A Preliminary Report on Red Storm RAS Performance

Jon Stearley and Robert A. Ballance
*Sandia National Laboratories**

April 28, 2006

Abstract

The Cray XT3 provides a solid infrastructure for implementing and measuring measuring reliability, availability, and serviceability (RAS). A formal model interpreting RAS information in the context of Red Storm was presented at CUG-2005. This paper presents an implementation of that model - including measurements, their implementation, and lessons learned.

1 Background

The high performance computing community would be well-served by establishing standardized definitions and metrics for supercomputer RAS performance [1]. Towards this goal, a specification for measuring the RAS performance of Red Storm has been proposed [2]. This report is a direct follow-on, presenting recent Red Storm RAS performance in a well-defined and quantitative manner, as well as descriptions of the processes and tools used in the assessment. Please refer to the preceding references for clear definitions of all terms and metrics used in this report.

Prior to 2006-01-01, Red Storm was configured to run varying hybrids of Sandia and Cray system software. For this paper we have chosen to present only the data since 2006-01-01 (we have data since 2005-10-01). Restricting the date range in his manner enables a concise report on Cray-only software.

2 System Configuration and State

Red Storm is a distributed memory, massively parallel supercomputer modeled on ASCI Red. System components can be functionally divided into two groups, the compute partition (where the computation occurs) and the service partition (which provides services such as login, compile, I/O, etc). Furthermore, Red Storm is a dual-headed machine, being split between classified (Red) and unclassified (Black) use. Each end is anchored in a specific network. There are three distinct compute partitions that can be attached to either head. Thus, there are four typical operating configurations:

Name	Unclassified Compute Nodes	Classified Compute Nodes
Unclassified Jumbo	10,368	0
Unclassified Large	7,680	2,688
Classified Large	2,688	7,680
Classified Jumbo	0	10,368

The service partition at each head remains available to users, even when there are no compute nodes available. For instance, in Jumbo Unclassified mode, users can still access their files on the Classified head.

While Cray has provided extensive support for physically separating the two systems, the current release of the system management tools does not adequately represent the planned and historical configuration of the system. For this reason, we maintain an additional, auxiliary data source describing the configuration for each time period.

System state is tracked according to the state diagram shown in Figure 1. Time periods spent in Engineering Time, Unscheduled Downtime, or Scheduled Downtime are reported via email by system operators, and then reviewed and manually recorded (as "incidents") by the system operations manager. All other time is counted as Production Time. These state transition records are used to produce system Availability and MTBIservice plots. Additional state transition attributes are also recorded, such as if the system was dedicated for use by a single user (optional attribute of Production Time), purpose regarding Engineering and Scheduled Downtimes, the cause of Unscheduled Downtimes, etc.

3 Availability

Understanding how much time the system spends in each state is useful for accounting, but also in order to set expectations regarding the likelihood that it will be available to meet one's needs in the future. System Availability plots provide this information (see Figure 2).

Plot Description

"Production Availability" is calculated as the total number of hours the system was in a Production Time state divided by Total Time,

*Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

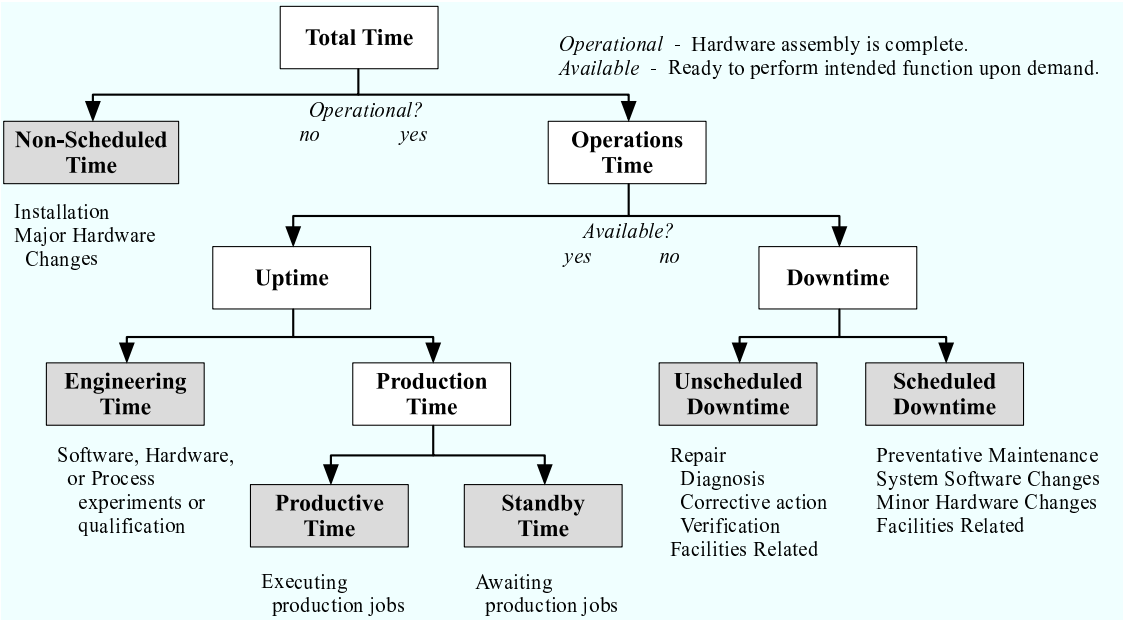

Figure 1: State Diagram

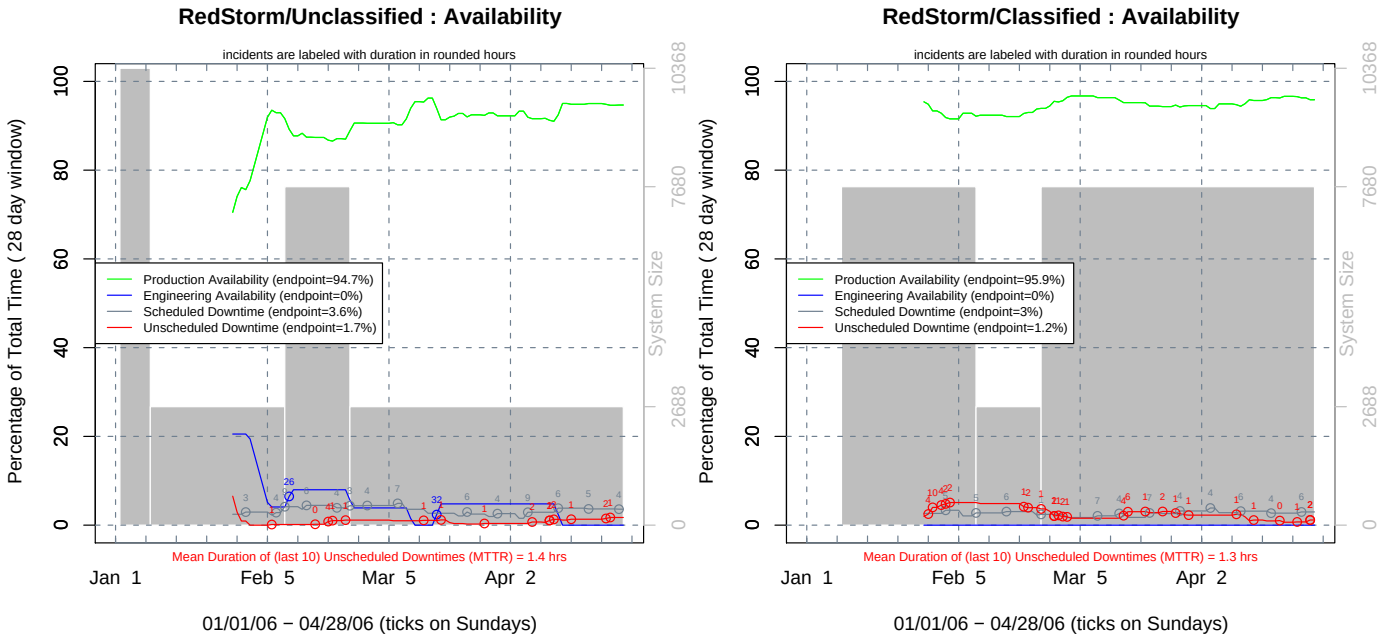

Figure 2: System Availability Plots

using a twenty-eight day window. Other plotted quantities are calculated analogously. Hours per state per day is used in order to show per-day sensitivity in the plot, and a twenty-eight day window is used in order to show monthly trending (window size is easily configurable). These sliding-time-window quantities do not exist (and are thus not plotted) until one full window into the available time range, however the horizontal axis is scaled to include the full time range of state records being used. This is done in order to reveal the "System Size" configuration (number of compute nodes attached to the system, see preceding Table) over the full time range. Data date range is shown at the bottom of the plot.

System size is a significant factor in RAS performance, and is thus plotted as useful background information. "Incident" (non-Production Time periods) durations are shown in the plot, with special emphasis given to Unscheduled Downtime incidents (the mean duration of the last ten is calculated and given as Mean Time to Repair (MTTR)).

How should one interpret availability when there are no compute nodes available? Since the service partition remains available (except during reboots), we treat this as 100% of the available compute nodes. Nodehour-scaled metrics (MNBIservice, MNBIjob, etc [1]) are expected to be more satisfying in this regard, but their implementation is not yet complete.

Plot Interpretation

2006 began with the Unclassified system in "Unclassified Jumbo mode", during which Engineering Time experiments were conducted. Engineering Time activities are conducted only on the Unclassified system, and these are decreasing over time. Scheduled Downtimes occur weekly on both systems, and take an average of five hours. More Unscheduled Downtimes have occurred on the Classified system (believed to be due to larger size and more stressful workload), but on both systems this totals less than two percent of the total time. When an Unscheduled Downtime does occur, it is returned for use within ninety minutes (regardless of configuration size). Overall, Red Storm is available to production users ninety-five percent of the time.

4 MTBIservice

