МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

ГРІД-СИСТЕМИ ТА ТЕХНОЛОГІЇ ХМАРНИХ ОБЧИСЛЕНЬ

Методичний посібник

УЖГОРОД – 2016

Грід-системи та технології хмарних обчислень: методичний посібник до вивчення курсу для студентів спеціальностей 121 ”Інженерія програмного забезпечення” та 122 “Комп’ютерні науки та інформаційні технології”.

У методичному посібнику до курсу «Грід-системи та технології хмарних обчислень» розглянуто базові складові грід-систем та їх зв'язок з веб-технологіями. Наведено приклади використання технологій віртуалізації, технологій серверних систем, комунікаційних засобів для розподілених обчислень та розроблення програмно-апаратних рішень центрів обробки даних.

Розробники:

Пецко В. І., к.т.н., старший викладач кафедри інформаційних управляючих систем та технологій ДВНЗ «Ужгородський національний університет»;

Міца О. В. ., к.т.н., доцент, завідувач кафедри інформаційних управляючих систем та технологій ДВНЗ «Ужгородський національний університет».

Рецензенти:

Стецюк П.І., д.ф.-м.н., зав. відділу Інституту кібернетики ім. В.М. Глушкова НАН України;

Левчук О. М., к.т.н., доцент кафедри інформаційних управляючих систем та технологій, заступник декана факультету інформаційних технологій ДВНЗ «Ужгородський національний університет».

Рекомендовано кафедрою інформаційних управляючих систем та технологій

Рекомендовано Вченою радою факультету інформаційних технологій.

ЗМІСТ

ВСТУП

Сучасні інформаційно-комунікаційні технології передбачають використання технологій віртуалізації, технологій серверних систем, комунікаційних засобів для розподілених обчислень та розроблення програмно апаратних рішень центрів обробки даних. Для управління неоднорідними обчислювальними ресурсами у віддаленому режимі потрібні програмні рішення для впровадження систем віртуалізації, а також віддалених сервісних функцій, що загалом створює можливості для організації та застосування технологій хмарних обчислень.

Метою викладання навчальної дисципліни "Грід-системи та технології хмарних обчислень" є формування теоретичних знань і придбання практичних умінь і навичок з питань використання технологій розподілених обчислень, віртуалізації серверних систем, проектування корпоративних обчислювальних систем та застосування кластерних і гетерогенних розподілених обчислювальних систем для проведення наукових досліджень.

Основними завданнями вивчення дисципліни "Грід-системи та технології хмарних обчислень" є формування у студентів компетенції з використання стандартів та технологій залучення та застосування розподілених комп’ютерних ресурсів, що надаються за замовленням, для проведення наукових досліджень та використання обчислювального середовища організацій від рівня стартапу до корпорації.

Об'єктом навчальної дисципліни є процеси розподілених обчислень.

Предметом навчальної дисципліни є принципи та стандарти функціонування технологій та розробка рішень на базі хмарних обчислень.

1. БАЗОВІ СКЛАДОВІ GRID І РЕСУРСИ

1.1. Технологія GRID. Загальні відомості. Напрями розвитку технології GRID

GRID-технологія – це розподілена обчислювальна інфраструктура, що об'єднує ресурси різних типів з колективним доступом до цих ресурсів у рамках віртуальних організацій, які складаються з підприємств та окремих фахівців і спільно використовують ці загальні ресурси.

Ідейна основа GRID-технології – об'єднання ресурсів через створення комп'ютерної інфраструктури нового типу, що забезпечує глобальну інтеграцію інформаційних і обчислювальних ресурсів на основі мережних технологій та спеціального програмного забезпечення проміжного рівня (між базовим і прикладним ПЗ), а також набору стандартних служб для забезпечення надійного спільного доступу до географічно розподілених інформаційних та обчислювальних ресурсів: окремих комп'ютерів, кластерів, сховищ даних і мереж.

Напрями розвитку технології GRID

Виходячи з результатів аналізу проектів по створенню GRID-систем (Більша частина цих проектів має експериментальний характер) можна зробити висновок про **три напрями розвитку технології GRID:**

- обчислювальний GRID;
- GRID для інтенсивної обробки даних;
- семантичний GRID для операцій з даними з різних баз даних.

Метою першого напряму є досягнення максимальної швидкості обчислень за рахунок глобального розподілу цих обчислень між комп'ютерами. **Метою другого напряму** є обробка величезних об'ємів даних відносно нескладними програмами за принципом «одне завдання – один процесор.

1.2. Концепція GRID. Основні властивості GRID

GRID є технологією забезпечення гнучкого, безпечного і скоординованого загального доступу до ресурсів. У термінології GRID сукупність людей і організацій, які спільно розв'язують ту чи іншу загальну

задачу і надають у користування один одному свої ресурси, називають **віртуальною організацією.**

Наприклад, віртуальною організацією може бути сукупність всіх людей, що беруть участь в якому-небудь науковому об'єднанні. Віртуальні організації можуть розрізнятися за складом, масштабом, часом існування, родом діяльності, цілями, відносинами між учасниками (довірчі, не довірчі) і т. д. Склад віртуальних організацій може динамічно змінюватися.

Є два основні критерії, що виділяють GRID-системи серед інших систем, що забезпечують доступ до ресурсів, що розділяються:

· **GRID-система координує розрізнені ресурси**. Ресурси не мають загального центру управління, а GRID-система займається координацією їх використання, наприклад, балансуванням навантаження. Тому проста система управління ресурсами кластера не є системою GRID, оскільки здійснює централізоване управління всіма вузлами даного кластера, маючи до них повний доступ.

· **GRID-система будується на базі стандартних і відкритих протоколів**, сервісів та інтерфейсів. Не маючи стандартних протоколів, неможливо легко і швидко підключати нові ресурси в GRID-систему, розробляти новий вигляд сервісів і так далі.

Не слід змішувати технологію GRID з технологією паралельних обчислень. В рамках конкретної GRID-системи, звичайно, можливо організувати паралельні обчислення з використанням існуючих технологій (PVM, MPI), оскільки GRID-систему можна розглядати як якийсь мета-комп´ютер, що має безліч обчислювальних вузлів. Проте технологія GRID не є технологією паралельних обчислень, в її завдання входить лише координація використання ресурсів.

1.3. Архітектура GRID

Open GRID Services Architecture (OGSA) направлена на стандартизацію адресації (для сумісності), за допомогою визначення основи структури додатку GRID. По суті стандарт OGSA визначає сервіси GRID, їх можливості і те, на яких технологіях вони засновані. Проте OGSA не розрізняє особливостей технічної сторони специфікації; метою є визначення – що є

системою GRID. OGSA називають архітектурою, оскільки вона направлена на побудову і установку інтерфейсів, з яких можуть бути побудовані, системи, засновані на відкритих стандартах WSDL.

Архітектура GRID визначає системні компоненти, цілі і функції цих компонент і відображає способи взаємодії компонент один з одним. Архітектура GRID є архітектурою взаємодіючих протоколів, сервісів і інтерфейсів, що визначають базові механізми, за допомогою яких користувачі встановлюють з'єднання з GRID-системою, спільно використовують обчислювальні ресурси для вирішення різного роду завдань. **Архітектура протоколів GRID розділена на рівні** (рис. 1.1), компоненти кожного з них можуть використовувати можливості компонент будь-якого з розташованих нижче рівнів. В цілому ця архітектура задає вимоги для основних компонент технології (протоколів, сервісів, прикладних інтерфейсів і засобів розробки ПО), не надаючи строгий набір специфікацій, залишаючи можливість їх розвитку в рамках прийнятої концепції.

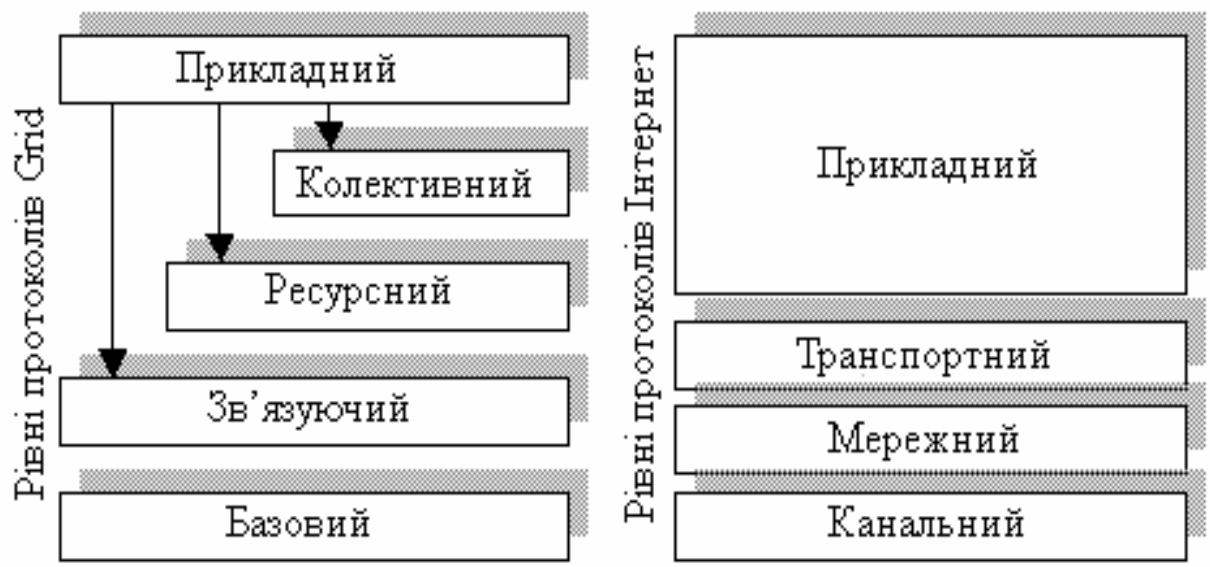

Рис. 1.1. Рівні архітектури протоколів GRID і їх відповідність рівням архітектури протоколів Інтернет

1.3.1. Базовий рівень

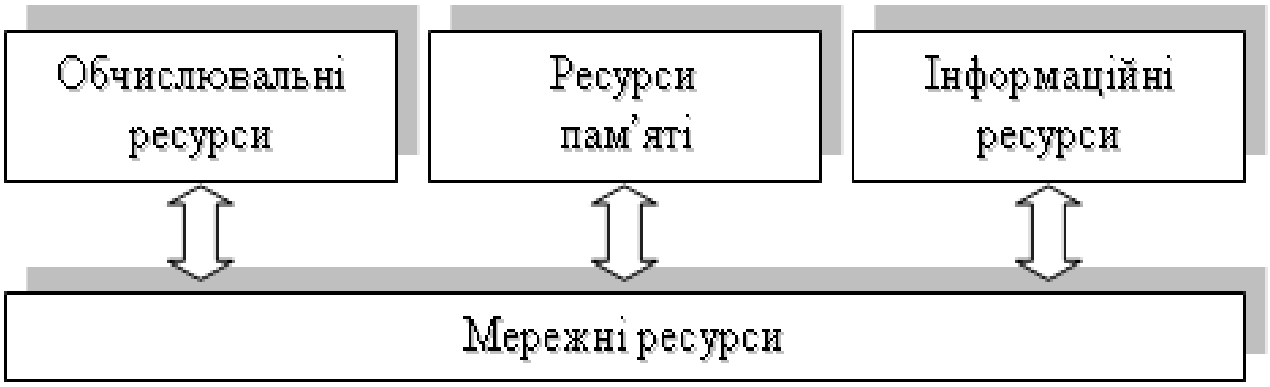

Рис. 1.2. – Ресурси GRID

Базовий рівень (Fabric Layer) архітектури GRID описує служби, що безпосередньо працюють з ресурсами. Ресурс є одним з основних понять архітектури GRID. Можна виділити декілька основних **типів ресурсів** (рис.1.2):

- обчислювальні ресурси;
- ресурси зберігання даних;
- інформаційні ресурси, каталоги;
- мережеві ресурси.

