Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Метод створення віртуальних полігонів на основі технологій хмарних обчислень
Назва теми

Галузь знань 12 – Інформаційні технології

Спеціальність 123 – Комп'ютерна інженерія

КРМКІ. 015065.19.01.13 ПЗ

Виконав: студент 2 курсу, група КІ1м-19-1 Підпис Лукін В.С.

Керівник проф., д. т. н, професор кафедри КбКСМ Підпис Чешун В.М.

Нормоконтролер доц., к. т. н, доцент кафедри КбКСМ Підпис Муляр І.В.

До захисту допускаю:
Зав. кафедри КбКСМ, к.т.н., доцент Підпис Кльоц Ю.П.

9.12.20 2020 р.

Хмельницький, 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ПРОГРАМУВАННЯ ТА КОМП'ЮТЕРНИХ І ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Кафедра КАФЕДРА КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ПРОГРАМУВАННЯ ТА ЗАХИСТ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П.Кльоц

"01" 09 2020 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лукін Владислав Сергійович

Прізвище, ім'я, по батькові студента

1. Тема роботи Метод створення віртуальних полігонів на основі технологій хмарних обчислень

Керівник роботи Чешун В.М.,

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

кандидат технічних наук, доцент

Затверджена наказом № 118 ректора університету додаток №23 від 01.09.2020

2. Строк подання студентом проекту (роботи) на кафедру 20.11.2020

3. Вихідні дані до проекту (роботи) віртуальні полігони, хмарні обчислення, комп'ютерні системи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ. Дослідження технологій віртуалізації. Обчислювальна підсистема. Гіпервізори. Мережева підсистема. Віртуалізація. Системи зберігання даних. Відкриті платформи віртуалізації. Архітектура віртуальних полігонів. Ресурсні пули. Висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Загальна характеристика магістерської роботи. Типи гіпервізорів. Фізична мережева інфраструктура. Віртуалізована мережева інфраструктура. Розподілений віртуальний комутатор. Апробація методу - результати моделювання. Висновки.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Муляр І.В. доцент кафедри КбКСМ		

7. Дата видачі завдання «_2_» _____вересня_____ 2020р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Вибір напряму дослідження та узгодження тематики КРМ з керівником	2.02.2020	
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	2.03.2020	
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	1.04.2020	
4	Робота над розділом 2 – розробка моделей і методів для вирішення поставленої задачі	1.05.2020	
5	Робота над науковою публікацією	1.06.2020	
6	Уточнення і затвердження теми	1.09.2020	
7	Робота над розділом 3 – розробка методу та алгоритмів, їх аналіз	2.09.2020	
8	Робота над розділом 4 – апробація запропонованих рішень	1.10.2020	
9	Узгодження отриманих результатів; оформлення пояснювальної записки згідно вимог	1.11.2020	
10	Оформлення графічної частини	8.11.2020	
11	Попередній захист роботи	10.11.2020	
12	Захист роботи на засіданні ЕК	5.12.2020	

Студент _____ _____
Підпис Ініціали, прізвище

Керівник роботи _____ _____
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод створення віртуальних полігонів на основі технологій хмарних обчислень

Автор роботи: Лукін Владислав Сергійович

Керівник роботи: к.т.н., доцент. Чешун В.М.

Загальний обсяг роботи: 92 сторінки, 31 рисунок, 16 таблиць, 2 додатки, 40 посилань.

МЕТОД СТВОРЕННЯ ВІРТУАЛЬНИХ ПОЛІГОНІВ НА ОСНОВІ ТЕХНОЛОГІЙ ХМАРНИХ ОБЧИСЛЕНЬ.

Метою кваліфікаційної роботи є розроблення методу, алгоритмів створення віртуальних полігонів.

Дана кваліфікаційна робота присвячена подальшій розробці методології проектування інфраструктури віртуальних полігонів, досліджені різні аспекти роботи системи і її підсистем: обчислювальної підсистеми (підсистема віртуалізації, гіпервізор), мережевої підсистеми, системи зберігання даних, системи автоматизації надання послуг і забезпечення універсального доступу.

Дата Підпис студента

ANNOTATION

a qualification work of Lukin Vladislav

entitled «A method of creating a shift cipher using optimal Huffman coding to increase the efficiency of protecting computer systems.».

Mentor: Ph.D. Cheshun V.

Total volume of work: 92 pages, 31 figures, 16 tables, 2 appendices, 40 references.

METHOD OF CREATING VIRTUAL POLYGONS BASED ON CLOUD COMPUTING TECHNOLOGIES

The purpose of the qualification work is to develop a method, algorithms for creating virtual polygons.

This qualification work is devoted to the further development of the methodology of designing the infrastructure of virtual landfills, explored various aspects of the system and its subsystems: computing subsystem (virtualization subsystem, hypervisor), network subsystem, storage system, automation system and universal access.

Date Signature

ЗМІСТ

ВСТУП

Хмарні обчислення - модель мережевого доступу до деякого загального фонду обчислювальних ресурсів (наприклад, мереж передачі даних, пристроїв зберігання даних, серверів, додатків і сервісів - як разом, так і окремо), які можуть бути оперативно надаватись та звільнятись з мінімальними експлуатаційними витратами або зверненнями до провайдера.

Споживачі хмарних обчислень можуть значно зменшити витрати на інфраструктуру інформаційних технологій (в короткостроковому і середньостроковому планах) і гнучко реагувати на зміни обчислювальних потреб, використовуючи властивості обчислювальної еластичності хмарних послуг.

З моменту появи в 2006 році концепція глибоко проникає в різні IT-сфери і займає все більш і більш вагому роль в практиці

Об'єкт дослідження – Технології організації хмарних обчислень.

Предмет дослідження – використання віртуальних полігонів при організації хмарних обчислень.

Мета і завдання дослідження дипломної роботи. Метою дослідження є створення методу побудові віртуальних полігонів.

Відповідно до поставленої мети в дипломній роботі поставлені, і вирішені наступні задачі:

1. Провести дослідження існуючих моделей, методів, а також алгоритмів використовуваних в побудові хмар.

2. Розробити метод створення віртуальних полігонів.

Методи дослідження. Для вирішення задач поставлених в дипломній роботі використовуються методи імовірнісних графів, системного аналізу, теорії ймовірності, теорії прийняття рішень, випадкових процесів і математичної статистики, методів комп'ютерного аналізу, методів модульного і структурного програмування, математичного моделювання.

Наукова новизна одержаних результатів:

1. Моделі представлення релевантності подібності рядків, дозволяє надати, для рядків пошуку, кількісну оцінку їх подібності.

2. Удосконалений метод оптимізації запису інформації в бази даних, забезпечує розпізнавання та виключення дублювання даних при використанні інформаційно – пошукових систем, на основі автоматичного вибору схеми ручної або автоматичної ідентифікації, що дозволяє зберегти інформаційну цілісність, а також знизити зашумленість даних, зумовлену наявністю помилок операторського введення.

Практична цінність результатів дипломної роботи. Запропоновані алгоритми під час виконання дипломної роботи, можуть бути використання в якості типових рішень при проектуванні та застосуванні подібних систем для підприємств середнього та малого бізнесу. Достовірність наукових положень, висновків отриманих в дипломній роботі результатів підтверджується коректною постановкою задач, результатами 10 моделювання та апробацією результатів отриманих на конференціях, коректністю використовуваного математичного апарату. Отримані в ході виконання дисертаційного дослідження результати не суперечать раніше отриманим даним, описаним в літературі іншими авторами. Особистий внесок. Всі дослідження, викладені в дипломній роботі, проведені автором в процесі наукової діяльності. Результати, які виносяться на захист, отримані автором особисто, запозичений матеріал позначений в роботі посиланнями.

Апробація роботи. За темою дипломної роботи ОКР «Магістр» опубліковано 1 теза. Структура і обсяг роботи. Дипломна робота ОКР «Магістр» складається зі вступу, основної частини, що містить 4 розділи, висновків і списку використаних джерел. Загальний обсяг роботи - 105 сторінок. Робота містить 11 рисунків та 10 таблиць. Список використаної літератури включає 33 бібліографічних джерела.

1 ТЕХНОЛОГІЙ ВІРТУАЛІЗАЦІЇ

1.1 Технології віртуалізації в освіті

Технології віртуалізації супроводжували розвитку комп'ютерної техніки протягом всієї історії розвитку електронних обчислювальних машин. З розвитком обчислювальної техніки, перед розробниками завжди стояли завдання подолати відмінності і обмеження конкретних реалізацій елементів електронних обчислювальних машин, уніфікувавши їх взаємодію і представивши їх для программного забезпечення і операційної системи у вигляді абстрактних інтерфейсів, відв'язаних від конкретних реалізацій.

Технології віртуалізації конкретних пристроїв з'являлися і застосовувалися для вирішення різних технічних завдань проектування обчислювальної техніки, щознайшловідображення в таких технічних рішеннях, як: технології планування процесів і забезпечення багатозадачності операційної системи, уявлення різних рівнів пам'яті у вигляді віртуальної пам'яті з єдиною системою адресації, віртуалізаці япристрої ввведення-виведення в операційних системах Unix і Linux, віртуалізація мережевих інтерфейсів і багатьох інших. Паралельно вирішувалося завдання віртуалізації всього обчислювального вузла для оптимізації роботи з апаратно-програмними комплексами.

Необхідність в віртуалізації всього обчислювального вузла вже в тому вигляді, як це реалізується в даний час для вирішення різного роду прикладних задач, вперше з'явилася більше 40 років тому при проектуванні мейнфреймів. Надати весь суперкомп'ютер в розпорядження одного користувача на необхідний для його роботи час було б занадто марнотратно з точки зору ефективності використання ресурсів, що не оптимально економічно і організаційно, тому спочатку при проектуванні супер-комп'ютерів вирішувалося завдання надання одночасного доступу до обчислювальних потужностей мейнфремів великої кількості користувачів з можливістю виконувати різні процеси і додатки в ізольованих віртуальних середовищах з виділеними ресурсними квотами, при

цьому надавати ресурси так, як якщо б кожен користувач працював на мейнфреймі один. Це знаходило відображення в усіх аспектах побудови апаратно-програмних комплексів мейнфреймів, тому що дані завдання були передбачені в самій архітектурі рішення.

Комп'ютери архітектури x86 на відміну від мейнфреймів спочатку не проектувалися для задач віртуалізації і не були призначені для розділу режиму роботи користувачів. Особливості та обмеження архітектури процесора, стоять перед комп'ютерами того часу завдання і особливості проектування системного програмного забезпечення на тривалий час відклали розвиток технологій повної віртуалізації всього обчислювального вузла для комп'ютерів даної архітектури.

Основним завданням того часу для архітектури x86 в розрізі розвитку технологій віртуалізації була завдання емуляції – надати програмному забезпеченню саме те апаратне оточення, під яке була колись написана програма. З'явилися виконавчі емулятори різних і часто застарілих архітектур, що працюють поверх операційної системи і дозволяють запускати програми не перевстановлюючи операційну систему. Дана технологія більше відповідала прикладному аспекту – спростити процеси тестування і налагодження програмного забезпечення, а так само для проведення навчання роботі з певними системними архітектурами.

Але стрімкездешевлення і зростання поширеності комп'ютерів сімействах86, тенденція до відкритості та переносимості програмного забезпечення, а так подібне зростання потужностей комп'ютерів даної серії на початку 90-х років і їх вихід в серверний сегмент, привели до того, що виникли проблеми, які знову змусили звернутися до технологій повної віртуалізації всього обчислювального вузла істворення віртуальних кластерів для їх вирішення.

Серед цих проблем можна виділити наступні [1]:

– Низькі коефіцієнти використання обчислювальних ресурсів при зростаннікількості обчислювальних вузлів. За даними компанії IDC,що спеціалізується на дослідженнях ринку, на сьогоднішній день середнійкоефіцієнт використання обчислювальних потужностей в звичайному середовищі

зсерверами x86 становить всього 10-15% від загального обсягу ресурсів. Здебільшогоце відбувається через те, що для великої кількості інформаційних систем,організації виконують один додаток на одному сервері, щоб уникнутивпливу вразливостей однієї програми на доступність іншого,працюючого на тому ж сервері.

– Зростаючі витрати на фізичну інфраструктуру. Експлуатаційні витрати напідтримку інфраструктури стабільно ростуть. Сьогодні велика частинаобчислювальної інфраструктури повинна працювати цілодобово і постійно, щотягне за собою витрати на електроживлення, охолодження і оренду, що незалежать від коефіцієнтів використання.

– Зростаючі витрати на службу підтримки. З ускладненням інформаційних системрівні спеціалізованої освіти і досвіду, якими повинні володітифахівці з управління інфраструктурою, і пов'язані з цим витратиістотно зростають. Організації витрачають непропорційно багато часу іресурсів на виконувані вручну завдання з обслуговування апаратно-програмних комплексів. Це означає необхідність в наймі додатковихспівробітників для виконання цих завдань.

– Тривалий і неефективне аварійне перемикання і слабкий захист від збоїв.Організації все більше страждають від простоїв важливих клієнтських і серверних додатків. Загрози інформаційної безпеки, природних і техногенних катастроф підвищує важливість плану забезпечення безперервності роботиінформаційних систем для настільних комп'ютерів і серверів.

Протягом більш ніж десятирічного періоду з 2000-х років технології віртуалізаціїкомп'ютерів архітектури x86 пройшли інтенсивний шлях становлення і на поточниймомент вийшли на рівень можливостей широкого промислового використання,продуктивність віртуалізованних пристроїв стала порівнянна з роботою в звичайномурежимі, що підтверджують останні тести продуктивності гіпервізора [2].Широке поширення і здешевлення технології, поява повнофункціональногопрограмного забезпечення з відкритим вихідним кодом, підтримка усімапоширеними виробниками процесорів режиму апаратної

віртуалізації -все це зробило можливим застосування технологій віртуалізації на звичайнихбюджетних серверах і робочих станціях.

В даний час технології віртуалізації набули широкого поширенняпри проектуванні IT-інфраструктур за рахунок можливості більш ефективновикористовувати наявні обладнання та розпоряджатися наявними ресурсами, а також дали абсолютно нові переваги, такі як:

– можливість легко переносити віртуальні машини між серверами по мережі і нарізних носіях інформації за рахунок подання віртуальних машин у вигляді звичайних файлів,
– можливість зберігати повне стан віртуальної машини за допомогою техніки миттєвих знімків (snapshot) і вільно переміщатися між цими станами,
– переносити працюючу віртуальну машину з усім набором програмногозабезпечення з одного сервера на інший без зупинки роботи всіх запущенихсервісів за рахунок технологій -Живий міграції,
– динамічне балансування навантаження між віртуалізованних серверами засобами систем віртуалізації,
– майже 100% -ва ізоляція віртуальних машин з метою забезпеченняінформаційної безпеки, наприклад для тестування програмногозабезпечення,
– створення відмовостійких рішень і кластерів засобами системвіртуалізації,
– і багато інших.

Крім незаперечних переваг при проектуванні промислових ITінфраструктур, технології віртуалізації внесли значний вклад в розвитокможливостей для забезпечення освітнього процесу в сфері інформаційнихтехнологій. Стало можливим швидке створення тестових стендів і освітніхполігонів, обмін готовими конфігураціями віртуальних машин, експериментуванняз різними конфігураціями і мережевими топологиями без серйозних витрат часуі на базі наявного обладнання.

Завдяки технологіям віртуалізації стало можливим проводити навчання іздійснювати моделювання в областях, які пов'язані зі створенням

розподіленихгетерогенних IT-інфраструктур, що складаються з великої кількості примірниківвіртуальних машин на базі невеликого кількість консолідованих північнихпотужностей, моделюванням багатопроцесорних конфігурацій для задач багато поточного програмування, моделюванням різних мережевих завдань.

1.2 Хмарна модель надання послуг

Технології віртуалізації несуть величезні можливості для створення віртуальних тестових середовищ (полігонів) для навчальних цілей, але їх слід доповнити хмарнимимоделями надання услугт.к. Зараз через масштабність проведенихекспериментів, вимог до зручності і швидкості проведення робіт, це пов'язано з рядомпевних технічних складнощів і обмежень через відсутність готовихінструментів.

Послуги, які надаються по хмарної моделі, згідно з визначеннямНаціонального Інституту Стандартів і Технологій США [3], повинні обов'язкововідповідати наступним п'яти характеристикам:

1. Самообслуговування на вимогу (англ. Selfserviceondemand) – споживачсамостійно визначає і змінює обчислювальні потреби, такі як,наприклад, серверне час, обсяг і параметри обчислювальних ресурсів,швидкості доступу і обробки даних, об'єм збережених даних, шаблони наборівіртуальних машин, без взаємодії з представником постачальника послуг;

2. Універсальний доступ по мережі - послуги повинні бути легко доступніспоживачам через мережу передачі даних незалежно від використовуваноготермінального пристрою;

3. Об'єднання ресурсів (англ. Resourcepooling) - постачальник послуг об'єднуєресурси для обслуговування великого числа споживачів в єдиний пул длядинамічного перерозподілу потужностей між споживачами в умовахпостійної зміни попиту на потужності і легкого масштабування загальногообсягу обчислювальної ферми; при цьому споживачі контролюють тількиосновні параметри послуги (наприклад, об'єм обчислювальних ресурсів, обсягзбережених

даних, швидкість доступу до ресурсів), але фактичний розподілресурсів, що надаються споживачеві, здійснює постачальник (в деякихвипадках споживачі все-таки можуть управляти деякими фізичнимипараметрами перерозподілу, наприклад, вказувати бажаний центр обробкиданих з міркувань географічної близькості);

4. Еластичність-обсяг послуг може бути збільшений або зменшений будь-який моментчасу, без додаткових витрат на взаємодію з постачальником, якправило, в автоматичному режимі;

5. Облікспоживання - постачальник послуг автоматично обчислює споживанняресурси на визначеному рівні абстракції, зручному для споживача (наприклад,об'єм збережених даних, пропускна здатність, кількість користувачів,кількість транзакцій), і на основі цих даних оцінює об'ємнаданих споживачам послуг.

Частина цих характеристик забезпечується функціональними можливостямидеяких гіпервізорів (моніторів віртуальних машин) і супутньогопрограмного забезпечення (toolstacks, наборів керуючих компонент), наприклад -можливість об'єднання ресурсів в ресурсні пули, і питання стоїть лише в виборінайбільш відповідного гіпервізора. Деякі характеристики на поточний моментзабезпечується гіпервізорами частково - наприклад, облік споживання ресурсів. Цейпитання вирішується вибором додаткового програмного забезпечення і інтеграцією йогов хмарну систему, і для цілей цієї роботи може бути менш важливий, ніж іншіхарактеристики хмарної моделі.

Але самі по собі технології віртуалізації майже не забезпечують важливих дляосвітніх цілей можливостей самообслуговування користувачів: автоматизаціїрозгортання віртуальних машин і управління освітніми полігонами,забезпечення зручними інструментами універсального доступу для користувачів допослузі.

Проектування, розгортання і тестування розподілених інфраструктур і систем для моделювання різних завдань може бути здійснено на хмарної інфраструктурі, підкріпленої усіма необхідними вищепереліченими інструментами, без значних інвестицій в закупівлю обладнання та складнощів зчастою

налаштуванням і перенастроюванням одного і того ж обладнання під різнінавчальні та наукові завдання і з точки зору автора роботи несе незаперечніпрактичні переваги від застосування даної технології в галузі освіти.

Щоб зробити технологію широко застосовувану для освітніх цілей, в рамкахроботи була зроблена спроба забезпечити створення докладної методології створенняапаратно-програмного комплексу і працюючого прототипу масштабируемойсервісної інфраструктури освітніх полігонів з урахуванням забезпечення найбільшважливих з перерахованих вище характеристик хмарної моделі надання послуг.

Створення такої інфраструктури пов'язане з різними аспектами проектуванняінформаційних систем - створенням оптимальної обчислювальної підсистеми (системивіртуалізації), мережевої підсистеми, системи зберігання даних, системи автоматизаціїнадання послуг і забезпечення універсального доступу.

1.3 Вибір теми та її актуальність

Тема роботи була вироблена і прийнята виходячи їх актуального завдання, що стоїтьперед відділом розподілених обчислень ІППІ РАН - завдання забезпечення навчальногопроцесу необхідними апаратно-програмними засобами для навчання студентів іаспірантів технологіями розподілених обчислень на працюючих багатопроцесорних багато вузлових системах. Апаратно-програмний комплекс повинен бутилегко-масштабується, легким в обслуговуванні і відповідним вимогамиосвітнього процесу.

В даний час для навчання використовуються фізичні сервера і робочістанції з встановленими на них операційними системами сімейства Linux іспеціалізованим програмним забезпеченням та бібліотеками: MapReduce, OpenMP,MPI, Hadoop, BOINC і іншим.

Кожен раз постає необхідність готувати апаратно-програмний комплекс долабораторних робіт, перевіряти його працездатність, вручну забезпечувати

доступстудентам до реальної фізичної системі, управляти обліковими записами користувачів ідоступом студентів до системи, а потім після роботи усувати неполадки абовручну приводити систему в первинний стан у зв'язку зі змінами,внесеними студентами.

У даній роботі зроблена спроба, по-перше, формалізувати і узагальнитизавдання, що стоять перед науковим або навчальним закладом, який прагневикористовувати технології віртуалізації і хмарних обчислень для створенняосвітніх полігонів. По-друге, підібрати відповідні засоби з відкритимвихідним кодом, якщо такі існують. По-третє, зробити необхіднідослідження, доопрацювання та удосконалення для забезпечення відповідності апаратно-програмного комплексу характеристикам хмарної моделі надання послуг. По-четверте, спроектувати готове працююче рішення, яке задовольняє завданнямнаукового або навчального закладу з даної тематики.

Складність реалізації даного завдання зумовлена великою кількістюпідсистем, що входять в інфраструктуру віртуальних полігонів, і необхідністюзабезпечити злагоджену роботу всіх підсистем для вирішення поставленого завдання, а так самовідсутністю літератури з побудови такого роду систем.

1.4 Постановка завдання

Метою цієї роботи є створення методології проектування і прототипу апаратно-програмного комплексу освітніх полігонів, який забезпечить виконання наступних завдань, що стоять перед науковою або освітнію установою:

– створенням розподілених гетерогенних IT-інфраструктур, що складаються звеликої кількості примірників віртуальних машин на базі невеликої кількісті консолідованих північних потужностей,

– моделювання багатопроцесорних і багатоядерних конфігурацій обладнанняз відповідним набором програмного забезпечення для виконання завданьбагатопоточного програмування,

– моделювання мережевих завдань на базі віртуальних кластерів,

– забезпечення універсального доступу до віртуальних освітнім полігонівдля студентів, аспірантів та академічного складу.

Прототип системи в якості тестових випробувань повинен мати можливість бути використаний для розгортання та забезпечення роботи:

– Одиничних не зв'язаних між собою примірників операційних систем в кількості, достатній для проведення лабораторної роботи (15 окремихвіртуальних машин одночасно) з встановленими компіляторами C ++ gcc,бібліотеками для багатопотокового програмування OpenMP і MPI;

– Кластерів для навчання технологіям розподілених обчислень MapReduce (5кластерів працюють одночасно, кожен з яких складається як мінімум з3-х віртуальних машин).

Завданнями цієї роботи є:

– дослідження і вибір існуючих систем, придатних для реалізації цілей і задач цієї роботи,

– проектування апаратно-програмного комплексу, включаючи дослідження іпобудова всіх вхідних в його склад підсистем,

– створення працюючого прототипу апаратно-програмного комплексу,

– вимір ефективності прототипу.

– В рамках поставлених завдань повинні бути розроблені наступні підсистеми апаратно-програмного комплексу та забезпечено їх взаємодія для виконання цілей цієї роботи:

– Обчислювальна підсистема (система віртуалізації);

– Мережева підсистема;

– Система зберігання даних;

– Система автоматизації надання послуг та забезпеченняуніверсальногодоступу.

