Red Cloud and Aristotle: campus clouds and federations

Richard Knepper
Susan Mehringer
Adam Brazier
Brandon Barker
Resa Reynolds
rich.knepper@cornell.edu
shm7@cornell.edu
brazier@cornell.edu
beb82@cornell.edu
resa.reynolds@cornell.edu
Center for Advanced Computing
Cornell University
Ithaca, NY

ABSTRACT

Campus cloud resources represent significant resources for research computing tasks, with the caveat that transitioning to cloud contexts and scaling analyses is not always as simple as it might seem. We detail Red Cloud, Cornell's campus research cloud, and some of the work undertaken by the Center for Advanced Computing (CAC) to help researchers make use of cloud computing technologies. In 2015, Cornell CAC joined with two other universities to develop the Aristotle Cloud Federation, composed of separate campus cloud resources and data sources, supporting a range of science use cases. We discuss the lessons learned from helping researchers leverage both of these science cloud resources as well as leveraging other research cloud infrastructure and transitioning to public cloud.

KEYWORDS

cloud computing, campus cyberinfrastructure, human elements

ACM Reference Format:
Richard Knepper, Susan Mehringer, Adam Brazier, Brandon Barker, and Resa Reynolds. 2019. Red Cloud and Aristotle: campus clouds and federations. 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/10.1145/3355738.3355755

1 INTRODUCTION

The Cornell Center for Advanced Computing (CAC) has explored a range of ways to leverage cloud computing in order to support research efforts. CAC provides a selection of cloud computing resources, both for researchers at Cornell as well as in the broader national context. This paper discusses CAC's cloud initiatives: Red Cloud, an on-premise OpenStack cloud resource for Cornell researchers, and Aristotle, a Federated Cloud resource that includes Red Cloud as well as resources from the University at Buffalo and University of California Santa Barbara. The technical components of the system are soundly outshined by the skills and knowledge of CAC staff, which drive the organization's ability to meet researcher needs. We discuss considerations about implementing cloud resources locally, including choosing and migrating between cloud platforms, moving researcher code into cloud resources, a cost recovery model for cloud services, and engagement with public cloud resources. We also describe future activities for research that the CAC has marked as strategic directions for supporting research conducted on cloud resources.

HARC '19, July 28–August 01, 2019, Chicago, IL

ACM ISBN 978-1-4503-7279-4/19/07…$15.00
https://doi.org/10.1145/3355738.3355755

2 THE CENTER FOR ADVANCED COMPUTING - OVERVIEW

The CAC provides services for researchers at Cornell University, based on a cost recovery model, including expert consulting, cluster management, storage, and cloud computing. Rather than provide a centralized resource for all Cornell users, CAC services are tuned to meet the needs of specific research units. Faculty engage CAC on a per-project basis and leverage a broad range of expertise, including systems and storage, database design and management, application development, optimization, and deployment, visualization, and cloud computing and containerization.

The project-focused nature of Cornell CAC's model means that staff time is spent in meeting individual researchers' needs. CAC staff members engage with researchers closely in their projects and have a high level of collaborative work as a result, occasionally to the extent that CAC take part in co-authorship and grant funding opportunities due to the depth of engagement. This model also means that particular technical needs of a given project can build staff member expertise in that area that can then be used to support other activities, so that knowledge building within the center's set of activities increases the range of expertise that can be offered to further projects.

In addition to project based consulting at the CAC, there are general service offerings which meet the needs of Cornell researchers:

storage, cluster management, and cloud computing. CAC storage is provided for both online and archival needs, with a moderate cost for recovering the costs of storage system hardware and components. The CAC filesystem provides both file and object store access for researchers, including mountable, Amazon S3, and Globus transfer capabilities, and a data transfer node with direct connection to Cornell's 100Gb Internet2 connection. For faculty who need require highly-coupled systems or have needs for continuous throughput of HPC workflows, CAC staff handle cluster set-up, scheduling, maintenance and patching for faculty-purchased clusters in the Cornell datacenter, with a scaled cost recovery model based on the number of nodes under CAC management. As the number of managed nodes increases, the per-node cost decreases. For faculty in need of more flexible computing with manageable infrastructure and costing, Cornell CAC has created Red Cloud, an Infrastructure-as-a-Service offering based on a pay-as-you-go plan.

