РОЗПОДІЛЕНІ СИСТЕМИ ОБРОБКИ ІНФОРМАЦІЇ

Жураковський Б.Ю.
Тимошин Ю.А.
2021-2022р.

Базові складові

1. Технології розподілених обчислень.
2. Мережеві БД
3. Розподілене апаратне середовище обробки даних
4. Технології дублювання зберігання даних
5. Розподілені сервіси

Фактори розподілу обробки

Джерела інформації та даних

РОЗПОДІЛЕНІ СИСТЕМИ ОБРОБКИ ІНФОРМАЦІЇ

- Об’єкти обробки
  - Дані: Числові ряди, розклад функцій, обчислення матриць
  - Інформація:
    - Електронні документи;Текстові дані
    - Повідомлення систем обробки чи їх компонентів
    - Фото - та відео - ряди чи файли
    - Оцифрований звуковий ряд (файл)
    - Компоненти бізнес-процесів
    - SQL – запити
    - Дані для Web-додатків
  - Програмний код
  - Знання

Об’єкти обробки

- *Інформація* («пояснення») - будь-які відомості про будь-яку подію, сутність, процес тощо, що є об'єктом деяких операцій: сприйняття, передачі, перетворення, зберігання та використання, для яких існує змістовна інтерпретація.

Об’єкти обробки

- *Дані* - відносяться до способу подання, зберігання і до елементарних операцій обробки інформації. Перш за все, дані - це носій інформації.

Існують три аспекти роботи з даними:

- визначення даних;
- маніпулювання (обробка) даних;
- керування даними (адміністрування даних).

Об’єкти обробки

- *Знання* - це така інформація, до якої застосовуються алгоритми логічного висновку, що дозволяють отримати нову інформацію.
- Знання повідомляють, що інформація має прагматичний аспект - тобто існує деяка мета, яка відома системі. Система, що побудована на знаннях, має момент цілеспрямування

Системи обробки

- *Інформаційна система - система, яка організовує накопичення і маніпулювання інформацією щодо проблемної сфери (без зворотнього зв’язку)*
- *Інформаційно-керуюча система – система із зворотнім зв’язком щодо параметрів керування*

Системи обробки

- *Інформаційна система* визначається як набір взаємозалежних компонентів, що збирають, обробляють, зберігають і розподіляють інформацію, щоб підтримувати процес прийняття управлінських рішень і управління організацією в цілому.

Варіанти обробки

- Розпаралелення обчислень даних – окремий випадок розподіленої обробки інформації
- Розпаралелення обробки інформації та знань
- Розпаралелення сервісів
- Розпаралелення застосунків
- Розпаралелення апаратних ресурсів

Варіант обробки №1

А)Причина виникнення задач розподілу обробки:

- вирішення задачі розробки паралельної програми знаходження значення f(x) заданої функції f у заданій точці x_0 із заданою точністю ε,
- інтегрування f(x),
- обчислення матриць;

Варіант обробки №1

Б)Типи даних чи функції

- експонента
- натуральний логарифм
- геометричний ряд
- тригонометричні функції
- стовпи/строки/блоки матриць

Варіант обробки №1

В) Приклади реалізації

- розклад математичної функцій в ряд
- програми чисельного інтегрування функції f(x)
- обчислювання матриць

Варіант обробки №2

А)Причина виникнення задачі розподілу обробки:

- Обробка допускає розбиття обчислень по різних ядрах і процесорах (в межах одного сервера чи робочої станції)

Варіант обробки № 2

Б)Типи даних чи функції:

- числові дані
- Повідомлення систем обробки
- SQL- запити

В) Приклади реалізації:

- віртуальні сервери
- SQL- сервери (СУБД, сервери транзакцій)

Варіант обробки № 3

А)Причина виникнення задачі розподілу обробки:

- Декілька вузлів корпоративної мережі повинні оброблювати одну і ту інформацію

Варіант обробки № 3

Б)Типи даних чи функції:

- документи
- текстові файли
- повідомлення систем обробки

В) Приклади реалізації:

- Віддалені філії організації (бухгалтерія, канцелярія, відділ кадрів, менеджери)

Варіант обробки № 4

А)Причина виникнення задачі розподілу обробки:

- Обчислення можна розділити по різних комп’ютерах чи вузлах однієї чи декількох мереж

Варіант обробки №4

Б)Типи даних чи функції:

- числові дані
- повідомлення
- SQL- запити

В) Приклади реалізації:

- GRID- мережі
- суперкомп’ютери

Варіант обробки № 5

А)Причина виникнення задачі розподілу обробки:

- Алгоритм обробки допускає чи вимагає паралельну обробку різних функціональних частин розподіленого додатку чи сервісу

Варіант обробки № 5

Б)Типи даних чи функції:

- числові дані
- бізнес-процеси
- SQL- запити

В) Приклади реалізації:

- сервери додатків в клієнт-серверній архітектурі
- сервіси в SOA і в Хмарі

Варіант обробки № 6

А)Причина виникнення задачі розподілу обробки:

- Необхідно забезпечити високу надійність зберігання даних за рахунок їх дублювання та надлишковості

Варіант обробки № 6

Б)Типи даних чи функції:

- числові дані
- документи
- фото-, відео-, звуковий ряд

В) Приклади реалізації:

- RAID – технології (системи зберігання DAS/NAS/SAN)
- дублювання ресурсів сервера

Варіант обробки № 7

А)Причина виникнення задачі розподілу обробки:

- Треба забезпечити високу надійність обробки обчислювальних процесів за рахунок архітектурних рішень взаємодії серверів

Варіант обробки № 7

Б)Типи даних чи функції:

- числові дані
- документи
- бізнес-процеси

В) Приклади реалізації:

- Комп’ютерні кластери
- багатовузлові катастрофостійки системи обробки

Варіант обробки № 8

А)Причина виникнення задачі розподілу обробки:

- Необхідно забезпечити балансування навантаження для групи серверів у різних вузлах корпоративній мережі чи для віртуальних серверів одного вузла

Варіант обробки № 8

Б)Типи даних чи функції:

- числові дані
- повідомлення
- SQL- запити

В) Приклади реалізації:

- сервери балансування навантажень
- сервери транзакцій
- розподілені ОС

Розпаралелення процесів обробки додатків

Включає використання:

1) технології “Клієнт-сервер” для створення і обробки розподілених додатків
2) обробку частин додатку у різних вузлах прикладної мережі з використанням технології “серелізація - десерелізація”
3) обробку застосунків на базі технологій з брокерами
4) технологія розподілених БД

Архітектура “Клієнт-сервер”

- Трьохрівнева архітектура

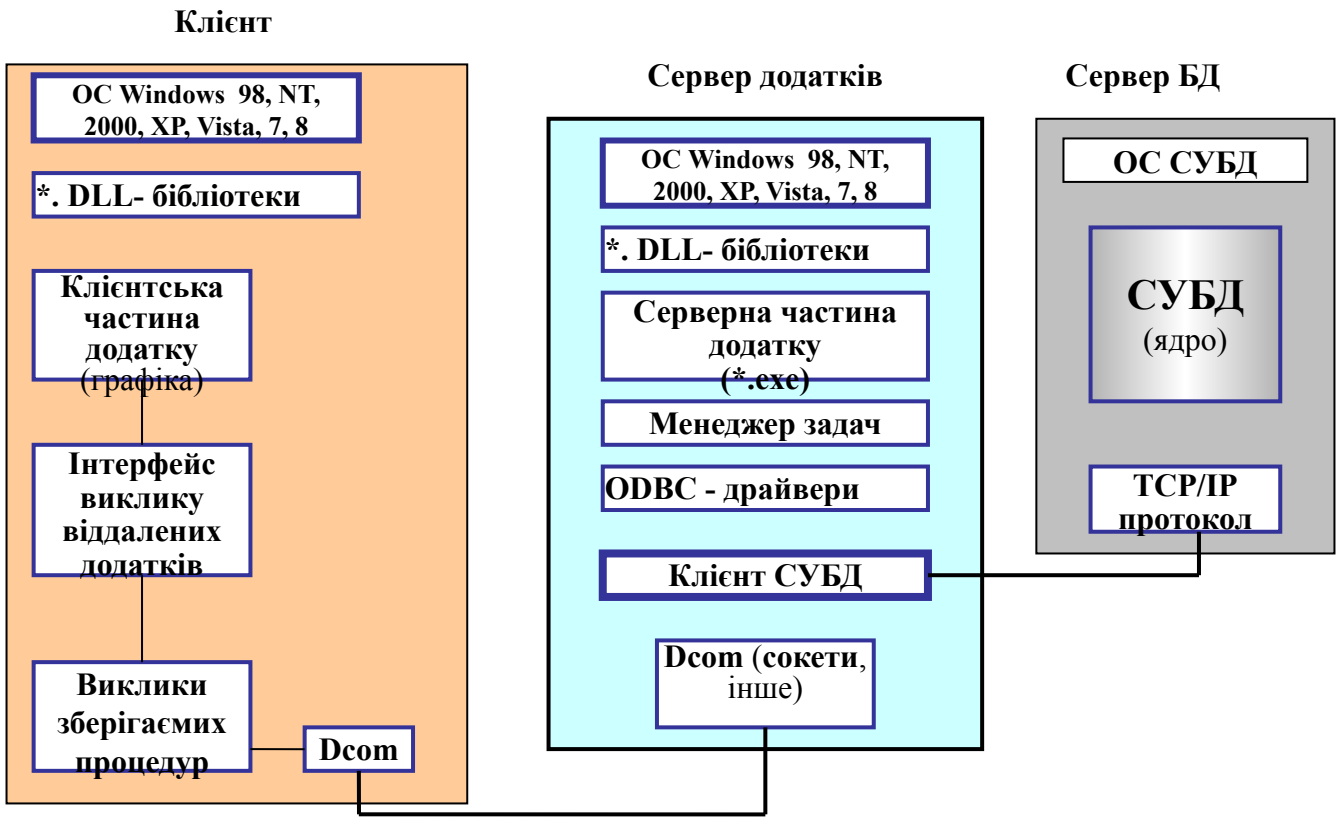


Рисунок 1.2 Трьохрівнева архітектура додатку в технології «клієнт-сервер»

3-х рівнева архітектура “Клієнт-сервер”

- Достоїнства

a) Кількість PC в межах 200-300
b) Використання архітектури “тонкого” клієнта
c) Можливість створення розподілених додатків
d) Використання обчислювальних потужностей сервера додатків
e) Можливість “балансування” навантаження між серверами і PC
f) Можливість використання різних платформ для сервера додатків і сервера СУБД
g) Відсутність ліцензій для PC
h) Наявність “серверної частини” додатків для гнучкості їх архітектури
i) Можливість використання СУБД на різних платформах одночасно
j) Зменшення завантаження комп’ютерної мережі за рахунок передачі між PC та серверами тільки даних (без графічних компонентів відповідних пакетів)
k) Відсутність необхідності перенавчання користувачів при оновленні додатків

3-х рівнева архітектура “Клієнт-сервер”

- Недоліки

a) Складність архітектури та супроводження
b) Складність супроводження додатків – обов’язкова синхронізація *.dll – бібліотек для РС та серверів додатків
c) Оновлення актуальними версіями графічних пакетів на всіх РС
d) Необхідність використання балансувального сервера транзакцій для декількох серверів додатків
e) Необхідність використання механізму реплікацій для багатовузлової архітектури обробки даних із складним його настроюванням
f) Необхідність використання різних “клієнтів” від різних вендорів для багатоплатформеної архітектури
g) Необхідність використання однотипних ОС для РС та серверів додатків
h) Обмеженість числа одночасно оброблюваних запитів на сервері додатків відповідно купленої серверної ліцензії

Технології розподілених обчислень

- RPC (Remote Procedure Call),
- ORB (Object Request Broker),
- MOM (Message-oriented Middleware),
- DCE (Distributed Computing Environment),
- монітори транзакцій,
- ODBC (Object DB Connectivity)

Технології розподілених обчислень

- RPC - процедурна синхронна технологія з блокуванням, запропонована фірмою Sun Microsystems
- пересилки здійснюються на основі транспортних протоколів TCP або UDP, дані представляються в єдиному форматі обміну XDR.
- Використовується мова IDL (Interface Definition Language), що дає користувачу можливість оперувати різними об'єктами безвідносно до їх розташування в мережі.

Технології розподілених обчислень

- ODBC

– Технологія віддаленого доступу до БД з додатку через джерело даних, яке створене на конкретному PC з використанням SQL- запитів

– Знайшла розвиток у технологіях ADO та DAO.

Технології розподілених обчислень

- **МОМ - об'єктна технологія**, що базується на повідомленнях. Реалізує, як правило, асинхронний зв'язок з серверами, але для деяких видів додатків, наприклад, фінансові дані, можуть вимагати синхронних зв’язків. Включає команди "послати" і "отримати", які здійснюють обмін повідомленнями. Використовується 3 сервери – сервер генерації повідомлень, сервер маршрутизації та сервер обробки повідомлень.

Технології розподілених обчислень

ORB – технологія

Є компонентом технології CORBA

Основні служби:

- **служба іменування**, привласнює об'єктам унікальні імена, в результаті користувач може шукати об'єкт в мережі;
- **служба обробки транзакцій**, здійснює управління транзакціями з додатків або з операційних систем;
- **служба подій**, забезпечує асинхронне поширення і обробку повідомлень про події;
- **служба забезпечення безпеки** - підтримки цілісності даних.

Технології розподілених обчислень

- ***Розподілене обчислювальне середовище*** (DCE) — це інтегрований набір служб і інструментів, які використовуються для створення та запуску розподілених програм. Це набір інтегрованих програмних компонентів/фреймворків, які можна встановити як узгоджене середовище поверх існуючої операційної системи та які можуть служити платформою для створення та запуску розподілених програм.

Розподілене обчислювальне середовище

- Використовуючи програми DCE, користувачі можуть використовувати програми та дані на віддалених серверах. Прикладним програмістам або клієнтам не обов’язково знати, де будуть працювати їхні програми або де будуть розташовані дані, до яких вони хочуть отримати доступ.

- DCE був розроблений Open Software Foundation (OSF) з використанням програмних технологій, наданих деякими компаніями-членами, які тепер широко відомі як The Open Group.

Розподілене обчислювальне середовище

I. Структура DCE включає:

- **Віддалений виклик процедур (RPC) :** це виклик, який здійснюється, коли комп'ютерна програма хоче виконати підпрограму на іншому комп'ютері в спільній мережі.

- **Розподілену файлову систему (DFS) :** забезпечує прозорий спосіб доступу до файлу в системі так само, ніби він знаходиться в тому самому місці де знаходиться сама програма.

Розподілене обчислювальне середовище

II. Сервіси DCE включають:

- ***Службу каталогів:*** **використовується для відстеження розташування віртуальних ресурсів у розподіленій системі. Ці ресурси включають файли, принтери, сервери, сканер та інші робочі станції. Ця послуга спонукає користувача запитувати ресурси (через процес) і надавати їм зручність доступу до ресурсу. Процеси не знають про фактичне розташування ресурсів.**

- ***Службу безпеки:*** **дозволяє процесу перевірити автентичність користувача. Тільки уповноважена особа може мати доступ до захищених і безпечних ресурсів. Це дозволяє лише авторизованому комп'ютеру в мережі розподілених систем мати доступ до захищених ресурсів.**

Розподілене обчислювальне середовище

Сервіси DCE включають (продовження)

- ***Служба розподіленого часу:*** **зв'язок між процесами між різними компонентами системи потребує синхронізації, щоб зв'язок відбувався лише у визначеному порядку. Ця служба відповідає за підтримку глобального годинника і, отже, синхронізацію локальних годинників із поняттям глобального часу.**
- ***Служба потоків (Thread Service):*** **забезпечує реалізацію легких процесів (потоків). Допомагає синхронізувати кілька потоків у спільному адресному просторі.**

Розподілене обчислювальне середовище

III. Архітектура DCE

- DCE підтримує структурування розподілених обчислювальних систем у так звані **осередки (cells),** які складаються з 3 типів машин: користувача, адміністратора та сервера. Це робиться для того, щоб зберегти контрольований розмір домену адміністрування. Осередок — це в основному набір вузлів, якими разом керує один адмін.
- Межі осередка представляють брандмауери безпеки; доступ до ресурсів у чужому осередку потребує спеціальних процедур автентифікації та авторизації, які відрізняються від безпечних взаємодій усередині “свого” осередку.
- Найвищі привілеї в осередку призначаються для ролі, яка називається “адміністратор осередку DCE”, який дистанційно контролює всі системні служби в мережі. Він має привілеї щодо всіх ресурсів в осередку розподіленого обчислювального середовища.

Розподілене обчислювальне середовище

Основні компоненти осередку

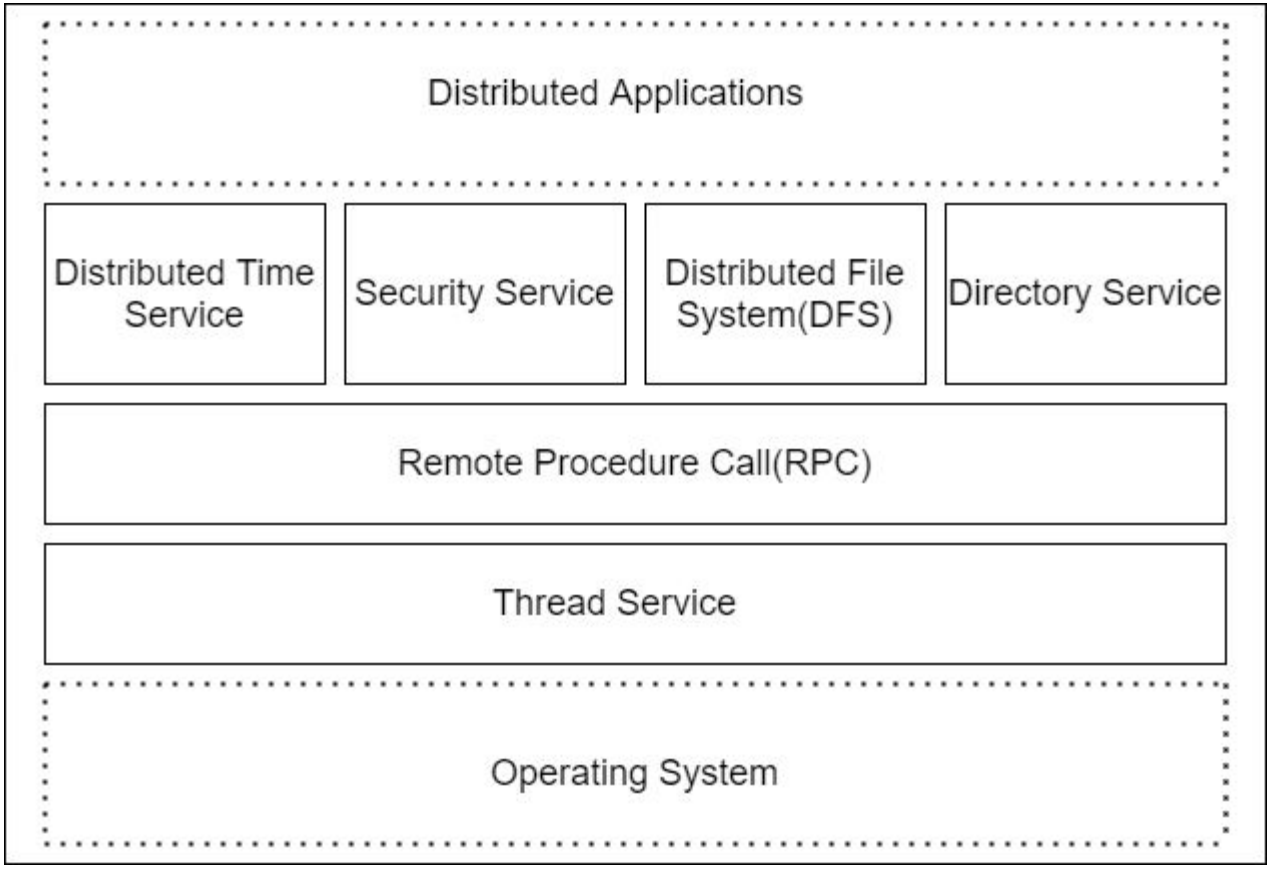
- **Сервер безпеки**, який відповідає за автентичність користувача
- **Cell Directory Server (CDS) –** сховище ресурсів
- **Розподілений сервер часу** – забезпечує годинник для синхронізації всього осередку компютерів

Розподілене обчислювальне середовище

Переваги DCE:

- Безпека
- Нижчі витрати на технічне обслуговування
- Масштабованість і доступність
- Зменшені ризики

Розподілене обчислювальне середовище



Технології розподілених обчислень

- Монітори транзакцій
- Реалізуються на серверах транзакцій
- Використовуються для балансування навантажень в технології клієнт-сервер:

a) Між клієнтами та сервером додатків
b) Між серверами додатків та серверами БД.

Системи з розподіленими додатками

- Основою більшості розподілених систем є явний обмін повідомленнями між процесами.
- Однак процедури *send* та *receive* не приховують взаємодії, що необхідно для забезпечення прозорості доступу.
- Було запропоновано дозволити програмам викликати процедури, які перебувають на інших машинах.

Системи з розподіленими додатками. RPC

I. Віддалений виклик процедур

Remote Procedure Call (RPC) – **основні властивості:**

a) Було запропоновано дозволити програмам викликати процедури, які перебувають на інших машинах.

b) Інформація може бути передана від процесу, який викликає, до процедури, яка викликається, через параметри і повернута процесу у вигляді результату виконання процедури.

Системи з розподіленими додатками. RPC

Основні властивості (продовження)

c) Ідея **виклику відалених процедур**(Remote Procedure Call – RPC) складається в розширенні добре відомого й зрозумілого механізму передачі керування та даних усередині програми, що виконується на одній машині, на передачу керування й даних через мережу.

d) Засоби віддаленого виклику процедур призначені для полегшення організації розподілених обчислень і створення розподілених клієнт-серверних інформаційних систем.

e) Найбільша ефективність використання RPC досягається в тих прикладних програмах, у яких існує інтерактивний зв'язок між віддаленими компонентами *з невеликим часом відповідей і відносно малою кількістю переданих даних*. Такі прикладні програми називаються RPC-орієнтованими.

Системи з розподіленими додатками. RPC

Основні властивості (продовження)

f) Характерними рисами виклику локальних процедур є:

– асиметричність, тобто одна із взаємодіючих сторін є ініціатором;

– синхронність, тобто виконання процедури, яка викликає віддалену процедуру, припиняється з моменту видачі запиту й відновлюється тільки після повернення з результатів запиту з викликаної процедури.

– процедура, яка викликає і процедура, яку викликають, виконуються на різних машинах, а вони мають різні адресні простори, і це створює проблеми при передачі параметрів і результатів, особливо якщо машини перебувають під керуванням різних операційних систем або мають різну архітектуру (наприклад, використається прямий або зворотній порядок байтів).

– використання операцій серілізації та десерілізації

(Під серіалізацією розуміють процес перекладу будь-якої структури даних у послідовність бітів. Зворотньою до операції серіалізації є операція десеріалізації– відновлення початкового стану структури даних з бітової послідовності. Серіалізація використається для передачі об’єктів по мережі й для збереження їх у файли).

Системи з розподіленими додатками. RPC

Основні властивості (продовження)

– віддалений виклик процедур обов'язково використовує транспортний рівень мережної архітектури (наприклад TCP), однак це залишається прихованим від розробника програмного забезпечення;

– виконання програми, яка викликає віддалену процедуру, і викликуваної локальної процедури в одній машині реалізується в рамках єдиного процесу.

g) Проблеми реалізації:

- проблеми, які пов'язані з неоднорідністю мов програмування й операційних середовищ: структури даних і структури виклику процедур, підтримувані в будь-який одній мові програмування, не підтримуються так само у всіх інших мовах.
- існує проблема сумісності, дотепер не вирішена ні за допомогою введення одного загальноприйнятого стандарту, ні за допомогою реалізації декількох конкуруючих стандартів на всіх архітектурах і у всіх мовах.

Системи з розподіленими додатками. CORBA

Брокери запитів. Їх функції і особливості.

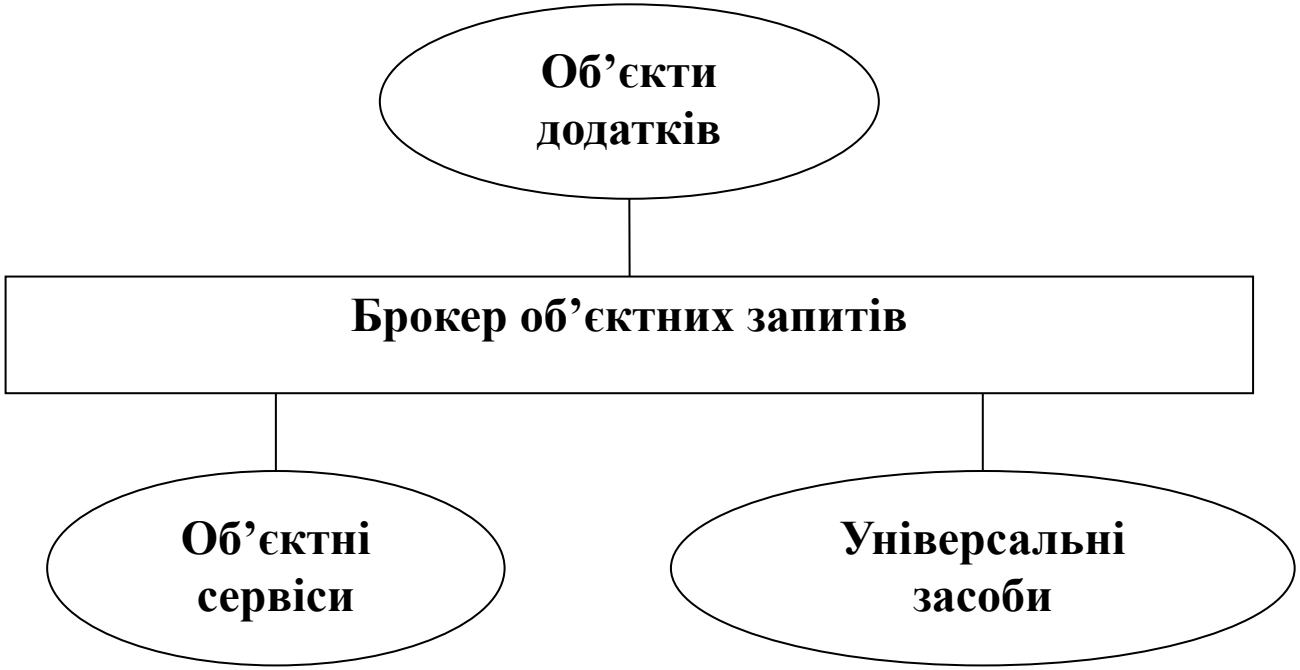
a) Необхідна специфікація доступу до розподілених додатків в гетерогенному середовищі була розроблена OMG і отримала назву Object Management Architecture (OMA).

b) OMA складається з чотирьох основних компонентів, що представляють собою специфікації різних рівнів підтримки додатків:

- архітектури брокера запитів об'єктів (CORBA - Common Object Request Broker Architecture) - встановлює базові механізми взаємодії об'єктів в гетерогенній зоні;
- сервісів об'єктів (Object services) - є основними системними службами, що використовуються розроблювачами для створення додатків;
- універсальних засобів (Common Facilities) – що орієнтовані на підтримку користувацьких додатків, таких, як електронна пошта, засоби друку і т.п.;
- об'єктів додатків (Application Objects) – які призначені для вирішення конкретних прикладних завдань.

Системи з розподіленими додатками. CORBA

- *Архітектура управління об'єктами (OMA)*



Системи з розподіленими додатками. CORBA

Наявність великої кількості мережевих протоколів в різних операційних системах вимагало розробки специфікації мережевих з'єднань.

COBRA визначає механізм, що забезпечує взаємодію додатків в розподіленому середовищі.

Головними компонентами стандарту CORBA є:

- об'єктний брокер запитів (Object Request Broker);
- мова визначення інтерфейсів (Interface Definition Language);
- об'єктний адаптер (Object Adapter);
- репозитарій інтерфейсів (Interface Repository).

Системи з розподіленими додатками. CORBA

- Місце CORBA в семирівневій моделі OSI

Семирівнева модель Open System Interconnection (OSI) забезпечує опис сервісів, які кожен рівень в CORBA повинен представляти для реалізації з'єднання.

1. Нижчий рівень - фізичний - визначає доступ до фізичної лінії.
2. Рівень даних забезпечує достовірну передачу даних з фізичної лінії.
3. Мережевий рівень має справу з установкою з'єднання і маршрутизацією.
4. Транспортний рівень відповідає за достовірну передачу до точки призначення.
5. Рівень сесій забезпечує управління з'єднанням.
6. Рівень уявлень описує синтаксис даних і забезпечує прозорість для додатків.
7. Останній рівень - рівень додатків - забезпечує відповідні мережеві функції для кінцевого користувача.

Системи з розподіленими додатками. CORBA

- Концептуально CORBA відноситься до рівнів додатків і уявлень. Вона забезпечує можливість побудови розподілених систем і додатків на найвищому рівні абстракції в рамках стандарту OSI. З її допомогою можливо ізолювати клієнтські програми від низькорівневих, гетерогенних характеристик інформаційних систем.
- Високий рівень абстракції CORBA в сьомирівневій моделі OSI дозволяє програмісту не працювати з низькорівневими протоколами.
- Програмісту не потрібна інформація про реальне місце розташування сервера та способи його активізації.
- Розробка клієнтської програми не залежить від серверної операційної системи і апаратної платформи.
- Мова опису інтерфейсів OMG IDL дозволяє визначити інтерфейс, незалежний від мови програмування, що використовується для реалізації.

Системи з розподіленими додатками. CORBA

Об'єктна модель CORBA

- визначає взаємодію між клієнтами і серверами. Клієнти - це додатки, які запитують сервіси. Сервери - це додатки, що представляють сервіси.
- Об'єкти-сервери містять набір сервісів, які розділяються між багатьма клієнтами. Операція вказує запитуваний сервіс об'єкта-сервера.
- Інтерфейси об'єктів є опис безлічі операцій, які можуть бути викликані клієнтами даного об'єкта.
- Реалізації об'єктів - це додатки, що реально виконують сервіси, які запитуються клієнтами.

Системи з розподіленими додатками. CORBA

Об'єктний брокер запитів (ORB)

- *Специфікація CORBA розроблена для забезпечення можливості інтеграції абсолютно різних об'єктних систем.*
- *Його завданням є представлення механізму виконання запиту, зробленого клієнтом:*

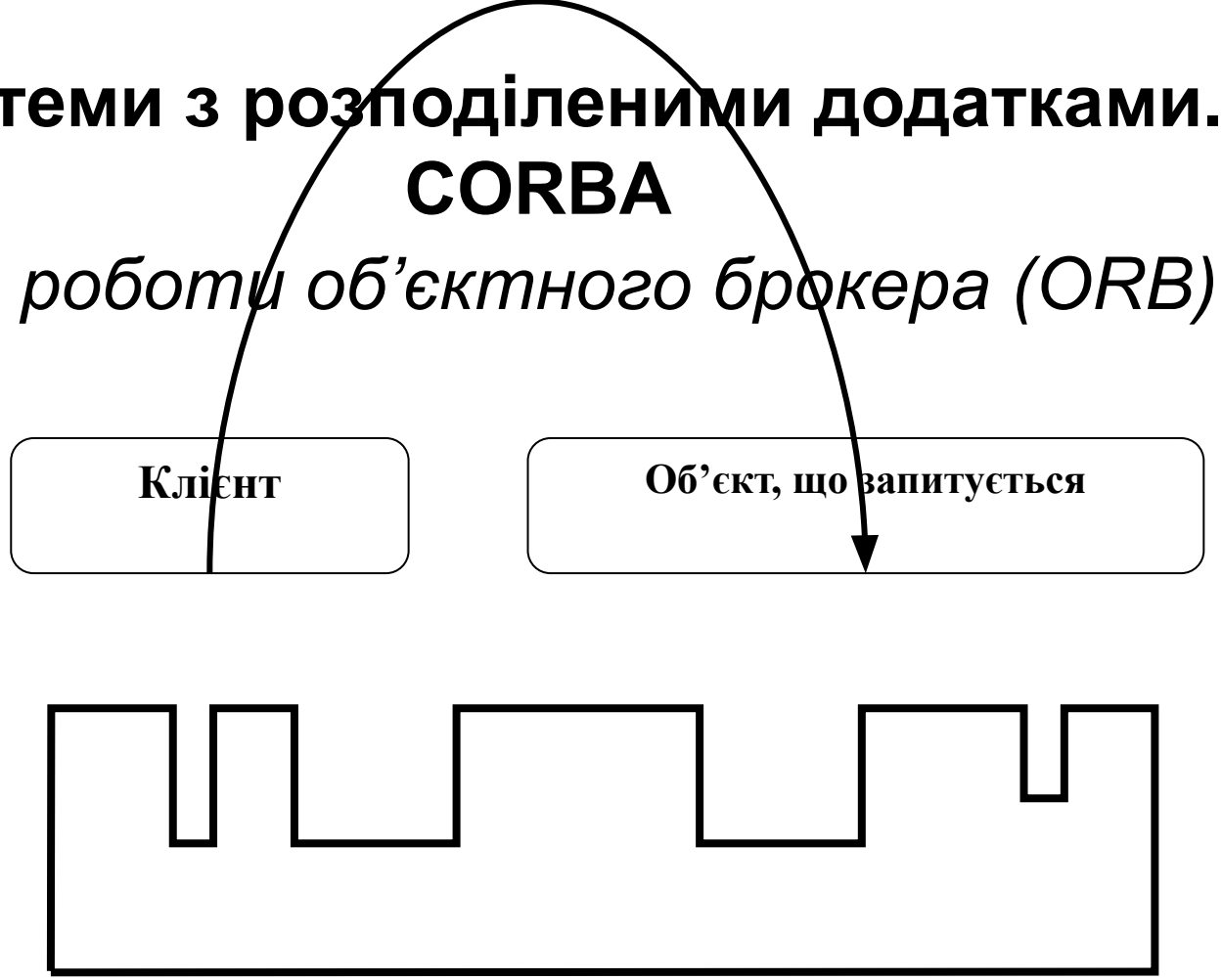
a) *пошук об'єкта, до якого відноситься даний запит, передача необхідних даних, підготовка об'єкта до обробки.*

b) *інтерфейс, за допомогою якого клієнт може запитувати виконання необхідних операцій, не залежить від місцезнаходження об'єкта та мови програмування, за допомогою якого він реалізований.*

c) *клієнт може запитувати виконання операцій за допомогою ORB декількома способами.*

Системи з розподіленими додатками. CORBA

- *Схема роботи об’єктного брокера (ORB)*



Системи з розподіленими додатками. CORBA

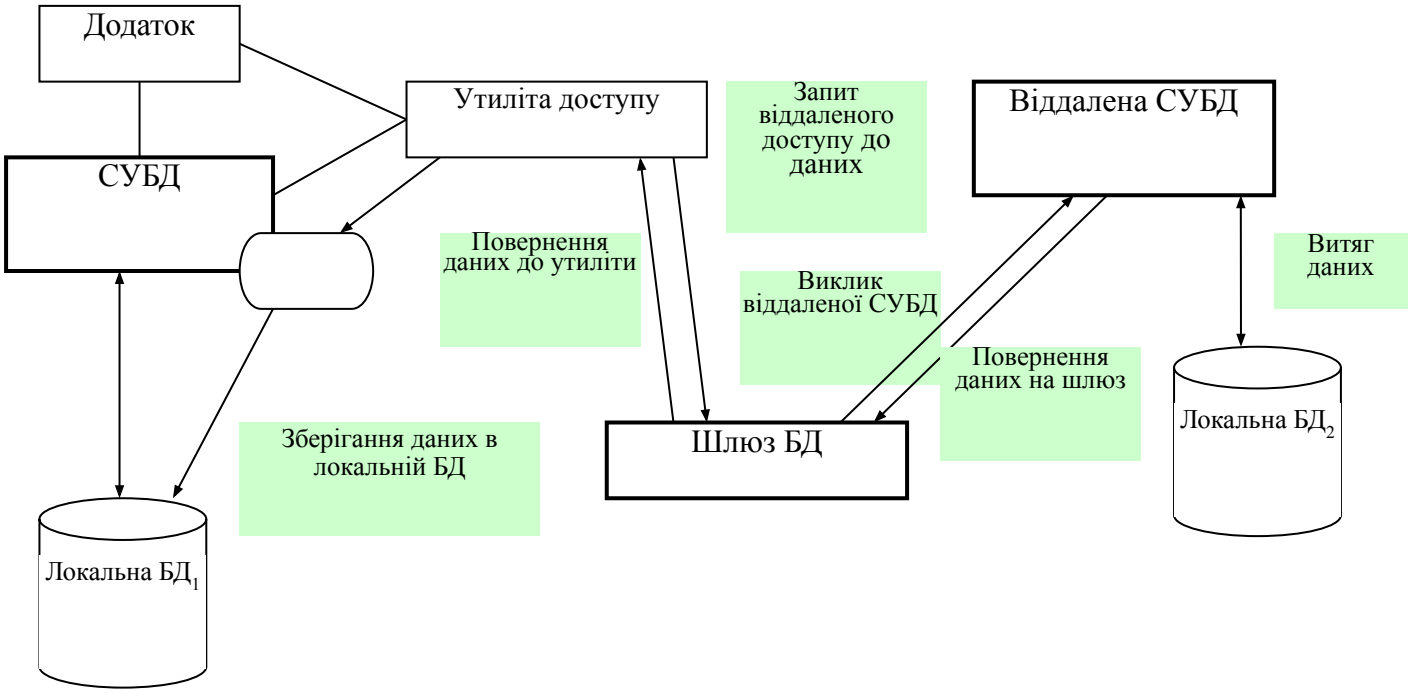


Схема функціонування компонентів *ORB*

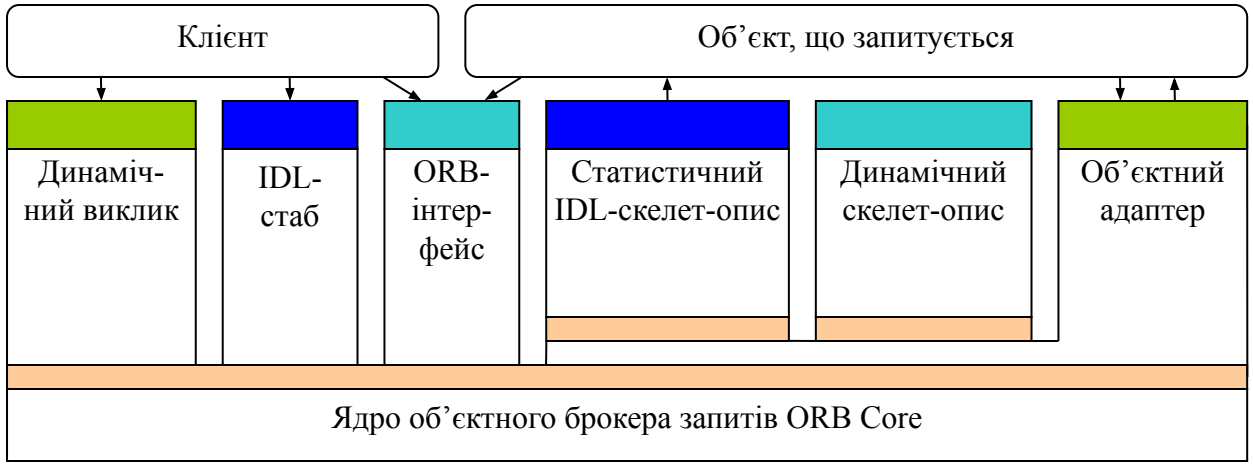
Системи з розподіленими додатками. CORBA

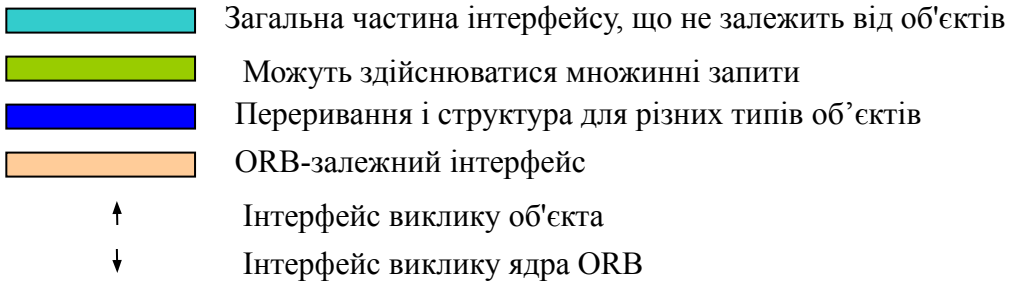
Опис ORB

- Клієнт - це користувач, який бажає виконати операцію над об'єктом в системі об'єктної реалізації.
- Система об'єктної реалізації - це фактично код і дані, які фізично містяться в об'єкті.
- ORB відповідальний за все механізми, необхідні, щоб знайти об'єктну реалізацію для запиту, підготувати її, реалізувати об'єкт і метод, що міститься в ньому, і відправити дані, що становлять запит, користувачеві.
- Інтерфейс, з яким має справу користувач, повністю незалежний від того, де об'єкт розміщений, яка мова програмування в ньому використовується, де це виконано, а також щодо будь-якого іншого аспекту, який абсолютно не впливає на інтерфейс об'єкта.

