Science World Journal Vol 13(No 4) 2018
www.scienceworldjournal.org
ISSN 1597-6343
Published by Faculty of Science, Kaduna State University

Full Length Research Article

UNDERSTANDING ERROR LOG EVENT SEQUENCE FOR FAILURE ANALYSIS

Nentawe Gurumdimma[1], Desmond Bala Bisandu[2*]

[1,2]Department of Computer Science, University of Jos, Nigeria

*Corresponding Author Email Address: yusufn@unijos.edu.ng / bisandud@unijos.edu.ng

ABSTRACT
Due to the evolvement of large-scale parallel systems, they are mostly employed for mission critical applications. The anticipation and accommodation of failure occurrences is crucial to the design. A commonplace feature of these large-scale systems is failure, and they cannot be treated as exception. The system state is mostly captured through the logs. The need for proper understanding of these error logs for failure analysis is extremely important. This is because the logs contain the "health" information of the system. In this paper we design an approach that seeks to find similarities in patterns of these logs events that leads to failures. Our experiment shows that several root causes of soft lockup failures could be traced through the logs. We capture the behavior of failure inducing patterns and realized that the logs pattern of failure and non-failure patterns are dissimilar.

Keywords: Failure Sequences; Cluster; Error Logs; HPC; Similarity

INTRODUCTION

High Performance Computer (HPC) systems have become the backbone for major tasks like weather forecasting, stocks trading, simulation etc. Its use has greatly exploded due to this reliance on it (Nagarajan et al., 2007). It posses great challenge to users not only for its high power dissipation due to size, but also in ensuring that these computers stay without failure (Zheng et al., 2010). In cloud, the servers or clusters that serve the clouds and other services are prone to failures too. These days, almost all applications make use of cluster computers that are yet very prone to failure. And the challenge is how can these failures be detected and / or avoided since any small down time can be significantly costly (Zheng et al., 2010).
This then means that cluster systems must be at their optimum performance and very effective. However, this is not always the case, the cluster nodes fail due to errors. The effects of a single node failure can be immense and propagate leading to failure of more nodes and eventually the whole cluster system if not corrected (Fu and Xu, 2007).
The challenge is to identify these failures for proper or necessary steps to be taken to mitigate the effects or avoid the failures. Root cause analysis is an approach towards solving this problem, and research in this area seeks to study the cluster log files for failures and their probable cause (Yang, 2003). These log file contains systems' health information and other activities going on within the cluster system. However, root cause analysis only seeks to know or identify the failures that occur and the probable cause after the damage is done. The failure could lead to huge cost; therefore, it is better to know if a failure will occur before it actually happens (Fronza et al., 2013). This is called *failure prediction*. A failure can be predicted from the initial symptoms or other means before it occurs; this will allow proactive or necessary corrective measures to be taken to avoid or prevent the failure (Gainaru et al., 2012). This is thought to go a long in helping not only the cluster system administrators, but also the businesses and other applications that runs on this (Cinque et al., 2014).
One of the challenges for researchers trying to do root cause analysis or failure prediction is the understanding of log files to perform their analysis (Fronza et al., 2013). Failure prediction tend to make use of these log events, study the existing failures and understand the pattern in order to build models that can predict future ones (Samak et al., 2012; Bisandu et al., 2018). These log events tends to be chatty, huge and maybe, without any particular standard format in which they were logged (Liang et al., 2006).
In this paper, we introduce an approach that seeks to find similarities in patterns of these logs events that leads to failures. Our contribution is as follows:

i. Work has shown that so many of these log events are similar events occurring or logged at different times. These events can be reduced to foster easy analyses. We use a distance metric and cluster these events according to their similarity and assign a unique Id to them. These IDs can be easily used for analysis algorithms.
ii. Message content of the events logs provides useful information to the errors that occurs. These English messages part of the events logs are extracted for every failure episode. We use Latent Semantic Indexing to perform dimensional reduction of our terms matrix for these failure sequences which can be easily used in our similarity metric algorithm.
iii. In order to understand the patterns in which these events occur, we studied failures and observe if similar failure sequences contains the same pattern, and how long in terms of time can these similarities remain observable across the failure sequences. Our approach employs the Jenson – Shannon Distribution which as a metric captures the similarity in patterns of the failure sequences.

The paper is structured as follows. Section II explains some related works previously done. Section III describes our data and the failures. We explain our data pre-processing in section IV, which also involves clustering log events based on similarity. Section V contains our approach towards finding similarity in the failure sequences patterns. We discuss our results in section VI and conclude in section VII.

Related Work

Over the years, logs have been considered as text files in human readable forms that are readable for administrators and developers. They signify one of the few mechanisms of gaining visibility of system behavior (Hadžiosmanović et al., 2012).

