An Exploratory Survey of Hadoop Log Analysis Tools

Madhury Mohandas
DCS Rajagiri School of Engineering and Technology
Cochin, India

Dhanya P M
DCS Rajagiri School of Engineering and Technology
Cochin, India

ABSTRACT

In view of the fact that clusters used in large scale computing are on the rise, ensuring the wellbeing of these clusters is of paramount significance. This highlights the importance of supervising and monitoring the cluster. In this regard, many tools have been contributed that can efficiently monitor the Hadoop cluster. The majority of these tools congregates necessary information from each of the node in the cluster and takes it for processing. These diagnosis tools are mostly post execution analysis tools. This paper presents an exploratory assessment of the different log analyzers used for failure detection and monitoring in Hadoop.

General Terms

Failure Monitoring

Keywords

Cloud computing, HDFS, Failure monitoring, Hadoop, Log analyzer

1. INTRODUCTION

This is the era of BigData. Massive and gargantuan amount of data is produced on per day basis. Such scenario elevates the need for apposite storage, supervision and processing of data. Hadoop[1], an extensively used technology includes distributed storage and processing of data. This framework is currently employed in almost all companies to deal with data-intensive applications. Conventional infrastructures used for data processing prove to be less efficient in distributed processing of data. Thus there was a move for developing models for large scale distributed storage and processing. Architectures like shared nothing proved to work better with BigData primarily because of the speed of processing, storing and accessing data. Apache Hadoop [1] is one among the most widely used large scale data processing paradigm which is currently being employed in Facebook, Google, Amazon etc. The chief advantage with Hadoop is that it allows for the storage of data in any format. The massive use of this framework calls for the faster analysis and diagnosis of failures. Due to the distributed nature of processing, it's difficult for cluster administrator to isolate the failures and failed nodes. Many contributions have been done for failure monitoring, analysis etc in the last few years.

2. HADOOP AND STORAGE

2.1 Framework

Hadoop[2] is gaining popularity mainly because of its ability to manage huge amount of data. The clusters used for data storage and processing may vary from a single server to a group of machines, generally, commodity machines. Though there are some limitations with single namenode [3], this framework is widely used. This software framework supports the MapReduce programming model for performing the distributed processing of data stored on the cluster. Hadoop Distributed File System (HDFS) is the default storage layer given by the Hadoop framework and for this reason the layer is used by all applications that run on this Hadoop framework. This layer got its structure from Google File System (GFS) [4] and the whole framework depends on the efficiency of HDFS. HDFS [5] layer has master-slave architecture and was designed for reliable storage of data [6]. This architecture consists of two types of nodes in a cluster specifically the NameNode and DataNodes. NameNode responsibility is to act like a master and DataNodes takes up the role of slaves. The execution flow is shown in figure 1.

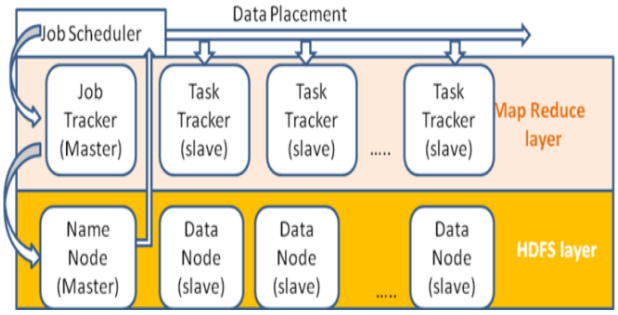

Figure 1: Flow of execution in cluster

2.2 MapReduce Layer

Just as the name indicates, the MapReduce [7] is a software framework that supports massive computations, and is based on two basic steps: Map and Reduce. It is a programming model and has an implementation associated with it. The runtime system is a helping hand for the programmers, in view of the fact that it take care of all background details like input data partitioning, fault tolerance, task schedules etc. in an environment having huge number of machine and large data amounts for computation, MapReduce becomes an aid. The master choose some slaves to perform the map task and others to perform the reduce task [7]. Those who have to do the map task will read input from the corresponding file and perform the map task, storing the intermediate data in the memory. These intermediate data will be periodically transferred to the local disk associated with the map workers and the location details will be passed to the master node. The master node then notifies the reducer about this intermediate result and then using RPC, they access the corresponding local disk of map workers to get the intermediate data. The reduce workers produce the output

files. After all map and reduce task are completed, the master then wake up the user program.

