ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерних наук

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

На тему: «ДОСЛІДЖЕННЯ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ВІРТУАЛІЗАЦІЇ І КОНТЕЙНЕРІЗАЦІЇ В СИСТЕМАХ ХМАРНИХ СЕРВІСІВ»

Виконав: студент 4 курсу, групи КНД-41
Спеціальність 122 Комп'ютерні науки
(шифр і назва спеціальності)
_____Брагінець Т.В._____
(прізвище та ініціали)
Керівник _____
(прізвище та ініціали)
Рецензент _____

КИЇВ – 2021

ВСТУП

Актуальність обраної теми дослідження. Ми дуже часто стикаємося з поняттям «віртуалізація». У сфері інформаційних технологій дане поняття використовується значно частіше, особливо в останні роки, ніж в інших науках. Віртуалізація застосовується до різних сфер інформаційних технологій, таких як комп'ютерні мережі, хмарні обчислення, кластеризація та ін. Визначення даного поняття часто розпливчасті і сильно узагальнені. Так «віртуальний» у перекладі (з лат. virtualis - можливий) розуміється як такий, який може з'явитися при певних умовах, або існуючий в уяві. Різні словники визначають, поняття «віртуальний» багато в чому схоже: існуючий в уяві, в зображеннях (кіно, тексти, малюнки тощо). При порушенні самосприйняття може ототожнюватися з дійсним, існуючим в реальності. Отже, «віртуальний» - це те, що не існує в реальності.

Однак в сфері інформаційних технологій під «віртуальними» технологіями розуміють цілком конкретні, реально існуючі сервіси, наприклад, віртуальне сховище даних - зберігає певні файли. Поряд з поняттям віртуальність, в сфері інформаційних технологій, часто використовується поєднане поняття - віртуалізація. Окремі джерела під віртуалізацією мають увазі процес створення віртуального середовища. Наприклад, обчислювальна система - віртуальна, а її оперативна пам'ять - віртуалізує ресурс, поміщений в певний адресний простір мікросхем пам'яті. Таким чином, віртуалізація - це створення віртуальної (а не фактичної) версії чого-небудь, наприклад, операційної системи, сервера, пристрою зберігання даних або мережевих ресурсів.

Віртуалізація - надання набору обчислювальних ресурсів або їх логічного об'єднання, абстрагованого від апаратної реалізації, що забезпечує при цьому логічну ізоляцію обчислювальних процесів, які виконуються на одному фізичному ресурсі. Віртуалізація - це метод приховування фізичних характеристик комп'ютерних ресурсів від тих прийомів, якими інші системи, додатки або користувачі використовують ці ресурси. У загальному розумінні віртуалізація означає процес, в якому один комп'ютер представлений користувачеві

віртуалізованої операційної системи іншим комп'ютером або ж, коли один комп'ютер представляється як кілька комп'ютерів зі своєю ОС на кожному.

На сьогоднішній день залишаються невирішеними або недостатньо дослідженими питання, пов’язані з теоретико-методологічними основами дослідження технологій віртуалізації, виокремленням основних особливостей використання технологій віртуалізації.

Актуальність бакалаврського проекту зумовлена потребою розробки теоретичних та практичних напрацювань в сфері застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів .

У зв'язку з цим **метою роботи** є комплексне дослідження застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів.

Для реалізації поставленої мети були сформульовані наступні **завдання:**

– дослідити понятійно-категоріальний апарат в сфері технологій віртуалізації;

– дослідити апаратні, програмні та організаційні особливості реалізації хмарних сервісів;

– здійснити порівняльний аналіз переваг і недоліків існуючих технологій створення хмарних сервісів;

– дослідити перспективи використання технологій віртуалізації і контейнерізації;

– вивчити особливості застосування апаратної віртуалізації;

– здійснити аналіз віртуальних машин;

– дослідити особливості використання гіпервізорів;

– вивчити використання апаратної віртуалізації в контексті інформаційної безпеки;

– дослідити технології контейнерної віртуалізації: особливості застосування;

– порівняти контейнерну та гіпервізорну віртуалізацію;

Об'єкт дослідження – процес застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів.

Предмет дослідження – технології віртуалізації і контейнерізації в системах хмарних сервісів.

Методика дослідження. В ході дослідження були використані такі загальнонаукові методи: спостереження, синтез теоретичної бази, опис, функціональний аналіз. Методологічною основою роботи є сучасна теорія наукового пізнання, яка передбачає комплексне застосування ряду загальнонаукових методів, вибір яких зумовлений особливостями об'єкта, предмета, мети і завдань дослідження.

Джерела дослідження – науково-дослідні, аналітичні праці фахівців в галузі комп'ютерних наук.

Ступінь новизни одержаних результатів – робота доповнює існуючі системні дослідження в галузі застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів. Зроблені висновки можуть бути використані для подальшого вивчення проблематики їх застосування, розробки практичних пропозицій на підприємствах і організаціях. У науковій роботі вдосконалено та систематизовано: понятійно-категоріальний апарат проблеми дослідження; визначення основних понять; виокремлено основні найбільш поширені методи застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів. Набуло подальшого розвитку дослідження застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів.

Практичне впровадження одержаних результатів – проведено комплексне дослідження застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів. Матеріали дослідження можуть бути корисні фахівцям в галузі комп'ютерних наук.

Галузь застосування – галузь комп'ютерних наук.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи складає 65 сторінок.

РОЗДІЛ 1 ТЕОРЕТИЧНО-МЕТОДОЛОГІЧНІ ОСНОВИ ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ВІРТУАЛІЗАЦІЇ

1.1 Понятійно-категоріальний апарат дослідження технологій віртуалізації

За останні десять років в сфері інформаційних технологій спостерігається стрімке зростання популярності хмарних обчислень - комплексу технологій, спрямованих на те, щоб дати користувачеві простий і зручний доступ до обчислювальних ресурсів. Все частіше у розробників програмного забезпечення (ПО) виникає необхідність створювати програми, здатні працювати в хмарному середовищі. Однак, незважаючи на численні переваги, які дають хмарні обчислення, з ними пов'язаний і ряд проблемних питань, що стосуються залежності від постачальника послуг, безпеки даних, зберігання персональних даних та ін. [2].

Історія використання хмарних обчислень почала свій розвиток ще у 50 - 60 ті рр. минулого століття. Обчислювальна інфраструктура у той час надавалася в якості сервісу задовго до появи хмарних обчислень. Такий підхід мав назву «комунальні обчислення» - термін, який зараз широко застосовується при описі інфраструктурного рівня хмарних систем [13].

Grid-технології, що реалізують розподілені обчислення на віддалених комп'ютерах, набули розголосу в кінці 1990-х років завдяки проекту пошуку позаземного розуму SETI @ home [32].

Початок широкомасштабного надання послуг з доступу до обчислювальних ресурсів через Інтернет відноситься до серпня 2006 року, коли компанією Amazon.com був запущений сервіс Elastic Compute Cloud. Термін «хмара» своєю появою зобов'язаний діаграмі, що зображує взаємодію користувачів з Інтернет, на яких останній представлявся у вигляді хмари, як якась складна інфраструктура за якою ховаються всі технічні деталі.

Хмарні обчислення (cloud computing) - це комплекс технологій, що забезпечують повсюдний і зручний мережевий доступ до динамічно

масштабуємих обчислювальних ресурсів (процесорного часу, оперативної пам'яті, пристроїв зберігання даних, мереж передачі даних, додатків та ін.).

В ідеалі, користувач в будь-який момент часу і в будь-якому місці, де б він не знаходився, повинен отримати в своє розпорядження апаратно-програмну платформу, характеристики якої можна змінювати «на льоту».

Під апаратно-програмною платформою розуміється єдиний комплекс засобів обчислювальної техніки і системних програм [28]:

З точки зору користувача, хмарні обчислення дають можливість отримувати обчислювальні ресурси по мережі у зовнішнього постачальника у вигляді послуги, оплата за яку проводиться в залежності від обсягу спожитих ресурсів, тобто приблизно так само, як це відбувається з послугами водо- чи електропостачання. При цьому обсяг обчислювальних ресурсів - віртуальний комп'ютер, який користувач отримує в своє розпорядження - оперативно підлаштовується під поточні запити користувача. Зручність доступу до послуги забезпечується підтримкою широкого спектру термінальних пристроїв: персональних комп'ютерів, мобільних телефонів, інтернет-планшетів.

Національний інститут стандартів і технологій США (NIST) виділяє такі обов'язкові характеристики хмарних обчислень 28]:

- самообслуговування на вимогу (self service on demand) - споживач самостійно вибирає, яким набором обчислювальних ресурсів він буде користуватися, і може при необхідності оперативно змінювати цей набір без узгодження з постачальником послуг;

- універсальний доступ по мережі - послуги доступні по мережі передачі даних незалежно від того, яке термінальне пристрій використовує споживач;

- об'єднання ресурсів (resource pooling) - постачальник послуг об'єднує наявні в його розпорядженні обчислювальні ресурси в єдиний пул (pool - загальний котел) для динамічного перерозподілу цих ресурсів між споживачами; при цьому споживачі контролюють тільки основні параметри послуги (наприклад,

обсяг даних, швидкість доступу), а фактичний розподіл наданих ресурсів здійснює постачальник;

- миттєва еластичність (гнучкість) - надаються користувачу обчислювальні потужності можуть оперативно збільшуватися або зменшуватися в автоматичному режимі, виходячи з потреб користувача;

- облік споживання - постачальник послуг автоматично обчислює спожиті ресурси (обсяг збережених даних, обсяг переданих даних, кількість користувачів, кількість транзакцій і т.п.), і на цій основі оцінює обсяг наданих споживачам послуг.

Хмарна інфраструктура являє собою набір апаратних і програмних засобів, що дозволяють досягти п'яти перерахованих вище основних характеристик. Вона може складатися з фізичного і абстрактного рівнів. Фізичний рівень - це апаратні засоби, необхідні для надання хмарних сервісів, вони зазвичай включають в себе сервери, пам'ять і мережеві компоненти. Абстрактний рівень - це програмне забезпечення, розгорнуте на фізичному рівні і наділене основними хмарними характеристиками. Абстрактний рівень розташовується над фізичним.

З точки зору постачальника послуг, найважливішою складовою хмарних обчислень є центр обробки даних або дата-центр (data center), який містить:

- інформаційну інфраструктуру, що служить для обробки і зберігання інформації - основних функцій дата-центру;

- телекомунікаційну інфраструктуру для передачі даних всередині дата-центру, а також між дата-центром і користувачами;

- інженерну інфраструктуру, що забезпечує нормальне функціонування основних систем центру (кондиціонування, безперебійне електропостачання, пожежно-охоронна сигналізація та ін.).

Поняття хмарного сервісу (Cloud Service) є більш широким порівняно з хмарними обчисленнями (Cloud Computing), оскільки включає технології зберігання й іншого сервісу.

Власне термін "хмарний сервіс" передбачає використання спеціальної клієнт-серверної технології використання користувачем ресурсів (процесорний час, оперативна пам'ять, дисковий простір, програмне забезпечення та ін.) групи серверів в мережі, взаємодіючих таким чином, що:

- для користувача вся група виглядає як єдиний віртуальний сервер;

- користувач може прозоро і з високою гнучкістю змінювати обсяги споживаних ресурсів у випадках зміни своїх потреб [5].

Таким чином, щоб систему назвати хмарною, вона повинна задовольняти кількома параметрам: автоматично змінювати обсяг сервісу за запитом користувача, доступ до сервісів повинен бути організований через стандартні протоколи Інтернет, сервіси повинні бути організовані таким чином, щоб одне і теж саме обладнання могло бути основою для надання сервісів різним користувачам, а вартість такого сервісу повинна бути набагато нижче, ніж вартість схожого сервісу на виділеному обладнанні (з використанням традиційних технологій).

Ключовими поняттями, пов'язаними з реалізацією «хмар» є масштабованість і віртуалізація [13]. Масштабованість є здатністю обчислювальної системи збільшувати свою продуктивність так, щоб справлятися зі збільшенням робочого навантаження. Завдяки віртуалізації хмарні обчислення абстрагуються від базової апаратної і програмної інфраструктури. Віртуалізовані ресурси надаються користувачеві через певні інтерфейси (програмні інтерфейси API або сервіси). Така архітектура забезпечує масштабованість і гнучкість фізичного рівня «хмари», ізолюючи зміни в центрі обробки даних від впливу на кінцевого користувача.

Віртуалізація - надання набору обчислювальних ресурсів або їх логічного об'єднання, абстрагованого від апаратної реалізації, і такого, що забезпечує при цьому логічну ізоляцію обчислювальних процесів один від одного, які виконуються на одному фізичному ресурсі [11].

Основним і найбільш популярним прикладом віртуалізації є можливість запуску і підтримки декількох гостьових операційних систем на одній: при цьому, у кожного з екземплярів таких гостьових операційних систем свій набір логічних ресурсів (процесорних, оперативної пам'яті, пристроїв зберігання), наданням яких із загального пулу, який доступний на рівні обладнання, управляє хостова операційна система - гіпервізор.

Також може бути віртуалізована мережева інфраструктура, системи зберігання даних, платформове і прикладне програмне забезпечення.

Існує три основних сфери застосування віртуалізації:

- віртуалізація серверів / робочих станцій (віртуальні машини);
- віртуалізація ресурсів;
- віртуалізація додатків.

Віртуальна машина - це обчислювальне оточення, яке «гостьова» операційна система бачить як апаратне. Проте, це програмне оточення, емульоване програмним забезпеченням хостової системи (гіпервізором).

Віртуалізація (поділ) ресурсів (англ. Partitioning) може представлятися як поділ одного фізичного сервера на кілька виділених частин, кожна з яких надається користувачеві і доступна йому як окремий сервер.

Віртуалізація додатків - процес використання додатка, який було перетворено з такого, що вимагає установки в операційну систему, на такий, що не потребує установки.

