État de l'art des solutions libres de virtualisation pour une petite entreprise

Lucas Bonnet — Bearstech
lbonnet@bearstech.com — http://bearstech.com

Conventions typographiques

Les termes techniques français sont suivis, lors de leur première mention, de leur équivalent anglais entre parenthèses et en italique *(comme ceci)*.

Les termes suivis d'un astérisque (*) sont définis dans le glossaire, page 83.

Les noms de programmes et de commandes systèmes sont composés en police à chasse fixe, `comme ceci`.

Licence

Sommaire

Introduction

Depuis quelques années, la virtualisation est au cœur des préoccupations des entreprises du secteur informatique. En effet, on assiste à une montée en puissance des acteurs du marché, que ce soit dans le domaine propriétaire avec Microsoft et VMware, ou dans le domaine des logiciels libres, avec l'émergence de nombreux projets autour de la virtualisation. Il suffit de voir le nombre de conférences liées aux technologies de virtualisation pour l'entreprise et le nombre d'articles de presse (en ligne ou papier) traitant de la virtualisation. Cette montée en puissance n'est pas due au hasard : elle suit de très près la demande du marché, qui se tourne de plus en plus vers les technologies de virtualisation.

La virtualisation

L'encyclopédie francophone en ligne Wikipédia définit la virtualisation comme « l'ensemble des techniques matérielles et/ou logicielles qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes » [WFv]. Il s'agit donc d'utiliser une seule machine physique en remplacement de plusieurs et d'utiliser les possibilités offertes par la virtualisation pour démultiplier le nombre de machines virtuelles.

Prenons l'exemple d'une solution de virtualisation faite pour le grand public, de type VMware ou QEMU : l'utilisateur possède un seul ordinateur, sur lequel est installé un système d'exploitation (Microsoft Windows, GNU/Linux, Mac OS X, etc.) ainsi qu'une application qui fait office de machine virtuelle : le logiciel installé par VMware ou QEMU. L'utilisateur peut à partir de là se servir de ce programme pour démarrer un nouveau système d'exploitation (qui peut être totalement différent de celui installé sur

la machine physique). Le système d'exploitation virtualisé — aussi appelé système invité *(guest system)* — est alors exécuté par la machine virtuelle et est complètement détaché de tout le matériel de l'ordinateur. La machine virtuelle se charge d'émuler* pour le système invité tout le matériel « standard » d'un ordinateur (disque dur, écran, clavier, souris, ...). L'utilisateur peut alors utiliser le système invité comme un système normal : installer des applications, naviguer sur Internet, exécuter un programme, etc. Le système hôte — installé sur la machine physique — et le système invité sont totalement indépendants : le système invité est vu par l'hôte comme un simple programme, il n'a pas d'accès direct au matériel contrairement à l'hôte.

Toutefois, la virtualisation ne se limite pas uniquement à une utilisation grand public : elle recouvre plusieurs champs d'application, via plusieurs technologies et pour plusieurs objectifs. La définition reste vague, car sous une appellation unique se cachent énormément de notions à prendre en compte. Les buts et les usages de la virtualisation varient également beaucoup selon les besoins et les catégories d'utilisateurs. Les paragraphes suivants seront donc consacrés à une brève présentation de quelques cas d'utilisation — sans détailler quelles solutions techniques sont adaptées à chaque usage —, ce qui permettra de cerner tout ce qu'apporte la virtualisation pour toutes les catégories d'utilisateurs.

Intérêt de la virtualisation

Pour le particulier, la virtualisation permet d'avoir accès à des applications ne fonctionnant pas sur le système d'exploitation principal de l'utilisateur. On peut notamment citer les applications trop vieilles pour s'exécuter sur la dernière génération du système d'exploitation (les anglophones parlent de *legacy applications**). Un autre domaine couvert par la virtualisation est l'utilisation de programmes non portés sur la plate-forme cible (architecture PC *vs* architecture Mac [1], par exemple). On peut aussi citer, même si c'est plus rare, l'utilisation de virtualisation pour les jeux : faire fonctionner un jeu fait pour Microsoft Windows dans une machine virtuelle s'exécutant sur un système GNU/Linux. La société VMware propose notamment l'accès à l'accélération 3D depuis une machine virtuelle pour son produit grand public, permettant d'atteindre des perfor-

1. Avant le passage aux processeurs Intel, amorcé fin 2005.

mances proches de l'original.

Pour les professionnels et les chercheurs en sécurité, un système d'exploitation virtualisé permet d'observer le comportement d'un logiciel malveillant *(malware*)* — virus, ver, *spyware*, etc. — dans un système sain sans avoir à infecter une machine physique. De plus, le processus d'infection est reproductible, car il suffit de sauvegarder l'état de la machine virtuelle avant l'infection pour pouvoir répéter l'opération plusieurs fois, dans des conditions contrôlées. Il est alors possible d'analyser l'état de la machine virtuelle après infection, et de tirer des conclusions sur l'action du logiciel.

Pour une entreprise, les technologies de virtualisation permettent de séparer des applications et des systèmes de manière logique, quand les prérequis des applications sont mutuellement exclusifs. Par exemple, une application critique mais incompatible avec une version donnée d'un logiciel ne peut pas cohabiter sur la même machine avec une autre application dépendant d'une autre version du même logiciel. Certains cas d'incompatibilités peuvent se résoudre en laissant installés les deux logiciels dans deux versions différentes, mais le surcoût de maintenance est non négligeable.

En plus de la simple incompatibilité de versions, deux applications peuvent aussi avoir le même rôle, mais dans des contextes différents. Par exemple une version de développement et une version finale d'un site web ne peuvent pas cohabiter de manière simple, à moins d'y consacrer un effort de maintenance là aussi conséquent. Pour le développement d'une application web, le test du site sous plusieurs navigateurs est primordial. La virtualisation de plusieurs systèmes d'exploitation permettra aux développeurs de tester le rendu de plusieurs navigateurs sur plusieurs plates-formes sans avoir à changer de machine — et donc d'environnement de travail — en permanence.

Au delà de la possibilité de faire fonctionner des applications qui ne peuvent normalement pas s'exécuter sur une machine donnée, la virtualisation permet aussi de les rassembler sur une même machine physique, sans avoir à maintenir un serveur distinct par application. Traditionnellement, l'usage était de consacrer une machine physique à un service (messagerie, stockage, hébergement d'intranet, etc.), tant pour des raisons pratiques (associer une machine à un rôle unique) que pour la sécurité (séparation des services).

Toutefois, cette dispersion a un coût qui n'est pas nul pour l'entreprise, que ce soit en es-

pace occupé (location au mètre carré dans les *datacenters**), en énergie (consommation électrique) ou en maintenance (plus de machines physiques implique plus de risques de pannes matérielles). De plus, la plupart des services fournis sur un réseau local (DHCP, DNS, Intranet, ...) ne consomment qu'une très faible partie des ressources offertes par une machine récente. Tous ces facteurs font qu'il n'est plus pertinent aujourd'hui d'utiliser des machines séparées pour héberger des services ne nécessitant qu'une fraction de la puissance d'une machine.

Aussi, à l'heure actuelle, la tendance est plutôt au rassemblement de plusieurs services, autrefois distincts, sur une seule machine, par le biais de l'utilisation de technologies de virtualisation pour maintenir une séparation entre les services. On parle de *consolidation* de serveurs.

Enfin, l'utilisation d'applications « anciennes » (au sens informatique du terme) est au moins aussi importante chez les entreprises que chez les particuliers. L'importance parfois critique de ces applications pour le fonctionnement de l'entreprise fait qu'il est souvent plus facile de continuer à maintenir un système et une machine obsolètes (et donc avec un risque de panne matérielle plus important) que d'entamer une migration vers une nouvelle plate-forme. La virtualisation permet dans ce cas d'exécuter l'application comme dans son environnement d'origine, mais sur du matériel récent. On peut citer, sans ordre particulier : une application de comptabilité (ou un progiciel quelconque) utilisée depuis des années mais non portée sur la nouvelle version d'un système d'exploitation ou encore un logiciel de pilotage de machine industrielle. Ce sont deux exemples classiques d'application de la virtualisation pour autre chose que de l'hébergement de services.

En plus des possibilités techniques citées ci-dessus, la virtualisation est également une technologie clef pour l'avenir de l'entreprise. En effet, elle ne permet pas seulement de contourner les limitations matérielles des ordinateurs, mais elle peut aussi fournir un avantage décisif sur la concurrence dans le milieu très disputé qu'est l'informatique de services.

Enjeux de la virtualisation

Pour une petite entreprise de services informatiques, la virtualisation peut apporter beaucoup en terme de réactivité et de flexibilité. En effet, une forte réactivité est un avantage certain pour l'entreprise et permet d'attirer et de conserver plus de clients. La flexibilité permet quant à elle d'adapter le processus de travail en fonction des besoins de la société.

Pouvoir tester très rapidement comment se comporte une application dans une configuration logicielle donnée est un avantage à ne pas négliger si l'on veut rester compétitif. Avec la virtualisation, on peut déployer très rapidement une nouvelle configuration logicielle (système d'exploitation, applications installées et configurées, environnement de développement, etc.) et l'installer aussitôt en production. Le gain de temps ainsi occasionné se mesure en heures dans une journée de travail.

De même, le déploiement d'un système ou d'une application peut très simplement se faire à distance, alors que l'installation du système d'exploitation d'une machine requiert la plupart du temps quelqu'un sur place, au moins pour les premières étapes. Si la société dispose de peu de personnel, l'économie d'un déplacement dans un *datacenter* peut être très intéressante.

En outre, la virtualisation permet de réduire le nombre de machines physiques à acheter, administrer et maintenir. Il y a donc une économie financière à la clef, qui peut être substantielle si l'entreprise a besoin de beaucoup de serveurs pour son activité. En plus du simple gain en nombre de machines, les économies réalisées en consommation d'électricité, location d'espace dans un *datacenter* et location de bande passante sont aussi à prendre en compte.

Les technologies de virtualisation sont donc très intéressantes car elles permettent de réduire le temps passé à administrer les machines et les systèmes en automatisant et centralisant la plupart des activités traditionnelles.

Toutefois, une solution de virtualisation complète requiert des compétences que n'a pas forcément la société. En effet, l'administration d'un système virtualisé diffère de l'administration d'une machine physique traditionnelle sur plusieurs points, notamment l'accès au matériel. L'enjeu est donc de savoir si le temps consacré à la formation et à

l'apprentissage vaut le temps gagné à l'utilisation, une fois la solution de virtualisation maîtrisée.

En plus de l'alternative « virtualisation ou non » il y a aussi le choix de la solution à mettre en place. En effet, il y a plusieurs projets et produits proposant de la virtualisation pour l'entreprise, tous ayant leurs points forts et leur discours commercial. Il est donc important de ne pas effectuer le mauvais choix, tant dans l'absolu (produit ou projet à l'abandon) que dans le contexte d'utilisation (inadéquation aux besoins de l'entreprise).

L'étendue du domaine couvert par l'ensemble des technologies de virtualisation est relativement important. Cela va de la virtualisation dédiée aux particuliers à la virtualisation de serveurs d'entreprise, avec à chaque fois des choix technologiques différents. L'étude menée dans le cadre de ce livre blanc sera donc consacrée à un domaine et à un type de projet précis.

Le domaine étudié sera celui de la virtualisation de serveurs pour une petite entreprise, les projets sélectionnés seront des projets ayant une licence libre.

La première partie de ce livre blanc sera tout d'abord consacrée à la définition des principes du logiciel libre, suivie d'une définition de la virtualisation, des différentes technologies utilisées ainsi qu'un historique. Les acteurs majeurs du marché de la virtualisation, tant du côté propriétaire que du côté des logiciels libres, seront ensuite traités.

La seconde partie du livre blanc portera sur la définition des besoins d'une PME en matière de virtualisation, puis sur l'étude détaillée de quelques projets phares, en explicitant notamment leurs choix techniques.

La troisième et dernière partie sera une étude comparative des solutions majeures retenues, en mettant en avant les besoins d'une PME avec une analyse de leurs forces et faiblesses.

Chapitre 1

État du marché de la virtualisation

1.1 Le logiciel libre

Les notions de logiciel libre et licence libre ont été mentionnées dans l'introduction, sans les définir. Il est important pour bien saisir les enjeux du logiciel libre de bien définir ces termes, car ce sont des éléments clés dans les choix des produits étudiés dans ce livre blanc. L'encyclopédie Wikipédia possède une définition concise du logiciel libre : « Un logiciel libre se dit d'un logiciel qui donne à toute personne qui en possède une copie, le droit de l'utiliser, de l'étudier, de le modifier et de le redistribuer » [WFl]. Le mouvement du logiciel libre trouve son origine au début des années quatre-vingt, quand Richard Matthew STALLMAN fonde la Free Software Foundation (FSF). Cette association a pour but de promouvoir et soutenir les logiciel libres, en établissant notamment les quatre libertés fondamentales :

1. La liberté d'exécuter le programme ;
2. La liberté d'étudier le fonctionnement du programme ;
3. La liberté de redistribuer des copies ;
4. La liberté d'améliorer le programme et de publier ses améliorations.

Ces libertés doivent être irrévocables, d'après la FSF : une personne (ou une entreprise) modifiant un logiciel libre n'a pas le droit de redistribuer ce logiciel en interdisant la modification, l'étude ou l'amélioration de ce dernier. Ainsi, les logiciels vont en s'améliorant, les altérations apportées étant reversées à la communauté. Le mouvement du

logiciel libre se fonde donc sur le partage de la connaissance. Les licences de ces logiciels sont appelées des licences libres et il en existe plusieurs sortes, pas toujours compatibles entre elles, en fonction des buts recherchés. En effet, certaines licences s'opposent par exemple à la commercialisation du logiciel alors que d'autres autorisent les modifications non reversées à la communauté. Cette grande diversité des licences — et les débats parfois houleux de leurs partisans respectifs — fait qu'il est souvent difficile pour un néophyte de bien saisir tous les enjeux du choix de licence d'un projet donné.

Pour résumer, il y a deux grands courants de pensée au sein du mouvement du logiciel libre, concernant les licences. Premièrement, les licences de type GNU GPL (GNU General Public License), qui garantissent les quatre libertés précédentes et sont « compatibles » avec la licence GPL. D'autre part, on retrouve les licences de type BSD (Berkeley Software Distribution) qui autorisent les sources à ne pas être redistribuées avec les programmes modifiés. Une licence de type BSD permet donc à un logiciel de devenir moins libre, alors qu'une licence de type GPL garantit exactement l'inverse, à savoir qu'un logiciel sous GPL restera tout le temps sous GPL [1]. Le lecteur intéressé pourra notamment se référer aux Fiches Libres de l'ALDIL, disponibles sur [AFL].

De même, on a souvent tendance à associer logiciel libre à gratuité, alors qu'un logiciel libre peut tout à fait être payant. Par exemple, Richard STALLMAN facturait 150 dollars la diffusion de son éditeur de texte Emacs. Rien ne s'oppose donc à ce qu'une société fasse payer le fruit de ses efforts de développement à un client, même dans le cas d'un logiciel libre. D'ailleurs, de nombreuses sociétés basent leur modèle commercial autour du logiciel libre, en mettant l'accent sur les services apportés : expérience, adaptation du logiciel aux besoins des clients, support, etc. On peut notamment citer Red Hat et Mandriva, qui éditent une distribution Linux libre et facturent le service associé.

En plus des sociétés qui basent leur *business model* sur le libre, on trouve aussi des entreprises qui soutiennent le développement de logiciels libres, par exemple IBM et Google paient des développeurs pour travailler sur des projets libres.

Des exemples de logiciels libres connus et réputés sont :
– le navigateur Internet Mozilla Firefox ;

1. En fait, le ou les auteurs du logiciel peuvent décider de diffuser leur code sous une autre licence, voire une double licence, mais le sujet des licences est trop vaste pour être abordé en détail ici.

– le système d'exploitation GNU/Linux (c'est à dire le noyau Linux et tous les programmes venant avec, issus du projet GNU) ;
– le logiciel de messagerie Mozilla Thunderbird ;
– le serveur web du projet Apache (qui représente plus de 60 % des serveurs web sur Internet)...

Par opposition, on appelle un logiciel propriétaire un logiciel qui ne fournit pas toutes ces libertés à l'utilisateur, la licence n'étant souvent qu'un simple droit d'*utilisation* concédé. On peut citer comme exemple de logiciel propriétaire le système d'exploitation Microsoft Windows (toutes versions confondues) ou encore la suite bureautique Microsoft Office.

Les logiciels libres ont su conquérir une grande communauté d'utilisateurs de par le monde, amenant toujours plus de contributeurs aux projets, améliorant ainsi la qualité des logiciels. Les avantages des logiciels libres sur les logiciels propriétaires sont nombreux, tant sur le plan technique (disponibilité du code source, modifications possibles) que sur le plan idéologique (ouverture à la communauté, partage des connaissances, non-enfermement des utilisateurs), c'est pourquoi ce livre blanc se concentre uniquement sur les solutions de virtualisation libres. Néanmoins, dans un souci d'exhaustivité, les solutions propriétaires majeures seront mentionnées.

1.2 La virtualisation — définitions

La virtualisation a été brièvement définie dans l'introduction comme le moyen de faire fonctionner sur une seule machine physique plusieurs systèmes d'exploitation ou plusieurs applications [WFv; WAv]. Cet objectif est atteint grâce à plusieurs technologies ayant des buts différents. Il est important de bien définir toutes ces technologies avant d'étudier les projets retenus.

Tout d'abord, il existe plusieurs catégories de virtualisation, utilisant chacune des technologies différentes. Les technologies les plus répandues sont :
– la virtualisation complète ;
– la paravirtualisation ;
– la virtualisation assistée par le matériel ;
– le cloisonnement.

Chacune de ces technologies est une technologie de virtualisation, mais elles ne fonctionnent pas de la même façon. Les principes et particularités de chaque technologie seront détaillés dans les pages suivantes.

Toutefois, avant de définir ces technologies, il est crucial de bien saisir le fonctionnement et le rôle d'un système d'exploitation. En effet, comme l'objectif de la plupart des technologies de virtualisation est de faire cohabiter plusieurs systèmes d'exploitation sur la même machine physique, il faut auparavant expliquer pourquoi il est nécessaire d'utiliser une solution de virtualisation pour ce faire. C'est pourquoi comprendre comment fonctionne un système d'exploitation et pourquoi il est nécessaire, même superficiellement, permettra de bien saisir les choix techniques et les problèmes rencontrés par la virtualisation. La section suivante sera donc consacrée à expliquer — de manière très succincte — le rôle d'un système d'exploitation et du matériel courant d'un ordinateur.

1.2.1 Fonctionnement d'un système d'exploitation

Le système d'exploitation *(operating system)* est un ensemble complexe faisant office de couche d'abstraction entre le matériel (*hardware*, niveau physique) et le logiciel (*software*, niveau logique). Il est composé d'une multitude de composants, chacun assigné à un rôle spécifique. Parmi les tâches dévolues au système d'exploitation, on retrouve notamment la gestion de la mémoire vive (RAM) et des périphériques (stockage, carte réseau, imprimante, écran, clavier, etc.). La gestion de la RAM est l'une des tâches les plus complexes du système d'exploitation. En effet, tous les programmes (que ce soit au niveau de l'utilisateur ou du système d'exploitation) ont besoin de mémoire pour fonctionner. C'est dans la RAM que seront stockés :
– le code des programmes en cours d'exécution ;
– les données des programmes en cours d'exécution ;
– le code du système d'exploitation ;
– les données du système d'exploitation.

Un composant du système d'exploitation est entièrement dédié à la gestion de la mémoire, notamment la réservation et la libération pour les applications et le système d'exploitation. Quand la RAM vient à manquer, le système peut utiliser une partie du disque dur comme extension de mémoire. Le disque dur est toutefois de plusieurs ordres de

grandeur plus lent que la RAM, aussi le gestionnaire de mémoire fait tout son possible pour en limiter l'usage.

Toutefois, le système d'exploitation n'a pas le contrôle direct sur la gestion de la mémoire au niveau physique. Ce rôle est dévolu au processeur. C'est donc par le jeu d'une interaction complexe entre le système d'exploitation (qui gère la RAM au niveau logique) et le processeur (qui gère la RAM au niveau physique) que se déroule l'exécution d'un programme.

Le programme est interprété par le processeur, et est composé d'une suite d'opérations élémentaires nommées instructions. Ces instructions consistent principalement en des demandes d'accès à la RAM, des opérations mathématiques et des appels spécifiques au matériel (carte graphique, carte réseau, clavier, écran, disque dur, etc.). Cette suite d'instructions élémentaires exécutées dans l'ordre donne au final le programme, qui peut être un programme du système d'exploitation ou un programme utilisateur.

Le système d'exploitation a aussi pour rôle de faire abstraction du matériel pour les programmes utilisateurs. Ainsi, un programme doit se comporter de la même manière quel que soit le modèle de carte réseau utilisé pour communiquer ou la marque et le type de disque dur contenant les fichiers. Cette abstraction est réalisée par les pilotes de périphériques. Ces pilotes sont en général destinés à un type de matériel particulier et offrent au système d'exploitation un ensemble cohérent d'opérations. Par exemple, tous les pilotes de disque dur permettent de lire le contenu du disque à un endroit donné et tous les pilotes de carte réseau permettent d'envoyer et de recevoir des données. Le détail est caché au système d'exploitation — et donc à l'utilisateur — car seul compte l'ensemble cohérent d'opérations. Le système d'exploitation se repose sur les pilotes de périphérique pour apporter des couches d'abstraction supplémentaires, accessibles aux programmes utilisateurs, par exemple pour la gestion des fichiers, des protocoles réseau... Les programmes utilisateurs ne peuvent accéder au matériel qu'à travers les couches d'abstraction, assurant ainsi la cohérence du système.

