"Piggybank for Mathematical Functions in Pig Latin"

Aditya Patil, Mahesh Kulkarni, Tahseen Mulla, Ishwar Tabe

***Abstract*— Now a days Hadoop is in a boom state. It can be easily accessed because it is open source software which reduces the financial investment of users. Anyone can get the source code easily and can make his own updates. As normally programmer needs lots of efforts to write a code for mathematical functions so, to reduce it we are going to implement jar modules for mathematical functions in Pig Latin called as *User Defined Function (UDF)*. And we will store it into libraries we called it *as piggybank* where a different user can also add their own code i.e. their own UDF to it. So any programmer can access this UDF from piggybank. Programmer can import this inbuilt jar files into his code so he can reduce his work efforts as well as enhance the throughput of the code. At present we are going to implement some mathematical functions like sum, count, average, ascending, descending order by using Pig Latin. This project offers the developers to achieve best time complexity while writing the code by simply importing already created UDF.**

***Index Terms*— Hadoop, Piggybank, Pig-Latin, Open Source, User Defined Function (UDF), Mathematical Functions, MapReduce.**

I. INTRODUCTION

Now a day Hadoop is very popular in global community, because it is an open source and easily available to programmer. At present if programmer wants to write a code of any function in any language then every time he require to writes the full code repeatedly. It takes lots of time to write the code. If there are two programmers from an organization and wants to write the same code at same time then they should write this code each time individually, so to avoid these efforts we are inventing a new concept that is **"Piggybank for Mathematical Functions in Pig Latin" (PMFPL).** We are creating jar files of these functions like sum, count, average, ascending and descending order and storing into the libraries and it called as P*iggybank*. If programmer wants to implement the program of these functions then he can import jar files which has been stored into piggybank, so programmer can reduce his code writing efforts. Through piggybank you can access Java UDFs written by other users and also contribute Java UDFs that you have written. By importing these UDFs, Pig developers can enhance the throughput of the code as well as he can save his time.

We are going to write UDFs in Java for 5 different operations –

- **Sum:** This UDF can be used for computing the sum of the numeric values in a single-column bag.
- **Count:** This UDF can be used for computing the number of elements in a bag.
- **Average:** This UDF can be used for computing the average of the numeric values in a single-column bag.
- **Ascending order:** This UDF can be used for Arranging from smallest to largest elements in a bag.
- **Descending Order:** This UDF can be used for Arranging from largest to smallest elements in a bag.

The Hadoop is software framework used for storage and large-scale processing of a data set on cluster. It is based on the MapReduce programming model and it allows the programmer to distribute work without handling the upcoming problems of the distribution. Pig provides an engine for executing data flows in parallel on Hadoop. It includes a language, Pig Latin, for expressing these data flows. Pig Latin can be extended using UDF where the user can write in Java, Python, JavaScript, etc.

The power and flexibility of Hadoop for big data are immediately visible to software developers primarily because the Hadoop ecosystem was built by developers, for developers. However, not everyone is a software developer. Pig is designed to make Hadoop more approachable and usable by non-developers.

The user can write UDF in Java, Python, JavaScript, Ruby or Groovy and then call directly to their code

Further we are going to elaborate more about Hadoop frameworks. i.e. Hadoop common, Hadoop Distributed File System (HDFS) in Section 1.1. Hadoop YARN (Yet another Resource Negotiator), Hadoop MapReduce we have described in Section 1.2.

Hadoop Common contains libraries and utilities needed by other Hadoop modules.

YARN (Yet another Resource Negotiator) is a resource management platform responsible for managing computer resources in clusters and using them for scheduling of users applications [10].

1.1 Hadoop Architecture:

Hadoop was developed by Doug Cutting and Mike Cafarella in 2005. Doug Cutting was working at Yahoo! at that time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project. [10].

