Consolidation de serveurs avec Linux VServer & Vmware ESX

Stéphane Larroque, Xavier Montagutelli

Consolidation de serveurs avec Linux VServer & VMware ESX

Stéphane Larroque
INSA de Toulouse - CRI
135 avenue de Rangueil
31077 Toulouse Cedex 4
stephane.larroque@insa-toulouse.fr

Xavier Montagutelli
Université de Limoges - SCI
123 avenue Albert Thomas
87060 Limoges
xavier.montagutelli@unilim.fr

Résumé

Notre expérience en consolidation de serveurs a débuté avec l'utilisation de Linux VServer, dans le cadre du regroupement sur une même machine physique de plusieurs serveurs web. Appréciant la souplesse apportée pour le déploiement de nouveaux systèmes, et souhaitant étendre ce principe afin de limiter l'accroissement du nombre de machines dans nos salles techniques, nous avons opté pour VMware ESX, qui permet l'exécution concomitante de plusieurs systèmes d'exploitation.

Cet article présente quelques enjeux liés à la consolidation de serveurs, dresse un panorama succinct des différentes techniques de virtualisation sur les machines à base de processeurs x86 et s'intéresse de plus près à deux d'entre elles : Linux VServer et VMware ESX. Leur architecture et leur mise en œuvre (installation, administration, supervision) sont détaillées. Les usages conseillés, les capacités et les limites de ces deux solutions sont également abordés.

Mots clefs

Consolidation, virtualisation, VMware, VServer.

1 Introduction

Le nombre de machines dans nos établissements ne cesse de croître, allant de pair avec un coût de possession (TCO) élevé. Nous mettons tous en œuvre diverses stratégies afin d'abaisser ce TCO mais elles concernent le plus souvent les postes des utilisateurs qui représentent la plus grande part de nos parcs. La même problématique se pose pour nos serveurs qui sont en constante augmentation.

Nous évoquerons les raisons de cette prolifération afin d'entrevoir quels peuvent être les bénéfices attendus d'une consolidation. Nous nous intéresserons ensuite à une méthode de consolidation, la virtualisation, dont nous exposerons quelques aspects théoriques afin de réaliser une classification rapide des différentes solutions existantes. Enfin nous aborderons sous les mêmes angles deux « environnements » de virtualisation : Linux VServer et VMware ESX.

2 La consolidation : effet de mode ?

Il est devenu difficile d'échapper à ce terme qui revient régulièrement dans les articles de revues informatiques spécialisées ou dans les présentations « marketing » que réalisent les acteurs du marché. Nous allons tenter de découvrir ce qui se cache derrière ce concept après en avoir étudié l'une des causes : l'augmentation du nombre de serveurs.

2.1 La prolifération du nombre de serveurs

La quantité de données numériques et les traitements informatiques associés connaissent une progression toujours plus forte et ininterrompue ; il s'agit là d'une des raisons de l'augmentation du nombre des serveurs mais ce n'est pas la seule. Nous allons essayer de déterminer les autres causes et les conséquences qui en découlent.

Un nombre sans cesse croissant d'applications généralistes ou spécialisées (métiers) voient le jour afin de répondre à nos attentes, chacune d'entre elles possédant des spécificités nécessaires à son exécution (version de librairies, moteur de base de données, etc.) qui les rendent incompatibles les unes avec les autres. D'ailleurs, nombre d'éditeurs, par souci de simplification – mais également parfois de tranquillité – exigent pour leurs logiciels une machine dédiée avec des caractéristiques techniques (processeur, mémoire, etc.) bien au delà des réelles nécessités.

Nous-mêmes sommes amenés (souvent par manque de temps) à déployer une machine par application car l'implantation est facilitée et le débogage en cas de problème est plus aisé. Ce gain de temps est parfois apparent et constitue un calcul à court terme.

Cette séparation possède malgré tout l'avantage de permettre d'effectuer des mises à jour, à la fois du système et de l'applicatif, en n'interrompant qu'un seul service et en évitant d'impacter une autre application qui dépendrait d'un composant modifié.

La maxime « une application par serveur » semble donc se vérifier de plus en plus.

Les contraintes sécuritaires peuvent également exiger que certaines applications soient isolées sur une machine (ou plusieurs dans le cas d'une application multi-tiers) et un

réseau indépendants. La sensibilité des données manipulées, leur confidentialité, la garantie de leur intégrité rentrent toutes dans ce cadre.

Les plates-formes de test et de développement contribuent elles aussi grandement à cette croissance. En effet, le portage d'un logiciel à des architectures, des systèmes d'exploitation, des environnements différents implique de posséder une machine de chaque type.

L'informatique occupant une place de plus en plus critique, elle suppose des performances suffisantes et une disponibilité sans cesse accrue. Rendre les services (et donc souvent les machines) redondants est devenu inévitable et la mise en place de grappes d'ordinateurs – que ce soit pour du calcul ou un service fortement sollicité – se banalise.

La mise en place de solutions, comme les clients légers, pour réduire le TCO dans nos parcs a également amené une recrudescence du nombre de serveurs.

Cette prolifération de serveurs n'est pas sans conséquences.

Elle a tout d'abord des répercussions sur nos salles techniques dont le volume n'est pas extensible et dont les climatisations et protections électriques (onduleurs, groupes électrogènes) doivent être capables d'absorber « la charge ». Le nombre de ports réseau et le matériel actif doivent être correctement dimensionnés pour assurer la connexion de ces serveurs en tenant compte de la redondance. La problématique est la même pour un SAN, au coût près.

La mise à disposition de nouvelles ressources est consommatrice de temps et d'énergie : commande, réception, montage, installation du système, déploiement de l'applicatif. Le matériel n'étant pas forcément homogène, il n'est pas toujours possible (ou facile) d'automatiser le processus d'installation. Cette difficulté est également présente dans l'administration au quotidien, par exemple lors des mises à jour.

Les solutions déployées même si elles sont théoriquement évolutives, sont rarement extensibles facilement ce qui rend l'adaptation « à la demande » assez inefficace.

Même si cela a été partiellement occulté jusque-là, tout ceci possède un coût non négligeable au niveau financier : achat des serveurs et des matériels induits par ceux-ci, contrats de maintenance et de support, consommation électrique, etc.

Le pire dans tout cela est que si vous réalisez un audit de vos serveurs vous constaterez qu'ils sont très probablement sous-utilisés et que le gaspillage de ressources est criant.

2.2 Un remède : la consolidation

La diminution des coûts passe par la maîtrise de la prolifération des serveurs et une meilleure mutualisation des ressources. Il s'agit de l'idée qui se cache derrière le terme de consolidation. Ce concept issu du monde économique désigne le fait d'agréger les comptabilités d'un groupe de sociétés afin d'en calculer les états financiers en simulant une entité unique. D'une façon plus générale, il s'agit de la fusion de plusieurs choses en une seule. Appliqué au monde de l'informatique, cela consiste à rassembler plusieurs ressources en une seule, principe qui s'adapte aussi bien au monde des serveurs que du stockage.

Pour la problématique qui nous intéresse, l'objectif est de tirer parti du sous-emploi des ressources (processeur, mémoire, disque et réseau) des serveurs qui peuvent être partagées afin d'optimiser leur utilisation. Ce processus de consolidation peut se révéler plus ou moins complexe à mettre en œuvre selon l'approche choisie.

2.3 Entreprendre une démarche de consolidation

Avant toute chose, la question est : quel but recherchez-vous ? Plus vous saurez y répondre de manière précise, et plus la démarche de consolidation sera facile à définir. La réponse aidera à choisir la solution technique et les serveurs qui passeront dans le monde virtuel.

Comme dans beaucoup de projets, une multiplication des buts peut être nocive ; trouvez une ou deux idées directrices qui sous-tendront votre action.

