容器虛擬化技術發展

Development of Container-based Virtualization Technology

陳詠翰
Yung-Han Chen

中文摘要

容器技術近幾年在雲端虛擬化應用的蓬勃發展之際，以及Docker、Google等國際系統廠商所引領的風潮下迅速發展，也將容器技術系統化建構及運用模式推向了另一個層次，並且促使全球資通訊產業重新思考新型態的應用服務架構。本文將介紹目前容器虛擬化技術與系統應用的發展狀況。

Abstract

In recent years, the rapid development of container-based virtualization technology owes much to the dazzling growth of cloud applications and leading of IT system vendors such as Docker and Google. It also pushes the system construction of container technology and application models to another stage, which also enforces IT industries to rethink about the new type of applications/services architecture。In this paper, the development of container-based virtualization technology and related system level applications are introduced.

關鍵詞(Key Words)

容器 (container)

虛擬化 (virtualization)

作業系統 (operating system)

1．前言

虛擬化技術在現今資通訊技術發展中扮演舉足輕重的角色，虛擬化技術能以軟體將一部主機硬體模擬成多個硬體資源，讓使用者建立各自的虛擬運算環境，執行多個獨立的作業系統和應用程式，藉此提昇主機的使用效率，進而降低硬體設備的投資與維運所需付出的成本。因此有越來越多的企業開始間接或直接使用虛擬化技術。虛擬化技術可分成以Hypervisor為基礎以及以容器(Container)為基礎兩大類。其中Hypervisor主要是實現實體資源抽象(Abstraction)化以及虛擬機資源管控(Virtual Machine Monitor, VMM)，而容器虛擬化技術則屬於作業系統層虛擬化(Operating System Level Virtualization)，主要依賴作業系統內核(kernel)虛擬化的支援，讓多個user-space instances得以在宿主機作業系統(Host OS)內核中各自獨立運行，每個承載user-space instance的資源空間就稱為容器，也有稱之為虛擬引擎(virtualization engine)。宿主機內核本身也具備資源管理功能，盡量讓各個容器間隔離運行，對於其他軟體容器的造成的交互影響最小化。因此對不同容器中所執行的程序所使用的運算、儲存、記憶、網路等資源，就像是

自己專用的一樣。

因為容器是直接以宿主機的資源分配運行，因此不需要再掛載完整的客戶機作業系統(Guest OS)，跟Hypervisor-based的虛擬機(VM)運作比較起來，可以節省相當的資源使用，因此也有稱之為輕量虛擬化技術。然而容器虛擬化技術主要的限制是以容器存在的虛擬機需宿主機作業系統使用相容或支援的內核。容器虛擬化技術在使用上最大的好處就是能實現軟體的即時遷移(Live Migration)與擴容/縮容(Scale-out/Scale-in)，一個容器所包含的軟體物件，能夠即時移動到另一個容器或主機上，再重新執行起來。當原容器運行的負載變重時，可以迅速啟用新的容器並將負載分散過來，提高處理能力；反之當負載變輕時，也可以在短時間內關閉不再使用的容器並釋出資源。

本文以下將針對容器技術相關的發展狀況進一步介紹。

2．容器技術發展現況

容器技術的概念與實現很早就出現，但直到近幾年才在雲端虛擬化應用的蓬勃發展之際，以及Docker、Google...等國際系統廠商所引領的風潮下，將容器技術的發展以及系統化建構及運用模式推向了另一個層次。容器技術發展仍持續增溫中，以下針對目前最受關注、發展也最積極的Docker以及CoreOS rkt (Rocket)概略介紹。

2.1 Docker 容器技術

Docker為首度發表於2013年，是一個基於Apache License 2.0的開源容器運行引擎(engine)與容器資源抽象化的工具，讓開發人員或系統管理員能創建、打包、運行容器中的應用程式。開發者能用Docker分層打包其應用程序以會用到及依存的函式庫或其他程序包，成為所謂的Docker Image，並能夠發佈在任何其他具備Docker Engine的環境，因此可讓應用服務輕易地跨實體機或虛擬機、以及在不同

