Тема 13. Класифікація архітектур паралельних комп’ютерних систем

В умовах постійно зростаючих вимог до продуктивності обчислювальної техніки все більш явними стають обмеження класичної фон-нейманівської архітектури. Подальший розвиток обчислювальної техніки зв'язаний з переходом до паралельних обчислень як у рамках однієї обчислювальної машини, так і шляхом створення багатопроцесорних систем і мереж, які об'єднують велику кількість окремих процесорів або окремих обчислювальних машин. Для такого підходу замість терміну «обчислювальна машина» більш підходить термін «обчислювальна система» (ОС). Головною особливістю такої системи є наявність у ній засобів, які реалізують паралельну обробку інформації.

13.1 Рівні паралелізму

Паралельна обробка інформації являє собою одночасне рішення двох або більшої кількості частин однієї й тієї ж програми двома чи більшою кількістю ЕОМ (процесорними елементами) обчислювальної системи. Реалізують три основних способи організації паралельної обробки:

1) суміщення у часі різноманітних етапів різних задач – це мульти-програмна обробка, яка широко використовується як у однопроцесорних ЕОМ, так і в складних ОС;

2) одночасне розв'язання різноманітних задач або частин однієї задачі (можливо тільки за наявності декількох обробляючих пристроїв);

3) конвеєрна обробка інформації.

Перші два способи використовують особливості паралельних задач або потоків задач, що дозволяє здійснювати той або інший паралелізм. Перший тип паралелізму – це природний паралелізм незалежних задач, який полягає у тому, що в систему поступає безперервний потік не зв'язаних між собою задач. У цьому випадку розв'язання будь-якої задачі не залежить від результатів розв'язання інших задач, що дозволяє підвищити продуктивність ОС у разі використання декількох обробляючих пристроїв.

Одним з найбільш розповсюджених типів паралелізму є паралелізм незалежних гілок. Суть його полягає у виділенні окремих незалежних частин великої задачі (гілок програми), які можуть виконуватись паралельно окремими обробляючими пристроями незалежно один від одного. Причому обробляючі пристрої ОС функціонують в однопрограмному режимі паралельної обробки інформації. Двома незалежними гілками програми визнаються гілки програми, що відповідають таким умовам:

- ні одна з вхідних для гілки програми величин не є вихідною величиною іншої програми (відсутність функціональних зв'язків);
- для двох гілок програми не повинен здійснюватись запис у одні й ті ж комірки пам'яті (відсутність зв'язку по оперативній пам'яті);
- умови виконання однієї гілки не залежать від результатів, що отримані під час виконання іншої гілки (незалежність по управлінню);
- обидві гілки повинні виконуватись у різних блоках програми (програмна незалежність).

Виділення незалежних гілок широко використовується під час паралельної обробки в задачах матричної алгебри, лінійного програмування, спектральної обробки сигналів, прямого та зворотного перетворення Фур'є та ін.

Методи та засоби реалізації паралелізму залежать від того, на якому рівні він повинен забезпечуватись. Зазвичай розрізняють такі рівні паралелізму:

- рівень завдань – декілька незалежних завдань одночасно виконуються на різних процесорах, які практично не взаємодіють один з одним. Цей рівень реалізується в ОС з множиною процесорів у багатозадачному режимі.
- рівень програм – частини однієї задачі виконуються множиною процесорів. Даний рівень досягається на паралельних ОС.
- рівень команд – виконання команди розділяється на фази, а фази декількох послідовних команд можуть бути перекриті за рахунок конвеєризації. Даний рівень використовується в ОС з одним процесором.

– рівень бітів (арифметичний рівень) – біти слова обробляються один за одним. Цей процес має назву біт-послідовна операція. Якщо біти слова обробляються одночасно, кажуть про біт-паралельну операцію. Даний рівень реалізується в звичайних і суперскалярних процесорах.

Паралелізм рівня завдання можливий між незалежними завданнями або їх фазами. Основним засобом реалізації паралелізму на рівні завдань є багатопроцесорні і багатомашинні обчислювальні системи, в яких завдання розподіляються за окремими процесорами або машинами. Однак, якщо кожне завдання трактувати як сукупність незалежних задач, реалізація даного рівня можлива і в рамках однопроцесорної ОС. У цьому випадку декілька завдань можуть одночасно знаходитись в основній пам'яті ОС, за умови, що в кожний момент виконується тільки одне з них. Коли завдання, яке виконується, потребує вводу/виводу, ця операція запускається, а до її завершення інші ресурси ОС передаються другому завданню. По завершенні вводу/виводу ресурси ОС повертаються до завдання, яке ініціювало цю операцію. В цьому випадку паралелізм забезпечується за рахунок того, що центральний процесор і система вводу/виводу функціонують одночасно і обслуговують різні завдання.

Паралелізм виникає також, коли у незалежних завдань, які виконуються в ОС, є декілька фаз, наприклад обчислення, запис у графічний буфер, системні виклики. За те, як різні завдання впорядковуються і витрачають загальні ресурси відповідає операційна система.

Паралелізм рівня програм. Про паралелізм на рівні програми можна говорити у двох випадках. По перше, коли в програмі можна виділити незалежні ділянки, які допустимо виконувати паралельно. Другий тип паралелізму програм можливий у межах окремого програмного циклу, якщо в ньому окремі ітерації не залежать один від одного. Програмний паралелізм можна реалізувати за рахунок великої кількості процесорів або множини функціональних блоків.

Загальна форма паралелізму на рівні програм організується розбиттям даних, які програмуються на підмножини. Цей розподіл називають декомпозицією області (domain decomposition), а паралелізм, який при цьому

виникає, має назву паралелізму даних. Підмножини даних призначаються різним обчислювальним процесам, і цей процес має назву розподілення даних (data distribution). Процесори виділяються певним процесам або за ініціативою програми, або в процесі роботи операційною системою. На кожному процесорі може виконуватись більше одного процесу.

Паралелізм рівня команд. Паралелізм на рівні команд має місце, коли обробка декількох команд або виконання різних етапів однієї і тієї ж команди може перекриватись у часі. Розробники обчислювальної техніки ще здавна зверталися до методів, відомих під загальною назвою «поєднання операцій», при якому апаратура ОМ у будь-який момент часу виконує одночасно більше однієї операції. Цей загальний принцип включає два поняття: паралелізм і конвеєризацію. Хоча у них багато загального і їх часто важко розрізняти на практиці, ці терміни відображають два принципово різних підходи.

Під час організації паралельного обчислювального процесу виникає задача вибору побудови розкладу.

Розклад паралельних обчислювальних процесів визначає порядок виконання програми в ОС, включаючи розподіл частин програми по обробляючих пристроях (процесорах, ОМ), і служить основою алгоритмів планувальника операційної системи та різноманітних управляючих програм. Як критерії оптимальних розкладів для паралельної програми можна назвати: мінімізацію часу виконання програми; мінімізацію кількості потрібних пристроїв обробки; мінімізацію середнього часу закінчення виконання завдань; максимізацію завантаження пристроїв ОС; мінімізацію часу простоювання пристроїв.

Найчастіше використовують перший критерій.

13.2 Класифікація архітектур комп'ютерних систем

Існує декілька класифікацій архітектур комп'ютерних систем. Найбільш вдалою є класифікація М. Флінна.

Архітектурні особливості комп'ютерних систем описано з погляду потоку команд (інструкцій) та потоку даних. Такий підхід дає змогу відносити архітектури комп'ютерів до одного з чотирьох класів (табл. 13.1, рис. 13.1).

Класифікація здійснена з погляду не структури, а того, як у комп'ютері його машинні команди взаємодіють з даними.

