ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерної інженерії

Пояснювальна записка
до магістерської роботи

на ступінь вищої освіти магістр

на тему: «МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ З ВИКОРИСТАННЯМ ХМАРНИХ ТЕХНОЛОГІЙ»

Виконав: студент 6 курсу, групи КСДМ-61

спеціальності **123** Комп'ютерна інженерія

(шифр і назва спеціальності)

Харченко В. В.

(прізвище та ініціали)

Керівник Лемешко А. В.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»
Напрям підготовки **123** «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ О.М.Ткаченко
"____" _____2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Харченку В'ячеславу Віталійовичу

(прізвище, ім'я, по батькові)

1.Тема роботи Моделювання та проектування комп'ютерних мереж з використанням хмарних технологій

Керівник роботи доктор філософії Лемешко А.В.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "__" __ 2021 року № _____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи:

3.1 Технічна документація по хмарним платформам ;

3.2 Інтернет-ресурси провайдерів сервісів;
3.3 Web- інтерфейс для користувача хмари;
3.4 Програмний інтерфейс для інтеграції програмних продуктів;

3.5 Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити:

4.1 Загальна характеристика хмарних технологій;

4.2 Системи авторизації хмарних технологій;

4.3 Дослідження системи управління у багатопроцесорних серверах;

4.4 Налаштування програмного забезпечення OPENFOAM у розподіленому обчислювальному середовищі.

5.Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

5.1 Назва роботи;
5.2 Мета роботи;
5.3 Моделі розгортання хмар;

5.4 Основні моделі обслуговування хмарних обчислень;
5.5 Переваги та недоліки моделей сервісів хмарних технологій;
5.6 Межі керованості
5.7 Схема мережі ЦОД;
5.8 Web-консолі ElasticFox;
5.9 Порівняння кількох платформ хмарних обчислень;
5.10 Аналіз експериментальних даних;
5.11 Схема технологій Mosix;
5.12 Загальна схема взаємодії компонентів Globus Toolkit;
5.13 Архітектура процесора UltraSPARC T1;
5.14 Компоненти Univa Grid Engine;
5.15 Компоненти для забезпечення захисту повідомлень;
5.16 Результати матричного множення для матриць різних розмірів;
5.17 Перегляд результатів у ParaView;
5. 18 Висновки;
5.19 Список публікацій.

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури		Виконано
2.	Загальна характеристика хмарних технологій		Виконано
3.	Системи авторизації хмарних обчислень		Виконано
4.	Дослідження системи управління у багатопроцесорних серверах		Виконано
5.	Налаштування програмного забезпечення OPENFOAM		Виконано
6.	Аналіз отриманих результатів дипломної роботи. Вступ, висновки, реферат		Виконано
7.	Розробка демонстраційного матеріалу		Виконано

Студент _____ Харченко В. В. _____

(підпис)

(прізвище та ініціали)

Керівник роботи _____ Лемешко А. В. _____

(підпис)

(прізвище та ініціали)

ЗМІСТ

Стор.

ВСТУП..... 9

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ХМАРНИХ ТЕХНОЛОГІЙ..... 12

1.1 Історія розвитку хмарних технологій…… 12

1.2 Огляд ринку «хмарних послуг» ……. 15

1.3 Сутність поняття хмарних технологій…….…….... 18

1.4 Види хмар та моделі хмарних сервісів…….. 20

1.5 Основні принципи та системи реалізації хмарних технологій..... 27

1.5.1 Особливості хмарних обчислень…….. 29

1.5.2 Виклики хмарних обчислень……. 29

1.5.3 Проблеми доступу користувачів та аналіз продуктів для запуску додатків у розподіленому обчислювальному середовищі……. 30

1.6 Мережевий трафік в центрах обробки даних…… 36

2 СИСТЕМИ АВТОРИЗАЦІЇ ХМАРНИХ ОБЧИСЛЕНЬ..... 38

2.1 Хмарна платформа (EUCALYPTUS) ……..... 40

2.2 Хмарна платформа (Abicloud)……..……... 47

2.3 Хмарна платформа (Nimbus)……..…….. 49

2.4 Аналіз безпеки програмного інтерфейсу API для управління ресурсами та сервісами хмарних обчислень..... 54

2.5 Дослідження продуктивності метаком'ютера з єдиним образом операційної системи……..…….. 56

2.6 Дослідження продуктивності MPI з і без міграції процесів MOSIC у віртуальному середовищі…….. 59

2.7 Аналіз продукту Globus Toolkit для організації системи доступу користувачів для розподіленого обчислювального середовища ……... 61

2.8 GSI – для забезпечення єдиного входу в Грід-систему……..…… 63

2.8.1 Призначення GRIDFTP – керування даними..... 65

2.8.2 Призначення GRAM – керування процесами..... 65

3 ДОСЛІДЖЕННЯ СИСТЕМИ УПРАВЛІННЯ У БАГАТОПРОЦЕСОРНИХ СЕРВЕРАХ..... 67

3.1 Конфігурація Univa Grid Engine ……. 68

3.1.1 Час виконання (секунди)…… 72

3.2 Проектування системи доступу користувачів та розробка технічних принципів запуску ресурсомістких додатків …….…… 74

3.2.1 Розробка системи безпеки Гріду в систему хмарних обчислень.... 75

3.2.2 Програмний інтерфейс DRMAA для інтеграції програмних продуктів……... 77

3.2.3 NAS Parallel Bencmarks……... 78

3.2.4 OpenFOAM……... 79

4 НАЛАШТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ OPENFOAM У РОЗПОДІЛЕНОМУ ОБЧИСЛЮВАЛЬНОМУ СЕРЕДОВИЩІ…….. 81
ВИСНОВКИ..... 86
ПЕРЕЛІК ПОСИЛАНЬ..... 87
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ..... 90

ВСТУП

Хмарні технології – це обчислювальна модель, у якій ресурси, такі як обчислювальні потужності, системи зберігання, мережі та програмне забезпечення, абстраговані та забезпечені як служби в Інтернет для віддаленого користувача. На його вимогу забезпечуються доступність виділення ресурсів, динамічна та фактично нескінченна масштабованість під час вирішення конкретних завдань. Переваги технології хмарних обчислень включають високу продуктивність, економію у витратах, високий ступінь доступності талегку масштабованість.

Все більш тісний зв'язок процесів у сфері інформаційно-комунікаційних технологій, стрімкий розвиток глобальних інформаційних та обчислювальних мереж та низка інших факторів ведуть до зміни фундаментальних парадигм обробки інформації внаслідок необхідності підтримки та розвитку розподілених інформаційно-обчислювальних ресурсів, технології використання яких отримують все більший пріоритет у сучасному інформаційному суспільстві. У цих умовах хмарні обчислення як етап і нова концепція в еволюції Інтернету дозволяють надання засобів користувачеві, коли може бути доставлена будь-яка послуга в будь-якому місці і коли це необхідно.

На сьогоднішній день багато експертів сходяться поглядами, що «хмара» по своїм можливостям перевершить Інтернет: обсяг світового ринку хмарних сервісів збільшується стабільними темпами. За останні кілька років концепція хмарних обчислень та віртуалізації набула чинності та стала популярною у сфері інформаційних технологій; багато організацій розпочали реалізацію цих технологій, прагнучи знизити витрати за рахунок покращеної віртуалізації машин, меншого часу на адміністрування та зниження витрат на інфраструктуру. З цих позицій відмічається актуальність дослідження інфраструктури хмарних обчислень, з розглядом завдань, пов'язаних із забезпеченням хмарних послуг.

Прогрес у розвитку сучасних інформаційних технологій полягає у прагненні задовольнити ресурсомісткі та зростаючі вимоги споживачів, які потребують обробки гігантських обсягів інформації, з абсолютно різних областей.

Для задоволення потреб творці інформаційних технологій як вирішення поставлених завдань суттєвого збільшення продуктивності засобів обробки інформації розробили та запропонували розподілені обчислення у вигляді Грід-технологій та їх подальшого розвитку у так званих хмарних обчисленнях. При організації хмарних обчислень постачальник послуг об'єднує ресурси обслуговування великої кількості споживачів на єдиний пул при динамічному перерозподілі потужностей між споживачами за умов постійного зміни попиту використовувані потужності.

З погляду постачальника сервісів, завдяки об'єднанню ресурсів та непостійному характеру споживання з боку споживачів, хмарні обчислення дозволяють заощаджувати на масштабах, використовуючи менші апаратні ресурси, ніж були б потрібні при виділених апаратних потужностях для кожного споживача, а за рахунок автоматизації процедур модифікації виділення ресурсів, застосування принципів віртуалізації суттєво знижуються витрати на абонентське обслуговування.

Однак, при її практичній реалізації виникає ціла низка ще невирішених наукових проблем, що перешкоджають повноцінному використанню всіх потенційних переваг такого підходу.

По-перше, прагнення створити універсальну хмарну систему неминуче стикається з необхідністю працювати в гетерогенному середовищі та відповідно організовувати доступ користувачів із їх індивідуальними додатками без зниження продуктивності.

По-друге, у хмарних середовищах при організації доступу довільного числа користувачів до цих пір представляє проблему забезпечення високого ступеня безпеки та надійності збереження індивідуальних даних. Тому проблема безпеки даних та ресурсів при хмарних технологіях є однією з критичних проблем. Безпека всієї системи залежить від безпеки програмних інтерфейсів для керування ресурсами, віртуальними машинами та сервісами. Починаючи від процедури аутентифікації та авторизації та закінчуючи шифруванням, програмні інтерфейси повинні забезпечувати максимальний рівень захисту від несанкціонованого

доступу. При забезпеченні безпеки у хмарах необхідно передбачити проведення зручного та одноманітного авторизованого доступу до ресурсів, обліку їх використання та захист ресурсів та даних від несанкціонованого використання.

По-третє, для забезпечення можливості практичного використання хмарного середовища в різних областях необхідно організувати універсальну систему запуску індивідуальних додатків.

Зважаючи на низку вище наведених невирішених проблем, тема магістерської роботи є досить актуальною.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ХМАРНИХ ТЕХНОЛОГІЙ

1.1 Історія розвитку хмарних технологій

Початком розвитку хмарних технологій у IT-індустрії прийнято вважати 50-ті роки минулого століття. Тоді, внаслідок дорожнечі комп'ютерів, вчені вигадали використовувати одну обчислювальну машину, доступ до якої мали одразу кілька співробітників тієї чи іншої компанії. Ідея про підключення низки користувачів до загального процесора з'явилася в 1954 році, реалізація почалася в 1959, а перше комерційно успішне рішення випустили в 1964. Тоді ж сформувалося ставлення до обчислювальних потужностей як ресурсу, відкрилися спеціальні комп'ютерні бюро: у них клієнти могли купувати необхідний обсяг потужності до виконання розрахунків.

Аж до 80-х років подібна модель надання потужностей була життєздатною, але потім поступово пішла в минуле, оскільки на ринку з'явилися відносно недорогі персональні комп'ютери.

У 1966 році виник також проект ARPANET, за створення якого відповідав вчений Джозеф Ліклайдер. Його ідея полягала в тому, щоб усі люди з різних кінців Землі були взаємопов'язані та могли отримувати доступ до програм та даних із будь-якої точки світу. Він заклав основу для ґрід-обчислень, раннього попередника хмари, в яких географічно розподілені комп'ютери були об'єднані для створення слабкої мережі. Саме ядро цього проекту на початку 1990-х еволюціонувало до сучасного інтернету. Іншим важливим фактором, що передвіщало появу «хмар», став розвиток систем віртуалізації. Йдеться про цифрові системи, які не залежать від конкретного обладнання та дозволяють починати та закінчувати роботу в будь-який момент. Комерційний варіант подібної технології випустила компанія IBM у 1972 році. На рисунку 1.1 представлено географічну карту розповсюдження проекту ARPANET.

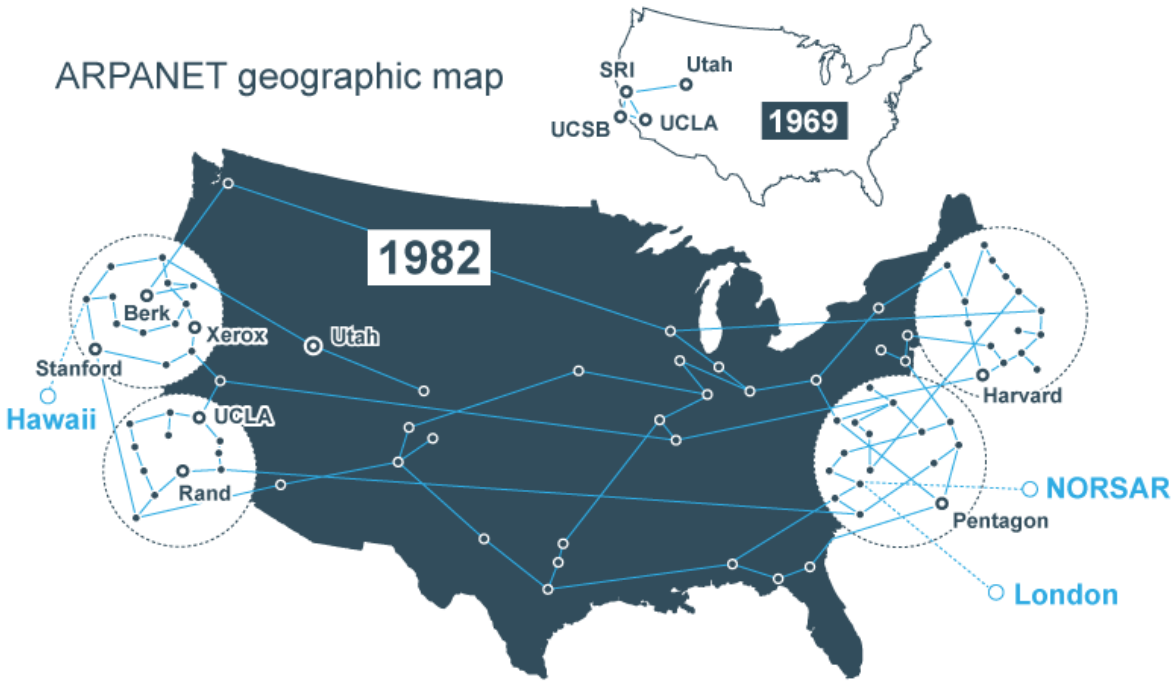


Рисунок 1.1 Географічна карта розповсюдження проекту ARPANET

Так чи інакше, активніший розвиток концепції хмарних технологій почався вже в 90-ті роки, зі значним збільшенням пропускної спроможності мережі Інтернет. Одним із піонерів майбутньої хмарної революції стала компанія Salesforce.com. Її фахівці розробили концепцію доставки корпоративних програм через простий веб-сайт. Ця компанія спочатку була створена з розрахунком на надання CRM-систем клієнтам як послуги з передплати (SaaS).

Як переваги нової моделі представники Salesforce.com називали: можливість аутсорсингу інформаційних технологій, захист від збоїв та оперативну технічну підтримку (оскільки всі апаратні ресурси розміщуються в зоні фізичної доступності постачальника), зниження сукупної вартості володіння інформаційними технологіями. Тобто багато в чому компанія виступила провісником відомих сьогодні хмарних моделей надання послуг. Salesforce.com є одним з найбільших гравців на ринку SaaS поряд з Microsoft, Oracle і SAP.

Поступово SaaS став популярною послугою – за допомогою цієї абревіатури онлайн-сервіси розрізняли серед десктопних програм, що вимагають установки на комп'ютер.

Іншим важливим етапом у розвитку SaaS можна назвати 2009 рік, коли Google та інші великі розробники стали пропонувати програми на базі браузера (йдеться про Google Apps). Приблизно в цей час Microsoft закріпила себе на ринку бізнес-додатків, вклавши сили в розвиток хмарної версії Office 365.

Стрімкий розвиток інтернету призвело до регулярного зростання кількості розробників програмного забезпечення. Процес розміщення нових програм потрібно спростити. Так на ринку, крім SaaS, з'явилася послуга PaaS (Platform as a Service — «платформа як послуга»). Першим подібним сервісом у 2006 році став Zimki. У 2008 році Google представила App Engine, який пізніше став хмарною платформою Google.

Ще на початку 2000-х років корпорації мали у своєму розпорядженні значні обсяги обчислювальних ресурсів, частина яких практично не була задіяна і виступала резервом на випадки пікових навантажень (наприклад, «чорну п'ятницю» у випадку з онлайн-продавцями). Бізнес почав передавати свої потужності третім особам – так з'явилася модель IaaS (Infrastructure as a Service – «інфраструктура як послуга»).

Першим IaaS-сервісом (або хмарою у сучасному розумінні цього терміна) став Amazon Web Services, запущений у 2002 році. Компанія надала пакет хмарних інфраструктурних послуг, включаючи зберігання, обчислення та навіть можливості людського інтелекту через Amazon Mechanical Turk. У 2006 році Amazon запустив Elastic Compute Cloud (EC2) - комерційний веб-сервіс, який дозволяв невеликим компаніям та окремим особам орендувати частину IT-інфраструктури, на якій можна запускати будь-які програми.

На сьогоднішній день Amazon залишається лідером у сфері хмарних послуг за моделлю IaaS. Згідно з аналітикою Gartner за 2020 рік, багато підприємств зараз витрачають понад п'ять мільйонів доларів на рік зі свого IT-бюджету на хмарні сервіси Amazon.

Поступово до Amazon на ринку IaaS приєдналися Microsoft (сервіс Azure, 2010) і Google (Google Compute Engine, 2012). Таким чином, на довгі роки

сформувалася трійка лідерів у галузі надання сервісів за моделлю «інфраструктура як послуга».

Інші постачальники послуг, спостерігаючи діяльність гігантів ринку, намагалися відповідним чином змінити свої продукти та скоригувати бізнес-стратегії. Слідувати «великій трійці» намагалися, наприклад, HPE, Dell та VMware. Частина подібних компаній згодом відмовилася від спроб надання публічної хмари, не витримавши конкурентного тиску. У той же час VMware і Rackspace обрали дещо інший маршрут, позиціонуючи себе як організації, які можуть допомогти підприємствам керувати програмами та робочими навантаженнями у публічних хмарах своїх конкурентів. Так, діяльність Rackspace змістилася у сферу допомоги підприємствам у керуванні своїми хмарними розгортаннями на платформах Amazon, Microsoft, Google та інших. VMware, залишаючись одним з лідерів на ринку програмного забезпечення для віртуалізації, продала свій публічний хмарний бізнес французькому IaaS-конкуренту, OVH, у квітні 2017 року. В даний час VMware позиціонує себе як постачальник гібридних хмарних обчислень.

Що стосується загального стану справ на ринку, розвиток апаратного забезпечення (а саме створення багатоядерних процесорів та збільшення ємності накопичувачів інформації) та технологій віртуалізації (зокрема програмного забезпечення для створення віртуальної інфраструктури, наприклад, Xen-віртуалізація) сприяло не тільки розвитку, а й більшу доступність хмарних технологій. Сьогодні хмарні обчислення це те, чим майже кожен користується щодня. За даними Citrix та IDC, понад 90% компаній у всьому світі орієнтовані на використання хмарних технологій. Згідно з прогнозами Gartner, 2022 р. ринок покаже зростання ще на 16,5%.

1.2 Огляд ринку «хмарних послуг»

Багато аналітичних компаній зазначають, що за останні роки тема хмарних технологій була однією з найактуальніших. За даними компанії International Data

Corporation (IDC), що займається аналітикою, ринок SaaS склав у світі близько 20 млрд. доларів у 2018 р. і виріс до 32 млрд. доларів. в 2020 р. Для того, щоб зрозуміти, які програми та компанії розвиваються більш активно на ринку SaaS-сервісів, корисно звернутися до досліджень Forrester, що розглядав стратегії вендорів, які пропонують програмне забезпечення у хмарному середовищі для колективної роботи.

Це дослідження компанією Forrester проведено шляхом опитування осіб, які приймали рішення про купівлю програмного забезпечення на підприємствах, які планують придбати програмне забезпечення для колективної роботи. Більшість респондентів заявила, що їх підприємство використовуватиме SaaS як доповнення існуючого рішення або заміни. До групи аналізованих Forrester компаній увійшли 8 постачальників: Cisco Systems, Box, Citrix Online, IBM, Google, salesforce.com, Microsoft та Yammer.

До програм (ринок США), які мігрують в «хмару» можна віднести програми для спільної роботи, програмне забезпечення для роботи з контентом, CRM-додатки (управління відносинами з клієнтами), інженерні програми, ERM-додатки (управління ризиками). Оскільки ринок хмарних сервісів США займає понад 60% від світового, то тенденції вважатимуться близькими із загальносвітовими. Оцінки українського SaaS ринку можна проаналізувати у публікаціях CNews, IDC, Parallels та інших компаній. Використовуючи категорії ринку «хмарних» послуг, можна проаналізувати темпи зростання ринку аналогічних рішень IT. Таке порівняння показує високий темп зростання ринку «хмарних» обчислень щодо ринку традиційних IT-рішень. Це пояснює таку сильну увагу до «хмарної» моделі з боку головних IT-вендорів.

У квітні 2014 року аналітична компанія Forrester Research опублікувала прогноз розвитку ринку публічних хмарних обчислень до 2022 р. Згідно з даними звіту, до 2022 р. обсяг ринку хмарних послуг становитиме $160 млрд.

До 2019 р. понад 75% усіх витрат підприємств на IT пов'язані з хмарними технологіями, понад 70% менеджерів з інформатизації перехід до хмарних рішень вважає стратегічним завданням номер один, і понад 80% рішень щодо вибору

хмарних IT-сервісів приймаються спільно з керівниками бізнесу. Зростання хмарного ринку в Україні випереджає загальносвітовий рівень: вітчизняний ринок хмарних послуг за прогнозами IDC зростатиме набагато швидше, ніж IT-ринок загалом.

Світовий ринок хмарних технологій невпинно та стрімко зростає. Згідно з результатами, наданими експертами Canalys, за підсумками третьої чверті 2020 року обсяг споживання в грошовому еквіваленті досяг 36,5 млрд дол, що на 33% більше в річному порівнянні.

Цей результат на 2,0 млрд вищий, ніж в попередньому кварталі і на 9,0 млрд вищий, ніж в минулому році. Підвищений попит на хмарні технології і послуги спостерігався у всіх великих секторах економіки, однаково стабільно підвищувались урядові потреби, витрати корпоративних замовників і споживачів.

На рисунку 1.2 представлено динаміку зростання попиту на послуги хмарної інфраструктури.

