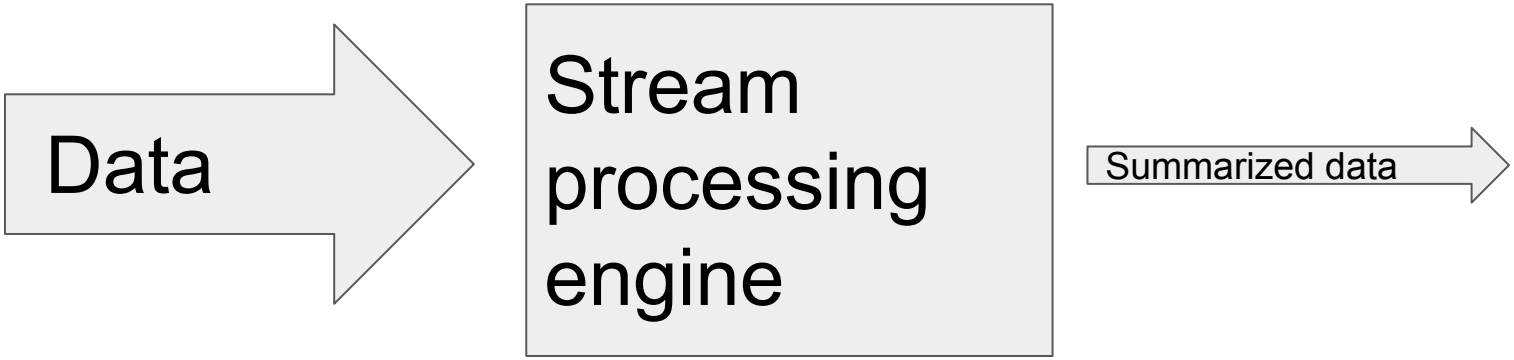
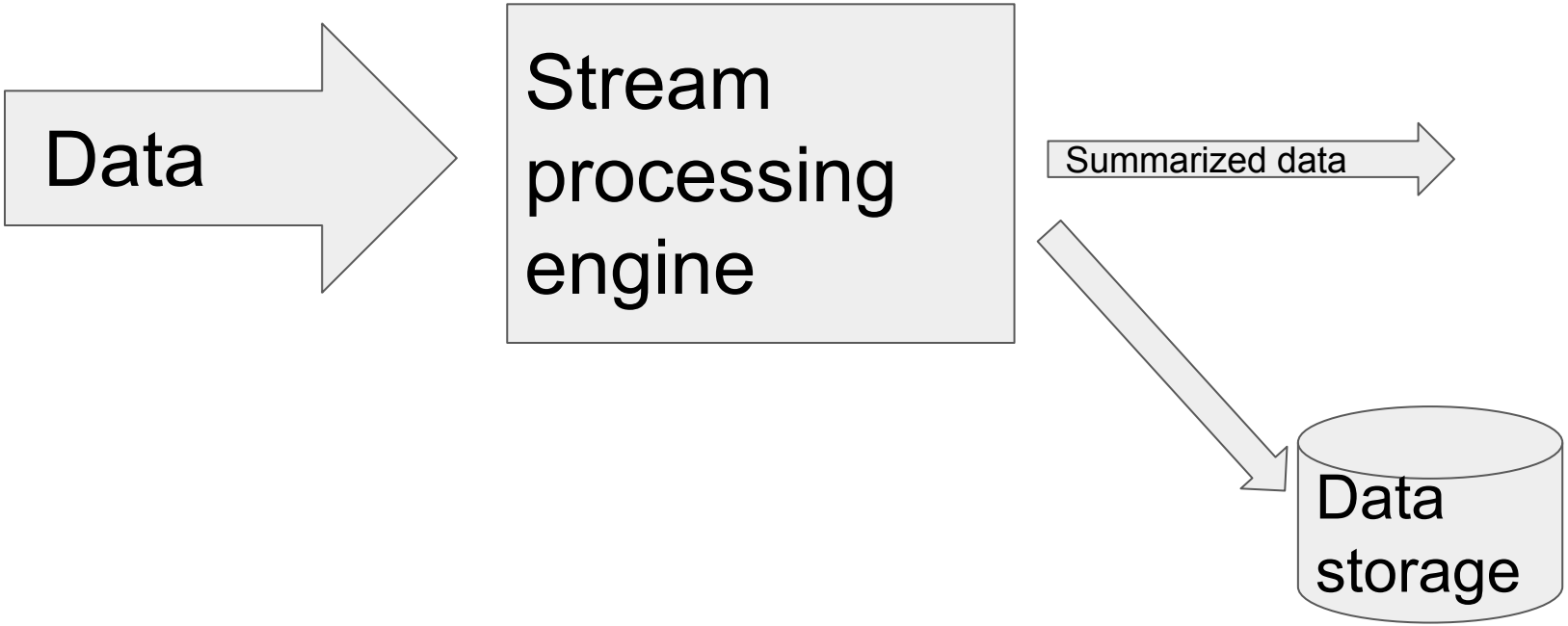
A Peek Into the World of Streaming

