PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

A Project Report
On

Data Quality enhancement and monitoring framework for High Performance Systems leveraging Machine Learning techniques

Submitted in fulfillment of the requirements for the
Project phase -1

Submitted by

Choragudi Praveen

PES1201802271

Under the guidance of

Prof. Suresh Jamadagni

MS

August- December 2019

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM M.TECH

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAM M.TECH

CERTIFICATE

This is to certify that the Dissertation entitled

Data Quality enhancement and monitoring framework for High Performance Systems leveraging Machine Learning techniques

is a bonafide work carried out by

Choragudi Praveen

PES1201802271

In partial fulfillment for the completion of 3rd semester course work in the Program of Study MTECH in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Aug. 2019 – Dec. 2019. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the 3rd semester academic requirements in respect of project work.

Signature with date & Seal	*Signature with date & Seal*	*Signature with date & Seal*
Internal Guide	*Chairperson*	*Dean of Faculty*

Name and Signatures of the Examiners

1.
2.
3.

Table of Contents

1 Introduction

HPC Business is unable to obtain actionable indicators due to diverse data sources that are not integrated, and data is not of right quality.

- Information flows in from over 150+ projects into a central repository.
- The Key master data entities across the sources aren't standardized which leads to ineffective/non-actionable business indicators.
- Information from various sources needs to flow into a centralized DataMart in a summarized fashion. Information lacks traceability; hence data is less actionable.

Potential points of Failure:

- Unable to connect datasets to arrive a meaningful 360^o perspective of data.
- Information drill down, data slicing and dicing is ineffective, and not 360^o.
- Problem finding/fact finding nearly impossible.
- The frequency of occurrence is many times in a day, for thousands of users dealing with billions of records per day.

Scope and its importance:

- Information sources integrated.
- Significant reduction in data quality deviations.

1.1 Problem Definition:

The data that is originating from the source systems is not standardized, integrated, deduplicated and cleansed leading to ineffective/non-actionable business indicators.

1.2 Generic Proposed Solution:

The data must be aggregated to identify the common master data sets; sourcing the "clean" dataset and feed it to the learning engine and apply the generated model to auto-cleanse, suggest or run human-assisted/semi-automated scenarios.

By doing this, we want to achieve significant reduction in data quality deviations with on-the-fly fixing of about 70% of the master data entities, as identified by Data Quality indicators and dynamic recommendations to fix about 20% of the persistent data sets.

2 Literature survey and/or Studies Done

2.1 "Data Lifecycle: From Big Data to Smart Data - an IEEE paper by M. EL ARASS & N. SOUISSI 2018"

Suggests a new Data Lifecycle (DLC) called Smart DLC that helps to make from underdone and insignificant data to Clever Data in a Big Data context. In order to do this, they have followed a method which entails firstly in recognizing and scrutinizing the life cycles and then in defining the stages of their cycle and finally in exhibiting it.

- They have showed data lifecycle that considers data management as "an Information System urbanization project" as **cartography** to support successfully the company data management tasks and their transformations.
- This cartography considers the present situation and makes it possible to better antedate the internal and external developments or limitations impacting data lifecycle and, if necessary, be sure of on technological opportunities.
- The method of sprawl in process cartography aims at a maturation capable of supporting data supervision in the best cost / quality / time.

<u>Conclusion & Inference:</u>

The authors have implemented at very abstract level, but I'm looking from the granular level of voluminous data with different approach achieving traceability.

2.2 "Big Data Quality Assessment Model for Unstructured Data_an IEEE paper by Ikbal Taleb, Mohamed Adel Serhani and Rachida Dssouli 2018"

Proposed a quality assessment model to lever quality of Unstructured Big Data (UBD) that captures and discover first key properties and provides some comprehensive mechanisms to sample, outline the UBD dataset and extract topographies and features from varied data types in diverse formats.

- A Data Quality fountain manage relationships between Data quality magnitudes, quality Metrics, features extraction methods, drawing out procedures, data types and data provinces. An analysis of the samples provides a data profile of UBD.
- Identified the key research challenges related to quality calculation of Unstructured Big Data and highlighted the importance of assessing the quality of such data.
- Their formless big data quality valuation model that stressed the data misuse and feature mining activities was conducted on Word-based, Media and Web data.

<u>Conclusion & Inference:</u>

The type of data dealt is different in the case of this project; but would consider some key aspects discussed in this paper and use them if feasible.

2.3 “Kafka: A Distributed Messaging System for Log Processing - an ACM journal by Jay Kreps, Neha Narkhede, Jun Rao”

A dispersed messaging structure they developed for accumulating and sending high bulks of log data with low dormancy. Their system combines thoughts from current log aggregators and messaging structures and is appropriate for both offline and online message ingesting with processing hundreds of gigabytes of new data daily.

- Kafka is like a messaging system which employs a pull-based ingesting model that allows an application to munch data at its own rate and wind back the consumption whenever wanted.
- They have worked on log processing applications and claim that Kafka attains much sophisticated throughput than conservative messaging structures.
- They are limited to a certain area but desire to build a library of helpful stream efficacies, such as different windowing jobs or join methods will be helpful to various kinds of applications.

Conclusion & Inference:

This is very much useful as we have implemented this for ingesting job history files and found very reliable and we were able to finish the data pre-processing much earlier than expected. But we limit this to data pre-processing only where as the scope of the project is bigger.

2.4 “Data Security and Quality Evaluation Framework: Implementation Empirical Study on Android Devices- by Igor Khokhlov, Leon Reznik, Ashish Kumar, Ankan Mookerjee, Rohan Dalvi”

The main feature of the new method wished-for in this paper comprises a task of security and data quality indicators to data objects. These pointers represent the honesty level, which a data consumer may have.