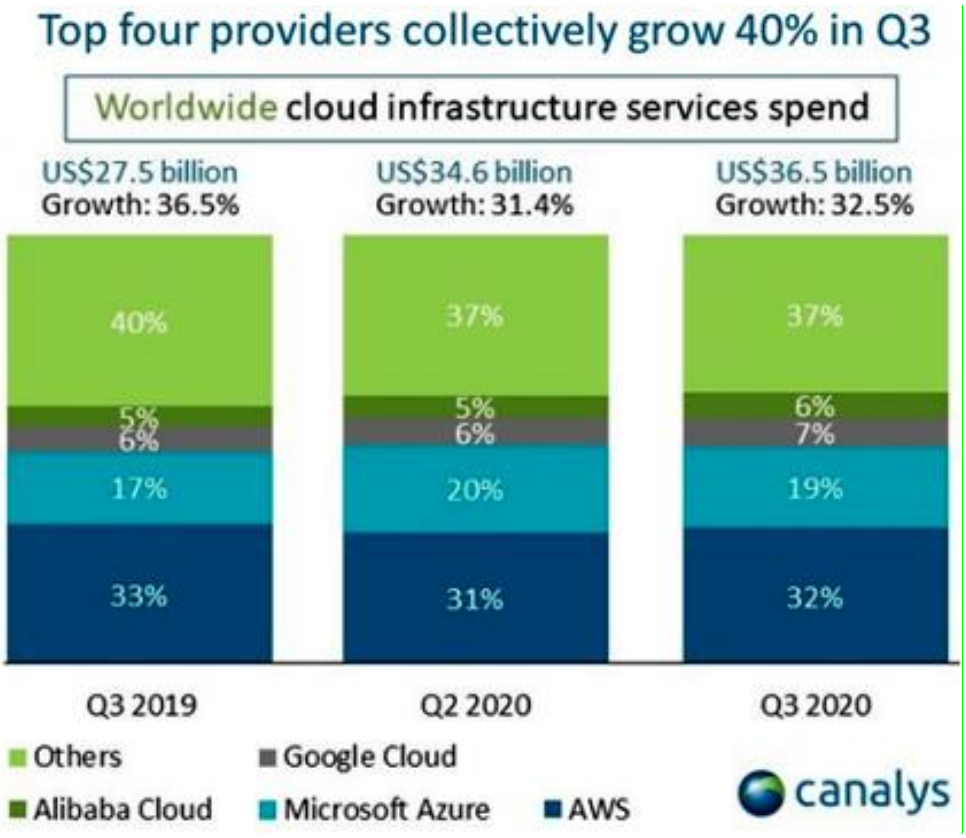


Рисунок 1.2 Динаміка зростання попиту на послуги хмарної інфраструктури

1.3. Сутність поняття хмарних технологій

Ідея хмарних технологій полягає в тому, що компанії, які використовують IT-послуги, можуть купувати ці послуги як послуги. Замість того, щоб купувати сервери для підтримки внутрішніх або зовнішніх сервісів і придбати ліцензії на програмне забезпечення, компанія може купити їх як сервіс.

Хмарні технології є комплексним рішенням, яке надає IT-ресурси у вигляді сервісів. Це рішення ґрунтується на інтернет-технологіях. У хмарі комп'ютери налаштовані на спільну роботу, а різні програми використовують загальну обчислювальну потужність, начебто виконуються у єдиній системі. Гнучкість використання хмарних технологій залежить від розподілу ресурсів за заявленими вимогами.

Такий розподіл може дозволити використовувати загальні ресурси системи без виділення будь-яких апаратних ресурсів для завдання. До хмарних обчислень серверні програми та Web-сайти виконувались на взятих окремо системах.

З появою хмарних технологій ресурси почали використовувати як єдиний віртуальний комп'ютер. Об'єднана таким чином конфігурація є середовищем, додатки в якому виконуються без прив'язки до конкретної конфігурації, тобто незалежно.

Завдяки хмарним сервісам стає можливим розгортання віртуальних робочих місць, створення корпоративних поштових серверів, в "хмару" можна перенести будь-який бізнес-процес, пов’язаний з IT-складовою підприємництва.

На рисунку 1.3 представлено інтеграцію різних видів послуг в хмарному середовищі.

Рисунок 1.3 Інтеграція різних видів послуг в хмарному середовищі

Ключові характеристики хмарних обчислень:

1. Користувач не повинен знати (і купувати) повний обсяг ресурсів, які можуть знадобитися під час пікових навантажень. Хмарні обчислення дають змогу масштабувати ресурси, доступні додатку. Фірма, яка тільки-но розпочала бізнес, не повинна турбуватися про те, що її рекламна кампанія спрацює надто добре і сервери будуть перевантажені.

2. Споживачі платять лише те, що використовують. Їм не потрібно купувати сервери або ресурси в обсязі, що відповідає їх максимальним потребам. Часто це заощаджує кошти.

3. Хмарний сервіс автоматично (або в деяких випадках у напівавтоматичному режимі) виділяє та звільняє на вимогу ресурси процесорів, дискового простору та пропускної спроможності мережі. При малому числі користувачів сервісу підтримки його роботи використовується дуже мало ресурсів, і навпаки.

4. Оскільки ЦОД, де виконуються сервіси, мають гігантські розміри та розподіляють ресурси серед великих груп користувачів, витрати на інфраструктуру (електроенергія, будинки тощо) зменшуються. В результаті витрати на окремого користувача виявляються нижчими.

5. Користувачі отримують необхідні послуги і платять лише за їхнє реальне використання. Якщо, наприклад, відвідуваність Web-сайту компанії в робочі дні велика, а у вихідні майже відсутня, оплачується лише необхідна потужність у робочі дні.

1.4 Види хмар та моделі хмарних сервісів

Перш ніж звернутися до провайдера хмарних рішень із замовленням тієї чи іншої послуги, необхідно зрозуміти, хмара якого типу буде оптимальною для компанії. За моделлю розгортання існує такий поділ: публічна, приватна та гібридна хмара. Вибір потрібної моделі може залежати від масштабу організації, її устрою, IT-інфраструктури та специфіки даних, які планується утримувати у хмарному сховищі.

Публічна хмара. Передбачає розміщення віртуальних інфраструктур одразу кількох замовників – від двох до необмеженої кількості. Дані компанії зберігаються на фізичному сервері поряд з інформацією інших організацій, при цьому вони надійно захищені та ізольовані (у тому числі від втручання хмарного провайдера). По суті, у разі сукупність фізичних ресурсів дата-центру провайдера ділиться на кілька віртуальних ЦОДів, якими користуються замовники хмарних послуг. Не можна точно сказати, на якому конкретно фізичному устаткуванні буде розміщено ваші дані: при зберіганні в кластері віртуальні машини переміщуються між серверами для балансування навантаження та підвищення відмовостійкості.

Серед основних плюсів публічної хмари – відносно невисока вартість і, як наслідок, доступність широкому загалу замовників. Також цей тип розгортання відрізняється гнучкістю, зручністю та простотою взаємодії з хмарним хостингом. При стабільному інтернет-з'єднанні, віртуальні машини у публічній хмарі легко розгорнути та згорнути. Обчислювальну потужність можна збільшити у моменти пікових навантажень та зменшити за потреби.

Рівень захисту даних у випадку публічної хмари не підходить компаніям, які пред'являють спеціальні вимоги до безпеки інформації - це помітний мінус такого способу розгортання. Однак, наприклад, у ситуації, коли обладнання в компанії застаріло і його заміна вимагатиме великих витрат, оренда хмари публічного типу може стати оптимальним рішенням для бізнесу.

Приватна хмара. Віртуальні ресурси розміщені на конкретному фізичному сервері, який надається одному орендарю. Іноді обладнання розташовується в ЦОД поруч із обладнанням інших замовників — проте, йдеться тут також про приватну хмару. Важливим є саме той факт, що хмарна інфраструктура виділена для одного клієнта — навіть якщо віртуальні ресурси поділені між внутрішніми підрозділами організації, користується ними один замовник.

Переваги приватної хмари: повна ізольованість IT-інфраструктури, підвищена надійність зберігання даних, конфігурованість системи. Такий тип розгортання підійде компаніям зі складною, розгалуженою IT-інфраструктурою, особливо якщо у цій компанії пред'являються спеціальні вимоги до приватності даних. До того ж, хмарна система обліку споживання ресурсів у разі приватної хмари дозволяє розуміти, скільки споживає той чи інший підрозділ організації.

До недоліків приватної хмари можна віднести її високу вартість, а також тривалість розгортання. Плюс, обмежений пул ресурсів не дозволить у разі потреби збільшити потужність хмарних обчислень.

Гібридна хмара. Поєднує в собі властивості публічної та приватної хмар. Застосовується у випадках, коли замовнику недостатньо потужності приватної хмари, або коли інфраструктура розміщена в приватній хмарі, але певні завдання зручніше проводити в публічній хмарі (за її рахунок знижуються витрати на комунікації та організацію).

Гібридна хмара виступає єдиним варіантом, навіть коли компанія має достатньо обладнання (якісного та у справному стані), але потребує великих ресурсів. Це призводить до переміщення поточної інфраструктури та окремих сервісів у хмару. Також перенесення може бути зроблено для того, щоб

забезпечити відповідність певним рівням безпеки, зазначеним у федеральних законах про захист персональних даних.

Незважаючи на те, що гібридна хмара поєднує деякі плюси приватних та публічних хмар, є у цієї моделі й недоліки. До ризиків використання гібридної хмари можна віднести, наприклад, збільшення загрози втрати даних у процесі передачі з приватної хмари на публічну, і навіть неможливість відстежити, де фактично перебувають дані поза приватного сервера.

На рисунку 1.4 представлено моделі розгортання хмар.

Рисунок 1.4 Моделі розгортання хмар

Специфіка хмарних технологій полягає в тому, що їхнє застосування створює додаткові ризики та вразливості:

- зберігання конфіденційних даних на стороні провайдера послуг (наприклад, персональних даних банківських карток при взаєморозрахунках);

- залежність від швидкості та якості з'єднання з мережею Інтернет;

- складність прогнозування робочого навантаження в піковий годинник з боку клієнтів і, як наслідок, небезпека виникнення мережевих штормів та сервісної відмови в обслуговуванні.

Компоненти хмарних обчислень. Модель хмарних технологій складається з внутрішньої та зовнішньої частин. Вони з'єднані в мережі, як правило, через Інтернет. Як сама хмара виступає внутрішня частина. Зовнішня частина включає клієнтський комп'ютер або мережі комп'ютерів організації та додатків, які

використовуються для доступу в хмару. Внутрішня частина надає комп'ютери, програми, сховища даних та сервери, що створюють хмарні послуги.

Існують наступні моделі обслуговування хмарних обчислень:

1) IaaS – «інфраструктура як послуга». IaaS-провайдери надають замовнику обчислювальну інфраструктуру (сервери, сховища даних, операційні системи та мережеві ресурси) для розгортання та запуску власних програмних рішень.

Варіант підійде компаніям, потреба яких у ресурсах не однакова в різні моменти часу - бувають сплески потреб, але вони поступово спадають (або організація швидко зростає, і виникає проблема постійного масштабування інфраструктури). Також IaaS буде оптимальним рішенням, коли компанія має недостатньо коштів на створення власної інфраструктури.

2) PaaS – «платформа як послуга». Провайдер хмарних сервісів надає замовнику готове програмне середовище та інструменти для його налаштування. Елементами PaaS є апаратне забезпечення, операційна система, СУБД, проміжне ПЗ, інструменти тестування та розробки.

Таку платформу клієнт може налаштувати під свої потреби, зробивши з неї майданчик для тестування програмного забезпечення або, наприклад, систему для автоматизації системи управління. Такий вид сервісу має особливу популярність у розробників програмного забезпечення.

3) SaaS – «програмне забезпечення як послуга». Розробник програмної платформи надає віддалений доступ до неї клієнту. Наприклад, саме за моделлю SaaS корпорація Microsoft забезпечує клієнтам користування MS Office Suite (Office Web Apps) поряд з SharePoint Server, Exchange Server та іншими сервісами та програмами. Також за моделлю SaaS надаються поштовий сервіс Gmail та хмарна версія 1С.

4) DRaaS – «аварійне відновлення як послуга». Варіант для забезпечення катастрофостійких рішень за допомогою хмари провайдера. На майданчик постачальника хмарних рішень реплікуються дані із основного майданчика клієнта. При виході з ладу послуг клієнта, вони протягом декількох хвилин

перезапускаються, але вже в хмарі. Такі рішення особливо цікаві компаніям з великою кількістю бізнес-критичних програм.

5) BaaS – «резервне копіювання як послуга». Як і слідує з розшифровки абревіатури, йдеться про резервне копіювання даних клієнта в хмару провайдера. Постачальник надає не тільки місце для зберігання інформації, але й інструменти для швидкого та надійного копіювання.

Також провайдери найбільших хмарних сервісів пропонують замовникам такі послуги, як DBaaS («база даних як послуга»), MaaS («моніторинг як послуга»), DaaS («робочий стіл як послуга»), STaaS («сховище як послуга») та NaaS («Мережа як послуга»). Хмарні технології дозволяють забезпечувати корпоративних клієнтів повним спектром послуг, здатних спростити вирішення багатьох бізнес-завдань.

На рисунку 1.5 представлено основні моделі обслуговування хмарних обчислень.

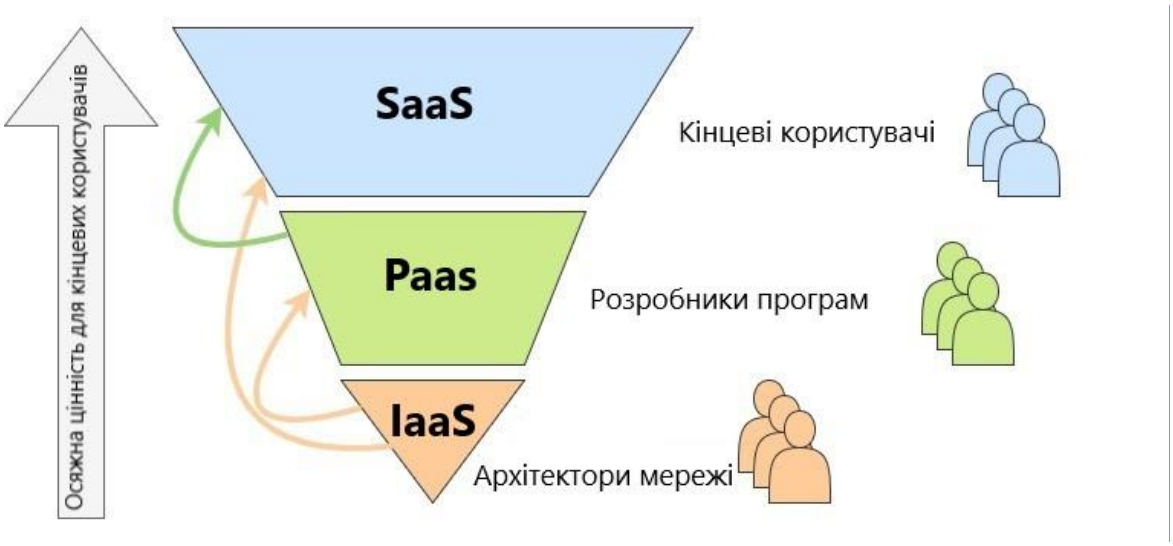


Рисунок 1.5 Основні моделі обслуговування хмарних обчислень

Для розуміння трьох моделей хмарних обчислень у таблиці 1.1 представлені їхні переваги та недоліки. Застосування моделей хмарних обчислень (IT-сервісів) означає перехід від парадигми як актив до парадигми як сервіс. На перший план

серед аргументів переходу в хмари для таких клієнтів виходить забезпечення надійної підтримки та хостингу програми – з відповідним рівнем відповідальності за SLA (Service Layer Agreement).

Таблиця 1.1 Переваги та недоліки моделей сервісів хмарних технологій

Моделі сервісів	Характеристики	Переваги	Недоліки та ризики
IaaS	Зазвичай залежить від платформи; витрати на інфраструктуру поділяються і тому знижуються; оплата за фактом використання; автоматичне масштабування.	Зниження капіталовкладення в апаратне забезпечення та трудові ресурси; зниження ризику втрати інвестицій; низький поріг застосування; плавне масштабування	Бізнес-ефективність та продуктивність залежать від постачальника; потенційно великі довгострокові витрати; централізація потребує інших підходів до заходів безпеки.
PaaS	Споживає інфраструктуру «хмари»; забезпечує методи динамічного управління проектами.	Плавне розгортання версій.	Централізація вимагає інших заходів безпеки, які дозволяють гарантувати, що шкідливі програми не зможуть використовувати вразливість у програмній платформі.
SaaS	Користувальницький інтерфейс; взаємодія у вигляді API; семантична сумісність.	Зниження капіталовкладень в апаратне забезпечення та трудові ресурси; зниження ризику втрати інвестицій;	Централізація даних потребує інших заходів безпеки, пов'язаних із конфіденційністю даних замовника.

		плавне ітеративне оновлення.	

Під час аналізу різноманітних типів хмарних послуг необхідно враховувати межі керованості, які контролюються як замовником, так і компанією-постачальником. На рисунку 1.6 представлено межі керованості.

Рисунок 1.6 – Межі керованості

Межі керованості, зображені на рисунку 1.5 дозволяють зазначити, що найкращим варіантом для контролю всіх компонентів системи є варіант розгортання локальної інфраструктури (виділено сірим кольором). Завдяки цьому, можна контролювати всю інфраструктуру від мережевих ресурсів до виконуваних додатків. В іншому варіанті моделі IaaS можна управляти майже половиною від повного списку послуг, а саме: середовищем виконання коду, базою даних, безпекою тощо.

При використанні моделі PaaS всі компоненти надаються як послуги з обмеженими можливостями управління ними. Це виконано для того, щоб клієнту була доступна платформа, яка оптимально налаштована для його потреб та не вимагає додаткових налаштувань. У моделі SaaS клієнт не може нічого контролювати, йому надається лише кінцевий продукт – програма, програмне забезпечення тощо.

1.5 Основні принципи та системи реалізації хмарних технологій

Сутність концепції хмарних обчислень полягає у наданні кінцевим користувачам віддаленого динамічного доступу до обчислювальних ресурсів, послуг та додатків (включаючи інфраструктуру та операційні системи) через Інтернет. Таким чином, хмарні обчислення – це програмно-апаратне забезпечення, яке доступне користувачеві через Інтернет (або локальну мережу) у вигляді сервісу, що дозволяє використовувати зручний веб-інтерфейс для віддаленого доступу до виділених ресурсів (обчислювальних ресурсів, програм та даних). Комп'ютер користувача у своїй виступає рядовим терміналом, підключеним до Мережі. Комп'ютери, що здійснюють хмарні обчислення, називаються "обчислювальною хмарою". При цьому навантаження між комп'ютерами, що входять до обчислювальної хмари, розподіляється автоматично.

Для користувачів, з погляду обліку ресурсів, хмарні обчислення – це реалізація режиму "плата використання на вимогу", який може зручно реалізовувати доступ до спільно використовуваних ресурсів IT через Інтернет. Якщо ресурси IT включають мережу, сервер, системи зберігання, програми, служби тощо, вони можуть бути розгорнуті швидко і легко, з мінімальними витратами на управління та взаємодії з постачальниками послуг. Хмарні обчислення можуть суттєво покращити доступність ресурсів IT та дають багато переваг у порівнянні з іншими обчислювальними методами. Наприклад, вони можуть мати доступ до послуг без взаємодії з постачальниками послуг. І всі ресурси на хмарі доступні будь-якому користувачеві, тобто користувачі можуть динамічно орендувати фізичні або віртуальні ресурси і не повинні знати їх походження або місце розташування. Крім того, всі ресурси на платформі хмарних обчислень можуть бути розгорнуті швидко і без зупинки обчислень.

Отже, основні принципи функціонування мережі на основі реалізації хмарних технологій наступні:

1. На вимогу самообслуговування - споживач може односторонньо налаштувати обчислювальні можливості, такі як час сервера та мережеве зберігання, як необхідний автоматично, не вимагаючи взаємодії користувача з постачальником кожної служби.

2. Широкий доступ до мережі - можливості доступні через мережу та стандартні механізми, які сприяють використанню неоднорідними платформами клієнта (наприклад, мобільні телефони, ноутбуки та PDA).

3. Об'єднання в пул ресурсу - обчислювальні ресурси провайдера об'єднані в пул, щоб служити багаторазовим споживачам, які використовують модель множинної оренди з різними фізичними ресурсами, коли віртуальні ресурси динамічно призначені та перепризначені відповідно до споживчого попиту. Є сенс незалежності розташування в тому, що клієнт зазвичай не має жодного контролю або знання з точного розташування забезпечених ресурсів, але може бути спроможним визначити розташування на більш високому рівні абстракції (наприклад, країна, стан, або центр обробки даних). Приклади ресурсів включають зберігання, обробку, пам'ять, мережну пропускну здатність та віртуальні машини.

4. Швидкість та гнучкість - можливості можуть бути швидко та гнучко налаштовані, в деяких випадках автоматично, щоб швидко зменшувати масштаби (scale-out) та швидко запущені, щоб швидко збільшувати масштаби (scale-up). Споживачу можливості, доступні для забезпечення часто необмежені і можуть бути куплені в будь-якій кількості в будь-який час.

5. Служба моніторингу - хмарні системи автоматично керують та оптимізують використання ресурсу, посилюючи можливість вимірювання на певному рівні абстракції, що відповідає типу служби (наприклад, зберігання, обробка, пропускна спроможність та активні облікові записи користувачів). Використання ресурсів може контролюватись, керуватися, та гарантувати забезпечення прозорості для провайдера та для споживача використовуваної служби.

1.5.1 Особливості хмарних обчислень

Можна зазначити наступні особливості хмарних обчислень:

1. Масштабованість та послуги на вимогу - хмарні обчислення надають ресурси та послуги для користувачів на вимогу. Ресурси масштабуються в межах декількох центрів обробки даних.

2. Гарантована якість обслуговування (QoS) - хмарні обчислення можуть гарантувати QoS для користувачів з точки зору продуктивності апаратних засобів/ЦП, пропускної спроможності та ємності пам'яті.

3. Автономна система – системи хмарних обчислень – автономні системи, якими керують прозоро користувачі. Однак, програмне забезпечення та дані у хмарах можуть бути автоматично реконфігуровані та консолідовані на просту платформу залежно від потреб користувача.

