OpenStack Log Analysis and Visualization

Dr. Dinkar Sitaram, Asso. Prof. Satya Srinivas
PES University
Bangalore, India
{dinkars, satyasrinivas}@pes.edu

Sanjana Rajagopala, Nisha Alva, Sahana Rao
PES Institute of Technology
Bangalore, India
{alvanisha999, sahana093, sanjanarajapoal93}@gmail.com

***Abstract*— System logs are rich sources of information and the more neglected one. The analysis of logs serves many purposes such as debugging, understanding performance of the system, predicting resource usage, profiling, etc. The log generated in OpenStack environment is voluminous and highly unstructured. It is humanly impossible for administrator or developer to manually skim through the logs and extract significant inferences from it. Also, the regular expressions to filter out segments of our choice from the logs are extremely complex, are difficult to write and support. Hence, an automated approach that generates user-friendly visualization of the log analysis and help detect anomalies in the system. Our paper aims at visualizing the logs generated in OpenStack environment by providing a user interface through which an administrator or a developer may upload log files that are to analyzed, for the two modules – Nova, Neutron and Cinder and do in-depth offline analysis to find and isolate anomalies. Our paper treats log analysis as a Big Data problem, owing to the nature of log files in an actual production environment and hence leverage Big Data tools and techniques to efficiently carry out the task.**

Keywords*— *System Logs; Cloud Computing; Big Data; OpenStack; Nova; Neutron; Cinder; Data Visualization; Hadoop; MapReduce; DbScan

I. INTRODUCTION

Logging is important in all computer systems, but the more complex the system, more difficult it shall be to spot failures and cut down on troubleshooting time. Understanding logging in OpenStack is important to make sure your environment is healthy, is able to isolate the problem by narrowing down on the log files and to send relevant log entries back to the community to help fix bugs. Since there are many interdependent software modules, creating huge volumes of log files, it is also very difficult to rely on earlier knowledge and experience to isolate the problem, as it is time-consuming and extremely tedious. It is also extremely important to note that the computing environment is not a simple one to deal with it manually.

System logs generated by OpenStack services are rich sources of information. Tracking a failure in OpenStack becomes difficult especially in the absence of a tool to make it easier to look through the log files, which is complicated by many interdependent components. To analyze the log files, we have to consider the interdependency of various system components in the OpenStack environment. These interdependent system components log various events in to their own log files at varying times, in huge volume. These various independent log files have to be treated as different data sources with varying formats and messages needs to be collected, integrated, organized and then analyzed to isolate the points of failure.

For automated analysis of these temporal log data across various interdepdent OpenStack components, a common message body needs to be created. Second, text mining of the descriptions of the log messages needs to be conducted. Then, each of the messages needs to looked for a common timestamp based on which the time-series data is aggregated among various OpenStack components to detect failures or anomalies or outliers.

II. BACKGROUND WORK

Actual OpenStack production environment has lot of moving parts and tracking down an event of failure becomes difficult. OpenStack log analysis is a big data problem as humongous amounts of data generated as a part of logging. The size of the data itself becomes a part of the problem. Extracting substantial inferences out of these logs is difficult without breaking the problem at hand in to the following steps:

- Log aggregation tools and their comparison
- Log indexing tools and their comparison
- Visualization tools and their comparison
- Distributed storage and their comparison

The above establishes a need to build a Big Data framework to collect these voluminous logs and give utilities to visualize, Index and search the logs, identify flows, detect anomalies and pin point a probable cause for failure.

III. PROPOSED METHOD

A. *Proposed solution outline*

We propose an approach for log analysis where we employ temporal, dynamic and linear logic to model the events and traces of OpenStack logs. After aggregating and indexing the

logs, the logs are dumped and stored in HDFS for analytics. The following is an enumeration of the basic stages of log processing for analysis and visualization:

- **Log Analysis:** Logs stored in HDFS are read by Map Reduce jobs to extract and group into key-value pairs, which are then fed into dbscan clustering algorithm to identify the flows and report anomalies.

- **Log Visualization:** To gain an insight through the OpenStack logs, the results of analysis are presented in the form of tables and charts.

1) Log Analysis

After log aggregation and indexing, the raw logs are first fed into the mapper M1 of the first map reduce job. M1 converts raw logs into key value pairs with Request Id as the key and message as the value and outputs into a HDFS file. This output with Request Id as the key and message as the value fed as input to the mapper M2 of the second Map-Reduce job. M2 matches the log lines against message signatures from the parsed sourced code and replaces them with the corresponding unique IDs along with the timestamp. It outputs Key-Value pairs, with Request ID as the key and UID + Timestamp as the value which is fed to its reducer R2.

