Large-scale data analysis at cloud scale

johnwilkes@google.com

2016-09

Google Cloud Platform

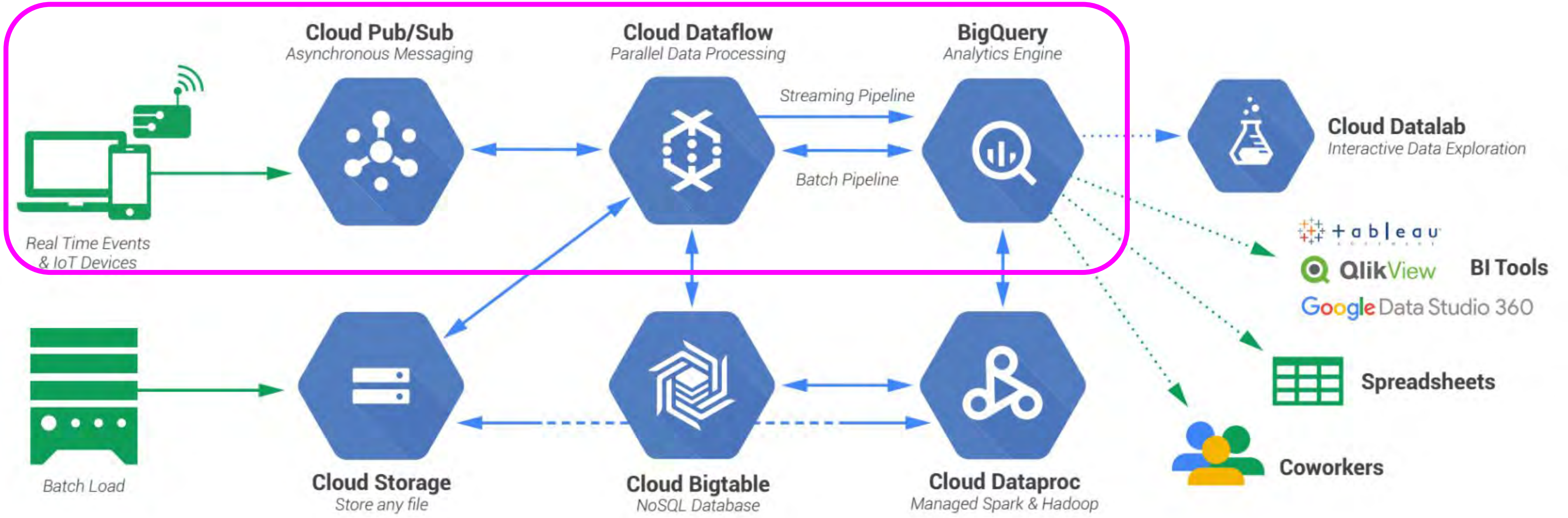
Based on slides from Eric Schmidt, Greg DeMichillie, Frances Perry, Tyler Akidau, Dan Halperin.

Data & Analytics

Application Development

Infrastructure & Operations

GCP Big Data reference architecture



One possible flow

Application logs

Impression
Start stream
Add to playlist
Share
....

Cloud Pub/Sub
Ingest and propagate

Cloud Dataflow
Processing and imperative analysis

BigQuery
Durable storage and interactive analysis

Questions you might ask
- Which application sections are receiving the most impressions?
- Who is the top artist by stream starts in the last minute?
- What is the average session length for users in Seattle?

10+ years of tackling Big Data problems

Google Papers

2002	2004	2005	2006	2008	2010	2012	2014	2015
GFS	Map Reduce	BigTable	Dremel	PubSub	Flume Java	Millwheel	Dataflow	Tensorflow

10+ years of tackling Big Data problems

2002	2004	2005	2006	2008	2010	2012	2014	2015
GFS	Map Reduce	BigTable	Dremel	PubSub	Flume Java	Millwheel	Dataflow	Tensorflow

Google Papers

The Dremel paper

In: *Proc. 36th VLDB,* Sept 2010. [Link](#).

Dremel: Interactive Analysis of Web-Scale Datasets

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer,
Shiva Shivakumar, Matt Tolton, Theo Vassilakis
Google, Inc.
{melnik,andrey,jlong,gromer,shiva,mtolton,theov}@google.com

ABSTRACT

Dremel is a scalable, interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, it is capable of running aggregation queries over trillion-row tables in seconds. The system scales to thousands of CPUs and petabytes of data, and has thousands of users at Google. In this paper, we describe the architecture and implementation of Dremel, and explain how it complements MapReduce-based computing. We present a novel columnar storage representation for nested records and discuss experiments on few-thousand node instances of the system.

exchanged by distributed systems, structured documents, etc. lend themselves naturally to a *nested* representation. Normalizing and recombining such data at web scale is usually prohibitive. A nested data model underlies most of structured data processing at Google [21] and reportedly at other major web companies.

This paper describes a system called Dremel[1] that supports interactive analysis of very large datasets over shared clusters of commodity machines. Unlike traditional databases, it is capable of operating on *in situ* nested data. *In situ* refers to the ability to access data 'in place', e.g., in a distributed file system (like GFS [14]) or another storage layer (e.g., Bigtable [8]). Dremel can execute many queries over such data that would ordinarily require a sequence of

Dremel in 2016

Dremel is **mission critical** for Google

In production for 10+ years, in every Google datacenter

Internal usage every month:

- O(**Exabytes**) analyzed
- O(**Quadrillions**) of rows scanned
- 80% of Googlers use it

BigQuery ≃ Dremel + cloud

Our idea of highly available, managed analytics:

- no indexing, no resharding, no storage resizing
- just ...

RUN QUERY

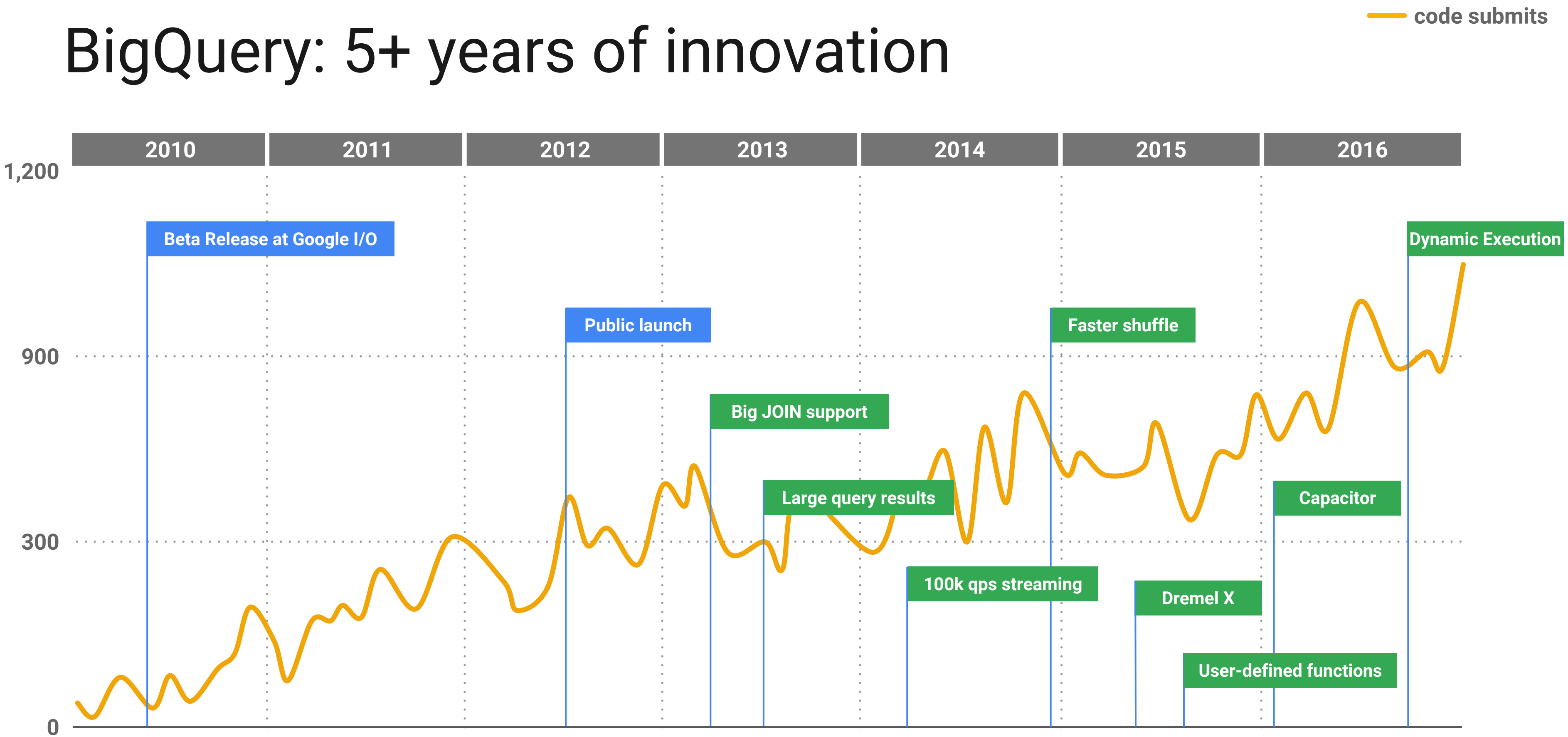
BigQuery ≃ Dremel + cloud

- Full-managed data warehouse
- Fast, petabyte-scale with SQL interface
- Encrypted, durable and highly available
- Near-unlimited resources, only pay for what you use
- Streaming ingest for unbounded data sets

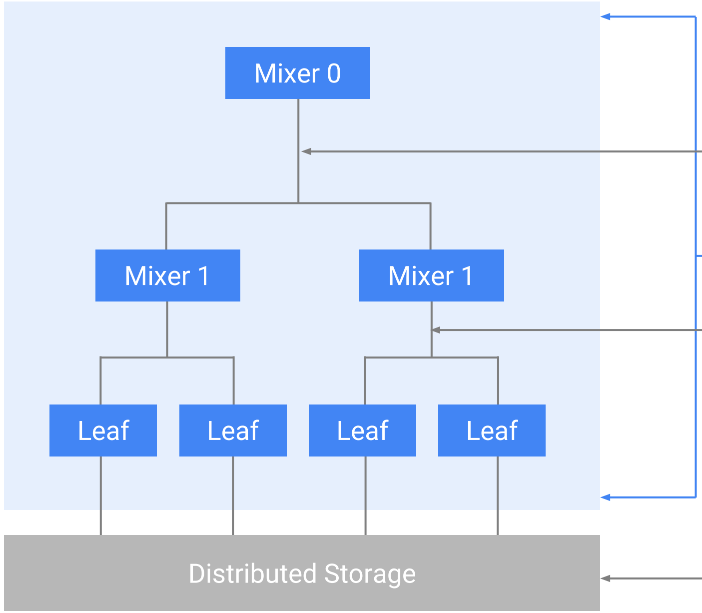
– General Availability in 2012
– Same engineering team
– Same code base

But what have we done lately?

