Study of the Big Data Collection Scheme Based Apache Flume for Log Collection

Sooyong Jung and Yongtae Shin

***Abstract*—With the advances in IT technology and the rapid adoption of smart devices, users can more easily produce, distribute and consume data through network access anytime, anywhere. The data generated by users in response to these changes has increased dramatically. This has required companies to collect large amounts of logs, and these companies are actively researching and developing big data collection technologies. In this paper, we have studied the big data collection technology based on Apache Flume for bulk log collection. The structure for bulk log processing is designed to be matched with one web server and one Flume agent, and the Flume agents connected to the web server are connected to the Flume agent that plays the role of storing in the Hadoop distributed file system. This makes the collection of big data logs more efficient.**

***Index Terms*—Big data, big data collection technology, Apache Flume, Apache Chukwa, hadoop distributed file system.**

I. Introduction

In recent years, the term Big Data has emerged to describe a new paradigm for data applications. New technologies tend to emerge with a lot of hype, but it can take some time to tell what is new and different. While Big Data has been defined in a myriad of ways, the heart of the Big Data paradigm is that is too big (volume), arrives too fast (velocity), changes too fast (variability), contains too much noise (veracity), or is too diverse (variety) to be processed within a local computing structure using traditional approaches and techniques. The technologies being introduced to support this paradigm have a wide variety of interfaces making it difficult to construct tools and applications that integrate data from multiple Big Data sources. This report identifies potential areas for standardization within the Big Data technology space [1].

Big data collection is a step of collecting information received from source data. Source data is divided into internal data and external data according to the source location. Internal data refers to information stored in a company's information system, DB and object Internet equipment. External data is data that is not owned by an organization, and it means information such as news, blogs, Facebook, Twitter, etc., as social media data.

As IT technology develops, internal data and external data are increasing rapidly.

This has required companies to collect large amounts of logs, and these companies are actively researching and developing big data collection technologies. In this paper, we have studied the big data collection technology based on Apache Flume for bulk log collection. The composition of this paper is as follows. In Section 2, we will look at the big data collection technologies Flume and Chukwa. In Section 3, we present a structure for mass log processing proposed in this paper. Finally, Section 4 presents conclusions and future work.

Manuscript received March 22, 2018; revised June 8, 2018.

Sooyong Jung and Yongtae Shin are with Dept. of Computer Science Graduate School, Soongsil University, 369 Sangdo-Ro, Dongjak-Gu, Seoul, Korea (06978) (e-mail: kevinhaha777@gmail.com, sooyong.jung@gmail.com, shin@ssu.ac.kr).

II. Big Data Collection Technology

A. Apache Flum

Flume has a dictionary meaning such as artificial water, water channel, etc. It is a big data collection technology that collects the logs that are loaded on a server providing several services to one log collection server. Flume is based on stream-oriented data flow, which collects logs from all specified servers and loads them on central storage, such as Hadoop's Hadoop distributed file system. Flume is suitable for building a Hadoop-based Big Data Analysis System [2].

Flume's core objectives are system reliability, scalability, manageability, and extensibility, designed to meet these four core objectives. (Table I) provides a description of Flume's core objectives.

TABLE I: Flume's Core Objectives

Core Objectives	Explanation
System Reliability	Ability to transmit without loss of log in case of failure
System Scalability	Easy-System to add and remove agents.
Manageability	Combined structure makes it easy to manage
Feature Extensibility	Easily add new features

Based on the core goals above, Flume easily addresses some of the most challenging issues of large scale distributed systems, such as physical failure of equipment, network bandwidth, resource shortage such as memory, software down, CPU usage overflow etc. , And the log can be continuously collected even if one of the various components of the system goes down [2].

Flume's data flow is based on stream-oriented data flow. A data flow is a method in which one data stream is transferred from a generation point to a destination point and processed. The data flow consists of a series of nodes that transmit and collect events. Fig. 1 shows the data flow of Flume [2].

