Virtualisation du calcul

M346 — Concevoir et réaliser des solutions cloud

Jérôme Frossard

EPAI

11 septembre 2024

Introduction

On peut dater la naissance du cloud à la mise sur le marché des premiers services web d'Amazon (AWS) au milieu des années 2000.

Mais aucune innovation ne surgit du néant. Ces services sont nés d'un besoin et ont été rendus possibles par un ensemble de technologies développées et théorisées au cours des décennies précédentes. Les technologies liées à l'Internet bien sûr, mais aussi celles liées à la virtualisation (calcul, stockage et réseau) que nous nous proposons d'explorer dans cette série de six présentations.

Dans la première présentation, nous avons abordé les notions de virtualisation et d'émulation. Le but de cette deuxième présentation est d'aborder les deux formes de virtualisation du calcul : les machines virtuelles et les conteneurs.

Plan

Plan

Qu'est-ce que la virtualisation du calcul ?

La virtualisation du calcul (ou du serveur), permet d'exécuter plusieurs systèmes d'exploitation sur un même serveur physique.

On peut distinguer deux formes de virtualisation du calcul :

- Machine virtuelle (*virtual machine* ou VM)
- Conteneur (*container*)

Parmi les principales raisons de virtualiser le calcul, on trouve :

- Isoler les différentes charges de travail (*workloads*)
- Optimiser l'utilisation des ressources matérielles
- Faciliter la migration d'une charge de travail sur un autre système physique.
- Automatiser le déploiement de charges de travail

Plan

Qu’est-ce qu’une charge de travail

Au sens strict, une charge de travail (*workload*) est la quantité de ressources informatiques (temps de calcul, espace de stockage, bande passante, etc.) nécessaires pour accomplir une certaine tâche en respectant certaines exigences.

Par extension, ce terme est plus couramment utilisé pour désigner un programme ou un système logiciel exécuté sur ordinateur permettant de réaliser cette tâche. C’est en ce sens que nous l’entendrons.

Une charge de travail peut être caractérisée notamment par :

- Puissance de calcul minimale,
- Quantité de mémoire minimale,
- Nombre d’opérations d’entrée-sortie par seconde (IOPS) pour le stockage,
- Bande passante minimale et latence maximale pour le réseau, etc.

Exemple de charge de travail

Dans une infrastructure d'entreprise, on peut typiquement identifier un grand nombre de charges de travail différentes. Par exemple :

- Serveur DHCP, SNMP, SYSLOG, RADIUS,
- Serveur DNS,
- Serveur d'annuaire (ActiveDirectory),
- Serveur d'authentification (Kerberos),
- Serveur de fichiers,
- Serveur Web (HTTP)
- Serveur de messagerie électronique (SMTP)
- Serveur de base de données

Pour des raisons de sécurité, de conformité ou de stabilité, il est souvent souhaitable, voire nécessaire d'isoler ces charges de travail les unes des autres.

Isolation des charges de travail

Avec un système d'exploitation (SE), un processus a son propre espace de mémoire, mais le reste est partagé. Une charge de travail est donc en concurrence avec les autres pour des ressources telles que ports TCP, système de fichiers, etc., et une faille de sécurité pourrait lui donner accès aux données stockées ou en transit d'autres processus.

Les conteneurs et les VM permettent d'augmenter le niveau d'isolation :

- Avec un conteneur, une charge de travail peut avoir son propre système de fichiers et ses propres interfaces réseau. Chaque conteneur a sa propre distribution de SE, mais partage le noyau de l'hôte. Tous les SE doivent donc être de la même famille.
- Avec une VM, une charge de travail peut avoir son propre SE avec son propre noyau. Les SE n'ont pas à être de la même famille.

De plus, la possibilité de limiter les ressources (CPU, mémoire, stockage, etc.) allouées à un conteneur ou à une VM permet un meilleur contrôle de l'utilisation des ressources et du niveau de performance.

Plan

Hyperviseur et machine virtuelle

L'utilisation de machines virtuelles (VM) permet d'exécuter différents SE sur une même machine physique.

Pour cela on utilise un système logiciel appelé hyperviseur[1]. Son rôle est d'assurer la séparation logique des VM et d'attribuer à chacune d'elles son quota de ressources (CPU, mémoire, stockage, etc.).

On distingue deux types d'hyperviseurs :

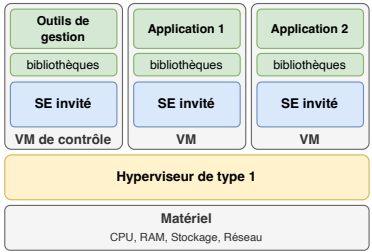
- Hyperviseur de type 1 ou *bare metal*
- Hyperviseur de type 2 ou hébergé

Remarque : Avant les années 2000, un hypervieur est souvent appelé moniteur de machine virtuelle (VMM). Aujourd'hui, le terme hyperviseur est d'usage plus courant. VMM reste valide dans ce contexte, mais ce sigle fait souvent référence au *Virtual Machine Manager*, la solution de gestion d'infrastructure virtuelle de Microsoft.