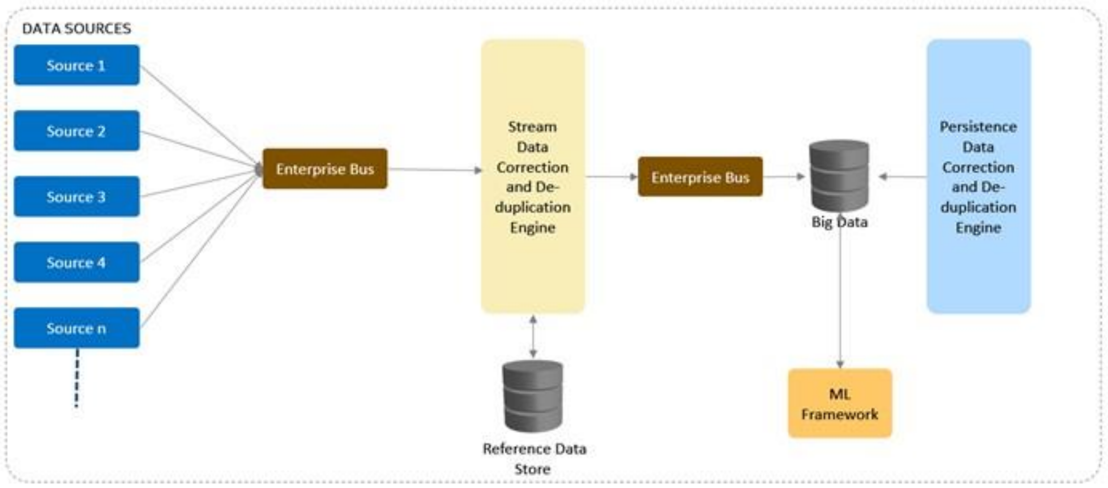
- The framework proposed here unites sensor data collection and security and quality assessment methods as well as actions for calculating various data quality metrics such as sensor correctness, dependability, suitability, precision and their amalgamation.
- It presents the results of an experiential study of the framework application and discusses its claim for an irregularity detection in sensor facts.
- The fallouts of the observed study verified noteworthy variations between the data quality received from different devices of the same model and even more substantial variations between various models.
- The framework proposed and tested allows not only for detecting irregularities in dimensions but also for finding out the sensors and devices, which were not operative.

Conclusion & Inference:

Since we are also dealing with voluminous data security and data quality needs to be addressed. Henceforth, this case study helped me to try similar kind of techniques for our data.

3 System Requirements Specification

3.1 High level block diagram of the Solution



There are two mitigation points:

- One engine that corrects the information on the fly. This engine will be designed to fix less-data-intensive checks.
- The second engine fixes much more regressive issues, that require data-intensive-operations.

Both the system will leverage Machine learning, big data framework, data de-duplication and other data quality techniques to ensure that data meets business expectations.

3.2 Environment Used in the Project

3.2.1 Hardware Required

The details of the VM allocated in order to run Apache Kafka Producer are:

Operating System	Windows Server 2016 Data Center
Processor	Intel® Xeon® CPU E5 – 2698 v4 @ 2.20GHz 2.20GHz (4 processors)
Installed RAM	16.0 GB
System type	64-bit operating system, x64-based processor
Storage	79.9GB

We are running the Apache Kafka Consumer program by submitting the job to the spark on our gateway systems which are Linux based and the processed data is being sent to the target Big Data system via the gateway as we're restricted to access it as there's a firewall built in between for various security concerns.

3.2.2 Software Required

- Python, NumPy - for data analysis
- Big Data technologies: Spark, Spark Streaming, HDFS, Hive, Pig, Scala (tentative), Beehive etc. - for data extraction and performing various operations on streaming and persistent data.
- Middleware: Apache Kafka - for real time data processing with distribution, performance and reliability, REST API - for communication with other modules.
- Spark MLlib - to make practical machine learning scalable and easy using it's built in tools such as ML Algorithms, Featurization, Pipelines, persistence and utilities.

3.2.3 Requirements for the Project

Functional Requirements:

F1. Given the input source location, the elastic dump should get the index file based on the current date and time of the execution.

F2. Once the index file gets downloaded from the source, all the job history files shall be dumped upon proper parsing.

F3. Before dumping of the files, we need to make sure that the Apache Kafka service is running in the specified server location which is given as a parameter to the producer.

F4. The flow of data from the disparate sources isn't limited by time. Henceforth, a time limit can't be defined for the incoming data. To achieve this, the producer must be continually running in the periodic intervals to avoid data loss.

F5. Consumer also should be running parallelly so that no data loss happens in between. The data types of the fields in the data are to be adjusted in order to make more sense of storage.

User Interface:

UI isn't planned as the part of the solution and we may have an UI after the proposed solution is in the production successfully as an additional feature.

Non-Functional Requirements:

NF1. We are creating staging tables because it helps us to identify any data type mismatches.

NF2. The data processing is done in batches so that we need to store for further phases of analysis and to ease the computing as data is voluminous.

3.2.4 Constraints and Dependencies

Risks/Constraints:

- As the data is flowing in from disparate systems, it should be parsed and stored in the database. The risk involved here is to be able to parse voluminous data avoiding memory related errors.

- For the real time data processing, data loss is expected. So, it must be taken care by proper cluster management or by deploying adequate consumer groups or by creating multiple topics in apache Kafka.

Dependencies:

In order to proper functioning of the system, it is essential that the data sources shall not be down which leads to inactivity.

