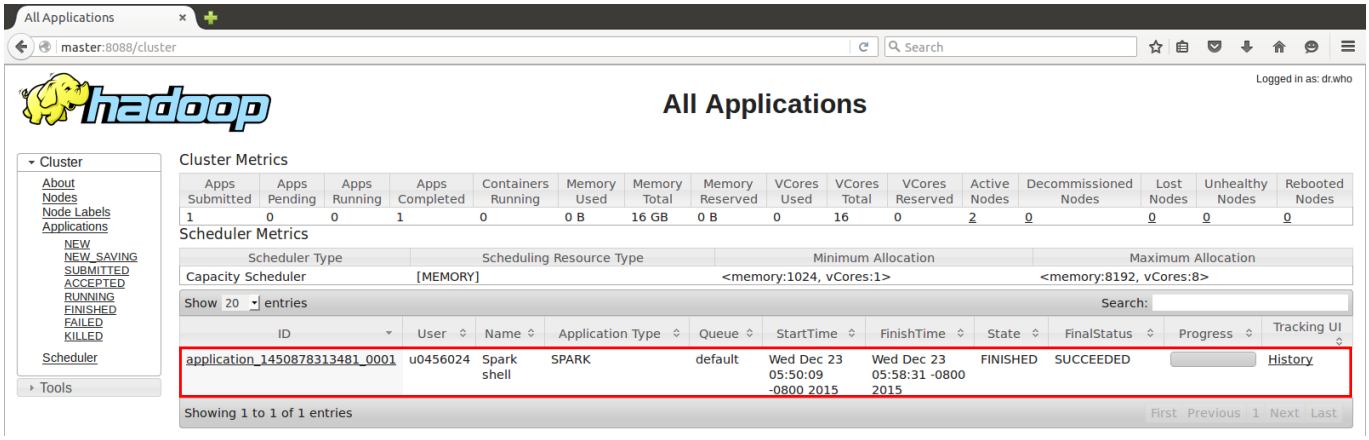
Cloud Computing Homework #4b

1.

