Event Block Analysis for Effective Anomaly Detection on Production HPC Systems

Zongze Li, Matthew Davidson and Song Fu
Department of Computer Science and Engineering
University of North Texas
{Zongzeli2, MatthewDavidson}@my.unt.edu, song.fu@unt.edu

Sean Blanchard and Michael Lang
UltraScale Systems Research Center
Los Alamos National Laboratory
seanb@lanl.gov, mlang@lanl.gov

ABSTRACT

As HPC systems grow dramatically in both scale and complexity, the sheer volume of syslogs and complicated interactions between system components make traditional manual diagnosis and even automated line-by-line analysis infeasible or ineffective. In this paper, we propose a System Log Event Block Detection (SLEBD) approach that can extract groups of log lines that appear following similar sequences and explore these event blocks for event analysis and prediction. Compared with the existing methods that analyze syslogs line by line, SLEBD is capable of characterizing system behavior and identifying intricate anomalies at a higher level. We evaluate the performance of SLEBD using syslogs from production HPC systems. Experimental results show that SLEBD can process streaming messages, which enables system operators and other tools to understand and process events in real-time.

1. INTRODUCTION

System logs provide a valuable resource for understanding system behavior and detecting anomalies on HPC systems. A number of methods and tools have been proposed and developed for log analysis. Existing approaches based on line-by-line log analysis can discover distribution and precedence relation between log lines. However, log messages are not isolated. A single event of a component or system may generate multiple messages. Analysis at the *event level* can provide a richer semantics of system behaviors and thus detect more subtle anomalies that the traditional line-by-line analysis methods cannot find.

We use *event block* (EB) to refer to the log messages that belong to a component or system event. The advantages of event block based log analysis are clear. By converting the original, lengthy and unstructured messages in syslogs into a compact and structured list of EBs, the complexity of log analysis can be significantly reduced. By working at the EB level, we can find the patterns of events, the evolution of system behavior, and the interactions between different system components. Variation among instances of the same event is also an indicator of possible anomalies.

In this paper, we present a System Log Event Block Detection (SLEBD) approach that extracts event blocks accurately and automatically. User only need to provide a system log format pattern to indicate time stamp, node id and log message. Then SLEBD can work on such format of HPC system log. LEBD leverages the law of total probability [1] to identify EBs from syslogs. The identified EBs are stored in an event block database (EBD). SLEBD is capable of processing streaming messages and analyzing system events and behavior in real time. We can do anomaly detection by comparing future system behavior and learned behavior model.

2. SYSTEM LOG PROCESSING

On large-scale HPC systems, messages from compute nodes and service nodes are often mixed together. We separate log messages into multiple files based on node IDs.

Some events may produce multiple lines of messages, but only show up once in a time period on one node. Moreover, multiple messages may come together in a time period on one node. However, they do not show up together in log files of other nodes. We merge them into one EB if only the log file of that node is considered. We call it a false-merged EB.

Thus, the separated files from different nodes are sequentially merge into a single file. This helps us identify those EBs whose sets of messages appear on multiple nodes and reduce the possibility of producing false-merged EBs.

3. EVENT BLOCK DETECTION METHOD

Using the processed log files as input, SLEBD performs in two steps: 1) building EBD from the processed log files, and 2) extracting an EB list from the log message stream.

3.1 Event Block Database Generation

3.1.1 Line Pattern Creation

Log messages for the same system events are generated by similar threads or devices. They have similar message pattern in syslogs. For example:

ACPI: PCI Root Bridge [PCI0] (domain 0000 [bus 00-fe])
ACPI: PCI Root Bridge [UNC0] (domain 0000 [bus ff])

SLEBD considers words which contain alphabet letters and ignore those having numbers in them. Comparing the preceding two messages, their similarity is 86% (i.e., 6/7). They have the same line pattern:

[1, “ACPI\:”], [2, “PCI”], [3, “Root”], [4, “Bridge”], [6, “\(domain”] [8, “\[bus”]

SLEBD annotates each single line pattern with “[Block_$num]”. Using these line patterns, we generate a temporary pattern list from the original log file.

