**PARALLEL COMPUTATION ON COMET USING SPARK REPORT**

**ABSTRACT**

Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process. As against a common belief, **Spark is not a modified version of Hadoop** and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark. Spark uses Hadoop in two ways – one is **storage** and second is **processing**. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only. The main intention of the project is to achieve parallel Computation on COMET – A super computer situated in San Diego where you can request for three to four machines and execute your program parallelly on those machines using the framework SPARK. A large dataset named Airline data has been fetched through a program which has various fields such as Airport ID, name, routes, distance etc. A graph is created making the airports as the vertices and the edges represented the distance between the airports. The graph generated is a multigraph and several operations are performed on the graph where the main ones include PageRank, Connected Components and Subgraphs.

**INTRODUCTION**

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

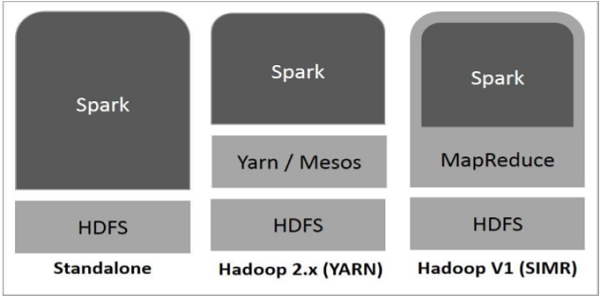
Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

**EVOLUTION OF APACHE SPARK**

Spark is one of Hadoop’s sub project developed in 2009 in UC Berkeley’s AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

**Features of Apache Spark**

Apache Spark has following features.

* Speed − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
* Supports multiple languages − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
* Advanced Analytics − Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.
* 

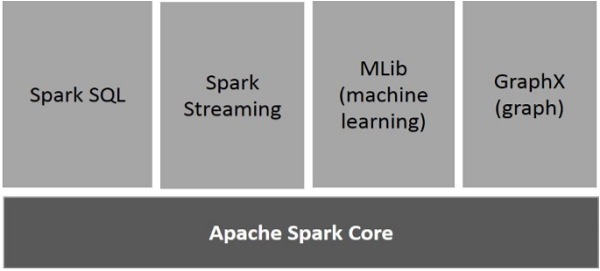
**Spark Built on Hadoop**

There are three ways of Spark deployment as explained below.

* Standalone − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
* Hadoop Yarn − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
* Spark in MapReduce (SIMR) − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

## Components of Spark

The following illustration depicts the different components of Spark.

****

**Apache Spark Core**

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

### **Spark SQL**

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

### **Spark Streaming**

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

### **MLlib (Machine Learning Library)**

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

### **GraphX**

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

### **Resilient Distributed Datasets**

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. Let us first discuss how MapReduce operations take place and why they are not so efficient.

## Iterative Operations on Spark RDD

The illustration given below shows the iterative operations on Spark RDD. It will store intermediate results in a distributed memory instead of Stable storage (Disk) and make the system faster.

Note − If the Distributed memory (RAM) is not sufficient to store intermediate results (State of the JOB), then it will store those results on the disk.



# **Overview on Graphx API of Spark**

GraphX is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark [RDD](http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.rdd.RDD) by introducing a new [Graph](http://spark.apache.org/docs/latest/graphx-programming-guide.html#property_graph) abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., [subgraph](http://spark.apache.org/docs/latest/graphx-programming-guide.html#structural_operators), [joinVertices](http://spark.apache.org/docs/latest/graphx-programming-guide.html" \l "join_operators), and [aggregateMessages](http://spark.apache.org/docs/latest/graphx-programming-guide.html" \l "aggregateMessages)) as well as an optimized variant of the [Pregel](http://spark.apache.org/docs/latest/graphx-programming-guide.html" \l "pregel) API. In addition, GraphX includes a growing collection of graph [algorithms](http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph_algorithms) and [builders](http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph_builders) to simplify graph analytics tasks.

# **The Property Graph**

The [property graph](http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.graphx.Graph) is a directed multigraph with user defined objects attached to each vertex and edge. A directed multigraph is a directed graph with potentially multiple parallel edges sharing the same source and destination vertex. The ability to support parallel edges simplifies modeling scenarios where there can be multiple relationships (e.g., co-worker and friend) between the same vertices. Each vertex is keyed by a*unique* 64-bit long identifier (VertexId). GraphX does not impose any ordering constraints on the vertex identifiers. Similarly, edges have corresponding source and destination vertex identifiers.

