Comparative Analysis of Cloud and Grid Computing Paradigms

Mona Saad Khalil Morgan

Information System Department, Faculty of Management & Economic and information system
Misr University for Science & Technology, Cairo, Egypt.

khalilmona2020@hotmail.com

Abstract

Despite the tremendous development within the information technology systems, we still need the continued development of these systems whereas, the problem is that there are many obstacles and problems facing the application of information and communication technology systems. The major motivation was that these high performance computing resources have been 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. Also the main challenge for efficient scaling of applications is the location of the data relative to the available computational resources while moving the data repeatedly to distant CPUs is becoming the bottleneck. [46]

This research will throw light on the 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, we envision that the next-generation Internet Computing will take the triangle model shown in Figure 4: Internet Computing will be centralized around Data, Clouding Computing, as well as Client Computing. We propose an efficient approach 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, The application results reveal the effectiveness of proposed method to achieve a better cost 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. Furthermore, many end-users will have various hardware driven end-functionalities, such as visualization and multimedia playback, which will typically run locally.

Keywords: *Cloud Computing, Grid Computing*, *Grid Resource Access and Management, Globus protocol, Message Passing Interface***,** *Web Services Resource Framework*

1. Introduction

This paper strives to compare and contrast Cloud Computing with Grid Computing from various angles (architecture, security model, business model, programming model, virtualization, data model, compute model, to provenance and applications) and give insights into the essential characteristics of both, also we describe the resource management found in Grids and Clouds, covering topics such as the compute model, data model, virtualization, monitoring, and provenance. with the hope to paint a less cloudy picture of what Clouds are, what kind of applications can Clouds expect to support, and what challenges Clouds are likely to face in the coming years as they gain momentum and adoption

Also we identify that the vision for Clouds and Grids are similar, details and technologies used may differ, but the two communities are struggling with many of the same issues . Cloud Computing and Client Computing will coexist and evolve hand in hand, while data management (mapping, partitioning, querying, movement, caching, replication, etc) will become more and more important for both Cloud Computing and Client Computing with the increase of data-intensive applications. Cloud Computing is hinting at a future in which we won't compute on local computers, but on centralized facilities operated by third-party compute and storage utilities .

We envision that the next-generation Internet Computing will take the triangle model, Internet Computing will be centralized around Data, Clouding Computing, as well as Client Computing. Cloud Computing and Client Computing will coexist and evolve hand in hand, while data management (mapping, partitioning, querying, movement, caching, replication, etc) will become more and more important for both Cloud Computing and Client Computing with the increase of data-intensive applications. As Clouds mature, and more sophisticated applications and services emerge that require the use of multiple Clouds, there will be growing incentives to adopt standard interfaces that facilitate interoperability in order to capture emerging and growing markets in a saturated Cloud market.

Today, Amazon essentially provides a centralized Cloud consisting of Compute Cloud (Elastic Compute Cloud) EC2 and Amazon EC2 also uses Amazon S3 to store snapshots (backup copies) of the ... S3 bucket . Ian Foster and others posited 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. More prosaically, the term was also co-opted by industry as a marketing term for clusters. But no viable commercial Grid Computing providers emerged, at least not until recently.

So is "Cloud Computing" just a new name for Grid? In information technology, where technology scales by an order of magnitude, and in the process reinvents itself, every five years, there is no straightforward answer to such questions. 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 no: things are different now than they were 10 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 be applied to problems with massive, potentially distributed data, is exciting! So we are operating at a different scale, and operating at these new, more massive scales can demand fundamentally different approaches to tackling problems.

It also enables—indeed is often only applicable to—entirely new problems. Nevertheless, yes: the problems are mostly the same in Clouds and Grids. There is a common need to be able to manage large facilities; to define methods by which consumers discover, request, and use resources provided by the central facilities; and to implement the often highly parallel computations that execute on those resources. Details differ, but the two communities are struggling with many of the same issues.

1.1 Defining Cloud Computing

There is little consensus on how to define the Cloud [49]. We add yet another definition to the already saturated list of definitions for Cloud Computing. 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 a few key points in this definition. First, Cloud Computing is a specialized distributed computing paradigm (model); it differs from traditional ones in that:

1) It is massively scalable.
2) 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 [44].
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 surge and 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) The exponentially growing data size in scientific instrumentation/simulation and Internet publishing and archiving.
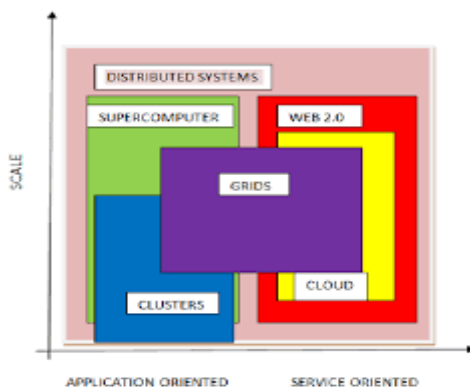3) The wide-spread adoption of Services Computing and Web

Figure 1: Grids and Clouds Overview

1.2 Applications

Grid Computing aims to "enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [18][20]. 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 (VO) where each VO can consist of either physically distributed institutions or logically related projects/groups. The goal of such a paradigm is to enable federated resource sharing in dynamic, distributed environments. The approach taken by the de facto standard implementation – The Globus Toolkit [16][23], 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. Half a decade ago, Ian Foster gave a three point checklist [19] to help define what is, and what is not a Grid:

1) coordinates resources that are not subject to centralized control,

2) Uses standard, open, general-purpose protocols and interfaces,

3) delivers non-trivial qualities of service.