3.1.2 Event Block Consolidation

As an illustrating example, the original log file has five lines, and its temporary pattern list is shown in Table 1.

Table 1. Example temporary pattern list

Block name	Start and finish line number
Block_1	[1, 1]
Block_3	[2, 2]
Block_1	[3, 3]
Block_2	[4, 4]
Block_3	[5, 5]

LA-UR-17-24869

SLEBD then generates the block conditional probability matrix based on the temporary block list as shown in Table 2.

Table 2. Block Conditional Probability Matrix

	Block_1	Block_2	Block_3	Last
Block_1		50%	50%	
Block_2			100%	
Block_3	50%			50%

SLEBD also creates a forward range block list for each starting block as shown in Table 3.

Table 3. Forward Block List

Block_1	Block_1, Block_2, Block_3
Block_2	Block_3
Block_3	Block_1, Block_2, Block_3, last

Bayes' theorem has an extended form, i.e., the Law of total probability [1], which can be expressed as

$$P(E|A) = \Sigma P(E \mid A \cap Bn) * P(Bn \mid A) \quad (1)$$

This inspires us to develop an algorithm that calculates the probability of a block, e.g., Block_3, happening in the forward range where another block, e.g., Block_1, appears.

$$PF(A \rightarrow E) = \Sigma PF(A \rightarrow Bn) * PF(Bn \rightarrow E) \quad (2)$$

We use "PF (B_1→B_3)" to denote this forward range probability as

PF (B_1→B_3)
= PF (B_1→B_2) * PF (B_2→B_3) + PF (B_1→B_3)
=50% * 100% + 50% = 100%

As PF (B_1→B_3) = 100%, greater than a predefined threshold, we can treat them as one pair of EB for possible merging. We also calculate the backward range probability PB (B_3->B_1) = 100% in a similar way. This indicates Block_1 and Block_3 are always happening together. We thus create a new EB pattern which have the start line pattern as Block_1 and the finish line pattern as Block_3.

This procedure is repeated until no more EBs can be merged. The produced EB patterns are stored in EBD.

3.1.3 Improving EBD by using multiple logs.

It is not guaranteed that EBD training files cover all possible events in a system. To tolerate false-merged EBs and add unseen EB patterns into EBD, SLEBD possesses a feature that improves EBD by using multiple logs. This helps merge newly found single line patterns and add them to EBD. It does not merge existing EB patterns with newly found EB patterns. These assure that new EB patterns can be captured and they do not affect existing EB patterns.

3.2 Event Block Extraction

SLEBD uses stack for messages from each node and records the number of log lines that have been received and processed for that node. For each log line, SLEBD generates a line pattern and searches for a possible EB pattern in EBD. If this line is the start of a block pattern, then the block ID and log line number is pushed into the stack for the node. If the line is the end of a block pattern, then the block ID saved at the top of the stack is popped out and compared with the end of the block pattern. In case that they match, this block ID and the log line number are written in an EB list report. If not, SLEBD stores them in a conflict list for further analysis and check if these two EB patterns are false-merged.

4. EXPERIMENTS ON MUTRINO HPC SYSTEM

We test SLEBD using logs collected from the Mutrino HPC system which is hosted at Sandia National Laboratories. The dataset has 553 console logs. We use the first 50 files for EBD building and the rest 503 files for event block extraction. We have detected 1409 single line patterns from the 50 building files, and finally got 737 EB patterns. Table 4 shows the results from EBD building. Table 5 presents those from EB extraction.

Table 4. EBD Building Results on Mutrino

Total number of log lines	617,255
Total number of valid lines	592,978
EB count	396,196
Number of lines covered by EBs	595,849
Single line patterns count	1,409
EB coverage ratio	96.5% (595,849/617,255)
Learned EBD count	737
Single line EB patterns	635
Multi line EB patterns	102

Table 5. Event Block Extraction Results on Mutrino

Total number of log lines	2,548,174
Total number of valid lines	2,455,553
EB count	894,487
Number of lines covered by EBs	1,501,461
EB coverage ratio	59%

We find that some files used in EB extraction have a coverage lower than 20%. This infers that these files have new information that the EBD building phase does not have. We can explore them to enhance EBD.

