Defining and Measuring Supercomputer Reliability, Availability, and Serviceability (RAS)

Jon Stearley <jrstear@sandia.gov>

Sandia National Laboratories[⋆⋆]
Albuquerque, New Mexico

Abstract. The absence of agreed definitions and metrics for supercomputer RAS obscures meaningful discussion of the issues involved and hinders their solution. This paper seeks to foster a common basis for communication about supercomputer RAS, by proposing a system state model, definitions, and measurements. These are modeled after the SEMI-E10 [1] specification which is widely used in the semiconductor manufacturing industry.

1 Impetus

The needs for standardized terminology and metrics for supercomputer RAS begins with the procurement process, as the below quotation excellently summarizes:

> "prevailing procurement practices are ... a lengthy and expensive undertaking both for the government and for participating vendors. Thus any technically valid methodologies that can standardize or streamline this process will result in greater value to the federally-funded centers, and greater opportunity to focus on the real problems involved in deploying and utilizing one of these large systems." [2]

Appendix A provides several examples of "numerous general hardware and software specifications" [2] from the procurements of several modern systems. While there are clearly common issues being communicated, the language used is far from consistent. Sites struggle to describe their reliability needs, and vendors strive to translate these descriptions into capabilities they can deliver to multiple customers. Another example is provided by this excerpt:

> "The system must be reliable... It is important to define what we mean by reliable. We do not mean high availability... Reliability in this context means that a large parallel job running for many hours has a high probability of successfully completing. It is measured by the mean time between job failures. Note that the system can undergo a failure that does not lead to loss of a job without affecting reliability - this is important to developing reliability enhancement strategies. A related requirement would be that if the system undergoes a failure that is local, only jobs using that local resource are affected. This

[⋆⋆] 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.

> kind of aspect of reliability we also call resiliency. Note that a system can have very high availability and not be reliable for our purposes. It is, by contrast, unlikely that a system that has low availability could have high reliability." [3]

Standardized terms and measurements for supercomputer RAS will streamline the procurement process.

Once a system is operational, even a simple phrase like "the system is up" can have very different meanings between who is speaking, who is hearing, and what system is being described. Categorizing the type and impact of undesired system events is similarly unclear - for example: is intermittent response from an I/O node an interrupt or a failure, how can its effect on users measured, etc? In both operational and design review, it is difficult to have meaningful discussions due to inability to agree on terminology. Making complex supercomputers reliable is difficult even with clear communication, but unclear communication further complicates the process and delays progress. Standardized terms and measurements will facilitate practical improvements in RAS performance.

Not all sites track RAS data for their supercomputer(s), and comparing data from those sites who do requires careful review of their definitions and calculations. For example, both NERSC and LLNL do an excellent job at tracking RAS data (NERSC data is public at `http://www.nersc.gov/nusers/status/AvailStats/`, LLNL provided extensive data to me upon request) - matching words and metric names are used, but it is unclear if the definitions and calculations also match exactly. Accurate RAS performance comparisons between these sites is possible via very careful review, but standardization would ease this process and benefit the high-performance computing (HPC) community as a whole.

All systems reach a point where it is more cost-effective to replace them than to continue to operate them. The reliability, availability, serviceability, utilization, cost effectiveness, (etc) of existing systems are compared to what can be procured - in most cases without clear terminology or quantitative metrics for either. And so the cycle continues.

It is not uncommon for users of supercomputers to express frustrations regarding system reliability - even when the cost of their systems ranges in the tens of millions of dollars. Accurate quantitative assessment of supercomputer RAS performance is arguably impossible without agreed-upon definitions and measurements - their lack is extremely expensive in time, effort, and money. In response to similar needs for standardization, the semiconductor manufacturing industry has developed the "Specification for Definition and Measurement of Equipment Reliability, Availability, and Maintainability" (SEMI-E10 [1]). The remainder of this document is largely modeled[1] after that Specification, and proposes a standardized system state model, definitions, and measurements for supercomputer RAS.

[1] Guidance was provided at Sandia's 2004 CSI External Reviews that relevant lessons and practices from the manufacturing industry be leveraged to improve supercomputer RAS.

