Тема 1: Базові поняття і терміни комп'ютерних систем

Дамо базове поняття системи обробки даних (СОД). СОД - сукупність технічних засобів і програмного забезпечення, призначена для інформаційного обслуговування користувачів і технічних об'єктів. До складу технічних засобів входить обладнання для введення, зберігання, перетворення і виведення даних. Програмне забезпечення (програмні засоби) - сукупність програм, що реалізують покладені на системи функції.

Комп'ютер - пристрій в якому об'єднані процесор, основна пам'ять і пристрої вводу-виводу.

Комп'ютерна система - СОД, налаштована на вирішення завдань конкретної області застосування. КС утворюється шляхом об'єднання комутаційним середовищем сукупності процесорів, блоків пам'яті і пристроїв вводу-виводу.

В системах керування реальними об'єктами, побудованих на основі КС, процес керування зводиться до вирішення функціонального набору завдань $A = \{A_1, ..., A_m\}$. Кожне завдання ініціюється або періодично, або при виникненні певних ситуацій в системі.

Режим при якому організація обробки даних підпорядковується темпу процесів поза СОД, називається обробкою в реальному масштабі часу (РМЧ). Особливість функціонування КС реального часу в тому, що КС має працювати в темпі з процесом, інформація про який автоматично надходить в КС і обробляється.

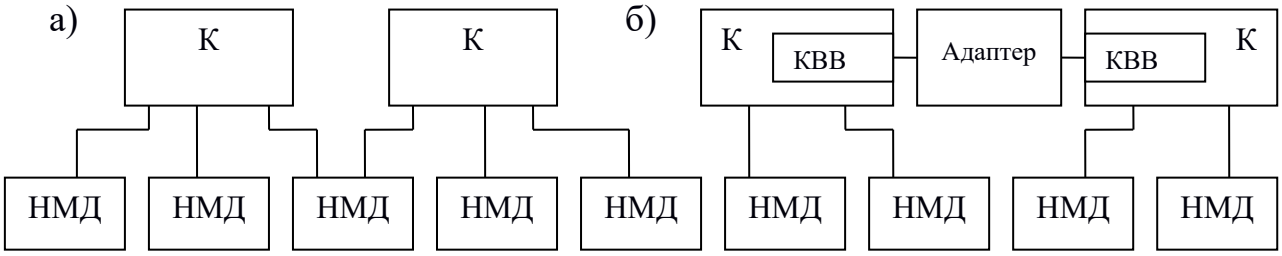
Обробка в РМЧ забезпечується за рахунок: вибору структури СОД і швидкодії пристроїв відповідно до завдань обробки *A* і вимогами до часу обробки; способів організації процесів обробки, що забезпечують необхідний час відповіді при обмеженій продуктивності пристроїв і заданої структурі СОД.

Високопродуктивна комп'ютерна система - утворюється шляхом об'єднання комутаційним середовищем сукупності високопродуктивних

процесорів, блоків пам'яті і пристроїв вводу-виводу. При цьому процесори можуть бути як однаковими, так і різними, спеціалізованими на виконання певних функцій.

Починаючи з 60-х років для підвищення продуктивності і надійності СОД кілька комп'ютерів зв'язувалися між собою, утворюючи багатомашинну комп'ютерну систему (БМКС).

В ранніх БМКС зв'язок між комп'ютерами забезпечувався через загальні зовнішні запам'ятовуючі пристрої - накопичувачі на магнітних дисках (НМД) або магнітних стрічках (рис. 1.1, а), тобто за рахунок доступу до загальних наборів даних. Такий зв'язок називається непрямим і виявляється ефективним в тому випадку, коли комп'ютери взаємодіють досить рідко, наприклад при відмові одного з комп'ютерів або в моменти початку і закінчення обробки даних.



К - комп'ютер, КВВ - канал вводу-виводу.

Рис. 1.1. Багатомашинна обчислювальна система з непрямим (а) і прямим (б) зв'язком між комп'ютерами.

Більш оперативна взаємодія комп'ютерів досягається за рахунок прямого зв'язку через адаптер, що забезпечує обмін даними між каналами вводу-виводу (КВВ) двох комп'ютерів (рис. 1.1, б) і передачу сигналів переривання.

У БМКС взаємодія процесів обробки даних забезпечується тільки за рахунок обміну сигналами переривання і передачі даних через адаптери канал-канал або загальні зовнішні пристрої, що запам'ятовують.

Комп'ютерна система, що містить кілька процесів із загальною оперативною пам'яттю і периферійними пристроями називається багатопроцесорною. Принцип побудови таких систем показано на рис. 1.2.

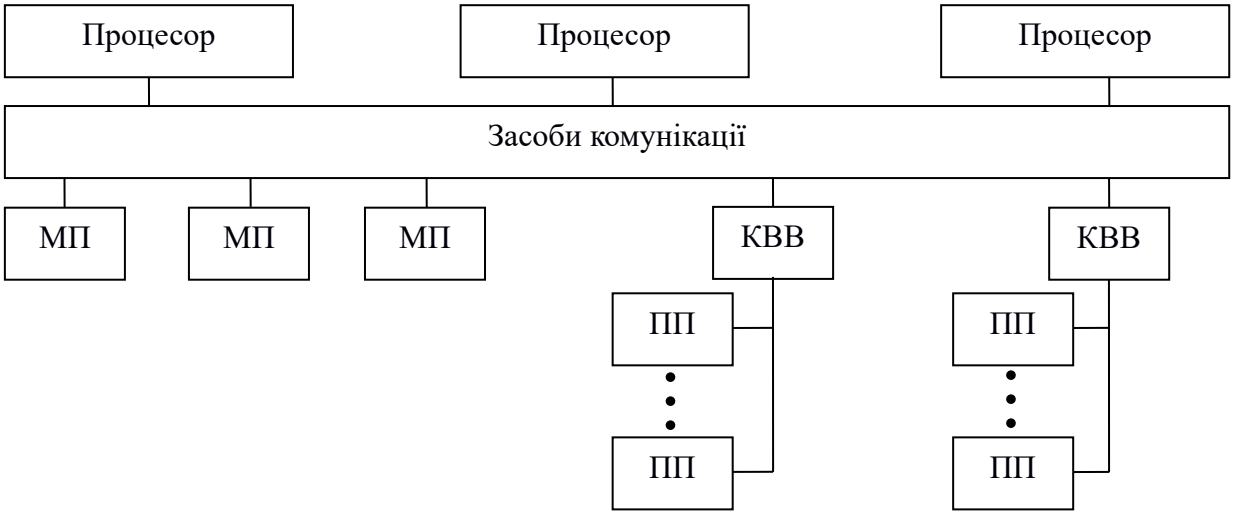


Рис. 1.2. Структурна схема багатопроцесорної комп'ютерної системи

Процесори, модулі оперативної пам'яті (МП) і канали вводу-виводу, до яких підключені периферійні пристрої (ПП), об'єднуються в єдиний комплекс за допомогою засобів комутації. Засоби комутації забезпечують доступ кожного процесора до будь-якого модулю оперативної пам'яті і каналу вводу-виводу, а також можливість передачі даних між останніми. Багатопроцесорні комп'ютерні системи (БПКС) мають більшу стійкість до відмов.

КС включає в себе технічні засоби та програмне забезпечення, спрямовані на рішення визначеної сукупності завдань. Існує два способи спрямування: 1) КС може будуватися на основі комп'ютерного комплексу загального застосування і спрямування системи забезпечується за рахунок програмних засобів - прикладних програм (або ОС). 2) Спрямування на заданий клас завдань може досягатися за рахунок використання

спеціалізованих комп'ютерів. Спеціалізовані КС найбільш широко використовуються при вирішенні завдань векторної і матричної алгебри, при інтегруванні диференціальних рівнянь, задач, пов'язаних обробкою зображень, розпізнаванням образів і т.д.

В останні роки розробляють адаптивні КС, вони вважаються найбільш перспективними з позиції забезпечення відмовостійкості і продуктивності. Адаптація таких КС з метою пристосування їх до структури реалізованого алгоритму досягається за рахунок зміни конфігурації системи.

Основними причинами появи КС є:

- необхідність підвищення продуктивності і надійності обчислювальних засобів;
- високі вимоги до достовірності інформації, що оброблюється;
- необхідність поліпшення експлуатаційних властивостей обчислювальних засобів.

Ці якості вигідно відрізняють КС від однопроцесорних комп'ютерів.

Основними характеристиками КС є ефективність, продуктивність, час відповіді, надійність і вартість. На додаток до них застосовуються такі характеристики: габарити, маса, споживана потужність, діапазон робочих температур, ремонтопридатність та ін.

Характеристики залежать від організації системи - структури, складу програмного забезпечення, режиму функціонування системи та ін. В математичному аспекті характеристики можна розглядати як найменування функцій, аргументами яких є параметри.

Розглянемо способи оцінки основних характеристик КС і набори параметрів, що впливають на характеристики.

Реальна ефективність конкретної обчислювальної системи визначається часом перебування системи в працездатному стані і використанням її за призначенням.

Критерієм *ефективності* є відношення продуктивності до вартості

$$E = W/C,$$

де Е – ефективність,

W – продуктивність КС,

C – вартість,

так як реальна продуктивність КС дорівнює

$$W_p = W_0 * K_в,$$

де W_0 – продуктивність КС при ідеальній надійності,

$K_в$ - коефіцієнт технічного використання.

то

$$E = K_в * W_0/C$$

Продуктивність – характеристика обчислювальної потужності системи, яка визначає кількість обчислювальної роботи, що виконується за одиницю часу. Існує два способи визначення W:

1. Нехай за час T система завершила обробку n задач (завдань). Тоді продуктивність за час T становить (для систем, що знаходяться в експлуатації)

$$W = n / T \qquad (1.1)$$

Продуктивність оцінюється числом завдань розв'язуваних системою за одиницю часу: W (завдань/год.).

2. Інший спосіб визначення продуктивності W – через середнє значення інтервалу між моментами закінчення обробки завдань. У цьому випадку протягом часу T реєструються інтервали між моментами завершення завдань $\tau_1, ..., \tau_n$. Середнє значення цього інтервалу

$$\tau = \frac{1}{n} \sum_{i=1}^{n} \tau_i$$

визначається інтенсивністю вихідного потоку завдань, і продуктивність системи

$$W = 1 / \tau \qquad (1.2)$$

Оцінки продуктивності (1.1) і (1.2) збігаються, якщо початок і кінець проміжку часу T збігаються з моментами закінчення обробки завдань.

На продуктивність найбільш суттєво впливають наступні параметри:

- число і швидкодія пристроїв, ємність оперативної і зовнішньої пам'яті, зі збільшенням яких продуктивність може зростати, а також структура системи і пропускна здатність зв'язків між елементами системи;
- режим обробки завдань і робоче навантаження, в першу чергу обсяг введених, збережених в пам'яті, виведених даних і число процесорних операцій, необхідних для виконання завдання.

Наприклад, оцінка продуктивності багатомашинної КС визначається через номінальні швидкодії машин, що входять в її склад, називається номінальною (розрахунковою). За визначенням

$$W_H = k(M) \sum_{i=1}^{M} B_{\text{Hi}}$$

де B_{Hi} – номінальна швидкодія і-ї машини;

M – число ЕОМ в системі;

$k(M)$ – системні витрати.

Номінальна швидкодія B_H – це кількість стандартних операцій (сама «коротка» - операція додавання), виконуваних машиною за одиницю часу при реалізації деякої програми, складеної тільки зі стандартних операцій:

$$BH = 1 / \tau_{cm}$$

де τ_{cm} - час, що припадає на виконання однієї стандартної операції.

Продуктивність системи, яка визначається сумою ефективної швидкодії машин КС, називається ефективної

$$W_e = k\,(M) \sum_{i=1}^{\grave{I}} B_{\text{åi}}$$

де B_{ei} – ефективна швидкодія *i*-ї машини

Ефективна швидкодія машини визначається числом стандартних операцій, виконуваних машиною в одиницю часу з урахуванням всіх втрат часу (введення, виведення та ін.).

Час відповіді. Час відповіді, інакше час перебування задач (завдань) в системі, тривалість проміжку часу від моменту надходження завдання в систему до моменту закінчення його виконання.

