Overcast: Running Controlled Experiments Spanning Research and Commercial Clouds

Paul Ruth
RENCI
pruth@renci.org

Kate Keahey
Argonne National Laboratory
keahey@anl.gov

Mert Cevik
RENCI
mcevik@renci.org

Zhuo Zhen
University of Chicago
zhenz@uchicago.edu

Cong Wang
RENCI
cwang@renci.org

Jason Anderson
University of Chicago
jasonanderson@uchicago.edu

***Abstract*—The Chameleon project developed a unique experimental testbed by adapting a mainstream cloud implementation to the needs of systems research community and thereby demonstrated that clouds can be configured to serve as a platform for this type research. More recently, the CloudBank project embarked on a mission of providing a conduit to commercial clouds for the systems research community that eliminates much of the complexity and some of the cost of using them for research. This creates an opportunity to explore running systems experiments in a combined setting, spanning both research and commercial clouds. In this paper, we present an extension to Chameleon for constructing controlled experiments across its resources and commercial clouds accessible via CloudBank, present a case study of an experiment running across such combined resources, and discuss the impact of using a combined research platform.**

***Index Terms*—research cloud, research testbeds, multi-cloud, networking**

I. Introduction

Computational experiments comprise a large spectrum of scientific computations, from large-scale simulations, to just-in-time analytics, or performance studies. Different types of such experiments place different requirements on the experimental container in which they execute from the perspective of isolation (shared, multi-tenant, or completely isolated environment), levels of access, configurability, interactivity, and performance. Within this spectrum, computer science systems experiments are perhaps the most challenging to provide for as many – though not all – often require high level of configurability and access (bare metal reconfigurability with specialized network configuration capabilities), strong performance isolation, interactive access, and the ability to experiment at high performance.

To adequately address this level of challenge, traditional testbeds that provided experimental capabilities for computer science research have generally been configured by technologies developed in-house [1], [2]. The Chameleon testbed [3] broke with this pattern by adapting a mainstream open source cloud technology, OpenStack, to provide similar capabilities. This implementation strategy carries a range of practical benefits – such as familiar interfaces for users and operators, or the opportunity to leverage contributions from a large development community – but it also creates the potential to contribute back, and thus influence the debate on the best cloud configuration to support computer science research. With commercial cloud providers, such as Amazon Web Services (AWS), increasingly willing to offer more flexibility in the form of e.g., bare metal instances, and NSF's investment in the CloudBank [4] initiative to make commercial clouds available to the systems research community, this debate is gaining in importance, and could result in significant broadening of the opportunities for computer science experimentation.

The ability to leverage commercial cloud resources could open up significant opportunities for research. With their greater geographic dispersion, greater diversity of resources, and greater scales – though perhaps lesser customization to research needs – the commercial clouds provide an interesting offering to supplement specialized testbeds like Chameleon. In [3] we articulate the specific extensions required to adapt clouds to the needs of systems research; in this paper we examine the question of what experiments could leverage both research and commercial clouds and how such experiments should be configured to control wide-area network performance to enable repeatably. Specifically, we describe our approach to constructing such an experiment in the context of the experimental workflow stages [5] of an experiment distributed over the Chameleon testbed and AWS cloud (via CloudBank) accessed via isolated layer-2 networks.

The specific contributions of the paper are as follows:

- We present Overcast, a recipe and a set of tools, expressed as a Jupyter notebook, that allows experimenters to construct a class of network isolated experiments that are distributed across research and commercial clouds.
- We describe an experiment case study using and experiment that leverages Overcast to evaluate isolated layer 2 network paths (AWS DirectConnect) spanning resources on Chameleon, Internet2 CloudConnect, and AWS; in particular, we show how VM instance types affect achievable bandwidth.
- We discuss insights obtained from using research and commercial clouds in conjunction, comparing their ca-

pabilities, in particular performance using non-Internet, wide-area network paths.

The rest of the paper is organized as follows. In Section 2 we describe tools and methods for constructing multi-cloud experiments. In Section 3, we show how those tools can be used to construct a non-trivial networking experiment. We follow by a discussion of cloud capabilities and cost, discuss related work, and conclude in Section 6.