The Hadoop can handle large volume of structured and unstructured data more efficiently than the traditional enterprise data warehouse. Because Hadoop is open source and can run on commodity hardware, the initial cost saving are dramatic and continue to grow as your organizational data

Manuscript received November 21, 2014.
Aditya Patil, Department of Computer Science and engineering, VVPIET Solapur,Solapur, India,9404395119.
Mahesh kulkarni, Department of Computer Science and engineering, VVPIET Solapur,Solapur, India, 9975117521,.
Tahseen Mulla, Department of Computer Science and engineering, VVPIET Solapur,Solapur, India,9595106139,.
Ishwar Tabe , Department of Computer Science and engineering, VVPIET Solapur,Solapur, India,9503714155

grows. Additionally Hadoop has robust Apache community behind it that continues to contribute to its advancement. The true beauty is ability to cost effectively scale to rapidly growing data demands. HDFS uses location awareness method when replicating data to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure, so that even if these events occur, the data may still be readable. Because of these advantages we are using Hadoop instead of other traditional file systems.[9]. The detailed architecture is shown in Fig. 1.1

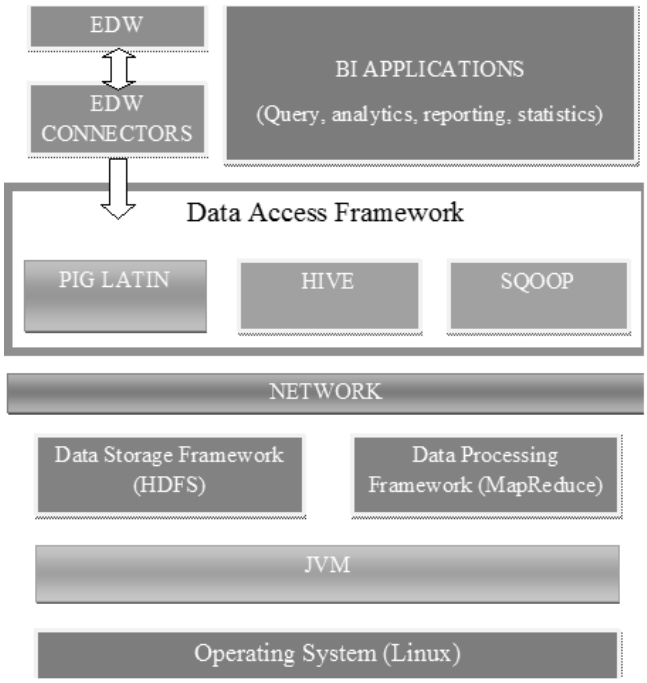

Fig.1.1 Hadoop Architecture for PMFPL

We are going to use this Hadoop architecture for **PMFPL**.

- **Operating System (Linux):** In this architecture we are using Linux Operating System Environment to develop this project (PMFPL). Linux is open source and widely used in organization
- **JVM:** JVM stands for Java Virtual Machine. For running the java code we require JVM.
- **Data Storage Framework (HDFS):** When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. File Systems that manage the storage across a network of machines are called distributed file Systems. Hadoop comes with a distributed file system called HDFS.HDFS is Hadoop's flag ship file System and it is the focus of this project.
- **Network:** To managing the number of systems (more than one) in a network, we are going to use cluster network for interaction between more than one system.
- **Data Access Framework:**
  - *Pig Latin*: High-level platform for creating map-reducing programs used on Hadoop. The language used to express data flows for this platform is called Pig Latin.
  - *Hive*: Hive was created to make it possible for analysts with strong SQL skills (but meagre Java programming skills) to run queries on the huge volumes of data that Facebook stored in HDFS. Today, Hive is a successful Apache project used by many organizations as a general-purpose, scalable data processing platform.
  - *Sqoop:* Sqoop is an open-source tool that allows users to extract data from a relational database into Hadoop for further processing. This processing can be done with MapReduce programs or other higher-level tools such as Hive.
- **EDW:** It stands for Enterprise Data Warehouse. EDW is a system used for reporting and data analysis. Integrating data from one or more disparate sources creates a central repository of data, a data warehouse (DW).
- **BI Applications:** It stands for Business Intelligence. It is used for retrieving the data. These are the high-level applications used in Business.