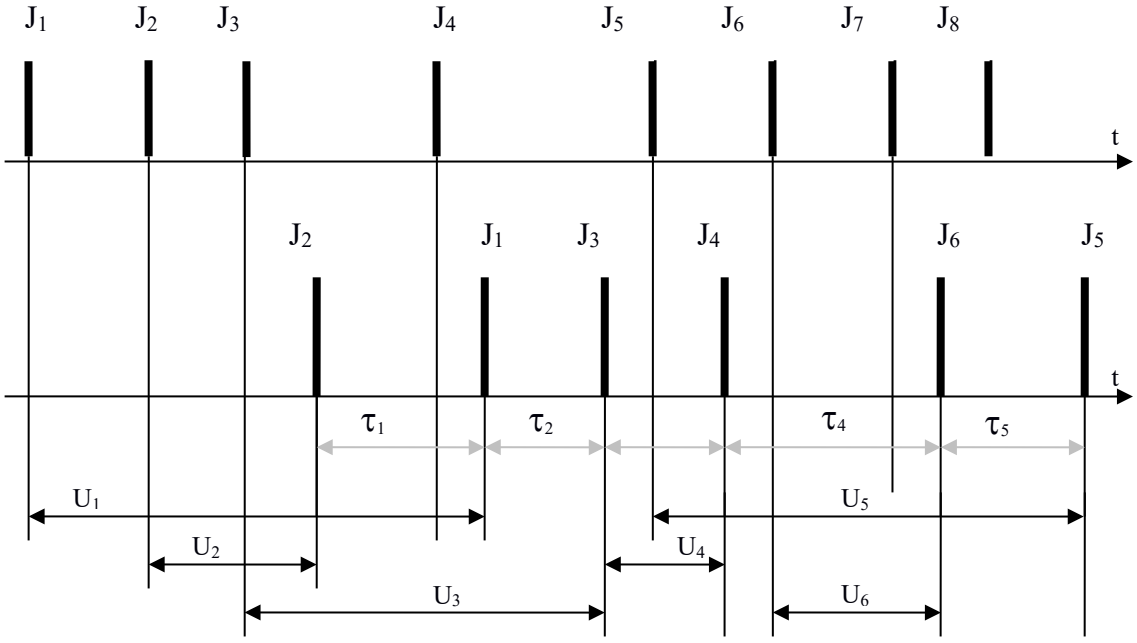


Рис. 1.3. Потоки завдань на вході і виході системи.

На рис. 1.3 зазначено час відповіді U_1, U_2 ... для завдань J_1, J_2 ... відповідно.τ_1/τ_n - інтервали між моментами завершення обробки завдань.

Час відповіді - випадкова величина, що обумовлена наступними факторами:

- впливом вихідних даних на число операцій введення, обробки і виведення даних і непередбачуваністю значень вихідних даних;
- впливом складу сукупності завдань, що одночасно знаходяться в системі, і непередбачуваністю складу сукупності через випадковість моменту надходження завдань на обробку.

Найчастіше час відповіді оцінюється середнім значенням, яке визначається як статистичне середнє випадкової величини U_i, $i = 1, ..., n$:

$$U = \frac{1}{n} \sum_{i=1}^{n} u_i$$

Час відповіді складається з двох складових: часу виконання завдання і часу очікування.

Час виконання завдання при відсутності паралельних процесів рівне сумарній тривалості всіх етапів процесу - введення, звернення до зовнішньої пам'яті, процесорної обробки і виведення.

Час виконання завдання залежить від складності обчислень $\theta_1,...,\theta_N$ і швидкодії $V_1, ..., V_N$ пристроїв $1, ..., N$:

$$V = \sum_{i=1}^{N} \theta_i / V_i ,$$

де, θ_i - складність обчислень,

V_i - швидкодія.

Час очікування – сума проміжків часу, протягом яких завдання перебувало в стані очікування необхідних ресурсів. Очікування виникає при мультипрограмній обробці, коли ресурс, необхідний задачі, зайнятий іншим завданням і перше завдання не виконується, чекаючи звільнення ресурсу. Час очікування залежить від режиму обробки завдань і інтенсивності вхідного потоку задач.

Час відповіді залежить від тих же параметрів, що і продуктивність: структури і характеристик технічних засобів, режиму обробки та характеристик завдань.

Таким чином, середній час відповіді характеризує швидкість реакції системи на вхідні дії: завдання, запити абонентів і т.п. Якість систем тим вище, чим менше середній час відповіді.

Характеристики надійності. Надійність – властивість системи виконувати покладені на неї функції в заданих умовах функціонування із

заданими показниками якості: достовірністю результатів, пропускною спроможністю, часом відповіді та ін.

Працездатність системи або окремих її частин порушується через відмови апаратури – виходу з ладу елементів або з'єднань.

Найважливіша характеристика надійності – інтенсивність відмов, що визначає середнє число відмов за одиницю часу (за одну годину).

Інтенсивність відмов залежить від числа елементів і з'єднань, що складають систему. Якщо будь-яка відмова носить катастрофічний характер, тобто призводить до порушення працездатності системи, то інтенсивність відмов в системі

$$\lambda_{\tilde{N}} = \sum_{i=1}^{n} \lambda_i \ ,$$

де λ_i - інтенсивність відмов *i*-го елемента або з'єднання і *n* - число елементів і з'єднань в системі.

Середній проміжок часу між двома суміжними відмовами називається середнім напрацюванням на відмову і дорівнює $t_{ср} = 1 / \lambda_c$.

Проміжки часу між відмовами – випадкові величини із середнім значенням $t_{ср}$, які розподілені за експоненціальним законом.

Працездатність системи, порушена в результаті відмови, відновлюється шляхом ремонту системи.

Ремонт полягає у виявленні причини порушення працездатності – діагностиці системи і у відновленні працездатності шляхом заміни несправного елемента.

Проміжок часу, що витрачається на відновлення працездатності системи, називається часом відновлення. Час відновлення – випадкова величина, яка характеризується середнім значенням t_B - середнім часом відновлення.

З урахуванням середнього напрацювання на відмову $t_{ср}$ і середнього часу відновлення t_B надійність системи характеризується коефіцієнтом готовності.

$$K_г = t_{ср} / (t_{ср} + t_B),$$

визначальним частку часу, протягом якого система працездатна.

Значення $1 - K_г$ являє собою частку часу, протягом якого система непрацездатна, ремонтується. Так, якщо $K_г$ = 0,95, то 95% часу система працездатна і 5% часу витрачається на її ремонт.

Крім того, $K_г$ визначає ймовірність того, що в довільний момент часу система працездатна, а значення $1 - K_г$ – ймовірність того, що в цей момент часу система знаходиться в стані відновлення.

Надійність системи може бути підвищена за рахунок резервування її елементів – дублювання; троювання і т.д.

Однак резервування призводить до збільшення вартості системи.

Вартість. Вартість КС - це сумарна вартість технічних засобів та програмного забезпечення.

Вартість технічних засобів визначається їх складом і технічними характеристиками. Пристрої з більш високими технічними характеристиками – швидкодією, ємністю, надійністю – мають більш високу вартість.

Вартість програмного забезпечення визначається в основному витратами на розробку програм і тиражованих програм числом систем, в яких використовуються програми. Витрати на розробку програм найбільш істотно залежать від складності програм. Вартість КС впливає на вартість рішення задачі, яка визначається вартістю ресурсів, використовуваних завданням:

$$S = \sum_{i=1}^{N} C_{\text{i}} \cdot \theta_i$$

C_i - вартісний коефіцієнт, який визначає вартість використання одиниці ресурсу і (мільйона процесорних операцій, кілобайти пам'яті та ін.), і θ_i - обсяг ресурсу і, який використовується завданням.

Тема 2: Паралельна обробка інформації в комп'ютерних системах

Паралелізм в обробці функціональних блоків комп'ютерної системи забезпечується за рахунок конвеєрного принципу обробки інформації, введення додаткових процесорів, збільшення видів оперативної пам'яті і розширення шинних каналів обміну.

Розглянемо деякі аспекти досягнення паралелізму в комп'ютерних системах.

Основним методом збільшення продуктивності КС є організація паралельної обробки інформації.

Під розпаралелюванням розуміють одночасну обробку двох або більше завдань (алгоритмів, команд, мікрокоманд і т.д.) або поєднання в часі кількох етапів розв'язання однієї задачі.

Існують кілька способів організації розпаралелювання (або паралельної обробки) основними з яких є:

- суміщення в часі різних етапів різних завдань;
- одночасне рішення різних завдань або частин однієї задачі;
- конвеєрна обробка інформації.

Перший шлях - поєднання в часі етапів вирішення різних завдань - це мультипрограмна обробка інформації. Вона можлива навіть в однопроцесорній ЕОМ і широко використовуються в сучасних СОД.

Другий шлях - одночасне рішення різних завдань або частин однієї задачі - можливий тільки при наявності декількох оброблювальних пристроїв.

Залежно від того, чи здійснюється одночасна обробка декількох частин одного завдання або відразу декількох завдань, розрізняють наступні типи розпаралелювання:

- природний паралелізм незалежних завдань;
- паралелізм незалежних гілок;
- конвеєрний режим роботи.

Природний паралелізм незалежних завдань полягає в тому, що в систему надходить безперервний потік не пов'язаних між собою завдань, тобто рішення будь-якої задачі не залежить від результатів вирішення інших завдань.

У цьому випадку використання декількох обробних пристроїв при будь-якому способі комплексування (непрямому або прямому) підвищує продуктивність системи.

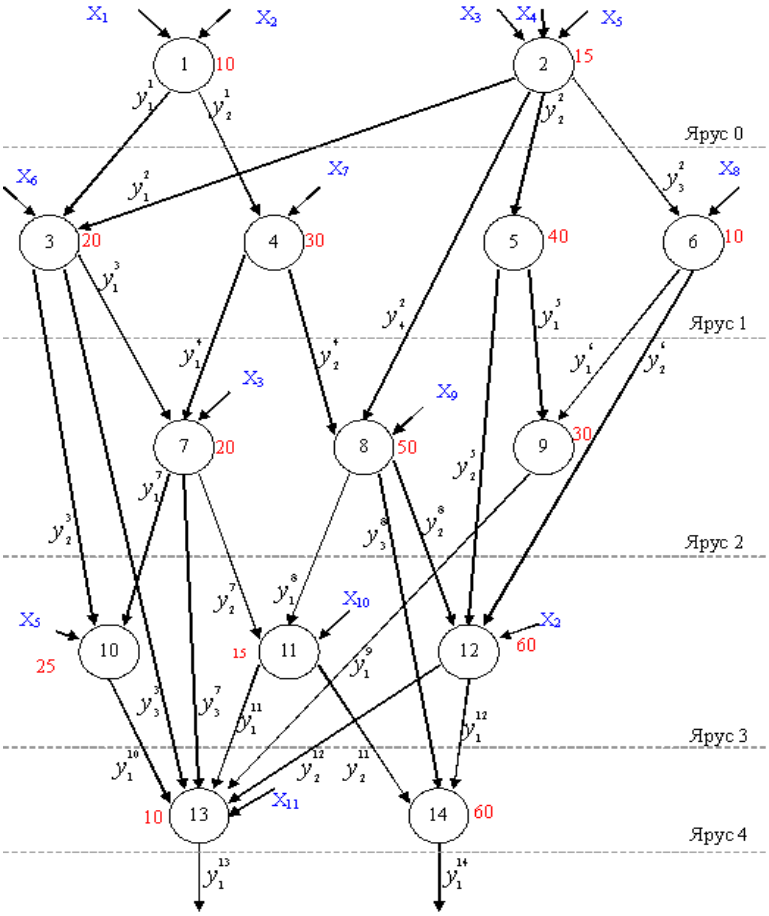
Максимальна продуктивність може бути досягнута при однаковому часі виконання всіх частин завдання. Обсяг обладнання при цьому зростає більш ніж в *n*-разів у порівнянні з варіантом без розпаралелювання.

Розглянемо суть паралелізму незалежних гілок, як однієї з найбільш поширених типів паралелізму в обробці інформації. Суть його полягає в тому, що при вирішенні великого завдання можуть бути виділені окремі частини - гілки програми, які при наявності декількох оброблювальних пристроїв можуть виконуватися паралельно і незалежно один від одного.

Двома незалежними гілками програми будемо вважати такі частини завдання, при вирішенні яких виконуються наступні умови:

- жодна з вхідних для гілки програми величин не є вихідною величиною іншої програми (відсутність функціональних зв'язків);
- для обох гілок програми не повинні робитися записи в одні і ті ж елементи пам'яті (відсутність зв'язку з використання одних і тих же полів оперативної пам'яті);
- умови виконання однієї гілки не залежать від результатів або ознак, отриманих при виконанні іншої гілки (незалежність з керування);
- обидві гілки повинні виконуватися по різних блоках програми (програмна незалежність).

Прикладом представлення про паралелізм незалежних гілок є ярусно-паралельна форма програми. Ярусно-паралельна форма програми приведена на рис. 2.1.



t_i - число часових одиниць. *14 - гілок*

i - гілки (довжина гілки) *5 - ярусів*

Рис. 2.1. Ярусно-паралельна форма програми

Програма представлена у вигляді сукупності гілок, розташованих на декількох рівнях-ярусах.

Кружками з цифрами всередині позначені гілки. Довжина гілки представляється цифрою, що стоїть біля кружка. Стрілками показані вхідні дані і результати обробки. Вхідні дані позначаються символом *x*, вихідні дані символом - *y*. Символи *x* мають нижні цифрові індекси, які означають номери вхідних величин;

Символи *y* мають цифрові індекси і внизу і вгорі: цифра вгорі відповідає номеру гілки, при виконанні якої отримано даний результат, а

цифра внизу означає порядковий номер результату, отриманого при реалізації даної гілки програми.

