Noname manuscript No.
(will be inserted by the editor)

Distributed Computing in the 21st Century: Some Aspects of Cloud Computing

Fabio Panzieri · Ozalp Babaoglu · Stefano Ferretti · Vittorio Ghini · Moreno Marzolla

Received: date / Accepted: date

Abstract The Cloud Computing paradigm has gained considerable attention owing to the notable commercial success of many Cloud service providers. Typically, a Cloud Computing service provides its customers with resources as a utility, using a pay-as-you-go model. Thus, Cloud Computing customers can reduce costs related to the acquisition and management of complex IT infrastructures, and Cloud service providers can make efficient use of large resource pools by consolidating multiple variable workloads. From the providers point of view a Cloud is a very large distributed system which poses many challenges,

F. Panzieri
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Tel.: +39-051-2094508
E-mail: panzieri@cs.unibo.it

O. Babaoglu
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Tel.: +39-051-2094504
E-mail: babaoglu@cs.unibo.it

S. Ferretti
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Tel.: +39-051-2094845
E-mail: sferrett@cs.unibo.it

V. Ghini
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Tel.: +39-051-2094846
E-mail: ghini@cs.unibo.it

M. Marzolla
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Tel.: +39-051-2094847
E-mail: marzolla@cs.unibo.it

including monitoring, management, efficient resource sharing, fault-tolerance and so on. The amount of knowledge and experience acquired in the development of distributed systems can be used to address some of these issues, while other problems pose new challenges that need new solutions. In this paper we introduce our approach to Cloud Computing and summarize recent results we have obtained by applying this approach to the solution of some critical problems in the Cloud Computing field.

Keywords Cloud Computing · Quality of Service · Virtualization · Peer to Peer Systems

Foreword (by Fabio Panzieri)

One of the most important lessons I have learned from Brian Randell is that, in order to master and control the complexity inherent in any computing system and to address effectively the variety of performance/reliability/security trade-off issues involved in its design, it is crucial to impose a structure on that system so as to study its components and their properties in isolation, and infer the properties of the whole from those of the parts. The principles behind this lesson have been elegantly summarized by John Rushby in (Rushby 2011); this paper is a testimony of the application of Brians lesson to the field of Cloud Computing by a group of colleagues of mine and I, in my Department.

1 Introduction

The Cloud Computing paradigm originates mainly from research on distributed computing and virtualization, as it is based on principles, techniques and technologies developed in these areas.

Although there may still be some confusion as to what exactly Cloud Computing means, and no general consensus on a definition for Cloud Computing has been reached (Armbrust et al 2010; Zhang et al 2010), for the scope of this paper we shall adopt the informal definition of Cloud Computing proposed in Mell and Grance (2011) and reported below:

> Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

In Mell and Grance (2011) the above definition is extended with a discussion of the essential characteristics a Cloud Computing environment is to possess, and both the service and the deployment models it can implement. In summary, those characteristics include:

1. on-demand self service: the ability to provision computing capabilities (e.g. CPU time, network storage) dynamically, as needed, without human intervention;
2. broad network access: computing capabilities can be accessed through the network by (thin or thick) client platforms using standard mechanisms;
3. resource pooling: virtual and physical resources can be pooled and assigned dynamically to consumers, according to their demand, using a multi-tenant model;
4. elasticity: capabilities can be provisioned dynamically in order to enable a customer application to scale out quickly, and can be dynamically released in order to enable it to scale in (in general, the customer perceives unlimited capabilities that can be purchased in any quantity at any time);
5. measured service: Cloud resource and service usages are optimized through a pay-per-use business model, and are to be monitored, controlled and reported transparently to both their customer and provider.

The service models define the level of abstraction at which a cloud customer interfaces a Cloud Computing environment. These consist of the Software as a Service (SaaS) model, the Platform as a Service (PaaS) model, and the Infrastructure as a Service (IaaS) model. In essence, in a SaaS Cloud, the capabilities provided to a Cloud customer are application services running in the Cloud infrastructure. The Cloud customer has no control over the Cloud infrastructure. "Google apps" is probably the first example of a widely used SaaS Cloud. Usually, all a user needs in order to access and use a SaaS Cloud is a browser. For this reason, one of the leading companies in delivering SaaS applications–Salesforce.com–coined the phrase "the end of software" to emphasize this service delivery approach. In contrast, the capabilities provided by a PaaS Cloud consist of programming languages, tools and a hosting environment for applications developed by the cloud customer. The difference between the SaaS and PaaS models is that while the user of a SaaS Cloud simply utilizes an application that runs in the cloud, the PaaS Cloud user develops an application service (often via browser) that can be executed in the Cloud and made available to service customers; the application service development is carried out using libraries, APIs and tools possibly offered by some other company. Examples of PaaS solutions are AppEngine by Google, Force.com from SalesForce, Microsoft's Azure and Amazons Elastic Beanstalk. Finally, a IaaS Cloud provides its customers with fundamental computing capabilities such as processing, storage and networks where the customer can run arbitrary software, including operating systems and applications. The number of companies offering such kind of services is continually growing; one of the earliest being Amazon with their EC2 platform.

The deployment models define the mode of operation of a Cloud infrastructure; these are the Private Cloud, the Community Cloud, the Public Cloud, and the Hybrid Cloud models. A Private Cloud infrastructure is operated exclusively for a customer organization; it is not necessarily managed by that organization. In the Community Cloud model the infrastructure is shared by sev-

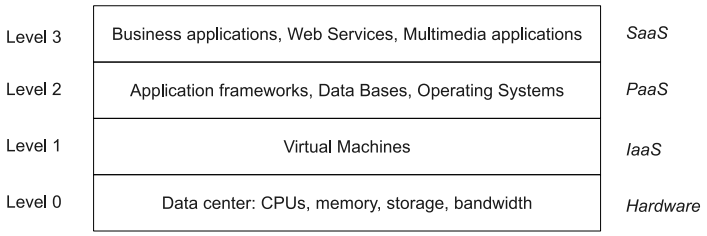

Fig. 1 Cloud Service Model Structuring

eral organizations and supports a specific community with common concerns (e.g. security requirements, policy). In the Public Cloud model the infrastructure is made available to the general public and is owned by an organization selling Cloud services. The Hybrid Cloud model refers to Cloud infrastructures constructed out of two or more private, public or community Clouds. These may remain unique entities but must enable data and application portability (e.g., for load balancing purposes) across separate Cloud infrastructures.

