International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
August 2015, PP 25-33
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Cloud, Cluster & Grid Computing

Prof. Pradeep Kumar Shriwas

H.O.D.– Computer Science
Shri Agrasen Girls College, Korba, Chhattisgarh(India)
shriwasji.ps@gmail.com, Mailme_pradeep1012@rediffmail.com

Abstract: *Cloud computing is really changing the way of computation. Many computer resources such as hardware and software are collected into the resource pool which can be assessed by the users via the internet through web browsers or light weight desktops or mobile devices. It is not a very new concept; it is related to grid computing paradigm, and utility computing as well as cluster computing. All these computing viz. Grid, cluster and utility computing, have actually contributed in the development of cloud computing. In this paper, we are going to compare all the technologies which leads to the emergence of Cloud computing.*

Keywords: *cluster computing; grid computing; cloud computing; resource balancing;*

1. INTRODUCTION

Cloud computing is the new computing paradigm which provides large pool of dynamical scalable and virtual resources as a service on demand. The main principle behind cloud computing model is to offer computing, storage, and software as a service or as a utility. We just need internet to use these utilities. Buyya et al. (2009) have defined it as follows: "Cloud is a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more agreements (SLA) established through negotiation unified computing resources based on service-level between the service provider and consumers."

The terms "grid computing" and "cluster computing" have been used almost interchangeably to describe networked computers that run distributed applications and share resources. They have been used to describe such a diverse set of distributed computing solutions that their meanings have become ambiguous. Both technologies improve application performance by executing parallelizable computations simultaneously on different machines, and both technologies enable the shared use of distributed resources.

However, cluster and grid computing represent different approaches to solving performance problems; although their technologies and infrastructure differ, their features and benefits complement each other. A cluster and a grid can run on the same network at the same time, and a cluster can even contribute resources to a grid.

Both of these forms of distributed computing have their roots in the UNIX operating system. However, as operating systems and networks have evolved, more operating systems have been adapted for use in both clusters and grids. Recent software releases have greatly improved both cluster and grid computing on the Microsoft® Windows® operating systems.

This document defines grid computing and cluster computing from the perspective of the Windows operating systems and development environments. After reading this document you will have a clear understanding of the Windows cluster computing and grid computing options, and you will understand how the two technologies complement each other.

2. WHAT'S THE DIFFERENCE?

The computers (or "nodes") on a cluster are networked in a tightly-coupled fashion--they are all on the same subnet of the same domain, often networked with very high bandwidth connections. The nodes are homogeneous; they all use the same hardware, run the same software, and are generally configured identically. Each node in a cluster is a dedicated resource--generally only the cluster applications run on a cluster node. One advantage available to clusters is the Message Passing Interface (MPI) which is a programming interface that allows the distributed application instances to communicate with each other and share information. Dedicated hardware, high-speed interconnects,

and MPI provide clusters with the ability to work efficiently on "fine-grained" parallel problems, including problems with short tasks, some of which may depend on the results of previous tasks.

In contrast, the nodes on a grid can be loosely-coupled; they may exist across domains or subnets. The nodes can be heterogeneous; they can include diverse hardware and software configurations. A grid is a dynamic system that can accommodate nodes coming in and dropping out over time. This ability to grow and shrink at need contributes to a grid's ability to scale applications easily. Grids typically do not require high-performance interconnects; rather, they usually are configured to work with existing network connections. As a result, grids are better suited to relatively "coarse-grained" parallel problems, including problems composed primarily of independent tasks. There is no dominant programming paradigm in grid computing today, and a key challenge to increasing the acceptance of grid computing is creating grid-enabled applications with familiar programming models. Digipede's object-oriented programming for grid (OOP-G) is one such model.

Grids can incorporate clusters. Often the best way to make use of all available resources is to manage the cluster resources as part of a larger grid, assigning jobs and tasks to the resources best suited to those jobs and tasks. For example, jobs requiring MPI would be assigned exclusively to the cluster, while loosely-coupled jobs could be assigned to all grid nodes, including those in the cluster (when available). Indeed, cluster compute nodes make excellent grid nodes, and many grids are composed exclusively of dedicated servers.

