SAND2005-2985C

Red Storm Capability Computing Queuing Policy

James A. Ang, Robert A. Ballance, Lee Ann Fisk, Jeanette R. Johnston, and Kevin T. Pedretti
Sandia National Laboratories*
Albuquerque, NM 87185-0817
jaang@sandia.gov

Abstract

Red Storm will be the first Tri-Lab [Sandia National Laboratories (SNL), Los Alamos National Laboratory (LANL), and Lawrence Livermore National Laboratory (LLNL)], U.S. Department of Energy/National Nuclear Security Administration, Advanced Simulation and Computing (ASC) platform to be managed under an explicit capability computing policy directive. Instead of allocating nodes among SNL:LANL:LLNL in the 2:1:1 ratio, *Red Storm* will use PBS-Pro (the commercial version of the Portable Batch System), to manage priorities among the labs so that in the long run their node-hours of usage will follow the 2:1:1 ratio. The basic queuing policy design is described along with extensions to handle switching between classified and unclassified, use by ASC university partners, priority access, etc.

Keywords

capability computing, fair-share, queuing policy.

1.0 Background

Our community now has valuable definitions of capability and capacity computing.

> *Two commonly used measures of the overall productivity of high-end computing platforms are capacity and capability. The largest supercomputers are used for capability or turnaround computing where the maximum processing power is applied to a single problem. The goal is to solve a larger problem, or to solve a single problem in a shorter period of time. Capability computing also enables the solution of problems that cannot otherwise be solved in a reasonable period of time (for example, by moving from a two-dimensional to a three-dimensional simulation, using finer grids, or using more realistic models). The main figure of merit is time to solution. Smaller or cheaper systems are used for capacity computing, where smaller problems are solved. Capacity computing can be used to enable parametric studies or to explore design alternatives; it is often needed to prepare for more expensive runs on capability systems. Capacity systems will often run several jobs simultaneously. The main figure of merit is sustained performance per unit cost. There is often a trade-off between the two figures of merit, as further reduction in time to solution is achieved at the expense of increased cost per solution different platforms exhibit different trade-offs. Capability systems are designed to offer the best possible capability, even at the expense of increased cost per sustained performance, while capacity systems are designed to offer a less aggressive reduction in time to solution but at a lower cost per sustained performance.* [1]

ASCI Red became the first ASC[1] capability platform, becoming operational at Sandia National Laboratories in 1997. *ASCI Red* still continues its long and distinguished presence on the Top500 list where it appeared in the Number One position from December 1996 to November 2000, before being surpassed by *ASCI White*. While *ASCI Red* may have lost the title of “World’s Fastest,” in the four following years it was still the “World’s Largest.” This was not in physical size, but rather in logical size – the sheer number of processors. It was only in the November 2004 list that *ASCI Red* surrendered this title to *BlueGene/L*

In many strategic ways system size is important. Tasks that are easy, perhaps even taken for granted at small scale, can become so problematic at large scale that the ability to obtain productive work from these systems is severely limited. Once the investment is made to achieve performance at large scale, that investment should not be squandered by using a capability system to run capacity jobs. This is not a matter of capability versus capacity systems. The fact is that our user communities need to have a balance of both types of systems with a recognition that those jobs/applications that do not need the scalability performance of a capability system should be run on capacity systems. The ASC program now recognizes that its investment in capacity systems is important to free up our capability systems for their intended workload.

In January 2004 Sandia first learned of the new ASC policy directive for capability systems. This policy is stated as: 80% of the node-hours of utilization must be allocated to jobs that run on 40% or more of the system. While *ASCI Red* could routinely be used for capability jobs, this was not the case for the subsequent ASC platforms, *Blue Pacific*, *Blue Mountain*, *ASCI White*, or *ASCI Q*. In fact, *ASCI White* and *Q* were allocated among the Tri-Labs according to a fixed node allocation, 50% for the host lab and 25% for each of the sister labs. These capability usage limitations led to the new ASC capability

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

[1] ASC was formerly known as the Accelerated Supercomputing Initiative (ASCI). This paper uses ASC to refer collectively to these programs.

policy directive to ensure *Red Storm* would not be carved up to allocate nodes to the different user institutions.