Because of this reason logs have been applied in many areas with different context of application domains in the last decades. A non-exhaustive list of such applications includes the following: mobile devices and control systems (Leveson, 2003), operating systems (Makanju et al., 2010), large-scale applications (Liang et al., 2006), and supercomputers (Gainaru et al., 2013). A significant understanding of failure mode of higher performance systems has been achieved due to the contributions made by these studies: Kang and Grimshaw, (2007), which made the possibility to improve the release of the systems successful. Data log models use many state-of-the-art techniques software packages for manipulating them, e.g. (Makanju et al., 2010; Barringer et al., 2010), ad-hoc algorithms and strategies for identifying failure logs in a failure-related entries to coalesce related entries of the same problem are needed (Goto et al., 2007). Achieving accurate measurements is the critical objective of the tasks which has been re-arranged in this area of research, such as, (Cotroneo et al., 2007), (Oliner and Stearley, 2007). Failure prediction approaches in relation to theory of reliability and preventive maintenance have been designed over the years (Dhiman et al., 2013). Incorporating many factors into the distribution, for example complexity of code Stearley, (2004) has been one of the reasons for model evolving (Kalbfleisch and Prentice, 2002). These methods have been tailored more through long-term predictions and fail to appropriately work for prediction of failures that are online.

More recent approaches for predicting short-term failures are based on runtime monitoring typically as the account of system current state is taking (Gurumdimma et al., 2016; Gurumdimma and Jhumka, 2017). The literatures have indicated two levels of predicting failures, they are: component level and system level. In the first level components are observed (mother board and hard-disk) using parameters that are specific to them and the knowledge of their domain with each having different approaches given the best results for prediction (Sullivan and Chillarege, 1991). Approaches that compare the execution of the failed components with good ones are an example. Several researches from different fields fitting this category are in (Bolander et al., 2009; Patra et al., 2010). The community of Higher Performance Computing (HPC), an example is in Zheng et al., (2007), where the record of system performance matrices using matrices at all intervals. Outliers are detected by the algorithms afterwards by identifying majority nodes and nodes that are far from them.

The second level which is the prediction of failures at the system level, different system parameters are observed by the monitoring daemons (scheduler logs, system logs, performance metrics, etc.) and the existence of correlation between different events are investigated. Significant number of researches has been proposed focusing on HPC systems prediction by analyzing them in the last couple of years. Most predictors however, uses information extracted in the phase of training for short span predictions which required a new phase of training. An example is the work done by the authors in Zheng et al., (2010) and Zhang et al., (2004) which for training it takes up to 3 months, and half month for the predictions, while the later authors make comparison of two approaches for predicting failure and the observation window influence on the results are also studied. The authors in Gu et al., (2008) applied meta-learning predictor in choosing between statistical and rule-based method pending on the predictor that give best result corresponding to the system state. Analogously the authors in Nakka et al., (2011) proposed an approach that analyzed logs by investigating both failure logs and usage.

The authors in Rouillard, (2004) presented their study making difference between application and system failures. Job logs and RAS logs to filter out failures having no effect on system running jobs, allowing them to make couple of observations that are interesting and could be helpful in future failure predictors. An approach more general was proposed by the authors in Rajachandrasekar et al., (2012) which is based on solution of middleware between various application and analysis modules. Decision-making engines and failure predictors relying on the information of distributed failure are to facilitate fault tolerance mechanisms such as preemptive job migration from either their framework. Differently authors in Lou et al., (2010a) and Xu et al., (2009) proposed approaches by investigating parameter correspondence among various log messages of application to extract dependencies between system components. Time-series analysis also have been used for the implementation of different methods of processing, such as subspace method and spike detection in finding patterns among outliers which shows anomalies in monitored systems. Author in Liang et al., (2006) analyze BlueGene/L system logs by combining spatial and temporal filtering, specifically, designed predictive methods for failure which was tested to be effective with about 80% of network and memory failures. Five supercomputer systems logs were analyzed in Oliner and Stearley, (2007), by providing an optimized algorithm of the algorithm proposed by the authors in Liang et al., (2006). However, the filtering algorithm proposed might remove alerts that are independent by coincidence, happening at same time on different nodes.

LOG DATA

A. Error Logs

Log files are mostly the only means and source of information about the workings of any computer systems. It is system administrator's guide to diagnosing faults in computer systems. Computer systems grow more complex and this means increased in the logs also. The task of the system's administrators become complex also or maybe impossible using the large log messages (Barringer et al., 2010; Janbeglou et al., 2010; Lou et al., 2010b). Log messages have become the main source of information for Root Cause Analysis of failures of systems.

Event log files which contains logs with basic information about the state of the system, the activities going on and the system's health related information is what we will focus on for this work.

The challenge with the log file data is that they are generally unstructured, often incomplete, not clearly understood, and most times has no particular message structure. This is often the challenge with the logs; therefore, we process our data to give it some structure. Formatting our data into a structure that is uniform and can give us the necessary information we need for our analysis. A careful investigation of the log messages showed that there is a pattern of occurrence of these errors before a failure (Pecchia et al., 2011). Therefore we decided to analyze further to find patterns and the relationship of the events to failures.

B. Failure Identification

It is necessary we know that our data contain failure events; and that these failures did take place at the super computing system where the log files are recorded.

