High Availability on Jetstream: Practices and Lessons Learned

John Michael Lowe
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
jomlowe@iu.edu

Jeremy Fischer
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
jeremy@iu.edu

Sanjana Sudarshan
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
ssudarsh@iu.edu

George Turner
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
turnerg@iu.edu

Craig A. Stewart
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
stewart@iu.edu

David Y. Hancock
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
dyhancoc@iu.edu

ABSTRACT

Research computing has traditionally used high performance computing (HPC) clusters and has been a service not given to high availability without a doubling of computational and storage capacity. System maintenance such as security patching, firmware updates, and other system upgrades generally meant that the system would be unavailable for the duration of the work unless one has redundant HPC systems and storage. While efforts were often made to limit downtimes, when it became necessary, maintenance windows might be one to two hours or as much as an entire day. As the National Science Foundation (NSF) began funding non-traditional research systems, looking at ways to provide higher availability for researchers became one focus for service providers. One of the design elements of Jetstream was to have geographic dispersion to maximize availability. This was the first step in a number of design elements intended to make Jetstream exceed the NSF's availability requirements. We will examine the design steps employed, the components of the system and how the availability for each was considered in deployment, how maintenance is handled, and the lessons learned from the design and implementation of the Jetstream cloud.

CCS CONCEPTS

• **Computer systems organization** → **Reliability; Availability; Maintainability and maintenance;**

KEYWORDS

XSEDE, research, cloud, hpc, Atmosphere, availability

ACM Reference Format:

John Michael Lowe, Jeremy Fischer, Sanjana Sudarshan, George Turner, Craig A. Stewart, and David Y. Hancock. 2018. High Availability on Jetstream:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ScienceCloud'18, June 11, 2018, Tempe, AZ, USA

© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.

ACM ISBN 978-1-4503-5863-7/18/06...\$15.00

<https://doi.org/10.1145/3217880.3217884>

Practices and Lessons Learned. In *ScienceCloud'18: 9th Workshop on Scientific Cloud Computing, June 11, 2018, Tempe, AZ, USA*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3217880.3217884>

1 INTRODUCTION

For decades, scientists in the US have been accustomed to a very simple trade-off in doing their computational work: do what you could on a desktop computer system and, if that was not sufficient computing power, then use some sort of high performance computing (HPC) system. It was a choice of greater computing power over less convenience in the user interface, runtime, and availability. With traditional supercomputers and HPC systems, administrators and users knew that the trade-offs for being at the cutting edge of computing power were that systems often went down unexpectedly, might need downtime for software updates on short notice, and non-zero queue times were expected. All of this seemed a reasonable trade-off and centers that required greater operational availability turned to acquiring multiple systems to combine operational and research workloads.

Two trends have been affecting researcher expectations and system design plans. First, there are many scientists who are using some sort of software that requires more computing power than the researcher has on their own desk but perhaps less than the resources by HPC systems. This means that their computing needs can often be met with a large virtual machine (VM) the likes of which one can buy routinely from a variety of commercial cloud operators. Generally, when one buys computing services through such a provider, the industry standard is simply that such commercial resources are up and running essentially 24 x 7 x 365. This does not mean cloud resources automatically enable end user applications to be highly available but does mean that individual regions are generally highly available, without a single point of failure, and the techniques are possible for users to construct their applications in a fully high availability (HA) manner.

In 2014, the National Science Foundation (NSF) announced an interest in funding new types of computing systems, different than a traditional supercomputer, to add to the suite of computing systems supported by the NSF. The NSF solicitation released in 2014 that led to the creation of the Jetstream cloud system stated specifically: "The current solicitation is intended to complement previous NSF investments ... by exploring new and creative approaches to

Figure 1: Jetstream cloud architecture

delivering innovative computational resources to an increasingly diverse community and portfolio of scientific research and education” [15]. While cloud-based systems were not explicitly mentioned, it seemed an obvious design idea for this solicitation.