Зображена на рисунку 2.1 програма містить 14 гілок, розташованих на 5 ярусах. Гілки кожного ярусу не пов'язані одна з одною, тобто результати вирішення будь-якої гілки даного ярусу не є вхідними даними для іншої гілки цього ж ярусу.

На цьому ж графі можуть бути зображені і зв'язку по управлінню або пам'яті.

Прикладом, що довжина i-ї гілки представляється число тимчасових одиниць t_i, які потрібно для її виконання. Тоді для виконання всієї програми буде потрібно $\dot{O} = \sum_{i=1}^{N=14} t_i = 375$ одиниць часу. Якщо уявити, що програма виконується двома обробними пристроями, що працюють незалежно один від одного, то час виконання завдання скорочується. Можливі кілька варіантів виконання програми.

Таким чином, для того щоб за допомогою декількох обробних пристроїв вирішити завдання, що має незалежні паралельні гілки, необхідна відповідна організація процесу, яка визначає шляхи вирішення завдання і виробляє необхідну інформацію про готовність кожної гілки. При вирішенні задачі кожен процесор перед початком виконання черговий гілки повинен мати інформацію про готовність даних для цього.

При вирішенні багатьох складних завдань тільки програмування з виділенням незалежних гілок дозволяє істотно скоротити час вирішення. Добре піддаються паралельний обробці сигналів, прямі і зворотні перетворення Фур'є та інші.

Складність: організація оптимального графіка роботи і виділення незалежних гілок при розробці програм.

Паралелізм об'єктів або даних має місце тоді, коли по одній і тій же програмі повинна оброблятися деяка сукупність даних, що надходять в систему одночасно.

Це можуть бути завдання векторної алгебри - операції над векторами і матрицями, що характеризуються деякою сукупністю чисел. Рішення завдання при цьому зводиться до виконання однакових операцій над парами чисел двох аналогічних об'єктів. Так, наприклад, додавання двох матриць розмірністю $m_x n$ полягає в додаванні відповідних елементів цих матриць:

$$A + B = [a_{ik}] + [b_{ik}] = [a_{ik} + b_{ik}]$$

При цьому операція додавання повинна бути проведена над $m_x n$ парами чисел.

Очевидно, всі ці операції можуть виконуватися паралельно і незалежно один від одного кількома обробними пристроями.

Інший приклад - обробка інформації від датчиків, які вимірюють одночасно один і той же параметр і встановлених на кількох однотипних об'єктах.

Далі, завдання обробки сигналів від радіолокаційної станції: всі сигнали обробляються по одній і тій же програмі.

Третій шлях паралельної обробки інформації - конвеєрна обробка - може бути реалізована в системі і з одним процесором, розділеним на деяке число послідовно включених операційних блоків. Кожен з операційних блоків спеціалізується на виконанні чітко визначеної частини операції.

При цьому процесор працює таким чином: коли і-й операційний блок виконує і-ю частину j-й операції, (i-1) -й операційний блок виконує (i-1) -у частину (j + 1) -й операції, а (i + 1) -й операційний блок виконує (i + 1) -ю частина (j-1) -й операції. В результаті утворюється свого роду конвеєр обробки.

Розглянемо це на прикладі.

Операція додавання двох чисел з плаваючою точкою $A*2^x + B*2^y = C*2^{xvy}$ можна розділити на чотири послідовно виконуваних етапи (або кроки): порівняння порядків (ПП); вирівнювання порядків - зсув мантиси з меншим порядком для вирівнювання з мантисою з більшим порядком (ВП); додавання

мантис (ДМ); нормалізація результату (НР). Відповідно до цього в складі процесора передбачені чотири операційних блоки, з'єднаних послідовно і реалізують чотири перерахованих вище операції додавання: блоки СП, ВП, СМ і НР. На рис. 2.2 приведена структурна схема конвеєра операцій.

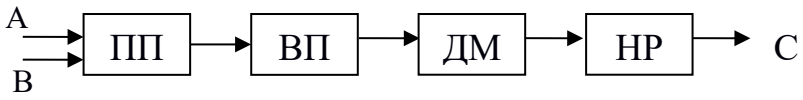


Рис. 2.2. Структурна схема конвеєра операцій.

Приймемо, що час виконання кожного кроку дорівнює відповідно 60, 100, 140, 100 нс. Таким чином операція додавання буде виконуватися послідовністю операційних блоків за час 400 нс.

На практиці буде розглянута задача додавання двох векторів, що містять *n* елементів з плаваючою точкою і тимчасова діаграма процесу. Підраховано загальний час додавання двох векторів за допомогою конвеєра

$$T_k = (n + m\text{-}1) * \tau,$$

де *m* - число операційних блоків,

τ - час виконання етапів,

n - число елементів векторів.

і без використання конвеєра.

$$T_0 = n \sum_{i=1}^{m} \tau_i$$

де τ_i - час виконання *i*-го етапу обробки.

Таким чином, при конвеєрному режимі роботи модулів здійснюється паралельне виконання частин різних завдань різними модулями. При цьому процес виконання кожного завдання розділяється на кілька приблизно однакових за часом використання послідовних частин, які виконуються в різних спеціалізованих модулях.

Робота модулів організовується таким чином, що будь-яка наступна задача починає виконуватись системою модулів до того, як буде виконано попереднє завдання.

Для ілюстрації принципу суміщення операцій при конвеєрній обробці розглянемо діаграми, наведені на рис. 2.3.

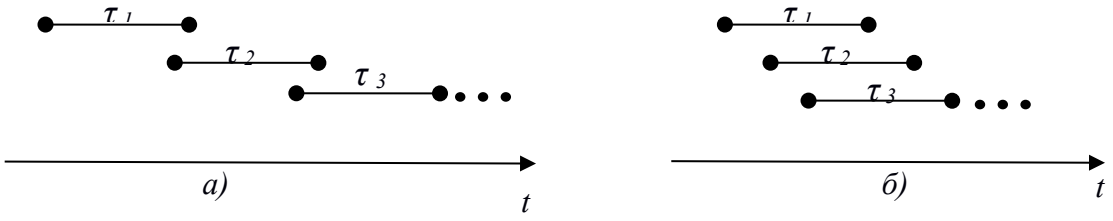


Рис. 2.3. Тимчасові діаграми послідовної і конвеєрної обробки.

При послідовній обробці команд (операцій) (рис. 2.3, а) кожна з них починає оброблятися тільки після завершення попередньої. При конвеєрній обробці (рис. 2.3, б) дві або більше команди виконуються одночасно, хоча і на різних стадіях.

Наприклад, одна з команд (операцій) може знаходиться в стані завершення, інша - в стані очікування операнда з оперативної пам'яті і т.д.

Необхідно відзначити, що чим довший ланцюжок даних і чим на більшу кількість етапів (а, отже, і операційних блоків) розбивається операція (при тій же тривалості її виконання), тим більший ефект від використання конвеєра може бути отриманий.

Ми розглянули конвеєрне виконання арифметичних операцій, але ідея конвеєра може бути поширена і на виконання команд.

У КС можна одночасно використовувати і конвеєр команд, і конвеєр арифметичних операцій, і навіть кілька паралельно працюючих конвеєрів команд і арифметичних операцій.

Конвеєрний метод дозволяє досягти підвищення продуктивності практично без додаткових апаратних витрат.

Тема 3: Шляхи підвищення продуктивності КС

Продуктивність є показником ефективності КС і включає в себе перш за все швидкість прийому, обробки та видачі інформації. Збільшення продуктивності КС прямо пов'язане з розширенням сфер застосування інформаційних технологій, збільшенням потоку і різноманітності оброблюваних даних.

З точки зору збільшення продуктивності основним архітектурним рішенням є паралелізм в обробці даних. Паралелізм в апаратурі має на увазі наявність декількох процесорів, або функціональних блоків ідентичного призначення-пам'яті, АЛП.

Для підвищення продуктивності комп'ютерних систем застосовують ряд архітектурних, технологічних, структурних та програмних рішень: зменшення проектних норм, конвеєризація та паралелізм обчислень, використання кешування і віртуальної пам'яті, багатопотокова організація обчислювального процесу, кластеризація, паралелізм програм і ряд інших.

Розглянемо передумови вдосконалення архітектур КС.

Досягнення високої продуктивності при вирішення поточних актуальних завдань, визначається взаємозалежним вдосконаленням компонентів тріади, таких як:

- елементна база;
- архітектура, структура і організація функціонування комп'ютерів і КС;
- системне і прикладне програмне забезпечення.

Збільшення продуктивності КС досягається за рахунок підвищення тактової частоти логічних елементів і елементів пам'яті, а також цих елементів, що дозволяють вводити паралелізм обробки і програмованість структури КС.

Архітектурно-структурні прийоми організації КС наступні**:**

- ієрархічна багаторівнева пам'ять;

- сторінкова організація основної пам'яті;
- реалізація буферної пам'яті як кеш-пам'яті;
- реалізація буферної пам'яті як векторних регістрів;
- реалізація буферної пам'яті як наборів реєстрових файлів процесора при мультипотоковій організації звернень до пам'яті;
- паралельне функціонування обробних пристроїв і блоків пам'яті;
- розпаралелювання звернень до пам'яті;
- програмне переналаштування зв'язків між компонентами пристроїв.

Використання кеш-пам'яті підвищує продуктивність КС. При використання кеш-пам'яті можна істотно зменшити час, який процесор витрачає на доступ до пам'яті. Кеш-пам'ять містить копії поточних значень полів основної пам'яті, доступ до яких здійснюється в попередніх командах. Зчитування з кеш-пам'яті здійснюється в кілька разів швидше ніж з динамічного ОЗУ.

Збільшення продуктивності досягається за рахунок використання в КС конвеєризації. Оскільки різні компоненти процесора КС реалізують виконання різних фаз команди, можна побудувати такий процесор, в якому будуть виконуватися одночасно кілька команд по конвеєру. Конвеєр можливий за умови послідовного запису команд в пам'яті.

Конвеєризація обчислень (Конвеєрна обробка) полягає в паралельному виконанні різних операцій або їх частин, що призводить до суттєвої економії часу, що витрачається на рішення. Як правило, операції ділять на елементарні частини, що називаються ланками конвеєра, так, щоб кожна ланка виконувалася за один такт конвеєра. Процесор являє собою сукупність функціональних блоків, кожен з яких орієнтований на виконання певної елементарної дії. Суміщення в часі виконання різних частин різних функцій здійснюється, завдяки паралельній роботі різних функціональних блоків. Наприклад, в процесорі Power 4 за один такт може виконуватися вісім інструкцій.

Щоб пояснити, що собою можуть представляти елементарні дії конвеєра, розглянемо ланки процесора Ultra SPARC-1, в якому реалізований дев'ятиланковий конвеєр. На першій ланці проводиться вибірка команд з кеш-пам'яті. На другій ланці команди декодуються і поміщаються в буфер команд. На третій ланці команди групуються і розподіляються за функціональними виконавчими пристроями. На четвертій ланці виконуються цілочисельні команди або обчислюється віртуальна адреса для звернення до пам'яті, а також здійснюються остаточне декодування команд плаваючої арифметики і звернення до регістрів. На п'ятій ланці відбувається звернення до кеш-пам'яті даних, визначається наявність запитаного операнда в кеш-пам'яті, здійснюються переходи. Якщо операнда в кеш-пам'яті немає, то відповідна команда завантаження надходить в буфер завантаження. З цього моменту цілочисельний конвеєр очікує завершення роботи конвеєрів плаваючою арифметики, які починають виконання відповідних команд. Потім проводиться аналіз виникнення виняткових ситуацій. На останньому щаблі всі результати записуються в регістри і команди вилучаються з обробки.

Ефективність конвеєризації обчислень істотно залежить від ряду факторів. По-перше, такт конвеєра визначається самою повільною ланкою, необхідно збалансоване виділення ланок. По-друге ефективність знижується через наявність конфліктів, що виникають з наступних причин:

- запит в деяких комбінаціях команд одних і тих же ресурсів, що перешкоджає поєднанню відповідних дій у часі;
- наявність залежності за даними, що виникає, якщо операнд чергової команди залежить від результату виконання попередньої;
- наявність залежності з керуванням, що призводить до неоднозначності передбачення подальших дій через команди умовного переходу.

Найбільш істотні затримки конвеєра, обумовлені залежностями з керуванням, тому приймаються різні заходи щодо зменшення негативного впливу цього фактора. Так, наприклад, можна за рахунок збільшення

апаратних витрат паралельно обробляти обидва альтернативні шляхи продовження обчислень після переходу. Наприклад, в процесорі Power 4+ використовуються гіперконвеєр з 17 ланками, а в процесорі Pentium 4 з 20 ланками (прогноз переходів в циклах засноване на інформації, отриманої на першому витку). У мікропроцесорах компанії Intel застосована суперскалярна структура, яка характеризується наявністю двох конвеєрів.

