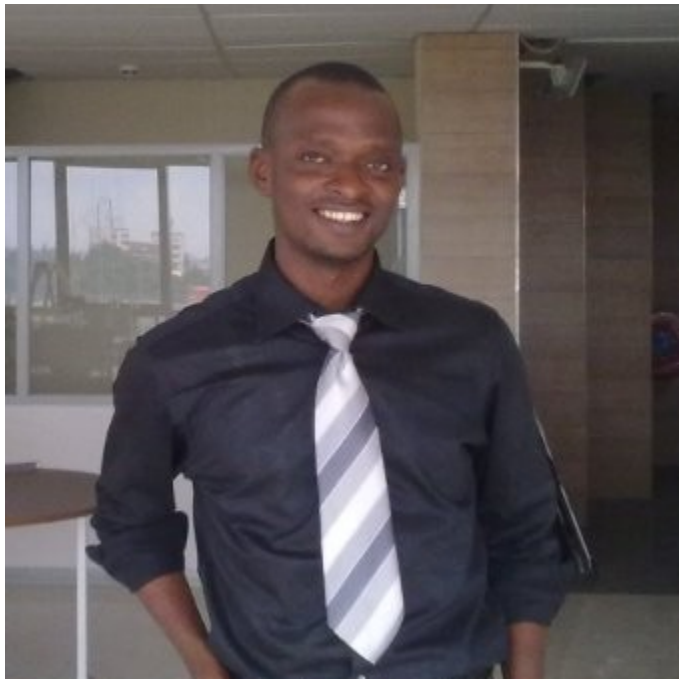
Analysis of Community Interactions using Spark GraphX

Faculty: Michael Enudi

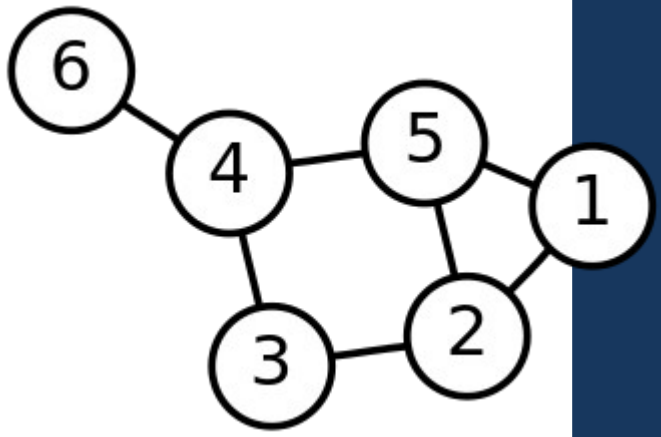
About Me.



- Lives and works in Johannesburg, South Africa
- Senior Software engineer with over 10years of working experience writing enterprise java applications, architecting data solutions.
- Cloudera Certified Spark and Hadoop Dev.
- Oracle Certified SQL Expert
- Oracle Certified Java Master
- Sun Certified Java Business Component Dev.
- Sun Certified Java Programmer
- Big data enthusiast

LinkedIn → https://www.linkedin.com/in/michaelenudi

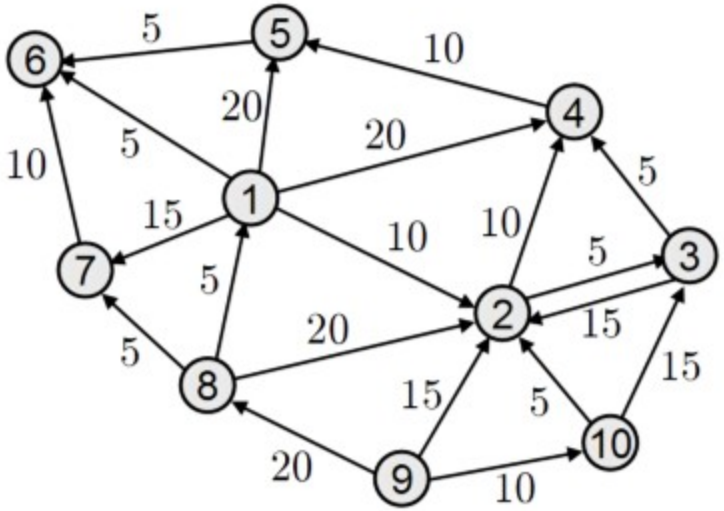
What are Graphs??