Le système d'exploitation doit, pour assurer cette abstraction, avoir un accès exclusif au matériel afin de le contrôler. De fait, presque tous les systèmes d'exploitation sont conçus comme s'ils étaient les seuls à accéder au matériel. Cette notion d'exclusivité est importante pour les solutions de virtualisation : le système virtualisé ne pourra pas accéder au matériel directement, comme s'il était le seul, car c'est le système hôte qui a

ce rôle. Il y a donc des solutions de contournement mises en place, qui varient selon les produits et les technologies utilisées.

La séparation en couches du système d'exploitation fait qu'une grande partie du code est indépendante du matériel. Par exemple, le système Linux est porté sur une vingtaine d'architectures et sous-architectures, mais seule une faible partie des millions de lignes de code le composant est spécifique à une architecture. Cette indépendance des différents composants et couches du système d'exploitation facilitent la tâche des solutions de virtualisation. En effet, quelle que soit la technologie utilisée, elle n'entrera en contact qu'avec une faible partie du système d'exploitation, ce qui permet de concentrer les efforts de développement sur ces portions spécifiques.

1.2.2 La virtualisation complète

La virtualisation complète *(full virtualization)*, dénommée ainsi par opposition à la paravirtualisation définie page 16, consiste à émuler l'intégralité d'une machine physique pour le système invité. Le système invité « croit » s'exécuter sur une véritable machine physique. Le logiciel chargé d'émuler cette machine s'appelle une machine virtuelle, son rôle est de transformer les instructions du système invité en instructions pour le système hôte. En effet, comme le montre la figure 1.1 page suivante, la machine virtuelle est un programme comme un autre du point de vue du système hôte, au même titre qu'un navigateur Internet ou un traitement de texte. Or, comme expliqué précédemment, un système d'exploitation doit normalement manipuler le matériel à un niveau très bas. Les programmes utilisateurs n'ont pas d'accès direct au matériel, mais uniquement aux couches d'abstraction. La machine virtuelle émule donc de manière logique (c'est à dire avec du code) tout le matériel habituel de l'architecture de l'ordinateur cible. Sur la figure 1.1 page suivante, le rectangle en fond vert est le système d'exploitation, seule partie à avoir un accès direct au matériel, ici représenté avec un fond bleu. Le rectangle en fond blanc est une application utilisateur, qui ne peut utiliser que la couche d'abstraction du système d'exploitation pour accéder indirectement au matériel.

En pratique, le disque dur de la machine virtuelle est la plupart du temps géré comme un (volumineux) fichier pour le système hôte, alors que la mémoire vive dont le système invité dispose est réservée par le programme de la machine virtuelle. Le reste de l'architecture de l'ordinateur peut varier grandement selon les implémentations, mais

on retrouve généralement au moins une carte réseau bas de gamme, un clavier 105 touches « standard » et une carte graphique bas de gamme.

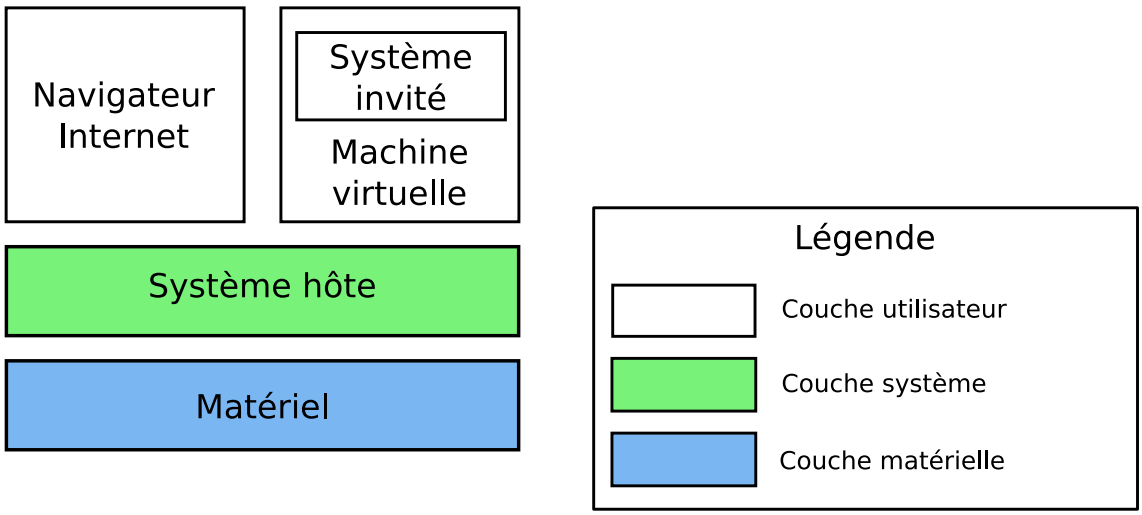

FIG. 1.1 – Virtualisation complète

L'utilisation de périphériques bas de gamme s'explique par le fait qu'il y a toujours un nombre minimal d'opérations supportées par toute catégorie de matériel sur un ordinateur : la vitesse de transfert la plus lente pour un disque dur, la résolution d'affichage la plus faible pour une carte graphique, etc. Or comme le comportement de ces périphériques est entièrement implémenté de manière logicielle par la machine virtuelle, émuler un périphérique avec le minimum de fonctionnalités permet de limiter la quantité de code à développer pour en couvrir le comportement. C'est la raison pour laquelle les cartes graphiques et les cartes réseaux sont la plupart du temps aux standards en vigueur dans les années quatre-vingt.

Ce n'est toutefois pas la seule raison de l'émulation de périphériques bas de gamme. En effet, la plupart des évolutions ultérieures du matériel visent à améliorer les performances des périphériques, par exemple en augmentant le débit du disque dur ou la résolution supportée par la carte graphique. Cependant, les optimisations de performances sont dans ce cas sans objet, car elles ne se répercutent de toute manière pas sur un matériel physique en mesure de les supporter. La rapidité du disque dur virtuel est par exemple limitée par la vitesse d'accès au fichier le représentant sur le système hôte.

Le système s'exécutant dans la machine virtuelle est un système d'exploitation à part entière, tel qu'on pourrait en installer sur une machine physique : Microsoft Windows, GNU/Linux, Mac OS X, etc. Cette particularité est la caractéristique principale de la virtualisation complète : les systèmes invités n'ont pas à être modifiés pour être utilisés dans une machine virtuelle utilisant une technologie de virtualisation. Dans la pratique, c'est le cas pour les systèmes d'exploitation et les machines virtuelles les plus répandus.

Le système invité peut à son tour exécuter n'importe quel programme prévu pour ce système, dans la mesure où il ne nécessite pas de matériel non fourni par la machine virtuelle (*i.e.* pas de carte graphique dernière génération ou de périphérique peu courant). Cette possibilité est due au fait que le système d'exploitation sert (entre autres) de couche d'abstraction entre le matériel et les applications, donc à partir du moment où le système fonctionne correctement, les applications s'exécutant par dessus fonctionneront aussi.

La traduction au vol des instructions du système invité est néanmoins une opération complexe et coûteuse en temps. En effet, la machine virtuelle ne peut pas, dans la plupart des cas, exécuter directement les instructions du système invité sur le système hôte. Les instructions de manipulation de la RAM, par exemple, doivent être interprétées par la machine virtuelle pour aboutir au résultat attendu, car c'est le processeur de la machine virtuelle qui est censé s'occuper de la gestion physique de la mémoire, et non le processeur de la machine hôte.

La machine virtuelle doit donc implémenter en logiciel une gestion complète de la mémoire de l'invité, en utilisant les couches d'abstraction de l'hôte pour accéder à la RAM. Cette empilement de couches réduit significativement les performances, surtout en cas de forte pression sur la mémoire (*i.e.* quand la mémoire est utilisée de manière intensive : lecture, écriture, déplacement de données, etc.). La figure 1.2 page suivante détaille les couches d'abstraction entrant en jeu pour la gestion de la mémoire.

Dans cette figure, le cadre en bleu foncé représente la couche matérielle ; le cadre vert est le système d'exploitation, qui a un accès privilégié au matériel. Le cadre en fond blanc est une application utilisateur, qui doit utiliser les couches d'abstraction du système d'exploitation — représentées en vert foncé — pour accéder au matériel. Le cadre bleu ciel représente le matériel émulé par la machine virtuelle, qui doit se comporter

comme le matériel réel d'un ordinateur pour le système invité.

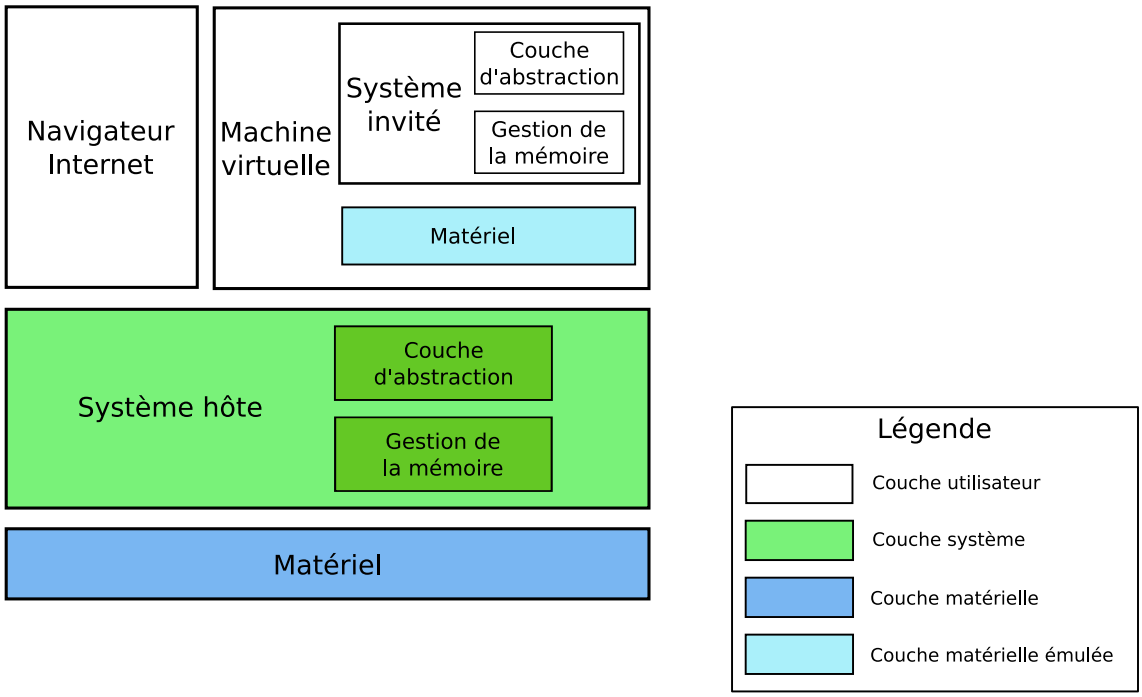

FIG. 1.2 – Couches d'abstraction pour la gestion de la mémoire

Cet empilage de couches est sensiblement identique pour tous les périphériques émulés par la machine virtuelle. On retrouve, du plus bas niveau au plus haut niveau :

1. Le matériel ;
2. Le pilote du matériel pour le système hôte ;
3. La couche d'abstraction du système hôte ;
4. Le matériel émulé par la machine virtuelle ;
5. Le pilote du matériel pour le système invité ;
6. La couche d'abstraction du système invité.

Les performances de la machine virtuelle sont donc limitées par les performances de la couche d'abstraction du système hôte et par la qualité de l'émulation du matériel implémenté.

La séparation nette entre la machine virtuelle et le système hôte est un avantage certain pour la sécurité et la stabilité de la machine. En effet, comme la machine virtuelle est un simple programme, on peut très facilement limiter la quantité de mémoire qu'elle peut

allouer, le temps processeur consommé, sa priorité par rapport aux autres programmes, etc. Toutes les possibilités d'administration et de configuration des applications offertes par le système hôte s'appliquent à la machine virtuelle. Cela permet par exemple d'attribuer une faible priorité à une machine virtuelle mais de lui permettre de réserver plus de mémoire, alors qu'une autre machine virtuelle aura plus de temps processeur à disposition, mais moins de RAM. Bien évidemment, comme les machines virtuelles sont de simples programmes utilisateurs, on peut en exécuter plusieurs à la fois sur le même système hôte, tant que les ressources sont disponibles. Ainsi, une machine multiprocesseur et disposant de suffisamment de mémoire vive et d'espace de stockage peut aisément accueillir plusieurs machines virtuelles, le système hôte répartissant au mieux la charge entre les différents processeurs.

Cependant, du fait de l'empilement de couches d'abstraction et de l'impossibilité pour la machine virtuelle d'accéder directement au matériel, les performances du système invité sont assez éloignées de celles d'un système « natif ». Selon les implémentations, diverses solutions sont utilisées pour accélérer les machines virtuelles, par exemple en passant la plupart des instructions destinées au processeur virtuel directement au processeur physique. Cela accélère la vitesse de calcul du système invité. Il reste cependant le problème des Entrées/Sorties (E/S), c'est à dire les accès au disque, à la RAM, à la carte graphique, à la carte réseau, etc. D'une manière générale, on appelle Entrées/Sorties (*I/O* ou *Input/Output*) tout ce qui consiste à transférer des informations ou des données entre un périphérique et le système d'exploitation. Les E/S sont beaucoup plus dures à optimiser, car chaque système d'exploitation a une façon propre de gérer cela. Il faut donc cohabiter étroitement à la fois avec le système hôte pour l'accès réel au matériel et avec le système invité pour que ses accès au matériel soient le plus rapide possible. Cela amène une plus grande complexité de code, et une séparation en couches moins marquée que dans le modèle vu sur la figure 1.1 page 13. Cette « rupture » dans le modèle en couches est exploitée par une autre technologie de virtualisation : la paravirtualisation.

1.2.3 La paravirtualisation

La paravirtualisation (*paravirtualization* ou encore *para-virtualization*) est très proche du concept de la virtualisation complète, dans le sens où c'est toujours un système

d'exploitation complet qui s'exécute sur le matériel émulé par une machine virtuelle, cette dernière s'exécutant au dessus d'un système hôte. Toutefois, dans une solution de paravirtualisation, le système invité est modifié pour être exécuté par la machine virtuelle. Les modifications effectuées visent à rendre le système émulé « au courant » du fait qu'il s'exécute dans une machine virtuelle. De ce fait, il pourra collaborer plus étroitement avec le système hôte, en utilisant une interface spécifique, au lieu d'accéder au matériel virtuel via les couches d'abstraction. Au final, l'architecture obtenue est plus performante que l'empilement de couches d'abstraction de la figure 1.2.

Le terme *para-virtualization* a été mentionné pour la première fois dans [WSG02], où les auteurs définissent la paravirtualisation comme la modification sélective de certaines parties de l'architecture virtuelle pour améliorer les performances, la réactivité sous forte charge et la simplicité de conception. L'idée de la paravirtualisation est toutefois plus ancienne que cela. Les premiers gros systèmes utilisant une architecture de virtualisation avaient déjà une technologie similaire, dès les années soixante-dix, même si elle n'avait pas de nom.

En pratique, un système paravirtualisé possède quelques pilotes de périphériques et sous-systèmes modifiés, qui lui permettent de communiquer directement avec la machine virtuelle, sans avoir passer par une couche d'abstraction pour parler au matériel virtuel. Les pilotes paravirtualisés échangent directement des données avec la machine virtuelle, sans avoir à passer par une émulation du comportement du matériel. Les parties du système hôte généralement modifiées pour tirer profit de la paravirtualisation sont la gestion de la mémoire et la gestion des E/S. En effet, ce sont véritablement les deux goulets d'étranglement d'un système virtualisé, du fait du nombre de couches d'abstraction à traverser. Il est donc logique que les optimisations se portent là dessus.

La figure 1.3 page suivante montre la structure d'une machine virtuelle et d'un système hôte supportant la paravirtualisation. Les pilotes non modifiés interagissent toujours avec le matériel émulé par la machine virtuelle (rectangle bleu ciel), alors que les pilotes modifiés communiquent directement les fonctions de la machine virtuelle (rectangle jaune). La simplification qui en résulte permet au système invité de collaborer plus efficacement avec l'hôte : les parties critiques du système communiquent presque directement avec le système hôte, en contournant les couches d'abstraction virtuelles (*i.e.* le matériel émulé). Le reste de l'architecture est inchangé, la machine virtuelle est toujours une application utilisateur (rectangle blanc) et le système d'exploitation

(rectangle vert) est toujours le seul à avoir un accès privilégié au matériel (rectangle bleu).

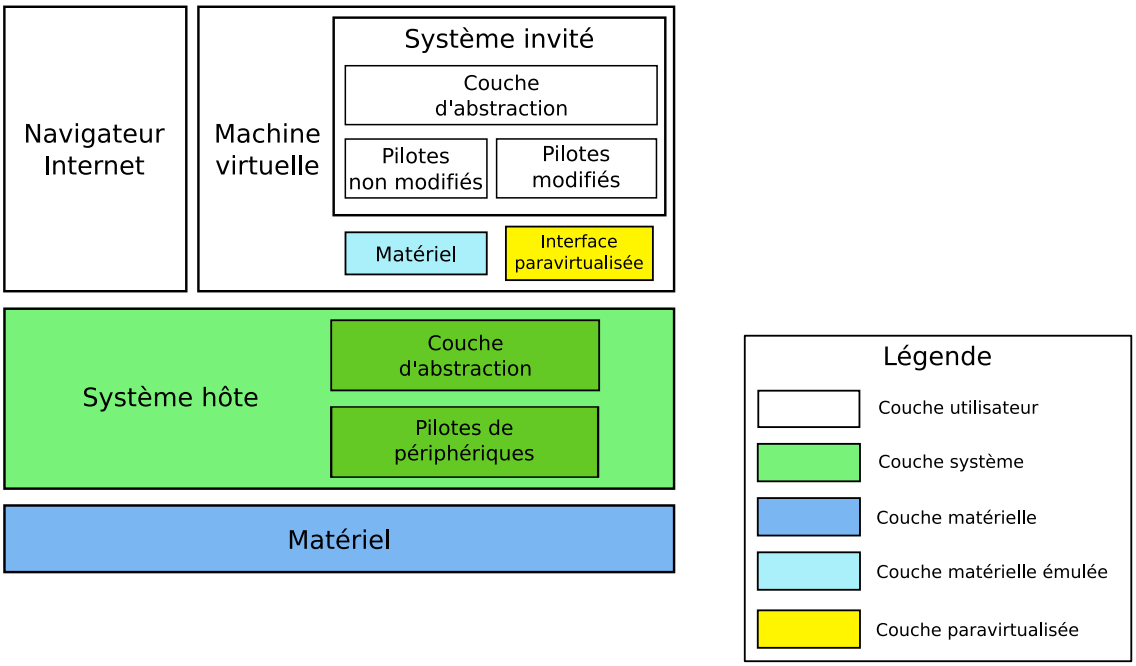

FIG. 1.3 – Paravirtualisation

Les détails sur comment sont réalisées ces optimisations varient selon les implémentations, mais il s'agit en général pour le système invité d'utiliser des appels systèmes ou des instructions spécifiques pour renseigner la machine virtuelle sur les actions à entreprendre. Cette dernière réalise alors ces actions, et communique le résultat au système invité. Le type d'actions à effectuer varie également selon les implémentations, mais on retrouve en général tout ce qui est déplacement de données entre l'hôte et l'invité (accès disque, transfert réseau, etc.) et gestion de la mémoire.

La paravirtualisation apporte un gain de performances avéré, du fait du contournement des couches d'abstraction. En effet, comme le système invité collabore activement avec la machine virtuelle, il ne se comporte plus comme un système d'exploitation à part entière s'exécutant directement sur du matériel. Au contraire, il adapte son comportement pour que les accès au matériel — souvent difficiles à interpréter de manière efficace par la machine virtuelle — soient transformés en des appels directs à cette dernière. De plus, étant donné que seules les couches de bas niveau du système invité ont été modifiées, toutes les applications qui pouvaient fonctionner dans une architecture de virtualisation complète peuvent aussi être utilisées dans une architecture paravirtualisée.

Toutefois, cette augmentation des performances est restreinte à certains systèmes. En effet, comme le système invité doit être modifié [2] pour être paravirtualisé, il faut bien évidemment que l'on ait la possibilité de réaliser cette opération de portage. Or, cela nécessite à la fois l'accès au code source du système d'exploitation et la permission du détenteur des droits de le modifier. Si cela ne pose aucun problème pour un système libre (notamment GNU/Linux et les systèmes BSD), il n'en va pas de même pour les systèmes propriétaires, tels que Microsoft Windows et Mac OS. L'usage de la paravirtualisation est donc généralement limité aux systèmes libres, sauf à utiliser une solution de virtualisation propriétaire compatible avec un seul système d'exploitation invité, comme les produits que Microsoft propose pour ses systèmes d'exploitation.

Tout comme la virtualisation complète, la paravirtualisation garde une séparation nette entre le système invité et le système hôte (cf. figures 1.1 et 1.3). De ce fait, seul le système hôte a un accès direct et exclusif au matériel. Le système invité doit donc toujours passer par la machine virtuelle pour accéder au matériel, qui passe à son tour par la couche d'abstraction. On peut donc améliorer davantage le processus en laissant au système invité un accès direct — mais contrôlé — au matériel. C'est le but des systèmes à hyperviseur.