3. LOG ANALYZERS

Since the big data is ever increasing (without any signs of decrease), the clusters dealing with this big data need to be monitored and maintained efficiently. Log analysis can aid in optimizing the overall system performance since realizing a system's performance is correlated with system resource usage understanding. Though there are many subprojects of Apache Hadoop including scribe [8] [9], Chukwa [10], Gridmix3 [11] etc, the analysis in this paper also includes some of the contributions.

3.1 Vaidya

Vaidya [12], a very helpful tool for performance analysis of the applications running on the Hadoop cluster, is one of the contributions to Hadoop framework. This tool enables the cluster administrator in spotting jobs with deteriorating performance. Vaidya analyses the application's performance through a set of diagnostic rules; which can be written by the user according to the application. In each diagnostic test rule, the importance, threshold and prescription can be detailed. Importance specifies the general significance of the test. The prescription specifies the intended advice to improve the performance of the particular job. The test report consists of test name, importance of test, description of diagnostic test, severity, result of the test and the prescription. In the default version of Vaidya, it incorporates five test rules [13]. A limitation with this tool is the difficulty in copping up with the consistency of rules against continuously increasing Hadoop code base.

3.2 Salsa & Mochi

Salsa [14] examines the Hadoop logs (Data Node logs and Task Tracker logs) to outline the data and control flow execution and presents with a state machine view of the execution on each node. Since a map reduce job includes logging statements [15], each activity will release log messages. In order to model the control flow, every log message is considered as an event which can be either a start or end state of execution. The identification of events from logs is by the use of pattern recognition. Salsa also tries to incorporate the fault detection and diagnosis of job MR job executions. This approach does not require the modification of operating system, middleware or the application. Hadoop debugging can also be done using Mochi, a log based analysis tool for Hadoop. Mochi [16] relates the behavior of execution in time, space and volume and also incorporates the interdependencies in the distributed environment. It first extracts the Job execution view per node by executing the Map Reduce nob and then constructs a Job Centric Data Flow (JCDF) which is a directed graph, by relating the collected execution views on each node and also with the HDFS layer. The JCDF is also distributed among nodes. Thus it tries to automatically generate and correlate the data and control flow between nodes and then analyze the Hadoop behavior. Mochi provides visualization to reason and debug performance issues by the users. The visualizations include: MIROS, REP and Swimlanes. MIROS captures the data flow on each node among the maps and reduces. REP check that states which process larger volumes of data should take larger time in processing. Swimlanes for capture task progress and shows the duration of each task.

3.3 Chukwa

Yet another subproject to Hadoop is Chukwa [10], similar to Ganglia [17] in the storage aspect. It is a data collection and network management system [18] that is built on top of Hadoop ie, Hadoop distributed File System and Map Reduce framework and so, is scalable. The storage of the logged data is done on the distributed file system of Hadoop, HDFS unlike other tools that store them on local storage. HDFS has high throughput and is highly flexible, efficient and reliable for storage of large files. Chukwa give support to failure diagnosis by continuous monitoring of the system. The basic architecture of Chukwa [20] consists of adaptor, agent, collector and HDFS storage. Chukwa collects system metrics, log files, arbitrary log files, logs from X-trace [19] etc. these data are collected by adaptors and are stored in HDFS as large files, since HDFS is efficient for operations on large files rather than small ones. For the same reason, the collector and agents are added between the adaptors and HDFS layer. Thus, this tool highly aids in the storage of outsized data and processing of these chunks.

4. ANALYTICAL STUDY

An epigrammatic comparison of the above discussed log analyzers are shown in table 1. Vaidya does a post execution log analysis and mainly focus on application failure diagnosis rather than hardware and network failure. The range of failure detection and diagnosis by Vaidya can be improved by incorporating further test rules. But this depends on the application to be monitored, and so the preciseness of failure diagnosis by Vaidya is mostly in the hands of the user. As discussed in section 3, the Salsa and Mochi identify the states using pattern recognition. But unfortunately when the log patterns have to be changed (as part of any source code modification) the patterns may not be easily identified by use of pattern matching. It can capture the task dependencies efficiently, as each task are depicted as states and the transition from one state to another shows the dependency between the two. Since the time aspect is considered, extra care has to be given for clock synchronization. Chukwa on the other hand takes as input all possible sources of information from various sources and monitors the system as a whole. It models a data collection and network supervision system for hadoop.

