International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 6, No.5 (Sep-2017)
http://dx.doi.org/10.12785/ijcds/060502

Evolution of the Distributed Computing Paradigms: a Brief Road Map

Haitham Barkallah[1], Mariem Gzara[1,2] and Hanene Ben Abdallah[1,3]

[1] *University of Sfax, MIRACL laboratory, Rt Tunis, B.P. 242, 3021 Sfax, Tunisia*
[2] *University of Mounastir, Higher Institute of Computer Science and Mathematics, Tunisia*
[3] *King Abdulaziz University, Jeddah, KSA*

Received 11Jul. 2017, Revised 8 Aug. 2017, Accepted 22 Aug. 2017, Published 1 Sep. 2017

Abstract: Today's computing development is being characterized by the rapid development of high speed networks and the increase in computing power. Computing is not any more limited to the supercomputers, PCs and laptops but also smart phones and tablets which are available for billions of users offering high computing performances at low cost and interconnected via Internet. This continuing technological development is leading the increase importance of the distributed computing paradigms and the apparition of new ones. This paper aims to review the most important distributed computing paradigms and the principal similarities and differences between them. This survey is a kind of a brief road map that would be useful for researchers, students, and commercial users.

Keywords: Distributed Computing Paradigms, Cluster, P2P, Redundant, Pervasive, Edge, Jungle, Volunteer, Utility, Grid, Cloud, Service, Review

1. INTRODUCTION

Computers has marked the 20st century and changed radically the way we live and work. In scientific research, they have enabled the resolution of many problems that where impossible to do without computers. The need for computing power continues to rise and new problems that need massive computing resources show up.

Taking advantage of the fast technological progress in networks and especially the emergence and evolution of Internet, new computing paradigms have been proposed based on the coordinated use of any available distributed resources in the world ranging from private to public and volunteer resources and from supercomputers to mobile phones.

The availability of high speed networks at low cost has revolutionized our everyday computing practices and enabled the emergence of new distributed computing paradigms.

In the literature, many authors tried to study the existing distributed computing paradigms like B. Kahanwal [1] and some others had the vision to anticipate and propose new ones like M. Weiser [2].

In [3], a comparison between the computing models of cluster, grid, and cloud computing is presented and in [4], the authors have included utility computing as a generic model. While in [5], [6], and [7], a side by side comparison between the grid and cloud computing model from various angles focusing on their essential characteristics are presented.

In [8], the authors analyzed the performance of three real scientific applications (AutoDock, Montage, and ThreeKaonOmega) and made an evaluation of their performance on different systems: Clusters, Grids, and Clouds. They aimed to show how the application performances can be significantly affected by the system's characteristics and the hardware specification of the computing nodes.

In [9], the authors discussed the reliability model of volunteer peer-to-peer cloud computing. They proposed a mechanism using dynamic replication to ensure trust relationships and results correctness and to provide efficient computation. While in [10], the authors investigated the different alternatives like cluster, grid computing and cloud computing (IaaS or PaaS) for running an Adhoc data intensive application rather than

E-mail: h.b.tn@ieee.org, mariem.gzara@gmail.com, hbenabdallah@kau.edu.sa

using supercomputers. Alternatives to supercomputers are explored in this paper.

A comprehensive explanation of cloud computing, volunteer computing and also volunteer cloud computing paradigms along with their advantages and also their open issues is presented in [11]. And in [12], K. Skala and al. presented the cloud computing, fog computing and dew computing as new emerging paradigms that can lower the cost and improve the performance of the Internet of Things (IoT) and the Internet of Everything (IoE) applications.

This work is the result of a large inspection of many scientific documents (papers, books chapters, and PHD reports) related to the distributed computing. A deep process of collection, selection, and analysis has enabled the establishment of this review.

This paper surveys the most important distributed computing paradigms. It is useful, especially, for debutant researchers, students, and commercial users. It consists of an introductory knowledge that help them get an idea about the past and the future of distributed systems. It helps them differentiate these paradigms and avoid confusion while providing deep and brief understanding. For each paradigm a brief introduction is presented then its goals, limits, and problems are discussed. We strive to compare and contrast the principal distributed computing paradigms from various angles and give insights into their essential characteristics.

This work can help to understand, share and predict the distributed computing paradigms' evolution.

This review paper is organized as follows. Section II presents the limits of the serial and parallel computing leading to the need of distributed computing. Section III, details and discusses the principal distributed computing paradigms: cluster, peer-to-peer, redundant, ubiquitous, pervasive, mobile, edge, jungle, volunteer, service, grid, and cloud computing. Then section IV aims at comparing the different paradigms according to the geographical distribution, the resource ownership, availability, and reliability and the system's scalability. Before concluding this work, section V discusses the distributed computing as a service. And we conclude by, the relation between the different distributed computing paradigms and how they coexist in real life systems.

2. LIMITS OF SERIAL AND PARALLEL COMPUTING

A. *Serial vs Parallel computing*

Traditionally, software has been written for serial computation: the problem is broken into a discrete series of instructions executed sequentially one after another on a single processor (*Figure 1*). While in parallel computing the problem is broken into discrete parts that can be solved concurrently. Instructions from each part execute simultaneously on different processors. An overall control/coordination mechanism is employed [13].

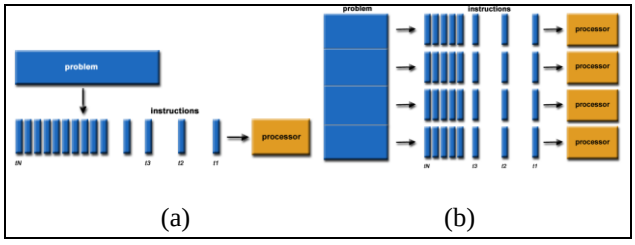

Figure 1. Serial computing (a) vs Parallel computing (b) [13]

B. *Parallel computing goal*

The basic idea of parallel computing is to divide a large problem into smaller ones which can be carried out simultaneously on multiple processors in order to be solved faster [14][15].

B. Barney confirmed that "*compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real world phenomena*" [13]. In fact, parallel computing had been considered to be "*the high end of computing*", and has been used to model very difficult problems in many areas of science and engineering.

Parallel computing had made a huge impact on a variety of problems ranging from computational simulations for scientific and engineering applications to commercial applications in transaction processing and data analysis [16].

It has enabled scientists and application developers to achieve computational results which could not be obtained efficiently by serial computing methods. Then, a new larger, and more complex applications has been developed solving problems that could not be solved previously [17].

C. *The Moore's law impact*

In 1965, Electronics magazine asked the research director of electronics pioneer Fairchild Semiconductor to predict the future of the microchip industry. Moore guessed that the number of electronic devices crammed onto microchips would roughly double every year. Moore's Law [18], became the golden rule for the electronics industry. And have had economic, technological and societal impacts.

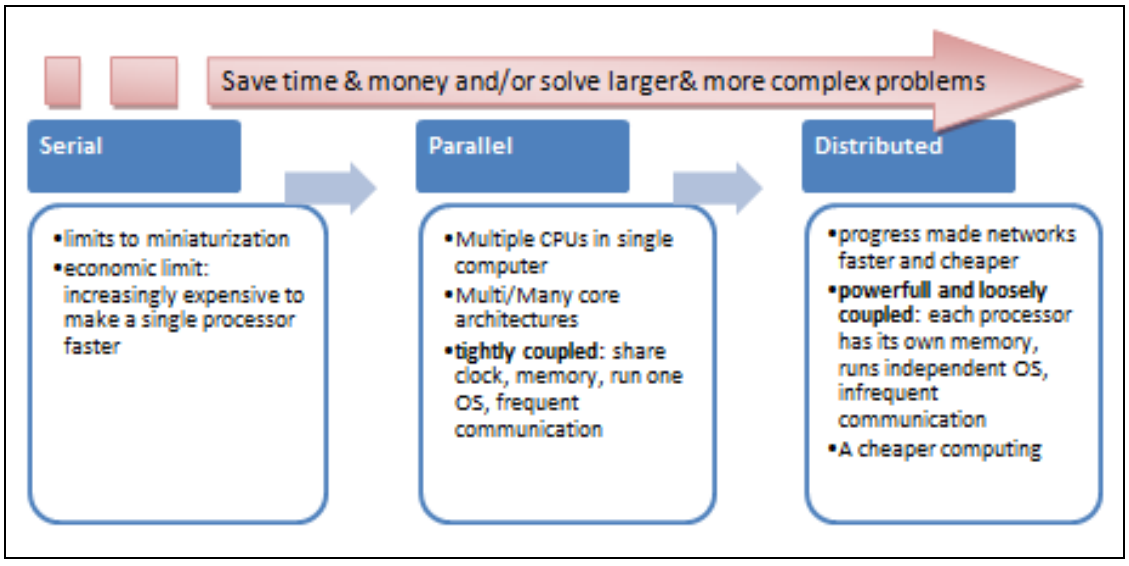

Figure 2. The need for distributed computing

In fact, for decades, the Von Neumann architecture [19] has known many improvements like: increasing Clock Frequency, Memory Hierarchy / Cache, Parallelizing ALU structure, Pipelining, Very-long Instruction Words (VLIW), Superscalar processors, Vector data types, Multithreaded concurrent programming, instruction-level parallelism (ILP), multicore processors, etc.

