Advanced Streaming Operations

Ahmad Alkilani
DATA ARCHITECT

@akizl

Advanced Streaming Operations

Checkpointing
Window Operations

Stateful Transformations
Streaming Cardinality Estimation

Advanced Streaming Operations

Checkpointing

Window Operations

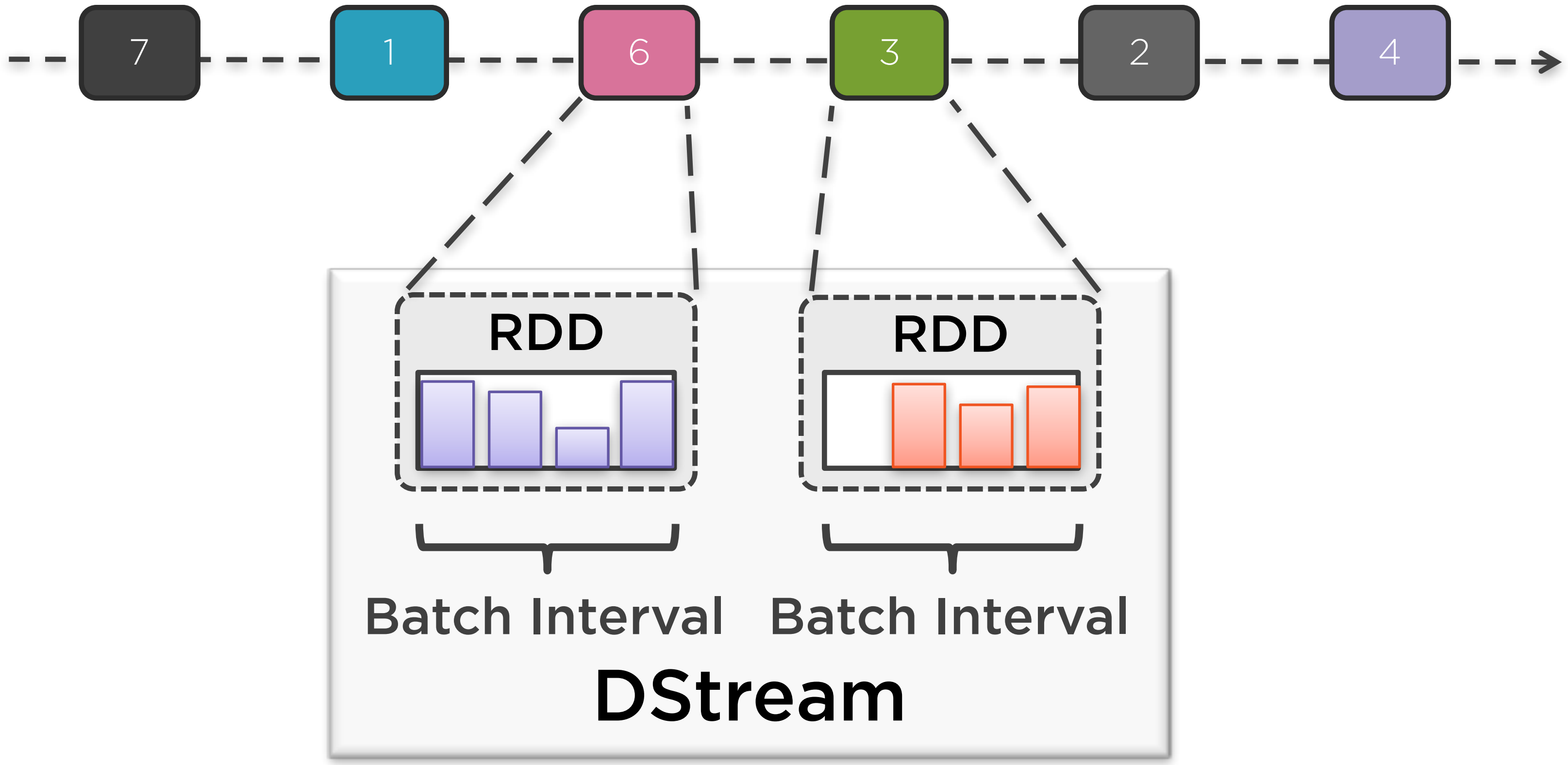
Stateful Transformations

Streaming Cardinality Estimation

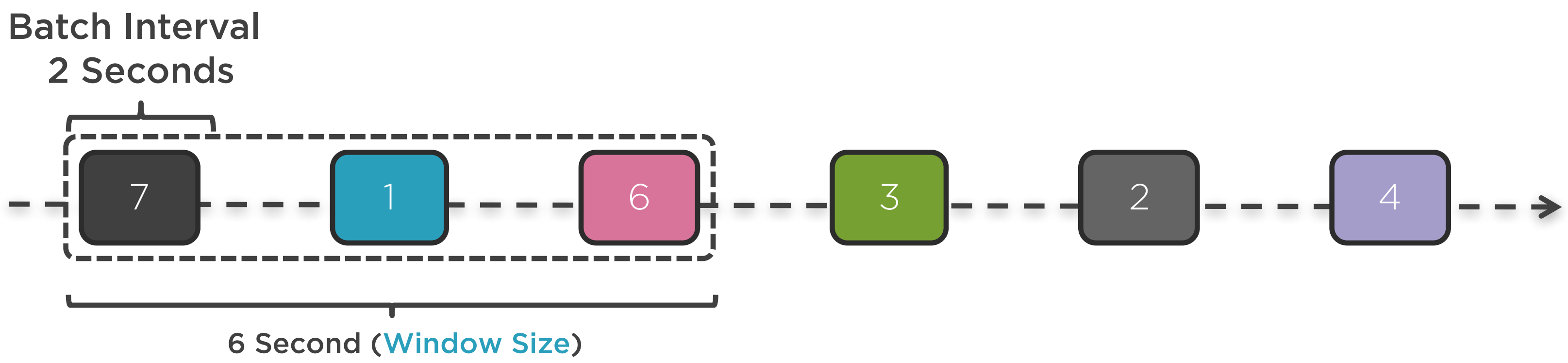
Checkpointing

```scala
def getSQLContext(sc: SparkContext) = {
  val sqlContext = SQLContext.getOrCreate(sc)
  sqlContext
}

def getStreamingContext(streamingApp : (SparkContext, Duration) => StreamingContext, sc : SparkContext, batchDuration: Duration) = {
  val creatingFunc = () => streamingApp(sc, batchDuration)
  val ssc = sc.getCheckpointDir match {
    case Some(checkpointDir) => StreamingContext.getActiveOrCreate(checkpointDir, creatingFunc, sc.hadoopConfiguration, createOnError = true)
    case None => StreamingContext.getActiveOrCreate(creatingFunc)
  }
  sc.getCheckpointDir.foreach( cp => ssc.checkpoint(cp))
  ssc
}
```

