SlapOS: a Multi-purpose Distributed Cloud Operating System Based on an ERP Billing Model

Jean-Paul Smets-Solanes*, Christophe Cérin† and Romain Courteaud*

* *Nexedi SA,*
270 boulevard Clémenceau,
59700 Marcq-en-Baroeul, France.
Email: jp@nexedi.com
†*Université de Paris 13, PRES Sorbonne Paris Cité*
LIPN UMR CNRS 7030,
99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
Email: christophe.cerin@lipn.univ-paris13.fr

***Abstract*—SlapOS is an open source grid operating system for distributed cloud computing based on the moto everything is a process. SlapOS combines grid computing and Enterprise Resource Modeling (ERP) to provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) through a simple, unified API which one can learn in a matter of minutes. Thanks to its unified approach and modular architecture, SlapOS has been used as a research testbed to benchmark NoSQL databases and optimize process allocation over intercontinental Cloud. SlapOS opens new perspectives for research in the area of resilience and security on the Cloud.**

***Keywords*-Service Architecture, Enterprise Level Transformation, Business Process Management and Integration, Business Grid and Cloud Computing.**

I. Introduction

Cloud Computing is traditionally divided in three market segments: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Cloud Computing standardization groups, government funded research projects and corporate marketing presentations often introduce Cloud Computing through a 3-layer approach. IaaS provides virtual machines and storage. PaaS is built on top of IaaS and provides core services such application servers and databases. SaaS is built on top of PaaS and provides end-user applications such as Content Management System (CMS) or Customer Relation Management (CRM) software. The traditional layered approach implicitly supposes that the IaaS layer of Public Clouds is implemented by very large server farms, which are supposed to provide optimal efficiency through economies of scale and automation. The IaaS layer of Private Clouds is implicitly supported by expensive Storage Area Networks (SAN) hardware.

There are several efforts already under way, including the Distributed Management Task Force (DMTF) Open Cloud Standards Incubator, the Open Grid Forum's Open Cloud Computing Interface working group, and the Storage Network Industry Association Cloud Storage Technical Work Group. In France the Free Cloud Alliance promotes the first Open Source Cloud Computing Stack which covers both Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) with a consistent set of technologies targetted at high performance and mission critical applications. A great resource to see the spectrum of cloud standards activity can be found at the OMG's cloud-standards.org wiki.

With the increasing adoption of IPv6, 1 Gbps fiber to the home, multi-core CPUs and Solid State Disks, the traditional view on Public Cloud based on very large server farms, or the traditional view on Private Cloud based on corporate Storage Area Networks, is no longer as relevant as it used to be. A new form of Cloud massively distributed cloud can be implemented nowadays to provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) with new levels of cost efficiency, drastically reduced environmental impact and higher protection of citizen Freedom. It is now possible to implement Cloud Computing through a myriad of servers located in everyone's home or in every worker's desk. Internet transit marginal cost becomes zero, distributed storage software is used instead of SAN. Energy is saved by eliminating the need for air cooling and by reusing natural server dissipation for house or office heating. Citizen Freedom is better protected by preventing data to be centralized under control of single entity, as it is now required for example by Law in India [XX]. Massively distributed cloud is already acknowledged and supported by the OW2 Consortium[1], by the Free Cloud Alliance[2] and by the Free Software Foundation[3].

[1]http://www.ow2.org
[2]http://www.freecloudalliance.org/
[3]http://www.fsf.org/

We will introduce in this article SlapOS, the first open source operating system for Distributed Cloud Computing. SlapOS is based on a grid computing daemon called slapgrid which is capable of installing any software on a PC and instantiate any number of processes of potentially infinite duration of any installed software. Slapgrid daemon receives requests from a central scheduler the SlapOS Master which collects back accounting information from each process. SlapOS Master follows an Enterprise Resource Planning (ERP) model to handle at the same time process allocation optimization and billing.

According to the classification in [2] , the paper is related to Service Architecture (how to design and build an 'operating system' for the grid), Enterprise Level Transformation (what we consider as best practices for transforming the main activity of the company), Business Process Management and Integration (how to organize and develop your business according to the use of Open Source Software), Business Grid and Cloud Computing (how to enhance your business with Cloud Technology).