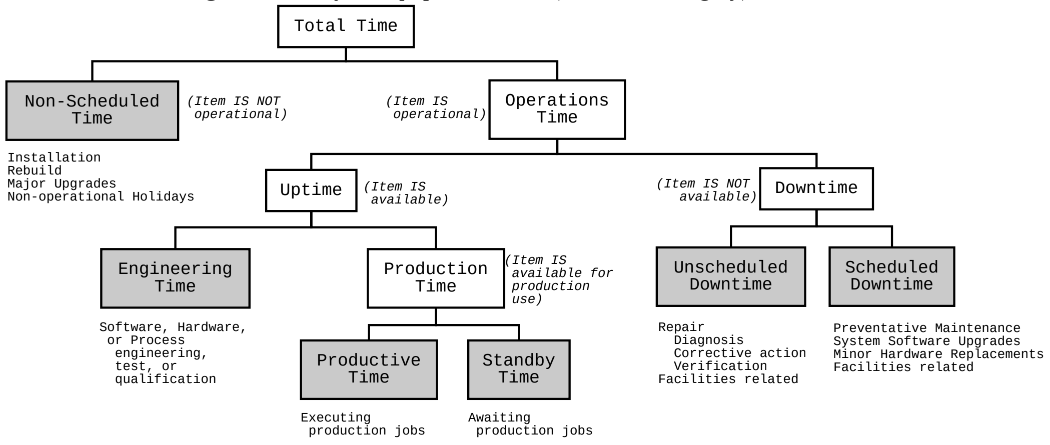
2 System State Model

It would take an incredibly dense state diagram to visualize all the possible states a supercomputer and its workload can reach. Navigating this diagram during system debugging is at least intimidating, and can feel humorously hopeless at worst:

“A computer is in one of two situations. It is either known to be bad or it is in an unknown state.” - Mike Levine (PSC)

Clear definitions of equipment states are a prerequisite to accurate RAS measurements; this document defines six basic states into which all equipment conditions and periods of time must fall (see Figure 1).

Fig. 1. Hierarchy of Equipment States (basic states in gray)



Boldface is used below for words defined in section 3.

2.1 PRODUCTIVE STATE

The time (productive time) when an **item** is performing computations on behalf of production users, for example:

– (the **system** is) executing **jobs** for one or more production users
– (the **node** is) executing a **job** for a production user

2.2 STANDBY STATE

The time (standby time) when an item is **available** to production users, but not in a productive state due to workload conditions, for example:

– no **jobs** in batch queue
– **jobs** in batch queue require more **nodes** than are currently in standby state, or are delayed due to queue priority configuration

2.3 ENGINEERING STATE

The time (engineering time) when an **item** is **available** to system engineering users, for example:

- system software engineering and qualification (e.g. operating system software, batch system software, etc)
- hardware engineering and evaluation (e.g. involving different hardware settings or configurations, new **components**, etc)
- process engineering (e.g. refining of support processes such as booting, shutdown, problem isolation, etc)

2.4 SCHEDULED DOWNTIME STATE

The time (scheduled downtime) when an **item** is not **available** due to planned events, for example:

- preventative maintenance
- hardware or software upgrades
- system verification (testing in order to verify that it is functioning properly)
- maintenance delay - time waiting for maintenance personnel or parts (maintenance delay may also be due to an administrative decision to postpone maintenance)
- facilities related (power, cooling, etc)

2.5 UNSCHEDULED DOWNTIME STATE

The time (unscheduled downtime) when an **item** is not **available** due to unplanned events, for example:

- repair (including time spent for diagnosis, corrective action, and verification of repair)
- maintenance delay
- facilities related (power, cooling, etc)

2.6 NON-SCHEDULED STATE

The time (non-scheduled time) when an **item** is not scheduled to be utilized by production or system engineering users, for example:

- initial installation
- rebuilds and upgrades which are beyond the scope of scheduled downtime
- holidays during which the item is not expected to be operational

2.7 State Hierarchy

Time spent in the six basic equipment states is hierarchically organized as follows (see Figure 1):

TOTAL TIME all time (at the rate of 24hrs/day, 7 days/week) during the period being measured. In order to have a valid representation of total time, all six basic equipment states must be accurately accounted for.

