УКД 004.738

Слюсарь О.І., студентка,
Слюсарь І.І., к.т.н., доцент,
Тузниченко В.О., студент,
Полтавський національний технічний
університет імені Юрія Кондратюка

МІКРОСЕРВІСНА АРХІТЕКТУРА НА ОСНОВІ ВІРТУАЛІЗАЦІЇ

Анотація. *Для забезпечення користувачів необхідним рівнем обслуговування з боку бізнес-додатків, спрощення управління IT-інфраструктурою, підвищення безпеки використовують її віртуалізацію. В роботі проведений аналіз властивостей різних видів даної технології, з прикладами програмного забезпечення що їх реалізують. Основна увага приділена контейнерній віртуалізації з використанням програмного забезпечення Docker.*

Ключові слова: *віртуалізація, віртуальна машина, гіпервизор, операційна система, docker.*

Вступ

В сучасних умовах, IT-сфера розвивається стрімкими темпами та прагне спростити життя, а on-line сервісів стає все більше. Люди вже звикли замовляти послуги через Інтернет, купувати квитки на кіно та викликати таксі за телефоном, більшість сайтів переглядати з мобільного частіше, ніж з комп'ютерів або ноутбуків. При цьому, існуючі технології Web 2.0 засновані на інтерактивності, під якою розуміється як обмін інформацією між користувачами; між користувачем і постачальником послуги; між самими постачальниками послуг.

В свою чергу, для забезпечення користувачів необхідним рівнем обслуговування з боку бізнес-додатків, спрощення управління IT-

інфраструктурою, підвищення безпеки використовують її віртуалізацію. Як наслідок, надалі доцільно проаналізувати види зазначеної технології та переваги, які можливо отримати при її застосуванні.

Основна частина

Традиційний підхід при організації серверної інфраструктури має на увазі використання для кожної програми окремого фізичного сервера. Це дозволяє гарантовано забезпечити такий додаток необхідними обчислювальними ресурсами при піковому навантаженні, а також ізолювати цей додаток від інших додатків, щоб збій одного з додатків не впливав на роботу інших [1].

Однак подібна стратегія пов'язана з лінійним зростанням числа фізичних серверів і, як наслідок, зі збільшенням витрат на придбання і експлуатацію устаткування.

Тим часом, середнє завантаження обчислювальних потужностей при такій схемі використання серверного обладнання не перевищує 10%, але це вважається явним марнотратством.

Вирішити цю проблему дозволяє віртуалізація серверної інфраструктури.

Згідно [1], віртуалізація – надання набору обчислювальних ресурсів або їх логічного об'єднання, абстрагованих від апаратної реалізації, і забезпечує при цьому логічну ізоляцію один від одного обчислювальних процесів, які виконуються на одному фізичному ресурсі.

Сьогодні ж, будь-який додаток – це в першу чергу додаток, написаний на деякому програмному інтерфейсі програми (Application Programming Interface, API), який знаходиться під управлінням ОС. Завдання ж ОС – надати даним API безпосередньо доступ до апаратних ресурсів.

Сутність технології полягає в наступному. Спочатку на фізичний сервер встановлюється спеціальна операційна система (ОС), яка називається гіпервізором. Потім «поверх» гіпервізора встановлюється одна або кілька гостьових операційних систем, в кожній з яких може бути розгорнуто свій

додаток.

З точки зору гостьової ОС сервер з гіпервізором виглядає як сервер, який складається з «віртуальних» стандартизованих серверних компонентів (процесори, пам'ять, контролери дискової підсистеми, жорсткі диски та ін.), хоча «реальні» компоненти фізичного сервера можуть бути якими завгодно. Сукупність таких «віртуалізованих» серверних компонентів, гостьової ОС і додатків називається віртуальною машиною (VM). На одному фізичному сервері може бути розміщено кілька VM.

Таким чином, гіпервізор ізолює гостьові ОС від апаратного сегменту комп'ютерної системи та забезпечує розподіл ресурсів сервера між VM.

Витрати на забезпечення роботи гіпервізора невеликі – близько 3% від обчислювальних ресурсів сервера. Але завдяки тому, що тепер можна використовувати один сервер одночасно для кількох додатків, віртуалізація дозволяє підняти ККД сервера з 10 до 70%. Через це немає необхідності для кожного нового додатку виділяти новий сервер. Більш того, кількість серверів можна навіть зменшити.

Технологія віртуалізації може випростовуватись за кількома напрямами, які визначаються її видом.