The organization of the paper is as follows. Section II is an overview of the SplapOS project. Section III provides with details about the SlapOS implementation. Section IV-B is a full description of the developper and accounting models. Section V is related to an example about SlapOS use case. Section VI is about research questions about the design and the future of SlapOS. Section VII concludes the paper.

II. A Grid Approach to Cloud Computing

SlapOS is an open source Cloud Operating system which was inspired by recent research in Grid Computing and in particular by BonjourGrid [3]–[5] a meta Desktop Grid middleware for the coordination of multiple instances of Desktop Grid middleware. It is based on the moto that "everything is a process". SlapOS is now an OW2 project. Figure 1 shows the current architecture.

SlapOS defines two types of servers: SlapOS Nodes and SlapOS Master. SlapOS Nodes can be installed inside data centers or at home. Their role is to install software and run processes. SlapOS Master acts as a central directory of all SlapOS Nodes, knowing where each SlapOS Node is located and which software can be installed on each node. The role of SlapOS Master is to allocate processes to SlapOS Nodes.

A. SlapOS Architecture

SlapOS Nodes and SlapOS Master exchange are interconnected through the HTTP and XML based SLAP protocol. SlapOS Master sends to each SlapOS Node a description of which software should be installed and executed. Each SlapOS Node sends to SlapOS Master a description of how much resources were used during a given period of time for accounting and billing purpose.

From a user point of view, SlapOS Node looks like an online shop for Cloud Computing resources. The user connects to

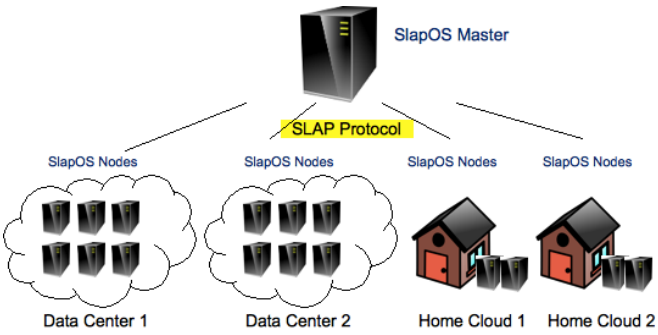

Figure 1. The SlapOS Architecture

SlapOS Master through a simplified front end, selects which software he or she needs. SlapOS Master then allocates the software onto a SlapOS Node and provides the connection information to the user. The allocated software can be of any type: virtual machine, database server, application server, web cache front end, etc.

Figure 2. An example of SlapOS front-end

B. An example of SlapOS front-end

From a developer point of view, see Figure 2, SlapOS is a simple and universal API to create instances of any software daemon through a programmatic interface. The sample code bellow shows how a developer can request a new instances of a memcache server by invoking the request method of SlapOS API. Memcache [19] is a widely adopted key-value store protocol which is used to cache values in large scale web infrastructure. It is usually installed and configured by system administrators using packaging systems such RPM or DEB. In this example, a single method call does in a few seconds what a human system adminstrator would have done in few minutes at best.

A sample source code to request a process is shown on Figure 3.

```
> python
from slapos.slap import slap
import memcache

# Register to SlapOS Master
s = slap()
s.initializeConnection("https://nexedivifib1.dyn.majimoto.net:40443/erp5/portal_slap")
cp = s.registerComputerPartition("nexedivifib8", "slappart197")

# Request a new instance of memcached server
ncp = cp.request(
https://nexedivifib1.dyn.majimoto.net:40443/erp5/web_site_module/erpypi/memcached_softwa
re_profile-0.0.3.cfg", "memcached", "memcached server 1")

# Wait a few seconds and connect to the memcached server
connection = ncp.getConnectionDict()
mc = memcache.Client(['%s:%s' % (connection['ip'], connection['tcp_port'])])

# Test memcached server
mc.set("some_key", "Some value")
value = mc.get("some_key")
```

Figure 3. Example of source code

III. Current implementation of SlapOS

SlapOS is implemented as an extension of widely adopted open source software: GNU/Linux, Buildout [20] and Supervisord [21] and as depicted on Figure 4. The only new software introduced by SlapOS is Slapgrid, a daemon in charge of implementing the SLAP protocol on each SlapOS Node.

