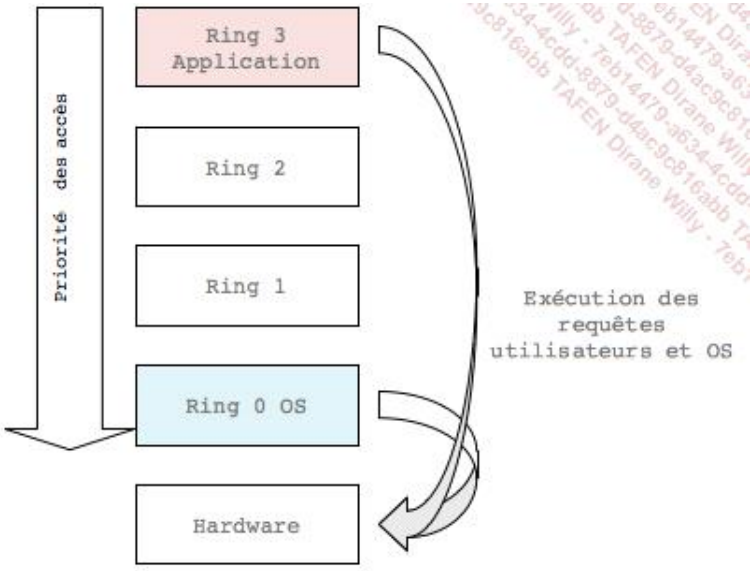
CPU

1. Le CPU

Avant toute chose, quelques informations sur le fonctionnement d'un processeur (CPU) x86/x86_64 sont nécessaires. L'architecture x86 fournit quatre niveaux (qu'on appelle anneaux ou rings) de privilèges pour les accès au CPU. Au ring 3, nous trouvons les applications. Ce niveau équivaut aux privilèges minimums. Au ring 2 et 1, nous retrouvons les pilotes des périphériques physiques. Le ring 0 correspond au noyau du système.

Les pilotes d'impression qui généraient des écrans bleus sous Windows NT et 2000, se trouvaient dans le ring 1, ce qui provoquait beaucoup d'instabilités.



Dans le monde de la virtualisation, nous appliquons trois méthodes pour exploiter le CPU physique via des machines virtuelles :

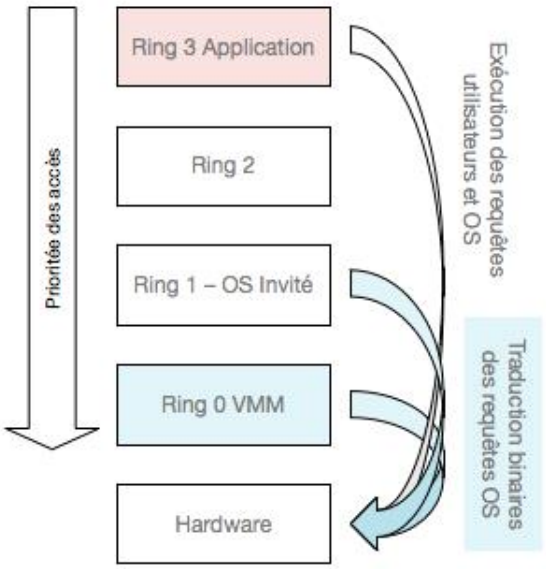
- Traduction binaire (*Binary Translation* - BT)
- Paravirtualisation
- Virtualisation « matérielle » (Hardware-assisted Virtualization)

a. Traduction binaire (Binary Translation - BT)

La traduction binaire autorise le VMM à fonctionner au niveau du ring 0 (avec le plus de droits). Cela implique que pour chaque OS, une table de traduction existe, et permet de capturer les interruptions système qui ont besoin de plus de privilèges. Lors de la création d'une machine virtuelle, nous sélectionnons un modèle d'OS, cette action sélectionne la table de traduction binaire spécifique à cet OS.