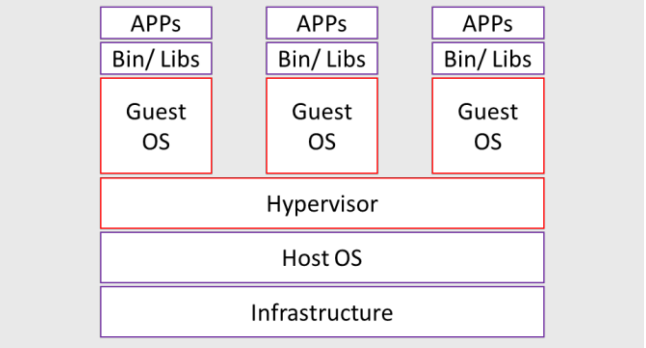

圖 1　Hypervisor-based VM

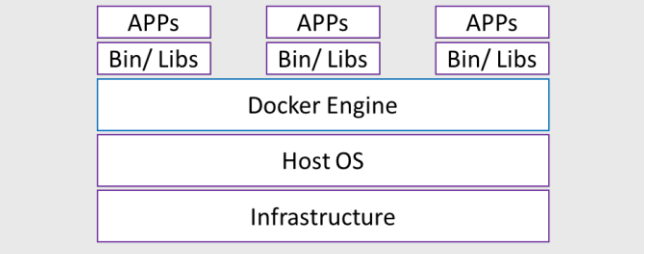

圖 2　Container-based (Docker) VM

Linux發行版的主機上運行。今年7月底所釋出的最新版本為1.12.0。

Docker使用的是容器技術，因此具備前述的「輕量」運行的特質，如圖1與圖2的比較可看出來，因為容器是直接分配宿主機作業系統的資源而運行，因此能以Docker engine將實體資源抽象化後直接分配給應用程序，因為容器內外共用相同的Linux內核，因此可省去客戶機作業系統。

而Docker之所以能實現多容器運行與管理主要運用了以下幾項Linux內核的技術：

(1) cgroup: 可隔離不同程序(組)之間的CPU、記憶體、儲存、網路等資源的使用限制、優先權安排等。利用cgroup，Docker所起的不同容器間在資源使用上就不會彼此影響。

(2) Linux namespace: 可隔離虛擬化的資源運行，包括PID、hostname, userid, network, ipc, filesystem等。任何只關連到某一個namespace的程序就只能看到分配給該namespace的資源。

(3) AUFS: 讓Docker Image得以分層包裝的檔案系統。

(4) Netlink: 用來讓不同Container之間的

行程進行溝通

(5) Linux Bridge: 讓不同Container或不同主機上的Container能溝通

此外，Docker也能進一步整合包括SELinux、Netfilter、AppArmor等開源軟體提昇其網路安全。

Docker可跨宿主機Linux作業系統版本的可攜性對許多程式開發者而言是一大福音，因為可以解決應用程序跨不同Linux發行版運行時常常遇到的相容性問題。程式開發者可以在自己慣用的發行版平台與環境中專注進行開發，並且利用Docker容器打包後發佈到其他支援Docker的Linux發行版中運行，如此一來可省去許多解決相容性問題的人力與時間。另一方面，由於不同的Linux發行版多半有其適合的用途與運作環境條件，例如適合伺服器運作、企業應用環境的發行版，或是能在小型主機或可攜式終端運行的輕量化版，透過Docker也可以輕易地將應用程序部署到各種應用場景中。

除了宿主機直接使用Linux作業系統外，Docker在今年初也推出原生Windows 10專業版與Mac OS X (10.10.3以上)版本的Docker。透過Windows Hyper-V以及Mac的xhyve建立虛擬機執行Alpine Linux中的Docker Engine，進一步擴大Docker應用的版圖。

2.2 CoreOS rkt (Rocket) 容器技術

CoreOS rkt (發音同rocket)，是由CoreOS所推出的開源容器軟體，目前最新版本為1.13.0。除了前面敘述到的容器化特點之外，rkt還強化了以下的特點：

(1) 安全性 – 容器技術最常被關注的議題就是安全性。由於容器屬於作業系統層的虛擬化技術，容器內的應用程序是直接運行在主機的內核上，因此駭客或惡意程序若有辦法透過容器攻擊或甚至侵入內核的話，就會影響到其他容器的運行甚至整個主機。因此Rocket除了限定容器使用者為un-privileged的權限之外，針對正常受信任的容器映像(images)也採用加密簽章的保護措施加以把關。

(2) 更高的可攜性 – 除了同樣可以跨作業系統發行版外，rkt更全力推動跨容器運行環境/引擎的可攜性，包括在Open Container Project (OCP) 下的開放容器標準如Application Container (appc) spec的發展與貢獻，這讓應用程式與服務的發展不會受到容器技術發展的限制，未來若有更好的容器技術推出時，現有的應用服務也可以很輕易地遷移到新技術的平台上。

