APACHE: BIG_DATA EUROPE

14-16.11.16

SEVILLE, SPAIN

Introduction to Apache Beam

Dan Halperin
Google
Beam podling PMC

JB Onofré
Talend
Beam Champion & PMC
Apache Member

The Apache Incubator Project
http://incubator.apache.org/

THE LINUX FOUNDATION

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

What is Apache Beam?

The Beam **Programming Model**

SDKs for writing Beam pipelines

- Java, Python

Beam **Runners** for existing distributed processing backends

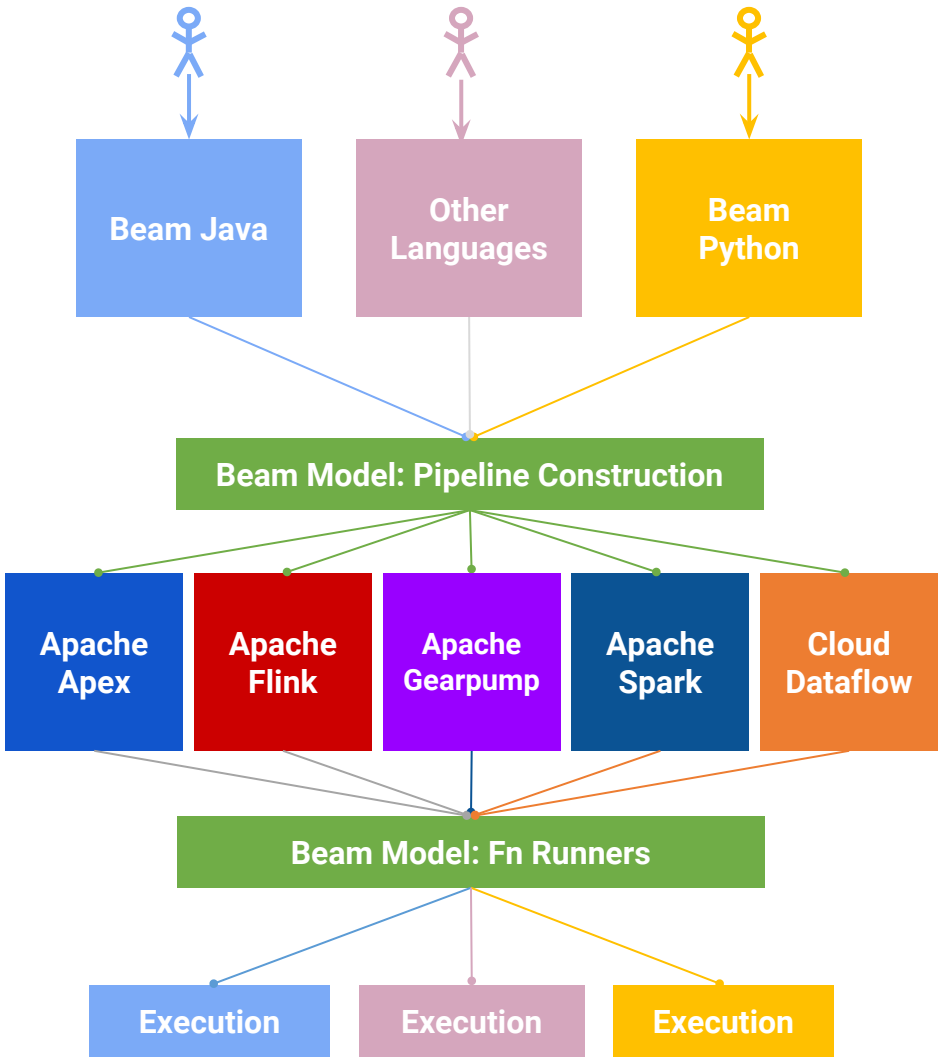
Apache Apex

Apache Flink

Apache Gearpump

Apache Spark

Google Cloud Dataflow



What's in this talk

- **Introduction to Apache Beam**
- The Apache Beam Podling
- Beam Demos

Quick overview of the Beam model

PCollection – a parallel collection of **timestamped elements** that are **in windows**.

Sources & **Readers** – produce PCollections of timestamped elements and a **watermark**.

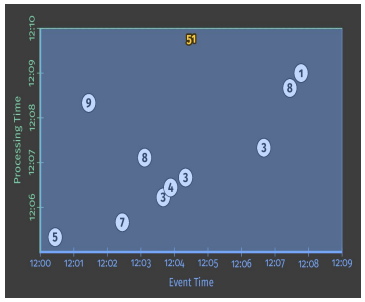
ParDo – flatmap over elements of a PCollection.

(Co)**GroupByKey** – shuffle & group `{{K: V}}` → `{K: [V]}`.

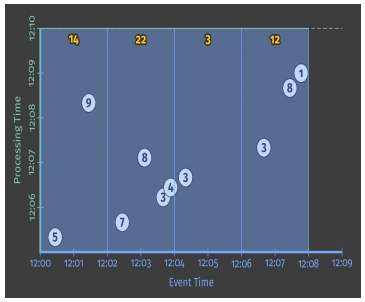
Side inputs – **global view** of a PCollection used for broadcast / joins.

Window – reassign elements to zero or more windows; may be **data-dependent**.

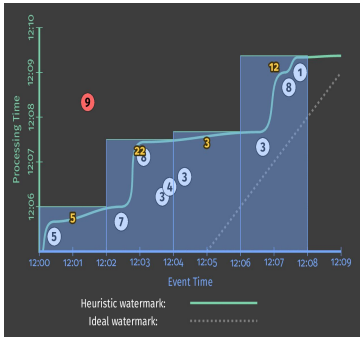
Triggers – user flow control based on **window**, **watermark**, **element count**, **lateness** - emitting zero or more *panes* per window.



