Journal of Computing and Information Technology - CIT 22, 2014, 2, 69–84
doi:10.2498/cit.1002381

A Review on Modern Distributed Computing Paradigms: Cloud Computing, Jungle Computing and Fog Computing

Majid Hajibaba and Saeid Gorgin

Department of Electrical and Information Technology, Iranian Research Organization for Science and Technology (IROST), Tehran, Iran

The distributed computing attempts to improve performance in large-scale computing problems by resource sharing. Moreover, rising low-cost computing power coupled with advances in communications/networking and the advent of big data, now enables new distributed computing paradigms such as Cloud, Jungle and Fog computing.

Cloud computing brings a number of advantages to consumers in terms of accessibility and elasticity. It is based on centralization of resources that possess huge processing power and storage capacities. Fog computing, in contrast, is pushing the frontier of computing away from centralized nodes to the edge of a network, to enable computing at the source of the data. On the other hand, Jungle computing includes a simultaneous combination of clusters, grids, clouds, and so on, in order to gain maximum potential computing power.

To understand these new buzzwords, reviewing these paradigms together can be useful. Therefore, this paper describes the advent of new forms of distributed computing. It provides a definition for Cloud, Jungle and Fog computing, and the key characteristics of them are determined. In addition, their architectures are illustrated and, finally, several main use cases are introduced.

Keywords: distributed system, cloud computing, jungle computing, fog computing

1. Introduction

The introduction of computer networks in the 1970s led to the development of distributed systems (Andrews, 1999). A distributed system is a collection of independent computers that appears to the user as a single computer (Tanenbaum & Steen, 2006) and provides a single system view. The coordinated aggregation of these distributed computers allows access to a large amount of computing.

Up to this time, a few technologies emerged in the distributed systems. Peer-to-peer (P2P) network is one of the primary distributed systems. However, an important class of distributed systems is the distributed computing system which uses for high performance computing tasks (Tanenbaum & Steen, 2006). In this way, with low-cost and more powerful personal computers, as well as high-speed networks, Cluster computing has become widely popular. Other well-known distributed computing paradigms, including Grid computing and Cloud computing, appeared with the evolution of the Internet in the mid-1990s and 2007, respectively.

Cloud computing has become the hottest technology within a few years (Qian, Luo, Du, & Guo, 2009). But, according to Gartner Hype Cycle for Emerging Technologies, 2013, Cloud computing has passed the "peak of inflated expectations" and is headed into the "trough of disillusionment" with two to five years away from its mature. (Gartner inc., 2013). Therefore, the trend in distributed systems is changing toward the use of newer computing paradigms.

Jungle computing came on the scene as a new paradigm to achieve better performance by using diverse and highly non-uniform distributed computing systems (Seinstra, et al., 2011; Kahanwal & Singh, 2012). Also, Fog computing

extended the Cloud computing paradigm to the edge of the network, in 2012, to enable a new kind of applications and services (Bonomi, Milito, Zhu, & Addepalli, 2012). A taxonomy of distributed computing paradigms is shown in Figure 1.

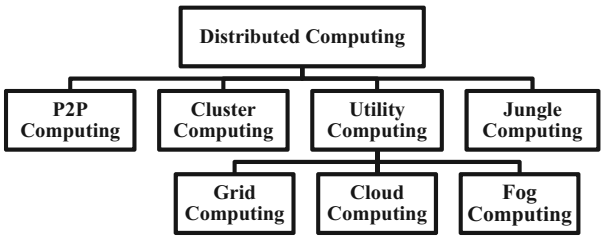

Figure 1. Taxonomy of distributed computing.

In this paper, we review these two new distributed computing paradigms, Jungle computing and Fog computing, along with Cloud computing which is related to them. We name these three as modern distributed computing paradigms. A review of these models and their characteristics helps to better understand modern distributed computing paradigms and their similarities and differences.

The rest of the paper is organized as follows. The Cloud computing model, characteristics and concepts are explained in Section 2. In Section 3, the Jungle computing is described and its characteristics are delineated. In Section 4, Fog computing and its characteristic are introduced. Finally, in Section 5, we conclude the paper with a summary of our review.

2. Cloud Computing

2.1. Overview

Cloud computing is neither a completely new concept (Antonopoulos & Gillam, 2010; Halpert, 2011) nor a new technology (Halpert, 2011; Yang, Chen, & Chen, 2012). It is just a new business operational model originated from other existing technologies such as Virtualization, SOA and Web2. Several definitions of Cloud computing exist in the academic and the commercial world (Buyya, Broberg, & Goscinski, 2011; Mell & Grance, 2009; Armbrust, et al., 2009; Vaquero, Rodero-Merino, Caceres, & Lindner, 2009; Forrest & inquiries, 2009), but we combine those definitions into a new one:

Cloud is a parallel and distributed system consisting of a shared pool of virtualized resources (e.g. network, server, storage, application, and service) in large-scale data centers. These resources can be dynamically provisioned, reconfigured and exploited by a pay-per-use economic model in which consumer is charged on the quantity of cloud services usage and provider guarantees Service Level Agreements (SLA) through negotiations with consumers. In addition, resources can be rapidly leased and released with minimal management effort or service provider interaction. Hardware management is highly abstracted from the user and infrastructure capacity is highly elastic.

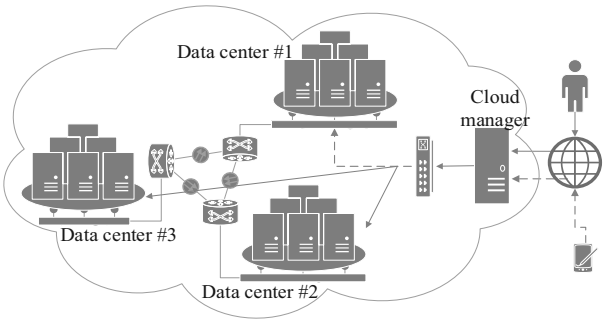

Figure 2. Cloud computing architecture.

Figure 2 depicts an abstract view of Cloud computing architecture. The aim is to concentrate computation and storage in data centers, where high-performance machines are linked by high-bandwidth connections, and all of these resources are carefully managed. The end-users make the requests that initiate computations and receive the results (Hayes, 2008).

In spite of the existing differences among definitions of Cloud computing, it has some basic characteristics that are described in Table 1.

There are some other characteristics of Cloud computing that are summarized in (Gong, Liu, Zhang, Chen, & Gong, 2010).

In addition to the outstanding characteristics, Cloud computing brought cost saving for consumers through removing capital expenses (CapEx) as well as reducing operating expenses (OpEx). In CapEx, this saving is achieved by eliminating the total cost of the entire infrastructures. In OpEx, saving is achieved by sharing

