Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed?
Adrien Lebre, Jonathan Pastor, Marin Bertier, Frédéric Desprez, Jonathan Rouzaud-Cornabas, Cédric Tedeschi, Paolo Anedda, Gianluigi Zanetti, Ramon Nou, Toni Cortes, et al.

To cite this version:

HAL Id: hal-00854204
https://inria.hal.science/hal-00854204
Submitted on 26 Aug 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed?


RESEARCH REPORT
N° 8348
August 2013
Project-Team Ascola, Asap, Avalon, Myriads
Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed?

A. Lèbre∗†, J. Pastor∗†, M. Bertier‡, F. Desprez*, J. Rouzaud-Cornabas‡ C. Tedesch††, P. Anedda∥ G. Zanetti∥, R. Nou∗‡, T. Cortes**, E. Riviere†† T. Ropars‡‡

Project-Team Ascola, Asap, Avalon, Myriads

Research Report n° 8348 — August 2013 — 23 pages

* INRIA, France, Email: FirstName.LastName@inria.fr
† LINA / Mines Nantes, France.
‡ IRISA / INSA de Rennes, France.
§ CC-IN2P3, CNRS, France.
¶ IRISA / Université de Rennes 1, France.
‖ Center for Advanced Studies, Research and Development in Sardinia (CRS4), Italy.
** Barcelona Supercomputing Center (BSC), Spain.
†† Université de Neuchâtel (UniNe), Switzerland.
‡‡ Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
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 larger and 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.

We claim that a disruptive change in UC infrastructures is required: UC resources should be managed differently, considering locality as a primary concern. 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 demand. Critical to the emergence of such locality-based UC (LUC) platforms is the availability of appropriate operating mechanisms. In this paper, 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, government and academic institutions to any idle resources that may be provided by end-users. Unlike previous researches on distributed operating systems, we propose to consider virtual machines (VMs) instead of processes as the basic element. System virtualization offers several capabilities that increase the flexibility of resources management, allowing to investigate novel decentralized schemes.

Key-words: Locality-Based Utility Computing, Peer To Peer, Self-*, Sustainability, Efficiency, Future Internet.
Informatique utilitaire, pourquoi et comment devons nous concevoir une nouvelle génération de centrales numériques?

Résumé :

Afin de supporter la demande croissante de calcul utilitaire (UC) tout en prenant en compte les aspects énergétique et financier, la tendance actuelle consiste à construire des centres de données (ou centrales numériques) de plus en plus grands dans un nombre limité de lieux stratégiques. Cette approche permet sans aucun doute de satisfaire la demande tout en conservant une approche centralisée de la gestion de ces ressources mais elle reste loin de pouvoir fournir des infrastructures de calcul utilitaire efficaces et durables.

Après avoir indiqué pourquoi cette tendance n’est pas appropriée, nous proposons au travers de ce rapport, une proposition radicalement différente. De notre point de vue, les ressources de calcul utilitaire doivent être gérées de manière à pouvoir prendre en compte la localité des demandes dès le départ. Pour ce faire, nous proposons de tirer parti de tous les équipements disponibles sur l’Internet afin de fournir des infrastructures de calcul utilitaire qui permettront de part leur distribution de prendre en compte plus efficacement la dispersion géographique des utilisateurs et leur demande toujours croissante. Un des aspects critique pour l’émergence de telles plates-formes de calcul utilitaire "local" (LUC) est la disponibilité de mécanismes de gestion appropriés. Dans la deuxième partie de ce document, nous défendons la mise en œuvre d’un système unifié gérant l’utilisation des ressources à une échelle sans précédent en transformant une infrastructure complexe et hétérogène en une collection d’équipements virtualisés qui seront à la fois plus simples à gérer et plus sûrs. En déployant un système de type LUC sur les cœurs de réseau, notre vision ultime est de rendre possible l’hébergement et la gestion de l’Internet sur sa propre infrastructure interne: un ensemble de ressources extensible et quasiment infini fourni par n’importe quel équipement constituant l’Internet, partant des gros nœud réseaux gérés par les ISPs, les gouvernements et les institutions académiques jusqu’à n’importe quelle ressource inactive fournie par les utilisateurs finaux. Contrairement aux approches précédentes appliquées aux systèmes distribués, nous proposons de considérer les machines virtuelles comme la granularité élémentaire du système (à la place des processus). La virtualisation système offre plusieurs fonctionnalités qui améliorent la flexibilité de la gestion de ressources, permettant l’étude de nouveaux schémas de décentralisation.