3.2.5 Assumption

No known instances of such an implementation within the organization. A very complex and a huge solution landscape where our aggressive goals include:

- Aggregate and integrate data from extremely diverse structured and unstructured data sources.

- Standardize and deduplicate static/master data using a wide variety of techniques including mapping, machine learning etc.

- Drive integration ratio from 0% to about 80%, helping business with indicators that have not been possible before.

- Integrate the solution as an API with other modules such as ETL and monitoring framework, driving a seamless, end-to-end data lifecycle framework.

3.2.6 Use Case Diagrams for the requirements

- Different persona (or role) deal with the system. Each of the persona have a different expectation of data quality ranging all the way from a summarized level to very detailed granular level.

- Senior management expects summarized data to be cross-mapped to the detailed data sets.

- Individual end users expect the data to be cross-mapped across data sets with 100% traceability.

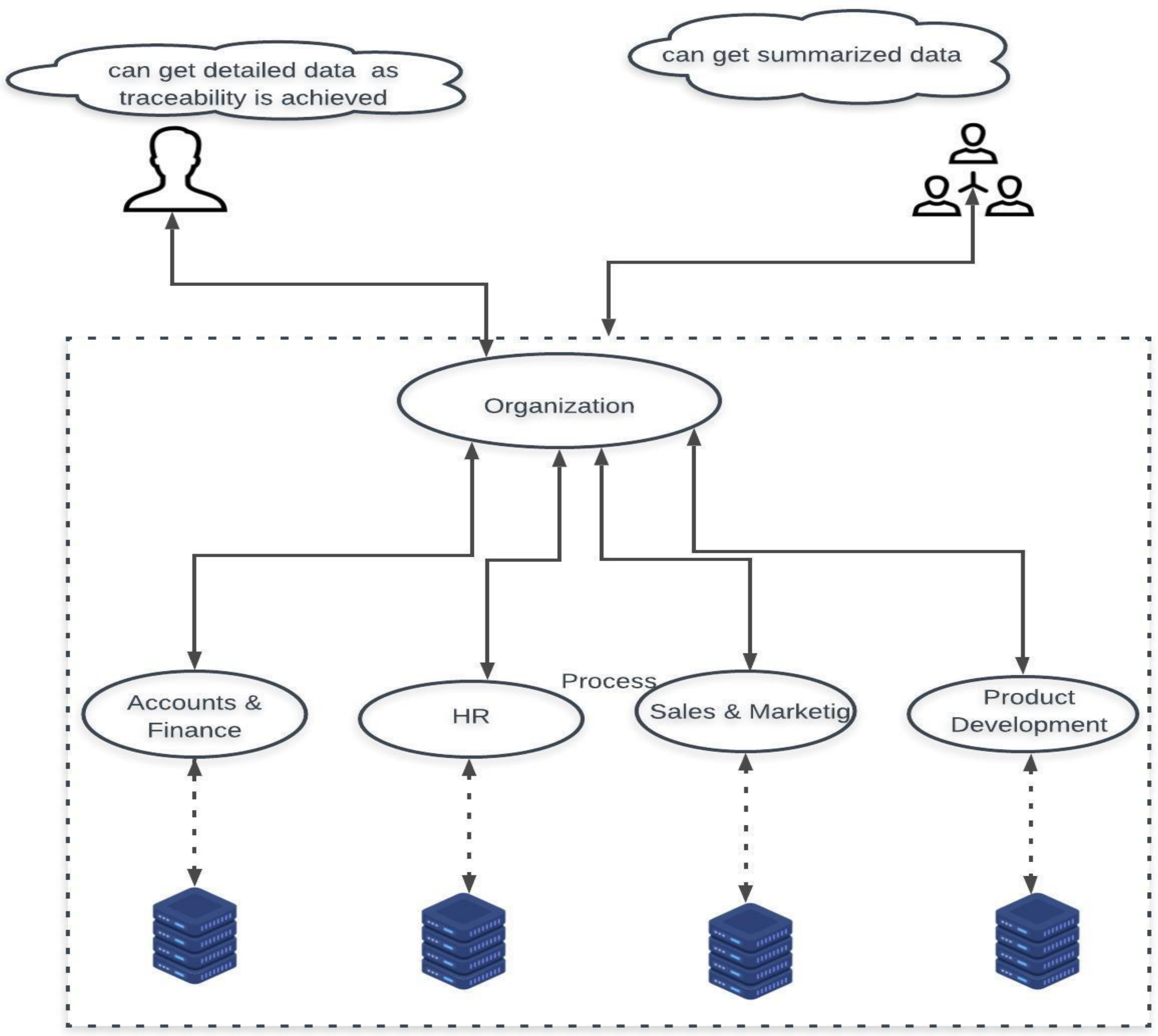


Fig: A sample Use Case diagram to illustrate data is from the disparate sources within the organization.

- Aggregate and integrate data from extremely diverse structured and unstructured data sources.
- Standardize and deduplicate static/master data using a wide variety of techniques including mapping, machine learning etc.
- Drive integration ratio from 0% to about 80%, helping business with indicators that have not been possible before.
- Integrate the solution as an API with other modules such as ETL and monitoring framework, driving a seamless, end-to-end data lifecycle framework.

4 Proposed Methodology

The **Extreme Programming** software development approach is adopted as the proposed solution is research and exploratory in nature.

Benefits:

- Rather than planning, analyzing, and designing in a linear fashion, we can do all such activities a little at a time throughout the development phase.
- Putting a minimal working system which in our case is to extract data from disparate sources to connect data sets to achieve meaningful 360 degrees perspective of data in a continuous manner.
- Facilitating iteration of the accommodating changes as the software evolves with the changing requirements when ingesting data from multiple sources to build a Machine Learning model further.

