DISCOVERY, Beyond the Clouds

DIStributed and COoperative Framework to Manage Virtual EnviRonments autonomicallY: A Prospective Study

Adrien Lèbre[1], Paolo Anedda[2], Massimo Gaggero[2], and Flavien Quesnel[1]

[1] ASCOLA Research Group, Ecole des Mines de Nantes, Nantes, France
{firstname.lastname}@mines-nantes.fr
[2] CRS4 Distributed Computing Group, Edificio 1, Polaris, Pula, Italy
{firstname.lastname}@crs4.it

Abstract. Although the use of virtual environments provided by cloud computing infrastructures is gaining consensus from the scientific community, running applications in these environments is still far from reaching the maturity of more usual computing facilities such as clusters or grids. Indeed, current solutions for managing virtual environments are mostly based on centralized approaches that barter large-scale concerns such as scalability, reliability and reactivity for simplicity. However, considering current trends about cloud infrastructures in terms of size (larger and larger) and in terms of usage (cross-federation), every large-scale concerns must be addressed as soon as possible to efficiently manage next generation of cloud computing platforms.

In this work, we propose to investigate an alternative approach leveraging DIStributed and COoperative mechanisms to manage Virtual EnviRonments autonomicallY (DISCOVERY). This initiative aims at overcoming the main limitations of the traditional server-centric solutions while integrating all mandatory mechanisms into a unified distributed framework. The system we propose to implement, relies on a peer-to-peer model where each agent can efficiently deploy, dynamically schedule and periodically checkpoint the virtual environments they manage. The article introduces the global design of the DISCOVERY proposal and gives a preliminary description of its internals.

1 Introduction

Since the first proposals almost ten years ago [15,20], the use of virtual technologies has radically changed the perception of distributed infrastructures. Through an encapsulation of software layers into a new abstraction – the virtual machine (VM) –, users can run their own runtime environment without considering, in most cases, software and hardware restrictions which were formerly imposed by computing centers. Relying on specific APIs, users can create, configure and upload their VMs to cloud computing providers, which in turn are in charge of deploying and running the requested virtual environment (VE) on their physical

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part II, LNCS 7156, pp. 446–456, 2012.

infrastructure. In some ways, users may consider the distributed infrastructure as a unique and large hardware where they can launch as many VMs as they want to compose and recompose their environment on demand.

Because of its flexibility and its indubitable economic advantage, this approach, known now as Infrastructure-as-a-Service (IaaS), is becoming more and more popular. However, running applications in those virtualized environments and upon those infrastructures is still far from reaching the maturity of more usual computing facilities such as clusters or grids. Indeed, most IaaS frameworks, such as Nimbus [1], OpenStack [3] and OpenNebula [32], have been designed with the ultimate goal of deploying VEs upon physical resources, setting aside or addressing only as secondary concerns large-scale infrastructure challenges.

Considering that cloud computing providers permanently invest in new physical resources to satisfy the increasing demand of VEs, all issues related to the management of large-scale infrastructures should be considered as major concerns of IaaS frameworks. This is reinforced with recent proposals promoting the federation of IaaS infrastructures, leading to larger and more complex systems [21]. From our point of view, both the design of IaaS frameworks and the management of VEs should be driven by:

- Scalability, targeting the management of hundred thousands of VMs upon thousands of physical machines (PMs), potentially spread across multiple sites;
- Reliability, considering "hardware failures as the norm rather the exception" [8];
- Reactivity, handling each reconfiguration event as swiftly as possible to maintain VEs' Quality of Service (QoS).

If the first point is a well-known challenge, some clarifications should be made regarding the expectations about the two latest ones. Concerning reliability, IaaS frameworks should be robust enough to face failures. Besides remaining operational – i.e. users can continue to interact with them –, they must provide mechanisms to resume any faulty VEs in a consistent state, while limiting the impact on the sound ones. Regarding reactivity, IaaS frameworks should swiftly handle events that require performing particular operations either on virtual or on physical resources. These events can be related to submissions or completions of VEs, to physical resource changes, or to administrator's interventions. The main objective is to maximize the system utilization while insuring the QoS expectations.

Although, the management and the use of VMs in distributed architectures is a hot topic leading to a significant number of publications, most of the current works only focus on one particular concern. To our best knowledge, no work currently investigates whether all these concerns can be tackled all together into a unified system.