5. CONCLUSIONS

Attractive features of SLEBD include that 1) it generates an event block pattern database from system logs. Users can use EBD to process real-time message streams. 2) It updates EBD by continuously analyzing multiple log files. The EBD can be evolved at run time. 3) It analyzes EB lists to identify the characteristics and dynamics of EBs, which enables operators to monitor system behavior and identify anomalies. For example, we can explore frequent sequential pattern mining methods [2, 3] on EB lists to capture execution sequence among EBs.

ACKNOWLEDGMENT

This work was supported in part by the U.S. Department of Energy and LANL ASC funding. The publication has been assigned the LANL identifier LA-UR-17-24869.

REFERENCES

[1] David Niju. "Law of Total Probability". Available at: https://ssrn.com/abstract=1310502.

[2] Mohanmmed J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences", Machine Learning, 42, pp. 31–60, 2001.

[3] Jiawei Han, Jian Pei, Behzad Mortazavi-asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. "FreeSpan: frequent pattern-projected sequential pattern mining". In ACM SIGKDD, 2000.

Event Block Analysis for Effective Anomaly Detection on Production HPC Systems

Zongze Li, Matthew Davidson and Song Fu
Department of Computer Science and Engineering, University of North Texas
{Zongzeli2, MatthewDavidson}@my.unt.edu, song.fu@unt.edu

Sean Blanchard and Michael Lang
UltraScale Systems Research Center, Los Alamos National Laboratory
seanb@lanl.gov, mlang@lanl.gov

Motivation

HPC systems grow dramatically in both scale and complexity. The traditional manual diagnosis and even automated line-by-line analysis on HPC system log become infeasible or ineffective.

We found a single event of a component or system may generate multiple messages. We use *Event Block* (EB) to refer to such a set of log messages.

We propose a **System Log Event Block Detection (SLEBD)** approach that extract groups of log line that appear following similar sequences and explore these event blocks for event analysis and prediction.

System Log Preprocessing

Messages from compute nodes and service nodes are often mixed together.

We group messages in a predefined time window from one node into a file.

Modeling System Logs

1. Creating single Line patterns

Log messages for the same system events are generated by similar threads or devices. They have similar message pattern. For example:

ACPI: PCI Root Bridge [PCI0] (domain 0000 [bus 00-fe])
ACPI: PCI Root Bridge [UNC0] (domain 0000 [bus ff])

SLEBD considers words which contain alphabet letters and ignore those having numbers in them. Comparing the preceding two messages, their similarity is 86% (i.e., 6/7). They have the same line pattern as

[1, “ACPI\:”], [2, “PCI”], [3, “Root”], [4, “Bridge”], [6, “\(domain”] [8, “\[bus”]

SLEBD annotates each single line pattern with a [message_$num]. Using these line patterns, we generate a temporary pattern list from the original logs.

2. Generating line pattern list and conditional probability matrix

Convert original logs into line pattern list

Message name	Start and finish line number
Message_1	[1, 1]
Message_3	[2, 2]
Message_4	[3, 3]
Message_1	[4, 4]
Message_2	[5, 5]
Message_3	[6, 6]
Message_4	[7, 7]

Generate forward conditional probability matrix

	Message_1	Message_2	Message_3	Message_4	Last
Message_1		50%	50%		
Message_2			100%		
Message_3				100%	
Message_4	50%				50%

Event Block Detection Method

1. The Law of total probability

Bayes' theorem has an extended form, i.e., the Law of total probability [1]:

$$P(E|A) = \Sigma P(E \mid A \cap Bn) * P(Bn \mid A)$$

2. Closest message pair detection

We then detect each message’s closest messages which tend to happen together in a short range. We design a function to calculate the probability that message_B happens when message_A happens:

$$P(A\text{->}B) = \sum_{i=1}^{\text{length(A\_follow\_list)}} P(\text{Message\_i} \rightarrow B) * P(A \text{ -> Message\_i})$$

where Message_i is the i-th message directly following Message_A.