Перший напрямок забезпечує віртуалізацію платформ. Під нею розуміють створення програмних систем на основі існуючих апаратно-програмних комплексів, які залежать або не залежать від них. Система, що надає апаратні ресурси та програмне забезпечення (ПЗ), називається хостовою (host), а системи, що симулюють її, – гостьовими (guest). Щоб гостьові системи могли стабільно функціонувати на платформі хостової системи, необхідно, щоб програмне та апаратне забезпечення хоста було досить надійним і надавало необхідний набір інтерфейсів для доступу до його ресурсів. Існує кілька видів віртуалізації платформ, в кожному з яких здійснюється свій підхід до поняття «віртуалізація». Види віртуалізації платформ залежать від того, наскільки повно здійснюється симуляція апаратного забезпечення. До сих пір немає єдиної угоди про терміни у сфері віртуалізації, тому деякі з наведених далі видів віртуалізації

можуть відрізнятися від тих, що надають інші джерела [2]. В цілому, можливо виділити наступні види віртуалізації платформ [3].

1. Повна емуляція. При такому виді віртуалізації VM повністю віртуалізує все апаратне забезпечення при збереженні гостьової ОС в незмінному вигляді. Такий підхід дозволяє емуляцію різних апаратних архітектур. Наприклад, можна запускати VM з гостьовими системами для x86-процесорів на платформах з іншою архітектурою (наприклад, на RISC-серверах компанії Sun). Довгий час такий вид віртуалізації використовувався, щоб розробляти ПЗ для нових процесорів ще до того, як вони були фізично доступними. Такі емулятори також застосовують для низькорівневого налагодження ОС. Основний мінус даного підходу полягає в тому, що емульоване апаратне забезпечення значно сповільнює швидкодію гостьової системи, що робить роботу з нею дуже незручною, тому, крім як для розробки системного ПЗ, а також освітніх цілей, такий підхід мало де використовується. В якості прикладів продуктів для створення емуляторів слід звернути увагу такі: Bochs, PearPC, QEMU (без прискорення), Hercules Emulator.

2. Часткова емуляція (нативна віртуалізація). В даному випадку, VM віртуалізує лише необхідну кількість апаратного забезпечення, щоб вона могла бути запущена ізольовано. Такий підхід дозволяє запускати гостьові ОС, розроблені тільки для тієї ж архітектури, що й у хоста. Таким чином, кілька примірників гостьових систем можуть бути запущені одночасно. Цей вид віртуалізації дозволяє істотно збільшити швидкодію гостьових систем в порівнянні з повною емуляцією та широко використовується на даний час.

Крім того, з метою підвищення швидкодії в платформах віртуалізації, що використовують даний підхід, застосовується спеціальний «прошарок» між гостьовою ОС і обладнанням (гіпервізор), що дозволяє гостьовий системі безпосередньо звертатися до ресурсів апаратного забезпечення. Гіпервізор (іноді використовується синонім – «Монітор віртуальних машин» (Virtual Machine Monitor)) – одне з ключових понять в галузі віртуалізації. Застосування гіпервізора, що є ланкою узгодження між гостьовими системами та апаратурою,

суттєво збільшує швидкодію платформи, наближаючи її до швидкодії фізичної платформи.

До недоліків даного виду віртуалізації можна віднести залежність VM від архітектури апаратної платформи.

Приклади продуктів для нативної віртуалізації: VMware Workstation, VMware Server, VMware ESX Server, Virtual Iron, Virtual PC, VirtualBox, Parallels Desktop та ін.

3. Часткова віртуалізація, а також «віртуалізація адресного простору» (Address Space Virtualization). При такому підході, VM симулює кілька примірників апаратного оточення (але не все), зокрема – адрес простору. Такий вид віртуалізації дозволяє спільно використовувати ресурси та ізолювати процеси, але не дозволяє розділяти екземпляри гостьових ОС.

Строго кажучи, при такому вигляді віртуалізації користувачем не створюються VM, а відбувається ізоляція будь-яких процесів на рівні ОС. На даний час, багато з відомих ОС використовують такий підхід. Прикладом може послужити використання UML (User-mode Linux), в якому «гостьове» ядро запускається в просторі користувача базового ядра (в його контексті).