2.1 Cornell Red Cloud

The Red Cloud on premise cloud computing resource is intended to meet the needs of researchers who have computational needs that can be served by an Infrastructure as a Service (IaaS) platform, with a broad variety of available needs and with a significant amount of resources in individual cloud instances. Red Cloud was initially implemented using the Eucalyptus cloud platform [5], but with the end of support for Eucalyptus, the CAC has migrated the system to the OpenStack cloud platform. CAC has engaged a number of different research projects on Red Cloud and has been able to support a wide range of scientific disciplines.

Red Cloud was designed to provide significant amounts of computational capacity to researchers within instances. Red Cloud nodes are connected by standard gigabit ethernet between nodes, with a 40Gb uplink to Cornell's 100Gb Internet2 connection. Currently, Red Cloud consists of over 1,000 cores, which are available in instance sizes up to 28 cores with a standard 8GB of memory allocated to each core, making the largest instance available (designated "c28.m224") a 28-core, 224GB system. Red Cloud has just made GPU instances generally available, with GPU-enabled instances up to 4 Nvidia V100 GPU's. Red Cloud users are provided a 50GB storage volume initially with additional storage available as users re quire. Block storage for Red Cloud instances is provided via a Ceph storage pool. CAC's Ceph pool currently provides roughly 1.5PB of storage.

Red Cloud was initially architected and implemented using the Eucalyptus virtual computing software. This software choice was based first and foremost on Eucalyptus' API-level compatibility with Amazon AWS services. Since the initial deployment of Red Cloud was considerably smaller than its current size, the reasoning was that researchers making use of Red Cloud would easily transition to AWS in the case that their analyses needed to scale beyond the scope of what was readily available in Red Cloud. When official support for Eucalyptus provided by a subsidiary of HP was discontinued, the CAC team elected to move to OpenStack as a cloud platform with considerable viability in the broader community and retain options to scale to AWS when needed, if without API capability.

2.2 The Red Cloud Subscription Model

Access to Red Cloud is available on a subscription basis, researchers purchase subscriptions to core-hours as they need resources. An individual subscription is equivalent to one core-year of computing time, and includes access to 50GB of storage. Subscriptions can be used as fast or as slowly as researchers desire and never expire. A subscription can be used for as many or as few cores as needed by the researcher at the time, in order to make the most of research dollars spent on Red Cloud services.

The subscription model was created in order to reduce risk and prevent cost overruns, commonly seen on public cloud services, where cloud instances run outside the necessary time and continue to incur usage fees. With a Red Cloud subscription, researchers use up to their purchased number of subscriptions and then are able to stop work or purchase additional subscriptions to keep working, preventing cost overruns that can be painful to rectify. Red Cloud resources are provisioned in a responsive fashion–as the cloud reaches a set amount of utilization, additional nodes are purchased to ensure that there is sufficient capacity for continued growth. Based on this model, as more subscriptions are purchased and usage increases, the cloud infrastructure is extended to meet demands for computational resources.

2.3 Red Cloud for CISER

The Cornell Institute for Social and Economic Research (CISER) has a long history of providing large scale computational resources to Cornell social scientists and their collaborators. Two years ago, CAC began moving CISER's computing infrastructure to Red Cloud. Red Cloud images are preconfigured with statistical software and users are granted access via remote desktop or vnc, one user per instance. Researchers have direct access to the CISER Data Archive as well as their home directories. The result is a significant reduction in systems management overhead (compared to bare metal) as well as flexible operating system offerings (Windows or Linux) and users are more productive because they have full access to the cores and memory available via Red Cloud instances. As the number of CISER users increases, more instances are employed to meet demand, and scaling needs have been met by Red Cloud resources without issue thus far. The CAC has begun planning to move the Cornell Restricted Access Data Center (CRADC), a secure computing environment for working with restricted data, to a campus cloud environment.

3 ARISTOTLE FEDERATED CLOUD

The Aristotle Federated Cloud provides access to resources across three academic institutions, based on a federated user and accounting model that allows for use of disparate data sets at each of the federation members' sites. Aristotle supports seven distinct science use cases, which previously leveraged traditional high-performance resources and required adaptation to make use of cloud resources. In the following section we describe the development of the Aristotle Federated Cloud project, detail a set of the science use cases supported by Aristotle, and discuss the process of migrating these research analyses to cloud computing. We conclude with prescriptions for moving research projects to the cloud and describe some of

the issues to be overcome and benefits of cloud computing models for scientific research.

