<u>Тема №7:</u> Розподілені обчислення. Модель клієнт - сервер. Сокети. Віддалені методи. Програмування для кластерних систем

Питання:

1. **Основні положення розподіленіх обчислень**
2. **Модель клієнт - сервер. Сокети. Віддалені методи**
3. **Кластерні системи. Програмування для кластерних систем**

Вправи і завдання до теми №7

1. Основні положення розподіленіх обчислень

Паралельні обчислення — це форма обчислень, в яких кілька дій проводяться одночасно. Грунтуються на тому, що великі задачі можна розділити на кілька менших, кожну з яких можна розв'язати незалежно від інших.

Є кілька різних рівнів паралельних обчислень: бітовий, інструкцій, даних та паралелізм задач. Паралельні обчислення застосовуються вже протягом багатьох років, в основному в високопродуктивних обчисленнях, але зацікавлення ним зросло тільки недавно, через фізичні обмеження зростання частоти. Оскільки споживана потужність (і відповідно виділення тепла) комп'ютерами стало проблемою в останні роки, паралельне програмування стає домінуючою парадигмою в комп'ютерній архітектурі, основному в формі багатоядерних процесорів.

Паралельні комп'ютери можуть бути грубо класифіковані згідно з рівнем, на якому апаратне забезпечення підтримує паралелізм: багатоядерність, багатопроцесорність — комп'ютери, що мають багато обчислювальних елементів в межах одної машини, а також кластери, MPP, та ґрід — системи що використовують багато комп'ютерів для роботи над одним завданням. Спеціалізовані паралельні архітектури іноді використовуються поряд з традиційними процесорами, для прискорення особливих задач.

Програми для паралельних комп'ютерів писати значно складніше, ніж для послідовних, бо паралелізм додає кілька нових класів потенційних помилок, серед яких є найпоширеніною стан гонитви. Комунікація, та синхронізація процесів зазвичай одна з найбільших перешкод для досягнення хорошої продуктивності паралельних програм.

Максимальний можливий приріст продуктивності паралельної програми визначається законом Амдала.

2. Модель клієнт - сервер. Сокети. Віддалені методи

Розподілені обчислення являють собою особливий тип програм, побудованих по архітектурі «клієнт-сервер» (*Рис. 7.1.*). Особливість їх полягає в тому, що сам додаток знаходиться і виконується на сервері. Клієнт при цьому одержує тільки результати роботи.

Обчислювальна мережа - це сукупність комп'ютерів і терміналів, з'єднаних за допомогою каналів зв'язку в єдину систему, що задовольняє вимогам розподіленої обробки даних, спільного використання загальних інформаційних і обчислювальних ресурсів.

Розподілені обчислення в комп'ютерних мережах засновані на архітектурі “клієнт-сервер”. Терміни “клієнт” і “сервер” позначають ролі, які відіграють різні компоненти в розподіленому середовищі обчислень.

Компоненти “клієнт” і “сервер” повинні працювати на різних машинах. Додаток - клієнт знаходиться на робочій станції користувача, а сервер - на спеціальній виділеній машині.

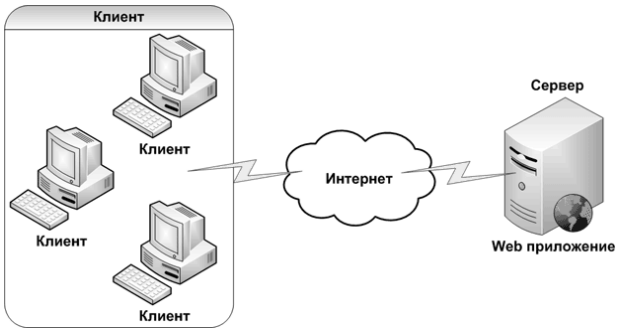

Рис. 7.1. – Клієнт-Серверна архітектура Web-Додатків

Сервер - це спеціальна програма, зазвичай запущена на окремому комп'ютері (хості, від слова host(eng.)), що і виконує якесь коло завдань.

Клієнт, у свою чергу - програма, яка запитує сервер виконати ту або іншу дію (завдання) і повернути отримані дані клієнту.

