Springer
the language of science

Dear Author/Editor,

Here are the proofs of your chapter as well as the metadata sheets.

Metadata

- Please carefully proof read the metadata, above all the names and address.
- In case there were no abstracts for this book submitted with the manuscript, the first 10-15 lines of the first paragraph were taken. In case you want to replace these default abstracts, please submit new abstracts with your proof corrections.

Page proofs

- Please check the proofs and mark your corrections either by
  - entering your corrections online
    or
  - opening the PDF file in Adobe Acrobat and inserting your corrections using the tool "Comment and Markup"
    or
  - printing the file and marking corrections on hardcopy. Please mark all corrections in dark pen in the text and in the margin at least ¼" (6 mm) from the edge.
- You can upload your annotated PDF file or your corrected printout on our Proofing Website. In case you are not able to scan the printout , send us the corrected pages via fax.
- Please note that any changes at this stage are limited to typographical errors and serious errors of fact.
- If the figures were converted to black and white, please check that the quality of such figures is sufficient and that all references to color in any text discussing the figures is changed accordingly. If the quality of some figures is judged to be insufficient, please send an improved grayscale figure.

Metadata of the chapter that will be visualized online

Book Title		Cloud Computing
Chapter Title		Beyond the Clouds: How Should Next Generation Utility Computing Infrastructures Be Designed?
Copyright		Springer International Publishing Switzerland 2014
Corresponding Author	Prefix	
	Family name	**Bertier**
	Particle	
	Given name	**Marin**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Marin.Bertier@inria.fr
Author	Prefix	
	Family name	**Desprez**
	Particle	
	Given name	**Frédéric**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Frederic.Desprez@inria.fr
Author	Prefix	
	Family name	**Fedak**
	Particle	
	Given name	**Gilles**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Gilles.Fedak@inria.fr
Author	Prefix	
	Family name	**Lebre**
	Particle	
	Given name	**Adrien**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Adrien.Lebre@inria.fr
Author	Prefix	
	Family name	**Orgerie**
	Particle	
	Given name	**Anne-Cécile**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Anne-Cecile.Orgerie@inria.fr

Author	Prefix	
	Family name	**Pastor**
	Particle	
	Given name	**Jonathan**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Jonathan.Pastor@inria.fr
Author	Prefix	
	Family name	**Quesnel**
	Particle	
	Given name	**Flavien**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Flavien.Quesnel@inria.fr
Author	Prefix	
	Family name	**Rouzaud-Cornabas**
	Particle	
	Given name	**Jonathan**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Jonathan.Rouzaud-Cornabas@inria.fr
Author	Prefix	
	Family name	**Tedeschi**
	Particle	
	Given name	**Cédric**
	Suffix	
	Division	Inria
	Organization	Campus universitaire de Beaulieu
	Address	35042 Rennes, France
	Email	Cedric.Tedeschi@inria.fr

Abstract	To accommodate the ever-increasing demand for Utility Computing (UC) resources while taking into account both energy and economical issues, the current trend consists in building even larger data centers in a few strategic locations. Although, such an approach enables to cope with the actual demand while continuing to operate UC resources through centralized software system, it is far from delivering sustainable and efficient UC infrastructures. In this scenario, we claim that a disruptive change in UC infrastructures is required in the sense that UC resources should be managed differently, considering locality as a primary concern. To this aim, we propose to leverage any facilities available through the Internet in order to deliver widely distributed UC platforms that can better match the geographical dispersal of users as well as the unending resource demand. Critical to the emergence of such locality-based UC (LUC) platforms is the availability of appropriate operating mechanisms. We advocate the implementation of a unified system driving the use of resources at an unprecedented scale by turning a complex and diverse infrastructure into a collection of abstracted computing facilities that is both easy to operate and reliable. By deploying and using such a LUC Operating System on backbones, our ultimate vision is to make possible to host/operate a large part of the Internet by its internal structure itself: a scalable and nearly infinite set of resources delivered by any computing facilities forming the Internet, starting from the larger hubs operated by ISPs, governments, and academic institutions to any idle resources that may be provided by end users.

Keywords	Utility Computing - UC - Locality-based UC - Distributed Cloud Computing - IaaS - Efficiency - Sustainability

Chapter 14
Beyond the Clouds: How Should Next Generation Utility Computing Infrastructures Be Designed?

Marin Bertier, Frédéric Desprez, Gilles Fedak, Adrien Lebre, Anne-Cécile Orgerie, Jonathan Pastor, Flavien Quesnel, Jonathan Rouzaud-Cornabas and Cédric Tedeschi

Abstract To accommodate the ever-increasing demand for Utility Computing (UC) resources while taking into account both energy and economical issues, the current trend consists in building even larger data centers in a few strategic locations. Although, such an approach enables to cope with the actual demand while continuing to operate UC resources through centralized software system, it is far from delivering sustainable and efficient UC infrastructures. In this scenario, we claim that a disruptive change in UC infrastructures is required in the sense that UC resources should be managed differently, considering locality as a primary concern. To this aim, we propose to leverage any facilities available through the Internet in order to deliver widely

M. Bertier (✉) · F. Desprez · G. Fedak · A. Lebre · A.-C. Orgerie · J. Pastor · F. Quesnel · J. Rouzaud-Cornabas · C. Tedeschi
Inria, Campus universitaire de Beaulieu, 35042 Rennes, France
e-mail: Marin.Bertier@inria.fr

F. Desprez
e-mail: Frederic.Desprez@inria.fr

G. Fedak
e-mail: Gilles.Fedak@inria.fr

A. Lebre
e-mail: Adrien.Lebre@inria.fr

A.-C. Orgerie
e-mail: Anne-Cecile.Orgerie@inria.fr

J. Pastor
e-mail: Jonathan.Pastor@inria.fr

F. Quesnel
e-mail: Flavien.Quesnel@inria.fr

J. Rouzaud-Cornabas
e-mail: Jonathan.Rouzaud-Cornabas@inria.fr

C. Tedeschi
e-mail: Cedric.Tedeschi@inria.fr

Z. Mahmood, (ed.), *Cloud Computing,* Computer Communications and Networks,
DOI 10.1007/978-3-319-10530-7_14

Book ID: 319761_1_En ChapterID: 14 Dispatch Date: 23/08/2014 ProofNo: 1

distributed UC platforms that can better match the geographical dispersal of users as well as the unending resource demand. Critical to the emergence of such locality-based UC (LUC) platforms is the availability of appropriate operating mechanisms. We advocate the implementation of a unified system driving the use of resources at an unprecedented scale by turning a complex and diverse infrastructure into a collection of abstracted computing facilities that is both easy to operate and reliable. By deploying and using such a LUC Operating System on backbones, our ultimate vision is to make possible to host/operate a large part of the Internet by its internal structure itself: a scalable and nearly infinite set of resources delivered by any computing facilities forming the Internet, starting from the larger hubs operated by ISPs, governments, and academic institutions to any idle resources that may be provided by end users.