BigQuery: 5+ years of innovation

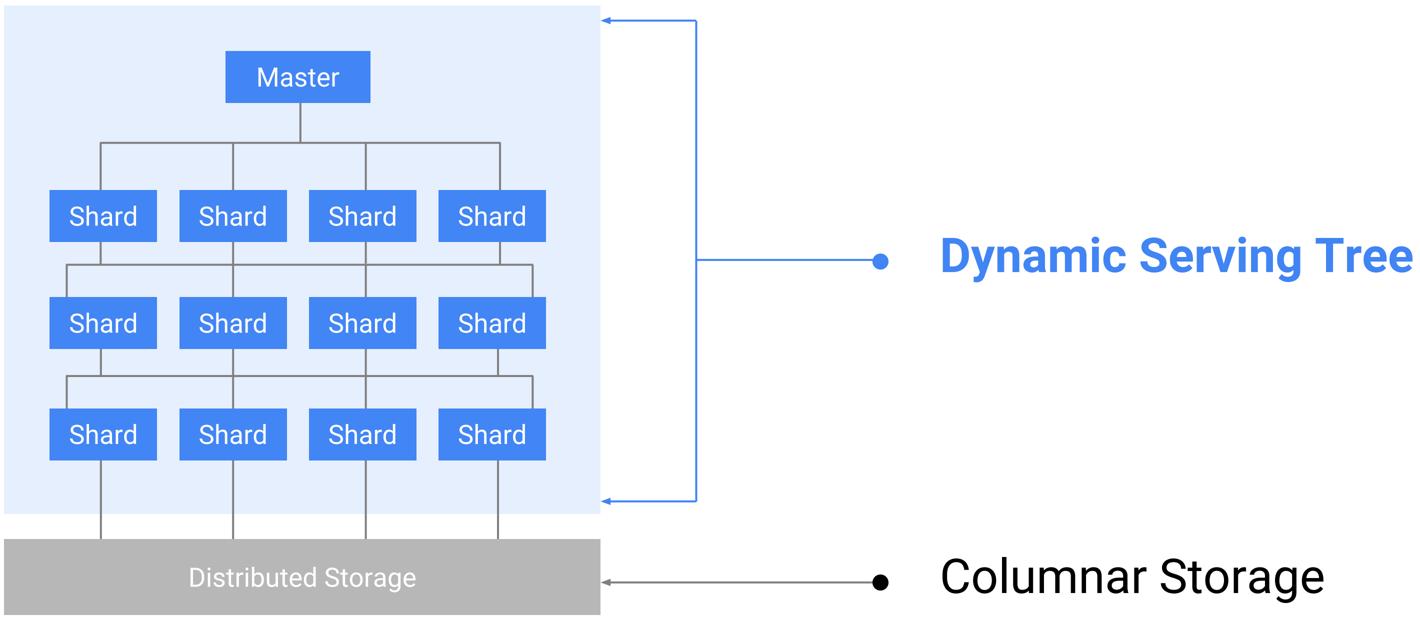


Dremel architecture: 2006–2015



- Partial Reduction
- **Long-lived Serving Tree**
- Diskless data flow
- Columnar Storage

Dremel X architecture (2015–now)



Storage engine: ColumnIO (2006–2015)

```
SELECT play_count FROM songs WHERE name CONTAINS “Sun”;
```

Data		Decompress	Filter	Emit
F$#h5	xc*	{Hey Jude, 5375}	CONTAINS(“Sun”)	
rm7y5	c8!	{My Michelle, 2188}	CONTAINS(“Sun”)	
rm7y5	8ec	{My Michelle, 9363}	CONTAINS(“Sun”)	
F$#h5	7h!	{Hey Jude, 9502}	CONTAINS(“Sun”)	
4t#@h	a7c	{Here Comes The Sun, 7383}	CONTAINS(“Sun”)	{7833}
rm7y5	c-%	{My Michelle, 3912}	CONTAINS(“Sun”)	

Storage engine: Capacitor (2016–now)

```
SELECT play_count FROM songs WHERE name CONTAINS "Sun";
```

Data

2	xc*
1	c8!
1	8ec
2	7h!
0	a7c
1	c-%

Emit

{7833}

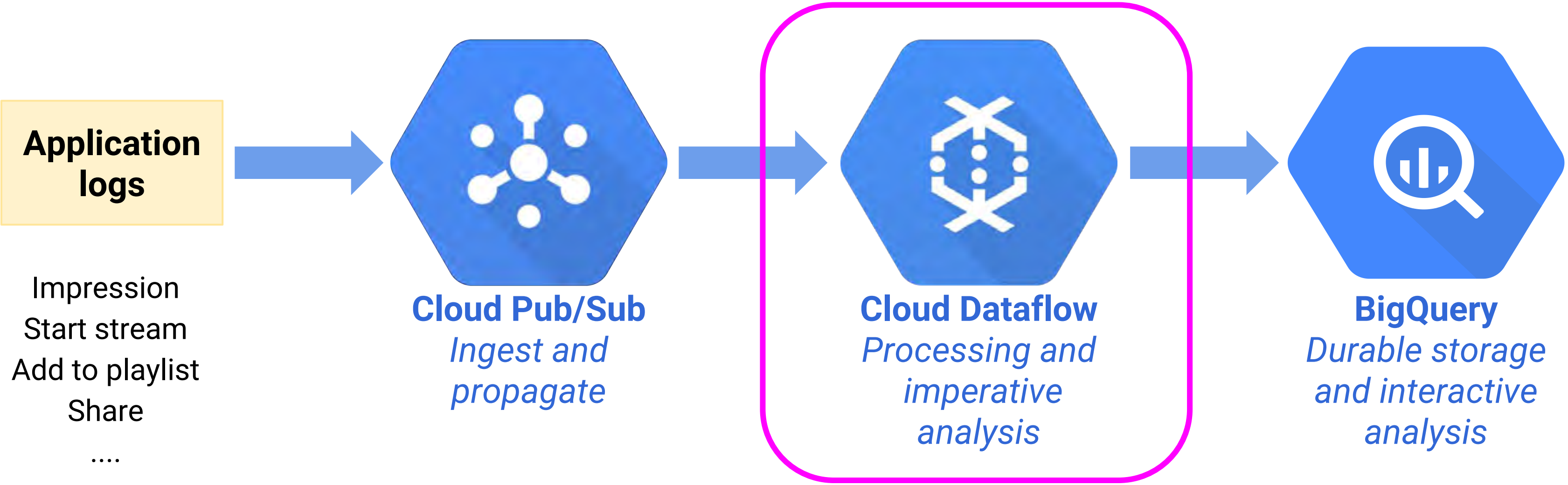
Dictionary		Filter	Lookup
0	Hey Jude	CONTAINS("Sun")	F
1	My Michelle	CONTAINS("Sun")	F
2	Here Comes the Sun	CONTAINS("Sun")	T

Performance improvement:

- 2x faster average over all queries
- 10–1000x faster for selective filters

Cloud Dataflow

One possible flow



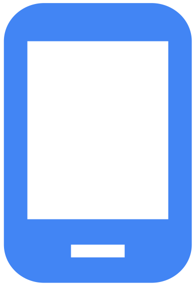
Questions you might ask

- Which application sections are receiving the most impressions?
- Who is the top artist by stream starts in the last minute?
- What is the average session length for users in Seattle?

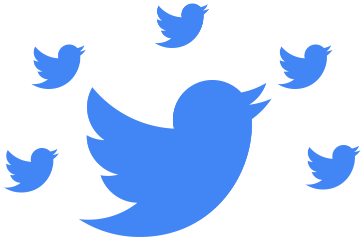
Time-to-answers matters

Sequence a human genome

Process point of sale transaction logs



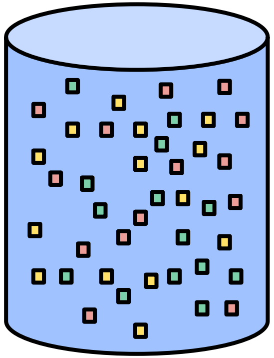
Who is the best at collecting Poké Balls?



Who/what is trending?