Because this is a new policy, Sandia will be the first ASC lab to define how to implement the capability policy. This paper provides a snapshot of our preparations to track Capability Usage Performance (CUP) as a metric that is factored into evaluating and setting job priorities. We also describe how we intend to allocate node-hours among the Tri-Lab and ASC Alliance Center user communities instead of using static node allocations. Since Red-classified/Black-unclassified switching is anticipated to be a regularly exercised capability of *Red Storm*, this paper also describes our analysis and process for dealing with this additional level of complexity.

2.0 Description of Red Storm

ASCI Red was designed to accommodate Red/Black switching. The *Red Storm* system carries on this tradition with a couple of important enhancements. First, *Red Storm* is designed to incorporate two more planes of switches to routinely accommodate changing the entire compute partition to either Red or Black operation. The *ASCI Red* system only had switches at the 25% and 75% planes, to shift between small and large configurations. Throughout its history, *ASCI Red* has actually had both its switch planes closed along with manually disconnecting the Red or Black disks to create a jumbo configuration. So we know this is a useful option – especially for a capability platform. Second, Cray engineered a very nice switch for simultaneously opening or closing over 2,200 signal pins with the throw of a single lever. This allows a switch plane to be opened or closed by moving only twelve levers.

By definition, whenever *Red Storm* is available to users for production use, the global *Red Storm* system will have nodes allocated between the Red Section and the Black Section according to one of the four global *Red Storm* system configurations shown in Table 1 below.

Red Section		**Black Section**	
40% Goal Nodes	Compute Nodes	Compute Nodes	40% Goal Nodes
0	0	10,368	4,147
1,075	2,688	7,680	3,072
3,072	7,680	2,688	1,075
4,147	10,368	0	0

Table 1. Allocation of the maximum available compute nodes between the Red and Black Sections for the four global Red Storm system configurations. Also indicated is the minimum job size for each section to qualify as a capability job.

3.0 Definition of Job Queuing & Scheduling Terms

We use the standard terminology for queuing systems and resource schedulers. In addition to some basic definition of terms we will also define some of the parameters that are specific to the *Red Storm* queuing model. New terms will be italicized.

UserID – The conventional Unix UserID; this must be verified to be unique.

GroupID – The conventional Unix GroupID

InstitutionID – An arbitrary string that identifies which Institution a user is affiliated with. The Institutions entitled to a share of *Red Storm* are: Sandia National Laboratories (SNL, S), Los Alamos National Laboratory (LANL, LA), Lawrence Livermore National Laboratory (LLNL, LL), and Alliance Center University Partners (AC). [2]

PBS-Pro – The Portable Batch System, a configurable job scheduling system from Altair Engineering.

Node – A single computer within the MPP System. On *Red Storm* a node is a single processor Opteron.

Node-hours – A measure of resource utilization calculated by multiplying the number of nodes used in a job by the total job duration.

Fair-share – A scheduling policy that examines past usage and adjusts job priorities to favor users with low comparative use. [3]

Fair share decay – The rate at which past usage is "forgotten", or has a reduced impact on calculating priority, over some arbitrary time period.

Backfill – A scheduling optimization that makes use of nodes that would otherwise be idle by scheduling jobs that will finish before all of the nodes needed for the next job are free.

Capability Usage Performance (CUP) – A measure of how much of a resource is being used for 'Capability' jobs. For ASC capability systems, the goal is for 80% or more of the node-hours to be used by jobs that use at least 40% of the nodes in a classified or unclassified section. This goal is represented as: $CUP_{40\%} \geq 80\%$.

