h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Performances d'un système virtualisé avec VMware ESX-ESXi

Travail de Bachelor

Session 2010

Professeur responsable : LITZISTORF Gérald
En collaboration avec : DELACOMBAZ Christian (CTI)
Diplômant : PENAS Cédric
Filière Informatique
Laboratoire de transmission de données

Table des matières

Avant-propos

Conventions Typographiques

Afin de faciliter la lecture de ce document, il nous faut définir les quelques règles typographiques suivantes qui seront appliquées :

- Le texte par défaut du document est en Computer Modern 11pt
- Les URL sont en bleu (`http://www.exemple.com`)
- Les noms des fichiers sont en italique, gras et en gris (***/dossier/exemple.conf***)
- Les contenus des fichiers sont en encadré et en Teletype (`Exemple de contenu`)
- Les commandes tapées dans le terminal sont en blanc sur fond noir

```
exemple de commande
```

Structure du document

Ce mémoire de diplôme se divise en 8 chapitres :

1. **Généralités**
   Introduction au projet de diplôme et à son déroulement

2. **Notions importantes**
   Description de toutes les notions importantes à connaître qui ont été utilisées dans ce projet

3. **Matériel à disposition**
   Récapitulatif des infrastructures et du matériel utilisé

4. **Outils utilisés**
   Description des outils qui ont été utilisés lors de ce travail de diplôme

5. **Analyse & optimisation des performances**
   Guide pratique sur le monitoring et l'analyse des performances

6. **Labo TD**
   Description du travail réalisé à l'HEPIA

7. **Stage au CTI**
   Description du travail réalisé au CTI

8. **Conclusion**

Remerciements

Je tiens à remercier M. Gérald LITZISTORF pour m'avoir proposé ce projet de diplôme et m'avoir suivi dans mon travail tout au long des huit semaines en me conseillant de la meilleure façon possible.

Je remercie également M. Philippe LOUTAN de m'avoir accueilli au CTI et de m'avoir permis d'étudier la virtualisation dans un environnement professionnel. Et plus particulièrement, M. Christian DELACOMBAZ pour m'avoir suivi durant 5 semaines et pour m'avoir fourni de précieux conseils et pour son aide dans certaines parties de ce travail.

1 Généralités

1.1 Énoncé du projet

La problématique des performances d'un système d'information reste présente dans le quotidien des activités de l'ingénieur système ; qu'il s'agisse d'atteindre un niveau de qualité suffisant pour les utilisateurs ou pour optimiser les investissements consentis.

Les spécialistes s'accordent aujourd'hui à reconnaître que l'utilisation d'une couche de virtualisation augmente la complexité de l'ensemble et rend donc l'analyse des performances plus difficile.

La finalité de cette étude vise la bonne maîtrise des performances d'une architecture virtualisée avec VMware ESX-ESXi, afin d'en identifier les goulets d'étranglement ainsi que les éventuelles opportunités d'optimisation.

1.2 Organisation du temps

Ce travail de diplôme s'est étalé sur huit semaines et a été réalisé dans deux cadres totalement différents.

Les deux premières semaines ont été consacrées à la recherche de documentations et à un tour d'horizon sur ce qui est possible de faire dans le domaine des performances en virtualisation. Cette partie s'est déroulée au laboratoire de transmission de données de l'HEPIA (6).

Les cinq semaines suivantes se sont passées en stage au CTI (7) qui est le centre des technologies de l'information de l'État de Genève. Elles auront eu pour but d'analyser la façon dont la virtualisation est utilisée en entreprise, quelles sont les contraintes, les technologies utilisées et bien entendu mettre un maximum en pratique ce qui a été réalisé durant les deux premières semaines.

La dernière semaine de ce projet de diplôme a été à nouveau réalisée à HEPIA, afin de finaliser le mémoire et terminer d'étudier quelques points théoriques.

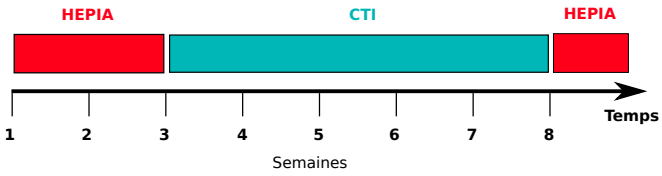


FIGURE 1.1 – Organisation du temps

1.3 Introduction aux performances

A l'heure actuelle, un grand nombre d'ordinateurs ne fonctionnent qu'à 10-15% de leur capacité de calcul et gaspillent ainsi de la ressource qui pourrait servir à des tâches parallèles. La virtualisation permet d'augmenter ce taux d'utilisation aux alentours de 85%, en permettant l'exécution de plusieurs systèmes d'exploitation simultanément sur une même machine physique.

Cependant, il existe de nombreuses différences entre la gestion d'une machine physique et celle d'une machine virtuelle. La couche ajoutée par l'hyperviseur, qui permet de faire le lien entre le matériel physique et virtuel, amène une complexité supplémentaire rendant l'étude des performances beaucoup plus difficile.

Il faut donc bien comprendre tous les mécanismes qui ont été ajoutés par rapport à une machine physique traditionnelle pour pouvoir analyser et optimiser les performances d'un système virtualisé.

La plupart des outils de mesure de performances que nous avons l'habitude d'utiliser sur des machines physiques, peuvent également être utilisés sur des systèmes virtualisés pourvu qu'il soit pris en compte dans l'analyse des résultats que les tests ne se sont pas faits directement sur le matériel physique afin de comprendre les différences qu'il pourrait y avoir.

Ce travail de diplôme n'a pas la prétention d'être le guide absolu des performances sur les systèmes virtualisés, mais il a pour but d'amener le lecteur à réfléchir aux conséquences de la virtualisation et ses impacts sur les performances du système tout entier. De plus, son autre objectif est de faire découvrir les notions importantes qui entrent en jeu, ainsi que d'expliquer les manipulations et recommandations indispensables qui optimisent la couche de virtualisation.

2 Notions importantes

2.1 Virtualisation VMware [1]

VMware est une société qui a démarré en 1998 et qui a acquis énormément d'expérience dans la virtualisation. Aujourd'hui, elle est l'un des *leaders* dans ce domaine, grâce à VMware vSphere 4 qui est actuellement leur dernier produit sorti sur le marché.

Il s'agit en réalité plus d'une suite d'outils, car elle regroupe plusieurs composants dont certains ont été utilisés pour ce projet de diplôme.

On retrouve parmi ceux-ci :

- **VMware ESX/ESXi** est le système d'exploitation, appelé *hyperviseur*, s'installant sur le matériel physique et qui permet de remplir son rôle de couche de virtualisation en présentant aux machines virtuelles du "matériel" virtualisé.

- **VMware vCenter Server** est un serveur central dans l'infrastructure de virtualisation, car il permet de regrouper la configuration et l'administration de tous les ESX/ESXi disponibles.

- **VMware vSphere Client** est un client avec un GUI s'installant sur un poste de travail et qui permet d'administrer soit directement ESX/ESXi, soit de passer par VMware vCenter Server pour administrer un groupe d'ESX/ESXi.

- **VMware vSphere Web Access** est une interface web d'administration des machines virtuelles et qui permet également de faire du *remote console*, principalement sur ESX.

En parallèle nous avons des mécanismes qui sont utilisés par ces différents outils, afin d'améliorer les performances des ESX/ESXi et des machines virtuelles.

- **VMware Virtual SMP** est une fonctionnalité permettant à une machine virtuelle d'utiliser plusieurs CPU physique simultanément.

- **VMware vMotion** est une fonctionnalité permettant à une machine virtuelle de migrer d'un hôte ESX/ESXi vers un autre sans arrêt de la machine ni d'interruptions [2].

- **VMware High Availability (HA)** est une fonctionnalité permettant d'avoir de la haute disponibilité pour une machine virtuelle.

1. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_intro_vs.pdf* »
2. « *http://www.tdeig.ch/vmware/sandmeier_M.pdf* »

- **VMware Distributed Resource Scheduler (DRS)** est une fonctionnalité permettant de faire du *load balancing* entre les ESX/ESXi d'un même *cluster*.

Il en existe encore d'autres, mais qui ne sont pas utilisées dans ce projet, de ce fait nous n'entrerons pas dans les détails.

2.1.1 Organisation Infrastructure

Nous retrouvons donc dans une infrastructure de virtualisation tous les outils cités en 2.1 avec quelques nouvelles notions qui font leur apparition comme :

- Datacenter
- Hosts
- Cluster
- Resource Pool

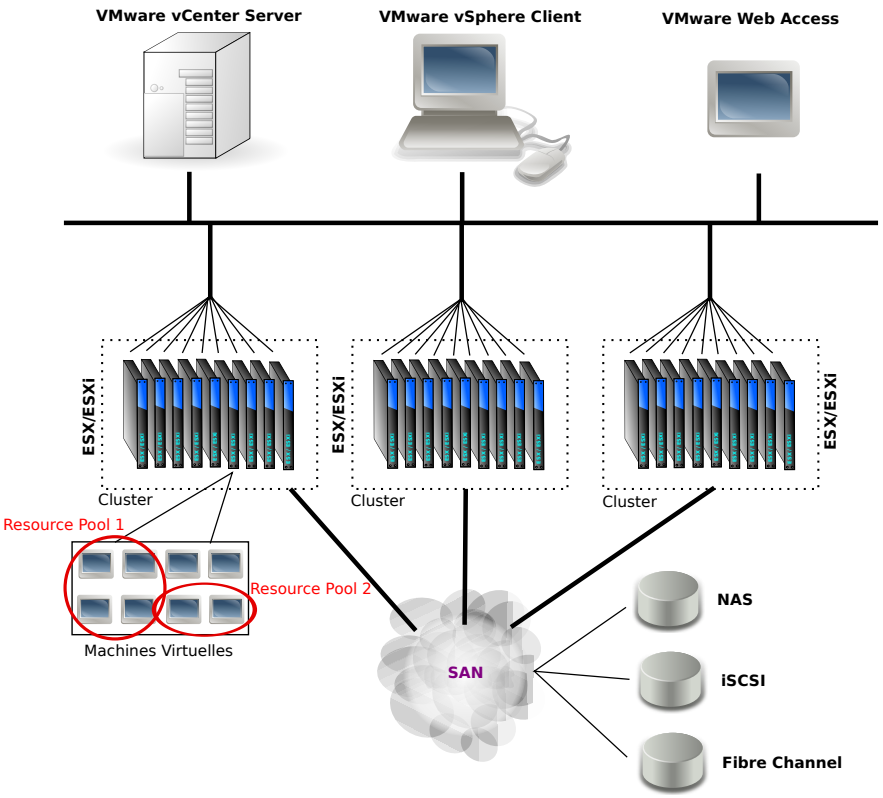


FIGURE 2.1 – VMware Datacenter

Ces notions servent avant tout à organiser la distribution des ressources au sein du *Datacenter* qui regroupe tout le matériel physique destiné à la virtualisation. Un *Datacenter* peut servir à séparer l'infrastructure de production et de test comme c'est le cas dans la plupart des entreprises utilisant la virtualisation.

Par *Host*, VMware désigne un ESX/ESXi installé sur une machine physique qui dispose normalement de toute la ressource disponible. Ainsi, si nous disposons physiquement de 4 CPUs dual-core de 4GHZ chacun et de 32GB de RAM, alors l'*host* disposera de 32GHZ de puissance de calcul et de 32GB

de RAM qu'il pourra distribuer à ses machines virtuelles.

Par *Cluster*, VMware désigne un groupe d'*hosts*. Le seul but pratique de créer des *clusters* est de pouvoir ainsi activer des mécanismes comme HA et DRS, afin de pouvoir créer de la haute disponibilité et du balance de charges entre les *hosts*.

Pour terminer, un *Resource Pool* est un sous-ensemble des ressources dont dispose le *host*, qui peut être attribué soit à un *host* particulier, soit à un *cluster*, voir même à un groupe de machines virtuelles. Par exemple, avec nos 32GHZ de CPUs et 32GB de RAM, nous pourrions décider que trois *hosts* ne disposeraient que de 10GHZ et 12GB de RAM. Il s'agirait, ici, d'un *Resource Pool* et par conséquent, le *cluster* ou l'*host* à qui serait attribué ce *pool* ne pourrait pas dépasser la quantité de ressources qui lui a été fixé.

2.1.2 ESX-ESXi [3]

VMware ESX/ESXi est la partie principale d'un *host*, puisqu'il s'agit de l'hyperviseur qui sert de couche de virtualisation. Il s'installe directement sur le matériel comme un système d'exploitation traditionnel. Son noyau est basé sur la distribution Linux Redhat 7.3 et dispose d'une grande partie des commandes de cet OS.

Ce système d'exploitation se compose de 3 parties principales :

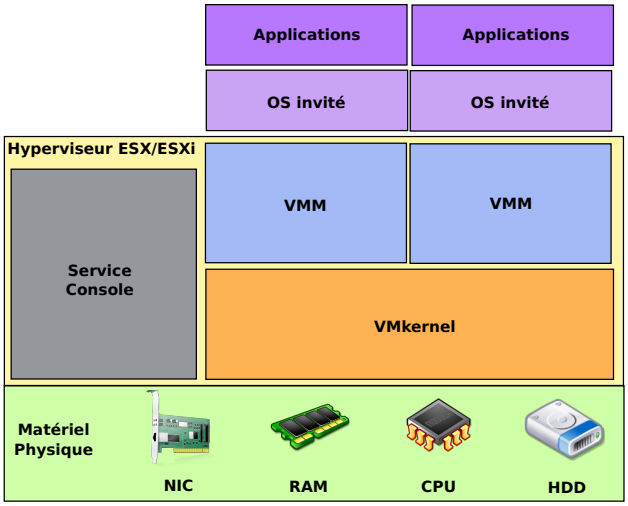


FIGURE 2.2 – Architecture ESX/ESXi

- **VMkernel** est le noyau de l'OS et contient tous les processus importants au bon fonctionnement du système.
- **Service Console** (disponible que pour ESX) est une machine virtuelle particulière permettant les connexions distantes, via le port 22 (SSH), ou locales pour l'administration du système en lignes de commandes.
- **VMM** sert à présenter aux VMs le matériel virtualisé et à faire le lien entre celui-ci et le matériel physique. Il existe un VMM par machine virtuelle.

3. http://fr.wikipedia.org/wiki/VMware

2.1.3 vMotion [4]

La fonctionnalité de vMotion permet de migrer une machine virtuelle d'un *host* vers un autre sans devoir l'éteindre (à chaud) et cela théoriquement sans interruption de services.

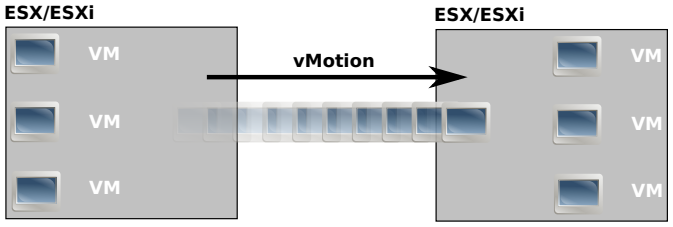


FIGURE 2.3 – VMware vMotion

2.1.4 HA [5]

La fonctionnalité de High Availability (HA) permet d'avoir de la haute disponibilité des machines virtuelles. Dans le cas où un *host* ESX/ESXi tomberait en panne, le mécanisme s'occupe de déplacer automatiquement les VMs touchées sur un autre *host* grâce au vMotion.

Pour que ceci puisse fonctionner, il faut obligatoirement que les *hosts* soient mis en *cluster* et que le HA soit activé manuellement via VMware vCenter Server.

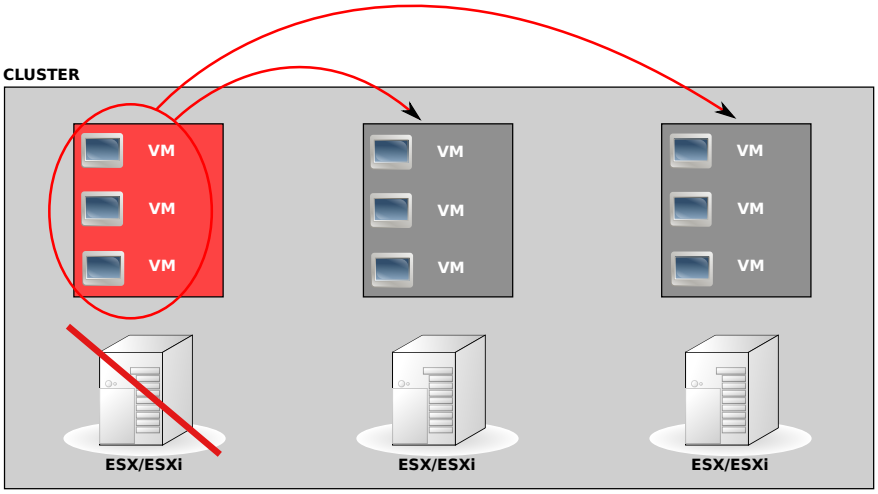


FIGURE 2.4 – VMware HA

Ce principe de HA fonctionne grâce à un agent qui tourne sur chaque *host* et qui maintient un lien (*heartbeat*) avec les autres *hosts* du *cluster*. Dès que ce lien cesse de fonctionner à cause d'une panne, le processus de migration est enclenché et les VMs sont déplacées sur un *host* en bonne santé.

4. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_intro_vs.pdf (p.14-15)* », « *http://www.tdeig.ch/vmware/sandmeier_M.pdf* »
5. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_intro_vs.pdf (p.16)* »

2.1.5 DRS [6]

La fonctionnalité de Distributed Resource Scheduler (DRS) permet d'avoir de la balance de charges entre les *hosts* d'un même *cluster*. Tout comme pour le HA, il faut l'activer sur chaque *host*.

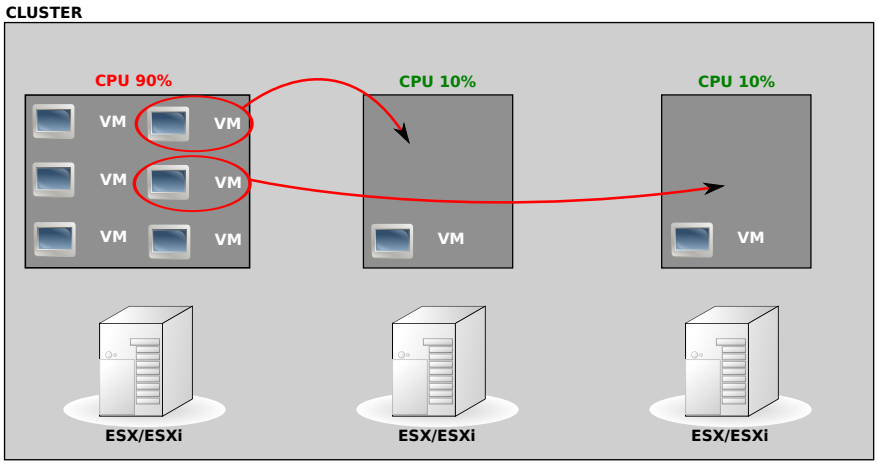


FIGURE 2.5 – VMware DRS

Par défaut, le mécanisme de DRS essaie de placer, grâce à vMotion, les VMs sur un sous-groupe du *cluster* et d'éteindre les *hosts* qui ne servent à rien. Lorsque la demande en ressource augmente, les *hosts* éteints peuvent être rallumés et les VMs déplacées sur ceux-ci. DRS compare donc en permanence les ressources disponibles et la demande, afin de placer de la meilleure façon possible les VMs sur les *hosts*.

2.1.6 VMware Tools [7]

Une des particularités de VMware, par rapport à d'autres solutions de virtualisation, est de pouvoir installer une suite d'outils appelés « VMware tools ». Ces « VMware tools »s'installent directement sur l'OS invité et sont disponibles pour les systèmes d'exploitation suivants :

- Windows
- Linux
- FreeBSD
- Netware

Ils permettent d'améliorer les fonctionnalités des systèmes invités et des interactions entre les invités et les hôtes. Ils permettent, par exemple, d'éteindre proprement un OS invité depuis l'hôte ou encore de copier des fichiers entre l'invité et l'hôte.

Ces « VMware tools »se divisent en quatre parties :

– **VMware Device Drivers**

Peut-être ce qui est le plus important dans les « VMware tools »à savoir les *drivers*. Tout comme

6. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_intro_vs.pdf (p.15-16)* »
7. http://www.virtuatopia.com/index.php/Understanding_and_Installing_VMware_Tools

sur un système traditionnel, il faut des *drivers* pour le matériel virtuel.

En effet, la couche de virtualisation VMware ou plutôt le VMM s'occupe de gérer le matériel physique de l'hôte et de le présenter sous forme de matériel virtualisé au système invité. Par exemple, l'hôte pourrait avoir une carte réseau Intel Gigabit Ethernet et l'invité verrait une AMD PCnet-PCI II.

Sans les « VMware tools », certains OS n'arrivent pas à faire le lien entre ces deux cartes et donc la connexion ne se fait pas. C'est pourquoi, pour des performances optimales au niveau de la compatibilité *hardware*, il est conseillé de les installer.

– **VMware Tools Service**

Ce composant des *tools* est un service tournant en tâche de fond du système d'exploitation invité.

- *VMwareService.exe* (Windows)
- *vmware-guestd* (Linux)

Une des principales fonctions de ce service est d'envoyer régulièrement un *heartbeat* à l'hôte, ce qui lui permet de savoir si une machine virtuelle est active ou non.

Sa deuxième fonction est de maintenir un canal de communication entre le système hôte et l'invité. Un système d'exploitation n'est normalement pas prévu pour accepter des requêtes du serveur VMware, à part en paravirtualisation, et c'est pourquoi ce processus existe. Il va permettre de, par exemple, pouvoir dire à une machine virtuelle de s'arrêter ou de redémarrer depuis l'interface de management web. Il va donc servir à exécuter des commandes qui ne sont pas accessibles depuis l'extérieur.

D'autres fonctionnalités sont disponibles grâce à ce service comme par exemple garder une synchronisation de l'horloge (NTP) entre l'hôte et l'invité, ou encore gérer la souris dans une console (Windows).

– **VMware User Process**

Tout comme « VMware Tools Service », il s'agit d'un processus qui tourne en tâche de fond.

- *VMwareUser.exe* (Windows)
- *vmware-user* (Linux, FreeBSD et Solaris)

Il se lance normalement automatiquement dès qu'une interface graphique est présente dans le système invité. Il permet par exemple de faire du copier-coller entre la *remote console* et le *desktop* de l'hôte.

Il amène également une gestion de la souris et de la résolution pour les systèmes Linux tout comme « VMware Tools Service »pour Windows.

– **VMware Tools Control Panel**

Ce composant offre un panneau de contrôle pour les « VMware tools », afin de configurer certaines options.

Depuis Windows, il se lance en allant dans *Démarrer -> Panneau de configuration* et en cliquant sur l'icône *VMware Tools*. Ce qui devrait faire apparaître le panneau (FIG 2.6).

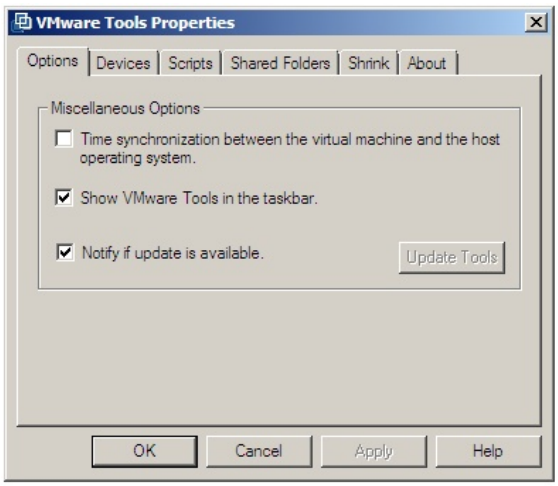


FIGURE 2.6 – VMware Tools Control Panel

Depuis Linux, il suffit de taper la commande suivante dans la console :

```
/usr/bin/vmware-toolbox&
```

2.1.7 Virtual Hardware Version

Le « Virtual Hardware »est un composant qui vient en supplément des « VMware Tools »dans lequel on retrouve tout le matériel virtuel utilisé par VMware. Il contient également les drivers associés à chaque composant et quelques fonctionnalités.

Avec *VMware Virtual Infrastructure 3.5* ce composant « Virtual Hardware »était disponible en version 4 et avec l'arrivée de *vSphere 4*, il est passé en version 7. A noter aussi qu'il est possible de faire fonctionner une version 4 sous *vSphere 4* mais pas l'inverse avec une version 7 sous *VMware VI 3.5*. Passons à présent aux changements, que la nouvelle version apporte, dont voici une liste non exhaustive :

- **USB :** Intégration d'un controller USB qui était déjà présent dans VMware Workstation, mais pas dans ESX 3.5.
- **Hot Plug support :** Permet de rajouter à chaud aux VMs de la mémoire, des vCPUs ainsi que des périphériques. Il faut néanmoins que le système d'exploitation le supporte. Voir la liste des OS avec leurs compatibilités (FIG 2.1.7).

Système d'exploitation	Hot-Add Memory	Hot-Add CPU
Windows Server 2008 Datacenter Edition x64	Oui	Oui
Windows Server 2008 Datacenter Edition x86	Oui	Non
Windows Server 2008 Enterprise Edition x64	Oui	Non
Windows Server 2008 Enterprise Edition x86	Oui	Non
Windows Server 2008 Standard Edition x64	Oui	Non
Windows Server 2008 Standard Edition x86	Oui	Non
Windows Server 2008 Web Server x64	Oui	Non
Windows Serve 2008 Essential Business Server Premium x64	Oui	Non
Windows Server 2008 Essential Business Server Standard x64	Oui	Non
Windows Server 2008 Small Business Server Premium	Oui	Non
Windows Server 2008 Small Business Server Standard	Oui	Non
Windows Server 2003 Enterprise Edition x64	Oui	Non
Windows Server 2003 Enterprise Edition x86	Oui	Non

FIGURE 2.7 – Compatibilité OS

- **256GB RAM :** Un ESX/ESXi peut maintenant supporter jusqu'à 256GB de mémoire RAM.

- **VMXNET Generation 3 :** Nouveau controller réseau qui permet d'apporter des améliorations, comme la gestion de l'IPv6, optimisation du trafic réseau etc...

- **8-way SMP :** Dans les versions précédentes vSphere 4, une VM ne pouvait se voir allouer que quatre vCPUs au maximum. Avec la nouvelle version, nous pouvons aujourd'hui monter jusqu'à huit vCPUs pourvu que l'OS les supporte.

- **VMDirectPath :** Permet d'améliorer les performances des vCPU lorsque ceux-ci ont besoin de faire beaucoup d'accès aux périphériques I/O en permettant à la VM de s'adresser directement au hardware de l'hôte.

- **Block Tracking :** Permet d'améliorer les opérations de sauvegarde et de restauration.

- **Serial Attached SCSI (SAS) virtual devices :** Permet le support des configurations en *clustering* de Windows Server 2008.

- **IDE virtual device :** Permet le support des anciens systèmes d'exploitation qui ne gèrent pas les pilotes SCSI.

2.1.7.1 Identifier la version utilisée

Sous vSpere Client, il suffit de se mettre sur la vue d'un cluster, puis d'aller sur l'onglet « Virtual Machines ». Il devrait normalement y avoir une colonne "VM Version" (FIG 2.8). Si ce n'est pas le cas, il suffit de faire un clique droit sur le haut d'une colonne et de l'ajouter.