4. Паравіртуалізація. При застосуванні паравіртуалізації немає необхідності симулювати апаратне забезпечення, однак, замість цього (або на додаток до цього), використовується спеціальний програмний інтерфейс (Application Programming Interface, API) для взаємодії з гостьовою ОС. Такий підхід вимагає модифікації коду гостьової системи. Системи для паравіртуалізації також мають свій гіпервизор, а API-виклики до гостьової системі, називаються «hypercalls» (гіпервізори). Багато хто сумнівається в перспективах цього підходу віртуалізації, оскільки, в даний момент, усі рішення виробників апаратного забезпечення щодо віртуалізації спрямовані на системи з нативною виртуалізацією, а підтримку паравіртуалізації доводиться шукати у виробників ОС, які слабо вірять в можливості пропонованого їм рішення. На даний час, провайдерами паравіртуалізації є компанії XenSource і Virtual Iron, які стверджують, що швидкодія паравіртуалізації вище.

5. Віртуалізація рівня додатків. Цей вид віртуалізації не схожий на всі інші: якщо в попередніх випадках створюються віртуальні середовища або віртуальні машини, які використовуються для ізоляції додатків, то, в даному випадку, сам додаток поміщається в контейнер з необхідними елементами для своєї роботи: файлами реєстру, файлами, призначеними для користувача і системними об'єктами. В результаті, виходить додаток, яке не потребує установки на аналогічній платформі. При перенесенні такого додатку на іншу машину та його запуску, віртуальне оточення, створене для програми, вирішує конфлікти між нею і ОС, а також іншими додатками. Такий спосіб віртуалізації схожий на поведінку інтерпретаторів різних мов програмування (недарма інтерпретатор «Віртуальна Машина Java» (JVM), теж потрапляє в цю категорію). Прикладом такого підходу служать: Thinstall, Altiris, Trigence, Softricity.

6. Віртуалізація рівня ОС. Суттю даного виду віртуалізації є віртуалізація фізичного сервера на рівні ОС з метою створення кількох захищених віртуалізованих серверів на одному фізичному. Гостьова система, в даному випадку, розділяє використання одного ядра хостової ОС з іншими гостьовими системами. VM являє собою оточення для додатків, що запускаються ізольовано. Даний тип віртуалізації застосовується при організації систем хостингу, коли в рамках одного примірника ядра потрібно підтримувати кілька віртуальних серверів клієнтів.

Враховуючі необхідність реалізації комунікації та контейнеризації мікросервісної архітектура на основі віртуалізації необхідно більш детально розглянутий останній наведений підхід.

Контейнерна віртуалізація не використовує VM, а створює віртуальне оточення з власним простором процесів і мережевим стеком. Примірники просторів користувача (часто мають назву: контейнери або зони) з точки зору користувача повністю ідентичні реальному сервера, але вони в своїй роботі використовують один екземпляр ядра ОС. Для Linux-систем, ця технологія може розглядатися як поліпшена реалізація механізму chroot. Ядро забезпечує

повну ізольованість контейнерів, тому програми з різних контейнерів не можуть впливати одна на одного [4].

Віртуалізація на рівні ОС дає значно кращу, ніж у альтернативних рішень продуктивність, масштабованість, щільність розміщення, динамічне управління ресурсами, а також легкість в адмініструванні.

Найбільш поширені зараз OpenVZ, LXC, FreeBSD jail і Solaris Containers.

OpenVZ – реалізація технології віртуалізації на рівні ОС на ядрі Linux. OpenVZ дозволяє на одному фізичному серверу запускати множину ізольованих копій ОС, так званих «віртуальні приватні сервери» (Virtual Private Servers, VPS) або «віртуальні середовища» (Virtual Environments, VE). Оскільки OpenVZ базується на ядрі Linux, на відміну від VM (наприклад, VMware, Parallels Desktop) або паравіртуалізаційних технологій (наприклад, Xen), в ролі «гостьових» систем можуть виступати тільки дистрибутиви Linux. Накладні витрати на віртуалізацію дуже малі, і падіння продуктивності складає всього 1÷3 %, у порівнянні зі звичайними Linux-системами.

LXC (Linux Containers) – система віртуалізації на рівні ОС для запуску кількох ізольованих примірників ОС Linux на одному вузлу. LXC заснована на технології cgroups, що входить до ядра Linux, починаючи з версії 2.6.29. Серед прикладів використання – застосування в PaaS-хостингу Heroku для ізоляції динамічних контейнерів (dynos).