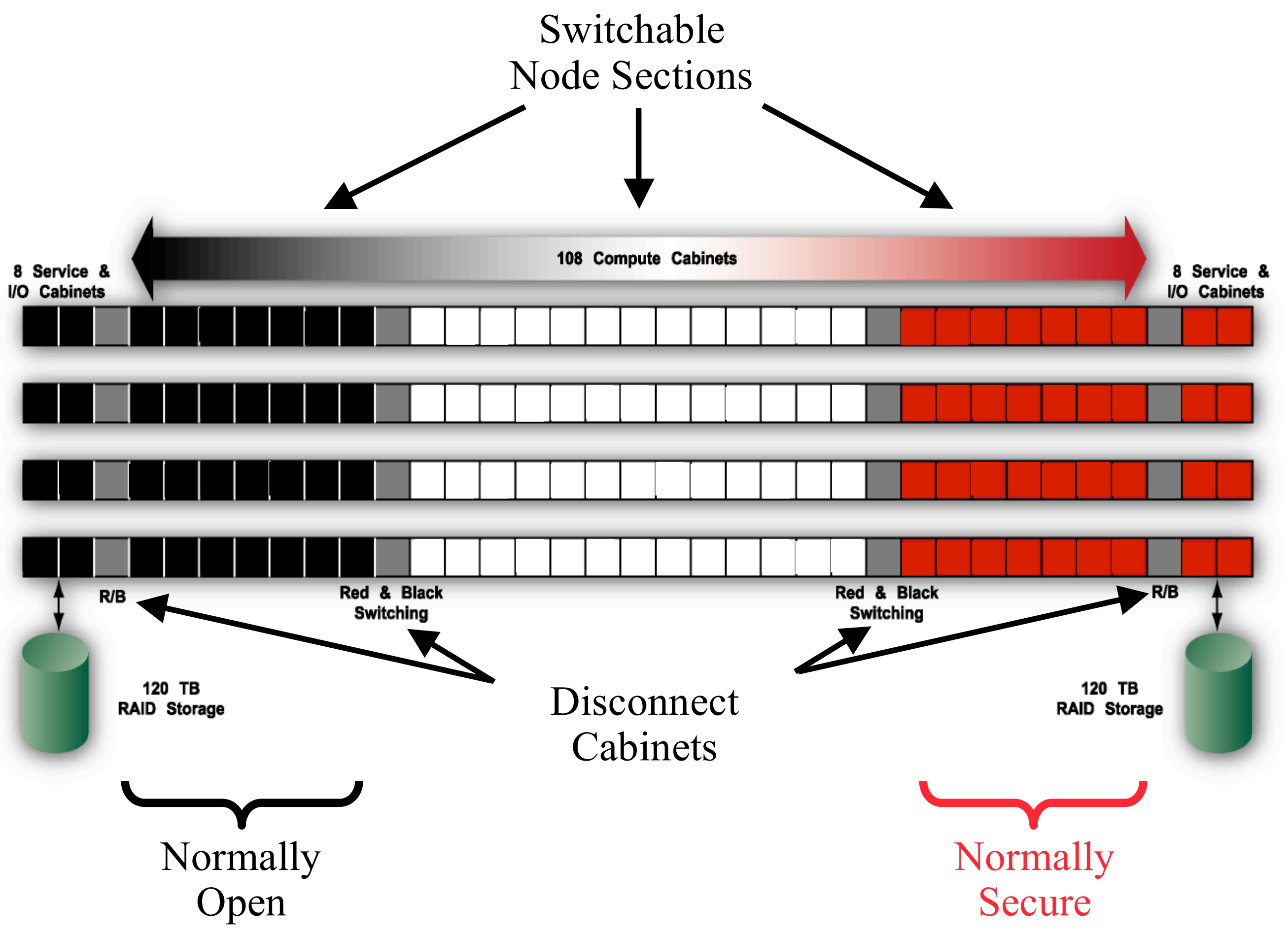
Other Job Limits:

Duration Limits – Job duration limits for *Red Storm* will typically be 72 hours, but there may be differences between workdays, workweek evenings, and weekends.

Size Limits – Job size limits will be minimum or maximum job limits.

[2] Note, the ASC Level Alliance Centers; University of Illinois, University of Chicago, University of Utah, Stanford, and Caltech are each entitled to an equal share of the Alliance Center global shares. In the following discussion the Alliance Centers are treated as a single institution, but they are really five distinct universities so the AC share is further divided by five.

[3] The convention for Tri-Lab fair share is not based on distribution of bank points. This means users and institutions are not guaranteed to receive their allocated node-hours. The only way to ensure receiving allocated shares is to be a regular, long term user.