У цілому алгоритм роботи системи клієнт-сервер виглядає в такий спосіб:

- Сервер підключається до порту і чекає з'єднання із клієнтом;
- Клієнт створює сокет і намагається з'єднати його з портом на хості;
- Якщо створення сокета пройшло успішно, то сервер переходить у режим очікування команд від клієнта;
- Клієнт формує команду і передає її серверу, переходить у режим очікування відповіді;
- Сервер приймає команду, виконує її і пересилає відповідь клієнту.

Сервер створюється в такий спосіб:

```
//=================================================================
// Создать сервер
//=================================================================
private void createServerTask() {
    //====== создать задачу 1
    server_task = new Task<Void>() {
        @Override
        protected Void call() throws Exception {
            try {

                //====== создать сокет сервера
                ServerSocket serverSocket = new ServerSocket(port);

                while (true) {
                    //====== ожидать подключения клиента
                    Socket socket = serverSocket.accept();
                    //====== запустить поток сервера
                    new Thread(new Server(socket)).start();

                }

            } catch (IOException ex) {
                ex.printStackTrace();
                server_task.cancel();
            }
            return null;
        }
    };
}
```

Сокет клієнта створюється в такий спосіб:

```
protected Void call() throws Exception {
    try {
        //====== создать сокет клиента
        try {
            socket = new Socket(host, port);
            flag_connected = true;
            //====== вывод сообщения Connected
            Platform.runLater(()
                    -> textArea.appendText(">>> Connected: "
                            + "\nPort: " + port
                            + "\nHost: " + host + '\n'
                    ));
        } catch (IOException ex) {
            flag_connected = false;
            //====== вывод сообщения Connected
            Platform.runLater(()
                    -> textArea.appendText("== Connection failed: "
                            + "\nPort: " + port
                            + "\nHost: " + host + '\n'
                    ));
            client_task.cancel();
            return null;
        }
```

1. Кластерні системи. Програмування для кластерних систем

Характеристики кластерної системи наведені в табл.7.1.

Таблиця 7.1. Характеристики кластерних систем

Архітектура	Набір робочих станцій (чи ПК) загального призначення, використовується як дешевий варіант масивно – паралельного комп'ютера. Для зв'язку вузлів використовується одна з стандартних мережних технологій (Fast/ Gigabit Ethernet, Myrinet) на базі шинної чи архітектури комутатора. При об'єднанні в кластер комп'ютерів різної потужності чи різної архітектури, говорять про гетерогенні (неоднорідні) кластери. Вузли кластера можуть одночасно використовуватися як користувацькі робочі станції.
Приклади	NT - кластер у NCSA , Beowulf - кластери.
Операційна система	Використовуються стандартні для робочих станцій ОС, найчастіше, вільно розповсюджувані - Linux/FreeBSD, разом зі спеціальними засобами підтримки паралельного програмування і розподілу навантаження.
Модель програмування	Програмування, як правило, у рамках моделі передачі повідомлень (найчастіше - MPI). Дешевизна подібних систем обертається великими накладними витратами на взаємодію паралельних процесів між собою, що сильно звужує потенційний клас розв'язуваних задач.

Для організації локальної мережі між обчислювальними вузлами кластера існує доступна та широко розповсюджена технологія Gigabit Ethernet. Вона повністю задовольняє вимогам до побудови мережі обміну за протоколом Message Passing Interface (MPI) як за пропускною здатністю, так і за параметрами затримок. Прогнозується, що в 2008 році набуде широкого розповсюдження стандарт 10 Гбіт Ethernet. При значеннях затримок 5–10 мкс він може успішно конкурувати зі спеціалізованою технологією Infiniband (високошвидкісна комутована послідовна шина, призначена для внутрішньо- та міжсистемних з'єднань [4]),

затримки останніх реалізацій якої лежать у межах 1–5 мкс. Крім того, в 2010 році прогнозується впровадження технології 100 Гбіт Ethernet. На сьогоднішній день більше 40 % кластерів, що входять у Top500, використовують Ethernet [5].