Although point 3 holds true for Cloud Computing, neither point 1 nor point 2 is clear that it is the case for today's Clouds. The vision for Clouds and Grids are similar, details and technologies used may differ, but the two communities are struggling with many of the same issues.

This paper strives to compare and contrast Cloud Computing with Grid Computing from various angles and give insights into the essential characteristics of both, with the hope to paint a less cloudy picture of what Clouds are, what kind of applications can Clouds expect to support, and what challenges Clouds are likely to face in the coming years as they gain momentum and adoption.

We hope this will help both communities gain deeper understanding of the goals, assumptions, status, and directions, and provide a more detailed view of both technologies to the general audience.

2. 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. Table 1 illustrates this comparison.

Table 1: Comparing Grids and Clouds Side-by-Side

	Grids	Clouds
2.1.Business Model	The business model for Grids (at least that found in academia or government labs) 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	In a cloud-based business model, a customer will pay the provider on a consumption basis. Today, Amazon essentially provides a centralized Cloud consisting of Compute Cloud(Elastic Compute Cloud) EC2 and Amazon EC2 also uses Amazon S3 to store snapshots (backup copies) of the ... S3 bucket.
2.2 Architecture	Grids focused on integrating existing resources with their hardware, operating systems, local resource management, and security infrastructure. In order to support the creation of the so called "Virtual Organizations"— Grids define and provide a set of standard protocols, middleware, toolkits, and services built on top of these protocols. Interoperability and security are the primary concerns for the Grid infrastructure Grids usually rely on existing fabric components.	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. As Clouds mature, and more sophisticated applications and services emerge that require the use of multiple Clouds, there will be growing incentives to adopt standard interfaces that facilitate interoperability in order to capture emerging and growing markets in a saturated Cloud market.
2.3 Resource Management: Compute Model:	Most Grids use a batch-scheduled compute model. 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	With resources in the Cloud being shared by all users at the same time. Allow latency sensitive applications to operate natively on Clouds, although ensuring a good enough level of QoS (Quality of Service) 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	The importance of data has caught the attention of the Grid community for the past decade; Data Grids [10] have been specifically designed to tackle data intensive applications in Grid environments, with the concept of virtual data [22] playing a crucial role.	Many end-users will have various hardware driven end-functionalities, such as visualization and multimedia playback, which will typically run locally
Programming Model	Clouds (such as Amazon Web Services, Microsoft's Azure Services Platform) have generally adopted Web Services APIs where users access, configure and program Cloud services using pre-defined APIs exposed as Web services, and HTTP and SOAP are the common protocols chosen for such services.	some general programming models in Grids. MPI (Message Passing Interface) [36] 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. MPICH-G2(Software) [32] is a Grid enabled implementation of MPI

Follow Table 1: Comparing Grids and Clouds Side-by-Side

	Grids	Clouds
Application Model	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.	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
Security Model:	Cloud infrastructure typically rely on Web forms (over SSL to create and manage account information for end-users, and allows users to reset their passwords and receive new passwords via Emails in an unsafe and unencrypted communication. Note that new users could use Clouds relatively easily and almost instantly, with a credit card and/or email address.	Resources belonging to one user cannot be accessed by unauthorized users, and cannot be tampered during transfer; coordinated resource allocation, reservation, and sharing, taking into consideration of both global and local resource usage policies. The public-key based GSI (Grid Security Infrastructure) protocols are used for authentication, communication protection, and authorization.

2.1 Business Model

Traditional business model for software has been a one-time payment for unlimited use (usually on 1 computer) of the software.

In a cloud-based business model, a customer will pay 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 (Elastic Compute Cloud) EC2 and Amazon EC2 also uses Amazon S3 to store snapshots (backup copies) of the ... S3 bucket.

The former is charged based on per instance-hour consumed for each instance type and the later is charged by per GB-Month of storage used.

In addition, data transfer is charged by Terabytes TB / month data transfer, depending on the source and target of such transfer. The prospect of needing only a credit card to get on-demand access to 100,000+ processors in tens of data centers distributed throughout the world—resources that be applied to problems with massive, potentially distributed data, is exciting. The business model for Grids (at least that found in academia or government labs) 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 Tera Grid operates in this fashion, and requires increasingly complex proposals be written for increasing number of computational power. The Tera Grid has more than a dozen Grid sites, all hosted at various institutions around the country.

What makes an institution want to join the Tera Grid?

When an institution joins the Tera Grid 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.

This same model has worked rather well for many Grids around the globe, giving institutions incentives to join various Grids for access to additional resources for all the users from the corresponding institution.