Keywords Utility Computing · UC · Locality-based UC · Distributed Cloud Computing · IaaS · Efficiency · Sustainability

14.1 Introduction

AQ1

The success of Cloud Computing has driven the advent of Utility Computing (UC). However, Cloud Computing is a victim of its own success. In order to answer the escalating demand for computing resources, Cloud Computing providers must build data centers (DCs) of ever-increasing size. As a consequence, besides facing the well-known issues of large-scale platform management, large-scale DCs have now to deal with energy considerations that limit the number of physical resources that one location can host.

Instead of investigating alternative solutions that could tackle the aforementioned concerns, the current trend consists in deploying larger and larger DCs in few strategic locations presenting energy advantages. For example, Western North Carolina, USA, an attractive area due to its abundant capacity of coal and nuclear power, brought about the departure of the textile and furniture industry [21]. More recently, several proposals suggested building next generation DCs close to the polar circle in order to leverage free cooling techniques, considering that cooling accounts for a big part of the electricity consumption [24].

14.1.1 Inherent Limitations of Large-scale Data Centers

Although, building large-scale DCs enables to cope with the actual demand, it is far from delivering sustainable and efficient UC infrastructures. In addition to requiring the construction and the deployment of a complete network infrastructure to reach each DC, it exacerbates the inherent limitations of the Cloud Computing model:

- The externalization of private applications/data often faces legal issues that restrain companies from outsourcing them on external infrastructures, especially when located in other countries.

- The overhead implied by the unavoidable use of the Internet to reach distant platforms is wasteful and costly in several situations: Deploying a broadcasting service of local events or an online service to order pizzas at the edge of the polar circle, for instance, leads to important overheads since most of the users are *a priori* located in the neighborhood of the event/the pizzeria.
- The connectivity to the application/data cannot be ensured by centralized dedicated centers, especially if they are located in a similar geographical zone. The only way to ensure disaster recovery is to leverage distinct sites [23].

The two first points could be partially tackled by hybrid or federated Cloud solutions [4], that aim at extending the resources available on one Cloud with those of another one; however, the third one requires a disruptive change in the way UC resources are managed.

Another issue is that, according to some projections of a recent IEEE report [25], the network traffic has been doubling roughly every year. Consequently, bringing IT services closer to the end users is becoming crucial to limit the energy impact of these exchanges and to save the bandwidth of some links. Similarly, this notion of locality is critical for the adoption of the UC model by applications that need to deal with a large amount of data as getting them in and out using actual UC infrastructures may significantly impact the global performance [18].

The concept of micro/nano DCs at the edge of the backbone [24] may be seen as a complementary solution to hybrid platforms in order to reduce the overhead of network exchanges. However, operating multiple small DCs breaks somehow the idea of mutualization in terms of physical resources and administration simplicity, making this approach questionable.

14.1.2 Ubiquitous and Oversized Network Backbones

One way to partially solve the mutualization concern enlightened by the defenders of large-scale DCs is to directly deploy the concept of micro/nano DCs upon the Internet backbone. People are (and will be) more and more surrounded by computing resources, especially those in charge of interconnecting the IT equipment. Even though these small- and medium-sized facilities include resources that are barely used [3, 8], they can hardly be removed (e.g., routers). Considering this important aspect, we claim that a new generation of UC platforms can be delivered by leveraging existing network centers, starting from the core nodes of the backbone to the different network access points in charge of interconnecting public and private institutions. By such a mean, network and UC providers would be able to mutualize resources that are mandatory to operate network/data centers while delivering widely distributed UC platforms able to better match the geographical dispersal of users. Figure 6.1 allows to better capture the advantages of such a proposal. It shows a snapshot of the network weather map of RENATER, the backbone dedicated to universities and research institutes in France. It reveals several important points:

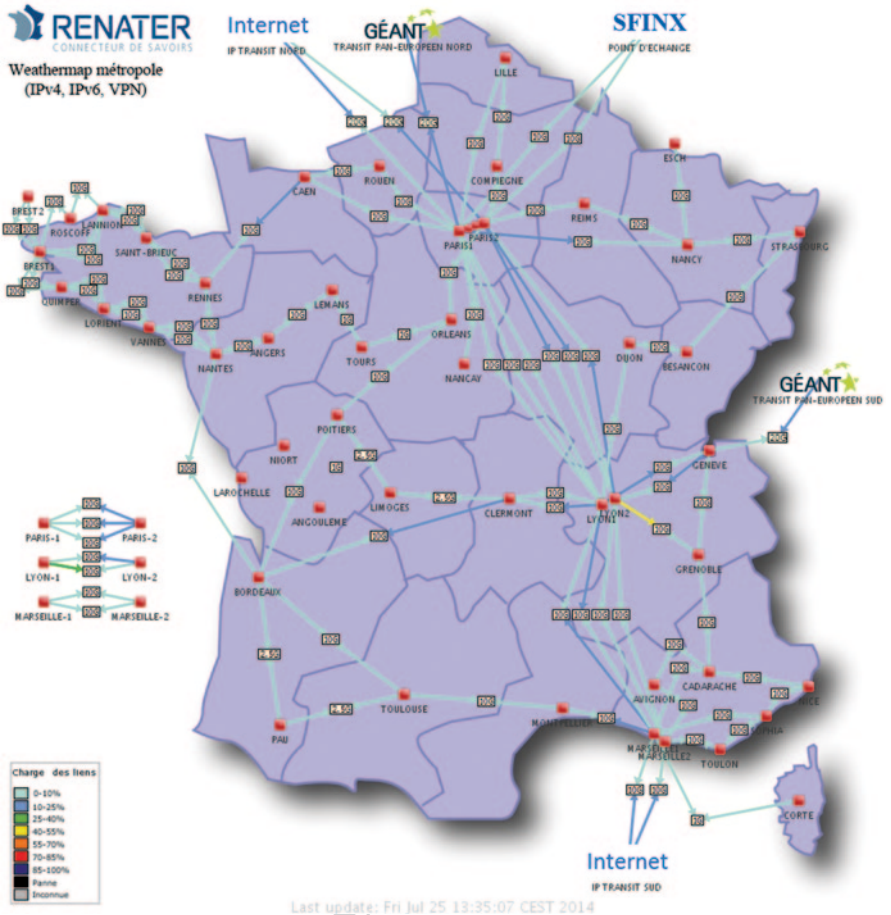

Fig. 6.1 The RENATER Weather Map on May 2013, the 27th, around 4 p.m. Each *red square* corresponds to a particular point of presence (PoP) of the network. The map is available in real-time at: http://www.renater.fr/raccourci

- As mentioned before, most of the resources are underutilized (only two links are used between 45 and 55 %, a few between 25 and 40 %, and the majority below the threshold of 25 %).
- The backbone was deployed and is renewed to match the demand: The density of points of presence (PoPs, i.e., small- or medium-sized network centers), as well as the bandwidth of each link, are more important on the edge of large cities such as Paris, Lyon, or Marseille.
- The backbone was designed to avoid disconnections, since 95 % of the PoPs can be reached by at least two distinct routes.

