Apache Pig – A Data Flow Framework Based on Hadoop Map Reduce

Article · August 2017

DOI: 10.14445/22315381/IJETT-V50P244

2 authors:

Cdsa Swa
Jefferson College of Health Science
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Zahid Ansari
P.A. College of Engineering
30 PUBLICATIONS **202** CITATIONS
SEE PROFILE

Apache Pig - A Data Flow Framework Based on Hadoop Map Reduce

Swarna C[#1], Zahid Ansari[*2]

[#]*Department of Computer Science and Engineering, P.A. College of Engineering, Mangaluru, India*
[*]*Department of Computer Science and Engineering, P.A. College of Engineering, Mangaluru, India*

Abstract *— Big Data is a technology phenomenon happened due to the increased rate of data growth, complex new data types and parallel advancements in technology stake. Big data can be structured, unstructured or semi-structured, resulting in ineffectiveness of conventional data management methods. Hadoop is a framework for the analysis and transformation of very large data sets using the Map Reduce paradigm. An important characteristic of Hadoop is the splitting of data and computation across thousands of hosts and running applications in parallel close to their data. Hadoop accomplish this by HDFS and Map Reduce. Pig is an apache open source project. It runs on Hadoop by making use of both HDFS and Map Reduce. There are two main components for Pig. First component Pig Latin is the parallel dataflow language which is designed in such a way to fit between the SQL and the Map Reduce. Pig Latin enables the use to define the reading, processing, storing the data in parallel. Pig Latin script explicates a directed acyclic graph, where data flows are represented as edges and operators are represented as nodes. The second component is the run time environment in which Pig Latin programs are executed.*

Keywords — *Big Data, Hadoop, Map Reduce, Pig, Pig Latin.*

I. Introduction

The term 'Big Data' describes inventive techniques and technologies to capture, store, distribute, manage and analyse petabyte or larger-sized datasets with high-velocity and different structures [1]. Hadoop is open-source software that enables reliable, scalable, distributed computing on clusters of less expensive servers [2]. In 2004 Google has invented a frame work called Map Reduce which is mainly used for parallel data processing in a distributed computing environment. But the Map Reduce is too low level and rigid. it has many drawbacks like writing low level Map Reduce code is slow, need a lot of expertise to optimize Map Reduce code, prototyping is slow, a lot of custom code required even for simple tasks and it is hard to manage more complex map reduce job chains. So a new language called Pig Latin was developed which is a high level declarative query language like SQL and a low level procedural programming like Map Reduce.

Pig Latin is implemented on Pig which is open source software which run on Hadoop. Pig Latin's main features include support for an adaptable nested data model, extensive support for user defined functions, and the ability to operate on input files without any schema information. Pig Latin also comes with a novel debugging environment that is particularly useful when dealing with massive data sets.

II. Pig Components And Architecture

Pig is an apache open source project .It is an engine for executing parallel data flows on Hadoop. It runs on Hadoop by making use of both HDFS and Map Reduce which are the two components of Hadoop [3]. Pig was initially developed at Yahoo. The Pig programming language is designed to handle any type of data that is reasonable. Pig is made up of two components: the first component is the Pig Latin-which is the language, and the second is the runtime environment where Pig Latin programs are executed [4]. Fig. 1 shows the components of Pig.

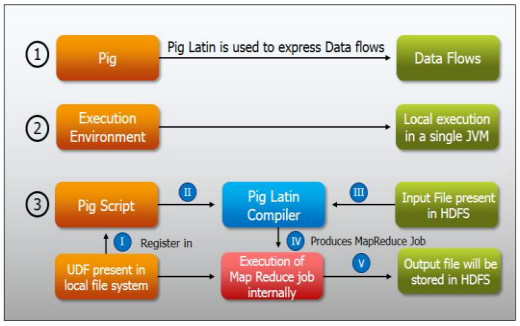

Fig 1: Components of Pig

Figure 1 also describes the various steps during the execution. The data is loaded from HDFS and it is then converted to many map and reduce tasks. Lastly the output is either stored in to a file or dumped to screen.