However, since 2005 there has been no increment in the processor's speed which attained the physical limit in traditional uni-processors [20]. Herb Sutter claimed "*We'll probably see 4GHz CPUs in our main stream desktop machines someday, but it won't be in 2005*" [20]. Then the trend in parallel architectures especially multicore-processors and distributed computing (Figure 2) is how to meet the need for speeding up computation.

3. DISTRIBUTED COMPUTING PARADIGMS

The technological progress in network speed and bandwidth made possible the emergence and evolution of distributed computing.

It started in the 1960s within the message-passing communication between two or more computing resources. Then the local area networks, ARPANET, and especially the introduction of the email service marked the 1970s.

The use of distributed computing has known a huge expansion benefiting from the availability of powerful computing resources and network speed to the public at low cost. It made possible the resolution of large complex problems that where considered extremely hard to solve.

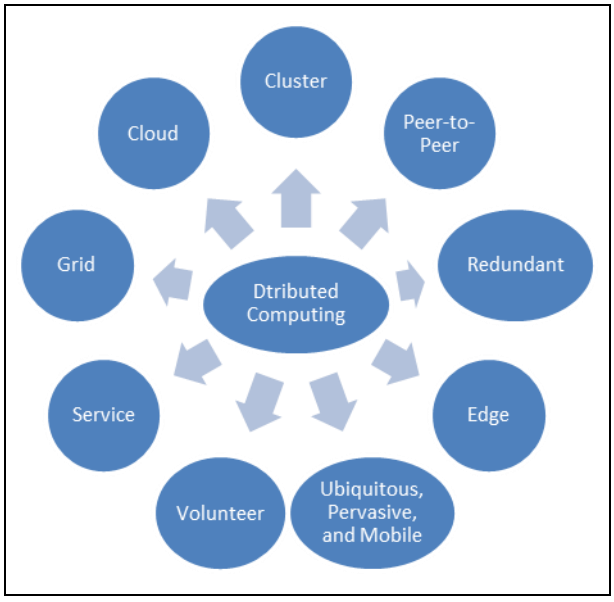

Figure 3. Principal distributed computing paradigms

In this section, we are going to present the principal distributed computing paradigms (Figure 3): the idea behind them, their goals, problems, and limits. Then in the next section we are going to compare them.

A. Cluster computing

Clusters are groups of co-located standalone computers interconnected using a high speed network [21], working together closely so that in many aspects they form a single computer. Clusters are, naturally, suited to systems that perform large numbers of independent (or nearly independent) small computations [22]. The components of a cluster are commonly, but not always, connected to each other through fast LAN

[14]. The set of loosely or tightly connected computers are viewed as a single system.

The success of cluster computing is based on the use of low-cost and widely-used commercial hardware and software. And its primary objective in high performance cluster computing is to reduce the execution time [23]. In practice, each node of a cluster runs a local copy of a uniprocessor operating system. Hence, any system-level management must be done by "middleware" above the OS [24].

Clusters can be classified according to:

- **The type of used hardware:**
  - **Work stations based Clusters:** Composed of low-cost computers.
  - **High-performance clusters:** High-performance clusters use supercomputers to solve advanced computation problems.
- **Their visibility to the publics**
  - **Open cluster:** All nodes are visible from outside, and hence they cause security concerns. But they are more flexible and can be used to perform Internet and web tasks for example.
  - **Close Cluster:** The nodes are hidden behind a gateway node which provides better security level.

1) Goals

Cluster computing has two major goals:

- **Aggregating computing power:** Aggregating many single computers working together so that in many aspects they form a single powerful computer.
- **High availability clusters:** The clusters are designed to maintain redundant nodes that can act as backup in case of system failure.
- **Load-Balancing clusters:** Load-balancing clusters are used to redistribute the workload efficiently between the active nodes in order to optimize the processing power efficiency.

2) Limits

The cluster computing paradigm presents two principal limits:

- The computing applications and the treated problems are increasingly complex and require high amounts of computing resources that exceed the clusters' capacity.
- Sometimes, the aggregated computing power is not enough to handle the peak of loads and produce the desired output in time creating the possibility of system's unavailability and/or unreliability.

3) Problems

The two major problems of cluster computing are:

- **High costs:** implementing and maintaining a cluster computing infrastructure is, of course, cheaper than the HPC supercomputers, but it is still considered expensive.
- **Low computing power utilization's rate:** usually, the cluster's resources are not entirely used most of the time.

B. Peer-to-Peer

In Peer-to-peer computing [25], in contrast to Client-Server systems, every node acts as both a client and a server providing part of the system resources. All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers. No central coordination or no central database is needed. In other words, no peer machine has a global view of the entire P2P system. The system is self-organizing with distributed control without the need for centralized coordination by central server [1].

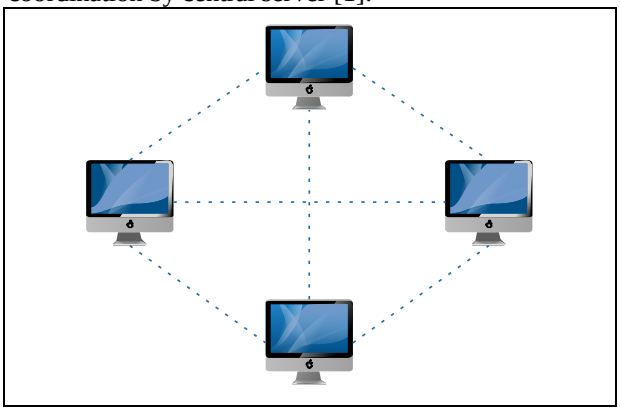

Figure 4. Peer-to-Peer network

The shared resources and services include the exchange of information, processing cycles, cache storage, and disk storage for files [26]. They are used in many application domains such as data transfer and data storage but the concept was popularized by file sharing systems such as the music-sharing application Napster [27].

1) Goals

Peer-to-peer computing resides on the edge of the Internet or in ad-hoc networks [25]. It has several goals [1][25]:

- **Cost sharing/reduction:** eliminating the need for costly infrastructure by enabling direct communication among clients. The maintenance cost is spread over all the peers [25].

- **Resource aggregation (improved performance) and interoperability:** each node brings with it certain resources such as compute power or storage space. "*The whole is made greater than the sum of its parts*" [25].
- **Improved scalability/reliability:** by avoiding dependency on centralized points and inducing maximum load distribution [28].
- **Increased autonomy:** the local node does work on behalf of its user.
- **Anonymity/privacy:** allowing peers a greater degree of autonomous control over their data and resources [25]. The system has no server that will typically be able to identify the client. The users can avoid having to provide any information about themselves to anyone else.
- **Dynamism:** resources enter and leave the system continuously.
- **Enabling ad-hoc communication and collaboration:** the system takes into account changes in the group of participants such as members come and go based on their current physical location or their current interests.

