utt
université de technologie
Troyes

Antoine Benkemoun SIT01
Romain Hinfray SIT05

Semestre Automne 2008

AC - Etude de la virtualisation et du fonctionnement de la solution libre Xen

Résumé :

A travers ce rapport établi dans le cadre d'une unité de valeur d'Acquisition de connaissances, nous établirons un tour d'horizon de la virtualisation à l'heure actuelle. Nous étudierons ensuite plus en profondeur la solution libre Xen en prenant soin de rappeler quelques notions de systèmes d'exploitation.

Projet encadré par :
Guillaume Doyen
Florent Retraint

SOMMAIRE

Terminologie

BSD	Berkeley Software Distribution
Control sensitive instructions	Instruction processeur altérant la configuration du matériel physique
Domaine de virtualisation	Appellation dans le cadre du projet Xen des systèmes d'exploitation invités
Emulation	Imitation du comportement physique d'un matériel par un logiciel
GDT	Global Descripton Table
Hot Spare	Serveur toujours sous tension servant à prendre le relai instantanément lorsque le serveur principal devient indisponible
HVM	Hardware Virtual Machine, machine virtualisée matériel assisté
LDT	Local Description Table
Network Attached Storage	Périphérique de stockage relié à un réseau
NUMA	Non Uniform Memory Access
MMU	Memory Management Unit
MV	Machine Virtuelle
Open Source	Dit d'un logiciel donc le code source est accessible et dont les modifications sont autorisées
Popek et Goldberg	Formal Requirements for Virtualizable Third Generation Architectures » publié dans Communications of the ACM
Privileged instructions	Instructions permettant d'accéder directement à des informations stockées dans des périphériques système ou de modifier la configuration du matériel
RAM	Random Access Memory
Registre	Emplacement mémoire interne au processeur
Storage Area Network	Réseau permettant de mutualiser des ressources de stockage par le biais des périphériques de stockages adressable par réseau par le biais de protocoles tels que le iSCSI

Système hôte	Système d'exploitation non virtualisé qui dispose des outils permettant d'exécuter les systèmes d'exploitation invites
System/360	Serveur d'applications lourdes (transaction financières, logiciels de gestion d'entreprise).
Virtualisation récursive	Principe par le biais duquel un système d'exploitation invité peut à son tour virtualiser d'autres systèmes d'exploitation.
zSeries	Dernière génération des mainframes d'IBM

Introduction générale

Nous avons établi ce rapport dans le cadre d'une acquisition de connaissances encadré par le département des Systèmes d'Informations et de Télécommunications de l'Université de technologie de Troyes et plus particulièrement par Guillaume Doyen et Florent Retraint. Ce rapport qui a été rédigé tout au long du semestre d'automne 2008 traite de la virtualisation ainsi que du projet Xen. La virtualisation est un sujet d'actualité pour les acteurs de l'industrie des systèmes d'informations. Etant donnée notre entrée imminente sur le marché de travail, acquérir des connaissances dans ce domaine constitue une valeur ajoutée à notre formation.

La virtualisation est applicable dans de multiples domaines des systèmes d'information. Cependant, nous ne traiterons ici uniquement de la virtualisation de systèmes d'exploitation.

Dans le cadre de la première partie, nous effectuerons un tour d'horizon de la virtualisation. Nous détaillerons les avantages ainsi que les techniques existantes.

Ensuite, nous clarifierons certaines notions relatives aux systèmes d'exploitation. Ces notions nous permettrons d'abord l'étude du projet Xen. Nous détaillerons la gestion mémoire, la gestion des informations du système ainsi que la gestion du temps de calcul.

Afin d'effectuer une étude approfondie d'une technologie en informatique, les solutions libres sont le compromis idéal entre technicité et accessibilité de l'information. Actuellement, un des projets libres les plus soutenus par les grands acteurs des systèmes d'information est le projet Xen. C'est pourquoi il constituait un support idéal pour notre étude.

1. Principe de la virtualisation

1.1. Introduction à la virtualisation

La virtualisation est une technique informatique consistant à faire fonctionner plusieurs environnements logiques indépendants séparément sur une même machine. Il s'agit d'une extension du principe d'émulation. L'émulation consiste à substituer un ou plusieurs éléments informatiques par une application. Appliquée à la virtualisation, un système prétend être plusieurs systèmes différents.

La virtualisation fait appel au multiplexage des systèmes d'exploitation comme on le trouve au niveau des processus. Différents processus cohabitent indépendamment tout en se partageant les ressources physiques de la machine. Un processus ne peut pas monopoliser toutes les ressources de calcul puisque dans ce cas là le système d'exploitation reprendra le contrôle pour allouer des ressources à d'autres processus. Au niveau des ressources de stockage, chaque processus a son espace virtuel d'adresse en mémoire lui donnant l'impression qu'il est seul à utiliser la mémoire vive.

1.2. Historique

Historiquement une grande partie des travaux de recherche ont été fait par IBM dans les années 1960 au centre de recherche de Grenoble aujourd'hui fermé. Ils développèrent un système expérimental faisant partie du projet System/360appelé VM/CMS (Virtual Machine / Conversation Monitor System). CMS est le système d'utilisation qui s'appuie sur VM. Une caractéristique de ce premier système de virtualisation était le fait que chaque CMS était attribué à un seul utilisateur, sachant que plusieurs CMS fonctionnaient sur la VM. Nous pouvons faire une analogie par rapport à la terminologie actuelle entre VM et hyperviseur ainsi qu'entre CMS et environnement logiciel invité. Le System/360 était déjà capable de gérer de la virtualisation récursive. La finalité de ce produit d'IBM était de pouvoir consolider les postes de travail liés. La dernière implémentation par IBM de VM est z/VM qui fonctionnait sur les zSeries.

Figure 1: System/360 de IBM

Entre la fin des années 80 et le milieu des années 90 Commodore International commercialise l'Amiga qui est ordinateur personnel très populaire à l'époque. Il était aussi bien capable de lancer de lancer des pc X386, des Macintoshs 6800 et des solutions X11 en multitâches.

Suite à l'Amiga on trouve des systèmes Unix basés sur l'architecture NUMA, qui est une architecture mémoire de systèmes multiprocesseurs. Cette architecture consiste à cloisonner et partitionner la mémoire, les accès se faisant via de multiples bus, un par processeur.

En 1999 VMware proposa un système propriétaire de virtualisation de systèmes x86 à base de systèmes hôtes x86. D'autres projets libres ont suivi VMware, tels que QEMU, Xen, Bochs, kvm, VirtualBox ainsi que des logiciels gratuits mais propriétaires tels que VirtualPC, VMware Server, Virtual Server.

Dans les années 2000 afin d'améliorer les capacités des solutions de virtualisation. Enfin dans l'année 2000 les fabricants de processeurs Intel et AMD ont implémenté des fonctions de virtualisation dans leurs processeurs permettant la prise en charge de systèmes d'exploitation non modifiés plus efficacement.

1.3. Pourquoi virtualiser?

A l'époque où les ordinateurs n'étaient capables de ne faire exécuter qu'un seul processus en même temps, l'intérêt de se diriger vers un système supportant la gestion multiprocessus était de pouvoir optimiser les ressources de calcul. La virtualisation va plus loin encore, nous allons voir ici les différents intérêts qu'elle porte.

Nous allons voir au cours de cette partie différents avantages que la virtualisation peut apporter. Nous aborderons différents points tels que la sécurité, le cout, ainsi que les performances.

1.3.1. La sécurité

La virtualisation va permettre une isolation des différents environnements logiciels au niveau des ressources physiques. La communication entre les différentes machines virtuelles sera uniquement possible via des connexions réseau de manière identique à la communication entre deux machines physiques. L'isolation est telle que la compromission d'un système invité par du code malicieux ne pourra pas se propager à d'autres systèmes invités. Il sera donc tout à fait possible d'isoler chaque service sans devoir acheter un nouveau serveur à chaque fois.

Un problème de sécurité pourrait apparaitre si un système invité avait la possibilité de lire les données mémoire ou disque d'un autre système invité. De la même façon, si deux applications de deux systèmes invités tentent d'accéder à une même ressource physique, il se produira une interférence. Avec un système de virtualisation, il sera possible de gérer les accès aux ressources physique de manière à éviter les conflits. Ceci pourra se faire soit par l'allocation de temps d'accès soit par l'allocation exclusive.

Une autre application très intéressante pour les professionnels et les chercheurs en sécurité est l'observation de logiciels malveillants à travers de systèmes d'exploitation invités surveillés. Ce type d'utilisation permettra de suivre l'évolution de l'infection d'un système. En outre, il sera possible de manipuler ces systèmes d'exploitation pour revenir en arrière et expérimenter diverses techniques de désinfection et de détection.

1.3.2. Le coût

Actuellement, nous réalisons le fait que les serveurs sont largement sous utilisés. En effet, une étude Gartner a montré qu'en moyenne les serveurs sont utilisés à 5% de leur capacité.

La sous-utilisation de serveurs empire avec la présence de serveurs en *hot spare*. Un tel serveur est un serveur qui attend un dysfonctionnement du primaire pour prendre le relais. En absence de dysfonctionnement, ce serveur n'effectue aucun travail alors qu'il consomme de l'énergie et prend de la place en centre de données. Avec les coûts grandissants des matières premières énergétiques et donc *a fortiori* l'électricité, les gestionnaires de parcs serveurs se retrouvent contraints de se soucier des enjeux d'optimisation de la consommation électrique. De plus, il est nécessaire de gérer des contraintes liées au manque d'espace dans les centres de données dont le prix augmente à cause de la demande croissante.

Un hébergeur de serveurs va pouvoir bénéficier de la virtualisation par le biais d'une gestion de ressources plus fine. Il va lui être possible de gérer finement la répartition des ressources physiques entre les différents serveurs. Il va également bénéficier de fonctionnalités de comptabilité de l'utilisation des ressources pour pouvoir proposer une facturation plus adaptée à l'utilisation des ces clients.

1.3.3. Criticité et performances

L'accumulation de systèmes invités sur un même système physique augmente sa criticité. Une panne matérielle rendra indisponibles tous les systèmes invités. Pour pallier à la criticité accrue du système physique, des solutions ont été prévues.

La solution la plus basique est de prévoir la possibilité de faire des images des systèmes invités. Des taches très longues à effectuer vont pouvoir être sauvegardés en cours d'exécution. Il sera ensuite possible de relancer ces tâches à partir de ce point en cas de panne.

Un autre exemple de l'utilisation de ce procédé est le clonage de différentes instances d'un invité. Par exemple, si plusieurs tâches nécessitent un même travail préliminaire long, il sera possible de faire effectuer dans une première instance ce travail, ensuite de la cloner autant de fois qu'il y a d'autres tâches et de les lancer séparément.

Une solution plus efficace pour palier à la criticité des serveurs de virtualisation est la migration dynamique de systèmes invités. Le pré-requis est d'avoir un support de stockage accessible en réseau. Un système invité est exécuté sur un premier serveur. Dans le cas d'une panne matérielle ou d'une maintenance planifiée, il sera possible de migrer dans de très courts délais ce système vers une autre machine tout en sauvegardant son état avant l'interruption. Ce procédé permet à lui seul de réduire la criticité des systèmes individuels à un niveau largement acceptable. Il fait reposer une plus grande criticité sur les systèmes de stockage en réseau qui peuvent cependant être dédoublés.

Une autre utilité de ce procédé en terme de performances va être de pouvoir allouer des ressources à la volée aux systèmes invités. Dans le cas d'une montée en charge d'un système invité, il sera possible de l'isoler sur un serveur en déplaçant les autres systèmes présents à ces cotés sur d'autres serveurs moins chargés. Ce procédé peut également être utile dans un objectif de réduction de la consommation électrique d'un parc de serveur. Il est envisageable de n'utiliser que peu de machines lorsque le système est soumis à une faible charge et d'allumer progressivement les autres machines en fonction de la montée en charge. Ceci nécessite une gestion de l'allumage électrique du serveur par le réseau mais bon nombre de serveurs récents disposent de tels outils.

1.3.4. Conclusion

Au final, la virtualisation présente de nombreuses réponses à des problèmes qui se posent aujourd'hui que ce soit au niveau logiciel avec une nécessité de sécurisation, ou au niveau énergétique avec la forte augmentation du cout de l'énergie électrique.

1.4. Les solutions techniques

Dans un souci de cohérence, il est nécessaire de clarifier certains termes. Nous avons choisi de parler de système hôte lorsqu'il s'agit du système accueillant le systèmes virtualisés. Les systèmes invités sont les systèmes s'exécutant par le biais d'une couche logicielle de virtualisation.

1.4.1. Introduction

Dans cette partie nous allons aborder les différents types de virtualisation. Nous pouvons voir à travers l'arbre ci dessous les solutions que nous allons expliquer ci-après. Nous pouvons diviser la virtualisation en deux catégories, celle des systèmes et celle des processus. La virtualisation au niveau du système d'exploitation permet de faire fonctionner des environnements utilisateurs complètement cloisonnés sur un système d'exploitation unique. Du coté des systèmes on trouve deux types de virtualisation. Celle qui accepte des environnements invités non modifiés et inversement. La virtualisation totale permet de faire fonctionner différents types d'architecture et donc différents systèmes d'exploitation en même temps, ceci sur la base d'une machine avec un système d'exploitation complet. La virtualisation matérielle assistée quand à elle permet le fonctionnement de machines virtuelles sur différents OS en tirant partie des instructions processeurs dédiées à la virtualisation. La base de la machine qui accueille les machines virtuelles (MV) est une couche logicielle plus légère qu'un OS complet. L'émulation du matériel reste complète. Enfin la paravirtualisation demande de son coté une modification des MV afin de modifier la vue que celles-ci ont du matériel sous jacent. C'est le seul type de virtualisation qui n'émule pas de matériel pour les MV.

