

Proposed Hadoop Ecosystem for Biometrics Statistical Programming Team

Contents

1. [Introduction](#)
 - a. [What is Big Data?](#)
 - b. [Why Big Data?](#)
 - c. [Big Data Challenges](#)
 - d. [A Brief Overview of Hadoop](#)
2. [Existing System](#)
 - a. [Directories](#)
 - b. [Data Structures](#)
 - c. [Hadoop Archives](#)
 - d. [Solr Search](#)
3. [Assumptions](#)
4. [Proposed Improvements](#)
 - a. [SAS to CSV Conversion](#)
 - b. [Data Profiling/Search](#)
 - c. [Additional Spark Layer](#)
5. [Appendix/Documentation](#)

Introduction

This is a proposal for the design of an experimental Hadoop ecosystem. First, before I elaborate on the design, I would like to clearly define “Big Data” and “Hadoop Ecosystem” and why they are important from a Statistical Programming and Analysis perspective.

What is Big Data?

There is no standardized definition for “Big Data”. A very lucid definition of Big Data will be “Data that cannot be stored or processed on a single machine”. From the context of the SPA department and Biometrics as a whole, clinical trial data come in different formats and sizes. There are around 200K studies, with each study containing 50 – 100 files with sizes varying between 50 KB to 1 GB. I don’t know how fast the data grows but I’m assuming it grows fast. It is not feasible to store all of this data on a single physical system because sooner or later we will run out of space and memory to store, process and even transfer the data.

Why Big Data?

Over time the velocity at which data grows has become larger and the cost of storing data has become cheaper. Previously, businesses had to discard data perceived to be not useful to save space and cost. All data is useful, the age of Big Data allows businesses to store all data in whichever format suits their need best. The advantages of Big Data are endless irrespective of the type of industry. In the case of SPA and Biometrics, Big Data Applications combined with Machine Learning Algorithms help predict clinical outcomes based on clinical data even before the clinical trials are completed and this is just one of the many new exciting avenues presented by Big Data.

Big Data Challenges

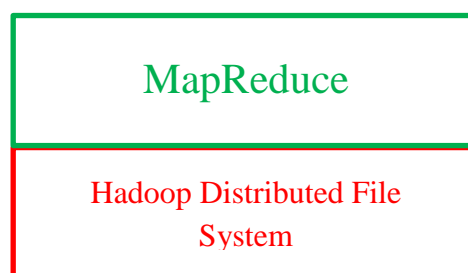
As the volume of data grows, more and more businesses are looking to tap into the data to optimize business processes and open up new avenues for innovation. However, Big Data comes with its own set of challenges that need to be addressed in order to efficiently leverage its power. I enumerate the most important challenges from our perspective below:

1. **Security** – In healthcare, especially in Pharma, compliance with industry regulations are non-negotiable. Keeping patient data safe is of paramount importance. Managing security in Big Data applications is cumbersome because Big Data applications often run on Open Source software which tend to have inconsistent and volatile security policies. However, many well-supported, commercial Big Data solutions are available such as Cloudera’s Enterprise Distribution which has very strong security policies that can be tuned to meet industry compliance requirements.
2. **Different Data Formats** – Big Data comprises of data that originate at different sources and come in different formats. Making these data work with each other is a challenge. From a biometrics perspective, studies have two basic types of data – Raw and Derived. These data are labeled as RAW, DERIVED in legacy studies and as SDTM, VAD based on current data models. Also, changes in industry data collection regulations over time would have resulted in differences in the data. Irrespective of these differences, the variety of the data is a major strength of Big Data as well. With a few changes in the code, we can perform the same analyses over different formats of the data.

3. **Big Data is still evolving** – Big Data technologies continually change and evolve. Newer systems are built to address challenges of older systems, like Apache Spark replacing MapReduce. To maintain the competitive edge that Big Data provides, we must continually be on our toes to keep pace with the evolving technologies in order to improve the efficiency of our Big Data platform.

