Feasibility Analysis of Big Log Data Real Time Search Based on Hbase and ElasticSearch

Jun Bai
Department of mechanical and electrical engineering
Northern BeiJing Vocational Education Institute
BeiJing, China

***Abstract*—**There are a lot of log events generated very day from modern enterprises. Form the pattern of these log data, enterprises can mine business value; it's even better if they can do it in real time. But managing this big log data is a big challenge because traditional technology is not powerful enough to process huge data. Luckily, Hadoop echo system provides a new way to process big data; ElasticSearch, which is based on Lucene, is the modern search engineen for cloud environment. This paper presents a real-time big data search method: First, Flume agent from the end user's machine collect log events, then ElasticSearch according to the search conditions are needed rowkey list; finally Hbase using these rowkey directly from the database to get the data, the paper-based hardware to create a virtual machine environment, the experiment proved, with the search for more log events, the search response time does not increase linearly. This article will explain Hbase based on modern distributed systems and cloud real-time search search engine ElasticSearch feasibility of large data logs.

Keywords ElasticSearch, Big Data, Hbase , real time search, Hadoop

I. 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 and other Computer information processing issues, and these major technological innovations Occurred in the Internet to provide consumers with services.

There are billions of log events generated everyday 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.

The rapid development of Internet, the amount of information to make the Internet an exponential growth. 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. 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.

HBase namely Hadoop Database, is a high-reliability, high performance, column-oriented, scalable distributed storage system, use HBase technology can be erected on a cheap PC Server massive structured storage cluster. HBase is open source implementation of Google Bigtable, similar to Google Bigtable use GFS as their file storage system, HBase use Hadoop HDFS as its file storage system; Google run MapReduce to handle massive amounts of data in Bigtable, HBase also use Hadoop MapReduce to deal in HBase massive data; Google Bigtable use Chubby as a collaborative service, HBase use Zookeeper as correspondence.

Hbase is a new emerging NOSQL database based on Hadoop. It's designed for big data real time access. But there is only one key in Hbase, the rowkey. It means there is no secondary index in Hbase. So when we want to search from Hbase, more data blocks will be load into memory then necessary.

Lucene is a full-text search and search for open source library that can do full-text indexing and search. In the Java development environment Lucene is a free open source tools mature. For its part, Lucene is currently and in recent years the most popular free Java information retrieval library.

ElasticSearch is a construct based on Lucene 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.

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 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.

Solr is an independent enterprise search application servers, which offer similar Web-service API interface. Users can through http requests must be submitted to the search engine server XML file format, generate an index; operation can also be made via Http Get lookup request and get back results in XML format;

Solr is a high-performance, using Java5 development, full-text search server based on Lucene. At the same time it was extended to provide richer than Lucene query language, while achieving a configurable, scalable and optimized query performance, and provides a complete function management interface, is a very good full-text search engine.

ElasticSearch is a construct based on Lucene 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.

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, 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 ElasticSearch cluster. The indexes can routine to any of the shards if you want to. And it's RESTful architecture.

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

Computation is about IO. Hard disk can be seen as modern cascade; Memory can be seen as modern hard disk. The most efficient way to do compute in modern system architecture is to use memory as much as possible. There are some in memory processing system which can do the OLAP thing for big data in real time such as SAP HANA.

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 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 is located in local file system; while in our experiment projects all log data are stored in hbase, only index itself in stored in local file system. It means more log data can be processed compared to Kibana. Our experiment is for big log data.

Cloudera Search is a new component for CDH; its core is SolrCloud. Cloudera Search is going to be used to provide indexing and searching ability for CDH and to be integrated with MapReduce, Flume and Hbase. All the index segments are stored in HDFS. HDFS itself can provide fault detection and recovery, and it's designed for big data. It means no matter whatever the index segments size is, they can be stored in HDFS. It's really for big log data management from the view of cloud computation. More research on Cloudera Search will continue after this paper.

Although it's not the purpose, 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.

II. SYSTEM DESIGN POINTS

Flume agent collects log events from end user machine

At flume collector side, Hbase plugin is used to feed log events into Hbase; 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.

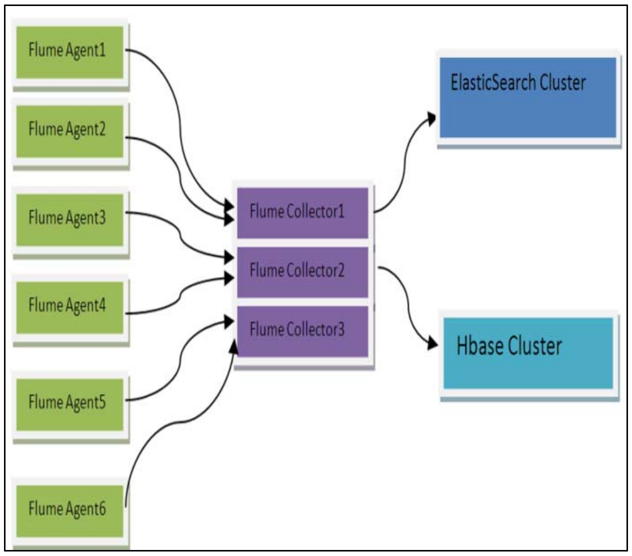

Figure 1: log events collecting and indexing flow

Search conditions are going to be committed to ElasticSearch cluster first; the rowkeys returned from ElasticSearch cluster are applied to Hbase;Hbase returns the log data by applying the rowkeys, illustrated as in Figure 2 .

