Topic: Big data case study in life sciences

Using Cassandra to store and retrieve evolutionary sequence conservation scores for Cancer mutation annotation

# Project Summary

**Problem:** Sequence conservation scores for each base pair of the human genome amounts to a large amount of data in the order of several gigabytes (~20) and regular RDBMS are not able to handle this amount of data in the speed that is needed to annotate hundreds or thousands of mutation data being generated by the increasing number of tumour samples being sequenced. The common method for analyzing this kind of data is keeping them in text files and using scripts for retrieving data. But when annotating large amounts of mutations from large amounts of samples, the time taken file read writes in the data retrieval process adds huge delays to data analysis pipelines.

**My goal:** Fast and accurate retrieval of sequence conservation scores to speed up pipelines for annotating mutations obtained from whole genome sequencing.

**Overview of steps:**

1. Installing and configuring Cassandra on Ubuntu
2. Using Cassandra Bulk Loader for inserting large datasets (> ~10,000,000 lines)
3. Implementing annotation solution in Java

**Big Data Set:** Evolutionary sequence conservation scores (PhastCons) for each base pair (only chromosome 1 was used for demo)

<http://hgdownload.cse.ucsc.edu/goldenPath/hg19/phastCons46way/placentalMammals/chr1.phastCons46way.placental.wigFix.gz>

Size : 1.3GB in wigFix format (4.5GB in csv format)

Format : .wigFix (This needs to be converted to a .csv format)

**Hardware:** Ubuntu 64 bit

**Software:** Cassandra 2.5.1, Eclipse and Datastax DevCenter

**Benefits (Pros/Cons):**

Pros: Compared to RDBMS, NoSQL DBs are able to handle storing, managing and retrieval of a large data source by spreading it over a cluster. Since data is in column format and mainly consists of read operations Cassandra is better suited that other NoSQL databases.

**Summary:** Spreading out per base pair conservation scores across a cluster in a per-chromosome (one node contains conservation scores of one chromosome, ideal # of nodes n > 23) manner allows for parallel retrieval of conservation scores for fast annotations thousands of single nucleotide variations and indels in cancer samples.

# Problem Definition

### Overview

Annotating mutations with sequence conservation scores (PhastCons) is an important part of the analysis process. However, PhastCons for each base pair of the human genome amounts to a large amount of data in the order of several gigabytes (~20GBs) which makes it extremely difficult (virtually impossible) to effectively store and retrieve using traditional RDBMS.

### Currently used methods and their limitations

The common method for analyzing this kind of data is converting them to binary files and retrieving them using scripts written for that purpose. But when annotating large amounts of mutations from large amounts of samples, the time taken for file read writes in the data retrieval process adds huge delays to data analysis pipelines.

### Proposed solution

Use Apache Cassandra for storing and retrieving PhastCons score to speed up mutation annotation pipelines.

# Overview of Technology

## NoSQL

References:

<http://nosql-database.org>

<http://planetcassandra.org/what-is-nosql/>

<http://www.mongodb.com/nosql-explained>

<http://nosqlguide.com>

### What is a NoSQL Database?

NoSQL databases refer to databases that stray from the traditional and sometimes rigid practices of SQL databases and use their own APIs to store and retrieve data. However since some of these types of databases do support the use of SQL or have a similar query language, the definition of NoSQL had been updated to “Not-only SQL”.

NoSQL databases offer a far more flexible approach to that offered by relational databases and its been optimized for todays data requirements. Their APIs allow for storing huge amounts of data and exchanging it with your application at high-speed.

NoSQL databases are categorised base on the kind of data that they are best suited to storing and the data structures they use for storing them.

### Types of NoSQL Databases

There are four general types of NoSQL databases, each with their own specific attributes:

1. Key-Value store –

* Simplest form of NoSQL DB
* Data is stored in a schema-less manner using key value pairs.
* E.g.: DyanmoDB, Azure Table Storage (ATS), Riak, BerkeleyDB.

1. Column store (also known as wide-column stores) –

* Shares similarities with to traditional database in its use of rows, columns and tables.
* Main difference is columns are created for each row rather than being predefined by the table structure.
* E.g.: HBase, Cassandra etc.

