Problem Solving Recipes Learned from Supporting Spark

Justin Pihony & Stavros Kontopolous
Lightbend

SPARK SUMMIT
EAST 2017

Table of Contents

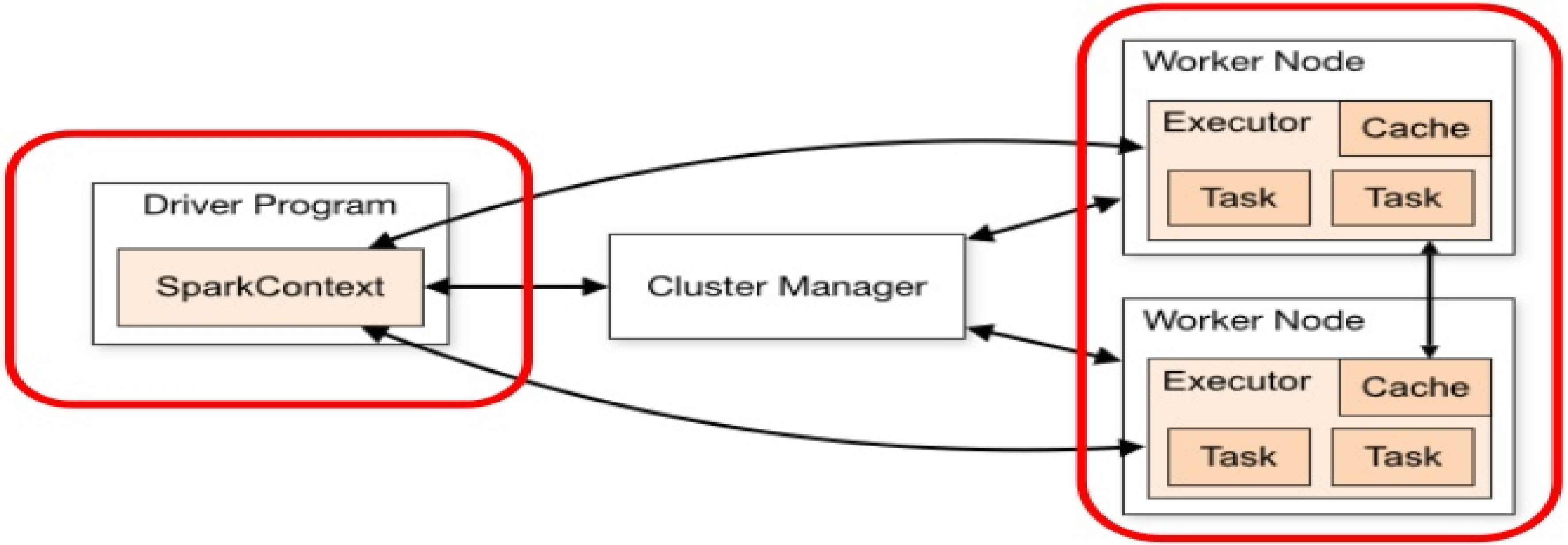
1. OOM
2. NoSuchMethod
3. Perplexities of Size
4. Struggles in Speculation
5. Strategizing Your Joins
6. Safe Stream Recovery
7. Handling S3 w/o Hanging

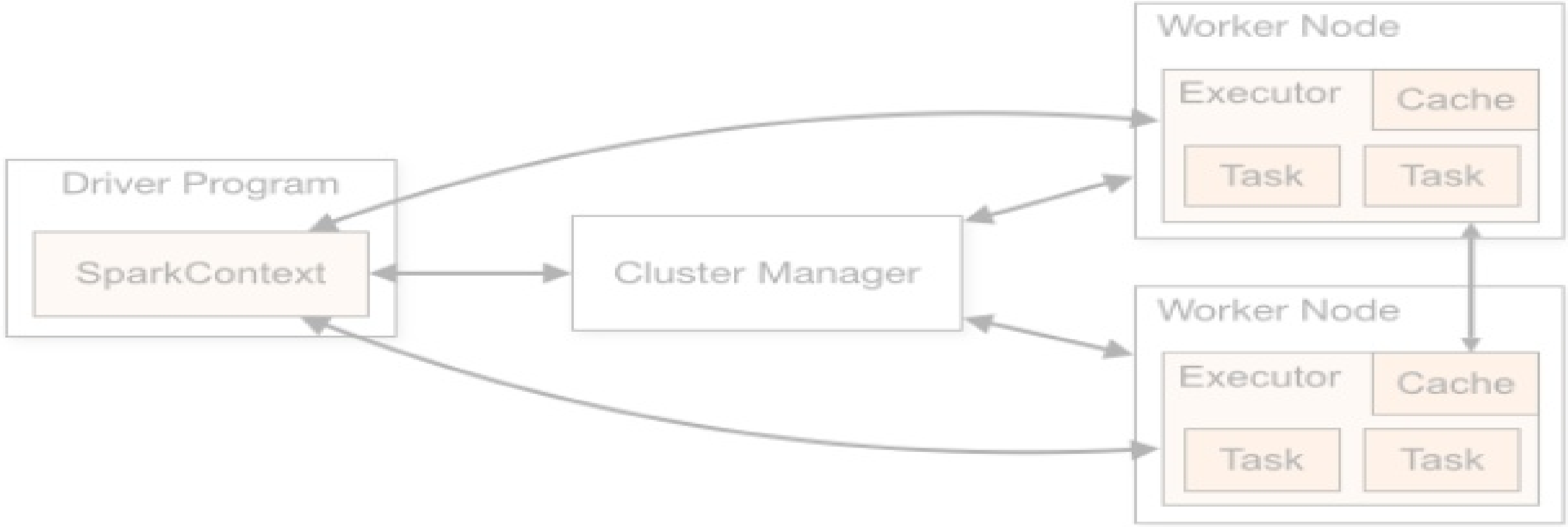
Prologue

Memory Problems?