Таблиця 6.1 – Класифікація архітектур комп'ютерних систем

Потік команд	Одиничний потік даних (ОД)	Множинний потік даних (МД)
Одиничний (ОК)	ОКОД (SISD) (однопроцесорні комп'ютери)	ОКМД (SIMD) (комп'ютери з паралельними або асоціативними процесорами)
Множинний (МК)	МКОД (MISD) (конвеєрні магістральні комп'ютери)	МКМД (MIMD) (багатопроцесорні або багатомашинні комплекси)

SISD (Single Instruction Stream/Single Data Stream) - одиничний потік команд і одиничний потік даних. Представник цього класу – класичний фон-нейманівський комп'ютер. Команди обробляються послідовно і кожна команда ініціює одну операцію з одним потоком даних. До цього класу комп'ютерів можна віднести також конвеєрні комп'ютери. Деякі спеціалісти вважають, що до SISD-систем можна віднести і векторно-конвеєрні ОС, якщо розглядати вектор як неподільний елемент даних для відповідної команди.

MISD (Multiple Instruction Stream/Single Data Stream) – множинний потік команд і одиничний потік даних. В архітектурі присутня множина процесорів, які обробляють один і той же потік даних. Ряд дослідників відносять до даного класу конвеєрні системи. Прийнято вважати, що доки даний клас незадіяний, однак може бути корисним для розробки принципово нових концепцій побудови обчислювальних систем [25].

SIMD (Single Instruction Stream/Multiple Data Stream) – одиничний потік команд і множинний потік даних. Ця архітектура дозволяє виконати одну арифметичну операцію відразу над багатьма даними – елементами вектору.

Представниками цього класу є системи з матрицею процесорів, де один управляючий пристрій контролює множину процесорних елементів. Усі процесорні елементи отримують від пристрою управління однакову команду і виконують її над власними локальними даними. В цей клас можуть бути включеними і векторно-конвеєрні ОС, якщо кожний елемент вектора розглядати як окремий елемент даних.

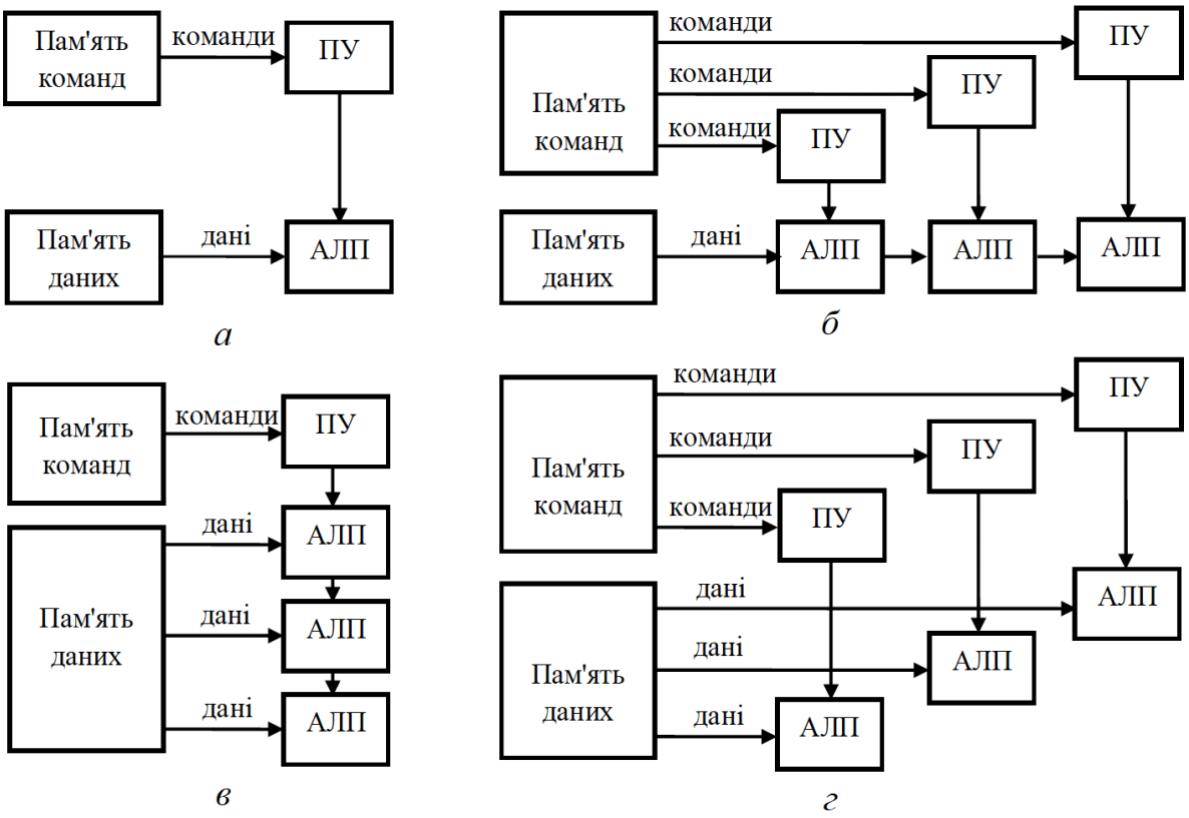

Рисунок 13.1 – Архітектура комп'ютерних систем за Флінном: а – SISD; б – MISD; в – SIMD; г – MIMD

MIMD (Multiple Instruction Stream/Multiple Data Stream) – множинний потік команд і множинний потік даних. До цього класу відносяться системи з множиною пристроїв обробки команд, які об'єднані в єдиний комплекс і кожний працює з власним потоком команд. Цей клас надзвичайно широкий, бо включає в себе різного роду мультипроцесорні системи.

Класифікація Флінна надто загальна, вона має деякі недоліки, наприклад відносить усі паралельні комп'ютери, крім мультипроцесорних, до одного класу і не вказує ніякої відмінності між конвеєрними комп'ютерами і матрицею МП.

Використовуються й інші класифікації архітектур, зокрема систематика Ф. Шара, структурна систематика Р. Хокні та К. Джессхоупа.

13.3 Обчислювальні системи класу SIMD (ОКМД)

SIMD-системи були першими обчислювальними системами, що складаються з великого числа процесорів, і серед перших систем, де була досягнута продуктивність порядку GFLOPS. Згідно з класифікацією Флінна, до класу SIMD відносяться ОС, де множина елементів даних піддається паралельній, але однотипній обробці.

На рис. 13.2 показані приблизні характеристики продуктивності деяких типів обчислювальних систем класу SIMD.

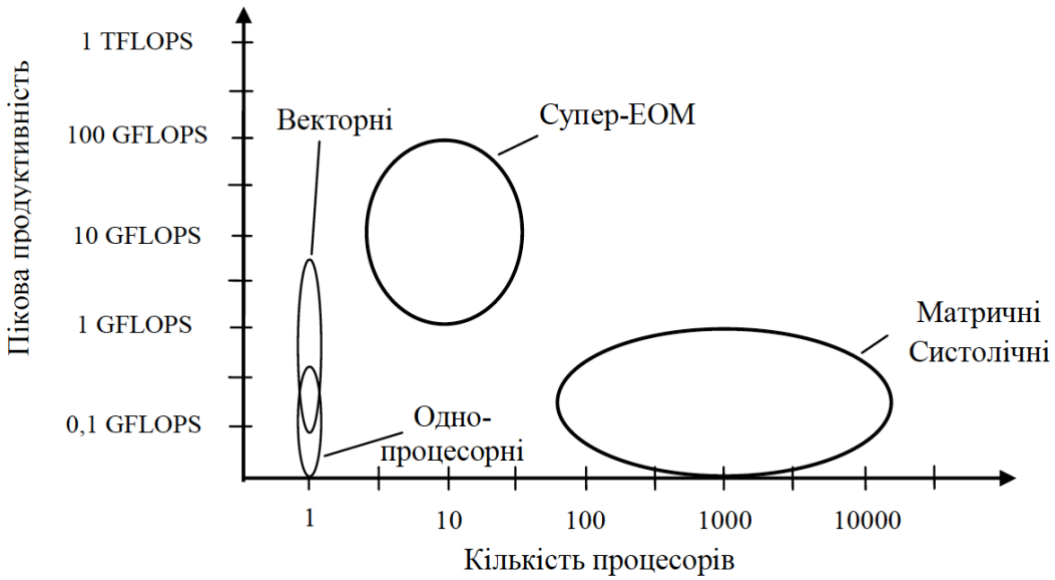

Рисунок 13.2 – Продуктивність SIMD-систем як функція їх типу та кількості процесорів

SIMD-системи багато в чому схожі на класичні фон-нейманівські ОМ: у них також є один пристрій управління, який забезпечує послідовне виконання команд програми. Відмінність стосується стадії виконання, коли загальна

команда транслюється великій кількості процесорів (у простому випадку – АЛП), кожен з яких обробляє свої дані.