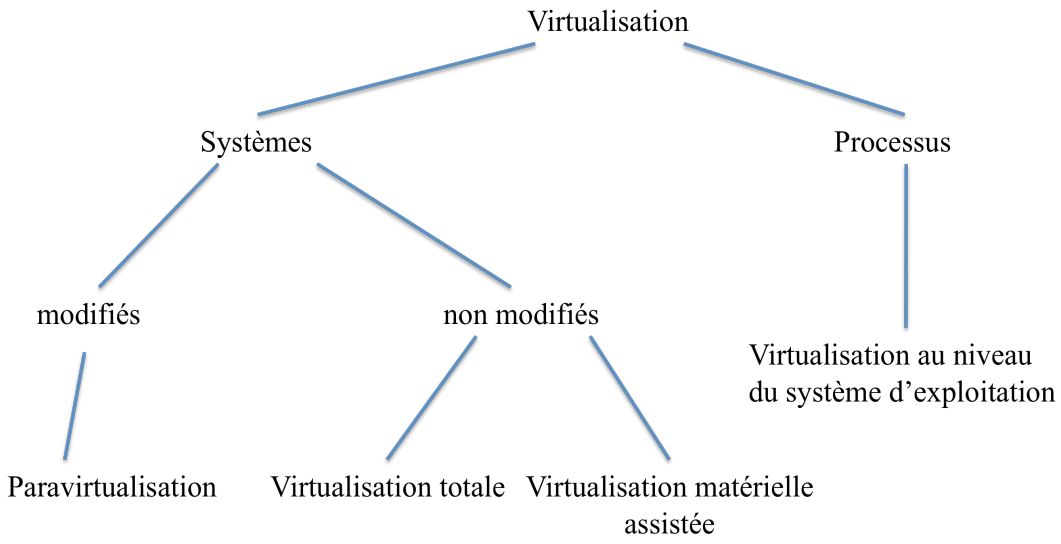

Figure 2 : Classification des différents types de virtualisation

1.4.2. La virtualisation au niveau du système d'exploitation

Ce type de virtualisation consiste à séparer le système d'exploitation (OS) d'une machine en différents environnements utilisateurs distincts. Ainsi les utilisateurs de la machine ne se voient pas entre eux, et l'accès aux données des autres n'est pas possible. Les environnements utilisateurs sont entièrement cloisonnés. En effet ici le matériel et l'OS sont les même pour tout le monde. On ne peut pas parler de « systèmes d'exploitations invités » pour ces systèmes invités mais plutôt d'environnements utilisateur cloisonnés ou « jails ».

L'allocation des ressources entre les différents environnements utilisateurs est possible. On donnera des quotas disque à ces espaces privés ainsi qu'une limite en terme d'utilisation de puissance processeur. La mémoire vive disponible et la bande passante réseau sont d'autres ressources que l'on peut limiter. Ce type de virtualisation est essentiellement utilisé sur les systèmes Unix, Linux et BSD.

Au niveau de l'architecture du système virtualisé, nous avons le système hôte avec ses invités au dessus. Le système hôte sera ici l'OS qui supportera les environnements utilisateurs invités. Ceux-ci n'ont pas de noyau propre. Il n'y a qu'un seul noyau pour faire fonctionner le système complet. Néanmoins les environnements utilisateurs croient être sur des machines différentes (montages disques différents, adressage réseau différent).

Un des principaux intérêts de cette technique est que l'on peut facilement faire migrer à chaud un processus d'un environnement utilisateur à un autre. L'exemple type serait qu'un programme « A » requiert une certaine puissance, que celle-ci ne peut être délivrée dans l'environnement utilisateur où il se situe. La solution est de le migrer à chaud vers un environnement plus puissant.

De plus ce type de virtualisation est utilisé chez les hébergeurs web qui proposent des environnements privé à moindre coût pour que les utilisateurs puissent gérer leurs sites web comme si ils possédaient chacun un serveur personnel (Primet/Vicat-Blanc 14 Septembre 2007) (Wikipédia n.d.).

Comme le montre la figure 2, C'est une couche logicielle en plus du système d'exploitation qui va permettre la séparation des environnements utilisateurs. A noter que le noyau est le même pour tous (machines virtuelles, système hôte).

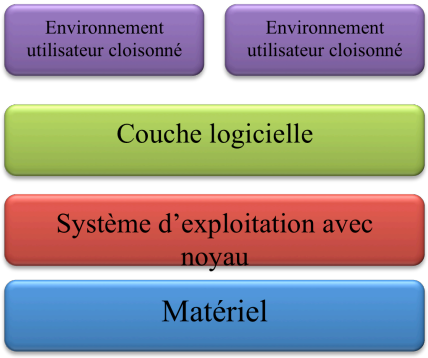

Figure 3 : Virtualisation au niveau du système d'exploitation

1.4.3. La virtualisation totale

La virtualisation totale 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 concept de virtualisation totale est déjà bien ancré dans la littérature, mais ce n'est pas toujours ce terme qui est employé (Wikipédia n.d.). La virtualisation partielle peut être confondue avec. Nous allons voir dans les paragraphes qui suivent les définitions de ces deux termes.

En virtualisation totale, la machine physique qui va émuler le matériel pour le système invité doit être doté d'un OS ainsi que d'une surcouche applicative. Un des gros intérêts de cette technique de virtualisation est de pouvoir émuler n'importe quelle architecture matérielle. On peut donc faire fonctionner les OS que l'on désire indépendamment de l'architecture du système hôte. On l'utilise essentiellement en milieu industriel pour faire fonctionner des applications sur des architectures matérielles non encore commercialisées (Primet/Vicat-Blanc 14 Septembre 2007). Il faut savoir que ce type de virtualisation n'est que logiciel, aucune fonctionnalité matérielle de virtualisation n'est utilisée.

On peut voir sur la figure 3 que les 3 premières couches sont identiques à la virtualisation au niveau de l'OS. Ici sur une 4ème couche on trouve l'émulation du matériel désiré afin de pouvoir accueillir tout type d'OS en tant que machine virtuelle.

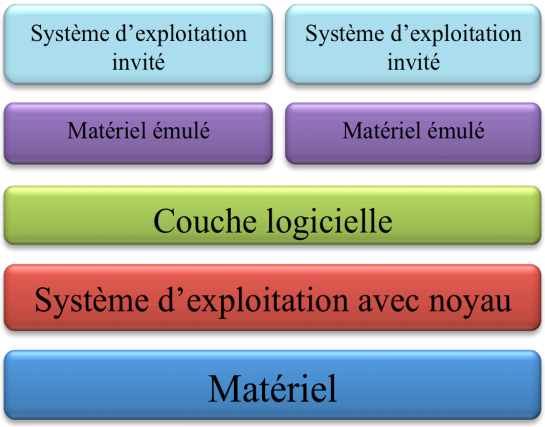

Figure 4 : Virtualisation totale

La virtualisation partielle quant à elle n'a pas pour but de simuler de nouvelles architectures pour les systèmes invités, mais uniquement une partie. En effet on va émuler la partie matérielle qui nous intéresse pour faire fonctionner une application particulière. Dans ce cas l'émulation d'une seule ressource peut être suffisante. On parle donc de virtualisation partielle. L'intérêt est donc de ne pas re-simuler tout le matériel de la plateforme émulée.

Lorsqu'IBM a inventé ce terme, on appelait virtualisation partielle le fait de cloisonner les utilisateurs dans des espaces d'adressages mémoire distincts. Ce terme n'est plus employé aujourd'hui dans ce sens (Wikipédia n.d.).

En virtualisation totale la machine accueillant les systèmes invités doit donc implémenter de façon logicielle une gestion complète du matériel des invités. La gestion de la mémoire est un des facteurs les plus critiques en terme de performances. 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 virtualisation totale est la technique qui s'éloigne le plus des performances que peut avoir un système classique.

1.4.4. La virtualisation matérielle assistée

Ce type de virtualisation a pour but de faire fonctionner des systèmes invités dont les OS peuvent être différents mais non modifiés. La différence avec la virtualisation totale est qu'ici on tire pleinement partie du matériel et de sa puissance. La perte de performances est minimum particulièrement au niveau du processeur.

Cette technique de virtualisation a été récemment implantée dans nos processeurs à base d'architecture x86 sous les noms de : technologies Intel-VTx (32 bits) et Intel VT-i (64 bits) pour Intel, de AMD-V pour AMD, Advanced Power Virtualization pour IBM et de Ultra SPARC T1 Hypervisor pour SUN.

Encore une fois on trouve plusieurs termes pour définir ce type de virtualisation, et ce sont les entreprises qui le dénomment différemment. Xen l'appelle HVM (Hardware Virtual Machine), Virtual Iron quand à lui la dénomme virtualisation native (Wikipédia n.d.).

Le but de ce type de virtualisation était de pouvoir faire fonctionner tout type d'OS non modifié de façon parallèle sur une même machine sans perte de performance. L'utilisation des fonctionnalités processeurs liées à la virtualisation a permis en partie cet exploit mais les différentes sources ne sont pas en accord vis-à-vis du facteur de perte de performance (Primet/Vicat-Blanc 14 Septembre 2007) (Wikipédia n.d.) (Nakajima n.d.).

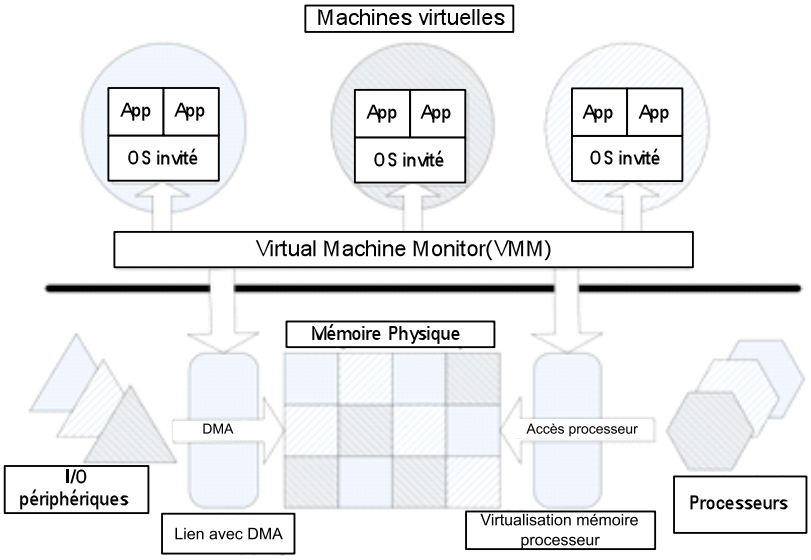

Figure 5 : Architecture Intel-VT

Nous pouvons voir ci-dessus une figure représentant le fonctionnement de l'architecture Intel-VT, nous avons en bas les composants matériels d'un ordinateur (processeurs, mémoires, périphériques d'entrées, sorties). Au dessus nous avons le VMM. Le VMM est une application qui permet d'utiliser pleinement partie de la technologie Intel-VT. Enfin en haut nous avons les machines virtuelles qui sont confinées chacune dans leur propre environnement.

Le VMM, Virtual Machine Monitor est un outil logiciel qui va s'assurer de la répartition et de la division des ressources matérielles pour chacune des machines virtuelles. Il est également capable de faire la distinction des interruptions entrées/sorties émisses par les périphériques de la machine et de les relayer à destination des différentes machines virtuelles (whatis.com n.d.).

La virtualisation assistée matérielle est donc une évolution de la virtualisation totale puisque elle émule toujours le matériel nécessaire au bon fonctionnement des systèmes invités, mais avec une perte de performances moindre. En effet ici l'architecture matérielle est doté de fonctionnalités spécifiquement développé pour ce genre d'utilisation. La perte de performances est donc plus faible vis à vis d'un système natif. (Bonnet 2008)

1.4.5. La paravirtualisation

Cette technique présente un logiciel en tant qu'intermédiaire entre le matériel et les systèmes d'exploitation invités et non un système d'exploitation. La deuxième différence par rapport aux techniques vues précédemment est que les systèmes invités sont modifiés. Ils sont conscients du fait qu'ils sont virtualisés.

L'application située entre le matériel est les systèmes invités est appelée VMM comme en virtualisation matérielle assistée. Ici on ne tire pas parti des instructions processeur liées à la virtualisation. Ce VMM aussi appelée hyperviseur est le plus simple possible afin d'avoir le minimum de pertes de performances. C'est lui qui est chargé d'appliquer la politique d'accès aux ressources matérielles pour les systèmes invités.

Etant donné des modifications apportées, ces systèmes ont été adaptés au niveau de leur noyau afin qu'ils puissent communiquer avec l'hyperviseur.

Dans les systèmes natifs, il existe des instructions privilégiées. Ces instructions processeurs permettent d'accéder directement à la mémoire physique du processeur sans passer par la mémoire virtuelle. Une autre utilisation de ces instructions est la possibilité de changer l'état de la machine dans le sens ou cela influe sur des processus. En paravirtualisation cela pose un problème non négligeable puisque les machines virtuelles n'ont pas accès directement au matériel et ne peuvent ainsi pas faire exécuter ces instructions privilégiées. Ce problème ne s'était pas posé jusqu'alors puisque dans les techniques vues précédemment, le processeur est émulé donc le système invité exécute ces instructions privilégiées sur ce « faux » processeur.