The Indiana University Pervasive Technology Institute (PTI) and a number of partners proposed to create a new cloud system for use by the national research community. This system we proposed to call Jetstream - by analogy with the Jetstream in the upper atmosphere which moves rapidly and lies at the boundary of two large air masses. The Jetstream cloud system was designed to be rapidly responsive to user needs and lie at the boundary of pre-existing NSF-funded cyberinfrastructure and new users of national cyberinfrastructure. With the Jetstream system, we planned to make a system designed for a new generation of users of NSF-funded cyberinfrastructure that would be highly available and highly user friendly [24]. This meant a system and interface design built around delivering applications to users that were not well-served by existing infrastructure and to do so with extraordinary customer service and application of high availability techniques that were beginning to become routine in commercial and private cloud providers. Our motto is that Jetstream is a “managed science cloud - a cloud managed for science.” In that motto, we embed our commitment to serve the scientific research community and, at the same time, set expectations that the level of service availability of Jetstream will match what researchers expect from commercial cloud providers.

In this report we provide an overview of Jetstream services, descriptions of how they are configured to support high availability, and the lessons that we have learned in the process of implementing and supporting such an environment.

2 ARCHITECTURE

Jetstream was designed in a distributed configuration that is modeled after commercial cloud systems in terms of uptime, reliability, and accessibility [25]. Jetstream consists of two geographically distributed clusters for public use, one at Indiana University and the other at the Texas Advanced Computing Center in Austin, TX, with a small, private test system at the University of Arizona that is only available to Jetstream developers and administrators. Each production instance is connected to Internet2 via a 100 Gbps link. Figure 1 shows the basic configuration of both production clouds and the

test cloud and connecting networks. The intent of this distributed architecture was to help maximize cloud resource availability. While services such as science gateways [30] might be locked into one specific cloud, on-demand resources would still be available even if one cloud was off-line.

When looking at system availability in this perspective, uptime loses meaning as a metric for a cloud environment. A traditional supercomputer resource is generally in operation or not, a binary condition. A distributed cloud resource could be up in the sense of being able to accept logins and having resources available for immediate use while still operating at well less than full capacity. Because of this consideration, total capacity availability of at least 96% was added as a metric by the NSF [24]. This metric was met for operational year one and is on track to meet or exceed goal for operational year two.

This high level architecture approach to building Jetstream was the first step in making a highly availability research system. Other factors such as the configuration of cloud services software, networks, and storage also contribute to making an HA system. A closer examination of these and other components show the efforts in making Jetstream highly available.

3 NETWORK

All compute nodes in Jetstream have two bonded 10Gig Ethernet connections to pairs of stacked chassis switches. The chassis switches are connected to top of rack via 2 x 40Gig Ethernet links. Top of rack switches are connected with 2 x 40Gig Ethernet to each of two stacked spine switches. Only degraded performance will be experienced in the event of link failures or the failure of a chassis or spine switch. The network topology from the top of the rack to spines and data center switch is depicted in Figure 2.

4 STORAGE

Ceph object storage [28] provides block storage for Jetstream. Ceph was selected by the Jetstream team due to its cloud-first approaches and it was one of the few storage projects that could realistically provide software defined storage, avoiding the need to provision the amount of block, object, or POSIX-like storage at the time of initial implementation. The Jetstream team at IU also has nearly 5 years of experience using Ceph for a similar gateway hosting environment

Figure 2: Network Topology

[13], a fact that increased our confidence in its reliability. The ceph cluster for this hosting environment ran for 6 years with a single 8 hour downtime. Ceph uses replication or erasure coding distributed over failure domains while automatically replacing lost or missing data for self-healing. The default mode of data protection for Ceph is synchronous replication with on disk checksums for all data. Volume data intended for persistent storage is replicated at 3x while images and ephemeral root disks for instances are replicated at 2x. In the event of the loss of a replica the system will continue Input/Output (IO) with some performance degradation while a new replica is created on a surviving storage node. Replicas are never stored on the same node which sets the failure domain at the node-level. A monitor process using the Paxos algorithm holds the current revision of the cluster state [9]. The cluster state is used with a deterministic algorithm to compute object placement on storage nodes by clients decentralizing all client data reads and writes. Once the cluster state has been acquired by a client the only centralized communication is receiving the infrequent updates of the cluster state. Coherency is maintained without the overhead and potential source faults of a distributed lock manager.

