Virtual Clusters in the Jetstream Cloud: A story of elasticized HPC

J. Eric Coulter
jecoulte@iu.edu
Indiana University
Bloomington, IN

Eroma Abeysinghe
eabeysin@iu.edu
Indiana University
Bloomington, IN

Sudhakar Pamidighantam
pamidigs@iu.edu
Indiana University
Bloomington, IN

Marlon Pierce
marpierc@iu.edu
Indiana University
Bloomington, IN

ABSTRACT

We discuss our work providing resources for batch computing via the Jetstream cloud, in the form of SLURM clusters. While these are mainly used by science gateways, there have been a few used in the more traditional commandline manner. The flexible nature of these has also lent itself well to educational work, and has provided the basis for a very successful series of tutorials and workshops. This paper discusses the technical evolution of the Virtual Cluster product, and gives an overview of the science enabled. We discuss the challenges in supporting an ecosystem of these virtual clusters, and in supporting research on cloud resources in general.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Information systems** → *Computing platforms*; • **Software and its engineering** → *Virtual machines*; *System administration*; • **General and reference** → General conference proceedings.

KEYWORDS

HPC, OpenStack, Batch computing, Clusters, Cloud

ACM Reference Format:
J. Eric Coulter, Eroma Abeysinghe, Sudhakar Pamidighantam, and Marlon Pierce. 2019. Virtual Clusters in the Jetstream Cloud: A story of elasticized HPC. In *Humans in the Loop: Enabling and Facilitating Research on Cloud Computing (HARC '19), July 29, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3355738.3355752

1 INTRODUCTION

One of the great challenges in research computing during the last decade has been understanding the role that so-called cloud computing will play in the future of scientific research. While debates over performance and cost rage on, one clear benefit remains: flexibility. National-scale projects such as Jetstream[21], ChameleonCloud[16], and RedCloud[17] strive to offer researchers the experience of commercial cloud systems, without the attendant costs and headache of using grant funding to pay for resources from outside of their institutions. Even at the national level, the NSF has begun requesting projects that aim to reconcile the gap between researchers and commercial cloud offerings[1] despite the potential for better return on investment when such resources are implemented on the research side of the fence[10]. Cloud resources themselves aside, however, an additional challenge remains of actually getting researchers to utilize them effectively. Outside of the swathes of researchers using community codes that are well-supported by traditional HPC computing, many individual researchers do not have the wherewithal to encompass all the tasks involved in disseminating quality research software or making it available on publicly funded cyberinfrastructure resources for use by other scientists, but such sharing of software is an absolute necessity for reproducible science and the spread of ideas. To combat this issue, the Science Gateways Research Center (SGRC) at Indiana University provides the Apache Airavata[19] platform, which allows researchers to share access to their software with an entire national community of users via easily configured Science Gateways. Additionally, there are still significant barriers to researchers at smaller institutions that prevent them from having access to modern research computing systems. The XSEDE Cyberinfrastructure Resource Integration team [11, 22] is funded to provide help in these situations, with implementing local compute resources, or integrating local resources with national ones through the use of data transfer or authentication software. Working under these umbrellas, the authors have created an easily modified method for providing researchers access to flexible cloud technologies while removing the burden of system administration from researchers' shoulders. We discuss the design and evolution of this cloud-based system, the challenges associated with creating and maintaining individual instances of it, and the variety of science enabled by these virtual clusters. We also discuss the issue of "humans in the loop", and why they are unlikely to disappear anytime soon.

1.1 Context

This paper discusses work done primarily within the Jetstream cloud, though the methods used are generalizable to any cloud provider with an API for creation of instances, images, and storage volumes. While many cloud operations hinge on cost-effectiveness,

HARC '19, July 29, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7279-4/19/07…$15.00
https://doi.org/10.1145/3355738.3355752

