Lessons Learned When Building a Greenfield High Performance Computing Ecosystem

Andrew R. Keen
Michigan State University
Dr. William F. Punch
Michigan State University
Greg Mason
Michigan State University

ABSTRACT
Faced with a fragmented research computing environment and growing needs for high performance computing resources, Michigan State University established the High Performance Computing Center in 2005 to serve as a central high performance computing resource for MSU's research community. Like greenfield industrial development, the center was unconstrained by existing infrastructure. The lessons learned are useful when building or maintaining an effective HPC resource and may provide insight for developing other computational services.

TAGS
HPC, infrastructure, customer support, imaging & automating builds, configuration management, security

1. UNDERSTANDING THE ECOSYSTEM

The primary goal of a capacity HPC resource is to meet the resource's users' computational needs, rather than targeting maximum theoretical performance. The whole HPC ecosystem must be considered, including facilities, the supporting computational services, the human resources necessary to deliver these resources, and the policies to ensure optimum system utilization. However, when choosing HPC resources many administrators neglect these requirements in pursuit of maximizing theoretical performance. We learned this lesson early in the center's life.

The center began operations in 2005. The first system implemented was a 64 processor, large memory SMP system. The system performed very well on the original benchmark suite as defined in the RFP, and the fast floating-point performance, large per-processor cache, low-latency inter-processor interconnect, and significant memory bandwidth of the SMP system ensured that it performed well on those benchmarks. However, after the system was put into production, the system's performance proved unacceptable when multiple I/O-intensive jobs were ran. It was determined that the performance of the supplied locally attached disk system was inadequate.

- While the attached disk subsystem performed well enough to support a single workload, the low rotational speed (and low IOPS) of the attached disks degraded performance significantly worse than linearly as multiple requests were sent to the array.
- The job management policy as implemented by the job scheduler was designed to maximize processor utilization instead of maximizing peak efficiency. A more intelligent scheduling policy could have limited the number of disk-intensive jobs while leaving multiple processors unused.
- The benchmark cases, while reflective of the individual workload components of the center, did not reflect the center's day-to-day workload.

After profiling the problematic applications at a system (iostat, libffio [1], Performance CoPilot [2]) and storage level (on-controller operation, bandwidth, and latency statistics) the vendor and the center determined that a faster storage subsystem was required to allow the server to run multiple disk-intensive jobs. Implementing the storage added about twenty percent to the original purchase price of the system but it resolved the I/O bottleneck and reduced time processes spent in iowait. The center was able to reuse the original storage in a role it was better suited for.

2. CLUSTER ADMINISTRATION

When managing a HPC resource, we have found it to be important to define our functional requirements and the tools we use to address them. Issues involving hardware and software management, data storage, environmental constraints, availability, and security are some of the issues we have had to address; cluster and workload management tools have helped us address these challenges.

2.1 CLUSTER MANAGEMENT

A modern HPC resource requires many software components, including: authentication and authorization, data storage, data transfer, network management, hardware management, job management, operating system and configuration management. There are many software packages available to HPC administrators to accomplish each of these tasks; however, many have potential pitfalls in their default configuration that are nonobvious to the inexperienced administrator. First-time HPC service system administrators should strongly consider using integrated open source (e.g., Warewulf [3], ROCKS [4][5]) or commercial (e.g., Bright [6], Platform HPC [7]) cluster management solutions [8] as a way to avoid common mistakes and as a way to familiarize staff with the interactions between software subsystems. We have been through three cluster management environments and have found that there are a number of things to consider when selecting cluster management software.

- How easily does it integrate with other technologies? One integrated cluster management product we used did not natively support using an external LDAP server for client account information. We had to build a RPM to manually distribute the basic settings that were not supported by the management software.
- Does the product lock you into a specific vendor? An extensible product that uses standards like IPMI and standard software distributions is preferable. We have had issues with both commercial and open source products not supporting newer or exotic hardware.
- Consider firmware management options, particularly when purchasing new clusters. Important features include the ability to automatically update firmware and the ability to set commonly used configuration options. We have used Dell's OpenManage Deployment Toolkit [9] and setupbios from the Dell PowerEdge C System Management tools [10] to manage these environments. In HPC environments, BIOS configuration options ([11],[12]) can have significant performance impacts.
- Do the configuration management options provided match your policies? Does the software reinstall a system to update installed software or configuration? Will that interfere with scheduling or your SLAs' availability requirements? We've found that keeping the software environment on all compute nodes identical prevents problems with incompatible software versions. However, the problem is to find the appropriate balance between the improved availability provided by deploying changes while the system is running the job (rather than making changes offline) versus the possible problems caused by a temporarily non-homogenous configuration? How sensitive are your workloads to the CPU utilization of the configuration management tool? Tightly coupled MPI applications can be particularly sensitive to small disruptions.
- Is your workload certified or well suited for a given cluster management system? Some applications have advanced integration with certain workload management software ([13],[14].)