• Hadoop administration website

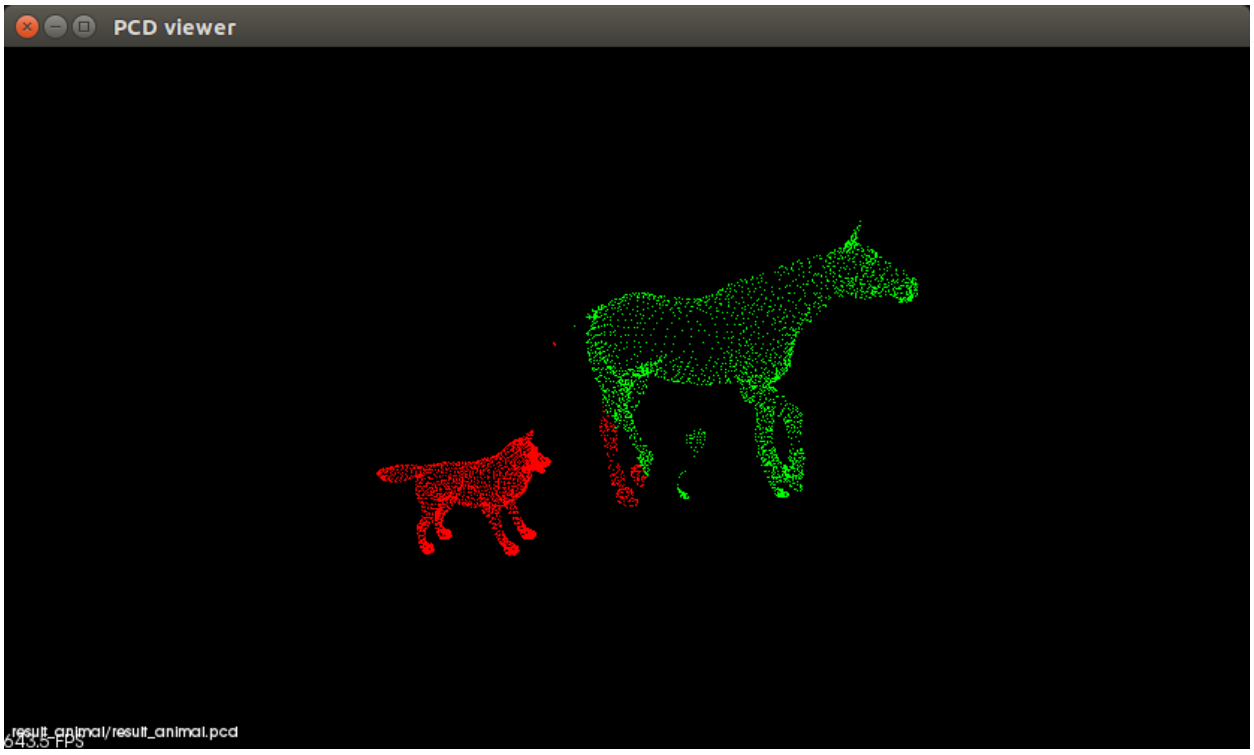


2.

In general case, Spark's in-memory MapReduce provides a better performance than Hadoop's disk-based MapReduce for the k-means application. The benefit of in-memory computing will be significant especially for large-scale data processing.

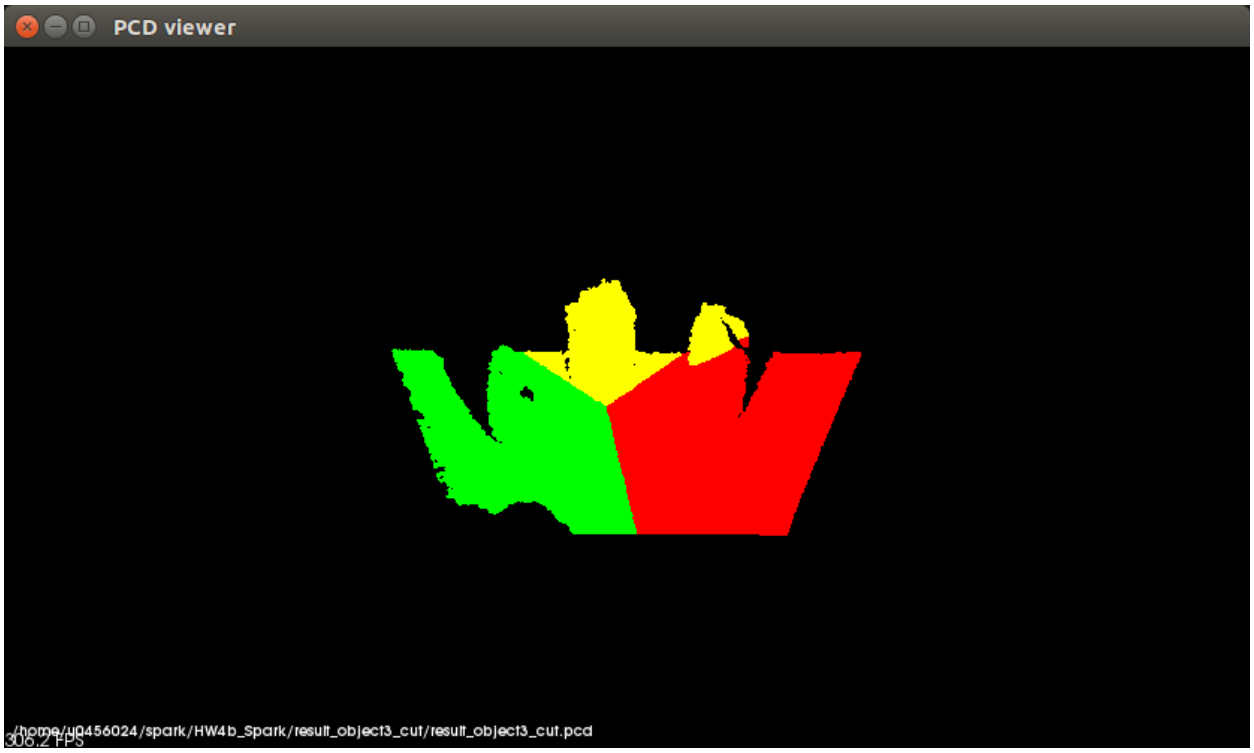
• k-means for animal.pcd

```
u0456024@master:~$ cat ~/spark/HW4b_Spark/result_animal/animal.txt
= Start training ========
= Centers: =========
7.926982958971372 -10.278627425051303 44.59656362277272
99.92372573377794 -198.65948545755026 110.68961797574278
=========== Elapsed Time : 24 s
```