A. SlapOS Kernel

Each time slapgrid receives a request from SlapOS master to install a software, it downloads a description of that software in the form of so-called buildout profile. It then runs the buildout boostrap process to install the software. Buildout is a Python-based build system for creating, assembling and deploying applications from multiple parts, some of which may be non-Python-based. Buildout can be used to build C, C++, ruby, java, perl, etc. software on Linux, MacOS, Windows, etc. Buildout can either build applications by downloading their source code from source repositories (subversion, git, mercurial, etc.) or by downloading binaries from package repositories (rpm, deb, eggs, gems, war, etc.). Buildout excels in particular at building applications in a way which is operating system agnostic and to automate application configuration process in a reproduceable way.

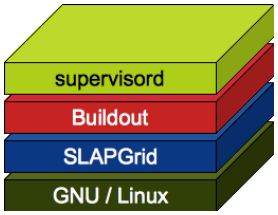

Figure 4. SlapOS kernel

Each time slapgrid receives a request from SlapOS master to run a software as a new process, it calls first buildout to create all configuration files for that process then delegates to supervisord the execution of the process. Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems. It provides a higher abstraction and flexibility than traditional sysinit.

After some time, a typical SlapOS Node will include multiple software applications (see Figure 5) and, for each software application, multiple instances, each of which running in a different process. For example, both Mediawiki and OS Commerce could be installed onto the same SlapOS Node, with six instances of each being run as processes. By running software instances as processes, rather than by creating a virtual machine for each software instance as one would do with Amazon EC2, SlapOS is able to use hardware resources and RAM in particular more efficiently.

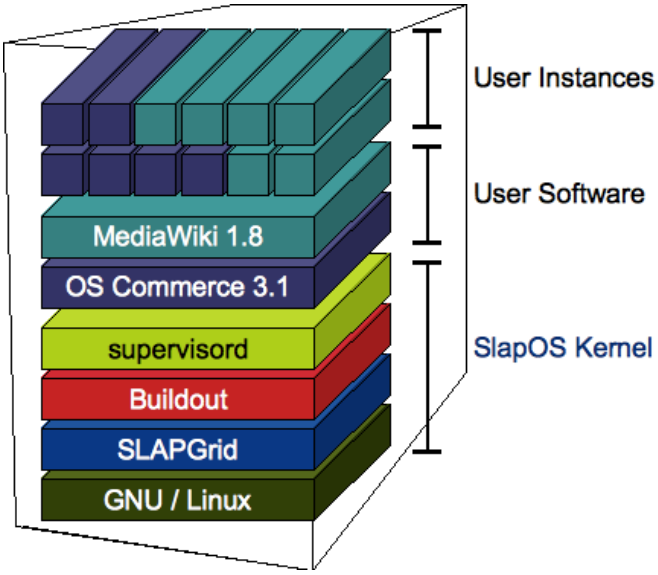

Figure 5. SlapOS implementation

B. Details about the implementation

SlapOS Master (see Figure 6) runs ERP5 Cloud Engine, a version of ERP5 open source ERP capable of allocating processes in relation with accounting and billing rules. Initial versions of SlapOS Master were installed and configured by human. Newer versions of SlapOS Master are implemented themselves as SlapOS Nodes, in a completely reflexive ways. A SlapOS Master can thus allocate a SlapOS Master which in turn can allocate another SlapOS Master, etc.

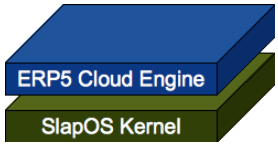

Figure 6. SlapOS Master

IV. SLAPOS MODEL

A. Developper model

For the end user or the developer, SlapOS API is based on a single core API method: request. This method is used to extend the buildout framework. It adds to buildout the possibility to remotely deploy and configure buildout based software. Additional SlapOS API methods include initializeConnection and registerComputerPartition. They are used to register to SlapOS master and are normally hidden to the developper of buildout profiles.

What is important to notice at this point is that SlapOS, through the software_release_url parameter, can build, install and instanciate about any type of software as long as the software is described through a buildout profile which can be stored almost anywhere on the Web and referred to through a URL. There are already more than a thousand a buildout profiles for many different types of software including storage and virtual machines (IaaS), application server (PaaS) and business applications (SaaS). Another important aspect of the request method is that it defines a paremeter called instance_type which can be used to specify which service of a given software should be instanciated and started. This is useful for encapsulating all services provides by a software into a single buildout software descriptor, rather than developing multiple ones and thus preventing mutualisation of shared libraries. A Qemu buildout profile for example provides both Kvm based virtual machines under the instance type kvm and nbd based block server under the instance type nbd. Both services are actually available as part of the same qemu binary, although they are mostly independent and can be allocated separately. Similarly, instances types are useful for application servers and distribute storage to differentiate master nodes vs. slave nodes in a non symetrical cluster.