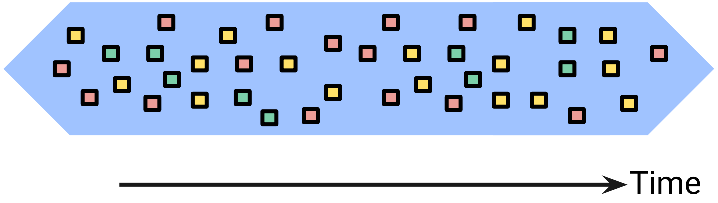
Data boundedness

Bounded



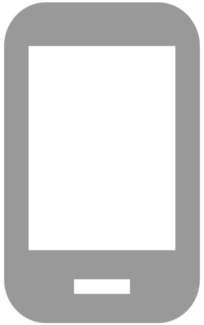
Finite data set
Fixed schema
Complete
Typically at rest

Unbounded



Infinite data set
Potentially changing schema
Never complete
Typically not at rest

Latency happens



Transmit

Network delay or unavailable

Ingest delay (write latency)

Ingest failure

Ingest

Throughput (read latency)

Backlog

Hardware failure

Process

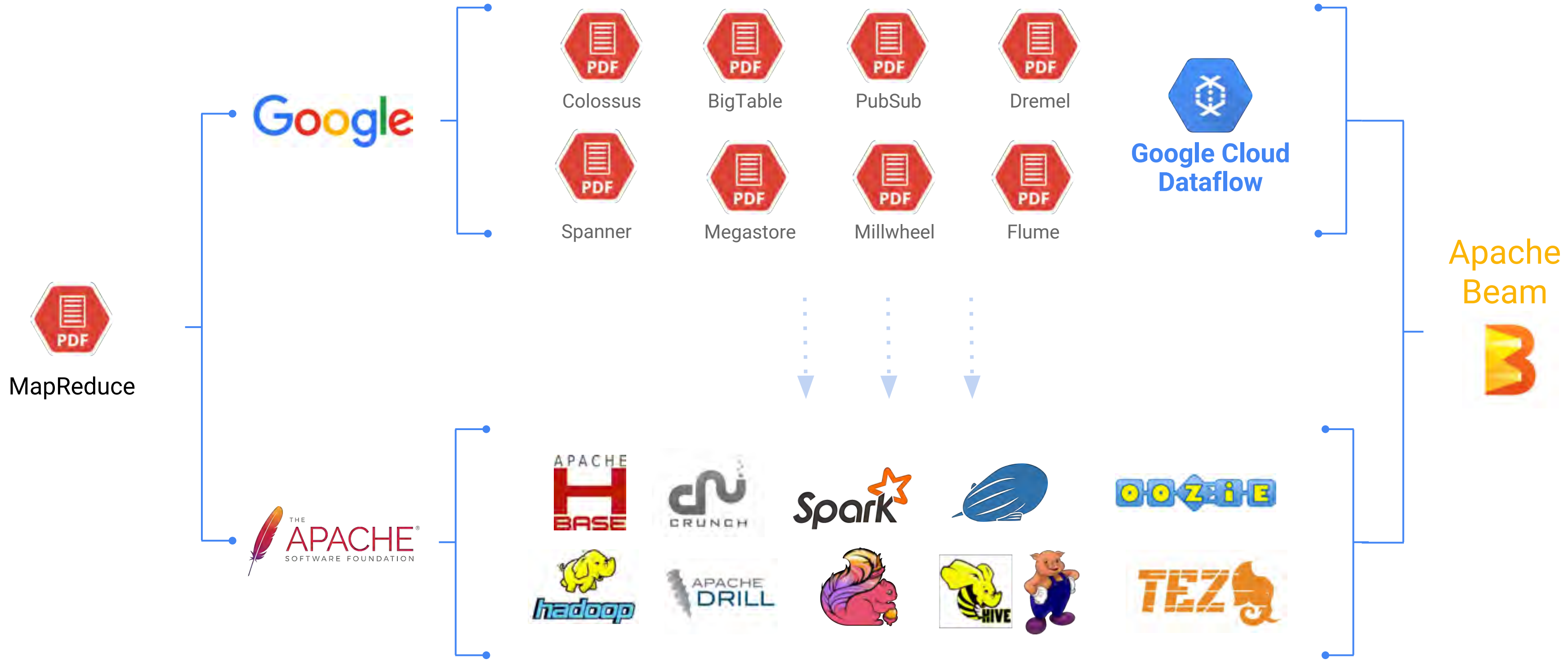
Poor engine design

Starved resources

Internal backlog

Confused heuristics

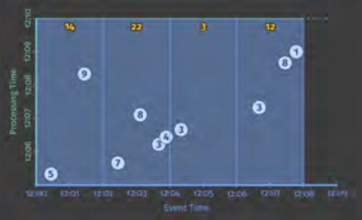
The evolution of Apache Beam



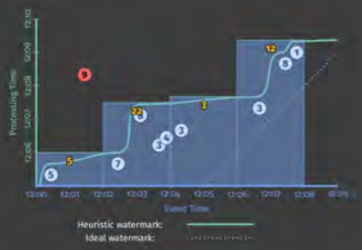
Apache Beam (incubating) is a unified programming model designed to provide efficient and portable data processing pipelines

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

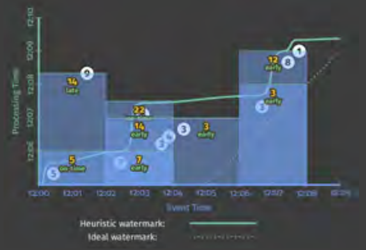
1.Classic batch



2. Batch with fixed windows

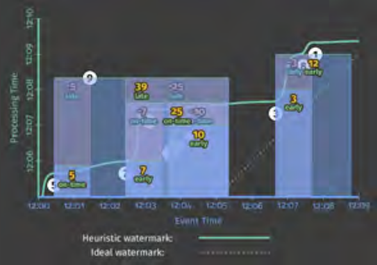


3. Streaming



4. Streaming with speculative + late data

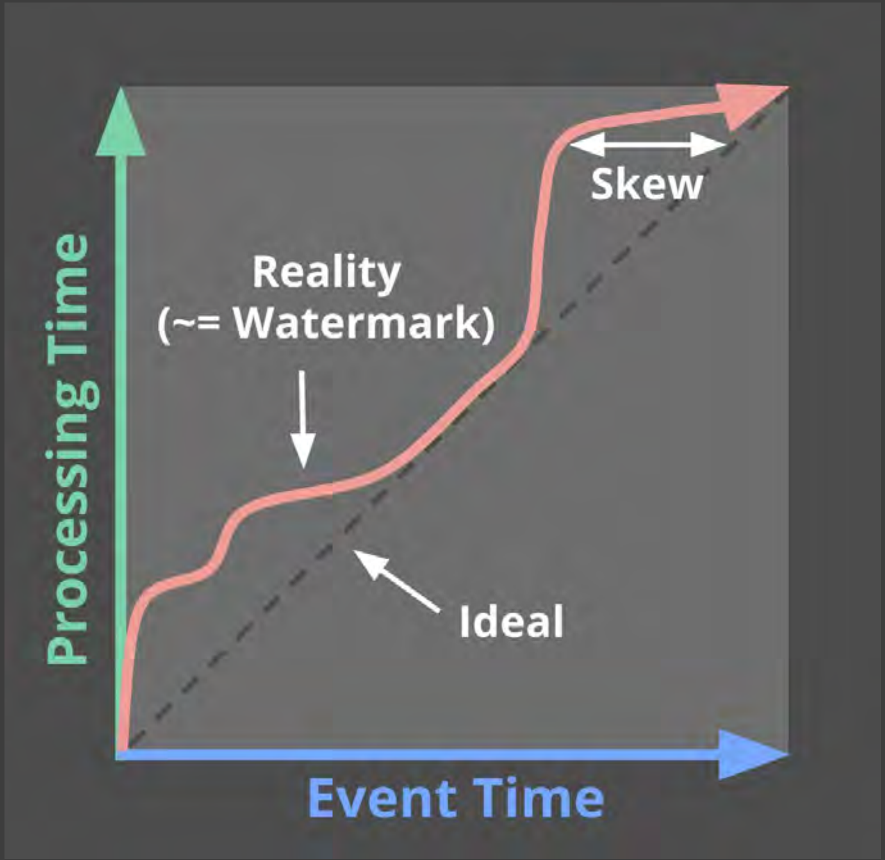
5. Streaming with retractions



6. Sessions

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

Formalizing Event-Time Skew



Watermarks describe **event** time progress in the **processing** time space:

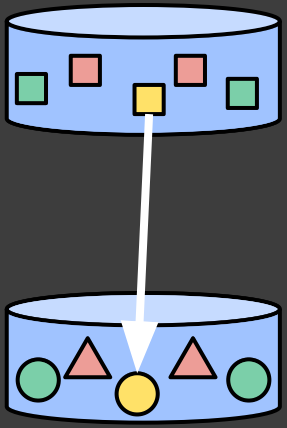
> *"No timestamp earlier than the watermark will be seen"*

Often based on heuristics

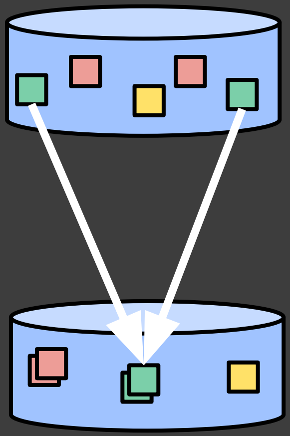
- Too slow? Results are *delayed :-(*
- Too fast? Some data is *late :-(*

- **What** are you computing?
- **Where** in event time?
- **When** in processing time?
- **How** do refinements relate?

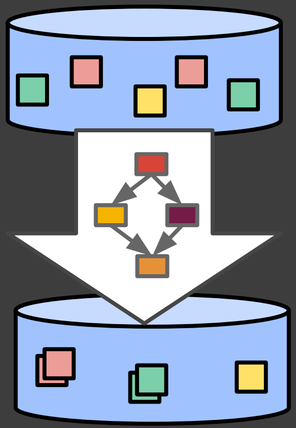
What are you computing?