(3) 插件式容器執行架構(pluggable Runtime Architecture) – rkt在架構設計上可依照應用業務的需求設定與建構不同權限層級的容器運行與隔離度。換句話說，用戶可以依照應用業務的特性而客制化rkt容器運行的屬性與功能。

rkt依據appc規格的定義，將最小部署與執行的單元訂為App Container Pod。一個pod裡可包含若干個應用容器映像 (Application Container Images，ACIs)，每個ACI則包含應用程序執行所需的程式檔案集以及映像設定清單(Image Manifest)。rkt支援appc規格ACI封裝，Docker Image則需透過額外的工具轉換。不同Pods的配置及運作彼此獨立，主要還是運用了多個namespaces的功能而達成的。rkt所採用pod的作法與Google的Kubernetes相同，認為從應用的角度來看，容器映像扮演的角色就好比是零組件，pod則像是多個零組件整合在一個載體上所構成的模組，而此模組可透過參數配置及I/O的設定獨自將功能運行起來，也可以跟其他模組連接進一步構成更完整、更複雜的應用服務系統，而且模組可輕易地抽換升級，某個模組出了問題也不會造成其他模組故障。透過這樣的架構能更輕易實現微服務(microservices)的設計及運行，並且提供企業用戶更有效的DevOps環境。

3 · Container OS 發展

容器技術在運用上已提昇到系統運作與部屬的層面，因此的在基礎的作業系統也有對應的發展，也就是容器作業系統 (Container OS) ，其主要目的就是打造成更適合容器的運

行與大規模部署的專用作業系統。許多知名的系統作業廠商除了支援Docker或rkt容器技術以外，也紛紛基於自家產品推出適合容器運行的作業系統。由於這些作業系統主要就是提供給容器運行用，因此多半會先大幅度的瘦身，移除不需要的功能與套件，並且賦予它們大規模容器部署的能力。以下將介紹目前發展狀況。

3.1 RancherOS (Rancher Labs)

Rancher Labs在2015年初推出一款超輕量的作業系統RancherOS，目的就純粹為了執行Docker Container，因此RancherOS將整個作業系統的架構主要分成system Docker與user Docker兩部分，其中system Docker就是把作業系統的服務都以各自獨立的容器運行，也包含啟動user Docker，而user Docker則是用來創建用戶運行各個應用服務所需的容器。這種作法主要就是為了將Docker作為作業系統的一部份，不像在一般發行版中需另行安裝與啟動Docker Engine/Service。也因此整個作業系統可以大幅縮減大小，目前最新版本為0.5.0版，採用Linux 4.4.10的內核，支援Docker 1.11.2，其安裝檔 (ISO) 僅有40MB。

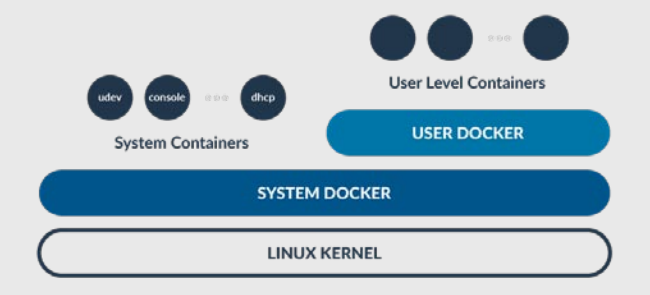

圖 3 RancherOS運行架構 (來源: rancher.com)

3.2 CoreOS (CoreOS)

CoreOS幾乎可說是是最早推出的Container OS，並且也是為了運行容器而進行了大幅的瘦身。此外，CoreOS也提供叢集部署管理的功能與工具，etcd以及fleet，讓用戶能輕易地建立分散式容器運行架構與叢集部署。etcd是分散式鍵-值(key-value)儲存開源軟體，提供跨叢集或主機且可靠的數據儲存。fleet則是一個分散式初始化系統，作法是將systemd與etcd整合到分散式初始化系統中，因此基本上就是像Linux systemd作為系統設定管理程式。但fleet在開發上只是提供最基本的管理、部署、排程的能力，而且有其擴充上限，因此若需要更大規模與複雜的部署與管理，則應考慮採用如Kubernetes這類的協調系統與工具(orchestrator)。