1.1 MapReduce:

MapReduce is a batch query process. It is a programming model designed for processing large amount of data. It process parallel by dividing the work into a set of independent tasks. It gives people the opportunity to alter or change with data. MapReduce can be seen as a complement to an RDBMS. MapReduce operates on Petabyte of data while traditional database operates on Gigabyte of data. Traditional database uses static structures while MapReduce uses dynamic structure. Because of these advantages we are using MapReduce instead of the traditional database. Here one example of How Hadoop runs a MapReduce job using the classic framework. In below Fig. 1.2 shown how Hadoop runs a MapReduce job using the classic framework.

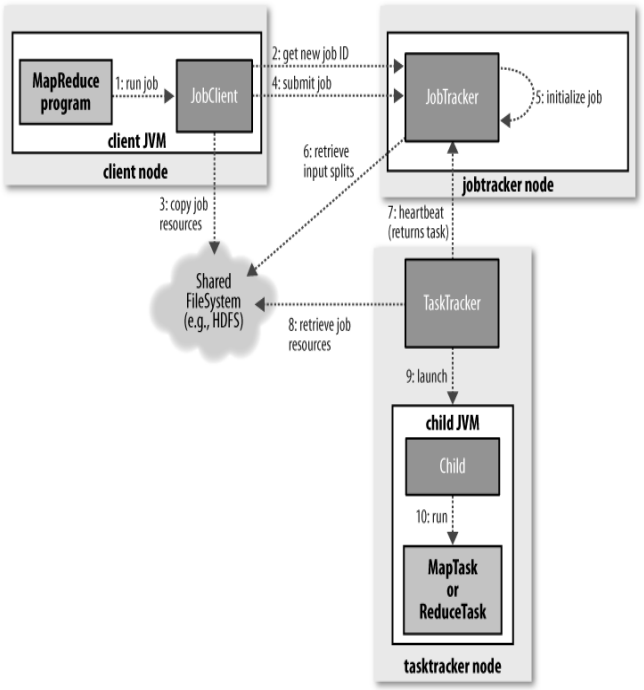

Fig.1.2 How Hadoop runs a MapReduce job using the classic framework

A job running in classic MapReduce is illustrated in Fig. At the highest level, there are four independent entities:
• The client, which submits the MapReduce job.
• The jobtracker, which coordinates the job run. The jobtracker is a Java application whose main class is JobTracker.
• The tasktrackers, which run the tasks that the job has been split into several parts. Tasktrackers are Java applications whose main class is TaskTracker.
• The distributed filesystem (normally HDFS), which is used for sharing job files between the other entities. [11].

1.3 Pig.

Pig provides an engine for executing data flows in parallel on Hadoop. It includes a language, Pig Latin, for expressing these data flows. Pig Latin includes operators for many of the traditional data operations (join, sort, filter, etc.), as well as the ability for users to develop their own functions for reading, processing, and writing data.

Pig is an Apache open source project. This means users are free to download it as source or binary, use it for themselves, contribute to it and under the terms of the Apache License use it in their products and change it as per their requirements.

1.4 Pig on Hadoop.

Pig runs on Hadoop. It makes use of the Hadoop Distributed File System, *HDFS*, and Hadoop processing system. Pig Latin is a dataflow language. This means it allows users to describe how data from one or more inputs should be read, processed, and then stored to one or more outputs in parallel. These data flows can be simple linear flows like the word count example given previously. They can also be complex workflows that include points where multiple inputs are joined, and where data is split into multiple streams to be processed by different operators. To be mathematically precise, a Pig Latin script describes a directed acyclic graph (DAG), where the edges are data flows and the nodes are operators that process the data.

1.5 User Defined Functions.

Much of the power of Pig lies in its ability to let users combine jars operators with their own or others code via UDFs. Up through version 0.7, all UDFs must be written in Java and are implemented as Java classes. This makes it very easy to add new UDFs to Pig by writing a Java class and telling Pig about your JAR file.

As of version 0.8, UDFs can also be written in Python. Pig uses Python to execute Python UDFs, so they must be compatible with Python 2.5 and cannot use Python 3 features. Pig itself comes packaged with some UDFs. Prior to version 0.8, this was a very limited set, including only the standard SQL aggregate functions and a few other functions. In 0.8, a large number of standard string-processing, math, and complex-type UDFs were added.

