www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 4 April 2017, Page No. 21154-21157
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i4.62

Flexible Data Processing Using Apache PIG

k. vikram [1], J. Sredevi[2], J. Sindhu[3], J. Shravya[4], K. Akrutha[5]

Assistant Professor, Computer Science & Engineering, Guru Nanak Institution Technical Campus, Hyderabad, India[1]

B.Tech Student, Computer Science & Engineering, Guru Nanak Institution Technical Campus, Hyderabad, India[2,3,4,5]

Abstract: The World Wide Web has an estimated 2 billion users and contains anywhere from 15 to 45 billion web pages , with around 10 million pages added each day with such large numbers , almost every website owner and developer who has a decent presence on the internet faces a complex problem : how to make sense of their web pages and all the users who visit their websites. Every web server worth its salt logs the user activities for the websites it supports and the web pages it serves up to the virtual world. These web logs are mostly used for debugging issues or to get insight in to the details , which are interesting from a business or performance point of view. Overtime , the size of the logs keeps increasing until it becomes very difficult to manually extract any important information out of them , particularly for busy websites. The Hadoop framework does a good job at tackling this challenge in a timely , reliable and cost-efficient manner. Web log analysis using the Hadoop framework and pig scripting language , which are well suited to handle large amounts of unstructured data. We propose a solution based on the pig framework that aggregates data at an hourly , daily or yearly granularity..

Index Terms: Apache Pig, Big Data, Hadoop, HDFS

I. INTRODUCTION

For the past two decades most business analytics have been created using structured data extracted from operational systems and consolidated into a data warehouse. Big data dramatically increases both the number of data sources and the variety and volume of data that is useful for analysis. A high percentage of this data is often described as multi-structured to distinguish it from the structured operational data used to populate a data warehouse. In most organizations, multi-structured data is growing at a considerably faster rate than structured data. Two important data management trends for processing big data are relational DBMS products optimized for analytical workloads (often called analytic RDBMSs, or ADBMSs) and non-relational systems processing for multi-structured data.

Hadoop is an open-source data processing framework that includes a scalable, fault-tolerant distributed file system, HDFS. Although HDFS was designed to work in conjunction with Hadoop's job scheduler, we have re-purposed it to serve as a grid storage element by adding GridFTP and SRM servers. We have tested the system thoroughly in order to understand its scalability and fault tolerance. The turn-on of the Large Hadron Collider (LHC) in 2009 poses a significant data management and storage challenge; we have been working to introduce HDFS as a solution for data storage for one LHC experiment, the Compact Muon Solenoid (CMS).

The High Performance Computing (HPC) and MapReduce have been doing large-scale and Parallel data processing for years, using such APIs as Message Passing Interface. Broadly, the approach in HPC is to distribute the work across a cluster of machines, which access a shared filesystem, hosted by a SAN. This works well for predominantly compute intensive jobs, but becomes a problem when nodes need to access larger data volumes (hundreds of gigabytes, the point at which MapReduce really starts to shine), since the network bandwidth is the bottleneck and compute nodes become idle.

In order to solve the scalability problem, we proposed a system that exploits parallel database processing over the distributed file system and the MapReduce framework [2]. The system design was inspired by the recent achievements of Google and Yahoo for handling petabyte scale Web data on the commodity hardware clusters. The distributed file system proposed as Google File System (GFS) [3] provides functionality to store a large file over multiple storage nodes by dividing it to fixed-size chunks, with fault-tolerance to node crashesusing the chunk replica on other nodes. MapReduce [4] is a programming model to compose a parallel job by defining sub-tasks as an arbitrary map operation processing the chunk and a reduce operation merging the outputs of the maps, and also an efficient and fault-tolerant execution model allowing retries of the sub-tasks on a distributed environment running GFS. Hadoop [5] is based on GFS and MapReduce, and it is an open-source software aimed at providing a similar functionality. We use Hadoop as the basic infrastructure and also use Pig [6], which provides a general data processing platform on top of MapReduce and allows users to write a script incorporating database operations (e.g. filter, join) in a procedural programming style. Pig compiles the script and generates the MapReduce code to run on the Hadoop installed system. The scalability of this software stack is shown by other works [7], [6].