MTBIService measures the stability of the operating platform as a whole, and is calculated using the same state transition records as used for the Availability plots. In a perfect world, MTBIservice would be measured by "time between reboots." Operationally, however, we measure the downtime from either the time of failure (if this can be determined) or otherwise the point at which corrective action is initiated, to the time the system was returned to normal use. At least one reboot is always required, but sometimes the system requires multiple reboots to correct an issue.

MTBIservice provides a useful production-user perspective of system reliability, as it describes how long the system remains available to them regardless of why (e.g. scheduled or unscheduled) interruptions in production service occurs.

Description

System MTBIservice plots are shown in Figure 3. The duration of each Production Time period is plotted as a circle, with its color indicating how that period was interrupted (e.g. indicates the state to which the system transitioned). The mean duration of the last ten Production Times duration using a sliding window (again, of configurable width) is plotted in order to reveal trend. Special emphasis is again given to Unscheduled Downtimes, with the weekly total number given along the bottom axis. System size is again provided for reference.

Interpretation

System reliability is inversely proportional to System Size (Unscheduled Downtimes occur more often when the system is large). We average one Unscheduled Downtime per week on small configuration (2,688) and two on large configuration (7,680). MTBIservice is increasing (at least on the Classified system), but has as not yet surpassed 100 hours as required by Red Storm's acceptance test plan [3]. Classified production users can expect uninterrupted service for at least sixty hours and unclassified users for at least forty-five hours.

5 MTBIjob

The primary function of Red Storm is to run large-scale capability jobs for production users. MTBIjob therefore is a significant measure of system reliability, as it describes how long production users can expect the system to be available for production jobs before any system-induced job interrupts occur. We again note that nodehour-scaled metrics will be more informational, but their implementation is still in progress.

The XT3 interconnect forms a 3-D mesh. Since the system is highly configurable, the size of the mesh, and whether directions in the mesh are toroidal, varies from system to system. Cray provides a static routing algorithm for the mesh that is created at boot time, and which depends on the availability of individual router chips that make up the nodes in the mesh.

