Middle-East Journal of Scientific Research 24 (Techniques and Algorithms in Emerging Technologies): 420-424, 2016
ISSN 1990-9233;
© IDOSI Publications, 2016
DOI: 10.5829/idosi.mejsr.2016.24.TAET23350

Big Log Search and Analysis Using Elk Environment

[1]Mrs. I. Nivetha and [2]Mrs. J. Preethi

[1]PG Scholar (CSE), Anna University Regional Campus, Coimbatore, India
[2]Assistant Professor (CSE), Anna University Regional Campus, Coimbatore, India

Abstract: The collection of log record becomes more important for any website. Because collecting these log records are used to understand trends for improving the site or making marketing/management decisions. Most of the companies are trying to reduce the amount of time required to access fresh information. The best solution is to develop real-time analytics systems along the log data collection. But collecting and analyzing big log based information of internet monsters like Twitter, facebook is not an easy one. And the traditional databases cannot hold huge amount of data and access the data in an efficient manner. This paper provides the real time search and analysis of big log using elastic search, which is the modern search engine based on Lucene. This paper presents a real-time big data search method: First, Logstash for huge log data collection; then Elasticsearch for search solution and storage; finally Kibana for visualization. The paper mainly focused on how to collect big data log and make it available to both analytics and search in near real time without using any Hadoop eco system unlike Hbase as NoSQL data base. Hbase is replaced by Elasticsearch. As Elasticsearch itself has a storage in NoSQL pattern. More over as it is distributed system (Master-Slave architecture).

Key words: Elastic Search · Big Data · Logstash · Realtime search · Kibana

INTRODUCTION

In this era, hundreds of millions of computers and mobile devices are constantly creating a surprising amount of information, which includes both human beings, but also a variety of other things. Moreover, this information will only accelerate creation continues, not stagnation. Change is so great that the last decade we Computational tools used has no ability to meet these new challenges.

However, this does not mean that we can only sit still; contrary, Like Google, Yahoo!, Amazon and other Internet giants Facebook and well-being Growing number of start-ups, as presented, we can adopt with a completely different approach to solve the database, data storage [1] and other. Computer information processing issues and these major technological innovations Occurred in the Internet to provide consumers with services.

There is billions of log events generated every day in giant company. Mining the business value from the big log is a big challenge because traditional technology cannot scale so much. For example, if we load thousands of billions of log events into Oracle database, then Oracle cluster is necessary. If the Oracle cluster scales, then the performance gets lower and some of the feature of SQL will be lost.

Existing centralized search engine from such a mass of information really needs to quickly retrieve the information is becoming more and more difficult, so the search engine system should have distributed processing capabilities, according to the need to deal with the growth of information, constantly expanding size of the system to enhance the system's ability to process information [2]. Therefore, building a distributed search engine becomes very meaningful.

Time is money. If we want to search and analyze the log in real time, it will give oracle cluster big pressure. From the experience from Internet Company like Facebook and Yahoo, traditional RMDBS cannot process so big data in a reasonable response time.

Elastic Search is a construct based on Lucene [3] open source, distributed, RESTful search engine. Designed for cloud computing, to achieve real-time search, stable, reliable, fast, easy to install. Support for data using JSON over HTTP index.

Corresponding Author: Mrs. I. Nivetha, PG Scholar (CSE), Anna University Regional Campus, Coimbatore, India.

Logstash is a tool for managing logs. It supports virtually any type of log, including system logs, error logs and custom application logs. It can receive logs from numerous sources, including syslog, messaging (for example, rabbitmq) and jmx and it can output data in a variety of ways, including email, web sockets and to Elasticsearch.

Elasticsearch is a full-text, real-time search and analytics engine that stores the log data indexed by Logstash. It is built on the Apache Lucene search engine library and exposes data through REST and Java APIs. Elasticsearch is scalable and is built to be used by distributed systems.

Kibana is a web-based [4] graphical interface for searching, analyzing and visualizing log data stored in the Elasticsearch indices. It utilizes the REST interface of Elasticsearch to retrieve the data and not only enables users to create customized dashboard views of their data, but also allows them to query and filter the data in an ad hoc manner.