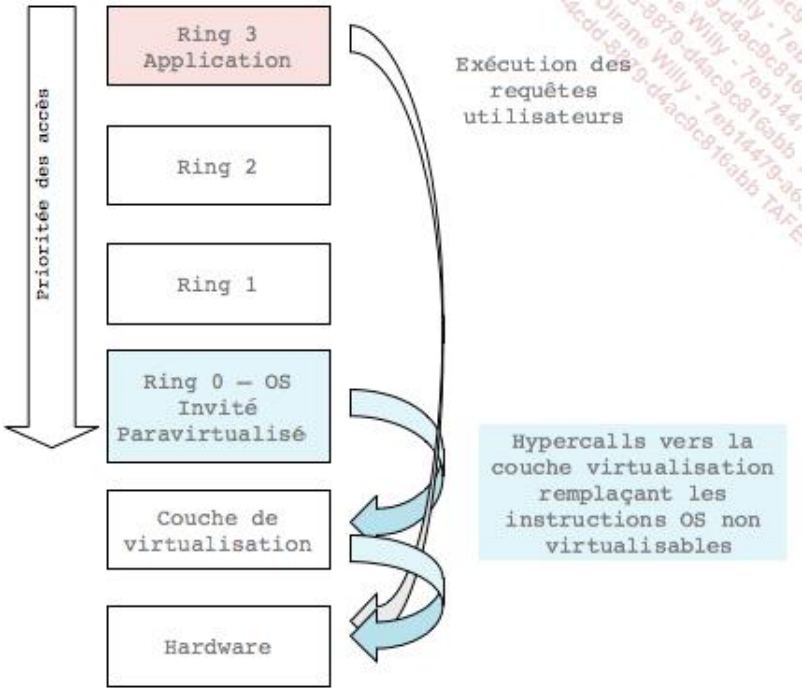
Cette technique est purement logicielle et assez ancienne. Elle est même nommée « émulation de CPU » par (au moins) un éditeur concurrent. C'est une technique de virtualisation dite d'ancienne génération et bien qu'elle ait été améliorée au fil des versions des hyperviseurs ESX et ESXi, l'overhead y reste important.

De fait il n'est pas étonnant que VMware ait annoncé que vSphere 6.5 soit la dernière version à supporter la technique de virtualisation par traduction binaire (https://kb.vmware.com/kb/2147608).



b. La paravirtualisation

La paravirtualisation laisse prendre conscience au système présent dans la machine virtuelle qu'il est virtualisé. Les instructions de type ring 0 ne sont pas envoyées au matériel telles quelles. Le kernel invité (modifié) envoie directement des instructions modifiées afin que l'hyperviseur puisse les traiter. Cela nécessite une modification du kernel du système d'exploitation invité. C'est le cas pour certaines distributions Linux virtualisées sur du Xen.



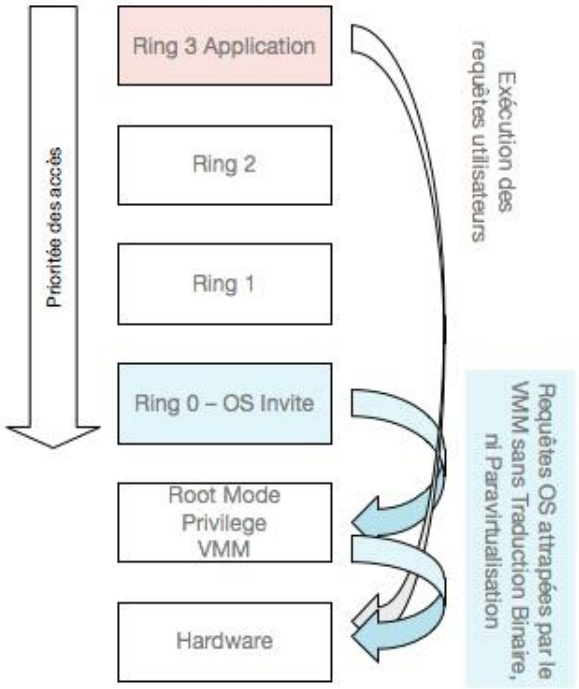
En ce qui concerne les hypercalls, on peut considérer que cette technique est appliquée de manière partielle pour les pilotes du matériel virtuel présenté au système d'exploitation invité. Ainsi quand vous installez les VMware Tools après installation du système, vous avez sûrement remarqué que les performances étaient bien meilleures sur presque tous les points.

