Framework for Enabling System Understanding

J. Brandt[1], F. Chen[1], A. Gentile[1], C. Leangsuksun[2], J. Mayo[1], P. Pebay[1], D. Roe[1], N. Taerat[2], D. Thompson[1], and M. Wong[1]

[1] Sandia National Laboratories, Livermore CA, USA,
brandt|fxchen|gentile|jmayo|pppebay|dcroe|dcthomp|mhwong@sandia.gov**
[2] Louisiana Tech University, Ruston, LA, USA
box|nta008@latech.edu

Abstract. Building the effective HPC resilience mechanisms required for viability of next generation supercomputers will require in depth understanding of system and component behaviors. Our goal is to build an integrated framework for high fidelity long term information storage, historic and run-time analysis, algorithmic and visual information exploration to enable system understanding, timely failure detection/prediction, and triggering of appropriate response to failure situations. Since it is unknown what information is relevant and since potentially relevant data may be expressed in a variety of forms (e.g., numeric, textual), this framework must provide capabilities to process different forms of data and also support the integration of new data, data sources, and analysis capabilities. Further, in order to ensure ease of use as capabilities and data sources expand, it must also provide interactivity between its elements. This paper describes our integration of the capabilities mentioned above into our OVIS tool.

Keywords: resilience, HPC, system monitoring

1 Introduction

Resilience has become one of the top concerns as we move to ever larger high performance computing (HPC) platforms. While traditional checkpoint/restart mechanisms have served the application community well it is accepted that they are high overhead solutions that won't scale much further. Research into alternative approaches to resilience include redundant computing, saving checkpoint state in memory across platform resources, fault tolerant programming models, calculation of optimal checkpoint frequencies based on measured failure rates, and failure prediction combined with process migration strategies. In order to predict the probability of success of these methods on current and future systems we need to understand the increasingly complex system interactions and how they relate to failures.

** These authors were supported by the United States Department of Energy, Office of Defense Programs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL8500.

The OVIS project [13], at Sandia National Laboratories, seeks to develop failure models that, combined with run-time data collection and analysis, can be used to predict component failure and trigger failure mitigating responses such as process migration, checkpointing, etc. OVIS [2, 3] is comprised of five main components: 1) numeric data collection and storage infrastructure, 2) text based information collection and storage, 3) numeric analysis and response, 4) text based analysis, and 5) user interface and visualization. In this paper we first describe the current OVIS infrastructure with respect to these components including new capabilities for data/information gathering and interactive exploration that have been recently developed, including integration of the Baler [5] tool and development of Baler visualizations. Next we provide use case context for its utility in enabling researchers to aggregate available system related information into a single infrastructure and manipulate it to gain insight into operational behaviors related both to normal and failure modes. We contrast this with other currently available tools built for providing similar insight and discuss their strengths and shortcomings.

2 Approaches

2.1 Numeric Information Collection and Storage

Numeric information is gathered from various sources in different ways as described in these subsections. Storage however is all accomplished through insertion into a distributed database where division of storage is done on the basis of which compute node the information is being collected on behalf of (e.g. Node A's numeric information will be stored into the same database whether it is collected raw from the node itself or harvested from log files stored on other servers. Scalability of the system is accomplished via this distributed storage architecture in that there is no duplication of raw information across the databases.

User-defined Samplers OVIS provides an interface by which the user can write information samplers. The OVIS framework handles invocation of the samplers and insertion of the resultant data into the OVIS database. OVIS is released with samplers that process, for example, `/proc`, lmsensors [11], and EDAC [8] data. Recent advancements include flexible regular-expression matching for dynamic discovery of data fields within known sources and of relative system components to ease user development of samplers.

Samplers can collect data at regular intervals or be prodded to sample on demand. Examples, motivated by a previous work where a system was insufficiently releasing claimed memory [1] include prodding a sampler to collect Active Memory utilization during the epilog script of a job (to determine if memory was released) or before the resource manager places a new job on a node (to determine if that node should be allocated or instead placed offline). An additional sampler exists that can record a string into the database on demand. While the current OVIS analyses operate on numeric data, this timestamped text data can be used as a marker by which to guide analyses. For example, one could insert

markers for the entry and exit of a particular part of a code and then use that information to determine the times for an analysis.

