Version Control Graphical Interface for Open OnDemand

Huan Chen
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio
hxc556@case.edu

Chris Fietkiewicz
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio
cxf47@case.edu

ABSTRACT

Use of high performance computing (HPC) clusters is challenging for the average researcher who is not familiar with the necessary tools such as the Linux operating system and job management software. Of those using HPC systems, the majority are not trained specifically in computer programming. Nearly 70% of users on the NSF-funded XSEDE system are in fields other than computer science and computing science. The graphical user interfaces (GUIs) that most users rely on for common software on personal computers are typically not available on HPCs. The Ohio Supercomputer Center has addressed these issues through the development of an open source, web-based GUI called Open OnDemand. Because it is open source, administrators are free to deploy it on their own systems, and developers are free to enhance it. To improve workforce development and diversity, we sought to: (1) install Open OnDemand on a private cluster and (2) develop a custom add-on module for version control of HPC application software. We successfully installed and configured Open OnDemand for a private cluster at Case Western Reserve University in order to evaluate the difficulty level of deployment. Despite our lack of experience in HPC system administration, the installation process was straight forward due to a streamlined installer package and thorough documentation. In order to evaluate the extensibility of Open OnDemand, we developed a custom web-based interface for use of the popular Git version control system. Version control tools are important in maintaining software by easily tracking and accessing file changes in both single- and multi-developer projects. Our module successfully integrates with the existing Open OnDemand interface and provides common version control operations that can be used during typical HPC workflows. It is our hope that making version control software easier to use will encourage HPC users to adopt the use of version control into their own workflows to improve productivity and repeatability. Source code for the app will be made available on the author's github site.

CCS CONCEPTS

• **Human-centered computing** → Accessibility systems and tools

KEYWORDS

high performance computing, graphical user interface, version control

ACM Reference format:

H. Chen and C. Fietkiewicz. 2018. SIG Proceedings Paper in word Format. In *Proceedings of ACM PEARC conference, Pittsburgh, Pennsylvania USA, July 2018 (PEARC'18)*, 4 pages. https://doi.org/10.1145/3219104.3229268

1 INTRODUCTION

The use of high performance computing (HPC) clusters in various fields of research involves a user experience that is drastically different from what many researchers are accustomed to with personal computers. The current era of desktop computing and graphical user interfaces (GUIs) has allowed researchers to avoid the limitations that were imposed in the previous era of mainframe computing. The increased use of cloud-based resources has, perhaps, increased researchers' familiarity with several important concepts in remote computing, such as servers, clients, and shared file systems. However, HPC systems typically are not accessible through the types of GUIs that are common with cloud-based tools. By looking at allocations for the eXtreme Science and Engineering Discovery Environment (XSEDE), we observed that nearly 70% are in primary fields other than computer science and computing science. As a result, most users likely have no previous experience with typical HPC interface tools such as Linux scripting and command line terminals.

One example of addressing the above issues is the Open OnDemand project by the Ohio Supercomputing Center [1, 2]. Open OnDemand (OOD) is open source and extensible. We had previous experience using an initial offering of OOD as users of the Ohio Supercomputing Center. We sought to evaluate OOD for use on the university HPC cluster at Case Western University. Our evaluation was intended to address two aspects of HPC software development. First, we would evaluate the process of deploying OOD on an existing HPC system. Second, we would develop an integrated GUI application to evaluate the extensibility of OOD.

The focus of the integrated GUI was chosen to be a version control manager for Git [3]. Version control is a software development tool that tracks file changes and provides controlled access to previous versions of the files in a project. The advantages of using version control in research computing and HPC include file documentation, reproducibility, and coordination between multiple researchers. We chose version control as an application in order to promote it to more researchers and because existing GUIs for version control cannot be directly integrated into HPC management software such as OOD.

2 METHODS

OOD [4] was installed using a virtual cluster created on the High Performance Computing cluster in the Core Facility for Advanced Research Computing at Case Western Reserve University. The virtual cluster was created by the system administrator and consisted of three virtual machines, including a login node and two compute nodes, all of which were running Red Hat Enterprise Linux 7. SLURM job management software [5] was installed by the system administrator. Initially, we installed OOD v1.1 using the documentation provided with that release. Development of the app coincided with updates to OOD v1.2 and v1.3. The virtual cluster was located behind the firewall of the primary, physical HPC system. Therefore, access to the OOD web server required an initial login to a head node on the primary system. Separate user accounts were created on the virtual cluster. The web server on the virtual cluster was accessed using Firefox v52.7.2 running on the head node of the primary system. The web browser was operated through a remote desktop using X2Go [6].