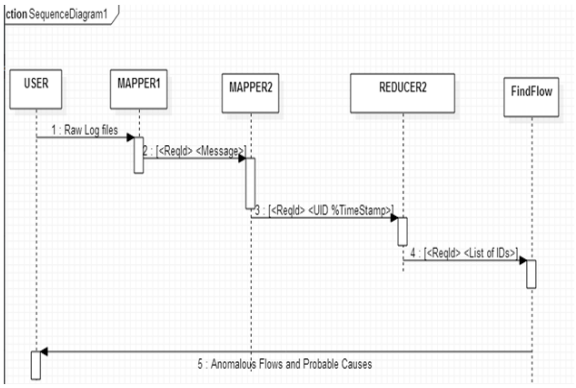

Fig 1: MapReduce Jobs

R2 groups all the messages based on Request IDs. R2 outputs Key-Value pairs, with Request ID as the key and the value as a list containing all related <UID + TSP> pairs. The output of R2 is finally fed to dbscan clustering algorithm that identifies flows, finds out erroneous flows, compares erroneous flows with a reference file containing a good run of the corresponding use case and finds out stages successfully completed along with the anomaly in the sequence.

2) Log Visualization

Apart from log analysis, the log visualization provides administrator and/or developer an ability to look at the causes of failure by showing the failure that is easy to understand.

- Use-Case visualization in the form of graph:

  A Line Chart of request ID versus timestamp is plotted. The erroneous data obtained as the output of clustering is used to visualize the error so that the user can determine the exact source of error. On hovering over the points in the line chart, log message associated with the request id.

- Table View:

  The log file of OpenStack module -Cinder or Neutron or Nova, consists of information like request ID, log level, log message, timestamp, location where the error has occurred and various other details. This information is helpful in determining the workflow and source of error in the OpenStack system. The error information is extracted from the log file and represented in the form of table so that if the user is aware of the probable error. A search box is incorporated so that user can search for specific errors in the system.

- Line Chart (Flow identification):

  The flow identification line chart is implemented so that the user can understand the workflow of the module – Cinder or Neutron or Nova. In this line chart, message ID is plotted against timestamp. The log messages contained in the log file are taken as input into the Map-reduce Job 1 and unique ID is generated for each of them. This is plotted against the timestamp thereby depicting the way the module works. The "search" feature is added so that the user can search for specific keywords in the log message. This helps to determine the time at which the task happened and the order in which the various tasks take place and hence, come to a conclusion about anomaly in the flow.

IV. EXPERIMENTAL DATA AND RESULTS

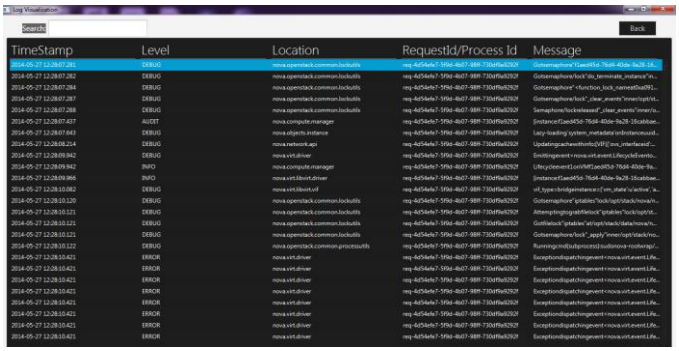

Fig 2: Table View of log Messages

Fig 2: OpenStack modules Use case view

Fig 3: Step-by-step Workflow view

V. Inference

The use case of OpenStack launch Instance was executed, and an error was forcefully introduced by deliberately trying to allocate more memory than any of the hosts had. The corresponding logs were analyzed using the proposed solution and the following observations were made:

- The log messages were correctly identified as belonging to Launch Instance Use case
- An anomaly was detected in the sequence
- Results showed that the following stages were successfully completed:
  - Allocating resources
  - Ensuring static filter
  - Filtering
- From the above results we can infer that the stage where a Host is chosen for spawning the instance was not executed.
- This is justified as there was no host satisfying the requested resources.
- Hence the reason displayed for failure was: Host Fails.