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 [8].

2.2 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 focused on integrating existing resources with their hardware, operating systems, local resource management, and security infrastructure. In order to support the creation of the so called "Virtual Organizations"—a logical entity within which distributed resources can be discovered and shared as if they were from the same organization, Grids define and provide a set of standard protocols, middleware, toolkits, and services built on top of these protocols. Interoperability and security are the primary concerns for the Grid infrastructure as resources may come from different administrative domains, which have both global and local resource usage policies, different hardware and software configurations and platforms, and vary in availability and capacity.

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 (i.e. PBS [5], Condor [48], etc). General-purpose components such as GARA (general architecture for advanced reservation) [17], and specialized resource management services such as Falkon [40] (although strictly speaking, Falkon also provides services beyond the fabric layer).
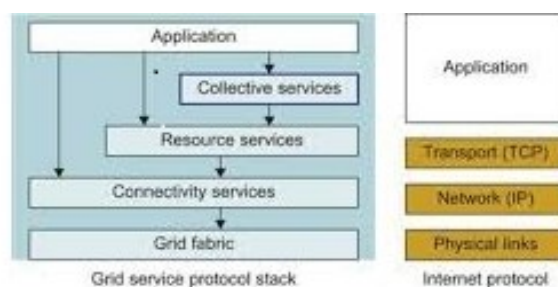
Figure 2: Grid Protocol Architecture

The connectivity layer defines core communication and authentication protocols for easy and secure network transactions. The GSI (Grid Security Infrastructure) [27] 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) [16] protocol is used for allocation of computational resources and for monitoring and control of computation on those resources, and GridFTP [2] 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) [43] allows for the monitoring and discovery of VO resources, Condor-G [24] and Nimrod-G [7] are examples of co-allocating, scheduling and brokering services, and MPICH [32] for Grid enabled programming systems, and CAS (community authorization service) [21] for global resource policies. 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 (i.e. QuarkNet e-learning environment [52], National Virtual Observatory (http://www.us-vo.org), TeraGrid Science gateway (http://www.teragrid.org)). 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.

Clouds can be built on top of many existing protocols such as Web Services (WSDL, SOAP), and some advanced Web 2.0 technologies such as REST, RSS, AJAX, etc.

In fact, behind the cover, it is possible for Clouds to be implemented over existing Grid technologies leveraging more than a decade of community efforts in standardization, security, resource management, and virtualization support. There are also multiple versions of definition for Cloud architecture, 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.
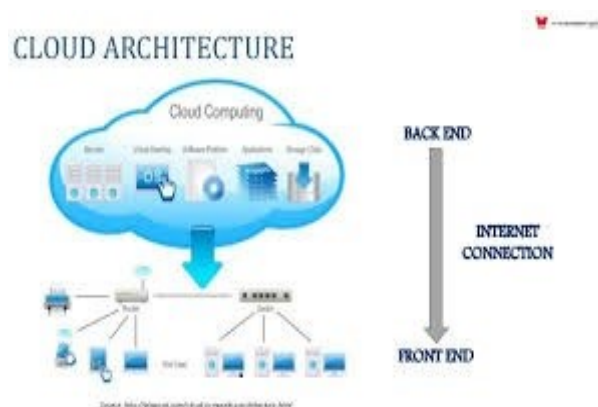
Figure 3: Cloud Architecture

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. For instance, a Web hosting environment, a scheduling service, etc.

Finally, the application layer contains the applications that would run in the Clouds. Clouds in general provide services at three different levels (IaaS(Cloud infrastructure services, known as Infrastructure as a Service (IaaS), are self-service models for accessing, monitoring, and managing remote datacenter infrastructures), PaaS(Cloud platform services, or Platform as a Service (PaaS), are used for applications, and other development, while providing cloud components to software.), and Saas(Cloud application services, or Software as a Service (SaaS), represent the largest cloud market and are still growing quickly.) [50]) as follows, although some providers can choose to expose services at more than one level.

Infrastructure as a Service (IaaS) [50] provisions hardware, software, and equipments (mostly at the unified resource layer, but can also include part of the fabric layer) 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 [3] and S3 (Simple Storage Service) [4] where compute and storage infrastructures are open to public access with a utility pricing model; Eucalyptus [15] 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) [50] 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 [28], which enables users to build Web applications on the same scalable systems that power Google applications. Software as a Service (SaaS) [50] delivers special-purpose software that is remotely accessible by consumers through the Internet with a usage-based pricing model.

Sales force 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. Although Clouds provide services at three different levels (IaaS, PaaS, and Saas), standards for interfaces to these different levels still remain to be defined.

This leads to interoperability problems between today's Clouds, and there is little business incentives for Cloud providers to invest additional resources in defining and implementing new interfaces.

As Clouds mature, and more sophisticated applications and services emerge that require the use of multiple Clouds, there will be growing incentives to adopt standard interfaces that facilitate interoperability in order to capture emerging and growing markets in a saturated Cloud market.