A Brief Overview of Hadoop

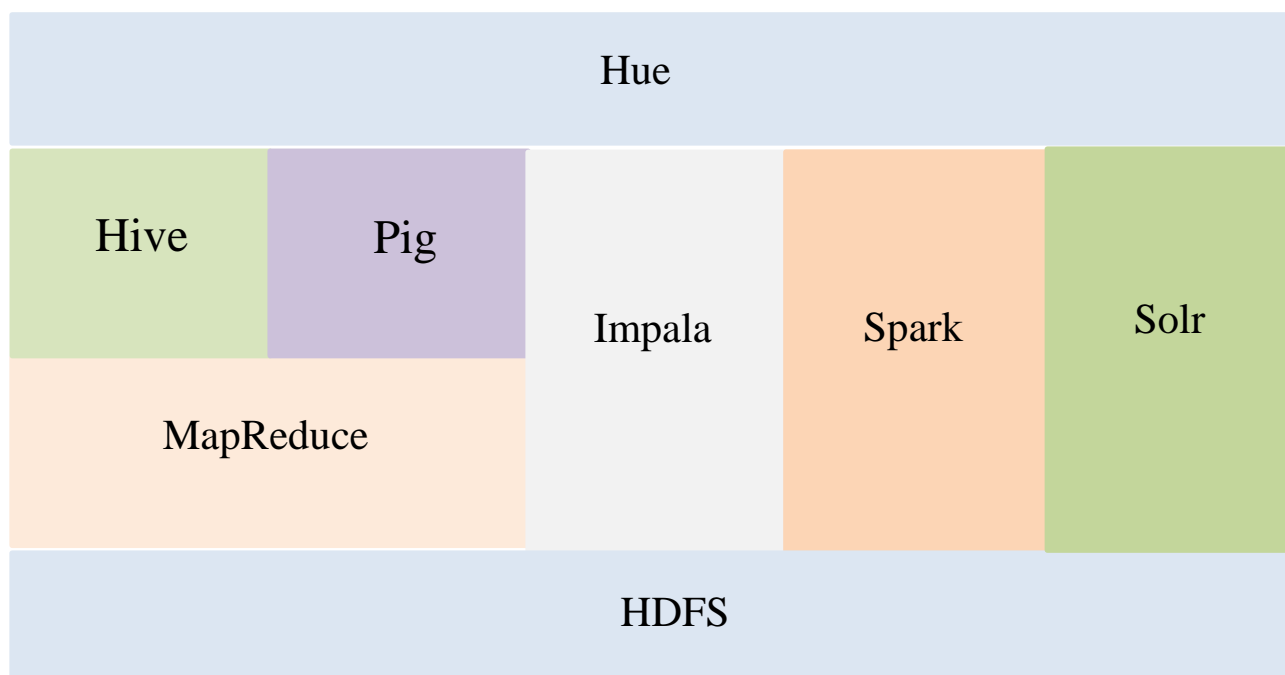
Simply put, Hadoop is a parallel-processing, computing framework for data storage and processing. It is a flexible Big Data Analytics framework. The core of Hadoop consists of a way to store the data known as the Hadoop Distributed File System (HDFS) and a way to process that data known as MapReduce. HDFS stores replicated chunks of data 64MB in size. MapReduce sits on top of HDFS and processes the data in HDFS in place.



But since Hadoop is an Open Source project and MapReduce has a significant learning curve, many Open Source projects have been created to make it easier to load, visualize, query and analyze data. There are many good products that improve the user experience of Hadoop, but not all are useful. It is very easy to get confused by all the terms being thrown around. I enumerate the useful ones below.

1. **Hive** – Hive is an SQL like interface built on top of MapReduce enabling users to directly write SQL code without having to worry about the MapReduce phases. The Hive SQL queries are converted to MapReduce and the results are returned.
2. **Pig** – Pig is a scripting language used for data analysis. Just like Hive it is built on top of MapReduce.
3. **Impala** – The disadvantage of Hive and Pig is that they still have to access HDFS via MapReduce, which is built for batch processing and therefore slow for firing queries and performing analysis. Impala on the other hand can directly access HDFS and is optimized for low-latency queries. Impala queries run many times faster than Hive queries.
4. **Spark** – Spark is a highly scalable framework that provides Application Programming Interfaces in Python, Scala and R to more efficiently access the Hadoop Distributed File System. It was developed to address the challenge of learning MapReduce. Spark also provides interfaces for Hive and Pig as well as Machine Learning Components to truly leverage the power of Big Data.
5. **Hue** – Hue is a graphical User Interface built on top of the cluster to provide easy and simple access to the Hadoop cluster below.
6. **Solr** – Solr is Hadoop's search engine which can search through semi-structured/unstructured documents stored in HDFS.

A cohesive collection of these Hadoop components is called the **Hadoop Ecosystem**. A typical Hadoop distribution like Cloudera or Hortonworks has an architecture as shown below:



Existing System

The current implementation is a Data Lake a storage methodology that allows end users to store data in its raw, schema-less form. It is an advancement over Data Warehousing methodologies that allow users to only store data in highly normalized schemas.

Directories

The current implementation is a Data Lake a storage methodology that allows end users to store data in its raw, schema-less form. It is an advancement over Data Warehousing methodologies that allow users to only store data in highly normalized schemas. For the POC 4 directories have been provided – SDTM, VAD, derived and raw. In addition there are directories to store documents, archives and data profiles. The directory system is extremely useful, however, for the directory structure to be optimal it must be standardized across all studies (i.e. all directory subfolders should have the same names across studies) in order to facilitate the seamless execution of queries and scripts. In the system that I had access to, different studies had different directory substructures. This is not really a major problem but as previously mentioned having a standard directory structure across studies makes the system optimal and reduces the time spent on modifying the code.

