재난관련 정보 수집 및 분석을 위한 빅데이터 플랫폼 설계 및 구축

신 응억 뉘엔, 퀴엣 뉘엔 반, 둑티엡부, 김경백
전자컴퓨터공학부
전남대학교

Design a Big Data platform for Collection and Analysis of Disaster Information

Sinh Ngoc Nguyen, Van-Quyet Nguyen, Duc Tiep Vu, Kyungbaek Kim
Dept of Electronics and Computer Engineering,
Chonnam National University
e-mail : sinhngoc.nguyen@gmail.com, quyetict@utehy.edu.vn, ductiep91@gmail.com, kyungbaekkim@jnu.ac.kr

요 약

Every year, we have been impacted with unexpected hurricanes, tornadoes, earthquakes, and other natural disasters. It happens in many places around the world and damages a lot of material property and personals in our society. Also, it has drained the national coffers of many countries on the way of going. There are several researches related to a local disaster management system proposed to reduce the damage from disasters. However, with the increase of disaster information which provided by disaster sensors or social network, these local systems are not efficient in collecting and analyzing huge amount of data. In this paper, we propose a big data platform for collecting and analyzing disaster information from various kinds of sources. We design collecting modules based on Flume and MapReduce for obtaining real-time and historical data from many kinds of sources. And we design storage modules with HDFS, HBase and Hive, and we design analysis modules based on Hadoop and Spark. Based on the prototype design, we implemented and deployed the prototype big data platform in our lab. To evaluate the viability of our prototype, we conducted case studies for arithmetic calculation on weather bigdata.

1. Introduction

Annually, we are impacted by many disasters such as hurricanes, tornadoes, earthquakes, and other natural ones. In order to reduce the damage from the disaster, we need to have the system to send the warning message to all people in the affected area. From there, people have enough time to reduce the damage into minimum.

At the current, there are several systems resolve above problem such as [5], [6]. They focus on the analyzing and notification message, but the efficient in processing huge amount of data is being restricted. The increasing of disaster information from the sensor plays an important role in precision of notification. Currently, most of disaster sensors provide data in real-time at each minute or each hour. So the data from sensor is being increased into big data. Several current systems have the problem with collecting, storage with large amount of data. And it spends more time for analyzing data. That is a crucial thing in disaster notification system.

To resolve all above problems, in this paper we propose a system to collect, store and analyze large amount of disaster information. In order to do this, we use Flume [1] and Sqoop to collect data from many sources such as sensors data and social network, then store it into HBase on Hadoop Distributed File System (HDFS) [3]. The data is preprocessed by Hadoop/MapReduce [2] to remove some redundant information. And we use Spark to analyze the data to give the prediction about the disaster. Also, our system provides personalized warning notification module to deliver alert message to people living in the affected region.

2. Architecture

We propose a big data platform for collecting and analyzing disaster information with four parts as follows: Data Collection, Data Storage, Data Analysis and Notification. In that, Data Collection will collect and aggregate data from many sources such as sensor data, social network data and historical data of disaster. And then store all of them into HDFS by Data Storage module. We use Data Analysis module to analyze and indicate the prediction result for disaster. Also, Personalized Warning Notification will deliver the warning message to people living in the affected area. Figure 1 shows the overview about the architecture of our system. We will explain each of components as below.