The extra parameters of the request method can be used to define Service Level Agreement (SLA) information and software configuration. SLA parameters can be used to specify that the software instance should be allocated on the same host, the same LAN, the same country, the same continent or on a different host, different LAN, different country or different continent. By allocating software instances on different LANs and different continents, resilience of a Cloud hosted environment can be increased. By allocating software instances on the same LAN or host, performance of distributed storage can be optimized.

Extra parameters can also be used to specify the configuration of the software instance. The number of backend nodes of a Web application is typically specified in this way. Elasticity is provided in SlapOS by changing the value which defines the number of backends. Automatic elasticity is provided by combining performance monitoring and calls to the request method with a number of nodes which depends on the current status of performance monitoring, following a control system pattern.

B. Accounting model

The accounting model of SlapOS is an application of ERP5 Unified Business Model[4] (UBM) [12]. ERP5 defines 5 concepts which match with any business problem: Resource, Movement, Item, Node and Path. The ERP5 model is based on the idea that everything in business is a matter of flow, stock, traceability and planning of future flows. Until now, this model has been applied successfully to banking, accounting, manufacturing, trading, HR, document management, open data management, patient records. The use of ERP5 model in SlapOS comes from the fact that allocating and billing processes in computers is not very different from allocating containers in ships. In both cases, once it has been allocated, it is difficult to move it to another computer (vs. another ship).

In the case of SlapOS, Nodes are the people and organisations who register to the SlapOS portal. Once they are registered, they can request any type of resource. They can also add to a SlapOS global Cloud their own servers in order to contribute to total amount of processing resources, just like SETI@Home did. SlapOS Resources - in the sense of ERP5 UBM - are the Software Products such as MySQL, KVM, Xwiki, Nuxeo, ERP5, Sheepdog which people and organisations can subscribe to and for which they will be later invoiced. It is important to notice that Software Products do not have a version of release number. It is just a marketing or commercial name for a service.

Movements in the sense of UBM represent the billing information such as the amount of GHz, the amount of storage, the number of users which have been used over a period of time by each process in the SlapOS cloud. Then comes the core part of SlapOS: traceability. In order to know which version of which software is being run on which computer and IP address for who and with which parameters, SlapOS defines 5 types of UBM Item: Computer, Computer Partition, Software Release, Software Instance, Subscription Item.

A Computer Item is created each time a server is added to the SlapOS cloud. Each Computer Item is divided into a given number of so-called Computer Partition. The number of independent services (Software Products) which can be instanciated on a given Computer is at most equal to the number Computer Partitions of the Computer. Computer Partitions are usually implemented as subdirectories of the Computer. On an IPv6 network, each Computer Partition can be associated to a different IPv6 address. On an IPv4 network, each Computer Partition can be associated to a different port range for the same IPv4 address. Some Computer Partitions can also associated to a tap virtual ethernet interface which is then bridged and used by virtual machines.

[4]http://www.erp5.org/UnifiedBusinessModel

Some Computer Partitions are associated to a physical disk partition, which can be used by virtual machines for higher access performance through Virtio[5]. The partitioning process in SlapOS is handled by optional utilities provided by SlapOS. There is no obligation to use such utilities. Any naming of directories, allocation of IP addresses or port range, disk partitions or network interfaces is acceptable as long as it can be described in SlapOS standard XML file which the slapgrid daemon uploads to SlapOS master.

A Software Release Item defines precisely how to build a Software and install it on a Computer. Currently, SlapOS uses buildout profiles to define a Software Release. Multiple Software Release can exist for the same Software Product. For example, the MySQL Software Product exists in Release 5.0, Release 5.1, Release 5.0 patched with Senna Full Text, etc. A major strength of SlapOS is its ability to provide complete freedom to extend the number of Software Release and Software Products. The Software Release Item is the equivalent in SlapOS master of the software_release_url of the request method.