Data Structures

The current implementation consists of Hive and Impala tables. Hive and Impala are optimal for running SQL queries and basic analytics but not the “heavy stuff” like data mining, machine learning or natural language processing. Impala is optimal for running low-latency queries and Hive is optimal for Batch-Processing queries. The biggest strength of Hadoop is its flexibility as well as

module encapsulation (separation). Even though the current system is built over Hive and Impala, Hadoop allows us to keep the existing infrastructure and add more to it.

Hive is useful for

1. Long running ETL processes.
2. Data Preparation.
3. Batch Processing.

Impala is useful for

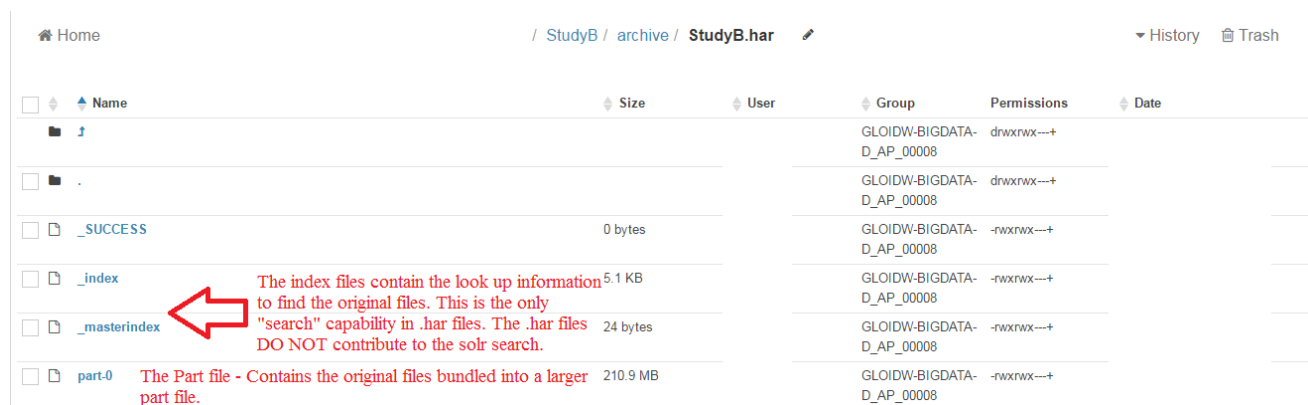
1. Business Intelligence purposes.
2. Support multi-user workloads.
3. Low latency interactive performance.

Both Hive and Impala are designed to work with existing data. They cannot handle real-time data. To handle real-time data we must use Apache Spark streaming.

Hadoop Archive (.har files)

The Hadoop Distributed File System is inefficient when it comes to handling a large number of small files. From a Hadoop context “Small Files” mean files smaller than 64 MB, the standard Hadoop block size. In order to get around this problem, the current implementation packs all the files into a Hadoop Archive or a .har file. This is a valid approach and will have to be used to bundle small files together. Files larger than 64 MB need not be archived into the .har file.

When you access a .har file in Hue, you see what it is made up of – Part files and Index Files.



	Name	Size	User	Group	Permissions	Date
	.			GLOIDW-BIGDATA-D_AP_00008	drwxrwx---+	
	._SUCCESS	0 bytes		GLOIDW-BIGDATA-D_AP_00008	-rwxrwx---+	
	._index	5.1 KB		GLOIDW-BIGDATA-D_AP_00008	-rwxrwx---+	
	._masterindex	24 bytes		GLOIDW-BIGDATA-D_AP_00008	-rwxrwx---+	
	part-0	210.9 MB		GLOIDW-BIGDATA-D_AP_00008	-rwxrwx---+	

The only limitation of .har files is that they are immutable. Once a har file has been generated, no other files can be added or removed. Since creating har files is fast and simple, this limitation can be overcome by creating a new updated har file. however that needs to be done manually.

Solr Search

The current implementation also involves a Solr Search Dashboard. From my meetings with Jennifer I have surmised that the search utility is the most important functionality required at the moment. Solr is a powerful open source search engine that can search Structured (SQL Tables), Semi-

Structured (CSV/SAS Files) and unstructured data (PDFs/DOCs). The current implementation seems to include the documents and the profiles of the SAS tables. However, on running multiple queries such as “sinusitis”, “gastroentitis” or “gas”, I noticed that :

1. Only documents containing those terms were being returned as results.
2. Entering half a keyword such as “gas” returns no results.

