GraphX

Analyse GitHub archives using GraphX

Trying to detect open source communies based on contributions

Setup the environment to work with GraphX and Json data

```
:local-repo /tmp/spark-notebook/
```

```
Repo changed to /tmp/spark-notebook!
```

```
:dp  org.apache.spark % spark-graphx_2.10 % 1.2.1
- org.apache.spark % spark-core_2.10 % _
- org.apache.hadoop % _ % _
com.fasterxml.jackson.module   %  jackson-module-scala_2.10     %    2
```

17 items

string value
/tmp/spark-notebook/cache/com.google.code.findbugs/jsr305/jars/jsr305-2.0.1.jar
/tmp/spark-notebook/cache/org.apache.spark/spark-graphx_2.10/jars/sparl graphx_2.10-1.2.1.jar
/tmp/spark-notebook/cache/org.eclipse.jetty/jetty-server/jars/jetty-server-8.1.14.v20131031.jar
/tmp/spark-notebook/cache/org.jblas/jblas/jars/jblas-1.2.3.jar
/tmp/spark-notebook/cache/com.thoughtworks.paranamer/paranamer/jars/paranamer-2.6
/tmp/spark-notebook/cache/org.eclipse.jetty.orbit/javax.servlet/orbits/javax.serv] 3.0.0.v201112011016.jar
/tmp/spark-notebook/cache/org.eclipse.jetty/jetty-continuation/jars/je continuation-8.1.14.v20131031.jar
/tmp/spark-notebook/cache/com.google.guava/guava/jars/guava-13.0.1.jar
/tmp/spark-notebook/cache/com.fasterxml.jackson.core/jackson-

core/bundles/jackson-core-2.3.3.jar
/tmp/spark-notebook/cache/com.fasterxml.jackson.core/jackson- annotations/bundles/jackson-annotations-2.3.3.jar
/tmp/spark-notebook/cache/org.eclipse.jetty/jetty-util/jars/jetty-util- 8.1.14.v20131031.jar
/tmp/spark-notebook/cache/com.fasterxml.jackson.module/jackson-module- scala_2.10/bundles/jackson-module-scala_2.10-2.3.3.jar
/tmp/spark-notebook/cache/org.scala-lang/scala-library/jars/scala-libr 2.10.4.jar
/tmp/spark-notebook/cache/com.fasterxml.jackson.core/jackson- databind/bundles/jackson-databind-2.3.3.jar
/tmp/spark-notebook/cache/org.eclipse.jetty/jetty-http/jars/jetty-http- 8.1.14.v20131031.jar
/tmp/spark-notebook/cache/org.eclipse.jetty/jetty-io/jars/jetty-io- 8.1.14.v20131031.jar
/tmp/spark-notebook/cache/org.spark-project.spark/unused/jars/unused- 1.0.0.jar

Import some github data

```
import sys.process._
if (!new java.io.File("/tmp/github.json").exists) {
  new java.net.URL("http://data.githubarchive.org/2015-01-01-15.json.g

  Seq("gunzip", "-f", "/tmp/github.json.gz")!!
}
```

The size of the data

```
:sh du -h /tmp/github.json
```

```
 25M    /tmp/github.json
```

First some Spark manipulation

```
val raw = sparkContext.textFile("/tmp/github.json")
```

MapPartitionsRDD[1] at textFile at <console>:47

The number of lines in the file

```
raw.count
```

11351

Convert line to JSON *(simple Map of Maps)*

```
val json = raw.mapPartitions{ lines =>
  import com.fasterxml.jackson._
  import com.fasterxml.jackson.core._
  import com.fasterxml.jackson.databind._
  import com.fasterxml.jackson.module.scala._
  val mapper = new ObjectMapper()
  mapper.registerModule(DefaultScalaModule)
  lines.map(x => mapper.readValue(x, classOf[Map[String,Any]]))
}
```

MapPartitionsRDD[2] at mapPartitions at <console>:49

Let's look at the two first rows

```
json.take(2).toList
```

2 items

bitmap	elems	size0
84025377	[Lscala.collection.immutable.HashMap;@35130a63	7
84025377	[Lscala.collection.immutable.HashMap;@6ad93582	7

The graph part

We could use the *actors* and the *repos* as vertices, and use the *event* as relationship between them.

There are *id*s for actor and repo, so we can directly use them in GraphX as such.

```
import org.apache.spark.rdd._
import org.apache.spark.graphx._
```

RDD vertices {Actors U Repos}

```
val actors:RDD[(VertexId, (Short, String))] = json.map{ x =>
  val actor = x("actor").asInstanceOf[Map[String, Any]]
  val id = actor("id").toString.toLong
  val login = actor("login").toString
  (id, (0, login))
}
val repos:RDD[(VertexId, (Short, String))] = json.map{ x =>
  val repo = x("repo").asInstanceOf[Map[String, Any]]
  val id = repo("id").toString.toLong
  val name = repo("name").toString
  (id, (1, name))
}
val vertices:RDD[(VertexId, (Short, String))] = actors union repos
```

UnionRDD[5] at union at <console>:69

RDD of Edges

Now an **RDD** with the edges (including reverse ones, that is from repo to actor)

```
// None → repo to actor
// Some("PushEvent") → actor pushed on repo
val edges:RDD[Edge[Option[String]]] = json.flatMap { x =>
  val event = x.get("type").map(_.toString)
  val actor = x("actor").asInstanceOf[Map[String, Any]]("id").toString
  val repo = x("repo").asInstanceOf[Map[String, Any]]("id").toString.t
  List(Edge(actor, repo, event), Edge(actor, repo, None))
}
```

MapPartitionsRDD[6] at flatMap at <console>:59

Graph

```
val graph = Graph(vertices, edges)
```

org.apache.spark.graphx.impl.GraphImpl@3de0c636

Open source working community

A very very simple example of such extraction would simply be to extract the connected components

So that, a component is the actors and repos having connections between them but not with other actor or repos. A connection being a collaboration.

