Indian Journal of Science and Technology, *Vol 8(29), DOI: 10.17485/ijst/2015/v8i29/86860, November 2015*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Cloud and Grid Part I: Difference and Convergence

Anna Melekhova[1*] and Vladimir Vinnikov[2]

[1]Parallels, Moscow, Russian Federation
[2]Acronis, Moscow, Russian Federation

Abstract

The first part of this paper is concerned with the generalized models of grid and cloud architectures as well as their mutual convergence. We study the examples of contemporary grids and clouds and next-generation mechanisms of achieving their unification into global confederation of clouds. The types and properties of virtual resources as well as approaches to their appropriate and efficient management in virtual machines are considered in the next following second part of the paper.

Keywords: Ballooning, Cloud Computing, Cloud Storage, Cluster as a Service, Data Grid, Grid Computing, Grid-Cloud Integration, Grid as a Service, Hypervisor, Intercloud, On-Demand Grid, Virtual Machine Oversubscription.

1. Introduction and Overview of the Literature

An idea of interactions between computers dates back to 1960s when the first commercially available modem had been released. Further technical and algorithmic advances allowed one to not only exchange messages, but also to provide access to databases, file storage and computational resources via standard protocols widely used nowadays. Initially, emerging technologies defined user needs. However, user demands race with available capabilities ever since and now they are at the point of overgrowing the latter. Compliance to these needs becomes a global challenge to modern applied technologies and science. The most promising solutions lie in the area of distributed computer activities.

Current distributed computer infrastructures originate from parallel clusters and are divided into following classes: grids and clouds. A cluster is usually considered as a group of computers deployed in a single location and tightly interconnected via high bandwidth network[51]. The works[12,50] define clusters as follows: “A cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource.” The nodes constituting the network are homogeneous in software and hardware specifications. The cluster operates mostly in shared memory mode and provides an interface to mimic a single physical machine.

There are no strict standard and widely accepted definitions for grids[24,26,10] thus their attributes sometimes overlap with that of the clouds and the mutual differences are somewhat fuzzy. However, various classifications are presented in papers[24,29,53,63,36].

A grid is devised as a collection of loosely interconnected heterogeneous computers at different locations with varying operating systems and hardware under multiple ownership and decentralized management[29,36]. The node of the grid can represent either a single machine or a whole cluster. Buyya[13] gave one of the popular definitions for grids as follows: “*A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed ‘autonomous’ resources*

* *Author for correspondence*

dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements." The grid definition by[10] is *"a large-scale geographically distributed hardware and software infrastructure composed of heterogeneous networked resources owned and shared by multiple administrative organizations which are coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. These applications can perform either distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing.*" According to[32] a grid is designed for scheduled computationally intensive operations on few large allocation requests. It is a common agreement that a grid should provide the sharing of computational resources, storage elements, specific applications and equipment not subjected to a centralized control via standard open general purpose protocols and interfaces[24,53]. Implementation of these features constitutes an additional layer of abstraction over the cluster. However, such level of abstraction is still not sufficient for common users to handle effectively general-purpose tasks.

A cloud is built upon a grid and it is devised as its generalization aimed to resolve previous complexities. As stated by[29] the cloud computing is: "*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*". According to Buyya, "*A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers.*" In contrast to the grid hardware resource sharing, the cloud provide its resources as service models at high layers of abstraction, namely: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). The NIST (2011) defines these terms as follows:

- IaaS. "*The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).*"
- PaaS. "*The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.*"
- SaaS. "*The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.*"

These service models are the part of the global principle everything as a service (XaaS). The examples of this conception are the interim layers "cluster as a service"[20,66] and "grid as a service"[1] between infrastructure and platform services. The unified interfaces allow one to form federation of clouds. Since there is no evidence on the theoretical limits on scalability, the extreme case of the federation would be the global cloud of all existing computational devices, making up the Next generation of internet[32].

This first part of this paper is concerned with an overview of the generalized models of grid and cloud architectures as well as their mutual convergence via virtualization, along with occurring obstacles like performance degradation of various nature. In the following second part of the study, we consider the types, properties of virtual resources along with their appropriate management, and propose novel ballooning approach to memory balancing on nested virtual machines.

1.1 Resource Virtualization on Physical Machine

Modern hardware performance is sufficient to emulate multiple computer systems on a single physical machine. Such emulated computer system is called a virtual machine and it is capable to run the majority of applications in a manner undistinguishable from that of a real counterpart. This term should not be confused with Java or Inferno virtual machines that only provide runtime environment for corresponding program code. The software, firmware, or hardware to run virtual machines are called hypervisor, virtualization engine, virtualization module, or virtualization system. A physical machine running a hypervisor is called a host. A software running in an emulated environment is called a guest.

A hypervisor maps available physical resources to virtual ones and distributes them among sibling virtual machines. The following virtual resources are distributed: CPU, memory, storage, and network. The CPU distribution relies on thread scheduling. To overcome single thread bottlenecks and achieve high performance the scheduler should utilize hyper-threading and multi-core technologies, as well as emerging chip-level multiprocessing[9].