4. Оцінка – хмарні обчислення не вимагають інвестицій. Жодне капіталовкладення не потрібно. Користувачі платять за служби та ємність, оскільки вони їх потребують.

1.5.2 Виклики хмарних обчислень

Нові парадигми хмарних обчислень надають переваги порівняно з попередніми парадигмами обчислень та багато організацій приймають їх. Однак, є ще проблеми, які зараз розглядаються дослідниками та практиками в області. Вони коротко викладені нижче:

Ефективність. Основна проблема у продуктивності може бути для деяких інтенсивних орієнтованих на транзакції (intensive transaction-oriented) та інших інформаційно ємних додатках (data-intensive applications), в яких хмарні обчислення зазнають проблем у відповідній продуктивності.Крім того, користувачі, які знаходяться на великій відстані від провайдерів хмари, можуть відчути високу латентність та затримки;

Безпека. Компанії, як і раніше, стурбовані безпекою при використанні хмарних обчислень. Клієнти стурбовані вразливістю до атак, коли інформація та критичні ресурси IT знаходяться за межами брандмауера. Рішення для безпеки передбачає, що постачальники хмарних обчислень дотримуються стандартної практики безпеки.

Управління. Деякі відділи інформаційних технологій стурбовані тим, що постачальники хмарних обчислень мають повний контроль над платформами. Постачальники хмарних обчислень зазвичай не розробляють платформи для конкретних компаній та їх бізнес-практики.

Витрати на пропускну здатність (Bandwidth Costs). На хмарних обчисленнях компанії можуть заощадити гроші на апаратне та програмне забезпечення, однак вони можуть нести вищі мережні витрати за пропускною здатністю. Вартість пропускної спроможності може бути низькою для невеликих інтернет-додатків, які не є інформаційно ємними, але можуть істотно зростати для інформаційно ємних додатків.

Надійність. Хмарні обчислення, як і раніше, не завжди пропонують цілодобову надійність. Були випадки, коли послуги хмарних обчислень постраждали через кілька годин відключень.

1.5.3 Проблеми доступу користувачів та аналіз продуктів для запуску додатків у розподіленому обчислювальному середовищі

Основна мета розподіленої системи полягає в тому, щоб полегшити для користувачів (і програм) доступ до віддалених ресурсів, і розділяти їх керованим та ефективним способом. Ресурси можуть бути будь-чим, але типові приклади включають такі речі як принтери, комп'ютери, системи зберігання (жорсткі диски, RAID масиви), дані, файли, веб-сторінки, та мережі. Аналогічно, має економічний сенс розділяти дорогі ресурси, такі як суперкомп'ютери, високоефективні системи зберігання та іншу дорогу периферію.

Однак при збільшенні кількості об'єднаних систем безпека стає все більш важливою. У поточній практиці системи забезпечують невеликий захист проти перехоплення або вторгнення комунікацій. Паролі та іншу значущу інформацію часто посилають як відкритий текст (тобто не зашифрований) через мережу, або зберігають на серверах, так що в цьому сенсі є багато місць для вдосконалення.

Інша проблема безпеки – проблема відстеження комунікацій, щоб створити привілейований профіль певного користувача. Таке відстеження явно порушує секретність, особливо якщо це зроблено, не повідомляючи користувача. У таких випадках те, чого ми, можливо, потребуємо, це захистити себе, використовуючи спеціальні інформаційні фільтри, які вибирають повідомлення, що надходять на основі їх змісту.

Таким чином, при організації доступу системи безпеки (контроль доступу) мають бути надійнішими. В даний час різні компоненти даного напрямку прийнято об'єднувати терміном «хмарні обчислення» (cloud computing), що розвиваються як технологія, що надає обчислювальну послугу як сервіс. Забезпечення інформаційної безпеки (ІБ) таких обчислювальних середовищ є найважливішою проблемою.. Мережевий трафік сприймається як сукупність віртуальних сполук. Завдяки тому, що розподілене віртуалізоване середовище надає гетерогенні обчислювальні ресурси, доцільно використовувати їх для забезпечення його інформаційної безпеки. Так як віртуальні з'єднання функціонують незалежно один від одного, можна організувати паралельну обробку мережного трафіку за допомогою організації «домену безпеки», що функціонує в рамках гіпервізора та використовує кількість ресурсів (ядра, пам'ять), яка потрібна для вирішення поточних проблем ІБ

В даний час розподілені та віртуальні обчислювальні середовища не мають ефективних засобів захисту інформації. Інфраструктура складається з ресурсних центрів, що надають користувачам обчислювальні та дискові ресурси, та інфраструктурних центрів, призначених для координації функціонування інфраструктури. Обчислювальні ресурси, як правило, є кластерами, побудованими зі стандартних ПК об'єднаних локальною мережею. Розмір такого

кластера може досягати кількох сотень вузлів. Дискові ресурси створюються з урахуванням роботизованих стрічкових бібліотек чи великих дискових масивів на жорстких дисках. Об'єм дискових ресурсів може змінюватись від сотень гігабайт до кількох петабайт.

Для використання інфраструктури користувачу необхідно пройти процес реєстрації, після чого всі ресурси стають йому доступними без будь-яких додаткових угод з окремими ресурсними центрами. Процес реєстрації нового користувача включає два основних кроки:

• Отримання персонального сертифіката користувача

• Реєстрація у Віртуальній організації

Персональний сертифікат користувача (Personal Certificate) це свого роду електронний документ, що підтверджує особистість користувача при доступі до Грід-ресурсів. Сертифікати видаються центрами сертифікації (Certification Authority). Віртуальна організація (Virtual Organization) це спільнота користувачів, які спільно використовують обчислювальні ресурси відповідно до узгоджених між ними та власниками ресурсів правил. Ці правила регулюють доступ до всіх типів засобів, включаючи комп'ютери, програмне забезпечення та дані. Кожна віртуальна організація має власний Центр реєстрації.

Безпека Хмарних обчислень - Одна з критичних проблем у реалізації хмарних обчислень з віртуальними машинами, в яких містяться важливі програми та чутливі дані до загальнодоступних та спільно використовуваних хмарних середовищ. Тому потенційні користувачі хмарних обчислень стурбовані такими питаннями безпеки.

• Чи будуть користувачі мати контроль над безпекою над своїми програмами та службами?

• Чи може бути доведено, що організації системи, як і раніше, безпечні та відповідають угоді про рівень обслуговування?

У традиційних центрах обробки даних загальні підходи до безпеки включають периметр брандмауера, демілітаризовані зони, мережну сегментацію,

виявлення проникнення та системи запобігання та контрольні інструменти мережі.

Вимоги безпеки для провайдерів хмарних обчислень починаються з тих самих методів та інструментів, що і для традиційних центрів обробки даних, які включають застосування сильного периметра мережевої безпеки. Однак, фізична сегментація та заснована на апаратних засобах безпека не можуть захистити від атак між віртуальними машинами на тому самому сервері. Сервери хмарних обчислень використовують ті самі операційні системи, підприємства та Веб-додатки як локалізовані віртуальні машини та фізичні сервери. Тому атакуючий може віддалено використовувати вразливість у цих системах та додатках. Крім того, зосередження багаторазових віртуальних машин збільшує поверхню атаки та ризик для компромісу MV-to-VM. Виявлення проникнення і системи запобігання повинні бути в змозі виявити зловмисну дію на рівні VM, незалежно від розташування VM у віртуальному середовищі хмари.

Таким чином, віртуальні середовища, які розгортають механізми безпеки на віртуальних машинах, включаючи брандмауер, виявлення проникнення та запобігання, контроль цілісності, та журнал перевірки, ефективно зроблять VM хмара безпеки, та готова до розгортання

Система входу користувача в систему досить проста, проте вона вирішує кілька важливих завдань за допомогою передових методів комп'ютерної криптографії.

Перша проблема – як зашифрувати ту інформацію, що передається, особливо параметри, пов'язані з входом систему. Для вирішення цієї проблеми використовується технологія асиметричного шифрування (шифрування з відкритим ключем). Кожен користувач або ресурс має пару ключів: відкритий (public), доступний для всіх, і закритий (private), доступ до якого має тільки його власник. При цьому практично неможливо підібрати другий ключ із пари, володіючи лише одним. Всі повідомлення кодуються та розкодуються, використовуючи цю пару ключів. Шифрування виконується відкритим ключем одержувача, а розшифровка закритим. Режим цифрового підпису реалізується у

зворотному порядку: закритий ключ відправника – створення підпису, відкритий ключ – його перевірка, у своїй цифровий підпис може створити лише власник закритого ключа. Однак варто зауважити, що володіння парою ключів автоматично не вирішує завдання автентифікації – тобто перевірки справжності користувачів та ресурсів з метою запобігання проникненню зловмисників. Як бути впевненим у тому, що повідомлення, зашифроване закритим ключем, насправді належить тій особі, за яку вона себе видає?

Цю другу проблему вирішують цифрові сертифікати. Цифровий сертифікат – це відкритий ключ власника сертифіката з інтегрованою персональною інформацією, такою як ім'я користувача, його електронна адреса, місце роботи, термін дії сертифіката тощо. Крім того, сертифікат містить цифровий підпис сертифікаційного центру (Certification Authority - CA) третьої сторони, яка засвідчує належність сертифіката тому, чиї дані записані у сертифікаті. Цифровий підпис – це деяка інформація про сертифікат, зашифрована за допомогою закритого ключа CA. Таким чином, прочитати цю інформацію можна лише за допомогою відкритого ключа CA, який відомий усім. Довіра сертифікату будується на довірі до сертифікаційного центру, який підписав цей сертифікат. Перед видачею (підписанням) сертифіката завдання CA – перевірити належність сертифіката даному індивідууму. Кожен сертифікаційний центр проводить свою політику, яка визначає правила створення та підписання сертифікатів. Зазвичай центри сертифікації існують або в рамках окремих організацій, що беруть участь у проектах, або в рамках цілого проекту або країни.

Реалізація способу отримання сертифіката залежить від центру сертифікації. В даний час, як правило, для отримання сертифіката користувач повинен зайти на сайт центру сертифікації і заповнити форму із запитом на видачу сертифіката. При цьому вказуються персональні дані та назва організації. Обов'язково потрібно вказати правильну адресу електронної пошти, оскільки інформація про отримання сертифіката буде надіслана за цією адресою. Іноді додатково потрібна більш детальна інформація, яка характеризує необхідність отримання сертифіката саме в цьому CA: про галузь наукових інтересів робіт, виконуваних проектах, тощо. Але

як у центрі сертифікації, який може бути в іншому місті чи навіть країні, можуть перевірити докази ідентичності особистості? Для цієї мети в більшості масштабних проектів передбачений інститут реєстраторів (Registration Authority - RA) або довірених осіб, які засвідчують належність об'єкта (користувача, ресурсу або сервісу), що сертифікується, до певної організації та підтверджують персональні дані власника сертифікату. Тому, перш ніж підписати сертифікат, CA зв'язується з відповідним RA і отримує від нього підтвердження справжності запиту отримання сертифіката. В результаті, якщо рішення про видачу сертифіката прийнято, на вказану адресу електронної пошти надходить лист із подальшими інструкціями. Зазвичай, спосіб отримання– чи безпосередній імпорт сертифіката в браузер, тобто. просто перехід за певним посиланням у браузері, або запуск надісланого у листі скрипта на комп'ютері з встановленим пакетом openssl. Залежно від способу створення сертифіката, він може бути збережений у різних форматах (PKCS12 або PFX, DER, PEM). Для того щоб надалі використовувати отриманий сертифікат для роботи через Web, його необхідно експортувати у формат, який розпізнається браузером. Процедура експорту та тип експортованого сертифіката залежить від браузера (*.p12 для Mozilla/Netscape/FireFox та *.pfx для Internet Explorer).

Після отримання цифрового сертифіката користувачеві необхідно зареєструватися у віртуальній організації. Залежно від області роботи користувача це може бути міжнародна, національна або локальна віртуальна організація. Правила реєстрації у віртуальній організації необхідно дізнатись у відповідному Центрі реєстрації. Можлива реєстрація того самого користувача (сертифіката) у кількох віртуальних організаціях. Цей сертифікат є заходом безпеки для запобігання використанню основного сертифіката зловмисниками. Проксі-сертифікат має невеликий термін дії (зазвичай не більше 24 годин) і використовується для підтвердження особи користувача при виконанні будь-яких операцій.

1.6 Мережевий трафік в центрах обробки даних

По суті, «хмара» розташована на фізичному обладнанні, яке є центром обробки даних. Таким чином, постачальники хмарних послуг використовують центри обробки даних для розміщення хмарних сервісів і хмарних ресурсів. Крім того, вони часто мають кілька центрів обробки даних у кількох географічних місцях, тому вони гарантують, що клієнт матиме доступ до своїх даних під час відключення електроенергії та інших відключень у деяких центрах обробки даних.

Мережевий трафік - це кількість інформації, що передається по комп'ютерній мережі за певний період часу.

Існують наступні типи трафіку:

- вхідний: інформація, що надходить із зовнішньої мережі в іншу мережу, комп'ютер, сервер тощо.

- вихідний: інформація надходить у зовнішню мережу;

- внутрішній: переміщення інформації в певній мережі або частині мережі, обмеженою певним чином (наприклад, всередині мережі провайдера);

- зовнішній: інформація, яка надходить за межі захищеної мережі, найчастіше інтернет-трафік.

Мережевий трафік можна виміряти в пакетах, бітах (кілобітах, мегабайтах, гігабітах), байтах (кілобайтах, мегабайтах, гігабайтах).

Залежно від джерела трафіку розрізняють:

- прямий: відвідування сайту в результаті прямого введення адреси в рядок браузера;

- пошуковий: переходи від пошукових систем до певних запитів;

- за посиланням: перехід за посиланнями з інших сайтів;

- платний: відвідувачі, які потрапили за оголошеннями з різної реклами;

- трафік із соціальних мереж: користувачі, які відвідали сайт за посиланням, розміщеним у соціальних мережах.

Центри обробки даних користуються попитом державними установами, великими організаціями, такими як банки, страхові та торгові корпорації, гірничодобувні компанії, телекомунікаційні компанії (біллінгові системи, хостинг, всі види веб-сервісів та соціальних послуг). Усі вони використовують складні бізнес-додатки, а їх діяльність залежить від надійності функціонування IT-інфраструктури. Основна проблема полягає в тому, як отримати максимальну віддачу від їх експлуатації в умовах постійного зростання витрат ресурсів. Це дає рекомендації щодо ключових питань, які необхідно вирішити при впровадженні ЦОД - як досягти підвищення ключових показників (продуктивність, надійність тощо) при мінімізації витрат (споживання енергії, управління), з урахуванням можливого зростання навантаження, забезпечення відновлення після збоїв і високу доступність критичних програм і служб.

На рисунку 1.7 представлено схему мережі ЦОД.

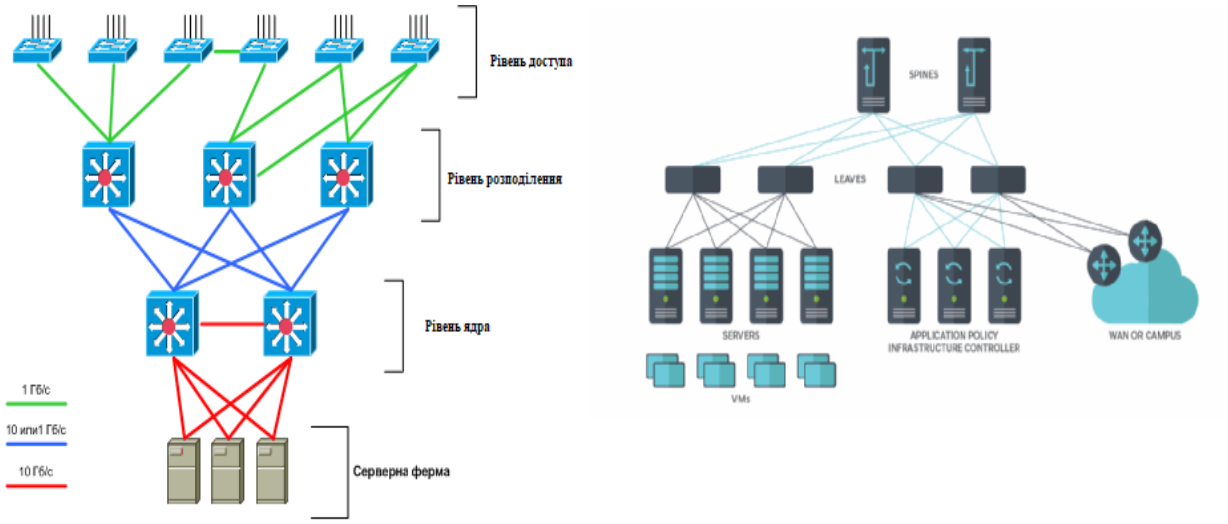


Рисунок 1.7 Схема мережі ЦОД

2 СИСТЕМИ АВТОРИЗАЦІЇ ХМАРНИХ ОБЧИСЛЕНЬ

На сьогоднішній день багато компаній переводять обчислювальні ресурси у віртуальну інфраструктуру, у тому числі провідні університети, використовуючи для цього як відкриті системи (Eucalyptus, OpenNebula), так і комерційні рішення (VmWare, Citrix, IBM). У зв'язку з цим гостро постає проблема забезпечення інформаційної безпеки у віртуальних обчислювальних системах такого роду. Таке віртуальне середовище має певні специфічні особливості, хоча багато його характеристики аналогічні тим, які зустрічаються в мережах розподілених обчислювальних ресурсів і ГРІД додатках. Серед важливих відмінностей можна виділити такі:

• Обробка інформації відбувається у віртуальних машинах, які перебувають під повним контролем гіпервізора, здатного контролювати всі дані, що обробляються віртуальними обчислювальними ресурсами;

• Засоби управління віртуальною інфраструктурою (наприклад, Eucalyptus) здійснюють розподіл навантаження між гіпервізорами та є новою сутністю в інформаційному середовищі, що потребує захисту;

• Традиційні засоби захисту інформації, такі як програмно-апаратні міжмережові екрани не можуть контролювати трафік усередині вузла віртуалізації, таким чином, мережна взаємодія між віртуальними машинами в рамках одного гіпервізора опиняється поза контролем;

• У віртуальному середовищі пристроями зберігання даних виступають файли, які розміщуються в мережевих сховищах, а не апаратні жорсткі диски;

• При міграції віртуальних машин між гіпервізорами виникає передача фрагментів оперативної пам'яті, що може містити конфіденційну інформацію.

Таким чином, завдяки перерахованим вище особливостям, виникають нові загрози інформаційній безпеці, серед яких:

• Атака на засоби управління віртуальними машинами, контролери обчислювального середовища (контролер хмари), кластери та на сховищі даних, на якому розташовуються віртуальні образи машин та дані користувача;

• Неавторизований доступ до вузла віртуалізації;

• Використання віртуальної мережі для передачі даних, не передбачених політикою інформаційної безпеки.

Важливою особливістю віртуальної інфраструктури є те, що атака або неавторизований доступ можуть бути зроблені з віртуальної мережі, де відсутні такі пристрої як комутатори, апаратні міжмережові екрани та фізичні лінії зв'язку, що суттєво ускладнює застосування існуючих методів та засобів захисту інформації у комп'ютерних мережах та ГРІД системах .

Розподілені та віртуальні обчислювальні середовища нині не мають ефективних засобів захисту інформації. Одна з проблем полягає у відсутності міжмережевих екранів, здатних функціонувати серед віртуальних машин так само ефективно, як існуючі на ринку програмно-апаратні засоби захисту інформаційних ресурсів і відображення комп'ютерних атак. Для ряду рішень, наприклад, вільно розповсюджуваного та відкритого хмарного середовища Eucalyptus, побудованого на гіпервізорах XEN або KVM, відсутні ефективні рішення щодо захисту віртуальних машин, незважаючи на популярність даного середовища, що швидко зростає, що забезпечується завдяки сумісності з інтерфейсами продуктів від компанії Amazon (Amazon EC2 , Amazon S3).

В даний час багато провідних корпорацій беруть участь у розвитку хмарних обчислень і було запропоновано багато платформ хмарних обчислень. Створилася сприятлива ситуація для вивчення та застосування хмарних обчислень. Для користувачів-початківців все ще дуже важко зробити розумний вибір серед великої кількості пропозицій. Які відмінності існують для різних платформ хмарних обчислень і які характеристики та переваги мають кожна? Щоб відповісти на ці запитання нижче докладно проаналізовані та обговорені проблеми, характеристики, архітектура та застосування кількох популярних платформ хмарних обчислень. З порівняння цих платформ користувачі можуть краще зрозуміти різні хмарні підходи та розумніше вибрати те, що вони хочуть. У цьому розділі наводяться результати тестування Eucalyptus та Opennebula на

гетерогенній платформі та аналізуються їх особливості та переваги для університетських проектів дата-центрів.

Вважається, що хмарні обчислення стануть початком нової революції у IT-сервісах. Передбачаючи величезний бізнес-потенціал такого підходу, багато країн, урядів та корпорацій вирішили підтримувати та вкладати кошти у розвиток методів хмарних обчислень. EC2 від Amazon, Azure від Microsoft, AppEngine від Google, Синя хмара від IBM і так далі - це все широко використовувані платформи хмарних обчислень. Однак, при всій кількості хмарних платформ, у кожної є власні особливості та переваги, і зробити розумний вибір між ними є великою проблемою. Для допомоги у вирішенні цієї проблеми у цьому розділі проводиться аналіз та порівняння кількох популярних хмарних платформ.

2.1 Хмарна платформа (EUCALYPTUS)

Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems) як проект було розпочато в Каліфорнійському університеті в Санта-Барбарі, і головним чином розвивався для створення платформи приватної хмари з відкритим вихідним кодом. Потім він був підхоплений системним інтегратором Eucalyptus. Eucalyptus - це реалізація хмари з відкритим вихідним кодом Amazon EC2 і сумісна з його інтерфейсами користувача. Він також здійснює віртуалізацію на базі Linux та Xen, як це робиться EC2.

Eucalyptus - еластична обчислювальна структура, яка може використовуватися, щоб з'єднати програми користувачів з корисними системами, це - інфраструктура з відкритим вихідним кодом, що використовує реалізацію для кластерів або робочих станцій еластичних, службових, "хмарних" обчислень та популярного обчислювального стандарту, що базується на протоколі обслуговування. На даний час Eucalyptus є сумісним з EC2 від Amazon, і може підтримувати інші види клієнтів із мінімальною модифікацією та розширенням. На рисунку 2.1. представлено структуру топології ресурсів Eucalyptus.