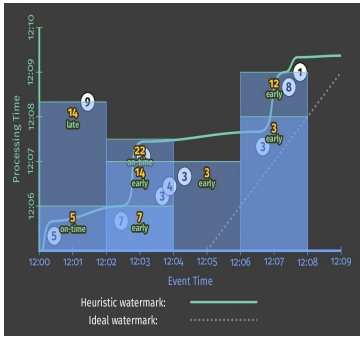
1.Classic Batch



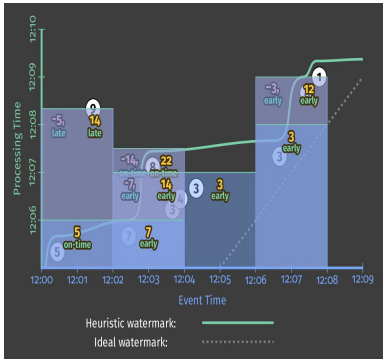
2. Batch with Fixed Windows



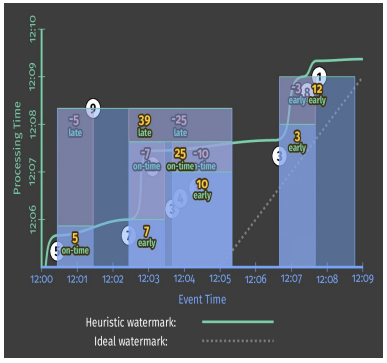
3. Streaming



4. Streaming with Speculative + Late Data



5. Streaming With Retractions



6. Sessions

Simple clickstream analysis pipeline

Data: JSON-encoded analytics stream from site

- {“user”:“dhalperi”,
  “page”:“blog.apache.org/feed/7”,
  “tstamp”:”2016-11-16T15:07Z”, ...}

Desired output: Per-user session length and activity level

- dhalperi, 17 pageviews, 2016-11-16 15:00-15:35

Other application-dependent user goals:

- **Live data** – can track ongoing sessions with speculative output
  dhalperi, 10 pageviews, 2016-11-16 15:00-15:15 (EARLY)
- **Archival data** – much faster, still correct output respecting event time

Simple clickstream analysis pipeline

```
PCollection<KV<User, Click>> clickstream =
    pipeline.apply(IO.Read(…))
            .apply(MapElements.of(new ParseClicksAndAssignUser()));

PCollection<KV<User, Long>> userSessions =
    clickstream.apply(Window.into(Sessions.withGapDuration(Minutes(3)))
                        .triggering(
                            AtWatermark()
                                .withEarlyFirings(AtPeriod(Minutes(1)))))
               .apply(Count.perKey());

userSessions.apply(MapElements.of(new FormatSessionsForOutput()))
            .apply(IO.Write(…));

pipeline.run();
```

Unified unbounded & bounded PCollections

```
pipeline.apply(IO.Read(…)).apply(MapElements.of(new ParseClicksAndAssignUser()));
```

Apache Kafka, Apache ActiveMQ, tailing filesystem...

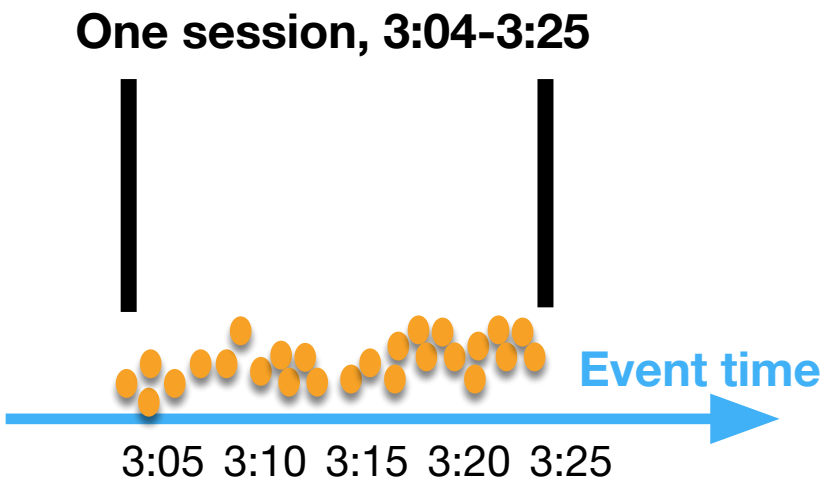
- A live, roughly in-order stream of messages, *unbounded PCollections.*
- **KafkaIO.read().fromTopic(“pageviews”)**

HDFS, Apache HBase, yesterday’s Apache Kafka log…

- Archival data, often readable in any order, *bounded PCollection*s.
- **TextIO.read().from(“hdfs://facebook/pageviews/*”)**