We use PF (M_1→*M_3) to denote the probability that Message_3 appears in Message_1’s forward range and PB(M_1→*M_3) to denote that Message_1 appears in Message_3’s backward range. In our example:

PF(M_1->*M_3)
= PF(M_1->M_3) + PF(M_1->M_2) * PF(M_2->M_3)
= 50% * 100% + 50% * 100% = 100%

We also generate a backward conditional probability matrix and calculate backward range probability in the same way:

PF(M_1->*M_3) = 100% and PB(M_1->*M_3) = 100%.

Since both probabilities are above a pre-defined threshold. We say Message_1 and Message_3 are a pair of closest messages that can be merged into one event block.

3. Event Block Consolidation

After finding the closest message list:

Message name	Closest Message
Message_1	Message_3
Message_3	Message_4

We merge closest messages together into an Event Block and annotate this block with a name as [Block_$num].

Event Block name	Message list
Block_1	Message_1, Message_3, Message_4

System Log Modeling with Event Blocks

After detecting all event block patterns from the learning file set, SLEBD converts original log files or streaming log messages into a single line pattern list and replaces line patterns which belong to a block by its corresponding block name and line numbers in the log file or message stream.

In our experiment, the test log file is converted to:

Block name	Start and end line number
Block_1	[1, 3]
Block_1	[4, 7]

Experimental Results

We test SLEBD using logs collected from Mutrino HPC system. The dataset has 553 console logs. We use the first 50 files to build our EB database.

Total number of log lines	617,255
Total number of valid lines	592,978
Learned EBD count	737
Single line EB patterns	635
Multi line EB patterns	102
Event block (EB) count	396,196
Number of lines covered by EBs	595,849
Single line patterns count	1,409
EB coverage ratio	96.5% (595,849/617,255)

Table 1. Experiment result on Muitrino log

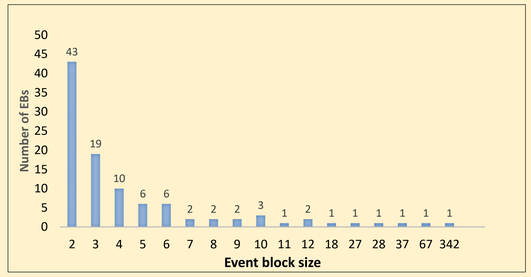

Figure 1. Distribution of multi-line event blocks.

We also extract event blocks from the rest 503 log files and compare them with the results from the first 50 files. Groups 13 and 14 show some change of system configuration or behavior.

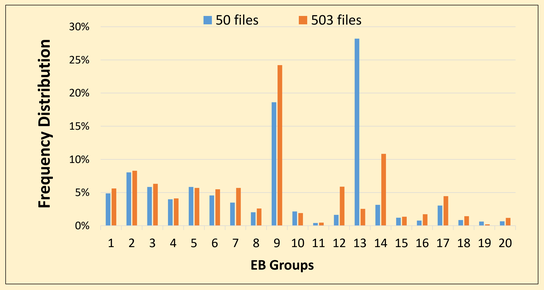

Figure 2. Multi-line event block distribution among 20 EB groups.

Conclusions

SLEBD possesses the following attractive features.

1) It automatically builds an event block database (EBD) from logs.
2) It uses EBD to process and analyze real-time message streams.
3) It updates EBD by continuously analyzing multiple log files.
4) It analyzes EB lists to identify characteristics and dynamics of EBs for system monitoring and anomaly detection.

References

[1] David Niju. “Law of Total Probability” (December, 2008). Available at: https://ssrn.com/abstract=1310502.
[2] Mohanmmed J. Zaki. “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, Machine Learning, 42, pp. 31–60, 2001.
[3] Jiawei Han, Jian Pei, Behzad Mortazavi-asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. “FreeSpan: frequent pattern-projected sequential pattern mining”. In Proc. of ACM Conference on Knowledge Discovery and Data Mining (KDD), 2000.

LA-UR-17-24869