Rapport — Hyperviseurs (Xen vs Firecracker)

Pougala Biko — Cai Chaolei

3 mars 2022

1 Introduction

2 Les hyperviseurs

Un hyperviseur, également connu sous le nom de **Virtual Machine Monitor (VMM)** est un logiciel qui permet de créer et gérer des machines virtuelles sur un ordinateur hôte. Un hôte va héberger plusieurs machines virtuelles en partageant virtuellement ses ressources, telles que la mémoire ou la puissance de calcul (cœurs de CPU et GPU). Pendant très longtemps les entreprises ayant des besoins numériques ont fait appel à des serveurs dits *bare-metal* (« métal nu » en anglais). Elles ont acheté des ordinateurs entiers et les ont installées directement au sein de leurs locaux. Ensuite, l'équipe IT de l'entreprise a configuré ses machines, y a installé les bons logiciels, configuré les accès réseau, bases de données et autres, avant de pouvoir y héberger les services et programmes de l'entreprise. Cette procédure prend en général de 9 à 10 mois, suivant les besoins de l'entreprise en question (au bas mot 6 mois dans certaines situations). L'essor de l'ordinateur personnel (PC) dans les années 80 et de l'Internet dans les années 90 et 2000 a encouragé de nombreuses entreprises à fermer leurs propres serveurs et à héberger leurs services sur des machines hébergées dans des data centers publics, sur des machines partagées avec d'autres clients, notamment grâce à la virtualisation et à la conteneurisation. Le fait d'héberger plusieurs machines virtuelles sur le même matériel a deux principaux avantages : le premier est de réduire considérablement le temps nécessaire pour déployer un nouveau service ou augmenter les capacités en termes de puissance de calcul d'un service existant (ceci afin de le rendre *scalable*). Ce qui pouvait prendre 10 mois avec des serveurs *on premises* prend avec une machine virtuelle de l'ordre de quelques minutes à quelques heures. En effet, une des propriétés des machines virtuelles est qu'elles ne dépendent pas du matériel sous-jacent ; elles peuvent donc être portées ou clonées vers n'importe quel autre appareil. Les services peuvent donc augmenter en capacité rapidement et avoir une certaine forme de résistance aux pannes. Le deuxième avantage est l'utilisation plus optimale des ressources des machines sous-jacentes qui a permis de réduire les coûts d'exploitation de ses serveurs, et donc le coût d'hébergement de ses services de façon significative. En effet, lorsqu'une entreprise détient ses propres serveurs, il est rare que toutes ses ressources (processeur, disque, bande passante réseau, mémoire...) soient utilisées à 100% 24h/24. Le cloud permet à des machines virtuelles d'utiliser toutes les ressources du matériel sous-jacent au maximum, augmentant ainsi le rendement de l'exploitation de ces machines. Pour cette raison, de nombreuses entreprises, grandes ou petites, ont décidé de ne plus du tout exploiter leurs propres machines et de passer entièrement au Cloud. C'est le cas des services de streaming Spotify et Netflix, ou encore de la banque américaine Capital One.

2.1 La virtualisation totale

Il y a deux types d'hyperviseurs : le type 2 est dit "grand public", car l'hyperviseur tourne comme un simple programme au sein du système d'exploitation de la machine hôte (par ex : Virtualbox, VMWare Fusion, Parallels...). Il y a deux couches logicielles entre le système d'exploitation de la machine virtuelle et le matériel sous-jacent : l'hyperviseur et le système d'exploitation hôte. Le coût en termes de performances est donc important.

