Data Streaming

- Many applications must process large streams of live data and provide results in near real time.
  - IoT data with sensors
  - Social Network trends
  - website statistics
  - monitoring
- We do not know the entire data set in advance
  - Data is generated and ingested continuously
- Think of data as infinite and non-stationary(the distribution changes over time)

Applications

- Mining query streams (which queries are more frequent today than yesterdays)
- Mining click streams (which of its pages are getting unusual number of hits in the past hour)
- Mining social network new feeds (look for trending topics)
- IP packets monitored at a switch (information for optimal routing, detect denial of service attacks)

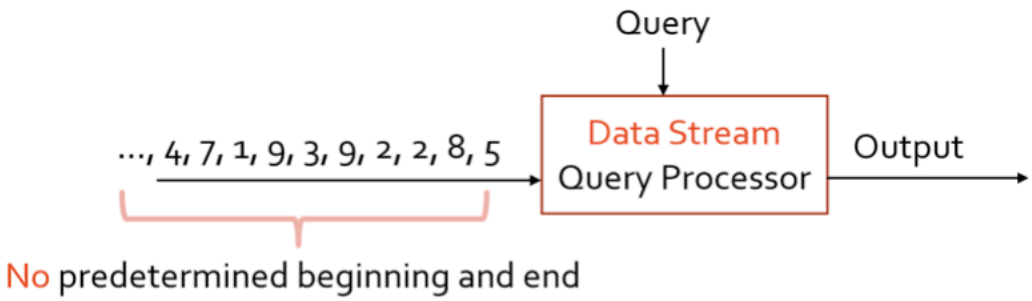
STREAM vs BATCH

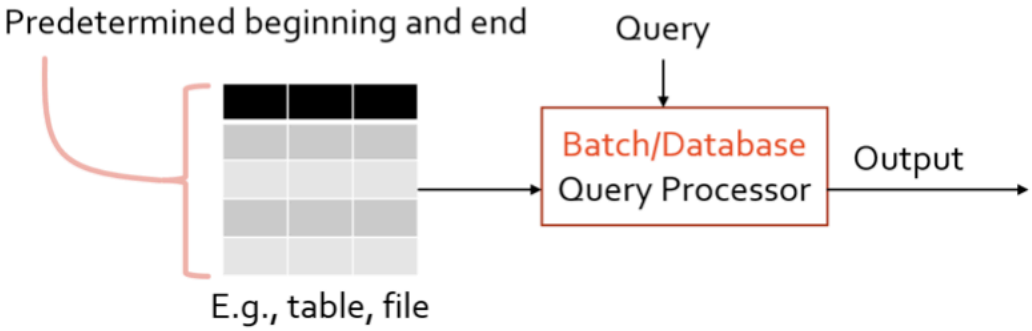
Batch processing:

- see the entire data in advance
- able to store all data

Stream processing:

- don't see the entire data in advance
- can't store all the data





Example streaming K-largest elements:

- Suppose we have a stream (infinite) of integers, how do we find the largest k integers so far?

Constraints:

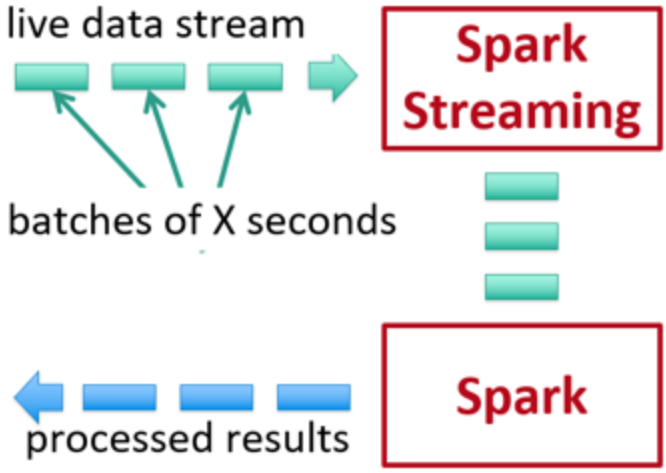
- cannot store all the elements
- can only look once
- use O(k) space