II. Overcast: Deploying an Experiment over Multiple Clouds

Computer science experiments are typically enacted in stages [5] beginning with experiment design, identifying suitable resources, allocating and configuring them, running the experiment itself, and finally analysis. We analyze these stages below, and compare and contrast approaches used for experiment development on Chameleon and commercial resources available via CloudBank.

Both Chameleon and CloudBank provide user access via federated login – Chameleon via Globus Auth [6], CloudBank via CILogon [7] – so that once logged into one system a user can use the other without entering a password again. To use system resources, a user needs to be associated with a project that has active allocations. In Chameleon those allocations consist of Service Units (SUs) which represent one hour of wall clock time on a mainstream server. To request an allocation, users are first certified for PI eligibility according to infrastructure-specific policies [8], [9]. In Chameleon they can then propose a project to be awarded an allocation. In CloudBank, verified PIs are allocated pre-determined funds which represent direct funding for a specific project to be spent on one or more public clouds. Within each system, the PI or their delegate can manage allocations to track spending and grant membership in the project to others.

In the first step of the experimental workflow, users can browse resources available for experimentation but their descriptions differ among the systems. Chameleon resource descriptions are fine-grained so that users can browse the exact node configurations including processor types, cache hierarchies, I/O device types, rack placement, etc. Commercial providers typically represent their resources as "instance types" (see e.g., [10]) and provide general information about vCPUs/CPUs, memory, storage, network, and much less detailed information about processors. Further, information about resources in Chameleon is being kept rigorously up-to-date – component upgrades might affect experiments sensitive to hardware such as e.g., power management or performance variability – the testbed is then versioned so that experimenters can verify this information at a glance and users are offered a suite of verification tools for sanity checks. Commercial clouds do not track the hardware evolution and do not provide such versioning though Chameleon's verification tools, based on standard Linux commands (such as e.g., "biosdecode" and "dmidecode" to get BIOS settings) can be used to mitigate these shortcomings to some extent.

Once selected, resources can be allocated. A notable difference between Chameleon and commercial resources is the support for advance reservations which allow Chameleon users to reserve availability of resources ranging from nodes, to networks, and IP addresses for a specific time in the future [11]. In contrast, commercial clouds rely primarily on on-demand availability; vehicles such as "reserved instances" [12]represent a billing discount, while "capacity reservations" based on an up-front payment that reserve capacity for specific type of instance [13]work in a similar way to allocation on Chameleon. Another significant difference is how resources are described: Chameleon allows users to specify resources at different levels: from model-level descriptions (e.g., "I need four nodes with at least 2GB per core") to indicating a specific node in the system, essential for experiments that require control of hardware variability. In commercial clouds the only way to describe resources to be allocated is the instance type; while this provides a high-level description, it might map to different types of resources, without the user being able to control the mapping [3].

In both Chameleon and commercial clouds allocated resources are configured by deploying disk images to create bare metal or virtual machine instances; since many users will want to use consistent configuration across a distributed experiment it is useful to consider how portable those images are. Most cloud platforms have specific requirements for the format of a disk image (e.g., RAW or QCOW2 [14]), its disk layout (e.g., whole disk or partition image), or the environment included in the image (e.g., cloud-init for injecting SSH keys or a DHCP client configured on specific interfaces) making images incompatible between various providers. They all however support the same general structure (e.g., the SSH key injection pattern via cloud-init) that can be leveraged by tools, such as OpenStack qemu-image convert [15] and AWS VM Import/Export [16] to convert between images; thus, to use a Chameleon image on AWS, it has to be converted to RAW format (if not RAW already) and then converted to AMI [17] using AWS VM Import. This common structure and the resulting portability options are an important consequence of the decision to configure a research testbed as a cloud [3]. The fact that Chameleon is not only a cloud, but an OpenStack cloud in particular, facilitates things even further: for example, the metadata service in OpenStack supports EC2-compatible API [18] which means that the images designed for EC2 will work with OpenStack directly. That said, some aspects of portability may be complicated by issues higher in the stack: for example, Chameleon includes utilities for system verification and snapshotting on its base images that will not be present on images imported from AWS. We provide a more general discussion of image conversion tools, including tools like Packer [19], in [20].

