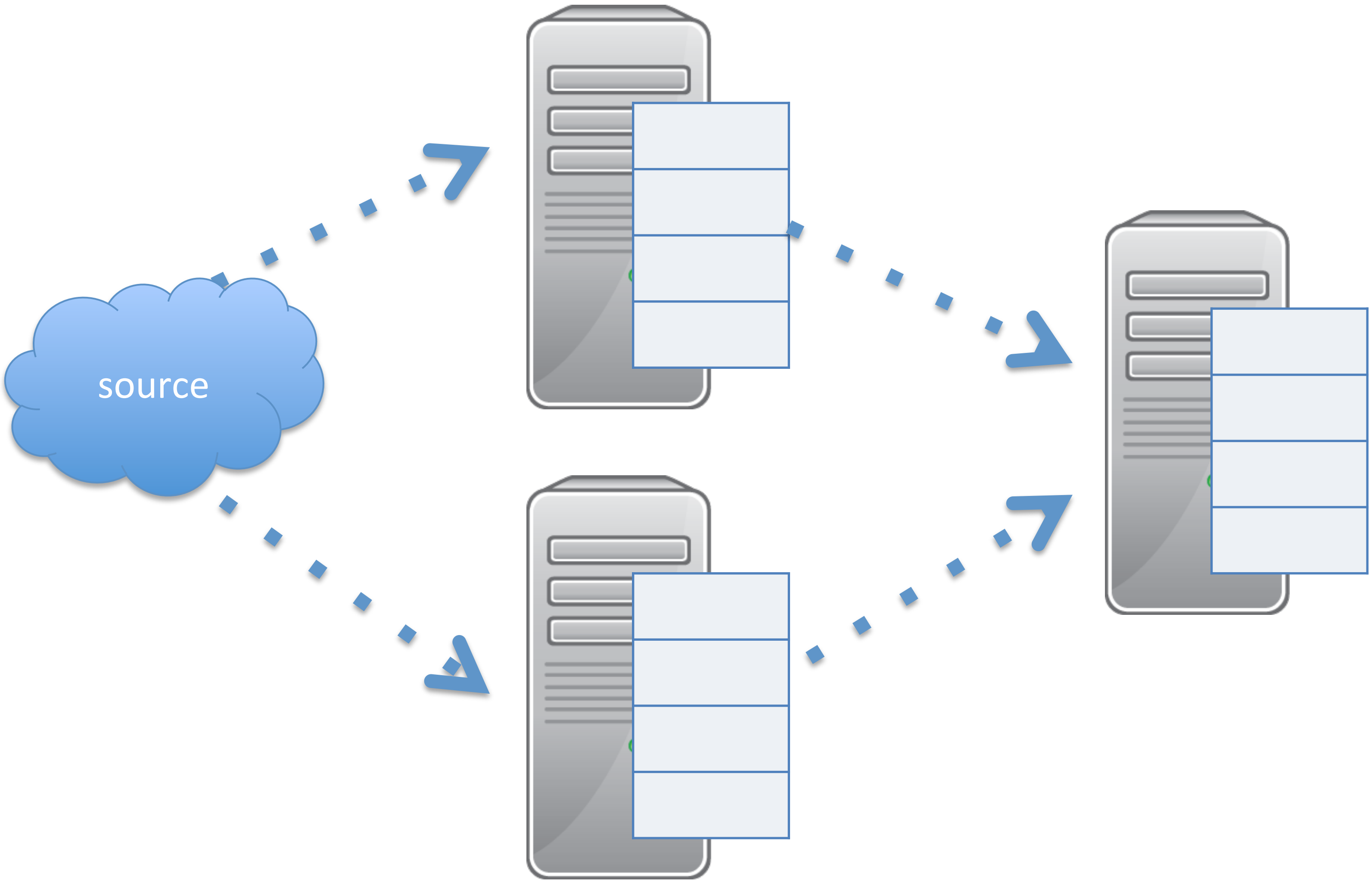
Spark Streaming

Summary by Lucy Yu

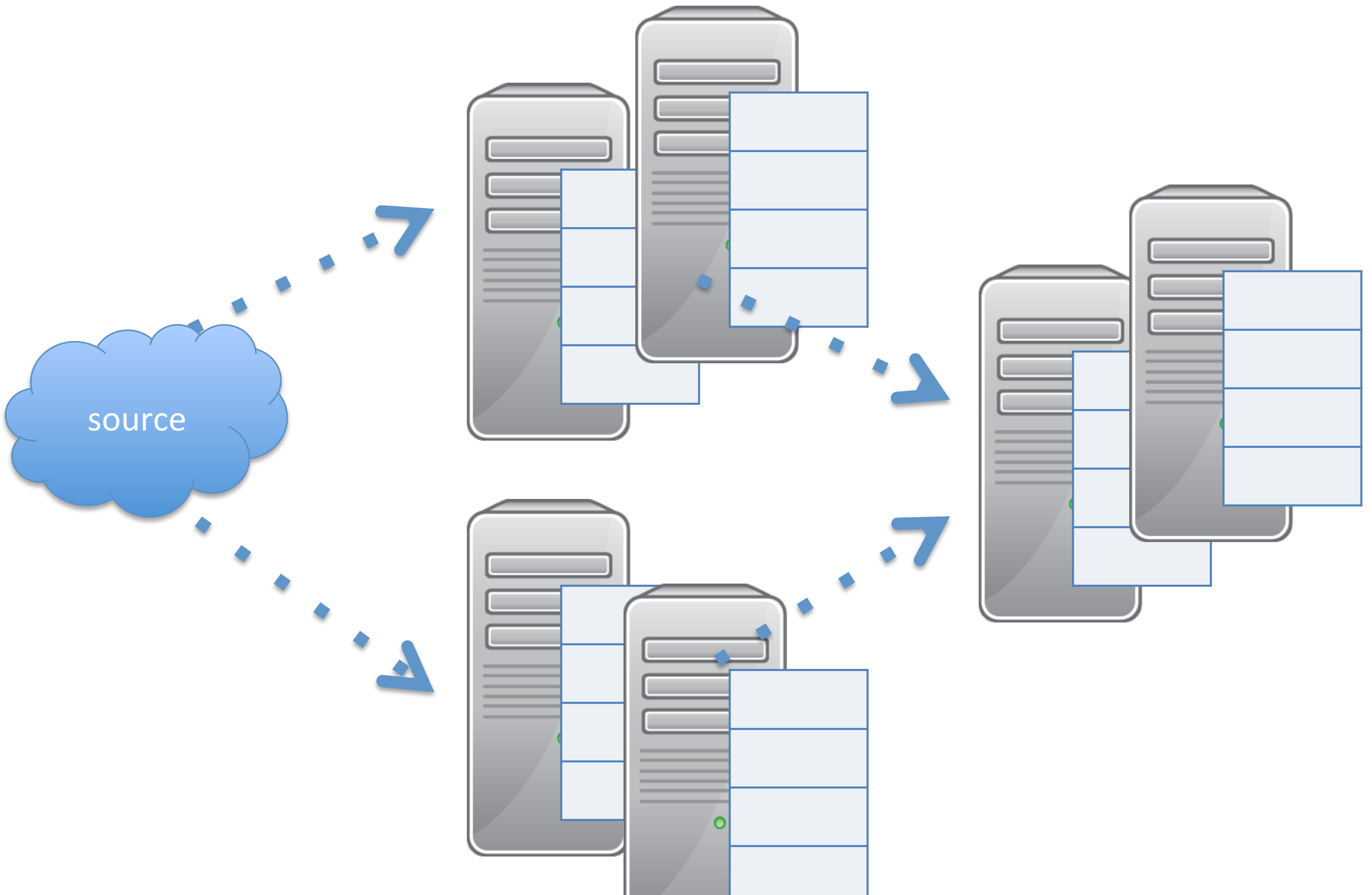
Motivation

- Most of “big data” happens in a streaming context
  - Network monitoring, real-time fraud detection, algorithmic trading, risk management
- Current Model: Continuous Operator Model
  - Fault tolerance achieved via replication or upstream backup