Main idea:

- use a min heap of size k
- top element is the smallest and it represents the kth largest element seen so far
- the heap stores the answer
- scan every element in the stream
- if its smaller than the top, we discard it
- otherwise
  - push e to the heap, will automatically do heap adjustment
  - delete the top, bingo! heap is updated
- time complexity is $nlog(k)$

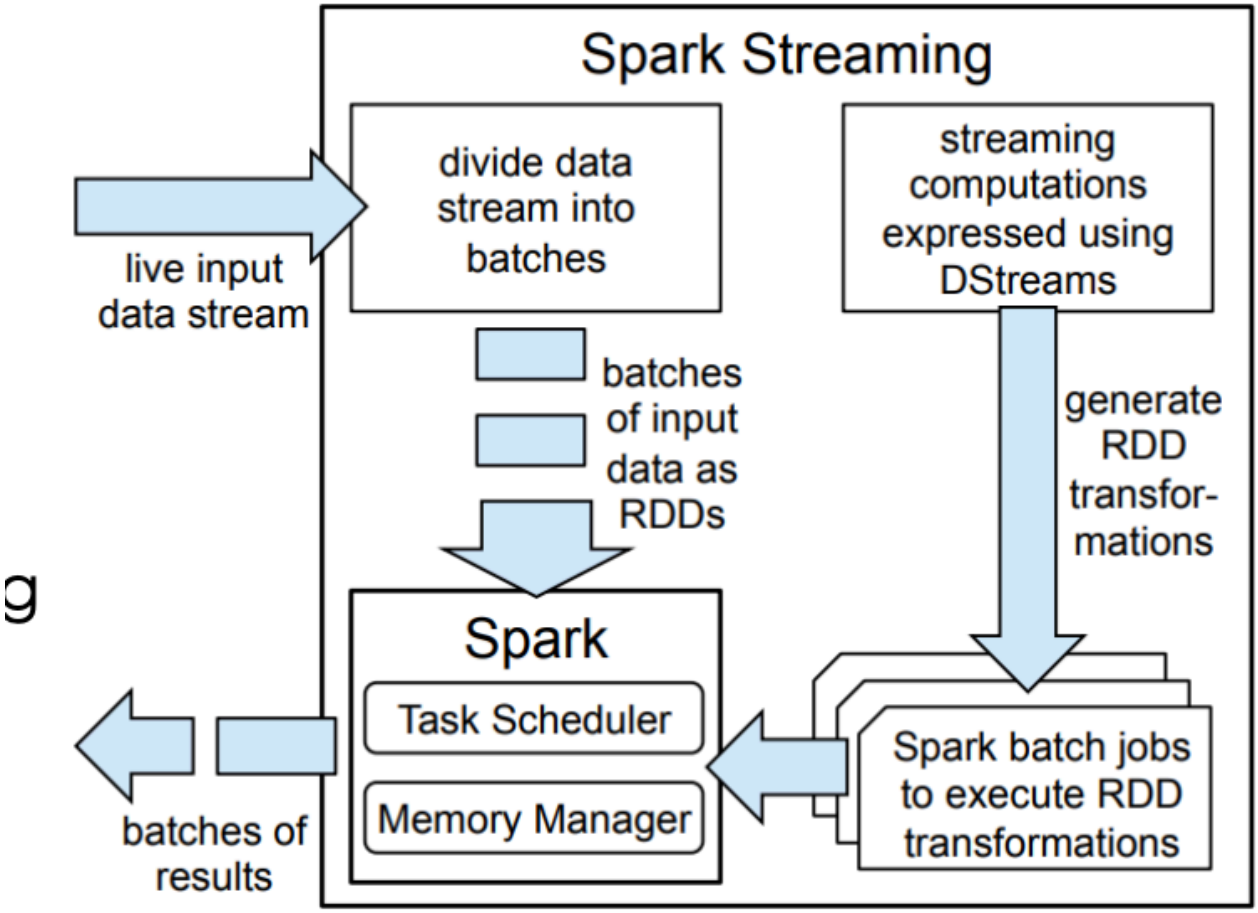
Spark streaming

- spark is a batch processing system
- main idea: discretized stream
  - run a streaming computation as a series of very small stateless deterministic batch jobs
  - chop the incoming data into intervals of X seconds (1 second)
  - run spark within each interval -> batch processing within each interval (using spark RDD)
  - final result: can be returned across different intervals