Orchestration, which allows users to deploy configurations consisting of multiple interdependent images , networks, and other cloud services automatically, is a related issue and provides a further demonstration of the advantages of configuring a CS research testbed as a cloud. OpenStack Heat [21] and

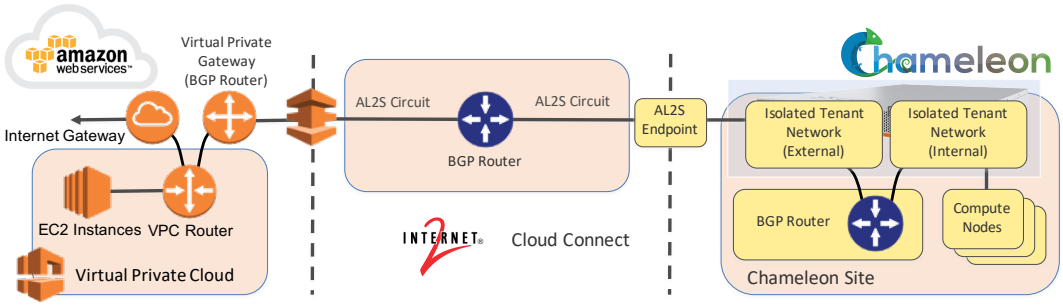


Fig. 1: Chameleon connected to AWS using Internet2 CloudConnect

CloudFormation [22], used for orchestration in Chameleon and AWS respectively, both use declarative languages (YAML or JSON) to define the desired configuration template. In fact, OpenStack Heat provides direct compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack [23] (see [24] for exceptions) though not vice versa. While orchestration templates still represent the most popular orchestration tool, their transactional and declarative nature makes them inflexible for experimentation. For this reason, the Chameleon project integrated Jupyter notebooks to provide an imperative-style alternative. A similar trend is followed by commercial clouds, e.g., the AWS Cloud Development Kit (CDK) [25] allows users to define a AWS infrastructure using a programming language.

Network configuration is a critical element of any experiment spanning research and commercial clouds. The traditional option is to assign domain specific public IP addresses to all nodes and rely in the Internet for wide-area communication. This simple approach suffers from limitations to security, performance, and repeatably caused by the open and uncontrolled public Internet. Another option used in CloudJoin [26] is to use a virtual private network (VPN) as a tunnel between distributed sites; the VPN protects the architecture from common security attacks and allows remote cloud resources to be assigned local IP addresses and managed as if they were on-site. Both of these options have the advantage of easy implementation, but are limited by the inconsistent performance of the public Internet, an important consideration for many experiments.

To better control the affect of wide-area network, experimenters should utilize direct low-level network connections between the research and public clouds, however creating them is challenging for an independent researcher. While most public clouds provide low-level networking services (e.g. AWS Direct Connect [27], Azure ExpressRoute [28], or Google Dedicated Interconnect [29]), using them is typically expensive [30]; on the research cloud side, they can involve complicated campus network configuration arrangements that often limit access to this type of experimental configuration to a few a few select scientists or campus IT staff themselves.

Since 2016, the Chameleon testbed has provided direct connect using Internet2's Advanced Layer 2 service (AL2S) [31] via ExoGENI [32]. More recently, Internet2 has deployed its CloudConnect service [33] that enables members to connect AL2S end points, such as those used for Chameleon direct connections, to AWS Direct Connect, Azure ExpressRoute, and Google Dedicated Interconnect sites. Thus, to create an experimental topology between Chameleon and commercial cloud the first step is to create a direct connection between Chameleon and a public cloud accessible using Internet2's CloudConnect. Additionally, since public cloud direct connections configure routing between the cloud and external facility using BGP, a user also needs to deploy a BGP router on their resources. To implement that, we developed a Jupyter notebook that deploys fully configured BGP routers. Further, the BGP router can, optionally, be deployed on a dedicated OpenFlow networking switch or as software Quagga router existing on a standard x86 compute host. The full networking configuration is depicted in Figure 1.

