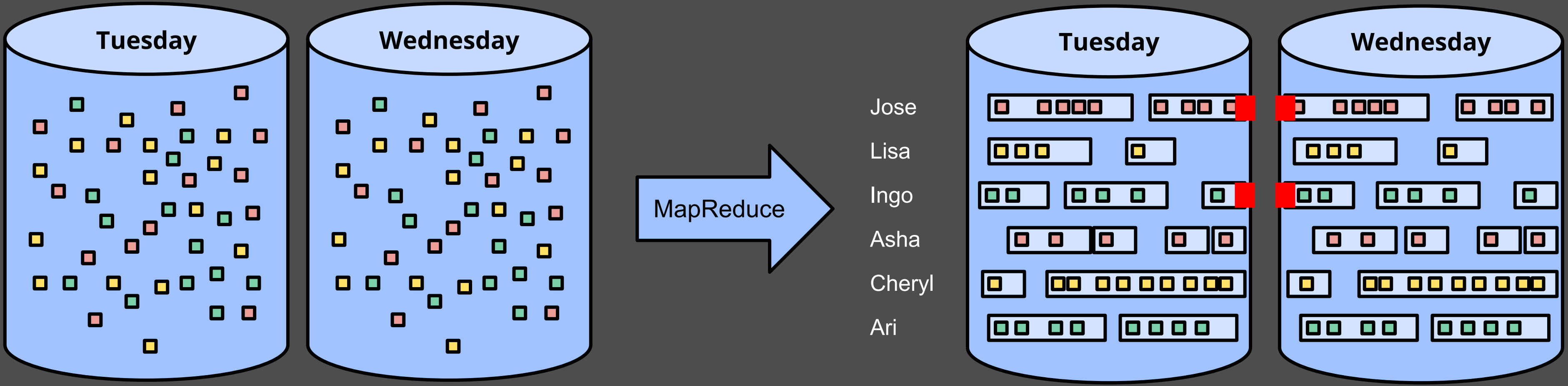
...really, really big...