We have established a Web site or application and want to add a search function, so we hit that: the search is very difficult. We want our search solution to be fast, we want to have a zero configuration and a completely free search mode, we want to be able to simply using JSON over HTTP index data, we want our search server [5] is always available, we hope to one Taiwan began and extended to hundreds, we want real-time search, we want simple multi-tenant, we hope to build a cloud solution. Elasticsearch designed to address all these issues and more.

Lucene is the de facto standard search engine of many internet companies. It's like the engine of a car but it's not suitable for big data and cloud environment computation. Solr and ElasticSearch are both based on Lucene; both of them are open source project. Solr is for standalone application. ElasticSearch is designed for modern, cloud environment.

Best of all the features of ElasticSearch is the nearly real time search. Although it's implemented in Java, there are many clients are supported like PHP, Ruby, Perl, Scala, Python,. NET, JavaScript, Erlang and Clojure. Django, Couchbase and SearchBox are integrated into ElasticSearch. MongoDB, CounchDB, RabbitMQ, RSS, Sofa, JDBC, FileSystem [6], Dropbox, ActiveMQ, LDAP, Amazon, SQS, St9, OAI and Twitter can be imported into Solr directly. Zookeeper and internal Zen Discovery can be used for automatic node discovery. All the shards and replicas can move to any node of the Elastic Search cluster. The indexes can routine to any of the shards if you want to. And it's RESTful architecture [7].

Collecting the log evens in real time is the first step for searching and analyzing. During this test, Logstash is used to do the job.

Although this paper is about real time search, actually it can be viewed as a starting entry of near real time OLAP system of big data. Kibana is such example that can make sense of a mountain of logs. Kibana helps you to chart it and rank it and play with the numbers. Kibana is a highly scalable [8, 9] interface for Logstash and ElasticSearch that allows you to efficiently search, graph, analyze and otherwise make sense of a mountain of logs. For Kibana and Logstash, all the log events are stored back up to ElasticSearch; by default the index are based on files. If its memory based, then the performance definitely improves a lot so that Kibana itself can be used an OLAP for log management.

Kibana data [10] is located in local file system; while in our experiment projects all log data are stored in hbase [11], only index itself in stored in local file system. It means more log data can be processed compared to Kibana., it good to mention that integrated with machine learning and data mining, OLAP of big data can bring even more value. There are some example applications of big data: Precision Marketing, Genetic Engineering, Climate Prediction etc.

System Design Points: Logstash collects log events from end user machine.

At logstash side, it collects log based events into ElasticSearch; ElasticSearch plugin is used to analyze the log events and index each log event into ElasticSearch cluster. Figure 1 shows how the log events are collected, indexed and stored from twitter Application. Logstash is

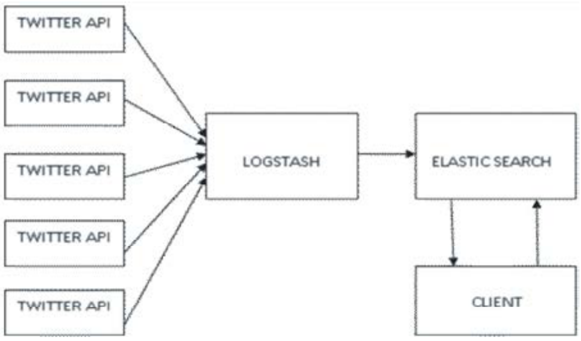

Fig. 1: log events collecting and indexing flow

a tool for receiving, processing and outputting logs. All kinds of logs. System logs, webserver logs, error logs and application logs.

Using Elasticsearch as a backend datastore and kibana as a frontend reporting tool, Logstash acts as the workhorse, creating a powerful pipeline for storing, querying and analyzing your logs. With an arsenal of built-in inputs, filters, codecs and outputs, you can harness some powerful functionality with a small amount of effort.

Data Structure and Algorithm: One of the reasons that object-oriented programming languages are so popular is that objects help us represent and manipulate real-world entities with potentially complex data structures.

The problem comes when we need to store these entities. Traditionally, we have stored our data in columns [12] and rows in a relational database, the equivalent of using a spreadsheet. All the flexibility gained from using objects is lost because of the inflexibility of our storage medium.

But what if we could store our objects as objects? Instead of modeling our application around the limitations of spreadsheets, we can instead focus on *using* the data. The flexibility of objects is returned to us.