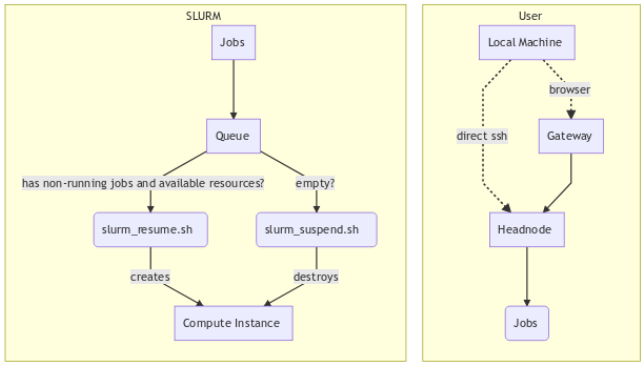

Figure 1: This illustrates the user workflow and steps taken by SLURM to automate job processing in the cloud. Dotted lines indicate steps requiring human intervention.

the primary driver of efficient computation in the case of Jetstream is the use of "Service Units", which are charged to a Users' allocation at the rate of 1 SU per CPU/hour. Thus, an "m1.large" VM with 10 CPUs will burn SU's at a rate of 10 per hour. The Jetstream cloud is open to any US-based researcher, at the minimal cost of asking for a startup allocation through the XSEDE Resource Allocation System[24].

2 VIRTUAL CLUSTER DESIGN

The Virtual Cluster (VC) has a simple cluster architecture: a single headnode, running SLURM[25], controls some number of compute nodes over an internal network. A large filesystem based on an Openstack volume is shared across the system via the Network FileSystem protocol (NFS), though in some cases alternate sources, such as Wrangler, have been used to provide access to large amounts of research storage. The headnode is accessible via a public IP address, which is persistent across reboots or shutdowns. No backup capabilities are provided by default; this is up to the individual user. In the current version, compute nodes are elastically created on-demand as jobs are submitted to the SLURM queue; this was not always the case, but now provides a very efficient mechanism of using precious compute allocations efficiently. The headnode is typically an "m1.small" instance, with 2 CPUs and 8 GB of RAM. Most science gateways do not produce sufficient load to require more resources for simple job submission, which allows the vast majority of an allocation to be spent on compute nodes, which often run to 44 CPU "m1.xxlarge" instances.

2.1 Evolution

The VC project began in 2017, and has undergone significant changes in the meantime. Originally, the project focused on creating templates for a static, HPC-style resource in the Jetstream cloud. The design was kept as simple as possible, similar to the XSEDE-Compatible Basic Cluster (XCBC) toolkit[15] managed by the XSEDE Cyberinfrastructure Resource Integration (XCRI) team. The original design [14] was based on Heat[4] templates, to create a simple static set of instances, on which the necessary software was installed via Ansible[8]. We envisioned eventually creating a dynamic system, in which compute instances could be created and destroyed based on the number of jobs in the queue. At the time, our options for linking the Openstack environment to the resource manager were non-existent, so this was left as a future exercise once the viability of HPC-in-the-cloud was proven. Given our initial investigation, it appeared that we would need to create a customized plugin to the resource manager, which would require a significant time investment. TORQUE[5] was chosen as the resource manager to begin with, to keep in sync with the XCBC, but as the OpenHPC project grew, we moved to SLURM, which was a fruitful decision on all counts. The main difference between the initial VC and applying the standard XCBC build to VMs was the lack of node management software on the headnode, as compute instances were created via Openstack, and configured with Ansible. Building custom images was not investigated initially, due to the perception that managing a catalogue of various compute images for each customer would rapidly become unwieldy.

After the SLURM v. 17.11 update, it became clear from some investigation on mailing lists and a bit of experimentation that having a truly dynamic virtual cluster was easily done without creating customized plugins for Openstack. The update fixed a set of features designed to allow SLURM to remotely power hardware nodes up and down through IMPI interfaces. By creating power management scripts which call the Openstack API, we were able to have an elastic virtual cluster. In the first version of the node creation scripts, we just created basic CentOS images, followed by applying Ansible playbooks to install any necessary software, set up the SLURM client daemons, shared filesystems, etc. This also required a large number of error checks, to be sure that the instance was created properly, that network services were available and that everything installed correctly. Due to the inherent fragility of distributed systems (networks will go down!), this was broke relatively often in practice. Creating new compute instances was also quite slow, as we depended on the yum package manager to install software.

After several iterations on the same theme, we arrived at a scheme in which we create an "image" which has the necessary scientific software, SLURM client configuration, and NFS mounts built-in, so that no other steps beyond instance creation are needed in the scripts. This led to a drastic reduction in node creation failures and wait times.