Graphs are mathematical structures used in computing to model entities (of same or different types) and their relationship.

It is based on the concept or subject of mathematics called graph theory.

The objects or entities are called Vertex/Vertices or Node/Nodes while the relationship are called Edge/Edges.



From the documentation page…

GraphX unifies ETL, exploratory analysis, and iterative graph computation within a single system. You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms using the Pregel API.

Types of Graphs

- Simple graphs
- Bipartite graphs
- Directed and Undirected graphs
- Multigraphs
- Property graphs
- See more ...

lines or curves for the edges. Graphs are one of the objects of study in discrete mathematic

The edges may be directed or undirected. For example, if the vertices represent people at
hands with a person *B* only if *B* also shakes hands with *A*. In contrast, if any edge from a p
The former type of graph is called an *undirected graph* and the edges are called *undirected*

Graphs are the basic subject studied by graph theory. The word "graph" was first used in th

Contents [hide]

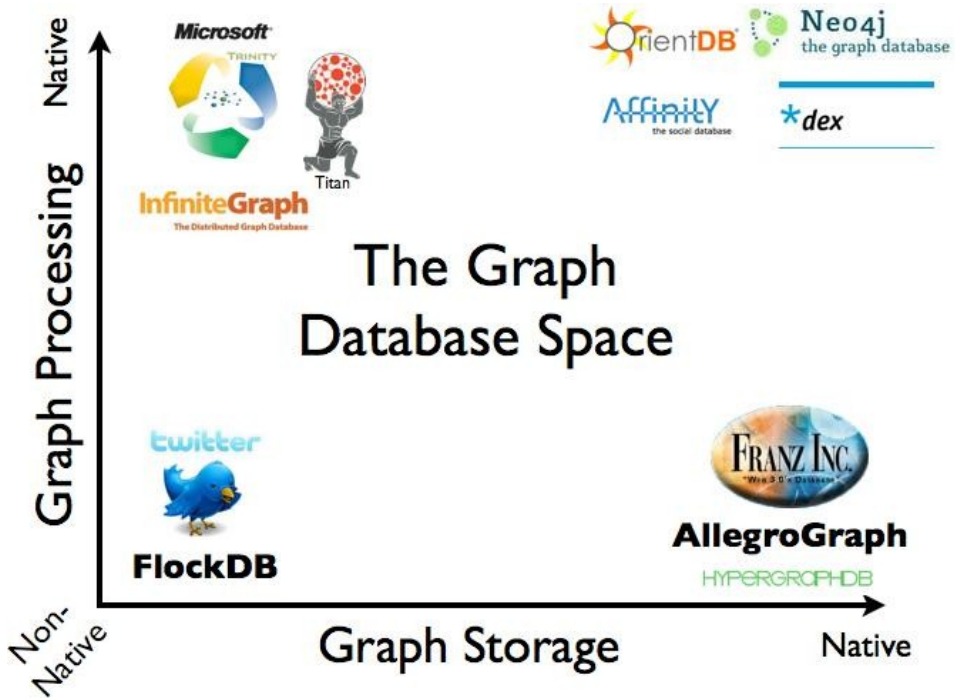
Graph storage or graph databases

A group of NoSQL databases that provide full transactional CRUD support for data modelled as graphs.

They
- Offer real time query support
- Are engineered for transactional integrity
- Can served as data sources for big data real time systems.
-

They can be used as a store for final or intermediate result of graph processing frameworks.

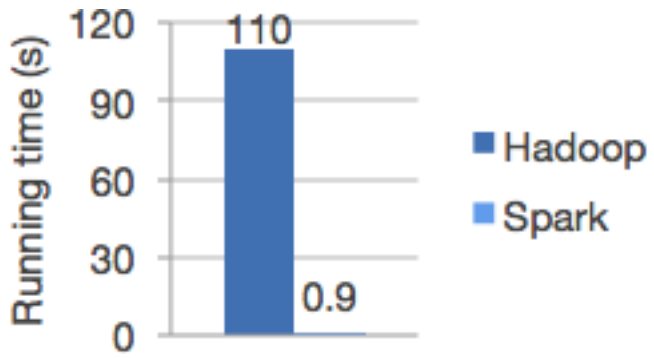
An example is MazeRunner extension that enables Spark application to read data from Neo4J database, process and store them back to the graph database.