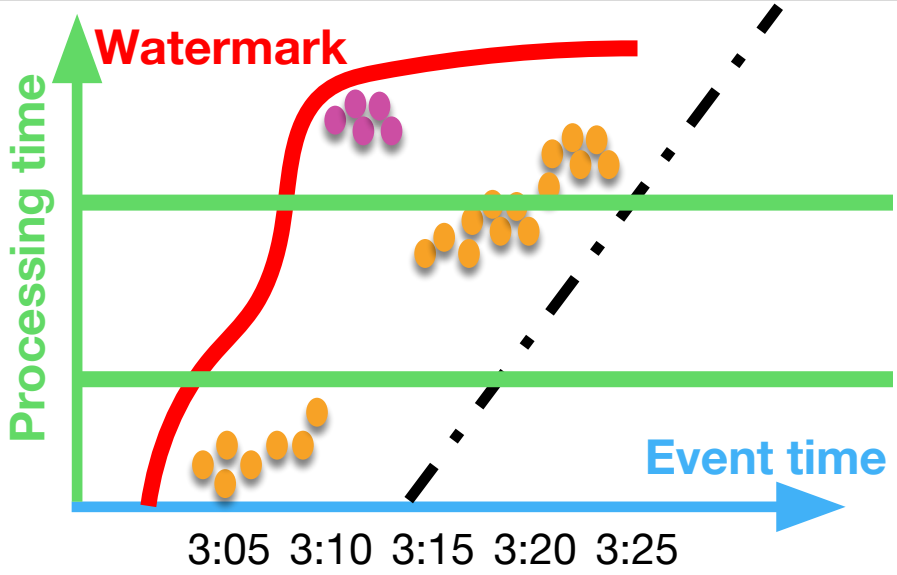
Windowing and triggers

```
PCollection<KV<User, Long>> userSessions =
    clickstream.apply(Window.into(Sessions.withGapDuration(Minutes(3)))
                      .triggering(
                          AtWatermark()
                              .withEarlyFirings(AtPeriod(Minutes(1)))))
```



Windowing and triggers

```
PCollection<KV<User, Long>> userSessions =
    clickstream.apply(Window.into(Sessions.withGapDuration(Minutes(3)))
                      .triggering(
                          AtWatermark()
                              .withEarlyFirings(AtPeriod(Minutes(1)))))
```



1 session,
3:04–3:25

(*EARLY*) 2 sessions, **3:04–3:10** &
3:15–3:20

(*EARLY*) 1 session,
3:04–3:10

Two example runs of this pipeline

Streaming job consuming
Apache Kafka stream

- Uses **10 workers**.
- Pipeline **lag of a few minutes**.
- With ~2 million users over 1 day.

- Total ~4.7M messages (early + final sessions) to downstream.
- 240 worker-hours

Daily batch job consuming
Apache Hadoop HDFS archive

- Uses **200 workers**.
- Runs for **30 minutes**.
- Same input.

- Total ~2.1M final sessions.
- 100 worker-hours

What does the user have to change to get these results?

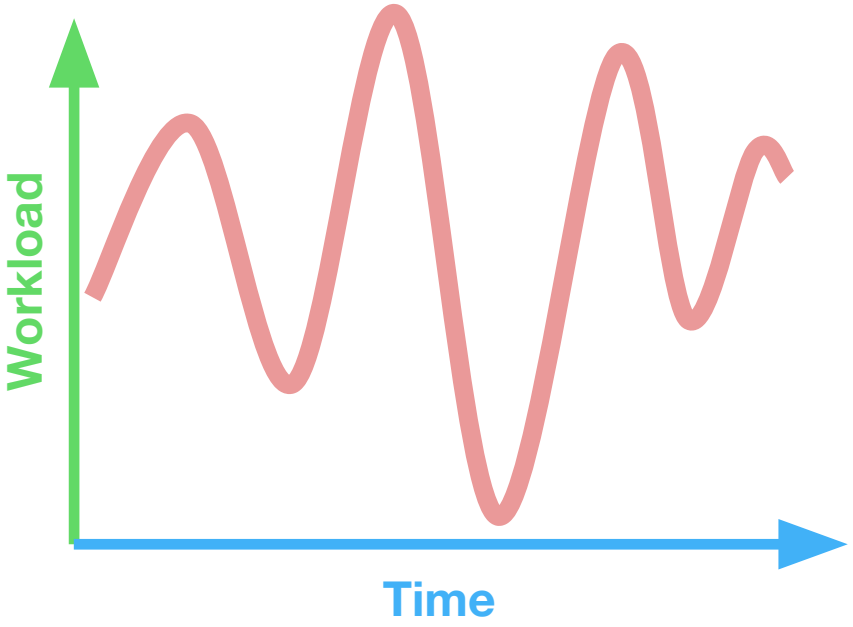
A: O(10 lines of code) + Command-line arguments

Summary so far

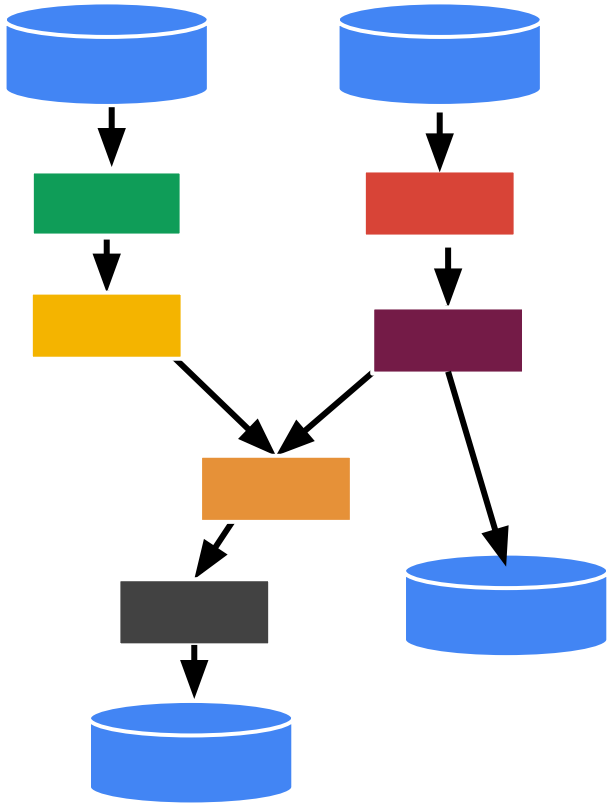
- Introduced Beam
- Quick overview of unified programming model
- Sample clickstream analysis pipeline
- Portability across both IOs and runners