While these samplers currently run a database client on the nodes, we have recently implemented a distributed metric service which propagates information to the database nodes itself for insertion.

Metric Generator OVIS provides a utility called the Metric Generator that enables a user to dynamically create new data tables and insert data. The interface allows the user to specify optional input data, a user-defined script to generate the additional data, and a new output data table. The script can operate on both metrics in the database and on external sources. Examples include a) scripts that grep for error messages in log files and insert that occurrence into the database, thus converting text to numeric representations and b) more sophisticated analyses, such as gradients of CPU temperature. This enables rapid development of prototype analyses. These scripts can be run at run-time or after-the fact still with the ability to inject data timestamped to be concurrent with earlier data in the system.

Resource Manager Resource Manager (RM) data includes information about job start and end times, job success/failure, user names, job names etc. OVIS does not collect RM data, but rather intends to natively interface to a variety of RMs' native representations. Currently, OVIS natively interfaces to databases produced by SLURM [15] (and associated tools) and enables search capabilities upon that data as described in Section 2.4.

2.2 Text Based Information Collection and Storage

Text based information is currently collected from three sources: resource manager logs, syslogs, and console logs.

As OVIS is intended to natively interface with system resource managers (Section 2.1), the only reason storage would be needed is if, for performance reasons, the system administrators would prefer the RM's database to be replicated. Replication can be accomplished either to a separate RM database node or to one of the database nodes being used for numeric storage depending on the relative amount of activity in each.

OVIS uses LATech's Baler [5] log file analysis engine as well as custom metric generator scripts to interact with both syslog and console log files. Thus collection and storage of this information is performed in two ways and incurs the cost of some redundancy for performance purposes. These log files are typically stored in flat file format on a server associated with a particular HPC system. Thus we periodically copy these files from the server to one of the database nodes in our system as flat files.

2.3 Numeric Analysis

In order to provide a standard interface for development of new analysis engines for manipulating numeric information and to leverage existing high performance

parallel analysis engines we adopted the visualization tool kit (VTK) statistical analysis interface. Current analysis engines available in OVIS are: descriptive statistics, multi-variate correlation, contingency statistics, principal component analysis, k-means, and wear rate analysis. These analyses allow the user to understand statistical properties and relative behaviors of collected and derived numeric information which can in turn provide insight into normal vs. abnormal interactions between applications and hardware/OS. These analysis engines can be accessed via a python script interface, or a high performance C++ interface either directly or through OVIS's GUI. Additionally the user can write scripts, described in Section 2.1, that manipulate numeric information from any source.

2.4 Text Based Analysis

Resource Manager RM information analysis capabilities include the ability to search on various quantities such as user name, job end state, resources participating in a job etc; and drill-down graphical capabilities for displaying the distribution of users, end states, etc. for a set of jobs; and some basic statistics for a set of jobs such as size and duration for a set of jobs. In general, such information is insufficient as a sole source of exploratory information. For example, in the out-of-memory analysis [1], the on-set of the problem condition actually occurs during jobs that successfully complete, with job failure as a longer-term effect. The Resource Manager analysis capability rather is intended to provide information that can be used to guide analysis. For example, an analysis of CPU Utilization may be relevant only over the processors of a given job and for the duration of that job; job failure information can be used to narrow down times and nodes of interest in which to search.

Baler Log File Analysis Textual analysis is provided within OVIS via integration of the Baler [5], a tool for log file analysis. Baler discovers and extracts patterns from log messages with the goal of reducing large log data sources into a reasonable number of searchable patterns. Its algorithm is efficient compared to existing log analysis tools in that it requires only one pass as opposed to contingency statistics algorithms which require several passes over the data. Baler generates patterns based on message context resulting in deterministic output patterns irrespective of input data variance, unlike contingency statistics algorithms which depend upon the variance. This results in a consistent pattern representation for a given message over time, facilitating consistent understanding over long-term data exploration. Baler stores pattern representations as unique numbers; such time-pattern-number tuples are consistent with the OVIS metric data storage. Thus complete integration of OVIS and Baler will enable the same statistical analysis and visualization capabilities for log file analysis as OVIS currently supports for its numerical data.