Ceci est dû au fait que ces fameux tools incluent des pilotes performants permettant un dialogue plus direct avec les éléments matériels vus par l'hyperviseur. VMware les appelle « pilotes paravirtualisés » : ils diminuent l'overhead au niveau réseau et stockage notamment.

De vue d'administrateur, on considère les tools comme obligatoires lors de la création d'une machine virtuelle.

c. Virtualisation « matérielle » (Hardware-assisted Virtualization)

Les premières générations de CPU embarquant la virtualisation du matériel sont les CPU INTEL VT-x et AMD-V. On peut considérer qu'ils autorisent la VMM à fonctionner dans un Ring de niveau -1 qui se trouve directement sur le matériel. Les instructions privilégiées sont envoyées sans modifications et on peut considérer que c'est au CPU de comprendre qui l'envoie (système hôte - hyperviseur ou système invité - dans la machine virtuelle) afin de la traiter « correctement ». Ainsi le CPU est capable de gérer la priorité du kernel hôte sur le kernel invité, bien que les deux systèmes puissent considérer envoyer des instructions de ring 0. C'est pour cela qu'on parle de niveau -1 (sous le ring 0) pour le système hôte (l'hyperviseur).



On active les technologies de virtualisation dans le BIOS ou UEFI des cartes mères portant des processeurs Intel. Pour AMD ce n'est pas nécessaire car tout CPU supportant les technologies de virtualisation les ont d'activées par défaut.

2. Allocation

Une machine virtuelle peut être configurée avec 128 processeurs virtuels. Un serveur ESXi supporte jusqu'à 480 (576 sous vsphere 6.5) cœurs logiques. Il y a correspondance entre un cœur virtuel et un cœur logique au moment où la machine virtuelle est démarrée. Cependant, ce n'est pas du « un pour un ». L'attribution d'un vCPU (simple cœur) à une machine virtuelle revient à l'autoriser à utiliser un contexte d'exécution matériel (HEC pour *Hardware Execution Context*).

Évidemment, la plupart des machines virtuelles sont configurées avec SMP (*Symmetric Multi Processing*) et utilisent plusieurs HEC.

Attention au terme « symmetric » : il signifie uniquement que les contextes d'exécution sont alloués simultanément à la machine virtuelle. Il est possible d'allouer un nombre impair de contextes à une machine virtuelle !

Dans le cas où les prérequis applicatifs ne sont pas connus, il convient de commencer par utiliser le moins de HEC possible pour une machine virtuelle, puis d'augmenter en accord avec la demande en ressource des applications. Un design minimum est requis pour toute application.

Au moment de configurer une machine virtuelle avec plusieurs cœurs, certaines questions peuvent se poser. Par exemple, si la machine virtuelle vient d'une conversion à partir d'un serveur physique (P2V), on se demande souvent si la configuration matérielle doit être respectée.

Le serveur type suivant sera considéré : 24 Go de mémoire vive, un processeur quadricœur, 30 Go pour le système, une capacité de 100 Go pour les données d'applications et deux cartes réseau gigabit.

En ce qui concerne le processeur, si l'on veut simplement reproduire la configuration, doit-on considérer :

4 processeurs monocœurs (la configuration par défaut pour 4 HEC) :

2 processeurs bicœurs :

ou 1 processeur quadricœur ?

Remarquez que VMware appelle CPU (virtuels) les HEC, soit le nombre total de cœurs logiques de la machine virtuelle. Si vous changez le nombre de cœurs par socket, le nombre de sockets (à droite) varie en fonction.

