Improving User Notification on Frequently Changing HPC Environments

Chris Fuson, William Renaud, James Wynne III
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
Oak Ridge, TN USA
e-mail: {fusoncb|brenaud|wynnejr} at ornl.gov

***Abstract*—Today's HPC centers' user environments can be very complex. Centers often contain multiple large complicated computational systems each with their own user environment. Changes to a system's environment can be very impactful; however, a center's user environment is, in one-way or another, frequently changing. Because of this, it is vital for centers to notify users of change. For users, untracked changes can be costly, resulting in unnecessary debug time as well as wasting valuable compute allocations and research time. Communicating frequent change to diverse user communities is a common and ongoing task for HPC centers. This paper will cover the OLCF's current processes and methods used to communicate change to users of the center's large Cray systems and supporting resources. The paper will share lessons learned and goals as well as practices, tools, and methods used to continually improve and reach members of the OLCF user community.**

Keywords-user; support; communication; hpc

I. INTRODUCTION

Today's High Performance Computing (HPC) centers' user environments can be very complex. Centers often contain multiple large complicated computational systems connected to multiple task-specific file systems, as well as data analysis, data transfer and workflow resources. Each system also contains its own user environment with libraries, compilers, software packages, environment variables, and batch queuing. Center-specific policies and procedures used to govern resource use can further complicate the user environment.

Users must utilize combinations of hardware and software in order to effectively use HPC resources. Software, library, compiler, and driver versions must fit together like a puzzle. In many cases, changing a single piece of the puzzle can break the process. Changes to a default software package or library can, for example, cause a build process or batch job to fail or produce unexpected results. File system data retention policy changes can result in data loss. Even changes to behind-the-scenes pieces, such as environment variables, can cause users to see failures.

But, a center's resources are frequently changing and evolving. It is not possible for a center or vendor to provide users with an unchanging environment for very long. Requests for additional features, bug fixes, and security patches are just some of the reasons and requirements that constantly drive centers to change user resources. A center's user environment is, in one-way or another, frequently changing.

While testing procedures and environments that hide underlying complexity, such as Cray's programming environment [1], can reduce the impact of change, in many cases, users must take action as the result of change. Change to libraries may, for example, require users to alter function calls, compute system hardware changes may require users to alter batch scripts, and center policy changes may require users to alter workflows.

It is vital for centers to notify users of hardware, software, and policy changes. It is similarly important for users to follow changes that impact their work. For users, untracked changes can be costly, resulting in unnecessary debug time as well as wasting valuable limited compute allocations and research time. Centers should carefully plan user notification schedules and work to provide notifications in a manner that reaches targeted audiences.

The diversity of a center's user population adds complexity to the notification process. A center's user population often contains hundreds to thousands of users with differing backgrounds, experience levels, science domains, schedules, and goals who connect remotely from around the globe and around the clock, *"Fig. 1"*. Often users have access to multiple systems at multiple centers and their use of each is frequently more cyclic than constant.

FIGURE I. OLCF HOURLY LOGINS

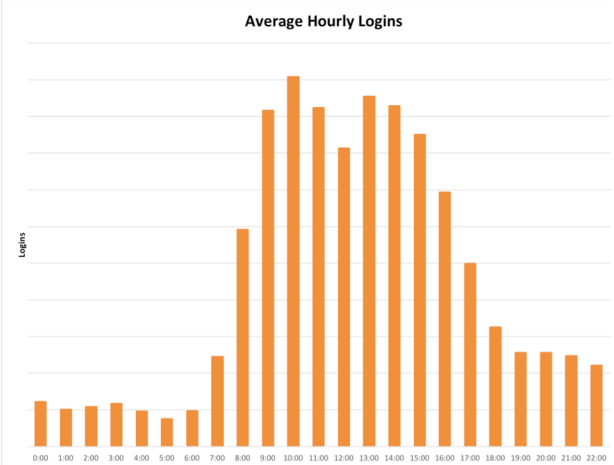

Figure 1. Average hourly OLCF logins over three month period. While peak logins occured between 09:00 and 17:00, approximately 30% of logins occured outside normal business hours.

Because a center's resources must cover a wide range of use cases, change can impact users differently or not at all. Changing a software package, for example, may impact

only a small group of users and not the entire user population. Notifying users of changes that do not impact them is not ideal; sending messages only to targeted user groups who are impacted by the change and not the entire user community is the better option. Notifying only users who are actively using a software package, for example, will reduce unnecessary communications and increase the likelihood that a message will be read.

Communicating frequent change to diverse user communities is a common and ongoing task for HPC centers with large Cray systems. The purpose of this paper is to share previous and ongoing work to improve Oak Ridge Leadership Computing Facility (OLCF) user communication. The paper will share historical communication struggles, lessons learned, and goals as well as practices, tools, and methods used to continually improve and reach members of the OLCF user community.

This paper will cover the OLCF's current processes and methods used to communicate change to users of the center's large Cray systems and supporting resources. Historical processes and lessons learned will be covered to help frame the center's current state. The paper will also discuss notification methods developed over the years and the tools and processes developed.

Existing and ongoing work to integrate notification into the user environment will also be shared. Integrating notices into center-managed modules has, for example, proven to be a successful practice to provide targeted messages to only those using the module. The paper will discuss the work to integrate notification into the center-provided and Cray-provided module environment.

II. Notification Methods

The OLCF utilizes a number of general tools and procedures to notify users of system change and notable events. Many of the methods utilize common communication tools while others are system specific or in-house developed. Some tools are designed to be interruptive while others allow users to view the notification at their convenience. Because users vary in the way they work and interact with the center, the OLCF utilizes multiple techniques and tools to notify users of change. A tool or method is selected based on the notification, the target user group, and the notification's level of importance.