3 Applications

This section describes use of OVIS integrated, interactive capabilities to enable system understanding.

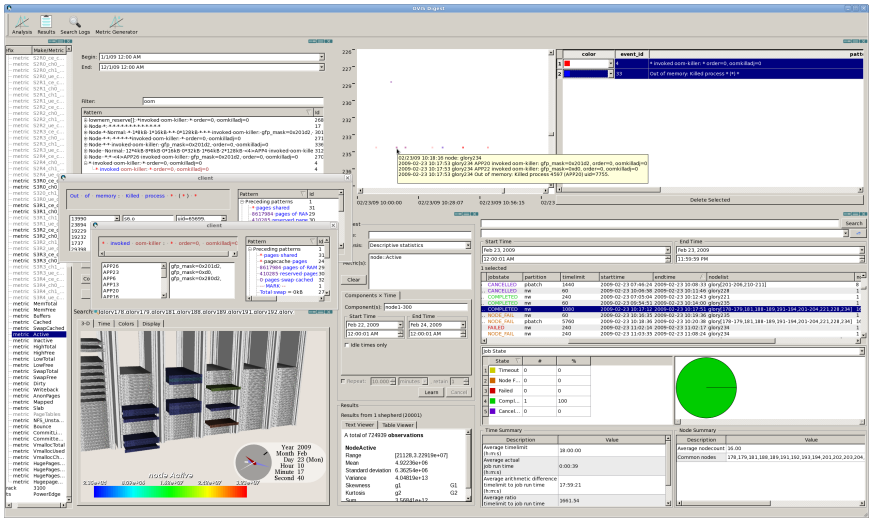

Fig. 1: OVIS-Baler integration user interface. The OVIS-Baler integration provides interacting capabilities for log pattern analysis and visualization, numerical data analysis and visualization, and job log search.

System diagnosis of precursor to failure In previous work [1], we demonstrated a statistically discoverable precursor to failure exists for one of our clusters. Proof of that the discoverable condition was related to failure required cross reference with job and log data. At the time, this cross referencing was a manual process, not integrated within OVIS. We have developed the additional capabilities within the OVIS framework to enable the required cross referencing with job and log data.

This is illustrated in Figure 1 which is the screenshot of the integrated capabilities addressing system data exploration. Zoom-ins on various sections are presented in the subsequent figures. In the lower right and in Figure 2, the log search is used to investigate the status of jobs. Note that while the highlighted job completes, subsequent jobs on Glory 234 fail. In the upper left and in Figure 3 (top), the Baler patterns are displayed. Results for both the `oom-killer` and the `out of memory` patterns are displayed. In previous work [5], the Baler pattern for `oom-killer` was presented, contrasting with other tools where the pattern was either missed or presented in a redundant fashion so that it was difficult to determine the number of oom killer events. The ability of Baler to discover patterns, as opposed to merely enabling filtering on pre-defined patterns makes it possible to obtain understanding for this problem that might have otherwise been missed. The occurrences of the patterns with node and time information are displayed in the upper right and in Figure 3 (bottom). No error messages with these patterns occur during the completed job, however the error messages occur during the subsequent failed job, as is explicitly presented in the mouse over. The lower left and in Figure 4 includes the OVIS display where system data can

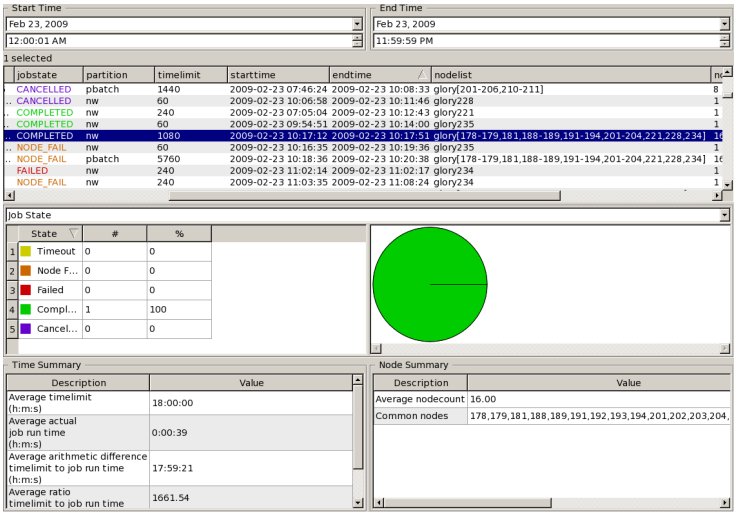

Fig. 2: OVIS Resource Manager (RM) view. Job information is searchable and is shown. Selecting a job automatically populates an analysis pane and the 3D view with job-relevant data (Figure 4).