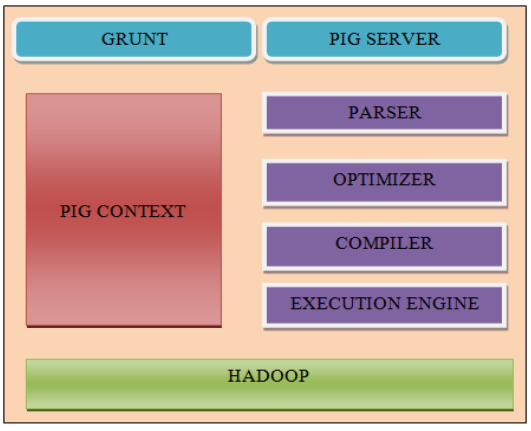

Fig 2: Pig Architecture

Fig. 2 describes Pig Architecture. Grunt is the interactive shell for the users to enter Pig Latin. Parser converts Pig Latin in to Logical Plan, which is further optimized by the optimizer. Compiler converts it in to a series of map reduce jobs. These jobs are executed by the execution engine. Pig allows three modes of user interaction [7]:

- *Interactive mode:* Here, the user is entering Pig Commands with an interactive shell which is known as Grunt. When the user asks for output through the STORE command plan compilation and execution is triggered.
- *Batch mode:* In this mode, a user submits a prewritten script containing a group of Pig commands, typically finishing with STORE. The semantics are identical to interactive mode.
- *Embedded mode:* Pig Latin Commands can be submitted through method invocation from a java program. For this a Java library is provided by Pig. Through this option dynamic construction of Pig Latin programs and dynamic control flow can be achieved. e.g. looping for a non-predetermined number of iterations, which is not currently supported in Pig Latin directly.

III. PIG LATIN

Through this section the details of Pig Latin language is described. We describe Pig data model in Section A, and the Pig Latin statements in the subsequent subsections. Pig Latin has the following key properties [15]:

- *Ease of programming*: 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 tasks are encoded to permit the system to automatically optimize their execution. It allows the user to focus on semantics rather than efficiency.
- *Extensibility:* Users can create their own functions to do special-purpose processing.

A. Data Model

Data in Pig Latin is categorized into two types [16]. Scalar and complex data type. Pig's scalar types are similar to the data types that appear in most programming languages. With the exception of bytearray, they are all represented in Pig interfaces by java.lang classes, making them easy to work with in UDFs: Table I describes the scalar data types .

TABLE I
SCALAR DATA TYPES

Scalar Data type	Description	Example
int	Four-byte signed integer.	12
long	Eight-byte signed integer.	80000L
float	Four byte floating-point number.	6.2f or 6.2e2f
double	Eight byte floating point.	2.718 or 6.626e-34.
chararray	A string or character array.	Hello
bytearray	A blob or array of bytes.	.

Pig's three complex data types are: maps, tuples, and bags. All of these types can contain data of any type, including other complex types. So it is possible to have a map where the value field is a bag, which contains a tuple where one of the fields is a map. Table II describes complex data types.

TABLE II
COMPLEX DATA TYPES

Complex Data type	**Description**	**Example**
tuple	An ordered set of fields	(1,`alice‘)
bag	A collection of tuples	{ (1,`alice’),(2)}
map	A map is a collection of data items, where each item has an associated key.	[‘a’#’pomegranate’]

TABLE III
DIAGNOSTIC OPERATORS

Operator	Description
Describe	Returns the schema of the relation
Dump	Dumps the results to the screen
Explain	Displays execution plans.
Illustrate	Displays a step-by-step execution of a sequence of statements

B. Pig diagnostic operators

Pig Latin provides four different types of diagnostic operators: Describe Dump, Explain and Illustrate. Describe, Explain and Illustrate are provided to allow the operator to work together with the logical plan, for debugging purposes. The Dump is a sort of diagnostic operator too because it is used only to permit interactive debugging of small result sets or in combination with Limit. Table III will give a brief description about these operators.