Email is likely the most common method of notification. For most, email is a part of daily life; most are already regularly checking and managing email. Because of this, it makes sense to utilize email as one of the center's main notification methods. Email management systems allow for easy dissemination of information to groups of varying sizes. For example, the GNU Mailman List Manager (Mailman) [2] software provides an easy method to manage email recipients and limit who can send email to the recipient lists. Messages and lists of email addresses can be stored on local servers as opposed to remote servers which allows more control over user email addresses. The software also provides the ability for multiple staff members to send messages on behalf of the center. Sending messages that appear to come from the center, instead of individual staff, help to provide a unified center notification.

While email is a vital information dissemination tool, it is also important to utilize other methods of communication. Web pages, for example, are an effective method to provide large amounts of information to a broad audience. Web sites provide the ability to display large amounts of data in a format that is easy to navigate and search. Because data is displayed from a single source, it also provides a method that can easily be updated without the need to redistribute or update distributed copies of the information. The OLCF utilizes its web site [3] to provide center information from a general overview of the center to detailed processes needed to utilize center resources. The center also utilizes web sites to notify users of change. Other notification methods discussed in this paper are limited in the amount of data that can be provided.

Web sites, login message of the day (motd), Twitter [4], write to all (wall), and wrapping common command line tools provide useful methods to reach users. Passive methods of providing information, such as websites, allow the center to provide information that can be viewed at the user's convenience. Other tools are more interrupt driven and can be used to reach users while they perform a task. For example, by wrapping common UNIX command line tools, messages can be printed to the screen when the user executes the command. Because the OLCF user base varies, it is useful to also vary the methods used to notify users of change users.

III. Targeting Smaller Groups

By emailing all OLCF users with a valid account and placing notices on the center's web site, the center can notify all users of center change. However, not all change impacts all users. In fact, most changes made to center resources likely impact only a subset of the user population. A change to a library, for example, only impacts those who use the library. Notifying all users of all center changes can greatly increase the amount of communication a user receives from the center and can reduce the likelihood that a change notice reaches the impacted users. Sending a notification containing a small number of change items is more effective than sending the same two items mixed into a communication with a larger number of change items. Notification provided to only those impacted by the change can also help to increase the likelihood that a message reaches the target audience. By working to provide messages that target those groups of users impacted by the change, center notifications can become more effective.

A. Wrapping Command Line Tools

Integrating the ability to display messages at execution time into existing tools provides the ability to target only the

groups who are actively using the tool. For example, messages printed during a module load can be used to relay change to only those using the package. This method allows notification to be targeted to only those users performing a specific action and at the time the action is performed. Often a combination of email, web, and command line tool wrapping messages are used to provide notification. In this case, email and web are used to provide a heads-up while command line tools provide a tap on the shoulder notice.

The OLCF wraps qsub, aprun, and environment modules, which are common tools required to utilize the OLCF Cray systems. The methods used to wrap a tool may vary depending on the tool. But, in each case wrapping the tool provides the ability to see arguments passed to the tool and review the arguments before calling the actual tool. Wrapping a tool simply allows the ability to interject checks between the time the user hits enter and the actual tool is executed.

1) Qsub

The TORQUE Resource Manager's [5] qsub utility is a tool used to submit jobs to a system's batch queue. The tool is used on all OLCF systems as the only route to submit jobs to the batch queue. Qsub is wrapped on all OLCF systems to help enforce center policies, provide user-friendly messages, and in some cases to alter the submission. Center batch policies are enforced within the submission wrapper as well as within the batch system. For policies enforced within the batch system, outliers may be handled by placing the job on hold indefinitely or by rejecting the job. When rejected, the system may not reject a batch job for hours or days after the job has been submitted. By wrapping batch submissions, immediate feedback can be provided for cases where the batch system will eventually reject the job. This allows the submitting user the ability to see a message describing the center policy that caused the job to be rejected, make the necessary corrections, and resubmit the batch job without waiting in the queue for some period of time only to have the job move into an indefinite hold state or rejected. The wrapper also provides the ability to print notice messages for all job submissions. This method is normally reserved for very large and impactful notices. By wrapping the qsub submission tool, the center policy notices can be provided to targeted groups of users upon batch job submission

2) Aprun (and ld)

We can provide both users and center staff with a great deal of information by wrapping the commands that users use to both build and execute parallel jobs on the system. The OLCF uses wrappers both to capture information about libraries used by an executable and to provide the user with alerts if the task layout within a job is not optimal.

Tracking loaded and unloaded modules can provide great insight into which software packages are being used, but there is no explicit guarantee that specific modules will be loaded every time the module's associated libraries are used. This is certainly the case with static linking where, once an executable is built, there is no longer a need for the system to find the library. It can be the case as well with shared objects, where the user merely needs to tell the loader where to find relevant libraries [6]. Ideally, the user will load the module to accomplish this, but users may simply opt to set appropriate variables to permit the system to find libraries necessary to resolve all symbols. Therefore, we can't rely solely on information from module actions (loads/unloads) to determine what software is being used.

The Automatic Library Tracking Database (ALTD) has been in use for a number of years at OLCF, as previously reported [7]. Readers desiring a more complete description of ALTD should refer to [8]; however, a brief overview will be provided here. ALTD is implemented by wrapping both the linker (ld) and parallel job launcher. The parallel job launcher on Titan is aprun. At link time, the ld wrapper stores an executable's link line in the database and adds a tag to the executable. When the executable is later launched via aprun, the aprun wrapper reads the tag provided by the ld wrapper and records in the database an instance of the particular executable being run. Center personnel can later query the database to determine a variety of statistics about particular libraries on the system, including how often they're linked against, by whom (both in terms of which users and which projects), and how often these executables are run. An executable's statistics can be of great interest to center staff in gathering information on which packages are being used and how often they are being used. In the case of users who are not loading modules at runtime, it provides center staff with critical information regarding library usage. This provides an additional tool over notifications previously discussed, in that center personnel can more directly query what software is in use so they can build a list for a targeted email regarding a version of software that is being considered for removal. It should be noted that OLCF is currently moving away from ALTD and toward XALT, which is a newer utility providing similar functionality [9][10]. This transition is in its infancy at OLCF, but is expected to be completed in the coming months.