The property graph is parameterized over the vertex (VD) and edge (ED) types. These are the types of the objects associated with each vertex and edge respectively.

GraphX optimizes the representation of vertex and edge types when they are primitive data types (e.g., int, double, etc…) reducing the in memory footprint by storing them in specialized arrays.

In some cases it may be desirable to have vertices with different property types in the same graph. This can be accomplished through inheritance. For example to model users and products as a bipartite graph we might do the following:

# **HPC Systems**

For almost 30 years, SDSC has led the way in deploying and supporting cutting-edge high performance computing systems for a wide range of users, from the campus to the national research community. From the earliest Cray systems to today’s data-intensive systems, SDSC has focused on providing innovative architectures designed to keep pace with the changing needs of science and engineering.

Whether you’re looking to expand computing beyond your lab or a business looking for that competitive advantage, SDSC’s HPC experts will guide potential users in selecting the right resource, thereby reducing time to solution while taking your science to the next level.

**ABOUT SDSC (SAN DIEGO SUPER COMPUTER)**

As an Organized Research Unit of the [University of California, San Diego](http://www.ucsd.edu/), the San Diego Supercomputer Center is considered a leader in advanced computation and all aspects of “Big Data”, which includes data integration and storage, performance modeling, data mining and predictive analytics, software development, and more. SDSC provides resources, services, and expertise to the national research community including academia, industry, and government. SDSC supports hundreds of multidisciplinary programs spanning a wide variety of domains, from astrophysics and bioinformatics to environmental sciences and health IT.

With its two newest supercomputers, a data-intensive system called *Gordon* and *Comet*, a petascale system entering production in 2015, SDSC is a partner in XSEDE (eXtreme Science and Engineering Discovery Environment), a National Science Foundation (NSF) program that comprises the most advanced collection of integrated digital resources and services in the world. SDSC has also pioneered advances in data storage and a cloud computing, and now houses several “centers of excellence” in the areas of large-scale data management, predictive analytics, health IT services, workflow automation, and Internet analysis.

SDSC was founded in 1985 with a $170 million grant from the NSF Supercomputer Centers program. From 1997 to 2004, SDSC extended its leadership in computational science and engineering to form the National Partnership for Advanced Computational Infrastructure (NPACI), teaming with approximately 40 university partners around the country.

A broad community of scientists, engineers, students, commercial partners, museums, and other facilities work with SDSC to develop cyber infrastructure-enabled applications to help manage their extreme data needs. Projects run the gamut from creating astrophysics visualization for the American Museum of Natural History, to supporting more than 286,000 visitors per month from about 190 countries to the Protein Data Bank, or performing large-scale, award-winning simulations of the origin of the universe or how a major earthquake would affect densely populated areas such as southern California.

#### http://www.sdsc.edu/assets/images/hpc_gordon_body.jpg

Proposed System

The proposed idea computes the Page Rank of the graph within seconds when compared to the same done on a single machine. It is 10 times faster than the one run on a single machine. The same methodology is used in many of the search engines to generate the most requested page using the Page rank mechanism. PageRank (PR) is an algorithm used by [Google Search](https://en.wikipedia.org/wiki/Google_Search) to rank websites in their search engine results. PageRank was named after [Larry Page](https://en.wikipedia.org/wiki/Larry_Page), one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

PageRank is a [link analysis](https://en.wikipedia.org/wiki/Network_theory#Link_analysis) algorithm and it assigns a numerical [weighting](https://en.wikipedia.org/wiki/Weighting) to each element of a [hyperlinked](https://en.wikipedia.org/wiki/Hyperlink) [set](https://en.wikipedia.org/wiki/Set_(computer_science)) of documents, such as the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), with the purpose of "measuring" its relative importance within the set. The [algorithm](https://en.wikipedia.org/wiki/Algorithm) may be applied to any collection of entities with [reciprocal](https://en.wikipedia.org/wiki/Reciprocal_link) quotations and references. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by {\displaystyle PR(E).} Other factors like Author Rank can contribute to the importance of an entity.

A PageRank results from a mathematical algorithm based on the [webgraph](https://en.wikipedia.org/wiki/Webgraph" \o "Webgraph), created by all World Wide Web pages as nodes and [hyperlinks](https://en.wikipedia.org/wiki/Hyperlink) as edges, taking into consideration authority hubs such as [cnn.com](https://en.wikipedia.org/wiki/Cnn.com) or [usa.gov](https://en.wikipedia.org/wiki/Usa.gov). The rank value indicates an importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined [recursively](https://en.wikipedia.org/wiki/Recursion) and depends on the number and PageRank metric of all pages that link to it ("[incoming links](https://en.wikipedia.org/wiki/Incoming_link)"). A page that is linked to by many pages with high PageRank receives a high rank itself.

