ERP and E-Business Application Deployment in Open Source Distributed Cloud Systems

George SUCIU[1], Traian-Lucian MILITARU[2,3], Gyorgy TODORAN[3]
[1, 2] Faculty of Electronics, Telecommunications and Information Technology, POLITEHNICA University of Bucharest
[3] Quality Department, POLITEHNICA University of Bucharest
george@beia.ro, gelmosro@yahoo.com, todoran.gyorgy@gmail.com

In this paper we present the way in which we combine SlapOS, the fist open source operating system for distributed cloud computing, and Enterprise Resource Modeling (ERP) to provide an simple, unified API for E-Business Applications based on Iaas, PaaS and SaaS models. 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 using a master-slave model. SlapOS Master follows an ERP model to handle at the same time process allocation optimization and billing.
Keywords: *Cloud, ERP, IaaS, PaaS, SaaS, Open Source*

1 Introduction

We will introduce in this article SlapOS, the first open source operating system for Distributed Cloud Computing and ERP5, the open Source e-Business suite. SLAP stands for “Simple Language for Accounting and Provisioning” and uses Slapgrid daemon that receives requests from a central scheduler the SlapOS Master which collects back accounting information from each process.

In this paper we present the way in which we combine open source grid computing for distributed cloud 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 for E-Business Applications.

Researches in cloud computing and standardizing efforts in the IT industry are presenting the layered cloud model as shown in Figure 1.

IaaS offers the infrastructure of virtual machines and storage. PaaS is built on top of IaaS and offers platform services such as operating systems, application servers and data bases. SaaS is the layer that offers services to the final user in the form of various applications: Customer Relationship Management (CRM), ERP systems, communication systems, blogs, multimedia application systems, etc.

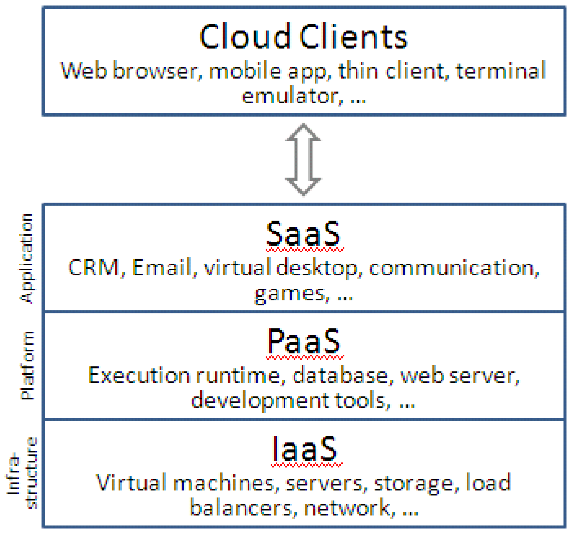

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
Application
SaaS
CRM, Email, virtual desktop, communication, games, ...
Platform
PaaS
Execution runtime, database, web server, development tools, ...
Infra-structure
IaaS
Virtual machines, servers, storage, load balancers, network, ...

Figure 1. The General Cloud Model

ERP systems are said to enable organizations to manage their resources efficiently and effectively by providing a total and integrated solution for their information processing needs. Due to technical and economical restrictions, ERP systems traditionally have been focused on larger organizations. In recent years however, a turn of the market towards Small and Medium Enterprises (SMEs) can be

observed [1]. This shows that SMEs are as likely to be interested in ERP as multinational organizations. ERP packages are being viewed as a key factor for gaining competitive advantage in the SME sector and empirical findings confirm these expectations [2].
Business models, where SMEs access ERP functionalities through the Internet instead of purchasing them could alleviate the problems of lack of human and financial resources and broaden the ERP market. Recently Software as a Service (SaaS) is associated to this kind of business model [3]. By providing applications directly through the Internet, SaaS eliminates installation and update tasks, thus saving clients from maintenance work and reducing IT expenses by on-demand pricing [4].
Another "disruptive business model" mentioned by Hofmann [3] is that of open source companies. Free / open source ERP systems might be an alternative for SMEs as they tackle their specific problems. They not only help to save license costs, but they also prevent lock-in. As their source code is free to everyone they lower the barrier for third parties to perform modifications [5].

