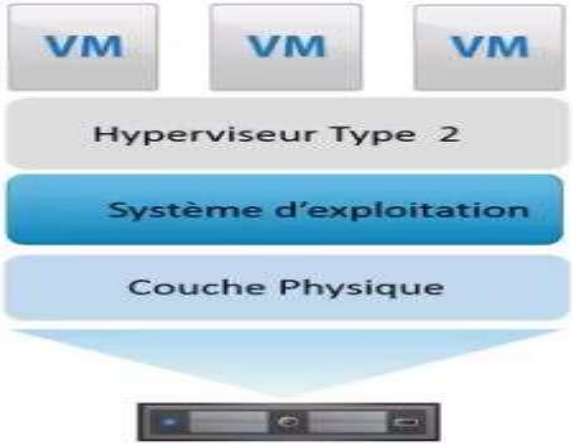
Hyperviseur type 2 (hébergé, host-based)

➢ Un hyperviseur de type 2 (**Host metal**) ou **hyperviseur hébergé** est un logiciel qui **s'exécute à l'intérieur d'un autre système d'exploitation classique (OS Hôte)**

○ Il s'installe comme n'importe quelle application et s'exécute sur un système d'exploitation déjà en place.

○ Un **OS invité (installé dans la machine virtuelle**) s'exécutera donc en **troisième niveau** au-dessus du matériel.

➢ L'hyperviseur de type 2 recrée, par voie logicielle, un environnement d'exécution complet pour un programme ou un système invité. Une fois installé, il crée des **VMs indépendantes de l'OS hôte.**

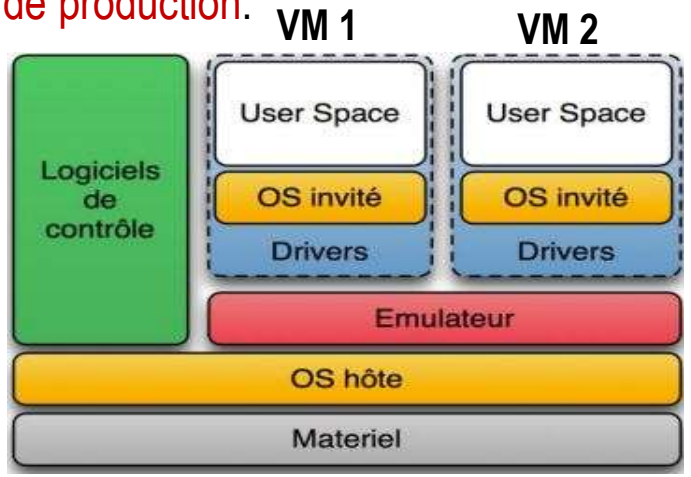


Hyperviseur de Type **2**

Source : Voir Références

Hyperviseur type 2 (hébergé, host-based)

- Les systèmes d'exploitation invités (OS invité) n'ayant pas conscience d'être virtualisés, ils n'ont pas besoin d'être adaptés. (OS invité « croit » s'exécuter sur une véritable machine physique)
- Le système s'exécutant dans la machine virtuelle est un système d'exploitation à part entière, tel qu'on pourrait en **installer sur une machine physique** : Microsoft Windows, GNU/Linux, Mac OS .
- Une solution de virtualisation avec un hyperviseur de Type 2 est plutôt destinée à des **usages de tests multiplateformes** (tests de compatibilité, application, OS, sécurité) dans le cas où vous avez une **seule machine (usage personnel)** et n'est pas adaptée à des contextes de production.
- L'hyperviseur de type 2 **ne prend pas** en charge l'**allocation dynamique / sur RAM**, il faut donc faire attention lorsque vous allouez des ressources à des machines virtuelles.



Source : Voir Références

Hyperviseur de Type 2

Hperviseurs type 2 : Usage

➢ La mise en place de ce type d'hyperviseurs est **assez facile** et **très efficace** pour des **utilisations multiples** telles que :

- Utiliser et tester une application sur un OS en particulier
- Tester un OS sans formater la machine physique
- Créer un petit réseau de plusieurs VMs (test des protocoles réseau, des règles de pare-feu….)
- Faire des tests de communications simples avec une deuxième machine

Source : Voir Références

Hperviseurs type 2 : Acteurs

- **VMware Workstation Pro** (payant) / **Vmware Workstation Player** (free) et **VMware Fusion** (pour Mac)
- **VirtualPC / Virtual Server** >>>> Hyper-V «Microsoft »
- **VirtualBox** : Oracle Corporation, dont une version en licence GNU GPL « General Public License »,libre)
- **QEMU** module
- **KQEMU** exclu (licence GNU GPL)
- **KVM:** Intégré au noyau GNU/Linux à partir de la version 2.6.20.