Раніше вже наголошувалася нечіткість класифікації Флінна, через що різні типи ОС можуть бути віднесені до того або іншого класу. Проте в теперішній час прийнято вважати, що клас SIMD складають векторні (векторно-конвеєрні), матричні, асоціативні, систолічні і VLIW-обчислювальні системи.

13.3.1 Векторні і векторно–конвеєрні комп'ютерні системи

Під час розв'язання на комп'ютері науково-технічних та інших задач часто зустрічається необхідність визначення значень однієї і тієї ж функції для групи даних, розташованих у комірках пам'яті (регістрах) з упорядкованими адресами (номерами). Такі набори даних називаються векторами.

Структурна схема векторного процесора показана на рис. 13.3.

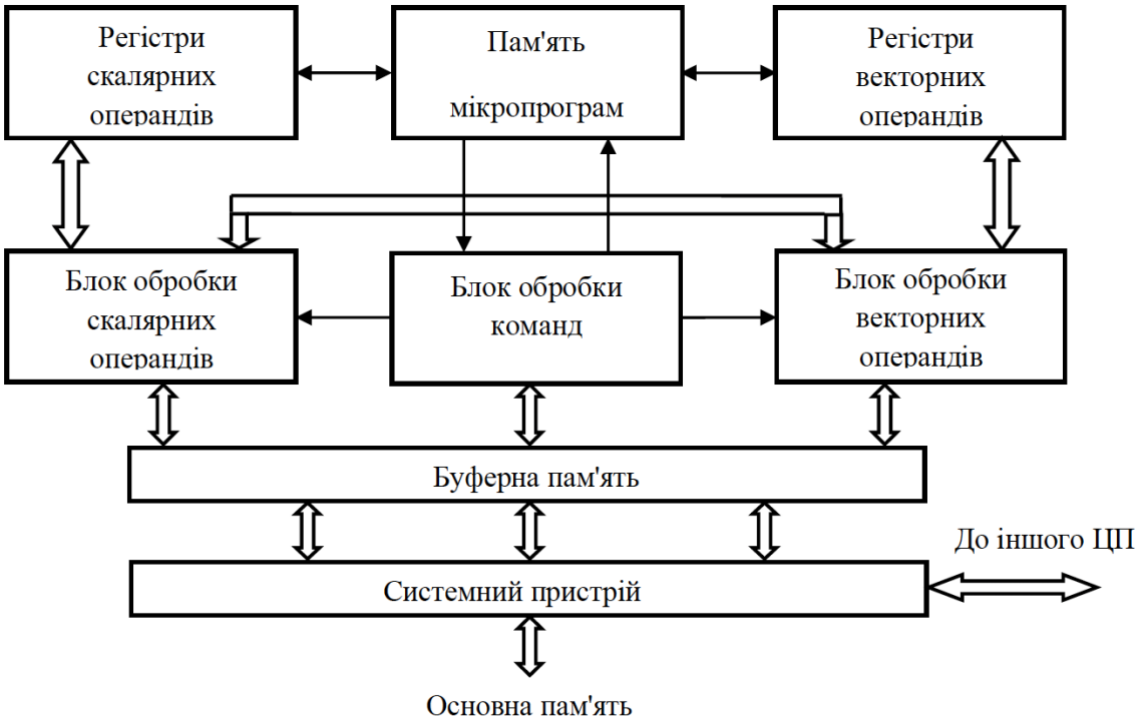

Рисунок 13.3 – Структурна схема векторного процесора

З метою підвищення продуктивності комп'ютера до складу його процесора разом зі скалярними засобами включають засоби векторної обробки даних або

вводять спеціалізований векторний співпроцесор. Векторні засоби включають векторні команди, векторні регістри та іншу апаратуру для реалізації цих команд.

Засоби векторної обробки дозволяють за допомогою єдиної команди виконувати дії над усіма елементами масивів. Кожний елемент такого масиву (вектор) обробляється незалежно від інших елементів на конвеєрному операційному пристрої (ОП), який за рахунок спеціальної структурної організації може приймати в кожному такті пару елементів векторів-операндів і через деякий час, який називається стартом конвеєра, видавати відповідний елемент вектора-результату. Оскільки пари елементів векторів-операндів надходять у конвеєрний ОП в кожному такті, то через час, який визначає довжину часу старту конвеєра після запуску векторної операції, з виходу ОП кожного такту будуть видаватись елементи вектора-результату.

До апаратних векторних засобів відносяться:

- арифметичні конвеєри (кожний такий конвеєр може містити декілька незалежних спеціалізованих конвеєрних арифметичних пристроїв);
- векторні регістри, які зберігають векторну інформацію, що використовується для обробки в арифметичних конвеєрах;
- конвеєри завантаження-запису, які виконують обмін інформацією між векторними регістрами і оперативною пам'яттю ЕОМ.

У ході розв'язання реальних задач на продуктивність ЕОМ істотно впливають:

- швидкість обробки скалярних величин і параметри векторного пристрою. Залежності збільшення продуктивності ЕОМ від коефіцієнта векторизації при зміні параметрів векторних засобів обробки приведені на рис. 13.4;
- об'єм векторних регістрів. У разі недостатнього об'єму векторні регістри не можуть згладити нерівномірність інформаційного потоку, що обробляється, невизначено збільшується інтенсивність обміну між векторними арифметичними конвеєрами і оперативною пам'яттю.

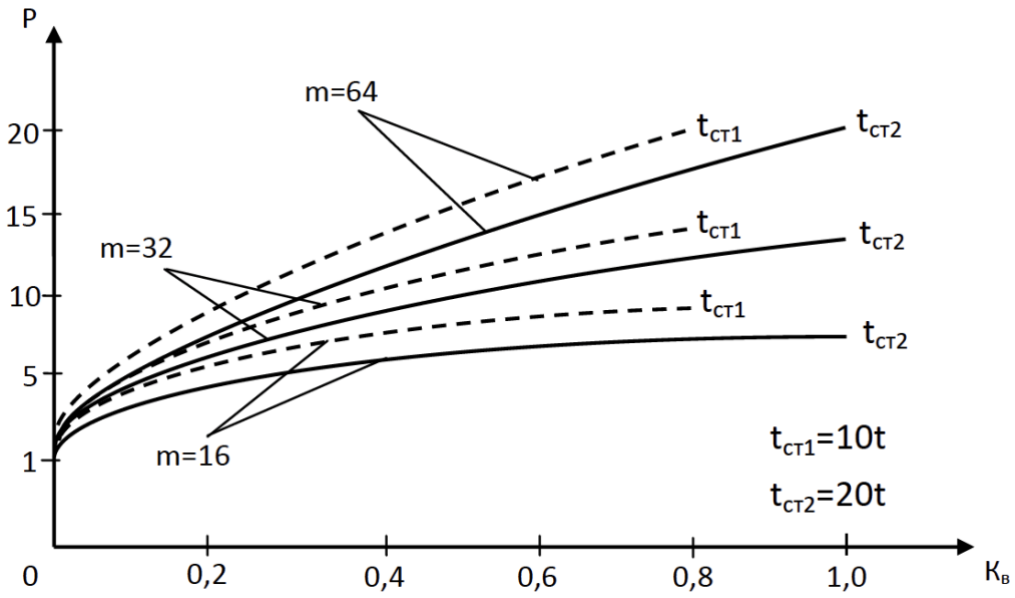

Рисунок 13.4 – Залежності P($k_в$, m)

Векторні команди дозволяють однією командою дати розпорядження на виконання деякої операції (векторної операції) над елементами вектора, наприклад, поелементне додавання векторів та організовують і управління обчислювальним циклом.