VI. Conclusion

The OpenStack logs have humongous data, which reveals the details about the workflow and statistics of the OpenStack system. The Administrator and/or Operator find it tedious, time consuming and difficult to manually go through this and make sense of the messages in the log files. We have shown successfully that the user can isolate the failure point by extracting the useful attributes (for example: log level, log message, timestamp, place of error, request id) from the log files and display it in a tabular manner, mapping the request ids with the associated message ids and providing the visualization in the form of line chart and depicting the source of error in the graph.

The user-friendly visualization of the workflow of OpenStack modules achieved by showing the line chart with all the use cases associated in the log file with colors (each color representing one use case) and allowing the user to zoom-in and zoom-out to have clear view of the log messages involved in the use case and to view the source of error by hovering over the red points in the graph. If the user is aware of the probable error and is not particular about graph visualization then he/she can opt for tabular representation and analyze the table to get details about the error. The search box included so that user can search for specific errors in the system.

Therefore, we can conclude that user-friendly visualization of the OpenStack logs to isolate the failure point and error message thereby reducing the effort that a user must put to extract required information from the logs and find the source of error.

VII. Future Work

The work can be extended to provide real-time analysis of OpenStack Logs. We would like to conclude as:

1. We can extend the solution for real-time analysis of OpenStack logs where the user can visualize the log files as and when the OpenStack is running.
2. This work can also be extended to other modules of OpenStack that record request id in the log file.
3. The resource utilization and other useful statistics regarding OpenStack modules can also be shown.

Acknowledgment

We extend our gratitude to Prof. Satya Srinivas for his constant guidance, encouragement, support and invaluable advice. Also, our sincere thanks to Prof. M. R. Doreswamy, PESIT founder, Prof. D. Jawahar, CEO and Dr. K. S. Sridhar, Principal for providing us with a congenial environment for carrying out the project.

References

[1.] Apache Kafta. http://kafka.apache.org/documentation.html#introduction
[2.] Facebook Scribe,. http://en.wikipedia.org/wiki/Scribe_(log_server)
[3.] Apache Chukwa. https://chukwa.apache.org/
[4.] Apache Flume. http://flume.apache.org/

[5.] Apache Flume. https://groups.google.com/a/cloudera.org/forum/#!topic/flume-user/BMtsyGedcPo
[6.] Flume versus Scribe. http://nosql.mypopescu.com/post/820711193/how-does-flume-and-scribe-compare
[7.] Solr vs. ElasticSearch, http://www.searchblox.com/solr-vs-elasticsearch
[8.] ElasticSearch. http://www.elasticsearch.org/
[9.] ElasticSearch. http://en.wikipedia.org/wiki/Elasticsearch
[10.] Apache Solr. http://lucene.apache.org/solr/
[11.] Solr vs. ElasticSearch. http://thinkbiganalytics.com/solr-vs-elastic-search/
[12.] Hadoop. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[13.] Hbase. http://hbase.apache.org/
[14.] Hbase. http://en.wikipedia.org/wiki/Apache_HBase
[15.] MapReduce. http://en.wikipedia.org/wiki/MapReduce
[16.] Detecting Large-Scale System Problems by Mining Console Logs, W Xu, L Huang, A Fox, D Patterson - Proceedings of the ACM , 2009
[17.] Extracting the Textual and Temporal Structure of Supercomputing Logs, International Conference IEEE, Jain, S., Singh, I., Chandra, A, Zhi-Li Zhang, High Performance computing (HiPC) IEEE Conference
[18.] Mining Invariants from Console Logs for System Problem Detection, JG Lou, Q Fu, S Yang, Y Xu, J Li - USENIX Annual Technical Conference, 2010
[19.] Server Log. http://en.wikipedia.org/wiki/Server_log
[20.] Common Log Format, http://en.wikipedia.org/wiki/Common_Log_Format
[21.] Advances and Challenges in Log Analysis, A Oliner, A Ganapathi, W Xu - Communications of the ACM, 2012
[22.] OpenStack, https://www.openstack.org/
[23.] OpenStack, http://en.wikipedia.org/wiki/OpenStack
[24.] OpenStack Architecture, http://docs.openstack.org/training-guides/content/module001-ch004-openstack-architecture.html
[25.] Logging and Monitoring, http://docs.openstack.org/trunk/openstack-ops/content/logging_monitoring.html
[26.] Cloud Computing, http://en.wikipedia.org/wiki/Cloud_computing
[27.] http://www.getfilecloud.com/blog/2014/02/a-game-of-stacks-openstack-vs- cloudstack/#.U5D1DpzO_jA
[28.] http://docs.openstack.org/training-guides/content/module001-ch010-vm-provisioning-indepth.html