Volume 6, Issue 2, February 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comparison on Cloud Computing and Grid Computing

Chhabi Sethi, Prof. Sateeth Kuamr Pradhan
P.G. Department of Computer Science and Applications
Utkal University, Odisha, India

Abstract— In this current world Cloud Computing has become very excitement concept after Web 2.0. There are so many different definitions for Cloud Computing and there is no consensus on what a Cloud is. On the other hand, this is not a completely new concept; it has very complicated connection to the relatively new but twenty-year established Grid Computing paradigm, and other relevant technologies such as utility computing, cluster computing, and distributed systems. Here we strive to compare and contrast Cloud Computing with Grid Computing from various angles with the essential characteristics.

Keywords— Cloud Computing, Grid Computing, Cluster Computing, Utility Computing, Service Computing, Distributed Computing.

I. INTRODUCTION

Cloud Computing is indicating at a future in which we won't compute on local computers, but on centralized facilities operated by third-party compute and storage utilities. It has begun to emerge as a very excitement in both industry and academia. It represents a new business model and computing environment, which enables on demand provisioning of computational and storage resources. Actually, this is not a new idea. In 1961 computing pioneer John McCarthy predicted that "computation may someday be organized as a public utility"— and it has occurred today.

In 1990 the term Grid was came to describe the technologies that would allow consumers to obtain computing power on demand. Ian Foster and others postulated that by standardizing the protocols used to request computing power, we could spur the creation of a Computing Grid, analogous in form and utility to the electric power grid. Researchers subsequently developed these ideas in many exciting ways, producing for example large-scale federated systems (TeraGrid, Open Science Grid, caBIG, EGEE, Earth System Grid) that provide not just computing power, but also data and software, on demand. Standards organizations (e.g., OGF, OASIS) defined relevant standards.

So one question is arising in our mind that is "Cloud Computing" just a new name for Grid?

***If we will say yes*:** The vision is the same to reduce the cost of computing, increase reliability, and increase flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party.

But, if we will say no: Now things are different before 15 years ago. We have a new need to analyze massive data, thus motivating greatly increased demand for computing. Having realized the benefits of moving from mainframes to commodity clusters, we find that those clusters are quite expensive to operate. We have low-cost virtualization. And, above all, we have multiple billions of dollars being spent by the likes of Amazon, Google, and Microsoft to create real commercial large-scale systems containing hundreds of thousands of computers. The prospect of needing only a credit card to get on-demand access to 100,000+ computers in tens of data centers distributed throughout the world resources that can be applied to problems with massive, potentially distributed data.

Defining Cloud Computing

Throughout this working text, we cite [1], which defines cloud computing as: "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet."

There are some points in this definition. First, Cloud Computing is a specialized distributed computing paradigm and it differs from traditional ones in that: 1) it is massively scalable, 2) it can be encapsulated as an abstract entity that delivers different levels of services to customers outside the Cloud, 3) it is driven by economies of scale [2], and 4) the services can be dynamically configured (via virtualization or other approaches) and delivered on demand.

Governments, research institutes, and industry leaders are rushing to adopt Cloud Computing to solve their ever-increasing computing and storage problems arising in the Internet Age. There are three main factors contributing to the interests in Cloud Computing: 1) rapid decrease in hardware cost and increase in computing power and storage capacity, and the advent of multi-core architecture and modern supercomputers consisting of hundreds of thousands of cores. 2) exponentially growing data size in scientific instrumentation/simulation. 3) wide-spread adoption of Services Computing and Web 2.0 applications.

Clouds, Grids, and Distributed Systems

Our definition of Cloud Computing overlaps with many existing technologies, such as Grid Computing, Utility Computing, Services Computing, and distributed computing in general. We argue that Cloud Computing not only overlaps with Grid Computing, it is indeed evolved out of Grid Computing and relies on Grid Computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources to one that is economy based aiming to deliver more abstract resources and services. As for Utility Computing, it is not a new paradigm of computing infrastructure. Rather, it is a business model in which computing resources, such as computation and storage, are packaged as metered services similar to a physical public utility, such as electricity and public switched telephone network. Utility computing is typically implemented using other computing infrastructure (e.g. Grids) with additional accounting and monitoring services. A Cloud infrastructure can be utilized internally by a company or exposed to the public as utility computing.