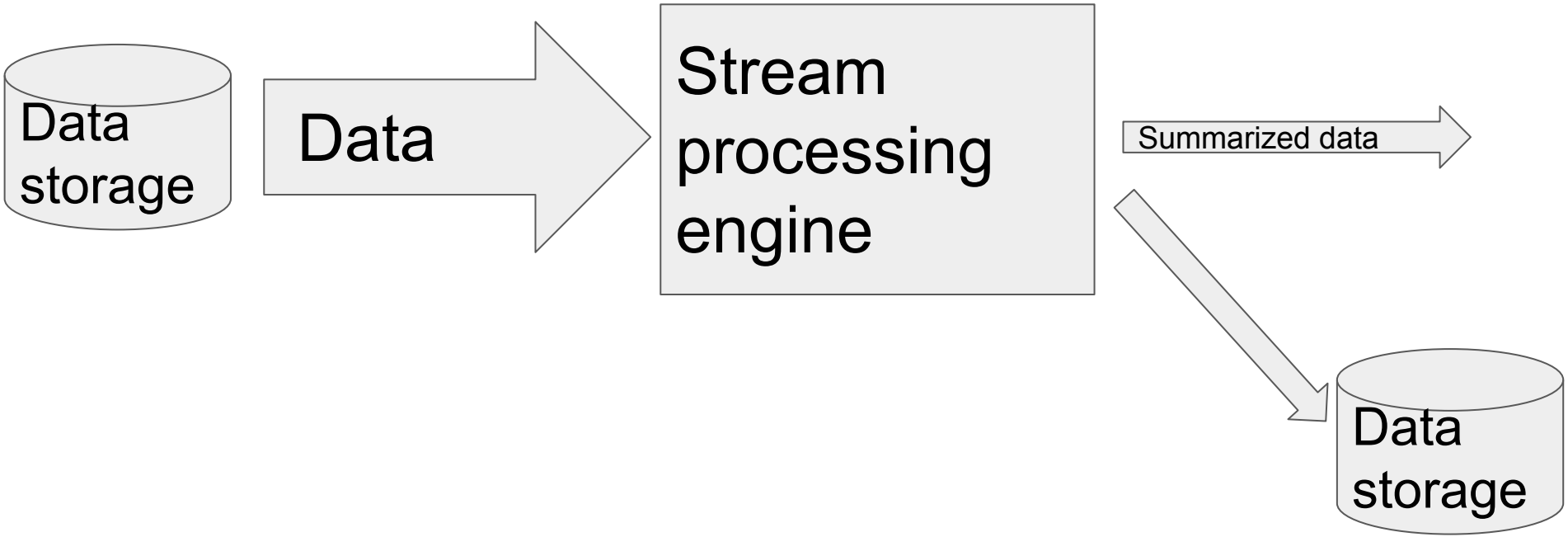
What’s Streaming?



What’s Streaming?



Funny thing: Streaming in practice often started on disk!



Outline

- Motivate streaming applications
- Apache Spark Streaming
- Dataflow/Apache Beam and Watermarks
- Apache Spark Structured Streaming and Watermarks.

Counting hashtags: batch

Input:

Timestamped twitter messages, some of them with hash tags.