We originally started out managing a single SMP system by hand. When we purchased our first cluster, the vendor implemented node imaging with SystemImager [15]. However, we struggled with maintaining the proper workflow of making a change to a node, updating the image on the server, and pushing it out. Instead, changed files were frequently distributed using pdcp and commands were used with pdsh [16]. Changes made would be lost when systems were reinstalled; leading to regressions. To address this, we've separated the node installation from the node configuration. We have a very simple Kickstart [17] installation to bootstrap into puppet [18][19], where all of our configuration changes are stored. We also store the puppet configuration manifests in git [20] and have integrated branching with puppet's dynamic environments [21], which has simplified testing and implementing changes.

Your cluster management environment, whether self-maintained or part of a package, should include robust monitoring and management infrastructure. We use Cacti [22] for environmental and hardware-level device monitoring, Nagios [23] for service monitoring and alerting, and Ganglia [24][25] for performance information on the compute nodes. Ganglia provides per-node OS performance metrics, and Cacti is focused on out-of-band metrics and polls less aggressively. We have also developed automated regression testing that runs nightly and after every maintenance event to ensure that the systems are healthy; and implemented lightweight health checks [26] that run periodically and before and after every job. Nodes flagged offline by automated checks currently require administrator attention, but we are working with Moab's event-based triggers [27] to automate a manual testing workflow for hardware and software failures.

An isolated environment for testing infrastructure changes that is as close to the real configuration as possible is desirable. If you have automatic provisioning and configuration software (like we have with xCAT [28] and Puppet) you can install a test environment from your existing configuration, or clone system infrastructure virtual machines.

2.2 Workload Management

A HPC resource's workload can be any mixture of interactive and batch processing work. Understanding both the technical and political requirements and choosing software and policy that best reflects your users' needs is important. Poor choices can cripple the effectiveness of the resource and frustrate users. We have had four major iterations of our scheduling policy as we've learned and adapted to our workload.

2.2.1 Workload Management Software

While UNIX and most of its descendants have supported multi-user time-sharing since 1974 [29], the traditional POSIX tools for managing user workloads break down at multi-node scales and on modern hardware. Therefore, most HPC sites use workload management software to address these problems. The simplest method is to run workload daemons provided by software vendors (like Mathworks' MATLAB [30] and Wolfram Alpha's Mathematica [31]) that integrate directly into the software's normal interface. While convenient, they are generally only well suited for small clusters or clusters that are dedicated to a single application. Anecdotally, most HPC resources use multi-node batch queuing software for workload management; we've used both PBSPro and TORQUE. We've chosen to use OpenPBS-derived systems to minimize disruption when transitioning between systems, and some of our application software integrates well with TORQUE and Moab.

2.2.2 *Job Management versus Job Scheduling*

Most resource management packages consist of two major components: a job manager that starts, monitors, and stops the workloads on the nodes in the cluster, and a job scheduler that directs the actions of the job manager. While TORQUE includes a scheduler, its functionality is very limited. We use a Moab Workload Manager [32], a commercial product from Adaptive Computing that allows us to set policies that provide better service to our users.

2.2.3 *Job Scheduling Policies*

In our experience, determining appropriate scheduling policy can be seen as balancing competing goals of responsiveness and utilization. To generalize, users want their work to complete as soon as possible (often expressed as a desire to have their job start sooner), while management wants high utilization to maximize return on their investment. We've implemented a number of limits (wall-clock, total jobs in use, the total number of CPUs used) to ensure fair and reliable access. These limits impact both user experience and system functionality. Limits should be high enough to meet user needs while maintaining fair access. Longer wall-clock limits limit the effectiveness of fair-share, make it harder to schedule other jobs and maintenance, and can increase the amount of work at risk from power or equipment failures. We currently allow up to a week of wall-time per job, but would like to reduce this. We've encouraged users to reduce their wall-clock time by allowing users who run less than four hours to run on idle nodes that other users have bought priority access to, by implementing trial support for system-level checkpointing with BLCR [33][34], and by educating users about ways to checkpoint and auto-resubmit their jobs [35].

We needed to choose the minimum resource increment that users could request, whether per-core, per-node, or the entire system. Our impression is that larger systems are generally scheduled in per-node or larger increments, and per-core was seen more frequently on small and medium-sized resources. We've chosen to schedule the majority of our cluster on a per-core basis, choosing throughput over peak single job performance. We can combine a 1 CPU, 20 GB job and seven 1 CPU, 256 MB jobs on a single 8 CPU node with 24 GB of RAM, or two jobs with 1 CPU and 1 GPU each, and 6 CPU-only jobs on a node with two GPUs and eight CPU cores. While there can be a performance hit when sharing nodes among workloads, we haven't found it to be problematic for most of our workload, as long as the cores themselves are not shared. Memory or I/O bound jobs, larger jobs, and benchmarking runs can still request whole nodes. Some sites use Moab's generic consumable resources to manage non-CPU resources as reserved resources to prevent contention [36].

