Institut Francophone Internationale

Université Nationale du Vietnam, Hanoi

Rapport Final : Unikernel avec Docker

Kouamen Tankoua Joseph.

Encadrants :
Nguyen Hong Quang

Master I : Réseaux et Systèmes Communicants |Travail Personnel Encadré

Table des matières

1 Introduction

1.1 Contexte

Unikernel, selon unikernel.org sont des images de machines spécialisé à espace d'addressage unique construites à l'aide de systèmes d'exploitations de bibliothèques. Certains [10]pensent aussi que les Unikernels sont de petites machines virtuelles, rapides, sécurisés et dépourvues de Systèmes d'exploitations. Pour www.wikipedia.org un **Unikernel**[18]est une image de machine à espace d'adressage unique, construite à l'aide de systèmes d'exploitation de bibliothèque. Un développeur sélectionne, dans une pile modulaire, l'ensemble minimal de bibliothèques correspondant aux constructions de système d'exploitation requises pour l'exécution de son application.

Pour cela, nous avons choisi d'utiliser l'outil **Docker**. **Docker** est un logiciel libre permettant d'automatiser le déploiement d'applications dans des conteneurs logiciels. D'après la firme 451 Research, "Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé qui pourra être exécuté sur n'importe quel serveur linux"[18]. La technologie de conteneurs Docker offre ainsi flexibilité et portabilité d'exécution des applications, peu importe que cela soit sur un cloud privé ou public, une machine nue, une machine locale, etc. Les conteneurs Dockers peuvent être utilisés pour étendre des systèmes distribués de sorte à ce qu'ils puissent s'exécuter de manière autonome depuis une seule machine physique ou une seule instance par noeud. Elle permet de déployer les noeuds au fur et à mesure que les ressources sont disponibles. Pour des systèmes comme Apache Cassandra, Riak ou autres systèmes distribués, il offre un déploiement transparent et similaire aux PaaS.

1.2 Problématique

On compte à présent plusieurs systèmes d'exploitations et des millions d'applications. Ces systèmes sont souvent multi-utilisateurs, multi-tâches et multi-fonctions. Ils trouvent une utilisation chez les utilisateurs finaux. On peut aussi constater le développement des infrastructures clouds, la présence des systèmes embarqués et la venue du concept de l'internet des objets. Toutes ces évolutions nous donne de constater l'incompatibilité des applications entre plateformes différentes, de grosses images (logiciel) de machines qui nécessitent beaucoup de ressources pour effectuer leurs tâches, la sécurité est de moins en moins garantit par l'utilisation des antivirus ou pare-feu, les temps d'exécutions et d'utilisation mémoire sont importants, les systèmes actuels doivent choisir l'architecture qui leur convient pour leur déploiement. Les Unikernels seront-ils capables d'apporter des éléments de solutions à ces problèmes ? Pendant notre module de Travail Personnel Encadré nous suivrons les pistes de solutions offertes par Unikernel couplé à l'outils Docker[10].

1.3 Etat actuel

Les unikernels sont riches et possèdent à ce jour plusieurs projets en cours. Nous avons par exemple le projet MirageOS qui est développé par l'équipe du même nom (MirageOS), et ayant comme tuteur le projet Xen et la Linux Foundation. MirageOS est un système d'exploitation de bibliothèque qui construit des identificateurs uniques pour des applications réseau sécurisées et hautes performances sur diverses plates-formes de cloud computing et mobiles[14][15]. Le code peut être développé sur un système d'exploitation normal tel que Linux ou MacOS X, puis compilé dans un unikernel spécialisé totalement autonome s'exécutant sous un hyperviseur Xen ou KVM.

Le projet **UniK** (prononcé you-neek) est un outil permettant de simplifier la compilation et l'orchestration d'unikernels. Semblable à la façon dont Docker construit et orchestre les conteneurs, UniK automatise la compilation de langages populaires (C / C ++, Golang, Java, Node.js. Python) en unikernels. UniK déploie des unikernels en tant que machines virtuelles sur OpenStack, VirtualBox, QEMU, AWS, vSphere, ukvm et Xen. UniK intègre les travaux des projets Rumprun, OSv, IncludeOS et MirageOS[15][16].

Il existe plusieurs projets unikernels voyons sur quels objectifs approfondir notre étude.

1.4 Objectifs

Les objectifs que nous nous fixons d'atteindre pour ce sujet se démenbres en deux catégories. Les objectifs théoriques pour approfondir notre compréhension du sujet, et les objectifs pratiques pour proposer une solution à la résolution de problème liés aux Unikernels.

1.5 Objectifs Théoriques

Nous avons comme objectifs théoriques les points suivants :

— Etudier l'historique des Unikernels
— Etudier Unikernel.
— Etudier la conception des SE
— Présenter les avantages des Unikernels.
— Présenter les inconvénients des Unikernels.
— Présenter Unikernel et les systèmes embarqués.
— Comparer Unikernels et les systèmes d'exploitation des systèmes embarqués.
— Présenter les méthodes de conceptions des systèmes embarqués.
— Présenter les possibilités de conception des systèmes embarqués avec un système d'exploitation Unikernel.
— Comprendre et utiliser l'outils Docker.
— Etudier les outils utilisés pour la construction des Unikernels.
— Justifier le choix de l'outils Docker pour notre travail.
— Etudier la construction des Unikernels avec l'outil Docker.

1.6 Objectifs pratiques

Nous avons comme objectif pratique de : Réaliser un cas pratique de construction d'Unikernel avec l'outil Docker. Nous utiliserons parmi plusieurs arcticles, l'article "Docker kicks off the Unikernel revolution"[13].

2 Etude de l'art

2.1 Unikernel

2.1.1 Définitions

Un unikernel est une images systèmes spécialisée, dans laquelle tous les processus partagent le même espace mémoire et qui est créée à partir de système d'exploitation bibliothèques. La tâche du développeur consiste à choisir parmi un ensemble de modules, un minimum de bibliothèque qui correspondent le mieux aux services du système d'exploitation nécessaire à l'exécution de l'application. Ces bibliothèques ainsi que l'application sont compilés ensembles et configurées pour donner une image fixe à but unique, qui peut fonctionner sur un hyperviseur ou directement sur du matériel dépourvu de systèmes d'exploitation comme Windows ou Linux [23].

Les systèmes d'exploitation bibliothèque de par leurs limites de protection qui sont repoussées vers les bas niveaux, implémentent des mécanismes qui sont utiles pour faire fonctionner le matériel ou pour gérer les protocoles réseau et appliquer une politique pour l'accès et l'isolation de l'application.