OPERATIONS TIME total time minus non-scheduled time.

UPTIME time when an **item** is **available**; the sum of engineering and production time.

DOWNTIME time when an **item** is not **available**; the sum of scheduled and unscheduled downtime.

PRODUCTION TIME [2] the time when an **item** is **available** to production users; the sum of productive and standby time.

3 Definitions

This section proposes standardized definitions of terms which are commonly used, but not commonly agreed upon. Great effort has been made to utilize established definitions wherever possible. Only those terms deemed necessary are given (consult referenced dictionaries for more information).

3.1 RAS Terminology

Reliability the probability that an item[3] will function without failure under stated conditions for a specified amount of time [4]. "Stated conditions" indicates prerequisite conditions external to the item being considered. For example, a stated condition for a supercomputer might be that power and cooling must be available - thus a failure of the power or cooling systems would not be considered a failure of the supercomputer itself.

Availability the fraction of a time period that an item is in a condition to perform its intended function upon demand [1] ("**available**" indicates that an item is in this condition); availability is often expressed as a probability [4].

Serviceability [4] the probability that an item will be retained in, or restored to, a condition to perform its intended function within a specified period of time [1].

Maintenance the act of sustaining an item in or restoring it to a condition to perform its intended function [1], usually during scheduled time.

[2] "Production time" herein is analogous to "manufacturing time" in SEMI-E10.

[3] The use of the term "item" intentionally allows for the calculation of reliability for individual components or for the system as a whole. Similarly for other uses of the term "item" in this document.

[4] Serviceability (widely used in the supercomputer HPC community) is herein defined as an exact synonym for the decades-old term "maintainability" (widely used in engineering and manufacturing [4,1]). Perhaps "maintainability" is not used in the HPC community in order to avoid an acronym conflict with Random Access Memory (RAM)?

Repair the act of restoring an item to a condition to perform its intended function.

Utilization the percentage of a time period that an item actually performs its intended function [1].

System Downtime Event a detectable occurrence significant to the system which causes it to transition from an uptime state to a downtime state (states are defined in section 2), regardless of why the transition is made (e.g. scheduled downtime, system failure, administrative decision, etc) [1].

3.2 Foundational Terminology

Supercomputer any of the group of computers that have the fastest processing speeds available at a given time [4]. Generally speaking, the intended function of a supercomputer is to quickly perform computations for users.

System a collection of components organized to accomplish a specific function or set of functions [4]. When dealing with a specific supercomputer, "the system" means "the (majority of components of the) supercomputer" - for example, a site may not consider "the system" to be in a production status until at least 95% of it's *nodes* (defined below) are in a production status.

Component one of the parts that make up a system. A component may be hardware or software and may be subdivided into other components. [4]

Item an all-inclusive term to denote any level of unit, including system and component [4].

Process a set of interrelated or interacting activities which transforms inputs into outputs [5].

Event any occurrence which affects the state of an item [4].

Interrupt the suspension of a process to handle an event external to the process [4]. See section 3.3 for specific types of supercomputer interrupts.

Failure the termination of the ability of an item to perform a required function [4]. External corrective action is required in order to restore the *ability* of an item to perform a required function, e.g. manual reboot, repair, or replacement.

– *Failures regard items, interrupts regard work (being performed by the items).*
– *It is important to categorize interrupts and failures in ways that facilitate the resolution of problems and improve overall system performance. Effective application of this specification requires agreement on such categorization.*

Fault an accidental condition that causes an item to fail to perform its required function [4].

3.3 Supercomputer Terminology

Job a user-defined unit of work that is to be accomplished by a computer [4]. For a job processed by a batch system, the following distinct stages are defined:

1. **submission** - a request is made for fast computation.
2. **wait** - delay may occur until sufficient resources are available to fulfill the request, including consideration of queuing priorities.

3. **shell-execution** - resources are allocated to fulfill request, a shell is invoked, and scripted commands may take place such as data preprocessing.
4. **application-execution** - computations are performed
5. **cleanup** - resources are made available for other requests, optionally: scripted commands such as post-processing, delivery of results, and notification of job completion.