It can be observed that the above definition of Cloud Computing, in addition to capturing and summarizing the fundamental scope of this paradigm, refers implicitly to one of the principal opportunities that Cloud Computing offers; namely, the separation between the process of constructing an infrastructure for service provision and that of providing end user services. This opportunity enables the existence of at least three basic categories of Cloud Computing providers and users: (i) the provider of essential Cloud Computing resources, (ii) the provider of services implemented using these resources, and (iii) the customer of these services.

Further, one can observe that the service models described above can be thought of as structured in four hierarchical levels of abstraction, as depicted in Figure 1.

Level 0 in Figure 1 consists of the data centers containing the Cloud physical resources. Level 1 (IaaS level) is responsible for instantiating and maintaining a pool of storage and computing resources using virtualization technologies, such as VMware, Xen and KVM (VMware 2010; Xen 2010). Level 2 (PaaS level) consists of application platforms deployed within the resources available at Level 1. Finally, Level 3 maintains actual Cloud applications.

The development of Internet-based, data-intensive applications, such as those for e-commerce (e.g., Amazon, eBay) and social networking (e.g., Facebook, Twitter), as well as the system support these applications require (e.g., large scale file systems such as the Google File System) are generating a constantly increasing demand for computing and communication resources.

The Cloud Computing paradigm can respond effectively to this demand, provided that a number of limitations affecting (the current instantiations of) this paradigm can be overcome. For example, there are no standard interfaces to the hierarchical architecture levels of Figure 1, as yet, even though notable efforts are being made in order to define adequate Cloud Computing standards (Cloud Standards Wiki 2010). Presumably, this is because there are

few business incentives to adopt such interfaces. However, as Cloud Computing evolves and mission critical applications and services emerge that require guaranteed availability and business continuity, the use of multiple, interoperating Clouds may become a necessity, in order to rule out the possibility that a single point of failure (see below) compromise that requirement. Within this scenario, the incentives to adopt standard interfaces that facilitate integration and inter-operation of Could services are likely to grow.

Note that, in the above scenario, the "single point of failure" we have mentioned may well be a Cloud provider that owns multiple and geographically distributed data centers. As these data centers belong to a single company, the possibility that that company go bankrupt makes them a "single point of failure" for customers, regardless of their physical distribution. As of today, a number of projects are investigating solutions that meet the high availability and business continuity requirements mentioned above through the provision of support for Cloud federations, e.g., Reservoir (Rochwerger et al 2009), Open Cirrus™ (Avetisyan et al 2010), InterCloud (Buyya et al 2010), Contrail (Contrail 2010).

Further Cloud Computing limitations, and the relative countermeasures necessary to overcome them, are thoroughly discussed in the already cited reference (Armbrust et al 2010); hence, we shall not examine them here. Rather, in the rest of this paper, we propose and discuss an alternative approach to the structuring of Cloud Computing architectures, and introduce the results of some experiments we carried out based on that structuring approach.

The rest of this paper is organized as follows. In Section 2 we introduce our architectural structuring approach. In Section 3 we summarize the state of the art in Cloud Computing within the limits of four topics of interest for the scopes of our discussion; these topics are: Cloud federations, Quality of Service (QoS) in Clouds, Peer-to-Peer (P2P) Clouds, and virtualization techniques. In Section 4 we discuss our experimental work. Specifically, in this Section we will describe our experimental work on energy saving in Cloud Computing environments through server consolidation (Subsection 4.1), the provision of QoS support in a Cloud environment (Subsection 4.2), and the use of Cloud Computing resources from mobile applications (Subsection 4.3). Finally, in Section 5 we provide some concluding remarks.

2 Cloud Computing as seen from Bologna

Cloud Computing is rapidly growing in popularity as attested by the success of commercial Cloud platforms such as Amazon EC2, Google App Engine and Windows Azure. Each of these platforms is instantiated in a number of large datacenters owned by a single organization. We term them centralized platforms as, owing to the single ownership feature mentioned above, they are vulnerable to the single point of failure risk introduced earlier, as in a centralized system. Centralized Clouds exhibit a number of shortcomings, including the following:

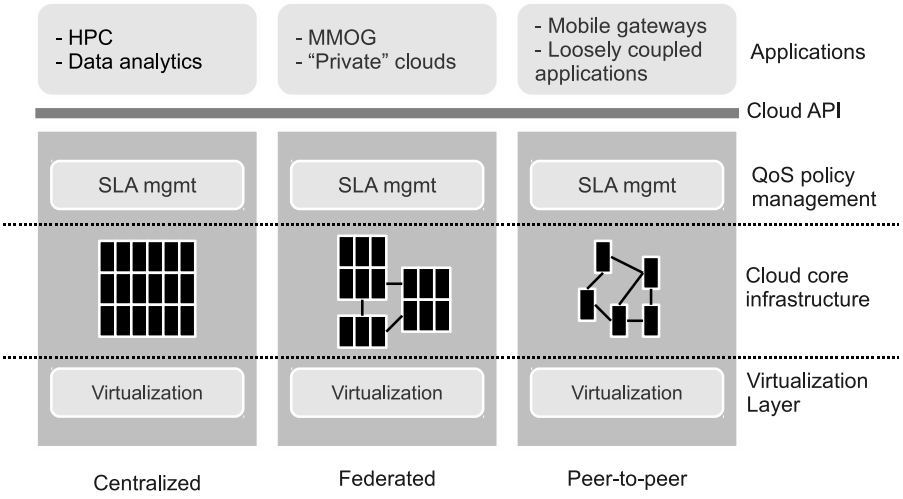

Fig. 2 Cloud Computing Dimensions

1. their creation and maintenance require notable investments,
2. their large size makes them complex to manage,
3. compute and storage services are offered on a best effort basis, ignoring problems such as QoS provisioning and Service Level Agreement (SLA) negotiation.