Primary causes of XT3 job interrupts are listed below:

1. A compute or service node allocated to the job can fail (enter Unscheduled Downtime). In this case, the underlying management system detects the failure and kills the job.

2. An I/O service node required for the job to progress could fail. This happens when a job is doing I/O to a file in a Lustre file system, and the node failure kills the file system. In this case, both the file system and the jobs using it hang. In theory, we could find the jobs that are clients of a given file system and backtrack the failure to kill the job. In practice, such a mechanism has not been implemented.

3. A router in the mesh required for communication between the yod node and the job's compute nodes could have failed, and

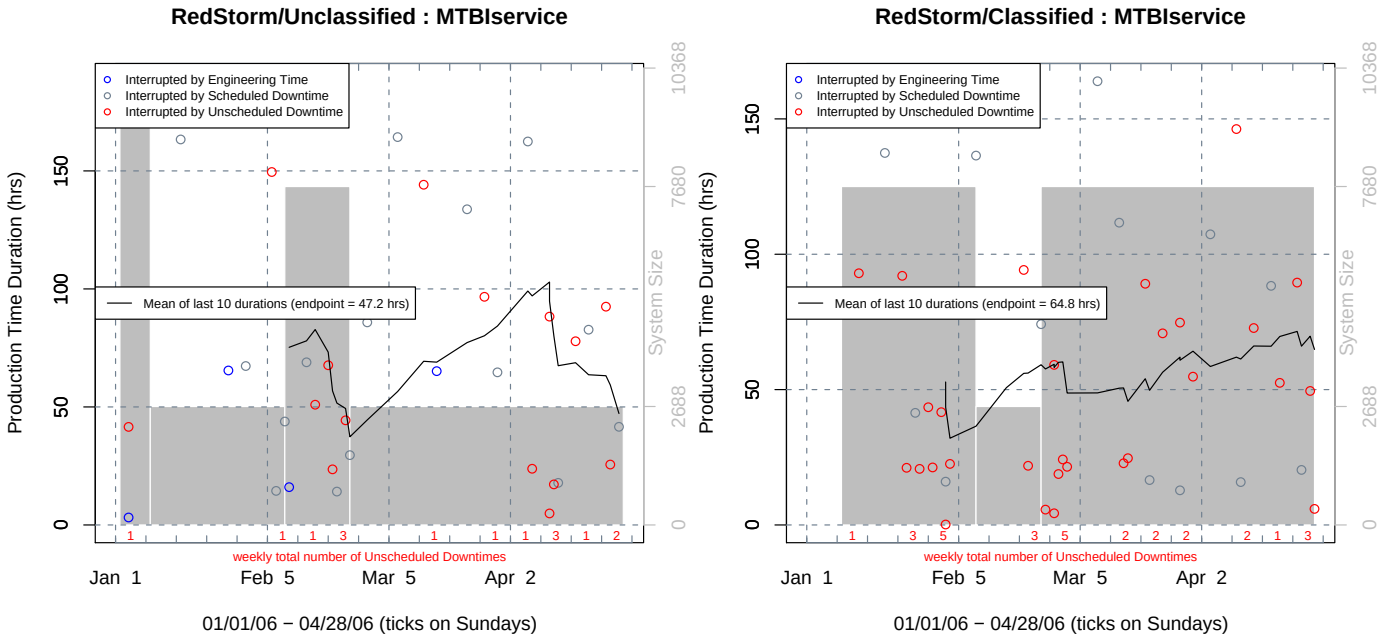

Figure 3: System MTBIservice Plots

so job launch would fail. Detection of this error has been added to the Cray software release as of 1.3.21. We are only now getting experience with this feature.

4. After job launch, a router in the mesh required for communication between compute nodes in the job or between the job's compute nodes, I/O nodes, or yod node could fail. In this case jobs silently hang.

5. During a reboot, all running jobs are killed.

For the MTBIjob plots below, we currently include failures from only 1 and 5.