Обчислювальні ресурси надають користувачеві GRID-системи (точніше кажучи, завданню користувача) процесорні потужності. Обчислювальними ресурсами можуть бути як кластери, так і окремі робочі станції. При всій різноманітності архітектури будь-яка обчислювальна система може розглядатися як потенційний обчислювальний ресурс GRID-системи.

Ресурси пам'яті є простором для зберігання даних. Для доступу до ресурсів пам'яті також використовується програмне забезпечення проміжного рівня, що реалізує уніфікований інтерфейс управління і передачі даних. Основною характеристикою ресурсу пам'яті є його об'єм.

Інформаційні ресурси і каталоги є особливим видом ресурсів пам'яті. Вони служать для зберігання і надання метаданих і інформації про інші ресурси GRID-системи. Інформаційні ресурси дозволяють структуровано зберігати величезний об'єм інформації про поточний стан GRID-системи і ефективно виконувати завдання пошуку.

Мережевий ресурс є сполучною ланкою між розподіленими ресурсами GRID-системи. Основною характеристикою мережевого ресурсу є швидкість передачі даних. Географічно розподілені системи на основі даної технології здатні об'єднувати тисячі ресурсів різного типу, незалежно від їх географічного положення.

1.3.2. Рівень зв'язку

Рівень зв'язку (Connectivity Layer) визначає комунікаційні протоколи і протоколи аутентифікації.

Комунікаційні протоколи забезпечують обмін даними між компонентами базового рівня.

Протоколи аутентифікації, ґрунтуючись на комунікаційних протоколах, надають криптографічні механізми для ідентифікації і перевірки достовірності користувачів і ресурсів.

Протоколи рівня зв'язку повинні забезпечувати надійний транспорт і маршрутизацію повідомлень, а також привласнення імен об'єктам мережі. Не дивлячись на існуючі альтернативи, зараз протоколи рівня зв'язку в GRID-системах припускають використання тільки стека протоколів TCP/IP, зокрема: на мережевому рівні – IP і ICMP, транспортному рівні – TCP, UDP, на прикладному рівні – HTTP, FTP, DNS, RSVP. Враховуючи бурхливий розвиток мережевих технологій, в майбутньому рівень зв'язку, можливо, залежатиме і від інших протоколів.

Для забезпечення надійного транспорту повідомлень в GRID-системі повинні використовуватися рішення, що передбачають гнучкий підхід до безпеки комунікацій. В даний час ці рішення ґрунтуються як на існуючих стандартах безпеки, спочатку розроблених для Інтернет (SSL, TLS), так і на нових розробках.

1.3.3. Ресурсний рівень

Ресурсний рівень (Resource Layer) побудований над протоколами комунікації і аутентифікації рівня зв'язку архітектури GRID. Ресурсний рівень реалізує протоколи, що забезпечують виконання наступних функцій:

- узгодження політик безпеки використання ресурсу;

- процедура ініціації ресурсу;
- моніторинг стану ресурсу;
- контроль над ресурсом;
- облік використання ресурсу.

Протоколи цього рівня спираються на функції базового рівня для доступу і контролю над локальними ресурсами. На ресурсному рівні протоколи взаємодіють з ресурсами, використовуючи уніфікований інтерфейс і не розрізняючи архітектурні особливості конкретного ресурсу.

Розрізняють два основні класи протоколів ресурсного рівня:

- **інформаційні протоколи**, які отримують інформацію про структуру і стан ресурсу, наприклад, про його конфігурацію, поточне завантаження, політику використання;
- **протоколи управління**, які використовуються для узгодження доступу до ресурсів, що розділяються, визначаючи вимоги і допустимі дії по відношенню до ресурсу (наприклад, підтримка резервування, можливість створення процесів, доступ до даних). Протоколи управління повинні перевіряти відповідність запрошуваних дій політиці розділення ресурсу, включаючи облік і можливу оплату. Вони можуть підтримувати функції моніторингу статусу і управління операціями.

Список вимог до функціональності протоколів ресурсного рівня близький до списку для базового рівня архітектури GRID. Додалася лише вимога єдиної семантики для різних операцій з підтримкою системи оповіщення про помилки.

1.3.4. Колективний рівень

Колективний рівень (Collective Layer) відповідає за глобальну інтеграцію різних наборів ресурсів, на відміну від ресурсного рівня, сфокусованого на роботі з окремо узятими ресурсами. У колективному рівні розрізняють **загальні** і **специфічні** (для додатків) **протоколи. До загальних протоколів** відносяться, в першу чергу, протоколи виявлення і виділення ресурсів, системи моніторингу і авторизації співтовариств. **Специфічні протоколи** створюються для різних додатків GRID (наприклад, протокол

архівації розподілених даних або протоколи управління завданнями збереження стану і тому подібне).

Функції і сервіси, що реалізуються в протоколах колективного рівня рівня:

- **сервіси каталогів** дозволяють віртуальним організаціям виявляти вільні ресурси, виконувати запити по іменах і атрибутах ресурсів, таким як тип і завантаження;
- **сервіси сумісного виділення, планування і розподілу ресурсів** забезпечують виділення одного або більше ресурсів для певної мети, а також планування виконуваних на ресурсах завдань;
- **сервіси моніторингу і діагностики** відстежують аварії, атаки і перевантаження;
- **сервіси дублювання (реплікації) даних** координують використання ресурсів пам'яті в рамках віртуальних організацій, забезпечуючи підвищення швидкості доступу до даним відповідно до вибраних метрик, таких як час відповіді, надійність, вартість і т. д.;
- **сервіси управління робочим завантаженням** застосовуються для опису і управління багатокроковими, асинхронними, багатокомпонентними завданнями;
- **служби авторизації співтовариств** сприяють поліпшенню правил доступу до ресурсів, що розділяються, а також визначають можливості використання ресурсів співтовариства. Подібні служби дозволяють формувати політики доступу на основі інформації про ресурси, протоколи управління ресурсами і протоколи безпеки зв’язуючого рівня;
- **служби обліку і оплати** забезпечують збір інформації про використання ресурсів для контролю звернень користувачів;
- **сервіси координації** підтримують обмін інформацією в потенційно великому співтоваристві користувачів.

1.3.5. Прикладний рівень.

Прикладний рівень (Application Layer) описує призначені для користувача застосування (додатки), що працюють в середовищі віртуальної організації. Додатки функціонують, використовуючи сервіси, визначені на

рівнях, що пролягають нижче. На кожному з рівнів є певні протоколи, що забезпечують доступ до необхідних служб, а також прикладні програмні інтерфейси (Application Programming Interface – API), відповідні даним протоколам.

2. ЗВ'ЯЗОК GRID ТА ВЕБ-ТЕХНОЛОГІЙ

2.1. Грід-сервіси та їх особливості

Розглядаючи грід-систему у статиці, грід-систему можна представити як множину компонентів – користувачів, апаратних ресурсів, програм, служб. Основу частину компонентів Грід складають грід-ресурси. **Під ресурсом** можна розуміти будь-який реальний чи уявний об’єкт, до якого потрібно надати доступ іншим сутностям типу користувачів чи програм. Виділяють **дві основні категорії грід-ресурсів**: **фізичні** (обчислювальні ресурси, сховища, мережі, периферія), та **логічні** (дані, знання, прикладні програми). Таке представлення Грід як взаємопов’язаної множини ресурсів відоме як “ресурсно-орієнтоване”. В той же час, ресурси та користувачі Грід перебувають у постійній взаємодії: вони взаємодіють між собою, надсилаючи чи отримуючи запити та відповіді на них, при цьому діючи як незалежні агенти. З такою динамічною поведінкою грід-систем добре узгоджується “сервісно-орієнтований” підхід, базовим поняттям якого є сервіс. **Під сервісом** можна розуміти деякий програмний модуль із визначеною функціональністю.

Серед головних принципів сервісно-орієнтованої архітектури (SOA) виділяють наступні:

1. Стандартизований контракт. Інтерфейс взаємодії сервісів описується документально.

2. Слабка зв’язність. Взаємозв’язки між сервісами мають бути такими, що мінімізують взаємозалежності.

3. Абстрагування сервісів. Внутрішня логіка сервісу має бути прихована від зовнішнього світу, який обізнаний лише з його контрактом.

4. Повторне використання. Логіка розбивається на сервіси з думкою про вигоду повторного використання.

5. Автономність сервісів. Сервіси контролюють логіку, яку вони інкапсулюють.

6. Відсутність внутрішнього стану. Задля мінімізації використання ресурсів та кращої масштабованості сервіси не повинні зберігати свій стан між викликами.

7. Автоматизоване виявлення. Сервіси супроводжуються метаданими, що уможливлюють їх автоматизоване виявлення та ідентифікацію.

8. Здатність до компонування. Сервіси мають бути добре придатними для поєднання, незалежно від складності композиціювання.

Зважаючи на ці особливості та відштовхуючись від сервісно-орієнтованого підходу, підходящою **абстракцією для одиниці функціональності Грід є грід-сервіс:** спеціалізована автономна служба, що є базовою складовою частиною середовища зі слабкими зв'язками між компонентами, яким є Грід.

Виділяють також **базові грід-сервіси,** що надають базову функціональність грід-системи користувачам (запуск задач, передача даних, моніторинг та ін.), та **прикладні, додаткові, спеціалізовані грід-сервіси,** що надають додаткову функціональність (напр., вирішуючи вузькоспеціалізовані обчислювальні задачі таких прикладних областей, як астрономія, біомедицина, фізика тощо) та можуть використовувати у своїй роботі базові грід-сервіси (в т. ч. агрегуючи їх для створення складних середовищ вирішення задач, грід-порталів).

Серед загальних особливостей та вимог до грід-сервісів порівняно з принципами SOA варто виділити такі :

1. Тривалий час існування та складний життєвий цикл. Грід-сервіси, зазвичай, мають складнішу логіку, ніж просто "запит-відповідь". Тому грід-сервіси можуть існувати довше, аніж від моменту надходження запиту до сервісу до надсилання сервісом відповіді.