	Windows Compute Cluster Server 2003	**Digipede Network**
Operating system	Homogeneous - Windows Server 2003 64-bit	Heterogeneous - Microsoft Windows Compute Cluster Server 2003, Windows 2000 SP4, Server 2000 SP4, XP SP2, Server 2003
Supported programming languages	C, C++, Fortran77, Fortran90, for MS MPI; any language for non-MPI	Any language that supports .NET 1.1, .NET 2.0, or COM
Development tools	Visual Studio 2005 Professional and Visual Studio 2005 Team	Visual Studio .NET, Visual Studio 2005, any IDE that supports .NET or COM interfaces
MPI	Yes	No
.NET API for distributed applications	No	Yes (1.1 and 2.0)
Distributed applications	Executables and scripts	.NET objects, COM Servers, executables, and scripts
CPU supported	64-bit only	32-bit and 64-bit
Task execution	Tightly coupled or loosely Coupled	Loosely coupled
File distribution model	Staged by user, usually via Scripts	Automatically staged by the Digipede Network
Compute nodes	Dedicated	Dedicated or shared
Scheduling model	Job Scheduler on head node (push model)	Agent driven (pull model)

Both systems use similar terminology to define submitted requests: A job defines the work submitted to the system which includes the required resources and the tasks to execute. A task is an individual unit of work that can be executed concurrently with other tasks.

3. CLUSTER COMPUTING ON WINDOWS

Cluster computing on Windows is provided by Windows Compute Cluster Server 2003 (CCS) from Microsoft. CCS is a 64-bit version of Windows Server 2003 operating system packaged with various software components that greatly eases the management of traditional cluster computing. With a dramatically simplified cluster deployment and management experience, CCS removes many of the obstacles imposed by other solutions. CCS enables users to integrate with existing Windows infrastructure, including Active Directory and SQL Server.

CCS supports a cluster of servers that includes a single head node and one or more compute nodes. The head node controls and mediates all access to the cluster resources and is the single point of management, deployment, and job scheduling for the compute cluster. All nodes running in the cluster must have a 64-bit CPU.

How Does It Work?

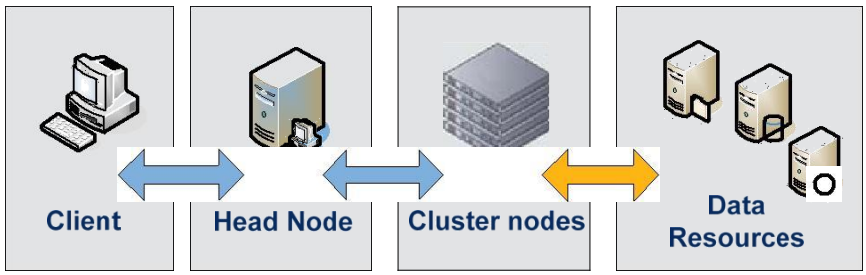

Figure. *Compute Cluster Topology*

As shown in **Figure**, a user submits a job to the head node. The job identifies the application to run on the cluster. The job scheduler on the head node assigns each task defined by the job to a node and then starts each application instance on the assigned node. Results from each of the application instances are returned to the client via files or databases.

Application parallelization is provided by Microsoft MPI (MSMPI), which supports communication between tasks running in concurrent processes. MSMPI is a "tuned" MPI implementation, optimized to deliver high performance on the 64-bit Windows Server OS. MSMPI calls can be placed within an application, and the mpiexec.exe utility is available to control applications from the command-line. Because MSMPI enables communication between the concurrently executing application instances, the nodes are often connected by a high-speed serial bus such as Gigabit Ethernet or InfiniBand.

Development Considerations

To understand the CCS development options, it is important to understand how CCS defines serial tasks and parallel tasks. All tasks are defined in executables or scripts. All executables must be command-line applications that do not require user interaction. Serial tasks are tasks that have no need to communicate with any other tasks. Parallel tasks communicate with the other running application instances.

Serial Tasks

Serial tasks are executables or scripts that do not communicate with concurrently running application instances; consequently they can be developed using any Microsoft language and tool.

Parallel Tasks

Parallel tasks are executables or scripts that do communicate with concurrently running application instances and require MSMPI. A parallel application cannot be written for CCS without MSMPI; language bindings for MSMPI are available for C++, C, Fortran90, and Fortran77. MSMPI development is supported in Visual Studio 2005 Professional and Visual Studio 2005 Team. At this time .NET support for a subset MSMPI functions is provided via a P/Invoke call (which allows a .NET application to make a COM interface call).