1. Document database –

* Designed for document-oriented data
* Similar approach to key-value store but each key is paired with a complex data structure known as a document.
* Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
* E.g.: MongoDB, CouchDB

1. Graph database –

* Designed to store information about networks, such as social connections, where relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them.
* Examples include: Neo4J, HyperGraphDB and Polyglot.

### Comparison of NoSQL Databases and Relational Databases

Both NoSQL and relational databases have their own advantages. Some of them are highlighted below

|  |  |
| --- | --- |
| Relational | NoSQL |
| * Suitable well structured datasets that are only subjected to small processing changes that can be logged and reversed if needed. * ACID properties are maintained and therefore good for reliable and secure processing of transactions by many simultaneous connections * Database can handle most operations like filtering and sorting data. * Well established and so many options of implementations are available. E.g. enterprise-level, commercial and open-source solutions. * Technology is stable and well supported. Also it is widely understood and supported. | * Suitable for datasets that have data which highly varying structures or no structure at all. * Data is managed by the users code and it can be accessed from the language that the software is programmed in. * Optimized for extremely fast storage and retrieval of whole documents or objects. * Wide variety of open-source solutions available. |

## Apache Cassandra

### What is Apache Cassandra?

Reference :

<http://planetcassandra.org/what-is-apache-cassandra/>

<http://www.infoworld.com/article/2610676/database/review--cassandra-lowers-the-barriers-to-big-data.html>

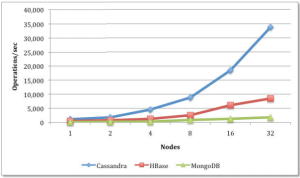
Apache Cassandra is a distributed storage system that is extremely scalable, highly available and fault tolerant. This NoSQL database was initiated by facebook, later open sourced as an apache project. It combines the strengths of both Amazon’s Dynamo and Google’s BigTable into a distributed storage system suitable for managing large amounts of structured data.

It is designed to manage extremely large data sets across large clusters of commodity servers. Cassandra is a column store type of NoSQL database and is best suited for structured data and its "master-less" architecture makes creating and expanding clusters relatively straightforward. However, even though its called a NoSQL database it has a SQL like query language called CQL which allows users to make a smoother transition from traditional database sytemts to using Cassandra.

### Why did I choose Cassandra?

Benefits of choosing Cassandra for given application–

* Data is in a column-based format as opposed to document-based or graph-based. So the two main options are Cassandra and HBase
* Application involves mainly read operations and Cassandra has been shown to out perform all other NoSQL DBs in this regard.



* The Partition key, cluster key organization structure of the data model of Cassandra is well suited for the dataset

### Apache Cassandra Bulk Loader

Reference:

<http://www.datastax.com/dev/blog/using-the-cassandra-bulk-loader-updated>

When you need to upload a large amount of data (> ~ 10 million rows) into a Cassandra DB, it is recommended that you use the bulk loader option.

It uses sstableloader which is a tool that, given a set of sstable data files, streams them to a live cluster. It does not simply copy the set of sstables to every node, but only transfers the relevant part of the data to each, conforming to the replication strategy of the cluster.

There are two primary use cases for this tool:

* Loading pre-existing sstables, typically snapshots, into another cluster with different node counts or replication strategy.
* Bulk loading external data into a cluster: for this you will have to first generate sstables for the data to load, as we will see later in this post.

In the case of this demo, we will be using the 2nd use case.

# Big Data Set: PhastCons 46 way conservation scores

Evolutionary sequence conservation scores (PhastCons) for each base pair (only chromosome 1 was used for demo)

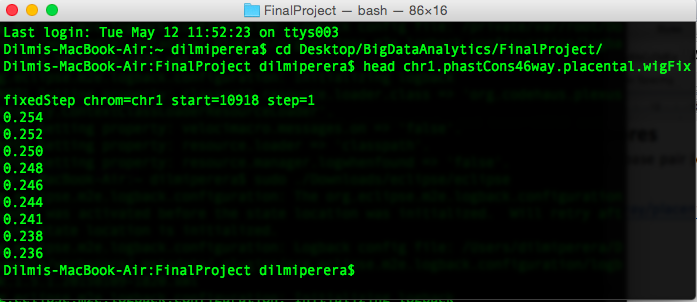
<http://hgdownload.cse.ucsc.edu/goldenPath/hg19/phastCons46way/placentalMammals/chr1.phastCons46way.placental.wigFix.gz>