Piggybank is a collection of user-contributed UDFs that is packaged and released along with Pig. Piggybank UDFs are not included in the Pig JAR, and thus you have to register them manually in your script. Of course you can also write your own UDFs or use those written by other users. Finally, you can use some static Java functions as UDFs as well.

II. System Overview of PMFPL

Here we have shown exact system architecture of our project. In this project actually we are working UDF functions. Here we are creating UDF for sum, average, count, etc. and will create JAR files of that function and store it into piggybank. When developers or programmers want to write code of this function then they can easily imports these JAR files to his program from piggybank so he can reduce his coding efforts.

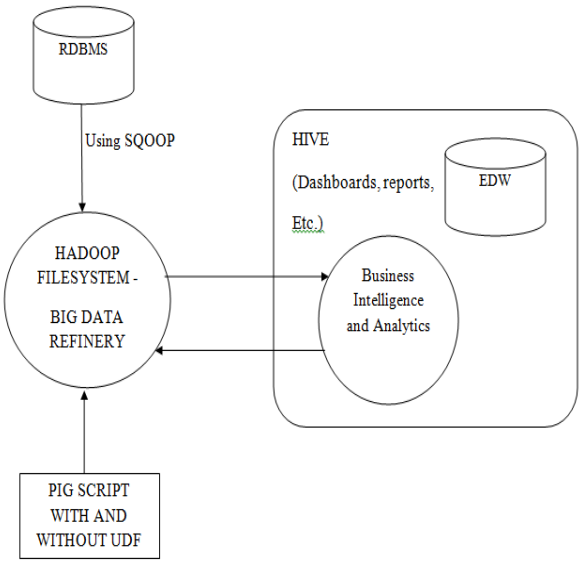

Fig.2 System Architecture of PMFPL

In organization Data has stored into the warehouse by using Relational Database Management System (RDBMS).We take this data from Warehouse to the Hadoop Distributed File System (HDFS) by using Sqoop commands. Sqoop is an open-source tool that allows users to extract data from a relational database into Hadoop for further processing. This processing can be done with MapReduce programs or other higher-level tools such as Hive.

After taking this data into HDFS we perform purification or refinery process on this data. Hive was created to make it possible for analysts with strong SQL skills (but meagre Java programming skills) to run queries on the huge volumes of data that Facebook stored in HDFS. Today, Hive is a successful Apache project used by many organizations as a general-purpose, scalable data processing platform.

BI Applications stands for Business Intelligence. It is used for retrieving the data. These are the high-level applications used in Business.

In Pig Script with and without UDF we will store UDF functions for further use. We can also call it as piggybank. In piggybank we will store JAR files of UDF and programmer can take this JAR files into His/ Her program for reducing the programming efforts.

III. Flow Chart of PMFPL

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analysing,

designing, documenting or managing a process or program in various fields. Here we have shown the overview and representation of how data flows in our system.