2.3 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), such as PBS, Condor, SGE manages the compute resources for a Grid site, and users submit batch jobs (via GRAM) (The Grid Resource Allocation and Management (GRAM) protocol supports remote submission of a computational request (for example, to run a program) to a remote computational resource, and it supports subsequent monitoring and control of the computation. GRAM is the Globus protocol that Condor-G uses to talk to remote Globus job managers.) 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.

For example, a job could say, stage in the input data from a URL to the local storage, run the application for 60 minutes on 100 processors, and stage out the results to some FTP server.

The job would wait in the LRM's(Location referencing management system) wait queue until the 100 processors were available for 60 minutes, at which point the 100 processors would be allocated and dedicated to the application for the duration of the job.

Due to the expensive scheduling decisions, data staging in and out, and potentially long queue times, many Grids don't natively support interactive applications; although there are efforts in the Grid community to enable lower latencies to resources via multi-level scheduling, to allow applications with many short-running tasks to execute efficiently on Grids [40]. Cloud Computing compute model will likely look very different, with resources in the Cloud being shared by all users at the same time (in contrast to dedicated resources governed by a queuing system).

This should allow latency sensitive applications to operate natively on Clouds, although ensuring a good enough level of QoS (Quality of Service) 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

We 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, we envision that the next-generation Internet Computing will take the triangle model shown in Figure 4: Internet Computing will be centralized around Data, Clouding Computing, as well as Client Computing. Cloud Computing and Client Computing will coexist and evolve hand in hand, while data management (mapping, partitioning, querying, movement, caching, replication, etc) 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.

Furthermore, many end-users will have various hardware driven end-functionalities, such as visualization and multimedia playback, which will typically run locally.
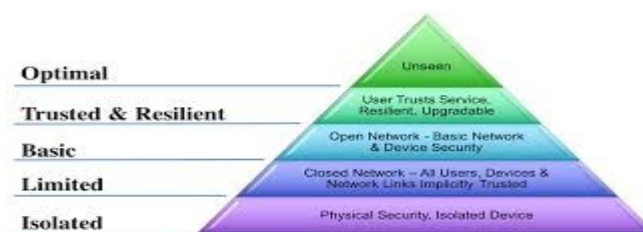

Figure 4 the triangle model of next-generation Internet Computing.

The importance of data has caught the attention of the Grid community for the past decade; Data Grids [10] have been specifically designed to tackle data intensive applications in Grid environments, with the concept of virtual data [22] playing a crucial role.

Virtual data captures the relationship between data, programs and computations and prescribes various abstractions that a data grid can provide: location transparency where data can be requested without regard to data location, a distributed metadata catalog is engaged to keep track of the locations of each piece of data (along with its replicas) across grid sites, and privacy and access control are enforced; materialization transparency: data can be either recomputed on the fly or transferred upon request, depending on the availability of the data and the cost to re-compute. There is also representation transparency where data can be consumed and produced no matter what their actual physical formats and storage are, data are mapped into some abstract structural representation and manipulated in that way.

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. [46]

There are large differences in IO speeds from local disk storage to wide area networks, which can drastically affect application performance.

To achieve good scalability at Internet scales for Clouds, Grids, and their applications, data must be distributed over many computers, and computations must be steered towards the best place to execute in order to minimize the communication costs [46]. Google's Map Reduce [13] 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 systems (e.g. NFS, GPFS(General Parallel File System, GPFS is a clustered file system and should provide excellent scalability, in cost and performance, as the department continues to grow.

The GPFS software allows us to re-export file systems via NFS for Linux clients and we clustered CIFS for Windows.), PVFS((Parallel Virtual File System), Luster(ustre is a type of parallel distributed file system, generally used for large-scale cluster computing. The name Lustre is a portmanteau word derived from Linux and cluster.)), where data locality cannot be easily applied. One approach is to improve schedulers to be data-aware, and to be able to leverage data locality information when scheduling computational tasks; this approach has shown to improve job turn-around time significantly [41].

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.

Attempting to address the storage and computational problems separately forces much data movement between computational and storage resources, which will not scale to tomorrow's peta-scale datasets and millions of processors, and will yield significant underutilization of the raw resources.

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 (the largest supercomputers today have over 200K processors and Grids surpassing 100K processors), there is an ever-growing emphasis for support of high throughput computing with high sustainable dispatch and execution rates.

We believe that data management architectures are important to ensure that the data management implementations scale to the required dataset sizes in the number of files,

objects, and dataset disk space usage while at the same time, ensuring that data element information can be retrieved fast and efficiently.

Grids have been making progress in combining compute and data management with data-aware schedulers [41], 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.

Although data-intensive applications may not be typical applications that Clouds deal with today, as the scales of Clouds grow, it may just be a matter of time for many Clouds.

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 (or even up to thousands or millions of) 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. Virtualization also enables each application to be encapsulated such that they can be configured, deployed, started, migrated, suspended, resumed, stopped, etc., and thus provides better security, manageability, and isolation.