2.4 Bons candidats

D'une façon générale, la consolidation de serveurs se heurte à des limites physiques : un serveur nécessitant beaucoup de ressources CPU ou des accès disques importants sera certainement plus à sa place sur une machine dédiée. Ainsi, sauf à étudier une architecture particulière, un serveur de fichiers, un serveur de calcul, un serveur de bases de données (s'ils sont intensivement sollicités) n'apprécieront pas la consolidation.

Inversement, toute autre machine pourrait représenter un candidat potentiel à la consolidation. Mais avant de faire un choix, voici quelques pistes de réflexion.

- Connaissez-vous vos serveurs ? Il faut évaluer suffisamment bien les ressources consommées par chaque serveur pour mesurer son aptitude à la consolidation. Cette connaissance doit inclure des valeurs moyennes, mais aussi les pics que la machine absorbe.
- Connaissez-vous vos administrateurs systèmes ? La démarche de consolidation peut susciter certaines craintes : « cela rajoute une couche », « est-ce fiable ? », « je suis dépossédé de ma machine » par exemple.

3 Techniques de virtualisation

La virtualisation est une famille de technologies qui combinent ou divisent les ressources d'une machine pour présenter un ou plusieurs environnements d'exécution, en utilisant un partitionnement matériel ou logiciel, une simulation partielle ou complète de machine, de l'émulation, etc. La virtualisation peut intervenir à plusieurs niveaux dans les couches composant une machine (Figure 1).

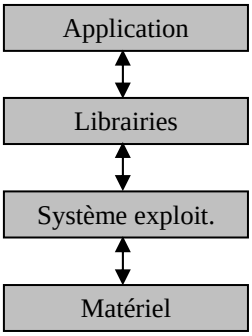

Figure 1 - *Couches logicielles d'accès à une machine*

Nous présentons succinctement dans cette section différentes méthodes de virtualisation, car ce concept est utilisé dans les solutions de consolidation que nous explorerons dans la suite de cet article.

Nous nous cantonnons aussi aux machines à base de processeurs x86, qui constituent maintenant la plate-forme de choix pour les serveurs « bas de gamme ». Mais avant de rentrer dans la classification proprement dite, nous montrons rapidement comment un processeur x86 exécute le code qui lui est fourni pour comprendre ce qui rend difficile la virtualisation sur ce type de processeur.

3.1 L'exécution du code dans les processeurs de la série x86

Les processeurs x86 possèdent deux modes de fonctionnement[1] :

- le mode **réel** conservé pour des raisons de compatibilité avec le 8086 (MS-DOS l'utilisait). Le processeur se trouve dans ce mode à l'allumage de la machine ;
- le mode **protégé** qui tire son nom des mécanismes de protection (de la mémoire, des programmes) qui y ont été ajoutés.

Nous nous intéressons ici à ce dernier mode, utilisé par tous les systèmes d'exploitation modernes. La protection est assurée par quatre niveaux de privilège [1] représentés par des anneaux concentriques ou anneaux de protection (Figure 2).

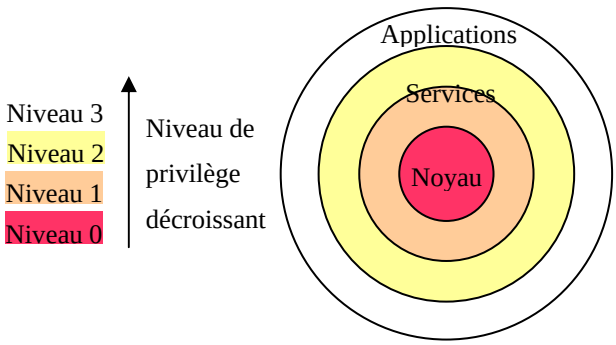

Figure 2 - *Anneaux de protection*

Le code de niveau 0 peut exécuter tout type d'instruction (gestion de la mémoire, des interruptions, changement d'état du processeur, etc.). Il est donc généralement réservé au noyau du système d'exploitation. Les applications utilisateur appartiennent en principe au niveau 3 (le moins privilégié) et certaines instructions leur sont interdites. Les deux anneaux intermédiaires sont plus rarement utilisés.

La protection réside dans le fait que pour accéder à des données ou appeler du code d'un niveau inférieur (0 par exemple), il faut passer par un point de contrôle : la porte (*gate*). Si ce mécanisme n'est pas employé et qu'une instruction privilégiée (i.e. qui nécessite un niveau de privilège plus élevé) est exécutée, une exception de type General Protection est générée par le processeur.

Or nombre de solutions de virtualisation s'appuient sur un hyperviseur ou moniteur de machine virtuelle (VMM) qui partage les ressources entre les divers environnements d'exécution. Ce moniteur s'exécutant en niveau 0, les machines virtuelles sous son contrôle sont dans un niveau moins privilégié.

Pour que ce dispositif fonctionne, il doit prendre en compte les différents types d'instructions que le processeur peut rencontrer :

- les instructions privilégiées qui nécessitent que le niveau de privilège soit 0 pour pouvoir être exécutées ;
- les instructions non privilégiées qui peuvent être traitées quel que soit le niveau de privilège courant ;
- les instructions sensibles qui bien que non privilégiées ont un comportement non souhaitable dans un contexte de virtualisation (consultation ou modification de l'état de la machine, accès à la mémoire, etc.).

Ce sont ces 17 dernières instructions [2] qui rendent les processeurs x86 non strictement virtualisables. En effet le code qui ne nécessite pas de privilège et qui ne risque pas de compromette la sécurité des autres machines virtuelles doit être, pour des raisons d'efficacité[2], directement exécuté par le processeur. Tout code qui déroge à ce principe (cas des instructions privilégiées et sensibles) doit être pris en charge par l'hyperviseur.

En tenant compte de cela, il est possible d'opter pour :

- la modification du code source du système d'exploitation hébergé pour éliminer toute instruction sensible voire privilégiée (cf. para-virtualisation) ;
- l'interception de toute instruction privilégiée ou sensible pour la faire exécuter par le VMM. Les instructions privilégiées ne posent pas de problème puisqu'elles génèrent, comme nous l'avons vu précédemment, des exceptions. En revanche ce n'est pas le cas des instructions sensibles, ce qui rend leur traitement beaucoup plus délicat. Diverses solutions existent pour prendre en compte cet écueil parmi lesquelles la détection et traduction à la volée du code problématique avec mise en cache des sections traduites (cf. virtualisation de machines).

Nous venons de voir le rôle de l'hyperviseur vis à vis de l'exécution du code dans le processeur. La virtualisation demande un travail plus important, car il faut prendre en

[1] depuis le 80286.

[2] Popek & Goldberg en font une caractéristique du VMM dans [3].

compte la gestion de la mémoire, des entrées-sorties, de l'horloge, etc. Autant d'aspects que nous ne traitons pas dans cet article.

3.2 Virtualisation au niveau matériel

Le concept assez général de « machine virtuelle » est apparu dans les années soixante chez IBM. Cette solution utilise une couche d'abstraction logicielle offrant une vue d'un matériel ressemblant à une vraie machine. La machine virtuelle ainsi obtenue comporte le système d'exploitation complet et les applications. Elle est alors appelée système invité (*guest*). Le système « réel » offrant la virtualisation est logiquement désigné comme l'hôte.