3.1 Aristotle Components

The Aristotle Federated Cloud allows researchers from any of the member sites to access other clouds in the federation. Aristotle project generation and management is conducted through the Aristotle Portal, which provides functions for user management, project initiation and monitoring, and administration. In order to allow access to individual clouds, all Federation clouds use Globus Auth [7] in order to authenticate to cloud resources. Cloud usage is tracked at each of the individual sites and sent to the Aristotle portal. Reconciliation for usage of other federation resources is done on a one-to-one core-hour basis: as a researcher uses cloud resources at another federation member's cloud, that federation member accrues credits which can be spent on other cloud resources. Based on the availability of different instance types, software, and data sources at other cloud federation members, researchers may elect to use credits where the infrastructure best meets the needs of their analyses.

The management of the science team effort evolves as the scientific users become more familiar with the infrastructure; initially, Aristotle personnel work with researchers to elicit requirements and architect a cloud-based solution on Aristotle hardware, but as researchers become more familiar with the move to cloud, as detailed in 4.1, they take ownership of ongoing development and architecturing new, related work. Communication with researchers is initially conducted via meetings and then ongoing communication is conducted via Slack; researchers regularly submit reports on progress, including any new unresolved issues.

Aristotle user accounts allow access to all Aristotle resources, so that inspecting the Aristotle dashboard allows an informed user decision to run on any particular location or resource at that location (for example, the GPU nodes at CAC). Allocations were initially given by an even division of available resources by number of research projects, to allow for user on-boarding and user testing, where researchers could request additional resources and re-balancing of allocations would be conducted. In full production, allocations are given by request via a lightweight allocation process allowing Aristotle Users to submit their request and the Aristotle Science Manager to approve or reject it via the User Portal.

3.2 Aristotle Science Cases

3.2.1 Weather Modeling with WRF. Weather modelling using the Weather Research and Forecasting Model (WRF) from the National Center for Atmospheric Research (NCAR) is a staple of supercomputer centers, and is used at a variety of scales. Although many applications require hundreds or even tens of compute cores, many researchers find considerable difficulty in installing WRF on local resources and are very concerned about portability of months-long, ongoing checkpointed runs from one installation to another. Consequently, a containerised WRF build based on NCAR's gives considerable portability and the ability to test the stability of runs even as they move across different hardware and clouds; meanwhile, the stability of Aristotle's cloud and the absence of a wall-time job limit ensures the jobs can be launched on single many-core instances and not monitored continually. Aristotle staff built a WRF container targeting single 28-core instances, where runs would last for several months, to model the direct climatic effects of turbines' meanwhile, this also allowed testing the portability of the code across clouds using Aristotle and a Jetstream allocation.

3.2.2 Radio Astronomy Transient Detection. Due to telescope scheduling, configuration and weather-related concerns, astronomy data are typically taken irregularly; the search for *transient* astronomical phenomena, in addition, typically comes with a requirement for low-latency processing to allow for potential follow-up observations. The processing of radio astronomy data to find transient phenomena such as Fast Radio Bursts (FRB), therefore, lends itself to a cloud-based solution; additionally, the breadth of software employed by the Scientists in this Science Use Case, and the relative difficulty in installing some of the software on processing resources, encouraged early adoption of software containers. The Aristotle team built an extensible software pipeline allowing astronomers, including undergraduate and graduate students, to write their own plug-ins in Python to explore new algorithms and computational approaches to teasing out astrophysical signals from a mass of Radio Frenquency Interference (RFI)

3.2.3 Water Resource Management using OPENMORDM Data. Water Management simulations using OPENMORDM allow users such as municipalities to test different approaches to inform decisions with considerable financial and environmental impact. These computations use the Message-Passing Interface (MPI) to marshal work on hundreds, thousands or more cores but with relatively low demand on communications latency, so as to require relatively rough parallelisation; the significance of this work, the desired portability of the code and the gentle demands on cross-worker communications make this an appealing case for a containerized cloud-based approach and this Science Use Case is testing the *scalability* of such an approach, compared to using physical cluster-based infrastructure. Working with the science researchers, Aristotle staff architected and produced a container-based virtual MPI cluster on Aristotle, the performance and scaling of which is currently being tested.