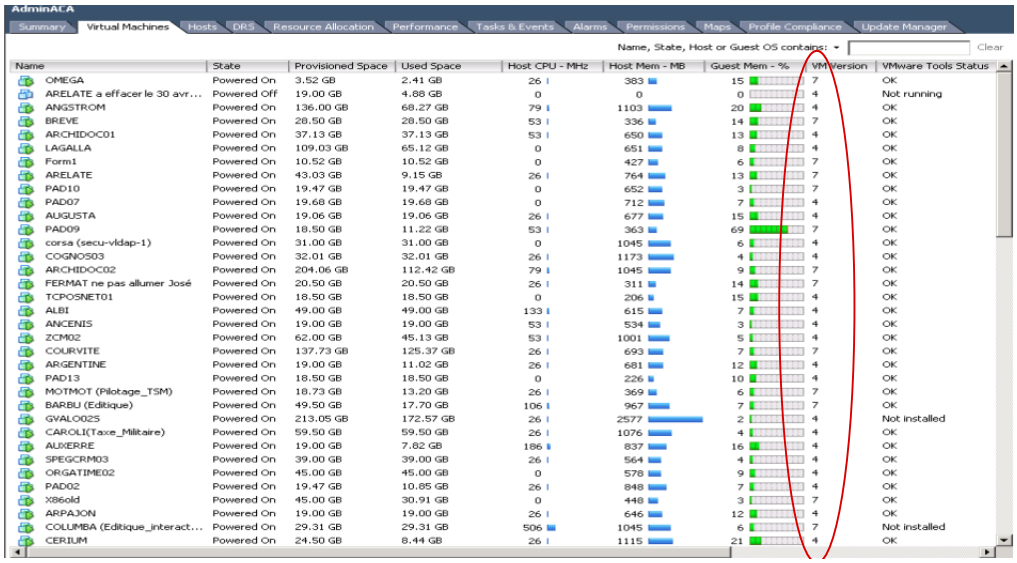


FIGURE 2.8 – Hardware Version

2.2 Architecture x86

2.2.1 Gestion de la mémoire

Dans cette section, la gestion de la mémoire sera décrite de manière globale et très synthétique. Le but n'étant pas de faire un cours détaillé, mais de rafraîchir la mémoire du lecteur sur les principes de base. Les chiffres qui seront utilisés pour les exemples seront en rapport direct avec le type de processeurs utilisés dans ce projet de diplôme, à savoir des processeurs Intel 64bits.

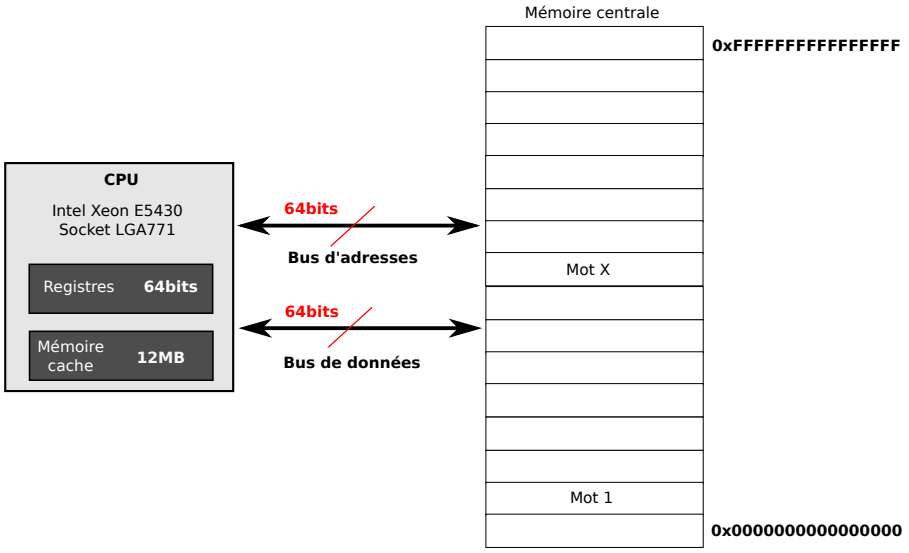


FIGURE 2.9 – Mémoire centrale

2.2.1.1 La mémoire physique [8]

La mémoire centrale physique d'un système qui est plus couramment surnommée « Mémoire RAM », peut être vue comme une suite de blocs (des mots) composés de 1, 2, 4, 8 octets selon l'architecture à disposition, dans notre cas 64 bits soit 8 octets.

La quantité de mémoire utilisable est définie par la grandeur du bus d'adresses qui relie le processeur à la mémoire (FIG 2.9). Il peut donc adresser 2^{64} mots ce qui aujourd'hui ne fixe aucune limite, puisqu'aucune carte mère actuelle ne pourrait supporter autant de mémoire RAM.

Il s'agit plus précisément de mémoire de type DRAM [9] qui travaille conjointement avec une mémoire cache se trouvant à l'intérieur du processeur et qui est de type SRAM [10], beaucoup plus rapide d'accès mais également plus chère.

Lorsque le processeur désire accéder à un mot mémoire, il le recherche d'abord dans ce cache et s'il ne s'y trouve pas, il le recherchera en mémoire centrale.

2.2.1.2 La mémoire virtuelle [11]

Dans les systèmes récents, la gestion de la mémoire est faite par le système d'exploitation avec l'aide d'une partie du processeur appelée MMU [12].

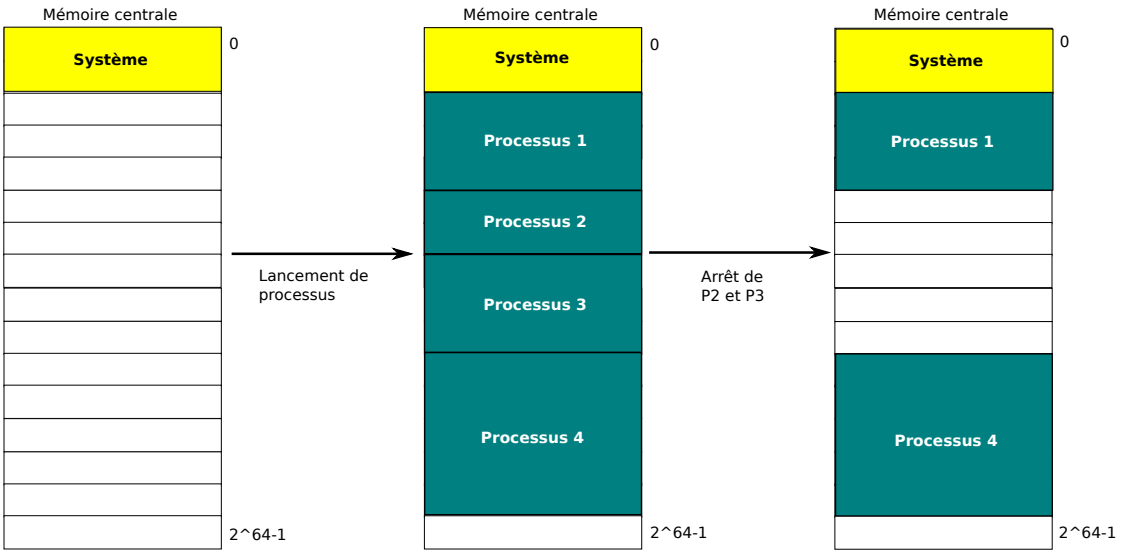


FIGURE 2.10 – Mémoire centrale & processus

En effet, les premiers systèmes allouaient la mémoire aux processus de façon linéaire et sans découpage. Nous pouvons facilement voir sur la figure 2.10 quel est le problème de cette solution. Le système

8. http://www.commentcamarche.net/contents/pc/ram.php3
9. Dynamic Random Access Memory
10. Static Random Access Memory
11. http://kurzweg.info/cnam/smb137/partie3/ar01s01.html, http://fr.wikipedia.org/wiki/M%C3%A9moire_virtuelle, http://deptinfo.unice.fr/twiki/pub/Minfo05/ApprofondissementSysteme/04_GestionMemoire.pdf, http://www.lis.inpg.fr/pages_perso/cayre/pub/os/os-3.pdf
12. Memory Management Unit

d'exploitation considère l'espace d'adressage d'un processus comme étant insécable et du coup il est obligé de le charger entièrement en mémoire pour pouvoir l'exécuter.

Ce qui nous embête, c'est que si les "trous" laissés entre les processus en mémoire centrale sont trop petits pour accueillir un nouveau processus, il faudra faire de la place. Une solution envisageable est de lancer une défragmentation pour boucher ces "trous", mais c'est une tâche longue à réaliser.

Un autre problème est que si un processus dépasse la capacité de la mémoire, il ne pourra pas être exécuté. Mais, la plus grosse difficulté, dans une architecture de ce type, est qu'il faut toujours savoir l'adresse exacte où doit être chargé un processus, ce qui implique une connaissance poussée de la mémoire.

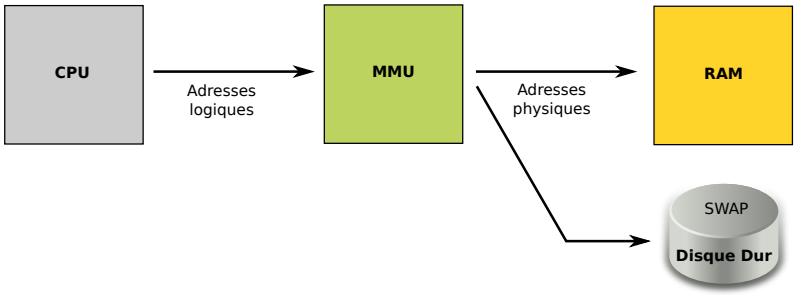


FIGURE 2.11 – MMU

C'est pourquoi le mécanisme de mémoire virtuelle à été mis au point dans les années 1960. Il permet d'utiliser une mémoire de masse comme un disque dur en complément de la mémoire centrale (ce qui est appelé SWAP), et ainsi pouvoir exécuter de plus gros processus que la capacité de la mémoire ne le permettrait ou tout simplement faire tourner plus de processus en même temps (FIG 2.11).

C'est ici qu'entre en jeu le MMU, car ce mécanisme implique que le processeur ne pourra plus utiliser directement des adresses physiques, mais il utilisera des adresses virtuelles qui seront traduites par le MMU afin de pointer sur l'emplacement correct. Chaque processus exécuté par le processeur possédera son propre espace d'adressage virtuel et croira ainsi posséder la totalité de la mémoire disponible.

Comme nous pouvons le voir, il y a trois types d'adresses :

1. Les adresses logiques ou « segmentées »(**Segmentation**) sont les adresses qui sont directement utilisées par les programmes en langage machine et sont composées d'un couple (segment, offset).
2. Les adresses linéaires ou « paginées »(**Pagination**) sont celles qui permettent d'adresser toute la mémoire virtuelle et elles vont de 0x0000000000000000 à 0xFFFFFFFFFFFFFFFF.
3. Les adresses physiques sont celles qui pointent directement vers la mémoire RAM.

Pour réaliser ce mécanisme, il faut introduire les notions de mémoire paginée ou segmentée qui permettent de "diviser" un processus en plusieurs parties.

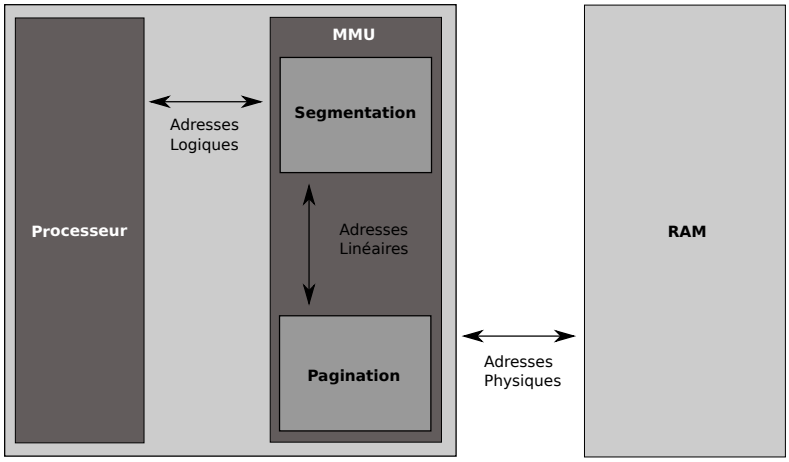


FIGURE 2.12 – Adressage dans les processeurs x86

2.2.1.3 La pagination [13]

La pagination est un mécanisme qui permet de découper la mémoire ainsi que les processus, respectivement en cadre de pages et en pages qui sont des blocs de taille fixe (4Ko pour les x86). Avec cette méthode le système d'exploitation pourra charger en mémoire centrale des morceaux de processus (FIG 2.13) et par conséquent utiliser l'espace disponible d'une meilleure façon.

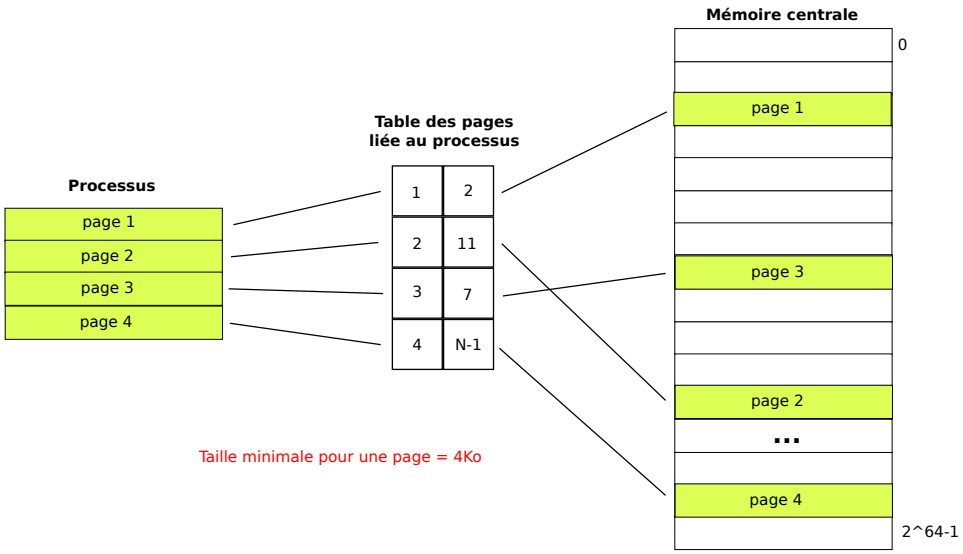


FIGURE 2.13 – Table des pages

Étant donné qu'un processus est découpé en pages, les adresses utilisées seront des adresses paginées, où un octet de la mémoire centrale est représenté par un couple (numéro de la page, déplacement dans la page). Il sera traduit en adresse physique grâce à une table de pages contenue dans le MMU (qui

13. http://kurzweg.info/cnam/smb137/partie3/ar01s02.html, http://deptinfo.unice.fr/twiki/pub/Minfo05/ApprofondissementSysteme/04_GestionMemoire.pdf

est aujourd'hui intégré au CPU) qui contient les relations entre les adresses virtuelles et les adresses physique (FIG 2.14). Cette relation est mise à jour continuellement par le système d'exploitation afin de savoir exactement où se trouve chaque page dans la mémoire.

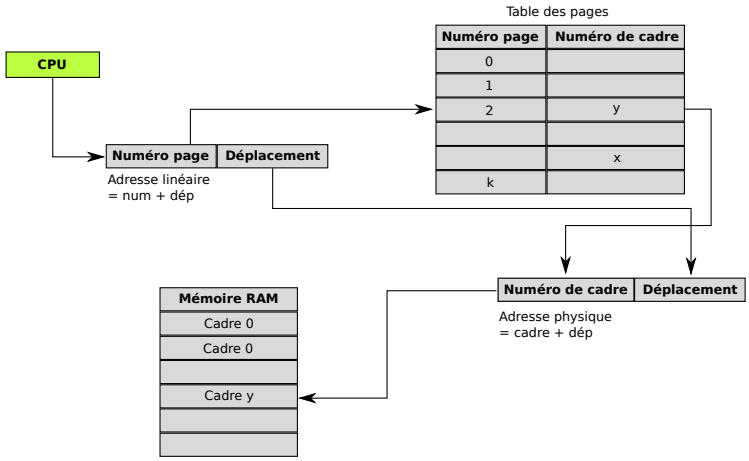


FIGURE 2.14 – Mécanisme de pagination

Chaque processus possède sa propre table de pages dans le cas où la pagination est la seule technique utilisée. Nous verrons par la suite qu'avec une combinaison des méthodes, cela devient différent.

2.2.1.4 La segmentation [14]

Le principe de la segmentation est sensiblement équivalent à celui de la pagination, à la différence près qu'à la place de découper les processus en pages de taille fixe, ils sont découpés en segments de taille variable. Ces segments sont créés par le compilateur qui les divise selon leur contenu et ils représentent un espace d'adressage linéaire composé d'adresses contiguës. Ainsi, nous aurons des segments de données ou de code, ce qui correspond à une partie logique du processus.

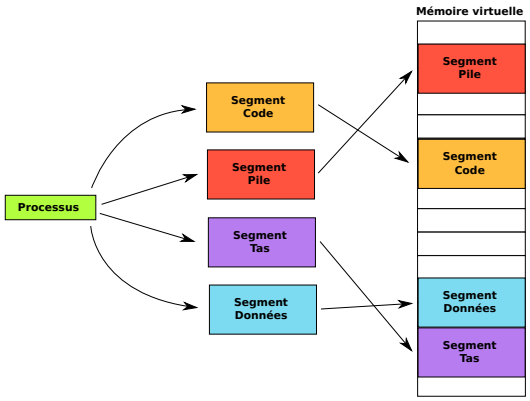


FIGURE 2.15 – Segmentation d'un processus

14. http://kurzweg.info/cnam/smb137/partie3/ar01s03.html, http://deptinfo.unice.fr/twiki/pub/Minfo05/ApprofondissementSysteme/04_GestionMemoire.pdf

Concrètement, ces segments sont divisés en 4 types :

- **Code :** correspond au code des fonctions.
- **Tas :** correspond à l'allocation dynamique des variables.
- **Pile :** correspond aux variables locales et paramètres, ainsi qu'aux adresses de retour des fonctions.
- **Données :** correspond aux constantes, variables initialisées et non initialisées.

L'adressage de ces segments se fait de la même façon que pour les pages avec un couple (segment, déplacement) et la résolution se fait grâce à une table de segments que le MMU peut consulter. Il existe également une table de segments par processus.

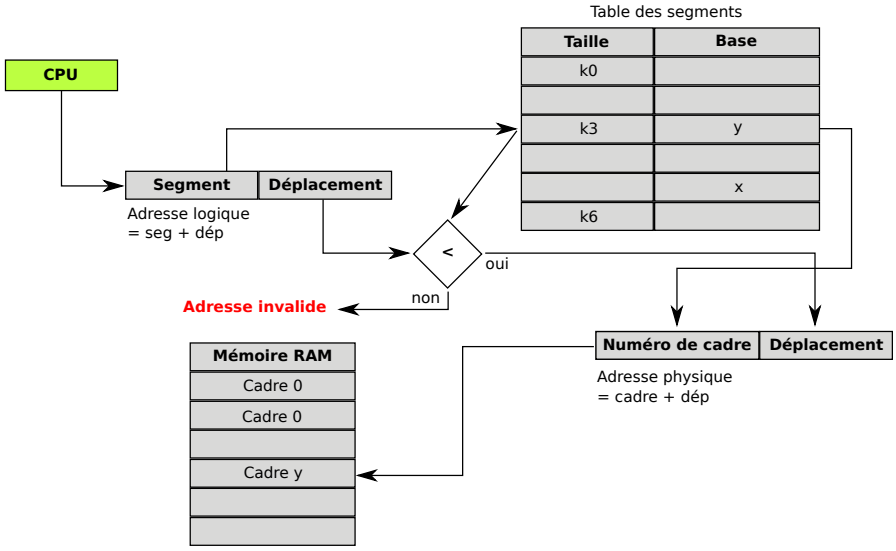


FIGURE 2.16 – Segmentation

2.2.1.5 Segmentation & Pagination

Dans les architectures modernes comme x86 ou x86_64, les deux mécanismes de segmentation et pagination sont utilisés conjointement.

Il faut voir la segmentation comme une structuration de l'espace d'adressage d'un processus et la pagination comme un moyen pour adapter la mémoire virtuelle à la mémoire physique.

Un processus est donc découpé en segments de différentes longueurs (une table des segments par processus) et ces segments sont découpés à leur tour en pages (une table des pages par segment) (FIG 2.17). C'est pourquoi, le MMU est composé d'une partie de segmentation et d'une autre pour la pagination (FIG 2.12).

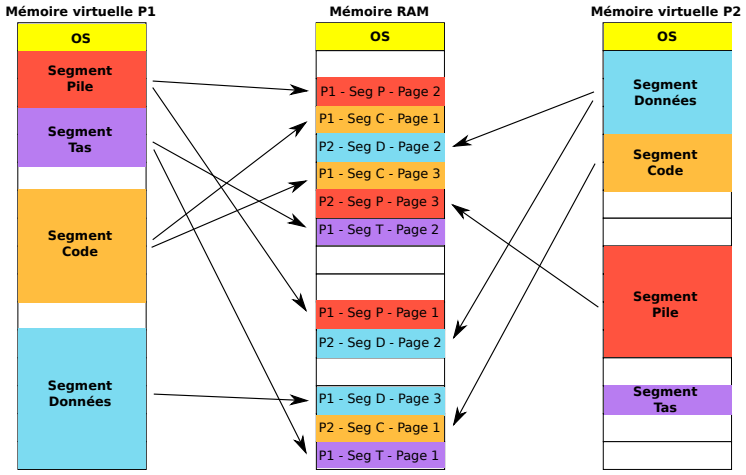


FIGURE 2.17 – Segmentation & Pagination des processus

Nous avons donc au départ des adresses logiques (segmentées) qui sont traduites en adresses virtuelles (paginées) par le MMU puis, celui-ci traduit encore une fois ces adresses en adresses physiques.

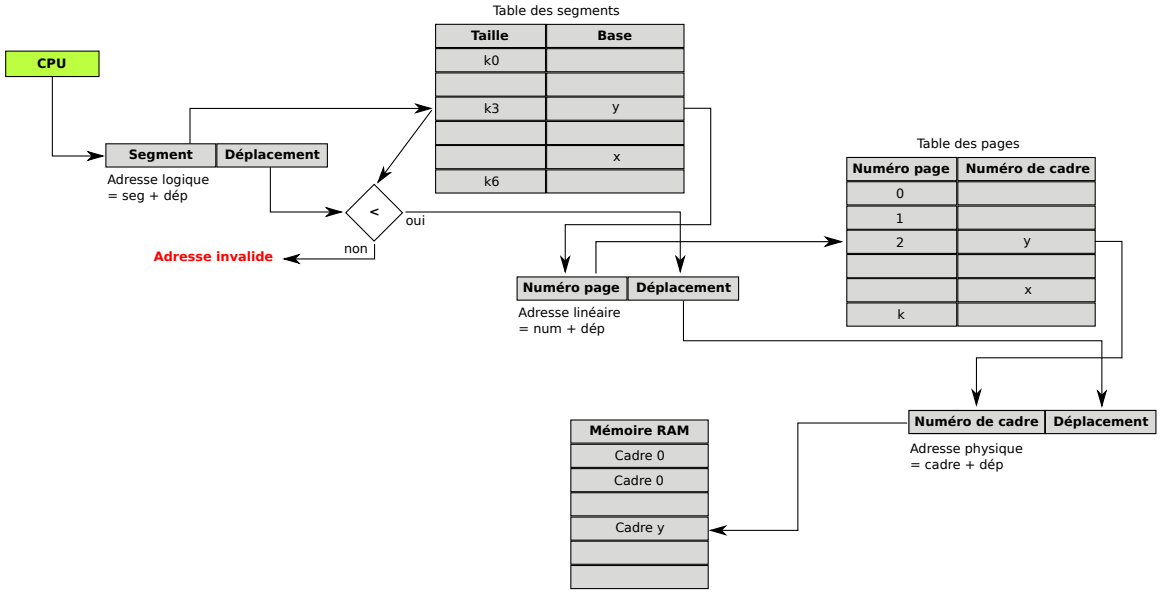


FIGURE 2.18 – Adressage global

2.2.2 ISA [15]

L'acronyme ISA vient de l'anglais pour « *instruction set architecture* »et est utilisé pour désigner l'architecture d'un processeur du point de vue du programmeur. Elle comprend plusieurs informations, comme un jeu d'instructions que le processeur peut exécuter, un ensemble de registres visibles par le programmeur, une organisation de la mémoire et des entrées-sorties, etc...

Voici une petite liste de quelques exemples d'ISAs :

- x86
- x86-64
- PowerPC
- MIPS
- ARM
- Motorola 68k
- IBM System/360
- IBM System/370

2.2.3 Les privilèges sur x86 [16]

Sur un processeur de la famille x86, chaque instruction de base ne s'exécute pas avec les mêmes privilèges. Ces protections ont été mises en place notamment pour éviter qu'une instruction n'accède à une partie de la mémoire qui ne devrait être accessible que par le système d'exploitation.

Il existe en tout quatre niveaux différents de privilèges qui se représentent la plupart du temps sous forme d'anneaux s'incluant les uns dans les autres en partant de l'anneau 0 (*ring 0*), qui possède le plus de privilèges pour arriver à l'anneau 3 (*ring 3*), qui est celui par défaut possédant des privilèges restreints (FIG 2.19).

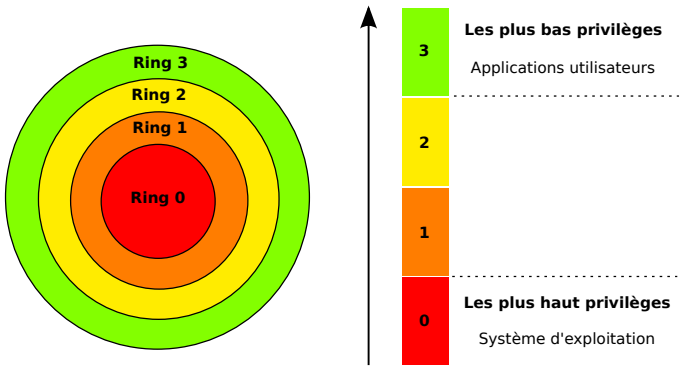


FIGURE 2.19 – Privilèges x86

Le système d'exploitation a tous les privilèges et, par conséquent, il se trouve au niveau 0 (appelé « mode noyau »), alors qu'une application utilisateur se trouvera toujours au niveau 3 (appelé « mode utilisateur ») limitant ainsi les infractions involontaires ou malveillantes envers la sécurité du système. Les niveaux 1 et 2 ne sont en général pas utilisés.

15. http://en.wikipedia.org/wiki/Instruction_set
16. http://fr.wikipedia.org/wiki/Anneau_de_protection

2.2.4 Virtualisation x86 [17]

La virtualisation est née dans les années 70 dans un centre de recherche IBM à Grenoble. Elle était alors pratiquée sur des *mainframes* de type IBM System/360-370 et était bien plus compliquée que ce que nous connaissons aujourd'hui, car elle faisait intervenir des technologies non seulement logicielles, mais également matérielles.