There are also many other reasons that Clouds tend to adopt virtualization:

1) Server and application consolidation, as multiple applications can be run on the same server, resources can be utilized more efficiently;

2) configurability, as the resource requirements for various applications could differ significantly, some require large storage, some compute, in order to dynamically configure and bundle (aggregate) resources for various needs, virtualization is necessary as this is not achievable at the hardware level;

3) increased application availability, virtualization allows quick recovery from unplanned outages, as virtual environments can be backed up and migrated with no interruption in service;

4) Improved responsiveness: resource provisioning, monitoring and maintenance can be automated, and common resources can be cached and reused.

All these features of virtualization provide the basis for Clouds to meet stringent SLA (Service Level Agreement) requirements in a business setting, which cannot be easily achieved with a non-virtualized environment in a cost-effective manner as systems would have to be over provisioned to handle peak load and waste resources in idle periods.

After all, a virtualization infrastructure can be just thought as a mapping from IT resources to business needs. 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 (i.e. by not virtualizing them). However, there are efforts in Grids to use virtualization as well, such as Nimbus [56] (previously known as the Virtual Workspace Service [26]), which provide the same abstraction and dynamic deployment capabilities.

A virtual workspace is an execution environment that can be deployed dynamically and securely in the Grid. Nimbus (Nimbus is a toolkit that, once installed on a cluster, provides an infrastructure as a service cloud to its client) provides two levels of guarantees:

1) Quality of life: users get exactly the (software) environment they need

2) Quality of service: provision and guarantee all the resources the workspace needs to function correctly (CPU, memory, disk, bandwidth, availability), allowing for dynamic renegotiation to reflect changing requirements and conditions.

In addition, Nimbus can also provision a virtual cluster for Grid applications (e.g. a batch scheduler, or a workflow system), which is also dynamically configurable, a growing trend in Grid Computing. It is also worth noting that virtualization − in the past − had significant performance losses for some applications, which has been one of the primary disadvantage of using virtualization. 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 TeraGri (TeraGrid integrated high-performance computers, data resources and tools, and experimental facilities. Resources included more than a petaflop of computing capability and more than 30 petabytes of online and archival data storage, with rapid access and retrieval over high-performance computer network connections.)) 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; for example, the Ganglia [25] (Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. Supports clusters up to 2000 nodes in size.) distributed (and hierarchical) monitoring system can monitor a federation of clusters and Grids and has seen wide adoption in the Grid community.

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.

Essentially monitoring in Clouds requires a fine balance of business application monitoring, enterprise server management, virtual machine monitoring, and hardware

maintenance, and will be a significant challenge for Cloud Computing as it sees wider adoption and deployments.

On the other hand, monitoring can be argued to be less important in Clouds, as users are interacting with a more abstract layer that is potentially more sophisticated; this abstract layer could respond to failures and quality of service (QoS) requirements automatically in a general-purpose way irrespective of application logic.

In the near future, user-end monitoring might be a significant challenge for Clouds, but it will become less important as Clouds become more sophisticated and more self-maintained and self-healing.

Provenance

Provenance refers to the derivation history of a data product, including all the data sources, intermediate data products, and the procedures that were applied to produce the data product.

Provenance information is vital in understanding, discovering, validating, and sharing a certain data product as well as the applications and programs used to derive it.

In some disciplines such as finance and medicine, it is also mandatory to provide what is called an "audit trail" for audition purpose.

In Grids, provenance management has been in general built into a workflow system, from early pioneers such as Chimera [22] (Virtual Data System for Representing, Querying, and Automating Data Derivation), to modern scientific workflow systems, such as Swift [53], Kepler[35], (Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience) and VIEW [34] to support the discovery and reproducibility of scientific results.

It has also been built as a standalone service, such as PreServ [29], to facilitate the integration of provenance component in more general computing models, and deal with trust issues in provenance assertion.

Using provenance information, scientists can debug workflow execution, validate or invalidate scientific results, and guide future workflow design and data exploration. While provenance has first shown its promise in scientific workflow systems [22] and database systems [47], a long-term vision is that provenance will be useful in other systems as well, necessitating the development of a standard, open, and universal representation and query model.

Currently, the provenance challenge series [39] and the open provenance model initiative [38] provide the active forums for these standardization effort and interaction.

On the other hand, Clouds are becoming the future playground for e-science research, and provenance management is extremely important in order to track the processes and support the reproducibility of scientific results [45].

Provenance is still an unexplored area in Cloud environments, in which we need to deal with even more challenging issues such as tracking data production across different service providers (with different platform visibility and access policies) and across different software and hardware abstraction layers within one provider.

In other words, capturing and managing provenance in Cloud environments may prove to be more difficult than in Grids, since in the latter there are already a few provenance systems and initiatives, however scalable provenance querying [55] and secure access of provenance information are still open problems for both Grids and Clouds environments.

2.4 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; large variations in resource heterogeneity, stability and performance; exception handling in highly dynamic (in that resources can join and leave pretty much at any time) 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) [36] 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.

MPICH-G2(Software) [32] is a Grid enabled implementation of MPI.

It gives the familiar interface of MPI while providing integration with the Globus Toolkit.