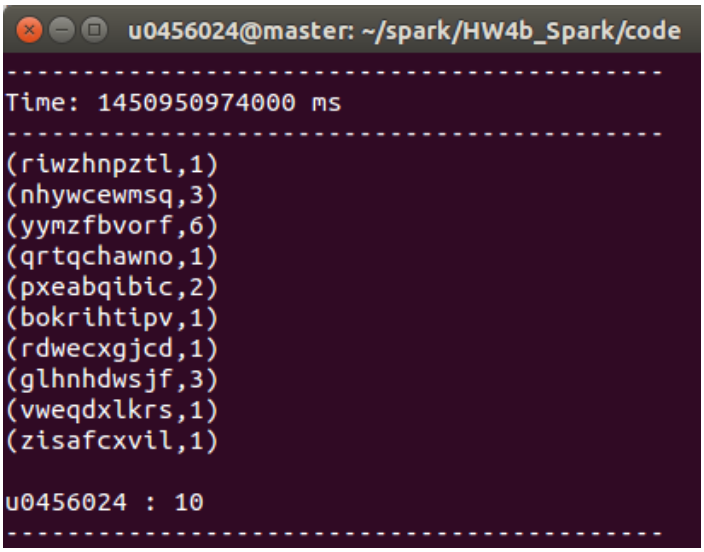
• k-means for object3_cut.pcd

```
u0456024@master:~$ cat ~/spark/HW4b_Spark/result_object3_cut/object3_cut.txt
= Start training ========
= Centers: =========
0.09404676179153634 -0.058282263475382734 -0.7012111096719481
-0.14208863702659844 -0.0705002787035392 -0.6636443015677607
-0.008733922117618462 0.1267819808044984 -0.6762290011003714
========== Elapsed Time : 25 s
```



3.

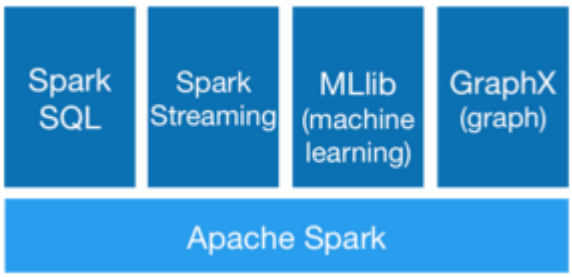
• Spark Streaming



4.

Spark is an open source cluster computing framework. In contrast to Hadoop's two-stage disk-based MapReduce, Spark's multi-stage in-memory primitives provides performance up to one hundred times faster. Hadoop stores the output of a mapper to the disk and waits for a reducer to collect while Spark caches data in memory for iterative computing, so Spark is more suitable for machine learning algorithms like k-means. Spark requires a distributed file system and a resource manager, here we deploy Spark with pre-built Hadoop DFS and Hadoop YARN respectively.

In developers' points of view, there are some advantages of Spark over Hadoop: Spark supports more programming languages like Java, Scala, Python and R, and it also offers high-level extensions including Streaming, SQL and DataFrames, GraphX, and MLlib for machine learning.



Since both Hadoop and Spark are open source, it is not hard to find tutorial about them. The above Spark Streaming experiment NetworkWordCount is an example written in Spark documents.

A Quick Example

Before we go into the details of how to write your own Spark Streaming program, let's take a quick look at what a simple Spark Streaming program looks like. Let's say we want to count the number of words in text data received from a data server listening on a TCP socket. All you need to do is as follows.

Scala Java Python

First, we import the names of the Spark Streaming classes and some implicit conversions from StreamingContext into our environment in order to add useful methods to other classes we need (like DStream). StreamingContext is the main entry point for all streaming functionality. We create a local StreamingContext with two execution threads, and a batch interval of 1 second.

```
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3

// Create a local StreamingContext with two working thread and batch interval of 1 second.
// The master requires 2 cores to prevent from a starvation scenario.

val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
val ssc = new StreamingContext(conf, Seconds(1))
```