A Hybrid Cloud Architecture for a Social Science Research Computing Data Center

Steven Abramson, William Horka, Leonard Wisniewski
Institute for Quantitative Social Science
Harvard University
1737 Cambridge St., Cambridge, MA 02138 USA
sabramson@fas.harvard.edu, whorka@hmdc.harvard.edu, lwisniewski@iq.harvard.edu

***Abstract*—Research computing in the social sciences requires access to statistical software and quantitative tools that perform embarrassingly parallel computation at moderate scale, large memory to fit entire data sets, and secure storage for potentially confidential data. The Research Computing Environment (RCE) was designed as a three-tier system to satisfy these requirements in a transparent manner. We extend the RCE to use cloud resources while maintaining the transparency of the resources from the user. This paper describes this use case and highlights the significant resource management and networking decisions made when designing and implementing a hybrid cloud architecture for a research computing environment to support the social sciences.**

Keywords— cloud computing, load balancing, research computing, secure storage, virtual private network

I. Introduction

There is an ongoing convergence in research computing as computational social science takes on more and more characteristics similar to the high-end computing traditionally reserved for the physical sciences. However, there are also computational and data requirements of social science applications that are unique and benefit from design choices that are different from the physical sciences when architecting a programming and data analysis environment. Examples include social networking data, geocoded data, political data, commercial data, and newly-digitized records, all of which are about people and most of which has at least some confidential component.[1]

The Research Computing Environment (RCE) at the Harvard-MIT Data Center (HMDC) was architected with the requirements of social scientists in mind. This three-tiered architecture enables the user unfamiliar with cluster computing to easily get started by using familiar applications and their respective user interfaces. A key requirement for the architecture is to make cluster computing as similar as possible to desktop computing while enabling the power of much larger and more scalable resources.

In this paper, we describe the use case of research computing in the social sciences. In Section II, we start off with a brief history of the Harvard-MIT Data Center (HMDC) and the requirements of the users it supports. Section III describes the architecture of Research Computing Environment and how it satisfies the requirements of its users as well as introducing how its architecture can be extended to make use of cloud resources to dynamically and automatically expand and contract the size of the cluster. Section IV describes some of the implementation details about how we automatically control the size of the cluster. Section V covers some of the networking considerations we encountered when including the cloud resources in our compute cluster and extending access to our secure local storage. Section VI identifies other social science data centers and other similar hybrid cloud architectures. Section VII concludes and explores future efforts for expanding our data center to use other clouds, both public and private, as well as for branching into supporting other emerging use cases.

II. History of Harvard-MIT Data Center

The Harvard-MIT Data Center (HMDC) was originally established as the Government Data Center in the early 1960s to collect, consolidate, and share social science research data, and had a strong role and affiliation with establishing the Interuniversity Consortium for Political and Social Research (ICPSR).[2] The community built around these data-sharing enterprises led to further sharing of data sources, statistical methods, computer technology, and software.

Since these data resources were shared with social scientists in all of Harvard's schools and departments, the name was changed to the Harvard Data Center. In the 1990s, the collaboration extended outside the campus to a partnership in 1996 with our neighbor, MIT, to become the Harvard-MIT Data Center. In 2005, HMDC became a founding member of the Institute for Quantitative Social Science (IQSS) at Harvard.

In addition to its many operational and research technology service responsibilities, IQSS has become a center of excellence within the university for systems and layered software development. In particular, IQSS/HMDC first developed the Virtual Data Center [3], which later evolved into the Dataverse Network, an operational, open-source, digital library for sharing, citing, reusing, and archiving quantitative research data.[4][5] Whereas the Dataverse currently serves as a repository and archive for polished research data sets, the Research Computing Environment (RCE) was established as a developer environment for

performing the data cleaning and analysis to arrive at those final research data sets.

III. Research Computing Environment

A. Three-Tiered Architecture

The Research Computing Environment (RCE) was designed as a three-tier architecture to enable quantitative research that scales in terms of both computation and data set size. The first tier servers are called *login nodes*, where users login to the RCE and receive a virtual Gnome desktop via NoMachine NX software.[6] On a login node, the user can use statistical software to perform computations on modestly-sized data sets since they are sharing a server and its memory with perhaps 20-40 users at any time.