Системи з розподіленими додатками. CORBA

- *Структура інтерфейсу об'єктного брокера*



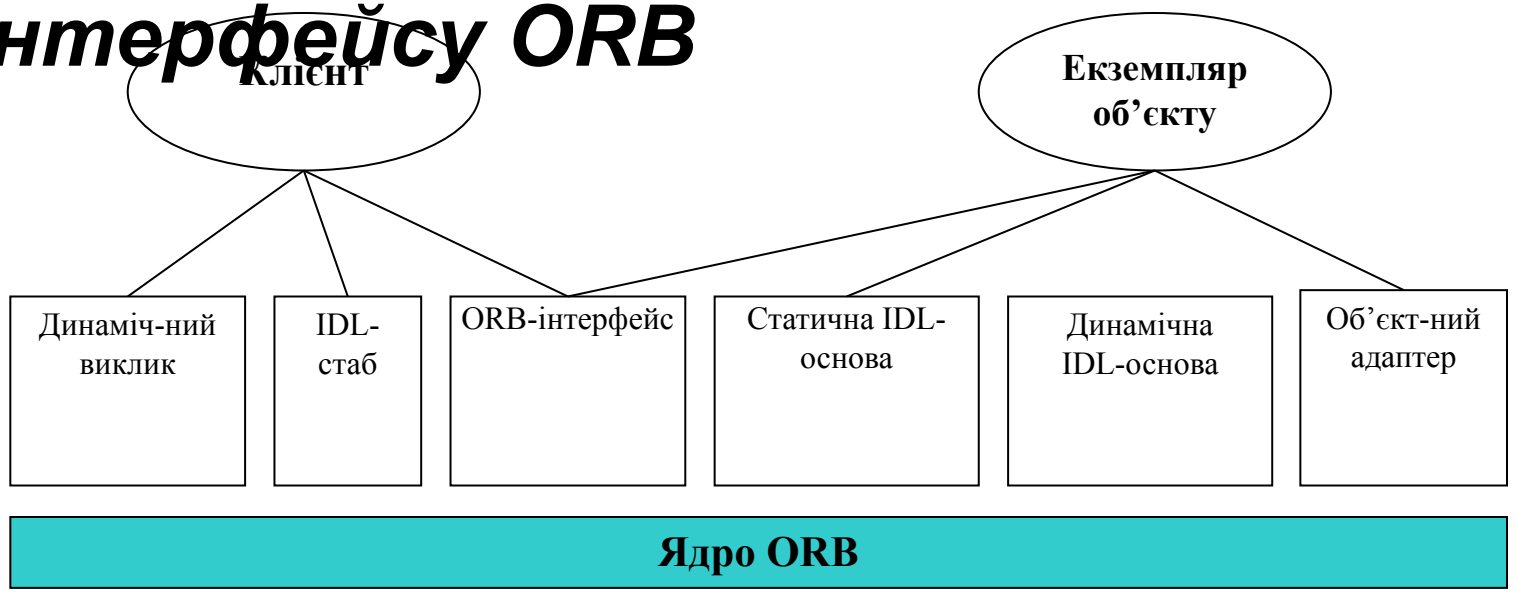


Системи з розподіленими додатками. CORBA

- Виклик операцій об'єкта-сервера, що розділяється, може бути зроблений статичним (IDL-стаб) і динамічним (Dynamic Invocation Interface) способом.
- Інтерфейси описуються, використовуючи мову визначення інтерфейсів, який отримав назву OMG Interface Definition Language (IDL). У разі статичного виклику ці описи інтерфейсів відображаються в програмний код на мовах C, C ++, Smalltalk. Інформація про інтерфейси об'єктів може бути отримана клієнтом двома способами: статично (compile time) і динамічно (runtime). Інтерфейси можуть бути також вказані за допомогою служби сховища інтерфейсів (Interface Repository). Цей сервіс представляє інтерфейси як об'єкти, забезпечуючи доступ до них під час роботи програми (в runtime-режимі).
- Користувач виконує запит при наявності доступу до об'єкта, коли знає і тип об'єкту і бажану функцію виконання.
- Користувач ініціалізує запит, викликаючи підпрограми статичного настроювання, які є специфічними для об'єкта, або будує запит, який динамічно оформлюється.

Системи з розподіленими додатками. CORBA

- ***Об'єктний адаптер (Object Adapter), як компонент в структурі інтерфейсу ORB***



Системи з розподіленими додатками. CORBA

- Головна функція об'єктного адаптера, що використовується для реалізації об'єкта, - надання клієнтам доступу до сервісів об'єктного брокера запитів ORB. Об'єктний адаптер забезпечує всі низькорівневі засоби для зв'язку об'єкта з його клієнтами.
- Основними завданнями об'єктного адаптера є:

-генерація посилань на віддалені об'єкти;

- виклик методу об'єкта, визначеного в IDL;

- забезпечення безпеки взаємодії;

- активізація і деактивізація об'єктів;

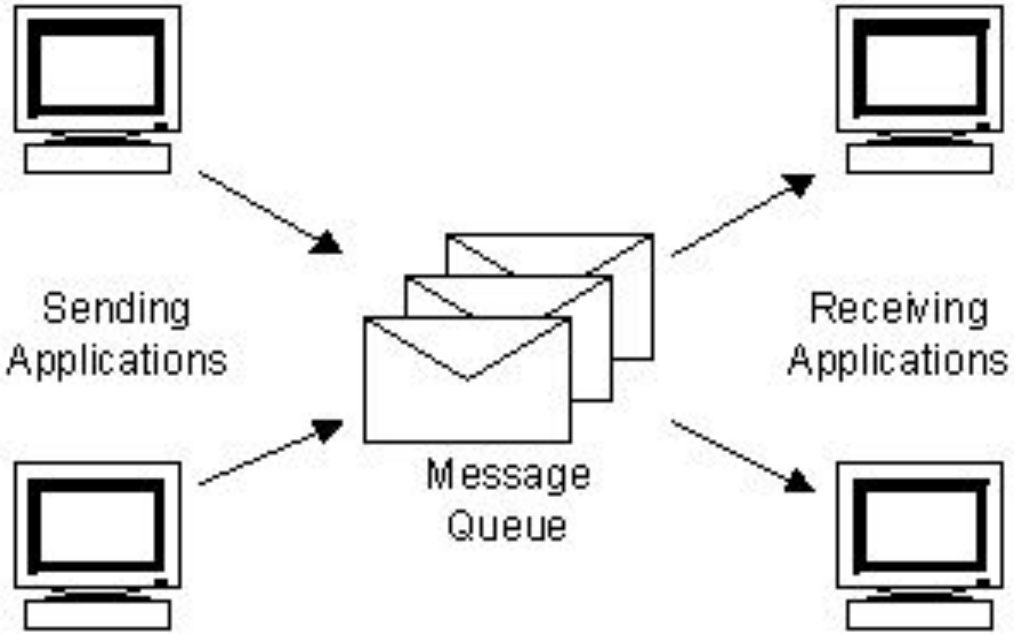
- встановлення відповідності між посиланнями на віддалені об'єкти (proxy) і реальними примірниками об'єктів;

- реєстрація об'єктів.

- Специфікація OMG CORBA визначає базовий об'єктний адаптер (Basic Object Adapter - BOA), який повинен бути реалізований у всіх брокерів запитів.

Системи з розподіленими додатками. MSMQ

- MSMQ - Служби черги повідомлень Microsoft (MSMQ)



Системи з розподіленими додатками. MSMQ

Базові компоненти архітектури MSMQ:

1) Сервер повідомлень;
2) Брокер (кластер) черги повідомлень;
3) Маршрутизатор повідомлень;
4) Планувальник карти портів (Port Mapper)
5) Менеджер постійного сховища (БД)
6) Менеджер безпеки

Системи з розподіленими додатками

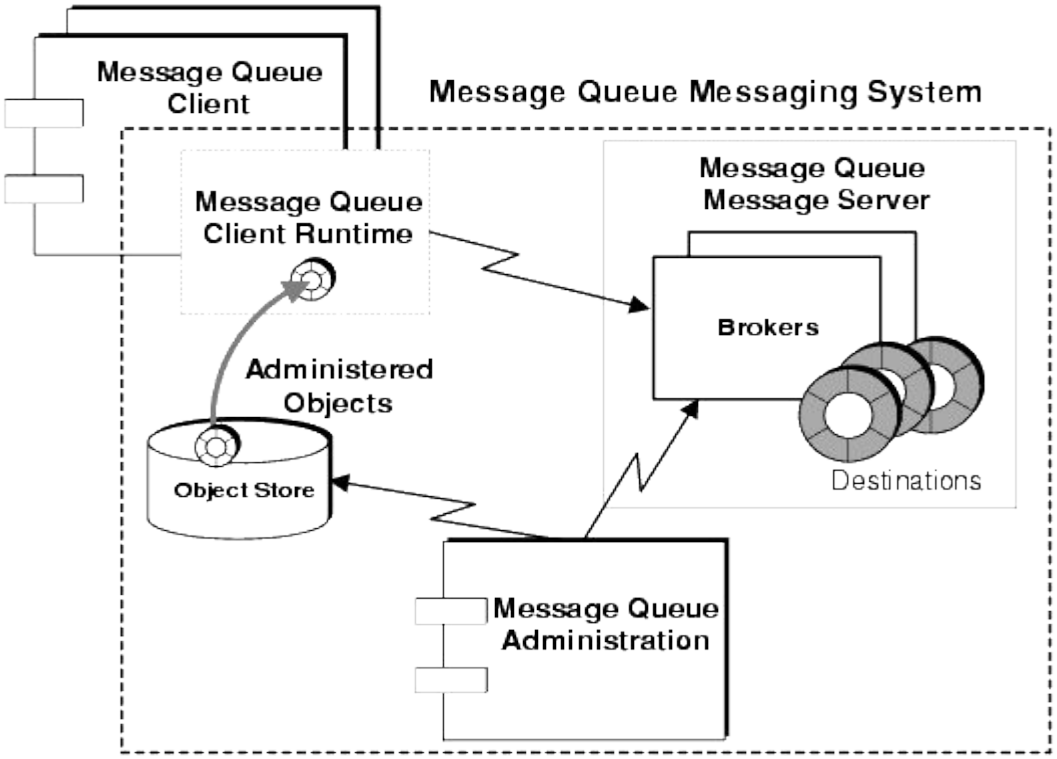
- Служби черги повідомлень Microsoft (MSMQ) - це життєво важлива інфраструктура обміну повідомленнями та засіб розробки для створення розподілених, слабосвязанних додатків обміну повідомленнями для операційної системи Windows всіх нотацій.
- Технологія черги повідомлень MSMQ дозволяє програмам, що працюють у різний час, обмінюватися даними через неоднорідні мережі та системи, які можуть бути тимчасово офлайн. Програми надсилають повідомлення в черги та читають повідомлення з черг.

Системи з розподіленими додатками

- Черга повідомлень забезпечує гарантовану доставку повідомлень, ефективну маршрутизацію, безпеку та обмін повідомленнями на основі пріоритетів.
- Вона може бути використана для реалізації рішень як для асинхронних, так і для синхронних сценаріїв, що вимагають високої продуктивності.
- Програми черги повідомлень можна розробляти за допомогою.
- API C ++ або COM-об'єктів. Програми можна створювати в будь-якому популярному середовищі розробки: наприклад, Visual C ++ ®, Visual Studio® .NET
- MSMQ 4.0 можна розгорнути на комп’ютерах під управлінням Microsoft Windows.

Системи з розподіленими додатками

- Система обміну повідомленнями Sun Java ™ System Message



Архітектура системи черг повідомлень

Системи з розподіленими додатками. MSMQ.

- Основними частинами системи обміну повідомленнями в черзі повідомлень є:

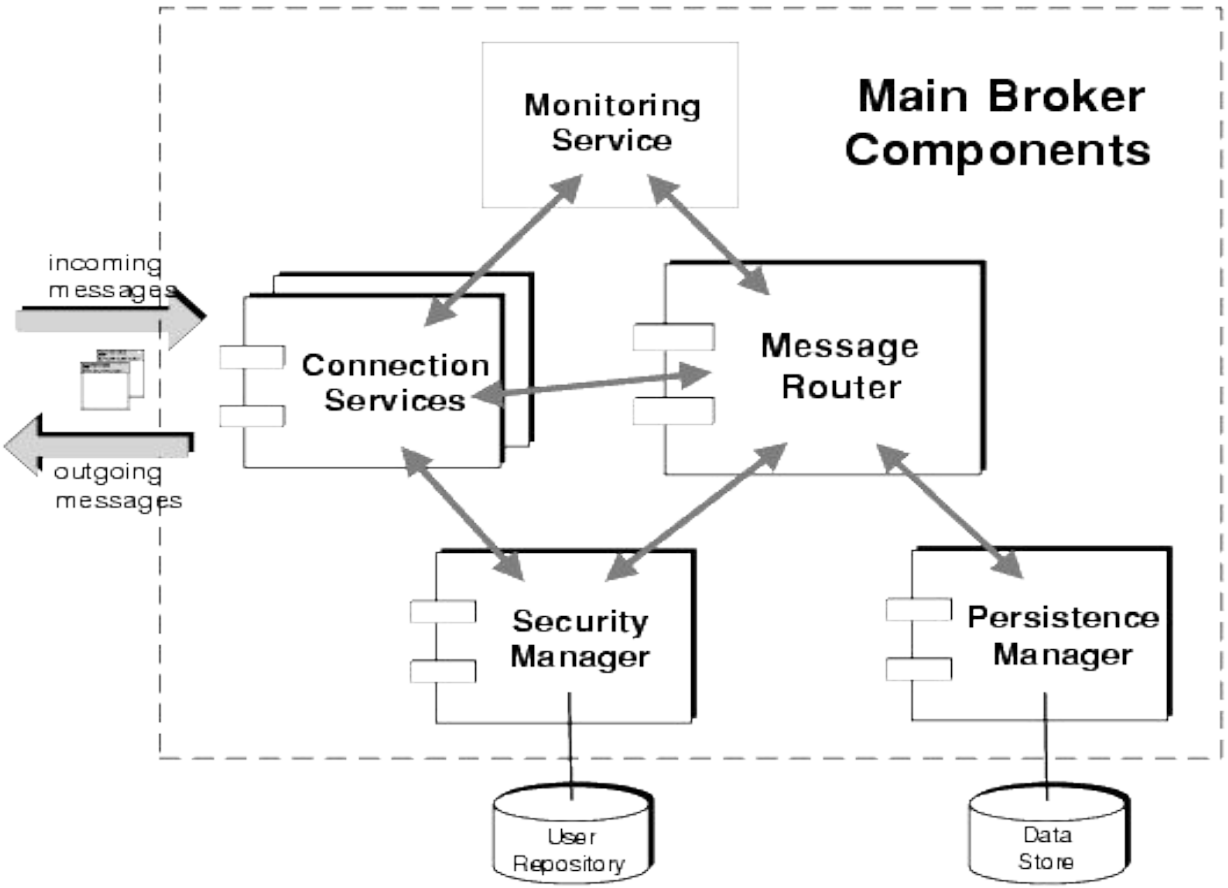
a) Message Queue Message Server (Сервер повідомлень черги повідомлень)
b) Message Queue Client Runtime (Час роботи клієнта черги повідомлень)
c) Message Queue Administered Objects (Об’єкти, якими керує черга повідомлень)
d) Message Queue Administration (Адміністрація черги повідомлень)

Системи з розподіленими додатками. MSMQ.

- Брокер черги повідомлень - надає послуги доставки системи обміну повідомленнями черги повідомлень. Доставка повідомлень покладається на ряд допоміжних компонентів, які обробляють послуги з’єднання, маршрутизацію та доставку повідомлень, постійність, безпеку та ведення журналу.
- Сервер повідомлень може використовувати один або кілька екземплярів посередника.
- Для здійснення доставки повідомлень брокер повинен налаштувати канали зв'язку з клієнтами, виконати автентифікацію та авторизацію, належним чином маршрутизувати повідомлення, гарантувати надійну доставку та надати дані для моніторингу роботи системи.

Системи з розподіленими додатками. MSMQ.

- Компоненти брокерської послуги

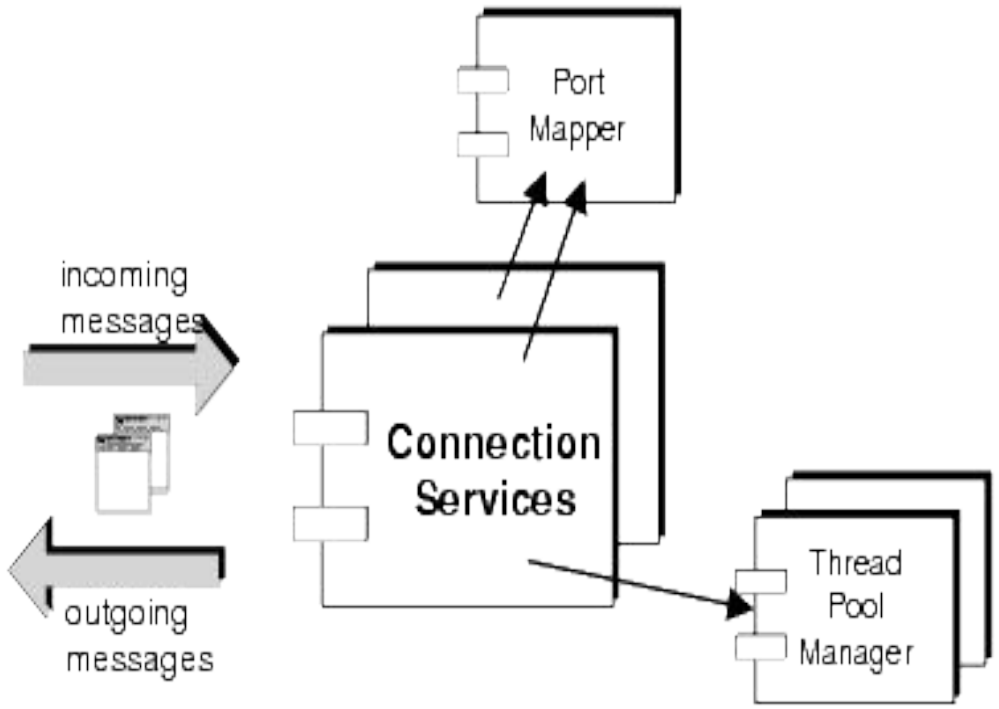


Системи з розподіленими додатками. MSMQ.

- Ви можете налаштувати брокера на запуск будь-якої або всіх цих служб підключення. Кожна послуга підключення доступна в певному порту, вказаному іменем хоста брокера та номером порту. Порт може бути динамічно розподілений або ви можете вказати порт, у якому доступна послуга з'єднання постійно.
- Кожна служба реєструє себе за допомогою спільного портового перетворювача, але має власний диспетчер пулу потоків, як показано на рисунку.

Системи з розподіленими додатками. MSMQ.

- Підтримка служб підключення



Системи з розподіленими додатками. MSMQ.

- Port Mapper

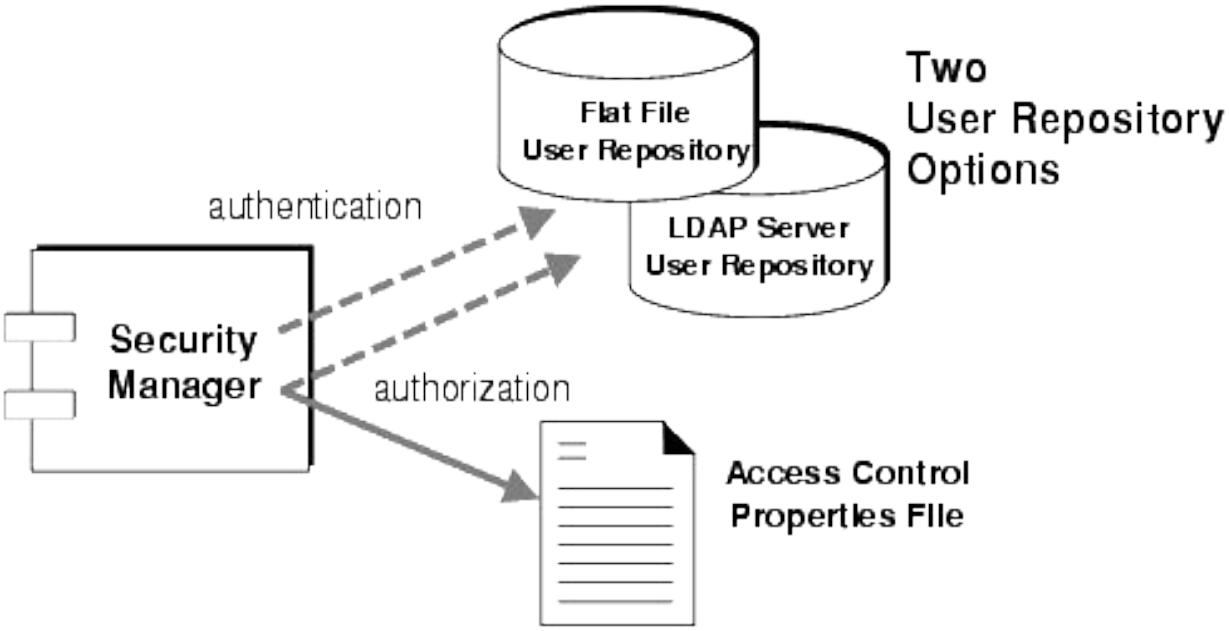
Черга повідомлень надає Port Mapper, який відображає порти для різних служб підключення. Port Mapper знаходиться на стандартному номері порту, 7676. Коли клієнт встановлює з'єднання з посередником, він спочатку зв'язується з Port Mapper із запитом номера порту потрібної послуги з'єднання.

- Маршрутизатор повідомлень

Після встановлення зв’язків між клієнтами та брокером за допомогою підтримуваних служб зв’язку маршрутизація та доставка повідомлень можуть продовжуватися.

Системи з розподіленими додатками. MSMQ.

- Менеджер безпеки



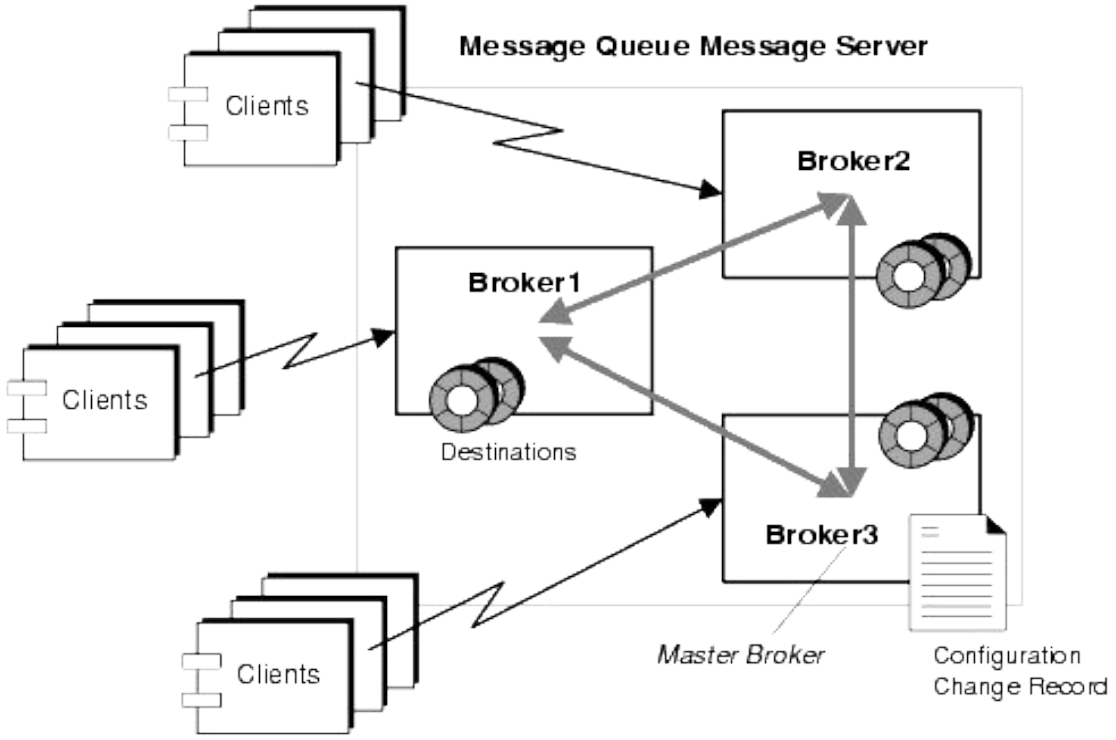
Системи з розподіленими додатками. MSMQ.

Функції Менежера безпеки забезпечують:

- Доступ до пунктів призначення: створення споживача, виробника або браузера черг для будь-якого даного пункту призначення або всіх пунктів призначення
- Автоматичне створення напрямків пересилання повідомлень.
- Послуга адміністратора дозволяє користувачеві виконувати адміністративні функції, такі як створення пунктів призначення, моніторинг та контроль брокера.
- Користувач будь-якої іншої групи, яку ви визначите, не може за замовчуванням отримати підключення до служби адміністратора.

Системи з розподіленими додатками. MSMQ.

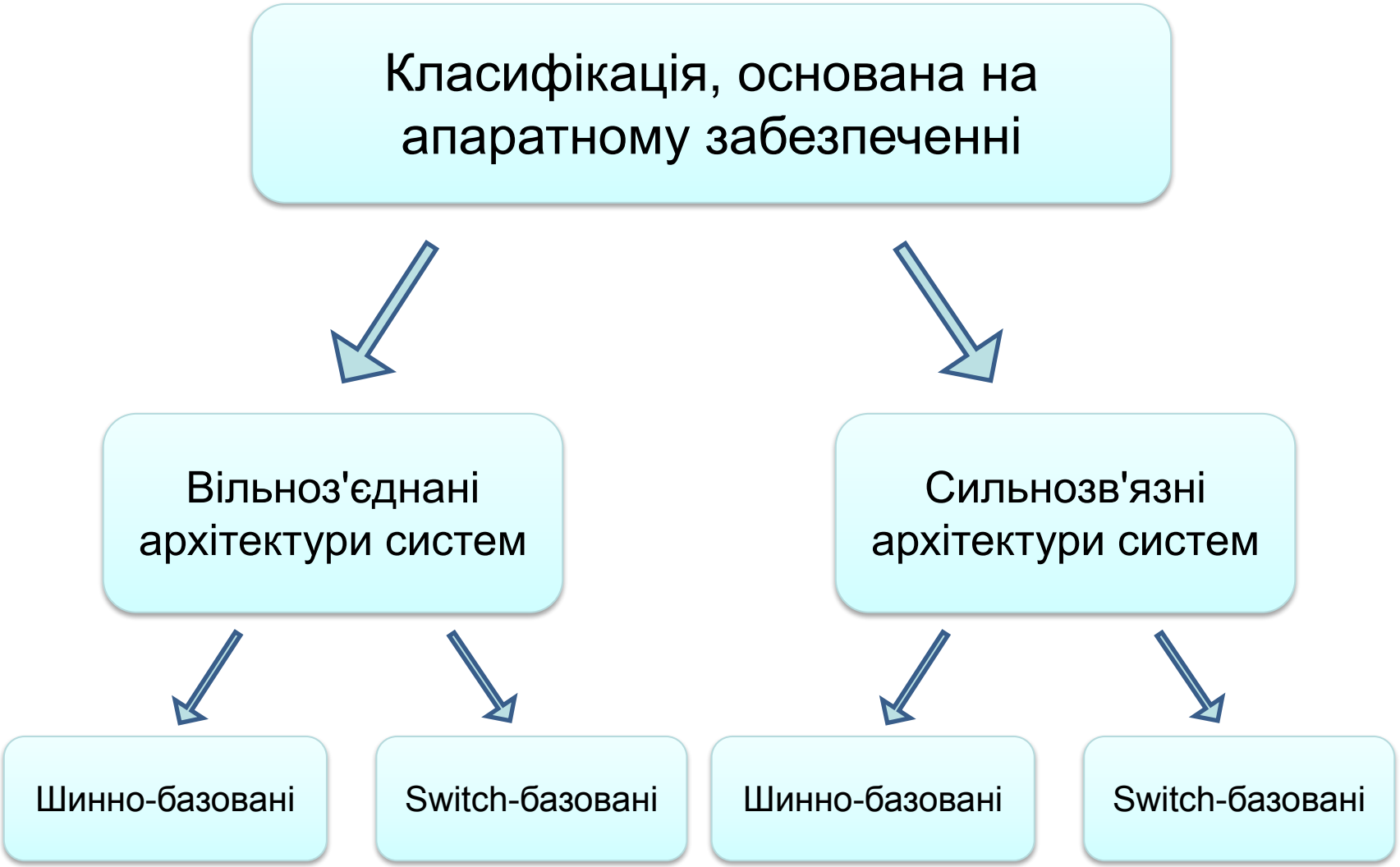
- Архітектура мульти-брокера (кластера)



Системи з розподіленими додатками. MSMQ.

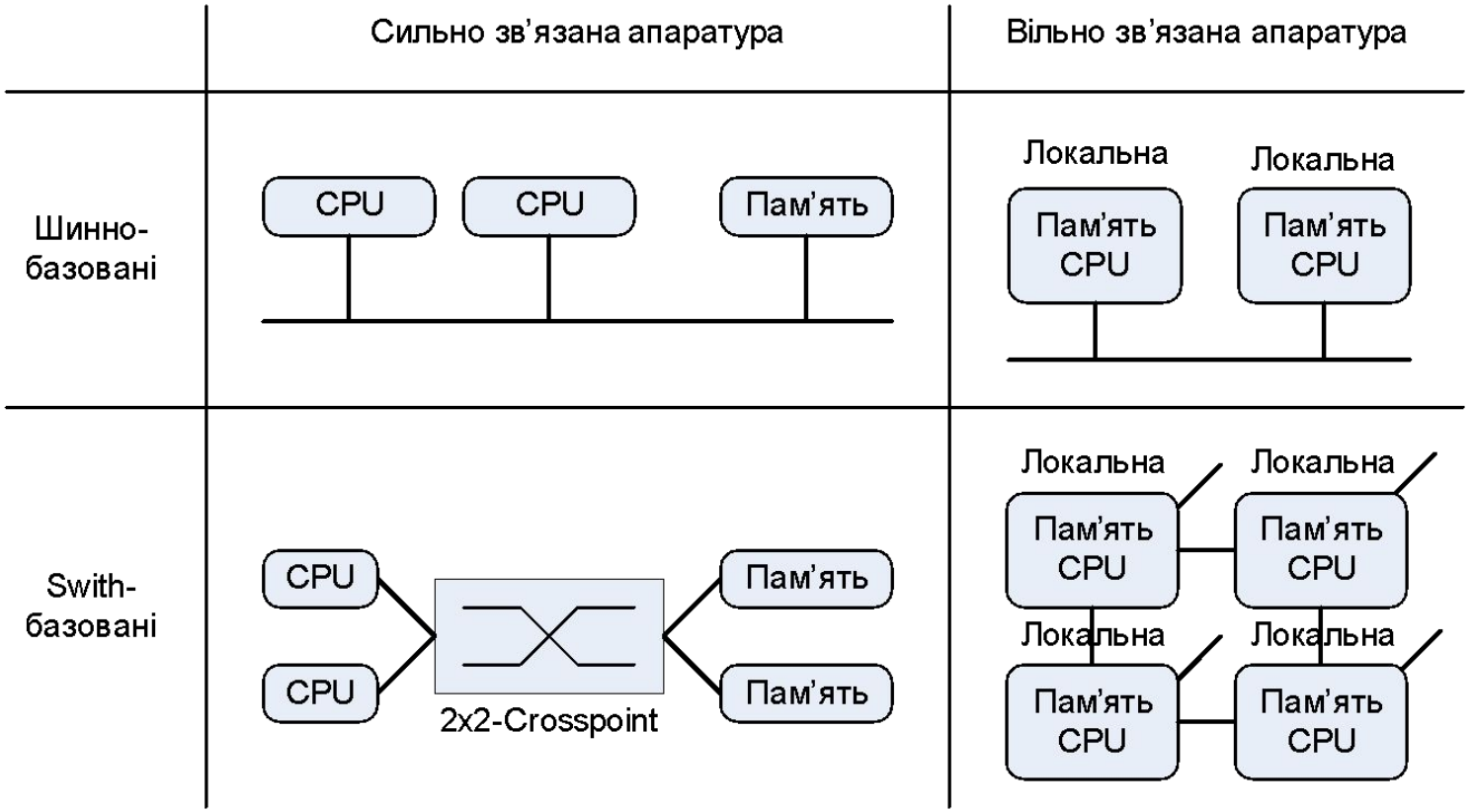
- Черга повідомлень підтримує лише повністю підключені кластери, тобто топологію, при якій кожен брокер безпосередньо підключений до кожного іншого брокера в кластері.
- Сервер мультиброкерських повідомлень дозволяє розподіляти підключення клієнтів між низкою екземплярів брокера
- Кожен клієнт підключається до окремого брокера (його домашнього брокера) і надсилає та отримує повідомлення так, ніби домашній брокер був єдиним брокером у кластері.
- Однак, домашній брокер працює в парі з іншими брокерами в кластері, щоб надавати послуги доставки виробникам та споживачам повідомлень, до яких він безпосередньо прив'язаний.