Наступний підхід пов'язаний з використанням віртуальної пам'яті. Цей підхід передбачає так звану сторінкову організацію пам'яті. Величина сторінок різна в різних системах (від 256 до 4096 байт). Фізична адреса операнда при такій організації обчислюється апаратними засобами за спеціальним алгоритмом.

Відповідно до закону паралельних процесорів, в паралельній системі з N процесорами однакової продуктивності загальна продуктивність КС зростає як $logN$. З цієї точки зору класичним прикладом паралелізму є застосування співпроцесорів, що несуть самостійне функціональне навантаження. До них відноситься арифметичні, матричні, сигнальні співпроцесори.

Використання співпроцесорів збільшує продуктивність КС. Найбільш поширені співпроцесори, що виконують операції з плаваючою точкою або призначені для обробки масивів. Співпроцесор виконує команду, поки центральний процесор знаходиться в стані очікування, а по завершенні операції передає йому керування.

Для підвищення продуктивності КС застосовують ряд технологічних рішень: зменшення проектних норм і багатопотокова організація обчислювального процесу. Зменшення проектних норм супроводжується збільшенням числа компонентів в кристалі НВІС, скороченням відстаней між зв'язуваними компонентами. Крім того, зменшуються паразитні ємності в електричних ланцюгах. Отже, знижуються спотворення сигналів і час їх поширення.

В даний час для подальшого підвищення продуктивності використовуються кілька підходів:

- реалізація двох або більше процесорів на одній НВІС (так зване мультиплексування на рівні мікросхем). Завдяки малим відстаням між основними регістрами і кеш-пам'яттю вдається помітно знизити затримки при паралельному виконанні операції в процесорах;
- архітектура надвеликого командного слова EPIC, яка передбачає програмування з високим рівнем паралелізму на рівні команд;
- згадана вище багатопотокова організація обчислень, полягає в багатопотоковій обробці інформації на основі паралелізму на рівні ниток TLP (Thread Level Parallelism). Суть TLP полягає в завантаженні конвеєра більш ніж одним завданням. Якщо в звичайному конвеєрі одне завдання (нитка) не в змозі завантажити конвеєр і ряд функціональних блоків протягом деяких тактів залишаються вільними, то в технології TLP ці блоки завантажуються операціями з інших ниток.

Технологія TLP реалізується за допомогою багатопотокової архітектури MTA (Multi Threading Architecture). В MTA передбачені спеціальні програмні і апаратні засоби, що стежать за правильним розподілом ресурсів між нитками. Перемикання з однієї нитки на іншу відбувається або при настанні певної події (наприклад, переривання або необхідність звернення в повільну пам'ять), або по черзі.

Відповідно до способу одночасного виконання ниток SMT (Simultaneous Multi-Threading) на кожному такті в кожен функціональний блок може направлятися команда з будь-якої нитки. Можливий і більш простий спосіб, при якому в будь-який момент часу працює тільки одна нитка, а при виникненні переривання відбувається перемикання з цієї нитки на іншу.

Кластеризація також один із шляхів підвищення продуктивності КС за рахунок спільного використання багатьох комп'ютерів. Кластер - сукупність

комп'ютерів, що функціонують як єдина система з загальними ресурсами. Основна мета, що зумовила появу кластерів, - збереження працездатності КС шляхом перерозподілу навантаження при виході з ладу частини ресурсів. Кластери дозволяють нарощувати обчислювальну потужність, оскільки легко масштабуються.

Для збільшення продуктивності з точки зору вимог до технології програмування, використовуються три підходи:

- локалізованість оброблюваних даних;
- виявлення паралельних дій в програмах;
- виявлення обчислювальних ядер (характерних фрагментів) програм, прискорення виконання яких сприяє істотному прискоренню виконання програми в цілому.

Тема 4: Структурна організація комп'ютерних систем паралельної обробки

Існує багато різних класифікацій систем паралельної обробки інформації. Це класифікаційні системи Шора, Ерланга, Флінна. Вдалою, на наш погляд, є класифікація Флінна, що полегшує розуміння особливостей структур і функціонування КС. Вона базується на ознаці одинарних або множинних потоків команд і даних.

В якості основних ознак класифікації схеми обрано такі: тип потоку команд; тип потоку даних; спосіб обробки даних; ступінь пов'язаності функціональних елементів; тип зв'язків між елементами КС.

Різні способи організації паралельної обробки інформації можна представити як способи організації одночасного впливу одного або декількох потоків команд на один або кілька потоків даних.

Поняття одиночних і множинних потоків команд і даних були введені М. Дж. Флінном.

Під потоком команд розуміється послідовний ряд команд виконуваних системою, а під потоком даних - послідовний ряд даних, що викликаються потоком команд.

Під множинним потоком команд або даних будемо розуміти наявність в системі декількох послідовностей команд, які перебувають в стадії реалізації, або декількох послідовностей даних, що піддаються обробці командами.

Виходячи з можливості існування одиночних і множинних потоків, всі системи можуть бути розбиті на чотири великі класи.

1. Системи з одиночним потоком команд і одиночним потоком даних (ОКОД) - SISD (Single Instruction Data Stream), тобто здійснюється звичайна послідовна обробка.
2. Системи з множинним потоком команд і одиночним потоком даних (МКОД) - MISD (Multiple Instruction Single Data Stream),

архітектура, при якій кілька потоків команд обробляють один потік даних.

3. Системи з одиночним потоком команд і множинним потоком даних (ОКМД) - SIMD (Single Instruction Multiple Stream) - архітектура, при якій один потік команд обробляє кілька потоків даних.
4. Системи з множинним потоком команд і множинним потоком даних (МКМД) - MIMD (Multiple Instruction Data Stream) - архітектура, при якій паралелізм в КС досягається шляхом виконання незалежних частин завдань над різними наборами даних.

Основні типи паралельних КС за класифікацією Флінна представлені на рис. 4.1.

До класу КС типу ОКМД відносяться матричні і асоціативні системи. До архітектури МКОД відноситься магістральна структура, що забезпечує підвищену продуктивність для задач, алгоритми яких допускають розпаралелювання. До MIMD типу відноситься мультипроцесорна і багатомашинна архітектура.

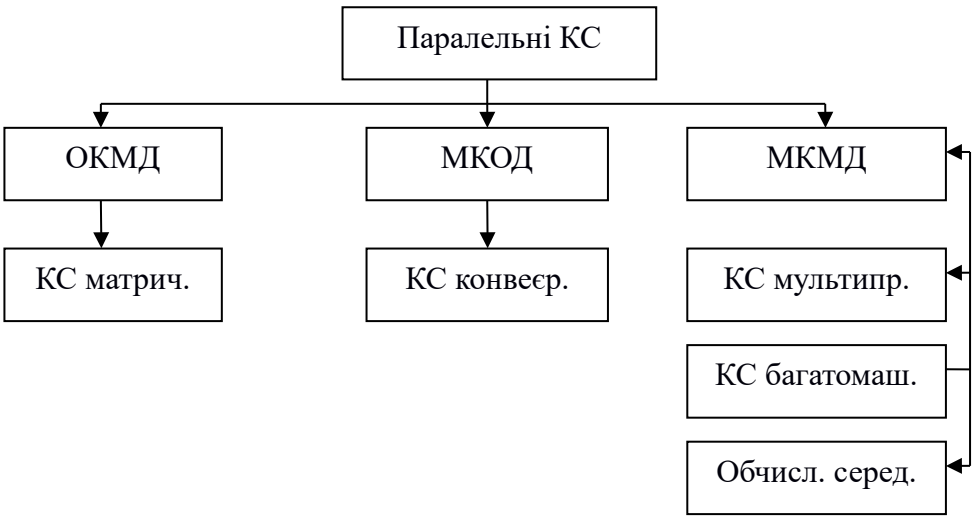


Рис. 4.1. Основні типи КС.

На рис. 4.2 наведені системи класу ОКОД. Це однопроцесорні ЕОМ, що включають в себе пам'ять (ЗП) для команд і даних і один процесор, що

містить АЛП і пристрій керування. Структура ЕОМ містить одинарний потік команд і одинарний потік даних.

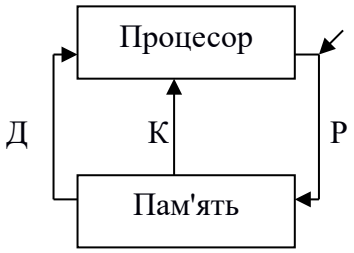


Рис. 4.2. Система ОКОД, Д - дані, К - команди, Р - результати.

В сучасних системах цього класу найбільш широко використовується перший шлях організації паралельної обробки, поєднання в часі різних етапів вирішення завдань, при якому в системі одночасно працюють різні пристрої: введення, виведення і обробки інформації.

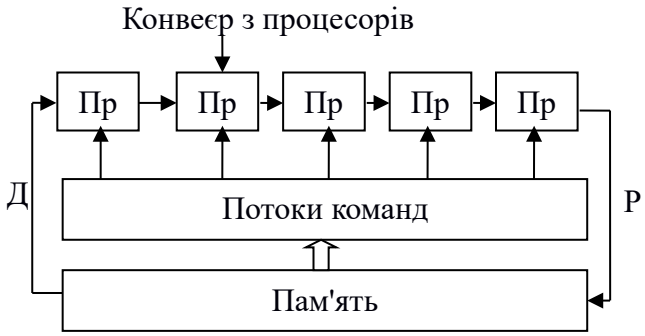


Рис. 4.3. Системи МКОД.

Системи класу МКОД. Структуру систем можна представити у вигляді схеми, зображеної на рис. 4.3. Ці класи КС прийнято називати конвеєрними або системами з магістральної обробкою інформації.

Система має регулярну структуру в вигляді ланцюжка послідовно з'єднаних процесорів або спеціальних обчислювальних блоків (СОБ). Так що інформація на виході одного процесора (СОБ) є вхідною інформацією для наступного в конвеєрному ланцюжку. Процесори (СОБ) утворюють конвеєр. На вхід конвеєра одинарний потік даних доставляє операнди з пам'яті. Кожен

процесор обробляє частину завдання, або операції, передаючи результати сусідньому процесору, який використовує їх в якості вихідних даних.

Таким чином, рішення задач для деяких вихідних даних розгортається послідовно в конвеєрному ланцюжку. Це забезпечується підведенням до кожного процесору (СОБ) свого потоку команд, тобто множинний потік команд.

Система цього класу розвиває максимальну продуктивність тільки при вирішенні завдань певного типу, в яких існують довгі послідовності однотипних операцій над великою послідовністю даних, тобто, коли має місце паралелізм об'єктів і даних.

Системи класу ОКМД. Структура типу ОКМД (SIMD) - матрична КС. На рис. 4.4 представлена узагальнена структура КС типу ОКМД. Система містить деяке число однакових простих швидкодіючих процесорів, з'єднаних один з одним і з пам'яттю даних регулярним чином так, що утворюється сітка (матриця), в вузлах якої розміщуються процесори.

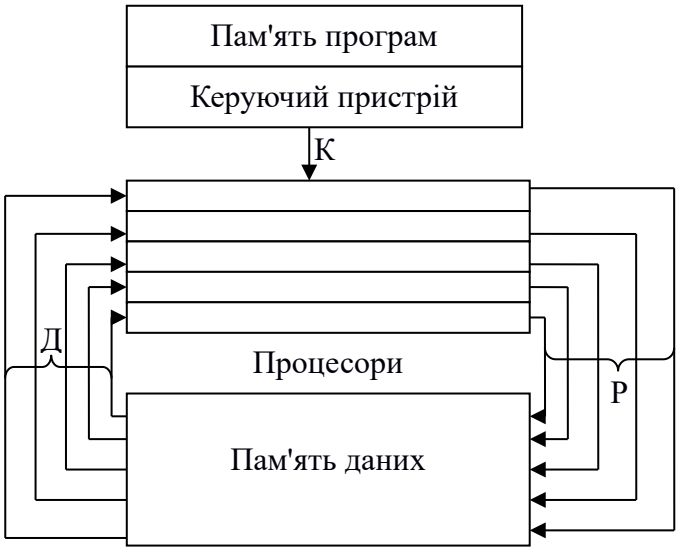


Рис. 4.4. Система ОКМД (SIMD).

В системі є кілька потоків даних і один загальний потік команд, тобто всі процесори виконують одночасно одну і ту ж команду, але над різними операндами, що доставляються процесорам з пам'яті з декількома потоками даних. Подібні КС часто називають системами із загальним потоком команд.

Пам'ять даних може мати не тільки адресну вибірку, а й асоціативну, тобто по вмісту пам'яті.

Системи цього класу також орієнтовані на використання паралелізму об'єктів або даних для підвищення продуктивності.

Системи класу МКМД. З розглянутих систем найбільш універсальними щодо класу вирішуваних завдань є системи МКМД. Їх програмне забезпечення не орієнтується на рішення тільки певного класу задач. Можливі два способи побудови систем МКМД: багатомашинний і багатопроцесорний (рис. 4.1). В багатомашинному варіанті вся система як би розпадається на кілька незалежних систем класу ОКОД. При цьому існують певні зв'язки між ЕОМ, які об'єднують ці ЕОМ в систему.