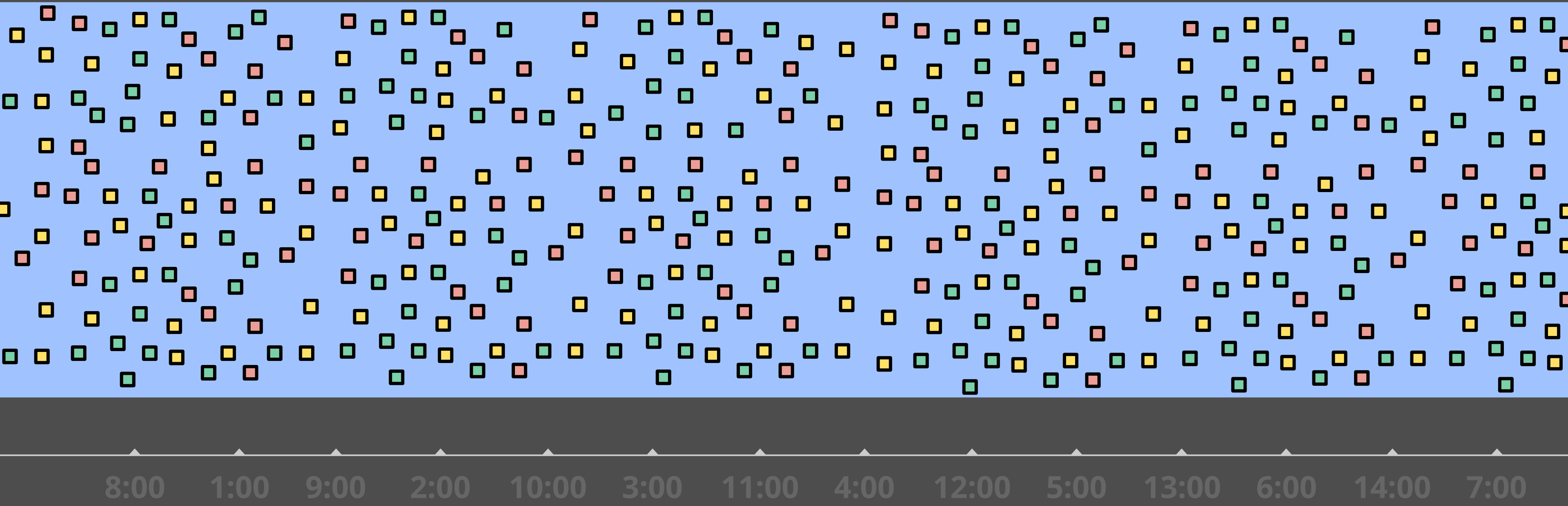
Batch failure mode #1

Latency

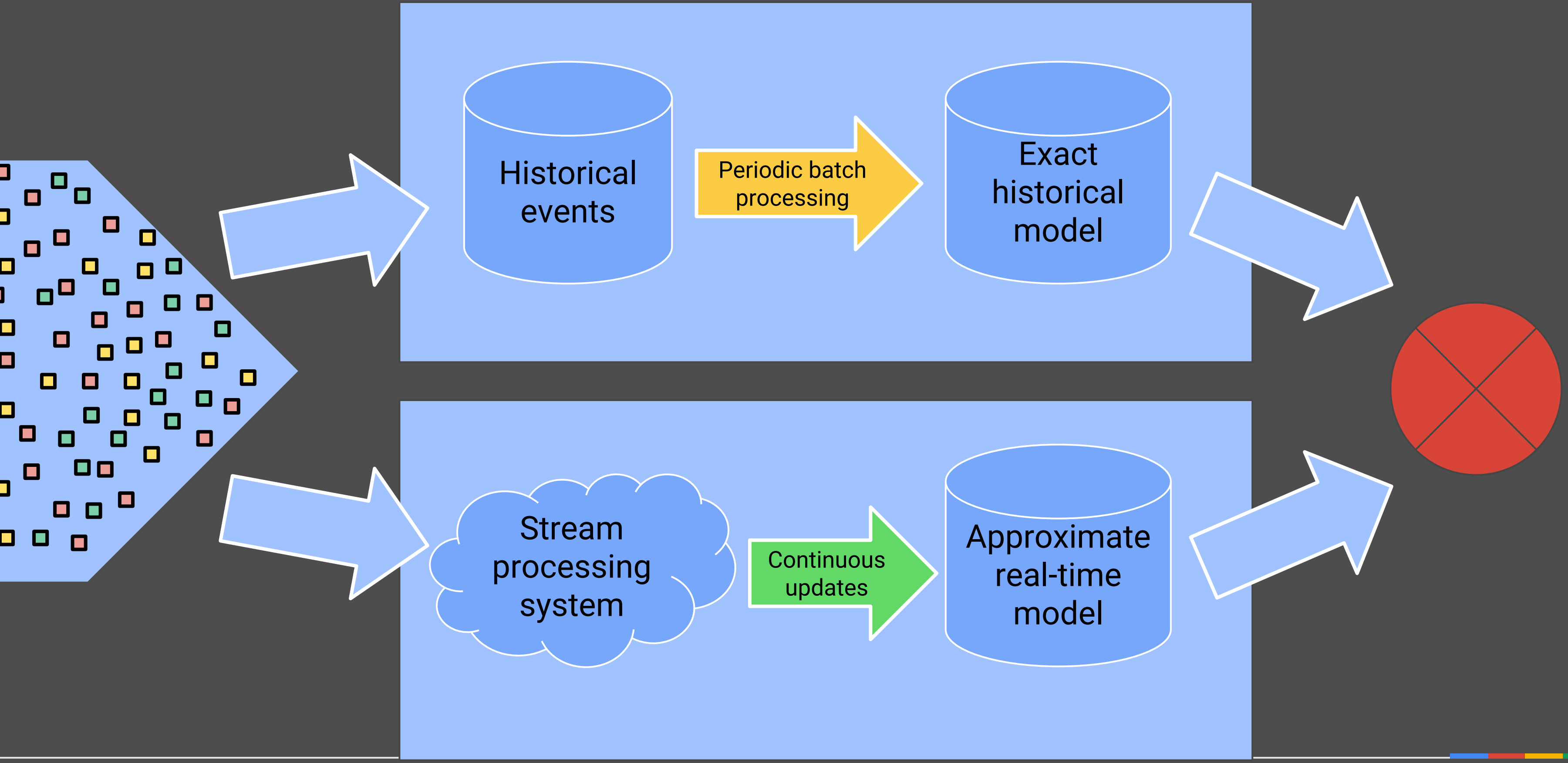
Batch failure mode #2: Sessions



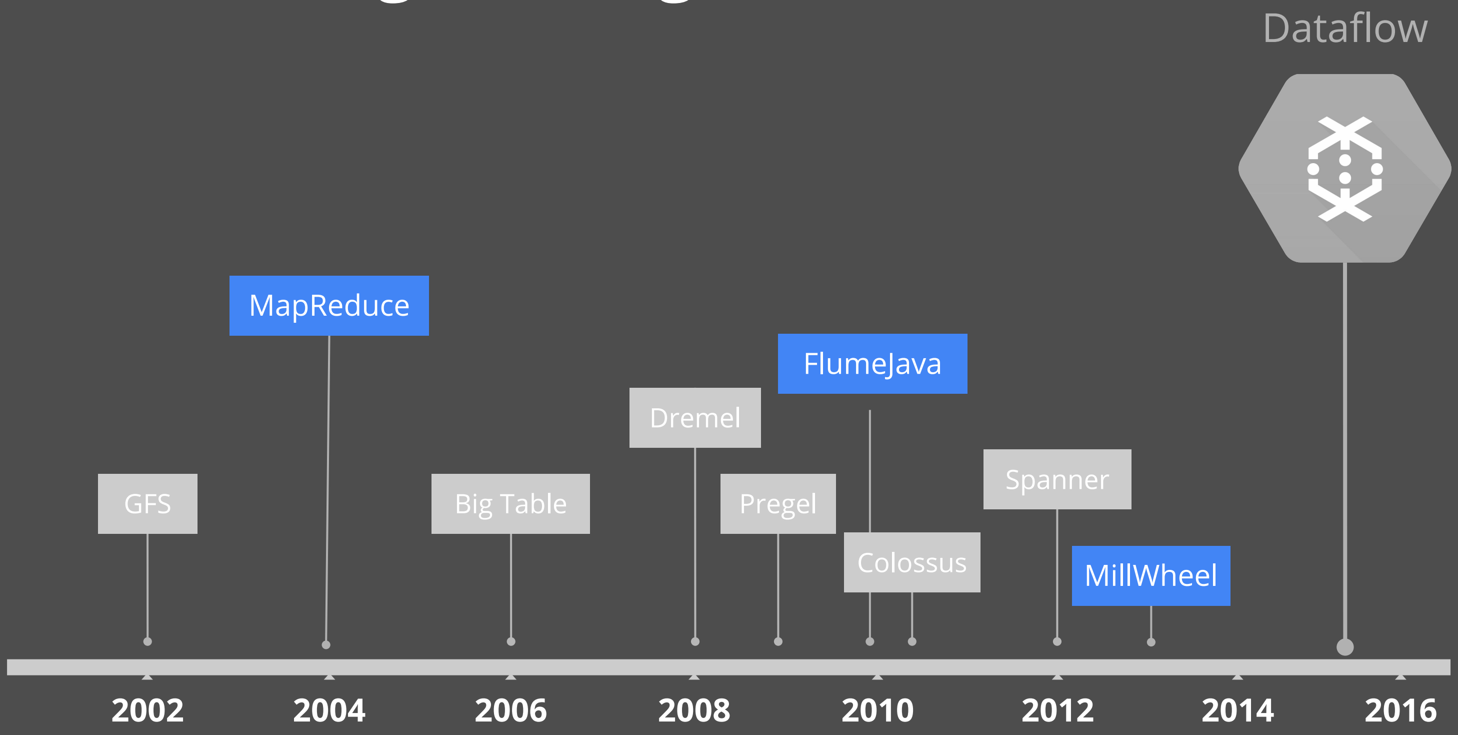
Continuous & Unbounded



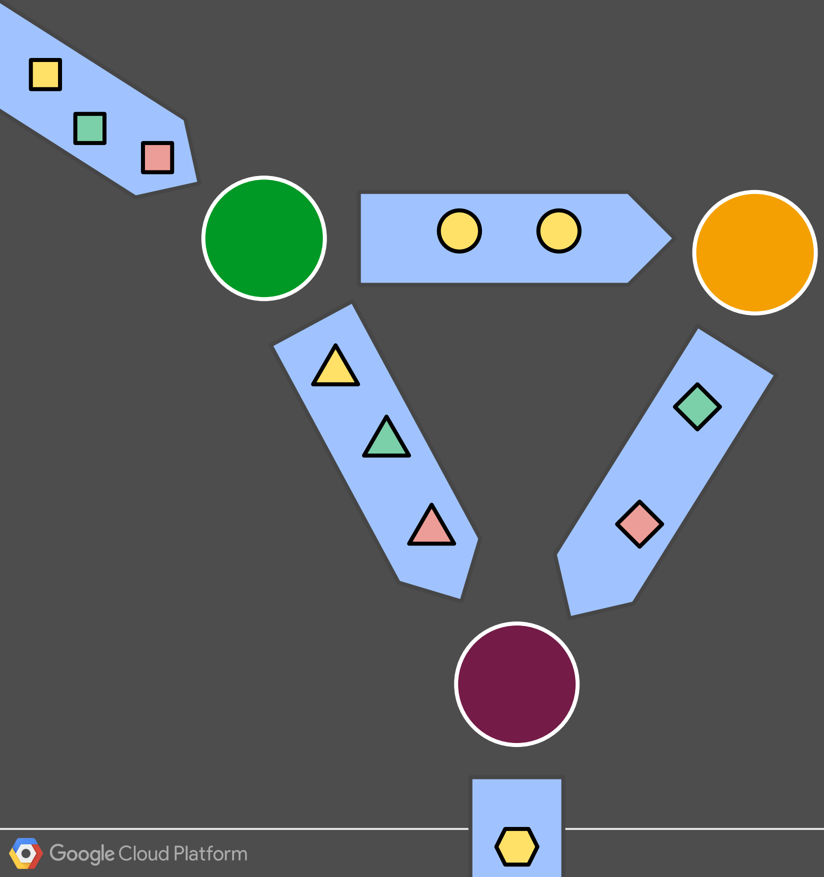
State of the art until recently: Lambda Architecture



Data Processing @ Google



MillWheel: Deterministic, low-latency streaming



