

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 of Section 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 section is generally much more efficient than taking two two-way joins. Moreover, the total execution time of the parallel algorithm is essentially the same as the execution 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

```
bcarancibia@bcarancibia-ThinkPad-X1-Carbon-2nd: ~/workspace/cuny_msda_is622/spark-1.4.1-bin-hac
bcarancibia@bcarancibia-ThinkPad-X1-Carbon-2nd... x bcarancibia@bcarancibia-ThinkPad-X1-Carbon-2nd... x
15/11/15 09:18:04 INFO util.Utils: Successfully started service 'sparkDriver' on
port 57801.
15/11/15 09:18:04 INFO spark.SparkEnv: Registering MapOutputTracker
15/11/15 09:18:04 INFO spark.SparkEnv: Registering BlockManagerMaster
15/11/15 09:18:04 INFO storage.DiskBlockManager: Created local directory at /tmp
/spark-172b5b02-4c8f-47c9-b036-1b4ae797fc6a/blockmgr-a3ab202c-0b52-43fc-a365-cdc
e5d4113b5
15/11/15 09:18:04 INFO storage.MemoryStore: MemoryStore started with capacity 26
5.4 MB
15/11/15 09:18:04 INFO spark.HttpFileServer: HTTP File server directory is /tmp/
spark-172b5b02-4c8f-47c9-b036-1b4ae797fc6a/httpd-8ffd7be2-e7a5-4e98-9153-ba0e4b9
41b96
15/11/15 09:18:04 INFO spark.HttpServer: Starting HTTP Server
15/11/15 09:18:04 INFO server.Server: jetty-8.y.z-SNAPSHOT
15/11/15 09:18:04 INFO server.AbstractConnector: Started SocketConnector@0.0.0.0
:38484
15/11/15 09:18:04 INFO util.Utils: Successfully started service 'HTTP file serve
r' on port 38484.
15/11/15 09:18:04 INFO spark.SparkEnv: Registering OutputCommitCoordinator
15/11/15 09:18:04 INFO server.Server: jetty-8.y.z-SNAPSHOT
15/11/15 09:18:04 INFO server.AbstractConnector: Started SelectChannelConnector@
0.0.0.0:4040
15/11/15 09:18:04 INFO util.Utils: Successfully started service 'SparkUI' on por
t 4040.
15/11/15 09:18:04 INFO ui.SparkUI: Started SparkUI at http://192.168.0.18:4040
15/11/15 09:18:04 INFO executor.Executor: Starting executor ID driver on host lo
calhost
15/11/15 09:18:04 INFO util.Utils: Successfully started service 'org.apache.spar
k.network.netty.NettyBlockTransferService' on port 54221.
15/11/15 09:18:04 INFO netty.NettyBlockTransferService: Server created on 54221
15/11/15 09:18:04 INFO storage.BlockManagerMaster: Trying to register BlockManag
er
15/11/15 09:18:04 INFO storage.BlockManagerMasterEndpoint: Registering block man
ager localhost:54221 with 265.4 MB RAM, BlockManagerId(driver, localhost, 54221)
15/11/15 09:18:04 INFO storage.BlockManagerMaster: Registered BlockManager
Welcome to

  ____  _
 / ___|| | | |
| |___| |_| |
|___|  \___|_|_|
version 1.4.1

Using Python version 2.7.6 (default, Mar 22 2014 22:59:56)
SparkContext available as sc, HiveContext available as sqlContext.
>>> □
```

```
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