Hyperviseur de type 1

Un hyperviseur de type 1 (ou *bare metal*) s'exécute directement sur le matériel, à la place du noyau d'un système d'exploitation avec lequel il a certaines similarités en matière de gestion des ressources.

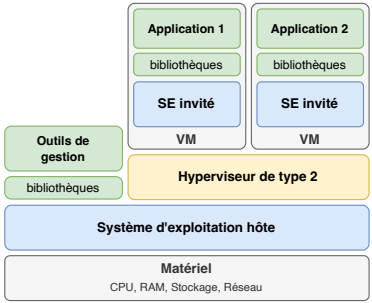
- L'hyperviseur et les VM sont le plus souvent gérés depuis une VM spéciale, souvent appelée VM de contrôle ou dom0. C'est notamment le cas des hyperviseurs Xen, Hyper-V et ESXi.
- KVM (Kernel-based Virtual Machine) fait exception ; il transforme le noyau Linux en hyperviseur sans le remplacer. Le SE de l'hôte reste inchangé, ce qui le rapproche d'un hyperviseur de type 2, bien qu'il soit plutôt classé comme type 1 ou hybride.



Hyperviseur de type 2

Un hyperviseur de type 2 (ou hébergé) s'exécute essentiellement dans l'espace utilisateur d'un SE hôte, et en partie dans son espace noyau grâce à des pilotes (drivers).

- L'hyperviseur et les VM sont gérés par des outils qui s'exécutent dans l'espace utilisateur du SE hôte.
- Les hyperviseurs de ce type (VirtualBox, VMware Workstation, Qemu/KVM, etc.) sont généralement utilisés sur un ordinateur personnel pour tester ou développer des applications.
- Bien que ce ne soit pas vraiment des hyperviseurs, il est possible de créer des VM dont l'architecture est différente de celle de l'hôte avec un émulateur comme Qemu sous Linux ou Rosetta sous macOS.



Hyperviseur de type 1 et de type 2

Avec des hyperviseurs de type 1 comme Hyper-V ou KVM (souvent décrit comme un hyperviseur hybride) utilisés aussi bien sur des serveurs que sur des ordinateurs personnels, il n'est pas toujours évident d'identifier le type d'un hyperviseur.

Plutôt que de distinguer les hyperviseurs selon leur type, on peut les distinguer selon la manière de les utiliser :

- Sur un serveur, un hyperviseur est principalement utilisé pour l'isolation des charges de travail et l'optimisation de l'utilisation des ressources.
- Sur un ordinateur personnel, un hyperviseur est principalement utilisé pour le test et le développement d'applications.

Plan

Espace utilisateur, espace noyau et instruction privilégiées

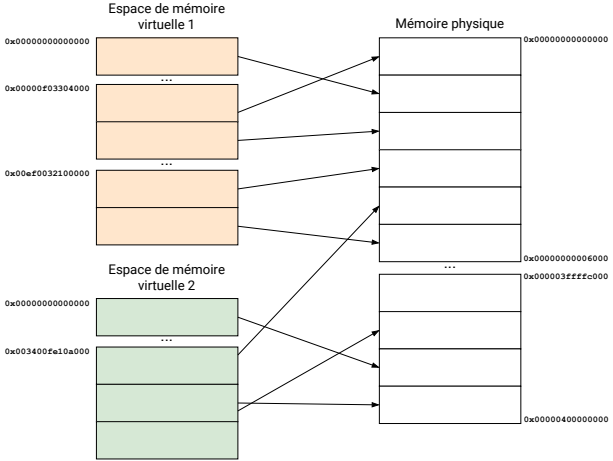
Pour comprendre le fonctionnement d'un hyperviseur, il faut aborder brièvement le fonctionnement d'un système d'exploitation (SE).

- Un CPU peut fonctionner dans différents modes qui déterminent les instructions qu'il peut exécuter ou non. P. ex., les CPU Intel ont 4 modes (*protection ring* 0 à 3).
- Un SE moderne a un espace utilisateur (*user space*) et un espace noyau (*kernel space*).
- Dans l'espace noyau, le CPU est en *ring 0* et peut exécuter n'importe quelle instruction, y compris un certain nombre d'instructions appelées instructions privilégiées.
- Dans l'espace utilisateur, le CPU est en *ring 3*. Dans ce mode, le CPU interdit l'exécution des instructions privilégiées, et lève une exception (*trap*) en cas de tentative.
- Si un processus dans l'espace utilisateur tente d'exécuter une instruction privilégiée, le noyau intercepte l'exception et met généralement fin au processus.
- Pour accéder au matériel, un processus de l'espace utilisateur doit passer par le noyau à l'aide d'un appel système (*system call*), généralement effectué par une bibliothèque.