Mots-clés : Calcul utilitaire basé sur la localité, systèmes pair-à-pair, self-*, durabilité, Internet du futur
1 Introduction

Current Trends. The success of Cloud Computing has driven the advent of Utility Computing. However Cloud Computing is a victim of its own success: In order to answer the ever-increasing demand for computing resources, Cloud Computing providers must build data centers (DCs) of ever-increasing size. Besides facing the well-known issues of large scale platforms management, these new DCs have to deal with energy considerations that limit the number of physical resources that a single location can host. As a consequence, several proposals suggest to build next generation DCs in strategical locations close to the polar circle. However, this approach is far from solving all issues. 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 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.

- The unavoidable use of the Internet to reach distant platforms implies wasteful and costly overheads in addition to increase security flaws in several situations.

Although hybrid or federated Cloud solutions that aim at extending the available resource delivered by one Cloud with another one can partially tackle the two former points, the latter one requires a disruptive change in the way of managing UC resources. Deploying a broadcasting service of local events or an online service to order pizza at the edge of the polar circle leads to important overheads in terms of energy footprint, network exchanges as well as latency since it can be assumed that a vast majority of the users are located in the neighborhood of the event/the pizzeria. According to some projections of a recent IEEE report, the network traffic continues to double roughly each year. Bringing the IT services closer to the end-users will become soon crucial to limit the energy impact of these exchanges and to save the bandwidth of some links. Similarly, this notion of locality is also critical for the adoption of the UC model by applications that should deal with a large amount of data as getting them in and out from actual UC infrastructures may significantly impact the global performance.

The concept of micro data centers at the edge of the backbone may be seen as the complementary solution to hybrid platforms in order to reduce the overhead of network exchanges. However, the number of such micro data centers will stay limited and the question of how federating a large number of such facilities is still not solved.

---


Inria
At the same time, people are (and will be more and more) surrounded by computing resources, especially the ones in charge of interconnecting all IT equipments. Even though these small and medium-size facilities include resources that are sometimes barely used, they can hardly be removed (e.g. routers). As a consequence, several initiatives started investigating how they could be better leveraged to support the requirements and constraints of current IT usages. The concept of data furnaces [27] is one of the promising idea that seeks to mitigate the cost of operating network/computing resources by using them as sources of heat inside public buildings such as hospitals or universities.

Oversized Network Backbone. Figure 1 is a snapshot of the network weather map of RENATER\(^2\), the network backbone dedicated to universities and research institutions in France. It reveals several important points:

- It clearly shows that most of the resources are barely used (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 has been deployed and is renewed according to the demand: the density of points of presence (PoP) of the network 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 has been deployed in order to face disconnections, i.e. 95% of the PoPs can be reached by at least two distinct routes.

Instead of building and deploying dedicated facilities, we claim that next UC infrastructures should be tightly coupled with any facilities available through the Internet, starting from the core routers of the backbone, the different network access points and any small and medium-size computing infrastructures that may be provisioned by public and private institutions. Although it involves radical changes in the way physical and virtual resources are managed, locating and operating computing and data on facilities close to the end-users is the only way to deliver highly efficient and sustainable UC services.

From the physical point of view, network backbones such as the RENATER one provide appropriate infrastructures, that is, reliable and efficient enough to operate UC resources spread across the different PoPs. In a perfect model, UC resources may directly take advantage of computation cycles available on network servers, i.e. the one 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 network hub with a number of servers dedicated to host VMs. As we can expect that the distribution between network traffics and UC demands would be proportional, larger network hubs will be completed by 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 both the energy footprint of the network as well as the latency impact.

From the software point of view, the main challenge is to design a complete distributed system in charge of turning a complex and diverse network of re-

\(^2\)http://www.renater.fr

RR n° 8348
Our Big Idea. By designing an advanced system that offers the possibly to operate in a unified manner a large number of UC resources spread throughout distinct sites, ISPs as well as academic and private institutions in charge of operating a network backbone will be able 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 that are already present on each hub of their backbone.

