Streaming Data Analytics with Apache Spark Streaming

Suresh Matlapudi
Suresh.Matlapudi@ibm.com

Agenda

- Overview
- Architecture and Execution Model
- Spark Streaming I/O
- Streaming Operations (api)
- Fault tolerance and reliability
- Performance Considerations

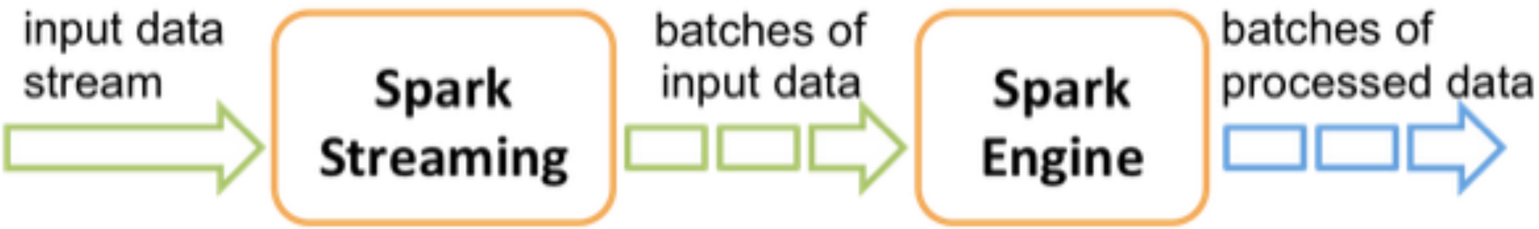
- Code

Spark Streaming

- Component of Spark
  - Project started in 2012

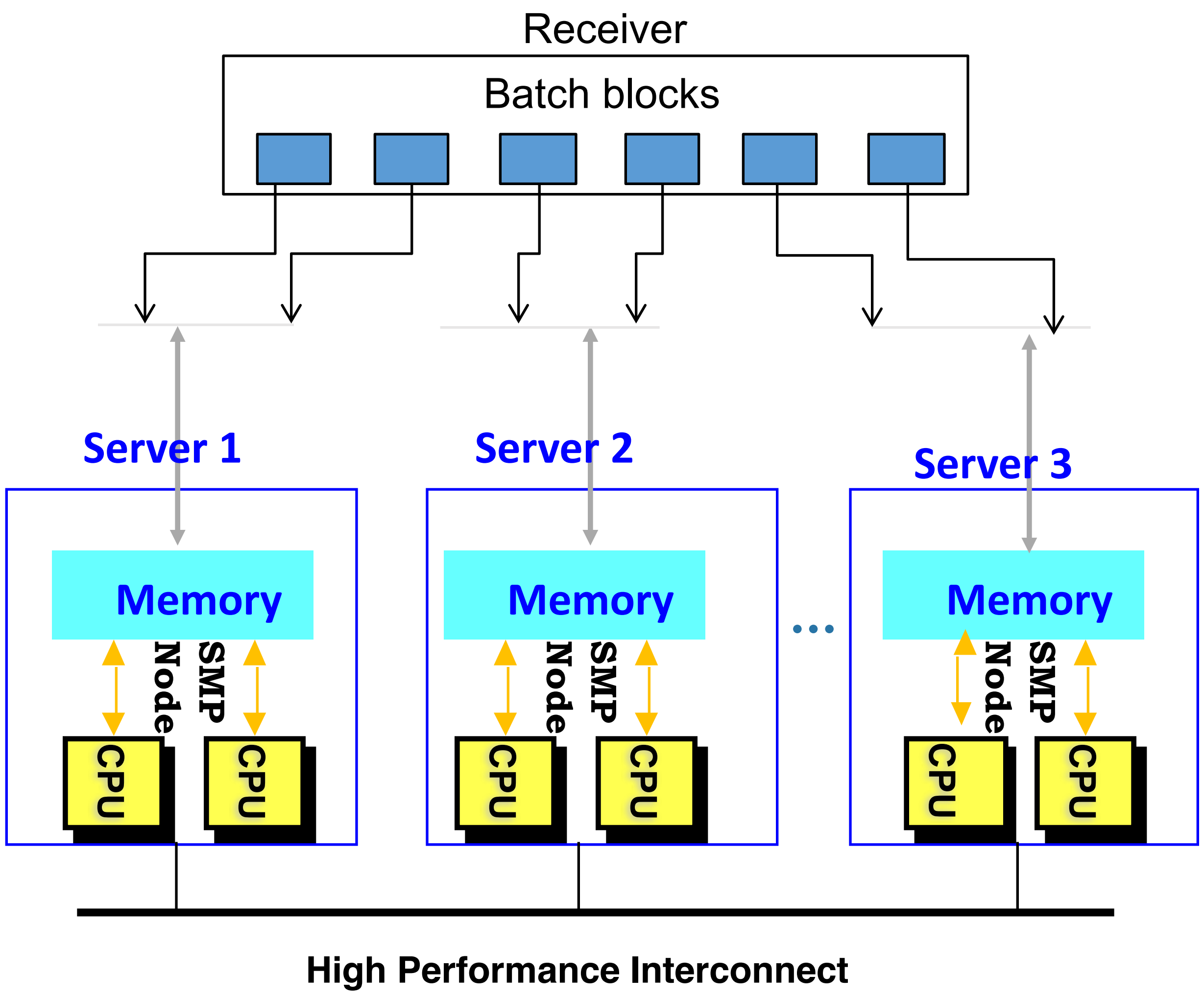
- Discretized Stream (DStream) programming abstraction
  - Represented as a sequence of RDDs (micro-batches)
  - RDD: set of records for a specific time interval
  - Supports Scala, Java, and Python