Mais un jeu d'instruction appelé « hypercalls » existe pour fournir des fonctionnalités similaires. Les applications n'étant pas modifiées, celles-ci émettrons des appels systèmes privilégiés classiques. Ces appels seront captés par le noyau modifié du système invité et seront transformés en hypercalls. Ceux-ci seront alors envoyés à l'hyperviseur qui pourra donc les interpréter.

Ce cheminement est expliqué par la figure ci-dessous. Du coté de la virtualisation native ou virtualisation matérielle assisté on voit clairement que les appels des MV sont directement captés par le noyau de la VMM. Nous expliciterons la notion de noyau dans la partie concernant les pré-requis sur les systèmes d'exploitation. (Bonnet 2008)

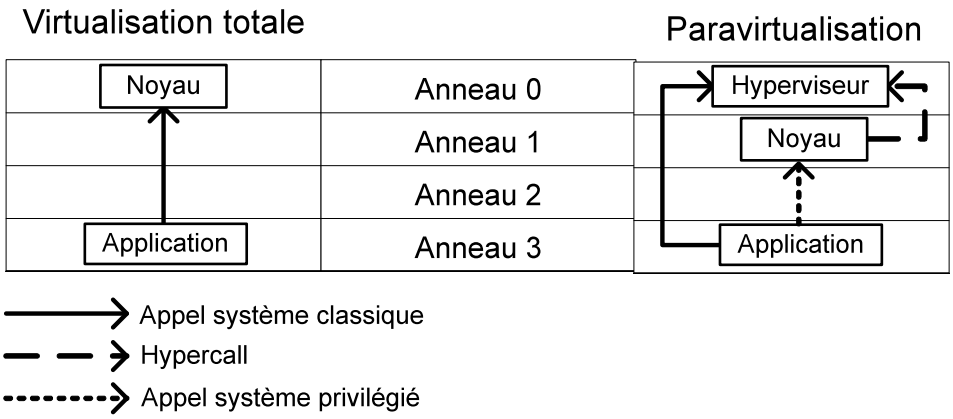

Figure 6 : Appels systèmes en para/native virtualisation

1.4.6. La virtualisation hybride

La virtualisation hybride consiste à coupler la paravirtualisation avec la virtualisation matérielle assistée. Elle permet de faire fonctionner des systèmes d'exploitation non modifiés qui profitent en partie des instructions processeur Intel-VT ou AMD-V ainsi que certaines extensions liées à la paravirtualisation. Dans ce cas la VMM ou hyperviseur est capable d'accomplir les taches requises par les 2 technologies de virtualisation.

1.4.7. Arbre résumé et conclusion

Sur la figure 6 on peut voir que concernant le niveau d'émulation du matériel au niveau des MV la virtualisation au niveau du système d'exploitation est la plus faible. En effet c'est le seul type qui ne propose pas d'autre OS ni d'autres architectures matérielles. La paravirtualisation précède la virtualisation matérielle car elle ne propose pas une émulation du matériel pour chaque MV. Enfin la virtualisation totale comme nous l'avons vu précédemment émule les plates-formes matérielles que l'on veut elle a donc le niveau le plus élevé en terme d'émulation matérielle.

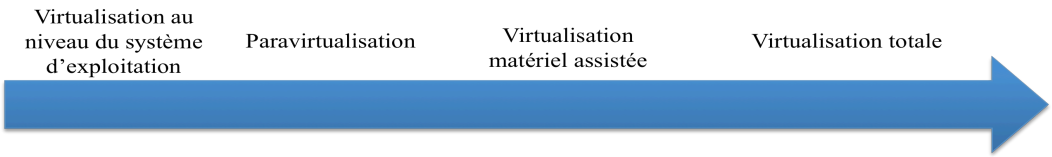

Figure 7 : Classification par rapport au niveau d'émulation matérielle

Cette figure résume le positionnement de l'ensemble des différentes méthodes de virtualisation. La dernière partie de ce rapport traitera plus particulièrement de la solution Xen et son hyperviseur dans le cadre de la paravirtualisation.

1.5. Les solutions commerciales

Dans cette partie, nous allons établir un aperçu des solutions commerciales de virtualisation. Nous séparerons cet aperçu en deux catégories. La première partie traitera des solutions propriétaires qu'elles soient gratuites ou non. La seconde partie traitera ensuite des solutions open source.

1.5.1. Les solutions propriétaires

Tout d'abord, l'éditeur d'applications de virtualisation le plus connu est VMWare. Il s'agit du premier éditeur dans le domaine. Sa gamme de solutions porte son nom. Elle ne comporte pas moins de 13 logiciels différents. Nous allons nous intéresser en détail qu'à un échantillon. Tout d'abord, VMWare Workstation est la solution entrée de gamme. Elle est clairement destinée aux utilisateurs désirant pouvoir utiliser plusieurs systèmes d'exploitation simultanément. Ses fonctionnalités sont basiques et couvrent les besoins élémentaires d'un utilisateur moyen. Il est possible d'exécuter un ou plusieurs systèmes d'exploitation en simultané tout en leur partageant la connexion réseau. Ensuite VMWare ESX Server est une solution orientée vers une utilisation entreprise sur des machines de type serveur. Un système d'exploitation minimaliste basé sur Red Hat est installé directement sur le matériel. Il n'est donc pas nécessaire d'installer un système d'exploitation hôte, il est directement fourni. VMWare parle d'hyperviseur lorsqu'il parle de cette couche logicielle minimaliste. Il ne s'agit cependant en aucun cas d'un hyperviseur au sens du terme établi dans le cadre du projet Xen. L'administration se fait soit par le biais d'une console soit par le biais d'une interface web. Les autres logiciels proposés dans le cadre de la gamme VMWare vont servir d'autres utilités plus spécifiques telles que la mise en haute disponibilité (VMWare HA), la supervision de machine virtuelles (VMWare Virtual Center) ou encore l'exécution de machines virtuelles pré-empaquetées (VMWare Player). En ce qui concerne les technologies utilisées, les solutions VMWare utilisent toutes la virtualisation complète et la virtualisation matérielle assistée.

Une autre solution propriétaire est Virtual PC édité par Microsoft. Elle est totalement gratuite. Ses fonctionnalités sont basiques et le public visé est l'utilisateur moyen qui souhaite exécuter un autre système d'exploitation simultanément. Une version serveur est prévue sous le nom de Hyper-V. Une partie des fonctionnalités de ce logiciel sera proposé de base dans Windows Server 2008 et une autre partie sera vendue séparément. Des fonctionnalités avancées telles que la migration rapide d'environnements invités ou l'équilibrage de charge seront incluses dans la version payante. Virtual PC utilise uniquement la virtualisation totale tandis qu'Hyper-V s'étendra à la virtualisation matérielle assistée.

Au final, nous avons pu faire un tour des solutions propriétaires de virtualisation. Nous pouvons remarquer qu'il n'existe pas de solutions propriétaire de paravirtualisation ou de virtualisation au niveau du système d'exploitation. Ces solutions souffrent donc de fortes pertes de performances liées aux technologies utilisées.

1.5.2. Les solutions libres

La solution libre de référence en matière de virtualisation est Xen. Nous ne ferons qu'évoquer Xen dans cette partie puisqu'il sera largement discuté et étudié par la suite de ce rapport. Il s'agit de la seule solution qui propose une paravirtualisation des systèmes d'exploitation. Une autre solution de virtualisation est le projet QEMU. Il permet d'exécuter un ou plusieurs systèmes d'exploitation invités à l'intérieur d'un système d'exploitation hôte tout en leur partageant la connexion réseau. Ces systèmes d'exploitation peuvent être conçus pour d'autres plateformes que celle de la machine physique. Par exemple, il est possible d'exécuter un système d'exploitation PC dans un système PowerPC. QEMU utilise la virtualisation totale. Ce choix de technologies résulte en une importante perte de performances. Il sera plus intéressant d'utiliser KVM dont QEMU fait partie.

Le projet KVM qui signifie *Kernel Virtual Machine* se décompose en deux composants. Le premier composant est un module contenu dans le noyau Linux. Il sert à donner la possibilité au noyau d'utiliser les extensions de virtualisation et communique avec le processeur. Le second composant est un programme utilisateur qui va envoyer au module toutes les instructions privilégiées. Les instructions privilégiées seront traitées ultérieurement dans le cadre de la section sur les problématiques liées à l'architecture x86. Le second composant est une version légèrement modifiée de QEMU. KVM reprend largement les fonctionnalités de QEMU. Il rajoute cependant quelques fonctionnalités intéressantes. KVM permet l'utilisation des extensions processeur de virtualisation ce qui n'était pas possible avec QEMU. Il propose également la migration à chaud de systèmes d'exploitations invités. Le principal inconvénient de KVM réside en sa jeunesse et son manque de maturité. Il est encore très peu utilisé en entreprise et ses fonctionnalités n'ont donc pas encore pu être éprouvées.

Nous pouvons aussi évoquer le projet OpenVZ dont l'objectif est de proposer une solution de virtualisation au niveau du système d'exploitation ou plutôt de cloisonnement. Cette solution a l'avantage d'être plus performante que toutes les autres dans la mesure où il n'y a pas d'émulation et que peu de traitement des informations émises par les systèmes invités.

2. Pré requis à la compréhension de Xen

2.1. La notion d'anneau

2.1.1. Introduction

Les anneaux de protection d'un ordinateur sont identifiés par les différents privilèges que peuvent avoir les différentes applications qui fonctionnent sur ce système. Ils sont couramment appelés Rings. Les anneaux sont numérotés, et le chiffre le plus faible correspond toujours à l'anneau le plus privilégié. Dans l'architecture x86 on retrouve 4 anneaux. L'intérêt de cette division des droits se situe dans le domaine de la sécurité du système. En effet, une application présente dans un anneau ne pourra pas accéder à des informations en dehors de cet anneau.

On retrouve au plus bas niveau le noyau du système qui a les droits d'accéder aux ressources critiques que sont le processeur, la mémoire. Ce type de restrictions est généralement renforcé par un fonctionnement similaire de l'architecture processeur, qui propose des modes avec plus ou moins de privilèges. Les anneaux 1 et 2 ne sont pas utilisés par les systèmes d'exploitation de nos jours dans le cadre de l'architecture x86-32. Le dernier anneau accueille toutes les applications autres que le noyau.

Sur la figure ci-dessous on retrouve les 4 anneaux explicités précédemment, de l'intérieur vers l'extérieur : du plus privilégié vers le moins.

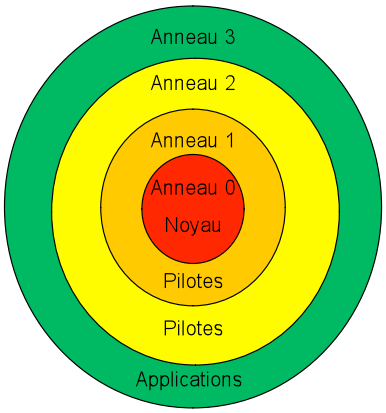

Figure 8 : Anneaux de privilèges de l'architecture x86

2.1.2. Exemples

Nous pouvons illustrer le fonctionnement des anneaux par la présence de spyware dans un système d'exploitation. Ceux-ci se retrouveraient à fonctionner sur l'anneau le moins privilégié le 3. De ce fait ce spyware ne pourrait pas allumer la carte son qui elle fonctionne dans un anneau inférieur.

Sachant que les périphériques fonctionnent dans l'anneau 1, si aucun pont n'est autorisé par l'utilisateur entre l'anneau 3 et le 1, le programme en question (ici le spyware) ne pourra pas accéder au périphérique désiré.

De la même manière un navigateur web fonctionne en anneau 3 si celui-ci veut accéder à internet, il lui faudra l'accès à la carte réseau qui est un périphérique et donc fonctionne en anneau 1.

Un dernier exemple de l'utilisation de ces anneaux de privilèges est le fait que si un processus de niveau 3 plante, son plantage n'influera pas sur la pérennité du système puisque cela n'aura aucune incidence sur les anneaux inférieurs.

Il y a eu beaucoup d'implémentations de cette notion d'anneau de protections dans les OS. Chacun utilise plus ou moins chaque anneau avec plus ou moins de flexibilité. La plupart (Linux et Windows) utilisent 2 des 4 anneaux. Le 0 qui a accès à tout le matériel de la machine, et le 3 dans le quel on a toutes les applications.

2.1.3. Application à la virtualisation

La problématique que l'on peut souligner pour lier cette explication avec la virtualisation est le besoin de placer un logiciel (appelé hyperviseur) qui peut avoir tous les droits et qui soit sous jacent du noyau.

Les architectures x86 ont 4 anneaux comme nous l'avons vu précédemment. Xen par exemple réussit à placer le noyau en anneau 1 et ainsi permettre à un hyperviseur de fonctionner en anneau 0. Mais cette technique n'est pas toujours possible suivant les architectures. Notamment dans les architectures 64 bits qui elles ne possèdent que 2 anneaux. La méthode développée par Xen est de placer le noyau dans le même anneau que les applications et ainsi libérer la couche 0 pour l'hyperviseur. On a ici un problème lié à la sécurité que cette dernière technologie propose.