Size : 1.3GB in wigFix

Format : .wigFix (This needs to be converted to a column format)

### Data preprocessing/cleaning

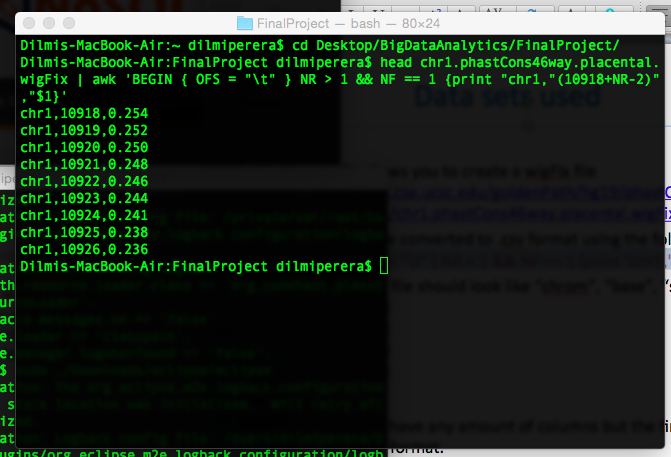
The original file is in .wigfix format which looks like this



The first line of the data provides the chromosome (i.e. chr1) and the starting base (which is 10918). This data can be converted to .csv format using th following script

awk 'BEGIN { OFS = "\t" } NR > 1 && NF == 1 {print "chr1,"(10918+NR-2)","$1}

A sample output is displayed below.



# Implementations details

Overview**:**

1. Installing and configuring Cassandra on Ubuntu
2. Data Model
3. Using Cassandra Bulk Loader for Inserting data
4. Implementing solution in Java

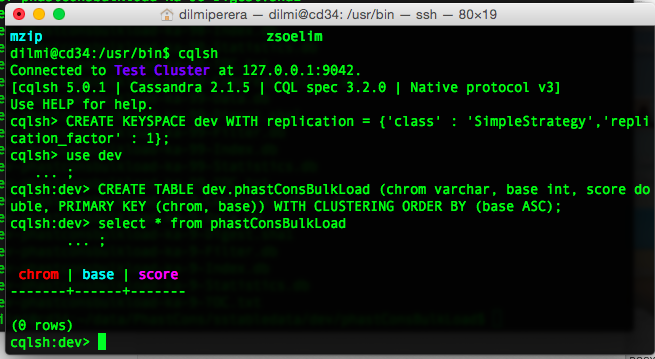
## Installing and configuring Cassandra on Ubuntu

1. Check java version (must be 1.7)
2. Add the DataStax Community repository to the /etc/apt/sources.list.d/cassandra.sources.list using “echo "deb http://debian.datastax.com/community stable main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list”
3. Add the DataStax repository key to your aptitude trusted keys.  
   “curl -L http://debian.datastax.com/debian/repo\_key | sudo apt-key add -”
4. Install the latest package:  
   sudo apt-get update  
   sudo apt-get install dsc21  
   sudo apt-get install cassandra-tools
5. Check by typing “cqlsh” to see if the cassandra prompt appears

## Data Model

* Data Model – The Cassandra data model is a partitioned row store with tunable consistency. This model is ideal for storing PhastCons score data.
* The “chrom” or chromosome variable will be the partition key which would allow the meaningful distribution of data in a per-chromosome manner across a distributed system which would allow for efficient parallel retrieval of data.
* The clustering key would be the “base” variable or the location of the base pair in the chromosome and the data will be ordered based on this.

Creating the table on Cassandra



## Using Cassandra Bulk Loader for Inserting data

\*(Courtesy of http://www.datastax.com/dev/blog/bulk-loading)

### Generate sstable from .csv files

* Specify the schema for bulk loading table.

String SCHEMA = String.format("CREATE TABLE dev.phastConsBulkLoad(” chrom varchar, base int, score double, PRIMARY KEY (chrom, base) WITH CLUSTERING ORDER BY(base ASC) ”);

* INSERT statement to bulk load. It’s like a prepared statement. You fill in place holder for each data.