5 OPENSTACK

OpenStack is an infrastructure delivery platform consisting of more than forty sub-projects all using RESTful application programming interfaces (APIs) [17]. The use of RESTful APIs and associated design patterns facilitate the use of clustering and load balancing technologies to achieve higher degrees of fault tolerance and scaling. Within a given OpenStack project there is a micro-service that implements the RESTful API and maintains all stateful information for the service in a database. MySQL [18] has strong multi-master database clustering provided by the Galera [7] clustering library. HTTP load balancers such as HAProxy [8] can be transparently placed in front of an arbitrary number of API servers for both scaling and fault tolerance. API servers do not take any direct action instead using a message bus, commonly RabbitMQ [19], direct agents to do tasks. RabbitMQ also has strong multi-master clustering support and if an agent fails to successfully complete a task or stops sending heartbeat messages tasks can be rescheduled to a healthy agent. For example if a compute agent fails to build a new instance another compute agent can be tasked with creating that instance, or if the

dynamic host configuration protocol (DHCP) agent fails to generate heartbeats it will not be tasked with providing DHCP services for tenant virtual networks. The three components needed to create a scalable and fault tolerant OpenStack deployment are multi-master databases, multi-master messaging clusters, and a load balancer to route client and inter-project RESTful API requests to healthy API servers. Load balancers can be clustered with a Virtual Router Redundancy Protocol (VRRP) implementation like Keepalived [1] and round robin DNS entries for the virtual IP (Internet Protocol) addresses, again distributing load and eliminating single points of failure.

The API servers for the five orthogonal components of OpenStack (Nova compute, Keystone identity, Glance image, Cinder block storage, Neutron networking) were deployed in triplicate for Jetstream. While not strictly necessary for HA, having three instead of two does provide a significant over-provisioning of API request servicing. The user experience for API request failures due to lack of resources to service API requests is indistinguishable from other failure modes. Agents providing networking services such as DHCP and virtual routers are spread one in each rack across the five physical racks with compute hosts. Virtual routers are scheduled to three agents in a master-backup-backup configuration using VRRP to move the IP addresses of the router to a healthy agent in the event of a fault. This also allows for full utilization of the 100 Gbps connection as each of the five network agent nodes has 20 Gbps of bandwidth and some degree of protection against bandwidth starvation by users competing for upstream network traffic.

Database and message bus services are co-located on nodes with solid state drives (SSDs) in three way master/master/master clusters. The higher mean time between failure (MTBF) and tolerance of adverse environmental conditions contribute to the uptime of the system. Database and message bus workloads are characterized by small random IO served well by the much higher IOPS of SSDs providing increased system capacity for servicing user requests. The design of OpenStack services leads to an alternation of database requests and message generation. Disk contention between database and message bus services is not an issue so co-location maximizes the utilization of the more expensive SSD storage.

6 ATMOSPHERE

Atmosphere is a key component of Jetstream's functionality and serves as the primary user portal for Jetstream projects. Atmosphere was designed by iPlant (now CyVerse) to be an easy-to-use graphical user interface (GUI) for on-demand launching, managing, and using virtual machines and storage volumes [23] [16]. While OpenStack provides a basic GUI environment for managing these services (Horizon), Atmosphere removes the need for researchers and educators to be experts in the techniques and methods for creating and managing VMs, particularly in the aspects of managing security groups, volumes, and provides additional capabilities through Ansible deployment scripts [6].

Atmosphere is currently served from an infrastructure machine in the Jetstream racks at Indiana University. At present, there is no mechanism to have a load-balanced or hot spare Atmosphere server. This makes for a potential single point of failure in Jetstream's user-facing services. VMs that were already operational

would continue to run and be accessible even if the Atmosphere service became unavailable but new launches and instance management of Atmosphere-initiated services would be off-line for users. Discussions with one of the Atmosphere developers yielded that plans for an HA version of Atmosphere is in development though a time line has not been established at the time of writing [11]. Atmosphere does interact with each of the two Jetstream production sites independently, meaning if OpenStack services at IU are unavailable then new and existing instances at TACC can still be managed.