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. For instance, Linda [1] defines a set of coordination primitives to put and retrieve tuples from a shared data space called the tuple space. It has been shown to be straightforward to use such primitives to implement a master-worker parallel scheduler.

The Ninf-G GridRPC [37](a GridRPC system on the Globus toolkit", Grid Computing: Making the Global Infrastructure a Reality) system integrates a Grid RPC (Remote Procedure Call) layer on top of the Globus toolkit. It publishes interfaces and function libraries in MDS, and utilizes GRAM (: Globus Resource Allocation Manager) to invoke remote executables.

In Grids, however, many applications are loosely coupled in that the output of one may be passed as input to one or more others—for example, as a file, or via a Web Services invocation. While such "loosely coupled" computations can involve large amounts of computation and communication, the concerns of the programmer tend to be different from those in traditional high performance computing, being focused on management issues relating to the large numbers of datasets and tasks rather than the optimization of inter processor communication.

In such cases, workflow systems [54] suit better in the specification and execution of such applications.

More specifically, a workflow system allows the composition of individual (single step) components into a complex dependency graph, and it governs the flow of data and/or control through these components.

An example is the Swift system [53], which bridges scientific workflows with parallel computing.

It is a parallel programming tool for rapid and reliable specification, execution, and management of large-scale science and engineering workflows.

The Swift runtime system relies on the CoG Karajan [33] workflow engine for efficient scheduling and load balancing, and it integrates the Falkon light-weight task execution service for optimized task throughput and resource efficiency [40]. WSRF (Web Services Resource Framework) has emerged from OGSA (Open Grid Service Architecture) [11] as more and more Grid applications are developed as services.

WSRF allows Web Services to become stateful, and it provides a set of operations to set and retrieve the states (resources). The Globus Toolkit version 4 contains Java and C implementations of WSRF, most of the Globus core services have been re-engineered to build around WSRF, these altogether will enable service oriented Grid programming model. MapReduce [13] (Simplified Data Processing on Large Clusters", Sixth Symposium on Operating System Design and Implementation)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; 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 [30] is the open source implementation of the MapReduce model, and Pig is a declarative programming language offered on top of Hadoop. Microsoft has developed the Cosmos distributed storage system and Dryad processing framework, and offers DryadLINQ [31] and Scope as declarative programming model on top of the storage and computing infrastructure. DryadLINQ uses the object oriented LINQ query syntax where Scope provides basic operators similar to those of SQL such as Select, Join, and Aggregation etc, both translate the abstract specification into detailed execution plan. Mesh-up's and scripting (Java Script, PHP, Python etc) have been taking the place of a workflow system in the Cloud world, since there is no easy way to integrate services and applications from various providers.

They are essentially data integration approaches, because they take outputs from one service/application, transform them and feed into another.

Google App Engine uses a modified Python runtime and chooses Python scripting language for Web application development, the interface to its underlying BigTable storage system is some proprietary query language (named, as you would think, GQL) that is reminiscent of SQL, although all these will probably change.

Clouds (such as Amazon Web Services, Microsoft's Azure Services Platform) have generally adopted Web Services APIs where users access, configure and program Cloud services using pre-defined APIs exposed as Web services, and HTTP and SOAP are the common protocols chosen for such services.

Although Clouds adopted some common communication protocols such as HTTP and SOAP, the integration and interoperability of all the services and applications remain the biggest challenge, as users need to tap into a federation of Clouds instead of a single Cloud provider.

2.5 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(delay) 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.

Related to HTC applications loosely coupled nature, there are other application classes, such Multiple Program Multiple Data (MPMD), MTC, capacity computing, utility computing, and embarrassingly parallel, each with their own niches [42].

These loosely coupled applications can be composed of many tasks (both independent and dependent tasks) that can be individually scheduled on many different computing resources across multiple administrative boundaries to achieve some larger application goal.

Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive.

The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely or tightly coupled.

The aggregate number of tasks, quantity of computing, and volumes of data could be small but also extremely large. 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 [51], 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.

Gateways provide access to a variety of capabilities including workflows, visualization, resource discovery and job execution services through a browser-based user interface (which can arguably hide much of the complexities).
Scientific gateways are beginning to adopt a wide variety of Web 2.0 technologies, but to date, much of the developments in Grids and Web 2.0 have been made in parallel with little interaction between them.

These new technologies are important enhancements to the ways gateways interact with services and provide rich user interactivity. Although scientific gateways have only emerged in Grids recently, Clouds seem to have adopted the use of gateways to Cloud resources almost exclusively for end-user interaction.

The browser and Web 2.0 technologies will undoubtedly play a central role on how users will interact with Grids and Clouds in the future.

2.6 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.

The key issues considered are: single sign-on, so that users can log on only once and have access to multiple Grid sites, this will also facilitate accounting and auditing; delegation, so that a program can be authorized to access resources on a user's behalf and it can further delegate to other programs; privacy, integrity and segregation, resources belonging to one user cannot be accessed by unauthorized users, and cannot be tampered during transfer; coordinated resource allocation, reservation, and sharing, taking into consideration of both global and local resource usage policies.