Вимоги

Загальними вимогами до апаратно-програмного комплексу є:

– використання вільно поширюваного програмного забезпечення звідкритим вихідним кодом - система повинна мати можливість вільноговикористання і доопрацювання без ліцензійних обмежень для навчальних,освітніх і наукових цілей. Дана вимога так само забезпечуєвільну тиражованою створеної системи під власним найменуванням,

– масштабованість системи - можливість збільшувати сукупну ємністьобчислювальної, мережевий підсистеми і системи зберігання даних при зростаннінавантажень на систему без істотної зміни архітектури системи,

– висока продуктивність і відмовостійкість системи.

2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Обчислювальна підсистема

В основі обчислювальної підсистеми (віртуальної інфраструктури) лежитьтехнологія гіпервізора або моніторів віртуальних машин – програмногозабезпечення близького за призначенням до операційної системи, що забезпечує ізоляцію і розподіл ресурсів між віртуальними машинами, які працюють поверх гіпервізора, і управління цими віртуальними машинами [4].

Застосування гіпервізора дозволяє по-перше, розділити всі ресурси системи на будь-яку кількість ізольованих віртуальних машин з заданими характеристиками і, по-друге, відв'язати операційну систему від конкретного обладнання за рахунок однаковості шаблонів і образів працюють віртуальних машин, зробивши віртуальну машину легко переносимо між фізичними пристроями різних архітектур, що працюють під управлінням одного і того ж гіпервізора.

Гіпервізор забезпечує виконання наступних основних функцій:
надання набору віртуального апаратного забезпечення для вищого програмного забезпечення (операційної системи і прикладного програмного забезпечення), що володіє всіма необхідними параметрами фізичної машини, і диспетчеризацію ресурсів для коректної та ефективної організації роботи вищого програмного забезпечення.

Рис. 2.1. Архітектура гіпервізора

Існують різні архітектури гіпервізора, основна відмінність яких полягає в тому, чи знає вищерозміщена операційна система про те, що вона працює у віртуальному середовищі і чи може з нею коректно взаємодіяти, або операційна система повинна працювати так, як вона працювала б на «голому» залізі без модифікацій, нічого не знаючи про архітектуру гіпервізора, тобто в режимі повної емуляції.

Перший підхід отримав назву паравіртуалізації, так як має на увазі тіснувзаємодія гіпервізора і операційної системи, за рахунок якого можна отримати значно більш ефективне використання ресурсів, але з іншого боку, тягне обов'язкове використання модифікованої операційної системи, яка повинна знати і вміти працювати з гіпервізором. Приклад такого гіпервізора - Xen, відмінні характеристики роботи досягнуті з операційними системами з відкритим кодом на основі ОС Linux, які можна легко модифікувати для ефективної роботи в зв'язці з гіпервізором.

Другий підхід - повна емуляція обладнання і можливість запускати не модифіковані операційні системи, що забезпечується сучаснимиможливостями процесорів з технологіями апаратної віртуалізації. прикладом таких гіпервізорів є KVM, VmwareESXi і HVM-режим роботи гіпервізораXen. Це єдиний прийнятний режим роботи для НЕ модифікуються операційних систем сімейства Windows.

Третя категорія ПО віртуалізації - виконавчі емулятори, які мають виглядзвичайних користувальницьких програм і відповідно працюють над рівнемкористувальницької операційної системи, виконуючи емуляціюнижче лежачогообладнання «на льоту», і не маючи прямого доступу до ресурсів, виключно за допомогою встановленої операційної системи. За рахунок цього вони мають високі накладні витрати на продуктивність, але дозволяють запускати віртуальні машини не переустанавлюючи кореневу операційну систему. Прикладами таких гіпервізора є VMware Workstation і OracleVirtualBox.

Розглянемо більш докладно найбільш популярні на сьогоднішній деньреалізації гіпервізора.

2.2 Гіпервізор KVM

Гіпервізор KVM є одним з найновіших та таких, що стрімко розвиваються гіпервізорів на сьогоднішній день. KVM був включений в ядро Linuxпочинаючи з версії 2.6.20 і на сьогоднішній момент є основою великої кількостіпрограмного забезпечення віртуалізації [5].

KVM сам по собі не є повноцінною платформою для віртуалізації, алеявляє собою частину більш великого технічного рішення. Підхід, якийреалізує KVM, аналогічний технології UML, і полягає в тому, щоб перетворитистандартне ядро Linux в гіпервізор простим завантаженням додаткового модуля ядра. Так само як і UML, KVM вимагає обов'язкової підтримки процесором режиму апаратної віртуалізації і працюємо в режимі повної віртуалізації.

Модуль ядра експортує пристрій, який називається /dev / kvm, що робитьможливим гостьовий режим ядра (до того ж до звичайних режимів ядра і користувачів). З dev / kvm віртуальна машина має свій власний адресний простір, окремий від адресного простору ядра або будь-яких інших працюючих віртуальних машин.

Пристрої в дереві пристроїв (/ dev) є загальними для всіх призначених для користувача процесів. Але модуль / dev / kvm забезпечує таку організацію роботи, що кожен процес, який отримує доступ до пристрою, бачить різні екземпляри пристроїв для підтримки ізоляції віртуальних машин.

Модуль KVM в складі ядра Linux

Модуль KVM знаходиться в директорії ./linux/drivers/kvm (у версіях ядра 2.6.20 і подальших). Ця директорія містить вихідні файли для KVM, а також файлипідтримки процесора для розширень Intel і AMD.

KVM перетворює ядро Linux в гіпервізор, коли відбувається інсталяція модуляядра KVM. Так як весь гіпервізор по суті є стандартним ядром Linux, він отримує переваги від змін в стандартному ядрі (підтримка пам'яті, планувальник і т.д.). Оптимізація цих компонент Linux (таких як новий O (1)планувальник в ядрі

2.6) дає переваги як гіпервізору (базова операційнасистема), так і гостьовій операційній системіLinux.

Коли модуль KVM активований, можна запускати будь-які інші операційнісистеми, такі як Linux або Windows в просторі користувачів. Кожна гостьоваоперaційна система являє собою окремий процес базової операційноїсистеми (гіпервізора).

На рисунку 2.2 показноархітектуругіпервізора KVM. В основі лежитьобладнання, яке має підтримку апаратної віртуалізації (в даний час це Intel VT або AMD-SVM процесор). Гіпервізор (ядро Linux з модулем KVM) працює на «голому» обладнанні, як будь-яка інша операційна система. З одногобоку гіпервізор виглядає точно також як стандартна операційна система Linux, в якої можна запускати інші програми. З іншого боку ядро може такожпідтримувати гостьові операційні системи, завантажені з допомогою утиліти kvm.

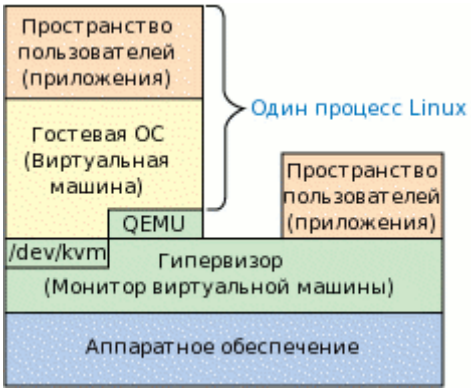


Рис. 2.2. Архітектура гіпервізора KVM

KVM є не зовсім самостійне, розроблене з нуля рішення, а «збірне» рішення віртуалізації. Процесор віртуалізується за рахунок підтримки фізичним процесором режиму апаратної віртуалізації, пам'ять віртуалізується за допомогою самого модуля kvm, система введення / виведення віртуалізується за допомогою модифікованого процесу QEMU, копія якого виконується з кожним процесом гостьовий операційної системи, мережева підсистема віртуалізується багато в чому засобами базової операційної системи.

KVM вводить новий режим процесів, який називається guest і використовується для запуску гостьової операційної системи. Режим guest забезпечує виконання сервісів гостьової операційної системи, але тільки сервісів, що не виконують операції введення / виводу.

Виконання операцій введення / виводу гостьовою операційною системоюзабезпечується QEMU. QEMU - це віртуалізаційна платформа, яка дозволяєвіртуалізувати майже все обладнання сервера або робочої станції (включаючи диски, графічні адаптери, мережеві пристрої). Будь-які запити введення / виведення, які робить гостьова операційна система, перехоплюються і направляються врежим користувача для емуляції за допомогою процесу QEMU.

Віртуалізація пам'яті забезпечується модулем / dev / kvm. Кожна гостьоваоперaційна система має свій власний адресний простір, який ініціалізується, коли створюється гостьова система. Фізична пам'ять, яка призначається для гостьової операційної системи, є в дійсності віртуальної пам'яттю процесу. Для перетворення гостьових фізичних адрес в реальні фізичні адреси використовується набір «тіньових» таблиць.

Процесор також підтримує процес перетворення пам'яті, передаючиуправління гіпервізору, коли є звернення до нерозподіленої адреси пам'яті.

Процес установки нової гостьової операційної системи забезпечуєтьсяпрограмою kvm. Ця програма працює з модулем kvm, використовуючи /dev/kvm для завантаження гостьової системи, підключає до неї віртуальний диск і потім завантажує її.

Контроль забезпечується набором ioctl-викликів, які працюють з пристроєм/dev/kvm. Коли файл пристрою відкривається вперше, створюється новий об'єкт віртуальної машини, який асоціюється з віртуальним процесором. Після цього можна використовувати ряд ioctl-викликів для створення віртуального процесора, перевірки версії kvm, створення області пам'яті і потім запуску віртуального процесора.

Переваги KVM

KVM є частиною стандартного ядра Linux, робота з гіпервізором багато в чомувизначається вмінням працювати з ядром операційної системи Linux. Оскільки модуль є частиною ядра, технологія отримує значні переваги від розвитку іоптимізації ядра.

Недоліки KVM

– Працює тільки в режимі повної віртуалізації, вимагає підтримки процесорамирежиму апаратної віртуалізації;
– Оскільки технологія є надбудовою над ОС, код гіпервізора є кодомвсієї операційної системи, а значить, має широкий профіль розломів інформаційної безпеки;
– Використання процесу QEMU в просторі користувача для забезпеченнявіртуалізації введення/виведення значно впливає на продуктивність підсистеми введення/виведення;
– Не має графічних інтерфейсів користувача під ОС Windows

2.3 ГіпервізорXen

ГіпервізорXen – багатоплатформовий гіпервізор, розроблений в Кембриджському університеті і розповсюджуваний на умовах ліцензії GPL. Основні особливості гіпервізора: підтримка режиму паравіртуалізації, апаратної віртуалізації, мінімалістичністькоду самого гіпервізора за рахунок виносу максимальної кількості компонентів за межі гіпервізора[6,7,8,9].

Xen починався як дослідницький проект Кембріджського університету підкерівництвом Яна Претта (IanPratt) і до сьогодні зберіг багато в чому свою академічність за багатьма аспектами: науковий підхід до розробки компоненту гіпервізора, публічність досліджень, залучення багатьох відомих дослідників в розробку.

Перший публічний реліз Xen був випущений в 2003 році. В даний час Xen переважно розвивається компанією Citrix і співтовариством LinuxFoundation.

Основною концепцією гіпервізораXen є домен (domain). Доменом називається запущена копія віртуальної машини. Якщо віртуальна машинаперезавантажується, то її домен термінується (в момент перезавантаження) і з'являється новий домен. Більш того, навіть при міграції віртуальної машини, вміст віртуальної машини копіюється з одного домену в інший домен. Таким чином, за час своєї життя практично всі віртуальні машини опиняються по черзі в різних доменах. ГіпервізорXen оперує виключно поняттям домену, а поняття «віртуальної машини»з'являється на рівні адміністрування (прикладних програм, які керують гіпервізором).

Домени бувають декількох типів, серед ключових доменів - dom0 і domU.

– dom0 - керуючий домен, запускається гіпервізором першим. Зазвичайстворюється і завантажується автоматично відразу після завантаження і ініціалізації гіпервізора. Цей домен має особливі права на управління гіпервізором і по замовчуванню все апаратне забезпечення комп'ютера доступно з dom0.Доменdom0 завжди один.

– domU -домен гостьовий операційної системи (скорочення від Userdomain),що містить в собі домен виконуються віртуальних машин. Зазвичай не маєдоступу до реального обладнання, використовує паравіртуальние або емулювані драйвери пристроїв. На відміну від dom0, доменів domU можебути безліч (зазвичай кілька десятків на один фізичний сервер).

– stub-domain - не обов'язковий для Xenдомен, в якому запускаєтьсяспеціалізована ОС, що забезпечує роботу з будь-яким обладнанням абобек-ендом драйвера. З'явився в зв'язку з розвитком моделі безпеки Xen.

– domainbuilder (конструктор доменів) - програма, яка створюєdomU(завантажує в нього потрібний код і повідомляє гіпервізору про необхідність запуску). Крім конструювання домену, зазвичай займається підключенням іконфігурацією віртуальних пристроїв, доступних для віртуальної машини. Вона ж відповідає за процес міграції віртуальної машини з хоста на хост.

Використання технологій паравіртуалізації - адаптації ядра виконуваною ОСдля роботи спільно з Xen - дозволяє досягти дуже високої продуктивності зарахунок відсутності емуляції апаратного забезпечення, простоти інтерфейсів та обліку існування гіпервізора при виконанні системних викликів в коді ядра. Виконання привілейованих операцій заборонено, замість них відбуваються гіпервиклики(Hypercalls) - звернення ядра гостьової ОС до гіпервізора з проханням про виконання тих чи інших операцій.

У більшості випадків зміни при портуванні ОС під Xen зачіпають тільки ядро ОС, хоча можуть припускати і незначні зміни в системних бібліотеках (наприклад, libc). Процес адаптації дляXen дуже схожий на портування для нової платформи, проте значно простіше зважаючи на простоту реалізації «гостьової» частини драйвера (драйвери в Xen складаються з двох частин - одна виконується поза віртуальної машини, друга знаходиться всередині неї. Частина драйвера в гостьовій системі вкрай примітивна і служить лише транслятором запитів до другої частини. Це зроблено навмисно для простоти портування ОС під Xen).

У паравіртуальном (PV-) режимі не підтримуються «вкладені» режими роботи процесора, такі як real-86, virtual-86, перемикання між 32-бітовим і 64-бітовим режимом, підтримка емуляції апаратної віртуалізації і т. д. У зв'язку з цим в PV-режимі відсутній початковий фрагмент завантаження комп'ютера (з імітацією коду BIOS, завантажувача і т. Д.), а ядро гостьової системи відразу ж запускається в потрібному режимі, подібно до того, як запускаються звичайні програми. У зв'язку з цим, зокрема, сам Xenне може працювати в PV-режимі (тобто неможливо запустити «вкладений» гіпервізорв PV-режимі).

У режимі апаратної віртуалізації (HVM) гостьова ОС не «знає» проіснування гіпервізора. Xen за допомогою модулів з QEMU емулює реальнеапаратне забезпечення і дозволяє провести початкове завантаження ОС. За еè закінчення для нормальної продуктивності повинні запускатися PV-драйвери, які реалізують швидкий інтерфейс з віртуальними пристроями, подібно до того, як це працює в PV-режимі). Оскільки більшість привілейованих операційемулюється,

можливий запуск Xen в HVM-режимі з-підXen. В цьому випадку вкладений гіпервізор зможе працювати тільки в PV-режимі.

Мінімізація коду гіпервізора

ГіпервізорXen реалізує мінімальний набір операцій для управління оперативною пам'яттю, станом процесора, таймерами реального часу ілічильниками тактів (TSC) процесора, переривань і контролем за DMA. Всі решта функції, такі як реалізація дискових і блокових пристроїв, створення та видалення віртуальних машин, їх міграція між серверами і т. д. реалізується в керуючому домені. За рахунок цього розмір гіпервізора виходить дуже малим (для версії 3.4 розмір двійкового коду всьогогіпервізора менше 600 КБ), так само як і розмір його вихідного коду. За задумом авторів це збільшує стійкість системи віртуалізації, так як помилка в компонентах поза гіпервізора не призводить до компрометації / пошкодження самого гіпервізора і обмежує пошкодження тільки компонентом, що вийшов з ладу, не заважаючи працювати іншим.

Всі функції, пов'язані із забезпеченням роботи мережі, блочних (дискових) пристроїв, емуляції відеоадаптерів та інших пристроїв винесені за межігіпервізора. Більшість таких пристроїв складається з двох частин: драйвери в domU і програми в dom0. Драйвер (найчастіше вбудований в ядро ОС або завантажується у вигляді модуля) реалізує мінімальний об'єм роботи, фактично, транслюючи запити відОС в програму в dom0. Програма в dom0 виконує основну частину роботи. При цьому програма найчастіше запускається у вигляді окремого процесу для кожного пристрою, який обслуговується. Збій в такій програмі веде до збою тільки одного пристрою (блочного, мережевого) і не зачіпає роботу інших копій програми (тобто не зачіпає мережеві / блокові пристрої інших доменів, або навіть іншіпристрої того ж самого домену).

Традиційно використовується наступна термінологія: фронтенд драйвера – частина модуля, що знаходиться в domU, бекенд драйвера - частина, яка перебуває в dom0. Для деяких типів пристроїв «задня» частина може бути різною при збереженні однієї і тієї ж «передньої» частини. Наприклад, драйвер блокового

пристрою може мати backendв формі програми роботи з VHD-образами, з блочними пристроями, з iscsi-ініціатором і т. д.

Міждоменна взаємодія. Xen надає доменам три механізми взаємодії: один – з гіпервізором (hypercalls), і два між доменами. Найчастіше, взаємодія відбувається між dom0 і domU, хоча модель допускає взаємодію і між двома domU.

Междоменна взаємодія зводиться до двох видів: events (події) isharedmem (загальний доступ до пам'яті). Третій варіант - передача сторінки пам'яті, є окремим випадком загального доступу до пам'яті.

Events служать приблизно для того ж, для чого служать переривання в архітектурі x86 або сигнали в Unix - швидка синхронна або асинхронна передача сигналу про настанні якоїсь події. Загальний доступ до пам'яті забезпечує можливість передачі значних об'ємов інформації, а події - швидкість передачі.

Події можуть бути замаскованими і незамаскованими. Незамаскованими подіївикликають callback (виклик функції, адреса якої переданий раніше) і дозволяють обробляти подію відразу ж за фактом його виникнення. масковані події лише встановлюють прапор про те, що подія відбулася, а обробник періодично дивиться, чи відбулося подія (одне або більше). Другий метод дозволяє не викликати callback по кожній події і в разі частих подій істотно знижує часобробки. Навпаки, перший варіант (з викликом callback) дозволяє збільшити швидкістьобробки події, яке, можливо, відбувається не дуже часто, але вимагає негайної реакції.

Міграція віртуальних машин. Xen (за допомогою набору керуючих компонент) підтримує міграцію гостьових віртуальних машин по мережі. Міграція паравіртуальних машин підтримується з версії Xen 2, а HVM - з версії 3. Міграція може відбуватися з вимиканням гостьової системи, або прямо в процесі роботи, так звана «жива» міграція (livemigration) без втрати доступності.

Для забезпечення можливості динамічної міграції віртуальних машин між фізичними серверами, необхідно, щоб сервериXenбули підключені до загального зовнішньому сховища, на якому знаходяться дані віртуальної машини. це потрібно тому, що при міграції віртуальної машини її файлова система не копіюється, так як

це вимагало б занадто багато часу навіть у випадку швидкої мережі. Загальне сховище може бути реалізовано на основі різних систем зберігання даних, наприклад FibreChannel, iSCSI або NFS.

Набір керуючих компонент (toolstack). У зв'язку з тим, що сам гіпервізор (близько 500-600 КБ) реалізує тільки «ядро» системи, весь інший функціонал виноситься на прикладний рівень, який працює в dom0. Набір програм, який реалізує функціональність за межами Xen називають toolstack, набором керуючих компонентів.

Існують дві найпоширеніші версії toolstack для Xen: заснована на xend (входитьв більшість дистрібутівів Xen) і заснована на xapi (входить до складу CitrixXenServeri XenCloudPlatform). Xend розвивався одночасного з Xen, написаний на Python і з самого початку йщов під відкритою ліцензією. Xapi був пробною розробкою Xensource (надалі Citrix), але в 2009 році був опублікований під ліцензією GPL. Xapi написаний на OCaml, забезпечує більший функціонал для роботи з віртуальними полігонами і забезпечують кращу стабільність роботи системи.

В обох версіях toolstack присутні наступні утиліти:

– xenstored - демон, який реалізує інтерфейс XenStore - проста деревоподібна база даних, що нагадує procfs. У xapi-toolstackXenStore переписаний на ocaml, алереалізує ту ж саму функціональність.

– xenconsoled - демон, що забезпечує в dom0 доступ до консолей віртуальнихмашин. Xenconsoled реалізує бекенд консольного пристрою для domU івикористовує API unix98 для створення псевдотерміналів в dom0. Відповідністьміж номером псевдотермінал і віртуальною машиною записується вXenStore.

Toolstack забезпечує управління віртуальними машинами (створення / видалення, запуск / зупинка, міграція, підключення ресурсів і т.д). Крім цього, інструментарій забезпечує управління ресурсами для великомасштабних систем: створює і підтримує репозиторії зберігання образів дисків віртуальних машин (SR - storagerepository), підтримує ресурсні пули серверів і може керувати складними

конфігураціями віртуальних мереж, в тому числі з підтримкою VLAN. Крім того, підтримується інтерфейс віддаленого управлінняXenApi на основі XML-RPC, який важливий для створення систем автоматизації управління віртуальними інфраструктурами.

Xen розроблявся науковим співтовариством, можливо, тому по Xen є великакількість літератури і наукових статей, як по архітектурі гіпервізора, так і попобудові віртуальних інфраструктур на базі гіпервізораXen.

На Xen побудовані найбільші хмарні інфраструктури в світі - Amazon, Rackspace, Lynode.

Переваги Xen:

– Якнайкраще забезпечення інформаційної безпеки серед гіпервізора;
– Стабільність роботи гіпервізора;
– Функціональні можливості Xen: можливість створювати ресурсні пули,проводити динамічну міграцію віртуальних машин, балансування навантаження між хостами, наявність графічного інтерфейсу управління під Windows;
– Наявність графічного інтерфейсу управління під Windows.

2.4 Мережева підсистема

Мережева підсистема повинна забезпечувати зв'язність віртуальних машин, як між собою, так і з зовнішнім світом, побудова довільних топологій мереж вінфраструктурі віртуальних полігонів. З точки зору прикладного програмногозабезпечення і зовнішніх пристроїв, віртуальні мережі повинні бути ідентичними фізичним мереж, підтримувати прозорість роботи використовуваних мережевих протоколів.

У ранніх реалізаціях віртуальних платформ можна було створювати віртуальнімережеві адаптери, сьогодні можлива віртуалізація більших мережевих компонент, наприклад, комутаторів, що підтримують комунікації між віртуальними

машинами, розташованими на одному сервері або розподіленими по декільком серверів [10].

У традиційній інфраструктурі, зображеної на мал. 4.3, підключення між серверами забезпечується за рахунок мережевих адапетров, підключених до зовнішньої мережевий інфраструктурі. Підключення серверів в локальну мережу реалізується комутатором (Switch), який відповідає за ефективний обмін пакетами між взаємодіючими мережевими інтерфейсами.

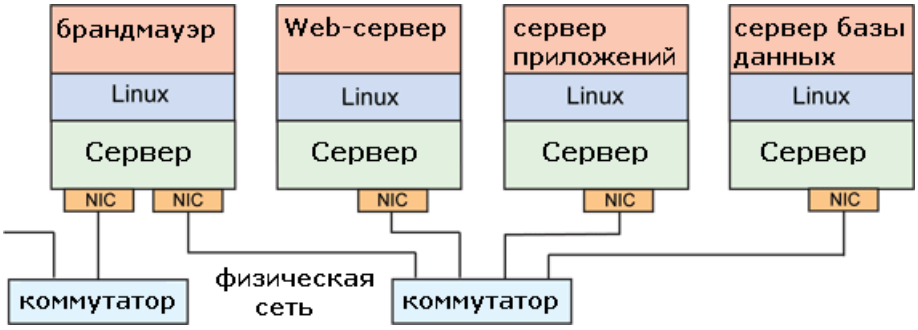


Рис. 2.3 Фізична мережева інфраструктура

Гіпервізор в інфраструктрі віртуальних полігонів може створити один абокілька віртуальних мережевих адаптерів для кожної віртуальної машини. Ціадаптери відображаються у віртуальній машині і для програмного забезпечення, встановленого на віртуальній машині, як фізичні, не дивлячись на те, що насправді вони тільки надають інтерфейс до реально існуючогоадаптера змінного струму.