Per-element

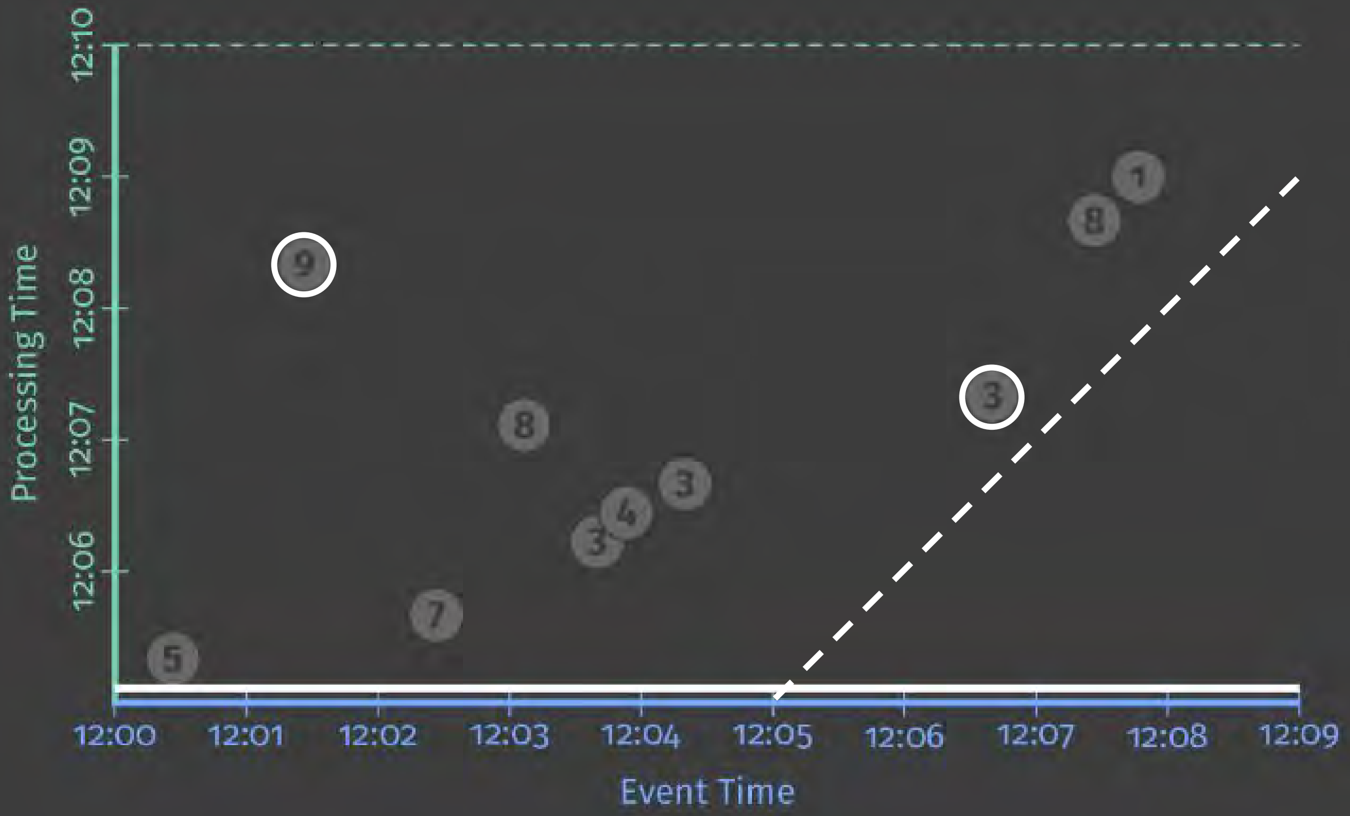


Aggregations

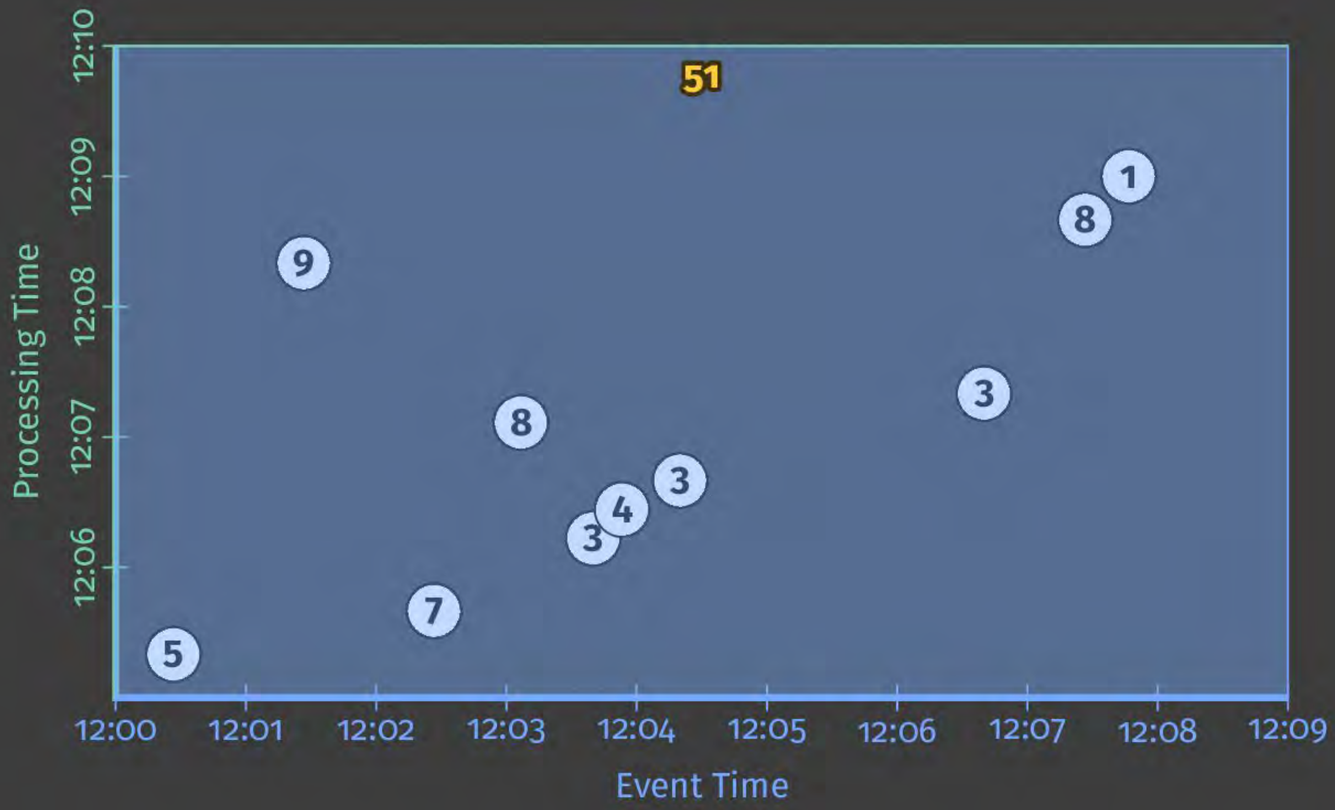


Compositions (subgraphs)

What: computing integer sums

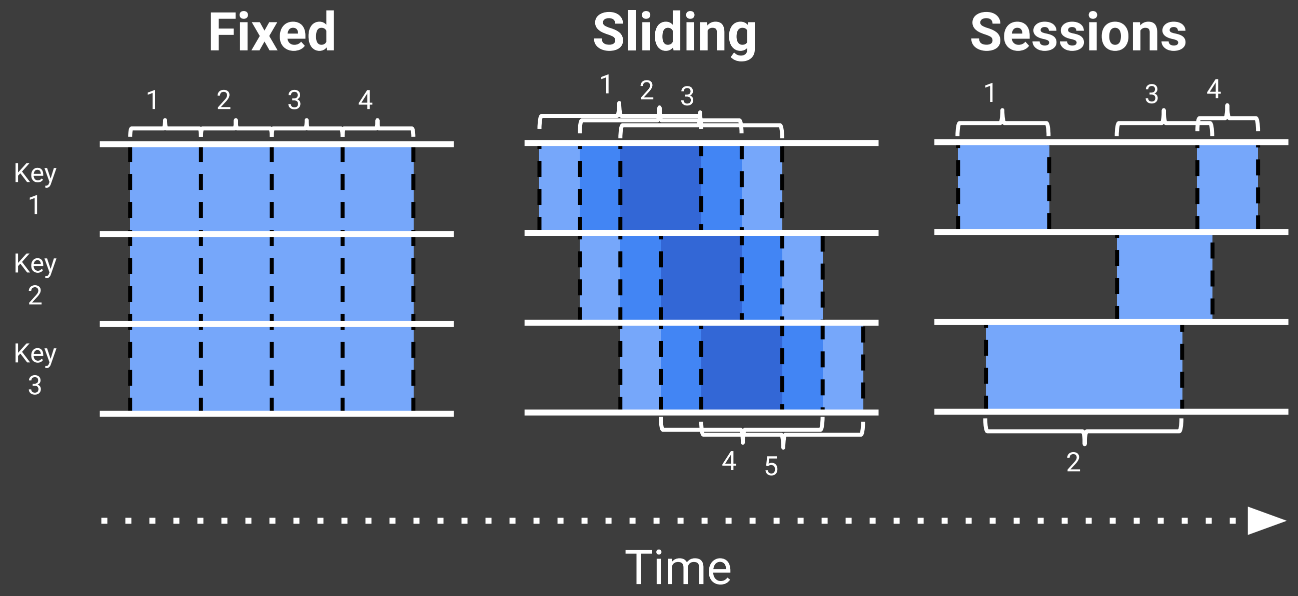


What: computing integer sums



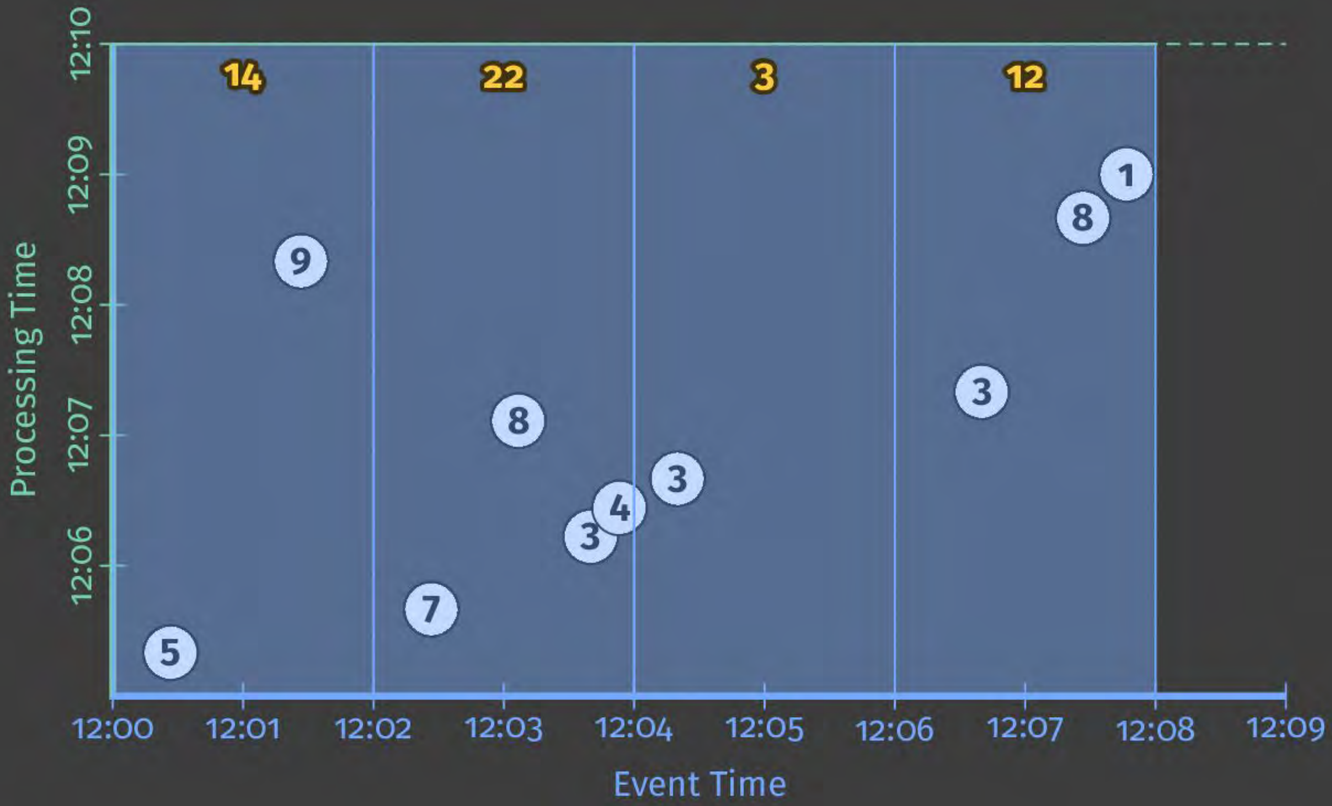
Where in event time?

