Impact Factor (SJIF): 3.632

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 3, Issue 2, February-2016

A SURVEY ON HADOOP PIG SYSTEM

Nikita Bhojwani[1] ,Asst Prof. Vatsal Shah[2]

[1 2]B.V.M. Engineering College, V.V. Nagar, Gujarat, India

Abstract *— Pig is a platform for analyzing large datasets using a high-level and expressive language called pig latin, which enable users to describe data-processing steps. It is the amalgamation of SQL and map-reduce. With pig, the programming becomes easy. With very few lines of code, it deals with intricate type of data. It deals with unstructured data. It works with real time applications. It is used for data mining. Pig is also like hive but there are some significant differences such as hive supports a declarative sql like language whereas pig supports a flow language that is suitable for data processing steps. It provides data structures such as relations that are similar to database tables that comprises of rows. It bolsters various types of LOAD, FILTER, JOIN, FOREACH, GROUP, STORE etc. along with that pig also provides extensible support for user defined functions(UDFs) as a way to custom processing. Pig's semantics are comparatively easier than map- reduce. Its language is pig latin and can be compared to python. Additionally, Apache pig processes network flow data to process large data sets and can be accessed in dynamic environments.*

KEYWORDS: *Pig, hadoop, SQl, Hive , MapReduce, Piglatin*

I. NTRODUCTION

1.1. Pig

Pig is a high-level dataflow system that aims at a sweet spot between SQL and Map-Reduce. SQL(Structured Query Language) is a special purpose programming language designed for managing data held in a relational database management system(RDBMS), or for stream processing in a relational database management system(RDSMS). Map reduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. Pig offers SQL-style high-level data manipulation constructs, which can be assembled in an explicit dataflow and interleaved with custom Map- and Reduce-style functions or executables. Pig programs are compiled into sequences of Map-Reduce jobs, and executed in the Hadoop Map-Reduce environment.
Pig system retains the properties of Map-Reduce systems that make them attractive for certain users, data types, and workloads[2]. In particular, as with Map-Reduce, Pig programs encode explicit dataflow graphs, as opposed to implicit dataflow as in SQL[2].

Pig dataflows can interleave built-in relational-style operations like filter and join, with user-provided executables (scripts or pre-compiled binaries) that perform custom processing. Schemas for the relational-style operations can be supplied at the last minute, which is convenient when working with temporary data for which system-managed metadata is more of a burden than a benefit[2].

Pig compiles these dataflow programs, which are written in a language called Pig Latin into sets of Hadoop MapReduce jobs, and coordinates their execution. By relying on Hadoop for its underlying execution engine, Pig benefits from its impressive scalability and fault-tolerance properties. On the other hand, Pig currently misses out on optimized storage structures like indexes and column groups. There are several ongoing efforts to add these features to Hadoop. Despite leaving room for improvement on many fronts, Pig has been widely adopted in Yahoo, with hundreds of users and thousands of jobs executed daily, and is also gaining traction externally with many successful use cases reported. Pig provides a scripting language to execute MapReduce jobs as an alternative to writing Java code. Pig's scripting language is called Pig latin[2].

Using Pig reduces the time needed to write mapper and reducer programs. This means that no Java is required, and there is no need for boilerplate code[5]. You also have the flexibility to combine java code with pig . Many complex algorithms can be written in less than five lines of human readable Pig code. 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[5].

Pig Latin, which has the following key properties[9][11]:

- Ease of programming**.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- Optimization opportunities**.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- Extensibility. Users can create their own functions to do special-purpose processing.

1.1.1 WHAT PIG DOES?

Pig was mainly designed for performing a long set of data operations, making it ideal for three categories of Big Data operations:

- Standard extract-transform-load (ETL) data pipelines,
- Research on raw data, and
- Iterative processing of data.

