Курс «Паралельні обчислення та засоби їх реалізації».

Лекція 1. Вступ до паралельних обчислень.

План лекції.

1. Поняття паралельних обчислень, передумови їх виникнення та розвитку.
2. Шляхи підвищення швидкодії ЕОМ.
3. Області застосування та задачі паралельних обчислень.
4. Підходи до організації паралельних обчислень (розпаралелювання).
5. Рівні розпаралелювання.

1. ***Паралельні обчислення*** – це форма обчислень, у яких кілька дій виконуються одночасно. Паралельні обчислення ***ґрунтуються*** на тому факті, що деякий заданий фрагмент обчислень можна розділити на декілька менших фрагментів і виконувати їх незалежно.

Для розв'язання багатьох задач традиційно використовували (і зараз ще використовують) *послідовні обчислення*. У даному випадку для розв'язання деякої задачі розробляють чисельний алгоритм, який програмно реалізується у вигляді послідовності команд. Ці команди виконуються процесором комп'ютера. При цьому у кожен момент часу може виконуватись лише одна команда. Після завершення її виконання починається виконання наступної команди.

Що ж стосується *паралельних обчислень*, то вони передбачають одночасне використання декількох обчислювальних пристроїв (процесорів) для розв'язання однієї задачі.

Розглянемо декілька прикладів *використання паралелізму*.

Приклад 1. Якщо деякий обчислювальний пристрій виконує одну операцію за одиницю часу, то тисячу таких же операцій він виконає за тисячу одиниць часу. Якщо вважати, що у нас є десять таких незалежних пристроїв, здатних працювати одночасно, то ту ж тисячу операцій система із десяти пристроїв може виконати уже не за тисячу, а за 100 одиниць часу. Аналогічно, система із N обчислювальних пристроїв ту ж роботу виконає за $1000/N$ одиниць часу. Подібні аналоги можна знайти і в житті.

Приклад 2. Якщо один робітник перекопає ділянку землі за 10 годин, то група робітників із п'ятидесяти чоловік з такими ж фізичними даними, працюючи одночасно, виконає цю роботу за 12 хвилин. Отже, принцип паралельності в дії!

Одним із перших, хто здійснив паралельні обчислення, був академік Самарський Олександр Андрійович, який на початку 50-х років виконав розрахунки, необхідні для моделювання ядерних вибухів. Він розв'язав цю задачу, посадивши декілька десятків молодих дівчат з ***арифмометрами*** за столи. Дівчата передавали дані одна одній просто на словах і відкладали необхідні цифри на арифмометрах. Так була розрахована еволюція вибухової хвилі. Це, можна сказати, і була перша паралельна обчислювальна система. Хоча розрахунки були майстерно виконані,

але точність їх була низькою, бо вузлів у використовуваній сітці було мало, а час розрахунку виходив досить великим.

Головною ***передумовою*** виникнення паралельних обчислень була необхідність збільшення швидкості послідовних обчислень. Сама ідея паралельного оброблення даних як потужного резерву збільшення продуктивності обчислювальних пристроїв була висловлена Чарльзом Беббіджем приблизно за 100 років до появи першої ЕОМ. Однак, тодішній рівень розвитку технологій (а це була середина XIX ст.) не дозволив йому реалізувати її. Перші ЕОМ появилися більше як півстоліття тому. Одна із перших обчислювальних машин Baby (1948 р.) була розроблена в Манчестерському університеті (Англія). Перша з обчислювальних машин у колишньому Радянському Союзі (і в континентальній Європі) МЭСМ (Малая электронно-счётная машина) була розроблена в Києві під керівництвом академіка Лєбєдєва Сергія Олексійовича протягом 1948–1950 рр. У 1969 р. під керівництвом академіка Глушкова Віктора Михайловича в Києві було створено прообраз ПК Мир-2 (Машина инженерных расчётов).

Із виникненням ЕОМ завжди залишалася актуальною проблема підвищення їх швидкодії, оскільки зростали обсяги оброблюваних даних, а обчислювальна техніка впроваджувалась у різні сфери людської діяльності.

2. Основними ***шляхами підвищення швидкодії ЕОМ*** є:

- вдосконалення елементної бази (збільшення тактової частоти роботи електронних схем, мініатюризація елементів цих схем);
- введення паралелізму в обчислювальний процес, тобто використання паралельних обчислень.

Окрім цього слід відзначити ще два такі напрямки удосконалення роботи з ЕОМ, як

- зменшення семантичного (смислового, змістового) розриву між прикладним і системним програмним забезпеченням, з одного боку, та архітектурою ЕОМ, з іншого боку;
- зменшення розриву між ЕОМ та користувачем (інтелектуалізація інтерфейсу).

Із появою ЕОМ досить помітним був саме розвиток їх елементної бази (електронні схеми з використанням: ламп $\rightarrow$ транзисторів, резисторів $\rightarrow$ ІС $\rightarrow$ ВІС $\rightarrow$ НВІС). Зараз цей шлях підвищення продуктивності обчислювальних засобів практично себе вичерпав, оскільки існують фізичні обмеження на збільшення тактової частоти та розміри елементів, з яких складається НВІС.

Дослідження, пов’язані із впровадженням паралелізму в обчислювальний процес, також інтенсивно проводились після появи перших ЕОМ. З’явилися і інші ***передумови*** для впровадження та розвитку паралельних обчислень. Це, зокрема, і сам факт появи паралельних обчислювальних систем. Так, у 70-ті роки минулого століття з’являються перші векторноконвеєрні та паралельні обчислювальні системи (IBM-370/195; STAR-100; CDC-7600; CYBER 203). Пік розробок паралельних обчислювальних засобів у колишньому Радянському Союзі припав приблизно на 1986 рік, коли в повному обсязі фінансувалася низка проектів (макроконвеєр ЕС 2701, м.Київ; комплекси «Эльбрус-1», «Эльбрус-2», м.Москва; інтелектуаль-

ний термінал для «Эльбрус-2», м.Київ; системи «Марс-М», «Сибирь», м.Новосибірськ тощо).

Зараз основним напрямком підвищення швидкодії обчислювальних систем є ***тотальне впровадження паралелізму*** в обчислювальний процес. Для широкого кола користувачів стали доступними багатоядерні комп'ютери, для наукових досліджень залучаються кластерні системи та обчислювальні мережі (GRID, розподілені обчислювальні середовища, утворені із доступних розподілених неоднорідних обчислювальних засобів). Слід зауважити, що є певні досягнення і в ***удосконаленні елементної бази***, але вони поки-що не досягли значного поширення на практиці у тому числі і через високу вартість. Це, зокрема, використання ***тривимірних інтегральних схем***, для виготовлення яких повністю підходить весь технологічний процес виготовлення двовимірних інтегральних схем; а також застосування ***оптоелектронних елементів***, унаслідок чого швидкодія обчислень зросте приблизно в десятки разів порівняно з традиційною елементною базою (транзисторно-резисторною).

3. Гостра ***необхідність застосування*** паралельних обчислень виникла у середині минулого століття у зв'язку із розвитком атомної енергетики, літакобудування, ракетно-космічних технологій, дослідженням навколишнього середовища (повітряного та водного басейнів).

Приклад 3. Щоб оцінити складність розв'язуваних на практиці задач, візьмемо конкретну предметну область, наприклад, оптимізацію процесу добування нафти (із посібника *Воеводин Вл.В. Курс лекций «Параллельная обработка данных». – Режим доступу: http://www.parallel.ru/vvv/lec1.html*). Маємо підземний резервуар з деякою кількістю свердловин: по одних на поверхню відкачують нафту, а по інших у зворотньому напрямку закачується вода. Необхідно змоделювати ситуацію у заданому резервуарі, щоб оцінити запаси нафти або зрозуміти потребу у додаткових свердловинах.