КЛАСИФІКАЦІЯ РС



КЛАСИФІКАЦІЯ РС

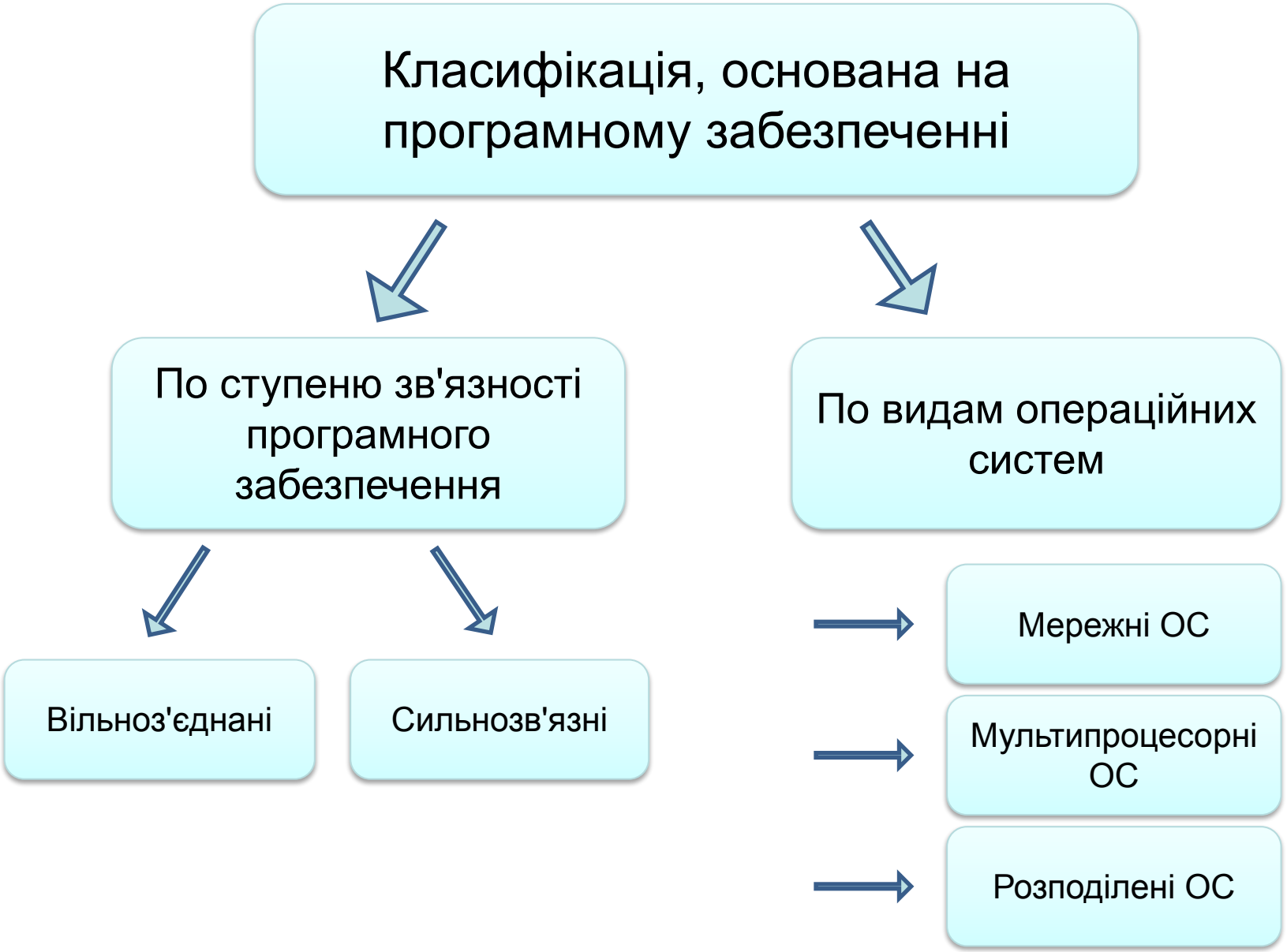
- *Приклади шинно- і switch-базованих систем з і без спільної пам'яті*



КЛАСИФІКАЦІЯ РС

- *Шинно-базовані* - плата, шина, кабель або інше середовище з'єднує всі машини між собою.
- *Switch-базовані* - не мають єдиної магістралі. Замість неї від машини до машини тягнуться окремі канали, виконані із застосуванням різних технологій зв'язку. Повідомлення передаються по каналах з узгодженням рішення про комутацію з конкретним вихідним каналом для кожного з них.

КЛАСИФІКАЦІЯ РС



КЛАСИФІКАЦІЯ РС

По ступеню зв'язності програмного забезпечення

- *Вільно з'єднане програмне забезпечення* дозволяє комп'ютерам і користувачам розподіленої системи, по суті, виконувати незалежну одну від іншої роботу й тільки в деяких випадках - якщо буде потреба - інтеграцію.

  Приклад - оператори персонального комп'ютера із власним CPU, власною пам'яттю й операційною системою, які спільно використовують лазерний принтер.
- *Сильно зв'язне програмне забезпечення* реалізує одну програму на різних комп'ютерах одночасно.

КЛАСИФІКАЦІЯ РС

По видам операційних систем

- *Мережна ОС* припускає, що кожний користувач має свою власну робочу станцію (Workstation) із власною операційною системою. Комунікація використовується в цьому випадку для доступу до спільних файлів.
- *Мультипроцесорна ОС* служить часто для спеціальних цілей, як, наприклад, для системи банку даних. Характерним у цьому випадку є наявність окремого процесу в спільній пам'яті. Комунікація між окремими компонентами такої системи відбувається для сполучення процесів інформаційного обміну.
- *Розподілена ОС*, створює для користувача ілюзію, що вся мережа є єдиним великим комп'ютером, де зберігаються вся інформація й всі прикладні програми. Комунікація необхідна в такій системі для обміну повідомленнями.

КОНЦЕПЦІЇ ПРОГРАМНИХ РІШЕНЬ

- ***Розподілені операційні системи:***
  - використовуються для управління мультипроцесорними й гомогенними мультикомп’ютерними системами.
  - основна мета розподіленої операційної системи полягає в прихованні тонкощів управління апаратним забезпеченням, що одночасно використовується багатьма процесами.
- ***Мережні операційні системи.***
  - використовуються для управління гетерогенними мультикомп’ютерними системами

МУЛЬТИПРОЦЕСОРНА ОС

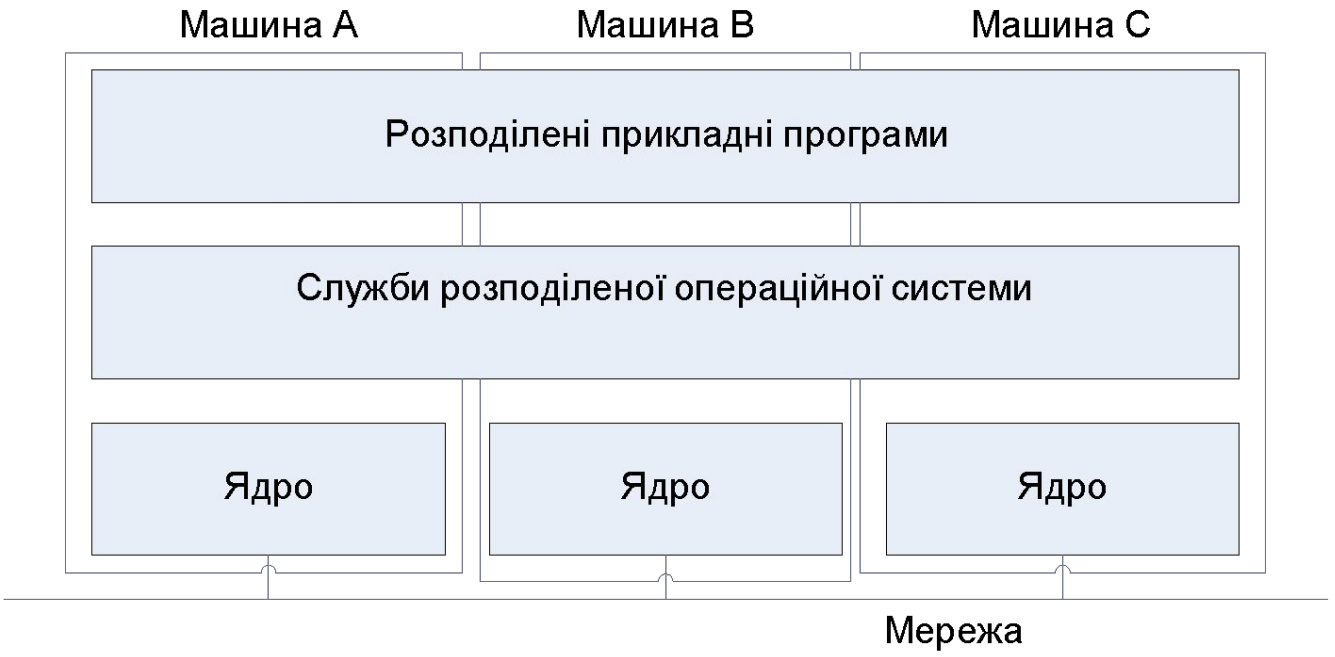
- Всі структури даних розміщуються в пам'яті
- Дані доступні декільком процесорам і мають бути захищені від паралельного доступу
- Націлені на підтримку високої продуктивності конфігурацій з декількома процесорами
- Основне завдання – забезпечити прозорість числа процесорів для прикладних програм.

МУЛЬТИКОМП’ЮТЕРНА ОС

- ***Мультикомп’ютерні операційні системи*** мають набагато більш різноманітну структуру й значно складніші, ніж мультипроцесорні.
- Для мультикомп’ютерних операційних систем структури даних, необхідні для управління системними ресурсами, не повинні відповідати умові легкості спільного використання, оскільки їх не потрібно розміщувати фізично у загальну пам'ять. Єдиним можливим видом зв'язку є передача повідомлень (message passing).

МУЛЬТИКОМП’ЮТЕРНА ОС

- Загальна структура мультикомп’ютерних ОС:



ПРОМІЖНЕ СЕРЕДОВИЩЕ

- Система проміжного рівня забезпечує скоординовану роботу мереж і ОС із можливостями використання їхнього програмного інтерфейсу.
- Ефективне проміжне середовище повинне мати можливість організації взаємодії групи комп'ютерів мережі без порушення стека протоколів TCP/IP.

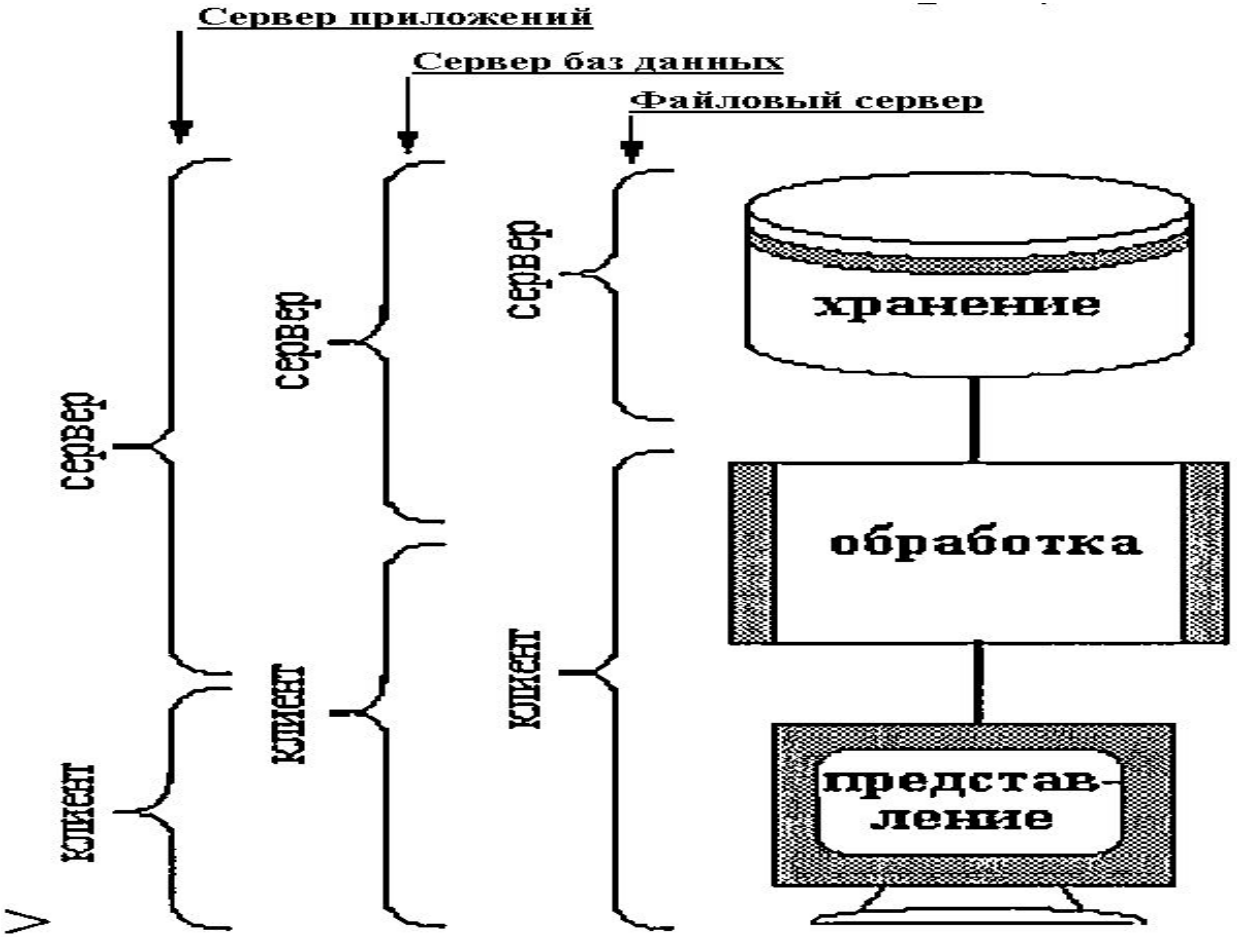
ПРОМІЖНЕ СЕРЕДОВИЩЕ

- Загальна структура розподілених систем із проміжним рівнем

ПОНЯТТЯ РОЗПОДІЛЕНОГО СЕРЕДОВИЩА

- Розподілене середовище являє собою віртуальний обчислювальний простір, який може обмежуватися однією розподіленою системою, а може містити кілька взаємодіючих розподілених систем.
- Такий віртуальний обчислювальний простір надається користувачеві у вигляді систематизованого сховища інформаційних та програмних ресурсів, має певну структуру, зрозумілу систему адресації ресурсів та певні моделі обчислювальних процесів або бізнес-процесів даного користувача.

Розподіл функцій для серверів



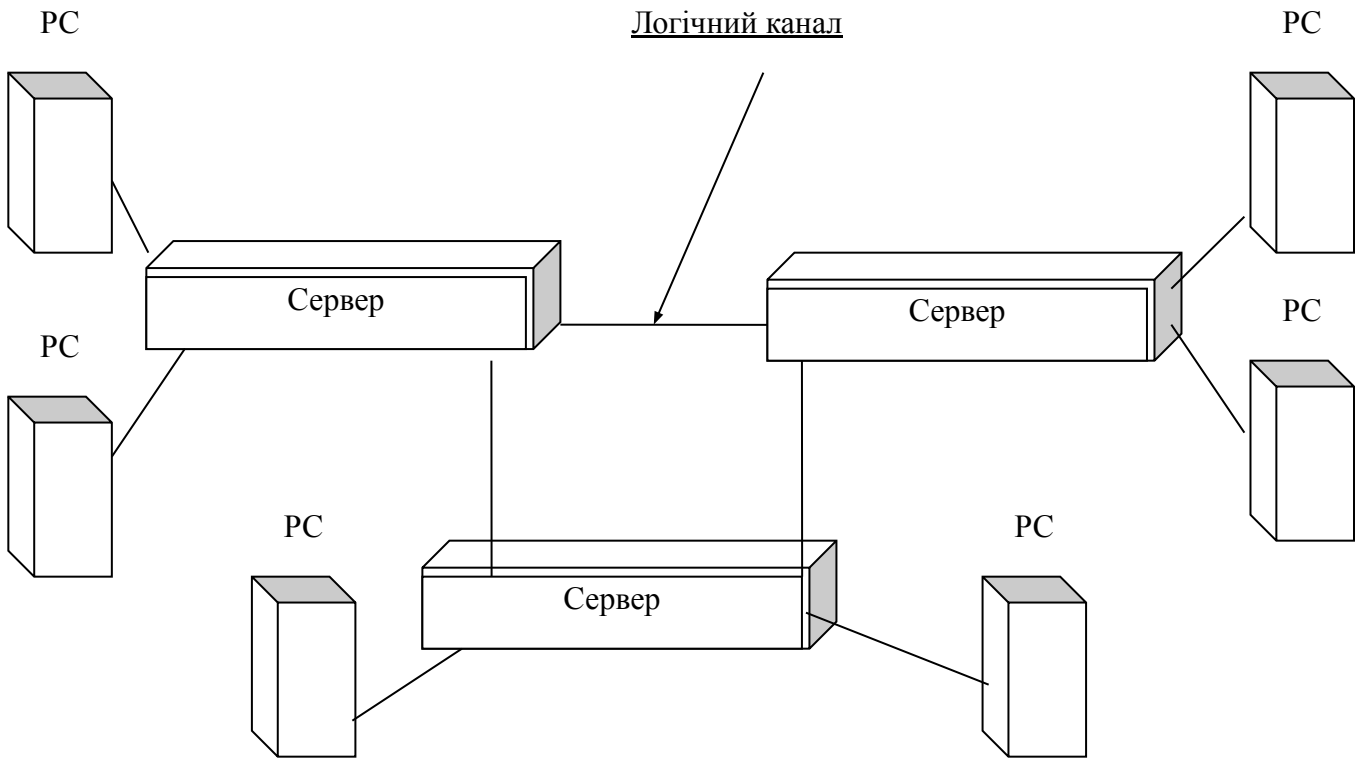
Розподілене середовище DDP

Протокол розподілених даних (**Distributed Data Protocol** або **DDP**)

- Це клієнт-серверний Це клієнт-серверний протокол для запитів та оновлення серверної бази даних, а також для синхронізації тих оновлень поміж усіма клієнтами.
- Протокол використовує шаблон повідомлень публікація-підписка.

Логічна структура середовища DDP

Середовище має трьохступеневу архітектуру:
прикладна програма - база даних - клієнт.



Планування обробки

Три моделі обробки:

- обробка в одноранговій локальній мережі;
- централизована обробка в Датацентрі;
- обробка в моделі клієнт/сервер.

Планування обробки

Три основних рівня маніпулювання даними:

- зберігання даних;
- виконання додатків у процесі вибірки і обробка даних для потреб прикладної задачі;
- надавання даних і результатів обробки кінцевому клієнту.

Функції розподіленого середовища

Прикладні служби:

- каталогів, яка дозволяє клієнтам знаходити потрібні сервери;
- інтерфейса багатопоточної обробки;
- віддаленого визову процедур;
- обслуговування файлів;
- безппеки даних;
- часу, яка синхронізує час в абонентських системах.

Комп’ютерні кластери

- «**Кластер** — це різновид паралельної або розподіленої системи, яка:

a) складається з декількох зв'язаних між собою комп'ютерів;

b) використовується як єдиний, уніфікований комп'ютерний ресурс».

I. Зазвичай розрізняють наступні основні види кластерів:

1) відмовостійкі кластери (High-availability clusters, HA, кластери високої доступності)
2) кластери з балансуванням навантаження (Load balancing clusters)
3) обчислювальні кластери (High perfomance computing clusters)
4) grid-системи

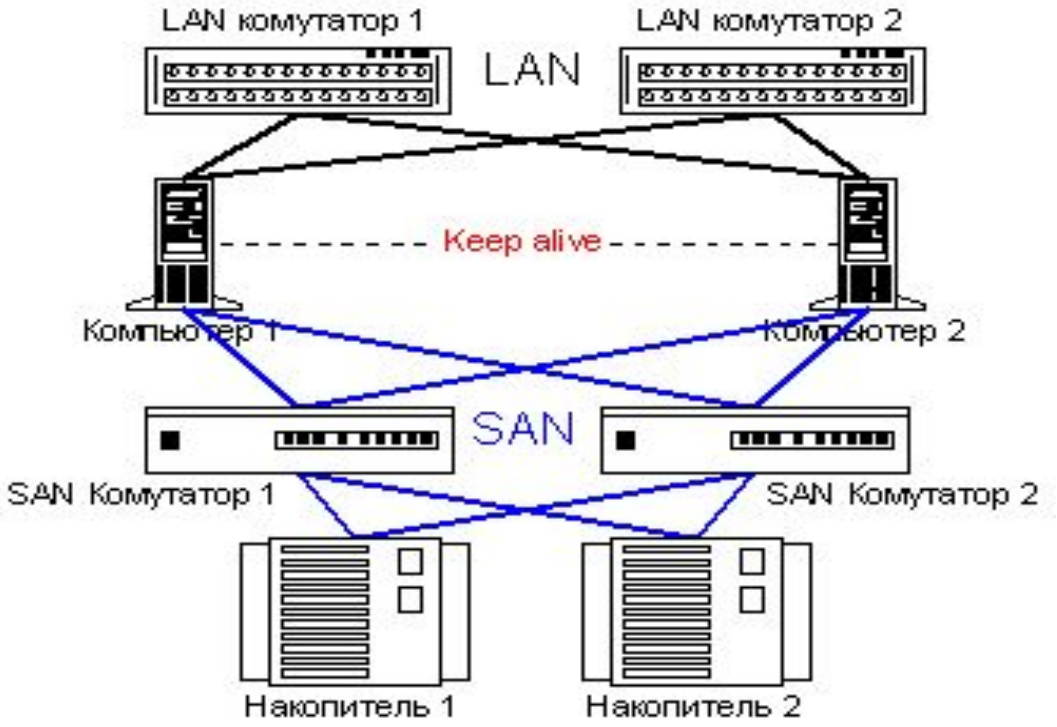
Комп’ютерні кластери

II. **Відмовостійкий кластер, кластер високої доступності** (англ. *High-Availability cluster, **HA** cluster*) — це кластер, що спроектований відповідно до методик забезпечення високої доступності і гарантує мінімальний час простою за рахунок апаратної надмірності.

- Без кластеризації збій сервера призводить до того, що підтримувані ним додатки або мережеві сервіси виявляються недоступними.
- Відмовостійка кластеризація виправляє дану ситуацію, перезапускаючи додатки на інших вузлах кластера без втручання адміністратора, в разі виявлення апаратних або програмних збоїв.

Комп’ютерні кластери

- *Кластерна система з відсутністю точек відмов типу* NSPF (No Single Point of Failure)



Комп’ютерні кластери

Найчастіше зустрічаються двовузлові ***HA***-кластери — це мінімальна конфігурація, необхідна для забезпечення відмовостійкості. Але часто кластери містять велику кількість вузлів. Всі ці конфігурації, як правило, можуть бути описані однією з наступних моделей:

a) **Active / active** — Частина трафіку — Частина трафіку, що обробляє відмову вузла, перенаправляється до працюючого вузла або розподіляється між кількома працюючими вузлами. Така схема використовується в тому випадку, коли вузли мають однорідну конфігурацію програмного забезпечення і виконують однакову задачу [2].

b) **Active / passive** — Має повне резервування (дійсну копію) кожного вузла. Резерв використовується тільки тоді, коли відмовляє відповідний основний вузол. Ця конфігурація вимагає значних надлишкових апаратних засобів.

c) **N + 1** — Має один повноцінний резервний вузол, до якого в момент відмови переходить роль вузла, що доступний в даний момент часу. У разі гетерогенної програмної конфігурації основних вузлів додатковий вузол, повинен бути здатний взяти на себе роль кожного з основних вузлів, за резервування котрих, він відповідає. Така схема застосовується в кластерах, які обслуговують кілька різнорідних сервісів, що працюють одночасно.

Комп’ютерні кластери

d) **N + M** — Якщо один кластер обслуговує кілька сервісів, включення в кластер єдиного резервного вузла може виявитися недостатнім для належного рівня резервування. У таких випадках в кластер включається кілька резервних серверів, кількість яких є компромісом між ціною рішення і необхідної надійності

e) N-к-1 — Дозволяє резервному вузлу вмикатися в роботу тимчасово, поки вузол, що відмовив, не буде відновлений, після цього вихідне навантаження повертається на основний вузол для збереження вихідного рівня доступності системи.

f) N-к-N — це поєднання active / active і N + M кластерів. У N-к-N кластері сервіси, екземпляри систем або об'єднання вузлів, що відмовили, перерозподіляються між іншими активними вузлами. Тим самим усувається (як в схемі active / active) необхідність окремого резервного вузла, але при цьому всі вузли кластера повинні володіти деякою надлишковою потужністю понад мінімально необхідною[3]. Терміни логічний хост або кластерний логічний хост використовуються для позначення мережевої адреси, яка використовується для доступу до сервісів, що надаються кластером.

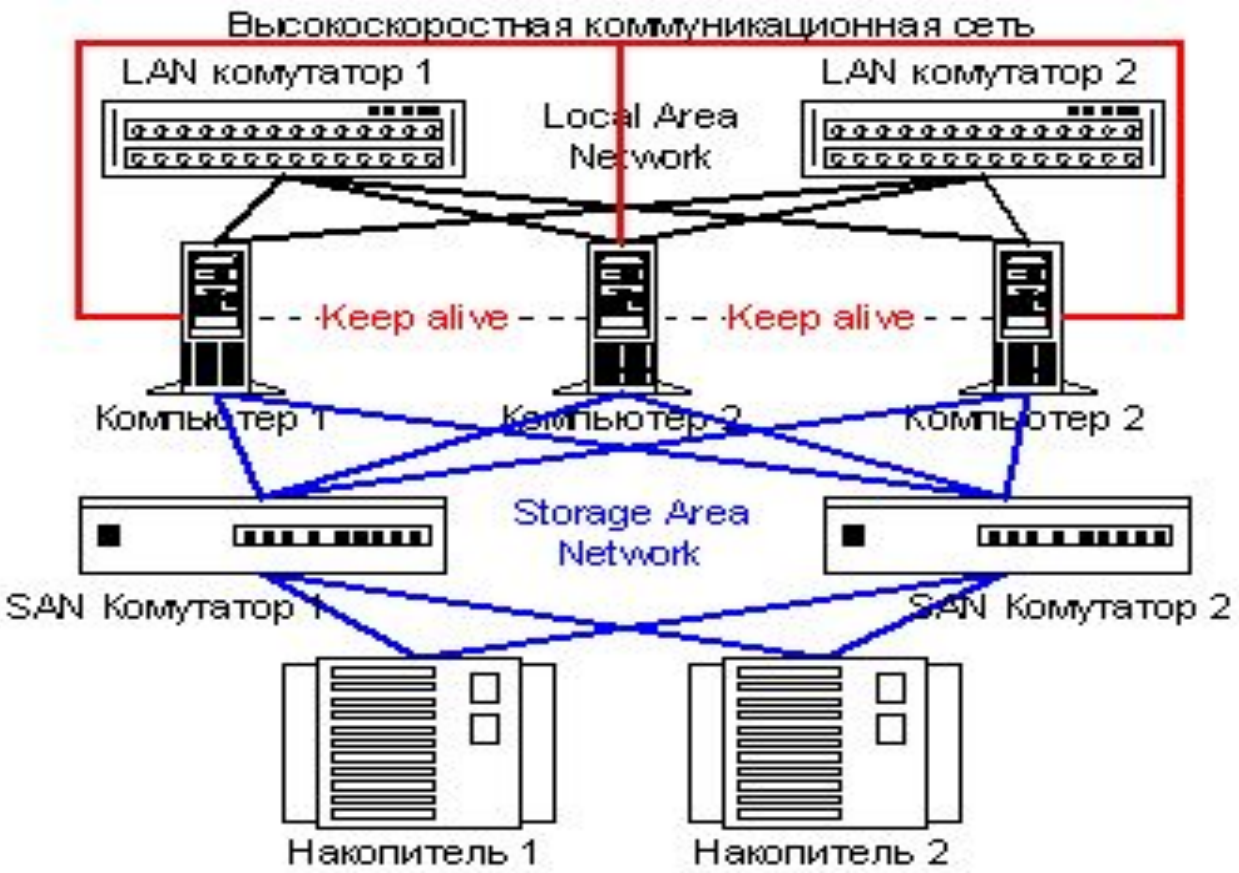
Комп’ютерні кластери

III. Методи забезпечення надійності кластерного вузла:

1) Резервування і реплікація дисків: відмова частини внутрішніх дисків не призводить до збоїв системи.
2) Резервування зовнішніх мережевих з'єднань: пошкодження кабелю, відмова комутатора або мережевого інтерфейсу не призводять до повного відключення від мережі.
3) Резервування внутрішніх з'єднань мережі зберігання даних (SAN): пошкодження кабелю, збій комутатора або мережевого інтерфейсу не приведуть до втрати з'єднання серверів зі сховищем.
4) Надлишкові вводи електроживлення різного устаткування, як правило, захищені джерелами безперебійного живленняНадлишкові вводи електроживлення різного устаткування, як правило, захищені джерелами безперебійного живлення і резервуються блоки живлення: відмова одиничного вводу, UPS або БЖ не призводить до критичної

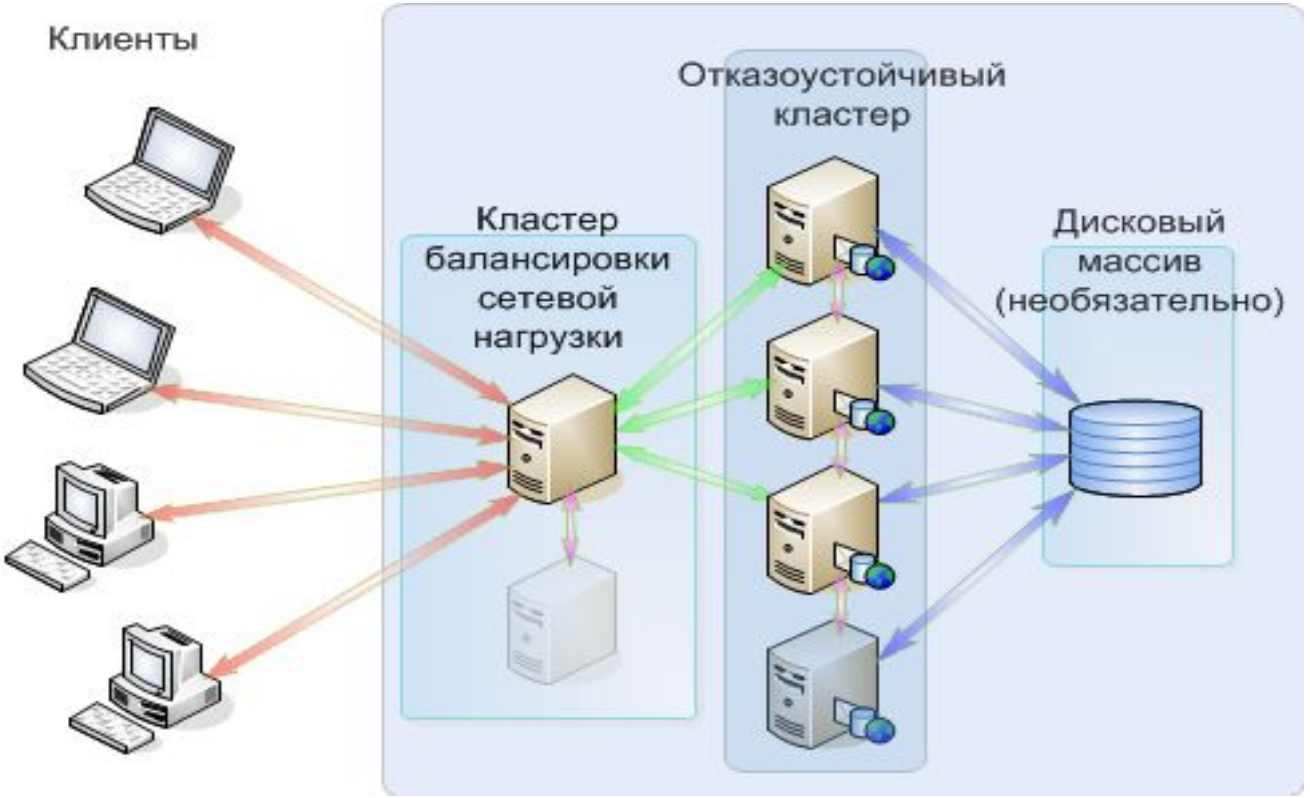
Комп’ютерні кластери

- Архітектура кластерного вузла з 3-х серверів



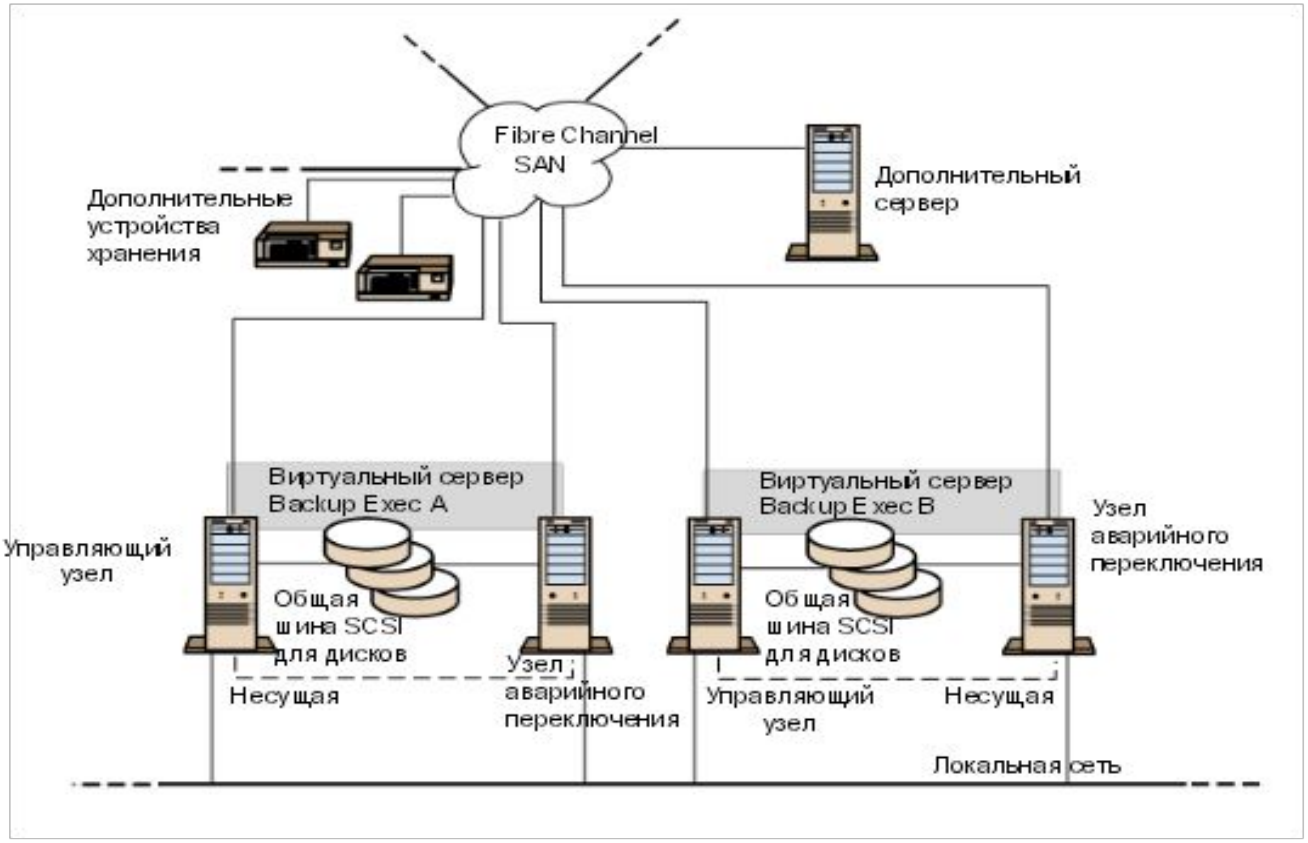
Комп’ютерні кластери

- Архітектура вузла з балансуванням навантаження



Комп’ютерні кластери

- Два 2-х вузлових кластера в оптоволоконній мережі з компонентом SAN SSO



Комп’ютерні кластери

- Не кожен додаток може працювати в високодоступному кластерному середовищі. Відповідні рішення повинні бути закладені на ранній стадії розробки програмного забезпечення.

Комп’ютерні кластери

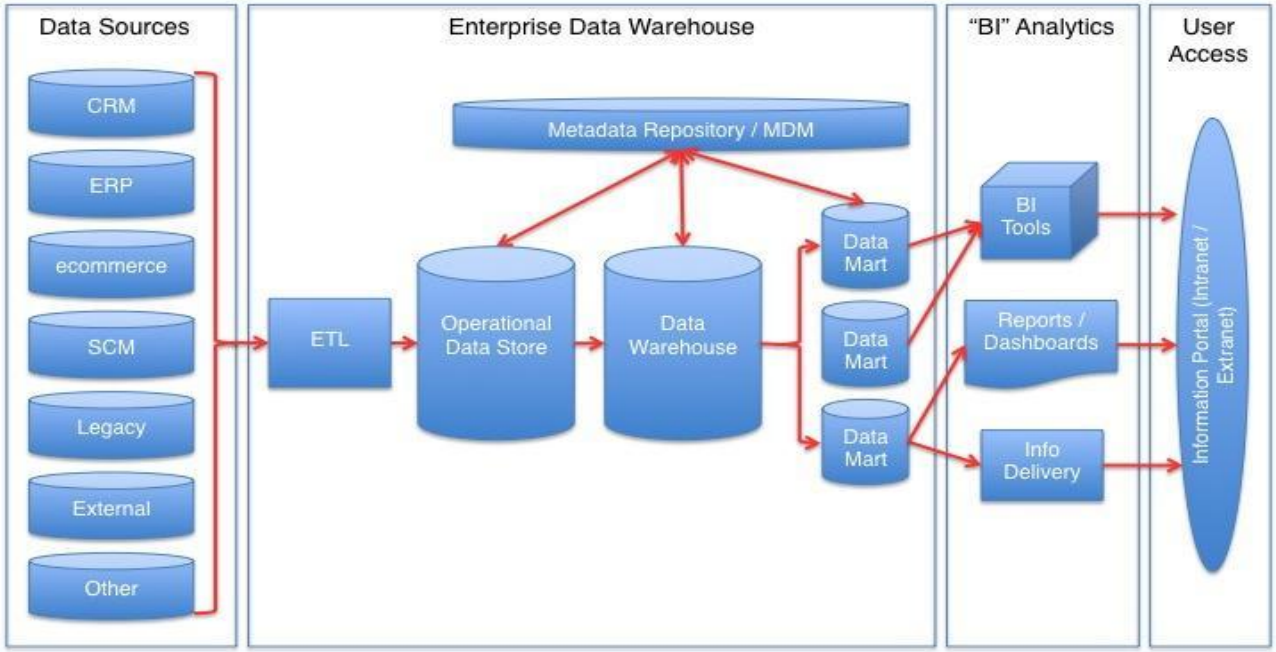
Вимоги до кластерних додатків

- Для роботи в HA-кластері додаток повинен відповідати, як мінімум, наступним технічним вимогам, останні два з яких мають вирішальне значення для його надійної роботи в кластері і які повинні повною мірою задовольнити наступне:

1) повинен бути відносно простий спосіб запуску, зупинки, примусової зупинки, і перевірки стану додатка. На практиці це означає, що додаток має мати інтерфейсповинен бути відносно простий спосіб запуску, зупинки, примусової зупинки, і перевірки стану додатка. На практиці це означає, що додаток має мати інтерфейс командної стрічки або скрипту для управління ним, в тому числі для роботи з декількома запущеними екземплярами додатка.
2) додаток повинен вміти використовувати загальне сховище даних (NASдодаток повинен вміти використовувати загальне сховище даних (NAS / SAN).
3) додаток має зберігати в загальному сховищі максимально можливу кількість даних про свій поточний стан.
4) додаток не повинен пошкоджувати дані при виході з ладу або відновленні з збереженого стану.