Characteristic	Comment
Virtualization	Hardware Virtualization mediates access to the physical resources, decouples applications from the underlying hardware and creates a virtualized hardware platform using a software layer. This software layer is virtual machine monitor (VMM), also called hypervisor, which creates and runs virtual machines (VM). A virtual machine is like a real computer, except that it uses virtual resources. Using Virtualization, virtual resources are isolated from each other and are independent of particular hardware, thus enabling the assignment of virtual resources to another physical hardware in case of capacity constraints or hardware failures. Through Virtualization, the underlying architecture is abstracted from the user while it still provides flexible and rapid access to it.
Multitenancy	This feature allows several customers (tenants) to share data center infrastructure, without being aware of it and without compromising the privacy and security of each customer's data (through isolation). Even though multitenancy is cost-effective, it causes performance degradation in simultaneously accessing shared services (multi-tenant interference) and performance unpredictability.
Service oriented architecture (SOA)	In this architecture everything is expressed and exposed as a service (Buyya, Broberg, & Goscinski, 2011) which delivers an integrated and orchestrated suite of functions to an end-user through composition of both loosely and tightly coupled functions (Vouk, 2008). Abstraction and accessibility are two keys to achieve the service oriented conception (Gong, Liu, Zhang, Chen, & Gong, 2010).
On-demand self-service	Cloud computing allows self-service access so that customers can request, customize, pay, and use services, as needed, automatically, without requiring interaction with providers or any intervention of human operators (Mell & Grance, 2009; Buyya, Broberg, & Goscinski, 2011).
Elasticity (dynamic provisioning)	To provide the illusion of infinite resources, more virtual *machines* (on two or more physical machines) can be quickly provisioned (scale out), in the case of peak demand, and rapidly released (scale in), to keep up with the demand. Also, more virtual *resources* (on a single physical machine) can be provisioned when the work load increases (scale up) and released when the work load decreases (scale down). These scaling methods can automatically be done according to the user's predefined conditions (Auto Scaling).
Broad network access	Services are available over the network and accessed through standard mechanisms that can be achieved by heterogeneous thin or thick client platforms such as mobile phones, laptops and PDAs (Mell & Grance, 2009).
Resource pooling	The Cloud provider offers a pool of computing resources to serve multiple consumers using a multi-tenant model, with different physical and virtual resources. Location transparency in the Cloud that hides resource's physical location from the customer (but provider may offer location at a higher level of abstraction, like country) provides more flexibility to Cloud providers for managing their own resource pool.
Business model	Cloud computing is mainly supported by gigantic IT companies. They plan that all investments on Cloud computing should get return on investment (ROI) in the near future (Gong, Liu, Zhang, Chen, & Gong, 2010). Still, there are many business models that can be used in Cloud computing, Cloud providers often employ a pay-per-use service-driven model.
Measured service	Cloud computing is based on the concept of utility computing which provides computing services through an on-demand, pay-per-use billing method. Cloud systems can transparently monitor, control and measure service usages for both the provider and the consumer by leveraging a metering capability at some level of abstraction appropriate to the type of service, similar to what is being done for utilities such as Electricity, Gas, Water, Telecommunication, etc.
Customization	Cloud computing allows users to deploy specialized virtual appliances and to be given privileged (root) access to the virtual servers in order to consider the great disparity between user needs in a multi-tenant environment (Buyya, Broberg, & Goscinski, 2011).

Table 1. Cloud computing characteristics.

the cost of electricity, system admins, hardware engineers, network engineers, facilities management, fire protection, and insurance or local and state taxes on facilities. There are other hidden OpEx costs that a Cloud instance can eliminate, such as purchasing and acquisition overhead, asset insurance, and business interruption planning and software (Joyent, 2012).

The development of Cloud computing technology faced many critical obstacles such as security, availability, accountability, confidentiality, privacy, data provenance, data lock-in, Cloud interoperability, lack of standardization,

SLA issues, customization, performance unpredictability, technology bottlenecks, etc., which are addressed in many researches (Vouk, 2008; Armbrust, et al., 2009; Dikaiakos, Katsaros, Mehra, Pallis, & Vakali, 2009; Zhang, Cheng, & Boutaba, 2010; Jansen, 2011; Hu, et al., 2011; Akande, April, & Van Belle, 2013; Xiao & Xiao, 2013).

At least three delivery models exist on the Cloud: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) deployed as public, private, community, and hybrid Clouds. These service types and boundaries are elaborated in the following sections.

2.2. Delivery Models

Consumers purchase Cloud services in the form of infrastructure, platform, or software.

Infrastructure services are considered to be the bottom layer of Cloud computing systems (Nurmi, et al., 2009). Infrastructure as a Service (IaaS) offers virtualized resources (such as computation, storage, and communication) on-demand to the infrastructure specialists (IaaS consumers) who are able to deploy and run arbitrary operating systems and customized applications. IaaS Cloud providers often provide virtual machines (VMs) with a software stack that allows to make them customized, similar to physical servers. They grant privileges to users for doing some operations on their virtual server (such as starting and stopping it). Therefore, an infrastructure specialist does not manage or control the underlying Cloud infrastructure while having control over operating systems, storage, deployed applications, and possibly limited control of some networking components, e.g. host firewalls (Marinescu, 2013). This type of service is particularly useful for start-ups, small and medium businesses (SMBs) with rapidly expanding or dynamic changes, that do not want to invest in infrastructure. IaaS providers focus on availability guarantees, specifying the minimum percentage of time the system will be available during a certain period in terms of SLA (Buyya, Broberg, & Goscinski, 2011). The SLA for the IaaS is the most complex as the infrastructure specialists have control over the virtual machines (Myerson, 2013).

Another service in the Cloud that offers a higher level of abstraction to make a Cloud easily programmable is known as Platform as a Service (PaaS) (Buyya, Broberg, & Goscinski, 2011). PaaS Cloud providers provide a scalable platform with a software stack containing all tools and programming languages supported by the provider. They allow developers (PaaS consumers) to create and deploy applications without the hassle of managing infrastructure, and regardless of the concerns about processors and memory capacity. Therefore, the developer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems or storage while having control over the deployed applications and possibly application-hosting environment configurations (Mell & Grance, 2009). PaaS is not useful for portable applications, applications written in a special programming language, or applications that need specific hardware or software to run. Besides the infrastructure availability, PaaS providers try to guarantee the accessibility and accuracy of their platform in terms of SLA. The SLA for the PaaS is not as complex as the SLA for the IaaS in which the developers have control over the application development life cycle, but not over the virtual machines.

Delivering applications supplied by service providers at the highest level of abstraction in the Cloud to the end-users (SaaS consumers) through a thin client interface such as a web portal is known as Software as a Service (SaaS). SaaS Cloud providers supply a software stack containing an operating system, middlewares such as database or web servers, and an instance of the Cloud application, all in a virtual machine. Therefore, the end-user 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 (Marinescu, 2013).

SaaS alleviates the burden of software maintenance for customers and simplifies development and testing for providers (Buyya, Broberg, & Goscinski, 2011). SaaS providers try to guarantee end-user access to the latest update of the SaaS application on a twenty-four hour, seven days a week (24x7) basis at a high rate and the number of end-users that can be served simultaneously in terms of SLA. The SLA for the

SaaS is the simplest as the end-users only have access to the SaaS application. These three delivery models are demonstrated in Figure 3.

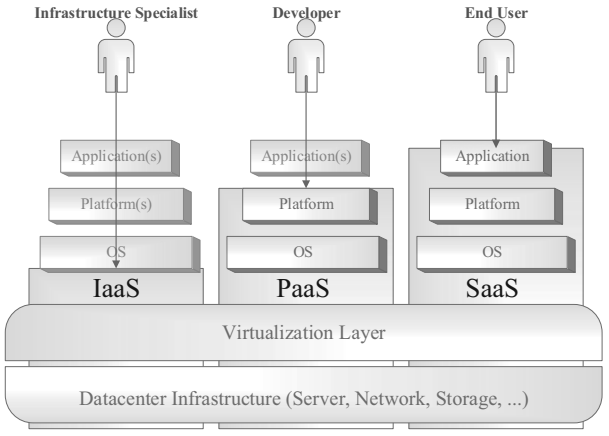

Figure 3. Infrastructure as a Service, Platform as a Service, Software as a Service.

Amazon EC2, Google App Engine and Salesforce.com are three main use cases of cloud service models that offer IaaS, PaaS and SaaS respectively.

2.3. Deployment Models

There are four general Cloud deployment models known as private, public, community, and hybrid Cloud.