The public-key based GSI (Grid Security Infrastructure) protocols are used for authentication, communication protection, and authorization.

Furthermore, CAS (Community Authorization Service) is designed for advanced resource authorization within and across communities.

Gruber (A Grid Resource Usage SLA Broker) [14] is an example that has distributed policy enforcement points to enforce both local usage policies and global SLAs (Service Level Agreement), which allows resources at individual sites to be efficiently shared in multi-site, multi-VO environments.

Currently, the security model for Clouds seems to be relatively simpler and less secure than the security model adopted by Grids. Cloud infrastructure typically rely on Web forms (over SSL (Secure Sockets Layer)) to create and manage account information for end-users, and allows users to reset their passwords and receive new passwords via Emails in an unsafe and unencrypted communication.

Note that new users could use Clouds relatively easily and almost instantly, with a credit card and/or email address.

To contrast this, Grids are stricter about its security. For example, although web forms are used to manage user accounts, sensitive information about new accounts and passwords requires also a person to person conversation to verify the person, perhaps verification from a sponsoring person who already has an account, and passwords will only be faxed or mailed, but under no circumstance will they be emailed.

The Grid approach to security might be more time consuming, but it adds an extra level of security to help prevent unauthorized access.

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 [6]:

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;

7) Long-term viability: your data should be viable even the Cloud provider is acquired by another company.

3. Recommendations

In this context, Egyptian universities and specialized educational institutions should pay attention to the preparation of cadres with a high degree of specialization and to mobilize their potential to deal with such modern technologies and the need to subject them to professional training and continuous rehabilitation in coordination with the Ministry of Communications and Information Technology in order to learn the techniques Modern era in order to be able to perform their effectiveness towards providing their technical expertise to their country to become a real added value to the national economy and technological progress.

4. Conclusions and lights to the future

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.

We will hazard a few predictions, based on our beliefs that the economics of computing will look more and more like those of energy.

We believe that we will nevertheless see parallel evolution, driven by similar forces. In building this distributed "Cloud" or "Grid", we will need to support on-demand provisioning and configuration of integrated "virtual systems" providing the precise capabilities needed by an end-user.

We will need to define protocols that allow users and service providers to discover and hand off demands to other providers, to monitor and manage their reservations, and arrange payment.

We will need tools for managing both the underlying resources and the resulting distributed computations.

We will need the centralized scale of today's Cloud utilities, and the distribution and interoperability of today's Grid facilities. Unfortunately, at least to date, the methods used to achieve these goals in today's commercial clouds have not been open and general purpose, but instead been mostly proprietary and specialized for the specific internal uses (e.g., large-scale data analysis) of the companies that developed them.

Some of the required protocols and tools will come from the smart people from the industry at Amazon, Google, Yahoo, Microsoft, and IBM. Others will come from the smart people from academia and government labs. Others will come from those creating whatever we call this stuff after Grid and Cloud.

References

[1]. S. Ahuja, N. Carriero, and D. Gelernter. "Linda and Friends", IEEE Computer 19 (8), 1986, pp. 26-34.

[2]. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. "Data Management and Transfer in High Performance Computational Grid Environments", Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771.

[3]. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2, 2008.

[4]. Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3, 2008.

[5]. B. Bode, D.M. Halstead, R. Kendall, Z. Lei, W. Hall, D. Jackson. "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters", Usenix, 4th Annual Linux Showcase & Conference, 2000.

[6]. J. Brodkin. "Gartner: Seven cloud-computing security risks", http://www.networkworld.com/news/2008/070208-cloud.html, 2008.

[7]. R. Buyya, D. Abramson, J. Giddy. "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", IEEE Int. Conf. on High Performance Computing in Asia-Pacific Region (HPC ASIA) 2000.

[8]. R. Buyya, K. Bubendorfer. "Market Oriented Grid and Utility Computing", Wiley Press, New York, USA, 2008.

[9]. A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi, "Storing and Querying Scientific Workflow Provenance Metadata Using an RDBMS", in Proc. of the 2nd International Workshop on Scientific Workflows and Business Workflow Standards in eScience, in conjunction with e-Science, pp.611-618, Bangalore, India, December, 2007.

[10]. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", Jrnl. of Network and Computer Applications, 23:187-200, 2001.

[11]. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke. "From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution," March 5, 2004.

[12]. S. Davidson, S. Boulakia, A. Eyal, B. Ludäscher, T. McPhillips, S. Bowers, M. Anand, J. Freire, "Provenance in Scientific Workflow Systems", IEEE Data Eng. Bull. 30(4): 44-50 (2007).

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

[14]. C. Dumitrescu, I. Raicu, I. Foster. "The Design, Usage, and Performance of GRUBER: A Grid uSLA-based Brokering Infrastructure", Intl. Journal of Grid Computing, 2007.

[15]. Eucalyptus, http://eucalyptus.cs.ucsb.edu/, 2008.

[16]. I. Foster, C. Kesselman. "Globus: A Metacomputing Infrastructure Toolkit", Intl J. Supercomputer Applications, 11(2):115-128, 1997.

