# RAVEN: RAS data Analysis through Visually Enhanced Navigation

Byung-Hoon Park, Guruprasad Kora, Al Geist, *Oak Ridge National Laboratory*
Junseong Heo, *National Institute of Computational Science, UTK*

***ABSTRACT:*** *Supercomputer RAS event data contain various signatures regarding system status, thus are routinely examined to detect and diagnose faults. However, due to voluminous sizes of logs generated during faulty situations, a comprehensive investigation that requires comparisons of different types of RAS logs over both spatial and temporal dimensions is often beyond the capacity of human operators, which leaves a cursory look to be the only feasible option. As an effort to better embrace informative but huge supercomputer RAS data in a fault diagnosis/detection process, we present a GUI tool called RAVEN that visually overlays various types of RAS logs on a physical system map where correlations between different event types can be easily observed in terms of their quantities and locations at a given time. RAVEN also provides an intuitive event navigation capability that helps examine logs by clustering them to their common locations, types, or user applications. By tracing down notable event patterns reflected on the map and their clustered logs, and superimposing user application data, RAVEN, which has been adopted at National Institute of Computational Science (NICS) at the University of Tennessee, identified root causes of several system failures logged in Kraken XT5.*

**KEYWORDS:** Log Analysis, XT5, Root cause analysis.

## 1. Introduction

Detection and diagnosis of failures in a supercomputer typically involve an analysis of reliability, availability, and serviceability (RAS) event logs that contain text descriptions about events of both hardware and software components [7, 8]. However, fueled by the ever-growing scale and complexity of computer systems, the volume and complexity of RAS logs already reached the point where the manual analysis by a human operator is no longer feasible, and will continue to grow [6].

Events described in RAS logs are irregular in their occurrences. Near the vicinity of the time of a failure, either before or after, an avalanche of events is generated portraying different views of the failure observed from different components. Although embarrassingly large in size, the entire logs generated during the span of this period are mostly redundant leaving only a fraction of data relevant for the analysis. However, sifting such spurious events is by no means tractable without a proper aid. In cases when not a single but multiple components are the source of a failure, the root cause is best identified by tracing a stream of highly correlated event types. However such correlations are very hard to capture unless temporal intervals when the correlations stand out are carefully predetermined. Moreover many correlations are implicit and spurious masking off real and important event correlations.

A much clearer picture of the system status can be obtained when contexts of logs are considered. First, in many cases logs describe events that have pair wise relations. For example, most Lustre messages report the failed I/O attempts with the target destinations such as between OSS and OSC. All Basic End to End Reliability (BEER) messages include the failed communication between two nodes. If most BEER messages generated at a given time address problems with a single node, it is most likely that the mentioned node is not in a normal status. Second, events can be clustered based on the context of a user application. User applications that are not properly tuned for the intended scale tend to impose unforeseen overheads to the system. This typically involves excessive amounts of communications between the nodes occupied by the application or ill coordinated checkpoint attempts. In such a case, certain types of events (e.g., Lustre, BEER, etc.) can be understood with respect to applications.

A plethora of tools have been introduced to aid system administrators with analysis of log files. Most of these tools are valuable to detect mere occurrences of faults or capture a temporal summary of event occurrences. However, their usage of tracing event patterns that lead to

the root cause of some system failures is highly limited. In this paper, we introduce RAS data Analysis through Visually Enhanced Navigation (RAVEN) that assists users with tracing event patterns through a context-driven navigation of RAS logs. By displaying the amount of occurrences of an event type observed during a selected time interval on the system map (physical layout of the system), RAVEN provides a compact and intuitive representation of event snapshots. By displaying pair wise contexts events or superimposing different event snapshots, users can trace coherent event correlation patterns. Also by superimposing the displacement of user applications on top of event snapshots, events or event patterns specific to a certain user application can be captured.

RAVEN has been employed at the National Institute of Computational Science (NICS) of the University of Tennessee for more than 6 months. During the period, it identified root causes of several software driven failures logged in Kraken XT5. RAVEN has also been adopted by the National Center for Computational Science at Oak Ridge National Laboratory, and used to monitor and analyze RAS logs of Jaguar and Spider.

The rest of the paper is organized as follows. First we gives background materials of RAVEN. Then the architecture of RAVEN is introduced. A detailed description of the event types used for Cray XT5 will follow. Three cases we examined with RAVEN from the Kraken and Jaguar logs will then be introduced. With discussion about the future direction of RAVEN, we will conclude the paper.

## 2. Background

RAS logs, especially those generated through *printk()* are in free text forms. Many attempts have been made to capture semantics from these logs by defining regular expressions [2, 5, 9] for the desired events. Often these regular expressions are the results of laborious efforts by human experts, and thus most reliable to detect mere occurrences of critical events. Systems such as Nagios [1] are real-time monitoring of the system based on such hard descriptions of faulty events. These tools are practically useful for immediate discovery of faults. However, they by no means be used to understand the system status or the cause of the events in a broader sense.

