Spark Broadcast Variables and Accumulators

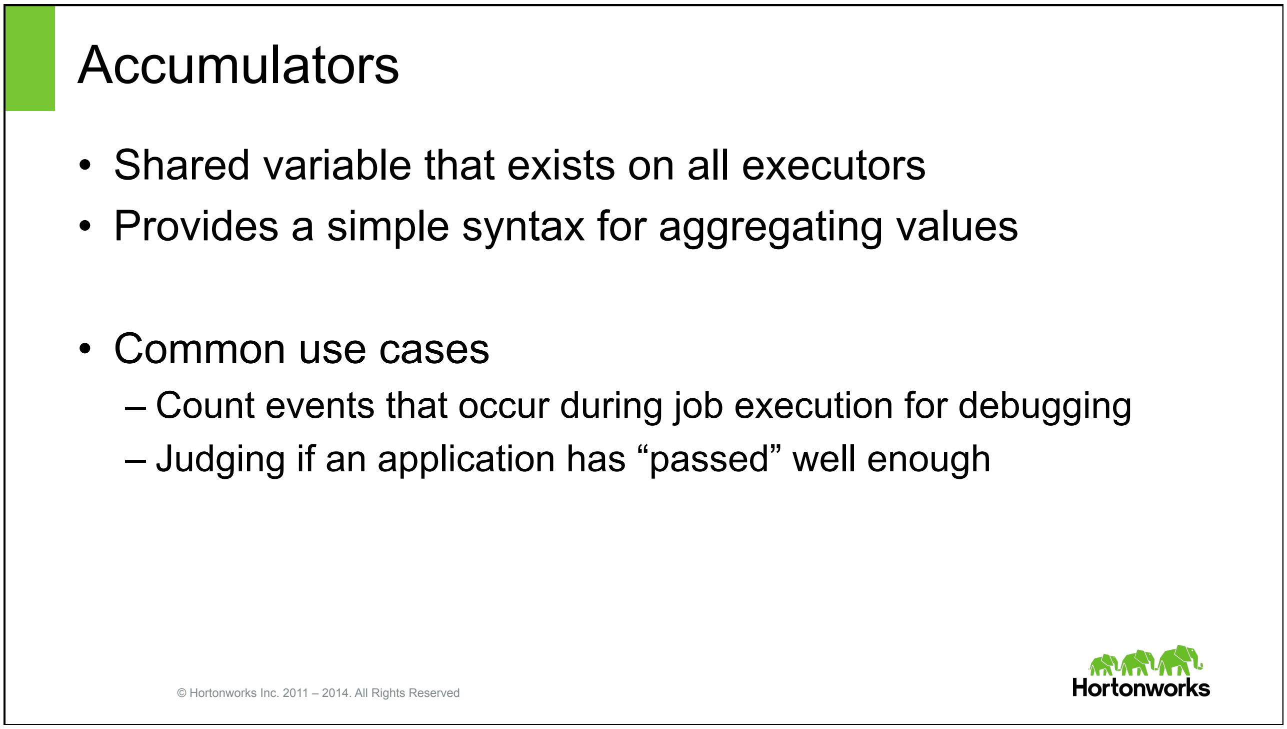
Lesson 7

Hortonworks

Learning Objectives

- After you complete this lesson you should be able to:
  - – Understand the benefit of Accumulators
  - – Be able to use a broadcast variable