У машині, яка має векторні команди, виконуються також звичайні команди над одиничними даними (скалярні команди).

Наявність векторних команд сприяє підвищенню швидкодії процесора за рахунок зменшення витрат часу на організацію обчислювального циклу.

Звичайно, з метою досягнення підвищеної швидкодії виконання самих векторних операцій ставиться на конвеєр, до того ж може бути кілька арифметичних конвеєрів (ліній), а окремі пристрої конвеєрної лінії можуть, у свою чергу, містити конвеєри для виконання покладених на них підфункцій.

Векторні команди (звичайно їх порівняно мало) мають ряд особливостей. Їхній код операції не тільки задає операцію, яка підлягає виконанню, але і визначає необхідну конфігурацію активних частин конвеєра для даної операції. Команда вказує початок вектора (векторів) в пам'яті (або блоці регістрів), довжину вектора, розмір і тип елементів вектора (ціле число, число з плаваючою комою і т.д), визначає місце знаходження вектора-результату.

Прикладами конвеєрно-векторних процесорів можуть служити спеціалізовані на виконання векторних і матричних операцій співпроцесори, при цьому продуктивність машини під час виконання векторних і матричних операцій збільшується до 30 Мфлоп/с.

Конвеєрно-векторні ОС в теперішній час є одним з основних напрямків у разі утворення супер-ЕОМ для широкого кола задач. До ОС такого типу відносяться найшвидкодіючі супер-ЕОМ сімейства CRAY (США).

13.3.2 Матричні комп'ютерні системи

Призначення матричних комп'ютерних систем багато в чому схоже з призначенням векторних комп'ютерних систем – обробка великих масивів даних. В основі матричних систем лежить матричний процесор, який складається з регулярного масиву процесорних елементів (ПЕ). Організація системи такого типу, на перший погляд, достатньо проста. Вона має загальний управляючий пристрій, який генерує потік команд, та велику кількість ПЕ, які функціонують паралельно і обробляють кожний свій потік даних. Проте на практиці, щоб забезпечити достатню ефективність системи у ході розв'язання широкого кола задач, необхідно організовувати зв'язки між процесорними елементами так, щоб найбільш повно завантажити процесори роботою. Саме характер зв'язків між ПЕ і визначає різні властивості системи.

Матричний процесор інтегрує множину ідентичних функціональних блоків (ФБ), які логічно об'єднуються в матрицю і працюють у SIMD-стилі. Матриця процесорних елементів може бути конструктивно реалізована на одному кристалі або на декількох. Важливим є принцип функціонування – ФБ логічно скомпоновані у матрицю і функціонують синхронно, тобто існує тільки один потік команд для усіх блоків.

Структуру матричної комп'ютерної системи можна подати у виді, який показаний на рис 13.5.

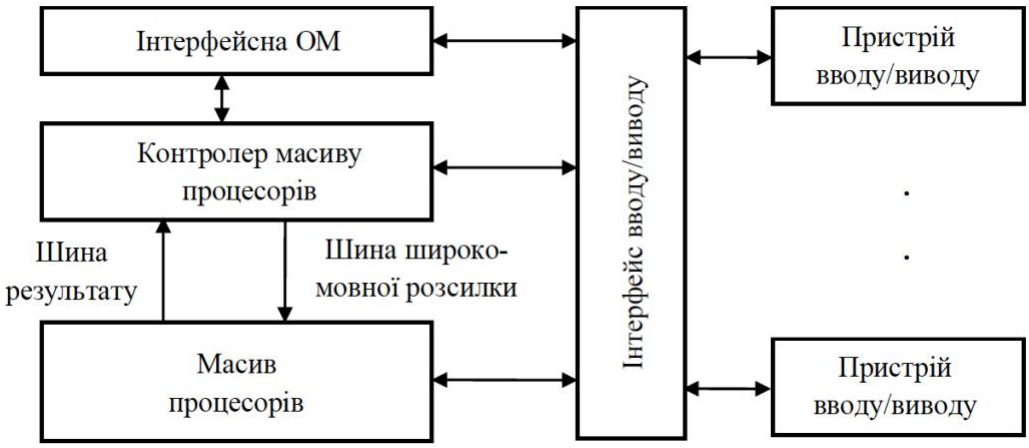

Рисунок 13.5 – Узагальнена модель матричної SIMD-системи

Власне паралельна обробка множинних елементів даних виконується масивом процесорів (МПр). Єдиний потік команд, який управляє обробкою даних у масиві процесорів, генерується контролером масиву процесорів (КМПр). КМПр виконує послідовний програмний код, реалізує операції умовного і безумовного переходів, транслює в МПр команди, дані та сигнали управління. Команди обробляються процесорами в режимі жорсткої синхронізації. Сигнали управління використовуються для синхронізації команд і пересилок, а також для управління процесом обчислень, зокрема визначають, які процесори масиву повинні виконувати операцію, а які – ні. Команди, дані і сигнали управління передаються із КМПр у масив процесорів по шині широкомовної розсилки. Оскільки виконання операцій умовного переходу залежить від результатів обчислень, результати обробки даних у масиві транслюються в КМПр по шині результату.

Для забезпечення користувача зручним інтерфейсом під час створення та налаштування програм у склад подібних комп'ютерних систем звичайно включають інтерфейсну обчислювальну машину (ІОМ). Як таку обчислювальну машину використовують універсальну обчислювальну машину, на яку додатково покладається задача завантаження програм і даних в КМПр. Інтерфейсна обчислювальна машина функціонує під управлінням операційної системи, частіше це ОС UNIX. На ІОМ користувачі готують, компілюють і налаштовують власні програми. У процесі виконання програми спочатку

завантажуються із інтерфейсної обчислювальної машини в контролер управління масивом процесорів, який виконує програму і розподіляє команди і дані по процесорних елементах масиву.

Крім того, завантаження програм і даних в КМПр може виконуватись і безпосередньо з пристроїв вводу/виводу, наприклад з магнітних дисків. Після завантаження КМПр приступає до виконання програми, транслюючи в МПр по широкомовній шині відповідні SIMD-команди.

Розглядаючи масив процесорів, слід враховувати, що для зберігання множинних наборів даних у ньому, крім множини процесорів, повинна бути присутньою і множина модулів пам'яті. Крім того, в масиві повинна бути реалізована мережа взаємозв'язків як між процесорами, так і між процесорами і модулями пам'яті. Таким чином, під терміном масив процесорів розуміють блок, який складається з процесорів, модулів пам'яті і мережі з'єднань.

У матричних SIMD-системах розповсюдження отримали два основних типи архітектурної організації масиву процесорних елементів.

У першому варіанті, який відомий як архітектура типу «процесорний елемент – процесорний елемент», N процесорних елементів (ПЕ) зв'язані між собою мережею з'єднань. Кожний ПЕ – це процесор з локальною пам'яттю. Процесорні елементи виконують команди, які отримують від КМПр по шині широкомовної розсилки, та обробляють дані як ті, що зберігаються у їх локальній пам'яті, так і ті, що потрапляють з КМПр. Обмін даними між процесорними елементами здійснюється по мережі з'єднань, в той час як шина вводу/виводу служить для обміну інформацією між ПЕ і пристроями вводу/виводу. Для трансляції результатів з окремих ПЕ в контролер масиву процесорів служить шина результату. В багатьох алгоритмах дії по пересиланню інформації в більшості локальні, тобто виконуються між найближчими сусідами, тому дана архітектура, де кожний ПЕ зв'язаний тільки з сусідніми, достатньо ефективна.