In Private Cloud, the infrastructure is owned and exclusively used by a single organization, and managed by the organization or a third-party and may exist on or off the premises of the organization. Many organizations, particularly governmental or very large organizations, embraced this model to exploit the Cloud benefits like flexibility, reliability, cost reduction, sustainability, elasticity, and so on. However, they are often criticized for being similar to traditional proprietary server farms and for not providing benefits such as no up-front capital costs (Zhang, Cheng, & Boutaba, 2010).

In Community Cloud, the infrastructure is shared by several organizations and supports a specific community with shared concerns such as mission, security requirements, policy, and compliance considerations (Mell & Grance, 2009). It may be owned and managed by the organizations or by a third-party and may exist on-premises or off-premises.

In Public Cloud, the infrastructure exists on the premises of the Cloud provider and is available to the general public or a large industry group, and is owned by an organization selling Cloud services. There are three major Public Cloud computing styles based on the underlying resource abstraction technologies, including Amazon EC2, Google App Engine and Windows Azure (Qian, Luo, Du, & Guo, 2009). However, a public Cloud can use the same hardware infrastructure (with large scale) as a Private one (Marinescu, 2013). In contrast with Private, Public Cloud lacks fine-grained control over data, network and security settings, which hampers their effectiveness in many business scenarios (Zhang, Cheng, & Boutaba, 2010). This model is suitable for small and medium businesses (SMBs) to support their growing business without huge investment in the infrastructure.

Sometimes the best infrastructure to fit an organization's specific needs requires both Cloud and on-premises environments. In Hybrid Cloud, the services within the same organisation are a composition of two or more Clouds (Private, Community, or Public) to address the limitations of each model with more scalability and flexibility whilst both saving money and providing additional security. On the down side, Hybrid Cloud environments involve complex management and governance challenges.

Some other deployment models, such as Virtual Private Cloud and Managed Cloud are well known but not widely used. Virtual Private Cloud is a Private Cloud that leverages a Public Cloud infrastructure using advanced network capabilities (such as VPN) in an isolated and secured manner. Managed Cloud is a type of private Cloud that is managed by a team of experts in a third-party company. Managed Cloud includes access to a dedicated, $24 \times 7 \times 365$ support team via phone, chat, online support, and so on to support Cloud servers from the OS up through the application stack. Amazon VPC (Amazon VPC, n.d.) and Rackspace Managed Cloud (Rackspace, n.d.) are examples of Virtual Private Cloud and Managed Cloud, respectively. There are also Managed Virtual Private

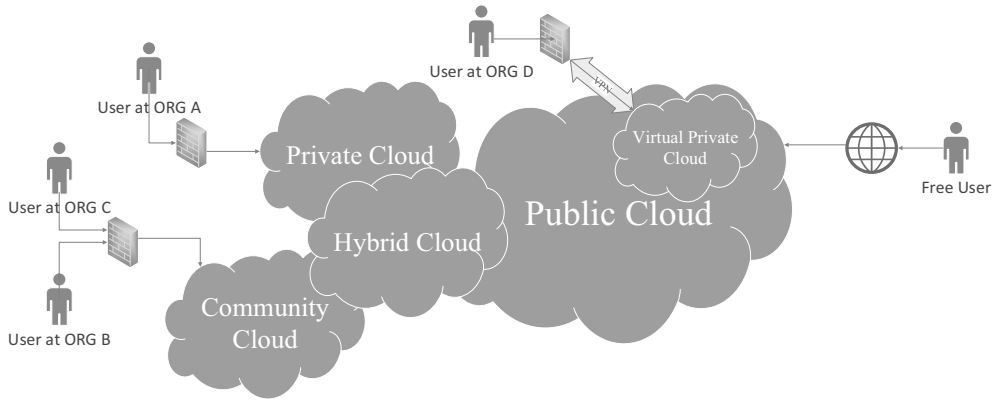

Figure 4. Cloud computing deployment models.

Cloud such as HP Helion Managed Virtual Private Cloud (HP Helion Managed Virtual Private Cloud, n.d.).

Figure 4 illustrates the above mentioned deployment models.

3. Jungle Computing

3.1. Overview

Jungle computing is a simultaneous combination of several distributed and high performance computing systems to achieve peak performance as well as reduce programming complexity. Jungle computing system is highly heterogeneous. It may include clusters, grids, clouds, supercomputers, and even mobile devices, possibly with accelerators such as GPUs and FPGAs (Drost, et al., 2012).

Traditional distributed systems are more equipped with state-of-the-art many-core technologies such as multi-core processors, processor clusters, GPUs, as well as supercomputers on-chip. Although these devices often provide orders of magnitude speed improvements, they make computing platforms more heterogeneous and hierarchical and vastly more complex to program and use (Seinstra, et al., 2011). Further complexities because of urgent desire for scalability and several issues like data distribution, software heterogeneity, and ad-hoc hardware availability force scientists into concurrent use of multiple platforms (Seinstra, et al., 2011). These platforms may use different middle-wares, programming interfaces, access policies, and protection mechanisms. Furthermore, the diverse computing paradigms differ in usage model. As an example, a stand-alone machine is usually permanently available, a Grid resource will have to be reserved, whilst a Cloud requires a credit card to gain access (Drost, et al., 2012).

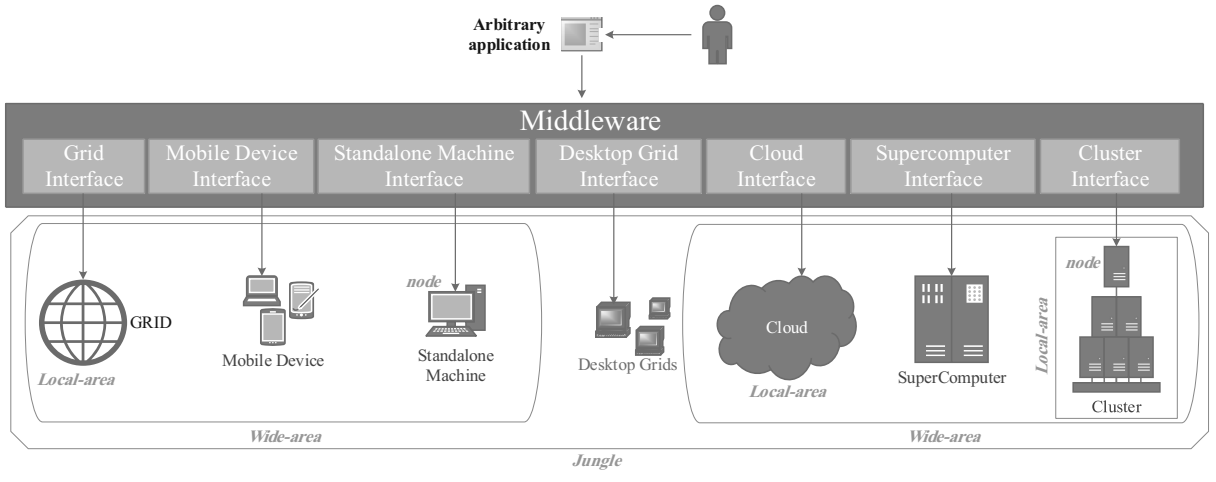

Figure 5. Jungle computing architecture.

This new distributed computing model makes available a diverse collection of resources to research scientists. These resources can be independent computers, supercomputers, cluster systems, grids, desktop grids, clouds, etc., which are all connected via fast networks. By looking at such systems from a high-level perspective, it can be seen that all these systems consist of a number of basic computing nodes. Each node has local memories and processors, and is capable of communicating over a local-area (multi-node) or wide-area (multi-local-area) connection and create a Jungle (multi-wide-area), as is shown in Figure 5.

