

Mémoire

1. La mémoire

La mémoire physique au niveau de l'hôte ESXi est gérée par le VMKernel. Elle est partagée entre la mémoire qui est allouée au système afin d'effectuer ses propres tâches, et la mémoire qui est disponible pour les machines virtuelles.

Cette mémoire disponible pour les machines virtuelles est divisée en blocs que l'on nomme pages. Lorsque toute la mémoire allouée à une machine virtuelle est remplie, le VMkernel peut utiliser le(s) équipement(s) de stockages locaux ou distants. La taille d'un bloc mémoire est de 4 ko ou de 2 Mo.

a. Virtualisation de la mémoire

Lorsque nous parlons de la virtualisation de la mémoire, nous devons garder à l'esprit qu'il existe deux procédés pour cela :

- La virtualisation (logicielle) de la mémoire
- La virtualisation de la mémoire assistée par le matériel

Les CPU modernes embarquent des composants pour la virtualisation de la mémoire :

- L'unité de gestion de la mémoire (*Memory Management Unit* - MMU) qui fait correspondre les adresses de la mémoire des systèmes présents dans les machines virtuelles aux adresses physiques.
- Une mémoire tampon propre aux CPU (*Translation Look aside Buffer* - TLB) qui liste toutes les entrées connues afin d'accélérer les futurs accès.
- L'explorateur d'index mémoire (*Page Table Walker*) qui explore l'arborescence de la mémoire physique à chaque fois qu'il reçoit une nouvelle adresse mémoire d'un système présent dans une machine virtuelle. À la fin de chaque exploration, il fait l'association entre l'adresse virtuelle vue par la machine virtuelle et l'adresse physique vue par l'hyperviseur. Cette association est insérée dans le TLB. Lorsqu'une entrée est présente, on dit qu'il y a un TLB hit (touché), dans le cas contraire il y a un TLB miss.