Приймемо спрощену схему, за якою модельована область відображається в куб, однак її буде досить для оцінювання кількості необхідних арифметичних операцій. Достатні розміри куба, за яких можна одержувати правдоподібні результати – це $100 \times 100 \times 100$ точок. У кожній точці куба необхідно обчислити від 5 до 20 функцій: три компоненти швидкості, тиск, температуру, концентрацію деяких інших компонент (вода, газ і нафта – це мінімальний набір компонент, а у більш реалістичних моделях розглядають, наприклад, різні фракції нафти). Далі, значення функцій знаходяться як розв'язок нелінійних рівнянь, що потребує від 200 до 1000 арифметичних операцій. І, насамкінець, якщо досліджується нестаціонарний процес, тобто потрібно зрозуміти, як ця система буде вести себе у часі, то здійснюється 100 – 1000 кроків за часом. Отже, ми одержали:

$$10^6 \text{ (точки сітки)} \times 10 \text{ (функції)} \times 500 \text{ (операції)} \times 500 \text{ (кроки за часом)} = \\ = 2.5 \times 10^{12}.$$

Тобто 2500 мільярдів арифметичних операцій для виконання лише одного розрахунку! А якщо сюди додати зміну параметрів моделі та відслідковування поточної

ситуації при зміні вхідних даних? Подібні розрахунки необхідно робити багато разів, що накладає дуже жорсткі вимоги на продуктивність використовуваної обчислювальної системи. Тому, у даному разі ПК не достатньо.

Приклади ***використання паралельних обчислень*** можна знайти не лише в нафтовидобувній промисловості, а й у інших областях людської діяльності, зокрема, в таких, як

- машинобудування;
- фармакологія;
- прогноз погоди та моделювання змін клімату;
- сейсморозвідка;
- проектування електронних пристроїв;
- синтез нових матеріалів;
- моделювання структури Всесвіту;
- прогноз промислового та економічного розвитку регіонів;
- моделювання процесів інтенсивних фізико-хімічних і ядерних реакцій;
- структурна біологія;
- генетика людини;
- квантова хромодинаміка.

Загалом ***задачі***, що потребують ***паралельних обчислень***, можна поділити на два типи:

- складні задачі, що вимагають виконання великих обсягів обчислень, наприклад:
    - задачі механіки взаємозв'язаних полів різної вимірності;
    - задачі гідро- та газодинаміки;
    - задачі, для розв'язання яких використовуються методи скінченних елементів або різниць з десятками, сотнями і більше тисяч вузлів;
- задачі, розв'язок яких необхідно одержати в режимі реального часу (час, протягом якого розв'язання задачі має сенс), зокрема, це
    - задачі цифрової фільтрації сигналів та зображень;
    - задачі розпізнавання образів;
    - задачі одночасного спостереження за багатьма рухомими об'єктами;
    - транспортні задачі;
    - розпізнавання та синтез мови.

4. Відома значна кількість послідовних (традиційних) методів і алгоритмів розв'язання практичних задач науки і техніки. «Пристосування» цих методів і алгоритмів для їх ефективної реалізації на паралельних обчислювальних системах називається ***розпаралелюванням*** (paralleling).

Однак, часто доводиться спеціально розробляти і обгрунтовувати паралельні алгоритми. Вирішення цих проблем є предметом нової галузі обчислювальної математики – ***паралельної обчислювальної математики***. Результатами остан-

ньої є як ефективні паралельні алгоритми для розв'язання конкретних задач, так і цілі класи принципово нових алгоритмів, яскравими представниками яких є нейромережні, систолічні та квазісистолічні алгоритми. Одним із важливих результатів паралельної обчислювальної математики є встановлення факту, що багато із традиційних послідовних алгоритмів не мають ефективних паралельних аналогів.

Для послідовних обчислень кращим обчислювальним алгоритмом є той, який потребує мінімальну кількість арифметичних операцій (за інших рівних умов). Однак, для паралельних обчислень важливим є і врахування складності міжоператорних зв'язків за даними та керуванням. Тому актуальною проблемою є розроблення паралельних алгоритмів, які можуть бути легко відображені на архітектури відомих паралельних обчислювальних систем.

Отже, більш строго ***означимо розпаралелювання*** як знаходження паралельного алгоритму розв'язання задачі та реалізацію цього алгоритму на відповідній паралельній обчислювальній системі.

Відзначимо, що розпаралелювання не обов'язково передбачає одержання паралельного алгоритму із послідовного. Ми будемо розуміти термін «розпаралелювання» як синонім до терміну «синтез паралельного алгоритму».

Зауважимо, що розпаралелювання розв'язання будь-якої задачі є багатоваріантним: для однієї і тієї ж математичної моделі можна побудувати різні паралельні алгоритми і для кожного із них – різні паралельні програми. У зв'язку з цим однією із центральних проблем у галузі паралельних обчислень є проблема оцінки ефективності алгоритмів для даної паралельної обчислювальної системи або алгоритмів розв'язання заданої задачі на паралельних системах різної архітектури.

Із викладеного випливає, що фактично є ***два підходи до організації паралельних обчислень***:

- розпаралелювання відомих послідовних методів і алгоритмів розв'язання деякої задачі;
- розроблення принципово нового паралельного методу та алгоритму розв'язання задачі.

5. ***Розпаралелювання*** обчислень можна здійснювати ***на різних рівнях***. Хоча не існує чіткої класифікації цих рівнів, ми зупинемося на деяких із них, щоб розширити своє уявлення про сутність розпаралелювання.

Розпаралелювання на рівні задач. Досить часто розпаралелювання на цьому рівні є найпростішим та найефективнішим. Таке розпаралелювання є можливим у тих випадках, коли розв'язувана задача природнім шляхом складається із незалежних підзадач, кожну з яких можна розв'язати окремо. Розпаралелювання на рівні задач нам демонструє операційна система, запускаючи на багатоядерному ПК програми на різних ядрах. Одночасне розв'язання однієї і тієї ж задачі за різних вхідних даних – це теж приклад розпаралелювання на рівні задач.

Розпаралелювання обчислень на рівні процедур. Цей рівень передбачає одночасне (паралельне) виконання різних розділів однієї і тієї ж програми або алгоритму розв'язання конкретної задачі. Згадані розділи називаються процесами і відповідають приблизно послідовним процедурам. Процеси можуть протікати

автономно або взаємодіяти між собою. Очевидно, що для збільшення швидкості виконуваних обчислень кількість таких взаємодій має бути мінімальною. Прикладами розпаралелювання на процедурному рівні є алгоритми паралельного сортування, множення матриць, розв'язання систем лінійних алгебраїчних рівнянь.

Розпаралелювання на рівні виразів. На цьому рівні відбувається одночасне виконання окремих частин (компонент) даного виразу. Прикладами у даному випадку є реалізація виразів

$A = a_1 + a_2 + a_3 + ... + a_{16}$; $B = b_1 \times b_2 \times b_3 \times ... \times b_{32}$; $C = \max\{c_1, c_2, c_3, ..., c_{64}\}$

в режимі повного двійкового дерева.

