Real Time Health Monitoring of the Cray XT3/XT4 Using the Simple Event Correlator (SEC)

Jeffrey Becklehimer, Cathy Willis, *Cray Inc;* **Josh Lothian, Don Maxwell, and David Vasil** , *Oak Ridge National Laboratory (ORNL)*

ABSTRACT: *The log files produced by the Cray XT contain all of the known events which have occurred on the system. Examples of logged events include a memory fault, a node panic, a link failure or the successful launch and execution of a user's application. When the system is experiencing problems the standard procedure is to perform a system dump and then examine the dump in hope of finding the cause. We have constructed a framework using the Simple Event Correlator (SEC) to monitor the log files in real time and to report events which are known to cause problems. The result is that many problems are now known before the users ever report a problem. This framework can be easily customized by the site to search for events of local interest.*

KEYWORDS: XT, CRMS, SEC

1. Introduction

In 2005, the National Center for Computational Sciences (NCCS) at the Oak Ridge National Laboratory (ORNL) installed their first Cray XT3 computer. Since that initial installation the system has been on a very aggressive upgrade path of performance improvements. The computer has grown from a system of an initial 25 Teraflops (TF) to 54 TF and is currently 119 TF. These upgrades mean we have had to go through many periods of troubleshooting and stabilization. This paper presents one of the tools that has been implemented to help the system administrators in their speed and efficiency of diagnosing and repairing system problems.

In addition to the number of upgrades in the past two years, the size of the resulting computer introduces new issues related to system health monitoring. The Cray XT computer located at ORNL is called Jaguar. Jaguar consists of 124 cabinets; 56 cabinets are XT3 and 68 are XT4. There are 11,708 nodes in the system. 11,508 are compute nodes and the remainder are service I/O (SIO) nodes. Each compute node has a 2.6 GHz dual-core AMD Opteron processor and 4 GB of memory.

The system is physically configured as 4 rows with 31 columns of racks. It has a full 3D torus that is logically a 31x16x24 cube.

In general, the XT is quite verbose in its logging. Early in the development it was learned that manual analysis of the log files was tedious and time consuming. As the system grew larger, the log files became voluminous and faced with an intractable and growing problem.

Realizing that an automated solution was needed the staff and system administrators began discussing possible solutions. There was agreement to write as little as code as possible and to leverage open source where applicable.

2. Cray RAS and Management System

The Cray RAS and Management System (CRMS) is the administrative interface into the hardware and is responsible for powering up, booting, monitoring and dumping the system. It consists of three interconnected subsystems: the CRMS hardware, the CRMS software and the CRMS event logging.

2.1 Hardware

There are 4 components to the CRMS hardware. The CRMS network, the System Management Workstation (SMW), the cabinet control processor (L1) and the module control processor (L0). Together these components monitor the health and status of the compute/service nodes, the attached RAID storage subsystems, the node temperatures, voltages, fan speeds and node heartbeats.

Each module in the system has a control processor called a L0. This processor is responsible for monitoring the components of the module, such as checking Opteron and Seastar registers. The L0 also monitors the voltage regulator modules (VRMs), temperatures and the operating system heartbeat.

Similarly, each cabinet has a L1 controller. It monitors the power supplies, regulates fan speeds and does other cabinet-related monitoring. It also routes CRMS traffic between the SMW and L0s.

The System Management Workstation (SMW) is a desktop workstation that is used to manage, boot and monitor all of the XT components. It has network interfaces into the users network, CRMS network and the XT boot node.

The CRMS network is a flat 100-BaseT private Ethernet network that carries messages between the SMW, L1 and L0 controllers.

.

2.2 Software

The CRMS software consists of resiliency communication agents (RCA); the administrator's CRMS interfaces; and event probes, loggers and handlers.

The RCAs run on every node in the system. They provide the interface between the operating system and CRMS components external to the node. RCAs also monitor software services and the operating system instance on each node. PBS (Portable Batch Scheduler) and LLRD (Lustre Lock Recovery Daemon) are examples of software services that run under RCA control.

