

Matthias Rechenburg (m.rechenburg@t-online.de)

Copyright (C) 2009 Matthias Rechenburg

This document is released under the GNU/GPL

Instant Cloud Computing with openQRM

What is Cloud Computing ?

Approximately since beginning beginning of 2008 the term “Cloud Computing” came up as new hype in the IT-World directly after “Virtualization”. Since then this term is used for different new and also well-known technologies for managing large data-centers and IT infrastructures. Some use it as a synonym for “Software as a service” (SAAS) coming from the Web 2.0 evolution, some other connect the term “Cloud Computing” with Automated Provisioning of a large number of virtual machines and “Servers on demand”. In general the definition of the term “Cloud Computing” leaves it open but defines in more generic way :

Cloud computing is [Internet](#) ("cloud") based development and use of computer technology ("[computing](#)")[\[1\]\[2\]\[3\]](#). It is a business [information management](#) style of computing in which typically [real-time scalable](#)[\[4\]](#) resources are provided "[as a service](#)"[\[5\]](#) over the [Internet](#)[\[6\]](#) to users who need not have knowledge of, expertise in, or control over the technology infrastructure ("in the cloud") that supports them[\[7\]](#).

(Wikipedia → Cloud Computing)

Even more, both aspects of Cloud Computing, SAAS and Automated Provisioning, are co-existing in a kind of symbioses relationship. SAAS means to provide a Service via a network (Inter- or Intranet) to its customers on demand. Most common for this case is that the applications and services are deployed to a separated virtual machine. This isolation of the application in its own VM results in better flexibility and security for the SAAS provider. The requirement to deploy a huge number of VMs (one per application instance) needs automatism which is solved through Automated Provisioning.

Cloud Computing is not a new technology but a new name for a combination of two or more already known technologies like HPC- and HA Clustering, Grid-Computing, Utility-Computing, Distributed-Computing etc. They are all there to manage large IT infrastructures used for different purposes in an automated way.

The role of the open-source community especially for Cloud Computing

Modern Clouds mostly consists of already existing, well-known and often open-source components. For every aspect of a Cloud environment a different set of utilities is used e.g. Puppet for automated configuration management, Nagios for system- and service monitoring, one or more virtualization technologies, Linux-HA for high-availability etc. Those utilities are already used in large production and accepted by system administrators

and the IT-management. It would make no sense to re-write all those tools from the scratch just because of a new name (Cloud Computing) for known methods of deploying and managing large server environments. For that reasons Clouds are mainly a set of “loosely connected” tools.

The challenge for modern Cloud Computing is the integration of those “loosely connected tools” into a single management User Interface. It can only be achieved via a plug-ability which combines all the separated utilities to benefit from their cooperation. This is exactly the goal of the openQRM data-center management platform.

The openQRM data-center management framework

Modern data-centers are often a very complex environment with many different aspects require to perfectly work together to keep all services up and running. Integrating all those facets of the IT infrastructure is a huge task which most likely costs a lot of development and QA time and on the other hand would balloon a single, standalone application.

OpenQRM is following a different approach. Instead of implementing all required features into a main application server basically all mechanism are “out-sourced” to plugins. In general the openQRM-server itself provides only the back-end and framework to manage plugins. Via a well defined API openQRM provides lots of integration options and “hooks” for plugins to execute certain commands in specified situations in the data-center. This enhanced plug-ability also makes it very easy to integrate other already existing or new third party utilities.

The concept of openQRM is then to break the different aspects of a data-center into small, generic and manageable objects. Those objects are purely generic like a lego-brick and openQRM can puzzle and combine them in every way for the system administrator.

One of the first questions which came up about this approach of the separation into manageable objects is :

What is Linux ?

A linux server as we know it today consist of a kernel file, an initial ram disk file (initrd), some kernel module files and a root filesystem (/) which also consist of “only” files. So basically a linux server (the software-part of it) is just a bunch of files which are then loaded onto a physical server (or virtual machine).