We've found that setting short default wall-clock limits (1 minute) encourages users to set an accurate request, which makes scheduling more predictable. We're implementing further TORQUE submit filters [37] to give users immediate feedback about potential problems with their jobs.

We have learned to avoid linking queues to resource selection if at all possible. It increases the barrier for new users learning to use the systems, requiring them to determine what queue is best suited for a particular job. This proved to be frustrating to our users and non-transferable to other sites. We originally had queues and dedicated limits for given job categories (high CPU count jobs, long running jobs.) Utilization suffered, and users were unhappy because there were idle cores that were not available for their workload. In addition, queues are usually assigned at submit-time; if jobs are linked to specific hardware by the queue the system is unable to take advantage when other hardware becomes available. While time and resource specifications are largely similar on most HPC resources, queue names and their policies are unique to each site. We use a combination of node features, job attributes, credentials, node utilization, and reservations to place jobs on appropriate hardware. By standardizing our hardware and environment we've maximized the number of nodes a given job can run on.

2.3 Understanding availability requirements

When building your system, consider the impact of a single component's failure. A single node crashing is not disruptive to the center's mission if your workload can be restarted from a checkpoint file or resubmitted, but if one of the critical services goes down you can lose the entire cluster's workload. Building a robust core infrastructure is important if the total cost of a downtime including lost productivity is considered. When building a HPC resource, you should consider building infrastructure at a higher reliability level than the general compute cluster.

One of our clusters was designed with two network fabrics, a high-speed Infiniband network and a gigabit Ethernet network. However, there was a significant amount of traffic on the gigabit Ethernet network that regularly caused the entry-level gigabit switches it was connected to to crash. By replacing these switches with higher quality switches we were able to prevent these interruptions in service. If we had better understood our user workloads we could have specified better Ethernet switching hardware in the bid instead of incurring the cost of replacing them later.

Initially, a single physical node served both as the user login server and the infrastructure server. This proved to be very extremely problematic, as a single user with a misconfigured application on the login node could disrupt important system-wide resources such as the LDAP server.

By separating user nodes and infrastructure systems, we provided a clear divide between the service environment and user environment. As the center has added additional systems (now numbering in the thousands of general purpose cores as well as specialized hardware) our infrastructure has grown as well. Where appropriate, we added redundancy to services. Later, we migrated from physical machines to virtual machines for all non-storage infrastructure systems, minimizing the impact of hardware failure. Using that framework we are now deploying user-requested VMs from templates for applications with special needs, such as web services.

2.4 Storage

Data management is an important component of a HPC environment. There are at least three categories of storage that you may need to consider in a HPC environment: home directories, temporary storage, and archives.

2.4.1 Temporary Storage

Initially, the only scratch space provided was directly attached to the large memory systems, as described previously. Beginning with our first cluster in 2005 we have deployed three generations of the Lustre [38][39] file system.

There are a number of challenges when dealing with parallel I/O. Our first Lustre installation was too heavily biased towards reliability; the lack of capacity significantly impeded users' ability to do large research. Our second implementation concentrated too heavily on capacity and had no server-level redundancy; a single server failure would cause disruptions in scheduling. That is, jobs with files on the failed server that would normally complete within the time originally requested would be delayed long enough to push them past the scheduled end-date, at which point it would be killed by the job scheduler. We designed our most recent purchase to be well balanced between performance, capacity, and reliability. The Lustre servers are once again highly available, and no single point of failure exists in the storage stack. As scratch is designed to be temporary storage, we automatically purge files older than 45 days from scratch. This too has proven problematic and requires gentle "reminders" as files age on scratch. We are greatly cognizant that permanently removing files is potentially disastrous to the user so we often rotate files to offline storage before deletion, where the length of time they remain there depends largely on the available storage.

2.4.2 Persistent Storage

For permanent home directory storage, the center provides hundreds of terabytes to internal clients via NFS and campus and VPN clients via CIFS. We added ZFS-based [40] storage in 2009 as a replacement for a 15 TB Linux server that used XFS [41] and LVM. The cost, performance, ease of administration, and data integrity features were compelling reasons to choose ZFS.

We use 7,200-RPM SATA or near-line SAS hard drives for bulk storage, and use SSDs for read (L2ARC) and write (ZFS Intent Log) caches. While we have found that a write cache can significantly improve NFS performance (an order of magnitude in improved performance on one user benchmark), we have found that the separate read cache device is unnecessary for our workload (by looking at read and write statistics from the Illumos [42] kernel), so we will not use them in future systems. In our environment, separate read-cache devices would only be helpful with per-filesystem tuning. For us, it is much more efficient to simply add more DRAM to a system to leverage the ZFS ARC and kernel page cache.