- Framework for building low-latency data-processing applications
- User provides a DAG of computations to be performed
- System manages state and persistent flow of elements

Streaming or Batch?

1+1=2

Correctness

Latency

Why not both?

What are you computing?

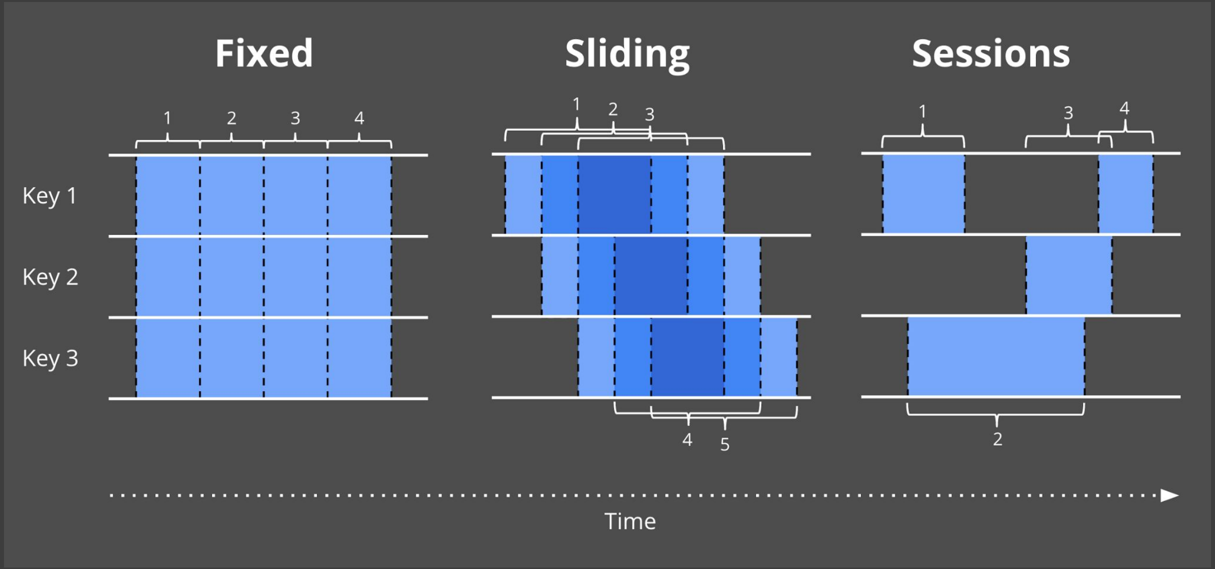
Where in event time?

When in processing time?

How do refinements relate?

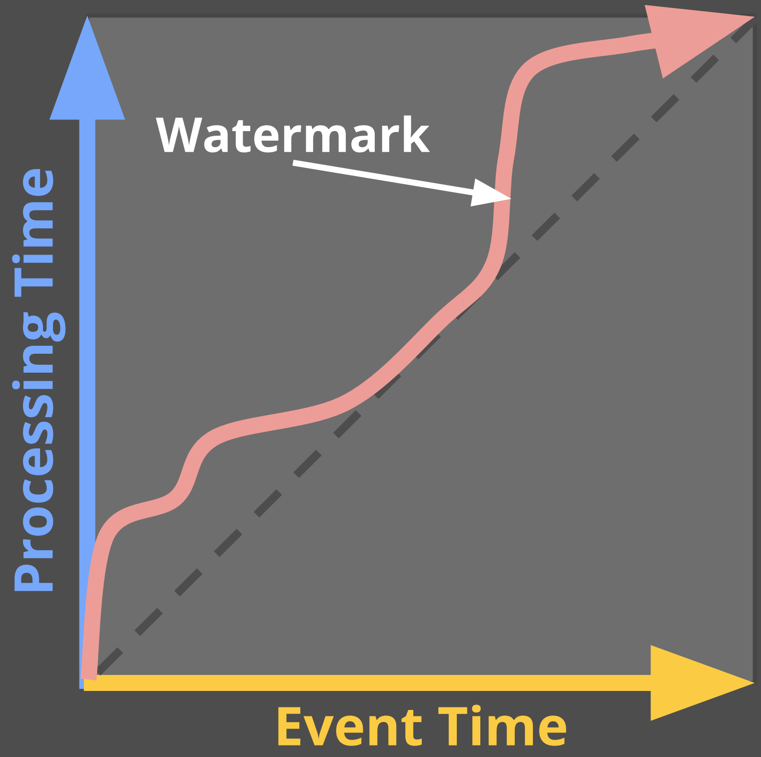
Where in event time?