The storage distribution is based on conception of "virtual hard disk" file format representing an image of storage devices within a corresponding VM. The concept of virtual storage contributes additional throughput and latency bottlenecks due to abstraction from underlying physical devices and shared access to them[37]. One approach to solve this issue is to refashion the traditional disk scheduler algorithms of the guest OS into workload-oriented form while restricting its hypervisor counterpart to a minimum activity[11]. The paper[40] shows that the host-guest combination of nested file systems also affects disk I/O performance depending on a prevailing workload pattern and is a subject to careful experimental selection.

The distribution on network resource is implemented via virtual networking, simulating various network infrastructure on top of existing hardware components. The virtual networking resource is also a subject to performance degradation arising from traffic interference between multiple single-hosted guests[47]. Non-optimal management of proportional sharing by conventional I/O schedulers leads to excessive triggering of congestion avoidance and results in additional delays[38]. One of the proposed solutions introduces the concept of Differential Virtual Time (DVT) and implements a latency smoothing host I/O scheduler, preserving fair proportionality and improving performance isolation across VMs.

The memory virtualization is the most challenging among resource mapping. This mapping has three levels of abstraction[62].

- Host physical memory. It is used by a hypervisor and treated as available on the system.
- Guest physical memory. It is maintained by hypervisor as contiguous addressable memory space and used by the guest OS running on the VM.
- Guest virtual memory. It is managed by the guest OS to applications and is used by them.

Some researchers consider the total host physical memory as a sum of fast volatile RAM and virtual pages on lower bandwidth media like magnetic or flash disks. However, the disk swapping on a host results in significant performance drop and means a resource exhaustion that should be avoided.

The host operating system via hypervisor is unable to manage virtual machines[61] since it cannot take pages from the guest transparently: the guest would be unaware of the mapping change and it would continue to work with the memory that does not belong to it. Such a situation could lead to unpredictable damages. So hypervisor have to take care of memory scheduling. It gets the resource from host OS and redistributes it among virtual machines. The most effective policy is to provide the resource in accordance to the guest's resource demand. However, the estimation of the demand is in turn a complex problem known as the problem of physical memory size estimation.

There are following approaches to physical memory size optimization: content-based sharing, ballooning, memory compression, and page replacement. These algorithms are described in detail in[44,64]. Ballooning concerned in the paper[45] is widely considered as a most promising method.

The idea behind ballooning is to provide guest OS with an auxiliary driver which effectively reclaims guest physical memory for a hypervisor on its request by inflating or deflating within guest virtual memory like any typical application. The amount of committed balloon memory deduced from a guest OS at which this OS initiates page swapping approximates the amount of pages in physical guest memory unused by processes other than ballooning itself. The detailed description of this technique is provided in[45]. We consider ballooning and other resource balancing techniques in the second part of the study.

2. The Generalized Grid Model

Classical grids are described and discussed in multiple publications. The properties of a grid can be briefly tabulated as follows [15,63]:

- Business model is project-oriented. Schedule-requested service units for sale are usually measured in CPU-hours;
- Computing model is batch-based and implemented via local resource managers (LRM) like Condor, PGS or Sun Grid Engine;
- Data model is metadata-based. Corresponding storage is managed by distributed file systems such as PVFS, Lustre, GlusterFS etc.;
- Monitor model is hierarchically based and federation-targeted. Most common implementation is Ganglia scalable distributed system monitor;
- Programming model is heterogeneous MPI-oriented. Widely used packages include MPICH-G2 (Karonis et al, 2003) from Globus Toolkit[25], GridRPC (Seymour et al, 2002), Workflow systems, WSRF (Web Services Resource Framework) etc;
- Security model is delegation-based. An asymmetrically encrypted signature is commonly implemented in GSI-toolkit[28].

The well-known examples of grids include: PrimeGrid, GPGPUGrid.net, World Community Grid, SETI, Folding@home, GIMPS (Great Internet Mersenne Prime Search), WLCG (Worldwide LHC Computing Grid) etc. Most of grid computing projects are based on BOINC middleware (Berkeley Open Infrastructure for Network Computing).

Grid abstraction layer hides variations in the underlying basic technologies (e.g. computer clusters, storage managers, application services, etc.) and is provided by the middleware, which implements a set of services and protocols to aggregate resources in a grid. Middleware services perform information discovery and monitoring, resource management, security policies, grid scheduling, load balancing, and data management[2].

The architecture of these services is defined by following objective-specific grid categories[39]

- Computational grids, delivering application performance via supercomputing and high throughput;
- Data grids, improving data access;
- Service grids, providing enhanced on-demand, collaborative, and multimedia services.

The computational grid category consists of distributed systems that grant single applications a high aggregated computational capacity far exceeding that of any employed machine. Supercomputing mode executes single application instance in parallel to reduce overall completion time. High throughput mode increases the completion rate of a stream of job tasks.