2 Characteristics of SlapOS Distributed Cloud Computing Platform

SlapOS is an open source Cloud Operating system which was inspired by recent research in Grid Computing and in particular by BonjourGrid a meta Desktop Grid middleware for the coordination of multiple instances of Desktop Grid middleware. It is based on the motto that "everything is a process". SlapOS is now an OW2 project. Figure 2 shows the current cloud architecture model [6].

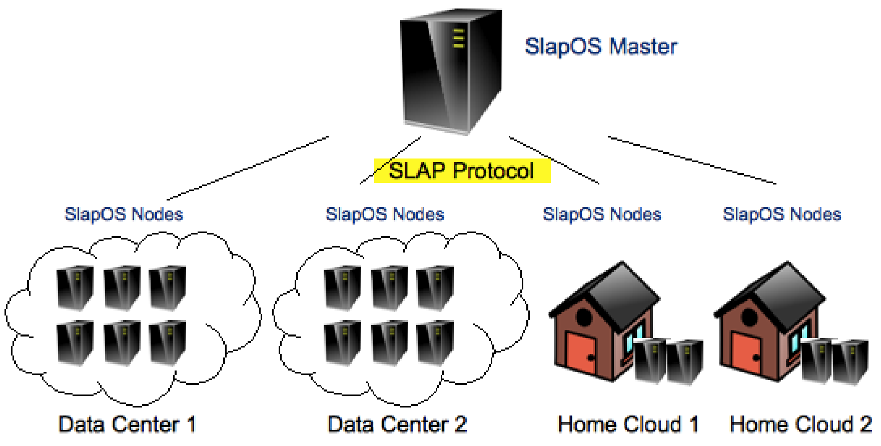


Figure 2. The SlapOS Cloud Architecture Model

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 by using the SLAP protocol.

2.1. SlapOS Architecture

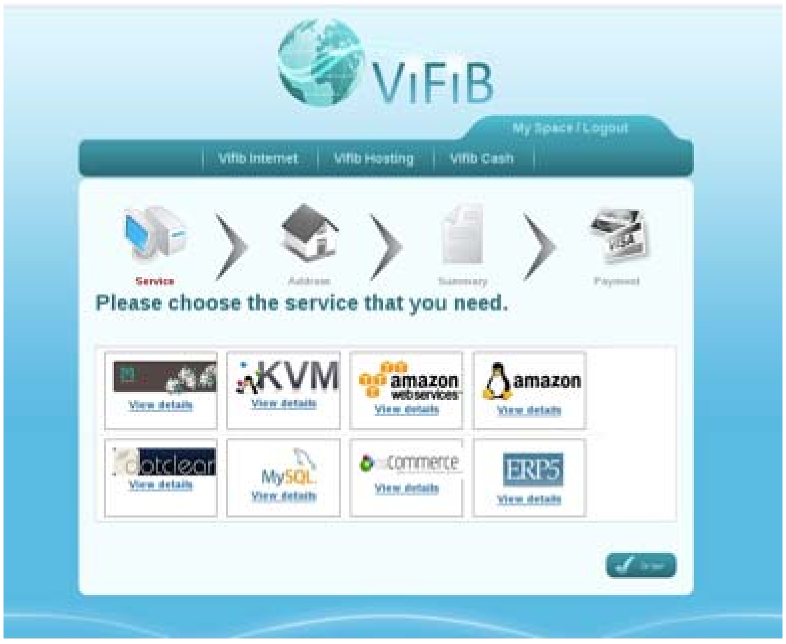
SlapOS Nodes and SlapOS Master exchange are interconnected through the HTTP and XML based SLAP protocol. SlapOS Master sends to each SlapOS Node a description of which software should be installed and executed. Each SlapOS Node sends to SlapOS Master a description of how much resources were used during a given period of time for accounting and billing purpose [7].
From a user point of view, SlapOS Node looks like an online shop for Cloud Computing resources. The user connects to

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.

2.2. An example of SlapOS front-end

From a developer point of view, as shown in Figure 3, SlapOS is a simple and universal API to create instances of any software daemon through a programmatic interface. A simple code allows a developer to request a new instance of a memcache server by invoking the request method of SlapOS API.

Memcache 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 administrator would have done in few minutes at best.



Please choose the service that you need.

Figure 3. An example of SlapOS front-end

2.3. SlapOS Kernel

SlapOS is implemented as an extension of widely adopted open source software: GNU/Linux, Buildout and Supervisord [6]. The only new software introduced by SlapOS is Slapgrid, a daemon in charge of implementing the SLAP protocol on each SlapOS Node with the functionality as depicted on Figure 4.