Source : Voir Références

Hperviseurs type 1 VS Hperviseurs type 2

La liste des principaux hyperviseurs qui existent sur le marché est :

Hyperviseur de type 1	Hyperviseur de type 2
VMware vSphere	Microsoft Virtual Desktop
Citrix XenServer	Virtual Box (Open Source)
Microsoft Hyper-V (intégré à Windows Server)	VMware Workstation & VMware Fusion
KVM (Linux)	Parallels Desktop

Source : Voir Références

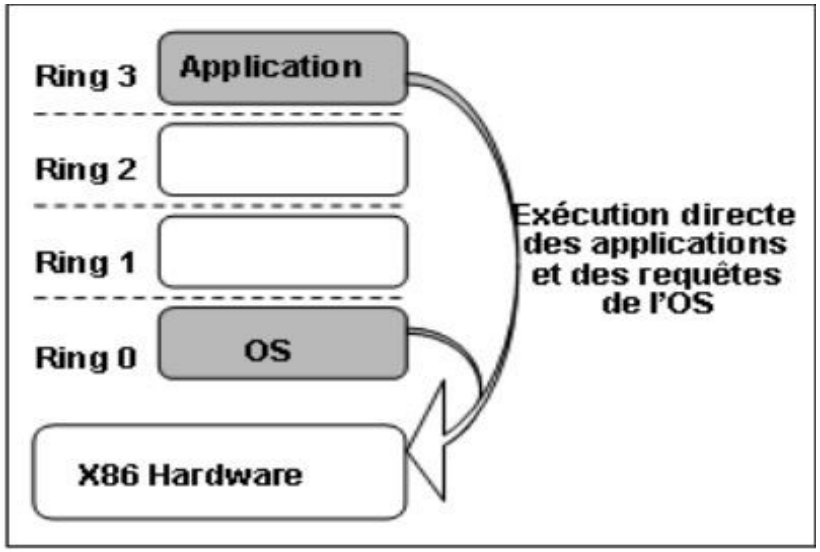
EXERCICES FAITS EN LIGNE

TP N°1 : INSTALLER ET CONFIGURER UNE MACHINE VIRTUELLE PERSONNALISEE SUR VMWARE WORKSTATION Pro

Types de virtualisation

- Emulation
- Virtualisation des serveurs : Hyperviseur Type 1 et Type 2
- Virtualisation complète (Full Virtualization)
- Para-virtualisation (Paravirtualization)
- Virtualisation du système d'exploitation
- Virtualisation des applications

Pourquoi la virtualisation complète et paravirtualisation ?



- Dans l'architecture des processeurs : Plus un programme est installé sur un **niveau d'exécution plus bas**, plus il exerce de **contrôle sur le système**.
- OS dispose de **plus haut niveau de contrôle** et accède **directement au ressources** en s'exécutant au **ring 0**

Challenge : **Comment faire pour placer un hyperviseur entre le matériel et un OS conçu pour s'exécuter sur le ring 0 dans ce type d'architecture?**

Solution : Appel aux techniques de la virtualisation complète et la paravirtualisation

Source : voir références

Virtualisation complète (Full Virtualization using binary translation)

- La **virtualisation complète** (full virtualization) est une technologie **utilisée par les hyperviseurs (type 2)** qui permet de faire fonctionner **n'importe quel système d'exploitation en tant qu'invité** dans une **machine virtuelle**.

Autrement dit, **émuler un environnement matériel complet** (intégralité d'une machine physique) sur **chaque machine virtuelle** (VM).

