CloudJoin: Experimenting at scale with Hybrid Cloud Computing

Jack Brassil
Dept. of Computer Science and Office of Information Technology
Princeton University
Princeton, NJ, USA
jbrassil@princeton.edu

Irene Kopaliani
Office of Information Technology
Princeton University
Princeton, NJ, USA
ik8@princeton.edu

***Abstract*—To continue innovating in an age of at-scale computer systems research, the academic computing and networking systems research community must explore new approaches to addressing growing researcher demands to support larger size experiments. CloudJoin explores a transformational approach to scaling out successful Computing Research Infrastructures (CRI) into larger testbeds by creating hybrid cloud computing systems. We describe how to create a seamless, scalable, single experiment testbed that spans *CloudLab* and the *Google Cloud Platform* (GCP), while requiring no infrastructure changes. In addition to added elastic computing capacity, CloudJoin experiments benefit from easy access specialized hardware and cloud services and APIs to leverage world class data analytics and experimental infrastructure monitoring. In this work-in-progress, we show how to integrate the infrastructures by creating a Virtual Private Network between a CloudLab experiment and a GCP Virtual Private Cloud (VPC). To simplify understanding of large-scale experiment behavior, problem diagnosing and debugging, we also demonstrate how to use scalable, single dashboard cloud monitoring and logging tools across the hybrid testbed infrastructure.**[1]

***Index Terms*—experimental testbeds, hybrid cloud computing, CloudLab, GCP**

I. Introduction

Advances in large-scale data center computing and networking has accelerated innovation across computer science during the past decade or longer. To support continued advances, operators of Computing Research Infrastructures (CRI) such as *Chameleon Cloud* [1] and *JetStream* [2] have been pressed with demands to support larger size experiments and diverse hardware components such as GPUs and programmable switches.

While public compute cloud platforms have grown exponentially in size, research infrastructures have grown modestly. All but a few academic researchers with privileged access to commercial platforms can't experiment with comparable scale research infrastructures. Systems researchers are increasingly suffering a computing research infrastructure *scale gap*. Unfortunately, this gap seems likely to grow over time, threatening to further inhibit academic researchers from innovating in large-scale computing systems research.

[1]This material is based upon work supported by the National Science Foundation under Grant No. CNS-1923692

CloudJoin seeks to develop a long-term, sustainable approach to shrink the scaling gap. We examine innovative ways to design and deploy CRIs that mirror approaches used in *hybrid clouds*. Using *CloudLab* [3] as a base system, CloudJoin creates a new hybrid, elastic research infrastructure by uniting CloudLab with GCP. We are developing methods, tools, and best practices for deeply integrating these systems across the data, control and management planes. CloudJoin seeks to scale experiments at low cost while striving to maintain experiment isolation and reproducibility, and preserving a rendezvous point and an intellectual 'home' infrastructure for experimenters.

This paper focuses on 2 questions that arise in creating a hybrid cloud to support large-scale experimentation. First, how should a CRI operator or individual experimenter architect and implement her infrastructure's connectivity to a public cloud? A key contribution of this paper is to help both CRI developers and experimenters understand the choices and tradeoffs associated with constructing an experiment that spans both a public cloud and a research testbed. The second focus recognizes that CRIs often provide few powerful tools to help researchers understand and debug experiment behavior. The question we address is how to improve a user's understanding of a deployed experiment, which is potentially much more challenging in a hybrid setting. Our second key contribution is to demonstrate that experiment operation and behavior can be better understood by embracing professional quality, multi-cloud management platforms equipped with monitoring, logging, alerting, error reporting and profiling capabilities. Finally, we make all the software to perform these functions available as open source on GitHub and on the CloudLab system.

II. Creating Large Scale Experimental Infrastructures

CloudJoin seeks to permit *experimenters* to scale investigations at low cost while preserving and nurturing an intellectual community. CloudJoin also permits CRI *operators* to 1) preserve legacy CRI capital investments, 2) maintain operational control of infrastructures, and 3) retain institutional control of critical infrastructure metadata. Finally, CloudJoin helps sustain community infrastructure while nurturing an

environment that facilitates both research collaborations and experiment reproducibility.