7 MANAGING UPGRADES

Maintenance is an essential part of system administration; in a traditional HPC system, all jobs have a maximum duration that can be scheduled against a system wide reservation for a quiescent period for maintenance. There is no concept of end time for instances launched within a cloud resource so a live migration capability must be used to quiesce nodes for maintenance. The general rule for estimating virtualization overhead is about ten percent slower than bare metal [27]. Most of the overhead of virtualization can be eliminated by using CPU pinning, Non-uniform memory access (NUMA) pinning, and Single Root I/O virtualization (SRIOV) however using these techniques is mutually exclusive with live migration capabilities. In the case of Jetstream live migration capabilities were prioritized, a direct consequence of this priority is reflected in our deployment using standard Ethernet instead of InfiniBand [2] for cluster networking. Prior to maintenance on a given compute all running VMs are live migrated to other compute nodes, maintenance is performed, then the compute node is placed back into service.

Maintenance for OpenStack service nodes is considerably simpler relying on load balancing and redundancy to continue servicing user requests. In the case of major upgrades between releases of OpenStack, one of the three service nodes is taken out of rotation and the packages fully updated. The dependency tree for OpenStack is both wide and deep with testing focusing on all software belonging to the same release. Minimum library versions are sometimes missed as dependencies so it is not advisable to upgrade service by service across all nodes. The past several OpenStack releases have focused on rolling upgrade capabilities withing projects so that within a single release step both older and newer versions can operate on the same database schema without problems. Once all of the packages on a single node have been updated services on that node can be restarted and the older versions on other nodes stopped relying on load balancing to direct user requests to the updated version of the service. The remaining nodes now out of service have package updates applied and are returned to the rotation as the updated services are started. Once all nodes are running the latest version database schema migrations can be applied to finalize the upgrade process.

8 AVAILABILITY AND UPTIME

As noted in Section 2 on Architecture, measuring uptime becomes an almost meaningless metric when discussing highly available cloud resources. The NSF has requested that Jetstream measure several key performance indicators: system availability, capacity

Figure 3: Jetstream Capacity Available

Figure 4: Jetstream Job Completion

availability, and job completion. System availability in essence becomes the notion of is the system in any usable state? Capacity availability asks how much of the system is usable. Job completion in the context of Jetstream cloud defines whether a featured virtual machine image, one maintained by the project, launches successfully for the user [24].

System availability since the early operations period in 2016 has maintained a rate close to 100%. Since coming online, Jetstream has had either IU or TACC cloud available and usable for almost two years now with no complete system interruption. Jetstream has exceeded the goal of 96% system availability for the entire production period to date.

The other two key measurements for Jetstream, capacity availability and job completion, have also met or exceeded the goals set forth by the NSF. In Figure 3 we show the available capacity of Jetstream since June 2016 that are collected monthly. There was one extended outage on the TACC cloud in mid January 2018 that brought availability to its lowest point to date of 95.38%.

Job completion has traditionally been defined in terms of HPC jobs. It refers to a traditional HPC job and whether it ran to completion without being resubmitted. Since cloud systems overall do not have the same sort of batch processing, this metric was redefined

to fit Jetstream. For Jetstream, job completion is defined as whether a featured VM, a VM built and maintained by the Jetstream staff, launches successfully and can be used by a Jetstream user. Figure 4 shows the quarterly collected results showing the success rate of featured VM launches. Jetstream dipped towards the 96% goal line in Project Year 2 Quarter 1 but has maintained a success rate above it since production measurement began. This downturn is explained in section 10 on successes, failures, and how to improve.

9 SUPPORT ASPECTS

Support for highly available systems is often a challenge. One of the biggest factors in support is expectation management, notably communicating well. The Jetstream support team strives to maintain communication with the user community to inform about planned outages as well as unplanned outages. As part of the eXtreme Science and Engineering Discovery Environment (XSEDE) [26], the team utilizes the XSEDE User News [31] page and mailing list to inform users. Also, while it is in beta testing, the team has utilized CachetHQ [3] to build a Jetstream-specific status page that will support a more complete breakdown of services and outages while also providing information about overall system usage and health [10].