Out Of Memory (OOM)

“Thrown when the Java Virtual Machine cannot allocate an object because it is ***out of memory****, and* ***no more memory could be made available by the garbage collector****.”*


Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task
Worker Node
Executor
Cache
Task
Task



`spark.memory.fraction vs spark.memory.storageFraction`

OOM Tips

- Don’t jump straight to parameter tuning
  - But if you do => Sparklint
- Be aware of execution time object creation

```
rdd.mapPartitions{ iterator =>  // fetch remote file }
```

OOM Tips

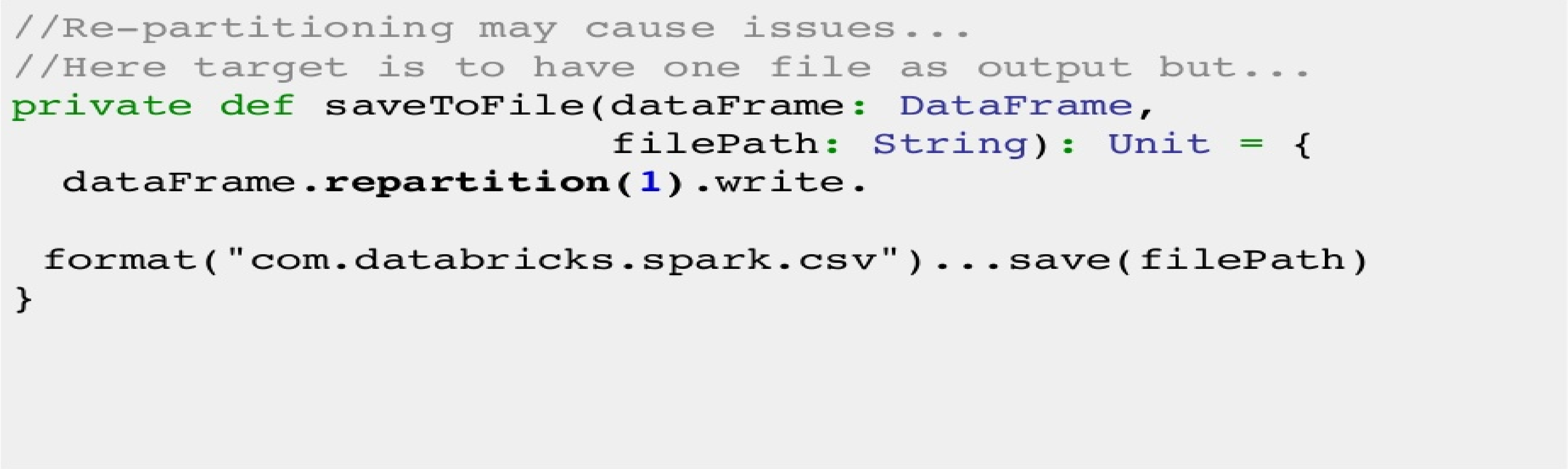
- Plan the resources needed from your cluster manager before deploying - when possible

EXAMPLE

YARN

- Cluster vs client mode
- yarn.nodemanager.resource.memory-mb
- yarn.scheduler.minimum-allocation-mb
- spark.yarn.driver.memoryOverhead

OOM



```
//Re-partitioning may cause issues...
//Here target is to have one file as output but...
private def saveToFile(dataFrame: DataFrame,
                       filePath: String): Unit = {
  dataFrame.repartition(1).write.

 format("com.databricks.spark.csv")...save(filePath)
}
```

NoSuchMethod

NoSuchMethod

"Thrown if an application tries to call a specified method of a class (either static or instance), and that ***class no longer has a definition*** *of that method. Normally, this error is caught by the compiler; this error can only occur* ***at run time if the definition of a class has incompatibly changed****."*

java.lang.NoSuchMethodError:
org.apache.commons.math3.util.MathArrays.natural

> "...spark 1.4.1 uses math3 3.1.1 which, as it turns out, doesn't have the `natural` method."

Dependency Collision

Compiled against **version A** Runtime used **version B**

Version B did not have the method!