Гіпервізор також дозволяє створити довільні віртуальні мережі з віртуальними комутаторами, щоб забезпечити підключення мережевих інтерфейсів віртуальних машини в ізольовані мережі, як це було б у випадку з фізичної комутацією. Тим самим забезпечується довільна топологія підключень на рівні L2. За рахунок підключення до віртуальної інфраструктурі фізичних адапетров сервера, стає можливим взаємодія між віртуальними машинами і зовнішнім світом.

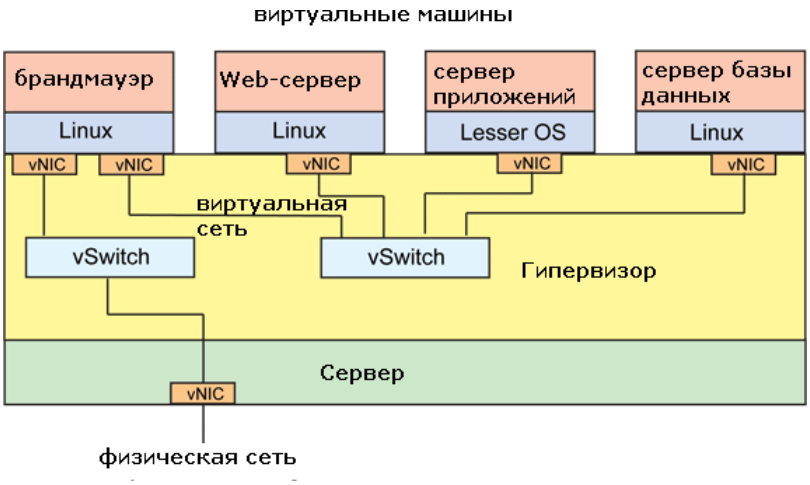


Рис. 2.4 Віртуалізована мережева інфраструктура

Іншим способом організації віртуальної комутації є використання готових шаблонів віртуальних машин (virtualappliance), що виконують функції традиційних мережевих пристроїв (комутаторів, маршрутизаторів).

Віртуальна комутація. Віртуальний комутатор - це ключовий компонент, необхідний для віртуалізації мережевої інфраструктури. Віртуальний комутатор з'єднує віртуальні мережеві адаптери з фізичними мережевими адаптерами, встановленими насервері, і, що більш важливо, пов'язує одні віртуальні мережеві адаптери з іншими для локального взаємодії в рамках сервера.

Для віртуального комутатора пропускна здатність визначається не швидкістю мережі, а можливостями оперативної пам'яті фізичного сервера. Тим самим навантаження на мережеву інфраструктуру в межах локального сервера зводиться до мінімуму, а ресурси фізичних мережевих пристроїв використовуються виключно для комунікації між серверами.

У ядрі Linux стандартно присутній функціонал створення комутатора другогорівня (bridge) [11] за допомогою утиліти brctl, який разом з іншими стандратнимимережевими інфструментамиLinux використовується в Xen для створення віртуальної мережі в межах одного фізичного сервера.

Для організації комутації між віртуальними машинами, розташованими на різних серверах, необхідно використовувати клас розподілених віртуальнихкомутаторів (distributedvirtualswitch). В цьому випадку, віртуальний

комутатор на одному сервері об'єднується з віртуальним комутатором на іншому сервері (див. мал.4.5).

Це значно спрощує управління мережами в масштабованої інфраструктурі,забезпечує довільну масштабованість інфраструктури віртуальних полігнов,тому в одну віртуальну мережу стає можливим об'єднувати довільну кількість віртуальних машин. Функціонал розподіленого комутатора значно полегшує міграцію віртуальних машин між серверами, так як переключення між інтерфейсами відбувається прозоро.

Так само, розподілений комутатор вирішують задачу контролю і моніторингутрафіку, тому що через ізоляції локального трафіку всередині сервера виникає проблема, пов'язана з тим, що цей трафік виявляється недоступний зовні (наприклад, для мережевих аналізаторів). У різних реалізаціях цю проблема вирішується за допомогою різних технологій, наприклад, OpenFlow, NetFlow і sFlow.

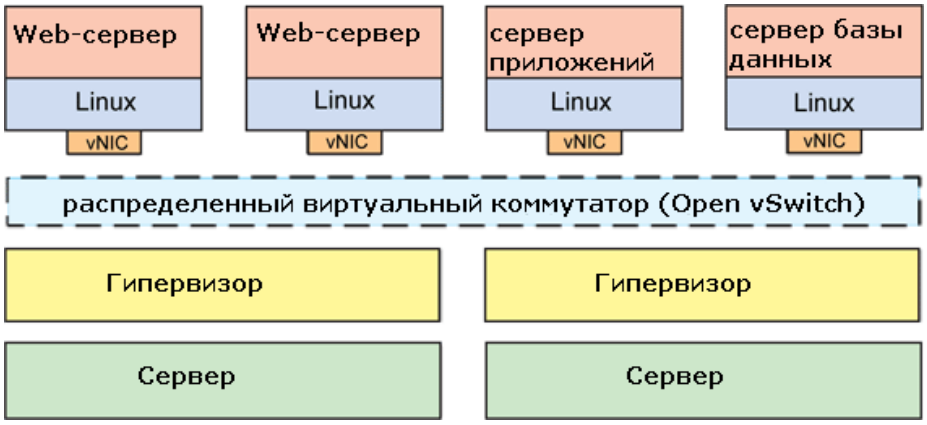


Рис. 2.5 Росподілений віртуальний комутатор

Один з найбільш поширених проектів в цій області - проект OpenvSwitch, який за замовчуванням використовується в гіпервізораXen (дистрибутив XenCloudPlatformiXenServer) при створенні об'єднаних віртуальних мереж міжфізичними серверами.

OpenvSwitch - це багаторівневий віртуальний комутатор з відкритим кодом,розповсюджуваний під ліцензією Apache 2.0 [12]. У загальному випадку,

OpenvSwitch підтримує всі найбільш популярні Гіпервізор з відкритим вихідним кодом, включаючи Kernel-based VM (KVM), VirtualBox, Xen. Також він розглядається в якості можливої заміни бридж-модуля (bridgemodule), використовуваного в даний момент в Linux. У дистрибутиві XenCloudPlatform модифікована версія OpenvSwitch за замовчуванням використовується для створення віртуальних мереж між фізичними серверами.

OpenvSwitch складається з демона-комутатора і супутнього модуля ядра, який керує процесом «потокової» (flowbased) комутації. Для керування комутацією існують і інші демони й утиліти (наприклад, перспективна реалізація від OpenFlow).

2.5 Віртуалізація мережевих пристроїв

Віртуалізація мережевого апаратного забезпечення в різних формах існувала ще до появи віртуальної комутації. У цьому розділі розглядаються кілька прикладів реалізації віртуальних мережевих пристроїв, а також прийоми апаратного прискорення, здатні підвищити швидкість віртуальних мереж.

QEMU

Хоча QEMU - це емулятор апаратних платформ, він також може програмно емулювати різні пристрої, включаючи мережеві адаптери. Також QEMU включає внутрішній DHCP-сервер (DynamicHostConfigurationProtocol - протокол динамічної конфігурації хоста) для призначення IP-адрес. ГіпервізорXeniспользует QEMU для емуляції пристроїв в процесі устновки операційної системи, до моменту завантаження паравіртуальних драйверів.

Virtio

virtio - це інфраструктура введення / виведення для Linux з підтримкою паравіртуалізації (para-virtualization), спрощує і прискорює обмін даними між віртуальною машиною і гіпервізором.

Технологія virtio створює стандартизований транспортний механізм для операцій введення / виводу між віртуальною машиною і гіпервізором, який можна

використовувати для віртуалізації блокових пристроїв, стандартних PCI-пристроїв, мережних пристроїв і т.д.

TAP і TUN

Віртуалізація досить давно реалізована в мережевому стеку, щоб забезпечитимережних інтерфейсів гостьовий віртуальної машини доступ до мережевих інтерфейсів фізичного сервера. Дві схеми подібного роду - це TAP і TUN.

TAP - це віртуальний мережевий драйвер ядра, який реалізує Ethernet-пристрій і працює на рівнів фреймів (пакетів) Ethernet. Драйвер TAP є "Перехоплювач" (англ. Tap) Ethernet-трафіку, який дозволяє Ethernet-кадри можуть містити гостьовий віртуальної машини брати участь в комунікаціях.

Технологія TUN (від networkTUNnel - мережевий тунель) емулює пристріймережевого рівня, спілкуватися на більш високому рівні IP-пакетів. це технологія застосовується для оптимізації, дозволяючи нижчого Ethernet-пристроїв управляти формуванням пакетів 2-го рівня з IP-пакетів TUN.

2.6. Віртуалізація введення-виведення

Віртуалізація введення-виведення забезпечується стандартною схемою від PCI SpecialInterestGroup (SIG), що дозволяє прискорити віртуалізацію на апаратному рівні. Технологія Single-root IOV (SR-IOV) надає інтерфейс, який робить один адаптер PCIe (PCI Express) доступною безлічі користувачів як кілька незалежних PCIe-адаптерів. Це дозволяє декільком драйверам незалежно і прозоро друг для друга підключатися до одного PCIe-адаптера. Технологія SR-IOV досягає такого ефекту, відкриваючи користувачам віртуальні функції, які відображаються як фізичні функції PCIe-адаптера, але насправді реалізовані в адаптері як функції для спільного використання.

Технологія SR-IOV підвищує продуктивність мережевий віртуалізації,позбавляючи гіпервізор від обов'язку організовувати спільне використанняфізичного адаптера і перекладаючи завдання реалізації

мультиплексування на сам адаптер. У цьому випадку забезпечується пряма пересилання введення / виведення з гостьової віртуальної машини безпосередньо на адаптер.

Платформа для віртуалізації Xen підтримує SR-IOV, використовуючи цютехнологію для надання віртуальних мережевих адаптерів гостьовим віртуальним машинам. Підтримка SR-IOV також варто в плані розвитку OpenvSwitch.

2.7 Віртуальні локальні мережі

Технологія віртуальних локальних мереж (virtual LAN - VLAN) широко використовується провайдерами мережевих сервісів, тому що дозволяє створювати віртуальні мережів рамках вже існуючої розподіленої мережі. За рахунок тегірування фреймів згідно стандарту IEEE 802.1Q, пристрої однієї віртуальної мережі стають повністю ізольовані від пристроїв інший віртуальної мережі. Ізоляція в таких віртуальних мережах здійснюється на програмному рівні, тому варто уважно розраховувати заплановане навантаження на мережу.

2.8 Апаратне прискорення віртуальних пристроїв

Технологія VT-d (VirtualizationTechnologyforDirected I / O – технологія віртуалізації спрямованого введення / виведення) від Intel® дозволяє ізолювати ресурси введення / виведення для підвищення надійності та безпеки, включаючи реорганізацію прямого доступу до пам'яті (на основі багаторівневих сторінкових таблиць) і переривань, пов'язаних з пристроєм. Ця технологія підтримується як для звичайних, так і для віртуальних пристроїв. Технологія Intel® VMDq (VirtualMachineDeviceQueues - черги для віртуальних пристроїв) також прискорює мережевий трафік при віртуалізації шляхом вбудовування черг і алгоритмів сортування безпосередньо в апаратне забезпечення. Завдяки цій технології гіпервізору для роботи потрібно менше

процесорних ресурсів, що призводить до значного зростання загальної продуктивності системи. В Xen є підтримка обох зазначених технологій.

2.9 Готові шаблони віртуальних машин

Готові шаблони віртуальних машин або virtualappliances виконують роль стандартних мережевих пристроїв, функіональний яких виненсена в операційну систему віртуальної машини. Такі віртуальні машини легко завантажуються гіпервізором і використовуються для виконання певних служб. Особливість даного підходу полягає в тому, що обчислювальні потужності (наприклад, процесорні ядра) і канали введення / виводу загального сервера можна динамічноконфігурувати для кожного віртуального пристрою, тим самим забезпечуючи більшу гнучкість таких віртуальних мережевих пристроїв, ніж стандартних «залізних» рішень.

Такий підхід робить віртуальні «коробкові» рішення вигідними з фінансової точки зору, так як для них не потрібно виділяти окремий сервер, а їх продуктивність можна динамічно змінювати в залежності від завантаження іншихдодатків, запущених на сервері.

Віртуальними пристроями також простіше управляти, так як додаток прив'язаний до операційної системи в рамках віртуальної машини, і вони не вимагають додаткового конфігурування, так як віртуальна машина спочаткуналаштована як єдине ціле.

Готові віртуальні рішення існують для багатьох корпоративних додатків, включаючи мережеві оптимізатори, маршрутизатори та віртуальні приватні мережі (VPN), брандмауери, системи пошуку і запобігання вторгнень, а також системи підтримки і класифікації електронної пошти. Крім мережевих служб, готові віртуальні пристрої існують для систем зберігання і захисту інформації,інфраструктур підтримки додатків і систем управління вмістом Web-сайтів.

2.10 Висновки

В даний час фізичні мережеві пристрої і служби можуть бути повністю замінені на віртуальні на рівні віртуальних машин в інфраструктурівіртуальних полігонів. Наявність готових віртуальних пристроїв підвищує гнучкість і керованість хмарних рішень, дозволяючи створювати довільні топології мереж.

У гіпервізораXen управління мережевою підсистемою багато в чому здійснюється вбудованими інструментами Linux, що дозволяє використовувати всю міць системи рішень для створення проівзольних конфігурації мережі віртуальних полігонів і розгортаються на них віртуальних машин. Додаткова функціональність забезпечується розподіленими віртуальними комутаторами і шаблонами віртуальних машин для необхідних мережевих служб.

3 СИСТЕМИ ЗБЕРІГАННЯ ДАНИХ

В якості системи зберігання даних для віртуальної інфрастркутури можуть використовуватися будь-які системи зберігання даних, що підтримують найбільш поширені протоколи передачі даних -iSCSI, NFS, FibreChannel (FC).

3.1 iSCSI

Протокол передачі даних iSCSI - протокол блочного рівня доступу, що працює поверх IP-протоколу. За рахунок цього можлива організаціясеті зберігання даних для інфраструктури віртуальних полігонів через мережеву інфраструктуру Ethernet, яка використовується для підключення обчислювальної підсистеми (гіпервізора).Протокол iSCSIпрошелсертіфікацію в IETF в кінці 2003 року (RFC3720) і на сьогоднішній день є широко поширеним і добре підтримуваним протоколом мереж зберігання даних. [13,14, 15,16]

За рахунок забезпечення доступу до дискової підсистеми на блочному рівні,iSCSI є функціональним еквівалентом відомого протоколу FibreChannel. Такожяк FC, технологія iSCSI дозволяє організовувати мережу зберігання даних, підключати безпосередньо до серверів або робочих станцій диски та інші пристрої зберігання (наприклад, стрічкові пристрої для бекапа) таким чином, що використовуватися вони будуть, як ніби вони підключені безпосередньо до цих пристроїв по локальним інтерфейсів SCSI.

Технічно це здійснюється шляхом інкапсулювання команд і блоків даних звичайного SCSI в IP-пакети. Інкапсульовані в IP пакети SCSI («SCSI-over-IP») можуть пересилатися по звичайній мережі Ethernet або навіть Інтернету. Потрапляючи до одержувача, вони витягуються з «обгортки» IP і надалі, з точки зору кінцевого користувача, це ті ж самі SCSI-пакети, немов вони пройшли не через Ethernet, а через звичайний SCSI-кабель.

Перевагою iSCSI є те, що він працює поверх IP протоколу, відповідно для транспорту може використовуватися наявну мережева інфраструктура. Для

інфраструктури віртуальних полігонів може використовуватися досить розповсюджених GigabitEthernet, 10 GigabitEthernetілі навіть 100 MbitEthernet для невисоконагруженних або тестовихзадач. За рахунок можливості об'єднати канали в «транки», можна пропорційно збільшувати сукупну швидкість підключення. Широкая доступність Ethernet-інфраструктури означає в тому числі і її дешевизну в практичній реалізації.

Переваги:

- Вартість, можливість використовувати наявну мережеву інфраструктуру.
- Програмний модуль для використання iSCSI є для більшості
- існуючих сьогодні на ринку OS (MSWindows, Solaris, Linux, AIX і т.д).
- Простота настройки і використання.

Оскільки носієм iSCSI є IP-протокол, можливе створення систем зберігання даних, рознесених на великі відстані і знаходяться в різних датацентрах, підключених один з одним через загальні канали зв'язку Інтернет.

Недоліки:

- Оскільки iSCSIіспользуетIP-протокол і спочатку IP не створювалася для цілеймасованої передачі даних з низькою латентністю і гарантованоїдоставкою, iSCSI чутливий до перезгузке мережі і сам сильно завантажує мережу примасованої передачі даних
- Через це, все ж потрібна побудова ізлоірованной IP-мережі для мережі зберіганняданих iSCSI.
- iSCSI може використовувати програмний модуль, який дозволяє здійснюєрозглянуту вище інкапсуляцію і декапсуляцію SCSI в IP (англ. - initiator).Однак, як будь-яке програмне рішення, це споживає певну кількістьпроцесорної потужності. У реальному житті, на доступних сьогодні процесорах,ця величина прагне до одиниць відсотків.

Існують і «апаратні» реалізації у вигляді iSCSIадаптеров(HBAQLogicQLA4050), що знімають ці проблеми, однак це може істотно збільшувати бюджет проекту.

Максимальна пропускна здатність протоколаiSCSIбез використання «транков»(ОБ'ЄДНАННЯ разом декількох портів для збільшення пропускної спроможності) зараз дорівнює 10Gb / s. Швидкість FC для найбільш поширених FC-пристроїв дорівнює 4 і 8Gb/s, що фактично можна порівняти за характеристиками.

Для побудови системи зберігання даних iSCSI необхідна наявність дисковогомасиву з підтримкою інтерфейсу iSCSI (або налаштованої серверної системи ziSCSIсерверним модулем - таргетом), окремого сегмента мережі передачі даних, вільних Ethernet портів в серверах або спеціального iSCSI адаптера, а так само програмний компонент «Initiator» під ОС серверів.

3.2 FibreChannel

Протокол FibreChannel або FC стандартизований технічним комітетом T11, Міжнародним комітетом стандартизації інформаційних технологій (INCITS), атакож комітетом стандартизації ANSI [17].

Спочатку FibreChannel використовувався для підключення суперкомп'ютерів, але нині набув поширення в промислових масштабах в мережах зберіганняданих (СГД). Сегменти мереж зберігання даних, побудованих на FC, прийнято називати фабриками.

FibreChannelProtocol (FC) являє собою транспортний протокол (можна порівняти з TCP, використовуваним в IP мережах), і використовується в основному для передачі SCSI команд в СГД. Також здійснена підтримка передачі протоколів ATM і IP мережами FC.

FibreChannel працює на швидкостях 1, 2, 4, 8, 16Gb/s. В даний час ведетьсярозробка стандарту передачі даних FibreChannel на швидкостях до 32Gb/s. СетьFibreChannel може містити спеціалізовані комутатори, шлюзи і маршрутизатори, мережі FibreChannel можуть використовувати як оптико-волоконні, так і мідні кабелі.

FibreChannel, так само як і iSCSI забезпечує передачу даних на блочному рівні, що дозволяє системам визначати віддалені пристрої, як пристрої з прямим підключенням. Основною перевагою використання прямих підключень, є можливість використання стандартних програм (таких як, ПО резервного копіювання, управління дисковими томами і ін.)

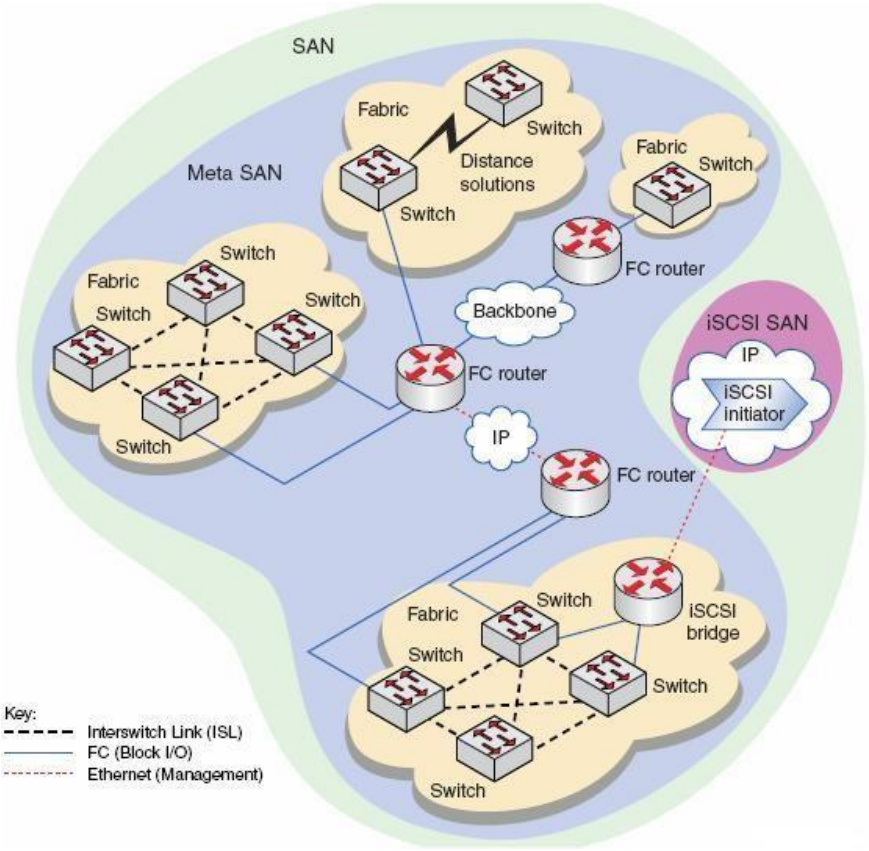


Рис. 3.1 Приклад реалізації мережі зберігання даних FibreChannel

Як показано на малюнку 4.6, мережі FibreChannel можуть складатися з декількох окремих сегментів, об'єднаних між собою маршрутизаторами. Також, мережі FibreChannel можна розбити на фабрики (fabrics) і мета SAN (Meta SAN). Залежно від того, як здійснюється міжмережний маршрутизація, мережею мета SAN може також вважатися мережу ISAN.

Завдяки високій швидкості передачі даних, малій затримці і дуже високою масштабованості практично не має аналогів в цій області. основним недолікомвикористання протоколу FCявляется ціна таких систем зберігання даних і складністьнастройки рішення.

В останні роки, область його застосування поступово переміщається в сегментвисокопродуктивних систем і рішень, а бюджетний сегмент з успіхом освоюєтьсянедорогими рішеннями iSCSI на базі GigabitEthernet і 10G Ethernet. намітилася такожтенденція до перенесення транспортного рівня протоколу FC в той же Gigabit і 10G Ethernetза допомогою протоколів FCoE і FCIP [19].

3.3 NFS

NetworkFileSystem (NFS) - протокол мережевого доступу до файлових систем,спочатку розроблений SunMicrosystems в 1984 році. Заснований на протоколі викликуудалѐнних процедур (ONCRPC, OpenNetworkComputingRemoteProcedureCall, RFC1057, RFC 1831). Дозволяє підключати (монтувати) удалѐнние файлові системи через мережу, описаний в RFC 1094, RFC 1813, RFC 3530 і RFC 5661 [20, 21].