Для надання кінцевих ресурсів віртуалізації (віртуальних машини, ресурсів або додатків) відповідно до «хмарної» моделі недостатньо виключно технології віртуалізації, як правило, її доповнюють платформи побудови віртуальної IT-інфраструктури, що включають в себе також і набір додаткових компонентів.

Платформа віртуалізації - комплекс проектів програмного забезпечення, що забезпечує реалізацію віртуалізації інфраструктурних компонентів, таких як віртуалізація мережі, віртуалізація ресурсів персональних комп'ютерів,

віртуалізації сховищ. Адміністратору платформи надається панель управління, з якої здійснюється оркестрація вищезгаданими компонентами. В якості основних платформ, можна виділити наступні: VMware, Openstack, Citrix, а також Microsoft Hyper-V Windows Server. Нижче розглянуті основні компоненти, які зазвичай використовуються платформами віртуалізації.

1.2 Апаратні, програмні та організаційні особливості реалізації хмарних сервісів

Архітектура хмарних платформ включає в себе безліч компонентів [26]; сервіси в системі ізольовані, розділені за наданою функціональністю і повинні підтримувати варіанти розгортання на незалежних фізичних вузлах. Більшість сервісів взаємодіє з обмеженою кількістю суміжних підсистем, проте у всіх хмарних платформах існує сервіс, який використовується усіма підсистемами.

До складу хмарних платформ входить спеціальний сервіс (сервіс ідентифікації), який відповідає за аутентифікацію і авторизацію користувачів і підсистем платформи. Контроль підсистем платформи забезпечується з двома цілями: по-перше, для захисту від підміни апаратних вузлів на скомпрометовані, по-друге, для перевірки прав доступу при прямих зверненнях до ресурсів системи. Чим більше сервісів надає хмарне середовище та чим більше екземплярів підсистем працює в системі, тим більше навантаження лягає на сервіс ідентифікації.

Архітектура хмарних систем складається з трьох основних рівнів, яким відповідають три види хмарних сервісів:

- інфраструктура як сервіс (Infrastructure-as-a-Service, IaaS);
- платформа як сервіс (Platform-as-a-Service, PaaS);
- програмне забезпечення як сервіс (Software-as-a-Service, SaaS) (рис. 1).

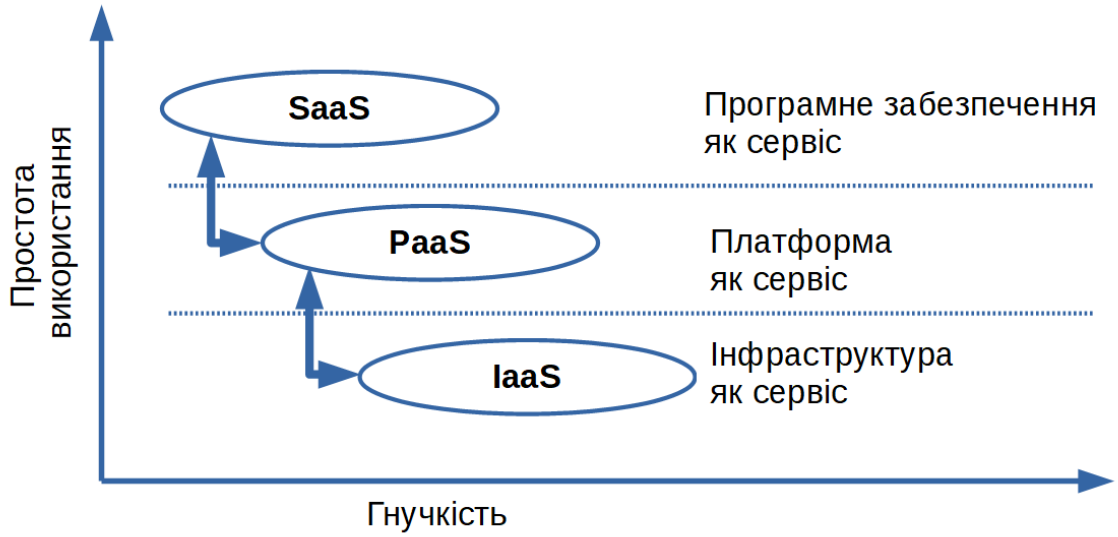

Рис. 1. Рівні хмарних обчислень

Хмарні обчислення можуть забезпечити три види режимів служби, включаючи IaaS, PaaS і SaaS. Тут SaaS означає, що послуга, надана клієнту, є додатками, що працюють на інфраструктурі хмарних обчислень, забезпеченої постачальниками послуг.

До них можна отримати доступ інтерфейсами тонкого клієнта, такими як браузер та ін. PaaS дається користувачеві для розгортання на хмарній інфраструктурі, створюваній споживачами, або для запуску додатків, побудованих при використанні мов програмування і інструментів, які підтримуються провайдером. Споживач не керує і не контролює базову хмарну інфраструктуру, включаючи мережу, сервери, операційні системи або системи зберігання, але керує розгорнутими додатками і, можливо, додатком, що визначає конфігурацію операційного середовища. IaaS пов'язаний з послугами для користувачів, які хочуть орендувати обчислювальні потужності, обсяг в системах зберігання, мережу та інші основні комп'ютерні ресурси - їх користувачі можуть розгорнути і виконати на них будь-яке програмне забезпечення, включаючи операційні системи і додатки.

Існує чотири моделі розгортання (Deployment Models):

- Приватна хмара (Private cloud). Хмарна інфраструктура функціонує цілком з метою обслуговування однієї організації з багатьма споживачами (наприклад, відділами). Інфраструктура може належати, управлятися і контролюватися як самою організацією, так і третьою стороною і перебувати в самій організації чи у зовнішнього провайдера.

- Хмара спільноти (Community cloud). Хмарна інфраструктура функціонує тільки для певної спільноти споживачів з організацій, які поділяють спільні принципи (наприклад, місію, вимоги до безпеки, політики, вимоги до відповідності). Інфраструктура може належати і управлятися однією чи кількома організаціями зі спільноти, третьою стороною або деякою їх комбінацією, перебувати у самих організаціях або поза ними.

- Публічна хмара (Public cloud). Хмарна інфраструктура функціонує в якості загальнодоступної для всіх. Вона може належати і управлятися бізнесом, академічними або урядовими організаціями, або якимось їх поєднанням і знаходитись у провайдера.

- Гібридна хмара (Hybrid cloud). Хмарна інфраструктура є поєднанням двох і більше різних хмарних інфраструктур (приватних, загальних або публічних), що залишаються роздільними одиницями, але об'єднаними разом стандартизованими технологіями, що забезпечують перенесення даних і додатків (наприклад, кілька хмар для забезпечення балансування навантаження між ними).

В даний час існують кілька платформ хмарних обчислень, у кожного є відмінні риси та переваги. Щоб краще зрозуміти ці платформи, проведемо їх докладний порівняння щодо різних аспектів реалізації. Характеристики та особливості даних платформ наведені у таблиці 1:

Таблиця 1. - Порівняльні характеристики декількох платформ хмарних обчислень

Параметри	Eucalyptus	OpenNebula	Nimbus	AbiCloud
Характер хмари	Публічне	Приватне	Публічне	Публічне/приватне
Маштабування	Маштабуєме	Динамічне, Маштабуєме	Маштабуєме	Маштабуєме
Форма хмари	IaaS	IaaS	IaaS	IaaS
Сумісність	Підтримка EC2, parS3	Відкрите, багатоплатформне	Підтримка EC2	Не має підтримки EC2
Розгортання	Динамічне розгортання	Динамічне розгортання	Динамічне розгортання	Пакетне розгортання
Спосіб розгортання	Командна строка	Командна строка	Командна строка	Веб-інтерфейс
Переносимість	Загальний	Загальний	Загальний	Легкий
Підтримка віртуальних машін	Xen,KVM иVMware vSphere, ESX иESX	Xen, KVM, VMware	Xen, KVM	VirtualBox, Xen, VMware,VM
Веб-інтерфейс	Веб-сервіс	EC2,WSDL, WSRF	Libvirt, EC2, OCCI, API	Libvirt
Надійність	-	Відкати хостів і віртуальних машин	-	-
Підтримка операційних систем	Linux	Linux	Linux	Linux
Розвиток	Java	Java	Java, Python	Ruby, C++, Python

З проведеного аналізу користувачі можуть краще зрозуміти характеристики і надійніше вибирати платформу хмарних обчислень, відповідно до протоколів, інтерфейсів, сумісності, реалізації, вимогу щодо розгортання, можливості розвитку та ін. [31]. Хоча у кожної платформи хмарних обчислень є свої переваги, одну річ дуже важливо підкреслити - незалежно від того, яка платформа обрана, залишаються невирішені проблеми. Зазначимо, як приклад, механізми кластерної відмови в хмарному середовищі, гарантії узгодженості і синхронізації на різних кластерах в хмарній платформі, стандартизацію, безпеку

хмарної платформи і даних під час передачі та ін. Всі ці проблеми потребують вирішення для кожної конкретної реалізації.

Завдяки парадигмі IaaS стає можливим віртуалізувати практично будь-яку апаратну складову, що є безумовно корисним при бажанні надати користувачеві значні обчислювальні ресурси віддалено. Такі завдання виникають і в процесі навчання інформаційним технологіям: зручно стає один раз встановити і налаштувати, скажімо, образ операційної системи з потрібними навчальними прикладами, а потім завдяки інтерфейсу, що надається хмарою, запускати цей образ для кожного студента. Подібна «навчальна» хмара стає рішенням проблем, пов'язаних з нескінченим налаштуванням і переносимістю навчальних середовищ.

Для того, щоб впровадити подібну хмару, необхідно перш за все ознайомитися з існуючими хмарними рішеннями. У класичному варіанті хмарної обчислювальної системи можна виділити шість основних компонент [20].

По-перше, є фізичні машини і їх операційні системи. Ми виділяємо їх, тому що: а) якщо процесори, використовувані в машинах, не підтримують технологію апаратної віртуалізації, то ми можемо говорити лише про паравіртуалізацію, і це істотно знижує як обчислювальні здатності системи, так і вибір програмного забезпечення в подальшому; б) системи з відкритим кодом більшою мірою, ніж інші, повинні бути достатньо гнучкими для роботи з багатьма гостьовими ОС (а комерційні можуть бути вузько спеціалізовані на роботу з певним набором програмних компонент).

По-друге, мережева компонента. Використання мережі тягне за собою DNS, DHCP і організацію підмереж фізичних машин. Це також включає віртуалізацію всередині мережі для того, щоб дати кожній віртуальній машині унікальну MAC адресу. Всі компоненти повинні бути відповідно сконфігуровані для роботи з віртуальними вузлами тією ж мірою, що і з фізичними.

По-третє, використання гіпервізора. гіпервізор дозволяє працювати віртуальним машинам. В добавок до моніторингу віртуальних машин, гіпервізор використовує засіб libvirt для того, щоб контролювати запуск і зупинку машин.

Крім того, кожен гіпервізор повинен бути запущений і налаштований індивідуально під кожну машину.

Четверта компонента хмарної інфраструктури — образи віртуальних машин, що зберігаються на віртуальному «жорсткому диску» - репозиторії, що охоплює всі образи всіх машин, запущених колись в хмарі. Природно, необхідно передбачити і архівацію, і належним чином обслуговувати сховища.

По-п'яте, необхідний інтерфейс для користувачів. І, нарешті, власне, хмарне середовище — як система, що зв'язує воєдино всі шість компонент, розподіляє ресурси користувачам, обслуговуюча мережа та ін.

1.3. Порівняльний аналіз переваг і недоліків існуючих технологій створення хмарних сервісів

В даний час можна говорити про існування двох типів хмарних технологій, що відрізняються характером взаємодії з обладнанням: технології, що спирається на віртуалізацію, і технології, яка передбачає пряму лінію і функціонування «хмари» на комп'ютерному «залізі».

Як ми вже зазначали, технології віртуалізації розвиваються, починаючи з середини 1960-х років. У той час ОС називали супервізором (supervisor). Згодом, коли стало можливим запускати одну ОС поверх іншої, з'явився термін гіпервізор (hypervisor), що позначає програму (або, рідше, пристрій), що дозволяє виконувати одночасний запуск декількох ОС на одному комп'ютері.

Гіпервізор здійснює управління ресурсами і їх поділ між різними ОС, виконує ізоляцію запущених ОС один від одного, а також забезпечує їх взаємодію (обмін файлами, мережева взаємодія та ін.). Гіпервізор сам по собі в деякому роді є мінімальною операційною системою. Він надає запущеним під його управлінням ОС сервіс віртуальної машини. При цьому він емулює реальне (фізичне) апаратне забезпечення конкретної машини і управляє цими машинами, виділяючи і звільняючи ресурси для них.

Гіпервізор дозволяє незалежне «включення», перезавантаження, «виключення» будь-якої з віртуальних машин з тією чи іншою ОС. З точки зору гостьової ОС (що працює під управлінням гіпервізора), віртуальна система нічим не відрізняється від фізичної.

Існують три основні типи програмних гіпервізора: автономний, гостьовий і гібридний.

Автономний гіпервізор (VMware ESXi, Xen) здатний працювати на «голому» залізі, тобто не вимагає ОС і безпосередньо надає доступ операційним системам до обладнання. Такий тип гіпервізора забезпечує найкращу продуктивність і тому використовується для роботи з серверними ОС.

Гостьовий гіпервізор працює на основі будь-якої базової ОС та всі операції введення-виведення здійснює через процес користувацького рівня, запущений під цією системою.

Гібридний гіпервізор працює автономно і містить в собі спеціальну сервісну ОС, через яку гостьові системи отримують доступ до устаткування.

Крім віртуалізації за допомогою гіпервізора існує так звана контейнерна віртуалізація. Відмінність між цими підходами полягає в наступному [4].

Гіпервізор емулює апаратне забезпечення, поверх якого запускаються гостьові ОС. При цьому все, що «вміє» робити обладнання, повинно бути доступно гостьовій ОС з боку базової ОС (тобто машини-«господаря»).