Jobs not processed by a batch system generally only consist of an application-execution stage. An “active job” is a job which is in shell-execution, application-execution, or cleanup stages.

Job Kill the expected interruption of an active job.
Job Interrupt (I_{Job}) the unexpected interruption of an active job.
System Interrupt (I_{System}) the unexpected interruption of all active jobs.
System Failure (F_{System}) an unscheduled event requiring that *the system* enter a downtime status before *any component* may transition into a productive status (e.g. the system must be repaired before new jobs can execute).
Service Interrupt ($I_{Service}$) any event that disrupts full service to users, including system transitions out of production or engineering status, or any drop below a promised number of compute nodes [6]. For example, from the perspective of production users, any time the system is not capable of running a job (sized at the intended function of the system) is a service interrupt.
Component Failure the failure of a component for any reason other than design flaws [4], which may or may not result in a job or system interrupt, or system failure.
Node a hardware component consisting of one or more[5] CPUs and capable of communicating with other nodes in order to perform parallel computations.
Nodehour [6] a unit of work equal to one node computing for one hour.
Wallclock Time regular time as displayed on a wallclock.
Production Nodehours the sum of all time on all nodes in production state (see Section 2.7), e.g. commonly estimated as production time (in hours) times the total number of nodes in the system.
Productive Nodehours the sum of all time on all nodes in a productive state (see Section 2.1), e.g. job duration times job size (the number of nodes the job utilized).
Field Replaceable Unit (FRU) [7] a hardware component that can be easily replaced in the field [4].

4 Measurements

“In the history of science and technology, it is clear that progress can be strongly correlated with the availability of quantitative data. ... The substitution of arm waving and hype has been a major contributor to the tragedies in the field...” [7]

[5] “Nodehours” is a commonly used term and is thus used herein. However, the use of “processor-hours” may be justified in systems containing more than one CPU.
[6] “nodehour” is used instead of “node-hour” in order to avoid ambiguity in equations.
[7] FRU herein is analogous to consumables in SEMI-E10.

Quantitative understanding of performance is a prerequisite for continual performance improvements ([5] sections 0.2{f,g}). Motivations to collect metrics can vary widely; the objectives of this document are:

1. to work towards the identification of those metrics which are truly useful in improving RAS performance, and
2. to facilitate effective communication about RAS performance (enabling clear requirements, accurate systems comparisons, etc).

Because scale (number of nodes) is a principal feature of supercomputers, it is useful to define metrics based on wallclock time (denoted herein by "T") and nodehour time (denoted herein by "N"). All references to "time" or "nodehours" in below equations are cumulative over the period of calculation, e.g. "production time" below means the sum of all wallclock hours spent in a production state during the period of time being considered.

Rigorous tracking of RAS events (e.g. node status transitions, job interrupts, etc) is a prerequisite for quantitative understanding of RAS performance. Tracking methods are beyond the scope of this document.

4.1 Reliability

Reliability is often [8] calculated as $R(t) = e^{-\lambda t}$ where $\lambda = \frac{1}{MTBF}$ is the *constant* failure rate (uses an exponential random variable model). I am not confident that supercomputer failure rates are constant, and therefore do not use this model for calculating reliability. Similarly for rates of repair in the calculation of serviceability. This document proposes standardization of low-level metrics only - selection of appropriate models (e.g. Poisson random variable?) for high-level metrics is left for future work.

Careful classification of events (e.g. interrupt versus failure) and their scope of impact (e.g. job versus system) enables clear communication about system reliability. Readers are encouraged to review these distinctions and consider their practical relevance.

Only uptime is included in reliability calculations (downtime is included in availability calculations). Furthermore, the below metrics focus on production time - similar metrics focused on engineering time may be appropriate for some systems (e.g. see System Production and Engineering Availability in Section 4.2).