2) Limits

Peer-to-peer computing suffers from different limits, especially:

- Offers Limited bandwidth especially in case of non-dedicated network resources compared to the high bandwidth speed allowed by cluster computing. Peer-to-Peer infrastructures usually relies on Internet in order to interconnect the resources, usually geographically distributed, while in cluster computing dedicated network infrastructure is required.
- Redundancy cost: redundancy is important in order to ensure reliable data storage within unreliable peers but this can affect the network bandwidth and the global storage capacity of the whole system.
- Incompatibility issues: many traditional algorithms and solutions perform poorly running on the peer-to-peer infrastructure considered to be unreliable and offering limited computing, storage, and bandwidth compared to cluster and super computers.

3) Problems

Peer-to-peer computing presents serious problems like:

- Security and privacy weaknesses: the peer-to-peer infrastructure is accessible to all the peers without the need of a centralized security access point. The exchanged and/or stored data needs to be encrypted in order to prevent manipulation. Usually, trust qualification mechanisms are implemented in order to evaluate the peers' behavior.
- Heterogeneity of the peers: unlike clusters, in peer-to-peer infrastructures, the peers are usually heterogeneous in terms of memory, storage and computing capacities, network bandwidth...
- Reliability of the resources: in peer-to-peer systems sudden arrival and departure of peers is very frequent. This problem is considered to be the most challenging and can, extremely, limit the system's availability and scalability.

C. Redundant Computing

Ensuring the reliability of a distributed system is primordial especially when the resources availability and trustiness are not guaranteed. In that case, redundant computing is the appropriate response.

Redundant computing is based on two techniques:

- Job replication: sending copies to different resources [29]:
  - If the deadline respect is not guaranteed to escape possible tardiness or job execution failure
  - The results received from different resources are compared to ensure trustiness [30][9][31].
- Resource redundancy: resources are doubled to be used in case of failure.

Redundant computing is especially used in two types of situations:

- Volunteer computing: when the resources are owned by users who voluntarily allow their utilization.
- Critical computing: when computing errors, execution failure [32] and hard deadline real-time job execution are never permitted and can cause life and/or financial loss.

1) Goals

By incorporating redundant processing elements in a distributed system, one can potentially:

- **Increase system's reliability and/or availability:** In fact, it allows applications to continue working even when failures occur[32].The users can accept a partial degradation in system performance, when a failure cripples a fraction of the resources or links of a distributed system. For S. Ghosh, a distributed system thus provides an excellent opportunity for incorporating fault-tolerance and graceful degradation [33].
- **Deal with erroneous results:** For Syed A. Ahson and Mohammad Ilyas[31], in addition to the mosaic of resources (high heterogeneity), the

application results returned to the master are subject to errors. These errors can occur because of hardware malfunction (particularly on over-clocked computers) or malicious volunteers attempting to get credit for computing not actually performed. Basically, the result is considered valid when it reaches a consensus (a set of similar results) by running the same computations on a number of resources.

- Used to address the issues to the volunteer computing, including malicious attacks, network latency, and hardware malfunctions [30]. This mechanism involves dispatching two or more replicas of a same job to different computers. A job is complete and the corresponding results are valid only if a certain number of volunteers respond and all report the exact same result. B. Sawicki[9] considered redundant computing (also called replication) as a method of protecting the system from dishonest nodes.

2) *Limits*

Redundant computing has two main limits:

- **Low resource utilization rates:** Each unit of work is sent to at least two computers. The results from those computers are returned to the server and compared to see if the results agree. If they do not, then additional copies are sent to different computers until a consensus can be reached. Results can be compared to see if they are either identical or within an acceptable range of variance. This technique is crucial especially in volunteer computing in order to overcome the problem of malicious peers and settle down a trust ranking system.

- **High Cost:** redundant computing is based on resources replication which involves increasing cost of additional computing resources that might not be used only during system failure or maintenance periods. This explains the fact that redundant computing is practically used in volunteer computing since the resources are free or in very critical computing infrastructures where system's reliability and availability problems are never acceptable.

3) *Problems*

The major problems of redundant computing are related to resources waste and data inconsistency:

- High amounts of resources use: in redundant computing based on job replication, *n* copies of the jobs/applications are sent to the computing resources in order to be executed. The necessary used computing power is then multiplied by n including the data storage capacity and the network bandwidth.

- Data replication management: due to the job replication process the same piece of data is held in many separate places. Data redundancy can lead to data inconsistency problems especially when the executed jobs are dependent (the output of one is the input of another).

D. *Edge Computing*

Edge computing is a natural extension of the Content Delivery Network (CDN) architecture [34][35]. It pushes application logic and the underlying data processing from corporate data centers out to proxy servers at the "edge" of the network in order to achieve scalable and highly available Web services [34] and for better efficiency and performance [36].

Edge computing has several applications in different domains. For G. Roussos and al. [37] this shift towards edge computing is typical in pervasive computing applications and is observed in a variety of related situations, notably with wireless sensor networks. Then all information processing, content routing, and persistence are all located at the edge, which gains a role of far greater importance.

Also, in modern Internet applications, by replicating application contents (e.g., Web objects and fragments of DBMS data) and logic (e.g., scripts for dynamic generation of Web contents and remote interaction with origin sites) across a large number of geographically distributed servers, edge computing platforms allow to achieve significant enhancements of the proximity between clients and contents, and of the system scalability [38]. Desktop machines are being used to perform some of the computation instead of all computation happening on servers. Client side scripts and web applications enable much of the computation occurs on the clients' side [21].

1) *Goals*

According to H. Pang and al. [34], performing computation at the edge of the network instead of machines at the core [21] has several potential advantages like:

- Cuts down network latency and produces faster responses to end-users "applications and partners" Web services. In fact, it moves some of the data processing closer to devices that require real-time interaction, thus reducing the number of network hops and hence latency [39].

- Adding edge servers near user clusters is also likely to be a cheaper way to achieve scalability than fortifying the servers in the corporate data center and provisioning more network bandwidth for every user.

- removes the single point of failure in the infrastructure by lowering the dependency to the corporate data center, hence reducing its

susceptibility to denial of service attacks and improving service availability [34].

2) Limits

Edge computing has two major limits:

- **Tasks and data partitioning:** in order to be executed at multiple geographic locations, the tasks and the necessary input data have to be partitioned into multiple independent sub entities which is not possible most of the time.
- **Technological limitations:** edge computing provides a generic template for applications development that imposes certain technological limitations in order to facilitate the application's distribution among the different nodes.

3) Problems

The principle edge computing problems are related to the load distribution and the system's security:

- **Data replication management:** Due to the job partitioning process the user's/application's data is held in different locations. This can lead to data inconsistency problems especially when the input data is shared between the different sub-jobs and is subject to modifications during the execution.
- Offloading the load to the nodes is associated with security risks especially when the computing resources are not privately owned. Publicly accessible nodes pose a number of security challenges that need to be addressed.

E. Ubiquitous, Pervasive, and Mobile computing

Ubiquitous computing, which means "existing everywhere", is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user [40][2]. Its beginning was in the Electronics and Imaging Laboratory of the Xerox Palo Alto Research Center [41] in the early 80's. And it offers a framework for new and exciting research across the spectrum of computer science [40].

Ubiquitous computing started in Xerox's Palo Alto Research Center. In 1991, Mark Weiser, the chief technology officer, described the ubiquity of personal computers: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [42]. He anticipated "the ubiquitous computer leaves you feeling as though you did it yourself. Its extensive use of video and audio, including voice communication, will transform electronic interfaces into interpersonal ones" [2].

1) Goals

Ubiquitous computing touches on a wide range of research topics, including distributed computing, mobile computing, location computing, mobile networking, context-aware computing, sensor networks, human-computer interaction, and artificial intelligence. It is different from pervasive computing by:

- Integrating large-scale mobility [43].
- Including invisibility to the user much more than mobile computing [44].
- Embedding microprocessors in everyday objects so they can communicate information.
- Introducing seamless access to remote information resources and communication with fault tolerance, high availability, and security [42].

2) Limits

Ubiquitous computing has three important limits:

- **Limited network bandwidth:** even though mobile network bandwidth is now more and more powerful, especially with the emerging 3G, 4G, and 5G technology, but it is still slower than direct cable connections which has direct impact on the type of applications. This has direct impact on the kind of applications which can be deployed and executed.
- **Energy consumption:** since the resources used in ubiquitous, pervasive, and mobile computing rely on battery power, energy consumption restrictions are imposed.
- **Human-machine interface:** in the mobile devices, screens and keyboards are small, which makes restrictions on the application's interface and input/output functionalities.