Building a centralized Cloud is not the only option (and in some cases, it might not be even an optimal one) to the provision of Cloud Computing services. Specifically, we can consider a whole spectrum of possible Cloud architectures, where the centralized approach is just one of several possibilities. In Figure 2 we examine Cloud Computing architectures along two dimensions. On the horizontal dimension, we distinguish between three different architectures: namely, Centralized, Federated and Peer-to-peer architectures. On the vertical dimension, we identify the principal functionalities used to build Cloud services.

As mentioned above, centralized Clouds constitute the current commercial offerings. Applications for scientific computation, data mining, and Web Services are typically hosted in a centralized Cloud.

Federated Clouds, where multiple Clouds are integrated to build a larger Cloud, are a logical evolution of the centralized approach. This is an interesting alternative for those companies who are reluctant to move their data to a Cloud provider, owing to security and confidentiality concerns. By operating on geographically distributed datacenters, companies can still benefit from the advantages of Cloud Computing (e.g., high availability, scalability, pay-per-use business model) by maintaining their data at their own resources, and federating them into a larger Cloud. Multimedia entertainment is another example where Cloud federations may be appropriate. For example, in the case of Massive Multiplayer Online Games (MMOGs), a large number of users

interact in a virtual space that must be handled meeting strict QoS requirements. Multiple MMOG servers can be operated on geographically distributed Clouds in order to automatically balance the load; all the server instances can be federated to maintain a coherent game state (Marzolla et al 2011b).

Finally, by stretching the idea of federated Clouds to the extreme, we can build a Cloud out of independent resources that are opportunistically assembled. We term these Clouds "P2P Clouds". These can be built by assembling individual peers without any central monitoring and management components. Gossip based epidemic protocols can be used to monitor the Cloud, handle churn and allocate resources. P2P Clouds can enable provisioning of resources at low or zero cost; applications that can be conveniently run in such Clouds include loosely-coupled applications, and distributed applications where the physical location of nodes is important.

The vertical dimension of Figure 2 represents the functionalities that are used or provided. A lightweight, low-level Virtualization Layer is used to build the Cloud infrastructure. The QoS policy management layer implements negotiation and SLA enforcement between the customers and the Cloud service provider. It is worth observing that, in general, different Cloud architectures will require specific virtualization features and SLA negotiation mechanisms. In addition, each architecture may or may not be able to provide QoS guarantees. Note that, as we move from left to right in Figure 2, the core architecture becomes increasingly less reliable so that progressively looser QoS guarantees are provided.

On top of the QoS policy management layer we have a Cloud API which allows users to interact with the different Cloud implementations using a common set of operations. The Cloud API should provide an interface for resource negotiation, allocation and monitoring, regardless of the specific Cloud architecture. Each architecture has its advantages and disadvantages; hence, there are applications that are more suited for one architecture than the others.

It is not necessary that Cloud customers be fully aware of the architecture they are connecting to; what really matters is the SLA that is negotiated between customers and the service providers. If a P2P Cloud is built on top of resources of unknown reliability such as nodes that could be switched on and off at any time then the P2P Cloud would be able to guarantee very loose (if any) QoS requirements, so that demanding applications will be rejected due to SLA negotiation failures. Needless to say, it is possible to build a P2P Cloud on top of reliable resources; in this case the middleware could provide greater QoS guarantees.

3 State of the art

In view of the above discussion, in this Section we summarize the state of the art in Cloud Computing, relative to the following four topics: (i) Cloud Federation, (ii) QoS in Clouds, (iii) P2P Clouds and (iv) Virtualization Techniques.

For a full state of the art assessment on Cloud Computing the interested reader can refer to Armbrust et al (2010); Zhang et al (2010).

3.1 Cloud Federation

Cloud federation, sometimes referred as "cloud of clouds" or "InterCloud" (the term was conied by Cisco), is a relatively new research topic in the Cloud Computing area. Regardless of how one names it, the integration of separate and independently owned and administered Clouds in a federation of Clouds poses many research challenges. Examples of such challenges include (i) the ability to support sudden workload spikes by dynamically leasing computational and storage capabilities from the federated Cloud service providers, and (ii) the ability to negotiate SLA contracts with the application service providers. Related work that discusses, under different perspectives and contexts, issues of development of Cloud federations are introduced below

In (Buyya et al 2010) a service-oriented architectural framework for utility-oriented federation of Cloud Computing environments is presented. In the vision of the authors the goal of this framework is that of facilitating just-in-time, opportunistic, and scalable provisioning of application services, consistently achieving QoS targets under variable workload, resource and network conditions. To this end, the infrastructure described in this paper supports (i) dynamic expansion or contraction of capabilities (Virtual Machines (VMs), services, storage, and database) for handling sudden variations in service demand; (ii) negotiation of SLA contracts and (iii) autonomic provisioning of QoS aware services across different Cloud providers while minimizing service costs.

Reservoir (Rochwerger et al 2009) is a European Union funded project whose aim is to enable massive scale deployment and management of complex IT services across different administrative domains, IT platforms and geographies. Essentially, the Reservoir project aims to support the emergence of Service-Oriented Computing as a new computing paradigm. In this paradigm, services are software components exposed through network-accessible interfaces that enable the composition of complex distributed applications out of loosely coupled components. In the Reservoir vision, the infrastructure that should support this paradigm is referred as federated Cloud.

Open Cirrus^{TM} (Avetisyan et al 2010) is a testbed of distributed data centers for Cloud Computing research. It provides interesting features which cannot be found in other testbeds; e.g., the federation of heterogeneous sites for the deployment of research systems and applications. In addition, Open Cirrus provides an open stack with non-proprietary APIs for Cloud Computing.

Finally, the scope of the Contrail project (Contrail 2010) is to develop an open source system that enables the construction and maintenance of Cloud Federations. In particular, Contrail aims to build a tightly integrated, open source software stack that includes a broad set of system, runtime and high level services that provide Cloud customers with standard interfaces for the support of cooperation and resource sharing over Cloud federations.