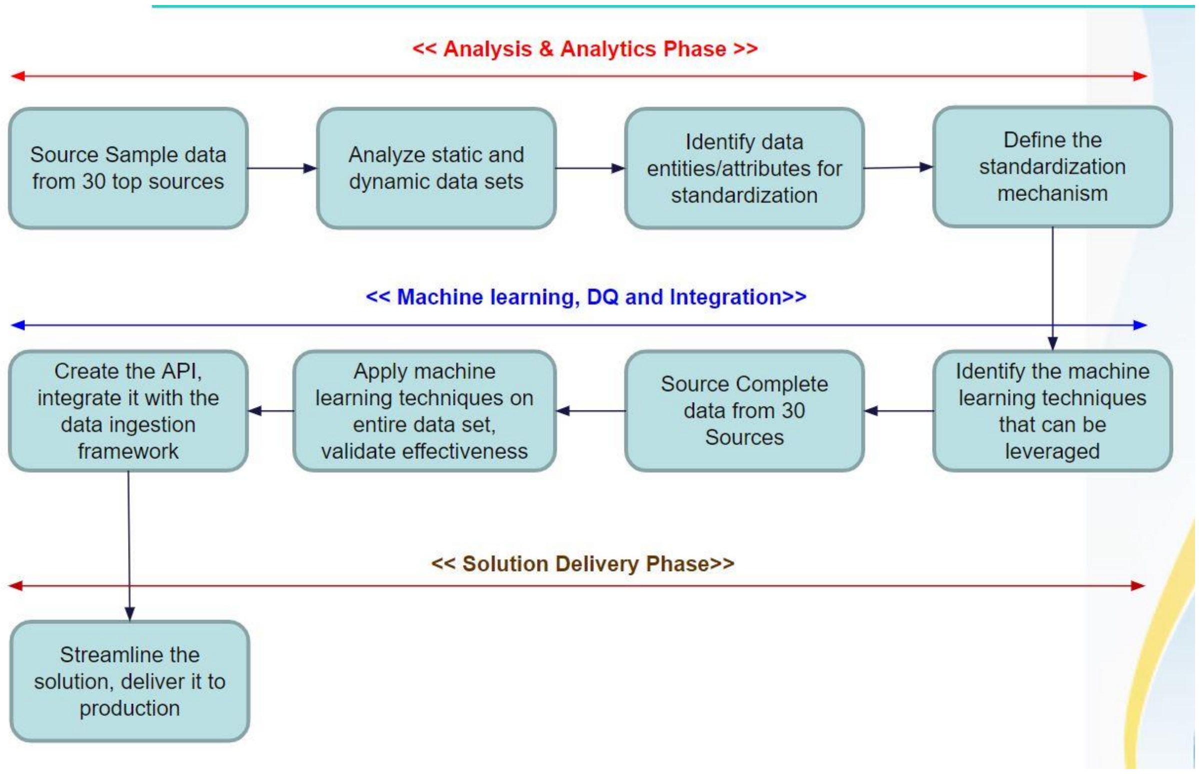
Drawbacks:

- There are no drawbacks so far by following this approach.

Alternate Design approach:

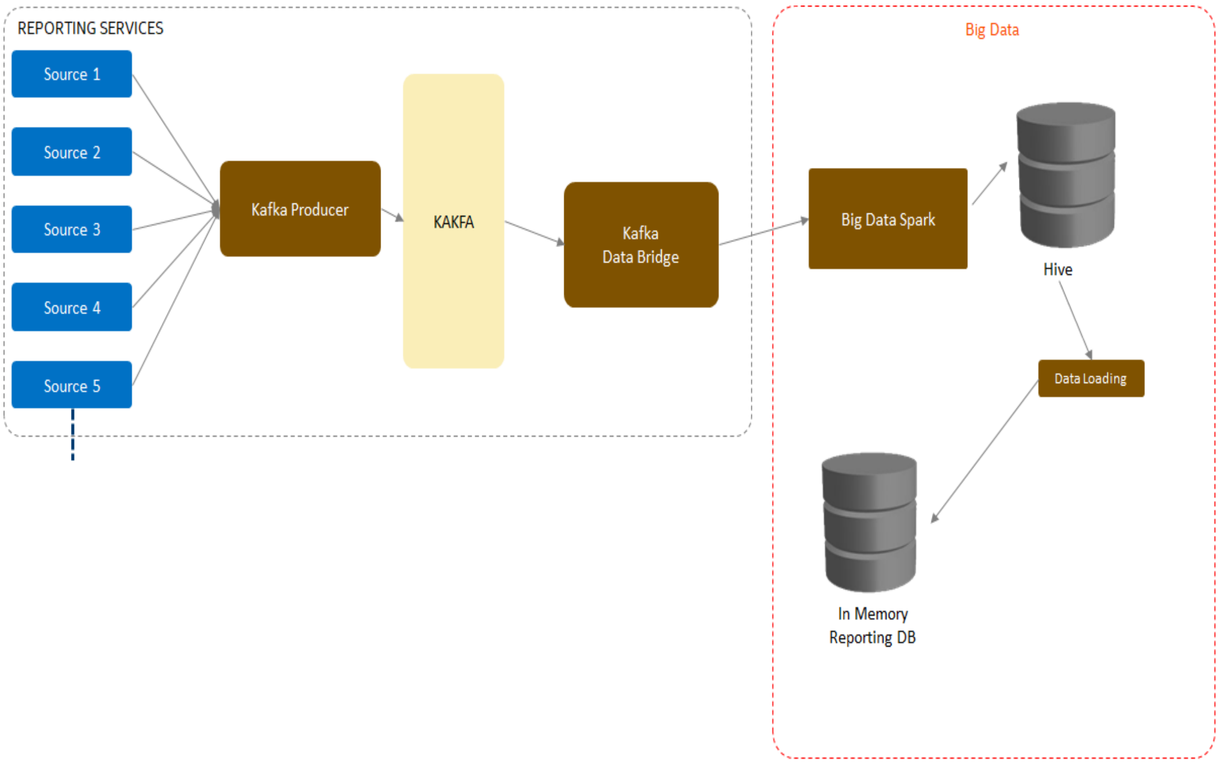
- Dynamic Systems Development Methodology can be an alternative, but this can lead to decrease in code robustness as it focusses on RAD primarily.