be displayed on a physically accurate representation of the cluster. Job-centric views are supported as in this figure where the highlighted (completed) job in the job-search display is dropped upon the physical display, limiting the colored nodes to only those participating in the job. It is seen that one of the nodes (Glory 234, colored red and circled) has significantly higher Active Memory than any of the other nodes participating in the job. Scrolling through time indicates that the node has high Active Memory, even during idle times on the node, and during the subsequent failed job.

Note that any one data source is insufficient to understand the entire situation. The Resource Manager data shows that there may be a problem on Glory 234, but it does not elucidate the cause of the problem. The log data shows that a possible cause of job failure is an out of memory condition on Glory 234, but it does not indicate the onset of the problem, nor if this is this due to a naturally occurring large demand for memory. The physical visualization with outlier indication shows the onset and duration of the abnormal behavior on the node, but does not directly tie it to a failure condition. The combination of all three pieces, each providing a different perspective and each working upon a different data source is necessary for overall system understanding. (this is in contrast to the condition detection itself, which can be done purely by statistical means).

Alternative views of similar information In some cases the same or similar information is available, but through different sources or in different formats. For instance the Error Detection and Correction (EDAC) [8] Memory Controller (MC) driver module provides an interface to DIMM error information. This in-

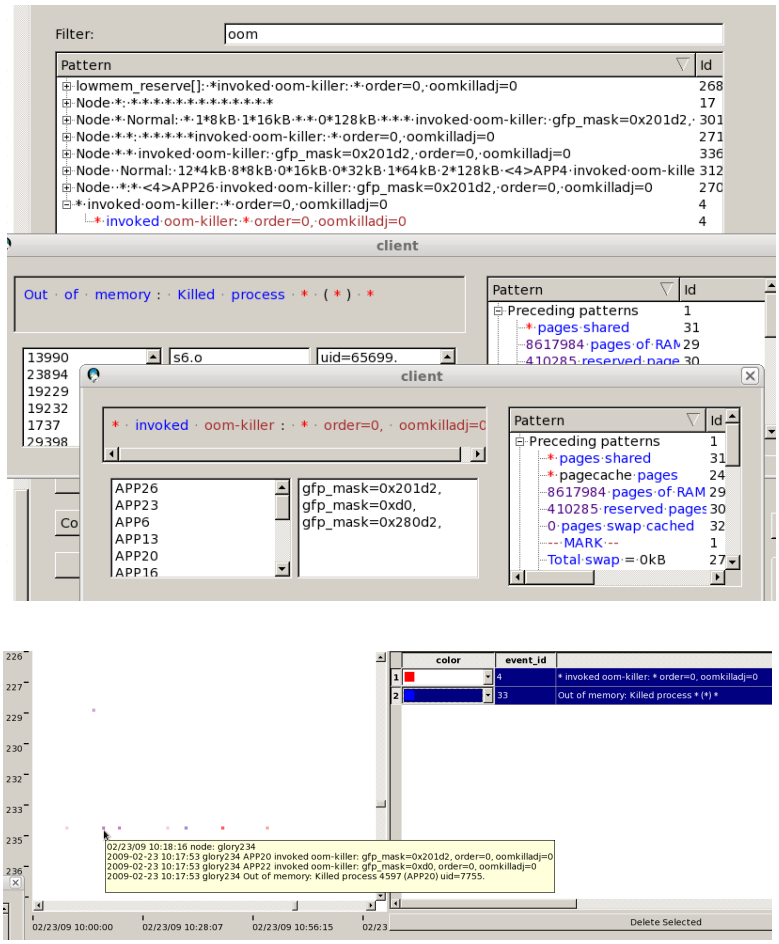

Fig. 3: Baler Pattern view (top) provides search capabilities and drill down viewing of wild card values. Out of Memory related meta-patterns, determined by Baler are shown. Baler Event view (bottom) shows color coded events in time and space. Mouseover highlights and displays message patterns. Some messages for the out of memory condition on a node are shown.