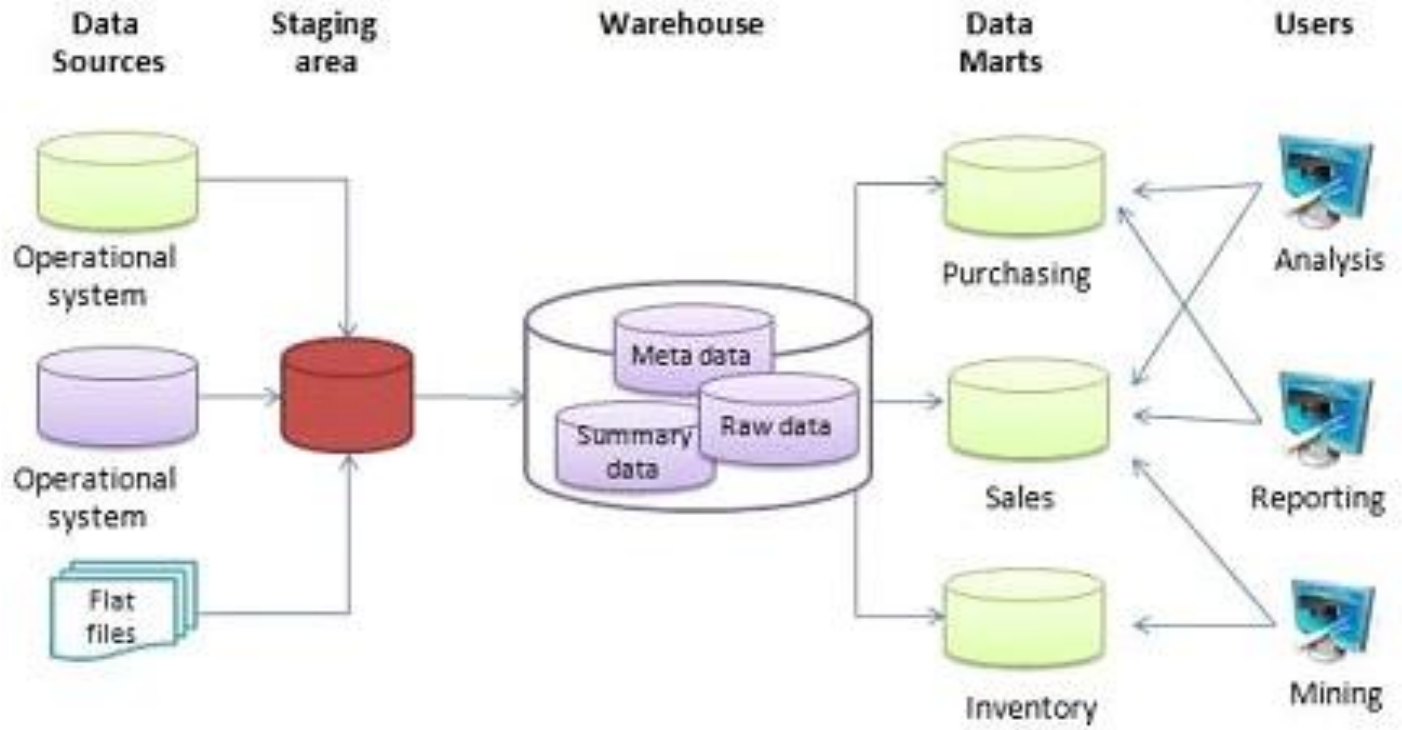
Сховища даних

- Загальна схема сховища даних



Сховища даних

- Базова архітектура сховища даних



Складові сховища даних

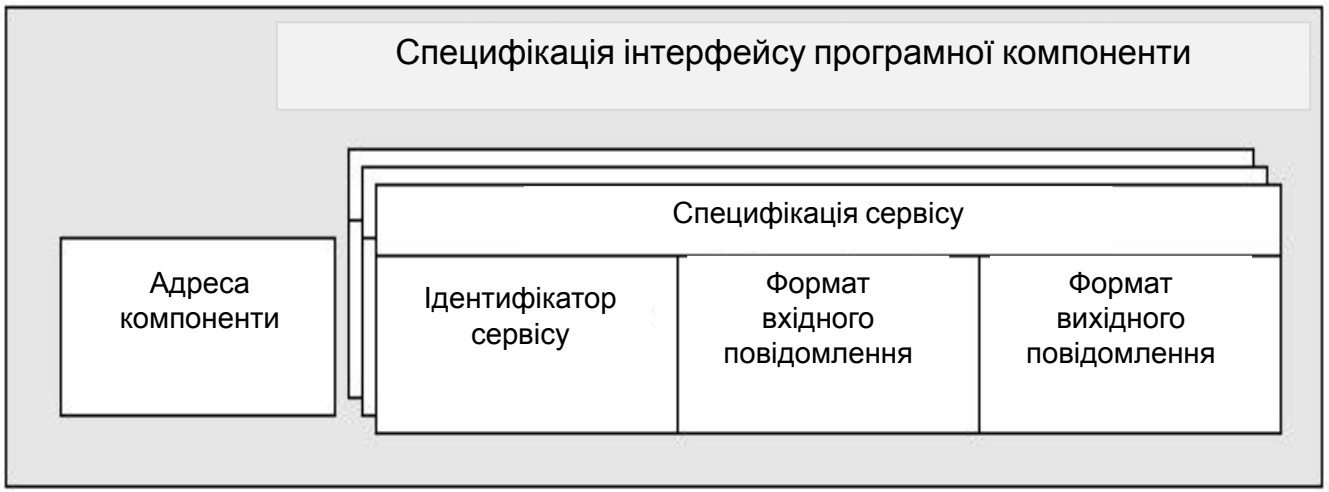
- Flat Files (однорідний, плоский файл)
- Staging area (проміжний вузол)
- Raw data (сирі, необроблені дані)
- Data Marts (вітрини даних)
- Inventory (обладнання)
- Warehouse (сховище даних)
- Mining (“добування” даних)

ПРОГРАМНІ КОМПОНЕНТИ РОЗПОДІЛЕНИХ СИСТЕМ

- Програмна компонента РС – це одиниця програмного забезпечення, що виконується на одному комп'ютері в межах одного процесу і надає деякий набір сервісів, які використовуються через її зовнішній інтерфейс іншими компонентами, які виконуються на цьому ж комп'ютері та на віддалених комп'ютерах

ПРОГРАМНІ КОМПОНЕНТИ РОЗПОДІЛЕНИХ СИСТЕМ

- Кожний сервіс програмної компоненти характеризується трьома сутностями:
  - повною адресою сервісу;
  - єдиною специфікацією прийнятих сервісом повідомлень (запитів);
  - єдиною специфікацією прийнятих від сервісу повідомлень (відповідей).



ВЗАЄМОДІЯ КОМПОНЕНТ РОЗПОДІЛЕНОЇ СИСТЕМИ

Обмін повідомленнями

- Безпосередній обмін
- Черги повідомлень

Віддалений виклик процедур

- Синхронний виклик
- Односпрямований асинхронний виклик
- Асинхронний виклик

ОБМІН ПОВІДОМЛЕННЯМИ

- Безпосередній обмін

– передача відбувається прямо, і вона можлива тільки в тому випадку, якщо приймаюча сторона готова прийняти повідомлення в цей же момент часу.

- Черги повідомлень

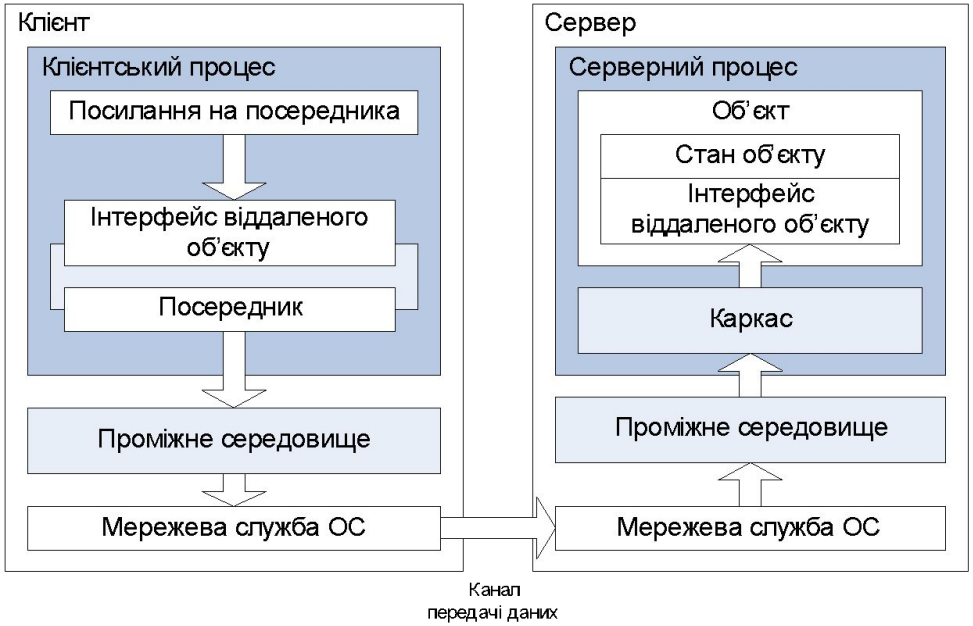
– використовується посередник – менеджер черг повідомлень. Компонента посилає повідомлення в одну із черг менеджера, після чого вона може продовжити свою роботу. Надалі сторона, яка одержує повідомлення, вилучить повідомлення із черги менеджера й приступить до його обробки.

ВІДДАЛЕНИЙ ВИКЛИК ПРОЦЕДУР

- Синхронний виклик
  - клієнт очікує завершення процедури сервером і при необхідності одержує від нього результат виконання віддаленої функції.
- Односпрямований асинхронний виклик
  - клієнт продовжує виконання свого завдання, одержання відповіді сервера або відсутнє, або його реалізація якось інакше передбачена при розробці
- Асинхронний виклик
  - клієнт продовжує своє виконання, при завершенні сервером виконання процедури він одержує повідомлення й результат її виконання, наприклад через callback-функцію, що викликається проміжним середовищем при одержанні результату від сервера

ВИКОРИСТАННЯ ВІДДАЛЕНИХ ОБ'ЄКТІВ

- Моделі використання віддалених об'єктів:
  - єдиного виклику (*singlecall*)
  - єдиного екземпляра (*singleton*)
  - активації об'єктів по запиту клієнта (*client activation*)



МОДЕЛІ ВИКОРИСТАННЯ ВІДДАЛЕНИХ ОБ'ЄКТІВ

- Єдиного виклику (*singlecall*)
  - об'єкт активується на час єдиного віддаленого виклику. В найпростішому випадку для кожного виклику віддаленого методу об'єкта клієнтом на сервері створюється й активується новий екземпляр об'єкта, що деактивується й потім знищується відразу після завершення віддаленого виклику методу об'єкта.
- Єдиного екземпляра (*singleton*)
  - віддалений об'єкт існує не більш ніж в одному екземплярі. Створений об'єкт існує, поки є хоч один клієнт, що використовує його.
- Активації об'єктів по запиту клієнта (client activation)
  - при кожному створенні клієнтом посилання на віддалений об'єкт (точніше, на посередника) на сервері створюється новий об'єкт, що існує, поки клієнт не видалить посилання на посередника.

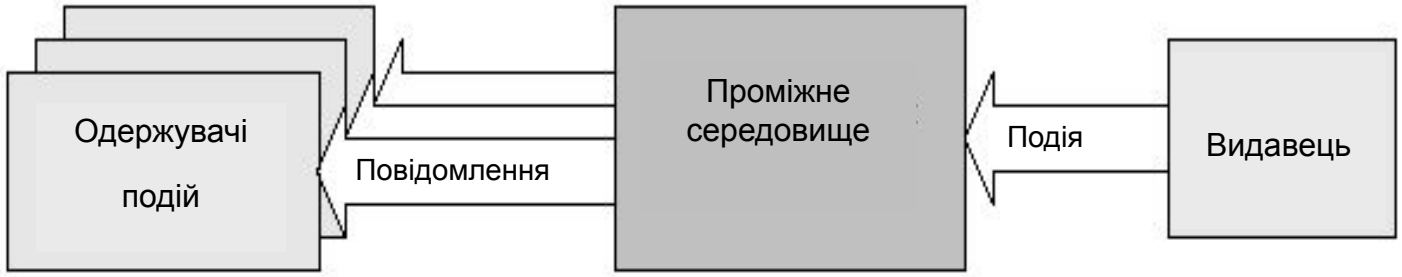
РОЗПОДІЛЕНІ ПОДІЇ

- Тіснозв'язні події

– пряме повідомлення однієї сторони іншою стороною

- Слабкозв’язні події

– джерела події (видавці) не взаємодіють прямо з одержувачами подій (передплатниками).



РОЗПОДІЛЕНІ ТРАНЗАКЦІЇ

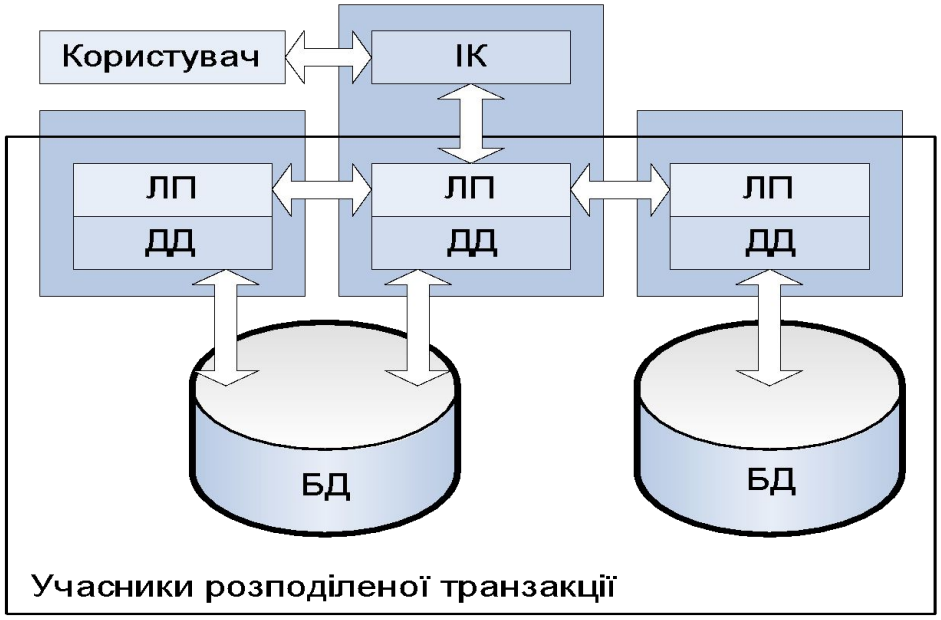
- Транзакція – послідовність операцій з якими-небудь даними, що або успішно виконується повністю, або не виконується взагалі.

РОЗПОДІЛЕНІ ТРАНЗАКЦІЇ

- Транзакція повинна мати наступні якості:
  - Атомарність
    - Транзакція виконується за принципом "все або нічого".
  - Погодженість
    - Після успішного завершення або відкоту транзакції всі дані перебувають у погодженому стані, їхня логічна цілісність не порушена.
  - Ізоляція
    - Для об'єктів поза транзакцією не видні проміжні стани, які можуть приймати дані, що змінюються в транзакції. З погляду "зовнішніх " об'єктів, до успішного завершення транзакції вони повинні мати той же стан, у якому перебували до її початку.
  - Сталість
    - У випадку успішності транзакції зроблені зміни повинні мати постійний характер .

РОЗПОДІЛЕНІ ТРАНЗАКЦІЇ

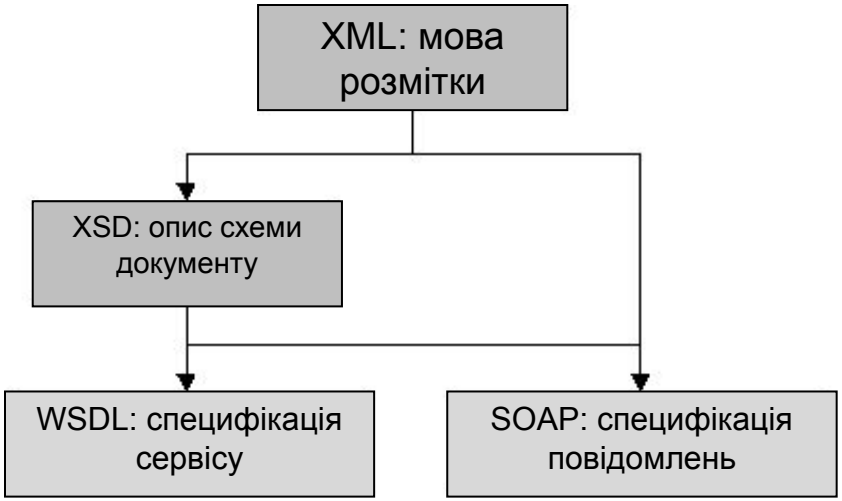
- Розподілена транзакція – це транзакція, що охоплює операції декількох взаємодіючих компонент розподіленої системи.



БЕЗПЕКА В РОЗПОДІЛЕНИХ СИСТЕМАХ

- Проміжне середовище повинне забезпечувати підтримку трьох функцій:
  - Аутентифікація
  - Авторизація
  - Електронний підпис та шифрування повідомлень

ОПИС ІНТЕРФЕЙСУ ПРОГРАМНОЇ КОМПОНЕНТИ



- Мова XML – це мова розмітки текстового документа, представленого сукупністю іменованих, деревоподібних вкладених елементів.
- Кожний елемент може мати деяке текстове значення й набір атрибутів, що мають ім'я й просте значення (рядок).
- Мова XML є абстрактною мовою розмітки, яка не визначає ніякого змісту елементів документу.

МОВА XML

- Оскільки властиве XML відкрите представлення інформації не завжди зручне з погляду безпеки, то існує специфікації XML-DigitalSignature і XML-Encrypton, призначені для передачі в XML конфіденційної інформації. Перша з них дозволяє додати до XML-документу цифровий підпис, інша – зашифрувати XML-документ або окремі його елементи.

- Однією із переваг XML – є наявність мов специфікацій, що визначають правильний XML документ.

ХАРАКТЕРИСТИКИ РОЗПОДІЛЕНИХ СИСТЕМ

1. Просторова розподіленість компонент розподіленої системи.
2. Компоненти розподіленої системи можуть працювати паралельно.
3. Кожний стан компоненти розглядається локально.
4. Компоненти працюють незалежно й можуть «випадати», не руйнуючи систему в цілому, також незалежно одна від одної.
5. Система працює асинхронно. Зміни й процеси синхронізуються.
6. У розподіленій системі функції управління розподіляються між різними автономними компонентами.
7. Розподілена система може утворюватися як об'єднання вже існуючих систем.

ХАРАКТЕРИСТИКИ РОЗПОДІЛЕНИХ СИСТЕМ

8. Програми й дані можуть переміщатися між різними вузлами, ця концепція називається міграцією.
9. Розподілена система повинна бути в змозі використовувати динамічні зміни структури.
10. Архітектура комп'ютерів може використовувати різні топології й механізми, зокрема, якщо апаратура надходить від різних виробників. Ця характеристика називається гетерогенністю.
11. Розподілена система підлягає еволюції, тобто за час її життя відбуваються різні зміни.
12. Джерела відомостей, одиниці обробки й користувачі можуть бути фізично мобільні.

ХАРАКТЕРИСТИКИ РОЗПОДІЛЕНИХ СИСТЕМ

Для досягнення цих характеристик розподілені системи повинні бути виконані у відповідності до певних вимог, яким повинні вони задовольняти:

- прозорість,
- відкритість,
- гнучкість,
- масштабованість,
- стійкість,
- безпека,
- ефективність.

Характеристики розподілених систем

Розподілена система - система, в якій обробка інформації зосереджена не на одній обчислювальній машині, а розподілена між декількома комп'ютерами. При проектуванні розподілених систем, яке має багато спільного з проектуванням ПЗ в загальному, все ж слід враховувати деякі специфічні особливості.

Існує шість основних характеристик розподілених систем.

1) *Спільне використання ресурсів.* Розподілені системи допускають спільне використання як апаратних (жорстких дисків, принтерів), так і програмних (файлів, компіляторів) ресурсів.
2) *Відкритість.* Це можливість розширення системи шляхом додавання нових ресурсів.
3) *Паралельність.* У розподілених системах кілька процесів можуть одночасно виконуватися на різних комп'ютерах в мережі. Ці процеси можуть взаємодіяти під час їх виконання.

Характеристики розподілених систем

4) *Масштабованість.* Під масштабованістю розуміється можливість додавання нових властивостей і методів.
5) *Відмовостійкість.* Наявність декількох комп'ютерів дозволяє дублювання інформації і стійкість до деяких апаратним і програмним помилок. Розподілені системи в разі помилки можуть підтримувати часткову функціональність. Повний збій в роботі системи відбувається тільки при мережевих помилках.
6) *Прозорість.* Користувачам надається повний доступ до ресурсів в системі, в той же час від них прихована інформація про розподіл ресурсів по системі.

Характеристики розподілених систем

Недоліки розподілених систем

1) *Складність.* Набагато важче зрозуміти і оцінити властивості розподілених систем в цілому, їх складніше проектувати, тестувати і обслуговувати. Також продуктивність системи залежить від швидкості роботи мережі, а не окремих процесорів. Перерозподіл ресурсів може суттєво змінити швидкість роботи системи.
2) *Безпека.* Зазвичай доступ до системи можна отримати з кількох різних машин, повідомлення в мережі можуть переглядатися і перехоплюватися. Тому в розподіленої системі набагато важче підтримувати безпеку.

Характеристики розподілених систем

3) *Керованість.* Система може складатися з різнотипних комп'ютерів, на яких можуть бути встановлені різні версії операційних систем. Помилки на одній машині можуть поширитися непередбачуваним чином на інші машини.
4) *Непередбачуваність.* Реакція розподілених систем на деякі події непередбачувана і залежить від повного завантаження системи, її організації та мережевого навантаження. Так як ці параметри можуть постійно змінятся, тому час відповіді на запит може істотно відрізнятися час від часу.

Проблеми розподілених систем

З цих недоліків можна побачити, що при проектуванні розподілених систем виникає ряд проблем, які треба враховувати розробникам.

1) *ідентифікація ресурсів*. Ресурси в розподілених системах розташовуються на різних комп'ютерах, тому систему імен ресурсів слід продумати так, щоб користувачі могли без праці відкривати необхідні їм ресурси і посилатися на них. Прикладом може служити система URL (уніфікований покажчик ресурсів), яка визначає імена Web-сторінок.
2) *комунікація*. Універсальна працездатність Internet і ефективна реалізація протоколів TCP / IP в Internet для більшості розподілених систем є прикладом найбільш ефективного способу організації взаємодії між комп'ютерами. Однак в деяких випадках, коли потрібна особлива продуктивність або надійність, можливе використання спеціалізованих засобів.

Проблеми розподілених систем

3) *якість системного сервісу*. Цей параметр відображає продуктивність, працездатність і надійність. На якість сервісу впливає ряд факторів: розподіл процесів, ресурсів, апаратні засоби і можливості адаптації системи.
4) *архітектура ПЗ*. Архітектура ПЗ описує розподіл системних функцій по компонентах системи, а також розподіл цих компонентів по процесорам. Якщо необхідно підтримувати високу якість системного сервісу, вибір правильної архітектури є вирішальним фактором.

Архітектура розподілених об'єктів

Завдання розробників розподілених систем - спроектувати програмне та апаратне забезпечення так, щоб надати всі необхідні характеристики розподіленої системи. А для цього потрібно знати переваги та недоліки різних архітектур розподілених систем. Виділяється три типи архітектур розподілених систем:

1) *Архітектура клієнт/ сервер з броузером.* У цій моделі систему можна представити як набір сервісів, що надаються серверами клієнтам. У таких системах сервери і клієнти значно відрізняються один від одного.

Архітектура розподілених об'єктів

2) *Триланкова архітектура з “тонким клієнтом”.*

   У цій моделі сервер надає клієнтам послуги не безпосередньо, а за допомогою сервера бізнес-логіки.

3) *Архітектура розподілених об'єктів*.

   У цьому випадку між серверами і клієнтами немає відмінностей і систему можна представити як набір взаємодіючих об'єктів, місце розташування яких не має особливого значення. Між постачальником сервісів і їх користувачами не існує відмінностей.

Архітектура розподілених об'єктів

Ця архітектура широко застосовується в даний час і носить також назву архітектури веб - сервісів. Веб-сервіс - це додаток, доступний через Internet і надає деякі послуги, форма яких не залежить від постачальника (так як використовується універсальний формат даних - XML) і платформи функціонування.

В даний час існує три різні технології об’єктних систем, що підтримують концепцію розподілених. Це технології EJB, CORBA і DCOM.

Технологія SOAP

Основний зміст технології SOAP (Simple Object Access Protocol) полягає в обміні повідомленнями між віддаленими об'єктами за протоколом *HTTP* з використанням XML як транспорту. Специфікація *SOAP* підтримується та розвивається консорціумом *W3C*.

За функціональними можливостями технологія *SOAP* дуже подібна до перших версій *CORBA*. Однак у неї є одна безперечна перевага: простота. На рівні передачі даних у глобальних мережах, між підприємствами, де великої складності взаємодії не передбачається - це оптимальне рішення щодо співвідношення - час розробки/функціональність.

Існують численні мости (*CORBA/SOAP, C++/SOAP, Java/SOAP).*

Технології COM/DCOM

COM (Component Object Model) - це стандарт Microsoft, що визначає структуру та взаємодію компонентів програмного забезпечення в сучасних операційних системах MS Windows. Архітектура сучасних Windows-додатків заснована на COM: світ цих додатків - це світ COM-компонент. Компоненти COM мають унікальність і надають іншим компонентам COM стандартно описані інтерфейси, що дозволяють отримати доступ до методів цих компонентів. COM визначає механізм зв'язку тільки між локальними компонентами.

Технології COM/DCOM

DCOM (Distributed Component Object Model) - це розподілена версія COM, що забезпечує механізм зв'язку між віддаленим COM-компонентами (тобто, що знаходяться на різних комп'ютерах, але в середовищі MS Windows). Фактично DCOM це COM з доданим до останнього механізмом RPC (remote procedure call). Подібну функціональність взаємодії віддалених Windows-додатків можна отримати з використанням активно розвивається останнім часом фірмою Microsoft технології .NET.Важливо підкреслити, що згадані в цьому розділі технології відносяться виключно до операційних систем Microsoft.

Технологія Enterprise Java Beans

Архітектура EJB - це компонентна архітектура, призначена для розробки та розгортання розподілених бізнес-додатків, що базуються на компонентах.

Програми, створені за допомогою архітектури EJB, є масштабованими, орієнтованими на транзакції та безпечними при роботі в розрахованому на багато користувачів режимі.

Технологія EJB (Enterprise JavaBeans) є компонентною моделлю проміжного рівня для розробки розподілених програмних застосувань для мови Java. За допомогою цієї технології розробникові досить програмувати лише бізнес-логіку застосування і можна не займатися стандартними речами, такими, як обробка транзакцій.

Enterprise Java Beans – це стандартна модель серверних компонент для моніторів компонентних транзакцій. Компоненти є Java (J2EE) об'єктами, що реалізують технологію Enterprise Java Beans (EJB).

Технологія Enterprise Java Beans

1) Кожен такий компонент виконується під управлінням сервера додатків, який має відповідати так званої специфікації *EJB*-контейнера, тобто. підтримувати відповідний *API - EJB Container API* (зазвичай сервер додатків у разі називають *EJB* - контейнером).
2) EJB-контейнер надає компонентам (Enterprise Beans) сервіси системного рівня (наприклад, багатопоточність, механізм транзакцій), залишаючись прозорим для розробника додатків.
3) Ці системні сервіси дозволяють розробнику швидко створювати та розгортати *Enterprise Bean*-компоненти: контейнер як би "закриває" від розробника *EJB* усі складнощі системного характеру (наприклад, вже згадані багатопоточність або механізм транзакцій), дозволяючи йому зосередитися виключно на бізнес-логіці програми.

Технологія Enterprise Java Beans

Enterprise Bean - компонент - це об'єкт необхідного класу, описаного мовою програмування Java, розташований на сервері додатків і виконує частину бізнес-логіки програми (цим займається власне код компоненти, здійснює завдання докладання). Наприклад, у додатку контролю інвентарю, Enterprise Bean -компоненти можуть реалізовувати бізнес-логіку програми у методах checkInventoryLevel() та orderProduct(). Викликаючи ці методи, віддалені клієнти можуть отримувати доступ до інвентарних послуг програми.

Технологія Enterprise Java Beans

Існує кілька причин, через які використання Enterprise Bean-компонентів спрощує розробку великих розподілених корпоративних додатків.

1) *Першою причиною* є той факт, що EJB-контейнер надає всі необхідні сервіси системного рівня, дозволяючи розробнику Bean-компонента сконцентруватися на вирішенні бізнес-завдань. Саме EJB-контейнер (а не Bean-розробник) є відповідальним за забезпечення працездатності таких механізмів, як транзакції та авторизація доступу.

Технологія Enterprise Java Beans

2) *Другою перевагою* EJB-компонентів є їхнє розташування на сервері. Внаслідок цього, розробникам клієнтів не доводиться включати бізнес-логіку до складу клієнтської програми - у такому коді не повинно бути функціональності, що реалізує бізнес-правила або доступ до баз даних. У результаті клієнтська програма виходить набагато меншого розміру, що дуже важливо для виконання на пристроях з обмеженими ресурсами.
3) *Третьою перевагою* Enterprise Bean-компонент є їхня переносимість, збирач додатків може збирати нові програми з вже існуючих Bean-компонент. Такі програми можуть бути запущені на будь-якому J2EE-сумісному сервері.

Технологія Enterprise Java Beans

- Основна ідея, що лежала в розробці технології Enterprise JavaBeans, - створити таку інфраструктуру для компонентів, щоб вони могли легко "вставляться"' (`"plug in") і видалятися з серверів, тим самим збільшуючи або зменшуючи функціональність сервера.
- Кожен компонент в середовищі EJB повинен функціонувати усередині контейнера, що ізолює його від ОС сервера.
- Контейнер автоматично виділяє компоненту потік процесів і управляє від його імені службами підтримки паралелізму, захисту, довготривалого зберігання, транзакційної обробки і іншими службами, що надаються застосуванням з боку серверного середовища.

Технологія Enterprise Java Beans

- До її конкурентів можна віднести RMI (remote method invocation - виклик віддалених методів), CORBA (common object request broker architecture), DCOM (distributed component object model) і деякі інші. Треба сказати, що багато принципів CORBA було свідомо покладено в основу EJB. Більш того, існує стандарт відображення EJB на CORBA, що стосується служби імен, управління транзакціями і безпекою.

Технологія Enterprise Java Beans

Базова структура EJB складається з п'яти частин:

- сервери EJB;
- контейнери EJB, які працюють усередині сервера;
- об'єкти типа home, remote, EJBObjects і EJB, які працюють в контейнерах;
- клієнти EJB;
- допоміжні служби, наприклад, JNDI, JTS, служби безпеки, тощо.

Сервіси (служб) серверного середовища EJB

1. **Протокол відаленого зв'язку**. Java Beans має доступ до протоколу віддаленого виклику методів (Remote Method Invocation, RMI), що входить до складу Java і TCP/IP, що працює безпосередньо поверх мережевих протоколів.
2. **Служба каталогів**. Ця служба забезпечується через незалежний від реалізації API-інтерфейс Java Naming & Directory Interface (JNDI) компанії Sun Microsystems, що дозволяє застосуванням, написаним на мові Java, користуватися наявними службами каталогів, наприклад, DNS.

Сервіси (служб) серверного середовища EJB

3. **Служба безпеки.** Платформа EJB може використовувати всі сервіси захисту, що надаються стандартним пакетом Java-security від Sun Microsystems. До них відносяться: аутентифікація із застосуванням відкритого і секретного ключів, шифрування, управління цифровими ключами і списками контролю доступу.
4. **Служба системного адміністрування.** Ця служба забезпечується через API-интерфейс Java Management API (JMAPI) компанії Sun Microsystems. Він надає набір служб моніторингу, управління і адміністрування, а також опис призначеного для користувача інтерфейсу консолі системного адміністратора.
5. **Служба транзакцій.** У середовищі EJB визначена модель обробки транзакцій, в основу якої покладена специфікація Object Transaction Service, розроблена Object Management Group (OMG) (API-інтерфейс Java Transaction Service, JTS, в цьому середовищі не використовується).

Технологія Enterprise Java Beans

У моделі Java Beans передбачені засоби, що дозволяють *упакувати компоненти Java Beans для вкладення в контейнери*, що підтримують модель DCOM, у тому числі в контейнери для MS Visual Basic, MS Internet Explorer, MS Office, Lotus Notes. Для цього служить спеціальний комунікаційний міст, технологічною основою якого служить ***утиліта упаковки.***

Ця утиліта для вибраних компонентів JavaBeans генерує бібліотеку OLE Type Library і реєстрову інформацію для інтерфейсу Win32 ОС Windows. Отримані в результаті дані дозволяють контейнерам DCOM правильно аналізувати, представляти і обробляти компоненти Java Beans,

Технологія Enterprise Java Beans

Контейнер EJB - це абстракція, яка управляє одним або більше класами EJB, в той же час роблячи необхідні служби доступними класам EJB через стандартні інтерфейси, як вказано в специфікації EJB.

Розробник контейнера також може надати додатковий сервіс, що виконується як на рівні контейнера, так і на рівні сервера.

Справжній об'єкт EJB міститься в контейнері EJB і ніколи не має бути безпосередньо доступний нікому, окрім контейнера.

Контейнер EJB повинен служити сполучною ланкою між всіма зверненнями до EJB.

Технологія Enterprise Java Beans

Технологія EJB вимагає розподіленої системи управління транзакціями, яка підтримує протоколи двохфазової фіксації розподілених транзакцій. Специфікація Enterprise JavaBeans передбачає, але не вимагає здійснення транзакцій, заснованих на інтерфейсі Java Transaction Service (JTS) API.

JTS є Java-технологією, пов'язаною з механізмом CORBA Object Transaction Service (OTS). Ця технологія підтримує розподілені транзакції, які можуть охоплювати безліч баз даних на багаточисельних системах, що координуються багатьма менеджерами транзакцій.

Завдяки своїй легко використовуваній Java-моделі, EJB є найпростішим і найшвидшим способом створення кросплатформних розподілених систем.

Переваги і недоліки EJB

Переваги

- Швидке і просте створення
- Java-оптимізація
- Кросплатформність
- Динамічне завантаження компонент-перехідників
- Можливість передачі об'єктів за значенням
- Вбудована безпека

Недоліки

- Підтримка лише однієї мови - Java
- Складність інтеграції з існуючими застосуваннями
- Погана масштабованість
- Продуктивність
- Відсутність міжнародної стандартизації

Виклик віддалених методів - RMI

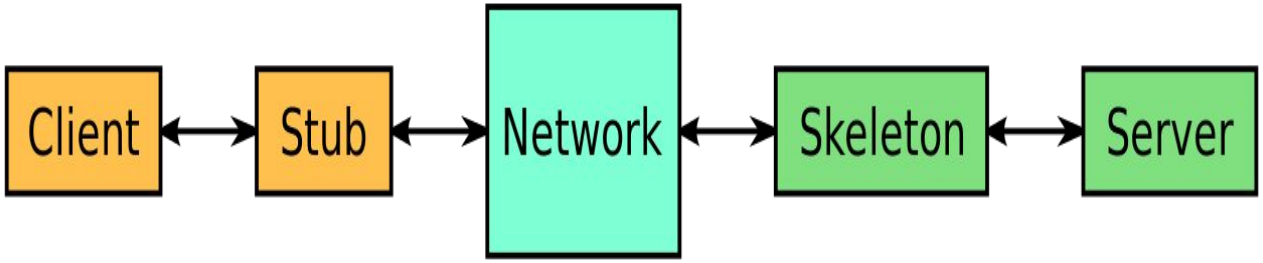
- **RMI** (*Remote Method Invocation*) — програмний інтерфейс викликів віддалених методів в мові Java.
- Розподілена об'єктна модель, що специфікує, яким чином здійснюється виклик віддалених методів, що працюють на іншій віртуальній машині Java.
- При доступі до об'єкту на іншому комп'ютері можна викликати методи цього об'єкта. Необхідно лише передати параметри методу на інший комп'ютер, повідомити об'єкт про необхідність виконання методу, а потім отримати назад значення, що повертається. Механізм RMI дає можливість організувати виконання цих операцій.

Виклик віддалених методів - RMI

В RMI комп'ютери виступають у ролі клієнта та сервера лише для конкретного виклику. Цілком можливо, що при виконанні наступної операції ці комп'ютери поміняються ролями, тобто сервер попереднього виклику може стати клієнтом при зверненні до об'єкта на іншому комп'ютері.

Виклик віддалених методів - RMI

- Типова реалізація моделі Java-RMI, що використовує об'єкти 'заглушки'(stub) та 'скелету'(skeleton).



Виклик віддалених методів - RMI

- При виклику методу віддаленого об'єкта насправді викликається звичайний метод Java, інкапсульований у спеціальному об'єкті-заглушці (stub), який є представником серверного об'єкта.
- Заглушка знаходиться на комп'ютері клієнта, а не на сервері. Вона упаковує параметри віддаленого методу в блок байтів. Кожен параметр кодується за допомогою алгоритму, що забезпечує незалежність від апаратури.

Виклик віддалених методів - RMI

- При цьому об'єкти піддаються серіалізації.
- Процес кодування параметрів називається розгортанням параметрів (parameter marshaling). Основна мета розгортання параметрів - перетворення їх у формат, придатний для передачі параметрів від однієї віртуальної машини до іншої.

Бездротові сенсорні мережі

Бездротові сенсорні мережі

- Простір, який покривається сенсорною мережею, називають сенсорним полем.
- Самоорганізованою мережею називається мережа, число вузлів в якій є випадковою величиною в кожен момент часу, змінюючись в межах від 0 до деякого значення Nmax.
- Бездротові сенсорні вузли це мініатюрні пристрої з обмеженими ресурсами: зарядом батареї, об'ємом пам'яті, обчислювальними можливостями і т.п.
- Зв'язки, що виникають між вузлами в таких мережах, також випадкові у часі тому утворюються для досягнення будь-якої мети або передачі даних в іншу мережу зв'язку
- Кожна мережа має одного координатора, який здійснює глобальну координацію, організацію та установку параметрів мережі

Бездротові сенсорні мережі

Переваги бездротових сенсорних мереж

- здатність до самовідновлення та самоорганізації;
- здатність передавати інформацію на значні відстані, від декількох метрів до декількох кілометрів, при малій потужності передавачів (шляхом ретрансляції повідомлень);
- низька вартість вузлів і їх мініатюрний розмір;
- низьке енергоспоживання і можливість електроживлення від автономних джерел;
- простота установки, відсутність необхідності в прокладанні кабелів;
- можливість установки таких мереж на вже існуючий об'єкт без проведення додаткових робіт;
- низька вартість технічного обслуговування.