1.2.4 Les systèmes à hyperviseur

L'utilisation d'un hyperviseur *(hypervisor)* est en quelque sorte l'évolution logique de la paravirtualisation, si l'on recherche encore une amélioration des performances. Dans les technologies précédentes, le système hôte était le seul à avoir un accès direct au matériel ; avec un hyperviseur, le système hôte partage cet accès avec les systèmes invités. Au démarrage de l'ordinateur, c'est normalement le système d'exploitation qui prend la main et contrôle le matériel. Dans le cas de l'utilisation d'un hyperviseur, c'est un système minimaliste — l'hyperviseur — qui prend le contrôle du matériel. Ensuite, il fait appel à un système d'exploitation complet, qui sera donc exécuté par dessus l'hyperviseur. Ainsi, le système d'exploitation doit passer par l'hyperviseur pour tout accès au matériel. On peut donc très facilement instancier un deuxième système d'exploitation, qui passera lui aussi par l'hyperviseur pour l'accès au matériel. Comme les systèmes d'exploitation doivent obligatoirement passer par ce dernier pour tout accès au ma-

2. On parle de portage, de la même manière qu'on porte un système ou une application vers une nouvelle architecture matérielle

tériel, l'hyperviseur peut s'assurer qu'ils n'accèdent qu'aux ressources autorisées, sans perturber le fonctionnement des autres systèmes.

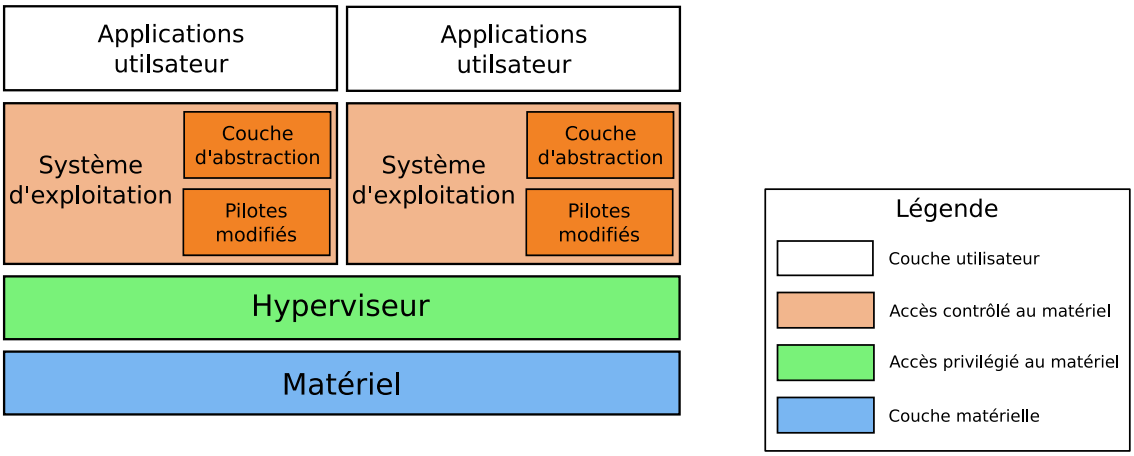

FIG. 1.4 – Hyperviseur

La figure 1.4 détaille le principe de fonctionnement de l'hyperviseur. À la différence des technologies exposées précédemment, il n'y a cette fois pas d'accès direct au matériel (rectangle bleu) pour le système d'exploitation, uniquement une couche d'abstraction minimale fournie par l'hyperviseur (rectangle vert). L'hyperviseur est le seul à avoir un accès privilégié au matériel. Dans cette représentation, les systèmes cohabitent au même niveau de privilège, uniquement régulés par l'hyperviseur. Toutefois, selon les implémentations, il y a souvent un système privilégié, qui est en général le premier système démarré par l'hyperviseur. Ce système est alors autorisé à modifier les paramètres de l'hyperviseur ou à instancier de nouveaux systèmes invités. À l'opposé, sur d'autres implémentations, la différence entre hôte et invité est inexistante, tous les systèmes ont les mêmes privilèges et l'hyperviseur est alors contrôlé d'une autre manière.

Si les deux technologies vues précédemment (virtualisation complète et paravirtualisation) utilisaient une machine virtuelle pour émuler le matériel, il n'en va pas de même avec un hyperviseur. Chaque système d'exploitation a un accès presque direct au matériel, par l'intermédiaire de l'hyperviseur. Il n'y a donc plus de couche d'abstraction logicielle, le matériel accessible est celui de la machine physique, avec toutes les fonctionnalités qu'il peut offrir. Le gain de performances est parfois significatif, notamment dans le cas des E/S, où le système peut utiliser toutes les extensions des périphériques pour accélérer les transferts.

Ce n'est toutefois pas la seule différence avec les autres technologies de virtualisation : avec un hyperviseur, le contrôle de l'accès au matériel et de l'utilisation des ressources est bien plus fin. En effet, dans les solutions à base de machine virtuelle, le système invité est vu comme un processus par le système hôte, le niveau de contrôle et de suivi est donc celui offert par le système hôte. Ici, c'est l'hyperviseur qui est chargé d'appliquer la politique d'accès aux ressources matérielles. Cela implique un contrôle directement au niveau de l'accès aux ressources. Il peut par exemple limiter la consommation de temps processeur d'un système, ou la quantité de RAM attribuée. La granularité de la configuration n'est pas la même qu'avec une architecture à base de machines virtuelles : au niveau de l'hyperviseur, on contrôlera donc qu'un système ne dérange pas les autres en consommant trop de ressources, alors que dans un système d'exploitation, on s'assurera qu'un programme utilisateur ne dérange pas les autres programmes du système.

Au final, le contrôle des priorités est plus précis, et permet de garantir qu'un système invité isolé n'influera jamais sur l'accès aux ressources d'un autre système. Avec une architecture à base de virtualisation complète, si le système hôte est monopolisé par un processus utilisateur (par exemple un programme consommant énormément de mémoire), les systèmes invités peuvent se voir fortement ralentis dans leur activité.

Tous les systèmes destinés à s'exécuter au dessus d'un hyperviseur doivent être portés, comme les systèmes invités pour la paravirtualisation. Cette opération vise à adapter les couches bas niveau du système d'exploitation pour qu'elles communiquent avec l'hyperviseur plutôt qu'avec le matériel. Les inconvénients sont donc les mêmes que pour la paravirtualisation : il est nécessaire d'avoir accès au code source de tous les systèmes ainsi que l'autorisation du détenteur des droits. En outre, le portage est beaucoup plus lourd à réaliser pour fonctionner sur un hyperviseur. En effet, pour la paravirtualisation, seuls quelques pilotes et sous-systèmes avaient besoin d'être réécrits pour tirer parti de l'accélération. Au contraire, un hyperviseur nécessite la modification de *toutes* les couches d'accès au matériel ; la complexité du code s'en trouve grandement augmentée, augmentant par là même la difficulté de maintenir le code. Le portage sur un hyperviseur revient quasiment à porter le système d'exploitation sur une nouvelle architecture matérielle.

Les techniques vues jusqu'à présent étaient de complexité croissante, c'est à dire que chaque technologie était un peu plus complexe à mettre en œuvre et à que la précédente. Avec les technologies ayant trait au cloisonnement, c'est différent. En effet, le

cloisonnement était au départ utilisé pour sécuriser et isoler des applications, sans rapport avec le fait d'isoler des systèmes d'exploitation, c'est seulement récemment que l'idée d'utiliser ces techniques pour la virtualisation a vu le jour.

1.2.5 Les techniques de cloisonnement

Une autre pratique répandue dans le domaine de la virtualisation est le cloisonnement. Derrière ce nom se cachent plusieurs technologies visant à séparer fortement les processus s'exécutant sur un même système d'exploitation. Le cloisonnement vise à isoler chaque processus dans un conteneur dont il est théoriquement impossible de sortir. Un processus isolé de la sorte ne saura pas quels autres processus s'exécutent sur le même système, et n'aura qu'une vision limitée de son environnement. Le but principal de cette technologie est d'améliorer la sécurité du système d'exploitation et des applications.

Selon les implémentations, cela peut aller du simple « emprisonnement » dans un environnement volontairement minimal à une image complète du système accessible uniquement au processus isolé. La plupart des systèmes d'exploitation basés sur UNIX proposent un moyen d'isoler les processus. Le plus répandu (et le plus ancien) est la commande *chroot*, qui permet de créer un environnement minimal contenant uniquement ce qui est strictement nécessaire pour exécuter le programme isolé. Les systèmes basés sur BSD proposent aussi *jail* (prison), qui est une évolution de *chroot*, plus sûre, plus complète et plus souple dans son utilisation. L'UNIX de Sun Microsystems, Solaris, propose un système de *zones* très évolué, plus proche d'une machine virtuelle que d'une simple isolation des processus. Toutes ces technologies ont en commun le fait de conserver exactement la même instance du système d'exploitation accessible aux processus isolés. Ainsi, on ne pourra pas avec le cloisonnement proposer un système d'exploitation différent pour un processus isolé. Par exemple, si le système hôte est Solaris, alors tous les processus s'exécutant à l'intérieur d'une *zone* auront accès à la même version de ce Solaris.

Les technologies de cloisonnement sont aussi utilisées dans d'autres domaines que les systèmes d'exploitation. Par exemple, le langage Java (de Sun Microsystems) propose une machine virtuelle (qui n'a rien à voir avec les machines virtuelles étudiées ici) qui est un interpréteur pour le langage Java. Cet interpréteur exécute tous les programmes Java dans un conteneur isolé, dont ils ne peuvent pas sortir. Le terme utilisé pour décrire

cette technologie est *sandbox* (bac à sable). Le *sandboxing* est aussi une technologie de cloisonnement, mais au niveau d'un processus, sans que le système intervienne.

Même si l'engouement pour la virtualisation est assez récent, les technologies de virtualisation et l'idée même de la virtualisation sont presque aussi anciennes que l'informatique. La section suivante fera un bref historique de la virtualisation.

1.3 Historique

Les premiers ordinateurs, qui occupaient plusieurs pièces d'un bâtiment, n'étaient pas faits pour exécuter *plusieurs* programmes à la fois. On concevait un programme (qui était à l'époque une simple succession de calculs), on le mettait dans la file d'attente des programmes, et quand le système d'exploitation avait fini de traiter un programme, on lui donnait le suivant dans la liste.

1.3.1 Premiers pas

Très vite, dès la fin des années cinquante, l'idée de pouvoir exécuter plusieurs programmes en parallèle voit le jour. On parle de temps partagé *(time sharing)*, de multiprogrammation, etc. L'idée était de pouvoir faire cohabiter plusieurs programmes au même moment, ayant tous accès au même matériel, sans qu'ils ne se gênent mutuellement. La virtualisation est très proche de concept.

1.3.2 Machines virtuelles

Au milieu des années soixante, IBM effectue des recherches sur les systèmes virtualisés avec le projet M44/44X. L'architecture du système se basait sur des systèmes d'exploitation virtualisés (nommés 44X) s'exécutant au dessus du matériel (une machine M44). Les systèmes invités étaient gérés par une simple multiprogrammation. En 1967 est lancé, toujours par IBM, le système CP-40, le premier système offrant une virtualisation complète. Le CP-40 sera suivi par plusieurs évolutions, amenant chacune de nouvelles

fonctionnalités pour les utilisateurs. On peut notamment citer le système VM/370, qui a connu un très fort succès dans les entreprises, et est parfois encore en usage dans certaines entreprises aujourd'hui.

1.3.3 Amélioration des technologies

Après le succès des machines virtuelles introduites par IBM, les technologies ont assez peu évolué. Le système hôte a vite été réduit à l'état de simple arbitre entre les systèmes invités, amenant la notion d'hyperviseur.

Toutefois, toutes ces technologies de virtualisation étaient réservées au monde professionnel, destinées à être utilisées sur des *mainframes** coûtant plusieurs millions de dollars.

Parallèlement à cela, le monde de la recherche (souvent financé par ces mêmes entreprises) a continué à étudier différentes possibilités pour améliorer les performances et à essayer de nouvelles technologies. La plupart de ces travaux de recherche sont toutefois restés assez confidentiels et n'ont que rarement été transposés sur un produit.

1.3.4 Intérêt du grand public

L'orientation « grand public » des technologies de virtualisation est beaucoup plus récente. Dans les années quatre-vingt-dix, l'intérêt pour les émulateurs de consoles de jeu ainsi que l'explosion du marché de l'informatique personnelle (les ordinateurs de type PC) ont fait prendre conscience aux entreprises qu'il y avait un marché pour la virtualisation sur PC. Des sociétés ont alors commencé à créer des produits de virtualisation basés sur des machines virtuelles pour les « petites » entreprises — c'est à dire celles ne pouvant s'offrir des serveurs à plusieurs millions de dollars — et pour les particuliers.

À partir de ce moment là, les technologies ont vraiment progressé, avec l'arrivée de nouveaux acteurs toujours prêts à innover pour se démarquer des concurrents.

1.4 Acteurs majeurs

Comme vu précédemment, les acteurs principaux dans le domaine de la virtualisation sont partagés entre les très grandes entreprises fournissant des solutions pour leurs ordinateurs d'un côté et les entreprises fournissant des solutions orientées PC de l'autre. Les deux types d'activité ne sont que très rarement en concurrence, car les buts visés ne sont pas les mêmes.

Les grandes entreprises sont celles qui fournissent des machines pour les centres de calcul et les *mainframes*. Les acteurs les plus connus sont IBM — qui a un fort historique de virtualisation —, HP, Sun, Bull, etc. Toutes ces sociétés fournissent des systèmes de virtualisation fonctionnant exclusivement avec leur propre architecture matérielle. Les technologies utilisées diffèrent selon les systèmes, mais en général ce sont des technologies à base de virtualisation complète ou d'hyperviseur.

Sur les architectures de type PC, il y a plus de sociétés proposant des produits et donc plus d'offres de virtualisation. On peut notamment citer Microsoft, qui a racheté la solution de virtualisation de Connectix en février 2003. Ce rachat a ensuite donné lieu à la diffusion de Virtual PC et Virtual Server, produits permettant de virtualiser des systèmes à base de Windows, respectivement pour un ordinateur personnel et pour un serveur. La version pour serveurs offre également la possibilité de virtualiser GNU/Linux.

La société VMware édite plusieurs produits à destination des entreprises souhaitant virtualiser leurs serveurs, qui couvrent sensiblement les mêmes applications que les solutions de Microsoft, mais avec en plus la possibilité de faire fonctionner leurs produits avec le système GNU/Linux en tant que système hôte.

Ces deux sociétés fournissent des solutions propriétaires destinées aux particuliers et aux entreprises. Les technologies utilisées sont soit de la virtualisation complète soit de la paravirtualisation, en fonction des produits et systèmes.

Du côté de la communauté du logiciel libre, il y a énormément de projets de virtualisation, ayant des buts variables. Certains d'entre eux sont soutenus par une société, qui fournit un service payant pour les clients le souhaitant. Les plus connus sont :

Bochs (prononcer « box ») Bochs est un émulateur très complet de l'architecture PC traditionnelle (processeur Intel) ;

KVM soutenu par la société Qumranet, KVM se veut une solution de virtualisation performante et facile à administrer ;

Linux-VServer projet de virtualisation destiné à fonctionner sur le système d'exploitation GNU/Linux ;

OpenVZ soutenu par la société Virtuozzo, OpenVZ est une solution de virtualisation visant à obtenir les meilleures performances possibles ;

QEMU créé et développé par Fabrice BELLARD, QEMU est un projet visant à offrir une machine virtuelle complète pour plusieurs architectures matérielles, y compris l'architecture PC ;

Xen soutenu par la société XenSource, Xen vise à proposer un solution performante pour la virtualisation de serveurs ;

Ces projets sont tous à des états d'avancement différents, certains sont d'ores et déjà utilisables en production alors que d'autres sont encore en phase de développement.

Avec l'attrait récent des entreprises et du grand public pour la virtualisation, il n'est guère étonnant de constater qu'il y a eu plusieurs avancées récentes dans les technologies utilisées.

1.5 Évolutions récentes

Que ce soit dans la communauté du logiciel libre ou dans le domaine du logiciel propriétaire, toutes les solutions de virtualisation sur le marché ont fait des progrès considérables ces dernières années.

Du côté du logiciel propriétaire, l'arrivée du géant Microsoft comme éditeur de solutions de virtualisation a relancé la course au développement de technologies performantes. En effet, la plupart des concurrents de la société Microsoft ont peur qu'il ne réitère ce qu'elle avait fait pour Netscape Navigator, alors en concurrence avec son logiciel Internet Explorer : diffuser en masse, avec son système d'exploitation, un produit techniquement inférieur dans l'espoir d'affaiblir, voire d'éliminer, leurs concurrents. Cette stratégie lui avait plutôt réussi par le passé, aussi les craintes des éditeurs de produits de virtualisation pour la plate-forme Windows sont légitimes. Microsoft a également annoncé que la

prochaine version de son système d'exploitation pour serveur, pour l'instant encore dénommée *Longhorn*, pourra fonctionner avec un hyperviseur, développé en collaboration avec la société XenSource.

En parallèle, la communauté du logiciel libre continue elle aussi à faire évoluer les différents projets. Récemment, une architecture visant à faciliter la paravirtualisation, nommée `paravirt_ops`, a été intégrée au noyau Linux. Elle permet à toutes les solutions de virtualisation de faire fonctionner un système GNU/Linux avec paravirtualisation sans avoir à modifier le code du noyau, mais simplement en utilisant `paravirt_ops` pour exploiter les appels paravirtualisés. Il est à noter que les développeurs de Xen et de VMware ont collaboré étroitement avec les développeurs de Linux pour l'élaboration de cette fonctionnalité.

De même, une interface pour la virtualisation des Entrées/Sorties, nommée `virtio`, est actuellement en cours de conception par les développeurs du noyau Linux. Cette interface permettra d'offrir une base commune pour les E/S, pour toutes les solutions de virtualisation, y compris pour la paravirtualisation et les hyperviseurs.

Toujours dans la communauté des développeurs du noyau Linux, Rusty Russell, développeur et mainteneur du pare-feu *(firewall)* de Linux, a récemment diffusé Lguest. Lguest est un projet de paravirtualisation centré sur GNU/Linux, qui vise à servir de base pour tester toutes les fonctionnalités en développement. C'est par exemple ici que la couche `paravirt_ops` a été testée et améliorée avant d'être intégrée au noyau. Lguest doit être vu comme une plate-forme de tests, et non comme un projet de paravirtualisation ayant pour but d'être utilisé en production. Le projet Lguest sera intégré dans la version 2.6.23 du noyau (prévue pour la fin de cet été), lui assurant une diffusion encore plus importante et potentiellement l'augmentation du nombre de développeurs intéressés par la virtualisation sous GNU/Linux.

Les fondeurs de processeurs de la famille PC — c'est à dire Intel et AMD — ont eux aussi compris l'intérêt qu'il y avait à développer les possibilités de virtualisation. En effet, les processeurs utilisés sur les architectures PC [3] sont notoirement difficiles à émuler de manière performante, du fait du nombre d'instructions nécessitant un niveau de privilège élevé (donc hors d'atteinte d'une machine virtuelle s'exécutant comme un processus utilisateur), ainsi que de la façon dont la mémoire est gérée. Intel et AMD ont donc décidé

3. On les appelle des processeurs *x86*, car leur conception descend directement des premiers processeurs Intel, les 8086

d'intégrer à leurs nouvelles générations de processeurs des instructions supplémentaires visant à améliorer les performances des solutions de virtualisation.

Ces nouvelles instructions, appelées VT *(Virtualization Technology)* chez Intel et SVM *(Secure Virtual Machine)* chez AMD, sont expliquées en détail dans l'article [NSL+06] et dans la documentation technique des nouveaux processeurs AMD, [AMD06]. Une machine virtuelle tirant parti de ces nouvelles instructions s'exécute à un niveau de privilège intermédiaire, plus élevé que celui d'une application utilisateur normale mais moins que celui du système d'exploitation. Dans ce niveau de privilège, la plupart des instructions du système invité sont exécutables directement, sans intervention de la machine virtuelle, le processeur s'occupant de tout. Seul un faible nombre d'instructions obligent encore le processeur à repasser la main à la machine virtuelle, qui émule alors le comportement attendu de manière logicielle. En utilisant les instructions de virtualisation les gains sont significatifs, car la machine virtuelle est déchargée d'une partie conséquente de son travail d'émulation par le processeur, qui est évidemment beaucoup plus rapide.

Toutefois, Intel et AMD vont encore plus loin, car récemment, ils ont tous les deux mis au point des technologies pour décharger encore plus la machine virtuelle, avec l'introduction de la virtualisation pour les E/S. Cette technologie, décrite dans [AMD07] et [AJM+06], permet au processeur d'effectuer tout le travail de traduction des adresses lors d'un processus d'E/S. Les traductions d'adresses effectuées par la machine virtuelle étaient le goulet d'étranglement des E/S. Avec une machine virtuelle tirant parti de ces fonctionnalités, le gain de performances est encore plus important. L'interface `virtio` devrait pouvoir tirer parti de ces nouvelles fonctionnalités.

On peut cependant regretter que les instructions de virtualisation d'Intel et d'AMD, même si elles ont le même usage et visent les mêmes buts, sont incompatibles et ne suivent pas de standard défini. C'est donc aux projets souhaitant bénéficier de ces instructions d'implémenter une couche d'abstraction supplémentaire pour tirer parti des deux types d'instructions de manière transparente pour l'utilisateur.

Intel vient également d'investir un peu plus de 200 millions de dollars en actions chez VMware au début du mois de juillet, lui assurant ainsi 2,5 % des parts de la société ainsi que la nomination d'un représentant d'Intel au conseil d'administration de VMware.

Parallèlement, la société VMware a effectué son introduction en bourse au mois d'août 2007, mettant environ 950 millions de dollars d'actions sur le marché, soit 10 % des parts de l'entreprise.

Ces deux substantielles rentrées d'argent permettront à la société VMware d'investir dans la recherche sur de nouvelles technologies, en collaboration avec Intel.

Toujours pendant le mois d'août 2007, la société XenSource, éditrice de la solution de virtualisation Xen, s'est faite racheter par Citrix, un des leaders dans le domaine de la fédération d'applications, c'est à dire la concentration des applications sur un serveur central, avec déport de l'affichage sur des clients légers. Citrix vise ainsi à étendre son offre de produits à la virtualisation pour offrir une gamme de produits complète, allant de la centralisation d'applications à la centralisation de systèmes d'exploitation.