3 RESULTS

3.1 Deployment of Open OnDemand

Our first objective was to evaluate the process of deploying OOD. In the present study, we summarize key aspects of the process. The following third party software was installed using Software Collections and the Linux "scl" utility (see documentation [7] for links and version numbers): Apache, NGINX, Phusion Passenger, Ruby, Node.js, and Git. Note that Git is also required for our custom app, described later. OOD v1.3 was installed as an RPM package. Job management software (e.g. SLURM) was installed separately. Following the software installation, numerous configurations tasks were required regarding security, launching the web server, and configuration files for use with job management software, as described in the documentation [7].

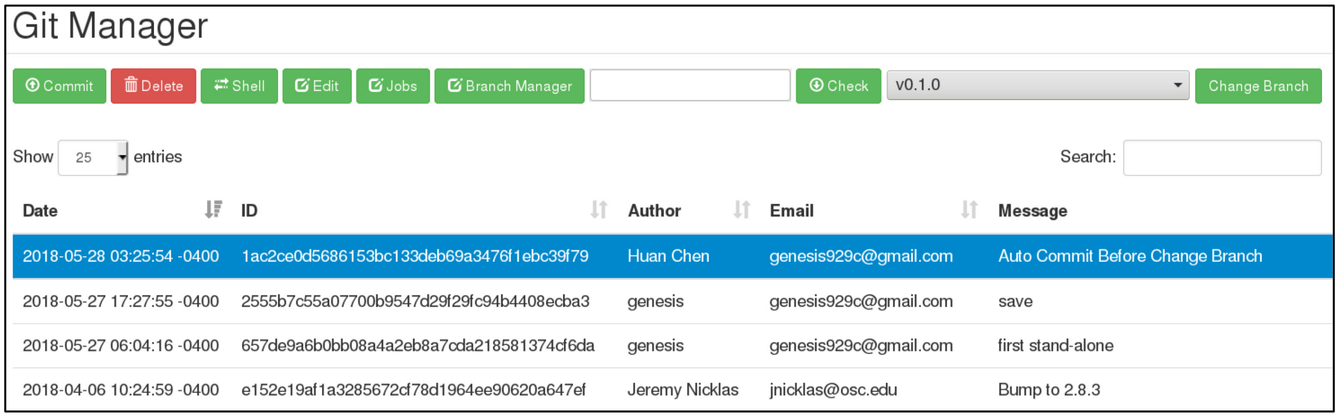
We make several observations regarding the process and considerations for other users. We found the instructions to be thorough and accurate. Note that the documentation includes details specific to CentOS, which is of importance to those who require an open source OS. Though the OOD installation is highly streamlined, the installation of job management software, not surprisingly, requires separate expertise and was performed by the system administrator. However, the configuration tasks are well documented and were performed by the authors. Note that a typical deployment would probably require additional setup tasks such as for interactive desktop support, standalone applications, and authentication. While we did not perform these additional tasks, they are extensively documented [7].

3.2 Custom Version Control App

OOD supports two categories of apps that are referred to in the documentation as "Passenger" and "interactive" apps, respectively. A Passenger app is accessible solely through the OOD interface and is executed through the Phusion Passenger server software. In contrast, an interactive app is executed as an interactive batch job running a separate web server. Our custom version control app is of the Passenger type.

Our app exchanges information with Git version control software and was developed using the Ruby on Rails framework. Our app requires OOD in order to operate, but users are not restricted to using our app exclusively because Git itself is functional without OOD. The intent is to make common tasks convenient and inviting by having Git controls that exist side-by-side with standard OOD controls. All Git operations are performed through independent Linux shell scripts that are launched through a Rails controller.