Jungle computing systems, in the worst case, include all computing platforms depicted in Figure 5. However, in practice, every possible subset of this figure represents a realistic scenario for Jungle computing system. Hence, if there are fundamental methodologies working in the worst case scenario, it can make sure that the solution applies to all possible scenarios.

For each used resource in the Jungle, the arbitrary application may have to be re-compiled, or even partially re-written, to handle the changes in available software and hardware (Drost, et al., 2012). Moreover, to run any application, a different middleware interface for each resource should be available.

Once an application has been successfully started in a Jungle, another aspect that hinders the use of Jungle computing systems is the lack of connectivity between resources. Resources, especially clusters and supercomputers, are usually not designed for communication with the outside world. Traditional tightly coupled HPC tools are not particularly suited for distributed, heterogeneous and hierarchical environments. Therefore, a Jungle computing platform overcomes these problems in order to simultaneously and transparently use all of the available computing powers. Consequently, Jungle must have at least the following characteristics (Seinstra, et al., 2011) which are listed in Table 2.

Characteristic	**Comment**
Resource independence	Resources differ considerably in their details such as processor architecture, amount of memory and performance. This feature hides the physical characteristics of resources from the applications or end-users interacting with those resources.
Middleware independence and interoperability	Middleware independence allows a Jungle to embrace new middleware technologies with minimal impact on their applications, thereby it eliminates the need for implementing a different interface for each middleware and provides interoperability between middlewares.
Robust connectivity and globally unique resource naming	This feature is needed to remove connectivity problems such as communication problems with firewalls, transparent renaming of IP addresses, and multi-homing (machines with multiple addresses). Resource names must be globally unique within a Jungle so that when a user or application on the Jungle specifies a resource name, the connection is routed to that resource. These resources can have the same name in their local system.
Malleability	Jungle with malleability will be able to adapt to resource changes. Malleability provides tolerance to the variations in resources availability, e.g., reservations ending, in order to correctly handle joining and leaving.
System-level fault-tolerance	This feature ensures that the Jungle computing system can be used by the end-user, even when something goes wrong in one or more elements in the hardware configuration. In case of a failure, the failing resource has to be replaced by a backup system.
Application-level fault-tolerance	This feature can complement system-level fault-tolerance by restoring the state of the applications that ran on a failed resource.
Parallelization	This feature relieves programmers from the tedious and error-prone manual parallelization process and makes programming models available to end-users. It hides all or most of the inherent complexities of parallelization by converting automatically sequential program into parallel program utilizing all available resources.
Integration with external software	This feature is needed for integrating black box legacy codes such as system-level software (e.g., specialized communication libraries) and architecture-specific compute kernels (e.g., CUDA-implemented algorithms for GPU-execution) with a software system for Jungle computing.

Table 2. Jungle computing characteristics.

Computational astrophysics, climate modelling (Drost, et al., 2012), remote sensing (Kessel, et al., 2014), image analysis, drug designing, astronomy or Neuroinformatics (Seinstra, et al., 2011) are some examples of the applications that can benefit from Jungle computing systems.

3.2. Computing in the Jungle

In this section, we briefly describe two software platforms designed to support Jungle computing. First, the Ibis high-performance distributed programming system (Seinstra, et al., 2011), and next the Constellation (Maassen, Drost, Bal, & Seinstra, 2011) that is a platform for creating flexible and efficient applications for Jungle computing systems, will be described.

3.2.1. Ibis

Ibis (Ibis: Computing in the Jungle, n.d.), is an open source Java distributed computing software platform with an integrated, layered architecture that offers an efficient and transparent solution for Jungle computing. The aim of Ibis is to create an efficient Java-based platform for distributed computing that combines all of the fundamental methodologies into a single integrated programming system to simplify the programming and deployment of Jungle Computing. Ibis provides high-level, architecture- and middleware-independent interfaces that allow transparent implementation of efficient applications. These features provide robustness and dynamic variations in the availability of resources (Seinstra, et al., 2011).

The Ibis platform consists of two distinct subsystems, which provide all functionality that is traditionally associated with programming languages and communication libraries, as well as operating systems. The former is Ibis Distributed Deployment System which allows users to easily start and deploy any application in the Jungle, while the latter is Ibis High Performance Programming System which allows programmers to write applications specifically designed to run in a Jungle computing system. An overview of the Ibis architecture is depicted in Figure 6, whilst detailed architecture can be seen in (Bal, et al., 2010).

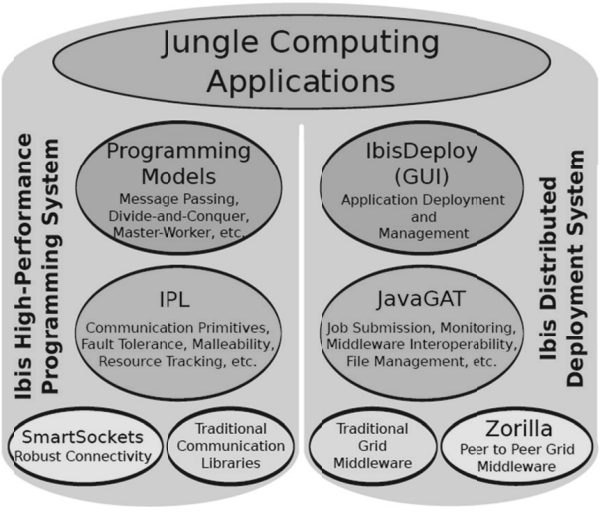

Figure 6. Overview of the Ibis architecture (Drost, et al., 2012).

Ibis is designed in a modular way and consists of a number of sub-projects, which are possible to be used individually. Each sub-project solves a specific problem. Some of them are showed in Table 3.

Although Ibis is written in Java, it interfaces easily with the existing software and supports running any application written in any programming language such as C, C++, FORTRAN, MPI, or otherwise (Drost, et al., 2012).

The Ibis platform meets the characteristics of Jungle computing (Seinstra, et al., 2011) as presented in Table 4.

3.2.2. Constellation

Ibis/Constellation, or in brief Constellation, is a lightweight software platform specifically designed for distributed, heterogeneous and hierarchical computing environments. As addition to the Ibis project, it enables applications to efficiently run on Jungle computing systems. In addition, Constellation makes it straightforward to re-targeting applications to completely different computing environments (Maassen, Drost, Bal, & Seinstra, 2011).

In Constellation, applications consist of multiple distinct, loosely coupled *activities* that communicate using *events*, with certain dependencies between them. Each *activity* represents a distinct action that needs to be performed by the application. Multiple implementations of