Plusieurs sous-familles peuvent être distinguées :

- L'émulation, avec des produits comme *Bochs*, *QEMU*, *xmame*. Toutes les instructions normalement interprétées par le processeur sont en réalité traduites par un logiciel, de façon dynamique. L'émulateur fonctionne comme un processus utilisateur (niveau de privilège 3 du processeur) et s'appuie sur le noyau de l'hôte (niveau 0). L'émulation demandant une traduction du code, les émulateurs sont lents. Ceci les exclut du champ de la consolidation. Cependant, le processeur émulé peut être différent du processeur hôte, ce qui peut se révéler intéressant dans d'autres cas : portage d'applicatif sans recompilation, plate-forme de développement, etc.
- La virtualisation de machines, avec *VMWare*, *Microsoft Virtual PC* et *Virtual Server*, *QEMU* avec module d'accélération. La couche logicielle du VMM s'intercale au-dessus du matériel (ou partiellement du noyau de l'hôte) en exportant une abstraction de celui-ci. Au contraire des émulateurs, le VMM présente une machine virtuelle dont le processeur est obligatoirement du même type que celui de l'hôte, et les périphériques virtuels existent dans la réalité. Ainsi, le VMM peut laisser s'exécuter en grande partie nativement le code des machines virtuelles, et les pilotes utilisés par le noyau de l'invité sont standards. Les performances obtenues sont alors bonnes. Le système d'exploitation utilisé sur le serveur virtuel est non modifié. Il n'a pas conscience du processus de virtualisation, ce qui impose des contraintes fortes sur le VMM, surtout dans le cadre du processeur x86 qui se prête mal à la virtualisation (cf. section précédente).
- La « para-virtualisation », variante de la solution précédente, à laquelle *Xen* appartient. Sous Xen (version 2), un hyperviseur s'exécute en niveau 0. Le noyau de l'invité s'exécute en niveau 1, et les applications sont en niveau 3. L'architecture de la machine virtuelle présentée est similaire sans être identique au système hôte. Le noyau de l'invité doit être porté dessus, notamment pour utiliser des pilotes de périphériques spéciaux, qui permettent d'optimiser les transferts, ainsi que pour faire appel à l'hyperviseur à travers des « hypercall », permettant d'exécuter les instructions privilégiées du niveau 0. En revanche, il n'y a pas de changement de l'interface du noyau de l'invité, donc les applications n'ont pas besoin d'être modifiées. Les performances sont très bonnes.
- *User Mode Linux* (UML) ressemble à Xen, dans le sens où le système d'exploitation invité est adapté à son nouvel environnement. Il s'exécute en niveau de privilège 3, comme un processus utilisateur, et il repose sur le noyau Linux (non modifié) de l'hôte pour accéder à la couche matérielle. Les performances obtenues sont moyennes.

3.3 Virtualisation au niveau du système d'exploitation

Sur un système d'exploitation multi-tâches, plusieurs processus cohabitent et le système leur alloue, de façon répartie, les ressources matérielles. Les processus peuvent aussi inter-agir entre eux (communications inter-processus, envoi de signaux, etc.).

Il est possible d'agir à ce niveau pour obtenir un ersatz de machine virtuelle, en isolant les processus dans des espaces séparés. Les interactions entre processus sont cantonnées à chaque espace, et divers autres mécanismes (isolation au niveau du système de fichiers, du réseau, réservation et affichage des ressources, etc.) permettent d'obtenir la sensation, pour les processus d'un espace, d'appartenir à une machine séparée. *Linux Vserver* [4], *BSD Jails* ou *Solaris Zones* sont des solutions de ce type.

Un seul noyau s'exécute, celui de l'hôte. Les processus utilisent l'espace disque de l'hôte et il n'y a pas de démon particulier sur ce dernier. Le noyau continue à assurer, comme sur un serveur traditionnel, la coexistence des processus et la sécurité du système. Les performances sont donc identiques à celles d'un vrai serveur.

4 Architecture de Linux VServer

Le projet Linux VServer implémente une solution de ce type, basée sur Linux[3]. Les unités distinctes regroupant des processus sont appelées des « serveurs privés virtuels » (« vserver » par la suite).

Linux VServer est constitué de commandes utilisateurs, et surtout d'un patch sur le noyau Linux. La version 2.0 applicable au noyau Linux 2.6.12.4 a été déclarée stable en août 2005. Cette version et ses fonctionnalités servent de démonstration à travers ce document.

La documentation présente sur le site internet du projet Linux Vserver [4] est variée, mais il faut de nombreuses lectures pour arriver à synthétiser les diverses fonctionnalités et les mécanismes sous-jacents, et c'est ce que nous nous proposons de faire dans cette section.

4.1 Séparation des processus

Le premier élément ajouté par le patch noyau est la notion de **contexte**, dont le but est d'établir une séparation entre processus :

- un contexte est identifié par un entier, le *xid* ;

[3] Linux VServer n'est donc pas restreint au processeur x86.

- les processus sont associés à un contexte, et un processus fils hérite du contexte de son créateur ;
- règle générale : les processus d'un contexte ne peuvent voir que les processus du même contexte ;
- le processus de PID 1 (*init*) est une exception[4] : il est visible depuis tous les contextes, mais seul le super-utilisateur (*root*) du contexte 0 peut interagir avec lui ;
- à l'amorçage de l'hôte, les processus sont dans le contexte 0 (contexte par défaut). Même les processus de ce contexte ne peuvent pas voir les processus d'autres contextes (moins de risques d'erreur lors de l'administration de l'hôte) ;
- afin de superviser les processus, un contexte spécial, appelé **spectateur** (xid 1), permet de voir les processus de tous les contextes, mais pas d'interagir avec (envoi de signaux avec la commande *kill* par exemple) ;
- seul root du contexte 0 peut créer un contexte ou lancer un processus dans un contexte existant.

Dans la suite, nous emploierons parfois par abus le mot vserver à la place de contexte, qui serait techniquement plus juste, car une bijection est souvent établie entre un vserver et un numéro contexte : une règle de bon usage veut qu'un vserver possède un xid statique[5] (i.e. constant).

4.2 Séparation réseau

Une isolation réseau est aussi nécessaire. L'hôte dispose en général de plusieurs adresses IP et les processus d'un vserver sont limités à une interface réseau, à travers un nouvel appel système ajouté par le patch Linux VServer. Un processus fils hérite de la restriction de son créateur.

L'interface de boucle locale (*loopback*) n'est pas accessible dans un vserver, pour assurer la sécurité de l'hôte. Cette limitation devrait être résolue avec la nouvelle approche choisie dans les versions en cours de développement.

4.3 Vue du système de fichiers

Un vserver est restreint à une partie du système de fichiers de l'hôte, à travers l'appel système classique *chroot*. Ainsi, les processus d'un vserver ne voient qu'une sous-partie de l'arborescence de fichiers de l'hôte.

En outre, chaque vserver est dans un espace de nom différent : cette notion du système de fichiers virtuel de Linux permet d'avoir une vue différente du système de fichiers (montages).

La « cage chroot » présentant des possibilités d'échappement, Linux VServer a établi une extension aux attributs de fichiers pour parer à cette éventualité[6].

Une conséquence directe est que chaque vserver devrait avoir sa propre arborescence, vue comme la racine du système de fichiers. Ceci peut entraîner des conséquences négatives :

- consommation élevée d'espace disque ;
- difficulté de maintenir à jour un nombre élevé de vservers ;
- consommation accrue de mémoire : les librairies partagées ne possédant pas le même inode, elles seront chargées plusieurs fois en mémoire, même si les vservers utilisent la même version. L'aspect coopératif des processus UNIX est ainsi perdu en partie.

Une réponse existe, **l'unification**, basée sur l'utilisation de liens en dur (*hard links*) entre les différents vservers. L'utilisation en plus de certains attributs déjà existants ou nouvellement créés[7], offre la possibilité à des vservers initialement unifiés de faire des mises à jour de façon indépendante par la suite.

4.4 Capacités du super-utilisateur

Une restriction des pouvoirs du super-utilisateur est définie pour chaque vserver, à travers le système de capacités natif de Linux (*capabilities*). Une capacité est un jeton utilisé par un processus pour prouver qu'il est autorisé à faire une opération sur un objet. Le système de capacités Linux, implémenté depuis le noyau 2.2, est basé sur les capacités POSIX, conçues pour diviser les privilèges du super-utilisateur en un ensemble de privilèges distincts.