Розпаралелювання на рівні команд. Програма є потоком команд, які виконує процесор. Інколи можна змінити порядок виконання цих команд, розподіливши їх за групами, які можуть виконуватись паралельно, без зміни результату роботи усієї програми. Це і є розпаралелюванням на рівні команд. Такий підхід до збільшення продуктивності обчислень переважав з середини 80-х до середини 90-х років минулого століття. Зараз програміст рідко заглядає на цей рівень. Роботу із розташування команд у найбільш зручній послідовності для процесора виконує компілятор.

Розпаралелювання обчислень на бітовому рівні. На цьому рівні відбувається паралельне виконання бітових операцій у межах одного слова. Паралельність на рівні бітів реалізована в будь-якому працюючому мікропроцесорі.

Зі створенням у 70-х роках минулого століття технології виготовлення НВІС збільшення швидкості обчислень на комп'ютерах відбувалось з допомогою подвоєння розміру машинного слова – кількості інформації, яку комп'ютер може обробляти за один такт. Збільшення розміру слова зменшує кількість команд, необхідних, щоб виконати операцію над даними, розмір яких є більший від розміру машинного слова. Наприклад, коли восьмибітний процесор має додати два шістнадцятирозрядні числа, він має спочатку додати 8 біт нижчого розряду з кожного числа, використовуючи стандартну команду додавання, а потім додати 8 біт вищого розряду. Тому восьмибітний процесор потребує дві команди для виконання однієї операції у даному випадку. А шістнадцятибітний – лише одну. Зауважимо, що у 2003–2004 рр. з'явились уже 64-х розрядні процесори.

Список рекомендованих для курсу джерел.

1. *Вальковский В.А.* Распараллеливание алгоритмов и программ. Структурный подход. – М.: Радио и связь, 1989. – 176 с.

2. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – СПб: БХВ-Петербург, 2002. – 608 с.

3. *Дорошенко А.Ю.* Лекції з паралельних обчислювальних систем. Методичний посібник. – Київ: Видавничий дім «КМ Академія», 2003. – 42 с.

4. *Жуков І.А., Курочкін О.В.* Паралельні та розподілені обчислення. Навч. посібник. – К.: «Корнійчук», 2005. – 226 с.

5. *Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И.* Реконфигурируемые мультиконвейерные вычислительные структуры. – Ростов н/Д: Изд-во ЮНЦ РАН, 2008. – 320 с.

6. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991. – 367 с.

7. *Рейтинговий* список найпотужніших обчислювальних систем світу. – Режим доступу: www.top500.org.

8. *Ремонтов А.П., Писарев А.П.* Вычислительные машины и системы. Учебное пособие. – Пенза: Пензенский государственный университет, 2006. – 96 с.

9. *Сайт* з паралельних обчислень. – Режим доступу: http://www.parallel.ru.

10. *Семеренко В.П.* Технології паралельних обчислень: навчальний посібник. – Вінниця: ВНТУ, 2018. – 104 с.

11. *Штейнберг Б.Я.* Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. – Ростов н/Д: Изд-во Рост. ун-та, 2004. – 192 с.

12. *Штейнберг Б.Я., Штейнберг О.Б.* Преобразование программ – фундаментальная основа создания оптимизирующих распараллеливающих компиляторов // Программные системы: теория и приложения.– 2021.– **12**, № 1. – С. 21–113.

Курс «Паралельні обчислення та засоби їх реалізації».

Лекція 2. Паралельні обчислювальні системи та їх класифікація (І ч.).

План лекції.

1. Засоби реалізації паралельних обчислень.
2. Конвеєризація та паралелізм.
3. Векторизація обчислень.
4. Поняття паралельної обчислювальної системи та показники її продуктивності.
5. Поняття архітектури паралельних обчислювальних систем та головні її параметри.
6. Головні напрямки розвитку паралельних систем.

1. Для реалізації паралельних обчислень використовують **апаратні та програмні засоби**. До апаратних засобів можна віднести паралельні обчислювальні системи **універсального та спеціального призначення**. Перші використовують для розв'язання багатьох класів практично важливих задач науки і техніки, а системи спеціального призначення – для чисельного розв'язання вузького (одного) класу задач або навіть однієї задачі. Завдяки врахуванню специфіки розв'язуваної задачі (кількість гілок, які можна виконати одночасно, структура обмінів між ними, режим подання вхідних даних тощо) обчислювальні системи спеціального призначення дозволяють досягти високої продуктивності виконуваних обчислень.

Найбільш поширеними паралельними обчислювальними системами універсального призначення зараз є **багатоядерні** комп'ютери та **кластерні** системи. Прикладами систем спеціального призначення є **систолічні** обчислювальні структури, **квазісистолічні** структури, **нейронні** мережі різних типів, обчислювальні засоби на базі однорідних обчислювальних середовищ, клітинних автоматів тощо.

Програмні засоби реалізації паралельних обчислень можна умовно також поділити на засоби **загального призначення та спеціальні** програмні засоби.

Засоби **загального** призначення:

- операційні системи (Windows, Unix (AIX (Advanced Interactive eXecutive), IRIX, Solaris, Linux), Mac OS тощо);
- стандартні бібліотеки на різних мовах програмування для розв'язання низки задач;
- мови програмування (C, C++, C#, Fortran GNS, Sisal, Ada, Lisp, Java, Python, Eiffel, Ruby, тощо);
- компілятори (ранні розпаралелюючі: Paradigm compiler, Polaris compiler, SUIF (Stanford University Intermediate Format) compiler (для програм на Фортрані); сучасні: Sun studio compiler, Intel C++ Compiler, gcc (GNU Compiler Collection)).

Спеціальні програмні засоби:

- інтерфейси програмування для паралельних обчислювальних систем: Open MP, MPI, PVM;

Open MP (Open Multi-Processing) – для систем із спільною пам’яттю;

MPI (Message Passing Interface) – для систем із розподіленою пам’яттю, які складаються із однакових вузлів;

PVM (Parallel Virtual Machine) – для гетерогенних обчислювальних систем із розподіленою пам’яттю, тобто систем, вузли яких відрізняються архітектурою процесора і операційною системою;

- засоби об’єднання обчислювальних ресурсів: Globus Toolkit (набір інструментів з відкритим вихідним кодом для грід-обчислень), gLite (комп’ютерне програмне забезпечення для грід-обчислень).

2. Ідея **конвеєризації обчислень** полягає у виділенні окремих **етапів** виконання деякої операції так, щоб кожен етап, виконавши свою роботу, передавав би результат наступному етапу, одночасно приймаючи нову порцію вхідних даних. Унаслідок конвеєризації одержуємо очевидний виграш у швидкості обчислень завдяки суміщенню спочатку рознесених у часі операцій.

Приклад 1. Нехай у деякій операції можна виділити п’ять мікрооперацій, кожна з яких виконується за одну одиницю часу. Якщо ми маємо один неподільний послідовний обчислювальний пристрій, то 100 таких операцій він виконає за 500 одиниць часу. Якщо ж кожну мікрооперацію виділити в окремий етап конвеєрного пристрою, то на п’ятій одиниці часу на різній стадії оброблення такого пристрою будуть знаходитися перші п’ять операцій, а весь набір із ста операцій буде оброблено за 5+99=104 одиниці часу. Прискорення порівняно з виконанням на послідовному пристрої буде приблизно в п’ять раз (за кількістю етапів конвеєра).

Здавалось би, що конвеєрну обробку можна з успіхом замінити звичайним розпаралелюванням, для чого потрібно продублювати основний неподільний пристрій стільки разів, скільки є етапів у конвеєрі. Дійсно, п’ять таких пристроїв із попереднього прикладу виконають 100 операцій за 100 одиниць часу, що є швидше, ніж з використанням конвеєрного пристрою. У чому ж проблема? Відповідь є доволі простою: збільшуючи у п’ять разів кількість пристроїв, ми значно збільшуємо як обсяг апаратури, так і її вартість.