- Fundamental architecture: batch processing of datasets

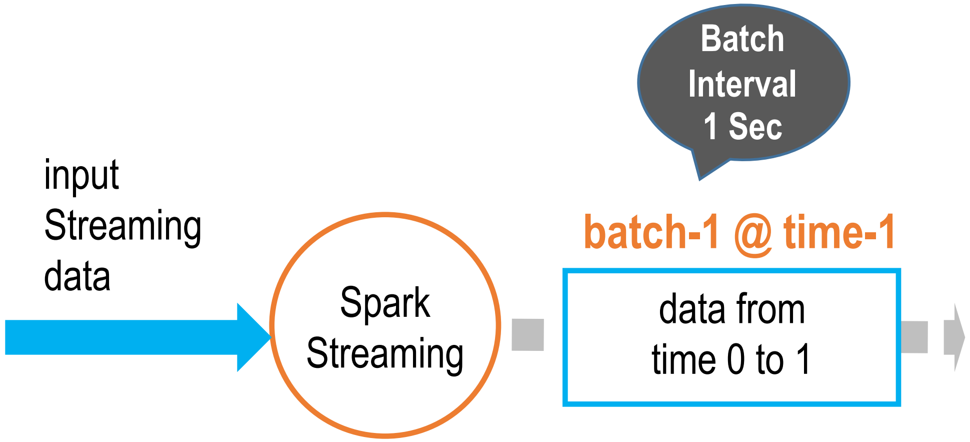


Spark Streaming Architecture

- Micro batch architecture.
- Operates on interval of time
- New batches are created at regular time intervals.
- Divides received time batch into blocks for parallelism
- Each batch is a graph that translates into multiple jobs
- Has the ability to create larger size batch window as it processes over time