- Windowing divides data into event-time-based finite chunks.



- Required when doing aggregations over unbounded data.

When in Processing Time?

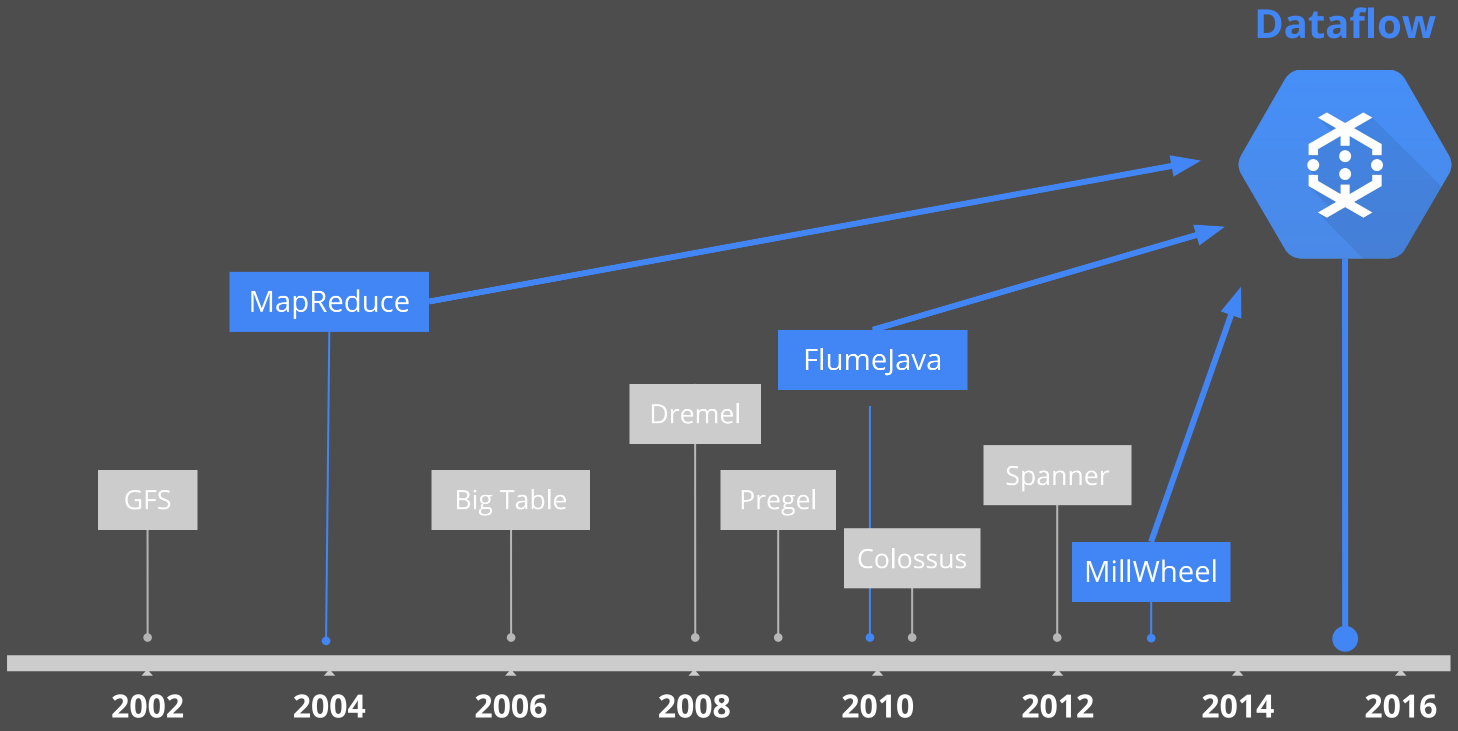


- Triggers control when results are emitted.
- Triggers are often relative to the watermark.

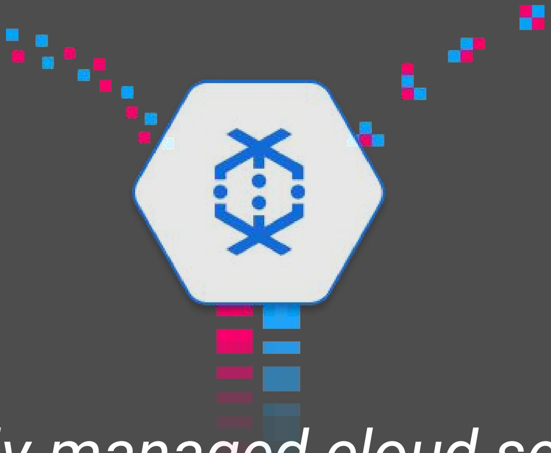
How do refinements relate?

```
PCollection<KV<String, Integer>> output = input
  .apply(Window.into(Sessions.withGapDuration(Minutes(1)))
               .trigger(AtWatermark()
                 .withEarlyFirings(AtPeriod(Minutes(1)))
                 .withLateFirings(AtCount(1)))
               .accumulatingAndRetracting())
  .apply(new Sum());
```