Приклад 2. Уявіть собі, що на автозаводі вирішили забрати конвеєр, зберігши при цьому темпи випуску автомобілів. Якщо раніше на конвеєрі одночасно знаходилось 1000 автомобілів, то, діючи за аналогією з попереднім прикладом, необхідно набрати 1000 бригад, кожна з яких:

1) здатна повністю скласти автомобіль від початку і до кінця, виконавши сотні різного типу операцій;

2) повинна зробити це за той самий час, протягом якого машина знаходилась раніше на конвеєрі.

Отже, не важко уявити собі досить високу собівартість такого автомобіля.

3. Під **векторизацією** фрагменту обчислень прийнято розуміти перетворення його до вигляду, який дозволяє реалізацію довгими серіями однотипних операцій. Програма, що здійснює таке перетворення, називається **векторизатором**. Векторизатор, вбудований у систему програмування векторної ЕОМ в якості блоку оптимізації, може значно прискорити виконання обчислень. Отже, векторизація обчислень здійснюється для ефективної їх реалізації на векторних та векторноконвеєрних системах. У таких системах є апаратно реалізовані команди для роботи з векторами. При цьому однією командою обробляються масиви незалежних даних.

У **векторних** системах кількість пристроїв у групі, яка складає векторний арифметико-логічний пристрій, визначають максимальну розмірність векторів, над якими векторний процесор може виконувати дії. Такі системи не набули широкого поширення із-за високої вартості організації векторного арифметико-логічного пристрою. Більш прийнятними за вартістю виявились **векторноконвеєрні** обчислювальні системи, для яких характерною є наявність конвеєрних функціональних пристроїв і набору векторних команд. На відміну від традиційного підходу, векторні команди (як уже було зазначено) оперують масивами незалежних даних, що дозволяє ефективно завантажувати доступні конвеєри, тобто команда вигляду $A=B+C$ означає додавання двох масивів, а не двох чисел.

4. Паралельна обчислювальна система – це мультипроцесорна система паралельної дії, що забезпечує одночасне (паралельне) виконання операцій однією або декількома програмами.

Як уже було зазначено на попередній лекції, основною метою створення паралельних обчислювальних систем є одержання високих показників їх продуктивності. Такими **показниками продуктивності** є:

- **швидкодія**, що вимірюється, зазвичай, у кількості операцій за секунду (оп/сек) і визначає обчислювальну потужність паралельної системи; найуживанішою одиницею вимірювання швидкодії спочатку був

  1 Mflops (мегафлопс) = 1 мільйон операцій з плаваючою комою за секунду (1964), а потім

  1 Gflops (гігафлопс) = 1 мільярд оп/сек (1987); (10^9);
  1 Tflops (терафлопс) = 1 трильйон оп/сек (1997); (10^{12});
  1 Pflops (петафлопс) = 1 квадрильйон оп/сек (2008); (10^{15});
  1 Eflops (екзафлопс) = 1 квінтильйон оп/сек (2020); (10^{18});
  1 Zflops (зетафлопс) = 1 секстильйон оп/сек (–); (10^{21});
  1 Yflops (йотафлопс) = 1 септильйон оп/сек (–); (10^{24});
  ……. = 1 октильйон оп/сек (–); (10^{27});
- **пропускна здатність** – кількість інформації (транзакцій, запитів, команд), оброблюваної мультипроцесорною системою за одиницю часу; опосередковано може вимірюватись пропускною здатністю каналів зв’язку мультипроцесорної системи в **Мбайт/сек**;
- загальний **обсяг оперативної пам’яті** паралельної системи (Мбайт, Гбайт, Тбайт);

1 Мбайт (мегабайт) =1024 Кб;
1 Гбайт (гігабайт) = 1024 Мб;
1 Тбайт (терабайт) = 1024 Гб;
1 Пбайт (петабайт) = 1024 Тб;
1 Ебайт (ексабайт) = 1024 Пб;
1 Збайт (зеттабайт) = 1024 Еб;
1 Йбайт (йоттабайт) = 1024 Зб.

5. Методи паралельних обчислень виникали і удосконалювались разом із розвитком архітектур обчислювальних систем.

У **другому поколінні** ЕОМ разом із операційними системами та системами програмування виникли методи мультипрограмування, тобто виконання кількох програм на одному й тому ж обладнанні. Проте обчислювальні системи залишались практично однопроцесорними. Машини **третього покоління** сприяли подальшому розвитку мультипрограмування. В архітектурі ЕОМ з'явилися канали – спеціалізовані процесори, що працювали паралельно з центральними процесорами і виконували виключно операції вводу/виводу, а також виникло поняття мультимашинних систем. Проте це ще не були в повному розумінні мультипроцесорні системи. Отже, ідея паралелізму обчислень втілювалась поступово.

Під **архітектурою ЕОМ** будемо розуміти її принципи роботи на функціональному рівні, тобто абстрактне уявлення про ЕОМ з погляду програміста.

Структуру обчислювальної системи можна означити, як сукупність окремих її блоків (пристрій вводу, процесор, запам'ятовуючий пристрій, пристрій виводу тощо) із зазначенням основних зв'язків між ними.

Головними параметрами архітектури паралельних систем є:

- **кількість і якість процесорів** системи, системи з кількістю процесорів $p > 100$ прийнято називати **масово паралельними**; паралельні обчислювальні системи, що складаються з однакових процесорів, називаються **однорідними** (гомогенними), а ті, що складаються з різних процесорів, – **неоднорідними** (гетерогенними);
- **вид основної пам'яті** – характеризує доступ до основної пам'яті з боку процесорів; спільна пам'ять є рівнодоступною для всіх процесорів паралельної системи і тому є **глобальною** для всієї системи; розподілена пам'ять розбита на приватні ділянки (за кількістю процесорів), доступ до кожної з яких має лише один процесор, тому така пам'ять є **локальною**;
- **система зв'язку** між процесорами – характеризує **топологію** зв'язку процесорів паралельної системи; прикладами такої топології є: шинний зв'язок, статична топологія (лінійна, кільцева, зіркова, матрична, у вигляді тору, повнозв'язна, гіперкуб (2^n процесорів мають попарні зв'язки довжиною n)), динамічна топологія (із змінними зв'язками), комутатори;
- **спосіб керування**: **синхронний** (централізований), коли команди в усіх процесорах виконуються за єдиним сигналом від центрального блоку керування, і **асинхронний** (розподілений), коли виконання команд у процесорах не синхронізується.

6. Продуктивність однієї **із перших** обчислювальних систем EDSAC дорівнювала приблизно 100 оп/сек, а пікова продуктивність **сучасної** надпотужної системи Tianhe-2 (MilkyWay-2) оцінюється в 54902.4 Tflops. Таке зростання продуктивності стало можливим завдяки збільшенню частоти роботи електронних схем і максимальному розпаралелюванню оброблення даних.

Принципово важливими кроками в підвищенні продуктивності обчислювальних систем були: запровадження конвеєрної організації виконання команд, залучення в систему команд векторних операцій, що дозволяють однією командою обробляти масиви даних, а також розподіл обчислень між багатьма процесорами. Як уже зазначалось у попередній лекції, визначна роль у становленні ЕОМ належить академікам Лєбєдєву С.О., Глушкову В.М., а також Сеймуру Крею.