The second tier of *large memory nodes* offers dedicated access to servers with much larger memories. A user requests a specific application to be run with a certain amount of memory and is granted one or more dedicated CPU cores running the application with the requested amount of dedicated memory. Like the login nodes, the large memory nodes run popular data analysis applications via their GUIs, essentially allowing the user to operate in the same way as they usually do, just with a much larger amount of available memory. This satisfies the cases where certain statistical methods within these applications require memory for the entire data set being analyzed. Currently, users can request as much as 250 GB of memory. Efficiencies are gained over instantiating virtual machines for each job by allowing one kernel / OS to control multiple dynamically partitioned "slots" on a single large memory node, each of which can then run a separate user job.

The third tier of *batch nodes* is accessed when the user wants to perform scalable computations. In particular, most social scientists use the batch nodes for "embarrassingly parallel" computations, i.e., computations that do not require communication during run time among the parallel processes. After collecting some statistics, we found that a large majority of our batch node users were running R scripts and hence we designed the architecture to optimize for this use case.

Figure 1 shows a picture of the RCE architecture. The user logs in securely via NX onto the login nodes. From the login nodes, they can request large memory or batch nodes. Allocations of dynamically-sized slots on large memory nodes are made automatically and transparently when a memory-intensive application is launched, whereas allocations of fixed-size slots on batch nodes are made by manual submission via a resource manager (Condor). Condor uses fair scheduling among all users and allocates the requested number of job slots on available node(s) and memory in the case of the large memory nodes. The RCE nodes all operate on a private subnet. All the nodes have access to a Netapp filer that also resides on the same subnet, which periodically transfers backup copies to tape.

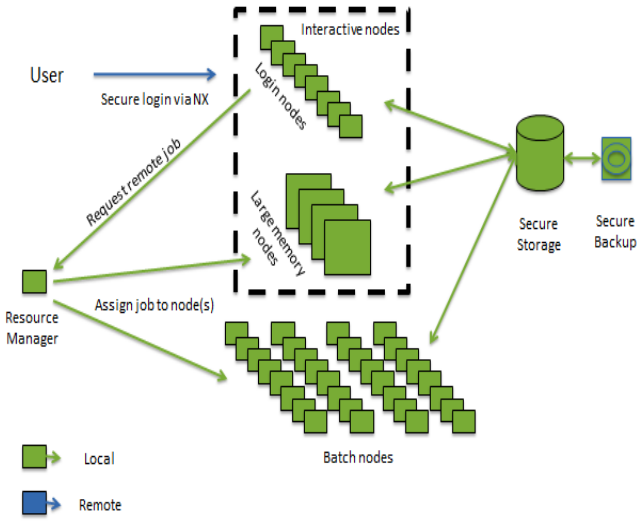

Fig. 1. RCE three-tiered architecture.

B. Extending to the Amazon Cloud

There were several motivations to extend the RCE to the cloud. First, there were several "super" users of the RCE who developed applications that required more nodes than we were willing (or budgeted) to maintain on a consistent basis. The elastic nature of making available a scalable set of cloud resources enables the possibility of running on a much larger set of resources for a short period of time, avoiding much of the overhead for supporting a very large cluster full-time.[7]

Second, given the specialized nature of our users, we would much rather spend our time developing software to make best use of computational and data resources rather than performing operations duties that groups like Amazon can handle far more efficiently through experience and economies of scale.

Third, it is very attractive to be able to use resources that are offsite for availability and disaster recovery. Although we are not ready to move our entire data center to the cloud, we do like the possibility, even if only in emergency situations.

Figure 2 shows our RCE hybrid cloud architecture. It is hybrid in that some nodes are local and some nodes reside in the cloud. Since a large portion of our data is confidential, we wanted to ensure that we still keep local resources, at least initially, for processing the confidential data. Likewise, the hybrid architecture could have some portion of secure storage both locally and in the cloud as also shown in Figure 2. However, in our first phase of development, we decided to avoid the need to manage any reasonable partitioning of data among the local and cloud nodes. Also, if we eventually move the entire cluster to the cloud, then we would have unnecessarily built a more elaborate distributed file management system.