2. Підтримка внутрішнього стану. Грід-сервіси можуть слугувати для доступу до реальних об'єктів грід, таких як задачі, файли, апаратні ресурси,

програми тощо. В цьому випадку, грід-сервіс має змінювати стан об’єктів, за які він відповідає, у відповідь на вхідні запити.

3. Оповіщення. Грід-сервіси мають підтримувати механізм оповіщень, за допомогою яких вони асинхронно, тобто, без потреби у вхідному запиті, надсилають своїм клієнтам повідомлення про зміни у стані.

4. Узгодженість із інфраструктурою безпеки Грід (GSI).

Життєвий цикл грід-сервісів

Якщо сервіс слугує для надання синхронних відповідей на запити типу “надати перелік доступних грід-вузлів”, то йому не потрібно підтримувати механізм оповіщень, існувати постійно між вхідними запитами, підтримувати свій внутрішній стан. Все, що має виконувати такий сервіс – зчитати збережений на поточний момент перелік грід-вузлів з бази даних і передати цю інформацію клієнту. Однак, якщо сервіс слугує для управління конкретною грід-задачею (сама задача представлена як окремий сервіс), то цей сервіс зобов’язаний існувати так довго, як існує сама задача у грід, і кожний новий вхідний запит має змінювати або опитувати поточний внутрішній стан сервісу.

2.2. Грід-сервіси та веб-сервіси: можливість інтеграції

Веб-сервіси та грід-сервіси ідеологічно мають і багато спільного, і декотрі відмінності. (таблиці 2.1). Розрізняють грід-сервіси, що адаптовані під стандарти Веб та веб-сервіси, адаптовані під вимоги Грід. Перший підхід полягає у адаптації архітектури грід-сервісів під існуючі стандарти веб-сервісів, другий – у розширенні стандартів веб-сервісів для задоволення додаткових вимог до грід-сервісів (прикладом є стандарт-розширення WSRF).

Таблиця 2.1. – Спільні та відмінні риси грід- та веб-сервісів

	Веб-сервіси	**Грід-сервіси**
Спільні принципи	Модульність, слабка зв'язність, прагнення до повторного використання, абстрагування, здатність до поєднання, моніторинг, категоризація	
Життєвий цикл	Від запита до відповіді, збереження даних у БДМ	Існування між запитами, впродовж часу існування грід-ресурсу
Внутрішній стан	Відсутність спеціальної підтримки як ідеологічно, так і в стандартах	Має підтримуватись через особливості грід-ресурсів
Оповіщення	Прийнято додаткові стандарти WS-Notification	Мають підтримуватись через специфіку роботи Грід
Безпека	Стандарти WS-Security	Підтримка GSIM
Стандартизація	Визначені, зрілі стандарти	Наявність різних реалізацій проміжного програмного забезпечення грід не сприяє стандартизації

Таблиця 2.2. – Можливі шляхи узгодження грід та веб-сервісів

Наслідки для...	**Адаптація Грід під стандарти веб-сервісів.**	**Адаптація Веб під вимоги грід-сервісів**
Грід	Архітектура грід-сервісів віддаляється від концепції “ресурс-як-сервіс”	Можливо реалізувати складну поведінку грід-сервісів, відповідну до життєвого циклу грід-ресурсів
Веб	Грід-сервіси автоматично сумісні з існуючими веб-сервісами, адаптувати численний WS-інструментарій не потрібно	Існуючі веб-рішення можуть мати проблеми із сумісністю з грід-сервісами, якщо не будуть підтримувати усіх нововведень.

2.3. Базові елементи технології веб-сервісів

Архітектура веб-сервісів базується на трьох основних елементах:

- SOAP (Simple Object Access Protocol, простий протокол доступу до об’єктів) як протокол обміну XML-повідомленнями;
- WSDL (Web Service Description Language, мова опису веб-сервісів) як стандартна мова опису контрактів сервісів;
- UDDI (Universal Description Discovery & Integration, універсальний опис, виявлення, інтеграція) як стандартний механізм для пошуку сервісів.

Взаємодія клієнта з веб-сервісом відбувається таким чином. Головні актори у взаємодії:

- програмний клієнт (не сам користувач, а програма, в т. ч. – інший веб-сервіс, тобто веб-сервіси слугують для взаємодії між машинами, а не людьми);
- реєстр сервісів (UDDI-сховище описів веб-сервісів, в якому публікуються контракти та інші метадані веб-сервісів для їх подальшого пошуку та використання клієнтами);

- сам веб-сервіс (програма, що здатна обмінюватись повідомленнями по протоколу SOAP відповідно до WSDL-контракту, постачальник веб-сервісу відповідальний за публікацію його опису у UDDI та забезпечення його доступності).

2.4. Протокол SOAP (Simple Object Access Protocol)

SOAP – це протокол обміну XML-повідомленнями. Відгук від сервісу повертається SOAP серверу, використовуючи SOAP протокол, а це повідомлення повертається SOAP - клієнту, що послав запит.

SOAP – простий, заснований на XML, протокол, для обміну інформацією в децентралізованому, розподіленому середовищі. SOAP підтримує різні стилі обміну інформацією, включаючи:

- обмін інформацією, що формується після видаленого виклику процедури. Цей тип обміну робить доступним процес запит - відповідь, в якому крайовий користувач отримує процедурне повідомлення і дає відповідь відповідним повідомленням.
- інформаційний обмін на основі механізму обміну повідомленнями. Цей тип обміну використовують організації і додатки, яким потрібно обмінюватися бізнес-документами, послане повідомлення не має на увазі негайний відгук на нього.

SOAP характеризується:

- Протокольною незалежністю.
- Мовною незалежністю.
- Незалежністю від ОС і платформи.
- Підтримкою SOAP XML-повідомлень взаємодіючих частин (використовуючи багатоскладну структуру MIME).

Повідомлення SOAP складається з (1) SOAP конверта, який містить дві структури даних, (2) SOAP - заголовка і тіла SOAP і (3) інформації про імена, службовців для їх опису. Заголовок є необов'язковою частиною, він передає інформацію про запит, визначений в тілі SOAP. Наприклад, він може містити інформацію по безпеці, ділову інформацію або профіль користувача. Тіло містить запит Web - сервісу або відповідь на нього.

Специфікація описує структуру і тип даних при обміні повідомленнями, використовуючи XML – схему. Спосіб, в якому SOAP використовується для посилки запитів і отримання відповідей від Web - сервіса:

- Клієнт SOAP використовує документ XML, який узгоджується із специфікацією SOAP і містить запит про послугу.
- Клієнт SOAP посилає документ серверу SOAP, а той обробляє його за допомогою HTTP, HTTPS.
- Web-сервіс отримує повідомлення SOAP, направляє його, у вигляді службового запиту, додатку, що надає запрошувану послугу.

3. GRID І БАЗИ ДАНИХ

3.1. Розподілені БД

Розподілена база даних – набір логічно пов'язаних між собою даних (і їх описів), що розділяються, які фізично розподілені в деякій комп'ютерній мережі.

Розподілена СКБД – програмний комплекс, призначений для управління розподіленими базами даних і який дозволяє зробити розподіленість інформації прозорою для кінцевого користувача.

Система керування розподіленими базами даних (СКРБД) складається з єдиної логічної бази даних, розділеної на деяку кількість фрагментів. Кожен фрагмент бази даних зберігається на одному або декількох комп'ютерах, які сполучені між собою лініями зв'язку і кожен з яких працює під керуванням окремої СКБД. Будь-який з сайтів здатний незалежно обробляти запити користувачів, що вимагають доступу до даних (що створює певну міру локальної автономії), які зберігаються локально, а також здатний обробляти дані мережі, що зберігаються на інших комп'ютерах мережі.

Користувачі взаємодіють з розподіленою базою даних через програми (застосування). Застосування можуть бути класифіковані як ті, які не вимагають доступу до даних на інших сайтах (локальні застосування), і ті, які вимагають подібного доступу (глобальні застосування). У розподіленій

СКБД повинно існувати хоча б одне глобальне застосування, тому будь-яка СКРБД повинна мати наступні особливості:

- набір логічно пов'язаних розподілених даних.
- дані, що зберігаються, розбиті на деяку кількість фрагментів.
- між фрагментами може бути організована реплікація даних. Фрагменти і їх репліки розподілені по різних сайтах.
- сайти пов'язані між собою мережевими з'єднаннями.
- Робота з даними на кожному сайті управляється СКБД.
- СКБД на кожному сайті здатна підтримувати автономну роботу локальних застосувань.
- СКБД кожного сайту підтримує хоч би одне глобальне застосування.

Розглянемо приклад розподіленої обробки даних.

Використовуючи технологію розподілених баз даних, компанія замість єдиного центрального комп’ютера може розмістити свою базу даних на декількох незалежних комп'ютерних системах. Подібні комп'ютерні системи можуть бути встановлені в кожному з існуючих відділень компанії у різних місцях. Мережеві з'єднання, що зв'язують комп'ютерні системи, дозволять відділенням компанії взаємодіяти між собою, а розгортання в системі СКРБД забезпечить доступ до даних, розміщених в інших відділеннях компанії. В результаті клієнт, що проживає в одному місті, зможе звернутися в найближче відділення компанії і ознайомитися з об'єктами нерухомості, що надаються в оренду в іншому населеному пункті, що позбавить його від необхідності для отримання цих відомостей зв'язуватися з потрібним містом по телефону або посилати поштове повідомлення.

Розподілена обробка – обробка з використанням централізованої бази даних, доступ до якої може здійснюватися з різних комп’ютерів мережі.

Паралельна СКБД – система управління базою даних, що функціонує з використанням декількох процесорів і пристроїв жорстких дисків, що дозволяє їй (якщо це можливо) розпаралелювати виконання деяких операцій з метою підвищення загальної продуктивності обробки.

Переваги і недоліки, властиві розподілених СКБД

Переваги:

- Віддзеркалення структури організації.
- Подільність і локальна автономність.
- Підвищення доступності даних.

У централізованих СКБД відмова центрального комп'ютера викликає припинення функціонування усієї СКБД. Проте відмова одного з сайтів СКРБД або лінії зв'язку між сайтами зробить недоступною лише деякі сайти, тоді як уся система в цілому збереже свою працездатність. Розподілені СКБД проектуються так, щоби забезпечувати продовження функціонування системи, незважаючи на подібні відмови. Якщо виходить з ладу один з вузлів, система зможе перенаправити запити до вузла, що відмовив, на адресу іншого сайту.

- ***Підвищення надійності***

Якщо організована реплікація даних, внаслідок чого дані і їх копії будуть розміщені на більш, ніж одному сайті, відмова окремого вузла або сполучного зв'язку між вузлами не приведе до недоступності даних в системі.

- ***Підвищення продуктивності***

Якщо дані розміщені на самому навантаженому сайті, який успадкував від систем-попередників високий рівень паралельності обробки, то розгортання розподіленої СКБД може сприяти підвищенню швидкості доступу до бази даних (в порівнянні з доступом до віддаленої централізованої СКБД).