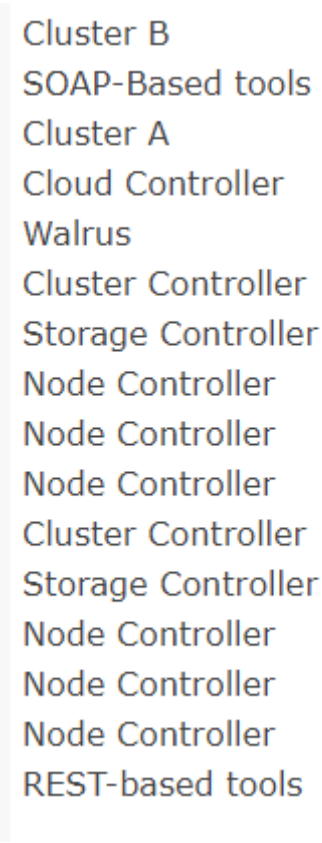


Рисунок 2.1 Структура топології ресурсів Eucalyptus

На рисунку 2.1 Контролер вузла (Node Controller, NC) керує запуском, роботою та зупинкою віртуальних машин на підвідомчому вузлі. Вузол – це машина з працюючим гіпервізором (наприклад, Xen), який здійснює роботу віртуальних машин (instances у термінології Eucalyptus).

Контролер кластера (Cluster Controller, CC) управляє підвідомчими контролерами вузлів (NC): збирає інформацію про завантаженості вузлів та приймає рішення, на яких вузлах буде виконано запуск віртуальних машин.

Контролер сховища (Storage Controller) – місце зберігання образів (image) віртуальних машин. В якості сховища виступає Walrus – сервіс, подібний до Amazon S3.

Контролер хмари (Cloud Controller, CLC) є точкою входу. З боку кінцевого користувача (або вищої програми) надходять запити на запуск віртуальних машин. З боку контролерів кластерів надходять дані про завантаженість вузлів хмари.

Результати тестування приватної хмари на гетерогенній платформі показали, що в нашій приватній хмарі запускали віртуальні машини. Потім користувачам надають доступ до віртуальних машин, на яких вони можуть встановити та виконати довільне програмне забезпечення, включаючи системи баз даних.

$ ~/.euca/eucarc

euca-describe-availability-zones verbose

AVAILABILITYZONE myanmar 192.168.45.41

AVAILABILITYZONE |- vm types free/max cpu ram disk

AVAILABILITYZONE |- m1.small 0000 / 0004 1 128 2

AVAILABILITYZONE |- c1.medium 0000 / 0004 1 256 5

AVAILABILITYZONE |- m1.large 0000 / 0002 2 512 10

AVAILABILITYZONE |- m1.xlarge 0000 / 0002 2 1024 20

AVAILABILITYZONE |- c1.xlarge 0000 / 0001 4 2048 20

$ euca-describe-instances

RESERVATION r-358E067B admin default

INSTANCE i-4DAD080A emi-DE1B1059 192.168.1.102 172.19.1.3 running mykey 0 m1.small 2010-04-10T00:51:19.394Z myanmar eki-F44

RESERVATION r-4C7C0843 admin default

INSTANCE i-47B5091E emi-DEB414FD 192.168.1.103 172.19.1.4 running mykey 0 m1.large 2010-04-10T00:53:58.901Z myanmar eki-238

RESERVATION r-544C09EB admin default

INSTANCE i-3F060729 emi-DFAD1076 192.168.1.101 172.19.1.2 running mykey 0 c1.medium 2010-04-10T00:39:24.834Z myanmar eki-F5EE10

У нашій хмарі розміщують у сховищі образи тих віртуальних машин, які будуть запускатися в хмарі (наприклад, Debian, CentOS, Ubuntu). Запит на запуск віртуальних машин визначає, які машини зі списку можливих і в якій кількості будуть підняті в хмарі. Отримавши запит, контролер кластера вирішує, куди перенаправити цей запит. Користувач має можливість вказати, в якому кластері, або зоні (availability zone в термінології Amazon) повинні бути запущені машини.

Відповідний контролер кластера отримує запит та приймає рішення, на яких вузлах буде виконано завдання. Запит йде або першому з вузлів з наявними вільними ресурсами до його повного заповнення або по черзі кожному з них. Отримавши завдання, контролер вузла завантажує відповідний образ зі сховища (або свого локального кешу, якщо цей образ вже завантажувався) і запускає машини засобами гіпервізора. Користувачам надається доступ до віртуальних машин, на яких вони можуть встановити та виконати довільне програмне забезпечення, включаючи системи баз даних.

Для кінцевого користувача Eucalyptus надає Web-інтерфейс, в якому можна зареєструватися в системі, а потім отримати необхідні дані для роботи з хмарою.

На рисунку 2.2 представлено Web-інтерфейс для кінцевого користувача хмари.

Рисунок 2.2 Web-інтерфейс для кінцевого користувача хмари

Авторизувавшись, користувач потрапляє на сторінку Credentials. Тут є:

· Користувальницька інформація облікового запису.

· Можливість редагувати цю інформацію.

· Можливість змінити пароль.

· Завантажити Credentials ZIP-file.

Можливість подивитися Query ID та Secret Key, не завантажуючи Credentials ZIP-file.

Також доступна сторінка Images, де можна переглянути список всіх образів, які користувач може запустити.

Необхідний для роботи із хмарою. Містить 5 файлів:

• eucarc. Містить опис всіх змінних оточення, необхідних для роботи з euca2ools.

• cloud-cert.pem. "Хмарний" сертифікат.

• euca2-username-*-cert.pem.- користувацький PEM-закодований сертифікат.

• euca2-username-*-pk.pem.- користувацький PEM-закодований приватний ключ.

• jssecacerts.

Для роботи з euca2ools необхідно виконати команду:

source /path/to/eucarc.

Для того щоб використовувати інстанси, запущені з образу, який ви створили (або завантажили), необхідно зв'язати його з «хмарним» сертифікатом, потім завантажити його та зареєструвати на хмарі.

Керувати хмарою можна за допомогою Web-консолі ElasticFox. ElasticFox предствляє собою plug-in, який вбудовується в браузер FireFox на стороні користувача і дозволяє:

• Керувати завантажувальними образами віртуальних машин

• Запускати та зупиняти віртуальні машини

• Керувати віртуальними машинами

• Керувати динамічним розподілом IP адрес

• Керувати доступом до груп

• Керувати ключами авторизації

• Керувати віртуальними блоковими пристроями зберігання даних

На рисунку 2.3 предствлено Web-консолі ElasticFox

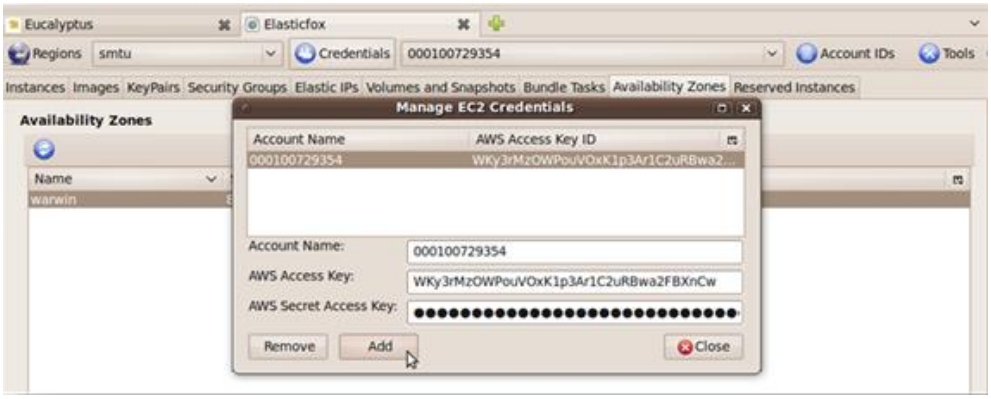


Рисунок 2.3 Web-консолі ElasticFox

В таблиці 2.1 представлено команди управління хмарами Euca2ools.

Таблиця 2.1 Команди управління хмарами Euca2ools

Дія	Команда	Параметри
Прив'язка образу	$euca-bundle-image	-i image.img
Завантаження образу	$euca-upload-bundle	-b image-bucket -m image.img.manifest.xml
Реєстрація образу	$euca-register	image-bucket/image.img.manifest.xml
Завантажити образ	$ euca-delete-bundle	-b image-bucket
Видалити завантажений образ	$euca-delete-bundle	-b image-bucket
Відв'язати образ	$euca-unbundle	-m image.img.manifest.xml
Список запущених інстансів	$euca-describe-instances	
Запустити інстанс	$ euca-run-instances	-k testkey --kernel kernel_id --ramdisk ramdisk_id image_id
Вимкнення інстансу	$ euca-terminate-instance	instance_id
Перезавантаження інстансу	$euca-reboot-instances	instance_id
Створити групу безпеки	$euca-add-group	-d "mygroup description" mygroup
Додавання мережевих правил до груп безпеки	$ euca-authorize	-P tcp -p 22 -s 0.0.0.0/0 mygroup
Скасування мережевих правил для групи безпеки	euca-revoke	-P tcp -p 22 -s 0.0.0.0/0 Mygroup

Запит на виділення IP-адреси	$euca-allocate-address	
Прив'язати IP до інстансу	$euca-associate-address	-i instance_id a.b.c.d
Відв'язати IP від інстансу	$euca-disassociate-address	a.b.c.d
Звільнити виділений IP	$ euca-release-address	a.b.c.d
Видалити групу безпеки	$euca-delete-group	mygroup
Підключитись до інстансу	$ssh	-i keypair root@PUBLIC_IP

Для налаштування знадобляться такі зміни: $EC2_URL, $EC2_ACCESS_KEY, $EC2_SECRET_KEY, $EC2_USER_ID, які можна взяти з файлу eucarc.

• Через розділ "Regions" необхідно вказати ім'я та адресу контролера хмари. У конфігураційному файлі ця адреса вказана у змінній $EC2_URL.

• Через розділ "Credentials" необхідно вказати параметри повноважень користувача, які зберігаються у змінних $EC2_ACCESS_KEY та $EC2_SECRET_KEY.

• Через розділ "Accounts IDs" необхідно вказати унікальний ID користувача, який зберігається у змінній $EC2_USER_ID

• Через розділ Tools необхідно вказати шлях до директорії, в якій встановлений клієнт SSH – Putty або будь-який інший клієнт SSH.

Після встановлення параметрів, ElasticFox з'єднається з контролером хмари та відобразить поточний стан.

На рисунку 2.4 представлено файл eucarc.

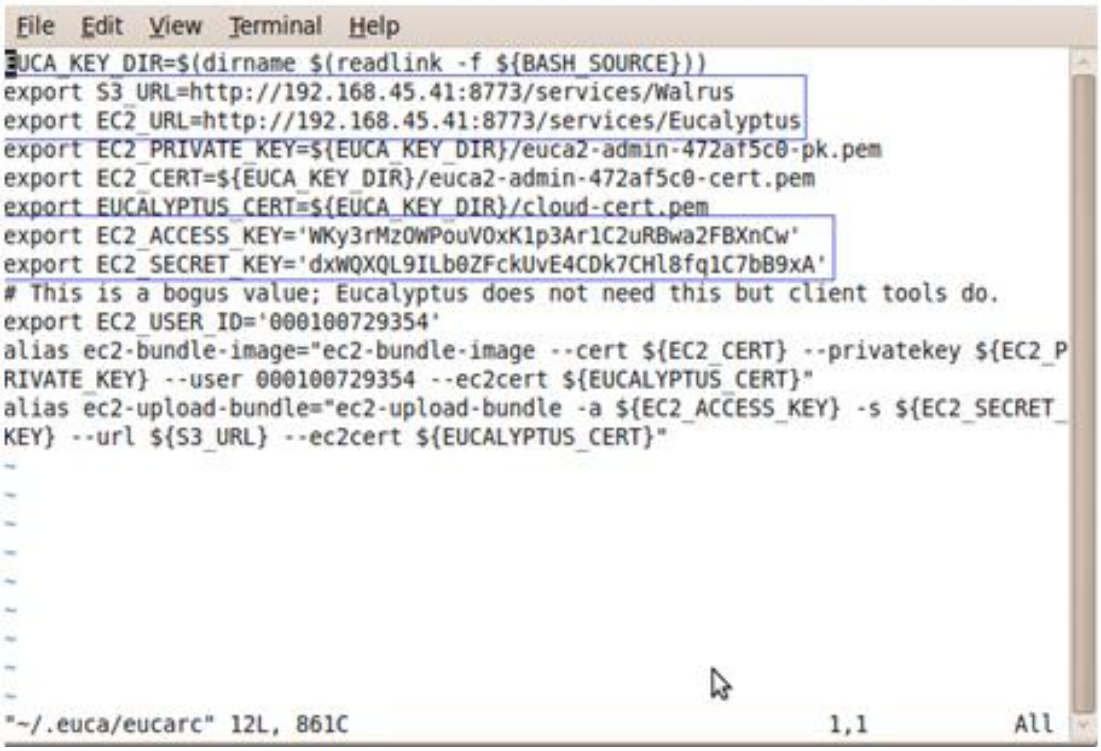


Рисунок 2.4 файл eucarc

2.2 Хмарна платформа (Abicloud)

Abicloud є платформою хмарних обчислень, розроблена Abiquo. Компанія знаходиться в Барселоні, Іспанія, і в основному зосереджена на розвитку хмарних платформ. Продукт може бути використаний для створення, інтеграції та управління громадською, а також приватними хмарами в однорідних середовищах. Основна відмінність між Abicloud та іншими платформами хмарних обчислень є її більш просунуті веб-функції управління та особливості інкапсуляції. Використовуючи Abicloud, користувач може розгортати нові послуги просто переміщенням віртуальної машини мишею. Це набагато легше і ефективніше, ніж для інших платформ хмарних обчислень, в які розгортаються нові сервіси тільки з командного рядка.

За Abicloud, немає ідеальної платформи для створення хмарних структур. Оскільки кожен користувач потребує свої інфраструктури хмари, і у кожного постачальника хмари є свої власні інструменти управління, то взагалі дуже важко розгорнути платформу хмари, що задовольняє всім вимогам користувача.

Найкращий спосіб відповісти на вимогу користувачів полягає в тому, щоб побудувати громадську або приватну хмару як однорідну базову платформу хмарних обчислень та інфраструктури, що розширюються, для конкретних додатків. Крім того, у платформи хмари повинні бути всі види інтерфейсів, які підтримують продукти сторонніх додатків.

Abicloud може бути використаний для розгортання та реалізації приватних хмар, а також гібридних хмар відповідно до запиту та конфігурації хмарних провайдерів. Він може також керувати EC2 відповідно до правил протоколу. Крім того, при розгортанні Abicloud, всі платформи хмари на основі Abicloud можуть бути упаковані та передані будь-якій іншій платформі Abicloud. Це дуже корисно для перетворення операційного середовища і зробить хмарну інфраструктуру легшою та гнучкішою. Архітектуру Abicloud представлено рисунку 2.5.

Рисунок 2.5 Архітектура Abicloud

Фактично Abicloud може підтримувати багато різних платформ віртуальних машин, які включають vBox, VMWare, Xen, KVM і так далі, що робить його дуже гнучким.

У порівнянні з Eucalyptus і OpenNebula цей підхід потребує великих зусиль на підтримку платформи приватної хмари та динамічного керування масштабованістю віртуальних машин на кластерах. Для гібридних хмар він забезпечує доступ за вимогою і такі ж гнучкі механізми, як Amazon EC2.

2.3 Хмарна платформа (Nimbus)

Nimbus – це IaaS-рішення, призначене для наукових обчислень. Використовуючи Nimbus, можна брати в оренду віддалені ресурси (наприклад, Amazon EC2) і керувати ними локально (конфігурувати, розгортати віртуальні машини, стежити за їх роботою і т.д.). Рішення Nimbus утворилося із проекту Workspace Service (який є частиною Globus.org). Nimbus базується на платформі Amazon EC2 і підтримує гіпервізори Xen та KVM/Linux.

На рисунку 2.6 представлено архітектуру платформи Nimbus.

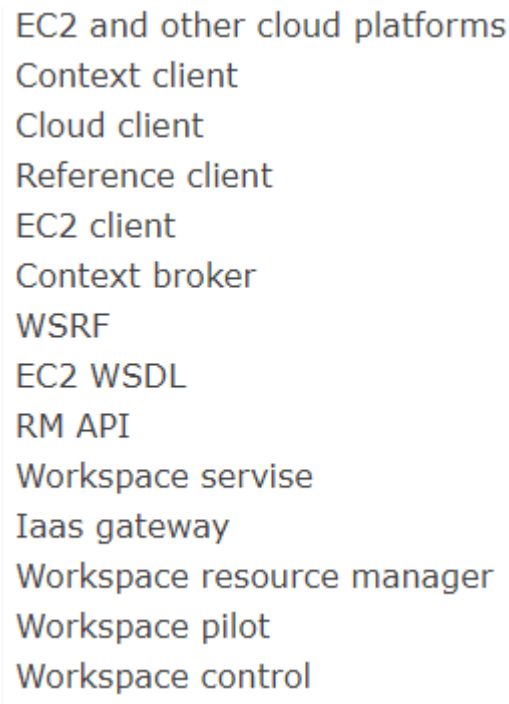


Рисунок 2.6 Архітектура платформи Nimbus

На даний момент існують кілька платформ "хмарних" обчислень, у кожного є відмінні риси та переваги. Щоб краще зрозуміти ці платформи, проведено їх докладне порівняння з різних аспектів реалізації. Характеристики та особливості цих платформ наведено у таблиці 2.2, аналіз якої показує наступне.

Як видно з таблиці 2.2, незважаючи на різні реалізації цих хмарних платформ, є багато спільного між ними. Наприклад, всі вони масштабуються, всі забезпечують IaaS, всі підтримують динамічне розгортання платформи, всі підтримують віртуалізацію за Xen технології, і всі підтримують операційну систему Linux і розробку програм на Java. Однак, є також важливі відмінності у вигляді їх мережевих інтерфейсів, структури та надійності тощо. Зазвичай, кожна хмарна платформа має свої власні переваги перед іншими. Наприклад, з точки зору надійності, OpenNebula є більш зрілою.

Таблиця 2.2 Порівняння кількох платформ хмарних обчислень

	Eucalyptus	OpenNebula	Nimbus	AbiCloud
Характер хмари	публічна	приватна	публічна	публічна/ приватна
Масштабованість	масштабована	динамічна, масштабована	масштабована	масштабована
Форма хмари	IaaS	IaaS	IaaS	IaaS
Сумісність	підтримка EC2, S3	відкрита, багатоплатформна	підтримка EC2	Немає підтримки EC2
Розгортання	динамічне розгортання	динамічне розгортання	динамічне розгортання	Пакетне розгортання
Спосіб розгортання	командний рядок	командний рядок	командний рядок	веб-інтерфейс
Переносність	загальна	загальна	загальна	легка
Підтримка	Xen, KVM та	Xen, KVM,	Xen, KVM	VirtualBox,

віртуальних машин	VMware vSphere, ESX та ESX	VMware		Xen, VMware, VM
Веб-інтерфейс	веб-сервіс	EC2 WSDL, WSRF	libvirt, EC2, OCCI, API	libvirt
Надійність	-	відкати хостів та віртуальних машин	-	-
Підтримка операційних систем	Linux	Linux	Linux	Linux
Розвиток	Java	Java	Java, Python	ruby, C++, python

Хмарні обчислення – нова технологія, широко вивчена останніми роками. З'явилося багато хмарних платформ, як від виробників, так і в академічній сфері. Розібратися з усіма проблемами та зрозуміти, як використовувати ці платформи, є певною проблемою. З проведеного аналізу користувачі можуть краще зрозуміти характеристики та надійніше вибирати платформу хмарних обчислень згідно протоколів, інтерфейсів, сумісності, реалізації, вимоги щодо розгортання та можливості розвитку тощо. Хоча кожна платформа хмарних обчислень має свої переваги, одну річ дуже важливо підкреслити - незалежно від того, яка платформа обрана, залишаються невирішені проблеми. Прикладами проблем є механізми кластерної відмови у хмарному середовищі, гарантії узгодженості та синхронізації на різних кластерах у хмарній платформі, стандартизація, безпека хмарної платформи та даних під час передачі тощо. Всі ці проблеми вимагають свого вирішення кожної конкретної реалізації.

Eucalyptus показав себе, з одного боку, як ефективний засіб об'єднання ресурсів у такому сильно неоднорідному середовищі, а, з іншого боку, інструментальні засоби, що є в ньому, дозволяють легко масштабувати і

переконфігурувати наявні ресурси, що дуже важливо для університетських мереж з їх неоднорідним завантаженням.

Представлено реліз платформи для організації управління хмарною інфраструктурою та віртуальними оточеннями OpenNebula. OpenNebula дозволяє організувати роботу локальної хмарної інфраструктури для надання послуг IaaS (інфраструктура як сервіс), схожої на Amazon EC2, або забезпечити роботу гібридної схеми, комбінуючи ресурси локального дата-центру та зовнішніх хмарних провайдерів. В наявності є засоби для організації розгортання віртуальних оточень, моніторингу, контролю доступу, забезпечення безпеки та управління сховищем. Код системи повністю відкритий під ліцензією Ap ache. Готові інсталяційні пакети доступні для Ubuntu, openSUSE, RHEL/CentOS та Debian.

В OpenNebula 3.6 представлено кілька цікавих нововведень, з яких можна відзначити такі:

· Реалізація механізму гарячого підключення дискових розділів, що дозволяє додавати до віртуального оточення нові розділи або образи віртуальних машин;

· Повністю переписані інструменти для призначення квот та облікового запису, які тепер включені в ядро OpenNebula, що дозволило покращити інтеграцію з такими механізмами, як AuthZ та AuthN, а також іншими компонентами платформи (наприклад, інтерфейсом Sunstone). Додано нові типи квот на сховища та пропускну спроможність мережі;

· Функція перепланування віртуальних машин, що дозволяє примусово мігрувати віртуальні машини на інші хости, наприклад, для звільнення заданого сервера з метою проведення робіт з його обслуговування;

· Підтримка жорсткого перезавантаження віртуальних машин, що зависли;

· Можливість визначення різних системних сховищ для різних кластерів віртуальних машин. Для роботи зі сховищами підготовлено новий LVM-драйвер;

· Функція клонування існуючих дискових образів та створення дублікатів шаблонів віртуальних машин;

· Поліпшення продуктивності підсистеми авторизації;

· Підтримка вказівки кількох LDAP-серверів для автентифікації користувачів;

· Надання розробникам документ-орієнтованого сховища, яке може бути використане для збереження станів програм, що запускаються;

· Редизайн деяких вкладок (dashboard, cluster view) у веб-інтерфейсі Sunstone. Підтримка налаштування через web-інтерфейс гарячого підключення дисків, клонування образів та встановлення квот;

· Повна інтеграція з сервісом OpenNebula Marketplace, що надає користувачам засоби для пошуку та встановлення готових віртуальних оточень, призначених для виконання тих чи інших функцій.