A Software Instance Item contains all parameters to configure an instance of Software Release. It also defines the type of service to run from Software Release. It is the equivalent in SlapOS master of the instance_type, instance_id and kw parameters of the request method.

The Subscription Item is used to group all accounting information for a given consistent set of Software Instances. It is also used to generate daily, weekly, monthly or yearly subscription billing. It plays a key role for the production of invoices. SlapOS movements which are either collected from each SlapOS node or generated by Subscription Item are collected on a monthly base, grouped by Subscription Item and by Customer, so that each customer can receive an invoice and understand for which service (Software Product), which release (Software Release) and configuration (Software Instance), on which computer (Computer Partition) and for what amount of which resource (Movement) he or she was invoiced.

The information needed to produce a detailed invoice is actually the same as the information needed to provision services on the Cloud. This is another reason why in SlapOS, accounting and provisionning are handled by the same software component, both to reduce duplication of information, increase consistency and make the system more simple.

C. Future evolutions of SlapOS models

SlapOS uses buildout and a single URL to describe how to build and install software. This approach could be extended in different ways. Ideally, it should be possible to install software using other build systems or even by using packages (DEB, RPM). This is not incompatible with SlapOS approach as long as the software which is installed provides itself a way to create multiple software instances through buildout or through any other type of templating system. It should also be possible to specify multiple URLs, embedded for example in a single file containing a list of URLs. This would be useful to specify different releases of the same software which are all considered as acceptable (ex. MySQL 5.0, MySQL 5.1, etc.).

Another evolution could be the encapsulation of best Cloud practices into either buildout recipes (for modularity) or SlapOS API (for simplicity). For example, self-controlled elasticity based on performance monitoring and feedback could be encapsulated into a class and reused by different Software Product. SlapOS will first consider the approach of buildout recipes which provides maximum flexibility before considering extending the current simple API.

V. Examples of application

SlapOS is multi-purpose. It is used for public cloud and private cloud. It is used for IaaS, PaaS and SaaS. Although SlapOS is universal, it is also extremely simple. SlapOS API consists mostly of a single method: request.

SlapOS is used by VIFIB, a French Internet Service Provider, to provide virtual machines through IPv6. VIFIB infrastructures consists of Asus barebone PCs with Intel i7 860 CPUs, 80 GB Solid Stade Disks and 8 GB RAM. Thanks to the use of SSD, running 8 virtual machines on a single PC which costs only 700 EUR is possible. VIFIB virtual machines are used mostly by developers who find them faster than their own laptop. SlapOS is used by Beteireflow, one of Ireland's largest CRM operation. SlapOS is used to instanciate a complex ERP5 solution which includes non transactional NoSQL (kumofs), distributed transactional NoSQL (NEO), load balancing (ha-proxy), a cluster of application servers (Zope), a set of front-ends (Apache, Varnish) and automatic backup daemons (repozo). Thanks to SlapOS, this complex system can be reinstalled and configured in a matter of seconds on a new infrastructure. SlapOS plays a key role here as part of a Disaster Recovery Plan.

SlapOS is used by Nexedi to run ERP5 test suite. ERP5 test suite consists of more than 5000 tests including unit tests, functional tests and scalability tests. By distributing test tasks to SlapOS, ERP5 test suite can be executed in a few minutes instead of a few hours. This productivity improvement plays nowadays a key role in the acceleration of ERP5 release process.

SlapOS is also a key component in the COMPATIBLE ONE project, French government sponsored project for open source Cloud. SlapOS acts as a unified billing platform for all types of IaaS, SaaS and PaaS thanks to the TioXML resource accounting standard.

SlapOS should also soon serve as a platform for open source software publisher to turn their software into multi-tenant

[5]http://lwn.net/Articles/239238/

SaaS. SlapOS has received a lot of attention from IT industry and telecommunication industry because it is capable of hosting as many as 200 ERP instances at a cost of less than 1 USD per month and per instance. It is also acknowledged that SlapOS can save huge investments in research and development for those companies which need to turn their existing software into SaaS.

VI. Research Questions and Design

A. Technology driven projects

SlapOS raises wide variety of research topics: process allocation optimization, energy management, software distribution, resilient storage, high availability, security.