TABLE IV
PIG COMMANDS

Command	Description
Load	Read data from the file system
Store	Write data to the file system
Dump	Write output to stdout
Foreach Generate	Apply expression to each record and generate one or more records
Filter	Apply predicate to each record and remove records where false
Group / Cogroup	Collect records with the same key from one or more inputs
Join	Join two or more inputs based on a key
Order	Sort records based on a Key
Distinct	Remove duplicate records
Union	Merge two datasets
Limit	Limit the number of records
Split	Split data into 2 or more sets, based on filter conditions
Cross	Creates the cross product of two or more relations

C. Pig Commands

Apache pig present different built in data processing operators. For input/output processing Load or Store commands are used. For filtering data, For each, Generate and Stream commands are used. There are also commands for grouping and joining data. Important data processing commands are described in Table IV.

IV. IMPLEMENTATION

Pig Latin is fully implemented by the system, Pig. Pig's architecture allows different systems to be plugged in as the execution platform for Pig Latin. Our current implementation uses Hadoop, an open-source, scalable implementation of map-reduce [2], as the execution platform. Pig Latin programs are compiled into map-reduce jobs, and executed using Hadoop. Pig, together with its Hadoop compiler, is an open-source project implemented by Apache and it is available for general use [11].

A. Building a Logical Plan

As clients issue Pig Latin commands, the Pig interpreter first parses it, and verifies that the input files and bags referenced by the command are valid. For example, if the user enters p = COGROUP q BY:: :, r BY :: :, Pig verifies whether the bags q and r have already been defined. Pig builds a logical plan for the bags q and r user defines. When a new bag is defined by a command, the logical plan for the new bag is constructed by combining the logical plans for the input bags, and the current command.

Thus, in the above example, the logical plan for p consists of a cogroup command having the logical plans for q and r as inputs. During the construction of logical plan processing is not carried out. Processing is activated when the user invokes a STORE command on a bag. While processing is activated, the logical plan for that bag is compiled into a physical plan, and is executed. This is illustrated in figure 3.This lazy style of execution is beneficial because it permits in-memory pipelining, and other optimizations such as filter reordering across multiple Pig Latin commands.

Pig is designed in such a way that the parsing of Pig Latin and the logical plan construction is independent of the execution platform. The compilation of the logical plan into a physical plan depends on which specific execution platform is chosen. Next, we describe the compilation into Hadoop map-reduce, the execution platform currently used by Pig.

