Cornell Log Analysis and Monitoring Project (CLAMP)

Michael D. Padula
Cornell University
Center for Advanced Computing
Ithaca, NY 14853
(607) 254-8733
Mdp15@cornell.edu

Lucia M. Walle
Cornell University
Center for Advanced Computing
Ithaca, NY 14853
(607) 254-8775
Lmw25@cornell.edu

ABSTRACT
The goal of this project was to determine if a real-time event monitoring process could enable the prediction or prevention of system failures in a complex system such as the Texas Advanced Computing Center's (TACC) Ranger compute cluster. The development process involved numerous research steps including system log collection and manipulation and the development, use and testing of multiple tools to analyze or display the log information in a meaningful way assuming that historical events are useful for future predictions. This paper includes information about related work, implementation details about the tools used, their purpose and limitations, an evaluation of the work, achievements, and proposed future work. Over the course of this project we were able to predict known events and minimize outages in various scenarios involving multiple distributed systems and subsystems.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information filtering, Query formulation, Selection process.*

General Terms
Management, Reliability, Experimentation.

Keywords
Realtime, log, monitoring, analysis, improve, reliability, prediction, event

1. INTRODUCTION
The goal of this project was to determine if a real-time event monitoring process could enable the prediction or prevention of system failures in a complex system such as TACC's Ranger compute cluster. On today's growing petascale computing systems the volume of log data that is produced typically exceeds many millions of lines of information every day from multiple sources (e.g., parallel file systems, networking equipment, and computer hardware) that make up the overall system. Human beings cannot read or analyze the extreme volume of log data fast enough to predict or prevent failures, much less to find the root cause of a problem that has already occurred. In order to maintain a reasonable level of availability and reliability, system administrators require software tools that can alert them to emerging conditions and errors and, in some cases, take preventative action.

Previous efforts have centered on a fully automated approach to predicting events. After reviewing this work, referenced in the subsequent section on Related Work, and noting inherent limitations in such fully automated approaches, our approach is unique in recognizing that some log files lack sufficient information to automatically predict failures. For example, syslog files from Ranger lack a proper date thereby imposing limits on any analysis or real-time monitoring. Additionally, some repeating events are logged as aggregates. More specifically, if an event occurs every second for 90 seconds one message will be logged stating that "the previous message was logged 90 times" without providing the actual message and without a reproducible timeline. We found that by including historical log analysis and institutional knowledge of the systems provided by TACC staff, we were able to minimize the limitations imposed by the inconsistent and sometimes spotty event logging and improve our ability to detect and prevent known conditions.

Our implementation included a set of tools, some purpose-built and others long in wide-spread use, used to visually represent log file activity, monitor log files in real time, discover potentially important or unique words, and generate a set of statistics from a given log file. Each of these tools, their purpose, and limitations are discussed in the subsequent Implementation section. The remainder of the paper includes a discussion of our evaluation of the work, specific achievements, and potential future work.

2. RELATED WORK
Many researchers and practitioners have addressed the issue of event log monitoring and analysis. There are numerous papers and tools available. Risto Vaarandi [1] has published numerous papers on event correlation and tools used for analysis. LogHound is incorporated in Sisyphus [2] which we installed, configured, and utilized. Jon Stearley [3] of Sandia National Laboratories who provides the Sisyphus toolkit has also written many papers and tools for log analysis. As computer systems change and evolve over time, so do analysis techniques. By appropriately utilizing these toolsets, gaining necessary insight, we investigated and created a process to effectively analyze and monitor log files from a wide variety of sources.

3. IMPLEMENTATION
Our implementation process evolved over time with numerous types of investigations. We will explain in this section the tools

we used or created. Briefly, this includes data collection and manipulation, the creation of a log activity graph, implementing tools such as Cayuga [4] and Sisyphus, creating graphs in Microsoft Excel, looking at tacc_stats, and evaluating SQLstream. Much of the legwork took place in parallel and this does not represent a sequential process through which data moves, but rather, a set of tools we employed to discover needed information about the logs. We also needed to learn more about the computer systems on which the events take place. While "High Performance Computing" can mean many things, we had to learn about the components that made up a particular supercomputer and understand the working relationship among these.