INSERT\_STMT = String.format("INSERT INTO dev.phastConsBulkLoad (”chrom , base , score) VALUES (?, ?, ?)”);

* Prepare SSTable writer

CQLSSTableWriter.Builder builder = CQLSSTableWriter.*builder();*

* set output directory

builder.inDirectory(outputDir)

* set target schema

.forTable(SCHEMA)

* set CQL statement to put data

.using(INSERT\_STMT)***;***

* Generate CQLSSTableWriter

CQLSSTableWriter writer = builder.build();

* Write to SSTable while reading data

List<String> line;

while ((line = csvReader.read()) != null){

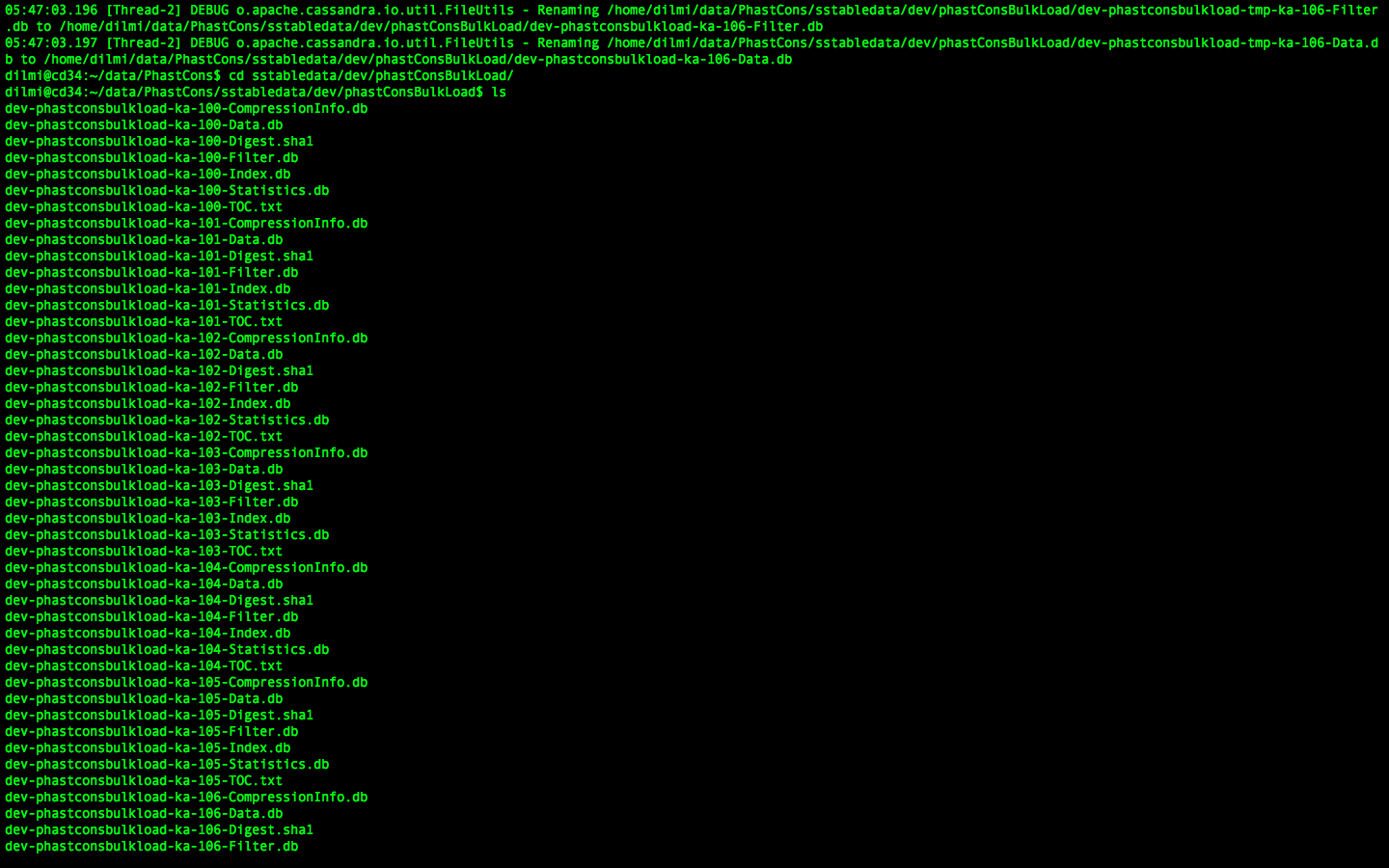
writer.addRow(line.get(0), //Chromosome

new Integer(line.get(1)), //Base

new Double(line.get(2))); //Score

}

List of files being created in the output directory during the creation of sstables