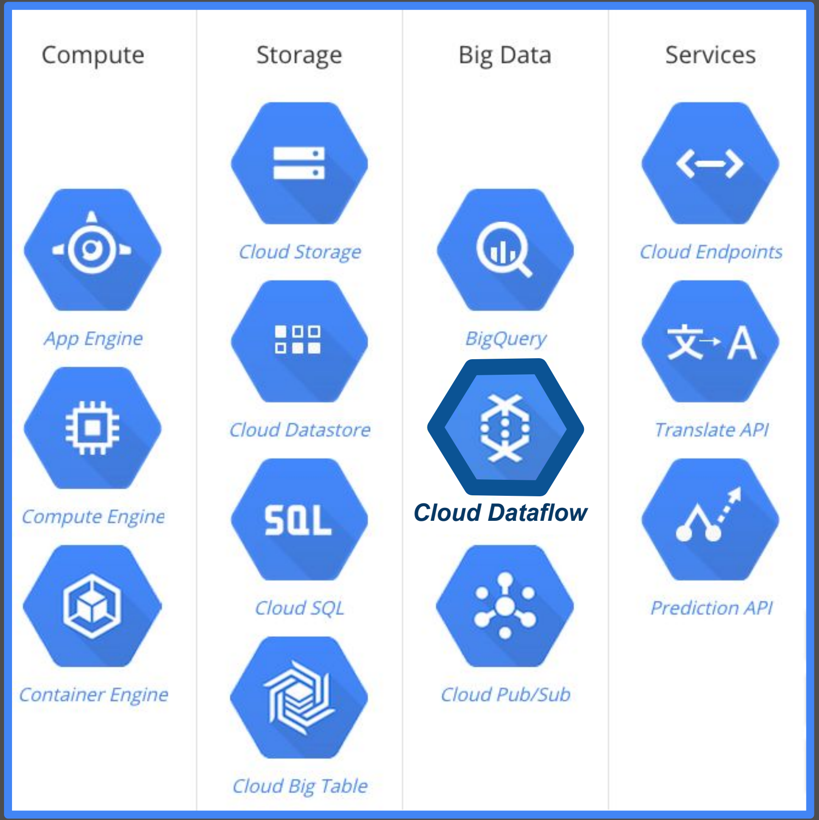
Data Processing @ Google

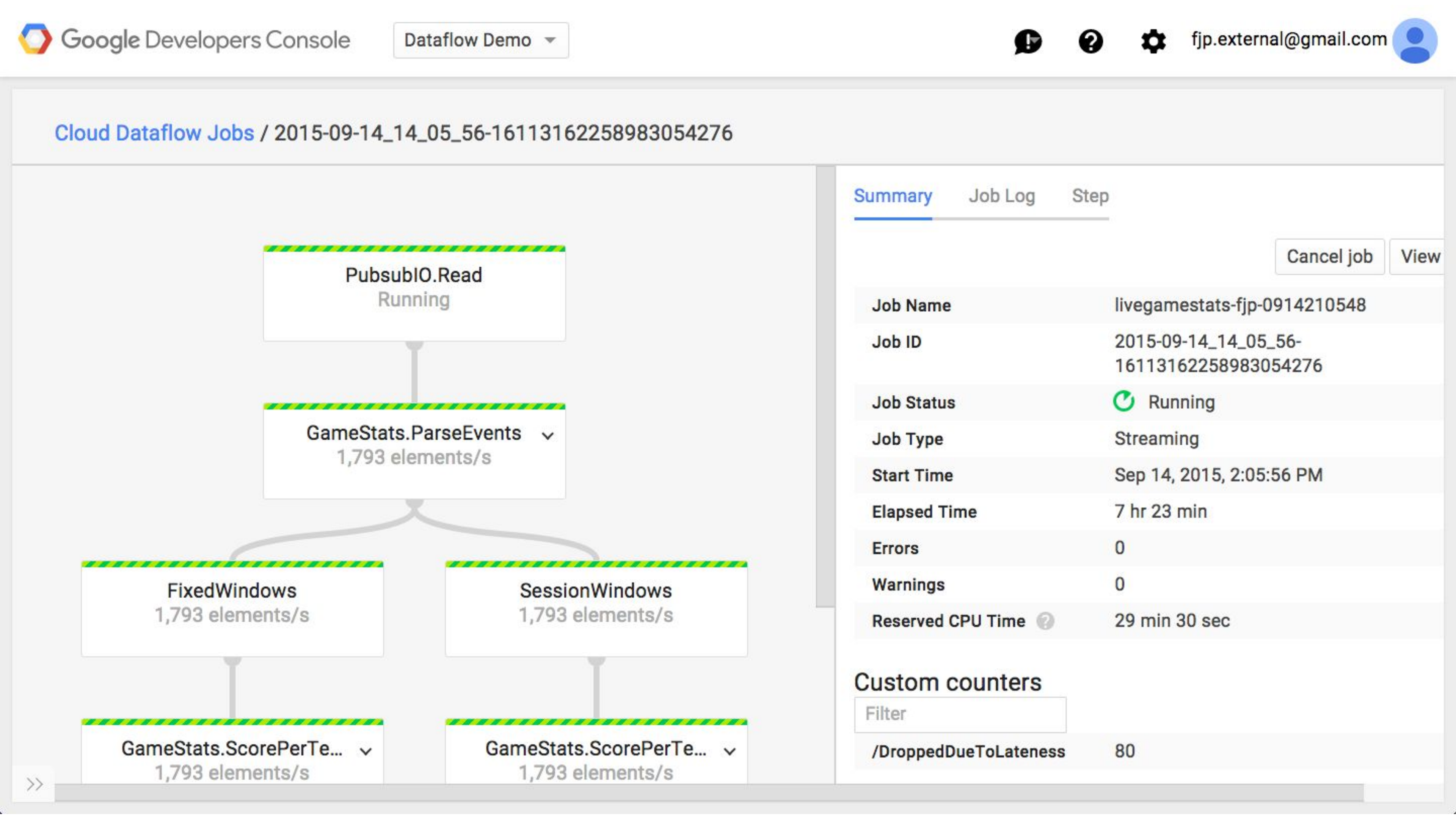


Google Cloud Dataflow



A fully-managed cloud service and programming model for batch and streaming big data processing.