A hybrid cloud approach has many potential advantages, including:

1) *scale-out* – A hybrid environment experiment permits on-demand resource acquisition at run-time. If a CloudLab experiment needs hundreds of VM instances that are not locally available, the experiment will be seamlessly extended to the cloud where these instances will be created.
2) *low cost* – The need for computing resources changes over the duration of a typical experiment lifecycle. A hybrid environment permits experiments to be developed and tested using CloudLab, minimizing use of pay-for-use cloud resources until required.
3) *specialty computing resources* – Systems researchers occasionally seek to evaluate less common hardware or software technologies without making a large initial capital or time investment. CloudJoin can provide on-demand access to nodes with Google TPUs [4]) or other specialty processors [5]. Software resources such as Database or Machine Learning stacks can also be selectively deployed.
4) *coordinated control and workflow management* – Systems researchers using both CloudLab and public cloud tend to do so in a awkward, unintegrated fashion [6]. A possible explanation for this behavior is the lack of integrated tools, with few exceptions such as the Google Collaboratory [7]. Hybrid solutions enable integrated management and control of an experiment spanning the environments.
5) *location flexibility* – CRI experimenters have limited control of where an experiment runs. But placement of data and compute is of growing importance, for example, in CyberPhysical systems experiments that are envisioned to operate in an 'edge computing' environment. Hybrid systems exploit cloud computing's international footprint to allow an experimenter to place computation relatively flexibly around the globe.
6) *reproducibility/knowledge sharing* – In contrast to commercial clouds, CRIs support effective multi-institution collaboration by emphasizing easy sharing of profiles, data, and knowledge. CloudJoin can provide a "shared home base" in a CRI to help researchers more easily form communities, supporting sharing of knowledge and aiding the reproducibility of experiments.

While a hybrid cloud approach potentially offers compelling technical benefits, it does introduce certain disadvantages and complexities for experimenters. For example, many research community members have limited experience using commercial cloud systems for experimentation. Some academic users report challenges associated with procuring cloud services under existing university and grant funding rules. Fortunately a number of these issues are currently being explored and remedied by the community [6].

A. Alternative approaches for Scalability

We acknowledge that experimenters can and do have platform options for large-scale experimentation outside of hybrid clouds. Public-private partnership initiatives [8], [9] support academic-industry exchange, increased access to industry-based experimental facilities, and opportunities to construct large infrastructures. CRI operators have also responded to demands to serve larger size experiments with both infrastructure growth and innovations in resource reservations and scheduling to share limited resources between experimenters. At the same time large commercial entities such as FaceBook, Apple, Amazon, Netflix, and Google have developed private, hyper-scale infrastructures that have served as both proprietary production service platforms *and* novel research instruments. The growth of commercial cloud resources is accelerating. We note that Google alone has recently made annual investments in data center capital equipment of roughly US $20B [10].

Academic systems research groups have responded to this scale gap with a collection of ad hoc strategies and tactics, including:

1) selecting research problems that can be studied at relatively modest scale;
2) collaborating with commercial entities to access their large proprietary infrastructures [11]; and
3) shifting research investigations and workloads entirely to commercial compute cloud platforms [12].

Research infrastructure *federation* also represents a path to supporting large experiments [13], [14]. A federated infrastructure is a collection of research computing or networking providers with agreed standards of interoperability to permit shared goals such as resource sharing, access control, or common user experience. While scaling experiments through federation can be achieved – Open Science Grid [15] is an exemplar – the technical complexities and policy constraints have made federation a challenging approach to experiment scaling.

III. Hybrid Cloud Testbed Design Tradeoffs

Hybrid experimental architectures can be architected in different ways. For example, a CRI can integrate with public cloud infrastructure at 1) an *infrastructure-wide level*, such as by a direct network connection of the entire CRI to cloud; and 2) an individual *experiment level*, as when adding cloud-based xPU nodes on demand to a single experiment at its runtime. The former type of integration benefits all experimenters, while the latter provides experiment-local benefits. In this paper we restrict our scope to experiment-local solutions, as they can help begin exploring hybrid systems while making no change to existing CRI platforms.