b. Virtualisation de la mémoire basée sur du logiciel

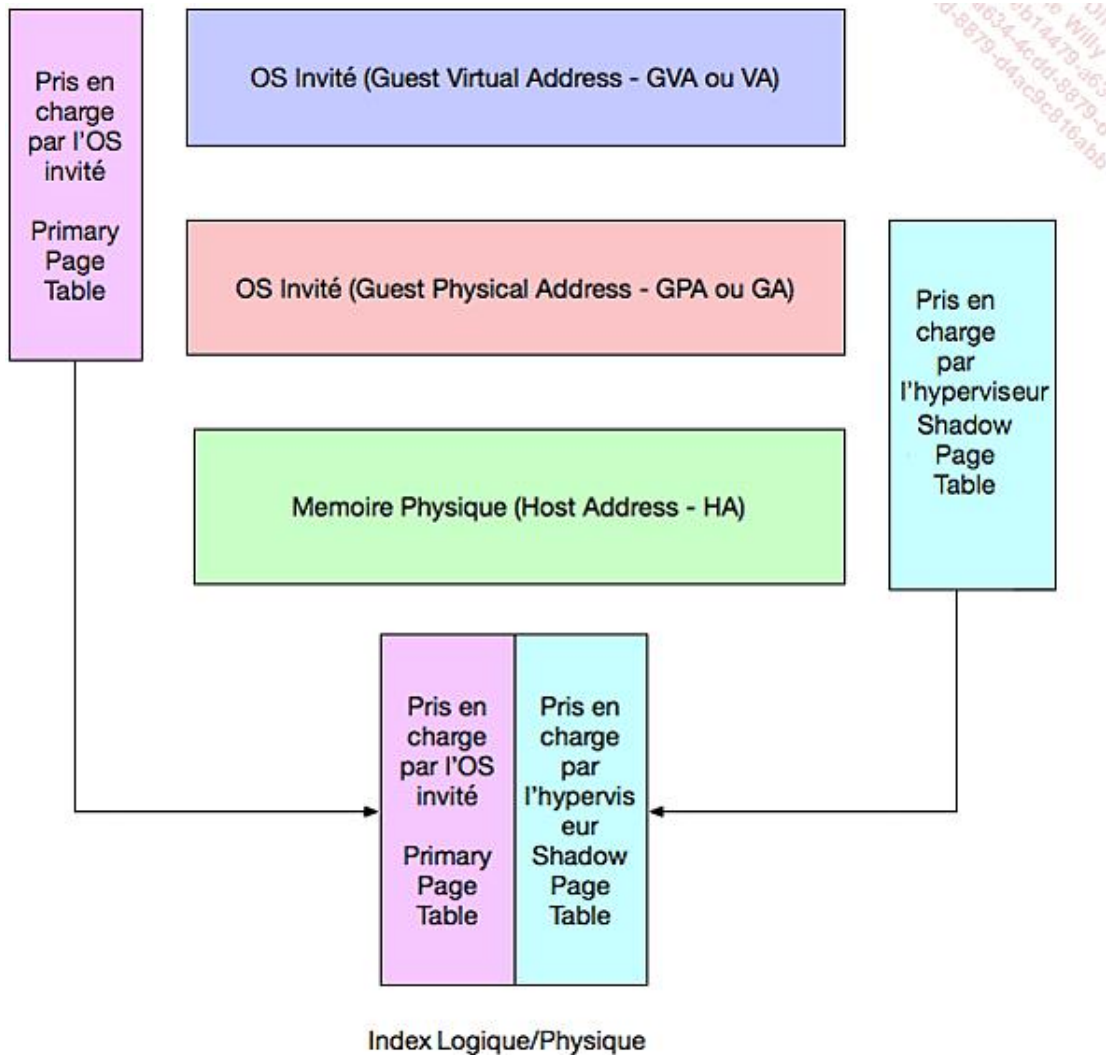
La virtualisation de mémoire basée sur du logiciel ou software based memory virtualization se base sur une version logicielle du MMU. Il est garant de l'association de la mémoire virtuelle entre les applications (*Virtual Memory* - VA) et l'OS (*Guest Address* - GA) et entre l'OS (GA) et l'hôte (*Host Address* - HA).

Un OS crée son propre index d'accès mémoire (*Primary Page Table*). Il est en écriture au niveau de l'OS de la machine virtuelle.

La VMM crée un index de page mémoire nommé Shadow Pages Table en se basant sur le Primary Page Table de l'OS :

- VA>GA : cette correspondance est propre à la machine virtuelle. Elle est obtenue en lisant la « primary page table » de la machine virtuelle.
- GA>HA : cette correspondance, faite par la VMM et le VMKernel, permet d'obtenir l'association entre l'adresse mémoire dans la machine virtuelle et l'adresse mémoire dans la mémoire physique.

Cette correspondance est connue sous le nom de correspondance logique/physique.