3) Problems

The most important problems in ubiquitous, pervasive, and mobile computing are:

- **Lack or poor network coverage:** the computing resources are supposed to be mobile and geographically distributed, but the signal reception is not guaranteed and sometimes poor or even unavailable in tunnels, some buildings, and rural areas.
- **Potential health risks:** some mobile devices may interfere with sensitive devices especially medical. And, due to the possibility of distraction, users using their mobile devices while driving are most likely involved in traffic accidents.
- **Security concerns:** using a huge number of resources relying on public networks, is usually source of security problems.

F. Volunteer Computing

Volunteer computing is often named Desktop computing as it uses desktop computers as the

underlying computational resources [31]. Most of the volunteer computing platforms have the same structure: a client program runs on the volunteer's computer. It periodically contacts project servers over the Internet, asking for jobs and sending back the results of completed jobs.

1) Goals

Volunteer computing is a type of distributed computing that allows people donating their computing resources. The idle time of PC's is used to do research and scientific projects. The system employs unused CPU cycles to fulfill the work. By breaking a monolithic job into a large number of units and distributing into different desktop computers called workers, it can perform jobs efficiently via massive parallelism [30].

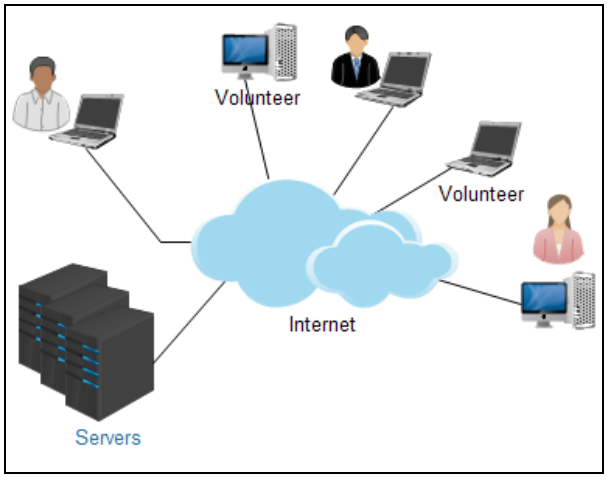

Figure 5. Volunteer Computing

Nowadays, volunteer computing provides Petaflops of processing power to solve large variety of problems like formalization of complex mathematics models, climate changes prediction, etc. [45].

2) Limits

Volunteer computing also has several limits:

- The resources need to be treated as totally independent entities and can be suitable most for independent tasks that do not require any inter-task communications.
- The task data input and output data transfers must be small and the task runtime has to be short.
- The resources are volatile and the whole system depends on the willingness of the public to share their computational power.

3) Problems

a) Environment complexity:

To develop Volunteer computing platforms, we have to consider many problems. One of these issues is we have to deploy easy to understand environment. This is because of the users (donors) are in a wide technical background ranges. Our platform must attract more volunteers to have more systems and so more powerful computing power [45].

b) Resource Availability:

Since volunteer computing heavily relies on the donation of computer CPU cycles by public participants whose identities are unknown, the resource availability must be taken into consideration before operating on these resources [30].

c) Resource heterogeneity:

Due to volunteer's nature, donors might own very different systems with a variety of OSs and different software and applications. So we have to consider framework independence to avoid from compatibility issues [45]. It is a "*mosaic of resources*" [31].

d) Trustiness:

As the computers are not controlled by the person or organization conducting processing on the grid, the volunteered computers cannot be trusted [29][31][9].

In fact, in addition to this mosaic of resources, the application results returned to the master are subject to errors. These errors can occur because of hardware malfunction (particularly on over-clocked computers) or malicious volunteers attempting to get credit for computing not actually performed [31].

In order to produce reliable results redundant computing is used. The system sends each unit of work to at least two computers. The results from those computers are returned to the server and compared to see if the results agree. If they do not, then additional copies are sent to different computers until a consensus can be reached. Results can be compared to see if they are either identical or within an acceptable range of variance [29].

Trust of particular node depends on how many jobs are confirmed and by whom. The more trusted a node is, the more valuable its confirmation is. As node has more of its results confirmed, its trust grows [9].

In [33], in order to guarantee fault-tolerance, processors cross-check one another at predefined checkpoints, allowing for automatic failure detection, diagnosis, and eventual recovery. The correct result is determined by a majority vote.

G. Service computing

Service computing (or service-oriented computing) is "a way of developing application systems with services as the basic elements" [46]. It is an effective approach for distributed computing paradigm and one of

keys of cloud computing used to model, create, operate, and manage business services [47].

The service computing model is based on a service registry, service consumer, and service provider. It is driving distributed computing towards a model of dynamic service based interactions. First, the service provider registers itself in the registry. The service consumer, then, discovers the service from the registry, and uses it.

1) Goals

The fundamental objectives of service computing are:

- To enable a maximum level of interoperability among different applications running on different platforms.
- It is widely used to realize distributed applications/solutions based on loosely coupled, reusable services that can be binned to form service compositions [46].
- **Easy Maintainability:** since a service is an independent entity, it can be easily updated and/or maintained.
- **Better Scalability and Availability:** in fact, multiple instances of a single service can run on different servers at the same time. This improves the response time and the satisfaction for the users.

2) Limits

The mutation to service oriented architectures is considered as hard and expensive. In most cases, the change is implemented incrementally and carefully during a long period.

3) Problems

The service oriented architecture is not considered well suitable to the applications requiring high amounts of data exchange or do not require request/response asynchronous communication and, also, standalone and short lifespan applications' implementations.

H. Grid Computing

Grid computing is the sharing of computing resources (computers, clusters, parallel machines, ...) by a collection of people and institutions in a flexible and secured environment [14]. It creates the illusion of a simple and powerful self-managing virtual computer out of a large collection of heterogeneous systems and resources [48]. The aim is to enable coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. Clabby Analytics defined the grid as a distributed network architecture that finds and exploits unused compute/storage resources which reside within a distributed computing environment [49].

Grid computing can be used in a variety of ways to address various kinds of application requirements. There are a large number of projects around the world working on developing Grids for different purposes at different scales in order to provide non-trivial services to users [50].

1) Goals

Using grid computing has many motivations [51]:

- **Allows sharing computing resources across networks:** This can increase the available computational power and reduce the number of computers needed by an organization.
- **Allows low-cost computing:** enables linking a large number of low-cost machines together rather than spending a large amount of money on a single machine or super-computer with a larger processing capability.
- **Exploiting underutilized resources:** computing resources are most of the day time idle. For example, desktop machines are busy less than 5 percent of the time especially during off business hours [52].
- **Access to additional resources:** Grid enables the virtualization of distributed computing and data resources such as processing, network bandwidth, storage capacity, special equipment, software, licenses, and other services to create a single powerful system image, granting users and applications seamless access to vast information technologies capabilities. In order to meet internal deadlines, it gives access to on demand additional capacity during peak production cycles, and improves utilization of existing resources.
- **Virtual resources and virtual organizations for collaboration:** Grid computing is more than just cluster computing in the "large". It offers the potential for groups of people both geographically and organizationally distributed to work together on problems, to share computers AND other resources such as databases and experimental equipment.
- **High system's scalability and reliability levels:** The grid model scales very well. In order to add more compute resources, one just have to plug them in by installing grid client on additional desktops or servers. They can be removed just as easily on the fly. The grid system doesn't have single points of failure. When there is any kind of detected failure at one location, Grid Management Service can automatically resubmit jobs to other machines on the grid. The user doesn't need to rush either because nobody should even notice that a node is down as long as there are other nodes

available to take jobs. It can also run multiple copies of important jobs on different machines in order to minimize failure probability [41].

2) Limits

Grid computing has great potential, but there are still some limits that must be overlooked.