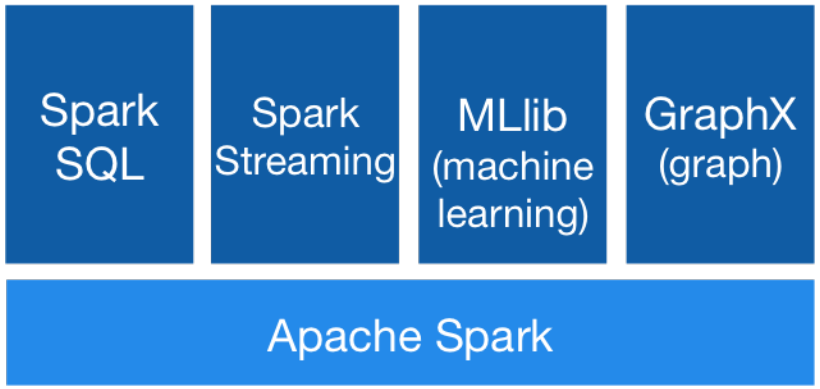


Our next hackerday will cover Modelling and Thinking in Graphs (Neo4J).

Graph processing with Spark

Apache Spark





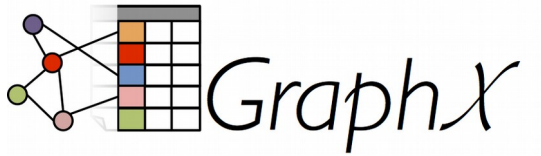
An engine for fast and efficient data processing. It is a natural success of Hadoop's MapReduce framework that comes with in-memory data structure (RDD), directed-acyclic graphs of steps in the job pipeline and multi-language platform to write data processing application on and beyond Hadoop.
It is seriously becoming a sub-ecosystem as it runs other sub framework for data processing like Spark SQL, Spark Stream, Spark ML and many more.

Scala and its build tool (Sbt)

Scala is a general purpose language that mixes various paradigms and runs on the java virtual machine (JVM). With Scala, you can perfectly mix functional and reactive styles of programming with Object-oriented programming.

SBT or Scala build tool - the maven for Scala.

Spark GraphX



GraphX is the graph processing layer on top of Spark that brings the power of Big data to Graph processing.

It extends the RDDs of Spark computing framework as well creates a new abstraction called Graph.

Graphs in GraphX are treated as directed property multigraphs. The purpose of GraphX is enabled graph processing on big data.

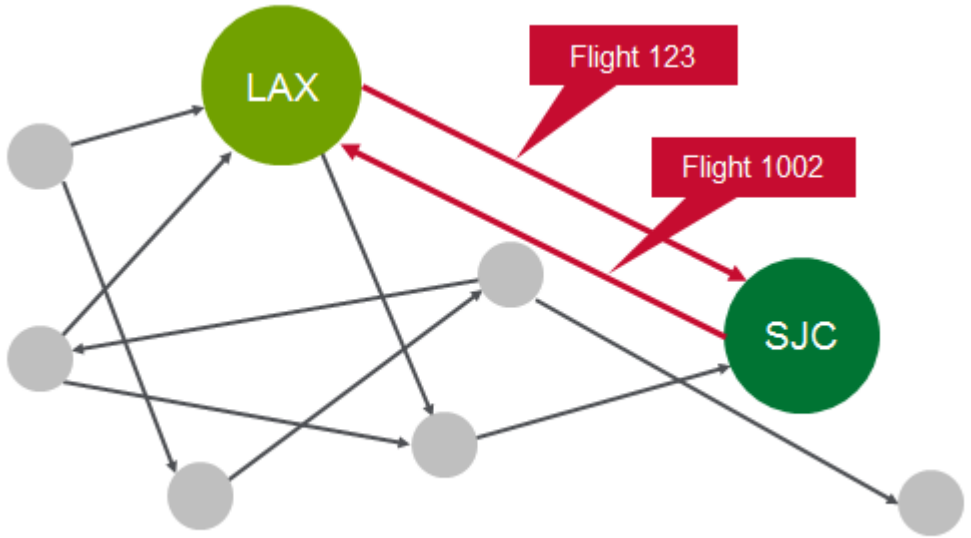
To support this processing, GraphX helps us with a some fundamental graph operators as wells built-in algorithms for graph analysis and operation

GraphX APIs

Spark graphX provides us with fundamental graph operators that make it easy to quickly assess or perform some operations on our graphs.

