Large-Scale Web Traffic Log Analyzer on Hadoop Distributed File System

Choopan Rattanapoka^{a,*}, Prasertsak Tiawongsombat^a

^aDepartment of Electronics Engineering Technology, College of Industrial Technology, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, Thailand

*Corresponding Author: choopan.r@cit.kmutnb.ac.th

Abstract

Large websites or web servers have a large number of visitors, which mean a large web traffic log need to be stored in the plain text or the relational database. However plain text and relational database are not efficient to handle a large number of data. Moreover, the web traffic log analysis hardware or software that can handle such a big data is also expensive. This research paper proposes the design of a large-scale web traffic log analyzer using PHP language to show the visitors' traffic data analysis in the form of graphs. The Hadoop Distributed File System (HDFS) is used in conjunction with other related techniques to gather and store visitors' traffic log. Cloudera Impala is used to query web traffic log stored in HDFS while Apache Thrift is an intermediary connecting Cloudera Impala to PHP web. Upon testing our large-scale web traffic log analyzer on HDFS Cluster of 8 nodes with 50 gigabytes of traffic log, our system can query and analysis web traffic log then display the result in about 4 seconds.

Keywords: Hadoop, Cloudera Impala, Log Analysis.

1. Introduction

Resource planning and data analysis are important for network services in order to increase the service efficiency. System administrators need tools that simple and easy to use for monitoring and analyzing the use of the services. Nowadays, the most active network service on the internet is Hypertext Transfer Protocol (HTTP) and its traffic is increasing gradually. Thus, the web traffic data that needs to be analyzed is large and need a big storage to store all of the traffic data. However, the software and hardware that can store and analyze these big data are too expensive. Some of

free web traffic analyzer software such as AWStats⁽¹⁾, Open Web Analytics (OWA)⁽²⁾ or Piwik⁽³⁾ store data in the plain text format or in the relational database system such as MySQL which are not efficient when data is too big.

Thus, this research paper presents the design and implementation of a PHP web-based large-scale web traffic analyzer using Hadoop Distributed File System (HDFS) to store large-scale web traffic data and using Cloudera Impala to query and analyze web traffic data from HDFS in real-time.

2. Background

In this section, we review Hadoop Distributed File System, Cloudera Impala, Apache Flume and Apache Thrift which are main 4 components to implement our large-scale web traffic log analyzer.

2.1 Hadoop Distributed File System

The Hadoop Distributed File System (HDFS)⁽⁴⁾ is a distributed file system that can be scalable to support thousands of commodity machines. The design of HDFS is fully inspired by Google File System (GFS)⁽⁵⁾ which has master/slave architecture. HDFS is designed to work with large data sets requiring tens of petabyte of storage. HDFS operates on top of file systems of the underlying OS. HDFS is written in Java language and is highly fault-tolerant. Each HDFS cluster has one master node, called *namenode*, which manages the metadata information and several nodes, called *datanodes*, which manage storage attached to the nodes that they run on, store the actual data.

Fig. 1. shows the mechanism of HDFS. Clients contact the namenode machine for file metadata and perform actual file I/O directly with the datanodes.

Fig. 1. The Mechanism of HDFS.

2.2 Cloudera Impala

Cloudera Impala⁽⁶⁾ is an open source Massively Parallel Processing (MPP) query engine that runs on Apache Hadoop. Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats. The fast response for queries enables interactive exploration and fine-tuning of analytic queries, rather than long batch jobs traditionally associated with SQL-on-Hadoop techniques. Impala provides access to data in Hadoop without requiring the Java skills required for MapReduce⁽⁷⁾ jobs. Impala can access data directly from the HDFS file system. Impala is pioneering the use of the Parquet⁽⁸⁾ file format, a columnar storage layout that is optimized for large-scale queries.

2.3 Apache Flume

Apache Flume⁽⁹⁾ is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flow. It is robust and fault-tolerant with tunable reliability mechanisms and many failovers and recovery mechanisms. It uses a simple extensible data model that allows for online analytic applications. Fig. 2. shows a Flume agent which receives log data, called *Event*. Then, the *Event* flows from *Source* to *Channel* and then to *Sink*.

Fig. 2. Apache Flume Architecture.

2.4 Apache Thrift

Apache Thrift⁽¹⁰⁾ is an interface definition language and binary communication protocol that is used to define and create services for numerous languages. It is used as a remote procedure call (RPC) framework that aims to make reliable, performant communication and data serialization as efficient and seamless as possible. It combines a software stack with a code generation engine to build services that work efficiently to a varying degree and seamlessly between C#, C++, Cappuccino, Cocoa, Delphi, Erlang, Go, Haskell, Java, Node.js, OCaml, Perl, PHP, Python, Ruby, and Smalltalk. It is originally developed at Facebook. Thrift was open sourced in April 2007 and entered the Apache Incubator in May, 2008. Thrift became an Apache TLP in October, 2010. Apache Thrift aims to embody the following values: Simplicity, Transparency, Consistency, and Performance.