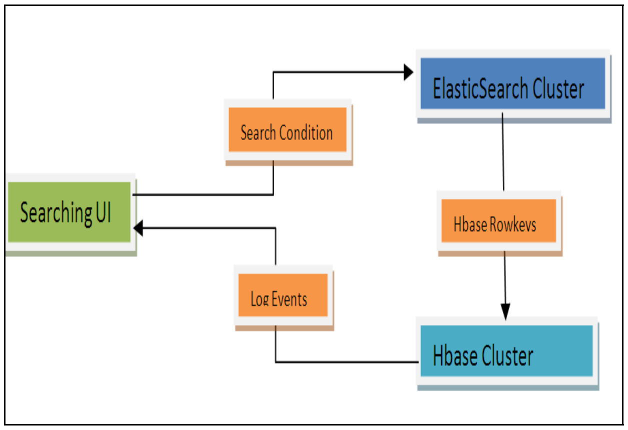

Figure 2: searching flow

Search condition includes: hostname, logfilepath, logfilename, logfiletype, env, keyword for eventbody, starttime and stoptime

III. DATA STRUCTURE AND ALGORITHM

Let's assume all the log evens are grouped by application. Due to survey from some company, SA prefers to view all the logs on a single screen.In order to give user a more friendly view, the log events should appear as the sequence they are generated. So hbase rowkey is designed as sequential rowkey. Sequential rowkey is suitable for sequential read as all the related rows are stored in the same block.