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 bootstrap 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.

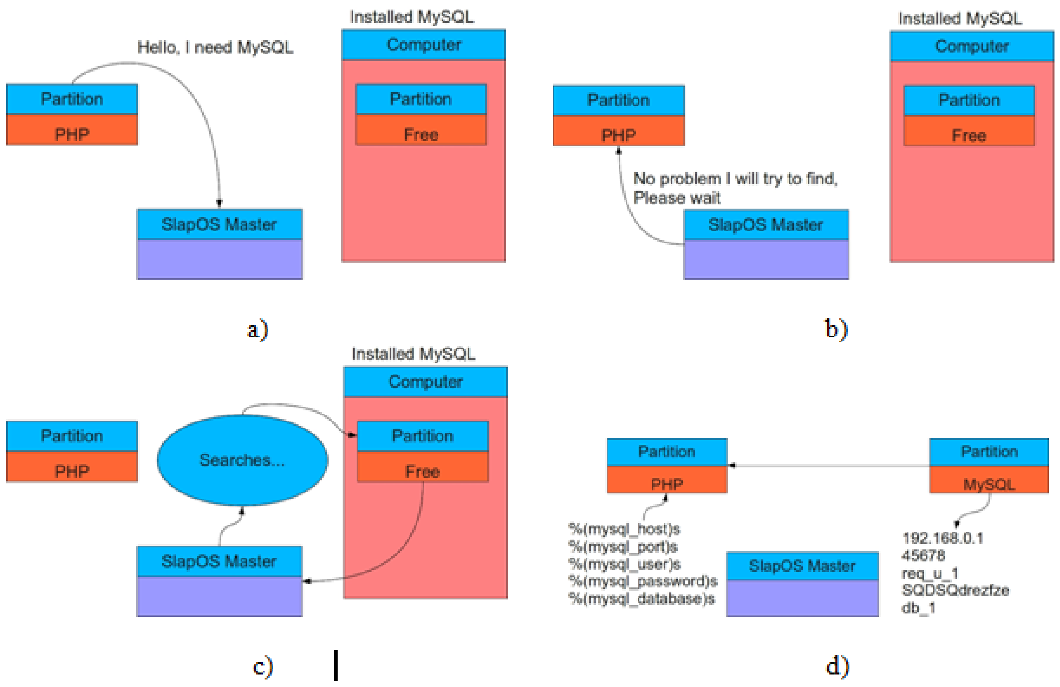


Figure 4. Example of allocating a partition in SlapOS

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 reproducible way.

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, as shown in 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.

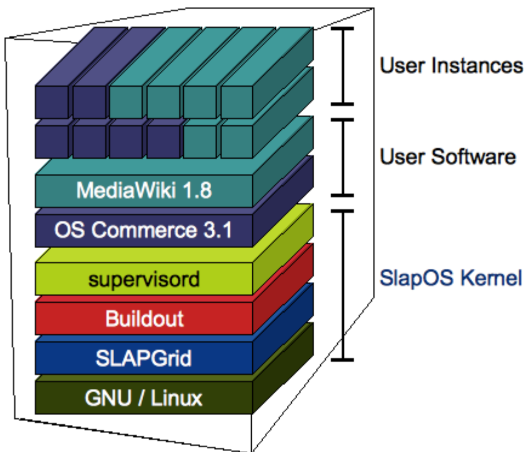


Figure 5. SlapOS kernel and user software implementation

Every computer partition consists of a dedicated IPv6 address, a dedicated local

IPv4 address, a dedicated tap interface (slaptapN), a dedicated user (slapuserN) and a dedicated directory (/srv/slapgrid/slappartN). Optionally, a dedicated block device and routable IPv4 address can be defined.

SlapOS is usually configured to use IPv6 addresses. Although use of IPv6 is not a requirement (an IPv4 only SlapOS deployment is possible) it is a strong recommendation. IPv6 simplifies greatly the deployment of SlapOS either for public Cloud applications or for private Cloud applications. In the case of public Clouds, use of IPv6 helps interconnecting SlapOS Slave Nodes hosted at home without having to setup tunnels or complex port redirections. In the case of private Cloud, IPv6 replaces existing corporate tunnels with a more resilient protocol which provides also a wider and flat corporate addressing space. IPv6 addressing helps allocating hundreds of IPv6 addresses on a single server. Each running process can thus be attached to a different IPv6 address, without having to change its default port settings. Accounting network traffic per computer partition is simplified. All this would of course be possible with IPv4 or through VPNs but it would be much more difficult or less resilient. The exhaustion of IPv4 addresses prevents in practice allocation some many public IPv4 addresses to a single computer. After one year of experimentation with IPv6 in Romania, using IPv6 native Internet access (more than 50% of worldwide IPv6 traffic), we found that IPv6 is simple to use and creates the condition for many innovations which would else be impossible.