La société Qumranet a diffusé il y a peu le logiciel KVM, qui est basé sur QEMU pour la machine virtuelle. Cependant, KVM tire parti des instructions de virtualisation offertes par les nouveaux processeurs x86, ce qui amène un gain de performances significatif par rapport à une solution uniquement basée sur QEMU. Le tour de force de Qumranet a été de faire intégrer KVM au noyau Linux, lui assurant ainsi une grande diffusion et un fort succès auprès de la communauté.

Une jeune société nommée Virtual Iron, financée en grande partie par Intel, vient de sortir un nouveau produit de virtualisation, destiné à concurrencer les produits de VMware et XenSource. Ce nouveau logiciel utilise les instructions de virtualisation des nouveaux processeurs Intel et AMD. Il se base sur le code source du projet Xen et rajoute une interface de supervision au dessus des systèmes virtualisés pour en faciliter l'administration.

Tous ces investissements et acquisitions en l'espace de quelques mois montrent qu'il y a un réel mouvement de fond autour de la virtualisation. Toutes les entreprises veulent en faire partie et proposer une solution à leurs clients. Le marché de la virtualisation est en pleine croissance, et cela ne fait que commencer.

Étant donné le nombre de produits et de technologies disponibles actuellement sur le marché, il est nécessaire de bien étudier les caractéristiques des différents produits avant de choisir. En effet, selon les besoins et contraintes de la société, les solutions à adopter ne seront pas les mêmes.

Le chapitre suivant sera consacré à l'étude des besoins d'une PME spécialisée dans l'hébergement de sites web en matière de virtualisation, puis à l'analyse détaillée de quelques solutions de virtualisation.

Chapitre 2

Analyse de solutions majeures de virtualisation

Le choix de *la* bonne solution de virtualisation dans l'absolu est très difficile, et la pléthore de projets de virtualisation disponibles ne facilite pas la décision. C'est pourquoi il est nécessaire de réduire le champ d'étude aux solutions applicables dans un cas bien spécifique. Dans ce livre blanc, nous nous limiterons aux solutions de virtualisation utilisables pour une PME ayant comme principale activité l'hébergement d'applications Web. La virtualisation pour les postes de bureau, comme par exemple pour le développement et le test d'applications Web, sera néanmoins survolée.

2.1 Expression des besoins et contraintes

Pour la plupart des petites entreprises faisant de l'hébergement, le parc de serveurs est très hétérogène, des machines récentes côtoyant des machines d'ancienne génération au fil des achats de nouveau matériel. Le taux d'utilisation des serveurs est quant à lui très disparate, certains serveurs étant déjà très chargés alors que d'autres sont loin des limites d'utilisation. Cet écart s'explique en partie par le type d'applications hébergées — certaines drainant plus d'utilisateurs que d'autres — ; mais aussi par le type de technologie utilisée par les applications (PHP, *framework* PHP, Ruby on Rails, ...).

Tous ces facteurs rendent très intéressants la possibilité de se détacher des machines physiques pour la gestion des applications hébergées. Il s'agit donc d'arriver à répartir le plus uniformément possible le taux d'utilisation sur l'ensemble de ses serveurs.

2.1.1 Contraintes

Dans le monde des serveurs et de l'hébergement, le logiciel libre est prédominant, il est donc tout à fait logique de se tourner vers des solutions de virtualisation libres pour l'hébergement. En effet, beaucoup de sociétés d'hébergement se basent sur des technologies libres (Linux, Apache, PHP, MySQL, PostgreSQL, etc.). Par conséquent, la solution de virtualisation utilisée doit obligatoirement être libre.

De plus, les systèmes d'exploitation utilisés pour l'hébergement à l'aide de technologies libres sont la plupart du temps des systèmes GNU/Linux. Le projet retenu doit donc bien évidemment fonctionner sur GNU/Linux, permettre de faire fonctionner des systèmes invités GNU/Linux et être facilement intégrable à la distribution GNU/Linux utilisée par l'enterprise.

2.1.2 Besoins

Si les contraintes énoncées précédemment sont des conditions *sine qua non* pour utiliser le produit choisi, les besoins ne sont pas tous aussi indispensables, et il est envisageable de faire quelques concessions. Les besoins typiques d'une PME varient grandement en fonction de l'activité de celle-ci. On peut notamment lister deux grandes catégories : l'hébergement d'applications web d'une part, et le développement d'applications web d'autre part.

Pour l'hébergement

Tout d'abord, la solution choisie doit faciliter la création d'images systèmes complètes. Par image système complète, on entend la base du système d'exploitation ainsi que

tous les logiciels nécessaires pour héberger une application (serveur web, base de données, etc.) dans une configuration donnée. On doit pouvoir très simplement créer et configurer un nouveau système virtuel, de manière à pouvoir déployer rapidement un environnement sur un serveur.

Avec la création rapide d'image système vient aussi le besoin de pouvoir déployer rapidement ces images sur les serveurs cibles, sans avoir à passer du temps à configurer le logiciel de virtualisation pour cela. Idéalement, il suffirait de déplacer l'image du système et éventuellement un simple fichier de configuration sur un serveur pour pouvoir instancier un nouveau système invité. La solution retenue doit donc faciliter la répartition des applications sur les serveurs physiques. Il y a plusieurs méthodes pour cela, mais la plus répandue est la migration des systèmes invités entre deux machines physiques. Il suffit en général d'arrêter le système invité, de déplacer l'image disque du système sur la nouvelle machine et redémarrer le système invité. On peut aussi envisager une migration « à chaud » *(live migration)*, où le système invité n'a pas à être effectivement arrêté pendant la migration, juste brièvement mis en pause.

Afin de faciliter l'administration, il est nécessaire que l'on puisse contrôler facilement l'état des systèmes invités. En effet, la plupart des entreprises ont un système de surveillance des machines et des services, et l'intégration de la solution de virtualisation à ce système serait appréciable. Il est tout à fait envisageable d'avoir à programmer soi-même cette intégration, mais il faut que cela soit possible, c'est à dire que la solution utilisée permette d'accéder simplement aux informations souhaitées.

Toujours dans l'optique de faciliter les tâches d'administration, il est indispensable que l'on puisse contrôler le produit utilisé entièrement par la ligne de commande. L'administration à distance sera ainsi possible, ce qui est indispensable quand il s'agit de vérifier un point de détail sur un serveur situé dans un *datacenter* éloigné géographiquement. De la même manière, l'obligation d'avoir une interface graphique sur la machine hôte est impensable : on doit pouvoir se servir de l'intégralité des fonctionnalités de la solution choisie sans écran ni affichage graphique ; un simple accès à distance en ligne de commande doit suffire.

Au niveau des performances, l'écart entre un système virtualisé et un système « natif » devrait être le plus faible possible. La performance d'un serveur web est la plupart du temps mesurée par son temps de réponse à une requête. Ce temps de réponse dépend

généralement de :
– la vitesse du processeur ;
– la vitesse des Entrées/Sorties ;
– la vitesse du matériel sous-jacent (disque dur, mémoire, carte réseau).
Le point le plus critique pour une solution de virtualisation est la vitesse des E/S. En effet, si la rapidité de traitement d'un système invité est quasiment celle du processeur de la machine physique — à quelques pour-cent près —, il n'en va pas de même pour les E/S. Il faut donc que la solution retenue soit performante pour le traitement des Entrées/Sorties, afin d'avoir un système invité à même d'héberger des applications exigeantes à ce niveau.

Il doit également être possible d'attribuer très rapidement une adresse IP publique à un système invité. En effet, une activité d'hébergement implique une connectivité à Internet, il faut donc que les systèmes invités soient joignables directement. Il suffit en pratique que la solution de virtualisation offre la possibilité d'avoir (au moins) une carte réseau virtuelle pour chaque système invité et qu'elle se charge de router le trafic à la bonne carte réseau virtuelle. Il est en effet souhaitable de conserver une IP publique par système d'exploitation (système hôte compris), afin de faciliter l'administration à distance. Avec un système d'exploitation GNU/Linux, ce n'est pas un problème, il existe plusieurs solutions pour obtenir ce résultat ; tout ce qui importe c'est que la solution de virtualisation soit en mesure d'exploiter ces mécanismes pour proposer les fonctionnalités voulues.

Les besoins en matière d'hébergement sont pour la plupart suffisamment généralistes et évidents pour que l'immense majorité des solutions de virtualisation les prennent en compte directement.

Pour le développement

Dans le contexte du développement web, la virtualisation est principalement utilisée pour effectuer des tests et pour accélérer le développement en local (par opposition au développement centralisé sur un unique serveur).

Tout d'abord, afin de faciliter le développement en local, il est nécessaire que chaque développeur dispose de sa propre machine virtuelle. Ainsi, il effectue les modifications

et visualise le résultat sur sa propre machine virtuelle, sans avoir à envoyer ses modifications au serveur central. Une fois les modifications validées, il pourra les enregistrer et les envoyer sur le serveur central. Il faut donc que la solution de virtualisation choisie soit facile à déployer dans ce contexte d'utilisation, notamment au niveau de l'environnement et de la configuration nécessaire pour l'application web développée. On peut tout à fait rassembler tous les systèmes invités des développeurs sur une seule machine physique située dans le réseau local de l'entreprise, il n'est pas nécessaire que la solution fonctionne sur leur machine de travail.

De même, il est parfois nécessaire de déployer un système invité qui hébergera une version donnée d'une application web, qui n'est pas forcément celle qui est hébergée en production ou en test pour le client. Il faut donc que les développeurs puissent facilement instancier un système virtuel préconfiguré pour une application donnée, et qu'ils puissent ensuite facilement restaurer la version souhaitée.

Une autre application de la virtualisation serait la gestion de tests automatisés. En effet, les développeurs souhaitent avoir une architecture complète pour la réalisation de tests pour chaque application en cours de développement. Ce serveur installerait chaque application avec l'environnement qui lui est associé, à la demande. Une fois le système invité démarré et l'application désirée déployée, une batterie de test serait exécutée, et les résultats seraient envoyés à l'équipe de développement de l'application.

Derrière tous ces besoins se dessine un thème récurrent : la facilité de déploiement d'une image système dans une configuration donnée. C'est donc là dessus que devront se porter tous les efforts de recherche et de mise au point d'une architecture de virtualisation.

Maintenant que les besoins pour l'hébergement et le développement sont connus, on peut passer à l'étude des solutions de virtualisation qui pourraient être utilisées.

2.2 Autres solutions

Avant de détailler les projets de virtualisation retenus pour l'analyse, il semble important de lister ci-dessous quelques projets qui, mêmes s'ils ne remplissent pas tous les critères

de choix listés précédemment, méritent d'être mentionnés.

2.2.1 Bochs

Bochs (prononcer « *box* ») est un émulateur très complet et multi plate-forme de l'architecture PC moderne. Il utilise la virtualisation complète. La première version publique de Bochs date de 1994. Il a été racheté puis « libéré » — c'est à dire diffusé sous une licence libre — en mars 2000 par la société Mandriva.

Contrairement à la plupart des logiciels de virtualisation, Bochs interprète *toutes* les instructions du système invité. En effet, la majorité des machines virtuelles passent la plus grande partie des instructions directement au processeur de la machine hôte, pour avoir des performances proches de celles du système hôte. Toutefois cela a un coût sur la complexité du code d'interprétation, qui doit être adapté pour chaque architecture hôte pour prendre en compte toutes les spécificités. Le projet Bochs vise exactement l'inverse : l'intégralité du code est interprété par Bochs, qui fait ensuite appel au processeur hôte pour réaliser les tâches demandées par le système invité.

L'avantage est que le code d'interprétation n'a à être écrit qu'une seule fois, dans un langage de programmation portable. On peut alors porter très facilement Bochs sur une nouvelle architecture hôte, sans avoir à modifier le code d'interprétation. L'inconvénient est par contre des performances moindres que d'autres projets de virtualisation complète, à cause de l'interprétation de toutes les instructions.

De par son code source très clair et très portable, Bochs est un sujet d'étude très utilisé, notamment pour comprendre comment l'émulation complète fonctionne et comment on réalise un émulateur de l'architecture PC. Ses faibles performances le rendent toutefois inutilisable pour autre chose que des tests et de la recherche.

2.2.2 VMware Server

La société VMware édite plusieurs logiciels de virtualisation, pour les utilisateurs finaux ou pour les serveurs. Elle est leader dans le marché de la virtualisation pour PC. Le pro-

duit à destination des entreprises est VMware Server, solution de virtualisation complète pour serveur sous GNU/Linux et/ou Microsoft Windows.

Les performances et les fonctionnalités offertes par VMware Server le placent parmi les meilleures solutions de virtualisation pour l'architecture PC. On peut optionnellement activer des modules de paravirtualisation pour augmenter davantage les performances.

Si ce n'était pour sa licence non-libre et relativement restrictive [1], VMware aurait tout à fait sa place parmi les solutions étudiées.

La section suivante étudiera les solutions de virtualisation qui pourraient être utilisées en production par une PME.

2.3 QEMU

2.3.1 Présentation

QEMU est le projet de virtualisation complète libre le plus abouti actuellement. Il a été fondé par Fabrice BELLARD et diffusé à la communauté en 2003. Fabrice BELLARD est également connu pour avoir fondé le projet FFmpeg, une suite d'utilitaires dédiés au traitement et à la conversion de flux numériques (vidéo et audio). À l'heure actuelle, M. BELLARD est toujours le mainteneur et développeur principal de QEMU, secondé par une communauté active.

2.3.2 Technologies

Techniquement, QEMU utilise la virtualisation complète (cf. tableau 2.1 page suivante), et il supporte de nombreuses architectures cibles (*i.e.* émulées), parmi lesquelles les processeurs x86 et l'architecture PC. Il fonctionne sur les plates-formes les plus courantes (Microsoft Windows, GNU/Linux, Mac OS X) et est très simple d'utilisation.

1. La licence d'utilisation interdit en effet la publication de bancs d'essais comparatifs.

L'article [Bel05] détaille précisément les choix techniques effectués pour émuler l'architecture PC, ainsi que les difficultés rencontrées pour virtualiser le processeur (difficultés communes à tous les projets de virtualisation pour x86). QEMU peut en option utiliser un module d'accélération système pour améliorer les performances. Ce module s'appelle `kqemu` *(kernel-QEMU)* et est disponible pour Windows et GNU/Linux. Il s'intègre au noyau (ou dans les services dans le cas de Windows) et permet à QEMU de contourner certaines couches d'abstraction du système hôte, amenant ainsi un gain de performances.

Virtualisation Complète	
Paravirtualisation	
Hyperviseur	
Cloisonnement	

TAB. 2.1 – Récapitulatif des technologies utilisées par QEMU

2.3.3 Fonctionnalités

Ce qui distingue QEMU de solutions plus intrusives (comme Linux-VServer ou Xen) est sa grande simplicité d'usage. En effet, comme c'est un projet utilisant la virtualisation complète, il y a un simple programme à exécuter sur le système hôte pour obtenir une nouvelle machine virtuelle contenant un système invité.

Un des gros points forts de QEMU est la flexibilité qu'il offre au niveau des options pour la configuration de la machine virtuelle. La section suivante détaillera la gestion des interfaces réseau et du trafic réseau en général, car c'est un des points forts de QEMU. L'étendue de ses possibilités dans ce domaine ont été l'un des critères pour sa sélection dans ce comparatif.

Gestion du réseau

QEMU offre plusieurs modes de fonctionnement pour la connexion du système invité au réseau. La première méthode, la plus simple, consiste à laisser un accès limité au réseau depuis la machine virtuelle. Dans ce mode — appelé *user mode networking* — QEMU offre un réseau privé au système invité, et filtre les communications. Du point de vue du système invité, il y a un réseau privé simple, avec un serveur DHCP qui attribue des adresses IP et une passerelle pour accéder à Internet. Ce mode est totalement indépendant du système hôte, le DHCP et la passerelle sont émulés par QEMU. Le déroulement du processus est le suivant :

1. QEMU est initialisé en *user mode networking* ;
2. le système invité démarre ;
3. à l'initialisation du réseau, le système invité effectue une requête DHCP pour configurer la carte réseau ;
4. la pile *user mode* de QEMU répond à la requête DHCP et attribue une adresse IP à la carte réseau du système invité ;
5. une fois la carte configurée, le système invité peut émettre du trafic sur le réseau virtuel, comme s'il était connecté à un véritable réseau ;
6. quand le système invité émet du trafic (par exemple quand une application utilisateur se connecte à Internet), QEMU récupère le trafic réseau et le passe au système hôte, qui se charge alors de l'émettre sur le véritable réseau physique.

Du point de vue du système hôte, QEMU est un programme comme un autre, y compris pour l'accès au réseau. Cette particularité fait que la configuration *user mode* est très simple, QEMU se chargeant de tout. Par contre, il n'est pas possible de faire fonctionner un programme hébergeant un service réseau (par exemple un serveur Web) avec ce mode, sauf à dire explicitement à QEMU d'écouter sur un port donné et d'avoir l'autorisation du système hôte pour écouter sur ce port. Ce mode est donc réservé aux tests rapides, ou aux cas d'utilisation ne nécessitant qu'un usage basique de l'accès à Internet.

Le deuxième mode proposé est le mode *TAP* (ou *tuntap*). Dans cette configuration, une interface réseau virtuelle est créée sur le système hôte. Cette interface virtuelle est ensuite reliée à l'interface physique par un pont *(bridge)*, qui servira à faire passer le trafic réseau de l'une vers l'autre. QEMU relie alors le système invité à l'interface

virtuelle. Dans ce mode de fonctionnement, il y a donc une interface virtuelle assignée à une machine virtuelle. Comme cette interface est reliée à une interface physique, on peut lui envoyer du trafic réseau, ce qui offre la possibilité d'héberger des serveurs dans la machine virtuelle. Chaque machine virtuelle ainsi créée dispose donc de son adresse IP au sein du réseau. Les systèmes invités deviennent indissociables de véritables machines, du point de vue du réseau.

Ce fonctionnement est toutefois un peu plus complexe à mettre en place que le mode précédent, et nécessite des privilèges administrateurs, au moins pour la création de l'interface virtuelle et le lien avec l'interface physique. C'est le mode de connexion au réseau le plus intéressant, dans l'optique de l'hébergement de services au sein des systèmes invités.

Une autre particularité intéressante de QEMU est la possibilité de relier les systèmes invités entre eux, indépendamment du réseau physique sous-jacent. Chaque instance de QEMU démarrée peut se voir préciser un numéro de VLAN (réseau virtuel). De cette manière, seules les machines virtuelles partageant le même numéro de VLAN peuvent communiquer entre elles. On peut ainsi relier les instances QEMU entre elles même si elles sont sur des machines physiques différentes.

QEMU dispose d'énormément de possibilités en ce qui concerne la gestion du réseau, mais ce n'est pas son seul point fort. En effet, il y a quantité d'options paramétrables au lancement du programme, qui contrôlent le type de matériel émulé, la connectivité au réseau, les disques et répertoires à rendre accessibles au système invité, etc.

Gestion des images systèmes

En ce qui concerne les images systèmes, QEMU se base sur des images disques complètes, avec son propre format de stockage *(QCOW — QEMU Copy On Write)*. Il y a un utilitaire qui permet de créer des images de la taille souhaitée. Les images au format *QCOW* sont « creuses », c'est à dire qu'à la création du fichier, il n'occupe que quelques octets. Il grossira au fur et à mesure que le système invité remplit l'espace disque. Cela permet d'économiser de la place et de ne pas avoir un énorme fichier de plusieurs gigaoctets à déplacer quand il y a seulement une image système minimale installée. À titre d'exemple, une distribution Debian minimale installé dans une image *QCOW* de

5 gigaoctets occupe sur le disque seulement 700 mégaoctets.

L'inconvénient principal des images disques est qu'il est relativement difficile d'accéder au contenu de l'image depuis le système hôte. En effet, le format *QCOW* n'est pas une simple image de disque dur. Si l'on souhaite pouvoir accéder aux fichiers dans une image disque créée avec QEMU, il y a deux alternatives possibles : la première est de démarrer le système invité installé sur l'image disque et d'effectuer les modifications. Ce n'est pas toujours possible, ni particulièrement rapide. La seconde possibilité est de créer les images disques au format « brut » *(raw)*, c'est à dire une copie conforme d'un disque dur. Par contre, avec cette solution, une image disque de 5 gigaoctets occupera effectivement 5 gigaoctets sur le disque.

2.3.4 Communauté

Le projet a su attirer de nombreux contributeurs, parmi lesquels Rusty Russell (auteur de Lguest, mentionné page 27). L'activité autour de QEMU est toujours importante et les contributions externes sont relativement régulières, gage que QEMU dispose d'une forte communauté. Cette communauté est rassemblée autour d'une liste de diffusion, d'un forum de discussion et d'un wiki*. De plus, il y a plusieurs projets autour de QEMU, qui visent pour la plupart à offrir une interface aux nombreuses options de ligne de commande du programme.

2.3.5 Réutilisation du projet

De par sa robustesse et ses fonctionnalités étendues, QEMU est réutilisé dans plusieurs solutions de virtualisation. On peut notamment citer :

KVM la partie utilisateur de KVM utilise une version légèrement modifiée de QEMU ;

Xen lorsque Xen est utilisé avec les processeurs supportant les instructions de paravirtualisation, QEMU est la machine virtuelle se chargeant des E/S ;

Virtualbox la solution de virtualisation Virtualbox se base également sur QEMU pour la réalisation des E/S.

2.3.6 Licence

QEMU est diffusé sous la licence GNU GPL. Jusqu'à récemment, le module d'accélération `kqemu` était diffusé sous une licence interdisant la redistribution, mais il est dorénavant distribué sous la même licence. QEMU est donc un logiciel libre parfaitement compatible avec les contraintes énoncées précédemment.

