TRANSCLOUD:
Design Considerations for a High-Performance Cloud Architecture Across Multiple Administrative Domains

Andy C. Bavier, Marco Yuen
Princeton University, Princeton, NJ,
acb@cs.princeton.edu, marcoy@gmail.com

Jessica Blaine, Rick McGeer, Alvin Au Young
HP Labs, Palo Alto, CA
jessica-ann.blaine@hp.com, rick.mcgeer@hp.com, alvin.auyoung@hp.com

Yvonne Coady, Chris Matthews, Chris Pearson
University of Victoria, Victoria, BC, Canada
ycoady@cs.uvic.ca, cmatthews@cs.uvic.ca, pearson@cs.uvic.ca

Alex Snoeren
University of California, San Diego
snoeren@cs.ucsd.edu

Joe Mambretti
Northwestern University
j-mambretti@northwestern.edu

Keywords: Distributed query infrastructures; federation across cloud domains; advanced network architecture

Abstract: In this position paper, we consider architectures of distributed interconnected clouds across geographically distributed, independently-administered storage and computation clusters. We consider two problems: federation of access across heterogeneous administrative domains, and computation jobs run over the wide area and heterogeneous data sets. We argue that a single, flexible architecture, analogous to the TCP/IP stack for networking, is sufficient to support these jobs, and outline its major elements. As with the networking stack, many elements are in place today to build an initial version of this architecture over existing facilities. With the sponsorship of the US National Science Foundation GENI project and the cooperation of the EU FIRE project, we are building an initial implementation, the TransCloud. We describe our initial results.

1 INTRODUCTION

The dramatic trend of the first decade of the 21st century in the information technology industry was the emergence of *society-scale systems*: online services such as Google, eBay, iTunes, Yahoo!, Twitter, and Facebook that routinely served millions of simultaneously-connected users. These systems gave rise to entirely new programming models and systems problems: management of the data center as a single, unified, "warehouse-scale" computer, each of which had more raw computing power than existed on the planet as late as 1990; programming models for loosely-coupled, data-intensive parallel operations ("data-intensive supercomputing"), most concretely realized in the MapReduce architecture from Google and its open-source cousin, Hadoop. vast, highly-efficient distributed data stores such as PNUTS and Cassandra; the re-emergence of virtualization of time, space, and computing, to permit services to migrate instantly around the globe and radically new notions in networking to support the new programming and management models.

As revolutionary as the last decade has been, the coming decade promises a far more profound transformation: the twin emergence of the Computation Cloud and the Internet of Things. For all of its power and promise, the Cloud today is little more than a massive, well-indexed repository of text, videos, photos, and music, and a vast, universal transaction engine. The Cloud has merely automated and made vastly more efficient traditional human communications and commerce. Over the next decade, widespread availability of massive computation – the Computation Cloud -- will give to everyone the ability not only to look up what someone knows, but to discover things that no one knows.

Paired with the Computation Cloud is the Internet of Things – a world where every object houses a computer, sensor or sensors, and a connection to the network. The applications range

from the trivial to the profound; milk that senses when it's going bad and tells its owner to drink it up, and buy more; automobiles that drive themselves and automatically (in coordination with other vehicles on the road, and a network of smart highways) avoid traffic jams; homes that pre-cool themselves when power is cheap and plentiful and turn off air conditioning when it is dear; smoke alarms that can tell the difference between a real fire and a ruined dinner, and call the firefighters for the former; intelligent buildings that vector people to a safe escape route in the event of an emergency; fine-grained climate sensors that can predict severe weather and evacuate people before the storm hits; and many more.

Nascent signs of the Internet of Things are already emerging. An American automobile has tens of operator-visible sensors, and they already operate the automobile in some ways better than the driver can; anti-lock brakes are a classic example. The RFID tags used to automatically pay tolls on many American highway systems are used to trace traffic speeds on major freeways. And the autodoc, long a science fiction dream, is already emerging. This year, the NHS rolled out a urine test kit that transmitted results through a user's cell phone.

