NIST Special Publication 1500-xx

NIST Big Data White Paper Series: Big data Analytics for Healthcare Data/Health informatics

D R A F T

NIST Big Data Public Working Group

Draft Version 1
<Month, Day>, 2016
http://dx.doi.org/10.6028/NIST.SP.1500-xx

NIST Big Data Public Working Group

BIG DATA ANALYTICS FOR HEALTHCARE DATA/HEALTH INFORMATICS

How-To Guide
for Data Scientists / Researchers / Engineers

by
<list of authors>

NIST Big Data Public Working Group

NIST SPECIAL PUBLICATION 1500-XX

DISCLAIMER

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1500-xx, Natl. Inst. Stand. Technol. Spec. Publ. 1500-xx, xx pages, (date), CODEN: NSPUE2

Organizations are encouraged to review all draft publications during public comment periods and provide feedback. All white paper publications from NIST Big Data Public Working Group are available at http://bigdatawg.nist.gov/whitepapers.

Comments on this publication may be submitted to: <need email>@nist.gov

Public comment period: <month day,> 2016 through <month, day,> 2016

NIST Big Data Public Working Group
National Institute of Standards and Technology
Email: wo.chang@nist.gov

Executive Summary

Large amount healthcare data is produced continually and store in different databases. With the wide adoption of electronic health records that has increased the amount of data available exponentially. Nevertheless, the healthcare providers have been slow to leverage the vast amount of data to improve health care system or use data to improve efficiency to reduce overall cost of healthcare.

Health care data has the potential to innovate the procedure of health care delivery in the US and inform healthcare providers about the most efficient and effective treatments. Value-based healthcare programs will provide incentives to both healthcare providers and insurers to explore new ways to leverage healthcare data to measure the quality and efficiency of care.

THE CHALLENGE

It is estimated that in the US healthcare spending approximately, $75B to $265B is lost each year to healthcare fraud[1]. With the amount of healthcare fraud, the importance of identifying fraud and abuse in healthcare cannot be ignored; so healthcare providers must develop automated systems to identify fraud, waste and abuse to reduce its harmful impact on their business.

THE SOLUTION

The NIST Big Data Public Working Group practice guide *<title>* demonstrates how data scientists can instantiate NIST Big Data Reference Architecture (NBD-RA) to solve the healthcare fraud problem when given the dataset and Big Data analytics algorithm.

We demonstrate how healthcare fraud can be supported throughout the Big Data analytic lifecycle. This includes how to interact NBD-RA components – Data Provider, Big Data Analytic Provider, Big Data Framework Provider, and Big Data Consumer.

The guide:

1. Identifies the healthcare fraud characteristics needed to sufficiently analyze the given dataset with analytics algorithm.

2. Maps healthcare fraud characterizes to Big Data Analytic Provider

3. <others...>

[1] White SE. Predictive modeling 101. How CMS's newest fraud prevention tool works and what it means for providers. J AHIMA. 2011;82(9): 46-47.

BENEFITS

<benefits statement>

Preface

With the development of massive search engines (such as Google and Yahoo!), genomic analysis (in DNA sequencing, RNA sequencing, and biomarker analysis), and social networks (such as Facebook and Twitter), the volumes of data being generated and processed have crossed the petabytes threshold. To satisfy these massive computational requirements, we need efficient, scalable, and parallel algorithms. One framework to tackle these problems is the MapReduce paradigm.

MapReduce is a software framework for processing large (giga-, tera-, or petabytes) data sets in a parallel and distributed fashion, and an execution framework for large-scale data processing on clusters of commodity servers. There are many ways to implement MapReduce, but in this book our primary focus will be Apache Spark and MapReduce/Hadoop. You will learn how to implement MapReduce in Spark and Hadoop through simple and concrete examples.

This book provides essential distributed algorithms (implemented in MapReduce, Hadoop, and Spark) in the following areas, and the chapters are organized accordingly:

- Basic design patterns
- Data mining and machine learning
- Bioinformatics, genomics, and statistics
- Optimization techniques

What is NIST Big Data Interoperability Framework?