Black Section, $f_B = 26.9\%$	Red Section, $f_R = 73.1\%$
$f_{LL,B} = 8.4\%$	$f_{LL,R} = 27.7\%$
$f_{LA,B} = 12.6\%$	$f_{LA,R} = 26.1\%$
$f_{S,B} = 41.8\%$	$f_{S,R} = 46.2\%$
$f_{AC,B} = 37.2\%$	$f_{AC,R} = 0\%$

Figure 1 . *Targets for Red/Black Section ratios and individual institutional target allocations with the Red and Black Sections as a function of the Tri-Lab requested allocation for the Red Section. For example, if LLNL wants 90% of its allocation for classified computing, then their choice for $LL_R = 0.90$. Similarly, if LANL wants 85% of its allocation for classified computing, then $LA_R = 0.85$ and if SNL wants 75% of its allocation on the Red Section, $S_R = 0.75$. Since the Global Red Storm allocations among LL, LA, S, and AC are respectively, 0.225, 0.225, 0.45 and 0.10, then, for the above choices of classified computing, the fraction of node-hours that Red Storm is integrated as a Red Section is $f_R = 73.1\%$ and analogously, the fraction of node-hours Red Storm is integrated as a Black Section is $f_B = 26.9\%$. In addition, we have all the information we need to assign target node-hour allocations for each institution on both the Red Section and the Black Section of Red Storm. Working through the arithmetic for this example we obtain the allocation values shown in the bottom of this figure.*

4.0 Definition of Red Storm resource allocation parameters

The following global *Red Storm* shares are set by agreement of the ASC executives:

$AC_{sh} = 0.10$
$LA_{sh} = 0.225$
$LL_{sh} = 0.225$
$S_{sh} = 0.45$

Where AC is the abbreviation for Alliance Centers, LA for Los Alamos, LL for Lawrence Livermore and S for Sandia. These global shares reflect the fraction of the total *Red Storm* system that is allocated to each institution. Institution-requested usage allocation for classified node-hours of usage (on the Red Section) results in assigning values between 0 and 1 for the following parameters:

$AC_R \equiv 0$
LA_R
LL_R
S_R

These requests also specify the Institution-requested usage allocation for unclassified node-hours of usage (on the Black Section):

$AC_B = 1 - AC_R = 1$
$LA_B = 1 - LA_R$
$LL_B = 1 - LL_R$
$S_B = 1 - S_R$

Based on these values, the fraction of node-hours the system should be Red is:

$$f_R = LA_R \times LA_{sh} + LL_R \times LL_{sh} + S_R \times S_{sh}$$

The fraction of node-hours the system should be Black is:

$$f_B = 1 - f_R$$

The fraction of node-hours that *Red Storm* spends integrated into either the Red or Black Sections, is the goal over some long time frame, e.g., one year. In practice, these node-hour fractions will be targeted by adjusting the duty cycle that Red and Black Sections spend in the configurations shown back in Table 1.

For the Red Section the following parameters define the node-hour allocations among the three labs:

$f_{LA,R} = [LA_R \times LA_{sh}] / f_R$
$f_{LL,R} = [LL_R \times LL_{sh}] / f_R$
$f_{S,R} = [S_R \times S_{sh}] / f_R$

For the Black Section, the following parameters define the node-hour allocations among the three labs and the Alliance Centers:

$f_{AC,B} = AC_{sh} / 1 - f_R$
$f_{LA,B} = [(1 - LA_R) \times LA_{sh}] / (1 - f_R)$
$f_{LL,B} = [(1 - LL_R) \times LL_{sh}] / (1 - f_R)$
$f_{S,B} = [(1 - S_R) \times S_{sh}] / (1 - f_R)$

An example problem is provided in Figure 1 where these node-hour fraction values are given assuming some initial values for Institution-requested classified computing allocations.

5.0 PBS fair share scheduling algorithm

To achieve the desired distribution of node-hours among the users of *Red Storm*, the PBS fair-share scheduling algorithm is being used. Each institution is assigned a percentage of the machine, which is calculated using the method of the previous section. Each user in each institution is assigned an equal share of that institution's node-hours. We have separate PBS configurations for the classified side and the unclassified side.

When a user runs a job, the user's node-hours are accumulated in a usage file. Usage for his or her institution is also incremented. A job's node-hours are the product of the number of nodes it used times the amount of time it had those nodes. The node-hour usage will be *decayed* over time. The standard way to do this in PBS is to define a configuration parameter indicating the *half-life* of used node-hours. When this half-life, perhaps three to fourteen days or longer, has elapsed, PBS will divide all usage totals in half. This roughly approximates a decay of resource usage corresponding to the rate of the specified half-life.