Further important issues in Cloud federations include the development of: (i) techniques for resource usage accountability in a pay-as-you-go federated Cloud model, and (ii) methods and tools that ensure organizations providing services constructed out of multiple, third party, Cloud-based components that the composed service is meeting the customer availability and responsiveness requirements. Technological solutions to this latter problem can be based on lightweight virtualization mechanisms and suitable algorithms for load distribution and data migration; these algorithms will have to take into account the physical location of data and computations in order to enhance the performance of applications running in the federated infrastructure.

3.2 QoS in Clouds

To the best of our knowledge, current Cloud technology is not fully tailored to manage QoS requirements and to honor possible SLAs, although both the industrial and the academic research communities are showing growing interest on issues of QoS assurance within the context of Cloud Computing. In general, honoring an SLA requires an accurate assessment of the amount (and characteristics) of the needed resources. Application services hosted in Clouds (e.g., Web applications, Web services) are often characterized by high load variance; hence, the amount of resources needed to honor their SLAs may vary notably over time.

As of today, in order to ensure that an application SLA is not violated, a resource overprovision policy is often adopted. This requires evaluating (either through application modeling or through application benchmarking) all possible resources a hosted application can require in the worst case, and then statically allocating these resources to that application. This policy can lead to a largely suboptimal utilization of the hosting environment resources. In fact, being based on a worst-case scenario, a number of allocated resources may well remain unused at run time. This limitation can be overcome by developing a middleware architecture that can be integrated in a Cloud Computing platform so as to manage dynamically the Cloud configuration, and honor the SLAs of the applications that platform hosts. We have developed one such architecture; some preliminary results we have obtained from our development are discussed in (Ferretti et al 2010) and summarized in Section 4.2. In the following paragraphs we briefly review a few papers on this topic that show some analogy with our approach.

Li et al (2009) describe a method for achieving resource optimization at run time by using performance models in the development and deployment of the applications running in the Cloud. This approach is based on a so-called Layered Queueing network performance Model (LQM, a kind of extended queueing network model) that predicts the effect of simultaneous changes (e.g., resource allocation/deallocation) to many decision variables (throughputs, mean service delays, etc.). Moreover such a model seems to take no account of variables that could heavily affect the performance of the applications hosted in the

Cloud. Examples of such variables include: (i) possible bottlenecks produced by load balancers, databases, and applications replication; (ii) the allocation times required for the introduction of a new VM in the virtual pool; (iii) the time elapsed from the introduction of the new VM in the virtual pool until it reaches a steady state.

Nathuji et al (2010) observe that the performance of an application hosted in a Cloud environment can be heavily affected by the existence of other virtual machines hosting other applications on a shared server. The authors present "Q-Cloud", a QoS-aware management framework that should ensure that the performance experienced by the applications is the same as they would have achieved if there was no performance interference. Q-Clouds rely upon a multi-input, multi-output (MIMO) model that captures performance interference interactions and describes the relationship between resource allocations and the QoS experienced by VMs.

Litoiu et al (2010) proposes a Cloud optimization infrastructure structured in layers, following the Cloud architecture depicted in Figure 1. A feedback adaptive loop based on model, estimation and prediction techniques specific to each layer allows the optimization component to take proactive steps to provision more hardware and software resources to avoid loss of revenue or users.

3.3 P2P Clouds

Implementing a Cloud infrastructure on top of a P2P network is still an open problem, which has not been fully investigated as yet. The BOINC project (Anderson 2004) coined the term "volunteer computing" to denote the possibility for PC owners to contribute to a public resource pool, which then could be used by scientific projects. In addition, there are examples of commercial offerings in the area of P2P, distributed storage space (Wuala 2010).

It should be observed that existing "volunteer computing" systems such as BOINC rely on centralized components. Maintaining the central services represents both a technical burden and a single point of failure. Thus, there is scope for investigating the development of a fully decentralized P2P Cloud as first proposed by Babaoglu et al (2006b).

In particular, the current state of the art can be advanced by providing a decentralized system offering computational and storage resources, such that the system can self-assemble and self-manage; we believe that this can be achieved by applying epidemic and gossip-based protocols for information dissemination and aggregation (Babaoglu et al 2006a). Avoiding central services allows to reduce further the cost of deploying a P2P Cloud, making it a viable solution for very low cost computing.

3.4 Virtualization techniques

Many implementations of IaaS Clouds use well known virtual machines or paravirtualization technologies. Eucalyptus (Nurmi et al 2009) supports both Kvm and Xen as its machine virtualization infrastructure, Amazon EC2 uses Xen. Clouds are often based on virtual resources assigned in an automatic and scalable manner. Virtualization for Cloud Computing has the dual purpose to customize the user environment and to confine the user activity for security reasons.

Partial virtualization approaches Gardenghi et al (2008) are promising methods to provide a light virtualization environment for Cloud Computing. In fact, partial virtualization runs at user level and needs neither administration privileges to run, nor the bootstrapping of an OS or an emulated main memory. This support seems appropriate especially for a peer-to-peer virtualization scenario where a community of users can create a Cloud by sharing some of their resources just by running a command.

Another area of research is network virtualization for Clouds, particularly for the support of distributed Clouds. Virtual Distributed Ethernet (VDE) (Davoli 2005) has been already successfully used as a communication infrastructure for Cloud Computing (Nurmi et al 2009). The challenge is to extend the idea of VDE for a more effective support of highly distributed and dynamic Clouds such as those provided by peer-to-peer components.

4 Experience Report

We have addressed three separate research issues located at the Virtualization layer, at the QoS Management layer and at the Application layer of Figure 2, respectively. Specifically, at the Virtualization layer we have addressed the problem of energy saving in a Cloud environment and developed a solution inspired by the gossip-based protocols for information dissemination described in (Jelasity et al 2005). At the QoS Management layer we have investigated how to respond effectively to the QoS requirements of Cloud customer applications, and developed an architecture that enables the instantiation of what we have termed "QoS-aware Clouds" (see Subsection 3.2). Finally, at the Application layer, we have studied how Cloud Computing can be used in the context of mobile computing in order to provide mobile devices, such as mobile phones and tablet PCs, with communication service continuity.

The results of our findings are summarized below. A full discussion of these results can be found in (Marzolla et al 2011a; Ferretti et al 2010, 2011).