4. GRID COMPUTING ON WINDOWS

The Digipede Network is a grid computing solution that provides the advantages of traditional grid solutions with additional features to simplify job creation and allow developers to grid-enable applications. The Digipede Network is a grid infrastructure which comprises a server to manage the system and many agent-nodes to execute the distributed work. The Digipede Server receives all job requests and maintains a prioritized queue of work to be done, along with a history of work completed on the system. It also guarantees execution of work on the system by monitoring the status of all Digipede Agents working on the grid. The Digipede Agent software is installed on each of the grid compute nodes; it manages that compute node's work on the grid. Although the Digipede Server receives all the job requests, the Digipede Agent decides what work can be performed on the compute node. It moves files and data as appropriate, controls the execution of the task, and returns results and status information. The Digipede Network also includes comprehensive job creation tools, detailed below.

The Digipede Server runs natively on all editions of Windows Server 2003; the Digipede Agent runs on any 32-bit or 64-bit Windows operating system since Windows 2000. As a result, the Digipede Network can be installed quickly and easily onto new or existing networked Windows computers.

How Does It Work?

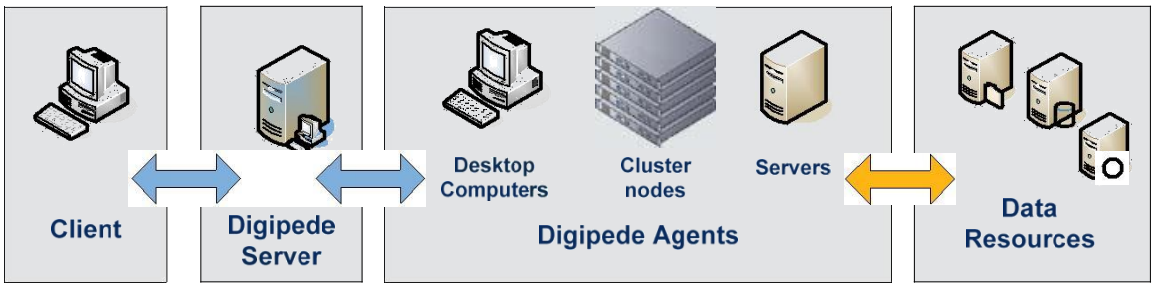

Figure: *Digipede Network Topology*

As shown in **Figure**, a user submits a job to the Digipede Server. The Digipede Agents check in with the Digipede Server and, if the compute node has the resources required by a job, the Digipede Agent takes a task. The executable, script, .NET object, or COM server assigned to the task is executed and the results are returned to the Digipede Server, which then returns the results to the client.

The Digipede Workbench is GUI application that allows users to distribute work without writing scripts. Using wizards and design forms, the user indicates the files that need to be distributed, designates input and output files, and sets command line parameters.

Beyond distributing command-line executables, the Digipede Framework SDK is a powerful tool that allows programmers to add the power of grid computing directly into applications. The Digipede Framework facilitates the distribution of .NET objects (or COM servers) instead of executables, allowing a programmatic rather than command-line interaction with the distributed application. With the Digipede Framework SDK programmers are able to take advantage of programming methodologies and tools that they are already trained in. The Digipede Framework SDK is described in more detail in the whitepaper entitled *The Digipede Framework™ Software Development Kit (SDK).*

Development Considerations

With the Digipede Network the development considerations are mainly architectural. Command-line applications and scripts can be written without the Digipede Framework SDK and easily distributed using the Digipede Workbench or command-line interface.

Applications writing using the Digipede Framework initiate the distribution of work themselves and can be any type of application: GUI, command-line, or script.

Executables and Scripts

Distributable applications can be written using any Windows development environment and language. These applications must be command-line applications or scripts that do not require user interaction. The user can then use the Digipede Workbench to define jobs to quickly and easily distribute

execution. The Digipede Network will distribute the appropriate applications and files and manage remote execution.

Grid-enabled Applications

Grid-enabled applications require the Digipede Framework SDK and can be written using any development environment and language that supports COM, .NET 1.1, or .NET 2.0, including all versions of Visual Studio. Because the Digipede Framework supports existing API methodologies, Windows developers can quickly and easily grid-enable applications and scripts.