Soft Lockup Failures: These occur when requests are not attended to maybe due to resource unavailability and also, a process of reconnection initiated is also refused. This repeated process may lead to deadlock processes or the nodes or servers may hang. Eventually, this will lead to loss of data or service access and causes termination of jobs. These soft lockups can be either node soft lockups or server soft lockups. The occurrence of the failure, soft lock is of interest to us. We want to know the kinds of events that would likely lead to this failure.
A study of the log files and the expert's knowledge has shown that Soft lockups are some of commonly occurring failure found in the Cluster or HPC systems. Yuan et al., (2014) has shown that Machine Check Exceptions causes soft lockup in computer systems and also, Evict/RPC events.
Machine Check Exception (MCE) is a way the Computer's hardware reports about an error that the hardware cannot correct. When the kernel logs an uncorrected hardware error, measures can be taken by the cluster software to rectify the problem, re-running the job on another node and/or reporting the failure to the administrator (Janbeglou et al., 2010). Therefore, MCE error makes it possible to predict failures early. Soft lockups led by Evict/RPC Events are characterized by evict and recovery events preceding the failure. Cotroneo et al., (2007) have verified these hypotheses using a correlation and regression technique and obtained a high correlation between the MCE and Evict/RPC events and the soft lockup failure events.

Log pre-processing
In this section we present the detail steps involved in processing the log files into the format we can easily use for our analysis.

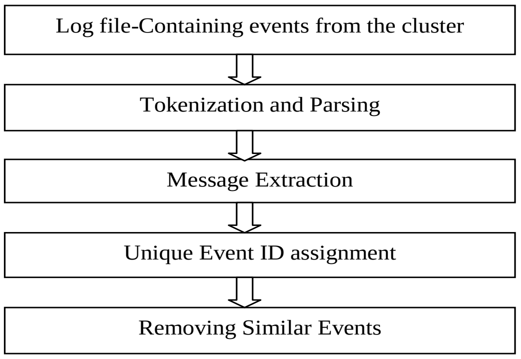
Log file-Containing events from the cluster
Tokenization and Parsing
Message Extraction
Unique Event ID assignment
Removing Similar Events

Figure 1: Log Pre-processing steps

C. Error log Tokenization and Parsing

The inconsistency in the format of error log entries means more challenge in dealing with the logs, especially when all information carried by the logs is important in the analysis (Liang et al., 2006). It becomes very difficult to automatically analyze/process log files because of the differences in the formats, system's information contained in the reports. These systems information can be proprietary in formats and contains so many messages that are not needed. See Figure 2.

Ranger (syslog)
```
Mar 31 15:56:57 i149-405 kernel: [9155992.130789] Machine
check events logged
Mar 31 15:57:07 i144-110 kernel: [9155996.614494] Machine
check events logged
Mar 31 15:58:07 i102-406 kernel: [414646.585574] BUG: soft
lockup detected on CPU#2, pid:22297, uid:0, comm:ldlm_bl_24
Mar 31 15:58:07 i102-406 kernel: [414646.689471] spurious soft
lockup detection on CPU#2
```
Blue Gene/L
```
1117838702 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-
15.45.02.981210 R02-M1-N0-C:J12-U11 RAS KERNEL INFO
instruction cache parity error corrected
1117838703 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-
15.45.03.145256 R02-M1-N0-C:J12-U11 RAS KERNEL INFO
instruction cache parity error corrected
1117838704 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-
15.45.04.007681 R02-M1-N0-C:J12-U11 RAS KERNEL INFO
instruction cache parity error corrected
```
Figure 2: Log events for RANGER syslog and BUE GENE/L

In an attempt towards overcoming these challenges, Fronza et al (Fronza et al., 2013) proposed that log files should not be written for systems administrators, who have a good understanding of the systems alone, but the logs should contain extra information in a well-structured format. In their proposal, the log file format standard must clearly define the type of information to be contained in the logs and must be uniquely represented. As long as information about structure, environment, error event's unique features are not captured in general formats, log pre-processing will always be a necessity for failure prediction purposes (Eason et al., 1995).
We use the standard Linux syslog error events obtained from Ranger supercomputer. Though this data has been given some structure or formats as shown Figure 2 above. The logs were recorded in 2010. In our work, some of the fields are considered not needed for the purpose of this prediction, and hence can be discarded. For example, a field like protocol is not necessary. We also remove all the unnecessary tokens from the messages, for example, tokens containing symbols. The message part, which contains the sequence of English words, explains the error, and this is important to us since we can use text mining techniques to analyze the data. The error message is broken down into tokens and fields that we consider important to our research.