Yet, we assume that the maturity of system virtualization capabilities and recent improvements in their usage [7, 14] enable to design and implement such a

system. To overcome the issues of traditional server-centric solutions, its design should benefit from the lessons learnt from distributed operating systems and Single System Image proposals [29]. Furthermore, to address the different objectives and reduce the management complexity, we advocate the use of autonomic mechanisms [22]. In other words, we argue for the design and the implementation of a distributed OS, sometimes referred as "Cloud OS", manipulating VEs instead of processes. We strongly support to use micro-kernel concepts to deliver a platform agnostic framework where a physical node, as well as a complete IaaS platform, can be seen as a simple bare hardware. As a consequence, the framework can manage each VM of a VE throughout a federation of bare hardware, using the capabilities provided by each of them (for example start/stop, suspend/resume).

To our best knowledge, XenServer [4] and VSphere [24] are probably the most advanced proprietary solutions targeting most of these goals. However, they are still facing scalability issues and do not address, for instance, IaaS federation concerns. In this paper, we propose to go further by giving an overview of the DISCOVERY architecture, a DIStributed and COoperative framework to manage Virtual EnviRonments autonomicallY.

The remaining of the paper is organized as follows. First, we present current IaaS frameworks and most advanced mechanisms to manage VEs in Section 2. Second, we introduce the global architecture and briefly discuss scientific and technical challenges of the components of the DISCOVERY system in Section 3. Section 4 presents an overview of the DISCOVERY engine. Finally, Section 5 concludes and highlights the importance to address such a proposal through a solid community composed of experts of each domain (storage, network, fault-tolerance, P2P, security ...).

2 Related Work

Due to the recent widespread diffusion of cloud computing (CC), there is a growing number of software projects that deal with the management of virtual infrastructures, especially in the context of private CC. Most of these systems are designed to substantially reduce the administrative burden of managing clusters of virtual machines while simultaneously improving the ability of users to request, control, and customize their virtual computing environment. Beyond the previously cited Nimbus, OpenStack and OpenNebula projects, there are a lot of Open Source projects. Among the others, we can mention: OpenQRM [2], SnowFlock [23], Usher [26] and Eucalyptus [28]. Although they differ in approach and technological aspects, most of these systems are designed with a traditional centralized approach. As reported in [13], these systems do not scale well and moreover, lead to the problem of Single Point Of Failures (SPOFs). Such drawbacks have not been really addressed until now and major improvements have rather focused on virtualization internals or on particular needs. For instance, VMs live-migration [12] provides flexibility by enabling to schedule VEs dynamically in a cluster-wide context [17]. But, migrating several VMs among

different physical nodes transparently, while ensuring the correctness of their computation, requires advanced memory management and data transfer strategies [18, 19]. Moreover, when live migration is done at a Wide Area Network (WAN) scale [9, 10], also VM image concerns must be taken into account. As we stated in Section 1, failures become an important issue to address and cloud resiliency is becoming an important task [16]. Checkpointing is a promising approach to system reliability [6,11], since it ensures a way for taking snapshots of the execution of a virtual environment and allows, in case of failure, to restart computations from a previously saved state. VM images management is another big concern. Using traditional network solutions for storage such as the Network File-System (NFS), while it is a perfectly adequate solution for small clusters, it will not scale as the number of nodes increases. Apart from the specifics of a given hardware setup, this is a direct consequence of having an external fixed storage system, whose bandwidth is independent from the computational cluster size. On the contrary, the use of distributed file-systems in the context of VMs management [5] seems very promising and is encouraging the development of dedicated distributed FS specifically tailored to the VMs management [27]. Finally, deploying several VMs in different administrative domains [34], while providing a unified network overlay, requires new solutions based on the creation of virtual isolated network environments [25, 31].

A lot of works have and still continue to be done in virtualization in distributed architectures. However none of these works are focusing on the design and the implementation concerns of a unified system leveraging recent contributions to efficiently manage VEs across a large-scale infrastructure.

3 The DISCOVERY Proposal

While considering previous and on-going works as foundations for the DISCOVERY initiative, we argue for the design and the implementation of a unified framework that aims at insuring scalability, reliability and reactivity in the management of a significant number of VEs. In this section, we present the architecture overview we designed to meet these objectives and then, highlight scientific and technical challenges of each component.