4.1 Modular Diagram



<< Analysis & Analytics Phase >>
Source Sample data from 30 top sources
Analyze static and dynamic data sets
Identify data entities/attributes for standardization
Define the standardization mechanism
<< Machine learning, DQ and Integration>>
Identify the machine learning techniques that can be leveraged
Source Complete data from 30 Sources
Apply machine learning techniques on entire data set, validate effectiveness
Create the API, integrate it with the data ingestion framework
<< Solution Delivery Phase>>
Streamline the solution, deliver it to production

4.2 Architectural Diagram

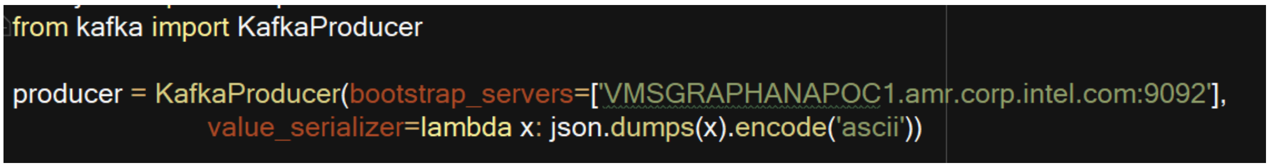


This is the end-to-end data ingestion pipeline which is done completely and now we have data to determine or choose which ML model to use in order to achieve prediction.

5 Implementation

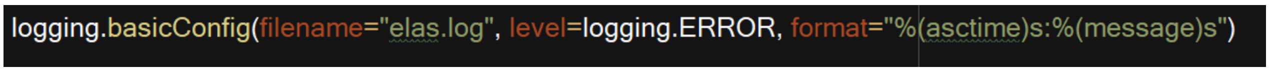
5.1 Pseudo code/algorithms

In our solution we'll create a **producer** that dumps data from http://inlc2748.iind.intel.com:9200/ and send them to our Kafka **broker** which is on the Virtual Machine.



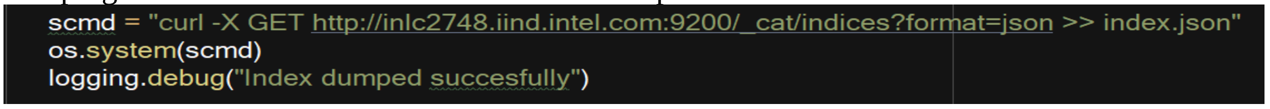
```
from kafka import KafkaProducer

producer = KafkaProducer(bootstrap_servers=['VMSGRAPHANAPOC1.amr.corp.intel.com:9092'],
                         value_serializer=lambda x: json.dumps(x).encode('ascii'))
```

While the Kafka Producer sends the data there may be errors that are happening at runtime, so for that we must debug them;



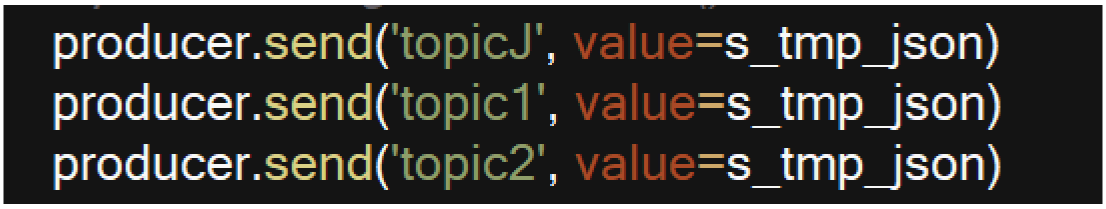
```
logging.basicConfig(filename="elas.log", level=logging.ERROR, format="%(asctime)s:%(message)s")
```

The data we are dealing with one of the sources is in the form of **json**. Hence, we are initially dumping the index of the files that we need to dump.



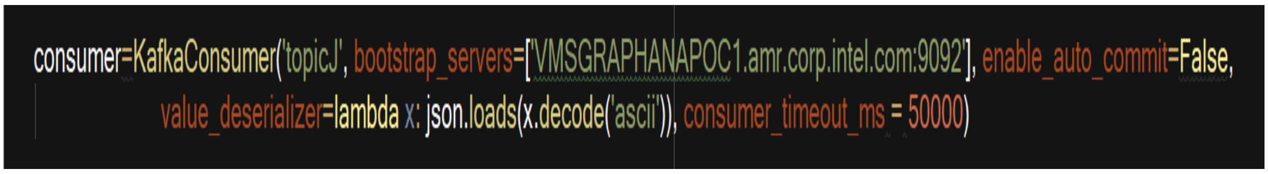
```
    scmd = "curl -X GET http://inlc2748.iind.intel.com:9200/_cat/indices?format=json >> index.json"
    os.system(scmd)
    logging.debug("Index dumped succesfully")
```