14.1.3 Locality-Based Utility Computing

This chapter aims at introducing locality-based UC (LUC) infrastructures, a new generation of UC platforms that solve inherent limitations of the Cloud Computing paradigm relying on large-scale DCs. Although, it involves radical changes in the way physical and virtual resources are managed, leveraging network centers is a promising way to deliver highly efficient and sustainable UC services.

From the physical point of view, network backbones provide appropriate infrastructures, i.e., reliable and efficient enough to operate UC resources spread across the different PoPs. Ideally, UC resources would be able to directly take advantage of computation cycles available on network active devices, i.e., those in charge of routing packets. However, leveraging network resources to make external computations may lead to important security concerns. Hence, we propose to extend each PoP with a number of servers dedicated to hosting virtual machines (VMs). As it is natural to assume that the network traffic and UC demands are proportional, larger network centers will be completed with more UC resources than the smaller ones. Moreover, by deploying UC services on relevant PoPs, a LUC infrastructure will be able to natively confine network exchanges to a minimal scope, minimizing altogether the energy footprint of the network, the impact on latency and the congestion phenomena that may occur on critical paths (for instance Paris and Marseille on RENATER).

From the software point of view, the main challenge is to design a comprehensive distributed system in charge of turning a complex and diverse network of resources into a collection of abstracted computing facilities that are both reliable and easy to operate.

The design of the LUC Operating System (OS), an advanced system being able to unify many UC resources distributed on distinct sites, would enable Internet service providers (ISPs) and other institutions in charge of operating a network backbone to build an extreme scale LUC infrastructure with a limited additional cost. Instead of redeploying a complete installation, they will be able to leverage IT resources and specific devices such as computer room air conditioning units, inverters, or redundant power supplies already present in each center of their backbone.

In addition to considering *locality* as a primary concern, the novelty of the LUC OS proposal is to consider the VM as the basic object it manipulates. Unlike existing research on distributed operating systems designed around the process concept, a LUC OS will manipulate VMs throughout a federation of widely distributed physical machines. Virtualization technologies abstract out hardware heterogeneity, and allow transparent deployment, preemption, and migration of virtual environments (VEs), i.e., a set of interconnected VMs. By dramatically increasing the flexibility of resource management, virtualization allows to leverage state-of-the-art results from other distributed systems areas such as autonomous and decentralized systems. Our goal is to build a system that allows end users to launch VEs over a distributed infrastructure as simply as they launch processes on a local machine, i.e., without the burden of dealing with resources availability or location.

14.1.4 Chapter Outline

Section 14.2 describes the key objectives of a LUC OS and the associated challenges. Section 14.3 explains why our vision differs from current and previous UC solutions. In Section 14.4, we present how such a unified system may be designed by delivering the premises of the DISCOVERY (DIStributed and COoperative framework to manage Virtual EnviRonments autonomouslY) system, an agent-based system enabling the distributed and cooperative management of virtual environments over a large-scale distributed infrastructure. Future work and opportunities are addressed in Sect. 14.5. Finally, Sect. 14.6 concludes this chapter.

14.2 Overall Vision and Major Challenges

Similar to traditional operating systems (OSes), a LUC OS is composed of many mechanisms. Trying to identify all of them and establishing how they interact is an on-going work (see Sect. 14.4). However, we have pointed out the following key objectives to be considered when designing a LUC OS:

- *Scalability*: A LUC OS must be able to manage hundreds of thousands of virtual machines (VMs) running on thousands of geographically distributed computing resources. These resources are small- or medium-sized computing facilities and may become highly volatile according to the network disconnections.
- *Reactivity*: To deal with the dynamicity of the infrastructure, a LUC OS should swiftly handle events that require to perform particular operations, either on virtual or on physical resources. This has to be done with the objective of maximizing the system utilization while meeting the Quality of Service (QoS) expectations of VEs. Some examples of operations that should be performed as fast as possible include: (i) the reconfiguration of VEs over distributed resources, sometimes spread across wide area networks (WANs), or (ii) the migration of VMs, while preserving their active connections.
- *Resiliency*: In addition to the inherent dynamicity of the infrastructure, failures, and faults should be considered as the norm rather than the exception at such a scale. The goal is therefore to transparently leverage the underlying infrastructure redundancy to: (i) allow the LUC OS to keep working despite node failures and network disconnections (LUC OS robustness), and to (ii) provide snapshotting as well as high availability mechanisms for VEs (VM robustness).
- *Sustainability*: Although the LUC approach would reduce the energy footprint of UC services by minimizing the cost of the network, it is important to go one step further by considering energy aspects at each level of a LUC OS and propose advanced mechanisms in charge of making an optimal usage of each source of energy. To achieve such an objective, the LUC OS should take account of data related to the energy consumption of the VEs and the computing resources, as well as the environmental conditions (computer room air conditioning unit, location of the site, etc.).

- *Security and Privacy*: Similar to resiliency, the security and privacy issues affect the LUC OS itself and the VEs running on it. Regarding the LUC OS, the goals are: (i) to create trust relationships between different locations, (ii) to secure the peer-to-peer layers, (iii) to include security and privacy decisions and enforcement points in the LUC OS, and (iv) to make them collaborate through the secured peer-to-peer layers to provide end-to-end security and privacy. Regarding the VEs, users should be able to express their requirements in terms of security and privacy; the LUC OS would then enforce these requirements.

In addition to the aforementioned objectives, working on a virtual infrastructure requires to deal with the management of VM images. Managing VM images in a distributed way across a wide area network (WAN) is a real challenge that will require adapting state-of-the-art techniques related to replication and deduplication. Also, the LUC OS must take into account VM images location, for instance: (i) to allocate the right resources to a VE, or (ii) to prefetch VM images, to improve deployment performance or VM relocations.

Finally, one last scientific and technical challenge is the lack of a global view of the infrastructure. Maintaining a global view would indeed limit the scalability of the LUC OS, which is inconsistent with our objective to manage large-scale geographically distributed systems. Therefore, we claim that the LUC OS should rely on decentralized and autonomous mechanisms, which can match and adapt to the volatile topology of the infrastructure. Several decentralized mechanisms are already used in production on large-scale systems; for instance, Amazon relies on the Dynamo service [15] to create distributed indexes and recover from data inconsistencies and Facebook uses Cassandra [29], a massive scale structured store that leverages peer-to-peer techniques. In a LUC OS, decentralized and self-organizing overlays will enable to maintain the information about the current state of both virtual and physical resources, their characteristics, and availabilities. Such information is mandatory to build higher level mechanisms ensuring the correct execution of VEs throughout the whole infrastructure.

14.3 Background

Several generations of UC infrastructures have been proposed and still coexist [19]. However, neither Desktop, Grid, nor Cloud Computing platforms provide a satisfying UC model. Contrary to the current trend that promotes large offshore-centralized DCs as the UC platform of choice, we claim that the only way to achieve sustainable and highly efficient UC services is to target a new infrastructure that better matches the Internet structure. As it aims at gathering an unprecedented amount of widely distributed computing resources into a single platform providing UC services close to the end users, a LUC infrastructure is fundamentally different from existing ones. Keeping in mind the aforementioned objectives, recycling UC resource management solutions developed in the past is doomed to failure.