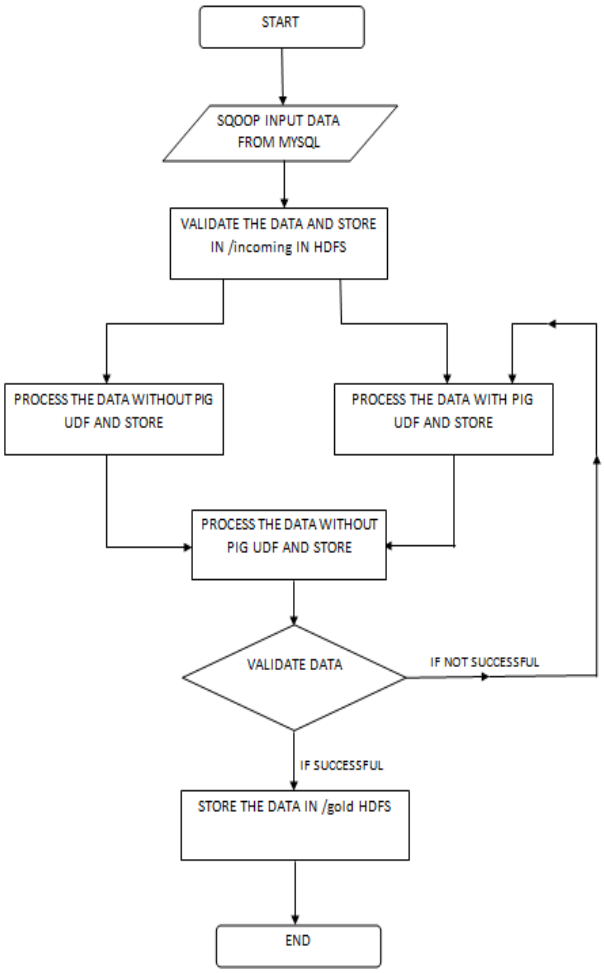

Fig.3 Flow Chart of PMFPL

3.1 Pseudocode Algorithm for PMFPL:

Pseudocode is a newer tool and has features that make it more reflective of the structured concepts. From above flowchart we have developed pseudocode algorithm for PMFPL. In pseudocode we have shown flow of our PMFPL system.

Bring the data in HDFS by using Sqoop Commands.
A1: Perform operations (Sum, Avg.etc) on data by using Pig Script with UDF.
Store result into Temp1.
Perform operations (Sum, Avg.etc) on same data by using Pig Script without UDF.
Store result into Temp2.
Verify Temp1 and Temp2.
If Temp1==Temp2 then
Goto **A2**.
Else
Update the UDF Function and
Goto **A1**.
A2: Store data into Final Output.

Fig.3.1 Pseudocode of PMFPL

3.2 Example of Pig Latin Script for Count:

Pig Latin program is translated in a one-to-one fashion to a logical plan. Fig 3.2 shows an example. Each operator is annotated with the schema of its output data, with braces indicating a bag of tuples. With the exception of nested plans and streaming, a Pig logical query plan resembles relational algebra with user-defined functions and aggregates. Pig currently performs a limited suite of logical optimizations to transform the logical plan, before the compilation into a Map-Reduce plan. We are currently enriching the set of optimizations performed. [13]

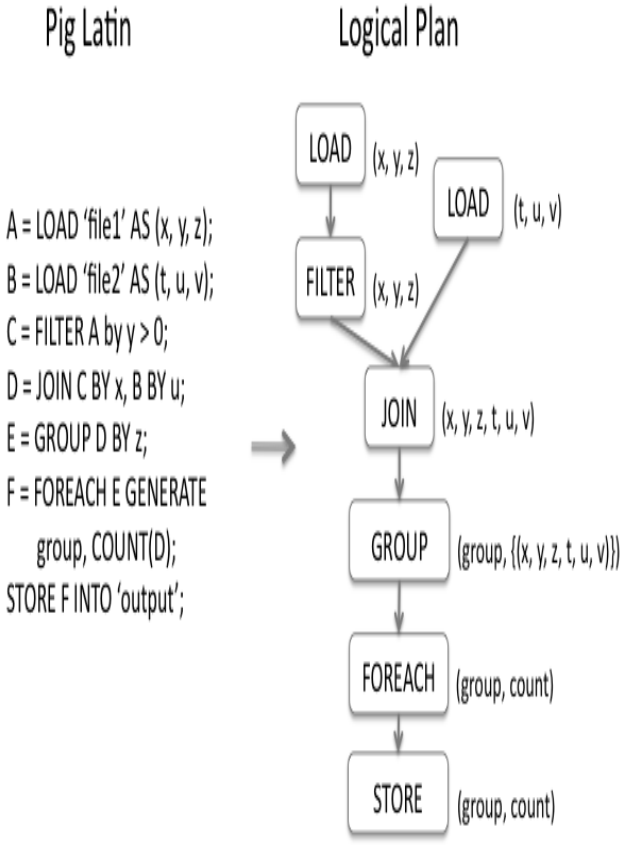

Fig. 3.2 Pig Latin to logical plan translation

Consider a simple Pig Latin example that counts the number of times each word appears in a given bag. Here we have shown example how to count the words. Here, firstly we load two files i.e. 'file 1' and 'file 2' then we shuffle and filter the data. Then we join two files later it forms a group where we count the words. The pig Latin script and logical plan of this example is shown in Fig.3.2.

