Unikernel : une technologie qui réunit les avantages de la virtualisation et de la création de conteneurs

Selon l'éditeur américain Lynx Software Technologies, focalisé sur les marchés de périphérie de réseau (edge) critiques, l'heure a sonné par la technologie des « unikernels » qui a le potentiel de réunir les avantages de la virtualisation et de la création de conteneurs. Une technologie qui offre un degré élevé de sécurité avec comme corollaires des performances accrues et une moindre consommation des ressources matérielles.

Si la technologie des « unikernels » existe depuis au moins 2016, son adoption reste partielle, y compris pour les applications critiques où elle pourrait apporter les avantages les plus significatifs. Cet article commence par un tour d'horizon des approches existantes, explore les avantages et les limitations des unikernels et conclut que si cette technologie n'est pas encore prête pour une adoption généralisée, ses avantages sont suffisamment attractifs pour que les équipes de développement envisagent son déploiement dans des contextes spécifiques. Les éditeurs peuvent les y aider en créant des solutions unikernel qui peuvent être facilement mises en œuvre en parallèle avec les systèmes d'exploitation temps réel (RTOS) classiques.

AUTEUR

Ian Ferguson, vice-président Ventes et marketing, Lynx Software Technologies.

Virtualisation et conteneurs

La technologie de virtualisation, qui consiste à exécuter plusieurs systèmes d'exploitation sur un même matériel partagé, est aujourd'hui bien comprise, même si son mode d'utilisation des ressources reste relativement inefficace. Il y a encore quelques décennies, tout le monde utilisait des machines virtuelles pour héberger et gérer les infrastructures. Plus récemment, la tendance chez les industriels est à l'utilisation de conteneurs avec des systèmes tels que Docker et Kubernetes.

A l'origine, l'architecture système de virtualisation reposait sur l'implémentation de plusieurs machines virtuelles. Dans cette approche, chaque machine virtuelle doit exécuter sa propre instance d'un système d'exploitation, ce qui a pour effet la duplication des responsabilités. Ce type d'infrastructure est également difficile à gérer, car il implique plusieurs serveurs qui sont tous des machines virtuelles indépendantes. Le déploiement de logiciels dans une architecture de machines virtuelles peut être traité de deux façons. Dans le premier cas, le système de build (construction logicielle) peut produire une image complète d'une machine virtuelle avec le logiciel intégré et la machine virtuelle redémarre à l'arrivée de la mise à jour. Dans le second cas, le système de build produit uniquement le pack logiciel qui est chargé sur les serveurs au moyen d'un ensemble de scripts.

Ces deux approches impliquent une configuration complexe et, au final, créent des incohérences entre les machines virtuelles. Cependant, l'administrateur maîtrise tous les aspects de l'environnement du système et peut le configurer pour répondre à des besoins spécifiques. Le débogage est en outre relativement simple, car il est possible de se connecter directement à une machine virtuelle.

Les conteneurs reprennent le concept des machines virtuelles mais éliminent les duplications de tâches d'une machine à l'autre. Au lieu de charger l'intégralité d'un système d'exploitation pour une application, Docker laisse les conteneurs utiliser le noyau du système d'exploitation hôte tout en leur permettant de charger les bibliothèques et programmes propres à l'application. En ajustant le conteneur et son image, il devient possible de paramétrer avec précision les bibliothèques et la configuration utilisées par votre application. Cela permet d'obtenir des gains de performances sans la lourdeur de l'exécution de l'intégralité du système d'exploitation.

Les conteneurs sont faciles à exécuter sur les machines de développement et le processus de développement est lui-même également beaucoup plus simple, dans la mesure où il suffit de charger des conteneurs prédéfinis dans un référentiel (de conteneurs) et que les systèmes de production peuvent en extraire la version mise à jour. L'approche reposant sur les conteneurs n'a toutefois pas que des avantages. Les logiciels doivent être adaptés à l'utilisation dans des conteneurs (et donc être conteneurisés), ce qui peut devenir complexe, surtout avec des bases de code anciennes. Les conteneurs disposent d'un nombre considérable de configurations pour l'affectation des ressources et les fonctionnalités d'interopération et, par conséquent, il est facile de faire des erreurs de configuration.