2.1.2 Avantages et Inconvénients[23]

Avantages

— Pas de transaction de privilège entre l'espace utilisateur et l'espace noyau. Ainsi l'OS offre un accès direct au matériel.
— Meilleure Sécurité en réduisant la taille du code déployé cela réduit la surface d'attaque et donc améliore la sécuritée.
— Taille réduite : la taille des unikernels est généralement 4% de la taille du code de base équivalent pour un système classique.
— Optimisation du système : il permet d'optimiser les applications, les pilotes et permet d'améliorer la spécialisation.
— Temps de démarrage faibles : les unikernels démarrent rapidement.

Inconvénients

— Difficulté à faire fonctionner plusieurs applications dans un OS bibliothèque qui a des ressources isolées.
— Réécriture des drivers pour le changement de matériel.
— Non appliqué pour une utilisation multi-utilisateur, multi-application.

2.1.3 Outils de construction d'Unikernels

On peut citer plusieurs outils, nous allons cependant nous limiter à quelques uns d'entre eux.
ClickOS : plateformes hautement virtualisée, ayant une petite taille (5Mo), démarre rapidement (entre 30 millisecondes), ajoute un petit delai (45 microseconds) et 100 ClickOs peuvent être exécutés et saturer une interface de 10 GB sur un serveur peu cher [6].
Clive : C'est un système d'exploitation conçu pour fonctionner dans des environnements distribués et dans le cloud. Il compile les applications et les services avec des bibliothèques qui leur permettent de s'exécuter sur du matériel nu[1].
IncludeOS : C'est un système d'exploitation unikernel minimal. Il permet **d'exécuter des applications dans le cloud sans système d'exploitation (et sur un matériel réel)**. Il ajoute une fonctionnalité de système d'exploitation aux applications pour créer des machines virtuelles sécurisées, performantes et économes en ressources [2].
MirageOS : C'est un système d'exploitation bibliothèque qui permet de construire des unikernels pour des applications réseaux haute performance, sécurisées, sur une variété de plateforme mobile et cloud computing. Le code peut être développé sur un système d'exploitation normal comme Linux, Windosw ou encore MacOS X puis compiler en un unikernel qui s'éxécute sur l'hyperviseur KVM et Xen [14].
Rumprun : c'est un unikernel qui fonctionne sur des hyperviseurs tels que KVM et xen, mais aussi sur du métal nu. Rumprun peut être avec ou sans les interfaces POSIX. Le premier permet aux applications POSIX de fonctionner directement, tandis que le second permet de créer des solutions personnaliser avec une empreinte minimale [4] et [5].

2.1.4 Origine d'Unikernel

Il remonte aux années 1990 avec **l'exokernel** ou **exo-noyau**. L'exokernel est un type de système d'exploitation dont l'architecture est dite modulaire. Le noyau ou kernel, est le premier élément codé venant exploiter les ressources matérielles de l'ordinateur. L'approche de l'exo-noyau consiste à rapprocher le développeur (l'application) du matériel en faisant une abstraction du système d'exploitation. Ceci n'est pas le cas dans les systèmes d'exploitation monolithique [1], les problèmes de l'abstraction des ressources matérielles qui ont été notés sont les suivants :

- Manque de fiabilité : le code est lourd et les mécanisme complexes, ce qui dégrade les performances
- Manque d'adaptabilité : la densité du code empêche une évolution simplifiée et les applications dépendent grandement du système d'exploitation.
- Performance diminuée : à cause de l'excès d'abstractions matérielles, les programmes sont loin des ressources et ceux qui n'utilisent pas ces abs-

1. C'est à dire que les fonctions du système et des pilotes sont regroupées (codées) dans un seul bloc de code, et un seul bloc binaire généré à la compilation

tractions en souffrent.
— Manque de flexibilité : s'il était possible pour les applications de passer outre le système d'exploitation et implémenter leurs propres abstractions, leur flexibilité serait meilleure.

Les objectifs des exokernels sont :
— Donner aux utilisateurs un système de code léger.
— Donner aux développeurs un environnement propice aux évolutions.
— Rapprocher les applications au maximum de la couche matérielle pour optimiser les performances.
— Permettre aux programmes ou applications de définir eux même leurs niveaux d'abstractions matérielles.

En 1990, **Dawson Engler** et **Frans Kaashoek** entreprennent d'étudier et de développer un nouveau type de noyau qui va exposer la partie matérielle du système d'exploitation. Ils souhaitent avoir un système qui fonctionne en espace utilisateur avec plus d'adaptabilité, efficacité et flexibilité d'utilisation pour améliorer les performances.

1995 : développement de **xok** dédié au processeur X86 32 bits par l'équipe de chercheur du MIT (Dawson Engler et Frans Kaashoek). Sa version la plus utilisée est xok/exos. L'équipe a également développé Aegis qui fonctionne sur l'architecture MIPS.
1996 : le MIT développe GLAZE pour l'architecture FUGU microprocesseur.
2000 : **SMP-Xok** voit le jour dans la même période et par sa simplicité et légèreté de code, il intéresse le domaine universitaire : **Nemesis**, voit le jour à l'université de cambridge. Nemesis a pour avantage d'être implémenté sur plusieurs plateformes (X86, RISC 64 bits et StrongARM).
2003 : A l'université des sciences et technologies de Lille, des chercheurs développent l'exokernel embarqué et embarqué temps réel avec les versions **Camille** et **Camille RT** sur des architectures RISC et CISC.
2010 : Des chercheurs américain développent **XomB** dédié à l'architecture multiprocesseur X86 64 bits.
2013 : Le projet Xen développe l'exokernel **Mirage** dédié au cloud computing.

Architecture des exokernels.

Les exokernels sont spécifiques au type de processeur. Le noyau protège, multiplexe et partage les ressources physiques de manière dédiée. Le noyau protège les ressources matérielles mais délègue la gestion de ces ressources aux bibliothèques d'applications. Le noyau fournit donc :
— Chargement de code
— Multiplexages des ressources
— Des liens sécurisés

Autour du noyau suit les **LibOS**. Une **LibOS** est l'interface entre le programme et le noyau. On distingue différente LibOS pour des systèmes différents.
— **ExOS :** libOS dédiée à l'exo-kernel **Xok** et spécialisée dans l'exécution d'applications UNIX non modifiées (POSIX).
— **Xok-CThreads :** aussi dédiée à **Xok** et spécialisée en environnement multithreading.