The main objective of this paper is to describe how a new generation of highly efficient and sustainable UC can emerge through advanced system and P2P mechanisms. To this aim, we outline first the key objectives that should drive the design of these mechanisms. Second, we describe the premises of what could be a LUC Operating System, allowing end-users to launch virtualized environments (VEs), i.e. a set of interconnected VMs, throughout a distributed
infrastructure as simply as they are used to launch processes on a local machine, *i.e.* without the burden of dealing with resources availability or location.

In addition to consider *Locality* as a primary concern, the novelty of the LUC OS proposal is a new way of designing systems to operate UC resources. Unlike existing research on distributed operating systems that consider processes as the basic object they manipulate, a LUC OS will manipulate VMs throughout a federation of widely distributed physical machines. Virtualization technologies abstract hardware heterogeneity, and allow transparent deployment, preemption, and migration of VEs. 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-based techniques.

**Paper Outline.** Section 2 describes the key objectives of a LUC OS and the associated challenges. Section 3 explains why our vision differs from actual and previous UC solutions. In Section 4 we present how such a unified system may be designed by delivering the premises of the DISCOVERY system, an agent based system enabling distributed and cooperative management of virtual environments through a large-scale distributed infrastructure. Future work as well as opportunities are addressed in Section 5. Finally Section 6 concludes this paper.

## 2 Overall Vision and Major Challenges

Similarly to traditional operating systems (OSes), a LUC OS will be composed of a significant number of mechanisms. Trying to identify all of them and establishing how they interact is an on-going work (see Section 4). However, in order to reach the goal of delivering a unified system in charge of operating a complex and diverse infrastructure into a LUC platform, we have already identified that the following objectives should 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, including small and medium-sized computing facilities as well as any idle resource that their owner would make available. These resources might be highly volatile, especially if the LUC infrastructure allows to include resources hosted by end-users.

- **Reactivity:** To deal with the infrastructure’s dynamicity, a LUC OS should swiftly handle events that require performing particular operations, either on virtual or on physical resources, with the objective of maximizing the system utilization while meeting QoS expectations of VEs. Reconfiguring VEs over distributed resources, sometimes spread across wide area networks, or moving VMs, while preserving their active connections, are examples of operations that should be performed as fast as possible.

- **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 con-
tinue working despite node failures and network disconnections and (ii) to
provide snapshotting as well as high availability mechanisms for VEs.

• Sustainability: Although the LUC approach natively reduces the energy
footprint of UC services by minimizing the network impact, it is important
to go one step further by considering energy aspects at each level of a LUC
OS system and propose advanced mechanisms in charge of making an
optimal usage of each source of energy. To achieve such an objective, data
related to the energy consumption of the VEs and the computing resources
as well as the environmental conditions (computer room air conditioning
unit, localization of the site, etc.) should be taken into account by the
system.

• Security and Privacy: Similarly to resiliency, security affects the LUC OS
itself and the VEs running on it. For the LUC OS security, the goals
are to (i) avoid attacks on the P2P layers, (ii) create trust relationships
between different locations, (iii) include security decision and enforcement
points in the LUC OS and (iv) make them collaborate to provide a secured
infrastructure. For the VEs security, we need to provide users with a way
to express their security requirements. The LUC OS security decision and
enforcement points will collaborate to enforce these requirements.

In addition to the aforementioned objectives, targeting a distributed sys-
tem where VM is the elementary granularity requires to deal with important
issues regarding the management of the VM images. Managing VM images in
a distributed way across a WAN is a real challenge that will require to adapt
state-of-the-art techniques such as replication and deduplication. Also, sev-
eral mechanisms of a LUC OS must take into account VM images’ location,
for instance to allocate the right resources to a VE or to request VM images
prefetching to improve deployment performance or VM relocations.