Навпаки, контейнери - це віртуалізація на рівні операційної системи, а не на рівні обладнання: кожна гостьова ОС використовує те ж саме ядро (а в ряді випадків - і інші частини ОС), що і базова. В результаті контейнери менше і компактніше гіпервізорних гостьових середовищ, оскільки у них з «базою» набагато більше спільного.

Контейнерна віртуалізація має і свої обмеження. Наприклад, внаслідок того, що використання ядра спільне, на одному сервері не можна запускати гостьові ОС різних типів. Наприклад, на системі з Linux-контейнерами неможливо

запустити FreeBSD або Windows, хоча при цьому можна запускати різні дистрибутиви Linux. Для гіпервізора такої проблеми не існує.

Найбільш відомими відкритими програмними засобами для контейнерної віртуалізації є пакети OpenVZ, LXC і Docker, що представляє собою надбудову над LXC. Зауважимо, що хоча технології віртуалізації і грають важливу роль в організації хмарних обчислень, але в запропонованому NIST списку невід'ємних властивостей цих обчислень поняття віртуалізації не міститься.

Віртуалізація дозволяє при створенні «хмари» абстрагуватися від реального обладнання, наявного в дата-центрі. За це зручність доводиться розплачуватися накладними витратами, що знижують продуктивність системи. Якщо ж вимоги до продуктивності високі, а обладнання досить однорідно, зручніше обійтися без віртуалізації.

Незважаючи на тенденцію до інтеграції обох технологій створення «хмар» (програми для управління високопродуктивними кластерами все частіше використовують віртуальні машини, а ряд хмарних платформ підтримує установку на «голе залізо»), існує ряд спеціалізованих хмарних сервісів, що надають послуги високопродуктивних обчислень, зокрема, IBM SoftLayer. Обидві технології мають свої переваги і недоліки, і вибір між ними диктується потребами проекту. Так, істотний перепад споживаних обчислювальних потужностей з різкими піками споживання, робить кращим використання віртуалізації. Навпаки, стабільне споживання обчислювальних ресурсів з високими вимогами до продуктивності дає перевагу технології безпосередньої роботи з обладнанням.

Застосування хмарних обчислень, поряд з численними перевагами, тягне за собою деякі ризики, пов'язані, в першу чергу, з залежністю користувача від постачальника «хмарних» послуг. Ця залежність набагато сильніше і має більше аспектів, ніж залежність від постачальників традиційного ПО.

Одним з аспектів проблеми є прив'язка користувача до певного постачальника (так званий vendor-lock). Якщо постачальник почне диктувати користувачам неприйнятні умови, то останні будуть змушені або прийняти ці

умови, або перестати користуватися сервісом. Якщо ж постачальник з якихось причин піде з ринку, то разом з ним може зникнути і сам сервіс, який їм надається. При використанні традиційного ПО ці проблеми не виникали: законно придбаний екземпляр програми можна використовувати і після того, як її виробник змінить свої умови або припинить існування.

Одним із способів боротьби з цією проблемою є стандартизація хмарних сервісів. Для розробників ПЗ, що працює в «хмарах», найістотніше наявність стандартного інтерфейсу (API) для роботи з хмарними сервісами. Тут за останні кілька років досягнуто ряд успіхів. З одного боку, існує стандарт де-факто - EC2, з іншого - розроблений і набуває все більшої популярності відкритий стандарт OCCI (Open Cloud Computing Interface) [30]. При цьому все частіше з'являються хмарні платформи, що підтримують обидва стандарти. Іншим способом є ізоляція роботи призначеного для користувача ПО від решти системи, зокрема, за допомогою контейнерної віртуалізації.

Так, за допомогою пакета Docker [21] можна запускати процеси в ізольованому оточенні - своєрідної «пісочниці», де крім самого процесу існують тільки його процеси-нащадки. Хоча при цьому процес працює в тій же ОС, що і інші, звичайні, процеси, він просто їх не бачить. Таким чином, за допомогою Docker розробник може відокремити свій додаток від системи, помістити його в Docker-контейнер і в разі необхідності перенести на іншу однотипну систему.

Наступна група питань стосується безпеки даних. У чиїй власності знаходиться дата-центр? Під чиїм керівництвом? Чи знаходиться дата-центр всередині або поза юрисдикцією власника хмарного сервісу? Широко поширена ситуація, коли компанія-постачальник використовує дата-центри, що знаходяться в різних країнах. При цьому держава, на території якого розташований дата-центр, може отримати доступ до будь-якої інформації, яка в ньому зберігається. Наприклад, за законами США, де знаходиться найбільше число дата-центрів, в цьому випадку компанія-постачальник навіть не має права розголошувати факт передачі конфіденційної інформації будь-кому, крім своїх адвокатів.

У разі хмарних сервісів персональні дані користувачів зберігаються на віддалених серверах, що вимагає більш високого рівня довіри до постачальника. Крім того, хмарні сервіси працюють на неконтрольованих з боку користувача комп'ютерах. Це обмежує можливості вивчення програми в роботі і зворотню розробку з метою забезпечення сумісності.

1.4 Перспективи використання технологій віртуалізації і контейнерізації

Вважається, що хмарні обчислення викличуть нову революцію в IT-сервісах. Передбачаючи величезний бізнес-потенціал такого підходу, багато країн, уряди і корпорації прийняли рішення підтримувати і вкладати кошти в розвиток методів хмарних обчислень. EC2 від Amazon, Azure від Microsoft, AppEngine від Google та ін. - це все широко використовувані платформи хмарних обчислень, які свідчать про те, що хмарні обчислення стануть ареною конкурентної боротьби [3].

Важливим аспектом хмарних технологій є їх економічна доцільність та енергоефективність. Базовою технологією яка реалізує хмарний підхід є технологія віртуалізації.

Нижче розглянуті основні компоненти, які зазвичай використовуються платформами віртуалізації.

1. Управління віртуальними машинами. Модуль - контролер, керуючий роботою віртуальної машини (далі - ВМ). Відповідає за виконання всіх необхідних операцій з підтримки життєвого циклу ВМ в інфраструктурі, реалізує наступний набір функцій і можливостей:

• управління життєвим циклом ВМ;

• управління обчислювальними ресурсами;

• управління мережею ВМ;

• управління ідентифікацією та авторизацією користувачів на ВМ;

• надання інтерфейсу взаємодії RESTbased програмного інтерфейсу додатку (далі — API).

2. Управління Мережею. Модуль, який забезпечує мережеве підключення між пристроями інтерфейсу, керованими іншими службами. Надає можливість створення мережевих топологій і настроювання мережевих політик у хмарі.

3. Управління блоковими пристроями. Модуль зберігання блокових пристроїв, призначений для подання ресурсів зберігання кінцевим користувачам, які можуть бути використані модулем управління ВМ. Він віртуалізує управління пристроями зберігання блоків і надає кінцевим користувачам API самообслуговування запитувати і споживати ці ресурси, не вимагаючи будь-яких знань про те, де їх сховище фактично розгорнуто або на якому типі пристрою.

Фізичні носії даних, диски, можуть бути розташовані всередині або напряму підключені до вузлів модуля, або вони можуть бути розташовані в зовнішніх системах зберігання від сторонніх постачальників.

4. Управління образами гостьових операційних систем (далі — ОС). Модуль управління образами включає виявлення та реєстрацію образів віртуальної машини.

Образи ВМ, доступні через модуль, можуть зберігатися в різних місцях від простих файлових систем до систем зберігання об'єктів.

5. Управління доступом. Модуль управління доступом повинен використовуватися для аутентифікації і авторизації.

6. Серверна ОС. Серверна ОС або «хостова операційна система» - це операційна система, встановлена на реальне обладнання. В рамках цієї операційної системи встановлюється програмне забезпечення віртуалізації, як звичайної програми.

7. Гіпервізор. Гіпервізор - це сутність, яка відділяє операційну систему комп'ютера або додатки від фізичного обладнання. Зазвичай являє собою програмне забезпечення, хоча створюються і вбудовані гіпервізори, наприклад, для мобільних пристроїв. Гіпервізор є рушійною силою концепції віртуалізації, дозволяючи фізичному хост-комп'ютеру керувати кількома віртуальними машинами в якості гостьових ОС, що в свою чергу допомагає максимально

ефективно використовувати обчислювані ресурси, такі як пам'ять, пропускна здатність мережі і кількість циклів процесора.

Висновки до першого розділу

1. Історія використання хмарних обчислень почала свій розвиток ще у 50 - 60 ті рр. минулого століття. Обчислювальна інфраструктура у той час надавалася в якості сервісу задовго до появи хмарних обчислень. Такий підхід мав назву «комунальні обчислення» - термін, який зараз широко застосовується при описі інфраструктурного рівня хмарних систем

2. (а) хмарні обчислення (cloud computing) - це комплекс технологій, що забезпечують повсюдний і зручний мережевий доступ до динамічно масштабуємих обчислювальних ресурсів (процесорного часу, оперативної пам'яті, пристроїв зберігання даних, мереж передачі даних, додатків та ін.); (б) хмарна інфраструктура являє собою набір апаратних і програмних засобів, що дозволяють досягти п'яти перерахованих вище основних характеристик. Вона може складатися з фізичного і абстрактного рівнів. Фізичний рівень - це апаратні засоби, необхідні для надання хмарних сервісів, вони зазвичай включають в себе сервери, пам'ять і мережеві компоненти. Абстрактний рівень - це програмне забезпечення, розгорнуте на фізичному рівні і наділене основними хмарними характеристиками. Абстрактний рівень розташовується над фізичним; (в) поняття хмарного сервісу (Cloud Service) є більш широким порівняно з хмарними обчисленнями (Cloud Computing), оскільки включає технології зберігання й іншого сервісу, термін "хмарний сервіс" передбачає використання спеціальної клієнт-серверної технології використання користувачем ресурсів (процесорний час, оперативна пам'ять, дисковий простір, програмне забезпечення та ін.); (г) віртуалізація - надання набору обчислювальних ресурсів або їх логічного об'єднання, абстрагованого від апаратної реалізації, і такого, що забезпечує при цьому логічну ізоляцію обчислювальних процесів один від одного, які виконуються на одному фізичному ресурсі.

3. Таким чином, щоб систему назвати хмарною, вона повинна задовольняти кількома параметрам: автоматично змінювати обсяг сервісу за

запитом користувача, доступ до сервісів повинен бути організований через стандартні протоколи Інтернет, сервіси повинні бути організовані таким чином, щоб одне і теж саме обладнання могло бути основою для надання сервісів різним користувачам, а вартість такого сервісу повинна бути набагато нижче, ніж вартість схожого сервісу на виділеному обладнанні (з використанням традиційних технологій).

4. Архітектура хмарних систем складається з трьох основних рівнів, яким відповідають три види хмарних сервісів: інфраструктура як сервіс (Infrastructure-as-a-Service, IaaS); платформа як сервіс (Platform-as-a-Service, PaaS); програмне забезпечення як сервіс (Software-as-a-Service, SaaS).

5. У класичному варіанті хмарної обчислювальної системи можна виділити шість основних компонент: (а) по-перше, фізичні машини і їх операційні системи. Ми виділяємо їх, тому що: а) якщо процесори, використовувані в машинах, не підтримують технологію апаратної віртуалізації, то ми можемо говорити лише про паравіртуалізацію, і це істотно знижує як обчислювальні здатності системи, так і вибір програмного забезпечення в подальшому; б) системи з відкритим кодом більшою мірою, ніж інші, повинні бути достатньо гнучкими для роботи з багатьма гостьовими ОС (а комерційні можуть бути вузько спеціалізовані на роботу з певним набором програмних компонент); (б) мережева компонента (використання мережі тягне за собою DNS, DHCP і організацію підмереж фізичних машин); (в) по-третє, використання гіпервізора. гіпервізор дозволяє працювати віртуальним машинам; (г) образи віртуальних машин, що зберігаються на віртуальному «жорсткому диску» - репозиторії, що охоплює всі образи всіх машин, запущених колись в хмарі; (д) необхідний інтерфейс для користувачів; (е) хмарне середовище — як система, що зв'язує воєдино всі шість компонент, розподіляє ресурси користувачам, обслуговуюча мережа та ін.

6. Крім віртуалізації за допомогою гіпервізора існує так звана контейнерна віртуалізація. Відмінність між цими підходами полягає в наступному: гіпервізор емулює апаратне забезпечення, поверх якого запускаються гостьові ОС. При

цьому все, що «вміє» робити обладнання, повинно бути доступно гостьовій ОС з боку базової ОС (тобто машини-«господаря»). Навпаки, контейнери - це віртуалізація на рівні операційної системи, а не на рівні обладнання: кожна гостьова ОС використовує те ж саме ядро (а в ряді випадків - і інші частини ОС), що і базова. В результаті контейнери менше і компактніше гіпервізорних гостьових середовищ, оскільки у них з «базою» набагато більше спільного.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ АПАРАТНОЇ ВІРТУАЛІЗАЦІЇ

2.1 Особливості застосування апаратної віртуалізації

Віртуалізація - сучасна технологія, що розвивається, яка дозволяє нам розділити програмне забезпечення і нижчерозташовану апаратну частину. Одним з цікавих і популярних напрямків у даній технології є віртуальні машини, які являють собою ізольовані програмні контейнери, здатні мати власну операційну систему і додатки, як реальний фізичний комп'ютер. Віртуальна машина в принципі працює так само, як фізичний комп'ютер, і містить власні центральний процесор, оперативний пристрій, жорсткий диск і мережеву інтерфейсну карту, але усі вони є програмними (віртуальними) [8].

Віртуалізація в даний час продовжує розвиватися. У загальному уявленні, віртуалізація відокремлює програмне забезпечення від апаратної інфраструктури та практично розриває зв'язок між певним набором програм і конкретним комп'ютером.