Бездротові сенсорні мережі

Компоненти бездротової сенсорної мережі

- **Сенсори**-джерела даних в сенсорної мережі;
- **Актуатори** – виконавчі пристрої, що реагують на сигнал управління для зміни стану керованого об'єкта
- **Сток** - спеціалізована компонента мережі, яка отримує дані від сенсорів і займається їх агрегуванням та є накопичувачем даних, а також оновленням таблиць вузлів;
- **Шлюз** – для управління сенсорною мережею.
- **Базова станція** – приймає данні від вузлів мережі. Це пристрій, обладнаний більшою кількістю ресурсів та більшим радіодіапазоном
- **Вузли передачі даних** - отримують дані від датчиків і відправляють їх в централізовану систему для подальшої обробки

Бездротові сенсорні мережі

- Існують готові інтелектуальні системи на базі бездротових сенсорних мереж
- Управління сенсорною мережею здійснюється за допомогою шлюзу, що може бути встановлений між ключовими компонентами мережі
- Протоколи маршрутизації реалізують механізми визначення отимального напрямку передачі даних з одного пункту в інший
- Обчислювально-комунікаційні вузли в БСМ обмінюються інформацією про топологію мережі

Бездротові сенсорні мережі

Рисунок 1.1 Класифікація сенсорних мереж

Бездротові сенсорні мережі

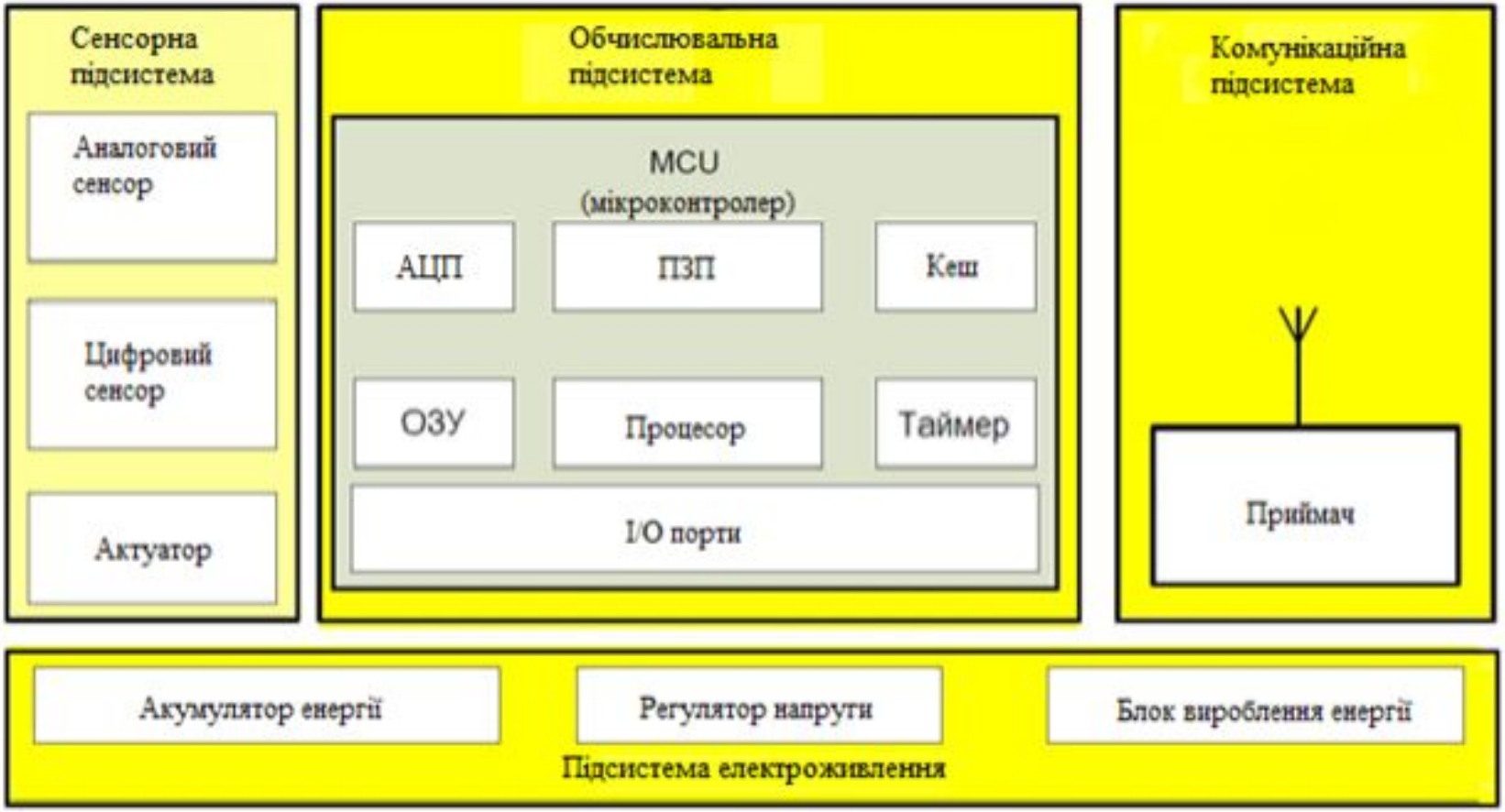


Рисунок 2.3 Вузол безпровідної сенсорної мережі

Бездротові сенсорні мережі

Обмеження для вузлів БСМ:

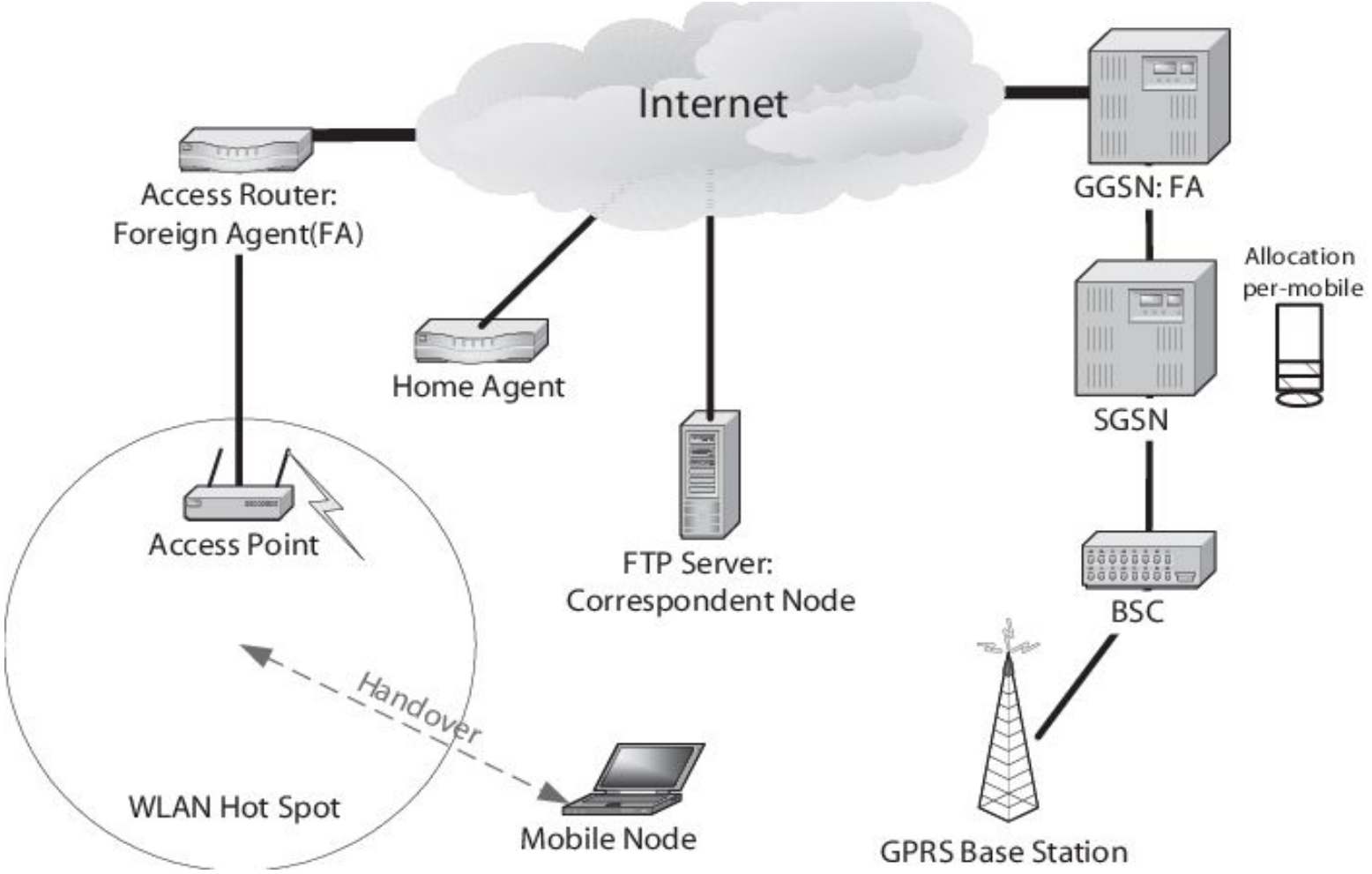
- повинні споживати мало енергії;
- працювати з великою кількістю вузлів;
- мати низьку вартість виробництва;
- бути автономними і працювати без обслуговування;
- адаптуватися до навколишнього середовища

Бездротові сенсорні мережі

Переваги систем на основі БСМ:

- можливість розташування в важкодоступних місцях, куди складно і дорого тягнути звичайні провідні рішення;
- оперативність і зручність розгортання і обслуговування системи;
- надійність мережі в цілому - в разі виходу з ладу одного з них, інформація передається через сусідні елементи;
- можливість додавання або виключення будь-якої кількості пристроїв з мережі;
- високий рівень проникнення крізь перешкоди (стіни, стелі) і стійкість до електромагнітних завад (завдяки високій частоті роботи системи - 2,4 ГГц);
- тривалий час роботи без заміни елементів живлення.

Застосування GPRS, бездротового зв’язку та Internet



AggreGate платформа

- *AggreGate* є однією з небагатьох в світі IoT-платформ, які дійсно підтримують *розподілену архітектуру.*

- Це забезпечує необмежену масштабованість для балансування та розподілу всіх операцій серверів AggreGate на різних рівнях.

- Така архітектура може бути основою як для вирішення поточних завдань, так і для забезпечення потреб в майбутньому.

AggreGate платформа

- На відміну від відмов кластеру, сервери AggreGate в розподіленій архітектурі повністю незалежні.

- Кожен сервер має свою власну базу даних, акаунти локальних користувачів і пов'язані з ними дозволу.

- Розподілена архітектура AggreGate надзвичайно гнучка. Технічно вона заснована на формуванні тимчасових зв'язків між серверами і прикріплення частин єдиної моделі даних одних серверів («постачальників») до інших («споживачам»).

AggreGate платформа

- Обслуговування інфраструктури хмарних сервісів вимагає сотні серверів AggreGate, більшість з яких можна об'єднати в дві групи:

- Сервери, що зберігають реєстр користувачів і їх пристроїв, що перенаправляють підключення операторів і пристроїв на сервери нижнього рівня, а також агрегує дані для подальшого аналізу інформації за участю серверів нижнього рівня
- Сервери, які здійснюють моніторинг і управління пристроями, а також отримання, зберігання і обробку даних

AggreGate платформа

Використання *AggreGate* для IoT середовища:

- Завдяки розподіленої інфраструктури *AggreGate* будь-яке рішення може включати в себе безліч серверів різних рівнів. Частина з них може працювати на IoT-шлюзах, збираючи дані, інші - зберігати і обробляти інформацію, а частина, що залишилася - здійснювати високорівневу агрегацію і розподілені обчислення.
- Польове обладнання, таке як сенсори і актуатори, може бути підключено до серверів безпосередньо, через агенти, через шлюзи або за допомогою їх комбінації.

AggreGate платформа

Це приклад заснованої на AggreGate багаторівневої архітектури для комплексної автоматизації великої групи будівель Smart City:

- **рівень 1**: Фізичне обладнання (мережеві маршрутизатори, контролери, промислове обладнання і т.д.)
- **рівень 2**: Сервери керування (сервери моніторингу мережі, сервери контролю доступу, сервери автоматизації будівель та інші)
- **рівень 3**: Центри управління серверами будівель (один сервер на будівлю, який збирає інформацію з усіх серверів другого рівня)
- **рівень 4**: Сервери районів міста (кінцевий пункт призначення для ескалації сповіщень нижчого рівня, моніторинг в реальному часі, інтеграція з Service Desk-системами)
- **рівень 5**: Сервери головного офісу (контроль серверів району, збір і узагальнення звітів, повідомлень)

Будь-який з вищевказаних серверів може являти собою відмовостійкий кластер, що складається з декількох вузлів.

AggreGate платформа

Управління Мультісегментною мережею

AggreGate Network Manager побудований на платформі AggreGate - є типовим прикладом використання розподіленої архітектури. Великі сегментовані мережі корпорацій і операторів зв'язку не можуть контролюватися з єдиного центру через обмеження маршрутизації, політики безпеки або обмежень пропускної здатності каналів зв'язку з віддаленими сегментами мережі.

- Таким чином, розподілена система моніторингу як правило складається з наступних компонентів:

a) **Первинний** або **центральний** сервер, який збирає інформацію з усіх сегментів мережі
b) **Вторинні** сервери або **сервери-зонди**, які виконують опитування пристроїв в ізольованих сегментах
c) **Спеціалізовані** сервери, такі як сервери аналізу трафіку, оброблюють мільярди NetFlow-подій в день

AggreGate платформа

Вторинні і спеціалізовані сервери є постачальниками інформації для основного сервера, надаючи частину своєї моделі даних центру управління. Це може бути:

- Весь зміст дерева контекстів сервера-зонда, що дозволяє повністю контролювати конфігурацію з центрального сервера. У цьому випадку сервер-зонд просто використовується в якості проксі - сервера для подолання проблеми сегментації мережі.
- Попередження, створювані сервером-зондом. У цьому випадку 99% робочих місць можуть бути віддаленими і оператор центрального сервера негайно буде отримувати повідомлення зі вторинних серверів.
- Призначені для користувача набори даних з серверів-зондів, такі як оперативна інформація про стан критично важливих пристроїв або узагальнені звіти. Вся пов'язана з цим робота буде виконана на вторинному сервері, дозволяючи розподілити навантаження.

SOA технології та моделі

- Модель, орієнтована на повідомлення (Message Oriented Model, MOM), зосереджена на повідомлення, їх структуру, способи транспортування і інші компоненти, ніяк не пов'язані з причинами обміну повідомленнями та їх семантикою.
- Модель, орієнтована на сервіси (Service Oriented Model, SOM), зосереджена на діях, які виконуються сервісами.

SOA технології та моделі

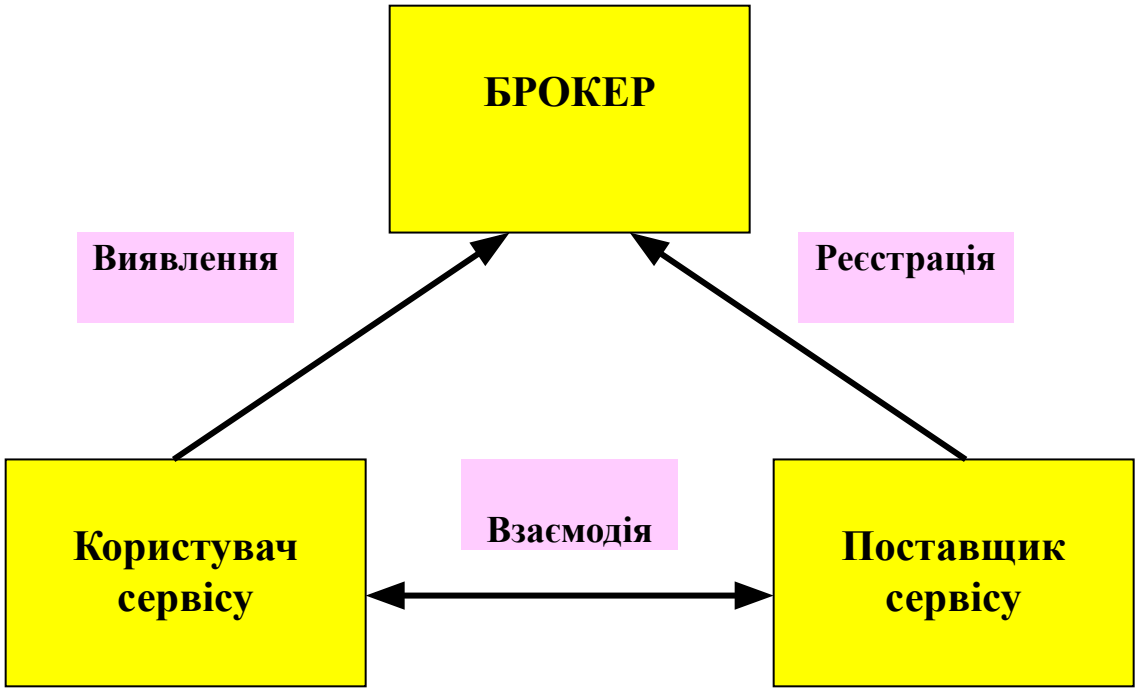
- Модель, орієнтована на ресурси (Resource Oriented Model, ROM), зосереджена *на розподілені системні ресурси* (кластери та суперкомп'ютери)
- Модель політик (Policy Model, PM) визначає політики, пов'язані з архітектурою, в основному вона являє собою обмеження, що накладаються на поведінки агентів і сервісів, в тому числі обмеження, пов'язані з вимогами безпеки, якості обслуговування.
- Модель управління (Management Model, MM).
- Grid – технологія створення *мережі комп'ютерів для паралельної обробки даних* з використанням вільних обчислювальних ресурсів PC. Ґрід є формою розподілених обчислень, у якій багато комп'ютерів об'єднані в один потужний віртуальний комп'ютер, і які працюють разом для виконання трудомістких завдань.

SOA технології та моделі

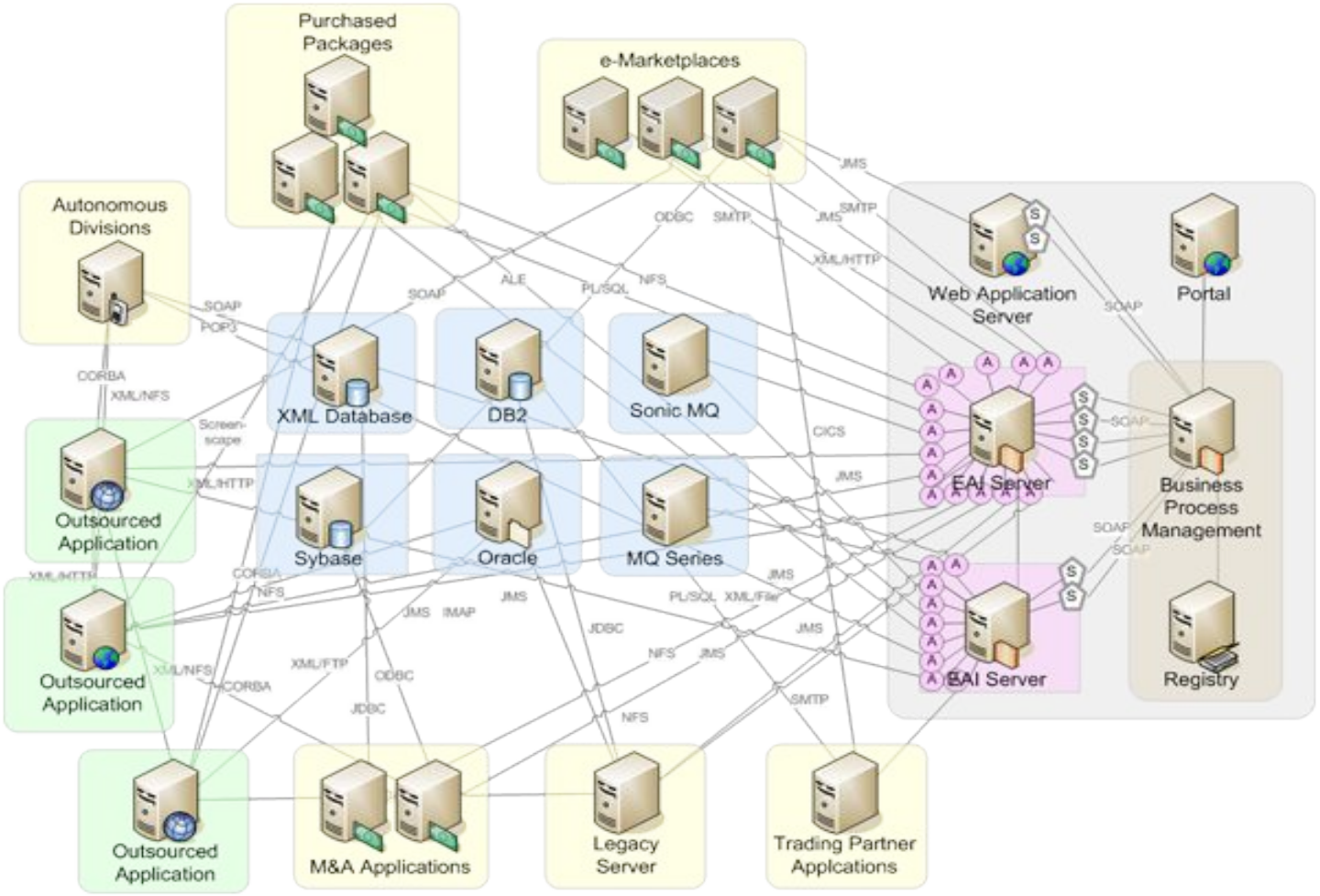
- Для встановлення зв'язків між компонентами і збірки єдиної системи з цих *слабосвязанних і асинхронних компонентів* потрібні зовсім інші прийоми засоби, які отримали досить незвичні назви: «оркестровка» (orchestration) і «хореографія» (choreography).

SOA. Сервісна модель

- Структура з брокером



Архітектура SOA технології



SOA технології та їх реалізації

На сьогодні є два підходи до реалізації технологій SOA:

1. *Перше покоління SOA*, яке базується на композиції веб-сервісів
2. *Друге покоління SOA*, яке базується на поєднанні мікросервісів та контейнерів

SOA технології та їх реалізації

Перше покоління SOA:

- Функціональність SOA простіше всього реалізується за допомогою веб-сервісів (служб).
- Під веб-сервісами розуміються програмні системи, які використовують формат даних *XML* (*Extensible Markup Language*), стандарти *Web Services Description Language* (*WSDL*) для опису своїх інтерфейсів*, Simple Object Access Protocol* (*SOAP*) для опису формату прийнятих повідомлень і повідомлень, що посилаються, та стандарт *Universal Description Discovery and Integration* (*UDDI*) для створення каталогів доступних сервісів.

Перше покоління SOA

- Веб-сервіси є технологічними специфікаціями
- Сервіс-орієнтована архітектура (SOA) є принципом проектування архітектури програмних систем
- Поєднання WSDL, UDDI, SOAP полегшує застосування сервіс-орієнтованих концепцій в Інтернеті, зокрема, для публікування, знаходження, пов'язування і виконання сервісів.

Перше покоління SOA

- SOA почав цю тенденцію за допомогою *технології дистанційного виклику процедур* (*RPC, Remote Procedure Call*) з мережі, зберігаючи при цьому контроль над компонентом доступу.
- Основні підходи до координації веб-сервісів відомі як оркестровка та хореографія

Перше покоління SOA

- Сьогоденним стандартом де-факто серед мов оркестровки є мова *WS-BPEL* (*Web Services Business Process Execution Language*), раніше відома як *BPEL4WS*.
- Для оркестрування робочих потоків (workflow) існує велика кількість інших конкуруючих мов: WSFL, XLANG, PDL, XPDL, ССДО, EDOC, BPML та інші.

Перше покоління SOA

- Для забезпечення комунікації та інтеграції сервісів у великомасштабних гетерогенних прикладних додатках найбільш зручним стало використання сервісної шини підприємства *Enterprise Service Bus* (*ESB*), яка діє в якості проміжного шару (або посередника).
- ESB включає у себе бізнес-процеси, сервіси та події, які *споживаються або виробляються паралельно*.

Перше покоління SOA

- Центральною ланкою SOA є **корпоративна сервісна шина - ESB (Enterprise Service Bus)**, хоча корпоративна сервісна шина не обов'язково повинна використовуватися в разі застосування сервіс-орієнтованої архітектури.

- Основний принцип сервісної шини - концентрація обміну повідомленнями між різними системами через єдину точку, в якій, при необхідності, забезпечується транзакційний контроль, перетворення даних, збереження повідомлень.

- Всі настроювання обробки і передачі повідомлень передбачаються також сконцентрованими в єдиній точці, і формуються в термінах служб, таким чином, при заміні будь-якої інформаційної системи, підключеної до шини, немає необхідності в переналаштуванні інших систем.

Перше покоління SOA

Використання ESB в якості проміжного ПЗ (middleware) забезпечує наступні групи сервісів:

1) *Транспортування*: транспортний сервіс гарантує доставку повідомлень та їх обмін між різними прикладними процесами. ESB може застосувувати динамічну маршрутизацію (також на основі змісту) і відправку запитів *на декількох напрямках*. Це дозволяє ESB балансувати навантаження або механізми відмовостійкості.

Перше покоління SOA

2) *Оброблення подій*: сервіс подій дозволяє ESB обробляти події, можливо, аналізувати та контролювати складні серії взаємопов'язаних подій.
3) *Оркестрування*: сервіс оркестрування заснований на використанні BPEL технологій, дає змогу ESB організовувати виконання ряду взаємозв'язаних сервісів. При роботі з подіями на кожному етапі бізнес-процесу сервіси вищого рівня організують роботу сервісів нижчого рівня.

Перше покоління SOA

4) *Пошуку*: сервіс пошуку, включений у ESB, сприяє тому, що прикладні процеси виявляють відповідні сервіси, з якими вони взаємодіють.
5) *Посередництва:* сервіс посередництва є фундаментальним для галузі бізнесу інтеграції. Сервіс відповідає за

(а) перетворення одного протоколу зв'язку на інший для того, щоб зробити можливим спілкування між гетерогенними середовищами,

(б) перетворення змісту будь-якого повідомлення, а також збагачення повідомлення додатковою інформацією для того, щоб будь-які дані, що передаються, були зрозумілими для будь-якого споживача.

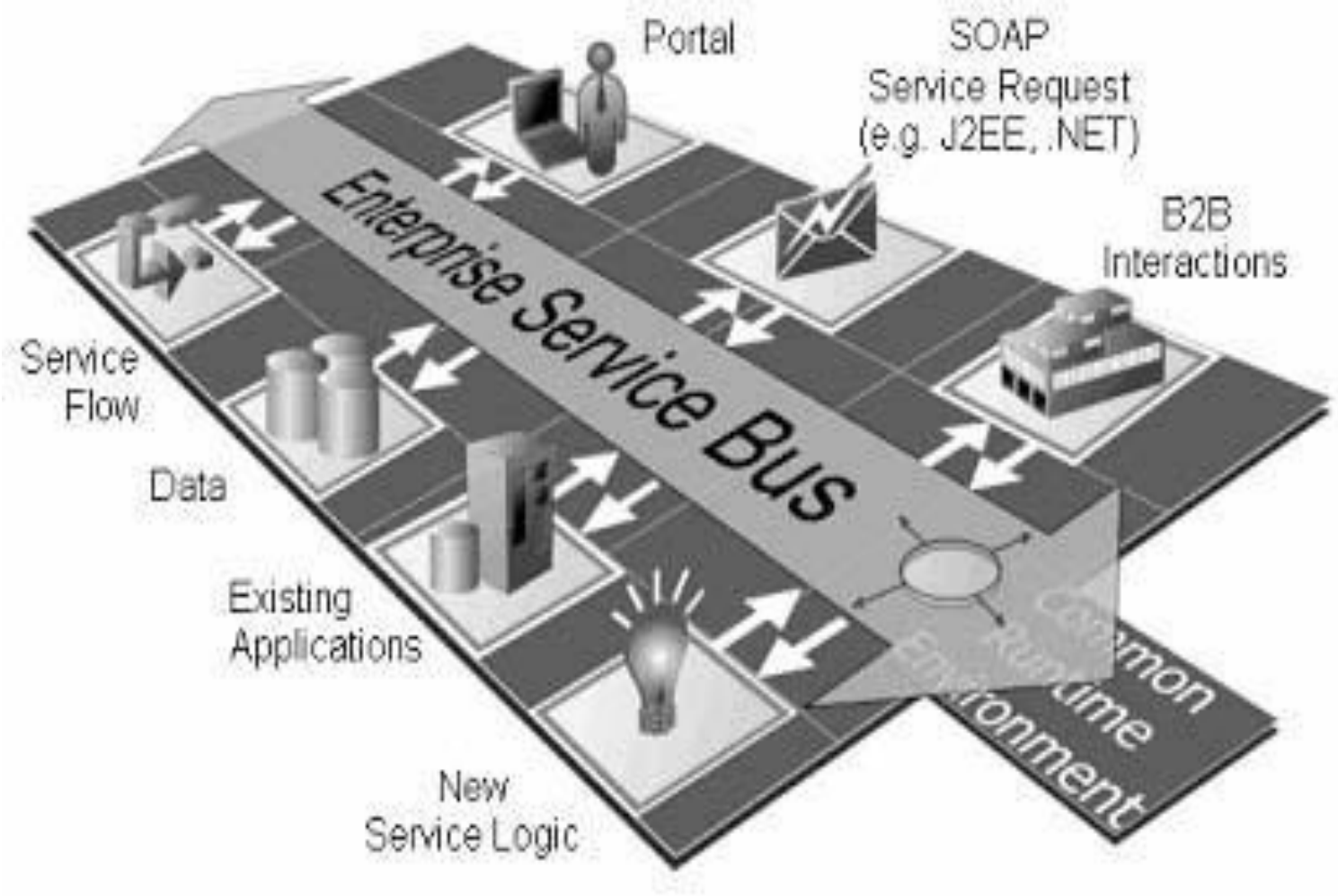
(в) Крім перетворень сервіс посередництва є відповідальним як за безпеку

Перше покоління SOA

ESB можна представити у формі п’ ятирівневої структури:

a) рівень сполучення (адаптери та інтерфейси);
b) транспортна підсистема;
c) рівень реалізації бізнес-логіки;
d) рівень управління бізнес-процесами;
e) рівень бізнес-управління.

SOA. Загальна шина підприємства



Реалізація ESB

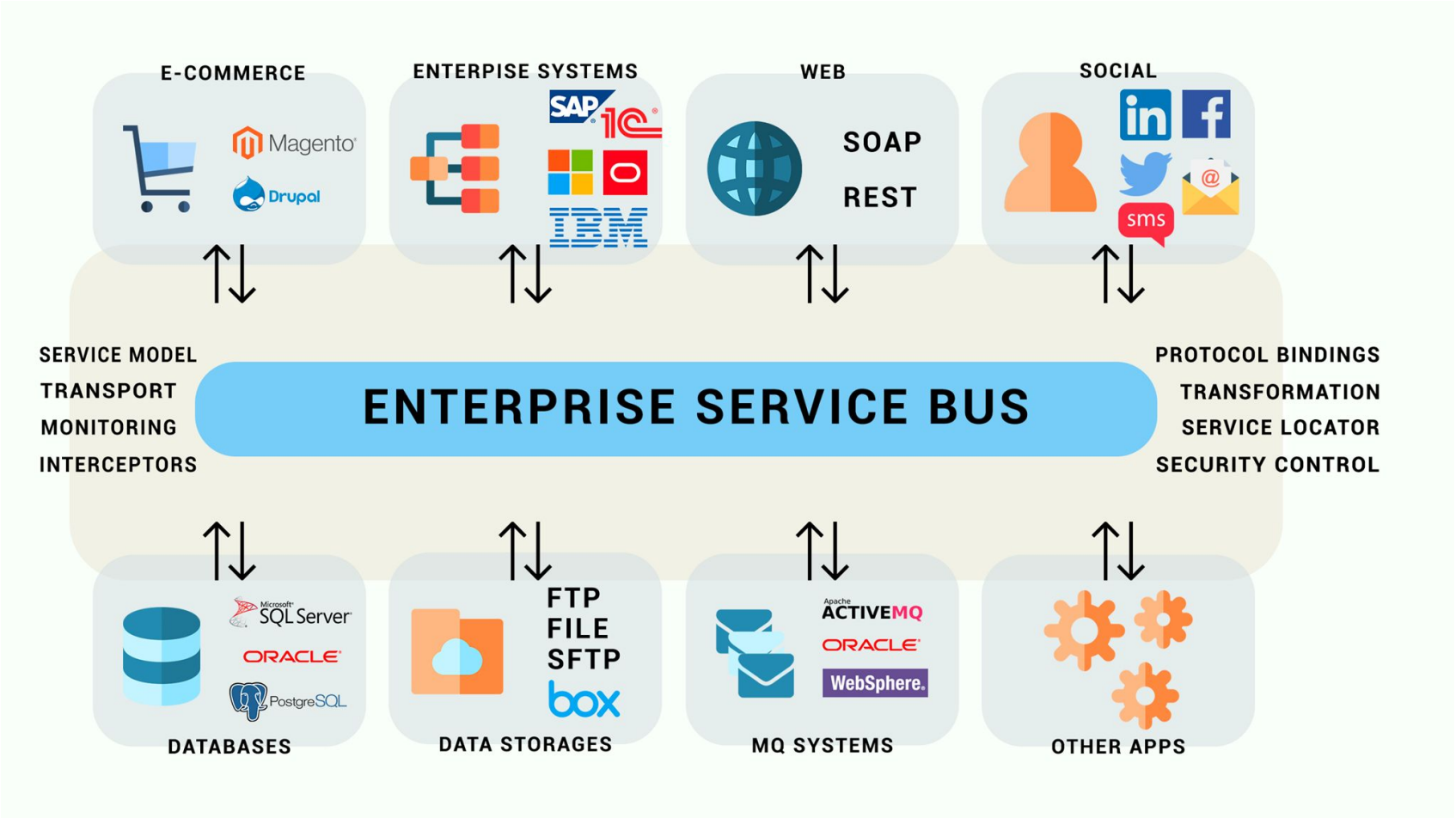
Сучасні тенденції реалізації технології ESB

- Підходи до реалізації ESB за останній час суттєво змінюються. ESB більше не рекомендується в якості центрального, негнучкого хребта для всього підприємства.
- Сьогодні "ESB" більш розглядається *як розподілена інфраструктура*, що масштабується, в якій можна створювати, розгортати і контролювати будь-які сервіси гнучким і ефективним засобом. За його допомогою можна створювати прикладні додатки із сервісів, які реалізують вимоги споживачів і рішення їх бізнес-завдань.
- Розгортання цих сервісів виконується автоматично незалежно один від одного зі стандартизованим інтерфейсом для масштабованої платформи під час виконання. При цьому для оброблення бізнес-сервісів передбачається використовувати ESB в сервісно-орієнтованому вигляді як платформу інтеграції.

SOA. Загальна шина підприємства

- На відміну від мікросервісів - або подібних стратегій, які опосередковують з'єднання API між компонентами - ESB є центром робочого процесу програми. По суті, це черга повідомлень, яка обробляє обмін інформацією в усій програмі.
- Це інструмент, який використовується як у *розподілених обчисленнях*, так і в в інтеграції компонентів.
- ESB — це програмна платформа, яка використовується для розподілу роботи між підключеними компонентами програми.

SOA. Загальна шина підприємства



SOA. Загальна шина підприємства

- Концепція шини передбачає відокремлення додатків один від одного. Зазвичай для цього використовуються сервер обміну повідомленнями типу *Java Message Service* або *Advanced Message Queuing Protocol*
- Дані, що проходять крізь шину, мають стандартизований формат; найчастіше використовується XML
- Між шиною та додатком є адаптер, який «вибудовує» дані в потрібний формат

SOA. Загальна шина підприємства

- Загальною проблемою, пов'язаною з концепцією ESB, є відсутність єдиного прийнятого стандарту для функцій або поведінки.
- ESB також забезпечує балансування навантаження , за допомогою якого можна створити декілька копій компонента для підвищення продуктивності, а також підтримку відмов у разі відмови компонента або ресурсу.

Друге покоління SOA

2. **Друге покоління SOA** яке базується на поєднанні мікросервісів та контейнерів

- На сьогоднішній день існує кілька способів доставки сервіс-орієнтованих додатків для кінцевих користувачів

a) *традиційно за допомогою е-інфраструктури підприємства,*
b) *за допомогою мобільних пристроїв,*
c) *через хмару.*

Друге покоління SOA

- Додатки, розміщені у хмарі, змінюють усі традиційні правила, на яких будуються SOA першого покоління.
- Щоб максимізувати переваги хмарних технологій, необхідно змінити моделі сервісних додатків і оптимізувати хмару відповідно до цих змін.

Друге покоління SOA

- Мікросервіси виглядають так само, як веб-API в тому, що вони можуть бути доступні за допомогою простих HTTP протоколів запиту і форматів представлення даних JSON (*JavaScript Object Notation*, укр. *запис об'єктів JavaScript*).

Друге покоління SOA

Починаючи з 2017 року, мікросервіси у поєднанні з контейнерами дозволили:

- прискорити розробку сервісних хмарних додатків;
- підвищити ефективність їх розгортання;
- забезпечити переміщення сервісів і їх перезапуск в умовах відмови;
- забезпечити *масштабування сервісів* зі зміною навантаження.

Друге покоління SOA

Проблеми реалізації мікросервісів

- *Однією з проблем*, пов'язаною з традиційним розгортанням мікросервісів у хмарі, є затримка доступу до них. Кожен мікросервіс є реалізацією процедур «запит-відповідь», і якщо до мікросервісів часто звертатися в ході виконання додатку, затримки, які накопичуються, можуть серйозно вплинути на час відгуку і продуктивність роботи користувача.