Au-delà de la partie licence (nombre de processeurs physiques licenciés sur l'hyperviseur, nombre de processeurs licenciés au niveau du système d'exploitation invité), la réponse doit être liée aux applications. Pour un même nombre de HEC, il peut être plus avantageux financièrement d'avoir un nombre réduit de sockets.

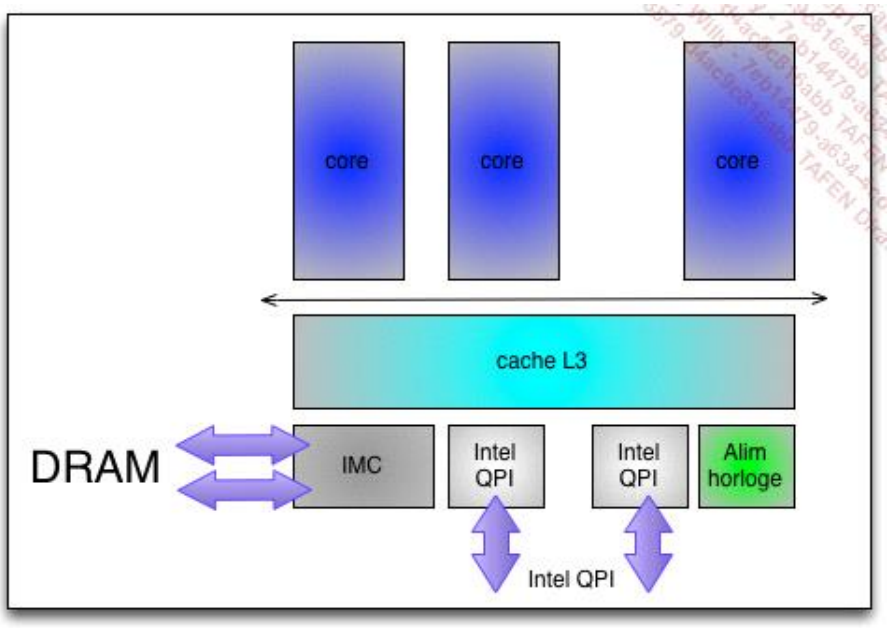
Pour revenir à une partie plus technique, nous descendons au niveau d'une application fonctionnant dans une machine physique ; si l'application est multithreadée, il y a de grandes chances que des instructions de même nature soient régulièrement envoyées aux processeurs. Dans ce cas on tire avantage des mémoires caches des processeurs car les données nécessaires aux calculs sont accessibles très rapidement (cache-hit). Dans le cas où la donnée n'est pas accessible dans le cache (cache-miss), la donnée est demandée au niveau de la mémoire vive et dupliquée dans la mémoire cache pour les prochaines demandes.

Les deux principes suivants permettent d'établir que les mémoires caches augmentent significativement les performances des processeurs :

Principe de localité spatiale : pour une donnée accédée située à une adresse donnée, la suivante sera certainement à une adresse proche.

Principe de localité temporelle : un accès à une zone mémoire donnée risque fortement de se reproduire dans la suite immédiate d'un programme.

Il existe plusieurs niveaux de cache, en général de niveau 1 (L1) au niveau 3 (L3) du plus rapide (et plus restreint) au plus lent (et plus conséquent). Ces mémoires sont très rapides et leur quantité est liée au niveau de performance et au prix du processeur.



En général les caches L1 et L2 sont propres au cœur, et le cache L3 est commun aux cœurs du processeur (socket). Pour certains processeurs, Intel a même créé un cache L4 (les « Crystalwell », série spéciale de la génération Broadwell).

Si nous revenons à l'hyperviseur, faire correspondre « l'architecture » virtuelle avec l'architecture physique permettra de faire une utilisation efficace des caches processeurs. Pour plus de performance, la machine virtuelle sera configurée pour correspondre au nœud NUMA sur laquelle elle se trouvera : si un nœud NUMA est constitué de 64 Go de mémoire vive et un processeur quadricœur (ce qui fait en général 8 processeurs logiques si on prend en compte l'hyperthreading), on configurera la machine virtuelle avec un processeur quadricœur aussi afin d'indiquer à l'hyperviseur d'utiliser les caches CPU.