Steps to be followed to connect to comet

* Log into the comet
* Request for three to four machines on COMET using the command

salloc -A sun117 –nodes=3 –p compute –t 2:00:00

* Use the head node of the cluster as the master machine and connect to the head node.

ssh comet-ab-cd

* Edit the ‘slaves’ file to list the other machines

comet-ab-ef

comet-ab-gh

* Start the master and slave machines by the following command

start-master.sh

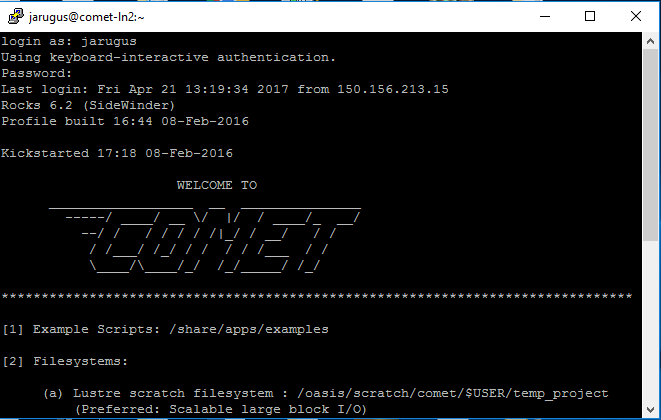
start-slaves.sh

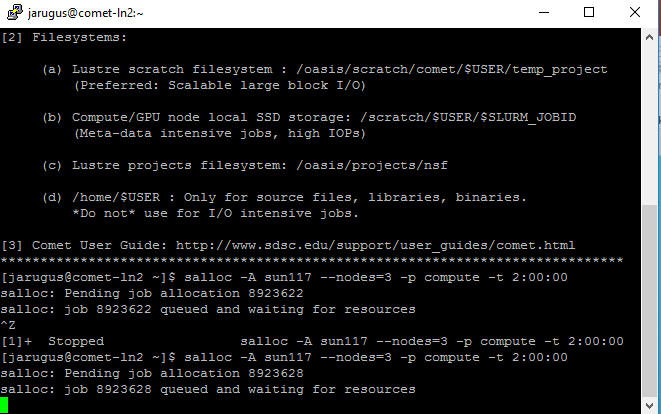
* Tunnel the web traffic using the following command

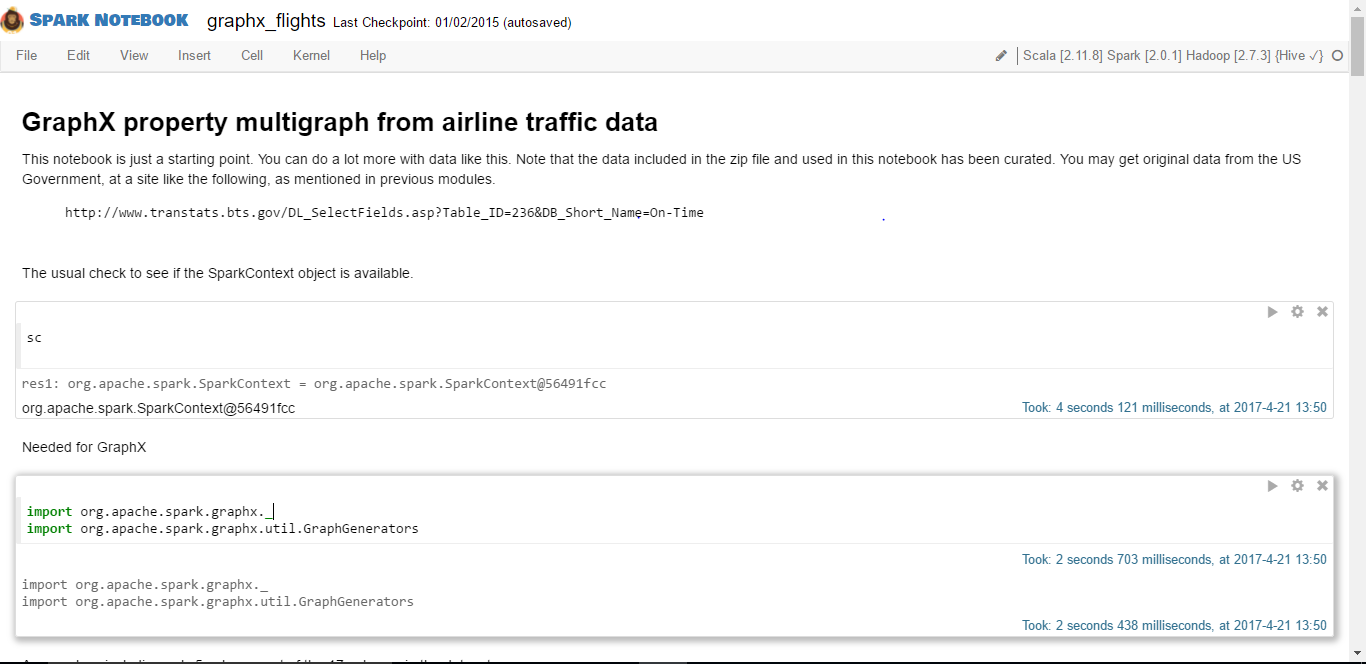
putty -L localhost:9000:localhost:9000 [you@comet-12-34.sdsc.edu](mailto:you@comet-12-34.sdsc.edu)