- Spark streaming divides input data streams into batches and stores them in Spark's memory

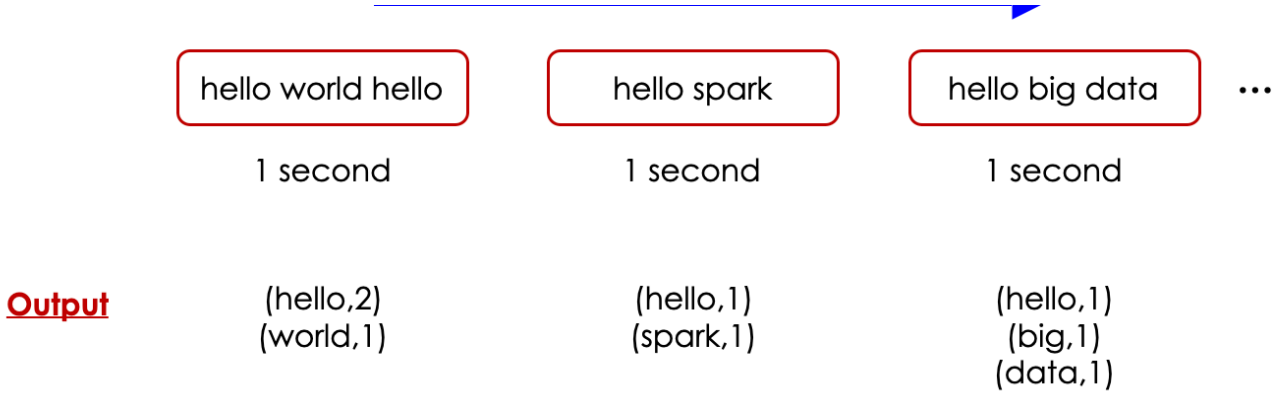
- It then executes a streaming application by generating Spark jobs to process the batches



DStream (discretized streams)

- Sequence of RDDs representing a stream of data
- DStream is a sequence of immutable, partitioned datasets (RDDs) that can be acted on by deterministic transformations.
- These transformations yield new D-Streams and might create an intermediate state in the form of RDDs
  Note there are stateless RDDs (default) and stateful RDDs