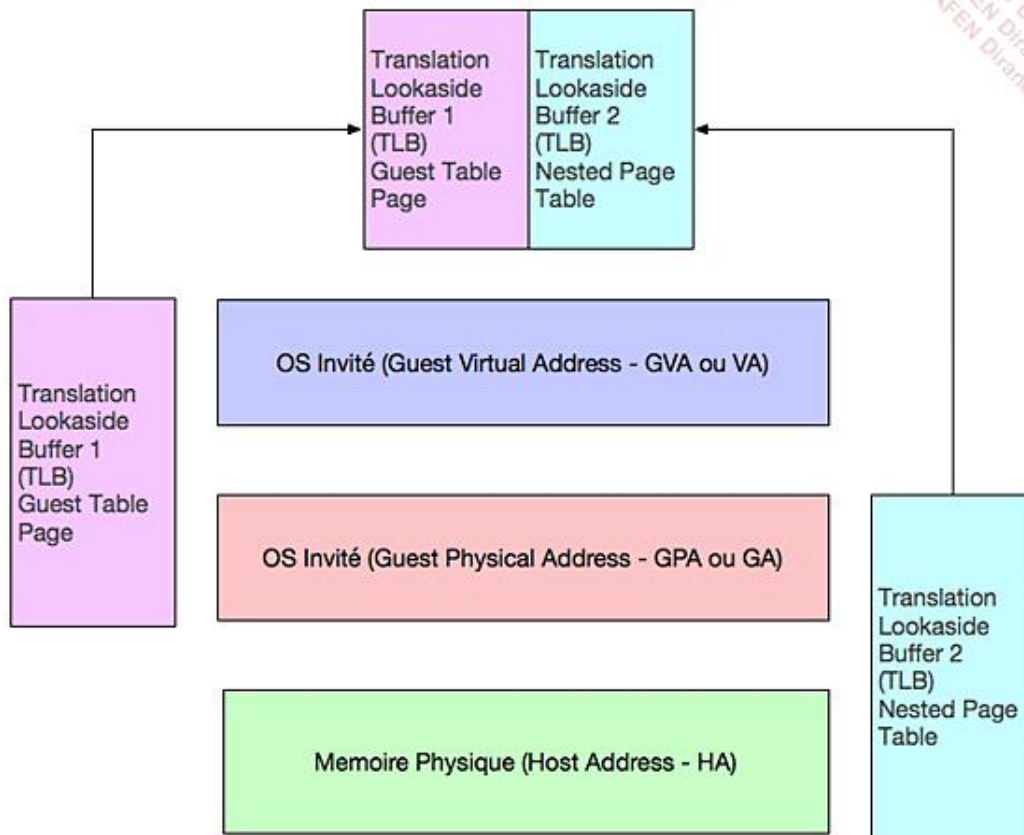
➤ Comme pour la virtualisation CPU par table de traduction binaire (technologie purement logicielle), la virtualisation de la mémoire par software MMU ne sera plus supportée dans les versions ultérieures à vSphere 6.5.