NFS абстраговані від типів файлових систем як сервера, так і клієнта,існує безліч реалізацій NFS-серверів і клієнтів длярізних операційних систем і апаратних архітектур. В даний часвикористовується найбільш зріла версія NFSv.4 (RFC 3010, RFC 3530), підтримуючарізні средстваутентіфікаціі (зокрема, Kerberos і LIPKEY з використаннямпротоколу RPCSECGSS) і списків контролю доступу (як POSIX, так і Windows-типів).

NFS надає клієнтам прозорий доступ до файлів і файлової системисервера. На відміну від FTP, протокол NFS здійснює доступ тільки до тих частин файлу,до яких звернувся процес, і основна перевага його в тому, що він робить цейдоступ прозорим. Це означає, що будь-який додаток клієнта, яке можепрацювати з локальним файлом, з таким же успіхом може працювати і з NFS файлом, безбудь-яких модифікацій самої програми.

NFS клієнти отримують доступ до файлів на NFS сервері шляхом відправки RPCзапросов на сервер. Це може бути реалізовано з використанням звичайнихпризначених для користувача процесів - а саме, NFS клієнт може бути

призначеним для користувачапроцесом, який здійснює конкретні RPC виклики на сервер, який так самоможе бути призначеним для користувача процесом.

Важливою частиною останньої версії стандарту NFS (v4.1) сталаспецифікація pNFS, націлена на забезпечення розпаралеленого реалізації загальногодоступу до файлів, що збільшує швидкість передачі даних пропорційно розмірамі ступеня паралелізму системи.

Переваги іспользованіясістеми зберігання даних, побудованої на NFS:

– Широка поширеність і протота настройки. БільшістьLinuxопераціонних систем може функціонувати як NFS-сервер;

– Файлова система вже має вбудовані функції, необхідні для коректноїроботи обчислювальної підсистеми, такі як множинний доступ хостівірозподілене блокування доступу до файлів

– NFS працює поверх стандартної мережі передачі даних Ethernet, використовуючи IPпротокол

– Віртуальні машини, шаблони віртуальних машин, віртуальні жорсткі дискиявляють собою файли. Оскільки NFSпротокол оптимізований під роботу зі звичайними файлами, він добре сумісний з усіма системами віртуалізації.

Недоліки:

Проблеми з продуктивністю. Оскільки реалізація протоколу програмна,файлу потрібне обчислювальні потужності для обробки запити клієнта. Протягом часу, поки сервер обробляє запит клієнта, сервер не блокуєзапити від інших клієнтів, які також повинні бути обслужені. Щобвпоратися з подібною ситуацією, більшість NFS-серверів запускаютьсякілька разів, тобто всередині ядра існує кілька NFS-серверів. Цестворює додаткові накладні витрати, але проблема вирішується, в тому числізбільшенням обчислювальної потужності сервера. Так само, методи вирішенняпроблеми з продуктивністю залежать від операційної системи. Вбільшості ядерUnix-систем запускається кілька користувальницьких процесів(Які зазвичай називаються nfsd), які здійснюють один системний викликі залишаються всередині ядра в якості процесу ядра. Так само існують

комерційні реалізації систем зберігання даних, оптимізовані під високу продуктивність протоколу.

3.4 Програмне забезпечення для автоматизації управління хмарними інфраструктурами

Використання самих гіпервізорів в інфраструкутрі віртуальних полігонів не забезпечує автоматизації управління інфраструктурою, надає недостатньо коштів для універсального доступу до системи. Для цілей зовнішнього управління, всі відомі Гіпервізор мають відкриття інтерфейси управління (API), доступ до яких можна отримати за допомогою зовнішніх протоколів, таких як xml-rpc, ssh та інших.

Цим самим фактично забезпечується необмежений доступ до функціональних можливостей гіпервізора без необхідності модифікувати його код. Винятки тут - пропрієтарні гіпервізори, які хоч і забезпечують можливість зовнішнього управління, але вони зазвичай обов'язково вимагають наявності свого керуючого центру, до того ж функціонал їх залишається сильно обмежений без покупки ліцензій.

Все це є основною для створення проміжного програмного забезпечення, серверів додатків, які беруть на себе задачу автоматизації управління інфраструктурою віртуальних полігонів, використовуючи вбудовані функціональні можливості гіпервізора і надані ними інтерфейси управління.

В даний час існує ряд готових програмних продуктів, покликаних вирішити задачу централізованого управління інфраструктурою. Серед програмного забезпечення з відкритим кодом можна виділити OpenNebula, OpenStack, CloudStack, Eucalyptus, серед пропрієтарного програмного забезпечення лідером є набір програмних продуктів VMware, серед яких VMware vSphere, vCloudDirector, ChargeBack і інші.

Предметом розгляду є продукти з відкритим вихідним кодом, в якості основи для аналізу було обрано програмне забезпечення OpenNebula [22] і

OpenStack [23] Дана програмне забезпечення є найбільш яскравим прикладом розглянутих платформ, маючи як схожі архітектурні особливості, так і відмінності, що, безумовно, представляє інтерес для вивчення.

Програмне забезпечення було окремо встановлено і протестовано на обладнанні в такій конфігурації: використовувався модульний сервер DEPO Storm 5302A1 з чотирма обчислювальними модулями DEPO Storm 3300JZ. Кожен обчислювальний модуль компонувався двома шестиядерними процесорами серії Intel Xeon 5600, що підтримують режим апаратної віртуалізації Intel VT, 24 Гб оперативної пам'яті, 3 жорсткими дисками (3.5 дюйма), двоканальним Gigabit Ethernet контролером Intel® 82574L [24].

Перший обчислювальний модуль використовувався в якості керуючого сервера OpenNebula, а так само в якості системи зберігання даних для образів, шаблонів, налаштувань віртуальний машин (в якості системи зберігання даних використовувався запущений nfs-сервер). Використовувана ОС - Ubuntu Server 10 [25], дистрибутив керуючого ПО - OpenNebula 3.6 (далі - OpenNebula).

Другий і третій обчислювальний модулі використовувалися в якості робочих серверів (вузлів) для запуску віртуальних машин під керуванням ПО OpenNebula, на них були встановлені і налаштовані Гіпервізор.

Четвертий модуль, що працює під управлінням ОС Ubuntu Server 12.04 LTS, був використаний для тестування ПО OpenStack Essex (2012.1) (далі - OpenStack). Він поєднував у собі ПО управління і робочий вузол віртуалізації, тому що дана конфігурація для ПО OpenStack є допустимою в тестових середовищах і інсталюється базовим інсталяційний скриптом.

В системі OpenNebula додаткове ПО на робочі вузли не встановлювалося, управління гіпервізором здійснювалося за допомогою бібліотеки libvirt [26], що входить до складу стандартних дистрибутивів Linux, за рахунок встановлення беспарольному rsa-ssh сесії між керуючим сервером OpenNebula і робочими вузлами.

Зведені результати дослідження представлені в розділі «Порівняння платформ OpenNebula і OpenStack».

3.5 Порівняння платформ OpenNebula і OpenStack

3.5.1 Загальні аспекти

	Історія проекту	**Розмір і активність спільноти**	**Примітка**
OpenStack	OpenStack спочатку проект хостингової компанії RackSpace і NASA, які відкрили частину вихідних кодів. Від NASA - обчислювальна частина Nova, від RackSpace - об'єктне сховище Swift. Критика OpenStack зазвичай стосується (1) непрозорість проекту, (2) спочатку складна структура і погана логічна зв'язність входять до складу проекту компонент, низька якість коду, складність многоузловой установки.	Спільнота швидко зростаюче. Розробку офіційно підтримують багато міжнародних компаній виробники обчислювального обладнання і ПЗ, але чіткого розмежування відповідальності і внесок компаній в розробку не зрозумілий.	Звернень в технічну підтримку не було, установка ПЗ в простій одноузловой конфігурації пройшла без інцидентів.
OpenNebula	Дослідницький проект,	Підтримка на рівні	При зверненні в

	що стартував в 2005 році, з 2008 року починається період активного росту і просування проекту, відкриті вихідні коди. Чітка логічна структура ПЗ. Стабільні і заплановані релізи, продумана міграція на нові версії ПЗ, прозорість встановлення та налаштування.	спільноти у OpenNebula на поточний момент більше ніж, підтримка у OpenStack. Міжнародних компаній, офіційно заявляють про підтримку проекту, істотно менше, ніж OpenStack, але проект впроваджений і використовується в багатьох європейських науково-дослідних центрах	технічну підтримку відповідають безпосередньо розробники ПЗ. Відповіді точні, оперативні, час реакції зазвичай не більше 24 годин.

3.5.2 Архітектурні аспекти

ПЗ	**Порівняння**

OpenStack	OpenStack - комплекс проектів в рамках єдиної платформи, складається з декількох різних за призначенням логічних частин: Nova - обчислювальний модуль, мережевий сервіс, контролер обчислювальних ресурсів, Swift - об'єктне сховище, Glance - сервіс управління образами віртуальних машин, Keystone - сервіс ідентифікації, Horizon - web-портал управління. Кожен модуль вимагає установки і налаштування спеціальних компонент і необхідних для їх роботи системних пакетів ОС. Комунікації між компонентами здійснюються за протоколом AMPQ через виділений брокер сполук - обчислювальний контролер. Для забезпечення роботи обчислювального модуля, необхідно на кожному робочому сервері запускати мережеві та обчислювальні агенти, які взаємодіють з керуючим контролером. Системні налаштування зберігаються в SQL базі даних (MySQL, PostgreSQL). Система може бути легко децентралізована за рахунок рознесення сервісів по різним фізичним серверам. **1) Реалізація обчислювальної підсистеми, взаємодія з гіпервізорами** Обчислювальний модуль nova-compute встановлюється на кожен робочий сервер, управляє роботою гіпервізора і віртуальних машин за допомогою локального виконання системних команд, підтримуваних гіпервізором. Обчислювальний модуль взаємодіє з обчислювальним контролером (nova-api), сервісом аутентифікації (keystone), мережевим сервісом (nova-network), диспетчером завдань (nova- scheduler) і іншими сервісами OpenStack. Управління блоковими пристроїв і їх підключенням до віртуальних машин в релізі Essex так само здійснюється даним модулем (пакет nova-volume). Реалізація блокових пристроїв здійснюється за допомогою

	функціональності системного ПЗ Linux LVM, або підключенням зовнішніх блокових систем зберігання даних iSCSI. **2) Реалізація підсистеми зберігання даних для централізованого зберігання і управління образами віртуальних машин, шаблонами налаштувань віртуальних машин** Управління зберіганням і використанням образів віртуальних машин здійснюється модулем glance, який за замовчуванням працює в зв'язці з об'єктним сховищем Swift. Як сховище для образів, крім Swift, можна використовувати звичайну файлову систему. Об'єктне сховище Swift (Object Store) дозволяє перетворити сервери в масштабується сховище даних з вбудованими функціями щодо забезпечення відмовостійкості. Система автоматично робить кілька надлишкових реплік даних між серверами і в разі збою одного з серверів, цілісність даних не порушується. Чи не є файловою системою і погано працює з OLTP даними, призначене для довгострокового зберігання великих об'єктів (образи віртуальних машин, мультимедіа-контент), аналог сервісу Amazon S3 [27]. **3) Реалізація міграції віртуальних машин між вузлами кластера без зупинки запущених сервісів** Міграція працюють віртуальних машин здійснюється виключно за допомогою функціональних можливостей гіпервізора, міграція можлива лише між вузлами до встановлених гіпервізорами одного типу. **4) Реалізація мережевої підсистеми** Мережеві налаштування реалізовані за допомогою використання вбудованих механізмів ОС Linux з управління мережею - створення мостів, vLAN-ів. Управляє мережею модуль nova-network. **5) Реалізація графічного інтерфейсу користувачів** Графічний інтерфейс користувача і адміністратора реалізований за допомогою модульного web-сервера, написаного на мові Python з

	використанням фреймворку Django. Є графічним інтерфейсом до всіх основних сервісів OpenStack.
OpenNebula	Архітектуру ПО OpenNebula можна представити у вигляді трьох основних шарів: функціональних драйверів, монолітного ядра системи і спеціалізованих утиліт. Драйвери відповідальні за роботу з системними компонентами і ОС - гіпервізорами, файловими і мережевими сервісами, віртуальними машинами, легко налаштовуються адміністратором під свої функціональні завдання. Ядро (системні процеси на керуючому сервері) управляє всіма компонентами системи – віртуальними машинами, системами зберігання даних, віртуальними мережами, здійснює балансування навантаження і диспетчеризацію запитів і команд. Утиліти забезпечують додатковий функціонал в роботі системи. **1) Реалізація обчислювальної підсистеми, взаємодія з гіпервізорами** Робота з гіпервізорами здійснюється за допомогою встановлення беспарольному rsa-ssh сесії з робочими вузлами, установка агентів не потрібно. **2) Реалізація підсистеми зберігання даних для централізованого**

	зберігання та управління образами віртуальних машин, шаблонами налаштувань віртуальних машин Підтримуються всі види сторонніх систем зберігання даних (СЗД) - блокові, файлові, локальні, розподілені, додаткових обмежень на тип системи не накладається. За взаємодію відповідають драйвери, які налаштовуються під використовуваний тип СГД. Відсутні сервіс автоматичного управління підключаються до віртуальних машин блокових пристроїв, необхідно налаштовувати вручну. Немає власної системи зберігання даних. **3) Реалізація міграції віртуальних машин між вузлами кластера без зупинки запущених сервісів** Міграція працюють віртуальних машин здійснюється виключно за допомогою функціональних можливостей гіпервізора, міграція можлива лише між вузлами до встановлених гіпервізорами одного типу. **4) Реалізація мережевої підсистеми** Мережеві налаштування реалізовані за допомогою використання вбудованих механізмів ОС Linux з управління мережею - створення мостів, vLAN-ів. Можливе використання розподіленого віртуального комутатора. **5) Реалізація графічного інтерфейсу користувачів** Графічний інтерфейс користувача і адміністратора реалізований за допомогою модульного web-сервера, написаного на мові Ruby з використанням фреймворку Sinatra.

3.5.3 Установка системи

	Керуючий сервер	**Робочі сервери (вузли)**	**Примітки**
OpenStack	Використовується офіційний дистрибутив Ubuntu Server, запускався автоматичний інсталяційний скрипт. Установка в одноузловой конфігурації, що поєднує роль керуючого сервера і робочих серверів.		Для простих тестових конфігурацій, установка проста, виконується без інцидентів і з мінімальною участю адміністратора. Для виробничої установки (многоузловой) інсталяція істотно складніше. Відсутня алгоритм централізованого перезапуску системи, необхідно перезапускати всі встановлені компоненти окремо, в т.ч. на робочих вузлах.
OpenNebula	Установка з вихідних по документації, тому що автоматична установка пакета зі сховищ проходить некоректно. Первинне налаштування	Установка ПЗ на робочі сервери не потрібно, потрібно налаштування адміністративного аккаунта і групи, підключення поділюваних	Частина залежностей автоматично можуть не вирішуватися, доводиться доустанавливать вручну. Немає автоматизованого скрипта повної

	системи: генерація ключів для безпарольного доступу до робочих вузлів, настройка адміністративного аккаунта і групи, настройка NFS сервера, настройка мережевих служб	файлових директорій для системи зберігання даних NFS, налаштування мережевих служб	установки. Гарна документація, в якій зафіксовані всі основні складності при установці і базовому налаштуванні. Системні сервіси централізованого зупинки і запуску керуючого сервера.

Висновки

Обидві розглянуті системи, не дивлячись на заяви спільнот-розробників, поки ще не є закінченими решеніяміпо автоматизації та централізованого управління хмарними інфраструктурами. У тому числі вони не можуть бути без доробок використані для побудови інфраструктури віртуальних полігонів.

З урахуванням наявності відкритих інтерфейсів управління у гіперізоров, доцільно реалізовувати функціонал автоматизації управління у вигляді окремої підсистеми, сервера додатків, написаного і оптімізованого під конкретне завдання з урахуванням специфіки роботи організації, яка буде використовувати інфраструктуру і вимог до бізнес-процесів.

4 АРХІТЕКТУРА ІНФРАСТРУКТУРИ ВІРТУАЛЬНИХ ПОЛІГОНІВ

4.1 Загальна архітектура рішення

Для вирішення поставлених в роботі завдань вироблена наступна архітектура інфраструктури віртуальних полігонів, зображена на мал. 5.1.

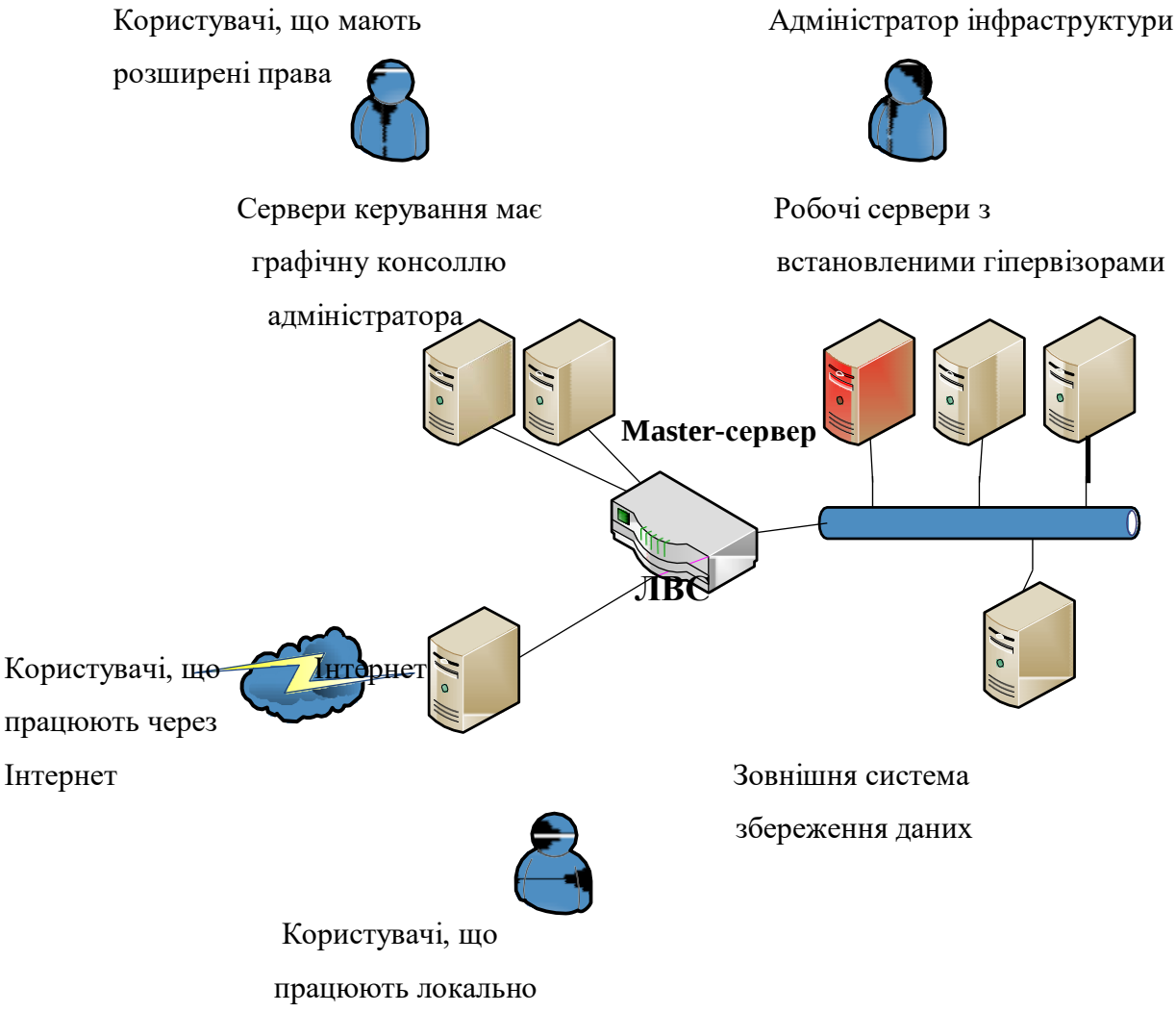


Рис. 4.1.Загальна архітектура системи

Робочі сервери віртуалізації з встановленим гіпервізором є обчислювальним ядром інфраструктури, що забезпечує створення і управління роботою віртуальних машин, диспетчеризацію розподілу ресурсів між віртуальними машинами. Робочі сервери об'єднуються в ресурсний пул для забезпечення еластичності інфраструктури, можливості масштабувати інфраструктуру прозоро для користувачів.

У ресурсному пулі виділяється керуючий, master-сервер, за допомогою якого здійснюється централізоване адміністрування як всією обчислювальною системою, так і управління системою зберігання даних і мережевий підсистемою. Так само, master-сервер, є центральним інтерфейсом управління всією системою за допомогою наданого їм API.

Обчислювальна підсистема забезпечує наступну функціональність:

– забезпечення продуктивності всіх компонентів віртуальних машин на рівні продуктивності пристроїв фізичних серверів;
– можливість створення готових шаблонів віртуальних машин з попередніми налаштованими пакетами програмного забезпечення;
– об'єднання фізичних серверів в ресурсні пули для динамічного розподілу віртуальних машин між фізичними серверами;
– можливість динамічної міграції віртуальних машин між серверами;
– підтримка режиму паравіртуалізації для найбільш ефективного використання обчислювальних та інших ресурсів операційними системами сімейства Linux.

Оскільки обчислювальна система є ядром всієї інфраструктури віртуальних полігонів, деякі вимоги до інших підсистем продиктовані функціональними можливостями обраного рішення віртуалізації.

Додавання нових серверів, а так же штатний або аварійне вимкнення наявних серверів, відбувається прозоро для адміністратора системи і головне - для користувачів, не вимагає перенастроювання системи і будь-яких додаткових дій з боку людини, яка не призводить до втрати інформації і збоїв в роботі

інфраструктури. При даних діях користувач або взагалі не помічає змін, що відбулися, або перерви в роботі сервісів мінімальні.

Така можливість є суттєвою з точки зору хмарної моделі надання послуг, тому що забезпечує об'єднання ресурсів в ресурсні пули для динамічного перерозподілу потужностей між користувачами в умовах постійної зміни попиту на потужності, функції динамічної міграції віртуальних машин при проведенні технічного обслуговування окремих серверів, балансування навантаження між робочими серверами.

В архітектурі системи дана вимога виражається в необхідності використання зовнішнього сховища даних для зберігання файлів віртуальних машин, шаблонів віртуальних машин, віртуальних жорстких дисків користувачів. Гіпервізор повинен підтримувати роботу не тільки з локальною системою зберігання даних робочого сервера, але і з зовнішніми системами зберігання даних. Проектування підсистеми зберігання даних докладно розглянуто в розділі 5.4 Підсистема зберігання даних.

Мережева підсистема забезпечує створення довільних топологій мережевої інфраструктури віртуальних машин, мережеву зв'язаність віртуальних машин як в межах внутрішніх віртуальних мереж між віртуальними машинами, так і по відношенню до зовнішніх до системи мереж і пристроїв, нормальне функціонування віртуальних машин з точки зору мережевих протоколів. Віртуальні машини, розгорнуті на різних фізичних серверах, мають можливість бути об'єднаними в єдину віртуальний домен комутації на рівні L2.

Для забезпечення вимоги до високої еластичності інфраструктури при реалізації мережевої підсистеми максимально використовуються програмні мережеві рішення: вбудовані можливості гіпервізора і стандартних мережевих засобів операційної системи Linux, з апаратних рішень використовується тільки фізичний комутатор рівня L2 для підключення серверів до локальної обчислювальної мережі.

Проектування мережевої підсистеми детально розглянуто в розділі 5.3 - мережива підсистема.

Технічне управління і адміністрування системою в режимі нормальної експлуатації здійснюється з окремої програми з графічним інтерфейсом користувача, встановленої на керуючому сервері адміністратора системи.