As time goes on, systems become more and more complex, especially from the perspective of the user running a parallel job. Systems with one single-core processor per node have given way to more complex designs such as the Cray XK7, which contains a multi-core processor, and a general-purpose GPU/accelerator per node [11]. As there is no one-size-fits-all model in terms of how codes run, these more complex systems allow the user a great degree of control over the layout of tasks across the compute nodes allocated to a user. Additionally, users can have accounts on multiple systems at multiple sites. Thus, they likely use numerous job launchers and therefore need to remember multiple settings and options for job task layout. Obviously, this kind of system information changes far less often than

system software, but is nonetheless important to communicate to the users.

One consequence of this complexity is the possibility to inadvertently select an inefficient layout for tasks allocated. An example of this is running a hybrid MPI+threaded code but not requesting resources in a manner that allows each thread to be placed on its own core (unless explicitly requested, all threads on the XK7 run on a single core)[12]. It's likely that the job would experience less-than-expected performance, but might not be aware of the root cause.

Another example applies when a job only partially populates a node with MPI tasks (which might be necessary to accommodate tasks using relatively large amounts of memory). The AMD Opteron 6200 series processor is built around compute units that pair integer cores with a single floating-point unit [13]. By default, the system will place tasks in a "first fill" method, that is, it will place tasks on both integer cores of one compute unit before placing tasks on another compute unit [12]. When a user only uses part of the node, it is likely they would prefer to only place one MPI task per floating-point unit to increase floating-point performance. This is not the default behavior, and can be a stumbling block for increasing performance.

While it is important to document these application-scheduling intricacies on user documentation websites and via emails notifications and to discuss them during formal training events, such notifications are not the best way to target individuals. Fortunately, we also have the opportunity to communicate directly by analyzing the aprun job layout request prior to launching the job. OLCF's aprun wrapper calls a tool named *aprun-usage*, which analyzes the given aprun's job layout request to detect potentially inefficient layout options. Feedback is printed to standard error when certain suboptimal job layouts are requested. This feedback is instant if the job is running interactively; for batch jobs submitted using a script the feedback is simply logged to the job's output file. In all cases, the job does continue to run, so the messages are merely informative (to help select a more optimal task layout in future runs) and not an obstruction to their processing.

To handle multiple wrappers for the aprun command, OLCF extended the model used by ALTD in which the initial aprun wrapper called a supplementary script called *aprun-prologue* [8]. The aprun-usage script was installed in its own directory but renamed aprun-prologue. The basic aprun wrapper script provided with ALTD was then extended to call both the ALTD and aprun-usage versions of aprun-prologue and was installed as a separate modulefile. Thus, it now functions as an overall aprun wrapper that can call additional task-specific wrappers. The additional wrappers (that is, the altd and aprun-usage aprun-prologue scripts) are installed in other module files. Additional wrappers, if identified in the future, could potentially be installed in a similar way. Users can control which aprun wrappers are used by unloading the corresponding modules (all are loaded by default).

As noted earlier, OLCF is transitioning from ALTD to XALT. While it is similar in function to ALTD, XALT does not use the same aprun/aprun-prologue calling sequence as ALTD. As an interim measure, the standard aprun wrapper provided with XALT has been modified slightly to call the aprun-usage script. With that configuration, execution of the aprun-usage script can be controlled via an environment variable. Additional work will be required to fully duplicate the model of a single aprun wrapper that can call multiple task-specific wrappers.

3) Modules

The OLCF user base is very diverse containing projects that cover a wide range of science categories. The applications, libraries, and compilers used by projects across science categories, within science categories, and often within a project vary. Because of the diversity of the user base, each OLCF system must provide and support multiple compilers, libraries, and application packages. The center must also provide multiple version of each supported package. Dependencies between compilers, libraries, and application packages must also be taken into consideration. For example, an application may require a particular non-default version of a library, the library may require a specific non-default compiler version, and the compiler may require other non-default versions of multiple libraries. Managing multiple versions of compilers, libraries, and applications as well as the dependencies that connect each is a very large task. To build applications, users must be able to select the needed compiler and library versions as well as the dependent libraries that are often unknown to the user.

Environment modules [14] is a tool to help manage users' shell environments. The tool provides the ability to set and alter environment variables in a user's shell. Setting environment variables, including key variables such as $PATH and $LD_LIBRARY_PATH, provides control over versions without the need to specify installation locations. Users can, for example, execute a module load command providing the package name and version to add the package to their environment. The module will alter any needed variables and will also often add additional modules to the environment based on the package's needs. The OLCF uses environment modules to help provide multiple versions of compilers, libraries, and applications and manage the complex dependencies between each version.

To use environment modules, users call an alias or function (depending on the calling shell) named module. The alias and additional variables needed to use environment modules is set upon login for all users. The module alias calls a binary named modulecmd passing along the user provided arguments. The basic arguments passed to module can include flags to list, load, and unload modules. For example, to add a package to an environment, 'module load package-name/version' could be executed

from a system's command line. The module alias would call modulecmd passing the given arguments. The modulecmd would then use a file specific to the package and version to update the environment. The file used to specify actions is called a modulefile and is written in Tcl (Tool Command Language) [15]. Modulefiles can be used to set new or alter existing variables, load or unload additional modulefiles, or other tasks such as executing a command. Calling the module command with a load will cause the execution of the package's modulefile; calling the module command with an unload will undo the environment settings made during the load.