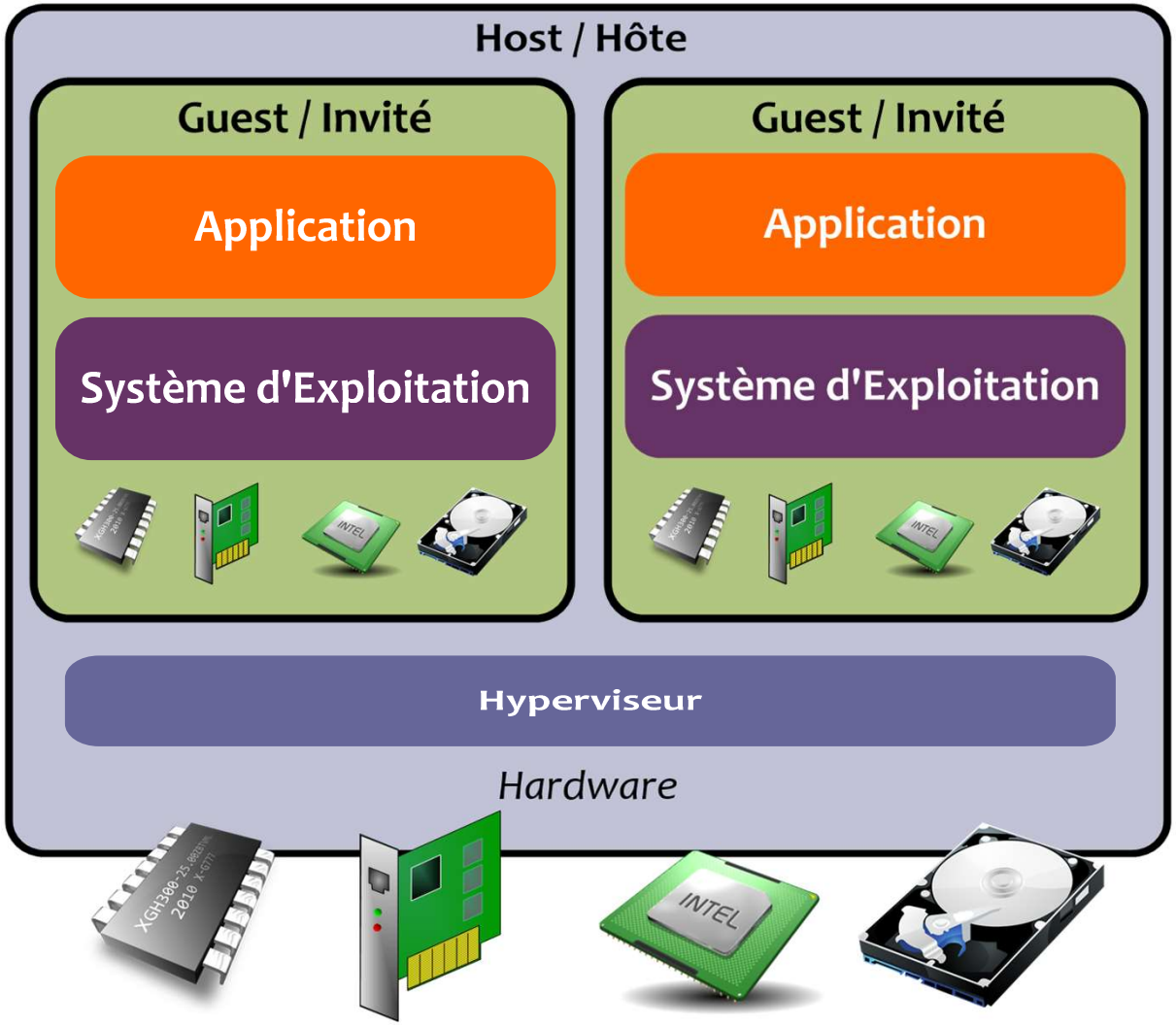
- L'hyperviseur crée un **environnement virtuel complet** simulant littéralement un **nouvel ordinateur complet, avec du "faux matériel".**

- En **virtualisation complète**, les systèmes d'exploitation et applications sont conçus pour fonctionner sur **la même architecture** que **celle de la machine physique hôte**.

