Meta Data Driven Big Data Testing Automation

Anita Kamdi
Department of Computer Engineering
PVPIT Bavdhan Pune, India

Abstract

More and more enterprises are adopting Big Data systems for the benefits it offer, however simpler systems/tools for data quality checks are not readily available and are not easier to implement. Lack of data quality checks leads to inconsistent or incorrect data flowing to the systems consuming it further. As a result data quality may not present correct picture of the insights and data analysis may not be useful for business and operational needs. This paper proposes a simple meta data driven solution to create a test automation tool that will help in executing regular big data loads with ease and perform data quality checks. It can be easily customized to include data quality checks as per requirement.

Keywords: Big Data Quality Assurance, Big data testing Automation, Spark, Hive, Impala

---

I. INTRODUCTION

Big Data systems are characterized by volume, variety and velocity of data that it can process. As such it has gained quick adoption with businesses who wants to leverage insights generated from variety of data sources in quick time. This has resulted in transition of data stored in traditional Data Warehouse Systems to Big Data.

Big Data Hadoop based systems constitute many distributed open source technologies namely distributed data storage, distributed processing, resource management, handling components failover, in-memory and real-time data processing. It is very important that these components work together for overall functioning of the Big Data system. However technical implementations of these systems are still evolving and currently are not comparable with established systems, where processes and policies are clearly in place. The implementation of these systems may not be complicated but when it comes to verifying and validating processes related to data quality and assurance it possesses a major challenge. Again, Big Data characteristics play a bigger role in identifying areas of focus for data quality checks for the organizations business needs. Accordingly, organizations need to identify the sources of data and the level of confidence in data it needs to generate results for deriving business value.

For technical teams, it is easier to verify and validate data quality in DWH systems, owning to the defined nature of the systems and tools. The checks could be performed using scripts or ETL tools and data quality results gathered for specific needs. In case of Big Data Systems, these involve implementing mostly commercial tools or disparate scripts to check. Hive provides a good abstraction of DWH model; however data is not limited to this system only. Querying data through hive and comparisons for data quality checks also presents resourcing constraints for larger datasets, as such tools like Spark, Drill and Impala can be leveraged for performance improvements.

For incremental and full data loads in Big Data the quality checks can be automated using a set of these tools. This paper shows how we can implement such a tool so that regular quality tasks can be automated. The primary objective is to have easy, configurable, customizable system in place to perform required quality checks and generate statistics.

II. RELATED WORK

There is a lot of focus on testing and quality assurance of big data which is evident from the availability of big data systems and tools in market. There are many big data ingestion and data quality tools such as Talend, IDQ, Databricks, Streamsets which can perform these tasks seamlessly. At the same time these tools have challenges such as technical learning curve and meeting expected technical outcomes leading to unsure ROI and extended implementation time.

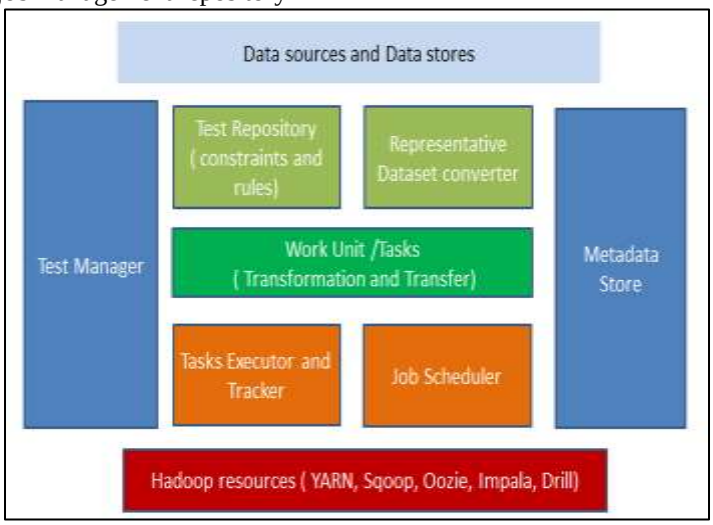
We have referred to many papers which emphasize on establishing quality assurance goals and provide framework/guidelines in doing so in the implementation of Big Data systems [1] [2] [10]. There are also technical papers which propose a good strategy for technical and performance testing of big data systems. However these are just best practices.

There are few open source tools [16] which try to implement ingestion and test the data sets involved but has a complex implementation. The solution we propose here tried to adopt solutions for data optimization techniques mainly generating representative data sets, masking & encryption of data, comparing of large data sets and bloom filter optimization [8] [9] [11] [12].