D. Message Extraction

The message is considered as one of the key part of the error event in our research. In our observation, the message contains English words and alphanumeric tokens. The English tokens show a pattern from our observation; and it provides us with clues regarding what an error message is all about or what is going on with the system. The alphanumeric tokens, according to experts, suggest the interacting components or software functions within involved. These component do not occur frequently and shows less or no pattern, hence it become less important in the message and is extracted out (Yuan et al., 2014)

E. Log Event Clustering and ID Assignment

Error log messages need to be labeled with a unique ID for every error event with different and unique message. It is based on the cluster similarity of the events. These messages, which are basically natural language texts providing more insights to the error logs are used for this ID assignment. In our algorithm to perform this, we use the well-known Levenshtein distance, which measures the difference between our message strings, to automatically assign an ID to the error events (Janbeglou et al., 2010). Messages with metric less than a threshold are regarded as similar hence the events will be assigned same ID as seen in Figure 3. The essence is to enable us identify unique events and the pattern in which they occur throughout the whole error log data.

```
Input = log events
LD= Levenshtein Distance
Int i=0;
For  i=1 to N {
  Sim(event(i)) = LD(event(1),event(i));

}
All events with close value of Sim are clustered together
as similar events
For  j = 1 to N {
   ID = getID(EventCluster(j))

//Assign ID to events according to their cluster.
    For each event in EventCluster(j){
              Assign ID to events.
    }
 }
```

Figure 3: Algorithm for event clustering and ID labeling

F. Time Conversion

Our error log messages contain date and time in which these errors are reported. This is very useful to us and for any meaningful failure prediction or analysis. However, this time format is not good for us to use for manipulation. The formats of the timestamps ***2010 Mar 31 15:56:57*** is for an error that occurred at 15th hour, 56th minute and 57th second on March 31, 2010. We convert this to epoch timestamp format. The equivalent epoch timestamp for the above is **1270051017** which can be manipulated easily.

G. Removing Similar Events

There are several seemingly same error events reported frequently in the logs, according to our observation. We also observed that these errors are sometimes reported by same cluster node and within a small time difference. Some are reported by different cluster nodes but same error message and at same time or within a small time difference. According to authors in Kornack and Rakic, (2001) and Heien et al., (2011), occurrence of similar or same events' errors within same time or small time difference might likely be caused by same fault. Therefore, removing the redundant messages is necessary. In another sense, one would say leaving the `redundant' events could be useful in understanding the behaviour of a particular fault in terms of the frequency of the event log generated within the period. However in our case, we consider this not wise, since it is not always the case in a cluster system that this behaviour is observed. Therefore, we reduce these redundant error events which we considered having the following properties:

i. Similar error events that are reported in sequence by same node within a small time threshold. This is because nodes can logs several similar messages that are triggered by same fault.
ii. Similar error events that are reported by different nodes in a sequence and within a time threshold.

This could be triggered by same fault resulting in similar misbehaviour by affected cluster nodes.

Time threshold is a time from the first similar event in a sequence. It is pertinent to note that it is possible that same error messages logged by different nodes are caused by different faults and at different times, hence the time threshold is necessary. Also, our error event similarity is obtained as explained earlier in the section. The process of identifying and grouping the error events exhibiting the above properties is done using a combination of both tupling and time grouping heuristics (Salfner, 2005; Kalbfleisch and Prentice, 2002; El-Sayed and Schroeder, 2013). We define some heuristics that captured the properties outlined in Section IV. It becomes easier to manipulate the data with reduced size. *One of the aims of this research is to determine if there is pattern of occurrence of events leading up to a particular failure, and if so, does these patterns have some similarity?* The challenge now is that we cannot work with the whole data to find patterns within the log events, hence leading to next section, error pattern and window size estimation (Fu and Xu, 2007).

Error pattern and window size estimation

Error log events are quite large as millions of them are logged within long period. This makes it difficult to handle. Given a series of events that occur before a failure, we want to understand the pattern of occurrence of these events and identify any signature. In an attempt to obtain failure pattern, it is necessary to note that different failures with different signatures or pattern can occur and all the error events are logged together (El-Sayed and Schroeder, 2013). Therefore it becomes necessary to know the time window that can adequately capture a particular failure pattern. From Figure 4, we want to know the best minimum time window t_w that can be considered 'good enough' to give us a pattern that led to failure f_1.

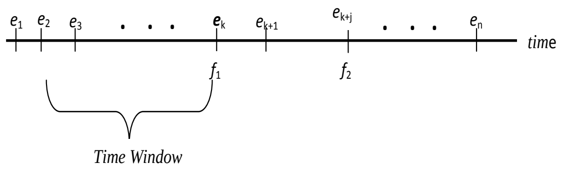

Figure 4: Error events sequence in time

We considered all messages within time window t_w, of 1 - 6 hours for some failure events that occurred within a period of time. We studied the events within the months of March to May 2010. In this section, we explain our attempt towards obtaining a pattern leading to failure. The workflow is as shown in Figure 5.