II. BACKGROUND THEORY

2.1.Pig Latin

A Pig Latin program is a collection of statements. A statement can either be an operation or a command[3]. For example, to load data from a file, issue the LOAD operation with a file name as an argument. A command could be an HDFS command used directly within PigLatin such as the ls command to list, say, all files with an extension of txt in the current directory. The execution of a statement does not necessarily immediately result in a job running on the Hadoop cluster.[3] All commands and certain operators, such as DUMP will start up a job, but other operations simply get added to a logical plan. This plan gets compiled to a physical plan and executed once a data flow has been fully constructed, such as when a DUMP or STORE statement is issued. Here are some of the kinds of statements in PigLatin. There are UDF statements that can be used to REGISTER a user-defined function into PigLatin and DEFINE a short form for referring to it. It has been mentioned that HDFS commands are a form of PigLatin statement. Other commands include MapReduce commands and Utility commands[3]. There are also diagnostic operators such as DESCRIBE that works much like an SQL DESCRIBE to show you the schema of the data. The largest number of operators falls under the relational operators' category[3]. PigLatin supports the standard set of simple data types like integers and character arrays. PigLatin also supports functions. These include eval functions, which take in multiple expressions and produce a single expression. For example, you can compute a maximum using the MAX function. A filter function takes in a bag or map and returns a Boolean. For example, the Empty function can tell you if the bag or map is empty[3]. A load function can be used with the LOAD operator to create a relation by reading from an input file. A store function does the reverse. It reads in a relation and writes it out to a file. You can write your own eval, filter, load, or store functions using PigLatin's UDF mechanism[3]. You write the function in Java, package it into a jar, and register the jar using the REGISTER statement. You also have the option to give the function a shorter alias for referring to it by using the DEFINE statement[3].

2.1.1. Advantages of Pig include:[11]

- Pig is much higher level declarative language than MapReduce which increases programmer productivity, decreases duplication of effort and also opens the MapReduce programming system to more users.
- Pig insulates Hadoop complexity from the user.
- Pig is similar to SQL query where the user specifies the "what" and leaves the "how" to the underlying processing engine.

2.1.2. Pig is useful for:[11]

- time sensitive data loads[6]
- processing many data sources and[6]
- for analytic insight through sampling.

2.1.3. Pig is not useful:[11]

- When processing really nasty data formats of completely unstructured data (e.g. video, audio, raw human-readable text).
- It is definitely slow in comparison to MapReduce jobs.
- When more power is required to optimize the code.

2.2. Modes of User Interaction with Pig

Pig has two execution type[2]s:

- **Local Mode**: To run Pig in local mode, users need access to a single machine; all files are installed and run using the local host and file system. Specify local mode using the -x flag (pig -x local). Note that local mode does not support parallel mapper execution with Hadoop 0.20.x and 1.0.0
- **MapReduce Mode**: To run Pig in mapreduce mode, users need access to a Hadoop cluster and HDFS installation. Mapreduce mode is the default mode; user can,but don't need to, specify it using the -x flag (pig OR pig -x mapreduce).

2.2.1Pig allows three modes of user interaction[2]:

- **Interactive mode**: The user is presented with an interactive shell called Grunt, which accepts Pig commands. Compilation and execution plan is triggered only when the user asks for output through the STORE command.
- **Batch mode**: In this mode, a user submits a pre-written script called Pig script, containing a series of Pig commands, typically ending with STORE. The semantics are identical to interactive mode.
- **Embedded mode**: Pig Latin commands can be submitted via method invocations from a Java program. This option permits dynamic construction of Pig Latin programs, as well as dynamic control flow.

3.1. Programming Language

- it’s significantly easier than having to write mapper and reducer programs.
- 1. The first step in a Pig program is to LOAD the data you want to manipulate from HDFS.
- 2. Then you run the data through a set of transformations (which, under the covers, are translated into a set of mapper and reducer tasks).
- 3. Finally, you DUMP the data to the screen or you STORE the results in a file somewhere.

- LOAD

- As is the case with all the Hadoop features, the objects that are being worked on by Hadoop are stored in HDFS. In order for a Pig program to access this data, the program must first tell Pig what file (or files) it will use, and that’s done through the LOAD 'data_file' command (where 'data_file' specifies either an HDFS file or directory). If a directory is specified, all the files in that directory will be loaded into the program. If the data is stored in a file format that is not natively accessible to Pig, you can optionally add the USING function to the LOAD statement to specify a user-defined function that can read in and interpret the data[5].

