The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, Sam Whittle

Google

VLDB 2015

Presented by Johann Schleier-Smith
Berkeley CS294-110
September 16, 2015

The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in **Massive-Scale**, Unbounded, Out-of-Order Data Processing

The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, **Unbounded**, Out-of-Order Data Processing

The Dataflow Model: A Practical Approach to Balancing **Correctness**, **Latency**, and **Cost** in Massive-Scale, Unbounded, Out-of-Order Data Processing

The Dataflow Model: A **Practical** Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing

Programming Model

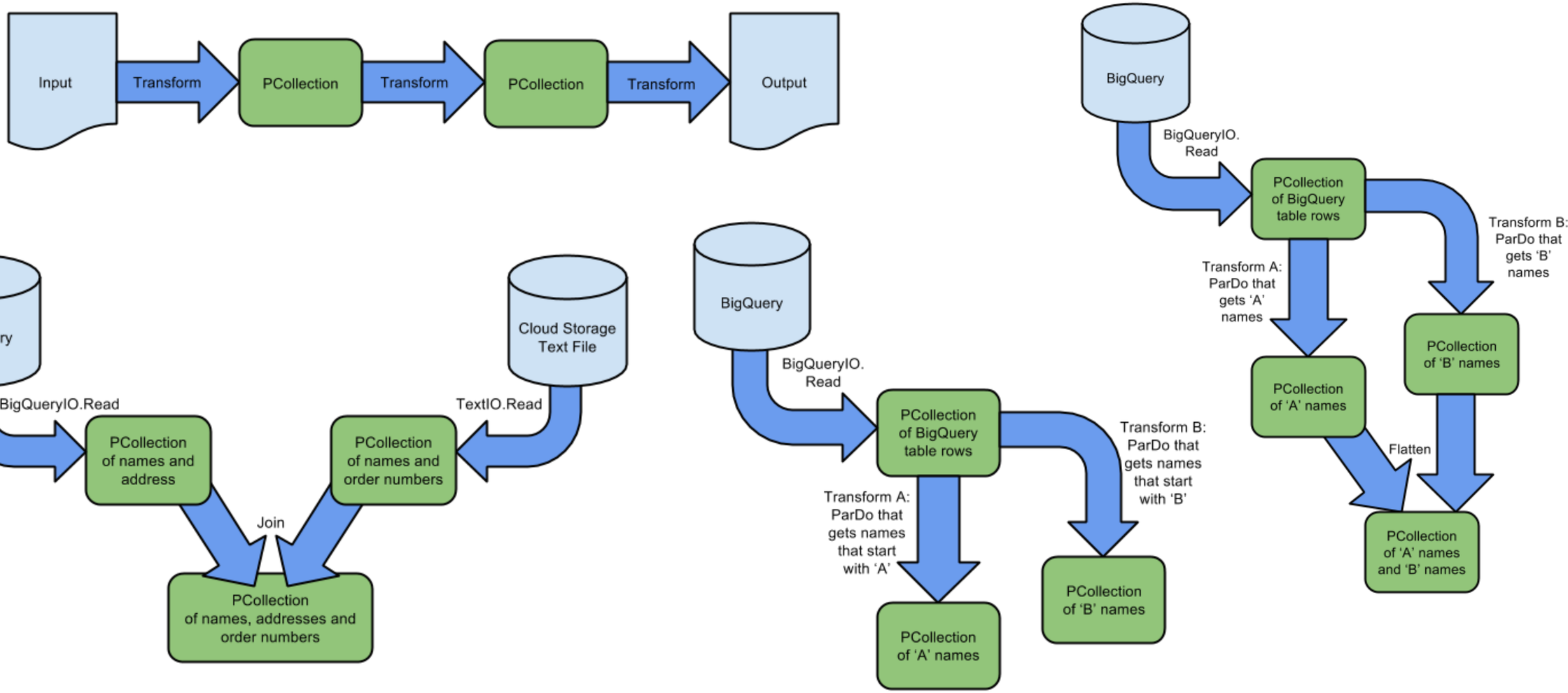
- Timestamped data
- Pipelines
- PCollections
- Core transformations
- Windows
- Triggers and watermarks

Timestamped Data

- Key and value
- Event timestamp
- Window timestamps: [*begin*,*end*)