Hortonworks

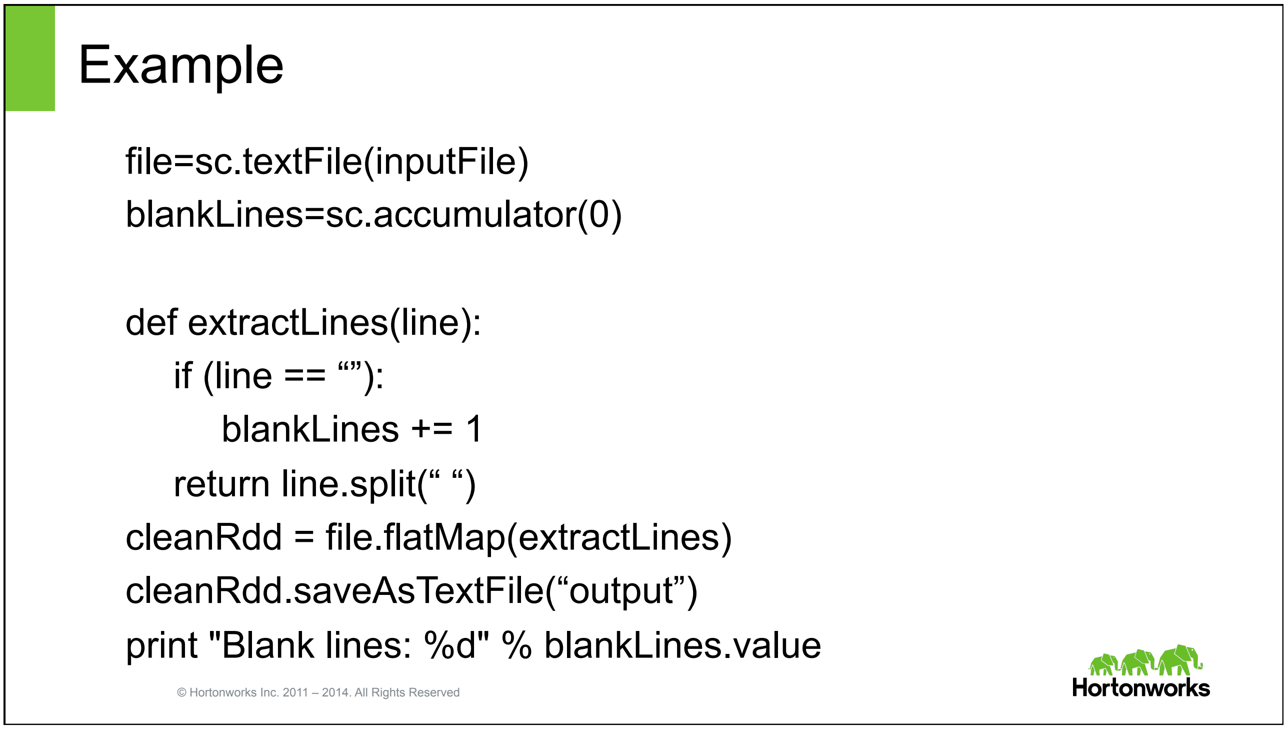


Accumulators

- Shared variable that exists on all executors
- Provides a simple syntax for aggregating values

- Common use cases
  - Count events that occur during job execution for debugging
  - Judging if an application has “passed” well enough

Hortonworks



Example

```
file=sc.textFile(inputFile)
blankLines=sc.accumulator(0)

def extractLines(line):
    if (line == “”):
        blankLines += 1
    return line.split(“ “)
cleanRdd = file.flatMap(extractLines)
cleanRdd.saveAsTextFile(“output”)
print "Blank lines: %d" % blankLines.value
```

Hortonworks

Accumulator Notes

- Created in the driver
    - SparkContext.accumulator(0)
    - SparkContext.accumulator(0.0)
- Worker code in Spark can add to them with +=
- The driver program is the only one that has access to the value

Broadcast Variables

- Allow the application to efficiently send a large, read-only value to all the worker node for use in one or more Spark operations
- Useful when an application needs to send a large, read-only look up table to all nodes
- With broad cast variables, we send the table to a node once, and multiple tasks can reference, avoiding costly suffling of data across the network

Broadcast example

```
rdd = sc.textFile(input.txt).map(….)…
toBroadcast = //some dictonary
dictbc = sc.broadcast(toBroadCast)
lookuprdd = rdd.map(lambda (key, value):
    (key, dictbc.value[value])))
```

Hortonworks

Conclusion and Key Points

Hortonworks