При створенні кластера, в якому необхідно передбачити процес тимчасового призупинення (в робочий час організацій, коли комп'ютери використовуються для інших цілей), виникає проблема призупинення розрахунків, які виконуються на кластері. Відсутність можливості тимчасової зупинки розрахунків є недоліком більшості традиційних кластерних систем.

Для об'єднання персональних комп'ютерів в єдину обчислювальну систему та побудови обчислювального кластеру на основі персональних комп'ютерів з використанням програмного забезпечення Microsoft Windows Compute Cluster Server 2003 (WCCS) [6] в Київському національному університеті імені Тараса Шевченка був розроблений програмний комплекс UACluster.

Стратегічна перспектива проекту базується на широкому розповсюдженні технологій TCP/IP Offload

Engine (TOE) та Remote Direct Memory Access (RDMA) (апаратна обробка мережевих протоколів, запис даних, отриманих мережевою картою, безпосередньо в пам'ять без участі процесора), що дозволить звільнити процесор від навантаження по передачі даних. Тенденції розвитку технологій оперативної пам'яті є такими, що її продуктивність підвищують шляхом розширення шини для передачі більшого обсягу даних при майже незмінних затримках. На фоні постійного зменшення затримок у мережі, можна говорити про можливість в майбутньому побудови кластерів з загальною пам'яттю (на основі технології OpenMP). Це приведе до усунення необхідності побудови окремих кластерних обчислювальних систем, достатньо буде об'єднати існуючі ресурси.

Принципи роботи кластеру

Введемо у розгляд сукупність працюючих в єдиному комунікаційному середовищі персональних комп'ютерів організації та визначимо два можливих варіанта їх роботи.

Режим **1**. Функціонування у складі кластеру.

Режим **2**. Функціонування локальних комп'ютерів, що забезпечують необхідні дії групи різних користувачів в складі мережі.

В неробочий для співробітників час (вночі, вихідні та святкові дні) комп'ютери організації можна об'єднати в кластер, тобто перевести їх у режим **1** та використовувати для розрахунку різноманітних задач. Для побудови кластера запропоновано використання програмного забезпечення WCCS, що працює під керуванням операційної системи (ОС) Windows 2003 Server (x64) (WS2003).

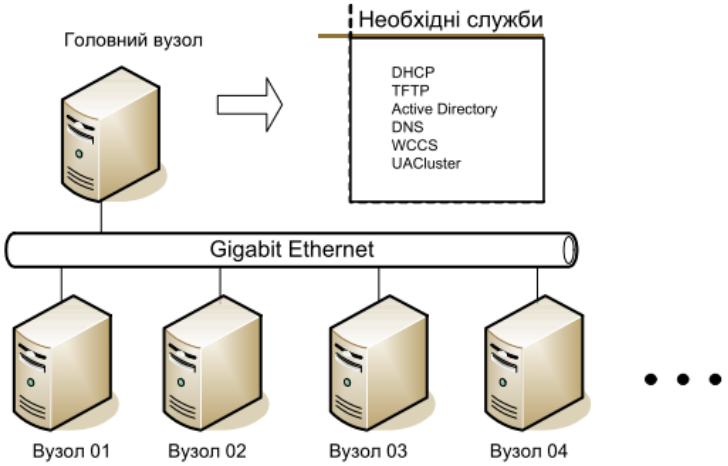

Рис. 7.2 -Організаційна структура кластера

Крок **4.** Після закінчення інсталяції еталонного вузла необхідно зняти образ жорсткого диска цього вузла та розмножити його на всі інші комп'ютери в системі. Гомогенність вузлів є основною вимогою для реалізації методу. За рахунок тиражування образу жорсткого диска копія MBR-запису для завантаження клієнтської та

кластерної ОС, яка знята з еталонного вузла, може бути в подальшому використана для всіх інших вузлів.

В результаті виконання цих кроків вигляд MBR-запису жорсткого диска буде визначати, яка саме ОС буде завантажена на певному комп'ютері. Переваги цього методу полягають у тому, що керування процесом завантаження відбувається незалежно від ОС, яка використовується. Лишається тільки створити механізм автоматичної заміни MBR.