3. Réservation et limite

La réservation de ressources se fait en temps CPU, en nombre de cycles d'instructions traités par secondes et donc en (méga) Hertz. La valeur configurée représente la quantité de ressources garantie à la VM. Au démarrage d'une VM pour laquelle une réservation est configurée, le contrôle d'admission se fait : il s'agit de vérifier que la ressource garantie est bien disponible. Si oui, la machine virtuelle démarre et utilise de fait sa réservation. Sinon la machine virtuelle ne peut démarrer. Il est possible de forcer la machine à démarrer alors que le contrôle d'admission échoue dans le cas d'un cluster HA qui aurait perdu un serveur hôte.

La limite représente la quantité de ressources maximale que pourra utiliser la machine virtuelle en toute circonstance. C'est prévu pour protéger le serveur hôte d'éventuelles défaillances applicatives internes aux VM pouvant conduire à réclamer toute ressource disponible au niveau du serveur ESXi.

Par défaut, aucune limite n'est configurée.

4. Shares

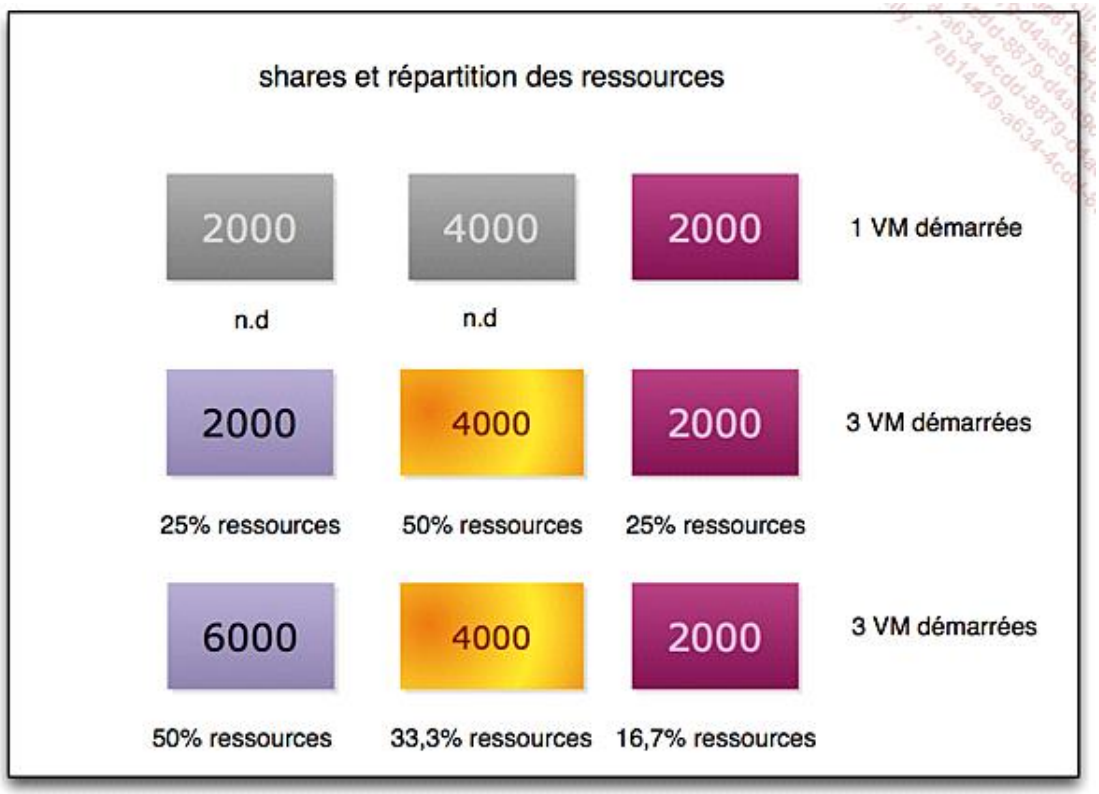
Il s'agit d'un poids relatif servant à la distribution des ressources vCompute (ici les shares processeur). Les shares sont aussi appelés « parts ».