Mean Time and Nodehours Between Job Interrupts It is very common for users to form estimates (and expectations) of how often they experience interrupts. It is also common however for these reports to vary widely. This metric conveys the time between such undesirable events.

$$MTBI_{Job} = \frac{production\,time}{number\,of\,job\,interrupts}$$

Inconsistent reports can result if this metric is (incorrectly) estimated using uptime pertaining to only a subset of jobs in the numerator rather than system-wide uptime. For example, a user who runs ten one-hour jobs of which five interrupt, may erroneously report that $MTBI_J = 2\, hours/interrupt$ (=10 hours / 5 job interrupts). The truth may however be that these were the only interrupts experienced on the system over five full days of service, thus yielding $MTBI_J = 24\, hours/interrupt$ (=5*24/5). Comparison of system-wide $MTBI_{Job}$ as above to a subset $MTBI_{Job}$ (e.g. per-user or per-application) may be useful towards identifying factors correlated with interrupts.

A key weakness of $MTBI_{Job}$ however, is that it does *not* convey any information about the *amount of work* which can be accomplished. A metric based on computational work (nodehours) is more informational:

$$MNBI_{Job} = \frac{productive\, nodehours}{number\, of\, job\, interrupts}$$

$MNBI_{Job}$ provides insight into how much computational work can be expected to complete without interrupt, and may be useful to users in estimating the job size and duration most likely to complete. A plot of $MTBI_{Job}$ as a function of job size would be useful - and an informational accompaniment to application scaling efficiency plots.

Contour plots of the probability of jobs completing without interrupt (with job size and duration as horizontal and vertical axes respectively) may also be useful (e.g. aggregating all jobs on the system over a period of time).

Mean Time and Nodehours Between Node Failures Node failures (e.g. including events requiring reboot, repair, or replacement of nodes) are a common cause of job interrupts - these metrics convey the average time and productive work between these events.

$$MTBF_{Node} = \frac{production\, time}{number\, of\, node\, failures}$$

$$MNBF_{Node} = \frac{productive\, nodehours}{number\, of\, node\, failures}$$

Mean Time and Nodehours Between System Failures System failures (e.g. including necessary unscheduled system reboots) are a primary undesirable event to nearly everyone (and are consistently evident in Appendix A). These metrics convey the average amount of time and productive work between such events.

$$MTBF_{System} = \frac{production\, time}{number\, of\, system\, failures}$$

$$MNBF_{System} = \frac{productive\, nodehours}{number\, of\, system\, failures}$$

Mean Time and Nodehours Between Service Interrupts Service interrupts are of principal concern to users - these metrics convey the average time and productive work between such events. They are aggregate metrics, affected by both scheduled and unscheduled service interrupts.

$$MTBI_{Service} = \frac{production\, time}{number\, of\, service\, interrupts}$$

$$MNBI_{Service} = \frac{productive\, nodehours}{number\, of\, service\, interrupts}$$

4.2 Availability

Total System Availability (%) This calculation measures the percentage of a time period that the system was available. The key feature of this metric is the use of total time (all states) in the denominator - for many users what matters is *that* the system be available, not *why* it was not.

$$Total\,Availability_{System}(\%) = \frac{uptime}{total\,time} * 100$$

Scheduled System Availability (%) This calculation measures how fully uptime expectations are met during a time period. The key feature of this metric is that quantitative expectations exist (e.g. uptime and downtime schedules are set at the beginning of the time period).

$$Scheduled\,Availability_{System}(\%) = \frac{uptime-downtime}{scheduled\,uptime} * 100$$

System Production and Engineering Availability (%) For systems having both significant engineering and production purposes, separate measurements of time spent fulfilling each function may be useful (systems without such dual-purposes are sufficiently served by Total System Availability above).

$$Production\,Availability_{System}(\%) = \frac{production\,time}{operations\,time} * 100$$

$$Engineering\,Availability_{System}(\%) = \frac{engineering\,time}{operations\,time} * 100$$

4.3 Serviceability

Calculation of these metrics on an overall system basis as well as per failure type basis is useful towards quantitative understanding of the practical impact of each failure type. Again, this requires the establishment of failure categories and accurate recording of events.