- In order to use all the benefits of grid computing, the users' applications require modifications before it can be run on the grid. These modifications can be expensive especially for the applications that not designed to use an MPI (Message Passing Interface)
- Since the job execution is shared among multiple nodes, the grid users have to wait until all processes send their sub results and then collaboratively assessed. Before that, it is not possible to define or to declare a final outcome. This is considered as a serious limit for time sensitive projects.
- In most cases, grid computing is considered to be more adapted to batch jobs [53]. In fact, the grid is usually aggregating non dedicated resources that is hard to rely on for critical real-time applications.

3) problems

Grid computing has two major problems:

- It suffers from security issues compared to other distributed paradigms. In fact, grid jobs are considered as external code that can be harmful for the local resources. Also, major vulnerabilities including resources disconnection resulting in denial of service, issues related to data integrity and confidentiality, vulnerable hosts sending back incorrect results.
- It relies heavily on dispersed data management and network connectivity. This has big impact on the system's availability and reliability especially when the resources are based on personal volunteering effort.

I. Cloud Computing

Cloud computing is a large-scale distributed computing paradigm that has evolved from work in areas such as Grid computing and other large distributed applications [54]. It is a model of supplement, consumption, and delegation of computing resources to the users on-demand [30].

Cloud computing is another form of utility computing which allows reasonably priced use of computing infrastructures and mass storage capabilities [1]. It 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 [30][6].

Cloud computing offers its benefits through three types of service or delivery models (Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)) and different types of Clouds (Private, Public, Hybrid, and Community) [1].

1) Goals

The cloud computing paradigm has different goals:

- **Reduced computing infrastructure's cost:** adopting cloud computing reduces the cost of managing and maintaining the IT systems. In fact, rather than purchasing expensive computing resources, cloud computing can reduce the costs by using the cloud resources offered and maintained by a service provider. The cloud's client has no longer to think about the operating costs: like energy consumption, wages for expert staff, resources maintenance and upgrade.
- **Better flexibility and scalability:** new computing resources can be added on the fly in order to meet the needs.
- **Better resource utilization:** cloud computing enables a better resource utilization rates and minimizes the resources waste. The users' only pay for use.
- **Data backup and disaster recovery:** cloud computing makes the entire process of backup and recovery much simpler than other traditional methods of data storage. The service providers are responsible for users' data storage, management, and to handle recovery of information.
- **Globalized workspace/easy data accessibility:** cloud users have the ability to access their data from wherever they are equipped with an internet connection. A globalized virtual workspace is offered in order to enable team work, collaboration and effectiveness.
- **Increased system's availability and reliability:** by achieving strong service level agreements, the cloud users' are offered effective and highly reliable computing infrastructures.

2) Limits

Cloud computing has different limitations like:

- **Peripherals**: Some of the peripheral devices might not work with cloud infrastructures and this may require some investment to inquire new peripherals.

- **Generic**: public clouds offer very generic and multi-tenancy services which require important efforts to adapt them to the needs.
- **Cloud platform dependent:** due to the platform differences between the cloud providers, it is not possible to migrate from one cloud platform to another without complex and expensive application's modification in order to meet the new provider's requirements.
- **Integration**: Integrating local applications with the cloud applications might be complex and in some cases not feasible.
- **Limited control and flexibility:** cloud users have limited control over the hosting infrastructure. The provider's policies might impose limits on the applications, data, and services, which should be taken into consideration.

3) Problems

The most important problems of cloud computing are:

- **Downtime:** Cloud computing makes the computing resources dependent on the reliability of the Internet connection. In case, the internet service at the client side suffers from frequent disconnections or slow download and upload speeds, especially for data massive applications, cloud computing may not be suitable.
- **Security and privacy:** The stored data is accessible from anywhere on the internet, meaning that can be susceptible to hacking or phishing attacks especially when it comes to managing sensitive data. The cloud provider is expected to manage and secure the clients' data, applications, and the underlying hardware infrastructure; however remote access is the clients' responsibility.
- **Hidden/additional costs:** Cloud computing allow reducing staff and hardware costs but the overall price tag could end up higher than expected. Some of the cloud providers charge the clients hidden or additional costs like for data transfer. The cloud computing may end up with higher costs compared to privately owned servers.

J. Jungle computing

Jungle computing was first introduced in 2011 by F. J. Seinstra and al. [55]. It refers to the use of diverse, distributed and highly non-uniform high performance computer systems to achieve peak performance [1].

The resulting distributed system, called Jungle Computing System, is both highly heterogeneous and hierarchical, potentially consisting of grids, clouds, stand-alone machines, clusters, desktop grids, mobile devices, and supercomputers, possibly with accelerators such as GPUs [56] (*Figure 6*). Jungle Computing Systems are Multi-Model / Multi-Kernel [56].

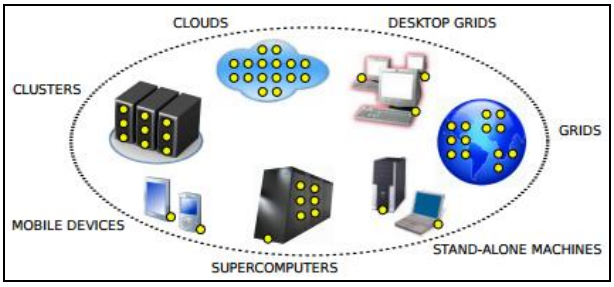

Figure 6. Jungle computing system [55]

1) Goals

There are several reasons for using Jungle Computing Systems. In fact, combining resources may be necessary if no single resource is available that is large enough to perform the required computation, or because different parts of the computation have different computational requirements [56].

2) Limits

Integrating such different kinds of resources at a very large scale is not an easy task. In fact, the aggregated resources belong to different administration domains, have their own security strategy, resource management policy, and objective functions, etc. Also limitations related to the system's development, deployment, and maintenance costs and economic feasibility has to be considered.

3) Problems

Jungle Computing System is highly heterogeneous. Resources differ in basic properties such as processor architecture, amount of memory, performance, and installed software such as compilers and libraries will also differ. The heterogeneity makes it hard to run applications on multiple resources. The application may have to be re-compiled, or even partially re-written, to handle the differences in software and hardware.

4. ANALYSIS OF THE DISTRIBUTED PARADIGMS

Technological progress doesn't stop changing the way we do computing. As a result, the different computing paradigms revolutionize the everyday practice of data storage and computing. These paradigms can be analyzed using different factors like geographical distribution of the computing resources, the system's performance, scalability, and availability to the users, fault tolerance, resources ownership, etc.

A. *Geographical distribution of the resources*

The emergence of high-speed networks at low costs and the availability of Internet to the large public have encouraged the geographical distribution of the resources at large scale (*Figure 7*).

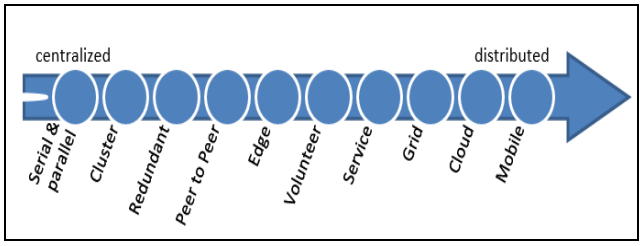

Figure 7. Geographical distribution of the resources

The local area networks have enabled the transition from serial and parallel computing to the cluster and redundant computing paradigms. After that, the apparition of Internet has, first, allowed the systems' designers to move the computing resources to the edge of the network creating, especially, local copies of the data in order to be closer to the users. Second, it enabled the raise of highly scalable utility computing paradigms like service, grid, and cloud.

B. *Resources ownership*

The distributed computing paradigms have evolved through time to include different types of resources (Figure 8) from the computing clusters based on private resources to the public grids and clouds integrating multiple types of public resources and offering them to the public using a pay per use business model.

It is also possible to have hybrid private and public resources combined together to form a large scale system. Or a community owned resources shared in the purpose of a specific common project between two or more teams and/or labs.

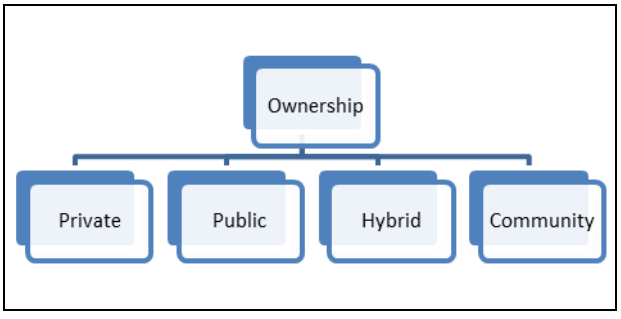

Figure 8. Resources classification based on ownership

C. *Resources availability and Reliability*

System's availability refers to the operational continuity of the system. While the reliability is "*related to systems and network failure, disconnection, availability of resources, etc.*" [1].