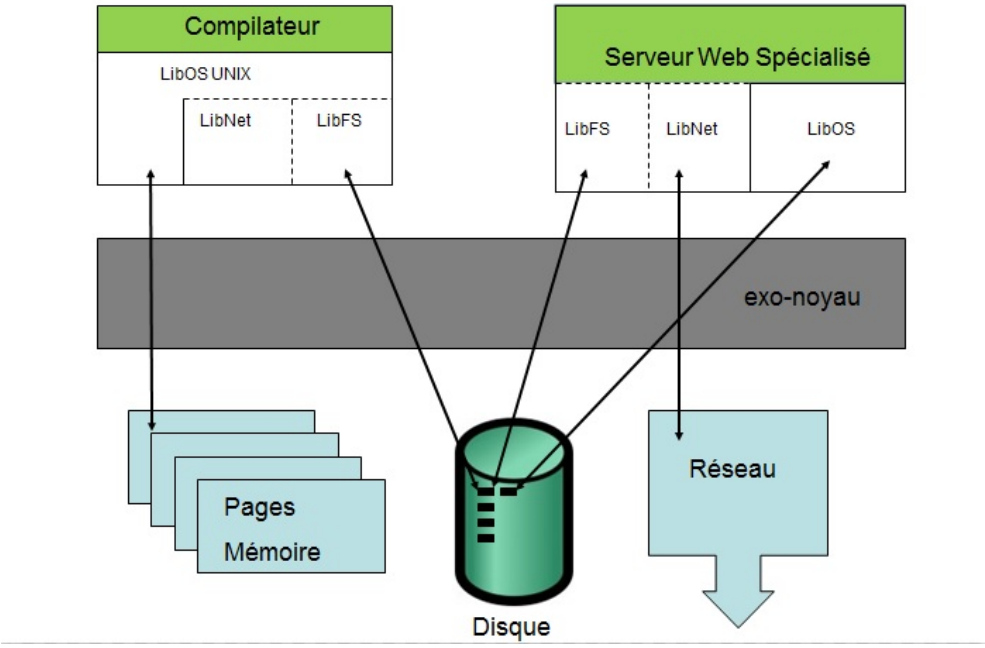

FIGURE 1 – Architecture des exokernels [21]

— **LibAMP :** dédiée à **Xok** il fait fonctionner l'interface d'un routeur actif.
— **VOS :** dédiée à **SMP-Xok** (architecture X86 - multiprocesseurs) pour faire fonctionner des applications en environnement multiprocesseur.

Les applications qui tournent sur les exokernels dialoguent avec les LibOS, et sont directement responsables de la gestion des ressources matérielles ce qui améliore les performances. Les programmes peuvent ainsi accéder aux ressources avec un niveau bas d'abstraction. La gestion de la sécurité et des privilèges est laissée au noyau. Dans les systèmes standards seul les applications ayant des privilèges ou le noyau peuvent gérer les ressources. Dans les systèmes exokernels la gestion de privilèges est séparée de celle de la sécurité et du multiplexage, ces éléments sont gérés par le noyau. Les applications gèrent à part et de manière indépendante leurs propres mémoire virtuelle, le système de fichier et le réseau. Pour exporter les ressources, l'exokernel utilise trois techniques :

— Les liens sécurisés (Secure bindings) entre le matériel et les LibOS.
— La libération de l'utilisation des ressources de manière ouverte et visible.
— Le noyau peut couper tous liens sécurisés vers une ressource quand l'application devient un point indiviuel de défaillance (SPOF) [2].

Voyons une comparaison entre l'architecture des systèmes actuels et celle de l'unikernel Mirage OS. Architecture des exokernels.

2. Single Point Of Failure : point dans un système informatique donc le reste (du système) est dépendant et dont la panne entraine l'arrêt du système

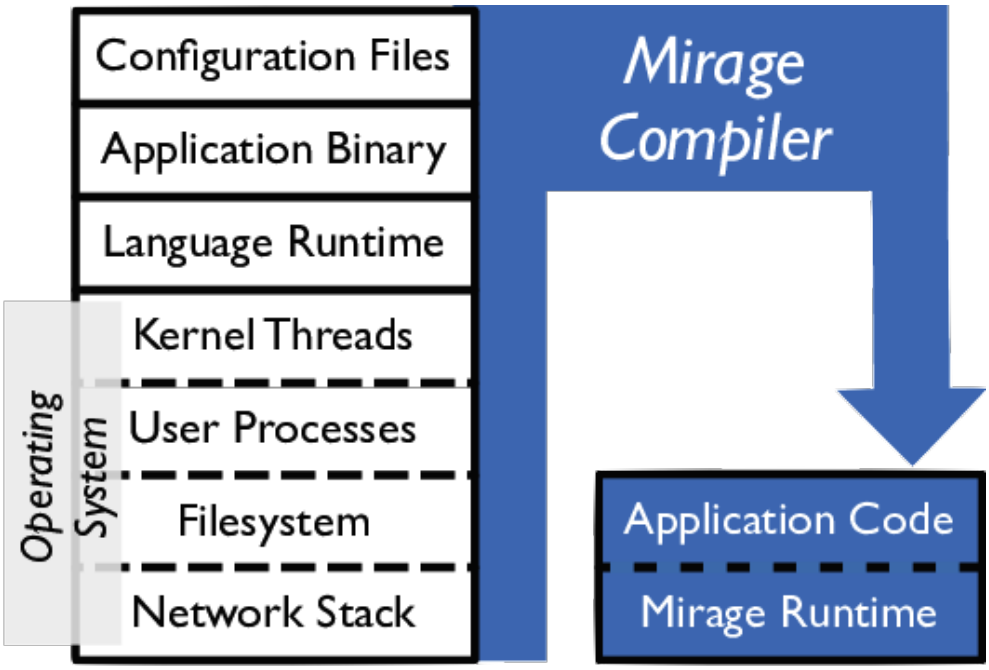

FIGURE 2 – Comparaison Architecture systèmes actuel et celle de Mirage OS[19]. CC BY-SA 3.0

Comme le montre la figure 2, les systèmes d'exploitations habituelles ont plusieurs couches au dessus du noyaux pour faire fonctionner les applications. En regardant les la pile proposée par MirageOS ont peut constater que la le nombre de couche a considérablement été réduite.