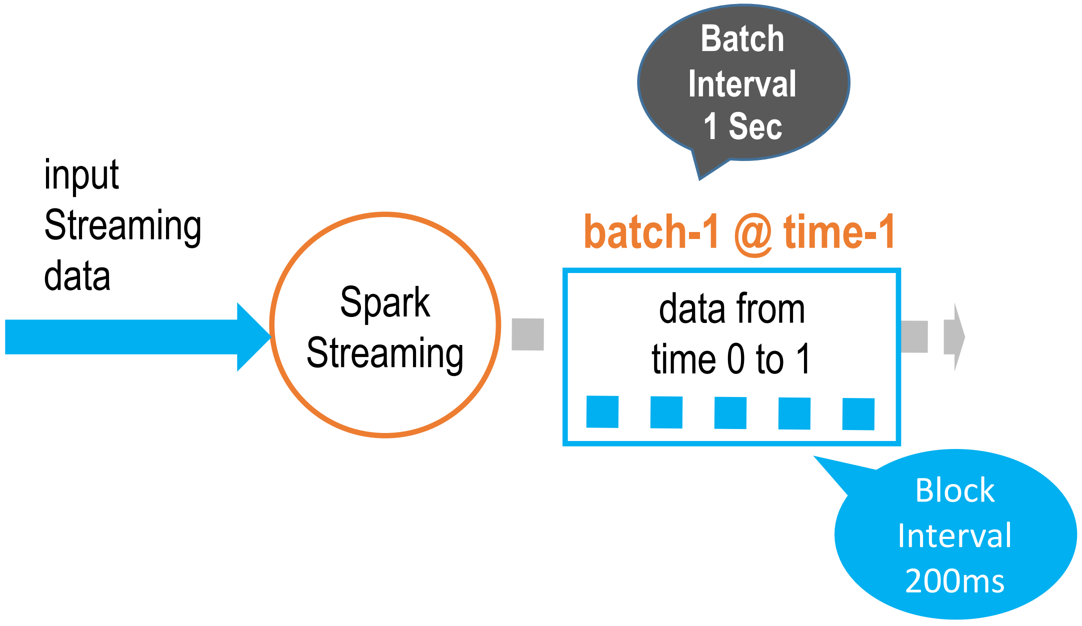


Spark Streaming – DStreams, Batches and RDDs



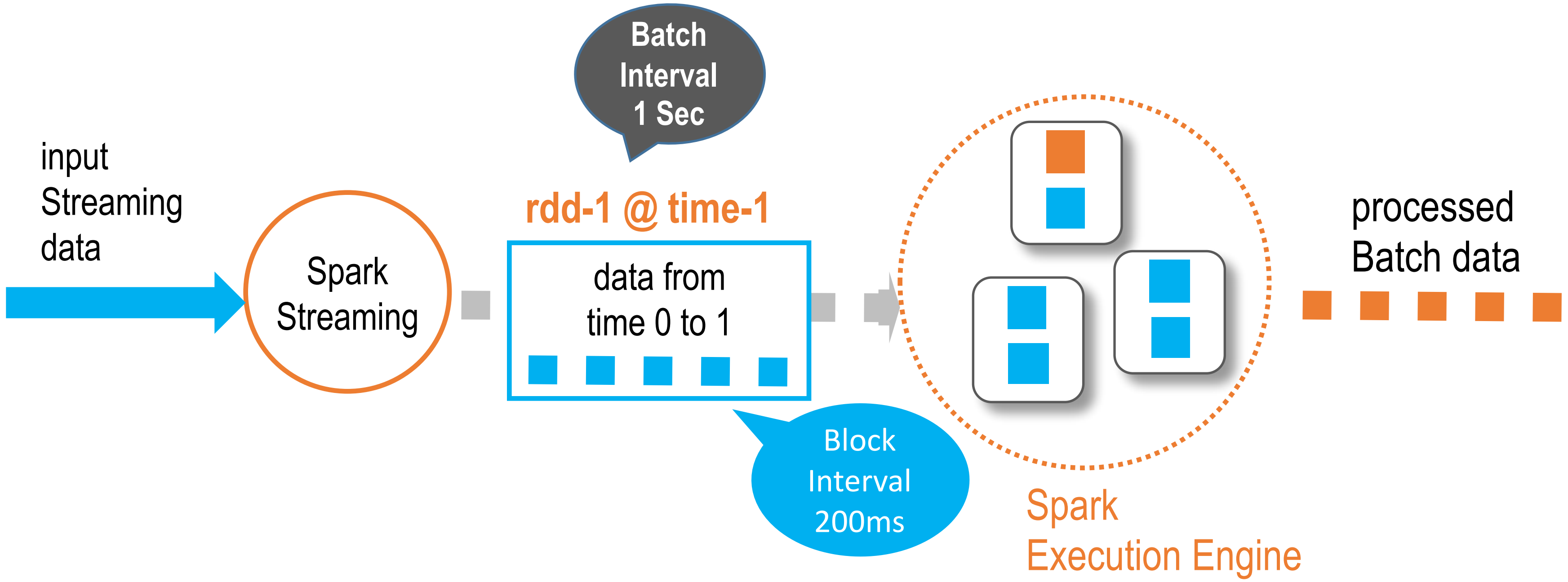
- A receiver thread collects data coming from a streaming source for a “batch” interval.