See Figure 1 for an overview of the relationship between Clouds and other domains that it overlaps with. Web 2.0 covers almost the whole spectrum of service-oriented applications, where Cloud Computing lies at the large-scale side. Supercomputing and Cluster Computing have been more focused on traditional non-service applications. Grid Computing overlaps with all these fields where it is generally considered of lesser scale than supercomputers and Clouds.

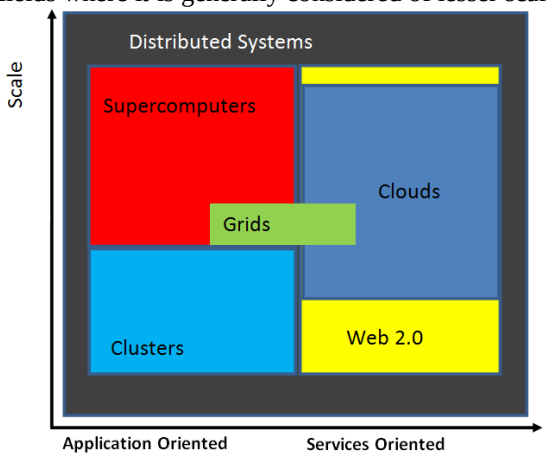

Figure 1: Grids and Clouds Overview

Grid Computing aims to "enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [3] [4]. There are also a few key features to this definition: First of all, Grids provide a distributed computing paradigm or infrastructure that spans across multiple virtual organizations where each virtual organization can consist of either physically distributed institutions or logically related groups. The Globus Toolkit [5] [21], is to build a uniform computing environment from diverse resources by defining standard network protocols and providing middleware to mediate access to a wide range of heterogeneous resources. Globus addresses various issues such as security, resource discovery, resource provisioning and management, job scheduling, monitoring, and data management.

After five years ago, Ian Foster gave a three point checklist [6] to help define what is, and what not a Grid is: 1) coordinates resources that are not subject to centralized control, 2) uses standard, open, general-purpose protocols and interfaces, and 3) delivers non-trivial qualities of service.

II. COMPARING GRIDS AND CLOUDS SIDE-BY-SIDE

This section aims to compare Grids and Clouds across a wide variety of perspectives, from architecture, security model, business model, programming model, virtualization, data model, compute model, to provenance and applications. We also outline a number of challenges and opportunities that Grid Computing and Cloud Computing bring to researchers and the IT industry, most common to both, but some are specific to one or the other.

Business Model

Traditional business model for software has been a one-time payment for unlimited use of the software. In a cloud-based business model, a customer will pay to the provider on a consumption basis, very much like the utility companies charge for basic utilities such as electricity, gas, and water, and the model relies on economies of scale in order to drive prices down for users and profits up for providers. Today, Amazon essentially provides a centralized Cloud consisting of Compute Cloud EC2 and Data Cloud S3.

The business model for Grids is project-oriented in which the users or community represented by that proposal have certain number of service units (i.e. CPU hours) they can spend. For example, the TeraGrid operates in this fashion, and requires increasingly complex proposals be written for increasing number of computational power. The TeraGrid has more than a dozen Grid sites, all hosted at various institutions around the country. When an institution joins the TeraGrid with a set of resources, it knows that others in the community can now use these resources across the country. It also acknowledges the fact that it gains access to a dozen other Grid sites.

There are also endeavors to build a Grid economy for a global Grid infrastructure that supports the trading, negotiation, provisioning, and allocation of resources based on the levels of services provided, risk and cost, and users' preferences;

so far, resource exchange (e.g. trade storage for compute cycles), auctions, game theory based resource coordination, virtual currencies, resource brokers and intermediaries, and various other economic models have been proposed and applied in practice [7][21].

Architecture

Grids started off in the mid-90s to address large-scale computation problems using a network of resource-sharing commodity machines that deliver the computation power affordable only by supercomputers and large dedicated clusters at that time. The major motivation was that these high performance computing resources were expensive and hard to get access to, so the starting point was to use federated resources that could comprise compute, storage and network resources from multiple geographically distributed institutions, and such resources are generally heterogeneous and dynamic.

Grids provide protocols and services at five different layers as identified in the Grid protocol architecture (see Figure 2). At the fabric layer, Grids provide access to different resource types such as compute, storage and network resource, code repository, etc. Grids usually rely on existing fabric components, for instance, local resource managers. General purpose components such as GARA (general architecture for advanced reservation) [21], and specialized resource management services such as Falkon [8] (although strictly speaking, Falkon also provides services beyond the fabric layer).