In SlapOS, infinite duration processes are allocated, rather than limited duration computing tasks in traditional grid computing [10], [11]. Once a process is allocated on a SlapOS server node, it can not be moved to another SlapOS server node. Process allocation should at the same time optimize hardware resource usage and guarantee acceptable service level. Allocating too few processes per server node is a waste of resource usage. Allocating too many processes per server may lead to poor service for customers and users. Early research on this topics is being conducted as part of TioSafe project [22] by Institut Télécom, based on the metaphor of bin packing.

Energy is becoming a key cost factors of Cloud Computing and a growing source of environmental impact. With SlapOS, heat dissipation during winter time is reused for household or office heating. During summer time, heat dissipation is either evacuated through natural air circulation or requires to spend additional energy for air cooling. In northern countries or mountain side, SlapOS leads to energy savings of more than 50% compared to a server farm. In southern countries, the energy balance is unclear. Improving the energy efficiency of SlapOS, either in combination with smart grids[6] or by implement optimal process allocation is another field of research.

SlapOS does not use traditional software packaging found for example in GNU/Linux distributions and which has been extensively modeled and studied, in particular by Roberto di Cosmo [8], [9]. Instead it combines source code access with heterogeneous packaging which are now specific to each programming language: python eggs [23], ruby gems [24], Java. Thanks to buildout technology, a single SlapOS server node can run any number of versions of a given software or library at the same time, thus providing the level of flexibility required for SaaS and PaaS. This new approach opens new fields of research in post-packaging software distribution.

SlapOS Nodes are not meant to be perfectly reliable. Instead, it is the role of the application to implement its own redundancy policy. This includes in particular redundant storage. Applications based on NoSQL database can leverage the native redundancy of most NoSQL software. Applications based on relational databases need to find relevant tactics to implement data redundancy. Some tactics are quite simple: for example, by connecting to two databases and maintaing an asynchronous copy of data from one database to the other. Other tactics are more complex and can involved clustered databases. Analysing the different tactics for data redundancy is an interesting field of education, if not research. Many students are not yet aware how to implement distributed data storage. SlapOS is a perfect test bed to educate students to the different approachs which are implemented in the industry.

High availability is another research topic with SlapOS. The introduction of mobile IPv6 could be consider to implement a kind of process migration, as it already exists in XtreemOS [25]. Other approaches, based on dynamic DNS and monitoring could be integrated at the core of SlapOS. Mapping processes to internet names is generally a interesting direction for the future of SlapOS in the context of high availability requirements.

Security is probably the most important research with SlapOS. Because it is based on IPv6 and on a flat view of Internet, SlapOS can not rely on firewalls or masquerading for security. Moroever, granting rights to install software on a SlapOS Node requires more flexibility than what SlapOS currently provides. The introduction of automated source code analysis and of distributed intrusion detection could help adding additional flexibility and create more trust for people to accept on their hardware software provided by other people. Operating system level security containers such as those found in ChromeOS [26] could also help reducing the risk breaking system security.

B. Distributed architecture and distribution of services

The coordination of SlapOS servers is a research issue as the coordination is the key point with Grids. In our context of Desktop Grids, the BonjourGrid [3]–[5] component aims at improving the infrastructure efficiency by breaking up a large, monolithic application into separate components (services). The rationale for doing this is that the large application no longer requires an equally large, monolithic mainframe computer to run on. BonjourGrid has been imaginated for answering to the following question: how to federate all the users and machines of all the Boinc, Condor and XtremWeb projects? (Boinc, Condor and XtremWeb are popular Desktop Grid middleware [11]).

In the past, Desktop Grid systems represented an alternative to super-computers and parallel machines and they offered computing power at low cost. Indeed, Desktop Grids have intrinsic features that explain the large number of international projects aiming to better exploit this computational potential. Many Desktop Grid system have been developed using a centralized model. These infrastructures run in a dynamic environment and the number of resources may

[6]See http://http://en.wikipedia.org/wiki/Smart_grid

increases dynamically. Hence, the need for decentralization is becoming increasingly important. BonjourGrid is a new decentralized approach of Desktop Grid systems.