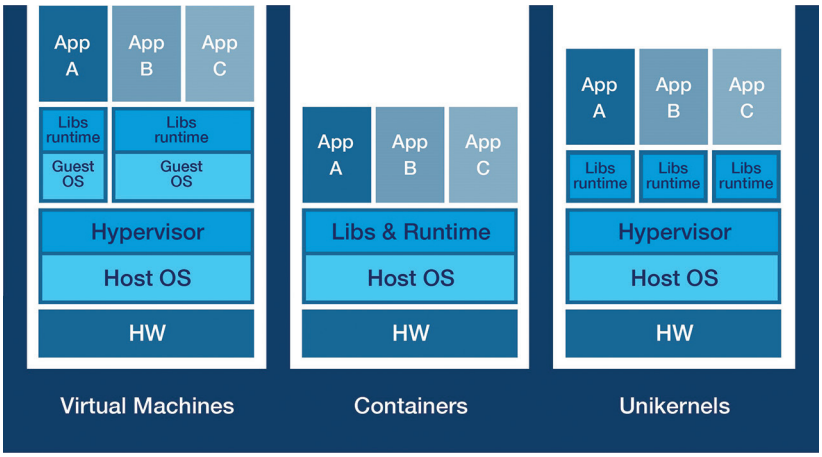
Unikernels – avantages et limitations

L'étape logique suivante de l'évolution des machines virtuelles vers les conteneurs est l'unikernel, qui va encore plus loin avec les concepts des conteneurs. Les unikernels sont avant tout un ensemble de bibliothèques binaires préconfigurées. Les unikernels ne gèrent pas l'allocation des ressources. C'est l'hyperviseur qui gère les interopérations directes avec le matériel. Tous les appels système propres à l'application sont poussés aussi près de l'application que possible. Et cela est géré par l'hyperviseur. Le concept d'architecture unikernel allie la sécu-

rité du partitionnement au niveau des machines virtuelles à la vitesse et au faible encombrement, propres aux conteneurs.
La figure 1 illustre la différence entre les différentes approches architecturales. Pour les systèmes sécurisés, il faut utiliser un hyperviseur de type 1 (où il n'y a pas de système d'exploitation « assistant » présent) qui s'exécute directement sur le matériel et charge les machines virtuelles. Les hyperviseurs qui dépendent d'un « système d'exploitation hôte » constituent à la fois un compromis et un point de faiblesse dans les systèmes.
Les unikernels génèrent encore moins de surcharge que les conteneurs et sont plus rationnels, étant donné les possibilités d'amélioration des performances. De plus, comme ils ne font pas appel à un noyau multi-utilisateur à espaces d'adressage multiples, la sécurité est considérablement renforcée.

1 COMPARATIF ENTRE MACHINES VIRTUELLES, CONTENEURS ET UNIKERNELS

Pour les systèmes sécurisés, il faut utiliser un hyperviseur de type 1 (où il n'y a pas de système d'exploitation « assistant » présent) qui s'exécute directement sur le matériel et charge les machines virtuelles. Les hyperviseurs qui dépendent d'un « système d'exploitation hôte » constituent à la fois un compromis et un point de faiblesse dans les systèmes.



