La virtualisation

Exposé technique, licence pro. A.S.R.A.L.L., IUT Charlemagne de NANCY

Nicolas FILLOT, Julien VAUBOURG

17 décembre 2009

Table des matières

Introduction

La solution de facilité pour une entreprise qui voudrait répartir ses différents services (serveur de mail, bases de données, logiciels de comptabilité, etc.) tout en respectant l'isolation des services serait d'allouer une machine physique pour chacun. C'est d'ailleurs la solution qui est a été mise en oeuvre pendant de nombreuses années. Depuis l'avènement de la virtualisation, sous toutes ses formes, ce n'est plus la meilleure solution à appliquer.

La virtualisation, c'est le fait de concentrer plusieurs systèmes (aussi bien distributions Linux, systèmes *BSD, Windows, Mac, ou même pourquoi pas des architectures plus exotiques comme les consoles de jeux) qui peuvent être hétérogènes et qui peuvent fonctionner simultanément, sur une même machine, tout en disposant de moyens d'automatisation dans la gestion de ceux-ci. Ainsi, selon le type de virtualisation choisi, un système principal s'interpose entre la machine physique et une collection de machines virtualisées. C'est ce système principal (qui peut être lui-même un système d'exploitation ou un logiciel spécifique) qui permet de gérer les autres de façon très aisées : création, duplication, suppression, supervision, etc, en quelques secondes seulement. Toujours selon le type de virtualisation retenu, le système interposé émule plus ou moins le matériel de la machine physique pour le simuler aux systèmes virtualisés. Cette technique permet d'établir des quotas précis de mémoire vive ou de processeur à chacune des machines, et d'accentuer la sécurité en isolant de façon plus ou moins hermétique les systèmes du vrai matériel.

Quelques arguments en faveur de cette technologie :

- **Coût :** Tout ajout de machine nécessite de prévoir des coûts de fonctionnement et de maintenance. La virtualisation concentre plusieurs serveurs en un seul.
- **Rentabilité :** Les serveurs sont sous-utilisés. D'après une étude de l'*International Data Corporation*, seuls 15% de la capacité totale est utilisée sur un serveur. La virtualisation permet d'utiliser les machines au maximum de leurs capacités.
- **Encombrement :** Il n'y a pas forcement toujours la place pour rajouter un serveur dans le local informatique, l'économie des serveurs liée à la fusion par la virtualisation résout cette problématique.
- **Ecologique :** Une salle serveur nécessite de l'électricité pour les serveurs mais aussi pour la climatisation. Il faudra également recycler les serveurs lors de leur fin de vie. Encore une fois, la virtualisation réduit les coûts (et éventuellement allège les consciences).
- **Migration :** Grâce à la migration à chaud des machines, la continuité de service est garantie (AMD migrant - via VMware Esx -, par exemple, permet une migration sans interruption de service[1]).

On voit que la virtualisation est indispensable pour réduire efficacement les coûts, garantir une stabilité de service et une véritable continuité associée à des performances élevées.

Cependant, il existe différents types de virtualisation, qui ne se valent pas forcément et qui correspondent à différents types d'utilisation. Certains nécessitent des modifications du système (hôte ou virtualisé), d'autres des processeurs adaptés, et tous possèdent des spécificités qui font leur intérêt. Dans ce rapport, nous tenterons de faire le point. Nous détaillerons le fonctionnement de leurs mécanismes lorsque c'est nécessaire pour en comprendre les spécificités.

1 La virtualisation de systèmes non-modifiés

On distingue deux catégories au sein de ce type de virtualisation :

1.1 La virtualisation non-assistée par matériel

Il s'agit de la virtualisation la plus connue car elle est la plus évidente à réaliser, même pour un novice en informatique.

Ce type de virtualisation fournit une abstraction totale du système physique présent. Le logiciel émule tous les composants : processeur, bios, disque dur, vidéo. Il intercepte la plupart des instructions du système invité et les

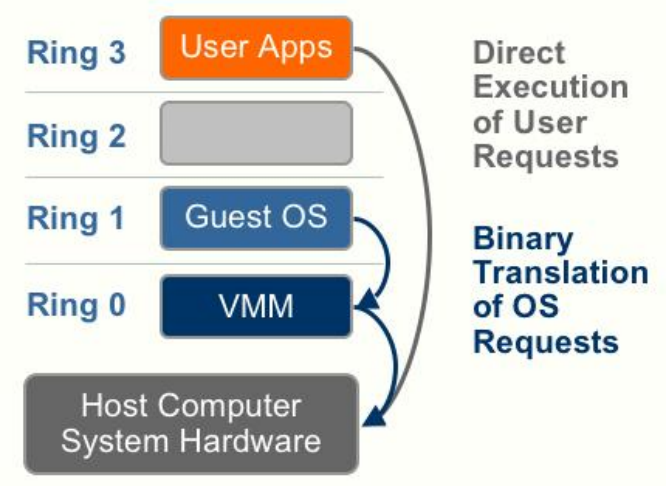
[1] http://www.zdnet.fr/actualites/informatique/0,39040745,39389029,00.htm

remplace par l'équivalent sur le système hôte.

Les machines n'ont pas conscience de l'émulation car elles sont virtualisées sans aucun changement dans le système d'exploitation.

Ce genre de virtualisation, dont les outils sont principalement graphiques, servent principalement dans le cadre de tests pour les développeurs, ou pour tester un système temporairement. Il n'est pas concevable d'héberger un serveur en production avec ce genre de virtualisation. Les machines virtualisées n'ont pas conscience de l'être, ce qui permet de faire tourner à peu près tous les systèmes possibles.

Exemples : Virtualbox (que nous testerons dans la suite de ce rapport), VMWare, Qemu, etc.