The computation cloud is tightly coupled to the Internet of things. The deployed sensors will range in capacity and bandwidth from a few bytes transmitted every few seconds to every few hours. Together, they will generate vast quantities of information; zettabytes and beyond. Extracting information from all of that data is a formidable computational task, well beyond current capabilities. Only a vast new computational infrastructure will suffice to process all that data. The challenges are vast:

- Computational infrastructure. Reduction of the data requires a flexible, universal programming interface. Most data will need to be reduced at or near the point of collection; the sheer volume of data and real-time requirements ensure that. Therefore, an open, standard, and sufficiently powerful computational infrastructure – an arm of the computational cloud – will need to be proximate to any collection of sensors.
- Security. New, robust, and secure protocols will be needed to access and manipulate the Internet of Things. To date, the notorious lack of security in cyberspace has been of limited consequence; while there were large consequences in the real world, they were indirect, typically requiring a separate physical transaction for real-world effects. The Internet of Things will enable direct manipulation of the physical world.
- Data management. The Internet of Things envisions a widely-distributed set of sensors, data consumers, and fusion of information from many disparate, distributed sources. *In-situ* reduction of data at the source on a per-query basis, coordination of widely distributed queries, and a wide variety of data access mechanisms ("data blades" in the parlance) will be required, including new distributed computation mechanisms. Some early innovations are already present: MapReduce and Hadoop in the data center, and Sector and Sphere in the distributed environment. But these are just the beginning. We can anticipate the Internet of Things will required the adaptation of many different data management and query mechanisms, from distributed monitoring systems (S3, Ganglia, PsePR) to sensor net systems (TinyDB), to distributed information systems (SWORD)
- Networking. Efficiently connecting the Internet of Things and the Computational Cloud will require new networking stacks and protocols, both wired and wireless
- Networks must be adaptive in the face of changing conditions, using one of a number of radio frequencies and choosing routing on an adaptive basis.

In this paper, we focus on three problems:

- Ensuring that Computation Cloud users can run computation jobs wherever they have access, as simply and transparently as they now download files from multiple computers across the web
- Ensuring that execution of remote queries is done efficiently and safely for both remote user and data host
- Designing a simple, efficient, network-aware architecture for queries over geographically-distributed heterogeneous data.

2 SCALABLE LIGHTWEIGHT FEDERATION

The computation cloud offers individuals, small companies, and researchers the ability to develop,

test and deploy Internet-scale services easily and at low cost. However, many of these services require the use of multiple facilities, or users wish to transparently move their services across multiple facilities. This gives rise to the desire to federate facilities. In this section, we view facility federation not as a set of agreements between federated facilities, but rather as a set of services to developers and facilities. This approach scales easily across heterogeneous facilities, operating in different environments.

We motivate our discussion by considering a somewhat simpler analogue: electronic document exchange. Of course there are many problems that a truly reliable, fully-functional document exchange system must have: there must be a rich permission and security system so that documents can be shared only with selected remote users; a rich variety of formats and clients must be supported; documents should be unforgeable, so an ironclad identification system must be in place; and so on.

Of course we have a ubiquitous electronic document exchange system: the World Wide Web, and even a casual user will recognize that the web has none of these features. Rather, the web has a bare-bones protocol to send a document from one computer to another, and a simple format that can be easily rendered by a wide variety of clients. Other features are layered by third-party software on a per-site basis as needed.

Our goal is to design a simple, ubiquitous system which will permit users to easily run their programs on a set of computers owned by a remote facility. As with electronic document exchange, there are many issues to be considered: acceptable use policies; resource allocation; effects on other facility users and, in a networked world, effects on the network; global user ID's and names; federated job control; and, most importantly, agreements between various hosting facilities as to what constitutes acceptable use.

In the spirit of the web, we consider *none* of these. Our goal is to design a system whereby users can manage their jobs on facilities to which they have independently obtained access. This involves designing two central components.

1. An architecture and set of interfaces which permit users to easily and rapidly upload, configure, and run virtual machines
2. A service which manages a user's access to and use of facilities.

The first component is the analogue in our system to a web server (more precisely, to the *specification* of a webserver); the second, to a web browser.

The architecture we choose is the Slice-Based Facility Architecture (SFA) (Peterson et al, 2007). It is an open, standardized set of facilities to manage individual VMs and networks of VMs. In order to demonstrate its utility for this purpose, we have added support for the SFA into the Eucalyptus cluster management system. These efforts have demonstrated that the functionality in the SFA is a superset of the functionality supported by Eucalyptus; in particular, the SFA offers the ability control *slices*, or sets of virtual machines, and the topology of the network of virtual machines.

The key feature in the SFA required is the *delegate* primitive, which permits one user (given by a public ssh key) to assign its privileges to a delegated ssh key.