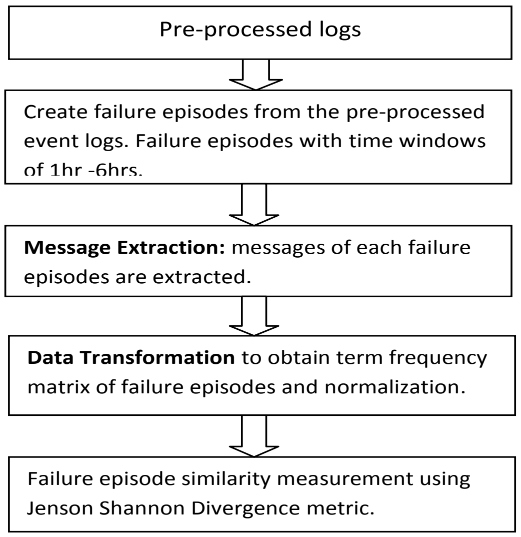
Pre-processed logs
Create failure episodes from the pre-processed event logs. Failure episodes with time windows of 1hr -6hrs.
Message Extraction: messages of each failure episodes are extracted.
Data Transformation to obtain term frequency matrix of failure episodes and normalization.
Failure episode similarity measurement using Jenson Shannon Divergence metric.

Figure 5: Failure sequences similarity and window estimation workflow

H. Data Transformation

Since we are considering only the message part of the error events for our analysis, the unstructured messages were extracted as explained in preprocessing. Hence, we can apply some text analysis techniques to obtain some form of relationship between the contents of the error messages and correlate semantically related terms in the messages. However, this is not just possible without transforming our data to the format that can easily be used by the analysis algorithms.

The messages from each failure sequences are transformed into term - frequency matrix. The rows of the matrix contains the terms while the columns are the failure sequences. A failure sequence consists of events that precedes the failure event within a given time window (Singh et al., 2012). The transformation of data to matrix format is because matrix format can be used easily by our pattern analysis algorithm. Hence we considered this a wise choice.

Consider event logs of a cluster system containing a particular failure f_i , and $i = 1...n$, we extracts all the events that occurred before the failure within a time window t_w ; (in our experiment, we use t_w : 1hr - 6 hrs). For example, given *n* number of soft lockup failures, with t_w = 1 hour, the matrix ***M***, for this will contain *m* rows of terms and *n* columns of failures as shown in Figure 6:

$$M = \begin{pmatrix} t_1 f_1 & t_1 f_2 & . & . & . & t_1 f_n \\ t_2 f_1 & & & & & \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ t_m f_1 & t_m f_2 & . & . & . & t_m f_n \end{pmatrix}$$

Figure 6: Data matrix M for *n* failure sequences

Matrix Normalization: Consider having many failure sequences with similar terms between them; the term frequency matrix may contain the combination of all the terms in the failure episode, for example, when we have 100 thousand terms, it means the covariance matrix will have be of dimension 100,000 by 100,000. This is quite high dimension. This high dimension data is reduced using Latent Semantic Indexing, LSI. LSI performs dimensional reduction just like PCA, the difference is that in LSI, pre-processing of data, which involve vector normalization to zero mean and normalization to unit variance is not done. Normalizing the feature data to unit variance will unnecessarily scale the weight of rarely occurring terms in the failure sequences. Some relationship between terms vectors which were not clearly known, LSI expressed this by reducing the noisy relationships (Nagarajan et al., 2007; Goto et al., 2007; Samak et al., 2012; Pecchia et al., 2011). It performs this by decomposing the raw matrix M into three reduced matrices, USV . Obtaining a k-dimension reduced matrix M_k as in Equation 1.

$$Mk = U_k S_k V^T_k \qquad (1)$$

Where *U* = term vector, *S* = computed diagonal matrix of decreasing singular values, *V* = failure sequences vector.

The term weights or frequency across the failure sequences are normalized to a value within 0 and 1 as in Equation 2.

$$n_t = \frac{w_t(f_i)}{\sum_{i=1}^{n} w_t(f_i)} \qquad (2)$$

Where n_t is normalized term weight, $w_t(f)$ is weight of term *t* in failure episode f.

I. Failure Pattern Similarity Measure Using Jenson – Shannon Divergence metric

After identifying different soft lockup failures, this section seeks to know how similar these failure patterns or the error events of each failure sequences are to the other. From section II, we established that soft lockup failures can be caused by either Machine Check Exception events (MCE) or Evict/RPC events. The similarities between these failures are obtained to be sure if MCE or Evict/RPC led soft lockups contain similar failure pattern.

The Jenson – Shannon Divergence (JSD), measures the divergence or similarity between two or more probability distributions (Kalbfleisch and Prentice, 2002; Rouillard, 2004). Messages from the failure sequences that are similar, yet semantically unrelated are not expected to be considered similar; however most of the metrics do not take that into consideration. JSD does this by considering the entropies of these messages, hence our choice for it (Makanju et al., 2010).

Consider failure sequences containing several log events having information regarding the likely cause of failure. We want to establish that failures led by the same events should not vary much.

Given distribution of failures sequences $F = \{f_1, f_2, ..., f_n\}$, and $f_i = \{t_1, ..., t_k\}$ contains events' term-frequency distribution, where $\sum_{i=1}^{k} t_i = 1$ and $0 \le t_i \le 1$ for all $i = 1, 2, \ldots, k$.

Let the weights of the distributions of failure sequences be π_i, then the JSD for f_i is given by Equation 3:

$$JSD(f_i) = H\left(\sum_{i=1}^{n} \pi_i f_i\right) - \sum_{i=1}^{n} \pi_i H(f_i) \qquad (3)$$

Where $H(f_i)$ is Shannon entropy for the distribution f_i and $\sum \pi_i = 1;\ 0 \le \pi_i \le 1$.