With ZFS, creating new snapshots is non-disruptive. We create hourly and daily snapshots locally, which are directly available to the user. Nightly backups are replicated from the snapshots offsite to an external server. We've used the zfs-auto-snapshot SMF service in OpenSolaris and are migrating to a simplified version of this same process on new systems. An internally developed script handles offsite replication.

ZFS also provides robust on-disk data integrity protection. We have not lost a single bit of data to either silent data corruption or hardware failures.

The ability to do thin provisioning is also useful when creating new user accounts. Later updates to ZFS have added block-level deduplication [43] and compression [44]. Compression is quite useful for offsite backups. We've found it quite easy to achieve 2x or greater savings with the gzip compression found in later versions of OpenSolaris and Illumos with our users' datasets.

2.4.3 Archival Storage

We do not provide archival storage, but are working with university departments to meet the needs of the research community, including ways to provide components of future NSF grants' data management plan requirements. Currently, we work with users to migrate data to national data repositories.

2.5 Physical Concerns

HPC system density has increased well beyond the traditional datacenter standard of three to five kilowatts per rack [45]. The center has undergone three major renovations as density has increased. The first systems were low-density that could be cooled with a forced air raised floor. The second set of systems averaged about 12 KW per rack with two CPU sockets per rack U. Presently, we target a density of twenty five kilowatts per compute rack, using blades or blade-like systems, with two systems with two CPU sockets (a total of 4 sockets) per rack U. Our data center is space constrained, so density is important. It would not be possible to cool these dense systems with traditional forced air distributed by the 12" raised floor in our facility. While some new facilities (like NCSA's National Petascale Computing Facility [46]) are able to support systems of similar or greater densities with a 6' raised floor, that is not feasible in our_facility. We instead use Liebert XD [47] high-density refrigerant-cooled in-row spot cooling systems to supplement the existing under floor cooling. We are also considering higher density water-cooled rack systems in the future. We've seen significant improvements by implementing cold aisle containment. We were able to prototype closing the cold aisle by using cardboard and were able to observe a significant reduction in overall room and system inlet, component, and output temperatures. We then chose to implement a permanent installation based on this evidence.

It is important to coordinate new purchases with the people responsible for your facilities to understand what your constraints are and choose the cluster configuration that best meets your needs within those constraints. We cannot add any more power in our current facility. This has informed our hardware retirement policy; we retire hardware sooner to make room for new, denser, more efficient hardware.

2.6 Security

HPC resources can be more difficult to secure than standard Unix systems, given that most environments allow users compiler access and the ability to upload arbitrary binaries and scripts. Most HPC systems do not use common security features like firewalls and virus scanners on individual nodes due to the associated performance penalty. Choosing the right balance between usability and security is a decision that each site must address based on the risk of attack and the institutional, regulatory, and legislative requirements for data security. The full range of permissions has been fielded, from only allowing administrator-installed executables and validated input files to allowing users root access to systems.

Administrators can limit the utility of stolen credentials by limiting the sources that those credentials are valid from and by using implementing two-factor authentication. We have mitigated the potential risks by separating user-accessible systems from administrative systems, by isolating system control traffic from the user accessible research networks and by isolating most cluster systems from the public Internet. We are also implementing SELinux [48] on infrastructure systems where appropriate and use Fail2Ban [49] to limit brute force attacks.

It is useful to think about the trust relationships between systems. Some cluster management software's default configuration allows any root user on a system that it manages to access any other system in the cluster as root, including administrative systems. This can effectively compromise the entire cluster if a single machine is compromised. In general, systems that are trusted should minimize the number of systems that they trust. Remote root access, when needed, should be restricted to connections from hardened hosts and should rely on agent-based public key authentication from external systems rather than password-based authentication.

3. User Experience

3.1 Provide a Stable Environment

It is important to choose an appropriate operating system whose lifecycle matches your needs and resources. The advantage of enterprise distributions (like Red Hat Enterprise Linux [50] and derivatives like Scientific Linux[51]) is that the long support window reduces the number of disruptions users from major upgrades. However, the adaptation of new hardware support, system libraries, and kernel features often lag community or development distributions like Fedora [52]. Many commercial software packages only support commercial enterprise distributions or their derivatives.

On our first system, user access was constrained to queue submissions to ensure that processors were not oversubscribed. Since the login system's architecture (x86_64) was incompatible with the SMP system (ia64), users were forced to submit a job or request an interactive job to compile or test their code. Depending on utilization, users could face significant delays for development. Rather than dedicating expensive processor slots on the large system to interactive or debugging queues, we instead purchased a small development system where users could do development work and simple testing without wait and at a much lower cost per processor. This has worked so well that we have replicated this model for our cluster systems. We've also added specialized development nodes that don't have a corresponding cluster as a way to allow users to develop, test and evaluate their applications on new systems. Some have shown significant adoption by users and lead to larger cluster purchases (GPUs) and some have not (Cell processors.)