Другий вид архітектури – «процесор – пам'ять». У такій архітектурі двонаправлена мережа з'єднань зв'язує N процесорів з M модулями пам'яті. КМП управляє процесорами через широкомовну шину. Обмін даними між

процесорами здійснюється як через мережу, так і через модулі пам'яті. Пересилка даних між модулями пам'яті та пристроями вводу/виводу забезпечується шиною вводу/ виводу. Для передачі даних з певного модуля пам'яті в КМП служить шина результату.

Додаткову гнучкість під час роботи з матричною обчислювальною системою забезпечує механізм маскування, який дозволяє залучати до участі в операціях тільки певну підмножину процесорів з тих, які входять у масив. Маскування реалізується як на стадії компіляції, так і на етапі виконання, при цьому ті процесори, які виключаються шляхом встановлення в нуль відповідних бітів маски, протягом виконання команди простоюють.

13.3.3 Комп'ютерні системи з систолічною структурою

У фон-нейманівських комп'ютерах дані, які зчитуються з пам'яті, однократно обробляються в процесорному елементі, після чого знову повертаються в пам'ять (рис. 13.6, а). Автори ідеї систолічної матриці Кунг і Лейзерсон запропонували організувати обчислення так, щоб дані на своєму шляху від зчитування з пам'яті до повернення назад пропускалися через якамога більшу кількість ПЕ (рис. 13.6, б).

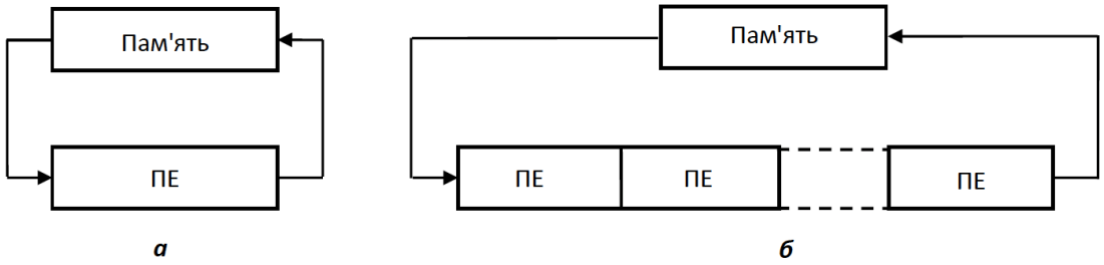

Рисунок 13.6 – Обробка даних у комп'ютерних системах:
а – фон-нейманівського типу; б – систолічної структури

Якщо порівняти положення пам'яті в комп'ютерній системі зі структурою живого організму, то за аналогією їй можна відвести роль серця, множині ПЕ – роль тканин, а потік даних розглядати як кров, що циркулює. Звідси і виходить назва систолічна матриця (систола – скорочення передсердя і шлуночків серця,

при якому кров нагнітається в артерії). Систолічні структури ефективні під час виконання матричних обчислень, обробки сигналів, сортування даних і так далі.

Таким чином, систолічна структура – це однорідне обчислювальне середовище з процесорних елементів, яке суміщає в собі властивості конвеєрної та матричної обробки і володіє такими особливостями:

- обчислювальний процес у систолічних структурах являє собою неперервну і регулярну передачу даних від одного ПЕ до іншого без запам'ятовування проміжних результатів обчислення;
- кожний елемент вхідних даних вибирається з пам'яті однократно та використовується стільки разів, скільки необхідно за алгоритмом, введення даних здійснюється в крайні ПЕ матриці;
- ПЕ, які створюють систолічну структуру, однотипні і кожний з них може бути менш універсальним, ніж процесори звичайних багатопроцесорних систем;
- потоки даних та управляючих сигналів володіють регулярністю, що дозволяє об'єднувати ПЕ локальними зв'язками мінімальної довжини;
- алгоритми функціонування дозволяють суміщати паралелізм з конвеєрною обробкою даних;
- продуктивність матриці можна покращувати за рахунок додавання до неї певного числа ПЕ, причому коефіцієнт підвищення продуктивності при цьому є лінійним.

У теперішній час продуктивність систолічних процесорів перевищує 1000 млрд операцій /с.

Аналіз різних типів систолічних структур та тенденцій їх розвитку дозволяє класифікувати ці структури по декільком ознакам.

За ступенем гнучкості систолічні структури можуть бути згруповані так:

- спеціалізовані;
- алгоритмічно орієнтовані;
- програмовані.

Спеціалізовані структури орієнтовані на виконання певного алгоритму. Ця орієнтація відображається не тільки в конкретній геометрії систолічної структури, статичності зв'язків між ПЕ та кількості ПЕ, але і в виборі типу операції, що виконується усіма ПЕ. Прикладами є структури, які орієнтовані на рекурсивну фільтрацію, швидке перетворення Фур'є для заданої кількості точок, матричні перетворення.

Алгоритмічно орієнтовані системи володіють можливістю програмування або конфігурації зв'язків у систолічній матриці або самих ПЕ. Можливість програмування дозволяє виконувати на таких структурах деяку множину алгоритмів, які зводяться до однотипних операцій над векторами, матрицями та іншими числовими множинами.

У програмованих систолічних структурах є можливість програмування як самих ПЕ, так і конфігурації зв'язків між ними. При цьому ПЕ можуть володіти локальною пам'яттю програм. Команди, які зберігаються в пам'яті таких ПЕ, можуть змінювати і напрямок передачі операндів.

За розрядністю процесорних елементів систолічні структури діляться так:

- однорозрядні;
- багаторозрядні.

В однорозрядних матрицях ПЕ в кожний момент часу виконує операцію над одним двійковим розрядом, а в багаторозрядних – над словами фіксованої довжини.

За характером локально-просторових зв'язків систолічні структури бувають:

- одновимірні;
- двовимірні;
- тривимірні.

Вибір структури залежить від виду інформації, що обробляється. Одновимірні структури застосовують при обробці векторів, двовимірні – матриць, тривимірні – множин іншого типу.

У теперішній час розроблені систолічні матриці з різною геометрією звязків: лінійні, прямокутні, гексагональні, тривимірні та ін. (рис. 13.7).

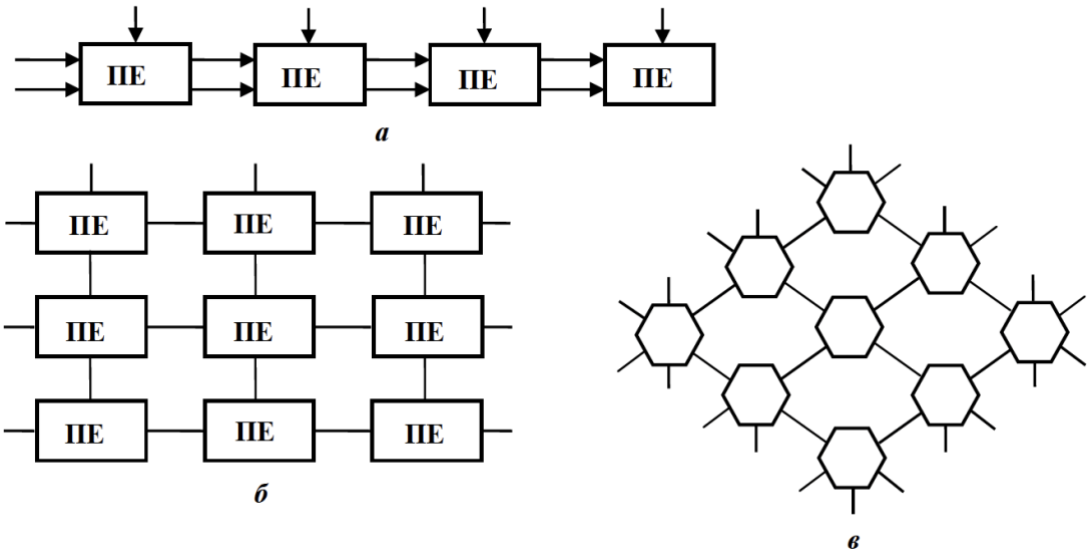

Рисунок 13.7 – Конфігурація систолічних матриць: а – лінійна; б – прямокутна; в – гексагональна

Кожна конфігурація матриці найбільш пристосована для виконання певних функцій, наприклад лінійна матриця оптимальна для реалізації фільтрів в реальному масштабі часу; гексагональна – для виконання операцій обернення матриць, а також для операцій над матрицями спеціального типу. Найбільш універсальними і розповсюдженими є матриці з лінійною структурою.