1.2 La virtualisation assistée du matériel

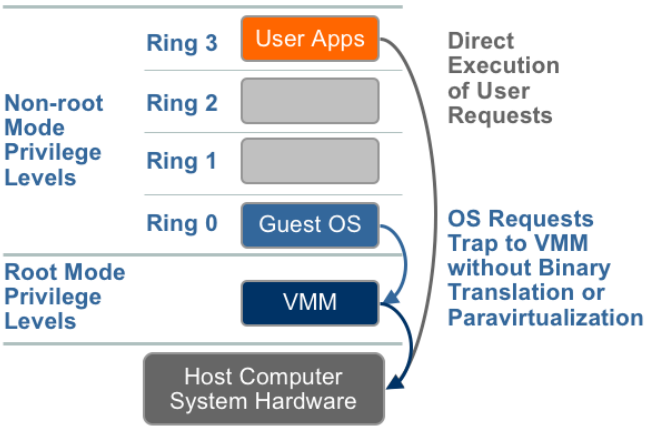
Le logiciel de virtualisation se sert d'un processeur particulier disposant de technologies AMD-V ou Intel-VT.

Ce type de processeur contient un jeu d'instructions implémentant un ensemble de primitives de bas niveau qui facilitent la virtualisation. Ces technologies permettent le support de plusieurs systèmes d'exploitations différents sans modifier le système invité.

Ce type de virtualisation apporte de meilleures performances que la solution vue ci-dessus. Elle est utilisée pour des tests ou virtualiser des serveurs en production. Les systèmes virtualisés n'ont pas conscience de l'être (aucune modification n'est nécessaire), ce qui permet de virtualiser d'autres systèmes que Linux. Le principal inconvénient de cette technologie par rapport à la précédente, réside dans la nécessité d'avoir un processeur récent et compatible.

A partir de la version 2.6.29 des noyaux Linux, des fonctions de virtualisation assistées par matériel seront intégrées. Ainsi, des solutions comme KVM pourront prendre leur envol puisqu'elles proposent de la virtualisation sans patch pour la machine hôte. Des distributions Linux pourront virtualiser d'autres systèmes sans qu'aucun système ne soit modifié. A conditions bien sûr de posséder un processeur compatible. Nous pouvons noter que Ubuntu et Debian prévoient d'abandonner le support de solutions comme Xen qui nécessitent une machine hôte patchée pour KVM.

Exemples : Xen, KVM, VMWare, etc.



Comment savoir si votre processeur supporte l'émulation

```
egrep '^flags.*(vmx|svm)' /proc/cpuinfo #(svm etant pour un processeur amd et vmx
    pour un intel)
```

1.3 Test de Virtualbox

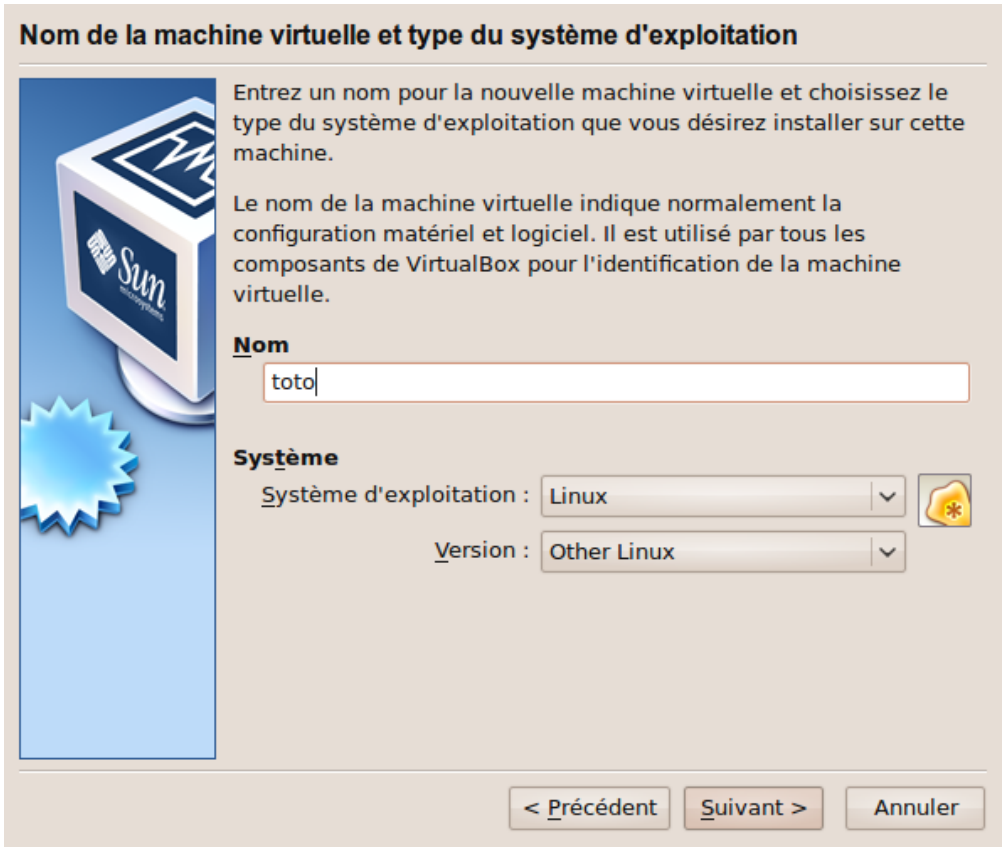
Virtualbox est un logiciel créée par la société Innotek en licence GNU GPL en janvier 2007 et a été repris par Sun Microsystems le 12 février 2008.

Il existe plusieurs versions de Virtualbox, une gratuite sous Licence PUEL et une version libre : Virtualbox OSE qui est entièrement libre mais amputée de certaines fonctionnalités (le support de l'usb par exemple).

On commence par installer la version libre de Virtualbox (le test se fait sous une Ubuntu).