Mémoire virtuelle (I/II)

- La mémoire physique est limitée et chaque octet est numéroté à partir de 0.
- Chaque processus a son espace de mémoire virtuelle, virtuellement infini (2^{64} octets) et chaque processus est également numéroté à partir de 0.
- Le numéro d'un octet est une adresse. L'adresse d'un octet de la mémoire physique est une adresse physique, celui d'un octet d'un espace de mémoire virtuelle est une adresse virtuelle.
- La mémoire physique et virtuelle est divisée en pages de quelques ko ou Mo.
- Le SE maintient une table de pages qui permet à l'unité de gestion de la mémoire (MMU) de convertir une adresse virtuelle en une adresse physique.
- Lorsque le processeur exécute une instruction, les adresses fournies sont des adresses virtuelles, le CPU passe par le MMU pour accéder à la mémoire physique.
- Comme le MMU est une unité matérielle, l'opération est très rapide.

Mémoire virtuelle (II/II)



Émulation des instruction

Le noyau d'un SE invité ne peut être réellement exécuté en *ring 0*, car il ne doit pas avoir un accès direct au matériel de l'hôte.

- Pour cela, une solution consiste à utiliser l'hyperviseur comme un interpréteur pour exécuter le SE invité. Les instructions privilégiées sont émulées, et les autres sont passées au CPU. L'hyperviseur doit également émuler le MMU en maintenant une copie de la table de pages de SE invité.
- Cette solution est coûteuse et ne permet pas d'obtenir de bonnes performances.
- Le défi est donc de permettre l'exécution du logiciel qui tourne sur les VM directement par le CPU, et de n'intercepter que les instructions qui doivent l'être absolument (l'émulation des instructions privilégiées est inévitable pour assurer la séparation des VM).

Support matériel de la virtualisation

Le support matériel des processeurs modernes (VT-x et AMD-V) permet de relever ce défi, en permettant :

- De configurer le CPU de telle manière qu'il génère une exception (*trap*) lors d'une tentative d'exécution d'une instruction privilégiée même en *ring 0* (non-root mode). Cela permet au logiciel d'une VM d'être exécuté directement par le CPU et de passer le contrôle à l'hyperviseur pour émuler les instructions privilégiées.
- Au CPU d'utiliser directement le MMU pour la conversion d'adresse virtuelle grâce aux tables de page étendues (*extended page table* ou EPT) ou imbriquées (*nested page table* ou NPT)
- Au SE invité d'avoir un accès direct à certains périphériques.

Plan

Qu'est-ce que l'OS-level virtualization ?

La virtualisation au niveau du SE (*OS-level virtualization*) consiste isoler des charges de travail dans des conteneurs (*container*).

Un conteneur est assez similaire à une VM. Il y a toutefois des différences importantes :

- La charge de travail est exécutée nativement, comme si elle était exécutée directement sur le SE hôte.
- Un conteneur peut avoir sa propre distribution d'un SE, mais comme il utilise le noyau de l'hôte, le SE invité et celui de l'hôte doivent être de la même famille (que des distributions Linux ou que des versions de Windows)
- Un container n'a pas de carte graphique, car il n'y a pas de matériel virtuel.

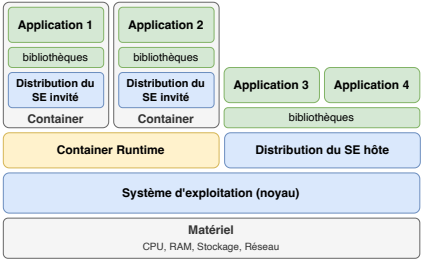


Image de conteneur et OCI

Un container est créé à partir d'une image qui contient tous les fichiers nécessaires et éventuellement des métadonnées pour décrire le conteneur.

Une image de container est essentiellement une archive *tarball* (fichier .tar.gz) qui contient tous les fichiers nécessaires (bibliothèque, outils, application, etc.) et éventuellement un manifeste pour décrire le conteneur et un fichier de configuration.

L'Open Container Initiative (OCI) maintient une norme qui spécifie le format d'une image pour faciliter l'interopérabilité entre les différents outils (création d'images, runtimes, orchestrateurs, etc.)

Container Runtime

Pour créer un conteneur à partir d'une image, on utilise un *container runtime*.

Par exemple :

- LXD,
- Docker,
- CRI-O,
- Containerd

Fonctionnement d'un conteneur

Les conteneurs reposent sur des fonctionnalités de base du SE :

- Les *cgroups* permettent de limiter les ressources CPU et mémoire utilisables par les processus qui en font partie.
- Les *namespaces* permettent d'isoler les différents aspects de l'environnement (utilisateurs, processus, interfaces réseau, nom de l'hôte, système de fichiers, etc.) pour des espaces séparés pour chaque conteneur.
- Les systèmes de fichiers union (p. ex., OverlayFS) permettent de créer, distribuer et charger l'image d'un conteneur tout en permettant des modifications persistantes pour chaque conteneur.