[17]. I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy. "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", Intl Workshop on Quality of Service, 1999.

[18]. 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.

[19]. I. Foster. What is the Grid? A Three Point Checklist, July 2002.

[20]. 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.

[21]. I. Foster, C. Kesselman, L. Pearlman, S. Tuecke, and V. Welch. "The Community Authorization Service: Status and Future," In Proc. of Computing in High Energy Physics (CHEP), 2003.

[22]. I. Foster, J. Vöckler, M. Wilde, Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation", SSDBM 2002: 37-46.

[23]. 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.

[24]. J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. "Condor-G: A Computation Management Agent for MultiInstitutional Grids," Cluster Computing, 5 (3). 237-246. 2002.

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

[26]. K. Keahey, I. Foster, T. Freeman, and X. Zhang. "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid", The Scientific Programming Journal, 2006.

[27]. The Globus Security Team. "Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective," Technical Report, Argonne National Laboratory, MCS, 2005.

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

[29]. P. Groth, S. Miles, W. Fang, S. Wong, K. Zauner, L. Moreau. "Recording and using provenance in a protein compressibility experiment", in Proceedings of the 14th IEEE Int. Symposium on High Performance Distributed Computing (HPDC), 2005.

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

[31]. M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", Euro. Conf. on Computer Systems (EuroSys), 2007.

[32]. N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A GridEnabled Implementation of the Message Passing Interface. Journal of Parallel and Distributed Computing, 2003.

[33]. G.v. Laszewski, M. Hategan, D. Kodeboyina. "Java CoG Kit Workflow", Taylor, I.J., Workflows for eScience, 2007.

[34]. C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi, "Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System", IEEE International Conference on Services Computing (SCC), pp.335-342, 2008.

[35]. B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, Y. Zhao, "Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience", 18(10).

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

[37]. H. Nakada, Y. Tanaka, S. Matsuoka, S. Sekiguchi. "Ninf-G: a GridRPC system on the Globus toolkit", Grid Computing: Making the Global Infrastructure a Reality, pp. 625--638, 2003.

[38]. Open provenance model initiative, http://twiki.ipaw.info/bin/view/Challenge/OpenProvenanceModelReview, 2008.

[39]. The provenance challenge series: http://twiki.ipaw.info/bin/view/Challenge/, 2008.

[40]. I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: a Fast and Light-weight tasK executiON framework", IEEE/ACM SuperComputing 2007.

[41]. I. Raicu, Y. Zhao, I. Foster, A. Szalay. "Accelerating Largescale Data Exploration through Data Diffusion", International Workshop on Data-Aware Distributed Computing 2008.

[42]. I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. "Towards Loosely-Coupled Programming on Petascale Systems", IEEE/ACM Supercomputing 2008.

[43]. 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.

[44]. J. Silvestre. "Economies and Diseconomies of Scale," The New Palgrave: A Dictionary of Economics, v. 2, pp. 80–84, 1987.

[45]. Y. Simmhan, Beth Plale, Dennis Gannon. "A survey of data provenance in e-science", SIGMOD Record 34(3): 31-36, 2005.

[46]. A. Szalay, A. Bunn, J. Gray, I. Foster, I. Raicu. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006.

[47]. W. C. Tan. "Provenance in databases: Past, current, and future", IEEE Data Eng. Bull., 30(4):3–12, 2007.

[48]. D. Thain, T. Tannenbaum, M. Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

[49]. "Twenty Experts Define Cloud Computing", SYS-CON Media Inc, http://cloudcomputing.sys-con.com/read/612375_p.htm, 2008.

[50]. "What is Cloud Computing?", Whatis.com. http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci1287881,00.html, 2008.

[51]. N. Wilkins-Diehr: Special Issue: Science Gateways - Common Community Interfaces to Grid Resources. Concurrency and Computation: Practice and Experience 19(6): 743-749 (2007) .

[52]. Y. Zhao, M. Wilde, I. Foster, J. Voeckler, J. Dobson, E. Gilbert, T. Jordan, E. Quigg. "Virtual Data Grid Middleware Services for Data-Intensive Science", Concurrency and Computation: Practice and Experience, 2005.

[53]. Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von vonLaszewski, I. Raicu, T. Stef-Praun, and M. Wilde. "Swift: Fast, reliable, loosely coupled parallel computation", IEEE Int. Workshop on Scientific Workflows, pages 199–206, 2007.

[54]. Y. Zhao, I. Raicu, I. Foster. "Scientific Workflow Systems for 21st Century, New Bottle or New Wine?" IEEE Workshop on Scientific Workflows, 2008.

[55]. Y. Zhao and S. Lu, "A Logic Programming Approach to Scientific Workflow Provenance Querying", IPAW, LNCS Volume 5272, pp.31-44, 2008.

[56]. K. Keahey, T. Freeman. "Contextualization: Providing OneClick Virtual Clusters", to appear, eScience 2008, 2008.

[57]. http://research.cs.wisc.edu/htcondor/manual/v7.6/5_3Grid_Universe.html