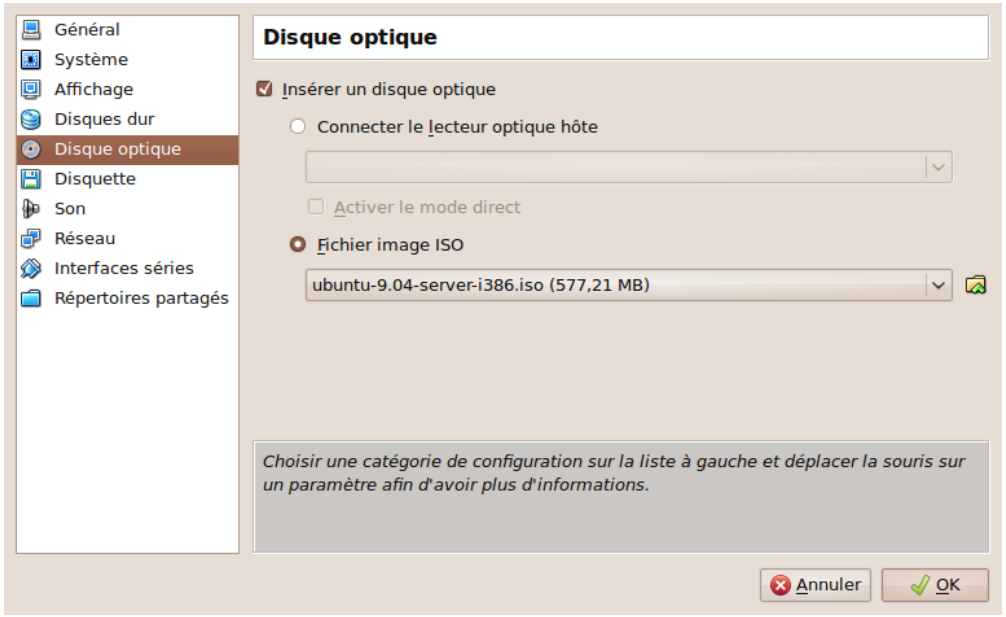
```
sudo apt-get install virtualbox-ose
```

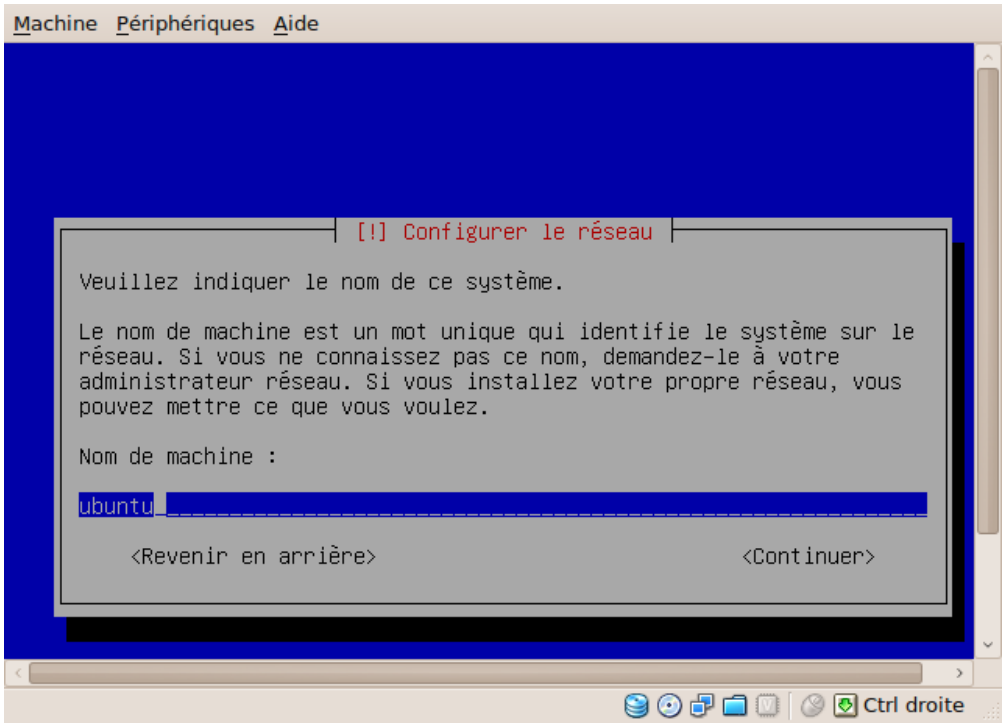
On crée ensuite une machine virtuelle en cliquant sur nouveau :



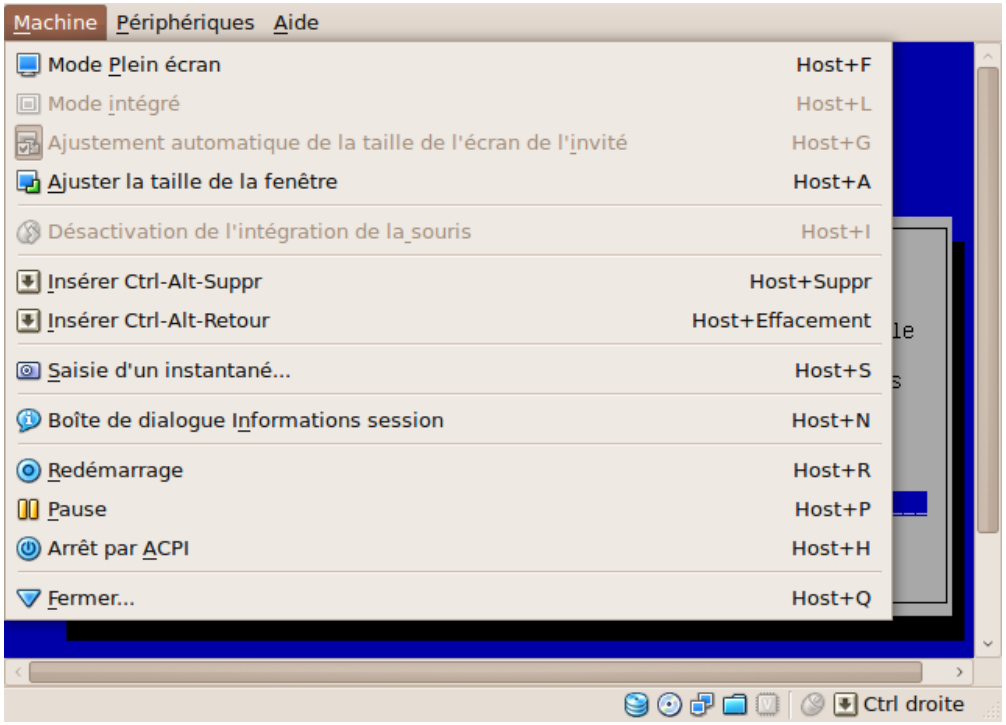
On spécifie ensuite le type de système voulu (noyau 2.4, 2.6, etc.), la taille de la mémoire vive et la taille du disque dur voulue.