Il s'avère parfois grossier[8] et Linux VServer a défini quelques capacités additionnelles pour accorder plus finement des droits aux administrateurs des vservers.

4.5 Communication inter-processus

Les ressources « System V IPC » sont propres à chaque vserver. Il s'agit de mécanismes de communications entre processus, faisant partie de la norme POSIX : files de messages, sémaphores et mémoire partagée. Classiquement, les droits d'accès à des objets IPC sont vérifiés comme pour un fichier. Avec Linux VServer, le contexte est utilisé comme une clé supplémentaire.

4.6 Sécurité et extensions de /proc

Le système de fichiers *proc*, donnant accès dans l'espace utilisateur à des données du noyau, est nécessaire pour un certain nombre d'outils utilisateurs. Il devra donc apparaître, mais de façon limitée, dans un vserver. Pour ne pas affaiblir la sécurité, des attributs ont été ajoutés pour cacher certaines entrées[9] dans les vservers.

Une extension de ce système de fichiers a aussi été apportée, pour rendre disponible des informations sur les vservers.

---

[4] sauf si l'option *fakeinit* est activée.

[5] les numéros 2 à 49152 sont réservés aux contextes statiques, 49153 à 65535 aux contextes dynamiques.

[6] nouvel attribut *barrier*.

[7] attribut existant *immutable* et nouvel attribut *iunlink*.

[8] CAP_SYS_ADMIN, CAP_NET_RAW par exemple.

[9] attributs *hidden*, *admin* et *watch*.

4.7 Limitation de ressources, ordonnanceur de tâches

Une limitation des ressources système est possible. Linux VServer a étendu les notions déjà implémentées dans le noyau Linux afin de fixer des limites aux processus d'un vserver (temps CPU, taille de la pile, d'un fichier core, etc.) ou à l'ensemble du vserver (nombre de processus, mémoire résidente et disponible, nombre de fichiers ouverts).

La consommation CPU des processus d'un vserver peut être contrôlée, afin qu'un vserver trop « gourmand » ne prenne pas toutes les ressources de l'hôte, par un algorithme « seau de jetons » (*token bucket*)[10].

4.8 Virtualisation des informations systèmes

Certaines informations affichées par le système d'exploitation peuvent être virtualisées. Chaque vserver a ainsi une vue différente du nom d'hôte, de l'*uptime*, du type de machine, de la quantité de mémoire disponible, etc.

4.9 Appartenance des fichiers à un vserver

Une extension des systèmes de fichiers a été apportée, pour pouvoir éventuellement associer un contexte aux fichiers. Le contrôle d'accès se base ensuite sur cette information. Elle peut aussi être utilisée pour imposer des limites d'espace disque ou des quotas par contexte[11].

5 Architecture de VMware ESX

5.1 Composants de l'architecture

Les composants principaux [5] de l'hyperviseur VMware ESX (ou ESX par la suite) sont présentés dans la Figure 3.

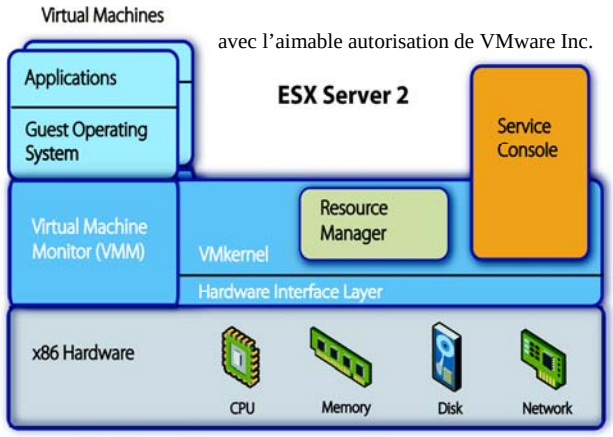

Figure 3 - *Architecture de VMware ESX*

[10]algorithme souvent utilisé en réseau. Ici, les processus d'un vserver se partagent les jetons d'un seau pour pouvoir s'exécuter. Le seau se remplit de N jetons tous les T intervalles de temps. S'il est « gourmand », le vserver aura en moyenne une part N/T de la CPU.

[11]si un système de fichiers est utilisé par vserver, cette extension n'est évidement pas nécessaire.

Le processeur, la mémoire, le disque et le réseau du système physique sont alloués dynamiquement aux machines virtuelles (VM) via le gestionnaire de ressources. Les pilotes des périphériques matériels ainsi que le système de fichiers VMFS (voir section 5.3) sont des éléments de la couche d'interface matérielle. Ces deux briques constituent le VMkernel.

La virtualisation du processeur incombe au moniteur de machine virtuelle : son rôle est de traiter le code fourni par la VM (exécution directe ou traduction). Il établit également la correspondance entre la mémoire physique de la VM et celle du système hôte et à ce titre prend en charge les opérations liées aux accès mémoire. Il se charge aussi de l'interception des requêtes d'entrées/sorties en provenance des machines virtuelles et les soumet au VMkernel. Une fois qu'elles sont traitées, il notifie la machine virtuelle.

La console de service est responsable de l'amorçage du système puis elle démarre la couche de virtualisation (VMkernel + VMM) à qui elle rend le contrôle. Elle est rechargée par le VMkernel en tant que VM privilégiée pour exécuter des applications de configuration, de gestion et de support ainsi que l'interface entre l'utilisateur et ESX. Elle prend en charge les périphériques dont la performance n'est pas critique, tels que clavier, souris, écran, lecteur de disquettes et de CD-ROM.

Le dernier élément de l'architecture et non le moindre est la machine virtuelle. Elle est constituée d'une carte mère avec un jeu de puces (*chipset*) Intel 440BX, du ou des processeurs[12] de la machine physique, de 3,6 Go de RAM au maximum, d'au plus 4 cartes SCSI, 4 cartes réseau[13], 2 lecteurs de disquettes et 2 lecteurs de CD-ROM. Elle possède également un BIOS (Phoenix) et sait exécuter plusieurs systèmes d'exploitation invités parmi Linux, Windows, Netware et FreeBSD.

5.2 Allocation des ressources aux VM

Maintenant que nous avons vu les composants essentiels de l'architecture de VMware ESX, nous allons nous intéresser à la façon dont les ressources principales (processeur, mémoire, disque, réseau) du serveur physique sont attribuées aux machines virtuelles [6].

Le principe général consiste à garantir aux VM une quantité de ressources fixée puis à ajuster dynamiquement ces dernières en fonction de la demande. En cas de contention, un mécanisme additionnel de partage rentre en action. Par défaut, les VM sont traitées équitablement.

Voyons au cas par cas comment cela se passe :

- le processeur est réparti entre les différentes VM en tenant compte de trois paramètres : le pourcentage minimum souhaité, l'utilisation maximale envisagée ainsi que le nombre d'unités de partage (*shares*) alloué. Le premier réglage garantit qu'un pourcentage minimum du processeur sera réservé à la VM. La contre-partie est que ce temps processeur est perdu pour les autres

[12]dans la VM le nombre de processeurs est limité à 2.

[13]le nombre total de cartes ne peut excéder 5.

machines. Le pourcentage maximum définit la limite du temps processeur qui sera attribué à la VM. L'ordonnanceur affecte au mieux le processeur[14] en respectant les seuils minimaux et maximaux fixés pour chaque machine virtuelle. En cas d'épuisement de cette ressource, les unités de partage sont appliquées. En fait, elles définissent le poids relatif de chaque VM. Ainsi une machine qui possède deux fois plus d'unités de partage qu'une autre se verra attribuer le processeur deux fois plus souvent en cas de compétition. Pour les VM biprocesseurs, intervient une contrainte supplémentaire : deux processeurs de la machine physique sont affectés simultanément pour conserver le fonctionnement SMP original.