c. Virtualisation de la mémoire assistée par le matériel

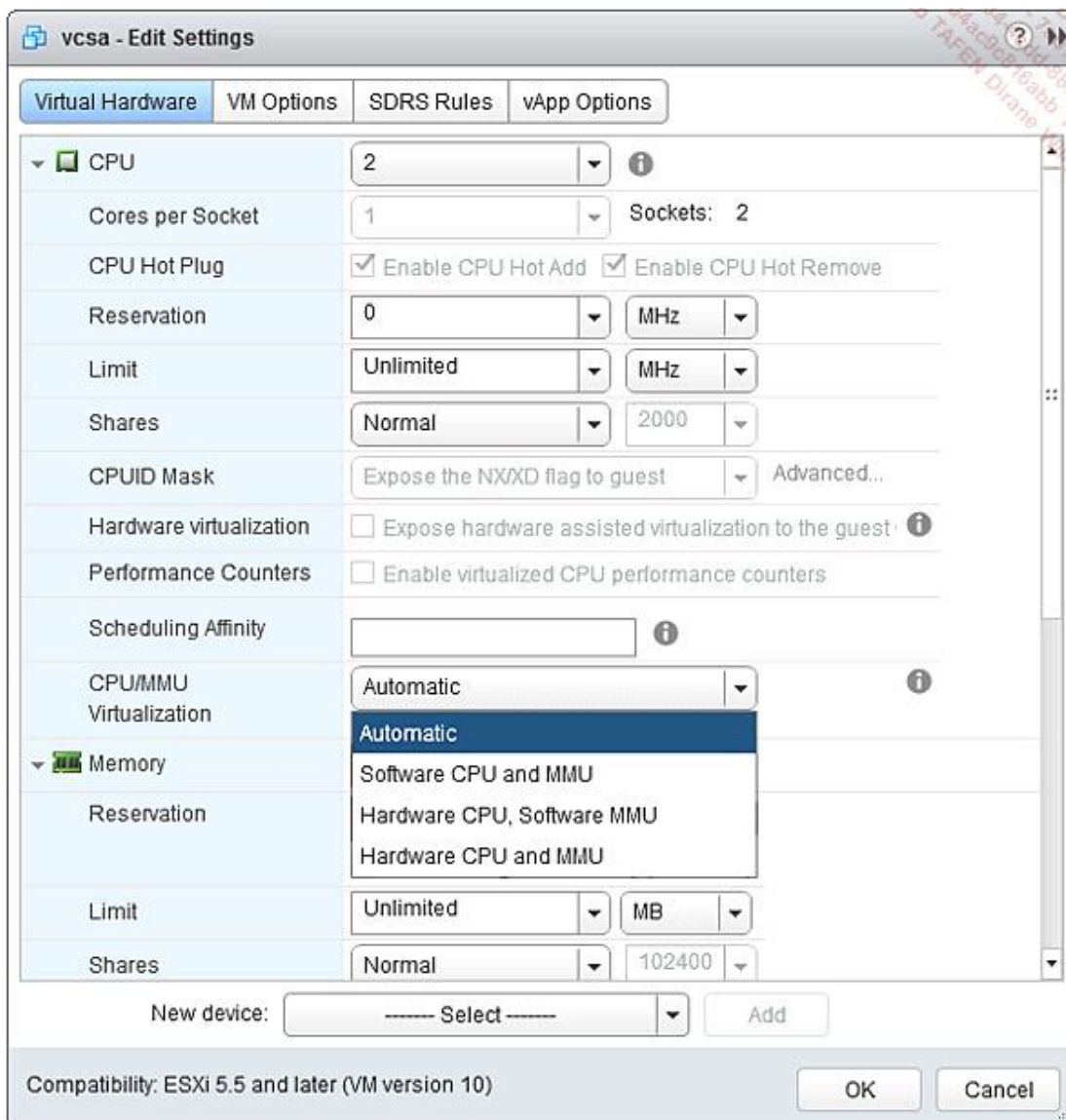
Les fournisseurs fournissent dans les CPU un MMU. Ce MMU est nommé chez INTEL Extended Page Table (EPT) et chez AMD Rapid Virtualization Indexing (RVI). Ils sont deux pointeurs d'index :

- Le Guest Page Table Pointer qui gère la correspondance au niveau de l'OS (VA>GA)
- Le Nested Page Table Pointer qui gère la correspondance au niveau de l'OS et de l'hôte. Ce dernier est maintenu via le VMM.

Ces deux pointeurs sont disponibles dans le matériel. Lorsqu'une adresse mémoire est accédée, le matériel explore ces deux pointeurs en parallèle.



La configuration de la gestion de la mémoire par MMU se fait machine virtuelle par machine virtuelle, au niveau du paramétrage du CPU. Il est nécessaire de bien connaître vos CPU physiques pour configurer ce paramétrage manuellement. Par défaut, il est configuré en automatique. C'est l'hyperviseur qui choisit quel type de MMU utiliser.



d. La surcharge liée à la mémoire

Quelle que soit la solution de virtualisation de la mémoire que l'on utilise, nous allons avoir de la surcharge CPU. Mais cette surcharge ne va pas se présenter au même moment.

Dans le cas de l'utilisation du MMU logiciel, cette surcharge se présente lors des actions suivantes :

- Utilisation de nouvelles adresses mémoire
- Changement d'adresse mémoire (*context switching*) :
 - Lorsqu'un grand nombre de processus sont exécutés en même temps
 - Lors de l'allocation et des-allocation d'adresses mémoires

Nous pouvons présenter Remote Desktop Service et Citrix comme étant impactés par l'utilisation du MMU logiciel.

Dans le cas de l'utilisation du MMU matériel :

- Lors des TLB miss

C'est le cas de certaines applications Java sensibles aux latences liées aux TLB miss.