4.1 Saving Energy in a Cloud

A Cloud is physically hosted within large datacenters, containing a large number of computing nodes. The energy requirement of the whole datacenter (for

illumination, power supply, cooling and so on) is a significant fraction of the total operating costs (Hoelzle and Barroso 2009). Thus, reducing the energy consumption is becoming an important issue, both for economical reasons (reducing costs) and for making IT services environmentally sustainable.

We have addressed this problem at the Virtualization Layer of our architecture of Figure 2, and developed a solution that moves VMs on a limited subset of the available (physical) computing resources, so that the remaining (idle) computing nodes can be switched to low power consumption mode. (The process of aggregating services running on multiple servers into a reduced number of more powerful servers is known as server consolidation. Below we use the phrase VM consolidation to denote the consolidation of multiple VMs in a reduced number of physical hosts.)

As an example, assume that a Cloud includes three hosts. Each host can execute four VMs (i.e., each processor has four cores so that it can host four VMs, each providing the computational power of a single CPU core). Assume further that at a given time, the three hosts are running four VM instances labeled from VM_1 to VM_4, as depicted in Figure 3.

According to the allocation shown in Figure 3(a), all three hosts are running user code, even though their relative utilizations are quite low: as a VM is confined to run in a single core, the utilization of hosts 1 and 3 is at most 25%, while the utilization of host 2 is at most 50%.

The situation above is far from optimal. It is known that most current hardware is power inefficient under light load (Hoelzle and Barroso 2009). Unfortunately, studies show that in practice servers operate at 10% to 50% of their maximum utilization level (Fan et al 2007). Thus, it makes sense to move VMs on fewer, highly utilized servers so that the remaining (empty) servers can be put in power-saving mode. Figure 3(b) shows the effect of migrating VM_1 and VM_4 to host 2. Hosts 1 and 3 can now be switched to the power-saving state, until new VMs need to be allocated.

In order to address the above problem, we have developed a fully distributed algorithm, named V-Man, for VM consolidation in Cloud systems. This algorithm (described in detail in Marzolla et al (2011a)) is based on a simple gossip protocol that does not require any central coordinator or globally shared data structure. It is completely VM and application agnostic, and does not require any instrumentation of either the VMs or the hosted applications. V-Man can be executed periodically to identify a new arrangement of existing VM instances so that the number of empty servers is maximized. Once the new allocation has been identified, it is possible to migrate VM instances to their final destination using the live migration feature provided by most Virtual Machine monitors, e.g., (Xen 2010; OpenVZ 2010; VMware 2010).

We have evaluated V-Man through simulation. The results obtained show that our algorithm converges quickly to an optimal allocation of VM instances, has good scalability and high resiliency to failures. In order to illustrate these properties of V-Man, we describe below two specific experiments that assume two opposite scenarios; namely, a static scenario and a dynamic scenario. The

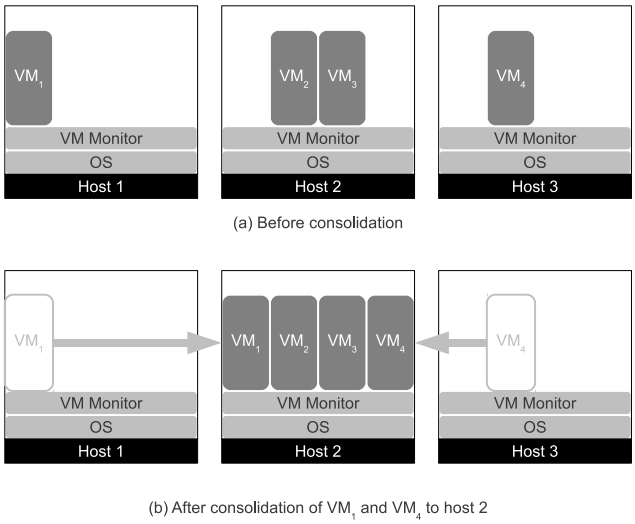

Fig. 3 VM Consolidation Example

full record of our experiments is described in the already cited reference (Marzolla et al 2011a).

We implemented V-Man using Peersim (Jelasity et al 2010), an open source Java simulator of peer-to-peer systems. All our tests have been performed on a (simulated) network with 10000 nodes; the maximum number of VMs each node can maintain concurrently is 8.

Experiment #1: Static system. In this first experiment we investigate the convergence speed of V-Man. To do so, we consider a static system, where the number of VMs is constant, and no servers join or leave the system. At time $t = 1$ the system is initialized by putting a random number of VMs on each server. Figure 4 summarizes our results. This Figure shows an area plot of the fraction of empty servers (white area at the bottom) and full servers (dark area at the top); the height of the light gray area on each plot denotes the fraction of hosts which are neither full nor empty, and thus can be consolidated. The thick horizontal line shows the value of the optimal number of empty servers. We observe that V-Man converges very quickly: after the first iteration the fraction of empty hosts produced by the V-Man allocation is only slightly lower than the optimal value.

Experiment #2: Dynamic system. In this experiment we consider a fully dynamic scenario, in which at each step we add and remove VMs, but also add and remove servers from the Cloud. We simulate a major Cloud outage at time $t = 5$, where 1000 random servers are removed from the system; all VMs running on them are lost. Then, 2000 new (empty) servers are added at time $t = 10$. At each step t we add or remove Δt VMs from the system, where Δt is