Sub-system	Sub-project	Problem to solve	Comment
Ibis Distributed Deployment System	Zorilla	Building a (desktop) grid quickly with as little effort as possible	Zorilla is a lightweight, easy-to-use and fully P2P middleware, with no central components (to hinder scalability or fault-tolerance), that creates a single distributed environment from any available set of computer resources on the systems ranging from clusters and desktop grids, to grids and clouds (N. Drost, 2011).
	JavaGAT	Confusing with complicated middleware such as Globus 2, SSH, gLite, Globus 4, etc.	Java Grid Application Toolkit is a generic and simple interface that provides a uniform interface. It sits between applications and numerous types of middleware, such as Globus, gLite, SSH or Zorilla, and offers a single and consistent system that dynamically forwards application calls on the JavaGAT API to one or more middlewares that implement the requested functionality (Nieuwpoort, Kielmann, & Bal, 2007).
	IbisDeploy	Thinking in terms of middleware operations	The IbisDeploy is a library for deploying applications in the Jungle by end-users that allows the user to manually load resources and applications, to add new resources to a running application and to pause and resume applications at any time.
Ibis High Performance Programming System	Smart-Sockets	Communication through firewalls, NATs, non-routed networks, etc.	SmartSockets, a communication library, offers a single integrated solution that automatically discovers a wide range of connectivity problems and attempts to solve them with little or no support from the user, behind a simple interface that closely resembles sockets (Maassen & Bal, 2007).
	IPL	Fault-tolerance, portability, streaming, malleability and multi-threading for customized Jungle applications	The Ibis Portability layer, a low level message-based communication library, is the interface between Ibis implementations for different architectures and the runtime systems that provide programming models suitable for writing applications, specifically for the use in a Jungle, to support fault-tolerance and malleability.
	Ibis RMI	Performance problems in RMI applications	Ibis RMI is an efficient re-implementation of Remote Method Invocation (RMI), which is an object-oriented form of Remote Procedure Call (RPC).
	Satin	Difficulty in writing parallel applications	It is a transparent, high-level and efficient divide-and-conquer programming model that recursively splits a program into sub-tasks and then waits until the sub-tasks are completed. Using the Satin model, parallelization is obtained automatically for divide-and-conquer applications.
	MPJ/Ibis	Using customized MPI	It is an MPI version implemented completely in Java, and so is platform independent for using customized MPI.
	JEL	Dealing with real-world Jungle computing systems, in which resources can crash and can be added or deleted	Join-Elect-Leave (JEL), a unified model for tracking resources in dynamic and distributed environments, is based on the concept of signalling (i.e., notifying the application when resources have Joined or Left the computation) and Elections (i.e. selecting resources with a special role). It names resources globally unique and keeps a track of the available resources at runtime.

Table 3. Ibis sub-projects.

an *activity* may be created to support different hardware architectures and they may consist of a script, C, CUDA, OpenMP or MPI, etc. Constellation uses *Executors* to represent hardware capable of running activities. An *executor* may represent a single core of a machine, a single machine with multiple cores, a specialized piece of hardware (e.g., a GPU), an entire Cluster, and so on (Maassen, Drost, Bal, & Seinstra, 2011). It is very easy to exploit new resources by deploying extra *executors* on those new resources. Figure 7 shows an abstract of the Constellation architecture.

This approach to application development vastly reduces the programming complexity (Maassen, Drost, Bal, & Seinstra, 2011), which is the aim of Jungle computing. Instead of creating a single application capable of running in a complex

Characteristic of Jungle Computing	**Ibis**
Resource independence	Since Ibis is a Java-based platform, JVM Virtualization provides resource independence.
Middleware independence and interoperability	JavaGAT provides this functionality using adapters which interact with a middleware to start the required task. JavaGAT will automatically select the appropriate adapter for each resource, and adapters are easily added, if needed.
Robust connectivity and globally unique resource naming	These features are supported by the SmartSockets and the JEL, respectively.
Malleability	These features provided by the Satin, the IPL and resource tracking mechanisms of the JEL, which is an integral part of the IPL. Zorilla also supports fault tolerance and malleability by implementing all functionality using peer-to-peer techniques.
System-level fault-tolerance	
Application-level fault-tolerance	Application-level fault-tolerance can be built on top of the system-level fault-tolerance mechanisms provided by the IPL.
Parallelization	This requirement is fulfilled through a number of programming models such as Satin model implemented on top of the IPL.
Integration with external software	This is achieved through JNI (Java Native Interface) or through adaptor interfaces (plugins).

Table 4. Ibis vs. Jungle characteristics.

distributed and heterogeneous environment, it is easier and often faster to just create several independent *activities* targeted at smaller and simpler homogeneous environments.

Using equi-kernels, which are different implementations of the same kernel functionality, integration with external software is provided while maintaining resource independence. By including a default equi-kernel for each operation, the Constellation can transparently exploit special hardware and codes, without failing to operate when such hardware is not available (Kessel, et al., 2014). Furthermore, the matchmaking mechanism and labelling approach of Constellation makes it very simple to configure. Matchmaking (Raman, Livny, & Solomon, 1998) is one of the main problems in heterogeneous applications that gives the right activities to the right resource while Labelling allows an application to tag different types of activities and hardware (Maassen, Drost, Bal, & Seinstra, 2011).

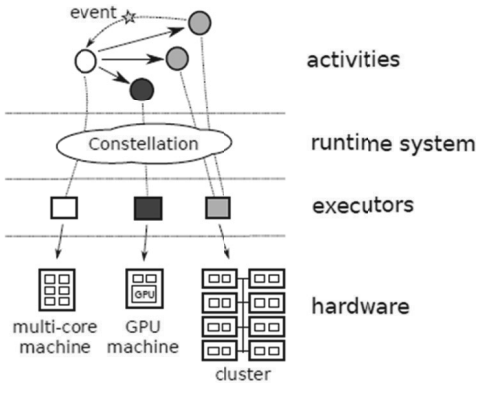

Figure 7. An abstract of the Constellation architecture (Maassen, Drost, Bal, & Seinstra, 2011).

At last, Constellation meets the characteristics of Jungle computing by leveraging Ibis platform that fully provides middleware independence, robust connectivity, and system-level fault-tolerance, while it offers mechanisms to support for malleability and application-level fault-tolerance (Kessel, et al., 2014).

4. Fog Computing

4.1. Overview

The term Fog computing has been embraced by Cisco Systems as a new paradigm (Bonomi, Milito, Zhu, & Addepalli, 2012). Fog computing is a systematic, highly virtual, secure, and network-integrated platform that provides computing, storage, and networking services between end points and traditional Cloud computing data centers (Bonomi, Milito, Zhu, & Addepalli, 2012). It is a model in which data, processing and applications are concentrated in devices at the network edge, rather than existing almost entirely in the Cloud, to isolate them

from the Cloud systems and place them closer to the end-user, which is the aim of Fog computing.

The Fog is organizationally located below the Cloud and serves as an optimized transfer medium for services and data within the Cloud, which is depicted in Figure 8.

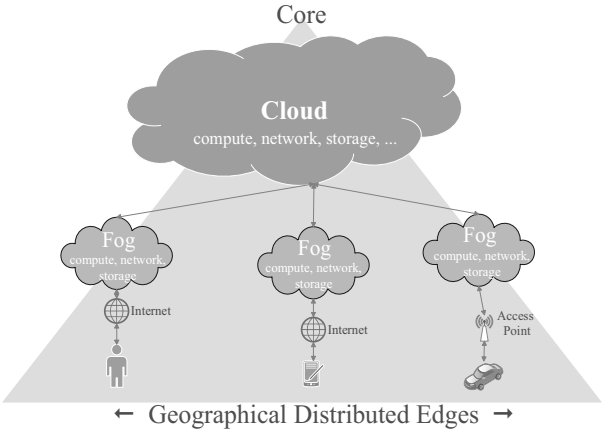

Figure 8. Fog computing architecture.

The Fog computing happens outside the Cloud and ensures that Cloud services, compute, storage, workloads, applications and big data can be provided at any edge of the network (Internet) in a truly distributed way. By controlling data in various edge points, Fog computing integrates core Cloud services to turning data center into a distributed Cloud platform for users. In other words, FOG brings computing *f*rom the c*o*re to the ed*g*e of the network (*fog*).

In this context, it may be just another name for Edge computing (Bonomi, The Smart and Connected Vehicle and the Internet of Things, 2013). Edge Computing is pushing the frontier of computing applications, data, and services away from centralized nodes to the logical extremes of a network. It enables analytics and knowledge generation to occur at the source of the data. This approach requires leveraging resources that may not be continuously connected to a network such as laptops, smartphones, tablets and sensors (LaMothe, 2013). Two systems that can provide resources for computing near the edge of the network are the MediaBroker (Lillethun, Hilley, Horrigan, & Ramachandran, 2007) for live sensor stream analysis, and Cloudlets (Satyanarayanan, Bahl, Caceres, & Davies, 2009) for interactive applications. However, neither currently supports widely distributed geospatial applications (Hong, Lillethun, Ramachandran, Ottenwälder, & Koldehofe, 2013).