3. Design and Implementation

In this paper, we propose a design and implementation of a large-scale web traffic log analyzer on Hadoop Distributed File System. Fig. 3. shows how our system works. We have a set of web servers that send web traffic log from Apache web servers (access log and error log) to HDFS via Apache Flume. Users use our web-based log analyzer implemented in PHP language to query and display web traffic log information such as Top Visiting IP, URL error, Top Website Access, and Top Webpage Access. Then, the web-based log analyzer invokes user's requests via PHP Thrift library to contact and order Cloudera Impala to query user's request in SQL style.

Fig. 3. The Design of Large-Scale Web Traffic Log Analyzer on HDFS.

Fig. 4. HDFS Cluster Diagram.

3.1 HDFS Cluster

We installed our testbed system on 9 PCs (nodes) which are one namenode and eight datanodes. All of the nodes connect to 1Gbps switch. We also use namenode as the web server for our large-scale web traffic log analyzer.

Thus, in namenode, we have installed 6 programs which are HDFS, Apache Flume, Cloudera Impala, Apache Web Server, PHP, and MySQL. For datanodes, we have installed only 2 programs which are HDFS and Cloudera Impala as shown in Fig. 4.

3.2 Apache Web Server and Apache Flume

In our web traffic log analyzer, the web traffic log (access log and error log) comes from Apache Web Server and we want to store them in HDFS. The design of Apache Flume in this paper is implemented as shown in Fig. 5.

Fig. 5. The Design of Apache Flume.

Each web server has to install Apache Flume and configure Apache Flume's *Source* with "tail -f" command. For web server access log and error log, by default, locates at /etc/httpd/logs/access_log and /etc/httpd/logs/error_log respectively. Thus, we configure Flume's *Source* with command "tail -f /etc/httpd/log/access_log" for access log and command "tail -f /etc/httpd/log/error_log" for error log.

Agent2 and Agent3 are agents of Apache Flume that read web server traffic log via *Source* and transfer them to another Thrift agent, Agent1 via *Sink*. Then Agent1 read traffic log from multiple *Sources* (*Sink* of Agent2 and Agent3) put that data to *Sink* which connected to HDFS.

3.3 Cloudera Impala Table Creation

Web traffic log in HDFS that receives from web servers are stored in plain text. Normally we need to write a Java program in MapReduce paradigm to extract and retrieve data⁽¹¹⁾. However, the cost of starting MapReduce process is expensive and its method to query information is not intuitive like the structural query language (SQL). With Cloudera Impala, we can turn HDFS plain text file format into Impala table and then we can use Impala to query information with SQL syntax. Moreover, The Impala's Parquet file format, a column-oriented binary file format, is good for queries scanning particular columns within a table. However, at first, we need to map HDFS plain text to Impala table with plain text file format and then query and insert records from Impala table with plain text file format to Impala table with Parquet file format. In our paper, we use *crontab* to transfer data of Impala table with plain text file format to Impala table with Parquet file format every day at midnight to keep the number of records in Impala table with plain text file format as low as possible.

Thus, we create 4 Impala tables in our database: *textAccessLog*, *textErrorLog* are Impala table with plain text file format that respectively maps today HDFS access_log and error_log data, *parquetAccessLog*, *parquetErrorLog* are the Impala table with Parquet file format that respectively store all of access_log and error_log data except today.

3.4 PHP and Cloudera Impala via Apache Thrift

We use PHP language to implement our web-based large-scale web traffic log analyzer. Then, our web cannot connect directly to Impala because its implementation is Java language. However, Thrift is a middle layer that can make the communication between PHP and Cloudera Impala possible.

4. Experiments

In the experiments, we have setup 8 datanodes HDFS cluster with 1Gbps Ethernet connection. Each datanode has the specification as follows: Intel Core 2 Quad Processor CPU, 8 GB of RAM, and 1 TB of Hard disk. Then, we simulate a 50 GB of web traffic log and store it in HDFS via Impala table with plain text file format and Impala table with Parquet file format. We adjust the number of datanodes that store our 50 GB of web traffic log from 3 to 8 datanodes and then we prepare 3 queries $q1$, $q2$ and $q3$ to measure the query time usage of Impala tables as follows:

$q1$: select COUNT(*) from *table*.

$q2$: select COUNT(*) from table GROUP BY ip.