The RCAs are also responsible for generating heartbeats. If the CRMS does not detect heartbeats coming from an RCA, the CRMS will conclude that the node has failed and mark it down.

The CRMS command *xtcli* is a command line utility that is used to power up, boot, dump and power down the system.

2.3 CRMS Event Logging

The CRMS event logger runs on the SMW and logs all status and event data generated by CRMS processes running on the L1/L0s and by the RCA daemons running on compute and service nodes. This data is logged to a file in the */opt/craylog* directory. These files are rotated based on a maximum file size parameter.

During the boot process three watcher programs are initiated. These programs subscribe to the event log coming from the L1/L0s and log that data to different files in a more verbose manner.

The program *xtconsumer* monitors CRMS events and logs its output to */opt/craylog/bootlog/consumer*.YYMMDDHHMM where YYDDMMHHMM is the year,month,day,hour,minute boot timestamp. The events of interest that appear in this file are node and Seastar failures.

The *xtconsole* utility is capable of displaying the console text of all the nodes. The text gets logged to */opt/craylog/bootlog/console*.YYMMDDHHMM. Kernel panics, portals errors and memory errors can be found in this log file.

The *xtnetwatch* program logs errors associated with the high speed network (HSN). This log will contain both recoverable and fatal link errors and their locations. This log is located at */opt/craylog/bootlog/netwatch*.YYMMDDHHMM

3. Simple Event Correlator (SEC)

3.1 Introduction

SEC is an open source and platform independent event correlation tool that was designed to fill the gap between commercial event correlation systems and home-grown solutions that usually comprise a few simple shell scripts. SEC and its documentation can be found at http://www.estpak.ee/~risto/sec/.

SEC accepts input from regular files, named pipes, and standard input. Thus, SEC can be employed as an event correlator for any application that is able to write its output events to a file stream. The SEC configuration is stored in text files as rules, each rule specifying an event matching condition, an action list, and optionally a Boolean expression whose truth value decides whether the rule can be applied at a given moment. Regular expressions, Perl subroutines, etc. are used for defining event matching conditions. SEC can produce output events by executing user-specified shell scripts or programs (e.g., snmptrap or mail), by writing messages to pipes or files and by various other means.

3.2 Event Correlation Rule Types

The following event correlation rule types are currently implemented in SEC:

Single - match input event and execute an action list.

SingleWithScript - match input event and execute an action list, if an external script or program returns a certain exit value.

SingleWithSuppress - match input event and execute an action list, but ignore the following matching events for the next t seconds.

Pair - match input event, execute an action list, and ignore the following matching events until some other input event arrives. On the arrival of the second event execute another action list.

PairWithWindow - match input event and wait for t seconds for other input event to arrive. If that event is not observed within the given time window, execute an action list. If the event arrives on time, execute another action list.

SingleWithThreshold - count matching input events during t seconds and if a given threshold is exceeded, execute an action list and ignore the following matching events during the remaining time window. The window of t seconds is sliding.

SingleWith2Thresholds - count matching input events during t1 seconds and if a given threshold is exceeded, execute an action list. Then start the counting of matching events again and if their number per t2 seconds drops below the second threshold, execute another action list. Both event correlation windows are sliding.

Suppress - suppress matching input event (used to keep the event from being matched by later rules).

Calendar - execute an action list at specific times.

Rules allow not only shell commands to be executed as actions, but they can also:

- create and delete contexts that decide whether a particular rule can be applied at a given moment,
- associate events with a context and report collected events at a later time (similar feature is supported by logsurfer),
- generate new events that will be input for other rules,
- reset correlation operations that have been started by other rules,
- spawn external event, fault, or knowledge analysis modules.

This makes it possible to combine several rules and form more complex event correlation schemes.