On Cray provided systems, Cray provides the environment modules package and many modulefiles. The Cray systems utilize a combination of environment modules, compiler wrappers, and login scripts to control versions and hide system complexity. Compiler wrappers and modules work together to provide a simplified and user-friendly environment. A compiler wrapper may key off of environment variables set by a modulefile in order to create a link line that adds needed libraries, headers, and link flags. To use a Cray provided library, a user only needs to load the necessary library's modulefile and build using the compiler wrapper. The compiler wrapper will link to the libraries, headers, and flags that match the calling compiler behind the scenes.

The OLCF also builds and maintains additional software and the software's associated module for center resources including Cray systems. Smithy [16], a software installation management tool, is used to manage center maintained software builds and modulefiles. On Cray systems, the center provided modulefiles exists alongside the Cray provided modulefiles. The same module alias used to load Cray provided modules is used to load center-managed modules. In most cases, general use and listing does not denote the difference.

Throughout a system's lifetime new modules will be added, old modules will be removed, and defaults will change. Because changes to available modules impacts use of the system, users must be notified of all module changes. The process to change defaults and remove old module versions has historically involved notifying users via email as well as adding the change to the website. However, because modulefiles must be loaded before using, they also provide an opportunity to target those using a package. By printing a notification when a module is loaded, change notification can be provided to those using the package. Providing a reminder of change at the point a package is being used has the ability to reach a target audience more effectively than email or other methods.

An initial opportunity to test providing targeted notices through modules came in December 2014 when approximately 180 builds of center provided software packages were scheduled to be removed from Titan [17]. The builds spanned approximately 46 software packages that often each contained multiple versions and multiple compiler specific builds per version. Each of the to-be-removed builds was managed by the center; because of this, center staff had access to the modulefiles. Modulefiles for each of the impacted package versions was updated to call a script. The script printed a notification message to standard error stating the package version was going to be removed from the system on a given future date. As a result of the added module notifications, the center received inquiries that allowed the opportunity to work with individual users prior to the change. In some cases, after working with users, package versions were taken off the list of to be removed versions. Often inquiries noted the module notice message as the reason for the inquiry. Following the change, the number of inquiries was limited and lower than previous software removals. Due to the amount of pre-change inquiries that noted the module messages combined with the limited amount of post-change inquiries, the method was deemed useful and the center proceeded to investigate a long-term implementation.

A long-term implementation plan discussed involved modifying all center-provided modulefiles to call a notification message printing script. The notification script could decide based on the module action, module package, and version if a notice should be printed to standard error before continuing with the module action. Since the software management system used by the center also manages modulefiles, the management system could add a call to the notification message script to each modulefile. This method is a valid option for center provided modules, but not for Cray provided modules.

Because the center uses modulefiles provided by both the center and Cray, a solution that provides the ability to print messages for both center provided modules and Cray provided modules is required. After investigating a number of options, wrapping the modulecmd binary was chosen as the path forward. Wrapping the modulecmd binary is accomplished by renaming the modulecmd binary and creating a script named modulecmd that calls the renamed original modulecmd binary. By wrapping the binary, scripts to parse the module's command line arguments could be inserted into module requests. The scripts would be executed for all module commands regardless of the modulefile location or maintainer. Wrapping the modulecmd binary provides the ability to insert center-managed actions into all module commands without the need to touch individual modulefiles.

To provide a straightforward implementation across all user facing center resources that use modules, the module environment software can be placed in the center's third party software installation directory tree. By relocating the software, shell specific login files on each system could simply source the modules.csh and modules.sh initialization scripts located in the center's third party software

area. Modifications to the module build could then be made without the need to modify system files.

During the development of the environment modules wrapper, it became apparent that having an extensive test suite was very important. If there were a bug in the custom environment modules setup, or in the modules wrapper, it would end up being very impactful on the user's experience and could greatly hinder usability of the system. Therefore, tests would have to be written for each of the functions of the wrapper (user messages, module usage logging, etc), as well as each of the base functions of environment modules itself. Not only would the test suite need to test all these, it would be necessary to test them under all available shells: sh, bash, zsh, csh, ksh, tcsh, etc. Having an extensive test suite helps prevent issues that could take time away from users' developing and running codes which directly translates to the computers being used more efficiently.

The modulecmd wrapper is currently used in production at the OLCF on the center's 744 node Cray XC30, Eos. Recently approximately 100 Cray provided software builds spanning over 25 packages were removed from Eos [18]. The module wrapper was used prior to removing the builds as one method to provide notification of the removal, *"Fig. 2"*. Based on success seen, the plan is to continue expanding the wrapper to other OLCF systems.

FIGURE II. MODULE NOTICE

```
eos-ext1 1003> module load cray-petsc/3.4.2.0

############################################################
##                         NOTICE                         ##
############################################################

The following cray-petsc versions will be removed on
Tuesday, March 29. Please use a more recent version.
  3.4.2.0
  3.4.3.1
  3.4.4.0
  3.5.1.0
If you still require a version from the list,
please contact help@olcf.ornl.gov

############################################################
```

Figure 2. Notitication of package version removal targeting users of the package and printed to standard error upon module load.

B. Email

The OLCF maintains multiple email lists to help target groups of users. Email lists are generally organized by system and storage access. In addition to a list of all enabled users, lists for each of the allocated systems, Lustre [19] filesystems, and High Performance Storage System (HPSS) [20] are maintained. As historic lessons have shown, two groups exist: those who like to receive limited amount of email and those who like receiving large amounts of email. To help address both groups, two email lists per system, filesystem, and HPSS are maintained. The first set of email lists is considered low-volume, mandatory membership lists. The lists contain users who have access to the system. The second set of email lists are considered optional and contain users who are active on the system. Users can opt-out of the lists if they do not wish to be on the list.