Завдяки віртуалізації у нас з'явилася можливість на одному комп'ютері виконувати роботу декількох «машин», завдяки розподілу його ресурсів за кількома середовищами. Використовуючи віртуальний сервер можна розмістити кілька операційних систем і кілька додатків в єдиному місці, у тому числі віддалено. У цьому випадку, у нас пропадають фізичні та географічні обмеження. Крім енергозбереження та скорочення матеріальних витрат, завдяки більш ефективному використанню апаратних ресурсів, віртуальна інфраструктура підвищує рівень доступності ресурсів, збільшує ефективність системи управління сервером, підвищує безпеку і удосконалює систему відновлення в критичних ситуаціях. В даний час не існує якоїсь єдиної класифікації віртуальних систем, у різних джерелах можна зустріти різні підходи до класифікації.

Два основних напрямки, які можна виділити - це апаратна і програмна візуалізація [15]. Апаратна віртуалізація (апаратна підтримка віртуалізації) — це єдину назва розробок AMD і Intel спрямованих на поліпшення продуктивності

процесора для завдань віртуалізації. У режимі підтримки віртуалізації запускається спеціальне програмне забезпечення, що є своєрідним прошарком між гостьовими операційними системами і обладнанням - монітор віртуальних машин або по-іншому гіпервізор.

Поняття "монітор віртуальних машин" (МВМ) з'явилося у кінці 60-х років як програмний рівень абстракції, який розділяв апаратну платформу на кілька віртуальних машин. Кожна з цих віртуальних машин (ВМ) була настільки схожа на базову фізичну машину, що існуюче програмне забезпечення могло виконуватися на ній в незмінному вигляді. Це був час, коли обчислювальні завдання виконувались на дорогих мейнфреймах (типу IBM/360), і тому користувачі не могли не оцінити здатність МВМ розподіляти дефіцитні тоді ресурси серед кількох додатків.

Отже, монітор віртуальних машин відокремлюючи програмне забезпечення від обладнання, формує проміжний рівень між програмним забезпеченням, яке виконується віртуальними машинами, і апаратними засобами. Цей рівень дозволяє монітору віртуальних машин повністю контролювати використання апаратних ресурсів гостьовими операційними системами, які можуть виконуватися на віртуальних машинах. Інтерфейс МВМ частково або повністю повторює інтерфейс віртуалізованої системи, і тому, програмне забезпечення не може визначити наявність гіпервізора. В даний час монітор віртуальних машин знову в центрі уваги. Багато корпорацій, в числі яких Intel, AMD, Sun Microsystems і IBM створюють стратегії віртуалізації, а в наукових лабораторіях і університетах для вирішення проблем мобільності, забезпечення безпеки і керованості розвиваються підходи, засновані на віртуальних машинах.

Незважаючи на певний попередній період забуття, в 90-і роки дослідники зі Стенфордського університету стали розглядати можливість застосування ВМ для подолання обмежень обладнання та операційних систем. Ця проблема виникла у комп'ютерів з масовою паралельною обробкою (Massively Parallel Processing, MPP), які погано піддавалися програмування і тому не могли виконувати наявні

операційні системи. І ось тоді дослідники виявили, що використовуючи віртуальні машини можна зробити цю незручну архітектуру схожою на існуючі платформи, і в подальшому використовувати переваги готових операційних систем. Надалі з цього проекту вийшли люди і ідеї, які стали золотим фондом компанії VMware, першого постачальника МВМ для комп'ютерів масового застосування.

Як приклад можна привести технології Intel VT-X, VT-D і AMDV, використовуючи їх, монітор віртуальних машин здатний передавати управління фізичними пристроями безпосередньо віртуальній машині.

Другий напрямок - це програмна віртуалізація саме вона є тим, що зазвичай мають на увазі, коли говорять про віртуалізацію, і при цьому представляють типову віртуальну машину [12]. Але навіть цей вид віртуалізації ми можемо розділити на наступні рівні: серверна віртуалізація; паравіртуалізація; віртуалізація рівня операційної системи; віртуалізація ресурсів; віртуалізація прикладних програм.

Розглянемо поняття віртуальної машини, це програмна або апаратне середовище, що виконує деякий код (наприклад, байт-код, шитий код, р-код або машинний код реального процесора), або специфікація такої системи (наприклад: «віртуальна машина мови програмування Сі») [12].

Для порівняння наведемо кілька інших визначень, а саме: віртуальна машина - це повністю ізольований програмний контейнер, здатний виконувати власну операційну систему і додатки, як фізичний комп'ютер. Віртуальна машина працює абсолютно так само, як фізичний комп'ютер, і містить власні віртуальні (тобто програмні) ЦП, ОЗУ, жорсткий диск і мережеву інтерфейсну карту (NIC).

Або по іншому, віртуальна машина - це програма, яку ви запускаєте зі своєї операційної системи. Програма емулює реальну машину. На віртуальні машини, як і на реальні, можна ставити операційні системи. У неї є BIOS, відведене місце на вашому жорсткому диску, мережеві адаптери для з'єднання з реальною машиною, мережевими ресурсами або іншими віртуальними машинами.

Чим же важливі для нас віртуальні машини? По-перше, підвищення ізоляції, тобто обмеження однієї чи групи тісно пов'язаних служб власною віртуальною машиною, крім того зниження ймовірності збоїв від взаємного впливу програм. По-друге, безпека, наприклад, можна розподілити завдання адміністрування і обмежити права кожного адміністратора тільки найнеобхіднішими, так само знижується ймовірність потенційно шкідливих наслідків злому будь-якої зі служб. По-третє, розподіл ресурсів - кожна машина отримує стільки ресурсів, скільки їй необхідно, але не більше того (пріоритезація завдань, виділення пам'яті на вимогу, гнучкий розподіл мережевого трафіку між машинами, розподіл дискових ресурсів).

В даний час можна зустріти достатньо багато розробок віртуальних машин. Розглянемо кілька прикладів і оцінимо можливості їх застосування. Програмний пакет віртуалізації для компанії Microsoft, розроблений, перш за все, для операційної системи Windows, Virtual PC, а також до деякого часу програма емуляції для Mac OS. Дана програма була створена у 1997 році компанією Connectix спочатку для операційної системи Mac OS на платформі PowerPC Macintosh. А через 5 років у 2001 році була випущена версія 4.0 для Windows. Connectix поставляла Virtual PC з різними гостьовими операційними системами, в тому числі Linux і OS/2. На початку 2003 року права на Virtual PC і Virtual Server були викуплені Microsoft. А влітку 2006 року Microsoft випустила безкоштовну версію даного продукту під Windows. У серпні 2006 року компанія оголосила, що версія для Mac OS не буде перенесена на нові Macintosh з процесорами Intel і, таким чином, розвиток цієї гілки продукту було припинено.

Microsoft Hyper-V (кодове ім'я Viridian), це система апаратної віртуалізації для x64-систем на основі гипервизора. Бета-версія Hyper-V була включена в x64-версії Windows Server 200S, а фінальна версія (автоматично, через Windows Update) була випущена 2б червня 200S. Раніше була відома як віртуалізація Windows Server (Windows Server Virtualization).

Hyper-V існує в двох варіантах. Перший варіант - це окремий продукт Microsoft Hyper-V Server. Існує три версії: Hyper-V Server 2012 R2 (поточна версія Hyper-V), Hyper-V Server 2012 Hyper-V Server 200S R2 і Hyper-V Server 2008. Перша версія була випущена в жовтні 200S року. Є базовим варіантом Windows Server 2008, тобто включає в себе повну функціональність Hyper-V, інші ролі Windows 200S Server відключені, також лімітовані служби Windows. Безкоштовна 64-бітна Core-версія Hyper-V обмежена інтерфейсом командного рядка, де конфігурація поточної ОС, фізичного апаратного і програмного обладнання виконується за допомогою команд оболонки. Нове меню інтерфейсу управління дозволяє виконати просту первинну конфігурацію, а деякі вільно поширювані скрипти розширюють дану концепцію.

Адміністрування та конфігурація віртуального сервера (або гостьових ОС) здійснюється за допомогою програмного забезпечення, встановленого на персональний комп'ютер під керуванням Windows Vista, Windows 7 або Windows 2008 Server з встановленим додатком для адміністрування Hyper-V. І другий варіант як роль Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, Windows Server 2008 і x64 бітна версія Windows 8 Pro. У разі Hyper-V ми вже не маємо проблем з підключенням інших операційних систем, причому як на серверних машинах, так і на машинах-клієнтах.

Ще один продукт програмної віртуалізації, що поширюється з відкритим вихідним кодом, Xen - це монітор віртуальних машин з підтримкою паравіртуалізації для процесорів x86 архітектури. Даний гіпервізор може організувати спільне безпечне виконання кількох віртуальних машин на одній фізичній системі з продуктивністю близькою до реальної. В Xen крім дуже високої продуктивності, близької до продуктивності при безпосередньому виконанні на залізі, так само є підтримка до 32 віртуальних процесорів на одну гостьову машину, причому з можливістю «гарячого» додавання процесорів, так само підтримка різних платформ, підтримка апаратної віртуалізації для запуску не

змінених операційних систем і відмінна підтримка обладнання. Це все робить Xen програмним продуктом корпоративного рівня.

Емулятор для PC, що створює DOS-оточення, іноді необхідне для запуску старих програм та ігор під MS-DOS, носить назву DOSBox. Його можна використовувати для запуску й іншого програмного забезпечення для DOS, але така можливість працює з обмеженнями. Емулятор має відкритий вихідний код і доступний для таких систем, як Linux, FreeBSD, Windows, OS X, iOS, OS / 2, BeOS, KolibriOS, Symbian OS, QNX, Android.

2.2 Аналіз віртуальних машин

Концепція віртуальної машини як сукупності ресурсів, які емулюють поведінку реальної машини, з'явилася в Кембриджі наприкінці 1960-х рр. З того часу її очевидні переваги: підвищення ізоляції, безпеку, розподіл ресурсів, постійна доступність, підвищення якості адміністрування та ін. [23] отримували подальший розвиток. Різкий поштовх бурхливому розвитку дала апаратна підтримка віртуалізації на найпопулярнішій платформі x86, коли в середині 2000-х компанії Intel і AMD анонсували технології VT-x і AMD-V.

До цієї події першою спробою корпорації Intel впровадити в свої процесори технології апаратної віртуалізації був режим віртуального процесора 8086 в процесорі 80386, який з'явився у 1985 р. Можливість апаратної віртуалізації додала до вищезазначених переваг ще цілий ряд:

- спрощення розробки програмних платформ віртуалізації за рахунок надання апаратних інтерфейсів управління і підтримки віртуальних гостьових систем;

- можливість збільшувати швидкодію платформ віртуалізації;

- поліпшення захищеності, можливість перемикання між декількома запущеними незалежними платформами віртуалізації на апаратному рівні;

- запуск 64-бітних гостьових систем на 32-бітних хостових системах.

На жаль, варто також зазначити, що апаратна віртуалізація потенційно несе в собі не тільки позитивні моменти. Можливість управляти гостьовими системами за допомогою гіпервізора і простота написання платформи віртуалізації з використанням апаратних технік дозволяють розробляти шкідливе програмне забезпечення (ПО), яке після отримання контролю над хостовою операційною системою (ОС) віртуалізує її та здійснює всі дії за її межами [25].

Програма, яка забезпечує або дозволяє одночасне, паралельне виконання декількох або навіть багатьох ОС на одному і тому ж хост-комп'ютері, називається гіпервізор, або монітором віртуальних машин. Загальноприйнята класифікація гіпервізора (рис. 2.1):

Рисунок 2.1 — Типи віртуалізації

- автономний гіпервізор (тип 1). Має свої вбудовані драйвери пристроїв, моделі драйверів і планувальник і тому не залежить від базової ОС.

Автономний гіпервізор працює безпосередньо на обладнанні і є більш продуктивним [24]. Приклад - VMware ESX;

- на основі базової ОС (тип 2). Цей компонент працює в одному кільці з ядром основної ОС (кільце 0). Гостьовий код може виконуватися прямо на фізичному процесорі, але доступ до пристроїв введення-виведення комп'ютера з гостьової ОС здійснюється через другий компонент, звичайний процес основної ОС - монітор рівня користувача. Приклади: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox;

- гібридний (тип 1 +). Гібридний гіпервізор складається з двох частин: з тонкого гіпервізора, контролюючого процесора і пам'яті, а також працює під його

керуванням спеціальної сервісної ОС в кільці зниженого рівня [35]. Через сервісну ОС гостьові ОС отримують доступ до фізичного обладнання. Приклади: Xen, Citrix XenServer, Microsoft Hyper-V.

В даний час існує, крім згаданих, величезна кількість усіляких гіпервізорів, в тому числі сертифікованих для промислових додатків, наприклад, з особливими вимогами до надійності, безпеки та ін. Такі розробники, як Wind River Systems, Green Hills Software, SYSGO AG та ін. - справжні лідери ринку, і встигнути за ними дуже складно.

2.3 Особливості використання гіпервізорів

Архітектура гіпервізора побудована за компонентним принципом (рис. 2.2). Всі компоненти мають чітко окреслені інтерфейси. Такий підхід є дуже важливим, щоб спростити модифікації при переході на іншу платформу. Кожна версія гіпервізора працює тільки на певній фіксованій апаратній конфігурації. Це необхідно для того, щоб ПО містило тільки необхідний в даний момент функціонал. Чим простіше - тим надійніше, простіше верифікація та сертифікація.

Компоненти поділяються на три шари: 1 - низькорівневі драйвери; 2 - управління платформою і налагоджувальні утиліти; 3 - управління віртуалізацією.

Рівень драйверів підрозділяється на два блоки — ядро і введення/виведення. У перший входить управління сегментованою і сторінковою пам'яттю, таблицею переривань, функціями ядра процесора і Local APIC. Введення/висновок включає наступні пристрої: графічний дисплей, клавіатуру, послідовний і паралельний порти.