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

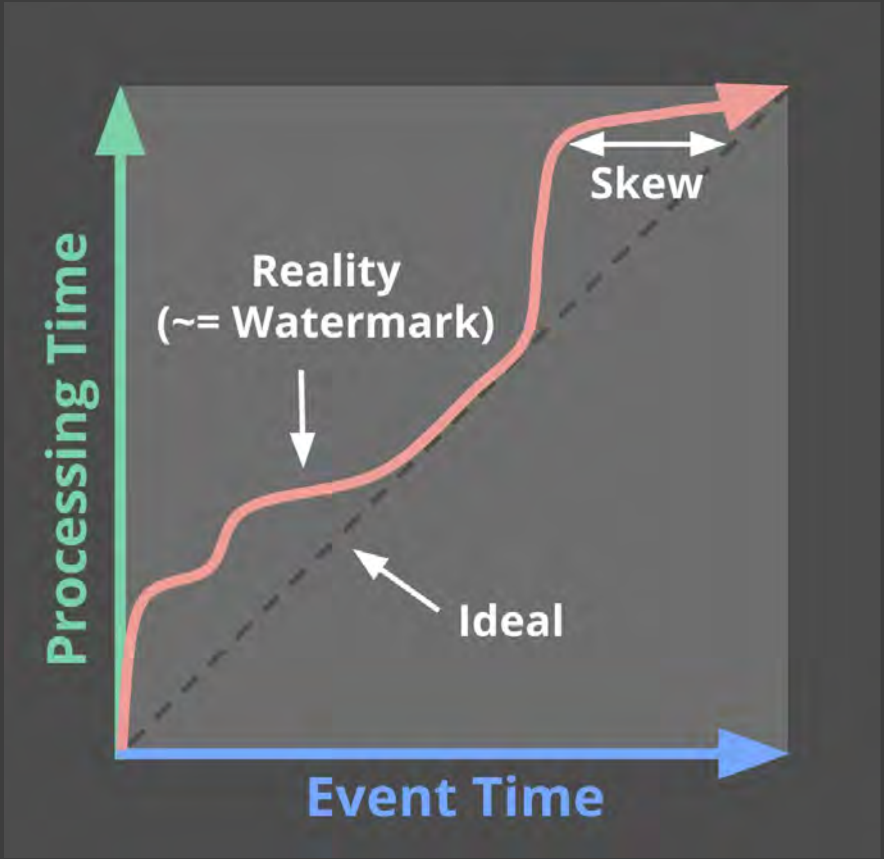


Often required when doing aggregations over unbounded data.

Where: Fixed 2-minute Windows

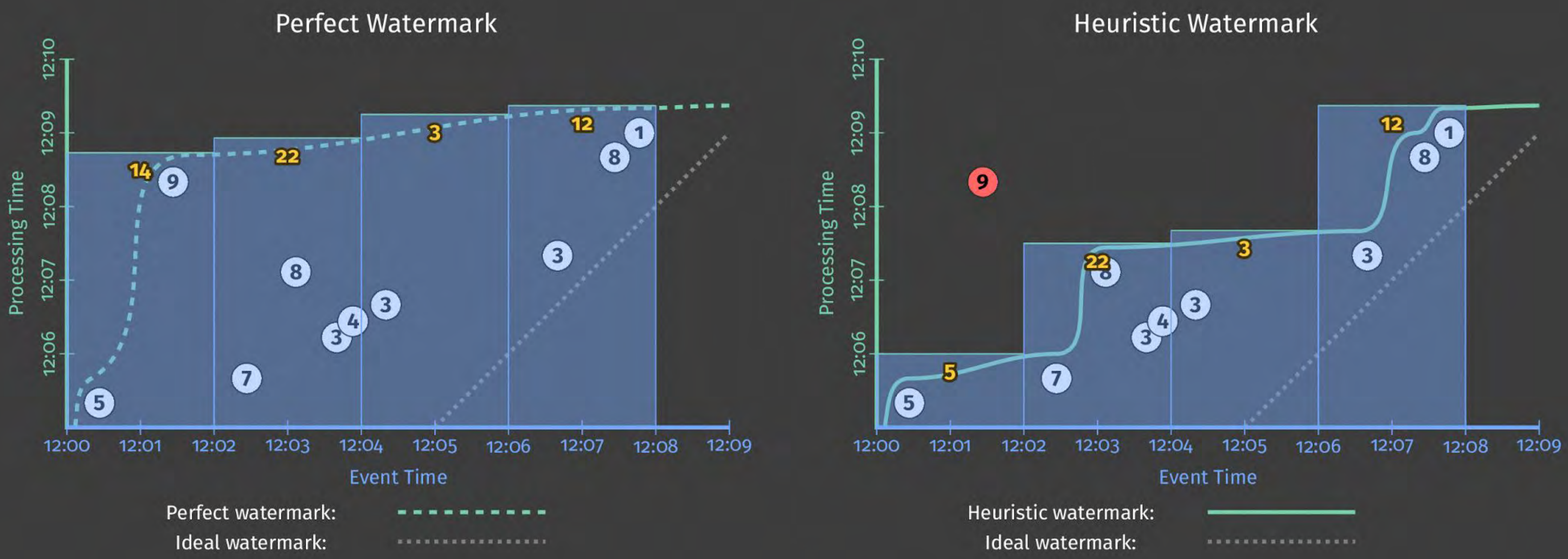


When in processing time?

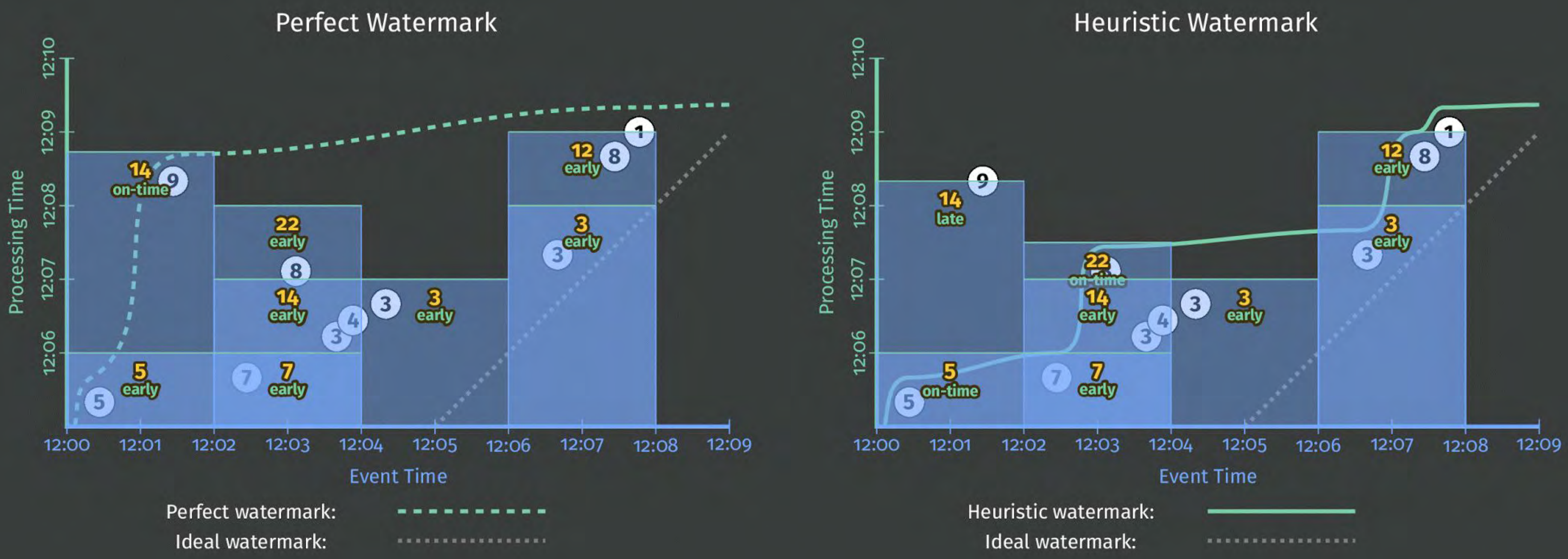


- Triggers control when results are emitted.
- Triggers are often relative to the watermark.

When: triggering at the watermark



When: early and late firings



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

How do refinements relate?