Mais heureusement les architectures récentes proposent désormais un support de la virtualisation et implémentent un anneau -1 ce qui permet de garder l'architecture en 2 couches précédente intacte avec une 3ème couche sous les 2 premières comme le montre la figure ci-dessous. (Wikipédia n.d.)

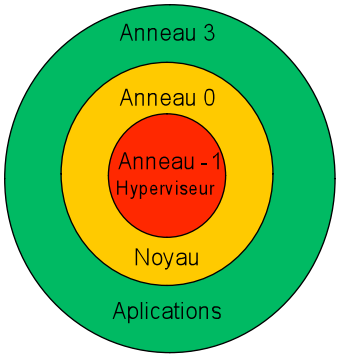

Figure 9 : Anneaux de privilèges sur architectures dédiées à la virtualisation

2.2. Les problèmes liés à l'architecture x86 et des solutions

La virtualisation de systèmes d'exploitation nécessite la gestion par le composant de virtualisation des ressources physiques pour pouvoir les allouer ensuite aux systèmes d'exploitation invités. Dans ce chapitre, nous évoquerons la gestion du processeur.

Dans un contexte classique, un processus utilise le processeur de manière exclusive pendant un certain temps puis il y a une interruption qui engendre une sauvegarde du contexte du processeur et ensuite un autre processus a accès au processeur. Ceci a lieu en moyenne toutes les 10 millisecondes et se répète indéfiniment tant qu'il y a des opérations à faire par le processeur. Le processeur fonctionnant en mode privilégié, c'est-à-dire, avec accès total aux ressources physique du système, n'est pas compatible directement avec la virtualisation. En effet donner accès à une machine virtuelle à toutes les ressources pose de gros problèmes de sécurité et d'intégrité des données.

Dans le but de formaliser les pré-requis à la virtualisation, Popek et Goldberg ont proposé un jeu de critères indispensables. Leur théorie consiste à décomposer les instructions processeur en trois catégories :

- Instructions privilégiées
- Instructions sensibles de configuration
- Instructions sensibles de comportement

Les instructions privilégiées sont des instructions qui nécessitent que le processeur soit en mode privilégié pour être exécutées. Dans le cas où le processeur n'est pas en mode privilégié, ces instructions sont interceptées. Les instructions sensibles sont les instructions qui visent à modifier la configuration des ressources physiques du système. Des exemples de ces instructions sont les instructions visant à modifier l'adressage de la mémoire virtuelle vers la mémoire physique, les instructions de communication avec les autres périphériques ou les instructions de manipulation des registres de configuration globaux. Les instructions sensibles de comportement sont celles dont le

comportement dépend de la configuration des ressources. Nous pouvons citer comme exemple les opérations de stockage et de rapatriement de données de la mémoire virtuelle.

Selon la théorie de Popek et Goldberg, pour qu'une architecture soit virtualisable, il faut que toutes les opérations sensibles soient également des instructions privilégiées. De manière plus pratique, cela signifie qu'une partie logicielle doit être capable d'intercepter des instructions susceptibles de modifier l'état de la machine. Une fois l'interception effectuée, la partie logicielle aura le choix entre faire exécuter par le processeur en mode privilégié ou la modifier.

Dans le cas de l'architecture x86, dix sept instructions dérogent à la règle de Popek et Goldberg, c'est-à-dire, qu'elles font partie du groupe des instructions sensibles mais pas du groupe des instructions privilégiées. Ces instructions sont problématiques dans la mesure où elles ne seront pas interceptées si le processeur ne fonctionne pas en mode privilégié. Des exemples des ces instructions sont les instructions LAR et LSL qui permettent de charger des informations sur une partie de la mémoire sans que le processeur ne soit en mode privilégié. La disponibilité de ces instructions aux systèmes d'exploitation invités empêcherait la partie logicielle sous-jacente de modifier la mémoire sans que les systèmes d'exploitation invités ne le sachent.

Bien que l'architecture x86 présente de nombreuses difficultés pour virtualiser, il s'agit néanmoins d'une plateforme très intéressante puisqu'elle est très largement diffusée. Du fait du fort potentiel d'utilisateurs de l'architecture x86, de nombreux projets se sont lancés pour trouver une solution aux problèmes intrinsèques à cette architecture.

La solution la plus immédiate qui a été mise en œuvre par VMWare et bien d'autres dès 1999 a été l'émulation totale d'un système physique, ce que nous avons appelé virtualisation totale précédemment. Bien que ce type de virtualisation ait été récemment amélioré avec l'arrivée des extensions processeur de virtualisation (Intel-VT et AMD-V), le différentiel de performance avec un système installé de manière classique reste très important (de 10 à 80%).

Le projet Xen a proposé une solution radicalement différente en proposant de contourner le problème de ces instructions problématiques en les remplaçant par d'autres. (Chisnall 2007)

2.3. Gestion de la mémoire

2.3.1. Définition et intérêt de la pagination

Deux notions sont à définir avant de parler de pagination : la mémoire physique et la mémoire virtuelle. La mémoire physique ou RAM permet de stocker et d'exécuter tous les programmes en cours. On parle ici du système d'exploitation et des programmes lancés à partir de celui-ci. On parle de mémoire virtuelle pour un programme. Elle est un espace d'adressage dédié à celui-ci.

La pagination est la technique de gestion de la mémoire utilisée par les systèmes d'exploitation. Cette technique permet d'avoir une gestion uniforme de la mémoire. De plus elle assure sa fragmentation qui consiste à la découper en multiples morceaux uniformes. Mais le réel intérêt d'avoir développé un tel outil de pouvoir utiliser des logiciels qui requièrent plus de mémoire que le système n'en dispose physiquement. A savoir que le disque dur peut aussi faire parti partiellement de la mémoire virtuelle sous le nom de swap.

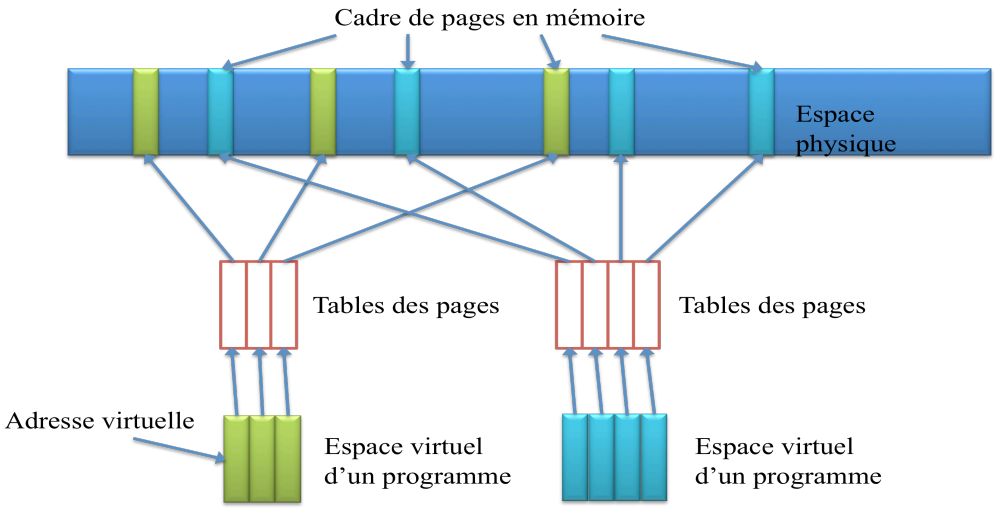

Figure 10 : Pagination de plusieurs programmes

Comme le montre la figure ci-dessus, chaque programme dispose de pages virtuelles d'une certaine taille. Dans l'espace physique on dispose de blocs, de même taille que les pages virtuelles. Ceci permet notamment à n'importe quelle page virtuelle d'être chargée dans l'espace physique. Le mécanisme de gestion qui fait le lien entre ces deux espaces d'adressage est appelé table des pages. Chaque programme dispose de sa propre table. Cette table peut donc grossir assez vite puisque qu'elle peut contenir toutes les adresses des blocs de la mémoire physique. La table des pages peut être vue par analogie au sommaire très détaillé d'un livre ou chaque information prendrait une page complète et ou chaque page serait référencée dans le sommaire. Sachant que l'on a une multitude de tables qui sont utilisées en même temps sur un système, on a donc besoin d'identifier chaque table. Pour cela on dispose d'un registre nommé : le registre de Base de la Table (RBTP). Celui-ci fait partie du contexte du programme

Sur la figure suivante on peut voir que grâce à l'adresse virtuelle on peut retrouver l'information recherchée dans la table des pages. De la table des pages on peut ainsi accéder au numéro de bloc physique. Une fois l'adresse du bloc physique obtenue et grâce au numéro de déplacement stockée dans l'adresse virtuelle, on peut localiser précisément dans la mémoire physique l'information recherchée.

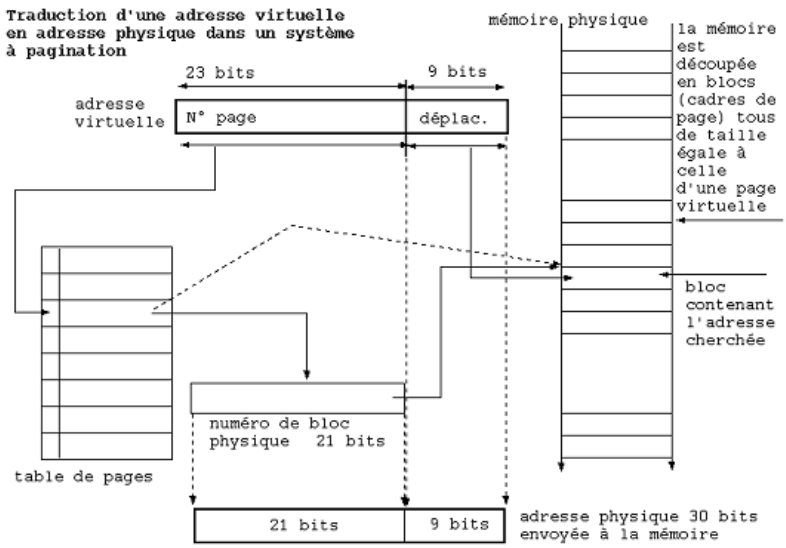

Figure 11 : Traduction d'adresse virtuelle en adresse physique

Ces informations sont prises d'un cours, pour plus d'informations vous pouvez consulter celui-ci. (Vayssade 2004)

2.3.2. La segmentation

Comme nous l'avons expliqué ci-dessus, la taille des tables de pages peut grandir très vite et devenir par la même occasion inexploitable car le temps d'accès serait trop long. Un autre mécanisme couplé à la pagination appelé segmentation vient consolider le système de gestion de la mémoire.

En effet la segmentation va découper et structurer les tables de pages. Ce découpage n'est pas fait aléatoirement. Cette segmentation va se faire par morceau de programme ou de programmes entier ou encore par un ensemble de sous-programmes. Maintenant que l'on a découpé nos tables de pages, il nous faut les organiser pour les retrouver facilement. Pour cela nous utilisons à nouveau une table qui composée de descripteurs de segments. Elle est appelée table de descripteurs de segments. Ceux-ci décrivent et pointent chacun vers une table de pages différente. Ceci est illustré par la figure suivante.

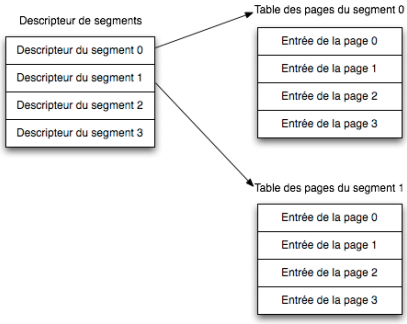

Figure : 12 Descripteurs de segments

Il existe dans l'architecture x86 deux tables de descripteurs de segments : la table des descripteurs locaux LDT et la table des descripteurs globaux GDT. Chaque programme possède sa propre LDT mais il n'existe qu'une seule GDT partagée par tous les programmes. Les LDT décrivent les informations des programmes alors que la GDT décrit les segments du système y compris ceux du système d'exploitation.

On a dit précédemment que chaque programme disposait de sa propre table de descripteurs appelée LDT. Le programme doit accomplir deux opérations afin de charger ce descripteur de segment. Il doit le sélectionner et le charger dans un registre.

Pour cela on dispose de sélecteurs de segments. Ceux-ci sont composés de 3 informations :

- Le numéro d'entrée dans la table
- La table de descripteurs (LDT ou GDT)
- Numéro de l'anneau de privilège du descripteur (0-3) cf 2.2.

On peut voir ces informations dans la figure ci-dessous :

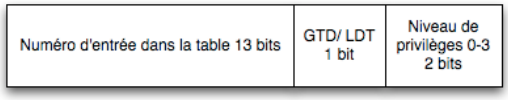

Figure 13 : Sélecteur de segment x86

Pour résumer on peut dire que la segmentation est la technique qui structure les multiples tables de pages qu'un système a à gérer.