Next: Quick dip into efficiency

Pipeline workload varies

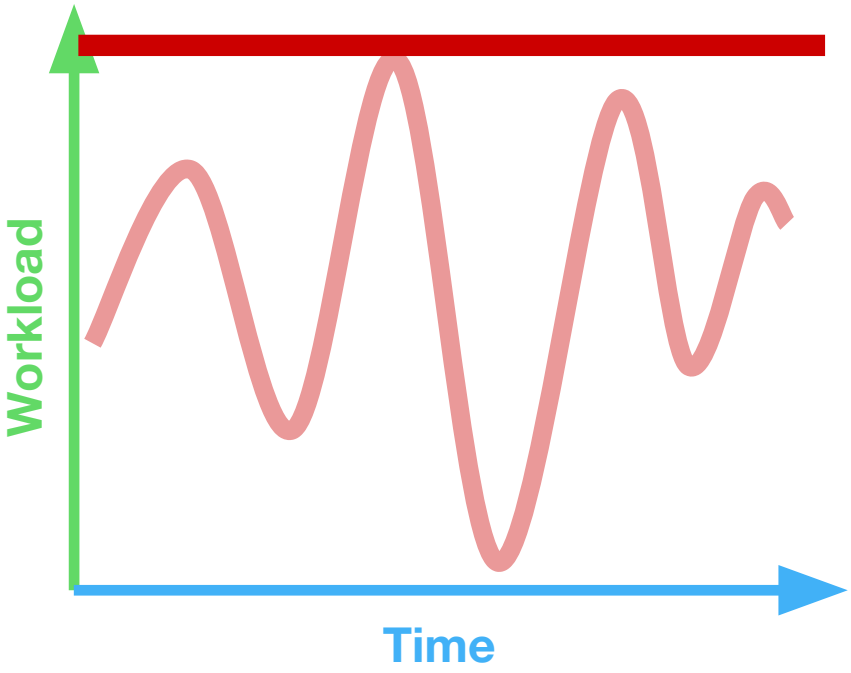


Streaming pipeline's input varies

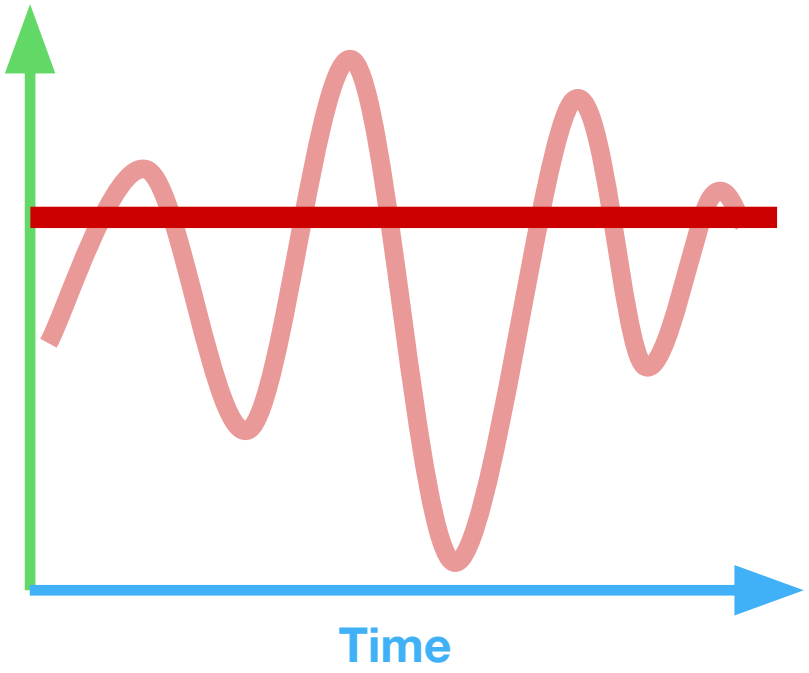


Batch pipelines go through stages

Perils of fixed decisions

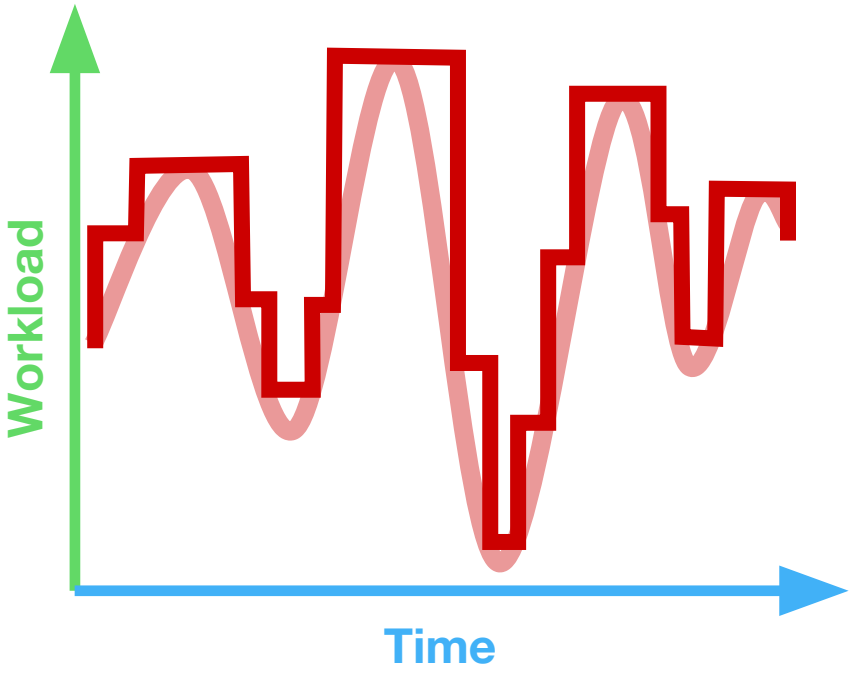


Over-provisioned / worst case

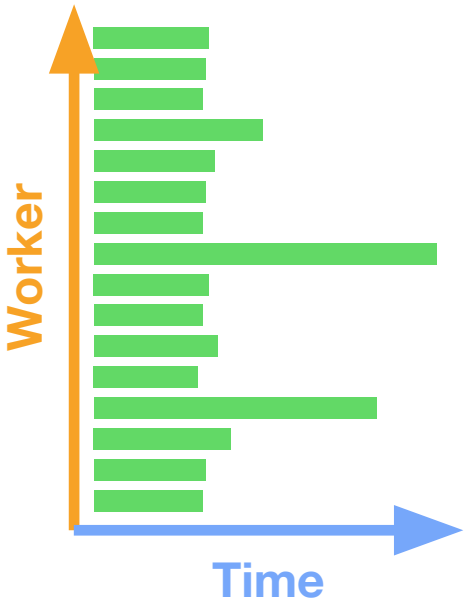


Under-provisioned / average case

Ideal case



The Straggler problem



Work is unevenly distributed across tasks.

Reasons:

- Underlying data.
- Processing.
- Runtime effects.

Effects are cumulative per stage.

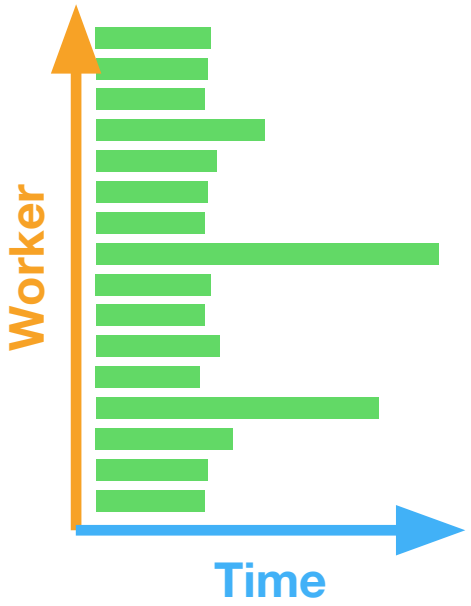
Standard workarounds for Stragglers

Split files into equal sizes?

Preemptively over-split?

Detect slow workers and re-execute?