Deciding among feasible and cost-effective networking approaches is complicated – among the considerations that come into play are:

1) the need to offer networking quality-of-service guarantees (bandwidth, latency, connection setup time, etc) to experimenters;
2) cloud ingress/egress and inter-datacenter data transfer costs;
3) the fraction of experimenters requiring CloudJoin services; and
4) the required experiment isolation between CRI users.

Network connectivity can be established at an infrastructure- or experiment-level. Infrastructure-level options are CRI-specific; for instance, options depend on whether the CRI host institution can provide network peering to the chosen cloud provider. A host institution able to meet the cloud provider's peering requirements could consider either direct or carrier peering.

Experiment-level connectivity is generally simpler and appropriate for CRI supporting connectivity for fewer, less demanding experiments. A Virtual Private Network (VPN) based on IPsec tunnels over the public internet might be adequate to initially explore CRI-cloud connectivity. A particular advantage of this approach is that VPNs can be dynamically constructed on-demand within an experiment, where infrastructure-level approaches require longer setup times. Though VPNs generally support lower bandwidth connectivity, they can represent a step in a CRI's evolution toward higher performance networking solutions.

Maintaining experiment isolation is another consideration when evaluating connectivity. Services such as Virtual Private Cloud (VPC) can allow experiments (or groups of experiments) to share a single network connection to the cloud, including to resources in different geographic regions. This approach can offer groups of CRI users additional forms of experiment protection over more narrowly focused segregation techniques such as VLAN-based isolation.

IV. Constructing your CloudJoin Experiment

A CloudJoin experiment data plane spans CloudLab and GCP; simply using Google services and APIs from a CloudLab compute node is a corner case of considerable value but not considered here. A spanning infrastructure requires the experimenter to be a 1) owner or member of a CloudLab project, and 2) an authorized owner or user of a GCP account. A crucial decision by the experimenter – to be discussed in Sect. IV-A is to decide what experiment infrastructure resources should be instantiated on which "side". The order of instantiation of the different sides is irrelevant, though coordination is required for seamless communications. GCP resources can be started and stopped at fine grain quickly, so it is preferable to keep the GCP side – or at least those difficult-to-reconfigure components incurring costs – reserved but stopped when not in use.

An experiment *Manager* is used to bootstrap a CloudJoin experiment. The manager function connects to the GCP console or CloudLab portal to initiate an experiment. The manager can initially be located on the experimenter's desktop, and migrate if desired to a host within the CloudJoin experiment. It is helpful to have programmatic control (e.g., google-cloud-sdk, geni-lib) capabilities installed on the manager machine. Authorization and authentication are manager functions.

The GCP side comprises 1) experiment resources (e.g., instances, Cloud storage); 2) a Virtual Private Cloud (VPC); and 3) a Cloud Virtual Private Network (VPN) gateway with encrypted tunnel connected to a peer CloudLab VPN gateway. Additional cloud capabilities can be optionally added, or cloud service APIs invoked, as needed by the experiment.

Experiment infrastructure resources are instantiated as needed by the experiment. Each compute instance has a NIC with public IP address primarily used for control traffic, a second NIC with an address on the VPC zone's subnet (i.e., the experiment dataplane), and perhaps one additional NIC on a local private (zone) subnet for purposes such as local coordination or control traffic.

A VPC establishes a seamless interconnected private network connecting cloud resources on subnets spanning cloud provider regions or zones (even globally); routing is automatically handled by a virtual cloud router. GCP side resources can be restarted at experiment runtime.

Cloud VPN gateways instantiate one or more tunnels to CloudLab. The experiment's CloudLab data plane subnet addresses are advertised by the virtual cloud router, so packets generated on the VPC are forwarded via the tunnels. Options for bonded high-availability tunnels exist.