2.3.3. Conclusion

Toutes des les opérations que nous venons de voir sont aujourd'hui exécutées par la MMU. Maintenant que nous avons fait un rappel sur le fonctionnement de la mémoire et plus particulièrement sur l'architecture x86, nous pourrons aborder de manière précise sa gestion dans Xen. Les informations liées à la segmentation ont été prises dans un ouvrage dédié aux systèmes d'exploitations et pour plus de détails veuillez le consulter. (Tanenbaum 2004)

3. Présentation de Xen

3.1. Introduction

Après avoir présenté la virtualisation d'un point de vue global, nous allons étudier plus spécifiquement le cas de la solution Xen. Le choix de cette solution a été largement encouragé par le fait qu'elle cette solution soit open source et donc qu'il soit plus facile d'obtenir des informations approfondies et précises à son sujet. Le projet Xen est soutenu par une communauté très active mais également par un grand nombre d'industriels tels que Novell, Sun, AMD, Cisco, Dell, HP, IBM, … Alors que de nombreuses autres solutions de virtualisation existaient déjà, Xen a su s'imposer en implémentant pour la première fois la paravirtualisation.

3.2. Les interactions entre l'hyperviseur, les applications, et le système d'exploitation

Tout d'abord, il est nécessaire de spécifier ce qu'est Xen. Xen est un hyperviseur de virtualisation. L'hyperviseur est une couche logicielle présente entre le matériel physique et les systèmes d'exploitation qui a pour but de traiter les instructions venant de ces derniers. La définition de l'hyperviseur au sens du projet Xen est différente de la définition de l'hyperviseur au sens d'autres projets tels que VMWare. Dans le dernier cas, l'hyperviseur est un logiciel présent au dessus d'un système d'exploitation permettant de virtualiser des systèmes d'exploitation. L'hyperviseur Xen a pour but d'être minimaliste. La raison derrière cela est qu'une erreur ou une faille dans ce dernier aurait des conséquences sur tous les systèmes d'exploitation invités qui peuvent être nombreux. Au fur et à mesure des versions, des fonctionnalités sont enlevées de l'hyperviseur et transférées vers un des systèmes d'exploitation invités.

Le projet Xen a proposé de contourner le problème des instructions ne répondant pas à la théorie de Popek et Goldberg en les supprimant des noyaux des systèmes d'exploitation invités. En échange, ils furent remplacés par des « hypercalls » dans le noyau Linux (CF 1.4.5.). Un hypercall est donc une instruction visant à remplacer une instruction processeur problématique par une instruction s'adressant directement à l'hyperviseur. Ceci est un changement par rapport aux solutions précédentes dans la mesure où les systèmes d'exploitation invités de paravirtualisation sont conscients du fait qu'ils se situent au dessus d'une couche de virtualisation dénommée hyperviseur. Le remplacement des

instructions problématiques du noyau a été rendu possible par le fait que les noyaux Linux et Unix soient open source.

Avec un système d'exploitation 32 bits, le noyau est exécuté dans l'anneau 0 qui est celui doté d'un plus grand niveau de privilège, et les applications sont exécutées dans l'anneau 3. Les anneaux 1 et 2 sont inutilisés dans la plupart des systèmes d'exploitation à l'exception d'OS/2 et avec certaines configurations NetWare. Comme il a été dit précédemment, l'hyperviseur a absolument besoin d'être exécuté dans le anneau le plus faible puisqu'il est l'intermédiaire exclusif avec le matériel. Ceci relègue donc les systèmes d'exploitation invités à l'anneau1 (voir figure ci-dessous). Du fait que l'hyperviseur soit dans le ring 0 et les systèmes d'exploitation invités soient dans le ring 1, il n'y a pas de système d'exploitation hôte lorsque l'hyperviseur Xen est installé, ils sont tous nécessairement virtualisés. Ceci illustre la différence de Xen avec les autres hyperviseurs qui présentent tous un système d'exploitation hôte même si celui-ci est minimaliste.

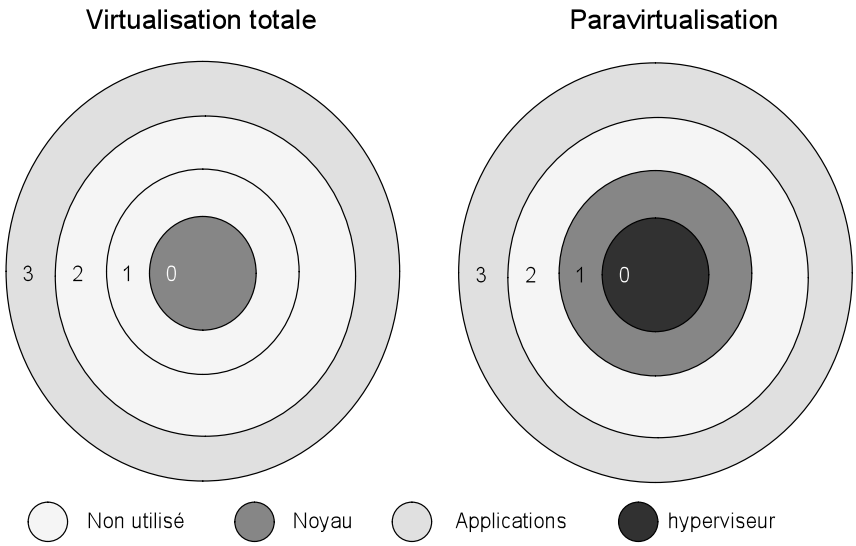

Figure 14 : Rings pour un système Xen en 32 bits

Lorsqu'AMD a révisé l'architecture x86-32 dans le but de créer son architecture 64-bits, connu sous le nom AMD64, des modifications ont été apportés au delà du passage en 64-bits. La modification qui nous intéresse le plus dans le cadre de la virtualisation est le passage de quatre à deux anneaux. Cette modification a été motivée par le fait que les deux anneaux intermédiaires n'étaient quasiment jamais utilisés comme vu précédemment. Ceci pose un problème dans la mesure où dans l'architecture x86-32, l'hyperviseur a pu simplement reléguer le noyau un anneau au dessus pour s'assurer d'être le seul dans l'anneau zéro. Or vu qu'il n'y a plus que deux anneaux dans l'architecture x86-64 et que nous avons trois couches logicielles à exécuter (hyperviseur, noyau et applications), il manque normalement un anneau. La solution a été de mutualiser l'anneau 1 pour les applications et les noyaux

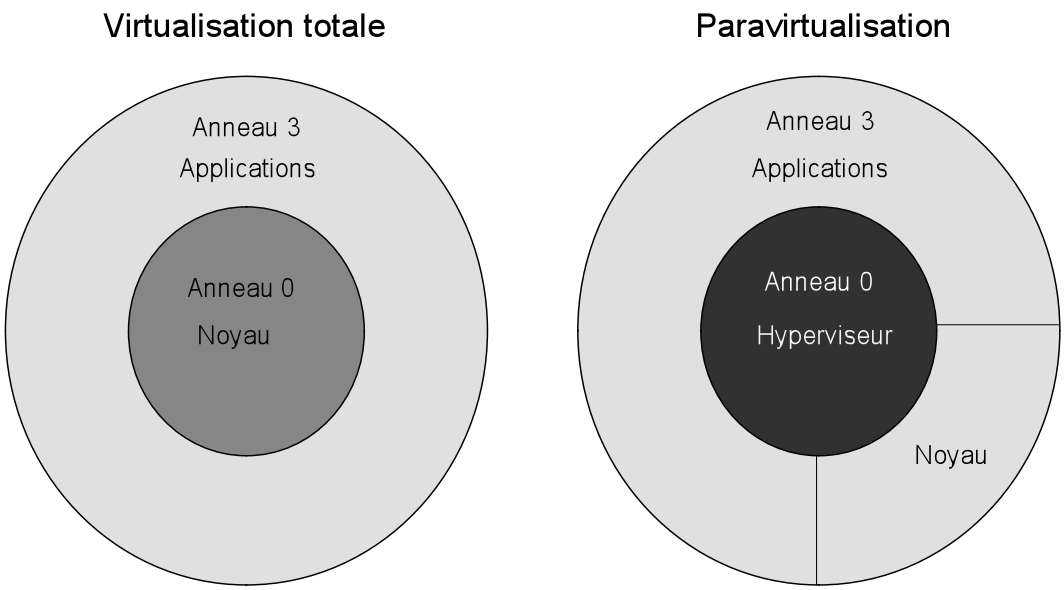

Figure 15 : Rings pour un système Xen x86-64 sans extensions de virtualisation

Une autre interaction primordiale entre l'hyperviseur et les systèmes d'exploitation invités est la gestion du temps. Dans le cas d'un système d'exploitation classique reposant sur un matériel, le système d'exploitation arrive à estimer le temps en utilisant la fréquence du processeur et le nombre d'instructions effectuées. Or dans le cas de la virtualisation, plusieurs systèmes d'exploitation invités se partagent les cycles processeurs et ce, de manière irrégulière puisque pondérée en fonction des besoins momentanés de chacun. Un dérèglement au niveau de l'horloge du système d'exploitation peut poser de graves problèmes au niveau des communications réseau qui nécessitent régulièrement un suivi précis du temps écoulé. Un exemple de ceci est le ping qui mesure le temps écoulé entre le départ du message et son arrivée. D'autres applications en font cependant un usage bien plus critique. Pour palier à ce problème, Xen doit très régulièrement mettre à jour l'horloge interne des machines virtuelles dans le cas de la paravirtualisation ou émuler une horloge système dans le cas de la virtualisation matérielle assistée. (Chisnall 2007)

3.3. Les domaines

Les systèmes d'exploitation invités que ce soit pour la paravirtualisation ou pour la virtualisation assistée matérielle sont appelés domaines dans le cadre de Xen. Dans le cas des autres produits de virtualisation, l'appellation de système d'exploitation invité reste présente. La raison de cette différence est due au fait que tous les systèmes d'exploitation sont virtualisés lorsque Xen est fonctionnel, même celui d'origine qui ne l'était pas avant l'installation de Xen. Or nous avons vu précédemment qu'il est impossible d'accéder à l'hyperviseur directement par le biais d'une interface graphique ou d'une interface en ligne de commande à cause des risques liés à la sécurité. C'est pour cette raison qu'il existe différents types de domaines disposant de différents droits.

3.3.1. Domaine privilégié

Le domaine privilégié est appelé *domain 0* ou plus couramment dom0. Lors du démarrage d'une machine disposant de Xen, la première application exécutée est l'hyperviseur Xen ce qui est normal à la vue de sa situation dans les rings. La seconde application est ensuite le domaine 0.

Le domaine 0 est un domaine essentiel pour Xen. Dans la mesure où il est impossible d'accéder directement à l'hyperviseur pour y modifier des paramètres, le domaine 0 endosse ce rôle à travers des outils et des fichiers de configuration. Le domaine 0 agit également en tant que VMM, c'est-à-dire, gestionnaire des machines virtuelles ou plutôt gestionnaire des autres domaines. Toutes les données de disque des autres domaines sont stockées dans le système de fichiers du domaine 0 qui y a totalement accès.

En ce qui concerne l'interaction avec le matériel, nous avons déjà vu que l'hyperviseur était l'intermédiaire avec le processeur et la mémoire. Les autres périphériques sont gérés par le domaine 0. La gestion des périphériques vidéo, de saisie, de disque ou autres est effectué par des pilotes situés dans le système d'exploitation du domaine 0. Le domaine 0 a un accès exclusif à ces périphériques par défaut. Cependant, les versions récentes de Xen implémentent la possibilité de déléguer à un domaine non privilégié l'accès exclusif à certains périphériques. L'interfaçage avec les domaines non privilégiés se fait par le biais de pilotes standardisés, comme schématisé dans la figure 17 ce qui permet une très grande compatibilité.

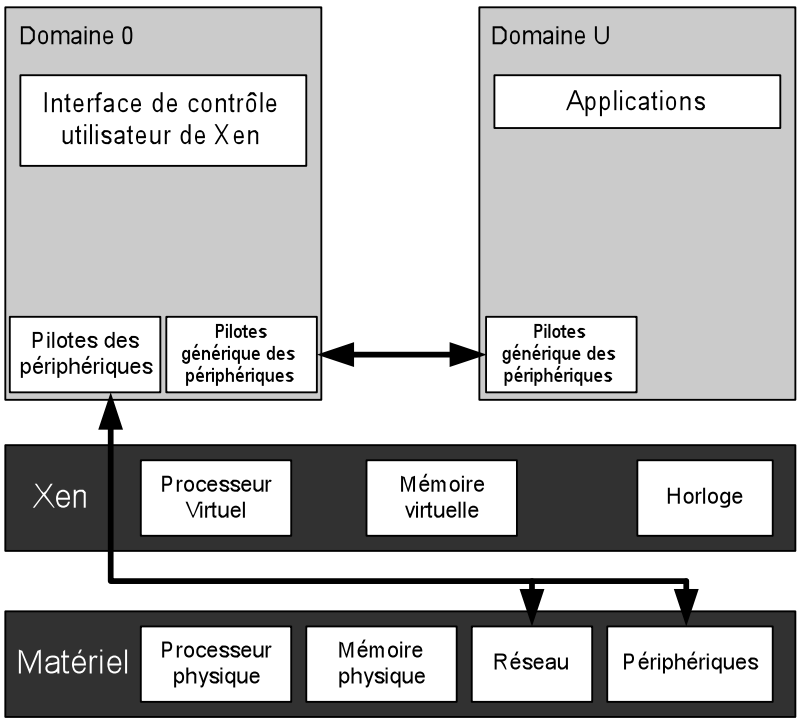

Figure 16 : Interactions domU/dom0/Xen

En ce qui concerne le réseau, le domaine 0 a pour rôle de centraliser toutes les communications. Tous les paquets envoyés à partir des domaines non privilégiés passent à travers la pile réseau du domaine 0 par le biais de diverses interfaces réseau virtuelles. Grace à cette configuration, il est possible de configurer des architectures réseau spécifiques avec les autres domaines tels que du NAT, du routage ou, plus simplement, un pont. Il est également possible de mettre en place des règles de filtrage du trafic réseau en fonction des fonctionnalités et du niveau de sécurisation souhaité.