В багатопроцесорному варіанті система жорстко пов'язана із загальною пам'яттю команд і даних.

Багатомашинні системи в найкращій мірі пристосовані для вирішення потоку незалежних завдань, що характеризуються паралелізмом незалежних гілок.

На рис. 4.5 представлений загальний випадок структури КС, в якому кілька потоків даних і кілька потоків команд.

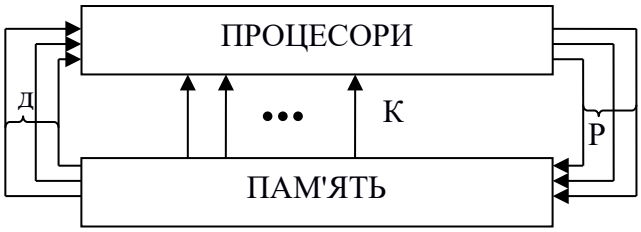
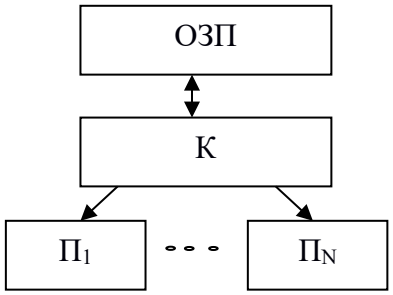


Рис. 4.5. Система МКМД (MIMD).

В даний час конкурують між собою, головним чином, дві структури побудови супер-ЕОМ: а) із загальним потоком команд; б) конвеєрна (зі спеціальними засобами обробки векторів).



К – комунікаційне середовище;
П – процесори.

Рис. 4.6. Системи MIMD із загальною пам'яттю

Більшість сучасних суперкомп'ютерів відносяться до класу MIMD. В цьому класі розрізняють кілька підкласів. У перших, це SM - системи (Shared Memory) із загальною пам'яттю, тобто мають загальний адресний простір. У системах з загальною пам'яттю (рис. 4.6) міжпроцесорний обмін відбувається через загальну пам'ять.

До цього підкласу відносяться багатопроцесорні векторні КС і SMP-системи (Symmetric Multiprocessing). У таких системах жорсткі вимоги пред'являються до швидкодії каналів. У векторних КС використовується високошвидкісна шина між оперативною пам'яттю і векторними регістрами, а в SMP-системах висока продуктивність підтримується за рахунок введення кеш-пам'яті великого обсягу, тобто вимоги до пропускної здатності тракту «оперативна пам'ять - кеш пам'ять» істотно знижується.

По-друге DM-системи (Distributed Memory) з розподіленою пам'яттю, в яких кожен процесор має свою оперативну пам'ять (рис. 4.7). Прикладом DM-систем служать кластерні системи.

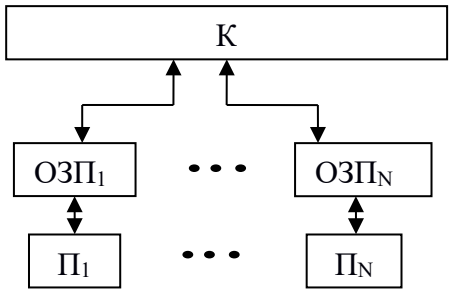


Рис. 4.7. Системи типу MIMD з розподіленою пам'яттю.

По-третє, це MPP-система (Massivelly Parallel Processing) з масовим паралелізмом, в яких кілька копій однієї програми паралельно виконуються в різних вузлах з різними даними. Відмітною ознакою MPP-систем є більше число процесорів, ніж SMP-системах.

Тема 5: Матричні комп'ютерні системи

В даний час принципи організації матричних систем реалізовані в стаціонарних комп'ютерах. Однак поява мікропроцесорних комплектів ВІС зумовило принципову можливість створення матричних КС з прийнятними масо-габаритними характеристиками.

Матричні КС відносяться до класу SIMD (ОКМД). Вони найкраще пристосовані для вирішення завдань, що характеризуються паралелізмом незалежних об'єктів або паралелізмом даних. Спрощена структурна організація матричних КС представлена на рис. 5.1.

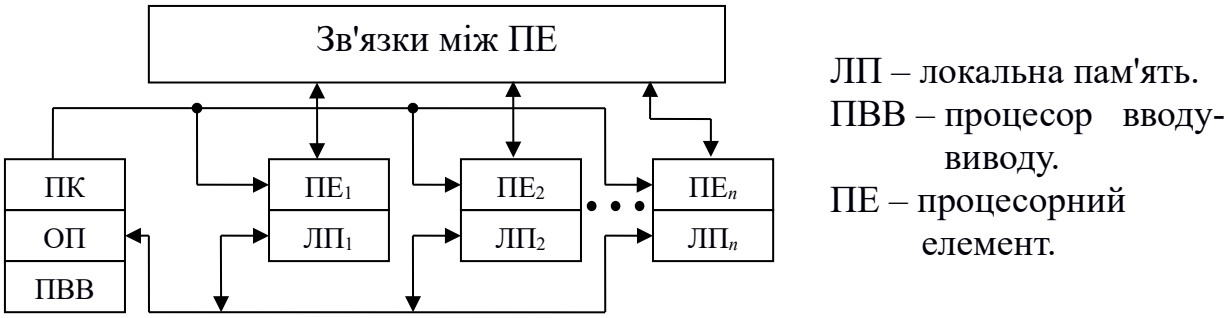


Рис. 5.1. Спрощена структура матричної КС

Особливістю даної організації структури є те, що загальний пристрій керування (ПК) визначає всім *n* процесорним елементам (ПЕ) однакові команди для кожного етапу обчислення програми алгоритму.

Всі процесори одночасно виконують одну і ту ж операцію - кожен над своїми даними, які зберігаються в локальних накопичувачах процесорів.

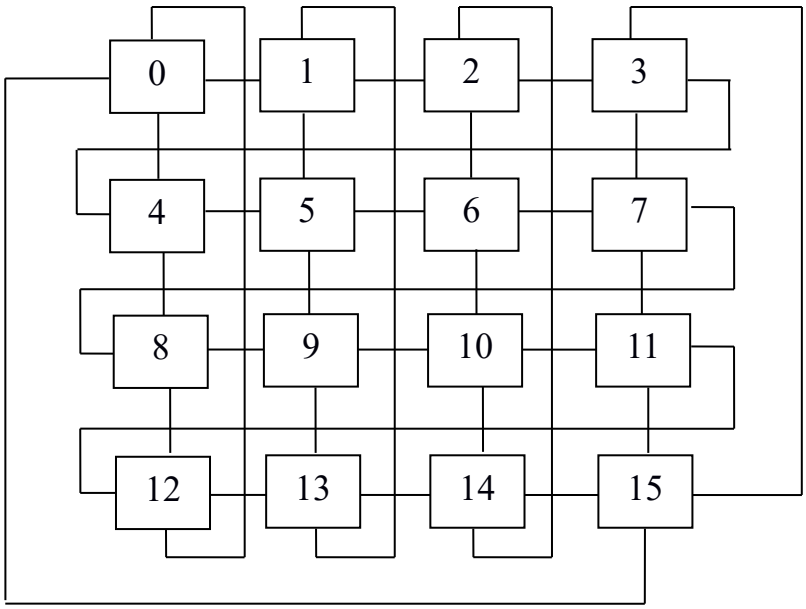


Рис. 5.2. Матриця процесорних елементів

До ПК пред'являються високі вимоги по швидкодії, що забезпечує безперервність потоку команд, які обирають з основної оперативної пам'яті. Команди переходів виконуються безпосередньо ПК з врахуванням стану ПЕ на момент завершення попередньої команди. За своєю структурою ПЕ однорідні і являють собою арифметико-логічні пристрої, що володіють досить високою швидкодією. Матричні КС є квадратною решіткою, в вузлах якої розміщуються ПЕ. Кожен ПЕ в даній решітці пов'язаний з чотирма сусідніми (рис. 5.2).

Блоки керування окремими ПЕ містять засоби для дешифрування команд, керування їх реалізацією, формування ознак результатів операцій і звернення до локальної пам'яті.

При виконанні потоку команд окремі ПЕ можуть пропускати і не виконувати деякі з них, якщо на вході ПЕ немає відповідних даних.

Команди вводу-виводу передаються для виконання в спеціальний процесор вводу-виводу (ПВВ).

Ступінь розпаралелювання максимальна, якщо обробку даних здійснюють одночасно всі ПЕ, тобто жоден з них не пропускає чергову

команду. Простої ПЕ викликані невідповідністю числа операцій розпаралелювання на різних етапах обчислень числа процесорних елементів.

Ефективність використання матричних обчислювальних систем з точки зору отримання високої реальної швидкодії вирішення завдань обумовлюється застосуванням високошвидкісних процесорних елементів, забезпеченням високого ступеня розпаралелювання процесів обробки даних і скороченням частки витрат на підготовчі операції.

Тема 6: Асоціативні комп'ютерні системи

Асоціативні КС відносяться до числа систем класу ОКМД. Вони характеризуються великою кількістю операційних пристроїв, здатних одночасно по командам одного керуючого пристрою вести обробку декількох потоків даних, (як матричні). Але ці системи суттєво відрізняються від матричних способами формування потоків даних.

У матричних системах дані надходять на обробку від загальних або роздільних запам'ятовуючих пристроїв з адресним виробленням інформації, або безпосередньо від пристроїв джерел даних.

В асоціативних КС інформація на обробку надходить від асоціативних запам'ятовуючих пристроїв (АЗП). АЗП характеризується тим, що інформація з них вибирається не по певній адресі, а по її змісту. Принцип роботи АЗП пояснює схема, розглянута нижче.

Асоціативну систему (асоціативний процесор) можна представити в загальному випадку як систему, що володіє наступними двома властивостями: 1) дані, що знаходяться в пам'яті, можуть вибиратися на підставі їх вмісту (не за їх адресами); 2) операції перетворення даних, як арифметичні, так і логічні, можуть здійснюватися над декількома множинами аргументів за допомогою однієї команди. Така система містить асоціативну пам'ять, арифметико-логічний пристрій, підсистему керування, пам'ять команд (керуюча пам'ять) і інтерфейс вводу-виводу (рис. 6.1).

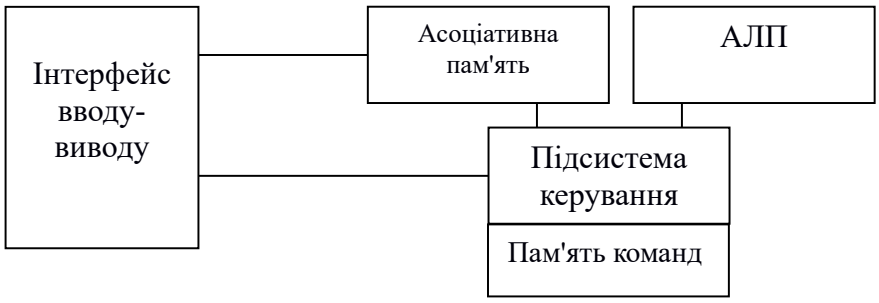


Рис. 6.1. Структурна схема асоціативної системи

У даній структурі інтерес представляє асоціативна пам'ять або АЗП.

Розглянемо принцип роботи АЗП на схемі, представленій на рис. 6.2.

Асоціативний ЗП складається з наступних елементів (модулів):

- запам'ятовуючого масиву (ЗМ);
- регістру асоціативних ознак (РгАО);
- регістру маски (РгМ);
- регістру індикаторів адреси зі схемами порівняння на вході.

ЗМ розділений на *m*-розрядні комірки, число яких *n*.

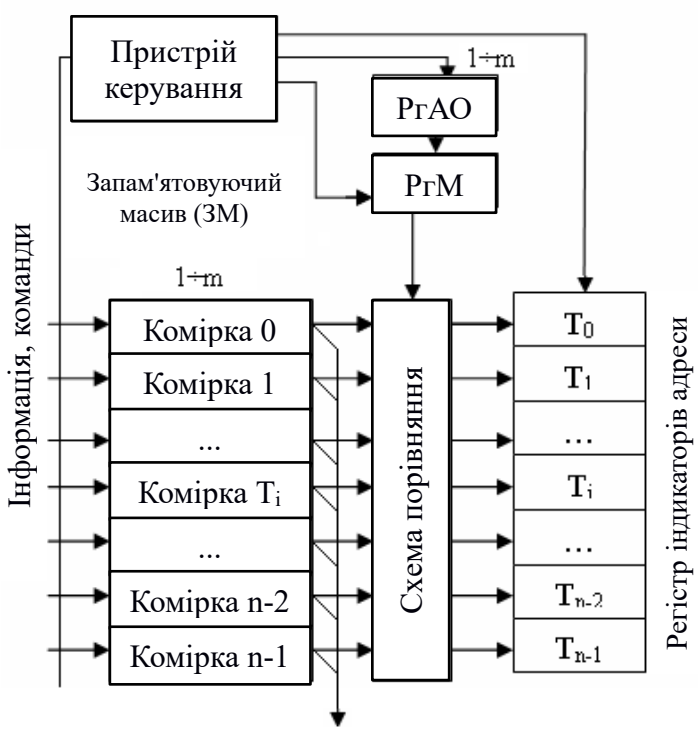


Рис. 6.2. Структурна схема АЗП

Вибірка інформації з АЗП відбувається наступним чином.