3 Open Source ERP and E-Business Application Deployment Prototype

Thanks to its abstract model, ERP5 is a very generic application system, thus many ERP5 tailoring tasks can be accomplished solely by configuration and still have great effect on how ERP5's functionalities behave. The deployment process that we followed contains analysis, implementation and test phases. Analysis is based on interviews and document research. Its purpose is to discover resource flows and decision flows in a company. It also aims to identify the demand for implementation of custom document types. The procedure is aligned to ERP5's document-centric approach to implement business processes.

SlapOS Master runs ERP5 Cloud Engine, a version of ERP5 open source ERP capable of allocating processes in relation with accounting and billing rules, as shown in Figure 6.

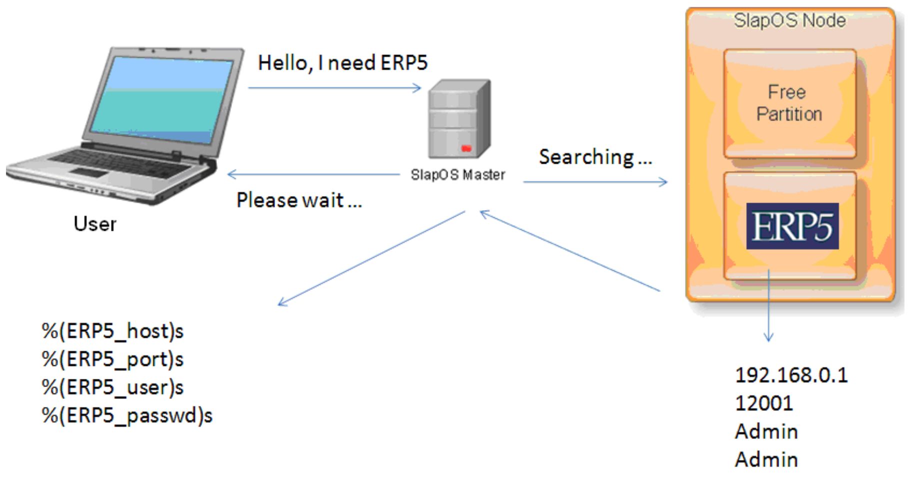


Figure 6. Example of allocating a ERP instance and sending access data to user

Initial versions of SlapOS Master were installed and configured by human. The new versions of SlapOS Masters that we use in our prototype 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.
The implementation process is supported by ERP5 through a series of tools for requirements, analysis, design and implementation as well as through general process related tools [8].
ERP5 has used from the design phase a single programming language for all the parts of the application: business logic, database interface, scripts and additional modules. The ERP5 framework is implemented in Python, a general-purpose, interpreted high-level programming language. By using a single programming language for scripts as well as the base application we have the possibility of reusing the code originally written for scripts for integration in the code of the main application.
The ERP5 application is built on top of open source technologies and takes up many of the characteristics of such solutions. Therefore a good knowledge of the architecture is necessary to understand the ERP5 framework, as presented in Figure 7