Адміністраторів системи може бути будь-яка кількість, так само можливо наділяти користувачів розширеними правами на управління частинами віртуальної інфраструктури. У цій програмі існує можливість створювати всі типові настройки, шаблони віртуальних машин, управляти життєвим циклом віртуальних машин, здійснювати адміністрування віртуальних машин, в тому числі в режимі консолі. Конфігурація робочих серверів віртуалізації так само має здійснюватися підключенням до них безпосередньо по SSH через інтерфейс командного рядка (CLI).

Звичайні користувачі отримують параметри доступу до віртуальних машин, а так доступ до консолі віртуальної машини через веб-сервер (портал) управління. Портал доступний як всередині локальної мережі організації, так і через мережу Інтернет.

На веб-поратле реалізована наступна функціональність:

– Користувач має можливість бачити настройки і опис наданої йому віртуальної машини (групи віртуальних машин), параметри доступу до віртуальної машини (машин);
– Користувач має можливість запускати, перезапускати або зупиняти віртуальні машини з наданого йому набору віртуальних машин, підключатися до віртуальної машини в будь-якому її стані через графічну консоль
– Адміністратор системи забезпечує активацію віртуальних машин з наявних шаблонів для заданих користувачів, при необхідності забезпечує вимикання і видалення віртуальних машин;

– Так само бажаною функціональністю є можливість для кожного користувача мати власний віртуальний жорсткий диск, який можна підключати до будь-якої віртуальної машині і відключати його при припиненні роботи з віртуальною машиною. Дані на віртуальному диску повинні зберігається постійно, поки адіністратор не вилучено цей диск.

Веб-сервер написаний на мові програмування Java і релизована на стандартній платформі, контйнере сервлетів, що підтримує функціонал віддалених процедур XML-RPC. Необхідний фунцонал веб-сервера забезпечується APImaster-сервера.

Далі викладено дослідження підсистем інфраструктури віртуальних полігонів і їх взаємодію для того щоб прийти до найбільш оптимальної схеми побудови всієї системи.

4.2 Обчислювальна підсистема

Виходячи з проведеного в частині 4 огляду, проаналізувавши всі переваги і недоліки найбільш поширених гіпервізора з відкритим вихідним кодом, а саме KVM і Xen, як ядра обчислювальної підсистеми використовується гіперівзор Xen, що встановлюється на робочі сервери віртуалізації (далі - робочі сервери).

Вибір Xen обумовлений такими факторами:

– Стабільність роботи гіпервізора і його висока продуктивність;
– Наявність вбудованих функціональних можливостей, що спрощують створення інфраструктури віртуальних полігонів;
– Наявність зовнішніх програмних бібліотек, що спрощують роботу з API гіпервізора;
– Наявність великої кількості літератури з дослідженням різних аспектів роботи гіпервізора (посилання).

Не дивлячись на те, що гіпервізор Xen є відкритим і вільним програмним забезпеченням, необхідності в доопрацюванні самого ядра гіпервізора в рамках роботи не виникло, тому що функціональні можливості гіпервізора і сполученого набору керуючих компонент є задовольняють вимогам до обчислювальної підсистемі розглянутого рішення, і достатніми для коректної роботи з іншими підсистемами.

4.3 Можливі варіанти установки гіпервізора на робочі сервери

Існують різні варіанти установки робочих серверів віртуалізації Xen: з готових дистрибутивів у вигляді ISO-образів, установкою пакетів із загальних репозиторіїв за допомогою вбудованих в попередньо встановлені операційні системи менеджерів пакетів, а так само компіляцією з офіційно опублікованих вихідних кодів. У всіх варіантах проводиться установка самого гіпервізора Xen, основні відмінності криються в керуючої операційній системі (dom0), використовуваних наборах керуючих компонент (toolstacks), які можуть встановлюватися окремо, наявності попередньо налаштованих шаблонів віртуальних машин і додаткової функціональності до Гіпервізор.

Більшість дистрибутивів Linux і Unix містять в своїх репозиторіях вже зібрані пакети Xen, які можуть бути завантажені і встановлені за допомогою вбудованих в операційну систему менеджерів пакетів. В цьому випадку настройка робочого сервера віртуалізації зводиться до таких кроків: 1) установка операційної системи, 2) установка пакетів з Xen і додаткових модулів керуючих компонентів, 3) модифікація параметрів ОС і параметрів завантаження ОС, 4) перезавантаження операційної системи.

Після перезавантаження, операційна система стає керуючим доменом (dom0) для Xen гіпервізора.

Установка гіпервізора Xen з вихідних кодів має сенс, коли для необхідної операційної системи немає готового бінарного дистрибутива.

В інших випадках доцільно використовувати добре скомпоновані готові дистрибутиви з гіпервізором Xen.

Готовими збірками є повністю відкритий і вільно поширюваний дистрибутив Xen Cloud Platform (остання версія на момент написання роботи -1.6) і комерційна гілка дистрибутива Xen Cloud Platform - продукт компанії Citrix XenServer, розповсюджуваний за ліцензією Citrix EULA.

Відмінності між збірками Xen Cloud Platfrom і XenServer відображені в таблиці №5.1 [28].

Функціональність	**Xen Cloud Platform (XCP 1.6)**	**XenServer 6.0 (безкоштовна редакція)**	**XenServer 6.0 (платні редакції)**
Модель ліцензування	LGPL, GPL, Q Public License v1	Citrix EULA	Citrix EULA
Можливість створення миттєвих знімків віртуальних машин	Так	Так	Так
Можливість створення миттєвих знімків стану оперативної пам'яті віртуальних машин	Так	Ні	Так
Можливість підключення ПО управління XenCenter з графічним інтерфейсом користувача	Так	Так	Так
Утиліти для конвертації віртуальних машин	Так	Так	Так

Можливість динамічної міграції віртуальних машин	Так	Так	Так
Можливість використання гетерогенних серверів в ресурсному полі	Так	Ні	Так
Динамічне управління оперативною пам'яттю віртуальної машини	Так	Ні	Так
Вбудована система моніторингу продуктивності робочих серверів і віртуальних машин	Так	Ні	Так
Віртуальний розподілений комутатор	Так	Так	Так
Графічна консоль для управління віртуальним розподіленим комутатором	Ні	Ні	Так
Управління правами доступу для користувачів	Так	Ні	Так
Балансування навантаження між робочими серверами при запуску нових віртуальних машин	Так	Так	Так
Динамічне балансування навантаження між серверами	Ні	Ні	Так
Автоматичний перезапуск віртуальних машин при збої робочого сервера	Ні	Ні	Так

Як видно з таблиці 5.1, дистрибутив Xen Cloud Platform 1.6 має деякі важливими функціональними можливостями, яких немає у безкоштовній версії комерційного продукту XenServer - управління правами доступу, можливість використовувати гетерогенні сервера в ресурсному полі і деякими іншими. Функціонал дистрибутива Xen Cloud Platform 1.6 з точки зору цілей і завдань цієї роботи є достатнім і саме він обраний в якості основи обчислювальної підсистеми.

З архітектурної точки зору дистрибутив Xen Cloud Platform представляє звичайний дистрибутив операційної системи CentOS працює в режимі керуючого домену (Dom0) Xen. У дистрибутиві використовується остання версія гіпервізора Xen - 4.2. Як набору керуючих компонент використовується набір XAPI (Xen Management API), що розробляється компанією Citrix.

Набір керуючих компонент XAPI є надбудовою над гіпервізором, написаний на мові Objective Caml і поширюється вільно разом з дистрибутивом Xen Cloud Platform. Вихідний код XAPI доступний на сайті спільноти розробників[45].

XAPI додає розширену функціональність до базових можливостей гіпервізора Xen, таку як (не обмежуючи):

– Створення ресурсних пулів з робочих серверів;

– Розширений можливості управління віртуальними машинами: створення ієрархії миттєвих знімків стану віртуальних машин (включаючи пам'ять), жива міграція віртуальних машин;

– Розширені можливості по роботі з мережевою підсистемою і системою зберігання даних;

– Вбудована система моніторингу;

– Можливості напів-автоматичного розгортання оновлень на робочих серверах;

– Збір параметрів продуктивності обчислювальної підсистеми в режимі реального часу.

Вимоги до робочих серверів.

Для роботи системи в високо навантаженому, безперебійному режимі, рекомендується використовувати обладнання серверного класу для робочих серверів, але гіпервізор Xen сумісний з багатьма моделями обладнання для робочих станцій і ноутбуків.

Максимально підтримувана гіпервізором Xen конфігурація робочих серверів включає:

- до 1 ТБ оперативної пам'яті,
- до 16 мережевих карт,
- до 64 логічних процесорів.

Загальні вимоги до обладнання, на яке встановлюється гіпервізор і ПО управління, вказані в таблиці №5.2 і таблиці №5.3.

Таблиця 5.2. Вимоги до робочих серверів

Параметр	**Вимога**
Процесор	Один або більше 64-бітний x86 процесор з частотою 1.5 ГГц мінімум, рекомендована частота - від 2 ГГц, мінімум рекомендується 2 ядра на процесор. Для підтримки режиму повної віртуалізації гостьових операційн систем, а так само для запуску віртуальних машин під керуванн ОС Windows, потрібна підтримка процесорами технологій апаратної віртуалізації Intel VT або AMD-V. Опцію апаратної віртуалізації необхідно активувати в BIOS робочого сервера, то що за замовчуванням вона може бути відключена. Для запуску паравіртуалізованних операційних систем Linux, досить стандартних 64 бітових процесорів архітектруи x86.
іеративна м’ять	2 ГБ мінімум, 4 ГБ і більше рекомендується

исковий остір	Потрібно мінімум 16 ГБ дискового простору на локально підключеному накопичувачі (PATA, SATA, SCSI), рекомендується 60 Гб дискового простору. Можливе підключення до мережі зберігання даних (SAN) через апаратний HBA-адаптер, якщо система встановлюється з завантаженням з зовнішньої системи зберігання даних. При установці Xen Cloud Platform створюються два 4 ГБ розділу.
ережа	Мінімальна вимога - одна мережева карта 100 Мбіт/с або швидша. Рекомендується використовувати одну або кілька гігабітних мережевих карт для забезпечення високої пропускної спроможності між віртуальними машинами, відмовостійкості рішення і балансування навантаження.

Таблиця 5.3 Вимоги до керуючого сервера

Параметр	**Вимога**
Операційна система	Windows 7, Windows XP, Windows Vista, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2 (всі редакції і версії)
NET Framework	Версія 3.5
Частота процесора	750 Мгц мінімум, 1 Ггц і більше рекомендується
Оперативна пам'ять	1 Гб мінімум, 2 Гб і більше рекомендується
Вільний дисковий простір	100 МБ мінімум
Мережа	100- мегабітний або швидший мережевий адаптер
Роздільна здатність	1024x768 пікселів мінімум

4.4 Створення ресурсного пулу з робочих серверів

Ресурсний пул - об'єднання безлічі робочих серверів віртуалізації в єдину логічну сутність, що складається із сукупності ресурсів робочих серверів включених в ресурсний пул, керовану ззовні як єдине ціле.

Ресурсний пул в поєднанні з зовнішнім розділяються сховищем даних дає наступні переваги при роботі з системою:

- можливість легко масштабувати систему, додаючи нові сервера в ресурсний пул,
- можливість автоматично розміщувати нову віртуальну машину на відповідному робочому сервері в залежності від наявності вільних ресурсів,
- можливість здійснювати -живую‖ міграцію віртуальних машин між робочими серверами без припинення роботи віртуальних машин і запущених на них сервісів,
- можливість проводити профілактичні роботи на одному з серверів, тимчасово розміщуючи віртуальні машини на інших серверах в пулі.

В один ресурсний пул може бути включено до 16 робочих серверів. В середньому на одному сервері може бути запущено до 30-50 віртуальних машин при нормальному навантаженні. З огляду на це, на одному ресурсному пулі може одночасно працювати до 800 запущених віртуальних машин. З огляду на те, що в системі може створюватися будь-яку кількість ресурсних пулів, дана ресурсна характеристика є більш ніж достатньою для цілей цієї роботи.

У ресурсному пулі завжди присутній, принаймні, один сервер, що позначається як майстер (master). Цей головний в ресурсному пулі сервер є єдиним адміністративним інтерфейсом для управління усіма іншими серверами, майстер автоматично перенаправляє команди інших серверів в пулі в міру необхідності.

4.5 Вимоги для створення ресурсного пулу

Ресурсний пул є об'єднанням одного, або більше (до 16) однорідних робочих серверів. Однорідність означає, що:

– Процесори на серверах повинні бути повністю однакові або майже однакові з точки зору виробника, моделі і функціональності.

– На робочих серверах повинно бути запущено абсолютно ідентичне програмне забезпечення Xen Cloud Platform, з однаковими встановленими патчами.

Додатковими обмеженнями є:

– Робочий сервер не повинен бути членом вже існуючого ресурсного пулу.

– Сервер на момент підключення до ресурсного пулу не повинен бути підключений до загального розділяється сховища даних.

– На робочому сервері не повинно бути запущених або призупинених віртуальних машин на момент приєднання;

– Всі операції з віртуальними машинами на момент приєднання повинні бути завершені.

Всі робочі сервери можуть мати різну кількість мережевих карт, різну конфігурацію локальних дисків і більш того - мати невеликі відмінності в моделях процесорів. У разі відмінності процесорних потужностей, неоднорідності серверів, все віртуальні машини будуть створюватися з віртуальними процесорами, відповідними по характеристикам найменш потужному і найменш функціональним фізичній процесору робочого сервера в ресурсному пулі.

Створення ресурсного пулу - не обов'язкова умова для роботи з серверами віртуалізації, переваги ресурсного пулу з'являються тільки тоді, коли до ресурсного пулу підключено розділяється сховище даних. Тому створювати ресурсний пул рекомендується тоді, коли вже підключено загальне поділюване сховище даних для підключення, а до цього - використовуємо робочі сервери віртуалізації окремо використовуючи в якості сховища даних локальні жорсткі

диски серверів. Після того, як ресурсний пул створений, можна мігрувати віртуальні машини разом з віртуальними жорсткими дисками з локального сховища на поділюване сховище. Це може бути зроблено за допомогою команди xe vm-copy, або за допомогою графічної консолі XenCenter.

4.6 Створення ресурсного пулу

Ресурсний пул може бути створений як з графічного інтерфейсу XenCenter, так і їх командного рядка. Коли новий робочий сервер підключається до ресурсного пулу, він синхронізує свою локальну базу даних з настройками з базою даних за все ресурсного пулу, при цьому частина налаштувань успадковується від налаштувань ресурсного пулу:

– Віртуальні машини робочого сервера, а так само підключається і локальне сховище даних, додаються в базу даних ресурсного пулу;

– Приєднуйтесь робочий сервер отримує настройки поділюваних сховищ ресурсного пулу і в у нього створюються відповідні PBD-пристрої, за допомогою яких робочий сервер може автоматично підключитися до разделяемому сховища ресурсного пулу;

– Частково успадковуються мережеві настройки ресурсного пулу.

Створити ресурсний пул і включити в нього робочі сервери, задати базові настройки можна в інтерфейсі управління XenCenter.

Віртуальні машини створюються на основі шаблонів. Шаблон - це «золотий образ», содержащійвсевозможние параметри конфігурації для створення екземплярів певної віртуальної машини.

XenCloudPlatform включає базовий набір шаблонів, які варіюються в діапазоні від «необроблених» віртуальних машин, на яких можна завантажити інсталяційний компакт-диск постачальника з операційною системою або виконати установку з мережевого сховища, доповнивши налаштованих примірників операційної системи.

Оптимальні параметри роботи в різних операційних системах дещо розрізняються. Шаблони віртуальних машин налаштовані спеціально для забезпечення максимальної продуктивності системи.

У шаблонах для ОС Linux передбачено створення паравіртуальних гостьових систем, на відміну від гостьових систем HVM, які створюються шаблонами для Windows і інших установочних носіїв ПЗ.

Існує три основних способи створення віртуальних машин за допомогою шаблонів:

– використання повністю налаштованого шаблону;
– установка з компакт-диска або з ISO-образу в відповідний шаблон;
– установка з носія ПЗ від постачальника на сервері мережевої установки безпосередньо в шаблон.

Крім цього, віртуальні машини можна створювати також наступними способами:

– за допомогою інструменту перетворення фізичної системи у віртуальну (P2V) і віртуальної системи в віртуальну (V2V) під назвою XenConvert;
– шляхом імпорту наявної експортованої віртуальної машини;
– шляхом перетворення наявної віртуальної машини в шаблон.

У нашій роботі будемо використовувати зовнішнє сховище образів операційних систем, і використовувати образи їх у міру необхідності. З образів створюються віртуальні машини з необхідною операційною системою, потім адміністратор вручну встановлює необхідний набір програмного забезпечення і бібліотек: MapReduce, OpenMP, MPI, Hadoop, BOINC. Мережеві настройки яка ставить в режим автоматичного отримання параметрів по DHCP, для того щоб після ініціалізації віртуальної машини не було необхідності вручну змінювати мережеві настройки і віртуальна машина була б готова до роботи. Коли віртуальна машина повністю підготовлена, вона конвертується в шаблон і потім може бути використана в встановленою конфігурації довільну кількість разів.

4.7 Мережева підсистема

Гостьова операційна система Xen отримує доступ до мережі за допомогою паравіртуальних мережевих інтерфейсів. Паравіртуальні мережеві інтерфейси забезпечують швидкий і ефективний спосіб мережевої взаємодії доменів гостьових операційних систем без витрат на повну емуляцію реального мережевого пристрою. Для більшості операційних систем, що підтримують паравіртуалізацію, в Xen за замовчуванням доступні спеціальні драйвери для мережевих пристроїв. Так само, Xen забезпечує паравіртуальними драйверами і деякі операційні системи, що працюють в режимі повної віртуалізації (HVM-режим), наприклад ОС Windows.

Кожний паравіртуальний мережевий пристрій складається з двох частин - фронтендів мережевого пристрою, який відображається в гостьовій операційній системі, і бекенда, який відображається в керуючому домені (dom0). Ця пара мережевих пристроїв завжди створюється для кожного мережевого інтерфейсу. Зв'язок між фронтендів і бекенда забезпечується через швидкий віртуальний комунікаційний канал на рівні гіпервізора [30, 31,32].

Крім режиму паравіртуалізації, для Xen завжди доступний режим повної емуляції мережевих пристроїв. В такому випадку адаптовані пристрої повністю відтворюють реальне мережеве обладнання й ефективні тоді, коли операційна система не має паравіртуальними драйверами або коли вони ще не доступні (наприклад, на етапі встановлення гостьової операційної системи).

Емулюючий мережевий пристрій завжди створюється в парі з паравіртуальним пристроєм на стороні керуючого домену, з тим же MAC-адресою і конфігурацією. Це дозволяє легко перейти на використання паравіртуального драйвера, коли драйвер стане доступним для операційної системи.

4.7.1 Віртуальні комутатори або мости

За замовчуванням (і як найбільш поширений варіант конфігурації мережі) Xen використовує механізм віртуальних комутаторів або мостів всередині керуючого домену.

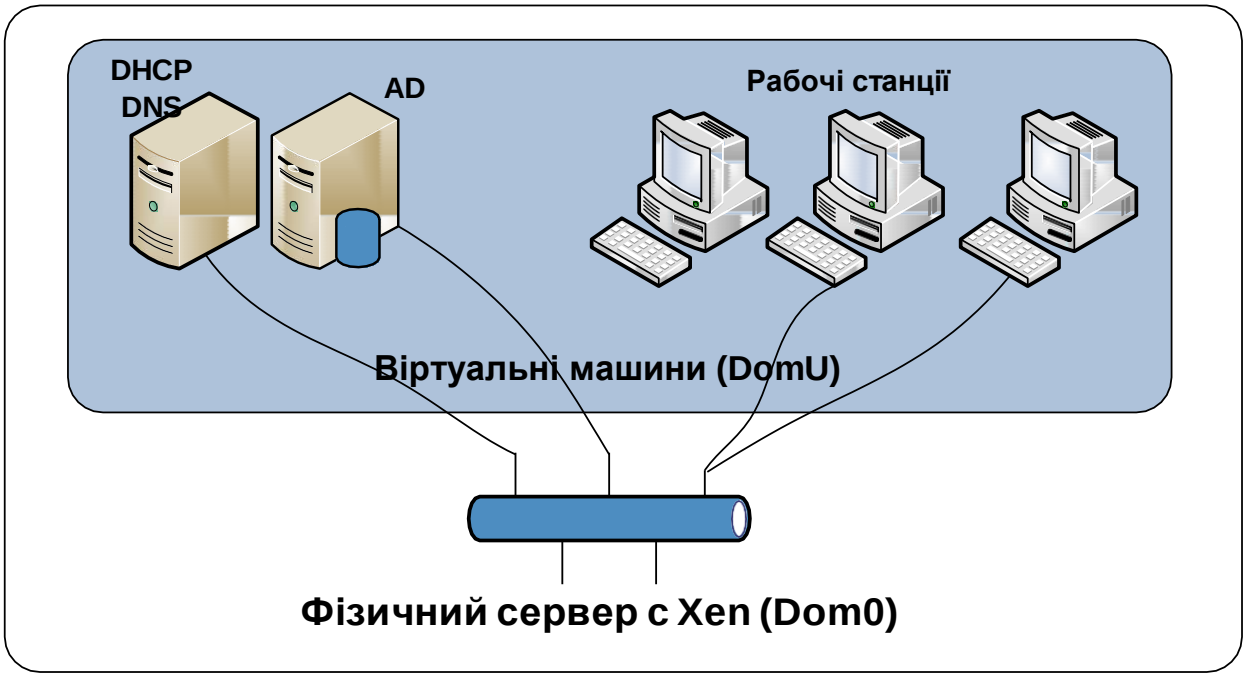


Рис. 4.2. Мережева підсистема окремого робочого сервера (внутрішня мережа)

На рис. 4.2. зображена спрощена схема мережі на окремому робочому сервері з встановленим гіпервізором Xen. Кожна віртуальна машина має віртуальний мережевий адаптер, який підключений до віртуальної мережі, яка управляється керуючим доменом (Dom0) робочого сервера.

За замовчуванням, зображена мережа працює точно так само як звичайний фізичний сегмент мережі Ethernet на 2 рівні моделі OSI. На цьому рівні не потрібно ніякої конфігурації мережевого рівня, наприклад конфігурація TCP / IP. Як і в разі реальної мережі, спосіб подальшого використання мережі визначається мережеве налаштування операційних систем віртуальних машин. Якщо необхідно налаштувати TCP / IP підключення, то це необхідно зробити в налаштуваннях операційної системи віртуальної машини. Таким же чином здійснюються настройки DNS і DHCP служб.

Те що зображено на рисунку 4.2. називається внутрішньою мережею (Internal Network). Назва обумовлено тим, що вона не має ніякого зв'язку із зовнішнім світом, працюючи виключно на рівні віртуальних машин окремо взятого робочого сервера.

На рисунку 4.3. зображена зовнішня мережа робочого сервера. Єдиною відмінністю між внутрішньою мережею і зовнішньою мережею є те, що віртуальна мережа підключена до реального фізичного адаптера сервера, в той час як внутрішня мережа існує сама по собі. Це означає що віртуальні машини підключені в зовнішній мережі можуть повноцінно використовувати всі реальні мережеві сервіси, що надаються зовнішнім світом за межами робочого сервера.