Computing connected components

```
val cc = graph.connectedComponents
```

org.apache.spark.graphx.impl.GraphImpl@40e7a90

The `cc` variable is the original graph but vertives' payload/properties is only the cluster to which is belongs. The cluster is characterized by the smallest `VertexId` in the cluster.

Number of connected components

Computing the number of clusters can easily be done by counting the number of distinct `payload` for the vertices.

```
<strong style="color: red">{cc.vertices.map(_._2).distinct.count}</str
```

4774

Clusters by language

We can try to concentrate our analysis to specific languages, since we don't have the language information in the events data (we need extra call to the GitHub API for that) we'll take a naive approach, that is, **we'll only consider the repo having the language in their name** -- albeit it's not 100% safe.

Utility functions

The following function compute retrieves the cluster for a given cluster.

```
import org.apache.spark.SparkContext._
def cluster(lgg:String) = {
  // collect all repos for the language `lgg`
  val lggRepos:List[(VertexId, (Short, String))] = vertices.filter { x
                    x._2._1 /*vertex type*/ == 1 /*repo*/ &&
                    x._2._2/*repo name*/.toLowerCase.contains(lgg) //h
                }.collect().toList
  // keep only the set
  // ***** IN A CLUSTER →→→ THIS NEEDS TO BE A BROADCAST VARIABLE ****
  val lggRepoIds:List[Long] = lggRepos.map(_._1).distinct
  // clusters "id" for these repos → BROADCAST
  val clusterIds:List[Long] = cc.vertices.filter(x => lggRepoIds.conta
                              .map(_._2)
                              .collect()
                              .toList
  // return the vertices being clustered sorted by decreasing cardinal
  val clusters:List[(Long, Iterable[Long])] = cc.vertices.filter{ x =>
                .groupBy(_._2)
                .mapValues(_.map(_._1))
                .collect().toList
                .sortBy(_._2.size)
                .reverse
  clusters
}
```

Shows the list of repos and actors included in the given cluster

```
def showCluster(lgg:String, clusterIds:List[Long]) = {
  val c = graph.vertices
               .filter(x => clusterIds.contains(x._1))
               .collect().toList

  <div>
  <p><strong>Repos</strong></p>
  <ul>{
  c.collect { case (x, (1, r)) =>
           //show the repo
           val t = if (r.toLowerCase.contains(lgg)) <strong style="col
             <li><a href={"http://github.com/"+r}>{t}</a></li>
          }
  }</ul>
  <p><strong>Users</strong></p>
  <ul>{
  c.collect { case (x, (0, n)) =>
           //show the repo
             <li><a href={"http://github.com/"+n}>{n}</a></li>
          }
  }</ul>
  </div>
}
```

Javascript

```
val js = cluster("js")
```

Let's look at the 3 biggest clusters

```
layout(3, js.take(3).map(r => html(showCluster("js", r._2.toList))))
```

Repos

- slackhq/SlackTextViewController (http://github.com/slackhq/SlackTextViewController)
- emirozer/fake2db (http://github.com/emirozer/fake2db)
- josephmisiti/awesome-machine-learning (http://github.com/josephmisiti/awesome-machine-learning)
- zhxnlai/ZLSwipeableView (http://github.com/zhxnlai/ZLSwipeableView)
- ddeboer/vatin (http://github.com/ddeboer/vatin)
- omergul123/LLSimpleCamera (http://github.com/omergul123/LLSimpleCamera)
- lexrus/VPNOn (http://github.com/lexrus/VPNOn)
- clockfly/gearpump (http://github.com/clockfly/gearpump)
- eigengo/lift (http://github.com/eigengo/lift)
- papers-we-love/papers-we-love (http://github.com/papers-we-

Repos

- apach (http://
- mwcli (http://
- spring (http:// projec
- alexz- (http://
- spring (http:// projec
- spring (http:/

Scala

```
val scala = cluster("scala")
```

```
layout(4, scala.map(r => html(showCluster("scala", r._2.toList))))
```

Repos • **scalaz/scalaz** (http://github.com/scalaz/scalaz) • pavelfatin/patterns (http://github.com/pavelfatin/patterns) **Users** • 0x414c (http://github.com/0x414c)	**Repos** • **itsvenkis/scala** (http://github.co • **itsvenkis/scala** (http://github.co **Users** • itsvenkis (http://github.com/itsve
Repos • **ummels/scala-prioritymap** (http://github.com/ummels/scala-prioritymap) **Users** • ummels (http://github.com/ummels)	**Repos** • **Scalarm/scalarm_information_** (http://github.com/Scalarm/scala **Users** • kliput (http://github.com/kliput)
Repos • **ornicar/scalachess** (http://github.com/ornicar/scalachess) **Users** • Happy0 (http://github.com/Happy0)	**Repos** • **inc-lc/ilc-scala** (http://github.co **Users** • Blaisorblade (http://github.com/E

Spark ^^

```
val spark = cluster("spark")
```

```
layout(4, spark.map(r => html(showCluster("spark", r._2.toList))))
```