- TRANSFORM
  The transformation logic is where all the data manipulation happens. Here you can FILTER out rows that are not of interest, JOIN two sets of data files, GROUP data to build aggregations, ORDER results etc.

- DUMP and STORE
  If you don’t specify the DUMP or STORE command, the results of a Pig program are not generated. You would typically use the DUMP command, which sends the output to the screen, when you are debugging your Pig programs. When you go into production, you simply change the DUMP call to a STORE call so that any results from running your programs are stored in a file for further processing or analysis[5]. You can use the DUMP command anywhere in your program to dump intermediate result sets to the screen, which is very useful for debugging purposes.

3.1.1 Pig Compilation and Execution Stages[2][11]

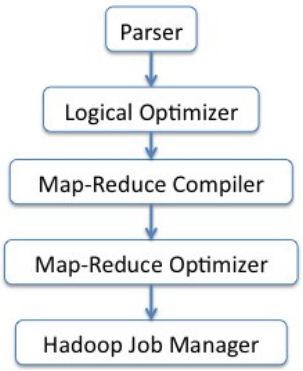

Figure1: pig compilation and execution stages[2]

- A Pig program goes through a series of transformation steps before being executed as shown in the above figure.
- **Parsing:** This is the first step. The parser duty is to verify that the program is syntactically correct and that all referenced variables are defined. Type checking and schema inference can also be done by parser. Other checks, such as verifying the ability to instantiate classes corresponding to user-defined etc., also occur in this phase. A canonical logical plan with one-to-one correspondence between Pig Latin statements and logical operators which are arranged in directed acyclic graph(DAG) is the output of the parser.
- **Logical Optimizer**: The logical plan generated by the parser is then passed through a logical optimizer. In this stage, logical optimizations such as projection pushdown are carried out.
- **Map-Reduce Compiler and Map-Reduce Optimizer**: The optimized logical plan is then compiled into a series of Map-Reduce jobs, which then pass through another optimization phase. An example of Map-Reduce-level optimization is utilizing the Map-Reduce combiner stage to perform early partial aggregation, in the case of distributive or algebric aggregaton functions.
- **Hadoop Job Manager**: The DAG of optimized Map-Reduce jobs is then topologically sorted, and then are submitted to Hadoop for execution in that order. Pig usually monitors the Hadoop execution status, and the user periodically gets the reports on the progress of the overall Pig program. Any warnings or errors that arise during execution are logged and reported to the user.

3.1.2 Pig Data Model

Data model can be defined as follows:

- A relation is a bag of tuples.
- A bag is a collection of tuples (duplicate tuples are allowed). It may be similar to a "table" in RDBMS, except that Pig does not require that the tuple field types match, or even that the tuples have the same number of fields. A bag is denoted by curly braces {}. (e.g. {(Anita, 1.0), (Anita, 2.0)}).
- A tuple is an ordered set of fields. Each field is a piece of data of any type (data atom, tuple or data bag) (e.g (Anita, 1.0) or (Anita, 2.0)).
- A field is a piece of data or a simple atomic value. (e.g. ‘Anita’ or ‘1.0’)
- A Data Map is a map from keys that are string literals to values that can be any data type. (Key is an atom while value can be of any type). Map is denoted by square brackets .

3.1.3 Pig Latin Relational Operators

Loading and Storing:
LOAD: Loads data from the file system or other storage into a relation.

STORE: Saves a relation to the file system or other storage.

DUMP: Prints a relation to the console.

Filtering:
FILTER: Removes unwanted rows from a relation.

FOREACH…GENERATE: Generates data transformations

STREAM: Transforms a relation using an external program.

Grouping and Joining:
JOIN: Joins two or more relations.

COGROUP: Groups the data in two or more relations or we can also say join in SQL.

GROUP: Groups the data in a single relation.