3.3 MapReduce Example:

A Hadoop Map-Reduce job consists of a series of execution stages, shown in Fig. 3.3. The map stage processes the raw input data, one data item at a time, and produces a stream of data items annotated with keys. Local sort stage orders the data produced by each machine's. The shuffle stage then redistributes data among machines. All data received at a particular machine is combined into a single ordered stream in the merge stage. Reduce function that merges all intermediate values associated with the same intermediate key. The

pictorial representation of MapReduce is shown in Fig. 3.3. [13]

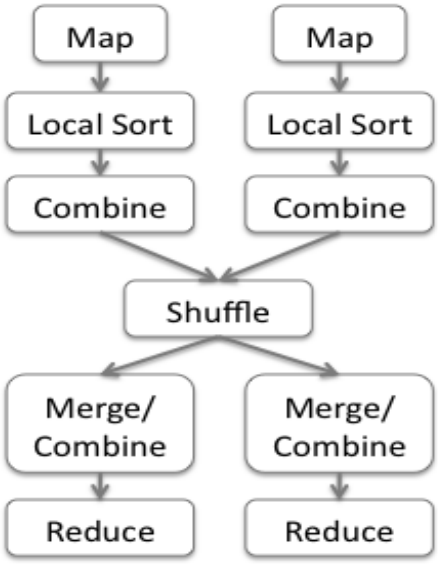

Fig.3.3 Map-Reduce execution stages

IV. Requirements for PMFPL

We have completed our system designing part and now we will proceed with exact development of the process. For developing this system we will requires this software and hardware as listed below.

4.1 Software:

1. Ubuntu 12.04 LTS 64 bit.
2. Eclipse IDE.
3. Java 1.8 64bit.
4. Hadoop Version 2.0.
5. Pig Latin 1.2.

4.2 Hardware:

1. CPU: Intel Core i3 (2.27 GHz) Processor with 2.27GHz.
2. Memory: 4 GB Minimum.
3. Disk Space: 100GB Minimum.

V. Future Scopes

The project is identified by the merits of the system offered to the user which will enhance its area of application in future. The main objective of this project is to optimize the developer's time and code length as just importing or calling these User Defined functions in their code. As these operations or function for which we are going to write the UDFs are very general and it can be used in many projects. So the scope of this is very high.

For testing purpose, we will write Pig Scripts in traditional way and in the UDF way. We will check the differences of the output for different input formats. If the output matches, the purpose of building these UDF will match. This project offers the developers to achieve best time complexity while writing the code by simply importing already created User Defined Functions.

VI. Conclusion

By using PMFPL, we will achieve the superfast processing on big data in any format and by writing User Defined Functions in Java for 5 different operations –

- Sum
- Count
- Average
- Ascending order
- Descending Order

We will write a reusable code with high scope as these operations can be done quite often on the big data.

References

[1] http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[2] http://hortonworks.com/hadoop/pig/
[3] http://pig.apache.org/
[4] https://developer.yahoo.com/hadoop/tutorial/
[5] http://hadoop.apache.org
[6] http://wiki.apache.org/pig/PigMix
[7] http://hadoop.apache.org/hive/
[8] http://hadoop.apache.org/pig
[9] http://www.metascale.com/why-hadoop.html
[10] http://en.wikipedia.org/wiki/Apache_Hadoop
[11] Hadoop the Definitive guide 3rd Edition by Tom White
[12] Programming Pig by Alan Gates
[13] Building a High-level Dataflow System on top of MapReduce by Alan F. Gates, Olga Natkovich
[14] Hadoop:Open-source implementation of MapReduce.
[15] The Hive Project.
[16] Pig Latin: A Not-So-Foreign Language for Data Processing.
[17] MapReduce: Simplified Data Processing on Large Clusters.
[18] Building a High Level Dataflow System on top of MapReduce: The Pig Experience