We began by collecting data from the logs generated by Ranger (http://www.tacc.utexas.edu/resources/hpc/#constellation) at TACC. Each week the system administrators at TACC sent us a list of newly created files. Five log types were made available to the Cornell University Center for Advanced Computing (CAC) from TACC and copied via Globus Online (https://www.globusonline.org) including syslogs, node cluster status, OSM or IB switch, job scheduler, and resource manager logs. We chose to download these via a manual process, rather than an automated one, in order to speed up our progress to the next steps in the project. In order to automate this step we would need TACC to provide us the means to poll for a list of log files on disk and their locations and develop the ability to reconcile that list against what had already been processed. Developing that ability would necessitate database development and programming to track files and their state in the pipeline or to implement a solution that provided the same capability. This would have been a substantial amount of work which was peripheral to the goals of the project. The work would have included significant effort from TACC and CAC staff. We opted instead to leverage the automated data processing and loading steps by simply reloading data when errors (missing or duplicate data) were encountered. In all, our database held nearly three years of logs from Ranger. An early challenge was gaining sufficient understanding of the log formats and making the content of each log suitable for entry into a relational database. For instance, one log contains a date entry without a year. With an understanding of the structure of each of the logs drawn from institutional knowledge at TACC and research at CAC, we were able to devise processes to shape the data in the manner necessary. With the initial data transformation phase complete, Ranger logs were then inserted in the log repository. This process was done using SQL Server Integration Services (SSIS). One particularly useful feature of an SSIS package is the data validation that can be embedded in the integration package. This prevents partial data inserts and disallows the insertion of erroneous data without relying exclusively on the database server to perform the validation. SSIS packages are also simple to automate and can be paired with the early data manipulation for a one click process capable of processing and inserting any number of log entries and log files. While analyzing the data we found that somehow duplicate data had been read in. The data then had to be reloaded implementing new steps to verify no duplicate data would be loaded into the database in the future. This duplicate data appears to be an unavoidable attribute of log file aggregation at present. This is part of the processes we are not involved in and unable to affect but it appears that during these steps the same log entries are getting placed in multiple log files resulting in duplicate data in a very small number of cases.

3.1 TOOLS INVESTIGATED

3.1.1 Log Activity Graph

The log activity graph was developed by CAC to assist with the initial analysis of the event logs by searching for patterns to identify failures, outages, and other potentially unknown conditions. This web-based tool allows an interested party to visually identify areas of potential interest in event logs based on the number of events logged over time. Having identified an area or areas of interest, further examination and analysis can be focused more effectively on a smaller subset of data. All event log entries in the repository share some common schema elements which allow this tool to identify points of interest in any of the log formats available. The data being used to generate the display can be exported in multiple formats for custom analysis or visualization. The raw data is not made accessible through this tool. Testing with this Adobe Flash-based tool revealed that using more than 2000 data points to generate the graph yielded unacceptable performance where as any given time span may contain millions of data points. Code was added to the database to 'bin' the data from a selected time span into 2000 data points. This yields good performance and protects the potentially sensitive underlying data while still making the results accessible for further study and analysis. For CAC staff working with this tool, it was the first step in identifying sections of the log file containing tens and hundreds of thousands of consecutive login failures.

3.1.2 Cayuga

Cayuga (http://www.cs.cornell.edu/bigreddata/cayuga/) is a stateful publish/subscribe application that can be used for an event monitoring framework and a variety of different types of events. Cayuga can accept data from files or streams. Stream processing makes it possible for software applications to access the log files as fast as they are written. For any hope of real-time event monitoring the capability of streaming is very important. Using files we can fine tune the monitoring of the logs and replay them for testing and evaluation.

As with all software applications there are some learning curves in setting up the application and utilizing it. CAC set up a Cayuga server and developed a client to read data from the SQL database and send it to Cayuga over a socket. All traffic was internal to CAC subnets within Cornell. Cayuga uses its own query language (Cayuga Event Language) to gather specified information as it comes through the socket. We set up some simple queries for testing. During testing the system processed more than one million records in about eight minutes. We then discovered that we needed to know more information about the actual data to make better queries and alerts for recognizing possible scenarios which would help alert a system administrator to an impending problem. The Cayuga download is available at http://sourceforge.net/projects/cayuga/.

3.1.3 Sisyphus

Sisyphus (http://www.cs.sandia.gov/~jrstear/sisyphus/) is a log file data mining toolkit that can be used with live data as it is generated and collected in the log files or it can be used on historical data. This tool gives color-coded information based on word counts which is helpful in analyzing large datasets of log files. This tool requires the datasets in original syslog form, so it will only work on one years' worth of data as the year is not kept in the syslog log files. We separated the logs by year and system

type, in this case, fileservers, logins nodes, and compute nodes to help gain a greater focus in our analysis. With this tool a user can look at specific time slices of data. We started by looking at unplanned down times in the historical data. The down times needed to be gathered manually from the TeraGrid web site as they were not specifically designated in any of the data we had access to where the log files were downloaded. Sisyphus also helps to identify the most (and least) unique words and phrases in files or areas of files. This is one of Sisyphus' most attractive features as both types of areas may be of interest in the context of log file analysis.