Створений програмний комплекс UACluster (деталі створення та роботи програмного комплексу описано на сайті проекту UACluster) складається з двох програм: програма керування завантажувальними записами (ПКЗЗ), пункт керування вузлами (ПКВ).

Рис. 7.3 - Структура розділів жорсткого диска

Заміна MBR відбувається до етапу завантаження ОС. Для цього використовується ПКЗЗ, яка завантажується по мережі та працює в середовищі Preboot eXectuion Environment [9] (PXE – середовище для завантаження комп'ютера по мережі без використання жорсткого диску).

Для керування завантаженням з використанням середовища PXE з боку клієнта необхідна робота двох серверних компонент Dynamic Host Configuration Protocol (DHCP) та Trivial File Transfer Protocol **(**TFTP**)** серверу, на якому зберігається ПКЗЗ та дві копії MBR-розділу жорсткого диска одного з вузлів кластеру. Функції середовища PXE полягають у зверненні до DHCP, отримані параметри налаштування стеку протоколів TCP/IP та розташування завантажувального файлу (ПКЗЗ). Далі, PXE завантажує ПКЗЗ з TFTP-серверу та передає їй керування.

Інша проблема, яку необхідно вирішити для побудови запропонованої концепції створення кластера, полягає в наступному. Перед перемиканням кластера в режимі **2**, тобто на клієнтську ОС, необхідно тимчасово призупинити всі розрахунки. Для цього, в найпростішому випадку, має вистачити переведення системи до сплячого стану (hibernate). При переведенні системи в такий стан, процеси та задачі, які виконувались, зберігаються на жорсткому диску, а система вимикається. Але на практиці виявилось, що такий механізм зупинки задач спрацьовує не завжди. Тільки після переведення всіх задач одночасно на всіх вузлах до стану паузи (suspend), а ОС до сплячого стану, можна виконати надійну тимчасову зупинку розрахунків. Після перезавантаження кластера (перемикання в режим **1**) задачі необхідно

відновити (resume). Для зупинки задач та відновлення їхньої роботи було використано функції psexec та pssuspend утиліти psTools, яка є частиною пакета Windows Sysinternals [10]. Для перевірки надійності процесу тимчасової зупинки задач був використаний пакет Intel MBI Benchmarks .

Процес завантаження кластера керується програмним комплексом UACluster, який встановлено на головному вузлі. В заздалегідь запланований час, коли комп'ютери в режимі **2** вже не використовуються і вимкнені (кінець робочого дня), головний вузол (він весь час залишається ввімкненим) автоматично переводить всі вузли в режим **1**. На кожному з вузлів цей процес складається з таких кроків (рис.).

Крок **1**. ПКВ – спеціально створена програма мовою C#, у встановлений час вмикає по мережі вузол, використовуючи технологію Wake-On-Lan (WOL – технологія, яка дозволяє ввімкнути комп'ютер дистанційно, відправивши пакет, сформований спеціальним чином).

Крок **2**. Використовуючи мережеве завантаження на стороні клієнта, PXE звертається до DHCP-серверу для отримання параметрів налаштування протоколу TCP/IP: IP-адресу, мережеву маску та IP-адресу шлюзу. Крім налаштувань мережевого інтерфейсу DHCP також повертає параметр boot filename (опція 067), в якому зазначено який саме файл необхідно використовувати для завантаження системи. Далі, PXE намагається прочитати з TFTP-серверу файл, ім'я якого було зазначено в полі boot filename (ПКЗЗ) (перші чотири етапи, рис. 7.4).