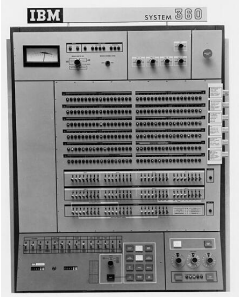


FIGURE 2.20 – IBM System/360

Ce qui nous intéresse dans l'historique de la virtualisation, c'est qu'à cette époque tous les systèmes virtualisés étaient basés sur le principe du « *Trap and Emulate* ». Cette technique consiste à laisser tourner la machine virtuelle en mode non privilégié et à générer des interruptions (*trap*) lorsque celle-ci exécute une instruction critique. Le VMM prend alors le contrôle et s'occupe de traiter l'instruction en question en l'appliquant non pas sur le matériel physique, mais sur le matériel « virtuel ».

C'est alors que vous vous demanderez sûrement pourquoi ne pas appliquer le même principe sur une architecture x86 ? Pour répondre à cette question, il faut se tourner vers le modèle d'une architecture virtualisée et vers l'ISA x86.

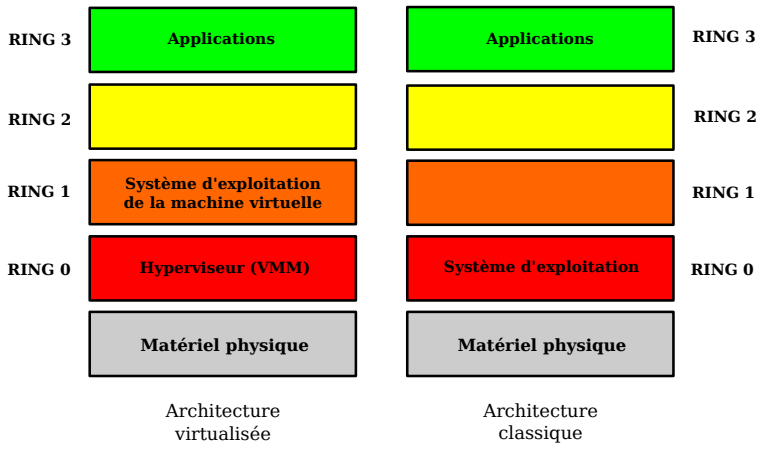


FIGURE 2.21 – Comparaison Virtualisation / Classique

Comme nous pouvons le voir sur la figure 2.21, le VMM s'exécute en ring 0 ce qui ne laisse dans

17. « *Implémentation matérielle du support de la virtualisation Intel VT (p.2)* »

le meilleur des cas que le ring 1 pour le *Guest OS*. C'est ici que nous avons donc un problème, car les instructions x86 ne sont de base pas prévues pour la virtualisation. Il se trouve que dix-sept d'entre elles, qui sont de niveau critique, ne génèrent pas d'interruptions lorsqu'elles sont exécutées en ring 1. Il n'est bien évidemment pas envisageable de les laisser s'exécuter, car elles pourraient accéder et modifier les ressources matérielles, ce qui provoquerait l'effondrement du système tout entier.

Pour pallier à ce problème, les trois méthodes suivantes ont été mises en place.

2.2.4.1 Paravirtualisation [18]

La paravirtualisation est une technique qui offre aux machines virtuelles une interface logicielle quasiment identique au matériel réel. Elle permet d'avoir des VMM beaucoup plus légers et d'atteindre des performances proches du matériel réel avec les machines virtuelles. Par contre, il y a un sacrifice non négligeable pour arriver à cela. Le *Guest OS* doit préalablement être modifié, afin de fonctionner avec un VMM paravirtualisé, ce qui implique l'utilisation d'OS dont il est facile de se procurer le code source.

Il s'agit donc d'une première méthode qui a de grands avantages, mais qui reste compliquée et longue à mettre en œuvre. Elle ne peut évidemment pas être utilisée par les produits VMware, puisqu'ils utilisent le principe de la *full virtualisation* qui s'appuie sur les principes de *Binary Translation* et de *Intel VT & AMD V*.

2.2.4.2 Binary Translation [19]

La « *Binary Translation* »était la seule technique utilisée par VMware jusqu'à la version 3.x d'ESX. Elle est appelée « *Software Technique* »par opposition à la technologie VT-x d'Intel qui est appelée « *Hardware Technique* »et que nous expliquerons dans la prochaine section.

Son principe est assez simple à comprendre. Les instructions x86 de la machine virtuelle vont être traduites avant leur première exécution par un composant « traducteur », qui se trouve dans le VMM, dans un sous-ensemble d'instructions x86. Ainsi, toutes les instructions privilégiées qui agissaient sur le matériel physique et qui nous posaient problème jusque-là, seront remplacées par des instructions non privilégiées n'agissant que sur le matériel virtualisé (FIG 2.22).

Il faut peut-être aussi préciser que ce principe ne s'applique qu'au code du noyau, le code utilisateur est exécuté directement par le processeur. Il est donc évident qu'avec cette méthode, les applications utilisateurs auront de meilleures performances que les applications faisant des appels systèmes. Lorsque la traduction est terminée, le code est placé dans un cache, afin de pouvoir être réutilisé et éviter les *overhead* dus à la *Binary Translation*.

Lorsque la BT est utilisée, ESX utilise le principe de la segmentation (2.16) pour la gestion de la mémoire, ce qui n'est plus utilisé par les OS actuels au profit de la pagination (2.14). Il n'y a donc aucun souci, mais avec l'arrivée des architectures 64 bits, Intel et AMD ont supprimé la segmentation au niveau du matériel, ce qui rend la *Binary Translation* impossible. Il faudrait une implémentation supplémentaire, pour la gestion mémoire, au niveau du VMM, mais l'overhead serait alors considérable et réduirait grandement les performances.

18. http://fr.wikipedia.org/wiki/Paravirtualisation
19. « *http://www.vmware.com/files/pdf/software_hardware_tech_x86_virt.pdf (p.3-4)* », « *http://www.vmware.com/pdf/asplos235_adams.pdf (p.2-5)* »

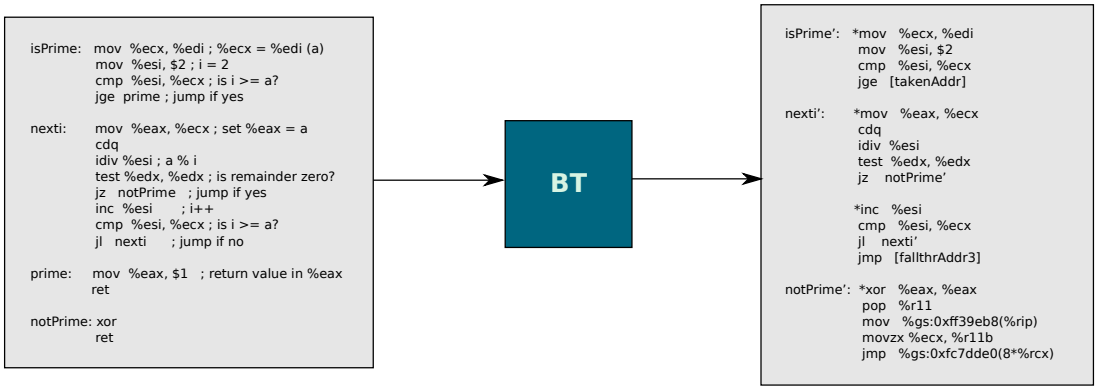


FIGURE 2.22 – Binary Translation

Par la suite, AMD a réimplanté la segmentation dans ses processeurs, mais ce n'est pas le cas d'Intel. Il faut donc prendre garde à l'architecture mise en place avant de vouloir utiliser cette technologie.

2.2.4.3 Intel VT & AMD V [20]

Les deux technologies Intel et AMD sont très similaires, c'est pourquoi, nous ne verrons ici que la première qui est la plus répandue.

Avec Intel VT, des nouvelles instructions, dites de virtualisation, ont vu le jour à savoir (VM Entry, VM Exit, VM Launch, VM Resume et d'autres instructions de contrôle). Les deux premières sont celles qui ont la plus grande importance pour la compréhension du fonctionnement de VT.

Comme nous l'avons vu, un OS est fait pour fonctionner sur un ring 0 et si ce n'est pas le cas, certaines instructions posent un problème. C'est donc à ce niveau que VT apporte une amélioration en faisant tourner les *Guest OS* sur un ring 0. Mais comment est-ce possible si ce ring est déjà utilisé par le VMM ?

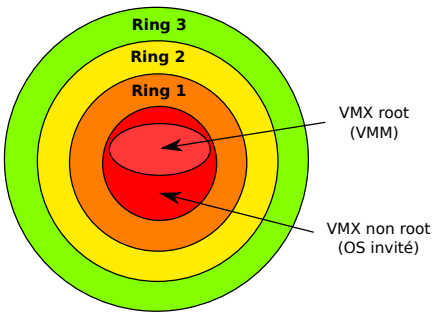


FIGURE 2.23 – Modes VMX d'Intel VT

Intel a introduit un nouveau mode d'exécution appelé VMX (Virtual Machine Extension) dans

20. « *http://www.vmware.com/pdf/asplos235_adams.pdf (p.5-6)* », http://en.wikipedia.org/wiki/Hardware-assisted_virtualization

lequel, il existe deux modes. Le premier est le VMX root et le deuxième le VMX non-root (FIG 2.23). Le VMM s'exécute sur le mode root qui se trouve sur un ring 0 disposant des comportements classiques d'une architecture x86. Il peut donner la main à une machine virtuelle en provoquant un *VM entry* et c'est alors que la VM en question peut exécuter son code librement. Le *Guest OS* s'exécute sur le mode non-root sur un ring 0 et lorsqu'une instruction privilégiée est exécutée, un *VM exit* survient ce qui rend la main au VMM, qui peut alors faire ce qui est nécessaire pour traiter l'exécution de cette instruction en l'appliquant sur le matériel virtualisé. Une fois ceci réalisé, il peut à nouveau rendre la main à la VM en faisant un nouvel *VM entry* (FIG 2.24).

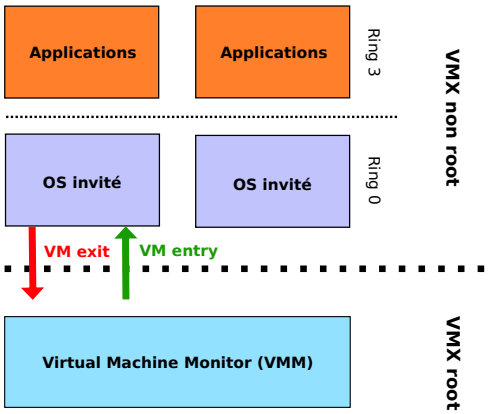


FIGURE 2.24 – VM entry & VM exit

Cette technique peut nous faire penser à la méthode du « trap and emulate »qui était réalisée dans les débuts de la virtualisation. Malheureusement, elle n'est pas parfaite et introduit une quantité non négligeable d'*overheads* avec les nombreux *VM exit* et *VM entry* mais permet finalement de virtualiser sur une architecture x86-64 ce qui était jusque-là un problème.

Vous trouverez une liste des processeurs supportant cette technologie à l'adresse suivante : http://ark.intel.com/VTList.aspx.

2.2.5 Hyper-Threading [21]

L'Hyper-Threading est une technologie propriétaire d'Intel datant de 2002 qui implémente le « Simultaneous Multi Threading (SMT) ». Elle n'est disponible que pour une certaine catégorie de processeurs Intel dont voici la liste :

- Atom
- Core i3
- Core i5
- Core i7
- Itanium
- Pentium 4
- Xeon CPUs

21. http://www.presence-pc.com/tests/intel-core-i5-23179/5/, http://www.01net.com/article/253531.html

Cette technologie permet d'améliorer les performances de calculs dans un microprocesseur en créant pour chaque cœur du processeur physique, deux processeurs virtuels. Ainsi, l'OS aura l'impression d'avoir deux fois plus de processeurs à sa disposition et pourra, par conséquent, exécuter autant de tâches simultanément qu'il y a de processeurs logiques et donc repartir sa charge de travail (FIG 2.25). Petit bémol, pour pouvoir utiliser de façon optimale l'hyper-threading, il faut non seulement que l'OS supporte plusieurs processeurs, mais également que l'OS ait subi une optimisation directement au niveau de son code source, afin de gérer correctement le *multi-threading*. Si ce n'est pas le cas, Intel déconseille l'utilisation de cette technologie qui pourrait amener à des performances négatives. Cette technologie est correctement prise en compte à partir des versions Windows XP et Windows 2003 Server.

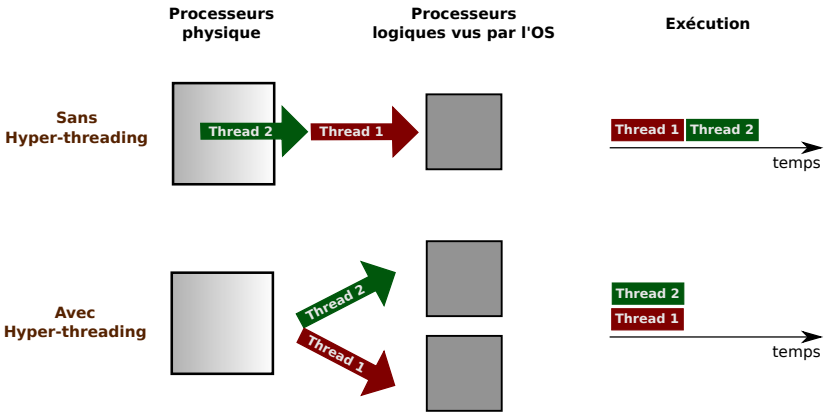


FIGURE 2.25 – Hyper-threading

Au niveau des processeurs virtuels, chacun possède ses propres registres de données et de contrôle ainsi qu'un contrôleur d'interruptions, mais ils partagent le reste des ressources, comme la mémoire cache, l'unité d'exécution et les bus système.

Concrètement, l'implémentation de cette technologie n'utilise que 5% de la surface physique en plus pour un gain en performance pouvant aller jusqu'à 30% selon la nature des applications. Les avantages qui sont apportés sont les suivants :

- Optimisation de l'utilisation du processeur, car les données lui sont envoyées en masse.
- Amélioration de la gestion des codes multi-threadé
- Meilleur temps de réaction et temps de réponse
- Augmentation du nombre d'utilisateurs dans le cas d'un serveur par exemple

2.3 Réseau de stockage [22]

Un réseau de stockage SAN (Storage Area Network) est un réseau dédié permettant de faire du stockage avec un accès à très haut débit.

Le SAN n'est pas à confondre avec un NAS (Network Attached Storage) qui est une ressource de stockage directement connectée à un réseau ethernet et qui peut supporter plusieurs systèmes de fichiers réseau différents, comme CIFS (Common Internet File System) de Microsoft, NFS (Network File

22. http://en.wikipedia.org/wiki/Storage_area_network

System) de UNIX ou encore AFP (AppleShare File Protocol) de Apple. Une fois connecté, il apparaîtra sur les différentes machines du réseau comme un serveur de fichiers partagés.

Dans le cas d'un SAN, les machines du réseau voient les espaces disques des baies de stockage comme s'ils étaient leurs propres disques durs. Elles ont donc accès à cet espace en mode block, un peu comme cela serait le cas avec un disque dur ATA ou SCSI.

Un serveur accède à l'espace disque de stockage à travers des HBA (Host Bus Adapter) en utilisant un protocole comme SCSI ou Fibre Channel. Dans le cadre du CTI et comme dans la plupart des cas, c'est le Fibre Channel qui est utilisé.

Les points forts du SAN par rapport au NAS sont :

- La scalabilité de l'espace de stockage. Si un espace disque devient insuffisant, il suffit simplement d'ajouter un disque ou alors directement un baie.
- Il y a entre les serveurs et les baies de stockage un mécanisme d'accusé de réception permettant de s'assurer que les transferts se sont réalisés correctement.
- La redondance peut être assurée en dupliquant les baies de stockage.
- Le SAN fonctionne dans un environnement hétérogène.

2.3.1 Host Bus Adapter [23]

Un *Host Bus Adapter* (HBA) ou contrôleur de bus en français, est une carte d'extension qui est utilisée pour connecter un serveur ou un poste client à un bus externe, et dans notre cas, à un réseau de stockage. On retrouve plusieurs types différents de HBA dont notamment des HBA SCSI et HBA Fibre Channel mais il existe également des HBA Ethernet, USB ou encore Firewire.

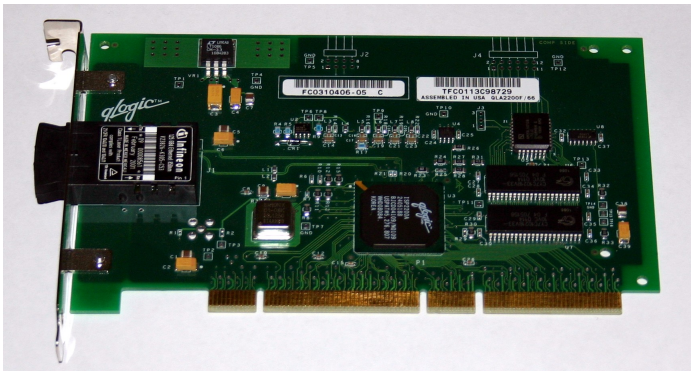


FIGURE 2.26 – Host Bus Adapter Fibre Channel

Celui qui nous intéresse plus particulièrement est celui utilisé pour la connexion au réseau SAN et qui supporte le protocole Fibre Channel. Il dispose d'une adresse unique un peu comme l'adresse MAC pour Ethernet. Cet identifiant unique, appelé World Wide Name (WWN), est codé sur 8 octets dont les 3 premiers sont attribués par l'IEEE et les autres par le constructeur de l'équipement.

23. http://fr.wikipedia.org/wiki/Contr%C3%B4leur_h%C3%B4te_de_bus

```
1    50:00:51:e3:63:a4:49:01
```

Exemple de WWN

2.3.2 Fibre Channel [24]

Fibre Channel est le protocole standard utilisé dans les réseaux SAN, car il permet une connexion haut débit entre le serveur et son espace de stockage. Son nom peut porter à confusion et nous faire croire qu'il utilise de la fibre optique. Il s'agit, en fait, d'un protocole fonctionnant en série qui ne nécessite que deux conducteurs physiques et qui peut, par conséquent, utiliser des supports comme de la paire torsadée, du câble coaxial ou de la fibre optique.

Ce protocole supporte actuellement quatre débits différents (1 Gbit/sec, 2 Gbit/sec 4 Gbit/sec et 8 Gbit/sec) et il utilise trois topologies différentes à savoir :

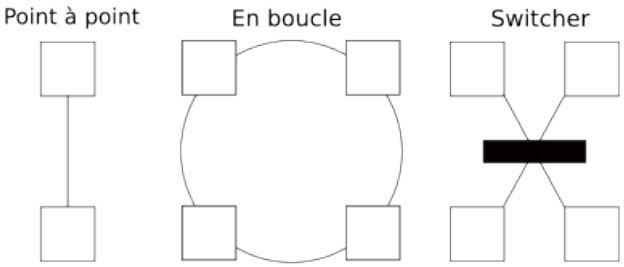


FIGURE 2.27 – Topologies Fibre Channel

1. **Point à point :** Liaison directe entre deux périphériques.
2. **En boucle :** Chaque périphérique est relié à deux autres, afin de former une boucle.
3. **Commuté :** Un switch Fibre Channel est utilisé pour interconnecter les périphériques entre eux et ainsi, permettre d'avoir de la redondance sur les chemins possibles. Dans ce cas, un switch est appelé *Fabric*, comme nous pouvons le voir sur le schéma réseau du CTI (FIG 3.2) avec F/A (*Fabric A*) et F/B (*Fabric B*).

2.3.3 LUN [25]

Un LUN pour *Logical Unit Number* est un identifiant d'unité logique qui est, en fait, un pointeur vers un espace disque. Il est en général défini par sa taille en GO et par les WWN des HBA des serveurs qui y ont accès.

Un administrateur de réseau SAN peut effectuer du *LUN Masking* [26], ce qui consiste à restreindre l'accès à un LUN, afin qu'il ne soit visible que de certains serveurs.

Le nombre de LUNs qu'un serveur peut gérer dépend fortement du système d'exploitation.

24. http://en.wikipedia.org/wiki/Fibre_Channel
25. http://fr.wikipedia.org/wiki/LUN
26. http://www.sansecurity.com/faq/lun-masking.shtml

Système d'exploitation	Nombres de LUNs
AIX	64000
HP	16000
Linux	256
IBM	64000
Windows	256
ESX •	256^3

FIGURE 2.28 – Nombre de LUNs par hôte

2.3.4 Raw Device Mapping

Lorsqu'on décide d'utiliser un LUN avec un ESX/ESXi, par défaut ce LUN va être formaté en VMFS et certaines fois, nous voudrions utiliser un autre système de fichiers. La solution est le « Raw Device Mapping »qui permet d'utiliser un LUN directement depuis une VM en (lecture/écriture).

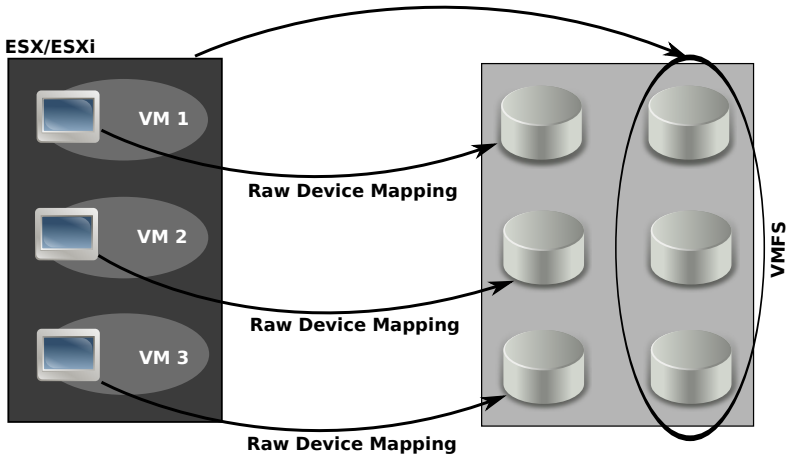


FIGURE 2.29 – Raw Device Mapping

Le lien se fait grâce à un fichier de *mapping* se trouvant sur le *datatstore* VMFS qui pointe sur le LUN en question. Ce LUN peut donc être ajouté depuis la configuration de la VM en mode « Raw Device Mapping ».

Ce mécanisme a été utilisé lors du stage au CTI pour les LUNs *Command Device* servant au « True Copy »sur les baies Hitachi (7.3).

2.4 Performances

2.4.1 SLA [27]

Le Service Level Agreement (SLA) est un document qui définit la qualité de service requise entre un prestataire et un client.

27. http://fr.wikipedia.org/wiki/Service_Level_Agreement

Dans le cadre des performances au sein même d'une entreprise, il définit la qualité minimale attendue pour un service spécifique et pour les performances d'un système informatique physique ou vrituel. C'est donc ce document qui servira de base pour la configuration des systèmes VMware dans une infrastructure de virtualisation.

3 Matériel à disposition

Avant de débuter, il est nécessaire de faire le tour du matériel utilisé dans les différents cadres de travail durant ce projet de diplôme.

3.1 HEPIA

Le laboratoire de transmission de données possède son intranet se trouvant à l'intérieur du réseau HEPIA. Le nom de domaine « tdeig.ch »est géré par un serveur DNS se trouvant dans la DMZ. Il y a également un serveur web Apache (hébergeant le site du labo http://www.tdeig.ch) qui se trouve dans la DMZ et qui est donc accessible depuis l'extérieur (voir schéma 3.1).

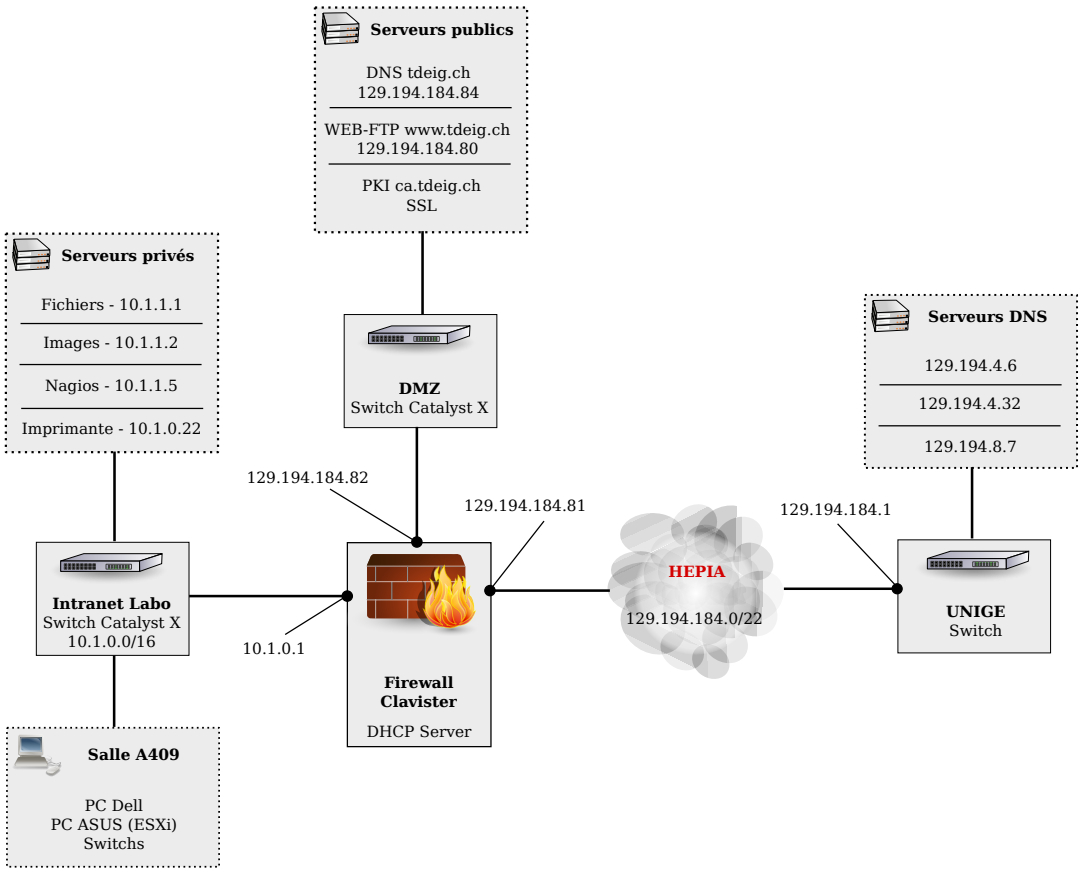


FIGURE 3.1 – Réseau du Labo TD

Le reste des serveurs, se trouvant au laboratoire, sont privés et par conséquent accessibles uniquement en local. C'est le cas du serveur sur lequel tourne l'application Nagios qui a été montée dans le cadre de ce projet de diplôme.

Les machines utilisées [1] pour la virtualisation avec VMware disposent de la configuration hardware suivante :

- Carte mère [2] : Gigabyte GA-G33M-S2L
- Processeur : Intel Core2Duo 3Ghz, 6Mb de cache et FSB de 1333Mhz
- Mémoire : 4GB DDR800

Sur ces machines est installée la **version 4 ESXi build 208167**.

1. http://www.tdeig.ch/vmware/Montage_PC_Gigabyte.pdf
2. http://www.tdeig.ch/vmware/motherboard.pdf

3.2 CTI

L'infrastructure dont dispose le CTI pour la virtualisation est décrite de manière globale sur l'image 3.2.