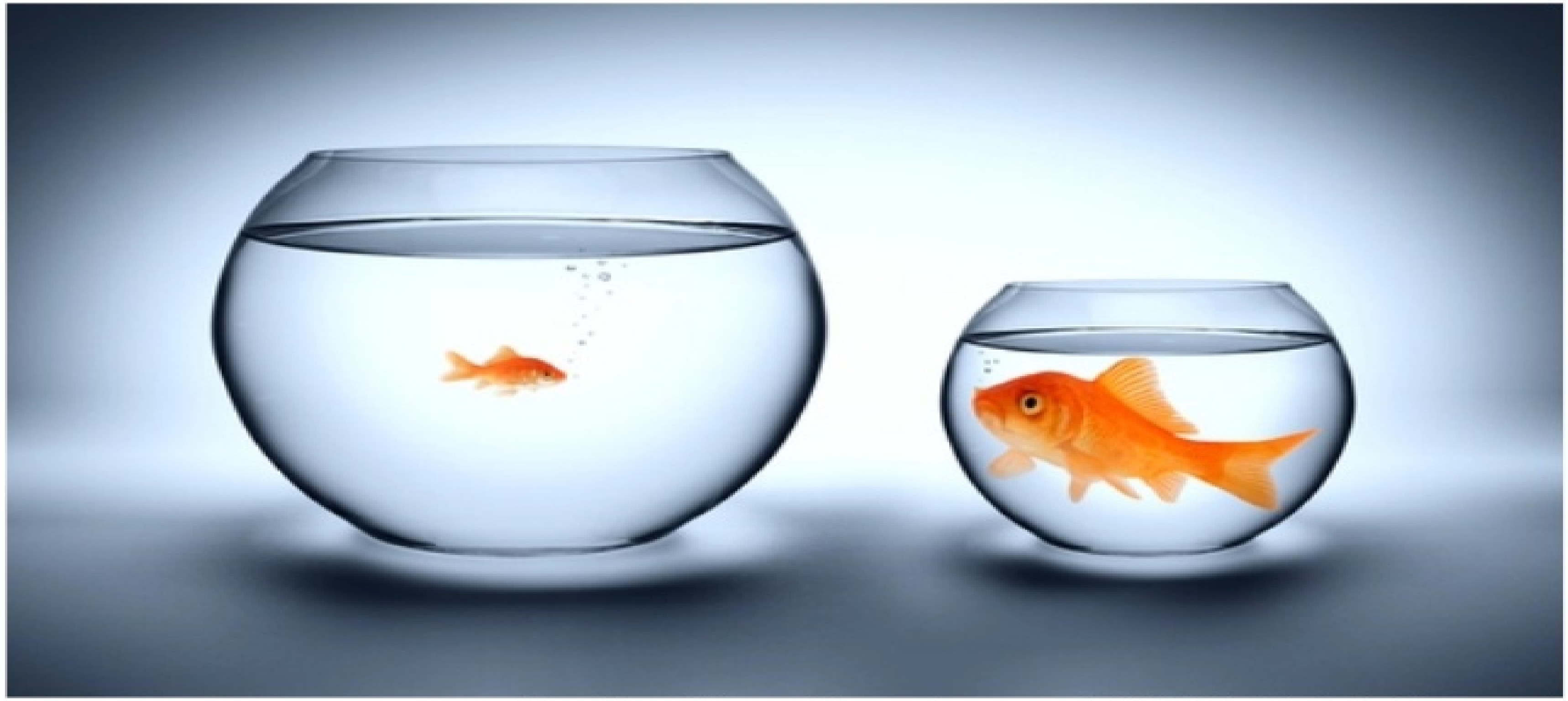
Solutions

- Upgrade Spark or downgrade your library
  - If you're lucky...
- Enforce lib load order

```
spark-submit --class "MAIN_CLASS"
  --driver-class-path commons-math3-3.3.jar YOURJAR.jar
```

- Shade your lib
  - sbt: https://github.com/sbt/sbt-assembly#shading
  - Maven: https://maven.apache.org/plugins/maven-shade-plugin/

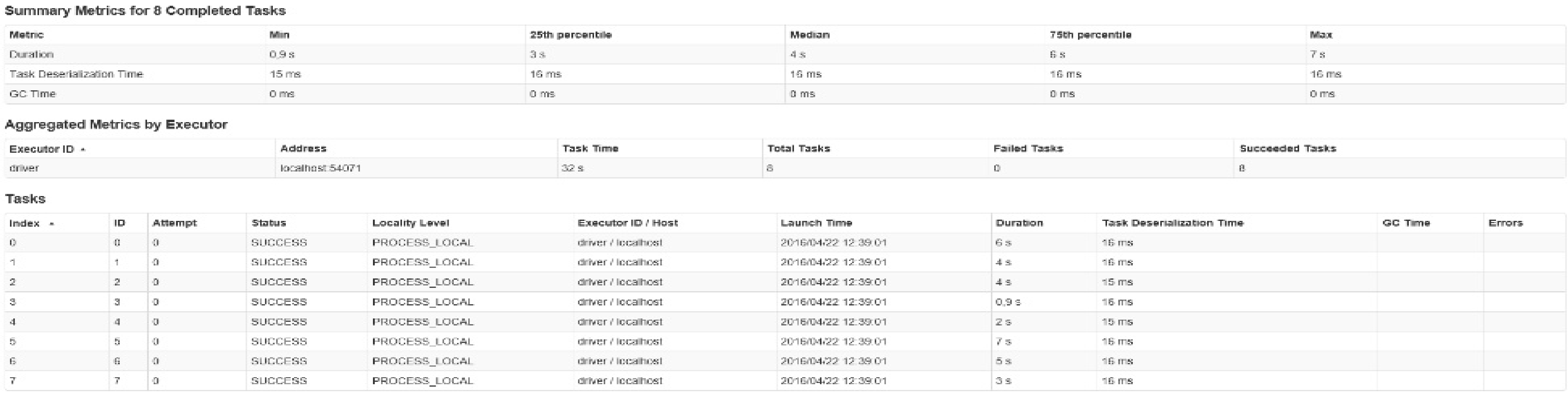
Perplexities of Size



https://josephderosa.files.wordpress.com/2016/04/too-big-by-half.jpg

Perplexities of Size

Perplexities of Size



Summary Metrics for 8 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.9 s	3 s	4 s	6 s	7 s
Task Deserialization Time	15 ms	16 ms	16 ms	16 ms	16 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms

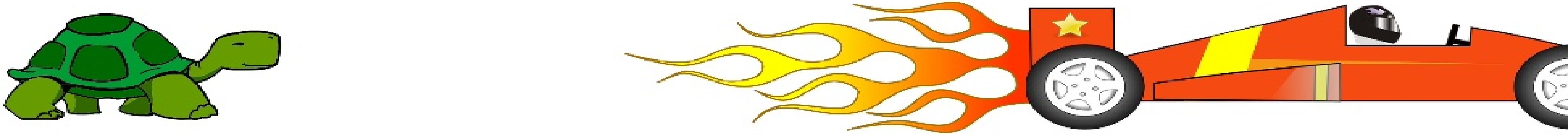
Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks
driver	localhost:54071	32 s	8	0	8