Пристрій керування передає код ознаки шуканої інформації в регістр асоціативних ознак (РгАО). Код може мати довільне число розрядів - від 1 до *m*. Якщо код ознак використовується повністю, то він без зміни надходить на схему порівняння. Якщо необхідно використовувати тільки частина коду, тоді непотрібні розряди маскуються за допомогою регістра маски (РгМ). Перед початком пошуку інформації в АЗП всі розряди регістра індикаторів адреси

встановлюються в стан 1. потім проводиться опитування першого розряду всіх комірок ЗМ і вміст порівнюється з першим розрядом РгАО. Якщо вміст першого розряду *i*-ї комірки не збігається з вмістом першого розряду РгАО, то відповідний цьої комірці розряд регістра індикаторів адреси T_i скидається в стан 0, якщо збігається, - на T_i залишається 1. Після цього операція повторюється з другим, 3-м та наступними розрядами до тих пір, поки не буде проведено порівняння зі всіма розрядами РгАО.

Після поразрядного опитування і порівняння в стані 1 залишаться ті розряди регістра індикаторів адреси, які відповідають коміркам, що містить інформацію, яка збігається із записаною в РгАО. Ця інформація може бути зчитана в тій послідовності, яка визначається пристроєм керування.

Головна перевага АЗП перед адресними ЗП визначається тим, що час пошуку інформації залежить тільки від числа розрядів ознаки і від швидкості опитування розрядів, але абсолютно не залежить від числа комірок ЗМ. (В адресних ЗП при операції пошуку необхідний перебір всіх комірок запам'ятовуючого масиву).

На відміну від адресованих ЗП, АЗП повинні не тільки зберігати інформацію, а й виконувати логічні функції. Тому АЗП дозволяють здійснити пошук не лише за рівністю вмісту комірки заданій ознаці, а й за іншими умовами: вміст комірки більший (менший) ознаки РгАО, а також більший або рівний (менший або рівний).

Тема 7: Комп'ютерні системи з проблемно і функціонально орієнтованими процесорами

Високопродуктивні системи загального призначення створюються на основі багатопроцесорних комплексів. Використання в таких системах однотипних процесорів (аналогічних процесорам ЕОМ загального призначення) виявляється неекономічним. Пояснюється це тим, що в кожному процесорі в кожен момент часу використовується лише частина ресурсів, що забезпечують обробку даних одного типу.

Найбільш економічний спосіб побудови багатопроцесорної системи загального призначення - використання спеціалізованих проблемно і функціонально орієнтованих процесорів. Вони орієнтовані на реалізацію певних функцій: обробки скалярних величин, текстів, керування даними, матричної обробки та ін.

Багатопроцесорні КС, побудовані на основі різнотипних процесорів, орієнтованих на реалізацію певних функцій називаються функціонально-розподіленими КС (ФРКС). Це неоднорідні системи і будуються вони як проблемно-орієнтовані - шляхом включення до їх складу набору процесорів, що відповідають потребам оброблюваних задач.

Розглянемо принцип структурної організації ФРКС. На рис. 7.1 представлена ФРКС.

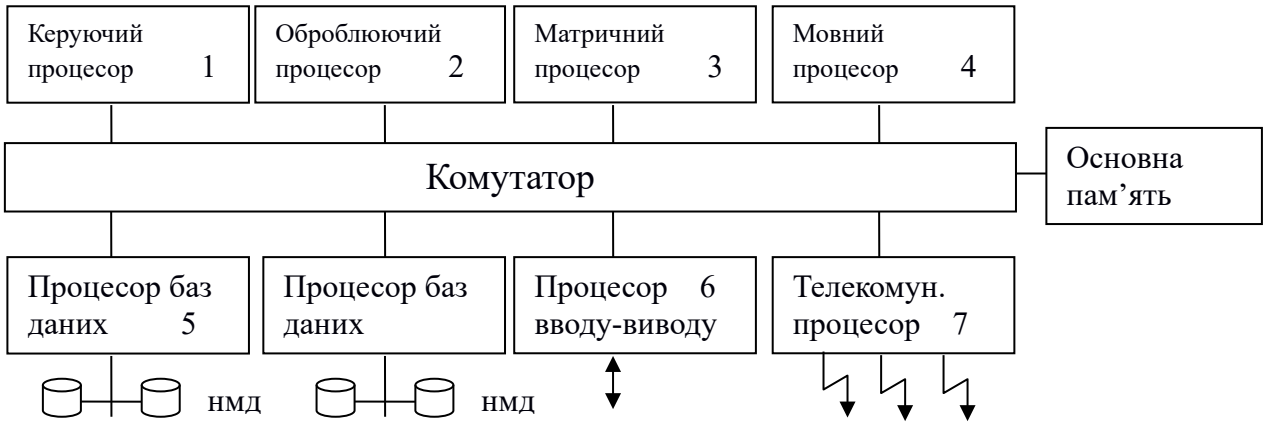


Рис. 7.1. Функціонально-розподілена КС

Система складається із сукупності процесорів, що мають індивідуальну пам'ять, і основну пам'ять.

Є ядро системи яке забезпечує інформаційне сполучення всіх пристроїв. Ядро може бути реалізовано у вигляді системної шини (магістралі), комунікаційного поля або комутатора основний пам'яті. У перших двох випадках кожен процесор може обмінюватися даними з будь-якими іншими процесорами і основною пам'яттю.

При використанні комутатора основної пам'яті, обмін даними здійснюється тільки через пам'ять. В даній структурі (рис. 7.1) керуючий процесор реалізує супервізорні функції - керування ресурсами і завданнями, що обробляє процесор - обробку числових і символьних даних, матричний процесор - матричну і векторну обробку, мовний процесор - трансляцію програм. Крім того, процесори баз даних реалізують доступ до наборів даних і керування базами даних, процесор вводу-виводу обслуговує пристрої вводу-виводу і телекомунікаційний процесор забезпечує передачу по каналах зв'язку. Склад процесорів в конкретній системі залежить від класу вирішуваних завдань. (В системі можуть використовувати два обробних процесора або кілька телекомунікаційних).

Обробка кожного завдання розподіляється між процесорами. Різні кроки завдань, програми та гілки (блоки) програм виконуються обробляючим, матричним і мовним процесорами. Розподіл ресурсів між завданнями і керування завданнями проводиться керуючим процесором, який реалізує керуючі програми операційної системи.

Спеціалізація процесорів забезпечується на різних рівнях - на рівні структури, мікропрограмному і програмному.

Спеціалізація на рівні структури досягається за рахунок використання в операційній частині процесора реєстрових структур і мікрооперацій, що реалізують заданий набір операцій. Такими є матричні процесори, що містять сукупність арифметико-логічних пристроїв, за допомогою яких паралельно

обробляються вектори і матриці. Спеціалізація на мікропрограмному рівні зводиться до створення за допомогою вбудованого програмного спеціалізованого набору операцій, орієнтованого на обчислення заданого набору функцій.

Функціональна спеціалізація процесорів на програмному рівні досягається за рахунок завантаження в процесор відповідного набору програм.

У ФРКС використовуються три рівні спеціалізації процесорів. Обробні і матричні процесори мають спеціалізовану структуру: обробні - для виконання операцій над логічними значеннями, цілими і дійсними числами, матричні - для виконання векторних і матричних операцій. Решта процесорів функціонально спеціалізуються на рівні мікропрограм або програм.

Застосування. ФРКС використовуються при реалізації великого набору функцій над різними типами і структурами даних: обробку цілочисельних значень, дійсних чисел, графічної інформації та зображень, текстів, матричну обробку, трансляцію програм, доступ до даних, організованих в набори або бази і т.д.

Проблема підвищення ефективності обчислень шляхом адаптації обчислювальної системи до завдань за допомогою динамічної програмної перебудови структури системи є сучасною і привертає дослідників.

Перебудовуваність структури забезпечує 1) високу продуктивність системи за рахунок її адаптації до обчислювальних процесів і складу оброблюваних завдань; 2) живучість системи при відмовах елементів (підвищення стійкості до відмов елементів).

В останні роки ведуться роботи в області створення паралельних відкритих (з можливістю розвитку системи за рахунок підключення до неї додаткових модулів без зміни принципів функціонування наявних модулів) багатомодульних систем з перебудовуваною структурою. Для створення таких систем необхідно вирішити ряд проблем:

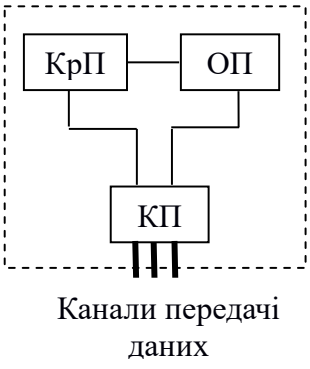
- структурна організація систем, що забезпечує створення ансамблів процесорів, запам'ятовуючих пристроїв і каналів обміну даними, які відповідають потребам обчислювального процесу, при помірних витратах ресурсів на їх організацію та координацію;
- організація обчислювальних процесів, що забезпечує їх паралелізм, розподіл по системі модулів і координацію асинхронно виконуваних підпроцесів при помірних витратах.
- створення мов високого рівня, що описують алгоритми в системно-незалежній формі і зберігають представлення про паралелізм обчислювального процесу.

Комп'ютерні системи з перебудовуваною структурою будуються на основі мікропроцесорних модулів. Модуль повинен реалізувати наступні функції:

- обробку даних, яка зводиться до обробки логічних значень, числових значень, представлених у вигляді цілих і дійсних чисел, і рядків символів;
- керування обчислювальними процесами, що забезпечує взаємодію модуля з ансамблем модулів, що реалізують процес, і з системою в цілому;
- встановлення з'єднання з іншими модулями і передачу даних між ними для забезпечення обчислювальних процесів.

З врахуванням цих функцій модуль КС розглядається як сукупність трьох процесорів: обробного (ОП), керуючого (КрП) і комунікаційного (КП).

На рис. 7.2 наведено склад модуля системи з перебудовуваною структурою.



КрП - керуючий процесор

ОП - обробляючий процесор

КП - комунікаційний процесор.

Рис. 7.2. Склад модуля системи з перебудовуваною структурою

Комунікаційний процесор забезпечує обслуговування декількох (зазвичай двох-шести) каналів передачі даних.

Фізично модуль може реалізуватися на основі однієї мікро-ЕОМ. Цей модуль в мультипрограмному режимі виконує функції обробки, керування процесами і передачі даних. Або модуль може бути реалізований на основі декількох мікропроцесорів, між якими розділяються функції керування, обробки і передачі інформації.

У КС з перебудовуваною структурою модулі об'єднуються в найпростіші структури: матричні, пірамідальні і кубічні. Найбільш поширеною є матрична. Фрагмент матричної структури зображений на рис. 7.3.

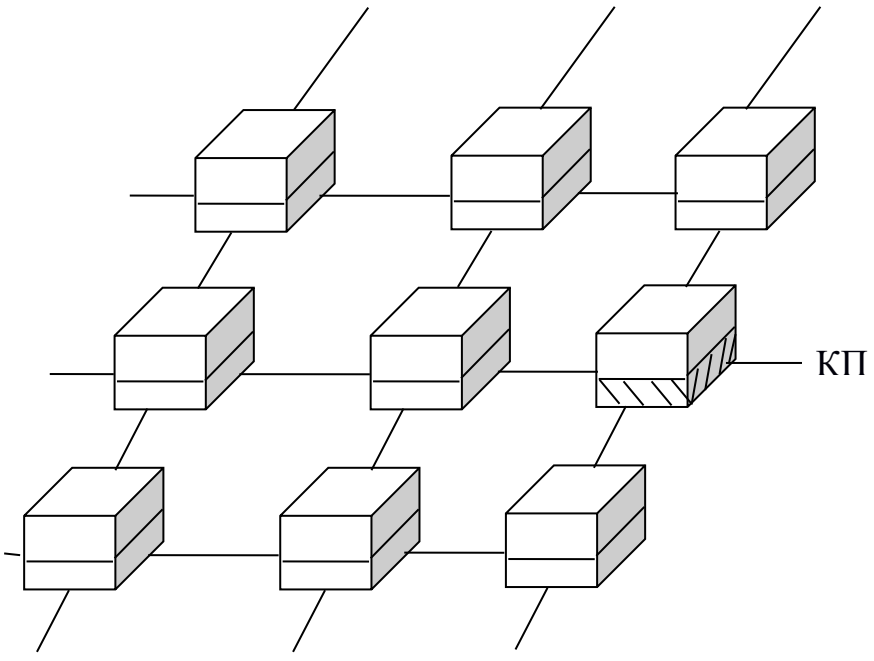


Рис. 7.3. Фрагмент матричної структури

Модулі, в яких виділено комутаційний процесор, з'єднується за допомогою КП в матрицю. Комутаційні процесори і канали зв'язку утворюють в сукупності комутаційне поле.