CROSS: Creates the cross product of two or more relations.

Sorting:
ORDER: Sorts a relation by one or more fields.

LIMIT: Limits the size of a relation to a maximum number of tuples.

Combining and Splitting:
UNION: Combine two or more relations into one.

SPLIT: Splits a relation into two or more relations.

4.1. PIG ON HADOOP

Pig runs on Hadoop. It makes use of both the Hadoop Distributed File System, HDFS, and Hadoop's processing system, MapReduce[6].

HDFS is a distributed filesystem and the first part of hadoop that stores files across all of the nodes in a Hadoop cluster[4]. It handles breaking the files into large blocks and distributing them across different machines, including making multiple copies of each block so that if any one machine fails no data is lost. It replicates the data frequently and hence making the process safe. It presents a POSIX-like interface to users. By default, Pig reads input files from HDFS, uses HDFS to store intermediate data between MapReduce jobs, and writes its output to HDFS[4,6].

MapReduce is a simple but powerful parallel data-processing paradigm. Every job in MapReduce consists of three main phases: map, shuffle, and reduce. In the map phase, the application has the opportunity to operate on each record in the input separately[4,6]. Many maps are started at once so that while the input may be gigabytes or terabytes in size, given enough machines, the map phase can usually be completed in under one minute[4,6].

Part of the specification of a MapReduce job is the key on which data will be collected. For example, if you were processing web server logs for a website that required users to log in, you might choose the user ID to be your key so that you could see everything done by each user on your website[6]. In the shuffle phase, which happens after the map phase, data is collected together by the key the user has chosen and distributed to different machines for the reduce phase. Every record for a given key will go to the same reducer[6].

In the reduce phase, the application is presented each key, together with all of the records containing that key. Again this is done in parallel on many machines. After processing each group, the reducer can write its output[6].

4.1.1 PIG SCRIPTS

I. PIG SCRIPT 1: QUERY PHRASE POPULARITY

The Query Phrase Popularity script (script1-local.pig or script1-hadoop.pig) processes a search query log file from the Excite search engine and finds search phrases that occur with particular high frequency during certain times of the day.

II. PIG SCRIPT 2: TEMPORAL QUERY PHRASE POPULARITY

The Temporal Query Phrase Popularity script (script2-local.pig or script2-hadoop.pig) processes a search query log file from the Excite search engine and compares the occurrence of frequency of search phrases across two time periods separated by twelve hours.

4.1.2 PIG LATIN STATEMENTS

A Pig Latin statement is an operator that takes a relation as input and produces another relation as output. Pig Latin statements can span multiple lines and must end with a semi-colon (;). Pig Latin statements are generally organized in the following manner:

A LOAD statement reads data from the file system.

A series of "transformation" statements process the data.

A STORE statement writes output to the file system; or, a DUMP statement displays output to the screen.

4.1.3 RUNNING PIG LATIN[7]

You can execute Pig Latin statements:

Using grunt shell or command line

- In mapreduce mode or local mode
- Either interactively or in batch

4.1.4 WORKING WITH DATA[7]

Pig Latin allows you to work with data in many ways. In general, and as a starting point:

Use the FILTER operator to work with tuples or rows of data. Use the FOREACH operator to work with columns of data.

Use the GROUP operator to group data in a single relation. Use the COGROUP and JOIN operators to group or join data in two or more relations.

Use the UNION operator to merge the contents of two or more relations. Use the SPLIT operator to partition the contents of a relation into multiple relations.

4.2 MEMORY MANAGEMENT[7]

Pig allocates a fix amount of memory to store bags and spills to disk as soon as the memory limit is reached. This is very similar to how Hadoop decides when to spill data accumulated by the combiner.

The amount of memory allocated to bags is determined by pig.cachedbag.memusage; the default is set to 10% of available memory. This memory is shared across all large bags used by the application.

4.3 ERROR HANDLING[7]

With multi-query execution Pig processes an entire script or a batch of statements at once. By default Pig tries to run all the jobs that result from that, regardless of whether some jobs fail during execution. To check which jobs have succeeded or failed use one of these options.