In the current version of the VC, the average wait time for a single job to start is only about 50 seconds, the amount of time it takes for the compute instance to become available. The total time from creation of the headnode to running a simple test job is about 10 minutes, which is similar to the startup time required on commercial cloud providers.

2.2 Architecture

The architecture of the virtual cluster in the space of virtual hardware is quite standard in the realm of research computing, but the architecture of the code to automatically create them may be of interest. As described in section 2.1, the project has gone through several cycles, the best parts of which we have tried to keep.

As much as possible, one of the primary drivers of automating the virtual cluster build has been the desire to enable researchers

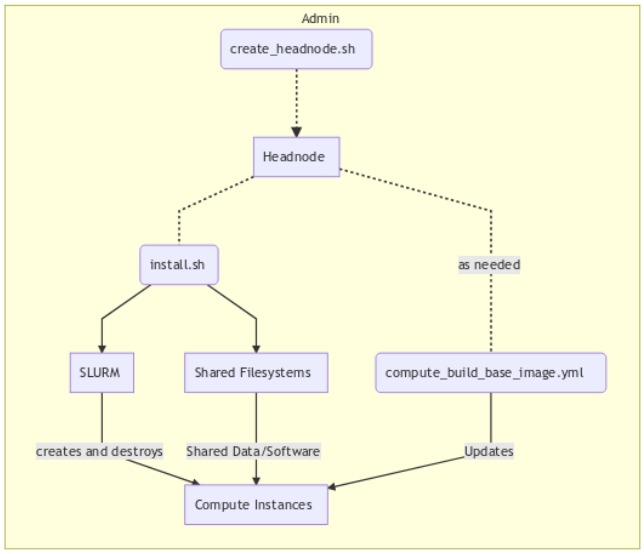

Figure 2: This illustrates the workflow for creation of a new virtual cluster. Dotted lines indicate steps in which a human is necessarily involved in modifying configuration files or running commands.

to stand up their *own* resources without intervention from one of the SGRC members, or from XCRI (the difficulties in managing a cluster resource notwithstanding). A couple of constraints have kept us from making the VC available through the Atmosphere interface:

- Persistent IP addresses are not supported via Atmosphere
- There was no Atmosphere CLI available for most of the lifetime of this project.

The VC is created via two main shell scripts. The first, run on a users local machine, simply creates the headnode instance. The second, run on the new headnode, installs and configures SLURM, and creates an image to be used for creating new compute nodes, via the Openstack CLI. An Ansible playbook is also included to allow straightforward customization of the image to be used for creating compute instances. In most cases, it has been sufficient to have software available on the shared filesystem, but occasionally it is necessary to "bake" dependencies in to the compute image. Beyond these, it is typically unnecessary for users to interact with any other pieces of the code.

Behind the scenes, SLURM does most of the heavy lifting through its built-in power-saving mechanisms, and awareness that not all nodes will respond at all times (this is the piece that was actually updated in the 17.11 release, allowing us to "elasticize" the virtual cluster). Two scripts handle the automated creation and destruction of compute instances, known to SLURM as "ResumeProgram" and "SuspendProgram" - these simply rely on the existence of a pre-made image specific to the cluster, which is configured to accept the headnode as its SLURM controller. Due to issues encountered during workshops, in which multiple users may spin up virtual clusters under a single allocation, all of the above create the internal Openstack pieces under user and project-delimited names.

2.3 Implementation Process

The process of implementing a new VC depends highly on the research project in question. In some cases, pieces of this architecture have been taken and fit into extant frameworks, and in other, members of the SGRC and XCRI team continue to support the virtual resource indefinitely through strong collaborative agreements.

Initial meetings with researchers involve uncovering their computational needs and expectations, to determine whether or not the VC is a good fit for their work. This architecture is particularly well suited to projects that run codes from smaller communities, or are working on their own software, as such applications are often difficult to have installed on large national HPC resources. Once a good fit has been determined, one of the SGRC team members is added to the projects' Jetstream allocation, to create a "blank" virtual cluster. Project members rarely take control of this step, as the entire point of this project is to *remove* the burdens of system administration from researchers. The VC "hardware" is tailored to the project in terms of storage - depending on the project, this can range from large storage volumes mounted from Jetstream, or network-mounting the Wrangler[20] filesystem, which can allow direct access to a huge amount of data.