An environment modules system allows administrators to deploy multiple versions of software such that the user can choose which versions and combinations of software are appropriate for their work. Furthermore, bundles of software modules can be created for common workflows. We have switched to lmod [53] [54] in March 2012, as it addressed a longstanding bug with Environment Modules [55]. When loading modules within modules (important for some tool chains and in particular bioinformatics, where the outputs and inputs of several independently-developed tools must be combined), we would see intermittent memory corruption [56].

3.2 Naming

As the center added systems and users, the naming conventions became more of an issue. The center's first system and its partner node were named after the university's colors. After we (predictably) ran out of school colors, we expanded the naming system to use notable former faculty, but they had additional, unintended meanings. In addition, we used the same name for the host name of the development node for that cluster and as an alias for cluster as a whole. Users often misunderstood the relationship between the cluster name and the login node. To address this we designed a functional and consistent naming convention that reflected the underlying hardware architecture of the cluster at an appropriate level of detail, providing useful but not over-specific information about the cluster. We chose to identify clusters based on the year acquired and the processor or accelerator family. However, naming critical infrastructure after the platform can be problematic when hardware is added or replaced. Our first generation ZFS file servers were named 'thor-XX', Sun's product codename for the Sun Fire X4540s. When we added new file servers, we had to name them 'thor-XX', despite not being X4540s, because of assumptions made when implementing the initial systems. We've since stopped extracting configuration information from hostnames, and instead use metadata from Facter [57].

When choosing naming standards, it is useful to keep the user in mind, as their insight can be most valuable. The default batch queue was originally named 'dque', but the abbreviation was unclear to the users; we renamed it to 'main' based on user suggestions.

Upgrading and changing software components should be done incrementally when not visible to the user, as it simplifies diagnostics if problems are detected. We've upgraded the components of the workload management and infrastructure software incrementally, but have tied major changes in user experience together (compute node operating system upgrades and changes to module structure) to minimize the overall number of changes users have to make to their workflow.

4. Communication

Ultimately, a HPC resource is a service organization and it is important that users feel informed about the resources they use. In particular, we have found that transparency is the key to this connection. Users should be able to see monitoring and utilization statistics and predictions as to when their jobs will run. It is important to avoid making changes to the system without communication. In the end, the administrators will have much more latitude in their ability to effectively administer a system if users are informed. Even what may be minor changes from a system administrator's perspective can be important to users.

Effective user communication is an ongoing challenge. It has been our experience that direct, personally addressed emails are only read slightly more often than announcements on mailing lists. Thus we have explored other avenues of communication and utilize many regularly. We send out weekly newsletters with upcoming events and funding opportunities. We post blog updates on our documentation wiki, which is syndicated via RSS. This information is republished via two Twitter accounts; one focused on system availability and maintenance information [58], and one for general center information [59].

Another important point of feedback is provided when the user can track the status of requests for help. We use RequestTracker [60] as both a help desk and a request tracking system to ensure that user requests are responded to in a reasonable timeframe and to gather metrics on staff responsiveness for reporting. While not quantified, informal feedback from users indicates *perceived* responsiveness improves when we use a ticketing system. Even basic monitoring, like the number of new tickets, open tickets, and resolved tickets can provide visibility to managers as to the current effectiveness of the staff.

Collaboration tools like wikis can be very effective, with some caveats. If you allow users to modify wiki content, staff should monitor changes. Don't anticipate that users will reduce the administrators' documentation workload; the vast majority of edits on our wiki are by staff.

If you frequently bring new users online, we have found that holding user training on standard issues (login, module system, parallel tools, etc.) in addition to one-on-one help. Such sessions can provide excellent feedback on how to improve the system or documentation. We have begun the process of moving those training sessions to video. We will provide them online, allowing users the opportunity to pick when they want to learn (or review) a topic.

User data is very, very important. You should communicate clearly with users the guarantees of the data they should expect and be sure to update them aggressively when those policies change. A failure to communicate with users early in the center's life about deletion from another temporary directory resulted in a researcher losing a significant amount of work. When implementing a new automatic deletion policy on our scratch system, after previously relying on the honor system, we archived data after the first few passes and were able to recover data for users who had not noticed the announcements we had published.

Reporting is important for internal and external use. Hard metrics like CPU-time consumed, utilization, and average wait time are useful, but soft metrics like papers published, grants received, resources discovered or products designed can be more useful for the majority of resources that are not leadership-class facilities [61].

Inspired by a conversation with staff from the University of Michigan's CAEN Advanced Computing Group, we've also instrumented our environment module infrastructure with syslog to track software usage. Harvard University's FAS Research Computing has also implemented a similar approach [62].

Well-developed internal technical documentation aids consistency in implementation and between administrators and reduces the amount of time administrators spend re-implementing fixes. External technical documentation (ideally, with domain examples) reduces the learning curve for new users significantly. If policies for users are not visible, they don't exist. Users have perceived the enforcement of poorly documented procedures as capricious targeting by administrators. When users can see the potential benefit of enforcement they usually accept the need for such measures.