As previously mentioned, our vision significantly differs from hybrid Cloud Computing solutions. Although these research activities address important concerns related to the use of federated Cloud platforms, such as interface standardization for supporting cooperation and resource sharing, their propositions are incremental improvements of existing UC models. Recent investigations on hybrid Clouds and Cloud federation are comparable in some ways to previous works done on Grids, since the purpose of a Grid middleware is to interact with each resource management system composing the Grid [11, 49, 55].

By taking into account the network issues, in addition to traditional computing and storage concerns in Cloud Computing systems, the European SAIL project [50] is probably the one which targets the biggest advances with regard to previous works on Grid systems. More concretely, this project investigates new network technologies to provide end users of hybrid/federated Clouds with the possibility to configure and virtually operate the network backbone interconnecting the different sites they use [37].

More recently, the *Fog Computing* concept has been proposed as a promising solution to applications and services that cannot be put into the Cloud due to locality issues (mainly the latency and mobility concerns) [10]. Although it might look similar to our vision as they propose to extend the Cloud Computing paradigm to the edge of the network, *Fog Computing* does not target a unified system but rather proposes to add a third party layer (i.e., the *Fog*) between Cloud vendors and end users.

In our vision, UC resources (i.e., Cloud Computing ones) should be repacked in the different points of presence of backbones and operated through a unified system, the LUC OS. As far as we know, the only system that investigated whether a widely distributed infrastructure can be operated by a single system was the XtreemOS Project [36]. Although this project shared some of the goals of the LUC OS, it did not investigate how the geographical distribution of resources can be leveraged to deliver more efficient and sustainable UC infrastructures.

To sum up, we argue for the design and the implementation of a kind of distributed OS, manipulating VEs instead of processes, and considering locality as a primary concern. Referred to as a LUC OS, such a system will include most of the mechanisms that are common to current UC management systems [17, 32, 35, 39–41]. However, each of them will have to be rethought in order to leverage peer-to-peer algorithms. While largely unexplored for building operating systems, peer-to-peer/decentralized mechanisms have the potential to achieve the scalability required to manage LUC infrastructures. Using this technology for establishing the base mechanisms of a massive-scale LUC OS will be a major breakthrough from current static, centralized, or hierarchical management solutions.

14.4 Premise of a LUC OS: The DISCOVERY Proposal

In this section, we propose to go one step further by discussing preliminary investigations around the design and implementation of a first LUC OS proposal: the DISCOVERY system. We draw the premises of the DISCOVERY system by emphasizing some of the challenges as well as some research directions to solve

them. Finally, we give some details regarding the prototype that is under development and how we are going to evaluate it.

14.4.1 Overview

The DISCOVERY system relies on a multi-agent peer-to-peer system deployed on each physical resource composing the LUC infrastructure. Agents are autonomous entities that collaborate with one another to efficiently use the LUC resources. In our context, efficiency means that a good trade-off is found between users' expectations, reliability, reactivity, and availability, while limiting the energy consumption of the system and providing scalability.

In DISCOVERY, each agent has two purposes: (i) maintaining a knowledge base on the composition of the LUC platform, and (ii) ensuring the correct execution of VEs. This includes the configuration, deployment, and monitoring of VEs as well as the dynamic allocation or relocation of VMs to adapt to changes in VEs requirements and physical resources availability. To this end, agents will rely on dedicated mechanisms related to:

- The localization and monitoring of physical resources
- The management of VEs
- The management of VM images
- Reliability
- Security and privacy

14.4.2 Resource Localization and Monitoring Mechanisms

Keeping in mind that DISCOVERY should be designed in a fully decentralized fashion, its mechanisms should be built on top of an overlay network able to abstract out changes that occur at the physical level. The specific requirements of this platform will lead to the development of a novel kind of overlay networks based on locality and a minimalistic design. More concretely, the first step is to design, at the lowest level, an overlay layer intended to hide the details of the physical routes and computing utilities, while satisfying some basic requirements such as locality and availability. This overlay needs to enable the communications between any two nodes in the platform. While overlay computing has been extensively studied over the last decade, we emphasize here on minimalism, especially on one key feature to implement a LUC OS: retrieving nodes that are geographically close to a given departure node.

14.4.2.1 Giving Nodes a Position

The initial configuration of the physical network can take an arbitrary shape. We choose to rely on the Vivaldi protocol [14]. *Vivaldi* is a distributed algorithm

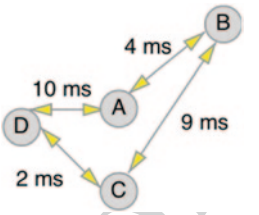

Fig. 6.2 Vivaldi plot before updating positions. Each node pings other nodes. Each node maintains a map of distance

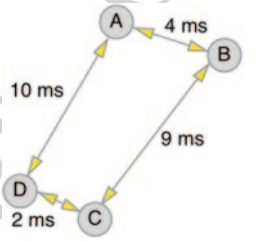

Fig. 6.3 Vivaldi plot after updating positions. The computed positions of other nodes have been updated

as-signing coordinates in the plane to nodes of a distributed system. Each node is equipped with a *view* of the network, i.e., a set of nodes it knows. This view is initially assumed as random. Coordinates obtained by a node reflect its *position* in the network, i.e., close nodes in the network are given close coordinates in the plane. To achieve this, each node periodically checks the round trip time between itself and another node (randomly chosen among nodes in its view) and adapts its distance (by changing its coordinates) with this node in the plane accordingly. Refer to Figs. 6.2 and 6.3 for an illustration of four nodes (A, B, C, and D) moving according to the Vivaldi protocol. A globally accurate positioning of nodes can be obtained if nodes have a few long-distance nodes in their view [14]. These long-distance links can be easily maintained by means of a simple gossip protocol.

14.4.2.2 Searching for Close Nodes

Once the map is achieved (each node knows its coordinates), we are able to decide whether two nodes are *close* by calculating their distance. However, the view of each node does not a priori contain its closest nodes. Therefore, we need additional mechanisms to locate a set of nodes that are close to a given initial node—Vivaldi gives a *location* to each node, but not to the neighborhood. To achieve this, we use a modified distributed version of the classic Dijkstra's algorithm used to find the shortest path between two nodes in a graph. The goal is to build a *spiral* interconnecting the nodes in the plane that are the closest ones to a given initial node. Note that the term *spiral* is here a misuse of language, since the graph actually drawn in the plane might contain crossing edges. The only guarantee is that when following the path constructed, the nodes are always further from the initial node.

Let us consider that our initial point is a node called *I*. The first step is to find a node to build a two-node spiral with *I*. Such a node is sought in the view of *I* by

selecting the node, say S, having the smallest distance with I. I then sends its view to S, I stores S as its successor in the spiral, and S adds I as its predecessor in the spiral. Then I forwards its view to S. S creates a new view by keeping the n nodes which are the closest to I in the views of I and S . This last view is then referred to as the *spiral view* and is intended to contain a set of nodes among which to find the next step of the spiral. Then S restarts the same process: Among the spiral view, it chooses the node with the smallest distance to I, say S', and adds it in the spiral—S becomes the predecessor of S' and S' becomes the successor of S. Then, the spiral view is sent to S' which updates it with the nodes it has in its own view. The process is repeated until we consider that enough nodes have been gathered (a parameter sent by the application).