The data grid category is devised to provide specialized infrastructure for intensive information processing such as synthesis of new data from digital libraries in a wide area network. The main data grid activity comprises of unified infrastructure-based data services across repositories, while a computational counterparts rely on application-based implementations of storage management and data access schemes. Typical data grid tasks include large-scale data mining to correlate information from multiple different data sources. The main developers and administrators of large-scale data organization, catalog, management, and access technologies are European Grid Infrastructure (http://www.egi.eu/) and Globus Aliance (https://www.globus.org/).

The service grid category denotes the systems that provide distributed yet collective services unavailable by means any single machine. A collaborative grid manages collaborative workgroups, allowing users and applications to interact in real time within a virtual workspace. An on-demand grid category dynamically allocates various resources to provide new services or scale up existing ones. A multimedia grid provides an infrastructure for real-time multimedia applications. This involves mandatory support of distributed QoS (Quality of Service) in contrast to a single dedicated machine where such functionality is arbitrary.

Generally the large-scale grids, especially data-grids, employ data availability mechanisms, initially intended to decrease the data access latency and the network bandwidth consumption via replication. Studies[52] indicate that different replica strategies (e.g. Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replica and Fast Spread) are suited best for various user access patterns (e.g. random access, small temporal locality, and small geographical and temporal locality). Economical auction-based models for long-term optimal replica decisions[8] as well as a network proximity dynamical replication HBR[49] are also developed. Since data access latency is dramatically reduced via contemporary technologies, the modern replication methods solve grid reliability problems. Address[41] the system-wide data availability problem, presenting two new data availability metrics (the System File Missing Rate and the System Bytes Missing Rate) and propose a novel heuristic algorithm that minimizes the Data Missing Rate (MinDmr) in the limited replica storage.

One of the current challenges is the development of general-purpose grid systems that possess capabilities of all above mentioned objective-specific grid categories. A classical grid can be straightforwardly upgraded to the generalized version if one substitutes physical machines with their virtual counterparts[23,57]. Such replacement creates new higher layer of abstraction, allowing dynamical instantiation and migration of virtual machines (VM). This generalized grid model regards entire computing environments as three independent logical entities (computation, state, and user data) which are handled as traditional OS processes and files and can be mapped onto servers of corresponding types. The VM hosts run dynamic VM guests (VM images) and represent computational resource. The data servers handle user data and represent storage resource. The image servers compress and archive static VM states and represent memory management resource. Various benchmark results indicate that the total overhead for deployed virtual grid can be at the acceptable level of 4.2 percent[23].

The In-Vigo system[3] is another example of decoupling the architecture of hardware and operational behavior of software resources from their physically implemented instances via the means of virtualization technologies. The In-VIGO approach to virtualization is depicted in Figure 1.

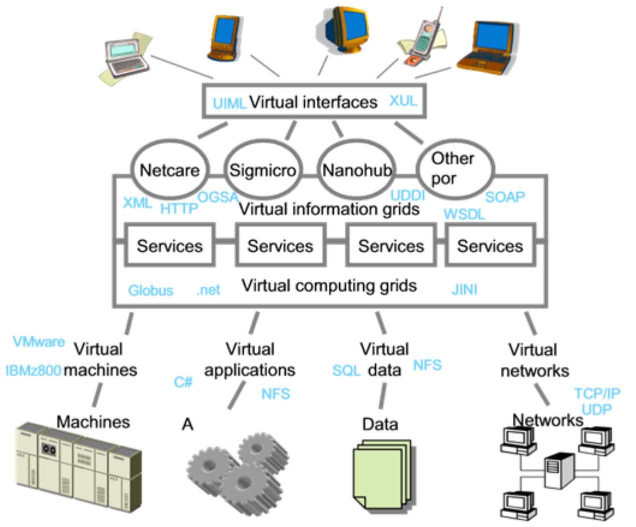

Figure 1. The In-VIGO approach according to[3].

The In-VIGO augments the traditional grid computing model with three additional layers of virtualization. The first layer aggregates the elementary components of a virtual computing grid into pools of virtual resources such as virtual machines, virtual data, virtual applications and virtual networks. This layer maps jobs to virtual resources (VMs) that are managed across domains and physical environments (e.g. physical machines with various OS at different locations). The second layer instantiates grid applications as services connected on demand to create virtual information grids. This layer supports multiple grid-computing mechanisms (e.g. Globus, Condor-G, .NET and JXTA) to run applications and employs encapsulation to compose them as services (e.g. via OGSI, OGSI.NET and Jini) and hide implementation details[27]. The third layer manages virtualization of interfaces (e.g. XML and UIML) from aggregated services, in order to customize displaying by various devices (e.g. as HTML for laptop, WHML for a palmtop and WAP WML for a cell phone). In other words the first layer decouples resource allocation for applications from jobs management, the second layer decouples the service composition and usage from the execution management of the underlying applications, and the third layer decouples the generation of service interfaces from corresponding device-specific rendering.