En pratique, le domaine 0 est particulièrement sensible dans la mesure où il a accès à toutes les données et contrôle tous les autres domaines. La mise en place sécurisée d'un serveur de virtualisation Xen passe donc par une forte sécurisation du domaine 0. De plus, une mauvaise modification au niveau de l'arborescence des fichiers utilisés par Xen rendra le système inutilisable et mènera donc à l'indisponibilité de tous les domaines.

3.3.2. Domaines non-privilégiés

Les domaines non privilégiés sont appelés domaine U ou plus couramment domU avec le U qui signifie *unprivileged*. Les domaines U sont exécutés par le domaine 0 automatiquement au démarrage de la machine la plupart du temps soit manuellement par le biais d'outils. Un nombre théoriquement illimité de domaines non privilégiés peuvent être exécutés dans la limite de la disponibilité de ressources telles que la mémoire vive ou l'espace disque.

En ce qui concerne les périphériques, les domaines non privilégiés y ont accès à travers des pilotes standardisés pour Xen indépendamment du matériel sous-jacent. Le référencement des périphériques auxquels ont accès les domaines non privilégiés se fait par le biais du XenStore que nous évoquerons plus en détail par la suite. De nombreux systèmes d'exploitation ont été modifiés pour rendre possible leur exécution en tant que domaine non privilégié grâce à la simplification du support des pilotes de matériel. La quasi-totalité de systèmes d'exploitation modifiés exécutables en tant que domaine non privilégié de paravirtualisation sont des systèmes d'exploitation de type open source et un support de la part de systèmes d'exploitation propriétaires n'est pas à prévoir avant longtemps.

3.3.3. Domaines matériels assistés

Les domaines assistés matériels sont appelés domaines HVM et sont un type de domaines non privilégiés. La différence avec les domaines non privilégiés classique est le type de virtualisation. Dans le cas d'un domaine HVM, la virtualisation mise en œuvre est la virtualisation assistée matérielle alors que dans le cas d'un domaine non privilégié, il s'agit de la paravirtualisation.

Pour qu'un domaine HVM fonctionne correctement, Xen émule tout le matériel pour que le système d'exploitation hôte puisse fonctionner correctement. Les pilotes qui seront utilisés par le domaine HVM seront des pilotes capables de communiquer avec le domaine 0 dans le but de pouvoir accéder au matériel même de manière indirecte. Le projet Xen a choisi de ne pas redévelopper un module de virtualisation assistée matérielle mais a choisi d'intégrer le projet QEMU. Par ailleurs, bien que le projet QEMU supporte la virtualisation de systèmes d'exploitation non modifiés sans la présence des extensions processeur adaptées, Xen a choisi de ne pas implémenter cette fonctionnalité. Il est donc nécessaire que les extensions processeur de virtualisation soient présentes pour que Xen puisse exécuter un domaine HVM.

Pour améliorer les fonctionnalités et les performances des domaines HVM, il est possible de leur ajouter des pilotes de paravirtualisation ce qui contribue largement à gommer la différence entre la paravirtualisation et la virtualisation assistée matérielle. Ces pilotes vont permettre l'envoi d'hypercalls pour les systèmes propriétaires. De tels pilotes sont disponibles pour Windows sous le nom de *GPLPV Drivers.* Pour communiquer avec les domaines HVM, Xen utilise un périphérique PCI virtuel qui simplifie la communication et le support des pilotes de paravirtualisation similairement à une carte de calcul physique. (Chisnall 2007)

4. L'architecture de Xen en paravirtualisation

4.1. La gestion des informations système

Un système d'exploitation a besoin d'un certain nombre d'informations sur la machine physique sous jacente pour pouvoir fonctionner correctement. Ces informations vont conditionner son fonctionnement par la suite. Des exemples d'informations nécessaires au démarrage sont la quantité de mémoire vive, les périphériques, le type d'architecture ou encore l'heure. Dans le cas d'un système d'exploitation classique installé directement sur une machine physique, ces informations sont fournies directement par le BIOS.

Dans le cas de la virtualisation, un hyperviseur est ajouté entre les systèmes d'exploitation invités et le matériel physique ce qui ne permet pas de gérer les informations système de la même manière que lorsqu'il n'y avait pas d'hyperviseur interposé. La solution va être de modifier le mécanisme de récupération des informations système. Ceci ne pose pas de problème particulier dans le cas de la paravirtualisation puisque les noyaux sont modifiés pour s'adapter à l'environnement de virtualisation. Dans le cas de la virtualisation matérielle assistée, QEMU sera chargé d'utiliser ces informations pour créer l'environnement virtuel. Les solutions qui seront exposées ici ne sont pas valables pour la virtualisation matérielle assistée (HVM). Nous verrons ce cas plus en détail ultérieurement.

Xen met donc en place un mécanisme dénommé *Shared Information Pages*pour pouvoir transmettre ces données aux domaines, que nous traduirons par pages d'informations partagées. Il n'y a pas, au niveau de ces pages, de différence de fonctionnement entre les domaines privilégiés et les domaines non privilégiés. Ces pages d'informations partagées ne reprennent pas totalement les fonctionnalités du BIOS. L'énumération des périphériques système qui est habituellement gérée par le BIOS est remplacée par le mécanisme du XenStore que nous verrons plus en détail ultérieurement. Le seul périphérique qui déroge à cette règle est le périphérique console. Ce dernier a été inclut dans les pages d'informations partagées dans un but de débogage de sorte à pouvoir obtenir des informations sur le lancement des domaines avant qu'ils accèdent au XenStore.

Les pages d'information partagées sont placées dans l'espace d'adressage du domaine à un endroit bien précis par le *domain builder*. Le *domain builder* est le logiciel qui va initier et préparer l'exécution du nouveau domaine en interagissant avec l'hyperviseur. On parlera de page d'information de démarrage dans le cas de la page d'information partagée contenant les informations dont le domaine va avoir besoin lors de son lancement. Dès son exécution le noyau va inspecter cette page pour pouvoir effectuer quelques vérifications élémentaires. Une des vérifications qui sera effectué est un contrôle de l'environnement de virtualisation afin de vérifier si le noyau en cours d'exécution est bien

compatible avec la version de l'hyperviseur Xen installé. Le projet Xen assure une rétrocompatibilité avec les versions majeures à venir mais pas avec les versions mineures. Un exemple de paramètre inscrit dans la page d'information de démarrage est la quantité de mémoire vive allouée au noyau qui sera identifié dans le champ *nr_pages*. En ce qui concerne la gestion du processeur, peu d'informations sont disponibles dans la page d'information de démarrage ce qui fera que le domaine démarrera initialement sur un seul processeur virtuel et passera à plusieurs à la suite de l'initialisation du système d'exploitation.

Le domaine va rapidement avoir besoin de plus d'informations sur son environnement qui ne seront pas présentes dans la page d'information de démarrage. Cette dernière va servir de support pour le référencement des pages d'informations suivantes. Ces références sont des adresses de mémoire physique. Pour accéder à ces informations, le domaine intéressé va devoir émettre un hypercall à l'hyperviseur pour adresser ces zones de la mémoire physique dans ses zones de mémoire virtuelle. La première référence vers une adresse de la mémoire physique de la machine pointe vers la page *shared_info* qui contient de nombreuses informations intéressantes pour la suite de l'exécution du domaine.

La page *shared_info* est utilisée régulièrement lors de l'exécution du domaine. Contrairement à la page d'information de démarrage, elle est mise à jour en permanence. Elle contient des informations relatives à l'horloge du système, à l'architecture et aux canaux d'événements que nous verrons par la suite.

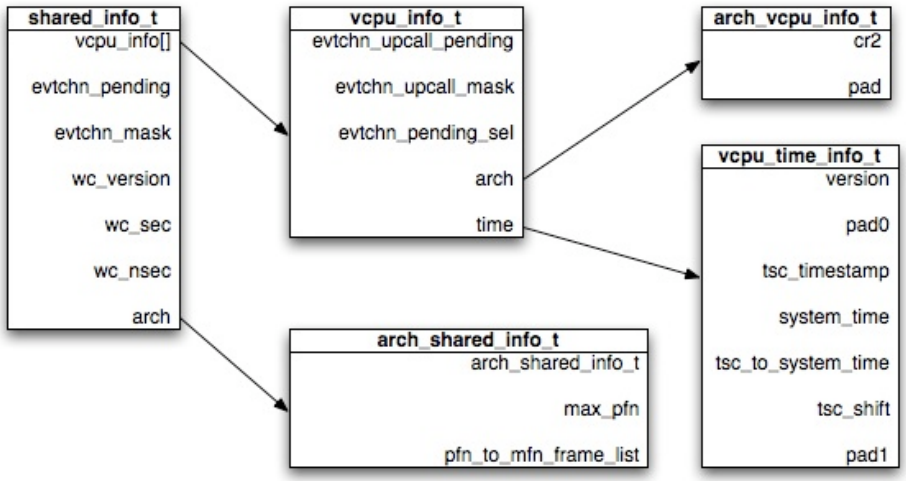

Figure 17 : Page *shared_info*

Le champ *vcpu_info* est un tableau de tableaux d'éléments de type *vcpu_info_t*. Leur nombre dépend du nombre de processeurs virtuels alloués au domaine. Les trois attributs de la table *vcpu_info_t* servent à informer le domaine sur le statut de ses processeurs virtuels afin de pouvoir les gérer. Le domaine prend en compte le fait que le processeur ne lui ait pas dédié et n'exécute pas que

des instructions émanant de lui. Le tableau *arch* permet de passer des informations sur l'architecture de la mémoire telle que l'adresse de la dernière erreur de pagination (*cr2)*. Le tableau *time* de type *vcpu_time_info_t* combiné avec les attributs de préfixe *wc_* permet la gestion de l'horloge système. La table *arch_info_t* contient des informations spécifiques à l'architecture. Dans le cas d'x86, il contient les deux attributs présents dans le schéma qui renseignent des références d'adressage entre la mémoire du domaine et la mémoire physique.

4.2. La communication inter domaines

Précédemment, nous avions évoqué l'isolation des machines virtuelles comme un des avantages de la virtualisation. Maintenant que nous nous intéressons à la virtualisation à un niveau plus bas, il est intéressant de nuancer cette affirmation. Un isolement total des domaines poserait de très sérieux problèmes puisqu'il ne leur serait possible d'accéder à des périphériques systèmes émulés ou non. Il est donc essentiel de prévoir des mécanismes de communication internes entre l'hyperviseur et les domaines mais aussi entre les domaines et plus particulièrement entre les domaines U et le domaine 0.

Dans le cas d'un système d'exploitation sans virtualisation, les processus sont amenés à communiquer entre eux qu'ils fassent parti de la même application ou non. Typiquement, il sera très intéressant de donner la possibilité au noyau de communiquer avec les applications directement pour pouvoir les couper en cas de surcharge du système ou en cas de perte de réponse de l'application. Ceci est fait par le biais de zones de mémoire partagées allouées par le noyau. Dans le cas où une application souhaite communiquer avec une autre, une demande explicite sera faite au noyau qui allouera un espace mémoire commun aux deux applications. Il sera ensuite possible à l'application d'inscrire diverses données dans cette espace à destination de l'autre application. Une fois la communication terminée, cette espace mémoire partagé sera réalloué pour des raisons de sécurité. On parle donc d'IPC (*InterProcess Communication*).

L'hyperviseur Xen a choisi d'adopter un mode de fonctionnement similaire en mettant à disposition de domaines souhaitant communiquer des zones de mémoire partagées. L'utilisation de ces zones de mémoire partagées sera gérée par les domaines communiquant. On parle dans ce cas-là d'IDC (*InterDomain Communication*). Dans un système d'exploitation classique, elles sont dynamiquement réparties entre la mémoire vive et les zones de pagination sur le disque dur (swap). L'hyperviseur Xen a choisi pour des raisons de minimalisme de ne fournir des zones de mémoire partagées uniquement en mémoire vive. La gestion de zones de pagination sur disque est entièrement déléguée aux domaines. Les zones de mémoire partagées sont identifiées par un entier dénommé *grant reference* que nous traduirons par référence d'allocation en Français.

L'interfaçage avec les mécanismes de zones de mémoire partagée de Xen se fait par le biais de l'hypercall *grant_table_op*. Cet hypercall est effectué par le noyau du domaine comme nous l'avons vu précédemment. Cet hypercall prend trois arguments : le type d'opération à effectuer, un tableau

contenant les opérations à effectuer et le nombres d'opérations à effectuer. Nous verrons la structure de la table d'allocation par la suite. Cette fonction est polymorphique dans la mesure où le type du second argument dépend du premier argument. Deux types d'opérations sont possibles avec cet hypercall, l'allocation et le transfert de zones. La différence entre ces deux opérations est qu'avec l'allocation, la zone de mémoire partagée est laissée dans la zone de mémoire globale du domaine originel alors que le transfert la supprime.

Le transfert de zones de mémoire partagées ne peut se faire que dans le cas où le récepteur a manifesté un intérêt à recevoir un transfert. Cette contrainte est mise en place pour sécuriser le transfert des zones et éviter qu'un domaine ne sollicitant pas de partage se voit ajouter des zones de mémoires extérieures dont il ne connait ni la provenance ni la fiabilité qui pourraient contenir du code malicieux. La manifestation de l'intérêt d'un transfert de page se fait par le biais de la table d'allocation du domaine.