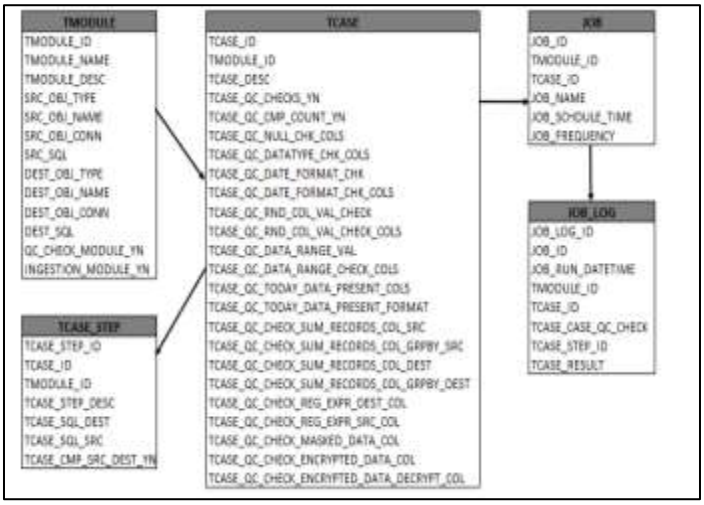
III. SOLUTION OVERVIEW

The proposed solution acts as an application layer between the big data sources and Hadoop system. It uses different libraries based on the test requirement to run queries on the intended data sources and store results to a metastore. The solution is similar to a test execution tool with following components:

- User interaction (Command line interface) – Test Manager
- Meta Store - test case and job management repository



- Scheduler - Scheduling of test cases
- Execute Engine - execution of jobs
- Logger – store job results and statistics in metastore



For each quality check in tcase, the job_log has a row with the results.

A. Execution Engine

The execution engine has libraries for executing Spark, Hive, Drill, Impala, Sqoop and Pig queries. Based on the scheduled job, a session is opened for each of the test case and results passed to logger to store in the metastore.

The execution engine can also perform additional tasks such as generating representative data sets, comparing two data sets using bloom filter optimization, column hashing comparing technique, validating masked data, handling data security for encryption and decryption.

B. Framework Design

The Interface tool performs the following tasks:

- Create and Update Test Module
- Create and Update Quality Checks & Test Cases for the selected Module
- Create and Schedule Jobs for a selected Module
- View the progress of a Module
- View test results and statistics for selected Module

C. Metadata Framework

The Meta Data is implemented using a light weight RDBMS. Every Module can be considered as a test scenario which can be performed on either tables or files. Each Module consists of fixed data quality checks which can be enabled or disabled. Data quality checks can be customized based on requirements. Additional test cases are added to tcase_steps and need to have sql where

clauses. These can be performed both on source and destination data as well as on one of them. Due to the nature of big data, the sql_clauses are not validated and run at execution time only.

Multiple jobs can be scheduled for the same module. Each job results and stats are stored to JOB_LOG table.

IV. IMPLEMENTATION

The solution is implemented using open source technologies. The tool development language can be done using PERL, Python or Shell Script. In our case we used Python for development.

The command line interfaces are developed as python modules for test case generation, job execution, viewing results. The test case specifications are captured as a YAML file.

The execution modules use pyspark, impyla, pyhive, pysqoop, pydrill, pigpy Python client libraries for executing the queries.
Metastore is implemented using sqlite3 DB using Python Alchemy apis.
The job scheduler is implemented using Python Advanced Scheduler libraries.
Create a test module-
.f_testmodule -create -modulename MyFirstTestModule -config /tmp/testmodule1.yaml
Creating a test case
f_testcase -create -modulename MyFirstTestModule -casename MyFirstTestCase -config /tmp/testcase1.yaml
Viewing test results
show all test results:
f_testresults -showresults -modulename MyFirstTestModule -jobname MyFirstJob
show results for a test case:
f_testresults -showresults -casename MyFirstTestCase -jobname MyFirstJob
Execute a job
f_testmodule -run -modulename MyFirstTestModule
Yaml configurations file for sample test case:

```
tcase:
        test_case_desc:Gen_quality_check
        tcase_qc_cmp_count_yn:y
        tcase_qc_null_chk_cols:sales_order_id, product_id
        tcase_qc_datatype_chk_cols:sales_order_dt
        tcase_qc_date_format_chk:MM-DD-YYYY
        tcase_qc_date_format_chk_cols:sales_order_dt,transaction_dt
        tcase_qc_rnd_col_val_check:US
        tcase_qc_rnd_col_val_check_cols:country_id
        tcase_qc_data_range_val:
        tcase_qc_data_range_check_cols:
        tcase_qc_today_data_present_cols:
        tcase_qc_today_data_present_format:
        tcase_qc_check_sum_records_col_src:
        tcase_qc_check_sum_records_col_grpby_src:
        tcase_qc_check_sum_records_col_dest:
        tcase_qc_check_sum_records_col_grpby_dest:
        tcase_qc_check_reg_expr_dest_col:
        tcase_qc_check_reg_expr_src_col:
        tcase_qc_check_masked_data_col:
        tcase_qc_check_encrypted_data_col:
        tcase_qc_check_encrypted_data_decrypt_col:
tstep:
        tcase_step_desc:step-1_check expiry_date>today
        tcase_sql_dest:expiry_date>sysdate
        tcase_sql_src:
        tcase_cmp_src_dest_yn:
tstep:
        tcase_step_desc:step-2_sales_agg
        tcase_sql_dest:product_id is not null
        tcase_sql_src:sales_order_id is not null
        tcase_cmp_src_dest_yn:y
```

