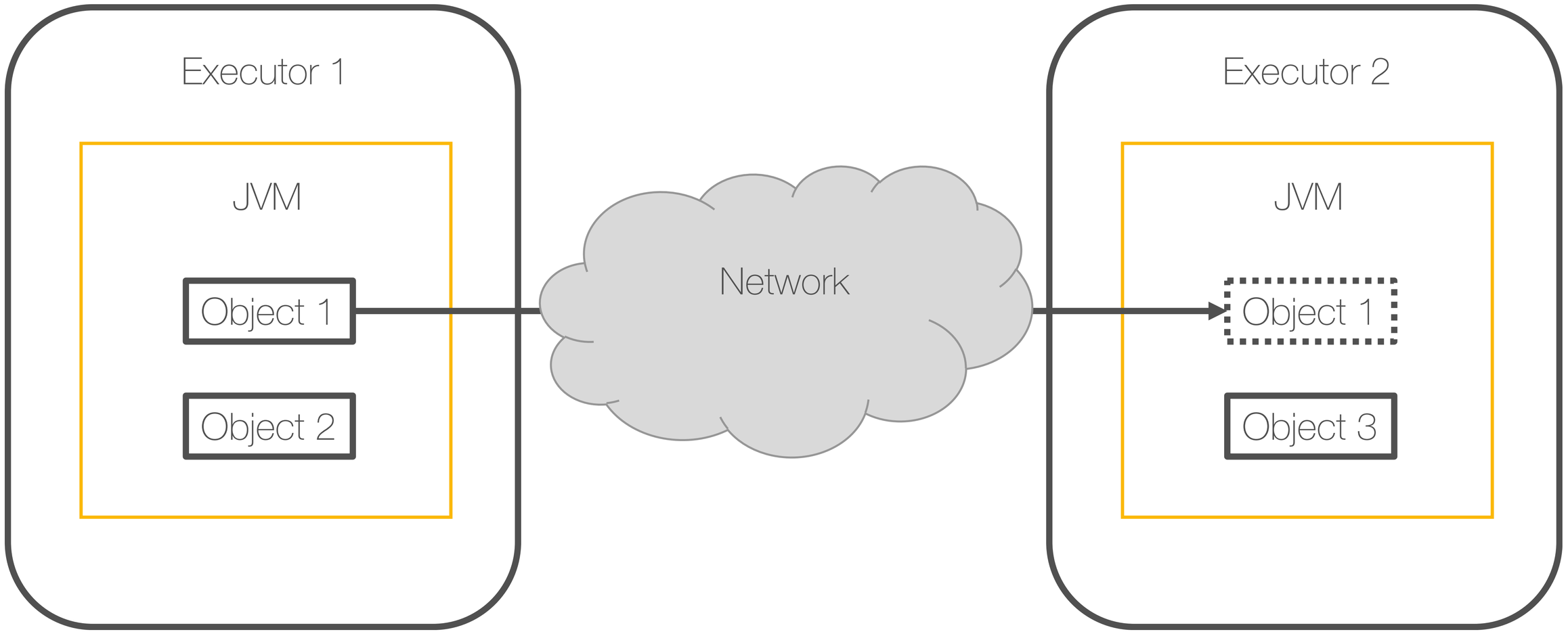
Shuffle. Serialization


Executor 1
JVM
Object 1
Object 2
Network
Executor 2
JVM
Object 1
Object 3

Serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and reconstructed later in the same or another computer environment