To provide a wider view of the system status, a number of visualization tools for monitoring log data have been introduced. OVIS [3] gives 3D visual display about state variables (temperature, fan speed, CPU utilization) and their simple statistics. By providing a close-to-real rendering of the system, it has been found to be a useful

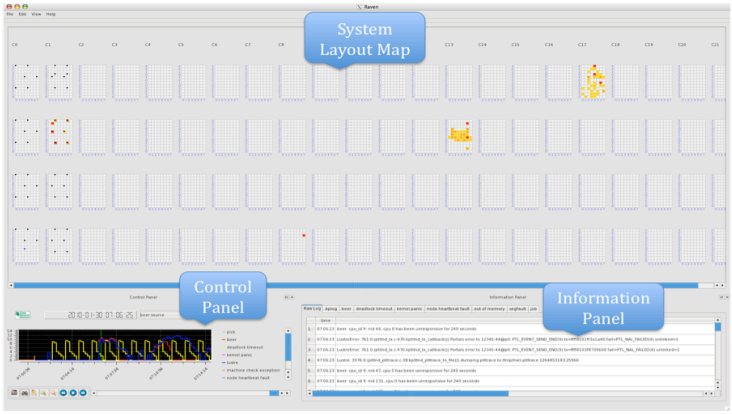


Figure 1. A screenshot of RAVEN Frontend User Interface displaying RAS logs of Kraken. The upper plane (System Layout Map) shows BEER messages between the nodes in two cabinets and Lustre OSSes. The bottom left shows the Control Panel where various event occurrence curves are displayed. The bottom right lists all the original log entries generated from a node in c13-2.

tool not only for monitoring state variables, but also tuning system set-up. Splunk aims to assist the identification of event occurrence pattern. For this, it parses logs, state variables, and other data and indexes them for an efficient searching.

To extract informative clues for the cause of a system failure, a tool that provides not only abstract views of the system but also the detailed information is desired. Most existing tools are practically useful in detecting faults, capturing system snapshots, or retrieving logs that match user defined regular expressions. However, their usages to navigate logs by tracing both temporal and spatial event patterns and retrieve detailed information of the system at the chosen location and time is highly limited.

### RAVEN Architecture

To provide a fast, intuitive, and context assisted navigation capability is the main design goal of RAVEN. RAVEN consists of two parts: the backend database server and the frontend user interface. Both the frontend and the backend were designed for fast retrieval/display of event synopsis and contexts of RAS logs.

The backend database server is a collection of MySQL tables that stores the records of events such as location of occurrences, their pair wise context, user applications, original text messages, etc. Besides these dynamic data, static information like system meta data, machine layout,

and identifiers for nodes and resources are created once during the initial installation.

From a technical point of view, the RAVEN frontend is a tool with which a user composes a query and displays the result on the physical map. A query comprises of 1) a time stamp (or an interval), 2) particular node(s), 3) specific event type, and 4) user application. Users constructs a query simply by clicking regions in the physical map, time line of interest, or choosing a particular job shown as a block of colours on the physical map.

The structure of the RAVEN frontend comprises of three parts.

1. System Layout Map
2. Control Panel
3. Information Panel

The system layout map shows the physical placement of cabinets with individual nodes depicted as squares therein. The system layout map is where events are displayed with colors representing their amounts of occurrences. The displacements of jobs are also shown in this layout map with the nodes allocated for the job having the same color.

The control navigation panel allows users to select time stamps and events of interest. To aid users to select appropriate time intervals, the panel displays a synopsis curve for each event type. It also provides multi-resolution views the curves so that users can navigate and select the most desired time stamp. When an event type and a time stamp is selected, the amounts of occurrences of the event type are shown in the system layout map.

The information panel is where information about a job or an event are detailed. When a node in the system layout map is clicked, the original log entries generated from the node or information about a job running in the node are displayed depending on user's intention.

### RAVEN on Cray XT5

RAVEN runs on CRMS log files. More specifically the backend server is built on four log files: *console, consumer, netwatch,* and *apsched*. From the first three logs, RAVEN currently extracts the following events.

- console: Lustre, Basic End to End Reliability (BEER), Segfault, Out of memory, Kernel panic, Machine Check Exception (MCE)
- consumer: node heartbeat failure, node voltage fault
- netwatch: Link Inactive, deadlock timeout

In addition to these events, as with the Simple Event Correlator (SEC), users can define and add a new event simply by providing a regular expression that describes patterns of the event in logs. Apsched logs are used to upload user application information such as duration of the run, application name, the list of nodes occupied, etc.