- ***Економічні вигоди***

Виявляється, що набагато вигідніше встановлювати в підрозділах організації власні малопотужні комп'ютери, крім того, значно дешевше додати в мережу нові робочі станції, ніж модернізувати систему з центральним сервером.

- ***Модульність системи***

У розподіленому середовищі розширення існуючої системи здійснюється набагато простіше. Додавання в мережу нового сайту не чинить впливу на функціонування вже існуючих. Подібна гнучкість дозволяє організації легко розширюватися. Перевантаження через збільшення розміру бази даних зазвичай усуваються шляхом додавання в мережу нових обчислювальних потужностей і пристроїв дискової пам'яті. У централізованих СКБД

зростання розміру бази даних може вимагати заміни і устаткування (потужнішою системою), і використовуваного програмного забезпечення (потужнішою або гнучкішою СКБД).

Недоліки:

Підвищення складності

Розподілені СКБД, здатні приховати від кінцевих користувачів розподілену природу використовуваних ними даних і забезпечити необхідний рівень продуктивності, надійності і доступності, безумовно є складнішими програмними комплексами, ніж централізовані СКБД. Той факт, що дані можуть піддаватися реплікації, також додає додатковий рівень складності в програмне забезпечення СКРБД.

Збільшення вартості

Збільшення складності означає і збільшення витрат на придбання і супровід СКРБД (в порівнянні із звичайними централізованими СКБД). Разширення розподіленої СКБД зажадає додаткового устаткування, необхідного для установки мережевих з'єднань між сайтами. Слід чекати і зростання затрат на оплату каналів зв'язку, викликаних зростанням мережевого трафіку. Крім того, зростуть витрати на оплату праці персоналу, який буде потрібно для обслуговування локальних СКБД і мережевих з'єднань.

Проблеми захисту

У централізованих системах доступ до даних легко контролюється. Проте в розподілених системах потрібно буде організувати контроль доступу не лише до даних, реплікованих на декілька різних сайтів, але і захист мережевих з'єднань самих по собі.

Ускладнення контролю за цілісністю даних

Цілісність бази даних означає коректність і узгодженість даних, що зберігаються в ній. Вимоги забезпечення цілісності зазвичай формулюються у вигляді деяких обмежень, виконання яких гарантуватиме захист інформації в базі даних від пошкодження.

Відсутність стандартів

Хоча цілком очевидно, що функціонування розподілених СКБД залежить від ефективності використовуваних каналів зв'язку, тільки останнім часом

стали вимальовуватися контури стандарту на канали зв'язку і протоколи доступу до даних. Відсутність стандартів істотно обмежує потенційні можливості розподілених СКБД.

Недолік досвіду

Нині в експлуатації знаходиться вже декілька систем-прототипів і розподілених СКБД спеціального призначення, що дозволило уточнити вимоги до використовуваних протоколів і встановити коло основних проблем. Проте на поточний момент розподілені системи загального призначення ще не набули широкого поширення. Відповідно, ще не накопичений необхідний досвід промислової експлуатації розподілених систем, порівнюваний з досвідом експлуатації централізованих систем. Такий стан справ є серйозним стримуючим чинником для багатьох потенційних прибічників цієї технології.

Ускладнення процедури розробки бази даних

Розробка розподілених баз даних, окрім звичайних труднощів, пов'язаних з процесом проектування централізованих баз даних, вимагає прийняття рішення про фрагментацію даних, розподіл фрагментів по окремих сайтах і організації процедур реплікації даних.

3.2. Гомогенні і гетерогенні розподілені СКБД

Розподілені СКБД можна класифікувати як **гомогенні і гетерогенні. У гомогенних системах** усі сайти використовують один і той же тип СКБД. **У гетерогенних системах** на сайтах можуть функціонувати різні типи СКБД, що використовують різні моделі даних, тобто гетерогенна система може включати сайти з реляційними, сітковими, ієрархічними або об'єктно-орієнтованими СКБД.

Гомогенні системи значно простіше проектувати і супроводжувати. Крім того, подібний підхід дозволяє поетапно нарощувати розміри системи, послідовно додаючи нові сайти до вже існуючої розподіленої системи. Додатково з'являється можливість підвищувати продуктивність системи за рахунок організації на різних сайтах паралельної обробки інформації.

Гетерогенні системи зазвичай виникають в тих випадках, коли незалежні сайти, що вже експлуатують свої власні системи з базами даних,

інтегруються в новостворювану розподілену систему. У гетерогенних системах для організації взаємодії між різними типами СКБД потрібно буде організувати трансляцію передаваних повідомлень. Для забезпечення прозорості відносно типу використовуваної СКБД користувачі кожного з сайтів повинні мати можливість вводити запити, що цікавлять їх, на мові тієї СКБД, яка використовується на цьому сайті. Система повинна узяти на себе локалізацію необхідних даних і виконання трансляції передаваних повідомлень. У загальному випадку дані можуть бути потрібні з другого сайту, який характеризується такими особливостями, як:

- інший тип використовуваного устаткування;
- інший тип використовуваної СКБД;
- інший тип використовуваного устаткування і СКБД.

Розробка розподілених реляційних баз даних.

Аспекти проектування розподілених систем:

- **Фрагментація.** Будь-яке відношення може бути розділене на деяку кількість частин, які називаються фрагментами, які потім розподіляються по різних сайтах. Існують два основні типи фрагментів: горизонтальні і вертикальні. Горизонтальні фрагменти є підмножинами кортежів, а вертикальні – підмножини атрибутів.

- **Розподіл**. Кожен фрагмент зберігається на сайті, вибраному з врахуванням "оптимальної" схеми їх розміщення.

- **Реплікація.** СКРБД може підтримувати актуальну копію деякого фрагмента на декількох різних сайтах.

Розподіл даних

Існують **чотири альтернативні стратегії розміщення даних в системі:** централізоване, роздільне (фрагментоване), розміщення з повною реплікацією і з вибірковою реплікацією.

Централізоване розміщення

Ця стратегія передбачає створення на одному з сайтів єдиної бази даних під управлінням СКБД, доступ до якої матимуть усі користувачі мережі (ця стратегія під назвою "розподілена обробка" вже розглядалася нами вище.) В цьому випадку локальність посилань мінімальна для усіх сайтів, за виключенням центрального, оскільки для отримання будь-якого доступу

до даних потрібна установка мережевого з'єднання. Відповідно рівень витрат на передачу даних буде високий. Рівень надійності і доступності в системі низький, оскільки відмова на центральному сайті викличе параліч роботи усієї системи.

Роздільне (фрагментоване) розміщення

В цьому випадку база даних розбивається на фрагменти, що не перетинаються, кожен з яких розміщується на одному з сайтів системи. Якщо елемент даних буде розміщений на тому сайті, на якому він найчастіше використовується, отриманий рівень локальності посилань буде високий. За відсутності реплікації вартість зберігання даних буде мінімальна, але при цьому буде невисокий також рівень надійності і доступності даних в системі. Проте він буде вищий, ніж в попередньому варіанті, оскільки відмова на будь-якому з сайтів викличе втрату доступу тільки до тієї частини даних, яка на ньому зберігалася. При правильно вибраному способі розподілу даних рівень продуктивності в системі буде відносно високим, а рівень затрат на передачу даних – низьким.

Розміщення з повною реплікацією

Ця стратегія передбачає розміщення повної копії усієї бази даних на кожному з сайтів системи.

Розміщення з вибірковою реплікацією

Ця стратегія є комбінацією методів фрагментації, реплікації і централізації.

4. БЕЗПЕКА ФАЙЛОВОЇ СИСТЕМИ. СЕРТИФІКАТ ВІДКРИТИХ КЛЮЧІВ

4.1. Основні положення

Інфраструктура безпеки GRID (GRID Security Infrastructure – GSI) забезпечує безпечну роботу в незахищених мережах загального доступу (Інтернет), надаючи такі сервіси, як аутентифікація, авторизація, конфіденційність передачі інформації і єдиний вхід в GRID-систему. Під єдиним входом мається на увазі, що користувачеві потрібно лише один раз пройти процедуру аутентифікації, а далі система сама потурбується про те, щоб аутентифікувати його на всіх ресурсах, якими він збирається

скористатися. **GSI** заснована на надійній і широко використовуваній інфраструктурі криптографії з відкритим ключем (Public Key Infrastructure – PKI).

Як ідентифікатори користувачів і ресурсів в GSI використовуються цифрові сертифікати X.509. **У роботі з сертифікатами X.509 і в процедурі видачі/отримання сертифікатів задіяно три сторони:**

– **Центр Сертифікації (Certificate Authority – CA)** – спеціальна організація, що володіє повноваженнями видавати (підписувати) цифрові сертифікати. Різні CA зазвичай незалежні між собою. Відносини між CA і його клієнтами регулюються спеціальним документом.

– **Підписчик** – це людина або ресурс, який користується сертифікаційними послугами CA. CA включає в сертифікат дані, що надаються підписчиком (ім'я, організація і ін.) і ставить на нім свій цифровий підпис.

– **Користувач** – це людина або ресурс, що покладається на інформацію з сертифікату при отриманні його від підписчика. Користувачі можуть приймати або відкидати сертифікати, підписані яким-небудь CA.

– **У GSI використовуються два типи сертифікатів X.509:**

– **Сертифікат користувача** (User Certificate) – цей сертифікат повинен мати кожен користувач, що працює з GRID-системою. Сертифікат користувача містить інформацію про ім'я користувача, організацію, до якої він належить, і центр сертифікації, що видав даний сертифікат.

– **Сертифікат вузла** (Host Certificate) – цей сертифікат повинен мати кожен вузол (ресурс) GRID-системи. Сертифікат вузла аналогічний сертифікату користувача, але в нім замість імені користувача указується доменне ім'я конкретного обчислювального вузла.

4.2. Вимоги безпеки GRID

Для розподілених операцій просто необхідне управління і взаємодія з множинними інфраструктурами безпеки. Наприклад, для комерційного банку даних, ізоляція клієнтів усередині цього банку даних – основна вимога; GRID повинен здійснювати не тільки контроль доступу, але і надавати ізоляцію.

Багато завдань вимагають, щоб додатки могли використовуватися і поза власним firewall'ом.

При створенні і впровадженні додатків в систему GRID потрібна аутентифікація /авторизація. У випадку з комерційним банком даних, банк даних пізнає клієнта і авторизує його запит, коли клієнт виставив запит на завантаження завдання. Банк даних так само визначає персональні настройки користувачів (безпека, планування і ін.).

Шифрування

IT інфраструктура і її управління вимагає шифрування комунікацій, принаймні найосновніших.

Firewall'и мережевого рівня і додатку

Це давня проблема. Особливо складною її робить величезна кількість правив і умов, а також різні обмеження на міжнародних сайтах.