- Processing time

Pipelines


Input
Transform
PCollection
Transform
PCollection
Transform
Output
BigQuery
BigQueryIO.Read
PCollection of names and address
Cloud Storage Text File
TextIO.Read
PCollection of names and order numbers
Join
PCollection of names, addresses and order numbers
BigQuery
BigQueryIO. Read
PCollection of BigQuery table rows
Transform A: ParDo that gets names that start with 'A'
Transform B: ParDo that gets names that start with 'B'
PCollection of 'A' names
PCollection of 'B' names
BigQuery
BigQueryIO. Read
PCollection of BigQuery table rows
Transform A: ParDo that gets 'A' names
Transform B: ParDo that gets 'B' names
PCollection of 'A' names
PCollection of 'B' names
Flatten
PCollection of 'A' names and 'B' names

PCollections

- Bag of (key, value, timestamp, window)
- Immutable
- No random access
- Must specify bounded or unbounded

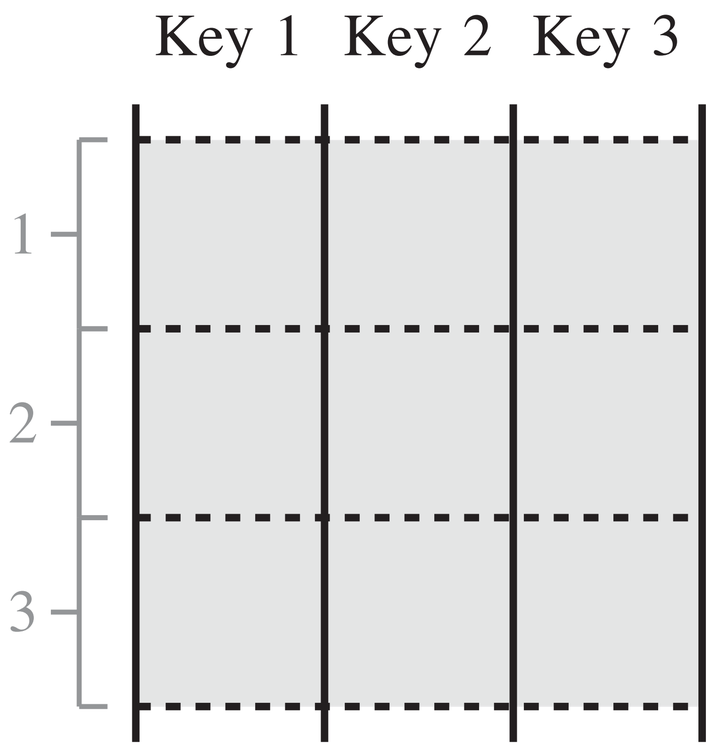
Core Transformations

ParDo(DoFn: (K_{in}, V_{in}) => Collection[(K_{out}, V_{out})])

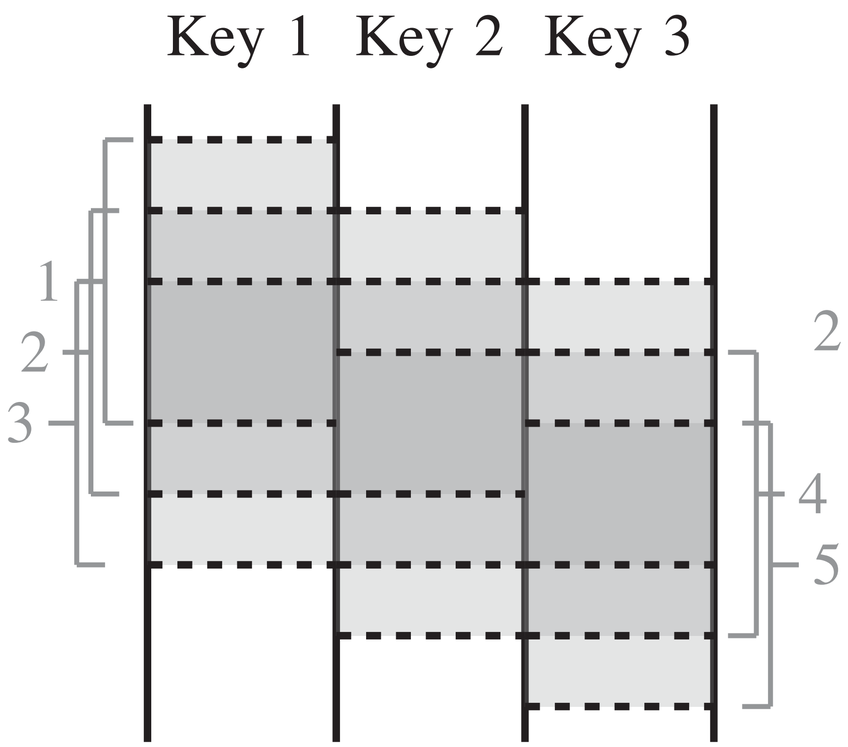
GroupByKey() / GroupByKeyAndWindow()