For readers who are unfamiliar with version control software, introductory tutorials are available for various levels of experience [8]. Here we provide a minimal vocabulary. A "repository" is a predefined collection of files that are under version control. When a repository is first created, an initial snapshot of the state of all the files is saved. At any time after files are modified, a "commit" operation can be performed in which



Date	ID	Author	Email	Message
2018-05-28 03:25:54 -0400	1ac2ce0d5686153bc133deb69a3476f1ebc39f79	Huan Chen	genesis929c@gmail.com	Auto Commit Before Change Branch
2018-05-27 17:27:55 -0400	2555b7c55a07700b9547d29f29fc94b4408ecba3	genesis	genesis929c@gmail.com	save
2018-05-27 06:04:16 -0400	657de9a6b0bb08a4a2eb8a7cda218581374cf6da	genesis	genesis929c@gmail.com	first stand-alone
2018-04-06 10:24:59 -0400	e152e19af1a3285672cf78d1964ee90620a647ef	Jeremy Nicklas	jnicklas@osc.edu	Bump to 2.8.3

Figure 1: Git Manager view. Includes features for common version control operations.

several pieces of information are saved, including any changes to the files, the time at which the commit was performed, and an optional, typed message for recording pertinent comments. Within a single repository, multiple "branches" can be created, where each branch is a separate chronological series of commits such that all branches have a common ancestry.

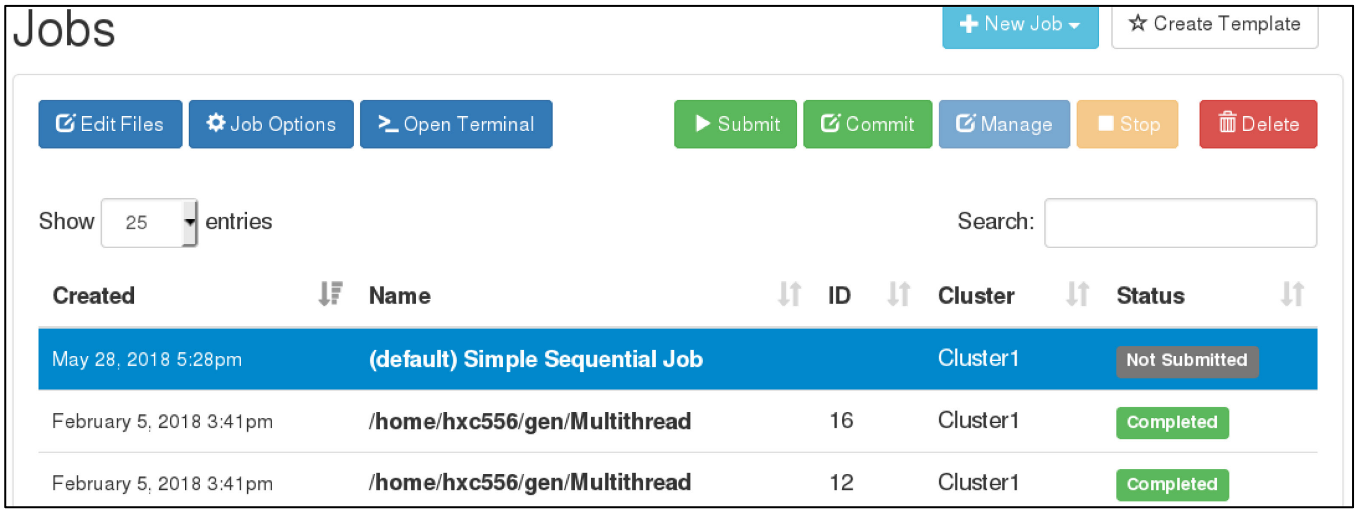
As of this writing, the app has three different page views: Git Manager, Jobs, and Branch Manager. When navigating to any of these views for the first time, the user is asked for a path to a Git repository. If the given path is not already configured as a repository, an option is given to do so. Generally, the Git Manager will be the first view that is accessed from another OOD page.

Figure 1 shows an example of the Git Manager view which is intended to provide controls typically found in other version control GUIs. This view includes a reverse chronological table showing each commit with the date, unique ID, author, email address, and associated message. Above the table are various buttons for Git and OOD operations. The Commit button causes the user to be prompted for a text message before performing a commit operation. The Delete button will undo any changes to files since the last Commit. The Shell and Edit buttons open standard OOD apps. The Shell button opens a terminal window using the repository path as the working directory. The Edit button opens a separate file browser app for the repository path. The Jobs and Branch Manager buttons open our custom Jobs and Branch Manager views, respectively (to be discussed below). The Check button, located next to a text entry box, allows the user to enter the partial ID of any Commit before reverting the files to that state. The Change Branch button and associated drop-down menu allow the user to switch between different branches.

The Jobs view is reachable from the Git Manager view and is nearly identical to a standard OOD app called the Jobs Composer. Figure 2 shows an example of the Jobs view. Most features in the Jobs view are identical to the standard OOD Jobs Composer app and will be described briefly below. Our app adds two unique controls: Commit and Manage. The Commit button results in the user being prompted for a Commit message before performing a Commit operation. We consider this to be a very convenient feature because the execution of a batch job is often a moment at which a user has either made a significant revision or has obtained significant results. The Manage button opens the Git Manager view that was discussed above.