2. Allocation

L'allocation mémoire est tout comme l'allocation CPU, configurée au niveau de la machine virtuelle. C'est par contre plus simple car il s'agit simplement d'indiquer la quantité de mémoire qu'on veut allouer à la machine virtuelle, en mégaoctets ou gigaoctets (Mo ou Go).

Attention : ce n'est pas parce qu'on a attribué une certaine quantité de mémoire que la machine virtuelle en utilisera 100 %. De plus, une partie de cette mémoire n'est peut-être pas de la mémoire physique mais peut être sous forme d'espace disque comme indiqué au chapitre Fonctionnement de l'hyperviseur dans la section Mémoire - Allocation.

En fait ce que configure l'administrateur est la quantité de mémoire dont le système d'exploitation invité « croit disposer ».

3. Réserve et limite

Pour la mémoire vive, on considérera la réservation uniquement. Il n'y a pas d'intérêt à configurer une limite car celle-ci est en fait la capacité allouée à une machine virtuelle (il n'est pas possible que l'hyperviseur accorde plus de mémoire que la valeur configurée par l'administrateur).

4. Shares

Les shares mémoire fonctionnent comme pour le processeur. Il s'agit d'un poids relatif utilisé en cas de contention. Cependant, l'attribution des shares se fait en fonction de la mémoire attribuée à la machine virtuelle et du niveau (bas normal ou haut) :

▼ Mémoire		
RAM	4096	Mo
Réserve	0	Mo
	<input type="checkbox"/> Réserver toute la mémoire client (entièrement verrouillée)	
Limite	Illimité	Mo
Parts	Normales	40960

En « normal » on peut voir que pour 4 Go de mémoire vive il y a 40 960 shares, soit 10 shares par Mo de mémoire.

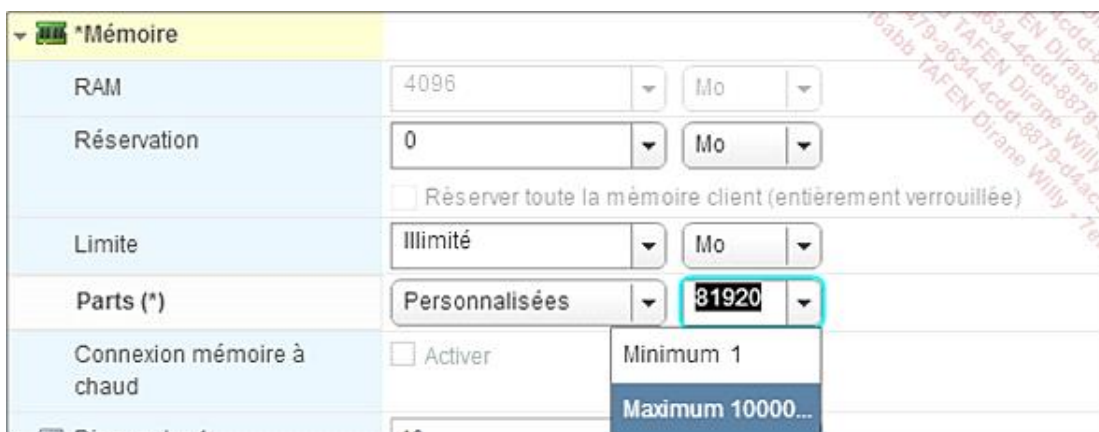
▼ *Mémoire		
RAM	4096	Mo
Réserve	0	Mo
	<input type="checkbox"/> Réserver toute la mémoire client (entièrement verrouillée)	
Limite	Illimité	Mo
Parts (*)	Basses	20480

En « bas » il y a 20 480 shares soit 5 par Mo.