III. An Experiment Case Study: DirectConnect

This section describes a multi-cloud experiment using Chameleon and AWS resources connected using dedicated circuits provided through Internet2's CloudConnect service. The aim of the experiment is to assess the bandwidth spanning these clouds. We also explore to what extent AWS instance types achieve targeted network performance. The experimental setup is packaged as a set of Jupyter notebooks available as a Chameleon Trovi artifact [34]

A. Setup

The configuration of the experiment can be seen in Figure 1. A BGP router was deployed on a Chameleon host connected to two dedicated 10 Gbps tenant networks. One was an externally connected network that was stitched to an Internet2 CloudConnect BGP router. The other was an internal network connected to other compute nodes on Chameleon. On the AWS side, the CloudConnect BGP router was connected to a Virtual Private Gateway (VPG). The router in a Virtual Private Cloud (VPC) was configured with default routes to a private Internet Gateway and custom routes through the dedicated Internet2 circuit to the isolated tenant network on Chameleon. The three BGP routers cooperate to advertise routes between user-controlled subnets hosted on Chameleon and AWS. The infrastructure's configuration is described in the Chameleon tutorial on using Internet2 CloudConnect.

TABLE I. EC2 Instance Type and the Bandwidth Limit

EC2 Instance Type	Description	AWS Network Limit
t2.micro	1 CPU, 1 GB mem (free)	Low to Moderate
t2.2xlarge	8 CPU, 32 GB mem	Moderate
t3.2xlarge	8 CPU, 32 GB mem	5 Gbps
m5a.2xlarge	8 CPU, 32 GB mem	10 Gbps
m5dn.2xlarge	8 CPU, 32 GB mem	Up to 25 Gbps
c5n.metal	72 CPU, 196 GB mem	100 Gbps

TABLE II. Host Network Tuning Parameters

Setting	Value
net.core.netdev_max_backlog	250000
net.core.rmem_max	67108864
net.core.wmem_max	67108864
net.ipv4.tcp_congestion_control	htcp
net.ipv4.tcp_rmem	4096 87380 33554432
net.ipv4.tcp_wmem	4096 65536 33554432
net.ipv4.tcp_mtu_probing	1
net.core.netdev_budget	600
net.core.default_qdisc	fq
interface MTU	9000
interface txqueuelen	10000

The Chameleon resources were located at the University of Chicago site and the AWS resources were from the us-east-2a region in Ohio. On Chameleon, the end hosts were baremetal x86 (Haswell) servers with 24 compute cores, 132 GB of RAM, and 10 Gbps network. Six different AWS instance types were evaluated. Each instance used the imported CentOS7 image and was hosted on a VM or bare metal server with the hardware properties and stated networking bandwidth limitations in Table I. The AWS stated networking limitations ranged from "Low to Moderate" to "100 Gbps". The CloudConnect circuit limited the maximum bandwidth to 5 Gbps. To configure Chameleon nodes, we used the Centos7 image provided by the testbed. To achieve a consistent configuration on AWS, the same image was converted from QCOW2 to RAW type, then uploaded to AWS S3, and then converted to AMI using AWS VM Import using one of the methods described in Section II. All hosts were tuned by using ESnet's recommendations [35] for large wide area data transfers as in Table II.

B. Experiment

The experiment used iperf3 [36] to test the TCP bandwidth achievable between the baremetal host on Chameleon and each of the targeted AWS instance types over the deployed direct connect circuit. Each instance type was tested using 30 second tests with both single and multiple streams in each direction. Each test was executed 15 times. The average, minimum, and maximum egress and ingress bandwidth are shown in Figure 2.

C. Discussion

Other studies have shown that achieving repeatable networking results on public clouds is difficult or even impossible [37]. Although using direct connections eliminates some of the unpredictability of the public Internet it does not change the way public clouds control network performance.