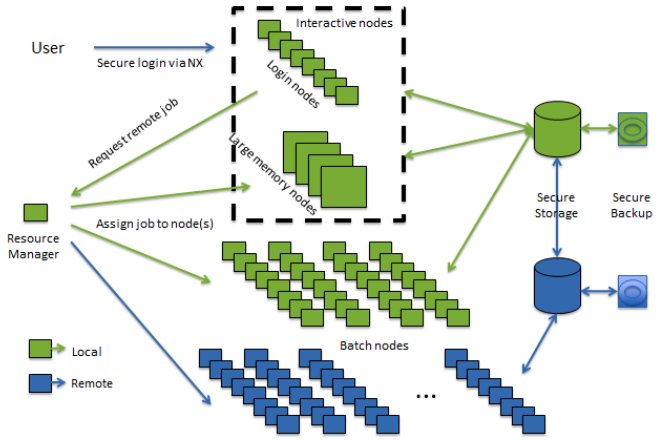

Fig. 2. RCE Hybrid Cloud Architecture. The lower set of batch nodes, storage device, and backup device are "in the cloud".

IV. RCE Cloud Software Services

A. Resource Management

The RCE uses the *Condor Resource Manager*.[8] Condor, an open source project developed at the University of Wisconsin, enables jobs to be scheduled and executed in parallel. At a high level, the Condor architecture consists of a *Central Manager Service* which manages incoming jobs and *Worker Nodes* which carry out the execution of client jobs. Each *Worker Node* maps 1:1 to a physical server in the network architecture. A Worker Node typically contains multiple *slots. A* slot maps 1:1 to a CPU on the physical machine. The Central Manager Service will match a Condor job to an available slot within the pool of Worker Nodes.

RCE Cloud Services provide a hybrid environment for processing jobs. RCE Cloud Services integrate local RCE servers with the Amazon Web Services (AWS) cloud.[9] A local server pool extends to the Amazon cloud, dynamically scaling up or down as utilization dictates. The cloud-based nodes are instantiated and utilized as the local pool utilization is nearing capacity. When the utilization levels are reduced, the cloud Worker Nodes are terminated.

B. Amazon Web Services

The following Amazon Web Services components were utilized:

- **Virtual Private Cloud (VPC)** – A VPC provides a private cloud infrastructure for a single AWS account holder. All underlying resources used in the VPC are dedicated to the single AWS account holder. In contrast, the AWS public cloud shares underlying resources across multiple AWS account users in a true multi-tenant deployment.
- **Virtual Private Network (VPN)** – A VPN is used in conjunction with the VPC to enable a uniform network environment which bridges the local network pool of Worker Nodes with a cloud network pool of Worker Nodes. The Central Manager Service cannot tell the difference between local Worker Nodes and cloud Worker Nodes – all appear as identical entities.
- **CloudWatch** – CloudWatch is a metrics service, typically used for watching metric values on a variety of AWS services such as EC2 instances (CPU utilization, disk utilization etc…). Metrics are time-stamped, averaged over variable periods, displayed, and enabled to trigger alarms. CloudWatch also allows users to create "custom metrics". We have created a custom metric which keeps track of Condor Pool Utilization (i.e. what percentage of our Condor pool is currently being utilized?).
- **DynamoDB** – DynamoDB is used to manage configuration data as well as run-time data concerning Worker Node termination.
- **Identity and Access Management** – Identity and Access Management (IAM) is used to control access to AWS components. Role based authentication is used for Worker Nodes. This removes the requirement for storing AWS access keys on our EC2 image.

In addition to utilizing these AWS services in our solution, two software services were developed using the AWS SDK for Java. These services are described in the next two subsections.

C. RCE Metrics Service

The RCE Metrics Service executes locally on the Condor master server. The service queries the local Condor deployment at 3-second intervals for utilization metrics. These metrics are then pushed to the AWS Cloudwatch metrics service for aggregation, persistence and later retrieval. The metrics gleaned from Condor include the following:

- Claimed slots
- Unclaimed slots
- Idle slots
- Busy slots
- Matched slots
- Owner slots
- Pre-empting slots
- Vacating slots
- Activity Utilization
- State Utilization

The parameter *State Utilization* represents the percentage of slot utilization in the Condor pool and this is the value which is pushed to AWS Cloudwatch. At the end of the 3-second interval the service queries AWS Cloudwatch and retrieves the current 3-minute average value of the utilization metric. If the value is greater than a pre-determined utilization threshold (currently 90%) the service will trigger a new Worker Node to fire up in the AWS Virtual Private Cloud.