Перші векторноконвеєрні та **паралельні** обчислювальні системи появилися в 70-ті роки XX ст. Спочатку ефект від їх застосування був доволі обмеженим, оскільки самих систем було мало і вони були доступними для невеликої кількості спеціалістів. Приклади таких систем: ASC TI, IBM-370/195, STAR-100, CDC-7600, CYBER 203, CYBER 205. У перших чотирьох системах були явно виражені блоки конвеєрної обробки, а у двох останніх системах операнди векторних команд вибиралися безпосередньо із оперативної пам’яті і результат записувався також в оперативну пам’ять (це системи типу «пам’ять – пам’ять»).

З середини 70-х років минулого століття фірма Cray Research Inc. почала **випуск векторних систем** Cray, які були прикладом процесорів типу «регістр-регістр». У цьому разі векторні команди отримували свої операнди із дуже швидкої пам’яті – векторних регістрів і записували результат знову ж у векторних регістрах. **Першою** векторноконвеєрною системою була Cray-1 (появилася в 1976 р.). Її архітектура виявилась настільки вдалою, що ця система започаткувала ціле сімейство обчислювальних систем. Деякі із векторноконвеєрних систем мали більш складну **ієрархію пам’яті**. Наприклад, в Cray-2 кожному процесору, крім векторних регістрів була надана швидка локальна пам’ять обсягом в 16000 слів, а системи серії Cray X-MP були облаштовані масовою пам’яттю, що була повільніша за оперативну, але значно швидша, ніж дискова пам’ять.

Для виконання векторних операцій векторні процесори мали конвеєризовані арифметичні пристрої. Обчислювальні системи фірми Control Data Corporation використовували арифметичні пристрої із змінною конфігурацією, в яких один і той же конвеєр міг виконувати різні арифметичні операції.

Системи Cray мали окремі конвеєри для виконання додавання, множення та деяких інших функцій.

Японська система NEC SX-2 мала декілька конвеєрів для операцій одного типу, наприклад, чотири конвеєри для додавання, чотири – для множення.

Більшість векторноконвеєрних систем мала окремі пристрої для скалярної арифметики, які могли теж бути конвеєризованими, але відрізнялися від векторних конвеєрів тим, що не використовували векторних операндів. Ці пристрої могли працювати паралельно з векторними конвеєрами.

Однією із **перших паралельних** обчислювальних **систем** була матрична система ILLIAC IV (1972 р.), що складалася із 64 процесорів, між якими обмін даними здійснювався через спеціальну матрицю комунікаційних каналів. Ця особли-

вість і дала назву «матричні системи» відповідному класу обчислювальних систем, в яких одним потоком команд оброблялися декілька потоків даних. Такі системи відзначалися великою кількістю процесорів, наявністю оперативної пам'яті в кожному із них та спільної оперативної пам'яті великого обсягу. Матричні обчислювальні системи були призначені для розв'язування спеціальних задач з допомогою високопаралельних алгоритмів. Спочатку вони виготовлялися поштучно і їх вартість була дуже високою. Тоді найбільш характерною тенденцією під час створення багатопроцесорних систем була розробка систем із великою кількістю процесорів і малим обсягом запам'ятовуючих пристроїв кожного процесора. Така архітектура мала більше ознак спеціалізованої, аніж універсальної. У зв'язку з цим швидко почали **поширюватись векторноконвеєрні** системи.

Тривалі дослідження та практичний досвід показали, що найбільш перспективними в сенсі підвищення продуктивності та надійності є універсальні системи, в яких декілька потоків даних обробляються декількома потоками команд.

Паралельна обчислювальна система може мати дуже прості процесори, придатні для розв'язування малих або обмежених задач, а може мати і досить потужні векторноконвеєрні процесори. Так, в системах Goodyear MPP та Connection Machine процесори були порівняно простими однобітовими пристроями. В Goodyear MPP їх було 16484, а в Connection Machine – 64936.

Перші промислові зразки **багатопроцесорних** обчислювальних **систем** появились на базі векторноконвеєрних процесорів у 80-х роках минулого століття. Прикладами таких систем були:

серія Cray X-MP (спочатку дво-, а потім чотирипроцесорна система);
Cray-2 (4 процесори);
ETA-10 (8 процесорів).

Найбільш розповсюдженими, але і надзвичайно дорогими були системи Cray, які об'єднували від 2 до 16 процесорів, що мали рівноправний (симетричний) доступ до спільної оперативної пам'яті.

Альтернативою стосовно таких дорогих систем була пропозиція будувати еквівалентні за потужністю багатопроцесорні системи із великої кількості дешевих **серійних мікропроцесорів**. Але досить швидко виявилось, що згадана архітектура має обмежені можливості стосовно нарощування кількості процесорів у системі із-за різкого збільшення конфліктів під час звертання до спільної шини пам'яті.

Виходом із даної ситуації стала ідея оснащення кожного процесора своєю оперативною пам'яттю і перетворення системи в об'єднання незалежних обчислювальних вузлів. Такий підхід значно збільшив степінь масштабування систем, але вимагав розробки спеціального способу **обміну даними** між вузлами, який зазвичай реалізується у вигляді **механізму передачі повідомлень**.

На даний час **технології розроблення паралельних** обчислювальних **систем** розвиваються за чотирма головними напрямками:

- векторноконвеєрні системи;
- SMP (Symmetric Multi-Processing)-системи;

- MPP (Massively Parallel Processing)-системи;
- кластерні системи.

Зауважимо, що дещо середнім між SMP та MPP є

NUMA (Non Uniform Memory Architecture)-архітектури,

в яких пам’ять фізично розподілена, але логічно є загальнодоступною. До того ж час доступу до різних блоків пам’яті в таких системах є неоднаковим. Однією з перших систем цього типу була Cray T3D.

Історично **векторноконвеєрні** обчислювальні системи були першими системами, які почали називати **суперкомп’ютерами**.

Курс «Паралельні обчислення та засоби їх реалізації».

Лекція 3. Паралельні обчислювальні системи та їх класифікація (II ч.).

План лекції.

1. Структура сучасних векторноконвеєрних систем.
2. Структурні особливості SMP-систем.
3. Системи масового паралелізму (MPP-системи).
4. Кластерні системи.
5. Деякі нетрадиційні підходи до побудови паралельних обчислювальних систем.
6. Приклади класифікацій паралельних систем (за Фліном, за Хокні, класифікація суперкомп'ютерів, V-класифікація). Переваги та недоліки цих класифікацій.

1. Як було зазначено у попередній лекції, основними принципами, закладеними в архітектуру **векторноконвеєрних систем**, є:

- конвеєрна організація оброблення потоку команд;
- введення в систему команд набору векторних операцій, які дозволяють оперувати з масивами даних.

Довжина оброблюваних векторів у сучасних векторноконвеєрних системах складає, зазвичай, 128 або 256 елементів. Основне призначення векторних операцій полягає в розпаралелюванні виконання операторів циклу, в яких зосереджена велика частина обчислювальної роботи.

Сучасні **векторноконвеєрні** системи мають **ієрархічну структуру**:

- на нижньому рівні ієрархії розташовані **конвеєри операцій** (наприклад, конвеєр додавання дійсних чисел, конвеєр множення таких чисел тощо);
- деяка сукупність конвеєрів операцій об'єднується в **конвеєрний функціональний пристрій**;
- векторноконвеєрний **процесор** складається з низки конвеєрних функціональних пристроїв;
- декілька векторноконвеєрних процесорів (від 2 до 16), об'єднаних спільною пам'яттю, утворюють обчислювальний **вузол**;
- декілька таких вузлів об'єднуються з допомогою комутаторів, утворюючи або NUMA-систему, або MPP-**систему**.