According to R. Buyya [57], the "*availability refers to the ability of a user's community to access the system—whether for submitting new work, updating or altering existing work, or collecting the results of the previous work. If a user cannot access the system, it is said to be unavailable*".

The different distributed computing paradigms aimed to provide a high level availability by eliminating the weakness of central point and by binning closer the service providing resources to the users which is very important especially to the services considered as critical requiring continuous availability.

First, as shown in Figure 9, the availability level can be directly seen in relationship with the resources ownership. In fact, volunteer computing provides the less level since the resources are not dedicated and they are being used by their authentic owners for their purpose too. Second, it has also a direct relationship with the geographical distribution of the resources. The more the resources are not located in one central location, the less probability of system's unavailability and/or failure is.

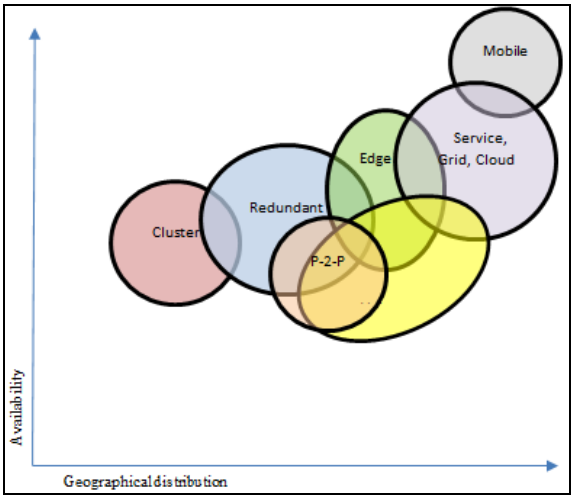

Figure 9. Resources Availability

D. *System's Scalability*

The system scalability, simply, means the possibility of the system to integrate a big and incremental number of computing resources when needed to be used by incremental number of users.

In [1], the authors defined it as "*how many systems can be reached from one node, how many systems can be supported, how many users can be supported, and how much storage can be used*".

As shown in the Figure 10, the distributed computing paradigms evolved from the cluster integrating tens to thousands of computers to Peer to Peer, volunteer, utility and now the mobile and pervasive computing integrating hundreds of thousands to millions of different types of computing resources like supercomputers, PCs, Mobile phones, etc.

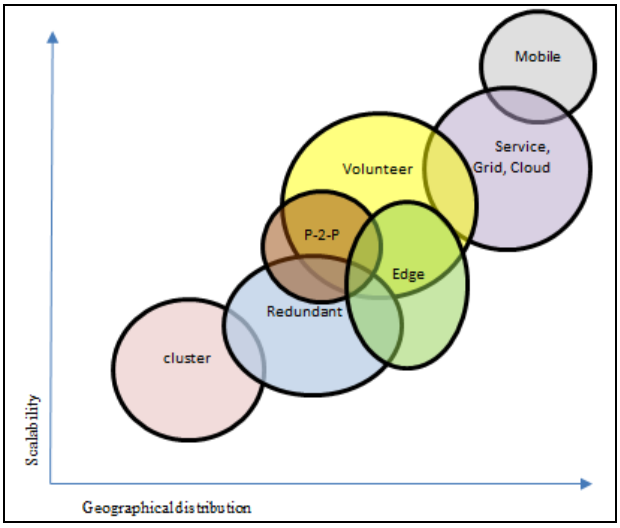

Figure 10. System's scalability

TABLE I. The table below (TABLE I.) summarizes the most important characteristics of some distributed computing paradigms.

TABLE I. COMPARING THE MOST IMPORTANT COMPUTING PARADIGMS

	Cluster	Volunteer	Grid	Cloud	Mobile
Scalability	★	★★★	★★★	★★★	★★★
Ownership	★★★	★	★	★★	★★★
Cost saving	★	★★★	★★★	★★★	★
Availability	★★★	★	★★	★★★	★★★
Performance	★	★★	★★	★★★	★★
Security	★★★	★	★★	★★	★★

5. DISTRIBUTED COMPUTING AS A SERVICE

Deliver computing power and storage capacity as a service to the users is a computing business model of distributed computing called utility computing (Figure 11). It is supported by technology that will enable companies to serve and consume IT resources and business functionalities as needed [58][49]. Its analogy is derived from the real world where service providers maintain and supply utility services, such as electrical power, gas, and water to consumers [54].

The utility computing is basically the grid computing and the cloud computing [1]. Computing resources, such as computation and storage, can be utilized internally by a company or packaged and exposed to the public as metered services [6].

When the system runs out of resources, using the utility model, it acquires those resources from another source (pay-as-you-use model) from a resources provider [49].

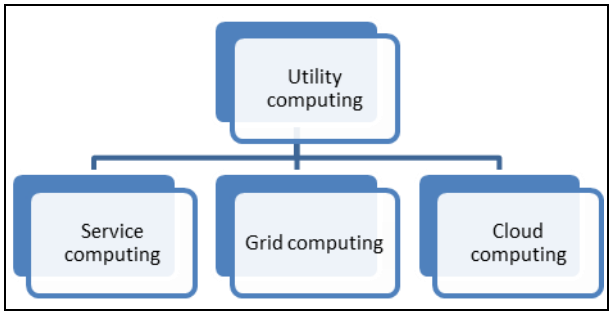

Figure 11. Utility computing paradigm

Therefore, the ability of utility computing to expand and contract ensures that an application can ride the hype curve to success [54]. Users (consumers) pay providers for using computing power only when they need to. All grid/cloud platforms are regarded as utility service providers. However, cloud computing offers a broader concept than utility computing [1].

A. *Utility vs. non-utility computing*

Utility computing and traditional non-utility computing are two different ways to do computing. The table below (TABLE II.) gives a brief comparison between the two paradigms.

TABLE II. GRID VS. CLUSTER COMPUTING

Utility computing	Non-utility computing
You don't have to own the computing infrastructure. You just need to rent one.	You have to own your private computing infrastructure (rooms, servers, network…), hire admins, etc.
A 'pay as you go' model is applied. The user has to pay only for the used resources.	Regardless the percentage of computing power, storage space, network bandwidth you actually use, fixed charges have to be paid.
Created for general purpose	Generally constructed to serve a specific application or project.
The users doesn't know the where the resources are located	Resources are located at one or some specific sites
Add, modify, and remove computing resources can be easily done "on-demand". Utility computing enables rapid elasticity.	It is not simple nor easy to add, modify or remove a computing resources
The high resource's virtualization enables excellent levels of isolation among the different users.	Offers poor isolation levels for the multiple users that might have conflicting objectives and storing their data on the same resources

The quality of service is guaranteed by a service level agreement between the service provider and the users.	The users are offered best effort quality of service which doesn't guarantee system's performance especially during peak of loads periods.
Only the service provider has access to the underlying infrastructure.	The user can has access to the physical computing infrastructure.
The user can design a whole computing infrastructure (servers, storage blocks, operating systems, software platforms, etc.) and deploy and run all applications he needs.	The user can execute specific applications on specific servers.

B. *Utility vs. grid computing*

In the report [49] issued by Connecticut-based Saugatuck Technology in 2004 indicated that IS managers have had difficulty understanding the differences between grid computing and utility computing. In fact, the utility model of computing can make use of grids to provide computing power to users and applications. They both:

- deal with finding and using computing resources
- can be deployed internally, or can make use of external resources
- helps enterprises to reduce computing costs.

Despite these resemblances, there are significant differences between grid computing and utility computing presented in the table below (BTABLE III.).

TABLE III. UTILITY VS. GRID COMPUTING

Utility computing	Grid computing
A model used to provide access to computing resources when an application runs out of computing resources	a distributed computing environment designed to find and exploit unused computing resources
utility computing relies heavily on accounting for resource utilization	most of the time used by research teams to share computing resources
when the system is out of computing power or storage, the utility model "acquires" additional resources to finish the jobs (often at a billed cost)	When a grid runs out of resources, it is out of resources.