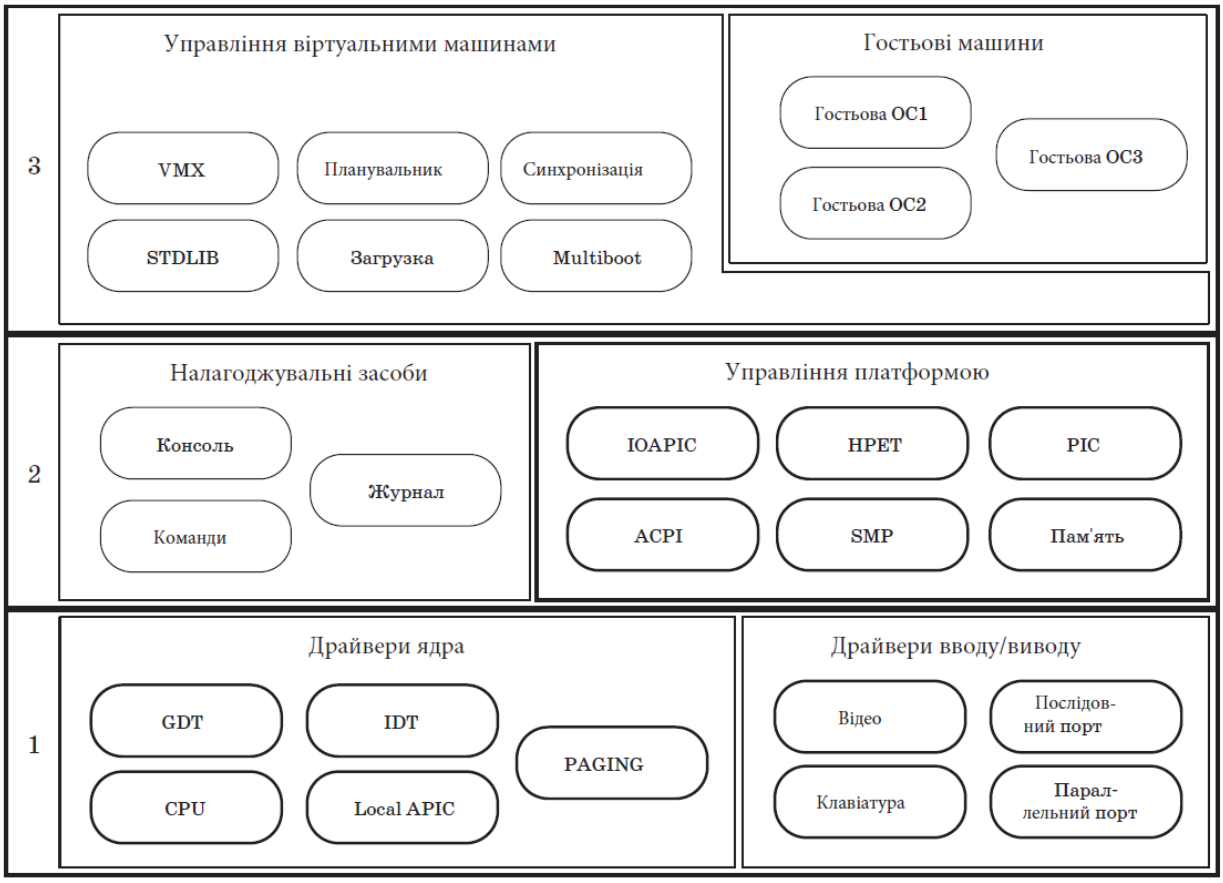

Рисунок 2.2 — Компонентна архітектура гіпервізора

Наступний рівень компонентів - це логічна надбудова над драйверами. HPET (high-precision event timer) використовується для засобів синхронізації і роботи профілювальника. Модуль I/O APIC призначається для конфігурації розподілу переривань по ядрах. Модуль ACPI дозволяє отримувати інформацію про конфігурацію обладнання, реалізовувати програмне скидання і вимикання. Модуль SMP має реалізацію функцій, для запуску, призупинення і скидання ядер. Блок управління налагодженням включає багатовіконну консоль (для кожного ядра), обробник команд і ведення журналу (протоколювання). На самому верхньому рівні розташовується шар компонентів менеджера віртуальних машин і самих гостьових машин.

2.4. Використання апаратної віртуалізації в контексті інформаційної безпеки

Оскільки в даний час інформація, як ресурс, набуває всю більшу цінність, при розгляді будь-якої нової технології або сервісу, необхідно враховувати питання забезпечення інформаційної безпеки. При цьому дане питання має значення, як для великої компанії, так і для окремо взятого користувача.

Забезпечення інформаційної безпеки хмарних послуг - задача, яка стоїть перед провайдером послуги. Проблеми хмарних сервісів, з точки зору інформаційної безпеки, можна розділити на три групи [4]:

1. Проблеми технологічні і фізичні. В першу чергу, необхідно взяти до уваги, в якому центрі обробки даних (далі - ЦОД) знаходиться обладнання. ЦОД повинен бути захищений від припинення надання послуги в результаті перебоїв з електроживленням, катастроф техногенного характеру, фізичного пошкодження обладнання в результаті аварій, необережних дій персоналу, диверсій зловмисників та ін. Прикладами заходів необхідного фізичного захисту можуть бути:

- кілька периметрів безпеки на території ЦОД;
- "модульна" система побудови серверних приміщень;
- розмежування прав доступу персоналу відповідно до виконуваних службових обов'язків;
- установка системи відеоспостереження та пожежогасіння та ін.

Статистики по інцидентах в хмарних сервісах недостатньо і рівень загроз ще не визначений, а існуючі методи захисту інфраструктури незастосовні, так як у хмарному сервісі немає чіткого поняття інфраструктури. Але треба розуміти, що в даний час передбачити атаку від мережевих атак так само необхідно, як і забезпечити фізичний захист периметра. Однією з характерних проблем в галузі мережевої безпеки є DDoS атаки (розподілена атака типу “відмова в обслуговуванні”, виконується з метою вивести систему з ладу шляхом подачі безлічі помилкових запитів). Наслідки такої атаки для хмарного провайдера

можуть бути вкрай серйозними, аж до повного розорення, в результаті масового звернення клієнтів за грошовою компенсацією.

У зв'язку з цим хмарний ЦОД необхідно обов'язково убезпечити від DDoS. Робиться це апаратними засобами, наприклад Juniper Net-Screen або Cisco Guard DDoS, за допомогою яких в трафіку розпізнаються і блокуються помилкові запити типу DDoS. Також таке обладнання повинно бути встановлено і в ЦОД, і безпосередньо у провайдера.

Додатково хмарному провайдеру необхідно передбачити брандмауери як програмні, так і апаратні. З додаткових засобів захисту також можна відзначити системи, які здатні виявляти аномалії трафіку, характерні для спроб вторгнення, і заздалегідь відсікати їх.

В цілому, ефективний захист досягається тільки шляхом застосування комплексних заходів [16]. Відповідне обладнання коштує дуже дорого, тому для малих компаній має сенс користуватися хмарними сервісами, ніж використовувати інфраструктуру, розгорнуту власними силами, тому що великі хмарні провайдери повинні володіти відповідними засобами захисту і можуть забезпечити належний рівень обслуговування.

Найбезпечнішим типом хмарних послуг можна вважати SaaS, маючи на увазі звичайно, що провайдер сервісу забезпечив належний рівень захисту програмного продукту - відсутність вразливостей у коді, забезпечене базовим захистом від злому, підтримує ізоляцію облікових записів та ін. Пояснюється це тим, що сервіси PaaS та IaaS дозволяють користувачам вносити в хмарну інфраструктуру практично будь-яке програмне забезпечення, аж до додатків зі шкідливим кодом, тоді як SaaS дозволяє працювати тільки з існуючими додатками в інфраструктурі [16].

Можливим вирішенням проблеми безпеки сервісів PaaS і IaaS є прийняття загального стандарту, який би обумовлював умови привнесення програмних продуктів в інфраструктуру хмари, а також визначав інструменти, за допомогою яких ці продукти створювалися.

Ще у 2004 році був створений подібний документ Application Packaging Standard (APS), який визначав певні рамки для розробників програмного забезпечення, однак, повсюдного застосування так і не зазнав. Ще одним способом захисту сервісів PaaS і IaaS є ізоляція різних візуальних середовищ, щоб "падіння" сервісу у одного користувача не могло відбитися на працездатності додатків у інших користувачів.

2. Проблеми психологічні. У нас практично відсутня культура аутсорсингу, використання зовнішніх сервіс провайдерів. Крім того, технологія нова, складна і її використання досить ризиковано. При використанні сервісу хмарних обчислень у користувача до провайдера виникає ряд питань, пов'язаних з інформаційною безпекою, на які провайдер ще не готовий повністю відповісти. Наприклад, яким буде розмір збитку в разі порушення роботи сервісу, як розслідувати інциденти безпеки, чи надійно видаляються дані, де будуть зберігатися дані, як виконувати вимоги регуляторів та ін. Провайдеру хмарного сервісу необхідно вирішити кілька питань, які будуть виникати у користувачів, пов'язаних із забезпеченням інформаційної безпеки: як переконати користувача, що його дані в безпеці, до якої міри довіряти користувачеві, як розмежувати доступ між користувачами у хмарі, як захищати хмарні інфраструктури і від кого та ін.

При використанні хмарних сервісів є свої плюси і мінуси з точки зору психології користувача. До плюсів моно віднести: зменшення витрат на захист інформації, перенесення частини ризиків на провайдера. Мінуси: не ясно, що очікувати від нової технології, витік інформації може призвести до великих збитків для компанії клієнта, виникає питання довіри до сервіс провайдерів.

3. Проблеми юридичні. Основна проблема хмарних технологій - це відсутність загальноприйнятих стандартів з безпеки [6].

З точки зору оцінки провайдера в галузі забезпечення їм інформаційної безпеки західними компаніями рекомендовано дотримуватися певної методології. В першу чергу, це аудит за стандартом SAS 70 Type II, плюсом є сертифікація провайдера по ISO 27001 або проходження практик ISO 27002.

Також хмарні сервіси можна проаналізувати і з іншого боку - їх застосування як хмарних сервісів безпеки.

Основне завдання хмарних сервісів безпеки це захист "мобільних" користувачів, які встановлюють з'єднання з Інтернетом віддалено через ноутбуки, планшетні комп'ютери, мобільні телефони та ін. Таке віддалене підключення значно збільшує ризики інформаційної безпеки. Сьогодні найпоширенішою послугою хмарної безпеки є саме безпека Інтернет-контента - Web і електронної пошти. Суть цих послуг проста - весь Web- або e-mail-трафік проходить через хмарну інфраструктуру, яка і перевіряє його на предмет порушення інформаційної безпеки. Треба зауважити, що хмарна безпека потрібна не тільки для мобільних користувачів. Вона корисна і для будь-якого стаціонарного офісу будь-якого розміру, бо знижує витрати. У цьому випадку вигода споживача - приблизно 30-40%, оскільки підприємству не потрібно купувати, розгортати і підтримувати обладнання безпеки, яке встановлюється на території замовника.

Для мобільних же користувачів така послуга в принципі не має альтернатив. В цілому можна помітити, що вигода від переходу до хмарної безпеки має ряд незаперечних переваг для споживача такого сервісу:

- зниження капітальних і операційних витрат;

- абсолютна прогнозованість витрат, щомісячна оплата, фіксована при незмінній кількості користувачів;

- безпека в режимі 24x7. Постачальник послуги забезпечує захист в цілодобовому режимі (звичайний режим в 99% організацій — 8x5);

- прозорість модернізації та її оперативність;

- висока і гарантована надійність;

- гнучкість розгортання, простота підключення нових користувачів і майданчиків замовника.

Для успішної організації інформаційної безпеки хмарної системи необхідно враховувати наступні елементи: підсистему, яка забезпечує надійне зберігання інформації у клієнта; підсистему яка організовує безпеку в мережі;

підсистему забезпечення надійності віртуальних середовищ; підсистему яка дозволяє організувати безпечну роботу в центрах обробки даних. Необхідно відзначити, що, як і в інших подібних системах, призначених для захисту інформації, говорити про успішно організовану безпеку хмарної системи можна тоді, коли роботу всіх підсистем організовано на найвищому рівні [1].

Для детального вивчення питання дослідження безпеки в хмарах проаналізуємо, зокрема, кожну з перерахованих вище підсистем.

Підсистема, яка забезпечує надійне зберігання інформації у клієнта може бути схильна до таких загроз як - Cross-site-scripting (CSS), Phishing, віруси, трояни це можна пояснити тим, що користувачі змушені працювати з сервісом хмарних обчислень використовуючи інтернет-браузер. Для того щоб організувати безпеку інформації на цьому рівні необхідно на стороні клієнта дотримуватися виконання наявності наступних елементів: антивірусні пакети програм для захисту інформації; засоби, які дозволяють шифрувати дані на диску; персональний брандмауер, який буде знаходитися в ОС; інтернет-браузер який правильно налаштований з точки зору безпечного підключення до мережі [18].

Підсистему, яка організовує безпеку в мережі необхідно організовувати таким чином щоб дані які знаходяться в публічній хмарі були в безпеці для цього потрібно використовувати спеціальний тунель віртуальної приватної мережі (VPN), він надає можливість об'єднати клієнта і сервер з метою отримання публічних хмарних послуг. VPN-тунель дозволяє здійснювати безпечні з'єднання і крім того користувач має можливість використовувати єдине ім'я і пароль для входу до різних хмарних ресурсів. VPN- з'єднання використовує ресурси Інтернет, доступ до яких мають всі користувачі, для того щоб організувати процес передачі даних для публічних хмар. Описаний механізм можливо реалізувати на практиці за допомогою режимів доступу з шифруванням з використанням двох ключів на базі протоколу Secure Sockets Layer.