Сертифікація

Авторитетні організації сертифікують роботу окремих сервісів. Наприклад, компанія може дотримуватися правил, які вимагають, щоб використовувалися сервіси електронної комерції, сертифіковані Yahoo.

4.3. Інфраструктура захисту GRID (GSI)

Фахівці в області захисту даних комп'ютера часто вважають, що "безпечна комунікація" – це просто будь-яка комунікація, де кодуються дані. Проте захист містить в собі набагато більше, ніж просто кодування і розшифровка даних.

Три ключові елементи безпечної комунікації

Більшість авторів розглядають три основні елементи безпечної комунікації (або "безпечного діалогу"): *конфіденційність*, *цілісність*, і *аутентифікація*. Безпечний діалог повинен представляти всі три елементи, але не завжди (іноді це, навіть не було б бажано). Різні сценарії захисту могли вимагати різні комбінації характеристик (наприклад "тільки конфіденційність", "конфіденційність і цілісність без аутентифікації", "тільки цілісність", і тому подібне).

Конфіденційність

Безпечний діалог повинен бути *приватним*. Іншими словами, тільки відправник і одержувач можуть розуміти діалог. Якщо хто-небудь прослуховуватиме через комунікації, то він буде не в змозі мати від цього користь. Конфіденційність, загалом, досягається алгоритмами кодування/розшифровки.

Цілісність

Безпечна комунікація повинна гарантувати цілісність переданого повідомлення. Це означає, що одержуюча сторона повинна знати напевно, що повідомлення, яке вона отримує, – є тим, що пересилаюча сторона передала. Важливе те, що зловмисник міг перехопити комунікацію маючи намір змінити її вміст без наміру прослуховування.

4.3.1. Аутентифікація

Безпечна комунікація повинна гарантувати, що сторони, залучені в комунікації є потрібними. Іншими словами, потрібно захистити від зловмисників, які пробують представитися однією із сторін в процесі безпечного діалогу. Знову-таки, це відносно просто виконати за допомогою деяких мережевих сніферів. Проте сучасні алгоритми кодування захищають також від цього виду нападів.

4.3.2. Авторизація

Авторизація посилається на механізми, які визначають, коли користувач авторизований для виконання певного завдання. Авторизація пов'язана з аутентифікацією, оскільки, загалом, потрібно переконатися, що користувач – той, хто потрібний (аутентифікація) перед тим, як можна буде ухвалити рішення щодо того, чи може він (або не може) виконати певне завдання (авторизація).

Наприклад, як тільки буде з'ясовано, що користувач входить до складу Відділу Математики, потрібно буде потім дозволити йому звернутися до всіх Мат. служб. Проте, можливо, йому заборонили б звертаються до інших служб, що не пов'язані з його відділом (BiologyService, ChemistryService, і тому подібне)

4.3.3. Авторизація порівняно з Аутентифікацією

Сплутати *аутентифікацію* і *авторизацію* дуже просто, важливо пам'ятати, що *аутентифікація* посилається на з'ясування достовірності чиєї-небудь особи (якщо вона дійсно та що потрібна) , а авторизація звертається до з'ясування того, хто авторизований виконувати певне завдання.

5. ТЕХНОЛОГІЇ ВІРТУАЛІЗАЦІЇ

В основі віртуалізації лежить можливість одного комп'ютера виконувати роботу декількох комп'ютерів завдяки розподілу його ресурсів на декілька середовищ. За допомогою віртуальних серверів і віртуальних настільних комп'ютерів можна розмістити кілька ОС і кілька додатків в єдиному розташування. Таким чином, фізичні та географічні обмеження перестають мати якесь значення. Крім енергозбереження та скорочення витрат завдяки більш ефективному використанню апаратних ресурсів, віртуальна інфраструктура забезпечує високий рівень доступності ресурсів, більш ефективну систему управління, підвищену безпеку і вдосконалену систему відновлення в критичних ситуаціях.

У комп'ютерних технологіях під терміном «віртуалізація» зазвичай розуміється абстракція обчислювальних ресурсів і надання користувачеві системи, яка «інкапсулює» (приховує в собі) власну реалізацію. Простіше кажучи, користувач працює з зручним для себе представленням об'єкта, і для нього не має значення, як об'єкт влаштований в дійсності.

Зараз можливість запуску декількох віртуальних машин на одній фізичній викликає великий інтерес серед комп'ютерних фахівців, не тільки тому, що це підвищує гнучкість IT – інфраструктури, але й тому, що віртуалізація, насправді, дозволяє економити гроші.

5.1. Переваги технологій віртуалізації

Наведемо основні переваги технологій віртуалізації:

1. Ефективне використання обчислювальних ресурсів. Замість 3х, а то і 10 серверів,

2. Скорочення витрат на інфраструктуру. Віртуалізація дозволяє скоротити кількість серверів і пов'язаного з ними IT -обладнання в інформаційному центрі. У результаті цього

3. Зниження витрат на програмне забезпечення. Деякі виробники програмного забезпечення ввели окремі схеми ліцензування спеціально для віртуальних середовищ.

4. Підвищення гнучкості і швидкості реагування системи: Віртуалізація пропонує новий метод управління IT – інфраструктурою і допомагає IT – адміністраторам затрачати менше часу на виконання повторюваних завдань – наприклад, на ініціацію, налаштування, відстеження і технічне обслуговування.

5. Несумісні додатки можуть працювати на одному комп'ютері. При використанні віртуалізації на одному сервері можлива установка linux і windows серверів, шлюзів, баз даних і інших абсолютно несумісних в рамках однієї не віртуалізованої системи додатків.

6. Підвищення доступності додатків і забезпечення безперервності роботи підприємства. Завдяки надійній системі резервного копіювання та міграції віртуальних середовищ цілком без перерв в обслуговуванні ви зможете скоротити періоди планового простою і забезпечити швидке відновлення системи в критичних ситуаціях. "Падіння» одного віртуального сервера не веде до втрати інших віртуальних сервері

7. Можливості легкої архівації. Оскільки жорсткий диск віртуальної машини зазвичай представляється у вигляді файлу певного формату, розташований на якому -небудь фізичному носії, віртуалізація дає можливість простого копіювання цього файлу.

8. Підвищення керованості інфраструктури: використання централізованого управління віртуальною інфраструктурою дозволяє скоротити час на адміністрування серверів, забезпечує балансування навантаження і "живу" міграцію віртуальних машин.

Віртуальною машиною будемо називати програмне або апаратне середовище, яка приховує справжню реалізацію якогось процесу або об'єкту від його справжнього подання.

Віртуальна машина – це повністю ізольований програмний контейнер, який працює з власною ОС і додатками, подібно фізичному комп'ютеру. Віртуальна машина діє так само, як фізичний комп'ютер, і містить власні віртуальні (тобто програмні) ОЗУ, жорсткий диск і мережевий адаптер.

ОС не може розрізнити віртуальну і фізичну машини. Те ж саме можна сказати про додатки та інших комп'ютерах в мережі. Навіть сама віртуальна машина вважає себе «справжнім» комп'ютером. Але незважаючи на це віртуальні машини складаються виключно з програмних компонентів і не включають обладнання. Це дає їм низку унікальних переваг над фізичною обладнанням.

5.2. Основні різновиди віртуалізації

Розглянемо основні різновиди віртуалізації, такі як:

- віртуалізація серверів (повна віртуалізація і паравіртуалізації)
- віртуалізація на рівні операційних систем,
- віртуалізація додатків,
- віртуалізація уявлень.

Віртуалізація серверів

Віртуалізація серверів має на увазі запуск на одному фізичному сервері декількох віртуальних серверів. Віртуальні машини або сервера являють собою програми, запущені на хостовій операційній системі, які емулюють фізичні пристрої сервера. На кожній віртуальній машині може бути встановлена операційна система, на яку можуть бути встановлені додатки і служби. Типові представники це продукти VmWare (ESX, Server, Workstation) і Microsoft (Hyper -V, Virtual Serer, Virtual PC).

Повна віртуалізація (Full, Native Virtualization). Використовуються не модифіковані екземпляри гостьових операційних систем, а для підтримки роботи цих ОС служить загальний шар емуляції їх виконання поверх хостової ОС, в ролі якої виступає звичайна операційна система. Така технологія застосовується, зокрема, в VMware Workstation, VMware Server (колишній GSX Server, Parallels Desktop, Parallels Server, MS Virtual PC, MS

Virtual Server, Virtual Iron. До переваг даного підходу можна зарахувати відносну простоту реалізації, універсальність і надійність рішення ; всі функції управління бере на себе хост -ОС. Недоліки – високі додаткові накладні витрати на використовувані апаратні ресурси, відсутність обліку особливостей гостьових ОС, менша, ніж потрібно, гнучкість у використанні апаратних засобів.

Паравіртуалізації (paravirtualization). Модифікація ядра гостьової ОС виконується таким чином, що в неї включається новий набір API, через який вона може безпосередньо працювати з апаратурою, не конфліктуючи з іншими віртуальними машинами. При цьому немає необхідності задіяти повноцінну ОС в якості хостового ПО, функції якого в даному випадку виконує спеціальна система, що отримала назву гіпервізора (hypervisor). Саме цей варіант є сьогодні найбільш актуальним напрямком розвитку серверних технологій віртуалізації і застосовується в VMware ESX Server, Xen (і рішеннях інших постачальників на базі цієї технології), Microsoft Hyper -V. Переваги даної технології полягають у відсутності потреби в хостової ОС – ВМ, встановлюються фактично на "голе залізо".

Віртуалізація на рівні ядра ОС (operating system – level virtualization). Цей варіант передбачає використання одного ядра хостової ОС для створення незалежних паралельно працюючих операційних середовищ. Для гостьового ПО створюється тільки власне мережеве та апаратне оточення. Такий варіант використовується в Virtuozzo (для Linux і Windows), OpenVZ (безкоштовний варіант Virtuozzo) і Solaris Containers. Переваги – висока ефективність використання апаратних ресурсів, низькі накладні технічні витрати, відмінна керованість, мінімізація витрат на придбання ліцензій. Недоліки – реалізація тільки однорідних обчислювальних середовищ.

Віртуалізація додатків має на увазі застосування моделі сильної ізоляції прикладних програм з керованою взаємодією з ОС, при якій віртуалізується кожен екземпляр додатків, всі його основні компоненти: файли (включаючи системні), реєстр, шрифти, INI – файли, COM – об'єкти, служби. Додаток виповнюється без процедури інсталяції в традиційному її розумінні і може запускатися прямо з зовнішніх носіїв (наприклад, з флеш – карт або з мережевих папок). З точки зору IT – відділу, такий підхід має очевидні

переваги: прискорення розгортання настільних систем і можливість управління ними, зведення до мінімуму не тільки конфліктів між додатками, а й потреби у тестуванні додатків на сумісність. Дана технологія дозволяє використовувати на одному комп'ютері, а точніше в одній і тій же операційній системі кілька несумісних між собою додатків одночасно.