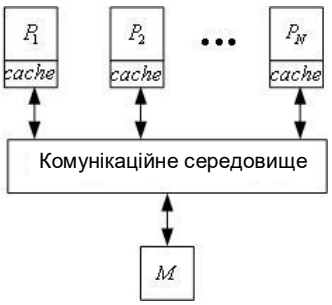
Типовими представниками такої архітектури є обчислювальні системи:

CRAY J90/T90; CRAY SV1; CRAY X1;
NEC SX-4/SX-5/SX-6/SX-7/SX-8/SX-8R/SX-9/SX-ACE;
Earth Simulator / 2 / 3.

2. Всі процесори **SMP-системи** мають симетричний доступ до пам'яті, тобто пам'ять такої системи є UMA (Unified Memory Access)-пам'яттю з одинаковим часом доступу. Під **симетричністю** розуміємо: рівні права всіх процесорів на дос-

туп до пам’яті, одну і ту ж адресацію для всіх елементів пам’яті, однаковий час доступу всіх процесорів системи до пам’яті (без врахування взаємного блокування).

Загальна **структура SMP-системи** має вигляд:



Тут P_i $(i = \overline{1, N})$ – процесори системи, а M – блок пам’яті.

Комунікаційне середовище SMP-системи будується на основі якої-небудь високошвидкісної системної шини або високошвидкісного комутатора. Окрім **однакових процесорів** і блоку спільної пам’яті M до цієї ж шини або комутатора підключаються **пристрої вводу-виводу**.

У таких системах виникають значні проблеми, пов’язані загалом з оперативною пам’яттю. Проблема конфліктів під час звертання до спільної шини була частково усунута завдяки **розділенню пам’яті на блоки**, підключення до яких за допомогою комутаторів дозволило розпаралелити звертання від різних процесорів. За такого підходу SMP-системи змогли об’єднувати вже до **32 процесорів**.

Щоб усунути розрив між швидкістю роботи оперативної пам’яті та швидкістю роботи процесора, сучасні процесори облаштовуються високошвидкісною буферною пам’яттю (**кеш- пам’яттю**). Швидкість доступу до цієї пам’яті в **декілька десятків разів** є більшою за швидкість доступу до **основної пам’яті** процесора. Однак, наявність кеш-пам’яті порушує принцип рівноправного доступу до будь-якої точки пам’яті, оскільки дані, що знаходяться в кеш-пам’яті одного процесора, недоступні для інших процесорів. Тому після кожної модифікації копії змінної, що знаходиться в кеш-пам’яті якого-небудь процесора, необхідно здійснювати синхронну модифікацію самої цієї змінної, що розташована в основній пам’яті. У сучасних SMP-системах **когерентність** кеш-пам’яті підтримується апаратно або операційною системою.

Із-за обмеженої пропускної здатності комунікаційного середовища SMP-системи **погано масштабуються**. Відомим **недоліком** їх є те, що їх вартість зростає швидше, ніж продуктивність при збільшенні кількості процесорів у системі. Зауважимо, що SMP-системи появилися як **альтернатива** для дорогих **багатопроцесорних систем** на базі векторноконвеєрних процесорів.

Найбільш відомими SMP-системами є, наприклад,

SMP-сервери IBM, HP, Compaq, Dell, Fujitsu

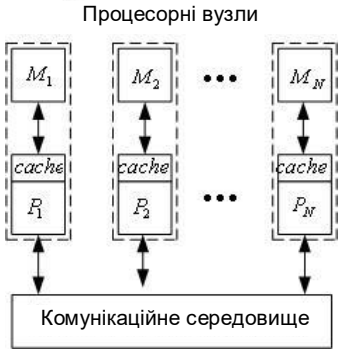
тощо. Така система функціонує під керуванням єдиної операційної системи (найчастіше – Unix і подібної до неї).

3. Обчислювальні системи масового паралелізму (**MPP-системи**) є багатопроцесорними обчислювальними системами з розподіленою пам'яттю, в яких з допомогою деякої комунікаційної мережі об'єднуються однорідні обчислювальні вузли. Кожен **вузол** складається з одного або декількох процесорів, власної оперативної пам'яті, комунікаційного обладнання, підсистеми вводу/виводу. Процесори в таких системах мають прямий доступ лише до своєї локальної пам'яті.

Порівняно з SMP-системами, архітектура MPP-системи **усуває** одночасно як **проблему конфліктів** під час звертання до пам'яті, так і **проблему когерентності** кеш-пам'яті. **Головною перевагою** MPP-систем є висока степінь масштабування. Успішно функціонують такі системи з **сотнями і тисячами** процесорів, зокрема, в

ASCI Blue Mountain – 6144 процесори (запуск в 1998 р.; зупинена в 2004 р.);
ASCI White – 8192 процесори (запуск в 2001 р.; списана в 2006 р.);
CRAY XC40 – 185088 ядер.

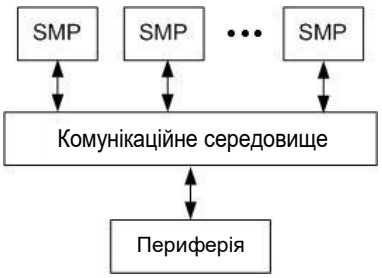
Структура MPP-системи має вигляд:



Тут кожен процесор P_j має свій блок пам'яті M_j для всіх $j = \overline{1,N}$.

Практично всі **рекорди з продуктивності** на сьогоднішній день встановлені саме на MPP-системах. Останнім часом для з'єднання обчислювальних вузлів у них все частіше використовується ієрархічна система **високошвидкісних комутаторів**, яка вперше була реалізована в системах **IBM SP2**. Ця топологія дає можливість прямого обміну даними між будь-якими вузлами без залучення проміжних вузлів.

У попередній лекції ми згадували про **NUMA-систему**, яка зазвичай будується на базі однорідних процесорних вузлів, які складаються з невеликої кількості процесорів і блоку пам'яті. Ці вузли об'єднуються з допомогою деякого високошвидкісного комунікаційного середовища. **Структура** такої системи має вигляд:



При цьому підтримується єдиний адресний простір. Апаратно підтримується доступ до віддаленої пам'яті, тобто до пам'яті інших вузлів. Доступ до локальної пам'яті здійснюється в декілька разів швидше, ніж до віддаленої.

По суті NUMA-система є MPP-системою, де як окремі обчислювальні елементи використовуються SMP-вузли. Логічна загальнодоступність пам'яті в NUMA-системах, з одного боку дозволяє працювати з **єдиним адресним простором**, а, з іншого боку, дозволяє досить просто забезпечити високу масштабованість системи. Сеймур Крей (Seymour R. Cray) додав до такої системи новий елемент – когерентний кеш, створивши так звану архітектуру cc-NUMA (Cache Coherent Non-Uniform Memory Access), яка розшифровується як «неоднорідний доступ до пам'яті із забезпеченням когерентності кешів». Найбільш відомими системами архітектури cc-NUMA є:

HP 9000 V-class в SCA-конфігураціях,
SGI Origin3000,
Sun HPC 15000,
IBM/Sequent NUMA-Q 2000.

На сьогодні максимальна кількість процесорів у cc-NUMA-системах може перевищувати 1000 одиниць.