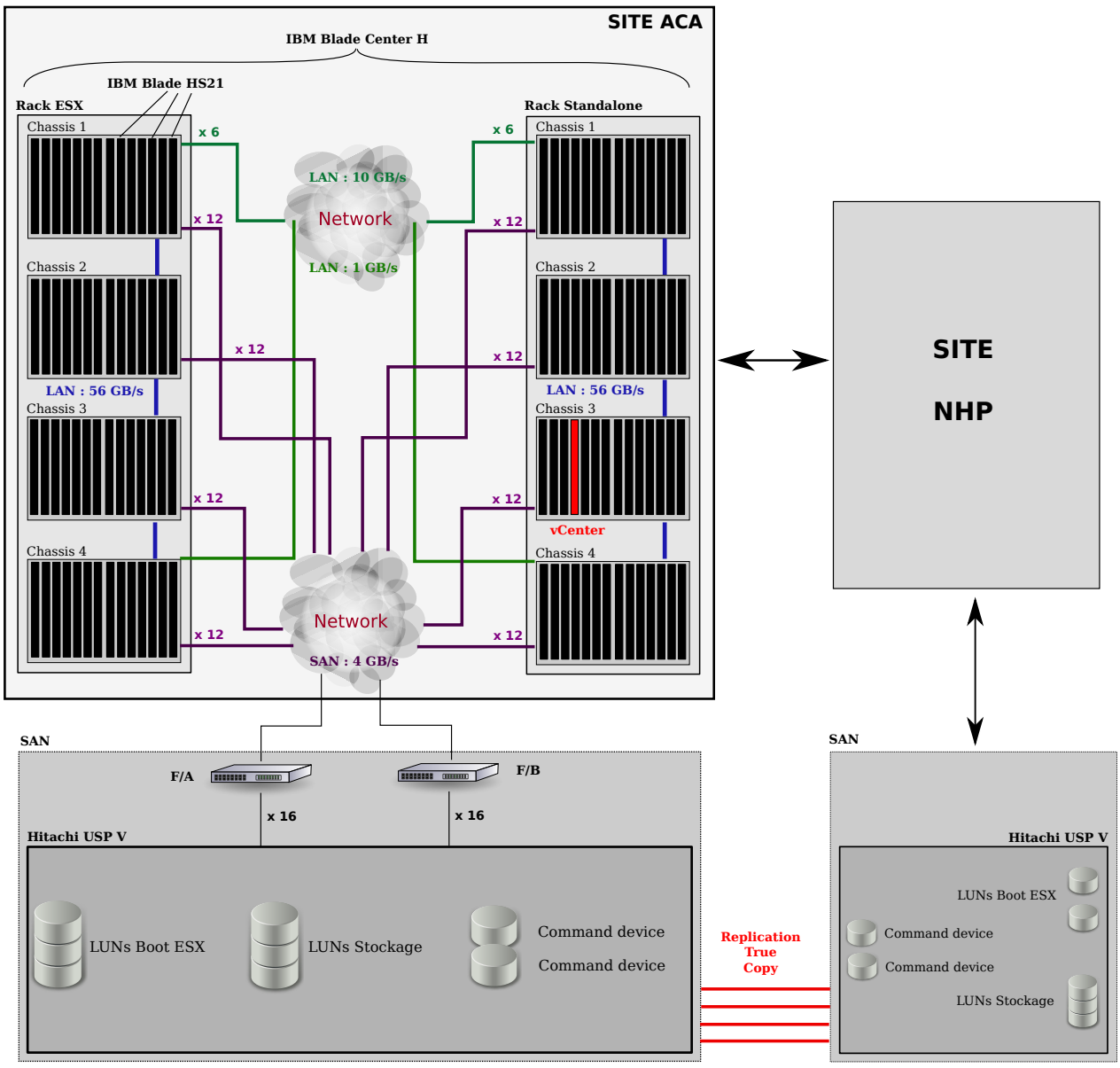


FIGURE 3.2 – Réseau de virtualisation CTI

Un *rack* IBM, que ce soit le *rack* contenant les ESX ou un autre, contient en tout quatre châssis qui possèdent chacun quatorze *blades* (FIG 3.3) avec la configuration matérielle suivante :

- **Modèle :** IBM eServer BladeCenter HS21
- **Processeur :** Intel(R) Xeon(R) CPU E5430 @ 2.66GHz
- **Mémoire :** 32GB
- **Nb Sockets :** 2

- **Nb Cores :** 8
- **Nb NICs :** 6
- **Nb HBA :** 5

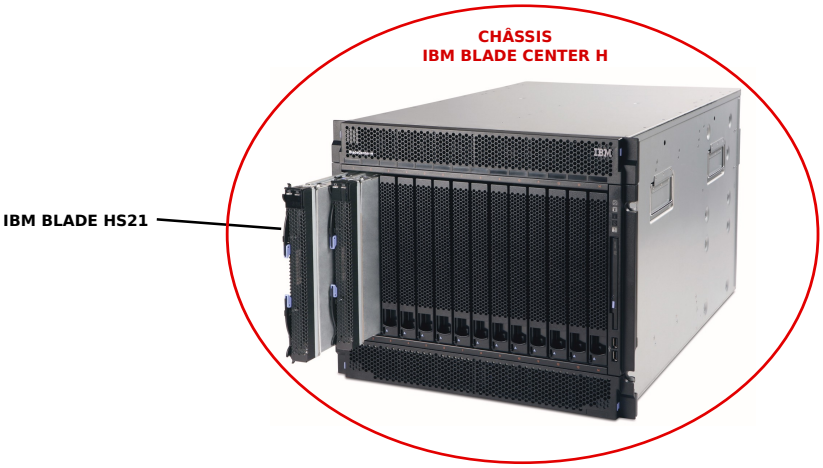


FIGURE 3.3 – Châssis IBM avec quatorze *blades*

Chaque ESX (**VMware ESX 4.0.0 build-244038**) dispose donc d'une blade entière et contient en moyenne six machines virtuelles.

Au niveau des I/O, chaque châssis possède **6 switchs Ethernet**, **2 switchs SAN** et **2 modules de management** (FIG 3.4)

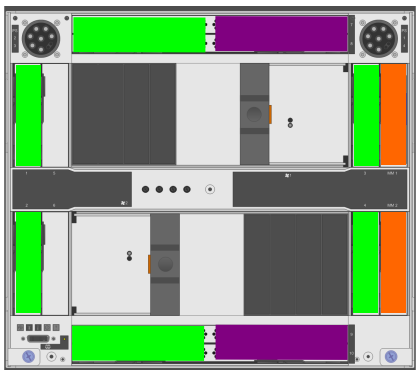


FIGURE 3.4 – I/O du Châssis IBM - Vue arrière

Une blade dispose de six ports Ethernet et de deux ports HBA FC, elle est donc reliée à chacun des switchs, ce qui permet d'améliorer les performances réseau grâce à cette redondance.

3.2.1 Réseaux

Les interfaces réseau d'un ESX sont configurées pour permettre à n'importe quel réseau virtuel de les utiliser. Ainsi, nous obtenons de la redondance grâce aux six ports disponibles (FIG 3.5).

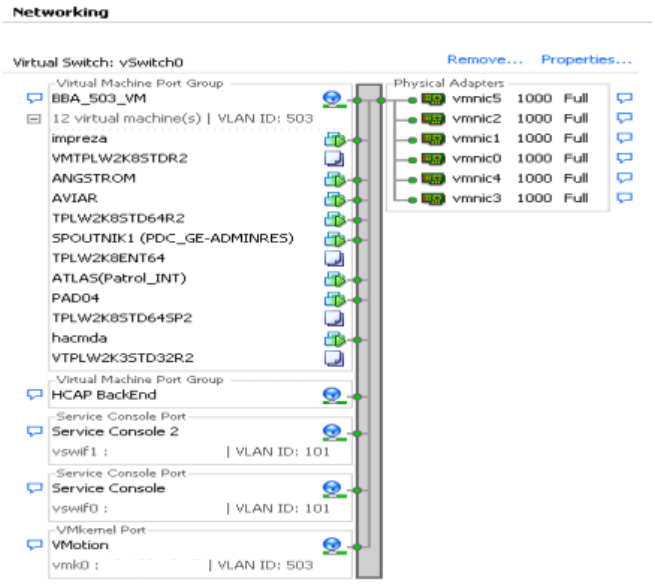


FIGURE 3.5 – Configuration réseau ESX

Il faut dès lors définir des VLANs, si une séparation des réseaux désire être réalisée. Cette séparation est dite « logique »puisqu'au niveau physique tout le trafic passe par les mêmes chemins.

Les ports des switchs CISCO doivent par conséquent être configurés en mode « trunk », afin de permettre à chaque VLAN d'utiliser tous les ports disponibles. Nous obtenons donc 2 réseaux physiques qui sont les réseaux LAN et SAN, mais 3 réseaux logiques :

1. Réseau applicatif (VLAN 503)
2. Réseau administratif (VLAN 101)
3. Réseau de stockage (SAN)

Sur le schéma (FIG 3.6), nous n'avons représenté que deux *blades*, mais en réalité, il y en a quatorze comme déjà expliqué. Nous pouvons voir que chaque *blade* possède une connexion physique avec chacun des switchs du châssis.

Il faut peut-être encore préciser qu'au niveau du réseau LAN du CTI, il existe plusieurs switchs sur lesquels sont connectés les châssis, mais nous n'en avons représenté qu'un seul ici pour une raison de simplicité.

Au niveau du réseau SAN, il y a également beaucoup de redondances sur les chemins d'accès à une baie de stockage pour assurer la communication. C'est ce que nous avons défini comme étant un réseau commuté dans la terminologie SAN (2.3.2).

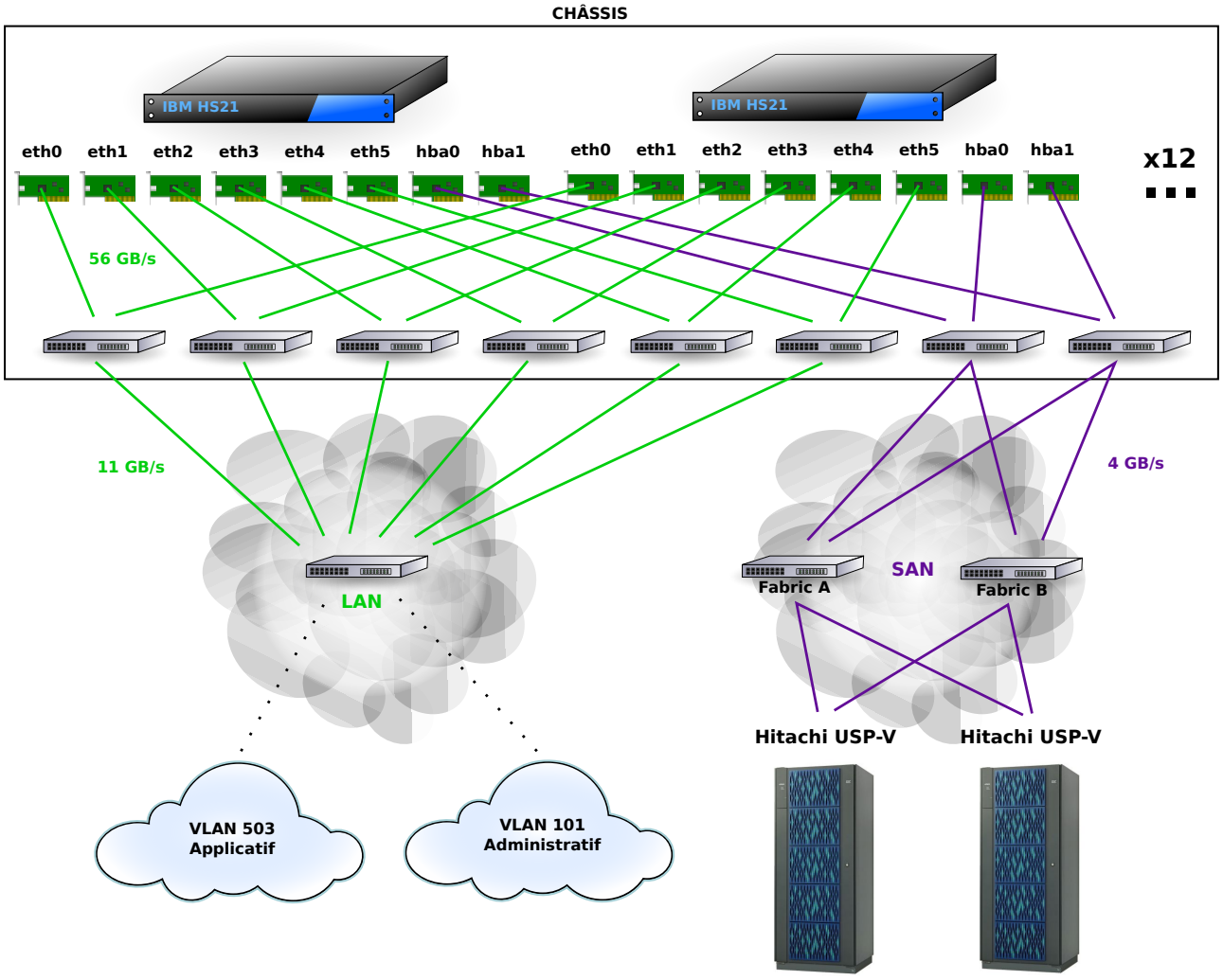


FIGURE 3.6 – Réseaux

3.2.2 Stockage

Une des particularités du stockage dans l'infrastructure de virtualisation du CTI est qu'aucune des machines ESX n'a d'espace de stockage local. Tous les fichiers se trouvent sur des baies de stockage Hitachi USP-V (Universal Storage Platform V) dont la *datasheet*[3] est disponible sur le site officiel d'Hitachi. Ces baies sont les derniers modèles proposés par cette marque et proposent des performances de haute qualité.

FIGURE 3.7 – Hitachi USP-V

Pour résumer les points importants de ces baies :

- Bande passante max. de 4 ou 8 GB/s.
- Bande passante interne max. de 106 GB/s.
- Espace de stockage total max. de 247 PB avec des disques de type SATA II, Fibre Channel ou Flash.
- 152 disques maximum.
- Jusqu'à 65,536 LUNs par volume.
- 1 à 14 switchs Fibre Channel avec chacun 8 ou 16 ports.

Nous retrouvons donc sur ces baies de stockage un certain nombre de LUNs, qui sont utilisés par les blades et par conséquent par les ESX (FIG 3.8). Parmi ces LUNs, nous avons :

- 1 LUN de boot par ESX, d'une capacité de 15 GB.
- 6 LUNs de données par Cluster, d'une capacité de 1 TB.

Les LUNs de boot sont celles qui contiennent l'installation des ESX, avec un système de fichiers ext3 et une partition vmfs3 pour le service console associé.

Les LUNs de données contiennent tous les fichiers servant au fonctionnement des machines virtuelles (vmx, vmdk, vswp etc...), ainsi que toutes les données utilisées par celles-ci.

La visibilité des LUNs par les ESX est définie par le mécanisme de *masking*, dont nous avons parlé en 2.3.3.

3. http://www.hds.com/assets/pdf/hitachi-datasheet-usp-v.pdf

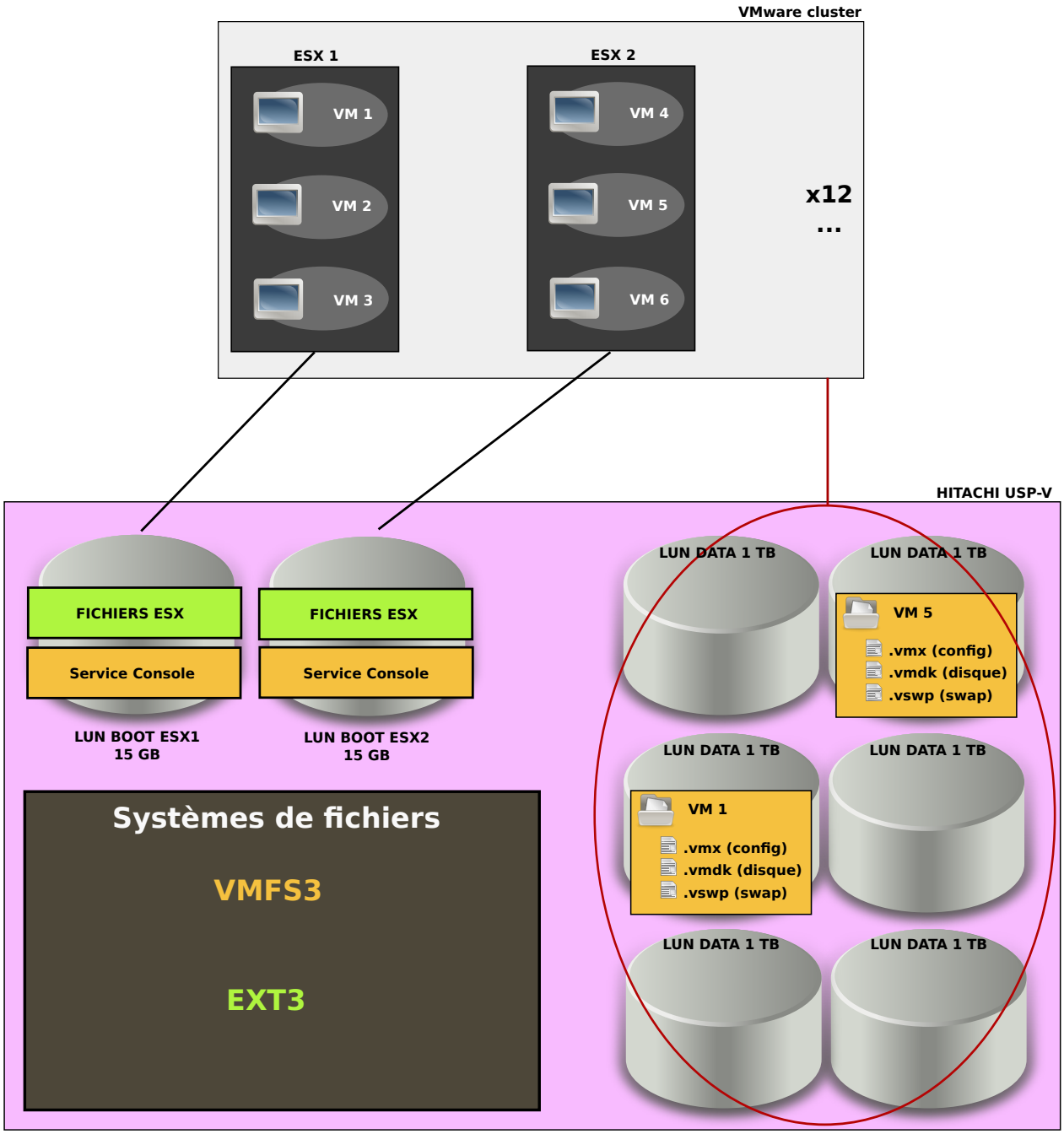


FIGURE 3.8 – Stockage Hitachi USP-V

4 Outils utilisés

4.1 VMware vCenter Server

4.1.1 Introduction

vCenter est un service qui permet de centraliser l'administration d'hôtes ESX/ESXi. Il permet de manager les hôtes, ainsi, que les machines virtuelles qui tournent sur ceux-ci. Il y a une possibilité d'avoir plusieurs vCenter et les regrouper. Nous pouvons, avec cette méthode, se connecter sur un seul vCenter qui verra et managera tous les autres vCenter et leurs ressources associées.

4.1.2 Pré-requis hardware minimum

« vCenter Server »peut être installé soit sur une machine physique, soit sur une machine virtuelle, pourvu qu'il ait accès à une base de données compatible. Il faut respecter les exigences minimales suivantes :

- 2 CPUs Intel ou AMD avec au minimum une fréquence de 2.0Ghz, plus si la base de données est locale.
- 3GB de RAM, plus si la base de données est locale. Il faut également compter un supplément de 128MB à 1.5GB pour le VMware vCenter Management Webservices qui est un service s'exécutant en parallèle.
- Un espace de stockage d'au minimum 2GB, plus si la base de données est locale.
- Microsoft SQL Server 2005 Express au minimum. D'autres bases de données SQL sont également envisageable (Oracle par exemple).
- Une connexion gigabit est conseillée.

4.1.3 Recommandations

Pour des performances optimales, il faut respecter les suggestions suivantes proposées par VMware :

Jusqu'à 50 ESX/ESXi et 250 VMs

Produit	CPUs	Mémoire	Disque
vCenter Server	2	4GB	3GB

Jusqu'à 200 ESX/ESXi et 2000 VMs

Produit	CPUs	Mémoire	Disque
vCenter Server	4	4GB	3GB

Jusqu'à 300 ESX/ESXi et 3000 VMs

Produit	CPUs	Mémoire	Disque
vCenter Server	4	8GB	3GB

4.1.4 Ports utilisés

- **80** : Utilisé pour accéder à l'interface web de vCenter. Par défaut redirigé sur le port HTTPS 443.
- **389** : Utilisé par LDAP pour la gestion des groupes de vCenter.
- **443** : Utilisé par défaut pour la communication entre vCenter et vSphere Client. Également utilisé pour l'accès à l'interface web, depuis un navigateur, et pour la connexion à un SDK, par exemple.
- **636** : Utilisé par SSL pour les vCenter utilisant la notion de groupe.
- **902/903** : Utilisé par défaut pour la communication avec les hôtes ESX/ESXi. Également utilisé entre vCenter et vSphere Client pour l'affichage de la console des VMs.
- **8080** : Utilisé par les Webservices en HTTP
- **8443** : Utilisé par les Webservices en HTTPS

4.1.5 Installation [1]

Nous allons voir ici l'installation réalisée au CTI, mais avant de débuter, il faut s'assurer de respecter les points suivants :

1. S'assurer d'avoir la configuration hardware minimale décrite en 4.1.2.
2. Disposer d'une base de données compatible et fonctionnelle.
3. Il ne doit pas y avoir de NAT [2] entre le serveur vCenter et les hôtes qu'il contrôle.
4. Le serveur vCenter doit appartenir à un domaine plutôt qu'à un groupe de travail, sans quoi, il ne sera pas capable de détecter automatiquement tous les domaines.
5. Le nom de la machine exécutant vCenter ne doit pas dépasser 15 caractères.
6. Assigner une IP statique au vCenter ainsi qu'un DNS [3] valide.

Dans notre cas, nous avons la configuration suivante :

- **OS :** Windows Server 2008 64bits
- **Modèle :** IBM eServer BladeCenter HS21
- **Processeur :** Intel(R) Xeon(R) CPU E5430 @ 2.66GHz
- **Nb CPU :** 2
- **Nb Cores :** 8
- **Mémoire :** 32GB
- **Espace disque :** 100GB

Une fois tout ceci vérifié, l'installation peut débuter en lançant *autorun.exe* contenu dans les fichiers d'installation de VMware vCenter Server.

Plusieurs composants seront alors installés, afin que tout fonctionne correctement :

1. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_esx_vc_installation_guide.pdf* »
2. Network Address Translation
3. Domain Name System

- **VMware vCenter Server** Un service Windows, qui permet de manager les différents serveurs ESX/ESXi.

- **Microsoft.NET 3.0 SP1 Framework** Utilitaire pour vSphere Client et quelques fonctionnalités liées à la base de données.

- **VMware vCenter Orchestrator** Module qui fournit plusieurs outils au vCenter, afin d'améliorer le management des systèmes virtualisés.

- **Microsoft SQL Server 2005 Express** Cette base de données est optionnelle, car, comme nous l'avons déjà précisé, nous pouvons en utiliser une autre, comme Oracle.

Nous trouverons également d'autres outils dans l'installation de vCenter comme :

- **vCenter converter** Permet de virtualiser des machines physiques ou de convertir des machines virtuelles fonctionnant sous une version d'ESX/ESXi vers une autre version.

- **vCenter Update Manager** Permet de mettre à jour la sécurité et les différents patchs des ESX/ESXi et de leurs machines virtuelles.

Lors de l'installation, plusieurs informations vous seront demandées. Pour la plupart, il faudra laisser la valeur par défaut.

1. Le nom d'utilisateur et l'organisation : Mettre des informations cohérentes avec le reste de l'infrastructure.
2. La licence vCenter : Entrez la clé que vous avez reçue de la part de VMware.
3. Dossier d'installation vCenter : Par défaut.
4. Standalone ou groupe : Savoir si le vCenter fait parti d'un groupe, afin de réaliser une hiérarchie de management, ou s'il est tout seul. Nous l'installerons en Standalone.
5. FQDN [4] du vCenter gérant le groupe : Aucun dans notre cas.
6. Port LDAP : Par défaut, mais utile que si le vCenter fait parti d'un groupe.
7. Infos de la base de données : Aucune info car nous utilisons la base de données par défaut.
8. Différents ports utilisés par vCenter : Nous avons laissé tous les ports par défaut.

4.2 VMware vSphere Client

4.2.1 Introduction

« vSphere Client »s'installe sur une machine Windows et est un client qui permet de se connecter à un vCenter Server ou directement à un ESX/ESXi afin de les administrer grâce à une interface graphique.

4.2.2 Pré-requis hardware minimum

« vSphere Client »peut être installé soit sur une machine physique soit sur une machine virtuelle. Il faut respecter les exigences minimales suivantes :

4. Full Qualified Domain Name

- 1 CPU Intel ou AMD avec au minimum une fréquence de 500Mhz.
- 200MB de RAM.
- Un espace de stockage d'au minimum 1GB.
- Une connexion gigabit est conseillée.

4.2.3 Recommandations

Pour des performances optimales, il faut respecter les suggestions suivantes proposées par VMware :

Jusqu'à 50 ESX/ESXi et 250 VMs

Produit	CPU	Mémoire	Disque
vSphere Client	1	200MB	1GB

Jusqu'à 200 ESX/ESXi et 2000 VMs

Produit	CPU	Mémoire	Disque
vSphere Client	1	500MB	1GB

Jusqu'à 300 ESX/ESXi et 3000 VMs

Produit	CPU	Mémoire	Disque
vSphere Client	1	500MB	1GB

4.2.4 Installation

L'installation se fait tout simplement à partir d'*autorun.exe* qui a servi à installer vCenter. Nous avons laissé toutes les options par défaut.

Attention : « Host Update Utility »n'est à installer que si l'architecture virtualisée n'utilise pas de clusters. Autrement, il faut privilégier vCenter Update Manager, pour la gestion des mises à jour.

4.3 Esxtop + perfmon

Esxtop est l'outil par excellence à utiliser pour monitorer les performances d'un ESXi. Nous pouvons le coupler à perfmon (Windows), pour un rendu graphique des données.

Comment utiliser esxtop ?
« *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf (p.159-171)* »

Les compteurs utiles au niveau du CPU sont %RDY et %USED
http://communities.vmware.com/docs/DOC-5240

Comment interpréter les compteurs esxtop ?
http://communities.vmware.com/docs/DOC-9279

Utilisation de perfmon

http://www.zebulon.fr/astuces/221-optimiser-vista-avec-le-rapport-de-performances.html
http://communities.vmware.com/docs/DOC-5100

4.4 Nagios

Le monitoring d'un réseau est une tâche assez complexe et parfois compliquée à mettre en œuvre, surtout sur de grandes infrastructures. Il existe, bien évidemment, plusieurs palettes d'outils permettant de le faire et grâce auxquels nous pouvons, par exemple, connaître l'état d'une machine (ping), effectuer une requête HTTP, pour savoir si un serveur web fonctionne, ou encore, visualiser les propriétés d'un disque, afin de savoir la place libre restante.