Usage des exokernels. Les exokernels trouvent leur utilité dans plusieurs domaines à savoir :

- **Cloud computing** : Application récente avec Mirage qui est créé pour faire fonctionner les applications orientées réseau sécurisées et de haute performance dans le cloud. Il est basé sur l'exo-noyau **Nemesis**. La solution Amazon EC2 est une utilisation de MirageOS
- **Systèmes embarqués :** développement de l'exo-kernel **Camille** et **CamilleRT** pour un fonctionnement en temps réel.
- **Serveur : Xok** X86 est l'exokernel le plus utilisé pour les implémentations en environnement serveur. Xok/ExOS qui fait fonctionner le serveur web **Cheetah** développé par le MIT.
- **Routeur actif** : utilisé dans le projet AMP en 2001, pour faire fonctionner leur interface pour routeur actif. Ils ont développé la libOS **libAMP**.

Nous pouvons aussi remarquer que le développement des unikernels s'oriente également vers les équipements de l'Internet des Objets [5].

2.2 Docker

2.2.1 Définitions

C'est un logiciel libre qui permet d'executer (lancer) facilement des applications dans des conteneurs logiciel[22]. Le concept de Docker est différent de la virtualisation [3], bien que pas très loin, c'est de la conteneurisation [4]. L'approche de Docker permet d'avoir une flexibilité et une portabilité d'exécution des applications afin de leur permettre de s'exécuter sur plusieurs environnement (cloud, machine locale, machine nue, etc.).

2.2.2 Présentation de Docker

Qu'est ce qu'un conteneur ? Un conteneur est une unité standard de logiciel qui regroupe le code et toutes ses dépendances afin que l'application s'exécute rapidement et de manière fiable d'un environnement informatique à un autre. Une image de conteneur Docker est un package logiciel léger, autonome et exécutable, qui inclut tout ce qui est nécessaire à l'exécution d'une application : code, exécution, outils système, bibliothèques système et paramètres[11].
Comme le montre la figure ci-dessus, les applications conteneurisées n'ont pas besoins de système d'exploitation pour fonctionner. Elles tirent au mieux profit d'un seul système d'exploitation et sont plus légère que les machines virtuelles.
Que fait la technologie Docker ? Elle étend le format de conteneur Linux standard, LXC [5], en utilisant une API [6] de haut niveau qui fournit une solution de virtualisation qui exécute de manière isolé les processus. Docker utilise entre autre LXC, cgroups(figure 4) [7] et le noyau linux lui même. Les conteneurs Docker n'intègrent pas en leur sein des systèmes d'exploitations. Les conteneurs Docker s'appuient sur les fonctionnalités5) du système d'exploitation fournies par la machine hôte. La technologie Docker peut être utilisée pour étendre les systèmes distribué [8][8] de sorte à ce que celui-ci s'exécute de manière autonome depuis une seule machine physique ou une instance [9] par noeud [10].

Docker Linux interfaces

3. La virtualisation c'est la capacité de pouvoir exécuter un ou plusieurs systèmes d'exploitation isolés sur une machine hôte(virtualisation système), ou executer des applications (virtualisation applicative)

4. La conteneurisation fait appel à la notion de conteneur qui est la capacité de pouvoir regrouper dans un conteneurs des objets précis. Dans notre cas on regroupe des fonctionnalités et des modules d'un système

5. Linux Container, c'est un système de virtualisation qui utilise l'isolation comme méthode de cloisonnement au niveau du système d'exploitation

6. Application Programming Interface

7. cgroups(control groups) : c'est une fonctionnalité de linux qui permet de limiter, compter et isoler l'utilisation des ressources (processeurs, mémoires ...)

8. Un système distribué est un ensemble d'ordinateurs indépendant qui apparaît à l'utilisateur comme un système unique et cohérent

9. une fonctionnalité, une partie ou un objet qui rentre dans la composition du système globale

10. Un noeud peut être considéré comme une machine

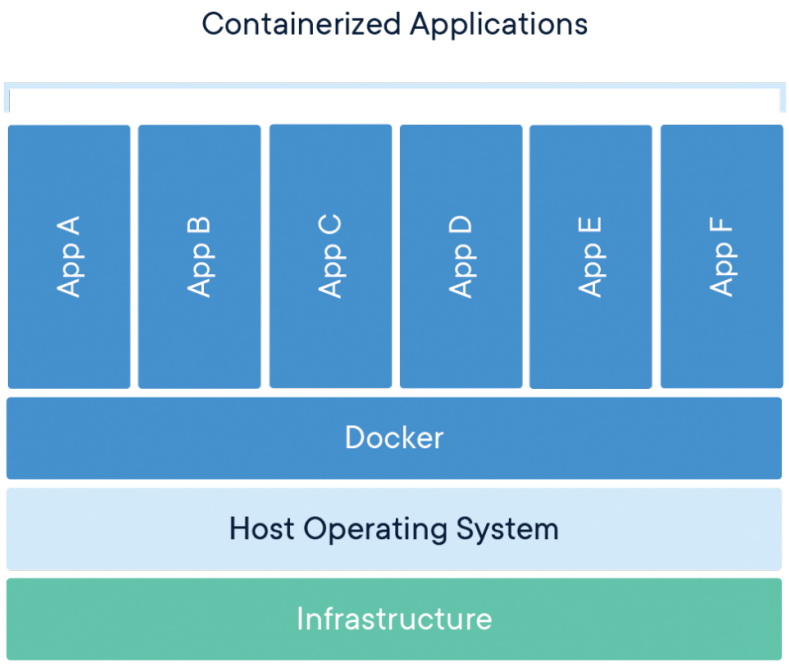

FIGURE 3 – Qu'est ce qu'un conteneur ?[11]. © 2019 Docker Inc. All rights reserved

2.2.3 Autres outils semblables à Docker[12]

Dans le monde des conteneurs Docker peut compter quelques concurrents. Des outils que l'on pourrait utilisé pour créer des conteneurs à savoir : **rkt**[3] lu **rocket**

Avantages rkt

- rkt supporte aussi les conteneurs Docker et permet de convertir n'importe quelle image de conteneur via Quay au format rkt ACI.
- la technologie KVM et Container d'Intel permettent de protéger les conteneurs logiciels les uns des autres.

Inconvénients rkt

- il y'a moins d'intégration tierces comparé à Docker.
- il est optimisé pour faire fonctionner les containeurs d'applications. Les systèmes complets de conteneurs ne sont pas pris en charge.