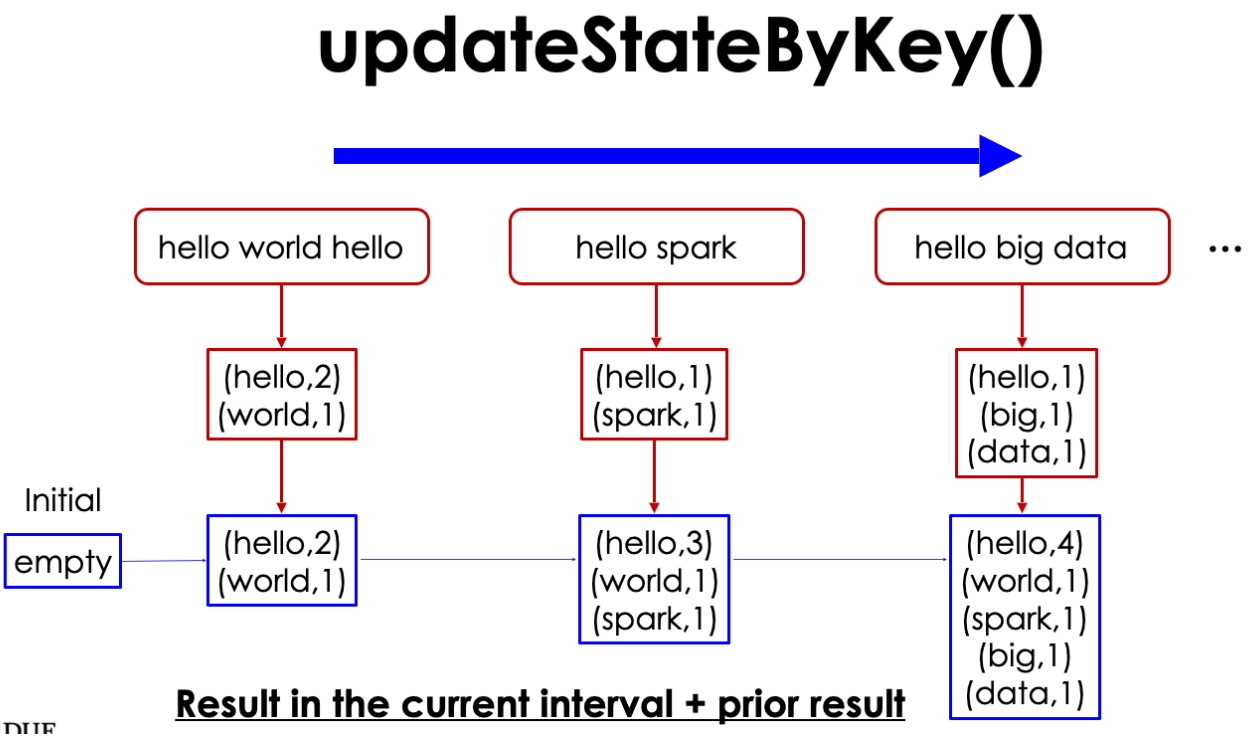
Streaming Word Count:



DStream Operations

- Stateless transformation
  - Only show the results within current time interval
  - word count of the current interval
  - API's that we've learned in spark core: map(), flatMap(), filter(), etc
- Stateful transformation
  - also consider the results of prior intervals
  - word count of the words received so far
  - word count in the last 10 seconds

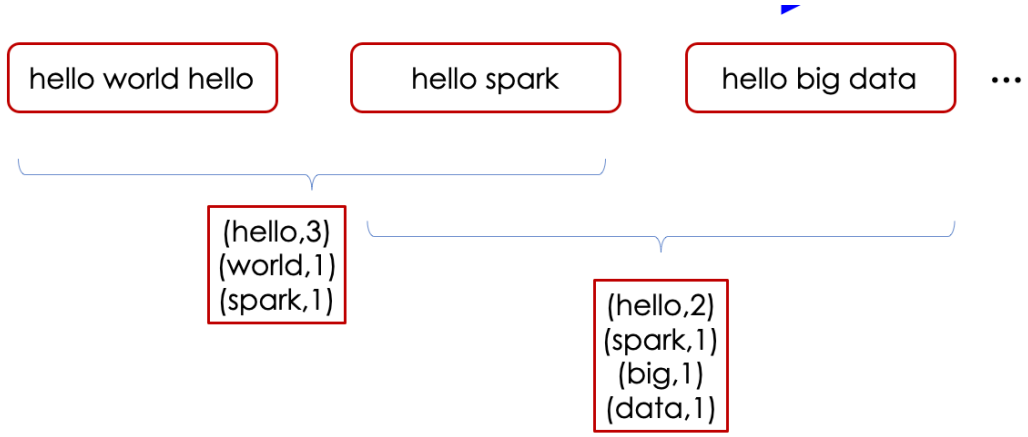
  Stateful Operations:
- update StateByKey()
  - compute the result based on all the history data received so far
  - need to specify an update function of how to change the status
- window operation
  - compute the results in a specified moving window (last 10 seconds)



Window Operations:

- compute the result of the last time window
  - window length: multiple units of a time interval
    - if the time interval is 3 seconds, then window size can be 3sec,6sec,9sec, etc
  - sliding interval: time to trigger computation: multiple units of a time interval:

- specify when to compute the results
- it's set to be 3 seconds, then it'll show the results at the end of each time interval



reduceByKeyAndWindow():

- computes the result in the current window instead of the current interval
-