On spécifie ensuite l'iso (ou le disque optique) que l'on veut utiliser pour installer la machine virtuelle (simulation de démarrage de la machine, avec reconnaissance d'un pseudo-cdrom) :





Simulation d'une installation d'Ubuntu.



La machine peut être mise en pause, redémarrée ou bien arrêtée.

2 La paravirtualisation

2.1 Définition

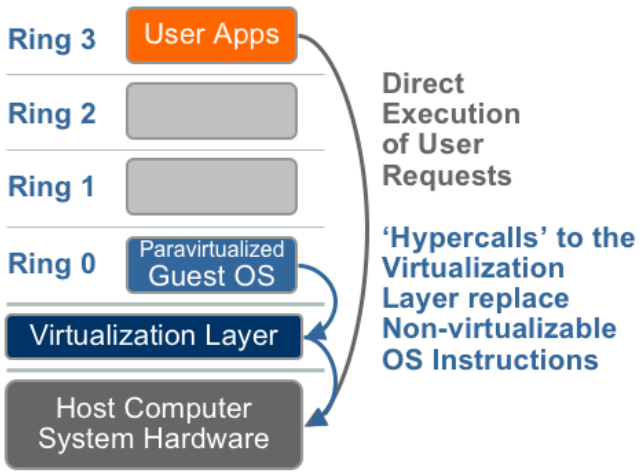
Il s'agit de la virtualisation d'un système d'exploitation modifié au préalable et sous le contrôle d'un système hôte.

Le système invité communique avec le système hôte via un hyperviseur (ou moniteur de machines virtuelles).

L'hyperviseur permet de faire fonctionner simultanément plusieurs systèmes d'exploitations. Ces derniers communiquent via une API définie par l'hyperviseur.

Ce type de virtualisation requiert la modification des OS invités. Le système se sait virtualisé et va donc travailler avec la couche de virtualisation plus efficacement.

Ce type de virtualisation permet des performances bien plus importantes que la virtualisation totale (assistée par matériel, que nous avons vu plus haut). Malgré tout, le système étant modifié, les systèmes compatibles sont réduits (Linux ou *BSD). Enfin, la nécessité de patcher les noyaux des systèmes virtualisés impose de disposer d'un noyau pour lequel la communauté a développé un patch. Certains hyperviseurs comme Xen sont très en retard par rapport à la version actuelle des noyaux (officiellement, le dernier noyau compatible est le 2.6.18), ce qui laisse l'avenir de Xen dans le doute si la communauté ne reprend pas le développement.



Ce schéma reprend l'approche expliquée précédemment.

Les systèmes modifiés communiquent avec l'hôte d'une façon ultra-optimisée, les performances sont vraiment très élevées et sont proches du niveau natif. Toutefois, les systèmes virtualisés, en plus de l'hôte, nécessitent un système patché. La contrainte d'être dépendant du développement de ceux-ci en fonction des nouveaux noyaux est donc encore plus forte, et peut s'avérer plus compliquée si un système virtualisé nécessite déjà d'autres optimisations du noyau.

Exemples : Xen (XenSource, qui a racheté il y a peu), xVM (Sun), Hyper-V (Microsoft), etc.

2.2 Test de Xen

Xen est un logiciel Open Source reconnu pour ce type de virtualisation (il supporte également la virtualisation assistée par matériel, vue plus haut). Il a été développé par l'université de Cambridge par Ian Pratt et a été dévoilé en 2003.

Il permet de virtualiser les ressources physiques d'une machine et de les répartir afin de pouvoir faire fonctionner plusieurs systèmes d'exploitations simultanément.

Le démarrage d'un système patché pour Xen, et virtualisant d'autres machines se fait en plusieurs étapes :

- Lors du démarrage de Xen via le bootloader, ce dernier charge l'hyperviseur Xen en lieu et place du noyau Linux.
- L'hyperviseur Xen se charge de gérer les temps d'utilisation CPU de chaque domaine et de détecter les processeurs non initialisés par le bios.
- L'hyperviseur charge le noyau Linux modifié. Le premier domaine de Xen est appellé dans la terminologie Xen : le domaine 0 (dom0) qui est crée automatiquement lors du démarrage du système.
- Enfin, les machines virtualisées sont démarrées par le dom0.

Le processus `Xend` est un *daemon* en Python qui gère les différentes machines à l'intérieur de ce domaine 0 en générant différentes consoles d'accès.

Les machines invitées sont appelées les « user domains » (domUs) et sont sous le contrôle du dom0.

Pour administrer les machines, le dom0 possède le programme `Xm` permettant de créer et gérer des domUs. C'est une application cliente qui envoie ses commandes au *daemon* Xend et qui lui-même transmet les commandes à l'hyperviseur.