The second component is easily added as a cloud service which manages cloud services. The user registers with the cloud service, and registers his public key with the service itself, as well as the URI of the services to which it is delegating permission. The cloud service then uses the standard SFA calls to instantiate slices, slivers, initialize and run VMs, allocate resources, and control jobs.

3. SAFE EXECUTION OF REMOTE JOBS AND QUERIES

At its most abstract level, a data management and query infrastructure is an instantiation of a distributed computing system of very wide applicability: a wide-area distributed data management, query, and computation system over heterogeneous computing nodes and data types. By "wide-area" here we mean nodes whose network connectivity can be relatively low-bandwidth, in the range of tens of kilobits/second and have internode latencies in the range of tens of milliseconds and beyond. Intermittent connectivity of some nodes is the norm.

Data types can range from small collections of simple data records to very large media files. For the former, consider RFID readings of vehicle positions, used to determine average traffic speeds on freeways. A record will consist of an RFID tag, a time, and a position, perhaps tens of bytes, and there will be at most a couple of records per second. For the latter, consider weather video data of the sort generated by the CASA experiment.

This general problem: large, heterogeneous data, spread over a distributed computing infrastructure with varying connectivity and no common administrative interface – is ubiquitous through the natural, social, and engineering sciences. We are designing and implementing a computing

infrastructure which addresses the distributed data management and query problem, and deploy it in a live service.

The service we will deploy is the State of the Internet service, deployed over the TransCloud infrastructure.

The basis of the query engine is a sandboxed environment which permits the user to run programs safely and efficiently at remote sites, and is based on two fundamental architectural building blocks:

1. Restricted Python (*Repy)*, a sandboxed execution environment originally used in the *Seattle* project
2. Google Native Client (*NaCl)*, a sandboxed native-code execution environment distributed with the Firefox and Chrome browsers, with x86 implementations.

The two central elements work together to provide a secure, efficient execution environment where side effects are tightly controlled. NaCl offers an efficient execution environment in a secure sandbox for computation-intensive code; safety is guaranteed by severely restricting access to system services. However, any real job requires more system services than NaCl provides. In particular, jobs in our context require access to network connectivity and resources. NaCl relies on a trusted service on the client in order to provide these services. RePy is the mechanism we choose: it has been widely deployed on a number of platforms, and offers secure access to a restricted but adequate set of system resources. Optionally, NaClRePy can run inside a virtual machine for added isolation and security. Our initial deployment of NaClRePy inside the TransCloud environment offers this.

4. A WIDE-AREA QUERY INFRASTRUCTURE

NaClRePy merely offers a safe execution environment. Above that, we require a distributed query/data reduction environment that is network-aware, processes and reduces data optimally with consideration of latency, bandwidth, and available processing capacity. Such an environment must support common data types, and must be extensible to new data types on an on-demand basis.

There are two central themes of the data processing environment. The first is a distribution mechanism, and the second an extensible data extraction and processing mechanism. For the first, we turn to early attempts to provide services of this form, notably Astrolabe, Hadoop, and Hadoop's wide-area cousins: Sector and Sphere. Hadoop has achieved widespread use and popularity in a cluster environment. However, extension to the wide area is still an unresolved issue. There are two major issues to be addressed:

1. Use of addressable subnets, easy in a data center environment but challenging in the wide area
2. Restrictions on bandwidth and large latencies in the wide area.

Sector, Sphere, and Astrolabe have addressed some of these issues. We will develop a hybrid approach and report on it as a deliverable from this research.

Extensible data mechanisms have been primarily addressed with Data Blade (Brown, 2001) in a SQL database context, and in object-oriented databases. We will extend the Data Blades mechanism, using an XML-based metadata template system to guide optimized queries.

5. PRIVATE NETWORKING

In this paper, we have largely considered computation over distributed, separately-administered clusters. However, connectivity between distributed, jointly-administered clusters must be considered.

The fundamental issue in inter-site networking over the Internet is that the basic Internet protocols are designed to ensure fairness and decentralized, orderly traffic management in the absence of global information. The price is performance. In particular, the TCP/IP protocol suite involves strong caps on performance based on latency and loss. These can be avoided (Bavier et al, 2006) (Brassil et al, 2009), but only with information obtained from the network. In the absence of such information, guaranteed performance can only be obtained by significant overprovisioning (McGeer et al, 2009). Indeed, vanilla TCP/IP (Linux 2.6 kernel, default parameters) can achieve a maximum throughput of 3.5 Mbit/sec on a coast-to-coast US (100 msec RTT) link.