C. *Utility vs. cloud computing*

Although utility computing and cloud computing are considered to be the same, the differences between them are important (TABLE IV.). In fact, utility computing refers to the ability to deliver the computing power just like other public utility services such as electricity. It is a business model proposed to meter the offered services and charge customers for usage.

TABLE IV. UTILITY VS. CLOUD COMPUTING

Utility computing	Cloud computing
relates to the business model in which computing resources and are delivered	A specific technology related to the way we design, build, deploy and run applications
Doesn't require resource virtualization	Virtualization is a key factor and different levels are offered
Might be restricted to specific networks	accessible through the Internet network
Can be applied for different scales	Used for big scale infrastructures and applications for economic efficiency

While cloud computing offers, in addition to the accounting service, a virtualized environment for sharing and dynamically acquire new resources to guarantee higher scalability and reliability levels.

D. *Grid vs. cloud computing*

Since they are both kind of distributed and utility computing models, Grid and cloud computing have many common points Ian Foster and al. [6]. They both:

- Reduce the cost of computing resources and storage for the enterprises and research projects
- Increase the system's reliability and fault tolerance; especially for the cloud computing.
- Increase flexibility by enabling the users to access new resources on the fly and giving the illusion of infinite computing power
- Massively scalable to deal with the huge number of computing resources they are interconnecting. They have both a common need to manage large facilities.
- Enables to implement highly parallel applications to serve problems that need massive computing power
- Can be encapsulated as an abstract entity that delivers different levels of service. The whole grid or cloud can be considered as one supercomputing resource

However, despite these similarities, grid and cloud computing differs from each other in many other points (TABLE V.).

TABLE V. GRID VS. CLOUD COMPUTING

Grid computing	Cloud computing
Focuses on computing power in terms of processing cycles as the main shared resource to execute the jobs (usually batch execution that often has a known/predictable start and finish time)	Usually used for analysis of massive data and mainly handles provisioning of user-driven services that are less predictable in terms of resource needs [59]
Reconfiguration of the different components is required	Resources, applications, and services can be dynamically configured (via virtualization or other approaches) and delivered on demand
Is tasks processing and user's jobs centric even though it can run and support web services	is being web-centric [60] with adoption of Services Computing and Web 2.0 apps
Is driven by the need to exploit of the underutilized resources and share them especially in scientific research domain	Is driven by economies of scale. Billions of dollars being spent by companies to create real commercial large-scale systems with hundreds of thousands of computers
The resources are abstracted and virtualized to be offered to the different users and applications as services	Enables resources abstraction and virtualization but at much higher level [61] offering three virtualization levels SaaS, PaaS, and IaaS

As a conclusion, the Figure 12 summarizes the most important characteristics of the utility computing paradigms: service, grid, and cloud.

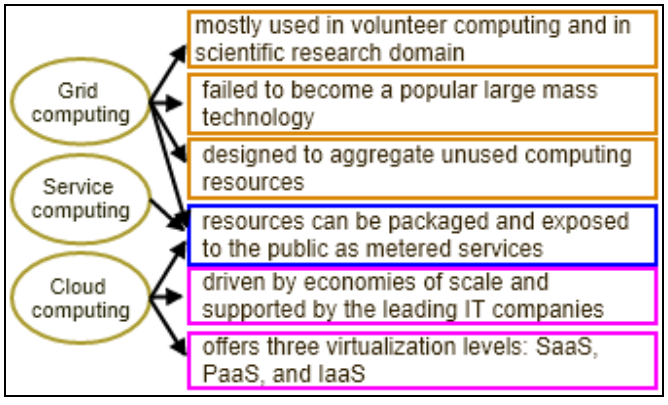

Figure 12. Comparing the utility computing paradigms

6. CONCLUSION

This paper reviewed the most important paradigms related to distributed computing and analyzed these paradigms according to different factors. It can be considered as a brief road map that would be useful for researchers, students, and commercial users. It consists of an introductory knowledge that helps them get an idea about the past and the future emerging computational technologies.

The idea is that the evolution of the distributed computing paradigms does not involve the vanish of the older ones. In fact, if one looks carefully the development of cloud computing, as an example, "swallowed" the grid and service computing paradigms.

The cloud users can easily design and run a whole grid infrastructure based on virtual cloud resources. Applications running on the grid or the cloud infrastructures can provide any kind of services to the public based on service computing. A good example is presented by N. Fallenbe et al. [62] who tried to merge grid and cloud computing paradigms and "enables users to set up their own VMs to be used as Grid job execution environments".

In the same way, all the rest of distributed computing paradigms and computing models still exist but in different forms. The users can use the cloud computing services to design computing infrastructures like clusters, pools, grids. But they can use their existing infrastructures to deploy private and public clouds. They can, also, share their computing resources voluntarily and contribute in the research efforts using volunteer computing power.

In real world projects, the different distributed computing paradigms coexist in order to serve a specific goal or fulfill a user's need in the best way. It is u to the solution's developer to make the necessary choices.

Few predictions can be made based on the belief that the economics of computing is leading big number of academic researchers and industrial giants to develop new computing paradigms. Many efforts are still needed to satisfy the users' "hunger" for computing power, mobility, scalability, availability, reliability, and efficiency.

REFERENCES

[1] B. Kahanwal, "The Distributed Computing Paradigms : P2P , Grid , Cluster , Cloud , and Jungle," *Int. J. latest Res. Sci. Technol.*, vol. 1, no. 2, pp. 183–187, 2012.

[2] M. Weiser, "Ubiquitous computing," in *ACM Conference on Computer Science*, 1994, p. 418.

[3] C. Engineering and B. Bili, "Comparison of Cluster , Grid and Cloud Computing using Three Different Approaches," pp. 1–4, 2015.

[4] D. Sood, "Survey of Computing Technologies : Distributed , Utility , Cluster , Grid and Cloud Computing," vol. 6, no. 5, pp. 99–102, 2016.

[5] M. Singh, "A Comparative Study on Grid Computing and Cloud Computing," no. June, pp. 708–713, 2015.

[6] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," *2008 Grid Comput. Environ. Work.*, pp. 1–10, Nov. 2008.

[7] S. M. Hashemi and A. K. Bardsiri, "Cloud Computing Vs . Grid Computing," vol. 2, no. 5, pp. 188–194, 2012.

[8] E. Hwang, S. Kim, T. Yoo, J.-S. Kim, S. Hwang, and Y. Choi, "Performance Analysis of Loosely Coupled Applications in Heterogeneous Distributed Computing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9219, no. c, pp. 1–1, 2015.

[9] B. Sawicki, "Reliable peer-to-peer computing system," no. 5, pp. 61–63, 2014.

[10] D. A. Prathibha, "Issues in adapting cluster , grid and cloud computing for HPC applications," vol. 2, no. 1, pp. 12–16, 2014.

[11] J. Melorose, R. Perroy, S. Careas, M. Anjomshoa, M. Salleh, and M. P. Kermani, "A taxonomy and survey of distributed computing systems," *J. Appl. Sci.*, vol. 15, no. 1, p. 46, 2015.

[12] K. Skala, E. Afgan, and Z. Sojat, "Scalable Distributed Computing Hierarchy : Cloud , Fog and Dew Computing," *Open J. Cloud Comput.*, vol. 2, no. 1, pp. 16–24, 2015.

[13] B. Barney, "Introduction to parallel computing," *Lawrence Livermore Natl. Lab.*, vol. 6, no. 13, p. 10, 2010.

[14] E. Yiannis and G. Joseph, "Introduction to Grid'5000," 2011.

[15] M. Estep, F. Moinian, J. Carroll, and C. Zhao, "Methods for Teaching a First Parallel Computing Course to Undergraduate Computer Science Students."

[16] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*. 2003.

[17] F. Berman and R. Wolski, "Scheduling from the perspective of the application," *Proc. 5th IEEE Int. Symp. High Perform. Distrib. Comput.*, 1996.

[18] R. R. Schaller, "Moore's law: past, present and future," *IEEE Spectr.*, vol. 34, no. 6, 1997.