Spark Streaming – DStreams, Batches and RDDs



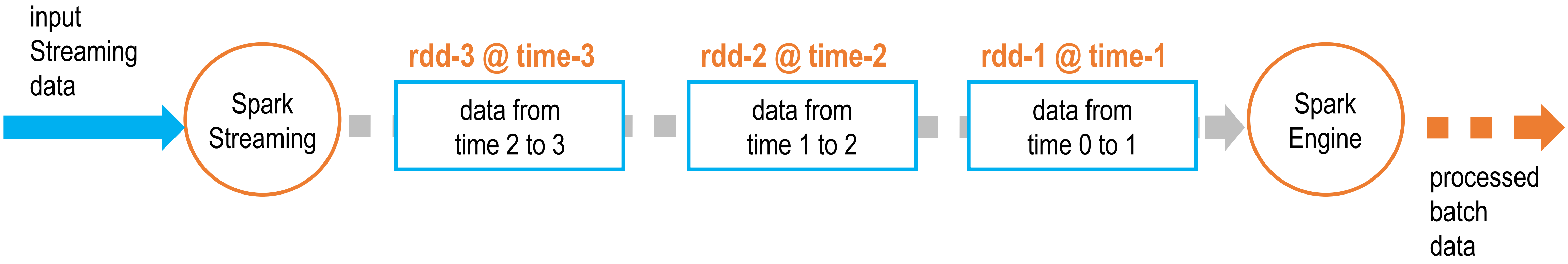
- A receiver thread collects data coming from a "batch" interval.
- It also has the ability to subdivide the batch into multiple blocks so they could be sent to multiple machines for parallel processing.
- Blocks are also duplicated for HA purpose

Spark Streaming – DStreams, Batches and RDDs



- A receiver thread collects data coming from a streaming source for a “batch” interval.
- It also has the ability to subdivide the batch into multiple blocks so they could be sent to multiple machines for parallel processing.
- Blocks are also duplicated for HA purpose
- Once a batch is assembled, It constitute the equivalent of a Spark RDD where each partition of the RDD can be processed in parallel.