2.3.7 Inconvénients

Tout d'abord, du fait de la grande complexité du code réalisant l'émulation du processeur, il sera difficile de comprendre et modifier les sources du programme, si jamais le besoin se fait sentir. L'évolution du projet est donc aux mains des personnes à même de comprendre l'architecture bas niveau de QEMU.

Toujours au niveau du code, il est dit dans [Bel05] que QEMU est dépendant de la version du compilateur GCC (*GNU Compiler Collection*, le compilateur du projet GNU). En effet, le projet se base sur GCC pour générer des extraits de code à la volée durant la compilation. Ces bouts de code seront ensuite utilisés pendant la phase d'émulation de certaines instructions du processeur. Or, à chaque changement de version du compilateur, le code généré change légèrement, « cassant » d'une certaine manière tout l'alignement prévu par QEMU. C'est pourquoi il est nécessaire de compiler QEMU avec des versions très précises de GCC : les versions 3.2 ou 3.4. Cela commence à devenir problématique, car ces versions ne sont plus vraiment supportées à l'heure actuelle ; la dernière version de la branche 3.2 date de 2003 et celle de la branche 3.4 de 2006. La branche actuelle de GCC est la 4.2. La solution la plus simple est donc pour l'instant d'utiliser la version de QEMU fournie par la distribution GNU/Linux utilisée (dans notre cas, la version de Debian). Toutefois si l'on souhaite tester un changement ou modifier les options de compilation, il est nécessaire d'installer une ancienne version de GCC.

Les performances de la machine virtuelle sont un autre facteur limitatif. Comme toutes les solutions de virtualisation complète, QEMU pêche au niveau de la rapidité des Entrées/Sorties. Le module d'accélération `kqemu` améliore certes les performances, mais on reste loin des performances « natives ». Il faut toutefois signaler que même si les performances sont éloignées de celles d'une véritable machine, QEMU se place parmi les

solutions de virtualisation complète les plus performantes.

2.3.8 Bilan

QEMU reste une solution intéressante dans les cas d'utilisation où l'on a simplement besoin de tester le comportement d'un programme ou d'un système. Les options d'interconnexion qu'il offre, tant avec le système hôte qu'avec d'autres machines du réseau sont très intéressantes, et permettent de mettre au point des architectures très complexes, impossibles à réaliser avec des machines physiques.

Cependant, pour l'hébergement de serveurs en production, QEMU seul offre des performances insuffisantes par rapport à la puissance de la machine hôte. Ce n'est donc pas une solution retenue dans notre étude.

Il sera potentiellement plus intéressant d'utiliser KVM, détaillé dans la section suivante.

2.4 KVM

2.4.1 Présentation

Le projet KVM (*Kernel-based Virtual Machine* – machine virtuelle dans le noyau) a été créé en 2006 par la société Qumranet. KVM a su très vite attirer les contributions externes et le code source du projet a été intégré au noyau Linux dès février 2007. Le mainteneur principal de KVM est Avi KIVITY. À l'heure actuelle, le projet KVM est financé principalement par Qumranet, mais IBM participe aussi au développement en salariant un développeur pour travailler sur KVM : Anthony LIGUORI. Il est d'ailleurs intéressant de constater qu'un des fondateurs de la société Qumranet, Moshe BAR, est aussi un des fondateurs de la société XenSource, la société à l'origine de Xen (cf. page 54).

2.4.2 Technologies

KVM est un projet de virtualisation complète qui utilise les instructions de virtualisation des processeurs x86 récents. Techniquement, KVM se compose :
– d'un module noyau qui utilise les instructions de virtualisation et communique avec le processeur ;
– d'un programme utilisateur, qui utilise le module noyau pour toutes les opérations privilégiées.
La partie utilisateur de KVM est une version légèrement modifiée de QEMU, simplement adaptée pour que les opérations pouvant bénéficier des instructions de virtualisation du processeur fassent appel au module noyau.

L'article [Qum07] présente brièvement les principes derrière KVM. Lors du dernier Linux Symposium, qui s'est tenu au mois de juin 2007, Avi KIVITY et d'autres développeurs de KVM ont tenu une conférence sur le fonctionnement détaillé de KVM. L'article correspondant (plus complet que le précédent) est [KKL+07].

Le tableau 2.2 récapitule les technologies utilisées par KVM. Il y actuellement en développement un module de paravirtualisation, qui pourra servir à accélérer encore davantage les performances des Entrées/Sorties, notamment au niveau de la communication réseau. Ce module est toutefois facultatif — en plus d'être instable et toujours en développement — et l'on peut tout à fait s'en passer, c'est pourquoi il est marqué entre parenthèses dans le tableau.

Virtualisation Complète	✓
Paravirtualisation	(✓)
Hyperviseur	
Cloisonnement	

TAB. 2.2 – Récapitulatif des technologies utilisées par KVM

2.4.3 Fonctionnalités

KVM étant basé sur QEMU, il reprend naturellement toutes les fonctionnalités de ce dernier, qui ont déjà été traitées en page 38.

Il y a toutefois une différence notable : grâce à l'utilisation des instructions de virtualisation des processeurs récents, les performances des systèmes invités sont bien plus élevées qu'avec QEMU.

Une autre fonctionnalité qui se démarque de QEMU est la migration des systèmes invités. KVM propose en effet la migration « à chaud » des hôtes, mais d'un fonctionnement différent de la migration d'images de QEMU. Le processus de migration de systèmes invités de KVM est actuellement plus performant que celui de QEMU, mais QEMU bénéficiera sous peu des améliorations de KVM.

2.4.4 Communauté

La communauté autour de KVM est à l'image de celle de QEMU : diversifiée et active. Les deux projets partagent d'ailleurs une bonne partie de leur communauté. Les mêmes moyens de communication sont utilisés : liste de diffusion et wiki.

Le développement est principalement assuré par les employés de Qumranet (notamment Avi KIVITY, mainteneur officiel), mais il y a des contributeurs externes réguliers. On peut notamment citer Rusty RUSSELL, décidément très actif dans le milieu de la virtualisation, ou encore des employés d'Intel, d'AMD ou d'IBM. Le projet a donc réussi à attirer l'attention des grands acteurs de la virtualisation.

La société Qumranet collabore étroitement avec les développeurs de QEMU, et la plupart des modifications apportées à QEMU pour le rendre plus performant sont répercutées dans le projet original. Seules les modifications spécifiques à KVM ne sont pas appliqués, car elles n'ont de sens que pour KVM.

2.4.5 Licence

KVM est majoritairement sous licence GNU GPL, avec quelques variations mineures sur certains composants. L'intégralité du projet est néanmoins sous une licence libre, le rendant compatible avec les contraintes énoncées précédemment.

2.4.6 Inconvénients

Étant basé sur QEMU, KVM souffre des mêmes inconvénients sur la complexité du code et la difficulté de manipuler les images disques.

En plus de cela, la partie propre à KVM, c'est à dire le module noyau utilisant les instructions de virtualisation, est encore jeune et manque de maturité. La stabilité de l'interface de programmation *(API)* vient tout juste d'être atteinte, mais il n'est pas exclu que l'*API* change à nouveau. Cela signifie qu'un outil très proche de KVM peut ne plus marcher après une mise à jour.

La jeunesse du projet a un autre inconvénient : le manque de retours d'expérience concernant des architectures complexes de virtualisation utilisant KVM. En effet, peu d'entreprises — voire pas du tout — ont publiquement annoncé qu'elles utilisaient KVM pour virtualiser des serveurs à grande échelle. Même si KVM est effectivement utilisable en production pour un petit nombre de machines virtuelles, la difficulté de maintenance et la montée en charge restent inconnues. Il y a donc un risque (certes faible, mais pas inexistant) d'affronter des difficultés inattendues en utilisant KVM pour virtualiser un ensemble de serveurs.

2.4.7 Bilan

Le projet KVM, en dépit de sa jeunesse et de son manque apparent de maturité, est très intéressant pour une société souhaitant virtualiser des serveurs. Son intégration au noyau Linux lui assure une durée de vie incomparable par rapport à un projet externe, et l'attrait de la communauté et des entreprises pour ce projet est un gage supplémentaire de pérennité.

Le potentiel d'évolution de KVM est très élevé, avec une communauté motivée et un développement pour l'instant ininterrompu. Même s'il n'a pas toutes les fonctionnalités nécessaires à l'heure actuelle, il y a des chances qu'elles soient implémentées rapidement.

C'est pourquoi, même s'il est actuellement risqué de l'utiliser en production, il été sélectionné dans la liste des projets à étudier plus profondément. En effet, KVM sera très certainement un des acteurs majeurs plus tard, il faut donc l'étudier et le maîtriser dès maintenant, ne serait-ce que pour être prêt le jour où il sera vraiment utilisable.

2.5 Linux-VServer

2.5.1 Présentation

Le projet Linux-VServer est un projet de virtualisation par cloisonnement pour la plateforme GNU/Linux. Au départ, ce projet s'appelait *Virtual private servers and security contexts* et était géré par la société Solucorp. Il a été ouvert à la communauté fin 2001, et a eu une évolution lente mais constante depuis. La version actuelle est la version 2.2.0.3, qui a été diffusée au début du mois de juillet 2007.

Parmi les utilisateurs connus de Linux-VServer, on retrouve notamment LinuxFr.org, site communautaire d'informations autour du logiciel libre et de GNU/Linux. Le site de LinuxFr.org génère actuellement 100 000 visites par jour et près de 10 millions de pages vues par mois. La charge engendrée par une telle activité est absorbée par un serveur HP très haut de gamme, et Linux-VServer est utilisé pour cloisonner les différents systèmes hébergés sur la machine. Les autres systèmes invités s'exécutant sur cette machine ne sont pas impactés par la forte charge du site LinuxFr.org.

Un autre utilisateur réputé de Linux-VServer est le projet *One Laptop Per Child (OLPC)*. Ce projet vise à créer et diffuser des ordinateurs portables à très bas coût dans les écoles des pays en voie de développement. Le projet *OLPC* se base intégralement sur des logiciels libres pour la création des applications éducatives à destination des enfants. Linux-VServer est utilisé pour cloisonner les applications, afin d'éviter tout risque de propagation de *malware**.

2.5.2 Technologies

Le projet Linux-VServer utilise le cloisonnement, c'est à dire qu'il isole des instances du système d'exploitation par dessus un système hôte. Le tableau 2.3 récapitule les technologies utilisées par le projet.

Virtualisation Complète	
Paravirtualisation	
Hyperviseur	
Cloisonnement	✓

TAB. 2.3 – Récapitulatif des technologies utilisées par Linux-VServer

Du point de vue technique, il se compose de *patches** à appliquer sur le code source du noyau Linux et d'utilitaires fonctionnant dans l'espace utilisateur pour contrôler la création et l'administration des systèmes invités.

Linux-VServer se base sur les fonctionnalités déjà implémentées au sein du noyau pour la séparation des processus et les droits d'accès pour implémenter une séparation complète au niveau du noyau. Les modifications apportées utilisent et étendent les fonctionnalités déjà présentes dans le noyau.

Il y a un seul noyau Linux, qui contrôle toujours l'accès au matériel, mais il est partagé entre plusieurs instances du système, qui ont chacune un accès complet au noyau, sans connaissance des autres systèmes invités. La possibilité d'avoir un accès complet au noyau signifie entre autres qu'il n'y a pas besoin de porter les applications utilisateurs pour qu'elles fonctionnent sur Linux-VServer. Pour l'administrateur, la machine physique comporte un seul système d'exploitation, mais pour les programmes utilisateurs, chaque partition du système voit sa propre instance du noyau.

2.5.3 Fonctionnalités

Comme le projet utilise une technologie de cloisonnement, il n'y a quasiment pas de perte de performances par rapport à un système natif, mais il y un net gain en sécurité et simplicité d'administration du fait de la séparation entre les systèmes invités.

Contrairement à QEMU et KVM, Linux-VServer ne travaille pas avec une image disque du système, mais avec un simple ensemble de fichiers dans un répertoire, qui représentera la racine du système invité. Cela implique une plus grande facilité d'utilisation. En effet, il est possible avec ce mode de fonctionnement de modifier des fichiers du système invité en les ouvrant depuis le système hôte, comme n'importe quel autre fichier. On peut donc opérer des modifications, ou déplacer une image système d'une machine à l'autre très facilement.

Par contre, les possibilités de configuration réseau sont moindres que pour QEMU, car Linux-VServer reste très proche du noyau Linux. Il y a quelques options pour assigner une carte réseau à un système invité ou au contraire répartir la carte entre plusieurs. Pour de l'hébergement de serveurs en production cela est amplement suffisant, mais certainement pas pour faire des tests avec des configurations spécifiques.

Il est intéressant de noter qu'il y a à l'heure actuelle deux projets d'outils pour la partie en espace utilisateur, nécessaire pour administrer et créer les systèmes invités. Le premier outil est `util-vserver`, qui est l'outil de gestion « historique » du projet Linux-VServer. Le second outil, *VServer Control Daemon*, est en phase de développement et permettra d'avoir une centralisation des serveurs sur une seule interface.

2.5.4 Communauté

La communauté du projet Linux-VServer est moyennement active, centrée autour d'un wiki et d'une liste de diffusion. Il y a quelques acteurs principaux, dont le fondateur du projet et deux mainteneurs.

Le projet souffre néanmoins d'une diffusion trop confidentielle et d'un manque de visibilité qui lui porte préjudice. La communauté ne s'agrandit pas, ou du moins pas assez vite pour que cela se remarque.

2.5.5 Licence

Le projet Linux-VServer est intégralement sous licence GNU GPL.

2.5.6 Inconvénients

Malgré des performances très élevées, le projet Linux-VServer souffre de quelques inconvénients. Tout d'abord, la séparation entre les systèmes invités est trop « faible », dans le sens où il y a un seul noyau Linux pour plusieurs systèmes invités. Si un système invité trouve un moyen de contourner les protections et affecte le noyau, tous les autres systèmes de la machine en pâtiront. C'est une faiblesse dont souffrent toutes les solutions de virtualisation utilisant le cloisonnement.

Un autre problème majeur est la difficulté de maintenir le projet à jour. En effet, comme le code de Linux-VServer est un *patch* pour le noyau Linux, il est possible — et même très fréquent — qu'une nouvelle version du noyau « casse » la compatibilité du *patch*. Il faut alors attendre qu'une nouvelle version du projet Linux-VServer sorte spécifiquement pour la nouvelle version du noyau. Ce reproche peut être adressé à tous les projets modifiant en profondeur le noyau mais qui sont maintenus séparément. Garder ce genre de projets à jour est difficile et très consommateur de temps. En cas de mise à jour de sécurité, il y a toujours un décalage entre la sortie du noyau et la sortie de projet, ce qui peut apporter un risque supplémentaire si le serveur est exposé.

2.5.7 Bilan

Pour conclure, le projet Linux-VServer, en dépit de ses qualités, n'offre pas toutes les garanties nécessaires, notamment sur le soutien de la communauté, trop réduite. Il est de plus relativement complexe à maîtriser, et rares sont les administrateurs connaissant ce projet. C'est pourquoi cette solution n'a pas été retenue pour une architecture de virtualisation. Cependant, si au sein de l'entreprise, un administrateur maîtrise Linux-VServer, ce dernier devient une alternative intéressante qu'il convient de ne pas négliger.

2.6 OpenVZ

2.6.1 Présentation

OpenVZ est un projet de virtualisation par cloisonnement géré par la société SWsoft. Le modèle commercial de SWsoft consiste en une offre propriétaire (et payante) nommée Virtuozzo, qui se base sur le projet libre OpenVZ. Les clients souhaitant un support technique de qualité et des performances très élevées peuvent se tourner vers le produit Virtuozzo, les clients préférant utiliser une technologie libre et pouvant se contenter du support fourni par la communauté utiliseront OpenVZ.

La société SWsoft a été créée aux États-Unis en 1997. En 2001, le produit Virtuozzo est diffusé, avec son pendant libre : OpenVZ. SWsoft rachète Parallels en 2004, une société spécialisée dans la virtualisation pour les particuliers, très populaire sur Mac OS X. La société SWsoft est également connue comme éditrice de Plesk, une interface web sous forme de panneau de contrôle pour gérer des hébergements de sites web.

2.6.2 Technologies

Le principe de fonctionnement d'OpenVZ est très similaire à celui de Linux-VServer, car ils se basent tous les deux sur une modification du noyau Linux pour implémenter un système de cloisonnement au niveau du système d'exploitation.

Virtualisation Complète	
Paravirtualisation	
Hyperviseur	
Cloisonnement	✓

TAB. 2.4 – Récapitulatif des technologies utilisées par OpenVZ

OpenVZ modifie le noyau Linux plus en profondeur que le *patch* de Linux-VServer. Il y a donc des fonctionnalités spécifiques à OpenVZ.

Parmi ces nouvelles fonctionnalités, on peut notamment citer l'ajout d'un niveau supplémentaire d'indirection pour les ordonnanceurs* du noyau. Ce nouveau niveau d'indirection permet de gérer les priorités entre les systèmes invités, puis ensuite au sein d'un système invité. C'est à dire que l'ordonnanceur de bas niveau décide d'abord à quel système invité passer la main, puis ensuite l'ordonnanceur traditionnel du noyau Linux décide à quel processus passer la main, au sein de ce système. La répartition de charge entre les systèmes invités est donc plus fine qu'avec Linux-VServer, qui utilise un système d'ordonnancement bien moins complexe qu'OpenVZ.

2.6.3 Fonctionnalités

Les fonctionnalités de haut niveau d'OpenVZ sont quasiment identiques à celles de Linux-VServer : arrêt, instanciation et contrôle des systèmes invités sont d'un fonctionnement similaire. Seule une étude approfondie permet de noter quelques différences, qui viennent pour la plupart d'une intégration plus en profondeur.

Une fonctionnalité intéressante d'OpenVZ est la possibilité de migrer un système invité « à chaud », sans avoir à le désactiver avant. Le terme utilisé dans le cas d'OpenVZ est *checkpointing*. Cela consiste à sauvegarder l'état complet du système invité dans un fichier sur le disque, puis à le transférer sur une autre machine où l'état pourra alors être restauré.

Au niveau du contrôle des ressources, OpenVZ propose un ensemble de limites par système invité, que l'administrateur peut configurer. Par exemple, il peut allouer un nombre maximal de processus qu'un système invité aura le droit de créer, ou encore le nombre de connexions réseau, etc. Ces limites par système invité permettent un très bon contrôle et garantissent qu'un système ne nuira pas aux autres.

La connectivité réseau est là aussi très similaire aux autres solutions, chaque système invité peut disposer de sa propre adresse IP, c'est le noyau modifié qui s'occupe de rediriger le trafic réseau vers le bon système.

2.6.4 Communauté

La communauté d'OpenVZ est centralisée autour d'un wiki, de plusieurs listes de diffusion et du site web de SWsoft. Dans l'ensemble, la communauté donne une impression de meilleure organisation et de plus grande activité que celle de Linux-VServer.

Le projet OpenVZ est certes mis en avant par la société, mais uniquement après la solution propriétaire Virtuozzo.

La documentation disponible est par contre très bien fournie, et on peut très rapidement avoir une idée du produit, tant du point de vue utilisation et administration que pour l'aspect technique.

2.6.5 Licence

Le projet OpenVZ est distribué sous la licence GNU GPL.

2.6.6 Inconvénients

Le principal problème avec OpenVZ est que SWsoft ne « joue pas le jeu » du logiciel libre. En effet, le projet libre dispose de moins de fonctionnalités que la version propriétaire et payante. La version payante est montrée comme la version complète, alors que la version libre est juste là pour appâter la communauté et attirer les contributions.

De plus, la société se targue de posséder plusieurs brevets sur les méthodes de virtualisation. Le mouvement du logiciel libre étant en général plutôt opposé à l'idée de brevetabilité des logiciels et des idées, cette mise en avant de brevets logiciels est plutôt malvenue.

2.6.7 Bilan

Pour les raisons citées précédemment, et malgré d'excellentes performances et une très bonne documentation, le projet OpenVZ n'est pas acceptable pour une PME impliquée dans le logiciel libre, en raison d'un décalage idéologique trop grand. Il n'a donc pas été retenu pour une étude approfondie. Il est cependant tout à fait envisageable de l'utiliser si l'on fait abstraction de ce problème.

2.7 Xen

2.7.1 Présentation

Xen (dont le nom vient du grec *xenos*, étranger) est un projet de virtualisation par hyperviseur géré par la société XenSource. Le projet était à l'origine mené au sein de l'Université de Cambridge, sous le nom de Xenoserver. Le but était alors d'héberger 100 systèmes invités sur une seule machine physique, avec les meilleurs performances possibles.

En 2003, les initiateurs du projet ont fondé la société XenSource et ont lancé le projet Xen en se basant sur le code source de Xenoserver.

Le rachat en août 2007 de la société XenSource par Citrix ne devrait rien changer pour la communauté, la licence du produit libre restant inchangée. Toutefois, les entreprises pourront être rassurées de voir que XenSource est maintenant soutenue par des moyens financiers importants.

La solution de virtualisation Xen est séparée en plusieurs produits, ayant tous des finalités différentes. Il y a tout d'abord la version libre, nommée Xen 3.0, qui concentre toute la technologie de virtualisation. Les autres versions (propriétaires) de la gamme se distinguent uniquement par le support proposé, les nombre de machines virtuelles supportées, les systèmes invités supportés et les logiciels annexes. Ainsi, la version Xen Enterprise se base intégralement sur Xen 3.0, mais rajoute des outils de contrôle pour gérer plusieurs dizaines d'instances de Xen.

Le modèle commercial de XenSource est donc très différent de celui de SWsoft (société éditrice de Virtuozzo et OpenVZ, cf. page 51). Dans le cas de Xen, la technologie de virtualisation est la même partout, les versions propriétaires ont seulement des outils de gestion à grande échelle et un support technique fourni par XenSource. Les performances sont donc identiques entre les produits.

