A Scalable Big Data Test Framework

Nan Li and Anthony Escalona
Research and Development
Medidata Solutions
{nli, aescalona}@mdsol.com

Yun Guo and Jeff Offutt
Department of Computer Science
George Mason University
{yguo7, offutt}@gmu.edu

***Abstract*—This paper identifies three problems when testing software that uses Hadoop-based big data techniques. First, processing big data takes a long time. Second, big data is transferred and transformed among many services. Do we need to validate the data at every transition point? Third, how should we validate the transferred and transformed data? We are developing a novel big data test framework to address these problems. The test framework generates a small and representative data set from an original large data set using input space partition testing. Using this data set for development and testing would not hinder the continuous integration and delivery when using agile processes. The test framework also accesses and validates data at various transition points when data is transferred and transformed.**

I. Introduction and Problems

The *big data* concept has different definitions but generally it has four "V" characteristics: volume, variety, velocity, and value [1]. The size of datasets used in industry is high; measured in terabytes, petabytes, or even exabytes. Various types of structured and unstructured data need to be processed. The velocity of data generation is high and these data have to be analyzed in a timely manner. People may derive valuable information by mining high volumes of datasets in various formats. Many programs that use big data techniques (e.g., *Hadoop*[1]) and processes big data are currently being developed. How to test such software effectively and efficiently is challenging. Alexandrov et al. [2] presented issues in big data testing and proposed an initial solution that generates huge and realistic data for databases. This paper focuses on generating small and representative datasets from very large sets of data. This can save the cost of processing large amounts of data, which hinders continuous integration and delivery during agile development. Hu et al. [1] agreed that one big data challenge is to test software with small data sets effectively. Our paper introduces a novel scalable big data test framework to test *Extract, Transform, and Load (ETL)* applications that use big data techniques.

In a typical ETL process, data is extracted from an original data source (e.g., a MySQL database), and then is transformed into a structured format to support queries and analysis. Finally, the data is loaded into a target source (e.g., a PostgreSQL database) for customers to view. At Medidata, we compute, store, and analyze high volumes of clinical trial data through ETL processes using the *Amazon Web Service* (AWS). Specifically, we use the Hadoop-based service *Amazon Elastic MapReduce* (EMR) to process the data. Amazon EMR allows us to process hundreds up to hundreds of petabytes of clinical trial data much faster than with conventional methods. We also use Amazon *Simple Storage Service* (S3) and *Redshift*. S3 provides data storage infrastructure on the cloud and Redshift is a data warehouse service. A common scenario at Medidata is to have an ETL application that gets data from S3 or a database, then processes the data on EMR clusters, and then saves the transformed data to S3 or Redshift.

We have identified three technical problems when testing ETL software. First, we have huge amounts of clinical trial data from various studies, subjects and embedded devices. Even with EMR, processing petabytes of data takes days or weeks. Generating small and "representative" data sets for different data sources and clients quickly is challenging. Running an entire or part of historical user data hinders the overall agile development process. We seek to find a smaller data set that represents from this larger population using the characteristics of domain-specific constraints, business constraints, referential constraints, statistical distribution, and other constraints. For brevity, we call this a *representative data set*.

Second, data is transferred and transformed at different transition points during an ETL process. Should we validate the data at one point or all of them? Third, how should we validate transferred and transformed data? Manually validating high volumes of data is prohibitively costly, so this must be automated.

The rest of this paper describes our test framework in as much detail as space allows.

II. Test Framework

To solve these three technical problems above, we are developing a scalable big data test framework to generate representative data sets and validate data transfers and transformation. Figure 1 shows that data coming from different projects (on the left) is stored at various Amazon services and regular storage places (on the right). The purpose of the test framework is to generate a representative data set from a large original data set. Both data sets may be stored at the same or different services. The representative data set can be used for the validation of data transfer and transformation. The test framework validates the data transfer by comparing the data before and after the transfer. Likewise, the framework also validates the data transformation with the requirements that specify how data is transformed. The test framework is currently under development and the prototype has shown promising results.

A. Test Data Generation

Alexandrov et al. [2] proposed a method that extracts constraints from a database and generates a representative data

[1]Apache Hadoop processes large data sets over clusters of computers using *Hadoop Distributed File System (HDFS)*.

978-1-4799-7125-1/15/$31.00 ©2015 IEEE

set for the database. Their proposal did not consider how to control the size of a representative dataset but solely focused on the data representativeness. Furthermore, they only considered data from databases. Other structural data was not considered.