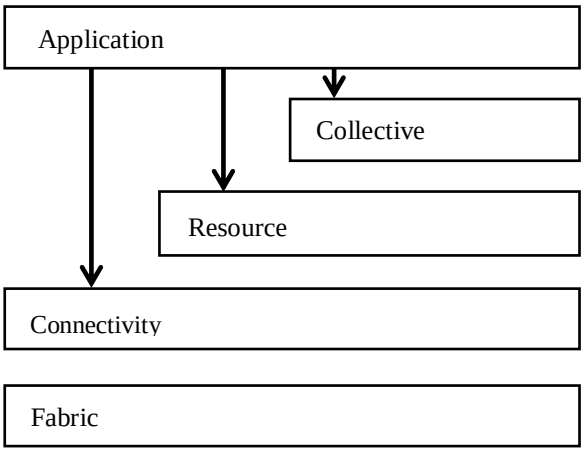

Figure 2: Grid Protocol Architecture

The ***connectivity layer*** defines core communication and authentication protocols for easy and secure network transactions. The GSI (Grid Security Infrastructure) [9] protocol underlies every Grid transaction. The ***resource layer*** defines protocols for the publication, discovery, negotiation, monitoring, accounting and payment of sharing operations on individual resources. The GRAM (Grid Resource Access and Management) [21] protocol is used for allocation of computational resources and for monitoring and control of computation on those resources, and GridFTP [21] for data access and high-speed data transfer. The ***collective layer*** captures interactions across collections of resources, directory services such as MDS (Monitoring and Discovery Service) [10] allows for the monitoring and discovery of virtual organization resources. The ***application layer*** comprises whatever user applications built on top of the above protocols and APIs and operate in VO environments. Two examples are Grid workflow systems, and Grid portals.

Clouds are developed to address Internet-scale computing problems where some assumptions are different from those of the Grids. Clouds are usually referred to as a large pool of computing and/or storage resources, which can be accessed via standard protocols via an abstract interface.

We define a four-layer architecture for Cloud Computing in comparison to the Grid architecture, composed of 1) fabric, 2) unified resource, 3) platform, and 4) application Layers.

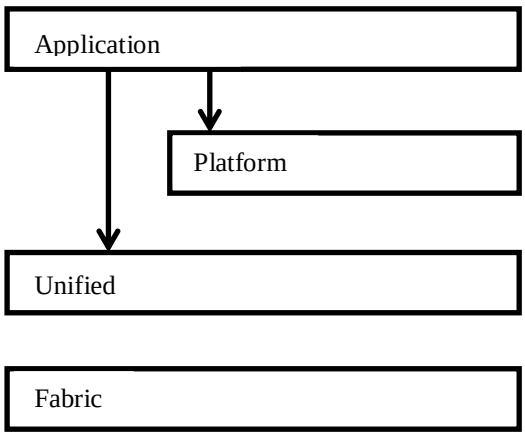

Figure 3: Cloud Architecture

The *fabric layer* contains the raw hardware level resources, such as compute resources, storage resources, and network resources. The *unified resource layer* contains resources that have been abstracted/encapsulated (usually by virtualization) so that they can be exposed to upper layer and end users as integrated resources, for instance, a virtual computer/cluster, a logical file system, a database system, etc. The *platform layer* adds on a collection of specialized tools, middleware and services on top of the unified resources to provide a development and/or deployment platform. Finally, the *application layer* contains the applications that would run in the Clouds.

Clouds in general provide services at three different levels (*IaaS*, *PaaS*, and *Saas* [1]) as follows, although some providers can choose to expose services at more than one level. *Infrastructure as a Service (IaaS)* [11] provisions hardware, software, and equipment to deliver software application environments with a resource usage-based pricing model. Infrastructure can scale up and down dynamically based on application resource needs. Typical examples are Amazon EC2 (Elastic Cloud Computing) Service [12] and S3 (Simple Storage Service) [13] where compute and storage infrastructures are open to public access with a utility pricing model. It is an open source Cloud implementation that provides a compatible interface to Amazon's EC2, and allows people to set up a Cloud infrastructure at premise and experiment prior to buying commercial services.