Note that one risk is to be blocked by having a spiral view containing only nodes that are already in the spiral, leading to the impossibility to build the spiral further. However, this problem can be easily addressed by forcing the presence of a few long-distance nodes whenever it is updated.

14.4.2.3 Learning

Applying the protocol described above, the quality of the spiral is questionable in the sense that the nodes that are actually close to the starting node s may not be included. The only property ensured is that one step forward on the built path always takes us further from the initial node.

To improve the *quality* of the spiral, i.e., reduce the average distance between the nodes it comprises and the initial node, we add a learning mechanism coming with no extra communication cost: when a node is contacted to become the next node in one spiral, and receives the associated spiral view, it can also keep the nodes that are the closest to itself, thus potentially increasing the quality of a future spiral construction.

14.4.2.4 Routing

In the context of a LUC infrastructure, one crucial feature is to be able to locate an existing VM. Having the same strategy consisting in improving the performance of the overlay based on the activity of the application, we envision a routing mechanism which will be improved by past routing requests. By means of the spiral mechanism, a node is able to contact its neighboring nodes to start routing a message.

This initial routing mechanism can be very expensive as the number of hops can be linear in the size of the network. However, from previous communications, a node is able to memorize long links to different locations of the network. Consequently, from each routing request, the source of the request and each node on the path to the destination are able to learn long links, which will significantly reduce the number of hops of future requests. We are currently studying the amount of requests needed to get close to a logarithmic routing complexity. More generally,

we are working on the estimation if the activity of the application is required to: (i) guarantee the constant efficiency of the overlay, and (ii) converge, starting from a random configuration, to a fully efficient overlay network.

14.4.3 VEs Management Mechanisms

In the DISCOVERY system, we define a VE as a set of VMs that may have specific requirements in terms of hardware, software, and also in terms of placement: For instance, some VMs must be on the same node/site to cope with performance objectives while others should not be collocated to ensure high-availability criteria [26]. As operations on a VE may occur in any place from any location, each agent should provide the capability to configure and start a VE, to suspend/resume/stop it, to relocate some of its VMs if the need arises, or simply to retrieve the location of a particular VE. Most of these mechanisms are provided by current UC platforms. However, as mentioned before, they should be revisited to leverage peer-to-peer mechanisms to correctly run on the infrastructure we target (i.e., in terms of scalability, resiliency, and reliability).

As a first example, placing the VMs of a VE requires the ability to find the available nodes that fulfill the VM's needs (in terms of resource requirements as well as placement constraints). Such a placement can start locally, close to the client application requesting it, i.e., in its local group. If no such node is found, a simple navigation ensures that the request will encounter a bridge, leading to the exploration of further nodes. This navigation goes on until an adequate node is found. A similar process is performed by the mechanism in charge of dynamically controlling and adapting the placement of VEs during their lifetime. For instance, to ensure the particular needs of a VM, it can be necessary to relocate other VMs. According to the predefined constraints of VEs, some VMs might be relocated on far nodes while others would prefer to be suspended. Such a mechanism has been deeply studied in the DVMS framework [16, 46]. DVMS (Distributed Virtual Machine Scheduler) is able to dynamically schedule a significant number of VMs throughout a large-scale distributed infrastructure while guaranteeing VM resource expectations.

A second example regards the networking configuration of VEs. Although it might look simple, assigning the right IP to each VM as well as maintaining the intraconnectivity of a VE becomes a bit more complex than in the case of a single network domain, i.e., a single site deployment. Keeping in mind that a LUC infrastructure is, by definition, spread WANwide, a VE can be hosted between distinct network domains during its lifetime. No solution has been chosen yet. Our first investigations led us to leverage techniques such as the IP over P2P project [20]. However, software-defined networking becomes more and more important; investigating proposals such as the Open vSwitch project [44] looks promising to solve such an issue.

14.4.4 VM Images Management

In a LUC infrastructure, VM images could be deployed in any place from any other location. However, being in a decentralized, large-scale, heterogeneous, and widely

spread environment makes the management of VM images more difficult than with conventional centralized repositories. At coarse grain: (i) the management of the VM images should be consistent with regard to the location of each VM in the DISCOVERY infrastructure, and (ii) each VM image should remain reachable or at least recoverable in case of failures. The envisioned mechanisms to manage VM images have been classified into two categories. First, some mechanisms are required to efficiently upload VM images and replicate them across many nodes, to ensure efficiency as well as reliability. Second, other mechanisms are needed to schedule VM image transfers. Advanced policies are important to improve the efficiency of each transfer that may occur either during the first deployment of a VM or during its relocations.

Regarding the storage and replication mechanisms, an analysis of an IBM Cloud concludes that a fully distributed approach using peer-to-peer technology is not the best choice to manage VM images, since the number of instances of the same VM image is rather small [42]. However, central or hierarchical solutions are not suited for the infrastructure we target. Consequently, an improved peer-to-peer solution working with replicas and deduplication has to be investigated to provide more reliability, speed, and scalability to the system. For example, analyzing different VM images shows that at least 30% of the image is shared between different VMs [28]. This 30% can become a 30% reduction in space, or a 30% increase in reliability, or in transfer speed. Depending on the situation, we should decide to go from one scenario to another.

Regarding the scheduling mechanisms, a study showed that VM boot time could be increased from 10 to 240 s when multiple VMs running I/O intensive tasks use the same storage system [53]. Some actions can provide a performance boost and limit the overhead that is still observed in commercial Clouds [33], like providing the image chunks needed to boot first [54], defining a new image format, and pausing the rest of the I/O operations.

More generally, the amount of data linked with VM images is significant. Actions involving data should be aware of consequences on metrics like (but not limited to): energy efficiency, reliability, proximity, bandwidth, and hardware usage. The scheduler could also anticipate actions, for instance, moving images when the load is low or the energy is cheap.

14.4.5 Reliability Mechanisms

Although, we can expect that the frequency of failures on LUC resources should be similar to that in current UC platforms, it is noteworthy to mention that the expected mean time to repair failed equipment might be much higher since resources will be highly distributed. For these reasons, specific mechanisms should be designed to manage failures transparently with a minimum downtime.

Ensuring the high availability of the DISCOVERY system requires the ability to autonomously relocate and restart any service on a healthy node in case of failure. Moreover, a Cassandra-like framework [29] is required to avoid losing or corrupting information belonging to stateful services, since it provides a reliable and highly available back-end.

Regarding the VEs reliability, leveraging periodical VM snapshotting capabilities can provide a first level of fault tolerance. In case of failure, a VE can be restarted from its latest snapshot. Performing VM snapshotting in a large-scale, heterogeneous, and widely spread environment, is a challenging task. However, we believe that adapting ideas that were recently proposed in this field [38] would allow us to provide such a feature.