2.2.1 Test de Xen

Cette installation se fera sur une Debian Lenny, disposant d'un noyau `2.6.26`.
mais patché afin de fournir les fonctionnalités Xen.

```
sudo apt-get install xen-linux-system-2.6.26-2-xen-686
```

On installe donc l'image xen (l'hyperviseur sera également installé).

```
sudo apt-get install install xen-tools
```

On peut également installer le paquet xen-tools permettant de créer facilement des images compatibles(via debootstrap par exemple).

```
sudo /etc/init.d/xend start
sudo /etc/init.d/xendomains start
```

On démarre ensuite les daemons si ils n'ont pas été démarrés lors du boot de Xen.
Ensuite il faut modifier la configuration du fichier xend-config.xsp afin de créer une configuration par pont réseau :

```
vim /etc/xen/xend-config.xsp

(network-script network-bridge)
(vif-script vif-bridge)
```

2.2.2 Création d'un domaine guest

On doit tout d'abord modifier la configuration par défaut de Xen dans le fichier xen-tools.conf en décommentant certaines parties :

```
#On demande le password voulu pour l'image qui va être créee.
passwd = 1

#Configuration automatique de l'ip via dhcp
dhcp=1
```

Grâce à xen-tools, on peut créer facilement un domaine invité avec la commande :

```
xen-create-image --hostname testing1
```

Les images sont ensuite créées dans le dossier utilisateur désiré et la configuration se trouve dans notre cas dans /etc/xen/testing1.cfg
Ensuite on lance la machine virtuelle :

```
xm create /etc/xen/testing1.cfg
Using config file "/etc/xen/testing1.cfg".
Started domain testing1
```

On peut se connecter à la machine virtuelle via ssh ou via la console en passant en argumant le nom de la machine virtuelle

```
xm console testing1
```

On peut également lister toutes les machines virtuelles présentes sur le dom0 et leur consommation RAM/CPU

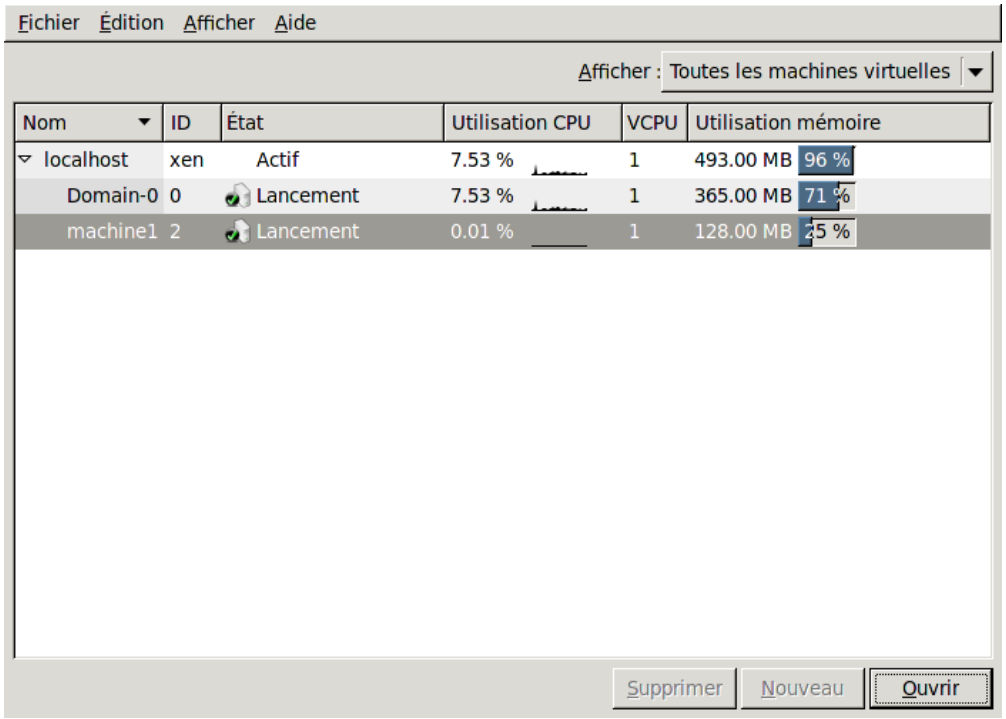
```
xm list
Name ID Mem(MiB) VCPUs State Time(s)
Domain-0 0    96   1    r----- 13840.7
testing1      1   128   1    -b----- 749.9
```

2.3 Méthodes de supervision

En tant qu'administrateur réseau, on a utilité à superviser les différentes consommations (CPU/RAM) des différentes machines virtuelles. Il serait long et fastidieux de superviser l'ensemble des machines virtuelles en se restreignant aux commandes comme `xm list` (pour Xen).

Il existe heureusement des outils facilitant le travail comme `convirt` ou encore `virt-manager`, qui sont plus graphiques et plus agréables.

`virt-manager` est une application en Python conçue par Red-hat pour gérer différentes machines virtuelles sous Xen ou KVM. L'API de virtualisation `libvirt` permet à Virt-manager d'interagir avec l'hyperviseur.



On peut voir ici l'utilisation en temps CPU/RAM du dom0 et des éventuels domUs. Il est également possible d'utiliser VNC pour communiquer avec les différents domUs.

3 La virtualisation au niveau noyau (containeurs)

3.1 Définition

La virtualisation par containeurs se démarque des autres solutions de virtualisation par plusieurs points. Ce type de virtualisation n'émule aucun matériel, il s'agit d'une simple gestion des machines par le noyau. La machine qui recevra les différents serveurs virtualisés devra donc disposer d'un noyau capable de supporter cette technologie : sous Linux, ce code n'est pas intégré aux sources des noyaux officiels, il faut donc installer un noyau patché. Les serveurs ainsi virtualisés partagent donc le noyau de la machine hôte (qui n'est donc plus dupliqué, comme pour les autres types de virtualisation), et ne nécessitent que très peu de ressources. C'est un système de virtualiation léger et peu intrusif. Il est possible pour quelques outils spécifiques (comme VServer) d'utiliser un utilitaire permettant de mutualiser un maximum de fichiers « constants » (à l'extrême, il devrait être possible d'obtenir des systèmes en production pour lesquels seuls les répertoires `/etc` et `/var`, et éventuellement un dossier de stockage, sont variables) entre toutes les machines, permettant ainsi de réduire le poids de chaque machine à quelques mégas-octets.

Ce type de virtualisation est particulièrement indiqué pour des solutions de sécurité. Il est courant de n'attribuer qu'un service par containeur (serveur virtualisé), ainsi si on décide de faire fonctionner SSH sur un serveur de ce type, il sera impossible pour la personne qui se connecte de « remonter » le serveur pour atteindre la machine hôte. Cette technique s'apparente au `chroot`, en beaucoup plus sécurisé.