CoreOS早期十分積極支援Docker的發展，但後來決定發展自己的容器技術路線，並且推出rkt容器技術。CoreOS的發展也獲得Google的重視，包括Google創投基金的支援，以及成為Google運算引擎(Google Compute Engine，GCE)中預設的映像檔之一。

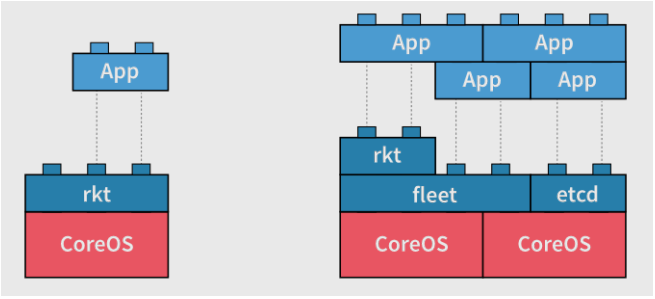

圖 4 CoreOS部署架構 (來源: coreos.com)

3.3 Atomic Host (Red Hat)

Red Hat在2015年3月首度推出專為Docker容器而打造的Red Hat Enterprise Linux 7 Atomic Host，其目的就是在整合Linux Docker Kubernetes (LDK) 的應用架構。除了RHEL 7，Atomic Host也可採用Red Hat其他關係作業系統為基礎，包括CentOS與Fedora。如同其他的Container OS發行版一樣，Red Hat也針對Atomic Host做了大幅的瘦身，但因仍保留原作業系統的安全性功能、版本控制功能、管理伺服器功能等，因此仍比其他發行版要大，但對於許多企業應用仍是主要考慮的選擇之一。

3.4 Photon OS (VMware)

Photon是VMware在vSphere運行環境中所優化且最小化的Linux container host，主要是提供其vSphere客戶能更有效的運行其容器打包

的應用服務程序。Photon OS主要也是採用Docker搭配Google Kubernetes的組合。

3.5 Windows Nano Server (Microsoft)

Windows Nano Server是微軟雲端作業系統Windows Server 2016的精簡伺服器版本，是為執行雲端原生應用程式及Container而設計，包括支援Hyper-V Container以及Windows Server Container。此外也為雲端環境及DevOps工作流程進行優化。為了達成輕量化與簡化的目的，除了疑除非必要的運作元件外，也限制僅限64位元的應用程式。

3.6 Snappy Ubuntu Core (Canonical)

作為最廣為使用的Linux發行版之一，Ubuntu的開發商Canonical也針對大規模雲端以及物聯網中容器運行所需的基礎作業系統推出輕量化的解決方案。事實上，Snappy Ubuntu Core原本是設計給雲端運算及物聯網應用的作業系統，也為了能更有效率運行Docker Container而進行了許多設計與優化，包括在架構上採用不同於一般Ubuntu的設計，如下圖所示。

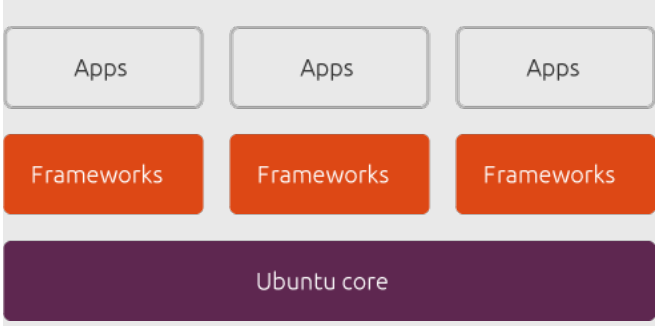

圖 5　Snappy Architecture (來源: http://www.ubuntu.com/cloud/snappy)

Snappy Ubuntu Core由3層不同的架構，分別為Ubuntu Core、Framework、以及Apps，其中Ubuntu Core層由Canonical提供，Framework層則是Canonical與客戶廠商合作，依據客戶的需求而開發。至於Apps層則是客戶廠商自行開發提供。在Ubuntu官網中，Snappy Ubuntu Core可以在Respberry Pi 2、Intel NUC、以及Samsung ARTIK 5 / 10上運行，而這幾款主機平台都能運用於物聯網之中，而且未來可能會有更多不同實體架構的主機能運行Snappy Ubuntu Core。Framework層則提供應用程式運作的基礎，如目前所支援的Docker。Apps應用程序則是在Frameworks中運行。以Docker打包的容器映像都可以在這樣的環境中部屬。