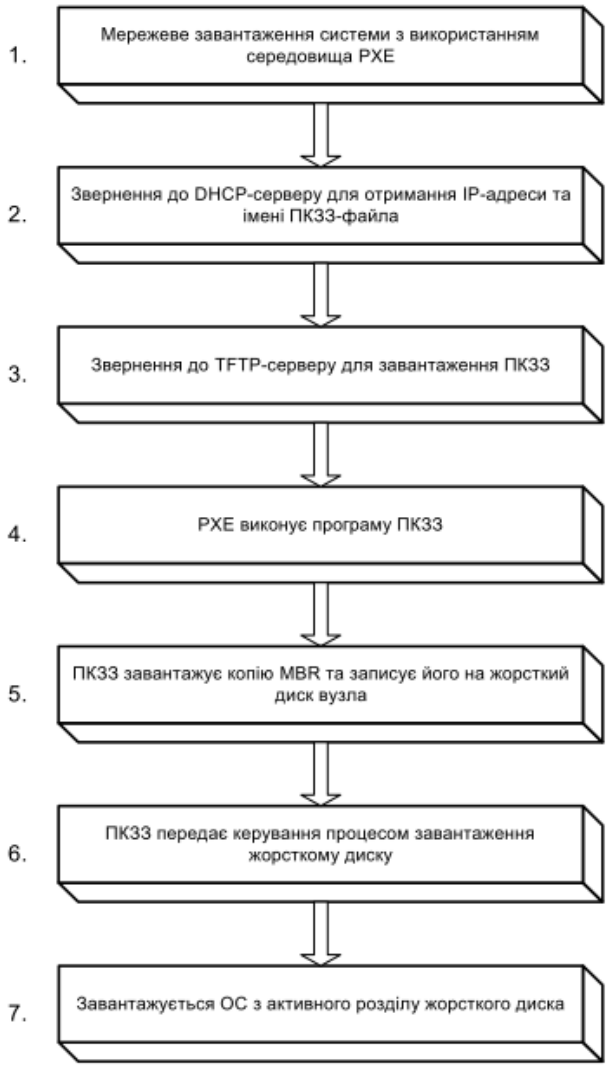

Рис. 7.4 - Завантаження вузла кластера

Крок **3**. ПКЗЗ змінює MBR-запис жорсткого диска на кластерний (п'ятий етап, рис. 3), внаслідок чого розділ, на якому знаходиться кластерна ОС, стає активним (кластерний MBR).

Крок **4**. ПКЗЗ передає керування завантаженням жорсткому диску (шостий етап, рис. 3).

Крок **5**. На жорсткому диску завантажується кластерна ОС, вона знаходиться на активному розділі

(сьомий етап, рис. 3).

Крок **6**. ПКВ відновлює всі задачі, які не встигли виконатися минулого разу. Якщо, таких задач немає, WCCS розпочинає розрахунки нових.

Перед початком нового робочого дня, у встановлений час ПКВ зупиняє всі розрахунки, переводячи їх до стану suspend, а кластер до стану hibernate, та вимикає всі вузли, використовуючи дистанційні команди. Далі при ввімкнені будь-якого з вузлів завантаження відбувається як і в режимі **1** (кроки 2-4). З тією відмінністю, що на жорсткий диск записується клієнтська копія MBR (рис. 2). Розділ з клієнтською ОС буде в цьому випадку активним, внаслідок чого вона завантажується.

Метод із заміною MBR забезпечує додатковий рівень надійності, навіть за втратою зв'язку з головним вузлом зберігається можливість локального завантаження ОС.

При необхідності кластерна частина жорсткого диска під час роботи клієнтської ОС може виглядати як вільне місце на диску або специфічна файлова система, тобто вона прихована від користувача і не може бути ушкодженою без прав адміністратора.

Вправи і завдання до теми №7

1. Мережа Клоса лише за умови N > 24 економічно вигідніша, ніж перехресний шинний розподілювач. Доведіть справедливість цього твердження за допомогою правила і із завдання № 5.
2. Якими параметрами характеризується комутаційна мережа обчислювальної системи?
3. Наведіть приклад статичної топології комутаційної мережі. Охарактеризуйте її.
4. Наведіть приклад динамічної топології комутаційної мережі. Охарактеризуйте її.
5. Комутаційна решітка комп'ютера має розміри 3x4x5. Вкажіть безпосередніх сусідів вузла, який розміщений на вершині (на верхній грані, на ребрі).
6. Чому спільну шину не використовують для об'єднання великої кількості процесорів?
7. Відомо, що алгоритм добре відображається на топологію "кільце". Чи можна гарантувати його хороше відображення на топологію "гіперкуб"? Відповідь обгрунтуйте.