4. Кластерні системи – це логічне продовження розвитку ідей, закладених в архітектурі MPP-систем, фактично кластери є більш дешевим варіантом таких систем. Кластерні технології дозволяють для досягнення необхідної продуктивності обчислень об'єднувати в єдині обчислювальні системи засоби різного типу, починаючи від ПК і закінчуючи потужними суперкомп'ютерами. Такі технології набули широкого розповсюдження як засіб створення систем суперкомп'ютерного класу із складових частин масового виробництва, що значно здешевлює вартість обчислювальної системи загалом.

Кластерна система складається з окремих обчислювальних вузлів, об'єднаних єдиним комунікаційним середовищем. Кожен вузол має свою локальну оперативну пам'ять. При цьому **спільної** фізичної **оперативної пам'яті** для вузлів зазвичай **не існує**. На склад та архітектуру вузлів не накладаються жодні обмеження і кожен з них може функціонувати під керуванням власної операційної системи. Комунікаційне середовище кластера зазвичай дозволяє вузлам **взаємодіяти** між собою лише за допомогою **передачі повідомлень**.

Зараз розроблено цілу низку **технологій об'єднання вузлів** кластера. Найбільш **поширеною** (бо **проста** у використанні та має **низьку вартість**) є

Fast Ethernet (100 Mbit/s).

Більш ефективними є технології

Gigabit Ethernet (1 Gbit/s),
10/40 Gigabit Ethernet,

а також швидкісні мережі, такі як

100 Gigabit Ethernet,

InfiniBand (100 Gbit/s, після 2023 р. – 250 Gbit/s),
Intel Omni-Path (100 Gbit/s, у 2019 р. відміна створення технології з продуктивністю 200 Gbit/s)

тощо.

У середині 70-х років XX ст. за кількістю застосувань лідерами були **векторноконвеєрні системи**, а після 2002 р. лідерами стали **кластери**. Векторноконвеєрні системи, обладнані спеціалізованими процесорами для оброблення одразу наборів даних, зберегли переваги у швидкодії та **простоті програмування** порівняно з кластерами. Зате останні використовують у вузлах **стандартні процесори** і є значно **дешевшими**, хоча потребують більш складного програмного забезпечення.

У першій лекції уже відзначалось, що пік розробок паралельних обчислювальних систем у колишньому Радянському Союзі припав на 1986 р. До наведених проектів, які фінансувалися тоді, можна додати:

- матричний спецпроцесор ЕС2700, м. Єреван;
- мультипроцесор з динамічною архітектурою ЕС2704, м. Ленінград;
- мультипроцесор ЕС2706, м. Таганрог;
- сімейство мультипроцесорів ПС ІПУ АН СРСР, м. Москва;
- системи НДІ «Квант», м. Москва.

Після розпаду Радянського Союзу майже 10 років на колишніх його теренах не приділялась належна увага дослідженням, пов'язаним із високопродуктивними обчисленнями. І лише у 2000 р. на науковій конференції в Черноголовці (Росія) було засвідчено значний інтерес до цієї проблеми.

Світова тенденція стосовно розробок та поширення кластерних систем спостерігається і в країнах СНД. Зокрема, в Росії кластерні системи почали створювати на базі систем сімейства МВС (МВС-100, МВС-1000); у Білорусі (спільно з Росією) розроблено кластери сімейства СКИФ; у Вірменії (спільно з Росією) розроблено Armcluster; в Україні створено ціле сімейство кластерів СКІТ (ІК ім. В.М. Глушкова НАН України). Однак це був лише початок…

Зараз триває **процес масової модернізації** кластерів на підставі переходу на **багатоядерні процесори** виробництва компаній IBM, Intel, AMD, Sun та залучення графічних процесорів. В Україні встановлено кластерні системи у багатьох вузах та НДІ НАН України, зокрема і у Львові (ІФКС НАН України, ФМІ НАН України, ІППММ НАН України тощо).

Одним із шляхів **створення значних** обчислювальних **ресурсів** та головним інтеграційним напрямком для всієї суперкомп'ютерної галузі є обчислювальні мережі **grid computing**. Також розвивається концепція **розподілених обчислювальних середовищ** для розв'язання складних обчислювальних задач на доступних розподілених неоднорідних засобах.

На підставі викладеного можна зробити **висновок**, що за останні більш як 20 років паралельні обчислювальні системи перетворились із засекречених засобів, які використовувались виключно з оборонною метою, у могутній високоінтелектуальний ресурс для відкритих застосувань.

5. Крім розглянутих вище **широко розповсюджених** підходів до побудови досить універсальних паралельних обчислювальних систем знаходять своє застосування і інші, можливо **менш поширені**, але перспективні підходи, наприклад:

- розроблення **нейрокомп'ютерів** – апаратних засобів для реалізації нейромережних методів і алгоритмів;
- створення **VLIW** (Very Long Instruction Word)-**систем** з дуже довгим командним словом, які дозволяють одночасне виконання набору простих команд;
- розроблення **систем із структурно-процедурною** організацією обчислень;
- побудова **систолічних та квазісистолічних** обчислювальних структур.

6. Із викладеного випливає, що існує велика кількість обчислювальних систем і, очевидно, виникає бажання ввести для них яку-небудь **класифікацію**. Класифікація потрібна для того, щоб легше орієнтуватися під час вибору обчислювальної системи (якщо є така можливість) для реалізації заданого алгоритму розв'язання задачі або програми, що описує цей алгоритм. Загалом відомо дуже багато різних класифікацій паралельних обчислювальних систем. Ми розглянемо деякі із них, які є **найбільш відомими**:

- класифікація за М.Д. Фліном;
- класифікація за Р. Хокні;
- класифікація суперкомп'ютерів;
- V-класифікація.

Найбільш розповсюдженою стала класифікація обчислювальних систем, запропонована в 1966 р. професором Стенфордського університету **М.Д. Фліном** (M.J. Flynn). Ця класифікація **охоплює** лише **дві ознаки** – тип потоку команд і тип потоку даних.

В **одинарному потоці** команд в один момент часу може виконуватися лише одна команда. У цьому випадку ця єдина команда визначає в даний момент часу роботу всіх або хоча б багатьох пристроїв обчислювальної системи.

У **множинному потоці** команд в один момент часу може виконуватися багато команд. У цьому випадку кожна із таких команд визначає в даний момент часу роботу лише одного або лише декількох (**але не всіх**) пристроїв обчислювальної системи.

В одинарному потоці послідовно виконуються окремі команди, а в множинному – групи команд.

Одинарний **потік даних** обов'язково передбачає наявність в обчислювальній системі лише одного пристрою оперативної пам'яті і одного процесора. Однак при цьому **процесор** може бути якзавгодно **складним**, так що процес обробки кожної одиниці інформації в потоці може вимагати виконання багатьох команд.

Множинний потік даних складається із багатьох одинарних потоків даних.

Відповідно до сказаного всі обчислювальні системи **за Фліном** діляться на чотири типи:

- SISD (Single Instruction Single Data);

- MISD (Multiple Instruction Single Data);
- SIMD (Single Instruction Multiple Data);
- MIMD (Multiple Instruction Multiple Data).

Обчислювальна система **SISD** є звичайною **послідовною** (однопроцесорною) ЕОМ фон-неймановської архітектури.

На обчислювальні системи **MISD** існують різні погляди. За одним із них – за всю історію розвитку обчислювальної техніки такі системи **не були створені**. За іншим (менш поширеним, ніж перший) MISD-система – це система, в якій кілька процесорів виконують різні потоки команд і типовим представником цих систем є векторноконвеєрні обчислювальні системи. Ми будемо дотримуватися першого погляду.