Mean Time To Repair This calculation is intended to reflect the average amount of time it takes to recover from a failure.

$$MTTR = \frac{unscheduled\,downtime}{number\,of\,failures}$$

Mean Nodehours To Repair This calculation measures the average computational ability lost per failure. Example usage of this metric would be to measure the scope of impact of failure events which cause portions of compute nodes to become unavailable, rather than the entire system.

$$MNTR = \frac{unscheduled\,downtime\,nodehours}{number\,of\,failures}$$

Mean time to Boot System Wallclock time to boot the complete system is a useful metric [9], whose importance increases with the number of times the system must be booted (e.g. the number of system failure events requiring a system reboot).

$$MTTB_{System} = \frac{sum\,of\,wallclock\,time\,booting\,the\,system}{number\,of\,boot\,events}$$

4.4 Utilization

Total System Utilization This calculation is intended to reflect overall production utilization of the system. Because it uses total time in the denominator, it is a meaningful aggregate of reliability, availability, and serviceability.

$$Total\,Utilization_{System}(\%) = \frac{productive\,time}{total\,time} * 100$$

Production Time System Utilization (%) This calculation measures the percentage of a system's available computational ability that was actually utilized. This is *not a RAS metric* - it is entirely a function of workload and queuing configuration - but is included here for completeness.

$$Production\,Time\,Utilization_{System}(\%) = \frac{productive\,nodehours}{production\,nodehours} * 100$$

5 Implementation

This document is intentionally platform-independent - it seeks to foster effective communication about supercomputer RAS. There are however multiple characteristics of Linux which are well aligned with this objective, and thus suggest it as a good candidate as an implementation platform:

– Linux is increasingly present in supercomputers (increasingly becoming a standard).
– Linux culture has strong aspects of cost-effectiveness and open, standardized implementations.
– Multiple packages are available which collect and present detailed system statistics from large sets of Linux nodes (e.g. Ganglia, Supermon).

Beyond the adoption of agreed-upon terminology, the following are needed towards practical implementation of this document:

1. The intended function(s) of the system and their time proportions must be clearly enumerated. For example, what exactly is the intended balance of the system being considered for production use, system-development use, etc?
2. Interrupt and failure modes must be clearly categorized, including their scope of impact. Key to this effort is keeping in mind "from who's perspective did this fail/etc?" Categorization hierarchies should be enumerated so that new (or rare) events can be accurately accounted for. Sharing of such categorization hierarchies and policies will benefit the HPC community.
3. Low-level statistics must be meaningfully aggregated into high-level metrics appropriate for inter-system and inter-site comparison.

6 Conclusions

It is easy for supercomputer users and administrators to have deep understanding of each other's frustrations regarding reliability, availability, and serviceability (RAS), but effective collaboration towards improvement is hindered by the lack of standardized terminology and measurements. This lack also increases the cost of supercomputers in all phases (design, procurement, operation, and retirement). Supercomputers today are complex, expensive, and relied upon - each in increasing measure. Significant improvements in system RAS are a prerequisite for sustained computation by future even larger and complex supercomputers.

RAS concepts are well understood in other industries and the HPC community would be wise to leverage these investments to improve supercomputer RAS. This document is largely modeled after the SEMI-E10 semiconductor manufacturing SEMI-E10 specification, and proposes a standardized system state model, definitions, and measurements for supercomputer RAS.

Acknowledgments

Sue Kelly, Bob Ballance, Doug Doerfler, Nathan Dauchy, Ron Brightwell, Neil Pundit, Mike Davis, Tim Ingebritson, Jim Ang, and William Kramer have provided reviews and/or contributions to this document - thank you! Thanks also to Mark Seager and Dave Fox for providing LLNL RAS data!

Postscript

Successful standards must be developed and established by a consensus-based process. Feedback on this document and contribution toward this effort are hereby solicited from any interested party.

Revision

Revision 1.35 of this document is published in the proceedings of the 6th LCI International Conference on Linux Clusters (April 2005). This is $Revision: 1.44 $, $Date: 2005/04/21 21:07:41 $. Updated revisions are available at `http://www.cs.sandia.gov/~jrstear/ras` or by contacting the author.

A Procurement Specification Excerpts

A.1 Sandia National Laboratories

"An Investigation into Reliability, Availability, and Serviceability (RAS) Features for Massively Parallel Processor Systems" by Kelly and Ogden [10] provides additional RAS details on Sandia Systems.