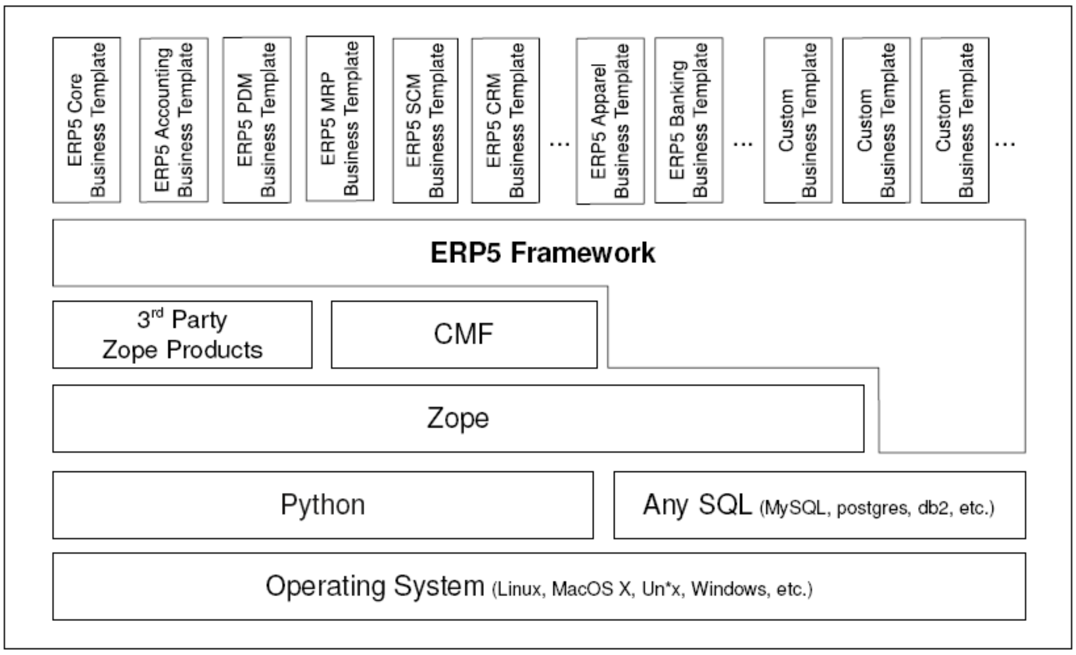

ERP5 Core Business Template
ERP5 Accounting Business Template
ERP5 PDM Business Template
ERP5 MRP Business Template
ERP5 SCM Business Template
ERP5 CRM Business Template
...
ERP5 Apparel Business Template
ERP5 Banking Business Template
...
Custom Business Template
Custom Business Template
Custom Business Template
...
ERP5 Framework
3rd Party Zope Products
CMF
Zope
Python
Any SQL (MySQL, postgres, db2, etc.)
Operating System (Linux, MacOS X, Un*x, Windows, etc.)

Figure 7. ERP5 technology implementation

The ERP5 framework is built on the Zope Application Server, an open source application that can run on different platforms, including Linux, Windows and MacOS [9]. Zope is similar to an Apache server but highly oriented on objects.
ERP5 uses Zope to:

- Publish objects (images, html files, java scripts, etc) on the Web
- Authentication and authorization of users
- Easy integration of new modules in ERP5

For data storage Zope includes Zope Object Data Base (ZODB), an object oriented database. Objects can be imported / exported in ZODB in XML format. ERP5 allows the synchronization of objects in ZODB with similar objects from other ERP5 instances by using the SyncML protocol [10].
Zope provides also the management of the ERP5 application components through the

Zope Content Management Framework (CMF). The *CMF* provides:

- storage and access to the ERP5 components
- script add
- workflow change of components
- archiving of existing components

This modularity gives ERP5 the technological possibility to be easily implemented in a SlapOS distributed cloud system by using the buildout technology.

We will use our deployment platform hosted on several servers running Ubuntu Linux – Apache – MySQL template with current software release.

Buildout will use the information from the configuration file and will install the ERP5 application only once on each available node. This first and only installation of the application is called *Software Release* or *Application Kernel* as shown in Figure 8.

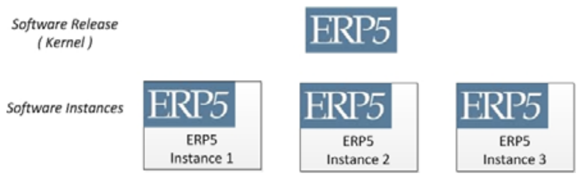


Figure 8. Kernel and Instances of ERP5 deployment

The user can request an instance of ERP5 from the SlapOS Master, www.wifib.net [11] in our prototype, after the *Kernel* of the application has been successfully installed on a SlapOS node.

It is noticed that the combination of Web and an ERP is E-Business, which is opening through the Web the corporate ERP to all partners of a company. Since business are increasingly distributed and relying on outsourcing and outside partners, it is a natural tendency for ERPs to become e-business tools. ERP5 does it for example with a few others as shown in Figure 9.