It is estimated that in the US healthcare spending approximately, $75B to $265B is lost each year to healthcare fraud[2]. With the amount of healthcare fraud, the importance of identifying fraud and abuse in healthcare cannot be

[2]White SE. Predictive modeling 101. How CMS's newest fraud prevention tool works and what it means for providers. J AHIMA. 2011;82(9): 46-47.

ignored; so healthcare providers must develop automated systems to identify fraud, waste and abuse to reduce its harmful impact on their business.

NIST Big Data Reference Architecture

The NIST Big Data Public Working Group practice guide *<title>* demonstrates how data scientists can instantiate NIST Big Data Reference Architecture (NBD-RA) to solve the healthcare fraud problem when given the dataset and Big Data analytics algorithm.

We demonstrate how healthcare fraud can be supported throughout the Big Data analytic lifecycle. This includes how to interact NBD-RA components – Data Provider, Big Data Analytic Provider, Big Data Framework Provider, and Big Data Consumer.

Big Data Analytics Lifecycle

The Cross Industry Standard Process for Data Mining (CRISP-DM) methodology (https://the-modeling-agency.com/crisp-dm.pdf)

Sample, Explore, Modify, Model, and Assess (SEMMA) (

Comparison between CRISP-DM and SEMMA: http://recipp.ipp.pt/bitstream/10400.22/136/1/KDD-CRIS

The focus of this book is to embrace the MapReduce paradigm and provide concrete problems that can be solved using MapReduce/Hadoop algorithms. For each problem presented, we will detail the map(), combine(), and reduce() functions and provide a complete solution, which has:

What is the Focus of This White Paper?

The focus of this book is to embrace the MapReduce paradigm and provide concrete problems that can be solved using MapReduce/Hadoop algorithms. For each problem presented, we will detail the map(), combine(), and reduce() functions and provide a complete solution, which has:

- A client, which calls the driver with proper input and output parameters.
- A driver, which identifies map() and reduce() functions, and identifies input and output.
- A mapper class, which implements the map() function.
- A combiner class (when possible), which implements the combine() function. We will discuss when it is possible to use a combiner.
- A reducer class, which implements the reduce() function.

One goal of this book is to provide step-by-step instructions for using Spark and Hadoop as a solution for MapReduce algorithms. Another is to show how an output of one MapReduce job can be used as an input to another (this is called chaining or pipelining MapReduce jobs).

Who is This White Paper For?

This book is for software engineers, software architects, data scientists, and application developers who know the basics of Java and want to develop MapReduce algorithms (in data mining, machine learning, bioinformatics, genomics, and statistics) and solutions using Hadoop and Spark. As I've noted, I assume you know the basics of the Java programming language (e.g., writing a class, defining a new class from an existing class, and using basic control structures such as the while loop and if-then-else).

More specifically, this book is targeted to the following readers:

- Data science engineers and professionals who want to do analytics (classification, regression algorithms) on big data. The book shows the basic steps, in the format of a cookbook, to apply classification and regression algorithms using big data. The book details the map() and reduce() functions by demonstrating how they are applied to real data, and shows where to apply basic design patterns to solve MapReduce problems. These MapReduce algorithms can be easily adapted across professions with some minor changes (for example, by changing the input format). All solutions have been implemented in Apache Hadoop/Spark so that these examples can be adapted in real-world situations.
- Software engineers and software architects who want to design machine learning algorithms such as Naive Bayes and Markov chain algorithms. The book shows how to build the model and then apply it to a new data set using MapReduce design patterns.

- Software engineers and software architects who want to use data mining algorithms (such as K-Means clustering and k-Nearest Neighbors) with MapReduce. Detailed examples are given to guide professionals in implementing similar algorithms.
- Data science engineers who want to apply MapReduce algorithms to clinical and biological data (such as DNA sequencing and RNA sequencing). This book clearly explains practical algorithms suitable for bioinformaticians and clinicians. It presents the most relevant regression/analytical algorithms used for different biological data types. The majority of these algorithms have been deployed in real-world production systems.
- Software architects who want to apply the most important optimizations in a MapReduce/distributed environment.