While support itself is only as available as the humans that provide it, Jetstream support consistently works on ways to improve how the users are informed and updated on issues. Twenty-four hour a day support is not presently available for Jetstream due to budget and personnel constraints, but when issues arise, multiple team members are able to update XSEDE User News to inform the user base. In the future, technical staff will be able to issue updates on outages and system status via API calls to the CachetHQ service [4] allowing for a somewhat automated approach of error and outage reporting.

The support aspects may not be a direct HA component, but it is one of the ways Jetstream is different from commercial clouds. Jetstream freely provides a level of support to researchers and educators that is not generally available to most individuals on commercial clouds. In addition, Jetstream is supported by organizations and staff that are familiar with researchers and how they generally work. In addition, the support staff creates and maintains both general development images and several domain science images for researchers to use and build from. These aspects help make Jetstream more usable for cloud novices.

10 SUCCESSES, FAILURES, AND HOW TO IMPROVE

When Jetstream was initially deployed unicast Virtual Extensible LAN (VXLAN) [14] was used with the intent of maximizing efficiency. In the unicast scenario, the OpenStack networking service has perfect information and publishes that along with any changes to all network and compute nodes. With Jetstream this worked very well initially but did not scale sufficiently as full operations began. As shown in Figure 4, the trends demonstrate the early reliability with increasing failures as more users were added and load increased (PY2, 1st quarter) then increased job completion once the change to multicast VXLAN was completed on both clouds. The networking services were reconfigured to use multicast VXLAN and

allow the traditional Ethernet and IP discovery of network topology. As a result, the reliability of tenant networking was greatly improved going from the most common failure modes to one of the least common.

At the time of Jetstream's procurement, OpenFlow Software Defined Networking was the *de facto* standard for cloud deployments. The network was designed to have a large number of chassis switches to distribute the workload of doing OpenFlow. During the deployment phase the decision was made to avoid OpenFlow and use VXLAN with Linux kernel bridging. This is now the *de facto* standard for cloud deployments. If the team was designing the network today, pass-through modules instead of chassis switches with redundant top of rack switches would be preferred. This would eliminate the rack-level single point of failure at the top of rack switch while eliminating the added latency of an extra switch hop. Maintaining switch configurations would be greatly simplified by a net reduction in the number of switches by thirty five.

For the 280 compute, five network, and three service nodes currently in operation for one of Jetstream's two clouds there are 622 daemons and agents requiring 1161 unique configuration files for OpenStack. OpenStack releases often require configuration options to be updated as part of the upgrade process. Without strong modern configuration management practices operation of the system would not be possible. Configuration management is done with a combination of Puppet[20], SaltStack[22], and Ansible[21] with as much of the configurations stored in public GitHub repositories as is feasible. While selecting the configuration system that was most familiar to each team allowed for faster deployment (15 business days for the system at IU) not using the same configuration management for both clouds has allowed some divergence in the configuration for each cloud. A potential lesson here would be to try to use the same configuration management software to allow for exact duplication of the cloud configuration.

During project year two, a new storage format for Ceph became available. Transition to new storage formats without service interruption using Ceph is made possible by its HA features. One of the capabilities of the new format is in-line compression. Enabling this feature for the image storage pool does not seem to have had a measurable effect for the performance but has slowed the rate of consumption by about one half. It is advisable for anyone concerned about the storage consumed by images in an OpenStack deployments backed with Ceph to enable this feature if available to them, provided they have sufficiently exceeded the minimum recommendation for storage server processor and memory.

The technical aspects are certainly important in measuring success and failure. The sentiments from educators and researchers ultimately determine the success of the cloud deployment. There have been multiple user surveys both by the Jetstream team and XSEDE. Jetstream users report very high user satisfaction in surveys [29]. The Jetstream team most recently surveyed a subset of users that held a trial allocation or a startup allocation for more than 6 months as of March 2018. Of those surveyed, 80.33% were satisfied or extremely satisfied. Over two thirds of the surveyed users stated that they were likely or extremely likely to request a research allocation to continue their work using Jetstream [5].