Its main objective is to provide a decentralized infrastructure of multi-coordinators (here multi SlapOS servers), using the services offered by a publish/subscribe system [7]. Unlike a classical Desktop Grid, BonjourGrid can create, on demand, a dynamic and decentralized execution environment for each user, based on existing computing systems such as XtremWeb, Boinc or Condor, to run any kind of applications, without the intervention of a system administrator. Since SlapOS Masters are implemented themselves as SlapOS Nodes, in a completely reflexive ways, BonjourGrid may help to coordinate the multiple MAster nodes in the same way it coordinates the XtremWeb, Boinc or Condor coordinators. The problem is even simpler because we have only one type of system to manage (a SplaOS master).

Moreover BonjourGrid comes with a fault tolerant approach based on passive replication and virtualization to tolerate the crash of coordinators (SlapOS master nodes). The novelty resided here in an integrated environment based on Bonjour (publication- subscription mecanism) for both the coordination protocol and for the fault tolerance issues. In particular, it is not so frequent to our knowledge to describe and to implement a fault tolerant protocol according to the pub-sub paradigm. Experiments, conducted on the Grid5000 testbed[7], have illustrated a comparative study between Boinc (resp. Condor) on top of BonjourGrid and a centralized system using Boinc (resp. Condor) and second prove the robustness of the fault tolerant mechanism.

To summarize, the key idea of BonjourGrid is to rely on existing Institutional Desktop Grid middleware, and to orchestrate and coordinate multiple instances, i.e multiple computing elements, through a publish/subscribe system. We propose to reuse this framework for the coordination of SlapOS masters that may appear or disappear on the fly. Several issues must be taken into account in our future works. The first issue, which is not really difficult, is the reservation of participations: in the current version, BonjourGrid allocates available resources for a user without any reservation rules. Thus, if a user demands all the available machines for a long time, BonjourGrid allocates them to him. The second issue is going up to a wide area network. The current version works only in a local network infrastructure because of Bonjour, it is important to bypass this constraint. Grafting the new package of Bonjour, Wide Area Bonjour from Apple, may be a good solution to resolve this problem...but it is not sufficient to capture all the issues. For instance, the advent of Internet-Scale middleware raises qualitatively different issues than "global scale" alone specifically those of bridging trust domains across organizational boundaries. So, we will also need to address mecanisms for authentification. In this new environment, we are planning to apply a re-engineering step for using the Extensible Messaging and Presence Protocol (XMPP [17], [18]) which is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, pub-sub.... We believe that the XMPP framework is of premier choice for the implementation of the coordination in SlapOS but we need to measure the architectural impact of this choice on the protocol itself. A good example of what is possible to do with XMPP is the Archipel[8] project whose aiming at coordinating multiple virtual machines. However, XMPP has some limitations, among them a scalability problem in the management of rosters (see [18] page 219). The problem could be a problem with implementation.

Another possibility for the coordination is a solution based on websockets / COMET / hookbox[9]. Hookbox offers a pub-sub interface but it is more oriented towards 'pure HTTP' protocol than XMPP. Consequently, programming is more easy and we have less potential problems with firewalls.

Anyway, the BonjourGrid protocol which is:

1) The user requests for computation; He selects machines based on their performance (CPU, RAM,...);
2) The user provides the control ow graph, binaries, input data;
3) The user deploys locally a coordinator and requests for participants.

becomes as follows in SlapOS:

1) The user requests an instance of software release (ie. a daemon);
2) The user provides a URL to a description of the sofware release which itseld provides a description of how to create software instances;
3) The user provides configuration parameters to crate an instance of software release.

but the main challenge remains: how to coordinate SLapOS masters between multiple distinct companies and based on contractual relationships but not based on technical relationships (CPU, RAM,...)?

VII. Conclusion and Outlook

SlapOS demonstrates that the borders between IaaS, PaaS and SaaS in Cloud Computing are no longer relevant. Through a single API and a single method inspired by 10 years of experience in grid computing, SlapOS is capable of allocating virtual machines, application servers, databases and even ERP applications.

SlapOS also demonstrates that server farms and data centers are no longer required to provide high quality Cloud Computing. SlapOS servers hosted at home on optical fiber and IPv6 networks are capable of providing reliable

[7]https://www.grid5000.fr

[8]http://http://archipelproject.org/

[9]http://hookbox.org/

Cloud service thanks to application level data redundancy. Moreover, SlapOS contributes to energy savings by reusing heat dissipation of servers for households heating and by removing the need for air cooling in many countries.