Producer sends the data to the broker and the topics were created internally where the subscribers will subscribe to a topic and get only the subscribed data. In our case, we are running three consumers where in each consumer is subscribed to a specific topic.



```
producer.send('topicJ', value=s_tmp_json)
producer.send('topic1', value=s_tmp_json)
producer.send('topic2', value=s_tmp_json)
```

Then a **consumer** will read the data from the **topic** and store them in a Hive table. The Kafka Consumer is enabled in the python script as follows:



```
consumer=KafkaConsumer('topicJ', bootstrap_servers=['VMSGRAPHANAPOC1.amr.corp.intel.com:9092'], enable_auto_commit=False,
            value_deserializer=lambda x: json.loads(x.decode('ascii')), consumer_timeout_ms = 50000)
```

The advantage of using Kafka is that, if our consumer discontinuities, the fixed consumer will pick up reading where the previous one stopped. This is a great way to make sure **all the data is fed into the Hive without duplicates or missing data**.

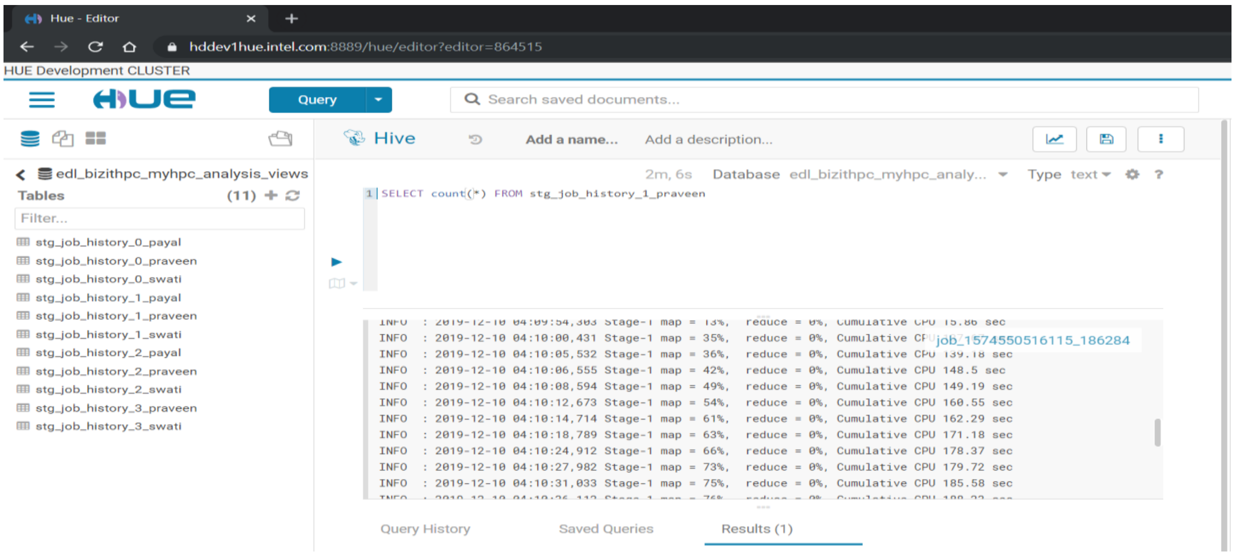
5.2 Data Preprocessing Results

Hadoop Hue is an open source user interface for Hadoop mechanisms. The user can access Hue right from within the browser. This is established by the Cloudera and is an open source project.

Through Hue, the handler can interact with HDFS and MapReduce applications.

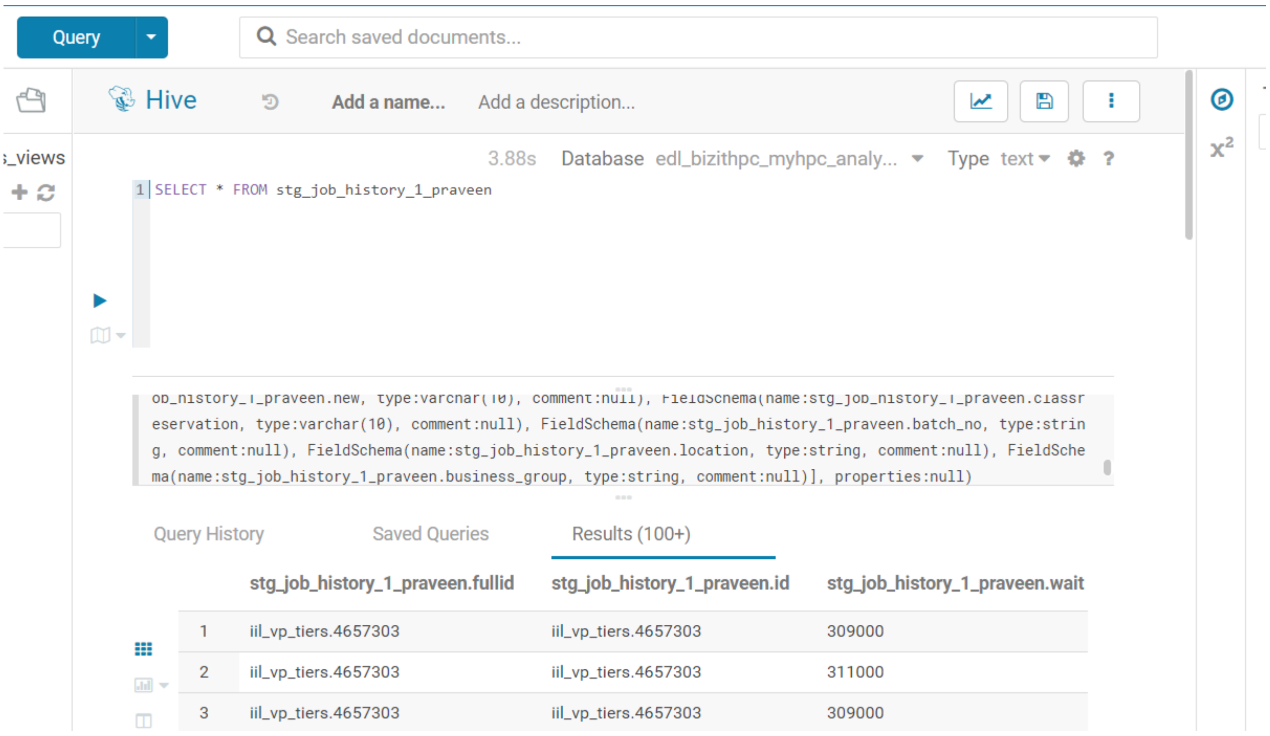
Features:

A lot of features are available in Hue in addition to just a web interface that it provides to the Hadoop developers. Hue provides the features to edit Hive Query. Following image shows the user interface of Hue:



```
SELECT count(*) FROM stg_job_history_1_praveen
```

```
INFO  : 2019-12-10 04:09:54,303 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 15.86 sec
INFO  : 2019-12-10 04:10:00,431 Stage-1 map = 35%,  reduce = 0%, Cumulative CP
INFO  : 2019-12-10 04:10:05,532 Stage-1 map = 36%,  reduce = 0%, Cumulative CPU 139.18 sec
INFO  : 2019-12-10 04:10:06,555 Stage-1 map = 42%,  reduce = 0%, Cumulative CPU 148.5 sec
INFO  : 2019-12-10 04:10:08,594 Stage-1 map = 49%,  reduce = 0%, Cumulative CPU 149.19 sec
INFO  : 2019-12-10 04:10:12,673 Stage-1 map = 54%,  reduce = 0%, Cumulative CPU 160.55 sec
INFO  : 2019-12-10 04:10:14,714 Stage-1 map = 61%,  reduce = 0%, Cumulative CPU 162.29 sec
INFO  : 2019-12-10 04:10:18,789 Stage-1 map = 63%,  reduce = 0%, Cumulative CPU 171.18 sec
INFO  : 2019-12-10 04:10:24,912 Stage-1 map = 66%,  reduce = 0%, Cumulative CPU 178.37 sec
INFO  : 2019-12-10 04:10:27,982 Stage-1 map = 73%,  reduce = 0%, Cumulative CPU 179.72 sec
INFO  : 2019-12-10 04:10:31,033 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 185.58 sec
```



```
SELECT * FROM stg_job_history_1_praveen
```

```
ob_history_1_praveen.new, type:varchar(10), comment:null), FieldSchema(name:stg_job_history_1_praveen.classr
eservation, type:varchar(10), comment:null), FieldSchema(name:stg_job_history_1_praveen.batch_no, type:strin
g, comment:null), FieldSchema(name:stg_job_history_1_praveen.location, type:string, comment:null), FieldSche
ma(name:stg_job_history_1_praveen.business_group, type:string, comment:null)], properties:null)
```

	stg_job_history_1_praveen.fullid	stg_job_history_1_praveen.id	stg_job_history_1_praveen.wait
1	iil_vp_tiers.4657303	iil_vp_tiers.4657303	309000
2	iil_vp_tiers.4657303	iil_vp_tiers.4657303	311000
3	iil_vp_tiers.4657303	iil_vp_tiers.4657303	309000

Fig: Query run to retrieve all contents in the hive table

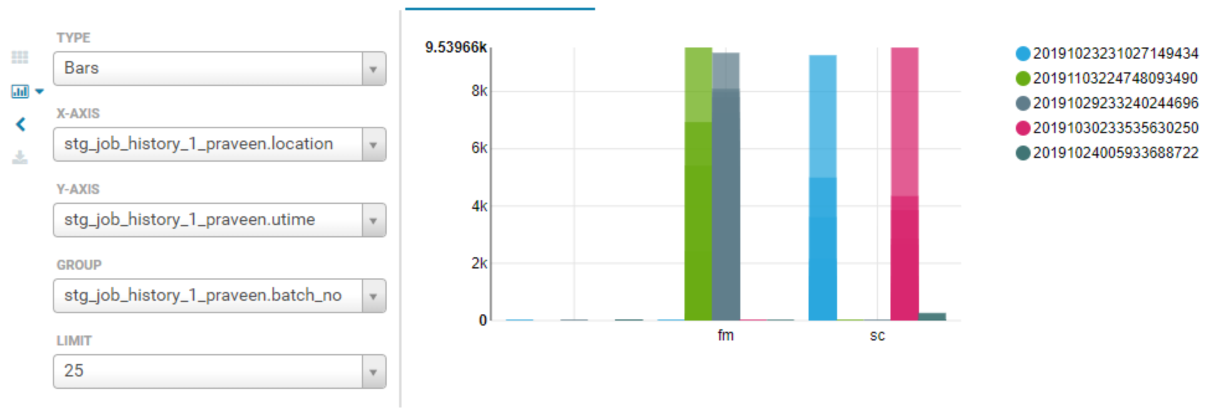
6 Results & Discussions/Conclusions

6.1 Results (intermediate)

Volume **of the incoming data from source system** as of today	
Number of files	6600 (in sizes of gigabytes each)
Number of records in each file	1000 (with 75 columns per record)
Velocity **of the incoming data from source system in** a day	
number of files per day	165 (in size of gigabytes each)
number of records in each file	1000(with 75 columns per record)