Les hyperviseurs de type 1 sont dits "bare-metal" car ils s'exécutent directement sur le matériel, sans aucune couche logicielle entre ce dernier et l'hyperviseur. Dans ce cas de figure, l'hyperviseur a la totale responsabilité d'allouer les ressources matérielles de la machine hôte entre les différentes machines virtuelles, de façon similaire à un OS. Dans le cas de la *virtualisation totale*, l'hyperviseur

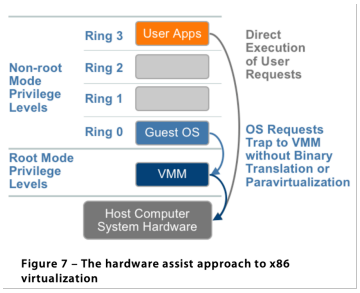

FIGURE 1 – La virtualisation totale et son accélération matérielle

virtualise toutes les ressources de la machine hôte et les présente ainsi à chaque machine virtuelle. Ce type de virtualisation a l'avantage de permettre d'exécuter des programmes et systèmes d'exploitation nativement, mais nécessite de virtualiser certains composants du CPU très complexes, tels que le **Memory Management Unit** (MMU) ou encore le **Translation Lookaside Buffer** (TLB), ce qui a un coût plus ou moins significatif en termes de performances. Une autre alternative est la *paravirtualisation*. Dans ce cas de figure, seuls certains composants matériels sont virtualisés, tandis que les systèmes d'exploitation virtuels sont modifiés pour qu'ils aient un accès direct aux composants matériels difficiles à virtualiser, comme la table des pages mémoire.

3 L'hyperviseur Xen

Xen est un hyperviseur libre de droits développé en 2003 par une équipe de l'université de Cambridge, au Royaume-Uni. Xen a été conçu à l'origine pour prendre en charge jusqu'à 100 machines virtuelles sur une même machine hôte. Xen utilise la paravirtualisation. Pour cette raison, les systèmes d'exploitation qui peuvent tourner avec Xen doivent être modifiés pour qu'ils puissent fonctionner. Les créateurs promettent que ces modificatione sont mineures et n'impactent pas l'**Application Binary Interface**, et donc le fonctionnement des applications utilisateur. Ils ont pour cela développé leur propre OS baptisé XenoLinux, qui est, contrairement à la philosophie originelle de la virtualisation, conscient qu'il est executé dans un environnement virtuel. Voici ci-bas un récapitulatif des choix d'implémentation architecturale de Xen :

Gestion de la mémoire	Processeur	Interruptions	Périphériques I/O
Pagination mémoire : Accès direct à la table des pages matérielles de l'OS hôte par chaque machine virtuelle. Les mises à jour sont validées par l'hyperviseur	— Gestion des exceptions : L'OS virtuel enregistre un tableau des exceptions auprès de Xen — Protection : OS virtuel a un niveau de privilège inférieur à Xen — Appels systèmes (syscalls) : L'OS virtuel implémente un handler	Hardware Interrupts : Remplacé par un système orienté événements	— Un mécanisme orienté événements remplace les interrupts matériels pour accéder aux périphériques. Des interfaces virtuelles sont utilisées par l'OS virtuel. — Un buffer circulaire avec des anneaux I/O pour accéder aux périphériques réels

4 L'hyperviseur Firecracker

Firecracker est une solution d'hyperviseur rendu public et open source en 2018 par l'équipe d'AWS. Par opposition à ses concurrents comme QEMU qui ont pour objective de créer des machines virtuelles dans son intégralité, le paradigme de Firecracker est de créer des instances de micro-VM qui sont réduite aux stricts minimums pour fonctionner.