Snapshotting is not enough for services that should be made highly available, but a promising solution is to use VM replication [43]. To implement VM replication in a WAN, solutions to optimize synchronizations between replicas [22, 47] should be investigated. Also, we think that a LUC infrastructure has a major advantage over other UC platforms, since it is tightly coupled with the network infrastructure. As such, we can expect *low* latencies between nodes which would enable us to provide a strong consistency between replicas while achieving acceptable response time for the replicated services.

Reliability techniques will of course make use of the overlays for resource localization and monitoring. Replicated VMs should be hosted on nodes that have a low probability to fail simultaneously. Following the previously defined overlay structure, this can be done through a navigation scheme where at least one bridge is encountered. A replica can then be monitored by a *watcher*, which is in the same local group as the replica.

14.4.6 Security and Privacy Mechanisms

To be successful, DISCOVERY needs to provide mechanisms and methods to construct trust relationships between resource providers. Trust relationships are known to be complex to build [34]. Providing strong authentication, assurance, and certification mechanisms to providers and users is required, but is definitely not enough. Trust covers socioeconomic aspects that must be addressed but are out of the scope of this chapter. The challenge is to provide a trusted DISCOVERY base.

As overlays are fundamentals to all DISCOVERY mechanisms, another challenge is to ensure that they are not compromised. Recent advances [12] might enable to tackle such concerns.

The third challenge will consist in: (i) providing end users with a way to define their own security and privacy policies, and (ii) ensuring that these policies are enforced. The expression of these policies itself is a complex task, as it requires to improve the current trade-off between security (and privacy) and usability. To ease the expression of these policies, we are currently designing a domain-specific language to define high-level security and privacy requirements [9, 30]. These policies will be enforced in a decentralized manner, by distributed security and privacy decision and enforcement points (SPDEPs) during the lifetime of the VEs. Implementing such SPDEP mechanisms in a distributed fashion will require conducting specific research, as currently there are only prospective proposals for classic UC infrastructures [5, 51]. Therefore, we need to investigate whether such proposals can be adapted to the LUC infrastructure by leveraging appropriate overlays.

Fig. 6.4 The *Peer Actor* Model. The *Supervisor Actor* monitors all the actors it encapsulates while the *Peer Actor* acts as an interface between the services and the overlay

14.4.7 Towards a First Proof of Concept

The first prototype is under heavy development. It aims at delivering a simple mock-up for integration/collaboration purposes. Following the coarse-grained architecture described in the previous sections, we have started to identify all the components participating in the system, their relationships, as well as the resulting interfaces. Conducting such a work now is mandatory to move towards a more complete as well as more complex system.

To ensure a scalable and reliable design, we chose to rely on the use of high-level programming abstractions. More precisely, we are using distributed complex event programming [27] in association with the actor model [1]. This enables us to easily switch between a push- and a pull-oriented approach depending on our needs.

Our preliminary studies showed that a common building block is mandatory to handle resiliency concerns in all components. Concretely, it corresponds to a mechanism in charge of throwing notifications that are triggered by the low level network overlay each time a node joins or leaves it. Such a mechanism makes the design and the development of higher building blocks easier as they do not have to provide specific portions of code to monitor infrastructure changes.

This building block has been designed around the *Peer Actor* concept (see Figs. 6.4 and 6.5). The *Peer Actor* serves as an interface between higher services and the communication layer. It provides methods that enable to define the behaviors of a service when a resource joins or leaves a particular peer-to-peer overlay as well as when neighbors change. Considering that several overlays may coexist in the DISCOVERY system, the association between a *Peer Actor* and its *Overlay Actor* is done at runtime and can be changed on the fly if need be. However, it is noteworthy that each *Peer Actor* takes part to one and only one overlay at the same time. In addition to the *Overlay Actor*, a *Peer Actor* is composed of a *Notification Actor* that processes events and notifies registered actors. As illustrated in Fig. 6.5, a service can use more than one *Peer Actor* (and reciprocally). Mutualizing a *Peer Actor* enables for instance to reduce the network overhead implied by the maintenance of the overlays. In the example, the first service relies on a *Peer Actor*

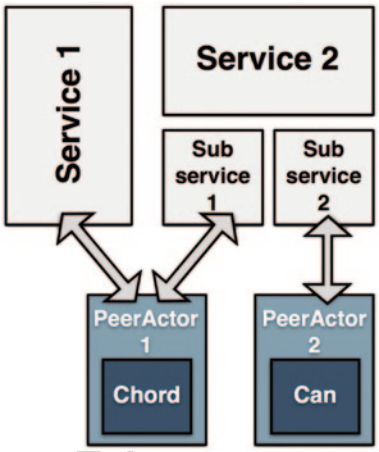

Fig. 6.5 A *Peer Actor* instantiation. The first service relies on a *Peer Actor* implementing a Chord overlay while the second service uses an additional *Peer Actor* implementing a CAN structure

implementing a Chord overlay [52], while the second service uses an additional *Peer Actor* implementing a CAN structure [48].

By such a mean, higher-level services can take the advantage of the advanced communication layers without dealing with the burden of managing the different overlays. As an example, when a node disappears, all services that have been registered as dependent on such an event are notified. Service actors can thus react accordingly to the behavior that has been specified.

Regarding the design and the implementation of the DISCOVERY system, each service is executed inside its own actor and communicates by exchanging messages with the other ones. This ensures that each service is isolated from the others: When a service crashes and needs to be restarted, the execution of other services is not affected. As previously mentioned, we consider that at the LUC infrastructure scale, failures are the norm rather than the exception; hence, we decided that a *Supervisor Actor* would monitor each actor (see Fig. 6.4). DISCOVERY services are under the supervision of the DISCOVERY agent: This design allows to precisely define a strategy that will be executed in case of service failures. This will be the way to introduce self-healing and self-organizing properties to the DISCOVERY system.

This building block has been fully implemented by leveraging the Akka/Scala framework [2], and is available online at https://github.com/BeyondTheClouds.

As a proof of concept (POC), we are implementing a first high-level service in charge of dynamically scheduling VMs across a LUC infrastructure by leveraging the DVMS [46] proposal (see Sect. 14.4.3). The low-level overlay that is being currently implemented is a robust ring based on the Chord algorithm combined with the Vivaldi positioning system: It enables services to select nodes that have low latency, so that collaboration will be more efficient.

To validate the behavior, the performance as well as the reliability of our proof of concept, we are performing several experiments on the Grid'5000 test bed [6] that comprises hundreds of nodes distributed on 10 computing sites that are geographically spread across France. To make experiments with DISCOVERY easier, we developed a set of scripts that can deploy thousands of VMs throughout the whole

infrastructure in a *one-click* fashion [64]. By deploying our POC on each node and by leveraging the VM deployment scripts, we can evaluate real scenario by injecting specific workloads in the different VMs. The validation of this first POC is almost completed. The resulting system will be the first to provide reactive, reliable, and scalable reconfiguration mechanisms of virtual machines in a fully distributed and autonomous way. This new result will pave the way for a complete proposal of the DISCOVERY system.