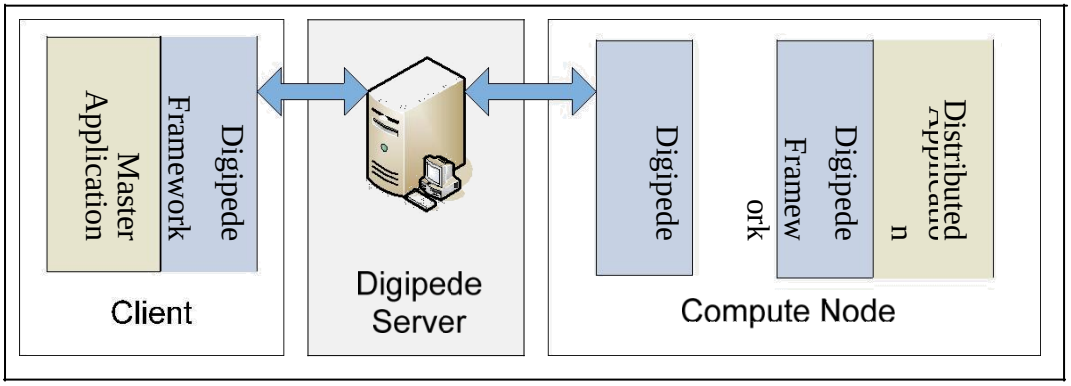

Figure: *Digipede Framework Topology*

In the client application, the programmer designates the classes that will be distributed. For each task in the job, the programmer instantiates an object from that class, initializes it, and adds it to the job. When the job is ready the programmer submits the job to the Digipede Network. Using serialization, each of the objects (and any assemblies necessary to instantiate it) are moved to the assigned compute node and deserialized by the Digipede Agent. The Digipede Agent starts the task and, when the work is completed, returns the object to the Digipede Server which then returns the object to the application.

There are many advantages to grid-enabling applications in this fashion:

- Parallelization is accomplished simply by instantiating objects from a class—the developer does not need to master complex threading schemes. Instead, by creating jobs consisting of multiple instances of objects, the parallel execution is handled automatically.
- Data is moved natively in the development language. In a typical grid system, all data moved from a client application to a distributed node must either be written to file or passed on the command line—a very limiting design restriction. By passing data with an API, the developer is able to pass data natively in their language of choice.
- Tasks are executed in parallel but the client application is notified via events so the results are processed serially—this allows for the power of parallel processing but avoids the complication of complex threading schemes in the client application.

Overview of the Grid Scheduling Problem

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities . It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications, and data, all connected through the Internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to access what, and under what conditions [48]. The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.

From the point of view of scheduling systems, a higher level abstraction for the Grid can be applied

by ignoring some infrastructure components such as authentication, authorization, resource discovery and access control. Thus, in this paper, the following definition for the term *Grid* adopted: "*A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements*" .

To facilitate the discussion, the following frequently used terms are defined:

- A ***task*** is an atomic unit to be scheduled by the scheduler and assigned to a resource.
- The ***properties*** of a task are parameters like CPU/memory requirement, deadline, priority, etc.
- A ***job*** (or ***metatask***, or ***application***) is a set of atomic tasks that will be carried out on a set of resources. Jobs can have a recursive structure, meaning that jobs are composed of sub-jobs and/or tasks, and sub-jobs can themselves be decomposed further into atomic tasks. In this paper, the term *job*, *application* and *metatask* are interchangeable.
- A ***resource*** is something that is required to carry out an operation, for example: a processor for data processing, a data storage device, or a network link for data transporting.
- A ***site*** (or ***node***) is an autonomous entity composed of one or multiple resources.
- A ***task scheduling*** is the mapping of tasks to a selected group of resources which may be distributed in multiple administrative domains.

The Grid Scheduling Process and Components

A Grid is a system of high diversity, which is rendered by various applications, middleware components, and resources. But from the point of view of functionality, we can still find a logical architecture of the task scheduling subsystem in Grid. For example, Zhu [123] proposes a common Grid scheduling architecture. We can also generalize a scheduling process in the Grid into three stages: resource discovering and filtering, resource selecting and scheduling according to certain objectives, and job submission [94]. As a study of scheduling algorithms is our primary concern here, we focus on the second step. Based on these observations, Fig. 1 depicts a model of Grid scheduling systems in which functional components are connected by two types of data flow: resource or application information flow and task or task scheduling command flow.'