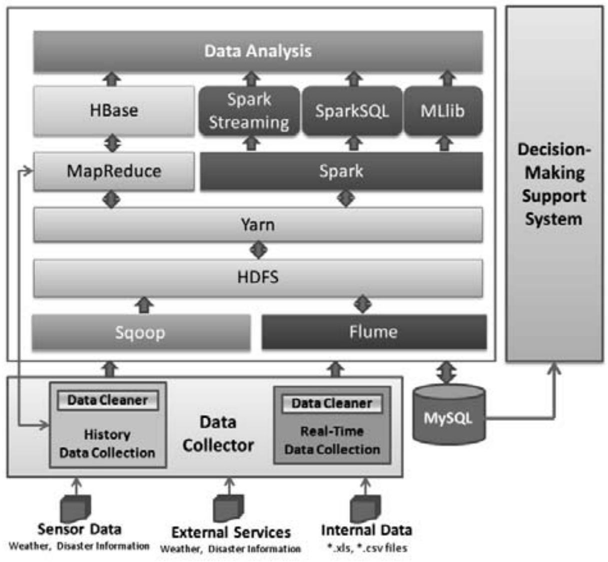


Figure 1: Architecture

The first component in our architecture is Data Collector. We have two kinds of input data, the first one is real-time data from sensor and the second one is historical data. Currently, most of sensors provide the pooling data mechanism for each minute or each hour. We wrote a Java program to make a pooling request for each minute and remove the redundant data. With a collected record for each pooling, we use Flume to puts it to HDFS. In historical data, that is a big volume of data. So we wrote a Java program using Map/Reduce in Hadoop to increase the speed of crawling data. Each map task in program is a process to crawl disaster.

In order to store the data from Data Collector, we use HBase Apache [7] for Data Storage. That is a distributed column-oriented database built on top of the Hadoop file system. Hbase is a data model designed to provide quick random access to huge amounts of structured data. Also, we use Sqoop [8] to import data as file such as *.csv, *.xls into HBase. To do that, we need to import data file into MySQL and transfer all of them into HBase by Sqoop.

Data Analysis is an important component in the system. After having the data, in order to process large amount of them, we wrote the Java program base on Spark as the main analysis module. Spark is a novel framework to process big data. That is an open source cluster computing framework developed at AMPLab. It provides an application programming interface called RDD [9] with in-memory processing. Also, Spark provides Spark Streaming for streaming data, SparkSQL for query data, and MLlib for machine learning application. From there, we use SparkRDD API to write the program for big data analyzing. We implement several algorithms such as K-Mean, Naive Bayes to cluster and classify the data.

The last module of the system is Personalized Warning Notification. It receives the result from the Data Analysis module, and deliver alert messages to people living in the affected area. Also, it provides the statistic data stored in MySQL for user with other purpose.

3. System Deployment

There are five machines in our system to deploy a platform with a master node and four slave nodes consequently named as Tiger, Lion, Jaguar, Cheetah, and Leopad, which shows at Figure 3. We deployed Spark, Hadoop, and Hbase on all of 5 machines. Each compute node has 4 CPUs and 8GB of RAM. The master node IP is 192.168.20.101, and other slave nodes, region servers, or workers are from 192.168.20.102 ~ 192.168.20.105. The network deployment of the big data frameworks in our platform is as shown in Figure 2.

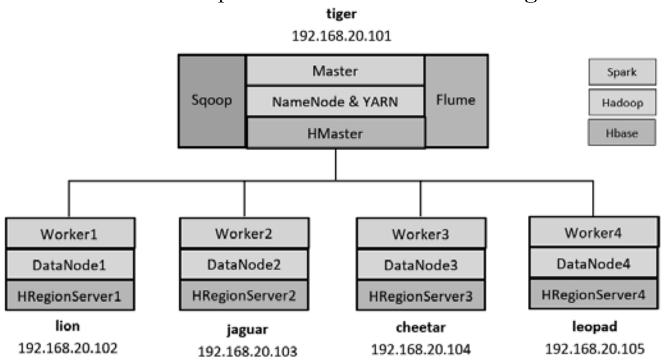


Figure 2: System Deployment

For Spark framework, the master process and the built-in standalone cluster are started on the master node (Tiger machine). Each worker is responsible for launching the executor process. Here, the amount of memory to use for the driver process is 1GB, while amount of memory to use per executor process is 6GB.

For Hadoop framework, the NameNode process and the YARN cluster manager are launched on the master node, and each slave node is responsible for launching its own DataNode process. In our system, the amount of physical memory that can be allocated for containers in our Hadoop system is 32GB. Therein, the minimum allocation for every container request at the resource manager is 1GB, the amount of memory to request from the scheduler for each map/reduce task is 2GB.

For HBase framework, the HMaster process is launched on the master node. This master processes may be collocated with the Hadoop NameNode and Resource Manager. Designate the remaining nodes as RegionServer nodes. Each node runs a RegionServer, which may be collocated with a Hadoop NodeManager and a DataNode.