LXC

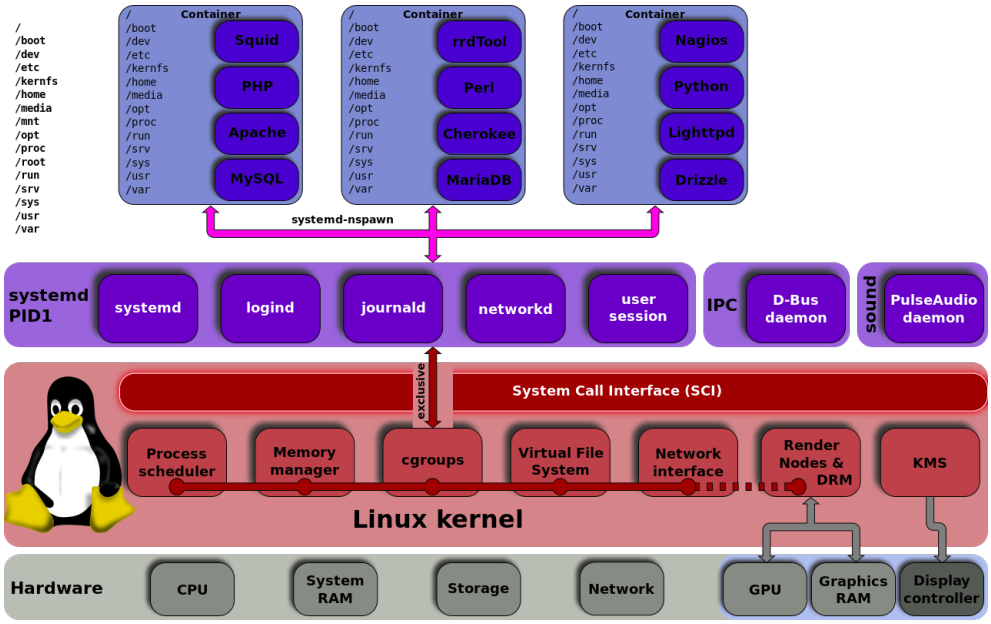

FIGURE 4 – Unified hierarchy cgroups et systemd.[20]. CC BY-SA 3.0

Avantages LXC

— il est optimisé pour l'exploitation de systèmes complet de conteneurs.

Inconvénients LXC

— Le fonctionnement standard de LXC ne prends pas en compte le fonctionnement des conteneurs d'applications.

— Son implémentation native est faite uniquement sur Linux (supporte seulement la plateforme Linux).

LXD de canonical

Prononcé "lexdi" il est une sorte d'hyperviseur de conteneur.

Avantages LXD

— il est optimisé pour l'exploitation de systèmes de complets de conteneurs

— un client LXD peut être configuré pour windows et MacOS et permettre le contrôle à distance du daemon LXD via REST-API.

Inconvénients LXD

— Les conteneurs d'applications ne sont pas considérés comme une application standard.

Linux-VServer[17]

Avantages Linux-Vserver

— il fournit un système de fichier commun pour tous les VPS sur l'hôte dans lequel les états actuels peuvent être sauvegardés.

Inconvénients Linux-Vserver

— il a besoin d'une modification du noyaux Linux.

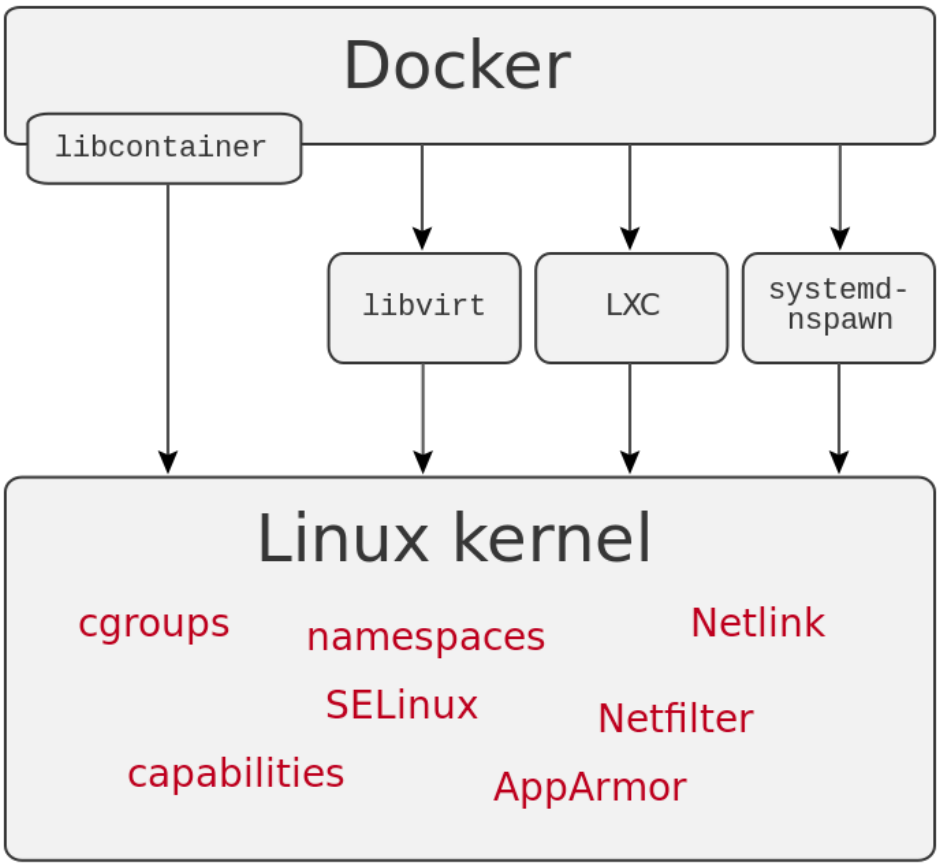

FIGURE 5 – Docker Linux Interfaces.[22]. CC BY-SA 3.0

— plus aucune nouvelle version depuis 2007 (version 2.2 est la dernière connue).

OpenVZ/Virtuozzo 7[9]

Avantages OpenVZ/Virtuozzo 7

— Parallels offre une distribution Linux complète optimisée pour les scénarios de virtualisation avec OpenVZ et Virtuozzo.

— il fait fonctionner les conteneurs de système, des machines virtuelles avec un overhead minimum.

Inconvénients OpenVZ/Virtuozzo 7

— ils sont limités aux distributions Linux RHEL7 et Virtuozzo Linux.

— ils fournissent des conteneurs pour faire fonctionner des systèmes d'exploitation complets. Ils n'isolent pas les processus individuels.

runC

Avantages runC

— il se base sur les standard industriel de l'Open Container Initiative (OCI).