Віртуалізація уявлень (робочих місць). Віртуалізація уявлень має на увазі емуляцію інтерфейсу користувача. Тобто користувач бачить додаток і працює з ним на своєму терміналі, хоча насправді додаток виконується на віддаленому сервері, а користувачеві передається лише картинка віддаленої програми. Залежно від режиму роботи користувач може побачити віддалений робочий стіл і запущене на ньому додаток, або тільки саме вікно програми.

6. ОСНОВИ ХМАРНИХ ОБЧИСЛЕНЬ

Хмарні обчислення (англ. cloud computing) - технологія розподіленої обробки даних, в якій комп'ютерні ресурси і потужності надаються користувачеві як Інтернет-сервіс. Надання користувачеві послуг як Інтернет-сервіс є ключовим. Проте під Інтернет-сервісом не варто розуміти доступ до сервісу тільки через Інтернет, він може здійснюватися також і через звичайну локальну мережу з використанням веб-технологій.

Основою для створення і швидкого розвитку хмарних обчислювальних систем послужили великі інтернет сервіси, такі як Google, Amazon і ін., а так само технічний прогрес, що по суті говорить про те що поява хмарних обчислень була усього лише справою часу.

6.1. Достоїнства хмарних обчислень:

доступність - хмари доступні усім, з будь-якої точки, де є Інтернет, з будь-якого комп'ютера, де є браузер. Це дозволяє користувачам (підприємствам) економити на закупівлі високопродуктивних, дорогих комп'ютерів. Також співробітники компаній стають мобільнішими так, як можуть отримати доступ до свого робочого місця з будь-якої точки земної кулі, використовуючи ноутбук, нетбук, планшетник або смартфон. Немає необхідності в покупки ліцензійного ПО, його налаштування і оновленні, ви

просто заходите на сервіс і користуєтеся його послугами заплативши за фактичне використання.

низька вартість - основні чинники використання хмар, що понизили вартість, наступні:

- **зниження витрат на обслуговування віртуальної інфраструктури**, викликане розвитком технологій віртуалізації, за рахунок чого потрібно менший штат для обслуговування усієї IT інфраструктури підприємства;
- **оплата фактичного використання ресурсів**, користувач хмари платить за фактичне використання обчислювальних потужностей хмари, що дозволяє йому ефективно розподіляти свої грошові кошти. Це дозволяє користувачам (підприємствам) економити на купівлі ліцензій до ПЗ;
- **використання хмари на правах оренди** дозволяє користувачам понизити витрати на закупівлю дорогого устаткування, і зробити акцент на вкладення грошових коштів на наладку бізнес процесів підприємства, що у свою чергу дозволяє легко почати бізнес;
- **розвиток апаратної частини обчислювальних систем,** у зв'язку з чим зниження вартості устаткування.

гнучкість - необмеженість обчислювальних ресурсів (пам'ять, процесор, диски), за рахунок використання систем віртуалізації, процес масштабування і адміністрування "хмар" ставати досить легким завданням, оскільки "хмара" самостійно може надати вам ресурси, які вам потрібні, а ви платите тільки за фактичне їх використання.

надійність - надійність "хмар", що особливо знаходяться в спеціально обладнаних ЦОД, дуже висока так, як такі ЦОД мають резервні джерела живлення, охорону.

6.2. Недоліки хмарних обчислень:

постійне з'єднання з мережею - для діставання доступу до послуг "хмари" потрібне постійне з'єднання з мережею Інтернет.

конфіденційність - конфіденційність даних що зберігаються на публічних "хмарах" в сьогодення викликає багато суперечок, але у більшості

випадків експерти сходяться в тому, що не рекомендується зберігати найбільш цінні для компанії документи на публічній "хмарі", оскільки нині немає технології, яка б гарантувала 100% конфіденціальності даних, що зберігалися.

надійність - що стосується надійності інформації, що зберігається, то з упевненістю можна сказати що якщо ви втратили інформацію що зберігається в "хмарі", то ви її втратили назавжди.

безпека - "хмара" сама по собі є досить надійною системою, проте при проникненні на нього зловмисник дістає доступ до величезного сховища даних. Ще один мінус це використання систем віртуалізації, в яких як гіпервізор використовуються ядра стандартні ОС такі, як Linux, Windows та ін., що дозволяє використати віруси.

дорожнеча устаткування - для побудови власної хмари компанії необхідно виділити значні матеріальні ресурси, що не вигідно тільки що створеним і малим компаніям.

6.3. Види послуг надаються хмарними системами

Концепція хмарних обчислень припускає надання наступних типів послуг своїм користувачам:

- все як послуга (Everything as a Service);

При такому виді сервісу користувачеві буде надано все від програмно апаратної частини і до управлінням бізнес процесами, включаючи взаємодію між користувачами, від користувача потрібно тільки наявність доступу в мережу Інтернет.

- інфраструктура як послуга (Infrastructure as a service);

Користувачеві надається комп'ютерна інфраструктура, зазвичай віртуальні платформи (комп'ютери) пов'язані в мережу.

- платформа як послуга (Platform as a service);

Користувачеві надається комп'ютерна платформа, зі встановленою операційною системою.

- програмне забезпечення як послуга (Software as a service);

Цей вид послуги зазвичай позиціонується як "програмне забезпечення на вимогу", це програмне забезпечення розгорнуте на віддалених серверах і

користувач може діставати до нього доступ за допомогою Інтернету, причому усі питання оновлення і ліцензій на це програмне забезпечення регулюється постачальником цієї послуги. Оплата в даному випадку робитися за фактичне використання програмного забезпечення.

- апаратне забезпечення як послуга (Hardware as a Service);

В даному випадку користувачеві послуги надається устаткування, на правах оренди яке він може використати для власних цілей. Цей варіант дозволяє економити на обслуговуванні цього устаткування, хоча за своєю суттю мало чим відрізняється від виду послуги "Інфраструктура як сервіс" за винятком того що ви маєте голе устаткування на основі якого розгортаєте свою власну інфраструктуру з використанням найбільш відповідного програмного забезпечення.

- робоче місце як послуга (Workplace as a Service);

В даному випадку компанія використовує хмарні обчислення для організації робочих місць своїх співробітників, настроївши і встановивши усе необхідне програмне забезпечення, необхідне для роботи персоналу.

- дані як послуга (Data as a Service);

Основна ідея цього виду послуги полягає в тому, що користувачеві надається дисковий простір, який він може використати для зберігання великих об'ємів інформації.

- безпека як сервіс (Security as a Service).

Цей вид послуги надає можливість користувачам швидко розгортати, продукти дозволяють забезпечити безпечне використання веб-технологій, безпеку електронного листування, а також безпеку локальної системи, що дозволяє користувачам цього сервісу економити на розгортанні і підтримці своєї власної системи безпеки.

6.4. Класифікація хмарних сервісів

Нині виділяють три категорії "хмар" :

1. Публічні;
2. Приватні;
3. Гібридні.

Публічна хмара – це IT-інфраструктура використовуване одночасно безліччю компаній і сервісів. Користувачі цих хмар не мають можливості управляти і обслуговувати цю хмару, уся відповідальність з цих питань покладена на власника цієї хмари. Абонентом пропонованих сервісів може стати будь-яка компанія і індивідуальний користувач. Вони пропонують легкий і доступний за ціною спосіб розгортання веб-сайтів або бізнес-систем, з великими можливостями масштабування, які в інших рішеннях були б недоступні. Приклади: онлайн сервіси Amazon EC2 і Simple Storage Service (S3), Google Apps/Docs, Salesforce.com, Microsoft Office Web.

Приватна хмара – це безпечна IT-інфраструктура, контрольована і експлуатована в інтересах однієї-єдиної організації. Організація може управляти приватною хмарою самостійно або доручити це завдання зовнішньому підрядникові. Інфраструктура може розміщуватися або в приміщеннях замовника, або у зовнішнього оператора, або частково у замовника і частково у оператора. Ідеальний варіант приватної хмари ця хмара розгорнута на території організації, обслуговувана і контрольована її співробітниками.

Гібридна хмара - це IT-інфраструктура використовує кращі якості публічної і приватної хмари, при рішенні поставленої задачі. Часто такий тип хмар використовується коли організація має сезонні періоди активності, іншими словам, як тільки внутрішня IT-інфраструктура не справляється з поточними завданнями, частина потужностей перекидається на публічну хмару (наприклад великі об'єми статистичної інформації, які в необробленому вигляді не представляють цінності для підприємства), а також для надання доступу користувачам до ресурсів підприємства (до приватної хмари) через публічну хмару.

Нагадаємо, що під хмарними обчисленнями ми розуміємо програмно-апаратне забезпечення, доступне користувачеві через Інтернет або локальну мережу у вигляді сервісу, що дозволяє використовувати зручний інтерфейс для віддаленого доступу до виділених ресурсів (обчислювальних ресурсів, програм і даних).

На даний момент більшість хмарних інфраструктур розгорнуто на серверах датацентрів, використовуючи технології віртуалізації, що фактично

дозволяє будь-якому призначеному для користувача додатком використовувати обчислювальні потужності, абсолютно не замислюючись про технологічні аспектах. Тоді можна розуміти «хмара» як єдиний доступ до обчислень з боку користувача.

Види хмарних обчислень

З поняттям хмарних обчислень часто пов'язують сервіси що мають (Everything as a service) технології, такі як:

• «**Інфраструктура як сервіс»** (" Infrastructure as a Service " або " IaaS ");

• **«Платформа як сервіс»** (" Plaatform as a Service ", " PaaS ");

• **«Програмне забезпечення як сервіс»** (" Software as a Service " або " SaaS ").

6.5. Інфраструктура як сервіс (IaaS)

IaaS – це надання комп'ютерної інфраструктури як послуги на основі концепції хмарних обчислень.

IaaS складається з трьох основних компонентів:

• Апаратні засоби (сервери, системи зберігання даних, клієнтські системи, мережеве обладнання);

• Операційні системи та системне ПЗ (засоби віртуалізації, автоматизації, основні засоби управління ресурсами);

• Сполучне ПО (наприклад, для управління системами).

IaaS заснована на технології віртуалізації, що дозволяє користувачу обладнання ділити його на частини, які відповідають поточним потребам бізнесу, тим самим збільшуючи ефективність використання наявних обчислювальних потужностей. Користувач (компанія або розробник ПЗ) повинен буде оплачувати всього лише реально необхідні йому для роботи серверний час, дисковий простір, мережеву пропускну спроможність та інші ресурси. Крім того, IaaS надає в розпорядження клієнта весь набір функцій управління в одній інтегрованої платформі.

6.6. Платформа як сервіс (PaaS)

PaaS – це надання інтегрованої платформи для розробки, тестування, розгортання і підтримки веб – додатків як послуги.

Для розгортання веб -додатків розробнику не потрібно купувати обладнання та програмне забезпечення, немає необхідності організовувати їх підтримку. Доступ для клієнта може бути організований на умовах оренди.

Такий підхід має такі переваги:

• масштабованість ;

• відмовостійкість ;

• віртуалізація ;

• безпека.