The most common way to deploy a server today is still to install all those files on the servers hard disk. Most likely this deployment method is causing troubles because as experiences hard disks are the weakest part of our current hardware. At some point the hard disk, or even hard disk RAID, may break and stop working. Without a good backup solution this situation can be harmful to any company. OpenQRM solves this situation via a full separation between hardware (the physical systems) and software (the server providing services). With this separation physical hardware becomes replace-able at any time.

openQRM is using an image based deployment mechanism to rapidly inject server-images to physical (or virtual) servers via netbooting. In the openQRM environment the servers are located and stored on dedicated storage-devices (NAS/SAN/FC/iscsi/Aoe/*) as “server-images”. Those server-images are basically “just” the root file systems for the deployment

(the software-part of the provided service in the data-center). Different from the automatic-installation deployment this method directly connects server root-file systems to starting servers after their PXE request.

System administrators benefit from this architecture in several ways. Deploying new servers gets very easy and fast because they simply can be started up immediately with “clones” (snapshots) of existing server-images. Those snapshotting-features of modern storage devices is also very useful to keep different version of the same server with the option to roll-back and roll-forward at any time.

OpenQRM also supports to transfer and install server-images during deployment time e.g. a server can be assigned to start an empty server-images located on an Iscsi-Lun which is being populated on boot-up with a server-template from an NFS-location. This fully plug-able mechanism can be also used to create server-images from already existing servers by “grabbing” the servers local disk content and transfer it to a Storage location. Also deployment to local hard disk is supported. In this case (since we experienced that local hard disk may break at some point) the server-image is still located safe on the storage-server and can be re-deployed to a new hardware within minutes.

With its integrated and plug-able storage-management openQRM automatically takes care to authorize server-images selected for deployment to the actual resource (the physical or virtual hardware). This security mechanism makes sure that each server (-hardware) is only able to access its dedicated root file system.

Plug-able virtualization types

Not only the storage types but also the virtualization types are fully plug-able in openQRM. Via its open plugin API openQRM integrates with VMware-ESX, VMware-Server 2, VMware-server (1), Xen, KVM, Citrix XenServer and Linux-VServer. Adding support for further virtualization technologies like Virtualbox, openVZ and others is on the future road map. To be able to seamlessly handle all those different kinds of virtual machines openQRM puts a layer on top of the virtualization methods to unify their management. In openQRM virtual machines are simply net-booting into the openQRM management environment in the same way as physical systems. Continuing with the full separation between hard- and software, meaning on one side physical- and also virtual machines (because the Vms are running on the Hypervisor which is running on the bare metal) and on the other side the software layer, the server-images located on a safe storage device, in an openQRM environment a Hypervisor becomes “just” a resource provider, just being responsible to host the virtual compute resource of the users choice. That way the appliance running on a virtual machine also gets fully independent from its Hypervisor Host and can be transparently (live-) migrated to another Hypervisors of the same or different virtualization technology or even from physical systems to virtual machines and the other way around. OpenQRM supports P2V, V2P, V2V, P2P migration without any changes on the actual server-images itself.

OpenQRM cannot only manage different types of Hypervisor technologies but it can also deploy them via the regular generic deployment mechanism of its framework. That offers scalability for the complete IT infrastructure because the data-center can grow (and shrink) as demanded by just adding (or removing) more Hypervisors.

Enhanced High-availability with N-to-1 fail-over

When talking about “Green IT” the current approach is to use Virtualization to consolidate “many” physical servers to run virtualized on “one” (or more) Hypervisor Hosts. While this method is good to save the overall power consumption it is often forgotten that in the new, virtualized situation in the case the “one” Hypervisor Hosts breaks those “many” servers running in virtual machines on this Hosts will also get unavailable. This means that in the modern, virtualized Worlds we need to especially take care of the high availability.