2.7.2 Technologies

Xen est un hyperviseur, c'est à dire qu'il vient s'insérer entre le matériel et le noyau. C'est donc Xen qui a l'accès exclusif au matériel, et les systèmes d'exploitation fonctionnant par dessus doivent obligatoirement passer par l'hyperviseur pour y accéder. Il peut ainsi répartir précisément les ressources entre les systèmes invités. Le tableau 2.5 récapitule le type de technologie que Xen utilise.

Xen étant une solution à base d'hyperviseur, les systèmes destinés à s'exécuter au dessus doivent être portés pour pouvoir cohabiter avec l'hyperviseur. Or, la modification des couches basses d'un système pour améliorer les performances est précisément la définition de la paravirtualisation. C'est pourquoi on considère que Xen est aussi une solution à base de paravirtualisation, car le système doit être modifié pour cohabiter.

Virtualisation Complète	
Paravirtualisation	✓
Hyperviseur	✓
Cloisonnement	

TAB. 2.5 – Récapitulatif des technologies utilisées par Xen

Au dessus de l'hyperviseur se trouvent les systèmes invités, qui sont tous contrôlés par Xen. Le premier système d'exploitation démarré par Xen a des privilèges particuliers. Ce système est appelé « domaine zéro » *(dom0)*, par opposition aux systèmes démarrés plus tard, appelés « domaines utilisateurs » *(domUs)*. Le domaine zéro (ou domaine 0) est le seul autorisé à créer et contrôler les domaines utilisateurs.

Le projet Xen se compose d'un ensemble de modifications à appliquer au noyau Linux (des *patches*) ainsi que des utilitaires en espace utilisateur qui permettent d'administrer les systèmes invités (les domaines utilisateurs, dans la terminologie de Xen). Les utilitaires sont dans le domaine zéro et interagissent avec l'hyperviseur par le biais d'appels systèmes spécifiques.

Une partie du code source de Xen a finalement été intégrée au noyau Linux durant l'été 2007, après plusieurs années d'attente. La plus grosse partie du code source est cependant encore maintenue séparément, avec les inconvénients que cela comporte.

2.7.3 Fonctionnalités

Au niveau des fonctionnalités, Xen est très certainement un des plus complets, il n'a pas à rougir de la comparaison avec des technologies propriétaires plus anciennes et mieux établies, telles que les solutions de la société VMware.

La configuration du réseau est à l'image de celle de QEMU : difficile à appréhender, mais néanmoins très puissante une fois maîtrisée. Au démarrage, Xen prend en charge la gestion des interfaces réseau, et crée une interface virtuelle pour chaque système invité, qui seront toutes reliées à une interface physique par un système de pont *(bridging)*. On peut donc, comme avec QEMU, attribuer une adresse IP distincte à chaque système invité, les rendant ainsi impossibles à différencier de véritables machines physiques sur le réseau.

Au niveau des images disques, Xen peut fonctionner de la même manière que les solutions à base de cloisonnement vues précédemment (Linux-VServer et OpenVZ) : le système invité sera un simple ensemble de fichiers dans un répertoire du système hôte.

Il y a néanmoins la possibilité de placer les systèmes invités sur des partitions dédiées du disque dur, voire sur un autre disque. De cette manière, les machines virtuelles bénéficient de leur propre espace, séparé du système hôte. Cela permet d'exploiter toutes les fonctionnalités de GNU/Linux pour la gestion des disques, notamment le RAID* logiciel et le redimensionnement [2] des partitions sans perte de données. On peut dans tous les

2. En pratique, seul l'agrandissement des partitions est sûr, pas la réduction. Il est de toute manière rare que l'on ait à réduire l'espace occupé par un système.

cas accéder directement aux fichiers des domaines utilisateurs, il suffit de « monter » la partition dans le système de fichiers du système hôte, de la même façon que l'on monte une clef USB ou un lecteur réseau.

Un autre point fort du projet Xen est la gestion des systèmes invités : on peut très facilement, avec les outils d'administration fournis, contrôler l'état d'un système, son activité (consommation du temps processeur et de la mémoire allouée), le mettre en pause ou l'arrêter complètement. Les outils de gestion permettent également de migrer les systèmes invités entre différents systèmes hôtes.

La migration « à chaud » est possible dans certaines conditions. En effet, Xen ne peut pas traiter correctement la conservation des connexions réseau si les systèmes d'origine et de destination ne sont pas sur le même domaine de collision (c'est à dire la même portion de réseau physique). La migration « à chaud » d'un système invité sans connexion réseau active est toutefois réalisable assez simplement, et le processus n'interrompt la machine virtuelle que quelques instants (le site web parle d'une durée d'interruption de 100 millisecondes).

Pour que la migration s'effectue rapidement, il est néanmoins nécessaire que les deux systèmes hôtes aient accès à l'image disque, par exemple par le biais d'un disque partagé sur le réseau. La rapidité de la migration dépend alors des performances du disque partagé.

Une migration avec le système invité arrêté est réalisable sans contraintes, et c'est de toute manière la fonctionnalité la plus pratique dans le cas d'utilisation envisagé.

2.7.4 Communauté

La communauté de Xen est rassemblée autour de plusieurs listes de diffusion, d'un wiki et de plusieurs forums. Elle est dans l'ensemble assez active, tant en contribution de documentation qu'en projets annexes, permettant de profiter au mieux de Xen.

L'attitude de la société XenSource vis à vis de la communauté est résumée par la phrase suivante, affichée de manière très visible sur leur site web :

> « *XenSource is totally committed to the Xen community and the open source process.* (XenSource est entièrement impliquée dans la communauté Xen et dans le processus *open source*.) »
>
> Ian PRATT, *leader* du projet Xen

Contrairement à SWsoft avec OpenVZ, la société XenSource est donc très ouverte et soutient activement la communauté utilisateur de Xen. La version propriétaire et payante de Xen apporte juste un contrat de support et des outils de gestion de plus haut niveau, mais pas de gains de performances.

Dans l'ensemble, la communauté Xen est très certainement la plus grand communauté parmi celles analysées ici. Le projet a su convaincre des milliers d'utilisateurs, gage de grande qualité et garantie que le projet ne sera pas à court de contributeurs du jour au lendemain.

2.7.5 Licence

Le projet Xen est diffusé sous la licence GNU GPL.

2.7.6 Inconvénients

Xen n'est toutefois pas dépourvu de défauts. Un des reproches majeurs fait à Xen est sa grande complexité. En effet, pour implémenter un hyperviseur performant, Xen modifie les couches basses de tous les systèmes d'exploitation, afin qu'ils puissent collaborer avec l'hyperviseur. Les modifications apportées au noyau Linux sont de l'ordre de plusieurs milliers de lignes de code. Elles portent essentiellement sur la création d'une interface avec l'hyperviseur et sur la gestion de l'accès au matériel à travers ce dernier. Ces modifications sont assez intrusives, et altèrent profondément la façon dont le noyau travaille en interne.

L'hyperviseur du projet Xen est également très complexe. En effet, pour assurer une répartition équitable des ressources, il faut un ordonnanceur très performant, capable d'assurer que chaque système invité aura accès aux ressources, tout en appliquant les

politiques définies. L'hyperviseur doit donc avoir un ordonnanceur capable de fournir d'excellentes performances, ce qui est loin d'être évident. Les développeurs du noyau Linux ont mis quelques mois avant d'arriver à obtenir un ordonnanceur de processus performant, et l'hyperviseur de Xen a besoin d'un ordonnanceur de systèmes d'exploitation, basé sur le même principe.

En plus de l'ordonnanceur, l'hyperviseur implémente aussi une gestion de la mémoire vive et des accès disque. Toutes les couches basses d'un système d'exploitation sont en fait réécrites au sein de l'hyperviseur. Cela augmente sensiblement la complexité du logiciel, ainsi que les risques d'erreurs.

En ce qui concerne la gestion du réseau, Xen est là aussi difficile à appréhender. La documentation utilisateur est très succincte à ce sujet, et il faut se contenter de laisser Xen tout gérer.

De plus, si la documentation est certes suffisante pour se *servir* de Xen, il n'y a pas de documentation technique officielle, qui détaille le fonctionnement interne de Xen. Ce type de documentation est nécessaire pour savoir comment optimiser l'architecture de virtualisation bâtie autour de Xen. Il y a bien quelques détails techniques sur le wiki, mais ils sont parfois incomplets, voire obsolètes.

Un autre aspect négatif du projet Xen est le fait qu'il est externe au projet Linux. Actuellement, seule une très faible portion du code modifiant le noyau Linux a été intégrée dans la branche officielle. Xen est donc en partie composé de *patches* de plusieurs milliers de lignes, maintenus et mis à jour séparément. Comme pour les autres projets extérieurs au noyau, cela pose un risque d'incompatibilité avec d'éventuelles modifications du noyau effectuées par l'utilisateur. De même, en cas de faille de sécurité, il y a un risque certain que la version Xen du noyau sorte plus tard que la version officielle, le système restant potentiellement vulnérable plus longtemps. Ce risque est donc à prendre en compte.

2.7.7 Bilan

La complexité de Xen a un avantage : les performances et les fonctionnalités sont impressionnantes, très proches de celles d'un système d'exploitation s'exécutant directe-

ment sur une machine physique, sans virtualisation.

Les fonctionnalités sont également à la hauteur des performances, permettant aux administrateurs de véritablement s'abstraire des machines physiques pour la gestion des systèmes. Les outils annexes proposés par la communauté sont aussi très efficaces, et apportent une véritable plus-value.

En plus de ses fonctionnalités complètes, Xen dispose d'une communauté très active, qui développe régulièrement de nouveaux outils pour simplifier l'administration ou la supervision des systèmes invités.

C'est pour toutes ces raisons que Xen a été retenu, et sera donc étudié en détail dans le chapitre suivant.

2.8 Récapitulatif

Dans ce chapitre, plusieurs solutions de virtualisation on été étudiées, chacune avec des technologies et des buts différents. Le tableau 2.6 récapitule les différentes technologies utilisées par les projets étudiés.

	QEMU	KVM	Linux VServer	OpenVZ	Xen
Virtualisation Complète	✓	✓			
Paravirtualisation		(✓)			✓
Hyperviseur					✓
Cloisonnement			✓	✓	

TAB. 2.6 – Récapitulatif des technologies utilisées par les différents projets

Parmi les projets non retenus, QEMU a été éliminé à cause de performances insuffisantes ; OpenVZ pour une « incompatibilité idéologique », malgré des performances de premier ordre ; Linux-VServer, enfin, n'a pas été retenu car il n'offrait pas assez de garanties sur la pérennité et l'évolution des développements à long terme.

Parmi les projets retenus, KVM a été sélectionné parce qu'il offrait le meilleur potentiel d'évolution à court et moyen terme ; Xen a lui été choisi car c'est tout simplement la solution libre la plus performante et la plus complète du marché à l'heure actuelle.

Le chapitre suivant analysera donc les deux solutions retenues pour une étude approfondie : KVM et Xen.

Chapitre 3

Étude comparative de Xen et KVM

Maintenant que la liste des projets potentiellement utilisables par une PME pour son architecture de virtualisation a été suffisamment réduite, ce chapitre va procéder à une analyse plus poussée des projets retenus.

Cette analyse portera à la fois sur le fonctionnement interne des logiciels, sur les possibilités de configuration et sur l'adéquation aux cas d'utilisation d'une PME.

3.1 Étude approfondie de Xen

3.1.1 Historique du projet

Le projet Xen a été fondé par Ian PRATT en 2003, en se basant sur un projet de recherche de l'université de Cambridge. La première version de Xen, sortie en 2003, était limitée dans ses performances et ses fonctionnalités. Elle a toutefois réussi à attirer des contributeurs et des investisseurs, permettant d'amorcer le développement.

La version 2 du projet a été diffusée fin 2004, et a apporté beaucoup d'améliorations, tant en terme de fonctionnalités (support des noyaux récents, couche réseau plus flexible, etc.) qu'en terme de performances. La version 2 a été maintenue jusqu'au milieu de l'année 2005, date de sortie de la version 2.0.6.

En parallèle, la communauté avait déjà commencé à développer la nouvelle version majeure du projet, la version 3.0. Elle a été diffusée début 2006, et a apporté davantage de fonctionnalités, notamment le support de plusieurs processeurs dans les domaines utilisateurs, améliorant sensiblement les performances. La version 3.0.2 a ajouté le support des instructions de virtualisation pour les processeurs le permettant, offrant la possibilité d'exécuter des systèmes invités au dessus de l'hyperviseur, sans avoir à les modifier.

En mai 2007 est sortie la version 3.1 de Xen, nouvelle version majeure améliorant encore le support des instructions de virtualisation et les performances en général.

Le projet Xen dispose en outre d'une feuille de route *(roadmap)* très complète, qui donne une idée très précise de la direction que vont prendre les développements à moyen et à long terme. Le fait qu'elle soit respectée et tenue à jour inspire confiance dans l'avenir de Xen, car on voit qu'il y a une réelle envie de progresser, et pas seulement des ajouts de fonctionnalités au hasard, selon l'humeur des développeurs.

3.1.2 Analyse détaillée

Xen est un projet très complexe, constitué de plusieurs composants visant à fournir une solution de virtualisation utilisable très simplement par l'administrateur. Cette simplicité apparente cache toutefois une complexité technique non négligeable, qu'il est souhaitable de maîtriser — ou au moins de connaître — si l'on veut pouvoir exploiter Xen au maximum de ses possibilités.

Il est de plus difficile d'obtenir une documentation technique pour connaître les détails de fonctionnement de Xen. En effet, la documentation utilisateur [XeM] est très succincte et ne couvre que les options de configuration, pas les détails d'implémentation. La documentation pour le programmeur [XeI] ne couvre que l'utilisation de l'*API*. Il est donc nécessaire d'effectuer un travail de recherche et de recoupement pour synthétiser toute la documentation disponible. L'analyse détaillée qui suit est donc une synthèse, d'après des informations tirées soit du wiki non officiel de Xen, soit de discussion avec la communauté des utilisateurs de Xen.

L'analyse détaillée de Xen portera sur les versions 2 et 3 du projet. Les évolutions apportées par la version 3 n'ayant pas fortement modifié l'architecture globale du logiciel, la plupart des documentations techniques portant sur la version 2 sont applicables à la version 3.

Le projet Xen peut être séparé en plusieurs modules :
– l'hyperviseur ;
– les *patches* à appliquer au noyau Linux ;
– les programmes de contrôle en espace utilisateur.
La figure 3.1 représente l'architecture générale de Xen. Par rapport à l'architecture théorique d'un système à hyperviseur (cf. figure 1.4 page 20), on peut noter la présence d'un système invité particulier, dénommé domaine 0. Ce système manipule l'hyperviseur Xen à travers une interface standardisée.

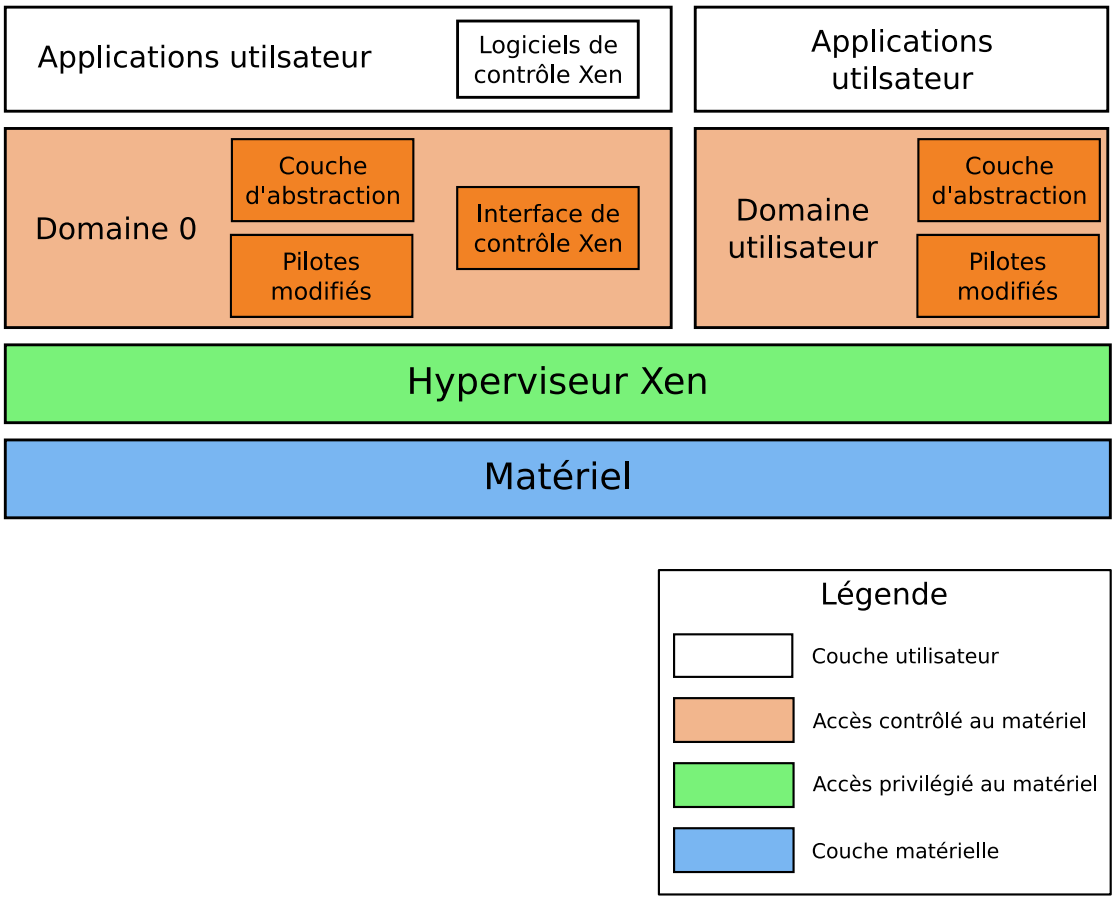

FIG. 3.1 – Architecture de Xen

Toutes les actions d'administration passent par l'interface de programmation *(API, Application Programming Interface)* de l'hyperviseur Xen. Ainsi, quand l'administrateur souhaite démarrer un domaine utilisateur, il exécute un utilitaire dans l'espace utilisateur, qui fait appel à l'interface de contrôle du domaine 0. Cette dernière communique alors directement avec l'hyperviseur pour instancier le domaine utilisateur.

Seul le domaine 0 a un accès complet au matériel (mais toujours à travers l'hyperviseur). C'est donc dans le domaine 0 que s'effectue toute la configuration des systèmes invités :
– réservation de l'espace disque ;
– création du fichier de configuration du domaine utilisateur ;
– création ou copie des fichiers du système d'exploitation...
Inversement, les domaines utilisateurs n'ont accès qu'à ce que l'administrateur a configuré (c'est à dire les partitions du système, plus éventuellement des lecteurs partagés sur le réseau).

L'hyperviseur

L'hyperviseur est démarré directement par le chargeur de démarrage de l'ordinateur, en lieu et place d'un système d'exploitation traditionnel. L'hyperviseur instancie ensuite le domaine 0 et lui passe la main pour le reste de l'initialisation des périphériques, tâche qui n'est pas réalisée par l'hyperviseur Xen.

En pratique, au démarrage de la machine, on voit s'afficher quelques lignes traitant de l'hyperviseur et de sa configuration avant que le noyau Linux ne prenne le relais et entame le processus de démarrage « traditionnel ».

Comme pour le noyau Linux, il y a une partie de la configuration de l'hyperviseur qui doit être passée par le chargeur de démarrage, soit parce qu'elle n'est pas modifiable en cours d'exécution (comme par exemple la quantité de RAM à affecter au domaine 0 ou les paramètres de la console série), soit parce que la lecture dans un fichier de configuration est impossible car le système de fichiers n'est pas encore activé au moment où l'information est nécessaire. Dans la configuration du chargeur de démarrage, l'hyperviseur apparaît comme un système d'exploitation de type GNU/Linux, avec simplement des options différentes de celles passées à un noyau Linux.

Techniquement, l'hyperviseur est le point de passage obligatoire pour tout accès au matériel. Il régule et répartit les accès aux ressources entre les systèmes invités (domaine 0 et domaines utilisateurs). À l'image des ordonnanceurs du noyau Linux pour la répartition des accès au processeur et aux périphériques d'Entrées/Sorties entre les différents processus du système d'exploitation, les ordonnanceurs de l'hyperviseur Xen répartissent les Entrées/Sorties et le temps processeur entre les systèmes invités. Il y a donc un niveau d'indirection supplémentaire, rendu nécessaire par la cohabitation de plusieurs systèmes d'exploitation sur la même machine.

Dans Xen, la mémoire physique installée est répartie sans recouvrement entre les systèmes invités. C'est à dire que si la machine dispose de 512 mégaoctets de RAM, l'administrateur pourra par exemple faire fonctionner 4 systèmes invités (incluant le domaine 0) disposant chacun de 128 mégaoctets. Il ne pourra pas allouer plus de mémoire qu'il n'y en a de disponible, la mémoire n'est pas une ressource « partageable » entre les systèmes d'exploitation. Il est par contre tout à fait possible d'arrêter un domaine utilisateur et d'en démarrer un autre à la place, du moment que le nouveau système ne nécessite pas plus de RAM qu'il n'en reste.

À l'inverse, l'accès au processeur n'est pas exclusif, tous les systèmes invités en cours d'exécution ont chacun une part du temps processeur total disponible sur la machine. Selon le type d'ordonnanceur choisi, l'administrateur peut assigner des priorités aux systèmes, afin d'affecter un quantum de temps plus important à un système par rapport aux autres.

En plus de gérer l'accès aux ressources, l'hyperviseur doit aussi gérer la correspondance entre la représentation de la mémoire des systèmes invités et la disposition effective de la mémoire pour le processeur. En effet, comme dit dans la section 1.2.2 page 14, c'est le processeur qui s'occupe de la gestion physique de la mémoire sur les architectures PC. Or, les systèmes d'exploitation ne sont pas conçus pour coopérer et tenir compte les uns des autres. C'est le rôle de l'hyperviseur de gérer la mémoire avec le processeur, et plus celui des systèmes d'exploitation, qui ne feront que communiquer avec l'hyperviseur pour cela.