Window Operations



Window Operations



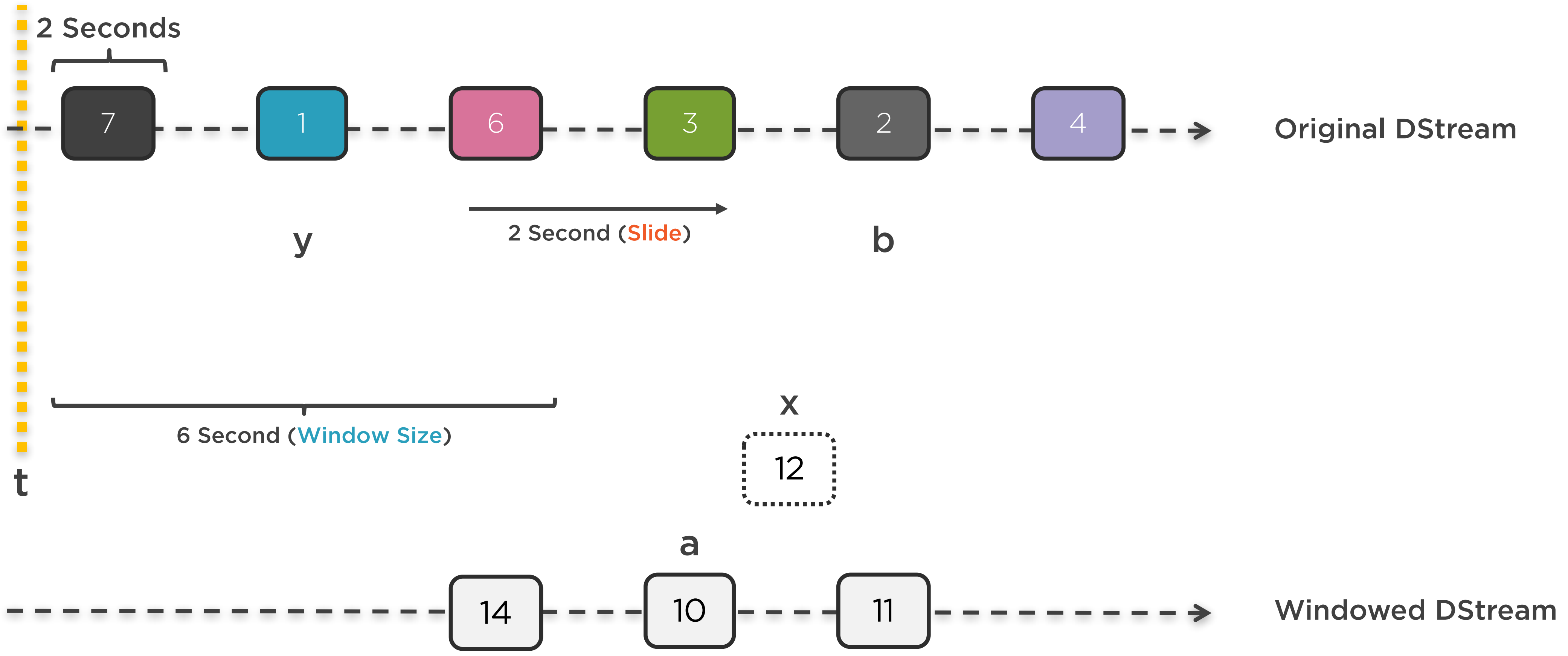
To create a window of a stream of data in Spark define:

- Window Size
- Slide Interval

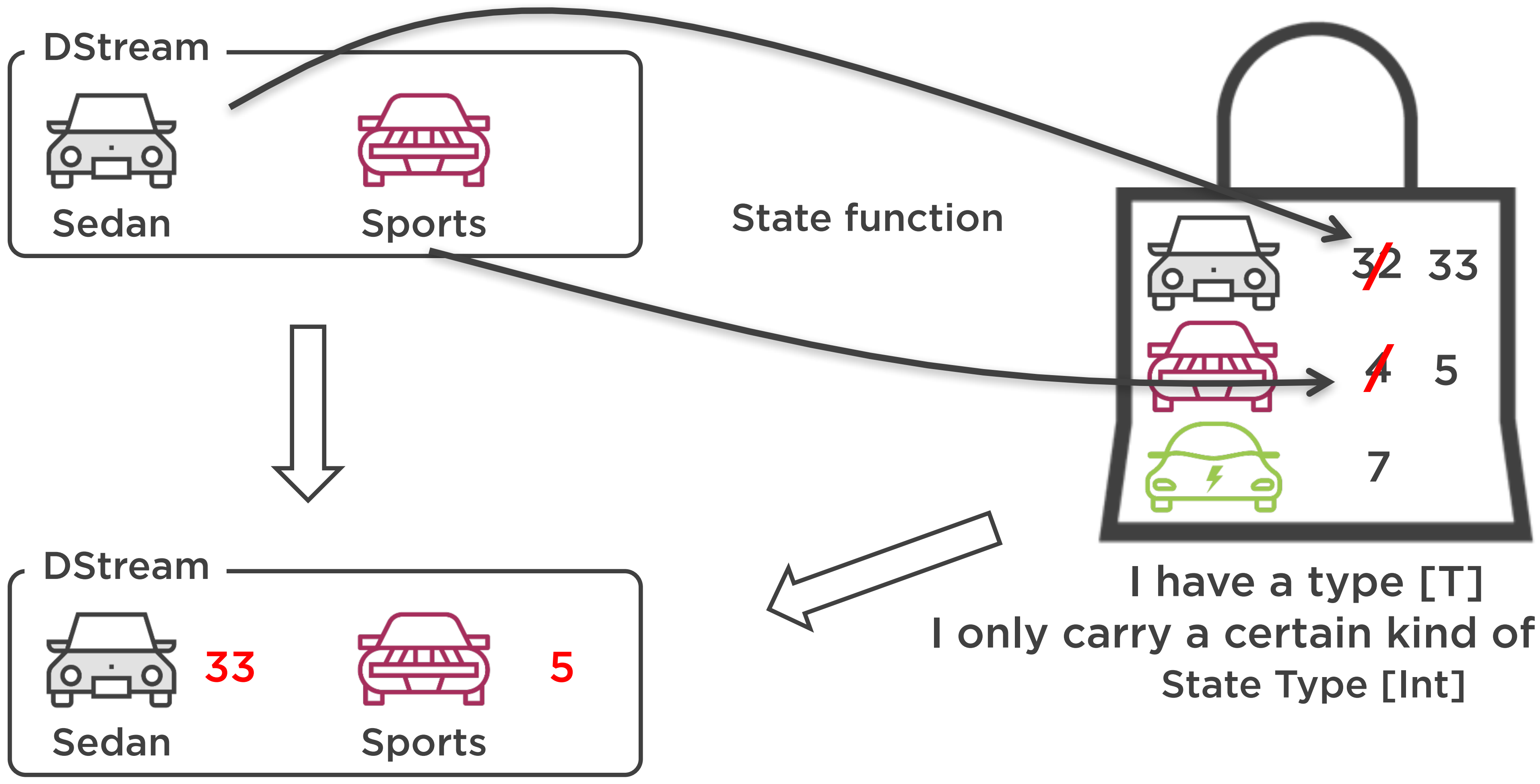
Must be multiples of batch interval

Window Operations

```
clicksDStream.window(Seconds(6), Seconds(2))
clicksDStream.reduceByWindow((a, b) => a + b, (x, y) => x - y, Seconds(6), Seconds(2))
```