$q3$: select ip, COUNT(*) from table GROUP BY ip ORDER BY COUNT(*).

4.1 Query Time Usage

For the Impala tables with plain text file format, the result in Fig. 6. shows that with 3 datanodes all of 3 queries' time usage are about 37 seconds and query time usage drop dramatically when we add more datanodes to HDFS cluster. Thus, we end up with around 15 seconds query time usage on 8 datanodes HDFS cluster.

For the Impala tables with Parquet file format, the result in Fig. 7. shows that with 3 datanodes the query $q1$ takes about 1 second and the query $q2$ and $q3$ take about 5 seconds. When we add more databases to HDFS cluster, the query time of all 3 queries does not effect that much because they are already fast. Thus, we end up with 0.9 seconds query time for $q1$ and about 3.8 seconds query time for $q2$ and $q3$.

Fig. 6. The Query Time Usage on Impala Tables with Plain Text File Format.

Fig. 7. The Query Time Usage on Impala Tables with Parquet File Format.

4.2 Query Speedup

We calculate the queries' speedup when adding more datanodes with the in equation (1) by using query time usage for 3 datanodes as the base of query time usage.

$$speedup = \frac{\text{query time usage for 3 datanodes}}{\text{query time usage for } n \text{ datanodes}} \quad (1)$$

The speedup for querying data from Impala tables with plain text file format and Impala tables with Parquet file format shown in Fig.8. and Fig.9, respectively. We found that the increasing of datanodes to work with Impala tables with plain text format of 50GB data size has linear speedup. The query can be speeded up almost 2.5 times when we have 8 datanodes comparing to 3 datanodes in the cluster.

However, the query speedup on Impala tables with Parquet file format does not scale very well because the query time is already very low. We can gain speed up about 1.2 times when we use 8 datanodes comparing to 3 datanodes in HDFS cluster.

Fig. 9. The Query Speedup on Impala Tables with Plain Text File Format.

Fig. 10. The Query Speedup on Impala Tables with Plain Text File Format.

4.3 Web Traffic Log Analyzer UI

Our large-scale web traffic log analyzer is a web application implemented in PHP language. On the dashboard, it displays the information of Top10 visiting IP, Top10 website access, Top10 URL error and Top10 webpage access as shown in Fig.11. Also, system administrators can generate reports in the specific range of time as shown in Fig.12. Each page response time is around 4 seconds on 50 GB of log size and 8 datanodes in the HDFS cluster.

Fig. 11. Web Traffic Log Analyzer Dashboard.

Fig. 12. Web Traffic Log Analyzer Detail Page.

5. Conclusions

This paper presents the design and implementation of a large-scale web traffic log analyzer using PHP language together with HDFS to store traffic log, Cloudera Impala to query data from HDFS, Apache Flume to receive apache web server traffic log and send them HDFS, and Apache Thrift which is a communication channel between our web in PHP language and Cloudera Impala API in Java language.

In our experiments, we have queried data from HDFS via Impala tables with plain text file format and Impala tables with Parquet file format. We found that the query time usage on Impala tables with Parquet file format is a lot faster than on Impala tables with plain text file format. With 50 GB of traffic log data stored on 8 datanodes of HDFS cluster, The query time usage for Impala table with plain text file format takes around 15 seconds whereas 0.9 seconds is used for querying data on Impala table with Parquet file format. Also, the use of hybrid between Impala tables with plain text file format to store the current day traffic log and Impala tables with Parquet file format to store past traffic log, each page of our web traffic log analyzer has only about 4 seconds response time.

References

- (1) AWstats, <http://www.awstats.org/>.
- (2) OWA, <http://www.openwebanalytics.com/>.
- (3) Piwik, <http://www.piwik.org/>.
- (4) K. Shvachko, H. Kuang, S. Radia, R. Chansler: "The Hadoop Distributed File System", The 26th IEEE Symposium on Massive Storage Systems and Technologies, 2010.
- (5) S. Ghemawat, H. Gobioff, and S.T. Leung: "The Google File System", Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003.
- (6) Cloudera Impala, <http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>.
- (7) J. Dean and S. Ghemawat: "MapReduce: simplified data processing on large clusters", ACM 51, pp 107-113, 2008.
- (8) Apache Parquet, <http://parquet.apache.org/>.
- (9) Apache Flume, <http://flume.apache.org/>.
- (10) Apache Thrift, <http://thrift.apache.org/>.
- (11) C. Rattanapoka: "The Design and Implementation of Computer Traffic Log Searcher System using Hadoop Map/Reduce Framework", The Journal of Industrial Technology, Vol. 8, No. 3, 2012. (In Thai).