Platform as a Service (PaaS) [14] offers a high-level integrated environment to build, test, and deploy custom applications. Generally, developers will need to accept some restrictions on the type of software they can write in exchange for built-in application scalability. An example is Google's App Engine [15], which enables users to build Web applications on the same scalable systems that power Google applications.

Software as a Service (SaaS) [14] delivers special-purpose software that is remotely accessible by consumers through the Internet with a usage-based pricing model. Salesforce is an industry leader in providing online CRM (Customer Relationship Management) Services. Live Mesh from Microsoft allows files and folders to be shared and synchronized across multiple devices.

Resource Management

This section describes the resource management found in Grids and Clouds, covering topics such as the compute model, data model, virtualization, monitoring, and provenance. These topics are extremely important to understand the main challenges that both Grids and Clouds face today, and will have to overcome in the future.

Compute Model: Most Grids use a batch-scheduled compute model, in which a local resource manager (LRM) manages the compute resources for a Grid site, and users submit batch to request some resources for some time. Many Grids have policies in place that enforce these batch jobs to identify the user and credentials under which the job will run for accounting and security purposes, the number of processors needed, and the duration of the allocation.

Due to the expensive scheduling decisions, data staging in and out, and potentially long queue times, many Grids don't natively support interactive applications.

Cloud Computing compute model will likely look very different, with resources in the Cloud being shared by all users at the same time. This should allow latency sensitive applications to operate natively on Clouds, although ensuring a good enough level of QoS is being delivered to the end users will not be trivial, and will likely be one of the major challenges for Cloud Computing as the Clouds grow in scale, and number of users.

Data Model: While some people boldly predicate that future Internet Computing will be towards Cloud Computing centralized, in which storage, computing, and all kind of other resources will mainly be provisioned by the Cloud. Cloud Computing and Client Computing will coexist and evolve hand in hand, while data management will become more and more important for both Cloud Computing and Client Computing with the increase of data-intensive applications.

The critical role of Cloud Computing goes without saying, but the importance of Client Computing cannot be overlooked either for several reasons: 1) For security reasons, people might not be willing to run mission-critical applications on the Cloud and send sensitive data to the Cloud for processing and storage; 2) Users want to get their things done even when the Internet and Cloud are down or the network communication is slow; 3) With the advances of multi-core technology, the coming decade will bring the possibilities of having a desktop supercomputer with 100s to 1000s of hardware threads/cores.

Data Locality: As CPU cycles become cheaper and data sets double in size every year, the main challenge for efficient scaling of applications is the location of the data relative to the available computational resources moving the data repeatedly to distant CPUs is becoming the bottleneck. Google's MapReduce [16] system runs on top of the Google File System, within which data is loaded, partitioned into chunks, and each chunk replicated. Thus data processing is collocated with data storage when a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk, and then schedules a "map" process on that node, so that data locality is exploited efficiently. In Grids, data storage usually relies on a shared file system where data locality cannot be easily applied.

Combining compute and data management: Even more critical is the combination of the compute and data resource management, which leverages data locality in access patterns to minimize the amount of data movement and improve end application performance and scalability. It is important to schedule computational tasks close to the data, and to understand the costs of moving the work as opposed to moving the data. Data-aware schedulers and dispersing data close to processors is critical in achieving good scalability and performance. Finally, as the number of processor-cores is increasing, there is an ever-growing emphasis for support of high throughput computing with high sustainable dispatch and execution rates. Grids have been making progress in combining compute and data management with data-aware schedulers, but we believe that Clouds will face significant challenges in handling data-intensive applications without serious efforts invested in harnessing the data locality of application access patterns.

Virtualization: Virtualization has become an indispensable ingredient for almost every Cloud. The most obvious reasons are for abstraction and encapsulation. Just like threads were introduced to provide users the "illusion" as if the computer were running all the threads simultaneously, and each thread were using all the available resources, Clouds need to run multiple user applications, and all the applications appear to the users as if they were running simultaneously and could use all the available resources in the Cloud. Virtualization provides the necessary abstraction such that the underlying fabric (raw compute, storage, network resources) can be unified as a pool of resources and resource overlays (e.g. data storage services, Web hosting environments) can be built on top of them.