One solution is private point-to-point connections between cluster instances. These private networks offer significant freedom to cluster operators. In particular, much higher performance than the default TCP/IP performance is possible. Indeed, the CHART system (Bavier et al, 2006; Brassil et al, 2009) demonstrated line rate performance over arbitrary latencies given explicit line rate information. This can be obtained with special-purpose equipment at each endpoint; it is guaranteed on a private network.

Since mid-2010 we have been operating a cloud over a private network between Northwestern University, HP Labs in Palo Alto, CA, and the University of California, San Diego using the CAVE Wave on the National Lambda Rail and the Global Lambda Integrated Facility. In directed throughput tests we have measured direct TCP/IP connections of over 5 Gb/sec using off-the-shelf endpoint and switching equipment (Lee et al, 2010). We intend to move this capability into production use in late 2011.

6. AN ARCHITECTURAL STACK

In this position paper, we have enumerated a number of issues and our solutions to them. Some (the Slice-based Facility Architecture, delegation, TCP acceleration over private networks) we have brought into being in our experimental cloud application. The others, including our distributed query architecture and secure, bare-metal execution environment, are under intense development.

We have largely built our prototype system from existing components: cluster managers such as Eucalyptus and PlanetLab, distributed programming environments (Hadoop, Sector/Sphere) and distributed query interfaces (Pig, Astrolabe). This is deliberate, both for ease of implementation and, far more important, ease of adoption.

The Computation Cloud must be ubiquitous; in order for it to be ubiquitous, it must be based on standards, *de jure* if possible, *de facto* by necessity. Every successful infrastructure has grown by standardizing and formalizing existing practice. The Web threw a hypertext skin over ftp; PlanetLab canonized virtual machines on Linux; SMTP/POP standardized sendmail. Our stack is no more than codification of common practice:

- Virtualization: Xen VMs running over and under Linux
- Cluster and VM Management: Eucalyptus augmented by the Slice Facility Architecture toolset
- Safe, high-performance programming environment: Google NaCl using Restricted Python (RePy)
- Wide-area programming environments and query systems: Hadoop, Pig, Sector, Sphere, and Astrolabe
- Delegation as a border primitive.

Though different in application, this resembles the classic wedding cake of the TCP/IP stack. We show the notional architecture in figure 1.

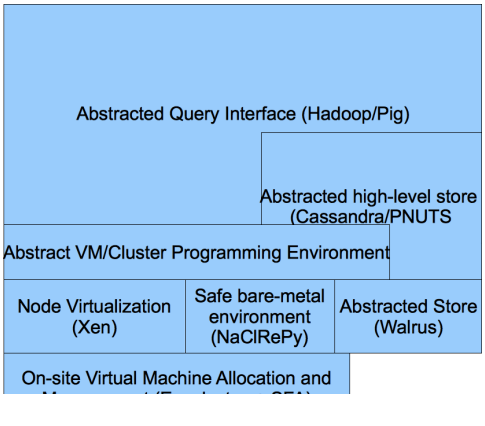

Figure 1: Notional Architecture.

In the figure, it's easy to see the layered services provided, from virtual machine/bare metal allocation to safe programming (alternately provided by virtualization and safe/restricted environments) to distributed query environments. Though our implementation is not the ideal stack by any means, exploration of this and similar stacks will yield a distributed, usable cloud computing environment analogous to the world wide web environment for data transfer.

Not shown here, but implicit, is the delegation primitive over the Slice Federation Architecture. One may picture this as intercepting the pictured stack at any level: it forms a ubiquitous authorization primitive at each level of the stack.

7. RELATED WORK

We have clearly built on a large number of existing pieces of work, from Xen to Eucalyptus and PlanetLab, to NaCl and RePy, to Hadoop, Sector, Sphere, and Pig. These are less related work than inspirations and building blocks for TransCloud.

Most closely related to our research is the Orca control framework of GENI, led by Jeff Chase and Ilia Baldine of Duke University and the Renaissance Computing Institute (RENCI). Orca similarly incorporates a unified SFA/Eucalyptus architecture and private networking over high-performance networks. The Emulab/ProtoGENI project under the late Jay LePreau and currently led by Robert Ricci has led the way to allocation of virtual networks.