PBS, when configured to use fair share, uses the institution and user usage totals and the institution and user machine share when choosing the next job in a queue to run. When comparing two jobs that are competing for machine resources, it first compares their institutions. If one has a lower usage/share quotient, that job will have higher priority. If the institutions measure the same, then the users' individual usage/share quotients are compared. The one that is lower will have higher priority.

This fair share approach does not guarantee that an institution will receive its assigned share of node-hours in any given year. Providing such guarantees would result in lower machine utilization rates. It does however ensure that at any point in time, the best effort is being made to assign nodes fairly to those institutions and users that are submitting jobs to the machine.

6.0 Definition and Integration of Queues

The previous section described how the PBS fair share algorithm is being used to allocate the machine in accord with the established shares. The queuing policies for past ASC platforms are described in reference [2] in much more detail. A related study [3], investigated the impact on queuing policy of very long job duration limits that were possible with high-reliability systems. This section describes how PBS queues will be set up to meet the $CUP_{40\%} \geq 80\%$ goal in a classified or unclassified section. We are accomplishing this goal through the use of four PBS job queues listed in order of increasing priority:

Standard – All users may submit jobs to the Standard queue. Jobs submitted to the Standard queue must use less than 40% of the available nodes.

Express – Access to the Express queue is controlled by the Red Storm system management team. Jobs submitted to this queue must also use less than 40% of the available nodes. All jobs in the Express queue will be considered for execution by the PBS scheduler before any job in the Standard queue. The Express queue exists to fast-track ordinary jobs under exceptional circumstances.

Large – Any user may submit jobs to the Large queue. Jobs submitted to the Large queue must use at least 40% of the section's nodes. All jobs in the Large queue will be considered for execution before any job in the Express queue. *Gaming* the system by requesting large numbers of nodes for jobs that do not require large numbers of nodes will be considered a very serious offense.

Expedited – The Expedited queue is for any job which is deemed to be extremely urgent. Admission to the priority queue is granted by the Expedited Priority Run committee. Jobs submitted to this queue are not subject to any job size or job duration limits. All jobs in the Expedited queue will be considered for execution before any job in the Large queue.

PBS will evaluate the queues in queue priority order. When evaluating the jobs in a queue against one another, PBS will use the fair share algorithm described in the previous section.

The PBS scheduler for *Red Storm* has been modified to treat the highest priority job for which there are insufficient nodes as a *starving* job. The scheduler will *backfill* while waiting for the highest priority job to begin. No other job in the queues will be placed into execution unless it will complete before the starving job can run, or unless it will use nodes that the starving job will not use when it begins to run.

7.0 Complicating perturbations

Dedicated Application Time

Our assumption for dedicated application time will be that all nodes in a partition to serve dedicated application time are all in use for the duration of this time. This is because these nodes are not available for any other users.

Interactive Usage

Interactive usage is an essential part of the effective use of any ASC capability platform. Activities such as application development and debugging, post processing and visualization all require interactive use of the system. Since interactive jobs will not be started through PBS, they will not currently be tracked in our $CUP_{40\%}$ metric.

Job failures

Our current assumption is that node-hours of usage from job failures that can be traced back to a system or machine issue, will not be counted as productive utilization, regardless of job size. Therefore, these node-hours will also not be tracked in our $CUP_{40\%}$ metric.

Backfill Jobs

The issue of how to treat backfill jobs illustrates the area were our capability computing policy directive is directly at odds with another common System-level performance metric – utilization rates should be maintained at or above 80%. One important way in which computer facilities maintain high utilization rates is through backfill. The inherent characteristic of backfill jobs is that they are small. They have to be small to fill in holes between currently running jobs. While backfilling jobs improves utilization rates, this will also reduce our $CUP_{40\%}$ value. The CUP metric is a ratio of utilizations:

$$\frac{\text{node-hour}_{\text{Large Jobs}}}{\text{node-hour}_{\text{Large Jobs}} + \text{node-hour}_{\text{Small Jobs}}}$$

Where "Large Jobs" use 40% or more of a section. One obvious way to increase this ratio is to suspend backfill. This will reduce the $\text{node-hours}_{\text{mall jobs}}$ and increase the $CUP_{40\%}$, however utilization rates will suffer.