Sample extensively and then split?



All of these have major costs; none is a complete solution.

No amount of upfront heuristic tuning (be it manual or automatic) **is enough** to guarantee good performance: the **system will always hit unpredictable situations** at run-time.

A system that's able to **dynamically adapt and get out of a bad situation** is much more powerful than one that **heuristically hopes to avoid** getting into it.

Beam Readers enable dynamic adaptation

Readers provide simple progress signals, enable runners to take action based on execution-time characteristics.

APIs for how much work is pending:

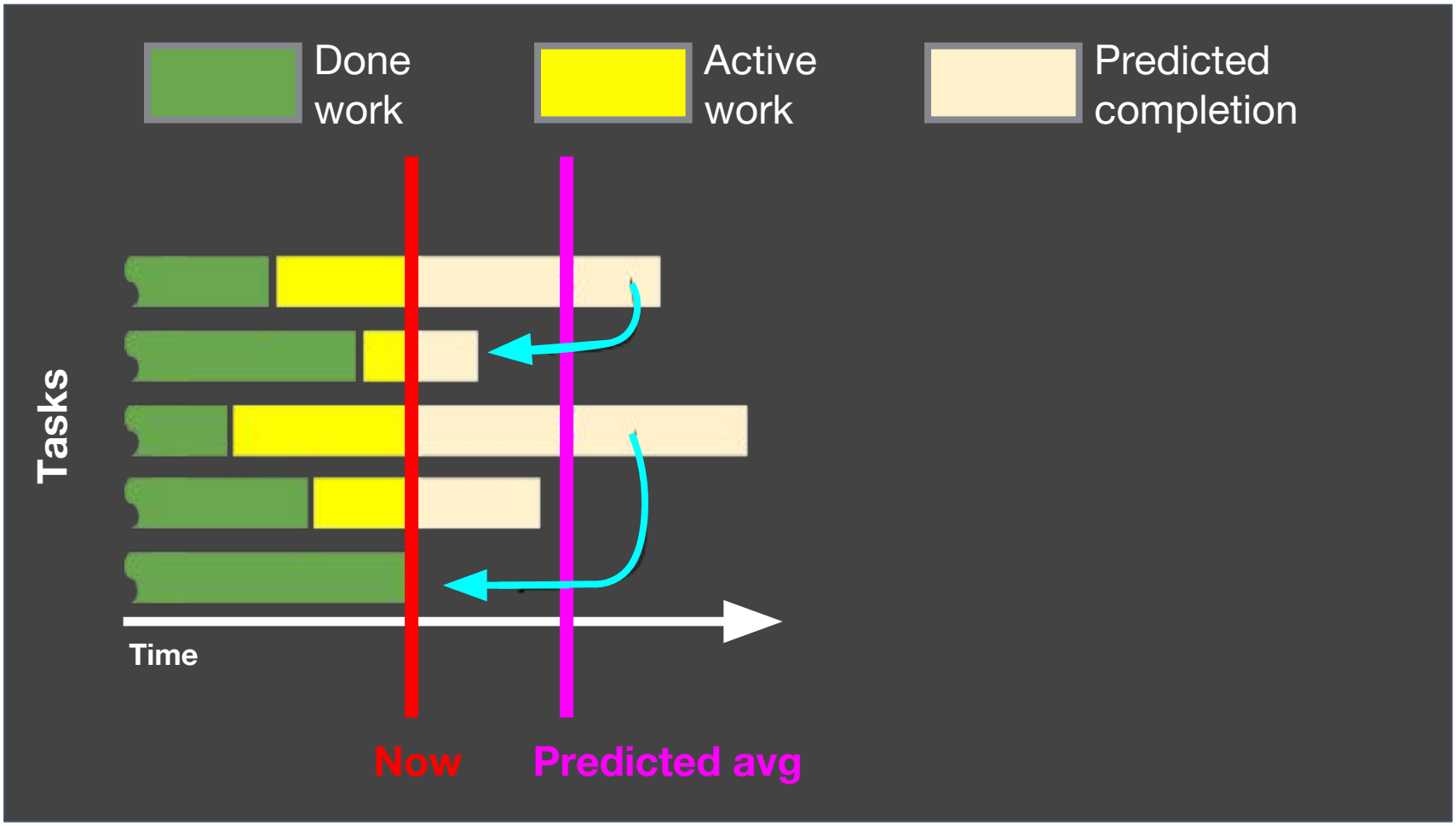
Bounded: double **getFractionConsumed**()