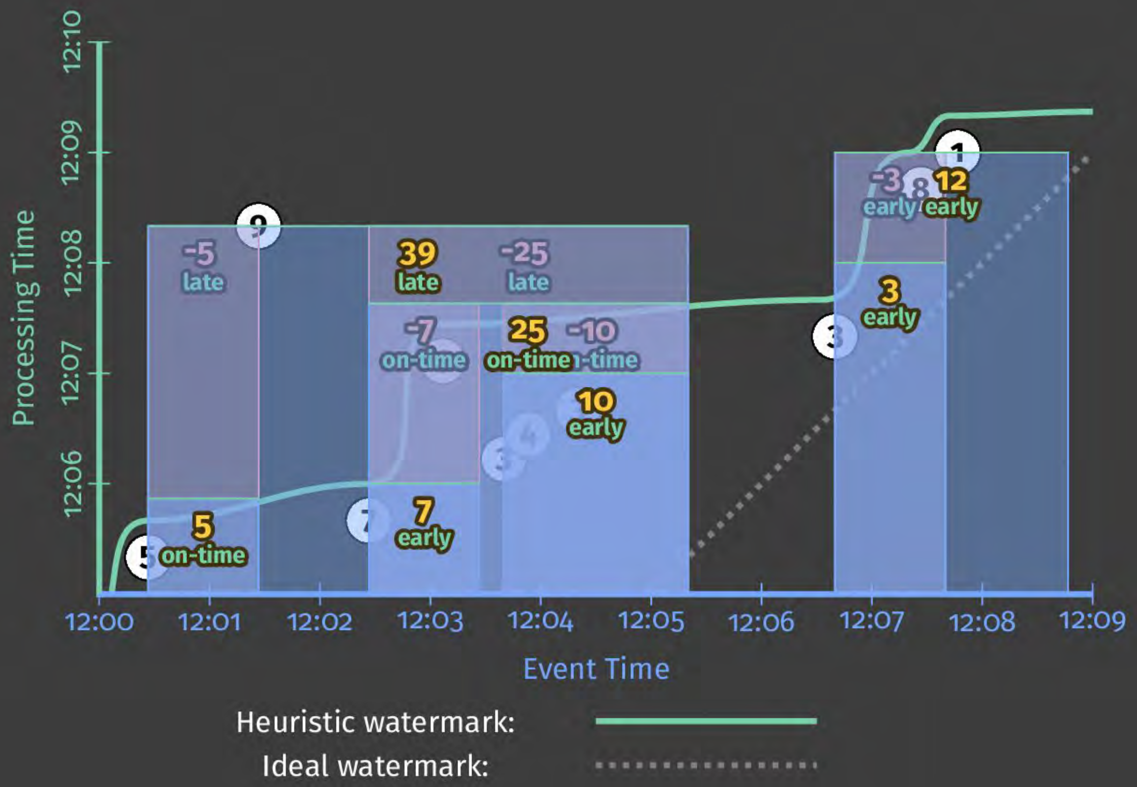
Q: if there are multiple panes per window … what should be emitted?

- discard – emit the delta over the pane
- accumulate – emit running total
- accumulate + retract* – retract last sum & emit new running total

A: drive by needs of the downstream consumer

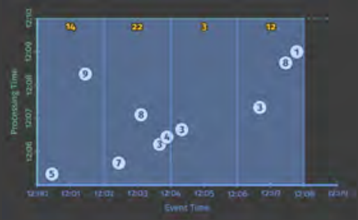
*Accumulating & Retracting not yet implemented in Apache Beam.

How: add newest, remove previous

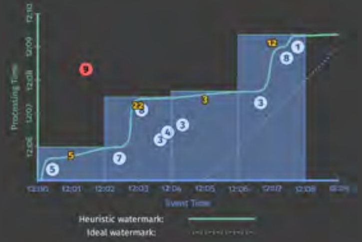


Customizing What When Where How

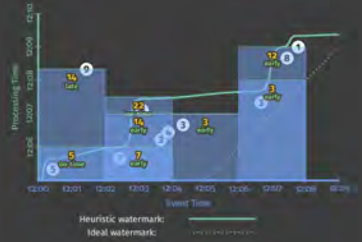
1.Classic batch



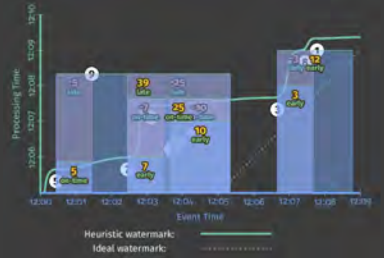
2. Batch with fixed windows



3. Streaming

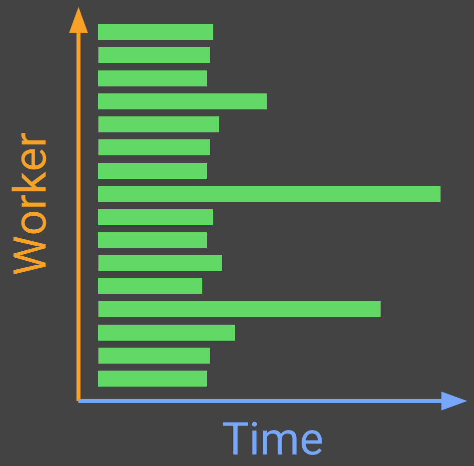


4. Streaming with speculative + late data



5. Streaming with retractions

The straggler problem



Work is unevenly distributed across tasks

- Underlying data size
- Processing differences
- Runtime effects

Effects are cumulative per stage

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

Beam readers enable **dynamic adaptation**

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

APIs for how much work is pending:

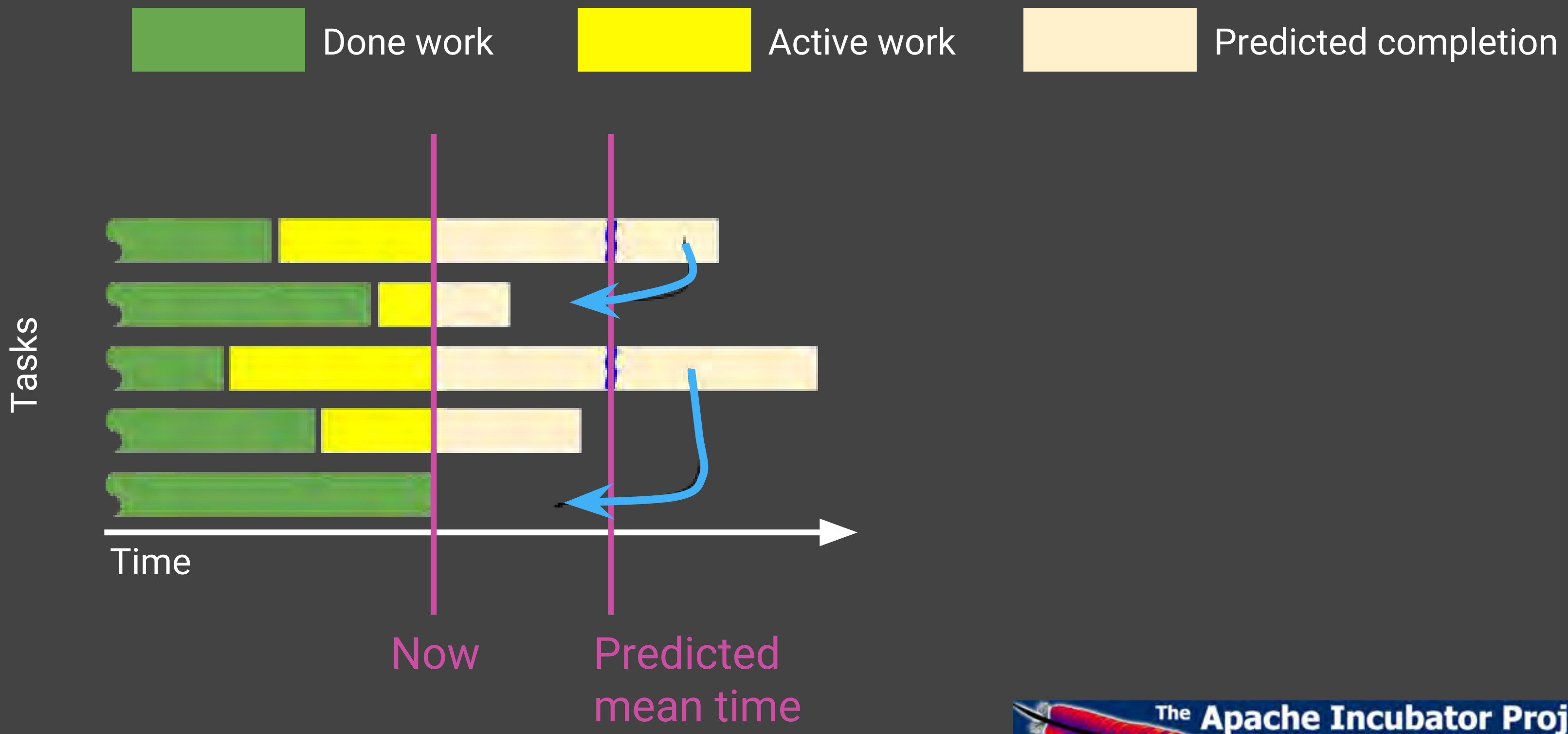
- Bounded: double **getFractionConsumed**()
- Unbounded: long **getBacklogBytes**()

Work-stealing:

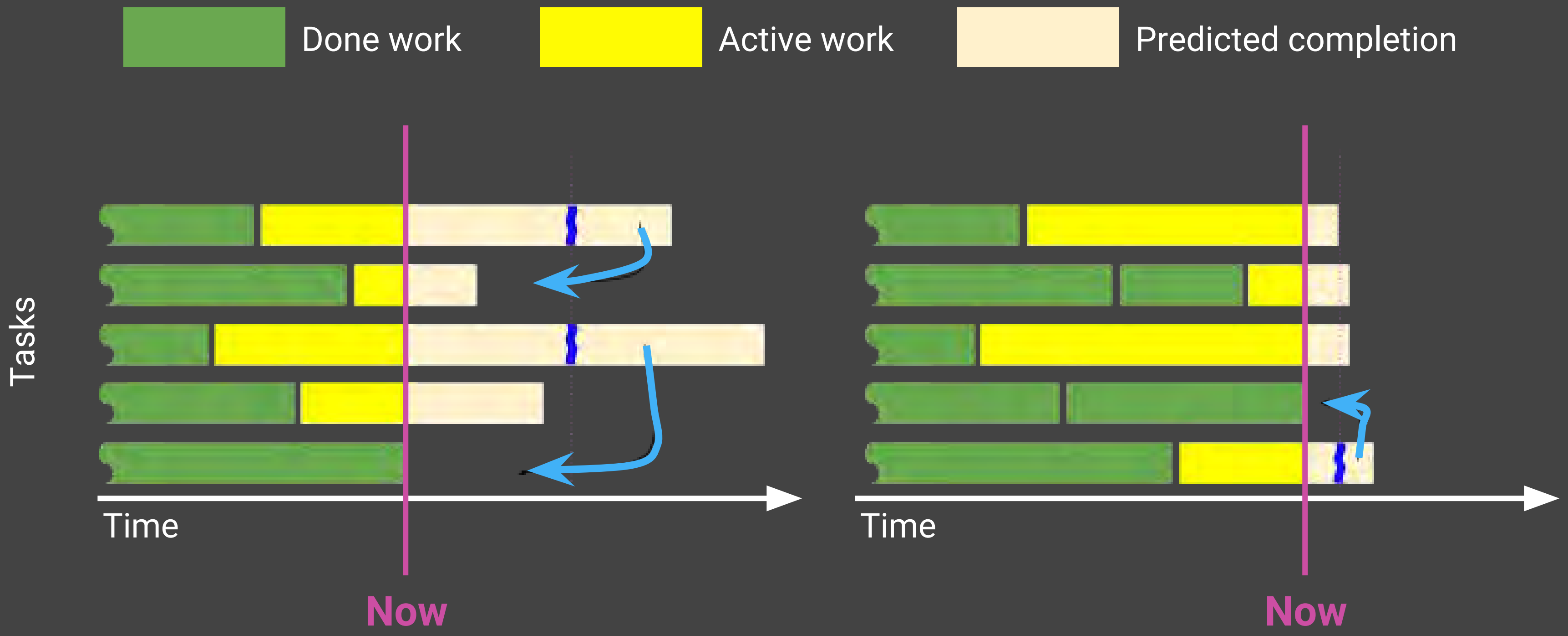
- Bounded: Source **splitAtFraction**(double)
  int **getParallelismRemaining**()

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

Dynamic work rebalancing



Dynamic work rebalancing

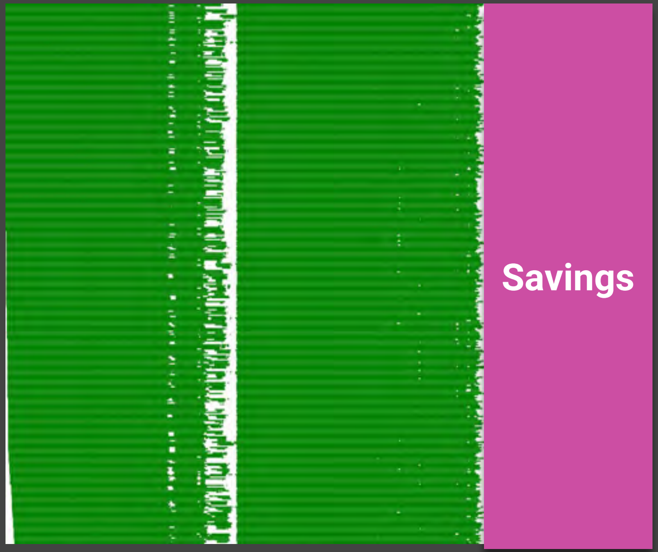


The Apache Incubator Project
http://incubator.apache.org/

Dynamic work rebalancing: a real example

Beam pipeline on the Google Cloud Dataflow runner

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



Same pipeline
with dynamic work rebalancing

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

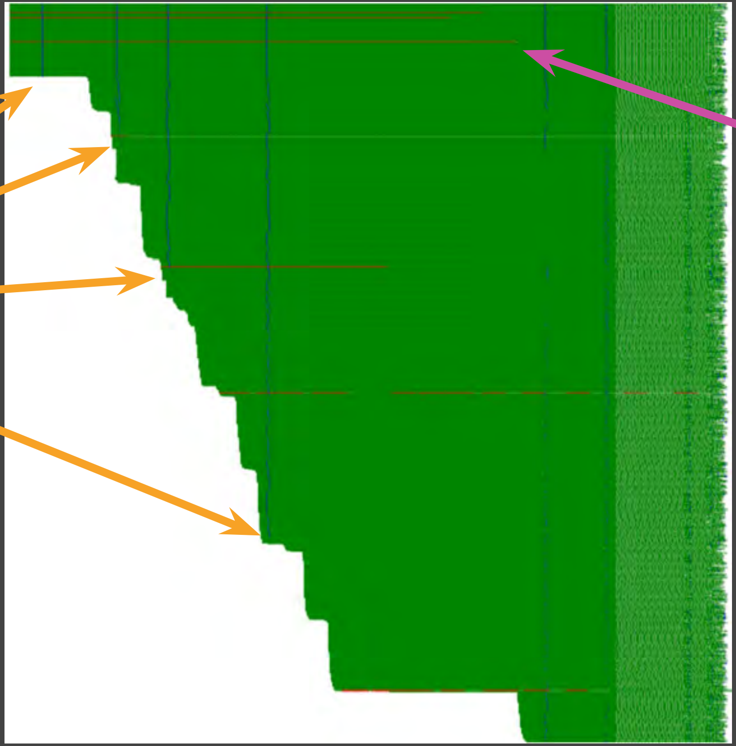
Dynamic work rebalancing + **Autoscaling**

Beam pipeline on the Google Cloud Dataflow runner

Initially allocate ~80 workers based on input size

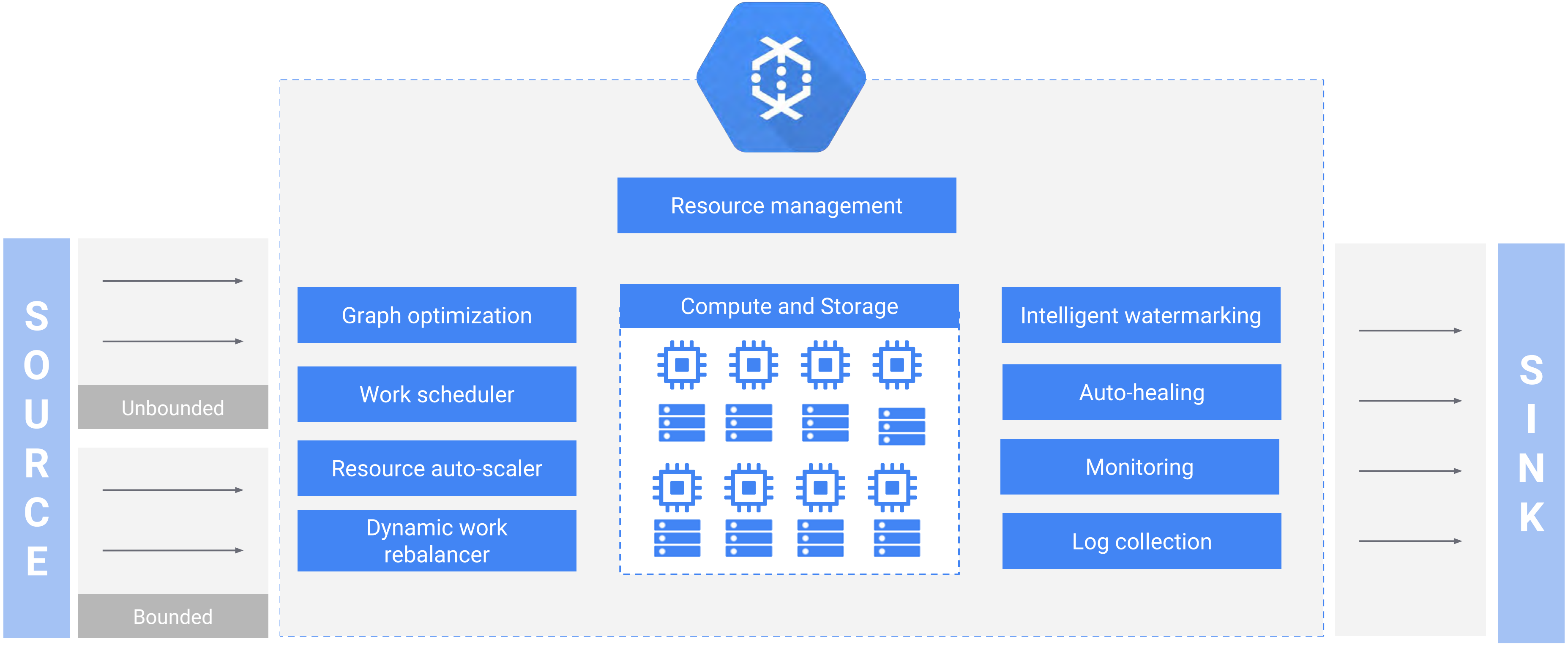
Multiple rounds of upsizing enabled by dynamic splitting

Scale up to 1000 workers
* tasks stay well-balanced
* without initial oversplitting



Long-running tasks aborted without causing stragglers

Cloud Dataflow execution runner



The Dataflow Model & Cloud Dataflow

Dataflow Model & SDKs

a unified model for
batch and stream processing

Google Cloud Dataflow

no-ops, fully managed service

The *Beam* Model & Cloud Dataflow

Apache Beam

a unified model for
batch and stream processing
supporting multiple runtimes

Google Cloud Dataflow

a great place to run Beam

What is in Apache Beam?