SlapOS is already in commercial production. It is used by companies or by people who are looking for a Cloud Computing solution which protects their strategic data at a low operating cost. It is used in particular by software publishers who are urged to transform their applications into SaaS.

Yet, SlapOS needs many improvements. The Resilience project, a project sponsored by more than 10 organisations including Morpho, Nexedi, Nuxeo, Wallix, Université de Paris 13, Institut Télécom will improve SlapOS into two directions: by removing and making SlapOS Master distributed and by adding to SlapOS extensive support for security.

SlapOS also nees more contributors. We are calling here all research, educational organization to join the SlapOS community and start building recipes for open source software so that it can be deployed on SlapOS. This effort is important for scientific reasons: by creating a rich library of open source applications hosted on the Cloud using open source SlapOS, research and education can keep access to the know how of Cloud Computing which is for now migrating increasingly to companies such as Google, Facebook and Microsoft, and remains secret. SlapOS is our proposed testbed to make sure that Distributed Cloud Computing knowledge remains shared and open.

REFERENCES

[1] Lesyng, B., Bala, P., Erwin, D.: Eurogrid: European computational grid testbed. J. Parallel Distrib. Comput. **63** (2003) 590–596

[2] Liang-Jie Zhang EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing IEEE Transactions on Services Computing Vol 1, number 2, April-June 2008

[3] Heithem Abbes, Christophe Cérin, and Mohamed Jemni. Bonjourgrid as a decentralised job scheduler. In APSCC 08. Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, pages 89–94,Washington, DC, USA, 2008. IEEE Computer Society.

[4] Heithem Abbes, Christophe Cérin, and Mohamed Jemni. Bonjourgrid : Orchestration of multi-instances of grid middlewares on institutional desktop grids. In 3rd Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2009), en conjonction avec IPDPS 2009, Rome, Italie, 29 Mai 2009.

[5] Heithem Abbes, Christophe Cérin, Mohamed Jemni: A decentralized and fault-tolerant Desktop Grid system for distributed applications. Concurrency and Computation: Practice and Experience 22(3): 261-277 (2010)

[6] Douglas Thain, Todd Tannenbaum, and Miron Livny. Condor and the grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, Grid Computing : Making the Global Infrastructure a Reality. John Wiley et Sons Inc., December 2002.

[7] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. ACM Comput. Surv., 35(2) :114131, 2003.

[8] Roberto Di Cosmo, Stefano Zacchiroli: Feature Diagrams as Package Dependencies. Software Product Lines: Going Beyond 14th International Conference, SPLC 2010, Jeju Island, South Korea, September 13-17, 2010. Proceedings: 476-480

[9] Pietro Abate, Roberto Di Cosmo, Jaap Boender, Stefano Zacchiroli: Strong dependencies between software components. Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement, ESEM 2009, October 15-16, 2009, Lake Buena Vista, Florida, USA: 89-99

[10] Ian Foster (Editor), Carl Kesselman (Editor) The Grid: Blueprint for a New Computing Infrastructure (The Elsevier Series in Grid Computing) Morgan Kaufmann, 1999

[11] Christophe Cerin (Editor), Gilles Fedak (Editor) Desktop Grid Computing Chapman & Hall, Fall 2011

[12] Smets-Solanes, J.-P., Atem de Carvalho, R. ERP5: a next-generation, open-source ERP architecture IT Professional, July-Aug. 2003, Volume: 5 Issue:4, page(s): 38 - 44

[13] Teragrid. URL: https://www.teragrid.org/

[14] Grid5000. URL: https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home

[15] Boinc. URL: http://boinc.berkeley.edu/

[16] XtremWeb. URL : http ://www.xtremweb.org.

[17] XMPP. URL: http://xmpp.org/

[18] Xmpp, the definitive guide. Peter St Andre, Kevin Smith and Remko Tronçon O'Reilly, 2009.

[19] Memcached: a free and open source, high-performance, distributed memory object caching system. http://memcached.org/

[20] Buildout - software build system reloaded http://www.buildout.org/

[21] Supervisor: A Process Control System http://supervisord.org/

[22] Tiosafe http://www.systematic-paris-region.org/en/projets/tiosafe

[23] Python Eggs http://www.python-eggs.org/

[24] Ruby Gems http://rubygems.org/

[25] Xtreemos http://www.xtreemos.eu/

[26] ChromeOS – http://www.chromium.org/