First, Pig logs all successful and failed store commands. Store commands are identified by output path. At the end of execution a summary line indicates success, partial failure or failure of all store commands.

Second, Pig returns different code upon completion for these scenarios:

1. Return code 0: All jobs succeeded

2. Return code 1: Used for retrievable errors
3. Return code 2: All jobs have failed
4. Return code 3: Some jobs have failed

In some cases it might be desirable to fail the entire script upon detecting the first failed job. This can be achieved with the "-F" or "-stop_on_failure" command line flag. If used, Pig will stop execution when the first failed job is detected and discontinue further processing. This also means that file commands that come after a failed store in the script will not be executed (this can be used to create "done" files).

4.4 DATA TYPES AND MORE[8]

Piglatin statements work with relations. A relation can be defined as follows

A relation is a bag (more specifically, an outer bag).

A bag is a collection of tuples.

A tuple is an ordered set of fields.

A field is a piece of data.

A Pig relation is a bag of tuples. A Pig relation is similar to a table in a relational database, where the tuples in the bag correspond to the rows in a table. Unlike a relational table, however, Pig relations don't require that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Also relations are unordered which means there is no guarantee that tuples are processed in any particular order. Furthermore, processing may be parallelized in which case tuples are not processed according to any total ordering.

4.5 SCHEMAS[8]

Schemas enable us to assign names to and declare types for fields. Schemas are optional, however type declarations result in better parse-time error checking and more efficient code execution.

Schemas are defined using the AS keyword with the LOAD, STREAM, and FOREACH operators. If you define a schema using the LOAD operator, then it is the load function that enforces the schema.

4.6 PIG UDFS

Pig provides extensive support for user defined functions (UDFs) as a way to specify custom processing. Pig UDFs can currently be implemented in three languages: Java, Python, JavaScript, Ruby and Groovy.

The most extensive support is provided for Java functions. You can customize all parts of the processing including data load/store, column transformation, and aggregation. Java functions are also more efficient because they are implemented in the same language as Pig and because additional interfaces are supported such as the algebraic interface and the Accumulator Interface.

Limited support is provided for Python, JavaScript, Ruby and Groovy functions. These functions are new, still evolving, additions to the system. Currently only the basic interface is supported; load/store functions are not supported. Furthermore, JavaScript, Ruby and Groovy are provided as experimental features because they did not go through the same amount of testing as Java or Python. At runtime note that Pig will automatically detect the usage of a scripting UDF in the Pig script and will automatically ship the corresponding scripting jar, either Jython, Rhino, JRuby or Groovy-all, to the backend.

Pig also provides support for Piggy Bank, a repository for JAVA UDFs. Through Piggy Bank you can access Java UDFs written by other users and contribute Java UDFs that you have written.

5.1. How Pig differs from MapReduce

Pig provides users with several advantages over using MapReduce directly. Pig Latin provides all of the standard data-processing operations, such as join, filter, group by, order by, union, etc. MapReduce provides the group by operation directly (that is what the shuffle plus reduce phases are), and it provides the order by operation indirectly through the way it implements the grouping. Filter and projection can be implemented trivially in the map phase. But other operators, particularly join, are not provided and must instead be written by the user.

Pig provides some complex, nontrivial implementations of these standard data operations. For example, because the number of records per key in a dataset is rarely evenly distributed, the data sent to the reducers is often skewed. That is, one reducer will get 10 or more times the data than other reducers. Pig has join and order by operators that will handle this case and (in some cases) rebalance the reducers. But these took the Pig team months to write, and rewriting these in MapReduce would be time consuming.

In MapReduce, the data processing inside the map and reduce phases is opaque to the system. This means that MapReduce has no opportunity to optimize or check the user's code. Pig, on the other hand, can analyze a Pig Latin script and understand the data flow that the user is describing. That means it can do early error checking and optimizations.

MapReduce does not have a type system. This is intentional, and it gives users the flexibility to use their own data types and serialization frameworks. But the downside is that this further limits the system's ability to check users' code for errors both before and during runtime.