Red Storm [11]

- "There shall not be any single-point of failure that can cause a system interrupt for high failure rate components such as power supplies, processors, compute nodes, 3-D mesh primary communications network, or disks. It is acceptable for the application executing on a failed processor or node to fail but when this happens applications executing on other parts of the system shall not fail."
- (regarding nodes responsible for booting the system) "There shall be an automatic fail over mechanism to prevent a system interrupt due to the loss of a boot node."
- "Mean time between Interrupt (MTBI) for full system shall be greater than 50 hours for continuous operation of the full system on a single application code. This means that the full system must be able to run continuously on an application using the full system for 50 hours without any hardware component failures or system software failures that cause an interrupt or failure of the application code."
- "MTBI for the full system, as determined by the need to reboot the system, shall be greater than 100 hours of continuous operation. This means that the system will be continuously operational for 100 hours with at least 99% of the system resources available and all disk storage accessible."
- "FRU (Field Replaceable Units) failures shall be able to be determined, isolated, and routed around without system shutdown."

ICC *(Linux cluster)* [12]

- (TAC3) "Each cluster shall be up and processing applications a minimum of 90% of the (test) wall clock time."
- (TAC4) "Each cluster will be shutdown at least twice and rebooted during this evaluation period. One test will be a complete power down condition. Each cluster must be production ready within one hour following return of power. Reboot of the cluster from shutdown without power loss shall be less than 30 minutes. Production ready clusters must have at least 95% of nodes available to run applications within these time limits."
- (HAM7) "Management of each cluster must have less than 1 percent impact on the performance or reliability of the cluster."
- (HAM15) "The clusters must be designed to prevent a single point of failure. It is acceptable for an application using a failed component to fail, but this failure should not effect applications executing on other parts of the cluster that have not failed."
- "the key criteria for measuring reliability is Mean Time Between Interrupts (MTBI) of an application. System availability, or percentage of the time the system is "up", is of secondary importance. In fact, it is possible to have a machine with high availability that is not useful for Sandia's problems because its MTBI is too short."
- (SPM1) "The Mean Time Between Interrupt (MTBI) for a single application running on one-half of the entire system shall be greater than 48 hours of continuous operation."
- (SPM2) "The MTBI for the entire system, as defined by the need to reboot the system, shall be greater than 336 hours of continuous operation."

- (POM1) “Each cluster shall provide a simple accounting and utilization tracking facility capable of supporting a subscription process.”
- (CMD11) “Each cluster should support comprehensive monitoring of the state of its components and provide real time notification of equipment malfunction (e.g., loss of nodes, file system down, etc.).”

A.2 Lawrence Livermore National Laboratories

ASC Purple [13]

- “User app spanning 80% of the SMPs will complete a run with correct results that utilizes 200hrs of system+user CPU time in at most 240 wall clock hours without human intervention. User app spanning 30% will complete 200hrs in 220 wall clock hours w/o human intervention.”
- “System hw and sw will execute 100 hour capability jobs (jobs exercising at least 90% of the computational capability of the system) to successful completion 95% of the time. If application termination due to system errors can be masked by automatic system initiated parallel checkpoint/restart, then such failures will not count against successful application completion. That is, if the system can automatically take periodic application checkpoints and upon failure due to system errors automatically restart without human intervention, then these interruptions to application progress do not constitute failure of an application to successfully complete.”
- “Over any 4 week period, the system will have an effectiveness level of at least 95%. Effectiveness level is computed as the average of period effectiveness levels. ... Period effectiveness level is computed as University operational use time multiplied by max [0,(p-2d)/p] divided by the period wall clock time. Where p is the number of CPUs in the system and d is the number disabled.”
- “SMP or node or fru failures will be determined by supplied diagnostic utils, isolated, and routed around w/o system shutdown.”
- “Failure of a single component such as cpu, single smp, or single comm. channel will not cause the full system to become unavailable.”
- ... “the SMPs will be able to tolerate failures through graceful degradation of performance where the degradation is proportional to the number of FRUs actually failing.”