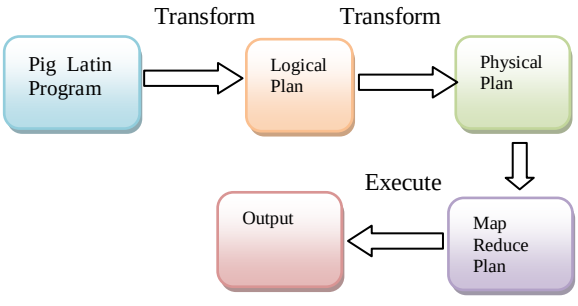

Figure 3: Pig Latin Workflow

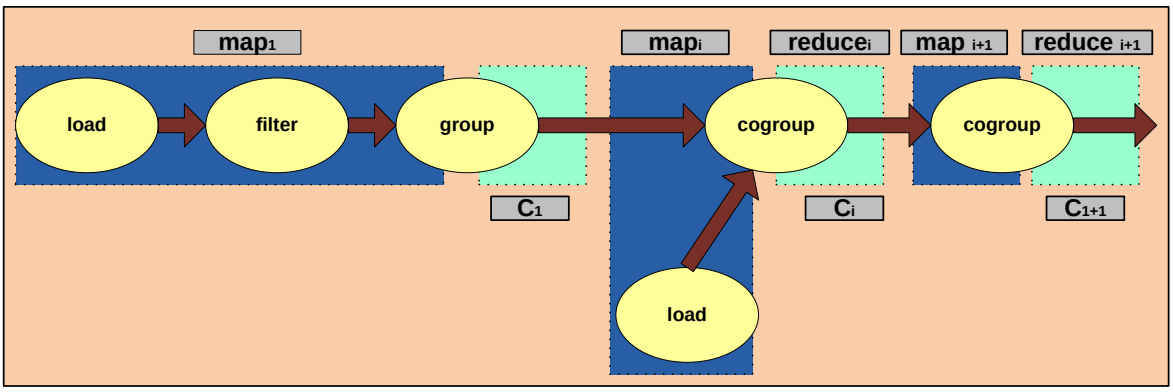

Fig 4: Pig Latin Map Reduce Compilation

B. Map-Reduce Plan Compilation

Compilation of a Pig Latin logical plan into map-reduce jobs is straight forward. The map-reduce task fundamentally provides the capacity to do a large-scale group by, where the map tasks assign keys for grouping, and the reduce tasks process a group at a time. Pig compiler begins by converting each (CO)GROUP command in the logical plan into a distinct map-reduce job with its own map and reduce functions. The map function for (CO)GROUP command p first assigns keys to tuples based on the BY clause(s) of p. The reduce function has no operation initially. The map-reduce boundary is the cogroup command. The sequence of FILTER, and FOREACH commands from the LOAD to the first COGROUP operation C_1, are pushed into the map function corresponding to C_1 (see Figure 4). The commands that lies between subsequent COGROUP commands C_i and C_{i+1} can be pushed into either

- The reduce function corresponding to C_i, or
- The map function corresponding to C_{i+1}.

Pig currently always follows option (a). Since grouping is often followed by aggregation, this approach reduces the amount of data that has to be materialized between map reduce jobs.

If the COGROUP command consists of more than one input data set, the map function appends an extra field to each tuple that indicates the data set from which the tuple originated. The corresponding reduce function decodes this information. The decoded information is used to insert the tuple into the appropriate nested bag when cogrouped tuples are generated.

Parallelism for LOAD is obtained as Pig operates over files which reside in the Hadoop distributed file system. Parallelism for FILTER and FOREACH operations are also achieved since for a given map-reduce job, several map and reduce instances are running in parallel. We also get Parallelism for (CO)GROUP since the output from the multiple map instances is repartitioned in parallel to the multiple reduce instances.

While implementing the ORDER command is two map-reduce jobs are compiled. The first job samples the input to find out quantiles of the sort key. The second job range partitions the input according to the quantiles, which follows a local sort in the reduce phase, finally resulting in a globally sorted file.

The inflexibility of the map-reduce primitive causes some overheads while compiling Pig Latin into map-reduce jobs. For example, data must be materialized and replicated on the distributed file system between successive map-reduce jobs. While dealing with multiple data sets, an additional field must be added in every tuple to indicate the origin of data set. Since the Hadoop map-reduce implementation does provide many desired properties such as parallelism, load balancing, and fault-tolerance, the associated overhead is often acceptable.

V. APPLICATIONS

Some of the important uses of Pig are described below:

- Pig is a powerful tool for querying data in a Hadoop cluster. It's so powerful that Yahoo estimates that between 40% and 60% of its Hadoop workloads are generated from Pig Latin scripts.[23]
- Pig is also used at Twitter (processing logs, mining tweet data); at AOL and MapQuest (for analytics and batch data processing); and at LinkedIn, where Pig is used to discover people you might know.[23]
- With continually increasing population, crimes and crime rate analyzing related data is a huge issue for governments to make strategic decisions so as to maintain law and order. The benefit of using Pig for analysis is that fewer lines of code have to be written which reduces overall development and testing time [20].
- Using pig script a large scale data processing system for analyzing web log data through Map Reduce programming in Hadoop framework is efficient[21].

- Pig is used to evaluate the performance of a commercial RDBMS and Hadoop in astronomy simulation analysis tasks [22].

V. CONCLUSIONS

This paper introduced the concept of Pig and its associated language Pig Latin which is a new data processing environment deployed at Yahoo. We have entered an era of Big Data and Hadoop is a framework for the analysis and transformation of this Big data using the Map Reduce paradigm. The Pig system compiles Pig Latin expressions into a sequence of map-reduce jobs, and orchestrates the execution of these jobs on Hadoop. Pig structure is susceptible to substantial parallelization.