## 3. Case Study with RAVEN

In this section we introduce three cases when RAVEN was used to decipher the causes of abnormal behaviours of the system. In particular, we highlight how RAVEN can be used to keep track of and get an immediate view on system status. The first two cases are from logs of Kraken and the third is from that of Jaguar.

### *Case I: A Flood of Basic End to End Reliability (BEER) Messages in Kraken*

Problem Description: The NICS system administrators detected a number of periods during which abnormally huge floods of BEER messages logged. Cray engineers spent days to identify the cause of these Portal errors, tracing hardware related causes.

We examined one of the periods when such a flood of BEER messages was observed. We first displayed the distribution of BEER messages on the system layout map. Figure 2-(a) shows the nodes that are generating BEER messages, and Figure 3-(b) illustrates the nodes that are addressed to have problem with the reporting nodes. From this pair wise context (source and destination), it became clear that the problem is confined to a single application (both reporting and the reported nodes are the same). Then we checked the application displacement at the time as shown in Figure 2-(c). From this, we could identify the yellow coloured application corresponds to the nodes generating the BEER messages. We then checked Lustre messages generated during the same interval (Figure 2-(d)), and found that they were all generated by the nodes allocated for the same application (yellow coloured). This confirms that the yellow coloured application was indeed the single source of the problem.

The identified application ran for only about two minutes, and Lustre and BEER messages quieted down thereafter. However, after examining all the periods during which the same application was running by the same user, we identified that it triggered the burst of Lustre and BEER log messages during the previous week. In summary, we can identify the strange new pattern of error messages using RAVEN and guide ourselves to the cause of it. The massive errors on BEER were caused by a single job that ran in short intervals and we can trace them for every incident.

### *Case II: HSN Blackout and Recovery in Kraken*

Problem Description: Lustre error codes -107 (ENOTCONN) and -110 (ETIMEOUT), which suggest High Speed Network (HSN) issues, were observed. We confirmed that the HSN was not responding. A huge surge of Lustre messages followed and quieted down. The blackout was lifted in about 15 minitues.

By setting the clock of RAVEN to three minutes earlier than the time of the surge of Lustre messages, we found a heavy congestion on HSN by mapping "Deadlock timeout errors" on the system layout map as depicted by Figure 3-(a). On the map we found C17 column drew our attention. The congestion was more denser in the column. We then map the application displacement at the time (Figure 3-(b)). RAVEN suggested four applications for further investigation.

aprunID 1345188 yellow in upper-left corner of c17-0
aprunID 1356576 dark pink in c17-0,1,2
aprunID 1356566 grey in lower c17-3
aprunID 1350563 orange in upper c17-3

We then examined "Deadlock timeout" patterns observed when each of these application was run during the past several days, and identified that the same Deadlock timeout pattern occurred in the previous day when the same application for aprunID 1345188 was run. More specifically, during the IO cycles of the AprunID 1345188, the HSN encountered Deadlock timeouts and it quickly propagated to the column of four cabinets and over the entire system. This suggested that this particular application might have caused the blackout of the HSN.

### *Case III: A Router Node Panic in Jaguar*

Problem Description: An avalanche of Lustre and BEER message was logged from most nodes in Jaguar for more than 30 minutes. No HSN congestion was observed during the period.

By setting the time of RAVEN 6 minutes prior to the avalanche of the logs, we noticed that several OSSes of the ORNL Lustre Spider System started reporting communication problems with one particular router in Jaguar. After examining the pair wise context of all subsequent Lustre and BEER messages, we confirmed that the communication problem with the same router was reported in most of the messages; all the compute nodes are pointing to the same router. Six minutes after the OSSes reported the communication problem first, a "kernel panic" was generated at the router node. RAVEN could identify the cause of the avalanche of messages 6 minute earlier using message contexts.

## 4. Conclusion

RAS logs are often only the existing resource from which clues for a system failure or abnormal behaviours can be deduced. Due to the ever increasing scale and the complexity of supercomputers, and redundant and implicit properties of logs, analyzing RAS logs requires special attentions. This paper introduced RAVEN that aims to assist the system administrators with tracing spatial and temporal patterns. We particularly demonstrated how RAVEN can be used to identify the root cause of system problems through context-driven navigation of logs.

RAVEN has to be refined in several directions. First, currently it does not serve as a real-time monitoring/analysis tool. More specifically, the backend database server is manually updated. The real-time update of database is under development. Second, for a better portability, we are also trying to remove the backend database by connecting the RAVEN frontend directly to CMS.

## Acknowledgement

This work is supported by US DOE, Office of Science, Advanced Computing Science Research Division.