3.1 Architecture Overview

The DISCOVERY architecture relies on a peer-to-peer model composed of several agents (see Figure 1). Each agent cooperates in managing VEs throughout the DISCOVERY network.

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: a user may express the wish to have particular VMs in a same location to cope with performance objectives whereas he/she can ask that others should not be collocated to insure high-availability criteria for instance.

In order to be platform agnostic, each agent leverages virtualization technologies wrappers. This enables to start, stop, suspend, resume and relocate VMs

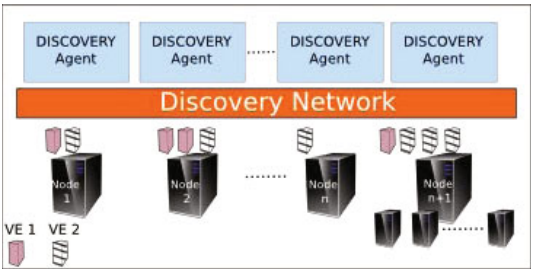

Fig. 1. The DISCOVERY infrastructure

without limiting the DISCOVERY proposal to a particular virtualization platform. Moreover, the adoption of the *Open Virtualization Format* (OVF) [14] by major virtualization actors should enable to soon assign any VM on whatever virtualization platforms. Similarly, we propose to leverage IaaS APIs. By means of specialized DISCOVERY agents that wrap the IaaS functionalities, it allows to treat IaaS frameworks as if they were "super" nodes of the system. Although it implies few restrictions such as the inability to use live-migration between an external PM and an IaaS framework yet, it enables to hide all the underlying instruments so that the VEs are unaware of the physical resources they are running on. Regarding the VM snapshotting capability that is required to insure reliability of VEs, we assume that IaaS providers will extend their API in order to offer it in a mid-term future.

3.2 The DISCOVERY Agent

Relying on the peer-to-peer approach, on the concept of the VEs and on the common set of VM operations, we designed the DISCOVERY agent. At coarse-grain, it is composed of three major services (see Figure 2) (i) the DISCOVERY Network Tracker (DNT), (ii) the Virtual Environments Tracker (VET) and (iii) the Local Resources Tracker (LRT).

DISCOVERY Network Tracker. The DNT is in charge of maintaining a logical view of the DISCOVERY network to make communications and information sharing between services transparent and reliable. Leveraging Distributed Hash Table (DHT) mechanisms [30, 33, 35], it relieves each service of dealing with the burden of nodes' resiliency. First works will focus on reducing as far as possible the DISCOVERY's system states that should be saved into the DHT. The objective is to minimize the performance degradation while insuring the reliability of the whole system. Mid-term challenges will concern the definition of one or several network overlays with respect to the network topologies so that when one peer leaves or fails, the one that takes over is "well" located. Finally, the study of voluntary split or merge of overlays can be also relevant.

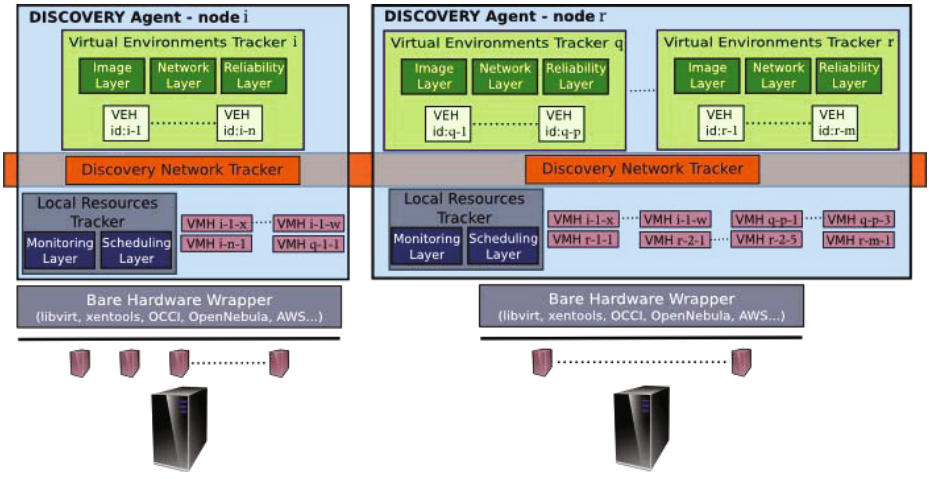

Fig. 2. Architecture overview

Virtual Environment Tracker. Each VET is charge of managing a set of VEs during their whole life cycle. This includes handling user-requests, uploading VM images into the DISCOVERY network and insuring that the VEs it manages can start and correctly run until their completion. The main challenges concern:

- The configuration of the network (covering VM IP assignments and use of advanced technologies to maintain intra-connectivity while insuring isolations and avoiding conflicts between the different VEs [25, 31, 34]).
- The management of the VM images that should be (i) consistent with regard to the location of each VM throughout the DISCOVERY network and (ii) reachable in case of failures.
- The efficient use of the snapshotting capability to resume a VE from its latest consistent state in case of failures.

These three concerns are respectively addressed through functionalities available in the *Network*, *Image* and *Reliability* layers. Each layer will rely on solutions such as the ones described in Section 2. Our objective is to let the possibility to developers to switch between several mechanisms.

Local Resources Tracker. The LRT is in charge of monitoring the resource usage of the bare hardware. It notifies events (such as overloaded, underloaded, extinction requested ...) to other LRTs in order to balance or suspend VMs of VEs with respect to the scheduling policy that has been defined (consolidation, load-balancing, ...). The main challenges concern :

- The management of the events (considering that each event may occur simultaneously throughout the infrastructure leading to several scheduling/reconfigurations processes).
- The scheduling process itself (keeping in mind that for scalability reason, it will not be able to rely on a global view of the resource usage).

– And finally the application of each reconfiguration that may occur concurrently throughout the infrastructure.

According to the lack of solutions that try to address these concerns and considering that the LRT component is central for the DISCOVERY architecture, we chose to start our investigation from it.

4 DISCOVERY in a Nutshell

Assuming that a lot of works has to be done to develop a framework as complex as the one we described, we present in this section, a basic overview of the DISCOVERY engine. This description has been driven by major events/actions that may occur throughout the DISCOVERY network.

When a peer joins the DISCOVERY network, it queries the DNT to get a VET instance. If one peer of the network manages more than one VET, the DNT will assign one of these VETs to the new peer. Otherwise, a new VET instance with a unique id is allocated on the new peer that starts to become active.

A user can query any active peer for the creation of a particular VE. His/her request is forwarded to one of the VETs available in the system according to the DISCOVERY balancing policy. Once the request has been assigned to a particular VET, a *VE handler* (VEH) is created. This VEH is identified by a unique id composed of the id of the VET and a local id incremented each time the VET launches a new VEH. The VEH will monitor and will apply each operation that is mandatory to correctly run the VE. Similarly to the VEH, a *VM handler* (VMH) is created for each VM composing the VE. The VEH and the VMHs interact during the whole execution of the VE.

At the beginning, the VEH starts, locally, as many VMHs as it is requested. The LRT detects these new VMHs and checks whether it will be able to host the related VMs. According to the available resources, each VMH may be relocated to another peer or be informed that the system cannot satisfy their requirements due to a lack of resources. When enough resources are available in the DISCOVERY system, each VMH contacts its VEH to notify it to effectively start the VMs. The VEH is then in charge of delivering the VM images to the right locations and configuring the network (including IP assignments and VLAN setup). When all VMs are started, the VE switches to the running state. Each time the LRT decides to relocate (or suspend) a VM, it notifies the VMH, which in turn informs its VET to perform the requested operation. By preventing direct interactions on VMs, we insure to keep VEs in consistent state. If one of the VMs should be suspended due to a lack of resources, the VEH will suspend the whole VE, keeping it in a consistent state.

When a peer wants to leave the DISCOVERY network, the LRT switches to an overloaded state where each VMH (and by transitivity the related VMs) have to be relocated somewhere else in the DISCOVERY network. In the meantime, the DNT associates the VET to another node so that VMHs can continue to contact it (as illustrated on Figure 2), a DISCOVERY agent can be composed

of several VETs). Once the VET has been assigned to another node and once all VMHs have been relocated (or suspended), the peer can properly leave.