  This book assumes you have a basic understanding of Java and Hadoop's HDFS. If you need to become familiar with Hadoop and Spark, the following books will offer you the background information you will need:

- Hadoop: The Definitive Guide by Tom White (O'Reilly)
- Hadoop in Action by Chuck Lam (Manning Publications)
- Hadoop in Practice by Alex Holmes (Manning Publications)
- Learning Spark by Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia (O'Reilly)

Online Resources

Two websites accompany this book:

https://github.com/mahmoudparsian/data-algorithms-book/

At this GitHub site, you will find links to the source code (organized by chapter), shell scripts (for running MapReduce/Hadoop and Spark programs), sample input files for testing, and some extra content that isn't in the book, including a couple of bonus chapters.

http://mapreduce4hackers.com

At this site, you will find links to extra source files (not mentioned in the book) plus some additional content that is not in the book. Expect more coverage of MapReduce/Hadoop/Spark topics in the future.

What Software Is Used in This White Paper?

When developing solutions and examples for this book, I used the software and programming environments listed in Table P-3.

Table P-3. Software/programming environments used in this book

Software	**Version**
Java programming language (JDK7)	1.7.0_67
Operating system: Linux CentOS	6.3
Operating system: Mac OS X	10.9
Apache Hadoop	2.5.0, 2.6.0
Apache Spark	1.1.0, 1.3.0, 1.4.0
Eclipse IDE	Luna

All programs in this book were tested with Java/JDK7, Hadoop 2.5.0, and Spark (1.1.0, 1.3.0, 1.4.0). Examples are given in mixed operating system environments (Linux and OS X). For all examples and solutions, I engaged basic text editors (such as vi, vim, and TextWrangler) and compiled them using the Java command-line compiler (javac).

In this book, shell scripts (such as bash scripts) are used to run sample MapReduce/Hadoop and Spark programs. Lines that begin with a $ or # character indicate that the commands must be entered at a terminal prompt (such as bash).

Using Code Examples

As mentioned previously, supplemental material (code examples, exercises, etc.) is available for download at https://github.com/mahmoudparsian/data-algorithms-book/ and http://www.mapreduce4hackers.com.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses

several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "Data Algorithms by Mahmoud Parsian (O'Reilly). Copyright 2015 Mahmoud Parsian, 978-1-491-90618-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

1 Introduction

<Background info>

1.1 Goals

<Goals statement>

1.2 Approach

<Approach description>

1.3 Technologies Used

<Technologies description>

1.4 What You Need to implement this White Paper

Programming experience in Java or Scala and some basic knowledge or understanding of any distributed computing platform such as Apache Hadoop would be desired.