Unbounded: long **getBacklogBytes**()

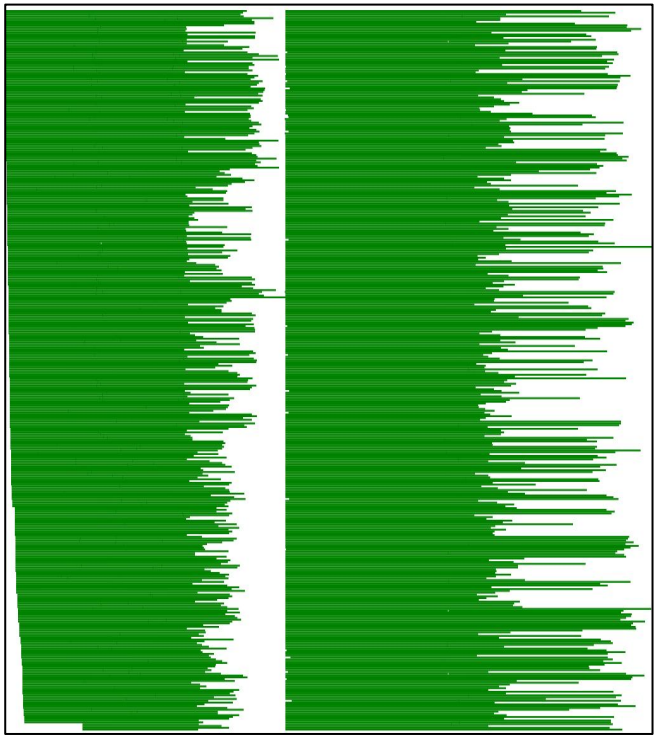
Work-stealing:

Bounded: Source **splitAtFraction**(double)
int **getParallelismRemaining**()