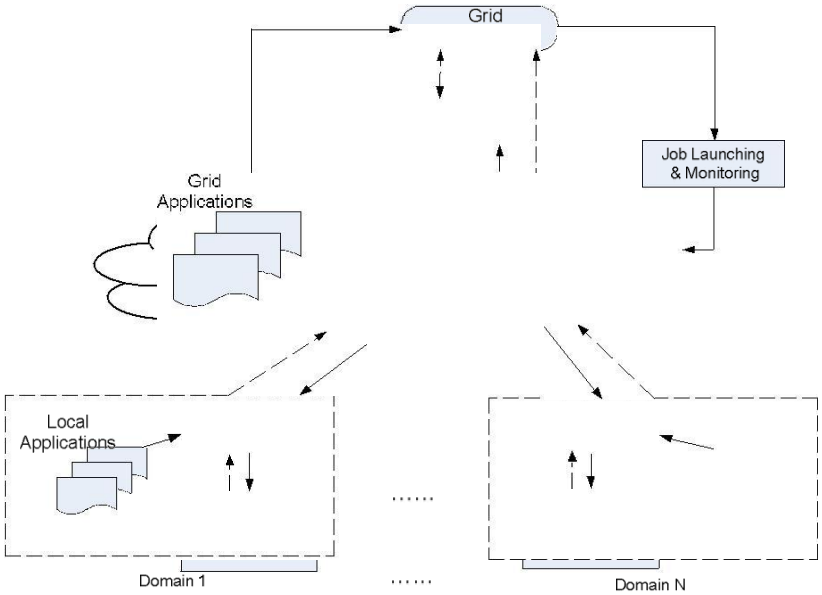

Fig. *A logical Grid scheduling architecture: broken lines show resource or application information flows and real lines show task or task scheduling command flows.*

A cluster is a type of parallel or distributed computer system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource [1][5]. The typical architecture of a cluster is shown in Figure 1. The key components of a cluster include multiple standalone computers (PCs, Workstations, or SMPs), operating systems, high-performance interconnects, middleware, parallel programming environments, and applications.

Cluster Computing	Grid Computing	Cloud Computing
Characteristics of Cluster computing 1:Tightly coupled systems 2: Single system image 3: Centralized Job management & scheduling system	Characteristics of Grid Computing 1: Loosely coupled (Decentralization) 2: Diversity and Dynamism 3: Distributed Job Management & scheduling	Characteristic of cloud computing 1: Dynamic computing infrastructure 2: IT service-centric approach 3: Self-service based usage model 4: Minimally or self-managed platform 5: Consumption-based billing
In cluster computing, a bunch of similar (or identical) computers are hooked up locally (in the same physical location, directly connected with very high speed connections) to operate as a single computer	In grid computing, the computers do not have to be in the same physical location and can be operated independently. As far as other computers are concerned each computer on the grid is a distinct computer.	In cloud computing, the computers need not to be in the same physical location.
The cluster computers all have the same hardware and OS.	The computers that are part of a grid can run different operating systems and have different hardware	The memory, storage device and network communication are managed by the operating system of the basic physical cloud units. Open source software such as LINUX can support the basic physical unit management and virtualization computing.
The computers in the cluster are normally contained in a single location or complex.	Grid are inherently distributed by its nature over a LAN, metropolitan or WAN	Clouds are mainly distributed over MAN
More than 2 computers are connected to solve a problem	A large project is divided among multiple computers to make use of their resources.	It does just the opposite. It allows multiple smaller applications to run at the same time.
Areas of cluster computing 1. Educational resources 2.Commercial sectors for industrial promotion 3.Medical research	Areas of Grid Computing 1.Predictive Modeling and Simulations 2.Engineering Design and Automation 3.Energy Resources Exploration 4.Medical, Military and Basic Research 5.Visualization	Areas of cloud Computing 1.Banking 2.Insurance 3.Weather Forecasting 4.Space Exploration 5.Software as a service 6.PaaS 7.Infrastructure- as -a-Service
Commodity computers	High-end computers (servers, clusters)	Commodity computers and high-end servers and network attached storage
Dedicated, high-end with low latency and high bandwidth Interconnection Network	Mostly Internet with high latency and low Bandwidth Interconnection Network	Dedicated, high-end with low latency and high Bandwidth Interconnection Network
Traditional login/password-based. Medium level of privacy depends on user privileges.	Public/private key pair based authentication and mapping a user to an account. Limited support for privacy.	Each user/application is provided with a virtual machine. High security/privacy is guaranteed. Support for setting per-file access control list (ACL).
Membership services discovery	Centralized indexing and decentralized info services discovery	Membership services discovery
Limited service negotiation	Yes, SLA based service negotiation	SLA based service negotiation
User management is centralized	User management is decentralized and also virtual organization (VO)-based	User management is centralized or can be delegated to third party