Tasks

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	Task Deserialization Time	GC Time	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	6 s	16 ms		
1	1	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	4 s	16 ms		
2	2	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	4 s	15 ms		
3	3	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	0.9 s	16 ms		
4	4	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	2 s	15 ms		
5	5	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	7 s	16 ms		
6	6	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	5 s	16 ms		
7	7	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2016/04/22 12:39:01	3 s	16 ms		

Struggles in Speculation



Struggles in Speculation

- `spark.speculation.interval`
- `spark.speculation.multiplier`
- `spark.speculation.quantile`

Strategizing Your Joins

SPARK SUMMIT EAST 2017

Common Issues

- Slow Joins
- Unavoidable Joins

Slow Joins

- Avoid shuffling if one side of the join is small enough

```
val df = largeDF.join(broadcast(smallDF), “key”)
```

- Check which strategy is actually used

```
df.explain
df.queryExecution.executedPlan
```

- For broadcast you should see :

```
 == Physical Plan ==
BroadcastHashJoin ... BuildRight
...
```

Join Strategies

- Broadcast
  - size < `SQLConf.AUTO_BROADCASTJOIN_THRESHOLD`
- Shuffle hash join
- Sort merge
  - `spark.sql.join.preferSortMergeJoin (default = true)`

Unavoidable Joins

```
val df = spark.sparkContext.parallelize(
  List(("Id1", 10, "London"), ("Id2", 20,  "Paris"), ("Id2", 1, "NY"), ("Id2", 20, "London"))
).toDF("GroupId", "Amount","City")
val grouped = df.groupBy("GroupId").agg(max("Amount"), first("City"))
grouped.collect().foreach(println)
```

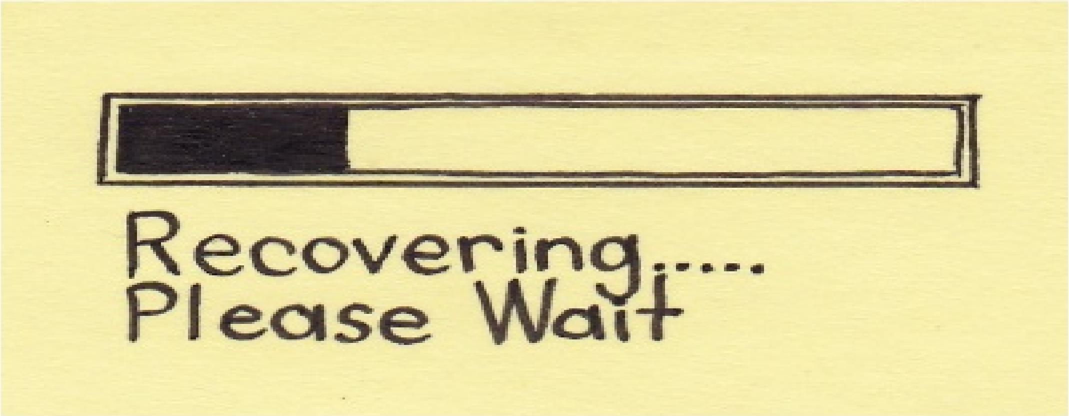
```
[Id1,10,London]
[Id2,20,Paris]
```

Joining is the only way to retain all related columns for

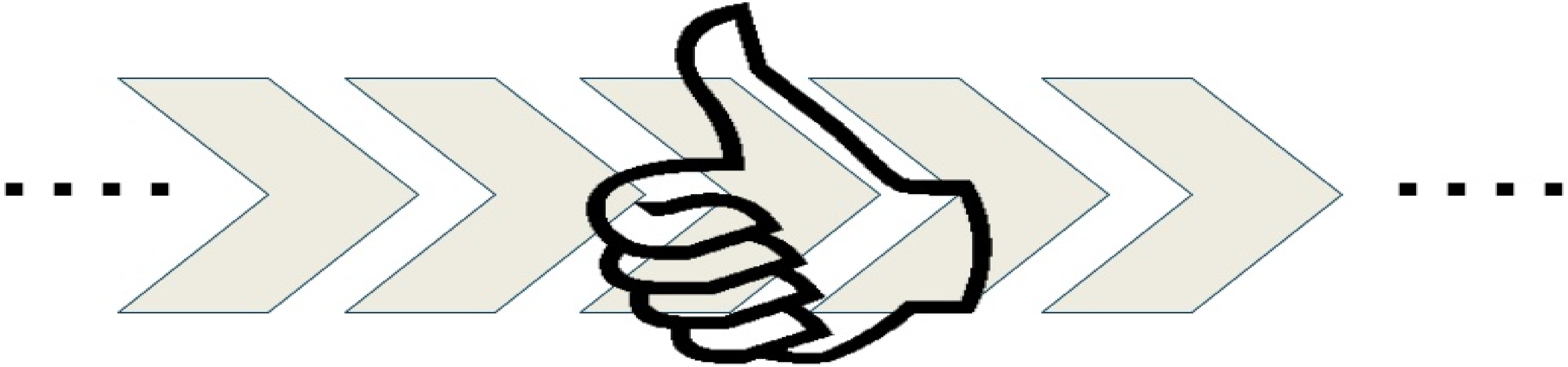
```
val joined = df.as("a").join(grouped.as("b"),
   $"a.GroupId" === $"b.GroupId" && $"a.Amount" === $"b.max(Amount)", "inner")
joined.collect().foreach(println)
```