2 Mapping Use Case to NBD Reference Architecture

Components

<Description>

2.1 NBD Reference Architecture Components

<General description>

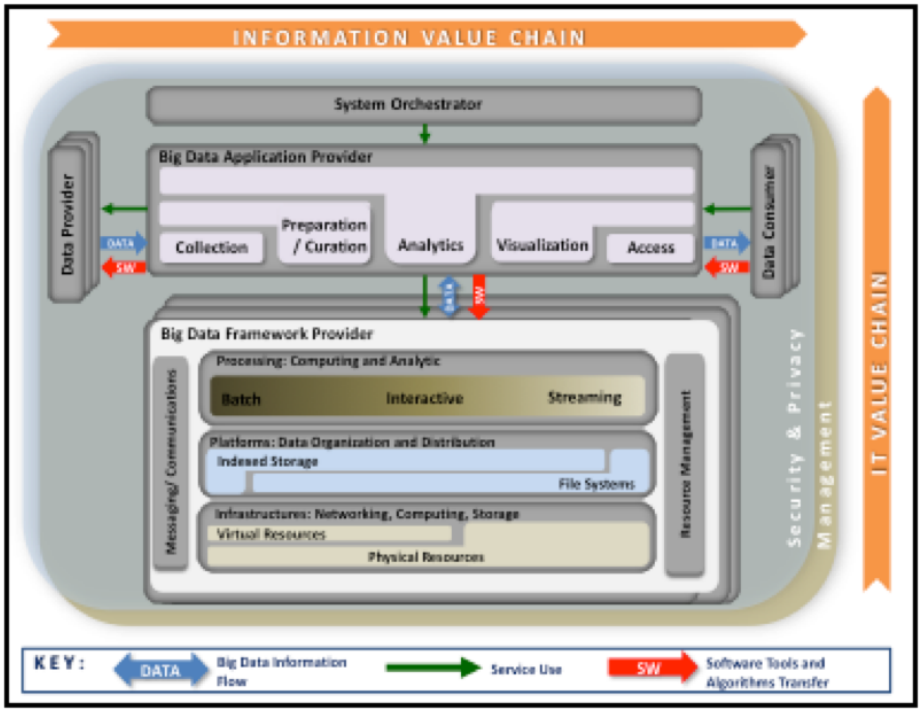

Figure 1: NIST Big Data Reference Architecture Diagram

- ***System Orchestrator (or data scientist)*** - provides high-level design dataflow between analytics tools and given datasets, computing system requirements, monitoring system resource and performance.

- ***Data Provider*** - provides abstraction of various types of data sources (such as raw data or data previously transformed by another system) and makes them available through different functional interfaces. This includes transfer analytics codes to data sources for effective analytic processing.

- ***Big Data Application Provider*** - provides analytics processing throughout the data lifecycle - acquisition, curation, analysis, visualization, and

access - to meet requirements established by the System Orchestrator.

- ***Big Data Framework Provider*** - provides one or more instances of computing environment to support general Big Data tools, distributed file systems, and computing infrastructure - to meet requirements established by the Big Data Application Provider.

- ***Data Consumer*** - provides interface to receive the value output from this NBD-RA ecosystem.

- ***Security and Privacy Fabric*** - provides System Orchestrator the security and privacy interaction to the rest of the NBD-RA components to ensure protection of data and their content.

- ***Management Fabric*** - provides System Orchestrator the management interaction to the rest of the NBD-RA components with versatile system and software provisioning, resource and performance monitoring, while maintaining a high level of data quality and secure accessibility.

The focus of this document is on the Big Data Application Provider for how Big Data analytics can be applied/performed at each of its subcomponents. For the given mass amount of complex data, the process may involve one or more machine learning or analytic processing techniques at each of the subcomponents.

Collection - Data ingestion and storage would be the first order of business to get the analytic environment going. Data transfer may involve various network transport protocols the same way as data storage may deal with simple/distributed file systems to databases.

Preparation/Curation - Data cleansing and transformation may needed before any analytics processing can be performed.

Data cleaning techniques may include:

- Filtering data - throw away unwanted data or keep data that match certain criteria

- Dealing with missing, incomplete or corrupted - seek appropriate methods (using average value, etc.) to keep data align.

- Dealing with outliers - apply outlier techniques to avoid skew the results.

- Aligning data - apply data normalization against certain given patterns
- Aggregating data -

  Data transformation may include:

- Transform numerical values, e.g., temperature conversion (Celsius to Fahrenheit), geolocation (street address to Lat/Long), etc.
- Transform data formats, e.g., image conversion (PNG to JPEG), video conversion (

  AVI to MPEG4), etc.

Analysis - Analytics processing can be performed once the trained data are in place. The process can be applied multiple times with different analytics tools for different processing goals throughout the entire data lifecycle.

Visualization - Data visualization can enable analysts to organize, interact, and express massive amount of data from a human perception and control.