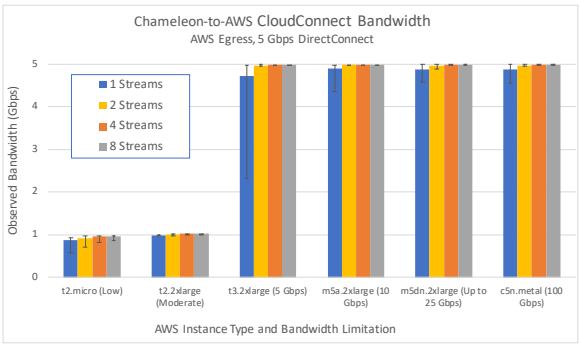
Figure 2 shows the bandwidth achieved between Chameleon and the AWS instances. Each group of columns shows the bandwidth achieved by an AWS instance type. The smallest two instance types were described by AWS as having 'low' and 'moderate' bandwidth limitations. In the experiments, these limitations were effectively 1 Gbps in either direction. The 'low' bandwidth instance achieved less consistent performance and appeared to be the subject of periodic rate limiting. Continuous transfers with the 'low' bandwidth node would result in extremely low rate limiting of approximately 65 Mbps. Each iperf3 test was executed for 30 seconds. In order to avoid artificial rate limiting, the tests were performed 5 minutes apart. Applications requiring continuous 1 Gbps performance should opt for at least 'moderate' network performance.

The CloudConnect circuit across Internet2 was limited to 5 Gbps and the remainder of the instance types should have been able to fully utilize the circuit. Their AWS network limitations ranged from 5 Gbps to 100 Gbps. The experiment attempted to maximize the bandwidth using 1, 2, 4, and 8 parallel network streams. Ideally, the the maximum bandwidth could be achieved with a single TCP stream. However, wide area TCP data transfers are adversely affected by small numbers of dropped packets. The data shows that all instance types achieved network performance near the maximum 5 Gbps. However, the single stream performance increased for higher bandwidth instance types even though the instance's network limitation was well beyond the CloudConnect circuit. Further, the m5dn.2xlarge virtual machine performed as well as the c5n.metal node for egress and slightly better for ingress even though the bare metal node had more memory, more compute cores, and a higher bandwidth limitation.

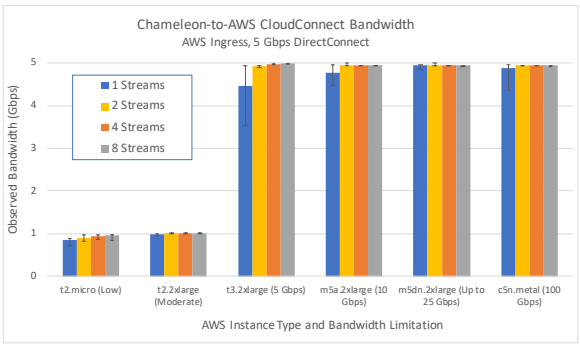
The reason that a single stream could not achieve the maximum bandwidth on virtual instances that had greater than or equal to 5 Gbps networks was that the DirectConnect configuration studied resulted in more dropped packets for instances with lower network limitations. The dropped packets occurred in both directions and were typically seen in small bursts. As a result, maximum AWS egress and ingress bandwidth could only be achieved with multiple TCP streams or on instances with over provisioned networks. In addition, the average single stream egress bandwidth is higher than the ingress bandwidth for each instance type but the minimum single stream ingress bandwidth is higher than the minimum egress bandwidth. This means that single stream ingress flows are more often affected by small bursts of dropped packets while single stream egress flows are more likely to experience large bursts of dropped packets.

IV. Discussion

Combining commercial and research clouds significantly expands the resource portfolio available to science, increasing opportunities to build more complex experimental topologies, deploy them on more sites, provide access to more diversity, as well as more scale. While bare metal offerings are few, information about resources and their evolution is not always



(a) AWS Egress



(b) AWS Ingress

Fig. 2: Chameleon-to-AWS CloudConnect bandwidth for various instance types.

available, and the user typically has no control over the specific architecture their experiment will map to, not all experiments require these features as exemplified by our case study.

Overall our experiences configuring an experiment spanning Chameleon and commercial clouds available via CloudBank demonstrated that configuring a research testbed as a cloud brought the unexpected benefit of improved portability. While digital artifacts the users produce, such as images and orchestration templates, are not always directly compatible (though we noted ready-made conversion tools for some scenarios), they represent a similar structure, captured by such tools as e.g., use of cloud-init for SSH key injection. Converting them to achieve consistent configuration on experiment resources deployed on Chameleon and AWS is thus a relatively simple matter of updating well-known qualities rather than rebuilding them from scratch. In addition to convenience, this also addresses an issue of consistency and ultimately also reproducibility of how exactly a configuration can be or was repeated in a different setting. Similarly, configuring networking relied on the existence of similar concepts at edge on both ends of the connection

