

# 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

---

\*\* 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. This paper is published in the proceedings of the 6th LCI International Conference on Linux Clusters (2005).

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 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 be 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 and widely adopted the “Specification for Definition and Measurement of Equipment Reliability, Availability, and Maintainability” (SEMI-E10 [1]). The remainder of this document is largely modelled<sup>1</sup> after that Specification, and proposes a standardized system state model, definitions, and measurements for supercomputer RAS.

---

<sup>1</sup> 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

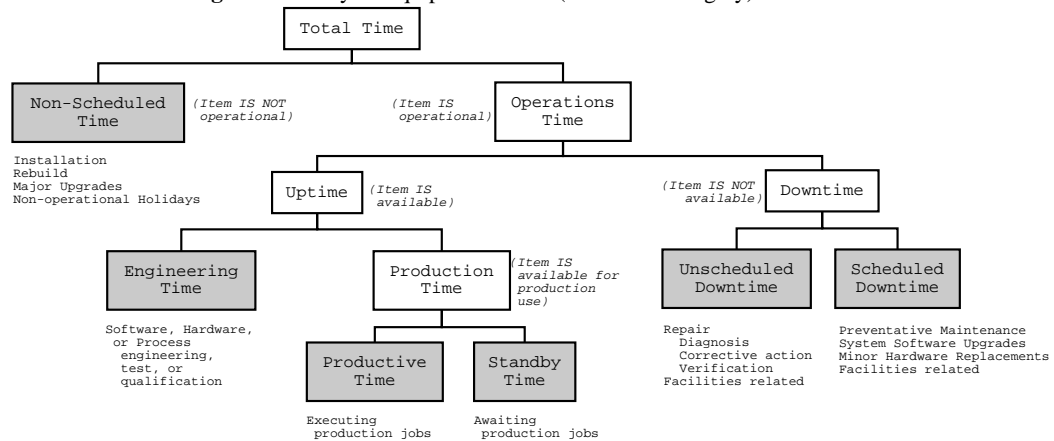
It would take an incredibly dense state diagram to visualize all the possible states a supercomputer and its workload can reach. Navigating this diagram during system debugging is at least intimidating, and can feel humorously hopeless at worst:

“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).

**Boldface** is used for words defined in section 3.

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



### 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

### 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 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**<sup>2</sup> 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 understood. 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<sup>3</sup> 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.

**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**<sup>4</sup> 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].

<sup>2</sup> “Production time” herein is analogous to “manufacturing time” in SEMI-E10.

<sup>3</sup> 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.

<sup>4</sup> 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)?

**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 perform computations for users (e.g. production users, system engineering users, etc).

**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 a *system* to be in a production status until at least 95% of its *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].

**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 the components of a system, interrupts regard the work being performed by a system.*
- *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 a 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. Jobs in shell-execution, application-execution, or cleanup stages are called “active.”

**Job Interrupt ( $I_j$ )** the unexpected interruption of an active job.

**Job Kill** the expected interruption of an active job.

**System Interrupt ( $I_s$ )** the unexpected interruption of all active jobs.

**System Failure ( $F_s$ )** an unscheduled system downtime event requiring that all application-execution cease before any new application-execution may begin, may or may not be coupled with an immediate interrupt. Using the states defined in section 2, system failure is an event which requiring that *all* components enter a downtime status before *any* component may enter a productive status.

**Component Failure** Failure of a component that is not precipitated through design flaws, due to faulty components or unanticipated conditions [4], and may or may not result in an interrupt or system failure.

**Node** a hardware component consisting of one or more<sup>5</sup> CPUs and capable of communicating with other nodes in order to perform parallel computations.

**Nodehour**<sup>6</sup> 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. the duration of a job (in hours) times the number of nodes the job utilized (e.g. “job size”).

**Field Replaceable Unit (FRU)**<sup>7</sup> 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...” [6]

<sup>5</sup> “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.

<sup>6</sup> “nodehour” is used instead of “node-hour” in order to avoid ambiguity in equations.

<sup>7</sup> 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”).

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 proposal.

#### 4.1 Reliability

Reliability is often[7] calculated using an exponential random variable model as  $R(t) = e^{-\lambda t}$  where  $\lambda = \frac{1}{MTBF}$  is the *constant* failure rate. I am not confident that supercomputer component failure rates are constant, and therefore do not use this model for 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?) for high-level metrics is left for future work.

Careful classification of the 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).

**Mean Time Between Job Interrupt** It is very common for users to form expectations of how often they experience interrupts.

$$MTBI_J = \frac{uptime}{number\ of\ job\ interrupts}$$

Inconsistent reports of reliability can result if this metric is (incorrectly) estimated using time 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  $MTBI_J = 2$  (=10 hours / 5 job interrupts), when in fact these were the only interrupts experienced on the system over five full days of service thus yeilding  $MTBI_J = 24$  (=5\*24/5). Comparison of system-wide  $MTBI_J$  as above to subset (e.g. per-user or per-application)  $MTBI_J$  may be useful towards identifying factors correlated with interrupts.  $MTBI_J$  does *not* convey any information about the *amount of work* which can be accomplished by the system between the interruptions of single jobs (in contrast, see  $MNBI_J$ ).