As shown in Fig. 1 the data flow can be expressed in three layers as Agent Tier, Collector Tier and Storage Tier as shown in Fig. 2. Fig. 2 shows the Flume hierarchy [2].

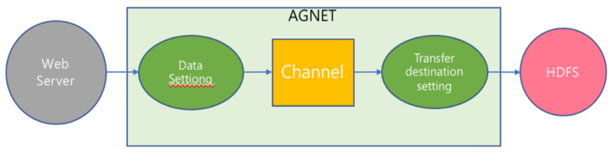

Fig. 1. Flume's data flow.

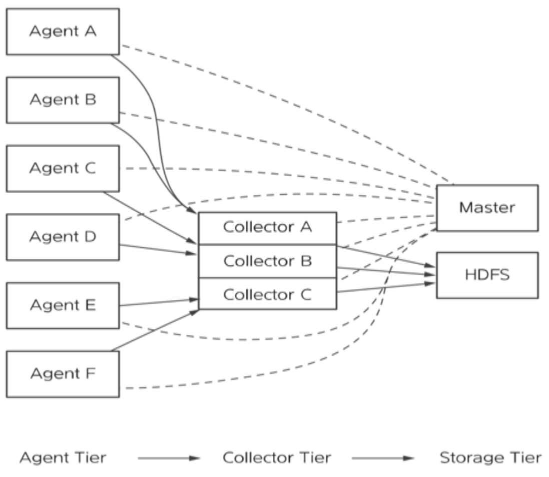

Fig. 2. Flume's data flow.

The agent tier is an area composed of each agent node. In the agent tier, each agent node is installed in the equipment where log data to be collected is generated. When there are several devices that generate data, agent installation installs agent nodes for each equipment, and these agent nodes form an agent tier [2].

The collector tier is an area for collecting information received from the agent node. The data collected by the agent node is transferred to the collector node. Collector nodes are usually on different machines and can be configured with multiple collector nodes. When data is transferred from an agent node to a collector node, a data flow can be set up such as which data is to be sent and where to be processed, and the data is moved and stored in the storage tier according to the setting [2].

The storage tier consists of a master node that manages settings for agent nodes and collector nodes, and a Hadoop distributed file system where data is stored. The main role of the master node sets up this data flow. In other words, the master node can set each logical node through the program, and this role is one of the great advantages of Flume. The Flume can be freely changed through the master node even when each node is running. This means that you can dynamically keep track of where you want to get log data, how to process it, and where to store it [2].

A node can be divided into a physical node and a logical node. The Physical Node is a Java process running in the Java virtual machine environment installed on the machine. A physical node operates the same as a logical node, but a plurality of logical nodes can be created on a physical node. Therefore, you can create various Logical Nodes to configure the data flow according to the required application. Each Logical Node (including agent nodes and collector nodes) follows a fairly flexible abstraction model.

Flume consists of an architecture in which the reliability of the system is maintained or increased even if additional nodes are installed. In this case, if the data flows are horizontally extended, the load increases for each layer due to the increased number of nodes, which may cause a problem in the overall throughput. However, Flume can add additional equipment to the system itself, which can be adjusted so that the overall performance is not compromised by distributing the load for each layer. [2].

Therefore, Flume guarantees reliability not only for horizontal extension but also for data transmission. Flume ensures End-to-End reliability, Store on Failure reliability and Best-Effort reliability. End-to-End reliability means that when the Flume receives an event, it ensures data processing to the end point. Store on failure reliability refers to architectural support that allows data to be stored and retransmitted on the local disk in the event of a failure, or to wait until another collector node is selected before resubmitting. Best-Effort reliability indicates that reliability may be reduced due to lost data in the sense that the data being processed may be lost when it fails [2].

Depending on the characteristics and hierarchical structure of the flume, the flume can be implemented by various architectural models by the user. Typical architecture of Flume is Hadoop distributed file system direct link structure, single plum agent linkage structure, multi-flume agent linkage structure, and bulk log processing structure [2].