## References

[1] Wolfgang Barth, "Nagios: System And Network Monitoring", No Starch Press, 2006.

[2] Jeffrey Becklehimer, Cathy Willis, Josh Lothian, Don Maxwell, and David Vasil, "Real Time Health Monitoring of the Cray XT3/XT4 Using the Simple Event Correlator (SEC)", Cray Users Group Meeting (CUG), 2007.

[3] J. Brandt, A. Gentile, J. Mayo, P. Pébay, D. Roe, D. Thompson, and M. Wong, "Resource Monitoring and Management with OVIS to Enable HPC in Cloud Computing", 5th Workshop on System Management Techniques, Processes, and Services (SMTPS) - Special Focus on Cloud Computing, 2009.

[4] Bryan Burns, Dave Killion, Nicolas Beauchesne, Eric Moret, Julien Sobrier, Michael Lynn, Eric Markham, Chris Iezzoni, Philippe Biondi, Jennifer Granick, Steve W. Manzuik, and Paul Guersch. *Security Power Tools*. O'Reilly Media, Inc..

[5] Logsurfer - a tool for real-time monitoring of text-based logfiles, http://www.cert.dfn.de/eng/logsurf

[6] Celso L. Mendes, Daniel A. Reed, "Monitoring Large Systems Via Statistical Sampling", International

Journal of High Performance Computing Applications, 18(2), 2004.
[7] A. Oliner, A. Aiken, and J. Stearley, "Alert Detection in System Logs", In Proceedings of the International Conference on Data Mining (ICDM), 2008.
[8] A. Oliner and J. Stearley, "What supercomputers say: A Study of five sysem logs", In Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007.
[9] R. Vaarandi, "SEC-a lightweight event correlation tool", In Proceedings of IEEE IPOM'03, 2003.

## About the Authors

Byung-Hoon Park and Guruprasad Kora are staff research scientists at Computer Science Research Group. They can be reached at parkbh@ornl.gov and koragh@ornl.gov. Al Gesit is the group leader of Computer Science Research Group. He can be reached at gst@ornl.gov. Junseong Heo is a system administrator at National Institute of Computational Science (NICS), UTK. He can be reached at jheo6@utk.edu.

Figure 2. BEER and Lustre snapshots, and application displacement when Kraken is flooded with BEER messages. From top to bottom, (a) Nodes generating BEER messages (source nodes), (b) Node the BEER messages are reporting (destination nodes), (c) Nodes generating Lustre messages, (d) application displacement. From the figures, it is clear that both BEER and Lustre messages are generated from the same application (colored yellow)

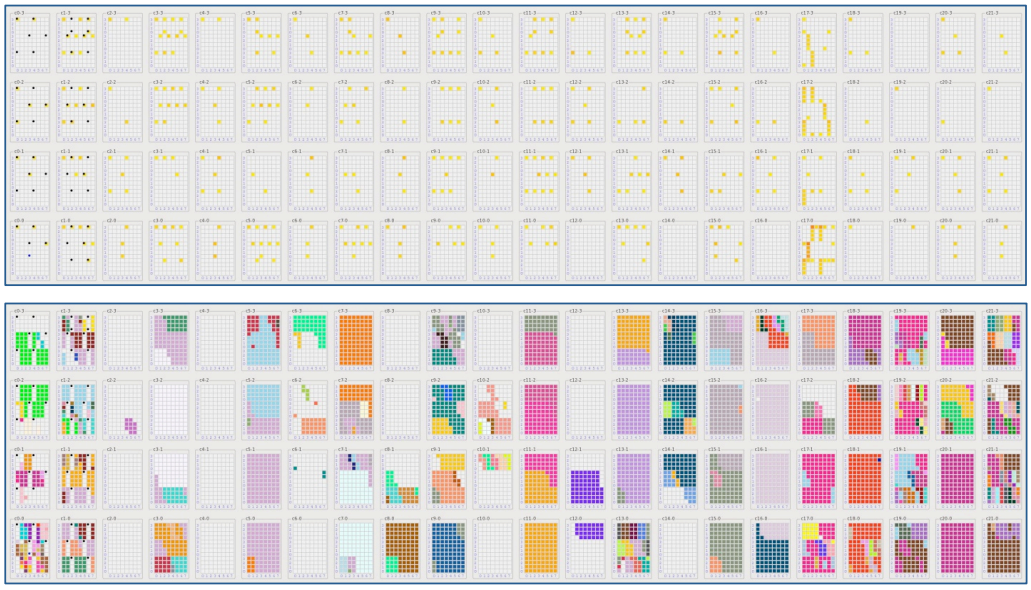

Figure 3. (a) Deadlock Timeout snapshot when HSN of Kraken is not responding (Top). (b) Application displacement (Bottom). Congestion is most severe at the column of C17.