Interconnection Technologies and Communication Software

Clusters need to incorporate fast interconnection technologies in order to support high-bandwidth and low-latency inter-processor communication between cluster nodes. Slow interconnection technologies had always been a critical performance bottleneck for cluster computing. Today, improved network technologies help realize the construction of more efficient clusters.

Selecting a cluster interconnection network technology depends on several factors, such as compatibility with the cluster hardware and operating system, price, and performance. There are two metrics to measure performance for interconnects: bandwidth and latency. Bandwidth is the amount of data that can be transmitted over the interconnect hardware in a fixed period of time, while latency is the time to prepare and transmit data from a source node to a destination node.

With the current popularity of cluster computing, it is increasingly important to understand the capabilities and potential performance of various network interconnects for clusters. Furthermore, due to the low cost of clusters and their growing acceptance within the scientific community, many recent cluster builders are not computer scientists or engineers and thus have limited technical computing skills. This new group of cluster builders is less interested in features as Network Interface Card (NIC) programmability and special messaging libraries. Instead, they are concerned with two primary factors: cost and performance. While cost is easily determined and compared, performance is more difficult to assess particularly for users who may be new to cluster computing.

5. CONCLUSION

Cloud computing is a new technology of computer network, providing the web services at lower cost comparing to normal technique. It contributes to improve the services in other related technologies such as Grid computing, cluster and utility computing. Presently, the security in clouds is less than the model in grid environment. Clusters and grids solve the problem of performance improvement in different ways and each has classes of applications which it services better than the other (and, of course, there is some overlap). However, the true power and flexibility of distributed computing is realized when the techniques of clusters and grids are combined. As processing power has increased so have the processing demands of applications. Grid computing and cluster computing provide a hardware and software infrastructure that significantly increases the processing power available to applications.

REFERENCES

[1]. Robert W. Lucky May 2009, Reflections Cloud computing, May 2009, IEEE Spectrum.

[2]. Mladen A. Vouk, Department of Computer Science, North Carolina State University, Raleigh,

[3]. C .Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. loud "Computing And Grid Computing 360 Degree Compared"

[4]. Jadeja Yashpal Singh and Modi Kirit (2012) "Cloud Computing- Concepts, Architecture and Challenges", International Conference on Computing, Electronics and Electrical Technologies [ICCEET], IEEE

[5]. Buyya Rajkumar, Yeo Chee Shin, Venugopal Srikumar, Broberg

[6]. James and Brandic Ivona, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems (2009),

[7]. Gandotra Indu, Abrol Pawanesh, Gupta Pooja, Uppal Rohit and Singh Sandeep (2011) "Cloud Computing Over Cluster, Grid Computing: a Comparative Analysis", Journal of Grid and Distributed Computing, pp-01-04

[8]. Raicu Ion (2008), "Cloud Computing and Grid Computing 360 Computing 360--Degree Compared", Distributed Systems Laboratory, Computer Science Department, University of Chicago Introduction to Grid Computing, Bart Jacob, Michael Brown, Kentaro Fukui, Nihar Trivedi; IBM, Red books

[9]. Buyya R., Yeo C. S., Venugopal S., Broberg J., and Brandic I. (2009) Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing the 5th utility, Future Generation Computer Systems.

[10]. Maria S. Perez. "Grid and Cloud Computing".

[11]. A. T. Velte, T. J. Velte, and R. Elsenpeter, Cloud Computing-A Practical Approach, The McGraw-Hill Companies, New York, 2010.

[12]. K. Kaur, and S. Vashisht. "Data Separation Issues in Cloud Computing", International Journal for Advance Research in Engineering and technology, I (10), pp.26-29, November, 2013.