Les shares sont valables sous deux conditions :

- Les machines virtuelles doivent être démarrées.
- Les ressources demandées par les machines virtuelles excèdent les ressources disponibles au niveau du serveur hôte.

Les shares sont attribués selon trois niveaux : low, medium, high (avec medium contenant deux fois plus de shares que low et high deux fois plus de shares que medium). Un niveau supplémentaire : custom, permet de personnaliser le niveau de priorité que l'on veut attribuer à une machine virtuelle. Par défaut le niveau normal est à 1 000 shares. On exprime le nombre de shares sans unité.

Voici un exemple d'évolution du pourcentage de ressources attribué à chaque VM selon le nombre de shares configuré :



Le pourcentage de ressources attribué est noté sur non disponible (n.d.) parce que tant que la VM concernée n'est pas démarrée, les shares n'ont aucune incidence sur les ressources distribuées à la VM. De plus, on voit qu'en première ligne une seule VM est démarrée, il n'y a donc pas de partage de ressources à effectuer avec d'autres VM.

Comment calculer le pourcentage (seulement valable à un moment donné) :

La valeur **a** est le nombre de shares attribués à une machine virtuelle, tandis que la valeur **b** est la somme des shares attribués à toutes les machines virtuelles démarrées au niveau d'un hyperviseur. Le calcul se fait en effectuant l'opération **a/b**.

Les shares comme la ressource allouée se configurent au niveau du matériel virtuel.

5. Optimisation

a. Le planificateur d'accès CPU

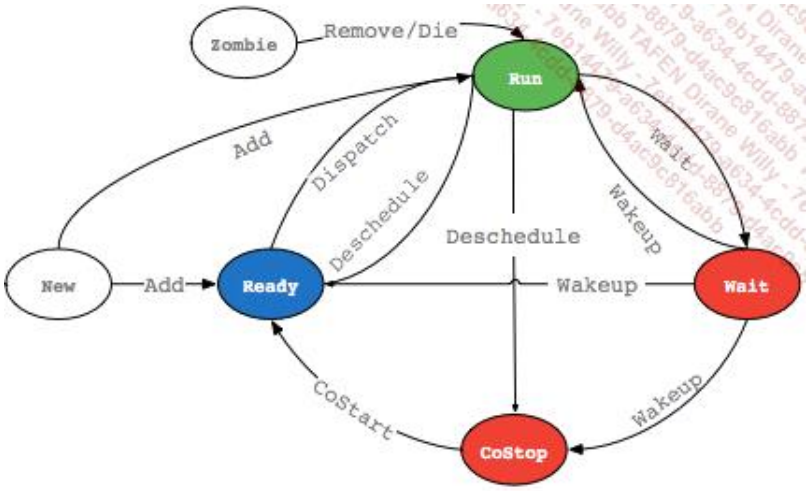
Dans le schéma d'introduction, dans la partie CPU, nous avons vu un composant du VMKernel nommé CPU Scheduler ou planificateur d'accès CPU. Son but est de fournir du temps [et donc l'accès] CPU aux machines virtuelles.

Dans le chapitre vCenter, nous avons vu qu'une machine virtuelle est un ensemble de fichiers exécuté par l'hyperviseur et que les processus correspondants se nomment « worlds ». Ces « worlds » représentent entre autres : le VMM lui-même, les clavier/vidéo/souris - le fameux KVM pour keyboard video mouse et surtout les fils - threads CPU).

Le planificateur d'accès CPU peut détecter et interpréter les relations entre les types de CPU, le nombre de sockets, de cœurs et des HEC/cœurs logiques. Ces relations sont utilisées pour planifier l'exécution des machines virtuelles, optimiser les performances et le placement des CPU virtuels (*Virtual CPU* - vCPU) sur les CPU/cœurs physiques (*physical CPU* - pCPU) et ainsi minimiser les migrations vCPU.