Access

2.2 Use Case Functionalities and Activities

CRISP-DM

<CRISP-DM description and activities>

Use Case Functionalities

<Use case activities description>

Use Case Activities

<Use case activities description>

3 Getting Started: Datasets, Configuration, and Processing

<Description>

3.1 Getting the Datasets

The Center for Medicare and Medicaid Services (CMS) (http://www.cms.gov) has released the "Medicare Part-B in 2014"dataset into the public domain known as part of the Administration's efforts to make the healthcare system more transparent. The dataset includes a set of records documenting about transactions between over 900,000 medical providers and CMS with information on services and procedures provided to Medicare beneficiaries by physicians and other healthcare professionals. The Physician and Other Supplier Public Use File contains information on utilization, payment (allowed amount and Medicare payment), and submitted charges organized by National Provider Identifier (NPI), Healthcare Common Procedure Coding System (HCPCS) code, and place of service.

Datasets can be found at:

[Note: This Compressed ZIP package contains the tab delimited data file (Medicare_Provider_Util_Payment_PUF_CY2012.txt) which is 1.7GB uncompressed and contains more than 9 million records, thus importing this file into Microsoft Excel will result in an incomplete loading of data. Use of database or statistical software is required; a SAS® read-in statement is supplied. Additionally, this ZIP package contains the following supporting documents: CMS_AMA_CPT_license_agreement.pdf, and Medicare-Provider-Util-Payment-PUF-SAS-Infile.sas]

http://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider

Or the direct link to the dataset (446MB compressed; 1.7 GB after uncompressed) is:

http://download.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Downloads/Medicare_Provider_Util_Payment_PUF_CY2012
We're going to use a sample data set from the UC Irvine Machine Learning Repository, which is a fantastic source for a variety of interesting (and free) data sets for research and education. The data set we'll be analyzing was curated from a record linkage study that was performed at a German hospital in 2010, and it contains several million pairs of patient records that were matched according to several different criteria, such as the patient's name (first and last), address, and birthday. Each matching field was assigned a numerical score from 0.0 to 1.0 based on how similar the strings were, and the data was then hand-labeled to identify which pairs represented the same person and which did not. The underlying values of the fields themselves that were used to create the data set were removed to protect the privacy of the patients, and numerical identifiers, the match scores for the fields, and the label for each pair (match versus

nonmatch) were published for use in record linkage research.

From the shell, let's pull the data from the repository:

```
$ mkdir linkage
$ cd linkage/
$ curl -o donation.zip http://bit.ly/1Aoywaq
$ unzip donation.zip
$ unzip 'block_*.zip'
```

3.2 Preparing the Datasets

Copy all three data files into HDFS. This chapter will assume that the files are available at /user/ds/. Start spark-shell. Note that this computation will take an unusually large amount of memory. If you are running locally, rather than on a cluster, for example, you will likely need to specify –driver-memory 6g to have enough memory to complete these computations.

The first step in building a model is to understand the data that is available, and parse or transform it into forms that are useful for analysis in Spark.

One small limitation of Spark MLlib's ALS implementation is that it requires numeric IDs for users and items, and further requires them to be nonnegative 32-bit integers. This means that IDs larger than about Integer.MAX_VALUE, or 2147483647, can't be used. Does this data set conform to this requirement already? Access the file as an RDD of Strings in Spark with SparkContext's textFile method:

3.3 Analysing the Datasets

<Analytics services used and description>

3.4 Configuring the Computing Cluster

[Grab your reader's attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

3.5 Running Analytics

<General description>

3.6 Analysis Results

<General description>

4 Executing Use Case

Although the data set is in nearly the right form for use with Spark MLlib's ALS implementation, it requires two small extra transformations. First, the aliases data set should be applied to convert all artist IDs to a canonical ID, if a different canonical ID exists. Second, the data should be converted into Rating objects, which is the implementation's abstraction for user-product-value data. Despite the name, Rating is suitable for use with implicit data. Note also that MLlib refers to "products" throughout its API, and so will this example, but the "products" here are artists. The underlying model is not at all specific to recommending products, or for that matter, to recommending things to people:

4.1 NBD Reference Architecture Components

<General description>

5 Lessons Learned and Future Works

<Description>

5.1 Lessons Learned

<General description>

5.2 Future Works

<General description>