Одночасно можна відзначити реліз системи Eucalyptus 3.1, що конкурує з OpenNebula, дозволяє створити сумісну з Amazon EC2 (API EC2, S3, EBS) локальну cloud-інфраструктуру для прозорого виконання образів віртуальних машин на базі власного набору серверів, що базуються на технологіях Xen або KVM. Eucalyptus 3.1 продовжується розвиток суттєвих нововведень, що з'явилися у гілці 3.x, таких як підтримка забезпечення високої доступності, управління ідентифікацією, модульна структура.

Нова версія показна поверненням до повністю відкритого характеру розробки. Нагадаємо, що Eucalyptus є однією з перших хмарних платформ, популярність якої останнім часом суттєво знизилася – наприклад, серверна редакція Ubuntu перейшла на платформу OpenStack. Основною причиною втрати інтересу розробників до Eucalyptus було переведення проекту на модель Open Core, при якій відкритим залишається лише базова частина, а розширені функції поширюються лише у платній версії. Відтепер ця модель визнана компанією Eucalyptus неефективною, і проект повернувся до повного відкриття всіх напрацювань без поділу на платну та community версії. Повний набір вихідних текстів Eucalyptus 3.1, які постачаються під ліцензією GPLv3, можна завантажити з GitHub. Примітно, що розробка відтепер проводитиметься з використанням GitHub, що спростить участь у проекті сторонніх ентузіастів.

З доданих у Eucalyptus 3.1 покращень можна відзначити:

· Нова архітектура створення плагінів та розбиття платформи на модулі, що дозволяють автоматизувати процес встановлення та спростити оновлення;
· Покращені засоби контролю залежностей, що дозволяють досягти кращої сумісності з різними дистрибутивами Linux;
· Новий автоматизований набір тестів для стеження за забезпеченням високої якості проекту;
· Засоби для спрощення розгортання хмарних систем на базі останньої версії Red Hat Enterprise Linux за допомогою EC2, EBS, S3 та IAM з можливістю використання платформ Red Hat Enterprise Virtualization та VMware;
· FastStart – самодостатній засіб для автоматизації створення IaaS-інфраструктури на базі Eucalyptus. Загальний час, який необхідно для створення робочої системи IaaS оцінюється в 20 хвилин. Режим FastStart може бути використаний у CentOS 5 з Xen або CentOS 6 з KVM. Для встановлення та налаштування складніших конфігурацій представлений набір інструментів SilverEye.

2.4 Аналіз безпеки програмного інтерфейсу API для управління ресурсами та сервісами хмарних обчислень

Провайдери хмарної інфраструктури надають користувачам набір програмних інтерфейсів для керування ресурсами, віртуальними машинами чи сервісами. Безпека усієї системи залежить від безпеки цих інтерфейсів. Починаючи від процедури автентифікації та авторизації та закінчуючи шифруванням, програмні інтерфейси повинні забезпечувати максимальний рівень захисту від різноманітних атак зловмисників.

Для безпеки взаємодії компонентів ec2 API використовується протокол SOAP та WS-security. WS-Security є базою для хмарних технологій у сфері безпеки веб-сервісів. WS-Security забезпечує переміщення завдань ідентифікації та авторизації в область обміну повідомленнями SOAP. WS-Security визначає базові механізми та формати використання security-token у складі SOAP-запитів.

SecurityToken> це -WS-Security автентифікація користувача за допомогою сертифіката X.509.

На рисунку 2.7 представлено компоненти WS-Security.

Рисунок 2.7 Компоненти WS-Security

WS-Policy визначає шаблони та правила опису політики безпеки для Web-сервісів.

WS-Trust визначає правила організації довірених відносин між учасниками Web-взаємодії.

WS-Privacy визначає формати політики конфіденційності під час обміну повідомленнями SOAP.

WS-SecureConversation регламентує правила безпечного обміну повідомленнями в архітектурі.

WS-Federation є специфікацією, що визначає встановлення довірених відносин між різними доменом безпеки.

WS-Authorization визначає формати опису правил розмежування доступу до Web-сервисів.

WS-Authorization визначає формати опису правил розмежування доступу до Web-сервисів.

На рисунку 2.8 представлено WS-Security автентифікація користувача у складі SOAP-запитів.

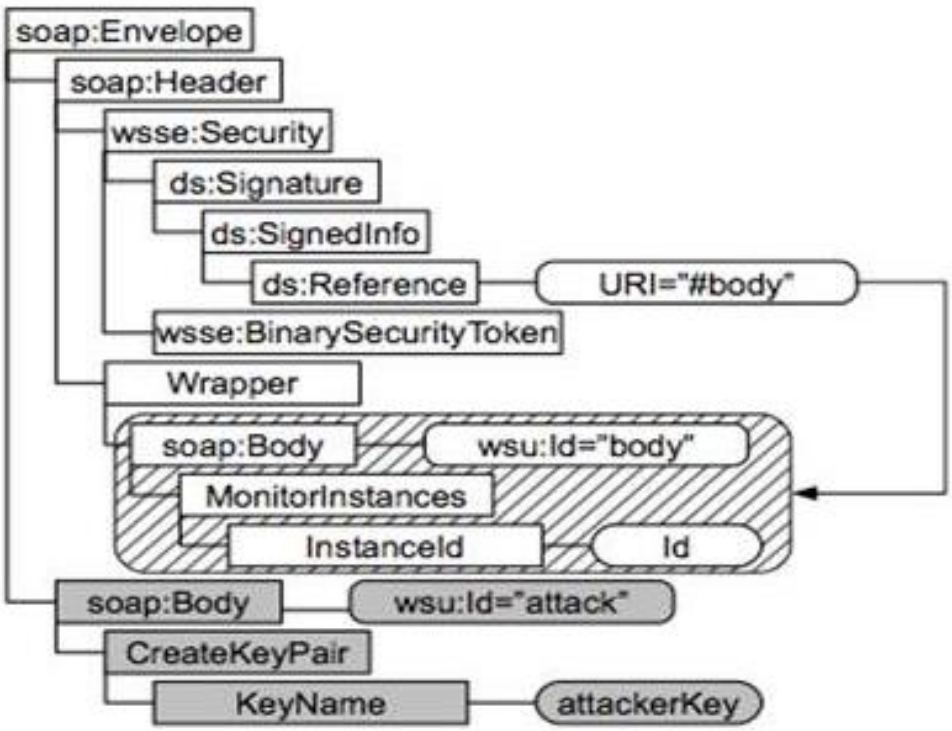


Рисунок 2.8 WS-Security автентифікація користувача у складі SOAP-запитів

2.5 Дослідження продуктивності метакомп'ютера з єдиним образом операційної системи

Сучасні комп'ютерні технології дозволяють створювати відносно дешеві багатомашинні комплекси із загальним математичним забезпеченням та загальними обчислювальними ресурсами – обчислювальні кластери. Такі системи забезпечують відносно низьку вартість обчислень, що добре масштабуються, мають високий рівень надійності, мають апробовані інструментальні засоби для конструювання, налагодження та аналізу паралельних програм. В роботі розглядається MOSIX технологія організації кластерних систем та засоби тестування її продуктивності, її особливості та можливості. MOSIX - системне програмне забезпечення для UNIX-подібних ОС, таких як Linux, що складається з

адаптивних алгоритмів поділу ресурсів. Це дозволяє безлічі однопроцесорних (UP) та SMP вузлів використовуватися для роботи під єдиним керуванням. Алгоритми поділу ресурсів MOSIX розроблені відповідно до використання ресурсів вузлів у режимі реального часу. Це досягається міграцією процесів з одного вузла на інший, переважно та прозоро, для балансування завантаження (load-balancing) та запобігання переповненню пам'яті. Метою такої організації обчислень є збільшення сумарної продуктивності та створення зручного розрахованого на багато користувачів середовища для запуску послідовних і паралельних додатків. Вона підтримує і інтерактивні процеси та пакетні завдання. MOSIX можна розглядати як операційну систему мульти-кластера, яка включає автоматичне виявлення ресурсу та динамічний розподіл робочого навантаження, зазвичай вона завантажується на відокремлених комп'ютерах з великою кількістю процесорів. Система MOSIX забезпечує автоматичне переміщення процесів між ними. Вона розширює ядро Linux механізмами міграції та надає комплект керуючих утиліт, призначених для налаштування системи розподілу процесів, а також для налагодження та контролю. Особливістю MOSIX є відсутність централізованого управління – кожен вузол кластера може працювати як автономна система та самостійно керувати обчислювальними процесами. Це дозволяє динамічно конфігурувати кластер, нарощуючи чи скорочуючи кількість вузлів без зупинки системи. При необхідності MOSIX допускає використання статичного управління, що дозволяє накладати явні обмеження для досягнення максимальної ефективності конкретної кластерної архітектури.

Основною метою цієї роботи стало дослідження та аналіз можливостей оптимізації метакомп'ютера з єдиним образом операційної системи на базі двох вузлового кластера з різною апаратною архітектурою процесорів та встановленим програмним забезпеченням, що забезпечує єдиний образ операційної системи для кластера в цілому.

Ключовим компонентом будь-якої розподіленої системи є файлова система. Як і в централізованих системах, у розподіленій системі функцією файлової системи є зберігання програм та даних, а також надання доступу до них у міру

потреби. Тут аналізується файлова система прямого доступу MOSIX (DFSA), умова, яка може покращити продуктивність кластерних файлових систем, дозволяючи міграційному процесу безпосередньо отримати доступ до файлів у його поточному розташуванні. Така можливість, у поєднанні з відповідною файловою системою, могла б суттєво збільшити продуктивність введення-виводу та зменшити мережеве навантаження, переміщуючи процес, що інтенсивно використовує засоби введення-виводу, на файловий сервер. DFSA підходить для кластерів, які керують пулом загальних дисків. З DFSA можна перемістити паралельні процеси з клієнтського вузла на файлові сервери для паралельного доступу до різних файлів. Щоб протестувати її продуктивність, використовуватимемо файлову систему MOSIX (MFS), яка дозволяє несуперечливі паралельні операції у різних файлах.

Файлова система прямого доступу DFSA була розроблена, щоб зменшити додаткові витрати на введення-виведення, переорієнтовуючи системні виклики міграційного процесу. Це було зроблено, щоб обмежити виконання більшості таких системних викликів локально на поточному вузлі процесу. На додаток до DFSA новий алгоритм, який враховує операції введення-виводу, було додано до системи балансування навантаження MOSIX. В результаті цих змін процес, який виконує від помірного до великого обсягу операцій введення-виводу, мігрує на вузол, який займається, в основному, операціями введення-виведення. Ще одна очевидна перевага полягає в тому, що у процесорів, задіяних як в операціях введення-виведення, так і для обчислень, з'являється велика гнучкість для такої їхньої міграції з відповідних вузлів, яка краще балансує навантаження. Система MOSIX розподілена і масштабована, оскільки вона дозволяє виконувати багато процесів одночасно, і їй доступні різні файли, які були попередньо виділені різними вузлами до виконання паралельних процесів. Поліпшення MOSIX реалізовані в ядрі операційної системи, без зміни інтерфейсу UNIX, і вони повністю прозорі для прикладного рівня.

На рисунку 2.9 представлено схему технологій Mosix.

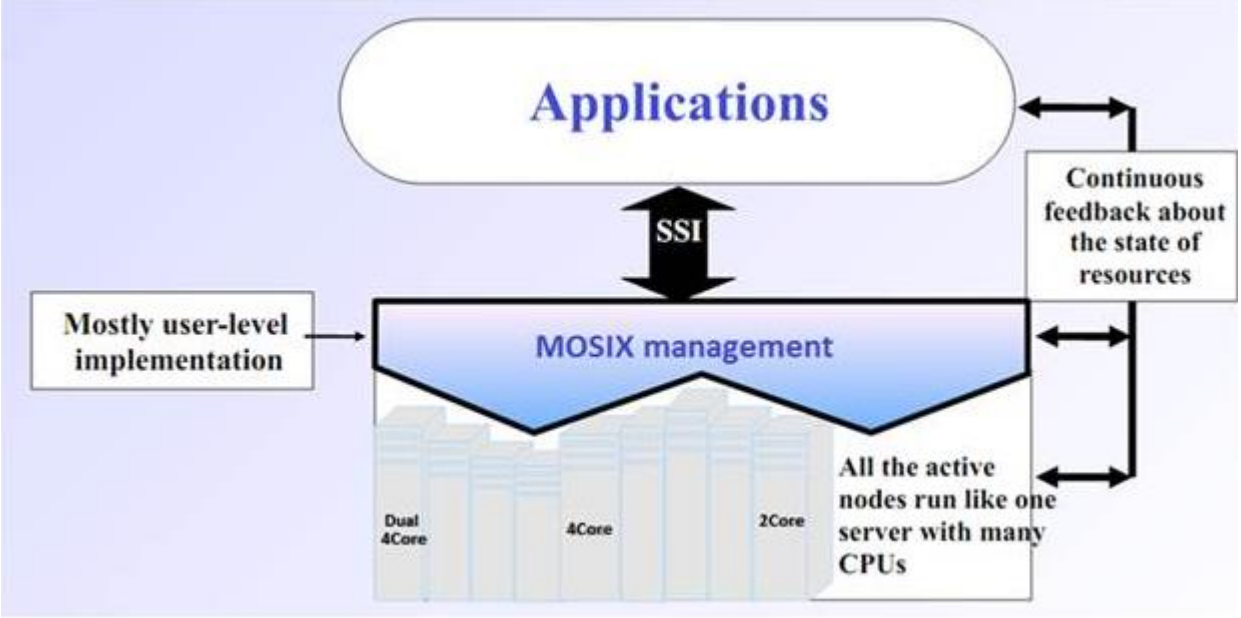


Рисунок 2.9 Схема технологій Mosix

2.6 Дослідження продуктивності MPI з і без міграції процесів MOSIX у віртуальному середовищі

У віртуальному середовищі були встановлені програми MPI і MOSIX і в ній запускалася програма визначення часу затримки при синхронізації процесів MPI. MPI – це стандарт на програмний інструментарій для забезпечення зв'язку між окремими процесами паралельного завдання. MPI надає програмісту єдиний механізм взаємодії процесів усередині паралельно виконуваного завдання незалежно від машинної архітектури (однопроцесорні, багатопроцесорні із загальною або роздільною пам'яттю), взаємного розташування процесів (на одному фізичному процесорі або на різних) та API операційної системи. Програма, що використовує MPI, легко налагоджується і переноситься на інші платформи, часто для цього досить просто перекомпіляції вихідного тексту програми.

У цьому прикладі виконується комунікаційний тест MPI (час затримки повідомлень). MPI-0 надсилає 1-байтове повідомлення до MPI-1, витрачаючи час на очікування відповіді між ними. Після цього виконується синхронізація для

кожного повторення, а при завершенні підраховується середнє очікування. Ці тести були виконані під управлінням операційного середовища MOSIX, з і без переважної схеми міграції процесу. Результати цих тестів довели переваги використання переважної міграції процесу. Для порівняння часу затримки при виконанні тестової комунікаційної програми на MPI та MOSIX, спочатку було запущено програму визначення часу затримки на MPI без MOSIX. І потім та сама операція повторювалася під MOSIX. Нижче в таблиці 2.3 показано результати визначення часу затримки в додатку MPI-latency. Ці результати також доводять переваги використання пріоритетних міграцій процесу.

Таблиця.2.3. Результати (Час затримки - мікросекунди) щодо визначення часу затримки в додатку MPI-latency у віртуальному середовищі

Number of Process (Round Trip Latency Timing Test)	Run MPI without MOSIX	Run with MOSIX		
	Avg round trip time	Avg one way latency	Avg round trip time	Avg one way latency

Відмінними особливостями виконання додатків на MOSIX є адаптивна політика розподілу ресурсів, симетрія та гнучкість конфігурації. Комбінований ефект цих властивостей має на увазі, що користувач не повинен знати поточного стану ресурсів на різних вузлах і навіть їх кількості. Паралельні програми можуть виконуватися, дозволяючи MOSIX призначати та перепризначати процеси на найефективніші з можливих вузлів, майже так само, як і SMP. На відміну від таких пакетів, як MPI або PVM, що фіксують процеси у конкретних вузлах кластера, MOSIX забезпечує їхню прозору динамічну міграцію. При цьому MPI та PVM можуть використовуватись спільно з MOSIX. Залежність продуктивності від швидкості процесора та об'єму пам'яті на стартовому вузлі та для підвищення продуктивності на сильно пов'язаних задачах необхідно підвищувати швидкість мережі, наприклад, використовуючи Gigabit Ethernet або Myrinet.

Завдяки технології MOSIX може створити ефективну обчислювальну систему, яка дозволяє здійснювати динамічне балансування обчислень на вузлах обчислювальної системи на основі технології хмарних обчислень.

2.7 Аналіз продукту Globus Toolkit для організації системи доступу користувачів до розподіленого обчислювального середовища

Щоб забезпечити зручний та безпечний доступ користувачів, у роботі досліджено надійні методики автентифікації та авторизації на базі інтерфейсів та протоколів Гріда як основу системи безпеки у хмарних обчисленнях. Для цього було використано програмний продукт Globus Toolkit. Globus Toolkit – один з основних інструментів, що застосовуються для побудови Грід-систем та їх додатків. Він був розроблений в 1996 для підтримки розвитку сервісно орієнтованих розподілених обчислень. Globus розростався завдяки open-source стратегії. Вона провокує більш широке і швидке поширення і призводить до великих технічних нововведень, оскільки спільнота користувача безперервно підвищує якість продукту.

Мета його створення - надання можливості програм працювати з розподіленими різнорідними обчислювальними ресурсами як з єдиною віртуальною машиною. Основна спрямованість даного проекту – обчислювальні Грід-системи. Під обчислювальної Грід-системою мається на увазі інфраструктура апаратних та програмних ресурсів, що реалізує надійний та повномасштабний доступ до високопродуктивних обчислювальних систем, незалежно від географічного розташування користувачів або ресурсів.

На рисунку 2.10 представлено загальну схему взаємодії компонентів Globus Toolkit.

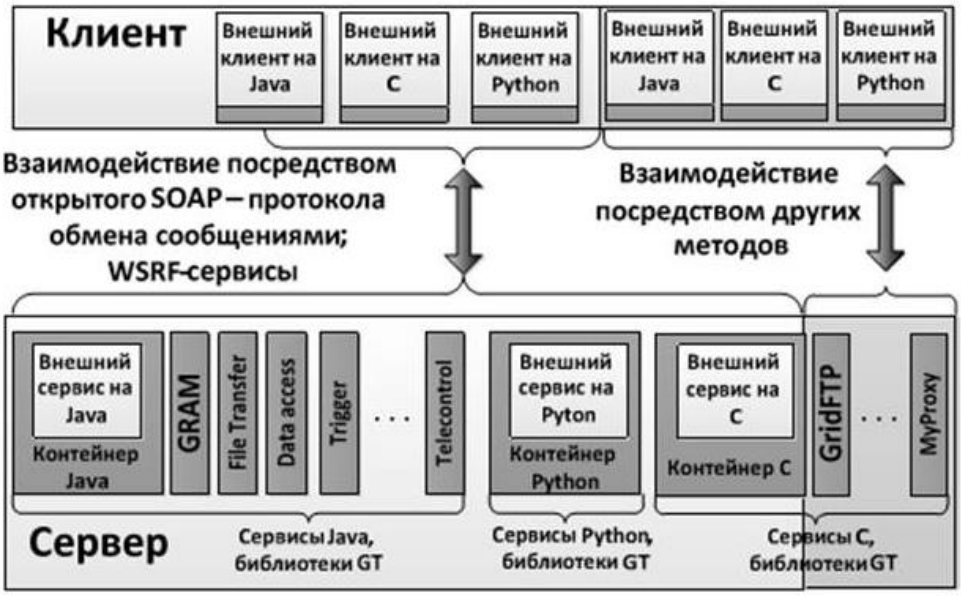


Рисунок 2.10 Загальна схема взаємодії компонентів Globus Toolkit

Базовим елементом системи Globus виступає Globus Toolkit (інструментарій Globus), що описує базові послуги та можливості, необхідні створення обчислювальних Грід-систем. Система Globus надає високорівневим програмам доступ до сервісів, кожен з яких додаток або розробник може використовувати для досягнення власних цілей. Такий спосіб роботи може бути реалізований лише за високого ступеня ізольованості окремих сервісів і чітко визначеному програмному інтерфейсі кожного сервісу.

Розглянемо базові послуги, що надаються системою Globus на сьогоднішній день.

1. Протокол GRAM (“Globus Toolkit Resource Allocation Manager” – Менеджер розподілу ресурсів Globus Toolkit) використовується для розподілу обчислювальних ресурсів та контролю обчислень, з використанням даних ресурсів.

2. Розширена версія протоколу передачі файлів GridFTP використовується для організації доступу до даних, включаючи питання безпеки та паралелізму високошвидкісної передачі даних.

3. Контейнери для сервісів, що підтримують автентифікацію, управління станом, пошук і т.п. що забезпечують підтримку стандартів WSRF, WS-Security, WS-Notification.

4. Сервіси автентифікації та безпеки з'єднань GSI (“Grid Security Infrastructure” – Інфраструктура Безпеки Грід).

5. Розподілений доступ до інформації про структуру та стан системи розподілених обчислень.

6. Віддалений доступ до даних за допомогою послідовних та паралельних інтерфейсів.

7. Створення, кешування та пошук виконуваних ресурсів.

8. Бібліотеки, для забезпечення взаємодії сторонніх додатків з Globus Toolkit та/або сервісами.