A sample list of operators include:

- inDegrees, outDegrees
- mapping edges, vertices or triplets
- subgraph
- aggregateMessages
- joinVertices
- groupEdges
- collectNeighbors



Also, some iterative algorithms like pageRank, triangle counting, connected components are available in Graphx as a single api call.
Thus we can perform iterative parallel-computations on big data modelled in a graph structure.

Spark GraphX Visualization

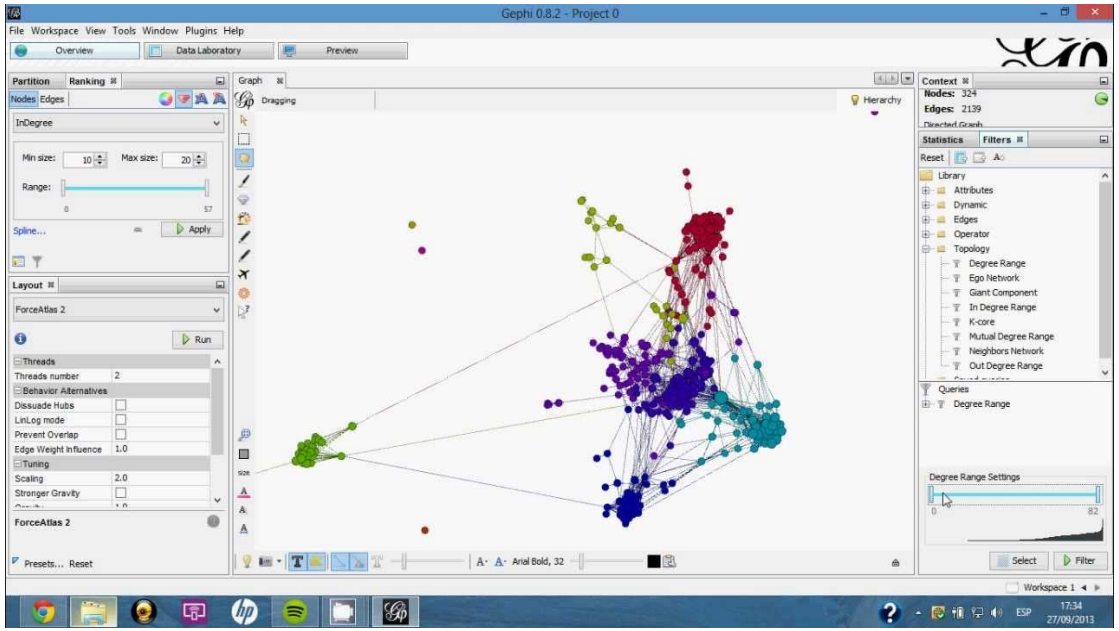
There are no visualization features with GraphX.

My thoughts

- GraphX is geared towards processing of large dataset such that visualization is not readily possible. Example, is visualizing a graph of 1 millions nodes and 200 millions edges.
-
- Just as Spark is for data processing, so is GraphX for graph processing.

To visualize graph, other tools have to be brought into the mix. They include

1. Gephi
2. Graphstream
3. Zeppelin



Case study

Requirement

Given the dataset, XYZ telecommunication company has limited resources to invest in improving their quality of service. Hence the upgrade across the city of Milan will be done in phases.

Given that it is important that we rank traffic flows across the city so as to know optimize call quality with overall target on profit, a decision has to be made as to which grid station will be upgraded in what batch. This decision is going to be made based on daily call traffic in both direction.

To make which parts of the city should benefit from the upgrade, XYZ will like to know

- the grid connections with most traffic
- the grid with most incoming and outgoing call
- the grid with the most interations to all other grid (the center of gravity)

…. and as much more information you can provide

But I am not a Data Scientist.