De manière simplifiée, chaque système invité gère lui-même sa mémoire comme s'il était seul, dans un espace mémoire réservé, et tient au courant l'hyperviseur des opérations relatives à la gestion de la mémoire. L'hyperviseur est lui en charge de faire la corres-

pondance entre l'adresse virtuelle du système invité et l'adresse réelle manipulée par le processeur.

Le noyau Linux et l'hyperviseur communiquent par le biais d'appels systèmes spécifiques, appelés *hypercalls* (pour *hypervisor calls*, appels à l'hyperviseur). Ces appels permettent de passer des messages à l'hyperviseur, sur les tâches à accomplir (par exemple l'allocation ou la libération de mémoire).

Modifications apportées au noyau Linux

La plupart des modifications apportées au noyau Linux portent sur l'ajout de paravirtualisation pour la coopération avec l'hyperviseur, notamment pour la gestion de la mémoire, qui concentre la majorité des modifications.

Les *patches* ont pour effet de rajouter une architecture matérielle dans les options de compilation du noyau : xen_x86. Cette architecture matérielle, « fictive » car elle ne correspond pas réellement à du matériel, permet de regrouper toutes les modifications, au lieu de modifier directement les fichiers concernés pour l'architecture x86, opération très intrusive.

La communication avec l'hyperviseur s'effectue au moyen d'*hypercalls*. Il y a en tout une trentaine d'*hypercalls* définis, couvrant toutes les opérations courantes de Xen : gestion de la mémoire, gestion des Entrées/Sorties, etc.

Si la communication depuis le système invité vers l'hyperviseur est réalisée par un *hypercall*, le passage d'informations dans le sens inverse est réalisé par un bus d'évènements, au fonctionnement très similaire à celui des interruptions matérielles. Dès qu'un évènement susceptible de nécessiter une action de la part du système survient (arrivée d'un paquet sur l'interface réseau, fin d'un transfert de données sur le disque, ...), il est signalé au système, qui prend alors le relais et traite l'évènement. Les bus d'évènements de Xen ont le même rôle, mais pour la transmission d'informations relatives à l'hyperviseur (par exemple domaine utilisateur créé ou arrêté avec succès).

Les modifications apportées couvrent donc la gestion des messages de l'hyperviseur Xen, les *hypercalls* et l'adaptation à l'hyperviseur.

Applications en espace utilisateur

Les applications utilisateurs sont des programmes permettant de contrôler l'exécution des domaines utilisateurs. Ils se situent exclusivement dans le domaine 0.

L'application principale de l'espace utilisateur est `xend` (pour *Xen daemon**). `xend` est démarré en tant que service avec le système d'exploitation et sert d'interface entre les *hypercalls* de l'hyperviseur et les outils de contrôle de Xen. Il est aussi chargé de faire passer les informations du bus d'évènement vers l'espace utilisateur.

La figure 3.2 représente l'architecture de Xen. Les applications utilisateurs communiquent avec `xend` sous le mode client/serveur, `xend` assurant le rôle du serveur. La communication avec le noyau s'effectue sous la forme d'appels systèmes traditionnels Unix. Le noyau fait alors un *hypercall* en fonction de l'appel système et des paramètres reçus.

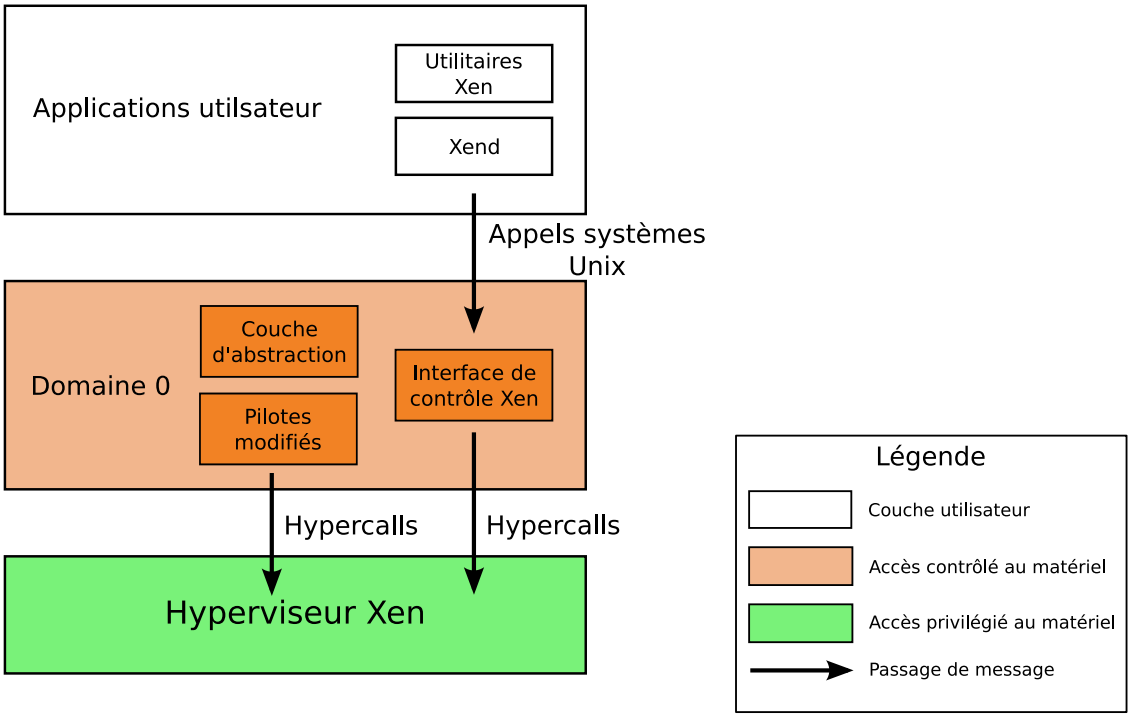

FIG. 3.2 – Enchaînement des appels systèmes au sein de Xen

L'empilement de couches d'abstraction est ici une bonne chose, car cela évite de lier la partie utilisateur à l'interface de programmation de l'hyperviseur. Avec des couches

d'abstraction entre eux, les composants de Xen peuvent évoluer séparément, il suffit de maintenir une simple interface entre les couches.

Les autres outils de gestion sont principalement commandes servant d'interface à `xend`, permettant de démarrer ou d'arrêter les domaines utilisateurs, ou encore de surveiller leur état (notamment la consommation mémoire et le temps processeur), un peu à la manière de la commande `top` de GNU/Linux.

Configuration

Xen stocke la configuration des domaines utilisateurs dans des fichiers de configuration, qui sont de simples fichiers textes éditables par l'administrateur.

Le fichier de configuration d'un domaine utilisateur comprend toutes les informations nécessaires à Xen pour pouvoir démarrer le système :
– le noyau à démarrer ;
– l'emplacement des fichiers du système invité (partition dédiée, LVM, répertoire, etc.) ;
– la quantité de mémoire vive à allouer ;
– l'adresse IP de la machine.
Avec ces informations et les fichiers du système invité, Xen est à même de démarrer un domaine utilisateur.

La configuration de l'hyperviseur se fait quant à elle soit dans le fichier de configuration du chargeur de démarrage, soit dans un fichier de configuration lu par `xend` au démarrage du domaine 0. En dehors de la configuration initiale lors de l'installation de Xen sur une machine, il n'est pas nécessaire de toucher à la configuration de l'hyperviseur.

3.1.3 Cas d'utilisation

Comme énoncé dans le paragraphe consacré aux fonctionnalités de Xen (section 2.7.3 page 56), Xen propose soit de dédier une partition à un système invité, soit de placer l'image du système dans l'arborescence du domaine 0. En pratique, cela implique peu de différences, car même dans le cas d'une partition dédiée, il est facile d'y accéder depuis le domaine 0, de la même façon qu'on accède à une clef USB ou un disque externe. Et,

de la même façon qu'avec une clef USB ou un disque externe, l'attribution à un système invité d'une partition distincte de celle de l'hôte permet de déplacer et de sauvegarder facilement le système invité.

Ce système, couplé à une gestion de haut niveau des partitions avec la technologie LVM (*Logical Volume Manager*, gestionnaire de volumes logiques) permet de s'affranchir des disques physiques pour gérer les partitions accessibles par les systèmes d'exploitation. En effet, avec une couche d'accès aux disques utilisant LVM, il est possible d'agréger plusieurs disques en un seul espace de stockage, partitionnable librement et indépendamment de la configuration des disques. Par exemple, avec deux disques de 40 gigaoctets, il est possible par le biais de la couche LVM de créer une partition de 60 gigaoctets et un partition de 20 gigaoctets. Ce principe permet de faciliter grandement la gestion de l'espace disque pour un système d'exploitation, offrant la possibilité de créer autant de partitions que nécessaire en rajoutant simplement des disques à la machine. On peut alors combiner ces partitions avec du RAID logiciel, pour rajouter de la tolérance aux pannes matérielles.

La combinaison LVM plus RAID logiciel est à la fois souple, performante et fiable. C'est d'ailleurs la configuration recommandée par la communauté Xen pour un serveur en production.

L'instanciation du système invité est effectuée via l'utilitaire `xen-create-image`. Cet utilitaire, développé par un membre de la communauté, permet d'automatiser tout le processus de création du système invité, tant sur la partie récupération des fichiers et des programmes nécessaires pour avoir un système utilisable que sur la partie configuration de Xen. L'avantage de `xen-create-image` est sa grande flexibilité au niveau des options de création : on passe toutes les options au lancement du programme et il procède ensuite automatiquement à la création des fichiers nécessaires. Quand le programme se termine, le nouveau domaine utilisateur est prêt à être démarré.

La flexibilité de `xen-create-image` nous permet de créer très rapidement des nouveaux domaines utilisateur, en changeant simplement quelques paramètres lors de l'appel de la commande.

Pour un serveur web « typique », destiné à héberger des applications en PHP avec une base de données MySQL (ce qui représente l'écrasante majorité des applications web),

l'utilisation de Xen au sein du processus d'hébergement est faite comme suit :

1. Détermination des caractéristiques du système invité (espace disque, mémoire vive nécessaire, etc.) en fonction de celles de l'application (nombre d'utilisateurs, charge système, etc.) ;
2. Allocation de l'espace disque sur un système hôte disponible ;
3. Allocation de la mémoire sur le système hôte ;
4. Instanciation du système invité par l'utilitaire `xen-create-image` ;
5. Démarrage du nouveau domaine utilisateur ;
6. Déploiement de l'application.

Avec cet utilitaire, l'utilisation de Xen pour l'hébergement de serveurs et le déploiement rapide de serveurs virtuels devient très simple. C'est un des gros avantages de Xen.

La section suivante sera consacrée à une analyse poussée du projet KVM, sur le même modèle que celle menée sur Xen.

3.2 Étude approfondie de KVM

3.2.1 Historique du projet

Comparativement à Xen, le projet KVM est beaucoup plus récent. La première version publique date en effet de la fin de l'année 2006, avec une intégration au noyau Linux dès le début de l'année 2007. Les premières versions publiques étaient déjà quasiment fonctionnelles, les nouvelles versions ont simplement étendu les fonctionnalités.

À la fin du mois de février 2007, peu après l'intégration au noyau, la migration « à chaud » a été ajoutée, permettant de déplacer un système invité d'une machine physique à une autre avec une très courte interruption, sans devoir l'arrêter.

Après le mois de février a suivi une longue phase de maturation et de correction de *bugs*, visant à stabiliser le projet et à pouvoir séparer le module noyau du code en espace utilisateur. L'interface entre les deux a été déclarée stable durant le mois de mai 2007.

La société Qumranet, qui finance le projet, devrait présenter à la fin du mois de septembre son premier produit commercial, dont on ne connaît l'instant rien, si ce n'est qu'il aura un rapport avec la virtualisation.

3.2.2 Analyse détaillée

Le projet KVM se compose de deux grandes parties : un module noyau et un programme utilisateur.

Module noyau

Le module noyau de KVM sert à communiquer avec le processeur et à utiliser les instructions de virtualisation. Les détails concernant ces instructions et les rôles respectifs du processeur et de la machine virtuelle pourront être trouvés dans [NSL+06] et [AMD06].

Concrètement, le module noyau sert à faire le relais entre la machine virtuelle, en espace utilisateur, le processeur et le système d'exploitation. La compréhension de son fonctionnement n'est pas nécessaire pour maîtriser l'utilisation de KVM.

Programme utilisateur

Le programme en espace utilisateur, sobrement dénommé `kvm`, est une version modifiée de QEMU, étendue pour pouvoir faire appel au module noyau.

Son fonctionnement général est identique à celui de QEMU, c'est à dire que c'est une machine virtuelle qui interprète et traduit les instructions du système invité. Toutefois certaines instructions ne sont plus traduites mais gérées directement par le processeur, améliorant ainsi les performances.

3.2.3 Cas d'utilisation

Sous KVM, la gestion des systèmes invités et totalement différente de celle de Xen. En effet, ici le système invité est installé dans une image disque complète, qui est un seul et unique fichier pour le système hôte.

Pour déployer un nouveau système invité, on effectue donc une installation « standard », comme si on installait un système d'exploitation sur une machine physique. L'image disque ainsi créée est ensuite utilisable sur n'importe quelle autre machine physique disposant également de KVM [1].

Afin d'éviter d'avoir à effectuer le processus d'installation à chaque fois que l'on a besoin d'un nouveau système, ce qui serait fastidieux, il est recommandé de stocker des images systèmes qui serviront de référence pour chaque nouveau système à déployer. Ainsi, on conserve sur un serveur des systèmes invités fraîchement installés, qu'on peut simplement copier sur la machine qui devra l'héberger en quelques minutes. Avec ce système, le déploiement consiste à copier l'image sur la bonne machine physique, démarrer le système invité, installer les programmes nécessaires puis installer l'application.

La configuration de KVM s'effectue par le biais des options de la ligne de commande, au moment de l'appel à `kvm`. C'est de cette manière que l'on décide de la quantité de mémoire vive à allouer au système, des périphériques supplémentaires à activer, de la configuration réseau, etc.

`kvm` exécute un script pour la configuration réseau, notamment pour la mise en place du pont entre les interfaces réseau, car il peut y avoir plusieurs méthodes pour cela, selon les besoins.

Le mode de fonctionnement *snapshot* (capture) de KVM permet de démarrer un système invité en lecture seule, sans altération de l'image système. Toutes les modifications de fichiers effectuées au sein du système invité sont uniquement conservées en mémoire, sans écriture sur le disque. Cela permet donc de démarrer un système invité, d'effectuer une action potentiellement destructrice (installation d'un programme inconnu, suppression de fichiers de configuration, etc.) et de revenir à l'état initial du système invité en

1. Il y a cependant quelques problèmes de compatibilité entre les processeurs AMD et Intel, ainsi qu'entre les architectures 32 et 64 bits, mais ils sont en cours de résolution.

redémarrant simplement le programme. Ce mode est extrêmement pratique quand on souhaite rapidement tester quelques modifications, sans avoir à copier une image système complète car le test est temporaire.

Une fonctionnalité complémentaire au mode *snapshot* est le mode *overlay* (surcouche). Avec ce mode de fonctionnement, très similaire au *snapshot*, les modifications apportées aux fichiers sont stockées dans un fichier supplémentaire, qui ne conserve donc que les éléments différents du fichier image original. On peut ainsi configurer deux systèmes séparément tout en conservant la même base. Ce mode de fonctionnement est très intéressant pour économiser l'espace disque, surtout lorsque les différences entre les systèmes sont mineures (par exemple, des détails de configuration d'une application).

3.3 Bilan

KVM et Xen sont très différents, techniquement parlant. Toutefois, pour l'utilisateur final du service (site web), la différence entre un serveur virtuel — quel qu'il soit — et une machine physique est inexistante, excepté au niveau des performances. Le choix se fait donc essentiellement sur le niveau de performances atteint. Il ne faut cependant pas négliger la facilité d'utilisation et d'administration de la solution.

3.3.1 Xen

La complexité de Xen, ainsi que son manque de documentation, sont des facteurs qui pourront se retourner contre le projet, à long terme. En effet, il est difficile dans cette situation d'attirer les contributeurs extérieurs, la période d'apprentissage nécessaire pour maîtriser le code source étant très longue (de l'ordre de quelques mois).

Il a fallu effectuer un véritable travail de recherche et de synthèse pour obtenir une bonne idée du fonctionnement interne de Xen, car les documents disponibles sont rarement à jour, et sont même parfois répartis sur plusieurs sites (site officiel, wiki, site de l'université de Cambridge, etc.). Cette complexité est un gros défaut si l'on souhaite un jour modifier le comportement de Xen. La documentation existante, notamment le

manuel utilisateur, renseigne plus sur des détails de configuration de Xen que sur comment Xen fonctionne. Or, il est pour nous indispensable que l'on sache comment se il comporte dans une situation donnée.

De plus, du fait même de la technologie utilisée par Xen, un modèle à hyperviseur, le projet est condamné à rester en grande partie hors du noyau Linux, seules les modifications suffisamment génériques et peu intrusives ont une chance d'être intégrée au noyau, comme cela a été le cas au mois d'août 2007.

L'utilisation d'un hyperviseur a un autre effet : Xen doit réimplémenter tous les algorithmes d'ordonnancement entre les systèmes invités. En effet, l'hyperviseur étant le logiciel qui contrôle l'accès aux ressources (processeurs, mémoire vive, disques durs, cartes réseau, etc.), c'est à lui que revient la tâche de répartir les accès de manière équitable. C'est une tâche très complexe, qui est toujours la source de nombreux *bugs*. Ce travail d'ordonnancement des tâches est déjà effectué par le noyau Linux au niveau des processus du système d'exploitation, et Xen doit refaire le même travail pour les systèmes d'exploitation. Dans le noyau Linux, les modules d'ordonnancement sont l'objet d'années de développement et d'optimisation, il est à craindre que le projet Xen consacre une grande partie de ses efforts de développement à dupliquer les mêmes fonctionnalités pour simplement pouvoir faire cohabiter plusieurs systèmes d'exploitation sur la même machine.

Du côté communautaire, le récent rachat de la société XenSource par Citrix inquiète beaucoup la communauté, qui craint de voir XenSource se tourner de plus en plus vers un modèle propriétaire similaire à celui de Virtuozzo, la version libre devenant un simple produit d'appel, aux performances inférieures à la version propriétaire.

Il y a eu par le passé quelques tentatives de se tourner vers un modèle commercial plus rentable de la part de certains projets libres. On peut notamment citer le détecteur d'intrusions Snort, qui se sert d'une base de signatures pour repérer le trafic suspect. Les signatures sont véritablement le cœur de Snort, ce qui fait toute la valeur du projet. Il y a quelques mois, l'accès aux signatures les plus récentes est devenu payant, avec interdiction de les redistribuer. En réaction à cette fermeture, un projet parallèle, nommé *bleeding Snort* (jeu de mots avec *bleeding edge*, qui signifie « à la pointe ») s'est créé, et diffuse une base de signatures libre, et ce très régulièrement.

Cet épisode, loin d'être isolé, montre bien que les projets doivent faire très attention à l'image qu'ils diffusent auprès de leur communauté, sous peine de voir cette dernière se retourner contre le projet.

La société XenSource devra donc faire attention à ne pas s'aliéner la communauté et les contributeurs, lors des prochaines évolutions de Xen.

En définitive, Xen souffre de quelques problèmes, notamment au niveau de la documentation technique disponible et de la grande complexité du code source. Il faudra surveiller de près le comportement de XenSource par rapport au code source du projet, pour pouvoir anticiper tout changement dans la politique de licence.

3.3.2 KVM

Comparativement à Xen, le projet KVM a l'air plus intéressant, tant au niveau de la simplicité d'utilisation que de la complexité du code source.

En effet, étant donné que KVM est un simple processus au sein du système hôte, il n'a pas à réinventer tous les algorithmes d'ordonnancement pour obtenir de bonnes performances. La différence peut se résumer simplement par la citation suivante :

> « *We get for free what Xen works hard to achieve.* (Xen doit travailler dur pour obtenir ce dont nous bénéficions gratuitement.) »
>
> Dor LAOR, développeur de KVM

Dor LAOR met ici l'accent sur la différence primordiale entre Xen et KVM : comme KVM s'exécute au sein du système hôte, il bénéficie des fonctionnalités du noyau Linux en ce qui concerne la gestion des processus et des accès aux ressources, alors que Xen doit tout redévelopper.

On retrouve une différence au niveau de la communauté : les contributions externes sont plus nombreuses pour KVM que pour Xen, en dépit de la jeunesse du projet KVM. La différence peut facilement s'expliquer par la jeunesse du projet, la nouveauté étant toujours très attractive, ainsi que par la simplicité — par rapport à Xen — du code source.

De plus, le risque de changement de licence est très faible, et quasiment sans risque pour la pérennité du projet. En effet, maintenant que le code du module KVM est intégré au noyau Linux, il sera toujours diffusé avec, et si Qumranet vient à changer la license du projet, il restera toujours la partie libre intégrée au noyau, que n'importe qui pourra reprendre pour en faire un nouveau projet libre.

Toutefois, les performances sont encore à l'heure actuelle en deçà de ce que Xen offre, même s'il y a récemment eu des améliorations en ce sens. Les performances sont le dernier facteur limitatif de KVM pour l'utilisation en production et l'hébergement de serveurs. Sa grande simplicité lui vaut d'être déjà utilisé par beaucoup de monde pour les tests d'installation et l'utilisation bureautique. Il ne manque pas grand chose pour que le monde de l'hébergement se tourne vers KVM, il y a déjà des discussions sur la liste de discussion parlant d'hébergements à base de KVM.