The CloudLab side comprises 1) compute, storage and network resources (e.g., bare metal nodes, mounted /project) as requested by the experimenter, organized as usual with separate data and control plane interfaces, and 2) a node or VM dedicated to serving as a VPN gateway. A routable public IP address for each VM is desirable for management and control. We have created a pair of public images (cloudjoin.vpn, cloudjoin.vm) on CloudLab to ease creation of CloudJoin experiments. The first is a Ubuntu 18.04 image used for bare

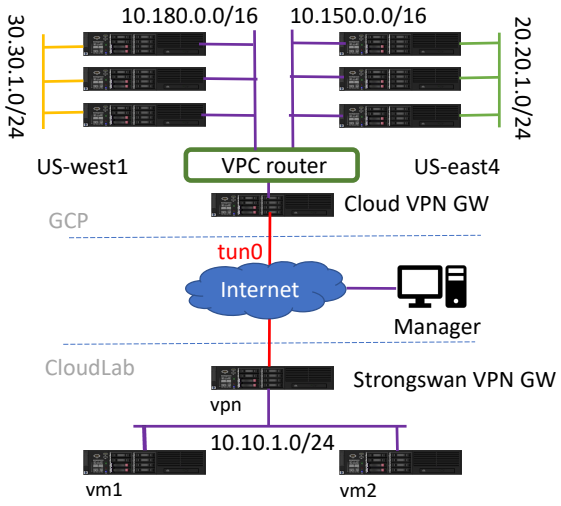

Fig. 1: Illustrative CloudJoin experiment topology.

metal machines that is recommended for high-performance VPN traffic, and the second a 16.04 Xen image for VMs. The images also have additional sample code, configuration files, and resources such as google-cloud-sdk for experimenter convenience.

We selected strongswan v5.8 [16] to implement the CloudLab-side VPN. This popular IPSec-based product works on a variety of different platforms and is known to be compatible with GCP Cloud VPN gateway [17], [18]. Unlike the GCP side, manual routing table setup is required on the CloudLab side to support communications to nodes on the cloud-side of the data plane subnet.

Fig. 1 depicts a small CloudJoin experiment network topology. The cloud-side VPN gateway persistently seeks to establish a tunnel to CloudLab, regardless of the CloudLab-side experiment state (e.g., even if not instantiated). Note that the a public IP address of the CloudLab-side VPN gateway is ephemeral; a new address appears for each newly instantiated CloudLab-side experiment, while the IP address of the cloud VPN gateway is persistent. Hence, coordinating tunnel establishment is best done from the CloudLab side after gateway instantiation. Table *A* presents a simplified bash script that can be invoked on the CloudLab-side gateway machine to create the connection.

A. Resource placement and Network Performance

Public clouds offer enough cloud locations to allow the experimenter to specify the geography of an experiment. Cloud-side resources can be instantiated in a geography minimizing inter-side communications latency. For instance, GCP instances connected to CloudLab Clemson *might* be best placed in the GCP South Carolina location, depending on the campus' cloud network connectivity via Cloud Connect, commodity internet, or regional or national R&E network. In this case we found that experiment data plane network latency to be 21 ms between sites, roughly comparable to the latency of a connection over the public internet connection.

The tunneled VPN connection limits throughput to the experiment. Running on an otherwise unused bare metal 2.6 GHz Xeon Gold 6142 class machine at Clemson, the tunnel offers roughly 750 Mbs upstream (CloudLab to GCP) and downstream. However, when the gateway node runs on a VM (same machine type) we find asymmetric throughput – approximately 476 Mbs upstream, and 250 Mbs downstream. Recall that control and monitoring traffic is transferred over the separate public internet connections (not shown in Fig. 1). Of course, additional tunnels may be constructed and logically aggregated to improve throughput performance. In addition, some CRI will be hosted at universities with higher performance cloud connections such as Direct Connects, which don't required VPNs and are typically 10 Gbs line rates and above. Exploiting this connectivity would both improve throughput and could simplify experiment infrastructure construction.