Комутаційне поле забезпечує з'єднання взаємодіючих модулів і передачу даних між ними. Частина модулів системи обслуговує периферійні пристрої, забезпечуючи взаємодію зовнішніх пристроїв з процесами, які реалізуються в будь-якому з модулів системи.

Існує інший спосіб організації системи з перебудовуваною структурою - на основі ВК, що мають комутаційне поле. У даній системі комутаційне поле утворено сукупністю комутаторів з децентралізованим керуванням. Найбільш економічними є поля з багаторівневою організацією.

Багаторівневі комутаційні поля дозволяють створювати з'єднання між будь-якими парами процесорних модулів.

Основні проблеми організації обчислень в системах з перебудовуваною структурою пов'язані із забезпеченням паралелізму обчислень і розподіленого децентралізованого керування процесами і ресурсами.

Розподілене керування означає, що в системі не повинно бути виділеного модуля, на який покладено централізоване керування функціонуванням системи. Розподілене керування засноване на злагодженій роботі всіх модулів, кожен з яких реалізує однаковий набір правил керування, що забезпечують ефективне використання ресурсів (таке керування підвищує надійність системи). Для керування процесами пропонуються евристичні процедури, які не потребують великої місткості пам'яті і трудомістких обчислень.

Тема 8: Конвеєрні комп'ютерні системи

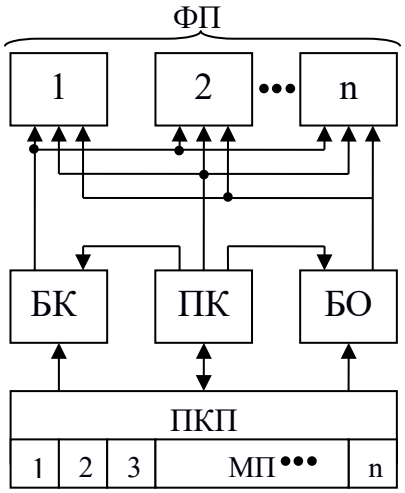
Конвеєрні КС (магістральні) відносяться до класу МКОД - системи з множинним потоком команд і одиночним потоком даних. Принцип конвеєрної обробки інформації знайшов широке застосування в ЕОМ. Особливо це відноситься до конвеєра команд. В сучасних КС, поряд з конвеєром команд, використовується і конвеєр даних. Поєднання цих двох конвеєрів дає можливість досягти дуже високої продуктивності систем. При цьому використовується кілька конвеєрних процесорів, здатних працювати одночасно і незалежно один від одного. Саме так і побудовані самі високопродуктивні системи.

Принцип конвеєрної обробки заснований на поділі обчислювального процесу на кілька процесів, кожен з яких виконується на окремому пристрої. Цей принцип може застосовуватися на різних ієрархічних рівнях обчислювального процесу, починаючи з рівня логічного елемента.

Спрощена структурна організація КС даного типу представлена на рис. 8.1.

Загальне пристрій керування (ПК) обслуговує кілька *n* різних за призначенням та структурою блоків обробки інформації - функціональних процесорів (ФП), причому кожен з них може мати свою локальну пам'ять. На структурній схемі також показані буфер команд (БК) і буфер операндів (БО). Загальний ПК можна використовувати для здійснення команд і операндів з оперативної пам'яті в буфер команд і операндів відповідно.

Попередньо декодує команди з метою визначення місця їх виконання, управляє передачею операндів відповідно до заданої послідовності їх обробки.



1,2, ..., n - ФП - функціональні процесори;
БК - буфер команд;
БО - буфер операндів;
ПКП - пристрій керування пам'яттю;
МП - модулі основної оперативної пам'яті;
ПК - пристрій керування.

Рис. 8.1.Структурная схема конвеєрної КС.

Кожен ФП виконує операцію над блоком даних, після чого передає його наступному процесору, а сам від попереднього процесора приймає на обробку новий блок даних.

Оперативна пам'ять конвеєрних КС будується за модульним принципом з розшаруванням звернень до неї. За допомогою пристрою керування пам'яттю (ПКП) число модулів основної оперативної пам'яті МП вибирається рівною кількості ФП.

Можуть бути різні варіанти організації структури і функціонування конвеєрних КС. Так, наприклад, статичний конвеєр відрізняється незмінністю своєї структури в процесі обробки даних, в той час як динамічний конвеєр може перебудовуватися відповідно до виконуваної програми.

Розрізняють також синхронні і асинхронні конвеєрні КС. При синхронній конвеєрній обробці інформації тривалість окремого етапу обробки команд постійна і вибирається з врахуванням найтривалішого етапу обробки команд, на одному з ФП.

Суть синхронної обробки полягає у використанні змінної тривалості найтривалішого етапу виконання команд, необхідну для кожної окремої команди. Тим самим скорочуються простої.

Існує два різних варіанти організації функціонування конвеєрних КС. У першому варіанті ПК прочитує з пам'яті безпосередньо вектори-команди, проте вектор-команда містить зазвичай *n* компонент, кожна з яких вказує, яку операцію повинен виконати відповідний процесор.

Другий варіант організації полягає в тому, що ПК прочитує команди «вперед» зі швидкістю, що перевищує швидкість виконання однієї команди окремими ФП. Прочитавши чергову команду, ПК аналізує, чи є умови для того, щоб почати її виконання, якщо можливо - доручає її виконання будь-якому вільному ФП.

Сучасні конвеєрні комп'ютерні системи є, як правило, статичними і синхронними. Для ілюстрації цих положень розглянемо приклад конвеєрної обробки команд, пронумерованих в порядку 1, 2, ..., 15, причому команди з номерами 1, 3, 6, 10, є командами переходу. На рис. 8.2 наведено приклад конвеєрної обробки інформації.

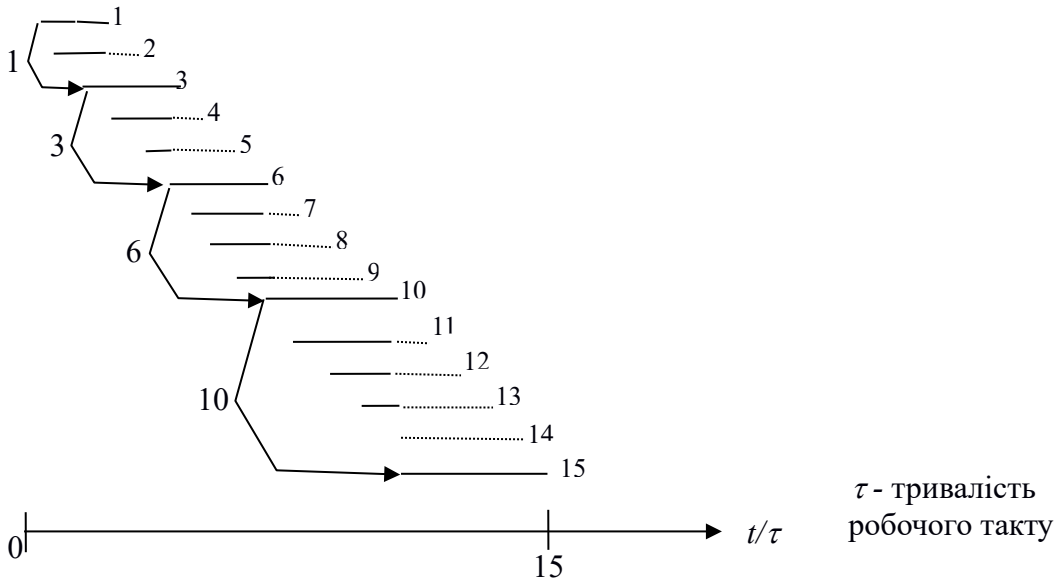


Рис. 8.2. Приклад конвеєрної обробки інформації.

Приймемо також, що глибина конвеєра $K = 4$, що відповідає одночасній обробці в конвеєрі чотирьох команд. В ході виконання команди з №1 в конвеєр викликаються і починають оброблятися команди з №2, 3, 4. Однак до моменту завершення команди з №1 відомо, що перехід здійснюється до команди 3. Таким чином, результат попередньої обробки команди з №2 анулюється. За аналогією при виконанні команд №3, 6, 10 анулюються результати часткової обробки команд 4, 5; 7, 9; 11; 12, 13 а команда з №14 не викликається.

З рис. 8.2 випливає, що при реалізації переходу через одну команду непродуктивно втрачається один такт часу процесора, при переході через дві команди - два такти і т.д.

Максимальне число непродуктивних тактів рівне k, що відповідає переходу через l команд, причому $l \geq k$.

Тема 9: Високопродуктивні відмовостійкі комп'ютерні системи

Для систем керування виробничими процесами і додатків по оперативній обробці інформації абсолютно природні підвищені вимоги до організації високонадійних обчислень. Сучасні телекомунікаційні системи, системи керування повітряним і наземним транспортом, медичні установи, фондові біржі, банки і промислові підприємства не можуть призупинити свою роботу через несправність комп'ютера. У подібних програмах простій може привести до затримки виходу продукції, втрати прибутків, поломки устаткування і до людських жертв. Тому проблема забезпечення надійності та відмовостійкості сучасних комп'ютерів є дуже актуальною і перспективною.

В даний час одним з основних завдань побудови комп'ютерних систем (КС) залишається забезпечення їх тривалого надійного функціонування. Це завдання має три складові: надійність, висока готовність і зручність обслуговування. Її рішення припускає, в першу чергу, боротьбу з несправностями системи, породжуваними відмовами і збоями в її роботі.

Відмовостійкість - властивість архітектури КС, що дозволяє користувачеві або функціональній програмі продовжувати роботу і тоді, коли в апаратних або програмних засобах виникають відмови.

За способом реалізації відмовостійкість підрозділяється на активну і пасивну.

Активна відмовостійкість базується на процесах виявлення відмови, локалізації відмови і реконфігурації системи. Відмови виявляються за допомогою засобів контролю, локалізуються за допомогою засобів діагностування і усуваються автоматичною реконфігуруючою системою.

Пасивна відмовостійкість полягає у властивості системи не втрачати свої функціональні властивості в разі відмови окремих елементів системи. Пасивна відмовостійкість пов'язана зі збільшенням кількості апаратури в кілька разів. Пасивна відмовостійкість застосовується в разі особливо відповідальних КС, коли не припустимі навіть короткочасні перерви в

обробці КС, а також для забезпечення відмовостійкості його найважливіших блоків або пристроїв.

Застосування активної відмовостійкості характеризується більш економними витратами апаратних засобів, ніж застосування пасивної відмовостійкості. Однак воно пов'язане з деякими втратами часу при відновленні роботи системи після відмови, а також втратами деякої частини даних. Активна відмовостійкість реалізується тільки в багатопроцесорних системах (із загальною пам'яттю, загальною шиною, матричною, кільцевою або іншою структурою). У той же час застосування пасивної відмовостійкості гарантує практично безупинну роботу КС і збереження всієї інформації. Ці обставини і визначають області застосування активної і пасивної відмовостійкості.

Введення відмовостійкості є одним з методів підвищення надійності КС. Питання про побудову і застосування відмовостійких систем виникає тоді, коли інші шляхи підвищення надійності не можуть забезпечити необхідного рівня надійності з технічних причин, або тоді, коли вони виявляються економічно виправданими.

В той час як випадкові відмови апаратури в нормальних умовах роботи - рідкісні події, ймовірність руйнування апаратури в важких умовах, наприклад в космосі, може бути значною.

У важких умовах роботи апаратури механізми відмови є часто залежними, зумовленими однією і тією ж зовнішньою причиною. Тому поряд з відомими імовірнісними методами теорії надійності для відмовостійких систем представляє інтерес детермінований підхід. В якості міри відмовостійкості при детермінованому підході служить d-стійкість - максимальне число d елементів або інших структурних одиниць системи, відмова яких ще не тягне за собою відмову системи.

Серед перераховані вище методів забезпечення та підвищення надійності найбільш перспективними є використання нових способів

відновлення, автоматичної реконфігурації та створення відмовостійких КС. Розглянемо ці питання.

Реконфігурація КС - зміна складу і способу взаємодії програмних і апаратних засобів системи з метою виключення відмовивших програмних або апаратних компонентів. Реконфігурація проводиться після виявлення відмови. Розрізняють статичну і динамічну реконфігурацію.

Статична реконфігурація системи здійснюється шляхом відключення несправних компонентів КС. Динамічна реконфігурація за принципом проведення ділиться на наступні види: заміщення, дублювання, поступова «деградація».

Після реконфігурації для продовження нормальної роботи системи необхідно її відновити, відновлення системи відбувається на двох рівнях:

1. Апаратний рівень. Тут проводиться відновлення відмовивших компонентів КС двома способами:
   - автоматичне відновлення, яке реалізується шляхом реконфігурації системи. При цьому передбачається, що в системі є ряд запасних блоків, завдяки яким вона повертається в працездатний стан;
   - ремонт (відновлення вручну). В цьому випадку відмовивший блок відключається від системи і вона продовжує роботу з меншою продуктивністю, або призупиняється до повернення відремонтованого блоку в активну частину КС.
2. Програмний рівень, тут здійснюється відновлення інформації про стан КС, необхідної для продовження її роботи.