2.8 GSI - для забезпечення єдиного входу в Грід-систему

В основі безпеки в Globus Toolkit лежить принцип авторизації та сертифікатів, що реалізуються за допомогою компонентів захисту. Вони дозволяють контролювати дії всіх користувачів у Гріді, захищаючи передані дані. Також ці компоненти підтримують редагування повноважень користувачів, збереження інформації про членство у групах доступу.

Grid Security Infrastructure (GSI) – компонент, що надає API для автентифікації, авторизації та сертифікації. GSI заснована на надійній і широко використовуваній інфраструктурі криптографії з відкритим ключем (Public Key Infrastructure – PKI). Як ідентифікатори користувачів та ресурсів у GSI використовуються цифрові сертифікати X.509. У роботі з сертифікатами X.509 та у процедурі видачі/отримання сертифікатів задіяні три сторони:

1. Центр Сертифікації (Certificate Authority – CA) – спеціальна організація, яка має повноваження видавати (підписувати) цифрові сертифікати. У GT5 це завдання виконується компонентом SimpleCA.

2. Передплатник – це людина або ресурс, який користується сертифікаційними послугами CA. SimpleCA включає в сертифікат дані, що надаються передплатником (ім'я, організація тощо) та ставить на ньому свій цифровий підпис.

3. Користувач – це людина або ресурс, що покладається на інформацію із сертифіката при отриманні її від передплатника. Користувачі можуть приймати або відкидати сертифікати, підписані CA.

У Globus Toolkit використовуються два типи сертифікатів X.509:

1. Сертифікат користувача (User Certificate) – цей сертифікат повинен мати кожен користувач, який працює з Грід-системою. Сертифікат користувача містить інформацію про ім'я користувача, організацію, до якої він належить, та центр сертифікації, який видав цей сертифікат.

2. Сертифікат вузла (Host Certificate) – це сертифікат повинен мати кожен вузол (ресурс) Грід-системи. Сертифікат вузла аналогічний сертифікату користувача, але замість імені користувача вказується доменне ім'я конкретного обчислювального вузла.

Для управління сертифікатами X.509 та повноваженнями у Globus використовується компонент MyProxy. MyProxy поєднує в собі онлайн репозиторій облікових записів та онлайн Центр Сертифікації, що дозволяють користувачам безпечно отримати доступ до своїх облікових записів де і коли потрібно. Користувач за допомогою команди myproxy-logon автентифікується та отримує права доступу, що включають CA-сертифікат і Списки Анульованих Сертифікатів (CRLs).

Останнім компонентом безпеки є GSI-OpenSSH – модифікований OpenSSH, в якому додано роботу з проксі-сертифікатами X.509 для забезпечення доступу до віддалених систем та передачі даних без введення паролів, покладаючись на достовірний proxy credential для аутентифікації (це комбінація проксі-сертифіката та відповідального) йому ключа).

Grid Security Infrastructure (GSI) надає API для автентифікації, авторизації та сертифікації. GSI заснована на надійній і широко використовуваній

інфраструктурі криптографії з відкритим ключем (Public Key Infrastructure – PKI). Як ідентифікатори користувачів та ресурсів у GSI використовуються цифрові сертифікати X.509. Проксі-сертифікат X.509 для забезпечення доступу до віддалених систем і передачі даних без введення паролів, покладаючись на достовірний proxy credential для аутентифікації (це комбінація проксі-сертифікату та ключа, що відповідає йому). Для керування сертифікатами X.509 та повноваженнями у GT використовується компонент MyProxy. Користувач за допомогою команди myproxy-logon автентифікується та отримує права доступу, що включають CA-сертифікат.

2.8.1 Призначення GRIDFTP – керування даними

Керування даними в Globus Toolkit здійснюється за допомогою інструментів GridFTP. Протокол GridFTP надає безпечну, стійку до помилок, швидку та ефективну передачу даних (особливо у великих обсягах).

Для доступу до даних використовується клієнт, що називається globus-url-copy (хоча існують також сторонні розробки з інтерактивним інтерфейсом). Цей клієнт дозволяє отримати доступ до інформації через різні протоколи (hhtp, https, ftp, gsiftp, file). Клієнт є простим командним рядком, який може бути елементом інших сценаріїв, наприклад: globus-url-copy gsiftp://remote.host.edu/path/to/file file:///path/on/local/host.

2.8.2 Призначення GRAM – керування процесами

Globus Toolkit надає доступ до обчислювальних процесів за допомогою компонента GRAM. З його допомогою ньому можна додавати, контролювати, відстежувати та скасовувати завдання у системі.

Щоб на даному обчислювальному вузлі можна було віддалено запускати виконання програми, на ньому повинен виконуватися спеціальний процес званий Gatekeeper. Програми користувача формують запити до GRAM спеціальною

мовою RSL (Resource Specification Language). Gatekeeper працює в привілейованому режимі та виконує наступні функції:

· здійснює взаємну автентифікацію з клієнтом;

· аналізує RSL запит;

· відображає клієнтський запит на облікову запис деякого локального користувача;

· запускає від імені локального користувача спеціальний процес, званий Job Manager, і передає йому список потрібних ресурсів.

3 ДОСЛІДЖЕННЯ СИСТЕМИ УПРАВЛІННЯ У БАГАТОПРОЦЕСОРНИХ СЕРВЕРАХ

Як відомо, продуктивність будь-якого процесора при виконанні заданої програми залежить від трьох параметрів: такту (або частоти) синхронізації, середньої кількості команд, що виконуються за один такт, та загальної кількості виконуваних у програмі команд. Змінити жоден із зазначених параметрів незалежно від інших неможливо, оскільки відповідні базові технології взаємопов'язані: частота синхронізації визначається досягнутим рівнем технології інтегральних схем та функціональною організацією процесора, середня кількість тактів на команду залежить від функціональної організації та архітектури системи команд, а кількість виконуваних у програмі команд визначається архітектурою системи команд та технологією компіляторів.

Зі сказаного ясно, що створення нового високопродуктивного процесора вимагає вирішення складних питань у всіх трьох напрямках розробки. У цьому ефективна з погляду вартості конструкція неспроможна покладатися лише на збільшення тактової частоти. Економічні міркування змушують розробників ухвалювати рішення, основою яких є масова технологія. Системи UltraSPARC-1 забезпечують високу продуктивність за досить помірної тактової частоти (до 200 МГц) шляхом оптимізації середньої кількості команд, що виконуються за один такт. Однак за такого підходу природно постають питання ефективного управління конвеєром команд та ієрархією пам'яті системи. Для збільшення продуктивності необхідно по можливості зменшити середній час доступу до пам'яті і збільшити середню кількість команд, що видаються для виконання в кожному такті, не перевищуючи розумного рівня складності процесора.

В архітектуру процесора UltraSPARC T1 входять вісім ядер, заснованих на специфікації SPARC V9 та 90-нанометрової дев'ятирівневої «мідної» технології CMOS, що забезпечують паралельну обробку 32 потоків даних (по 4 потоки на кожне ядро процесора). Процесор UltraSPARC T1 можна представити як систему, що складається з 32 однопотокових логічних процесорів. Вісім ядер зібрано в

загальну серверну SMP-архітектуру. Кожне її ядро здатне до апаратної багатопоточності (Hardware Multithreading), тобто може розділяти власний процесорний час і ресурси між чотирма потоками команд, що паралельно виконуються. На рисунку 3.1 представлено архітектуру процесора UltraSPARC T1.

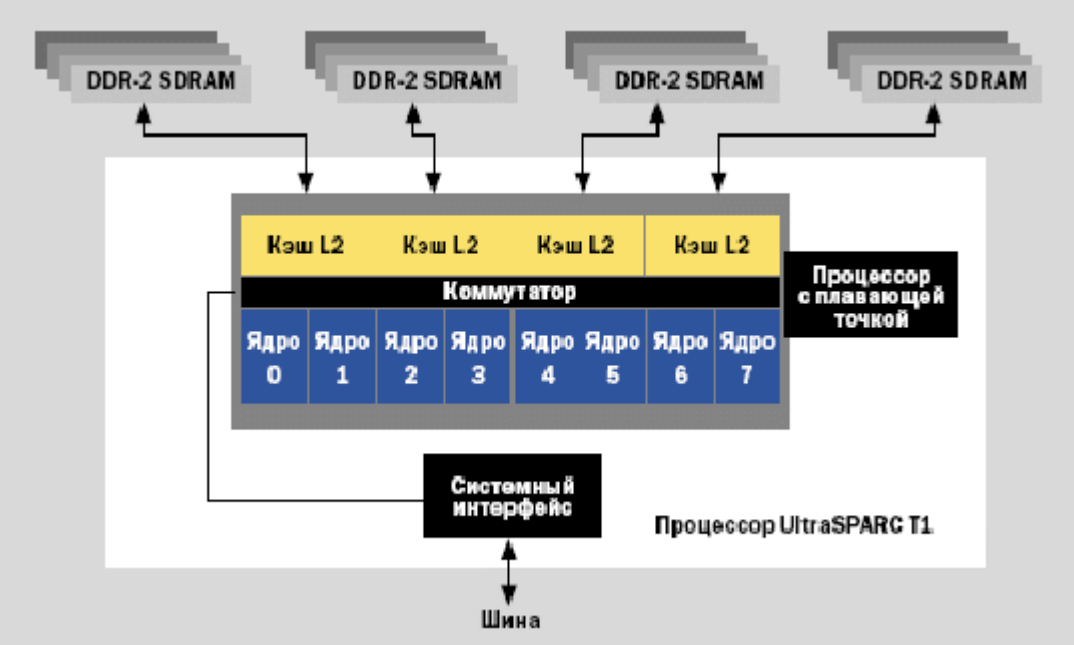


Рисунок 3.1 Архітектура процесора UltraSPARC T1

3.1 Конфігурація Univa Grid Engine

Компанія Univa анонсувала створення форки проекту Sun Grid Engine, призначеного для організації процесу виконання завдань у мережі розподілених обчислень. Компанія Univa вже досить давно займається створенням розширень та надає послуги підтримки для рішень на базі Univa Grid Engine, маючи на це відповідну ліцензію OEM. Після поглинання Univa компанією Oracle розробка відкритого проекту була припинена, тому компанія Univa вирішила взяти ініціативу в свої руки і спільно з незалежною спільнотою розробників продовжити розробку незалежно від Oracle. Продукт Univa Grid Engine призначений насамперед для мереж середнього розміру, що охоплюють відділ або

невелике підприємство. Цей продукт призначений для мереж класу Cluster Grid і доступний безкоштовно.

Пакет дозволяє об'єднати кілька серверів або робочих станцій в єдиний обчислювальний ресурс, який може бути використаний як для пакетних завдань, так і для високопродуктивних пакетних обчислень.

Адміністратор обчислювальної мережі може отримувати дані моніторингу та статистики та на їх основі оптимізувати рівень використання ресурсів. Адміністративний інтерфейс дозволяє задавати різні параметри обчислювальних завдань, такі як пріоритети, необхідні ресурси обладнання, ліцензії на програмне забезпечення, тимчасове вікно виконання, права користувачів на доступ до тих чи інших ресурсів.

На рисунку 3.2 показано діаграму компонентів Univa Grid Engine кластеру.

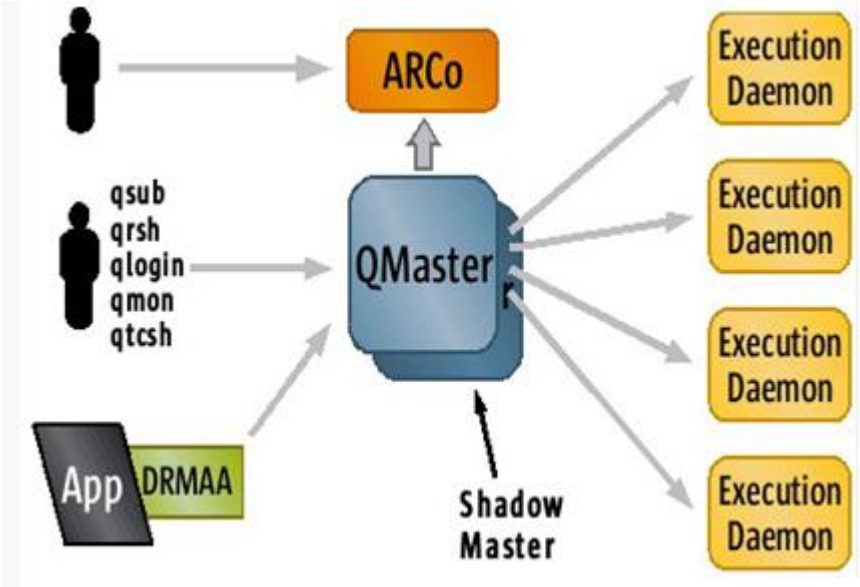


Рисунок 3.2 Компоненти Univa Grid Engine

У центрі діаграми – qmaster. Цей центральний компонент Univa Grid Engine керує кластером, приймаючи завдання, що надходять від користувачів, призначаючи завдання на ресурси, контролюючи поточний статус кластера, і обробляючи команди управління. Qmaster - багатопотоковий демон, який працює на одному хості в обчислювальному кластері. Щоб зменшити незапланований простій кластера, один або більше лабораторних майстрів можуть виконати

завдання на додаткових вузлах в кластері. У разі невдалого завершення завдання qmaster'ом або хостом, завдання передається новому qmaster, запускаючи нового qmaster демона. Кожен хост у кластері, який має виконати завдання, має запустити відповідний демон. Демон отримує завдання від qmaster і виконує їх у певному місці на своєму хості. Програмне забезпечення Univa Grid Engine не встановлює обмежень на кількість завдань, які може розподілити демон, але в більшості випадків кількість завдань визначена кількістю ядер центрального процесора, доступних на хості. Коли завдання завершено, демон повідомляє qmaster, що може планувати нове завдання.

У стандартному режимі кожний демон надсилає повідомлення про свій стан qmaster'у. Якщо qmaster невдало завершує одне із завдань, отримуючи кілька послідовних повідомлень від демона, то qmaster не зареєструє цього хоста та всі його ресурси як доступні та видалить його зі списку планувальника як доступного.

Завдання посилаються в qmaster різноманітними шляхами. DRMAA забезпечує програмний інтерфейс для програми, щоб забезпечити запуск та контроль завдання. Програмне забезпечення Univa Grid Engine працює з C та Java, дозволяючи використовувати DRMAA для широкого діапазону програм. qmon – це графічний користувальницький інтерфейс Univa Grid Engine. Через qmon користувачі та адміністратор можуть запускати, контролювати та керувати завданнями, а також керувати всіма функціями кластера. qsub – це командний рядок утиліта для того, щоб запускати черги, пакети та паралельні завдання. Останній компонент, показаний на діаграмі – це ARCo (Accounting and Reporting Console), інтернет-додаток, що забезпечує доступ до Univa Grid Engine для обліку інформації, що зберігалася в базі даних. Використовуючи ARCo, кінцеві користувачі та адміністратори можуть створювати та виконувати запити на облік роботи кластера.

Послуги та інші об'єкти, які існують та спілкуються у Grid, реалізовані як демони UN*X. Крім того, UGE пропонує величезний набір інструментів у режимі командного рядка для планування завдань, моніторинг та загального управління, з

можливостями резервного копіювання та дружній до користувача інтерфейс. Такий підхід відкриває великі можливості для роботи зі скриптами операційного оточення, наприклад в режимі терміналу.

На додаток до зазначених особливостей важливо розуміти, що реалізація інфраструктури Grid у середовищі UN*X дозволяє використовувати всі вже доступні стандартні та звичні інструменти комунікації, такі як rsh та ssh, просування X-Window, NFS, NIS+, RPC тощо.

Перший крок, необхідний для початку роботи, це зібрати дані з топології доступних ресурсів, вибираючи відповідні машини для поставлених завдань. UGE використовує в основному 4 типи хостів: Master host, Execution host, Administration host and Submit host.

Кожен хост може бути членом в той же час більше, ніж однієї категорії і буде керувати відповідними демонами UGE. Єдине обмеження до цього підрозділу хостів полягає в тому, що може існувати тільки один Master хост на верхньому рівні UGE grid (названий Cell).

На рисунку 3.3 представлено конфігурацію Univa Grid Engine (Графічний інтерфейс користувача – qmon)

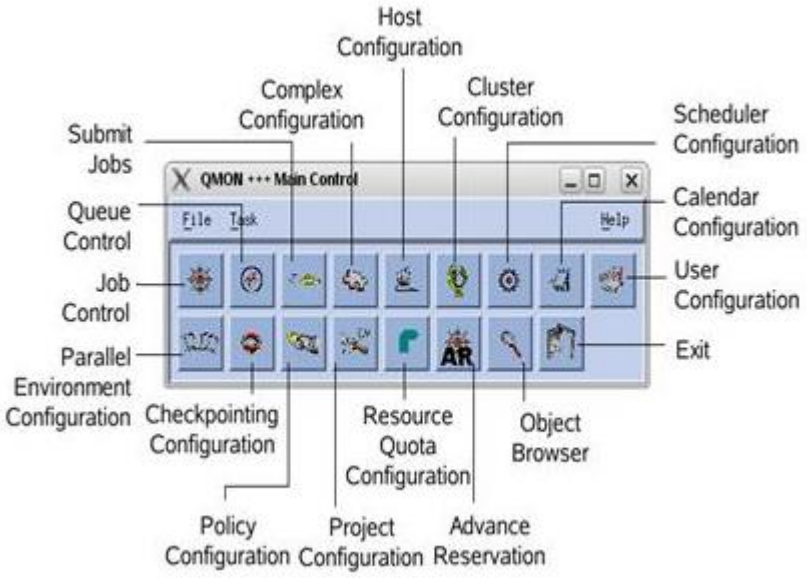


Рисунок 3.3 Конфігурація Univa Grid Engine (Графічний інтерфейс користувача – qmon)

У цій роботі реалізували систему управління розподіленими ресурсами UGE у двох багатопроцесорних серверах Ultra Sparc T1. Розглянуто виконання додатка PI для розрахунків у розподіленому обчислювальному середовищі. Як видно, на рисунку 3.4, було отримано дуже добрі результати: прискорення розрахунків близьке до ідеального.

yemyint@sun2:~$ qhost

HOSTNAME ARCH NCPU NSOC NCOR NTHR LOAD MEMTOT MEMUSE SWAPTO SWAPUS

---

global - - - - - - - - - - -

sun1 sol-sparc64 32 1 8 8 2.59 15.9G 10.1G 2.0G 0.0

sun2 sol-sparc64 32 1 8 8 2.70 15.9G 13.6G 2.0G 0.0

Количество потоков	Время Выполнения (секунды) на Univa Grid Engine	Время Выполнения (секунды) на OPENMP
	37.7104	37.7091
	18.863	18.8585
	9.43742	9.43435
	6.50427	6.50327
	4.5323	4.50626

Рисунок 3.4 Результати часу виконання на UGE та Openmp

3.1.1 Час виконання (секунди)

Продукт Univa Grid Engine призначений для мереж класу Cluster Grid та доступний безкоштовно. Univa Grid Engine об'єднує обчислювальні ресурси для вирішення особливо складних завдань та пропонує ефективний підхід для вирішення таких завдань. У системі UGE існує спеціальна структура даних – паралельне оточення, яке дозволяє враховувати особливості роботи додатків, що використовують технології паралельного програмування. Для вирішення

паралельних завдань робили інтеграцію UGE та Openmp з додатковим пакетом скриптів. У роботі було обрано Univa Grid Engine для системи керування розподіленими ресурсами.

У цьому розділі проаналізовані та обговорені проблеми, характеристики, архітектура та програми кількох популярних програмних продуктів, щоб користувачі розумніше вибрали те, що вони хочуть і могли краще зрозуміти різні підходи до побудови операційного оточення для доступу користувачів та запуску додатків у розподіленому обчислювальному середовищі на основі технології хмарних обчислень. Globus Toolkit - це один з основних інструментів, що застосовуються для побудови Грід-систем та їх додатків. Globus - це open-source ПЗ, має велике користувальницьке співтовариство, хорошу документацію, потужну та стандартизовану систему безпеки. Univa Grid Engine - цей продукт призначений для мереж класу Cluster Grid та доступний безкоштовно. Univa Grid Engine об'єднує обчислювальні ресурси для вирішення особливо складних завдань та пропонує ефективний підхід для вирішення таких завдань. MOSIX – системне програмне забезпечення та однойменна технологія для реалізації масштабованих обчислювальних кластерів. MOSIX можна розглядати як операційну систему мульти-кластера, яка включає автоматичне виявлення ресурсу та динамічний розподіл робочого навантаження, зазвичай вона завантажується на відокремлених комп'ютерах з великою кількістю процесорів.

У цій роботі було створено операційне оточення для доступу користувачів до розподіленого обчислювального середовища на основі технології хмарних обчислень. Для організації системи доступу користувачів до розподіленого обчислювального середовища на основі технології хмарних обчислень можна розділити на 5 частин. Перша частина - Grid Security Infrastructure (GSI) надає API для автентифікації, авторизації та сертифікації. GSI заснована на надійній і широко використовуваній інфраструктурі криптографії з відкритим ключем (Public Key Infrastructure – PKI). Як ідентифікатори користувачів та ресурсів у GSI використовуються цифрові сертифікати X.509. Друга частина – Globus GRAM.

Globus Toolkit надає доступ до обчислювальних процесів за допомогою компонента GRAM. З його допомогою можна додавати, контролювати, відстежувати та скасовувати завдання у системі. Програми користувача формують запити до GRAM спеціальною мовою RSL (Resource Specification Language). Третя частина – GridFTP використовується для організації доступу до даних, включаючи питання безпеки та паралелізму високошвидкісної передачі даних. Протокол GridFTP надає безпечну, стійку до помилок, швидку та ефективну передачу даних (особливо у великих обсягах). Четверта частина – Система управління розподіленими ресурсами. Остання частина – Mosix Обчислювальний кластер, який приймає завдання від користувачів. У роботі були використані ПЗ для розрахунків у розподіленому обчислювальному середовищі на основі Хмари. І було проведено тестування обчислювального середовища на різних завданнях.