Conclusion

Les différentes solutions de virtualisation existantes utilisent des technologies variées, en fonction des buts du projet. Certaines technologies permettent de faire cohabiter plusieurs systèmes d'exploitation, d'autres cloisonnent un unique système en plusieurs compartiments indépendants. Certaines s'appuient sur les capacités du matériel pour améliorer les performances alors que d'autres nécessitent un système d'exploitation modifié pour cohabiter avec la solution de virtualisation. Ces technologies ont toutes leurs avantages et inconvénients, et il est important de faire le bon choix en fonction de l'utilisation que l'entreprise en fera.

La virtualisation est un domaine en pleine croissance, qui évolue très rapidement. Les entreprises peuvent se servir de la virtualisation pour différents usages, aux besoins variés.

Le premier secteur susceptible de bénéficier des apports de la virtualisation est celui du développement d'applications. En effet, la possibilité pour un développeur de tester rapidement les modifications, dans un environnement contrôlé, est un avantage considérable. Les besoins en performance sont relativement faibles ce cas d'utilisation, car les systèmes virtuels sont uniquement utilisés à des fins de tests. L'avantage sera ici donné à la simplicité d'utilisation de la solution.

Toutefois, ce n'est pas le seul domaine d'application de la virtualisation. Le secteur d'activité qui sera le plus amené à croître est sans aucun doute le marché de la consolidation de serveurs. En effet, les entreprises sont maintenant dans une phase de réduction des coûts. La consolidation de serveurs est précisément la solution dont les entreprises ont besoin pour économiser de l'argent sur la partie matérielle. Il faut néanmoins que les performances de la solution de virtualisation soient vraiment à la hauteur.

La consolidation de serveurs concerne principalement le domaine de l'hébergement de sites web, que cela soit pour des applications internes ou pour l'hébergement de sites clients. Les besoins dans ce cas sont simples : haute disponibilité, performances élevées, facilité d'administration et de configuration.

Parmi tous les produits étudiés dans ce livre blanc, aucun ne satisfait parfaitement aux besoins énumérés ci-dessus. Cependant, certaines solutions s'en approchent sensiblement, et sont déjà utilisées en production par de nombreuses entreprises.

C'est notamment le cas de Xen, qui n'a comme inconvénient technique que sa complexité et son manque de documentation. La communauté de Xen est certes active, mais moins que ce que l'on aurait pu espérer compte tenu du nombre d'utilisateurs du produit. De plus, le rachat récent de XenSource (principal contributeur du projet libre Xen) par Citrix, une société qui ne fait absolument pas de logiciel libre, peut causer quelque inquiétudes. Le risque de voir le projet se tourner vers du propriétaire est à envisager.

Un autre projet à surveiller de près est KVM, qui — s'il est pour l'instant clairement moins performant que Xen — a un potentiel de croissance très élevé. En effet, KVM est bien plus simple à utiliser que Xen, son seul défaut majeur restant les faibles performances obtenues actuellement. Cela est cependant en train de changer, et KVM pourrait bien rattraper Xen d'ici quelques mois.

Toutefois, il ne faut pas s'arrêter aux seules fonctionnalités du produit pour faire le bon choix, les utilitaires annexes sont également très importants. Ces outils sont de nature diverse, servant parfois à simplifier le processus de création de systèmes invités, ou encore à permettre de gérer plus facilement les différents systèmes invités installés. De ce côté là, Xen dépasse KVM, même si ce dernier est là encore en train de refaire son retard.

Parmi les outils annexes, un des domaines importants est celui de la supervision. En effet, si l'on commence à avoir beaucoup de serveurs virtuels, notamment dans des environnements hétérogènes et dispersés géographiquement, une solution permettant de centraliser la supervision et l'administration des serveurs virtuels devient très intéressante.

En plus des solution propriétaires proposées par XenSource aux clients choisissant une version « Entreprise » de Xen, il y a plusieurs projets libres visant à fournir une interface de supervision centralisée performante pour Xen. On peut notamment citer XenMan, enomalism, DTC-Xen, etc. Ces solutions permettent toutes de gérer un grand nombre d'instances de Xen de manière centralisée, sans avoir à se connecter sur chaque domaine 0 pour contrôler l'état des domaines utilisateurs. On peut donc administrer tous ses serveurs Xen depuis une seule application, sur un seul ordinateur, sans avoir à se connecter à chaque serveur.

Le projet KVM dispose quant à lui de bien moins de solutions de supervision. Il y a toutefois un projet qui mérite d'être mentionné : *Virtual Machine Manager*. Ce projet vise à offrir une interface de contrôle unifiée pour différentes solutions de virtualisation (notamment QEMU, KVM et Xen). Ce projet permettra à terme de centraliser la gestion de plusieurs solutions de virtualisation au sein de la même application.

Avec Xen et KVM, accompagnés d'outils de supervision et d'administration adaptés, les entreprises disposent de solutions utilisables en production pour virtualiser des serveurs. Il est donc tout à fait envisageable à l'heure actuelle d'utiliser une architecture complète de virtualisation constituée uniquement de logiciels libres.

Il faut toutefois garder à l'esprit que la virtualisation n'est pas la solution miracle à tous les problèmes d'infrastructure. En effet, une architecture de virtualisation mal pensée coûtera certainement plus cher à l'entreprise que l'architecture « une machine par système d'exploitation » en place dans la plupart des sociétés. Il faut acquérir des compétences suffisantes avant de mettre en place des solutions de virtualisation.

La concentration de plusieurs systèmes d'exploitation sur une seule machine accroît en outre le risque de pertes financières en cas de panne matérielle, car plusieurs applications seront touchées. Si en plus la société ne dispose pas d'une machine de secours aussi puissante que la machine en panne, elle risque même de perdre des clients.

Si l'entreprise prend en compte tous ces risques, et a les ressources nécessaires pour répartir la charge des applications sur plusieurs machines, avec des serveurs de secours pour minimiser les risques d'impacts en cas de panne ; alors la virtualisation peut apporter une réactivité permettant de se démarquer dans le milieu concurrentiel de l'hébergement.

Le marché de la virtualisation est encore très jeune, et tous les acteurs préparent d'ores et déjà de nouvelles évolutions, attendues d'ici quelques semaines ou quelques mois.

Par exemple, les instructions de virtualisation des processeurs Intel et AMD vont très prochainement être mises à jour, pour rajouter davantage de fonctionnalités et améliorer un peu les performances. De même, un nouveau jeu d'instructions va bientôt faire son apparition sur les processeurs Intel et AMD, pour l'accélération des Entrées/Sorties dans une machine virtuelle. Les projets qui sauront tirer parti de ces nouvelles instructions dès leur disponibilité sur le marché prendront très certainement l'ascendant sur leurs concurrents.

Les éditeurs propriétaires sont aussi en train de préparer de nouvelles versions de leurs produits, que ce soit du côté de VMware ou de Microsoft. Ils espèrent apporter une réponse au succès grandissant des projets libres, en se focalisant notamment sur les facilités de supervision et d'administration de leurs solutions ainsi que sur le support avancé des systèmes d'exploitation de la famille Windows.

Il est clair qu'il faudra suivre de très près l'activité du marché de la virtualisation et de ses acteurs dans les prochains mois, pour pouvoir anticiper les mouvements et faire les bons choix stratégiques à long terme.

En plus des aspects purement économiques liés à l'activité de l'entreprise, la virtualisation touche aussi d'autres domaines, tout aussi susceptibles d'intéresser les entreprises. C'est notamment le cas de l'écologie. En effet, il y a depuis quelques temps un regain d'intérêt du grand public et de l'industrie pour l'écologie et la réduction du gaspillage. Or, la virtualisation et le logiciel libre ont tous les deux un rôle à jouer dans ce domaine.

Le logiciel libre peut souvent être considéré comme plus écologique que le logiciel propriétaire. En effet, avec le logiciel propriétaire, on retrouve souvent des emballages, des manuels imprimés, des CD-ROM ou des DVD-ROM, des encarts publicitaires, etc. Une simple visite des rayons consacrés aux logiciels dans les magasins d'informatique suffit pour s'en convaincre. Il est difficile, voire souvent impossible, de se procurer des versions dématérialisées (c'est à dire juste un ensemble de fichiers numériques) de ces logiciels. Tout cela a un fort impact écologique, car la fabrication de tous ces matériaux nécessite des matières premières qui ne sont pas toujours renouvelables.

Au contraire, avec le logiciel libre l'impact écologique est bien plus faible, car il est tout à fait possible de ne récupérer qu'une version numérique du produit, que l'on pourra graver sur un CD-ROM ou un DVD-ROM seulement si on le souhaite. Il en va de même pour la documentation, disponible par défaut sous forme numérique. C'est pour cela qu'on qualifie parfois le logiciel libre d'« éco-logiciel », car la distribution des logiciels libres est dans la plupart des cas bien plus respectueuse de l'environnement que la distribution des logiciels propriétaires.

Toujours dans le cadre d'une entreprise plus respectueuse de l'environnement, la consolidation de serveurs par le biais de la virtualisation est là aussi un facteur important. Une enquête interne de HP a ainsi révélé que le taux moyen d'utilisation de ses serveurs était largement inférieur à 50 %, et descendait même jusqu'à 10 % dans certains services. Alimenter en énergie des machines sous-utilisées est très coûteux. Les ordinateurs sont en effet des appareils électroniques qui consomment énormément d'énergie, dont la plus grande partie est dissipée en chaleur ou rayonnements. Le rendement d'un ordinateur est donc très faible.

La consolidation permet dans ce cas d'augmenter sensiblement le taux d'utilisation des serveurs en réduisant le nombre total de machines physiques. Le seul fait de réduire le nombre de serveurs utilisés a un impact positif sur la nature : l'entreprise réalise des économies d'énergie, qui se répercutent sur la consommation électrique.

En plus de sa consommation électrique élevée, un ordinateur est également constitué de produits très polluants, voire toxiques (comme par exemple du mercure). Ces produits sont souvent difficiles à extraire des composants pour le recyclage. La fabrication de certains composants a en outre un impact très fort sur la nature, en terme de consommation de ressources naturelles, notamment l'eau.

Il apparaît maintenant clair qu'utiliser un ordinateur a un coût écologique non négligeable, et que la virtualisation peut aider à minimiser ce coût en permettant de réduire le nombre de machines physiques nécessaires au fonctionnement de l'entreprise.

La consommation électrique et la fabrication des machines ne sont toutefois pas les seuls éléments à prendre en compte dans le bilan écologique global. En effet, les serveurs sont souvent rassemblés dans des *datacenters*. Or ces salles sont elles aussi fortement sujettes au gaspillage et à la surconsommation. Par exemple, l'écrasante majorité des *datacenters*

maintiennent une climatisation à 21° C ainsi qu'un éclairage permanent de la pièce. Une étude de l'institut Gartner a récemment annoncé que la plupart des datacenters, même les plus récents, n'étaient pas aux normes écologiques bientôt en vigueur, et qu'il faudrait les reconstruire très prochainement.

Les solutions libres de virtualisation ne sont pas de simples versions gratuites et moins performantes des versions propriétaires. Les logiciels libres ont dans ce domaine une crédibilité et un potentiel d'innovation très importants. Dans les mois à venir, les solutions libres vont encore faire preuve d'innovation et vont probablement évoluer plus vite que les solutions propriétaires.

Glossaire

D

daemon Terme anglais désignant un processus s'exécutant en tâche de fond, sans interaction directe avec un utilisateur. Par exemple, un serveur Web s'exécute très souvent en tant que *daemon*, en étant uniquement à l'écoute des connexions réseau.

datacenter Aussi appelé « salle blanche » ou « centre de traitement de données », c'est une salle où sont entreposés les serveurs pour l'hébergement de services. La salle dispose en général d'une grande connectivité à Internet et d'une climatisation pour maintenir la température constante. La plupart du temps, la salle est gérée par un opérateur, qui loue de l'espace aux entreprises souhaitant une forte connectivité à Internet et des garanties.

E

émulation L'émulation consiste à utiliser un processus logiciel pour remplacer un processus normalement réalisé au moins en partie par le matériel. Par exemple, un émulateur de console de jeu est un programme qui *se comporte* comme une console de jeu, en émulant le matériel de la console. Pour la virtualisation, l'émulation intervient par exemple quand la machine virtuelle émule le matériel habituel d'un ordinateur pour le système invité (disque dur, clavier, écran, etc.).

L

legacy application Terme anglais désignant un logiciel relativement ancien par rapport aux autres logiciels de la société, mais toujours utilisé, en général pour des raisons historiques.

M

mainframe Aussi appelé « ordinateur central », ce terme désigne un ordinateur disposant d'une très forte puissance de traitement, du plusieurs ordres de grandeur supérieure à celle d'un ordinateur type PC.

malware Terme anglais générique désignant un logiciel programmé dans le but de nuire à une victime, comme par exemple un virus, un logiciel espion *(spyware)*, etc.

O

ordonnanceur Dans un système d'exploitation, l'ordonnanceur est le module noyau qui choisit les processus qui vont être exécutés par le système. Il est appelé plusieurs centaines de fois par seconde pour décider quel processus exécuter jusqu'à sa prochaine intervention. Il existe des ordonnanceurs pour les processeurs et pour les Entrées/Sorties.

P

patch Terme anglais désignant une section de code source que l'on ajoute à un logiciel pour en modifier le comportement. On parle aussi parfois de rustine ou de correctif.

R

RAID Acronyme anglais signifiant *Redundant Array of Independent Disks*, matrice redondante de disques indépendants. Les technologies RAID visent à utiliser plus de disques que nécessaire pour stocker les données. Il y a plusieurs types de RAID, et selon le type choisi on peut améliorer les performances ou la tolérance aux pannes. Ainsi, avec un système en RAID 5, le système devient tolérant à la perte d'un disque dur et peut continuer ses activités sans interruption de service.

W

wiki Un wiki est un site web collaboratif, dont le contenu est directement modifiable par les utilisateurs du site.

Index

Liste des tableaux

Table des figures

Table des matières

Références

[AFL] ALDIL, *Les fiches libres*.
URL `http://www.aldil.org/projets/fiches_libres/`.

[AJM+06] Darren ABRAMSON, Jeff JACKSON, Sridhar MUTHRASANALLUR, *et al.*, *Intel Virtualization Technology for Directed I/O*. *Intel Technology Journal*, août 2006, ISSN 1535-864X.
Un article (très) technique sur les évolutions à attendre en matière d'amélioration de performances pour la virtualisation. Le sujet traité est celui des Entrées/Sorties, qui sont généralement le goulet d'étranglement des machines virtuelles. Les auteurs décrivent les technologies développées par Intel à ce sujet, en vue d'améliorer grandement les performances.

[All07] Yann ALLANDIT, *Problématique de consolidation et atteinte des objectifs de niveau de service (SLO) avec Xen*. *GNU/Linux Magazine France*, (numéro 92), mars 2007.
Cet article décrit les enjeux de la consolidation de serveurs ainsi que les moyens de gérer les priorités entre plusieurs machines virtuelles Xen.

[AMD06] AMD, *AMD64 Architecture Programmer's Manual, Volume 2 : System Programming*, chapitre 15, pages 355–412. septembre 2006.
Ce chapitre décrit très en détail le fonctionnement des instructions de virtualisation d'AMD, nommées SVM. C'est l'équivalent en plus technique de l'article [NSL+06] du côté Intel.

[AMD07] AMD, *AMD I/O Virtualization Technology (IOMMU) Specification*. février 2007.
Ce document, à destination des programmeurs, décrit l'architecture utilisée par AMD pour virtualiser les Entrées/Sorties. C'est l'équivalent

de [AJM+06] côté Intel.

[BDF+03a] Paul R. BARHAM, Boris DRAGOVIC, Keir A. FRASER, *et al.*, *Xen 2002*. janvier 2003.
Ce document présente les principes et choix techniques ayant conduit au développement de la première version de Xen, en se basant sur un projet existant développé à l'université de Cambridge.

[BDF+03b] Paul R. BARHAM, Boris DRAGOVIC, Keir A. FRASER, *et al.*, *Xen and the Art of Virtualization*. octobre 2003.
Tout comme [Qum07] est le document de référence sur KVM, ce document est la référence ayant véritablement lancé le projet pour le grand public. Il suit et complète l'article [BDF+03a].

[Bel05] Fabrice BELLARD, *QEMU, a Fast and Portable Dynamic Translator*. USENIX Annual Technical Conference, 2005.
Un article de Fabrice BELLARD, le développeur principal et mainteneur de QEMU, qui présente les choix techniques et le fonctionnement interne du logiciel.

[Ker] KernelNewbies, *Linux Virtualization Wiki*.
URL `http://virt.kernelnewbies.org/FrontPage`.
Le wiki dédié à la virtualisation du site KernelNewbies contient des informations très utiles pour recenser les différences entre les technologies de virtualisation, ainsi que les buts visés par la virtualisation. Il est toutefois centré, comme son nom peut le laisser penser, sur les projets (libres) tournant sur la plate-forme GNU/Linux.

[KKL+07] Avi KIVITY, Yaniv KAMAY, Dor LAOR, *et al.*, *KVM : the Linux Virtual Machine Monitor*. Linux Symposium, 2007.
Cet article, tiré d'une conférence faite au Linux Symposium d'Ottawa, présente le projet KVM d'un point de vue technique.

[KKTS07] Kasem KHARBAT, Omar KHAN, Ovidiu TUDOSA, Shiraz SIDDIQUI, *Exploring the linux KVM*. 2007.
Cet article de recherche étudie l'architecture et le *design* de KVM, puis établit une comparaison avec une série de patches visant à introduire des

fonctionnalités supplémentaires à KVM, notamment au niveau des performances.

[MCZ06] Aravind MENON, Alan L. COX, Willy ZWAENEPOEL, *Optimizing Network Virtualization in Xen*. USENIX Annual Technical Conference, 2006.
Un article technique présentant plusieurs pistes pour l'optimisation du trafic réseau des machines virtuelles Xen, qui recoupe les recherches menées à ce sujet pour l'optimisation des Entrées/Sorties dans une machine virtuelle (voir aussi [AJM+06]).

[NSL+06] Gil NEIGER, Amy SANTONI, Felix LEUNG, *et al.*, *Intel Virtualization Technology : hardware support for efficient processor virtualization*. *Intel Technology Journal*, août 2006, ISSN 1535-864X.
Cet article (très technique) du *Intel Technology Journal* détaille très précisément l'apport des instructions de virtualisation pour l'accélération des performances des machines virtuelles.

[Orm07] Travis ORMANDY, *An empirical study into the security exposure to hosts of hostile virtualized environments*. 2007.
Ce court article de recherche analyse la sécurité de quelques solutions de virtualisation, dont certaines traitées dans mon mémoire. L'auteur analyse la robustesse et la résilience des applications testées.

[PZW+07] Pradeep PADALA, Xiaoyun ZHU, Zhikui WANG, *et al.*, *Performance evaluation of virtualization technologies for server consolidation*. avril 2007.
Un article du laboratoire de recherche HP *Enterprise Systems and Software Laboratory* qui compare les performances de Xen et de OpenVZ pour des cas d'utilisation typiques d'un serveur d'applications.

[Qum07] Qumranet Inc., *KVM : kernel-based virtualization driver*. 2007,
URL `http://www.qumranet.com/wp/kvm_wp.pdf`.
Ce *white paper* est l'article « fondateur » du projet KVM, diffusé en même temps que la première version du logiciel. Il expose les choix techniques faits pour implémenter KVM et les buts du projet.

[Sin06] Amit SINGH, *An introduction to virtualization*. 2006,
URL `http://www.kernelthread.com/publications/virtualization/`.

Article très complet sur la virtualisation, qui commence par un historique remontant assez loin, puis liste les problèmes que la virtualisation cherche à résoudre. L'auteur liste ensuite plusieurs technologies de virtualisation.

[WAc] Wikipédia anglophone, *comparaison de différentes machines virtuelles*.
URL `http://en.wikipedia.org/wiki/Comparison_of_virtual_machines`.
Ce tableau, sur le Wikipédia anglophone, recense les principales solutions de virtualisation et leurs fonctionnalitéss respectives, permettant une comparaison entre différents projets.

[WAh] Wikipédia anglophone, *article sur le principe de l'hyperviseur*.
URL `http://en.wikipedia.org/wiki/Hypervisor`.
Cet article de Wikipédia présente le principe de l'hyperviseur, son origine et ses applications actuelles.

[WAv] Wikipédia anglophone, *article sur la virtualisation*.
URL `http://en.wikipedia.org/wiki/Virtualization`.
Cet article de Wikipédia présente la virtualisation, de manière plus détaillée que son homologue francophone.

[WFl] Wikipedia francophone, *article sur le logiciel libre*.
URL `http://fr.wikipedia.org/wiki/Logiciel_libre`.
Cet article de Wikipédia définit en détail le principe du Logiciel Libre, des quatre libertés et des licences libres.

[WFv] Wikipédia francophone, *article sur la virtualisation*.
URL `http://fr.wikipedia.org/wiki/Virtualisation`.
Cet article de Wikipédia présente la virtualisation dans les grandes lignes, avec des liens vers des articles plus détaillés. C'est une bonne base pour commencer une recherche sur la virtualisation.

[WSG02] A. WHITAKER, M. SHAW, S. GRIBBLE, *Denali : Lightweight virtual machines for distributed and networked applications*. USENIX Annual Technical Conference, juin 2002,
URL `http://citeseer.ist.psu.edu/whitaker02denali.html`.
Cet article parle de la conception d'une machine virtuelle, Denali, et est le premier article à introduire la notion de para-virtualisation.

[XeI] Xen, *Interface manual*.
URL `http://wiki.xensource.com/xenwiki/XenDocs`.
Documentation sur l'implémentation de Xen, les choix techniques effectués, et les options avancées de Xen 3.0.

[XeM] Xen, *User manual*.
URL `http://wiki.xensource.com/xenwiki/XenDocs`.
Documentation utilisateur de Xen 3.0, qui liste quelques options de configuration et fonctionnalités propres à Xen.