All of these points mean that Pig Latin is much lower cost to write and maintain than Java code for MapReduce.

5.1.1Difference between Pig and Hive

PIG	HIVE
1)pig has a procedural data flow language.(Pig latin)	1)Hive has a declarative SQLish language(Hiveql)
2) Pig is mainly used for programming.	2) hive is used for creating reports
3)pig is generally used by researchers and programmers	3)hive is mainly used by data analyst.
4)pig operates on the client side of any cluster	4) hive operates on the server side of any cluster.
5) Pig does not have a metadata database and the schemas or data types will be defined in the script itself.	5)hive makes use of exact variation of the SQL DLL language by defining the tables beforehand and storing the schema details in any local database.
6)Pig supports avro.	6)hive does not support avro.
7)Pig is also SQL-like but varies to a great extent and thus it will take some time efforts to master pig.	7)hive directly leverages SQL expertise and thus can be learnt easily.

TABLE1:Pig and Hive

6.1. USING APACHE PIG TO PROCESS NETWORK FLOW DATA [1]

Apache Pig is a platform that supports substantial parallelization which enables the processing of huge data sets. The structure of Pig can be described in two layers – An infrastructure layer with an in-built compiler that can produce MapReduce programs and a language layer with a text-processing language called "Pig Latin". Pig makes data analysis and programming easier and understandable to novices in Hadoop framework using Pig Latin which can be considered as a parallel data flow language. The pig setup has also got provisions for further optimizations and user defined functionalities. Pig Latin queries are compiled into MapReduce jobs and are executed in distributed Hadoop cluster environments. Pig Latin looks similar to SQL[Structured Query Language] but is designed for Hadoop data processing environment just like SQL for RDBMS environment.[10] Another member of Hadoop ecosystem known as Hive is a query language like SQL and needs data to be loaded into tables before processing. Pig Latin can work on schema-less or inconsistent environments and can operate on the available data as soon as it is loaded into the HDFS. Pig closely resembles scripting languages like Perl and Python in certain aspects such as the flexibility in syntax and dynamic variables definitions. Pig Latin is efficient enough to be considered as the native parallel processing language for distributed systems such as Hadoop.

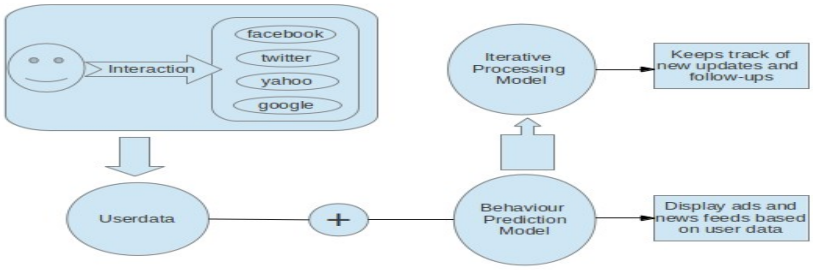

Figure 2. Data processing model in PIG[1]

Pig is gaining popularity in different zones of computational and statistical departments. It is widely used in processing weblogs and wiping off corrupted data from records. Pig can build behavior prediction models based on the user interactions with a website and this feature can be applied in displaying ads and news stories that keeps the user entertained. It also accentuates on an iterative processing model which can keep track of every new updates by combining the behavioral model with the user data. This feature is largely used in social networking sites like Facebook and micro-blogging sites like Twitter. When it comes to small data or scanning multiple records in random order, pig cannot be considered as effective as MapReduce. The aforementioned data processing model used by Pig is depicted in Figure 1.

6.1.1 IMPLEMENTATION ASPECTS[1]

Here we present the NetFlow analysis using Hadoop, which can manage large volume of data, employ parallel processing and come up with required output in no time. A map-reduce algorithm need to be deployed using Hadoop to get the result and writing such map-reduce programs for analyzing huge flow data is a time consuming task. Here the Apache-Pig will help us to save a good deal of time.