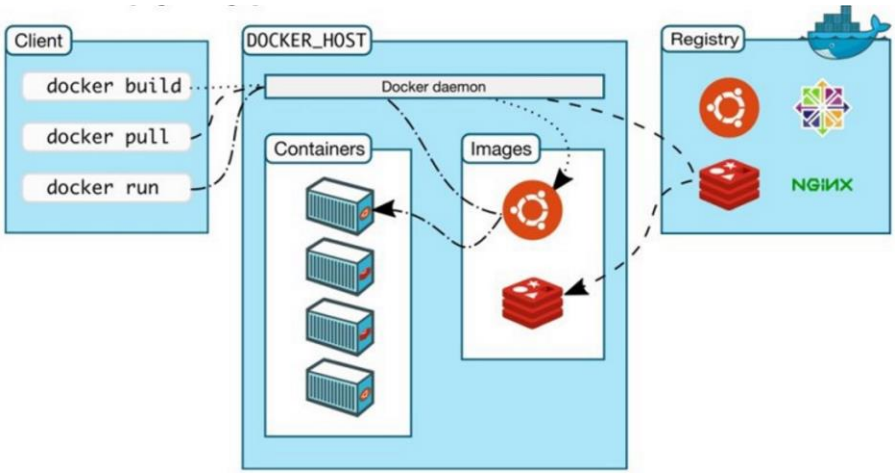

Рис. 1. Структура ПЗ Docker

Docker (рис. 1) – це ПЗ для автоматизації розгортання та управління

додатками в середовищі віртуалізації LXC. Docker дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на будь-які Linux-системі з підтримкою cgroups в ядрі, а також надає середовище з управління контейнерами. Docker, наприклад, використовується такими відомими корпораціями як PayPal, eBay, BBC News, New York Times, Spotify, Uber, Badoo. До складу даного ПЗ входять наступні елементи.

1. Клієнт, який може створювати заготовки, запускати додатки або проштовхувати репозиторії.

2. Хости – це машини по віртуалізації. Тут знаходяться контейнери (рис. 2), в які завантажуються образи.

3. Репозиторії, звідки ми можемо всі ці образи забирати або куди ми можемо ці образи віддавати.

Фактично, користувач отримує відразу хмару і можливість спільної роботи над різними додатками. Недолік такої системи – наявність одної ОС. Тому емулювати ОС Windows на Linux не вийде. Однак, можна використовувати в рамках docker-a Kernel-based Virtual Machine і робити повну віртуалізацію для іншої ОС [5]. Doker працює поверх платформи, що встановлена всередині ОС. Це робить Docker вкрай автоматизованим і універсальним [6].

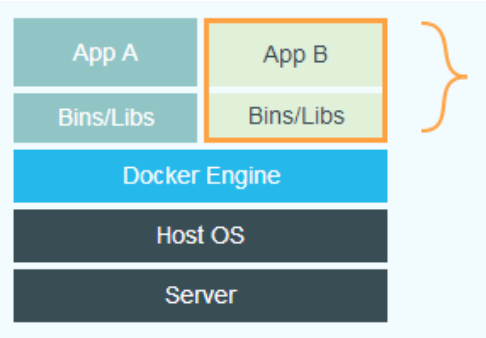

Рис. 2. Структура контейнеру Docker

В ході проведеного аналізу визначено наступні переваги віртуалізації.

Інкапсуляція. Умовно, VM є програмним комп'ютером з повним набором віртуального обладнання, гостьовою ОС і додатками. При виключенні VM записується (інкапсулюється) на диск у вигляді звичайного набору файлів, а при

включенні – зчитується з цього набору. Завдяки інкапсуляції VM можна легко переносити на інший фізичний сервер, клонувати або створювати їх резервні копії на будь-яких пристроях зберігання. Щоб відновити VM після збою, не потрібно заново встановлювати ОС і додатки, досить просто перезапустити її з резервної копії [1].

Ізоляція. При спільній роботі кількох VM на одному фізичному сервері вони повністю ізольовані одна від одної. Це означає, по-перше, що кожна VM може використовувати тільки виділену для неї частину апаратних ресурсів і, як наслідок, не впливає на продуктивність інших VM. По-друге, VM працюють незалежно одна від одної, тому навіть якщо на одній з машин відбудеться збій внаслідок програмної помилки, робота інших машин не буде порушена. Завдяки ізоляції надійність, доступність і безпека додатків, що працюють у віртуальному середовищі, не поступаються характеристикам традиційних не віртуалізованих систем, а часто і перевершують їх.

Сумісність. На відміну від фізичних комп'ютерів, апаратна конфігурація яких може бути різноманітною, VM включають стандартний набір віртуальних «апаратних» компонентів. Як наслідок, VM повністю сумісні з усіма поширеними ОС і додатками для платформи x86. Внесення будь-яких змін в ОС або додатки не потрібно.