3.1.4 Generated Graphs

CAC developed a Windows PowerShell script (http://technet.microsoft.com/en-us/library/bb978526.aspx) to pull data from the database into Microsoft Excel so that graphs could be generated by counting the events from the different types of systems. When looking for patterns it is helpful to look at the data in different ways or time slices. The script is set up to ask for a start date, end date and increment number or bin size. We then looked at multiple graphs to see if there are any indicators for the frequency of events on different systems. For instance, utilizing these graphs we can easily see to what degree, relative to each other, the fileservers and compute nodes' logging patterns change and, consequently, discern a normal pattern of events from an abnormal pattern. We generated numerous graphs and used the information as input into other tools like Sisyphus to get a look at word counts and messages around various unexpected outages to see if we could find a trigger for these unexpected events. Unfortunately a trigger was not found in the logs that we evaluated.

3.1.5 tacc_stats

Tacc_stats (https://github.com/TACCProjects/tacc_stats) is a job-orientated system performance monitor. We were interested in the data this tool produces in the context of the potential to discover correlations between log file activity or specific logging conditions and a wide range of performance statistics including memory use, CPU utilization, and I/O statistics. Several days of tacc_stats output files were copied to CAC for this purpose. A procedure to put the data into a format suitable for a relational database was devised and the data was processed and reorganized using CAC's compute cluster in an embarrassingly parallel job which used tacc_stats output files as input and yielded the same statistics in a format suitable to easily and quickly place into a relational database for analysis. The data was placed into a simple relational database and a series of queries devised to generate basic information that described various performance attributes of jobs which started, ran, or finished in the period of time represented in our sample.

3.1.6 SQLstream

SQLstream (http://www.sqlstream.com/) is a tool much like Cayuga in that one can query data through time whether it be live or historical. However, the formats of the queries are composed using standard SQL, not a proprietary language. Since this is a commercial product, support is available if users have questions or need assistance. CAC set up SQLstream and used it with the historical data from TACC. We played back the TACC data and monitored specifically for client eviction events in the Lustre file system occurring more than three times in five minutes across multiple servers. This enabled us to take corrective action when the specified condition was met or to notify system operators. We also recorded client eviction events in Ganglia. Since Ganglia is a widely used tool to monitor systems, integrating the results from SQLstream can give an administrator an overall system health status and respond to emerging circumstances. CAC also set up a SQLstream monitor on a Linux login node monitoring the security log. Using SQLstream we monitored for more than five failed login attempts within 5 minutes from the same IP address. When that condition was met, a python script was used to block that address for 30 minutes.

4. EVALUATION

Using the set of tools in section 3.1, we experimented with multiple streams of data as well as high frequency points in time. Our non-algorithmic approach is to identify a point or period in time that has a high frequency of events and then look at the data just previous to this time and see if it is indicative of a failure of service or hardware. We separated the syslog events by system type, compute nodes, login nodes, or file systems, for analysis using the Sisyphus toolkit. Each system type produces different types of events. We wanted to look at each system type individually to see if we could narrow down keywords or phrases that would indicate some type of issue. We talked to other system administrators asking for known events that cause problems. We used the database to simulate data coming from client computers and then read the data in streams into either Cayuga or SQLstream.

We also looked at the parallel distributed file system, Lustre events, which are contained in the syslogs. The file system has metadata servers and object storage servers as well as clients to interact with the overall file system. We looked specifically for client evictions at the suggestion of TACC. One client evicted message may not indicate any problem with the system, but if several clients have evicted notices and occur at the same time it could possibly mean there is a problem. A network switch could have failed or a computer program running on the compute nodes could be writing many small files to the file system. A notification to a system administrator would be helpful at this point to indicate that there is a potential for some type of failure which could be checked as soon as the alert was received. In the event it is not a system failure but a user's application causing an issue, then that user could be contacted in order to resolve the problem thereby increasing the performance and reliability of the file system and minimizing the impact on other users.

Using the historical data, we also experimented with the sequence of events from the Lustre metadata, object storage servers, and compute nodes to see if, given an unplanned down time, the frequency of the events by system, previous to the downtime, show a pattern that could be detected. Using Microsoft Excel to graph the number of events by system type, we could see no discernible patterns between the frequency of events by type being logged and an unplanned system down time.