Another issue distinguishing research and commercial clouds is the cost to the scientific community. To create a rough comparison of cost we first paired the resource types available on Chameleon with AWS instances that are comparable or less powerful to create a conservative estimate. For example, our Haswell compute nodes [38] were paired with "c5d.metal" – but our GPU P100 nodes were paired with "p3.8xlarge" even though "p3.8xlarge" are not bare metal as AWS does not offer comparable bare metal resources. We then took actual usage numbers for each Chameleon resource type, computed the number of hours they were used over the first 5 years of the project, and multiplied it by the price of the corresponding AWS resource. The total number of node hours over all the resource types were 5,676,114, representing an estimated cost of $30,353,133 ($49,172,075 including overhead which would normally be applied to this type of purchase though is waived through CloudBank usage). In comparison, the total funding received for Chameleon over this period of time is projected to be $16.6M. Further, our rough estimate did not include the Chameleon KVM cloud, data download or storage, any of the special features or specialized services (like BYOC [39] used in the experiment), or startup costs, all of which would have made the Chameleon resources even more cost-effective. On the other hand, Chameleon provides significantly lesser availability than commercial clouds (a factor partially mitigated by advance reservations described in Section II) and probably lesser reliability; while those are of course desirable to the research application, increasing them would also increase the cost. Thus, this is another area where research and commercial clouds provide different offerings at a different cost.

V. Related Work

Investigating methods of building an integrated environment across multiple clouds is almost as old as cloud computing itself [40]. In particular, works like [20], [41], [42] investigate issues of resource management, image portability, and orchestration; our focus is different in that we investigate this issue from the perspective of providing an experimental container for computer science systems research which among others includes management of low-level networking resources, and consider the issue of cost.

Multiple projects also sought to characterize the networking performance of research versus commercial clouds [43], [44]. Our approach pushes this line of investigation further by focusing on low-level networking services like DirectConnect and shows a measure of similarity in the problems.

VI. Conclusions

In this paper, we present a recipe and a set of tools, expressed as a Jupyter notebook, that allows experimenters to construct a class of experiments distributed over research and commercial clouds, specifically the Chameleon testbed and commercial clouds available via the CloudBank project. In doing so, we discuss the different capabilities of these two different types of clouds and show how they can be used to construct experiments spanning both.

Our observations show that the decision to configure Chameleon as an enhanced cloud pays dividends in the context

of such experiments. Specifically, similar structure of disk images and orchestration templates used in experiments of this type facilitates portability and maintaining a consistent experimental environment. Further, similarities between the ExoGENI stitchport and direct connects implemented by commercial clouds allow experimenters to leverage the same concepts when constructing wide-area circuits.

ACKNOWLEDGMENT

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation and AWS credits provided by NSF CloudBank. This work was also partially supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.

REFERENCES

[1] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, et al. Adding Virtualization Capabilities to the Grid'5000 Testbed. In I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.

[2] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1–14, 2019.

[3] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S Gunawi, Cody Hammock, et al. Lessons Learned from the Chameleon Testbed. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 219–233, 2020.

[4] CloudBank. https://www.cloudbank.org/.

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

[6] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. Globus Auth: A Research Identity and Access Management Platform. In *2016 IEEE 12th International Conference on e-Science (e-Science)*, pages 203–212. IEEE, 2016.

[7] Jim Basney, Heather Flanagan, Terry Fleury, Jeff Gaynor, Scott Koranda, and Benn Oshrin. Cilogon: Enabling Federated Identity and Access Management for Scientific Collaborations. *Proceedings of Science*, 351:031, 2019.

[8] Chameleon PI Elibibility. https://chameleoncloud.readthedocs.io/en/latest/getting-started/pi_eligibility.html.

[9] CloudBank Managing Cloud Funds and Billing Accounts. https://www.cloudbank.org/training/managing-cloud-funds-and-billing-accounts.

[10] Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/.

[11] Kate Keahey, Pierre Riteau, Jason Anderson, and Zhuo Zhen. Managing Allocatable Resources. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 41–49. IEEE, 2019.

[12] Amazon EC2 Reserved Instances. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-reserved-instances.html.

[13] Amazon EC2 On-Demand Capacity Reservations. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-capacity-reservations.html.