Table 1: Analytical Study of Log Analyzers

Log Analyzer	Feature	Input	Output	Limitation	Future Enhancements
Vaidya [12]	Rule supported scrutiny on logs Does rule based assessment for every job execution For every problem diagnosed it gives intended guidance	Job History log Job Configuration log (XML)	XML report of evaluated result	Worthless outcome, proviso parameters are improperly placed	Online progress analysis Incorporate additional rules to capture all aspects of failure
Salsa [14]	State Machine outlook of the execution of job in an individual node Portrays control flow and data flow execution incorporates semantic information for analysis	Datanode Log Tasktracker Log	State machine with inter dependencies highlighted	Synchronization of clocks in cluster	Automate visualization of analysis generalize format and structure of logs for enhanced analysis Correlate network logs for failure diagnosis
Mochi [16]	Hadoop performance analysis in time, space and volume Spotlights the execution of job in distributed environment	Datanode Log Tasktracker Log	Number of nodes, jobs, task per job, their progress, duration & interaction among jobs	Synchronization of clocks in cluster	Online Mochi analysis Regression testing on programs
Chukwa[10]	Distributed gathering of logs and hasty analysis Large scale storage (on HDFS) and processing of data in pipelined manner (using MapReduce)	All Hadoop Logs System metrics Application metrics	Toolkit that demonstrate analyzed and monitored results	Being HDFS as the default storage, has limited throughput during heavy access concurrency to same file	HICC graph accuracy improvement for machines with local time zone

5. CONCLUSION

This era of BigData calls for the debugging, performance monitoring, storage and operation monitoring of data since the data is increasing exponentially without any signs of dwindling. Consequently a dreadful need for scalable and reliable scheme for the same is felt which resulted in the contribution of many tools for monitoring and diagnosis of huge clusters. Hadoop being the most popularly used methodology for storage and processing of BigData, has several subprojects for failure monitoring and analysis. The majority of these tools seize the log files to capture the behavior of the cluster and the running application. They process the logs to retrieve the necessary information required for failure diagnosis, and some of the tools even support the failure recovery. An expository survey of some of these log analyzers shows that most tools try to capture only the application failure diagnosis aspect ignoring the hardware and network failures. In such clusters, the failures are not an exception and so diagnosis of failures must be extended to all possible levels of failures ranging from node failure to application failures.

6. REFERENCES

[1] Hadoop, http://hadoop.apache.org/.

[2] W. Tom, Hadoop:the definitive guide(O'reilly media, May 2009)

[3] K. Shvachko, Hdfs scalability: The limits to growth, The USENIX Magazine , 35(2), 2010

[4] S. Ghemawat, H. Gobioff, and Leung, "The Google File System," SIGOPS Oper. Syst. Rev., 37(5):29–43, 2003

[5] D. Borthakur, HDFS Architecture, http://hadoop. apache. org/common/ docs/r0.20.0/ hdfs_design.html, April 2009

[6] K. Shvachko, H. Huang, S. Radia, and R. Chansler, The hadoop distributed file system, In 26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies, May 2010.

[7] J. Dean and S. Ghemawat, Mapreduce: simplified data processing on large clusters, In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, pages 10–10, Berkeley, CA, USA, 2004.

[8] Scribe, https://github.com/facebook/scribe.

[9] Scribe logfile aggregation system described by Facebook's Jeff Hammerbacher https://issues.apache.org/jira/browse/HADOOP-2206?focusedCommentId=12542775#action 12542775

[10] Chukwa, http://wiki.apache.org/hadoop/Chukwa

[11] Gridmix3 – Emulating Production Workload for Apache Hadoop, www.usenix.org/conference/fast-10/gridmix3-emulating-production-io-workload-apache-hadoop

[12] Vaidya, http://hadoop.apache.org/docs/stable/vaidya.html

[13] Revisiting the physician : Hadoop Vaidya, http://www.hadoopsphere.com/2013/01/revisiting-physician-hadoop-vaidya.html

[14] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, Salsa: Analyzing logs as state machines, In Workshop on Analysis of System Logs, San Diego, CA, Dec 2008.

[15] Log4J, http://logging.apache.org/log4j, 2007

[16] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, Mochi: visual log-analysis based tools for debugging hadoop, In Proceedings of the 2009 conference on Hottopics in cloud computing, HotCloud'09, Berkeley, CA, USA, 2009.

[17] Matthew L. Massie, Brent N. Chun, and David E.Culler, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, In Parallel Computing Volume 30, Issue 7, pp 817-840, 2004

[18] J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang, and M. Yang, Chukwa, a large-scale monitoring system, In First Workshop on Cloud Computing and its Applications (CCA '08), Chicago, IL, 2008

[19] Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica, X-Trace: A Pervasive Network Tracing Framework, In 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07), Cambridge, MA, USA, April 2007

[20] A. Rabkin, R Katz, Chukwa: a system for reliable large-scale log collection, In Proceedings of the 24th International Conference on Large Installation System Administration LISA'10, USENIX Association Berkeley, CA, USA.