An *object* is a language-specific, in-memory data structure. To send it across the network or store it, we need to be able to represent it in some standard format. JSON is a way of representing objects in human-readable text. It has become the de facto standard for exchanging data in the NoSQL world. When an object has been serialized into JSON, it is known as a *JSON document*.

Elasticsearch is a distributed *document* store. It can store and retrieve complex data structures— serialized as JSON documents—in *real time*. In other words, as soon as a document has been stored in Elasticsearch, it can be retrieved from any node in the cluster. Of course, we don't need to only store data; we must also query it, en masse and at speed. While

NoSQL solutions exist that allow us to store objects as documents, they still require us to think about how we want to query our data and which fields require an index in order to make data retrieval fast.

In Elasticsearch, all data in every field is indexed by default. That is, every field has a dedicated inverted index for fast retrieval. And, unlike most other databases [13], it can use all of those inverted indices in the same query, to return results at breathtaking speed.

Documents are indexed-stored and made searchable-by using the index API. But first, we need to decide where the document lives. As we just discussed, a document's _index, _type and _id uniquely identify the document. We can either provide our own _id value or let the index API generate one for us.

If the document has a natural identifier (for example, a user_account field or some other value that identifies the document), you should provide your own _id, using this form of the index API:

```
PUT /{index}/{type}/{id}
{
 "field": "value",
...
}
```

For example, if our index is called website, our type is called blog and we choose the ID 123, then the index request looks like this:

```
PUT /website/blog/123
{
 "title": "My first blog entry",
 "text": "Just trying this out...",
 "date": "2014/01/01"
}
```

Elasticsearch responds as follows:

```
{
 "_index": "website",
 "_type": "blog",
 "_id": "123",
 "_version": 1,
 "created": true
}
```

The response indicates that the indexing request has been successfully created and includes the _index, _type and _id metadata and a new element: _version.

As we want to support full text search, so we need to index every part of log events; but at the same time, log events can be abandoned after a certain time. The ElasticSearch schema is designed as below.

```
{
"_default_": { "_ttl": { //Default TTL is 6 months.
"enabled": true,
"default": 7776000000
},
"_source": {
"enabled": false
},
"properties": { "env": { //whole env is used as term in Lucene. "type": "string",
"index": "not_analyzed"
},
    "eventbody": { //log event body is indexed. "type": "string"
    },
    "hostname": {
    "type": "string",
    "index": "not_analyzed"
    },
    "logfilename": {
    "type": "string",
    "index": "not_analyzed"
    },
    "logpath": {
    "type": "string",
    "index": "not_analyzed"
    },
    "logtype": {
    "type": "string",
    "index": "not_analyzed"
    },
    "timestamp": { //timestamp and nanotime are used for sorting "type": "long",
    }
    "ignore_malformed": false
    },
    "nanotime": {
    "type": "long",
    "ignore_malformed": false
    }
    }
    }
```

Test Environments: Because there is not enough finance to support our research, we created virtual machines based on our hardware.

Although commodity hardware can be enough for Hadoop, [14, 15] it's better to use more powerful server in production environment. It's necessary to use powerful hardware for ElasticSearch.

Table 1: Hardware test environment.

Elasitc Search Server	
NO of CPUs	4
CPU frequency	2.67 G
Memory	8G
HardDisk Size	100G
NO of ES servers	3
JVM Heap Size of ES	6G
shards	5
replica	0

Table 2: test result matrix

Total_ Events_ NO	Total_Matc hed_ Events	Wanted_LOG_ EVENTS	SearchTime (Seconds)
148928992	4375	25	6
148928992	4375	100	10
148928992	4375	500	19
148928992	4375	1500	34
148928992	4375	2000	26
148928992	4375	2500	51
148928992	4375	3000	63
148928992	4375	3500	46

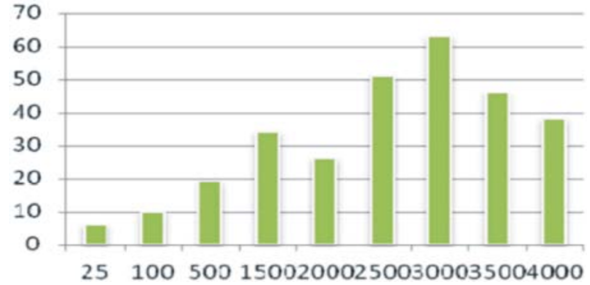

Fig. 2: Test result chart.