V. OBSERVATIONS

- Adding python libraries did not impact execution time of the queries as compared to independent execution of the same query from respective tools
- Session management works as expected

- It is advisable to use sqoop for data ingestion of files, whereas querying of data files is easier for Impala and Drill. Drill provides excellent performance for parquet data
- Query performance is based on the cluster available resources. A three node Hadoop cluster was used to test the tool performance.
- A dataset containing 1 million rows with 20 columns took overall 2 mins to perform record, null, data comparison checks using hive.
- A file containing 1 million rows with 20 columns took ~ 1 min to perform record, null, data comparison checks using drill.
- Encryption was not tested. However masking was tested using regular expression provider.
- Sqllite database capacity testing was not done. This will need to be looked into for large number of jobs.
- Pig load scripts were not tested.

VI. LIMITATIONS

- At present the tool can perform loads and quality checks for batch jobs only. Streaming data checks are not supported.
- Query syntax checks are done only at run time - query validations can be done using calcite
- Job execution statistics are not captured at present. This requires interfacing with yarn cluster.

VII. CONCLUSION

Data quality plays a critical role in Big Data applications. As data goes through various transformations and meanders from upstream to downstream applications, data quality errors propagate and accumulate. These errors have the potential to cause detrimental consequences for an organization or individual. The meta data based data quality testing solution we proposed is intended to serve as a solution model to promote data quality checks in Big Data context. It offers a solution for performing Daily /Bulk load data checks and automate them for a Big Data System. Future customizations may be done to accommodate streaming data checks, different data formats and utilizing pySpark for workflow type parallel checks on the same dataset.

ACKNOWLEDGMENT

My sincere thanks to my guide and all the faculty members who supported in the completion of my work.

REFERENCES

[1] Venkat N. Gudivada, Dhana Rao, and William I. Grosky – "Data Quality Centric Application Framework for Big Data"
[2] Anita Kamdi "Big Data Quality Assurance and Testing Framework" https://ijirst.org/Article.php?manuscript=IJIRSTV4I10010
[3] SQLAlchemy - The Database Toolkit for Python - https://www.sqlalchemy.org
[4] White, T. (2015), Hadoop - The Definitive Guide 4th Edition, OReilly Media.
[5] Apache Hive. Available at http://hive.apache.org
[6] Kelly, J. (2012), Big data: Hadoop, Business Analytics and Beyond, A Big data Manifesto from the Wikibon Community. Available at http : ==wikibon:org=wiki=v=Big Data : Hadoop Business Analytics and Beyond, Mar 2012
[7] QA Consultants : A Primer on Big Data Testing
[8] Donal Miner, Adam Shook : Hadoop MapReduce Design Pat-terns,O'reilly Media Inc, 2012
[9] Hung chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker.Map-reduce-merge:simplified relational data processing on large clusters.Proceedings of the 2007 ACM SIGMOD interna-tional conference on Management of data, pages 1029-1040, 2007.
[10] Drs. D.R. (Dennis) Voges: QUALITY ASSURANCE ON BIG DATA and ANALYTICS, Vrije Universiteit Amsterdam, 2014.
[11] Ritu Jain, Mukesh Rawat, Swati Jain: Data Optimization Techniques using Bloom filter in Big Data: International Journal of Computer Application(0975-8887, May 2016)
[12] Bing Dong: Research and optimization of the Bloom filter algorithm in Hadoop: Uppsala Universitet, Mar 2013
[13] IBM. (2013). Enterprise Information Protection - The Impact of Big Data. Business Intelligence Strategies.
[14] PySqoop - https://pypi.org/project/pysqoop/
[15] pydrill-PyPI – https://pypi.org/project/pydrill
[16] Apache Gobblin – https://apache.gobblin.com