At present moment only the first layer is successfully implemented with exception of virtual networks. Users are allowed to developed applications and are provided with interactive and batch-oriented interfaces for grid-enabled tools. The second and third proposed layers are de-facto aimed to implement the functional paradigm of clouds.

3. The Generalized Cloud Model

Classical clouds present the most recognized solutions for reliable processing and storage of large amounts of general-purpose data. For example, the vast impact of emerging cloud technologies is stated in[6] "*The first similarity between cloud computing and traditional utility models, electricity or telephony, for example, is that they all have characteristics of a disruptive general-purpose technology which make a surge of associated innovation possible.*" Contrary to classic grids, the cloud provides users not with granted machine resources on schedule but with services on demand. The cloud properties can be described in the following list[29,30,63]

- Business model is customers-oriented. Payments are usually a posteriori defined on consumption level (pay-as-you-go model);
- Computing model is demand-based. Distributed resources in the cloud are instantly shared among all active users;
- Data model is MapReduce-based. Corresponding underlying storage is managed by distributed file systems such as GoogleFS, Lustre, GlusterFS etc.;
- Monitor model is server-based. The dedicated implementation is arbitrary;
- Programming model is MapReduce-oriented. Widely used implementation include Hadoop and its analogs in other scripting languages;
- Security model is web-based. Implementation relies on webforms and SSL-protocol. Strong encryption is not fully resolved.

The well-known examples of clouds include Amazon Web Service (AWS), Amazon Elastic Compute Cloud (EC2), Google apps, etc.

The Cloud computing infrastructures are divided into several categories: commercial clouds (e.g. Amazon EC2), scientific cloud (e.g. Nimbus) and open-source technologies (e.g. Eucalyptus, Globus VWS). According to the NIST established definitions for deployment models[32,34,46] the cloud can also be classified as a private (internal) like the Eucalyptus or a public (external) like the Amazon EC2. Since these categories and classifications are not always mutually exclusive, many hybrid clouds exist, like CLEVER, GoGrid, VOC, OpenNebula, and Globus Nimbus[2].

Clouds usually provide following basic service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS)[19]. The first model of service provides computer infrastructure in a form of a hardware such as storage, servers and data center space or network components. Second model offers an access to an OS environment to develop and run applications, while hiding all infrastructure aspects from the scope of the user. The third model implements highest abstraction level, bringing to the user only typical commonly used software applications.

Much like their grid-counterparts, clouds follow the trend of generalization. For example the software solution of Ravello systems takes advantage of the nested virtualization (Figure 2-3), which provides a mechanism to construct a unified intercloud storage. In particular, this advanced virtualization made possible to acquire on demand identical 4000 VM from AWS and 1000 from Google Compile Platform to organize training for networking engineers with each workspace consisting of five VM[22].

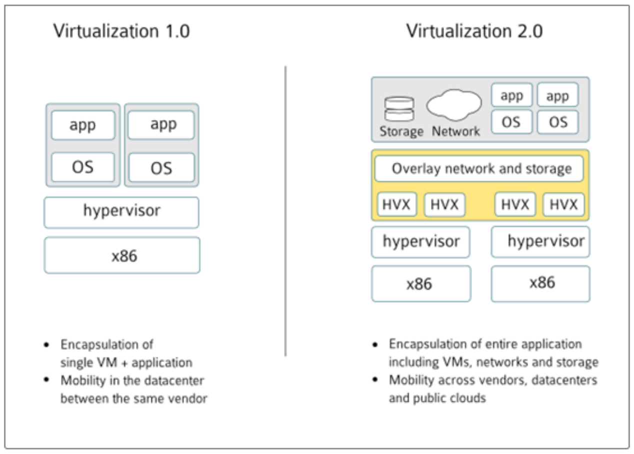
Virtualization 1.0
app
app
OS
OS
hypervisor
x86
• Encapsulation of single VM + application
• Mobility in the datacenter between the same vendor
Virtualization 2.0
Storage
Network
app
app
OS
OS
Overlay network and storage
HVX
HVX
HVX
HVX
hypervisor
hypervisor
x86
x86
• Encapsulation of entire application including VMs, networks and storage
• Mobility across vendors, datacenters and public clouds

Figure 2. Second generation of virtualization compared to the predecessor: The courtesy of Ravello systems.

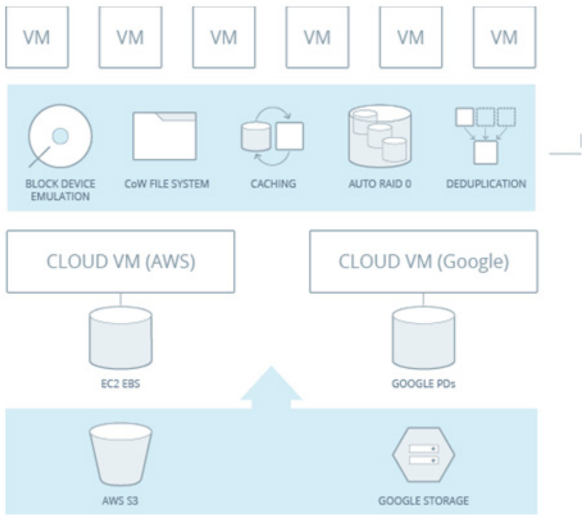

Figure 3. Unified intercloud storage via nested virtualization: The courtesy of Ravello systems.