1) Hadoop distributed file system direct link structure

The Hadoop distributed file system direct link structure is a structure that connects directly to the Hadoop distributed file system to collect logs of web servers. Hadoop Distributed File System Directly linked structure requires complex code usage and continuous management for each server to work with Hadoop distributed file system. Therefore, Hadoop distributed file system direct linkage structure is not suitable in terms of maintenance cost or scalability. Fig. 3 represents the Hadoop distributed file system direct linkage structure. [2].

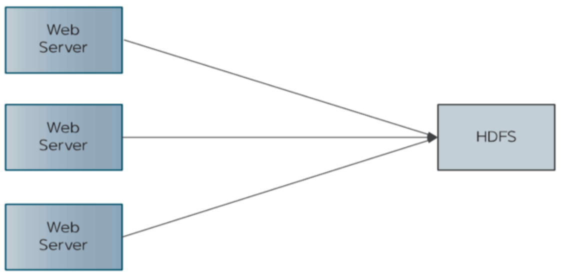

Fig. 3. Hadoop distributed file system direct linkage structure.

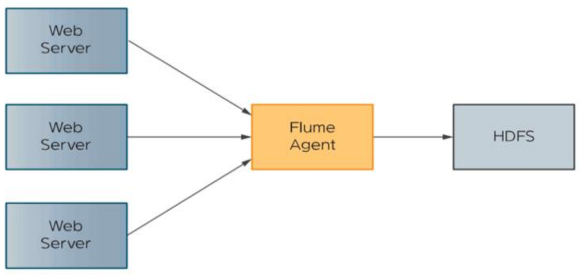

Fig. 4. Single flume agent linkage structure.

2) Single flume agent linkage structure

A single Flume Agent linkage structure delegates the log collection authority to one Flume Agent (Flume Agent) so that each Web Server can provide faster service to its customers. One Flume agent collects logs from a web server, and if a Flume agent fails, it cannot collect logs from all web servers. Also, since all log transfers are concentrated in Flume agents, they are not suitable for large-scale systems. Fig. 4 shows the structure of the single flume agent linkage. [2].

3) *Multi-flume agent linkage structure*

Multi-flume agent linkage structure is similar to linkage structure of single Flume agent. The difference is that log collection is performed using multiple Flume agents. Therefore, the multi-flume agent linkage structure can guarantee availability with minimum investment cost in case of failure. Even if the Flume agent is stopped due to a fault, the multi-Flume agent linkage structure can continue log collection through other Flume agent, thus ensuring service continuity. Multi-Flume Agent linkage structure can be applied according to situation such as fail over for failure response and load balancing function for distributing log event information. Fig. 5 shows the multi-Flume agent linkage structure [2].

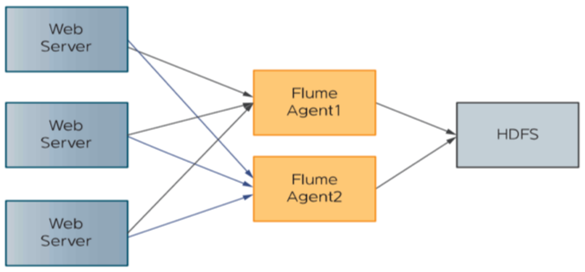

Fig. 5. Multi flume agent linkage structure.

B. *Apache Chukwa*

Apache Chukwa is an open source data collection system for managing large distributed systems. Apache Chukwa is built on top of the Hadoop Distributed File System (HDFS) and Map/Reduce framework and inherits Hadoop's scalability and robustness. Chukwa also includes powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.

Apache Chukwa is a system that collects various logs such as system monitoring logs, application logs, Hadoop logs, etc. of distributed nodes and moves them to the Hadoop distributed file system and processes them. Chukwa was also developed to monitor terabytes of data from thousands of hosts every day. Fig. 6 shows the system configuration of Chukwa.