Description

MTBIjob plots are shown in Figure 4. MTBIjob is calculated as the total number of Production Time hours divided by the total number of job interrupts, using a 28-day (configurable) sliding window. We currently only track two types of job interrupts - those caused by individual nodes entering Unscheduled Downtime (1 above), or the whole system entering Unscheduled Downtime (5 above). Data for both types of interrupts are gathered from Red Storm's XTAcct database using the SQL queries shown in Table 1 (the former via the "unavail" query, the latter via the "shutdown" query). It has been observed that the time at which a node became unexpectedly unavailable can be slightly later than the affected job's destroy_time - "INTERVAL 60 SECOND" is used in order to include such interrupts in the query results. The results from these queries are then culled using the system state records as follows: unavail interrupts occurring outside Production Time are ignored, shutdown interrupts not occurring within twenty minutes (plus or minus - a reporting margin of error) of the beginning of a system Unscheduled Downtime are ignored. Remaining interrupts are then totalled for the window and used as the denominator in the MTBIjob calculation. The weekly subtotal for each type of interrupt is shown along the horizontal axes. These are the only job interrupts we can currently identify with high confidence, so MTBIjob is considered best-case (if job hangs and other interrupt types were counted, MTBIjob would be decreased). In order to mitigate this, the maximum value of MTBIjob is clipped at 168 (we have weekly Scheduled Downtimes, 7*24=168).

Interpretation

The need to account for more types of job interrupts is emphasized by Unclassified plot, which peaks at the clipped value of 168. Unscheduled Downtimes interrupt a significant number of jobs. Node reliability is increasing (node failures are decreasing). Both plots show a clear trend of increasing MTBIjob, but have not surpassed the required level of 50 hours [3].

6 Lessons Learned

Every time you run a job, you place a wager:
Your job will complete (or fail) before your hardware fails.