Amongst the numerous scientific and technical challenges that should be
addressed, the lack of a global view of the system introduces a lot of complex-
ity. In order to tackle it while addressing the above-mentioned challenges, we
claim that internal mechanisms of a LUC OS should be based on decentralized
mechanisms specifically designed for it. These techniques should provide mech-
nisms which are fully decentralized and autonomous, so to allow self-adapting
control and monitoring of complex large-scale systems. Simple locality-based
actions by each of the entities composing the system can lead to the global emer-
gence of complex and sophisticated behaviors, such as the self-optimization of
resource allocation, or the creation of decentralized directories. These tech-
niques are starting to be used in well-known large systems. As an example, the
Amazon website relies on its Dynamo service [10], based on fully-decentralized
mechanisms, to create massive scale distributed indexes and recover from data
inconsistencies. Facebook’s Cassandra massive scale structured store [25] also
leverages P2P techniques for its core operation. In a LUC OS, decentralized and
self-organizing overlays will enable to reflect the current state of both virtual
and physical resources, their characteristics and availabilities. Such information
is mandatory to build higher mechanisms ensuring the correct execution of VEs
throughout the whole infrastructure.
3 Background

Several generations of UC infrastructures have been proposed and still co-exist [15]. However, neither Desktop, nor 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. Because 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 the standardization of the interfaces for supporting cooperation and resource sharing over Cloud federations, their propositions are incremental improvements of the existing UC models. From our point of view, hybrid and cloud federation investigations are comparable in some ways to previous works that have been done for Grids and where, the purpose of the Grid middleware is to interact with each resource management system composing the Grid [12, 40, 44]. By taking into account network issues in addition to traditional computing and storage concerns in Cloud Computing systems, the European SAIL project is probably the one which targets the biggest advances in comparison with previous works on Grid systems. Concretely, this project investigates new network technologies in order to provide end-users of hybrid/federated Clouds with the possibility to configure and virtually operate the network backbone that interconnects the different sites they use [32]. 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 latency and mobility concerns) [11]. 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 network hub of backbones and operated through a unified system, i.e. 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 XtremOS Project [31]. Although this project was sharing 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 distributed OS, manipulating VEs instead of processes and considering locality as a primary concern. Referred as a LUC Operating System, such a system will include most of the mechanisms that are common to actual UC management systems [11, 33, 4, 5, 25, 30]. However, each of them will have to be rethought in order to leverage P2P algorithms. While largely unexplored for building
operating systems, P2P/decentralized-based techniques have the potential to achieve the scalability required for LUC systems. Using all these technologies for establishing the foundation mechanisms of massive-scale distributed operating systems will be a major breakthrough from current static, centralized or hierarchical management solutions.

4 Premises 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 the implementation of a first LUC OS proposal: the DISCOVERY system (DIStributed and COoperative Management of Virtual EnviRonments autonomouslY). 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.

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 to efficiently use the LUC resources. In our context, efficiency means that a good trade-off is found between satisfying user’s expectations, ensuring reliability, reactivity as well as availability of the services while limiting the energy consumption of the system and providing scalability. We propose thus to leverage P2P techniques, that allow self-* properties, such as self-adaptation and self-repairing of overlays. To reduce the management complexity as well as the design and the implementation of the different mechanisms that are mandatory, we strongly support to use micro-kernel concepts. Such an approach should enable to design and implement services at higher level while leveraging P2P mechanisms at the lower ones. Furthermore, to address the different objectives and reduce the management complexity, we also underline that self-* properties should be present at every level of the system. We think that relying on a multi-agent peer-to-peer system is the best solution to cope with the scale as well as the network disconnections that may create temporary partitions in a LUC platform.

In DISCOVERY, each agent has two purposes: (i) maintaining knowledge base on the LUC platform composition (ii) ensuring the correct execution of the VEs. Concretely, the knowledge base will consist of overlays that will be used for the autonomous management of the VEs life cycle. 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 need to rely on dedicated mechanisms that can be classified as follows:

- Mechanisms related to physical resource localization and monitoring,
- Mechanisms related to VEs management,
- Mechanisms related to the VM images management,
Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed?

• Mechanisms related to reliability,
• Mechanisms related to security and privacy.

4.2 Resource Localization and Monitoring Mechanisms

Keeping in mind that DISCOVERY should be designed in a fully distributed way, most of the mechanisms should build on top of overlays in order to abstract changes that occur at the physical levels. The specific requirements of this platform will lead to develop a particular kind of overlays, having minimalism in mind.