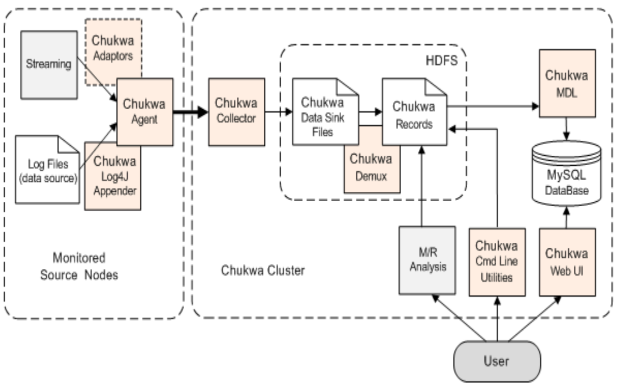

Fig. 6. The system configuration of Chukwa.

III. Math Big Data Distributed File System

A. *Hadoop Distribute File System*

The HDFS (Hadoop Distributed File System) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject. Fig. 7 shows the HDFS Architecture [3].

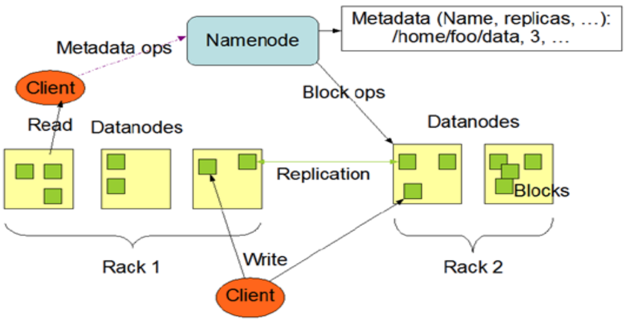

Fig. 7. The HDFS architecture.

B. *GlusterFS*

GlusterFS (Gluster File System) is an open source, clustered file system capable of scaling to several petabytes and handling thousands of clients. GlusterFS can be flexibly combined with commodity physical, virtual, and cloud resources to deliver highly available and performant enterprise storage at a fraction of the cost of traditional solutions [4].

GlusterFS clusters together storage building blocks over Infiniband RDMA and/or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design, delivering exceptional performance for diverse workloads [4].

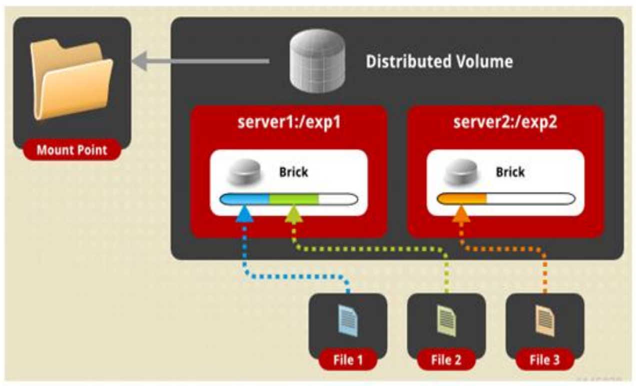

Fig. 8. The gluster FS architecture.

GlusterFS is designed for today's high-performance, virtualized cloud environments. Unlike traditional data centers, cloud environments require multi-tenancy along with the ability to grow or shrink resources on demand. Enterprises can scale capacity, performance, and availability on demand, with no vendor lock-in, across on-premise, public cloud, and hybrid environments. Fig. 8 shows the GlusterFS architecture [4].

C. GlusterFS

The CFS (Cassandra File System) was designed by DataStax Corporation to easily run analytics on Cassandra data. Now implemented as part of DataStax Enterprise, which combines Apache Cassandra, and Solr™ together into a unified big data platform, CFS provides the storage foundation that makes running Hadoop-styled analytics on Cassandra data hassle-free [5].

In contrast to a master-slave architecture like HDFS, CFS is based on Cassandra, so the implementation is peer-to-peer and "masterless." A user is able to create a cluster that seamlessly stores real-time data in Cassandra, performs analytic operations on that same data, and also handles enterprise search operations. Cassandra's built-in replication transparently takes care of replicating the data among all realtime, analytic, and search nodes. A user may configure any type of cluster they desire. Fig. 9 shows the Simple DataStax Enterprise Cluster [6].