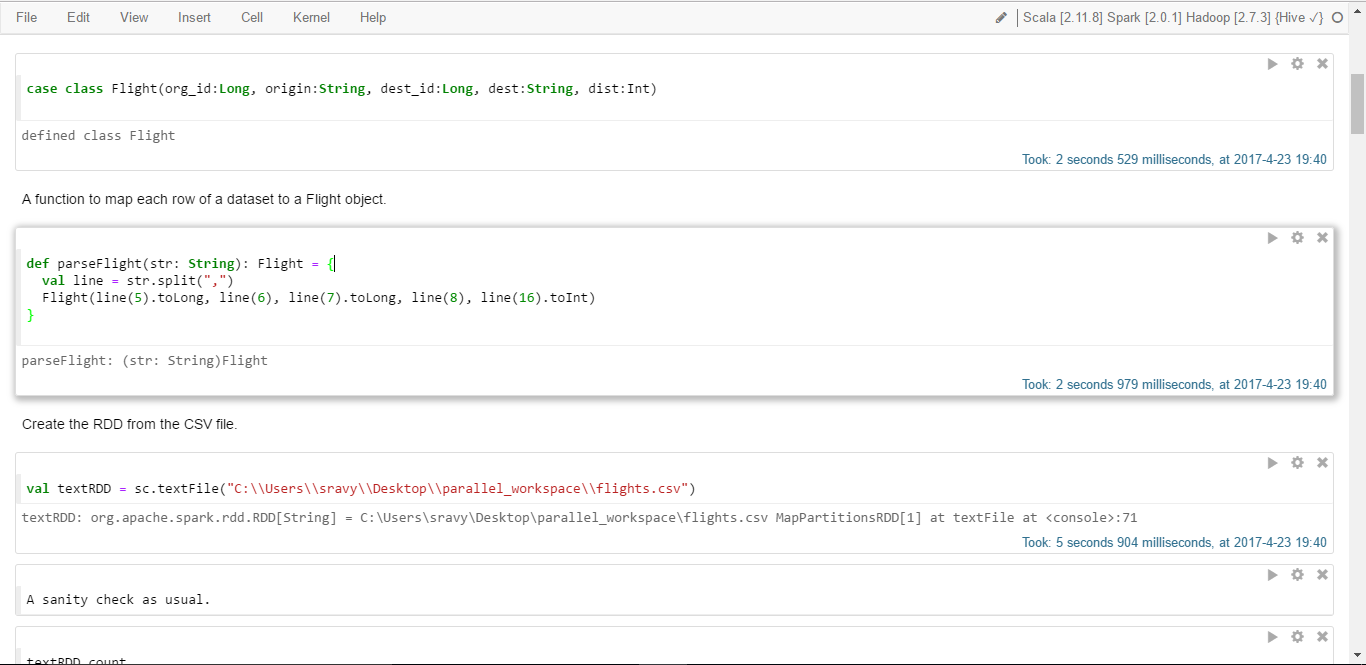
* Start Your Spark Notebook then by the command