- Chaque système invité « **croit** » s'exécuter sur une véritable machine physique

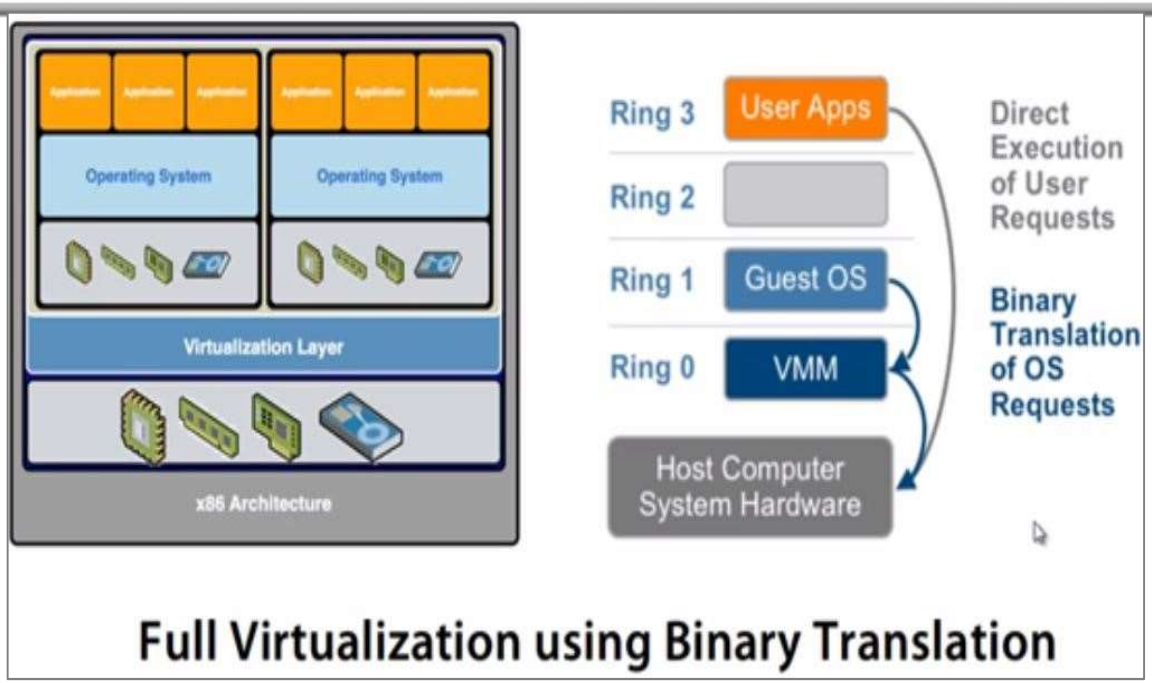
Source : voir références

Virtualisation complète : Modèle



Source : voir références

Virtualisation complète : Binary Translation



- VMware a développé en 1988 une technique appelée **translation binaire « Binary Translation (**traduction de la demande (instruction) des composants virtuels pour s'exécuter par le matériel) permet de :
  1) Placer **l'hyperviseur en Ring 0,**
  2) Déplacer les **OS à un niveau supérieur (Ring 1)** tout en leur garantissant un niveau de privilège supérieur à ceux des applicatifs (Ring 3)