8. CONCLUSIONS

In this paper, we've discussed the requirements for federated, transcontinental cloud architectures over networks of varying connectivity, with edge nodes and clusters of varying capability. We've designed a prototype services and protocol stack for such Clouds, and are constructing an initial three-site implementation under the auspices of the GENI project.

We stress this represents only a first attempt at an infrastructure. Further, this field continues under rapid development. Eucalyptus was simply our first cloud manager of choice. Other recent entrants

include Tashi (Kosuch et al., 2009), Nimbus (Keahey, 2008), and OpenStack (http://www.openstack.org). These have different strengths and weaknesses, and are optimized for different envrionments. For example, Tashi is tuned for big data. Cloud programming and query infrastructures are also undergoing rapid development, and we expect a plethora of programming environments in the coming years.

Our interest is not in presenting the notional architecture in figure 1 as the optimum, but in identifying common layers across multiple architectures and abstracting the interfaces as common APIs. In this we take inspiration from the network community, which has enjoyed enormous success from standardizing protocol specifications while retaining implementation freedom. We seek the minimum required APIs to permit easy interoperation. Our current standard is the Slice Facility Architecture, with a delegate primitive. We will continue to build on this and develop it in the coming years, and expand the TransCloud facility.

ACKNOWLEDGEMENTS

We are indebted to our colleagues on the GENI/FIRE projects for their inspiration, conversations, and constant encouragement. We have already mentioned Jeff Chase and Ilia Baldine, whose parallel project has provided a constant source of inspiration; Chip Elliot, Aaron Falk, Mark Berman, Heidi Dempsey and Vic Thomas of the GENI Project Office, for inspiration and support. Justin Cappos of the University of Washington developed Seattle and has been a constant source of advice and counsel. Paul Muller of the University of Kaiserslautern and the G-Lab project was invaluable in bringing in early experimenters, and together with Michael Zink of the University of Massachusetts provided the data stores used in our pilot query project. The SFA-based Eucalytpus was first used for a Cloud transcoding service suggested and inspired by Ericsson research. We are indebted to Jim Chen and Fei Yeh of Northwestern, Eric Wu and Narayan Krishnan of HP Research Engineering and System Support for IT support, Dejan Milojicic of the Open Cirrus project, and Tony Mackey of the HP Strategic Innovation Office. This research has been supported by the GENI Project office under contract 1779B

REFERENCES

Bavier, A., et. al. 2004. “Operating system support for planetary-scale network services”, *Proceedings NSDI*, 2004

Barham, P., et. al. 2003. “Xen and the art of virtualization”, *Proceedings SOSP,* 2003

Peterson, L., et. al. 2007 “Slice-based facility architecture”, http://www.cs.princeton.edu/~llp/arch_abridged.pdf, 2007

Brett, P., et. al. 2004. “A Shared Global Event Propagation System to Enable Next Generation Distributed Services”, *Proceedings WORLDS,* 2004

Dean, J. and Ghernawat, S. 2004. “MapReduce: Simplified Data Processing on Large Clusters”, *Proceedings of OSDI 2004*, December, 2004.”

Borthakur, D., 2009. “The Hadoop Distributed File System: Architecture and Design”, http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf

Cooper, B. F., et. al. 2010. “PNUTS: Yahoo!’s Hosted Data Serving Platform”, http://research.yahoo.com/files/pnuts.pdf

E. Evans. 2010. “Cassandra by Example”, http://www.rackspacecloud.com/blog/2010/05/12/cassandra-by-example/

Paul Brown. 2001 *Object Relational Database Development - A Plumber's Guide*, 2001, Prentice-Hall, Upper Saddle River, NJ 07458

2010. “Mobile Phone Kit to Diagnose STDs”, *The Guardian*, http://www.guardian.co.uk/uk/2010/nov/05/new-test-mobile-phones-diagnose-stds, 2010

Cooperative Atmospheric Sensing Apparatus (CAA), 2010. University of Massachusetts, http://www.casa.umass.edu/

Gu, Y., Lu, L., Grossman, R., and Yoo, Y.. 2010. “Processing Massived Sized Graphs using Sector/Sphere” , *Proceedings 3rd Workshop on Many-Task Computing on Grids and Supercomputers*, co-located with SC10, New Orleans, LA, Nov. 15, 2010.

Gu, Y. and Grossman, R.. 2009. “Lessons Learned From a Year's Worth of Benchmarks of Large Data Clouds,” *Proceedings 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, co-located with SC09, Portland, Oregon -- November 16th, 2009.