14.5 Future Work/Opportunities

14.5.1 Geo-diversification as a Key Element

The Cloud Computing paradigm is changing the way applications are designed. In order to benefit from elasticity capabilities of Cloud systems, applications integrate or leverage mechanisms to provision resources, i.e., starting or stopping VMs, according to their fluctuating needs. The ConPaaS system [45] is one of the promising systems for elastic Cloud applications. At the same time, a few projects have started investigating distributed/collaborative ways of hosting famous applications such as Wikipedia or Facebook-like systems by leveraging volunteer computing techniques. However, considering that resources provided by end users were not reliable enough, only few contributions have been done yet. By providing a system that will enable to operate widely spread but more reliable resources closer to the end users, the LUC OS proposal may strongly benefit to this research area. Investigating the benefit of locality provisioning (i.e., combining elasticity and distributed/collaborative hosting) is a promising direction for all Web services that are embarrassingly distributed [13]. Image sharing systems, such as Google Picasa or Flickr, are examples of applications where leveraging locality will enable to limit network exchanges: Users could upload their images on a peer that is close to them, and images would be transferred to other locations only when required (pulling versus pushing model).

LUC infrastructures will allow envisioning a wider range of services that may answer specific SMEs requests such as data archiving or backup solutions, while significantly reducing the network overhead as well as legal concerns. Moreover, it will make the deployment of UC services easier by relieving developers of the burden of dealing with multi-Cloud vendors. Of course, this will require software engineering and middleware advances to easily take advantage of locality. But proposing LUC OS solutions, such as the DISCOVERY project, is the mandatory step before investigating new APIs enabling applications to directly interact with the LUC OS internals.

14.5.2 Energy, a Primary Concern for Modern Societies

The energy footprint of current UC infrastructures, and more generally of the Internet, is a major concern for the society. Although we need to conduct deeper

investigations, we clearly expect that by its design and the way to operate it, a LUC infrastructure will have a smaller impact with a better integration in the whole Internet ecosystem.

Moreover, the LUC proposal is an interesting way to deploy the data furnaces proposal [31]. Concretely, following the Smart City recommendations (i.e., delivering efficient and sustainable ICT services), the construction of new districts in metropolises may take advantage of each LUC/Network PoP in order to heat buildings while operating UC resources remotely by means of a LUC OS. Finally, taking into account recent results about passive data centers, such as solar-powered micro-data centers, might extend this idea. The idea behind passive computing facilities is to limit as much as possible the energy footprint of major hubs and DSLAMS by taking advantage of renewable energies to power them, and by using the heat they produce as a source of energy. Combining such ideas with the LUC approach would allow reaching an unprecedented level of energy efficiency for UC platforms.

14.6 Conclusion

Cloud Computing has entered into our daily life with a great speed. From classic high performance computing simulations to the management of huge amounts of data coming from mobile devices and sensors, its impact can no longer be disregarded. While a lot of progress has already been made in Cloud technologies, there are several concerns that limit the complete adoption of the Cloud Computing paradigm.

In this chapter, we have outlined that, in addition to these concerns, intrinsic issues limit the current model of UC. Instead of following the current trend by trying to cope with existing platforms and network interfaces, we proposed to take a different direction by promoting the design of a system that will be efficient and sustainable at the same time, putting knowledge and intelligence directly into the network backbone itself.

The innovative approach, we introduced, will definitely tackle and go beyond Cloud Computing limitations. Our objective is to pave the way for a new generation of Utility Computing infrastructures that better match the Internet structure by means of advanced operating mechanisms. By offering the possibility to tightly couple UC servers and network backbones throughout distinct sites and operate them remotely, the LUC OS technology may lead to major changes in the design of UC infrastructures as well as in their environmental impact. The internal mechanisms of the LUC OS should be topology-dependent and resources-efficient. The natural distribution of the nodes through the different points of presence should be an advantage, which allows to process a request according to its scale: Local requests should be computed locally, while large computations should benefit from a large number of nodes.

Finally, we believe that LUC investigations may contribute to fill the gap between the distributed computing community and the networked ones. This connection

between these two communities has already started with the different activities around Software-Defined Networking and Network as a Service. In the long term, this may result in a new community dealing with UC challenges where network and computational concerns are fully integrated. Such a new community may leverage the background of both areas to propose new systems that are more suitable to accommodate the needs of our modern societies.

We are well aware that the design of a complete LUC OS and its adoption by companies and network providers require several big changes in the way UC infrastructures are managed and WANs are operated. However, we are convinced that such an approach will pave the way towards highly efficient and sustainable UC infrastructures, coping with heterogeneity, scale, and faults.

References

1. Agha G (1986) Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge
2. Akka (2013) Build powerful concurrent & distributed applications more easily. http://www.akka.io. Accessed: March 2013
3. Andrew O (2003) Data networks are lightly utilized, and will stay that way. review of network economics. Rev Netw Econ 2(3):210–237
4. 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(4):50–58
5. Bacon J, Evans D, Eyers DM, Migliavacca M, Pietzuch P, Shand B (2010) Enforcing end-to-end application security in the cloud (big ideas paper). In: Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware, Springer-Verlag, Berlin, Middleware'10, pp 293–312
6. Balouek D, Carpen Amarie A, Charrier G, Desprez F, Jeannot E, Jeanvoine E, Lèbre A, Margery D, Niclausse N, Nussbaum L et al (2013) Adding virtualization capabilities to the Grid'5000 testbed. In: Ivanov I, Sinderen M, Leymann F, Shan T (eds) Cloud computing and services science, Springer, Berlin
7. Balouek D, Lebre A, Quesnel F (2013) Flauncher and DVMS: deploying and scheduling thousands of virtual machines on hundreds of nodes distributed geographically. In: The sixth IEEE international scalable computing challenge (collocated with CCGRID), Delft, The Netherlands
8. Benson T, Akella A, Maltz DA (2010) Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, New York, IMC'10, pp 267–280
9. Blanc M, Briffaut J, Clevy L, Gros D, Rouzaud-Cornabas J, Toinard C, Venelle B (2013) Mandatory protection within clouds. In: Nepal S, Pathan M (eds) Security, privacy and trust in cloud systems, Springer, Berlin
10. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on mobile cloud computing, ACM, New York, USA, MCC'12, pp 13–16
11. Buyya R, Ranjan R, Calheiros RN (2010) InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Proceedings of the 10th international conference on algorithms and architectures for parallel processing, Springer-Verlag, Berlin, ICA3PP'10, pp 13–31
12. Castro M, Druschel P, Ganesh A, Rowstron A, Wallach DS (2002) Secure routing for structured peer-to-peer overlay networks. SIGOPS Oper Syst Rev 36(SI):299–314