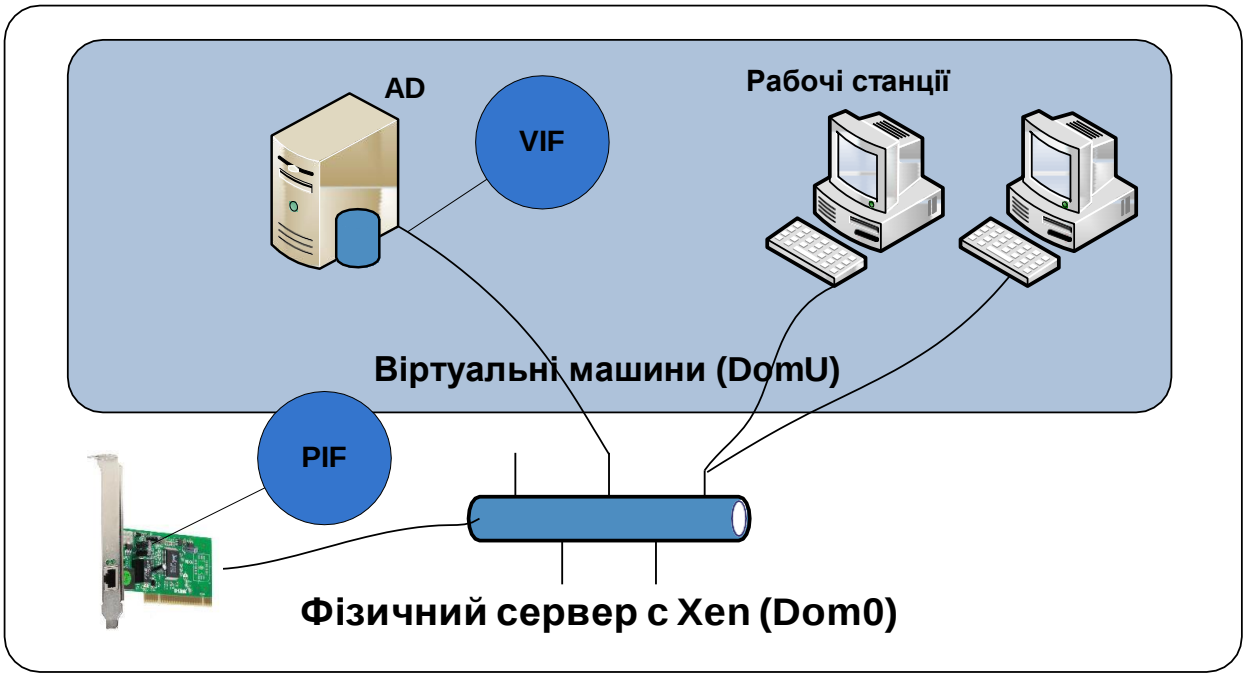


Рис. 4.3. Мережева підсистема окремого робочого сервера (зовнішня мережа)

У термінології Linux і Unix систем будь-яка мережева карта називається інтерфейсом.

У термінології Xen реальний інтерфейс називається фізичним інтерфейсом або PIF (Physical Interface), а кожен віртуальний інтерфейс підключений до віртуальної машини називається віртуальним інтерфейсом або VIF (Virtual Interface).

У занедбаній на віртуальній машині операційній системі віртуальний інтерфейс виглядає і працює так, як ніби це реально встановлений фізичний інтерфейс. Єдиною відмінністю може бути назва пристрою в залежності від того, який паравіртуальний драйвер використовується.

Xen використовує механізм віртуальних комутаторів або мостів для об'єднання різних мережевих інтерфейсів в єдиний сегмент мережі Ethernet. На рисунку 4.4. зображений віртуальний комутатор, до якого можуть бути підключені будь-яку кількість віртуальних інтерфейсів віртуальних машин, а так само, опціонально, один або більше фізичних інтерфейсів сервера, якщо мережа є зовнішньою.

Принцип роботи віртуального комутатора такий же як і фізичного комутатора. Складаючи таблиці MAC-адрес підключених до нього інтерфейсів, він перенаправляє трафік саме на той порт, до якого підключений інтерфейс-одержувач пакета.

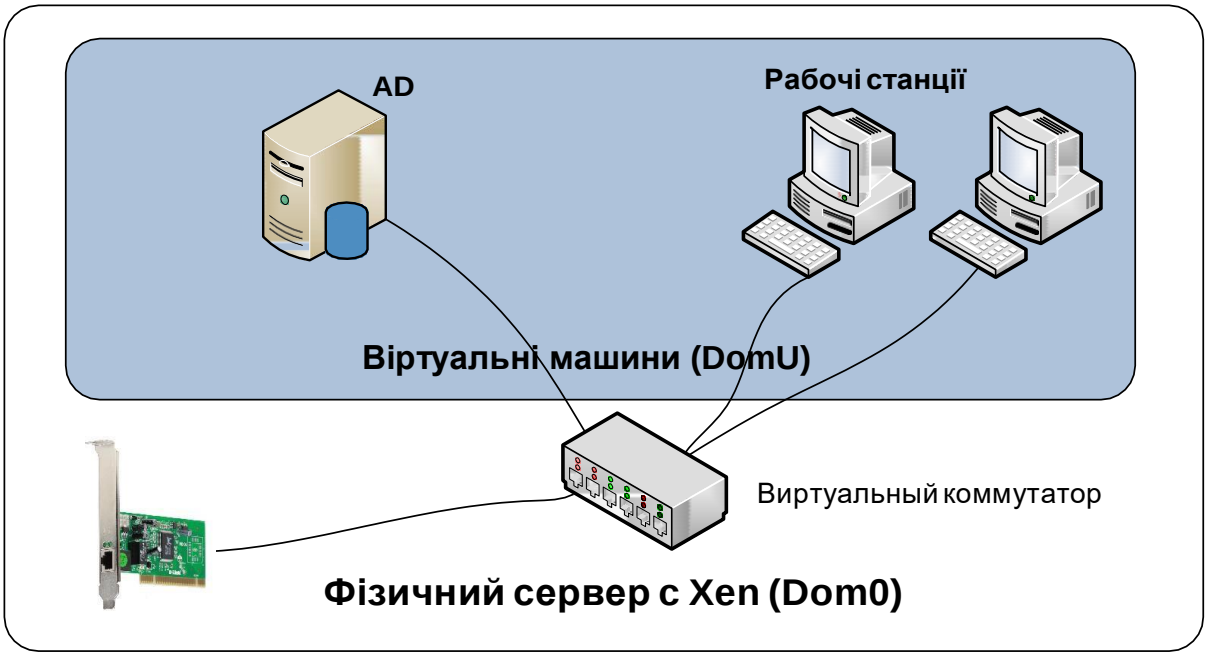


Рис. 4.4. Віртуальний комутатор

На відміну від фізичних комутаторів, які крім завдань фізичного підключення кінцевих пристроїв до мережі, призначені для поділу доменів колізій між мережами, віртуальні комутатори в першу чергу виконують

завдання забезпечення підключення віртуальних машин до загальної віртуальної мережі і до зовнішнього світу.

На робочому сервері можна створити будь-яку кількість ізольованих віртуальних мереж, але єдиним способом підключення декількох внутрішніх віртуальних мереж один до одного буде створення віртуальної машини з двома віртуальними інтерфейсами, кожен з яких підключений до своєї мережі, як це зображено на рисунку 4.5. Кожен інтерфейс належить виключно до однієї мережі, але віртуальна машина може мати кілька віртуальних інтерфейсів.

Таким чином, віртуальна машина у внутрішній мережі Q може отримати доступ до зовнішньої мережі R через віртуальну машину, яка має два віртуальних інтерфейсу кожен у своїй мережі. Внутрішні мережі N і P ніяким чином не зможу спілкуватися один з одним або з фізичною інтерфейсом.

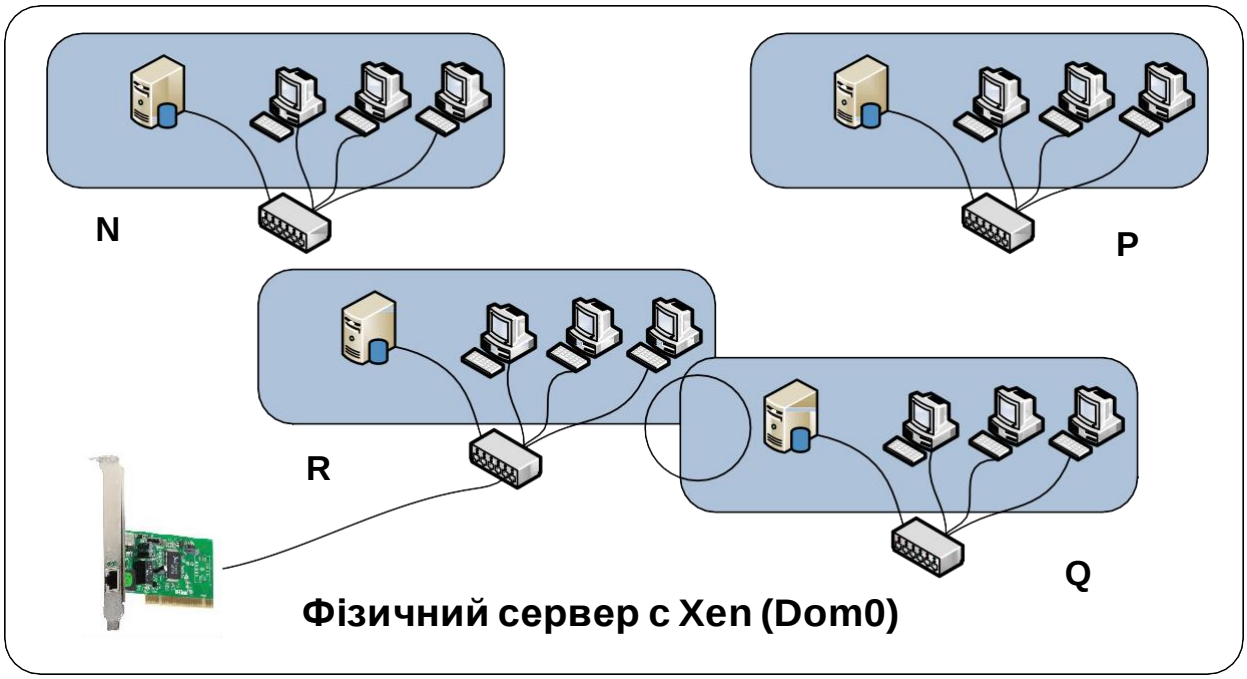


Рис. 4.5. Створення декількох віртуальних мереж

На рисунку 4.6. зображено приклад конфігурації мережі робочого сервера. Кожній віртуальній машині присвоюється унікальний ідентифікатор, в зазначеному прикладі віртуальної машині Windows присвоєно номер 1, а віртуальній машині Linux присвоєно номер 2.

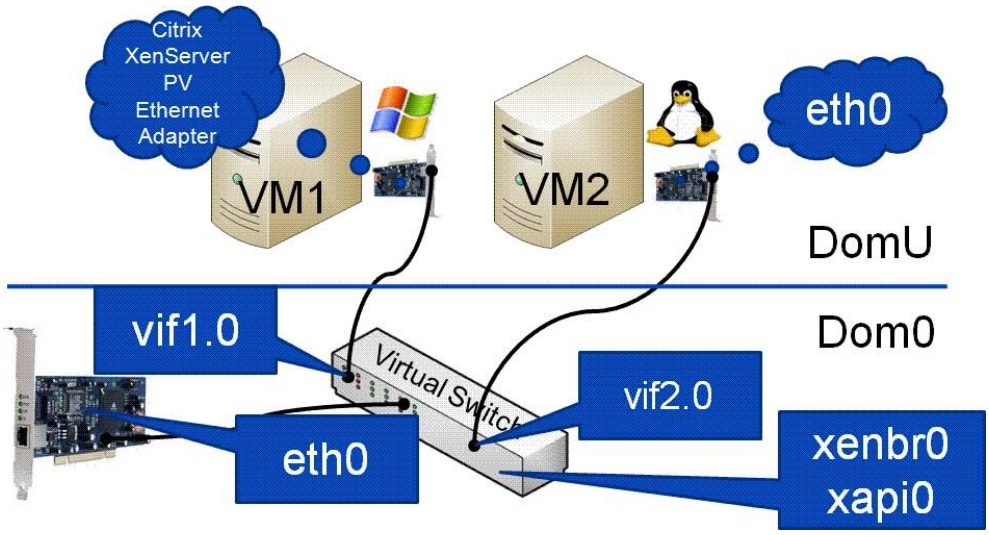


Рис. 4.6. Приклад конфігурації мережі робочого сервера

Припускаючи, що на віртуальні машини встановлені пара-віртуальні драйвери, кожна віртуальна машина буде бачити реальний для її мережевий адаптер. У віртуальній машині Windows в диспетчері пристроїв буде відображатися «Citrix XenServer PV Ethernet Adapter», а віртуальна машина Linux буде бачити -eth0.

Мережеві інтерфейси віртуальних машин відображаються на стороні керуючого домену робочого сервера (dom0), як vif1.0 і vif2.0. Перша цифра перед точкою означає відповідну віртуальну машину, а друга цифра після точці - порядковий номер інтерфейсу в віртуальній машині. Мережевий адаптер віртуальної машини №1 буде відображатися як VIF1.0 внтурі керуючого домену. Кожен фрейм, переданий цій віртуальною машиною на свій мережевий адаптер буде отримано на VIF1.0.

Реальний мережевий інтерфейс буде відображатися як eth0 і так само буде підключений до мосту.

Кожен міст також відображається як інтерфейс. Типово если міст створений для зовнішньої мережі, то його назва починається з -xenbr‖. Якщо міст створений для внутрішньої мережі, то він його назва починається з -xapi.

Іншими варіантами організації мережі на рівні робочого сервера, крім організації мостів, є використання механізмів маршрутизації (routing) і трансляції мережевих адрес (NAT), а так само їх комбінації.

Для цілей роботи підходять два варіанти побудови мережі - прив'язка кожної нової створеної віртуальної машині виділеного публічного IP-адреси з пулу доступних публічних IP-адрес (плоска організація мережі, публічні IP-адреси розподіляються дінмаіческі) і організація машрутізіруемой мережі.

При цьому, самими робочим серверів так само привласнюється публічний IP-адресу, вони знаходяться в публічному секторі мережі.

Плоска організація мережі

При плоскій організація мережі - відбувається динамічне присвоєння публічних адрес віртуальних машин. Оскільки в настройках шаблонів віртуальних машин вказується динамічне отримання мережевих налаштувань, при ініціалізації віртуальної машини і завантаженні операційної системи, походить автоматичне отримання IP-адреси і віртуальна машина стає доступна для роботи.

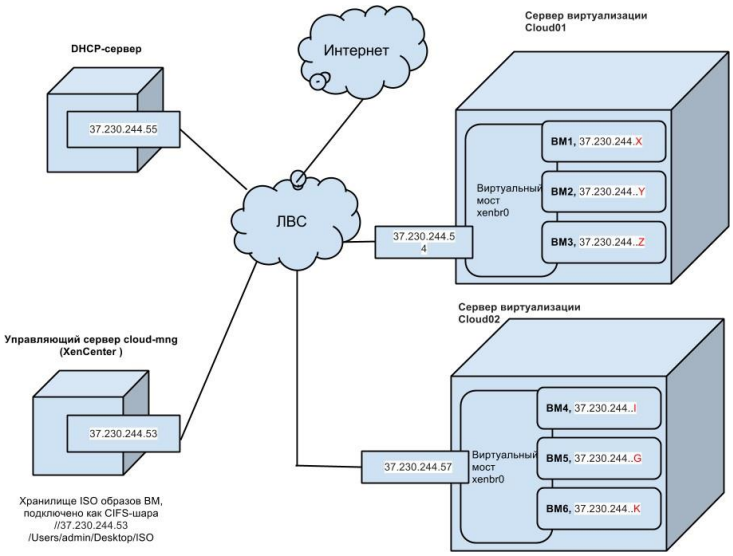


Рис. 4.7 Плоская організація мережі

У цьому випадку Xen працює як міст, будь-якої машрутизаціі не потрібно. Підключення до віртуальних машин відбувається по SSH або

RFBпротоколу, якщо це операційні вашої системи Linux, або по RDPпротоколу, якщо це операційні системи Windows.

Переваги такого підходу:

– Прямий доступ користувачів до віртуальних машин з будь-якої точки через Інтернет;
– Простота масштабування рішення.

Недоліки:

– Ризики несанкціонованого доступу, відсутність захищеного периметру мережі;
– Неможливість створити ізольовані підмережі;
– Дифіцит і вартість публічних IP-адрес.

Створення ізольованих мереж з приватної адресацією. При такій реалізації мережі забезпечення доступу до віртуальних машин відбувається через графічну консоль vnc через браузер. За ssh –
основне підключення для роботи з віртуальною машиною. Ssh прокидати по портам на зовнішній IP-адреса, при створенні віртуальної машини повідомляється номер порту.

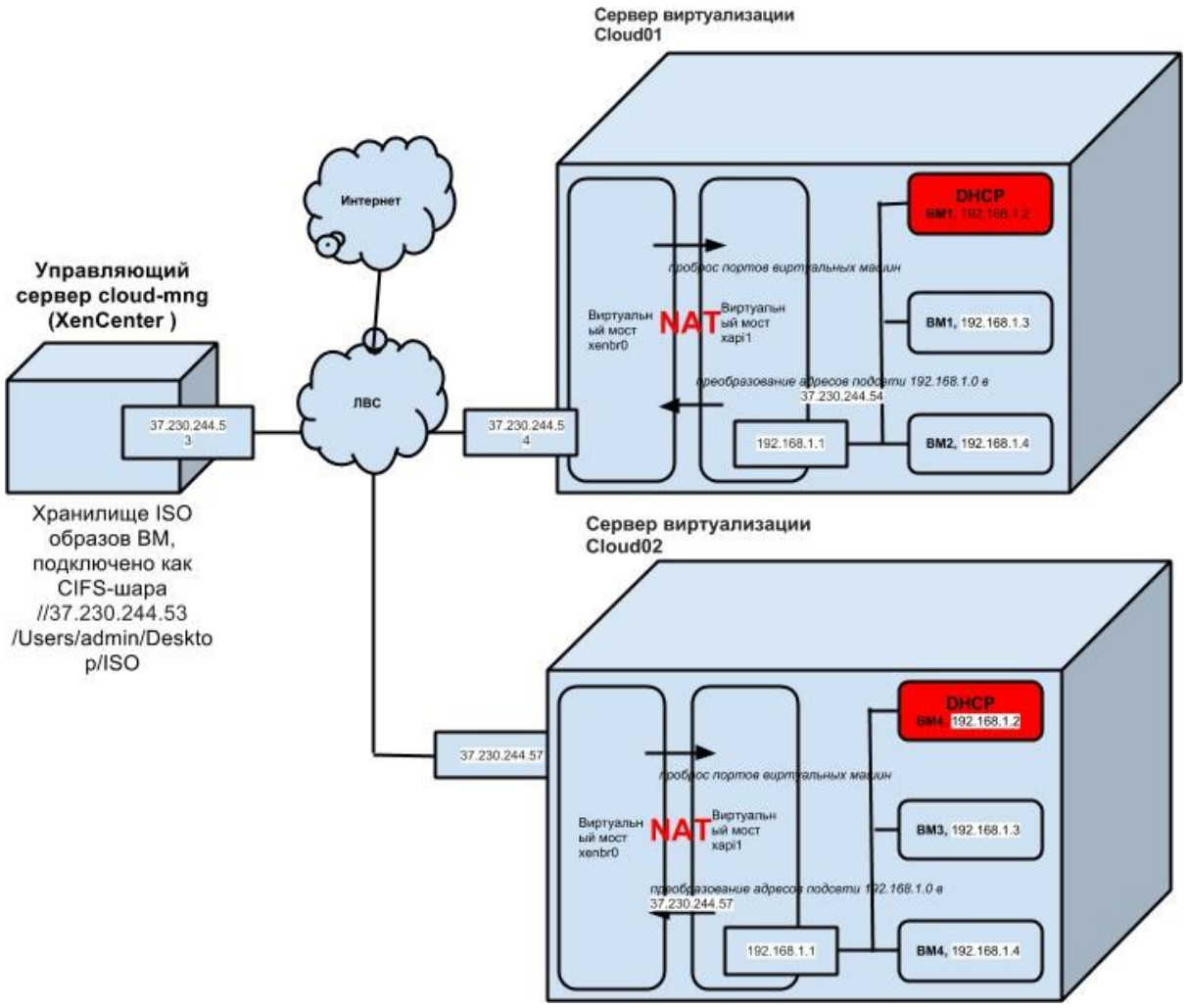


Рис. 4.8.Маршрутизуєма мережа

4.8 Особливості проектування мережевої підсистеми

На кожному робочому сервері повинно бути не менше одного фізичного мережевого адаптера. Проте, точна кількість мережевих адаптерів необхідно прораховувати виходячи з вимог до отказоуйстойчівості мережевої підсистеми і обсягу трафіку, що проходить через мережеві адаптери, вимог до ізоляції трафіку, наявності мережі зберігання даних, що працює через Ethernet. Мінімально рекомендованої конфігурацією не для тестових цілей, а для робітників розгортання, є мінімум 2 мережевих інтерфейсу на кожному робочому сервері: один для трафіку віртуальних машин, інший для керуючого трафіку.

Визначення точної смуги пропускання складається з наступних факторів:

– Кількість віртуальних машин;
– Тип навантажень на віртуальних машинах і тип трафіку, що проходить;
– Спеціальні вимоги до смуги пропускання для певних сервісів;

Необхідно визначити необхідність у використанні розподіленого віртуального комутатора. Ця необхідність виникає, коли потрібно забезпечити:

– створення віртуальних мереж між фізичними серверами;
– забезпечити QoS;
– можливість аналізувати трафік за допомогою стандартних засобів і протоколів, наприклад, таких як RSPAN і NetFlow;
– Спрощене адміністрування мережевої підсистеми.

4.9 Підсистема зберігання даних

Логічна схема організації системи зберігання даних для гіпервізора XenCloudPlatformізображена на рис. 4.9

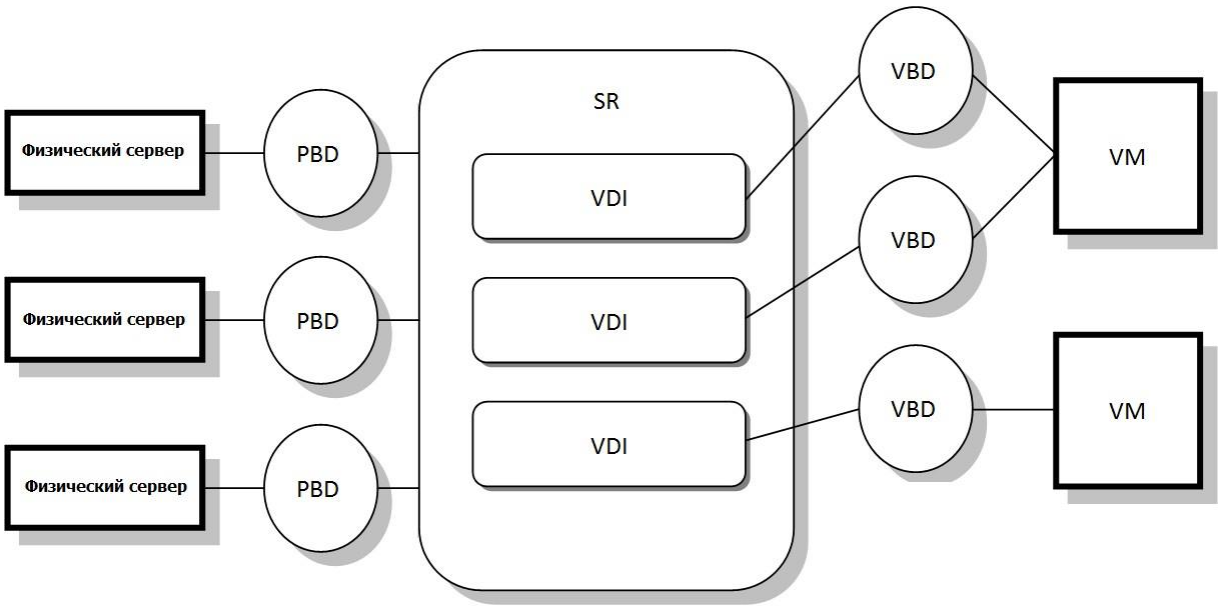


Рис. 4.9. Логічна схема системи зберігання даних гіпервізора XenCloudPlatform

У гіпервізора Xen система зберігання даних складається з самої системи зберігання даних, яку представляють програмним інтерфейсом SR (storage repository) на робочому сервері, а так само пов'язаних з нею об'єктів в самому гіпервізора, і об'єктів, які гипервизор створює на системі зберігання даних [33].

