1. **Pagerank**

Code:

[cloudera@quickstart ~]$ spark-shell

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel).

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 1.6.0

/\_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0\_67)

Type in expressions to have them evaluated.

Type: help for more information.

20/11/16 22:46:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

20/11/16 22:46:07 WARN util.Utils: Your hostname, quickstart.cloudera resolves to a loopback address: 127.0.0.1; using 192.168.186.128 instead (on interface eth2)

20/11/16 22:46:07 WARN util.Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address

Spark context available as sc (master = local[\*], app id = local-1605595579062).

20/11/16 22:46:46 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.

SQL context available as sqlContext.

scala> import org.apache.spark.\_

import org.apache.spark.\_

scala> import org.apache.spark.graphx.\_

import org.apache.spark.graphx.\_

scala> import org.apache.spark.rdd.RDD

import org.apache.spark.rdd.RDD

scala> import java.util.Calendar

import java.util.Calendar

scala> val graph = GraphLoader.edgeListFile(sc, "/user/cloudera/data/page-rank-yt-data.txt")

graph: org.apache.spark.graphx.Graph[Int,Int] = org.apache.spark.graphx.impl.GraphImpl@1e5c61e6

scala> val vertexCount = graph.numVertices

vertexCount: Long = 1134890

scala> val vertices = graph.vertices

vertices: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[10] at RDD at VertexRDD.scala:57

scala> vertices.count()

res0: Long = 1134890

scala> val edgeCount = graph.numEdges

edgeCount: Long = 2987624

scala> val edges = graph.edges

edges: org.apache.spark.graphx.EdgeRDD[Int] = EdgeRDDImpl[4] at RDD at EdgeRDD.scala:40

scala> edges.count()

res1: Long = 2987624

scala> val triplets = graph.triplets

triplets: org.apache.spark.rdd.RDD[org.apache.spark.graphx.EdgeTriplet[Int,Int]] = MapPartitionsRDD[19] at mapPartitions at GraphImpl.scala:51

scala> triplets.count()

res2: Long = 2987624

scala> triplets.take(5)

res3: Array[org.apache.spark.graphx.EdgeTriplet[Int,Int]] = Array(((1,1),(2,1),1), ((1,1),(3,1),1), ((1,1),(4,1),1), ((1,1),(5,1),1), ((1,1),(6,1),1))

scala> val inDegrees = graph.inDegrees

inDegrees: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[23] at RDD at VertexRDD.scala:57

scala> inDegrees.collect()

res4: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1080592,1), (672890,5), (113843,3), (194402,1), (651447,1), (913363,1), (34207,5), (1092269,1), (332918,1), (1133078,1), (1056524,1), (170792,14), (875847,2), (375199,1), (386896,1), (691634,1), (291526,6), (1052416,1), (38926,7), (956625,1), (211245,2), (832861,1), (46781,2), (226602,5), (714321,7), (284450,2), (162877,1), (117509,4), (324206,3), (443801,1), (444564,1), (1042867,1), (377471,8), (1055187,2), (658135,6), (1139381,1), (571157,1), (313834,1), (610189,1), (1017371,1), (585693,1), (30114,60), (333678,1), (769382,3), (551291,4), (947251,1), (884124,1), (409494,1), (558999,1), (884977,1), (1005400,3), (235627,7), (282554,1), (147144,2), (199027,6), (345802,1), (1087488,1), (65463,1), (848724,2), (167454,1), (748293,...

scala>

scala> val outDegrees = graph.outDegrees

outDegrees: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[27] at RDD at VertexRDD.scala:57

scala> outDegrees.collect()

res5: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((672890,11), (113843,15), (194402,1), (34207,7), (1092269,1), (170792,5), (291526,2), (38926,3), (211245,4), (832861,1), (226602,6), (714321,1), (284450,1), (117509,2), (324206,7), (377471,1), (1055187,2), (585693,2), (30114,267), (551291,9), (947251,1), (641978,1), (1005400,2), (147144,1), (199027,8), (848724,2), (780402,1), (523064,1), (1048547,3), (513561,2), (412987,2), (753119,2), (435368,2), (243731,3), (36945,1), (331968,1), (260680,3), (710672,1), (763757,2), (519003,1), (259964,1), (106814,2), (444418,2), (186615,2), (57091,1), (61979,1), (711782,1), (31021,4), (582207,2), (349628,6), (324651,1), (223771,15), (313693,3), (472311,3), (67480,2), (131123,3), (944817,1), (332058,1), (32420,3), (936911,1), (332250,1), (291...

scala>

scala> val degrees = graph.degrees

degrees: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[31] at RDD at VertexRDD.scala:57

scala> degrees.collect()

res6: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((1080592,1), (672890,16), (113843,18), (194402,2), (651447,1), (913363,1), (34207,12), (1092269,2), (332918,1), (1133078,1), (1056524,1), (170792,19), (875847,2), (375199,1), (386896,1), (691634,1), (291526,8), (1052416,1), (38926,10), (956625,1), (211245,6), (832861,2), (46781,2), (226602,11), (714321,8), (284450,3), (162877,1), (117509,6), (324206,10), (443801,1), (444564,1), (1042867,1), (377471,9), (1055187,4), (658135,6), (1139381,1), (571157,1), (313834,1), (610189,1), (1017371,1), (585693,3), (30114,327), (333678,1), (769382,3), (551291,13), (947251,2), (641978,1), (884124,1), (409494,1), (558999,1), (884977,1), (1005400,5), (235627,7), (282554,1), (147144,3), (199027,14), (345802,1), (1087488,1), (65463,1), (848724,4),...