[19] W. Aspray, *John von Neumann and the origins of modern computing*, vol. 191. Mit Press Cambridge, MA, 1990.

[20] H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobb's J.*, pp. 1–9, 2005.

[21] H. Suleman, "Utility-based High Performance Digital Library Systems," *Proc. Second. Very Large Digit. Libr. VLDL2009 A Work. conjunction with Eur. Conf. Digit. Libr. 2009*, 2009.

[22] R. Friedman, K. P. Birman, S. Keshav, and W. Vogels, "Reliable time delay-constrained cluster computing." Google Patents, 21-May-2002.

[23] Y. Li and Z. Lan, "Exploit failure prediction for adaptive fault-tolerance in cluster computing," *Sixth IEEE Int. Symp. Clust. Comput. Grid, 2006. CCGRID 06*, no. June 2015, pp. 531–538, 2006.

[24] M. Li, D. Goldberg, W. Tao, and Y. Tamir, "Fault-tolerant cluster management for reliable high-performance computing," in *Int. Conf. on Parallel and Distributed Computing and Systems*, 2001, no. August, pp. 480–485.

[25] D. S. Milojicic *et al.*, "Peer-to-peer computing." 2002.

[26] M. Arumugam, A. Sheth, and I. B. Arpinar, "Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web," *Knowl. Creat. Diffus. Util.*, vol. Master of, p. 51, 2002.

[27] L. L. C. Napster, "Napster," *URL http//www. napster. com*, 2001.

[28] H. Barkallah, M. Gzara, and H. Ben Abdallah, "A fully distributed Grid meta scheduling method for non dedicated resources," in *International Conference on Parallel and Distributed Processing with Applications (ICPDPA'2014), IEEE WCCAIS'2014 Congress*, 2014, no. 1, pp. 1–6.

[29] K. Reed, "Reducing Heterogeneity in Volunteer Computing using Virtual Machines," 2008.

[30] L. P. Chen, J. A. Lin, K. C. Li, C. H. Hsu, and Z. X. Chen, "A scalable blackbox-oriented e-learning system based on desktop grid over private cloud," *Futur. Gener. Comput. Syst.*, vol. 38, pp. 1–10, 2014.

[31] M. I. Syed A. Ahson, *Cloud Computing and Software Services*. 2010.

[32] R. Riesen, K. Ferreira, J. Stearley, R. Oldfield, J. H. L. Iii, and R. Brightwell, "Redundant Computing for Exascale Systems," *Sandia Natl. Lab.*, 2010.

[33] S. Ghosh, *Distributed Systems: An Algorithmic Approach*. Iowa, 2007.

[34] H. Pang and K. L. Tan, "Authenticating query results in edge computing," *Proc. - Int. Conf. Data Eng.*, vol. 20, pp. 560–571, 2004.

[35] A. Davis, J. Parikh, and W. E. W. Weihl, "Edgecomputing: extending enterprise applications to the edge of the internet," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, 2004, pp. 180–187.

[36] C. Canali, M. Rabinovich, and Z. Xiao, "Utility computing for Internet applications," in *Web Content Delivery*, Springer, 2005, pp. 131–151.

[37] G. Roussos and V. Kostakos, "rfid in pervasive computing: State-of-the-art and outlook," *Pervasive Mob. Comput.*, vol. 5, no. 1, pp. 110–131, 2009.

[38] P. Romano, F. Quaglia, and B. Ciciani, "Design and evaluation of a parallel edge server invocation protocol for transactional applications over the Web," *Proc. - 2006 Int. Symp. Appl. Internet, SAINT 2006*, vol. 2006, pp. 206–209, 2006.

[39] R. Want, B. N. Schilit, and Jenson Scott, "Enabling the Internet of Things," 2011.

[40] M. Weiser, "Some computer science issues in ubiquitous computing," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, p. 12, 1999.

[41] M. Weiser, R. Gold, J. S. Brown, B. Sprague, and R. Bruce, "The origins of ubiquitous computing research at PARC," *IBM Syst. J.*, vol. 38, no. 4, pp. 693–696, 1999.

[42] D. Saha and A. Mukherjee, "Pervasive computing: A paradigm for the 21st century," *Computer (Long. Beach. Calif).*, vol. 36, no. 3, p. 25–31+4, 2003.

[43] K. Lyytinen and Y. Yoo, "The shift toward ubiquitous computing poses multiple novel technical, social, and organizational challenges. capability," *Commun. ACM*, vol. 45, no. 12, pp. 643–4, 2002.

[44] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, 2001.

[45] D. Meenakshi and S. Thirunavukkarasu, "An effective cluster score job scheduling algorithm for grid computing," *Int. J. Appl. Eng. Res.*, vol. 9, no. 22, pp. 7232–7236, 2014.

[46] Z. Wu, S. Deng, and J. Wu, *Service Computing: Concept, Method and Technology*. Academic Press, 2014.

[47] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, 2008, pp. 5–13.

[48] A. Jain and R. Singh, "An Innovative Approach of Ant Colony Optimzation for Load Balancing in Peer to Peer Grid Enviomment," pp. 1–5, 2014.

[49] J. Clabby and C. Analytics, "The Grid report, 2004 edition," Technical report, Clabby Analytics, 2004.

[50] F. Dong and S. G. Akl, "Scheduling Algorithms for Grid Computing : State of the Art and Open Problems," pp. 1–55, 2006.

[51] Y. Zhu and L. M. Ni, "A survey on grid scheduling systems," *Department of Computer Science, Hong Kong University of science and Technology*, vol. 32. pp. 1–47, 2003.

[52] M. J. Kim, "Resource virtualization and optimization via Grid and Cloud Computing."

[53] H. Barkallah, M. Gzara, and H. Ben Abdallah, "Dynamic and adaptive topology-aware load balancing for Grids," in *FCST'2014*, 2014.

[54] S. Fiore, G. Aloisio, and G. A. Sandro Fiore, *Grid and Cloud Database Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[55] F. J. Seinstra *et al.*, "Jungle computing: Distributed supercomputing beyond clusters, grids, and clouds," in *Grids, Clouds and Virtualization*, Springer, 2011, pp. 167–197.

[56] N. Drost *et al.*, "High-performance distributed multi-model / multi-kernel simulations: A case-study in jungle computing," *Proc. 2012 IEEE 26th Int. Parallel Distrib. Process. Symp. Work. IPDPSW 2012*, pp. 150–162, 2012.

[57] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing: Principles and Paradigms*. John Wiley & Sons, 2011.

[58] A. Mendoza, *utility computing technologies, standards, and strategies*. .

[59] S. Disaggregation, *Dynamic Cloud Resource Management*. 2014.

[60] A. Marinos and G. Briscoe, "Community cloud computing," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5931 LNCS, pp. 472–484, 2009.

[61] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," *2014 IEEE Int. Adv. Comput. Conf.*, pp. 866–872, Feb. 2014.

[62] N. Fallenbeck, M. Schmidt, R. Schwarzkopf, and B. Freisleben, "Inter-site virtual machine image transfer in grids and clouds," in *Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010, pp. 1–19.

B. Haitham received a master degree in Computer Science and multimedia from the University of Sfax, Tunisia. Currently, he is a Ph.D. student and a member of the Multimedia, Information systems and Advanced Computing Laboratory (Mir@cl), University of Sfax. His research interests include grid and cloud computing, scheduling and load balancing optimization.

M. Gzara received a PhD in automatic and industrial computing from the University of Lille 1, France. She is currently an Associate Professor in Computer Science at the Higher School of Computer Science and Mathematics of Monastir, University of Monastir, Tunisia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory (Mir@cl), University of Sfax. Her research interests include data mining techniques, Optimization, Parallelization, Distributed Compuattion and Information Retrieval.

H. Ben-Abdallah received a BS in Computer Science and BS in Mathematics from the University of Minnesota, MPLS, MN, a MSE and PhD in Computer and Information Science from the University of Pennsylvania, Philadelphia, PA. She worked at University of Sfax, Tunisia from 1997 until 2013. She is now full professor at the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory (Mir@cl), University of Sfax. Her research interests include software design quality, reuse techniques in software and business process modeling.