Для розв'язання складних задач конфігурація систолічної структури може являти собою набір окремих матриць, складну мережу взаємозв'язаних матриць або оброблювану поверхню. Під оброблюваною поверхнею розуміють безкінечну прямокутну мережу ПЕ, де кожний ПЕ є з'єднаним зі своїми чотирма сусідами (або більшим числом ПЕ).

13.3.4 Обчислювальні системи з командними словами надвеликої довжини (VLIW)

VLIW (Very Long Instruction Word) – це набір команд, які організовані аналогічно організації горизонтальної мікрокоманди в мікропрограмному пристрої управління.

Ідея VLIW базується на тому, що задача ефективного планування паралельного виконання декількох команд покладається на «розумний» компілятор. Такий компілятор спочатку досліджує початкову програму з метою виявити всі команди, які можуть бути виконані одночасно, причому так, щоб це не приводило до виникнення конфліктів. У процесі аналізу компілятор може навіть частково імітувати виконання програми, яка розглядається. На наступному етапі компілятор намагається об'єднати такі команди в пакети, кожний з котрих розглядається як одна наддовга команда. Об'єднання декількох простих команд в одну наддовгу виконується за такими правилам:

- кількість простих команд, які об'єднуються в одну команду надвеликої довжини, дорівнює числу функціональних (виконавчих) блоків (ФБ);
- у наддовгу команду входять тільки прості команди, які виконуються різними ФБ, тобто забезпечується одночасне виконання усіх складових наддовгої команди.

Довжина наддовгої команди звичайно складає від 256 до 1024 біт. Така метакоманда містить декілька полів (по числу простих команд, які її утворюють), кожне з яких описує операцію для конкретного функціонального блока. На рис. 13.8 показаний можливий формат наддовгої команди та взаємозв'язок між її полями і ФБ, які реалізують окремі операції.

Як видно з рисунка, кожне поле наддовгої команди відображається на свій функціональний блок, що дозволяє отримати максимальну віддачу від апаратури блока виконання команд.

VLIW-архітектуру можна розглядати як статичну суперскалярну архітектуру. Мається на увазі, що розпаралелювання коду здійснюється на етапі компіляції, а не динамічно в час виконання. Як уже було відмічено, у

виконуваної наддовгої команди виключена можливість конфліктів, а це дозволяє значно спростити апаратуру VLIW-процесора та збільшити швидкодію.

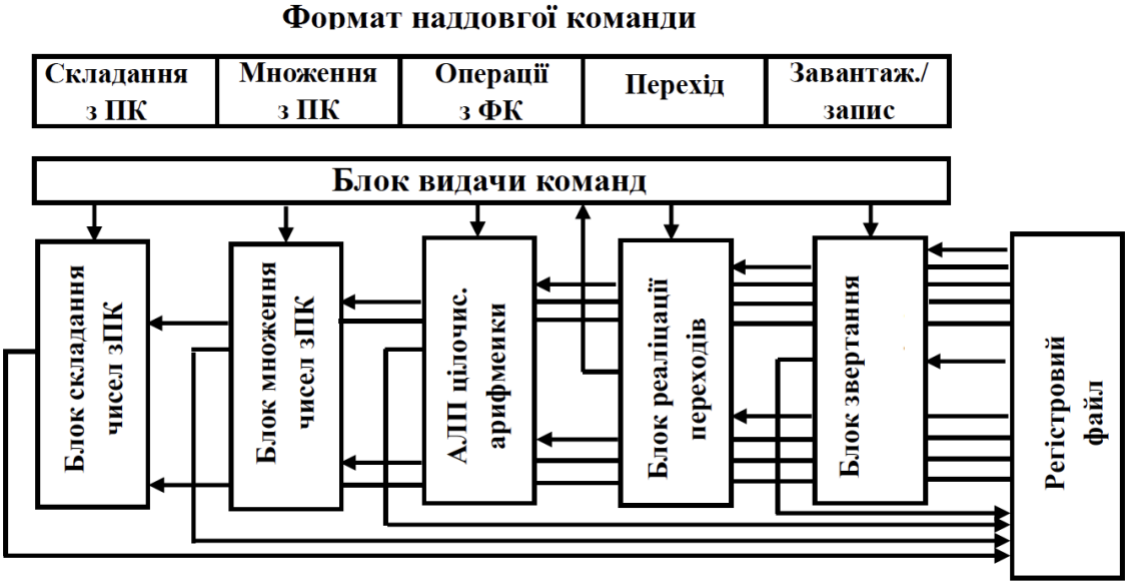

Рисунок 13.8 – Формат наддовгої команди та взаємозв'язок полів команди зі складовими блока виконання

Як видно з рисунка, кожне поле наддовгої команди відображається на свій функціональний блок, що дозволяє отримати максимальну віддачу від апаратури блока виконання команд.

VLIW-архітектуру можна розглядати як статичну суперскалярну архітектуру. Мається на увазі, що розпаралелювання коду здійснюється на етапі компіляції, а не динамічно в час виконання. Як уже було відмічено, у виконуваної наддовгої команди виключена можливість конфліктів, а це дозволяє значно спростити апаратуру VLIW-процесора та збільшити швидкодію.

Як прості команди, що створюють наддовгу, звичайно використовують команди RISC-типу, тому архітектуру VLIW іноді називають постRISC-архітектурою. Максимальне число полів у наддовгій команді дорівнює числу обчислювальних засобів і звичайно знаходиться в діапазоні від 3 до 20. Усі обчислювальні засоби мають доступ до даних, які зберігаються у єдиному багатопортовому регістровому файлі. Відсутність складних апаратних механізмів, які є характерними для суперскалярних процесорів (передбачення

переходів, позачергове виконання і ін.), дає значний виграш у швидкодії та можливість більш ефективно використовувати площину кристалу. Переважна більшість цифрових сигнальних процесорів та мультимедійних процесорів з продуктивністю більш 1 млрд операцій/с базується на VLIW-архітектурі. Серйозна проблема VLIW-архітектури – ускладнення регістрового файла та зв'язків цього файла з обчислювальними пристроями.

13.4 Комп'ютерні системи класу MIMD (МКМД)

MIMD-системи володіють великою гнучкістю, зокрема вони можуть робити і як високопродуктивні однокористувацкі обчислювальні системи, і як багатопрограмні ОС, які паралельно виконують множину задач. Крім того, архітектура MIMD дозволяє найбільш ефективно розпорядитися усіма перевагами сучасної мікропроцесорної технології.

Приблизні значення пікової продуктивності для різних типів систем класу MIMD показані на рис. 13.9.