- la mémoire est allouée à l'aide des trois mêmes réglages que le processeur : quantité minimale souhaitée, limite maximale et nombre d'unités de partage. Le nombre d'unités de partage varie en fonction d'une taxe sur la mémoire inutilisée. Plus la quantité de mémoire inactive est importante dans une VM, plus son nombre d'unités diminue. ESX peut forcer le système d'exploitation invité à déplacer des pages mémoire vers le disque. Il réalise cela grâce à un mécanisme dit de « ballooning » consistant à gonfler et dégonfler un ballon qui représente la pression exercée sur une VM pour qu'elle libère de la mémoire. Si ce mécanisme n'est pas activé, ESX utilise un fichier d'échange dans lequel il place arbitrairement les pages les moins accédées par le système invité. L'utilisation de la mémoire est par ailleurs optimisée grâce à la mutualisation des pages mémoires communes à différentes VM.
- le disque ne possède pas de réglages minimum et maximum. Seul le système d'unités de partage est actif. La consommation de chaque VM est déterminée par le nombre d'opérations SCSI qu'elle réalise ainsi que par la quantité de données transférée.
- le réseau s'appuie sur un module de mise en forme de trafic (*traffic shaping*) qui ne prend en compte que le trafic sortant. Il est possible de définir la bande passante moyenne et celle atteinte lors des pics. Il faut également spécifier la taille des données lors d'une rafale qui lorsqu'elle est atteinte indique au module de ramener la bande passante de la VM à sa valeur moyenne.

5.3 VMFS

Il s'agit du système de fichiers propriétaire de VMware sur lequel résident les fichiers représentant les disques virtuels des VM. Il a été conçu dans le but d'offrir un niveau de performance élevé, la possibilité de stocker de très gros fichiers et l'accès concurrent depuis plusieurs machines[15] (dans le cas d'un SAN). La fonctionnalité de migration d'une machine virtuelle d'un hôte à un autre ou la mise en place d'une grappe de serveurs virtuels requièrent cette fonctionnalité.

[14]il est également possible de jouer sur l'affinité du processeur pour permettre à une VM de tourner sur un processeur physique particulier.

[15]les verrous sont posés au niveau fichier.

5.4 Les réseaux des mondes virtuel et réel

Nous allons découvrir rapidement dans cette section les possibilités qui nous sont offertes pour la configuration des réseaux physiques et virtuels dans ESX.

Au niveau physique, il est recommandé de dédier une carte réseau à la console de service. Les autres cartes peuvent être groupées afin d'assurer la redondance et éventuellement l'équilibrage de charge (en fonction de l'adresse Ethernet de la VM ou de l'adresse IP de destination[16]).

Au niveau virtuel, chaque VM peut disposer de plusieurs cartes réseau connectées à des commutateurs virtuels de deux types :

- commutateurs privés : ils ne sont pas reliés au monde réel. Les VM qui y sont connectées peuvent communiquer entre elles comme si elles étaient branchées sur un commutateur réel sans toutefois que le réseau physique ne soit impacté ;
- commutateurs publics : ils possèdent un lien vers un groupe de cartes physiques et vers les machines virtuelles.

Ces commutateurs peuvent ou non posséder des VLAN : dans ce dernier cas, des groupes de ports sont définis – chacun d'entre eux représentant un VLAN – et le marquage de trames 802.1q est activé.

Les combinaisons entre les mondes réel et virtuel sont multiples et couvrent la majorité des besoins.

6 Mise en œuvre de Linux VServer

6.1 Pré-requis

L'élément le plus important pour bien mettre en œuvre des vservers est avant tout de posséder une bonne connaissance de Linux et des distributions à transformer en vserver.

Il faut aussi s'assurer de disposer de suffisamment de ressources (par exemple espace disque, une distribution Linux « légère » prenant 200 à 400 Mo) sur l'hôte.

Enfin, il faut garder à l'esprit les limites de ce que peut offrir un vserver (pas d'interface de boucle locale, droits super-utilisateur restreints, etc.).

6.2 Installation et configuration du serveur hôte

La première étape consiste à installer le système hôte, en incluant un nombre minimal de logiciels. N'oubliez pas que la sécurité de tous les vservers sera conditionnée à celle de l'hôte. Vous devrez donc veiller à ne démarrer que les services **strictement** nécessaires, par exemple *syslogd*, *klogd*, *atd*, *cron*, *ntpd*, et peut-être *sshd* pour un accès distant.

[16]nécessite la configuration du matériel actif avec LACP (802.3ad) pour gérer la duplication d'adresses Ethernet.

La deuxième étape réside en la compilation d'un noyau Linux patché avec Linux VServer. La version 2 s'appuie sur le noyau Linux 2.6. Un patch de Linux VServer est dédié à une version particulière du noyau. Ainsi, nous avons utilisé le patch nommé *patch-2.6.12.4-vs2.0.diff*, qui apporte la version 2.0 sur le noyau 2.6.12.4. Les sites de téléchargement sont :

- http://www.kernel.org pour le noyau ;
- http://www.13thfloor.at/vserver/s_rel26/v2.0/ pour le patch VServer.

Le patch ajoute principalement un menu « Linux VServer » dans les options du noyau (voir Figure 4).

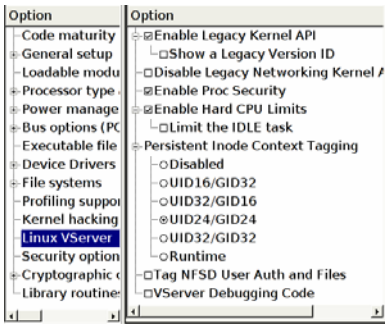

Figure 4 - *Section Linux VServer du noyau*

La troisième étape consiste à compiler les commandes utilisateurs (*util-vserver*), disponibles sur le même site que le patch. Ces commandes permettent de manipuler les nouvelles notions introduites dans le noyau (contexte, attributs des fichiers, etc.), de gérer les vservers (démarrer, éteindre, créer un vserver), de lancer certains scripts à l'amorçage de l'hôte, etc. En pré-requis, il faut installer certaines librairies ou commandes (*vlan*, *iproute*, *e2fslibs-dev*, *dietlibc-dev*, *beecrypt2-dev*). Afin de respecter une hiérarchie de fichiers plus classique, nous avons fait la compilation avec :

```
./configure --sysconfdir=/etc \
  –-localstatedir=/var
make && make install
make install-distribution
```

Quelques étapes additionnelles doivent être réalisées « à la main » pour achever l'installation :

- création de liens pour démarrer automatiquement les services *vprocunhide*, *rebootmgr* et *vservers-default* ;
- mise en place de l'attribut *barrier* pour protéger le chroot :
  `setattr --barrier /vservers`
- éventuellement, création du répertoire /var/lock/subsys (s'il n'existe pas).

Pour achever l'installation de l'hôte, il faut veiller à ce qu'aucun démon ne fasse de bind sur l'interface « 0.0.0.0 » (qui désigne toutes les interfaces réseaux). Il faut en effet les restreindre à n'écouter que sur l'interface dédiée à l'hôte afin de ne pas interférer avec les vservers (par exemple, paramètre *ListenAddress* de sshd). Des *wrappers* fournis avec util-vserver peuvent aussi être utilisés pour atteindre ce but, le lancement du démon étant restreint par l'utilisation de la commande *chbind*.

L'hôte est maintenant prêt. Avant de vous lancer dans l'installation d'un vserver, vous pouvez jouer avec les différentes commandes de util-vserver pour vérifier leur fonctionnement et vous habituer à leur usage. Beaucoup d'exemples sont donnés dans [7].

6.3 Administration : déploiement d'invités, migration de machines réelles

La création d'un vserver, que ce soit ex-nihilo ou par recopie d'un serveur réel, est une tâche à peu près identique. Il faut à la fois créer l'arborescence de configuration du vserver [8], /etc/vservers/<vserver>/, et l'arborescence contenant l'environnement pour le chroot, /vservers/<vserver>.