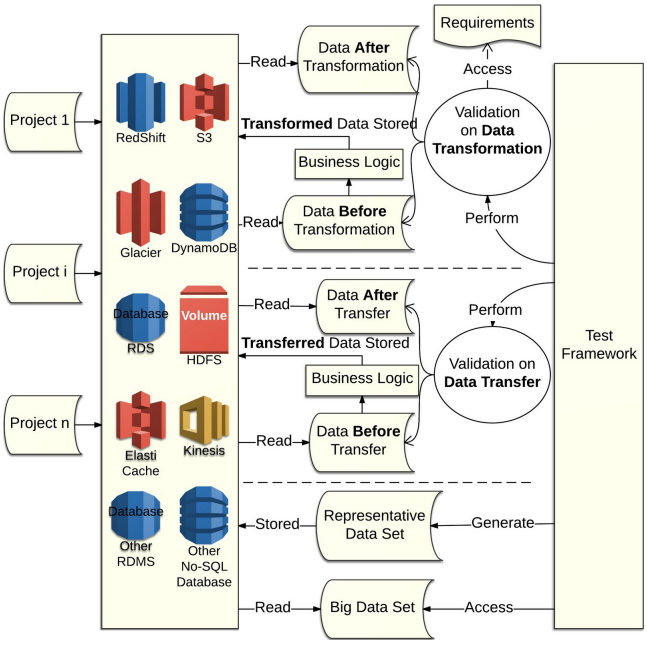

Fig. 1. Big Data Test Framework

Our test framework implements a different test generation method, as shown in Figure 2. To control the size of the test set, we use input space partitioning [3]. Input space partition testing starts with an input-domain model (IDM). The tester partitions the IDM, selects test values from partitioned blocks, and applies combinatorial coverage criteria to generate tests. To the best of our knowledge, this is the first time space partition testing has been used to generate representative data sets in the big data context. Previous input space partition testing methods focused on generating test values to satisfy constraints and seek faults, without processing high volumes of data.

Figure 2 shows the general process of test data generation. At Medidata, audits (another structural data other than database) are generated to reflect how data are changed. To create a representative data set from lots of audits using IDMs, we need to extract test values for every attribute of the audits. We write a grammar to describe the structure of these audits. The test framework then parses the grammar against the audits to collect all test values, including nested attributes, and computes statistical distributions of the test values, resulting in a parse tree. After analyzing the test values and constraints, the IDMs for every attribute are generated and merged into the parse tree. The final representative data set is generated from the parse tree, which has audit attributes, their relationships, constraints, and IDMs. The representative data set is used and evaluated through repeated use. In the future, we plan to use machine-learning techniques to improve the IDMs based on feedback from managers, architects, developers, and testers.

We also plan to use parallel computing and Hadoop to speed up data generation. Even if the process is slow, we only need to generate an initial representative data set once for a project. Then the data set will be refined incrementally to adjust to changing constraints and user values. Our initial prototype took 15 seconds to reduce 21,000 audits to 120.

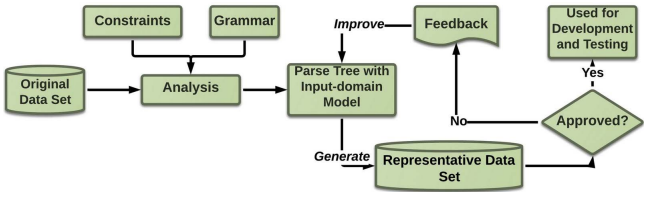

Fig. 2. Simplified Representative Test Data Generation

B. Data Validation

During ETL processes, we need to validate data transfer and transformation to ensure that data integrity is not compromised. Validating data transfer is relatively simple because we know expected values are equal to original values. If the data is transferred from a database to another, we can validate the source and target data quickly by checking the number of columns, rows, and the columns' names and data types. If time allows, every data cell should be evaluated. The validation can be automated when source and target data are provided.

Validating data transformation is more complicated. For instance, we may aggregate data from ten source tables into one target table. Some of the columns of the target table use the same data types as original while other columns may use different data types. We propose two plans at different validation granularity levels. First, we validate whether the target data has correct data types and value ranges at a high level. The test framework derives data types and value ranges from requirements, then generates test to validate the target date. The second plan derives detailed specifications to validate every transformation rule. The test framework compares the source data with the target data to evaluate whether the target data was transformed correctly. Both plans require testers to write the transformation specification in a format that the test framework can read. Then the framework automatically analyzes the specification and generates tests to validate the transformation.

We only validate the source and target data, not the transition points in the middle. These are only checked for failure diagnosis (debugging).

III. Summary

This short paper introduced a significant problem that faces all users of big data, including Medidata, and discusses our solution. The novel test framework described here is under development and will soon be released to the entire company.

References

[1] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *Access, IEEE*, vol. 2, pp. 652–687, June 2014.

[2] A. Alexandrov, C. Brücke, and V. Markl, "Issues in big data testing and benchmarking," in *Sixth International Workshop on Testing Database Systems*, ser. DBTEST '13, New York, NY, USA, 2013.

[3] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge, UK: Cambridge University Press, 2008.