5. Other Resources

We have used external resources when we found them to be particularly suitable. The center was able to use our campus Kerberos authentication and to gather information about users from the campus LDAP. Some of our Ethernet network switches and our external firewall are managed by campus IT, and HPC staff manages HPC-focused resources (storage, high speed networking.) We try to avoid duplication of efforts and try to focus our resources on the goals of our center.

External resources include people as well as technology. We've worked with IT staff from other departments on campus that use our systems to add domain-specific support for users in their departments. By utilizing their knowledge, we were able to provide a better service for the resource's users.

Domain experts are valuable resources for users and for administrators as a bridge between dedicated systems administrators and users. They are as important of a component of the ecosystem as the hardware, software, and facilities.

The Information Technology Infrastructure Library [63] has been useful when addressing some of the challenges in this paper. While the author was initially skeptical, there have been a number of times it has been helpful when understanding the ways our procedures could improve. The library's sheer size and scope make it difficult implement whole cloth. We've had the best luck when using components of that fit well within the environment and as a reference when implementing support workflows.

Attending technical conferences, joining mailing lists, listening to technical podcasts and reading personal and professional blogs has been very helpful; networking with peers can provide some valuable insight into HPC.

6. CONCLUSION

We have encountered many challenges when deploying a HPC resource; understanding the service sponsor's goals, the requirements of the resource's users, the technologies and policies best suited to meet them, and the risks associated with providing that resource. We have found that published best practices and advice from colleagues can help staff understand HPC requirements more easily.

Have a clearly defined service sponsor and understand what their goals are for the service. Ideally, the service sponsor would be the person responsible for deciding whether to continue the project. A single person who can arbitrate between different stakeholders ensures that there is clear direction for the staff.

Make sure that you understand your stakeholders' needs and bring them into the decision making process; try to ensure that you have a representative sampling on any advisory boards. Your users should be invested in the success of the resource. Your harshest critics can provide useful information about your areas for improvement.

7. Acknowledgements

Michigan State University, for its recognition and support of HPC services on campus. Dr. Wolfgang Bauer, the current Institute director. Dr. Leo Kempel, who helped found the Center. Dr. Dirk Colbry and Dr. Ben Ong, research specialists, current and previous colleagues at the Center, including Kelly Osborn, Ed Symanzik, Ed Kryda, Eric McDonald, Jim Leikert, Greg Mason and John Johnston. The Practice of System and Network Administration by Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup, great reading for any system administrator. Doug Hughes for his invaluable help with this paper, and the LISA09 workshop on HPC.

[1] "SGI TPL (Linux: Developer/LX_AppTune - Chapter 7. Flexible File I/O)." [Online]. Available: http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=linux&db=bks&fname=/SGI_Developer/LX_AppTune/ch07.html. [Accessed: 12-Sep-2012].

[2] "Performance Co-Pilot." [Online]. Available: http://oss.sgi.com/projects/pcp/. [Accessed: 12-Sep-2012].

[3] "Warewulf." [Online]. Available: http://warewulf.lbl.gov/trac. [Accessed: 12-Sep-2012].

[4] "www.rocksclusters.org | Rocks Website." [Online]. Available: http://www.rocksclusters.org/wordpress/. [Accessed: 12-Sep-2012].

[5] P. M. Papadopoulos, M. J. Katz, and G. Bruno, "NPACI Rocks: tools and techniques for easily deploying manageable Linux clusters," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 7–8, pp. 707–725, Jun. 2003.

[6] "Bright Cluster Manager - Advanced Linux Cluster Management Software." [Online]. Available: http://www.brightcomputing.com/Bright-Cluster-Manager.php. [Accessed: 12-Sep-2012].

[7] "IBM Platform HPC." [Online]. Available: http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/hpc/index.html. [Accessed: 12-Sep-2012].

[8] D. Cable and M. Diakun, *A Review of Commodity Cluster Managment Tools*. Science and Technology Facilities Council, 2011.

[9] "Dell OpenManage Deployment Toolkit - Systems Management - Wiki - Systems Management - Dell Community." [Online]. Available: http://en.community.dell.com/techcenter/systems-management/w/wiki/1772.dell-openmanage-deployment-toolkit.aspx. [Accessed: 17-Sep-2012].

[10] "Dell PowerEdge C Server | System Management." [Online]. Available: http://poweredgec.com/. [Accessed: 12-Sep-2012].

[11] "Optimal BIOS settings for HPC workloads - High Performance Computing - Blog - High Performance Computing - Dell Community." [Online]. Available:

http://en.community.dell.com/techcenter/high-performance-computing/b/weblog/archive/2012/08/17/optimal-bios-settings-for-hpc-workloads.aspx. [Accessed: 17-Sep-2012].