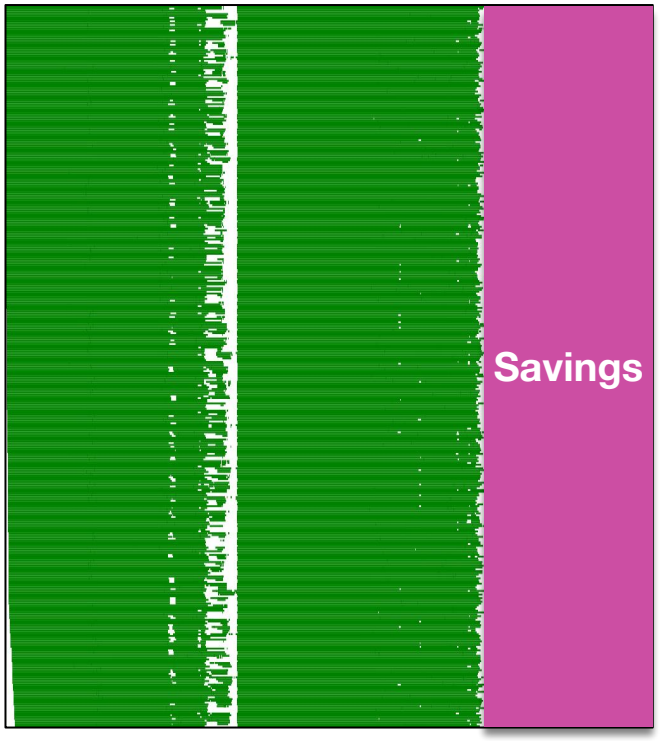
Dynamic work rebalancing



Dynamic work rebalancing: a real example



2-stage pipeline,
split “evenly” but uneven in practice



Same pipeline
dynamic work rebalancing enabled

What's in this talk

- Introduction to Apache Beam
- **The Apache Beam Podling**
- Beam Demos

The Evolution of Apache Beam

Google

PDF Colossus

PDF Bigtable

PDF PubSub

PDF Dremel

Google Cloud Dataflow

PDF Spanner

PDF Megastore

PDF Flume

PDF Millwheel

PDF MapReduce

Apache Beam

APACHE HBASE

CRUNCH

Spark

OOZIE

hadoop

APACHE DRILL

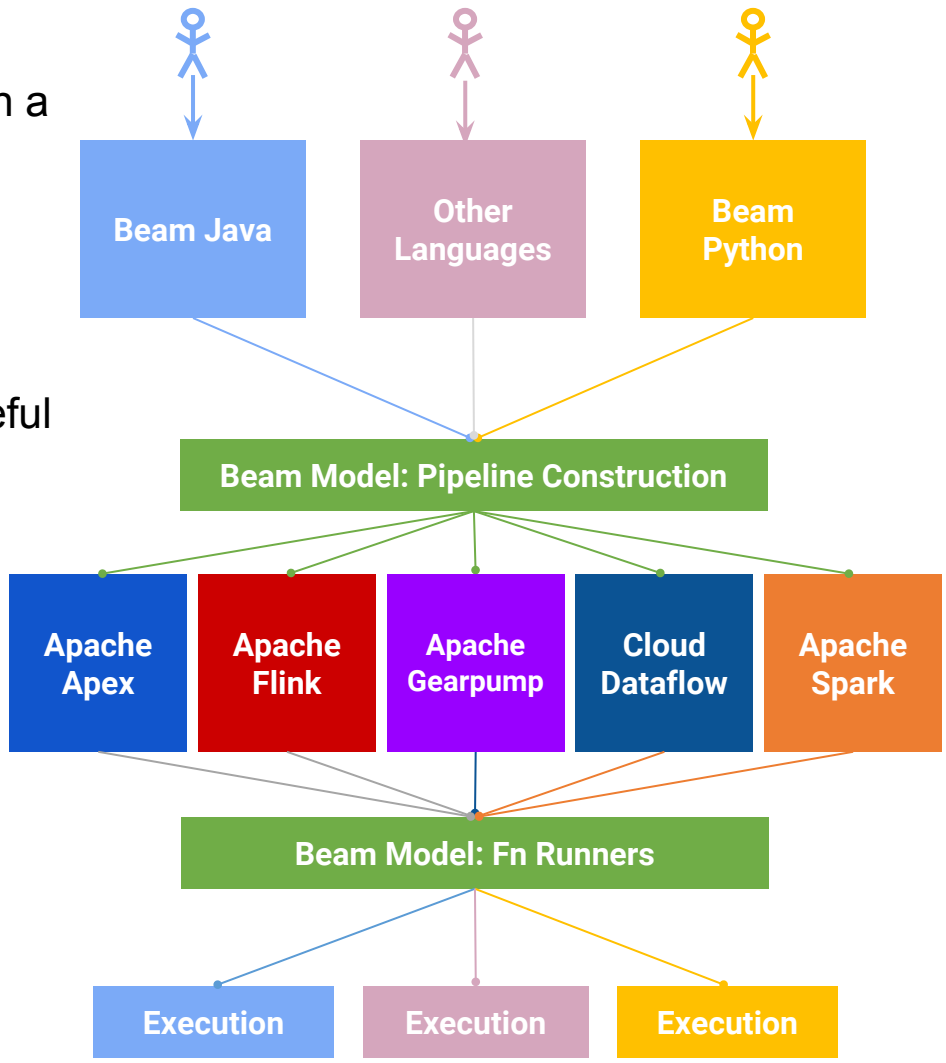
HIVE

TEZ

THE APACHE SOFTWARE FOUNDATION

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. **Library writers**: who want to provide useful composite transformations.
4. **Runner writers:** who have a distributed processing environment and want to support Beam pipelines.
5. **IO providers**: who want efficient interoperation with Beam pipelines on all runners.
6. **DSL writers**: who want higher-level interfaces to create pipelines.



February 2016: Beam enters incubation

Code donations from:

- Core Java SDK and Dataflow runner (Google)
- Apache Flink runner (data Artisans)
- Apache Spark runner (Cloudera)

Initial podling PMC

- Cloudera (2)
- data Artisans (4)
- Google (10)
- PayPal (1)
- Talend (1)

First few months: Bootstrapping

Refactoring & De-Google-ification

Contribution Guide

- Getting started
- Process: how to contribute, how to review, how to merge
- Populate JIRA with old issues, curate “starter” issues, etc.
- Strong commitment to testing

Experienced committers providing extensive, public code review (onboarding)

- No merges without a GitHub pull request & LGTM

Search Results

< 1 > displaying 1 to 2 of 2

GroupId	ArtifactId	Version	Updated	Download
com.google.cloud.dataflow	google-cloud-dataflow-java-examples-all	1.8.0	03-Oct-2016	pom jar javadoc.jar sources.jar
com.google.cloud.dataflow	google-cloud-dataflow-java-sdk-all	1.8.0	03-Oct-2016	pom jar javadoc.jar sources.jar

org.apache.beam	beam-examples-java8	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-maven-archetypes-examples	0.1.0-incubating	08-Jun-2016	pom jar sources.jar test-sources.jar
org.apache.beam	beam-sdks-java-maven-archetypes-starter	0.1.0-incubating	08-Jun-2016	pom jar sources.jar test-sources.jar
org.apache.beam	beam-runners-spark	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar spark-app.jar test-sources.jar tests.jar
org.apache.beam	beam-runners-flink_2.10-examples	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-runners-flink_2.10	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-examples-java	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-runners-direct-java	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-runners-google-cloud-dataflow-java	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-runners-core-java	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-java8tests	0.1.0-incubating	08-Jun-2016	pom jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-extensions-join-library	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-io-kafka	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-io-hdfs	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-io-google-cloud-platform	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-core	0.1.0-incubating	08-Jun-2016	pom jar javadoc.jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-sdks-java-build-tools	0.1.0-incubating	08-Jun-2016	pom jar sources.jar test-sources.jar tests.jar
org.apache.beam	beam-parent	0.1.0-incubating	08-Jun-2016	pom source-release.zip

Since June Release

- Community contributions
  - New SDK: Python (feature branch)
  - New IOs (Apache ActiveMQ, JDBC, MongoDB, Amazon Kinesis, …)
  - New libraries of extensions
  - Two new runners: Apache Apex & Apache Gearpump
- Added three new committers
  - tgroh (core, Google), tweise (Apex, DataTorrent), jesseanderson (Smoking Hand, Evangelism & Users)
- Documented release process & executed two more releases
  3 releases, 3 committers, 2 organizations
- >10 conference talks and meetups by at least 3 organizations