4 EXPERIENCES FROM CAC'S SUPPORT OF CLOUD COMPUTING EFFORTS

4.1 Bringing new use cases to cloud

The majority of our Red Cloud users previously computed on desktops, shared workstations, or clusters via batch schedulers. The transition to cloud computing for all starting points held a number of real and perceived hurdles. Few of our users had the system administration skills needed to build images; some found a group member with the required skills to keep costs down, and others funded the CAC team to provide the service. New issues arise in this area, such as understanding how to open ports for given security groups to provide access. Setting up ssh key pairs is likewise a new experience for many new-to-cloud users, despite some familiarity with ssh login. At this point, we are only providing the environment to create linux or windows instances; therefore some users also must learn to use a new-to-them operating system. After an image is built, the researchers must learn how to use the dashboard or user

interface effectively; the interface is not difficult to use, but learning new concepts and terminology can be a perceived barrier and is essential. We have frequently seen confusion over instance states and accidental use of compute time, resulting in using compute time without realizing it. Dusty decks persist through all compute shifts, including to cloud. While MPI may not be the best programming solution for a given algorithm on cloud, we have found people are reluctant to rewrite code for cloud computing.

The runtime of research code in the cloud is an important consideration; our users have been quite happy with the cloud experience even when the total runtime is lower, because the actual response time is often less on a cloud resource, due to not having to operate within the framework of a batch processing system or other queue. There tend to be very few issues when using cloud resources for loosely coupled, distributed computations, or computations where large-single nodes suffice, but as the chattiness of a distributed computation increases, scheduling and cluster architecture becomes an important factor in decreasing latency. This was an issue with the WRF use case, but despite their prior experience running on specialized systems such as Cray supercomputers, the researchers were in fact quite happy with the ability to keep several of our largest nodes running continuously, inspecting results every few weeks or months, rather than having to operate with a queue on a large shared resource.

4.2 Creating Containers for Researchers

A goal for many of our users has been to containerize code. This makes the resulting code more portable between clouds and VM image types, and has the added benefits of making the research code more maintainable in the long term and giving the researcher the ability to run the same environment on their development system and on whatever cloud or HPC resources they wish to target; though for HPC there is the caveat that Singularity must be the supported container platform. This is quite valuable in practice, since over the course of a research project, a PI may contracts or cloud credits with multiple clouds or on-site institutional clouds. Several of our users have expressed an interest in doing development and testing on Aristotle and then running larger "production" runs on public clouds. Another situation that may arise for multi-PI projects (especially those spanning institutions) is that each PI may have access to different cloud or HPC resources.

Container technology was not yet mainstream at the inception of the Aristotle project, so none of the existing researchers from the Cornell sites brought containerized code to the table. However, in the case of our WRF project, there was an existing container example from NCAR that served as a basis for our own WRF container. Other projects gradually adapted to using containerized code. There are two primary hurdles when it comes to containerization of software, other than the docker/Singularity dichotomy mentioned previously. These are user training and hosting container images with proprietary code.

The success of user training largely depends on the user involved, though even when a particular user or group may have difficulty embracing containerization concepts, it is still advantageous for CAC staff to have the containers to work with, greatly diminishing maintenance and deployment costs that often stem from project-specific build issues.

With regard to proprietary code in container images, we've had two strategies: mount from the VM, or private distribution of container images. The former option was ideal in the case of MATLAB, since it has a large footprint, generating and distributing container images would be slightly more unwieldy, but this admittedly results in an additional step that may go wrong. The other, more common approach, is simply to distribute the container images through private means. The downside here is that a bit more infrastructure is required, though most of the time, we simply build the image from the (publicly distributed) container recipe (e.g. Dockerfile, Singularity Recipe), and use scp or rsync to distribute the image to whatever nodes need the image; we generally do not worry about storing the image during the development process, as maintaining a version-controlled container specification is good enough. With regard to proprietary code in container images, we've had two strategies: mount from the VM, or private distribution of container images. The former option was ideal in the case of MATLAB, since it has a large footprint, generating and distributing container images would be slightly more unwieldy, but this admittedly results in an additional step that may go wrong. The other, more common appro