More concretely, the first step is to design, at the lowest level, a first overlay layer intended to abstract out the details of the physical routes and computing utilities, while satisfying the basic topology requirements needed – 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 the minimalistic design envisioned.

We need to separate far nodes from close nodes. An envisioned way to achieve this is to group nodes according to their physical proximity. Simple message exchanges are enough to discover the underlying topology so each node keeps a link to its closest nodes (for instance all nodes I can reach in less than a given threshold). Each node doing so in parallel, a number of clusters are created inside which we can ensure a certain level of locality – all nodes inside this group can communicate with each other very efficiently. Given the number of nodes in these groups, the inner topology of a group can either rely on very simple graphs, such as rings, or more connected graphs, to accelerate the dissemination and retrieval of information in the group. Note that, still based on simple gossiping techniques, such graphs can be easily maintained as the network’s conditions change. For instance, if a link becomes overloaded, the other nodes will react to this change by removing nodes with which they communicate through this link from their local group. Such groups are exemplified on Figure 1 (for the west part of the platform).

As we can see on Figure 1 it follows from the way the overlay is built that some nodes may be a member of several groups. The Nantes site for instance is part of three of these groups, given its physical position. More generally, the overlay will take the shape of a set of groups with some bridges between the groups. Note that several bridges can interconnect two local groups. Thus, any request first goes through local nodes allowing for its local processing, and avoiding the need for global coordination mechanisms.

To be able to localize a particular VM, the system has to be able to route a request from any node to any other node. This functionality is the classic problem of P2P overlays, which is traditionally solved in structured overlay networks by maintaining a routing table on each node, and by a flooding mechanism or with random walks in unstructured overlays. These techniques are usually designed for very large scale networks with different guarantees and costs. Here, given the particular “intermediate” scale of the platform, and its specific requirements, we believe that these existing techniques are too powerful. As we want to design a minimalistic overlay, there is room to try to design a new routing technique specifically fitting our requirements. The aim is then to maintain, in all groups, information about the distance between this group and close groups.

RR n° 8348
in terms of number of bridge nodes to go through. Hence, the system will be able to route quickly between close groups. The routing of requests between far groups will be based on a random decision when no information are available, but oriented by the aim of going away from the request’s source.

This overlay will provide the basic building block of the platform, on which will rely higher level overlays and functionalities, which are described in the following sections.

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: some VMs must be on the same node/site in order to cope with performance objectives while others should not be collocated in order to ensure high-availability criteria. 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 VM if need be...
or simply to retrieve the location of a particular VE. Most of these mechanisms are integrated on current UC platforms. However as mentioned, they should be revisited to correctly run on the infrastructure we target (i.e., in terms of scalability, resiliency and reliability). To this aim, the DISCOVERY system relies on the aforementioned P2P mechanisms. As a first example, placing the VMs of a VE requires to be able to find available nodes that fulfills the VM needs (in terms of resource needs 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, simple navigation ensures that the request will encounter a bridge eventually, leading to the exploration of further nodes. This navigation goes on until one sufficiently available 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, in order to ensure particular needs of a VM, it can be necessary to relocate other ones. According to the predefined constraints of VEs, some VMs might be relocated on far nodes while other would prefer to be suspended. Such a mechanism has been deeply studied and validated in the DVMS mechanism [2, 3]. DVMS (Distributed Virtual Machines Scheduler) enables to dynamically schedule a significant number of VMs throughout a large-scale distributed infrastructure while guaranteeing VM resource expectations.

A second example regards the configuration of the network elements of a VE. Although it might look simple, assigning the right IPs to each VM as well as maintaining the intra-connectivity of a VE becomes a bit more complex than in the case of a single network domain, i.e., a mono-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 [19]. However, the definition of software network becomes more and more important. Investigating proposals such as the Open vSwitch project [36] looks a promising direction to solve such issue.