Cette tâche est facilitée par la commande *vserver*, qui est essentielle pour gérer les vservers. Elle offre, entre autres, une option *build* pour créer un vserver. L'arborescence du vserver sera soit un squelette, soit peuplée avec une arborescence a minima. Exemples :

- ```
  vserver foo build \
    --context 103 \
    --interface foo=eth0:10.1.1.1 \
    --flags virt_uptime \
    -m skeleton
  ```
  construit un vserver nommé « foo », de numéro de contexte (statique) 103 et qui aura une interface réseau d'adresse IP 10.1.1.1 (alias « foo » sur eth0). Un drapeau (*flag*) est ajouté sur le vserver (*virt_uptime*) pour virtualiser l'affichage de l'uptime.
  L'arborescence chroot devra être peuplée par un autre moyen. Le squelette ne contient que le répertoire /dev, avec les quelques fichiers périphériques nécessaires à un vserver.
- Si le serveur hôte a la chance d'être basé sur Debian, et que « `-m skeleton` » devient « `-m debootstrap -- -d sarge` », l'arborescence chroot sera déjà peuplée avec une distribution Debian.

Pour virtualiser un serveur existant, vous pouvez faire un squelette, puis transférer une archive tar de son système de fichiers. L'archive devra exclure les répertoires inutiles sur un vserver, à savoir par exemple /proc, /dev, /sys, /mnt .

Quelle que soit la méthode choisie, voilà les points à ne pas oublier :

- le répertoire /dev d'un vserver ne contient (et c'est normal) que quelques fichiers périphériques, pour ne pas donner à root d'un vserver l'accès à des composants matériels ou au noyau[17] ;
- le fichier /etc/fstab du vserver n'est pas utilisé ;
- il faut arrêter une grande partie des services, qui deviennent inutiles sur un vserver, par exemple ceux ayant trait à la gestion du matériel (chargement de