Друге покоління SOA

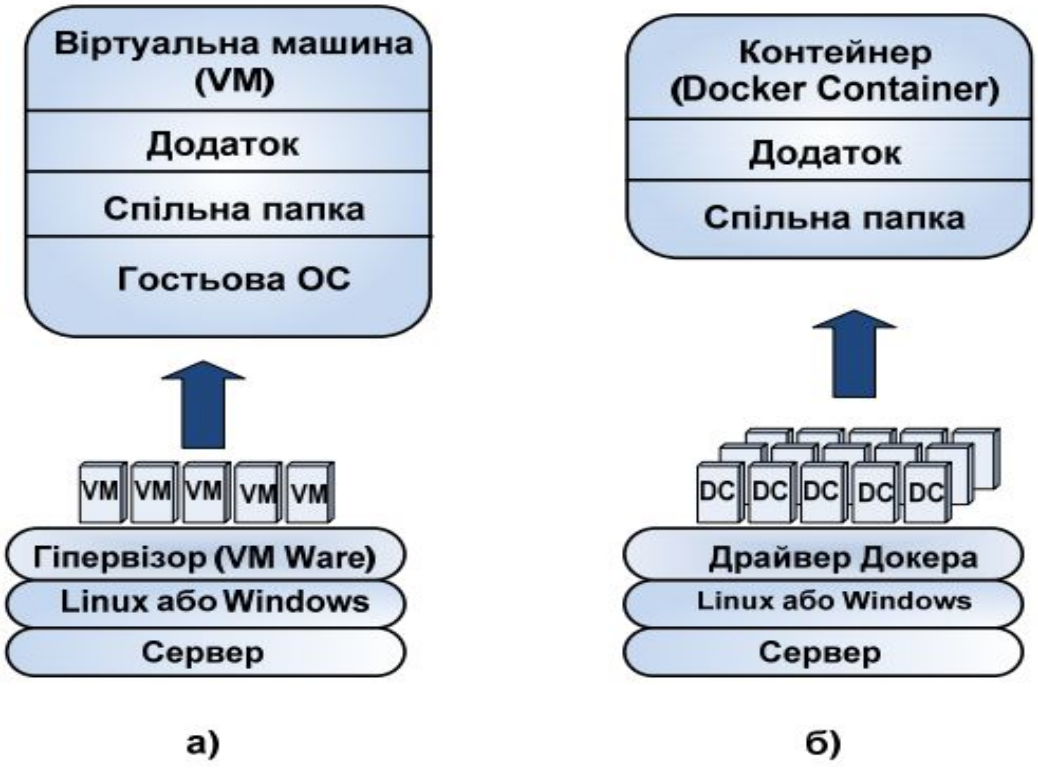
- *Друга проблема* на шляху використання мікросервісів полягає у марній траті ресурсів. Мікросервіси, як правило, мають розміри, набагато менші, ніж традиційні компоненти SOA програмних додатків.

Друге покоління SOA

- Ресурси, що необхідні для розгортання у хмарі контейнера з мікросервісом, можуть скласти біля 90% від ресурсів, що витрачаються на підтримку віртуальної машини (ВМ),
- Контейнери відрізняються від віртуальних машин у тому, що додатки, які працюють в контейнерах, розділяють спільно операційну систему і велику частину проміжного шару.

Друге покоління SOA

- Порівняння процесів віртуалізації (а) і контейнеризації (б)



Друге покоління SOA

- Усунення дублювання таких великих програмних елементів, як гостьова ОС (Gest OS), дозволяє набагато більшій кількості контейнерів з мікросервісами працювати на одному сервері у порівнянні з кількістю віртуальних машин.
- Ця різниця досягає зазвичай значень від 5 до 10 разів, хоча є повідомлення про рекордні значення у 30 разів *паралельності обробки*.

Друге покоління SOA

- Мікросервіси також розгортаються швидше в контейнерах, ніж на віртуальних машинах. Це може бути дуже корисним під час горизонтального масштабування сервісів зі зміною навантаження або при перерозподілі мікросервісів у зв’язку з відмовою мережевого серверу.
- Якщо кожен мікросервіс розмістити в окремій віртуальній машині з незалежною ОС, то буде витрачено багато пам’яті та інших ресурсів.

Друге покоління SOA

- Ефективність використання ресурсів і прискорення розгортання додатку не є єдиною перевагою контейнерів при переході на мікросервіси.
- Хмарні контейнери (наприклад, Docker) здатні природно групувати компоненти додатків на певні *кластери*, розміщуючи, наприклад, спільні програмні елементи (мікросервіси, API) на тому же самому хості, що і компоненти, які їх використовують.

Друге покоління SOA

- Відносини між контейнерами та мікросервісами не є простими. Контейнери не є необхідними для розгортання мікросервісів, але при цьому мікросервіси необхідні для обґрунтування доцільності використання контейнера.
- Мікросервіси є одним із способів підвищення манев-реності додатків і повторного використання коду.
- Контейнери є формою віртуалізації, яка дозволяє уникнути дублювання операційної системи у машинному образі, забезпечуючи загальну платформу, яка поділиться між всіма компонентами додатку.

Друге покоління SOA

- Контейнери пропонують новий спосіб реалізації для SOA, який є більш ефективним у багатьох відношеннях.
- Наприклад, з Docker можна запустити кожен веб-сервіс всередині контейнера. Потім можна використовувати контейнерний оркестратор (скажемо, Kubernetes) для управління зв’язком між контейнерами та гарантувати, що кожний сервіс всередині програми працює стабільно.

Друге покоління SOA

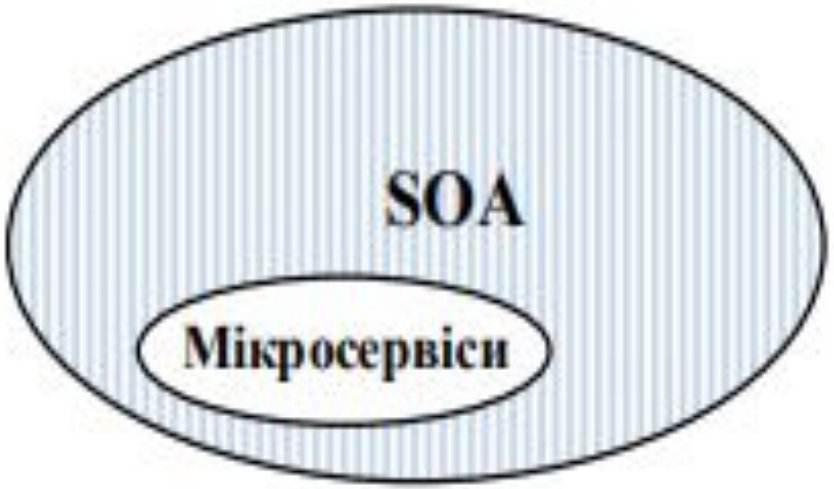
- Контейнери, в свою чергу, можна розгорнути *в кластери*, при цьому контейнери можуть бути попередньо вбудованими як компоненти платформи, що дозволить розмістити їх разом, щоб створити образи додатків.
- Такий підхід відповідає концепції *DevOps* про швидке розгортання нових можливостей програмного забезпечення безпосередньо в операційному середовищі.

Друге покоління SOA

- Ієрархія нових концепцій SOA другого покоління

Друге покоління SOA

- Співвідношення між SOA і MSA



Цілі розподілених систем

Для розподіленої багаторівневої структури з центральним сервером основними цілями є:

1) **масштабованість**. Сервери нижнього рівня можуть бути сильно завантажені, збираючи дані і керуючи великою кількістю пристроїв в режимі, близькому до реального часу. Однак на практиці кількість пристроїв, які можуть обслуговуватися за допомогою одного сервера, обмежена до декількох тисяч. При масштабуванні системи для управління великим числом пристроїв розумно встановити кілька серверів і об'єднати їх в рамках розподіленої системи.
2) **балансування навантаження**. Кожен сервер в розподіленій системі вирішує свою задачу. Сервери управління мережею перевіряють доступність і продуктивність мережевої інфраструктури, сервери контролю доступу обробляють запити від відповідних контролерів. Операції контролю, такі як генерація звітів і їх розсилка поштою, можуть виконуватися на центральному сервері.

Цілі розподілених систем

3) **Захист від вторгнень**. Вторинні сервери-зонди можуть бути встановлені у віддалених місцях і підключені до центрального серверу. Системні оператори підключаються тільки до центрального серверу, при цьому відпадає необхідність в налаштування VPN і “пробросу портів” до цих серверів.
4) **Централізація управління**. Вторинні сервери можуть працювати повністю в автоматичному режимі, В той час як їх налаштування і моніторинг здійснюється через основний сервер, встановлений у центрі моніторингу та управління.

Великомасштабна хмарна IoT-платформа

Постачальники телекомунікаційних і хмарних послуг пропонують IoT-сервіси за моделями *IaaS / PaaS / SaaS*. У цих випадках мова йде про мільйони пристроїв, що належать тисячам користувачів. Обслуговування такої величезної інфраструктури вимагає сотні серверів *AggreGate,* більшість з яких можна об'єднати в такі групи:

- Сервери користувача і пристроїв,що відповідають за взаємодію з хмарної системою управління, яка займається розгортанням нових серверів зберігання даних і аналітики, а також контролює їх роботу.

Великомасштабна хмарна IoT-платформа

- Сервери обробки і зберігання даних, що використовують ресурси (тривоги, моделі, робочі процеси, інструментальні панелі і т.п.), отримані від серверів шаблонів, які в свою чергу зберігають майстер-копії даних ресурсів.
- Маршрутизатори і шлюзи IoT мережі, що мають електронні карти маршрутизації, які можуть змінюватися з часом, для можливого переключення потоків даних, що надходять від сенсорного рівня, на інший сервер при відмові серверу верхнього рівня. Таким чином здійснюється адаптація структури сенсорної мережі до можливих подій і стану її пристроїв.

Багаторівнева інфраструктура Інтернету речей

- Завдяки розподіленій інфраструктурі *AggreGate* будь-яке рішення може включати в себе безліч серверів різних рівнів. Частина з них може працювати на IoT-шлюзах, збираючи дані, інші - зберігати і обробляти інформацію, а частина, що залишилася - здійснювати високорівневу агрегацію і розподілені обчислення.
- Польове обладнання, таке як сенсори і актуатори, може бути підключено до серверів безпосередньо, через агенти, через шлюзи або за допомогою їх комбінації.
- Дані від кібер-фізичних систем, ERP –систем, інформаційних систем Індустріального Інтернету речей (IIoT), інформаційно-управляючих систем Smart Water, Smart Energy, Smart Warm, Smart Healthy поступають на сервера ЦОД та Центрів моніторингу та управління Smart City для аналізу і прийняття рішень.

Принципи створення системи обробки інформації на підприємстві

Поява локальних мереж з часом призвело до розвитку нової області розробки програмного забезпечення - створення розподілених додатків. Займатися цим довелося, що називається, з нуля, але, на щастя, зацікавленість в таких додатках відразу ж виявили великі компанії, структура бізнесу яких вимагала подібних рішень. Саме на етапі створення корпоративних розподілених додатків були сформовані основні вимоги і розроблено основні архітектури подібних систем, що використовуються і в даний час.

Стало ясно, що основними перевагами розподілених додатків є:

1) Хороша масштабованість - при необхідності обчислювальна потужність розподіленого додатка може бути легко збільшена без зміни його структури;
2) Можливість управління навантаженням - проміжні рівні розподіленого додатка дають можливість управляти потоками запитів користувачів і перенаправляти їх менш завантаженим серверам для обробки;
3) Глобальність - розподілена структура дозволяє досягнути просторового розподілу бізнес-процесів і створювати клієнтські робочі місця в найбільш зручних точках.

Принципи створення системи обробки інформації на підприємстві

Основні вимоги до сучасних додатків масштабу підприємства, які диктуються самою організацією виробничого процесу:

- *Просторове розділення.* Підрозділи організації рознесені в просторі і часто мають слабо уніфіковане програмне забезпечення.
- *Структурна відповідність.* Програмне забезпечення повинно адекватно відображати інформаційну структуру підприємства - відповідати основним потокам даних відповідно його бізнес-молям.
- *Орієнтація на зовнішню інформацію.* Сучасні підприємства змушені приділяти підвищену увагу роботі з замовниками, поставщиками та суміжниками. Отже, ПЗ підприємства повинно вміти працювати з новим типом користувачів та їх запитами. Такі користувачі свідомо мають обмежені права і мають доступ тільки до певного виду даних.

Принципи створення системи обробки інформації на підприємстві

- Отже, парадигма розподілених обчислень має на увазі наявність декількох центрів (серверів) зберігання і обробки інформації, що реалізують різні функції і рознесених в просторі. Ці центри крім запитів клієнтів системи повинні виконувати і запити один одного, так як в деяких випадках для вирішення першого завдання можуть знадобитися спільні зусилля кількох серверів.
- Для управління складними запитами і функціонуванням системи в цілому необхідно спеціалізоване керуюче ПЗ. І нарешті, вся система повинна бути "занурена" в якесь транспортне середовище, яке забезпечує взаємодію її частин, що і робить Інтернет технологія.
- Доступ до даних в розподілених додатках можливий з клієнтського ПЗ і з інших розподілених систем може бути організований на різних рівнях - від клієнтського ПЗ і транспортних протоколів до систем захисту серверів БД.

Парадигма розподілених обчислень

Розподілені обчислювальні системи мають такі загальні властивості, як:

1) *Керованість* - має на увазі здатність системи ефективно контролювати свої складові частини. Це досягається завдяки використанню керуючого ПЗ;
2) *Продуктивність* - забезпечується за рахунок можливості перерозподілу навантаження на сервери системи за допомогою керуючого ПЗ;
3) *Масштабованість* - при необхідності фізичного підвищення продуктивності розподілена система може легко інтегрувати в своїй транспортному середовищі нові обчислювальні ресурси;
4) *Розширюваність* - до розподіленим додаткам можна додавати нові складові частини (серверне ПЗ) з новими функціями.

Перераховані властивості розподілених систем є достатньою підставою, щоб миритися зі складністю їх розробки і дорожнечею обслуговування.

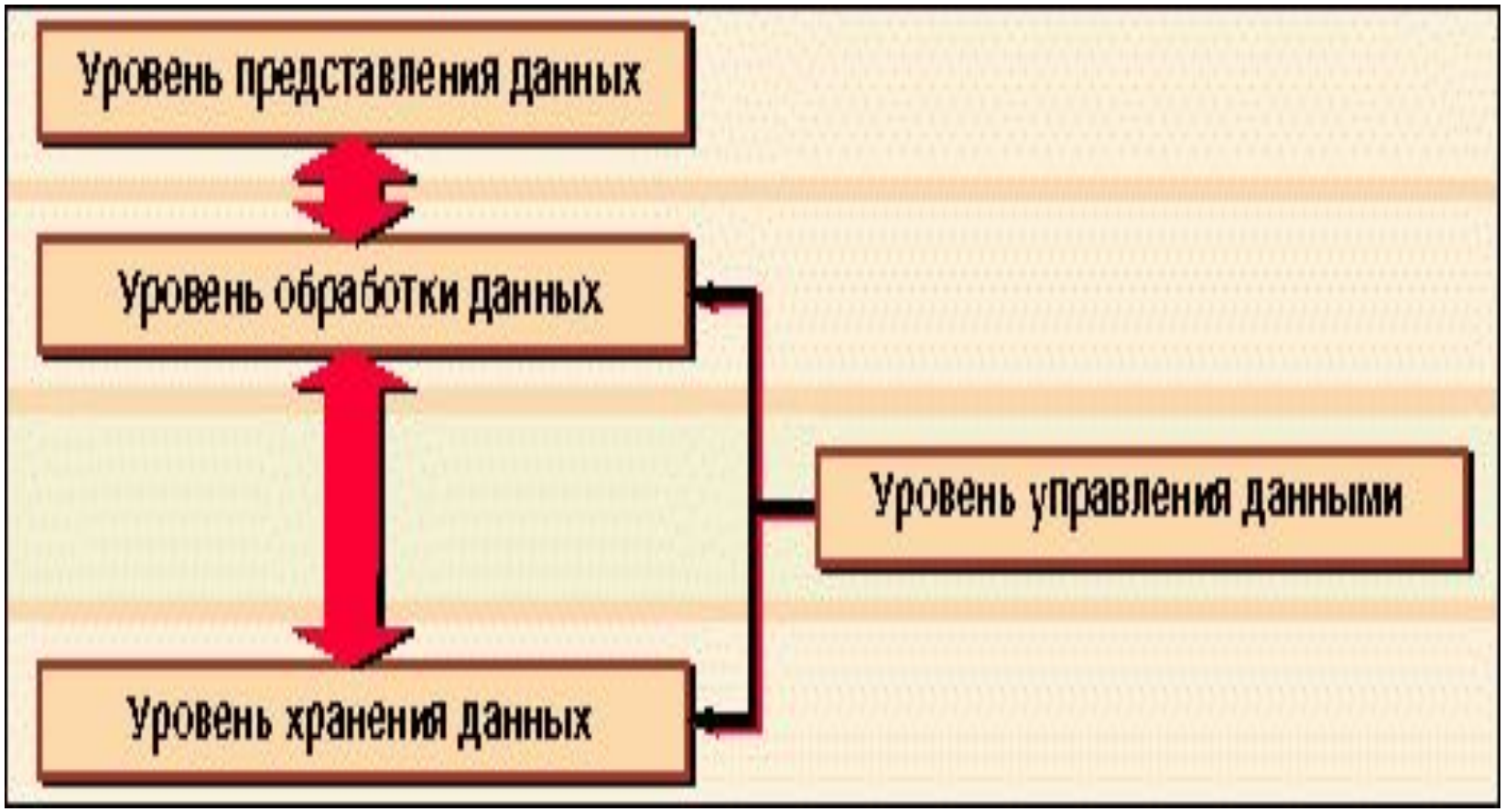
Архітектура розподілених додатків

Існує найбільш загальна архітектура розподіленого додатка, згідно з якою вона розбивається на кілька логічних шарів, рівнів обробки даних, і тут ми можемо виділити три найголовніші їх функції:

1) *Уявлення даних* (інтерфейсний рівень). Тут користувачі програми можуть переглянути необхідні дані, відправити на виконання запит, ввести в систему нові дані або відредагувати їх;
2) *Обробка даних* (проміжний рівень, middleware). На цьому рівні сконцентрована бізнес-логіка програми, здійснюється управління потоками даних і організовується взаємодія частин програми. Саме концентрація всіх функцій обробки даних і управління на одному рівні вважається основною перевагою розподілених додатків;
3) *Зберігання даних* (рівень даних). Це рівень серверів баз даних. Тут розташовані самі сервери, бази даних, засоби доступу до даних, різні допоміжні інструменти.

Розподілений додаток повинен володіти ще одним логічним рівнем - рівнем управління даними.

Основні рівні архітектури розподіленого додатка

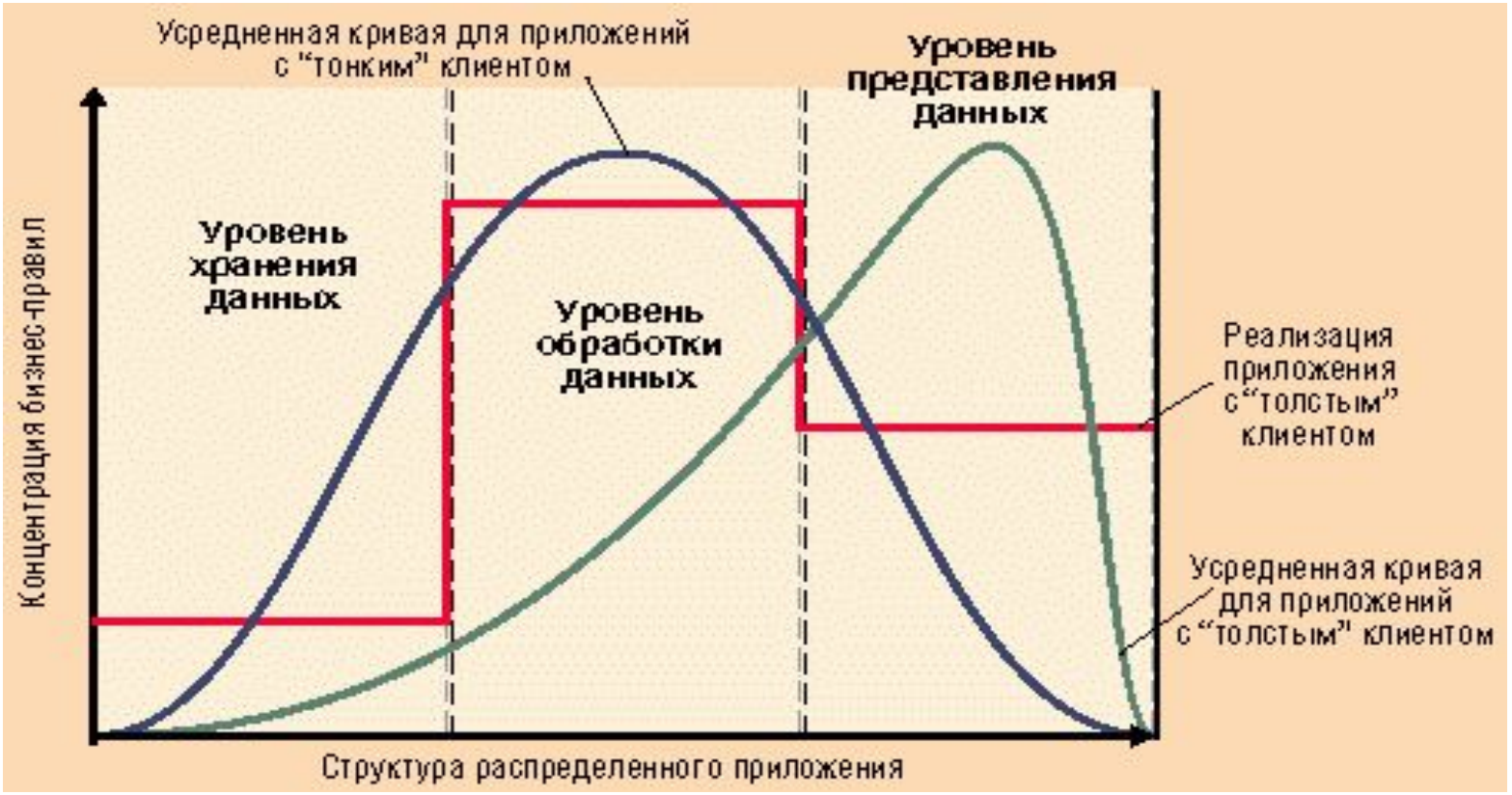


Основні рівні архітектури розподіленого додатка

- Ще один рівень - *управління даними* - стоїть осторонь від магістрального потоку даних, але він забезпечує безперебійне функціонування всієї системи, керуючи запитами і відповідями і взаємодією частин програми.
- Отже, доцільно розділяти проміжний рівень на два самостійних:
  - рівень обробки даних (так як необхідно враховувати важливу перевагу, яку він дає, - концентрацію бізнес-правил обробки даних)

  - рівень управління даними,який забезпечує контроль виконання запитів, обслуговує роботу з потоками даних і організовує взаємодію частин системи.

Розподіл бізнес-логіки за рівнями розподіленого додатка



Розподіл бізнес-логіки за рівнями розподіленого додатка

- Сервери БД можуть не тільки зберігати дані в базах даних, а й містити частину бізнес-логіки додатка в збережених процедурах, тригерах і т. д.
- Клієнтські програми також можуть реалізовувати правила обробки даних. Якщо набір правил мінімальний і зводиться в основному до процедур перевірки коректності введення даних відповідно вбудованих шаблонів, ми маємо справу з “надтонким" клієнтом - браузером. “Тонкий” клієнт має функціональність відповідно встановлених на PC графічних пакетів. "Товстий" клієнт, навпаки, містить велику частку функціональності додатку.
- Рівень же обробки даних власне і призначений для реалізації бізнес-логіки додатка, і тут сконцентровані всі основні правила обробки даних.

Розподіл бізнес-логіки за рівнями розподіленого додатка

- В загальному випадку, функціональність програми виявляється "розмазаною" по всьому додатку. Все розмаїття розподілу бізнес-логіки за рівнями додатків можна представити у вигляді плавної кривої, яка б показала частку правил обробки даних, сконцентрованої в конкретному місці. Криві на рисунку носять якісний характер, але тим не менш дозволяють побачити, як зміни в структурі програми можуть вплинути на розподіл правил.
- Практика підтверджує цей висновок. Адже завжди знайдеться парочка правил, які потрібно реалізувати саме в збережених процедурах сервера БД, і дуже часто буває зручно перенести деякі початкові операції з даними на сторону клієнта - хоча б для того, щоб запобігти обробку некоректних запитів.

Інфраструктура розподілених додатків

- Топологія розподіленої системи має на увазі поділ на кілька серверів баз даних, серверів обробки даних і сукупність локальних і віддалених клієнтів. Всі вони можуть розташовуватися де завгодно: в одній будівлі або на іншому континенті.
- У будь-якому випадку частини розподіленої системи повинні бути з'єднані надійними і захищеними лініями зв'язку.
- Що стосується швидкості передачі даних, то вона в значній мірі залежить від важливості з'єднання між двома частинами системи з точки зору обробки і передачі даних і в меншій мірі від їх віддаленості.

Рівень управління даними

Рівень управління даними потрібен для того, щоб залишити програму єдиним цілим, вона була стійкою і надійною, мала можливості модернізації та масштабування. Він забезпечує виконання системних завдань, без нього частини програми (сервери БД, сервери додатки, проміжне ПЗ, клієнти) не зможуть взаємодіяти один з одним, а зв'язки, порушені при підвищенні навантаження, не можна буде відновити.

- Крім того, на рівні управління даними можуть бути реалізовані різні системні служби додатків. Адже завжди існують загальні для всього програми, які необхідні для роботи всіх рівнів додатків, отже, їх неможливо розташувати на жодному з інших рівнів.
- Наприклад, *служба єдиного часу* забезпечує всі частини програми системними мітками часу, синхронізуючими їх роботу. Уявіть, що розподілений додаток має якийсь сервер, що розсилає клієнтам завдання із зазначенням конкретного терміну їх виконання. При зриві встановленого терміну, завдання повинне реєструватися з обчисленням часу затримки.

Рівень управління даними

- Якщо клієнтські робочі станції розташовані в тій же будівлі, що і сервер, або на сусідній вулиці, - немає проблем, алгоритм обліку простий. Але що робити, якщо клієнти розташовані в інших часових поясах - в інших країнах або взагалі за океаном? У цьому випадку сервер повинен вміти обчислювати різницю з урахуванням часових поясів при відправці завдань і отриманні відповідей, а клієнти будуть зобов'язані додавати до звітів службову інформацію про місцевий час та часовий пояс. Якщо ж до складу розподіленого додатка входить *служба єдиного часу*, то такої проблеми просто не існує.
- Крім служби єдиного часу рівень управління даними може містити служби зберігання загальної інформації (відомості про програму в цілому), формування загальних звітів і т. п.

Рівень управління даними

- До функцій рівня управління даними відносяться:

1) Управління частинами розподіленого додатка;
2) Управління сполуками і каналами зв'язку між частинами програми;
3) Управління потоками даних між клієнтами і серверами і між серверами;
4) Управління навантаженням;
5) Реалізація системних служб додатків.

- Необхідно відзначити, що часто рівень управління даними створюється на основі готових рішень, що поставляються на ринок програмного забезпечення різними виробниками. Якщо розробники вибрали для свого застосування архітектуру CORBA, то в її складі є брокер об'єктних запитів (Object Request Broker, ORB), якщо платформу Windows, - до їхніх послуг різноманітні інструменти: технологія DCOM (розвиток технології Microsoft Transaction Server, MTS), технологія обробки черг повідомлень MSMQ, технологія Microsoft BizTalk та ін.

Розширення базових рівнів

Розглянуті вище рівні архітектури розподіленого додатка є базовими. Вони формують структуру створюваного додатка в цілому, але при цьому, природно, не можуть забезпечити реалізацію усіх програм - предметні області і завдання занадто великі й різноманітні.

Для таких випадків архітектура розподіленого додатка може бути розширена за рахунок додаткових рівнів, які покликані відобразити особливості створюваного додатка.

Серед інших можна виділити два найбільш часто вживаних розширення базових рівнів:

1) ***Рівень бізнес-інтерфейсу*** розташовується між рівнем призначеного для користувача інтерфейсу і рівнем обробки даних. Він приховує від клієнтських додатків деталі структури і реалізації бізнес-правил рівня обробки даних, забезпечуючи абстрагування програмного коду клієнтських додатків від особливостей реалізації логіки додатка.

Розширення базових рівнів

В результаті розробники клієнтських додатків використовують певний набір необхідних функцій - аналог інтерфейсу прикладного програмування (API). Це дозволяє зробити клієнтське ПЗ незалежним від реалізації рівня обробки даних.

Безумовно, при внесенні серйозних змін в систему без глобальних переробок не обійтися, але рівень бізнес-інтерфейсу дозволяє не робити цього без крайньої необхідності.

2) ***Рівень доступу до даних*** розташовується між рівнем зберігання даних і рівнем обробки даних. Він дозволяє зробити структуру програми що не залежить від конкретної технології зберігання даних. У таких випадках програмні об'єкти рівня обробки даних передають запити і отримують відповіді за допомогою засобів обраної технології доступу до даних.

При реалізації програм на платформі Windows найчастіше використовується технологія доступу до даних ADO, так як вона забезпечує універсальний спосіб доступу до найрізноманітніших джерел даних - від серверів SQL до електронних таблиць. Для додатків на платформе.NET служить технологія ADO.NET.

Технологія Грід

- Термін «грід» було введено в обіг Яном Фостером на початку 1998 року публікацією книги «Грід. Нова інфраструктура обчислень»:
- ***Грід*** – це система, яка *координує розподілені ресурси* за допомогою стандартних, відкритих, універсальних протоколів та інтерфейсів для забезпечення нетривіальної якості обслуговування (QoS – Quality of Service).
- Основна ідея, закладена в концепції грід-обчислень, - централізоване віддалене надання ресурсів, необхідних для розв'язання різноманітних обчислювальних завдань.

Технологія Грід

- Ідеологія грід: ми можемо запустити будь-яке завдання з будь-якого комп'ютера або мобільного пристрою на обчислення, ресурси повинні бути автоматично надані на віддалених високопродуктивних серверах.

Технологія Грід

- Основне завдання «грід» - узгоджений розподіл ресурсів та вирішення завдань в умовах динамічних, багатопрофільних віртуальних організацій. Розподіл ресурсів - це не просто обмін файлами, а прямий доступ до комп'ютерів, програмного забезпечення, даних та інших ресурсів, які потрібні для спільного вирішення завдань. Віртуальною організацією (ВО) називають ряд окремих людей чи установ, об'єднаних єдиними правилами колективного доступу до розподілених обчислювальних ресурсів.
- Для організації роботи у рамках ВО необхідні:

- *гнучкі механізми поділу ресурсів;*
- *розвинена система контролю використовуваних ресурсів;*
- *розподілений доступ до різних ресурсів, починаючи від програм, файлів та даних до комп'ютерів, сенсорів і мереж;*
- *різні моделі використання ресурсів (від однокористувальних до розрахованих на багато користувачів, від високопродуктивних до мало витратних), що включають регулювання якості обслуговування, планування, перерозподіл і ведення обліку ресурсів.*

Основні принципи технології Грід

- Застосування технологій побудови РСОІ, що існували на той момент, не дозволяло повною мірою досягти виконання всіх зазначених вимог. Тому було запропоновано альтернативну архітектуру Грід. Дослідження та розробки у співтоваристві Грід призвели до розробки протоколів, сервісів та інструментарію, спрямованого саме на ті проблеми, які виникають при спробі створення ВО, що масштабуються. Ці технології включають:

- *рішення щодо безпеки, що підтримують управління сертифікацією та політиками безпеки, коли обчислення здійснюються кількома організаціями;*
- *протоколи управління ресурсами та сервісами, що підтримують безпечний віддалений доступ до обчислювальних ресурсів та ресурсів даних, а також перерозподіл різних ресурсів;*
- *протоколи запиту інформації та послуги, що забезпечують налаштування та моніторинг стану ресурсів, організацій та сервісів;*
- *сервіси обробки даних, які забезпечують пошук та передачу наборів даних між системами зберігання даних та додатками.*

Рівні архітектури Грід

5 **Прикладний рівень (Applications)** - інструментарій для роботи з Грід і користувацькі додатки.

Колективний рівень (Collective)

4 управління каталогами ресурсів, діагностика, моніторинг.

3 **Ресурсний рівень (Resource)** реалізує протоколи взаємодії з ресурсами РОС та їх управління.

2 **Зв'язуючий рівень (Connectivity)** визначає комунікаційні протоколи та протоколи аутентифікації.

1 **Базовий рівень (Fabric)**

вміщує різні ресурси, такі як комп'ютери, пристрої зберігання, мережі, сенсори та інше.

Базовий рівень

На базовому рівні визначаються служби, які забезпечують безпосередній доступ до ресурсів, використання яких розподілено через протоколи Грід.

- 1. Обчислювальні ресурси надають користувачеві Грід-системи процесорні потужності. Обчислювальними ресурсами може бути як кластери, і окремі робочі станції. Будь-яка обчислювальна система може розглядатися як потенційний обчислювальний ресурс Грід-системи.
- 2. Ресурси пам'яті є простір для зберігання даних. Для доступу до ресурсів пам'яті використовується програмне забезпечення проміжного рівня, що реалізує уніфікований інтерфейс керування та передачі даних.
- 3. Інформаційні ресурси та каталоги є особливим видом ресурсів пам'яті. Вони служать для зберігання та надання метаданих та інформації про інші ресурси Грід-системи.
- 4. Мережевий ресурс є сполучною ланкою між розподіленими ресурсами Грід-системи. Основною характеристикою мережного ресурсу є швидкість передачі.

Рівень зв’язування

- Зв'язуючий рівень визначає комунікаційні протоколи та протоколи аутентифікації, забезпечуючи передачі даних між ресурсами базового рівня. Зв'язуючий рівень грід заснований на стеку протоколів TCP/IP:

1) Інтернет (IP, ICMP);
2) Транспортні протоколи (TCP, UDP);
3) Прикладні протоколи (DNS, OSRF...).

Ресурсний рівень

- *Ресурсний рівень реалізує протоколи, що забезпечують виконання таких функцій:*

1) *узгодження політик безпеки використання ресурсу;*
2) *процедура ініціації ресурсу;*
3) *моніторинг стану ресурсу;*
4) *контроль над ресурсом;*
5) *облік використання ресурсу.*

- *Окремо виділяються 2 типи протоколів ресурсного рівня:*

1) *Інформаційні протоколи - використовуються для отримання інформації про структуру та стан ресурсу.*
2) *Протоколи управління - використовуються для узгодження доступу до роздільних ресурсів, визначаючи вимог і допустимих дій по відношенню до ресурсу (наприклад, підтримка резервування, можливість створення процесів, доступ до даних).*

Колективний рівень

- Колективний рівень відповідає за глобальну інтеграцію різних наборів ресурсів і може включати:

1) служби каталогів;
2) служби спільного виділення, планування та розподілу ресурсів;
3) служби моніторингу та діагностики ресурсів;
4) служби реплікації даних.

Прикладний рівень

- На прикладному рівні розташовуються користувацькі додатки, що виконуються в середовищі. Вони можуть використовувати ресурси, що знаходяться на будь-яких нижніх шарах архітектури Грід.

Стандарти Грід

Ключовий момент в розробці грід додатків – ***стандартизація***

Представлені різноманітні методи реалізації грід-обчислень.

Загальний вибір: **сервісно-орієнтована модель**

2001 - для створення стандарту архітектури Грід додатків була обрана **технологія веб-сервісів**.

WSDL у совокупності зі спеціальними механізмами зв'язування забезпечує м***ожливість динамічного пошуку та компоновки сервісів*** в гетерогенних середовищах.

Широко розповсюджена адаптація механізмів веб-сервісів означає, що інфраструктура, яка побудовпана на базі веб-сервісів, може ***використовувати вже існуючі утіліти і сервіси***.

Стандарти Грід

- Розроблений стандарт архітектури грід отримав назву *OGSA (Open Grid Services Architecture – відкрита архітектура грід-сервісів*).
- *Грід-сервісом називається сервіс, що підтримує надання повної інформації про поточний стан екземпляра сервісу, а також підтримує можливість надійного та безпечного виконання, управління часом життя, розсилки повідомлень про зміну стану екземпляра сервісу, управління політикою доступу до ресурсів, управління сертифікатами доступу та віртуалізації.*

Стандарти Грід

- Грід-сервіс підтримує такі стандартні інтерфейси:

- ***Пошук.*** Грід додаткам необхідні механізми для пошуку доступних сервісів та визначення їх характеристик.
- ***Динамічне створення сервісів.*** Можливість динамічного створення та управління сервісами – це один із базових принципів OGSA, що вимагає наявності сервісів створення нових сервісів.
- ***Управління часом життя.*** Розподілена система має забезпечувати можливість знищення екземпляра грід-сервісу.
- ***Повідомлення.*** Для забезпечення роботи Грід програми набори Грід сервісів повинні мати можливість асинхронно повідомляти один одного про зміни у їхньому стані.