Au démarrage de la machine physique, tous les processus lancés par le noyau (init) sont considérés comme appartenant au contexte 0. En effet, chaque processus est « taggué » avec un numéro de contexte (pour un VServer, il s'agit du « xid »), permettant de savoir à quelle machine il appartient. Ainsi, l'utilisateur d'un serveur virtualisé sera lui-même « taggué » et ne pourra interagir qu'avec les processus de son contexte. Cette façon de fonctionner assure un niveau de sécurité supplémentaire au cas où un individu réussirait à « remonter » le serveur pour atteindre la machine hôte ainsi que les autres serveurs virtualisés.

De plus, la technologie de virtualisation par containeur se sert des possibilités du noyau Linux pour ne pas passer par les couches basses telles que les couches des stockage et de réseau. Ainsi, les appels systèmes et les manipulations de l'environnement sont largement bridés. Impossible de manipuler les interfaces (i.e. changer l'adresse I.P.), d'utiliser iptables, ou même de sniffer le réseau (du moins pour VServer, OpenVZ permettant par exemple d'autoriser ce genre d'outils au sein d'un containeur). Comme ce genre de serveur est souvent dédié à une tâche unique, il est possible de limiter le nombre de processus, assurant encore une fois un nouveau cran de sécurité.

Au démarrage d'un serveur-containeur, les outils associés à la technologie vont consulter un fichier de configuration correspondant au serveur virtuel à démarrer. Un nouveau contexte unique est créé, puis le processus change de racine, à l'instar d'un `chroot`. Chaque serveur possède une adresse I.P. inscrite dans sa configuration. Une adresse secondaire de l'interface de la machine hôte est alors créée avec cette I.P., et des règles de routage internes au noyau sont définies pour rediriger les accès à la machine par cette adresse vers le serveur virtualisé concerné. Un processus init propre au serveur est alors lancée, le serveur virtuel est démarré. Seul le pseudos système de fichiers `/proc` est monté, en étant restreint strictement à tout ce qui concerne le contexte du serveur créé.

Les disques et la mémoires sont directement partagés, ce type de virtualisation n'émulant rien. Les processus isolés et bridés peuvent alors avoir directement accès aux ressources de la machine, sans limite et sans pertes de performances. D'un autre côté, il est important de contrôler cette utilisation des ressources physiques, au risque de pénaliser les autres machines en en surchargeant une. Les différents outils permettant ce genre de technologie permettent souvent de brider les ressources allouées aux différents serveurs. Il est également conseillé d'allouer des partitions spécifiques aux différents serveurs, afin de cloisonner efficacement les possibilités d'utilisation des disques.

L'installation d'un serveur se fait souvent grâce à des outils permettant de récupérer les fichiers de la distribution souhaitée. Elle est alors installé puis « adaptée » par les outils associés au type de virtualisation. Dans la mesure où les machines virtualisées ne partagent que le noyau avec la machine hôte, il est envisageable d'installer une distribution différente de celle de la machine hôte (nous n'avons pas tenté l'expérience). Par contre, il est évidemment impossible d'installer autre chose que des serveurs Linux.

3.1.1 Les avantages

L'avantage majeur des containeurs est sans conteste la légèreté. Avec un service par serveur, des fichiers mutualisés et une absence de noyau, c'est sans aucun doute la solution qui permettra au mieux la multiplication des serveurs sur une même machine physique, sans nécessiter des ressources démesurées. De plus, il est souvent possible de créer des « patrons » de containeurs afin de multiplier encore plus facilement les serveurs.

Le fait-même que rien ne soit émulé donne la possibilités aux processus des serveurs virtualisés d'utiliser directement les ressources de la machine physique de la même façon que s'ils n'étaient pas virtualisés (hors la gestion des contextes, qui demande des ressources très minimes).

Les restrictions d'accès au système rendent cette solution très sécurisée. Les redirections de port et les règles de routage sont définies par le `netfilter` de la machine hôte (par le biais notamment des règles de pré-routage qui permettent de rediriger un service vers l'interface d'un VServer en fonction du demandeur, par exemple[2]) et peuvent n'être spécifiées qu'en fonction du service précis auquel est attribué chaque machine. Ainsi, même si un utilisateur se retrouve root sur une machine profitant de ce type de virtualisation, il sera impossible pour le hacker d'utiliser les outils classiques (nmap, tcpdump, etc) pour analyser le réseau. Il ne pourra donc pas initier de connexions depuis le serveur et ne pourra pas installer de services comme un SSH clandestin, puisqu'il n'aura aucun accès à la gestion des ports et du routage. Il ne pourra pas non plus supprimer les données de la machine hôte (même pourra tout de même supprimer les données du serveur virtualisé). Contrairement à un `chroot` classique, il est impossible de sortir de la racine imposée, que ce soit par un code C ou par l'usage de `/proc` (qui sont les deux méthodes bien connues pour « casser » un `chroot`).

Enfin, il est intéressant de noter que des distributions comme Debian (depuis Sarge) proposent dans leurs systèmes de paquets des sets d'outils pour la gestion de ces serveurs, ainsi que des noyaux déjà patchés prêts à l'emploi. La mise en place de cette technologie est donc très facile d'accès.

Avec une charge aussi faible, une sécurité aussi accrue et une gestion aussi simple, les serveurs virtualisés de type containeurs sont d'excellents candidats pour la division des différents services de serveurs Linux, en offrant une sécurité solide pour une perte de performances très faible.

3.1.2 Les inconvénients

Ce type de virtualisation n'émulant aucun matériel et laissant les machines virtualisées en lien constant avec la machine physique et le système hôte, il est bien plus sensible au exploits de hacking. Ainsi, il est arrivé une fois qu'une faille de sécurité du noyau patché permette de casser la virtualisation en remontant sur la machine hôte. Celle-ci ayant accès à tous les serveurs qu'elle gère, pouvant accéder sans contrainte à leurs fichiers et au processus qui y sont démarrés, en plus de pouvoir rentrer dans chacune d'entre elle en utilisateur root, c'est dans ce cas non plus un simple serveur qui est hacké mais bien une multitude de serveurs servis sur un plateau. Il n'y a cependant eu qu'un seul exploit de ce type, basé sur une faille vite réparée, et aucun probléme de sécurité n'est recensé à ce jour.