Users are clamoring for access to the massive quantities of information at any time, in any place and with any device that is extremely dispersed and produced by and about people, things, and their interactions. The flexibility of the Cloud makes it a good choice for this need. As the technology advances, the question for many businesses is how they can benefit from big data and how to use Cloud computing to make it happen. In order to make an effective and optimized Cloud model, businesses require a new approach to crunching huge quantities of data and delivering them to their users via geographically distributed platforms and not via the Cloud which is located in one place. Thus, the idea of Fog computing has emerged to distribute all data and place it closer to the end-user, eliminate service latency, improve QoS and remove other possible obstacles connected with data transfer. Because of its wide geographical distribution, the Fog paradigm is well positioned for big data and real time analytics and it supports mobile computing and data streaming.

Fog computing is not a replacement for Cloud computing. It is just an addition which develops the concept of Cloud services. Services are hosted at the network edge or even end devices such as set-top-boxes or access points, etc. Conceptual Fog computing builds upon existing and common technologies like Content Delivery Networks (CDN), but based on Cloud technologies it should ensure the delivery of more complex services (Buest, 2013; Kleyman, 2013). However, developing applications using fog computing resources is more complex (Hong, Lillethun, Ramachandran, Ottenwälder, & Koldehofe, 2013).

A number of characteristics that make the Fog computing a non-trivial extension of the Cloud computing are listed in Table 5.

Characteristic	Comment
Proximity of data to end-users	Services would be located closer to end-user to improve latency concerns and data access. Instead of storing information in centralized data center sites far away from end-user, the Fog computing ensures direct proximity of the data to the customer.
Hierarchical organization	To support low-latency and scalability, the Fog computing platform follows a multi-tier architecture from the core to the edges. Also, the control and management are hierarchical to support interplay with the Cloud.
Edge location, location awareness, low latency	The Cloud is too far from many mobile users for latency-sensitive applications. Fog computing extends existing Cloud services by spanning up an edge network which consists of many distributed endpoints (Buest, 2013). Fog collectors at the edge ingest the data generated by sensors and devices. Fog nodes provide localization, therefore enabling low latency and context awareness (Bonomi, Milito, Zhu, & Addepalli, 2012).
Dense geographical distribution	In sharp contrast to the more centralized Cloud, the services and applications targeted by the Fog demand widely distributed deployments (Bonomi, Milito, Zhu, & Addepalli, 2012). This feature includes a faster elicitation and analysis of big data, a better support for location-based services (by the entire WAN links can be better bridged) as well as the capabilities to evaluate data massively scalable in real-time (Buest, 2013).
Large-scale sensor networks	Fog uses sensor networks in a large scale to monitor the environment. The Fog computing makes possible not only Internet of Things (IoT) development, but also Ubiquitous Computing (UC) approaches, by extending cloud computing services to include smart sensors and intelligent devices (Madsen, Albeanu, Burtschy, & Popentiu-Vladicescu, 2013).
Large number of nodes	Fog computing involves a very large number of nodes as a consequence of the wide geo-distribution, as evidenced in sensor networks in general (Bonomi, Milito, Zhu, & Addepalli, 2012).
Support for mobility	Administrators are able to control where users are coming in and how they access the information and support location-based mobility demands and don't have to traverse the entire WAN (Kleyman, 2013). This improves user performance, quality of service (Rudenko, 2013), security and privacy issues.
Real-time interactions	In the Cloud, edge devices must communicate across the Internet to reach the cloud data centers. This causes WAN latencies that can be high and interfere with interactive applications. In contrast, in the Fog, servers belong to the same network as the end-users. Therefore, Fog includes interactive applications rather than batch processing and so, real-time data analytics become more prevalent on Fog computing.
Predominance of wireless access	Fog is drop-based (Rudenko, 2013) with a wireless connection and extremely low power to support mobility, scalability, etc. in a really distributed network. Smart connectivity of intelligent devices supporting wireless communication with existing networks and participating to computational tasks using network resources is an important objective within IoT vision (Madsen, Albeanu, Burtschy, & Popentiu-Vladicescu, 2013).
Heterogeneity	Fog nodes come in different form factors and will be deployed in a wide variety of environments (Bonomi, Milito, Zhu, & Addepalli, 2012). These resources are highly dynamic and heterogeneous at different levels of network hierarchy to support low latency and scalability requirements of applications (Hong, Lillethun, Ramachandran, Ottenwälder, & Koldehofe, 2013).
Dynamic per user optimization	Unlike the Cloud that is separated from the user by wide area networks (WAN), a Fog server has the distinct advantage of knowing the network conditions local to an end-user. This is because the Fog server belongs to the same network as the end-users. So the Fog server can have knowledge of each user. Knowledge of the user's behaviour can help dynamically select the best parameters. Specifically, the Fog server can use this information to customize the optimization (Zhu, et al., 2013).
Interoperability and federation	Fog components are able to interoperate with different providers seamless support of certain services (such as streaming) that are federated across domains (Bonomi, Milito, Zhu, & Addepalli, 2012).
Integration with the Cloud and support for on-line analytic	Cloud provides global centralization, whilst Fog provides localization. Many applications require both Fog localization and Cloud globalization, particularly for analytics and Big Data (Bonomi, Milito, Zhu, & Addepalli, 2012). Fog platform supports real-time, actionable analytics, processes and filters the data, and pushes to the Cloud that is global in geographical scope and time (Cisco, 2013). With Fog services, it is able to enhance the Cloud experience by isolating user data that needs to live on the edge. From there, administrators are able to tie-in analytics, security or other services directly into their Cloud model (Kleyman, 2013).

Table 5. Fog computing characteristics.

Already, we are seeing everything-as-a-service models. This means that the future of the computing paradigms must support the idea of the Internet of Thing (IoT) in order to successfully emerge, wherein sensors and actuators blend seamlessly with the environment around us and the information is shared across platforms in order to develop a common operating picture (Gubbi, Buyya, Marusic, & Palaniswami, 2013). Fog computing supports emerging Internet of Thing applications that demand real-time or predictable latency such as industrial automation, transportation, sensor networks and actuators.

This model is called Fog computing simply because the Fog is a Cloud close to the ground and Fog promises to take hardware and software virtualization back down from cloud to earth, where it belongs. Therefore, the differences between the Fog and the Cloud can be summarized in "The Ground" term (Nemirovsky, Milito, & Yanuzzi, 2012), which is described in Table 6.

The basic technology for Fog computing is the concept of drop (Rudenko, 2013). A drop is a chip of a microcontroller with built-in memory and data transfer interface, which is combined with wireless connection Mesh chip. Such a drop works on a small battery which is enough for a couple of years. Users can connect different temperature, light, voltage sensors, etc.

With the help of such mini-chips it is possible to create really distributed network of data or devices all around the planet. Constant data circulation in the world makes providers create new technologies for their local storage and caching. The drops allow keeping the data close to a user instead of storing them in a data center somewhere far away. It helps to avoid possible delays in data transfer (Rudenko, 2013).

The concept of Fog computing is not something to be developed in the future. It is already here and a number of distributed computing and storage start-ups are adopting the phrase (Kleyman, 2013). A lot of companies have already introduced it, while other companies are ready for it (Rudenko, 2013). Actually, any company which delivers content can start using Fog computing. A good example is Netflix (NetFlix, n.d.), a provider of media content, who is able to reach its numerous globally distributed customers. With the data management in one or two central data centers, the delivery of video-on-demand service would not be efficient enough. Fog computing thus allows providing very large amounts of streamed data by delivering the data directly into the vicinity of the customer (Buest, 2013). Another use case is Symform (Symform, n.d.), a Cloud storage provider, which uses a decentralized, distributed, virtual, and crowd-sourced Cloud. Its approach can provide better disaster resilience than data centers hundreds of miles apart. It can do that in a way that is extremely cheap, and in some cases free (Perry, 2013). Smart Grid is another rich Fog use case (Bonomi, Milito, Zhu, & Addepalli, 2012). It is the next generation of computing and provides tremendous value to providers and users.