By maintaining system specific email lists, system notices can be sent to only those who have access to a system. Additionally, maintaining two email lists per system, a mandatory membership low-volume list containing all users and an optional high-volume list containing active users, provides the ability to target user groups based on system activity. Weekly center update messages and other high-level or high-impact messages can be sent to all center users. Events that impact only users actively using the system can be sent to the high-volume optional list that contains only those actively using the system.

C. Web

The center's main web site [3] is publically accessible and does not require authentication. The site can be organized by content to target groups of viewers by subject category. But, the ability to target the individual viewer is limited because we only know the viewer based on the subject category of the page they are viewing. In addition to the publically available web site, the OLCF also maintains a site, named MyOLCF, whose access is limited to only users on a current OLCF project. Because the site requires authentication, the content can be targeted to the viewer. For example, the site can display only information that pertains to projects to which the viewer has access, *"Fig. 3"*. By providing a web site that requires authentication, the displayed content can be adjusted to target the viewer.

FIGURE III. AUTHENTICATED WEB NOTICE

Figure 3. Notification examples taken from authenticated MyOLCF web site.

D. Write to All (wall)

The UNIX write to all (wall) [21] tool is not new technology, but it is still a very effective tool to send messages to those connected and actively using a system. The wall tool allows the center to print a message to the screen of all who are connected to a system. While email provides the ability for the center to passively send notification of upcoming outages, the wall tool allows the center to send an interruptive reminder to those actively using a system. The tool provides the ability to essentially 'tap users on the shoulder' and remind them that the system will be taken off line within a short period of time. Because the tool is very interruptive, it is used infrequently and normally only to notify of impending system or file system outages, *"Fig. 4"*. Often the center utilizes the tool to notify those actively using a system before a system outage. The center also uses the tool in combination with login messages to notify of impending emergency system component maintenance. The wall tool provides the center the ability to target those connected to and actively using a system.

FIGURE IV. WALL NOTICE

```
Broadcast Message from root@titan-login2
        at 07:43 ...

System will be shutting down at 08:00 for
scheduled maintenance.
```

Figure 4. Wall message providing notification of system outage targeting users connected to and activily using system.

IV. AUTOMATION

Manually updating email address lists can be a time-consuming and error prone operation. Similarly, manually sending notifications of system state change in a timely manner can be a difficult task especially for changes that occur outside normal business hours. Automating list populations can improve notifications by producing accurate email lists with shorter turnaround time and with increased frequency. By automating system state change notifications the time between the state change and the notification can be set to a constant period of time.

A. Enabled Users Mail Lists

The OLCF maintains multiple low volume mandatory membership email lists. Membership to each of the low volume lists is mandatory for all users with current OLCF accounts. The center maintains a center-wide list containing all current users as well as lists for each HPC resource. Each list is updated as accounts are created and disabled. Because the OLCF adds and removes accounts throughout the calendar year, manually adding and removing emails as accounts are created and disabled is an error prone process. Due to the frequency of account creations, account state changes, and the maintenance issues created by manually updating multiple mail lists, each list is updated automatically.

The account creation and disabling process requires a number of steps to be completed across the center on multiple systems. By adding a step to add or remove users from the mail lists to the account process, users can be automatically added or removed from the mail lists each time an account is created or disabled. The GNU Mailman List Manager (Mailman) software [2], managed by ORNL, is used to by the center to send mass email to OLCF user groups. The Mailman tool allows addresses to be added and removed by sending an email to the server. The account process emails the mailman server to update the relevant lists according the account process event.

B. Populate Email Lists with Active Users

In addition to the mandatory membership low volume email lists; the center also maintains a higher volume optional membership email list for each center resource including the HPC systems, the Lustre [19] file systems, and the mass storage system. Each resource list is populated with users who are actively using the resource. For this purpose, an active user is defined as a person who has logged into a system and/or has had a job, in any state, in the system's batch queue within a recent period of time.

Outside of outage periods, users have the ability to log into OLCF resources and submit batch jobs 24 hours a day seven days a week. Because authentications are required to interact with center resources and center policy regulates the amount of time a login session can be inactive, looking at successful login attempts over a recent period of time provides a list of users who are interactively utilizing center resources. However; the batch system will hold jobs until resources are free and center policies are met. Because of this, it is possible that a batch job will not run during the window of time used to monitor system login sessions. Center policy and resource availability may cause a batch job to wait in the queue for a longer period of time than the window used to gather active user logins. A batch job in the queue, even if the job was not submitted recently, is likely still impacted by system changes. Because of this, users who have had a batch job in the queue within a recent period of time are also considered an active user.

The center uses a combination of successful authentications and batch jobs to capture active users. Because users frequently log into and submit batch jobs to center resources, calculating the active user list manually is not feasible. To automate the process, a tool to gather authentications and batch jobs, and then update mail lists was created. The tool runs daily to gather user IDs from logs and then uses the IDs to retrieve email addresses and update mailman lists.

Each authentication to center resources creates a detailed record in the center's logging system. Because successful authentications are used for a number of activities, the logs are parsed daily to create simplified authentication

records. The active user-gathering tool uses the simplified authentication records to quickly retrieve successful authentications that occurred within a recent period of time.

Snapshots of each user facing resource's batch system are stored on 15-minute intervals. The snapshots provide a list of jobs in the batch system queue in all states at the instance the snapshot was taken. A number of activities use the queue snapshots to view the state of the batch queues over time. The active user-gathering tool utilizes snapshots taken within the defined active user window to retrieve users with batch jobs in the queue during the timeframe.