Le fait que le matériel ne soit pas émulé mais bridé pose quelques problèmes pour les applications qui tentent d'accéder à des couches de communications qui leur seront inaccessibles. Installer et faire tourner un serveur X sur une machine sera compliqué. De plus, l'utilisation de la couche réseau par plusieurs interfaces à la fois entraîne des régles de routage internes compliquées, et rend notamment l'utilisation de IPv6 à ce jour impossible.

Le partage du noyau permet des gains de performance extraordinaire, mais interdit la virtualisation de tout autre système que Linux. Enfin, un sérieux problème réside dans la fréquence de l'équipe de développement du projet à produire des patchs récents.

[2] C'est ainsi que fonctionne par exemple les accès aux services de la passerelle de la salle A.S.R.A.L.L. : selon le vlan à qui appartient l'adresse I.P. du demandeur d'une connexion SSH, sa requête est « pré-routée » vers l'adresse I.P. d'un VServer proposant lui-même un accès SSH. Cette astuce permet de dérouter (sans jeu de mots) les tentatives de pénétration du serveur - que nous avons tout de même réussi à pénétrer dans son intégralité - en ne proposant insidieusement qu'un accès sur un VServer dénué d'intérêt.

3.2 Quelques solutions

3.2.1 Linux-Containers

C'est probablement le projet le plus prometteur à ce jour. Sa spécificité est qu'il n'impose pas de noyau patché : les noyaux récents (à partir de 2.6.29) disposent de suffisamment de possibilités pour son fonctionnement. Il est prévu qu'il soit intégré dans la future version de Ubuntu, ainsi que de Debian. Outre sa récente popularité, le projet se démarque par le soutien que lui accorde IBM[3]. Toutefois, le projet encore jeune n'offre pas encore les mêmes possibilités que ses concurrents (comme la mutualisation des fichiers, par exemple). Nous n'avons pas eu l'occasion de tester cette solution.

3.2.2 OpenVZ

OpenVZ est une solution libre basée sur le code d'un produit propriétaire nommé Virtuozzo (SWsoft). Plutôt qu'un long discours, voici un tableau récapitulatif, qui compare OpenVZ à VServer (qui est la solution que nous avons le plus étudié) :

	Quotas disque	Quotas RAM	Quotas CPU	Migration à chaud	Isolation réseau	Iptables	Loopback	IPv6
VServer	1	1	1	0	1	Pour l'hôte	0	0
OpenVZ	1	1	1	1	1	1	1	1

On peut constater que OpenVZ n'a pas la même politique en matière de privilèges pour les systèmes invités. Si il peut être très utile de pouvoir permettre aux invités d'accéder à des fonctionnalités comme iptables, on peut préférer VServer pour la rigueur qu'il impose. Le confort ou la sécurité, c'est un peu le dilemne entre ces deux solutions. Au point de vue patchs et fonctionnement, les deux solutions sont à peu près similaires. On peut noter que la documentation semble plus importe pour OpenVZ.

3.3 Test de VServer

3.3.1 Linux-VServer

Le projet Linux-VServer (que nous appellerons « VServer » dans la suite de ce rapport), est un projet open source tournant exclusivement sous Linux et destiné à n'héberger que des distributions linux (le fonctionnement de la virtualisation par containeurs imposant elle-même cette contrainte). Il date de 2001, et est maintenu par une communauté. Depuis 2005, le projet propose une version 2.0 destinée aux noyaux 2.6, qui apporte un lot important d'améliorations. Des noyaux patchés sont disponibles directement dans les dépôts pour Debian (et probablement d'autres distributions), si on ne souhaite pas (pour l'instant) un noyau supérieur à la version 2.6.26.

Au travers d'une arborescence de fichiers de configuration, il est possible de définir assez finement les différentes ressources ressources disponibles (CPU, RAM, etc.). Par ce même moyen, il est possible de définir une collection de privilèges à accorder à une machine (droit de chroot, killer un processus, redémarrer le serveur, etc.).

Enfin, une collection d'outils permettent de facilement créer un VServer, « entrer » dedans, le redémarrer, voir son statut, le supprimer, etc.

3.3.2 Installation

Cette installation se fera sur une Debian Lenny, disposant d'un noyau `2.6.26`.

Il devra être installé :
- un noyau patché pour VServer (Debian disposant de noyaux patchés dans ses paquets, il est préférable de les utiliser)
- les outils de VServer, qui permettront de créer de nouveaux contextes, les gérer et les supprimer

En une ligne (`linux-image-vserver-amd64` est aussi disponible) :

[3] http://www.ibm.com/developerworks/linux/library/l-lxc-containers

```
apt-get install linux-image-vserver-686 util-vserver
```

Après un redémarrage :

```
expose:~# uname -r
2.6.26-2-vserver-686
```

Des noyaux patchés peuvent aussi être trouvés sur le site de Zbla[4] et les patchs sont téléchargeables sur le site de Linux-VServer[5] ou 13thfloor[6]. Attention, les seuls patchs disponibles pour des noyaux récents (au dessus de `2.6.22`) sont en expérimental.

Votre système fait à présent office d'hôte (« *host* »), et sera le seul habilitée à créer, gérer et supprimer des machines hébergées : les « *guests* ». Lors du redémarrage, le noyau a considéré qu'il s'agissait du contexte zéro.

3.3.3 Création d'un VServer (guest)

La création d'un *guest* nécessite de récupérer l'arborescence d'une distribution. Dans la mesure où seul le noyau est partagé, on pourrait envisager l'installation d'une distribution différente de la machine hôte. Pour cet exemple, nous garderons Debian. La méthode la plus simple et la plus efficace est d'utiliser debootstrap qui nous permettra de récupérer les sources d'une Debian Lenny.