Inconvénients runC

— Des outils externes sont nécessaires pour créer des images de conteneurs.

2.3 Unikernel vs Docker

La containerisation est une forme de virtualisation qui encapsule uniquement l'application et ses dépendances, telles que les bibliothèques et les infrastructures, au dessus d'un système hôte partagé. C'est une virtualisation allégée, sans plusieurs instances du système d'exploitation lourd.
Unikernel est comme un conteneur rétréci. La différence entre unikernel et les conteneurs se trouve dans le fait que les conteneurs nécessitent et dépendent d'un système d'exploitation normal, ce qui n'est pas le cas des unikernels.
Unikernels = code application + composant du SE nécessaire pour exécuter le code de l'application.
Unikernel est utile dans les équipements et dispositifs de l'Internet des Objets (IoT). Unikernel est optimal pour les applications spécifiques. Unikernel est optimal pour les dispositifs ne nécessitant pas de fonctionner avec tout le système d'exploitation. Unikernel dans le cloud, Aws (Amazon Web Services), offre des espaces de stockage, et des machines sans système d'exploitation. L'unité d'allocation commence à 1go et ont peut avoir des unikernels avec des tailles de l'ordre des Kilo octets et Méga Octets.

2.3.1 Pourquoi Docker ?

Le choix de Docker s'est basé sur les éléments ci après :

— La dominance de docker sur le marché des conteneurs ;
— La maturité de docker ;
— La sécurité offerte par les conteneurs docker ;
— Rapidité de déploiement ;
— Réduction de l'infrastructure IT ;
— Légèreté : les conteneurs exploitent et partagent le noyau hôte ;
— Interchangeable : la possibilité de déployer des mises à jour et des mises à niveau à la volée ;

2.4 Articles

2.4.1 unikernels : The Rise of the Virtual Library operating System

Dans les travaux de **Madhavapeddy**[7] sur les unikernels, on peut constater qu'il présente la technologie de virtualisation de système d'exploitation comme étant la technologie de base pour les systèmes clouds.
La virtualisation a permis d'intégrer sur un même hôte sans aucune modification, les applications qui étaient souvent déployées individuellement sur des machines individuels. Les applications qui étaient déployées avaient besoin de très peu de ressources parmis celles qu'offraient la machine.

La virtualisation augmente une couche logiciel en plus au dessus de la pile logiciel déjà existante, ce qui rend le débogage complexe.

Il est nécessaire d'optimiser les machines virtuelles du cloud en réduisant les fonctionnalitées redondantes et en réduisant la surface d'attaque des machines dans le cloud.

Les hyperviseurs fournissent une abstraction de ressources qui leur permet d'augmenter le nombre de coeur ou de machine virtuelles. "L'objectif de MirageOS est de restructurer des machines virtuelles entières (y compris tous les codes du noyau et de l'espace utilisateur) en composants plus modulaires, flexibles, sécurisés et réutilisables dans le style d'un système d'exploitation de bibliothèque"[7].

Plus besoin de système d'exploitation pour agir en tant que multiplexeur mais cependant besoins de grosse interfaces ou API win 32 et posix, d'ou viendra la pile de communication TCP / IP et l'interface de système de fichiers pour le stockage de données persistantes. Les unikernels sont des noyaux d'OS spécialisés écrits dans un langage de haut niveau et qui agissent en tant que composants logiciels individuels. Une application complète consiste en un ensemble d'unikernels en cours d'exécution travaillant comme un système distribué.
La libOS permet aux applications d'accéder directement aux ressources matérielles sans avoir à effectuer des transitions répétées de privilèges pour déplacer des données entre l'espace utilisateur et l'espace noyau.
inconvénient de libOS : Il lui est délicat d'exécuter plusieurs applications côte à côte avec une forte isolation des ressources. Les pilotes de périphériques doivent être réécrits pour s'adapter au nouveau modèle. Ce ne sont pas tout les systèmes d'exploitations et applications qui peuvent utiliser cette abstraction. Les systèmes d'exploitations actuelles ne sont pas optimisés pour la taille et le temps de démarrage

Avantages

- Déploiement et gestion : le temps de configuration et de déploiement et les efforts de déploiement sont réduit pour les applications multiservices complexes.
- Efficacité des ressources et personnalisation : petite taille binaire des unikernels facilite le déploiement sur des centres de données distant, le temps de démarrage inférieure à 1 seconde rends possible le démarrage d'un système pour répondre à une requête.
- Une nouvelle frontière de portabilité : les besoins de la bibliothèque sont codés dans leur environnement d'exécution.

Inconvénients

- ajout d'une autre couche sur un ensemble aynt un nombre grand de couche logiciel
- Grand nombre de déconnexion sur le cloud pour redéployer les unikernels.

3 Proposition de solution

Pour ce sujet, nous comptons compiler un unikernel qui aura le rôle de serveur web et qui pourra être installer sur une machine virtuelle virtualbox sans système d'exploitation. Pour cela nous aurons besoins du logiciel de virtualisation **virtualBox**. virtuelBox est libre et gratuit mais aussi il permet de virtualiser n'importe quel système d'exploitation, ce qui nous donne la possibilité de tester aisément notre unikernel.

3.1 Expérimentation

Pour la phase d'expérimentation, nous prévoyons déployer dans virtualBox, un unikernel conteneurisé capable d'exécuter un serveur web. Ce choix est fait car l'exécution d'un serveur en production demande généralement une plus grande machine, consomme beaucoup de ressources(énergie, espaces, taille...) et est exposé aux problèmes de sécurité.

3.2 Solutions Existantes

A ce stade, il existe déjà la possibilité de faire fonctionner les serveurs web sur virtualBox. La nouveauté viens de ce que nous cherchons à exploiter la puissance combiner des unikernels et de docker pour déployer le serveur web sur virtualBox.

3.3 Réalisation du projet

Pour réaliser ce projet, nous allons nous appuyer sur la technologie docker et celle des unikernels (mirageOS, rumprump, unik), et du logiciel de virtualisation virtualBox.
Les ressources que nous allons également utiliser sont les site internet officiels d'unikernel et de docker. Nous aurons également besoins de nous appuyer sur les conseils et les recommandations de l'encadreur, M. Quang. Nous allons également prendre conseil auprès de M. Vinh pour les remarques et les conseils liés à notre travail. Pendant la réalisation de notre travail, il est possible que pour les phases de test nous fassions appel à l'aide des camarades pour les tests. Pour la relecture du rapport, pour éliminer les fautes de grammaire et autres nous aurons besoins de correcteur pour lire les rapports et nous mentionner les zones qui ont besoins d'être modifiées.