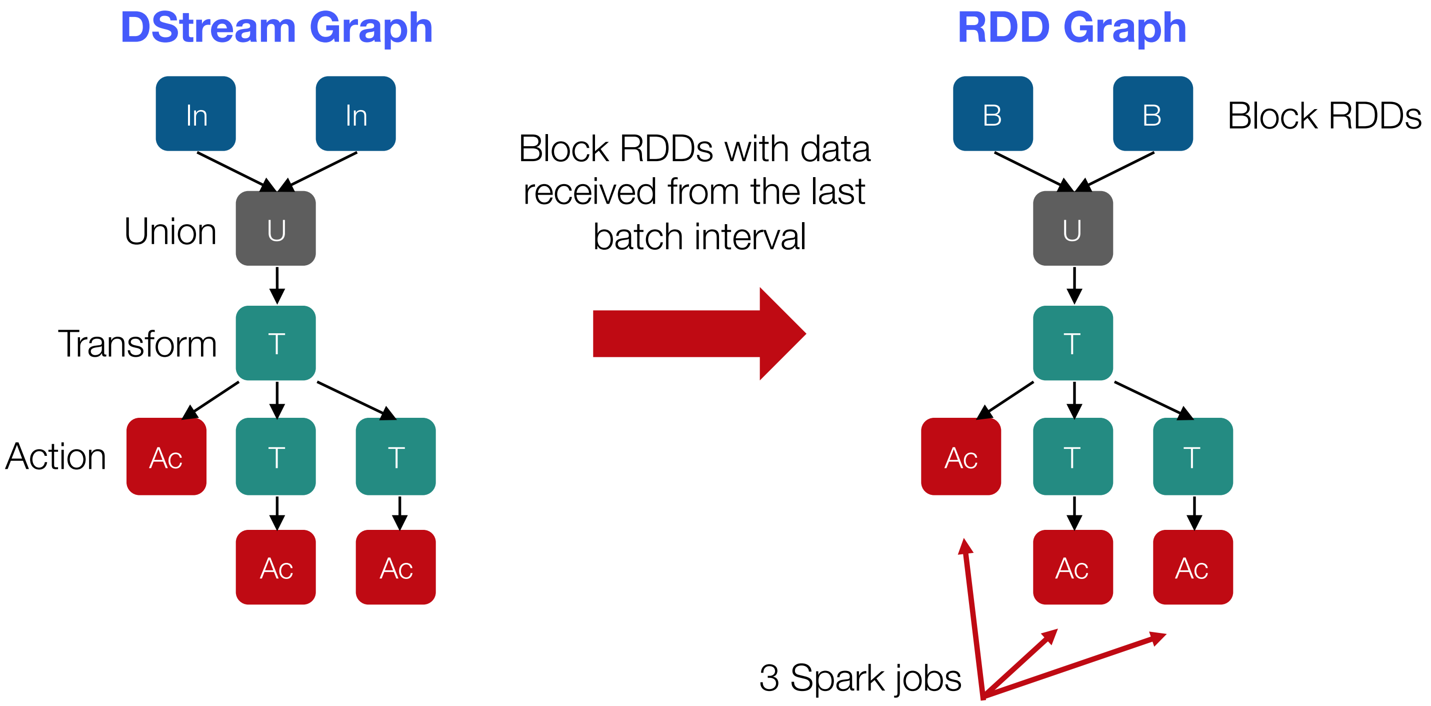
Spark Streaming – DStreams, Batches and RDDs



- These steps repeat for each batch.. Continuously
- Because we are dealing with Streaming data. Spark Streaming has the ability to "remember" the previous RDDs… to some extent.
- More of this this windowing system in following slide.

From DStreams to Spark Jobs

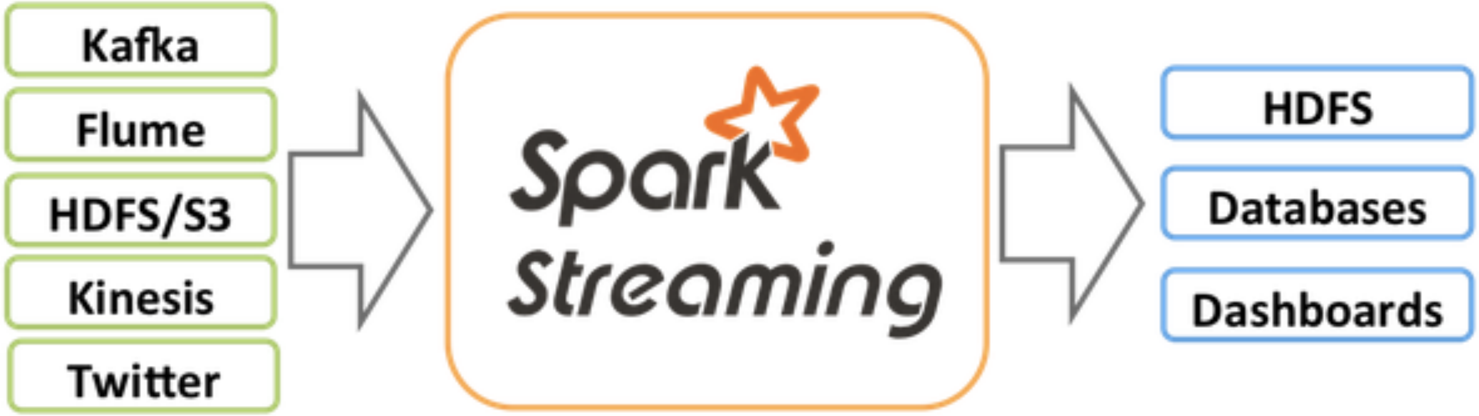
- Every interval, an RDD graph is computed from the DStream graph
- For each output operation, a Spark action is created
- For each action, a Spark job is created to compute it