*Mémoire		
RAM	4096	Mo
Réservation	0	Mo
<input type="checkbox"/> Réserver toute la mémoire client (entièrement verrouillée)		
Limite	Illimité	Mo
Parts (*)	Hautes	81920

En « haut » : 81 920 shares, soit 20 par Mo.



*Mémoire		
RAM	4096	Mo
Réservation	0	Mo
<input type="checkbox"/> Réserver toute la mémoire client (entièrement verrouillée)		
Limite	Illimité	Mo
Parts (*)	Personnalisées	81920
Connexion mémoire à chaud	<input type="checkbox"/> Activer	Minimum 1 Maximum 10000...

Le niveau personnalisé permet d'entrer le nombre de shares souhaité.

5. Optimisation


Afin d'économiser la mémoire, l'hyperviseur dispose de quatre techniques (voir chapitre Fonctionnement de l'hyperviseur), nous allons approfondir le transparent page sharing et la compression.

Transparent page sharing : mutualisation des pages mémoire utilisées plusieurs fois au sein d'une même machine virtuelle (intra-VM) ou au niveau de plusieurs machines virtuelles (inter-VM). Une page mémoire partagée n'est accessible qu'en lecture ou marquée Copy-on-Write (CoW) ce qui signifie que lorsqu'une VM tente de faire un accès en écriture, l'hyperviseur crée une copie de la page mémoire réservée à cette VM.

TPS se configure via les paramètres suivants :

- **Mem.ShareScanTime** : spécifie le temps en minutes pendant lequel la mémoire d'une VM est scannée afin de trouver des pages mémoires identiques. Le temps par défaut est de 60 minutes.
- **Mem.ShareScanGhz** : spécifie la quantité maximale de pages mémoire scannées par seconde pour chaque GHz de CPU disponible sur le serveur physique. La configuration par défaut est de 4Mo/sec pour 1 GHz.
- **Mem.ShareRateMax** : spécifie le nombre de pages scannées par VM.

Depuis vSphere 5.5 Update 2, le TPS inter-VM est désactivé par défaut. Pour l'activer, nous devons modifier le paramètre **Mem.ShareForceSalting** à 0. Lorsque ce paramètre est à 1 ou 2, le TPS inter-VM est désactivé. On nomme cela le salting.

 Du fait du fonctionnement du TPS, il n'a pas été possible, et ce pendant plusieurs années, de mettre le système en défaut c'est-à-dire d'obtenir un accès à des pages mémoire d'une machine virtuelle à partir d'une autre fonctionnant sur le même hyperviseur. Néanmoins, fin 2014, dans des environnements de recherche, il a été possible d'obtenir des informations sur la clé de chiffrement AES utilisée sur une autre machine virtuelle fonctionnant sur le même CPU physique, via TPS. Depuis, VMware a pris des mesures de précaution en désactivant par défaut le TPS inter-VM. Plus d'informations [ici : https://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2080735](https://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2080735)


Balloon driver : récupération de mémoire libre d'une machine virtuelle par un composant des VMware Tools.

Memory compression : compression de pages mémoire avec un ratio de 50 %. Chaque machine virtuelle dispose de son propre cache.

Les paramètres de configuration sont les suivants :

- **Mem.MemZipEnable** (active la compression)
- **Mem.MemZipMaxPct** (pourcentage maximal de RAM d'une VM pouvant être compressée)

VMkernel swap : écriture des pages mémoire sur disque. Ces pages mémoire peuvent être de la mémoire active pour la machine virtuelle, ce qui conduit à une baisse de performance généralement visible.

 Le VMkernel swap n'est pas lié à l'utilisation de fichiers d'échange dans la machine virtuelle. Le swap dans le VM dépend du fonctionnement du système d'exploitation invité et peut être déclenché alors qu'il y a une quantité suffisante de mémoire vive disponible.

À quel moment l'ESXi utilise-t-il chacune des techniques décrites ?

Cela dépend du minimum de mémoire que l'ESXi doit garder libre. Il s'agit de la valeur **Mem.Minfreepct** (minFree).