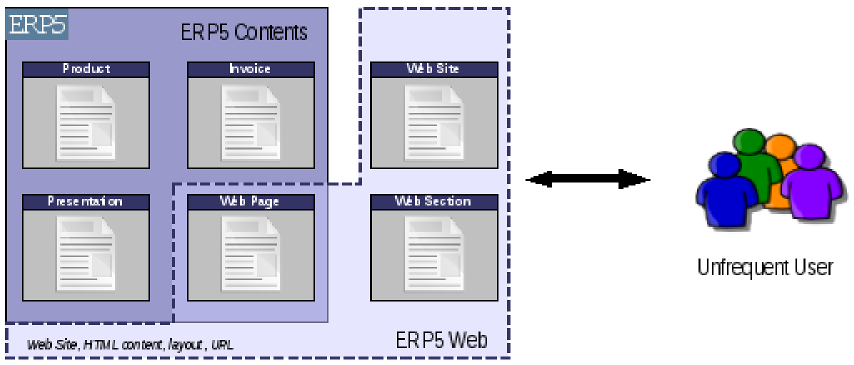
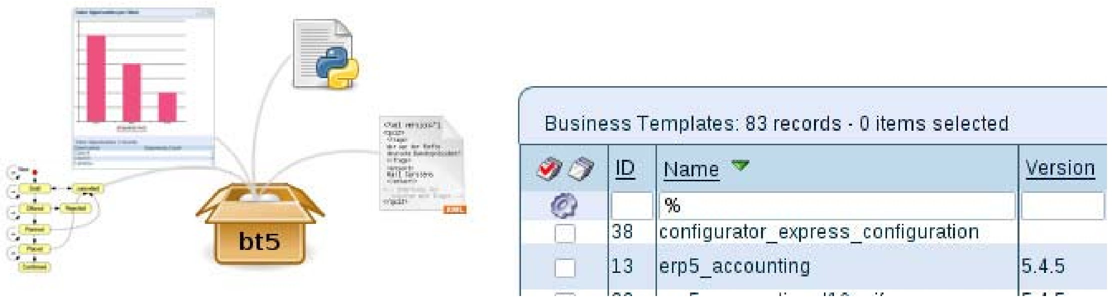


Figure 9. E-business model based on ERP5 Web site

After installing ERP5 needs to be configured by defining, selecting, interconnecting and populating modules of the program that provide a model of processes, documents, actions or persons of a company.

Each instance of ERP5 is a web site that can contain one or more modules such as standard ERP modules: Accounting, Products, Customers, etc. but also modules that are not usual for ERP such as Blog, News, Forum.

The instances of ERP5 use the ERP5 business templates (bt5) as shown in Figure 10. These software packages contain code lines written in Python that add new functionalities to the existing platform.



Business Templates: 83 records - 0 items selected

	ID	Name	Version
		%	
	38	configurator_express_configuration	
	13	erp5_accounting	5.4.5

Figure 10. ERP5 Business Templates

Based on the implemented scenario, several question rise regarding its performances and efficiency. SlapOS nodes report on resources used and trusting client to report billing values is a well-known security issue. The security mechanisms included in SlapOS are setup to prevent a node from cheating on billing values reported. However traffic on unencrypted links could be intercepted and it is possible for a node to join the cloud and start sniffing sensitive data.

Also some time is needed before a new application gets installed on a node and becomes ready to use. Questions arise about slow response and how does the master node handle this delay? The specific provisioning strategy could be improved to always keep a buffer of ready-to-use applications.

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).

SlapOS should also soon serve as a platform for open source software publisher to turn their software into multi-tenant SaaS. On the other hand know how is attracted continuously by companies such as Google, Facebook and Microsoft where it remains secret.

Nevertheless development is still needed for Slaprunner, the SlapOS buildout web based runner so that software profiles and release versions can be better managed and many other web-based applications added to the software release directory.

5 Conclusions

The research objective is to investigate the deployment of ERP and E-Business package on open source distributed cloud systems. This has been done on the basis of SlapOS that can leverage the resources of multiple, independent providers of computing power in order to create a resilient cloud and a single node can handle as much as 200 independent ERP databases running at the same time as tested on our deployment prototype.

E-Business is in 2011 one of the very few fast growing businesses in IT and in industry. More than 1000 different solutions of E-Business are available, with fantastic open source packages available.

However, E-Business integration is still difficult: people get orders from one place (ex. Ebay), follow clients on another place (ex. SalesForce CRM) and keep their accounting with Microsoft which can cost as much as 2000 Euro/seat and involves high risk of failure.