### 4.4 VM Images Management

In a LUC infrastructure, VM images could be deployed in any place from any other location, but being in a large-scale, heterogeneous and widely spread environment makes the management of VM images more difficult than more conventional centralized repositories. At coarse grain, the management of the VM images should be (i) consistent with regards to the location of each VM throughout the DISCOVERY infrastructure and (ii) reachable in case of failures. The envisioned mechanisms dedicated to the management of the VM images have been classified into two sub-classes. First, it will be mandatory to deliver appropriate mechanisms to efficiently upload and replicate VM images among a high number of nodes in order to ensure efficiency as well as reliability. Second, the DISCOVERY system should include specific mechanisms devoted to the scheduling of the VM image transfers. Advanced policies are important to improve the efficiency of each transfer that may occur either at the boot time or during VM relocations.

Regarding storage and replication mechanisms, an analysis of an IBM Cloud concludes that a fully distributed model using P2P technology is not the best choice to manage VM images as the number of instances of the same VM image
is rather small [34]. However, central or hierarchical solutions are not suited for the infrastructure we target. Consequently, an augmented P2P solution working with replicas and deduplication will have to be investigated in order 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. This 30% can become a 30% space reduction, a 30% increased reliability or a 30% of speed increase. Depending on the situation, we should decide to go from one scenario to another.

Regarding the scheduling mechanisms, it has been shown that a storage system with VMs being used by I/O intensive tasks can increase boot time from 10 to 240 seconds [42]. Some actions like providing the image chunks needed to boot first [43], defining a new image format, and pausing the rest of the I/O operations, can provide a performance boost and limit the overhead that is still observed in commercial Clouds [29].

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

4.5 Reliability Mechanisms

In a LUC, failures will be much more frequent than in actual UC platforms. Furthermore, since resources could be highly distributed, the expected mean time to repair failed equipments might be much larger than in other UC platforms. For all these reasons, a set of dedicated mechanisms should be designed in order to provide fully transparent failure management with minimum downtime.

Ensuring the high availability of the DISCOVERY system requires being able to autonomously restart any service by relocating it on a healthy agent each time it is mandatory. To avoid losing or corrupting important information regarding the state of the system, a Cassandra-like system [25] is required to provide a reliable and highly available back-end for stateful services.

Regarding VEs reliability, a first level of fault tolerance can be provided by leveraging VMs snapshotting capabilities. Periodical snapshots will allow restarting the VE from its last snapshot in the event of a failure. Performing VM snapshotting in a large-scale, heterogeneous, and widely spread environment is a challenging task. However, we believe that adapting recently proposed ideas [33] in this field 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 [35]. To implement VM replication in a WAN, solutions to optimize synchronizations between replicas [20, 39] should be investigated. Also, we think that a LUC has the major advantage over other UC platforms, that it is tightly coupled with the network infrastructure. As such, we can expect low latencies between nodes and so, to be able to provide strong consistency between replicas while achieving acceptable response time for the replicated services.

Reliability techniques will of course make uses 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 navigating scheme where at least
one bridge is encountered. Monitoring a replica can then be done by having a watcher in the same local group as the replica.

4.6 Security and Privacy Mechanisms
As overlays are fundamentals for all DISCOVERY mechanisms, the first challenge is to ensure that they are not compromise. Recent advances [14] might enable to tackle such concerns.

The second challenge will consist in providing end-users with a way to define their own security policies and to ensure that such policies are enforced. The expression of these requirements itself is a complex task. To ease the expression of these policies, we are currently defining a domain specific security language that defines high-level security requirements [10, 26]. These security policies will be enforced by security decision and enforcement points (SDEPs) during the whole lifetime of the VE. Implementing such SDEP mechanisms in a distributed fashion will require to conduct specific research as they are currently only prospective proposals for classic UC infrastructures [8, 41]. The challenge will consist in investigating whether such proposals can be adapted to the LUC infrastructure by leveraging appropriate overlays.

Finally, as traditional distributed systems, DISCOVERY should ensure the good behavior of resources and users. To this aim, authentication and certification mechanisms should be provided.

4.7 Toward a First Proof-of-Concept
The first prototype is under heavy implementation. 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 [24] in association with actors model [6]. This enables to easily switch between a push and a pull oriented model according to our needs.

Our preliminary studies showed that a common building block is mandatory to handle resilience 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 Figure 3 and 4). 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 P2P overlay as well as when neighbors change. Considering that several overlays may co-exist through 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
The Supervisor actor monitors all the actors it encapsulates while the Peer actor acts as an interface between the services and the overlay.