Un world peut avoir quatre états, après le démarrage d'une machine virtuelle (état Nouveau - New). En fonction de la disponibilité des pCPU, le world est soit dans un état prêt (Ready) soit dans un état exécution (Run). Un world dans un état Ready sera trié par le planificateur d'accès CPU afin d'être exécuté (Run).

Dans le cas où le planificateur décide de déplanifier un world, ce dernier aura un statut prêt (Ready) ou costop. Un world en costop peut évoluer vers le statut prêt via une action costart. Il est possible qu'un world entre dans l'état en attente (Wait) en bloquant/ attendant un accès à une ressource. Ce world, lorsqu'il aura eu un retour de la ressource sera soit en costop soit en Prêt, grâce à une action de réveil (Wakeup).



Il est possible de modifier le comportement du planificateur d'accès au CPU lorsque le serveur est dans une

situation de contention (dynamic entiltment) en utilisant :

- Les Réservations : le minimum garanti de CPU alloué à une machine virtuelle.
- Les Limites : maximum de CPU alloué à une machine virtuelle.
- Les Shares : la priorité relative d'une machine virtuelle en cas de contention CPU.

Pour information, voici ce que signifie chaque état :

- Prêt (Ready) : prêt à utiliser le CPU physique, est dans la file d'attente du planificateur.
- Attente (Wait): en attente d'utilisation d'une ressource VMKernel.
- Execution (Run) : utilisation du CPU physique.
- Costop (Costop) : période durant laquelle une VM multi-vCPU est prête à utiliser le CPU, mais ne peut pas suite à un conflit d'ordonnancement d'accès des multiples vCPU. C'est souvent le signe qu'on a donné trop de vCPU à une machine virtuelle par rapport au nombre de CPU logiques disponibles simultanément au niveau de la machine physique.

b. Planification simultanée stricte

La planification simultanée stricte ou Strict Co Scheduling nécessite que l'ensemble des vCPU soient planifiés et exécutés en même temps. C'est tout ou rien. Cela génère un temps d'attente pour les vCPU causant à la machine virtuelle des problèmes de lenteurs lors de bascule entre les états Co-stop ou Wait. Ce temps d'attente s'appelle le CPU Ready. Le fait d'observer du CPU ready pour une machine virtuelle est en général le signe que quelque chose ne se passe pas bien : la machine virtuelle peut manquer de ressources de type compute (CPU) ou être en attente. Il faut y prêter attention.

c. Planification simultanée souple

La planification simultanée souple ou Relaxing Co Scheduling a été introduite avec la version 4.0 d'ESXi. Il a remplacé le Strict Co Scheduling. Le Relaxing Co Scheduling permet de manipuler de manière plus souple les états de chaque vCPU. Lorsqu'un vCPU est de moins en moins sollicité (skew), et dépassant un certain seuil de décalage par rapport aux autres vCPU de la machine virtuelle, ce vCPU passe dans un état CoStop.

À cause du planificateur d'accès au CPU et de la consolidation de machine virtuelle avec différentes charges de travail, il se crée une surcharge au niveau CPU. Lorsqu'il y a trop de surcharge CPU et que le vCPU passe plus de temps à attendre un accès au pCPU, nous appelons cela le CPU Ready.

Le CPU Ready se calcule de la manière suivante :

% CPU Ready= (valeur de la somme CPU/(intervalle de temps du graphique (seconde) *1000)) * 100

Il faut prendre comme un signe d'alarme pour 1 vCPU à 1 core 5 % de CPU Ready. Lorsque l'on arrive à 10 % de CPU Ready, nous sommes en risque. Il existe une matrice permettant de connaître les seuils d'alarme et de risque sur le site vmtoday.com : http://vmtoday.com/2013/01/cpu-ready-revisted-quick-reference-charts/