REFERENCES

[1] Bhosale, Harshawardhan S., and Devendra P. Gadekar. "A Review Paper on Big Data and Hadoop." *International Journal of Scientific and Research Publications* 4.10 (2014):

[2] Chavan, Ms Vibhavari, and Rajesh N. Phursule. "Survey paper on big data." *Int. J. Comput. Sci. Inf. Technol* 5.6 (2014): 7932-7939.

[3] Samak, Taghrid, Daniel Gunter, and Valerie Hendrix. "Scalable analysis of network measurements with Hadoop and Pig." *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012.

[4] Goyal, Vikas, and Deepak Soni. "SURVEY PAPER ON BIG DATA ANALYTICS USING HADOOP TECHNOLOGIES."

[5] Wang, MingXue, Sidath B. Handurukande, and Mohamed Nassar. "RPig: A scalable framework for machine learning and advanced statistical functionalities." *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012.

[6] Ouaknine, Keren, Michael Carey, and Scott Kirkpatrick. "The PigMix Benchmark on Pig, MapReduce, and HPCC Systems." *Big Data (BigData congress), 2015 IEEE International Congress on*. IEEE, 2015.

[7] Samak, Taghrid, Daniel Gunter, and Valerie Hendrix. "Scalable analysis of network measurements with Hadoop and Pig." *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012.

[8] Gates, Alan F., et al. "Building a high-level dataflow system on top of Map-Reduce: the Pig experience." *Proceedings of the VLDB Endowment* 2.2 (2009): 1414-1425.

[9] Adnan, Muhammad, et al. "Minimizing big data problems using cloud computing based on Hadoop architecture." *High-capacity Optical Networks and Emerging/Enabling Technologies (HONET), 2014 11th Annual*. IEEE, 2014.

[10] Shang, Weiyi, Bram Adams, and Ahmed E. Hassan. "Using Pig as a data preparation language for large-scale mining software repositories studies: An experience report." *Journal of Systems and Software* 85.10 (2012): 2195-2204.

[11] Shvachko, Konstantin, et al. "The hadoop distributed file system." *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. IEEE, 2010.

[12] Olston, Christopher, et al. "Pig latin: a not-so-foreign language for data processing." *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008.

[13] Shvachko, Konstantin, et al. "The hadoop distributed file system." *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. IEEE, 2010.

[14] Wang, Yaoguang, et al. "Improving MapReduce performance with partial speculative execution." *Journal of Grid Computing* 13.4 (2015): 587-604.

[15] Agarwal, Shafali, and Zeba Khanam. "Map Reduce: A Survey Paper on Recent Expansion." *International Journal of Advanced Computer Science and Applications* 6.8 (2015): 209-215.

[16] Olshannikova, Ekaterina, et al. "Conceptualizing Big Social Data." *Journal of Big Data* 4.1 (2017): 3.

[17] Tom White foreword by Doug Cutting; ―Hadoop: The Definitive Guidel; ISBN: 978-1-449-38973-4 [SB] 1285179414.

[18] Bhardwaj, Vibha, Rahul Johari, and Priti Bhardwaj. "Query execution evaluation in wireless network using MyHadoop." *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*. IEEE, 2015.

[19] Tanimura, Yusuke, et al. "Extensions to the Pig data processing platform for scalable RDF data processing using Hadoop." *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 2010.

[20] Arushi Jaina, Vishal Bhatnagara Ambedkar" Crime Data Analysis Using Pig with Hadoop*", International Conference on Information Security &Privacy (ICISP2015)*, 11-12 December 2015

[21] Prasad, PS Durga, T. Vivekanandan, and A. Srinivasan. "A Methodology for WebLog Data analysis using HadoopMapReduce and PIG." *i-manager's Journal on Cloud Computing* 3.1 (2015): 13.

[22] Loebman, Sarah, et al. "Analyzing massive astrophysical datasets: Can Pig/Hadoop or a relational DBMS help?." *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009.

[23] www.wikepedia.org 12/04/2017 at 8:30 pm