Current Spark Streaming I/O

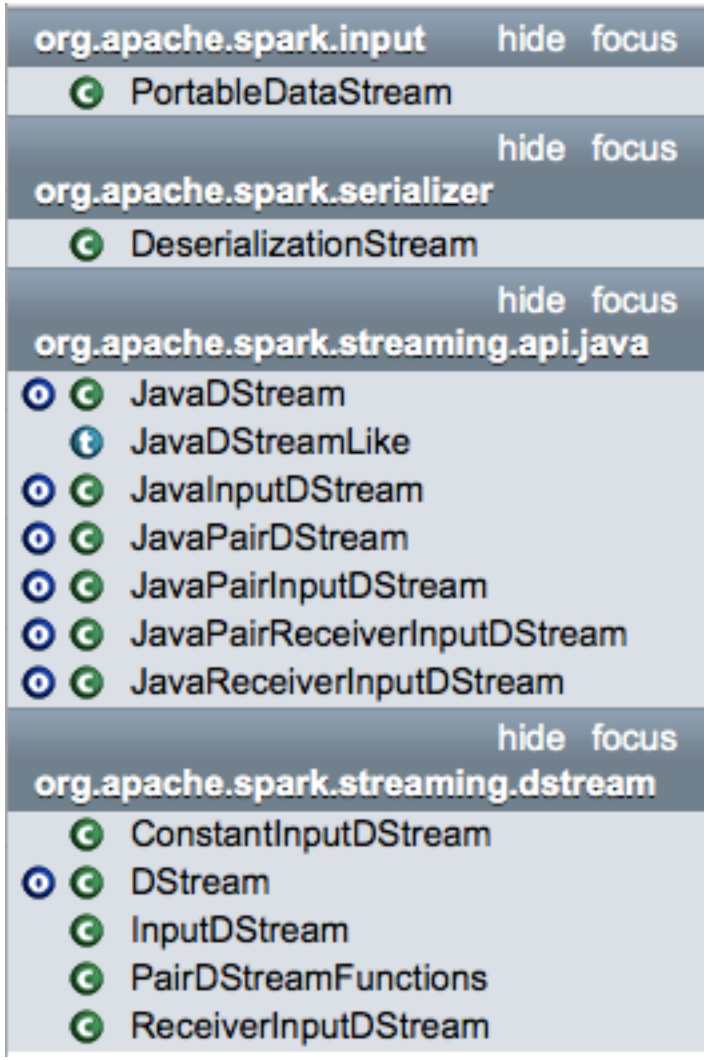
- Input Sources
  - Kafka, Flume, Twitter, ZeroMQ, MQTT, TCP sockets
  - Basic sources: sockets, files, Akka actors
  - Other sources require receiver threads

- Output operations
  - Print(), saveAsTextFiles(), saveAsObjectFiles(), saveAsHadoopFiles(), foreachRDD()
  - foreachRDD can be used for message queues, DB operations and more



DStream Classes

- Different classes for different languages (Scala, Java)
  - DStream has 36 value members
- Multiple types of DStreams
- Separate Python API



org.apache.spark.input hide focus

PortableDataStream

hide focus
org.apache.spark.serializer

DeserializationStream

hide focus
org.apache.spark.streaming.api.java

JavaDStream
JavaDStreamLike
JavaInputDStream
JavaPairDStream
JavaPairInputDStream
JavaPairReceiverInputDStream
JavaReceiverInputDStream

hide focus
org.apache.spark.streaming.dstream

ConstantInputDStream
DStream
InputDStream
PairDStreamFunctions
ReceiverInputDStream