Comparaison entre un système d'exploitation et un unikernel

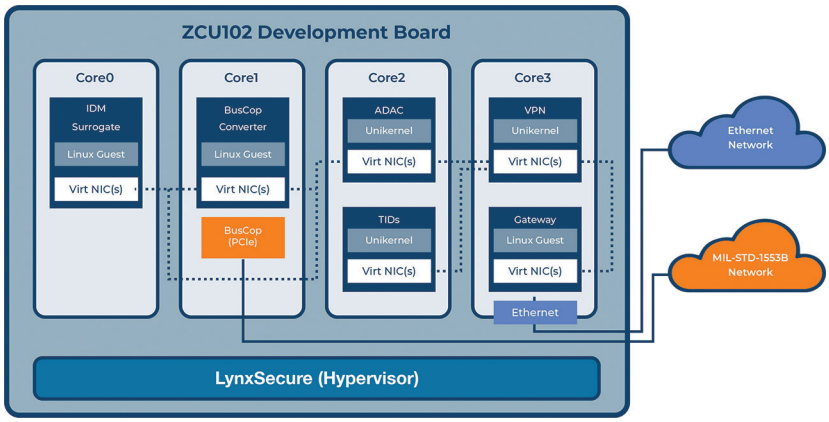
Les unikernels présentent des avantages considérables en matière de sécurité par rapport à un système d'exploitation, mais ils présentent également des inconvénients.
L'un des principaux avantages est que, contrairement aux systèmes d'exploitation, les unikernels s'exécutent en mode utilisateur, plutôt qu'en mode noyau. De façon générale, le mode noyau est réservé aux fonctions de plus bas niveau du système d'exploitation, qui sont aussi celles qui ont le degré de confiance le plus élevé. Les défaillances en mode noyau sont catastrophiques, car elles bloquent l'ensemble du système. En mode utilisateur, le code en cours d'exécution ne peut pas accéder directement au matériel, ni référencer la mémoire. C'est la raison fondamentale pour laquelle l'utilisation des unikernels est désormais envisagée par l'industrie. De fait, quasiment chaque semaine apporte son lot de cyberattaques sur des infrastructures critiques, une grande entreprise ou une entité étatique. Le système d'exploitation est de fait un point de vulnérabilité.
De plus, et contrairement à un système d'exploitation, les unikernels ne gèrent pas l'allocation des ressources. L'hyperviseur gère les interopérations directes avec le matériel. Tous les appels système propres à l'application sont poussés aussi près de l'application que possible et cela est géré par l'hyperviseur comme dit plus haut. Sachant que pour les systèmes sécurisés, il faut utiliser un hyperviseur de type 1. Les hyperviseurs qui dépendent d'un « système d'exploitation hôte » constituent à la fois un compromis et un point de faiblesse dans les systèmes, pour la raison indiquée ci-dessus et doivent, selon nous, être évités.
L'un des principaux inconvénients à l'usage des unikernels est qu'ils sont monoprocessus et mono-utilisateur. Et l'ajout de la gestion de processus s'avère une surcharge considérable. Dans ce cas, il faut en effet absolument démarrer/arrêter/inspecter un processus, assurer la communication entre les processus, etc. L'aspect multi-utilisateur impliquent des autorisations et des authentifications, l'isolation des ressources, et ainsi de suite. Par conséquent, certaines applications sont quasiment dans l'impossibilité d'être mises en œuvre comme des unikernels. Dans les domaines de spécialité de Lynx, à savoir l'aérospatial et la Défense, l'utilisation de bibliothèques ARINC 653 serait extrêmement difficile dans un contexte unikernel. Nous pensons que pour les gérer, il faudrait mettre en œuvre plusieurs applications à processus unique et déployer des systèmes d'exploitation en parallèle avec les unikernels, tout en s'assurant que l'architecture des systèmes permette d'atténuer l'impact de l'infiltration d'une instance de système d'exploitation…
Parmi les autres problèmes qui ont limité l'application des unikernels jusqu'à présent, citons :
1 - Le débogage. Dans la mesure où un unikernel n'exécute aucun système d'exploitation, il n'est pas possible de se connecter directement à son « shell » pour remonter à la source du problème.
2 - La production d'images unikernel est compliquée et implique une connaissance approfondie du sujet.
3 - Les frameworks applicatifs actuels doivent s'adapter et fournir de la documentation sur l'utilisation au sein d'unikernels.
4 - L'absence d'unikernels certifiables/certifiés vis-à-vis de la sûreté de fonctionnement pour les applications critiques.

Elargissement du domaine d'application des unikernels

Afin de surmonter ces limitations, notamment la dernière, et de profiter des avantages de l'approche unikernel pour de nombreuses applications, Lynx a mené début 2022 une étude sur les différentes options d'unikernel disponibles dans le domaine public. Nous avons découvert que si plusieurs d'entre elles sont liées à un langage de compilation spécifique, tel que runtime.js (javascript), Clive (Go), LING (Erlang) et Mirage (Ocaml), l'une, OSv, peut exécuter plusieurs langages. Cela nous a conduit à approfondir cette option.

2 MISE EN ŒUVRE DE L'UNIKERNEL OSV SUR UNE CARTE DE DÉVELOPPEMENT XILINX

OSv est un unikernel modulaire open source conçu pour exécuter de façon sécurisée une seule application Linux non modifiée au-dessus d'un hyperviseur. Il a été conçu pour exécuter des fichiers binaires Linux x86-64 et Aarch64 non modifiés, ce qui en fait un unikernel compatible avec les fichiers binaires Linux.