Once system specific lists of active users are retrieved from the authentication logs and batch queue snapshots, the active user-gathering tool retrieves email addresses for each user ID. Email addresses and user IDs are stored and maintained for each user in a center database. The database is used to map needed user IDs to up-to-date email addresses. Once a list of email addresses is retrieved, the tool then communicates to the Mailman [2] server, via email, addresses that should be removed and added. Because interactions with the mailman server are performed through a GUI or via email, list memberships are maintained locally to simplify the automation process. The tool maintains a per system membership list containing a email address; the list matches the system's live Mailman list. Each time the tool is executed the newly calculated list of addresses is compared against the local copy of the live Mailman lists. The comparison provides addresses that should be removed and added to and from the live Mailman lists. Using the calculated add/remove address list, the tool updates the Mailman server via email and updates the local copies of the lists. The local copies of the lists are considered definitive. Because the Mailman server can be updated outside the active user-gathering tool, each Mailman list is regularly updated to match the local copy.

C. *System Status Notifications*

One of the items of greatest interest to any center's user base is the current state of the center's resources. Ideally, system status changes are infrequent and limited to previously announced outages. Unfortunately, unplanned outages are inevitable for a variety of factors. While not a change in the sense of software update or other configuration change, such events nonetheless are important issues to communicate to a center's user base. Such information can not only alert a user that a running job may potentially need to be restarted or resubmitted, it can also alert the user that a given resource is temporarily unavailable, permitting the user to target other systems to make the most efficient use of their time.

In general, it is likely that a login attempt will occur first versus polling an information source to see if a system is known to be down. Such an event can distract the user from their computing task and instead direct their attention towards finding more information about the current state of the system. This could be an email or phone call to the center's user support group, but that may not always be the best option, especially during "off hours" such as overnight and weekends when the full support staff may not be readily available and responses delayed. Even if such a request was made at a time when the center was fully staffed, it could be mixed with numerous other user requests, which could delay a response to the user.

Timely notification regarding system outages, therefore, remains critical. Large computing centers often have users working at all hours of the day thus timely notification regarding system outages remains critical at all times. And much like user activity, system outages do not adhere to the same workday as system administrators and support staff and even when they do, communication between individual users and the center staff is an inefficient way to pass this information. A method of notifying all potentially impacted users at once is therefore ideal. At the beginning of an outage, however, system administrators are likely working to bring the systems back online. Other center staff may not be immediately aware of the outage.

Centers must find a way to notify users in a timely manner at all hours. The solution as implemented at OLCF is to automate user notifications of system unavailability. The system in use remains largely unchanged from that described previously for both the National Oceanic and Atmosphere Administration's National Climate-Computing Research Center [22] and the OLCF [23], which is based on parsing logs from the center's Nagios [24] instance and making an educated guess of the system status based on the results of the various tests run by Nagios. These status decisions can then be used to provide user notifications via websites *"Fig. 5"*, emails to the high-volume lists discussed earlier, or other utilities [23].

FIGURE V. SYSTEM STATUS INDICATORS

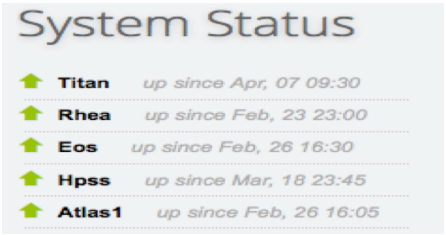

Figure 5. Web system status indicators updated automatically.

As discussed in [22], the status script does have the potential for false positives (which are often corrected during the next iteration of the status script), which are deemed 'flip-flops'. Any notifications must take this possibility into account. Notifications via websites can simply be updated, but email notifications require more caution. It is better to send an email a little late than to send an incorrect one. At other times, the script may detect (and continue to detect) that a system is unavailable when in reality all is well. To combat this, administrators and

support staff have a window of time during which they can cancel a message. For this reason and the potential for a flip-flop, the sending of any user notification from the script is delayed for a set amount of time. If either the state that resulted in the message being created turns out to be a flip-flop or center staff determine the message to be in error, the message is discarded.

Currently, the cancellation of a message does that and nothing else. The script continues to "think" that the system is in the state that was deemed errant, and notifications via the website [25] and Twitter [4] continue to show the errant state. Correction of this is a potential future enhancement; however, the major focus has been on the correctness of email notifications since those are expected to be the most visible to the user.

In the time that has passed since the aforementioned publications, the system has been modestly enhanced. The most notable such change is inclusion of not only Nagios data but also additional data sources such as the center's Splunk [26] infrastructure to provide additional information. Various statistics regarding OLCF Lustre [19] filesystems are stored via Splunk, and a subset of those statistics are queried to provide better information on the status of the center's Lustre filesystems. At present, the script does not use Splunk data in determining the status of any other system. Should other systems begin storing information pertinent to the status script in Splunk, it would be a trivial task to incorporate a query of that information into the status script.

Another such change was to reduce potentially confusing notification messages. Note again that there is the potential for center staff to cancel a message that the script has generated. When this is done, there is the potential for the script to continue to think the system is unavailable. The potential confusion occurs when the script detects that the system has returned to service. Without intervention to cancel the resulting message that's generated, users will be notified that the system has returned to service (even though it was never down). Recently logic was added to the script to prevent this from happening. The script retains a database of sent messages (including the system state that generated the message), and it will not send a message for a system if the state for that message matches the state of the last sent message for that system.

A final update improved system state notification. Previously, states were simply returned as "up" or "down"; however, the recent additions noted in the previous paragraph have added a "degraded" state, informing the user that the system is available but may have performance limited in some way, such as slower than usual response times. Logic in earlier versions of the script was rudimentary and assumed binary states, so the addition of a third state required changes to several parts of the code. The end result is better communication of the system's state, which is an important improvement.