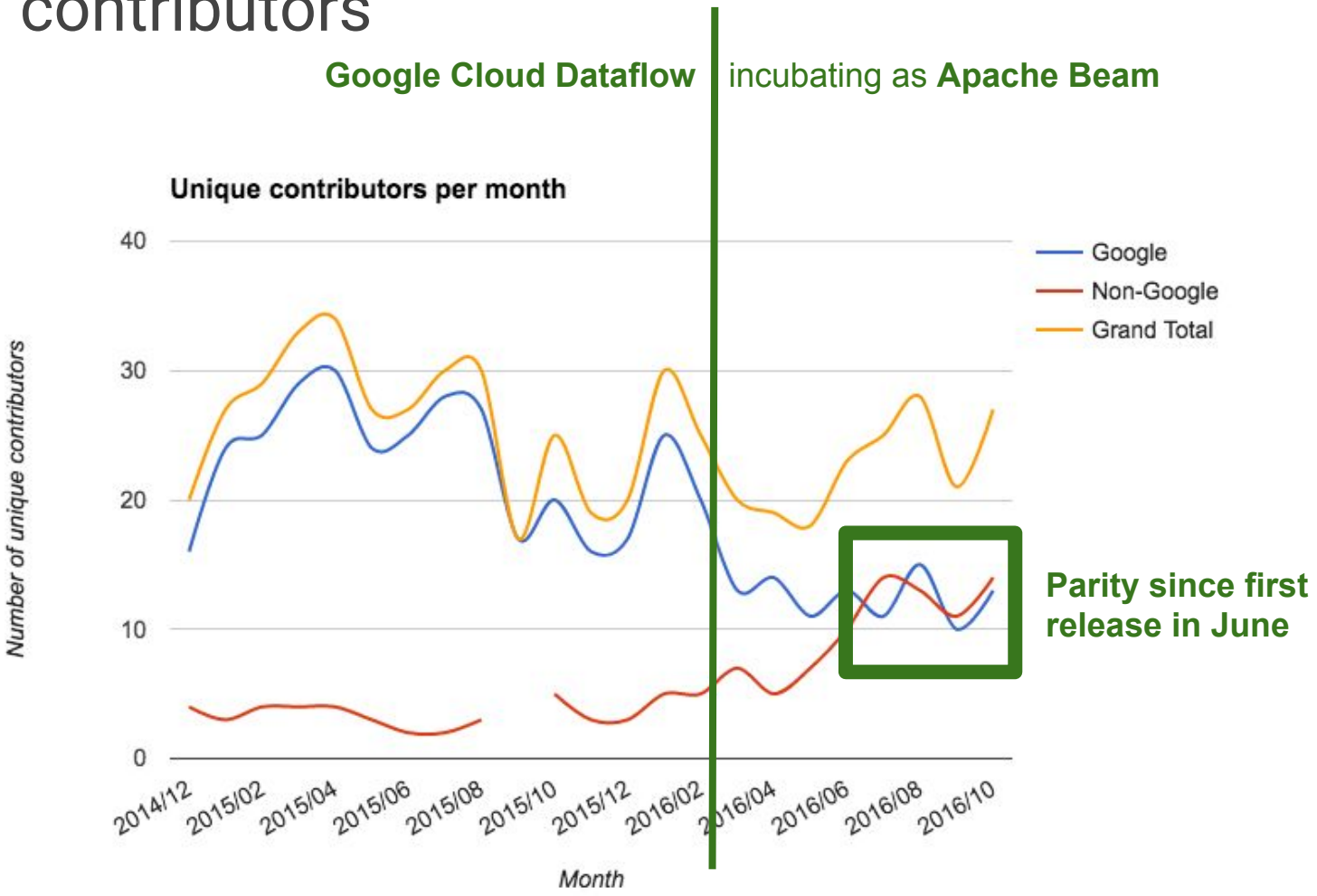
Beam is community owned

- Growing community
  - more than 1500 messages on mailing lists
  - 500 mailing lists subscribers
  - 4000 commits
  - 950 Jira

- 1350 pull requests - **2nd most in Apache** since incubation

Beam contributors

Google Cloud Dataflow | incubating as **Apache Beam**



Parity since first release in June

What's in this talk

- Introduction to Apache Beam
- The Apache Beam Podling
- **Beam Demos**

Demo

Goal: show WordCount on 5 runners

- Beam's Direct Runner (testing, model enforcement, playground)
- Apache Apex (newest runner!)
- Apache Flink
- Apache Spark
- Google Cloud Dataflow

(DEMO)

Conclusion: Why Beam for Apache?

1. **Correct** - Event windowing, triggering, watermarking, lateness, etc.
2. **Portable -** Users can use the same code with different runners (agnostic) and backends on premise, in the cloud, or locally
3. **Unified** - Same unified model for batch and stream processing
4. **Apache community enables a network effect** - Integrate with Beam and you automatically integrate with Beam's users, SDKs, runners, libraries, …

Apache Beam next steps

Graduation to TLP - Empower user adoption

New website - Improve both look'n feel and content of the website, more focused on users

Polish user experience - Improve the rough edges in submitting and managing jobs

Keep growing - Integrations planned & ongoing with new runners (Apache Storm), new DSLs (Apache Calcite, Scio), new IOs (Apache Cassandra, ElasticSearch), etc.

Learn More!

Apache Beam (incubating)
http://beam.incubator.apache.org

Beam contribution guide:
http://beam.incubator.apache.org/contribute/contribution-guide

Join the Beam mailing lists!
user-subscribe@beam.incubator.apache.org
dev-subscribe@beam.incubator.apache.org

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

Follow @ApacheBeam on Twitter