Integration between the different aspects of business, on the Web or offline, is made by open source SlapOS and ERP5 to create a simple prototype in short time at low cost, because there are no license costs and is therefore probably the safest way to adopt an ERP or E-business.

6 Acknowledgment

This paper is presented as part of the project “Valorificarea capitalului uman din cercetare prin burse doctorale (ValueDoc)” Project co-financed from the European

Social Fund through POSDRU, financing contract POSDRU/107/1.5/S/76909 and part of the project "Cloud Consulting".

References

[1] A. Deep, P.Guttridge, S. Dani and N. Burns, "Investigating factors affecting ERP selection in made-to-order SME sector," *Journal of Manufacturing Technology Management,19*(4), pp. 430–446, 2008.

[2] S. C. L. Koh and M. Simpson, "Could enterprise resource planning create a competitive advantage for small businesses?," *Benchmarking: An International Journal, 14*(1), pp. 59–76, 2007.

[3] P. Hofmann, "ERP is dead, long live ERP," *Internet Computing, IEEE, 12*(4), pp. 84–88, July–August, 2008.

[4] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer and W. Karl, "Scientific cloud computing: early definition and experience," *10th IEEE international conference on High Performance Computing and Communications.* New York, N.Y: Institute of Electrical and Electronics Engineers, pp. 825–830, 2008.

[5] R. Campos, R. Carvalho and J. S. Rodrigues, "Enterprise modeling for development processes of open-source ERP," *18th Production and Operation Management Society Conference,* Dallas, TX, pp. 3-8, May 2007.

[6] J.-P. Smets, "ERP5 products documentation," *Nexedi SA*. Retrieved December 10, 2011, from https://www.myerp5.com/kb/documentation section/ (Restricted), pp. 1-9, December 2011.

[7] G. Suciu, O. Fratu, S. Halunga, C. G. Cernat, V. A. Poenaru and V. Suciu, "Cloud Consulting: ERP and Communication Application Integration in Open Source Cloud Systems," *19th Telecommunications Forum - TELFOR 2011, IEEE Communications Society*, pp. 578-581, Belgrade, Serbia, November 2011.

[8] R. Carvalho and R. M. Monnerat, "Development support tools for enterprise resource planning," *IT Professional*, *10*(5), pp. 39–45, Sept.–Oct. 2008.

[9] Zope site „ Zope Resources" [Online] Available at: http://www.zope.org, July 2012.

[10] A.Traud, J. Nagler-Ihlein, F. Kargl and M. Weber, "Cyclic Data Synchronization through Reusing SyncML," *The Ninth International Conference on Mobile Data Management MDM 2008*, pp.165-172.

[11] J.-P.Smets-Solanes, C. Cérin and R. Courteaud, "SlapOS: A Multi-Purpose Distributed Cloud Operating System Based on an ERP Billing Model," *2011 IEEE International Conference on Services Computing* 2011 765-766.

George SUCIU graduated from the Faculty of Electronics, Telecommunications and Information Technology at the "Politehnica" University of Bucharest in 2004. He holds a Master diploma in Project Management from the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies Bucharest from 2010 and currently, he is a Ph.D. Student in the field of Electronics Engineering and Telecommunication. His scientific fields of interest include: project management, electronics and telecommunication, cloud computing, data acquisition and signal processing. Known languages: German, French, English; Experience and participation in research projects (FP7, Eureka, Eurostars, National Structural Funds), more than 10 years activity in information and telecommunication systems.

Traian-Lucian MILITARU graduated from the Faculty of Electronics, Telecommunications and Information Technology at the "Politehnica" University of Bucharest in 2004. He holds a Master diploma in Quality in Electrical Engineering from the "Politehnica" University of Bucharest from 2006 and currently he is a Ph.D. Student in the field of Quality Assurance. His research activity can be observed in many international proceedings. His scientific fields of interest include: quality assurance, telecommunication, programming and cloud computing. Currently, he is software engineer for more than 8 years, with vast experience in both programming and quality assurance.

Gyorgy Todoran has graduated the Faculty of Electronics, Telecommunications at "Politehnica" University in Bucharest in 2000. He holds a Master degre in Quality Management (2001) and Strategic Management (2002) from the "Politehnica" University of Bucharest. Currently he is working on his Ph.D. thesis in security tehnologies with focus on open source, cloud computing, mobile and BYOD initiatives. He has more than 10 years experience in commercial and governmental telecommunication systems, mainly in system administration, system management, design, project management, consulting.