V. Lessons Learned

Over the years, the center's goal to communicate change has remained consistent; however, methods used to communicate change have varied. In many cases, user feedback drives methods used to communicate change. In other cases, trial and error exposes issues with a communication method or points to new methods.

A. Email Size

Large emails can unintentionally hide useful information. A five-page email, for example, may only be perused or set aside to be read at a less busy time. Topics placed in a short concise email are more likely to reach the target audience. Email format is also important. For longer emails, adding a bulleted lists to the top of the email, *"Fig. 6"*, can help the audience see at a glance areas that may impact their work. Brief descriptions of each topic placed in a section below the bulleted list provide more information for topics of interest.

FIGURE VI. Email Bulleted Menu

```
From: OLCF Announcements
Reply-To: "help@olcf.ornl.gov"
Date: Wednesday, March 23, 2016 at 11:26 AM
To: ccs-announce
Subject: OLCF: Weekly Update, March 23, 2016

*** IN THIS MESSAGE ***
* Center Announcements
  - Gordon Bell Award Nominations (due Apr 15)
  - Allinea Forge (MAP + DDT) Now Available
  - Removal of Old Software Versions on Eos (Mar 29)

* Meetings & Workshops
  - INCITE Proposal Writing Webinar (Apr 13 & May 19)
  - 2016 User Meeting: Registration is Open
  - 2016 GPU Hackathons
  - March OLCF User Conference Call (Mar 23)

* Scheduled Outages Through April 1 (none)
```

Figure 6. Bulleted menu from OLCF weekly update email.

For topics that require more than a few sentences, it is useful to place only a brief description of the topic in the email and place in depth information on the center's web site. The email's brief description can contain a link to the topic's more in depth web page allowing interested readers the ability quickly find more information on the topic. Formatting and presentation options provided on the center's web site provides a more favorable location to display large amounts of detailed information. Center resource and infrastructure changes can, for example, require pages of in depth information to relay requirements and impact. By emailing the top need-to-know topics regarding the change in a concise bulleted format with links to a web page containing detailed descriptions of each topic, an overview of the change can be quickly viewed by the reader. For topics of interest, the reader can follow provided links to the topic's section on the center's web site, which contains a more detailed description of the topic.

B. Email Volume

The user community is diverse containing a wide variety of center interaction workflows. History has shown that some workflows work best when most notifications are received via email, others may prefer limited amounts of email opting instead to retrieve the bulk of center notifications from the web or other methods. It was difficult to address both limited and larger volume email notices with a single email list. With a single list, the center was either always sending too many emails or too few emails. To help address the issue, the center now maintains two sets of email lists, a high-volume optional membership list and a low-volume mandatory membership list. The low-volume lists contain all enabled users; users do not have the option to remove their address from the list. Under normal circumstances, an email is sent to the list once a week. The high-volume list contains users who are actively using center resources. Access to the list is optional allowing users to remove their address from the list. Higher volumes of notices are sent to the list; for example, automated system status change notices are sent to the list each time a system changes state. Maintaining high and low volume email lists provides the center with the ability to address multiple email volume preferences.

C. Multiple Communication Channels

Because the OLCF user base varies, a single form of communication is not ideal for all users. While email may be the preferred method to reach some, it may not be the best method to reach others. It is important to maintain multiple notification avenues. Placing a change notification in an email, on the web, presented at monthly user calls, and echoed at module load time, increases the avenues to which a message can be viewed. Providing the same message in multiple locations increases the odds that the message will reach its intended audience.

D. Changing Opt-In to Opt-Out

The OLCF regularly utilize email lists as one method to notify users of system change and center events. Utilizing multiple lists allows the ability to notify targeted groups of users and improve email effectiveness. For example, the OLCF utilizes a set of system-specific email lists to provide automated notification of system state change and other notable system events. Because the lists are higher volume than our standard email notifications, membership to the lists is optional. Historically users had to sign-up to join the lists, but after evaluating the opt-in process, it was determined that an opt-out method would be more beneficial to the user community. In 2015, the OLCF developed a process to automate populating each system's high volume mail list with users who are actively using the system. An active user is defined as one who connects to a system or utilizes the system's batch queue within a recent period of time. Maintaining an email list of active users prevents the need for users to sign-up and provides the center with the ability to email only users actively utilizing the resource without sending unnecessary emails to users who are not currently utilizing the center's resources.

E. Placeholder Modulefiles

Occasionally, modules are removed or change names requiring users to find other packages to complete tasks previously provided by the removed or altered module package. Unlike default version updates, package name changes require loading a new module to utilize a tool or library previously loaded through a module of another name. Removing modules, especially modules that are regularly used, can cause confusion. In addition to notification of the change through standard processes, it is also useful to create a placeholder modulefile to note the change. The placeholder modulefile has the same name as the removed modulefile and can be used to print a message noting the change. In addition to printing a message, the placeholder modulefile can also load the new modulefile. When a module must be removed or renamed, creating a module with the same name provides the ability to give the calling user a notice of the change and helps to reduce confusion caused by the change.

FIGURE VII. MODULE PLACEHOLDER

```
titan-ext3 1315> module load hdf5

************************************************************
***
**     Cray Provided HDF5 Module Naming Convention Change
***
************************************************************
*
* Cray hdf5 modules now prepends 'cray-' to the module name.
*
*        Old           ->            New
* ------------------       ----------------------
*  module load hdf5    ->   module load cray-hdf5
*  module avail hdf5   ->   module avail cray-hdf5
*
* To help in the transition from hdf5 to cray-hdf5
*  this temporary module will load the default cray-hdf5
*  module.  You should begin using the new cray-hdf5
*  naming convention to use Cray's hdf5.
*
************************************************************

titan-ext3 1316>
```

Figure 7. Notice of module naming convention change targeting users of the module and printed to standard error upon module load.