Cette valeur n'est pas une proportion fixe de la mémoire vive installée sur l'hyperviseur. Elle est calculée selon les seuils suivants :

Mémoire installée	Pourcentage	Taille en Mo
De 0 à 4 Go	6 %	246 Mo
De 4 à 12 Go	4 %	328 Mo
De 12 à 28 Go	2 %	328 Mo
Au-delà de 28 Go	1 %	(1 % de la RAM - 28 Go) Mo


Ensuite, suivant la quantité de mémoire vive disponible par rapport à la valeur **Mem.Minfreepct**, le serveur ESXi entre dans des états différents : High, Clear, Soft, Hard, Low. À chaque état correspondent une ou plusieurs optimisations mémoire.

État	Seuil	Actions
High	300 % de minFree	Préparation TPS
Clear	100 % de minFree	Préparation TPS - mutualisation des pages mémoires
Soft	64 % de minFree	TPS et ballooning
Hard	32 % de minFree	TPS, compression et VMkernel swap

Low	16 % de minFree	Compression, VMkernel swap et blocage VM
-----	-----------------	--

Ces valeurs ont changé avec vSphere 6.5.

État	Seuil	Actions
High	minFreePct > 400%	Préparation TPS
Clear	100% < minFreePct < 400%	Préparation TPS - mutualisation des pages mémoires
Soft	64% < minFreePct < 100%	TPS et ballooning
Hard	32% < minFreePct < 63%	TPS, compression et VMkernel swap
Low	minFreePct < 31%	Compression, VMkernel swap et blocage VM

 Un hyperviseur gère la mémoire des machines virtuelles. En plus de ces pages mémoire, il faut conserver et faire évoluer des informations sur la mémoire des différentes machines virtuelles. Ces informations sont conservées dans la mémoire physique de l'hyperviseur. C'est donc autant de mémoire vive non disponible pour les machines virtuelles !

La préparation TPS est en fait l'ESXi qui divise les grandes pages mémoire en pages plus petites et attend la mutualisation de ces pages mémoires.

Le blocage VM est un stade critique où l'hyperviseur bloque l'exécution des machines virtuelles qui consomment plus de mémoire que leur allocation (avec les shares pris en considération).

Exemple avec un serveur comportant 512 Go de mémoire vive

La valeur de minFree est calculée comme suit :

Seuil mémoire	Quantité (serveur) en Go	Calcul part minFree
0 à 4 Go	4	$4 * 6\% = 0,24$
4 à 12 Go	8	$8 * 4\% = 0,32$
12 à 28 Go	16	$16 * 2\% = 0,32$
Au-delà de 28 Go	484	$484 * 1\% = 4,84$

Au total la valeur minFree est de $0,24 + 0,32 + 0,32 + 4,84 = 5,72$ Go.

Voici le tableau des états du serveur ESXi avec 512 Go de mémoire vive :

État	Seuil	Actions
High	17,16 Go	Préparation TPS
Clear	5,72 Go	Préparation TPS - mutualisation des pages mémoires
Soft	3,6608 Go	TPS et ballooning
Hard	1,8304 Go	TPS, compression et VMkernel swap
Low	0,9152 Go	Compression, VMkernel swap et blocage VM

On voit qu'il faut que le serveur ait moins de 6 Go de mémoire vive disponibles (sur 512) pour que les mécanismes d'optimisation (présentant un overhead) se déclenchent.

Peut-être aviez-vous remarqué un comportement troublant sur les versions d'ESXi antérieures ?

Effectivement, l'état « clear » est apparu avec vSphere 6.0. Dans les anciennes versions, on passait de l'état High (100 % de minFree) à l'état Soft (64 % de minFree). Schématiquement, on se retrouvait dans la situation suivante : le TPS se déclenche, et avant que les pages mémoire soient effectivement mutualisées, le ballooning se déclenche.