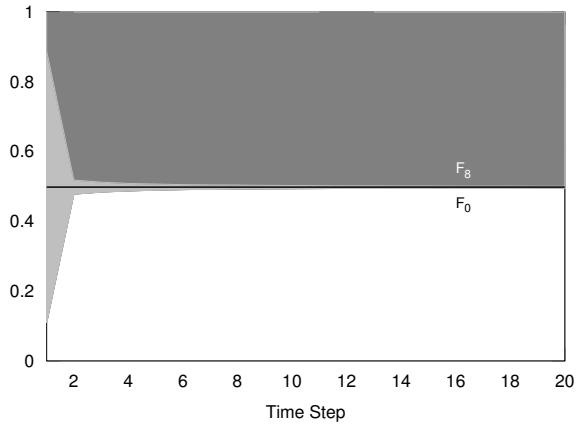

Fig. 4 Static system

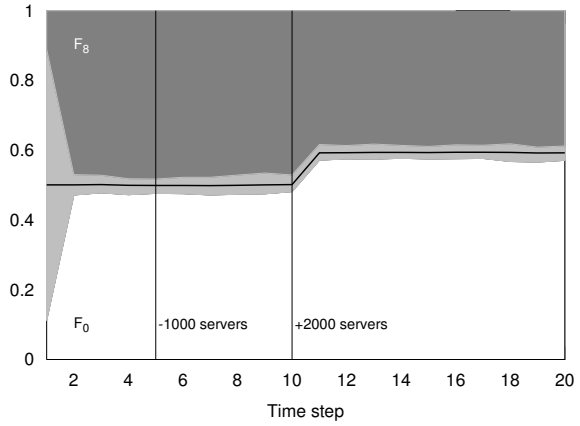

Fig. 5 Fully dynamic system

uniformly distributed in $[-500, 500]$. Figure 5 show that V-MAN is resilient to node failures, which is a common features of many gossip-based protocols (Jelasity et al 2005). Note that the average fraction of empty servers is always a good approximation of the optimal value.

4.2 Meeting QoS Application Requirements in a Cloud

Our work at the QoS Management Layer of Figure 2 has been motivated by the following observation. The success of next-generation Cloud Computing infrastructures will depend on how effectively these infrastructures will be able to instantiate and dynamically maintain computing platforms, constructed out of Cloud resources and services, that meet arbitrarily varying resource and

service requirements of Cloud costumer applications. Typically, these applications will be characterized by QoS requirements, such as timeliness, scalability, high availability, trust, security, specified in the so-called SLAs; an SLA is a legally binding contract which states the QoS guarantees that an execution environment, such as a Cloud based computing platform, has to provide its hosted applications with.

To the best of our knowledge, current Cloud technology is not fully tailored to honor possible SLAs, although both the industrial and the academic research communities are showing growing interest on issues of QoS assurance within the context of Cloud Computing. In general, honoring an SLA requires an accurate assessment of the amount (and characteristics) of the needed resources. Application services hosted in Clouds (e.g., Web applications, Web services) are often characterized by high load variance; hence, the amount of resources needed to honor their SLAs may vary notably over time.

As of today, in order to ensure that an application SLA is not violated, a resource overprovision policy is often adopted, that is based on evaluating (either through application modeling or through application benchmarking) all possible resources a hosted application can require in the worst case, and then statically allocating these resources to that application. This policy can lead to a largely suboptimal utilization of the hosting environment resources. In fact, being based on a worst-case scenario, a number of allocated resources may well remain unused at run time.

This limitation can be overcome by developing a middleware architecture, as we propose, that can be integrated in a Cloud Computing platform so as to manage dynamically the Cloud configuration, and honor the SLAs of the applications that platform hosts. This can be accomplished by adding to, and removing from, the Cloud resources at run time, as needed, in order to respond effectively to the QoS requirements of the Cloud customer applications. For the purposes of our discussion, we term "QoS-aware Cloud" a Cloud Computing environment augmented with our middleware architecture.

Thus, a QoS-aware Cloud can change dynamically the amount of resources made available to the applications it hosts, in a proactive way. Optimal resource utilization is achieved by providing (and maintaining at run time) each hosted application with the number of resources which is sufficient to guarantee that the application SLA is not violated.

The architecture we have developed, fully described and evaluated in (Ferretti et al 2010), is summarized below. The principal services it incorporates are a Configuration Service and a Load Balancer. This latter Service includes a Monitoring Service and an SLA Policy Engine. The Load Balancer is responsible for implementing load dispatching and balancing functionalities. It receives requests from the application clients and dispatches these requests to platform resources according to one of a host of load balancing strategies the Load Balancer can be configured to deploy.

The Load Balancer incorporates a Monitoring Service in charge of monitoring incoming requests and related responses, so as to check and log whether the SLAs associated to these requests are met or violated. The SLA Policy

Engine embodied in the Load Balancer examines the logs of the Monitoring Service to identify possible SLA violations, and determines whether a reconfiguration is necessary, i.e. if the Cloud platform hosting the application needs additional virtual nodes (scaling up) or if some virtual nodes may be released (scaling down). In case a reconfiguration is necessary, the Configuration Service is invoked so as to reconfigure the Cloud platform. This Service, in addition to adding or removing virtual resources as necessary, is responsible for detecting the upper boundary above which adding new resources does not introduce further significant performance enhancements.

The effectiveness of our architecture has been assessed through simulations, which are described in detail in (Ferretti et al 2011); for the purpose of current discussion, we summarize below two results of this assessment. In our simulation, we have assumed that our middleware can manage a pool of spare VMs available for allocation. If our middleware detects that the SLA is being violated, it allocates a new VM (removing it from the pool) to the application hosting platform. In this case the hosting platform (augmented with the new VM) will become more responsive and the average response time will return under the limit specified in the SLA. Similarly, the resource optimization principle requires to deallocate VMs (and insert them in the pool of spare VMs) when they are no longer necessary in order to meet the SLA.

The objective of our assessment was to verify whether our approach enables to instantiate application hosting platforms that both meet their SLAs and optimize the use of the resources in the Cloud. Specifically, we have evaluated the ability of our architecture to meet the application response time requirements and to limit the SLA violation rate. As in practice a small percentage of SLA violations (termed the SLA efficiency attribute) can be tolerated, in our experimental evaluation we set the SLA efficiency attribute to 95%; thus, the SLA response time requirement could be violated by 5% over a predefined timeframe, at most. The allocation time, i.e. the time necessary to set up a new VM and to include it in the execution environment, was set to 2 seconds.

We injected artificial load into the Cloud following a simple request distribution: the load has been progressively increased until it reached about 90 requests per second (the limit specified in the SLA was 100 reqs/sec) and then progressively decreased. This experiment ran for approximately 400 seconds of simulated time; the results obtained are illustrated in Figure 6. As this Figure shows, our middleware components dynamically adjust the hosting platform size as necessary, augmenting the number of clustered VMs as load increases and releasing VMs as load decreases. However, note that the SLA Policy Engine tends to deallocate a VM and to re-allocate it again quickly. Although this behavior could cause additional overhead, it optimizes the resource utilization as it permits to use as little VMs as possible to honor the SLA.