Підсистема забезпечення надійності віртуальних середовищ залежить від безпеки механізмів віртуалізації. Якщо успішно реалізована атака на гіпервізор, то

зловмисник може в таємниці, непомітно для систем захисту інформації, які функціонують у віртуальних машинах, здійснювати наступні дії: копіювати блокувати інформаційний потік даних, який йде на різні пристрої ((HDD, принтер, USB); читати і редагувати інформацію на дисках віртуальних машин, слід зазначити що ці дії будуть можливі і при вимкнених машинах, без участі їх програмного забезпечення. Для того щоб вирішити описану проблему необхідно захистити гіпервізор для цього потрібно розмежувати права доступу до сервера віртуалізації, це можливо зробити шляхом своєчасної установки оновлень програмного забезпечення середовища віртуалізації, а також потрібно обмежити запуск програм [28].

Диск віртуальної машини знаходиться на SAN/NAS, в зв'язку з цим стає актуальним питання про захист інформаційних даних віртуальних машин шляхом обмеження доступу до дисків машин за допомогою сертифікованих засобів захисту інформації та спеціальними екранами, які здатні контролювати протоколи та інші елементи віртуальної інфраструктури. Якщо зловмисник отримає доступ до засобів адміністрування, то він буде мати можливість до редагування даних, знищення, викрадення та ін., у всій мережі віртуальної інфраструктури. Щоб уникнути подібних випадків атак, необхідно захистити доступ до мережі адміністрування, серверів віртуальних машин, всіляких засобів управління інфраструктурою.

Висновки до другого розділу

1. Віртуалізація - сучасна технологія, що розвивається, яка дозволяє нам розділити програмне забезпечення і нижчерозташовану апаратну частину. Одним з цікавих і популярних напрямків у даній технології є віртуальні машини, які являють собою ізольовані програмні контейнери, здатні мати власну операційну систему і додатки, як реальний фізичний комп'ютер. Віртуальна машина в принципі працює так само, як фізичний комп'ютер, і містить власні центральний процесор, оперативний пристрій, жорсткий диск і мережеву інтерфейсну карту, але усі вони є програмними (віртуальними).

2. Поняття "монітор віртуальних машин" (МВМ) з'явилося у кінці 60-х років як програмний рівень абстракції, який розділяв апаратну платформу на кілька віртуальних машин. Кожна з цих віртуальних машин (ВМ) була настільки схожа на базову фізичну машину, що існуюче програмне забезпечення могло виконуватися на ній в незмінному вигляді. Це був час, коли обчислювальні завдання виконувались на дорогих мейнфреймах (типу IBM/360), і тому користувачі не могли не оцінити здатність МВМ розподіляти дефіцитні тоді ресурси серед кількох додатків.

3. Можливість апаратної віртуалізації додала цілий ряд переваг:

- спрощення розробки програмних платформ віртуалізації за рахунок надання апаратних інтерфейсів управління і підтримки віртуальних гостьових систем;

- можливість збільшувати швидкодію платформ віртуалізації;

- поліпшення захищеності, можливість перемикання між декількома запущеними незалежними платформами віртуалізації на апаратному рівні;

- запуск 64-бітних гостьових систем на 32-бітних хостових системах.

4. Загальноприйнята класифікація гіпервізора: - автономний гіпервізор (має свої вбудовані драйвери пристроїв, моделі драйверів і планувальник і тому не

залежить від базової ОС, автономний гіпервізор працює безпосередньо на обладнанні і є більш продуктивним, приклад - VMware ESX; на основі базової ОС, працює в одному кільці з ядром основної ОС, приклади: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox; гібридний, складається з двох частин: з тонкого гіпервізора, контролюючого процесора і пам'яті, а також працює під його керуванням спеціальної сервісної ОС в кільці зниженого рівня, приклади: Xen, Citrix XenServer, Microsoft Hyper-V.

РОЗДІЛ 3 ОСОБЛИВОСТІ КОНТЕЙНЕРНОЇ ВІРТУАЛІЗАЦІЇ ХМАРНИХ СЕРВІСІВ

3.1 Технології контейнерної віртуалізації: особливості застосування

Віртуалізація на рівні операційної системи - метод віртуалізації, при якому ядро операційної системи підтримує кілька ізольованих примірників простору користувача. Ці екземпляри (часто звані контейнерами або зонами) з точки зору користувача повністю ідентичні реальному обчислювальному модулю. Ядро забезпечує повну ізольованість контейнерів, тому програми з різних контейнерів не можуть впливати один на одного.

LXC [22] - система віртуалізації на рівні операційної системи для запуску декількох ізольованих екземплярів ОС Linux на одному комп'ютері. LXC не використовує віртуальні машини, а створює віртуальне оточення з власним простором процесів і мережевим стеком. Всі екземпляри LXC використовують один екземпляр ядра ОС. LXC заснована на технології ядра Linux під назвою «Контрольні групи» (cgroups, додано у версії ядра 2.6.29) с використанням механізму ізоляції «простір імен» (namespaces). Часто технологія контейнерної віртуалізації розглядається як поліпшена реалізація механізму «пісочниці» chroot.

Для автоматизації розгортання і управління додатками в середовищі контейнерної віртуалізації використовується спеціальне програмне забезпечення Docker [22]. Docker дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на будь-яку Linux-систему з підтримкою механізму контрольних груп в ядрі, а також надає середовище з управління контейнерами.

До складу програмного засобу Docker входять сервер контейнерів, клієнтські засоби, що дозволяють за допомогою командного інтерфейсу управляти образами і контейнерами, а також API керування контейнерами. Сервер контейнерів забезпечує повну ізоляцію контейнерів, що запускаються на обчислювальному модулі на рівні файлової системи (у кожного контейнера власна

коренева файлова система), на рівні процесів (процеси мають доступ тільки до власної файлової системи контейнера, а ресурси розділені засобами LXC), на рівні мережі (кожен контейнер має доступ тільки до прив'язаному до нього мережевого простору імен і відповідним віртуальним мережевим інтерфейсам).

Набір клієнтських засобів дозволяє запускати процеси в нових контейнерах, зупиняти і запускати контейнери, припиняти і відновлювати процеси в контейнерах. Окремий набір команд дозволяє здійснювати моніторинг запущених процесів.

Нові образи можна створювати зі спеціального сценарного файлу, передбачена можливість записати всі зроблені в контейнері зміни в новий образ. Всі команди можуть працювати як з docker-сервером локальної системи, так і з будь-яким доступним по мережі сервером docker.

За допомогою ПО Docker Registry є можливість створення сховища для зберігання готових переднастроєних образів контейнерів. Таким чином, користувач може використовувати контейнер з довіреного сховища з необхідним для виконання завдання набором бібліотек і ліцензій.

Docker - це вільно-розповсюджувана платформа для розгортання та експлуатації додатків. Метою Docker є більш швидкий та простий запуск додатків, а також легке перенесення з одного хоста в інший. Він дозволяє швидше тестувати і швидше розробляти додаток. Docker - це контейнерна віртуалізація, тобто легка платформа віртуалізації, яка вміє запускати різні програми, написані на різних мовах програмування і на різних технологіях.

Docker використовує архітектуру клієнт (docker клієнт) - сервер (docker демон). Клієнт спілкується з так званим демоном Docker, який бере на себе завдання створення, розподілу і запуску контейнерів. Обидва, клієнт і сервер, можуть працювати в одній системі, також сервер може бути віддаленою.

Docker складається з трьох компонентів:

• Образи - шаблони тільки для читання. Наприклад, образ може містити в собі операційну систему Ubuntu з Apache і додатком. Вони призначені для створення контейнерів.

• Реєстр - сховище образів. Розробник може створити свій образ або завантажити вже створений образ з реєстру Docker Hub.

• Контейнери - створюються з образів. Можна зупиняти, запускати і видаляти контейнери. У контейнерах міститься все, що потрібно для роботи програми.

Образи складаються з набору рівнів, а Docker використовує Union File System для перетворення цих рівнів в один образ. Union file system дозволяє файлам і директоріями з різних систем прозоро накладатися, створюючи когерентну файлову систему. Завдяки цим рівням Docker легковагий і має високий рівень швидкості.

Як створюються образи? Образи можна створити двома способами - вручну і за допомогою Dockerfile. Dockerfile - це файл, який містить в собі кілька рядків команд.

```
FROM                                                    node:argon
RUN                         mkdir           -p          /usr/src/app
WORKDIR                                                 /usr/src/app
COPY                                    package.json    /usr/src/app/
RUN                                     npm             install
COPY                                                    /usr/src/app
EXPOSE                                                  8080
CMD                                                     [ "npm", "start" ]
```

Цей файл створює образ з операційною системою Ubuntu node Argon і налаштовує програми для запуску в порте 8080. Потім програмісту потрібно просто запустити цей образ, створений Dockerfile і у нього вже буде запущений сервер, що працює.

Docker був розроблений на мові Go і використовує деякі можливості ядра Linux. Також він використовує технологію namespaces для організації ізольованих робочих просторів, які називаються контейнерами [9].

(1) Контейнерна віртуалізація може бути використана при організації високопродуктивних обчислень як засіб розгортання спеціалізованих програмних платформ, в тому числі створених користувачем. Накладні витрати на контейнерну віртуалізацію при розгортанні програмної платформи істотно менше, ніж на віртуалізацію за допомогою гіпервізора.

(2) Накладні витрати на контейнерну віртуалізацію істотно зростають, якщо мережа інформаційних обмінів використовує стек протоколів TCP / IP.

(3) На сьогоднішньому етапі розвитку контейнери не здатні виключити взаємний вплив завдань, які поділяють один і той же обчислювальний модуль, застосування контейнерів не вирішує проблеми фрагментації обчислювальних ресурсів при колективному доступі.

3.2 Порівняння контейнерної та гіпервізорної віртуалізації

При побудові системи для віддаленої роботи з віртуальними машинами виникає питання про вибір головного принципу віртуалізації. Зазвичай програмне забезпечення Oracle VirtualBox, яке було вибрано в якості основи для побудови названої системи, використовують на хост-системі, і виходить гіпервізорная віртуалізація, де в ролі гіпервізора виступає хост-система з Ubuntu Server 16.04.

Переваги і недоліки цього підходу в цілому відомі. Однак проблеми ізоляції віртуальних машин і мереж одного користувача від віртуальних машин і мереж іншого користувача легко вирішуються при використанні контейнерів і платформи Docker. Docker - відкрита платформа для запуску додатків в вільно ізольованому середовищі (в контейнері) [10]. Тому виникла ідея провести дослідження, які відповіли б на питання: наскільки можна заощадити оперативної пам'яті (ОП) в хост-системі в разі використання контейнерної віртуалізації в порівнянні з гіпервізорною і наскільки при цьому можна втратити швидкодію.

Проведені експерименти несподівано показали, що при використанні контейнерів і програмного забезпечення Docker при організації кластерних обчислень ми не тільки легко вирішуємо проблему ізоляції віртуальних машин і мереж одного віддаленого користувача від іншого, але і отримуємо навіть виграш в швидкості обчислень і витратах ОП.

Нагадаємо, що контейнерна віртуалізація - це такий метод віртуалізації, при якому ядро операційної системи підтримує замість одного відразу кілька ізольованих примірників простору користувача [10]. Швидше за все, ця ідея була запозичена від реєнтерабельних програм, які дозволяють мати один екземпляр коду, але кілька процесів можуть його виконувати паралельно, кожен - зі своїми наборами змінних, що знижує витрати ОП. Як відомо, звернення до змінних реєнтерабельних програм здійснюється за допомогою непрямої адресації, причому поточні змінні, як правило, розташовуються на вершині стека, з якого під новий процес виділяється новий блок осередків пам'яті. Запит (і звільнення) такого блоку пам'яті здійснюється за допомогою звернення до системної підпрограми, яка виконує витребувані дії при відключеній системі переривань для того, щоб забезпечити цілісність цієї операції.

Таким чином, контейнерна віртуалізація є більш ефективною в плані витрачання пам'яті, але втрачає в швидкодії через додаткові витрати на обчислення адрес. Сам гіпервізор працює наступним чином: в операційній системі хоста емулюється апаратне забезпечення, поверх якого запускаються гостьові операційні системи. Це дає нам зрозуміти, що зв'язок між гостьовою і хостовою операційними системами дотримується чіткого правила: все, що робить використовувана апаратна частина, має бути доступне і гостьовій операційній системі з боку хостової [8].

Контейнери, в свою чергу, навпаки, є віртуалізацією на рівні операційної системи. А це вже має на увазі, що є гостьова операційна система, яка використовує те ж саме ядро, що і хостова. Такий підхід дає контейнерам велику перевагу: вони можуть бути менше і компактніше за гіпервізорні гостьові

середовища, оскільки у них з хостом набагато більше спільного, ніж у віртуальних машин.

Тобто контейнерна віртуалізація не використовує повноцінні віртуальні машини, як це робиться в разі гіпервізора, а створює віртуальне оточення з власним простором процесів і мережевим стеком. Екземпляри просторів користувача (часто звані контейнерами або зонами) з точки зору користувача повністю ідентичні реальному серверу, але вони в своїй роботі використовують один екземпляр ядра операційної системи. Ядро забезпечує повну ізольованість контейнерів, тому програми з різних контейнерів не можуть впливати одна на одну. Очевидно, що головним недоліком цієї технології є те, що віртуальне середовище задається хост-системою, і вона буде одна і та ж для всіх просторів, які ми можемо створити. Для подолання даного недоліку було запропоновано модернізувати цю технологію до поняття докера (Docker). Докери можуть бути з різним віртуальним оточенням, хоч і працювати на одній і тій же хост-системі [27].

Для можливості запускати на GNU/Linux-системі віртуальні машини з ОС Microsoft Windows можна створити відповідні Linux-контейнери і за допомогою програмного забезпечення Docker встановити пакет програм VirtualBox. Контейнери ізолюють віртуальні машини VirtualBox, що запускаються різними користувачами. Важливо, що всередині одного контейнера можуть працювати відразу кілька віртуальних машин одного користувача, а кожному користувачеві надається свій контейнер.