The remaining features in the Jobs view are explained in detail in the OOD documentation [7] and are only briefly described here. They are: New Job creates a new batch job; Create Template creates a job template; Edit Files opens a file browser; Job Options defines specific aspects of the batch job; Open Terminal opens a terminal window; Submit executes batch jobs; Stop stops a batch job; and Delete deletes a batch job.

The Branch Manager view provides features that are common for users who work with multiple branches. Figure 3 shows an example of the Branch Manager view. On the left side, two drop-down menus allow the user to select two branches by name. The Compare button produces a new page containing a comparison between the branches for each file in the repository (not shown). The Merge button attempts to merge modifications from the second branch into the first branch, assuming there are no conflicts. In the case of a merge conflict, the Git error message is displayed, and no other action is taken. Note that merging branches is an advanced version control technique that may require a standard Git interface. Lastly, Create New Branch creates a new branch originating from the master branch.



Created	Name	ID	Cluster	Status
May 28, 2018 5:28pm	(default) Simple Sequential Job		Cluster1	Not Submitted
February 5, 2018 3:41pm	/home/hxc556/gen/Multithread	16	Cluster1	Completed
February 5, 2018 3:41pm	/home/hxc556/gen/Multithread	12	Cluster1	Completed

Figure 2: Jobs view. Includes batch job features standard to OOD plus two features unique to our app: Commit and Manage.

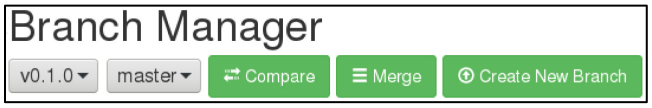


Figure 3: Branch Manager view main controls. Includes common features for branch operations.

4 DISCUSSION

Having been personally trained on the use of HPC systems through traditional methods, and having subsequently trained others for the same methods, we feel OOD represents an important tool in opening HPC to a wider community. In order for OOD and GUI development in general to be successful, open source tools must be tested and extended by the community. The present study has contributed toward both tasks. It is noteworthy that the researchers in this study did not have experience as system administrators, particularly with regard to a job management system on an HPC cluster. Our lack of experience highlights how OOD has been developed to include a streamlined installation process and excellent documentation.

Our custom version control app also served as a means of testing the OOD framework with regard to extensibility. Not surprisingly, the development of the app depended significantly on our own previous experience with web app development, including experience with Ruby on Rails. However, many excellent resources exist for researchers to learn web development skills. Additionally, OOD includes tools, examples, and documentation to assist developers (see the online documentation for details [7]).

In choosing version control to be the focus of our app development exercise, we hoped to address a lack of awareness among researchers regarding this important software development tool. As HPC methods evolve to be more accessible, opportunities exist to promote good practices such as version control. We have presented a work in progress, and we intend to make a future version available to the public at the author's github site.

ACKNOWLEDGMENTS

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. We thank Hadrian Djohari, Emily Dragowsky, and the other HPC staff for their assistance.

REFERENCES

[1] John Russell. 2018. Democratizing HPC: OSC Releases Version 1.3 of OnDemand. *HPCwire*.

[2] David Hudak, Thomas Bitterman, Patricia Carey, Douglas Johnson, Eric Franz, Shaun Brady, Piyush Diwan. 2013. OSC OnDemand: A Web Platform Integrating Access to HPC Systems, Web and VNC Applications, in: *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, XSEDE '13*. ACM, New York, NY, USA, pp. 49:1–49:6. https://doi.org/10.1145/2484762.2484780.

[3] Git: 2018. https://git-scm.com. Accessed: 2018-06-01.

[4] GitHub - OSC/Open-OnDemand: 2018. https://github.com/OSC/Open-OnDemand. Accessed: 2018-06-01.

[5] Slurm Workload Manager: 2018. https://slurm.schedmd.com. Accessed: 2018-06-01.

[6] X2Go - everywhere@home: 2018. https://wiki.x2go.org. Accessed: 2018-06-01.

[7] Open OnDemand: 2018. https://osc.github.io/ood-documentation/master/. Accessed: 2018-06-01.

[8] Git - Documentation: 2018. https://git-scm.com/doc. Accessed: 2018-06-01.