Another option may be to modify the CUP metric so backfill jobs are not included in the small job total. This modified $CUP_{40\%}$ would be calculated as:

$$\frac{\text{node-hour}_{\text{Large Jobs}}}{\text{node-hour}_{\text{Large Jobs}} + \text{node-hour}_{\text{Small Jobs}} - \text{node-hour}_{\text{Backfill Jobs}}}$$

Such a modified CUP could provide resolution of the conflict between maintaining high overall utilization rates and having high capability utilization. In order to report on this modified CUP metric we may need to add the ability to separately track node-hours used by backfill jobs.

Heterogeneous nodes

Jobs run on *Red Storm* may request service nodes as well as compute nodes from the queuing system, and can launch jobs on either service nodes or compute nodes or both. Hence, *Red Storm* is a heterogeneous system. (The service nodes run the Linux OS, the compute nodes run Cray's Catamount OS.) Further, the system management tools permit labeling each service or compute node with a node specification, indicating special properties the node has. For example, some of the service nodes will have graphics cards, and may be labeled with the specification *viz*.

The scheduling system will permit users to request these special nodes types. To improve overall usage of the machine, the scheduling software will change a job's request for generic nodes to one for special nodes, if an insufficient number of generic nodes are available.

8.0 Unintended consequences

Our user community is reasonably intelligent, and as is typically the case, while they are waiting in the queue for their jobs to start, they usually have time to think of ways to game the queuing policy. We will need to think through how to preserve the intent of the new ASC capability computing policy directive. Some unintended responses by users could include:

- In their efforts to raise job priority, users move away from sizing jobs based on the amount of memory they need to contain their problem.
- In another effort to raise job priority, users may force applications to run at scales where they have poor parallel efficiency. This could be worse than using a capability system to run capacity jobs.
- Another factor is that some users may simply bundle 10, 100, or more small jobs into one big job that unfolds to run on 40% or more of the system.
- Allow nodes to idle, i.e., run while (1).

9.0 Future Work

This paper is a report of a work in progress. While the general structure of this approach is believed to be sound, to date we have identified a few additional capabilities that will be required before we can test this queuing policy implementation. However, we have begun to think through some of the options for how we can add $CUP_{40\%}$ as an additional factor that is weighed when determining fair share job priority. Identification of additional needs include:

- Provide a way to do the nightly evaluation of $CUP_{40\%}$. While most of this data is available in the Compute Processor Allocator Database, the current database implementation does not track the size of the system. An alternate approach is to obtain this data from Sandia's Scientific Computing AIRS [4] capability but again, some work is needed to add section size to that database as well. With the ability measure and update our $CUP_{40\%}$ performance on a daily basis we can explore the potential to weight PBS fair share priorities for capability jobs.
- If $CUP_{40\%} < 80\%$, then use $P_{CUP} = [0.80 - CUP_{40\%}] \times [JobSize_k/(0.40 \times \text{Red Section Size})]^p \times [\text{Standard PBS Fair share factor}]$, where P_{CUP} is the capability metric-weighted fair share job priority, p is a real value number to vary the power law sensitivity of the capability job size weighting. We will need to use system time to gain some test data and understand how to set this parameter.
- Correct the fair share calculation to refund "usage" tracked by jobs that actually failed due to node hangs.

10.0 References

[1] Susan L. Graham, Marc Snir, and Cynthia A. Patterson, editors, *Getting Up to Speed: The Future of Supercomputing*, National Research Council, Committee on the Future of Supercomputing, National Academies Press, page 24, 2005.

[2] Scott H. Clearwater, Stephen D. Kleban: "ASCI Queuing Systems: Overview and Comparisons," IPDPS 2002.

[3] Stephen. D. Kleban, Jeanette. R. Johnston, James. A. Ang, and Scott. H. Clearwater, "With Great Reliability Comes Great Responsibility: Tradeoffs of Run-time Policy on High Reliability Systems, 2004 IEEE Symposium on Cluster Computing and the Grid.

[4] Robert A. Ballance, Jared Galbraith, and Roy Heimbach, "Supercomputing Center Management using AIRS," Proceedings, Cluster World Conference and Expo, June 2003.