While examining the log files from the various systems we learned that in the Lustre file system, to reduce system log traffic and the size of log files, errors that occur which are duplicates are counted and entered as "Skipped" messages and displayed only once in the log file with the same time stamp as the last event. When a different error type is logged that message will then be displayed after the "Skipped" message and event. When looking at the data graphically in Microsoft Excel, the "Skipped" messages are only counted once and could have occurred any

number of times; therefore, looking for patterns of frequencies will not be conclusive for the Lustre file system. When thinking about how to resolve this issue we initially thought to duplicate the messages while reading the historical data into the database. However, the timestamp information of the "Skipped" messages is not available. Although this exercise did not show conclusive patterns of frequencies, we believe that patterns would emerge if given more information or different system types.

5. SUCCESS STORIES

One of the first goals met in our work was the development of a framework that enabled us to replay a log at any speed to facilitate the development and testing of analysis tools and evaluation of monitoring techniques. This includes techniques and practices for consuming logs and data standards to ensure compatibility with our toolset. With this framework we can replay logs in real time to evaluate tools like SQLstream in a monitoring scenario just as well as we can replay 12 months of logs in a day to analyze the data.

As a first step in looking at log data, a useful tool is a graph showing the number of event logs over time. We built a web-based tool, driven by our database, which allows us to easily add new data, export the values used to generate the graph for more comprehensive analysis in a variety of file types, as well as navigate through years of event data and 'zoom' in on potentially meaningful areas in the log data. This tool is also very helpful in spotting, identifying, and solving some easily preventable issues like dictionary and denial of service attacks as well as misconfigured tasks or jobs.

Using our collection of tools we are able to identify meaningful events in logs and have the ability to monitor them in real time. We can then react to an event or a condition (some number of events within a period of time) by alerting an operator, recording the occurrence, taking corrective action, or any combination thereof. For example, a particular circumstance we were able to easily identify as a problem using our approach was a login attack. On a login node this condition could represent a dictionary attack, a [distributed] denial of service attack, or a rogue process configured by a user or administrator. Our solution was simple; we would configure SQLstream to detect n failed logins within t minutes from the same address. When that condition was met, we would automatically block that address for 30 minutes. We implemented this procedure at CAC on a login node and were able to stop a dictionary attack from an out of network host that very same day with no operator intervention.

In another instance we were able to combine the features of our toolset with specific institutional knowledge from TACC to define a process with the ability to minimize the impact of an emerging condition on user jobs. Specifically, we were told by staff at TACC of a set of events commonly referred to as "client evictions" that indicated one of several potential problems with the Lustre file system that would ultimately affect running and queued jobs. The key to success was that simply finding a client eviction event was not indicative of a wide ranging problem. We needed to use SQLstream and exploit its monitoring ability to detect multiple client evictions from multiple sources at any point in time or, over a definable period of time. Once the stated conditions were met, we had the ability to notify operators immediately, thereby providing the ability to minimize the impact on user's jobs of the file system problem by alerting operators as the problem emerged rather than waiting for complaints or failed job trouble tickets from users.

6. FUTURE WORK

By monitoring events through time and multiple sources all at once, we believe there are more research and testing we can do to predict issues and possible failures on distributed systems. While Nagios and Ganglia provide real-time monitoring, it is really based on an individual system or service. Depending on the severity, patterns, and type of events, either an action can be scripted, a system administrator notified, or counters added to Nagios or Ganglia.

CAC can provide a novel approach to system monitoring. By using logs from TACC[1]s Ranger, we have made substantial strides towards predictive log analysis and near real-time reactive monitoring. Testing using historical data has demonstrated promising results. By using SQLstream and/or Cayuga to process data and generating data visualizations, our team is now ready for additional XSEDE sites to participate in the next stage of testing and evaluation.

7. ACKNOWLEDGMENTS

This project was conducted using the resources of CAC, which receives funding from Cornell University, the National Science Foundation, and other leading public agencies, foundations, and corporations. The authors also wish to acknowledge TACC at The University of Texas at Austin for helping to enable this research.

8. REFERENCES

[1] R. Vaarandi. Mining Event Logs with SLCT and LogHound. In Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium. (ISBN: 978-1-4244-2066-7).

[2] Jon Stearley. Towards Informatic Analysis of Syslogs. In Proceedings of the 2004 IEEE International Conference on Cluster Computing Pages 309-319.

[3] A.J. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In Proceedings of the 2007 International Conference on Dependable Systems and Networks (DSN), 2007.

[4] Demers, J. Gehrke, M. Hong, B. Panda, M. Riedewald, V. Sharma, and W. White. Cayuga: A general purpose event monitoring system. Proc. CIDR, 2007.