Для побудови кластерної системи це дуже важливо, тому що віртуальні машини одного користувача зможуть легко і швидко взаємодіяти один з одним через віртуальні мережі. При цьому віртуальні машини і мережі різних користувачів можуть бути запущені на різних вузлах кластера. Оскільки це кластер серверів, то в якості загального зовнішнього пристрою виступає мережева система зберігання даних NetGear 3200, яка надає за i-SCSI-протоколом єдиний том X-RAID-6-рівня.

На цьому томі розташовуються файли віртуальних машин, завдяки чому вони можуть бути доступні для будь-якого вузла кластера. Користувач працює зі своїми віртуальними машинами через той сервер, який буде мати найменше навантаження в кластері.

Відповідно, для порівняльного аналізу розглянутих методів віртуалізації за допомогою тестування з метою вибору найбільш ефективного методу потрібно було розробити методику отримання потрібних нам параметрів - споживаних обчислювальних ресурсів.

Було запропоновано в якості навантаження на сервер, на якому працюватимуть віртуальні машини, взяти процедуру інсталяції операційної системи. Ця процедура задіє відразу всі види ресурсів і супроводжується високим процессорним навантаженням, великим споживанням ОП, інтенсивною роботою з зовнішніми пристроями і, що дуже важливо, активною роботою з накопичувачем на магнітних дисках (як правило, це віртуальні накопичувачі, в якості яких використовуються файли хост-системи).

Частина процедури інсталяції, яка зазвичай вимагає інтерактивної взаємодії з користувачем (системний адміністратор під час інсталяції відповідає на деякі питання), була автоматизована. Автоматизація відповідей була виконана створенням нового файлу відповідей, в якому заздалегідь підготовлені всі відповіді на всі питання. Це дозволило проводити експерименти без впливу інтерактивної складової, яка могла б вплинути на результати експерименту. Для зняття показань був написаний спеціальний скрипт на мові Bash, який з моменту його запуску записує в окремий файл час в секундах, навантаження CPU (Central Processing Unit) в процентах і витрачання оперативної пам'яті в процентах.

Наступним кроком підготовки до проведення дослідження був вибір алгоритму, за яким безпосередньо проводилися б експерименти. В ході підготовки був розроблений наступний метод: так як в нашому кластері є два ідентичних сервера на базі платформи Supermicro E210C-M3, то було прийнято рішення на одному з них проводити експеримент зі звичайними віртуальними машинами, а на

другому проводити експерименти з докерами (контейнерами). А так як ми будуємо кластер серверів, то обидва цих обчислювальних сервера мають доступ до загального сховища. Далі необхідно було підготувати сервери, встановивши VirtualBox для сервера, на якому використовується гіпервізорова віртуалізація, а для сервера з контейнерною - програмне забезпечення Docker. Це відкрита платформа для розробки, доставки і експлуатації додатків [27]. У своєму ядрі це програмне забезпечення дозволяє запускати практично будь-який додаток, безпечно ізольоване всередині контейнера, що дає можливість убезпечити роботу хост-системи. Тобто події, що відбуваються в контейнері, ніяк не впливають. Завдяки безпечній ізоляції можна запускати на одному хості багато контейнерів одночасно. За рахунок легковагої природи контейнера, який запускається без додаткового навантаження гіпервізора на апаратну частину сервера, використовується менше ресурсів. Також в зв'язку з можливістю використовувати різні методи зберігання даних необхідно провести порівняльний аналіз швидкодії мережевого сховища і звичайного (локального) накопичувача на жорстких магнітних дисках, встановленого безпосередньо в сервер сховища даних. Потрібно проаналізувати, як ці кошти і технології зберігання даних вплинуть на швидкодію і продуктивність віртуальних машин і контейнерів, і вибрати найкращий з наявних варіантів.

Було визначено необхідний мінімум віртуальних машин і контейнерів, який відповідав би одночасній роботі, наприклад, п'яти користувачів. Так, на першому сервері потрібно створити 10 віртуальних машин - по дві машини на користувача. Після створення, тобто визначення конфігурації віртуального комп'ютера, а це процес досить швидкий і не вимагає значних ресурсів, запускається скрипт документування споживаних обчислювальних ресурсів. Далі запускаються власне віртуальні машини, і вже безпосередньо в них відразу починається автоматизована установка операційних систем. Вищезгаданий скрипт веде протокол споживання обчислювальних ресурсів цими віртуальними машинами.

Для імітації навантаження на другий сервер були створені п'ять контейнерів зі спеціальним набором програмного забезпечення, необхідного для організації роботи з графічними додатками в контейнерах. Потім в кожному контейнері було створено по дві віртуальні машини, і вже безпосередньо після закінчення процедури їх створення запускався скрипт документування навантаження, а слідом запускалися все віртуальні машини і йшла автоматизована установка операційних систем.

У обробку результатів входив ряд послідовних етапів, які включають в себе приведення даних до табличного вигляду і будівництво представлених нижче графіків.

Розглянемо графіки, на яких показаний тестовий експеримент з однією віртуальною машиною і одним контейнером з однією віртуальною машиною (рис. 3.1, а і б).

Наочно видно, що звичайна віртуальна машина працює швидше, ніж аналогічна, але в контейнері, споживаючи при цьому більше обчислювальних ресурсів.

Як і очікувалося, контейнер показав менше споживання ресурсів, але більш тривалу роботу.

Якщо подивитися на форму і поведінку графіків, то вони ідентичні: основним піком навантаження на ОП є період безпосередньої установки операційної системи, це друга частина графіка.

У порівнянні з першою частиною графіка, який відображає фазу розпакування файлів з образу, навантаження зросло приблизно в 2 рази. Це говорить про те, що власне побудова (збірка) системи з двоїчних файлів вимагає істотно більше ресурсів, ніж підготовка цих двоїчних файлів.

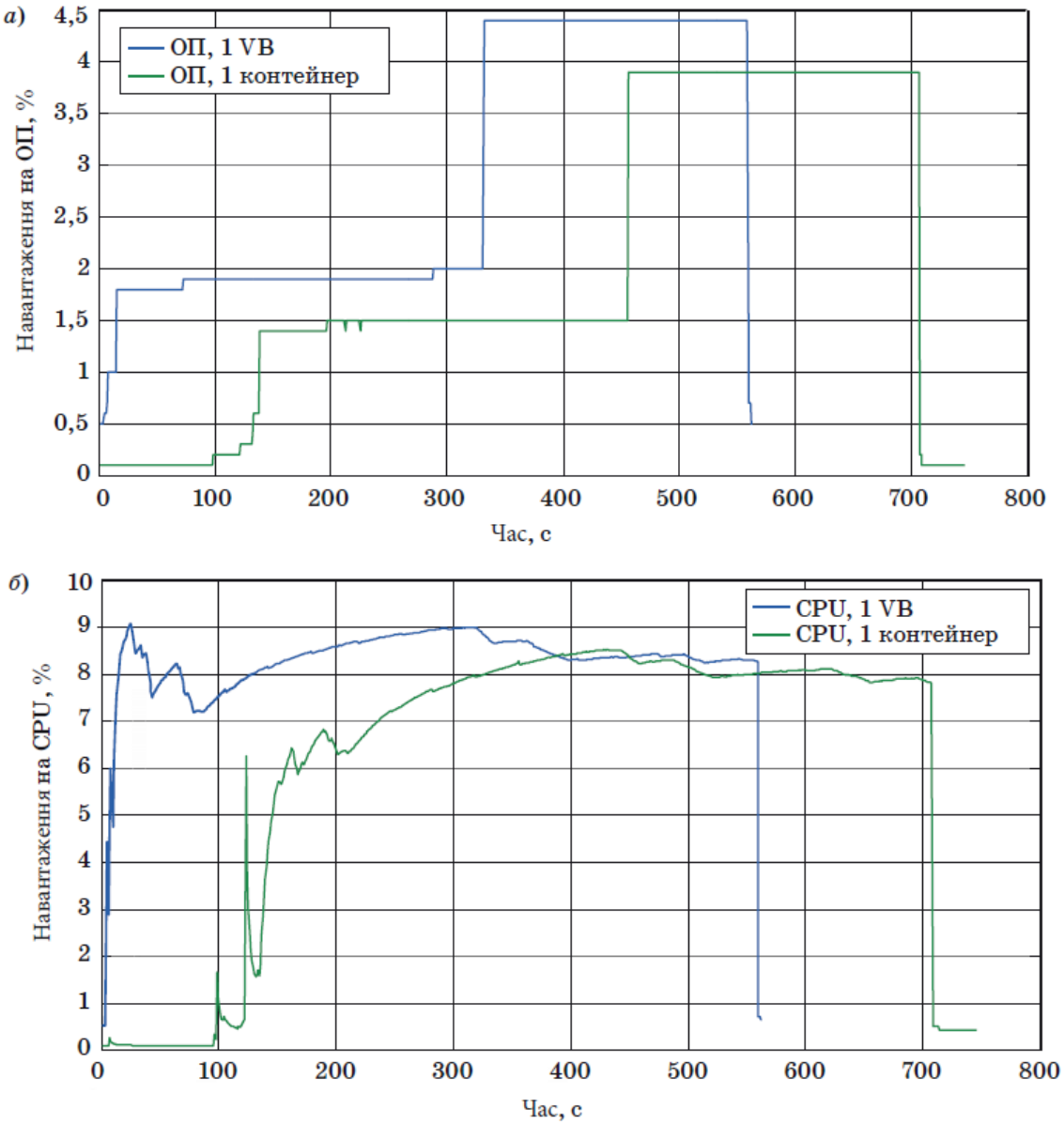

Рисунок 3.1 — Навантаження на оперативну пам'ять (а) і CPU (б) в експерименті з одним контейнером і однією віртуальною машиною

Подивимося на графіки тестування віртуальних машин, розташовані на рис. 3.2, а і б.

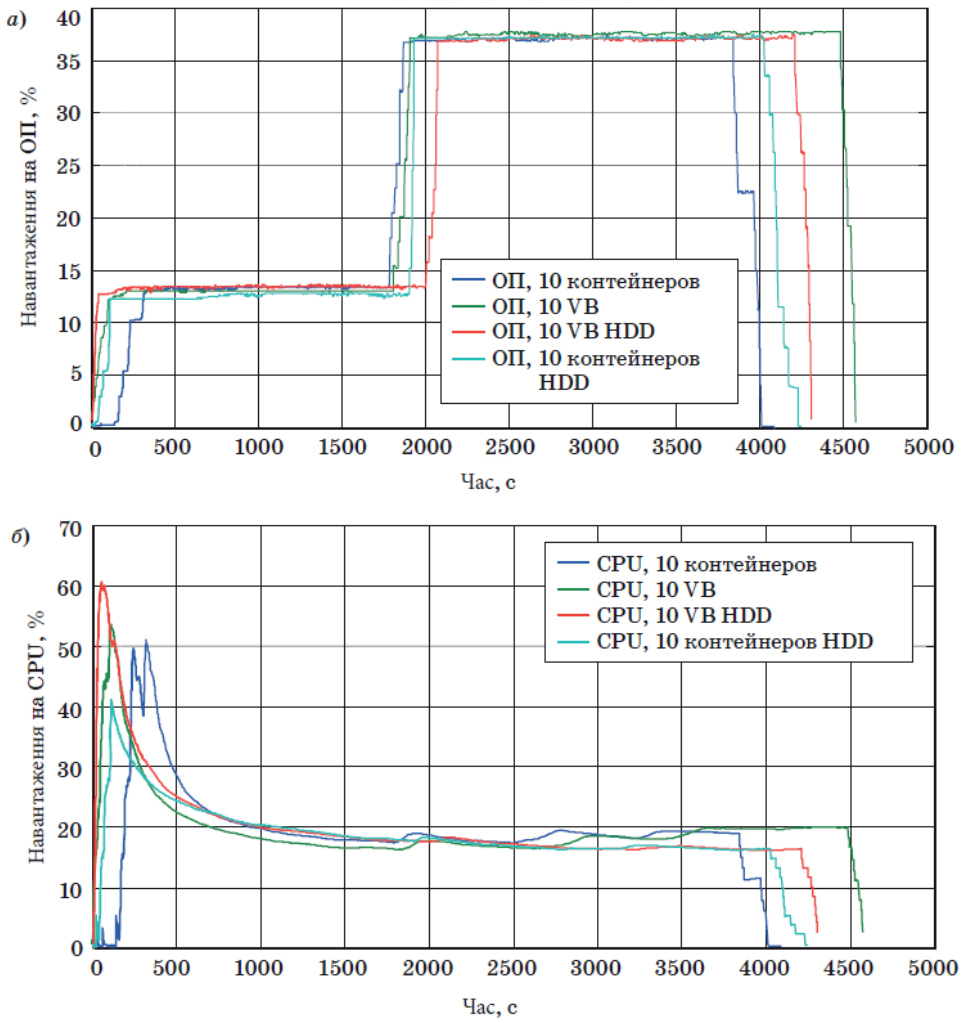

Рисунок 3.2 — Навантаження на оперативну пам'ять (а) і CPU (б) в експерименті з десятьма контейнерами і десятьма віртуальних машинами

За навантаженням на CPU віртуальна машина більш вимоглива, їй необхідно більше обчислювальної потужності. Звичайно, в масштабах сервера це не настільки важливо, якщо ми говоримо про одну віртуальну машину.

Тут ми запускаємо вже по 10 віртуальних машин на одному сервері. Звернемо увагу на те, що тут представлені результати двох експериментів: криві зеленого і синього кольорів ілюструють поведінку серверів з використанням зовнішнього мережевого сховища, а червоного і блакитного - роботу серверів з використанням локального накопичувача. Відмінності в тому, що були використані різні технології доступу до накопичувачів даних. На рис. 3.2, представлена залежність навантаження на ОП від часу. Експеримент номер один був проведений

з використанням мережевої системи зберігання даних NetGear 3200, яка надає єдиний том X-RAID-6-рівня за i-SCSI-протоколом. З графіка видно, що контейнерне рішення відпрацьовує швидше на 600-800 с. Тобто при конфігурації з використанням протоколу i-SCSI контейнерна віртуалізація виявляється швидше. Якщо подивитися на графік рис. 3.2, б, то можна виявити, що контейнерна віртуалізація із застосуванням Docker не тільки швидше, але вона ще й менше навантажує процесор. Тобто за графіком рис. 3.2, рівень споживання ОП однаковий, а споживання ресурсів процесора менше.