TABLE A: Illustrative bash script showing key GCP SDK commands for connecting a CloudLab experiment to a GCP virtual private cloud with an encrypted tunnel.

```
#!/usr/bin/env bash
### GCP variables and resource names
GCP_VPN_ADDRESS="34.86.208.69"
GCP_VPN_TUN_NAME="tun0"
GCP_VPN_GW_NAME="vpn-gw-east1"
GCP_TRAFFIC_SELECTOR="10.150.0.0/24"
# GCP VPN tunnel named *tun0*
# GCP VPN gateway named *vpn-gw-east1*
# GCP VPC route named *cloudjoin*

### Cloudlab variables
PEER_ADDRESS="130.127.133.66"
REMOTE_TRAFFIC_SELECTOR="10.10.1.0/24"
VPC_ROUTE="cloudjoin"

# GCP authentication and project setup
gcloud auth login
gcloud config set project "cs-research-*"
gcloud config set compute/region "us-east4"

# GCP VPN tunnel setup
gcloud compute vpn-tunnels list
# Example tunnel details:
#tun0 us-east4 vpn-gw-east1 130.127.133.66
GCP_VPN_TUN_NAME1=\\
  $(gcloud compute vpn-tunnels list\\
  --filter="name=$GCP_VPN_TUN_NAME"\\
  --format=list |\\
  grep "name" | awk '{print $2}')
gcloud compute vpn-tunnels delete\\
  $GCP_VPN_TUN_NAME1
# get public IP address
PEER_ADDRESS=$(/sbin/ifconfig xenbr0 |\\
  grep "inet " | awk '{print $2}')
gcloud compute vpn-tunnels create\\
  $GCP_VPN_TUN_NAME\\
  --shared-secret=mQbdk5qV...jT2\\
  --peer-address=$PEER_ADDRESS\\
  --target-vpn-gateway=$GCP_VPN_TUN_NAME\\
  --ike-version=2\\
  --remote-traffic-selector=\\
    $REMOTE_TRAFFIC_SELECTOR\\
  --local-traffic-selector=\\
    $GCP_TRAFFIC_SELECTOR\\
  --description=peer-address-
    $PEER_ADDRESS

# Example GCP VPC route details:
# cloudjoin net1 10.10.1.0/24 testtun0
gcloud compute routes describe $VPC_ROUTE
gcloud compute routes delete $VPC_ROUTE
gcloud compute routes create $VPC_ROUTE\\
  --destination-range=\\
    $REMOTE_TRAFFIC_SELECTOR\\
  --next-hop-vpn-tunnel=tun0\\
  --network=net1

# config and start strongswan
sed -i -s 's/$OLD_IPADDR/$PEER_ADDRESS'\\
  /etc/ipsec.conf /etc/ipsec.secrets
service strongswan restart
```

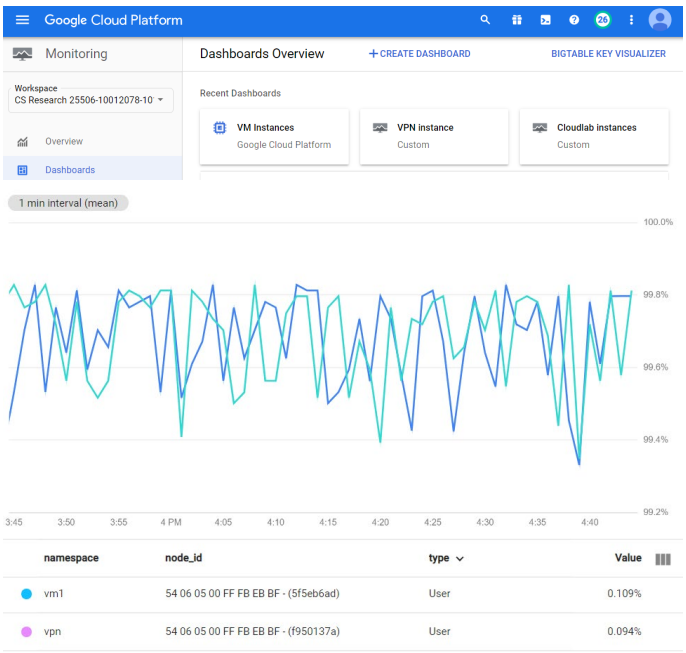

Fig. 2: GCP Stackdriver monitoring dashboard depicting cpu utilization of CloudLab Clemson nodes running BindPlane collectors.

