DStreams

Objective

Understand Spark Streaming low-level APIs

Work with Discretized Streams (DStreams)

Discretized Streams

Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD

Discretized Streams

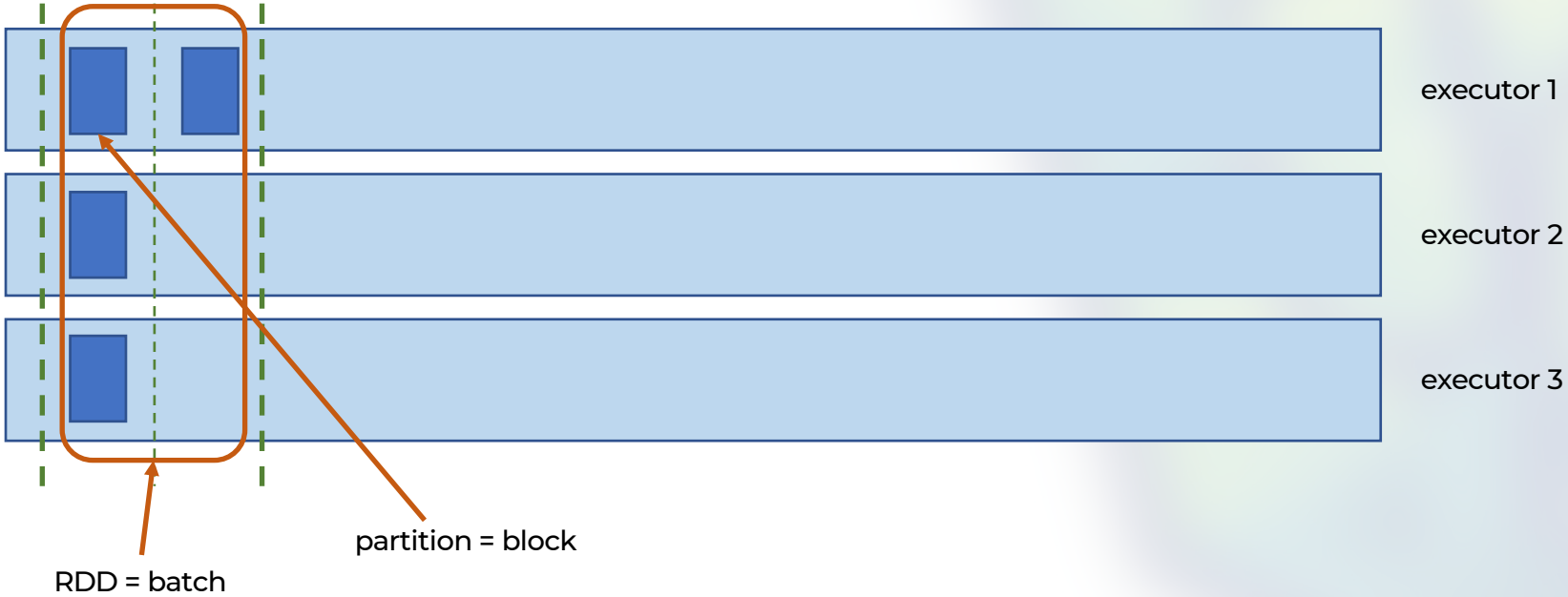
Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD

Discretized Streams

Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD



Discretized Streams

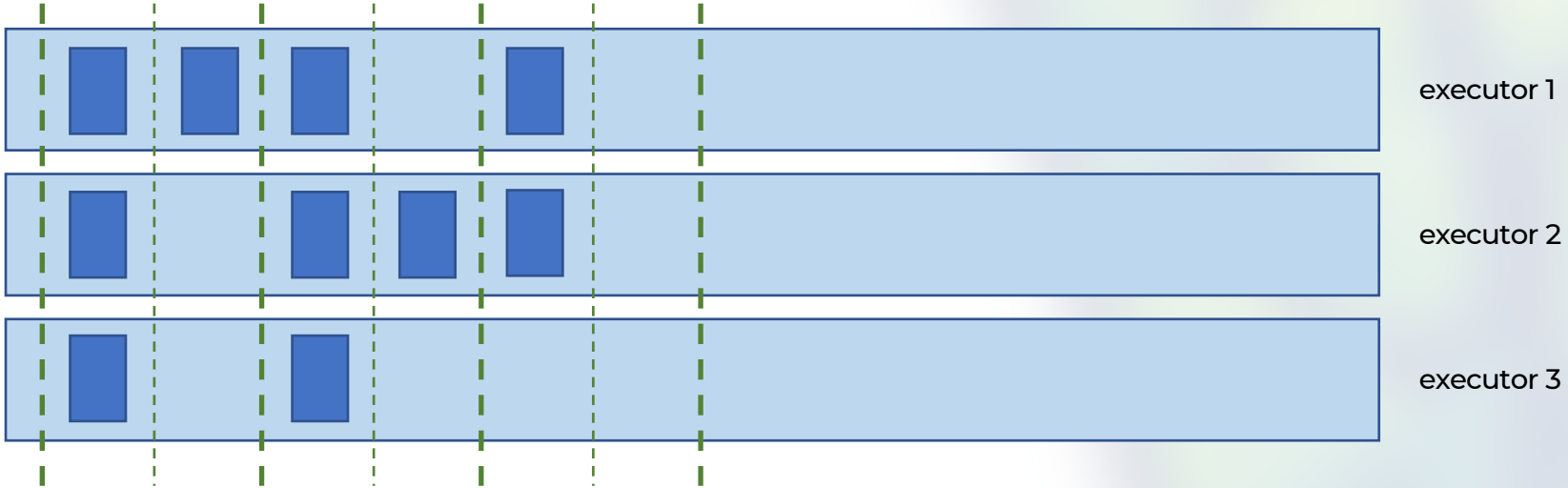
Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD

Discretized Streams

Never ending sequence of RDDs

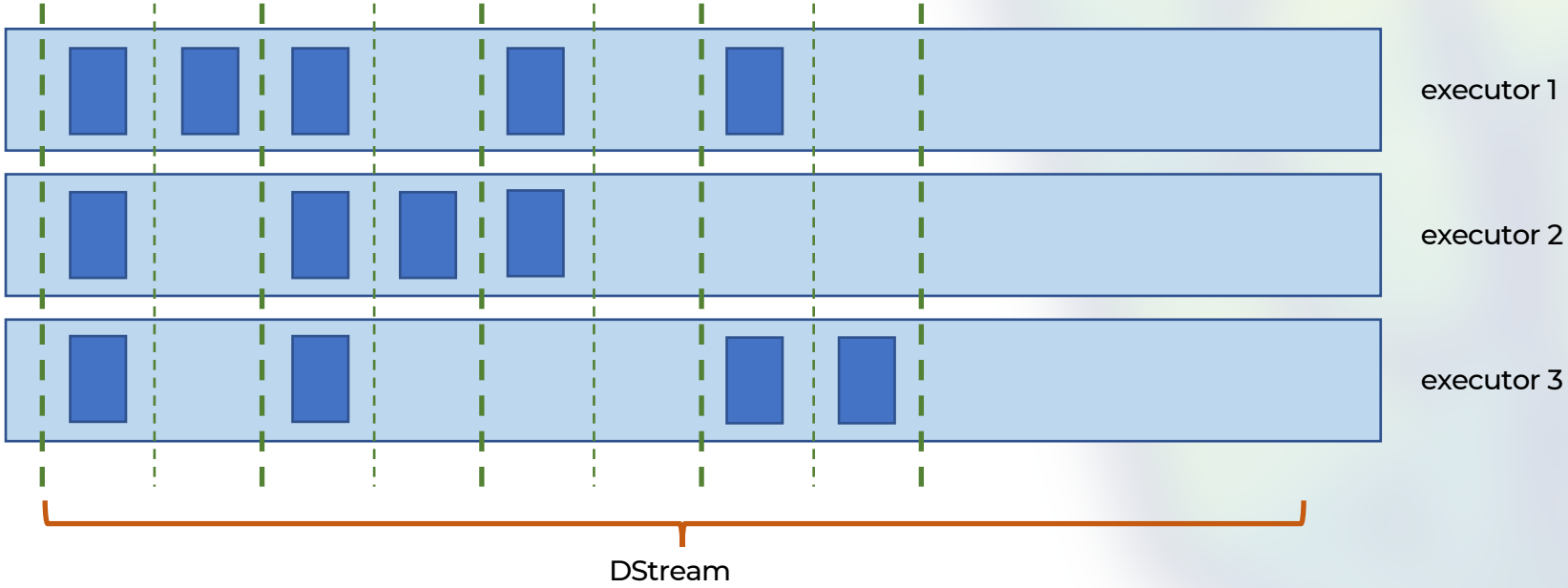
- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD



Discretized Streams

Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD



Discretized Streams

Never ending sequence of RDDs

- nodes' clocks are synchronized
- batches are triggered at the same time in the cluster
- each batch is an RDD

Essentially a distributed collection of elements of the same type

- functional operators e.g. map, flatMap, filter, reduce
- accessors to each RDD
- more advanced operators (later)

Needs a receiver to perform computations

- one receiver per DStream
- fetches data from the source, sends to Spark, creates blocks
- is managed by the StreamingContext on the driver
- occupies one core on the machine!

Takeaways

DStreams are a never-ending sequence of RDDs

Available under a StreamingContext


```
val ssc = new StreamingContext(spark.sparkContext, Seconds(1))
```

```
val socketStream: DStream[String] = ssc.socketTextStream("localhost", 9999)
val textStream = ssc.textFileStream("path/to/my/files") // file system is monitored
```

Support various transformations

```
socketStream.flatMap(_.split(" ")).map(_.length)
```

Computation started via an action + start of the streaming context

```
socketStream.print()      // action
ssc.start()               // no transformations/actions past this point
ssc.awaitTermination()
```

Regular slide

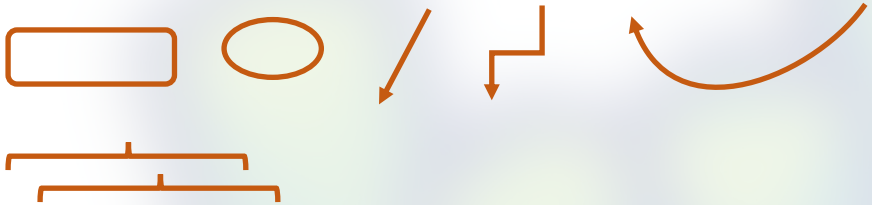
big comment

small comment

Error message

Text & bullet

- big bullet
- small bullet



```
(x, y) => x + y
```

```
keyword method[G](param: ActorSystem) => member + "some text" + aValOrVar
// comment
```

Spark rocks