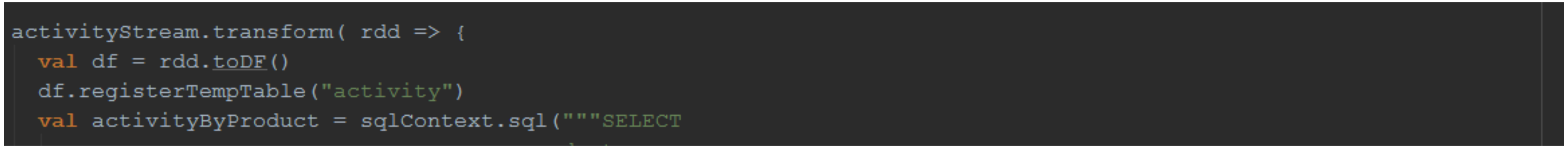
Stateful Transformations



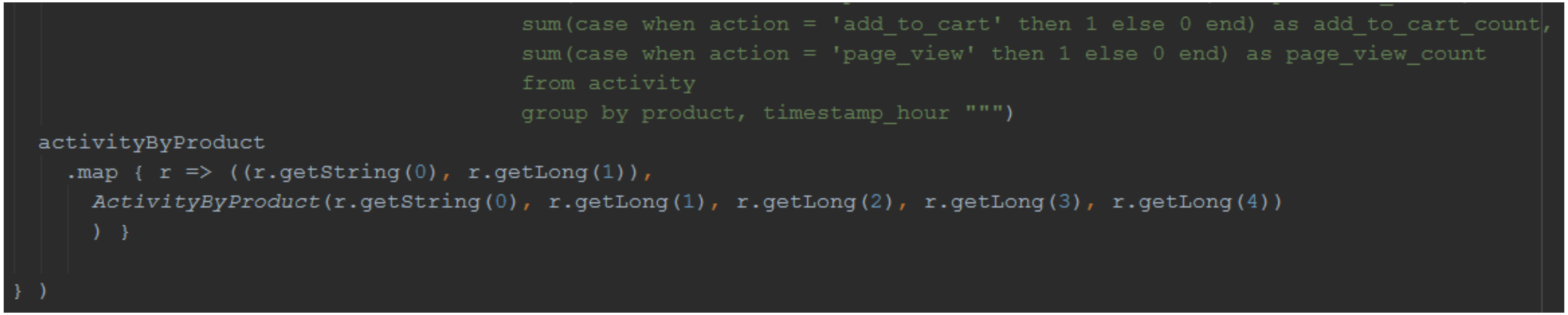
I have a type [T]
I only carry a certain kind of goods
State Type [Int]

Stateful Transformations

- updateStateByKey
- mapWithState



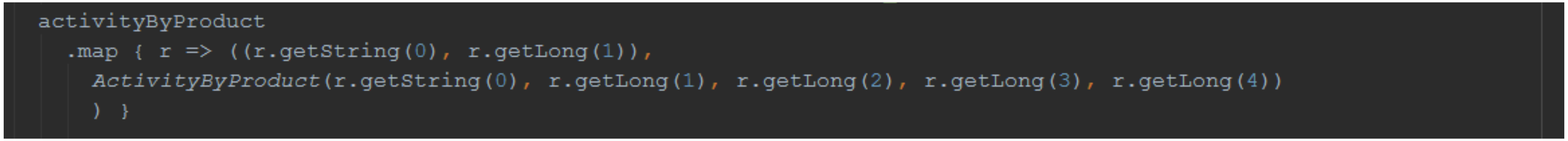
```
activityStream.transform( rdd => {
  val df = rdd.toDF()
  df.registerTempTable("activity")
  val activityByProduct = sqlContext.sql("""SELECT
```



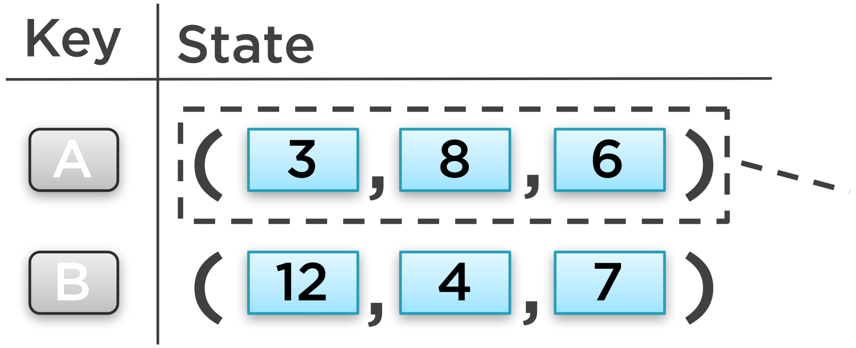
```
                                      sum(case when action = 'add_to_cart' then 1 else 0 end) as add_to_cart_count,
                                      sum(case when action = 'page_view' then 1 else 0 end) as page_view_count
                                      from activity
                                      group by product, timestamp_hour """)
  activityByProduct
    .map { r => ((r.getString(0), r.getLong(1)),
      ActivityByProduct(r.getString(0), r.getLong(1), r.getLong(2), r.getLong(3), r.getLong(4))
      ) }

} )
```