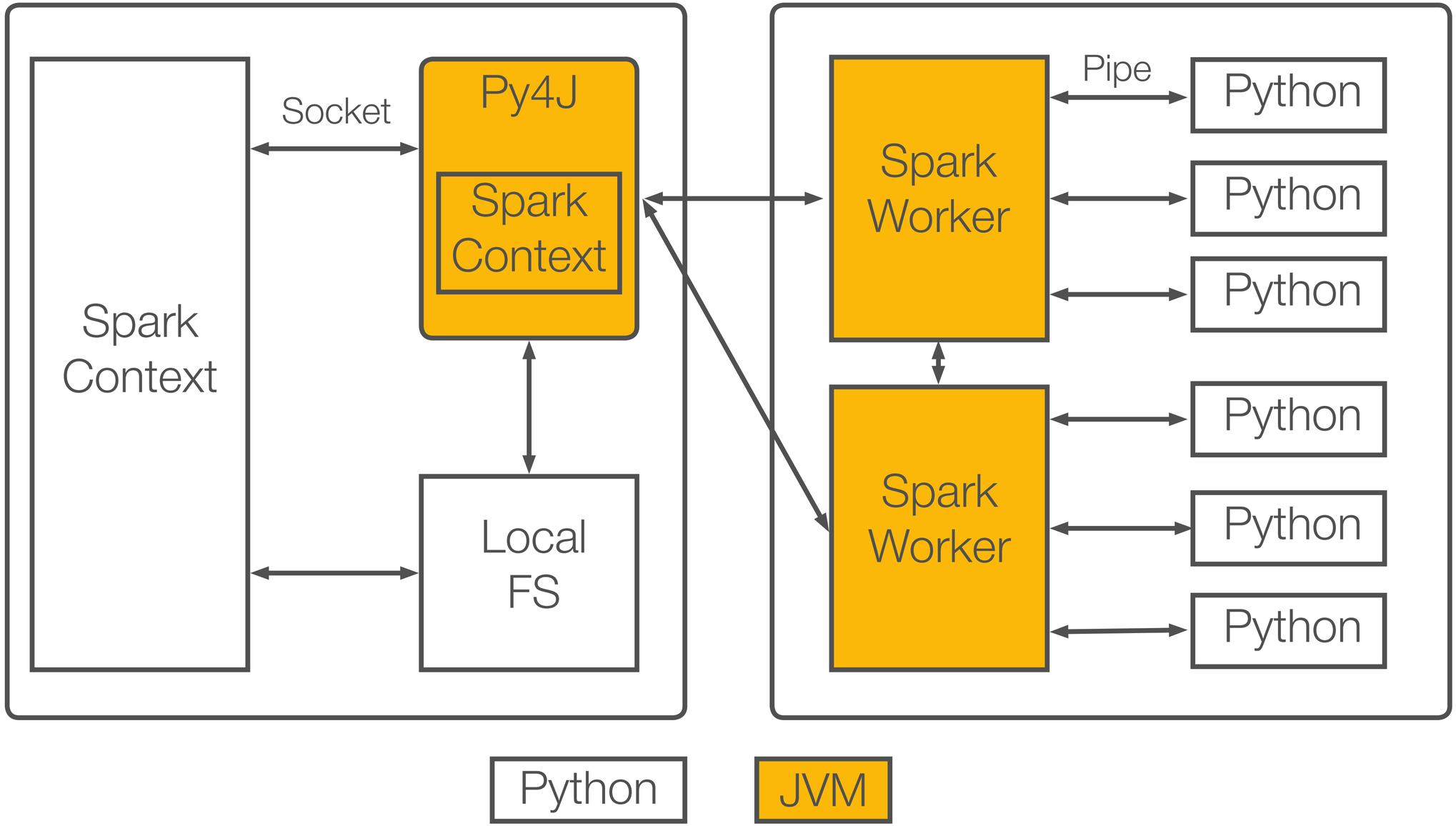
Spark serializers

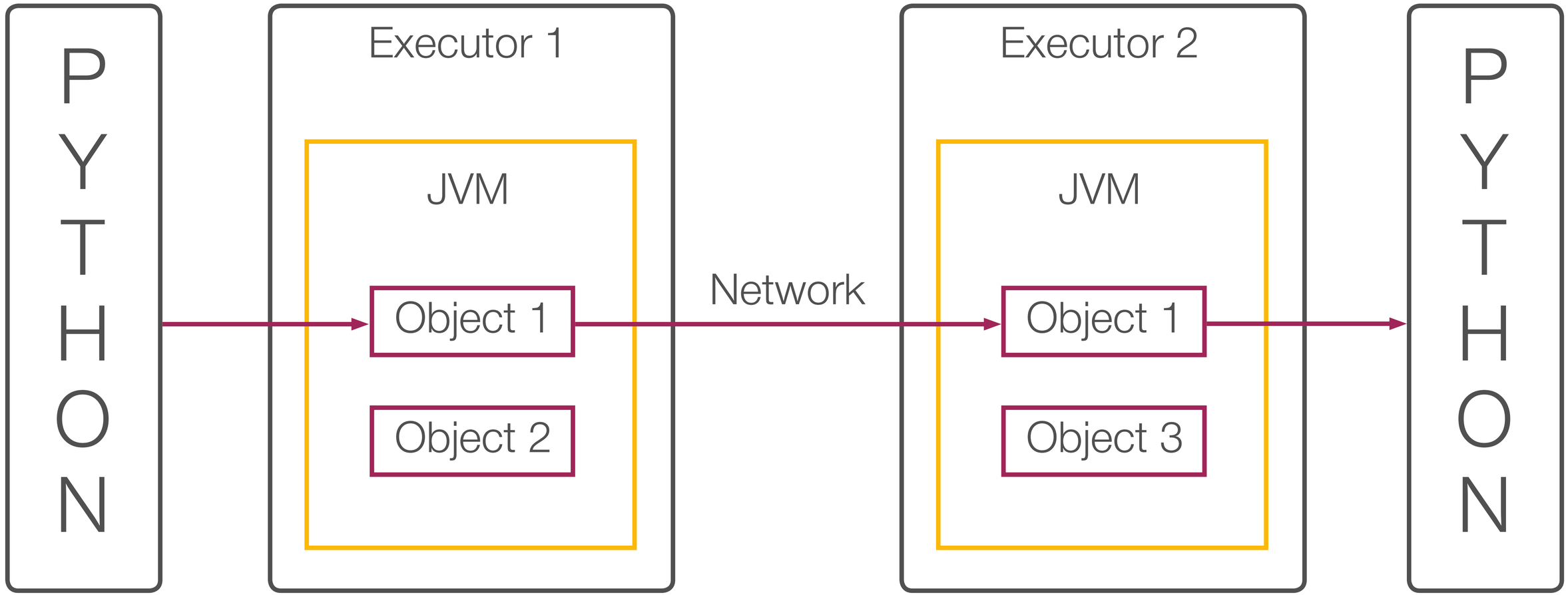
- Java – slow, but robust
- Kryo – fast, but has corner cases

Kryo

It’s not that useful for PySpark, but you can try it:

```
conf.set("spark.serializer",
"org.apache.spark.serializer.KryoSerializer")
```


Spark
Context
Socket
Py4J
Spark
Context
Local
FS
Spark
Worker
Spark
Worker
Pipe
Python
Python
Python
Python
Python
Python
Python
JVM


PYTHON
Executor 1
JVM
Object 1
Object 2
Network
Executor 2
JVM
Object 1
Object 3
PYTHON

PySpark reduces serialization costs by pipelining inside Python

```
logs = sc.textFile("log.txt")\
         .filter(lambda x: 'INFO' not in x)\
         .map(lambda x: (x.split('\t')[1], 1))\
         .reduceByKey(lambda x, y: x + y)

logs.collect()
```