Дані віртуальних машин зберігаються на системі зберігання даних в об'єктах, які називаються віртуальними дисками - VirtualDiskImages (VDI).

SR підтримує підключення IDE, SATA, SASдисков як локальної системи зберігання даних, а так само iSCSI, NFS, SASi FibreChannelcістем зберігання даних в якості зовнішніх поділюваних систем. Функціонал SR дозволяє використовувати такі можливості системи зберігання даних, як миттєві знімки, швидке клонування, «розумне» використання дискового простору (коли враховується тільки той дисковий простір, який реально зайнято, а не просто зарезервовано під віртуальні диски). Якщо система зберігання даних сама не підтримує розширений функціонал (наприклад NFS), то програмне забезпечення системи зберігання даних Xen використовує розроблену компанією Майкрософт технологію віртуальних жорстких дисків (Microsoft'sVirtualHardDisk або VHD), яка сама реалізує даний функціонал на програмному рівні.

Кожен робочий сервер може бути підключений до різних систем зберігання даних різних типів одночасно. Ці SRмогут бути загальними для всієї інфраструктури, або виділеними для обслуговування окремих робочих серверів. При підключенні загальних систем зберігання даних робочі сервери об'єднуються в ресурсний пул. У ресурсному пулі повинен бути підключений хоча б одна система зберігання даних.

Дані віртуальних машин зберігаються на віртуальних дисках (VDI). Для підключення віртуальних дисків до віртуальних машин використовуються спеціальні конектори, звані віртуальним блоковими пристроями (VirtualBlockDevice або VBD). Віртуальні диски є незалежними реальними фізичними об'єктами, постійно зберігається на системі зберігання даних, в той час як VBD являються програмної абстракцією.

Аналогічно VBD для підключення до системи зберігання даних використовується інтерфейсний об'єкт PBD (PhysicalBlockDevice). У PBD зберігається конфігураційна інформація, що відноситься до певної системи зберігання даних. Наприклад, при використанні NFSсистеми зберігання даних, в PBD буде IP адреса NFS сервера і параметри доступу до нього. PBD об'єкти керують підключенням систем зберігання даних до фізичних серверів.

Існують типу підключень VDI:

– Файловий спосіб підключення VHD, при якому образи віртуальних машин зберігаються в VHD-файлах або на локальних дисках, або на зовнішній NFS-системі зберігання даних;
– Підключення логічних томів на LUN, при якому використовується блоковий доступ системі зберігання даних;
– Підключення LUN-ів безпосередньо до віртуальних машин.

Підключення системи зберігання даних здійснюється через панель управління XenCenter, або за допомогою команди sr-create.

Для інфраструктури віртуальних полігонів будемо використовувати NFS систему зберігання даних, тому що дана система забезпечує:

– Низьку вартість реалізації системи для невеликих навантажень на інфраструктуру за рахунок можливості використання стандартного апаратного забезпечення і програмного забезпечення з відкритим вихідним кодом;
– Простоту настройки і експлуатації системи;
– Можливість адміністратору підключатися до системи зберігання даних і вручну копіювати файли віртуальних машин;
– Достатню продуктивність системи для цілей освітніх установ.

При використанні NFS систем зберігання даних VHDфайли зберігаються на зовнішній файлової системи. Xen дозволяє використовувати все реалізації NFSсерверів, що підтримують третю версію протоколу NFS. VDI зберігаються виключно в форматі VHD.

Для підключення NFS системи зберігання даних потрібно тільки IP адреса NFSсервера. NFSсервер повинен бути налаштований так, щоб було можливим підключити всі робочі сервера Xen.

NFS дозволяє здійснювати «розумне» зберігання VDI-файлів. Це означає, що VDI-файлу буде виділятися рівно стільки дискового простору, скільки записано на диск у віртуальній машині. Якщо наприклад на 100 ГБ диск записана тільки операційна система, це буде означати, що реально зайнята тільки мала частина цього диска.

Так само, VHD формат підтримує розумне клонування файлів віртуальних машин. При клонуванні VHD-файлу, новий файл містить посилання на старий файл і реально зберігає виключно нову інформацію. За рахунок цього досягається швидке створення віртуальних машин з шаблонів, а так само - клонування віртуальних машин.

У порівнянні з використанням блокових систем зберігання даних, NFS-система зберігання даних може мати проблеми з продуктивністю. Це пов'язано з тим, що VHDфайли містять багато мета-інформації (в т.ч. для реалізації «розумного» клонування і «розумного» використання дискового простору) і робота з цією мета-інформацією вимагає високої продуктивності фізичного сервера, на якому працює NFS- система зберігання даних.

Так само, при використанні NFS системи зберігання даних, адміністратор системи може безпосередньо бачити файли віртуальних машин і весь вміст NFS сервера, віддане системі віртуалізації. Непродумане ручне втручання в ці дані, їх пошкодження може викликати непрацездатність всієї системи.

При підключенні NFSсистеми зберігання даних в Xen по замовчуванням використовується TCPпротокол. UDP протокол можна подключати опціонально.

Процес підключення NFS системи зберігання даних VBD. Для підключення NFS сервера з IP адресою 192.168.1.10 і папкою/ Exprot1, необхідно виконати наступну команду в командному рядку робочого сервера:

xe sr-create host-uuid=<host_uuid> content-type=username-label=<"Example shared NFS SR"> shared=truedevice-config:server=<192.168.1.10> device-config:serverpath=</export1> type=nfs

Підключення блокових пристроїв у віртуальній машині. У разі паравіртуалізації в ОС Linux блокові пристрої переносяться у вигляді паравіртуалізірованних пристроїв. В Xen не робиться спроба емуляції SCSI або IDE, а надається більш зручний інтерфейс для віртуального середовища в формі пристроїв xvd * [34].

Для Windows або інших повністю віртуалізованих гостьових систем емулює шину IDE у вигляді пристрою hd *. У разі використання Windows при установці XenServer Tools виконується установка спеціального паравіртуалізірованного драйвера, принцип роботи якого аналогічний Linux, крім випадку повністю іртуалізірованной середовища.

Будь-яке блоковий пристрій хост-системи може бути експортовано в гостьовій домен паравіртуальной машини. Для цього в файлі конфігурації домену використовується параметр disk, а в разі гарячого підключення пристроїв підкоманди block-attach і block-detach команди xm.

Експортовану блоковий пристрій бачиться всередині домену, забезпечується драйвером blkfront, які працюють усередині домена, і бачиться як звичайне блоковий пристрій, під ім'ям виду / dev / hda1 або / dev / sda1.

Приклад використання директиви disk:

disk = ['phy: / dev / PLAIN2 / test, hda1, w']

Блоковий пристрій / dev / PLAIN2 / test хост-системи передаèтся всередину гостьового домена як пристрій / dev / hda1.

Для HVM-доменів блокові пристрої емулюються як диски (QEMU IDE) або передаються за допомогою паравіртуальних драйверів.

4.10 Система автоматизації надання послуг та забезпечення універсального доступу

Технічне управління і адміністрування системою здійснюється з програми з з графічним інтерфейсом користувача XenCenter (поширюється вільно), яка встановлюється на будь-який комп'ютер з операційною системою Windows.

Можливі два варіанти роботи з XenCenter: локально і через термінальний доступ. У разі використання термінального підключення, допустимий ping для роботи з RDP інфраструктурою - не більше 200 мс, смуга - в середньому 40 кбіт / користувача.

У загальному випадку, можливий варіант організації роботи з віртуальної інфраструктурою виключно за допомогою графічної консолі XenCenter. Але оскільки доступ до консолі здійснюється через термінальний сервер, що пов'язане з певними витратами на закупівлю ліцензій для цього сервера, а так само фукциональность XenCenter занадто велика і націлена в першу чергу на адміністратора системи, доцільно користувачам системи надавати доступ через веб поратл самообслуговування, в якому буде доступний лише органічним, строго заданий адміністратором, функціонал роботи з системою. Так само, за допомогою функціоналу веб-порталу можна організувати більш складні бізнес-процеси по роботі з віртуальною інфраструтури вже засобами самого сервера додатків.

Веб-портал використовується користувачами системи наступним чином:

– Користувач має можливість бачити настройки і опис наданої йому віртуальної машини (групи віртуальних машин), параметри доступу до віртуальної машини (машин);

– Користувач має можливість запускати, перезапускати або зупиняти віртуальні машини з наданого йому набору віртуальних машин, підключатися до віртуальної машини в будь-якому її стані через графічну консоль

– Адміністратор системи забезпечує активацію віртуальних машин з наявних шаблонів для заданих користувачів, при необхідності забезпечує вимикання і видалення віртуальних машин;

Розглянемо алгоритм побудови веб-поратала, який зможе реалізувати необхідну фукнціонального.

Встановлений на сервер дистрибутив XenCloudPlatform містить весь необхідний інструментарій для доступу до своєї функіональние допомогою протоколавизова віддалених процедур XML-RPC. Не дивлячись на те, що можна використовувати безпосередні XML-RPCвиклики для взаємодії віртуальної інфраструктурою, для прискорення розробки програмного забезпечення, віддалено взаємодіючих з віртуальної інфраструктурою, розробники пропонують скористатися вільно поширюваним набором бібліотек SDK, розробленим для мов C, C #, Java, Python [35, 36,37].

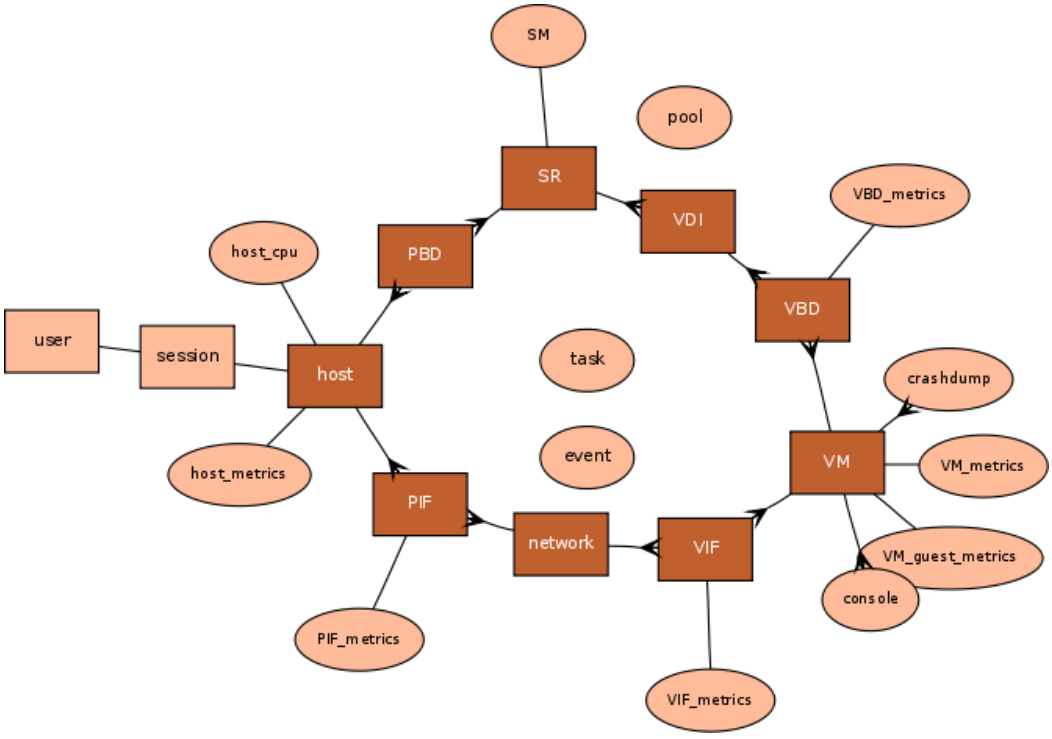


Рис. 5.10. Діаграма класів XenAPI

API, за допомогою якого відбувається управління віртуальною інфраструктурою, має наступні характеристики:

– Можливість управляти всіма аспектами роботи робочого сервера, включаючи управління життєвим циклом віртуальних машин, системою зберігання даних, мережею, конфігурацією фізичного сервера, ресурсних пулів.

– Можливість отримувати повну статистику про роботу фізичного сервера і віртуальних машин

– Об'єктна модель зі збереженням стан (persistetobjectmodel) - результати всіх операцій зберігаються в сервеній базі даних

– Механізм подій. Можливість отримувати клієнтом повідомлення про зміни, що відбулися на сервері.

– Синхронні і асинхронні виклики процедур

– Забезпечення безпечного доступу і аутентифікації користувачів.

Для реалізації заявлених функцій веб-порталу, використовуємо сервер додатків ApacheTomcat, додаток проектуємо на мові Java. Реалізація функцій бізнес-логіки здійснюється стандартними методами JavaEE при клієнт-серверній розробці, зупинимося більш детально на взаємодії веб-сервера з віртуальною інфраструктурою. Для забезпечення даної фукніональності вSDK є бібліотеки для Java, додатково потрібна бібліотека з реалізацією xml-rpc протоколу для клієнта.

Типовий сценарій роботи програми на веб сервері наступний, жирним відзначені моменти, пов'язані із взаємодією веб-сервера з робочим сервером віртуалізації:

А) для звичайного користувача:

1. Авторизація користувача. На цьому етапі засобами сервера додатків необхідно визначити чи є користувач адміністратором, або звичайним користувачем. В залежності від цього буде представлена та чи інша фукнціональність.

2. **Ініціалізація сесії з робочим сервером.** Для зручності управління, авторизація клієнта завжди походить від імені користувача root, управління

звичайними користувачами і розмежування прав здійснюється виключно засобами сервера додатків.

3. **Отримання повного списку шаблонів віртуалних машин.**

4. Відображення статусів віртуальних машин, видимих користувачеві, **запуск віртуальної машини.** Можливість отримання параметрів для віддаленого доступу до віртуальних машин (IP-адреса, порт, логін, пароль), можливість підключиться до графічної консолі віртуальної машини через веб-браузер. Отримання інформації за рахунок xml-rpcвизовов до робочого сервера або з попереднього списку всіх віртуальних машин.

5. Здійснення необхідних дій з віртуальними машинами, при якому відбувається обмін xml-rpcвизовамі.**Управління віртуальною машиною.**

6. Отримання доступу до консолі віртуальної машини.

7. Завершення сесії.

Б) для адміністратора буде точно такий же алгоритм дій, але йому будуть доступні всі віртуальні машини а так само прикріплені до них користувачі, можливість призначати доступ користувачам до тих чи інших віртуальних машин.

Ініціалізація сесії з робочим сервером. На етапі ініціалізації сесії з робочим сервером відбувається процес аутентіфікаціі за рахунок виклик процедури **Session.login_with_password** (<username>, <Password>, <client_API_version>). Тільки після цього можна здійснювати інші запити до сервера. В результаті успішної авторизації клієнт отримує ідентифікатор сесії (sessionreference). Наступні виклики використовують ідентифікатор сесії в якості обов'язково параметра. За рахунок цього забезпечується гарантія того, що тільки автозованние користувачі можуть працювати з віртуальною інфраструктурою.

Отримання повного списку доступних шаблонів віртуальних машин. Наступним кроком при роботі з сервером є отримання повного списку доступних шаблонів віртуальних машин. Щоб отримати список шаблонів, необхідно знайти всі об'єкти віртуальних машин, у яких параметр **is_a_template**

встановлений як ІСТИНА. Це здійснюється викликом **VM.get_all_records (session),** де session - раніше отриманий ідентифікатор сесії. Серед всіх повернутих екземплярів віртуальних машин необхідно методом сортування знайти шаблони.

Запуск віртуальної машини з шаблону. Для запуску віртуальної машини з шаблону здійснюються такі дії:

– Клонування об'єкта віртуальної машини з ідентифікатором t**_ref - VM.clone (session, t_ref, "my first VM").** Цей виклик повертає ідентифікатор, щойно створеної віртуальної машини - **new_vm_ref.**

– На даному етапі, об'єкт з ідентифікатором **new_vm_refвсе** ще є шаблонів віртуальної машини, а не повноцінною віртуальною машиною. Щоб перетворити шаблон в віртуальну машину, необхідно здійснити телефонний дзвінок **VM.provision (session, new_vm_ref).** Після того, як виклик поверне ідентифікатор об'єкта віртуальної машини, параметр **is_a_template** вже буде встановлений як НЕ ПРАВДА, тому що тепер це повноцінна віртуальна машина.

Процес створення віртуальної машини, а саме виконання виклику **VM.provision** може зайняти деякий час, тому що вимагає часу на створення нового віртуального жорсткого диска для віртуальної машини.

Управління віртуальною машиною. Управління віртуальною машиною здійснюється наступними викликами:

• VM.start(session, new_vm_ref) – запуск

• VM.suspend (session, new_vm_ref) - припинення

• VM.resume (session, new_vm_ref) - відновлення роботи

• VM.shutdown (session, new_vm_ref) - повне виключення, аналог виключення живлення

Отримання доступу до консолі віртуальної машини. У більшості випадків для цілей повноцінного управління віртуальною машиною,

необхідно мати доступ до її графічної консолі. Різні віртуальні машини автоматично надає доступ до консолі за допомогою різних технологій.

HVM-віртуальні машини надають графічну консоль за допомогою VNC додатків [38,39]. Для паравіртуальних німи використовується вбудована в Xenутиліта vncterm.

Все реалізації VNC використовують протокол RFB (RemoteFramebuffer). Існує безліч реалізацій вьюверов для VNC-клієнтів, в тому числі для веб-додатків, тому завдання веб-поратал отримати необхідні параметри подклчюенія до VNC- консолі і передати їх спеціалізованому додатком. Мінімальна версія RFB для коректної роботи з сервером - RFB 3.3.

Отримання параметрів для VNC-підключення відбувається за наступним алгоритмом:

1. Запит від клієнта серверу **Session.login_with_password ()**

2. Сервер повертає клієнту ідентифікатор сесії

3. Запит від клієнта серверу: **VM.get_by_name_label ()**

4. Сервер повертає клієнту ідентифікатор конкретної віртуальної машини

5. Запит від клієнта серверу: **VM.get_consoles ()**

6. Сервер повертає клієнту списках асоційованих з віртуальною машиною консолей

7. Запит від клієнта серверу: **VM.get_location ()**

8. Сервер повертає URI консолі в наступному форматі: https: //192.168.0.1/console? Ref = OpaqueRef: c038533a-af99-a0ff-9095- c1159f2dc6a0.

9. Подключеніекліентак 192.168.0.1: HTTP CONNECT "/ console? Ref = (...)"

Після того, як останній запит виконаний, дане підключення може використовуватися як VNCсервер і бути видимим з будь-якого стандартного VNC-вюверів.

Завершення сесії. Після того, як користувач вирішив вийти з веб-поратал, необхідно здійснити вихід з сесії з робочим сервером за допомогою

виклику **Session.logout (session).** Це автоматично інвалідірует ідентифікатор сесії і одновреенно звільняє пам'ять робочого сервера, яка використовувалася для зберігання інформації про об'єкт сесії.

4.11 Висновки

Спроектована система автоматизація надання послуг та забезпечення універсального доступу, з урахуванням зроблених доробок у вигляді реалізації окремого модуля веб-порталу, дозволяє повністю пердоставіть користувачам і адміністраторам системи універсальний доступ до послуг посредвом звичайних браузерів. Так само, портал управління вносить елементи самообслуговування в інфраструктуру віртуальних полігонів - користувачі можуть самостійно управляти виділеними ресурсами, здійснювати необхідні при навчанні операції.

При необхідності, з урахуванням запропонованої методології, можна легко реалізувати більш складну фукнціонального по самоосбслуживанію користувачів, включаючи самостійне конфігурація віртуальних машин.

ВИСНОВОК

У роботі була розглянута задача розробки і реалізації інфраструктури віртуальних тестових середовищ (полігонів) для освітніх цілей на базі технологій хмарних обчислень.

Отримані наступні результати: отримана методологія проектування інфраструктури віртуальних полігонів, досліджені різні аспекти роботи системи і її підсистем - обчислювальної підсистеми (підсистема віртуалізації, гипервизор), мережевої підсистеми, системи зберігання даних, системи автоматизації надання послуг та забезпечення універсального доступу.

Описана реалізація прототипу системи і техніко-технічні характеристики прототипу.

Методологія проектування системи і практичні висновки і пропозиції, що містяться в магістерській дисертації, можуть бути використані освітніми установами для створення власних систем віртуальних полігонів.

СПИСОК ЛІТЕРАТУРИ

1. Michelle Bailey. The Economics of Virtualization: Moving Toward an Application- Based Cost Model. IDC.URL:http://www.vmware.com/files/pdf/Virtualization- application-based-cost-model-WP-EN.pdf

2. A Performance Comparison of Hypervisors. VMware. URL: http://www.cc.iitd.ernet.in/misc/cloud/hypervisor_performance.pdf

3. Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. NIST Special Publication 800-145. URL: http://csrc.nist.gov/publications/nistpubs/800- 145/SP800-145.pdf

4. Chris Wolf, Erick M. Halter. Virtualization: From the Desktop to the Enterprise. Apress. 2005

5. М. Тим Джонс. Узнайте о виртуальной машине ядра Linux (KVM). URL: http://www.ibm.com/developerworks/ru/library/l-linux-kvm/

6. Xen. URL: http://www.cl.cam.ac.uk/research/srg/netos/xen/

7. Xen. URL:http://ru.wikipedia.org/wiki/Xen

8. William von Hagen. Professional Xen Virtualization. Wiley / Wrox. 2008

9. Chris Takemura, Luke S. Crawford. The Book of Xen. A Practical Guide for the System Administrator. No Starch Press. 2009

10. М. ТимДжонс. ВиртуализациясетейвLinux. URL: http://www.ibm.com/developerworks/ru/library/l-virtual-networking/

11. Bridging Network Connections. URL: http://wiki.debian.org/BridgeNetworkConnections

12. Open vSwitch. URL: http://openvswitch.org/

13. Open iSCSI. URL: http://www.open-iscsi.org/

14. Xen and iSCSI. URL: http://old-list-archives.xenproject.org/archives/html/xen- users/2006-01/msg01117.html

15. iSCSI. URL: http://www.ietf.org/rfc/rfc3720.txt

16. Системы хранения данных.URL: http://ru.wikibooks.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

17. T11 - Fibre Channel Interfaces. URL:http://standards.incits.org/a/public/group/t11

18. Fibre Channel. URL: http://fcoe.ru/index.php?lang=russian

19. Fibre Channel. URL:http://ru.wikipedia.org/wiki/Fibre_Channel

20. Network File System Version 4 (nfsv4). URL: http://www.ietf.org/html.charters/nfsv4-charter.html

21. Network File System. URL: http://ru.wikipedia.org/wiki/Network_File_System

22. OpenNebula. URL: http://opennebula.org/

23. OpenStack. URL: http://www.openstack.org/

24. Пярн А.В. «Программное обеспечение с открытым исходным кодом для построения и управления облачными средами на распределенных гетерогенных инфраструктурах». Распределенные вычисления и ГРИД технологии в науке и образовании. Труды 5-ой международной конференции. ДУБНА, 16-21 июля 2012. Издательский отдел Объединенного института ядерных исследований. С. 371-376.