Next, the researcher will log on to the VC with administrator privileges, and begin installing their particular software. A period of testing follows, in which the researcher may modify their codes to fit a batch computing paradigm, or result in changes to the handling of job submission scripts by the gateway. At this point, compute instance sizes are also tested, to ensure that maximum CPU useage is achieved by each job - multiple queues with different size VMs are also possible, or even with different available software.

Once testing is complete, the VC enters production mode, and is rarely touched by the SGRC team, except when updates are needed or outages occur.

The code for creating virtual clusters with this framework is also freely available on Github at https://github.com/XSEDE/CRI_Jetstream_Cluster, and linked to in the Jetstream documentation[3]. This means that VCs may also be freely implemented by anyone with a Jetstream account, although we have not been contacted by any users who have done so without consulting with one of our teams first.

2.4 Maintenance and Updates

While in theory these dynamic clusters are easily created and destroyed, and can be "backed up" in the sense that one can create a snapshot of the headnode itself for future use, in practice researchers rarely take this tack. For one thing, having to take the additional step of creating a resource prior do doing work is exactly the type of thing this project is intended to avoid. In practice, the headnodes to these clusters have stayed up continuously for years, with only occasional reboots. In the case of heavily used resources, it is often difficult to convince faculty that restarts are needed, which is understandable given the role in which gateway administrators may be facing dozens or hundreds of users, or may themselves be pushing hard to make progress on a particular research project. Thankfully, the design of the VC lends itself well to updates - the OpenHPC project provides regular updates to the

SLURM software, as well as varieties of compiler and MPI implementations. The biggest challenge lies in keeping the compute nodes and headnode updated in sync. In a middle incarnation of the VC, it was possible for compute nodes to be created with a newer version of SLURM when new releases were out, causing incompatibility with the SLURM controller on the headnode, for example. In any case, human intervention is always necessary to keep these machines updated safely, due to potential changes in configuration files, awareness of security updates, and knowledge of when it is necessary to rebuild the compute node image for a particular cluster.

In the case of updates to the VC architecture itself, things become a bit more complicated. By this, we mean changes to the methods by which compute nodes are created and destroyed. These kind of systematic updates require downtime of uncertain length across multiple projects, which can be difficult to coordinate. Gateway admins, cluster admins, and researchers all must be either amenable to a pause in their work or free to apply and test updates at the same time. Additionally, the relationship between virtual cluster admins and gateway operators is frequently more flexible than that between local hardware resource operators and their supported faculty, in which central IT may simply dictate monthly or quarterly downtime for cluster updates. The sheer number of supported machines also becomes overwhelming in the case of virtual clusters, which is in some since a good problem to have.

2.4.1 Monitoring. While it is possible to monitor the VCs with Zabbix or Nagios, these solutions still require the addition of each new VC to a central location, and are not site-independent (i.e., any given user may have a different local monitoring solution). In order to provide a generally useful, minimal monitor of the state of the VC, we've implemented scripts to monitor the state of the compute nodes as known by SLURM. These are triggered by "cron" regularly throughout the day, and simply email the cluster administrator in case of any nodes are in a "down" state. In most cases of malfunction so far, this has been sufficient to catch most errors before the "user based alarm system" is triggered. The main causes of malfunctions have been updates to the underlying cloud infrastructure, interruptions in network of some service or another, or changes to the yum repositories on which compute nodes depended.

3 EDUCATION

A natural outgrowth of this work has been the use of Virtual Clusters in educational content. Due to our involvement with the XCRI group, such content is primarily aimed at HPC cluster administrators, although we imagine that such topics are useful for users in order to better understand the computational resources they take advantage of. The flexible resources made available by the cloud have been useful in two ways: providing a framework to aim for when teaching attendees how to build their *own* CI resources, and in providing ready-made, zero-wait-time compute resources to workshop attendees.