In addition, in this experiment we measured the percentage of SLA violations (see Figure 7). In this Figure, the peaks correspond to the instant in which a new node had to be added to the hosting platform for not incurring SLA efficiency requirement violations; as shown in Figure 7, the SLA violation rate is maintained below the 5% limit imposed by the hosting SLA.

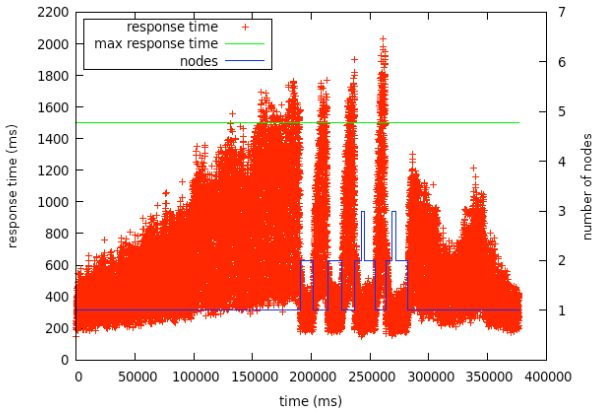

Fig. 6 Resource Utilization and Response Time

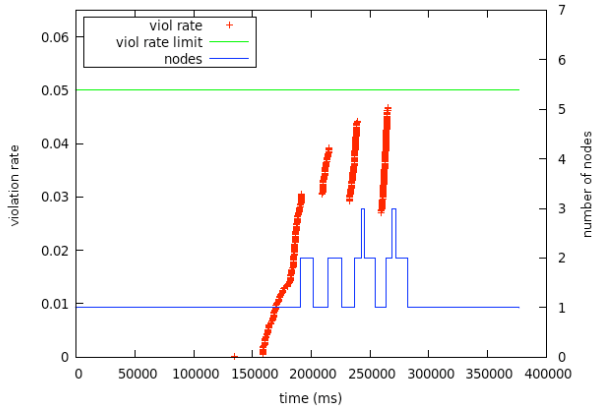

Fig. 7 Resource Utilization and Violation Rate

4.3 Supporting Mobile Multimedia Communications through Cloud Computing

It has been pointed out that mobile interactive application services can contribute to augment the Cloud Computing momentum (Armbrust et al 2010). These services are likely to be attracted to the Cloud as they may require both high availability and the use of large data sets that can be hosted conveniently in a Cloud (e.g., video streaming); in addition, we observe that mobile interactive applications such as VoIP applications, video streaming and on-line games require both seamless communications and support for real time interactions.

In view of these requirements, we have developed a (cross-layer) proxy-based distributed architecture which provides mobile devices with the abstraction of always-connected services (Ghini et al 2010). This abstraction consists of maintaining seamless communications across multiple networks. Thus, in

essence, our architecture makes use of all the networks available to a mobile device, and dynamically adapts their use on the basis of their performance and costs, transparently to the applications that device is running (and to the remote service that device may be interacting with).

The Cloud in this context is used to guarantee the applications running in a mobile device with continuity of the communication service, regardless of the number of networks that are involved in the mobile application nomadic activities.

Our architecture is based on a pair of proxies for each mobile device; namely, a client proxy in the mobile device, and a correspondent server proxy in the Cloud (Ferretti et al 2011).

A client (a mobile device) entering the system contacts a gateway that allocates the Cloud resources that client needs, and provides it with details on the specific server proxy configured to support its communications.

In summary, our architecture consists of the following three principal subsystems: the General-purpose Cloud Subsystem, the Cloud-side Mobility Support Subsystem, and the Client-side Mobility Support Subsystem. Their relative responsibilities can be summarized as follows. The General-purpose Cloud Subsystem (i) manages the physical resources of the hosts, (ii) runs the virtual machines on the selected host of the Cloud, (iii) implements admission control, accounting and billing mechanisms, and (iv) provides libraries, tools and the execution environment for use from the applications. The Cloud-side Mobility Support Subsystem provides the mobile devices with the application specific always-connected service, and consists of the previously cited gateways and server proxies. It is responsibility of the gateway to select the most suitable location of the server proxy for each given mobile user, so as to minimize the network latency between the pair of proxies. Finally, the Client-side Mobility Support Subsystem runs in the mobile device and executes both the user application and the client proxy that hides the effects of the user movements.

To conclude this Section, two observations are in order. Firstly, the Cloud is employed as an IaaS as a virtual machine executing a server proxy is set up each time a (new) mobile client makes a request. Secondly, the system outlined above differs from common applications/services running in a Cloud as these typically do not impose constraints on the geographical location of the Cloud node where the application service is executed. In contrast, in our proposal a crucial requirement is that the server-proxy be located so as to minimize the communication latency with its client, and optimize the servers responsiveness. Hence, the use of a single datacenter for the Cloud may not be an appropriate choice; rather, completely distributed (or federated) Clouds may offer effective solutions.

5 Concluding Remarks

In this paper we have introduced a novel architectural structuring of Cloud Computing environments. This structuring can be thought of as consisting

of two orthogonal dimensions; namely, a vertical dimension that captures the Cloud functionalities, and a horizontal dimension that captures the Cloud organization model.

This architectural structuring has enabled us to isolate and address a number of relevant problems in the Cloud computing field; our proposed solutions to these problems have been summarized in this paper.

Our plan is to develop in greater detail the Cloud structuring introduced in this paper and use it to classify Cloud services and applications according to the two dimensions mentioned above. In addition, through the development of our architectural structuring and the above classification, we aim at the following twofold objective.

On the one hand we would like to make available to Cloud customers (i.e., both Cloud service end users and Cloud service providers) a tool, based on our classification, that enable them to make an informed choice as to which Cloud architecture can be the most suitable for their relative scope. On the other hand, through our structuring and classification, we would like to provide Cloud designers with assistance in taking the appropriate design decisions as to the properties the Cloud Computing environment they are developing is to posses, in order to respond effectively to the requirements of the expected users of that environment.