Source : voir références

Virtualisation complète : Binary Translation

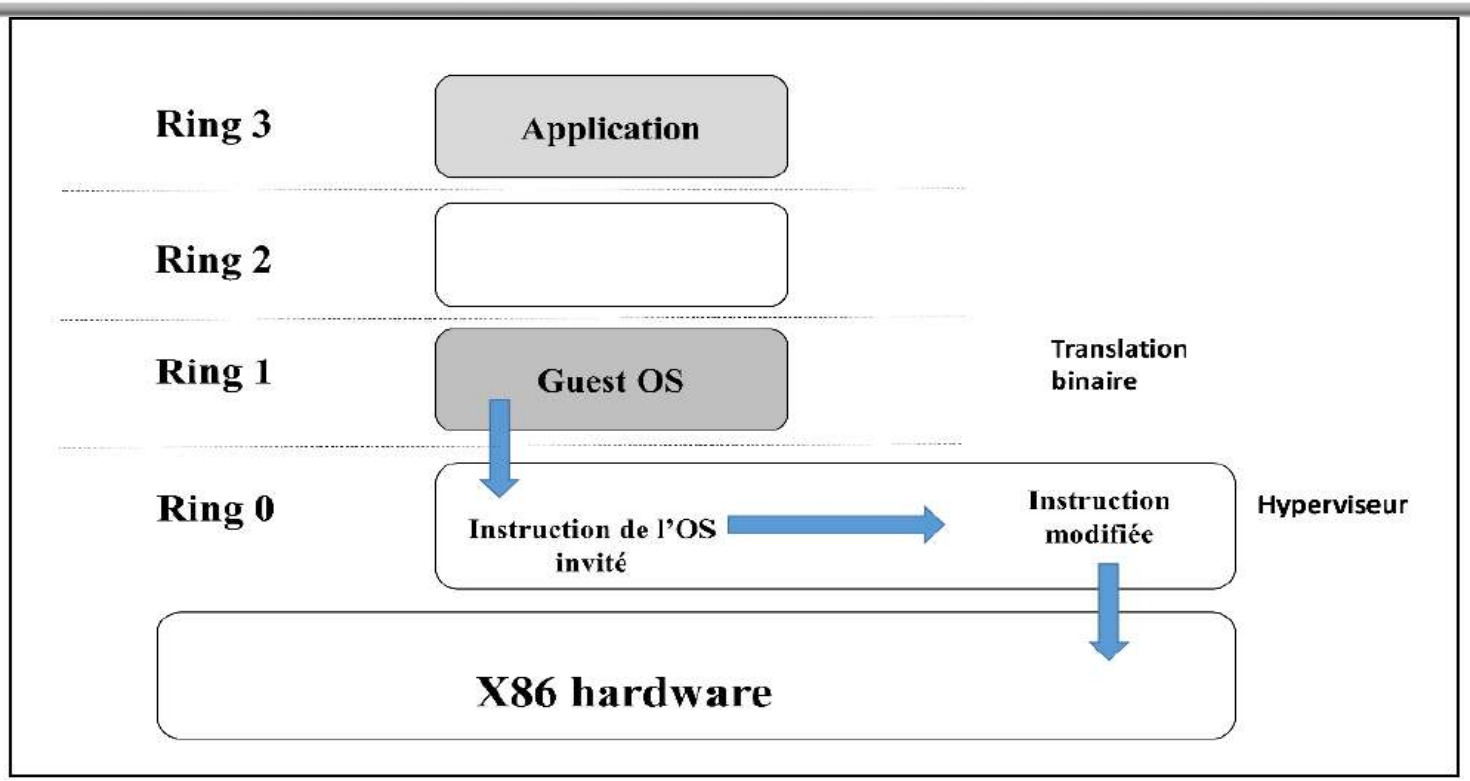
➢ La machine virtuelle doit **donc implémenter en logiciel une gestion complète du matériel** (mémoire) **de l'invité**, en utilisant les couches d'abstraction de l'hôte (système hôte) pour accéder au matériel.

➢Dans ce type de virtualisation, ce composant est appelé **VMM (Virtual Machine Manager)** ou gestionnaire de Machines virtuelles est **responsable du bon fonctionnement de l'environnment émulé**.

➢ OS invité dépend de l'environnment émulé via VMM , ce qui rend la VM émulée est indépendante des spécificités du matériel

➢ Cette solution responsable de la **translation de tous les appels du VMM** vers les ressources matérielles spécifiques de la machine physique.

Virtualisation complète – Binary Translation



- La virtualisation complète est basée sur 2 types d'opérations :
  1) Translation binaire d'instructions de l'OS invité vers le matériel.
  2) Exécution directes d'instructions vers le matériel

N.B : La translation binaire modifie certaines instructions provenant du Guest OS avant de les envoyer pour traitement aux processeurs physiques.

Source : voir références

Virtualisation complète – Avantages et Inconvénients (1)

Avantages:

- **Pas de modification au niveau du noyau du guest OS** (OS invité) car la translation binaire est exécutée au niveau du code binaire par le processeur.
- **Simplicité de mise en place**, ils sont fournis sous forme de packages ou d'exécutables avec une interface graphique soignée permettant de créer très rapidement des machines virtuelles
- **Simplification des processus de migration** des VMs - OS invités
- **Bon niveau d'isolation et de sécurité** des VMs (séparation nette entre VM et le système hôte)
- Produits de la virtualisation complète sont **gratuits**

Source : voir références

virtualisation complète : Avantages et Inconvénients (2)

Inconvénients:

➢ Ce type de virtualisation **ne permet de virtualiser que des OS prévus pour la même architecture matérielle** que le **processeur physique** de l'ordinateur hôte.