25. UbuntuServer. URL: http://www.ubuntu.com/server

26. Libvirt. The virtualization API. URL: http://libvirt.org/

27. Amazon Web Services. URL: http://aws.amazon.com/

28. Сравнение фунциональностиXCPиXenServerю URL:

http://wiki.xen.org/wiki/XCP/XenServer_Feature_Matrix

29. XAPI. URL:http://wiki.xensource.com/xenwiki/XAPI_Developer_Guide

30. Understanding XenServer Networking – The Linux Perspective. Citrix. 2008

31. Xen Networking.URL: http://wiki.xen.org/wiki/Xen_Networking

32. Citrix XenServer Design: Designing XenServer Network Configurations. Citrix

33. Citrix XenServer 6.0 Administrator's Guide. URL: http://support.citrix.com/article/CTX130420

34. Руководство по установке виртуальной машины CitrixXenServer 5.6. URL: https://support.citrix.com/servlet/KbServlet/download/28081-102-666359/guest.pdf

35. Citrix XenServer 6.1.0 Software Development Kit. URL: http://support.citrix.com/servlet/KbServlet/download/32310-102-691303/

36. Citrix XenServer Management API. URL: http://docs.vmd.citrix.com/XenServer/6.0.0/1.0/en_gb/api/

37. Citrix SDK Downloads. URL: http://community.citrix.com/display/xs/Download+SDKs

38. The RFB Protocol. URL: http://www.realvnc.com/docs/rfbproto.pdf

39. VNC. URL: http://ru.wikipedia.org/wiki/Virtual_Network_Computing

40. XCP ISO. URL: http://www.xen.org/download/xcp/index.html

Додаток А

Перелік посилань

1. Юдін О. К. Аналіз стеганографічних методів приховування інформаційних потоків у контейнери різних форматів / О. К. Юдін, Р. В. Зюбіна, О. В. Фролов // Радиоэлектроника и информатика. — Х. : НХНУРЕ, 2015. — № 3. — С. 24-31.

2. Steganography and Digital Watermarking: a global view [Електроний ресурс] - Режим доступу до ресурсу: http://lia.deis.unibo.it/Courses/RetiDiCalcolatori/Progetti00/fortini/proiect.pdf.

3. LSB стеганография [Електроний ресурс]. - 2019. - Режим доступу до ресурсу: https://habr.eom/ru/post/112976

4. Рейда О.В. Аналіз та дослідження форматних стеганоалгоритмів на основі графічних контейнерів / Рейда О.В., Джулій В.М. // Тези доповідей Всеукраїнської науково-практичної конференції "Інтелектуальний потенціал – 2018". – 2018 – С. 86-90.

5. Cristi Cuturicu, JPEG - Алгоритм стиснення, Code Net [Електронний ресурс] / Формати файлів, - Режим доступу: http://www.codenet.ru/progr/formt/jpeg_00.php

Метод створення віртуальних полігонів на основі технологій хмарних обчислень системи управління базами даних

Джулій В.М., Лукін В.С., Чешун В.М.
Хмельницький національний університет

При виконанні дослідження було поставлено наступні задачі:

– дослідження і вибір існуючих систем, придатних для реалізації цілей і задач цієї роботи;

– проектування апаратно-програмного комплексу, включаючи дослідження і побудову всіх його підсистем;

– створення працюючого прототипу апаратно-програмного комплексу;

– визначення ефективності прототипу.

В рамках поставлених завдань розроблені наступні підсистеми апаратно-програмного комплексу та забезпечено їх взаємодію для виконання цілей цієї роботи:

– обчислювальна підсистема (система віртуалізації);

– мережева підсистема;

– система зберігання даних;

– система автоматизації надання послуг та забезпечення універсального доступу.

Для вирішення поставлених завдань розроблена архітектура інфраструктури віртуальних полігонів.

Робочі сервери віртуалізації [1,2] з встановленим гіпервізором є обчислювальним ядром інфраструктури, що забезпечує створення і управління роботою віртуальних машин, диспетчеризацію розподілу ресурсів між віртуальними машинами. Робочі сервери об'єднуються в ресурсний пул для забезпечення еластичності інфраструктури, можливості масштабувати інфраструктуру прозоро для користувачів. У ресурсному пулі виділяється керуючий master-сервер, за допомогою якого здійснюється централізоване адміністрування як всією обчислювальною системою, так і управління системою зберігання даних і мережевий підсистемою. Так само, master-сервер є центральним інтерфейсом управління всією системою за допомогою наданого їм API.

Обчислювальна підсистема забезпечує наступну функціональність: забезпечення продуктивності всіх компонентів віртуальних машин на рівні продуктивності пристроїв фізичних серверів;

– можливість створення готових шаблонів віртуальних машин з попередніми налаштованими пакетами програмного забезпечення;

– об'єднання фізичних серверів в ресурсні пули для динамічного розподілу віртуальних машин між фізичними серверами;

– можливість динамічної міграції віртуальних машин між серверами;

– підтримка режиму паравіртуалізації для найбільш ефективного використання обчислювальних та інших ресурсів операційними системами сімейства Linux.

Оскільки обчислювальна система є ядром всієї інфраструктури віртуальних полігонів, деякі вимоги до інших підсистем продиктовані функціональними можливостями обраного рішення віртуалізації.

Додавання нових серверів, а також штатне або аварійне вимкнення наявних серверів, відбувається прозоро для адміністратора системи і головне - для користувачів. Операція не вимагає переналаштовування системи і будь-яких додаткових дій з боку людини, не призводить до втрати інформації і збоїв в роботі інфраструктури. При цих діях користувач або взагалі не помічає змін, що відбулися, або перерви в роботі сервісів мінімальні.

Така можливість є суттєвою з точки зору хмарної моделі надання послуг, тому що забезпечує об'єднання ресурсів в ресурсні пули для динамічного перерозподілу потужностей між користувачами в умовах постійної зміни попиту на потужності, функції динамічної міграції віртуальних машин при проведенні технічного обслуговування окремих серверів, балансування навантаження між робочими серверами.

В архітектурі системи дана вимога виражається в необхідності використання зовнішнього сховища даних для зберігання файлів віртуальних машин, шаблонів віртуальних машин, віртуальних жорстких дисків користувачів. Гіпервізор повинен підтримувати роботу не тільки з локальною

системою зберігання даних робочого сервера, але і з зовнішніми системами зберігання даних.

Мережева підсистема забезпечує створення довільних топологій мережевої інфраструктури віртуальних машин, мережеву зв'язаність віртуальних машин як в межах внутрішніх віртуальних мереж між віртуальними машинами, так і відносно зовнішніх до системи мереж і пристроїв, нормальне функціонування віртуальних машин з точки зору мережевих протоколів.

Віртуальні машини, розгорнуті на різних фізичних серверах, мають можливість бути об'єднаними в єдиний віртуальний домен комутації на рівні L2. Для забезпечення вимоги до високої еластичності інфраструктури при реалізації мережевої підсистеми максимально використовуються програмні мережеві рішення – вбудовані можливості гіпервізора і стандартних мережевих засобів операційної системи Linux. З апаратних рішень використовується тільки фізичний комутатор рівня L2 для підключення серверів до локальної обчислювальної мережі.

Технічне управління і адміністрування системою в режимі нормальної експлуатації здійснюється з окремої програми з графічним інтерфейсом користувача, встановленої на керуючому сервері адміністратора системи. Адміністраторів системи може бути будь-яка кількість, так само можливо наділяти користувачів розширеними правами на управління частинами віртуальної інфраструктури. У цій програмі існує можливість створювати всі типові настройки, шаблони віртуальних машин, управляти життєвим циклом віртуальних машин, здійснювати адміністрування віртуальних машин, в тому числі в режимі консолі.

Конфігурація робочих серверів віртуалізації так само має здійснюватися підключенням до них безпосередньо по SSH через інтерфейс командного рядка (CLI). Звичайні користувачі отримують параметри доступу до віртуальних машин, а так доступ до консолі віртуальної машини через веб-сервер (портал) управління. Портал доступний як всередині локальної мережі організації, так і через мережу Інтернет.

Веб-сервер написаний на мові програмування Java і релизована на стандартній платформі, контйнері сервлетів, що підтримує функціонал віддалених процедур XML-RPC. Необхідний фунцонал веб-сервера забезпечується з допомогою APImaster-сервера.

Стосовно запропонованих рішень проведене дослідження підсистем інфраструктури віртуальних полігонів і їх взаємодії для того, щоб прийти до найбільш оптимальної схеми побудови всієї системи.

Проведені дослідження стали основою для створення віртуальних полігонів на основі технологій хмарних обчислень системи управління базами даних.

Перелік посилань
1. Michelle Bailey. The Economics of Virtualization: Moving Toward an Application- Based Cost Model. IDC.URL: http://www.vmware.com/files/pdf/Virtualization-
2. Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. NIST Special Publication 800-145. URL: http://csrc.nist.gov/publications/nistpubs/800- 145/SP800-145.pdf

Метод захисту від загрозливих програм, заснований на реалізації контролю доступу до файлових об'єктів

Казіміров В.О., Мостовий С.В., Орленко В.С.
Хмельницький національний університет

Використання сучасних систем інформаційної безпеки вимагає, з одного боку, відстеження швидких змін в інформаційних технологіях та нових загроз, а з іншого - з урахуванням реальних характеристик апаратного та програмного забезпечення корпоративних мереж та систем. Процедура придбання пристроїв захисту інформації проста. Набагато складніше вирішити проблему - як захистити і які заходи безпеки застосовувати, мінімізуючи витрати. Впроваджуючи різні засоби захисту, необхідно визначити баланс між можливим збитком від несанкціонованого витоку інформації та обсягом інвестицій, які витрачаються на забезпечення безпеки інформаційних ресурсів. З метою підвищення ефективності захисту інформаційних ресурсів необхідно дослідити підходи до оцінки рівня їх захисту та систем захисту. Ця оцінка для кожного випадку індивідуальна і залежить від багатьох факторів (вартості інформації, статусу організації, важливості інформації, рівня технічного та програмного забезпечення тощо).

В роботі здійснено дослідження основних типів загрозливих програм, та запропоновано класифікацію шкідливого програмного забезпеченні (ШПЗ) за способом їх виконання. Враховуючи аналіз існуючої статистики зроблено висновок, що найбільш актуальними для захисту є виконувані двійкові і файли сценаріїв.

Можна виділити два найбільш поширених способи зараження: соціальна інженерія; технічні прийоми впровадження ШПЗ, що заражається без відома користувачf [1].

Ці види ШПЗ передбачають обов'язкове збереження файлу на вінчестері перед виконанням.

Тому можна зробити висновок що застосування розмежувальної політики доступу до виконуваних об'єктів, дозволяє мінімізувати загрози.

Проведено дослідження існуючих підходів до оцінки ефективності методів і засобів захисту від загрозливих програм, в результаті якого зроблені

Додаток Б

Презентація

Метод створення віртуальних полігонів на основі технологій хмарних обчислень

• Виконав
• Студент гр. КІ1м-19-1
• Лукін В.С.

• Керівник
• Чешун В.М.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА МАГІСТЕРСЬКОЇ РОБОТИ

Метою роботи є розроблення методу побудови віртуальних полігонів

Об'єкт дослідження – технології організації хмарних обчислень.
Предмет дослідження – використання віртуальних полігонів при організації хмарних обчислень.

Задачі дослідження:

1) Провести дослідження існуючих моделей, методів, а також алгоритмів використовуваних в побудові хмар.
2) Розробити метод створення віртуальних полігонів.

Методи дослідження базуються на основних положеннях методів аналізу даних, нечіткої логіки, теорії графів, теорії множин.

Наукова новизна одержаних результатів:

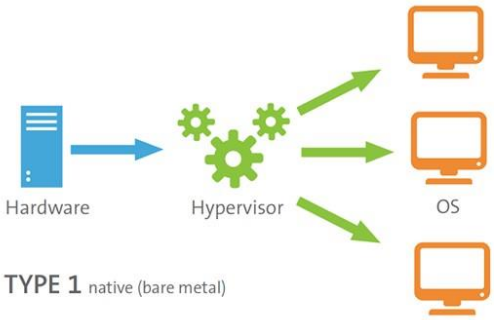
1. Архітектура інфраструктури віртуальних полігонів, що дозволяє реалізувати виконання поставленого завдання.
2. Дістав подальшого розвитку метод створення віртуальних полігонів на основі технологій хмарних обчислень, що дозволяє використовувати хмарні ресурси для задач антивірусного діагностування.

Апробація роботи. Наукові результати і основні положення кваліфікаційної роботи магістра доповідались і обговорювались на всеукраїнській науково-практичній конференції.

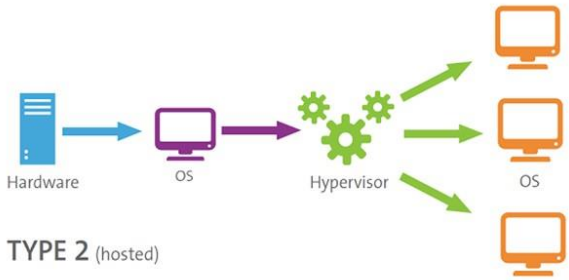
Публікації. За темою кваліфікаційної роботи опубліковано 1 стаття у збірнику наукових праць.

Завдання магістерської роботи:

Тип гіпервізора



Бареметальна архітектура



Розміщена архітектура

Найбільш популярні на сьогоднішній день реалізації гіпервізора

Гіпервізор KVM

Переваги KVM:

KVM є частиною стандартного ядра Linux, робота з гіпервізором багато в чому визначається вмінням працювати з ядром операційної системи Linux. Оскільки модуль є частиною ядра, технологія отримує значні переваги від розвитку і оптимізації ядра.

Недоліки KVM:

- Працює тільки в режимі повної віртуалізації, вимагає підтримки процесорами режиму апаратної віртуалізацій;
- Оскільки технологія є надбудовою над ОС, код гіпервізора є кодом всієї операційної системи, а значить, має широкий профіль розломів інформаційної безпеки;
- Використання процесу QEMU в просторі користувача для забезпечення віртуалізації введення/виведення значно впливає на продуктивність підсистеми введення/виведення;
- Не має графічних інтерфейсів користувача під ОС Windows

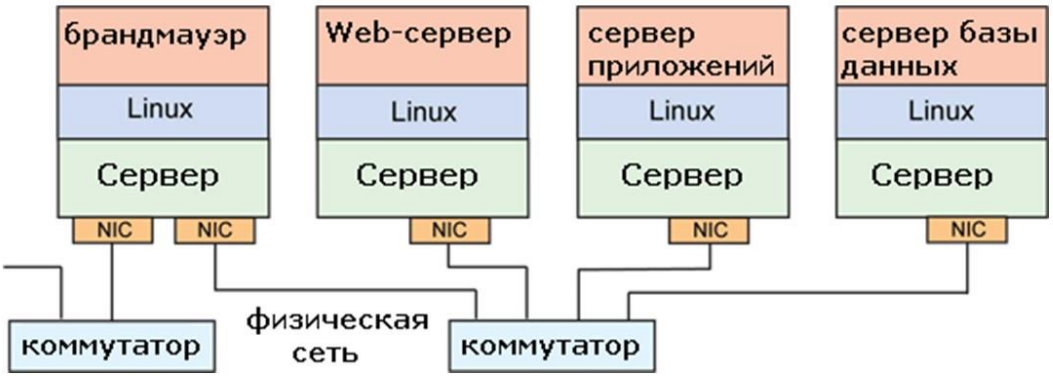
Найбільш популярні на сьогоднішній день реалізації гіпервізора

Гіпервізор Xen

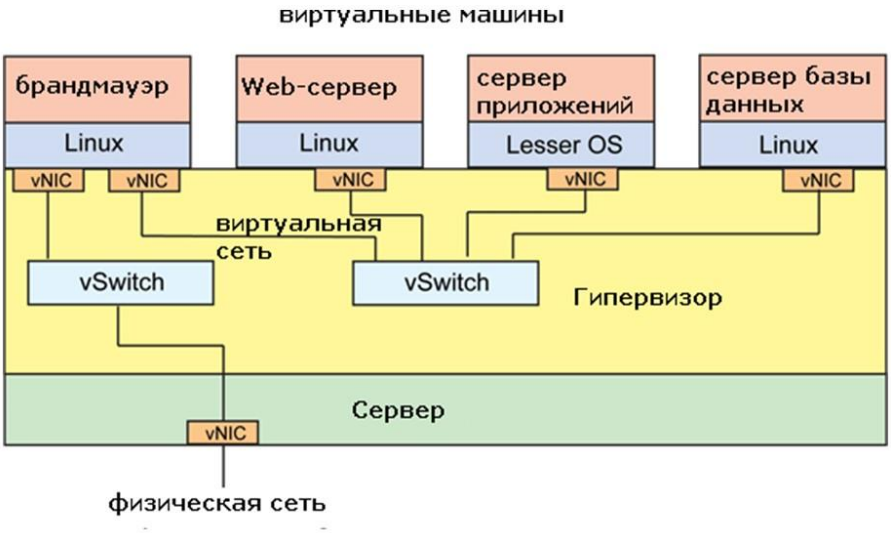
Переваги Xen:

- Якнайкраще забезпечення інформаційної безпеки серед гіпервізора;
- Стабільність роботи гіпервізора;
- Функціональні можливості Xen: можливість створювати ресурсні пули, проводити динамічну міграцію віртуальних машин, балансування навантаження між хостами, наявність графічного інтерфейсу управління під Windows;
- Наявність графічного інтерфейсу управління під Windows.

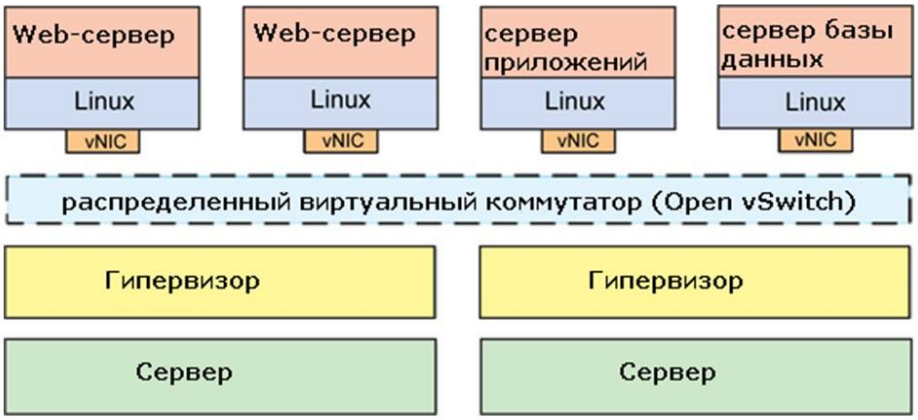
Фізична мережева інфраструктура



Віртуалізована мережева інфраструктура



Розподілений віртуальний комутатор



Протокол передачі даних iSCSI

Переваги:

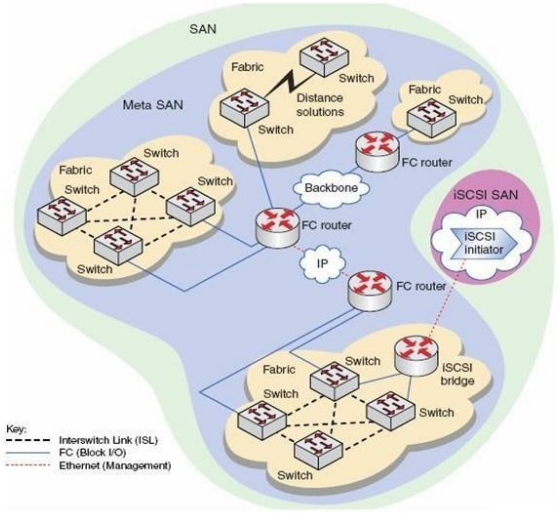
- Вартість, можливість використовувати наявну мережеву інфраструктуру.
- Програмний модуль для використання iSCSI є для більшості існуючих сьогодні на ринку OS (MSWindows, Solaris, Linux, AIX і т.д).
- Простота настройки і використання.

Оскільки носієм iSCSI є IP-протокол, можливе створення систем зберігання даних, рознесених на великі відстані і знаходяться в різних дата центрах, підключених один з одним через загальні канали зв'язку Інтернет.

Недоліки:

- iSCSI чутливий до перенавантаження мережі і сам сильно завантажує мережу передачі даних
- Через це, все ж потрібна побудова ізольованої IP-мережі для мережі зберігання даних iSCSI.
- iSCSI може використовувати програмний модуль, який дозволяє здійснює розглянуту вище інкапсуляцію і декапсуляцію SCSI в IP (англ. - initiator).Однак, як будь-яке програмне рішення, це споживає певну кількість процесорної потужності.

Протокол FibreChannel або FC



Приклад реалізації мережі зберігання даних FibreChannel

NetworkFileSystem (NFS)

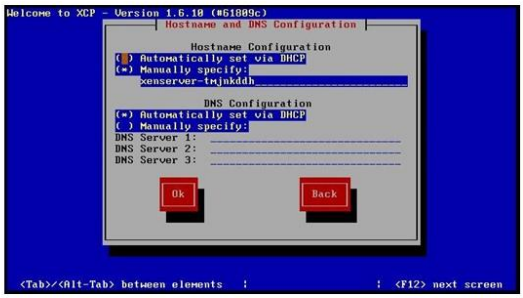
Переваги використання системи зберігання даних, побудованої на NFS:

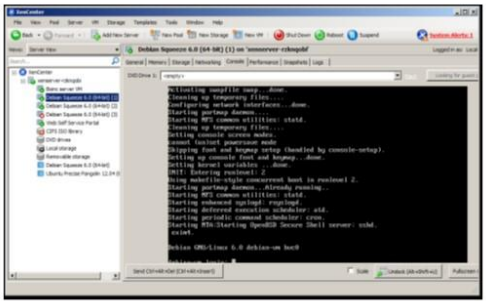
- Широка поширеність і простота настройки. Більшість Linux операційнних систем може функціонувати як NFS-сервер;
- Файлова система вже має вбудовані функції, необхідні для коректної роботи обчислювальної підсистеми, такі як множинний доступ хостів і розподілене блокування доступу до файлів
- NFS працює поверх стандартної мережі передачі даних Ethernet, використовуючи IP протокол
- Віртуальні машини, шаблони віртуальних машин, віртуальні жорсткі диски являють собою файли. Оскільки NFS протокол оптимізований під роботу зі звичайними файлами, він добре сумісний з усіма системами віртуалізації.

Недоліки:

Проблеми з продуктивністю. Оскільки реалізація протоколу програмна, файлу потрібне обчислювальні потужності для обробки запити клієнта. Протягом часу, поки сервер обробляє запит клієнта, сервер не блокує запити від інших клієнтів, які також повинні бути обслужені.

Початкове налаштування робочих серверів





Техніко-технічні параметри системи

Загальний обсяг віртуальної пам'яті на двох серверах - 8 Гб.

Оптимальна кількість віртуальних машин, яке можна одночасно запустити на інфраструктурі прототипу: 15 віртуальних машин з розміром оперативної пам'яті близько 500 МБ на віртуальну машину.

За рахунок використання технології динамічного розподілу пам'яті, кількість віртуальних машин можна збільшити до 25.

При використанні локальних дисків робочих серверів в якості системи зберігання даних, параметри роботи віртуальних машин такі:

• Час запуску однієї віртуальної машини з зупиненого стану: менше 10 секунд.

• Час створення віртуальної машини з шаблона віртуальної машини: менше 10 секунд.

ДЯКУЮ ЗА УВАГУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 13%**

ID: 82190 Название: Метод створення віртуальних полігонів на основі технологій хмарних обчислень Добавлено в БД: 2020-12-02 Авторы: Лукін В.С. Руководители: Чешун В.М. Консультанты: Опоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	125831	881	48 (0%)	1 (0%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

User name:
Кафедра кибербезпеки

Check ID:
1005428831

Check date:
11.12.2020 06:48:56 EET

Check type:
Doc vs Internet

Report date:
11.12.2020 06:49:32 EET

User ID:
100005590

File name: **дип_робота_Лукін**

Page count: **92** Word count: **17051** Character count: **138231** File size: **1.62 MB** File ID: **1005720229**

7.34% Matches

Highest match: **4.24%** with Internet source (http://master.cmc.msu.ru/files/master2013_1_piarn.pdf)

7.34% Internet sources | 387 ..... Page 94

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters | 309

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод створення віртуальних полігонів на основі технологій хмарних обчислень

Автор: Лукін В.С.

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: Програмування та захист комп'ютерних систем і мереж

Науковий керівник: Чешун Віктор Миколайович, к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 7.24% і адресується до 387 першоджерел, усі запозичення фрагментарні або мають належним чином оформленні посилання, що відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи як оригінального тексту.

Керівник роботи В.М. Чешун

Завідувач кафедри КбКСМ, гарант ОП Ю.П. Кльоц

Дата: 11.12.2020