F. Software Change Table

Changing defaults and removing old software versions are required tasks to provide an up-to-date software environment. Often, changes to software default versions and removal of old software is performed for a large number of packages. A change to a library may, for example, require a large number of Cray and center provided packages as well as user maintained applications to be rebuilt. In some cases, a new version of the package may also be required because only new versions of the package support the new library. In other cases, the center may take

the rebuild requirement as an opportunity to move package defaults forward. Old software versions are also often removed in large bundles. To provide user notification, software default changes and removal are appended to a web page on the externally facing web site [27]. Notification of the change sent via email and other methods point to the software change web page. As opposed to previous methods that included all software change in email notification, this method allows notification of the change to be concise. The web page provides a format better suited than email for displaying large amounts of data. Unlike email, maintaining a web site with software changes over a recent six to twelve month period also provides the ability to view and search a historical timeline of change in one location. The ability to view historical software changes is very useful in the debug processes and can help reduce time spent debugging an issue.

VI. Conclusions

HPC systems are very complex and frequently changing in one way or another. Change is often required, but can be very impactful to users of a system. To reduce the impact of change, notification of change must be provided to the system's user community in a manner that reaches all impacted by the change. The diversity of user communities and complexity of HPC systems often complicates change notification. For example, not all change impacts all users and center use is often more cyclic than constant. Because of this, notifying the entire user community of all change unnecessarily increases the amount of communication sent from the center and reduces the odds that a notification will reach the target audience impacted by the change. Employing multiple communication avenues and tools improves the odds that notifications will reach target audiences. Utilizing methods and creating tools that target small groups of users improves the impact of the notice by helping to ensure the notice is relevant to the user's work. Notifications given while an action or tool is being executed provides the opportunity for a "tap on the shoulder" reminder of change. It is also important to regularly evaluate new tools and methods and alter notification methods based on user feedback. Communicating frequent change to diverse user communities will likely continue to be a common and ongoing task for HPC centers with large Cray systems; because of this, it is important for centers to continue to evolve and improve change notification procedures and methods.

Acknowledgment

The authors would like to thank Cathy Willis for her assistance implementing the module wrapper, Ross Miller for his work on the script to query Splunk for Lustre data, Scott Atchley for his work creating the aprun layout tool, and Robert French for his work on module testing. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Cray Programming Environment, http://docs.cray.com/books/S-2529-116/S-2529-116.pdf

[2] GNU Mailing List Manager, https://www.gnu.org/software/mailman/

[3] Oak Ridge Leadership Computing Facility, http://olcf.ornl.gov

[4] Twitter, https://twitter.com

[5] Adapitve TORQUE Resource Manager, http://www.adaptivecomputing.com/products/open-source/torque-resource-manager/

[6] D. Wheeler, Program Library HOWTO, version 1.36, 15 May 2010, http://www.dwheeler.com/program-library/Program-Library-HOWTO.pdf

[7] B Hadri, M Fahey, T Robinson, W Renaud, Software Usage on Cray Systems across Three Centers (NICS, ORNL and CSCS), Proceedings of the Cray User Group Conference (CUG 2012)

[8] M. Faney, N. Jones, B. Hadri, The Automatic Library Tracking Database, Conference: Cray User Group 2010, Edinburgh, United Kingdom

[9] M. Fahey, R. McLay, K. Agrawal, XALT Users Manual, version 0.5, retrieved from https://github.com/Fahey-McLay/xalt/blob/master/doc/XALTUsersManual-0.5.pdf

[10] M. Fahey, R. McLay, K. Agrawal, XALT Design and Installation Manual, version 0.5, retrieved from https://github.com/Fahey-McLay/xalt/blob/master/doc/XALTDesignandInstallationManual-0.5.pdf

[11] Cray, Inc. Cray XK7 Brochure, http://www.cray.com/Assets/PDF/products/xk/CrayXK7Brochure.pdf

[12] aprun man page, http://docs.cray.com/cgi-bin/craydoc.cgi?mode=Show;q=;f=man/alpsm/52/cat1/aprun.1.html

[13] J. Larkin, Titan Architecture, presentation from Titan Users and Developers Workshop (East Coast) and Users Meeting, https://www.olcf.ornl.gov/wp-content/uploads/2013/02/Titan_Architecture_1-JL.pdf

[14] Environment Modules, http://modules.sourceforge.net/

[15] Tcl (Tool Command Language), https://www.tcl.tk/

[16] Smithy, http://anthonydigirolamo.github.io/smithy/

[17] Titan software removal, https://www.olcf.ornl.gov/kb_articles/software-news/#Titan:_Software_Removal_Dec_09_2014

[18] Eos software removal, https://www.olcf.ornl.gov/kb_articles/software-news/#Eos:_Cray_Provided_Software_Removal_March_29_2016

[19] Lustre. http://www.lustre.org

[20] HPSS (High Performance Storage System), http://www.hpss-collaboration.org/

[21] Wall (Write to All), http://www.unix.com/man-page/suse/1/wall/

[22] A. Carlyle, R. Miller, D. Leverman, W. Renaud, and D. Maxwell, "Practical support solutions for a workflow-oriented cray environment" in CUG Conference Proceedings, Stuttgart, Germany, May 2012

[23] A. Carlyle, R. French, W. Renaud, Designing Service-Oriented Tools for HPC Account Management and Reporting, CUG Conference Proceedings, Lugano, Switzerland, May 2014

[24] Nagios, http://www.nagios.org

[25] OLCF Website, Top-level User Support Page, https://www.olcf.ornl.gov/support

[26] Splunk. http://www.splunk.com

[27] OLCF Software News, https://www.olcf.ornl.gov/kb_articles/software-news