4. Model Convergence

Grids and clouds in many aspects share similar features and purposes. For example, computational grid corresponds to cloud computing, data-grid resembles cloud storage, and interaction grid matches cloud collaboration. Both distributed systems consist of the typical elements and processes with the same role: data, metadata and client nodes, as well as replication, monitoring and load-balancing procedures.

However, conventional grids and clouds have differences in operation schedules and user interaction models. Cloud schedulers are designed to maintain system and data integrity via regular scrubbing (checksum validation). The scrubbing procedure is vital for detection of latent bit errors on hard drives[18,33,42,48,55]. Grid schedulers behave similarly, but their main load comes from queued user jobs. Contrary to that, on-demand user services are provided by clouds, but not by classical grids.

Recent publications show the intensifying research on the unification of the Grid and Cloud computing[4,14,56]. The popular different approaches to the integration, namely, the "Grid on Cloud" and the "Cloud on Grid" undergo consolidation into new Grid-Cloud integration paradigm, accommodating advances in architectures, communications and user demand patterns. The publication of[2] presents taxonomy for the classification of grid-cloud integration: the disjointed, the partial and the full grid-cloud integration. This paper also considers the software tools used for on-demand grid deployment over cloud infrastructure and contributes comprehensive references on corresponding studies.

For example[7] considered two scenarios for deployment of scientific phylogeny application MetaPIGA on combined Grid/Cloud architecture. The challenge was to use joint advantages of Grid and Cloud infrastructures to build a high performance, reliable and open platform. In the first case, the Cloud infrastructures provided no direct access to the clients, and all interactions obligatory passed through the Grid. In the second case, the Cloud accepted tasks directly submitted by the clients. The verifications had been carried out via the MetaPIGA system deployed on Amazon, Azure and VenusC Cloud infrastructures. Presents[16] another example of obtaining a computational resource through a Grid middleware (DIET) using existing Cloud infrastructure (EUCALYPTUS). The research of Di Costanzo, et al. shows that the InterGrid system can be used to build scalable virtualized computational environments working on cloud infrastructures, such as EUCALYPTUS and Amazon EC2.

The emerging unified grid-cloud systems employ so high abstraction layers to the client side that users receive mutually separated pure uniform resources. Computational capacity and storage space are provided explicitly and network resource is delivered implicitly. The complete separation of virtualized forms of CPU, Storage and Network resources from their physical counterparts forms an NVF-infrastructure (Network Function Virtualization) which allows to construct an distributed network (grid or cloud) from above mentioned typical virtual elements. This model is under implementation via Cloud Conductor with SLA-management developed by ARCCN[60].

The study[54] reveals that despite recent advances up-to-date hardware capabilities like I/O performance still impede full virtualization for low latency and high throughput data processing. However, the Intel roadmap papers[5,9] suggest that future microprocessors will possess several levels of virtualization, concealing the hardware details from the system software and acting as unified yet partitionable virtual machines with global interfaces.

5. Discussion

The reviewed studies consider the virtualization as one of key factors in achieving grid-cloud convergence into globally distributed architecture with standardized interfaces at highest abstraction levels. However, the virtualization is rivaled by alternative technologies, namely: OS containers and application containers[59]. These technologies essentially provide operating system virtualization discarding hardware emulation level and sharing the same kernel of the host OS. The main benefit of omitting hardware abstraction layer is the much lower performance overhead[21] allowing the physical host to run more containers that virtual machines. On the other hand, the containers suffer from much weaker isolation and security[65]. Still, the containers allow optimal usage while running within guest OS and providing auxiliary functionality that is insignificant to employ dedicated VM.

6. Conlusion

In the conducted study, we showed differences between grids and clouds as well as both the premises and obstacles to mutual convergence of these distributed architectures. The analysis of related works confirms that virtualization plays significant role in system scalability, interoperability across the various systems, formation of interclouds, unification into confederation of clouds and eventual emergence of next generation of internet. This paper also reveals that the remaining obstacles to complete virtualization consist of hardware bottlenecks, including CPU, storage, and network I/O congestion. Ongoing high-priority research considers the means to overcome these performance issues and leverage latencies via both novel algorithmic means and advanced hardware solutions. In the following second part of the paper, we provide the detailed study the types, properties of virtual resources along with their appropriate management, and propose novel ballooning approach to memory balancing on nested virtual machines.

7. Acknowledgments

The article is published within applied scientific research performed with financial support of the Ministry of Education and Science of the Russian Federation. Subsidy provision agreement 14.579.21.0010. Universal identifier of the agreement is RFMEFI57914X0010.