In the first case, we actually walk attendees through the process of building an HPC-style cluster step-by-step on the commandline - first creating instances in Openstack, installing SLURM, and configuring compute nodes by hand. While somewhat tedious, it's a valuable method of exposing a new administrator to the moving pieces of a computing resource that can break, in a single session. At the end of the day, of course, we demonstrate the speed with which one can build a VC using scripts. While in some sense the automated version makes knowledge of the underlying components obsolete, administrators of persistent hardware resources do not have the luxury of rapidly destroying and re-deploying their machines. While our "push-button" solution is capable of abstracting the creation of an HPC resource , we feel it is disingenuous to promise that everything will just work to a degree that user/admin education is unnecessary.

Without hands-on knowledge of the actual system, it's not completely feasible to hand these off to users (researchers) to be supported without help, due to the necessity for human intervention during maintenance. While the automated version of the Virtual Cluster is truly useful for helping researchers increase the resources available to them, or for maximizing efficient use of the Jetstream cloud, the steps taken towards creating it have been most valuable in furthering the pipeline of CI professionals. Additionally, these hands-on experiences have been invaluable for highlighting pain points encountered when users attempt to build and operate batch computing systems for the first time.

In the second case, we have found great success using the VC as a bespoke resource for tutorials and workshops on the Apache Airavata gateway middleware. While Airavata is a truly powerful platform capable of providing seamless access to many of the publicly available research computing resources in the United States, it is *not* always possible to have quick and easy access to those resources for a new batch of 20 workshop attendees. By creating our own resource for a workshop, it is possible to have a completely empty queue, save for our attendees.

4 PROJECTS USING THE VC

The projects using VCs on Jetstream run the gamut from traditional molecular dynamics to textual analysis.

The first project to use this VC architecture on Jetstream was the SEAGrid (Science and Engineering Applications Grid) project[13] SEAGrid offers researchers access to a variety of computational chemistry, molecular dynamics, structural mechanics, and fluid dynamics tools across the national CI landscape.

The Galaxy project[6], which provides users with a wide array of bioinformatics tools, through science gateways, uses a cloud infrastructure[7] system that now involves a modified version of this architecture, in which SLURM control nodes launch the necessary dynamic compute instances via openstack commands, though the infrastructure is controlled via Ansible, and relies on a variety of other platforms, such as CernVMFS.

The Ultrascan project[12], which allows researchers access to automated image analysis workflows for ultracentrifugation analysis, has been a heavy user of the VC project for 1.5 years, running tens of thousands of jobs, and surviving several architectural updates. The Ultrascan headnode has an average uptime of 106 days, which speaks to both the stability of the platform and the difficulties in convincing active researchers of the necessity of maintenance downtime. This platform has also been used in a handful of tutorials throughout its lifetime, and has proven quite capable of handling the load of 30+ simultaneous users on a small virtual machine.

The 3-Dimensional Quantitative Phenotyping Gateway[18] operated a VC allowing users access to process and analyze imagery and 3-D modeling data for model organisms for one year. Access was primarily granted via the commandline, which is unusual for a gateway project.

The InterACTWEL (Interactive Adaptation and Collaboration Tool for managing Water, Energy and Land) project [9] provides an interactive gateway for the intersection of Land, Water, and Energy research, to maximize effective land use in the Umatilla region of Oregon. The VC allows users to run SWAT analysis tools on user-uploaded data sets, via a highly interactive gateway backended by Apache Airavata and customized post-processing tools.

The Distant Reader project[2], run by Eric Morgan of Notre Dame University, gives users a suite of text analysis tools for so-called automated "distant reading" of a text vs. the close reading that a human reader is capable of. The project relies heavily on custom software.

The Neuroscience Gateway project[23] utilizes a VC to allow researchers to run analysis workflows with the neuroscience modelling tools and data processing tools, including community tools based on MATLAB. Thanks to a site license available to Jetstream, such tools are available to any researcher in the US, even at institutions which do not themselves have such a license. This increases researcher access to state-of-the-art tools in two ways: making software more available to researchers without strong financial support, and removing the system administration burden of installing community software. This was the first VC to feature multiple queues for different software stacks - those requiring MATLAB, and those without.

5 CONCLUSION: HUMANS IN THE LOOP

As illustrated in Figure 3, the human involvement required at each layer of interaction with the VC system is quite different. Behind the scenes, the VC project has taken hundreds of hours of work to arrive at the relatively stable platform it offers today (note, however, that this includes time spent customizing individual instances of the VC for new projects). While it is still necessary to include human effort in maintaining these cloud systems, the existence of a scripted, easy-to-configure starting point has enabled us to develop this toolkit and support the previously discussed projects on approximately 0.25 FTE across a couple of individuals. This type of invisible work supporting research infrastructure is highly important in supporting the national ecosystem, but continually suffers from a lack of funding, similar to the actual physical infrastructure in the US.