**Mean Nodehours Between Job Interrupt** This calculation is intended to provide 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 without interruption<sup>8</sup>. “Productive nodehours” is the sum of nodehours spent in productive state.

$$MNBI_J = \frac{\text{productive nodehours}}{\text{number of job interrupts}}$$

**Mean Time Between System Interrupt** This metric describes the average amount of time the system is available before all work on the system is interrupted.

$$MTBI_s = \frac{\text{availability time}}{\text{number of system interrupts}}$$

$MNBI_J$  conveys no information regarding the amount of work accomplished on a system between system interrupts.

**Mean Nodehours Between System Interrupt** This calculation is intended to convey the average amount of work which can be accomplished between system interrupts.

$$MNBI_s = \frac{\text{productive nodehours}}{\text{number of system interrupts}}$$

#### Mean Time Between System Failures

$$MTBF_s = \frac{\text{availability time}}{\text{number of system failures}}$$

#### Mean Nodehours Between System Failures

$$MNBFS = \frac{\text{productive nodehours}}{\text{number of system failures}}$$

## 4.2 Availability

**System Availability (%)** This calculation measures the percentage of a time period that the system was available to engineering or production users, and is often used as an expectation of future availability.

$$\text{Availability}_S(\%) = \frac{\text{uptime} * 100}{\text{operation time}}$$

**System Production Availability (%)** This calculation measures the percentage of operational time that the system available to production users, and is often used as an expectation about future production availability.

$$\text{Production Availability}_S(\%) = \frac{\text{production time} * 100}{\text{operation time}}$$

<sup>8</sup> Further, envision a plot relating job size, duration, and the probability of completing without interrupt, for example a contour plot with job size and duration as horizontal and vertical axes. How to best compute and visualize this has not yet been determined.

**System Production Utilization** This calculation measures what percentage of a system's available computational ability was actually utilized by production users.

$$ProductionUtilization_S(\%) = \frac{productive\ nodehours * 100}{production\ nodehours}$$

Utilization is an *indirect* measure of RAS - it high utilization implies high reliability and/or serviceability, but utilization alone does not convey information regarding interrupts or failures.

### 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, and thus how to best set engineering priorities. Again, this requires accurate recording and agreement of failure categories (see Section 3.2).

**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

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

**Mean time to Boot System** Wallclock time to boot the complete system is a useful metric [8], and generally scales in importance with the scale of the system.

$$MTTB_S = \frac{sum\ of\ wallclock\ time\ to\ boot\ system}{number\ of\ boot\ cycles}$$

As with any mean, the quality of this metric increases with the number of samples used (e.g. use more than one boot cycle).

## 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 (standardizable) 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 implementation of these concepts:

1. Intended functions and their time proportions must be clearly enumerated - for each system. For example, what exactly is the intended balance of this system 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. This must be implemented in such a way that sites can use default categorizations (e.g. HPC community standardized), or customize (e.g. evolution of best practices). Sharing of such categorization hierarchies and policies will benefit the HPC community.
3. Low-level statistics must be mathematically aggregated into high-level metrics appropriate for inter-system and inter-site comparison. Although the math is not complex, the use of flexible numerical software (R, Matlab, etc) would foster exploration and verification of models (e.g. exponential model in section 4.1).

## 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 difficult because terminology and measurements vary so widely. The current lack of standardization increases the cost of supercomputers in all phases (design, procurement, operation, and retirement) and delays RAS performance improvements. Today's supercomputers are important, complex, and expensive - and these characteristics are increasing with each new system generation. Significant improvements in system RAS are prerequisite for sustained even-faster computing.

RAS concepts are however well studied and largely overcome in other industries, and it is possible for the HPC community to leverage these investments. This document is largely modelled after the SEMI-E10 semiconductor manufacturing SEMI-E10 specification, and proposes a standardized system state model, definitions, and measurements for supercomputer RAS.

### Acknowledgements

Sue Kelly, Bob Ballance, Doug Doerfler, Nathan Dauchy, Ron Brightwell, Neil Pundit, Mike Davis, Tim Ingebritson, and Jim Ang have provided 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.

## 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 [9] provides additional RAS details on Sandia Systems.

#### Red Storm [10]

- “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*) [11]

- (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 [12]

- “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*) [13]**

- (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 [9]**

- “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. 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.
7. Enrique Vargas Sun Microsystems Enterprise Engineering. High availability fundamentals. Revision 01, November 2000. <http://www.sun.com/blueprints>.
8. Adrian Wong, Leonid Oliker, William Dramer, Teresa Kaltz, and David Bailey. ESP: A system utilization benchmark. In *Proceedings of the Supercomputing 2000 Conference*, 2000.
9. Suzanne M. Kelly and Jeffrey B. Ogden. An investigation into Reliability, Availability, and Serviceability (RAS) features for massively parallel processor systems. SAND2002-3164, 2002.
10. ASC red storm acceptance test plan. Cray and Sandia National Laboratories internal document.
11. Institutional computing cluster (ICC) - statement of work. RFQ 5031, Sandia National Laboratory.
12. Purple - statement of work (B519700). UCRL-PROP-145639DR, University of California Lawrence Livermore National Laboratory.
13. Thunder - statement of work (B532746). UCRL-MI-200098, University of California Lawrence Livermore National Laboratory.