11 CONCLUSIONS

Jetstream was created to enable U.S. scientists and researchers who did not make use of National Science Foundation funded cyber-infrastructure in spite of a clear statement of need for additional resources [12]. Interactive use, processing streaming data from instruments, providing resources for classes with hundreds of students are all use cases served best by cloud computing models instead the traditional batch model of HPC. *On demand* necessitates a resource that can service user requests for provisioning resources at any time leading to HA capabilities being a design constraint. We believe this is a fundamental change from implementation of traditional HPC environments and, over time, the practices discussed will become commonplace not only in cloud deployments, but in research computing environments everywhere.

While the HA aspects of system deployment do not always translate to applications also being HA, such practices are the next logical step. Applications do not automatically inherit all of the HA aspects of the system in which they are deployed. They must be designed to take advantage of existing features and supplement with additional resource requests to cover the applications specific failure modes and tolerances. We believe that the desire for Jetstream-like cloud research environments will continue to be important for researchers. These environments can be closely coupled with software defined storage and traditional HPC systems to provide a more complete ecosystem that serves communities with needs well beyond batch-driven computing and caters to a new generation of research applications that employ cloud-native approaches.

Jetstream provides such an environment not just for convenience but increasingly as an expectation from end users. There were certainly questions during project conception of whether the techniques discussed would become mainstream, or provide the stability necessary when using software that was almost exclusively open source. We believe the results of both in Jetstream's implementation and nearly two years of production operations provide a definitive answer. The hardware environment may not excite most HPC enthusiasts but providing highly usable interfaces, enthusiastic customer service, capacity availability in excess of 99%, job completion near 99%, and near-100% availability is certainly appreciated, and perhaps even exciting, to researchers and educators that have embraced Jetstream.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Grant Number ACI-1445604.

This research was also supported by the Indiana University Pervasive Technology Institute and major funding for IUPUI from the Lilly Endowment, Inc.

Atmosphere is developed by CyVerse (formerly The iPlant Collaborative) under grant NSF DBI-1265383.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Lilly Endowment, CyVerse, or XSEDE leadership as a whole.