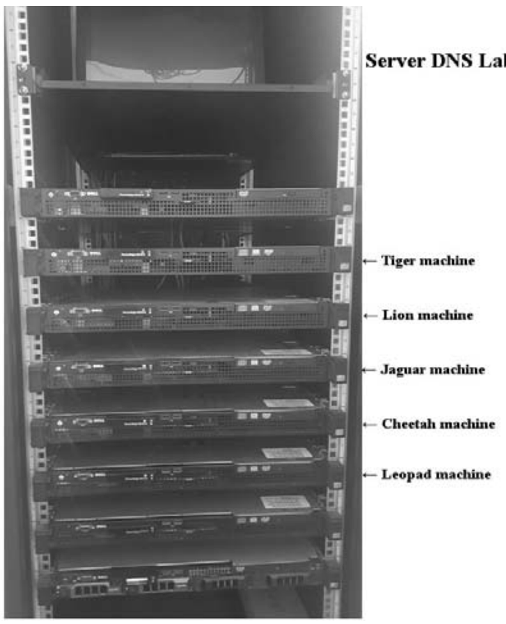


Figure 4: Real System Deployment

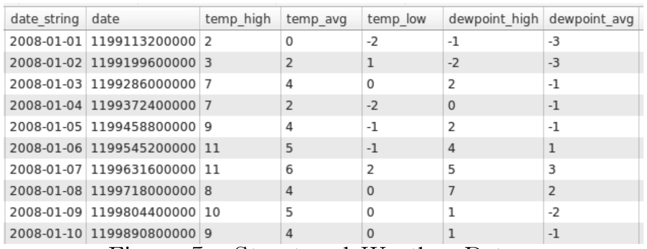
Besides, other frameworks which support to import and transfer the data to HDFS such as Sqoop and Apache Fume are launched on the master node.

4. Case Study of Analyzing Weather Data at South Korea

In this paper, we propose a big data platform for collecting and analyzing disaster information. In order to validate our system, we provide a case study of analyzing weather data at Gwangju city, South Korea. In actually, we will calculate the average of temperature from a dataset which collected from the website on the internet from 2008 to 2015

A. Dataset

Website "https://www.wunderground.com/weather/api/" provides information about the weather. It has real time updated for information, so it is big data when we get all data in daily or hourly. To speed up for crawling data, we write a program using MapReduce to get data in history from 2008 to 2015. Each map-task in the program runs on a core of system. We have total 32 tasks run on concurrently. After having the data, we process it to remove the redundant information and put it into HDFS. Figure 4 shows show the structure of weather data after crawling. There are more 11 million records in dataset.



date_string	date	temp_high	temp_avg	temp_low	dewpoint_high	dewpoint_avg
2008-01-01	1199113200000	2	0	-2	-1	-3
2008-01-02	1199199600000	3	2	1	-2	-3
2008-01-03	1199286000000	7	4	0	2	-1
2008-01-04	1199372400000	7	2	-2	0	-1
2008-01-05	1199458800000	9	4	-1	2	-1
2008-01-06	1199545200000	11	5	-1	4	1
2008-01-07	1199631600000	11	6	2	5	3
2008-01-08	1199718000000	8	4	0	7	2
2008-01-09	1199804400000	10	5	0	1	-2
2008-01-10	1199890800000	9	4	0	1	-1

Figure 5 : Structured Weather Data

B. Implementation of Analysis

Method 1: Calculate the average temperature in monthly

Require: *date* is the lable to identify for each day. temp_avg is value of temperature in a day

```
dataset = ctx.textFile(Weather.csv);
filter_data = dataset.filter(){
        return (dataset.split(",")[temp_avg].contains("-")
};
maper_data = filter_data.mapToPair{
        return new Tuple<key, value>(date,<temp_avg,count>);
};
reduce_data = maper_data.reduceByKey(){
        return new Tuple2<Float>(temp1i+temp1j, count1i+count1j);
};
result = reduce_data.mapValues(){
        return new Tuple2<Float>(temp / count);
};
result.saveAsHadoopFile();
```

In order to calculate the average of temperate with bigdata, we write a program using Spark API for processing. In that, we read dataset on HDFS and split data into key and value pair by mapper task in MapReduce. The key is the data in date column and value is the data in temp_avg column. Then, we use Reducer task to group by all value with the same key. Finally, we can calculate easily the average after having the result of Reducer task. In our case study, we calculate the monthly average of temperate from 2013 to 2015, and the yearly average of temperate from 2008 to

2015 at Gwangju city as Method 1.