Cloud Dataflow Jobs / 2015-09-14_14_05_56-16113162258983054276

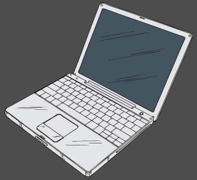
Job Name	livegamestats-fjp-0914210548
Job ID	2015-09-14_14_05_56-16113162258983054276
Job Status	Running
Job Type	Streaming
Start Time	Sep 14, 2015, 2:05:56 PM
Elapsed Time	7 hr 23 min
Errors	0
Warnings	0
Reserved CPU Time	29 min 30 sec

Custom counters

/DroppedDueToLateness	80

Dataflow SDK

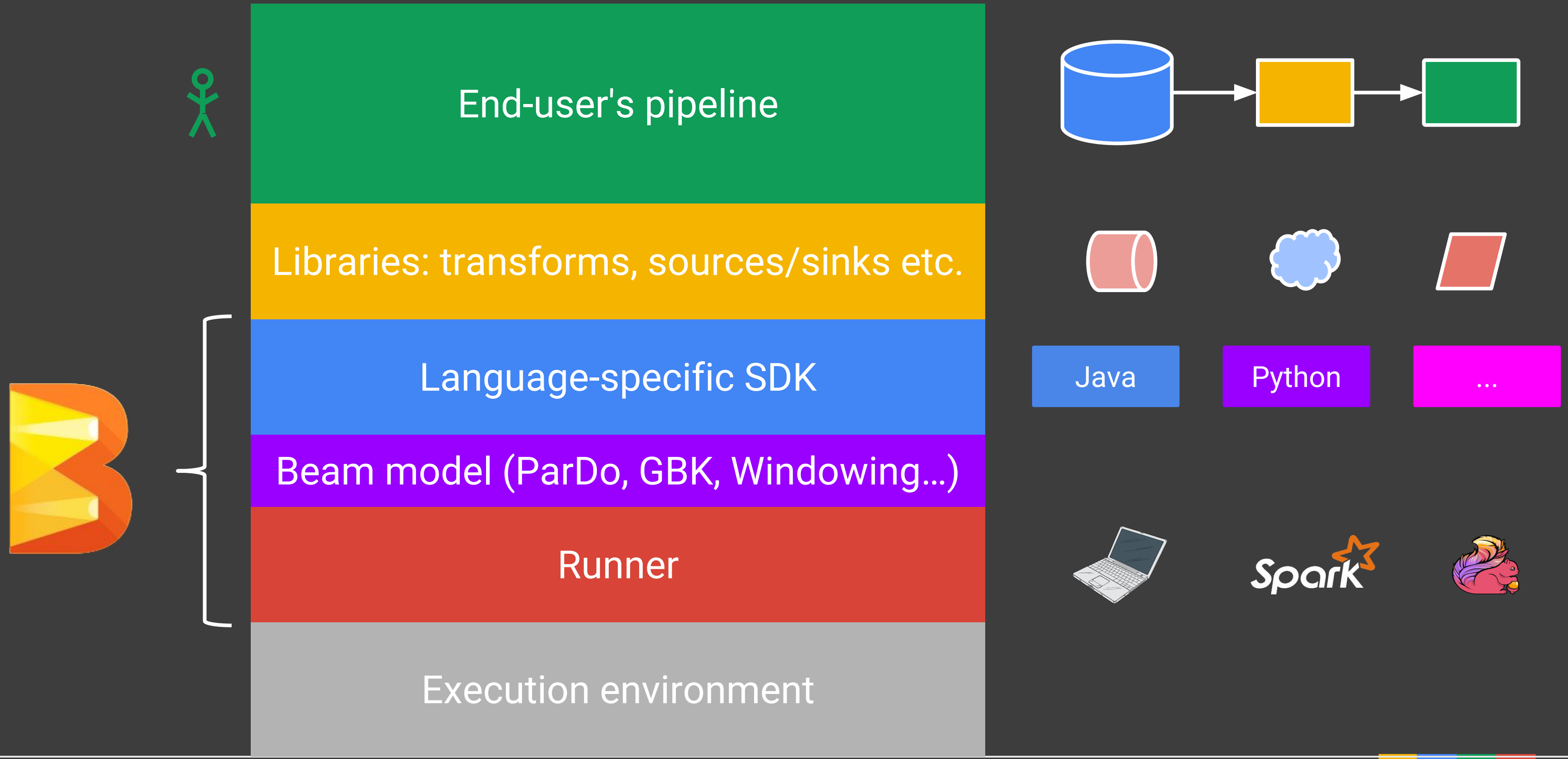
- Portable API to construct and run a pipeline.
- Available in Java and Python (alpha)
- Pipelines can run…
  - On your development machine
  - On the **Dataflow Service** on **Google Cloud Platform**
  - On third party environments like Spark or Flink.



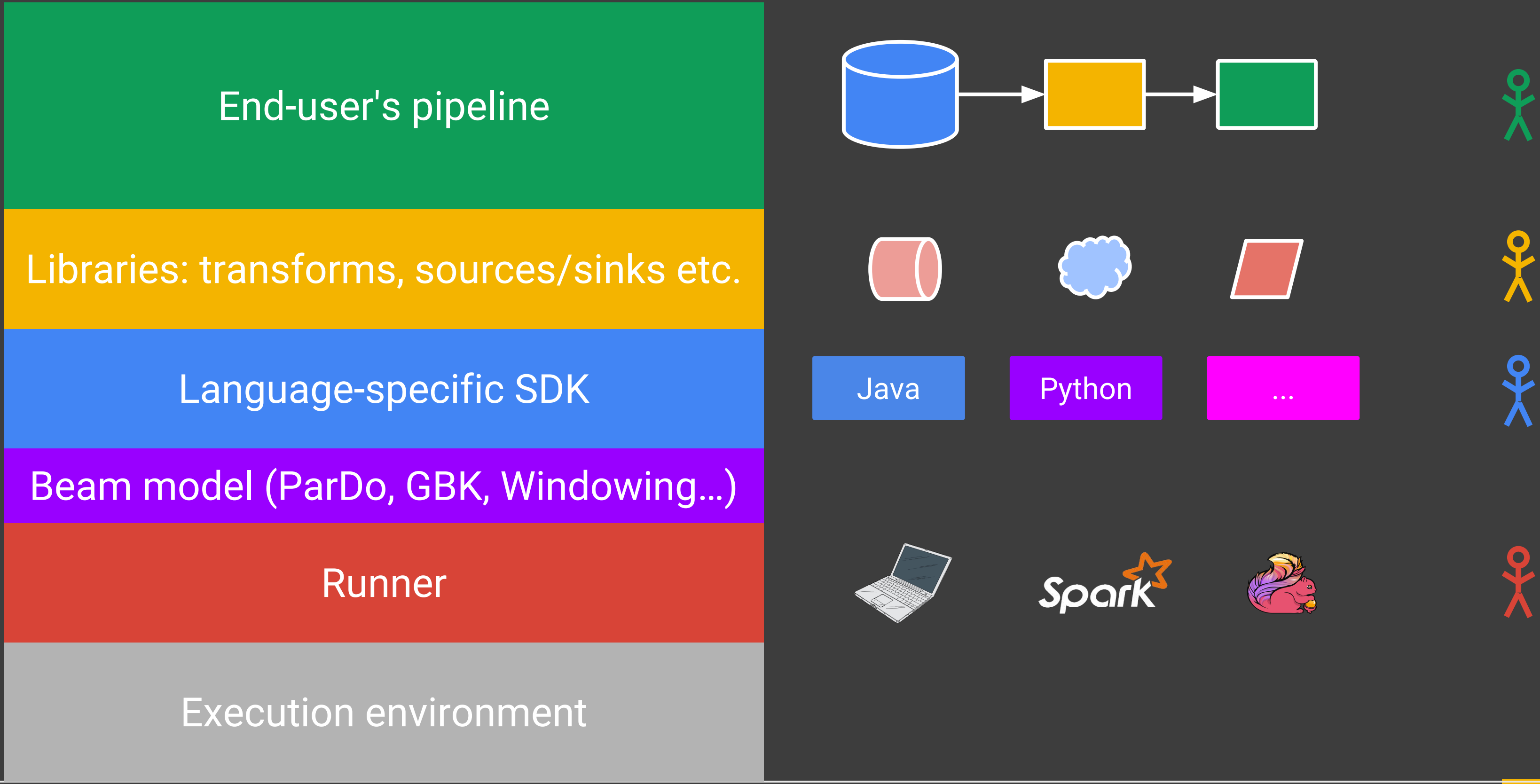
3 Dataflow ⇒ Apache Beam

```
Pipeline p = Pipeline.create(options);
p.apply(TextIO.Read.from("gs://dataflow-samples/shakespeare/*"))
 .apply(FlatMapElements.via(
     word → Arrays.asList(word.split("[^a-zA-Z']+"))))
 .apply(Filter.byPredicate(word → !word.isEmpty()))
 .apply(Count.perElement())
 .apply(MapElements.via(
     count → count.getKey() + ": " + count.getValue())
 .apply(TextIO.Write.to("gs://.../..."));
p.run();
```