Засоби відновлення включаються в роботу при виявленні помилки системою контролю. Спосіб відновлення залежить від рівня, на якому виявлено помилку. Залежно від порушень в роботі системи можна виділити наступні способи відновлення: маскування, повторення операції, повернення для виміру тис

Маскуванням називається виправлення помилки за допомогою коригувальних кодів або резервування. Відновлення шляхом маскування або

повторення операції виконується в тому випадку, коли помилку виявлено засобами контролю логічного рівня і, отже, не встигла поширитися.

Відновлення шляхом повторення операції може бути успішним, якщо помилка була випадковою і самоусунулася при повторенні, тобто проявилася як збій. Оскільки тривалість випадкової помилки може бути різною, система повинна повторювати операції кілька разів. Повторення може бути на рівні мікрокоманд, команд і операцій вводу-виводу.

В тих випадках, коли помилка виявляється засобами функціонального або системного рівня, тобто встигла спотворити інформацію, використовується відновлення шляхом повернення до контрольної точки.

Контрольною точкою називається деяка точка в обчислювальному процесі (програмі), для якої збережені проміжні результати обчислень, і до якої в разі помилки можна повернутися (передати керування). Цей спосіб відновлення вимагає по ходу обчислювального процесу формування контрольних точок (тобто періодичного запам'ятовування проміжних результатів обчислювального процесу).

Програмним рестартом називається повторний запуск обчислювального завдання. Цей спосіб відновлення використовується в тому випадку, коли помилку виявлено засобами контролю призначеного для користувача рівня. Маскування і повторення операції як способи відновлення більш кращі, тому що забезпечують раннє виявлення помилки і відновлення. Однак вони вимагають значних витрат апаратури.

Збереження працездатності комп'ютерів при відмовах має велике значення при експлуатації систем, що працюють в реальному масштабі часу, систем з поділом часу, систем телеобробки, діалогових систем та ін.

Автоматичне відновлення обчислювального процесу при відмовах може бути досягнуто шляхом введення в КС властивостей відмовостійкості.

Автоматичне відновлення в комп'ютерних системах є новим підходам, що забезпечує високу ступінь надійності, готовності та відмовостійкості КС.

Відновлення може виявитися безуспішним у разі наявності помилки в програмах, руйнування інформації в контрольних точках, вичерпання резервів або зниження продуктивності системи через відмови.

Тема 10: Високопродуктивні кластерні системи

Кластерні системи - це КС MIMD-архітектури, вони є розвитком архітектур з масовим паралелізмом.

Кластерна система являє собою однорідну КС у вигляді сукупності комп'ютерів, об'єднаних в єдину мережу для вирішення єдиного завдання. Користувач розглядає цю мережу в якості єдиного ресурсу.

Кластерні системи відрізняються застосуванням стандартних рішень в апаратурі і програмному забезпеченні. Використання компонент звичайних комп'ютерних систем (центральні процесори, операційні системи) в поєднанні з недорогими комунікаційними пристроями сприяли появі і бурхливому розвитку відносно дешевих кластерних систем. Вперше їх представила в 1983 р компанія DEC.

Кластери використовується для обробки великих обсягів даних при вирішенні складних наукових або графічних завдань. Складовим елементом кластерної системи є самостійний комп'ютер з процесором, пам'яттю, підсистемою вводу-виводу, своєю операційною системою і додатками. Зазвичай для цих цілей використовується персональні комп'ютери, об'єднані в єдину архітектуру із загальною (SMP) або розподіленою (MPP) пам'яттю. Кластери є слабозв'язаними системами, для зв'язку вузлів використовуються мережеві технології на базі шинної архітектури (рис. 9.1).

Кластер, що складається з двох або більше вузлів задовольняє таким умовам:

- кожен вузол працює зі своєю копією ОС;
- кожен вузол працює зі своєю копією додатка;
- вузли ділять загальний пул інших ресурсів, таких як накопичувачі на дисках і ін.

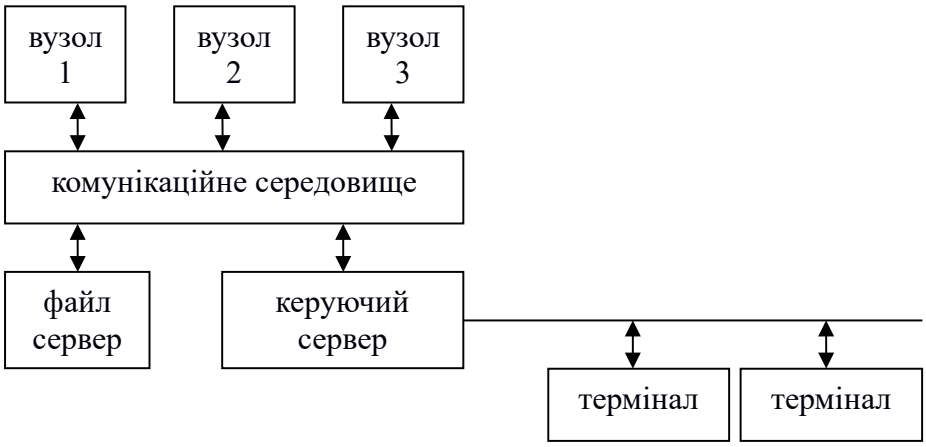


Рис. 9.1. Шинна архітектура кластерів.

Застосування стандартних технічних і програмних засобів робить кластерні системи дешевими. Тому комп'ютерні системи, створювані з масово випускаючих компонентів, стали альтернативою традиційним суперкомп'ютерним системам. Вони демонструють продуктивність, що не поступається продуктивності комп'ютерів як із загальною, так і з розподіленою пам'яттю. При цьому кластерні системи мають ряд переваг: більш низька вартість, короткий цикл розробки, використання звичних обчислювальних, комунікаційних і програмних компонентів.

Отже, кластером називається паралельна комп'ютерна система з розподіленою пам'яттю, побудована з компонентів загального призначення, з єдиним вхідним доступом, однорідними обчислювальними вузлами і мережевою архітектурою, що забезпечує ефективний обмін даними.

Характерні особливості кластерних систем.

1. Кластер розглядається як єдина комп'ютерна система, доступ до ресурсів якої, керування і адміністрування здійснюється через одну вхідну точку доступу, при цьому ресурси кластера повинні використовуватися тільки в монопольному режимі.
2. Єдиною точкою доступу є керуючий сервер, його засоби віддаленого доступу або консоль. На керуючому сервері встановлюються програмні засоби, що забезпечують моніторинг і керування

ресурсами кластера. Керуючий сервер при необхідності може здійснювати функції файлового сервера.

3. Апаратура обчислювальних вузлів повинна бути ідентична, а їх програмне забезпечення (операційна система, керування кластером, комунікаційна бібліотека програм) повинна бути сумісним і однаково налаштованим.
4. Комунікаційні компоненти обчислення і керування повинні бути розв'язані, тобто повинна бути як комп'ютерна мережа, так і виділене комунікаційне середовище. Комунікаційна частина системи повинна володіти високою пропускною здатністю при передачі довгих повідомлень, мати ефективні протоколи обміну.
5. Кластерні системи мають властивість зростання продуктивності при додаванні ресурсів (масштабованість).

Тісна взаємодія комп'ютерів, що утворюють кластер (вузли кластера), гарантує максимальну продуктивність і мінімальний час простою додатків за рахунок того, що:

- в разі збою програмного забезпечення на одному вузлі додаток продовжує функціонувати (або автоматично перезапускається) на інших вузлах кластера;
- відмова вузла кластера з будь-якої причини (включно з помилками персоналу) не означає відмови кластера в цілому;
- профілактичні та ремонтні роботи, реконфігурацію і зміну версій програмного забезпечення в більшості випадків можна здійснювати на вузлах кластера по черзі, не перериваючи роботи додатків на інших вузлах кластера.

Наявність в кластерних системах комунікаційного каналу необхідно для:

- скоординованого використання загальнокластерних ресурсів;
- взаємного контролю працездатності;

- обміну інформацією про конфігурацію кластера і іншою специфічною «кластерною» інформацією.

Саме розвиток комунікаційних технологій і створення високошвидкісного мережевого обладнання, що реалізує механізм передачі повідомлень за стандартними протоколами, зробили кластерні системи конкурентоспроможними. Це пов'язано з тим, що продуктивність систем з розподіленою пам'яттю сильно залежить від латентності (часу затримки при передачі повідомлень) і пропускної здатності комунікаційного середовища.

Для керування системою, організації доступу до неї і максимального використання її обчислювального ресурсу в кластерній системі використовується потужний керуючий сервер забезпечений необхідною периферією.

Розвиток кластерних систем нерозривно пов'язаний з розвитком мережевих технологій. Кластери в високопродуктивних обчисленнях - це практично завжди суперкомп'ютери. Різних варіантів побудови кластерів дуже багато. Одна з основних відмінностей лежить у використовуваній мережевій технології, вибір якої визначається, перш за все класом вирішуваних завдань.

Комп'ютерні системи як потужні засоби обробки інформації завдань користувачів широко використовуються не тільки автономно, але і в комп'ютерних мережах в якості серверів. Зі збільшенням розмірів мереж і їх розвитком зростають щільність інформаційних потоків, навантаження на засоби доступу до мережевих ресурсів і на засоби обробки завдань. Коло завдань, що вирішуються серверами, постійно розширюється, стає різноманітним і складним. Чим вище ранг мережі, тим більше спеціалізованими вони стають.

Одним з перспективних напрямків є кластеризація, тобто технологія, за допомогою якої кілька серверів, які самі є комп'ютерними системами, об'єднуються в єдину систему більш високого рангу для підвищення ефективності функціонування системи в цілому.

Цілями побудови кластерів можуть служити:

- поліпшення масштабованості (здатність до нарощування потужності);
- підвищення надійності та готовності системи в цілому;
- збільшення сумарної продуктивності;
- ефективний перерозподіл навантажень між комп'ютерами кластера;
- ефективне керування і контроль роботи системи і т.п.

Поліпшення масштабованості або здатність до нарощування потужності передбачає, що всі елементи кластера мають апаратну, програмну та інформаційну сумісність. У поєднанні з простим і ефективним керуванням зміна обладнання в ідеальному кластері має забезпечувати відповідну зміну значень основних характеристик, тобто додавання нових процесорів, дискових систем повинно супроводжуватися пропорціальним зростанням продуктивності, надійності і т.п. У реальних системах ця залежність має нелінійний характер.

Наступною важливою метою створення кластерів є підвищення надійності та готовності системи в цілому. Саме ці якості сприяють популярності і розвитку кластерних структур. Надмірність, спочатку закладена в кластери, здатна їх забезпечити. Основою цього є можливість кожного сервера кластера працювати автономно, але в будь-який момент він може переключитися на виконання робіт іншого сервера в разі відмови.

Більшість сучасних серверів має 99%-ву готовність. Це означає, що близько чотирьох днів в році вони не працюють. Готовність 99,9, що досягається зазвичай з паркою серверів - основного і резервного, означає річний простій рівний близько 500 хв., 99,9999 - 5 хв., а 99,99999 - 30 с.

Поява критично важливих додатків в областях бізнесу, фінансів, телекомунікацій, охорони здоров'я та ін. вимагає забезпечення готовності не менше ніж «п'ять дев'яток» і вище.

Підвищення сумарної продуктивності кластера, що об'єднує кілька серверів, забезпечується автоматично. Адже кожен сервер кластера сам є

досить потужною КС, розрахованою на виконання ним усіх необхідних функцій в частині керування відповідними мережевими ресурсами. З розвитком мереж все більшого значення набувають розподілені обчислення.

Сукупні обчислювальні потужності кластерів можуть бути порівнянні з потужностями суперкомп'ютерів і навіть перевищувати їх при незмірно меншій вартості. Такі технології стосовно до окремих класів задач добре відпрацьовані.

Робота кластера під керуванням єдиної операційної системи дозволяє оперативно контролювати процес обчислень і ефективно перерозподіляти навантаження між комп'ютерами кластера.

Керування такими проектами вимагає створення спеціального клієнтського і серверного програмного забезпечення, що працює у фоновому режимі. Комп'ютери при цьому періодично отримують завдання від сервера, включаються в роботу і повертають результати обробки.

Ефективне керування і контроль роботи системи має на увазі можливість роботи окремо з кожним вузлом, вручну або програмно відключати його для модернізації або ремонту з подальшим поверненням його в працюючий кластер. Ці операції приховані від користувачів, вони просто не помічають їх. Кластерне ПЗ інтегроване в операційні системи серверів, дозволяє працювати з вузлами як з єдиним пулом ресурсів, вносячи необхідні загальні зміни за допомогою однієї операції для всіх вузлів.