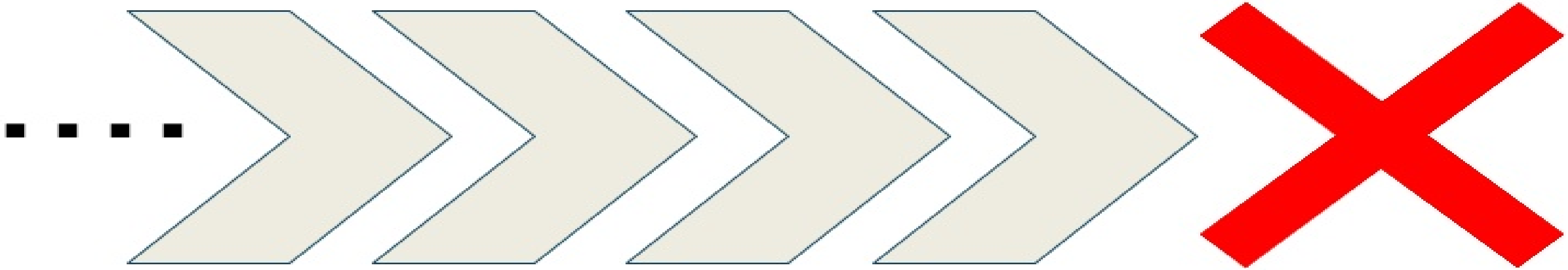
```
[Id1,10,London,Id1,10,London]
[Id2,20,Paris,Id2,20,Paris]
[Id2,20,London,Id2,20,Paris]
```

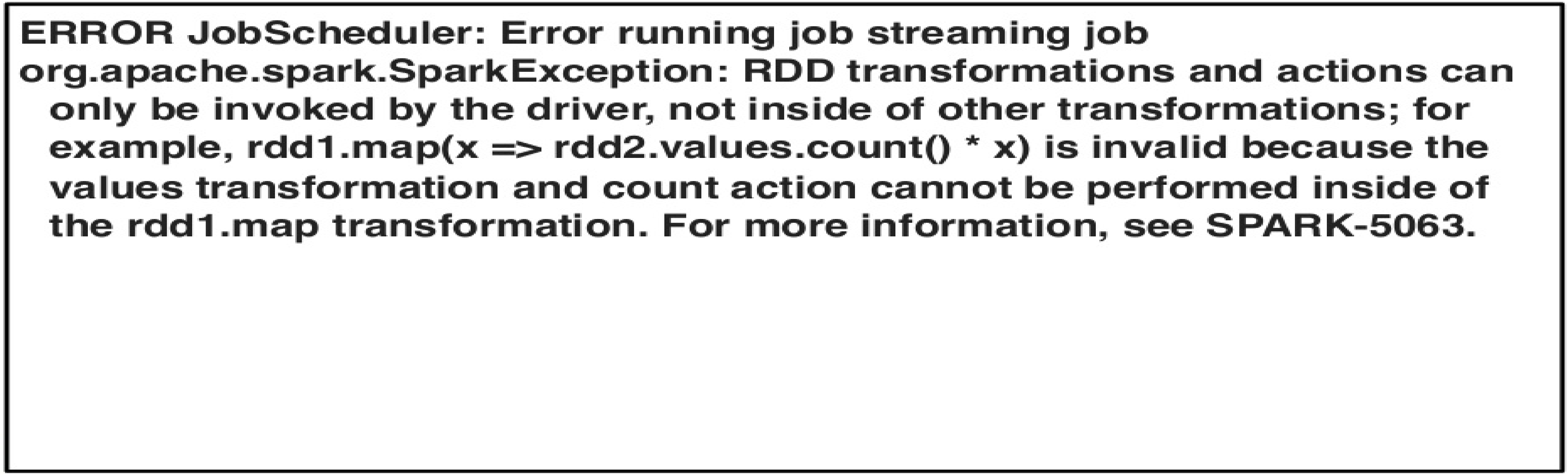
Safe Stream Recovery



```
val products = ssc.cassandraTable[(Int, String, Float)]
 (...)
  .map{ case (id, name, price) => (id, (name, price)) }
  .cache

orders.transform(rdd => {
  rdd.join(products)
})
```