See also: FlumeJava: easy, efficient data-parallel pipelines (PLDI 2010).

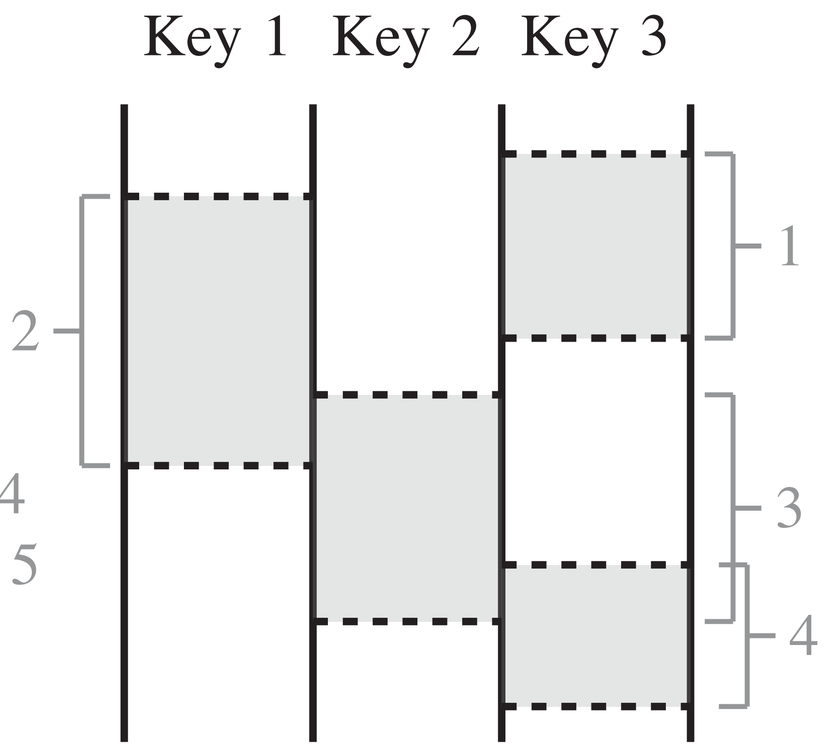
Windows



Fixed



Sliding



Sessions

Triggers

- For GroupByKeyAndWindow()
- Fires whenever a window is ready
- Watermark suggests lower bound for processed data

See also: MillWheel: Fault-tolerant stream processing at internet scale (VLDB 2013)

Programming Highlights

- Unified API for batch and streaming
- Collections interface familiar from DryadLINQ, FlumeJava, Spark
- Must never rely on any notion of completeness

Correctness, Latency, Cost

- Trigger conservatively for low cost
- Trigger aggressively for low latency
- Skip trigger on old data for low correctness