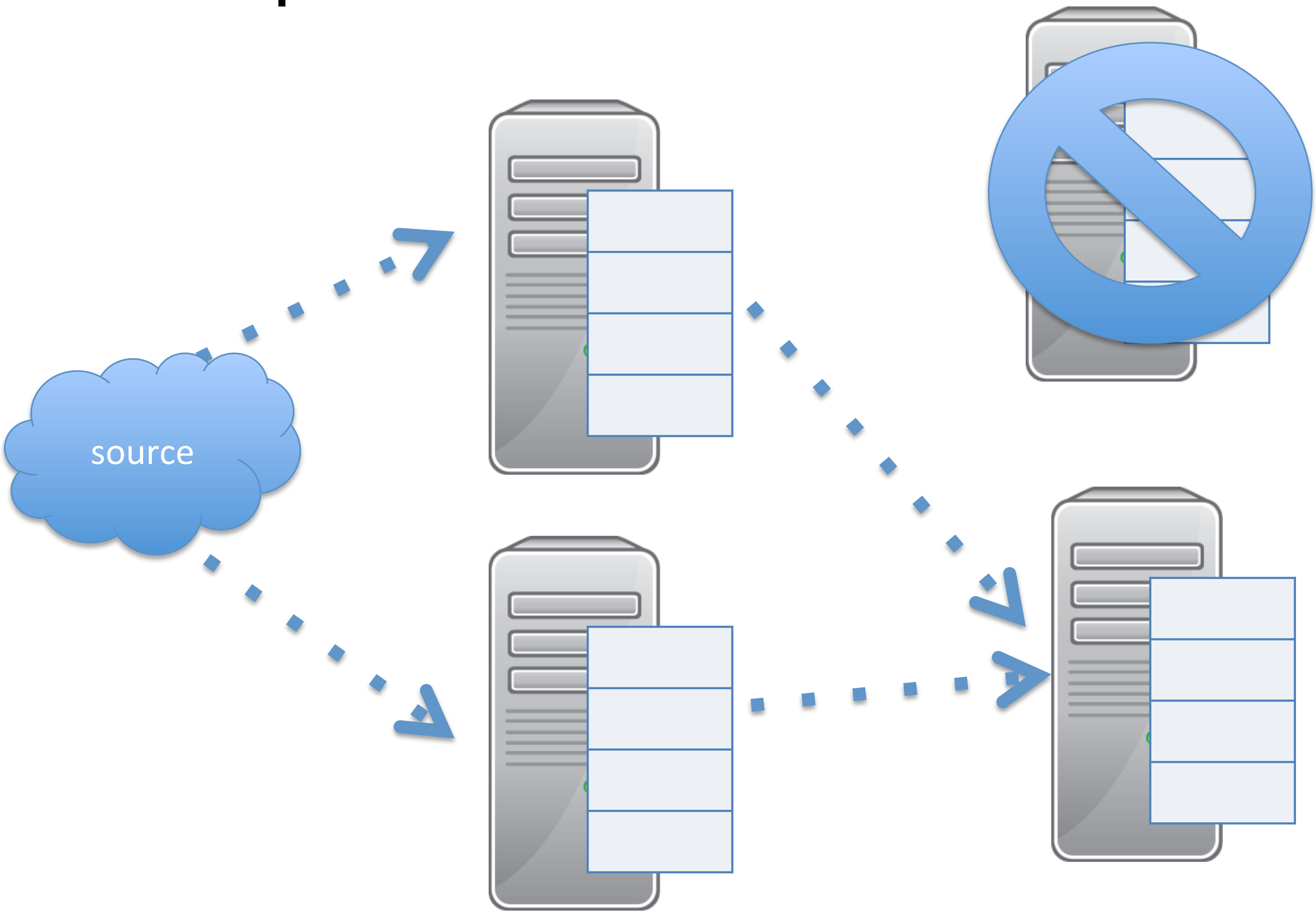
Motivation



Replication



Upstream Backup

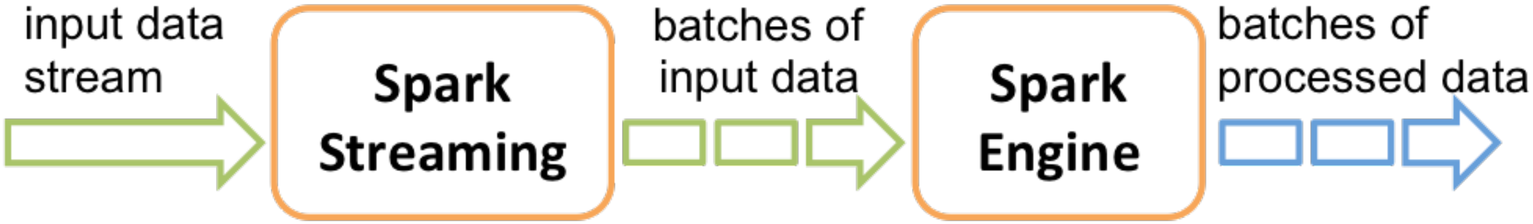


D-Streams

- "Instead of managing long-lived operators, the idea …is to structure a streaming computation as a series of stateless, deterministic batch computations on small time intervals."

- Use a data structure: Resilient Distributed Datasets (RDDs)
  - keeps data in memory
  - can recover it without replication (track the lineage graph of operations that were used to build it)