scala>

scala> val staticPageRank = graph.staticPageRank(10)

20/11/16 23:25:46 WARN storage.MemoryStore: Not enough space to cache rdd\_168\_0 in memory! (computed 103.3 MB so far)

staticPageRank: org.apache.spark.graphx.Graph[Double,Double] = org.apache.spark.graphx.impl.GraphImpl@6cc34931

scala> staticPageRank.vertices.collect()

res7: Array[(org.apache.spark.graphx.VertexId, Double)] = Array((1080592,0.2206850947591837), (672890,0.16187775562793652), (113843,0.1603435669540214), (194402,0.19251900903662278), (651447,0.17990261227013257), (913363,0.17826526402808332), (34207,0.15140331777545338), (1092269,0.3322094984870753), (332918,0.1659423548166951), (1133078,0.1646003078115635), (1056524,0.18750826763846146), (170792,0.21454247599674936), (875847,0.2642345518482841), (375199,0.16462091518014232), (386896,0.15355172403405837), (691634,0.15002286229000292), (291526,0.17074328675617956), (1052416,0.152441783212926), (38926,0.15174734461619488), (956625,0.30422916705418807), (211245,0.163308090538168), (832861,0.15847807117386611), (46781,0.1504565186677391), (226602,0.355453478835428), (714321,0.45304910926747...

scala> Calendar.getInstance().getTime()

res8: java.util.Date = Mon Nov 16 23:27:38 PST 2020

scala> val pageRank = graph.pageRank(0.001).vertices

20/11/16 23:33:30 WARN storage.MemoryStore: Not enough space to cache rdd\_411\_0 in memory! (computed 113.7 MB so far)

pageRank: org.apache.spark.graphx.VertexRDD[Double] = VertexRDDImpl[423] at RDD at VertexRDD.scala:57

scala> Calendar.getInstance().getTime()

res9: java.util.Date = Mon Nov 16 23:36:37 PST 2020

Output:

**scala> println(pageRank.top(5).mkString("\n"))**

**(1157827,0.3521975420826442)**

**(1157826,0.34313248965268933)**

**(1157825,0.1893125)**

**(1157824,0.1893125)**

**(1157823,0.2693013467442329)**

**Most popular communities**

1. **Triangle counting**

Code:

scala> val graph = GraphLoader.edgeListFile(sc,"/user/cloudera/data/triangle-count-fb-data.txt")

graph: org.apache.spark.graphx.Graph[Int,Int] = org.apache.spark.graphx.impl.GraphImpl@2748cdb1

println("Number of vertices : " + graph.vertices.count())

Number of vertices : 4039

println("Number of edges : " + graph.edges.count())

Number of edges : 88234

scala> graph.vertices.foreach(v => println(v))

**(777,1)**

**(2317,1)**

**(468,1)**

**(1312,1)**

**(3580,1)**

**(3654,1)**

**(1137,1)**

**(1757,1)**

**(3393,1)**

**...**

scala>

scala> val tc = graph.triangleCount()

tc: org.apache.spark.graphx.Graph[Int,Int] = org.apache.spark.graphx.impl.GraphImpl@6276200b

scala> tc.vertices.collect

res16: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((384,5), (1084,73), (3702,374), (3007,129), (667,3), (1053,11), (1894,1051), (2493,12), (1325,101), (3517,97), (3877,558), (140,49), (204,136), (956,37), (291,240), (1,57), (3809,74), (3973,24), (3706,188), (2334,7139), (755,103), (1813,5145), (2335,230), (2506,6129), (1393,691), (450,148), (2021,396), (160,1), (1596,1793), (1780,77), (3165,14), (2797,233), (2346,451), (3932,25), (2117,4339), (2904,734), (355,354), (1500,217), (3449,555), (347,17), (548,224), (2351,4066), (2541,1), (1732,741), (3903,102), (2602,14207), (2325,1174), (1550,550), (528,51), (2938,61), (1716,1102), (453,5), (2912,1199), (1310,253), (1031,5), (2192,569), (2718,220), (1202,11), (196,56), (2706,1055), (1377,8146), (3118,328), (1005,83), (3296,327), (...

scala>

scala> println("tc: " + tc.vertices.take(5).mkString("\n"));

tc: (384,5)

(1084,73)

(3702,374)

(3007,129)

(667,3)

Output:

**scala> println("Triangle counts: " + graph.connectedComponents.triangleCount().vertices.top(5).mkString("\n"));**

**Triangle counts: (4038,20)**

**(4037,4)**

**(4036,1)**

**(4035,0)**

**(4034,1)**

scala> val sum = tc.vertices.map(a => a.\_2).reduce((a, b) => a + b)

sum: Int = 4836030