Now for two failure sequences, f_1 and f_2,

$$JSD(f_1, f_2) = H\left(\tfrac{1}{2}(f_1 + f_2)\right) - \tfrac{1}{2}\left(H(f_1) + H(f_2)\right)$$

; where $H(f) = -\sum_{i=1}^{k} f_1 \log_2 f_i$ is the Shannon entropy and $\pi = \tfrac{1}{2}$.

Hence, the similarity between the two failure sequences is given by Equation 4:

$$Sim(f_1, f_2) = 1 - JSD(f_1, f_2) \qquad (4)$$

With similarity value ranging between 0 and 1.

RESULTS AND DISCUSSION

The result of our experiment with error log events was carried out on logs from Ranger supercomputer of Texas Advanced Computing Centre (TACC), University of Texas, Austin. The logs are for the period March 2010 and June 2010. In performing this experiment, we manually study the logs and identify soft lockup failure events within the stipulated time (March - June 2010). Figure 7 shows the distribution of soft lockup, interrupt, rpc/evict events within these months. These events are highly correlated to soft lockup failures, which are regarded as events that like precedes these failures.

There are several root causes of soft lockup failures but we want to focus on machine check events and RPC/Evict events led soft lockups. These events formed the failure sequences. The events preceding these failures are obtained within time window of 1 -6 hours. From Figure 8, MCE events led failures, f_1, f_2 and failures led by RPC/evict events f_4, f_6 shows some form of variations in the similarity of the patterns. This is as expected. Within time window of 1 – 3 hours, the similarity in pattern seems to be clearer than for time window 4 - 6 hours, which suggests that for large *time window*, it is possible that different other failures would have occurred and led by different other events. Again, removing redundant events greatly improve the clarity of failure patterns. This suggests that, preprocessing is an important step in understanding logs for failure analysis. Some researchers argue that the redundant events also constitute integral part of the failure patterns, however, we realized that it is not always the case.

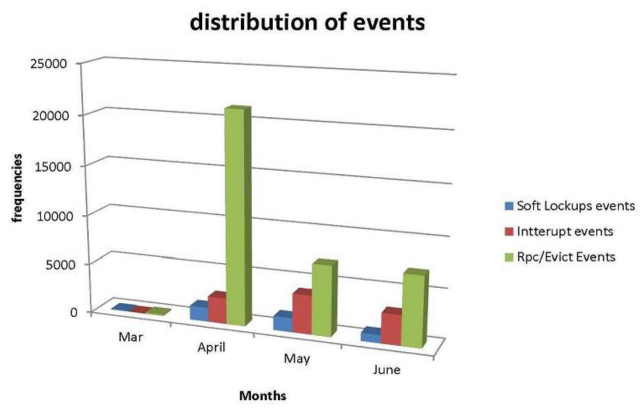

Figure 7: Distributions of some events

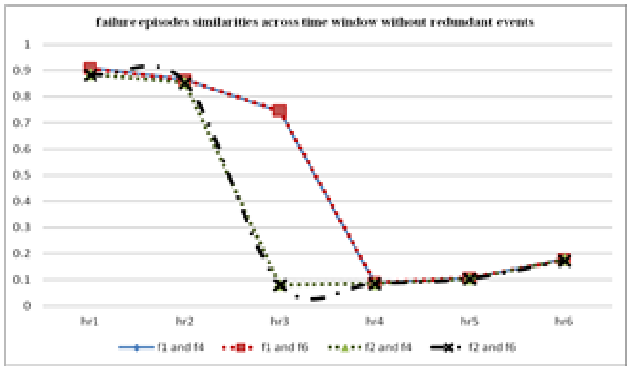

Figure 8: Failure sequences Similarity Across time windows with no redundant events

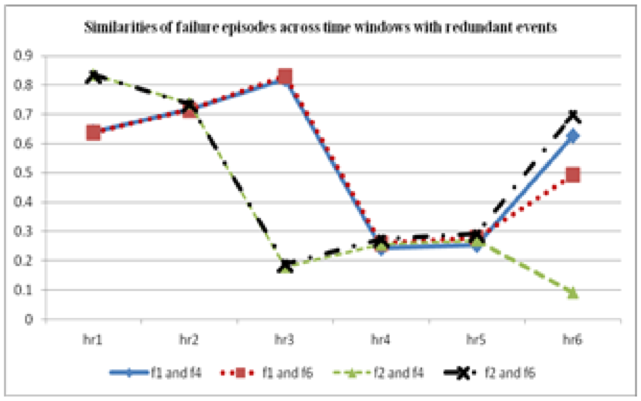

Figure 9: Failure sequences Similarity Across time windows with redundant events

Conclusion

Accurate detection of failure patterns in logs of supercomputers, understanding behavior of the systems with their generated logs is crucial. As the sizes of application scales and increase, the failure tends to occur more often within short range of time. The failure's impact on the system performance becomes more pronounced, making the task of analyzing and quantifying the extent of the failure impact difficult. The failure traces from large-scale systems are mostly unavailable. Our approach seeks to find similarities in patterns of these logs events that leads to failures.