Letter	Comment
T	**T**iered organization involving multiple administrations in the access
H	**H**ierarchical control and management supporting interplay with the Cloud
E	**E**xpanded mobility model
G	**G**eo-distribution of computational power with strong focus on service locality
R	**R**eal-time analytics at different tiers
O	**O**rchestration involving coordinated control and actuation in multi-tier settings
U	**U**nified exposure of virtualized resources (consolidated Virtualization)
N	**N**egligible latency
D	**D**istributed policy exposure and policy management involving multiple tiers

Table 6. Differences between the Fog and the Cloud.

5. Conclusion

The future of computing is heading toward using shared heterogeneous resources and is concerned about Big Data. These requirements result in emerging new distributed computing paradigms. In this article, we have strived to clarify modern distributed computing paradigms, namely Cloud, Jungle and Fog computing.

In Cloud computing, resources are moving away from end-users towards centralized systems that possess huge processing power and storage capacities. In this model, problems arise for latency-sensitive applications, which require nodes in the vicinity to meet their delay requirements and users have no control over the manner in which they can access their data. Fog computing, in contrast, at a really distributed level, provides computing services between end points and traditional Cloud computing data centers, away from centralized nodes to the edge of a network. On the other hand, distributed computing infrastructures such as Cloud, Cluster or Grid currently are undergoing revolutionary change and becoming more heterogeneous and hierarchical. This leads scientists to use a simultaneous combination of heterogeneous, hierarchical, and distributed computing resources as a Jungle computing system, to access more computing power.

Cloud computing has got the momentum in the distributed computing in recent years and may be mature with new technologies and paradigms such as Jungle and Fog. It is obvious that Cloud computing is used in Fog computing and may or should be used in Jungle computing. Therefore, bear in mind that knowing Cloud computing is essential in distributed computing. On the other hand, since Cloud computing may not be a mainstream technology in the near future, as Grid computing is no longer a concept to be discussed, it is useful to take a glance at the other, newly distributed computing paradigms.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] A. O. AKANDE, N. A. APRIL, J.-P. VAN BELLE, Management Issues with Cloud Computing. *Second International Conference on Innovative Computing and Cloud Computing*, (2013) New York, NY, USA: ACM, pp. 119–124.

[2] AMAZON VPC, (n.d.). Retrieved from Amazon: `http://aws.amazon.com/vpc/`

[3] G. R. ANDREWS, *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley, 1999.

[4] N. ANTONOPOULOS, L. GILLAM, *Cloud Computing: Principles, Systems and Applications*. Springer, 2010.

[5] M. ARMBRUST, A. FOX, R. GRIFFITH, A. D. JOSEPH, R. H. KATZ, A. KONWINSKI, M. ZAHARIA, *Above the Clouds: A Berkeley View of Cloud Computing*. EECS Department, University of California, Berkeley, 2009.

[6] H. E. BAL, J. MAASSEN, R. V. NIEUWPOORT, N. DROST, N. PALMER, G. WRZESINSKA, C. JA, Real-world distributed computing with Ibis. *IEEE Computer*, (2010), 54–62.

[7] F. BONOMI, The Smart and Connected Vehicle and the Internet of Things. *Workshop on Synchronization in Telecommunication Systems (WSTS)*, (2013) San Jose, California, USA.

[8] F. BONOMI, R. MILITO, J. ZHU, S. ADDEPALLI, Fog Computing and Its Role in the Internet of Things. *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, (2012), pp. 13–16.

[9] R. BUEST, *Fog Computing: Data, Information, Application and Services need to be delivered more efficiently to the end user*. (2013, October 19). Retrieved from clouduser.de: `http://clouduser.de/en/analysis/fog-computing-data-information-application-and-services-needs-to-be-delivered-more-efficient-to-the-enduser-22362`

[10] R. BUYYA, J. BROBERG, A. GOSCINSKI, *CLOUD COMPUTING Principles and Paradigms*, (1 ed.). Wiley, 2011.

[11] CISCO. *Fog Computing, Ecosystem, Architecture and Applications*. (2013). Retrieved from `http://www.cisco.com/web/about/ac50/ac207/crc_new/university/RFP/rfp13078.html`

[12] CSA, *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1.*. Cloud Security Alliance, 2009.

[13] M. D. DIKAIAKOS, D. KATSAROS, P. MEHRA, G. PALLIS, A. VAKALI, Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing*, (2009), 10–13.

[14] GARTNER, INC., Gartner's 2013 Hype Cycle for Emerging Technologies Maps Out Evolving Relationship Between Humans and Machines, (2013) Stamford, Connecticut, USA: Gartner.

[15] N. DROST, J. MAASSEN, M. A. MEERSBERGEN, H. E. BAL, F. I. PELUPESSY, S. P. ZWART, F. J. SEINSTRA, High-performance Distributed Multi-model/Multi-kernel Simulations: A Case-study in Jungle Computing. *IEEE International Parallel and Distributed Processing Symposium Workshops*, (2012) Shanghai, China: IEEE, pp. 150–162.

[16] W. FORREST, *Clearing the Air on Cloud Computing*. McKinsey & Company, 2009.

[17] C. GONG, J. LIU, Q. ZHANG, H. CHEN, Z. GONG, The Characteristics of Cloud Computing. *39th International Conference on Parallel Processing Workshops*, (2010) San Diego, CA: IEEE, pp. 275–279.

[18] J. GUBBI, R. BUYYA, S. MARUSIC, M. PALANISWAMI, Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, **29**(7) (2013), 1645–1660.

[19] B. HALPERT, *Auditing Cloud Computing: A Security and Privacy Guide*. John Wiley & Sons, 2011.

[20] B. HAYES, Cloud Computing. *Communications of the ACM*, (2008), 9–11.

[21] K. HONG, D. LILLETHUN, U. RAMACHANDRAN, B. OTTENWÄLDER, B. KOLDEHOFE, Mobile Fog: A Programming Model for Large–Scale Applications on the Internet of Things. *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, (2013) New York, NY, USA: ACM, pp. 15–20.

[22] HP HELION MANAGED VIRTUAL PRIVATE CLOUD, (n.d.). Retrieved from HP: `http://www8.hp.com/us/en/business-services/it-services.html?compURI=1079571`

[23] F. HU, M. QIU, J. LI, T. GRANT, D. TYLOR, S. MCCALEB, R. HAMNER, A Review on Cloud Computing: Design Challenges in Architecture and Security. *Journal of Computing and Information Technology*, (2011), 25–55.

[24] IBIS: COMPUTING IN THE JUNGLE, (n.d.). Retrieved from VU University Amsterdam: `www.cs.vu.nl/ibis/`

[25] W. A. JANSEN, Cloud Hooks: Security and Privacy Issues in Cloud Computing. *the 44th Hawaii International Conference on System Sciences*, (2011) Washington, DC, USA: IEEE Computer Society, pp. 1–10.

[26] JOYENT, *The Compelling Economics of Cloud Computing*. Joyent, 2012.

[27] P. KACSUK, Z. NÉMETH, T. FAHRINGER, *Distributed and Parallel Systems: From Cluster to Grid Computing*. Springer, 2007.