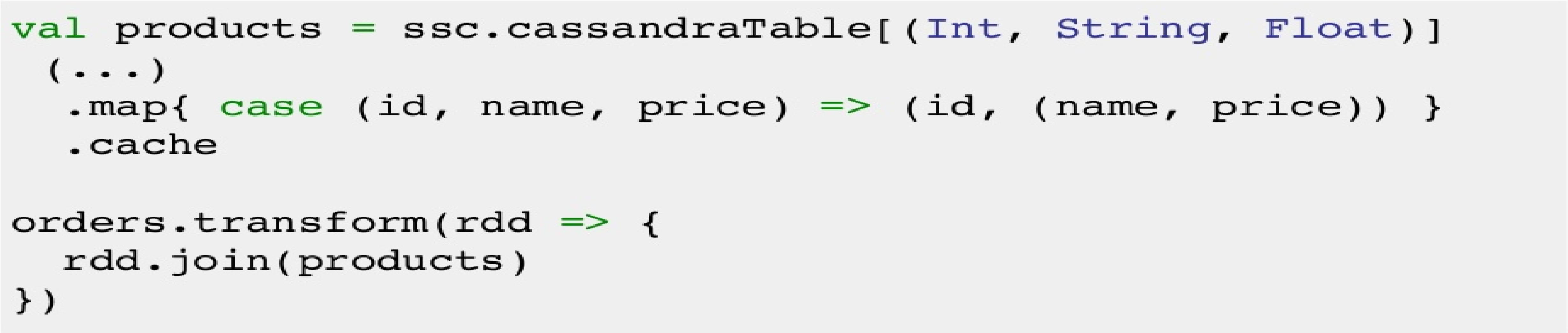




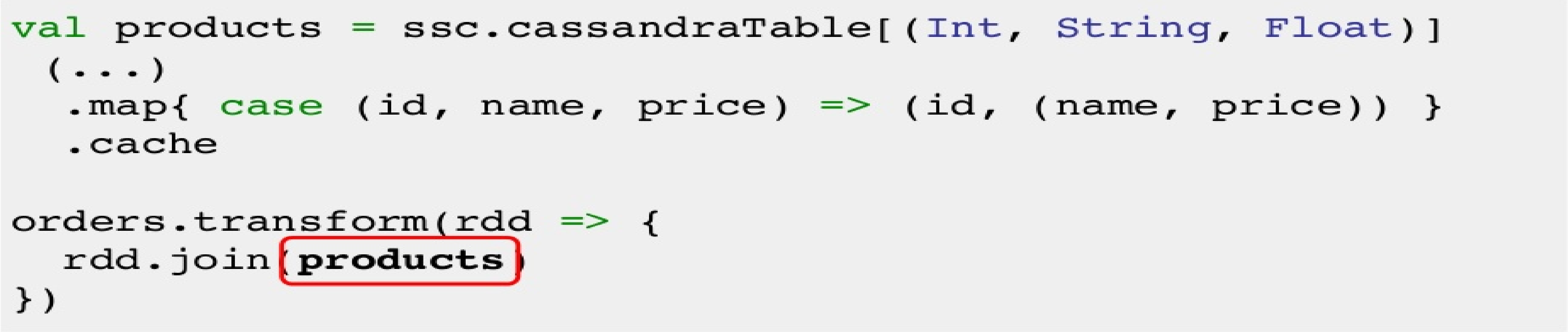


```
ERROR JobScheduler: Error running job streaming job
org.apache.spark.SparkException: RDD transformations and actions can
  only be invoked by the driver, not inside of other transformations; for
  example, rdd1.map(x => rdd2.values.count() * x) is invalid because the
  values transformation and count action cannot be performed inside of
  the rdd1.map transformation. For more information, see SPARK-5063.
```

```
ERROR JobScheduler: Error running job streaming job
org.apache.spark.SparkException: RDD transformations and actions
  can only be invoked by the driver, not inside of other
  transformations; for example, rdd1.map(x => rdd2.values.count() * x) is
  invalid because the values transformation and count action cannot be
  performed inside of the rdd1.map transformation. For more information,
  see SPARK-5063.
```



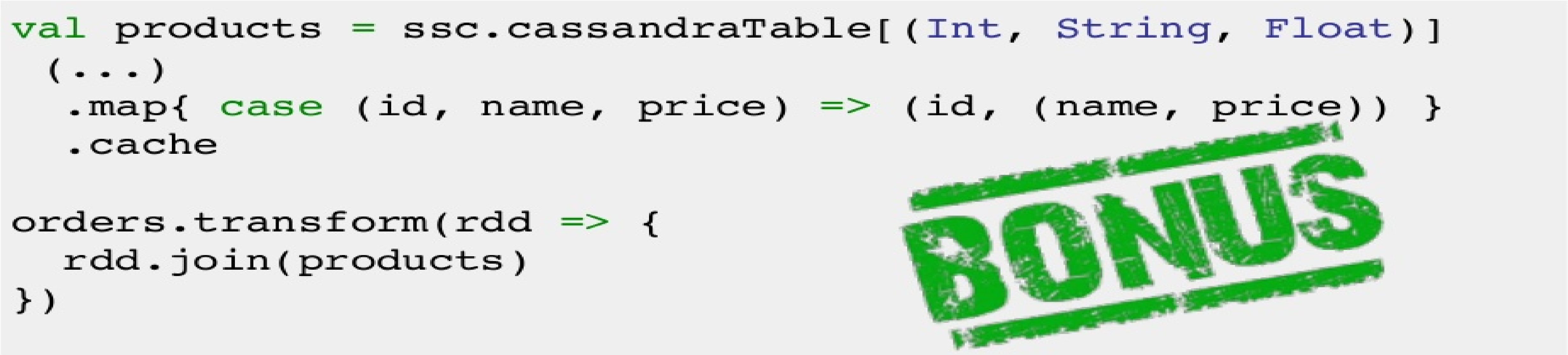
```
val products = ssc.cassandraTable[(Int, String, Float)]
 (...)
  .map{ case (id, name, price) => (id, (name, price)) }
  .cache

orders.transform(rdd => {
  rdd.join(products)
})
```



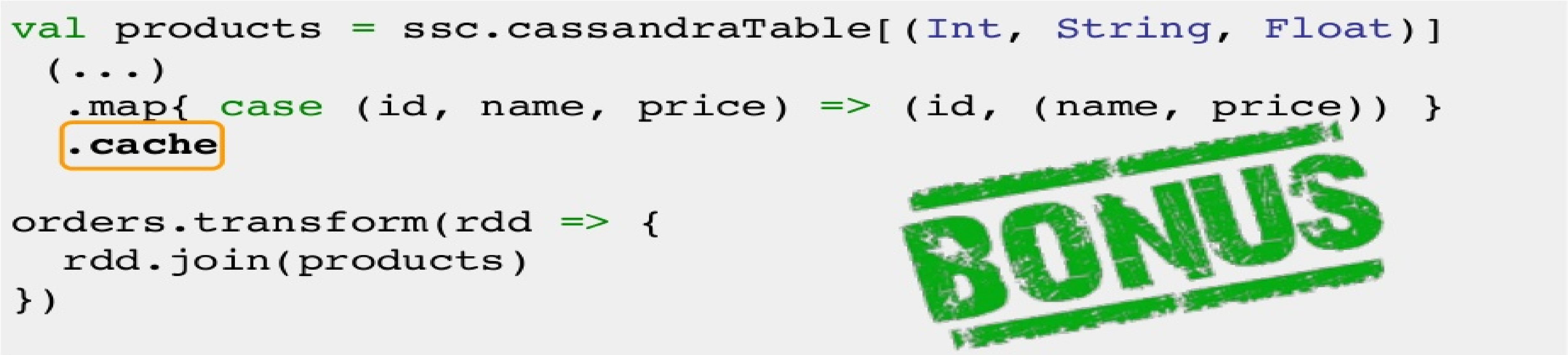
```
val products = ssc.cassandraTable[(Int, String, Float)]
 (...)
  .map{ case (id, name, price) => (id, (name, price)) }
  .cache

orders.transform(rdd => {
  rdd.join(products)
})
```

```
orders.transform(rdd => {
  val products = rdd.sparkContext.cassandraTable(…).map(…)
  rdd.join(products)
})
```

```
orders.transform(rdd => {
  val products = rdd.sparkContext.cassandraTable(…).map(…)
  rdd.join(products)
})
```