We use latent semantic indexing for reducing the dimension of the data before finding the similarity between the patterns by knowing the time, locations, and the down time failure. The failure traces generated by the model is used to understand the behavior of certain failures in the system. The result of the experiment has revealed some insightful knowledge: we discovered that from logs, system failure behavior could be traced. Traditional failure correction methods such as regular checkpoints would be properly done if failure behaviours can be detected early. Finally, removing redundant event logs provides better understanding the sequence of event logs for which failure inducing patterns can be traced.

REFERENCES

Barringer, H., Groce, A., Havelund, K., Smith, M., 2010. Formal Analysis of Log Files. J. Aerosp. Comput. Inf. Commun. 7, 365–390.

Bisandu, D.B., Prasad, R. and Liman, M.M. 2018. Clustering news articles using efficient similarity measure and N-grams, *Int. J. Knowledge Engineering and Data Mining*,Vol. 5, No. 4, pp.333–348.

Bolander, N., Qiu, H., Eklund, N., Hindle, E., Rosenfeld, T., 2009. Physics-based remaining useful life prediction for aircraft engine bearing prognosis, in: Annual Conference of the Prognostics and Health Management Society.

Cinque, M., Cotroneo, D., Della Corte, R., Pecchia, A., 2014. Assessing direct monitoring techniques to analyze failures of critical industrial systems, in: Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium On. IEEE, pp. 212–222.

Cotroneo, D., Pietrantuono, R., Mariani, L., Pastore, F., 2007. Investigation of failure causes in workload-driven reliability testing, in: Fourth International Workshop on Software Quality Assurance: In Conjunction with the 6th ESEC/FSE Joint Meeting. ACM, pp. 78–85.

Dhiman, M.P., Anand, D., Singh, E., Grover, K., 2013. PC based speed control of induction motor. Int. J. Emerg. Trends Electr. Electron. IJETEE–ISSN 2320-9569 2, 81–84.

Eason, G., Noble, B., Sneddon, I.N., 1995. On certain integrals of Lipschitz-Hankel type involving products of Bessel functions. Phil Trans R Soc Lond 247, 529–551.

El-Sayed, N., Schroeder, B., 2013. Reading between the lines of failure logs: Understanding how HPC systems fail, in: Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference On. IEEE, pp. 1–12.

Fronza, I., Sillitti, A., Succi, G., Terho, M., Vlasenko, J., 2013. Failure prediction based on log files using random indexing and support vector machines. J. Syst. Softw. 86, 2–11.

Fu, S., Xu, C.-Z., 2007. Exploring event correlation for failure prediction in coalitions of clusters, in: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. ACM, p. 41.

Gainaru, A., Cappello, F., Snir, M., Kramer, W., 2013. Failure prediction for HPC systems and applications: Current situation and open issues. Int. J. High Perform. Comput. Appl. 27, 273–282. https://doi.org/10.1177/1094342013488258

Gainaru, A., Cappello, F., Snir, M., Kramer, W., 2012. Fault prediction under the microscope: A closer look into hpc systems, in: High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference For. IEEE, pp. 1–11.

Goto, H., Hasegawa, Y., Tanaka, M., 2007. Efficient scheduling focusing on the duality of MPL representation, in: Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium On. IEEE, pp. 57–64.

Gu, J., Zheng, Z., Lan, Z., White, J., Hocks, E., Park, B.-H., 2008. Dynamic meta-learning for failure prediction in large-scale systems: A case study, in: Parallel Processing, 2008. ICPP'08. 37th International Conference On. IEEE, pp. 157–164.

Gurumdimma, N., Jhumka, A., 2017. Detection of Recovery Patterns in Cluster Systems Using Resource Usage Data, in: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC). Presented at the 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 58–67.

Gurumdimma, N., Jhumka, A., Liakata, M., Chuah, E., Browne, J., 2016. CRUDE: Combining Resource Usage Data and Error Logs for Accurate Error Detection in Large-Scale Distributed Systems, in: 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS). Presented at the 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS), pp. 51–60.

Hadžiosmanović, D., Bolzoni, D., Hartel, P.H., 2012. A log mining approach for process monitoring in SCADA. Int. J. Inf. Secur. 11, 231–251.

Heien, E., Kondo, D., Gainaru, A., LaPine, D., Kramer, B., Cappello, F., 2011. Modeling and tolerating heterogeneous failures in large parallel systems, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, p. 45.

Janbeglou, M., Zamani, M., Ibrahim, S., 2010. Redirecting network traffic toward a fake DNS server on a LAN, in: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference On. IEEE, pp. 429–433.

Kalbfleisch, J.D., Prentice, R.L., 2002. The statistical analysis of failure time data, 2. ed. ed, Wiley series in probability and statistics. Wiley, Hoboken, NJ.