1. The Beam Model: **What** / **Where** / **When** / **How**

2. SDKs for writing Beam pipelines – starting with Java

3. Runners for existing distributed processing backends
   - **Apache Flink** (thanks to data Artisans)
   - **Apache Spark** (thanks to Cloudera)
   - **Google Cloud Dataflow** (fully managed service)
   - **Local** (in-process) runner for testing

Categorizing runner capabilities

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	✕	✕	✕	✕

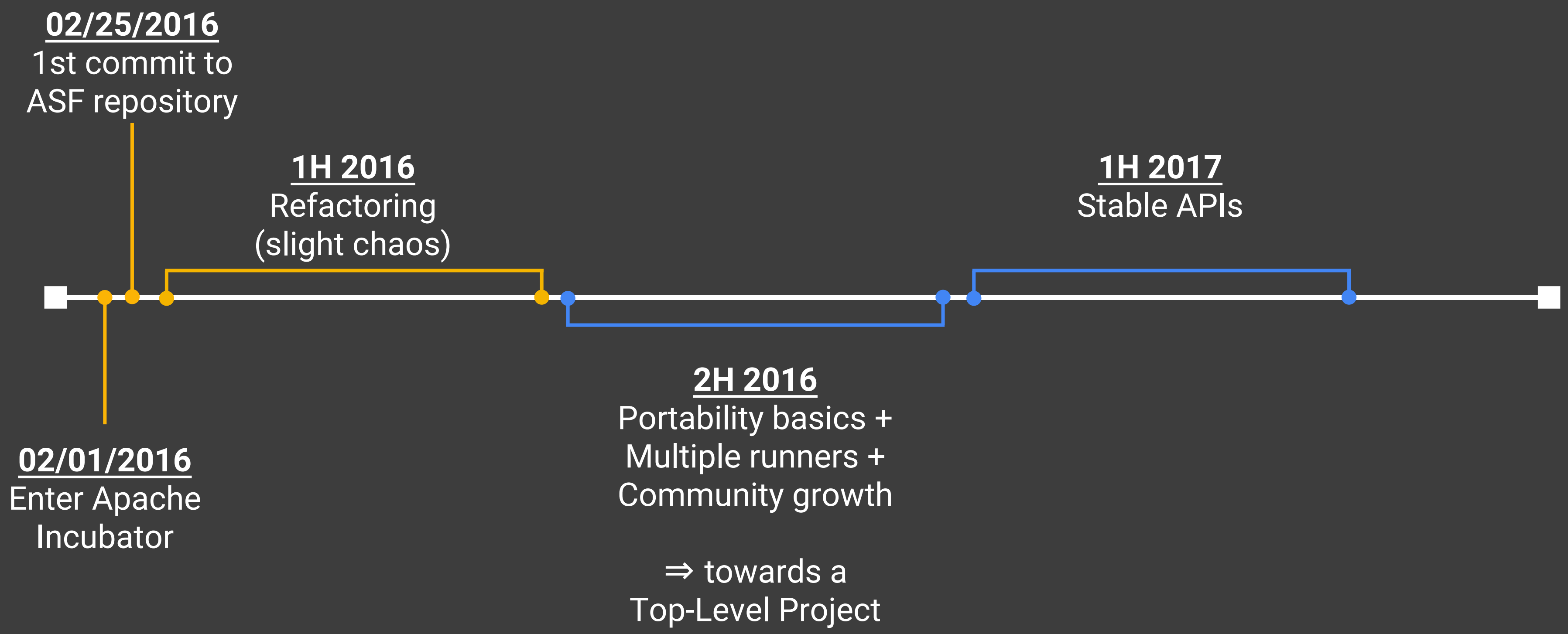
How do refinements relate?

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

http://beam.incubator.apache.org/capability-matrix/

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

Apache Beam roadmap



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

Growing the Beam community

Collaborate – a community-driven effort across many organizations and contributors

Grow – the Beam ecosystem and community with active, open involvement

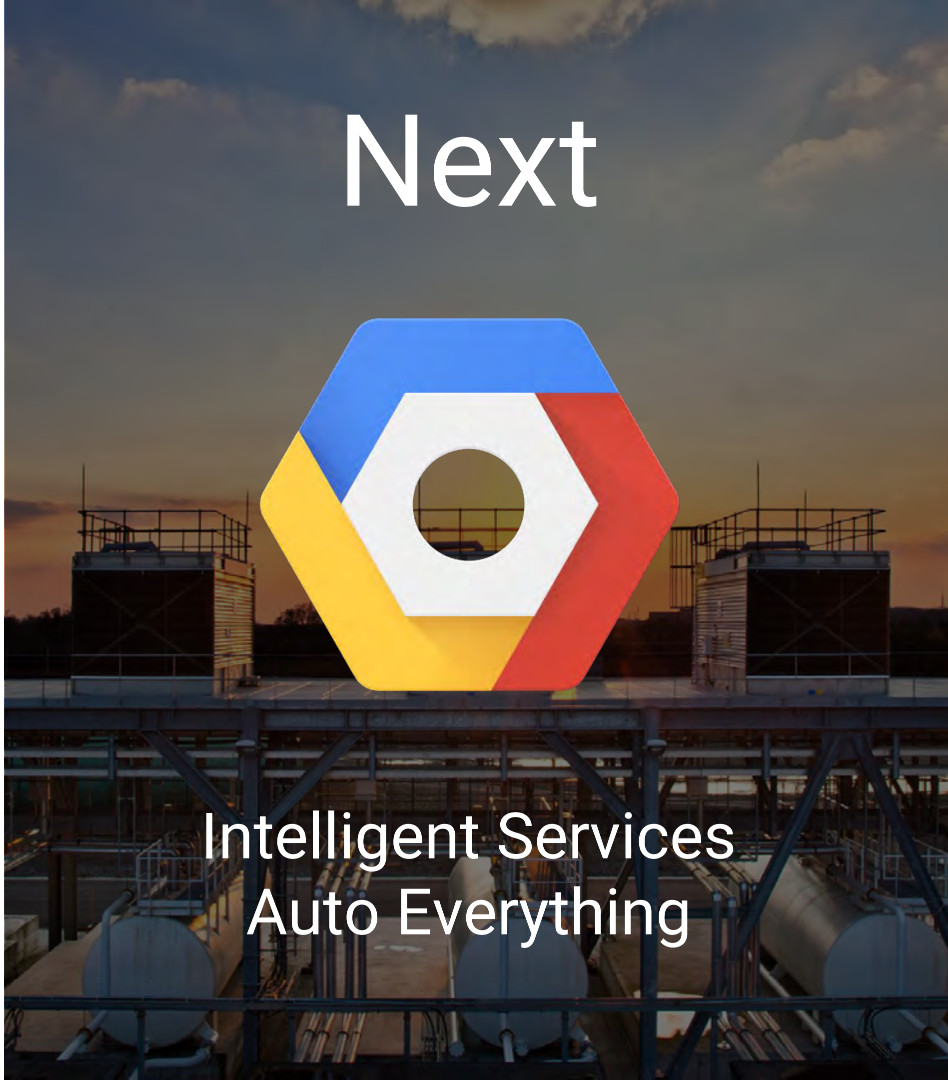
The Apache Incubator Project
http://incubator.apache.org/

Now

1st Wave
Colocation

2nd Wave
Virtualized
Datacenters

User Managed
User Configured
User Maintained

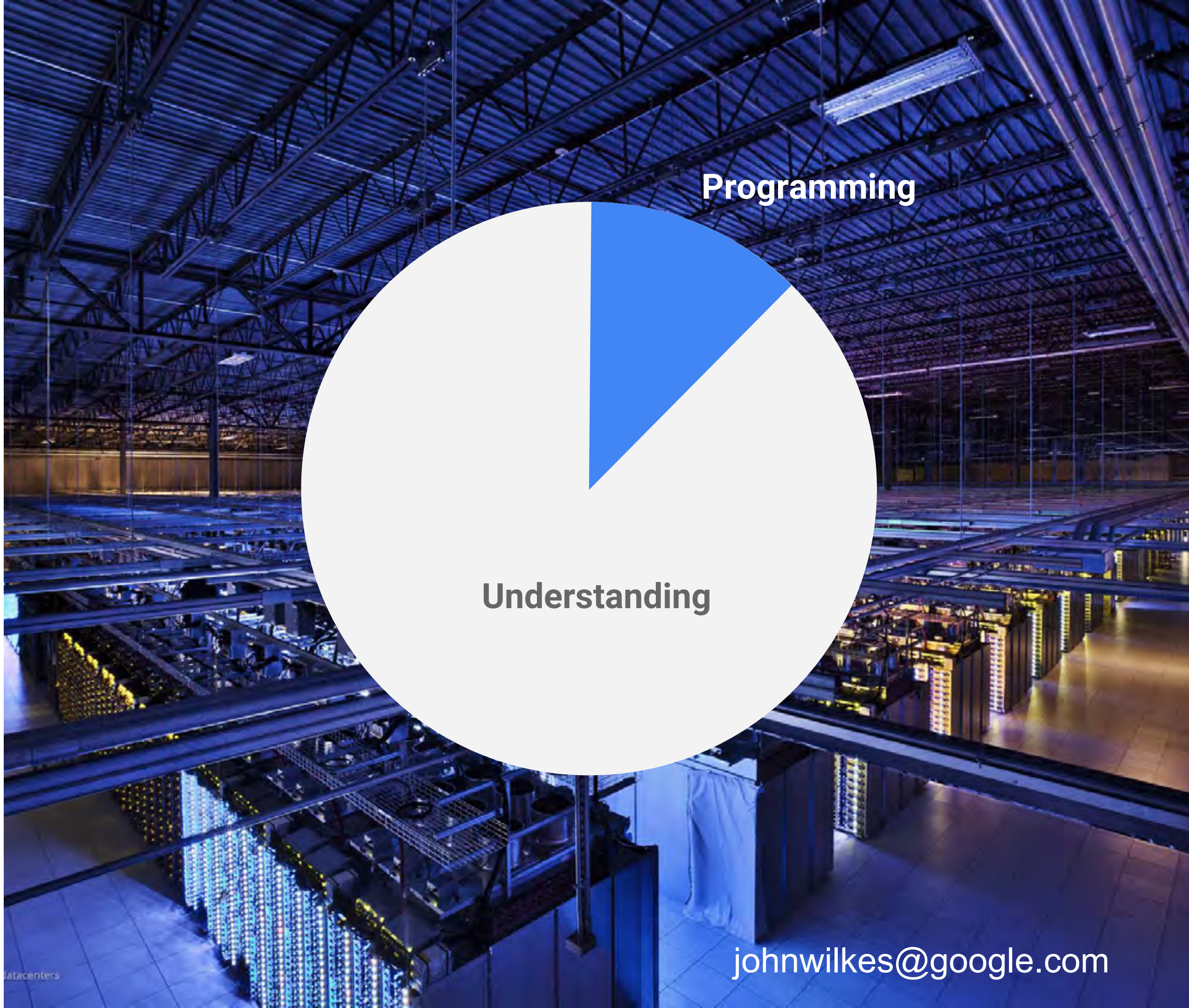


Big Data with Google

Focus on insights, not infrastructure

From batch to real-time

Apache Beam offers a common model across execution engines: please contribute!



johnwilkes@google.com