While the virtual cluster system works at a high level of automation once fully configured, there are many places in the "loop" that require human intervention. Much of this is due to the fact that it is intended to serve a wide range of scientific need - the research community does not standardize on anything, nor should it. However, this makes the creation of "plug and play" compute resources a fool's errand. The most we can hope for is to minimize the boring parts, and require human interference *only* for those aspects of a resource which hinge on the details of the research project in question. Additionally, our experiences with the virtual cluster project highlight the need for regular human intervention

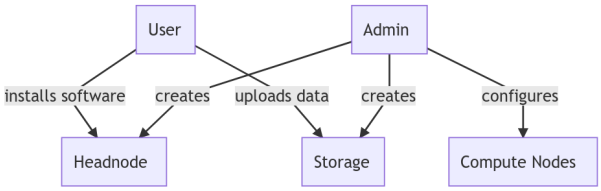

Figure 3: Illustration of the cardinal ways in which different humans interact with the VC system.

in updating, maintaining, and improving the performance of persistent cloud-based platforms. Due to the ever-changing nature of cloud providers, and the lack of control guests in their ecosystem have (particularly in large commercial environments), any platform living on top of the sea of parts that make up a cloud could encounter breaking updates from simple things like naming conventions, product updates, or network outages. Providers of such tools must be aware of changes to their ecosystem, to forestall angry faculty knocking at their inbox. At the same time, however, the flexibility of "infrastructure-as-a-service" cloud ecosystems is undeniably powerful when leveraged with the long tail of science in mind, and enables a much wider range of computational science than traditional HPC systems.

ACKNOWLEDGMENTS

This work was funded by the eXtreme Scientific and Engineering Discovery Environment (XSEDE) - NSF Grant ACI-1548562 - as ongoing work in the Cyberinfrastructure Resource Integration team. This work also made use of the Wrangler system (NSF Grant OAC-1341711) and the Jetsream Research Cloud funded by NSF Grant OAC-1445604.

REFERENCES

[1] [n. d.]. NSF Campus Cyberinfrastructure Solicitation 19-533. https://www.nsf.gov/pubs/2019/nsf19533/nsf19533.htm
[2] 2019. Distant Reader Gateway. https://distantreader.org/
[3] 2019. Jetstream User Wiki. https://wiki.jetstream-cloud.org/
[4] 2019. Openstack Heat Project. https://wiki.openstack.org/wiki/Heat
[5] 2019. Torque Resource Manager. http://www.adaptivecomputing.com/products/torque/
[6] Enis Afgan, Dannon Baker, BÃĪrÃĪnice Batut, Marius vanÂădenÂăBeek, Dave Bouvier, Martin ÄŇech, John Chilton, Dave Clements, Nate Coraor, BjÃűrn A GrÃijning, Aysam Guerler, Jennifer Hillman-Jackson, Saskia Hiltemann, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. 2018. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research* 46, W1 (05 2018), W537–W544. https://doi.org/10.1093/nar/gky379 arXiv:http://oup.prod.sis.lan/nar/article-pdf/46/W1/W537/25110642/gky379.pdf
[7] Enis Afgan, Andrew Lonie, James Taylor, and Nuwan Goonasekera. 2019. CloudLaunch: Discover and deploy cloud applications. *Future Generation Computer Systems* 94 (2019), 802 – 810. https://doi.org/10.1016/j.future.2018.04.037
[8] Ansible Configuration Management Software 2018. Ansible Configuration Management Software. http://www.ansible.com/. [Online; accessed 17-August-2018; Red Hat Inc.].
[9] Meghna Babbar-Sebens, Snehasis Mukhopadhyay, Ganti Murthy, Jeff Reimer, Arjan Durresi Jenna H. Tilt, and Samuel J Rivera. 2019. InterACTWEL. http://interactwel.oregonstate.edu/
[10] Evan F. Bollig and James C. Wilgenbusch. 2018. From Bare Metal to Virtual: Lessons Learned when a Supercomputing Institute Deploys Its First Cloud. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC*