Depuis la machine hôte :

```
expose:~# vserver asrall build \
-m debootstrap --context 42 \
--hostname vsasrall.asrall.iuta.univ-nancy2.fr \
--interface eth0:192.168.10.102/24 \
-- -d lenny -m http://ftp.fr.debian.org/debian
```

Pour forcer l'installation en 32 bits sur une machine hôte en 64 bits, ajouter l'option `- -arch i386`.

Ainsi est créé un serveur virtualisé basé sur Debian Lenny, de nom « asrall » et d'adresse I.P. `192.168.10.102`. On peut noter que nous avons forcé un numéro de contexte, ici 42. Ne pas le préciser aurait laissé le système l'allouer dynamiquement (nous reviendrons sur l'allocation dynamique des contextes lorsque nous détaillerons les différents fichiers systèmes de VServer).

Il ne reste plus qu'à le lancer :

```
expose:~# vserver asrall start
```

3.3.4 Utilisation et configuration

Pour entrer dans un VServer :

```
expose:~# vserver asrall enter
asrall:/#
```

Vous pouvez alors en toutes liberté installer le service que vous déciderez de lui associer. Il est donc possible, par exemple, d'y installer SSH et d'accéder directement au VServer en utilisant l'adresse I.P. demandée lors de sa création. Attention toutes fois, dans le cas de SSH il est nécessaire de brider la plage d'adresses d'écoute (par défaut, toutes) du démon SSH de la machine hôte.

Pour ce faire, ajouter dans le fichier `/etc/ssh/sshd_config` :

```
ListenAddress 192.168.10.74
```

Il faut bien entendu adapter l'adresse I.P. avec celle de la machine hôte. Sans cette manipulation, n'importe quelle connexion SSH via l'I.P. d'un VServer sera interceptée par le démon de la machine hôte.

[4] http://zbla.net/debian
[5] http://linux-vserver.org
[6] http://www.13thfloor.at/vserver/s_rel26/overview/

3.3.5 Conclusion

VServer est une solution performante et facile d'accès pour la virtualisation par containeurs. Toutes fois, les patchs suivent difficilement les évolutions des noyaux Linux, et le retard est aujourd'hui relativement important. L'équipe Debian a lancé un ultimatum à la communauté en leur signifiant que les noyaux patchés ne feraient pas partis des dépôts de la future version de la distribution si un patch officiel pour la dernière version du noyau Linux ne sortait pas d'ici là.

Dans tous les cas, Debian a décidé (aux côtés de Ubuntu) de désormais privilégier le projet Linux-Containers[7] qui a l'avantage non-négligeable de ne nécessiter que des fonctionnalités intégrées dans les noyaux Vanilla, les noyaux de base de Linux. Malgré tout, ce projet est encore jeune et n'offre pas encore les mêmes possibilités que VServer, qui a encore probablement quelques années devant lui.

L'objectif de l'équipe de Linux-VServer est de se rapprocher peu à peu des fonctionnalités de virtualisation par containeurs offertes par les nouveaux noyaux pour arriver à pouvoir se passer de patchs[8], et donner ainsi un nouveau souffle au projet, qui le replacera au niveau de son dangereux concurrent... et pourquoi pas lui reprendre la place si l'écart des possibilités ne s'est pas comblé d'ici là.

Pour ceux qui souhaiteraient migrer leurs serveurs de VServer à Linux-Containers (LXC), il existe des retours d'expérience intéressants[9].

Conclusion

La virtualisation est une technologie récente qui révolutionne l'organisation des systèmes. Malgré tout, on trouve un peu tout et n'importe quoi sur Internet, à tel point qu'on constate souvent de grosses incohérences et confusion entre les différents types de virtualisation. Outre les aspects techniques, il nous a donc semblé important de mettre l'accent sur les différents types de virtualisation, et souligner le type d'utilisation auquel ils sont destinés. C'est pourquoi nous conclurons ce rapport avec un tableau récapitulatif des principaux types de virtualisation, en faisant un point sur la nécessité de patcher, en citant quelques exemples de logiciels concernés et en ajoutant un court commentaire sur les spécificités et le cadre d'utilisation potentiel.

Type de virt.	Patchs	Solutions	Commentaires
Sys. non-modifiés (non-assistés par matériel)	Aucun	Virtualbox, VMWare	Tests pour des développeurs, dans un environnement graphique. Très lourd.
Sys. non-modifiés (assistés par matériel)	Patch uniquement pour l'hôte avant les noyaux 2.6.29 et aucun au delà, pour certaines solutions	KVM, Xen	Plus optimisé, mais nécessite un processeur compatible. Utilisé pour des serveurs en production, qui virtualisent d'autres systèmes que Linux.
Paravirtualisation	Pour l'hôte et les invités	Xen, Xvm, Hyper-V	Très performant, mais nécessite des noyaux modifiés. Parfait pour virtualiser des systèmes Linux.
Containeurs	Pour l'hôte, qui est le seul à avoir un noyau, avant les noyaux 2.6.29, et aucun au delà pour des solutions comme LXC	LXC, VServer, OpenVZ	Ultra-performant, un seul noyau est partagé par tous. Utilisé pour séparer des services au sein d'un serveur, un service étant attribut à un containeur.

[7] Réf. : Lucas Naussbaum, en revenant d'une réunion organisée par Canonical concernant la future version d'Ubuntu.
[8] Réf. : Laurent Vallar lors d'une pause cigarette.
[9] http://blogpmenier.dynalias.net/docext/vs2lxc

4 Webographie

- http://doc.ubuntu-fr.org
- http://www.xensource.com
- http://www.bortzmeyer.org/xen.htm
- http://cesar.com.univ-mrs.fr/spip.php?article127
- http://fr.wikibooks.org/wiki/Linux_VServer.fr/IMG/pdf/Les_linux-servers_au_quotidien.pdf
- http://fr.wikibooks.org/wiki/Linux_VServer
- http://linux-vserver.org/Overview
- http://blogpmenier.dynalias.net/index.php?category/Linux