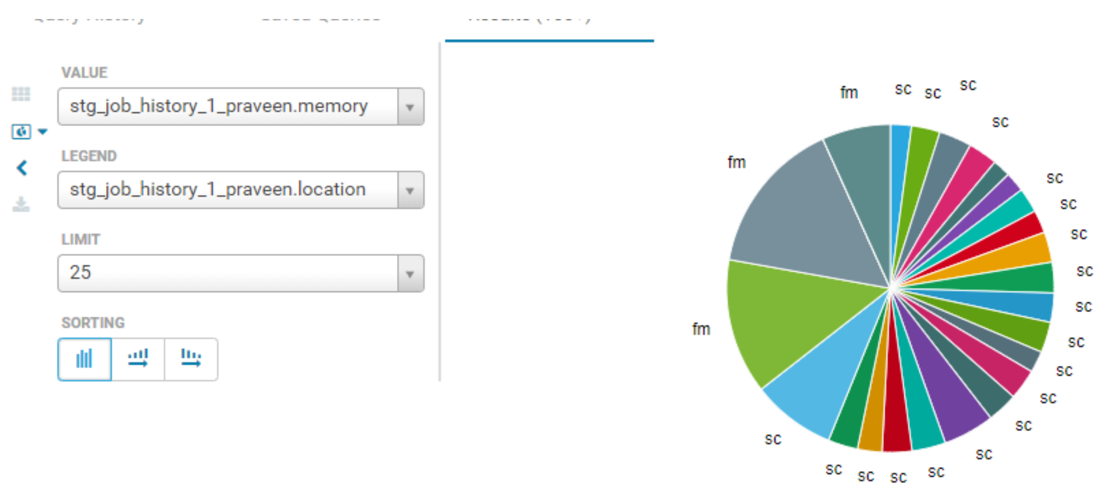
<u>Note</u>: The above status is from sample source of data from one of the disparate systems only.

6.2 Analytical Results post Data Pre-processing



TYPE
Bars
X-AXIS
stg_job_history_1_praveen.location
Y-AXIS
stg_job_history_1_praveen.utime
GROUP
stg_job_history_1_praveen.batch_no
LIMIT
25

Fig: Bar Graph with Location data on X axis and uptime on Y axis based on the batch number



VALUE
stg_job_history_1_praveen.memory
LEGEND
stg_job_history_1_praveen.location
LIMIT
25
SORTING

Fig: Pie diagram showing the memory utilization by location.

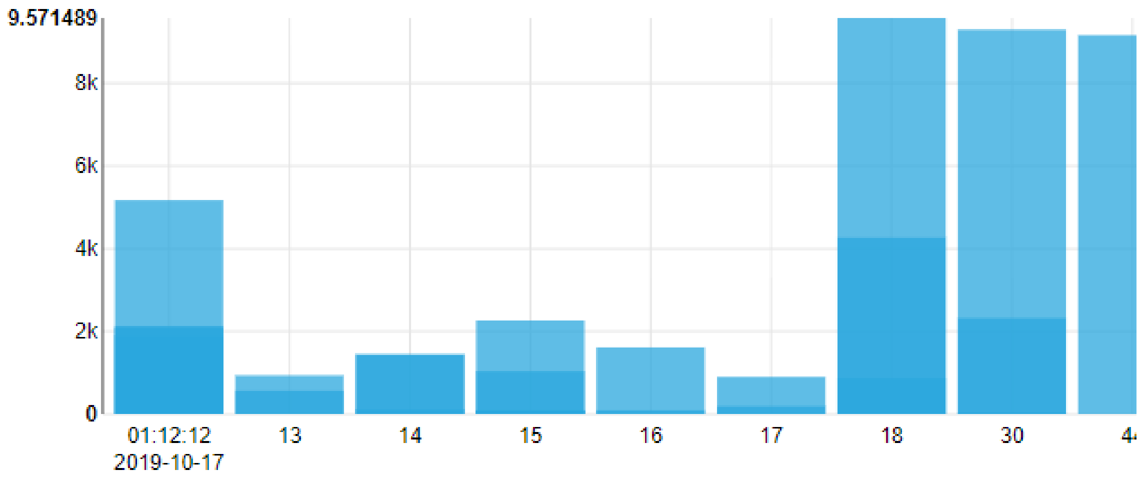


Fig: Above bar graph shows job finished time versus waiting time batch wise.

7 Future Work

7.1 Machine Learning, Data Quality and Integration:

- Identifying the machine learning techniques that can be leveraged.
- Sourcing complete data.
- Apply Machine Learning techniques on entire data set, validate effectiveness.
- Create the API, integrate it with the data ingestion framework.

7.1.1 Solution Delivery:

- Streamlining the solution and deliver it to production.

8 Reference or Bibliography

[1] “http://activemq.apache.org/”

[2] “http://avro.apache.org/”

[3] “Cloudera’s Flume, https://github.com/cloudera/flume”

[4] http://developer.yahoo.com/blogs/hadoop/posts/2010/06/enabling_hadoop_batch_processi_1/

[5] Efficient data transfer through zero copy:

https://www.ibm.com/developerworks/linux/library/jzerocopy/

[6] Facebook’s Scribe, http://www.facebook.com/note.php?note_id=32008268919

[7] IBM Websphere MQ: http://www-01.ibm.com/software/integration/wmq/

[8] http://hadoop.apache.org/

[9] http://hadoop.apache.org/hdfs/

[10] http://hadoop.apache.org/zookeeper/

[11] http://www.slideshare.net/cloudera/hw09-hadoop-baseddata-mining-platform-for-the-telecom-industry

[12] http://www.slideshare.net/prasadc/hive-percona-2009

[13] Kafka, http://sna-projects.com/kafka/