Of course, there are many factors (i.e., cloud egress charges) in determining placement of experiment nodes. These decisions are highly specific to the characteristics of a particular experiment. Our focus is on developing tools, methods and best practices to assist individual experimenters with these deployment decisions.

V. Monitoring Your CloudJoin Experiment

Experimenters often express frustration with a CRI's user experience, particularly when encountering and debugging unexpected experiment behaviors. Experiment monitoring can be even more difficult in hybrid cloud settings, where experimental resources span systems and locations. Management tools that might have been adequate to observe experiments with tens of nodes are likely to be inadequate for thousands of nodes. In addition, experimenters need to monitor new billing management interfaces and dashboards to oversee cloud resource usage and credit consumption. CloudJoin addresses the complexities of scaling experiments by embracing commercial-grade multi-cloud management platforms equipped with monitoring, logging, alerting, error reporting and profiling capabilities.

We integrate both sides of a CloudJoin experiment using a single monitoring suite, and consequently offer experimenters a single pane of glass. Here we describe operations using *Stackdriver* [19], the GCP logging and monitoring visualization framework. CloudJoin experimenters can customize display 'cards' to assemble a professional quality system dashboard to visualize their experiment's resource use and behavior.

The implementation of CloudJoin monitoring is simple. The *stackdriver-agent* service (*collectd* daemon) is loaded and run on all GCP-side experiment nodes [20]. CloudLab nodes run the comparable Blue Medora *BindPlane* collector (*google-fluentd*) [21]. All collectors export data to a single Google Monitoring service [22]. Fig. 2 depicts a custom dashboard for a monitored experiment. The upper panel shows that both CloudLab nodes and GCP instances are being observed. The lower panel depicts a cpu utilization dashboard plot of CloudLab nodes *vpn* and *vm1* of Fig. 1 at 1 minute intervals.

While infrastructure monitoring is crucial to support large-scale experiments, we find that experimenters increasingly rely on sophisticated software components and stacks (e.g., hadoop, docker, kubernetes, mongoDB, puppet, etc) and cloud data analytics (e.g, datalab) to run and evaluate their experiments. The integrations available with commercial cloud monitoring frameworks permit experimenters to examine in parallel the operation of both infrastructure and software stacks to maintain a comprehensive view of experiment operation.

VI. Conclusion

Research advances in large-scale computing systems and software ultimately require researchers to have access to "big" experimental infrastructure and datasets to validate their ideas. While many problems can be initially studied at small scale, key learnings and deeper understanding come only by studying large datasets and workloads. But the academic computing systems research community has increasingly struggled to tackle problems at scale due in part to unavailability of sufficiently large campus-based or shared community testbeds. This problem has been particularly acute at underserved institutions with limited campus cyberinfrastructure deployments.

We have argued that successful CRI's can be more easily sustained by embracing recently introduced programming interfaces, SDKs, tools, services and integrations provided by commercial compute clouds. We stress that this solution can't independently satisfy all large-scale experiment needs, but we believe it can address some that neither federated testbeds or public clouds can on their own.

Adoption of commercial cloud computing by much of the broad science research community has proceeded slowly [23]. We are cognizant of the additional learning burden for time-pressed researchers, but we have found the benefit worthwhile. CloudJoin will continue to develop tools and practices to reduce the learning curve for those reaching beyond familiar testbeds into the less familiar public cloud computing systems.

The contributions described in this work focused on experiment-local use of hybrid cloud systems – that is, steps that an experimenter can take without explicit changes to CRI. We anticipate that our future work will extend to helping CRI developers evaluate mechanisms and deploy shared infrastructure to allow their experimenters to more easily create hybrid experiments. If successful, CRI developers stand to

benefit from elasticity by offloading some demand during peak utilization periods.

REFERENCES

[1] "Chameleon Cloud" https://www.chameleoncloud.org.

[2] "JetSteam", https://jetstream-cloud.org/.

[3] R. Ricci, et al, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," *USENIX ;login:*, vol. 39(6), Dec. 2014.