Mais le problème est que ces actions sont assez simples à réaliser, lorsque le réseau est de petite taille et qu'il n'y a pas beaucoup de services, alors que, dans le cas contraire, cela devient rapidement ingérable. C'est pourquoi Nagios est utilisé dans le domaine professionnel afin de monitorer les performances d'un réseau.

Avec son aide, les tâches fastidieuses de monitoring réseau sont faites automatiquement et périodiquement. Il est possible d'assigner autant de tâches que nous le souhaitons à un composant du réseau. Il les exécutera et nous signalera le moindre problème (par e-mail), selon des critères fixés.

Une fois maîtrisé, il devient vite indispensable pour n'importe quel ingénieur réseau.

4.4.1 Installation

L'installation réalisée dans le cadre du labo de transmission de données a été faite sur un Ubuntu Server 10.04 et, par conséquent, Nagios se trouve dans les dépôts officiels, ce qui facilite la procédure, puisqu'il faut simplement installer le paquet **nagios3** .

```
sudo apt-get install nagios3
```

Il devrait également installer Apache2, si ce n'est pas déjà fait. Pour terminer, un mot de passe pour le compte « nagiosadmin »vous sera demandé. Il s'agit du compte principal pour la gestion de Nagios.

Pour une installation manuelle vous pouvez suivre le tutoriel officiel à l'adresse suivante : http://nagios.sourceforge.net/docs/3_0/quickstart-ubuntu.html

Une fois l'installation terminée, la page principale de Nagios devrait être accessible via le lien http://localhost/nagios3/

4.4.2 Configuration

4.4.2.1 Fichier principal

La configuration principale de Nagios se trouve dans */etc/nagios3/nagios.cfg* . Ce qui nous intéresse plus particulièrement, dans ce fichier, est le chemin vers les fichiers de configuration pour les services, hôtes, contacts etc... Il faut donc le préciser sur la ligne suivante :

```
cfg_dir=/etc/nagios3/conf.d/
```

Extrait de nagios.cfg

Dans notre cas, tous nos fichiers de configurations seront dans le répertoire */etc/nagios3/conf.d/* .

Afin de valider une configuration, Nagios nous met à disposition la commande suivante qui nous permet de détecter les incohérences :

sudo nagios -v /etc/nagios3/nagios.cfg

4.4.2.2 Les utilisateurs

Pour ajouter un utilisateur à Nagios, il suffit de taper la commande suivante :

sudo htpasswd -c /etc/nagios3/htpasswd.users <username>

Ensuite, les permissions de cet utilisateur peuvent être modifiées dans le fichier */etc/nagios3/cgi.cfg* en ajoutant le nom de cet utilisateur aux lignes commençant par « **authorized_for** ». Elles sont au nombre de sept et correspondent aux autorisations suivantes :

authorized_for_system_information	voir l'état des services
authorized_for_configuration_information	voir la configuration du serveur
authorized_for_system_commands	exécuter des commandes systèmes
authorized_for_all_services	voir l'état de tous les services
authorized_for_all_hosts	voir l'état de tous les hôtes
authorized_for_all_service_commands	exécuter des commandes pour tous les services
authorized_for_all_host_commands	exécuter des commandes pour tous les hôtes

FIGURE 4.1 – Source : http ://doc.ubuntu-fr.org/nagios

4.4.2.3 Les périodes de temps

Les périodes de temps servent à définir quand est-ce qu'un service doit être surveillé. Ainsi, on peut, par exemple, dire à Nagios de ne contrôler l'accès Web qu'aux heures de travail.

Pour créer une nouvelle période de temps, il faut ajouter ou éditer, par exemple, un fichier */etc/nagios3/conf.d/timeperiods.cfg* qui contiendra des heures de travail :

```
define timeperiod{
        timeperiod_name heures_travail
        alias           Heures de travail
        monday          07:00-17:00
        tuesday         07:00-17:00
        wednesday       07:00-17:00
```

```
7       thursday        07:00-17:00
        saturday        07:00-17:00
9       }
```

Extrait de timeperiods.cfg

4.4.2.4 Les contacts

Un contact est une personne physique que Nagios pourra contacter, pour lui signaler les incidents éventuels.

Pour créer un contact, il faut ajouter ou éditer, par exemple, un fichier */etc/nagios3/conf.d/contacts.cfg* comme ceci :

```
1  define contact{
       contact_name                    dupont
3      alias                           Henry Dupont
       service_notification_period     heures_travail
5      host_notification_period        heures_travail
       service_notification_options    w,u,c,r
7      host_notification_options       d,u,r
       service_notification_commands   notify-service-by-email
9      host_notification_commands      notify-host-by-email
       email                           henry.dupont@domaine.net
11     pager                           +32336254324
       }
```

Extrait de contacts.cfg

Descriptif des différents champs :

- **contact_name** : Pseudo du contact qui sera utilisé dans les autres fichiers de configuration pour lui faire référence.
- **alias** : Description du contact, souvent utilisé pour entrer le nom et le prénom.
- **notification_period** : Périodes durant lesquelles le contact sera averti pour les services et les hôtes. Reprendre les périodes de temps de *timeperiods.cfg*
- **notification_options** : Quand est-ce qu'une notification doit être envoyée. Pour les services les options possibles sont (w : warning, u : unknown, c : critical, r : recovery) et pour les hôtes (d : down, u : unreachable, r : recovery).
- **notification_commands** : La méthode d'envoi des notifications.
- **email et pager** : Les infos pour l'envoi des notifications soit par e-mail soit par sms.

4.4.2.5 Les groupes de contacts

Il est également possible de regrouper les contacts et d'avertir, par exemple, un groupe complet au lieu de s'amuser à paramétrer pour chaque contact.

Un nouveau groupe peut être défini dans un fichier */etc/nagios3/conf.d/contacts_group.cfg* comme ceci :

```
define contactgroup{
        contactgroup_name     admins
        alias                 Administrateurs du reseau
        members               dupont, smith, toto
        }
```

Extrait de contacts_group.cfg

Il suffit simplement de préciser pour le paramètre « members »quels sont les contacts qui font partis de ce groupe.

4.4.2.6 Les hôtes

Les hôtes sont les équipements du réseau qui seront monitorés comme les postes fixes, les routeurs, les serveurs etc...

Pour plus de clarté et pour faciliter la maintenance, nous pouvons spécifier un hôte par fichier. Par exemple, si nous avons une machine sur laquelle tourne un serveur web, on pourra créer le fichier */etc/nagios3/conf.d/serveur_web.cfg* :

```
define host{
        use                   generic-host
        host_name             webserver
        alias                 Serveur Web
        address               192.168.x.x
        check_command         check-host-alive
        }
```

Extrait de serveur_web.cfg

Avec ce fichier, le serveur web sera *pingé* périodiquement grâce à la commande « check-host-alive ».

4.4.2.7 Les groupes d'hôtes

Pareil que pour les contacts, nous pouvons définir des groupes d'hôtes pour, par exemple, limiter des zones du réseau comme une DMZ.

Il faudra donc créer un fichier */etc/nagios3/conf.d/hostgroups.cfg* dans lequel nous pourrons placer un groupe DMZ :

```
define hostgroup{
        hostgroup_name dmz
        alias          Serveurs de la DMZ
        contact_groups admins
        members        webserver, mailserver
        }
```

Extrait de hostgroups.cfg

4.4.2.8 Les services

Les services sont le dernier point important de la configuration de Nagios. Comme nous l'avons vu, un hôte est *pingé* pour tester sa disponibilité mais rien de plus. Ces services permettront de monitorer un hôte sous différents aspects, par exemple, tester un port, vérifier le hardware, récupérer sa configuration etc...

Si nous désirons tester le port 80 du serveur web en effectuant une requête HTTP, nous pourrons créer un fichier ***/etc/nagios3/conf.d/services_web.cfg*** :

```
define service{
        use                 generic-service
        host_name           webserver
        service_description Test HTTP
        check_command       check_http
        }
```

Extrait de services_web.cfg

5 Analyse & optimisation des performances

Dans cette partie du document, nous allons voir en détail quelles sont les bonnes pratiques à respecter pour des performances optimales, sur des systèmes VMware ESX/ESXi. Ensuite, nous verrons une méthodologie permettant d'identifier différentes sources de problèmes, puis, dans la mesure du possible, quelles sont les manipulations à effectuer pour les régler ou alors des pistes à explorer pour les résoudre.

Nous débuterons avec un aspect général pour ensuite aller plus en profondeur avec des problèmes liés directement au hardware (CPU, RAM, Stockage et Réseau).

5.1 Performances générales

Avant de débuter avec l'analyse des performances du CPU ou bien de la RAM, il faut prendre en compte quelques aspects généraux, qui peuvent également influencer sur les performances du système virtualisé. Il y a donc quelques points à respecter et à surveiller, afin que la base de notre système soit optimisée au maximum.

Nous allons également voir quels sont les outils qui peuvent nous permettre de surveiller de façon globale le système et de détecter les éventuels problèmes.

5.1.1 Les bonnes pratiques

Les différents points globaux à respecter et à contrôler sur tous systèmes ESX/ESXi, sont les suivants :

1. **Valider le hardware** de la machine hébergeant ESX/ESXi en s'assurant qu'il respecte la configuration minimale conseillée par VMware [1] dont notamment pour *ESX/ESXi 4* :
   - **Processeur 64-bits**
   - **2GB de RAM minimum**

	ESX 1.0-2.5	**ESX 3.0**	**ESX 3.5**	**ESX 4.0**
VMkernel	32-bit	32-bit	32-bit	64-bit
Virtual CPU	32-bit	32-bit & 64-bit	32-bit & 64-bit	32-bit & 64-bit

FIGURE 5.1 – Pré-requis des CPUs physiques

1. « *http://www.vmware.com/resources/compatibility/pdf/vi_systems_guide.pdf* », « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_esx_vc_installation_guide.pdf (p.13-20)* »

2. S'assurer que le hardware fonctionne correctement en faisant **tourner un benchmark**, afin d'éliminer toutes les possibilités de défauts.

3. Vérifier que le **« Service Console »** dispose d'assez de ressources (256MB de RAM minimum mais 512MB pour un fonctionnement optimal). Dans le cas contraire, nous pouvons observer les symptômes suivants (exemples) :

   – Mauvais temps de réponse du vSphere Client (Pas assez de mémoire)
   – Impossible de se loguer dans le service console (Pas assez d'espace disque)

4. Utiliser des **systèmes d'exploitation** compatibles.

   « *http://www.vmware.com/pdf/vsphere4/r40/vsp_compatibility_matrix.pdf (p.17-19)* »

5. Toujours installer la dernière version des **VMware tools** sur le *Guset OS* et la maintenir à jour, surtout après une mise à jour de l'ESX/ESXi.

   « *Voir section 2.1.6* »

6. Disposer de la dernière **version hardware** compatible avec l'ESX/ESXi installé.

   « *Voir section 2.8* »

7. Désactiver les **écrans de veille** et autres **animations** Windows sur les machines virtuelles. Sur Linux, désactiver le **server X**, s'il n'est pas utilisé. Toutes ces fonctionnalités consomment de la ressource CPU non négligeable.

8. Dans la mesure du possible, activer le **HA** et le **DRS**.

   « *HA, voir section 2.4* »
   « *DRS, voir section 2.5* »

9. Déconnecter tous **les périphériques non utilisés** sur l'*host* et sur les *guest* comme :

   – Les ports COM
   – Les ports LPT
   – Lecteurs Floppy
   – Lecteurs CD-ROM
   – Les ports USB

10. Programmer les **backups** et les **scans antivirus** dans des heures creuses comme le milieu de la nuit. À appliquer, aussi bien pour les ESX/ESXi que pour les VMs.

5.1.2 Détection de problèmes

Il est évident qu'il faut optimiser au maximum un système en production et le monitorer mais à partir de quel moment pouvons nous considérer qu'un problème existe ?

En règle général, il existe des SLA (2.4.1) dans chaque entreprise qui définissent quels sont les seuils critiques pour la production. Il est évident que pour une ressource, comme le CPU, le seuil critique sera pratiquement toujours au même niveau pour tout système informatique, mais pour une ressource moins critique, comme l'espace de stockage, la limite critique peut varier.

Il faut donc s'appuyer sur ces SLA, afin d'utiliser au mieux les outils de monitoring à disposition. Dans le cadre du stage au CTI, les deux outils utilisés sont :

- **Nagios** pour un monitoring constant et pour le déclenchement d'alertes.
- **vSphere Client** + **vCenter** pour un monitoring ponctuel et global.

Ce que nous pouvons conseiller, d'après les nombreuses lectures effectuées, c'est d'utiliser trois outils pour cibler un problème. Les deux premiers sont ceux utilisés au CTI, mais il faudrait en rajouter un troisième et qui reste l'outil le plus puissant, pour le monitoring en temps réel, qui est **Esxtop**.

Ce dernier est parfois laissé de côté, car peut-être un peu plus lourd à utiliser, comme il n'a pas d'interface graphique avec de jolis compteurs. Mais, il reste néanmoins l'outil le plus intéressant, car il donne accès à des compteurs auxquels les deux autres outils n'ont pas accès et le taux de rafraîchissement est plus élevé, donc plus précis.

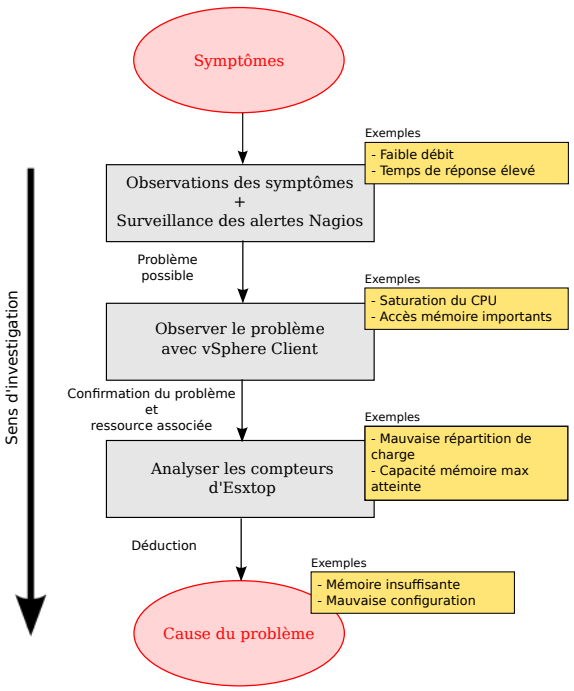


FIGURE 5.2 – Marche à suivre

Nous pouvons donc définir la marche à suivre représentée sur l'image (FIG 5.2), comme étant la meilleure façon de monitorer une infrastructure virtualisée.

5.1.2.1 Observation avec Nagios

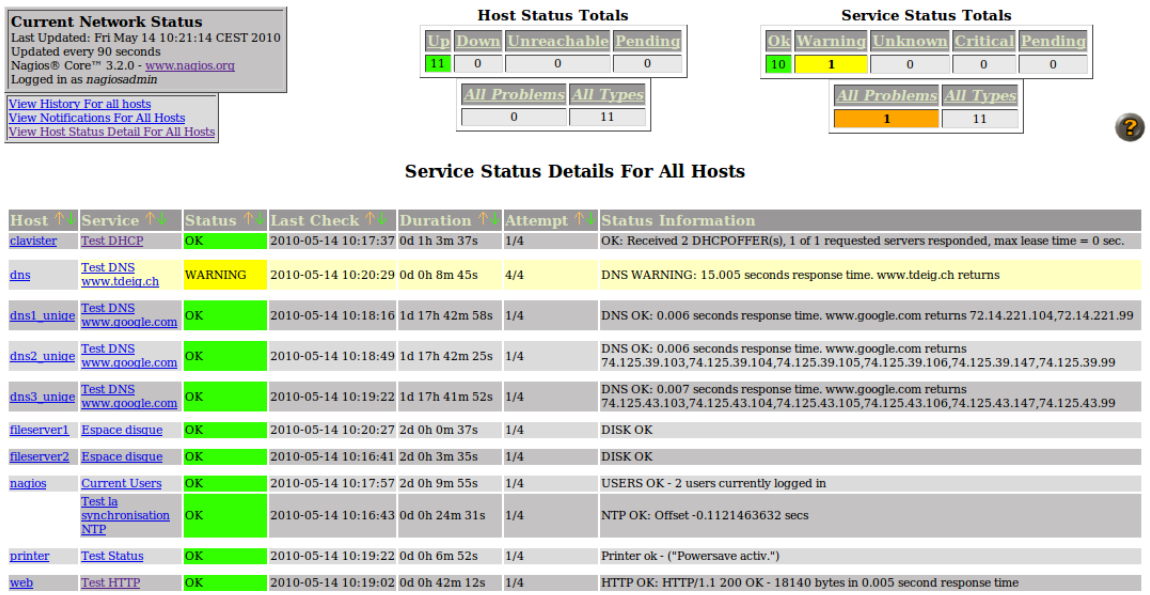
Le monitoring sous Nagios se fait grâce à des greffons, qui ne sont rien d'autre que des scripts, programmes ou autres qui peuvent être développés par n'importe qui.

La seule contrainte est que ces greffons doivent retourner certains codes précis pour que Nagios puisse les comprendre et les utiliser. Il existe en tout quatre codes :

- 0 : OK
- 1 : WARNING
- 2 : CRITICAL
- 3 : UNKNOWN

Pour exemple de greffon utilisé par Nagios, vous trouverez en annexe le code source du programme C qui permet de tester le ping avec *check_ping* (Annexe A.2)

Nagios va ensuite exécuter tous les greffons périodiquement et nous reporter les états qui sont retournés. Ce sont ces états qui définiront si une alerte doit être déclenchée. La vue offerte par Nagios pour le monitoring des services, nous permet d'avoir une vision globale sur l'état de tous ces services (FIG 5.3)



Current Network Status
Last Updated: Fri May 14 10:21:14 CEST 2010
Updated every 90 seconds
Nagios® Core™ 3.2.0 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
11	0	0	0

All Problems	All Types
0	11

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
10	1	0	0	0

All Problems	All Types
1	11

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
clavister	Test DHCP	OK	2010-05-14 10:17:37	0d 1h 3m 37s	1/4	OK: Received 2 DHCPOFFER(s), 1 of 1 requested servers responded, max lease time = 0 sec.
dns	Test DNS www.tdeig.ch	WARNING	2010-05-14 10:20:29	0d 0h 8m 45s	4/4	DNS WARNING: 15.005 seconds response time. www.tdeig.ch returns
dns1_unige	Test DNS www.google.com	OK	2010-05-14 10:18:16	1d 17h 42m 58s	1/4	DNS OK: 0.006 seconds response time. www.google.com returns 72.14.221.104,72.14.221.99
dns2_unige	Test DNS www.google.com	OK	2010-05-14 10:18:49	1d 17h 42m 25s	1/4	DNS OK: 0.006 seconds response time. www.google.com returns 74.125.39.103,74.125.39.104,74.125.39.105,74.125.39.106,74.125.39.147,74.125.39.99
dns3_unige	Test DNS www.google.com	OK	2010-05-14 10:19:22	1d 17h 41m 52s	1/4	DNS OK: 0.007 seconds response time. www.google.com returns 74.125.43.103,74.125.43.104,74.125.43.105,74.125.43.106,74.125.43.147,74.125.43.99
fileserver1	Espace disque	OK	2010-05-14 10:20:27	2d 0h 0m 37s	1/4	DISK OK
fileserver2	Espace disque	OK	2010-05-14 10:16:41	2d 0h 3m 35s	1/4	DISK OK
nagios	Current Users	OK	2010-05-14 10:17:57	2d 0h 9m 55s	1/4	USERS OK - 2 users currently logged in
	Test la synchronisation NTP	OK	2010-05-14 10:16:43	0d 0h 24m 31s	1/4	NTP OK: Offset -0.1121463632 secs
printer	Test Status	OK	2010-05-14 10:19:22	0d 0h 6m 52s	1/4	Printer ok - ("Powersave activ.")
web	Test HTTP	OK	2010-05-14 10:19:02	0d 0h 42m 12s	1/4	HTTP OK: HTTP/1.1 200 OK - 18140 bytes in 0.005 second response time

FIGURE 5.3 – Monitoring Nagios

5.1.2.2 Observation avec vSphere Client + vCenter

vSphere Client nous offre quelques vues globales de l'infrastructure virtualisée qui sont intéressantes pour se donner une idée rapide de la santé des machines hôtes, sur lesquelles sont exécutés les ESX/ESXi.

Par exemple, nous pouvons obtenir une vue nous fournissant l'utilisation des ressources globales dont dispose le cluster (5.4).

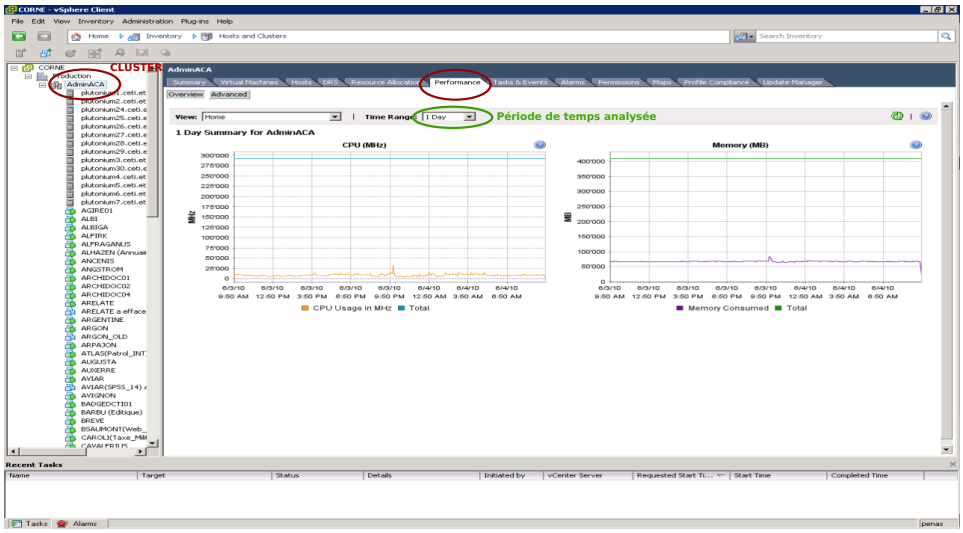


FIGURE 5.4 – Cluster

Nous pouvons également afficher une vue d'un cluster sur laquelle, il y a un listing de toutes les machines ESX/ESXi (FIG 5.5) disponibles, avec l'utilisation mémoire et processeur (en %) qui sont, malgré tout, les deux ressources les plus importantes au niveau des performances d'un système. Nous pouvons également y retrouver le statut de la machine.

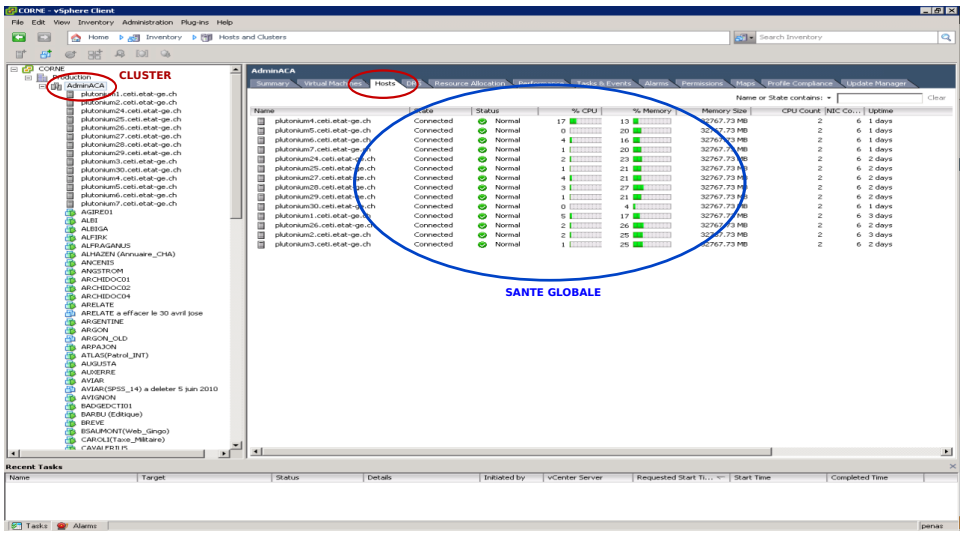


FIGURE 5.5 – Cluster & ESX/ESXi

Et pour terminer pratiquement la même vue, mais cette fois-ci sous forme de graphique avec l'addition des ressources utilisées par chaque hôte ESX/ESXi (FIG 5.6).

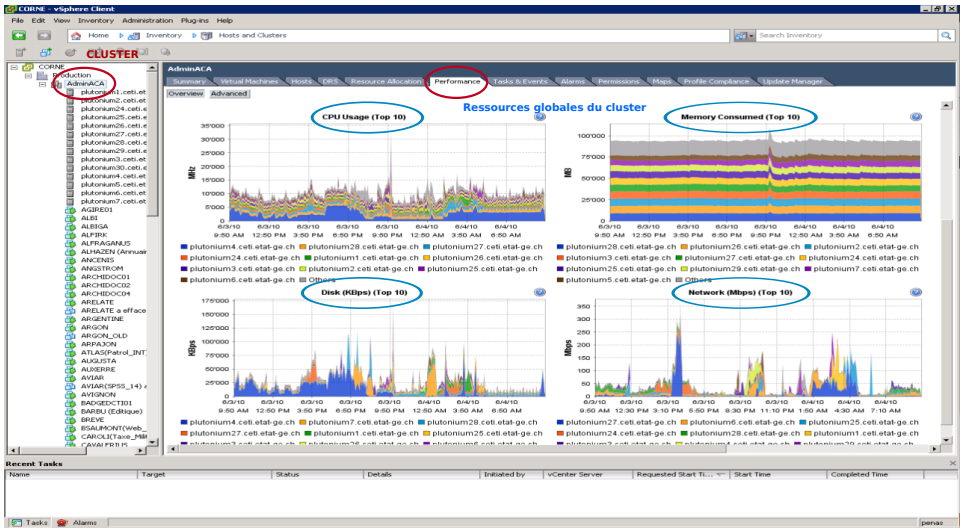


FIGURE 5.6 – Cluster & ESX/ESXi

5.1.2.3 Observation avec Esxtop [2]

L'outil Esxtop nous fournit un certain nombre de compteurs qui nous permettent de monitorer les ressources du système. Nous pouvons très facilement modifier la vue qu'il nous fournit, grâce à des options et nous pouvons également exporter les données dans des fichiers comme, par exemple, *.csv* .

```
 1:54:42pm up 33 min, 143 worlds; CPU load average: 0.03, 0.01, 0.00
PCPU USED(%):   0.7   0.4 AVG:   0.5
PCPU UTIL(%):   0.7   0.5 AVG:   0.6

    ID    GID NAME             NWLD   %USED    %RUN    %SYS   %WAIT    %RDY   %IDLE  %OVRLP   %CSTP  %MLMTD  %SWPWT
     1      1 idle                2  199.01  198.99    0.00    0.00    1.00    0.00    0.04    0.00    0.00    0.00
     2      2 system              6    0.00    0.00    0.00  599.99    0.00    0.00    0.00    0.00    0.00    0.00
     3      3 vim                 1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
     6      6 helper             57    0.03    0.04    0.00 5700.00    0.01    0.00    0.00    0.00    0.00    0.00
     7      7 drivers             9    0.00    0.00    0.00  900.00    0.00    0.00    0.00    0.00    0.00    0.00
     8      8 vmotion             4    0.00    0.00    0.00  400.00    0.00    0.00    0.00    0.00    0.00    0.00
    13     13 vmkapimod           7    0.00    0.00    0.00  700.00    0.00    0.00    0.00    0.00    0.00    0.00
    16     16 init.4171           1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   157    157 busybox.4318        1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   161    161 vmklogger.4322      1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   626    626 FT                  1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   710    710 busybox.4885        1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   756    756 ntpd.4930           1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   772    772 sh.4945             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   782    782 hostd.4955         11    0.33    0.33    0.00 1099.70    0.02    0.00    0.03    0.00    0.00    0.00
   794    794 sh.4967             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   804    804 vobd.4977           8    0.00    0.00    0.00  800.00    0.00    0.00    0.00    0.00    0.00    0.00
   824    824 slpd.5006           1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   893    893 sh.5075             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   903    903 openwsmand.5085     3    0.00    0.00    0.00  300.00    0.00    0.00    0.00    0.00    0.00    0.00
   920    920 sh.5104             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   948    948 sh.5132             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   954    954 sh.5138             1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   955    955 busybox.5139        1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   964    964 net-cdp.5148        1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   972    972 ash.5158            1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   973    973 ash.5159            1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   980    980 dcui.5166           1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   981    981 ash.5167            1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
   985    985 busybox.5171        1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
  1086   1086 sfcbd.5270          1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
  1087   1087 cimslp.5271         1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
  1088   1088 sfcbd.5272          1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
  1089   1089 sfcbd.5273          1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
  1090   1090 sfcbd.5274          1    0.00    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00
```

FIGURE 5.7 – Esxtop

Par défaut, Esxtop se rafraîchit toutes les 5 secondes. Nous pouvons modifier cette valeur en pressant la touche "s", puis, en spécifiant la nouvelle valeur [secondes].