Regarding reliability, two cases must be considered: the crash of VMs and the crash of nodes. In the first case, the reliability relies on (i) the snapshots of the VE, which is periodically performed by the VEH and (ii) the heartbeats that are periodically sent by each VMHs to the VEH. If the VEH does not receive one of the VMHs' heartbeats, it has to suspend all remaining VMs and resume the whole VE from its latest consistent state. This process is similar to the starting one: the missing VMHs are launched locally and the LRT is in charge of assigning them throughout the DISCOVERY network. When the LRT completes this operation, the VMHs receive a notification and in turn contact the VET to resume all VMs from their latest consistent state. Before resuming each VM, the VET checks whether it has to deliver the snapshot images to the nodes. Regarding the crash of a node, the recovery process relies on DHT mechanisms used by the DNT. When a VET starts a new VEH, the description of the associated VE is stored in the DHT. Similarly, this description is updated/completed each time the VEH snapshots the VE (mainly to update the locations of the snapshots). By such a way, when a failure of a node is detected (either by leveraging DHT principles or simply by implementing a heartbeat approach between nodes), the "neighbor" node is able to restart the VET and the associated VEHs from the information that have been previously replicated through the DHT. Once all VEHs have recovered, the VMHs heartbeat mechanism is used either to reattach the VMHs to the VEH or to resume the VE from its latest consistent state if it is needed.

5 Conclusion

It is undeniable: virtualization technology has become a key element of distributed architectures. Although there have been considerable improvements, a lot of works continue to focus on virtualization internals and only few actions address design and implementation concerns of the frameworks that leverage virtualization technologies to manage distributed architectures. Considering the growing size of infrastructures in terms of nodes and virtual machines, new proposals relying on more autonomic and decentralized approaches should be discussed to overcome the limitations of traditional server-centric solutions.

In this paper, we introduced the DISCOVERY initiative that aims at leveraging recent contributions on virtualization technologies and previous distributed operating systems proposals to design and implement a new kind of virtualization frameworks insuring scalability, reliability and reactivity of the whole system. Our proposal relies on micro-kernel approaches and peer-to-peer models. Starting from the point that each node may be seen as a bare-hardware providing basic functionalities to manipulate VMs and monitor resources usages, we design an agent composed of several services that cooperate in managing virtual environments throughout the DISCOVERY network.

Although the design may look simple at the first sight, the implementation of each block will require specific expertise. As an example, strong assumptions

on the internals of the *Virtual Environments Tracker* have been done (considering that the three layers: *Image*, *Network* and *Reliability* were available). Each of them requires deeper investigations with the contributions of the scientific community. Furthermore, the DISCOVERY framework should be extended with other concerns such as security, user quota ... to meet our objective to design and implement a complete distributed OS of VMs. Again, this cannot be done without querying the scientific community.

References

1. Nimbus is cloud computing for science, `http://www.nimbusproject.org/`
2. Openqrm, `http://www.openqrm.com/`
3. Openstack: The open source, open standards cloud. open source software to build private and public clouds, `http://www.openstack.org/`
4. XenServer Administrator's Guide 5.5.0. Tech. rep., Citrix Systems (February 2010)
5. Anedda, P., Leo, S., Gaggero, M., Zanetti, G.: Scalable Repositories for Virtual Clusters. In: Lin, H.-X., Alexander, M., Forsell, M., Knüpfer, A., Prodan, R., Sousa, L., Streit, A. (eds.) Euro-Par 2009 Workshop. LNCS, vol. 6043, pp. 414–423. Springer, Heidelberg (2010)
6. Anedda, P., Leo, S., Manca, S., Gaggero, M., Zanetti, G.: Suspending, migrating and resuming hpc virtual clusters. Future Generation Computer Systems 26(8), 1063–1072 (2010)
7. Bolte, M., Sievers, M., Birkenheuer, G., Niehörster, O., Brinkmann, A.: Non-intrusive virtualization management using libvirt. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2010, pp. 574–579. European Design and Automation Association, Leuven (2010)
8. Borthakur, D.: The Hadoop Distributed File System: Architecture and Design. The Apache Software Foundation (2007)
9. Bose, S.K., Brock, S., Skeoch, R., Rao, S.: CloudSpider: Combining Replication with Scheduling for Optimizing Live Migration of Virtual Machines Across Wide Area Networks. In: 11th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid), Newport Beach, California, U.S.A (May 2011)
10. Bradford, R., Kotsovinos, E., Feldmann, A., Schiöberg, H.: Live wide-area migration of virtual machines including local persistent state. In: Proceedings of the 3rd International Conference on Virtual Execution Environments, VEE 2007, pp. 169–179. ACM, San Diegoe (2007)
11. Chanchio, K., Leangsuksun, C., Ong, H., Ratanasamoot, V., Shafi, A.: An efficient virtual machine checkpointing mechanism for hypervisor-based hpc systems. In: High Availability and Performance Computing Workshop, Denver, USA (2008)
12. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, NSDI 2005, vol. 2, pp. 273–286. USENIX Association, Berkeley (2005)
13. Claudel, B., Huard, G., Richard, O.: Taktuk, adaptive deployment of remote executions. In: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, HPDC 2009. ACM, Munich (2009)
14. DMTF: Open Virtualization Format Specification (January 2010), `http://www.dmtf.org/standards/ovf`

15. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS). IEEE, Washington, DC (2003)
16. Ghosh, R., Longo, F., Naik, V.K., Trivedi, K.S.: Quantifying resiliency of iaas cloud. In: SRDS, pp. 343–347. IEEE (2010)
17. Hermenier, F., Lèbre, A., Menaud, J.M.: Cluster-wide context switch of virtualized jobs. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC 2010. ACM, New York (2010)
18. Hines, M.R., Gopalan, K.: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE 2009, pp. 51–60. ACM, Washington, DC (2009)
19. Jin, H., Deng, L., Wu, S., Shi, X., Pan, X.: Live virtual machine migration with adaptive, memory compression. In: IEEE International Conference on Cluster Computing and Workshops, CLUSTER 2009, pp. 1–10 (September 2009)
20. Keahey, K.: From sandbox to playground: Dynamic virtual environments in the grid. In: Proceedings of the 5th International Workshop on Grid Computing (2004)
21. Keahey, K., Tsugawa, M., Matsunaga, A., Fortes, J.: Sky computing. IEEE Internet Computing 13, 43–51 (2009)
22. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
23. Lagar-Cavilla, H.A., Whitney, J., Bryant, R., Patchin, P., Brudno, M., de Lara, E., Rumble, S.M., Satyanarayanan, M., Scannell, A.: Snowflock: Virtual machine cloning as a first class cloud primitive. Transactions on Computer Systems (TOCS) 19(1) (February 2011)
24. Lowe, S.: Introducing VMware vSphere 4, 1st edn. Wiley Publishing Inc., Indianapolis (2009)
25. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (2008)
26. McNett, M., Gupta, D., Vahdat, A., Voelker, G.M.: Usher: An Extensible Framework for Managing Clusters of Virtual Machines. In: Proceedings of the 21st Large Installation System Administration Conference (LISA) (November 2007)
27. Nicolae, B., Bresnahan, J., Keahey, K., Antoniu, G.: Going back and forth: Efficient multi-deployment and multi-snapshotting on clouds. In: Proceedings of the 20th ACM International Symposium on High Performance Distributed Computing, HPDC 2011. ACM, New York (2011)
28. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID, Washington, DC, USA (2009)
29. Quesnel, F., Lebre, A.: Operating Systems and Virtualization Frameworks: From Local to Distributed Similarities. In: Cotronis, Y., Danelutto, M., Papadopoulos, G.A. (eds.) PDP 2011: Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing, pp. 495–502. IEEE Computer Society, Los Alamitos (2011)
30. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)

31. Ruth, P., Rhee, J., Xu, D., Kennell, R., Goasguen, S.: Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In: IEEE International Conference on Autonomic Computing, ICAC 2006 (June 2006)
32. Sotomayor, B., Montero, R., Llorente, I., Foster, I., et al.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing 13(5) (2009)
33. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking 11(1), 17–32 (2003)
34. Tsugawa, M., Fortes, J.: A virtual network (vine) architecture for grid computing. In: International Parallel and Distributed Processing Symposium, p. 123 (2006)
35. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 22(1), 41–53 (2004)