[12] "How to Enable 'HPC-mode' to Achieve up to 6% Improvement in HPL Efficiency - General HPC - High Performance Computing - Dell Community."[Online]. Available: http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2012/07/05/how-to-enable-quot-hpc-mode-quot-to-achieve-up-to-6-improvement-in-hpl-efficiency.aspx. [Accessed: 17-Sep-2012].

[13] "MATLAB Distributed Computing Server - Supported Schedulers." [Online]. Available: http://www.mathworks.com/products/distriben/supported/index.html. [Accessed: 17-Sep-2012].

[14] "ANSYS High Performance Computing Features." [Online]. Available: http://www.ansys.com/Products/Workflow+Technology/High-Performance+Computing/Features. [Accessed: 17-Sep-2012].

[15] "SystemImager." [Online]. Available: http://systemimager.sourceforge.net/. [Accessed: 17-Sep-2012].

[16] "pdsh - Parallel Distributed Shell - Google Project Hosting." [Online]. Available: https://code.google.com/p/pdsh/. [Accessed: 17-Sep-2012].

[17] "Anaconda/Kickstart - FedoraProject." [Online]. Available: http://fedoraproject.org/wiki/Anaconda/Kickstart. [Accessed: 17-Sep-2012].

[18] "What is Puppet? | Puppet Labs." [Online]. Available: http://puppetlabs.com/puppet/what-is-puppet/. [Accessed: 17-Sep-2012].

[19] L. Kanies, "Puppet: Next-generation configuration management," *The USENIX Magazine. v31 i1*, pp. 19–25, 2006.

[20] "Git - About." [Online]. Available: http://git-scm.com/about. [Accessed: 17-Sep-2012].

[21] "Git Workflow and Puppet Environments | Puppet Labs." [Online]. Available: http://puppetlabs.com/blog/git-workflow-and-puppet-environments/. [Accessed: 17-Sep-2012].

[22] "Cacti® - The Complete RRDTool-based Graphing Solution." [Online]. Available: http://www.cacti.net/. [Accessed: 17-Sep-2012].

[23] "Nagios - The Industry Standard in IT Infrastructure Monitoring." [Online]. Available: http://www.nagios.org/. [Accessed: 17-Sep-2012].

[24] "Ganglia Monitoring System." [Online]. Available: http://ganglia.sourceforge.net/. [Accessed: 17-Sep-2012].

[25] F. D. Sacerdoti, M. J. Katz, M. L. Massie, and D. E. Culler, "Wide area cluster monitoring with ganglia," in *Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on*, 2003, pp. 289–298.

[26] "10.2 Compute Node Health Check." [Online]. Available: http://www.clusterresources.com/torquedocs/10.2healthcheck.shtml. [Accessed: 17-Sep-2012].

[27] "Moab Workload Manager - Object Triggers." [Online]. Available: http://www.adaptivecomputing.com/resources/docs/mwm/19.0triggers.php. [Accessed: 17-Sep-2012].

[28] "xCAT - Extreme Cloud Administration Toolkit." [Online]. Available: http://xcat.sourceforge.net/. [Accessed: 17-Sep-2012].

[29] D. M. Ritchie and K. Thompson, "The UNIX time-sharing system," *Commun. ACM*, vol. 17, no. 7, pp. 365–375, 1974.

[30] "MATLAB Distributed Computing Server - Level of Support for Schedulers." [Online]. Available: http://www.mathworks.com/products/distriben/supported/level-of-support.html#MWjobmanager. [Accessed: 17-Sep-2012].

[31] "Introduction to the Wolfram Lightweight Grid System - Wolfram Mathematica 8 Documentation." [Online]. Available: http://reference.wolfram.com/mathematica/LightweightGridClient/tutorial/Introduction.html. [Accessed: 17-Sep-2012].

[32] "Moab HPC Suite Basic Edition." [Online]. Available: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-basic-edition/. [Accessed: 17-Sep-2012].

[33] "BLCR » FTG." [Online]. Available: https://ftg.lbl.gov/projects/CheckpointRestart/. [Accessed: 17-Sep-2012].

[34] P. H. Hargrove and J. C. Duell, "Berkeley lab checkpoint/restart (blcr) for linux clusters," in *Journal of Physics: Conference Series*, 2006, vol. 46, p. 494.

[35] "New Powertool to help checkpoint jobs - Dirk Joel-Luchini Colbry - HPCC Wiki." [Online]. Available: https://wiki.hpcc.msu.edu/display/~colbrydi@msu.edu/2011/10/06/New+Powertool+to+help+checkpoint+jobs. [Accessed: 17-Sep-2012].

[36] "Moab Workload Manager - Managing Consumable Generic Resources." [Online]. Available: http://www.adaptivecomputing.com/resources/docs/mwm/12.6consumablegres.php. [Accessed: 17-Sep-2012].

[37] "Appendix J: Job Submission Filter (aka 'qsub Wrapper')*."[Online]. Available: http://www.clusterresources.com/torquedocs21/a.jqsubwrapper.shtml. [Accessed: 18-Sep-2012].