➢ **Génération latence** car toutes les requêtes provenant du « GuestsOS » seront traduites par l'hyperviseur et nécessitent un **travail supplémentaire** de la part du **CPU** lors de la translation binaire.

➢ **Filtrage au niveau de l'hyperviseur** = perte de performance car certaines instructions ne peuvent pas être virtualisées.

➢Quelques hyperviseurs de virtualisation complète :

- VirtualBox
- Logiciels de virtualisation de VMWare : VMWare Player, VMWare Workstation
- Parallels Desktop 4 for Windows & Linux
- KVM

Paravirtualisation (Paravirtualization or OS assisted virtualization)

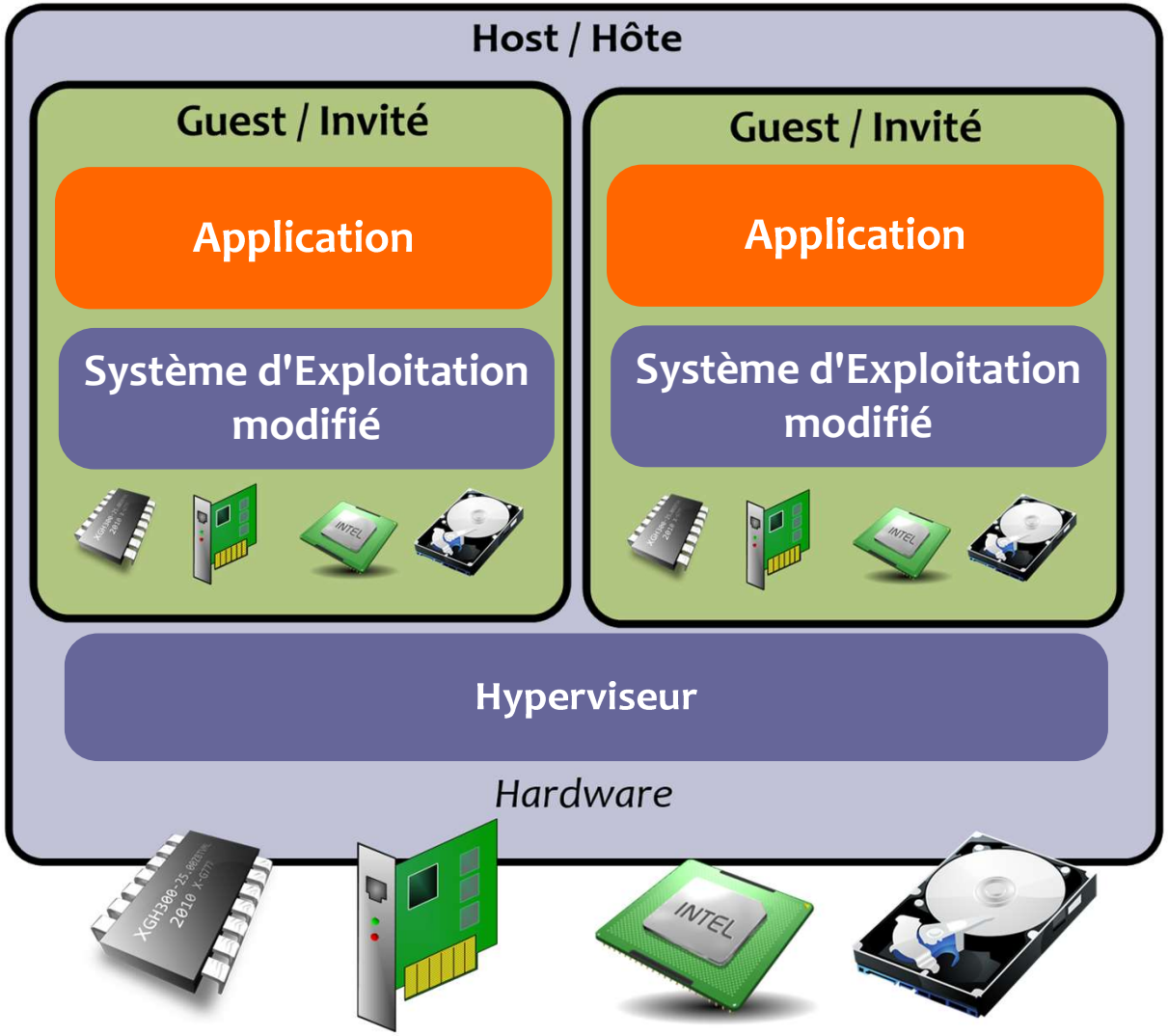
➢La **paravirtualisation ou Virtualisation assisté par OS** est l'une des techniques développée par **XenServer de Citrix**. Elle est très proche du concept de la virtualisation complète, dans le sens où c'est toujours un **système d'exploitation complet qui s'exécute sur le matériel émulé par une machine virtuelle**, cette dernière s'exécutant au dessus d'un système hôte.