Le transfert est un mécanisme efficace de transfert de données volumineuses entre différents domaines puisqu'une fois que les données sont écrites, il n'y a plus qu'à mettre à jour la table d'allocation de la mémoire vive. Ceci est cependant peu efficace dans le cas du transfert de faibles volumes de données. Une solution plus efficace peut être la copie à partir de la zone de mémoire propre au domaine récepteur vers la zone de mémoire partagée. Ceci ne peut être intéressant que dans la mesure où le partage est déjà mis en place puisqu'il n'est pas nécessaire de modifier la table d'allocation. Dans le cas où le partage n'a pas été mis en place, il sera plus intéressant de faire un transfert.

De plus, il est possible de faire un transfert entre deux domaines sans qu'aucun des deux n'ait fait de demande. Par exemple, un domaine peut avoir un accès exclusif à la carte réseau dans le but de limiter l'impact d'une erreur avec ce périphérique. Le domaine 0 sera amené à faire des copies de données entre le domaine responsable de la carte réseau et d'autres domaines nécessitant l'accès au réseau. Tout ceci sera parfaitement transparent pour tous les domaines sauf le domaine 0.

4.3. La gestion du temps

La gestion du temps est un aspect important d'un système d'exploitation. Ce dernier en a besoin constamment pour son bon fonctionnement. Des exemples sont l'exécution de tâches programmées dans le temps, l'ordonnancement des processus, l'affichage d'horloges, l'horodatage de systèmes de fichiers ou encore différents protocoles réseaux. L'utilitaire ping est un bon exemple dans la mesure où il est nécessaire de connaitre l'heure de départ et d'arrivée du paquet avec une précision de l'ordre du dixième de milliseconde. Un problème dans la gestion du temps pourra résulter en un temps de latence négatif ou, inversement, un temps très grand alors que le paquet a été transmis promptement et correctement.

Dans le cadre d'un domaine Xen, le temps peut s'écouler de deux manières différentes : lorsque le domaine a des ressources de calcul à sa disposition et lorsqu'il est mis en attente. Lorsqu'un domaine a des ressources de calcul à sa disposition, il reçoit un signal tous les 10 ms ce qui lui permet de gérer l'ordonnancement des différents processus en cours. On parle donc dans ce cas de gestion du temps virtuel qui correspond au temps d'exécution effectif du système. Le temps virtuel exclue donc toute période d'attente.

Bien que le temps virtuel soit très utile pour l'ordonnancement des processus du domaine, il est inefficace pour d'autres fonctionnalités telles que l'horodatage de fichiers ou les protocoles de communication réseau. Le domaine va donc avoir besoin de connaitre le temps réel. Pour obtenir le temps réel, le domaine va avoir accès à trois valeurs qui sont contenus dans page *shared_info* vue précédemment.

La première valeur est l'heure initiale qui correspond à l'heure du lancement du domaine. Cette valeur servira d'origine pour la suite de la gestion du temps réel. Elle est contenue dans le champ comportant pour préfixe *wc_* (abréviation de *wall clock*). La seconde valeur est l'heure système qui correspond au temps écoulé depuis le démarrage du domaine. Elle est mise à jour dès que des ressources de calcul sont allouées au domaine. Cette valeur est également stockée dans le champ *system_time*de la page *shared_info*. La troisième valeur est le TSC (*Time Stamp Counter*) qui correspond au nombre de "ticks" depuis le démarrage. Il s'agit de l'unité de base du comptage du temps système. Le nombre de ticks est mis à jour tous les cycles d'horloge. Dans le cas de processeurs plus récents tels que le Core 2 Duo d'Intel, le TSC n'est mis à jour qu'une fois tous les quatre cycles processeur. Cependant deux propriétés sont assurées : les valeurs augmentent de manière monotone et l'augmentation est proportionnelle aux cycles du processeur. L'incrémentation du TSC est effectuée par l'hyperviseur et est propagée aux domaines par le biais des pages *shared_info*.

Les deux premières valeurs vont permettre un calcul approximatif mais simplifié de l'heure. Ce calcul suffira dans le cas d'applications nécessitant peu de précision telle que l'affichage d'une horloge.

Ce mode de calcul ne sera cependant pas suffisant dans le cas d'applications nécessitant une forte précision. Dans ce dernier cas, les applications vont pouvoir bénéficier de l'apport du TSC.

Au final, les domaines disposent de deux grandeurs de temps : le temps virtuel et le temps réel. Le temps virtuel est calculé par le domaine de manière autonome. Le temps réel a cependant besoin de l'intervention de l'hyperviseur pour fournir des données plus précises et nécessitant des informations système.

4.4. La gestion de la mémoire

4.4.1. Modèle de gestion de mémoire en para-virtualisation

Dans un système virtualisé on trouve 3 niveaux d'indirection de la mémoire. L'indirection est le fait de ne pas accéder directement à l'espace d'adressage physique. Au niveau le plus haut on retrouve l'application qui est lancé dans un système invité. Celle-ci dispose d'un certain espace d'adressage mémoire. C'est le premier niveau d'indirection. Cette application fonctionnant dans un système invité, celui-ci dispose de son espace d'adressage propre. Nous avons ici le deuxième niveau d'indirection. Enfin puisque nous sommes en paravirtualisation, ce système invité fonctionne au dessus d'un hyperviseur. Celui-ci possède tous les droits pour manipuler la mémoire physique. C'est son rôle de lier la mémoire allouée aux systèmes invités avec la mémoire physique. C'est le 3ème niveau d'indirection.

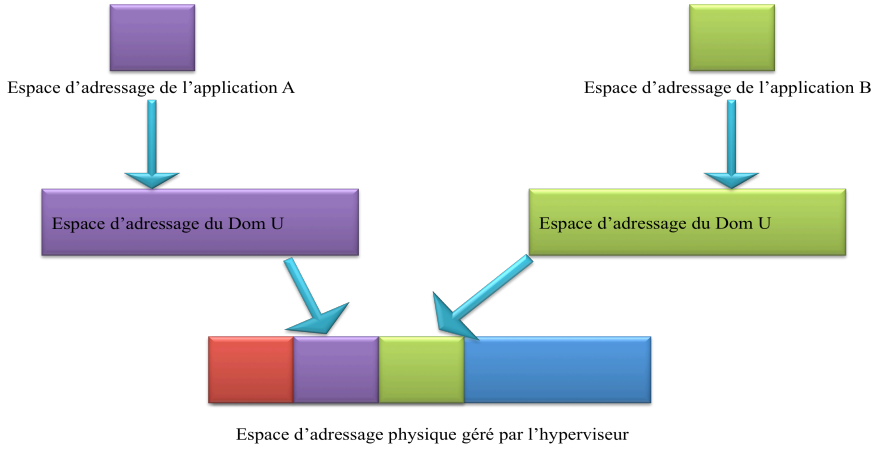

Figure 18 : Couches de gestion mémoire Xen

La question que l'on peut se poser est pourquoi avoir ajouté une couche d'indirection mémoire. En effet il aurait pu être possible que les différents systèmes invités existent dans l'espace d'adressage physique tout en faisant attention à ne pas n'accéder à des segments appartenant à un autre système. Cette solution n'a pas été retenue car les systèmes d'exploitation ont été développés de façon à ce qu'ils soient seuls sur une machine physique. De ce fait ils croient que leur espace d'adressage est continu. La seconde raison qui justifie l'existence d'une 3ème indirection est la gestion de la migration

à chaud. Il est possible sous Xen de sauvegarder l'état complet d'un machine, de la mettre en pause pour de la remettre en fonctionnement sur une autre machine physique. Les adresses physiques des pages mémoire que la machine va se voir ré allouée seront à coup sur différentes. De ce fait on est obligé d'avoir cette couche d'indirection entre les systèmes invités et l'hyperviseur. Ainsi une fois le système migré, le nouvel hyperviseur réalloue un certain espace d'adressage physique qui n'est pas forcement continu au système invité. Celui-ci peut reprendre ces taches une fois que les pages mémoires seront chargées à nouveaux.

Nous avons parlé précédemment des tables LDT et GDT. Dans Xen les systèmes invités ont accès aux LDT qui sont des espaces d'adressages mémoires dédiées aux applications. Mais ils ne peuvent modifier les valeurs des espaces GDT que par un hypercall spécifique. A chaque hypercall Xen doit pouvoir accéder à la mémoire dédiée à l'hyperviseur ainsi qu'à l'espace d'adressage du système invité qui demande l'hypercall. La solution retenue pour remplir cette tache est d'opérer un changement de contexte à chaque hypercall. C'est à dire que l'on met en attente d'exécution un processus par le processeur. Ensuite on redirige les demandes d'accès aux pages d'adresses mémoire du système invité vers l'espace d'adressage de l'hyperviseur comme le demande ce système via l'hypercall.

Ainsi grâce à l'hypercall on permet a système invité d'accéder à des informations qu'il n'est pas censé obtenir sans augmentation de privilèges. Mais ceci n'est pas une tache courante et heureusement car c'est très gourmant en ressources. Ceci peut nécessiter 500 cycles processeur sur un Pentium 4. Afin d'adresser ce type de demande, sur un système x86-32Bits, Xen se réserve les 64 premiers Méga Octets de la mémoire vive. Les descripteurs de segments qui permettent l'accès à ce type de mémoire ont le champ de privilèges à 0 pour n'autoriser l'accès qu'a l'hyperviseur.

Les descripteurs de segments mémoire alloués aux noyaux dans un système Xen retrouvent le champ lié aux privilèges à 1 dans ces descripteurs de segments. La lecture des informations est possible par les systèmes invités mais pas leur modification. Seul le noyau et l'hyperviseur peuvent modifier ces informations.

Pour résumer, lorsqu'un système invité désire accéder à des tables de pages (qui permettent l'accès à la mémoire physique) qui ne sont pas dans son anneau de permissions, un hypercall va permettre d'effectuer ce lien entre le système invité et l'hyperviseur. Hyperviseur qui fera cette requête par la suite à destination de la MMU. (Chisnall 2007)

4.4.2. Mise à jour de la table des pages

Comme dit précédemment en conclusion, lorsqu'un système invité « aussi appelé guest » va vouloir mettre à jour ses tables de pages il devra passer par un hypercall. Plus précisément pour tout accès en lecture dans l'espace d'adressage qui lui est propre, le guest n'aura pas besoin de faire un hypercall ces manipulations de lecture sont autorisées. Néanmoins pour tout autre accès, on pense bien évidemment à la mise à jour de ces propres tables de pages il devra effectuer un hypercall. Cela permet d'être sur qu'aucun système invité ne puisse modifier un espace d'adressage sans mécanisme d'hypercall validé par l'hyperviseur.

En effet un guest ne peut modifier ces tables directement, mais un hypercall effectuera cette demande de mise à jour si celle-ci est fondée. Toute demande de modification d'espace mémoire qui n'appartient pas au guest se soldera par un échec.

Du fait qu'un hypercall et bien plus gourmand en ressources qu'un simple accès mémoire on va pouvoir passer dans un seul hypercall la mise à jour de multiples pages.

La fonction en C disponible pour se faire se présente de la forme :

```
HYPERVISOR_mmu_update( mmu_update_t * req, int count, int * sucess_count) ;
```

Ou mmu_update est la structure suivante :

```
Struct mmu_update {
        Uint64_t        ptr ;     /*Adresse*/
        Uint64_t        val ;     /*Contenu*/
}
```

La fonction HYPERVISOR_mmu_update prend en paramètres un tableau de structures de type mmu_update. Elle prend également le nombre d'occurrences de cette structure et donne en retour le nombre d'occurrence qui ont abouties sur un succès.

La structure mmu_update contient deux valeurs qui sont respectivement l'adresse de la table de page à modifier ainsi que la nouvelle valeur à attribuer.

4.4.3. La création d'un DomU : l'aspect mémoire

A l'initialisation d'un système d'exploitation invité, certaines choses doivent être chargées dans sa mémoire. C'est le Dom0 qui sera chargé de cette tache vu que c'est par son intermédiaire que l'on créée un nouveau DomU. Les informations à charger en mémoire sont le noyau du guest et la *start info page* qui est l'équivalent de son bios (cf 4.1.1). La mémoire doit également contenir les accès nécessaires à la lecture de la table des pages. Les instances de chaque pilote doivent également

être chargées dans la mémoire. Un Hypercall utilisé depuis le Dom0 appelé HYPERVISOR_memory_op permet de configurer tous ces paramètres. Il permet également de définir l'espace mémoire initial d'un DomU.

4.4.4. Les gestions des fautes de pages

Un défaut de page correspond à une série d'événements se déroulant lorsqu'un programme essaie d'accéder à des données ou à un code qui se trouvent dans son espace d'adressage mais ne sont pas actuellement placées dans la mémoire vive. Le système d'exploitation doit traiter les défauts de pages en permettant, d'une manière ou d'une autre, l'accès à la mémoire des données recherchées afin que le programme puisse continuer ses opérations, comme si le défaut de page ne s'était jamais produit. (Red Hat Inc s.d.)