References

Anderson DP (2004) Boinc: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society, Washington, DC, USA, GRID '04, pp 4–10, DOI 10.1109/GRID.2004.14

Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53:50–58, DOI 10.1145/1721654.1721672

Avetisyan A, Campbell R, Gupta I, Heath M, Ko S, Ganger G, Kozuch M, O'Hallaron D, Kunze M, Kwan T, Lai K, Lyons M, Milojicic D, Lee HY, Soh YC, Ming NK, Luke JY, Namgoong H (2010) Open cirrus: A global cloud computing testbed. Computer 43(4):35–43, DOI 10.1109/MC.2010.111

Babaoglu O, Canright G, Deutsch A, Caro GAD, Ducatelle F, Gambardella LM, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T (2006a) Design patterns from biology for distributed computing. ACM Trans Auton Adapt Syst 1:26–66, DOI 10.1145/1152934.1152937

Babaoglu O, Jelasity M, Kermarrec AM, Montresor A, van Steen M (2006b) Managing clouds: a case for a fresh look at large unreliable dynamic networks. SIGOPS Oper Syst Rev 40:9–13, DOI 10.1145/1151374.1151379

Buyya R, Ranjan R, Calheiros RN (2010) Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu CH, Yang LT, Park JH, Yeo SS (eds) ICA3PP (1), Springer, Lecture Notes in Computer Science, vol 6081, pp 13–31, DOI 10.1007/978-3-642-13119-6_2

Cloud Standards Wiki (2010) Cloud standards wiki. URL http://cloud-standards.org/

Contrail (2010) Contrail project. URL http://contrail-project.eu/start

Davoli R (2005) Vde: virtual distributed ethernet. In: Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on, Trento (Italy), pp 213–220, DOI 10.1109/TRIDNT.2005.38

Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proc. ISCA'07, ACM, New York, NY, USA, pp 13–23, DOI 10.1145/1250662.1250665

Ferretti S, Ghini V, Panzieri F, Pellegrini M, Turrini E (2010) QoS-Aware Clouds. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pp 321–328, DOI 10.1109/CLOUD.2010.17

Ferretti S, Ghini V, Panzieri F (2011) Structuring clouds for mobile multimedia. IEEE COMSOC MMTC E-Letter 6(3):27–30

Gardenghi L, Goldweber M, Davoli R (2008) View-os: A new unifying approach against the global view assumption. In: Proceedings of the 8th international conference on Computational Science, Part I, Springer-Verlag, Berlin, Heidelberg, ICCS '08, pp 287–296

Ghini V, Ferretti S, Panzieri F (2010) Mobile games through the nets: a cross-layer architecture for seamless playing. In: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, ICST, Brussels, Belgium, SIMUTools '10, pp 7:1–7:8, DOI 10.4108/ICST.SIMUTOOLS2010.8656

Hoelzle U, Barroso LA (2009) The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1st edn. Morgan and Claypool Publishers, DOI 10.2200/S00193ED1V01Y200905CAC006

Jelasity M, Montresor A, Babaoglu O (2005) Gossip-based aggregation in large dynamic networks. ACM Trans Comput Syst 23:219–252, DOI 10.1145/1082469.1082470

Jelasity M, Montresor A, Jesi GP, Voulgaris S (2010) The Peersim simulator. URL http://peersim.sf.net

Li J, Chinneck J, Woodside M, Litoiu M, Iszlai G (2009) Performance model driven qos guarantees and optimization in clouds. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE Computer Society, Washington, DC, USA, CLOUD '09, pp 15–22, DOI 10.1109/CLOUD.2009.5071528

Litoiu M, Woodside M, Wong J, Ng J, Iszlai G (2010) A business driven cloud optimization architecture. In: Proceedings of the 2010 ACM Symposium on Applied Computing, ACM, New York, NY, USA, SAC '10, pp 380–385, DOI 10.1145/1774088.1774170

Marzolla M, Babaoglu O, Panzieri F (2011a) Server consolidation in clouds through gossiping. Technical Report UBLCS-2011-01, Department of Computer Science, University of Bologna, Italy, URL http://www.cs.unibo.it/research/reports/

Marzolla M, Ferretti S, D'Angelo G (2011b) Dynamic scalability for next generation gaming infrastructures. In: Proc. 4th ACM/ICST International Conference on Simulation Tools and Techniques (SIMUTools 2011), Barcelona, Spain, pp 1–8

Mell P, Grance T (2011) The NIST Definition of Cloud Computing (Draft)– Recommendations of the National Institute of Standards and Technology. Special publication 800-145 (draft), Gaithersburg (MD)

Nathuji R, Kansal A, Ghaffarkhah A (2010) Q-clouds: managing performance interference effects for qos-aware clouds. In: Proceedings of the 5th European conference on Computer systems, ACM, New York, NY, USA, EuroSys '10, pp 237–250, DOI 10.1145/1755913.1755938

Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2009) The eucalyptus open-source cloud-computing system. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, USA, CCGRID '09, pp 124–131, DOI 10.1109/CCGRID.2009.93

OpenVZ (2010) OpenVZ. URL http://wiki.openvz.org/

Rochwerger B, Galis A, Levy E, Caceres J, Breitgand D, Wolfsthal Y, Llorente I, Wusthoff M, Montero R, Elmroth E (2009) Reservoir: Management technologies and requirements for next generation service oriented infrastructures. In: Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on, New York, USA, pp 307–310, DOI 10.1109/INM.2009.5188828

Rushby J (2011) Distributed secure systems: Then and now. Newcastle upon Tyne (UK), presented at Dependable and Historic Computing: The Randell Tales–Prof. Brian Randell's 25th Birthday Seminar

VMware (2010) VMware vMotion. URL http://www.vmware.com/products/vmotion/

Wuala (2010) Wuala. URL http://www.wuala.com/

Xen (2010) Xen hypervisor. URL http://www.xen.org/

Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications 1:7–18, DOI 10.1007/s13174-010-0007-6, 10.1007/s13174-010-0007-6