4.3 Transitioning to Public Cloud

Public cloud offers scale, and recognizability to researchers, but cost and technology adoption continues to be an issue. While the public cloud offers considerable benefits for a number of computational efforts, the experiences of researchers on public cloud tend to be varied. Many public clouds provide a number of services, which continues to grow, with minimal documentation for those who want to adopt those services. For the technologist who is running an enterprise, cloud service technology adoption is critical to the business. For researchers who are hoping to leverage technologies, science remains the critical component, and the information technology supporting it is a tool to support it. Training materials for public clouds tend to be sparse and frequently do not track the state of play of the services on offer. The fast-moving evolution of public cloud services also makes it difficult for external organizations to effectively develop training and documentation materials for public clouds.

In addition to complexities with adopting public cloud technologies, a number of cost factors confront researchers who do want to make the transition to these resources. Public clouds are designed to enable computation, regardless of the budget of the consumer. This means that researchers must identify their own ways of controlling costs and efficiently using services. Some efforts to leverage the public cloud spot market and predict probabilities of job completion based on pricing have been somewhat successful [8], but changing availability of data means that those that create predictive analytics for controlling cloud costs are in an arms race against the providers. In addition, data movement costs represent an additional component of cloud usage that can stymie researchers. In addition to costs to transfer data in and out of cloud resources, which tend to make data easier and cheaper to get in than it is to get out, various campuses have institutional agreements which lessen these

costs, depending on network configuration and the level of resource consumption. This means that researchers with different home institutions may be confronted with different costs for data egress, which makes cross-institutional collaboration additionally complex. Software licensing represents another complexity of public cloud consumption, while Microsoft offers "bring your own license" or software assurance-based programs, other software licensing policies remain in the bare-metal server world. Finally, institutional policies on intellectual property may be affected by public cloud terms and conditions, creating another hurdle to navigation.

4.4 Future activities

Docker [1] is, by far, the most prevalent container technology and is generally satisfactory for most cloud uses we are interested in, since the user also controls the VM the container is running in, many of the typical Docker security concerns are not an issue. Conversely, the Aristotle cloud has also been a satisfactory environment for many of our users, and bursting to larger public clouds could also be achieved with Docker. For two of our use cases (OpenMORDM [3] and WRF [6]) where running on the cloud and HPC platforms are both desirables, we do have plans to support Singularity [4], but Singularity does not fit our Docker model when dealing with MPI, and both of these use cases require MPI. In particular, Singularity supposes that MPI is external to the container (as it must be on a shared HPC resource), whereas this is not the case in our Docker containers. This Singularity assumption also introduces some build-time difficulties, as the resulting image must target a specific version and implementation of MPI. At CAC we have worked on making containers portable between Singularity and Docker by using common build scripts and continue to explore new ways to make the user experience of using Singularity in the cloud and on HPC more amenable to researchers and CAC staff.

In addition to containers, we have also investigated the use of declarative instance configuration using Nix [2]. Nix can also be used from within a container, or can provision and entire Linux OS. An example of the former use is the OpenMORDM (Waterpaths) container, and of the latter is the Metabolic Modeling use case, which also employs MATLAB. Our experience has been that Nix takes more effort to create the initial environment, but provides more deterministic builds with smaller footprints (no images needed, assuming stable repositories used for retrieving dependencies) and more flexibility: a container or a VM can be used, a shared system can also be used with built packages being pulled from a package store on disk (no image overlap), and the ability to override build options in package dependencies.

One of the future activities that is vital to researcher uptake is the understanding of the costs involved in making use of local and public cloud resources. The public cloud advertises that it is less expensive to complete tasks and more flexible in terms of the services on offer, but public cloud vendors have few incentives to be clear about what costs are incurred in the course of getting research done. The Aristotle team has engaged with a Flexera, a cost optimization firm, to use its RightScale product, which can provide insight into how costs are broken down and provide suggestions on strategies to reduce overall cloud costs. Using the work on containerized codes described above, the Aristotle team is exploring the use of pilot jobs that can be used to predict the cost of getting a job run in public cloud. By preparing a container and submitting it to be run in the public cloud, and analyzing the billing output created by the pilot job, the Aristotle team believes it can provide an estimator that a researcher can then use to decide whether using a public cloud resource is cost effective for her particular research.