[38] "Wiki Front Page - Whamcloud Community Space - Whamcloud Community." [Online]. Available:

http://wiki.whamcloud.com/display/PUB/Wiki+Front+Page. [Accessed: 17-Sep-2012].

[39] P. Schwan, "Lustre: Building a file system for 1000-node clusters," in *Proceedings of the 2003 Linux Symposium*, 2003, vol. 2003.

[40] J. Bonwick and B. Moore, "Zfs: The last word in file systems," *online][retrieved on Jan. 22, 2008] Retrieved from the Internet*, 2007.

[41] "SGI - Developer Central Open Source | XFS." [Online]. Available: http://oss.sgi.com/projects/xfs/. [Accessed: 17-Sep-2012].

[42] "About illumos - illumos - illumos wiki." [Online]. Available: http://wiki.illumos.org/display/illumos/About+illumos. [Accessed: 17-Sep-2012].

[43] "ZFS Dedup FAQ (Community Group zfs.dedup) - XWiki." [Online]. Available: http://hub.opensolaris.org/bin/view/Community+Group+zfs/dedup. [Accessed: 17-Sep-2012].

[44] "The Blog of Ben Rockwood." [Online]. Available: http://cuddletech.com/blog/pivot/entry.php?id=983. [Accessed: 17-Sep-2012].

[45] N. Rasmussen, "Air distribution architecture options for mission critical facilities," *ELEKTRON JOURNAL-SOUTH AFRICAN INSTITUTE OF ELECTRICAL ENGINEERS*, vol. 22, no. 10, p. 68, 2005.

[46] "National Petascale Computing Facility." [Online]. Available: http://www.ncsa.illinois.edu/AboutUs/Facilities/npcf.html. [Accessed: 17-Sep-2012].

[47] "Precision Cooling - High Density Modular Cooling." [Online]. Available: http://www.emersonnetworkpower.com/en-US/Products/PrecisionCooling/HighDensityModularCooling/Refrigerant-Based/Pages/Default.aspx.

[48] "Main Page - SELinux Wiki." [Online]. Available: http://selinuxproject.org/page/Main_Page. [Accessed: 17-Sep-2012].

[49] "Fail2ban." [Online]. Available: http://www.fail2ban.org/wiki/index.php/Main_Page. [Accessed: 17-Sep-2012].

[50] "Red Hat | Red Hat Enterprise Linux for Scientific Computing – HPC." [Online]. Available: http://www.redhat.com/products/enterprise-linux/scientific-computing/. [Accessed: 17-Sep-2012].

[51] "Scientific Linux - Welcome to Scientific Linux (SL)." [Online]. Available: https://www.scientificlinux.org/. [Accessed: 17-Sep-2012].

[52] "Fedora Project - What is Fedora and what makes it different?" [Online]. Available: http://fedoraproject.org/en/about-fedora. [Accessed: 17-Sep-2012].

[53] "Lmod | Free Development software downloads at SourceForge.net." [Online]. Available: http://sourceforge.net/projects/lmod/. [Accessed: 17-Sep-2012].

[54] R. McLay, K. W. Schulz, W. L. Barth, and T. Minyard, "Best practices for the deployment and management of production HPC clusters," in *State of the Practice Reports*, 2011, p. 9.

[55] "Modules -- Software Environment Management." [Online]. Available: http://modules.sourceforge.net/. [Accessed: 17-Sep-2012].

[56] "Modules that load modules segfault when unloading due to invalid memory accesses." [Online]. Available: http://sourceforge.net/mailarchive/forum.php?thread_name=201204201559.01764.twhitehead%40gmail.com&forum_name=modules-interest. [Accessed: 17-Sep-2012].

[57] "The Facter Program Quickly Gathers Basic Node Information | Puppet Labs." [Online]. Available: http://puppetlabs.com/puppet/related-projects/facter/. [Accessed: 17-Sep-2012].

[58] "HPCC@MSU (hpccmsu) on Twitter." [Online]. Available: https://twitter.com/hpccmsu. [Accessed: 17-Sep-2012].

[59] "iCER (icermsu) on Twitter." [Online]. Available: https://twitter.com/icermsu. [Accessed: 17-Sep-2012].

[60] "RT: Request Tracker - Best Practical." [Online]. Available: http://bestpractical.com/rt/. [Accessed: 17-Sep-2012].

[61] S. C. Ahalt, A. Apon, A. H. P. C. C. Director, D. Lifka, and H. Neeman, "Sustainable Funding and Business Models for Academic Cyberinfrastructure Facilities: Workshop Report and Recommendations," in *Workshop Report and Recommendations*, 2010.

[62] "fasrc/module-usage · GitHub." [Online]. Available: https://github.com/fasrc/module-usage. [Accessed: 18-Sep-2012].

[63] "ITIL® Home." [Online]. Available: http://www.itil-officialsite.com/. [Accessed: 18-Sep-2012].