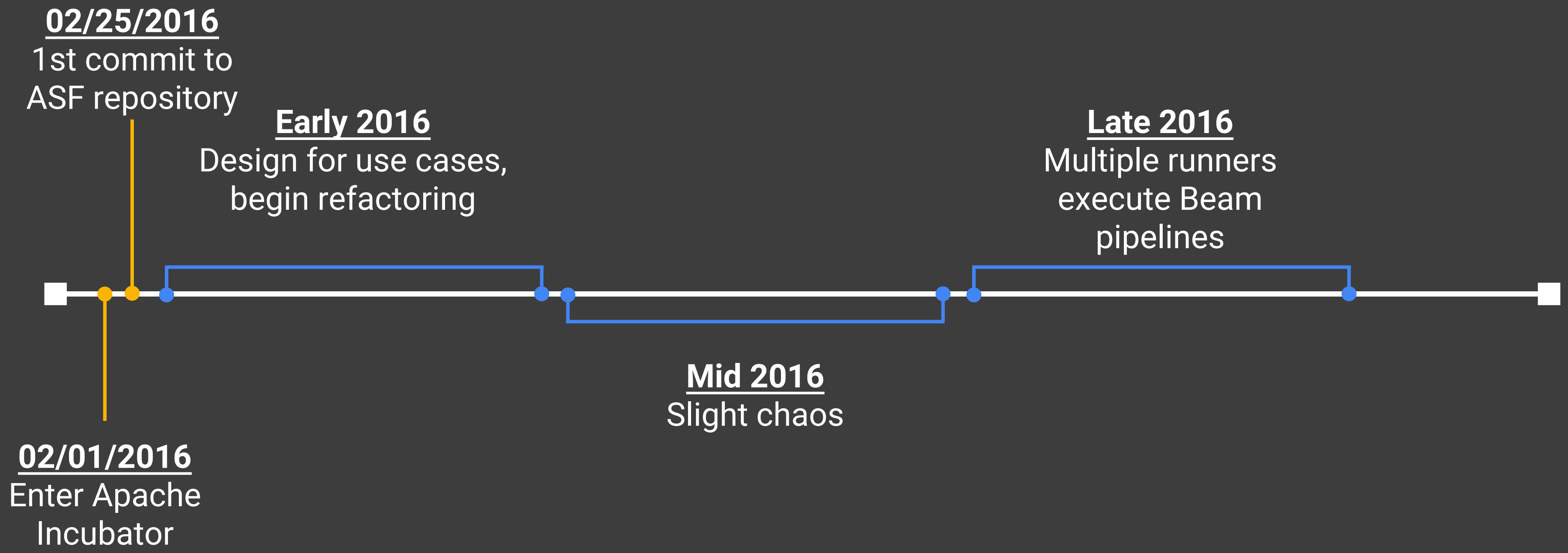
Apache Beam ecosystem



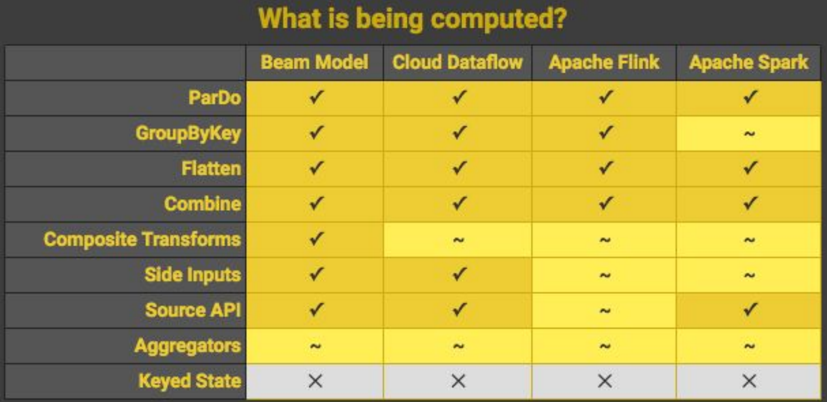
Apache Beam ecosystem



Apache Beam Roadmap

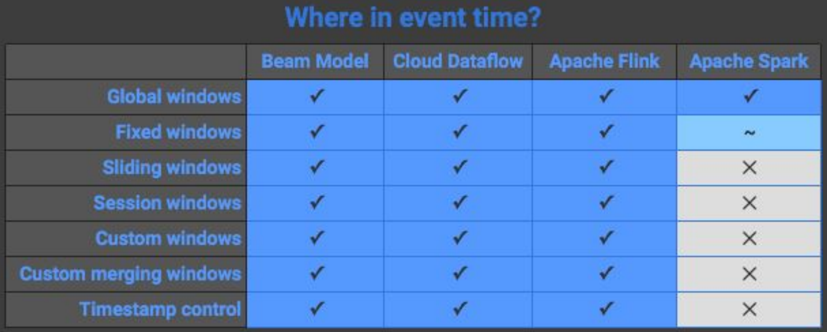

02/25/2016
1st commit to ASF repository
Early 2016
Design for use cases, begin refactoring
Late 2016
Multiple runners execute Beam pipelines
Mid 2016
Slight chaos
02/01/2016
Enter Apache Incubator