This cost estimate can be incorporated into the Aristotle portal and provide an overall view of options available to the researcher. In a complete cost-projection page in the Aristotle portal, a researcher can see what resources are available through Aristotle, what it would cost to run the same job in Amazon, Azure, or Google, and what other resources they might be able to leverage (such as an allocation through another NSF program). This would give a researcher the flexibility to run for free on Aristotle to get regular work done, or if up against a deadline, to spend money and scale up on a public cloud provider.

5 CONCLUSION

In our description of the Red Cloud and Aristotle resources, we have detailed some of the activities involved in supporting researchers transitioning to and making full usage of cloud technologies for science. These resources represent considerable potential, and can be delivered at minimal costs, but researchers do need to change their means of interacting with compute resources and be prepared to engineer some of their analyses around the differences between cloud and traditional scientific computing practices. Human facilitators provide considerable benefit to researchers who want to make use of cloud resources in an efficient and useful manner, and the Aristotle project provided a significant amount of assistance to the science teams involved in order to support the transition to cloud analyses–with considerable benefits and potential for further develop to those science cases.

The complexity of cloud offerings, both on-premise campus cloud resources as well as public cloud offerings, means that human guides are indispensable to the process of adopting these technologies. In addition to making sure that applications run smoothly and quickly in the new context, researchers often require support in deciphering the pricing and costs of these technologies. The proliferation of offerings and rapid pace of change in services means that facilitators must dedicate effort to keeping up with technological change in order to ensure users have safe and efficient usage of these resources. Future work will need to focus on making it possible to leverage regulated data in the cloud and manage data retention and sharing agreements in an environment where providers can simply delete data when funds for retention run out. The cloud environment represents considerable potential to computational scientists but also requires adaptation and flexibility to make truly workable.

ACKNOWLEDGMENTS

This work is supported by NSF Award 1541215.

REFERENCES

[1] Ryan Chamberlain and Jennifer Schommer. 2014. Using Docker to Support Reproducible Research. https://doi.org/10.6084/m9.figshare.1101910.v1

[2] EELCO DOLSTRA, ANDRES LÃŰH, and NICOLAS PIERRON. 2010. NixOS: A purely functional Linux distribution. *Journal of Functional Programming* 20, 5-6 (2010), 577âĂŞ615. https://doi.org/10.1017/S0956796810000195

[3] David Hadka, Jonathan Herman, Patrick Reed, and Klaus Keller. 2015. An open source framework for many-objective robust decision making. *Environmental Modelling Software* 74 (2015), 114 – 129. https://doi.org/10.1016/j.envsoft.2015.07.014

[4] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE* 12, 5 (05 2017), 1–20. https://doi.org/10.1371/journal.pone.0177459

[5] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. 2009. The eucalyptus open-source cloud-computing system. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 124–131.

[6] Jordan G. Powers, Joseph B. Klemp, William C. Skamarock, Christopher A. Davis, Jimy Dudhia, David O. Gill, Janice L. Coen, David J. Gochis, Ravan Ahmadov, Steven E. Peckham, Georg A. Grell, John Michalakes, Samuel Trahan, Stanley G. Benjamin, Curtis R. Alexander, Geoffrey J. Dimego, Wei Wang, Craig S. Schwartz, Glen S. Romine, Zhiquan Liu, Chris Snyder, Fei Chen, Michael J. Barlage, Wei Yu, and Michael G. Duda. 2017. The Weather Research and Forecasting Model: Overview, System Efforts, and Future Directions. *Bulletin of the American Meteorological Society* 98, 8 (2017), 1717–1737. https://doi.org/10.1175/BAMS-D-15-00308.1 arXiv:https://doi.org/10.1175/BAMS-D-15-00308.1

[7] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. 2016. Globus Auth: A research identity and access management platform. In *2016 IEEE 12th International Conference on e-Science (e-Science)*. IEEE, 203–212.

[8] Rich Wolski and John Brevik. 2016. Providing statistical reliability guarantees in the aws spot tier. In *Proceedings of the 24th High Performance Computing Symposium*. Society for Computer Simulation International, 13.