Where do this fit in the data processing pipeline ???

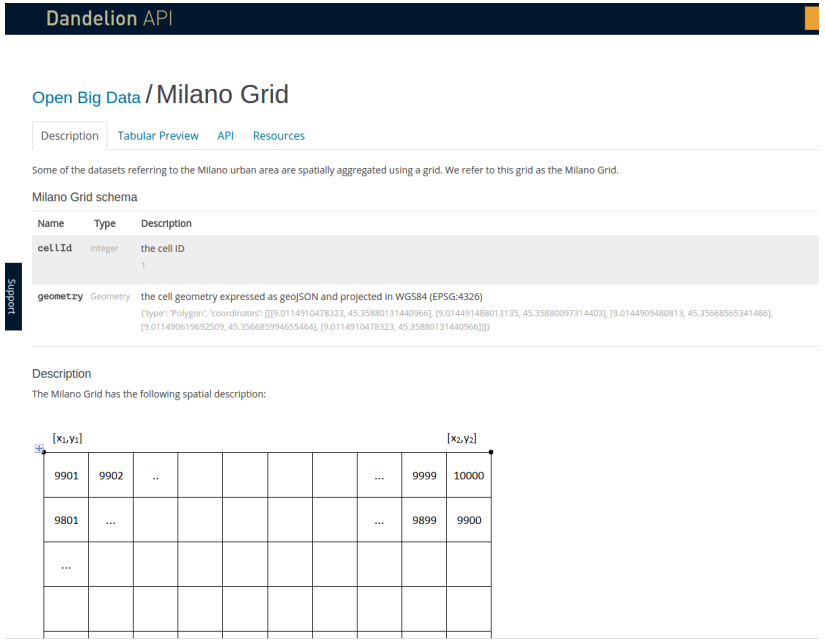
Dandelion Open Big Data link

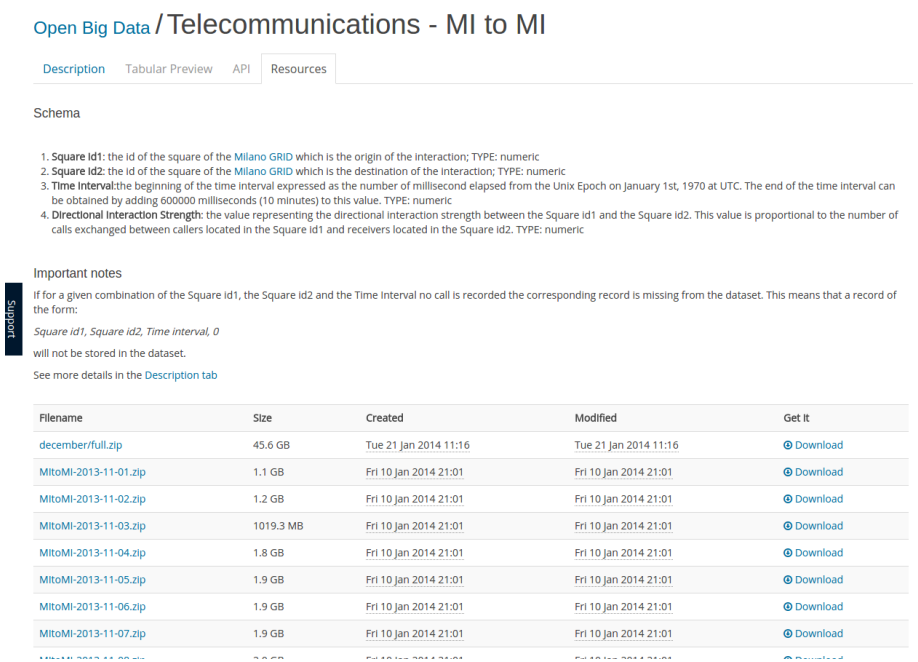
Milano Grid

Some of the datasets referring to the Milano urban area are spatially aggregated using a grid. We refer to this grid as the Milano Grid.
https://dandelion.eu/datagems/SpazioDati/milano-grid/description/

Inter-city interaction

This dataset provides information regarding the directional interaction strength between the city of Milan different areas based on the calls exchanged between Telecom Italia Mobile users.
https://dandelion.eu/datagems/SpazioDati/telecom-mi-to-mi/resource/





Dataset Analysis

We must note that we are using only a day's worth of data which in the real-world, is not worth making decision on.
We are doing this because of the sheer size.