3.2 Проектування системи доступу користувачів та розробка технічних принципів запуску ресурсомістких додатків

Для проектування системи доступу для користувачів системи розподілених обчислень у Хмарі було використано продукти Globus Tooolkit та Univa Grid Engine. Ці продукти було проаналізовано на різних платформах. Для забезпечення зручного та безпечного доступу користувачів, у роботі досліджено надійні методики автентифікації та авторизації на базі інтерфейсів та протоколів Гріда як основу системи безпеки у хмарних обчисленнях. Для цього було використано програмний продукт Globus Toolkit. Globus Toolkit - це один з основних інструментів, що застосовуються для побудови Грід-систем та їх додатків. Globus є open-source ПЗ, має велике користувальницьке співтовариство, хорошу документацію, потужну та стандартизовану систему безпеки. Тому було вибрано Globus Toolkit для системи доступу. Univa Grid Engine - цей продукт призначений для мереж класу Cluster Grid та доступний безкоштовно. Univa Grid Engine об'єднує обчислювальні ресурси для вирішення особливо складних завдань та пропонує ефективний підхід для вирішення таких завдань. Тому було вибрано

Univa Grid Engine для системи управління розподіленими ресурсами. MOSIX – системне програмне забезпечення та однойменна технологія для реалізації масштабованих обчислювальних кластерів. MOSIX можна розглядати як операційну систему мульти-кластера, яка включає автоматичне виявлення ресурсу та динамічний розподіл робочого навантаження, зазвичай вона завантажується на відокремлених комп'ютерах з великою кількістю процесорів. Тому перевагу було надано Mosix для створення ефективної обчислювальної системи.

3.2.1 Розробка системи безпеки Гріду в систему хмарних обчислень

Для використання ресурсів користувачеві необхідно пройти процес реєстрації, щоб отримати персональний сертифікат користувача. це свого роду електронний документ, що підтверджує особу користувача при доступі до хмарних ресурсів. На даному етапі розроблено автентифікацію та авторизацію користувачів за допомогою сертифікатів для забезпечення системи безпеки хмарних додатків. Grid Security Infrastructure (GSI) – компонент, що надає API для цих цілей. Щоб зашифрувати ту інформацію, яка передається в хмару, особливо параметри, пов'язані з входом в систему хмари, в GSI використовується технологію асиметричного шифрування з «відкритим ключем». Кожен користувач або ресурс має пару ключів: відкритий (public), доступний для всіх, і закритий (private), доступ до якого має тільки його власник. В якості ідентифікаторів користувачів та ресурсів у GSI використовуються цифрові сертифікати X.509. Цифровий сертифікат – це відкритий ключ власника сертифіката з інтегрованою персональною інформацією, такою як ім'я користувача, його електронна адреса, місце роботи, термін дії сертифіката тощо. Кожен сертифікаційний центр проводить свою політику, яка визначає правила створення та підписання сертифікатів.

Globus використовує SAML (Security Authorization Markup Language) для забезпечення захисту повідомлень, автентифікації, делегування та авторизації таким чином:

· TLS (на транспортному рівні) або WS-Security та WS-SecureConversation (на рівні повідомлень) використовуються як механізм захисту повідомлень у комбінації з SOAP.

· сертифікат Кінцевого Користувача X.509 використовується як реквізити користувача при аутентифікації.

· сертифікат Проксі X.509 використовується для делегування.

· SAML використовується для авторизації.

· файл авторизації Globus (grid-mapfile)

На рисунку 3.5 представлено компоненти для забезпечення захисту повідомлень.

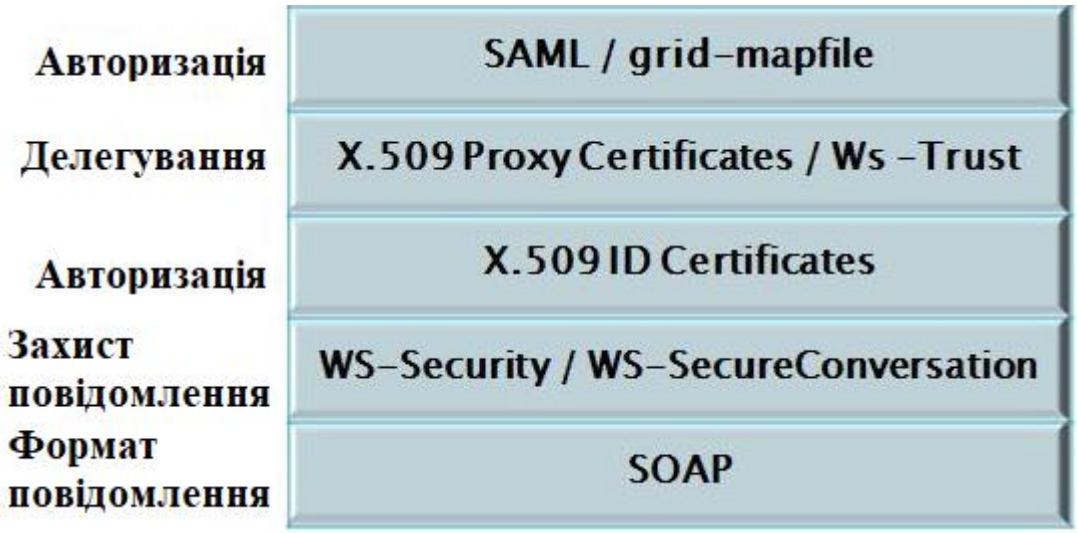


Рисунок 3.5 Компоненти для забезпечення захисту повідомлень

У хмарі створено систему безпеки з використанням надійних методів автентифікації та авторизації для забезпечення системи безпеки хмарних програм. Тепер необхідно протестувати віртуальний хмарний обчислювальний комплекс на різних завданнях і розпочинати роботу з сервісами. Спочатку треба встановити GridFTP – сервіс передачі даних по протоколу gsiftp. Для доступу до даних клієнт використовує команду globus-url-copy. Цей клієнт може отримати доступ до різних протоколів (hhtp, https, ftp, gsiftp, file). Для запуску додатка використовувалися UGE та GRAM. Univa Grid Engine (UGE) використовується

для системи управління ресурсами хмари. Компанія Univa вирішила продовжити розробку проекту Sun Grid Engine, незалежно від Oracle. Univa Grid Engine призначений для мереж класу Cluster Grid. Univa Grid Engine об'єднує обчислювальні ресурси для вирішення особливо складних завдань та пропонує ефективний підхід для вирішення таких завдань.

3.2.2 Програмний інтерфейс DRMAA для інтеграції програмних продуктів

Distributed Resource Management Application API (DRMAA) – програмний інтерфейс для програм управління розподіленими ресурсами. Він надає розробникам додатків програмну модель, яка дозволяє розробляти розподілені програми тісно пов'язані з системою управління розподіленими ресурсами (Distributed Resource Management System, DRMS). Для впровадження таких розподілених додатків, DRMAA зберігає гнучкість і можливість вибору архітектури системи.

Специфікація DRMAA покликана уніфікувати інтерфейси систем DRMS для досягнення переносимості між ними. Вона розробляється спеціальною робочою групою, що входить до складу OGF (Open Grid Forum).

Завдяки тому, що у розробці стандарту DRMAA брали участь представники найрізноманітніших комерційних та дослідних організацій, він швидко був прийнятий спільнотою. В даний час існує кілька реалізацій DRMS, що підтримують цей API, з яких найбільш повною та стабільною є Univa Grid Engine. У специфікації API описується абстрактно мовою опису інтерфейсів IDL (Interface Definition Language), що дозволило лише на рівні мов програмування реалізувати підтримку DRMAA для C/C++, Java, Perl, Python, Ruby.

Специфікація DRMAA забезпечує незалежність Грід програми від використовуваної DRMS. Для цього до неї введено поняття категорії завдання. При надсиланні завдання на Грід програміст задає категорію, яка на конкретній системі відображається на сукупність налаштувань, які можуть включати вказівку

вимог додатка, пріоритету виконання та інших специфічних для DRMS параметрів.

Специфікація DRMAA включає такі процедури:

• Ініціалізація та завершення Грід-програми.

• Завдання шаблону завдання, що включає ім'я команди, початковий стан завдання, параметри середовища виконання, категорію завдання, потоки стандартного вводу/виводу та інші параметри.

• Процедури відправлення на Грід окремих завдань та групових завдань.

• Моніторинг та контроль виконання завдань.

3.2.3 NAS Parallel Bencmarks

Насамперед хотілося б перевірити роботу кластера на «хорошому» завданні, щоб оцінити перспективи його використання для реальних завдань. Тести NAS Parallel Benchmarks підійдуть для цього завдання якнайкраще.

NAS Parallel Benchmarks – безліч тестів продуктивності для перевірки можливостей високопаралельних обчислювальних систем. Вони були розроблені в NASA Advanced Supercomputing (NAS) Division.

У цьому наборі є такі тести як:

• LU - LU-розкладання матриць;

• EP – генерація незалежних нормально розподілених випадкових величин;

• FT – швидке перетворення Фур'є;

• IS – сортування малих цілих чисел;

• DT – тести націлені на оцінку швидкості передачі даних.

Тести розраховані на обчислювальні комплекси різного масштабу. Існує кілька класів завдань для тестів:

• S – невеликі завдання для тестових цілей;

• W – завдання окремих робочих станцій;

• A, B, C – стандартні задачі;

• D, E, F – великі завдання.

У цій роботі використовуватиметься тест LU класу S. Він дозволить оцінити продуктивність комплексу у першому наближенні.

3.2.4 OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) – вільний пакет для чисельного моделювання завдань механіки суцільних середовищ (зокрема обчислювальної гідродинаміки). Його використовують безліч різних організацій у всьому світі, як комерційні, так і некомерційні. Набір бібліотек надає інструменти для вирішення систем диференціальних рівнянь у приватних похідних як у просторі, так і в часі. Написаний C++. OpenFOAM дозволяє вирішувати такі завдання, як розрахунок гідродинаміки ньютонівських та неньютонівських в'язких рідин, як у стисливому, так і стисливому наближенні з урахуванням конвективного теплообміну та дії сил гравітації; розрахунки на міцність; завдання теплопровідності у твердому тілі та інші.

У роботі реалізували матричне множення за різних розмірів матриці в розподіленому середовищі. Як видно з таблиці 3.1, отримано дуже добрі результати: прискорення розрахунків близьке до ідеального.

Таблиця 3.1 Результати матричного множення для матриць різних розмірів

No. Matrix	2 Instances	4 Instances
200x200	0.029763	0.094822
1000x1000	16.619291	6.760375
1500x1500	64.405873	21.981079
2000x2000	172.414044	56.308349
3000x3000	904.551970	242.351147

На рисунку 3.6 представлено результати матричного множення для матриць різних розмірів.

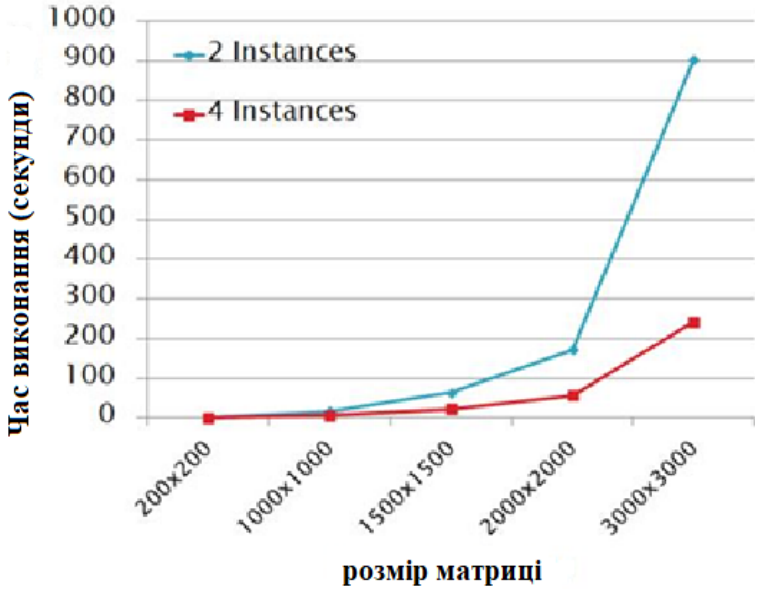


Рисунок 3.6. Результати матричного множення для матриць різних розмірів

У розділі 3 було створено операційне оточення для системи безпечного доступу користувачів та запуску додатків у розподіленому обчислювальному середовищі на основі технології хмарних обчислень. Для використання ресурсів користувачеві необхідно пройти процес реєстрації, щоб отримати персональний сертифікат користувача. Це свого роду електронний документ, що підтверджує особу користувача при доступі до хмарних ресурсів. На даному етапі

налаштовано аутентифікацію та авторизацію користувачів за допомогою сертифікатів. Було використано технологію асиметричного шифрування (шифрування з «відкритим ключем») для того, щоб зашифрувати інформацію, яка передається в хмару, особливо параметри, пов'язані з входом в систему хмари. Кожен користувач або ресурс має пару ключів: відкритий (public), доступний для всіх, і закритий (private), доступ до якого має тільки його власник. В якості ідентифікаторів користувачів та ресурсів у GSI використовуються цифрові сертифікати X.509. Цифровий сертифікат – це відкритий ключ власника сертифіката з інтегрованою персональною інформацією, такою як ім'я користувача, його електронна адреса, місце роботи, термін дії сертифіката тощо. Кожен сертифікаційний центр проводить свою політику, яка визначає правила створення та підписання сертифікатів.

4 НАЛАШТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ OPENFOAM У РОЗПОДІЛЕНОМУ ОБЧИСЛЮВАЛЬНОМУ СЕРЕДОВИЩІ

Для ілюстрації можливостей розробленого підходу наведемо рішення для складного інженерного застосування – розрахунок завдання течії. Оскільки при розв'язанні рівнянь динаміки течій взаємодія між паралельними процесами дуже інтенсивна, реалізація таких завдань є викликом для будь-якої системи розподілених обчислень.

Цей пакет буде встановлений на диск з програмним забезпеченням, доступним для всіх користувачів кластера. Він підмонтований у папку /opt/. Отже, для встановлення OpenFOAM необхідно виконати такі кроки:

1) У вікні терміналу, додати OpenFOAM до списку розташування в сховищі для apt, щоб шукати,

```
VERS=$(lsb_release -cs)
sudo sh -c "echo deb http://www.openfoam.org/download/ubuntu $VERS main > /etc/apt/sources.list.d/openfoam.list"
```

2) Оновити список пакетів APT для врахування нового розташування сховища

sudo apt-get update

3) Встановлення OpenFOAM (211 у назву відноситься до версії 2.1.1):

sudo apt-get install openfoam211

4) Встановити Paraview

sudo apt-get install paraviewopenfoam3120

Щоб використати встановлений пакет OpenFOAM, треба виконати наступні кроки

1) Відкрити Bashrc файл у домашньому каталозі користувача

gedit ~/.bashrc

2) У нижній частині цього файлу додати наступні рядки

source /opt/openfoam211/etc/bashrc

3) Відтестувати програми icoFoam з пакету OpenFOAM,

icoFoam -help

4) " Usage " має з'явитися повідомлення, що установки та конфігурації користувача завершено.

Для запуску послідовних розрахунків необхідно виконати такі кроки:

1. Створити директорію проекту в $HOME/OpenFOAM каталозі з ім'ям <користувач> -2.1.1 і створити каталог з ім'ям, запустивши в ньому,

mkdir -p $FOAM_RUN

2. Копіювати папку приклади в OpenFOAM каталогу. Якщо змінні середовища OpenFOAM встановлені правильно, наступна команда призведе до результату:

cp -r $FOAM_TUTORIALS $FOAM_RUN

Наприклад розглянемо перший випадок ламінарного потоку в порожнині, який не стискається:

cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity

blockMesh

icoFoam

paraFoam

Файл system/decomposeParDict визначає параметри для розпаралелювання завдання:

• параметр numberOfSubdomains повинен відповідати числу паралельних процесів;

• параметр метод задає метод для розбиття області;

• блок simpleCoeffs використовується у разі методу «simple» («n(a b c)» вказує, як саме розбивати область (a * b * c = numberOfSubdomains)).

Директорія 0 містить початкові умови завдання, фізичні властивості змінних вказані в constant/transportproperties.

Цей приклад є одним із стандартних прикладів з розрахунками з Δt=10-5.

Перед запуском розрахунків потрібно підготувати дані. Для цього необхідно перейти в папку із завданням та виконати:

blockMesh

Ця команда створить сітку, для контролю можна виконати:

checkMesh

Тепер можна запускати власне розрахунки. Для розрахунків використовується програма (solver) pisoFoam із пакету OpenFOAM:

pisoFoam

Ці обчислення завершились без помилок. Після цього можна запускати програму візуалізації ParaView за допомогою скрипта paraFoam (результат показано на рисунку 4.1).

paraFoam

Для запуску паралельних розрахунків необхідно виконати такі кроки:

Спочатку необхідно виконати такі ж дії, що й для послідовних розрахунків:

blockMesh

checkMesh

Тепер необхідно провести декомпозицію завдання. Для цього потрібно створити файл decomposeParDict у папці системи даного прикладу, в якому будуть вказані параметри розпаралелювання завдання. Його зразковий вміст:

numberOfSubdomains 2;

метод simple;

simpleCoeffs

{

n (211);

delta 0001;

}

Параметр «numberOfSubdomains» повинен відповідати числу паралельних потоків, які запускаються (він вказує число підзавдань, на яке розбивається завдання). Параметр «method» визначає метод, за допомогою якого буде розбито завдання. Насамкінець необхідно визначити, як саме буде розділене завдання (на 2 області по осі x у даному випадку). Тепер необхідно запустити програму decomposePar, яка і проведе розбиття:

decomposePar

Тепер можна запускати розрахунки

qsub <скрипт>

Вміст скрипта для запуску:

#!/bin/bash

~/ openfoam.sh <директорія_з_даними> pisoFoam

Після виконання розрахунків необхідно запустити програму reconstrucPar:

reconstructPar

Тепер можно переглянути результати за допомогою ParaView (paraFoam – скрипт OpenFOAM, який готує дані для ParaView і запускає цю програму):

paraFoam

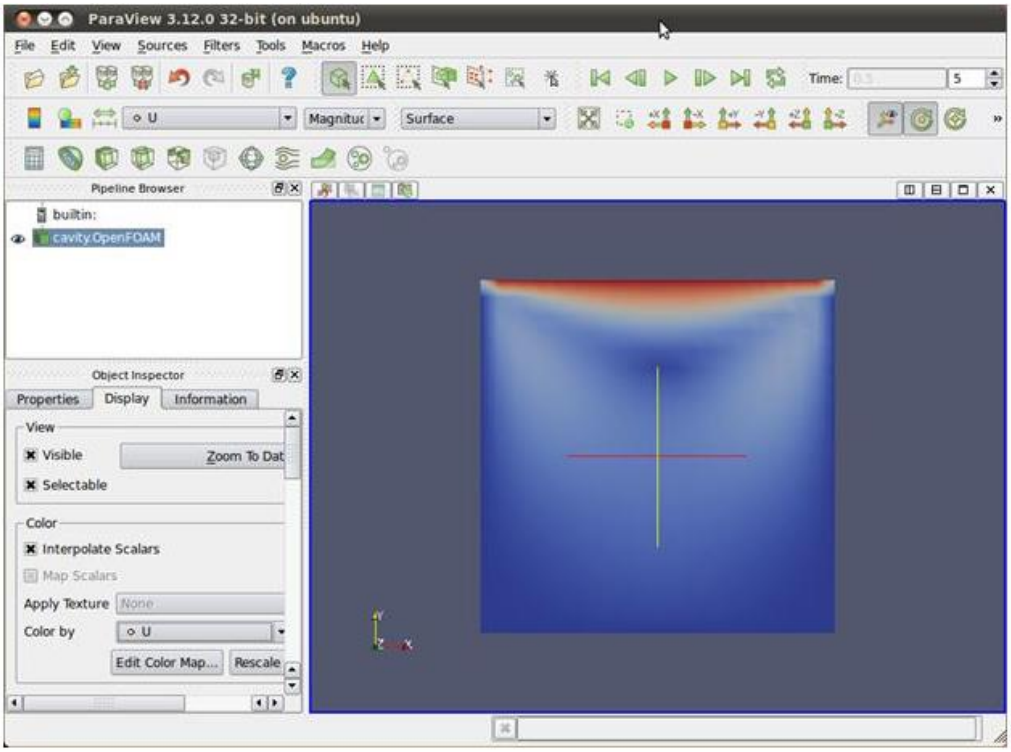


Рисунок 4.1 Перегляд результатів у ParaView

Наведений приклад показує, що навіть дуже складні завдання із сильною взаємодією між паралельними процесами вдається досить ефективно вирішувати за допомогою хмарних технологій.

ВИСНОВКИ

Сучасні хмарні технології (cloud computing) є прогресивним та перспективним напрямком, одним з елементів революційної «третьої IT-платформи». В даний час їх швидке поширення є одним з тих ключових трендів, що в найближчі роки помітно вплинуть на глобальний розвиток.

Розвиток хмарних обчислень, незважаючи на виклики та ризики, є тенденцією сьогодення. Перспектива розвитку хмарних обчислень полягає в тому, що існуючі види технологій будуть замінені єдиною структурою, коли абсолютно все, що може знадобитися для роботи, буде доступно у вигляді сервісу через хмару.

В магістерській роботі було розглянуто різні типи хмарних сервісів і продемонстровано переваги таких систем у порівнянні з традиційними. Також розглянуто переваги та перспективи використання хмарних систем, їх масштабованість та зручність доступу до даних з будь-якого кінця світу практично з будь-якого пристрою при наявності Інтернет-підключення.

В роботі було визначено основні рішення побудови архітектури хмарних систем, виявлено їх переваги та недоліки з погляду інформаційної безпеки, визначено основні моделі обслуговування напряму хмарних обчислень, описана еталонна архітектура хмарних обчислень з погляду захисту даних та моделі безпеки.

В роботі було проаналізовано проблеми, характеристики, архітектура та програми кількох популярних програмних продуктів з метою кращого розуміння різних підходів до побудови операційного оточення для доступу користувачів та запуску додатків у розподіленому обчислювальному середовищі на основі технології хмарних обчислень. Також були використані ПЗ для розрахунків у розподіленому обчислювальному середовищі на основі хмари та проведено тестування обчислювального середовища.

В даній роботі було розглянуто основи реалізації ефективної хмарної обчислювальної системи на базі Eucalytpus та OpenNebula з відкритим кодом, що дозволили реалізувати прототип віртуального хмарного середовища, що має необхідний рівень захищеності даних.

Експериментальне дослідження реалізованих на реальному інформаційному комплексі розподілених обчислювальних середовищ з єдиним образом операційної системи MOSIX у гетерогенному віртуальному середовищі та дослідження продуктивності PVM та MPI з міграцією процесів MOSIX і без неї продемонстрували суттєве поліпшення технічних та експлуатаційних характеристик комплексу при вирішенні ресурсів.