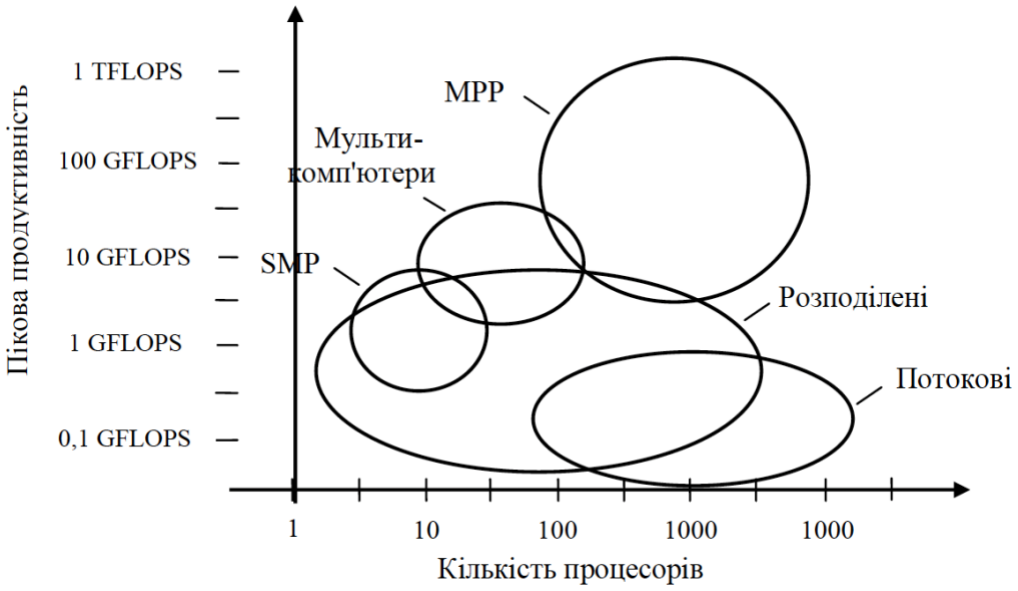

Рисунок 13.9 – Продуктивність MIMD-систем як функція їх типу та кількості процесорів

У MIMD-системі кожний процесорний елемент (ПЕ) виконує власну програму достатньо незалежно від інших ПЕ. У той же час ПЕ повинні якось

взаємодіяти один з одним. Відмінність у способі такої взаємодії визначає умовне ділення MIMD-систем на ОС з загальною пам'яттю та системи з розподіленою пам'яттю. В системах із загальною пам'яттю, які характеризують як сильно зв'язані, є загальна пам'ять даних і команд, до якої мають доступ усі ПЕ за допомогою загальної шини або мережі з'єднань.

До цього типу, зокрема, відносяться симетричні мультипроцесори (SMP, Symmetric Multiprocessor) та системи з неоднорідним доступом до пам'яті (NUMA, Non-Uniform Memory Access).

У системах з розподіленою пам'яттю або слабо зв'язаних багатопроцесорних системах уся пам'ять розподілена між процесорними елементами, і кожний блок пам'яті доступний тільки «власному» процесору. Мережа з'єднань зв'язує процесорні елементи один з одним. Представниками цієї групи є системи з масовим паралелізмом (MPP, Massively Parallel Processing) та кластерні обчислювальні системи.

Базовою моделлю обчислень на MIMD-системі є сукупність незалежних процесів, що епізодично звертаються до сумісно використовуваних даних.

Існує багато варіантів цієї моделі. На одному кінці спектра – розподілені обчислення, у рамках яких програма ділиться на достатньо велике число паралельних задач, що складаються з множини підпрограм. На другому кінці – модель потокових обчислень, де кожна операція в програмі може розглядатись як окремий процес. Така операція очікує надходження вхідних даних (операндів), які повинні бути переданими їй іншими процесами. Після цього операція виконується, і результуюче значення передається тим процесам, які його потребують.

13.4.2 Симетричні мультипроцесорні системи (SMP)

Поняття симетричні мультипроцесорні обчислювальні системи, так звані SMP-системи (Symmetric Multiprocessor), відноситься як до архітектури обчислювальної системи, так і до поведінки операційної системи, яка відображає дану архітектурну організацію. SMP можна визначити як обчислювальну систему, що має такі характеристики:

- є два або більше процесорів порівнянної продуктивності;
- процесори сумісно використовують основну пам'ять і функціонують в єдиному віртуальному і фізичному адресному просторі;
- усі процесори зв'язані між собою за допомогою шини або за іншою схемою так, що час доступу до пам'яті будь-якого з них є однаковий;
- усі процесори розділяють доступ до пристроїв вводу/виводу або через одні й ті ж канали, або через різні канали, які забезпечують доступ до одного й того ж зовнішнього пристрою;
- усі процесори здатні виконувати однакові функції (цим пояснюється термін «симетричні»);
- будь-який з процесорів може обслуговувати зовнішні переривання;
- обчислювальна система управляється інтегрованою операційною системою, яка організовує і координує взаємодію між процесорами та програмами на рівні завдань, задач, файлів і елементів даних.

В порівнянні з однопроцесорними схемами SMP-системи мають переваги по таких показниках:

Продуктивність. Якщо задача, яка повинна бути розв'язана, підлягає розбиттю на декілька частин так, що окремі частини можуть виконуватись паралельно, то множина процесорів дає виграш у продуктивності відносно одиничного процесора того ж типу.

Готовність. У симетричному мультипроцесорі відмова одного з компонентів не приводить до відмови системи, тому що будь-який з процесорів здатний виконувати ті ж функції, що й інші.

Розширюваність. Продуктивність системи може бути збільшена додаванням додаткових процесорів.

Масштабованість. Варіюючи число процесорів у системі, можна створювати системи різної продуктивності і вартості.

На рис. 13.10 у загальному вигляді показана архітектура симетричної мультипроцесорної обчислювальної системи.

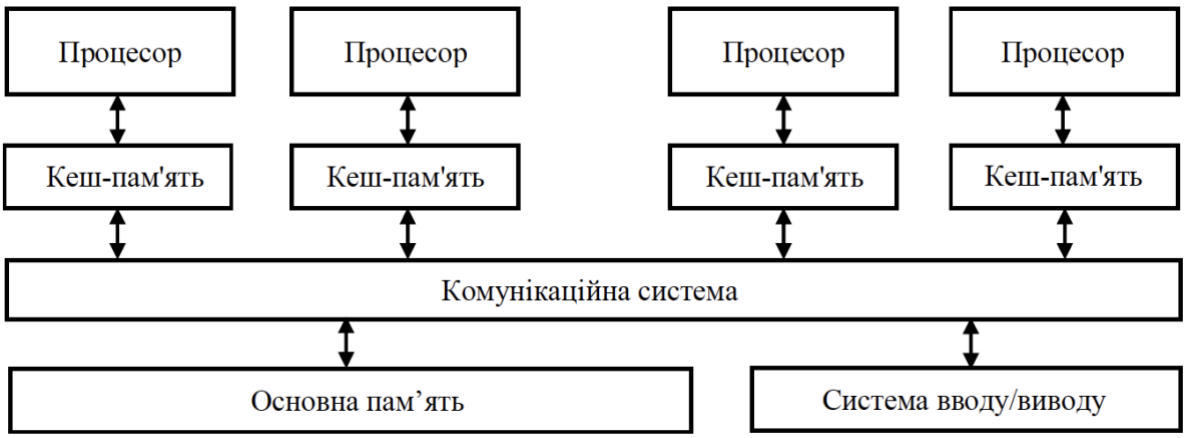

Рисунок 13.10 – Організація симетричної мультипроцесорної системи

Типова SMP-система містить від двох до 32 ідентичних процесорів, у ролі яких звичайно використовують дешеві RISC-процесори. В останній час намітилася тенденція оснащення SMP-систем також і CISC-процесорами, зокрема Pentium.

Кожний процесор забезпечений локальною кеш-пам'яттю, яка складається з кеш-пам’яті першого(L1) та другого (L2) рівнів. Узгодженість вмісту кеш-пам'яті всіх процесорів забезпечується апаратними засобами. В деяких SMP-системах проблема когерентності знімається за рахунок загальної кеш-пам'яті. Застосування загальної кеш-пам'яті супроводжується збільшенням вартості і зменшенням швидкодії кеш-пам'яті.

Усі процесори обчислювальної системи мають рівноправний доступ до основної пам'яті і пристроїв вводу/виводу, що розділяються. Така можливість забезпечується комунікаційною системою. Звичайно процесори взаємодіють між собою через основну пам'ять. В деяких SMP-системах передбачається також прямий обмін сигналами між процесорами.

Пам'ять системи будується по модульному принципу і організована так, що дозволяється одночасне звернення до різних її модулів (банків). В деяких конфігураціях в доповнення до ресурсів, які використовуються сумісно, кожний процесор володіє також власною локальною основною пам'яттю та каналами вводу/виводу.