This Worker Node seamlessly integrates into the Condor pool as the VPC is connected to the local network via a Virtual Private Network (VPN). See the diagram below:

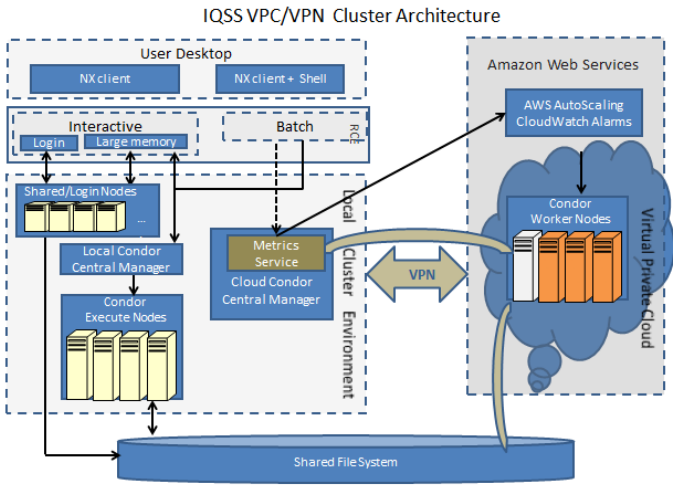

Fig. 3. RCE Cloud Services VPC/VPN architecture

D. *RCE Termination Service*

Each Worker Node instance fired up in the AWS Virtual Private Cloud executes the RCE Termination Service on boot. The termination service wakes up at 30-second intervals and retrieves the current utilization metric value from AWS Cloudwatch. If the metric value is lower than a pre-determined threshold (currently 80%) the Worker Node will attempt to terminate itself. In order for the Worker Node to self-terminate:

- The utilization metric must be below the designated threshold (80%)
- Only one Worker Node can terminate at any given time. The Worker Node must acquire a lock on a particular record in an AWS DynamoDB table. Acquiring the lock will prevent any other Worker Node from termination.
- Worker Node terminations maintain a cool-down period (currently 5 minutes) which specifies the minimum time spacing for node terminations. The AWS DynamoDB table records a timestamp of the last Worker Node termination. The last node termination timestamp and the current time must be at an interval greater than the cool-down period for termination to proceed. If the interval is greater, the timestamp on the record is updated, the prior timestamp is also saved, and the process proceeds to action in the next bullet.
- The Worker Node must be idle (none of its slots should be processing jobs). If the first 3 requirements listed above pass the termination test, the service then checks the node it is running on for its local utilization status. If the Worker Node is seen to be idle the Termination Service will put the node in a suspended state where it can no longer accept jobs. One final check of the node utilization is made to ensure no new jobs were accepted while the node was placed in suspended state. If the node is clear of jobs the Termination Service then terminates the AWS instance that it is running on. If the node is not idle at any point during this step the Termination Service rolls back the timestamp which was set in the action in the previous bullet. Once the rollback is complete the node is taken out of suspended mode and returned to an on-line status.

E. *Determining Utilization Thresholds and Cool-Down Periods*

The operational goal of this system is to provide as many cloud-based resources (currently Condor Worker Nodes) as needed when the influx of user jobs overload local (on-premise) Worker Nodes. When the influx of jobs has diminished and cloud-based resources are no longer needed, these resources are terminated. Two "Utilization Thresholds" and two "Cool-down Periods" are required for the system:

- Start utilization threshold – When this threshold is crossed a new Worker Node is fired up in the remote cloud. In addition to the utilization value crossing above the start threshold, the "start cool-down period" must be in the "ready" state. **Default setting for the Start Utilization Threshold is 90%.**
- Terminate utilization threshold – When this threshold is crossed an existing Worker Node running in the cloud is terminated. In addition to the utilization value crossing below the terminate threshold, the "terminate cool-down period" must be in the "ready" state. **Default setting for the Terminate Utilization Threshold is 80%.**
- Start Cool Down Period – The period of time which must pass between successive starts of Worker Nodes. This period needs to be tuned to ensure the system does not start too many Worker Nodes (some of which may be unused) or may not start up enough Worker Nodes to ensure managing the load of jobs quickly enough. **Default setting for the Start Cool-Down Period is 15 minutes.**
- Terminate Cool Down Period – The period of time which must pass between successive Worker Node terminations. This period needs to be tuned to ensure that we are not terminating Worker Nodes so quickly that we then need to start new Worker Nodes. **Default setting for the Terminate Cool-Down Period is 15 minutes.**

Ultimately the precision in setting these thresholds will correlate to how the system performs with regards to the following metrics:

- Job wait times (from the time the job enters the queue until the time processing begins)

- Total job processing time (from the time the job enters the queue to the time the job completes)
- Cost to operate Worker Nodes on the remote system

Initially we propose to use the default settings and to provide an analysis of the following data:

- Number of jobs waiting in queue to be processed
- Size(estimation of processing time) of each job waiting in queue to be processed
- Job volume which correlates to time of day.