4．容器技術系統應用發展

容器技術要真正進到營運或商用，還需要上層容器資源調度管理與協調的機制。Docker在最初推出之際，就與Apache Mesos整合，進一步對各個容器所配置的CPU數、記憶體容量、儲存容量、以及其他實體或虛擬資源進一步抽象化以及調度管理，並且再更進一步透過Marathon這套開源軟體來啟閉、控制、管理Docker Container運行，實現某種程度的服務協調 (orchestration)。此種整合模式在經過一段時間驗證後證明其可行性，而且也有越來越多的企業採用此種組合模式，因此Mesosphere便將這樣的組合進一步發展到成為足以成為數據中心的資源管理平台與系統，例如Mesosphere的Enterprise Data Center OS，以及OpenDC/OS (由Mesosphere主導的open source DC/OS project)。

Google在Docker 1.0發表之時就宣布在其雲端系統上支援Docker，並且也發表自家Container Orchestration的系統，稱為Kubernetes，這是一套開源系統，目的在自動部署、擴容與管理容器化的應用。如前所述，Kubernetes對於微服務應用需求定義了管理的最小單位pod，一個pod內可包含一個或多個容器。由Kubernetes所定義的服務(Service)則是抽象化的結果，其中定義了相關連的pods以及存取的規則或策略。Kubernetes依據這樣的型態提供應用業務的建構、啟閉、管理、協調、以及業務運行方面的調控。事實上，這樣的方式特別適合微服務架構的運行與管理。

Microsoft對於Docker的支援也越來越積極。在2015年9月舉行的Azure年度技術會議AzureCon發表結盟Mesos推出可支援Docker的雲端Container服務，提供用戶以Apache Mesos

及Docker為基礎的雲端容器運行環境，結合Azure的巨型規模（Hyperscale）運行架構與服務，讓用戶能在多個主機上部署與配置Mesos叢集，並且對Docker容器化的應用程式排程，實現Container協作與管理。此一新服務其實包含多項開源計畫，如Docker、Mesos以及Mesosphere的資料中心作業系統（Data Center Operating System，DC/OS），未來也將支援Windows Server Container。

5．結論

根據iThome 2015 CIO大調查統計顯示台灣企業平均超過14.5%的應用使用雲端環境，而且超過53.4%的企業有意願採用雲端，雲端平台特性與企業類型、業務內容、服務型態，直接影響雲化的可行性與比例。一家企業要成為雲端企業平均需時4.2年。若能提供可靠、安全、彈性化的平台技術與解決方案將可加速企業應用雲端化，提高營運效率。此外，有越來越多的創新構想需要更快速的被實現與持續開發驗證，並且需要具備高度跨平台的可移植性以及分散式的運行，同時也需高度支援DevOps。而容器技術具備許多特質適合需要迅速開發、部署、運行、反應，並且對於微服務類型的應用架構有明顯的效能表現。多家公有雲的廠商，包括Google Cloud，AWS，Azure，都已支援容器的雲端運行。容器技術以及相關的應用與產業發展值得我們持續關注。

參考文獻

[1] Docker (https://www.docker.com/)
[2] Apache Mesos (mesos.apache.org/)
[3] Mesosphere (mesosphere.com)
[4] DC/OS (dcos.io)
[5] Kubernetes (kubernetes.io)
[6] 6 大 Container OS 介紹 (http://www.ithome.com.tw/news/95756)
[7] Windows Nano Server (http://www.ithome.com.tw/news/95103)
[8] Photon OS (https://vmware.github.io/photon)
[9] CoreOS Rocket 1.0 Container Runtime Engine Comes Gunning For Docker (http://www.techtimes.com/articles/131071/20160206/coreos-rocket-1-0-container-runtime-engine-comes-gunning-for-docker.htm)
[10] Visakh S, “What is Rkt (Rocket) container technology? Should you use it?”13 April, 2016 (https://bobcares.com/blog/rkt-rocket-container-technology-use/)
[11] CoreOS rtk (coreos.com/rtk)
[12] Snappy Ubuntu Core (https://developer.ubuntu.com/en/snappy/)
[13] Red Hat Atomic Host (http://www.projectatomic.io/)

作者簡介

陳詠翰

2007年取得國立交通大學電信工程學系博士學位後，即進入工研院資通所從事資通領域相關之研發。研究領域包括行動通訊網路標準協定、行動通訊系統組網技術與系統整合、網路測試與優化、網路功能虛擬化技術開發