8. References

1. Abdulhamid SM, Latiff A, Shafie M, Bashir M B. Scheduling techniques in on-demand grid as a service cloud: a review. Journal of Theoretical & Applied Information Technology. 2014a; 63(1). Available from: http://www.jatit.org/volumes/Vol63No1/2Vol63No1.pdf
2. Abdulhamid SM, Latiff MSA, Bashir MB. On-demand grid provisioning using cloud infrastructures and related virtualization tools: a survey and taxonomy. arXiv preprint arXiv. 2014b; 1402.0696. Available from: http://arxiv.org/ftp/arxiv/papers/1402/1402.0696.pdf
3. Adabala S, Chadha V, Chawla P, Figueiredo R, Fortes J, Krsul I, Zhu X. From virtualized resources to virtual computing grids: the In-VIGO system. Future Generation Computer Systems. 2005; 21(6):896-909. Available from: http://users.cs.fiu.edu/~zhaom/research/fgcs.pdf
4. AlHakami H, Aldabbas H, Alwada'n T. Comparison between cloud and grid computing: review paper. International journal on cloud computing: services and architecture (IJCCSA). 2012:2(4). Available from: http://www.airccse.org/journal/ijccsa/papers/2412ijccsa01.pdf
5. Balakrishnan S, Rajwar R, Upton M, Lai K. The impact of performance asymmetry in emerging multicore architectures. ACM SIGARCH Computer Architecture News. IEEE Computer Society. 2005 June; 33 (2):506-517.
6. Bayrak E, Conley JP, Wilkie S. The economics of cloud computing. The Korean Economic Review. 2011; 27(2):203-230.
7. Belgacem MB, Hafsi H, Abdennadher N. A Hybrid Grid/Cloud Distributed platform: a case study. In Grid and Pervasive Computing Springer Berlin Heidelberg; 2013. p. 162-169.
8. Bell WH, Cameron DG, Carvajal-Schiaffino R, Stockinger K, Zini F. Evaluation of an economy-based file replication strategy for a data grid. In Cluster Computing and the Grid. Proceedings of the 3rd IEEE/ACM International Symposium; 2003 May. p. 661-668. Available from: https://gridpp.ac.uk/papers/OptorSimCCGrid2003.pdf
9. Borkar S, Dubey P, Kahn K, Kuck D, Mulder H, Pawlowski S, Rattner J. Platform 2015: Intel processor and platform evolution for the next decade. Technology. 2005; 1.
10. Bote-Lorenzo ML, Dimitriadis YA, Gómez-Sánchez E. Grid characteristics and uses: a grid definition. In Grid Computing. 2004 Jan. p. 291-298.
11. Boutcher D, Chandra A. Does virtualization make disk scheduling passé?. ACM SIGOPS Operating Systems Review. 2010; 44(1):20-24.
12. Buyya R., editors. High Performance Cluster Computing: Architectures and Systems. Vol. 1, Prentice Hall, Upper Saddle River. USA; 1999.
13. Buyya, R. Economic Paradigm for Service-Oriented Grid Computing. Paper presented at: The INT Media's Grid Computing Planet Conference and Expro; 2002 June 17-18; San Jose, California, USA.
14. Buyya R, Ranjan R, Calheiros RN. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In Algorithms and architectures for parallel processing. Springer Berlin Heidelberg. p.13-31. Available from: http://www.buyya.com/papers/InterCloud2010.pdf
15. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems. 2009b; 25(6):599-616.
16. Caron E, Desprez F, Loureiro D, Muresan A. Cloud computing resource management through a grid middleware: A case study with DIET and eucalyptus. In Cloud Computing, 2009. p. 151-154. Available from: https://hal.inria.fr/inria-00435785/document
17. Costanzo AD, Assunção D, Dias M, Buyya R. Harnessing cloud technologies for a virtualized distributed computing infrastructure. Internet Computing. 2009; 13(5):24-33. Available from: http://www.cloudbus.org/~raj/papers/InternetComp-Cloud-2009.pdf
18. Daci G, Bezhani A. A Review of Disc Scrubbing and Intra Disk Redundancy for Reducing Data Loss in Disk FileSystems. Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing; 2013 Jan; Springer New York; 2013. P. 585-589
19. Dillon T, Wu C, Chang E. Cloud computing: issues and challenges. In Advanced Information Networking and Applications (AINA). Proceedings of the 24th IEEE International Conference on; 2010 April; P. 27-33.
20. Doelitzscher F, Held M, Reich C, Sulistio A. (2011, November). Viteraas: Virtual cluster as a service. In Cloud Computing Technology and Science (CloudCom). Proceedings of the 2011 IEEE Third International Conference on. IEEE. P. 652-657 Available from: http://wolke.hs-furtwangen.de/assets/downloads/CRL-2010-03.pdf