The main objective of this project is finding the business insights of current user records data. And get the benefits for business growth. The parameters to be considered for analysis are Daily user count and bytes transmitted on a particular time slot. Area wise business (usage) share in the total business. Since every market owner will be depending on partners to get the service where they does not have the service provider

II. EXISTING TECHNOLOGY

In the existing technology ,we have the datasets stored in the local file system and the data is transferred into the HDFS by using the Pig Latin commands. Here data is more sparse and unclear to analyse them in an efficient way. Therefore there is

a need to provide a solution to analyse the datasets easily. As data sets have explosively increased, there have been proposed big data platforms for processing huge data sets. Hadoop MapReduce platform is a representative data processing scheme for handling large data sets on distributed computing nodes. The Hadoop platform can support highly scalable distributed data processing capabilities, but it is difficult to program and easy to make serious errors because it supports a low-level interface. In order to solve these problems, the apache Pig platform has been proposed. Then apache Pig system provides a high-level interface language, called Pig Latin. Pig system enables users to generate data processing services with ease of development, high productivity by using the high-level data flow Pig Latin language. Pig system compiles Pig Latin programs, which are abstract data flow expressions, into one or more physical data flow jobs, and then orchestrates the execution of these jobs. And the compiled service jobs are executed on the MapReduce engine. The MapReduce platform is suitable for batch processing on large data sets.

III. Proposed Technology

In the proposed technology, there is large amount of datasets present in the Local file system is transferred into the HDFS and there the group of data, that is essential for data processing is dumped into the separate files and then we can query on the existing data using the Pig Latin Commands. Where the desired result are obtained without killing the users time. The open source platform for analyzing enormous data sets that consists of a high-level language for expressing data analysis programs, coupled with infra-structure for evaluating these programs. It supports the parallel programming model of MapReduce jobs to be executed on a Hadoop cluster system. The infrastructure layer of apache Pig is composed of a complier that generates sequences of MapReduce programs, for which large-scale parallel implementations already exist. The language layer in Pig system currently consist of simple scripting language, called Pig Latin. The Pig Latin is a procedural language that explicitly defines the data flow, so we can easily create the program for data processing . It supports optimization opportunities, so users can focus on semantics rather than efficiency. Also, users can create their own functions, user-defined functions (UDFs), to do special-purpose processing. Pig supports basic relational operators for processing large data sets. The Pig Latin scripts as an input source, Next, the Pig Latin programs will be compiled as one or more Map-Reduce jobs. There are several stages of compilation such as parsing, semantic checking, optimizations, and translators. Next, a translated MapReduce jobs jar is launched on.

Fig. 1 Evolution of the Big data and usage

IV. System Architecture

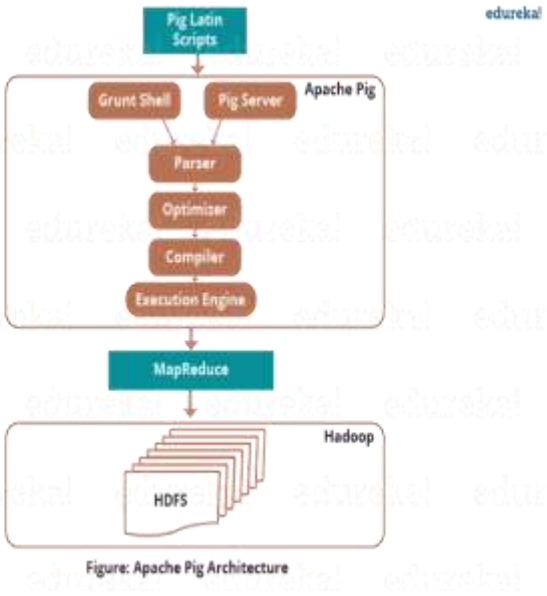

Fig. 2 System Architecture of Pig

The systems architect establishes the basic structure of the system, defining the essential core design features and elements that provide the framework for all that follows, and are the hardest to change later. The systems architect provides the architects view of the users' vision for what the system needs to be and do, and the paths along which it must be able to evolve, and strives to maintain the integrity of that vision as it evolves during detailed design and implementation. The system has three integrated faces; general data processing interface provided as Pig Latin standard commands, RDF data processing interface that we extend, and custom data processing interface given by the users as UDF. Our extension will be limited to the RDF data processing part. The storage schema is defined at the bottom layer and the optimization using the schema is implemented in Pig's query engine.