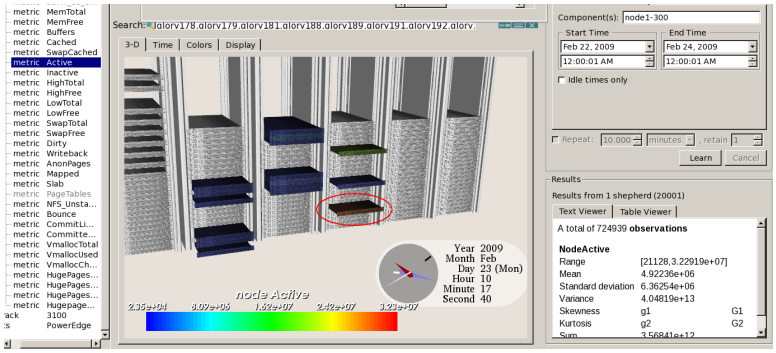

Fig. 4: OVIS physical cluster display. An outlier in Active Memory is seen (red, circled) across nodes in this job (colored nodes). Job selection in the RM view (Figure 2) automatically populates a) the analysis pane with relevant nodes and time and b) the 3D view with nodes.

formation is made available through system files, command line interface, and in the syslog files. In the system file representation a separate file is written for each row and channel, which can be mapped to slots for DIMMS and/or physical CPU sockets. Each of these files (e.g., `.../edac/mc/mcX/csrowY/ue_count`) containers counters of errors that have occurred. This same information can be extracted via the command line calls. In the syslog output, such errors are reported as: `Feb 20 12:41:22 glory259 EDAC MC1: CE page 0x4340a1, offset 0x270, grain 8, syndrome 0x44, row 1, channel 0, label DIMMB 2A: amd64_edac`. Thus presenting the row, channel, DIMM, and error categorization, but in a different format.

In OVIS the same innate information is harvested from the two different sources, and it is processed in complimentary and different fashion. Baler reduces specific occurrences of this pattern to: `EDAC *: CE page * offset * grain *, syndrome * row * channel * label * *: amd64_edac` with the `*` indicating wild card entries. This enables the user to get the number and, in the display, the node-level placement of the memory errors with respect to time. However, it hides the exact mapping to the hardware. While that information could be extracted, such a procedure would be antithetical to the goal of Baler, which seeks to provide pattern information to the user without requiring input on the message format. In contrast, OVIS samplers collect the EDAC information via the file sources inherently with the hardware associations. OVIS is capable of presenting this information at the socket of DIMM level. In general OVIS will collect and enable investigation at as high a fidelity representation of the display as the user cares to provide.

4 Related Work

There has been much work done and various tools built within the HPC community with respect to information collection, analysis, and visualization some of which we describe here. In each case, however, only a portion of the wealth of information available has been harvested and hence the understanding that can be realized is limited. By contrast we seek through the OVIS project to create an integration framework for available information and tools to, through knowledge extraction and system interaction, build more resilient HPC platforms.

Both Ganglia [9] and VMware's Hyperic [10] provide a scalable solution to capturing and visualizing numeric data on a per-host basis using processes running on each host and retrieving information. While Ganglia uses a round robin database [14] to provide fine grained historic information over a limited time window and coarser historic information over a longer time Hyperic uses a server hosted database. Each retains minimal information long term. While Hyperic, unlike Ganglia, supports job based context in order to present more job centric analysis or views, neither has support for complex statistical analysis. Ganglia is released under a BSD license making it an attractive development platform while Hyperic releases a stripped down GPL licensed version for free and a full featured version under a commercial license.

Nagios [12] is a monitoring system for monitoring critical components and their metrics and triggering alerts and actions based on threshold based directives. It provides no advanced statistical analysis capability nor the infrastructure for performing detailed analysis of logs, jobs, and host based numeric metrics in conjunction. Nagios Core is GPL licensed.