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 Figure 4, 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 overlays’ maintenance. In the example, 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.

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 scale, failures are the norm rather than the exception, hence we decided that each actor will be monitored by a Supervisor Actor (see Figure 3). DISCOVERY services are under the supervision of the DISCOVERY agent: this design makes it possible to precisely define strategy that will be executed in case of service failures. This will be the way we introduce self-healing and self-organizing properties to the DISCOVERY system.
This building block has been fully implemented by leveraging the SCALA/akka framework.

As a proof of concept, we are implementing a first high level service in charge of dynamically scheduling VMs across a LUC infrastructure by leveraging the DVMS proposal (see Section 4.3). The low-level overlay that is currently implemented, is a robust ring based on the Chord algorithm.

To validate the behavior, the performance as well as the reliability of our POC, we rely first on the Simgrid toolkit. Simgrid has been recently extended to integrate virtualization abstractions and accurate migration models. Simulations enable us to analyze particular situations and get several metrics that cannot be easily monitored on a real platform. Second, results obtained from simulations are then compared to real experiments on the Grid'5000 platform. Grid'5000 provides a testbed supporting experiments on various types of distributed systems (high-performance computing, grids, peer-to-peer systems, Cloud Computing, and others), on all layers of the software stack. The core testbed currently comprises 10 sites geographically spread across France. For the Discovery purpose, we developed a set of scripts that enables to deploy in a one-click fashion a large number of VMs throughout the whole infrastructure.

By deploying our POC on each node and by leveraging the VM deployment scripts, we can evaluate real scenario usages by conducting 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.

5 Future Work/Opportunities

5.1 Geo-Diversification as a Key Element

The Cloud Computing paradigm is changing the way applications are designed. In order to benefit from the elasticity capability 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 is one of the promising systems for elastic Cloud applications. At the same time, a few projects have started investigating distributed/collaborative way 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 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. Image sharing system such as Google Picasa or Flickr are examples of applications where leveraging locality will enable to limit network exchanges: Users could

4 Code is available at: https://github.com/BeyondTheClouds

5 http://www.akka.io

RR n° 8348
upload their images on a peer close to them and images would be transferred to other locations only when required (pulling vs. 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.

5.2 Energy, a Primary Concern for Modern Societies

The energy footprint of current UC infrastructures and more generally of the whole Internet is a major concern for the society. By its design and the way to operate it, a LUC infrastructure will have a smaller impact. Moreover, the LUC proposal is an interesting way to deploy the data furnaces proposal [27]. Concretely, following the smart city recommendations (i.e. delivering efficient as well as sustainable ICT services), the construction of new districts in metropolises may take the advantage of each LUC/Network PoP in order to heat buildings while operating UC resources remotely thanks to a LUC OS. Finally, this idea might be extended by taking into account recent results about passive data centers, such as solar-powered micro-datacenters[6]. The idea behind passive computing facilities is to limit has 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 product as a source of energy. Combining such ideas with the LUC approach would allow reaching an unprecedented level of energy efficiency for UC platforms.

6 Conclusion

Cloud Computing has entered our everyday life at a very high speed and huge scale. 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 minimized. 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. Amongst others, resource management as well as scale and heterogeneity of modern IT environments are two of the research issues highlighted in the recent Advance in Clouds report[7]. In this paper, we outlined that in addition to these concerns, the actual model of UC is limited by intrinsic issues. Instead of the current trend 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 gen-

\[\text{http://parasol.cs.rutgers.edu}\]
eration of Utility Computing that better matches the Internet structure thanks to 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 mechanism 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 have to be computed locally and 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 network one. This connection between these two communities has already started with the different activities around Software Defined Network and Network as a Service. This may result in the long view 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 of course 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 wide area networks operated. However we are convinced that such an approach will pave the way towards highly efficient as well as sustainable UC infrastructures, coping with heterogeneity, scale, and faults.

References


[38] Flavien Quesnel, Adrien Lèbre, and Mario Südholt. Cooperative and Reactive Scheduling in Large-Scale Virtualized Platforms with DVMS. Concurrency and Computation: Practice and Experience, June 2012.