[17] le super-utilisateur du vserver ne pourra pas créer de nouveaux fichiers périphériques car il est restreint par une capacité.
  ```

modules, usb) puisque cette partie relève du noyau, donc de l'hôte ;

- par défaut, un vserver est démarré avec la commande « /etc/init.d/rc 3 », ce qui signifie que le runlevel sera 3 (et non pas celui indiqué par défaut dans inittab) ;
- il faut parfois modifier le script /etc/init.d/rc : par exemple sous Mandrake, il fait une redirection vers la console, ce qui empêche l'affichage des démarrages ou des arrêts de services sur le terminal ;
- de même, le script final d'arrêt doit parfois être modifié : certaines distributions utilisent un seul script (*halt*) pour faire toutes les opérations finales. Or certaines n'ont pas de sens dans un vserver : démontage des systèmes de fichiers, de la swap, arrêt des interfaces réseau, etc.
- certaines distributions incluent des cron quotidiens s'exécutant à heure fixe (/etc/cron.daily/). Si tous les vservers lancent leur tâche en même temps, il y a un risque de surcharge de l'hôte, vous pouvez donc modifier l'horaire de lancement.

Ces opérations nécessitent donc une bonne connaissance de Linux, et un petit investissement lorsque vous commencez à construire votre premier vserver. Mais une fois celui-ci pleinement opérationnel, vous pouvez simplement en faire une archive tar, à désarchiver pour votre prochain vserver. Certains sites web proposent des archives déjà construites pour différentes distributions.

Vous pouvez aussi simplement utiliser « vserver-copy » pour faire une copie d'un vserver.

6.4 Supervision

Peu d'outils spécifiques existent pour mesurer l'activité des vservers. En voici quelques uns :

- *vps* et *vtop*, qui permettent au serveur hôte d'avoir une vue de tous les processus ;
- *vserver-stat*, qui affiche le nombre de processus, la mémoire et le temps CPU utilisés par chaque vserver ;
- le fichier /proc/virtual/<xid>/limit, montrant les limites systèmes fixées au vserver, les valeurs minimum et maximum atteintes, et le nombre de fois où la limite a été atteinte.

La virtualisation de l'affichage de la charge CPU sur un vserver[18] permet par exemple par la suite d'utiliser des outils standard comme *mrtg* pour tracer la charge sur différents vservers. Mais la dépendance entre vservers n'est pas complètement gommée, et le résultat ne pourra pas être parfaitement exact.

6.5 Disponibilité, récupération en cas de sinistre

Un vserver ne contient pas de noyau, il est simplement constitué de fichiers de configuration et d'une arborescence chroot. En cas de sinistre, il suffit donc de restaurer ces deux éléments sur un nouveau serveur. Peu importe le matériel sous-jacent, puisqu'il est pris en charge par le noyau de l'hôte.

Il n'y a pas d'outils spécifiques offerts par Linux VServer pour garantir une haute disponibilité. A cette fin, il faut utiliser des outils tiers (*rsync* et *heartbeat* par exemple).

7 Mise en œuvre de VMware ESX

7.1 Pré-requis

Toute démarche de virtualisation avec VMware ESX doit commencer par la consultation des guides de compatibilité – disponibles chez l'éditeur [9] – afin de vérifier que le matériel (carte réseau, contrôleur disque et RAID, etc.) a bien été certifié. Outre le fait que le matériel soit supporté, certains pilotes sont optimisés pour le VMkernel afin de garantir une meilleure gestion des ressources et des performances accrues. Si le serveur est attaché à un SAN, il faut également en vérifier la validation.

Il vous reste à décider quels produits complémentaires à ESX, vous allez mettre en place parmi ceux-ci[19] :

- Virtual SMP, module complémentaire permettant d'exploiter plusieurs processeurs dans une VM ;
- VirtualCenter, outil d'administration et de supervision de l'ensemble des serveurs ESX. Il facilite également le déploiement des VM ;
- VMotion, associé à VirtualCenter, il implémente la migration des VM d'un serveur physique à un autre.

Vous devez ensuite adapter la quantité et le dimensionnement de vos serveurs physiques au nombre et type de VM que vous allez exploiter et aux fonctionnalités attendues (p. ex. redondance).

7.2 Installation et configuration du serveur hôte

Il est possible de réaliser une installation automatisée du serveur ESX grâce à un fichier *kickstart* préalablement généré et qui sera utilisé par *anaconda*.

Cependant, nous allons nous cantonner dans cet article aux étapes d'une installation manuelle et personnalisée d'un serveur[20]. Celle-ci ressemble à celle d'une Red Hat à laquelle ont été rajoutés des écrans spécifiques à ESX.

Une fois le CD inséré et la machine démarrée, une invite *syslinux* demandant le type d'installation – texte, graphique, sur un SAN – apparaît.

Le clavier et la souris configurés il faut saisir les numéros de licence du produit puis définir la configuration (mémoire, contrôleurs et périphériques accessibles) de la console de service.

---

[18] flag *virt_load.*

[19] Cette liste de produits développés par VMware ne comporte pas volontairement P2V dont le fonctionnement n'a pas été très convaincant !

[20] l'installation diffère un peu si elle a lieu sur un serveur lame ou qu'elle est destinée à être amorcée depuis un SAN.

Le disque est ensuite divisé en trois partitions au minimum[21] : /boot, la partition d'échange (*swap*) et / ; puis il est formaté.

S'ensuit le réglage des paramètres de la carte réseau de la console de service (première carte réseau détectée).

Les étapes qui se déroulent après sont classiques : réglage de la zone de temps, du mot de passe du super-utilisateur et création optionnelle d'un compte.

La procédure graphique s'achève par l'installation des paquetages (dont ceux propres à ESX[22]).

La configuration se poursuit après le redémarrage du serveur à travers une interface web qui permet de :

- rentrer des numéros de série additionnels ;
- changer le profil de démarrage : mémoire et périphériques alloués à la console. Ces derniers peuvent être strictement réservés à la console, aux machines virtuelles ou partagés ;
- modifier la configuration du stockage afin de créer par exemple une partition VMFS ;
- éditer la configuration du réseau en créant les commutateurs virtuels ;
- définir la taille du fichier d'échange de ESX. Il s'agit du fichier utilisé lorsque le ballooning n'est pas en action et que ESX réclame de la mémoire ;
- ajuster les paramètres de sécurité pour les échanges avec le serveur (protocoles utilisés, chiffrement).

Une fois ces actions achevées, les machines virtuelles peuvent commencer à être déployées.

7.3 Administration : déploiement d'invités, migration de machines réelles

Les tâches d'administration présentées dans cette section ne concernent pas directement le serveur ESX mais plutôt ce qu'il héberge : les machines virtuelles. La plupart des opérations peuvent être réalisées en ligne de commande via la console de service, par l'interface web fournie avec le produit ou en utilisant un produit additionnel comme VirtualCenter. Chaque méthode possède ses particularités et s'avère plus ou moins adaptée en fonction de l'opération à effectuer. Il faut cependant reconnaître que VirtualCenter apporte une souplesse et une simplicité indéniables pour des environnements ESX multi-serveurs.

L'installation d'une machine virtuelle sur un serveur ESX se passe comme dans le monde réel – choix dans le BIOS ou dans le menu de démarrage, du périphérique pour l'amorçage (CD-ROM, réseau, etc.) puis procédure classique – à ceci près qu'il faut commencer par sélectionner le serveur hôte de la VM puis définir la configuration de cette dernière (via l'interface web) : connexion au réseau virtuel, taille du(es) disque(s) et mode d'utilisation[23], limitation des ressources processeur, mémoire, disque et réseau (cf. section 5.2). Certaines options (nombre de processeurs, taille des disques) bien que modifiables dans ESX ne le sont pas toujours facilement au niveau des systèmes d'exploitation. Une fois la configuration achevée, il suffit d'allumer la VM et de commencer l'installation du système invité. Celle-ci terminée, il reste à ajouter les *vmware-tools* qui contiennent des pilotes spécifiques ou mis à jour par VMware ainsi que des outils nécessaires pour améliorer la gestion des VM (gestion de la mémoire par ballooning, synchronisation du temps, etc.). Il faut aussi penser à synchroniser l'horloge de l'invité avec celle de son hôte car contrairement à ce qui se passe sur un serveur physique l'horloge des VM dérive naturellement, parfois même avec les outils classiques de synchronisation du temps. En effet, les interruptions liées à l'horloge ne sont généralement pas traitées dès qu'elles arrivent car la VM ne dispose souvent pas du processeur réel à ce moment-là.

Le déploiement de nouvelles machines virtuelles peut être grandement simplifié en tirant parti des spécificités amenées par la virtualisation, notamment le fait que le matériel présenté à la VM est, d'une manière générale, toujours identique. Ainsi la création de quelques images (par des outils tiers ou natifs) assurera un gain de temps appréciable. Le déploiement peut selon le cas être couplé avec des fichiers de réponses automatisés (exploitables entre autres par *sysprep*). Il est préférable que ces modèles (« golden masters » dans le jargon de VMware) incorporent, comme dans le monde réel tout ce qui sera commun aux VM, comme, par exemple, les dernières mises à jour de sécurité, l'anti-virus, l'agent de sauvegarde et les vmware-tools. Enfin, les machines virtuelles peuvent exploiter directement des images de CD-ROM ou de disquettes stockées sur le serveur.

Le choix du serveur hôte, dans une architecture avec plusieurs serveurs ESX, est crucial et pas toujours facile à réaliser car il faut prendre en compte de nombreux paramètres liés aux ressources consommées en moyenne et en crête : une VM mal localisée risque d'avoir un impact sur les performances de l'ensemble des systèmes hébergés. Par contre, la mémoire du serveur physique peut être économisée en groupant sur le même serveur ESX les machines qui exécutent un système invité identique (beaucoup de pages mémoires étant communes).

La migration d'une machine physique du monde réel au monde virtuel (sans réinstallation) peut nécessiter le recours à des outils tiers et c'est d'ailleurs le cas pour les machines Windows. Pour Linux, en revanche, la recopie de l'arborescence du serveur source, la recompilation de son noyau et l'ajout des vmware-tools s'avèrent en général suffisants.