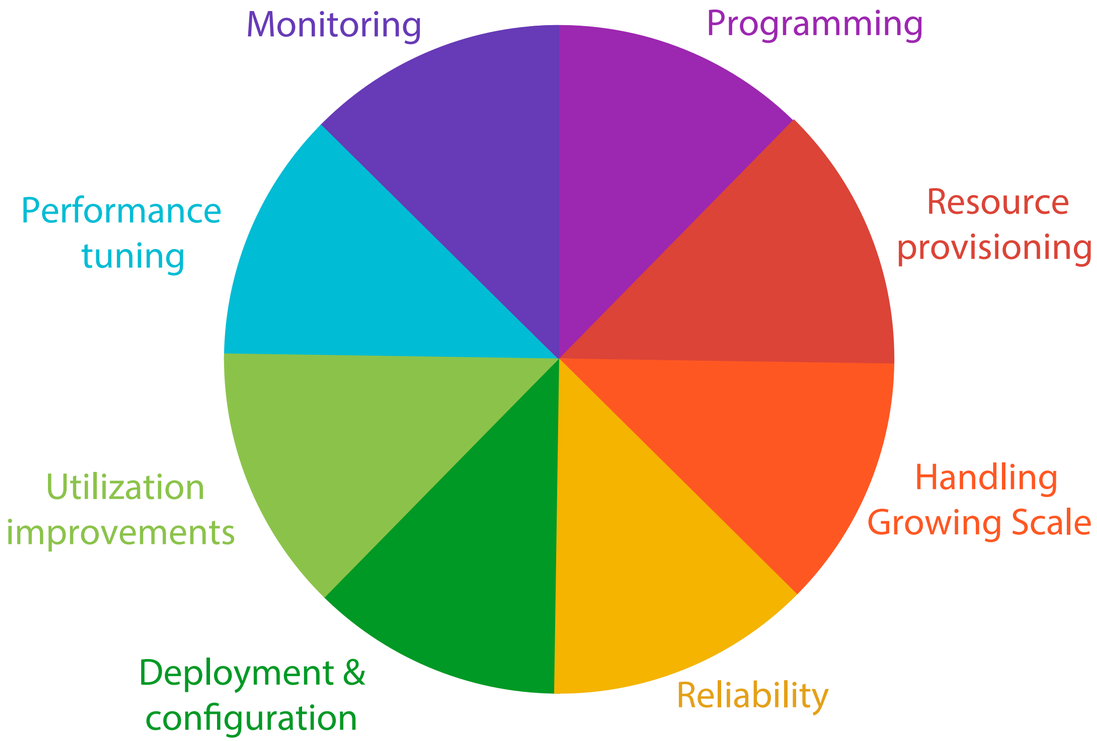
Cloud Service

- Public SDK derived from internal software
- Automatic resource scaling
- Job cost =

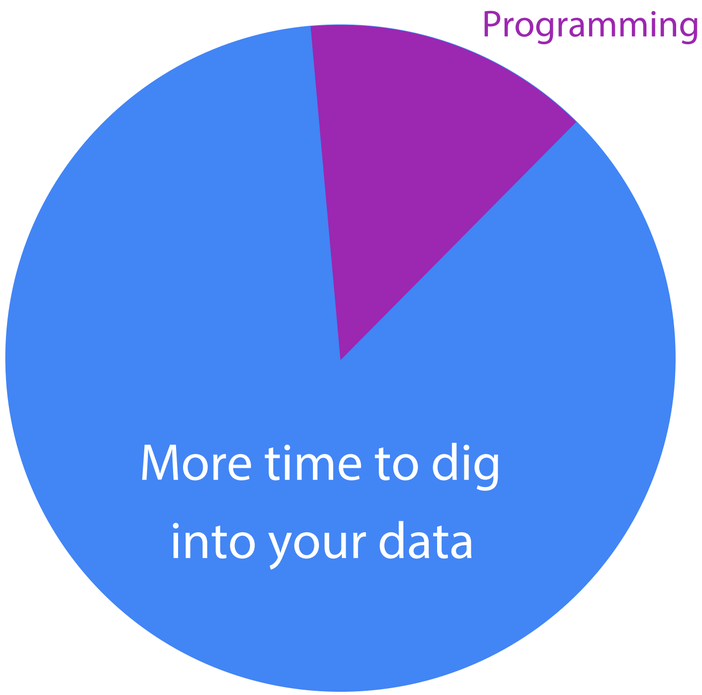
  (work time x $0.084/hr) + (shuffled bytes x $0.0025/GB)

Pricing source: https://cloud.google.com/dataflow/new-pricing

Optimizing your time: no-ops, no-knobs, zero-config



Typical Data Processing



Data Processing with Google Cloud Dataflow

Source: http://ictlabs-summer-school.sics.se/slides/google%20cloud%20dataflow.pdf

Discussion

- Is promised unification real?
- Is the future of data unbounded data?
- Beyond sessions, what windowing methods are useful? Does windowing apply to all problems?
- Is it a just a reporting solution? E.g., can it train machine learning?
- Is programming model still too complicated?
- Has the “fluffy cloud” arrived?