I have diagnosed the problem as follows – Solr Search has no problem finding search terms in unstructured data like documents since we do not have to specify the documents’ schema to Solr. However, the search may not be working across studies because :

1. The data profiles were created using the SAS files. (I do not have access to the Solr Schema that has been used (I have requested for access).). It is always advisable to convert the SAS files to CSV files for searching with Solr.
2. **To search across studies and CSV files, one must create and load schemas for each CSV file. This schema is an XML file and gives Solr a way to infer fields, their types and most importantly indices.** This process needs to be done manually and that is a major limitation, however, the schema creation process needs to be done only once. There may be a way to automate the schema creation process; I am currently looking into it.

Consider the following snippet of “sim1.csv”

STDT	RAND	ENDT	AGE	ETHNIC	HIV	BP
9/23/2009	YES	9/23/2009	64	Not Hispanic or Latino	POSITIVE	HIGH
12/9/2009	YES	12/9/2009	57	Not Hispanic or Latino	POSITIVE	HIGH
12/8/2009	YES	12/8/2009	61	Not Hispanic or Latino	POSITIVE	HIGH
1/23/2010	YES	1/23/2010	62	Not Hispanic or Latino	NEGATIVE	HIGH
1/23/2010	YES	1/23/2010	62	Not Hispanic or Latino	NEGATIVE	HIGH
7/21/2009	YES	7/21/2009	40	Not Hispanic or Latino	POSITIVE	LOW

The corresponding XML schema for this will look like this :

```
1 <field name="STDT" type="date" indexed="true" stored="true" required="true"/>
2 <field name="RAND" type="string" indexed="false" stored="true"/>
3 <field name="ENDT" type="date" indexed="true" stored="true" required="true"/>
4 <field name="AGE" type="int" indexed="false" stored="true" omitNorms="true"/>
5 <field name="ETHNIC" type="string" indexed="true" stored="true"/>
6 <field name="HIV" type="string" indexed="true" stored="true" required="true"/>
7 <field name="BP" type="string" indexed="true" stored="true"/>
```

Once all the schemas are created and loaded, we can use the full power of Solr Search to search across studies and tables. We can use regular expressions and other search functionalities to narrow down search results.

Assumptions

I am making a few assumptions regarding the current implementation as I do not have confirmed knowledge about the underlying implementations especially the Hadoop Distribution used. I will base my suggested improvements on these assumptions as well as my observations as noted above :

1. The Hadoop distribution used is Cloudera Enterprise.
2. SDTM folders store the raw format of the data and VADs store the derived formats. I assume this is the new naming convention; previous studies have RAW and DERIVED folders.
3. Each study contains between 50 – 100 files with sizes varying between 50KB to 1GB.
4. The search functionality is a priority over the SQL queries.
5. There is a possibility that the system may be used for data mining and machine learning.

Proposed Improvements

SAS to CSV Conversion

The current implementation suggests using a Spark converter. However, that process might be cumbersome as it uses SparkSQL and invokes a lot of modules. I have designed and implemented an R executable file that converts all the sas7bdat files in a given directory and converts them into CSV files with the same file names. For example, ae.sas7bdat gets converted to ae.csv. All of this happens on the local machine and I am looking to move it into the Hadoop system by using R on Spark. This can also be done using Python User Defined Functions for Pig. The main aim here is that once a SAS file is uploaded onto the system it can be converted into a CSV file by simply running a script. I would suggest converting SAS files into CSV files early in the pipeline. These CSVs can later be used for SQL queries on Impala/Hive and for the Solr Search and advanced analytics using Spark.

Search/Data Profiling

While searching documents/unstructured data, Solr performs full text search. In the case of CSV files or semi-structured data, Solr requires a schema to infer the indices and types in the data. I suggest building Solr schemas from CSV files to yield a more powerful and effective search engine. XML schemas help ensure that the search results are more comprehensive. The profiles built from SAS tables did not seem to be satisfying the search. While creating XML Schemas may be a manual and cumbersome task, it needs to be done only once for the system. Plus if there is existing standards documentation regarding the data files (for example, a standards document that says ae.sas7bdat should have the following attributes), then building schemas will be simpler. It is definitely feasible but will be time consuming.

Additional Spark Layer

The current implementation is adequate for running basic analytics using Hive/Impala. However, as the number of datasets as well as their sizes grow, it will become more difficult to analyze those datasets using SQL in Hadoop. I suggest the addition of a Spark layer that allows us to perform more advanced analytics on the data. Given a new Spark layer we can run computationally intensive tests on the data like supervised and unsupervised learning models. Spark is many times faster than Hive/Impala and has a strong Application Programming Interface (APIs). As a result no new

languages need to be learnt. Spark's python API, PySpark and its R API, SparkR should be enough for programming most computationally intensive tasks. Also because Spark is extremely fast, one can perform ETL tasks on the Hadoop ecosystem rather than having to do them on local machines. The tasks can be performed on a set of files in Batch Processing mode or on just the newly added file.

Appendix

Code to create Hadoop Archive

```
hadoop archive -archiveName myhar.har /input/location /output/location
```