La solution se repose essentiellement sur la technologie de virtualisation KVM (Kernel-based Virtual Machine) intégrée à Linux. KVM permet de mettre en œuvre des machines virtuelle en tant que processus Linux standard, donc par la même occasion de lui imposer des contraintes comme namespaces qui permet d'associer des uids et pids. D'un point de vue noyau, le visiteur (le micro-VM) n'a jamais l'accès au noyau du machine hébergeur, il ne dispose que de l'accès à un noyau visiteur proposé par KVM. Cela permet d'avoir une meilleure isolation comparée aux paradigme 'conteneur' type docker à laquelle le noyau de l'hébergeur est partagé (et donc une possibilité de failles malgré isolation du conteneur). Ainsi, l'hébergeur peut monitorer ces micro-VM via les outils qui existent déjà sur Linux comme top, ps, vmstat. Néaumoins, pour configurer les micro-VM, il faudra passer par une interface REST que propose Firecracker. Il est possible donc d'attribuer une quantité de mémoire et un nombre de cœur arbitraire et de fixer une débit maximal réseau (c'est normal puisque c'est le gagne-pain d'Amazon) à une instance de micro-VM. Un point majeur à cité est que Firecracker est écrite en Rust qui offre de meilleure garantie en termes de sécurité mémoire. En choisissant délibérément de garder un nombre de fonctionnalité minimal, Firecracker ne fait qu'environ 50 milles lignes de code Rust comparé à son homologue QEMU avec plus de 3 millions de code en C. Il n'y a pas de BIOS, pas de port PCI, port USB, GPU, pas de port d'imprimante etc. Certes il y a moins de fonctionnalités, mais les champs d'attaque possible est aussi réduite. Dans notre présentation nous avons cité brièvement VENOM (Virtuali[s]ed Environment Neglected Operations Manipulation) qui se résume à un buffer overflow du module qui devait simuler un lecteur de disquettes. Le visiteur pouvait alors s'échapper de son cloisonnement et accéder à l'espace mémoire de l'hébergeur.

4.1 Sauvegarde et migration

La migration des micro-VM n'est pas possible car les micro-VM devait avoir un court cycle de vie avec le service lambda. Néaumoins, Amazon a implémenté un mécanisme de snapshot. Un snapshot est donc une sauvegarde à un moment t de l'état d'un micro-VM Firecracker, il est alors possible de reconstruire ce micro-VM à ce moment t d'exécution. Firecracker effectue le snapshot de la mémoire, de son état ainsi que les disques de données.
Pendq

4.2 Systèmes d'exploitation

Firecracker est donc un hyperviseur de type 2 qui peut être installé sur n'importe quelle machine hébergeur linux (d'architecture x86, AMD ou ARM) disposant d'un noyau supérieur à la version 4.14. Du coup les instances micro-VM Firecracker qui y tournent par-dessus n'ont pas de contrainte de système d'exploitation tant que c'est du Linux. Nous avons été un peu confus par cette question car les services d'Amazon portaient confusion.
En résumé, il y a 2 types d'architecture avec Firecracker chez Amazon, la première avec le service Lambda, qui permet d'exécuter des « lamdba function » d'amazon. Cela se résume à compacter vos fonctions à exécuter grâce à un package de déploiement d'Amazon qui lui va être exécuté sur une instance de micro-VM firecracker. Dans ce cas-là, Amazon propose un système d'exploitation custo ou alors il est possible de lui fournir un parmi une liste d'image supporté par Amazon.
Enfin, avec le service Fargate, vous donnez plus une 'fonction lambda' mais un conteneur (type docker) qui va être exécuté sur une instance de micro-VM Firecracker avec un système d'exploitation parmi ceux proposé par Amazon). D'un point de vue utilisateur, Amazon souhaite épargner au mieux l'interaction client-système d'exploitation, à la place il propose d'utiliser conteneur afin réaliser la partie compatibilité.
J'espère vous avoir éclaircie sur ce dernier point, donc en résumé, Firecracker peut proposer des micro-VM avec diverse système d'exploitation mais dans ce cas-là, le manque de composant matériel est-il justifié en dehors du contexte « serverless » « cloud » proposé par Amazon ? La solution peut se

résumer avec la phrase « fait par Amazon, pour Amazon ». L'idée de micro-VM est vraiment très bien, il est possible d'utiliser Firecracker mais il est plus préférable de se tourner vers des projets basés sur rust-vmm qui offre plus de fonctionalités.

Références

[ABI+20] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker : Lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 419–434, Santa Clara, CA, February 2020. USENIX Association.

[Bar] et al Barham, Dragovic. Xen and the art of virtualization, year =.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. volume 37, pages 164–177, 01 2003.

[BK10] Diane Barrett and Gregory Kipper. Virtualization and forensics. Syngress, Boston, 2010.

[Yua19] Moxing Yuan. deep-analysis-aws-firecracker-principle-virtualization-container-runtime-technology/. *Amazon AWS blog*, 09 DEC 2019.