2.

```
s 2
```

Pour obtenir de l'aide sur les touches disponibles, il suffit de presser la touche "h".

Ce qui nous donnera la liste suivante :

c affiche les compteurs relatifs au CPU
m affiche les compteurs relatifs à la RAM
d affiche les compteurs relatifs aux disques
n affiche les compteurs relatifs aux interfaces réseau
v affiche les compteurs relatifs aux disques des VMs
V valable pour les autres vue et permet de ne voir que les VMs

Il existe bien évidemment d'autres options mais qui sont moins importantes.

Afin de récupérer toutes les données produites par Esxtop, nous pouvons les exporter dans un fichier qui sera lisible avec par exemple **Perfmon** ou **Excel** grâce à la commande suivante :

```
esxtop -b -d 2 -n 100 > capture.csv
```

L'option -d définit le taux de rafraîchissement, ici fixé à 2 secondes et -n définit le nombre d'itérations qui seront effectuées, ici 100. Cela nous produira donc une capture de $2 * 100 = 200[s]$.

5.2 Processeur (CPU)

Avant de se lancer dans la configuration ou l'analyse des performances d'un processeur virtualisé, il convient de comprendre quels sont les différents types de virtualisation et quelles sont les conséquences qu'elles peuvent avoir sur le système.

La virtualisation n'est pas à confondre avec l'émulation. Sur cette dernière, toutes les instructions sont exécutées par une partie logicielle qui porte le nom d'émulateur. Elle permet de pouvoir exécuter du code, qui n'est normalement pas fait pour être lancé sur la machine hôte. Par exemple, du code prévu pour une architecture x86 pourrait très bien être exécutée sur du PowerPC. Par contre, il ne s'agit là nullement de virtualisation, car ce procédé ne revient qu'à écrire un logiciel qui émule et reproduit le comportement de l'architecture de destination, acceptant les même entrées et fournissant normalement un résultat similaire.

Contrairement à ceci, la virtualisation du CPU possède deux modes d'exécution comme nous l'avons déjà évoqué (2.2.4). Les instructions peuvent être directement exécutées par le processeur physique, lorsqu'il s'agit d'instructions non privilégiées, alors que, dans le cas contraire, la couche de virtualisation prend la main pour les exécuter, cette fois ci, par le processeur virtuel.

Il est donc évident qu'un overhead dû à la virtualisation ne sera présent que dans le cas d'une exécution d'instruction privilégiée.

5.2.1 Les bonnes pratiques [3]

VMware propose également au niveau de CPU quelques bonnes pratiques à respecter, afin d'optimiser les performances. Lorsqu'un problème survient à ce niveau du hardware, il faut donc, tout d'abord, vérifier que les différents points suivants sont respectés :

- Toujours activer les **Vmware Tools**.

  « *Voir section 2.1.6* »

- Disposer de la dernière **version hardware** compatible avec l'ESX/ESXi installé.

  « *Voir section 2.8* »

- Utiliser la dernière génération de processeurs qui permettent d'améliorer les performances grâce à la technologie **VT (Intel)** ou **V (AMD)**.

  « *Voir section 2.2.4.3* »

- Dans la mesure du possible, **associer certaines tâches à un seul vCPU** pour éviter les « overhead » dus au changement de vCPU lors du traitement.

- Utiliser le **moins de vCPU possible**, ne pas mettre plusieurs vCPU pour une application ou plus particulièrement un OS qui n'accepte pas le multi-threading.

- Ne pas laisser des **vCPU inutilisés**, car ils consomment de la ressource en exécutant l'*idle loop* de l'OS invité.

- Toujours contrôler si l'**OS invité** supporte plusieurs ou un seul CPU.

- Toujours activer **l'hypertreading** s'il est disponible (BIOS et ESX/ESXi)

5.2.2 Monitorer le CPU

Nous avons pu voir dans la section 5.1.2.3 qu'il y a un grand nombre de compteurs disponibles pour chaque ressource et en particulier pour le CPU. L'important, ici, est de faire le tri et de bien savoir lesquels sont importants et permettent de diagnostiquer un éventuel problème des systèmes physiques et virtuels.

VMware nous fournit donc une liste [4] des compteurs, qui d'après eux sont les plus importants et c'est sur cette base que le monitoring des systèmes a été réalisé.

3. « *http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf (p.11-12, 19-22)* »
4. http://communities.vmware.com/docs/DOC-5240

Nous avons donc deux compteurs CPU qui se mettent en avant par rapport aux autres :

%RDY qui indique le pourcentage de temps qu'un groupe ou une VM passe à attendre la ressource CPU pour pouvoir exécuter ses tâches.

%USED qui indique le pourcentage de ressource CPU utilisée par un groupe ou une VM.

Ces deux compteurs peuvent bien évidemment être monitorer via vSphere Client et Esxtop.

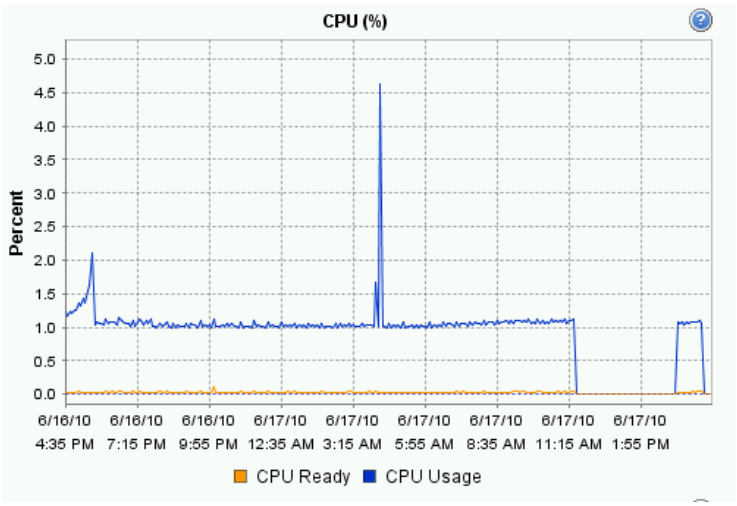
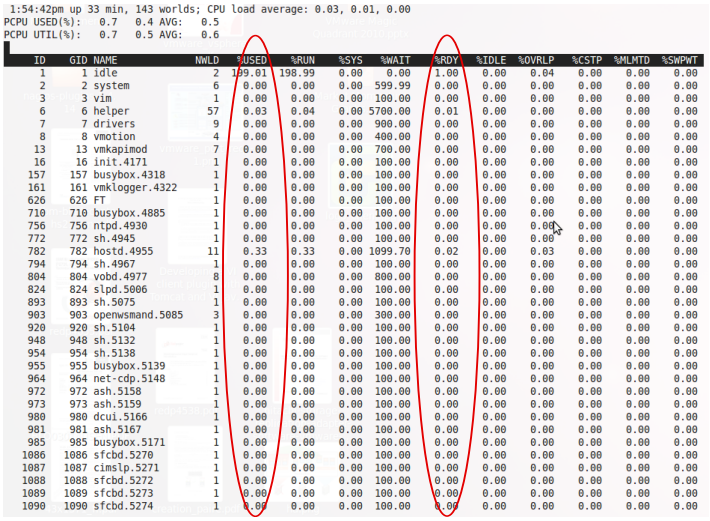


FIGURE 5.8 – CPU - vSphere Client



```
1:54:42pm up 33 min, 143 worlds; CPU load average: 0.03, 0.01, 0.00
PCPU USED(%):   0.7   0.4 AVG:   0.5
PCPU UTIL(%):   0.7   0.5 AVG:   0.6
```

ID	GID	NAME	NWLD	%USED	%RUN	%SYS	%WAIT	%RDY	%IDLE	%OVRLP	%CSTP	%MLMTD	%SWPWT
1	1	idle	2	199.01	198.99	0.00	0.00	1.00	0.00	0.04	0.00	0.00	0.00
2	2	system	6	0.00	0.00	0.00	599.99	0.00	0.00	0.00	0.00	0.00	0.00
3	3	vim	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
6	6	helper	57	0.03	0.04	0.00	5700.00	0.01	0.00	0.00	0.00	0.00	0.00
7	7	drivers	9	0.00	0.00	0.00	900.00	0.00	0.00	0.00	0.00	0.00	0.00
8	8	vmotion	4	0.00	0.00	0.00	400.00	0.00	0.00	0.00	0.00	0.00	0.00
13	13	vmkapimod	7	0.00	0.00	0.00	700.00	0.00	0.00	0.00	0.00	0.00	0.00
16	16	init.4171	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
157	157	busybox.4318	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
161	161	vmklogger.4322	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
626	626	FT	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
710	710	busybox.4885	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
756	756	ntpd.4930	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
772	772	sh.4945	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
782	782	hostd.4955	11	0.33	0.33	0.00	1099.70	0.02	0.00	0.03	0.00	0.00	0.00
794	794	sh.4967	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
804	804	vobd.4977	8	0.00	0.00	0.00	800.00	0.00	0.00	0.00	0.00	0.00	0.00
824	824	slpd.5006	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
893	893	sh.5075	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
903	903	openwsmand.5085	3	0.00	0.00	0.00	300.00	0.00	0.00	0.00	0.00	0.00	0.00
920	920	sh.5104	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
948	948	sh.5132	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
954	954	sh.5138	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
955	955	busybox.5139	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
964	964	net-cdp.5148	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
972	972	ash.5158	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
973	973	ash.5159	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
980	980	dcui.5166	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
981	981	ash.5167	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
985	985	busybox.5171	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
1086	1086	sfcbd.5270	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
1087	1087	cimslp.5271	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
1088	1088	sfcbd.5272	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
1089	1089	sfcbd.5273	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
1090	1090	sfcbd.5274	1	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00

FIGURE 5.9 – CPU - Esxtop

5.2.3 Problèmes & Solutions [5]

Avec l'aide et l'analyse des 2 compteurs vus dans la section précédente, nous pouvons détecter les principaux problèmes qui ont un lien avec la ressource CPU. Il est évident que nous ne pouvons pas fournir une liste exhaustive de tous les problèmes qui pourraient survenir sur un système virtualisé mais la liste suivante comporte les 5 problèmes qui sont le plus fréquemment observés selon VMware :

- Saturation du système hôte
- Saturation du système invité
- Utilisation d'un seul vCPU sur une VM configuré en SMP
- Faible utilisation du vCPU de l'invité
- Forte utilisation du CPU physique 0

Nous allons donc expliquer, pour chacun d'eux, quels sont les causes les plus probables et dans la mesure du possible donner des solutions à explorer pour aboutir à une résolution des problèmes.

5.2.3.1 Saturation du système hôte

- **Symptômes**

  Le problème de saturation peut être détecter grâce aux deux compteurs *%USED* et *%RDY*. Si l'utilisation de la ressource CPU de l'hôte est en permanence en dessus des 75% ou que les pics en dessus des 90% sont trop fréquent, alors il existe une saturation de l'hôte.

- **Causes**

  La principale raison pour laquelle peut survenir une saturation du système hôte, est que les VMs qui tournent dessus demandent plus de ressources que disponible. Plusieurs scénarios peuvent amener à ce problème dont voici les principaux :

  – L'hôte possède un petit nombre de VMs, avec chacune une grande demande en CPU.
  – L'hôte possède un grand nombre de VMs, avec une demande CPU qui peut être élevée ou faible peu importe.

- **Solutions**

  Pour corriger ce problème, nous pouvons envisager plusieurs solutions.

  – Dans un premier temps, si l'hôte ESX/ESXi ne peut pas être mis en *cluster*, il faut réduire le nombre de VMs qu'il possède ou alors en migrer certaines sur un second hôte ce qui revient à faire du *load balancing* manuellement. Il convient bien évidemment d'analyser au préalable les compteurs des VMs, afin de détecter celles qui demandent le plus de ressource CPU.

  – Si l'infrastructure le permet, nous pouvons choisir la solution du *load balancing* automatique avec le mécanisme et DRS, mais il faut disposer d'un *cluster* auquel ajouter notre hôte qui est

5. « *http://communities.vmware.com/servlet/JiveServlet/download/10352-2-28235/vsphere4-performance-troubleshooting.pdf (p.29-35)* »

saturé.

– Le matériel virtuel que possède une VM a également une influence directe sur les performances. Si une VM est configurée pour utiliser 3 vCPU alors que l'OS ne le supporte pas et bien, ce sera de la ressource gaspillée, puisqu'en diminuant le nombre de vCPU de la VM nous libéreront de la ressource pour les autres VMs.

– Si aucune de ces solutions ne fonctionne, nous pouvons toujours limiter manuellement une VM en ressource CPU. En effet, il arrive que certaines applications ne soient pas de bonnes candidates à la virtualisation et qu'elles consomment énormément de ressources.

5.2.3.2 Saturation du système invité

- **Symptômes**

  Pareil que pour la saturation de l'hôte, mais au niveau de la VM. Si le compteur *%USED* de la charge CPU d'une VM dépasse constamment les 75% ou a trop de pics au dessus des 90% alors nous pouvons affirmer qu'une saturation de l'invité existe.

- **Causes**

  La saturation du système invité provient du fait que le système d'exploitation ou une des applications qui tournent dessus consomme toute la ressource CPU qui a été fournie à la VM par l'hôte ESX/ESXi. C'est le cas avec notamment des applications de calculs intensifs qui prennent toute la ressource disponible.

- **Solutions**

  – Si la VM ne dispose pas déjà du maximum de ressources qu'il peut lui être alloué, nous pouvons lui ajouter des vCPU pour autant que le système d'exploitation invité le permette.

  – Si l'application, qui cause la saturation du système invité, le permet, nous pouvons lancer une seconde VM faisant tourner la même application et faire du *load balancing* entre les deux.

  – Pour certaines applications, il est utile de maintenir à jour les versions car certains *patchs* peuvent améliorer la comptabilité avec la virtualisation, ce qui est un des problème principal pour des applications anciennes.

5.2.3.3 Utilisation d'un seul vCPU sur une VM configuré en SMP

- **Symptômes**

  Lorsque qu'une machine virtuelle est configuré avec plusieurs vCPU et que nous observons qu'un seul des vCPUs est utilisé.

- **Causes**

Plusieurs raisons peuvent être la cause de ce problème, mais pour la plupart, il ne s'agit que de problèmes de compatibilité ou de mauvaise configuration.

- Si nous pouvons observer le vCPU0 travailler seul, alors il se peut que le système d'exploitation invité ne supporte pas de multiple processeurs. Il faut alors se référer à la documentation du système d'exploitation installé pour confirmer.

- Certains systèmes d'exploitation permettent d'assigner un seul cœur à une application. Si votre système comporte cette fonctionnalité, il faut vérifier que la configuration est correcte.

- Enfin, il existe encore beaucoup d'applications qui ne sont pas prévues pour être virtualisées ou qui ne supporte tout simplement pas le multi-threading. Dans ce cas, il faut se renseigner sur les mises à jour possible ou sur les produits équivalents qui pourraient corriger ces problèmes.

- **Solutions**

  - Le système d'exploitation invité doit être étudié un minimum avant d'être installé et surtout virtualisé. Il faut au moins se renseigner sur la configuration maximale qu'il peut supporter et donc configurer les VMs en fonction.

  - Les documentations des applications qui souhaitent être virtualisées doivent également être parcourues pour s'assurer de la compatibilité et se renseigner si l'utilisation du CPU est optimisé.

5.2.3.4 Faible utilisation des vCPUs de l'invité

- **Symptômes**

  La faible utilisation d'un processeur n'est normalement pas un problème mais si elle est couplé à une mauvaise performance d'une application, il faut le prendre comme un signal d'alarme et donc investiguer.

- **Causes**

  Ce problème peut survenir à cause des raisons suivantes :

  - Le système de stockage peut avoir un taux de réponse anormalement élevé et donc provoquer une utilisation des CPUs trop faible.

  - L'application peut avoir besoin de systèmes externes comme une base de données et il se peut, au même titre que le stockage, qu'il y ait un mauvais temps de réponse.

  - Pour en revenir au nombre de vCPUs, il se peut que l'application ne soit pas optimisée pour tirée profit des tous les vCPUs qui lui ont été alloués.

- **Solutions**

– En ce qui concerne le mauvais temps de réponse des systèmes externes comme le stockage ou les bases de données, le problème ne vient en général pas de l'hôte ESX/ESXi, ni des VMs mais plutôt d'un problème au niveau du réseau ou alors de performances au niveau du système en question.

– Si le problème vient d'une incompatibilité entre l'application est le nombre de vCPUs, il suffit juste d'en enlever et de relancer un test de performances.

5.2.3.5 Forte utilisation du CPU physique 0

- **Symptômes**

  La forte utilisation du pCPU0 par rapport aux autres pCPUs sur un système utilisant le SMP peut être détectée lorsque le pCPU0 a une activité disproportionnée par rapport à l'activité des autres.

- **Causes**

  Ce problème ne survient en général que sur un hôte ESX car il a un lien direct avec le « Service Console ». En effet, il est rattaché au pCPU0 et ne fonctionne que sur celui-ci. Plusieurs agent de management s'exécutent dans le « Service Console »et donc l'utilisation du pCPU0 est fortement liée avec l'activité des ces agents.

  Il est évident que cette forte utilisation du pCPU0 a un impacte sur les performances globales de l'ESX car il s'agit de ressource en moins pour les VMs.

- **Solutions**

  Pour corriger ce problèmes, nous pouvons reduire le nombre d'agents en arrêtant tous ceux qui ne sont pas indispensable. Mais il faut étudier cette solution au cas par cas, car il est impossible de prévoir à l'avance ceux qui ne seront pas utilisés.

5.2.4 Configurations avec vSphere Client

5.2.4.1 Activer l'assistance *hardware* VT (Intel) [6]

1. Sélectionner la VM à configurer
2. Clique droit -> **Edit virtual machine settings** -> **Options** -> **CPU/MMU Virtualization**.
3. Faire le choix entre les options suivantes :

   - **Automatic** laisse ESX choisir la meilleure option. La façon dont le choix est fait dépend fortement du type de processeur. On retrouve une grille de ces choix à l'adresse suivante http://communities.vmware.com/docs/DOC-9882.

6. « *http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf (p.22)* »

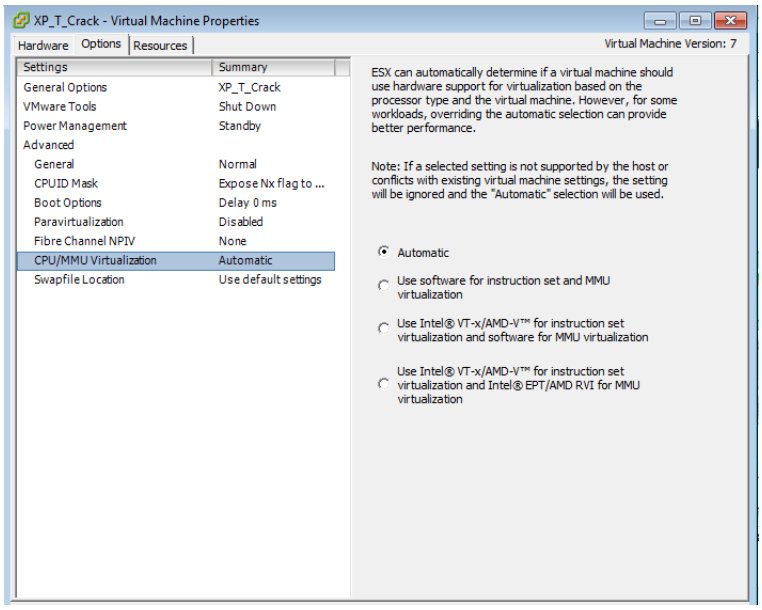


FIGURE 5.10 – Options Intel VT

- **Use software for instruction set and MMU virtualization** désactive l'assistance *hardware* pour le CPU et pour le MMU. Il y a donc utilisation de la *Binary Translation* dans la mesure du possible.

- **Use Intel® VT-x/AMD-VTM for instruction set virtualization and software for MMU virtualization** active l'assistance *hardware* VT-x si elle est disponible.

- **Use Intel® VT-x/AMD-VTM for instruction set virtualization and Intel® EPT/AMD RVI for MMU virtualization** active l'assistance *hardware* VT-x et celle pour le MMU, si elles sont disponibles.

Les choix de l'utilisateur qui a créé une machine virtuelle, ainsi que les choix d'ESX/ESXi peuvent être observés à partir d'une fichier de logs après avoir démarré au moins une fois la VM (*/vmfs/volumes/datastore1/nom_machine_virtuelle/vmware.log*). Ainsi nous obtenons quelque chose qui ressemble à ceci :

```
Jun 25 16:10:52.536: vmx| MONITOR MODE: allowed modes    : BT HV
Jun 25 16:10:52.536: vmx| MONITOR MODE: user requested modes : BT HV HWMMU
Jun 25 16:10:52.536: vmx| MONITOR MODE: guestOS preferred modes: HWMMU HV BT
Jun 25 16:10:52.536: vmx| MONITOR MODE: filtered list    : HV BT
Jun 25 16:10:52.536: vmx| HV Settings: virtual exec = 'hardware'; virtual mmu = 'software'
```

Extrait de vmware.log

– La première ligne nous donne les modes disponibles (BT = Binary Translation, HV = Intel VT).
– La deuxième ligne sont les choix fait par l'utilisateur sur la VM par ordre de priorité.
– La troisième ligne sont les préférences de l'OS invité.

– La cinquième ligne est ce qui est effectivement appliqué.

5.2.4.2 Activer l'hyperthreading [7]

1. S'assurer que le système supporte la technologie hyperthreading.

2. Activer l'hyperthreading dans le BIOS. Le nom utilisé par les constructeurs peut être « Logical Processor »ou alors tout simplement « Enable Hyperthreading ».

3. Activer l'hyperthreading sur l'ESX/ESXi.

   - Dans vSphere Client, cliquer sur l'hôte ESX/ESXi puis sur l'onglet **Configuration**
   - Cliquer sur **Processors** -> **Properties**
   - Activer l'option **Hyperthreading**

7. « *http://www.vmware.com/pdf/vsphere4/r40/vsp_40_resource_mgmt.pdf (p.19)* »

6 Labo TD

6.1 Serveur Nagios

6.1.1 Nagios

L'installation de l'application Nagios sur le serveur 10.1.1.5 du laboratoire a été réalisés grâce à la méthode décrite au point 4.4.1.

La connexion à l'interface web se fait grâce à un navigateur, en entrant l'adresse http://10.1.1.5/nagios3.

6.1.1.1 Problèmes rencontrés

Après l'installation, la configuration de Nagios peut être vérifiée avec la commande suivante :

```
sudo nagios3 v /etc/nagios3/nagios.cfg
```

Ce qui a mis en évidence les 2 erreurs suivantes :

- Le fichier resource.cfg ne peut pas être lu
- Le chemin /var/lib/nagios3/spool/checkresults est invalide

Ces deux problèmes sont dus aux droits sur ces fichiers/dossiers. Pour les corriger, il suffit d'effectuer les manipulations suivantes :

```
sudo chown root :root resource.cfg
```

```
sudo chmod o+r resource.cfg
```

Ce qui fixe l'*owner* et le groupe du fichier à « root » et donne la permission de lecture à tout le monde.

```
sudo chmod uog+rwx /var/lib/nagios3
```

```
sudo chmod uog+rwx /var/lib/nagios3/spool
```

```
sudo chmod uog+rwx /var/lib/nagios3/spool/checkresults
```

Ce qui donne tous les droits sur ces dossiers/fichiers.

6.1.2 NTP [1]

Afin que les heures présentes sur les logs de Nagios soient toujours justes, il a fallu installer et configurer un client NTP sur le serveur Nagios (10.1.1.5), qui se synchronisera avec le serveur NTP de l'UNIGE.

Le client, par défaut sur Ubuntu, est « ntpdate »et c'est par conséquent celui qui a été installé.

Il faut ensuite le configurer pour qu'il aille chercher les infos à l'UNIGE, en éditant */etc/default/ntpdate* , comme ceci :

```
1 NTPDATE_USE_NTP_CONF=no
  NTPSERVERS="129.194.184.1"
```

Contenu de ntpdate

A côté de ça, il faut un démon qui lance la synchronisation à intervalle régulier. C'est la tâche de « ntp », qui s'installe avec la commande suivante :

```
sudo aptget install ntp
```

Puis, également lui indiquer le serveur NTP dans le fichier */etc/ntp.conf* :

```
server 129.194.184.1
```

Contenu de ntp.conf

Et pour terminer redémarrer le service :

```
sudo /etc/init.d/ntp restart
```

1. http://blog.nicolargo.com/2010/03/installation-dun-serveur-ntp-sous-ubuntu.html

7 Stage au CTI

7.1 Migration vCenter

Le premier travail effectué au CTI a été la migration de vCenter d'un Microsoft Windows Server 2003 32bits sur un Microsoft Windows Server 2008 64bits.

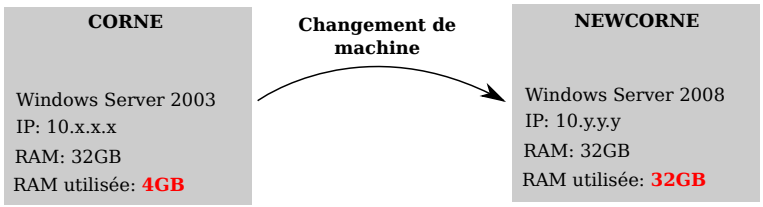


FIGURE 7.1 – Migration vCenter

Ce changement a été décidé principalement pour permettre l'utilisation de la totalité de la mémoire RAM disponible, car Windows Server 2003 ne prenait en compte que 4GB au maximum.