Kang, W., Grimshaw, A., 2007. Failure prediction in computational grids, in: Simulation Symposium, 2007. ANSS'07. 40th Annual. IEEE, pp. 275–282.

Kornack, D.R., Rakic, P., 2001. Cell proliferation without neurogenesis in adult primate neocortex. Science 294, 2127–2130.

Leveson, N., 2003. A new accident model for engineering safer systems. Saf. Sci. 42, 237–270.

Liang, Y., Zhang, Y., Sivasubramaniam, A., Jette, M., Sahoo, R., 2006. Bluegene/l failure analysis and prediction models, in: Dependable Systems and Networks, 2006. DSN 2006. International Conference On. IEEE, pp. 425–434.

Lou, J.-G., Fu, Q., Wang, Y., Li, J., 2010a. Mining dependency in distributed systems through unstructured logs analysis. ACM SIGOPS Oper. Syst. Rev. 44, 91–96.

Lou, J.-G., Fu, Q., Yang, S., Xu, Y., Li, J., 2010b. Mining Invariants from Console Logs for System Problem Detection., in: USENIX Annual Technical Conference. pp. 1–14.

Makanju, A., Zincir-Heywood, A.N., Milios, E.E., 2010. An evaluation of entropy based approaches to alert detection in high performance cluster logs, in: Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on The. IEEE, pp. 69–78.

Nagarajan, A.B., Mueller, F., Engelmann, C., Scott, S.L., 2007. Proactive fault tolerance for HPC with Xen virtualization, in: Proceedings of the 21st Annual International Conference on Supercomputing. ACM, pp. 23–32.

Nakka, N., Agrawal, A., Choudhary, A., 2011. Predicting node failure in high performance computing systems from failure and usage logs, in: 2011 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2011. Presented at the 25th IEEE International Parallel and Distributed Processing Symposium, Workshops and Phd Forum, IPDPSW 2011, pp. 1557–1566.

Oliner, A., Stearley, J., 2007. What supercomputers say: A study of five system logs, in: Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference On. IEEE, pp. 575–584.

Patra, A.P., Bidhar, S., Kumar, U., 2010. Failure prediction of rail considering rolling contact fatigue. Int. J. Reliab. Qual. Saf. Eng. 17, 167–177.

Pecchia, A., Cotroneo, D., Kalbarczyk, Z., Iyer, R.K., 2011. Improving log-based field failure data analysis of multi-node computing systems, in: Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference On. IEEE, pp. 97–108.

Rajachandrasekar, R., Besseron, X., Panda, D.K., 2012. Monitoring and predicting hardware failures in HPC clusters with FTB-IPMI, in: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. IEEE, pp. 1136–1143.

Rouillard, P.J., 2004. Real-time Log File Analysis Using the Simple Event Correlator (SEC), in: Proceedings of LISA '04: Eighteenth Systems Administration Conference, (Atlanta, GA: USENIX Association, November, 2004). Presented at the LISA, pp. 133–150.

Salfner, F., 2005. Predicting failures with hidden Markov models, in: Proceedings of 5th European Dependable Computing Conference (EDCC-5). pp. 41–46.

Samak, T., Gunter, D., Goode, M., Deelman, E., Juve, G., Silva, F., Vahi, K., 2012. Failure analysis of distributed scientific workflows executing in the cloud, in: Proceedings of the 8th International Conference on Network and Service Management. International Federation for Information Processing, pp. 46–54.

Singh, V.P., Vaibhav, K., Chaturvedi, D.K., 2012. Solar power forecasting modeling using soft computing approach, in: Engineering (NUiCONE), 2012 Nirma University International Conference On. IEEE, pp. 1–5.

Stearley, J., 2004. Towards informatic analysis of syslogs, in: Cluster Computing, 2004 IEEE International Conference On. IEEE, pp. 309–318.

Sullivan, M., Chillarege, R., 1991. Software defects and their impact on system availability: A study of field failures in operating systems, in: FTCS. pp. 2–9.

Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M., 2009. Online system problem detection by mining patterns of console logs, in: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference On. IEEE, pp. 588–597.

Yang, S.K., 2003. A condition-based failure-prediction and processing-scheme for preventive maintenance. IEEE Trans. Reliab. 52, 373–383.

Yuan, D., Luo, Y., Zhuang, X., Rodrigues, G.R., Zhao, X., Zhang, Y., Jain, P., Stumm, M., 2014. Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems., in: OSDI. pp. 249–265.

Zhang, Y., Squillante, M.S., Sivasubramaniam, A., Sahoo, R.K., 2004. Performance implications of failures in large-scale cluster scheduling, in: Workshop on Job Scheduling Strategies for Parallel Processing. Springer, pp. 233–252.

Zheng, Z., Lan, Z., Gupta, R., Coghlan, S., Beckman, P., 2010. A practical failure prediction with location and lead time for blue gene/p, in: Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference On. IEEE, pp. 15–22.

Zheng, Z., Li, Y., Lan, Z., 2007. Anomaly localization in large-scale clusters, in: Cluster Computing, 2007 IEEE International Conference On. IEEE, pp. 322–330.