Week 14 Discussion

Ben Arancibia

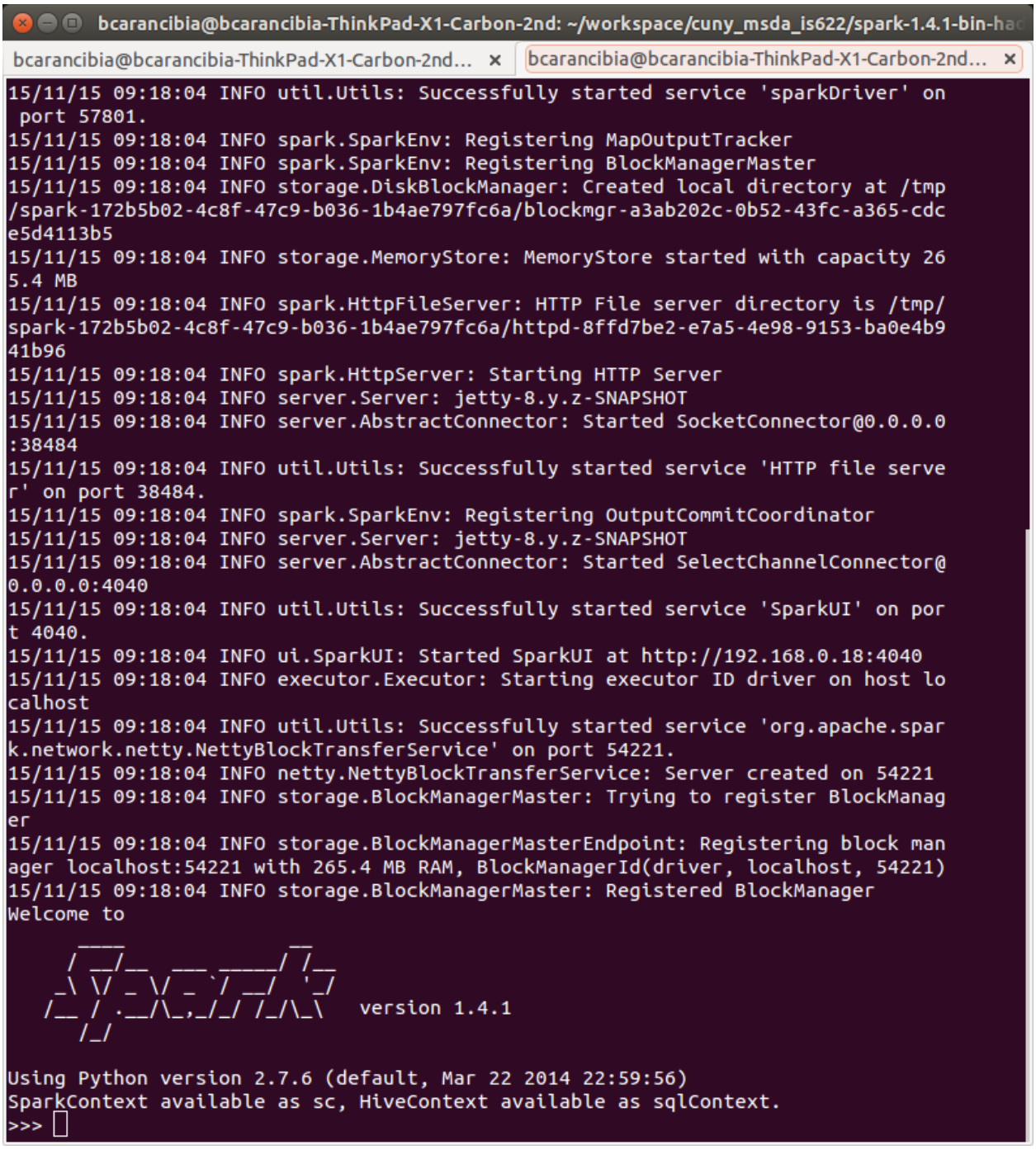
November 28, 2015

Implement the example in Section 10.7.4 using Hadoop, Spark, or my mapreduce emulator. For the same graph, how does the performance compare to the implementation you wrote for Exercise 10.7.1? NOTE: Do not answer question 10.7.4 in Section 10.7.6. The question starts with:

For a very large graph, we want to use parallelism to speed the computation.We can express triangle-finding as a multiway join and use the technique ofSection 2.5.3 to optimize the use of a single MapReduce job to count triangles.It turns out that this use is one where the multiway join technique of that sectionis generally much more efficient than taking two two-way joins. Moreover, thetotal execution time of the parallel algorithm is essentially the same as theexecution time on a single processor using the algorithm of Section 10.7.2.

Comment on whether your results are better or worse than one of your classmates. What factors affect this measurement?

As a note, this work was done in the PySpark Interactive shell because so far I have been unsuccessful in linking Spark to Ipython/Jupyter notebooks. To account for this code was will be shown below




```
import time
import numpy as np
import pandas as pd
from pyspark.sql import SQLContext, Row
sqlContext = SQLContext(sc)

edges = pd.DataFrame(
    np.array([["T1", "U1"], ["T1", "W1"], ["T2", "U1"], ["T2", "U2"], ["T2", "W1"],
              ["T2", "W2"],["T2", "W3"], ["T3", "U1"], ["T3", "U2"], ["T3", "W2"],
```

```
              ["T4", "U2"], ["T4", "W2"],["T4", "W3"], ["U1", "W1"], ["U1", "W2"],
              ["U2", "W2"], ["U2", "W3"]]))

edges.columns = ["A", "B"]
```

After creating the matrix read them into Spark and then clean up the RDD.

```
edge.columns = ["A", "B"]


edge.columns = ["B", "C"]


edge.columns = ["A", "D"]

rdd_one = sqlContext.createDataFrame(edges)
rdd_two = sqlContext.createDataFrame(edges)
rdd_three = sqlContext.createDataFrame(edges)

sdf = rdd_one.join(rdd_two, on=["B"]).join(rdd_three, on=["A"]).filter("C = D")
```

Get the triangles

```
pdf = sdf.toPandas()
pdf = pdf[["A", "B", "C"]]
pdf.columns = ["Vertex 1", "Vertex 2", "Vertex 3"]
pdf
```

Vertex 1	Vertex 2	Vertex 3
T1	U1	W1
T2	U1	W1
T2	U1	W2
T2	U2	W2
T2	U2	W3
T3	U1	W2
T3	U2	W2
T4	U2	W2
T4	U2	W3

```
num_iters = 1000
times = []
for i in range(num_iters):
    start = time.time()
    sdf = rdd_one.join(rdd_two, on=["B"]).join(rdd_three, on=["A"]).filter("C = D")
    end = time.time()
    times.append(end - start)
```

```
#hidden from this pdf because of amount of iterations
    print((end - start))

#print mean
print(np.mean(times))
```

Mean: 0.278614