3.4 Estimation de la difficulté de réalisation

Une des difficultés de réalisation que nous pensons rencontrer, consiste premièrement à trouver, le bon mariage entre l'unikernel à utiliser et docker. Par la suite il sera question de pouvoir créer un unikernel du serveur web, de containeuriser l'unikernel, de trouver comment déployer l'unikernel sur virtualBox. Une fois cela réaliser nous allons donc déployer l'ensemble sur virtualBox.

3.5 Estimation du temps nécessaire

Pour estimer le temps de réalisation de notre soulution, nous avons trouver judicieux de séparer les différentes tâches que nous devons exécuter avant de pouvoir arriver à la solution recherchée. Une fois la séparation faite, nous pensons :

- Trouver le bon mariage entre unikernel et docker(mirageOS, rumprump)
- créer l'unikernel du serveur
- Conteneuriser l'unikernel
- Compiler sur virtualBox
- Tester.

Nous n'oublions pas d'inclure les tâches de rédactions du rapport et de rencontres avec l'encadreur pour diriger le travail que nous menons. Les delais pour chacune des tâches mentionnées ci-dessus est d'une semaine en moyenne pour chaque tâche. Le diagramme de gantt ci-dessous nous permet d'avoir une planification du travail restant.

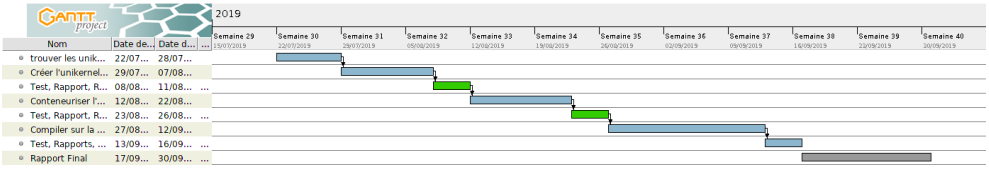

FIGURE 6 – Diagramme Gantt Estimation temps nécessaire

3.6 Conclusion

En définitive, nous pouvons constater que le concept des unikernels a été penser depuis plusieurs années. A l'époque les unikernels s'orientaient déjà vers les systèmes embarquées et les systèmes distribués. Aujourd'hui avec l'internet des objets (IoT), il est plus judicieux d'avoir dans une montre connecté un système pas lourd et sécurisé. D'autres systèmes à fonctionnement autonome comme les drônes, les voitures connectés ... ne sont pas en reste. Pour ce qui est des systèmes distribué, ont constate l'implication des unikernels dans les architectures de clouds computing qui offrent plus de sécurité aux applications lancées dans les serveurs clouds avec un bon temps de réponse aux requêtes des clients.

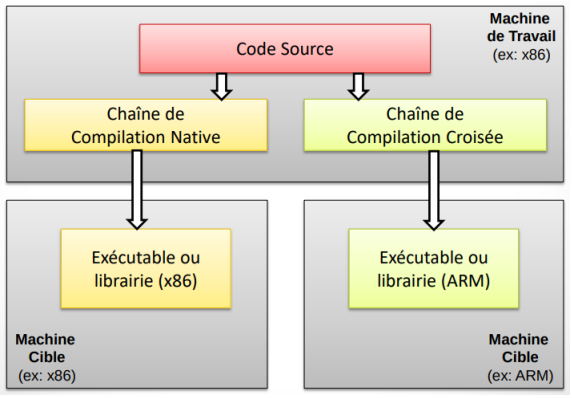

FIGURE 7 – Compilation Croisée

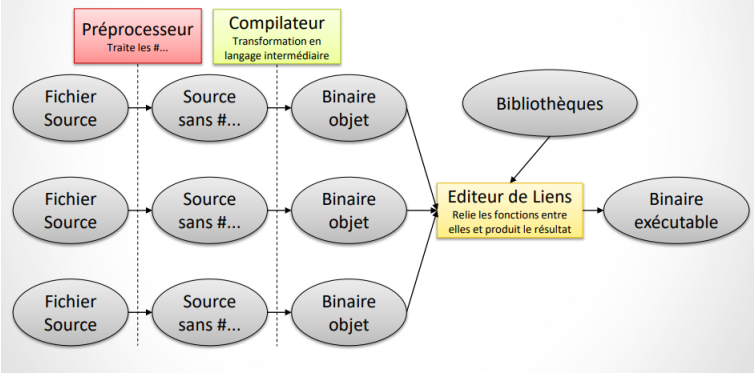

FIGURE 8 – Compilation

4 Pratique

4.1 Implémentation

Pour effectuer notre implémentation nous avons utilisé le projet rumprun unikernel et l'outil Docker. Rumprun Unikernel est un projet basé sur l'unikernel rump. Il donne la possibilité de compiler une application en un unikernel utilisable sur plusieurs architectures.

Notre but est d'utiliser les méthodes de compilation croisé et

Les étapes de la création de l'unikernel sont les suivantes :

— clonage du répertoire git de rumprun
— Mettre à jour les sous-modules du répertoire
— Construction de la chaine d'outils de Rumprun
— Ajout de la chaine de compilation dans la variable d'environnement PATH. export PATH="$PATH$:(pwd)/rumprun/bin"

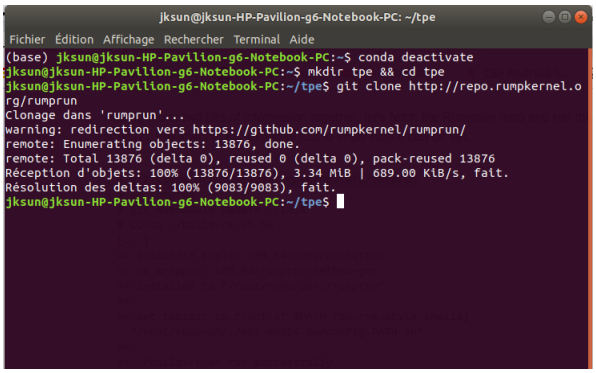

FIGURE 9 – Clonage Répertoire Rumprun

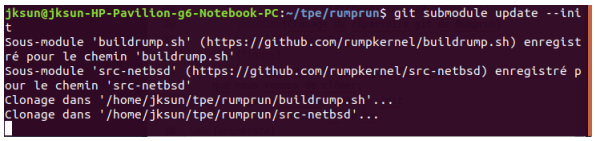

FIGURE 10 – Mise à jour des sous-modules

— Création de l'application
— Compilation de l'application
— Utiliser la chaine de compilation de rumprun
— Utiliser l'option générique de compilation générique.
— conteneurisation de l'application.