➢ Au lieu de chercher à **faire croire** aux systèmes d'exploitation qu'ils s'exécutent sur une machine physique, il est possible d'**adapter le système d'exploitation à la couche de virtualisation**

➢ Pour une **meilleur collaboration**, via une interface de programmation (API), entre le **système hôte et l'invité , le système invité (la couche du noyau) est modifié pour être exécuté par la machine virtuelle (ailleurs que sur le Ring 0.)** au lieu d'accéder directement au matériel via des couches d'abstraction standards en Full virtualisation (fichiers sur le disque, *sockets* TCP-IP, etc.) .

>> OS modifié n'est pas **owner** du matériel.

Paravirtualisation : Modèle



Paravirtualisation (Paravirtualization or OS assisted virtualization)

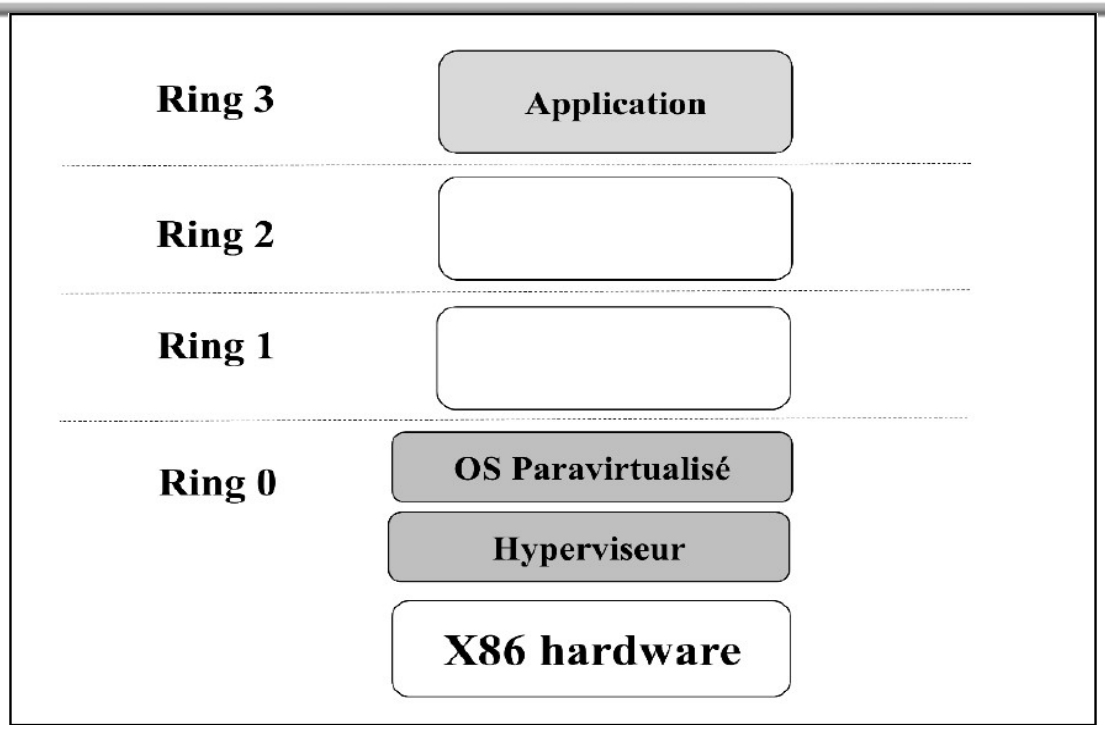
➢ Il **s'agit d'une modification des couches basses de l'OS**. Cela évite que **l'hyperviseur ne ralentisse les échanges entre les OS et le matériel**.

Tout le monde parle la **même langue,** la **langue de l'hyperviseur**. Ce qui rend les **OS coopératifs par cette** méthode.