'18). ACM, New York, NY, USA, Article 13, 8 pages. https://doi.org/10.1145/3219104.3219164

[11] Eric Coulter, Jeremy Fischer, Barbara Hallock, Richard Knepper, and Craig Stewart. 2016. Implementation of Simple XSEDE-Like Clusters: Science Enabled and Lessons Learned. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16)*. ACM, New York, NY, USA, Article 10, 8 pages. https://doi.org/10.1145/2949550.2949570

[12] B Demeler. 2005. UltraScan A comprehensive data analysis software package for analytical ultracentrifugation experiments. *Modern Analytical Ultracentrifugation: Techniques and Methods* (01 2005), 210–229.

[13] Rion Dooley, Kent Milfeld, Chona Guiang, Sudhakar Pamidighantam, and Gabrielle Allen. 2006. From proposal to production: Lessons learned developing the computational chemistry grid cyberinfrastructure. *Journal of Grid Computing* 4, 2 (2006), 195–208.

[14] Sudhakar Pamidighantam Eric Coulter, Richard Knepper. 2017. Using the Jetstream Research Cloud to provide Science Gateway resources. *CCGrid*.

[15] Jeremy Fischer, Richard Knepper, Matthew Standish, Craig A. Stewart, Resa Alvord, David Lifka, Barbara Hallock, and Victor Hazlewood. 2014. Methods For Creating XSEDE Compatible Clusters. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE '14)*. ACM, New York, NY, USA, Article 74, 5 pages. https://doi.org/10.1145/2616498.2616578

[16] Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. 2019. Chameleon: a Scalable Production Testbed for Computer Science Research. In *Contemporary High Performance Computing: From Petascale toward Exascale* (1 ed.), Jeffrey Vetter (Ed.). Chapman & Hall/CRC Computational Science, Vol. 3. CRC Press, Boca Raton, FL, Chapter 5, 123–148.

[17] Steven Lee. 2017. Cornell Red Cloud: Campus-based Hybrid Cloud. http://hdl.handle.net/2022/21730

[18] Murat Maga. 2018. 3-Dimensional Quantitative Phenotyping Gateway. http://faculty.washington.edu/maga/

[19] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, Aleksander Slominski, Ate Douma, Srinath Perera, and Sanjiva Weerawarana. 2011. Apache Airavata: A Framework for Distributed Applications and Computational Workflows. In *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments (GCE '11)*. ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/2110486.2110490

[20] Craig Stewart. 2017. Big Data, Big Red II, Data Capacitor II, Wrangler, Jetstream, and Globus Online: A national science & engineering cloud. http://hdl.handle.net/2022/19680

[21] Craig A. Stewart, Timothy M. Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, Matthew Vaughn, and Niall I. Gaffney. 2015. Jetstream: A Self-provisioned, Scalable Science and Engineering Cloud Environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure (XSEDE '15)*. ACM, New York, NY, USA, Article 29, 8 pages. https://doi.org/10.1145/2792745.2792774

[22] Craig A. Stewart, Richard Knepper, James Ferguson, Felix Bachmann, Ian Foster, Andrew Grimshaw, Victor Hazlewood, and David Lifka. 2012. What is Campus Bridging and What is XSEDE Doing About It?. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the Campus and Beyond (XSEDE '12)*. ACM, New York, NY, USA, Article 47, 8 pages. https://doi.org/10.1145/2335755.2335844

[23] Kenneth Yoshimoto Vadim Astakhov Anita Bandrowski Mary Ann Martone Nicholas T. Carnevale Subhashini Sivagnanam, Amit Majumdar. 2013. Introducing The Neuroscience Gateway. In *Proceedings of the 5th International Workshop on Science Gateways*, Vol. 003. CEUR-WS. http://ceur-ws.org/Vol-993/

[24] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. 2014. XSEDE: Accelerating Scientific Discovery. *Computing in Science & Engineering* 16, 5 (Sept.-Oct. 2014), 62–74. https://doi.org/10.1109/MCSE.2014.80

[25] Andy B. Yoo, Morris A. Jette, and Mark Grondona. 2003. SLURM: Simple Linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*, Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 44–60.