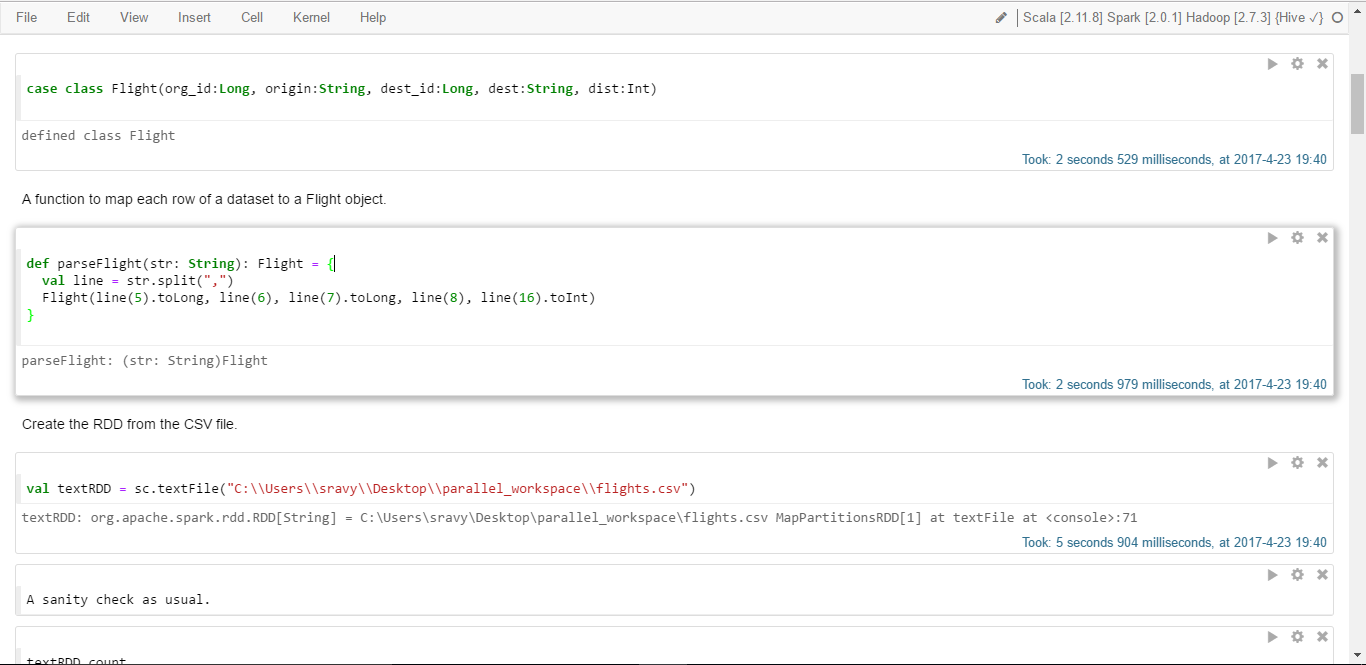
Spark-notebook



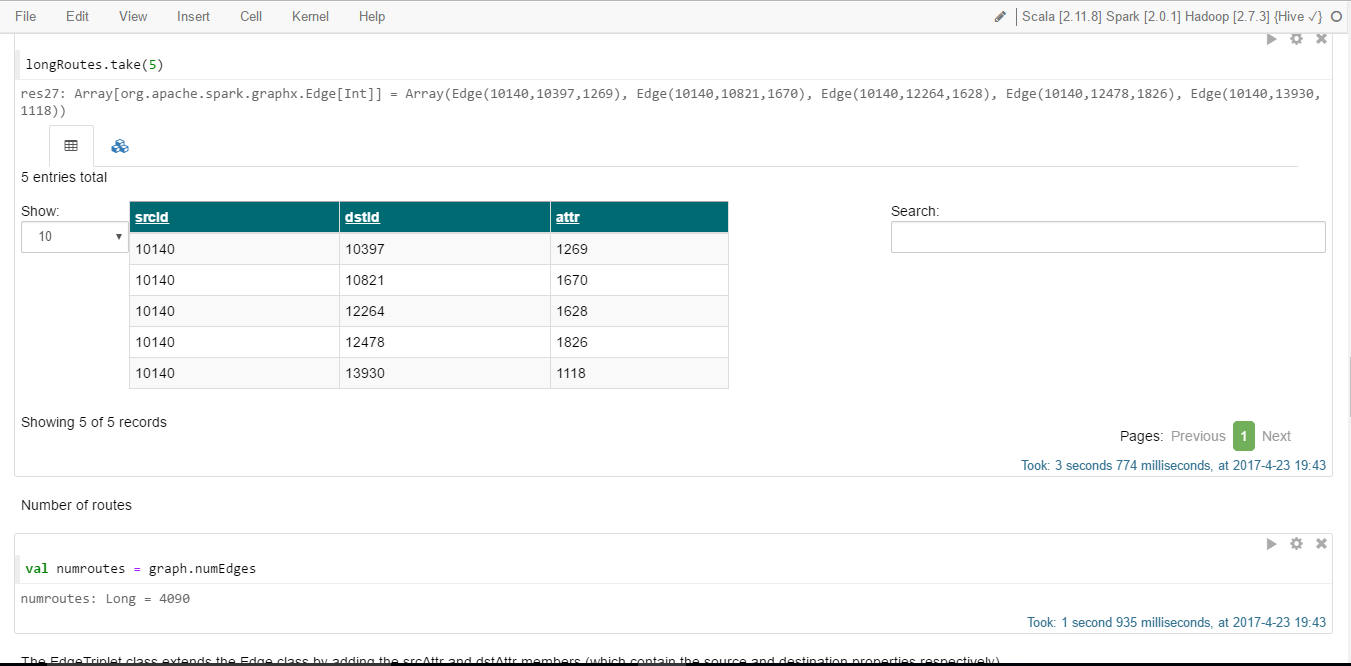




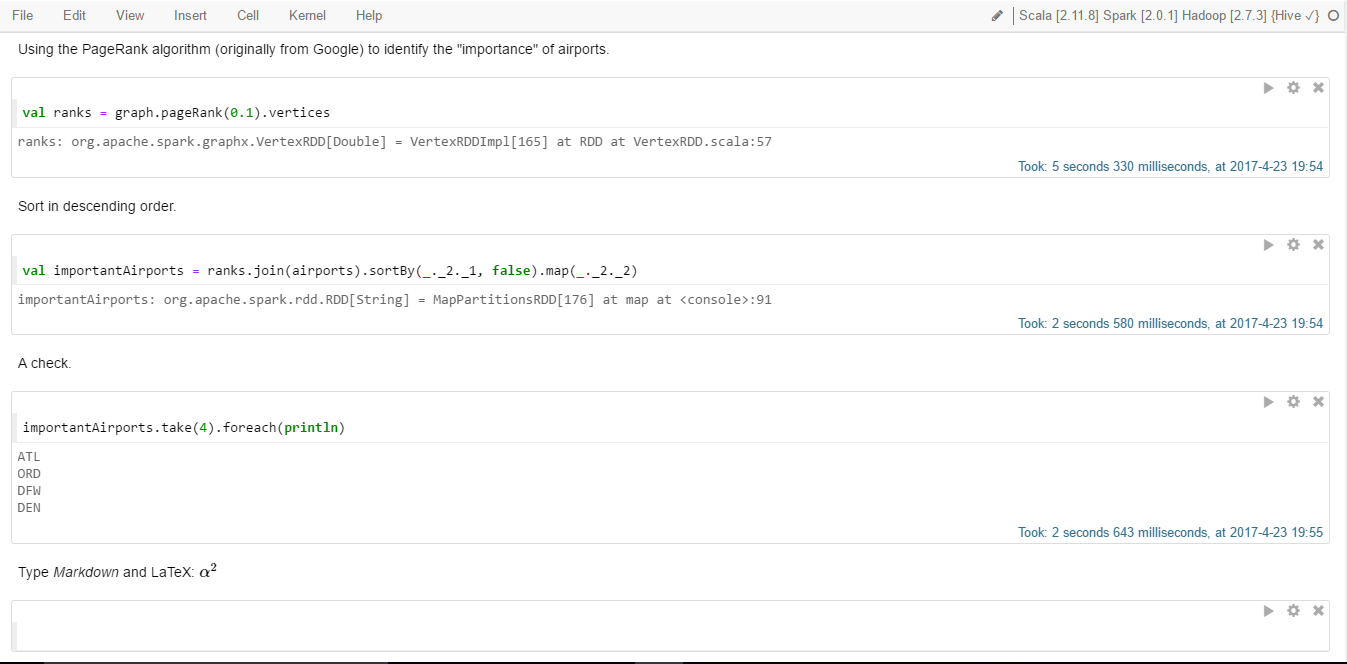












References:

<http://spark.apache.org/docs/latest/graphx-programming-guide.html>

<http://www.sdsc.edu/services/hpc/hpc_systems.html>

<http://spark.apache.org/downloads.html>

<https://en.wikipedia.org/wiki/PageRank>