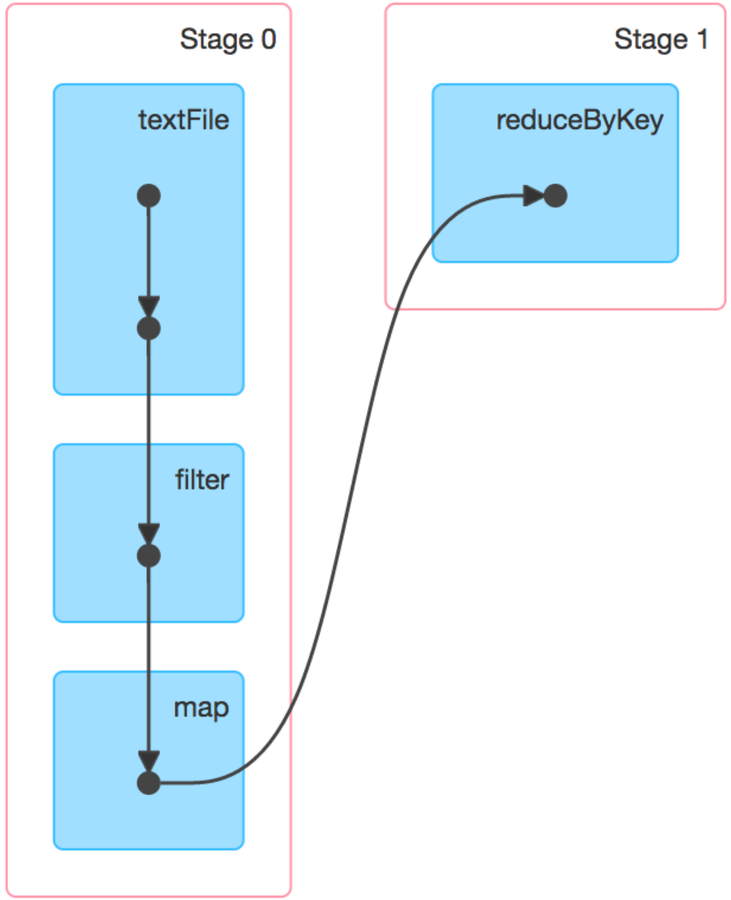
Details for Job 0

Status: SUCCEEDED

Completed Stages: 2

▸ Event Timeline

▾ DAG Visualization



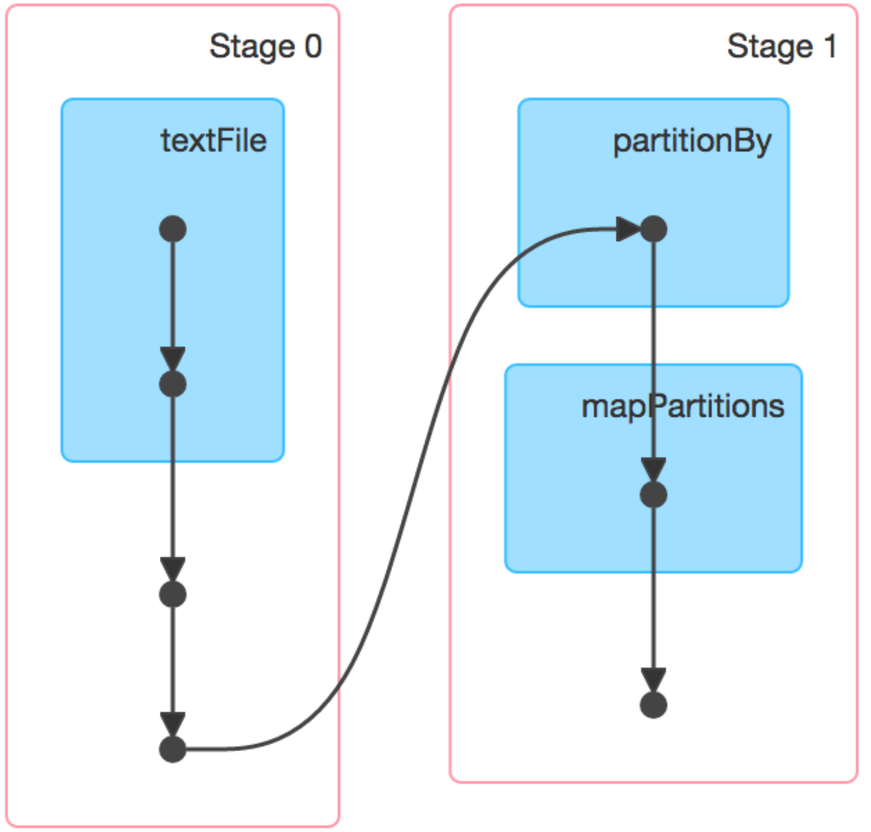
Details for Job 0

Status: SUCCEEDED

Completed Stages: 2

▸ Event Timeline

▾ DAG Visualization



Summary

- Spark uses serialization to transfer data and code
- There are two serializers:
  - Java (slow, but robust)
  - Kryo (fast, but has corner cases)
- PySpark adds double serialization
- PySpark tries to reduce serialization by pipelining
  - This produces strange DAGs