Nous avons donc fait, dans l'ordre, les étapes suivantes :

1. Installation de vCenter sur la nouvelle machine NEWCORNE, en suivant la procédure décrite en 4.1.5.
2. Installation de vSphere Client sur NEWCORNE.
3. Désactivation des services VMware sur CORNE, ainsi que l'arrêt de la machine.
4. Renommer NEWCORNE en CORNE.
5. Modification de l'IP du nouveau CORNE de 10.y.y.y en 10.x.x.x.

Une fois le nouveau serveur prêt, il ne nous reste plus qu'à lancer vSphere Client pour se connecter au nouveau vCenter.

La partie de configuration de vCenter se fait en 3 étapes :

1. Ajout d'un « Data Center »nommé **Production**.
2. Ajout d'un cluster par zone de production.
3. Ajout des hôtes ESX dans leurs clusters correspondants.

Le point important, à respecter ici, est de ne pas activer le HA, ni le DRS au moment de la création des *clusters*, mais une fois que toutes les machines ESX ont été ajoutées correctement, afin qu'elles puissent être configurées toutes en même temps.

7.2 CPU

Lors du stage, certains tests ont pu être effectué au niveau du CPU.

Premièrement, comparer les implications du choix du système d'exploitation invité, sur la configuration vCPU d'une machine virtuelle.

Deuxièmement, utiliser un logiciel de benchmarking afin de comparer les performances entre un système d'exploitation natif et deux autres qui ont été virtualisés.

7.2.1 Choix du système d'exploitation

Comme nous l'avons déjà expliqué, le choix du système d'exploitation que nous désirons virtualiser a une importance toute particulière sur les performances CPU.

En effet, certains systèmes sont limités et ne peuvent pas supporter autant de CPU que l'on souhaiterait.

Pour illustrer ceci, nous avons installé « Microsoft Windows Server 2003 »et un « Microsoft Windows Server 2008 ». Les tests ont été effectués sur les IBM Blade HS21 donc il y a 8 cores à disposition.

Deux VMs ont été créées avec chacune huit vCPUs. Sachant que le premier OS, qui est également le plus ancien, ne gère que quatre CPU au maximum, nous pouvons facilement nous douter du résultat. Par contre, le second les gère entièrement.

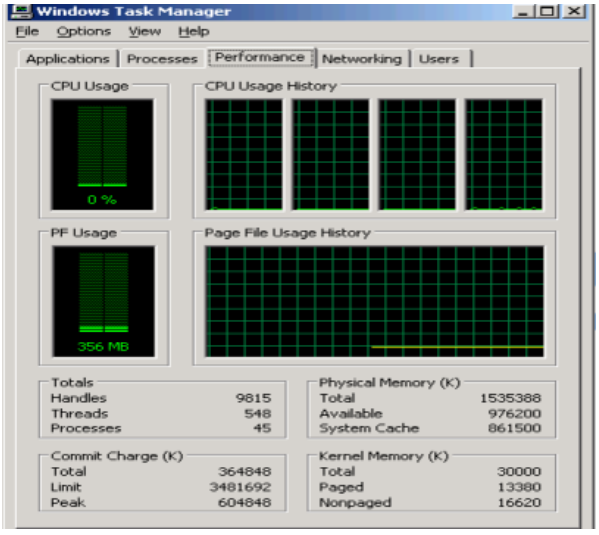


FIGURE 7.2 – Windows 2003

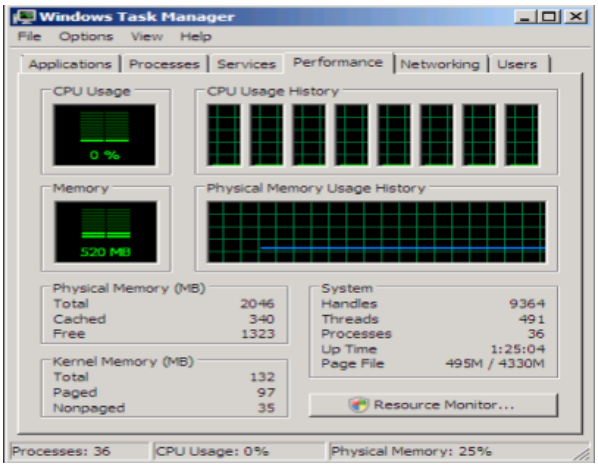


FIGURE 7.3 – Windows 2008

7.2.2 Benchmarking

En ce qui concerne le benchmarking, VMware recommande certains logiciels, notamment les logiciels de PassMark [1] parmi lesquels se trouvent le très célèbre « BurnInTest ».

Pour effectuer nos tests, nous avons utilisé « PerformanceTest 7.0 »en version d'évaluation.

Nous avons donc lancé ce *benchmark* sur trois systèmes différents sur des *blades* IBM HS21 :

1. Windows Server 2008 en natif
2. Windows Server 2008 virtualisé
3. Windows Server 2003 virtualisé

Voici les résultats obtenus en ce qui concerne les performances CPU :

	Win 2008	VM (Win 2008)	VM (Win 2003)
Integer (Op/s)	3261.1 M	3085.5 M	508.1 M
Floating (Op/s)	4898.2 M	4700.6 M	1976.6 M
Prime (P/s)	1423.0 K	1380.0 K	1073.1 K
Multimedia (Mat/s)	26.8 M	26.2 M	8.8 M
Compression (Oct/s)	11737.2 K	11445.3 K	5879.0 K
Cryptage (Oct/s)	33.5 M	32.3 M	16.7 M

FIGURE 7.4 – Résultats du *banchmarking*

Ce que nous pouvons remarquer, c'est que la différence entre Windows Server 2008 en natif ou virtualisé est quasiment inexistante. Alors que si nous observons la machine avec Windows Server 2003, les performances chutes d'une façon assez impressionnante. C'est la une bonne illustration de l'importance du choix de l'OS selon la configuration qu'on souhaite attribuer à une machine virtuelle.

1. http://www.passmark.com/products/index.htm

La conclusion que nous pouvons tirer de ce test, est que la virtualisation pratiquée avec prudence et avec des choix réfléchis, fournie des performances proche d'un système natif. au contraire, si elle est utilisée sans avoir pris la peine de se renseigner sur les bonnes pratiques à mettre ne place, elle amène des performances plus que désastreuses.

7.3 Hitachi : Replication True Copy

Nous avons pu voir en 3.2.2 que le système de stockage Hitachi, dont dispose le CTI, permet d'avoir des performances élevées mais à côté de cela, ces baies permettent de mettre en place un mécanisme appelé « Replication True Copy ».

Il a été étudié et mis en place durant le stage et donc, nous allons en expliquer les principes et la configuration réalisée.

7.3.1 Problématique

En analysant l'infrastructure de stockage 3.2.2 utilisée pour la virtualisation avec VMware, nous pouvons vite nous rendre compte que les baies Hitachi et le réseau SAN sont des éléments principaux et surtout indispensables de cette infrastructure. A partir de cet instant, il devient obligatoire de se demander ce qui se passerait si la baie de stockage, où se situent les fichiers des différentes VMs, venait à avoir un problème la rendant totalement indisponible ?

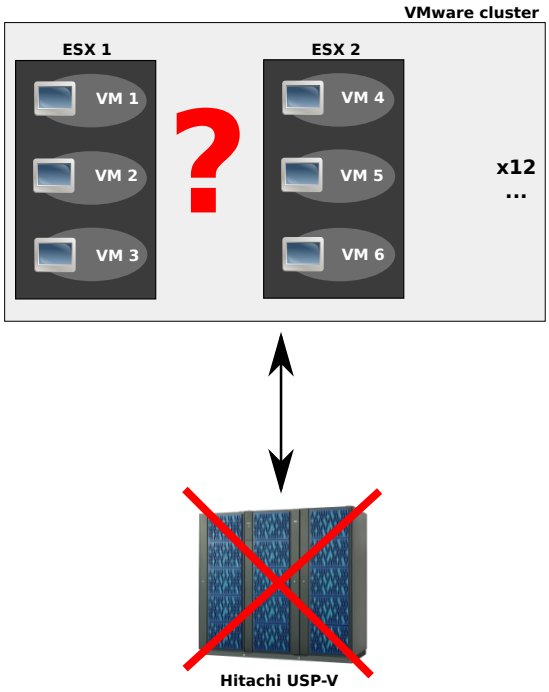


FIGURE 7.5 – Problématique SAN

Il y a bien évidemment de la technologie RAID à l'intérieur même d'une baie qui permet de prévenir un *crash* du disque, mais rien ne protège contre le *crash* de la baie entière. C'est ici qu'entre en jeu ce mécanisme de « True Copy », qui permet de faire de la duplication/synchronisation de LUNs entre

baies, afin de sauvegarder tous les fichiers utiles des VMs.

Ainsi, lorsque la baie principale, où se trouvent les fichiers des VMs, venait à être indisponible pour une raison ou une autre, l'administrateur pourrait très facilement basculer sur la baie secondaire et faire repartir les VMs.

7.3.2 Mise en place

Afin de mettre en place ce mécanisme, il nous faut quelques pré-requis au niveau hardware et software.

Tout d'abord, il nous faut obligatoirement deux baies (de même type), afin qu'il y ait un sens à cette installation.

Ensuite il faut prévoir sur la baie secondaire autant de LUNs que nous souhaitons sauvegarder et en plus 2 LUNs supplémentaires par baie qui serviront de *command device* qui est une spécificité du mécanisme. Ces *Command Device* sont des LUNs spéciaux, qui permettent de transmettre des commandes à la baie via un logiciel. Et pour terminer, deux machines virtuelles qui nous serviront à installer ce fameux logiciel Hitachi.

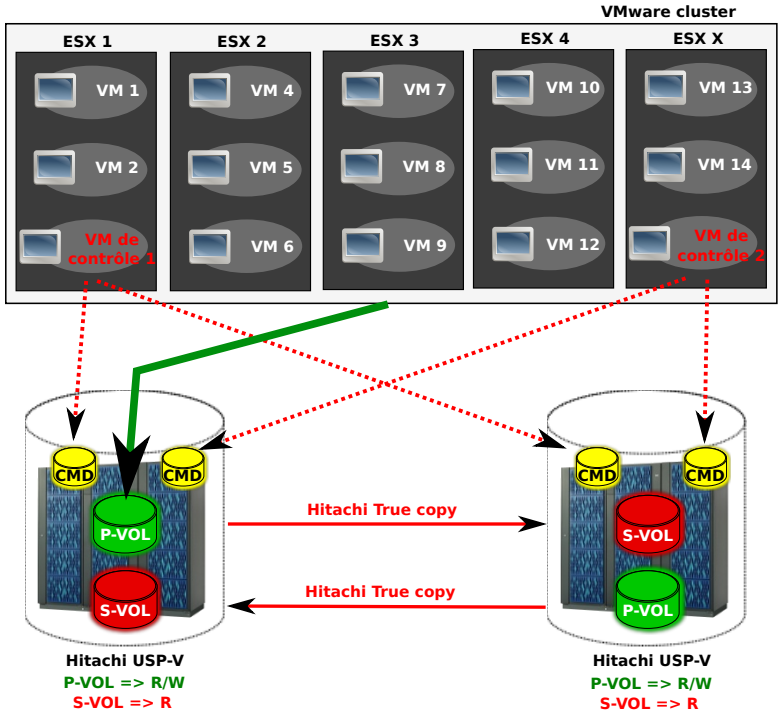


FIGURE 7.6 – Replication True Copy

Pour résumer, nous avons besoin de :

- Deux baies Hitachi USP-V
- 4 LUNs de *Command Device*
- X LUNs de backup sur la baie secondaire

- 2 VMs pour faire tourner le logiciel Hitachi

7.3.3 Fonctionnement

Le principe de fonctionnement de ce mécanisme de « Replication True Copy »est assez simple à comprendre mais un peu plus compliqué à administrer.

Nous avons donc un certain nombre de VMs qui fonctionnent en utilisant un LUN de la baie principale. Ce LUN, nous voulons le dupliquer sur la baie secondaire afin de faire un backup des fichiers des VMs. Dans la terminologie Hitachi, le LUN principal qui sera dupliqué, se nomme P-VOL (Primary Volume) et le LUN de backup le S-VOL (Secondary Volume). Le P-VOL se trouve toujours en mode Read/Write, puisque c'est celui qui est utilisé par les VMs et le S-VOL est toujours en mode Read Only.

Ces deux LUNs formeront ce qui est appelé "PAIR", qui est un lien ou une paire en français. C'est ce lien qui définira la synchronisation entre les LUNs. Il est défini par les deux LUNs à synchroniser et par le sens du backup.

Il suffira donc de remplir correctement deux fichiers de configuration par paires, qui sont en fait des « instances Horcm », qui s'adresseront chacune à une baie, et de créer la paire grâce à une commande HORCM. Ensuite le reste du travail se fera automatiquement.

7.3.4 Exemple concret

7.3.4.1 Création d'une paire de synchronisation

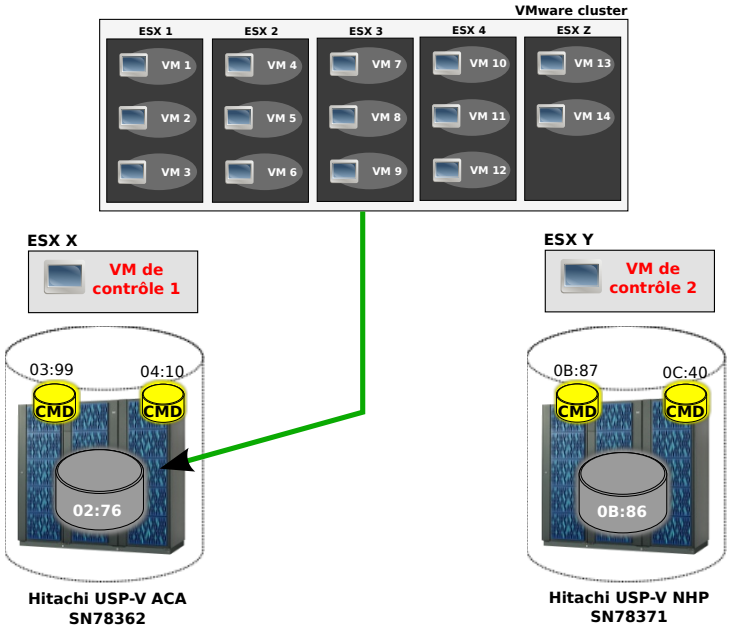


FIGURE 7.7 – Situation de départ

Nous sommes dans la situation de départ où un cluster d'ESX utilise un LUN sur la baie USP-V ACA. Nous souhaitons donc créer un backup/synchronisation de ce LUN sur la baie USP-V NHP.

Nous avons donc à disposition :

- Deux VMs de contrôle sur deux ESX différents, avec le logiciel HORCM d'Hitachi installé.
- Deux LUNs de *Command Device* sur chaque baie qui sont montés en *Raw Device Mapping* (2.3.4) sur les VMs.
- Un LUN principal sur la baie ACA et un LUN de backup sur la baie NHP.

La première chose à faire est de créer et configurer les instances HORCM sur les VMs de contrôle. Il y aura en tout quatre fichiers de configuration qui se trouvent dans */etc/* . Chacun de ces fichiers sera une instance HORCM et elles fonctionnent par paires, c'est pourquoi il y a deux fichiers de conf par VM :

- horcm0.conf et horcm1.conf sur la VM de contrôle 1, qui gèrent respectivement les *Commande Device* de la baie ACA (03 :99) et NHP (0B :87).
- horcm10.conf et horcm11.conf sur la VM de contrôle 2, qui gèrent respectivement les *Commande Device* de la baie ACA (04 :10) et NHP (0C :40).

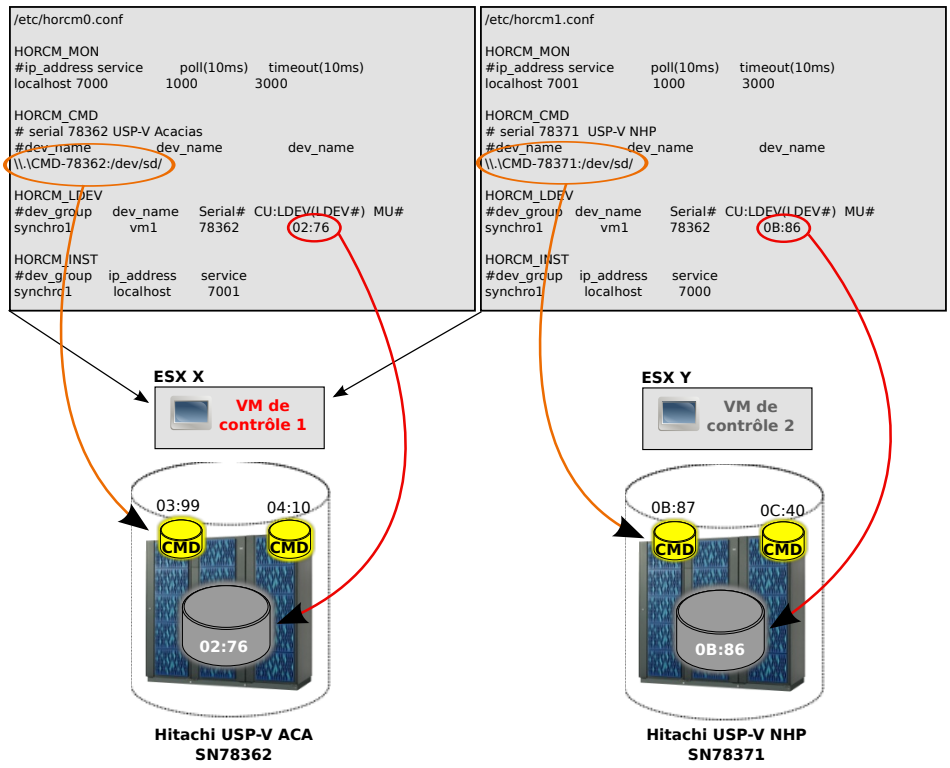


FIGURE 7.8 – HORCM sur VM 1

Les fichiers horcm10.conf et horcm11.conf, qui se trouveront sur la VM de contrôle 2 seront identiques à horcm0.conf et horcm1.conf, à part qu'il faudra changer les identifiants des *Command Device* en 04 :10 pour horcm10.conf et en 0C :40 pour horcm11.conf.

Détaillons à présent les deux fichiers de conf horcm0 et horcm1 ligne par ligne.

```
1  /etc/horcm0.conf
2
3  HORCM_MON
   #ip_address   service    poll(10ms)    timeout(10ms)
5  localhost      7000      1000          3000
6
7  HORCM_CMD
   # serial 78362 USP-V Acacias
9  #dev_name              dev_name              dev_name
   \\.\CMD-78362:/dev/sd/
11
   HORCM_LDEV
13 #dev_group    dev_name Serial#   CU:LDEV(LDEV#) MU#
   synchro1         vm1      78362         02:76
15
   HORCM_INST
17 #dev_group    ip_address    service
   synchro1       localhost      7001
```

horcm0.conf

Ligne 3-5 : On définit ici l'instance du fichier courant, donc dans notre cas, l'instance horcm0 grâce à son IP et le port qui sera utilisé.

Ligne 7-10 : On définit ici le chemin d'accès du *Command Device*, qui est monté sur la VM.

Ligne 12-14 : On définit ici le départ de la paire de synchronisation en spécifiant un nom de groupe auquel appartient la paire (au choix), nom de LUN (au choix), numéro de série de la baie, identifiant du LUN.

Ligne 16-18 : On définit ici la deuxième instance se trouvant sur la VM 1, grâce au groupe de la paire, son IP et le port qu'elle utilise.

Le horcm1.conf sera configuré de la même façon en gardant bien à l'esprit qu'il va gérer le *Command Device* à l'autre bout de la paire de synchronisation, qui se trouve sur la baie NHP.

```
   /etc/horcm1.conf
2
   HORCM_MON
4  #ip_address   service    poll(10ms)    timeout(10ms)
   localhost      7001      1000          3000
6
   HORCM_CMD
8  # serial 78371 USP-V NHP
   #dev_name              dev_name              dev_name
10 \\.\CMD-78371:/dev/sd/
12 HORCM_LDEV
   #dev_group    dev_name Serial#   CU:LDEV(LDEV#) MU#
```

```
14 synchro1        vm1      78371        0B:86

16 HORCM_INST
   #dev_group    ip_address    service
18 synchro1       localhost      7000
```

horcm1.conf

A présent que les fichiers de configuration sont prêts sur les deux VMs, nous allons pouvoir créer la paire de synchronisation. Il faut peut-être juste rappeler que les instances 10 et 11 ne seront pas utilisées, car elles serviront seulement dans le cas où la VM de contrôle 1 serait indisponible.

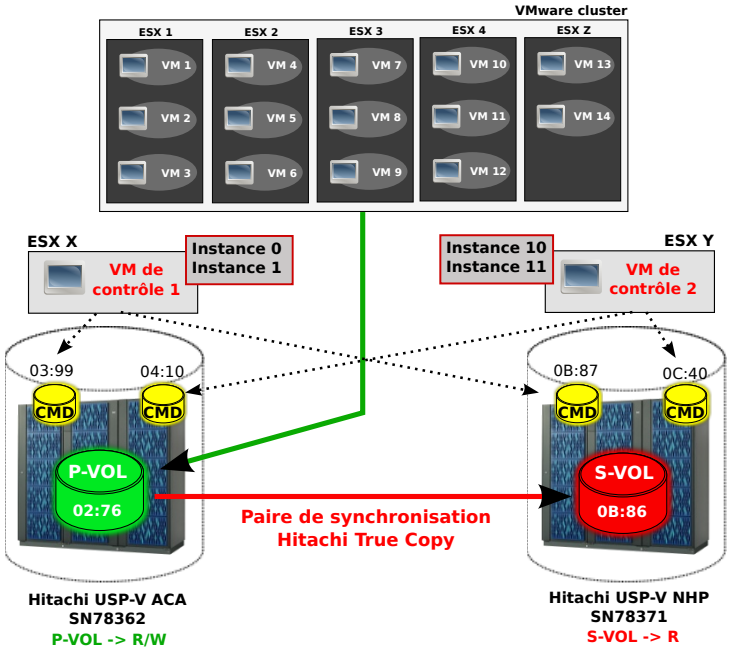


FIGURE 7.9 – Synchronisation

Nous allons donc voir toutes les commandes à taper depuis la VM 1 :

1. Plaçons-nous tout d'abord dans le bon répertoire

   cd /HORMC/usr/bin

2. Démarrons les instances 0 et 1

   ./horcmstart.sh 0 1

3. Création de la paire de synchronisation

   ./paircreate -ITC0 -g synchro1 -vl -f never

4. Vérifions que la paire a bien été créée

   ./pairdisplay -ITC0 -g synchro1 -fcx

   Nous devrions obtenir quelque chose qui ressemble à ce qui suit :

```
Group   PairVol(L/R) (Port#,TID, LU), Seq#, LDEV#.P/S, Status, Fence, \%, P-LDEV# M

synchro1  nom_ESX(L)   (CL5-B-9, 0,  1)  78362 276.P-VOL   PAIR   NEVER  100    b86
synchro1  nom_ESX(R)   (CL5-B-10, 0, 2) 78371 b86.S-VOL   PAIR   NEVER  100    276
```

Ce qui est important de vérifier dans ce retour, c'est la colonne **status** qui nous donne le statut de la paire, et la colonne %, qui nous indique le taux de synchronisation. Nous remarquons également que le premier LUN 02 :76 est en P-VOL (Primary-Volume) pour indiquer la source et que le second LUN 0B :86 est en S-VOL (Secondary-Volume) pour indiquer la destination.

Nous sommes donc à présent dans la situation décrite sur l'image (FIG 7.9)

8 Conclusion

Comme nous avons pu le voir tout au long de ce travail de diplôme, la virtualisation reste un domaine complexe dès qu'il s'agit de comprendre tous les mécanismes qui entrent en jeu. Les notions qui y sont rattachées sont également nombreuses et ne sont pas toujours simples à saisir, ce qui ne facilite pas l'apprentissage.

Néanmoins, la virtualisation a tendance à se populariser et à devenir indispensable pour ceux qui en ont compris les avantages et qui savent la maîtriser. Dans certains domaines du monde professionnel, on en arrive presque à n'avoir plus que des systèmes virtualisés et, c'est là, tout l'enjeu de comprendre et savoir comment configurer ces systèmes pour une optimisation parfaite.

Dans ce projet de diplôme, je pense avoir fait le tour des principaux outils de management/monitoring (Nagios, vCenter, Esxtop) et des points importants à respecter lors de l'utilisation de « VMware vSphere4 », essentiellement au niveau du CPU. Il est évident que ce domaine est en perpétuelle évolution et, par conséquent, de nouveaux outils et de nouvelles découvertes voient le jour quotidiennement.

Ce projet m'aura également permis de pouvoir travailler et me familiariser avec du matériel professionnel grâce à l'aide de M. Christian DELACOMBAZ (CTI), que je tiens encore à remercier. Ainsi, mes connaissances, des produits VMware et de matériels professionnels, tels que les « Hitachi USP-V »et les « IBM BladeCenter H », se sont vue enrichies.

Cette étude sera sans doute un bon point de départ pour les personnes désirant manager et optimiser une infrastructure virtualisée, afin d'acquérir les bases importantes.