Gu, Y. and Grossman, R. 2009 “Sector and Sphere: The Design and Implementation of a High Performance Data Cloud”, *Theme Issue of the Philosophical Transactions of the Royal Society A: Crossing Boundaries: Computational Science, E-Science and Global E-Infrastructure*, 28 June 2009 vol. 367 no. 1897 2429-2445.

Gu, Y. and Grossman, R. 2008. “Exploring Data Parallelism and Locality in Wide Area Networks”, *Proceedings of the Workshop on Many-task Computing on Grids and Supercomputers (MTAGS*), co-located with SC08, Austin, TX. Nov. 2008.

Gu, Y. and Grossman, R. 2008. “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere”, *Proceedings SIGKDD 2008*, Las Vegas, NV, Aug. 2008.

511 Service, http://www.511.org

Van Renesse, R., Birman, K., and Vogels, W., 2003. “Astrolabe: A robust and scalable technology for

distributed system monitoring, management, and data mining", *ACM Transactions on Computer Systems*, May, 2003

Cappos, J., Dadgar, A., Rasley, J., Samuel, J., IBeschastnikh, I., Barsan, C. , Krishnamurthy, A., and Anderson, T. 2010. "Retaining Sandbox Containment Despite Bugs in Privileged Memory-Safe Code." *Conference on Computer and Communications Security (CCS '10)*. Chicago, IL, 2010.

Cappos, J., Beschastnikh, I., Krishnamurthy, A., and Anderson. T. 2009. "Seattle: A Platform for Educational Cloud Computing." *SIGCSE '09.*

Yee, B., Sehr, D., Dardyk, G., Chen, B., Muth, R., Ormandy, T., Okasaka, T. Narula, N., and Fullagar, N. 2009. "Native Client: A Sandbox for Portable, Untrusted x86 Native Code", *IEEE Symposium on Security and Privacy (Oakland'09)*, 2009.

Matthews, C., Cappos, J.,, Coady, Y., Hartman, J., Jacky, J., and McGeer, R. 2010. "NanoXen : Better Systems Through Rigorous Containment and Active Modeling", *OSDI 2010* (Poster).

Yalagandula, P., Sharma, P., Banerjee, S., Lee, S-J., and Sujoy Basu, S. 2006."S3: A Scalable Sensing Service for Monitoring Large Networked Systems", *Proceedings of ACM INM 2006(in conjunction with Sigcomm 2006),* Pisa, Italy, September 2006.

The Neptune Ocean Observatory, http://www.neptunecanada.ca

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2005. "TinyDB: An Acqusitional Query Processing System for Sensor Networks" . *ACM TODS*, 2005.

Oppenheimer, D., et al., 2004. "Distributed Resource Discovery on PlanetLab with SWORD", Proceedings *WORLDS*, 2004.

Massie, M. et. al. "The Ganglia Distributed Monitoring System: Design, Implementation And Experience", *Parallel Computing*, 2003.

Apache Project. 2010. *Pig,* http://pig.apache.org

Bavier, A., et al. 2006. "Increasing TCP Throughput with an Enhanced Internet Control Plane, *Proceedings MILCOMM 2006*

Brassil J., et. al, The CHART System: A High-Performance, Fair Transport Architecture Based on Explicit-Rate Signaling, *ACM SIGOPS Review*, February, 2009

R. McGeer, B.L. Mark, J. Brassil, P. Sharma, P. Yalagandula, S. Schwab, and S. Zhang, 2009. "The Case for Service Overlays," *Proc. 18th IEEE Int. Conf. on Computer Communications and Networks* (ICCCN'09), San Francisco, CA, Aug. 2009.

Lee, J., Sharma, P., Tourrilhes, J., McGeer, R., Brassil, J., and Bavier, A. 2010. "Network Integrated Transparent TCP Accelerator", *Proceedings AINA 2010*, May 2010

Nurmi, D. et al. 2009. "The Eucalyptus Open-Source Cloud-Computing System", *CCGRID '09*, 2009

Kosuch, M., et al. 2009. "Tashi: Location-aware Cluster Management", *First Workshop on Automated Control for Datacenters and Clouds (ACDC'09)*, June 2009

Keahey, K., Freeman, T. 2008. "Contextualization: Providing One-Click Virtual Clusters", 2008 Fourth IEEE International Conference on eScience, pp.301-308. doi:10.1109/eScience.2008.82