[4] Cloud TPU, https://cloud.google.com/tpu/

[5] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, B. Falsafi, "A case for specialized processors for scale-out workloads," *IEEE Micro*, vol. 34(3), pp. 31–42, 2014.

[6] J. Brassil, M. Chowdhury, Eds. "Report on the NSF Workshop on Next Generation Cloud Computing Research Infrastructure," Nov. 2019, https://sites.google.com/view/workshop-on-cloud-cri/workshop-report.

[7] "Chameleon Cloud: The Jupyter Notebook Interface" https://chameleoncloud.readthedocs.io/en/latest/technical/jupyter.html.

[8] NSF 18-540, "NSF/VMware Partnership on Edge Computing Data Infrastructure (ECDI)", https://www.nsf.gov/pubs/2018/nsf18540/nsf18540.htm, 2018.

[9] NSF 16-585, "Platforms for Advanced Wireless Research (PAWR): Establishing the PAWR Project Office (PPO) (PAWR/PPO)", https://www.nsf.gov/pubs/2016/nsf16585/nsf16585.htm, 2016.

[10] Yevgeniy Sverdlik, "Cloud Giants Continue Pouring Billions Into Data Centers". Data Center Knowledge, Oct 26, 2018, https://www.datacenterknowledge.com/cloud/cloud-giants-continue-pouring-billions-data-centers.

[11] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren, "Inside the Social Network's (Datacenter) Network", *Proceedings of the ACM SIGCOMM* London, England, August 2015.

[12] L. Peterson, "Central Office Re-envisioned as a Data center," https://opencord.org/about/, 2019.

[13] M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, "GENI: A Federated Testbed For Innovative Network Experiments," *ComNets*, vol. 61(0),

[14] Yuanjun Yao, Qiang Cao, Jeff Chase, Cong Wang, Mert Cevik, Paul Ruth, "Extending ExoGENI Slice-based L2 Network Transit Service to Chameleon", https://users.cs.duke.edu/ yjyao/chameleon-sdx.pdf.

[15] "Open Science Grid", https://opensciencegrid.org.

[16] "strongswan: the OpenSource IPsec-based VPN Solution," https://strongswan.org.

[17] Vladimir Smirnov, Bronislav Robenek, "How to set up a VPN between strongSwan and Cloud VPN," Google Cloud Community Tutorial, Jan. 2019, https://cloud.google.com/community/tutorials/using-cloud-vpn-with-strongswan.

[18] Ng Chiang Lin, "Setting up a Test Lab using Google Cloud and strongSwan Ipsec VPN," Night Hour https://www.nighthour.sg/articles/2018/google-cloud-platform-test-lab-strongswan-vpn.html. June 2018.

[19] "Google Stackdriver," https://cloud.google.com/stackdriver/.

[20] "Installing the Cloud Monitoring agent," https://cloud.google.com/monitoring/agent/install-agent.

[21] "bluemedora BindPlane," https://bluemedora.com/products/bindplane/.

[22] "Monitoring on-premises resources with Blue Medora," https://cloud.google.com/solutions/monitoring-on-premises-resources-with-blue-medora.

[23] NSF 19-510, "Enabling Access to Cloud Computing Resources for CISE Research and Education (Cloud Access)", https://www.nsf.gov/pubs/2019/nsf19510/nsf19510.htm, 2018.

[24] A. Akella, "Experimenting with next-generation cloud architectures using CloudLab," IEEE Internet Computing, vol. 19(5), 2015.

[25] D. Duplyakin, R. Ricci, "Introducing Configuration Management Capabilities into CloudLab Experiments," *Proceedings of the International Workshop on Computer and Network Experimental Research Using Testbeds (CNERT)*, April 2016.

[26] J. Aikat, et al, "The future of CISE distributed research infrastructure," *ArXiV*, https://arxiv.org/abs/1803.09886, March 2018.

[27] F. Hermenier, R. Ricci, "How To Build a Better Testbed: Lessons From a Decade of Network Experiments on Emulab," *Tridentcom*, June 2012.