After collection of this data over an extensive period of time we will identify time periods when the default utilization thresholds and cool-down periods were not optimal. Correlation of the above data items with this period will enable us to develop a policy which implements an algorithm to adjust the utilization threshold and cool-down periods dynamically.

V. NETWORKING TO THE CLOUD

A Virtual Private Cloud (VPC) is a set of cloud resources accessible from the local customer site through an encrypted VPN tunnel. A VPC comprises:

- **VPN Connection** -- this is the configuration for the VPN tunnel, which defines the Customer Gateway (i.e. local tunnel endpoint). It can be exported as a text file to install on the VPN router.
- **At least one Subnet** -- this is a specified range of IP addresses and an Amazon Availability Zone into which nodes in the VPC will be instantiated. We are using two subnets, a private one with a route to the customer site through the Virtual Private Gateway (i.e. remote tunnel endpoint), and a public one which has a route to the world through an Internet Gateway (i.e. an external-facing AWS router) in addition to the route to the customer site. Packets exiting the Internet Gateway must have a publicly-addressable source IP, and for this we use a pair of highly-available NAT router instances with a shared Elastic IP (instantiated from an Amazon Linux NAT AMI) on the public subnet.[10] Routing is permitted between the subnets, so nodes on the private subnet can access the Internet through the NAT.
- **Route Tables** -- each Subnet is associated with a routing table that instructs the virtual router in the Amazon cloud how to route packets. The route itself is not visible from the instance (guest) OS. Route tables on the instance OS are created via DHCP, and the default gateway is actually the Amazon virtual router.
- **Security Groups** -- sets of firewall "allow incoming" rules. When a node is instantiated in the VPC it must be instantiated into a Security Group, which defines what firewall rules are applied to packets destined to it. We have two Security Groups, a default one to allow inbound SSH from the customer site, and a NATsg to allow all traffic from the private subnet to the NAT (so that it may be forwarded to the Internet) as well as allowing inbound SSH from the customer site.

VI. COMPARABLE SYSTEMS

A. Other Social Science Data Centers

There are many universities with similar-sized research compute clusters that support social science research. Northwestern University's Social Sciences Compute Cluster offers a similar centralized university-wide service with similar software and batch node services available.[11] The Social Science Gateway at Cornell University provided services to enable users to leverage in a user-friendly manner nationally-shared XSEDE/Teragrid compute resources.[12] Duke's Social Science Research Institute offers remote desktop access to a small cluster of larger memory Windows machines.[13] Stanford's Institute for Research in the Social Sciences offers a GPU cluster in addition to a batch cluster.[14]

B. Other Hybrid Cloud Architectures

Many enterprises and software toolkits offer some form of hybrid cloud service in varying manners. We chose to implement our own software because many of the alternative solutions offered more infrastructure and framework than was necessary for our environment. As early as 2009, there were several efforts into developing methods for managing hybrid resources from both private and public clouds. Zhang et al. provide a hybrid cloud computing model where a dedicated private resource runs a base workload and a separate public resource hosts a "trespassing peak load".[15] Sotomayor et al. propose a combination of OpenNebula as a virtual infrastructure manager for deploying virtualized services on both a local pool of resources and external IaaS clouds and Haizea as a resource lease manager that acts as a scheduling backend for OpenNebula.[16]

VII. CONCLUSIONS AND FUTURE WORK

We have presented a hybrid cloud architecture suited to the needs of social science researchers. We have described an implementation that expands and contracts a cluster adaptively to the current overall cluster workload as well as explored methods that are secure, efficient, and relatively simple for storing data and providing secure networking.

A. Simulation

As we move into production use, we would like to further optimize the use and administration of our hybrid cloud environment. We believe a lot of these optimizations could be learned from simulation of new tuning policies and strategies. The following are several examples where simulation of our environment would help us identify more efficient modes of operation.

1) ***Workload Analysis*** – Getting a better handle on the job types, sizes, I/O patterns, etc. would help us establish a harness for simulation.

2) ***Job Placement (Local vs. Remote)*** – Testing policies that help determine whether a job is a candidate to run remotely on such criteria as confidentiality of the data set and data set size.