Stateful Transformations

- **updateStateByKey**
- mapWithState



```
activityByProduct
  .map { r => ((r.getString(0), r.getLong(1)),
    ActivityByProduct(r.getString(0), r.getLong(1), r.getLong(2), r.getLong(3), r.getLong(4))
    ) }
```



Key	State
A	(3 , 8 , 6)
B	(12 , 4 , 7)

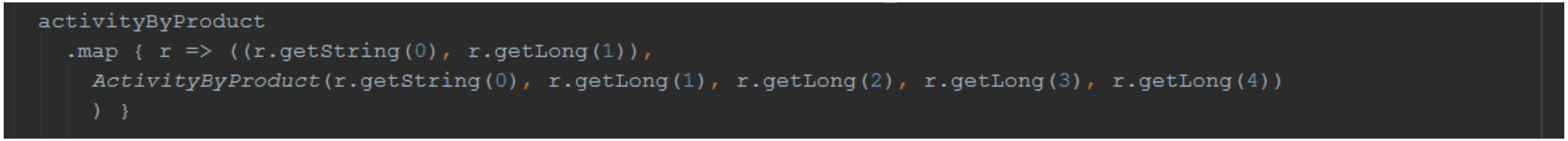
State is a tuple of 3 integers
[(Int, Int, Int)]

Option[(Int, Int, Int)]

```
.updateStateByKey[(Int, Int, Int)] (
  (newItemsPerKey: Seq[ActivityByProduct] , currentState: Option[(Int, Int, Int)]) => {    }
)
```

Stateful Operations

- updateStateByKey
- **mapWithState**



```
activityByProduct
  .map { r => ((r.getString(0), r.getLong(1)),
    ActivityByProduct(r.getString(0), r.getLong(1), r.getLong(2), r.getLong(3), r.getLong(4))
    ) }
```

mapWithState[StateType, MappedType](spec : StateSpec[K, V, …., ….])

: MapWithStateDStream[K, V, StateType, MappedType]

(Int, Int, Int)

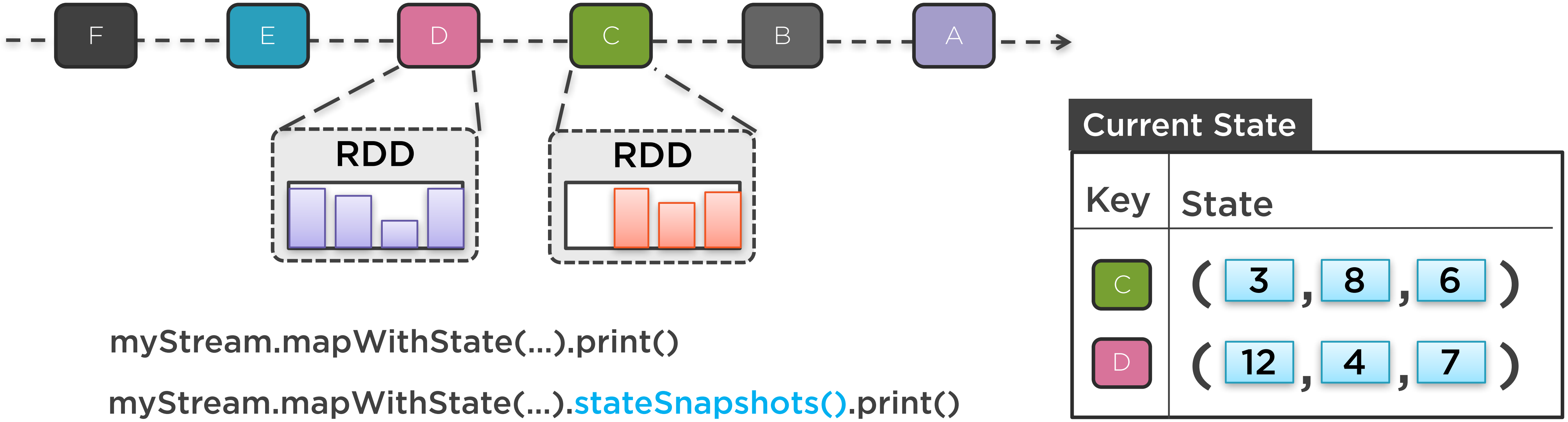
```
val spec = StateSpec
              .function((K, Option[V], State[StateType]) => Option[MappedType])
              .timeout(t: Duration)
```