OSv est un unikernel modulaire open source conçu pour exécuter de façon sécurisée une seule application Linux non modifiée au-dessus d'un hyperviseur. Il a été conçu pour exécuter des fichiers binaires Linux x86-64 et Aarch64 non modifiés, ce qui en fait un unikernel compatible avec les fichiers binaires Linux. Son adoption dans les applications d'infrastructure IT signifie qu'OSv dispose d'un nombre raisonnable de fonctionnalités d'observabilité, telles la journalisation, l'analyse de trace et le débogage. La licence BSD est simple à utiliser et toute une gamme de périphériques et de protocoles (TCP/IPv4, PCIe, ACPI par exemple) sont pris en charge. Lorsque nous avons réalisé notre étude, la prise en charge d'IPv6 n'était pas encore mature.

Les unikernels dans les applications de sécurité critiques

Pour Lynx, l'intérêt principal de la technologie unikernel réside dans sa sécurité, car cette approche réduit radicalement la surface d'attaque. D'autre part, ces types d'applications ne nécessitent pas de synchronisations garanties, ni d'artefacts de certification de sécurité.

Par exemple, un unikernel comme OSv peut être utilisé pour exécuter des composants de sécurité comme IDS (Intrusion Detection System) et VPN (Virtual Private Network). Le figure 2 montre comment cela pourrait être mis en œuvre sur une carte de développement Xilinx. Les entités TIDS (Tactical Intrusion Detection System) et DAC (Dynamic Access Control) sont des détecteurs d'anomalies statistiques qui sont déployés par l'armée américaine sur certains de leurs réseaux pour superviser le trafic IP et 1553. L'utilisation d'une diode de données et d'un filtre sur l'unikernel permettrait à un client de remplacer une machine virtuelle Linux, ce qui économiserait de l'espace mémoire et réduirait radicalement la surface d'attaque.

L'un des avantages de cette approche architecturale est, selon nous, que l'introduction d'un unikernel réduit le nombre de scénarios pour les applications dites « bare metal ». Les unikernels peuvent fournir des API (le produit Lynx prend en charge Posix par exemple), ce qui permet aux développeurs de créer plus simplement des applications. Ceci dit, si les unikernels permettent de réduire considérablement l'encombrement mémoire du logiciel, il reste de l'espace pour les applications bare metal ayant des fonctions uniques spécifiques. Ainsi, il serait encore bien plus efficace d'implémenter une application de 500 lignes en tant qu'application bare metal (par exemple pour travailler avec des clés privées).

Maturité du marché

La technologie unikernel est-elle prête pour les applications critiques de l'aérospatial et de la Défense ? Si elle offre des avantages considérables, il subsiste des domaines où les critères sont très élevés et pour lesquels l'architecture unikernel reste un peu limitée.

Conformité à POSIX et ARINC : la demande évolue vers des architectures logicielles plus modulaires et devant s'aligner sur des API populaires, telles que POSIX (et, aux États-Unis, la norme FACE poussée par l'armée américaine).

Taille : plus le système d'exploitation est lourd, plus les processus de certification, tel que DO-178, sont longs et coûteux. Le nombre de lignes de code source (SLOC - Source Lines of Code) permet de se faire une idée de la complexité d'un logiciel. Pour les versions les plus strictes de la norme DO-178, il faut en moyenne 2 à 4 heures pour traiter une simple ligne de code source dans un processus de certification.

Ordonnancement : OSv met en œuvre un algorithme de d'ordonnancement préemptif et équitable (fair) entre les différentes tâches, ce qui signifie que l'accès au processeur est distribué de façon équitable entre les utilisateurs et les groupes. Avec l'ordonnancement préemptif, le processeur est alloué aux processus pendant une période limitée, tandis qu'avec l'ordonnancement non préemptif, le processeur est alloué à un processus jusqu'à ce que ce dernier se termine ou bascule à l'état en attente. Dans les systèmes critiques, certains scénarios impliquent l'allocation des ressources en fonction de leur priorité (par exemple si toutes les tâches ne sont pas égales) avec un ordonnancement non préemptif.

Conclusion

Si Lynx pense que les unikernels ne sont pas encore prêts pour une adoption généralisée dans les scénarios critiques, les avantages qu'ils présentent sont suffisamment significatifs pour pousser les équipes de développement à regarder de plus près cette technologie pour des contextes spécifiques et limités.

Pour cela, les produits unikernel doivent être associés à des systèmes d'exploitation temps réel qui ont fait leurs preuves commercialement. En maintenant un niveau élevé de compatibilité entre l'unikernel et le produit autonome, les clients peuvent porter librement leurs applications entre différents environnements, afin d'explorer et d'exploiter les avantages de chacun sans duplication notable de l'effort de développement. ■