Splunk [16] is an information indexing and analysis system that enables efficient storage, sorting, correlating, graphing, and plotting of both historic and real time information in any format. Stearley et al give a variety of examples of how Splunk can be used to pull information from RM's, log files, and numeric metrics and present a nice summary including descriptive statistics about numeric metrics [4]. Some missing elements though are numeric data collection mechanisms, spatially relevant display, and a user interface that facilitates drag and drop type exploration. Like Hyperic, Splunk provides a free version with limited data handling capability and limited features as well as a full featured commercial version where the cost of the license is tied directly to how much data is processed.

Analysis capabilities outside of frameworks exist. Related work on algorithms for log file analysis can be found in [5], as the algorithmic comparison is not directly relevant to this work. Lan et al have explored use of both principal component analysis and independent component analysis [7] [6] as methods for identifying anomalous behaviors of compute nodes in large scale clusters. This work shows promise and would be more useful if the analyses were incorporated into a plug and play framework such as OVIS where these and other analysis methods could be easily compared using the same data.

5 Conclusions

While there are many efforts under way to mitigate the effects of failures in large scale HPC systems, none have built the infrastructure necessary to explore and understand the complex interactions of components under both non-failure and failure scenarios nor to evaluate the effects of new schemes with respect to these interactions. OVIS provides such an infrastructure with the flexibility to allow researchers to add in new failure detection/prediction schemes, visualize interactions and effects of utilizing new schemes in the context of real systems either from the perspective of finding when prediction/detection would have happened and validating that it is correct or by comparing operation parameters both with and without implementation of such mechanism(s).

References

1. Brandt, J., Gentile, A., Mayo, J., Pebay, P., Roe, D., Thompson, D., Wong, M.: Methodologies for Advance Warning of Compute Cluster Problems via Statistical Analysis: A Case Study. Proc. 18th ACM Int'l Symp. on High Performance Distributed Computing, Workshop on Resiliency in HPC, Munich, Germany (2009).
2. Brandt, J., Gentile, A., Houf, C., Mayo, J., Pebay, P., Roe, D., Thompson, D., Wong, M.: OVIS-3 User's Guide. Sandia National Laboratories Report, SAND2010-7109 (2010).
3. Brandt, J., Debusschere, B., Gentile, A., Mayo, J., Pebay, P., Thompson, D., Wong, M.: OVIS-2 A Robust Distributed Architecture for Scalable RAS. Proc. 22nd IEEE Int'l Parallel and Distributed Processing Symp., 4th Workshop on System Management Techniques, Processes, and Services, Miami, FL (2008).
4. Stearley, J., Corwell, S., Lord, K.: Bridging the gaps: joining information sources with Splunk. Proc. Workshop on Managing Systems Via Log Analysis and Machine Learning Techniques, Vancouver, BC, Canada (2010).
5. Taerat, N., Brandt, J., Gentile, A., Wong, M., Leangsuksun, C.: Baler: Deterministic, Lossless Log Message Clustering Tool. Proc. Int'l Supercomputing Conf., Hamburg, Germany (2011).
6. Lan, Z., Zheng, Z., Li, Y.: Toward Automated Anomaly Identification in Large-Scale Systems. IEEE Trans. on Parallel and Distributed Systems 21 (2010) 174-187.
7. Zheng, Z., Li, Y., Lan, Z.: Anomaly localization in large-scale clusters. Proc. IEEE Int'l Conf. on Cluster Computing (2007).
8. EDAC. Error Detection and Reporting Tool see, for example, Documentation in the Linux Kernel [linux/kernel/git/torvalds/linux-2.6.git]/Documentation/edac.txt.
9. Ganglia. http://ganglia.info.
10. Hyperic. VMWare. http://www.hyperic.com.
11. lm-sensors. http://www.lm-sensors.org/.
12. Nagios. http://www.nagios.org.
13. OVIS. Sandia National Laboratories. http://ovis.ca.sandia.gov.
14. RRDtool. http://www.rrdtool.org.
15. SLURM. Simple Linux Utility for Resource Management. http://www.schedmd.com.
16. Splunk. http://www.splunk.com.