(Int, Int, Int)

Stateful Operations

- updateStateByKey
- mapWithState

```
mapWithState[StateType, MappedType](spec : StateSpec[K, V, …., ….])
                                        : MapWithStateDStream[K, V, StateType, MappedType]
```



```
myStream.mapWithState(…).print()

myStream.mapWithState(…).stateSnapshots().print()
```

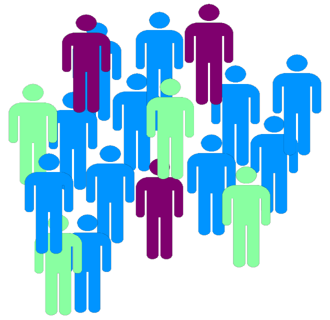
Cardinality Estimation using HyperLogLog

“Let’s shrink Big Data into Small Data ... and hope it magically becomes Great Data.”

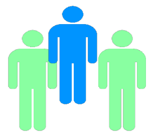
HLL - Cardinality estimation

Unique observations

Product “X” unique visitors in a certain timeframe



Timeframe expanded



Cardinality Estimation using HyperLogLog

"Let's shrink Big Data into Small Data ... and hope it magically becomes Great Data."

HLL - Cardinality estimation

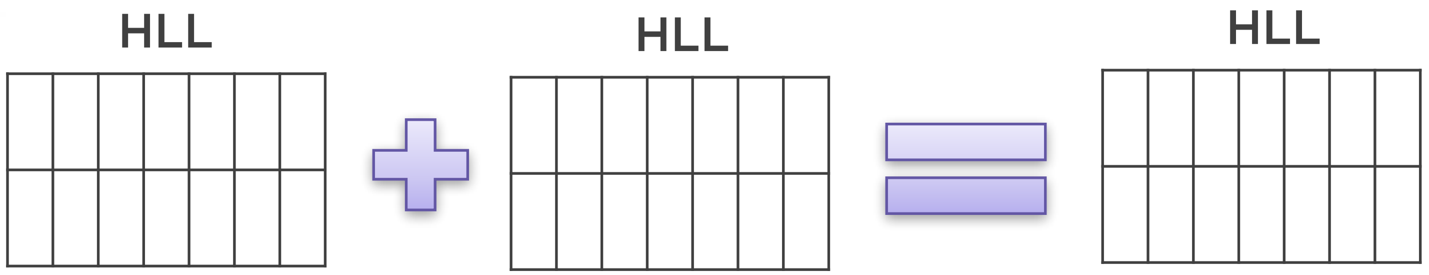
Unique observations

Naïve solution 3 Billion events > 60 GB of memory

HLL solves this in < 100s of KB

Based on bit pattern observables

Associative Data Structure



Summary

- **Window Operations**
- **Stateful Transformations**
  - updateStateByKey
  - mapWithState
- **Cardinality Estimation using HyperLogLog**
- **Algebird Library**
  - CountMinSketch
  - Bloom Filters
  - Priority Queues