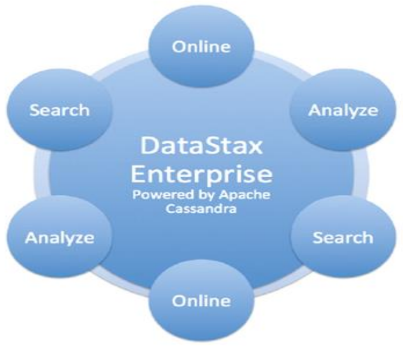

Fig. 9. The simple DataStax enterprise cluster.

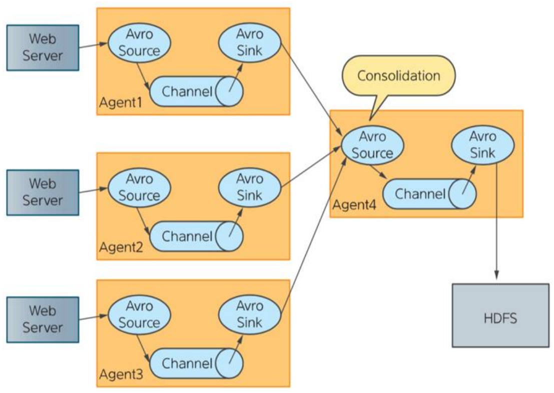

Fig. 10. The structure for bulk log processing.

IV. Proposed Big Data Collection Scheme

The structure for bulk log processing is designed to be matched with one web server and one Flume agent, and the Flume agents connected to the web server are configured to be connected to the Flume agent which plays a role of storing in the Hadoop distributed file system. The Flume agent the structure for bulk log processing is composed of an Avro source for generating events, an Avro sink for consuming events, and a channel for connecting Avro source and Avro Sink, and collects and processes logs from a web server. The structure for bulk log processing is important for the number of agents and the composition ratio of each tier. Basically, Flume has a reliable design that lowers the load factor of the system. However, as the number of nodes increases, the performance of the flume can be improved by designing and proceeding the entire system proportionally. Fig. 10 shows the structure for bulk log processing.

V. Conclusion

In this paper, we have studied a large data collection technology based on Apache Flume for bulk log collection. The structure for bulk log processing is designed to be matched with one web server and one Flume agent, and the Flume agents connected to the web server are connected to the Flume agent that plays the role of storing in the Hadoop distributed file system. This makes the collection of big data logs more efficient.

References

[1] ISO/IEC/JTC. Study Group on Big Data (SGBD). [Online]. Available: http://jtc1bigdatasg.nist.gov
[2] Apache flume. Flume 1.6.0 user guide. [Online]. Available: https://flume.apache.org/releases/content/1.6.0/FlumeUserGuide.html
[3] D. Borthakur. (2008). HDFS architecture guide. *Hadoop Apache Project*. [Online]. Available: http://hadoop. apache. org/common/docs/current/hdfs design. pdf
[4] GlusterFS Developers: Gluster File System 3.3.0 Administration Guide. [Online].
[5] Datastax Corporation, Comparing the Hadoop Distributed File System (HDFS) with the Cassandra File System (CFS), white paper, August 2013.
[6] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, vol. 11, no. 11, pp. 1-10, 2007.

Sooyong Jung received the master's degree from School of Information Sciences, Soongsil in Aug. 2007. He is working as a sales consultant in Openbase Group since April. 2004. His research interests include cloud computing, bigdata, information security, IoT, IT service science.

Yongtae Shin received his bachelor's degree from Department of Industrial Engineering, Hanyang University in Feb. 1985 and master's degree and Ph.D from Univ. of Iowa, Computer Science in Dec. 1990 and May 1994. He is now serving as a Prof. in School of Computer Science and Engineering, Soongsil Univ. from Mar. 1995. His research most interests in multicast, IoT, information security, content security, mobile internet, next generation internet.