Незалежність від обладнання. Оскільки VM не запускаються безпосередньо на фізичному обладнанні, а в середовищі гіпервізора, вони повністю незалежні від конфігурації цього обладнання. Тому, VM разом з їх ОС, програмами та драйверами віртуальних пристроїв можна без будь-яких змін переносити з одного фізичного сервера на інший фізичний сервер із зовсім іншою апаратною конфігурацією.

Висновок

Віртуалізація бізнес-додатків дозволяє запускати кілька додатків на одному фізичному сервері (хості) замість того, щоб виділяти для кожної програми свій

сервер. Цей процес називається консолідацією серверів. Тепер всі необхідні для роботи організації додатки можуть працювати на меншій кількості серверів. Консолідація серверів дозволяє знизити витрати на утримання серверної інфраструктури на 50÷60 %.

Віртуалізація також дозволяє істотно підвищити надійність роботи додатків і їх стійкість до збоїв. У разі відмови одного із серверів розміщені на ньому віртуальні машини можуть бути автоматично перезапущені на іншому сервері і продовжать роботу. Цим способом може бути забезпечена висока доступність, в т. ч., і для таких додатків, які «не піддаються» традиційній кластеризації засобами Microsoft Windows Server.

Більш того, для критично важливих додатків, які повинні працювати в безперервному режимі, можна створити на різних фізичних серверах дві VM – основну та її дзеркальну копію. У разі збою основної VM її дзеркальна копія забезпечить безперервність роботи такого додатку.

Кілька фізичних серверів (хостів), на яких працюють VM, можна об'єднати в кластер. Апаратні ресурси всіх серверів кластера утворюють загальний пул ресурсів, який спільно можуть використовувати VM кластера. ПЗ віртуалізації дозволяє автоматично здійснювати балансування навантаження на сервери, що входять до складу кластера, переміщаючи працюючі VM з більш завантажених серверів на менш завантажені. Якщо загальне навантаження на кластер знижується (наприклад, в нічний час), VM можуть бути автоматично «зібрані» на невеликому числі серверів, а інші сервери будуть вимкнені.

Посилання

1. Введение в виртуализацию [Електронний ресурс] – Режим доступу до ресурсу: http://www.team.ru/virt_intro.php.

2. Немного о системах виртуализации [Електронний ресурс] – Режим доступу до ресурсу: http://www.trading-servers.com/knowledgebase.php?action=displayarticle&id=19.

3. Виртуализация: новый подход к построению IT-инфраструктуры [Електронний ресурс] – Режим доступу до ресурсу: http://www.vmgu.ru/articles/Virtualizatsiya-novii-podkhod-k-postroeniu-IT-infrastrukturi

4. Контейнерная виртуализация в Linux [Електронний ресурс] – Режим доступу до ресурсу: https://web-creator.ru/articles/linux_containers

5. Виртуализация на Open source: доклад о новинках и подходах к виртуализации корпоративной инфраструктуры [Електронний ресурс] – Режим доступу до ресурсу: https://habrahabr.ru/company/muk/blog/263043/

6. Docker виртуализация на уровне операционной системы [Електронний ресурс] – Режим доступу до ресурсу: https://www.linuxspace.org/archives/5940

Authors:
Sliusar O.I., Sliusar I.I., Tuznichenko V.O.

MICROSERVICE ARCHITECTURE BASED ON VIRTUALIZATION

Abstract. To provide users with the necessary level of service from business applications, simplify the management of IT infrastructure, enhance security using its virtualization. In this work, an analysis of the properties of different types of this technology, with examples of software that implements them. The focus is on container virtualization using the Docker software.

Keywords: virtualization, virtual machine, hypervisor, operating system, docker.

Авторы:
Слюсарь О.И., Слюсарь И.И., Тузниченко В.А.

МИКРОСЕРВИСНА АРХИТЕКТУРА НА ОСНОВЕ ВИРТУАЛИЗАЦИИ

Аннотация. Для обеспечения пользователей необходимым уровнем обслуживания со стороны бизнес-приложений, упрощения управления IT-инфраструктурой, повышение безопасности используют ее виртуализацию. В работе проведен анализ свойств различных видов данной технологии, с примерами программного обеспечения которые реализуют. Основное внимание уделено контейнерной виртуализации с использованием программного обеспечения Docker.

Ключевые слова: виртуализация, виртуальная машина, гипервизор, операционная система, docker.