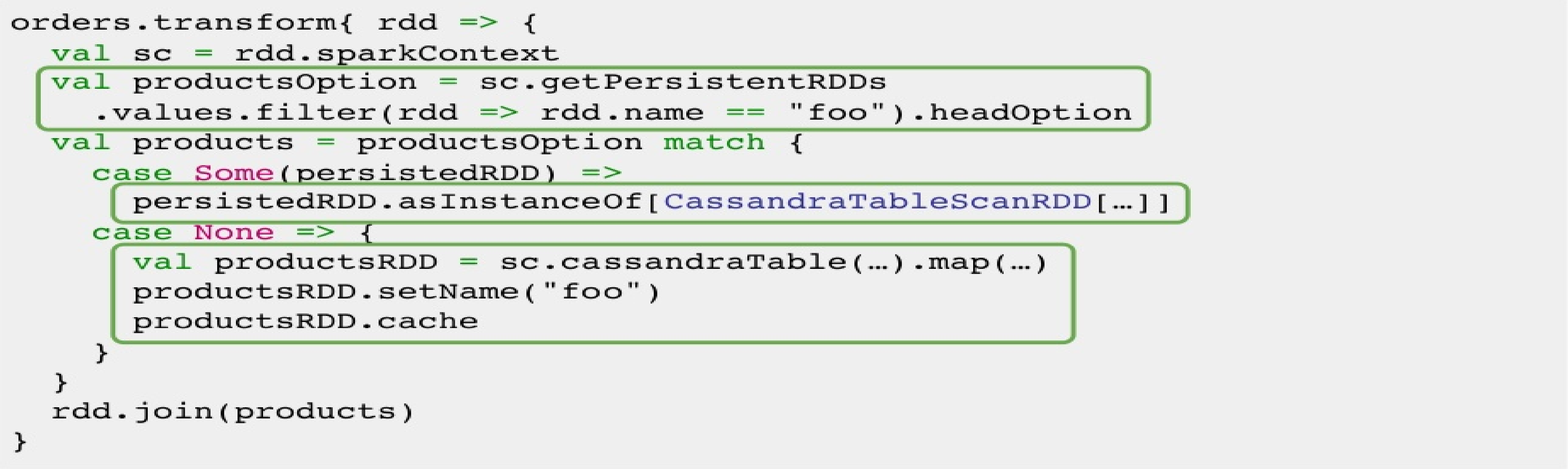


```
val products = ssc.cassandraTable[(Int, String, Float)]
 (...)
  .map{ case (id, name, price) => (id, (name, price)) }
  .cache

orders.transform(rdd => {
  rdd.join(products)
})
```



```
val products = ssc.cassandraTable[(Int, String, Float)]
 (...)
  .map{ case (id, name, price) => (id, (name, price)) }
  .cache

orders.transform(rdd => {
  rdd.join(products)
})
```

BONUS



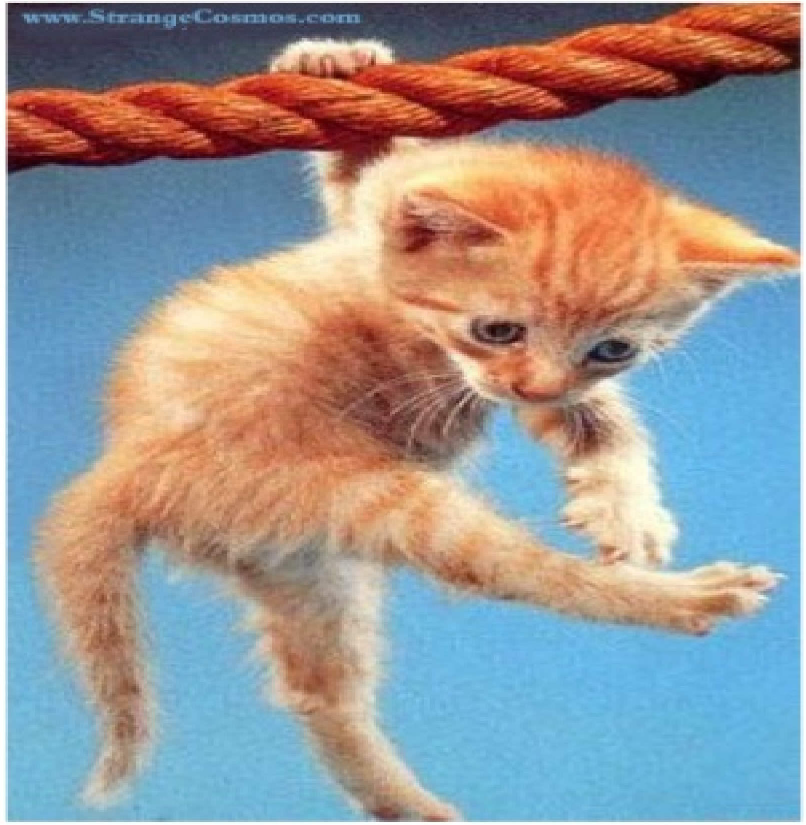
```
orders.transform{ rdd => {
  val sc = rdd.sparkContext
  val productsOption = sc.getPersistentRDDs
    .values.filter(rdd => rdd.name == "foo").headOption
  val products = productsOption match {
    case Some(persistedRDD) =>
      persistedRDD.asInstanceOf[CassandraTableScanRDD[…]]
    case None => {
      val productsRDD = sc.cassandraTable(…).map(…)
      productsRDD.setName("foo")
      productsRDD.cache
    }
  }
  rdd.join(products)
}
```