C. Result

In the first case of computation, we calculate the monthly average of temperature from 2013 to 2015. Figure 5 shows the result of calculation. From there, we have a perspective about the temperature in 3 years.

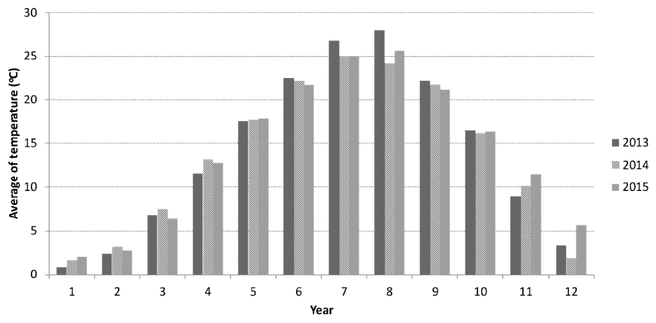


Figure 5: Monthly average of temperature

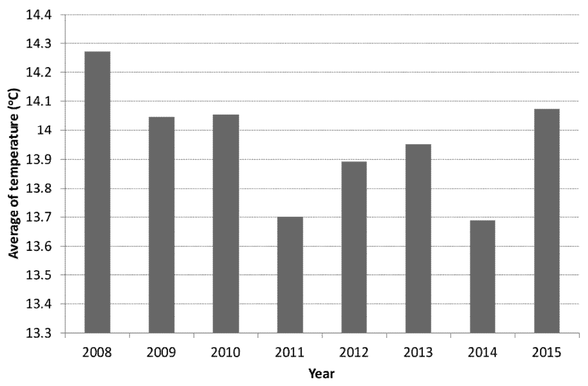


Figure 6: Yearly average of temperature

In the second of computation, we calculate the yearly average of temperature from 2008 to 2015. Figure 6 shows the result of calculation.

5. Conclusion

In this paper, we proposed a big data platform to collect and process the information in disaster. Our system collects and stores huge amount of data from many sources. Also, it provides big data processing to process a lot of data in concurrently. Through the architecture, we have deployed a system in the reality with four nodes for data processing. Also, we evaluate our platform with case study of analyzing weather data which crawled from website to calculate the monthly average of temperature or yearly average of temperature.

Acknowledgements

This work was supported by the National Research Foundation of Korea Grant funded by the korean Government(NRF-2014R1A1A1007734). This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-R2718-16-0011) supervised by the IITP(Institute for Information & communications Technology Promotion)

References

[1] Hoffman, Steve. Apache Flume: Distributed Log Collection for Hadoop. Packt Publishing Ltd, 2013.

[2] Chu, Cheng, et al. "Map-reduce for machine learning on multicore." Advances in neural information processing systems 19 (2007): 281.

[3] Borthakur, Dhruba. "HDFS architecture guide." HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf (2008): 39.

[4] Boszormenyi-Nagy, Ivan, and Geraldine M. Spark. Invisible loyalties: Reciprocity in intergenerational family therapy. Harper & Row, 1973.

[5] Tiep Vu Duc, Quyet Nguyen-Van, Kyungbaek Kim. "Design and Implementation of Geo-Social information based Personalized Warning Notification system" KISM 2016

[6] Lee, Jungki, et al. "A GIS-based design for a smartphone disaster information service application." Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on. IEEE, 2011.

[7] Vora, Mehul Nalin. "Hadoop-HBase for large-scale data." Computer science and network technology (ICCSNT), 2011 international conference on. Vol. 1. IEEE, 2011.