Output:

For each five-minute window, the top ten hashtags along with their counts.

10:01 “I love cats #cats”
10:04 “My cat just ate a bug, gross. #cats”
10:06 “My cat is so cute! #cats”

10:00-10:05	#cats	2
10:05-10:10	#cats	1

Counting hashtags: batch, cont

- Compute the time interval the tweet falls into (eg, 10:00-10:05, or 10:05-10:10)
- reduce by a key of time-interval,hashtag

Counting hashtags: streaming

Input:

Timestamped twitter messages, some of them with hash tags.

Output:

Output the top ten hashtags along with their counts.

10:01 “I love cats #cats”
10:04 “My cat just ate a bug, gross. #cats”
10:06 “My cat is so cute! #cats”

10:00-10:05	#cats	2

10:05-10:10	#cats	1

Apache Spark Streaming

- Idea: divide input up into micro-batches

10:01 “I love cats #cats”
10:04 “My cat just ate a bug, gross. #cats”
10:06 “My cat is so cute! #cats”

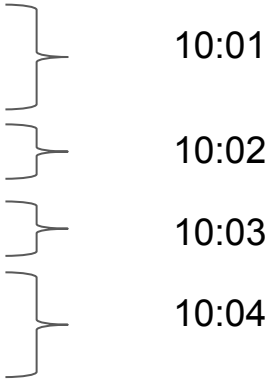
Batch 10:00-10:05

Batch 10:05-10:10

For each batch, group by the hashtag (reduceBy in Apache Spark), and perform the count. DONE.

Apache Spark Streaming: Batch boundary does not need to match aggregation boundary

10:01:30 “I love cats #cats”
10:01:55 “RT #cats are the best.”
10:02:12 “Dead mouse #cats”
10:03:52 “Live mouse. Wish I had a cat. #cat”
10:04:23 “My cat just ate a bug, gross. #cats”
10:04:44 “My #cat had kittens!”
10:06 “My cat is so cute! #cats”



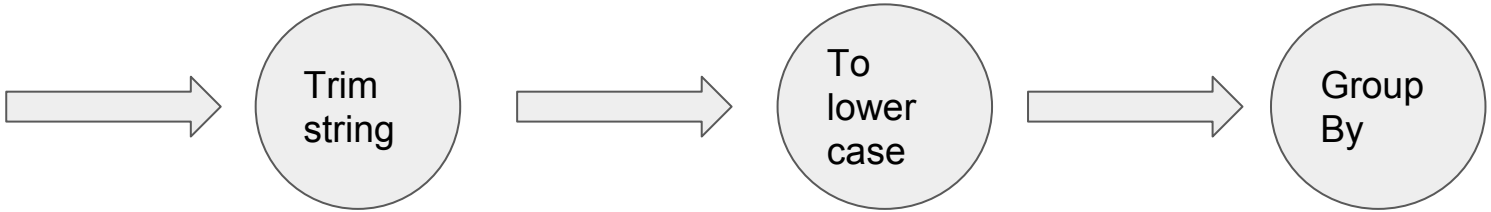
- Aggregate by batches and sum over five batches.

Dataflow/Apache Beam

- Data is in a PCollection
- Programmer provides transformations on the PCollection
- Helpers for basics like groupBy
- When done, Run!

```
Pipeline myPipeline = Pipeline.create(options)
PCollection<String> inputdata = myPipeline.apply(/* read the data */)  // Nothing
PCollection<String> foo =inputdata.apply(...).apply(...);  // Nothing happens.
//… more stuff ...
myPipeline.run(); // Actually does something
```