Safe Stream Recovery

```
orders.transform{ rdd => {
  val sc = rdd.sparkContext
  val productsOption = sc.getPersistentRDDs
    .values.filter(rdd => rdd.name == "foo").headOption
  val products = productsOption match {
    case Some(persistedRDD) =>
      persistedRDD.asInstanceOf[CassandraTableScanRDD[…]]
    case None => {
      val productsRDD = sc.cassandraTable(…).map(…)
      productsRDD.setName("foo")
      productsRDD.cache
    }
  }
  rdd.join(products)
}
```

✓

Handling S3 w/o Hanging



SPARK SUMMIT EAST 2017

Simple Streaming

```
val ssc = new StreamingContext(sc, Seconds(10))
val lines = ssc.textFileStream("s3n://some/folder")
lines.print
```

Simple Streaming, Right???

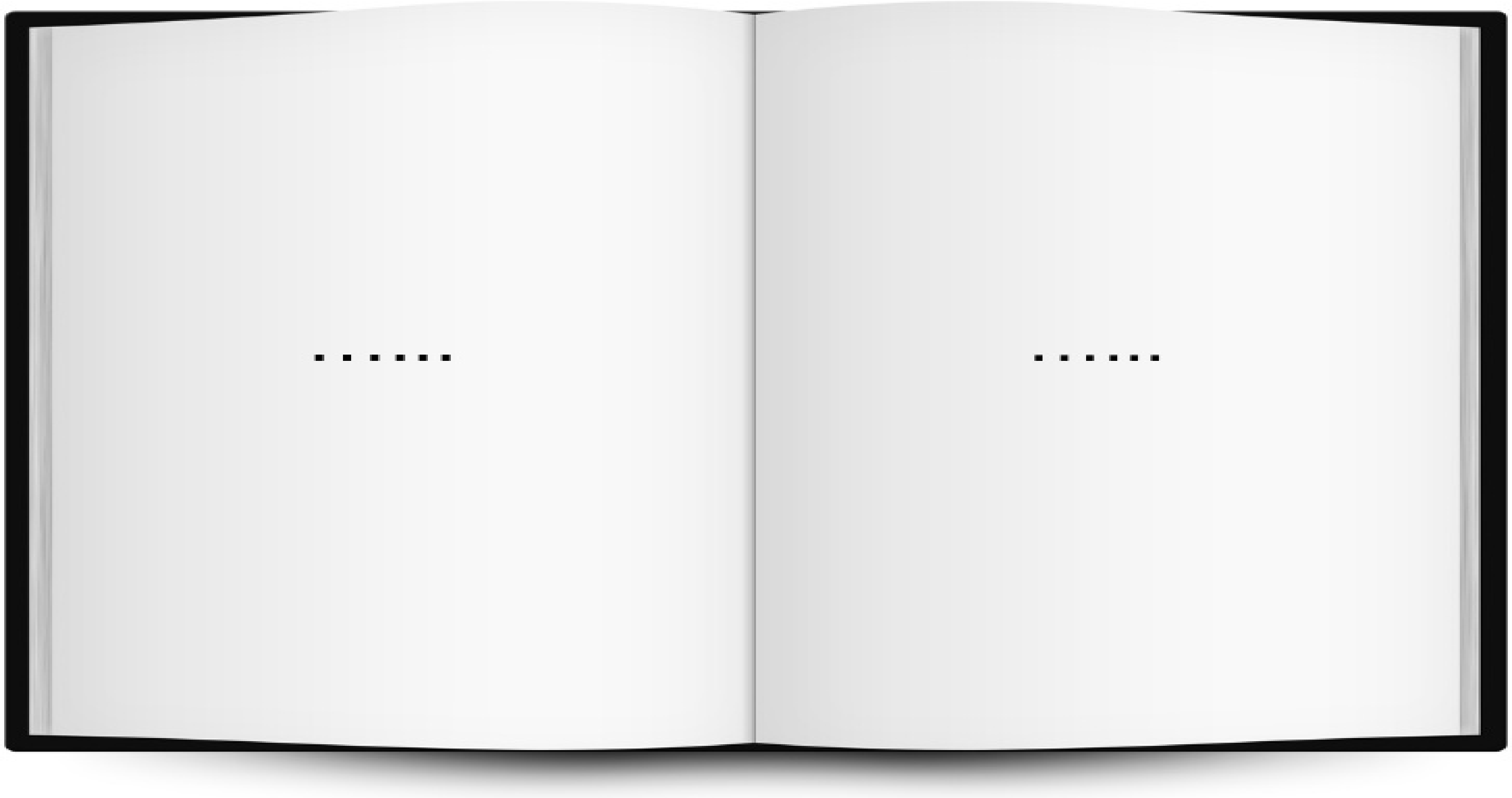
```
val ssc = new StreamingContext(sc, Seconds(10))
val lines = ssc.textFileStream("s3n://some/folder")
lines.print
```

Wrong!!!

44

Solutions

- Pace it with a script
- Custom s3n handler
- Explicit file handling
  - http://stackoverflow.com/a/34681877/779513
- Increase the heap
- Complain



THANK YOU.

justin.pihony@lightbend.com
stavros.kontopolous@lightbend.com

SPARK
SUMMIT
EAST 2017