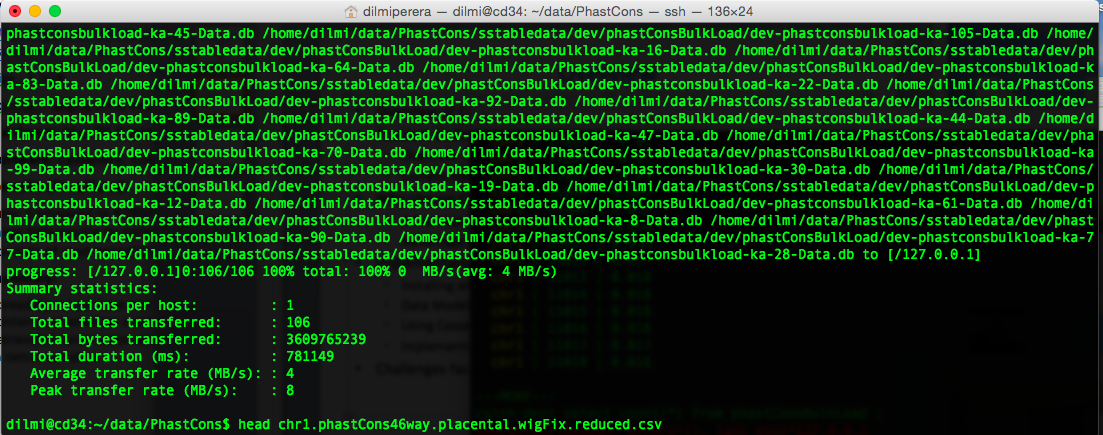
### Load generated sstables into database

Load the generated sstables to the database using the cassandra sstableloader tool using the command

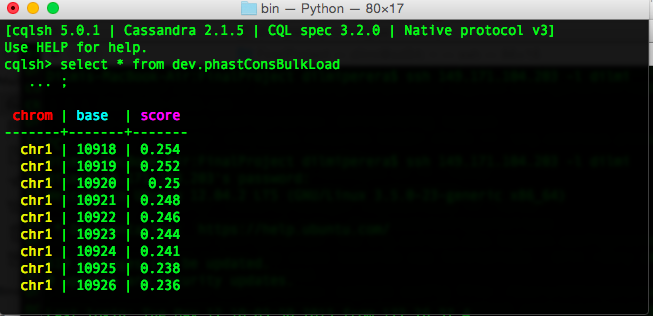
sstableloader –d <database ip address> <folder containing sstable files>



This process will take some time depending on how much data you are attempting to insert into the DB. Wh en the job is complete, you should see the following output.



Verify that the data was uploaded correctly



## Java implementation for annotating mutations using PhastCons

* Read mutation file in the following format (in 0 base coordinates, can have any number of columns after the first 3)

chr1,10919,10920,..,..,..,..  
 chr1,10921,10922,..,..,..,..  
 chr1,10920,10925,..,..,..,..

* E.g For a mutation located in the following location
* chr1,10920,10925

public double getScore(String chrom, int start, int end ) {

String query="SELECT \* FROM dev.phastConsBulkLoad WHERE chrom='"+chrom+"' and base>”+start+"' and base<”+end+1+";";

ResultSet results = session.execute(query);

* Read the returned result set and provide average score for region

# Lessons Learned/Issues found

* You need to use the bulk loader option when loading data files larger than ~10,000,000 rows
* Memory requirements while generating sstables.
  + Depending on the size of the file, sstable generation component of the bulk loader has a significant memory requirement
* Maven repository
  + You need the marven repo for running the CQLSSTableWriter component.
  + However, the guava library that comes with the marven repository is an older version(14.0). You need the newer version (16.0) and it need to be added seperately.