Andy Ramiandrasoa, responsable marketing chez Trango

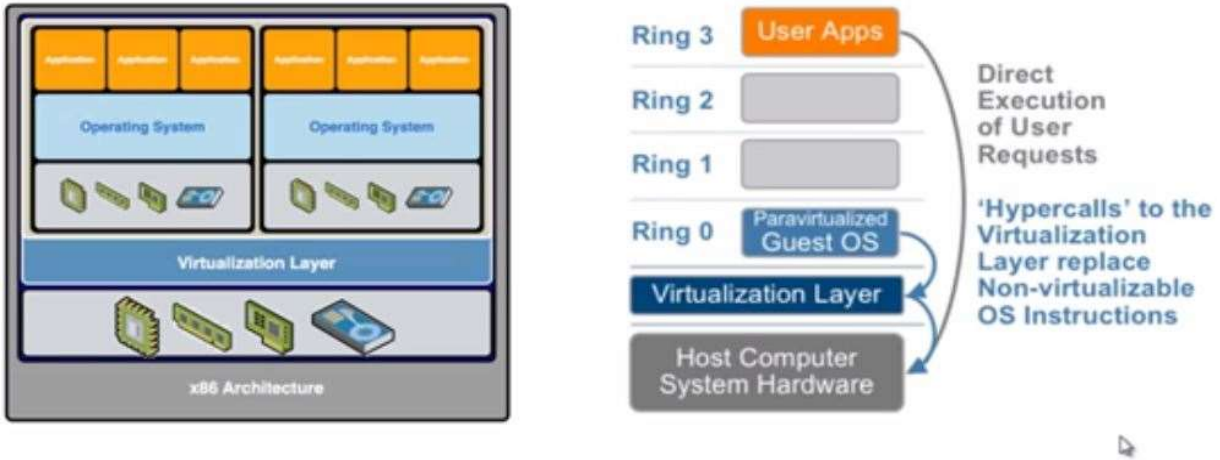
➢La paravirtualisation impose d'utiliser un **"Guest OS" (OS invité**) spécialement **adapté à l'hyperviseur,** qui consiste à **changer les BSP (Board Support Package) (**logiciel de support des cartes mères**)** de chaque système d'exploitation par un **BSP conçu pour cet hyperviseur**.

Paravirtualisation : Modèle



- Guest OS est conscient d'être virtualisé et modifie certaines instructions bas niveau avant de les envoyer au hardware.
- Il n'y a donc pas d'interception d'instructions ni de translation binaire.

Paravirtualisation (Paravirtualization or OS assisted virtualization)



OS Assisted Virtualization or Paravirtualization

- VMs ont la possibilité **d'invoquer directement la couche inférieure** de l'hyperviseur à partir du **noyau système d'exploitation invité.**
- Ces appels , appelés **Hypercalls**, sont dirigés directement vers les interfaces hypercall pour des opérations importantes en termes de performances telles que la gestion des interruptions, des timers aussi les fonctions de gestion mémoire.
- Mettre en œuvre un environnement s'exécutant en **ring0** (**privilège 0**) permettant de lire et d'écrire dans la mémoire vive sans restrictions. (attaquer le matériel via Hyperviseur)
- Il n'y a donc pas d'interception d'instructions ni de translation binaire.

Source : voir références

Paravirtualisation (Paravirtualization or OS assisted virtualization)

Avantages:

- **Limitation des surcharges liées à la translation binaire en virtualisation complète** , ce qui augmente la performance de VMs

- **Bonne gestion des performances** matérielles et excellente performances d'exécution des VMs se font par **diminution du temps d'accès** et du **consommation de ressources processur**.

- **Modifications** apportées au OS invités sont relativement **faciles à implémenter**.

Source : voir références

Inconvénients :

➢La portabilité et la compatibilité des VMs peuvent être mises en doute, car elle doit aussi prendre en charge l'OS non modifié.

➢Le **coût** de la gestion des OS para-virtualisés est **élevé**, car ils pourraient nécessiter de conserver les modifications de noyau d'OS

➢Les produits populaires **Xen, KVM et Oracle VM Server** en constituent de bons exemples.

VMware Tools pour certains Guest OS : drivers spécifiques développés par VMware ont conscience de la couche de virtualisation et qui communiquent plus facilement avec l'hyperviseur

Source : voir références