Масштабованість PaaS передбачає автоматичне виділення і звільнення необхідних ресурсів залежно від кількості обслуговуваних додатком користувачів. PaaS як інтегрована платформа для розробки, тестування, розгортання і підтримки веб-додатків дозволить весь перелік операцій з розробки, тестування, та розгортанню веб-додатків виконувати в одному інтегрованому середовищі, виключаючи тим самим витрати на підтримку окремих середовищ для окремих етапів. Здатність створювати вихідний код і надавати його в загальний доступ всередині команди розробки значно підвищує продуктивність по створенню додатків на основі PaaS.

Найвідомішим прикладом такої платформи є AppEngine від Google, яка пропонує хостинг для веб-додатків з можливістю купувати додаткові обчислювальні ресурси (наприклад, для тестування високих навантажень). Для запуску додатків Google AppEngine на віртуальних кластерних системах була розроблена платформа AppScale, яка не має ніякого відношення до Google.

У системах веб-пошуку і контекстної реклами компанії Yahoo використовується платформа Hadoop, орієнтована на передачу великих обсягів даних між мережевими серверами. На базі Hadoop побудовані HBase (аналог бази даних Google BigTable), а також HDFS (Hadoop Distributed File System, аналог Google File System).

Ще одним яскравим представником PaaS є продукти компанії Mosso:

-Cloud Sites – веб-хостинг (Linux, Windows, Mail) для навантажувальних веб – проектів з можливістю розширювати базові безкоштовні – можливості за додаткову плату (трафік, сховище даних, обчислювальна потужність).

-Cloud Files – файловий cloud-хостинг з щомісячною погігабайтною оплатою за обсяг збережених файлів. Управління здійснюється через браузер, або за допомогою API (PHP, Python, Java, .NET, Ruby).

-Cloud Servers – погодинна оренда серверів (RAM на годину), з можливістю вибору серверної ОС. Можна змінювати характеристики сервера, але не в режимі реального часу. Незабаром розробники обіцяють зробити API для управління серверами.

Ну а в центрі всієї хмарної інфраструктури Microsoft – операційна система Windows Azure. Windows Azure створює єдине середовище, що включає хмарні аналоги серверних продуктів Microsoft (реляційна база даних SQL Azure, що є аналогом SQL Server, а також Exchange Online, SharePoint Online і Microsoft Dynamics CRM Online) і інструменти розробки (.NET Framework і Visual Studio, оснащена в версії 2010 набором Windows Azure Tools). Так, наприклад, програміст, що створює сайт в Visual Studio 2010, може не виходячи з програми розмістити свій сайт в Windows Azure.

6.7. Програмне забезпечення як сервіс (SaaS).

SaaS – модель розгортання програми, яка надає додаток кінцевому користувачеві як послугу на вимогу (on demand). Доступ до такого додатку здійснюється за допомогою мережі, а найчастіше за допомогою Інтернет-браузера.

У даному випадку, основна перевага моделі SaaS для клієнта полягає у відсутності витрат, пов'язаних з установкою, оновленням і підтримкою працездатності обладнання та програмного забезпечення, що працює на ньому. Цільова аудиторія – кінцеві споживачі.

У моделі SaaS:

•додаток пристосований для віддаленого використання;

•одним додатком можуть користуватися декілька клієнтів;

•оплата за послугу стягується або як щомісячна абонентська плата, або на основі сумарного обсягу транзакцій;

•підтримка програми входить вже до складу оплати;

•модернізація програми може проводитися обслуговуючим персоналом плавно і прозоро для клієнтів.

З точки зору розробників програмного забезпечення, модель SaaS дозволить ефективно боротися з неліцензійним використанням програмного забезпечення, завдяки тому, що клієнт не може зберігати, копіювати і встановлювати програмне забезпечення.

Розвитком логіки SaaS є концепція **WaaS (Workplace as a Service – робоче місце як послуга).** Тобто клієнт отримує в своє розпорядження повністю оснащене всім необхідним для роботи ПЗ віртуальне робоче місце.

За нещодавно опублікованими даними SoftCloud попитом користуються наступні SaaS додатки (у порядку убування популярності):

•Пошта

•Комунікації (VoIP)

•Антиспам і антивірус

•Helpdesk

•Управління проектами

•Дистанційне навчання

•CRM

•Зберігання і резервування даних

7. ПОНЯТТЯ ОБЧИСЛЮВАЛЬНОГО КЛАСТЕРА

Обчислювальний кластер - це мультикомп’ютерна архітектура, яка може використовуватися для паралельних обчислень. Це система, зазвичай складається з одного серверного вузла і одного або більше клієнтських вузлів, з'єднаних за допомогою Ethernet або деякої іншої мережі.

Даний обчислювальний кластер - це не специфічний пакет програм, нова топологія мережі або новітня модифікація ядра ОС, а технологія кластеризації комп'ютерів, що працюють під управлінням ОС Linux на різновид паралельного, віртуального суперкомп’ютера. Хоча існує багато

програмних пакетів, таких як модифікації ядра, бібліотеки PVM і MPI, конфігураційні утиліти, які роблять архітектуру кластера більш швидкою, простою у конфігуруванні і ефективною, використовуючи тільки стандартний дистрибутив Linux, без будь-якого додаткового математичного забезпечення.

Кластер складається з окремих машин (вузлів) і об'єднуючої їх мережі (комутатора). Крім ОС, необхідно встановити та налаштувати мережеві драйвери, компілятори, програмне забезпечення для підтримки паралельного програмування і розподілу обчислювального навантаження.

Вузли кластера: підходящим вибором в даний момент є системи на базі процесорів Intel Core 2 Duo або Intel Core 2 Quad. Варто встановити на кожен вузол не менше 1Gb оперативної пам'яті. Бажано 2-4Gb. Одну з машин слід виділити в якості центральної (консоль кластеру) куди можна (але не обов'язково) встановити досить великий жорсткий диск, можливо більш потужний процесор і більше пам'яті, ніж на інші (робочі) вузли. Робити консоль кластеру більш потужною машиною має сенс, якщо необхідно мати на цьому комп'ютері крім інтерфейсу командного рядка більш зручне операційне оточення, наприклад віконний менеджер (KDE, Gnome), офісні програми, програми візуалізації даних і т.п.

7.1. Будова кластера

В загальному розгортання кластеру як такого - завдання актуальне та доволі не складне. Причому, для цього підійде будь-який дистрибутив. Який саме з дистрибутивів Linux ставити в якості базової ОС - не має значення. Ubuntu, Mandriva, Alt Linux, Red Hat, SuSE. Вибір залежить тільки від уподобань користувача.

Отже, щоб розвернути кластер, використовуючи дистрибутив загального призначення, слід виконувати наступне:

1. Встановити операційну систему на комп'ютер, який буде виступати в ролі консолі кластеру. Тобто на цьому комп'ютері будуть компілюватися і запускатися паралельні програми. Іншими словами, за цим комп'ютером буде сидіти людина, запускати програми і дивитися, що вийшло.

2. Після інсталяції базової ОС на консолі кластера, якщо це не зроблено в процесі первісної установки, необхідно буде встановити необхідні компілятори (фортран, С) і всі необхідні бібліотеки, desktop environment (GNOME або KDE), текстові редактори і т. д., тобто перетворити цей комп'ютер в робочу станцію розробника.

3. Встановити з репозитарію або з вихідного пакет MPICH або OpenMPI.

4. Описати в /etc/hosts майбутні вузли вашого кластеру, в тому числі і консоль кластеру.

5. Встановити NFS і розшарити для всіх вузлів кластера якусь директорію, в якій будуть розміщуватися виконавчі модулі паралельних програм та файли даних, якими ці програми будуть користуватися в процесі своєї роботи.

6. Встановити на консолі кластеру ssh-клієнт (обов'язково) та ssh-сервер (опціонально, якщо буде надаватися доступ до консолі кластеру по мережі).

7. На всіх вузлах кластера встановити операційну систему, бібліотеки, необхідні для виконання користувальницьких паралельних програм, встановити MPICH, NFS-client, ssh-server. Вузли кластера в цілях економії ресурсів повинні навантажуватися в runlevel 3, так що ставити туди GNOME або KDE не треба. Максимум - поставити ряд бібліотек, якщо вони потрібні для користувача.

8. Описати в /etc/hosts всіх вузлів кластера майбутні вузли вашого кластеру, в тому числі і консоль кластеру.

9. На всіх вузлах кластера необхідно автоматично при завантаженні монтувати розшарений ресурс. Причому, шлях до цього ресурсу повинен бути однаковий, як на консолі кластера, так і на його вузлах. Наприклад, якщо на консолі кластеру дати доступ каталогу /home/mpiuser/data, то на вузлах кластеру цей ресурс також має бути змонтований в /home/mpiuser/data.

10. На всіх вузлах кластера забезпечити безпарольний доступу по ssh для консолі кластеру.

Оскільки від мережі прямо залежить ефективність роботи кластеру, то хотілося б зробити наступне. Необхідно, щоб функціонування мережевої файлової системи NFS не заважало обміну даними, який здійснюють між собою частини паралельної програми, що працюють у різних вузлах. Щоб це

здійснити, необхідно фізично розділити мережу на два сегменти. В одному сегменті буде працювати NFS, в іншому - відбуватиметься обмін даними між частинами програми.

7.2. Організація мережі обчислювального кластеру

Мережа - це модульна і адаптуюча комутаційна система, яку можна налаштувати відповідно до самих різних вимог. Її модульність полегшує додавання нових компонентів або переміщення існуючих, а адаптивність спрощує внесення змін і удосконалень.

Мережа кластера нічим принципово не відрізняється від мережі робочих станцій, тому в найпростішому випадку для побудови кластеру необхідні звичайні мережеві карти та хаби / комутатори, які використовувалися б при облаштуванні якогось комп'ютерного класу. Однак, у випадку кластеру є одна особливість. Мережа кластера в першу чергу призначена не для зв'язку машин, а для зв'язку обчислювальних процесів. Тому чим вищою буде пропускна здатність мережі, тим швидше будуть вважатися паралельні завдання, запущені на кластері, отже робочі характеристики мережі набувають перчергове значення.

Для побудови обчислювальних кластерів використовують найрізноманітніше мережеве обладнання. При цьому, так як характеристики стандартних мережних пристроїв помітно поступаються характеристикам спеціалізованих комунікацій в "нормальних" MPP комп'ютерах, пропускна здатність мережі, що зв'язує вузли кластеру, у багатьох випадках виявляється вирішальною для продуктивності кластера. Використовуване мережеве обладнання характеризують зазвичай двома параметрами:

Латентність - це середній час між викликом функції передачі даних і самою передачею. Час витрачається на адресацію інформації, спрацювання проміжних мережних пристроїв, інші накладні витрати, що виникають при передачі даних.