3) ***Cluster Expansion / Contraction Algorithms*** – We've identified two use cases where a) gradual increase or decrease in cluster size adapts to a changing overall workload, and b) instant large parallel requests require a sharp jump in cluster size. Based on our workload analysis, how might we anticipate future use to optimize cluster size and its associated per resource per time cost.

4) ***Data Placement Strategies*** – For the first phase of our work, we decided on sharing our local file system with remote nodes over the VPC. Does this become impractical for certain classes of jobs where I/O latency and bandwidth become a limiting factor?

B. Expanding to Other Clouds

Our initial work has concentrated on expanding to the Amazon cloud. However, there are situations where an efficiently-run large private cloud makes sense for a large enterprise, such as a University with many schools and departments. The same hybrid cloud architecture can be applied to other clouds as long as the resource management layers work well together and there is virtual private cloud networking among all clouds, both private and public. Eventually, this concept can expand to simultaneous support for multiple clouds that can be leveraged for various criteria such as cost, appropriate match of resource to job, security, etc.

C. Distributed File Systems

As resource usage scales and branches out to multiple clouds, it may make sense to move to a distributed file system model, leveraging virtual distributed file system technologies to manage file replicas that benefit from residing closer to the computation for which they are used.

D. Securely Isolating Jobs

Harvard University has a policy on the use of various levels of confidential data.[17] The highest risk confidential data requires an environment that isolates all resources associated with storing and processing the data. What methods and policies might one employ with the RCE cloud architecture to ensure the isolation of such data and its associated processing in a shared resource environment.

E. Hierarchical Database Systems

With the emerging computational paradigms in mind, what type of shared cluster system might best serve its users with diverse database needs. How might a multi-tiered hierarchical database be employed to automatically match the user's application need with a database type on shared stores for those users who don't want or need to know the theory behind database efficiencies.

ACKNOWLEDGMENTS

Thanks to Micah Altman, Matt Cox, and Bob Kinney for designing and implementing the original three-tiered architecture. Thanks to Wes Harrell and his team for keeping the operations and update of the cluster to its next generation true to its architecture and requirements. Thanks to Gary King for providing requirements and feedback every step of the way.

REFERENCES

[1] G. King, "Ensuring the Data Rich Future of the Social Sciences," Science 331, no. 11, pages 719-721, February 2011.

[2] Institute for Quantitative Social Science, Harvard University, http://iq.harvard.edu.

[3] M. Altman, L. Andreev, M. Diggory, E. Kolster, M. Krot, G. King, D. Kiskis, A. Sone, S. Verba, "An Introduction to the Virtual Data Center Project and Software," Proceedings of the 1st ACM/IEEE Joint Conference on Digital Libraries (JDLC 01), pp. 203-204, 2001.

[4] Dataverse web site, http://thedata.harvard.edu/dvn/.

[5] M. Crosas, "The Dataverse Network: An Open-Source Application for Sharing, Discovering, and Preserving Data", D-Lib Magazine, vol. 17, no. 1, January 2011.

[6] NoMachine web site, https://www.nomachine.com.

[7] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report No. UCB/EECS-2009-28, EECS Department, University of California Berkeley, February 2009.

[8] Condor web site, http://research.cs.wisc.edu/htcondor.

[9] Amazon Web Services web site, http://aws.amazon.com.

[10] J. Varia, "High Availability for Amazon VPC NAT Instances (Using AWS CloudForm Templates), May 2013, http://aws.amazon.com/articles/6079781443936876.

[11] Social Science Computing Cluster (SSCC), Northwestern University, http://www.it.northwestern.edu/research/sscc/index.html .

[12] Social Science Gateway, Cornell University, http://www2.vrdc.cornell.edu/news/social-science-gateway/.

[13] Social Science Research Institute, Duke University, https://ssri.duke.edu/about/facilities/computer-labs.

[14] Institute for Research in the Social Sciences, Stanford University, https://iriss.stanford.edu/tools/computing.

[15] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, A. Saxena, "Intelligent Workload Factoring for a Hybrid Cloud Computing Model," Proceedings of the 2009 Congress on Services, Washington, DC, pp. 701-708, 2009.

[16] B. Sotomayor, B., R.S. Montero, I.M. Florence, I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds", IEEE Internet Computing, 13(5):14-22, 2009.

[17] Harvard Research Data Security Policy, http://vpr.harvard.edu/pages/harvard-research-data-security-policy.