Runner capability matrix



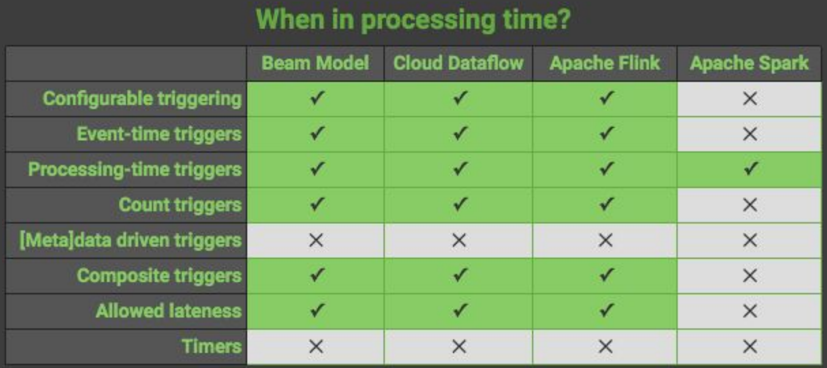
What is being computed?

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



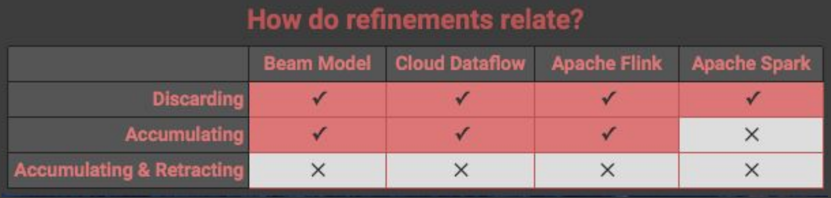
Where in event time?

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



When in processing time?

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

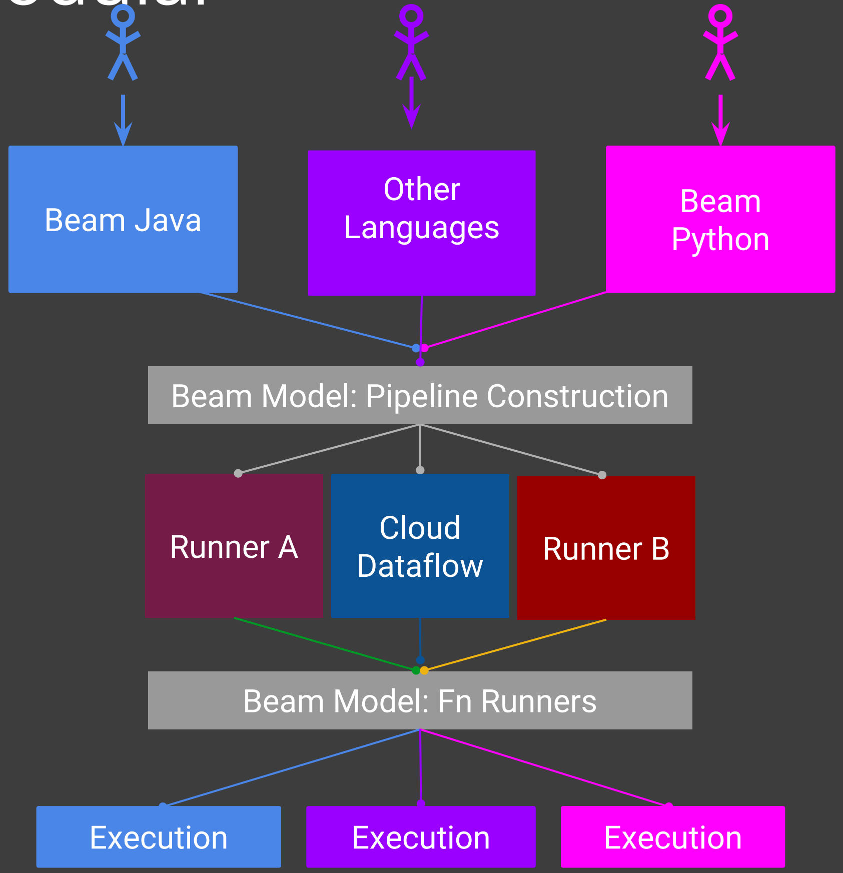


How do refinements relate?

	Beam Model	Cloud Dataflow	Apache Flink	Apache Spark
Discarding	✓	✓	✓	✓
Accumulating	✓	✓	✓	✕
Accumulating & Retracting	✕	✕	✕	✕

Technical Vision: Still more modular

- **Multiple SDKs**
  with shared pipeline representation
- **Language-agnostic runners**
  implementing the model
- **Fn Runners**
  run language-specific code



Recap: Timeline of ideas

2004	**MapReduce** (SELECT / GROUP BY) Library > DSL Abstract away fault tolerance & distribution
2010	**FlumeJava:** High-level API (typed DAG)
2013	**MillWheel:** Deterministic stream processing
2015	**Dataflow:** Unified batch/streaming model Windowing, Triggers, Retractions
2016	**Beam:** Portable programming model Language-agnostic runners

Learn More!

Programming model

The World Beyond Batch: Streaming 101, Streaming 102

The Dataflow Model paper

Cloud Dataflow

http://cloud.google.com/dataflow/

Apache Beam

https://wiki.apache.org/incubator/BeamProposal

http://beam.incubator.apache.org/

Dataflow/Beam vs. Spark

Thank you

Google Cloud Platform