3.3 Example rule

Below is a simple example of a SEC rule.

```
#
# Detect nodes that panic
#
type= SingleWithSuppress
window= 120
ptype= RegExp
pattern= \[([0-9A-z._-]+)  (\d\d:\d\d:\d\d)\]\
[([0-9A-z._-]+)\]0- PANIC_SP
desc= $1 $2 Node $3 Paniced
action=  add PANICED_$3  $1 $2 Node $3 Paniced; \
         report PANICED_$3 /bin/mail -s "Hood
Node $3 Paniced"
root@jaguar.ccs.ornl.gov
```

As is common in many scripted languages, the use of the "#" in SEC indicates a comment. This SEC rule attempts to detect and report nodes that panic.

The first two lines after the comments in this example indicate that this panic rule will trigger on a single event and suppress duplicates for a window of 120 seconds. Panics often result in repetitive lines filling log files within the first few minutes of the panics before the node dies.

The event is defined by a regular expression. That regular expression is defined in `pattern`. When the pattern is matched, an email will be sent with the physical address of the node that paniced.

4. XTSec

Using standard SEC rules and events an XT-specific event list and XT-specific event scripts were created to handle those events. The initiation and shutdown of SEC has also been automated. Collectively this group of rules and scripts are called XTSec.

4.1 XT Event List

Using standard SEC, rules can be constructed to report just known errors or to remove routine traffic from the logs.. Filtering out normal events allows

administrators to more quickly and efficiently find new and unexpected problems. However, the task of defining what is "normal traffic" can be daunting. It was decided to focus efforts on reporting events known to cause system problems.

In order to deploy SEC to monitor the XT a list of errors to report was created. The list was generated by the staff in cooperation with the system administrators based on their experience triaging past failures. It was decided that this list would first contain system events which require the system to be rebooted or that could cause user jobs to fail. This list is by no means exhaustive nor static. New rules get added as new failure modes are discovered. The initial list included:

Link Inactive
RX message header CRC error
RX message CRC error
Recv Sequence Error
Send Buffer Overrun
MCA Error
QK Panic
Node Corefail
Node Thermtrip
VDDIO Fail
Verty Fail
Voltage Faults
Seastar HB Faults
Node HB faults
SSNAL Looping too Long
SCSI Errors

Analysis of a single event can often be misleading or not provide all the information required to diagnosis a problem. In these situations event threading or grouping can be especially useful. A good example of this would be a node panic. It would be useful to not only know that a node paniced but to also gather all the information about what job was running during the panic and to see the entire traceback. No event threading has been attempted in this first attempt but many opportunities exist to implement threading.

4.2 Event Scripts

Some of the XTSec rules make use of the *SingleWithScript* rule type. These rules require helper scripts to do further processing. Examples of these scripts are:

Decode MCA – this script is invoked upon an MCA error. It decodes the error to determine if the error is correctable or not. If uncorrectable it will send out a notification

Get Node HDT – this script is called when a node fails. It will read information from a module L0 to determine the health of the hardware on the node.

Get Node SS – this script processes Seastar faults. It will gather information from the Seastar buffer that is used later for diagnosis.

4.3 Utility Scripts

To aid in further automation of XTSec, several utility scripts have been created.

A startup script has been created and is optionally used in the autoboot sequence. Similarly, a shutdown script has been inserted into the xtshutdown configuration. The shutdown script can also be run independently.

Likewise, a restart script is available to stop and retstart SEC. This is used to insert new rules into a running configuration.

7. Conclusion

SEC has proven itself to be an ideal framework for implementing a real time monitoring system for the Cray XT class of computers. Rules can be rapidly implemented by staff and administrators with a basic understanding of regular expressions. The use of SEC with XTSec rules and scripts has proven to be very effective at improving real time health monitoring. It has been so effective that in most cases the administrators are now aware of problems before users begin to notice issues.

Acknowledgments

The authors would like to thank the staff and colleagues who have contributed material to this paper. Also, we would like to thank Risto Vaarandi the author of SEC for providing such a useful tool.

About the Authors

Jeff Becklehimer is a Principal Engineer with Cray Inc. He can be reached by E-Mail at jlbeck@cray.com. Cathy Willis is a on-site Systems Analyst with Cray Inc. Don Maxwell, Josh Lothian and David Vasil are all staff members of the National Center for Computation Sciences (NCCS) at Oak Ridge National Laboratory (ORNL).