4.2 Résultats

Les résultats que nous avons obtenus nous ont permis de :
— Créer un unikernel
— Conteneurisé l'unikernel
Tableau **??** : "*Quelques résultats*".
Unikernel compilé, exécuter dans l'émulateur Qemu :
Unikernel conteneurisé, lancé à l'intérieur du conteneur :

images/metodologia/dockerfileB.png

FIGURE 11 – Dockerfile

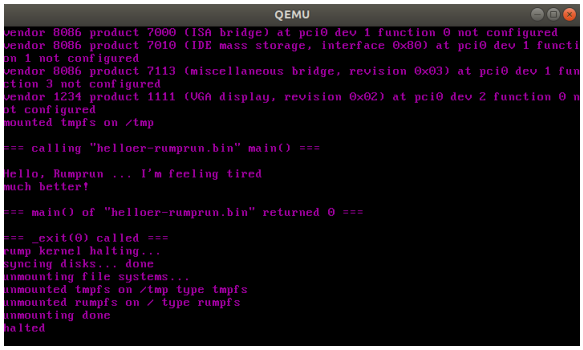

FIGURE 12 – Unikernel lancer sur l'hyperviseur Qemu

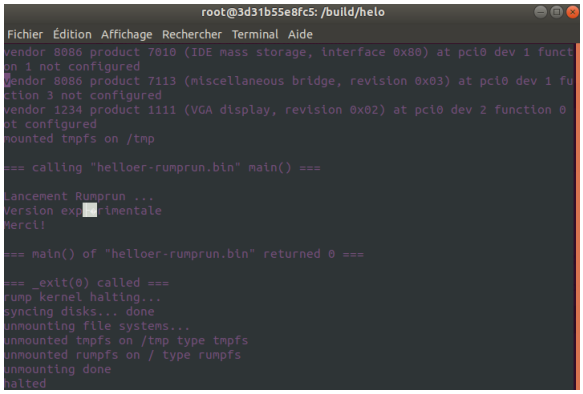

FIGURE 13 – Unikernel lancer dans un conteneur Docker

Description	taille	Plateforme	Comments
Unikernel	80 mégaoctets	x86, ARM	taille du dossier de l'application
executable binaire	26.5 mégaoctets	x86, ARM	taille de l'application binaire exécutable
Conteneurs	1.33 giga octets	x86,x64	taille de l'unikernel conteneurisé
Image iso	8.91 méga octet	x86,x64	taille de l'unikernel compilé en iso

TABLE 1 – Quelques résultats.

4.3 Conclusion

Après avoir fait créer notre unikernel nous pouvons constater que la taille de l'unikernel est bien plus petites comparer à celle des systèmes d'exploitation conventionnel. En effet, il ne contient dans son noyaux que le code nécessaire pour les routines de bases comme le démarrage du système et les appels systèmes utiles à notre application. Si nous voulons ajouter des fonctionalités, nous devons aller effectuer des modifications dans le code github de rump kernel. Cependant cela n'est pas souvent nécessaire parce que rump kernel a déjà un minimum de routine nécessaire au fonctionnement de base et à la communication sur le réseau des applications compilées.

5 Annexe

Pour voir l'implémentation vous pouvez visiter le lien github suivant :

Références

[1] Copyright © LSUB 2014. Removing (most of) the software stack from the cloud, 2019. [Page consultée le 06-juillet-2019.].

[2] Alfred Bratterud. Includeos - run your application with zero overhead, 2019.

[3] coreos. rkt, 2019. [Page consultée le 21-juin-2019].

[4] Antti Kantee. Rump kernel, 2019.

[5] Antti Kantee. The rumprun unikernel and toolchain for various platforms, 2019.

[6] NEC Laboratories. Clickos, 2019.

[7] Anil Madhavapeddy, David J Scott, J Lango, M Cavage, P Helland, and D Owens. Unikernels : the rise of the virtual library operating system. *Commun. ACM*, 57(1) :61–69, 2014.

[8] M. Mosbah. Modèles et approches formels pour les systèmes distribués. [En ligne ; Page consultée le 21-juin-2019].

[9] openvz.org. Open source container-based virtualization for linux., 2019. [Page consultée le 23-juin-2019.].

[10] Russell C Pavlicek. *Unikernels : Beyond Containers to the Next Generation of Cloud*. O'Reilly Media, 2017.

[11] © 2019 Docker Inc. All rights reserved. What is a container ?, 2019. [Page consultée le 14-juillet-2019.].

[12] 11 IONOS SARL. Alternative à docker : aperçu des technologies de conteneurs, 2019. [Page consultée le 21-juin-2019].

[13] Serdar Yegulalp, senior writer at InfoWorld. Docker kicks off the unikernel revolution, 2019. [Online ; accessed 1-May-2019].

[14] unikernel.org. A programming framework for building type-safe, modular systems, 2019. [Online ; accessed 04-May-2019].

[15] unikernel.org. Projects open source work on unikernels, 2019. [Online ; accessed 04-May-2019].

[16] unikernel.org. The unikernel microvm compilation and deployment platform, 2019. [Online ; accessed 04-May-2019].

[17] Linux vserver. Welcome to linux-vserver.org, 2019. [Page consultée le 23-juin-2019. Dernière modification 13 Janvier 2018, à 23 :38.].

[18] Wikipedia contributors. Unikernel — Wikipedia, the free encyclopedia, 2019. [Online ; accessed 1-May-2019].

[19] Wikipedia contributors. Unikernel — Wikipedia, the free encyclopedia, 2019. [Online ; accessed 20-June-2019].

[20] Wikipédia. Cgroups — wikipédia, l'encyclopédie libre, 2018. [En ligne ; Page disponible le 31-mars-2018].

[21] Wikipédia. Exo-noyau — wikipédia, l'encyclopédie libre, 2018. [En ligne ; Page disponible le 28-décembre-2018].

[22] Wikipédia. Docker (logiciel) — wikipédia, l'encyclopédie libre, 2019. [En ligne ; Page disponible le 21-juin-2019].

[23] Wikipédia. Unikernel — wikipédia, l'encyclopédie libre, 2019. [En ligne ; Page disponible le 12-avril-2019].