Фактично пропускна здатність і латентність не тільки характеризують кластер, але й обмежують клас задач, які можуть ефективно вирішуватися на ньому. Так, якщо завдання вимагає частої передачі даних, кластер, який використовує мережеве обладнання з великою латентністю (наприклад

GigabitEthernet), буде велику частину часу витрачати навіть не на передачу даних між процесами, а на встановлення зв'язку, в той час як вузли будуть простоювати, і ми не отримаємо значного збільшення продуктивності. Втім, якщо пересилаються великі обсяги даних, вплив періоду латентності на ефективність кластеру може знижуватися за рахунок того, що сама передача вимагатиме досить великого часу, може бути навіть у рази більшою за період латентності.

7.2.1. Мережеві карти

В якості мережевих адаптерів можна використовувати будь-які наявні в продажу карти, що підтримують роботу в стандартах 100BaseTx і GigabitEthernet. Що стосується списку переваг, то можна порекомендувати в першу чергу 3Com. Серед інших варіантів можна назвати Compex, Intel, Macronix, інші карти, підтримувані драйвером tulip, наприклад карти на чіпсетах DC21xxx. Особливо популярними при побудові кластерів являються плати на базі мікросхем Intel 21142/21143. Популярність цих карт викликана існуючим механізмом високої продуктивності, в той час як їх ціна в порівнянні з конкуруючими пропозиціями зазвичай досить невелика. Що стосується мережевих карт фірми 3Com, то вони мають деякі переваги, помітно впливають на продуктивність мережевих комунікацій. Наведемо лише кілька прикладів можливостей апаратного забезпечення карт 3Com.

Розвантаження процесора при обчисленні контрольних сум TCP/UDP/IP. Звільняє центральний процесор від інтенсивних обчислень контрольних сум, виконуючи їх в самій мережевій платі. Тим самим підвищується продуктивність системи і час життя процесора.

Звільнення ЦП при відновленні сегментованих пакетів TCP: знижує навантаження на центральний процесор, підвищуючи продуктивність системи.

Об'єднання переривань: дозволяє групувати декілька отриманих пакетів. Оптимізує обчислювальну ефективність хост-комп'ютера, скорочуючи число переривань і максимально звільняючи процесорні ресурси для роботи додатків.

Режим Bus mastering DMA: забезпечує більш ефективний обмін даними для зниження завантаження центрального процесора.

7.2.2. Комутатори

Другим важливим елементом мережі кластеру є пристрої комутації мережевих каналів. При виборі комутуючих пристроїв так само слід враховувати можливість використання channel bonding. Залежно від того, чи буде використовуватися технологія зв'язування каналів при побудові кластеру, можна зупинити свій вибір на різному мережевому обладнанні.

Комутатори та інші елементи мережевої структури використовуються для забезпечення комунікацій між процесорами, для підтримки паралельного програмування і різних функцій управління. Для паралельного програмування організації взаємодії між процесами (Inter Process Communication, IPC) широко використовується комутатор Myrinet-2000 - дуже швидкий, добре масштабований широкосмуговий пристрій.

7.3. Паралельна віртуальна машина (PVM)

Основою обчислювального середовища кластеру є паралельна віртуальна машина PVM. PVM - це пакет програм, який дозволяє використовувати пов'язаний в локальну мережу набір різнорідних комп'ютерів, що працюють під операційною системою Unix, як один великий паралельний комп'ютер. Таким чином, проблема великих обчислень може бути досить ефективно вирішена за рахунок використання сукупної потужності та пам'яті великого числа комп'ютерів. Пакет програм PVM легко переноситься на будь-яку платформу. Вихідні тексти, вільно розповсюджувані netlib, були скомпільовані на комп'ютерах починаючи від laptop і до CRAY.

Паралельну віртуальну машину можна визначити як частину коштів реального обчислювального комплексу (процесори, пам'ять, периферійні пристрої і т.д.), призначену для виконання безлічі завдань, що беруть участь в отриманні загального результату обчислень. У загальному випадку число завдань може перевершувати число процесорів, включених в PVM. Крім

того, до складу PVM можна включати досить різнорідні обчислювальні машини, несумісні з систем команд і форматів даних. Інакше кажучи, паралельною віртуальною машиною може стати як окремо взятий ПК, так і локальна мережа, що включає в себе суперкомп'ютери з паралельною архітектурою, універсальні ЕОМ, графічні робочі станції і все ті ж малопотужні ПК. Важливо лише, щоб процеси, які включаються до PVM обчислювальних засобах були інформацією у використовуваному програмному забезпеченні PVM. Завдяки цьому програмному забезпеченні користувач може вважати, що він спілкується з однією обчислювальною машиною, в якій можливе паралельне виконання безлічі завдань. PVM дозволяє користувачам використовувати існуючі апаратні засоби, для вирішення набагато більш складних завдань при мінімальній додаткової вартості. Сотні дослідних груп в усьому світі використовують PVM, щоб вирішити важливі наукові, технічні, і медичні проблеми, а так само використовують PVM як освітній інструмент, для викладання паралельного програмування. В даний час, PVM став де факто стандартом для розподілених обчислень.

Головна мета використання PVM – це підвищення швидкості обчислень за рахунок їх паралельного виконання. Функціонування PVM засноване на механізмах обміну інформацією між завданнями, виконуваними в її середовищі. У цьому відношенні найбільш зручно реалізовувати PVM в рамках багатопроцесорних обчислювальних комплексах, виділивши віртуальній машині кілька процесорів і загальну або індивідуальну (залежно від умов) оперативну пам’ять. Використання PVM доспускається як на багатопроцесорних комп'ютерах (SMP) так і на обчислювальних комплексах, побудованих за допомогою кластерної технології. При використанні PVM, як правило, значно спрощуються проблеми швидкого інформаційного обміну між завданнями, а також проблеми узгодження форматів представлення даних між завданнями, виконуваними на різних процесорах.

Ефективне програмування для PVM починається з того, що алгоритм обчислень слід адаптувати до складу PVM і до її характеристик. Це дуже творче завдання, яке в багатьох випадках повинне вирішуватися програмістом. Крім завдання розпаралелювання обчислень з необхідністю

виникає і завдання управління обчислювальним процесом, координації дій-завдань учасників цього процесу. Іноді для управління доводиться створювати спеціальну задачу, яка сама не беручи участь в обчисленнях, забезпечує узгоджену роботу решти завдань-обчислювачів.

Раніше передчасно згадувалося, що при паралельних обчисленнях необхідно програмувати спеціальні дії з координації роботи завдань, такі як процеси запуску задач на процесорах кластеру, управління обміном даних між завданнями та ін. Також слід чітко визначити "область діяльності" для кожного завдання.

Найбільш простий і популярний спосіб організації паралельного процесу виглядає наступним чином. Спочатку запускається одне завдання (master), яке в колективі завдань буде відображати функції координатора робіт. Це завдання виробляє деякі підготовчі дії, наприклад ініціалізація початкових умов, після чого запускає інші завдання (slaves), яким може відповідати або той же виконуваний файл, або різні виконувані файли. Такий варіант організації паралельних обчислень переважно використовується при ускладненні логіки керування обчислювальним процесом, а також коли алгоритми, реалізовані в різних завданнях, істотно розрізняються або є великий обсяг операцій (наприклад, введення - виведення), які обслуговують обчислювальний процес в цілому.

ЛІТЕРАТУРА

1. Петренко А.И., Застосування GRID технологій в науці та освіті: роздатковий матеріал до вивч. курсу для студ. спец. «Інформаційні технології проектування». − К.: НТУУ «КПІ», 2008. − 144 С.
   Режим доступу: http://moodle.ntu-kpi.kiev.ua (дата звернення 30.05.2016)
2. Петренко А.И., Вступ до GRID технологій в науці та освіті: навчальний посібник. - К.: НТУУ «КПІ», 2008. − 120 с.
   Режим доступу: http://moodle.ntu-kpi.kiev.ua (дата звернення 30.05.2016)
3. Пономаренко В.С., Листровой С.В., Минухин С.В., Знахур С.В., Методы и модели планирования ресурсрв в GRID системах. – Х.:ВД. «ІНЖЕК», 2008. – 408 с.
4. Introduction to GRID Computing, December 2005. – IBM Redbook, − 241 с. [Електронний ресурс]
   Режим доступу: www.ibm.com/redbooks (дата звернення 30.05.2016)
5. GRID Computing in Research and Education, April 2005. – IBM Redbook, − 145 с. [Електронний ресурс]
   Режим доступу: www.ibm.com/redbooks (дата звернення 30.05.2016)
6. GRID Services Programming and Application Enablement, May 2004. – IBM Redbook, − 273 с. [Електронний ресурс]
   Режим доступу: www.ibm.com/redbooks (дата звернення 30.05.2016)
7 . Паклин Н.Б., Орешков В.И., Бизнес-аналитика: от данных к знаниям (+ CD), Издательский дом Питер, 1-е издание, 2009. − 624 с.
8. А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод. Методы и модели анализа данных: OLAP и Data Mining (+ CD-ROM). Издательство: БХВ-Петербург, 2004. − 336 с.
9. А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод, Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP (+ CD-ROM). - Издательство: БХВ-Петербург, 2007. − 384 с.
10. NorduGRID project. [Електронний ресурс]
    Режим доступу: http://www.norduGRID.org (дата звернення 30.05.2016)
11. The NorduGRID GRID Manager And GRIDFTP Server: Description And Administrator’s Manual. [Електронний ресурс]

Режим доступу: http://www.norduGRID.org/papers.html (дата звернення 30.05.2016)

12. EGEE User's Guide, WMS SERVICE [Електронний ресурс]
Режим доступу: https://edms.cern.ch/document/572489/1 (дата звернення 29.05.2016)

13. JDL Attributes Specification, EGEE-JRA1-TEC-555796-JDL-Attributes-v0-6. [Електронний ресурс]
Режим доступу: https://edms.cern.ch/file/555796/1/. (дата звернення 29.05.2016)

14. The Resource Broker Info file, DataGRID-01-TEN-0135-0_0. [Електронний ресурс]
Режим доступу:
http://www.infn.it/workload-GRID/docs/DataGRID-01-TEN-0135-0_0.doc
(дата звернення 29.05.2016)

15. Web Services [Електронний ресурс]
Режим доступу: http://www.w3.org/2002/ws/ (дата звернення 29.05.2016)

16. GRID Computing Making the Global Infrastructure a Reality, edited by Fran Berman, Geoffrey Fox, Tony Hey. – (Wiley series in communications networking & distributed systems), 2003, − 1007 c.

17. Portal Application Development Using WebSphere Portlet Factory, IBM Redbook, January 2008, − 697 c. [Електронний ресурс]
Режим доступу: http://www. ibm.com/redbooks (дата звернення 29.05.2016)

18. Openldap. [Електронний ресурс]
Режим доступу: http://www.openldap.org (дата звернення 29.05.2016)

19. Петренко А.І., Булах Б.В.,Хондар В.Д. Семантичні грід- технології для науки і освіти:додатковий матеріал. -// К.: НТУУ «КПІ», 2010.- 178 с.
Режим доступу: http://moodle.ntu-kpi.kiev.ua (дата звернення 29.05.2016)