Стандарти Грід

2003 – перша реалізація моделі OGSA – OGSI (Open Grid Service Infrastructure)

Неспроможність існуючих стандартів веб-сервісів забезпечити вимогу до функціональних можливостей грід-сервісів

Неможливість сумісного використання веб-сервісів та грід-сервісів в одному середовищі

Розробка стандартів веб-сервісів ***другого покоління***, зокрема, WSRF

Сьогодні реалізація моделі **OGSA** за допомогою **WSRF та** супутніх стандартів **WS-Addressing і WS-Notification** найбільш розповсюджена в середовищі Грід

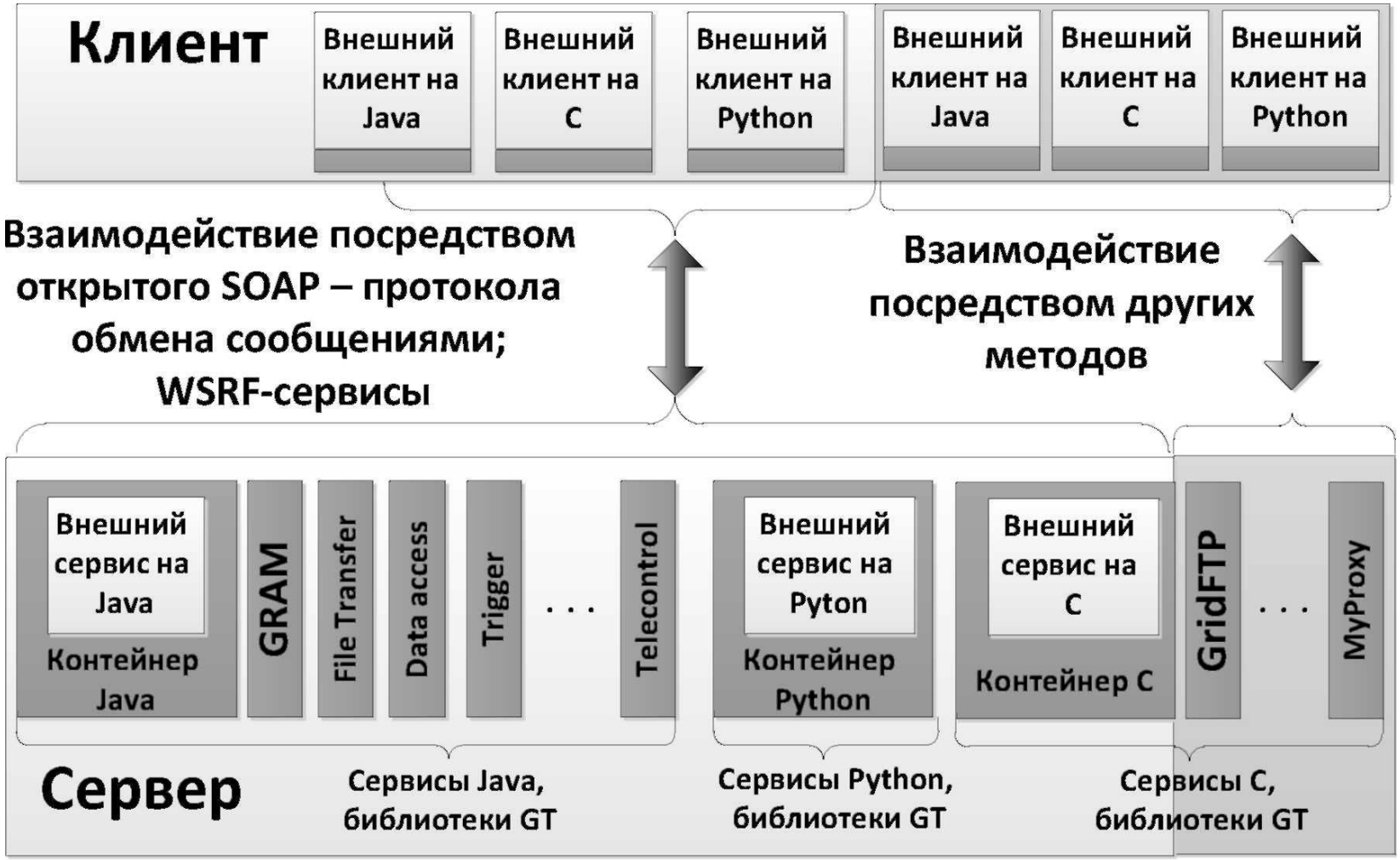
Система Globus

- ***Globus -*** це проект з розробки та надання інфраструктури для грід-обчислень.
- Спочатку Globus був розвитком проекту I-WAY, але в процесі розвитку основний акцент був перенесений з підтримки високопродуктивних обчислень у бік сервісів підтримки віртуальних організацій.
- Мета створення Globus - надання можливості програм працювати з розподіленими різнорідними обчислювальними ресурсами як з єдиною віртуальною машиною.
- Основна спрямованість проекту - обчислювальні грід-системи.

Система Globus

- Під обчислювальною 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 чи ресурсів.
- Базовим елементом системи є *Globus Toolkit* (інструментарій Globus), що описує базові сервіси та можливості, необхідні для створення обчислювальних грід-систем.
- *Система Globus* надає високорівневим додаткам доступ до сервісів, кожен з яких додаток або розробник може використовувати для досягнення власних цілей, тому окремі сервіси мають бути ізольовані та мати чітко визначені програмні інтерфейси.

Загальна схема взаємодії компонентів *Globus Toolkit 4.0*



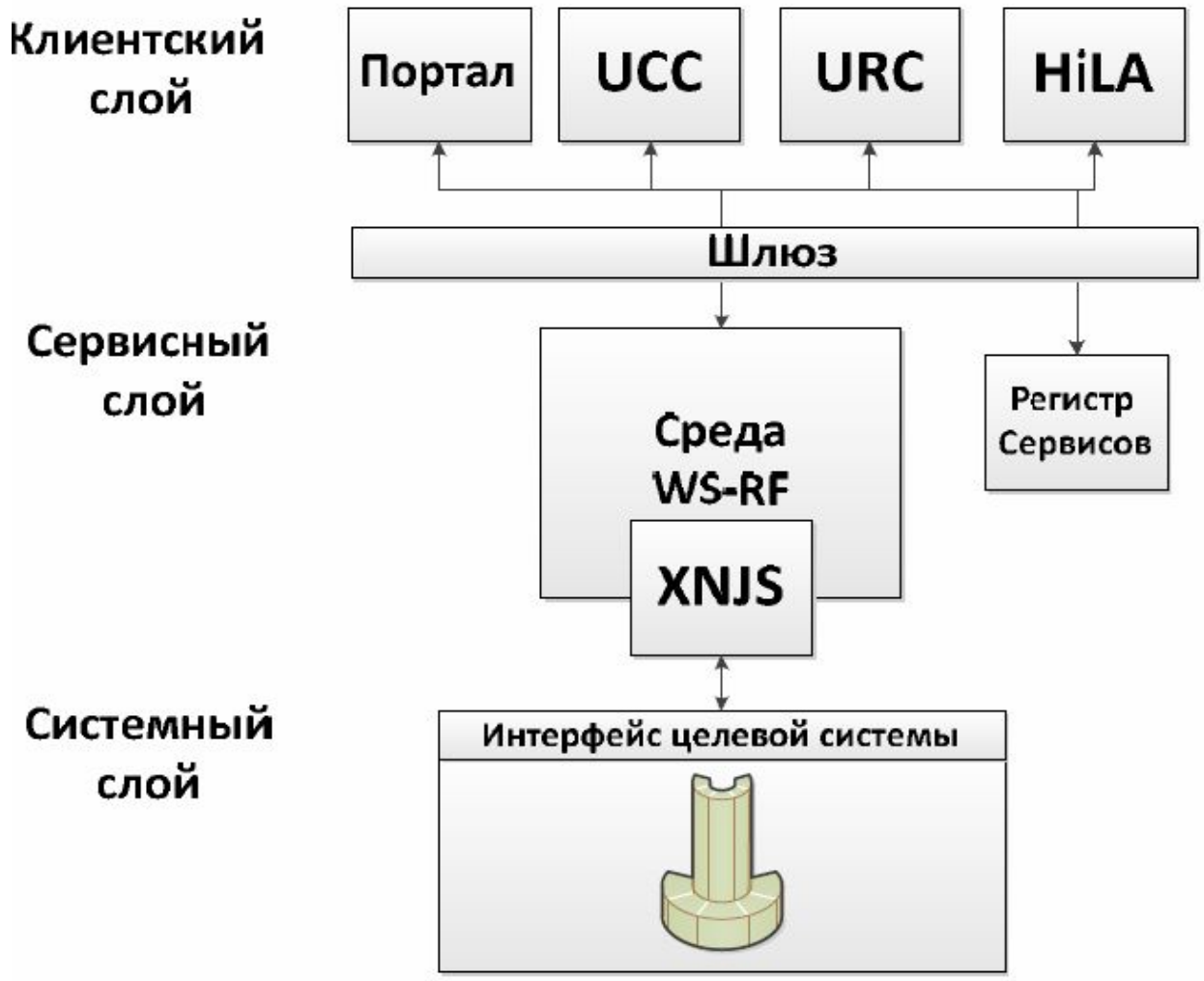
Базові сервіси, які надаються системою Globus

- 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 користувача, що підтримують автентифікацію, управління станом, пошук тощо, а також забезпечують підтримку стандартів *WSRF, WS-Security, WS-Notification.*

Базові сервіси, які надаються системою Globus

- 4. Сервіси аутентифікації 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 *GTK 4.0* та/або користувальницькими сервісами.

Система UNICORE



Проект UNICORE (Uniform Interface to Computing Resources - єдиний інтерфейс до обчислювальних ресурсів) зародився в 1997 році, і зараз є комплексним рішенням, орієнтованим на забезпечення прозорого безпечного доступу до ресурсів грід.

Архітектура UNICORE 6 формується з клієнтського, сервісного та системного шарів.

Система UNICORE

- Верхнім шаром в архітектурі є шар клієнта. В ньому розташовуються різні клієнти, що забезпечують взаємодію користувачів з грід середовищем:

- *UCC (Unicore Command Line Client – клієнт командного рядка для UNICORE)*: клієнт, що забезпечує інтерфейс командного рядка для постановки завдань та отримання результатів;
- *URC (Unicore Rich Client – багатофункціональний клієнт UNICORE):* клієнт, заснований на базі інтерфейсу середовища Eclipce, надає у графічному вигляді повний набір всіх функціональних можливостей системи UNICORE;
- *HiLA (High Level API for Grid Applications* – високорівневий програмний інтерфейс для додатків грід): забезпечує розробку клієнтів до системи *UNICORE*;
- Портали: доступ користувачів до грід-ресурсів через інтернет, за допомогою інтеграції *UNICORE* та систем інтернет-порталів.

Система UNICORE

- Проміжний сервісний шар містить усі сервіси та компоненти системи *UNICORE*, засновані на стандартах *WSRF* та *SOA*P:
- ***Шлюз*** - це компонент, що забезпечує доступ до сайту *UNICORE* за допомогою аутентифікації всіх вхідних повідомлень.
- ***Компонент XNJS*** забезпечує управління завданнями та виконання ядра *UNICORE* 6.
- ***Регістр сервісів*** забезпечує реєстрацію та пошук ресурсів, доступних у грід-середовищі.Також на рівні сервісного шару забезпечується підтримка безпечних з'єднань, авторизації та автентифікації користувачів.

Система UNICORE

- В основі архітектури *UNICORE* лежить системний шар. Інтерфейс цільової системи *(TSI - Target System Interface)* забезпечує взаємодію між *UNICORE* та окремим ресурсом грід-мережі. Він забезпечує трансляцію команд, що надходять із грід-середовища локальній системі.
- Основною перевагою використання системи *UNICORE 6* для розробки РОС можна вважати наявність багатого арсеналу різних клієнтів, які забезпечують взаємодію користувача з ресурсами обчислювальної мережі, а також розвинені засоби забезпечення безпеки при розробці грід-додатків.

Параметричні моделі продуктивності Грід

Основна відмінність грід-систем - можливість агрегування та спільного використання великих наборів гетерогенних ресурсів, розподілених між географічно розділеними областями.

До високогетерогенного, динамічно-формованого розподіленого середовища важко безпосередньо застосувати традиційні метрики продуктивності

- Швидкість обчислень
- Пропускна спроможність каналу

Використання спеціалізованих метрик оцінки якості сервісів, що надаються.

Параметричні моделі продуктивності Грід

- Припустимо, що в грід-середовищі є *m* ресурсів і існує система розподілу завдань т, що забезпечує розподіл поставлених завдань *j є t* доступні ресурси. У рамках цієї системи, кожне завдання може бути розбите на дії *k є j*. Кількість завдань у системі | *т* |*;* кількість дій у завданні | *j* |. При постановці завдання вказується час *dj*, до якого користувач бажає отримати результати рішення. Кожне завдання *j* та всі її дії *k є j* надходять у грід у момент часу *rj*. У зв'язку з тим, що грід працює в "online"-режимі,значення *rj* невідомо заздалегідь для більшості завдань. Як тільки з'являється певне завдання, проводиться планування її роботи, після чого проводиться пошук та виділення ресурсів необхідних для запуску.

Метрики, що залежать від часу

- 1. Мінімально можливий час вирішення *j*-ої задачі:

$$C_j(S) = \max_{k \in j} C_k(S),$$

де Ck(S) - час виконання k-того (k є j) фінального розподілу S.

2. Загальний час вирішення *j*-ої задачі:

$$p_j = C_j(S) - \min_{k \in j}(C_k(S) - p_k),$$

де pk – час реалізації k-ої дії.

3. Показник максимального запізнення задач:

$$L_{max} = \max_{j \in \tau}(C_j(S) - d_j)$$

При оптимізації розподіленого середовища намагаються забезпечити

Lmax - min

Метрики, що залежать від часу

- **4. Кількість завдань, що запізнилися:**

$$TJ_{(j \in \tau \wedge C_j > d_j)}$$

Визначимо споживання ресурсів певним підзавданням RC_k як добутоквідповідного часу рішення на кількість використовуваних ресурсів:

$$RC_k = p_k \cdot m_k$$

тоді споживання ресурсів j-ою задачею дорівнює: $RC_j = \sum_{k \in}^{1} RC_k$

А споживання ресурсів усіма завданнями планувальника: $RC(S)^1 = \sum_{j \in r}^{1} RC_j$

Величина використання доступних ресурсів може бути обчислена за такою формулою:

$$U = \frac{RC(S)}{m(\max_{j \in \tau} C_j(S) - \min_{j \in \tau}(C_j(S) - p_j))}$$

Метрики, що залежать від часу

У виконання завдання можуть відбуватися збої. Тоді завдання має бути запущене кілька разів, щоб успішно виконатися. Внаслідок цього ми можемо визначити повне споживання ресурсів *RC{k,j}true* та повну величину використання ресурсів *Utrue*, як відповідну величину плюс витрати на виконання збійних завдань. Тоді метрика витрат може бути визначена за формулою:

$$WASTE = U^{true} - U.$$

При оптимізації ця величина також має бути мінімізована.

Користувачі та адміністратори часто висувають різні (і навіть конфліктуючі) вимоги до грід системи, тому важко підібрати метрику, яка б задовольняла всіх. З погляду користувача цінними є метрики:

середній час відповіді (*Average Response Time – ART*): $ART = \frac{1}{|\tau|}\sum_{j\in\tau} C_j(S)$

Метрики, що залежать від часу

 середній час очікування (*Average Wait Time - AWT*): $$AWT = \frac{1}{|\tau|}\sum_{j\in\tau}(C_j(S) - p_j)$$

Відносно хорошим та простим методом вимірювання справедливості використання ресурсів є розрахунок девіації середнього часу очікування:

$$AWTD = \frac{1}{|\tau|}\sqrt{\sum_{j\in\tau}(C_j(S) - p_j)^2} - \left(\sum_{j\in\tau}\frac{C_j(S)-p_j))^2}{|\tau|}\right.$$

Для обробки результатів моніторингу грід-системи використовують метрику ефективності грід (*Grid Efficiency - GE*):

$$GE = \frac{\sum_{j\in\tau}\left(\left(EndTime_j - StartTime_j\right)CPU_{S_J}CPUSpeed_j\right)}{\left(EndTime_{lastJob} - Submit_{firstJob}\right)\sum_{m\in M}\left(CPU_{S_m}\,CPUSpeed_m\right)}100\%$$

де

$\left(EndTime_{lastJob} - SubmitTime_{firstJob}\right)$ - час роботи системи,

Метрики, що залежать від часу

- *CPUSj* та *CPUSpeedj* - кількість процесорів, використаних завданням j та їх продуктивність,
- *CPUSm* та *CPUSpeedm* - кількість процесорів у машині m та їх продуктивність.

Метрики, що залежать від обсягу роботи

- У сучасних Грід системах, можливість завершити виконання цього обсягу роботи може бути навіть важливішим, ніж прискорення, отримане за допомогою такого виконання.
- Грід вимагає перевизначення поняття помилки програми: Грід додаток, який не зміг успішно виконатися в рамках відведеного йому бюджету, генерує повідомлення про помилку, як тільки виявиться неможливість успішного виконання.
- Наприклад, якщо не знайдено ресурсів для виконання обчислень або у зв'язку з настанням крайнього терміну роботи програми. Відмовостійкість грід-системи можна визначити як можливість на якомога більший термін переносити час появи помилки, поки є хоч якісь шанси того, що програма завершиться успішно.

Метрики, що залежать від обсягу роботи

- 1. *Метрика завершеного обсягу роботи (Workload Completion)* – співтавлення успішно завершених завдань до обсягу всіх завдань, поставлених планувальнику грід-середовища:
- 2. *Метрика завершення дій (Task Completion)* –відношення кількості успішно завершених дій до загальної кількості дій, виконаних у рамках системи розподілу завдань:
- 3. *Метрика завершення розблокованих дій (Enabled Task Completion):* – Розблокованою називається дія, яка може бути виконана тільки після виконання всіх залежностей для даної дії.

БЕЗПЕКА ФАЙЛОВОЇ СИСТЕМИ. СЕРТИФІКАТ ВІДКРИТИХ КЛЮЧІВ

- *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).*

БЕЗПЕКА ФАЙЛОВОЇ СИСТЕМИ. СЕРТИФІКАТ ВІДКРИТИХ КЛЮЧІВ

- 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.
- **Підписник** – 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.

БЕЗПЕКА ФАЙЛОВОЇ СИСТЕМИ. СЕРТИФІКАТ ВІДКРИТИХ КЛЮЧІВ

- **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.

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.

Вимоги безпеки GRID

- **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.

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).

Інфраструктура захисту GRID (GSI)

- ***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.

Інфраструктура захисту GRID (GSI)

– **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.

– **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).

RAID – технології

- RAID (Redundant Array of Independent Disks) – технологія *розподіленого* зберігання даних

1) Базуються на дискових масивах зовнішніх накопичувачів, які мають різну логічну архітектуру та методи організації дублювання і зберігання блоків даних та інформації.
2) Відрізняються швидкістю доступу до даних і кількістю дискових масивів для реалізації.
3) Управляються спеціальними RAID – контролерами (один контролер на 2-3 диски)
4) Допускають “гарячу” заміну дисків “на ходу” при їх відмові - технологія Hot Swap (резервні диски).

RAID – технології

- RAID – технології

5) Мають декілька рівнів, які відрізняються кількістю фізичних дисків – від 0 до 60-го
6) Принцип роботи полягає в одночасному читанні (або запису) блоків даних з декількох масивів дисків
7) Кожен з рівнів RAID має свої сильні сторони, забезпечуючи або більш високі швидкості, або виняткову ємність пам'яті, або підвищену відмовостійкість.
8) У зв'язці «контролер-шина-диск» найповільнішим є диск.
9) Використовуючи велику кількість дисків, можна отримати збільшення швидкості запису / зчитування системи до тих пір, поки дозволяє пропускна здатність інтерфейсної шини.

RAID – технології

- Технологія RAID розроблена в 1980-х роках була задумана як об'єднання декількох дисків в єдиний дисковий масив з метою збільшення ємності, підвищення надійності та доступності даних (швидкості читання/запису).
- ***RAID*** – *це система, яка координує розподілення даних по різних дискових масивах за допомогою стандартних механізмів читання/запису для забезпечення високої якості обслуговування*

RAID – технології

Базові рівні RAID:

- RAID0: Чергування (Striping)
- RAID1: Віддзеркалення (Mirroring)
- RAID3: Чергування з виділеним диском парності (Virtual disk blocks)
- RAID4: Чергування з виділеним диском парності (Dedicated parity disk)
- RAID5: Чергування парності (Striped parity)
- RAID6: Подвійне чергування парності (Dual parity)
- RAID10:Будується RAID0 з груп масивів RAID1
- RAID60:Будується RAID0 з груп масивів RAID6
- Дисковий масив **JBOD** (Just a bunch of disks - просто пачка дисків).

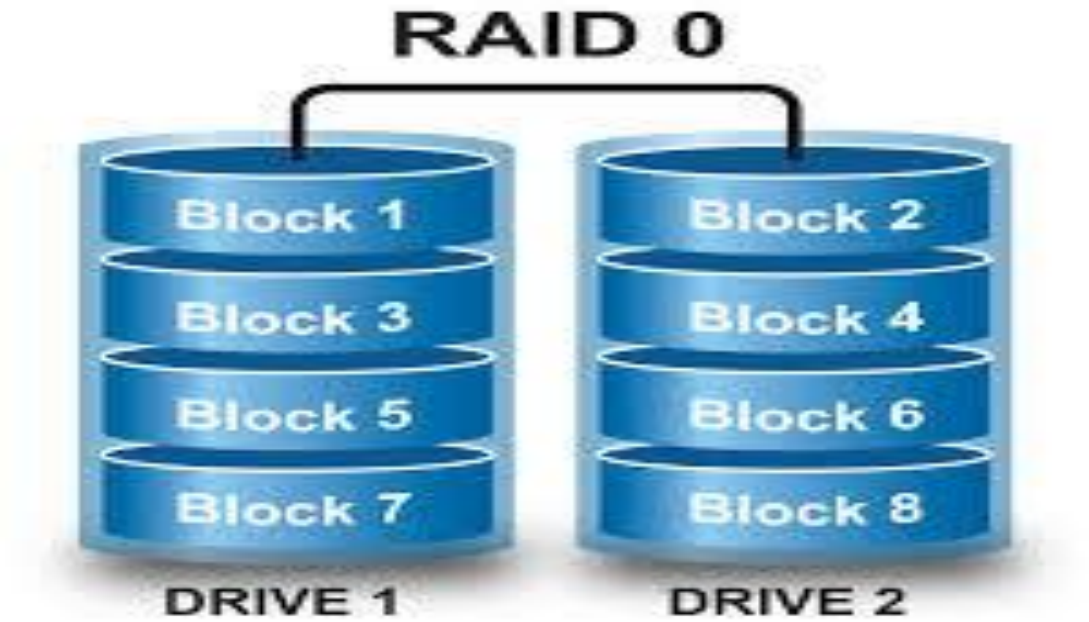
Системи зберігання даних

- Апаратний RAID **AgeStar 3C4B3A (Black)**

RAID – технології

- Конфігурація RAID 0

Недолік – низька надійність.

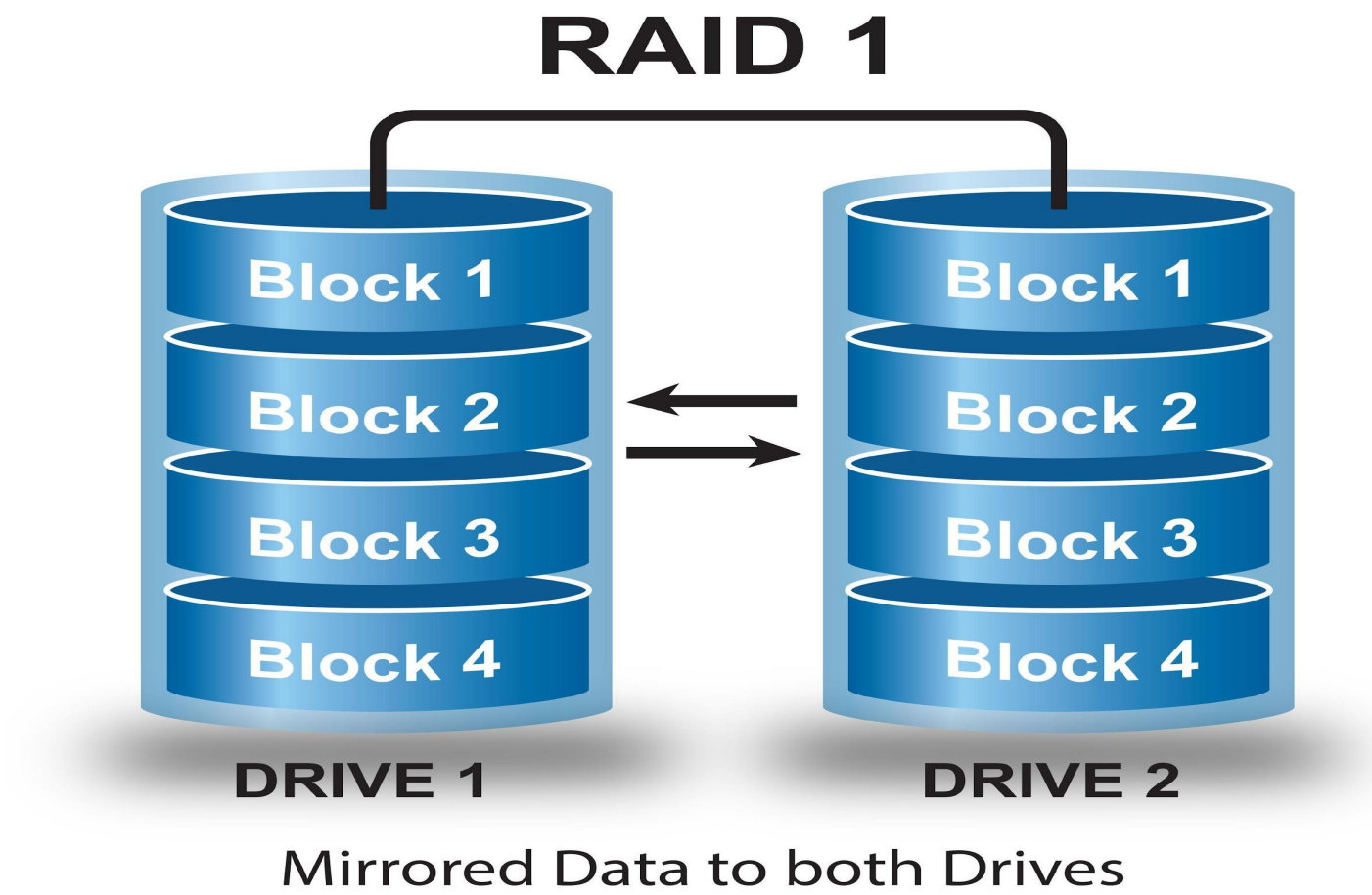


RAID – технології

I. Характеристики RAID 0

a) Принцип роботи - striping (чергування). Масив при якому інформація розбивається на однакові по довжині блоки, а потім записується по черзі на кожен диск в структурі.
b) Основне призначення такої системи - фактичне збільшення продуктивності в 2 рази, при цьому вам буде доступний повний обсяг усіх дисків.
c) Можна використовувати необмежену кількість дисків різної ємності. У разі якщо диски володіють різними показниками швидкості, то кінцевий результат буде вираховуватися за найповільнішого HDD.

RAID – технології



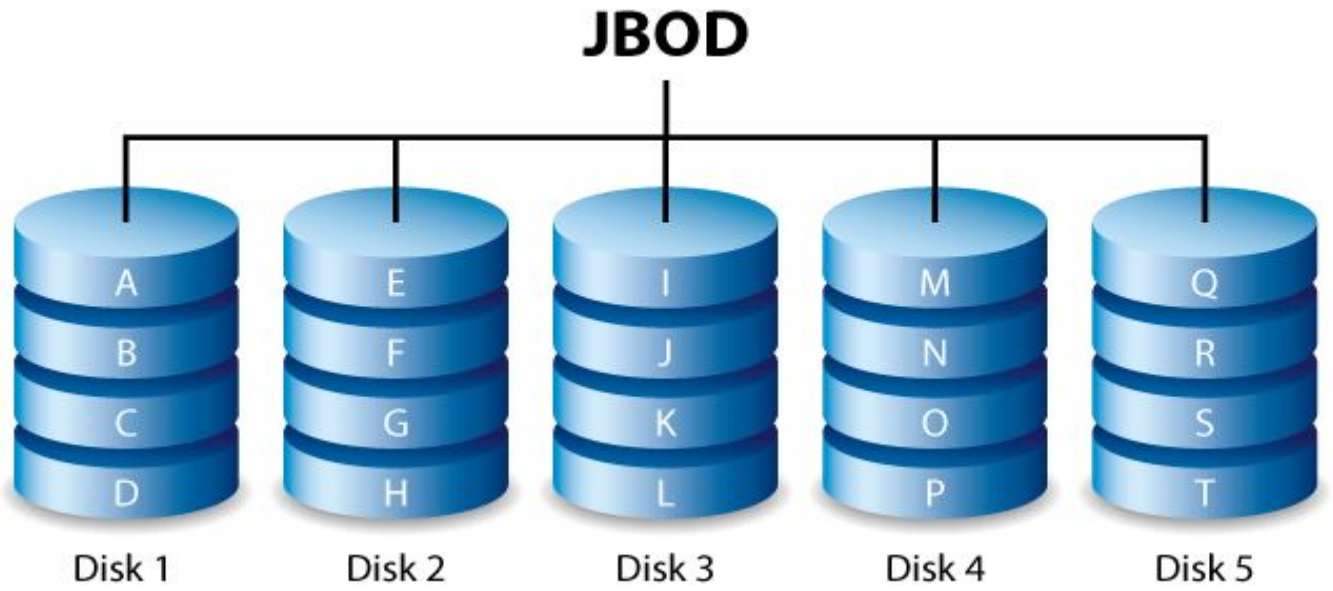
RAID – технології

II. Особливості RAID 1

a) Призначення таких систем - резервація та клонування інформації.
b) Зеркалювання даних на 2-х чи 3-х дисках
c) Інтерфейси підключення SATA1/SATA2/ SATA3
d) Переваги – висока надійність, а недолік – дорожнеча зберігання байта даних

RAID – технології

- **JBOD** (Just a bunch of disks - просто пачка дисків) — дисковий масив.



RAID – технології

III. Особливості дисків JBOD

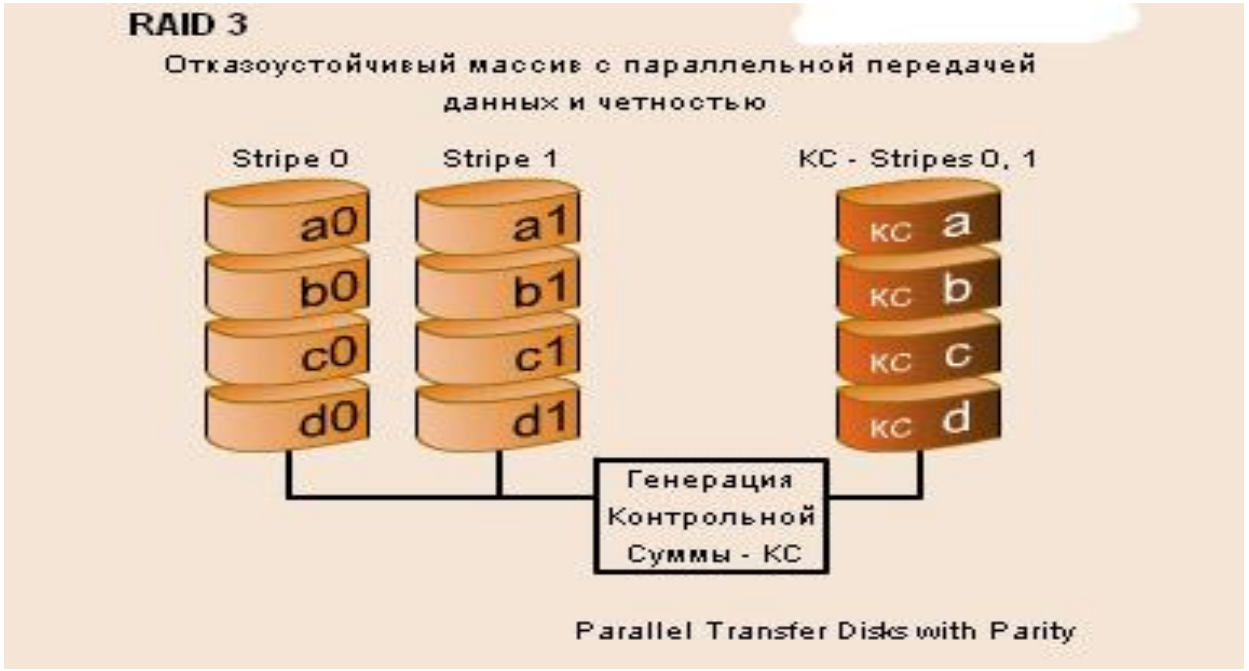
a) Диски в конфігурації JBOD зберігають дані послідовно. Наприклад, дані спочатку записуються на Диск 1. Після заповнення Диска 1 дані будуть записані на Диск 2, потім на Диск 3 і т. д.

b) Двома перевагами цього рівня RAID є наявність 100% загальної ємності дисків та легке розширення.

c) Однак усі дані будуть втрачені, якщо вийде з ладу один диск.

RAID – технології

- RAID 3- Чергування парності (Striped parity) на рівні байтів



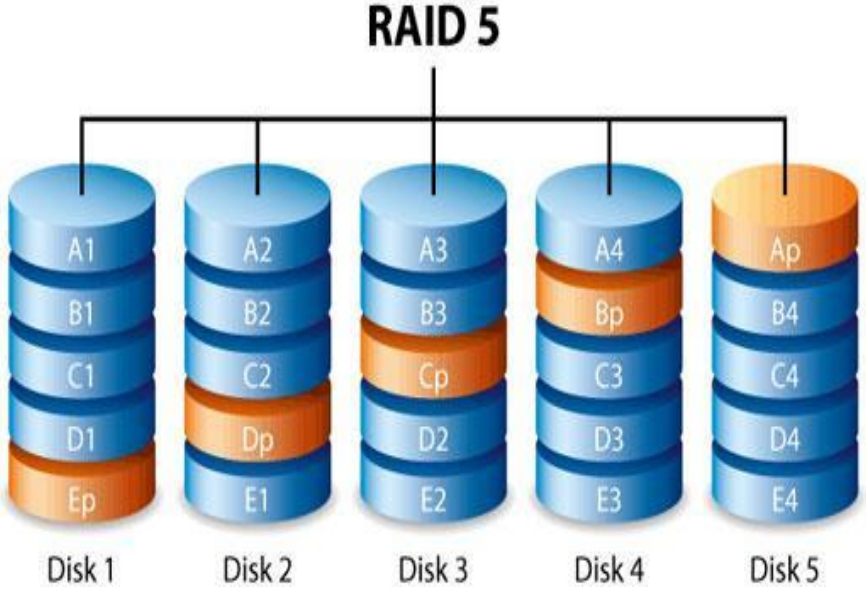
RAID – технології

IV. Особливості RAID 3

a) Опис: Дані чергуються по дисках масиву на рівні байтів. Необхідний додатковий диск на якому зберігається інформація про парності. Мінімально три диска в масиві
b) Продуктивність: Низька на операціях запису
c) Плюси: Дані залишаються повністю доступними при виході з ладу одного диска
d) Мінуси: Продуктивність
e) Використання: Рідко мінливі, дані, що часто зчитуються

RAID – технології

- Масив RAID 5 - Чергування парності (Striped parity)



RAID – технології

V. Особливості RAID 5

a) Сильно схожий з RAID 1 за своїм принципом роботи. Потрібно мінімум 3 накопичувача, на одному з яких буде зберігатися продубльована інформація. В цьому випадку вам буде доступний практично весь обсяг в системі, крім одного диска з даними під відновлення.
b) Крім того, збільшиться і продуктивність, але не в кілька разів, як у випадку з RAID 0.
c) Основна відмінність RAID 5 від RAID 10 - це рівень надійності і доступний обсяг. Даний масив призначений для більш специфічних завдань, коли разом зібрано величезну кількість дисків.