Grids do not rely on virtualization as much as Clouds do, but that might be more due to policy and having each individual organization maintain full control of their resources. However, there are efforts in Grids to use virtualization as well. A virtual workspace is an execution environment that can be deployed dynamically and securely in the Grid. However, over the past few years, processor manufacturers such as AMD and Intel have been introducing hardware support for virtualization, which is helping narrow the performance gap between applications performance on virtualized resources as it compares with that on traditional operating systems without virtualization.

Monitoring: Another challenge that virtualization brings to Clouds is the potential difficulty in fine-control over the monitoring of resources. Although many Grids (such as TeraGrid) also enforce restrictions on what kind of sensors or long-running services a user can launch, Cloud monitoring is not as straightforward as in Grids, because Grids in general have a different trust model in which users via their identity delegation can access and browse resources at different Grid sites, and Grid resources are not highly abstracted and virtualized as in Clouds. In a Cloud, different levels of services can be offered to an end user, the user is only exposed to a predefined API, and the lower level resources are opaque to the user (especially at the PaaS and SaaS level, although some providers may choose to expose monitoring information at these levels). The user does not have the liberty to deploy her own monitoring infrastructure, and the limited information returned to the user may not provide the necessary level of details for her to figure out what the resource status is. The same problems potentially exist for Cloud developers and administrators, as the abstract/unified resources usually go through virtualization and some other level of encapsulation, and tracking the issues down the software/hardware stack might be more difficult.

Programming Model

Although programming model in Grid environments does not differ fundamentally from traditional parallel and distributed environments, it is obviously complicated by issues such as multiple administrative domains such as large variations in resource heterogeneity, stability and performance, exception handling in highly dynamic environments, etc. Grids primarily target large-scale scientific computations, so it must scale to leverage large number/amount of resources, and we would also naturally want to make programs run fast and efficient in Grid environments, and programs also need to finish correctly, so reliability and fault tolerance must be considered.

We briefly discuss here some general programming models in Grids. MPI (Message Passing Interface) [17] is the most commonly used programming model in parallel computing, in which a set of tasks use their own local memory during computation and communicate by sending and receiving messages. Coordination languages also allow a number of possibly heterogeneous components to communicate and interact, offering facilities for the specification, interaction, and dynamic composition of distributed (concurrent) components.

MapReduce [16] is only yet another parallel programming model, providing a programming model and runtime system for the processing of large datasets, and it is based on a simple model with just two key functions: "map" and "reduce," borrowed from functional languages. The map function applies a specific operation to each of a set of items, producing a new set of items and a reduce function performs aggregation on a set of items. The MapReduce runtime system automatically partitions input data and schedules the execution of programs in a large cluster of commodity machines. The system is made fault tolerant by checking worker nodes periodically and reassigning failed jobs to other worker nodes. Sawzall is an interpreted language that builds on MapReduce and separates the filtering and aggregation phases for more concise program specification and better parallelization. Hadoop [18] is the open source implementation of the MapReduce model, and Pig is a declarative programming language offered on top of Hadoop.

Application Model

Grids generally support many different kinds of applications, ranging from high performance computing (HPC) to high throughput computing (HTC). HPC applications are efficient at executing tightly coupled parallel jobs within a particular machine with low-latency interconnects and are generally not executed across a wide area network Grid. These applications typically use message passing interface (MPI) to achieve the needed inter-process communication. On the other hand, Grids have also seen great success in the execution of more loosely coupled applications that tend to be managed and executed through workflow systems or other sophisticated and complex applications.

On the other hand, Cloud Computing could in principle cater to a similar set of applications. The one exception that will likely be hard to achieve in Cloud Computing (but has had much success in Grids) are HPC applications that require fast and low latency network interconnects for efficient scaling to many processors. As Cloud Computing is still in its infancy, the applications that will run on Clouds are not well defined, but we can certainly characterize them to be loosely coupled, transaction oriented (small tasks in the order of milliseconds to seconds), and likely to be interactive (as opposed to batch scheduled as they are currently in Grids).

Another emerging class of applications in Grids is scientific gateways, which are front-ends to a variety of applications that can be anything from loosely-coupled to tightly-coupled. *A Science Gateway is a community-developed set of tools, applications, and data collections that are integrated via a portal or a suite of applications.*

Security Model

Clouds mostly comprise dedicated data centers belonging to the same organization, and within each data center, hardware and software configurations and supporting platforms are in general more homogeneous as compared with those in Grid environments. Interoperability can become a serious issue for cross-data center, cross-administration domain interactions, imagine running your accounting service in Amazon EC2 while your other business operations on Google infrastructure. Grids however build on the assumption that resources are heterogeneous and dynamic, and each Grid site may have its own administration domain and operation autonomy. Thus, security has been engineered in the fundamental Grid infrastructure.