21. Felter W, Ferreira A, Rajamony R, Rubio J. An updated performance comparison of virtual machines and linux containers. Technology. 2014; 28, 32.
22. Ferry J. The Men Who Broke the Cloud. Daily Cloud. 2015 Jun
23. Figueiredo RJ, Dinda P, Fortes J. A case for grid computing on virtual machines. In Distributed Computing Systems, 2003. Proceedings of the 23rd International Conference on. IEEE. 2003 May. P. 550-559 Available from: http://www.presciencelab.org/Virtuoso/icdcs03.pdf
24. Foster I. What is the GRID? A three point checklist. GRIDtoday. 2002 Jul 22; 1 (6) Available from: http://dlib.cs.odu.edu/WhatIsTheGrid.pdf
25. Foster I, Kesselman C. The Globus project: A status report. Future Generation Computer Systems. 1999; 15(5):607-621.
26. Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Journal of High Performance Computing Applications. 2001; 15(3):200-222.
27. Foster I, Kesselman C, Nick JM, Tuecke S. The physiology of the grid. Grid computing: making the global infrastructure a reality. 2003. P. 217-249.
28. Foster I, Kesselman C, Tsudik G, Tuecke S. (1998, November). A security architecture for computational grids. Proceedings of the 5th ACM conference on Computer and communications security; 1998 Nov; ACM; 1998. P. 83-92 Available from: http://cseweb.ucsd.edu/groups/csag/html/teaching/cse225s04/Reading List/gsi-security.pdf
29. Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In Grid Computing Environments Workshop. IEEE; 2008. P. 1-10 Available from: http://arxiv.org/ftp/arxiv/papers/0901/0901.0131.pdf
30. Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Stoica I. Above the clouds: A Berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS; 2009; 28, 13.
31. Grimshaw A. What is a Grid? Grid Today. 2002; 1(26).
32. Hashemi SM, Bardsiri AK. Cloud Computing Vs. Grid Computing, ARPN Journal of Systems and Software. 2012; 2(5) Available from: http://scientific-journals.org/journalofsystemsandsoftware/archive/vol2no5/vol2no5_4.pdf
33. Iliadis I, Haas R, Hu XY, Eleftheriou E. Disk scrubbing versus intra-disk redundancy for high-reliability raid storage systems. In ACM SIGMETRICS Performance Evaluation Review. 2008; 36(1):241-252.
34. Indi TS, Dangare CS. Cloud Computing: Overview & Comparison with Grid Computing. International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE). 2014; 4(3) Available from: http://www.ijarcsse.com/docs/papers/Volume_4/3_March2014/V4I3-0426.pdf
35. Karonis NT, Toonen B, Foster I. MPICH-G2: A grid-enabled implementation of the message passing interface. Journal of Parallel and Distributed Computing. 2003; 63(5):551-563 Available from: http://arxiv.org/pdf/cs/0206040.pdf
36. Kaur K, Rai AK. A comparative analysis: Grid, cluster and cloud computing. International Journal of Advanced Research in Computer and Communication Engineering. 2014; 3(3):5730-5734 Available from: http://www.ijarcce.com/upload/2014/march/IJARCCE9BaanjanAComparativeAnalysis GridCluster and Cloud Computing.pdf
37. Kesavan M, Gavrilovska A, Schwan K. On disk I/O scheduling in virtual machines. Proceedings of the 2nd conference on I/O virtualization; 2010 March; USENIX Association; 2010. P.6
38. Kesavan M, Gavrilovska A, Schwan K. Differential virtual time (DVT): rethinking I/O service differentiation for virtual machines. Proceedings of the 1st ACM symposium on Cloud computing; 2010 June; ACM; 2010. P. 27-38
39. Krauter K, Buyya R, Maheswaran M. A taxonomy and survey of grid resource management systems for distributed computing. Software-Practice and Experience. 2002; 32(2):135-64. Available from: http://www.cloudbus.org/papers/gridtaxonomy.pdf
40. Le D, Huang H, Wang H. Understanding performance implications of nested file systems in a virtualized environment. In FAST; 2012. P. 8.
41. Lei M, Vrbsky SV, Hong X. An on-line replication strategy to increase availability in Data Grids. Future Generation Computer Systems. 2008; 24:85-98. Available from: https://www.researchgate.net/profile/SV_Vrbsky/publication/222707797_An_on-line_replication_strategy_to_increase_availability_in_Data_Grids/links/0a85e52d97a158e2a0000000. pdf
42. Liu J, Zhou K, Wang Z, Pang L, Feng D. Modeling the Impact of Disk Scrubbing on Storage System. Journal of Computers. 2010; 5(11):1629-1637.
43. McLaughlin, K. Startup Ravello Pitches Nested Virtualization As Way To Solve Tough Cloud Challenges. CRN; 2015 Mar 2. Available from: http://www.crn.com/news/virtualization/300075964/startup-ravello-pitches-nested-virtualization-as-way-to-solve-tough-cloud-challenges.htm
44. Melekhova A. "Machine Learning in Virtualization: Estimate a Virtual Machine's Working Set Size", Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference, June 28 2013-July 03 2013.
45. Melekhova A, Markeeva L. Estimating Working Set Size by Guest OS Performance Counters Means. CLOUD COMPUTING. 2015; 48.
46. NIST SP 800-145, "A NIST definition of cloud computing"; 2011. Available from: http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf
47. Ongaro D, Cox AL, Rixner S. Scheduling I/O in virtual machine monitors. Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments; 2008 Mar; ACM; 2008. P. 1-10.
48. Pâris JF, Schwarz ST, Amer A, Long DD. Improving disk array reliability through expedited scrubbing. In Networking, Architecture and Storage (NAS). Proceeding of the 2010 IEEE Fifth International Conference on; 2010 Jul; IEEE; 2010. P. 119-125.