AQ2

13. Church K, Greenberg A, Hamilton J (2008) On delivering embarrassingly distributed cloud services. In: HotNets
14. Dabek F, Cox R, Kaashoek MF, Morris R (2004) Vivaldi: a decentralized network coordinate system. In: Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications, SIGCOMM'04, pp 15–26
15. DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W (2007) Dynamo: Amazon's highly available key-value store. In: Proceedings of twenty-first ACM SIGOPS symposium on operating systems principles, ACM, SOSP'07, pp 205–220
16. Discovery (2013) Distributed VM scheduler. http://beyondtheclouds.github.io/DVMS/. Accessed: March 2013
17. CloudStack (2013) CloudStack, open source cloud computing. http://cloudstack.apache.org. Accessed: March 2013
18. Foster I (2011) Globus online: accelerating and democratizing science through cloud-based services. IEEE Internet Comput 15(3):70–73
19. Foster I, Kesselman C (2011) The history of the grid. In: Foster I, Gentzsch W, Grandinetti L, Joubert GR (eds) Advances in parallel computing—volume 20: HPC: from grids and clouds to exascale. IOS Press, Amsterdam
20. Ganguly A, Agrawal A, Boykin PO, Figueiredo R (2006) IP over P2P: enabling self-configuring virtual IP networks for grid computing. In: Proceedings of the 20th international conference on Parallel and distributed processing, IEEE Computer Society, Washington, DC, USA, IPDPS'06
21. Gary Cook JVH (2013) How dirty is your data? Greenpeace International report
22. Gerofi B, Ishikawa Y (2012) Enhancing TCP throughput of highly available virtual machines via speculative communication. In: Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments, NY, USA, VEE'12
23. Gigaom Consortium (2012) Amazon outages—lessons learned. http://gigaom.com/cloud/amazon-outages-lessons-learned/. Accessed: 23 Feb. 2014
24. Greenberg A, Hamilton J, Maltz DA, Patel P (2008) The cost of a cloud: research problems in data center networks. SIGCOMM Comput Commun Rev 39(1):68–73
25. Group IEW (2012) IEEE 802.3TM industry connections ethernet bandwidth assessment
26. Hermenier F, Lawall J, Muller G (2013) BtrPlace: a flexible consolidation manager for highly available applications. IEEE Transactions on Dependable and Secure Computing
27. Janiesch C, Matzner M, Muller O (2011) A blueprint for event-driven business activity management. In: Proceedings of the 9th international conference on business process management, Springer-Verlag, BPM'11, pp 17–28
28. Jin K, Miller EL (2009) The effectiveness of deduplication on virtual machine disk images. In: Proceedings of SYSTOR 2009: the Israeli Experimental Systems Conference, ACM, New York, USA, SYSTOR'09, pp 7:1–7:12
29. Lakshman A, Malik P (2010) Cassandra: a decentralized structured storage system. SIGOPS Oper Syst Rev 44(2):35–40
30. Lefray A, Caron E, Rouzaud-Cornabas J, Zhang HY, Bousquet A, Briffaut J, Toinard C (2013) Security-aware models for clouds. In: Poster Session of IEEE Symposium on High Performance Distributed Computing (HPDC)
31. Liu J, Goraczko M, James S, Belady C, Lu J, Whitehouse K (2011) The data furnace: heating up with cloud computing. In: Proceedings of the 3rd USENIX conference on hot topics in cloud computing, HotCloud'11
32. Lowe S (2011) Mastering VMware vSphere. Wiley: Indianapolis
33. Mao M, Humphrey M (2012) A performance study on the VM startup time in the cloud. In: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, IEEE Computer Society, CLOUD'12, pp 423–430
34. Miller KW, Voas J, Laplante P (2010) In trust we trust. Computer 43:85–87
35. Moreno-Vozmediano R, Montero R, Llorente I (2012) IaaS cloud architecture: from virtualized datacenters to federated cloud infrastructures. Computer 45(12):65–72

36. Morin C (2007) XtreemOS: a grid operating system making your computer ready for participating in virtual organizations. In: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, IEEE Computer Society, ISORC'07, pp 393–402
37. Murray P, Sefidcon A, Steinert R, Fusenig V, Carapinha J (2012) Cloud networking: an infrastructure service architecture for the wide area. HP Labs Tech Report-HPL-2012–111R1
38. Nicolae B, Bresnahan J, Keahey K, Antoniu G (2011) Going back and forth: efficient multideployment and multisnapshotting on clouds. In: Proceedings of the 20th international symposium on High performance distributed computing, ACM, New York, USA, HPDC'11, pp 147–158
39. Nimbus (2013) Nimbus is cloud computing for science. http://www.nimbusproject.org. Accessed: March 2013
40. OpenNebula (2013) Open source data center virtualization. http://www.opennebula.org. Accessed: March 2013
41. OpenStack (2013) The open source, open standards cloud. http://www.openstack.org. Accessed: March 2013
42. Peng C, Kim M, Zhang Z, Lei H (2012) VDN: virtual machine image distribution network for cloud data centers. In: INFOCOM, 2012, pp 181–189
43. Petrovic D, Schiper A (2012) Implementing virtual machine replication: a case study using Xen and Kvm. In: Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, pp 73–80
44. Pfaff B, Pettit J, Koponen T, Amidon K, Casado M, Shenker S (2009) Extending networking into the virtualization layer. In: ACM HotNets
45. Pierre G, Stratan C (2012) ConPaaS: a platform for hosting elastic cloud applications. IEEE Internet Comput 16(5):88–92
46. Quesnel F, Lebre A, Sudholt M (2012) Cooperative and reactive scheduling in large-scale virtualized platforms with DVMS. Concurr Comput Pract Exp 25(12):1643–1655
47. Rajagopalan S, Cully B, O'Connor R, Warfield A (2012) SecondSite: disaster tolerance as a service. In: Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments, ACM, VEE'12, pp 97–108
48. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content-addressable network. In: SIGCOMM'01: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications, ACM, New York, USA, SIGCOMM'01, pp 161–172
49. Rochwerger B, Breitgand D, Levy E, Galis A, Nagin K, Llorente IM, Montero R, Wolfsthal Y, Elmroth E, Caceres J, Ben-Yehuda M, Emmerich W, Galan F (2009) The reservoir model and architecture for open federated cloud computing. IBM J Res Dev 53(4):4:1–4:11
50. Sail Consortium (2012) Scalable and adaptive internet solutions—European Project FP7 program. http://www.sail-project.eu. Accessed: March 2014
51. Sandhu R, Boppana R, Krishnan R, Reich J, Wolff T, Zachry J (2010) Towards a discipline of mission-aware cloud computing. In: Proceedings of the 2010 ACM workshop on cloud computing security workshop, pp 13–18
52. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM, New York, USA, SIGCOMM'01, pp 149–160
53. Tan T, Simmonds R, Arlt B, Arlitt M, Walker B (2008) Image management in a virtualized data center. SIGMETRICS Perform Eval Rev 36(2):4–9
54. Tang C (2011) FVD: a high-performance virtual machine image format for cloud. In: Proceedings of the 2011 USENIX conference on USENIX annual technical conference, USENIX Association, USENIXATC'11
55. Zhao H, Yu Z, Tiwari S, Mao X, Lee K, Wolinsky D, Li X, Figueiredo R (2012) CloudBay: enabling an online resource market place for open clouds. In: Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, UCC'12

Chapter 14: Author Query

AQ1. The author “Marin Bertier” has been marked as the corresponding author. Please confirm.

AQ2. “[7]” is not cited in the text. Please provide the citation or delete the entry from the reference list.