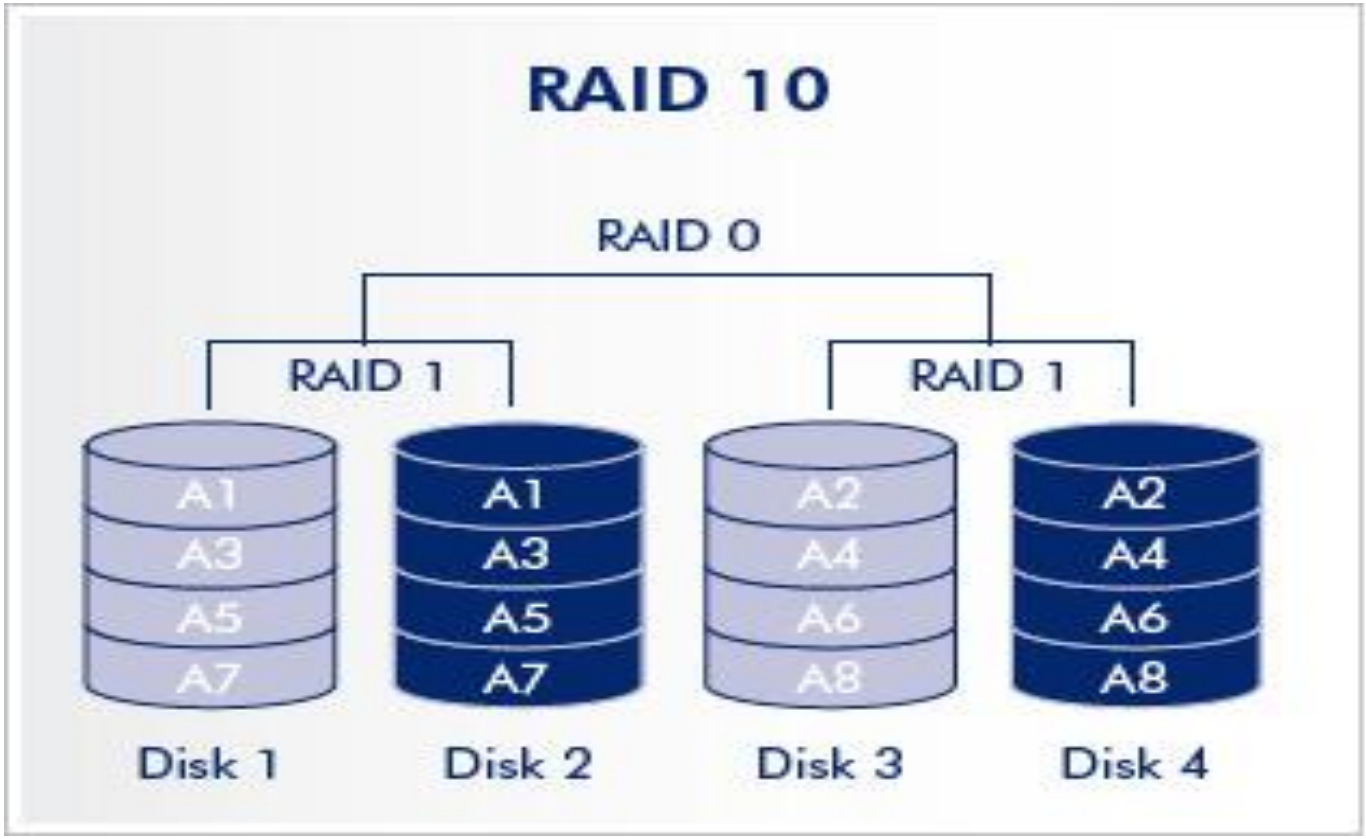
RAID – технології

Особливості застосування RAID 5

1) Опис: На відміну від RAID3 дані і парність чергуються по всіх дисках масиву. Дуже добре мати додатковий вакантний диск (hot spare disk) на випадок якщо один з дисків масиву вийде з ладу. Тоді контролер підхопить вакантний диск і масив буде перебудований. Мінімально три диска в масиві
2) Продуктивність: Краще ніж в RAID3, тому що вирішена проблема виділеного диска парності
3) Плюси: Досягнуто баланс читання / запису / резервування
4) Мінуси: Просідання продуктивності під час перебудови масиву. Якщо не використовується кеш запису (рейд-контролер не обладнаний батарейкою і не налаштований), то просадка буде особливо чутлива
5) Використання: Веб-сервера, файлові сервера де використовується інтенсивне читання даних

RAID – технології

- Дисковий масив RAID10



RAID – технології

VIII. Система **RAID 10 (1+0)**

a) RAID10 - поєднує в собі все найкраще з RAID 1 та RAID 0.
b) Вам буде потрібно мінімум 4 носія, і їх кількість завжди має бути парним.
c) В даному масиві ви отримуєте високу продуктивність і високу надійність. Однак, як у випадку із RAID 1, вам буде доступна лише половина від загального обсягу всієї системи.
d) Переваги – високі надійність та продуктивність
e) Недоліки - підсумковий обсяг рівний 1/2 від загального + дорожнеча вартості байта даних

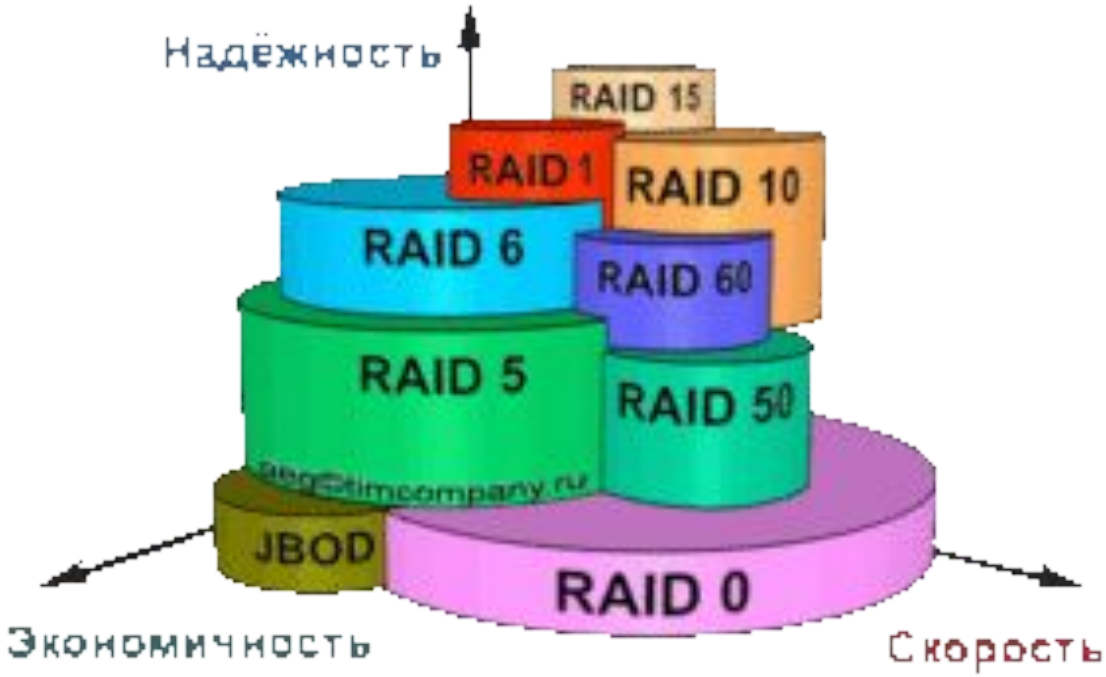
RAID – технології

Особливості застосування RAID10

1) Опис: 3 груп масивів RAID1 будується RAID0
2) Продуктивність: Вважається найшвидшим і надійним масивом
3) Плюси: Підвищено надійність збереження даних. Масив буде життєздатний поки в кожній групі масивів RAID1 буде робочим останній диск
4) Мінуси: Вартість, один з найдорожчих
5) Використання: Веб-сервера з активним читанням даних, додатки, які використовують транзакції

RAID – технології

- Ієрархія дискових масивів RAID



RAID – технології

RAID mode	Minimum hard drives
SimplyRAID	1 (no data protection) or 2 (with data protection)
JBOD	1
RAID 0	2
RAID 1	1 (no data protection) or 2 (with data protection)
RAID 5	3
RAID 6	4
RAID 10	4

RAID – технології

- На практиці найчастіше використовують три види RAID-масивів.
- Це RAID 1, RAID 10 і RAID 5.

Розподілені бази даних

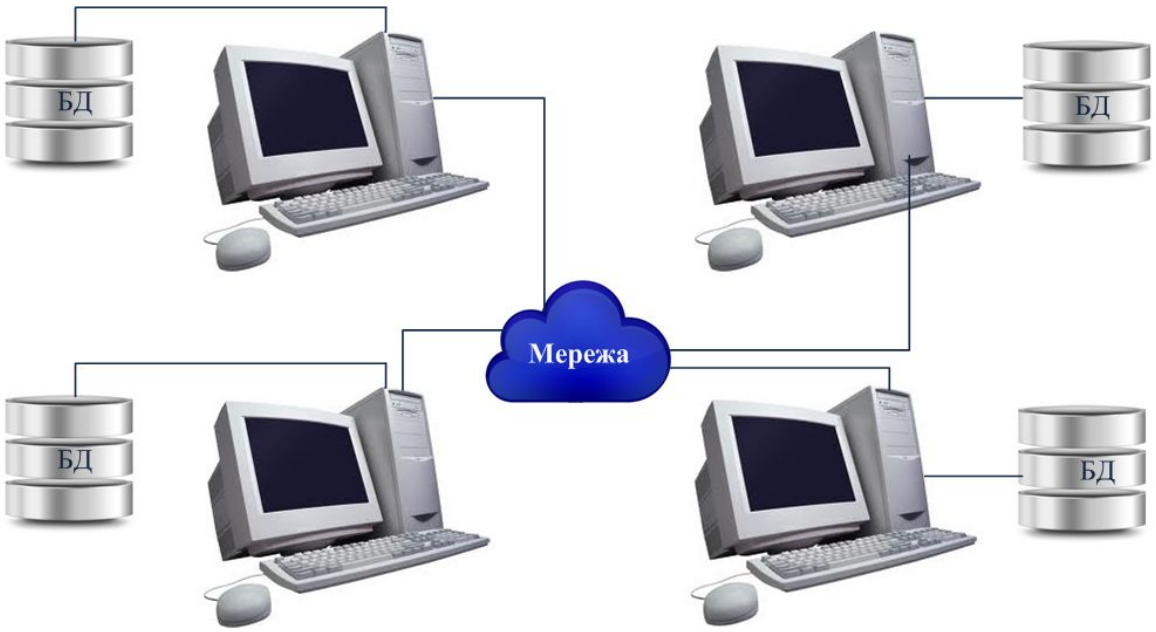
- **Розподілена база даних** (РБД) – це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.
- **Розподілена система управління базами даних** (РСУБД) – це програмне забезпечення, яке керує РБД і надає такі механізми доступу до них, що їх застосування дає користувачу можливість працювати з РБД як з однією цілісною базою даних.
- **Розподілена система баз даних** (РСБД) – це РБД разом із РСУБД.

Розподілені бази даних

- **Розподілена база даних** (англ. *distributed database, DDB*) — сукупність логічно взаємопов'язаних баз даних) — сукупність логічно взаємопов'язаних баз даних, розподілених у комп'ютерній мережі. Логічний зв'язок баз даних в розподіленій базі даних забезпечує система управління розподіленою базою даних, яка дозволяє управляти розподіленою базою даних таким чином, щоб створювати у користувачів ілюзію цілісної бази даних.

Розподілені бази даних

- Структура Розподіленої бази даних



Розподілені бази даних

- Кожний з вузлів мережі містить свою базу даних, однак вони розглядаються як логічно єдина база, а не як сукупність розкиданих у мережі файлів.
- Усі дані є логічно взаємозалежними.
- Розподілена СУБД — це повноцінна СУБД, що виконує всі необхідні функції з керування даними.

Розподілені бази даних

Система управління розподіленою базою даних (СУРБД) складається з (можливо, порожнього) набору вузлів прийому запитів і набору вузлів збереження даних.

Вузли прийому запитів реалізують прозорий інтерфейс доступу до даних, що зберігаються в вузлах збереження даних, та приховують фрагментацію даних між вузлами.

Кожен з вузлів може бути представлений незалежним комп'ютером в комп'ютерній мережі з власною (можливо відмінною від інших вузлів) операційною системою.

Розподілені бази даних

Система управління розподіленою базою даних є однорідною (гомогенною), якщо на кожному з вузлів збереження даних застосовуються однакові СУБД, в іншому випадку система управління розподіленою базою даних є неоднорідною (гетерогенною). Внаслідок застосування стандартизованих механізмів доступу до баз даних відмінності між однорідними та неоднорідними системами нівелюються і не є критичними.

Розподілені бази даних

Властивості архітектури

В архітектури програмно-технічного комплексу розподілених СКБД є три основні характеристики:

1) **Неоднорідність.**
   Важливою характеристикою є міра однорідності програмно-технічних засобів СКБД.
2) **Розподіленість.**
   Спосіб розподілу компонентів системи баз даних за комп'ютерами мережі визначається тим, чи є в мережі єдиний комп'ютер з повноваженнями розподіленої СКБД, або ж їх кілька.
3) **Автономність.**
   Характеризує, наскільки самостійно компоненти СКБД можуть виконувати свої функції.

До питань автономності належать:

- автономність проектних рішень;
- автономність обчислень.

Розподілені бази даних

Різновиди архітектури

Виділяють такі основні різновиди архітектури програмно-технічних засобів розподіленої СУБД, як:

а) **Архітектура одноранговової мережі.**

Якщо всі комп'ютери мережі є серверами та на кожному комп'ютері розміщено розподілену СУБД і базу даних та з кожного комп'ютера можна надіслати до іншого запит на отримання необхідних даних. То така архітектура буде **архітектурою одноранговової мережі**

Розподілені бази даних

b) **Архітектура з багатьма незалежними серверами.**

Якщо існують багато серверів, що мають доступ до своїх локальних баз даних. У такому випадку на комп'ютерах клієнтів має зберігатись інформація стосовно того, які дані на яких серверах розташовані, а також має бути розміщене програмне забезпечення, що дає змогу розкласти запити для їхнього виконання на різних серверах і потім об'єднати результати.

Розподілені бази даних

c) **Архітектура із взаємодіючими серверами.**

У цьому випадку передбачається, що кожен сервер містить повну інформацію стосовно того, які дані на яких серверах зберігаються, а також здатний обробляти розподілені запити. У разі потреби один сервер може звернутися до іншого для одержання необхідних даних. Кожен клієнт має свій сервер, до якого звертається для виконання операцій над розподіленою базою даних.

Розподілені бази даних

d) **Клієнт-серверна архітектура.**

При даному виді архітектури передбачено наявність єдиного комп'ютера-сервера і багатьох комп'ютерів-клієнтів, що взаємодіють між собою через канали зв'язку. На сервері розташована СУБД та інтегрована (централізована) база даних. Ніякого розподілу баз даних за вузлами мережі немає. На клієнтських комп'ютерах виконуються додатки, які працюють із серверною базою даних, а також розміщене програмне забезпечення для зв'язку з віддаленою СУБД. Як клієнти, так і сервер оснащені комунікаційним програмним забезпеченням.

Розподілені бази даних

1. Принципи розподілу даних

- Розподіл даних між вузлами збереження даних забезпечується на основі **механізмів фрагментації та реплікації**, і досягається шляхом вертикального (на окремі поля записів бази даних) або горизонтального (на окремі записи бази даних) поділу даних. Наприклад, при вертикальному поділі даних інформація про покупців може зберігатись на одному вузлі збереження даних, про їх купівлі — на другому, про товари — на третьому тощо; при горизонтальному поділі даних інформація про покупців, купівлі та товари зберігається на одному вузлі, а поділ між вузлами здійснюється за країнами покупців (для кожної країни — окремий вузол збереження даних), або за типами товарів тощо.
- **Фрагментація** здійснюється за принципами, що дозволяють наблизити дані до місць їх найбільш інтенсивного використання для зменшення витрат на пересилання даних.
- Наближення даних до місць їх найбільш інтенсивного використання також забезпечується **реплікацією** даних.

Розподілені бази даних

2. **Механізми доступу до даних**

- Доступ до даних в розподіленій базі даних звичайно забезпечується в трирівневій моделі: клієнт — сервер додатків — вузли збереження даних. При інтернет-доступі до даних роль сервера додатків відіграє *веб-сервер* або *спеціальний додаток* на боці клієнта.
- Для вирішення спеціальних задач обслуговування даних можливий локальний доступ до окремих вузлів збереження даних, при цьому недоступні можливості доступу, що базуються на прозорості інтерфейсів доступу до даних.

Розподілені бази даних

3. Розподілене зберігання даних

3.1 Фрагментація

- Суть фрагментації полягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі.
- Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися.
- За іншого підходу допускається, що будь-яке відношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Розподілені бази даних

Фрагментація відношень. Завдання фрагментації відношень формулюється в такий спосіб. Нехай задане відношення R. Його потрібно зобразити у вигляді сукупності відношень R1, …, Rn так, щоб ця сукупність відповідала критеріям ефективності (за часом доступу, пам'яттю, завантаженістю комп'ютерів тощо).

- Фрагментація є коректною, якщо вона повна, не містить перетинів і може бути реконструйована.
- Є три типи фрагментації відношень:

– горизонтальна;
– вертикальна;
– змішана

Розподілені бази даних

- *Горизонтальна фрагментація* полягає в розподілі кортежів відношення за фрагментами БД.
- Суть *вертикальної фрагментації* полягає в тому, що відношення поділяється на дві чи більше проєкцій, тобто схема відношення поділяється на певну множину підсхем БД.
- *Змішана фрагментація* передбачає послідовне застосування вертикальної і горизонтальної фрагментацій.

Розподілені бази даних

3.2 Декомпозиція

Декомпозиція відношення R на фрагменти R1, R2, … Rn є повною тоді й лише тоді, коли кожен елемент даних з R належить якомусь із відношень Ri.

- Декомпозиція відношення R на фрагменти R1, R2, … Rn може бути реконструйована, якщо існує такий реляційний вираз φ(R1, R2, … Rn), що R=φ(R1, R2, … Rn).
- Декомпозиція відношення R на фрагменти R1, R2, … Rn не містить перетинів, якщо будь-який елемент даних з R міститься не більш ніж в одному фрагменті.

Розподілені бази даних

4. Розподіл даних за вузлами мережі

- Після отримання усіх необхідних фрагментів відношень постає проблема розподілу цих фрагментів за вузлами мережі. Єдиних рекомендацій стосовно того, як це робити, немає. Потрібно знайти оптимальний розподіл фрагментів F за вузлами мережі S за умови, що відомий розподіл додатків Q за вузлами мережі.

Розподілені бази даних

5. Реплікація

- Реплікація є механізмом розподілу даних за вузлами, що в свою чергу дозволяє зберігати копії тих самих даних на різних вузлах мережі для прискорення пошуку і підвищення стійкості до відмов. Відношення або фрагмент є реплікованим, у випадку якщо його копії(або репліки) зберігаються на двох або більше вузлах .
- За повної реплікації відношення його копії зберігаються на всіх вузлах мережі. Допускається ситуація, коли вся база даних зберігається на всіх вузлах мережі — це називається *повною реплікацією бази даних.*

Розподілені бази даних

5.1 Механізми реплікації

- *Видавець* — сервер, що надає розміщені на ньому дані для копіювання на інші сервери.
- *Дистриб'ютор* — сервер, що підтримує розподілену базу даних.
- *Передплатник* — сервер, що отримує копії даних, надані видавцем.

Існують два методи відновлення даних передплатників:

- **Реплікація за запитом** полягає в тому, що передплатник періодично звертається до дистриб'ютора із запитом про зміни, що відбулися з моменту останнього з'єднання.
- **Примусова реплікація,** с свою чергу, коли дистриб'ютор сам встановлює з'єднання з передплатником і пересилає йому необхідні дані.

Розподілені бази даних

- Залежно від методу реплікації, передплатники можуть, або не можуть вносити зміни в репліковані дані. У найпростішому випадку змінювати дані може тільки видавець, у складніших моделях реплікації — передплатники і видавці.
- Змінені дані, отримані від усіх передплатників, синхронізуються і поєднуються з даними видавця, а потім розсилаються передплатникам.

5.2 Моделі реплікації

- реплікація моментальних знімків;
- реплікація транзакцій.

Розподілені бази даних

5.3 Топологія реплікацій

Топологія реплікацій описує характер взаємозв'язків між учасниками реплікації:

- реплікація «один-до-багатьох» передбачає наявність одного видавця і кількох передплатників;
- реплікація «багато-до-одного» має місце, коли дані від кількох видавців пересилаються одному передплатнику;
- реплікація «багато-до-багатьох» означає, що дані від кількох видавців пересилаються кільком передплатникам.

Розподілені бази даних

6. Обробка розподілених транзакцій

- Транзакція — набір команд, що виконується як єдине ціле. У транзакції або всі команди будуть виконані, або жодна з них не виконається. Якщо хоча б одна з команд транзакції не може бути виконана, здійснюється відкочування (відновлюється стан системи, в якому вона перебувала до початку виконання транзакції).
- Транзакції мають задовольняти вимоги ACID (Atomicity, Consistency, Isolation, Durability — атомарність, несуперечність, ізольованість, довговічність), що гарантують правильність і надійність роботи системи.

Розподілені бази даних

6.1 Атомарність передбачає таке:

- виконуються всі операції транзакції або жодна з них не виконується;
- якщо виконання транзакції було перерване, то всі зроблені транзакцією зміни мають бути скасовані

6.2 Несуперечність означає, що транзакція, яка працює з несуперечною базою даних, після завершення роботи залишає її також у несуперечному стані. Транзакція не повинна порушувати цілісності бази даних. Для вирішення проблем одночасного доступу інститут ANSI розробив спеціальний стандарт, який визначає чотири рівні блокування

6.3 Властивість **ізольованості** означає, що на роботу транзакції не мають впливати інші транзакції. Транзакція «бачить» дані в тому стані, в якому вони перебували до початку роботи іншої транзакції, або в тому стані, в якому вони перебувають після її завершення.

6.4 Після того, як було підтверджено успішне завершення роботи транзакції (Commit), система має гарантувати, що її результати не будуть втрачені, незважаючи на можливі перебої. Це й називається **довговічністю**.

Розподілені бази даних

Властивості розподілених баз даних

- 12 властивостей розподілених без даних, були формульовані Крістофером Дейтом, одним з найбільших діячів в галузі баз даних.

Розподілені бази даних

1) **Локальна автономія** — управління даними на кожному з вузлів розподіленої системи виконується локально.
2) **Незалежність вузлів** — всі вузли рівноправні і незалежні, а розташовані на них БД є рівноправними постачальниками даних в загальний простір даних.
3) **Безперервні операції** — можливість безперервного доступу до даних в рамках розподіленої БД незалежно від їх розташування і незалежно від операцій, що виконуються на локальних вузлах.
4) **Прозорість розташування** — користувач, що звертається до БД, нічого не повинен знати про реальне, фізичне розміщення даних у вузлах інформаційної системи.
5) **Прозора фрагментація** — можливість розподіленого (тобто на різних вузлах) розміщення даних, логічно поєднаних в єдине ціле. Існує фрагментація двох типів: горизонтальна і вертикальна.
6) **Прозоре тиражування** — тиражування даних — це асинхронний процес перенесення змін об'єктів вихідної бази даних в бази, розташовані на інших вузлах розподіленої системи

Розподілені бази даних

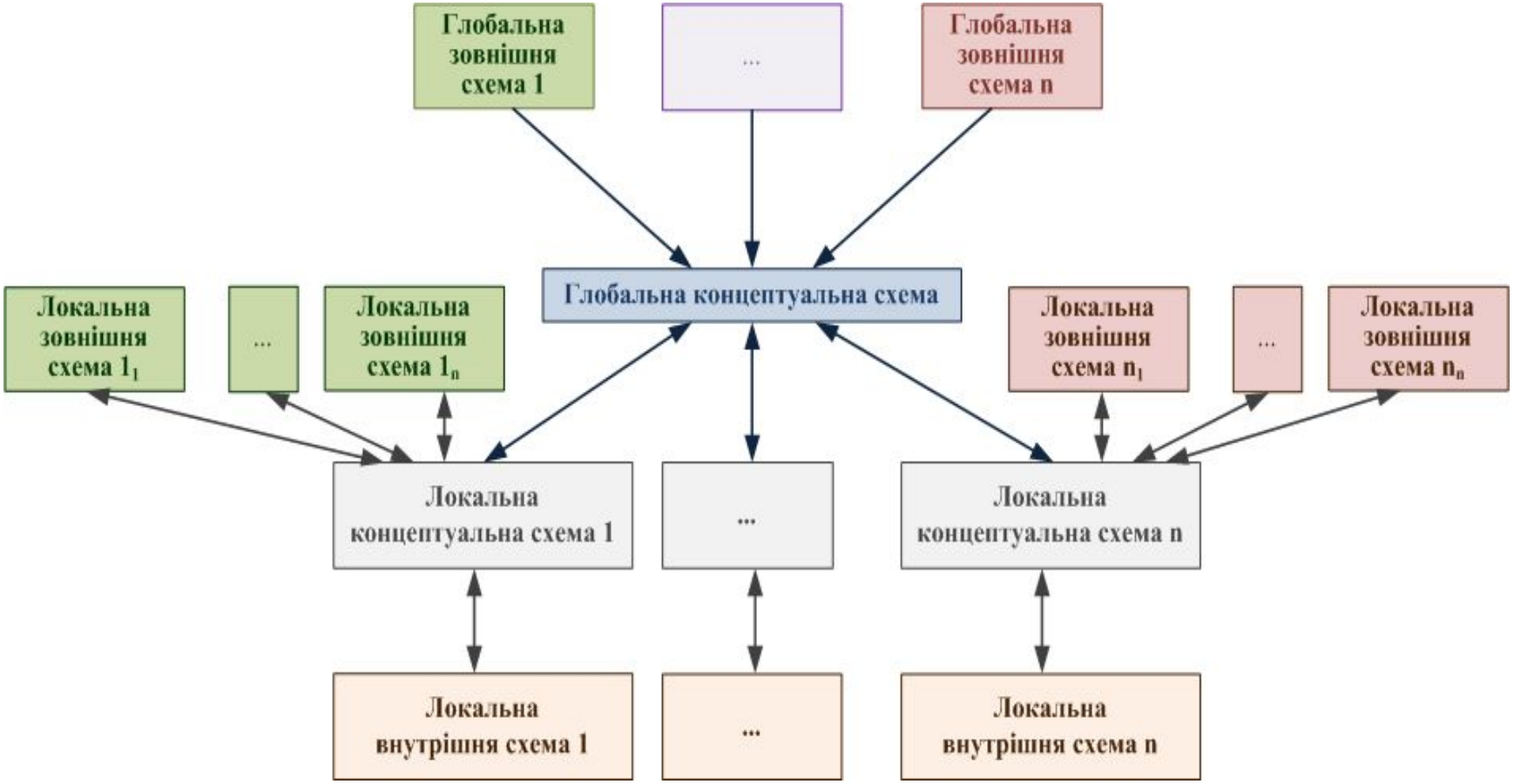
7) **Обробка розподілених запитів** — можливість виконання операцій вибірки даних з розподіленої БД, за допомогою запитів, сформульованих на мові SQL
8) **Обробка розподілених транзакцій** — можливість виконання операцій оновлення розподіленої бази даних, які не порушують цілісність і узгодженість даних.
9) **Незалежність від устаткування** — як вузли розподіленої системи можуть виступати ПК будь-яких моделей і виробників
10) **Незалежність від операційних систем** — різноманіття операційних систем, керуючих вузлами розподіленої системи
11) **Прозорість мережі** — спектр підтримуваних конкретною СУБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД
12) **Незалежність від баз даних** — в розподіленій системі можуть працювати СУБД різних виробників, і можливі операції пошуку і оновлення в базах даних різних моделей і форматів.

Архітектура розподілених баз даних

Логічна архітектура розподілених баз даних

- Це архітектура логічно взаємозалежних даних. Графічне зображення логічної архітектури розподілених баз даних наведене на рис. Особливість архітектури РБД полягає в тому, що виникає ще один рівень, глобальний концептуальний, завданням якого (як і в моделі ANSI/SPARC) є зображення концептуальної моделі предметної області в цілому. На цьому рівні описується характер розподілу даних.

Архітектура розподілених баз даних



Архітектура розподілених баз даних

- На локальному концептуальному рівні здійснюється локальний опис предметної області (***ПО***). Тобто схема цього рівня містить опис тільки тієї частини предметної області, що є специфічною для конкретного вузла розподіленої ***ПО***; дані інших вузлів розподіленої ***ПО*** в схемі не вказуються. Відображення «глобальний-концептуальний/ локальний-концептуальний» дають змогу зобразити глобальну концептуальну схему у вигляді сукупності локальних концептуальних схем, і навпаки.

Архітектура розподілених баз даних

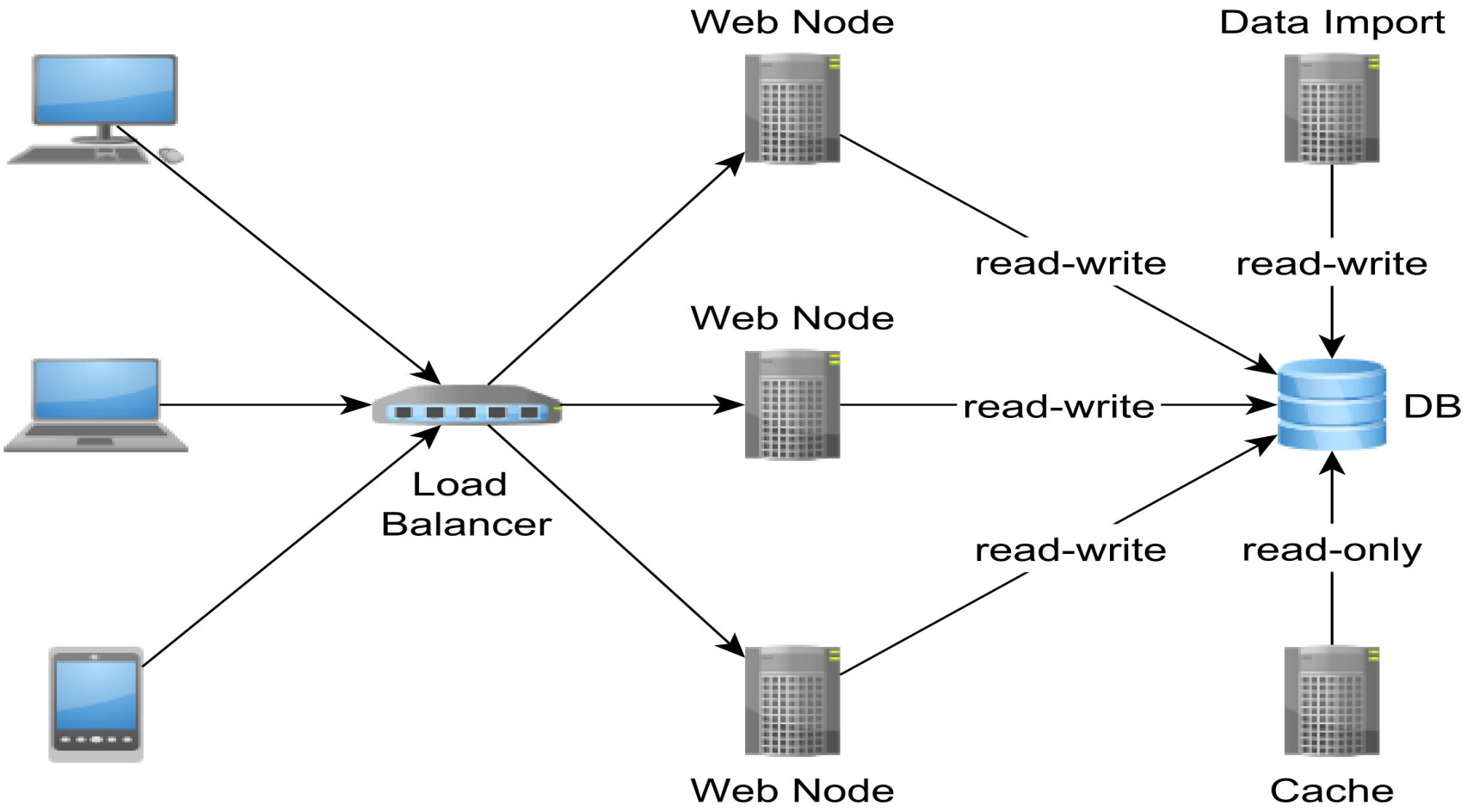
- *Глобальна зовнішня схема* зображує дані, необхідні користувачам і додаткам, у бажаному для них вигляді, незалежно від способу розподілу даних за вузлами. Це завдання вирішується завдяки тому, що глобальні зовнішні схеми базуються на глобальній концептуальній схемі.
- *Локальні зовнішні схеми* базуються на локальних концептуальних схемах, відтак вони можуть посилатися лише на ті дані, що розташовані на відповідному вузлі розподіленої ***ПО***.
- *Локальні внутрішні схеми* – це схеми зберігання даних на конкретних вузлах розподіленої ***ПО***. Вони пов’язані з відповідними локальними концептуальними схемами.

Архітектура розподілених баз даних

Єдина точка відмови

- Сервер бази даних є центральною частиною корпоративної системи, і, якщо він вийде з ладу, доступність служби може статися не можливою.
- Якщо сервер бази даних працює на одному сервері, то у нас є єдина точка збою. Будь-яка проблема з апаратним забезпеченням (наприклад, збій дисковода) або несправність програмного забезпечення (наприклад, проблеми з драйверами, несправні оновлення) призведе до того, що система стане недоступною.

Архітектура розподілених баз даних

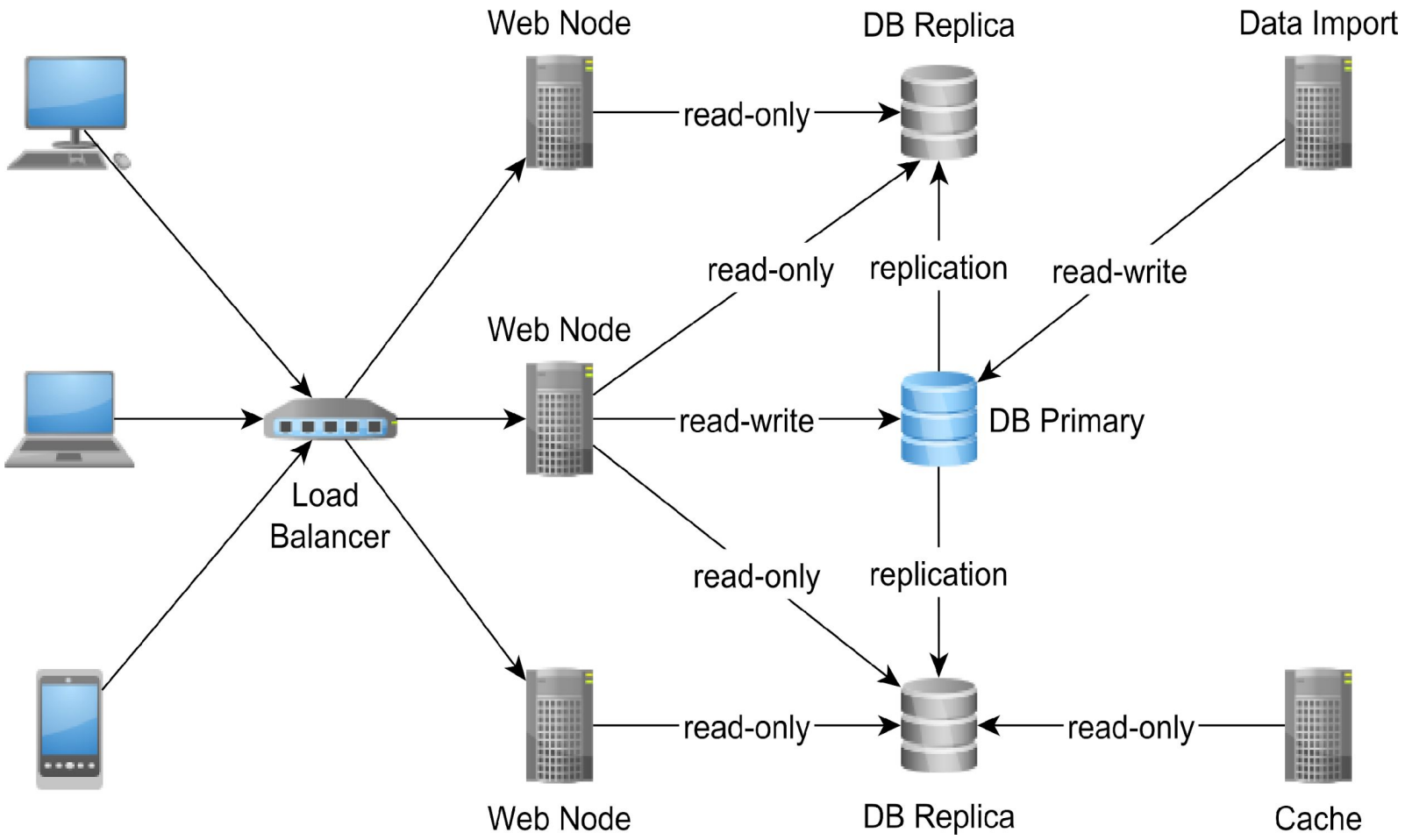


Архітектура розподілених баз даних

Реплікація одноосновної бази даних

- Основний вузол, також відомий як головний вузол, приймає записи, тоді як вузли-репліки можуть обробляти транзакції лише для читання. Наявність єдиного джерела актуальних даних дозволяє нам уникнути конфліктів даних.
- Щоб підтримувати синхронізацію реплік, основні вузли повинні надати список змін, які були внесені всіма зафіксованими транзакціями.

Архітектура розподілених баз даних



Розподілені бази даних

Переваги розподілених баз даних

1. **Модульна розробка —** якщо систему потрібно розширити до нових місць розташування або нових підрозділів у централізованих системах баз даних, дія потребує значних зусиль і порушення існуючого функціонування. Однак у розподілених базах даних робота просто вимагає додавання нових комп'ютерів і локальних даних до нового сайту та, нарешті, підключення їх до розподіленої системи без переривання поточних функцій.

Розподілені бази даних

Переваги РБД (продовження)

2. **Більш надійні** – у разі збою бази даних загальна система централізованих баз даних припиняє роботу. Однак у розподілених системах, коли компонент виходить з ладу, функціонування системи може продовжуватися зі зниженою продуктивністю. Тому DDBMS більш надійна.
3. **Краща відповідь** – якщо дані поширюються ефективним чином, тоді запити користувачів можуть задовольнятися з самих локальних даних, що забезпечує швидшу відповідь. З іншого боку, у централізованих системах усі запити мають проходити через центральний комп'ютер для обробки, що збільшує час відповіді.
4. **Нижча вартість зв'язку** – у розподілених системах баз даних, якщо дані розташовані локально, де вони здебільшого використовуються, тоді витрати на зв'язок для маніпулювання даними можна мінімізувати. Це неможливо в централізованих системах.

Розподілені бази даних

Недоліки розподілених баз даних

1) Потреба у складному та дорогому програмному забезпеченні − DDBMS вимагає складного та часто дорогого програмного забезпечення для забезпечення прозорості даних та координації між кількома сайтами.
2) Накладні витрати на обробку . Навіть прості операції можуть вимагати великої кількості зв’язків і додаткових обчислень, щоб забезпечити однаковість даних на сайтах.
3) Цілісність даних – необхідність оновлення даних на кількох сайтах створює проблеми з цілісністю даних.
4) Накладні витрати на неправильний розподіл даних − швидкість реагування на запити значною мірою залежить від правильного розподілу даних. Неправильний розподіл даних часто призводить до дуже повільної реакції на запити користувачів.

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)

Література

- К. Дж. Дейт. Введение в системы баз данных / An Introduction to Database Systems. — 7-е изд. — «Вильямс», 2001. — ISBN 5-8459-0138-3 (рос.)
- *Oracle Database Administrator's Guide 10g* (Release 1) (Release 1) [Архівовано (Release 1) [Архівовано 27 листопада 2016 у Wayback Machine.]
- Розподілені бази даних (стаття на scientificpapers.org) [Архівовано [Архівовано 15 квітня 2018 у Wayback Machine.](англ.)
- Конспект з розподілених баз даних.Конспект з розподілених баз даних. [АрхівованоКонспект з розподілених баз даних. [Архівовано 24 березня 2017 у Wayback Machine.]
- Datbase SystemsDatbase Systems [АрхівованоDatbase Systems [Архівовано 28 листопада 2016 у Wayback Machine.]
- M. T. Özsu and P. Valduriez, *Principles of Distributed Databases* (3rd edition)

Рекомендована література

- Мейер Д. Теорія реляційних баз даних: пров. з англ. — М., 2005.
- Ревунков Г.І., Самохвалов Е.Н., Чистов В.В. Бази і банки даних і знань. Підручник для вузів / / Під ред. В. Н. Четверікова. — М., 2003.
- Фаронов В.В., Шумаков П.В. Керівництво розроблювача баз даних. — М.: Нолидж, 2000.
- Конноли Т. Базы данных : проектирование, реализация и сопровождение. Теория и практика: учеб. пособие / Томас Конноли, Каролин Бегг. — 3-е изд. ; [пер. с англ.]. — М.: Издательский дом «Вильямс», 2003. — 1440 с. : ил. — Парал. тит. англ.
- Спирли Э. Корпоративные хранилища данных. Планирование, разработка, развитие. / Эрик Спирли. — М.: Издательский дом «Вильямс», 2001. — 400 с.
- Уидом Дж. Введение в системы баз даннях / Джеффри Д. Ульман, Дженнифер Уидом. — М.: Издательство Лори, 2006.— 374 с. ISBN: 5-85582-069-6.