49. Park SM, Kim JH, Ko YB, Yoon WS. Dynamic data grid replication strategy based on Internet hierarchy. In Grid and Cooperative Computing. Springer Berlin Heidelberg; 2004. P. 838-846.
50. Pfister GF. In Search of Clusters. 2nd ed., Prentice Hall, Upper Saddle River, USA; 1998.
51. Ramamoorthy G. Distributed systems and cloud computing. A Comparative Study; 2011 Available from: http://www.resumegrace.appspot.com/pdfs/DistributedSystemsandCloudComputing.pdf
52. Ranganathan K, Foster I. Identifying dynamic replication strategies for a high-performance data grid. In Grid Computing—GRID 2001. Springer Berlin Heidelberg; 2001. P. 75-86 https://www.researchgate.net/profile/Ian_Foster/publication/2413833_Identifying_Dynamic_Replication_Strategies_for_a_High-Performance_Data_Grid/links/02e7e51bc1b4a47b4d000000.pdf
53. Rani S, Suri PK. Grid Computing and Cloud Computing: Description and Comparision. International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE). 2012; 1(7):136.
54. Regola N, Ducom JC. Recommendations for virtualization technologies in high performance computing. In Cloud Computing Technology and Science (CloudCom). Proceeding of the 2010 IEEE Second International Conference on; 2010 Nov; IEEE; 2010. P. 409-416.
55. Rozier EW, Belluomini W, Deenadhayalan V, Hafner J, Rao KK, Zhou P. Evaluating the impact of undetected disk errors in raid systems. In Dependable Systems & Networks, 2009. DSN'09. Proceeding of the IEEE/IFIP International Conference on; 2009 Jun; IEEE; 2009. P. 83-92.
56. Sadashiv N, Kumar SD. Cluster, grid and cloud computing: A detailed comparison. In Computer Science & Education (ICCSE). Proceeding of the 2011 6th International Conference on; 2011 Aug; IEEE; 2011. P. 477-482. Available from: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=FE5D1EF1A8B4AE8B6E48E823C2521F63?doi=10.1.1.465.8919&rep=rep1&type=pdf
57. Santhanam S, Elango P, Arpaci-Dusseau AC, Livny M. Deploying Virtual Machines as Sandboxes for the Grid. In WORLDS. 2005; 5:7-12. http://research.cs.wisc.edu/htcondor/doc/SandboxingWorlds053.pdf
58. Seymour K, Nakada H, Matsuoka S, Dongarra J, Lee C, Casanova H. Overview of GridRPC: A remote procedure call API for Grid computing. In Grid Computing—GRID. Springer Berlin Heidelberg; 2002:274-278. https://www.researchgate.net/profile/Craig_Lee/publication/221548007_Overview_of_GridRPC_A_Remote_Procedure_Call_API_for_Grid_Computing/links/00b49519a68aced5b3000000.pdf
59. Strauss D. Containers-Not Virtual Machines, Are the Future Cloud. The Linux Journal. 2013; 228:118-123.
60. Tabolin V. "From Virtual Applience to Virtual Network Functions (VNF)"; 2015. Available from: http://www.slideshare.net/ARCCN/vnf
61. Tanenbaum A. Modern Operating Systems Third Edition; 2009. P. 209 – 210.
62. Understanding Memory Resource Management in VMware® ESX™ Server / VMware white paper; 2009. Available from: http://www.vmware.com/files/pdf/perf-vsphere-memory_management.pdf
63. Vachhani MK, Atkotiya KH. Similarities and Contrast between Grid Computing and Cloud Computing. Indian Journal of Applied Research (IJAR). 2013:3(3). Available from: http://www.worldwidejournals.com/ijar/file.php?val=March_2013_1362144057_53166_19.pdf
64. Waldspurger CA. Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review. 2002; 36(SI):181-194.
65. Walsh Daniel J. "Are Docker containers really secure?" Opensource.com. 2014 Jul 22. Available from: http://opensource.com/business/14/7/docker-security-selinux
66. Wuhib F, Stadler R, Lindgren H. Dynamic resource allocation with management objectives—Implementation for an OpenStack cloud. In Network and service management (cnsm). Proceedings of the 2012 8th international conference and 2012 workshop on systems virtualization management (svm). 2012 Oct; IEEE; 2012. P. 309-315. Available from: http://dl.ifip.org/db/conf/cnsm/cnsm2012/WuhibSL12.pdf