Security is one of the largest concerns for the adoption of Cloud Computing. We outline seven risks a Cloud user should raise with vendors before committing [19]: 1) *Privileged user access:* sensitive data processed outside the enterprise needs the assurance that they are only accessible and propagated to privileged users; 2) *Regulatory compliance:* a customer needs to verify if a Cloud provider has external audits and security certifications and if their infrastructure complies with some regulatory security requirements; 3) *Data location:* since a customer will not know where her data will be stored, it is important that the Cloud provider commit to storing and processing data in specific jurisdictions and to obey local privacy requirements on behalf of the customer; 4) *Data segregation:* one needs to ensure that one customer's data is fully segregated from another customer's data; 5) *Recovery:* it is important that the Cloud provider has an efficient replication and recovery mechanism to restore data if a disaster occurs; 6) *Investigative support:* Cloud services are especially difficult to investigate, if this is important for a customer, then such support needs to be ensured with a contractual commitment; and 7) *Long-term viability:* your data should be viable even the Cloud provider is acquired by another company.

III. CONCLUSIONS

In this work, we show that Clouds and Grids share a lot commonality in their vision, architecture and technology, but they also differ in various aspects such as security, programming model, business model, compute model, data model, applications, and abstractions. We also identify challenges and opportunities in both fields. We believe a close comparison such as this can help the two communities understand, share and evolve infrastructure and technology within and across, and accelerate Cloud Computing from early prototypes to production systems.

ACKNOWLEDGMENT

This work is supported by the P. G. Department of Computer Science and Applications, Utkal University and guided by Prof. Sateesh Kumar Pradhan.

REFERENCES

[1] Z. Xiao, Y. Xiao, "Security and Privacy in Cloud Computing" IEEE Communications Surveys & Tutorials, 2012.
[2] J. Silvestre. "Economies and Diseconomies of Scale," The New Palgrave: A Dictionary of Economics, v. 2, pp. 80–84, 1987.
[3] I. Foster, C. Kesselman, S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. The Intl. Jrnl. of High Performance Computing Applications, 15(3):200--222, 2001.
[4] I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002.
[5] I. Foster. "Globus Toolkit Version 4: Software for ServiceOriented Systems." IFIP Int. Conf. on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006 (2002) The IEEE website. [Online]. Available: http://www.ieee.org/
[6] I. Foster. What is the Grid? A Three Point Checklist, July 2002.
[7] R. Buyya, K. Bubendorfer. "Market Oriented Grid and Utility Computing", Wiley Press, New York, USA, 2008.
[8] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: a Fast and Light-weight tasK executiON framework", IEEE/ACM SuperComputing 2007.
[9] The Globus Security Team. "Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective," Technical Report, Argonne National Laboratory, MCS, 2005.
[10] J. M. Schopf, I. Raicu, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D'Arcy. "Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4", Technical Report, Argonne National Laboratory, MCS Preprint #ANL/MCS-P1315-0106, 2006.
[11] Cloud Security Alliance (CSA). "Top Threats to Cloud Computing V 1.0," released March 2010.
[12] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2008.
[13] Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3, 2008.
[14] "What is Cloud Computing?", Whatis.com. http://searchsoa.techtarget.com/sDefinition/0, ,sid26_gci1287881,00.html, 2008

[15] Google App Engine, http://code.google.com/appengine/, 2008.

[16] J. Dean, S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters", Sixth Symposium on Operating System Design and Implementation (OSDI04), 2004.

[17] M.P.I. Forum. "MPI: A Message-Passing Interface Standard", 1994

[18] Hadoop, hadoop.apache.org/core/, 2008.

[19] Z. Xiao and Y. Xiao, "Accountable MapReduce in Cloud Computing," Proc. The IEEE International Workshop on Security in Computers, Networking and Communications (SCNC 2011) in conjunction with IEEE INFOCOM 2011.

[20] Ganglia, http://sourceforge.net/projects/ganglia/, 2008.

[21] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," Grid Computing Environments Workshop, 2008. GCE'08, 2008, pp. 1-10.