IV. Results

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Apache Pig.

To write data analysis programs, Pig provides a high-level language known as Pig Latin. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

DOI: 10.18535/ijecs/v6i4.62

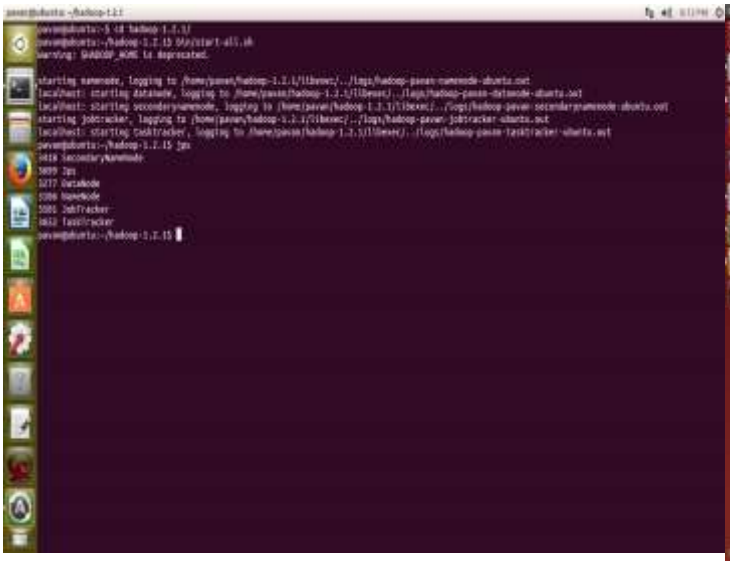

Fig. 3 Entry stage of the Hadoop

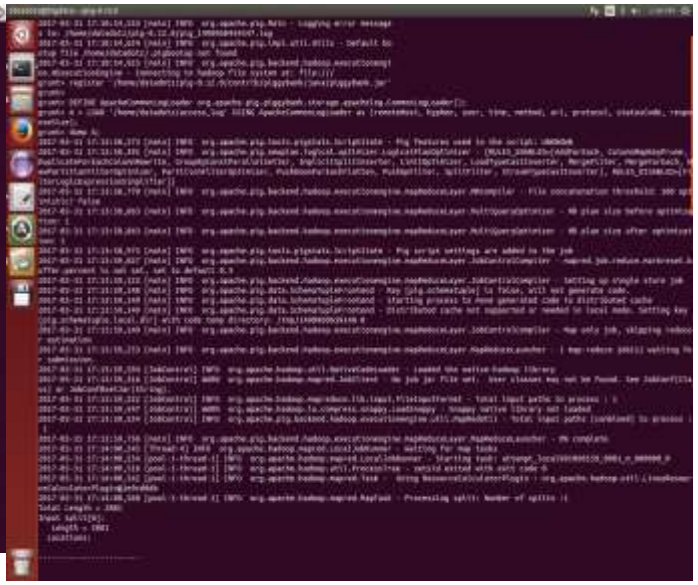

Fig. 6 Dumping the data from local to HDFS

Fig. 4 Hadoop local terminal

Fig. 7 Generating the desired result

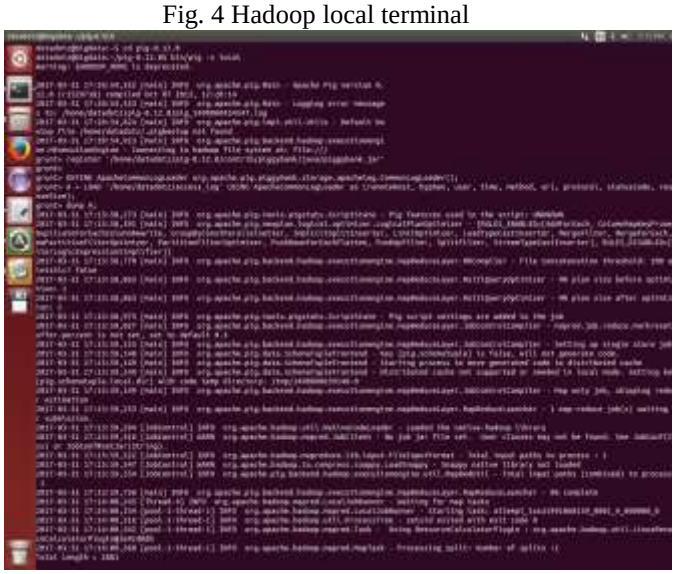

Fig. 5 Entering into the Pig Terminal

V. CONCLUSION

Analysts can talk about data insights all day (and night), but the reality is that 70% of all data analyst time goes into data processing and not analysis. At Sigmoid Analytics, we want to streamline this data processing pipeline so that analysts can truly focus on value generation and not data preparation. We focus our efforts on three simple initiatives. Make data processing more powerful. Make data processing more simple. Make data processing 100x faster than before Where ever you using HDFS (Hadoop Distributed File System) data and securely to access that data from anywhere. And also security is provided by using Kerberos technique. We propose a data flow language, which supports to deploy services that continuously process huge streams of data in real-time. To handle unbounded streams of data, we provide a data flow stream processing extended from Pig Latin. Finally, the generated stream processing service jobs are submitted and executed to process large streams of data in real-time on a highly scalable distributed stream processing system.

VI. REFERENCE

[1] Liming Lu, Mun Choon Chan, Ee-Chien Chang, "A General Model of Probabilistic Packet Marking for IP Traceback", ASIACCS '08, March 18-20, Tokyo, Japan.

[2] Chao Gong, Kamil Sarac, "IP Traceback based on Packet Marking and Logging", IEEE International Conference on Communication (ICC), May 16-20, 2005., Seoul, Korea.