La première chose à faire en cas de faute de page est de trouvé l'adresse mémoire qui a causé cet évènement. Sous x86 ces informations sont stockées dans un registre nommé CR2. Sous Xen ces informations sont copiées dans l'élément *cr2* de la structure *arch_vcpu_info* (cf 4.1.1). Cette opération de copie est nécessaire car le mécanisme de stockage des fautes de pages accède à la mémoire et peut lui aussi générer des fautes de page. Si cela se produit, le contenu du registre CR2 est écrasé. Une fois l'adresse mémoire qui a causé la faute de page retrouvée, il reste à mettre à jour son contenu. Une fois fait l'hypercall HYPERVISOR_update_va_mapping va lier l'adresse virtuelle avec la nouvelle adresse physique. (Chisnall 2007)

4.5. La gestion de l'ordonnancement

Nous avons déjà vu précédemment que le principe de la virtualisation consiste à exécuter plusieurs systèmes invités sur une même machine physique. Cette propriété essentielle implique la gestion de l'allocation de ressources de calcul parmi tous les domaines. L'hyperviseur est responsable d'assurer cette répartition de ressources. Cette répartition est très similaire à la répartition de ressources parmi des processus dans un système d'exploitation classique.

Un processus présent dans un domaine va donc être amené à être géré par plusieurs ordonnanceurs avant de pouvoir accéder à des ressources de calcul. Un ordonnanceur (*scheduler* en anglais) est un algorithme qui va gérer selon divers critères l'allocation des ressources de calcul à une entité demandeuse. Cette entité demandeuse peut être un processus comme dans le cas d'un système d'exploitation classique ou bien un domaine dans le cas de Xen.

Un processus exécuté dans un domaine va tout d'abord être géré par l'ordonnanceur utilisateur qui a pour but d'envoyer le processus vers l'ordonnanceur du noyau. Celui-ci va ensuite répartir les processus parmi les différents processeurs virtuels alloués au domaine (*VCPU*). Finalement,

l'hyperviseur va répartir les processeurs virtuels parmi les processeurs physiques. Etant donné que l'ordonnanceur se retrouve au fond de la pile, il est nécessaire qu'il soit prévisible puisque les ordonnanceurs au dessus vont être amenés à faire des prévisions sur l'allocation de ressources.

Dans les versions actuelles de Xen, nous pouvons retrouver deux ordonnanceurs : le SEDF (*Simple Earliest Deadline First*) et le *Credit Scheduler* que nous pouvons traduire par ordonnanceur à crédit. Ce dernier a tendance à prendre le dessus du SEDF et est activé par défaut.

L'ordonnanceur SEDF peut se traduire par ordonnanceur du plus urgent d'abord. Dans le cas d'un ordonnanceur de processus dans un système d'exploitation, il n'est pas possible de prévoir le temps d'exécution d'une tache ou la périodicité de son exécution. Dans le cas du SEDF, le temps d'exécution et la périodicité sont définis. Par exemple, un domaine va se voir alloué 5 ms de calcul toutes les 10ms.

Cet algorithme d'ordonnancement a le mérite d'être parfaitement déterminé et donc tout autant prévisible. Il comporte cependant un inconvénient de taille. Si un domaine n'a pas besoin de sa part de ressources, elle ne sera pas réalloué à un autre domaine. Ceci résulte en une sous consommation de ressources processeur alors qu'un domaine n'en aura pas suffisamment pour tout exécuter à temps. Une autre problématique de cet algorithme se pose dans la configuration suivante[1] :

- Domaine 1 : 20 ms toutes les 100 ms
- Domaine 2 : 2 ms toutes les 10 ms
- Domaine 3 : 5 ms toutes les 10 ms

Les domaines 2 et 3 se verront alloués des ressources de manière prioritaire puisque leur exécution est la plus urgente étant donné qu'ils doivent être exécutés dans les 10 ms à venir. Entre ces deux domaines, le domaine 3 sera exécuté en premier puisque le domaine 2 pourra encore attendre 8 ms. Il arrivera un moment où le domaine 1 sera le plus urgent à exécuter. Il se pose donc un problème car si le domaine 1 est exécuté entièrement, les domaines 2 et 3 ne pourront pas obtenir les ressources dont ils ont besoin. La solution à ce problème est tout simplement de raccourcir le temps d'exécution du domaine 1 et de prévoir une répartition du temps d'exécution du domaine 1 sur le temps laissé libre par les deux autres domaines.

L'ordonnanceur nommé *Credit Scheduler* que nous pouvons traduire par ordonnanceur à crédit est celui par défaut des versions actuelles de Xen. Pour qu'il puisse fonctionner, il est nécessaire d'associer deux attributs à un domaine : le poids et la limite. Le poids détermine la part des ressources de calcul qui sera allouée au domaine. La limite détermine la part maximale des ressources qui pourra être allouée au domaine. Il est tout à fait possible de ne spécifier aucune limite. Lorsqu'aucune limite n'est spécifiée, cet ordonnanceur permet d'utiliser au maximum les ressources du processeur. A l'inverse, si

[1] Nous supposerons dans cet exemple que chaque domaine ne possède qu'un seul VCPU

la somme des limites est inférieure à la totalité des ressources de calcul disponibles, le processeur sera sous utilisé. Cet algorithme utilise un mélange d'autres algorithmes tels que le tourniquet et l'ordonnancement à priorité.

Pour pouvoir expliquer plus clairement le fonctionnement de cet algorithme d'ordonnancement, nous allons prendre l'exemple suivant :

- VCPU 1 : poids 64, limite 25%
- VCPU 2 : poids 64, pas de limite
- VCPU 3 : poids 128, pas de limite

Au début de l'exécution, chaque domaine se verra alloué un nombre de crédits proportionnel à son poids. Les deux premiers VCPU auront par exemple 64 crédits alors que le troisième VCPU aura 128 crédits. L'ordonnanceur va ensuite allouer des ressources à chaque VCPU selon la méthode du tourniquet. Les deux premiers domaines vont donc effectuer leurs calculs chacun leur tour jusqu'à l'épuisement de leurs crédits. Les crédits sont décrémentés de manière régulière et linéaire tous les 10 ms. Le troisième domaine va ensuite avoir le processeur pour lui seul jusqu'à épuisement de ses crédits.

Dans le cas où le troisième domaine n'aurait pas de calcul à effectuer, les deux premiers domaines se partageront le processeur de manière équitable jusqu'à ce que le premier VCPU atteigne sa limite. Le second VCPU sera donc le seul à utiliser le processeur. Il sera donc amené à utiliser plus de crédits que ce qu'il lui a été alloué initialement mais vu qu'il est seul à vouloir accéder au processeur cela ne pose pas de problème. Des nouveaux crédits sont alloués à chaque tour du tourniquet. Si une allocation de crédit a lieu lorsque le premier VCPU est toujours bloqué par sa limite, les crédits disponibles seront répartis entre les autres VCPU. Une fois que le troisième VCPU souhaiterait à nouveau soumettre des calculs au processeur, le second VCPU ne pourra plus dépasser son montant de crédit et les ressources de calcul seront allouées équitablement entre ces deux VCPU.

Au final, nous avons explicité la nécessité pour l'hyperviseur Xen de disposer d'un ordonnanceur pour répartir les ressources de calcul de manière dynamique entre les différents domaines et nous avons expliqué le fonctionnement des deux ordonnanceurs présents dans les versions actuelles de Xen. La problématique des applications synchrone et temps réel se pose donc. Par nature, la virtualisation ne favorise pas ces applications puisque le temps de calcul est réparti parmi plusieurs environnements logiciels ce qui ajoute un temps de latence entre les allocations de ressources. Une solution va être de faire tendre le ratio de VCPU par processeur physique vers 1.

4.6. Le réseau sous Xen

Le fonctionnement du réseau sous Xen utilise la méthode expliquée en 4.1.2. En effet la communication inter domaine est utilisée dans la communication réseau dans Xen.

Toutes les communications réseau des DomU passent per le Dom0. Deux tunnels sont établis entre le DomU qui veut communiquer avec l'extérieur et le Dom0. Le premier est un tunnel de service et le second de données. Ces tunnels sont en réalité des zones de mémoire partagées entre les domaines, à noter que celles-ci sont distinctes. Le mécanisme de communication réseau inter domaine utilisé dans Xen est identique aux autres mécanismes de communication système. La seule différence entre ce mécanisme de communication et les autres, est le fait qu'ici il y a deux zones mémoire par type de tunnel, une pour l'émission et une autre pour la réception.

Le tunnel de service comprendra les données qui vont permettre de localiser dans les pages mémoires les « données brutes réseau » à transmettre.

Lorsque le DomU veut transmettre des informations sur le réseau, il va faire une demande dans un espace mémoire. L'action est faite via le « tunnel de service ». Une fois la demande traitée par le Dom0, l'espace mémoire précédent va être remplacé par sa réponse. Une fois l'échange fait, le paquet réseau peut être émit via le « tunnel de données » puis retransmis sur le réseau via le Dom0.

Lorsqu'un paquet réseau arrive sur l'interface de la machine physique, l'information arrive systématiquement au Dom0. Celui-ci adresse le paquet au DomU qui convient, ceci par l'intermédiaire d'une zone mémoire commune.

La communication réseau entre les DomU d'une même machine physique est gourmande en ressources processeur. Pour cela un allègement des trames de la couche liaison a été effectué. Le champs checksum qui est le dernier champ de la trame réseau n'est jamais calculé pour des communications réseau inter domaines. Ainsi une puissance de calcul non négligeable est préservée, assurant ainsi de bonnes performances dans les transferts réseau inter domaines. La disparition de ce champ de contrôle n'influe pas sur la fiabilité des transmissions réseaux puisque toutes les communications ont lieu en mémoire.

Toutes les communications réseau passent par le DomO. Concernant celles qui vont à l'extérieur de la machine, la segmentation de niveau réseau et transport sera assurée par la carte réseau. Concernant les communications inter domaines, aucune segmentation n'est effectuée. Ceci pour le simple fait que tout est écrit et lu en mémoire et que la segmentation des paquets n'est plus nécessaire.

Conclusion générale

La première partie nous a permis de faire un tour d'horizon des différentes techniques de virtualisation de systèmes d'exploitation. Ceci nous a permis d'établir un arbre de classification de ces dernières.

La seconde partie nous a ensuite permis d'établir les notions nécessaires à l'étude en profondeur du fonctionnement d'un système d'exploitation. Grace à ces notions, nous avons pu étudier le projet Xen. Nous avons étudié le fonctionnement de divers composantes telles que la mémoire, le temps, la communication entre les domaines et le réseau.

Ce rapport nous a permis de prendre connaissance des différentes techniques de virtualisation ainsi que leurs spécificités. Ces connaissances pourront être directement appliqués à des choix d'intégration et de conception dans le milieu professionnel. L'étude du projet Xen nous a permis d'approfondir nos connaissances en système d'exploitation et de virtualisation.

L'étude d'un domaine si spécifique a nécessité un effort de clarification permanent et de rédaction. Ces compétences que nous avons développées durant ce projet seront directement applicables en milieu professionnel.

Etant donné la prise de conscience globale vis à vis de l'efficacité énergétique, les solutions de virtualisation vont avoir tendance à séduire du plus en plus de professionnels. Le contexte économique actuel et l'optimisation des coûts feront ressortir la virtualisation en tant que solution efficace. L'étude approfondie de ce sujet nous a permis de prendre conscience et d'acquérir des connaissances sur ce sujet d'avenir.

Travaux cites

Bonnet, Lucas. *État de l'art des solutions libres de virtualisation pour une petite entreprise.* 01 2008.

Chisnall, David. *The Definitive Guide to the Xen Hypervisor.* 2007.

Intel. *Intel-VT Technologie.* http://www.intel.com/cd/business/enterprise/emea/fra/technologies/275248.htm (accessed 10 2008).

—. *Intel-VT Technologie.* http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf (accessed 10 2008).

Nakajima, Jun. *Intel, Hybrid Virtualization.* http://www.valinux.co.jp/documents/tech/presentlib/2007/2007xenconf/Intel.pdf (accessed 10 2008).

Primet/Vicat-Blanc, Pascale. "Hipcal :State of the Art of OS and Network virtualization solutions for Grids, page 7-10." 14 Septembre 2007.

Red Hat Inc. *Mémoire virtuelle informations détaillées.* http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-isa-fr-4/s1-memory-virt-details.html (accès le 12 2008).

Tanenbaum, Andrew. *Systèmes d'exploitation.* 2004.

Vayssade, Michel. *Système d'exploitation, mémoire virtuelle et pagination.* http://www.unit.eu/ori-oai-search/notice.html?id=unit-ori-wf-1-651&format=lom_id&printable=true. Compiègnes, 2004.

whatis.com. *Monitor, Virtual Machine.* http://searchservervirtualization.techtarget.com/sDefinition/0,,sid94_gci1190932,00.html (accessed 10 2008).

Wikipédia. *Les anneaux et la sécurité des ordinateurs.* http://en.wikipedia.org/wiki/Ring_(computer_security) (accessed 11 2008).

—. *Virtualisation, Wikipédia Full.* http://en.wikipedia.org/wiki/Full_virtualization (accessed 10 2008).

—. *Virtualization, Wikipédia Hardware Assisted.* http://en.wikipedia.org/wiki/Native_virtualization (accessed 10 2008).

—. *Virtualization, Wikipédia Operating System-Level.* http://en.wikipedia.org/wiki/Operating_system-level_virtualization (accessed 10 2008).

—. *Virtualization, Wikipédia Partial.* http://en.wikipedia.org/wiki/Partial_virtualization (accessed 10 2008).