[21] une partition supplémentaire de 100 Mo est créée pour permettre au VMkernel d'écrire un vidage de la mémoire (*core dump*) et un fichier journal en cas de plantage.

[22] VMware-esx, VMware-perftools, VMnix, VMware-mui.

[23] il existe 4 modes : persistant (comportement identique à un disque normal), non persistant (données perdues à l'arrêt de la VM), réversible (modifications enregistrées dans un fichier séparé et possibilité de choisir de les appliquer, rejeter ou conserver à l'arrêt de la VM), ajout (identique au mode réversible hormis que l'action à effectuer doit être faite manuellement, aucun choix n'étant proposé à l'arrêt de la VM).

Un administrateur de machine virtuelle doit posséder au moins les mêmes prérogatives que dans le monde réel[24] : allumage et arrêt de la machine, modification de sa configuration (matérielle), accès à la console, connexion au réseau, etc. Tout ceci doit pouvoir être réalisé en toute sécurité et être dévolu à une personne ou un groupe d'individus. Pour répondre à cette problématique ESX définit des utilisateurs et des groupes qui ont des droits (rôles) particuliers sur le serveur ESX et/ou les VM [6]. En réalité, ce sont des permissions qui sont appliquées sur des fichiers, ce qui n'est pas très souple. VirtualCenter amène cette flexibilité en définissant ses propres rôles – lecture seule, utilisateur de VM, administrateur de VM et administrateur VirtualCenter – pour les individus et les groupes de sa base de comptes (indépendante de ESX).

7.4 Supervision

La supervision parfois négligée pour nos serveurs traditionnels est vitale pour les serveurs ESX car, hébergeant de nombreux systèmes leur bonne santé est critique. VMware met à la disposition des administrateurs quelques outils avantageux, dédiés à la gestion des ressources, point crucial dans ce type d'environnement :

- *esxtop* qui s'exécute en mode texte et qui ne dépaysera les habitués de la commande *top* sur système Unix. Cependant outre les classiques informations d'uptime, de consommation mémoire et de processus en cours d'exécution, cette version adaptée à ESX (elle prend ses renseignements auprès du VMkernel) fournit d'autres données précieuses dont les entrées-sorties au niveau disque et réseau, le pourcentage d'utilisation des processeurs logiques (hyperthreading) et les statistiques propres aux VM (processeur d'exécution, pourcentage d'utilisation du processeur, pourcentage de temps où le processeur était indisponible, pourcentage de consommation des ressources allouées, etc.) ;
- *vmkusage* qui collecte les données de performance pour générer à la volée des pages web de statistiques sur différentes durées. Les tableaux et graphiques donnent une vue macroscopique (par serveur ESX) ou microscopique (par VM) ;
- VirtualCenter qui permet d'avoir une vision synthétique des ressources de l'ensemble des serveurs et de raffiner la vue en fonction du type d'investigation menée. À cet effet, les données sont disponibles en instantané ou dans des graphes (journaliers, hebdomadaires, mensuels, annuels, personnalisables) et elles peuvent être groupées et consolidées. En dehors de cet aspect de surveillance, l'outil permet la programmation de seuils associés à des alarmes (alertes SNMP, messages électroniques, etc.) et des actions (suspension ou arrêt de VM par exemple). Un exemple d'écran est présenté dans la Figure 5.

[24] d'autres possibilités existent pour les machines virtuelles : suspension, migration sur un autre serveur, clonage, déploiement de modèles, etc.

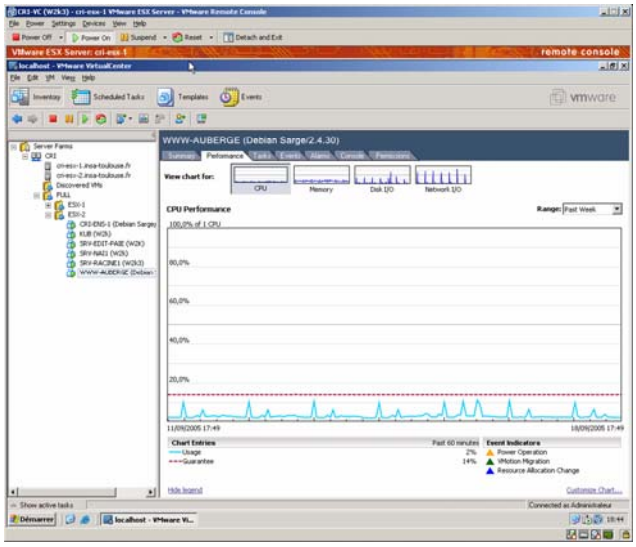

Figure 5 - *Consommation CPU d'une VM sur une semaine*

Ces outils permettent de prévenir toute contention de ressource et d'optimiser la charge des serveurs ESX en fonction des VM exécutées et des installations à venir.

7.5 Disponibilité, récupération en cas de sinistre

VirtualCenter inclut (optionnellement) une fonctionnalité à laquelle nous avons déjà fait allusion : Vmotion. Elle permet de migrer une machine virtuelle en fonctionnement d'un serveur ESX à un autre. Elle requiert cependant que les machines virtuelles soient hébergées sur des serveurs ESX possédant une ressource disque partagée (dans la majorité des cas un SAN) et une configuration identique (modèle et nombre de processeurs, commutateurs virtuels, etc.). Les différentes étapes nécessaires à la migration sont : la vérification de la disponibilité des ressources pour démarrer la VM sur la cible, la copie de l'état de la VM (mémoire, NVRAM[25], configuration) sur la destination, la suspension de la VM sur la machine source, le transfert du delta restant au niveau de la mémoire et de la NVRAM vers le serveur de destination, l'enregistrement de la VM auprès de ce dernier, puis la réactivation de la VM sur le deuxième serveur. La phase critique (suspension, copie du delta et réactivation) ne prend que quelques secondes et elle est transparente pour les applications et les utilisateurs. Cette fonctionnalité permet ainsi de procéder à la maintenance ou de rééquilibrer la charge d'un serveur ESX sans interruption de service d'autant plus que les migrations peuvent être planifiées.

En ce qui concerne la sauvegarde, il est possible de combiner deux stratégies : sauvegarde des VM comme des serveurs physiques et sauvegarde des fichiers du serveur ESX représentant les VM[26]. Les fichiers contenant les disques des VM sont, sous condition, exploitables sous d'autres produits de VMware et évidemment sur un autre

[25] elle contient le BIOS.

[26] il faut utiliser des scripts (tirant parti par exemple du mode réversible) ou des outils tiers pour assurer l'intégrité de la machine virtuelle.

serveur ESX. Pour restaurer une VM, il suffit donc de disposer de sa configuration et de son fichier NVRAM.

Quant aux solutions de disponibilité, elles nécessitent de mettre en place des architectures de type grappe (*cluster*) dont les différents nœuds sont virtuels et/ou mixtes.

8 Comparaison VMware – Linux VServer

Autant l'avouer, une comparaison entre VMware et Linux VServer n'a pas beaucoup de sens. Les deux solutions présentées dans cet article sont en effet à l'opposé l'une de l'autre sur beaucoup de points.

Le premier produit est commercial, il a un coût, possède une bonne documentation, et les intégrateurs pouvant vous aider dans votre démarche sont nombreux. Linux VServer est un logiciel libre, gratuit, avec une documentation laissant encore à désirer et demandant une bonne connaissance de Linux.

Au niveau technologique, VMware ESX reproduit une machine physique à base de processeur x86, dans laquelle il est possible d'installer un système d'exploitation complet, y compris le noyau. L'administrateur ne verra aucune différence avec un vrai serveur. Il n'aura aucune restriction, si ce n'est qu'il ne pourra pas toucher physiquement le matériel. Linux VServer n'est rien d'autre qu'un espace restreint où sont cloisonnés un ou plusieurs processus Linux. Le super-utilisateur ne peut pas y faire les mêmes actions que sur un vrai serveur, il est « contrôlé » et ses capacités réduites. Ceci peut être un avantage ou un inconvénient, selon le but recherché.

VMware fournit de nombreux et très importants outils pour la gestion et la supervision des machines virtuelles, ce qui n'est pas le cas de Linux VServer.

Nous allons maintenant essayer de résumer rapidement les éléments essentiels de chaque solution.

8.1 Points clés de Linux VServer

- Libre et gratuit.
- Virtualise un environnement d'exécution pour des applications Linux.
- Performances excellentes.
- Utilisable en dehors du champ de la consolidation : un vserver compromis, même avec un accès super-utilisateur, sera éventuellement moins dangereux grâce aux restrictions imposées.

8.2 Points clés de VMware

- Produit commercial.
- Matériel supporté restreint.
- Virtualise une machine x86 complète, pour n'importe quel système d'exploitation.
- Actions « réversibles ».
- Sécurité optimale, avec un cloisonnement entre les machines virtuelles.
- Outils très précieux

9 Conclusion

Nous avons implanté avec beaucoup de satisfaction les deux solutions décrites dans cet article, pour répondre à une problématique double d'isolation de serveurs et de limitation du gaspillage des ressources. L'investissement humain et financier est justifié par la simplification de la mise à disposition de nouveaux « serveurs » et par les fonctionnalités apportées. Mais, l'aspect supervision des serveurs hôtes est à intégrer dans la démarche.

De nombreuses innovations (ouverture du code de ESX aux partenaires, arrivée des architectures Intel VT et AMD Pacifica permettant par exemple à Xen version 3 de faire tourner des systèmes d'exploitation non modifiés, etc.) montrent que ces technologies de virtualisation suscitent un intérêt soutenu, et tout laisse à penser qu'elles occuperont une place plus importante dans nos centres informatiques.

Enfin, il faut savoir les intégrer dans une démarche plus vaste pour mieux répondre à la demande et plus vite, en garantissant aussi une meilleure disponibilité.

Bibliographie

[1] Intel Corporation, *IA-32 Intel® Architecture Software Developer's Manual, Volume 1 : Basic Architecture*. Intel Corporation, Juin 2005

[2] John Scott Robin et Cynthia E. Irvine, Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. Dans *Proceedings of the 9th USENIX Security Symposium*, Denver, Août 2000.

[3] Gerald J. Popek et Robert P. Goldberg, Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7) : 412-421, Juillet 1974

[4] Projet Linux VServer : http://www.linux-vserver.org

[5] VMware Incorporated, *VMware ESX Server 2 Architecture and Performance Implications*. VMware Incorporated, Août 2005

[6] Ron Oglesby et Scott Herold, *VMware ESX Server Advanced Technical Design Guide*. Brian Madden, Août 2005

[7] Herbert Pötzl, La Technologie Linux-VServer : http://linux-vserver.org/Linux-VServer-Paper-French, 2004

[8] Linux VServer, répertoire de configuration : http://www.nongnu.org/util-vserver/doc/conf/configuration.html

[9] Documentation VMware ESX et VirtualCenter : http://www.vmware.com/support/pubs

[10] Rami Rosen, Introduction to the Xen Virtual Machine. *Linux Journal*, Septembre 2005 http://www.linuxjournal.com/article/8540

[11] Susanta Nanda et Tzi-cker Chiueh, A Survey on Virtualization Technologies. *RPE Report*, State University of New York at Stony Brook, Février 2005