We need to analyze traffic at a particular time and Pig plays an important role in this regard. For data analysis, we create buckets with pairs of (SrcIF, SrcIPaddress), (DstIF,DstIPaddress), (Protocal,FlowperSec). Later, they are joined and categorized as SrcIF and FlowperSec; SrcIPaddress and FlowperSec. Netflow data is collected from a busy Cisco switch using nfdump. The output contains three basic sections-IP Packet section, Protocol Flow section and Source Destination packets section.

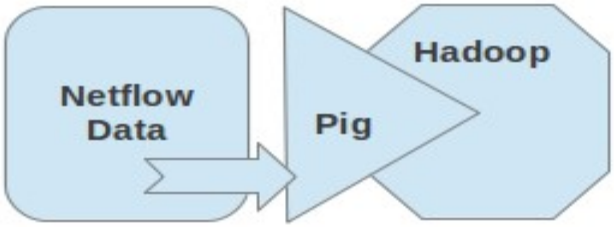

Figure 3. Netflow Analysis using Hadoop Pig[1]

Pig needs the netflow datasets in arranged format which is then converted to SQL-like schema. Hence the three sections are created as three different schemas. A sample Pig Latin code is given below:

Protocol = LOAD „NetFlow-Data1‟ AS (protocol:chararray, flow:int, …)

Source = LOAD „NetFlow-Data2‟ AS (SrcIF:chararray, SrcIPAddress:chararray, …)

Destination = LOAD „NetFlow-Data3‟ AS (DstIF:chararray, DstIPaddress, …)

As shown in Figure 2, the network flow is deduced into schemas by Pig. Then these schemas are combined for analysis and the traffic flow per node is the resultant output**.**

7.1.2. ADVANTAGES OF USING HADOOP PIG OVER MAPREDUCE[1]

The member of Hadoop ecosystem popularly known as Pig can be considered as a procedural version of SQL. The most attractive feature and advantage of Pig is its simplicity. It doesn't demand highly skilled Java professionals for implementation. The flexibility in syntax and dynamic variable allocation makes it much more user-friendly and provisions for user-defined functions add extensibility for the framework. The ability of Pig to work with un-structured data greatly accounts to parallel data processing. The great difference comes when the computational complexity is considered. Typical map-reduce programs will have large number of java code lines whereas the task can be accomplished using Pig Latin in very fewer lines of code. Huge network flow data input files can be processed with writing simple programs in Pig Latin which closely resembles scripting languages like Perl/Python.

7.2. PIG APPLICATIONS

Pig Latin use tend to fall into three separate categories: traditional extract transform load (ETL) data pipelines, research on raw data, and iterative processing[6].

The largest use case is data pipelines. A common example is web companies bringing in logs from their web servers, cleansing the data, and pre-computing common aggregates before loading it into their data warehouse[6]. In this case, the data is loaded onto the grid, and then Pig is used to clean out records from bots and records with corrupt data. It is also used to join web event data against user databases so that user cookies can be connected with known user information[6]. It is also used for data mining such as digging interesting kinds of pattern(useful data) which would be useful in future use[6].

Another example of data pipelines is using Pig offline to build behavior prediction models. Pig is used to scan through all the user interactions with a website and split the users into various segments[6]. Then, for each segment, a mathematical model is produced that predicts how members of that segment will respond to types of advertisements or news articles. In this way the website can show ads that are more likely to get clicked on, or offer news stories that are more likely to engage users and keep them coming back to the site[6].

Traditionally, ad-hoc queries are done in languages such as SQL that make it easy to quickly form a question for the data to answer. However, for research on raw data, some users prefer Pig Latin. Because Pig can operate in situations where the schema is unknown, incomplete, or inconsistent, and because it can easily manage nested data, researchers who want to work on data before it has been cleaned and loaded into the warehouse often prefer Pig[6]. Researchers who work with large data sets often use scripting languages such as Perl or Python to do their processing. Users with these backgrounds often prefer the dataflow paradigm of Pig over the declarative query paradigm of SQL[6].