Dataflow: Program vs execution may be different

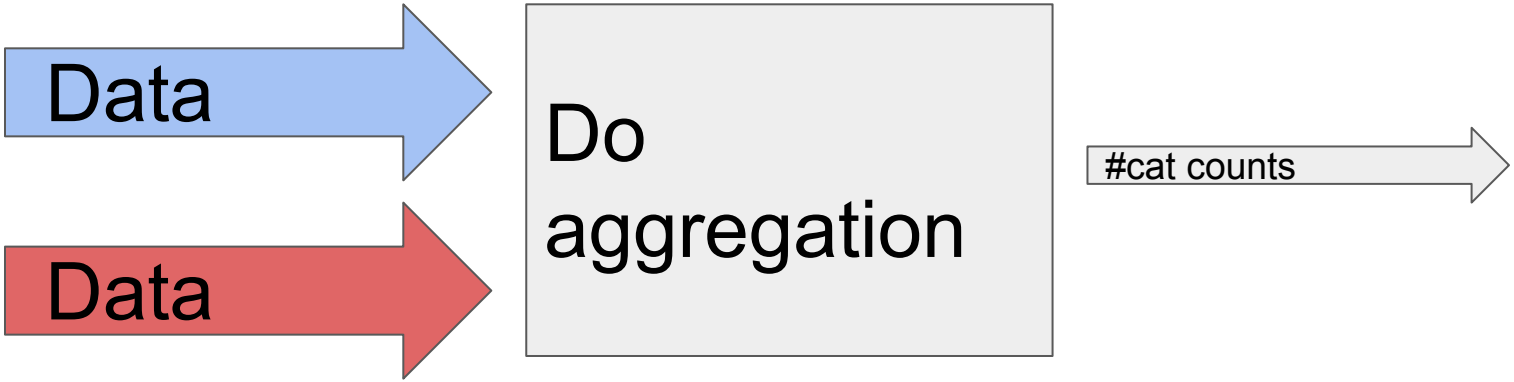


- Dataflow program is given all operations “in advance” and may re-order and rearrange
- Execution engine essentially “pluggable” because what is provided is a program description.
- Program can be batch or streaming
- Uses a **watermark** in streaming applications

Back to Counting #cats

Or, why watermark is useful

What problems can arise here?



Counting hashtags: real world

Data as seen by the stream processor:

10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”
10:04 “My cat just ate a bug, gross. #cats”

Really common!

Question: What’s the count for 10:00-10:05? When do we output it?

10:00-10:05	#cats	?

10:05-10:10	#cats	?

Counting hashtags: real world

Data as seen by the stream processor:

10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”
10:04 “My cat just ate a bug, gross. #cats”

Really common!

Question: What’s the count for 10:00-10:05? When do we output it?

10:00-10:05	#cats	2

10:00-10:05	#cats	1

Counting hashtags: real world

Case 1:

10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”
10:04 “My cat just ate a bug, gross. #cats”

Case 2:
10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”
… three hours later ...
10:04 “My cat just ate a bug, gross. #cats”

Question: What’s the count for 10:00-10:05? When do we output it?

What to do with out-of-order data?

What to do with out-of-order data?

- Discard anything earlier than what's already been seen
  - One early data item, and you miss a lot!
  - You could miss an entire slow-to-arrive source!
- When nothing from 10:00-10:05 has been seen for x minutes
- Depends on data input source
  - Maybe the source has some idea of how out-of-order data can be
- Special business logic:
  - When fewer than x things seen and ….

Question: How do we program this?

Watermark: Data is complete up until this point

- Watermark is X → all data earlier than X has arrived
- In our example: Watermark is 10:05 → all data up until 10:05 has arrived → we can output the #cat count
- **Doneness for a Stream!**
- Refinement 1: Early outputs
- Refinement 2: Late data treatment: drop it all, allow it all, drop it if later than some time.

Dataflow with watermarks

- Runtime tracks the watermark (with potential source-specific logical)
- API provides way to specify whether and when to output before watermark is reached
- API provides way to specify whether and when to output after watermark is reached

To see how much of a difference that can make:
https://cloud.google.com/dataflow/blog/dataflow-beam-and-spark-comparison
(google spark vs dataflow)

Apache Spark 2.0 structured streaming

Incorporates idea of a watermark. It maintains a table of output results, and updates them as data is processed.

10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”

10:00-10:05	#cats	1
10:05-10:10	#cats	1

Apache Spark 2.0 structured streaming

Incorporates idea of a watermark. It maintains a table of output results, and updates them as data is processed.

10:01 “I love cats #cats”
10:06 “My cat is so cute! #cats”
10:04 “My cat just ate a bug, gross. #cats”

10:00-10:05	#cats	2
10:05-10:10	#cats	1

Questions?