Milano Grid dataset

The city of Milan is spatial aggregated using a grid.

The dataset for this grid is a complex, multiple-nested json collection. Each item in the json collection contain the id, cellTowerId, geometery, shape, etc for that grid.

We will use features of Spark SQL to read the json file and construct an RDD from this dataset. Each item in this dataset will serve as a node in our graph.

One practical way to think of a grid is a region i in the city that will be served by cell tower i – 1.

Dataset Analysis

We must note that we are using only a day's worth of data which in the real-world, is not worth making decision on.
We are doing this because of the sheer size.

Aggregated CDR dataset

According to the description page, this dataset provides information regarding the directional interaction strength between the city of Milan different areas based on the calls exchanged between Telecom Italia Mobile users.
The directional interaction strength between the area A and the area B is proportional to the number of calls issued from the area A to the area B.

It is a tab delimited file that contains the following fields

- Timestamp
- GridID1 for the grid with the calling mobile number
- GridID2 for the grid with the receiving mobile number
- Directional interaction strength between the Grid id1 and the Grid id2. This value is proportional to the number of calls exchanged between callers located in the Grid id1 and receivers located in the Grid id2.

```
hadmin@hadoop-ss-lab: ~/sample-data/dandelion_mitomi
hadmin@hadoop-ss-lab:~/sample-data/dandelion_mitomi$ head MItoMI-2013-11-03.txt
 -n 40
1383468000000   1       1       1.415182132451453E-4
1383474600000   1       1       3.6471276544197196E-4
1383476400000   1       1       1.5103247348560638E-4
1383477000000   1       1       3.6349656926791334E-4
1383483000000   1       1       1.478153689175954E-4
1383485400000   1       1       7.395236931786679E-5
1383499200000   1       1       2.172210647570028E-4
1383501600000   1       1       7.551623674280319E-5
1383507600000   1       1       7.22991321747922E-5
1383508200000   1       1       4.73328824751369E-6
1383510000000   1       1       7.22991321747922E-5
1383519000000   1       1       7.22991321747922E-5
1383464400000   1       10      2.814243229598746E-5
1383467400000   1       10      2.6943522612461817E-5
1383473400000   1       10      7.588125267884338E-5
1383475800000   1       10      1.8573731606172118E-4
1383489600000   1       10      1.6248613856696166E-4
1383492000000   1       10      2.591626890161992E-4
1383498000000   1       10      8.322838720443674E-5
1383498600000   1       10      1.0402368497483085E-4
1383500400000   1       10      7.925775136252494E-5
1383501000000   1       10      1.0620127397498675E-4
1383507000000   1       10      1.340085494001695E-4
1383460200000   1       1001    3.6865383222377156E-6
1383468600000   1       1001    5.63104352596541E-5
1383487200000   1       1001    6.740586716042852E-6
1383493800000   1       1001    1.5927026876965018E-4
1383496200000   1       1001    3.6865383222377156E-6
1383502200000   1       1001    5.9996973581891815E-5
1383469200000   1       1002    5.391152654393628E-5
1383472200000   1       1002    3.6865383222377156E-6
1383480600000   1       1002    3.6865383222377156E-6
1383505200000   1       1002    3.6865383222377156E-6
1383505800000   1       1002    4.314569789933472E-5
1383470400000   1       1003    5.3761298308481875E-5
1383487800000   1       1003    2.9157936000638 67E-6
1383506400000   1       1003    3.5196511189015013E-6
1383466200000   1       1004    4.6649398250341986E-5
1383488400000   1       1004    5.63104352596541E-5
1383489000000   1       1004    5.03359365725797E-5
hadmin@hadoop-ss-lab:~/sample-data/dandelion_mitomi$
```

Environment & tools

- Cloudera Quickstart VM 5.7 or 5.8
- Scala SDK and Runtime
- Scala build tool
- Gephi (optional)

Lab