Users building iterative processing models are also starting to use Pig. Consider a news website that keeps a graph of all news stories on the Web that it is tracking[6]. In this graph each news story is a node, and edges indicate relationships between the stories. For example, all stories about an upcoming election are linked together. Every five minutes a new set of stories comes in, and the data-processing engine must integrate them into the graph. Some of these stories are new, some are updates of existing stories, and some supersede existing stories[6]. Some data-processing steps need to operate on this entire graph of stories. For example, a process that builds a behavioral targeting model needs to join user data against this entire graph of stories. Rerunning the entire join every five minutes is not feasible because it cannot be completed in five minutes with a reasonable amount of hardware. But the model builders do not want to update these models only on a daily basis, as that means an entire day of missed serving opportunities[6].

To cope with this problem, it is possible to first do a join against the entire graph on a regular basis, for example, daily[6]. Then, as new data comes in every five minutes, a join can be done with just the new incoming data, and these results can be combined with the results of the join against the whole graph[6]. This combination step takes some care, as the five-minute data contains the equivalent of inserts, updates, and deletes on the entire graph. It is possible and reasonably convenient to express this combination in Pig Latin[6].

One point that is implicit in everything I have said so far is that Pig (like MapReduce) is oriented around the batch processing of data[6]. If you need to process gigabytes or terabytes of data, Pig is a good choice. But it expects to read all the records of a file and write all of its output sequentially. For workloads that require writing single or small groups of records, or looking up many different records in random order, Pig (like MapReduce) is not a good choice[6].

III. CONCLUSION

In this paper, we have accomplished that pig is a framework for the execution of complex queries to analyze data. Its built on hadoop and takes advantage of the distributed nature and implementation of map-reduce. Pig is a two part of the ecosystem, the actual language(Pig) and the execution environment(where the programmer enter the logic). It ease programming for hadoop by allowing cursory development. Pig is designed for batch processing of data. Pig's infrastructure layer consists of a compiler that turns (relatively short) Pig Latin programs into sequences of MapReduce programs. Pig is a Java client-side application, and users install locally – nothing is altered on the Hadoop cluster itself. Grunt is the Pig interactive shell. It adds a layer of abstraction on top of Hadoop to simplify its use by giving a SQL-like interface to process data on Hadoop and thus help the programmer focus on business logic and help increase productivity. It supports a variety of data types and the use of user-defined functions (UDFs) to write custom operations in Java, Python and JavaScript. Due its simple interface, support for doing complex operations such as joins and filters, Pig is popular for performing query operations in hadoop. Pig could be used for ETL(Extraction Transformation Load) tasks naturally as it can handle unstructured data. It bolsters a lot of clever optimizations like multiquery execution which makes our complex queries executes faster. however, pig has also problems dealing with unstructures data such as images, text, audio that is ambiguously delimited, log data etc. And it cannot deal with proper design of XML OR JASON and flexible schemas. Its process is also time consuming as compared to map-reduce. Hence, work should be done to overcome these problems.

IV. REFERENCES

[1] Anjali, binu, "NETWORK TRAFFIC ANALYSIS: HADOOP PIG VS TYPICAL MAPREDUCE".
[2] Alan f. gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh shrivastava, "building a high-level dataflow system on top of map- reduce: the pig experience."
[3] sanjeev dhwan, sanjay rathee, "big data analytics using hadoop components like pig and hive, American international journal of research in science, technology engineering & mathematics.
[4] big data hands on- workshop, "data manipulation with hive and pig".
[5] Hortonworks, " tutorial on apache pig".

[6] alan gates, “programming pig”.
[7] the apache software foundation “pig latin reference manual 1”.
[8] the apache software foundation “pig latin reference manual 2”.
[9] (hadoop) pig dataflow language, “b. Ramamurthy”, based on cloudera’s tutorials and apache pig manual.
[10] Sherif Sakr, Anna Liu, Ayman G. Fayoumi, "The Family of MapReduce and Large Scale Data Processing Systems" published in arXiv:1302.2966v1 [cs.DB] 13 Feb 2013
[11] http://www.stratapps.net/intro-pig.php