The usual method to keep system High available (10 custom servers) :

- get additional 10 servers preferred of the same manufacture consisting of the same parts
- configure syncing of the disks between the 10 pairs of server
- implement a Fail-over solution for the service running on the 10 clusters

As a result this method requires 20 physical systems to keep 10 servers high available.

HA in openQRM :

- deploy the 10 custom servers via openQRM
- add 1 server as Hot-Standby

In the case one of the 10 custom servers break openQRM will use the one available system to restart it via its rapid deployment methods. As the result 10 (or more) servers can be made high available with just a single Hot-Standby system (since resource types are not linked to the actual server-image in openQRM physical servers even could fail-over to virtual machines).

This saves the power consumption of 9 servers !

Plug-ability → Automatism → Cloud Computing

On top of this straight and generic openQRM framework sits the Cloud plugin as a “simple UI” for the end user. It provides a fully automated provisioning cycle from physical systems or virtual machines deployment through automated application and configuration management according to the user's requests up to automated de-provisioning to free up the user's compute resources. Via an additional external Cloud web-portal users registered to the Cloud can login to manage their own Cloud environment by requesting new resources, de-provisioning existing ones or managing their active Cloud requests.

For the system administrator the Cloud provides an internal interface plugged into the openQRM server. It provides a fine grained overview about the current Cloud activities and several configuration options to tune the Cloud behavior e.g. the Cloud can be set to automatically or manually approve new Cloud requests, automatically create new virtual machines if not enough existing ones are available, enable or disable the clone-on-deploy features etc. It also consists of a Cloud Ip-manager which is used to automatically configure the external network-interfaces of the provisioned machines.

The default behavior of the openQRM Cloud is to use the “clone-on-deploy” mechanism to

provision resources requested by the users. This means that for every Cloud request the integrated storage management will execute a clone command on the storage server hosting the “golden-image” (server-image template) and then use the snapshotted server-image to deploy it for the user. This method has the huge advantage that the snapshots on the storage server (e.g. via the LVM snapshot storage feature) normally do not consume any space because they are only read-only copies from the original logical volume (server-image). This means only the changes the users makes to its server are saved to the storage and no additional space for the actual root file system plus its applications.

For automated leveraging the application layer of the provisioned systems according to the users request openQRM integrated with the Puppet configuration management system. Puppet takes care to setup and pre-configure the users machines via pre-made and known-to-work recipes. The integration of Puppet as an additional plugin and the cooperation of the Puppet-plugin and the Cloud add-on provides a selection of “out-of-the-box” application servers and gives added value by automatism for the users and administrators.

The openQRM Cloud comes with an integrated billing system for the compute resources consumed by the end users. Based on “Cloud Computing Units” (CCU's), the virtual currency in the openQRM Cloud, system administrators and users can plan and keep track of their used compute power and costs. OpenQRM simply calculates the amount of CCU's per request and subtract it from the Users account. That way the Cloud manager is completely open how to sell the computing power to the market end users (e.g. via Ebay).

Private and public Cloud Computing / Security Aspects

Clouds can be divided into “Public Clouds” and “Private Clouds”. While within a private cooperated data-center environment security aspects may depend on the system administrator and on their actual usage. For a internally QA network, protected by the companies firewall, security may not be that critical compared to the load balancer cluster of the main database application. In any way for Public Cloud Computing an exhaustive security concept is needed to ensure the integrity of the users data and provide continuously reliability for the Cloud service.

OpenQRM's thoroughly designed security concept consist of automated authentication of the server-images, the separation of the hard- and software layer to ensure the data-integrity and enable service high availability via rapid re-deployment plus the separation between the actual Cloud management system and the end users Cloud portal.

To gain additional security on top of the Cloud-security virtual private networking (VPN) and file system encryption can (and should) be additionally used in the application layer.

Conclusion

With its unique architecture and generic concepts of rapid and fully automated deployment openQRM provides a complete and extensible data-center management framework for large IT-infrastructures. Its Cloud Computing User Interface additionally gives instantly and on-demand access to the compute resources for the end-users.