13.4.3 Системи з масовою паралельною обробкою (MPP)

Основною ознакою, за якою обчислювальну систему відносять до архітектури з масовою паралельною обробкою (MPP – Massively Parallel Processing), служить кількість процесорів n. Строгої межі не існує, але при n 128 вважається, що це вже MPP, а при n 32 – ще ні.

Головні особливості, за якими обчислювальну систему відносять до класу MPP, можна сформулювати таким чином:

- стандартні мікропроцесори;
- фізично розподілена пам'ять;
- мережа з'єднань з високою пропускною здатністю і малими затримками;
- добра масштабованість (можливість варіювання числом процесорів (до тисяч процесорів));
- асинхронна MIMD-система з пересилкою повідомлень;
- програма являє собою множину процесів, які мають окремі адресні простори.

Узагальнена структура MPP-системи показана на рис. 13.11.

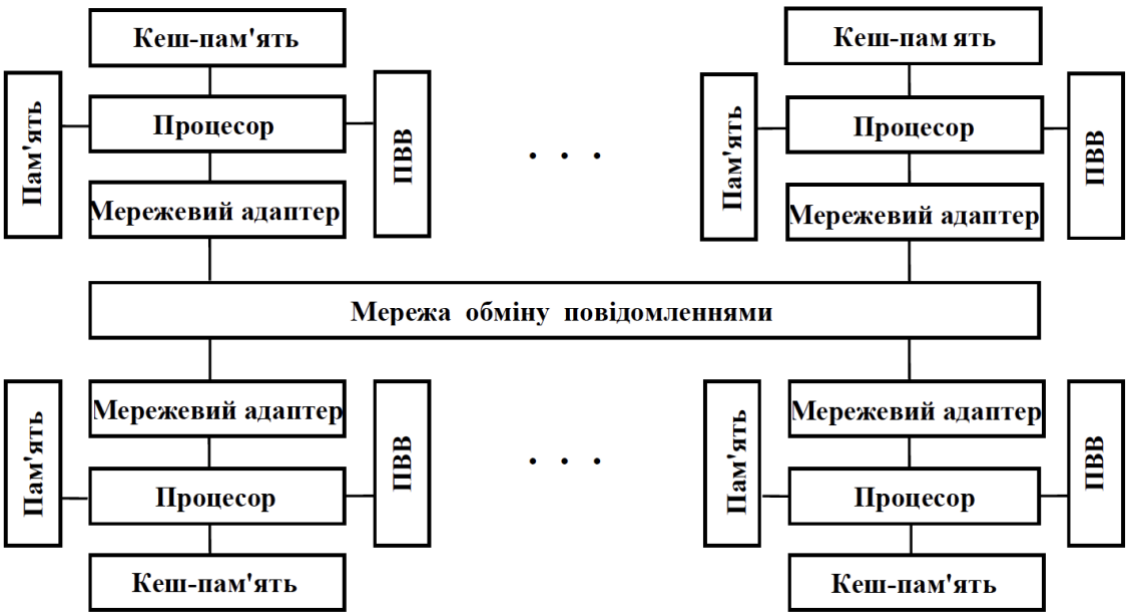

Рисунок 13.11 – Структура обчислювальної системи з масовою паралельною обробкою

Основні причини появи систем з масовою паралельною обробкою – це, по-перше, необхідність побудови ОС з дуже великою продуктивністю і, по-друге, зменшення її вартості.

Характерна риса MPP-систем – наявність єдиного управляючого пристрою (процесора), що розподіляє завдання між множиною підлеглих пристроїв. Схема взаємодії в загальних рисах достатньо проста:

- центральний управляючий пристрій формує чергу завдань, кожному з яких призначається деякий рівень пріоритету;
- по мірі звільнення підлеглих пристроїв їм передаються завдання з черги;
- підлеглі пристрої оповіщають центральний процесор про хід виконання завдань, зокрема про завершення виконання або про потребу в додаткових ресурсах;
- у центрального пристрою є засоби для контролю роботи підлеглих процесорів, зокрема для виявлення нештатних ситуацій, переривання виконання завдань у випадку появи більш пріоритетної задачі та ін.

У деякому наближенні є сенс вважати, що на центральному процесорі виконується ядро операційної системи (планувальник завдань), а на підлеглих – додатки. Підлеглість між процесорами може бути реалізована як на апаратному, так і на програмному рівні.

Завдяки властивості масштабованості, MPP-системи на сьогодні є лідерами по досягнутій продуктивності. З іншого боку, розпаралелювання в MPP-системах є складною задачею. Ефективність розпаралелювання у багатьох випадках сильно залежить від деталей архітектури MPP-системи. Для синхронізації паралельно виконуваних процесів необхідний обмін повідомленнями, які повинні доходити з будь-якого вузла системи у будь-який інший вузол. Час передачі інформації від вузла до вузла залежить від стартової затримки та швидкості передачі. Продуктивність процесорів набагато більше пропускної здатності каналів зв'язку, тому інфраструктура каналів зв'язку в MPP-системах є найбільш проблемною.

Слабким місцем МРР було і є центральний управляючий пристрій (ЦУП) – при виході його зі строю вся система стає непрацеспроможною. Підвищення надійності ЦУП здійснюється за рахунок спрощення або дублювання його апаратури.

Сфера застосування ОС з масовим паралелізмом постійно розширюється. Різні системи цього класу експлуатуються у багатьох провідних суперкомп'ютерних центрах світу.

13.4.4 Кластерні обчислювальні системи

Один із самих найсучасніших напрямів у області створення обчислювальних систем – це кластеризація. За продуктивністю та коефіцієнтом готовності кластеризація являє собою альтернативу симетричним мультипроцесорним системам. Поняття кластер визначається як група взаємоз'єднаних обчислювальних систем (вузлів), що сумісно функціонують, складаючи єдиний обчислювальний ресурс і створюючи ілюзію наявності єдиної обчислювальної машини. Як вузол кластера може бути однопроцесорна ЕОМ і ОС типу SMP або MPP. Важливим є тільки те, що кожний вузол є здатним функціонувати самостійно і окремо від кластера. В плані архітектури сутність кластерних обчислень зводиться до об'єднання декількох вузлів високошвидкісною мережею. Для опису такого підходу, крім терміну «кластерні обчислення», достатньо часто використовують такі назви: кластер робочих станцій, гіперобчислення, паралельні обчислення на базі мережі, ультраобчислення.

Як вузли кластерів можуть використовуватись однакові ОС (гомогенні кластери), а також різні (гетерогенні кластери). За своєю архітектурою кластерна ОС є слабкозв'язаною системою.

На рівні апаратного забезпечення кластер – це просто сукупність незалежних обчислювальних систем, які об'єднані мережею. У разі з'єднання ЕОМ у кластер майже завжди підтримуються прямі міжмашинні зв'язки. Рішення можуть бути простими, що ґрунтуються на апаратурі Ethernet, або складними з

високошвидкісними мережами з пропускною здатністю у сотні мегабайтів у секунду.

Вузли кластера контролюють працездатність один одного та ведуть обмін специфічною інформацією, що є характерною для кластера. Контроль працездатності здійснюється за допомогою спеціального сигналу, який часто називають heart-beat, що значить «серцебиття». Цей сигнал передається вузлами кластера один одному, щоб підтвердити їх нормальне функціонування.

Невід'ємна частина кластера – спеціалізоване програмне забезпечення (ПЗ), на яке покладається задача забезпечення безперебійної роботи у разі відмови одного або декількох вузлів. Таке ПЗ виконує перерозподіл обчислювального навантаження у разі відмови одного або декількох вузлів кластера, а також відновлення обчислень у випадку виникнення збою у вузлі. Крім того, якщо в кластері сумісно використовуються диски, кластерне ПЗ підтримує єдину файлову систему.

Можливість практично необмеженого нарощування числа вузлів та відсутність єдиної операційної системи робить кластерні архітектури успішно масштабованими, і навіть системи з сотнями і тисячами вузлів показують себе на практиці з позитивного боку.