D-Streams

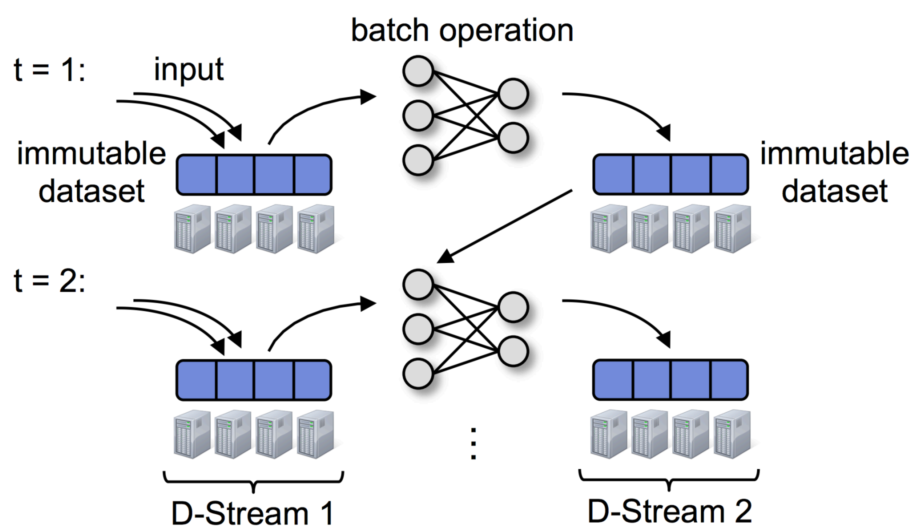


https://spark.apache.org/docs/1.2.0/streaming-programming-guide.html

D-Streams

- The data received in each interval stored reliable across the cluster to form an **input dataset** for that interval

- Do batch operation to get another RDD, which acts as a state or output



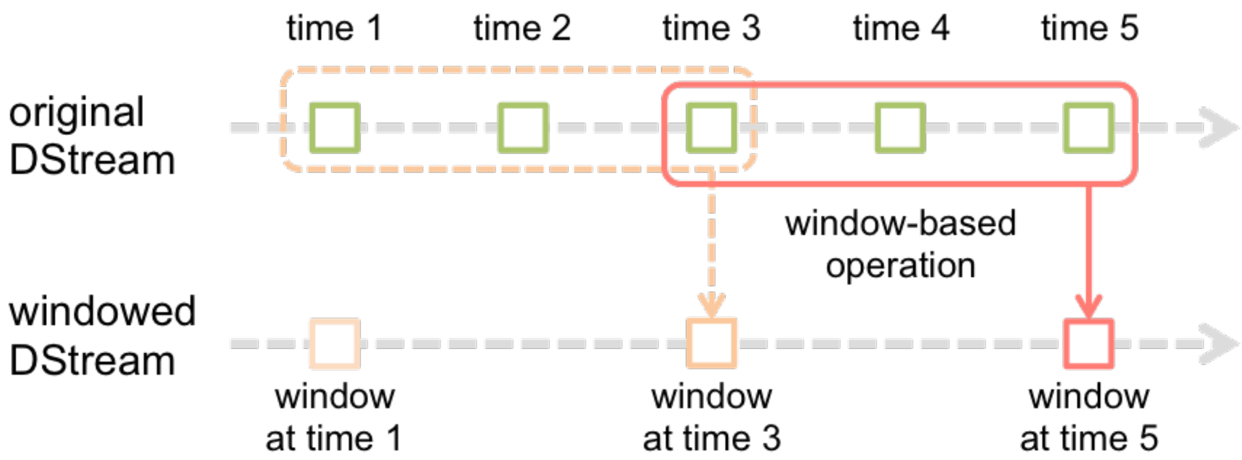
D-Stream API

- Users register one or more streams using a functional API
- **Transformations** create a new D-Stream from parent stream(s)
  - Stateless: map, reduce, groupBy, join
  - Stateful: window operations
- **Output operations** let the program write data to external systems (e.g. save)

Transformations

map(func)	Return a new DStream by passing each element of the source DStream through a function func.
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items.
filter(func)	Return a new DStream by selecting only the records of the source DStream on which func returns true.
reduce(func)	Return a new DStream of single-element RDDs by aggregating the elements in each RDD of the source DStream using a function func (which takes two arguments and returns one). The function should be associative so that it can be computed in parallel.
updateStateByKey(func)	Return a new "state" DStream where the state for each key is updated by applying the given function on the previous state of the key and the new values for the key.

Window Operations



https://spark.apache.org/docs/1.2.0/streaming-programming-guide.html

Window Operations

window(windowLength, slideInterval)	Return a new DStream which is computed based on windowed batches of the source DStream.
countByWindow(windowLength, slideInterval)	Return a sliding window count of elements in the stream.
reduceByWindow(func, windowLength, slideInterval)	Return a new single-element stream, created by aggregating elements in the stream over a sliding interval using func. The function should be associative so that it can be computed correctly in parallel.
reduceByKeyAndWindow(func, windowLength, slideInterval, [numTasks])	When called on a DStream of (K, V) pairs, returns a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function func over batches in a sliding window.
reduceByKeyAndWindow(func, invFunc, windowLength, slideInterval, [numTasks])	A more efficient version of the above reduceByKeyAndWindow() where the reduce value of each window is calculated incrementally using the reduce values of the previous window.

Fault Recovery

- Parallel recovery of a lost node's state.
  - When a node fails, each node in the cluster works to recompute part of the lost node's RDDs, resulting in significantly faster recovery than upstream backup without the cost of replication.

- In a similar way, D-Streams can recover from stragglers using speculative execution

- Checkpoint state RDDs periodically