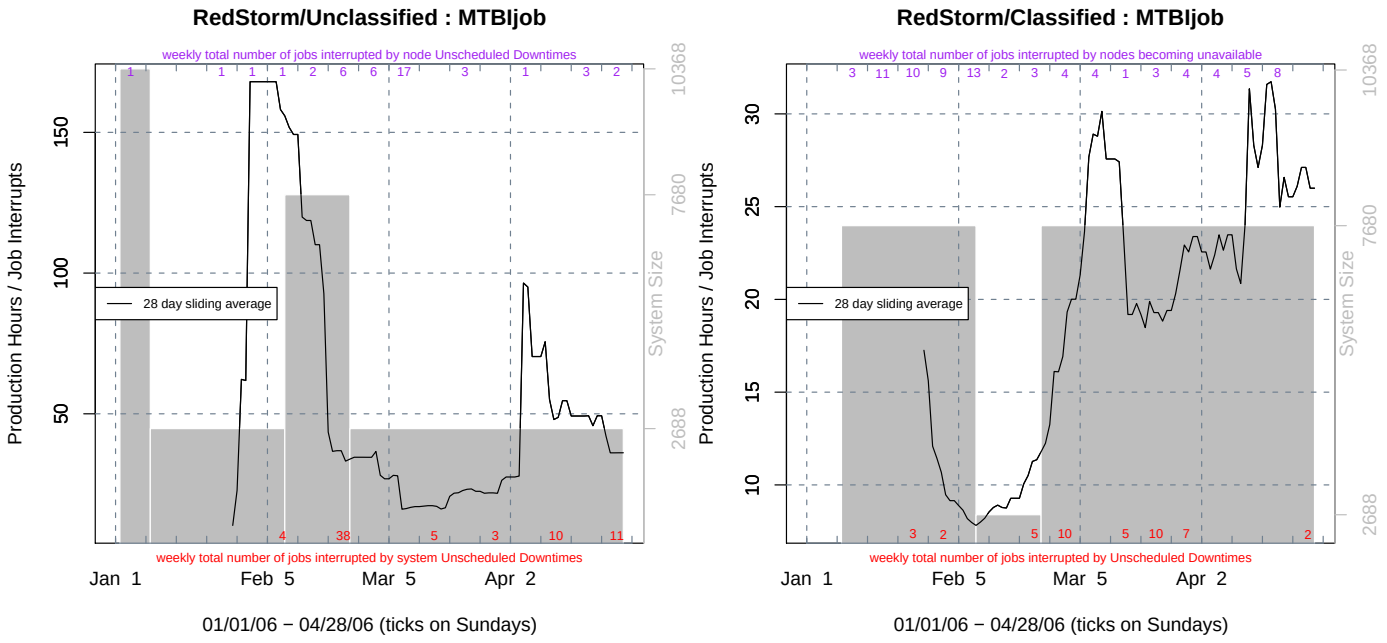

Figure 4: System MTBIjob Plots

Query Name	SQL Statement
unavail	SELECT MIN(etimestamp), yod_accounting.yod_id, 'unavail' FROM yod_accounting JOIN yod_accounting_processor_list USING (yod_id) JOIN node_event_log USING (processor_id) WHERE status='unavail' AND etimestamp > '2006-01-01' AND etimestamp BETWEEN create_time AND (destroy_time + INTERVAL 60 SECOND) GROUP BY yod_accounting.yod_id;
shutdown	SELECT destroy_time, yod_id, 'shutdown' FROM yod_accounting WHERE destroy_time > '2006-01-01' AND exit_info LIKE '%shutdown%';

Table 1: Job Interrupt SQL Queries

- It will be the sustained, usable operation rates, rather than the peak rates, that determines the success of Red Storm. With 10^4–10^5 hardware and software components in high-end XT3 systems, system administration, monitoring, and management become key issues. It is necessary to design the systems to be managed effectively while remaining thrifty with personnel resources. Thrift, in this area, is essential since qualified personnel are scarce and expensive.

  Experience with the current Cray RAS tools indicates that there is room for improvement in detection, in data handling, in configuration management, and in event correlation. Those charged with managing and operating XT3 systems should continue to voice their requirements in order to assure the continued growth in system capabilities.

- It is not sufficient for RAS data to be captured onto a single device. Effective understanding of the system operations requires managing and correlating data from multiple sources — such as determining availability from system size and incident information. Ideally the incident data would link to Cray's hardware and software incident databases. Alternatively, we envision hooks from the CRMS RAS system into our own incident database.

- The volume of data present in a large-scale XT3 requires graphical presentations in order to make the information understandable. It is hard to act on data until we can make correlations.

- It is difficult to distill the high dimensionality of such systems into concise and meaningful plots. The authors have attempted numerous data formats, influenced in part by the work of Edward Tufte [4]. Still, the charting techniques are not as effective as we'd like.

7 The RASM toolkit

The "RASM" R [5] toolkit implements the subset of the RAS Metrics [1, 2] presented herein. It includes documentation, as well as the script used to generate the plots in this report (taking simple tab-separated ASCII as input data), and is available at http://www.cs.sandia.gov/~jrstear/ras/. We are in the process of producing similar reports for other Sandia supercomputers in order to quantitatively understand and compare their RAS performance. Efforts are also underway by Sandia, Livermore, and Los Alamos National Laboratories to establish a tri-lab endorsed "Specification for defining and measuring high performance computing reliability, availability, and serviceability."

Feedback on RAS metrics and their implementation is hereby solicited.

Acknowledgements

Thanks to Sue Kelly, Jim Tomkins, John Noe, and Rob LeLand for their feedback regarding clear and meaningful presentation of RAS performance data.

Document Revision

This is $Revision: 1.4 $, $Date: 2006/04/28 18:49:29 $.

References

[1] Jon Stearley. Defining and measuring supercomputer Reliability, Availability, and Serviceability (RAS). *Proceedings of the 2005 Linux Clusters Institute Conference*, 2005. see http://www.cs.sandia.gov/~jrstear/ras.

[2] Jon Stearley. Towards a specification for measuring red storm Reliability, Availability, and Serviceability (RAS). *Proceedings of the 2005 Cray Users Group Conference*, 2005.

[3] ASC Red Storm acceptance test plan. Cray and Sandia National Laboratories internal document.

[4] Edward Tufte. *Envisioning information*. Graphics Press LLC, 1990.

[5] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.