A Annexes

A.1 Configuration Nagios du labo

A.1.1 Périodes de temps

```
##############################################################################
# timeperiods.cfg
##############################################################################

# This defines a timeperiod where all times are valid for checks,
# notifications, etc. The classic "24x7" support nightmare. :-)

define timeperiod{
        timeperiod_name 24x7
        alias           24 Hours A Day, 7 Days A Week
        sunday          00:00-24:00
        monday          00:00-24:00
        tuesday         00:00-24:00
        wednesday       00:00-24:00
        thursday        00:00-24:00
        friday          00:00-24:00
        saturday        00:00-24:00
        }

# Here is a slightly friendlier period during work hours
define timeperiod{
        timeperiod_name workhours
        alias           Standard Work Hours
        monday          09:00-17:00
        tuesday         09:00-17:00
        wednesday       09:00-17:00
        thursday        09:00-17:00
        friday          09:00-17:00
        }

# The complement of workhours
define timeperiod{
        timeperiod_name nonworkhours
        alias           Non-Work Hours
        sunday          00:00-24:00
        monday          00:00-09:00,17:00-24:00
        tuesday         00:00-09:00,17:00-24:00
        wednesday       00:00-09:00,17:00-24:00
        thursday        00:00-09:00,17:00-24:00
        friday          00:00-09:00,17:00-24:00
```

```
        saturday        00:00-24:00
        }

# This one is a favorite: never :)
define timeperiod{
        timeperiod_name never
        alias           Never
        }

# end of file
```

A.1.2 Contacts & Groupes

```
###############################################################################
# contacts.cfg
###############################################################################



###############################################################################
###############################################################################
#
# CONTACTS
#
###############################################################################
###############################################################################

# In this simple config file, a single contact will receive all alerts.

define contact{
        contact_name                    penas
        alias                           Penas Cedric
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,r
        service_notification_commands   notify-service-by-email
        host_notification_commands      notify-host-by-email
        email                           cedric.penas-villamisar@etu.hesge.ch
        }



###############################################################################
###############################################################################
#
# CONTACT GROUPS
#
###############################################################################
###############################################################################

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.

define contactgroup{
```

```
        contactgroup_name     admins
        alias                 Nagios Administrators
        members               penas
        }
```

A.1.3 Hôtes

A.1.3.1 Firewall Clavister

```
# a host definition for the gateway of the default route
define host {
   use     generic-host
        host_name   clavister
        alias       Routeur_Firewall vers l'exterieur
        address     10.1.0.1
        parents     router_unige
   }

define service {
   use        generic-service
   host_name     clavister
   service_description Test DHCP
   check_command   check_dhcp
   }
```

A.1.3.2 Serveurs DNS

```
define host{
        use                   generic-host          ; Name of host template$
        host_name             dns
        alias                 Serveur DNS
        display_name          Serveur DNS
        address               129.194.184.84
        parents               clavister
        }
```

A.1.3.3 Serveurs de fichiers

```
# A simple configuration file for monitoring the local host
# This can serve as an example for configuring other servers;
# Custom services specific to this host are added here, but services
# defined in nagios2-common_services.cfg may also apply.
#

define host{
        use                   generic-host          ; Name of host template$
        host_name             fileserver1
        alias                 Serveur de fichiers1
        display_name          Serveur de fichiers1
        address               10.1.1.1
        parents               clavister
        }
```

```
16 define host{
       use                     generic-host          ; Name of host template to use
18     host_name               fileserver2
       alias                   Serveur de fichiers2
20   display_name    Serveur de fichiers2
       address                 10.1.1.2
22     parents        clavister
   }
24
# Define a service to check the disk space of the root partition
26 # on the local machine. Warning if < 20% free, critical if
# < 10% free space on partition.
28
define service{
30     use                           generic-service       ; Name of servi$
       host_name                     fileserver1
32     service_description           Espace disque
       check_command                 check_all_disks!20%!10%
34     }

36 define service{
       use                           generic-service       ; Name of servi$
38     host_name                     fileserver2
       service_description           Espace disque
40     check_command                 check_all_disks!20%!10%
       }
```

A.1.3.4 Serveur Nagios

```
1  # A simple configuration file for monitoring the local host
   # This can serve as an example for configuring other servers;
3  # Custom services specific to this host are added here, but services
   # defined in nagios2-common_services.cfg may also apply.
5  #

7  define host{
       use                     generic-host          ; Name of host template to use
9      host_name               nagios
       alias                   Serveur Nagios
11   display_name    Serveur Nagios
       address                 127.0.0.1
13   parents       clavister
   }
15
   # Define a service to check the disk space of the root partition
17 # on the local machine. Warning if < 20% free, critical if
   # < 10% free space on partition.
19
   #define service{
21 #       use                           generic-service       ; Name of service template to use
   #       host_name                     nagios
23 #       service_description           Disk Space
   #       check_command                 check_all_disks!20%!10%
25 #       }
```

```
27
29 # Define a service to check the number of currently logged in
   # users on the local machine. Warning if > 2 users, critical
31 # if > 5 users.

33 define service{
       use                         generic-service      ; Name of service template to use
35     host_name                   nagios
       service_description         Current Users
37     check_command               check_users!2!5
       }
39

41 # Define a service to check the number of currently running procs
   # on the local machine. Warning if > 250 processes, critical if
43 # > 400 processes.

45 define service{
       use                         generic-service      ; Name of service template to use
47     host_name                   nagios
       service_description         Total Processes
49   check_command               check_procs!250!400
       }
51


53
   # Define a service to check the load on the local machine.
55
   define service{
57     use                         generic-service      ; Name of service template to use
       host_name                   nagios
59     service_description         Current Load
     check_command               check_load!5.0!4.0!3.0!10.0!6.0!4.0
61     }
```

A.1.3.5 Imprimante

```
1  define host{
       use                 generic-host          ; Name of host template$
3      host_name           printer
       alias               Imprimante A408
5      display_name        Imprimante A408
       address             10.1.0.22
7      parents             clavister
       }
```

A.1.3.6 Routeur UNIGE

```
1  define host{
       use                 generic-host          ; Name of host template$
3      host_name           router_unige
```

```
        alias                   Routeur UNIGE
        display_name            Routeur UNIGE
        address                 129.194.184.1
        }
```

A.1.3.7 Serveur Web

```
define host{
        use                     generic-host          ; Name of host template$
        host_name               web
        alias                   Serveur Web
        display_name            Serveur Web
        address                 129.194.184.80
    parents       clavister
        }
```

A.1.4 Groupes d'hôtes

```
# Some generic hostgroup definitions

# A simple wildcard hostgroup
define hostgroup {
        hostgroup_name all
    alias          Tous les equipements et serveurs
    members        *
        }

define hostgroup {
        hostgroup_name lan
    alias          Equipements du reseau LAN du labo
    members        nagios, fileserver1, fileserver2, clavister, printer
        }

define hostgroup {
    hostgroup_name  dmz
    alias   DMZ
    members    web, dns
    }

define hostgroup {
        hostgroup_name unige
        alias          Reseau UNIGE
        members        router_unige
        }

# A list of your web servers
#define hostgroup {
#       hostgroup_name http-servers
#  alias          HTTP servers
#  members        localhost
#       }

#define hostgroup {
```

```
#       hostgroup_name ssh-servers
#  alias          SSH servers
#  members        localhost
#       }

# nagios doesn't like monitoring hosts without services, so this is
# a group for devices that have no other "services" monitorable
# (like routers w/out snmp for example)
#define hostgroup {
#       hostgroup_name ping-servers
#  alias          Test du ping pour tous
#  members        *
#       }
```

A.1.5 Services

```
# check all is alive
#define service {
#  hostgroup_name  all
#  service_description ISALIVE
#  check_command   check-router-alive_4
#  use        generic-service
#  }


# check that web services are running
#define service {
#       hostgroup_name              http-servers
#       service_description         HTTP
#  check_command               check_http
#       use                         generic-service
#  notification_interval       0 ; set > 0 if you want to be renotified
#}

# check that ssh services are running
#define service {
#       hostgroup_name              ssh-servers
#       service_description         SSH
#  check_command               check_ssh
#       use                         generic-service
#  notification_interval       0 ; set > 0 if you want to be renotified
#}

# check that ping-only hosts are up
#define service {
#       hostgroup_name              ping-servers
#       service_description         PING
#  check_command               check_ping!100.0,20%!500.0,60%
#       use                         generic-service
#  notification_interval       0 ; set > 0 if you want to be renotified
#}
```

A.2 Exemple de greffon Nagios

```
/*****************************************************************************
*
* Nagios check_ping plugin
*
* License: GPL
* Copyright (c) 2000-2007 Nagios Plugins Development Team
*
* Description:
*
* This file contains the check_ping plugin
*
* Use the ping program to check connection statistics for a remote host.
*
*
* This program is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program. If not, see <http://www.gnu.org/licenses/>.
*
*
*****************************************************************************/

const char *progname = "check_ping";
const char *copyright = "2000-2007";
const char *email = "nagiosplug-devel@lists.sourceforge.net";

#include "common.h"
#include "netutils.h"
#include "popen.h"
#include "utils.h"

#define WARN_DUPLICATES "DUPLICATES FOUND! "
#define UNKNOWN_TRIP_TIME -1.0 /* -1 seconds */

enum {
   UNKNOWN_PACKET_LOSS = 200, /* 200% */
   DEFAULT_MAX_PACKETS = 5    /* default no. of ICMP ECHO packets */
};

int process_arguments (int, char **);
int get_threshold (char *, float *, int *);
int validate_arguments (void);
int run_ping (const char *cmd, const char *addr);
int error_scan (char buf[MAX_INPUT_BUFFER], const char *addr);
void print_usage (void);
```

```
void print_help (void);

int display_html = FALSE;
int wpl = UNKNOWN_PACKET_LOSS;
int cpl = UNKNOWN_PACKET_LOSS;
float wrta = UNKNOWN_TRIP_TIME;
float crta = UNKNOWN_TRIP_TIME;
char **addresses = NULL;
int n_addresses = 0;
int max_addr = 1;
int max_packets = -1;
int verbose = 0;

float rta = UNKNOWN_TRIP_TIME;
int pl = UNKNOWN_PACKET_LOSS;

char *warn_text;



int
main (int argc, char **argv)
{
   char *cmd = NULL;
   char *rawcmd = NULL;
   int result = STATE_UNKNOWN;
   int this_result = STATE_UNKNOWN;
   int i;

   setlocale (LC_ALL, "");
   setlocale (LC_NUMERIC, "C");
   bindtextdomain (PACKAGE, LOCALEDIR);
   textdomain (PACKAGE);

   addresses = malloc (sizeof(char*) * max_addr);
   addresses[0] = NULL;

   /* Parse extra opts if any */
   argv=np_extra_opts (&argc, argv, progname);

   if (process_arguments (argc, argv) == ERROR)
      usage4 (_("Could not parse arguments"));

   /* Set signal handling and alarm */
   if (signal (SIGALRM, popen_timeout_alarm_handler) == SIG_ERR) {
      usage4 (_("Cannot catch SIGALRM"));
   }

   /* If ./configure finds ping has timeout values, set plugin alarm slightly
    * higher so that we can use response from command line ping */
#if defined(PING_PACKETS_FIRST) && defined(PING_HAS_TIMEOUT)
   alarm (timeout_interval + 1);
#else
   alarm (timeout_interval);
#endif

```

```
   for (i = 0 ; i < n_addresses ; i++) {
111
#ifdef PING6_COMMAND
113      if (address_family != AF_INET && is_inet6_addr(addresses[i]))
           rawcmd = strdup(PING6_COMMAND);
115      else
           rawcmd = strdup(PING_COMMAND);
117 #else
         rawcmd = strdup(PING_COMMAND);
119 #endif

121      /* does the host address of number of packets argument come first? */
#ifdef PING_PACKETS_FIRST
123 # ifdef PING_HAS_TIMEOUT
         asprintf (&cmd, rawcmd, timeout_interval, max_packets, addresses[i]);
125 # else
         asprintf (&cmd, rawcmd, max_packets, addresses[i]);
127 # endif
#else
129      asprintf (&cmd, rawcmd, addresses[i], max_packets);
#endif
131
         if (verbose >= 2)
133        printf ("CMD: %s\n", cmd);

135      /* run the command */
         this_result = run_ping (cmd, addresses[i]);
137
         if (pl == UNKNOWN_PACKET_LOSS || rta < 0.0) {
139        printf ("%s\n", cmd);
           die (STATE_UNKNOWN,
141                 _("CRITICAL - Could not interpret output from ping command\n"));
         }
143
         if (pl >= cpl || rta >= crta || rta < 0)
145        this_result = STATE_CRITICAL;
         else if (pl >= wpl || rta >= wrta)
147        this_result = STATE_WARNING;
         else if (pl >= 0 && rta >= 0)
149        this_result = max_state (STATE_OK, this_result);

151      if (n_addresses > 1 && this_result != STATE_UNKNOWN)
           die (STATE_OK, "%s is alive\n", addresses[i]);
153
         if (display_html == TRUE)
155        printf ("<A HREF='%s/traceroute.cgi?%s'>", CGIURL, addresses[i]);
         if (pl == 100)
157        printf (_("PING %s - %sPacket loss = %d%%"), state_text (this_result), warn_text,
                   pl);
159      else
           printf (_("PING %s - %sPacket loss = %d%%, RTA = %2.2f ms"),
161                state_text (this_result), warn_text, pl, rta);
         if (display_html == TRUE)
163        printf ("</A>");

165      /* Print performance data */
```

```
      printf("|%s", fperfdata ("rta", (double) rta, "ms",
167                            wrta>0?TRUE:FALSE, wrta,
                               crta>0?TRUE:FALSE, crta,
169                            TRUE, 0, FALSE, 0));
      printf(" %s\n", perfdata ("pl", (long) pl, "%",
171                            wpl>0?TRUE:FALSE, wpl,
                               cpl>0?TRUE:FALSE, cpl,
173                            TRUE, 0, FALSE, 0));

175      if (verbose >= 2)
           printf ("%f:%d%% %f:%d%%\n", wrta, wpl, crta, cpl);
177
         result = max_state (result, this_result);
179      free (rawcmd);
         free (cmd);
181   }

183   return result;
}
185


187
/* process command-line arguments */
189 int
process_arguments (int argc, char **argv)
191 {
   int c = 1;
193   char *ptr;

195   int option = 0;
   static struct option longopts[] = {
197      STD_LONG_OPTS,
      {"packets", required_argument, 0, 'p'},
199      {"nohtml", no_argument, 0, 'n'},
      {"link", no_argument, 0, 'L'},
201      {"use-ipv4", no_argument, 0, '4'},
      {"use-ipv6", no_argument, 0, '6'},
203      {0, 0, 0, 0}
   };
205
   if (argc < 2)
207      return ERROR;

209   for (c = 1; c < argc; c++) {
      if (strcmp ("-to", argv[c]) == 0)
211         strcpy (argv[c], "-t");
      if (strcmp ("-nohtml", argv[c]) == 0)
213         strcpy (argv[c], "-n");
   }
215
   while (1) {
217      c = getopt_long (argc, argv, "VvhnL46t:c:w:H:p:", longopts, &option);

219      if (c == -1 || c == EOF)
         break;
221
```

```
     switch (c) {
223  case '?':  /* usage */
       usage5 ();
225  case 'h':  /* help */
       print_help ();
227    exit (STATE_OK);
       break;
229  case 'V':  /* version */
       print_revision (progname, NP_VERSION);
231    exit (STATE_OK);
       break;
233  case 't':  /* timeout period */
       timeout_interval = atoi (optarg);
235    break;
     case 'v':  /* verbose mode */
237    verbose++;
       break;
239  case '4':  /* IPv4 only */
       address_family = AF_INET;
241    break;
     case '6':  /* IPv6 only */
243 #ifdef USE_IPV6
       address_family = AF_INET6;
245 #else
       usage (_("IPv6 support not available\n"));
247 #endif
       break;
249  case 'H':  /* hostname */
       ptr=optarg;
251    while (1) {
         n_addresses++;
253      if (n_addresses > max_addr) {
           max_addr *= 2;
255        addresses = realloc (addresses, sizeof(char*) * max_addr);
           if (addresses == NULL)
257          die (STATE_UNKNOWN, _("Could not realloc() addresses\n"));
         }
259      addresses[n_addresses-1] = ptr;
         if ((ptr = index (ptr, ','))) {
261        strcpy (ptr, "");
           ptr += sizeof(char);
263      } else {
           break;
265      }
       }
267    break;
     case 'p':  /* number of packets to send */
269    if (is_intnonneg (optarg))
         max_packets = atoi (optarg);
271    else
         usage2 (_("<max_packets> (%s) must be a non-negative number\n"), optarg);
273    break;
     case 'n':  /* no HTML */
275    display_html = FALSE;
       break;
277  case 'L':  /* show HTML */
```

```
            display_html = TRUE;
279         break;
          case 'c':
281         get_threshold (optarg, &crta, &cpl);
            break;
283       case 'w':
            get_threshold (optarg, &wrta, &wpl);
285         break;
          }
287     }

289     c = optind;
        if (c == argc)
291       return validate_arguments ();

293     if (addresses[0] == NULL) {
          if (is_host (argv[c]) == FALSE) {
295         usage2 (_("Invalid hostname/address"), argv[c]);
          } else {
297         addresses[0] = argv[c++];
            n_addresses++;
299         if (c == argc)
              return validate_arguments ();
301       }
        }
303
        if (wpl == UNKNOWN_PACKET_LOSS) {
305       if (is_intpercent (argv[c]) == FALSE) {
            printf (_("<wpl> (%s) must be an integer percentage\n"), argv[c]);
307         return ERROR;
          } else {
309         wpl = atoi (argv[c++]);
            if (c == argc)
311           return validate_arguments ();
          }
313     }

315     if (cpl == UNKNOWN_PACKET_LOSS) {
          if (is_intpercent (argv[c]) == FALSE) {
317         printf (_("<cpl> (%s) must be an integer percentage\n"), argv[c]);
            return ERROR;
319       } else {
            cpl = atoi (argv[c++]);
321         if (c == argc)
              return validate_arguments ();
323       }
        }
325
        if (wrta < 0.0) {
327       if (is_negative (argv[c])) {
            printf (_("<wrta> (%s) must be a non-negative number\n"), argv[c]);
329         return ERROR;
          } else {
331         wrta = atof (argv[c++]);
            if (c == argc)
333           return validate_arguments ();
```

```
      }
335   }

337   if (crta < 0.0) {
         if (is_negative (argv[c])) {
339         printf (_("<crta> (%s) must be a non-negative number\n"), argv[c]);
            return ERROR;
341      } else {
            crta = atof (argv[c++]);
343         if (c == argc)
               return validate_arguments ();
345      }
      }
347
      if (max_packets == -1) {
349      if (is_intnonneg (argv[c])) {
            max_packets = atoi (argv[c++]);
351      } else {
            printf (_("<max_packets> (%s) must be a non-negative number\n"), argv[c]);
353         return ERROR;
         }
355   }

357   return validate_arguments ();
}
359


361
int
363 get_threshold (char *arg, float *trta, int *tpl)
{
365   if (is_intnonneg (arg) && sscanf (arg, "%f", trta) == 1)
         return OK;
367   else if (strpbrk (arg, ",:") && strstr (arg, "%") && sscanf (arg, "%f%*[:,]%d%%", trta,
          tpl) == 2)
         return OK;
369   else if (strstr (arg, "%") && sscanf (arg, "%d%%", tpl) == 1)
         return OK;
371
      usage2 (_("%s: Warning threshold must be integer or percentage!\n\n"), arg);
373   return STATE_UNKNOWN;
}
375


377
int
379 validate_arguments ()
{
381   float max_seconds;
      int i;
383
      if (wrta < 0.0) {
385      printf (_("<wrta> was not set\n"));
         return ERROR;
387   }
      else if (crta < 0.0) {
```

```
389      printf (_("<crta> was not set\n"));
         return ERROR;
391   }
      else if (wpl == UNKNOWN_PACKET_LOSS) {
393      printf (_("<wpl> was not set\n"));
         return ERROR;
395   }
      else if (cpl == UNKNOWN_PACKET_LOSS) {
397      printf (_("<cpl> was not set\n"));
         return ERROR;
399   }
      else if (wrta > crta) {
401      printf (_("<wrta> (%f) cannot be larger than <crta> (%f)\n"), wrta, crta);
         return ERROR;
403   }
      else if (wpl > cpl) {
405      printf (_("<wpl> (%d) cannot be larger than <cpl> (%d)\n"), wpl, cpl);
         return ERROR;
407   }

409   if (max_packets == -1)
         max_packets = DEFAULT_MAX_PACKETS;
411
      max_seconds = crta / 1000.0 * max_packets + max_packets;
413   if (max_seconds > timeout_interval)
         timeout_interval = (int)max_seconds;
415
      for (i=0; i<n_addresses; i++) {
417      if (is_host(addresses[i]) == FALSE)
            usage2 (_("Invalid hostname/address"), addresses[i]);
419   }

421   if (n_addresses == 0) {
         usage (_("You must specify a server address or host name"));
423   }

425   return OK;
}
427


429

int
431 run_ping (const char *cmd, const char *addr)
{
433   char buf[MAX_INPUT_BUFFER];
      int result = STATE_UNKNOWN;
435
      if ((child_process = spopen (cmd)) == NULL)
437      die (STATE_UNKNOWN, _("Could not open pipe: %s\n"), cmd);

439   child_stderr = fdopen (child_stderr_array[fileno (child_process)], "r");
      if (child_stderr == NULL)
441      printf (_("Cannot open stderr for %s\n"), cmd);

443   while (fgets (buf, MAX_INPUT_BUFFER - 1, child_process)) {
```

```
445      if (verbose >= 3)
           printf("Output: %s", buf);
447
         result = max_state (result, error_scan (buf, addr));
449
         /* get the percent loss statistics */
451      if(sscanf(buf,"%*d packets transmitted, %*d packets received, +%*d errors, %d%% packet
             loss",&pl)==1 ||
           sscanf(buf,"%*d packets transmitted, %*d packets received, +%*d duplicates, %d%%
               packet loss", &pl) == 1 ||
453        sscanf(buf,"%*d packets transmitted, %*d received, +%*d duplicates, %d%% packet
               loss", &pl) == 1 ||
           sscanf(buf,"%*d packets transmitted, %*d packets received, %d%% packet loss",&pl)
               ==1 ||
455        sscanf(buf,"%*d packets transmitted, %*d packets received, %d%% loss, time",&pl)==1
                ||
           sscanf(buf,"%*d packets transmitted, %*d received, %d%% loss, time", &pl)==1 ||
457        sscanf(buf,"%*d packets transmitted, %*d received, %d%% packet loss, time", &pl)==1
                ||
           sscanf(buf,"%*d packets transmitted, %*d received, +%*d errors, %d%% packet loss",
               &pl) == 1 ||
459        sscanf(buf,"%*d packets transmitted %*d received, +%*d errors, %d%% packet loss", &
               pl) == 1
           )
461        continue;

463      /* get the round trip average */
         else
465        if(sscanf(buf,"round-trip min/avg/max = %*f/%f/%*f",&rta)==1 ||
               sscanf(buf,"round-trip min/avg/max/mdev = %*f/%f/%*f/%*f",&rta)==1 ||
467            sscanf(buf,"round-trip min/avg/max/sdev = %*f/%f/%*f/%*f",&rta)==1 ||
               sscanf(buf,"round-trip min/avg/max/stddev = %*f/%f/%*f/%*f",&rta)==1 ||
469            sscanf(buf,"round-trip min/avg/max/std-dev = %*f/%f/%*f/%*f",&rta)==1 ||
               sscanf(buf,"round-trip (ms) min/avg/max = %*f/%f/%*f",&rta)==1 ||
471            sscanf(buf,"round-trip (ms) min/avg/max/stddev = %*f/%f/%*f/%*f",&rta)==1 ||
               sscanf(buf,"rtt min/avg/max/mdev = %*f/%f/%*f/%*f ms",&rta)==1)
473        continue;
     }
475
     /* this is needed because there is no rta if all packets are lost */
477  if (pl == 100)
       rta = crta;
479
     /* check stderr, setting at least WARNING if there is output here */
481  /* Add warning into warn_text */
     while (fgets (buf, MAX_INPUT_BUFFER - 1, child_stderr)) {
483    if (! strstr(buf,"WARNING - no SO_TIMESTAMP support, falling back to SIOCGSTAMP")) {
         if (verbose >= 3) {
485        printf("Got stderr: %s", buf);
         }
487      if ((result=error_scan(buf, addr)) == STATE_OK) {
           result = STATE_WARNING;
489        if (warn_text == NULL) {
             warn_text = strdup(_("System call sent warnings to stderr "));
491        } else {
```

```
                asprintf(&warn_text, "%s %s", warn_text, _("System call sent warnings to
                    stderr "));
493           }
            }
495       }
    }
497
    (void) fclose (child_stderr);
499

501 /* close the pipe - WARNING if status is set */
    if (spclose (child_process))
503   result = max_state (result, STATE_WARNING);

505 if (warn_text == NULL)
      warn_text = strdup("");
507
    return result;
509 }


511


513 int
    error_scan (char buf[MAX_INPUT_BUFFER], const char *addr)
515 {
    if (strstr (buf, "Network is unreachable") ||
517   strstr (buf, "Destination Net Unreachable")
      )
519   die (STATE_CRITICAL, _("CRITICAL - Network Unreachable (%s)"), addr);
    else if (strstr (buf, "Destination Host Unreachable"))
521   die (STATE_CRITICAL, _("CRITICAL - Host Unreachable (%s)"), addr);
    else if (strstr (buf, "Destination Port Unreachable"))
523   die (STATE_CRITICAL, _("CRITICAL - Bogus ICMP: Port Unreachable (%s)"), addr);
    else if (strstr (buf, "Destination Protocol Unreachable"))
525   die (STATE_CRITICAL, _("CRITICAL - Bogus ICMP: Protocol Unreachable (%s)"), addr);
    else if (strstr (buf, "Destination Net Prohibited"))
527   die (STATE_CRITICAL, _("CRITICAL - Network Prohibited (%s)"), addr);
    else if (strstr (buf, "Destination Host Prohibited"))
529   die (STATE_CRITICAL, _("CRITICAL - Host Prohibited (%s)"), addr);
    else if (strstr (buf, "Packet filtered"))
531   die (STATE_CRITICAL, _("CRITICAL - Packet Filtered (%s)"), addr);
    else if (strstr (buf, "unknown host" ))
533   die (STATE_CRITICAL, _("CRITICAL - Host not found (%s)"), addr);
    else if (strstr (buf, "Time to live exceeded"))
535   die (STATE_CRITICAL, _("CRITICAL - Time to live exceeded (%s)"), addr);

537 if (strstr (buf, "(DUP!)") || strstr (buf, "DUPLICATES FOUND")) {
      if (warn_text == NULL)
539     warn_text = strdup (_(WARN_DUPLICATES));
      else if (! strstr (warn_text, _(WARN_DUPLICATES)) &&
541             asprintf (&warn_text, "%s %s", warn_text, _(WARN_DUPLICATES)) == -1)
        die (STATE_UNKNOWN, _("Unable to realloc warn_text"));
543   return (STATE_WARNING);
    }
545
    return (STATE_OK);
```

```
547 }
549
551 void
    print_help (void)
553 {
      print_revision (progname, NP_VERSION);
555
      printf ("Copyright (c) 1999 Ethan Galstad <nagios@nagios.org>\n");
557   printf (COPYRIGHT, copyright, email);

559   printf (_("Use ping to check connection statistics for a remote host."));

561  printf ("\n\n");

563   print_usage ();

565   printf (_(UT_HELP_VRSN));
      printf (_(UT_EXTRA_OPTS));
567
      printf (_(UT_IPv46));
569
      printf (" %s\n", "-H, --hostname=HOST");
571  printf ("   %s\n", _("host to ping"));
     printf (" %s\n", "-w, --warning=THRESHOLD");
573  printf ("   %s\n", _("warning threshold pair"));
     printf (" %s\n", "-c, --critical=THRESHOLD");
575  printf ("   %s\n", _("critical threshold pair"));
     printf (" %s\n", "-p, --packets=INTEGER");
577  printf ("   %s ", _("number of ICMP ECHO packets to send"));
     printf (_("(Default: %d)\n"), DEFAULT_MAX_PACKETS);
579  printf (" %s\n", "-L, --link");
     printf ("   %s\n", _("show HTML in the plugin output (obsoleted by urlize)"));
581
      printf (_(UT_TIMEOUT), DEFAULT_SOCKET_TIMEOUT);
583
     printf ("\n");
585   printf ("%s\n", _("THRESHOLD is <rta>,<pl>% where <rta> is the round trip average travel
          "));
     printf ("%s\n", _("time (ms) which triggers a WARNING or CRITICAL state, and <pl> is the")
         );
587  printf ("%s\n", _("percentage of packet loss to trigger an alarm state."));

589  printf ("\n");
      printf ("%s\n", _("This plugin uses the ping command to probe the specified host for
          packet loss"));
591  printf ("%s\n", _("(percentage) and round trip average (milliseconds). It can produce HTML
          output"));
     printf ("%s\n", _("linking to a traceroute CGI contributed by Ian Cass. The CGI can be
         found in"));
593  printf ("%s\n", _("the contrib area of the downloads section at http://www.nagios.org/"));

595 #ifdef NP_EXTRA_OPTS
     printf ("\n");
597  printf ("%s\n", _("Notes:"));
```

```
  printf (_(UT_EXTRA_OPTS_NOTES));
599 #endif

601    printf (_(UT_SUPPORT));
}
603
void
605 print_usage (void)
{
607   printf (_("Usage:"));
    printf ("%s -H <host_address> -w <wrta>,<wpl>%% -c <crta>,<cpl>%%\n", progname);
609   printf (" [-p packets] [-t timeout] [-4|-6]\n");
}
```