Hbase rowkey schema is designed as appid:timestamp:nanotime:hostname

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
                           }
                          }
                        }
                      }
```

IV. 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, it's better to use more powerful server in production environment. It's necessary to use powerful hardware for ElasticSearch. You can see the hHardware test environment in Table 1.

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

NO of DataNodes	3

Table 1: Hardware test environment.

A. *Test Result matrix*

During this experiment, we load enough log events for just one log file and caching 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.

Total_Events_NO	Total_Matched_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
148928992	4375	4000	38

Table 2: test result matrix

B. *Test Result Chart*

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

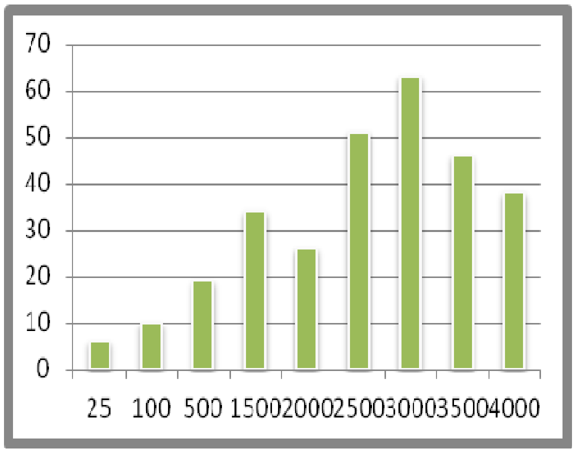

Figure 3: test result chart.

C. *Conclutions*

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 flume;Log events can be indexed and searched out using ElasticSearch nearly in real time; With more log events searched, the responge time does not increace lenearly; 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.
- As rowkey is the only key for Hbase, in some cases we can redesign the rowkey to get more distributed regions. Managing distributed cluster is not an easy job to do, we should utilize some SCM tool to mange the Hadoop cluster and ElasticSearch cluster. When more and more log events are collected, the index file will grow very large; searching across large index files is not possible in production. We can split index by date time awareneses.
- Hadoop 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 friend, the friend Zhang Yanping's help, I finished this article, I thank my good friend Mr. Zhang Yanping busy schedule, gave me a lot of technical guidance and advice, to thank him for I provide technical guidance and spiritual encouragement;

Again, I want to thank my husband, Mr. Yue Weijia, husband to help me take care of the family and children, so I have time to sit down to write this article, in my difficulties want to compromise, he encouraged me to persevere;

Finally, I want to thank my parents and children, parents good health, gave me a lot of comfort, the children grow up happy, give me their support, all of which, I can safely write papers the most important safeguard, I love my friends, I love my husband, I love my parents and children, I love you!

References

[1] Tom White, " Hadoop: The Definitive Guide" , Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, pp. 357–377.

[2] Lars George, " HBase: The Definitive Guide", Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, pp. 41–73.

[3] Michael McCandless, Eric Hatcher and Otis Gospodnetic, "Lucene In Action" 2cd ed., Special Sales Department Manning Publications Co. 180 Broad St. Suite 1323 Stamford, CT 06901, pp. 2-110

[4] David Smiley and Eric Pugh, "Solr 1.4 Enterprise Search Server", Published by Packt Publishing Ltd. 32 Lincoln Road Olton Birmingham, B27 6PA, UK, pp.280-281.

[5] Yair Sovran et al. "Transactional storage for geo-replicated systems". Proc. of SOSP. 2011, pp. 385–400.

[6] Michael Stonebraker et al. "The end of an architectural era: (it'stime for a complete rewrite)". Proc. of VLDB. 2007, pp. 1150–1160.

[7] Ashish Thusoo et al. "Hive — A Petabyte Scale Data WarehouseUsing Hadoop". Proc. of ICDE. 2010, pp. 996–1005.

[8] F. Schmuck and R. Haskin, "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.

[9] S. Weil, S. Brandt, E. Miller, D. Long, and C. Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," Proceedings of OSDI '06: 7th Conference on Operating Systems Design and Implementation (USENIX Association, 2006).

[10] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable Performance of the Panasas Parallel File System," Proceedings of FAST '08: 6th Conference on File and Storage Technologies (USENIX Association, 2008), pp. 17–33.

[11] VIKTOR MAYER-SCHONBERGER AND KENNETH CUKIER, "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

[12] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis Google, Inc. "Dremel: Interactive Analysis of WebScale Datasets"

[13] J. Dean. Challenges in Building Large-Scale Information Retrieval Systems: Invited Talk. In WSDM, 2009

[14] D. J. Abadi, P. A. Boncz, and S. Harizopoulos. Column-Oriented Database Systems. VLDB, 2(2), 2009

[15] SEAN OWEN, ROBIN ANIL, TED DUNNING and ELLEN FRIEDMAN, "Mahout in Action," Manning Publications Co. 20 Baldwin Road PO Box 261 Shelter Island, NY 11964 , pp.11-114

[16] David K. Gifford. Information Storage in a Decentralized Computer System. Tech. rep. CSL-81-8. PhD dissertation. Xerox PARC, July 1982. [17] Lisa lendenning et al. "Scalable consistency in Scatter". Proc.of SOSP. 2011.

[17] Jim Gray and Leslie Lamport. "Consensus on transaction commit".ACM TODS 31.1 (Mar. 2006), pp. 133–160.

[18] Pat Helland. "Life beyond Distributed Transactions: an Apostate's Opinion". Proc. of CIDR. 2007, pp. 132–141.

[19] Maurice P. Herlihy and Jeannette M. Wing. "Linearizability: acorrectness condition for concurrent objects". ACM TOPLAS12.3 (July 1990), pp. 463–492.

[20] Leslie Lamport. "The part-time parliament". ACM TOCS 16.2 (May 1998), pp. 133–169.

[21] Jeff Shute et al. "F1—The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business". Proc. of SIGMOD. May 2012, pp. 777–778.

[22] Leslie Lamport, Dahlia Malkhi, and Lidong Zhou. "Reconfiguring a state machine". SIGACT News 41.1 (Mar. 2010), pp. 63–73.

[22] Barbara Liskov. "Practical uses of synchronized clocks in distributed systems". Distrib. Comput. 6.4 (July 1993), pp. 211–219.

[23] David B. Lomet and Feifei Li. "Improving Transaction-Time DBMS Performance and Functionality". Proc. of ICDE (2009),pp. 581–591.

[24] Jacob R. Lorch et al. "The SMART way to migrate replicated stateful services". Proc. of EuroSys. 2006, pp. 103–115.

[25] Keith Marzullo and Susan Owicki. "Maintaining the time in adistributed system". Proc. of PODC. 1983, pp. 295–305.

[26] [Sergey Melnik et al. "Dremel: Interactive Analysis of Web-Scale Datasets". Proc. of VLDB. 2010, pp. 330–339.

[27] Andrew Pavlo et al. "A comparison of approaches to large-scale data analysis". Proc. of SIGMOD. 2009, pp. 165–178.

[28] Daniel Peng and Frank Dabek. "Large-scale incremental processing using distributed transactions and notifications". Proc.of OSDI. 2010, pp. 1–15.

[29] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. "System level concurrency control for distributed databasesystems". ACM TODS 3.2 (June 1978), pp. 178–198.

[30] Alexander Shraer et al. "Dynamic Reconfiguration of Primary/Backup Clusters". Proc. of USENIX ATC. 2012, pp. 425–438.