Thunder *(Linux Cluster) [14]*

- (TR-1) “nodes will be mechanically designed so that complete node disassembly and reassembly can be accomplished in less than 20 minutes by a trained technician.”
- (MTBF) “The Offeror will provide the MTBF calculation for each FRU and node type. The Offeror will use these statistics to calculate the MTBF for the provided aggregate Thunder cluster hardware. This calculation will be performed using a recognized standard. Examples of such standards are Military Standard (Mil Std) 756, Reliability Modeling and Prediction, which can be found in Military Handbook 217F, and the Sum of Parts Method outlined in Bellcore Technical Reference

Manual 332. In the absence of relevant technical information in the proposal, the University will be forced to make pessimistic reliability, availability, and serviceability assumptions in evaluating the proposal."

A.3 Los Alamos National Laboratories

Q [10]

- "Hot swap of FRU"
- "Node failures shall be determined, isolated, and routed around w/o system shutdown. Reconfig system around failed node for continued operation from a network workstation."
- "Failure of a single component such as single node, disk, or comm. channel shall not cause the full system to become unavailable."
- "Soft memory component failure in user memory shall not cause the node to fail."

References

1. Semiconductor Equipment and Materials International. Specification for definition and measurement of equipment reliability, availability, and maintainability. SEMI E10-0304, 1986, 2004.
2. Steven Ashby (LLNL), David H. Bailey (LBNL), Maurice Blackmon (UCAR), Patrick Bohrer (IBM), Kirk Cameron (U. SC), Carleton DeTar (U. Utah), Jack Dongarra (U. Tenn.), Douglas Dwoyer (NASA Langley), Peter Freeman (NSF), Ahmed Gheith (IBM), Brent Gorda (LBNL), Guy Hammer (DOD-MDA), Wesley Felter (IBM), Jeremy Kepner (MIT/LL), David Koester (MITRE), Sally McKee (Cornell), David Nelson (DOE), Jeffrey Nichols (ORNL), Michael Vahle (Sandia), Jeffrey Vetter (LLNL), Theresa Windus (PNNL), and Patrick Worley (ORNL). Performance modeling, metrics and specifications: Report of HECRTF working group six. USC CSCE TR-2003-016, August, 2003.
3. Ron Brightwell, William Camp, Benjamin Cole, Erik DeBenedictis, Robert Leland, Jim Tomkins, and Arthur B. Maccabe. Architectural specification for massively parallel computers - an experience and measurement-based approach. *Concurrency and Computation: Practice and Experience*, (Special Issue) The High Performance Architectural Challenge: Mass Market Versus Proprietary Components, September 2004.
4. Standards Coordinating Committee 10 (Terms and Definitions) Jane Radatz (Chair). *The IEEE Standard Dictionary of Electrical and Electronics Terms*, volume IEEE Std 100-1996. IEEE Publishing, 1996.
5. The International Organization for Standardization (ISO). Quality management systems - fundamentals and vocabulary. ISO-9000, 2000.
6. W. T. Kramer. How are we doing? A self-assessment of the quality of services and systems at NERSC, 2001. LBNL-47712.
7. David J. Kuck. High performance computing challenges for future systems. from High-End Computing Revitalization Task Force (HECRTF) presentation by Alan Laub and John Grosh on November 21, 2003.
8. Enrique Vargas Sun Microsystems Enterprise Engineering. High availability fundamentals. Revision 01, November 2000. http://www.sun.com/blueprints.
9. Adrian Wong, Leonid Oliker, William Dramer, Teresa Kaltz, and David Bailey. ESP: A system utilization benchmark. In *Proceedings of the Supercomputing 2000 Conference*, 2000.

10. Suzanne M. Kelly and Jeffry B. Ogden. An investigation into Reliability, Availability, and Serviceability (RAS) features for massively parallel processor systems. SAND2002-3164, 2002.
11. ASC red storm acceptance test plan. Cray and Sandia National Laboratories internal document.
12. Institutional computing cluster (ICC) - statement of work. RFQ 5031, Sandia National Laboratory.
13. Purple - statement of work (B519700). UCRL-PROP-145639DR, University of California Lawrence Livermore National Laboratory.
14. Thunder - statement of work (B532746). UCRL-MI-200098, University of California Lawrence Livermore National Laboratory.