[28] B. KAHANWAL, T. P. SINGH, The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle. *International Journal of Latest Research in Science and Technology*, (2012), 183–187.

[29] T. V. KESSEL, N. DROST, J. MAASSEN, H. BAL, F. SEINSTRA, A. PLAZA, Towards a High-Performance Distributed CBIR System for Hyperspectral Remote Sensing Data: A Case Study in Jungle Computing. In *High-Performance Computing on Complex Environments* (E. JEANNOT, J. ZILINSKAS), (2014). Wiley.

[30] B. KLEYMAN, *Welcome to Fog Computing: Extending the Cloud to the Edge*. (2013, August 23). Retrieved from Data Center Knowledge: `http://www.datacenterknowledge.com/archives/2013/08/23/welcome-to-the-fog-a-new-type-of-distributed-computing/`

[31] R. LAMOTHE, *Edge Computing*. Richland, USA: Pacific Northwest National Laboratory, 2013.

[32] D. LILLETHUN, D. HILLEY, S. HORRIGAN, U. RAMACHANDRAN, MB++: An integrated architecture for pervasive computing and high-performance computing. *International Conference on Embedded and Real-Time Computing Systems and Applications*, (2007) Daegu, South Korea: IEEE, pp. 241–248.

[33] J. MAASSEN, H. E. BAL, SmartSockets: Solving the Connectivity Problems in Grid Computing. *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, (2007) Monterey, California, USA: ACM, pp. 1–10.

[34] J. MAASSEN, N. DROST, H. E. BAL, F. J. SEINSTRA, Towards jungle computing with Ibis/Constellation. *Workshop on Dynamic Distributed Data-intensive Applications, Programming Abstractions, and Systems*, (2011) New York, NY, USA: ACM, pp. 7–18.

[35] H. MADSEN, G. ALBEANU, B. BURTSCHY, F. POPENTIU-VLADICESCU, Reliability in the Utility Computing Era: Towards Reliable Fog Computing. *International Conference on Systems, Signals and Image Processing (IWSSIP)*, (2013) Bucharest, Romania: IEEE, pp. 43–46.

[36] D. C. MARINESCU, *Cloud Computing: Theory and Practice*. Morgan Kaufmann, 2013.

[37] P. MELL, T. GRANCE, *The NIST Definition of Cloud Computing*. Information Technology Laboratory: National Institute of Standards and Technology, 2009.

[38] J. M. MYERSON, *Best practices to develop SLAs for cloud computing*. IBM, 2013.

[39] N. DROST, Zorilla: a peer-to-peer middleware for real-world distributed systems. *Concurrency and Computation: Practice & Experience*, **23**(13) (2011), 1506–1521.

[40] M. NEMIROVSKY, R. MILITO, M. YANUZZI, *Fog Computing*. Barcelona Supercomputing Center, 2012.

[41] NETFLIX, (n.d.). Retrieved from `http://www.netflix.com`

[42] R. V. NIEUWPOORT, T. KIELMANN, H. E. BAL, User-friendly and reliable grid computing based on imperfect middleware. *Proceedings of the ACM/IEEE Conference on Supercomputing*, (2007) Reno, Nevada, USA.

[43] D. NURMI, R. WOLSKI, C. GRZEGORCZYK, G. OBERTELLI, S. SOMAN, L. YOUSEFF, D. ZAGORODNOV, The Eucalyptus Open-source Cloud-computing System. *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, (2009) Shanghai, China: IEEE Computer Society, pp. 124–131.

[44] T. PERRY, *What Comes After the Cloud? How About the Fog?* (2013, Feburay 8). Retrieved from IEEE Spectrum: `http://spectrum.ieee.org/tech-talk/computing/networks/what-comes-after-the-cloud-how-about-the-fog`

[45] L. QIAN, Z. LUO, Y. DU, L. GUO, Cloud Computing: An Overview. *the 1st International Conference on Cloud Computing(CloudCom 09)*, (2009) Berlin, Heidelberg: Springer, pp. 626–631.

[46] RACKSPACE, (n.d.). Retrieved from `http://www.rackspace.com`

[47] R. RAMAN, M. LIVNY, M. SOLOMON, Matchmaking: istributed resource management for high throughput computing. *Procceding of the Seventh IEEE International Symposium on High Performance Distributed Computing*, (1998) Chicago, IL, USA, pp. 140–146.

[48] E. RUDENKO, *Fog Computing Is a New Concept of Data Distribution*. (2013, December 5). Retrieved from CloudTweaks: `http://www.cloudtweaks.com/2013/12/fog-computing-is-a-new-concept-of-data-distribution/`

[49] M. SATYANARAYANAN, P. BAHL, R. CACERES, N. DAVIES, The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, (2009), 14–23.

[50] F. J. SEINSTRA, J. MAASSEN, R. V. NIEUWPOORT, N. DROST, T. V. KESSEL, B. VAN WERKHOVEN, A. H. BAL, Jungle Computing: Distributed Supercomputing Beyond Clusters, Grids, and Clouds. In *Grids, Clouds and Virtualization* (M. CAFARO, G. ALOISIO), (2011) pp. 167–199. London: Springer.

[51] SYMFORM, (n.d.). Retrieved from `http://www.symform.com/`

[52] A. S. TANENBAUM, M. V. STEEN, *Distributed systems: Principles and Paradigms*, (2nd Edition). Pearson Prentice Hall, 2006.

[53] L. M. VAQUERO, L. RODERO-MERINO, J. CACERES, M. LINDNER, A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, **39**(1) (2009), 50–55.

[54] M. A. VOUK, Cloud Computing – Issues, Research and Implementations. *Journal of Computing and Information Technology*, (2008), 235–246.

[55] Z. XIAO, Y. XIAO, Security and Privacy in Cloud Computing. *IEEE Communications Surveys & Tutorials*, (2013), 843–859.

[56] C.T. YANG, B.-H. CHEN, W.-S. CHEN, On Implementation of a KVM IaaS with Monitoring System on Cloud Environments. In *Communication and Networking* (T.-H. KIM, H. ADELI, W.-C. FANG, T. VASILAKOS, A. STOICA, C. Z. PATRIKAKIS, Y. XIAO), (2012) pp. 300–309. Berlin: Springer.

[57] Q. ZHANG, L. CHENG, R. BOUTABA, Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, (2010), 7–18.

[58] J. ZHU, D. S. CHAN, M. S. PRABHU, P. NATARAJAN, H. HU, F. BONOMI, Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture. *IEEE Seventh International Symposium on Service-Oriented System Engineering*, (2013) IEEE, pp. 320–323.

Received: March, 2014
Revised: June, 2014
Accepted: June, 2014

Contact addresses:

Majid Hajibaba
Department of Electrical and Information Technology
Iranian Research Organization for Science and Technology (IROST)
Tehran
Iran
e-mail: `hajibaba.m@irost.ir`

Saeid Gorgin
Department of Electrical and Information Technology
Iranian Research Organization for Science and Technology (IROST)
Tehran
Iran
e-mail: `gorgin@irost.ir`

MAJID HAJIBABA received his MS degree in computer science from Iran University of Science and Technology. Currently, he is a Ph.D. student at Iranian Research Organization for Science and Technology. His research in Ph.D. is focused on distributed computing. Other research interests include programming languages, compilers, software testing, cloud computing, big data and high performance computing.

SAEID GORGIN received the BS and MS degrees in computer engineering from the South Branch and the Science and Research Branch of Azad University of Tehran in 2001 and 2004, respectively. He received the Ph.D. degree in computer engineering from Shahid Beheshti University in 2010. He is currently an assistant professor of computer engineering in the Department of Electrical and Information Technology of Iranian Research Organization for Science and Technology. His research interests include computer arithmetic, cryptography, VLSI design, and high performance computing.