[14] QEMU Images. https://en.wikibooks.org/wiki/QEMU/Images.

[15] The qemu-img Tool. https://docs.openstack.org/image-guide/convert-images.html.

[16] Amazon EC2 VM Import/Export. https://aws.amazon.com/ec2/vm-import/.

[17] Amazon Machine Image. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html.

[18] OpenStack EC2-compatible Metadata. https://docs.openstack.org/nova/latest/user/metadata.htmlmetadata-ec2-format.

[19] HashiCorp Packer. https://www.packer.io/.

[20] Kate Keahey, Pierre Riteau, and Nicholas P Timkovich. Lambdalink: an Operation Management Platform for Multi-cloud Environments. In *Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 39–46, 2017.

[21] OpenStack Heat. https://docs.openstack.org/heat/latest/.

[22] AWS CloudFormation. https://aws.amazon.com/cloudformation/.

[23] OpenStack Wiki Heat. https://wiki.openstack.org/wiki/Heat.

[24] OpenStack CloudFormation Compatible Resource Types. https://docs.openstack.org/heat/rocky/template$_guide/cfn.html$.

[25] AWS Cloud Development Kit. https://docs.aws.amazon.com/cdk/latest/guide/home.html.

[26] Jack Brassil and Irene Kopaliani. Cloudjoin: Experimenting at scale with hybrid cloud computing. In *2020 IEEE 3rd 5G World Forum (5GWF)*, pages 467–472, 2020.

[27] AWS Direct Connect. https://aws.amazon.com/directconnect/.

[28] Azure ExpressRoute. https://azure.microsoft.com/en-us/services/expressroute/.

[29] Google Cloud Dedicated interconnect. https://cloud.google.com/network-connectivity/docs/interconnect/concepts/dedicated-overview.

[30] AWS Direct Connect Pricing. https://aws.amazon.com/directconnect/pricing/.

[31] Internet2 Layer 2 Services. https://www.internet2.edu/products-services/advanced-networking/layer-2-services/.

[32] Rick McGeer, Mark Berman, Chip Elliott, and Robert Ricci. *The GENI Book*. Springer, 2016.

[33] Internet2 Networking for Cloud. https://www.internet2.edu/products-services/advanced-networking/networking-for-cloud/.

[34] Paul Ruth. Overcast - aws cloudconnect, January 2021.

[35] ESNet Linux Tuning. http://fasterdata.es.net/host-tuning/linux/.

[36] iperf3. https://software.es.net/iperf/.

[37] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan S. Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. Is big data performance reproducible in modern cloud networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 513–527, 2019.

[38] Per Hammarlund, Alberto J Martinez, Atiq A Bajwa, David L Hill, Erik Hallnor, Hong Jiang, Martin Dixon, Michael Derr, Mikal Hunsaker, Rajesh Kumar, et al. Haswell: The Fourth-generation Intel Core Processor. *IEEE Micro*, 34(2):6–20, 2014.

[39] Mert Cevik, Paul Ruth, Kate Keahey, and Pierre Riteau. Wide-area Software Defined Networking Experiments using Chameleon. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 811–816. IEEE, 2019.

[40] Katarzyna Keahey, Mauricio Tsugawa, Andrea Matsunaga, and Jose Fortes. Sky Computing. *IEEE Internet Computing*, 13(5):43–51, 2009.

[41] Kate Keahey, Patrick Armstrong, John Bresnahan, David LaBissoniere, and Pierre Riteau. Infrastructure Outsourcing in Multi-cloud Environment. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, pages 33–38, 2012.

[42] Gregor Von Laszewski, Fugang Wang, Hyungro Lee, Heng Chen, and Geoffrey C Fox. Accessing Multiple Clouds with Cloudmesh. In *Proceedings of the 2014 ACM international workshop on Software-defined ecosystems*, pages 21–28, 2014.

[43] Devarshi Ghoshal, Richard Shane Canon, and Lavanya Ramakrishnan. I/O Performance of Virtualized Cloud Environments. In *Proceedings of the second international workshop on Data intensive computing in the clouds*, pages 71–80, 2011.

[44] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. Is Big Data Performance Reproducible in Modern Cloud Networks? In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 513–527, 2020.