В роботі налаштована та реалізована експериментально інтеграція пакетів Globus Toolkit та Sun Grid Engine на основі стандарту DRMAA дозволила реалізувати можливість побудови операційного оточення для запуску власних програм у віртуальному хмарному середовищі.

Результати, отримані в магістерській роботі внаслідок дослідження, дозволяють зазначити, що організоване хмарне середовище дозволяє ефективно здійснювати динамічне балансування обчислень на вузлах хмарної обчислювальної системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. А.В. Богданов, В.В. Корхов, В.В. Мареев, Е.Н. Станкова. Архитектуры и топологии многопроцессорных вычислительных систем. 2004.

2. Андрей Ященко. Облачные вычисления: прошлое, настоящее, будущее. 04.04.2009. [http://www.ferra.ru/ru/techlife/85658/print/]

3. Владимир Романченко. Распределенные облачные (Cloud) вычисления. [https://sites.google.com/site/moiknigiilekcii/lekcii/informatika/lekcia-no25/cloud]

4. Дмитрий Шепелявый. Обеспечение безопасности Web-сервисов. Журнал "Information Security/ Информационная безопасность" #1, 2008.

5. Ла Мин Хтут. Организация системы доступа для пользователей в распределенной вычислительной среде. Диссертация, 2011.

6. М. Тим Джонс, инженер-консультант, Emulex Corp. Анатомия облака с открытым кодом, 15.06.2011. [http://www.ibm.com/developerworks/ru/library/os-cloud-anatomy/index.html?ca=drs-]

7. About Eucalytpus Cloud [http://www.eucalyptus.com/eucalyptus-cloud]

8. About Globus Toolkit [http://www.globus.org/]

9. About Mosix [http://www.mosix.org/txt_about.html]

10. About OpenFOAM [http://www.openfoam.com/]

11. About Opennebula Cloud [http://opennebula.org/documentation:rel3.6]

12. A. Barak and A. Shiloh, A White Paper, The MOSIX Management System for Linux Clusters, Multi-Clusters, GPU Clusters and Clouds. A White Paper.

13. Alexander Bogdanov, Thurein Kyaw Lwin, Ye Myint Naing, Database use for Consolidation Of CloudComputing, Computer Science and Information Technologies 26 - 30 September, 2011, Yerevan, Armenia . page 237-239.

14. Amnon Barak, Avner Braverman, Ilia Gilderman, Oren Laden. Performance of PVM with the MOSIX Preemptive Process Migration Scheme. The Hebrew University of Jerusalem, Israel.

15. A.V. Bogdanov, M. Dmitriev, Ye Myint Naing, Eucalyptus Open-source Private Cloud Infrastructure, GRID 2010, Proceedings of the 4th International Conference Dubna, June 28- July 3, 2010. page: 57-63.

16. Jinesh Varia. Архитектуры Грид-облаков. Gridclub.ru.

17. J. J. Peng, X. J. Zhang, Z. Lei, B. F. Zhang, W. Zhang, Comparison of Several Cloud Computing Platform, Second International Symposium on Information Science and Engineering, 2009,12, IEEE, page: 23-27.

18. Johnson D, Kiran Murari, Murthy Raju, Suseendran RB, Yogesh Girikumar. Eucalyptus Beginner's Guide – UEC Edition. v1.0, 25 May 2010.

19. NAS Parallel Benchmarks [http://www.nas.nasa.gov/publications/npb.html] Nimrod Vax. Securing Virtualized Environments and Accelerating Cloud Computing. white paper. march 2010.

20. Open source & cloud computing:on-demand, innovative it on a massive scale. White Paper June 2009. Peter Sempolinski, Douglas Thain. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. University of Notre Dame.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАГІСТЕРСЬКА РОБОТА

МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ З ВИКОРИСТАННЯМ ХМАРНИХ ТЕХНОЛОГІЙ

виконав студент: **Харченко В.В.**
керівник: **Лемешко А.В.,** доктор філософії, доцент

1

Мета бакалаврської роботи:

Створення операційного оточення для доступу користувачів до розподіленого обчислювального середовища та розробка принципів запуску додатків у розподіленому обчислювальному середовищі на основі технології хмарних обчислень.

Наукова новизна:

1. Запропоновано новий підхід до побудови операційного оточення для ефективного доступу довільного числа користувачів, запуску ресурсомістких додатків у розподіленому обчислювальному середовищі на основі технології хмарних обчислень.

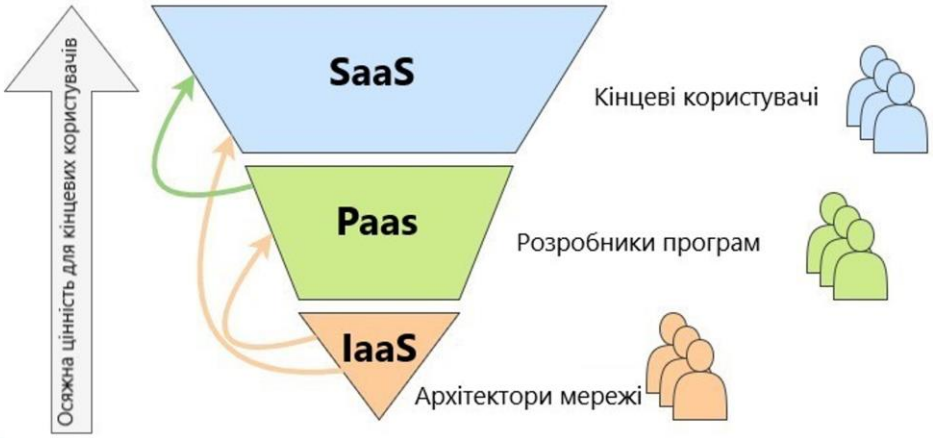
2. Запропоновано методику організації ефективної обчислювальної системи, що дозволяє здійснювати динамічне балансування обчислень на вузлах хмарного обчислювального середовища.

2

Моделі розгортання хмар

3

Основні моделі обслуговування хмарних обчислень



4

Переваги та недоліки моделей сервісів хмарних технологій

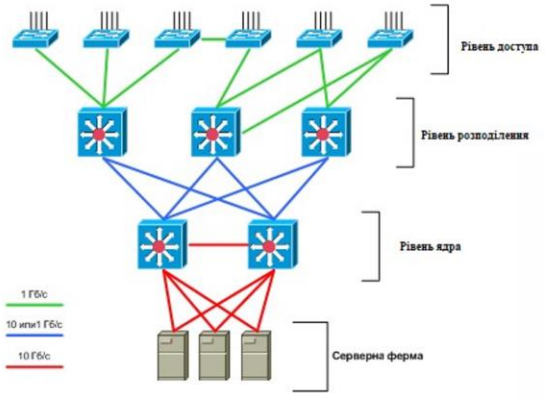
Моделі сервісів	Характеристики	Переваги	Недоліки та ризики
IaaS	Зазвичай залежить від платформи; витрати на інфраструктуру поділяються і тому знижуються; оплата за фактом використання; автоматичне масштабування.	Зниження капіталовкладення в апаратне забезпечення та трудові ресурси; зниження ризику втрати інвестицій; низький поріг застосування; плавне масштабування	Бізнес -ефективність та продуктивність залежать від постачальника; потенційно великі довгострокові витрати; централізація потребує інших підходів до заходів безпеки.
PaaS	Споживає інфраструктуру «хмари»; забезпечує методи динамічного управління проектами.	Плавне розгортання версій.	Централізація вимагає інших заходів безпеки, які дозволяють гарантувати, що шкідливі програми не зможуть використовувати вразливість у програмній платформі.
SaaS	Користувальницький інтерфейс; взаємодія у вигляді API; семантична сумісність.	Зниження капіталовкладень в апаратне забезпечення та трудові ресурси; зниження ризику втрати інвестицій; плавне ітеративне оновлення.	Централізація даних потребує інших заходів безпеки, пов'язаних із конфіденційністю даних замовника.

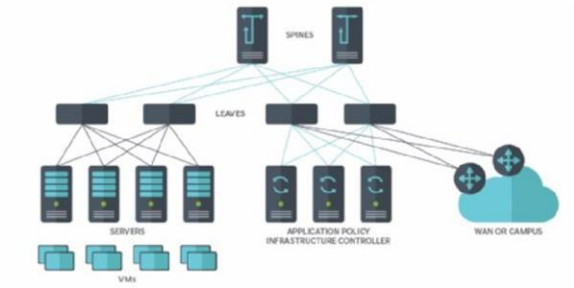
5

Межі керованості

6

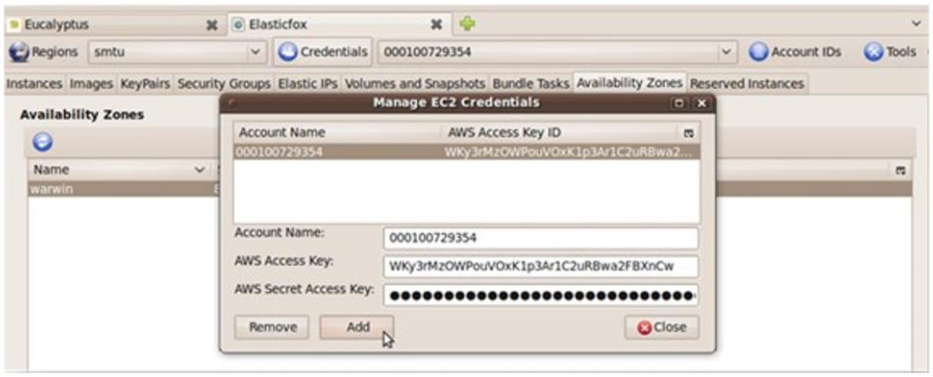
Схема мережі ЦОД





7

Web-консолі ElasticFox



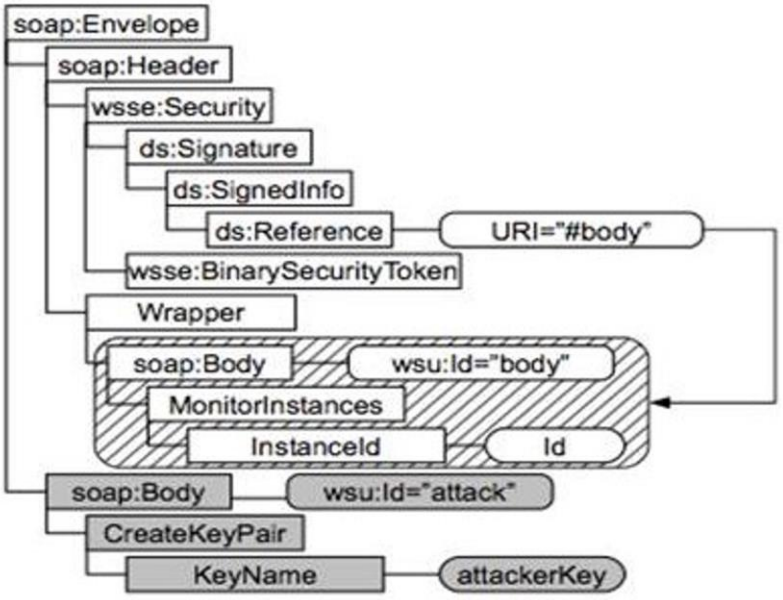
8

Порівняння кількох платформ хмарних обчислень

	Eucalyptus	OpenNebula	Nimbus	AbiCloud
Характерхмари	публічна	приватна	публічна	публічна/ приватна
Масштабованість	масштабована	динамічна, масштабована	масштабована	масштабована
Форма хмари	IaaS	IaaS	IaaS	IaaS
Сумісність	підтримка EC2, S3	відкрита, багатоплатформна	підтримка EC2	Немає підтримки EC2
Розгортання	динамічне розгортання	динамічне розгортання	динамічне розгортання	Пакетне розгортання
Спосіб розгортання	командний рядок	командний рядок	командний рядок	веб-інтерфейс
Переносність	загальна	загальна	загальна	легка
Підтримка віртуальнихмашин	Xen, KVM та VMware vSphere, ESX та ESX	Xen, KVM, VMware	Xen, KVM	VirtualBox, Xen, VMware, VM
Веб-інтерфейс	веб-сервіс	EC2 WSDL, WSRF	libvirt, EC2, OCCI, API	libvirt
Надійність	-	відкати хостів та віртуальних машин	-	-
Підтримка операційнихсистем	Linux	Linux	Linux	Linux
Розвиток	Java	Java	Java, Python	ruby, C++, python

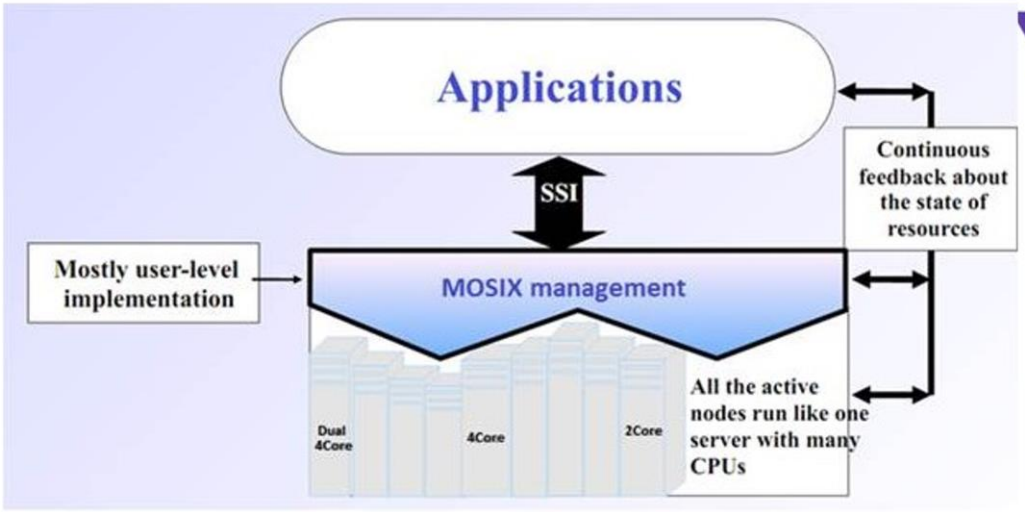
9

Аналіз експериментальних даних



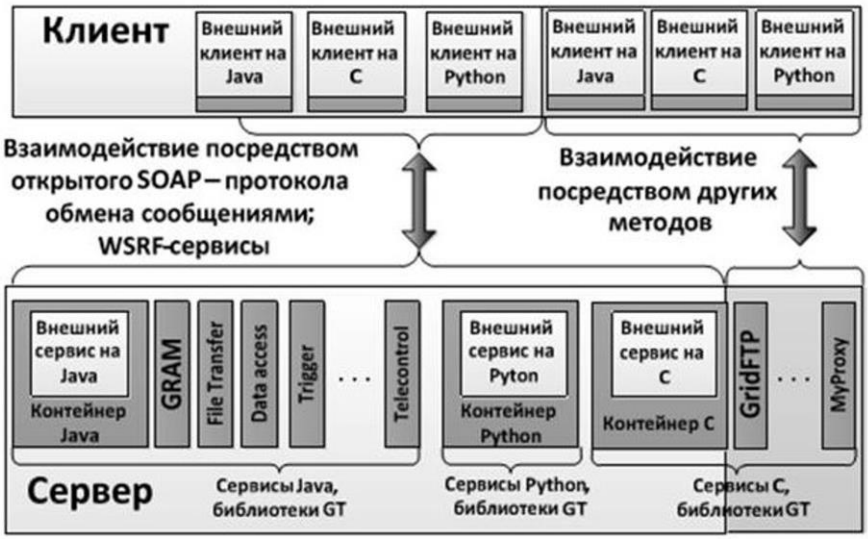
10

Схема технологій Mosix



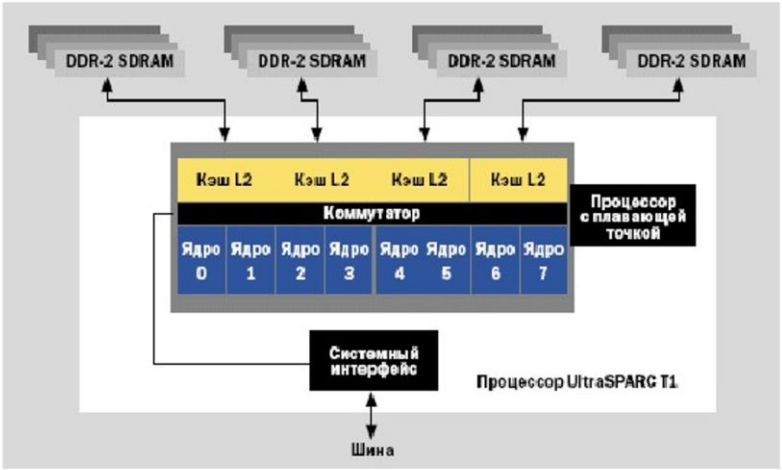
11

Загальна схема взаємодії компонентів Globus Toolkit



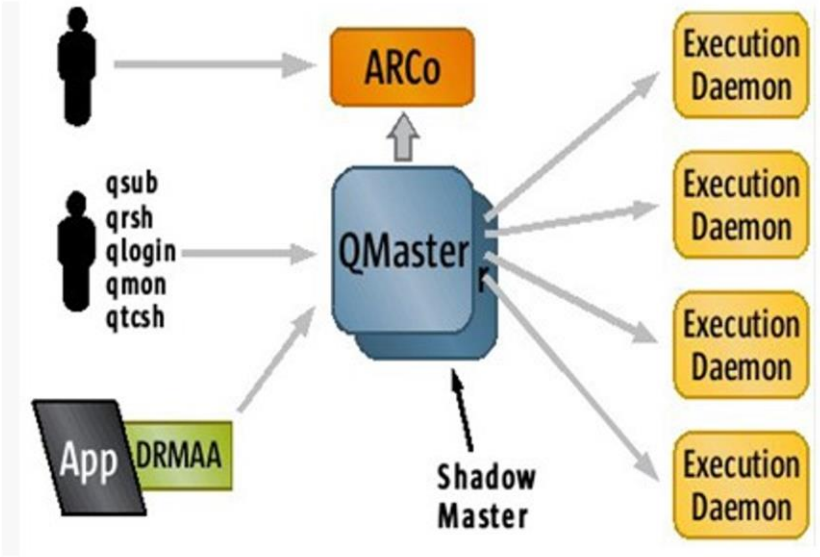
12

Архітектура процесора UltraSPARC T1



13

Компоненти Univa Grid Engine



14

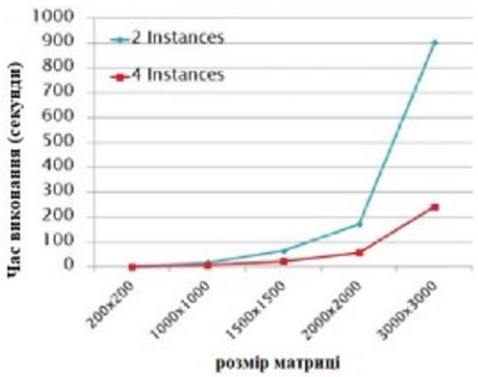
Компоненти для забезпечення захисту повідомлень

Авторизація	SAML / grid-mapfile
Делегування	X.509 Proxy Certificates / Ws -Trust
Авторизація	X.509 ID Certificates
Захист повідомлення	WS-Security / WS-SecureConversation
Формат повідомлення	SOAP

15

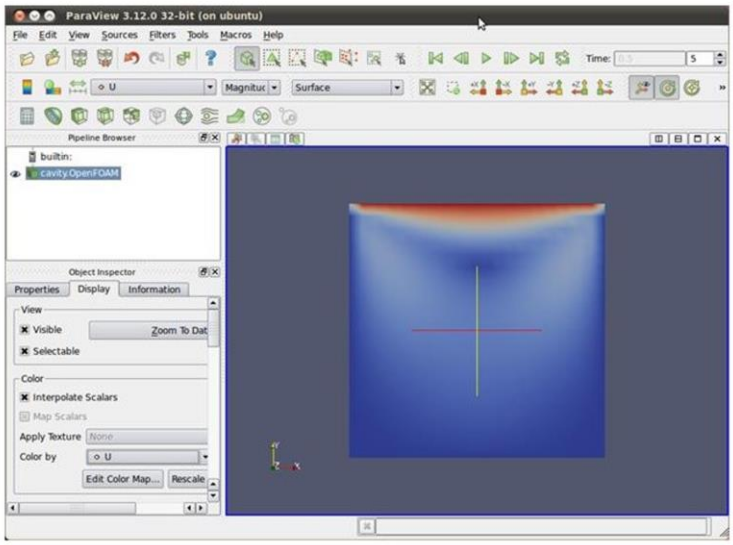
Результати матричного множення для матриць різних розмірів

No. Matrix	2 Instances	4 Instances
200x200	0.029763	0.094822
1000x1000	16.619291	6.760375
1500x1500	64.405873	21.981079
2000x2000	172.414044	56.308349
3000x3000	904.551970	242.351147



16

Перегляд результатів у ParaView



17

Висновки

В даній роботі було розглянуто основи реалізації ефективної хмарної обчислювальної системи на базі Eucalyptus та OpenNebula з відкритим кодом, що дозволили реалізувати прототип віртуального хмарного середовища, що має необхідний рівень захищеності даних.

Експериментальне дослідження реалізованих на реальному інформаційному комплексі розподілених обчислювальних середовищ з єдиним образом операційної системи MOSIX у гетерогенному віртуальному середовищі та дослідження продуктивності PVM та MPI з міграцією процесів MOSIX і без неї продемонстрували суттєве поліпшення технічних та експлуатаційних характеристик комплексу при вирішенні ресурсів

Розроблена в роботі та реалізована експериментально інтеграція пакетів Globus Toolkit та Sun Grid Engine на основі стандарту DRMAA дозволила реалізувати можливість побудови операційного оточення для запуску власних програм у віртуальному хмарному середовищі.

Результати, отримані в магістерській роботі внаслідок дослідження, дозволяють зазначити, що організоване хмарне середовище дозволяє ефективно здійснювати динамічне балансування обчислень на вузлах хмарної обчислювальної системи.

18

Список публікацій

1. Стаття на тему «Аналіз існуючих VPN рішень для організації захищеної передачі даних»
2. Тези на тему: «З'єднання корпоративних мереж з хмарами»

ДЯКУЮ ЗА УВАГУ!