REFERENCES

- [1] Alexandre Cassen. 2018. Keepalived. (2018). <http://www.keepalived.org/>
- [2] InfiniBand Trade Association. [n. d.]. *InfiniBand architecture specification : release 1.0*. [Portland, OR] : InfiniBand Trade Association, [2000]. <https://search.library.wisc.edu/catalog/999915691102121>
- [3] CachetHQ. 2018. CachetHQ. (2018). <https://cachethq.io/>
- [4] CachetHQ. 2018. CachetHQ API Reference. (2018). <https://docs.cachethq.io/reference>
- [5] Jeremy Fischer, Brian W. Beck, Sanjana Sudarshan, George Turner, Winona Snapp-Childs, Craig A. Stewart, and David Y. Hancock. 2018. Methodologies and practices for adoption of a novel national research environment. In *PEARC '18: Practice and Experience in Advanced Research Computing*. ACM, Pittsburgh, PA, 6. <https://doi.org/10.1145/3219104.3219115>
- [6] J. Fischer, D.Y. Hancock, J.M. Lowe, G. Turner, W. Snapp-Childs, and C.A. Stewart. 2017. Jetstream: A cloud system enabling learning in higher education communities. In *Proceedings ACM SIGUCCS User Services Conference*, Vol. Part F1317. <https://doi.org/10.1145/3123458.3123466>
- [7] Galera. 2018. Galera Cluster. (2018). <http://galeracluster.com/>
- [8] HAProxy. [n. d.]. HAProxy Home Page. ([n. d.]). <http://www.haproxy.org/>
- [9] Inktank Storage. 2014. Monitor Config Reference. (2014). <http://docs.ceph.com/docs/jewel/rados/configuration/mon-config-ref/>
- [10] Jetstream Support Team. 2018. Jetstream Status. (2018). <http://status.jetstream-cloud.org/>
- [11] Julian Pistorius. 2018. Personal communications - Discussions of HA Atmosphere. (2018).
- [12] David Lifka, Ian Foster, Susan Mehringer, Manish Parashar, Paul Redfern, Craig Stewart, and Steve Tuecke. 2013. *XSEDE Cloud Survey Report*. Technical Report. 137 pages. <http://hdl.handle.net/2142/45766>
- [13] M Lowe, C Shields, D Hancock, M Link, C A Stewart, and M Pierce. 2009. Gateway Hosting at Indiana University. In *the TeraGrid 2009 Conference*. Arlington, VA. <http://hdl.handle.net/2022/13944>
- [14] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. 2014. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Network. (2014). <https://doi.org/10.17487/RFC7348>
- [15] National Science Foundation. 2014. NSF Solicitation 14-536. (2014). <http://www.nsf.gov/pubs/2014/nsf14536/nsf14536.htm>
- [16] Merchant Nirav and Eric A N D Goff Stephen A N D Vaughn Matthew A N D Ware Doreen A N D Micklos David A N D Antin Parker Lyons. 2016. The iPlant Collaborative: Cyberinfrastructure for Enabling Data to Discovery for the Life Sciences. *PLOS Biology* 14, 1 (4 2016), 1–9. <https://doi.org/10.1371/journal.pbio.1002342>
- [17] OpenStack. 2018. Home page. (2018). <http://www.openstack.org>
- [18] Oracle. 2018. MySQL Home Page. (2018). <https://www.mysql.com/>
- [19] Pivotal. 2018. RabbitMQ Home Page. (2018). <https://doi.org/>
- [20] Puppet. 2018. Puppet. (2018). <https://puppet.com/>
- [21] Red Hat. 2018. Ansible. (2018). <https://www.ansible.com/>
- [22] SaltStack. 2018. SaltStack. (2018). <https://saltstack.com/>
- [23] Edwin Skidmore, Seung-jin Kim, Sangeeta Kuchimanchi, Sriramu Singaram, Nirav Merchant, and Dan Stanzione. 2011. iPlant atmosphere: a gateway to cloud infrastructure for the plant sciences. (2011), 59–64 pages. <https://doi.org/10.1145/2110486.2110495>
- [24] C.A. Stewart, J. Fischer, N. Merchant, D.C. Stanzione, D.Y. Hancock, T. Cockerill, T. Miller, J. Taylor, M. Vaughn, L. Liming, J.M. Lowe, and E. Skidmore. 2016. Jetstream - Performance, early experiences, and early results. In *ACM International Conference Proceeding Series*, Vol. 17-21-July. <https://doi.org/10.1145/2949550.2949639>
- [25] Craig A Stewart. 2014. Jetstream: A national science and engineering cloud. (2014). <http://hdl.handle.net/2022/19582>
- [26] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, Ralph Roskies, J Ray Scott, and Nancy Wilkins-Diehr. 2014. XSEDE: Accelerating Scientific Discovery. *Computing In Science & Engineering* 16, 5 (2014), 62–74. <https://doi.org/10.1109/MCSE.2014.80>
- [27] John Paul Walters, Vipin Chaudhary, Minsuk Cha, Salvatore Guercio Jr., and Steve Gallo. 2008. A Comparison of Virtualization Technologies for HPC. In *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*. IEEE, 861–868. <https://doi.org/10.1109/AINA.2008.45>
- [28] Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell D E Long, and Carlos Maltzahn. 2006. Ceph: A Scalable, High-performance Distributed File System. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI '06)*. USENIX Association, Berkeley, CA, USA, 307–320. <http://dl.acm.org/citation.cfm?id=1298455.1298485>
- [29] Julie A Wernert, Lizanne DeStefano, and Lorna Rivera. 2017. *2017 eXtreme Science and Engineering Discovery Environment (XSEDE) Annual User Satisfaction Survey Evaluation Report*. Technical Report. https://www.ideals.illinois.edu/bitstream/handle/2142/99064/XSEDE_Survey_2017_FINALREPORT.pdf?sequence=2

[30] XSEDE. 2017. Science Gateways. (2017). <https://www.xsede.org/ecosystem/science-gateways>

[31] XSEDE. 2017. XSEDE User News. (2017). <https://portal.xsede.org/user-news>