Отже, за результатом дослідження було визначено коло параметрів, від яких залежить продуктивність двох технологій віртуалізації - гіпервізорної та контейнерної. Як показав перший (тестовий) експеримент, якщо проводити порівняння однієї віртуальної машини з одним контейнером, то віртуальна машина працює набагато швидше (рис. 3.1), споживаючи більше обчислювальних ресурсів сервера. Але при збільшенні кількості віртуальних машин до 10 обстановка кардинально змінюється у зворотний бік, і контейнери починають вигравати не тільки за швидкодією, але і за споживанням апаратних ресурсів, яке у них знижене, завдяки використанню UnionFS.

Це допоміжна файлова система, яка виробляє каскадно-об'єднане монтування інших файлових систем. Вона дозволяє файлам і каталогам ізольованих файлових систем, відомих як гілки, прозоро перекриватися, формуючи єдину пов'язану файлову систему. Каталоги, які мають той же шлях в об'єднаних гілках, будуть спільно відображати вміст в об'єднаному каталозі нової віртуальної файлової системи. Коли гілки монтуються, то вказується пріоритет однієї гілки над іншою. Отже, коли обидві гілки містять файл з ідентичним ім'ям, одна гілка матиме більший пріоритет. Різні гілки можуть одночасно перебувати в режимі «тільки читання» і «читання-запис».

Union File System складається з шарів (layers). Шари як би накладаються один на одного. Всі використовувані нами контейнери використовують загальні захищені від запису шари, в яких знаходяться незмінні файли операційної системи

контейнера, це призводить до прискорення доступу до даних. А для змінюваних файлів - файлів віртуальних машин, що працюють в Docker, - кожен з контейнерів матиме власний шар. Природно, Docker використовує це рішення не тільки для операційної системи, але і для будь-яких загальних частин контейнерів, які були створені на основі загальних «предків» їх образів; швидше за все, що за рахунок цього він і виграє по швидкості роботи. Тобто виходить, що віртуальні машини, що використовують технологію гіпервізорної віртуалізації, створюють багато ідентичних копій операційних систем, які ніяк не пов'язані між собою і споживають більше апаратних ресурсів сервера, ніж контейнери, що працюють на технології контейнерної віртуалізації з використанням Union File System.

Висновки до третього розділу

1. Віртуалізація на рівні операційної системи - метод віртуалізації, при якому ядро операційної системи підтримує кілька ізольованих примірників простору користувача. Ці екземпляри (часто звані контейнерами або зонами) з точки зору користувача повністю ідентичні реальному обчислювальному модулю. Ядро забезпечує повну ізольованість контейнерів, тому програми з різних контейнерів не можуть впливати один на одного.

2. LXC - система віртуалізації на рівні операційної системи для запуску декількох ізольованих екземплярів ОС Linux на одному комп'ютері. LXC не використовує віртуальні машини, а створює віртуальне оточення з власним простором процесів і мережевим стеком. Всі екземпляри LXC використовують один екземпляр ядра ОС. LXC заснована на технології ядра Linux під назвою «Контрольні групи» (cgroups, додано у версії ядра 2.6.29) с використанням механізму ізоляції «простір імен» (namespaces). Часто технологія контейнерної віртуалізації розглядається як поліпшена реалізація механізму «пісочниці» chroot.

3. Docker - це вільно-розповсюджувана платформа для розгортання та експлуатації додатків. Метою Docker є більш швидкий та простий запуск додатків, а також легке перенесення з одного хоста в інший. Він дозволяє швидше тестувати і швидше розробляти додаток. Docker - це контейнерна віртуалізація, тобто легка платформа віртуалізації, яка вміє запускати різні програми, написані на різних мовах програмування і на різних технологіях.

4. Docker складається з трьох компонентів:

• Образи - шаблони тільки для читання. Наприклад, образ може містити в собі операційну систему Ubuntu з Apache і додатком. Вони призначені для створення контейнерів.

• Реєстр - сховище образів. Розробник може створити свій образ або завантажити вже створений образ з реєстру Docker Hub.

• Контейнери - створюються з образів. Можна зупиняти, запускати і видаляти контейнери. У контейнерах міститься все, що потрібно для роботи програми.

Образи складаються з набору рівнів, а Docker використовує Union File System для перетворення цих рівнів в один образ. Union file system дозволяє файлам і директоріями з різних систем прозоро накладатися, створюючи когерентну файлову систему. Завдяки цим рівням Docker легковагий і має високий рівень швидкості.

5. (1) Контейнерна віртуалізація може бути використана при організації високопродуктивних обчислень як засіб розгортання спеціалізованих програмних платформ, в тому числі створених користувачем. Накладні витрати на контейнерну віртуалізацію при розгортанні програмної платформи істотно менше, ніж на віртуалізацію за допомогою гіпервізора.

(2) Накладні витрати на контейнерну віртуалізацію істотно зростають, якщо мережа інформаційних обмінів використовує стек протоколів TCP / IP.

(3) На сьогоднішньому етапі розвитку контейнери не здатні виключити взаємний вплив завдань, які поділяють один і той же обчислювальний модуль, застосування контейнерів не вирішує проблеми фрагментації обчислювальних ресурсів при колективному доступі.

6. Таким чином, контейнерна віртуалізація є більш ефективною в плані витрачання пам'яті, але втрачає в швидкодії через додаткові витрати на обчислення адрес. Сам гіпервізор працює наступним чином: в операційній системі хоста емулюється апаратне забезпечення, поверх якого запускаються гостьові операційні системи. Це дає нам зрозуміти, що зв'язок між гостьовою і хостовою операційними системами дотримується чіткого правила: все, що робить використовувана апаратна частина, має бути доступне і гостьовій операційній системі з боку хостової.

ВИСНОВКИ

В ході виконання наукового дослідження було проведено комплексне дослідження застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів.

За результатами проведеного дослідження було отримано наступні висновки:

1. Досліджено понятійно-категоріальний апарат в сфері технологій віртуалізації: (а) хмарні обчислення (cloud computing) - це комплекс технологій, що забезпечують повсюдний і зручний мережевий доступ до динамічно масштабуємих обчислювальних ресурсів (процесорного часу, оперативної пам'яті, пристроїв зберігання даних, мереж передачі даних, додатків та ін.); (б) хмарна інфраструктура являє собою набір апаратних і програмних засобів, що дозволяють досягти п'яти перерахованих вище основних характеристик. Вона може складатися з фізичного і абстрактного рівнів. Фізичний рівень - це апаратні засоби, необхідні для надання хмарних сервісів, вони зазвичай включають в себе сервери, пам'ять і мережеві компоненти. Абстрактний рівень - це програмне забезпечення, розгорнуте на фізичному рівні і наділене основними хмарними характеристиками. Абстрактний рівень розташовується над фізичним; (в) поняття хмарного сервісу (Cloud Service) є більш широким порівняно з хмарними обчисленнями (Cloud Computing), оскільки включає технології зберігання й іншого сервісу, термін "хмарний сервіс" передбачає використання спеціальної клієнт-серверної технології використання користувачем ресурсів (процесорний час, оперативна пам'ять, дисковий простір, програмне забезпечення та ін.); (г) віртуалізація - надання набору обчислювальних ресурсів або їх логічного об'єднання, абстрагованого від апаратної реалізації, і такого, що забезпечує при цьому логічну ізоляцію обчислювальних процесів один від одного, які виконуються на одному фізичному ресурсі.

2. Досліджено апаратні, програмні та організаційні особливості реалізації хмарних сервісів: У класичному варіанті хмарної обчислювальної системи можна виділити шість основних компонент: (а) по-перше, фізичні машини і їх операційні системи. Ми виділяємо їх, тому що: а) якщо процесори, використовувані в машинах, не підтримують технологію апаратної віртуалізації, то ми можемо говорити лише про паравіртуалізацію, і це істотно знижує як обчислювальні здатності системи, так і вибір програмного забезпечення в подальшому; б) системи з відкритим кодом більшою мірою, ніж інші, повинні бути достатньо гнучкими для роботи з багатьма гостьовими ОС (а комерційні можуть бути вузько спеціалізовані на роботу з певним набором програмних компонент); (б) мережева компонента (використання мережі тягне за собою DNS, DHCP і організацію підмереж фізичних машин); (в) по-третє, використання гіпервізора. гіпервізор дозволяє працювати віртуальним машинам; (г) образи віртуальних машин, що зберігаються на віртуальному «жорсткому диску» - репозиторії, що охоплює всі образи всіх машин, запущених колись в хмарі; (д) необхідний інтерфейс для користувачів; (е) хмарне середовище — як система, що зв'язує воєдино всі шість компонент, розподіляє ресурси користувачам, обслуговуюча мережа та ін.

3. Здійснено порівняльний аналіз переваг і недоліків існуючих технологій створення хмарних сервісів: Крім віртуалізації за допомогою гіпервізора існує так звана контейнерна віртуалізація. Відмінність між цими підходами полягає в наступному: гіпервізор емулює апаратне забезпечення, поверх якого запускаються гостьові ОС. При цьому все, що «вміє» робити обладнання, повинно бути доступно гостьовій ОС з боку базової ОС (тобто машини-«господаря»). Навпаки, контейнери - це віртуалізація на рівні операційної системи, а не на рівні обладнання: кожна гостьова ОС використовує те ж саме ядро (а в ряді випадків - і інші частини ОС), що і базова. В результаті контейнери менше і компактніше гіпервізорних гостьових середовищ, оскільки у них з «базою» набагато більше спільного.

4. Вивчено особливості застосування апаратної віртуалізації: Можливість апаратної віртуалізації додала цілий ряд переваг:

- спрощення розробки програмних платформ віртуалізації за рахунок надання апаратних інтерфейсів управління і підтримки віртуальних гостьових систем;

- можливість збільшувати швидкодію платформ віртуалізації;

- поліпшення захищеності, можливість перемикання між декількома запущеними незалежними платформами віртуалізації на апаратному рівні;

- запуск 64-бітних гостьових систем на 32-бітних хостових системах.

5. Здійснено аналіз віртуальних машин: поняття "монітор віртуальних машин" (МВМ) з'явилося у кінці 60-х років як програмний рівень абстракції, який розділяв апаратну платформу на кілька віртуальних машин. Кожна з цих віртуальних машин (ВМ) була настільки схожа на базову фізичну машину, що існуюче програмне забезпечення могло виконуватися на ній в незмінному вигляді. Це був час, коли обчислювальні завдання виконувались на дорогих мейнфреймах (типу IBM/360), і тому користувачі не могли не оцінити здатність МВМ розподіляти дефіцитні тоді ресурси серед кількох додатків.

6. Досліджено особливості використання гіпервізорів: загальноприйнята класифікація гіпервізора: - автономний гіпервізор (має свої вбудовані драйвери пристроїв, моделі драйверів і планувальник і тому не залежить від базової ОС, автономний гіпервізор працює безпосередньо на обладнанні і є більш продуктивним, приклад - VMware ESX; на основі базової ОС, працює в одному кільці з ядром основної ОС, приклади: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox; гібридний, складається з двох частин: з тонкого гіпервізора, контролюючого процесора і пам'яті, а також працює під його керуванням спеціальної сервісної ОС в кільці зниженого рівня, приклади: Xen, Citrix XenServer, Microsoft Hyper-V.

7. Вивчено використання апаратної віртуалізації в контексті інформаційної безпеки: (1) проблеми технологічні і фізичні. В першу чергу, необхідно взяти до

уваги, в якому центрі обробки даних (далі - ЦОД) знаходиться обладнання. ЦОД повинен бути захищений від припинення надання послуги в результаті перебоїв з електроживленням, катастроф техногенного характеру, фізичного пошкодження обладнання в результаті аварій, необережних дій персоналу, диверсій зловмисників та ін.; (2) проблеми психологічні; (3) юридичні проблеми.

8. Досліджено технології контейнерної віртуалізації: особливості застосування: (1) контейнерна віртуалізація може бути використана при організації високопродуктивних обчислень як засіб розгортання спеціалізованих програмних платформ, в тому числі створених користувачем. Накладні витрати на контейнерну віртуалізацію при розгортанні програмної платформи істотно менше, ніж на віртуалізацію за допомогою гіпервізора; (2) накладні витрати на контейнерну віртуалізацію істотно зростають, якщо мережа інформаційних обмінів використовує стек протоколів TCP / IP; (3) на сьогоднішньому етапі розвитку контейнери не здатні виключити взаємний вплив завдань, які поділяють один і той же обчислювальний модуль, застосування контейнерів не вирішує проблеми фрагментації обчислювальних ресурсів при колективному доступі.

9. Порівняно контейнерну та гіпервізорну віртуалізацію: контейнерна віртуалізація є більш ефективною в плані витрачання пам'яті, але втрачає в швидкодії через додаткові витрати на обчислення адрес. Сам гіпервізор працює наступним чином: в операційній системі хоста емулюється апаратне забезпечення, поверх якого запускаються гостьові операційні системи. Це дає нам зрозуміти, що зв'язок між гостьовою і хостовою операційними системами дотримується чіткого правила: все, що робить використовувана апаратна частина, має бути доступне і гостьовій операційній системі з боку хостової.

Практичне впровадження одержаних результатів – проведено комплексне дослідження застосування технологій віртуалізації і контейнерізації в системах хмарних сервісів. Матеріали дослідження можуть бути корисні фахівцям в галузі комп'ютерних наук.

Галузь застосування – галузь комп'ютерних наук.