Spark Streaming Operations Available

- All the Spark RDD operations
  - Some available through the transform() operation

map/flatmap	filter	repartition	union
count	reduce	countByValue	reduceByKey
join	cogroup	transform	updateStateByKey

- Spark Streaming window operations

window	countByWindow	reduceByWindow
reduceByKeyAndWindow	countByValueAndWindow	

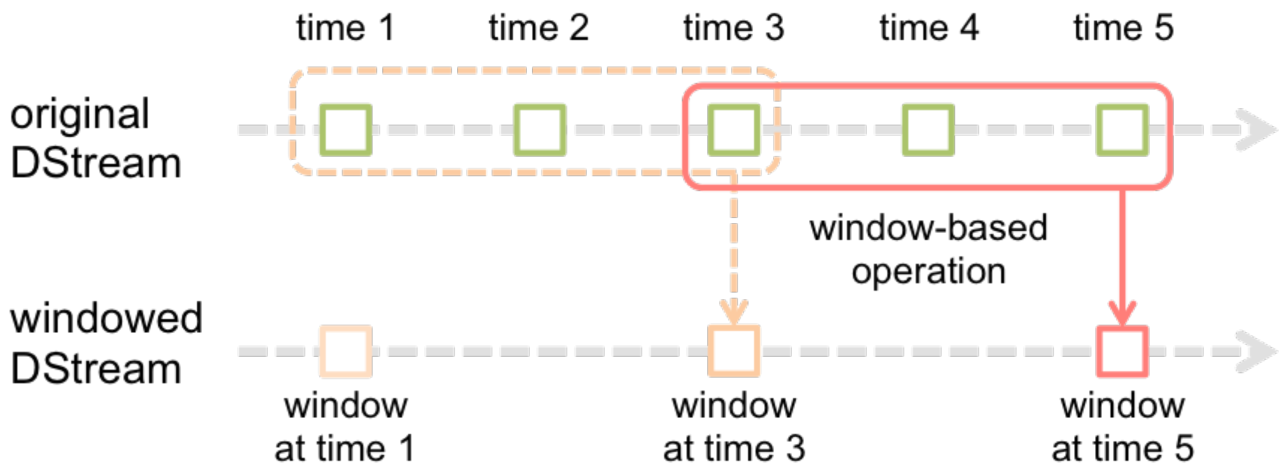
- Spark Streaming output operations

print	saveAsTextFiles	saveAsObjectFiles
saveAsHadoopFiles	foreachRDD	

Spark Streaming Windowing Capabilities

- Parameters
  - Window length: duration of the window
  - Sliding interval: interval at which the window operation is performed
  - Both parameters must be a multiple of the batch interval

- A window creates a new DStream with a larger batch size



Fault Tolerance

- Received data is replicated among multiple Spark executors
  - Default factor: 2

- Checkpointing
  - Saves state on regular basis, typically every 5-10 batches of data
  - A failure would have to replay the 5-10 previous batched to recreate the appropriate RDDs
  - Checkpoint done to HDFS or equivalent

- Must protect the driver program
  - If the driver node running the Spark Streaming application fails
  - Driver must be restarted on another node.
  - Requires a checkpoint directory in the StreamingContext

- Streaming Backpressure
  - spark.streaming.backpressure.enabled
  - spark.streaming.receiver.maxRate

Performance Recommendations

- Tuning:
  - Batch size and partitioning (block interval)
  - Find the optimal Batch size for your application by testing and monitoring.

- Minimum recommended block interval: 50 milliseconds

- Number of tasks is considered high when it exceed 50/sec
  - One task per stage per partition

- Other:
  - Watch for task launching overhead
  - Watch for garbage collection issues

“If the number of tasks launched per second is high (50 or more), then the overhead of sending out tasks to the slaves may be significant and will make it hard to achieve sub-second latencies”

Thank You

Suresh Matlapudi
Suresh.Matlapudi@ibm.com