Test Result Matrix: During this experiment, we load enough log events for just one log file and caching [16] is disabled because we want to compared the real search performance. The log file size is 7GB and there are 148928992 log events. We want to search the keyword ¯bigdata?, the number of total matched log events are 4375. If we just want to get the first 25 matched log events, it will takes 6 seconds as you can see in Table 2.

Test Result Chart: Table 2 is visualized as below Figure. We can get some obvious conclusions based on the chart as in Figure 3.

CONCLUSIONS

According to the algorithm and experimental data, the paper is summarized as follows:

From the test result and the matrix we can get that Log events can be collected in real time by logstash; Log events can be indexed and searched out using ElasticSearch nearly in real time; With more log events it responds in a reasonable time, this is the feature we want; Of course if more powerful server are in place, the performance is definitely going better.

Elasticsearch Ecosystem and Lucene can be viewed as open source implementations of some of Google systems. We should keep an eye on the latest publications from Google, FaceBook and other internet company to get better idea regarding our goal.

In memory distributed DB and Index engine can be used together to do OLAP for big data. More and more matured products like SAP HANA are emerging in market.

ACKNOWLEDGMENT

First, I thank the technology developed, so that we can faster data processing, thanks to their predecessors for technological development efforts, thanked the experts put forward many valuable exchange of experience, without your support, I can not finish this article work.

Secondly, I want to thank my project guide of the CSE department Mrs.J.PREETHI, AP/CSE who guided me throughout this project.

REFERENCES

1. Yair Sovran, *et al.,* 2011. Transactional storage for Georeplicated systems?. Proc. of SOSP, pp: 385-400.
2. Dean, J., 2009. Challenges in Building Large-Scale Information Retrieval Systems: Invited Talk. In WSDM.
3. Michael McCandless, Eric Hatcher and Otis Gospodnetic, 0000.Lucene In Action? 2cd ed., Special Sales Department Manning Publications Co. 180 Broad St. Suite 1323 Stamford, CT 06901, pp. 2-110
4. Inc.–Dremel: Interactive Analysis of WebScale Datasets
5. David Smiley and Eric Pugh, 0000. Solr 1.4 Enterprise Search Server?, Published by Packt Publishing Ltd. 32 Lincoln Road Olton Birmingham, B27 6PA, K, pp: 280-281.
6. Schmuck, F. and R. Haskin, 2002. GPFS: A Shared-Disk File System for Large Computing Clusters,? Proceedings of FAST ’02: 1st Conference on File and Storage Technologies (USENIX Association, 2002), pp: 231-244.
7. Michael Stonebraker, *et al.,* 2007. The end of an Architectural era: (it’stime for a completerewrite)?. Proc. of VLDB., pp: 1150-1160.
8. Weil, S., S. Brandt, E. Miller, D. Long and C. Maltzahn, 2006. Ceph: A Scalable, High-Performance Distributed File System,? Proceedings of OSDI ’06: 7th Conference on Operating Systems Design and Implementation (USENIX Association, 2006).
9. Small, J. Zelenka and B. Zhou, ?Scalable Performance of the Panasas.
10. viktor mayer-schonberger and kenneth cukier, 2013. ¯ig data: a revolution that will transform how we live, work and think,? eamon dolan/houghton mifflin harcourt; 1 edition (march 5, 2013), pp: 28-69.
11. Lars George, 0000. HBase: The Definitive Guide?, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, pp: 41-73.
12. Abadi, D.J., P.A. Boncz and S. Harizopoulos, 2009. Column-Oriented Database Systems. VLDB, 2(2).
13. Database Systems. VLDB, 2(2), 2009.
14. Tom White, 0000. Hadoop: The Definitive Guide?, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, pp: 357-377.
15. Ashish Thusoo, *et al.,* 2010. Hive-A Petabyte Scale Data Warehouse Using Hadoop?. Proc. of ICDE., pp: 996-1005.
16. Parallel File System,□ Proceedings of FAST '08: 6th Conference on File and Storage Technologies (USENIX Association, 2008), pp: 17-33.