[3] Turgay Korkmaz, Chao Gong, Kamil Sarac, Sandra G. Dykes, "Single packet IP traceback in AS-level partial deployment scenario", Int. J. Security and Networks, Vol. 2, Nos. 1/2, 2007

[4] Shui Yu, Wanlei Zhou, Song Guo, Minyi Guo, "A Feasible IP Traceback Framework through Dynamic Deterministic Packet Marking", DOI 10.1109/TC.2015.2439287, IEEE Transactions on Computers

[5] T. K. T. Law, J. C. S. Lui, and D. K. Y. Yau, "You can run, but you can't hide: An effective statistical methodology to trace back DDoS attackers," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 9, pp. 799–813, 2005.

[6] S. Yu, W. Zhou, S. Guo, and M. Guo, "A dynamical deterministic packet marking scheme for DDoS traceback," in IEEE International Conference on Global Communication, 2013.

[7] Darshan Lal Meena, Dr. R. S. Jadon, "Distributed Denial of Service Attacks and Their Suggested Defense Remedial Approaches ", Volume 2, Issue 4, April 2014

[8] World Wide Web Consortium (W3C) Recommendation, "Resource Description Framework (RDF)," http://www.w3.org/RDF/, 2004.

[9] Y. Tanimura, A. Matono, I. Kojima, and S. Sekiguchi, "Storage Scheme for Parallel RDF Database Processing Using Distributed File Ssytem and MapReduce," in *Proceedings of HPC Asia*, 2009, pp. 312–319.

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proceedings of the 19th ACM Symposium on Operating System Principles*, 2003.

[11] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proceedings of the 6th Symposium on Operating System Design and Implementation*, 2004.

[12] "Hadoop," http://hadoop.apache.org/.

[13] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-So-Foreign Language for Data Processing," in *Proceedings of ACM SIBMOD*, 2008, pp. 1099–1110.

[14] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience," in *Proceedings of VLDB*, 2009.

[15] O. O'Malley and A. C. Murthy, "Winning a 60 Second Dash with a Yellow Elephant," http://sortbenchmark.org/Yahoo2009.pdf, 2009.