У системах **SIMD** кілька процесорів одночасно (синхронно) виконують **одну** і ту ж **команду**, але над **різними потоками даних**. Ці системи діляться на два великі класи:

- векторноконвеєрні системи;
- векторнопаралельні або матричні обчислювальні системи.

Представниками SIMD-систем є:

ICL DAP, STARAN, BSP,
ILLIAC IV, Connection Machine, ПС-2000 тощо.

Принцип SIMD реалізується в сучасних графічних процесорах

Обчислювальні **системи MIMD** складаються із багатьох процесорів, які (зазвичай, асинхронно) виконують **різні команди** над **різними даними**. Зараз це найпоширеніший клас архітектур паралельних систем. Ці системи ще називають **багатопроцесорними**. Представниками MIMD-систем є:

Intel Paragon, Cray T3D, CM (Connection Machine) -5,
Tianhe-2A - TH-IVB-FEP Cluster, Pioneer-EUS - NDv4 cluster,
Amazon EC2 Instance Cluster us-east-1a тощо.

Розглянута **класифікація Фліна дозволяє** за приналежністю обчислювальної системи до класу SIMD або MIMD зрозуміти базовий принцип її роботи. Досить часто цього є достатньо. **Недоліком** цієї класифікації є «**перевантаженість**» класу **MIMD**. Крім цього, вона є неповною бо, наприклад, вона не включає систолічні обчислювальні структури – набори зв'язаних однотипних процесорів (іноді дуже простих), що **синхронно виконують різні програми**.

Пізніше, у 1985 р., появилася класифікація **Р. Хокні** (R. Hockney), яка детально класифікує обчислювальні системи, які за Фліном потрапляють у клас MIMD. У цьому випадку виділяється три групи систем:

- конвеєрні системи;
- з комутатором (системи зі спільною та розподіленою пам'яттю);
- мережеві системи (регулярні решітки, ієрархічні структури, системи зі змінною конфігурацією, гіперкуби).

Зараз **статус суперкомп'ютерів** у світі визначається лише за обчислювальними системами, що увійшли до останньої версії випуску рейтингу top500

(www.top500.org). Згідно з редакцією рейтингу за листопад 2021 року найбільше таких систем встановлено в Китаї (173), далі йдуть США (149), Японія (32), Німеччина (26), Франція (19).

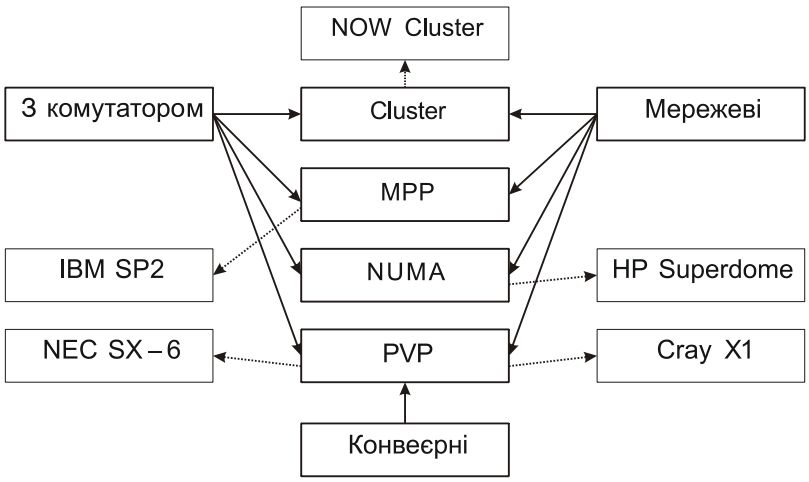
Для класифікації суперкомп'ютерних систем прийнято такі терміни:

- SMP-системи;
- NUMA-системи;
- MPP-системи;
- Cluster Systems – кластерні системи;
- PVP (Parallel Vector Processing) – паралельні векторні системи.

Наведені класи суперкомп'ютерних систем мають особливості, властиві різним класам систем за Хокні. До того ж всі обчислювальні системи, що потрапляють в список надпотужних систем світу, мають властивості як **мережевих** систем, так і систем **з комутатором**.

У 2004 р. була введена V-класифікація для сучасних суперсистем і їх відповідності класифікації Хокні. На цю класифікацію відображаються найбільш розповсюджені системи, наприклад, IBM SP2, NEC SX-6, HP Superdome, Cray X1, NOW Cluster тощо.

Згідно V-класифікації Cluster-, MPP-, NUMA-, PVP-системи мають властивості як систем з комутатором, так і мережевих і крім цього PVP-системи мають властивості конвеєрних систем. Прикладом MPP-системи з комутатором є IBM SP2; PVP-системи з комутатором – NEC SX-6; NUMA-системи з мережею – HP Superdome; PVP-мережевої системи – Cray X1; Cluster-системи – NOW Cluster.



В архітектурі сучасних суперсистем, як і понад 10 років назад, переважає **напрямок**, пов'язаний з побудовою масовопаралельних систем (**MPP-систем**). Використання стандартних обчислювальних процесорів під час розробки таких систем є економічно доцільним і найбільш ефективним для досягнення **оптимального** співвідношення **ціна/швидкодія**. У цьому разі використання саме процесорів Intel і AMD дозволяє досягти високої обчислювальної потужності за мінімальних витрат, а **проблема** збільшення **швидкодії** ефективно вирішується лише шляхом **нарощування** кількості **процесорів** у системі.

Курс «Паралельні обчислення та засоби їх реалізації».

Лекція 4. Продуктивність паралельних обчислювальних систем та оцінювання паралельних алгоритмів.

План лекції.

1. Асимптотична продуктивність векторноконвеєрних систем.
2. Асимптотична продуктивність векторнопаралельних і багатопроцесорних систем.
3. Довжина напівпродуктивності.
4. Реальна продуктивність паралельних обчислювальних систем, гіпотеза Мінського.
5. Моделі машин для послідовних та паралельних обчислень, характеристики оцінювання паралельних алгоритмів (середній ступінь паралелізму, прискорення та ефективність).
6. «Парадокс» паралелізму.

1. Як уже підкреслювалось, одним із показників **продуктивності** паралельної обчислювальної системи є її **швидкодія**. Часто ці дві характеристики ототожнюють. Під **продуктивністю** ми будемо розуміти кількість операцій, які виконуються на заданій обчислювальній системі за одиницю часу. Продуктивність вимірюється в мільйонах команд за секунду (**MIPS** – millions instructions per second) або мільйонах операцій з плаваючою комою за секунду (**Mflops** – millions floating point operations per second).

Окрім продуктивності важливими характеристиками обчислювальних систем є їх **масштабованість**, тобто придатність до нарощування та зменшення ресурсів, а також **реконфігурованість**, тобто можливість зміни кількості обчислювальних вузлів і графа їх зв'язків, що дозволяє забезпечувати **надійність** обчислювальної системи.

Зауважимо, що **добре масштабована** система забезпечує **лінійний ріст** продуктивності у разі збільшення в ній кількості процесорів.

Як уже зазначалось, основною ознакою **векторноконвеєрних** систем є наявність **конвеєрних функціональних пристроїв**, які містять низку конвеєрів операцій (наприклад, конвеєр операції додавання дійсних чисел, конвеєр операції множення таких же чисел тощо). Тому оцінка продуктивності таких систем базується на оцінці продуктивності конвеєрів операцій.

Методику оцінки продуктивності конвеєрів операцій розглянемо на прикладі конвеєра операції додавання. Вважаємо, що у нас є l-етапний конвеєр операції додавання і всі етапи конвеєра вимагають однакового часу виконання Δt. Тоді для виконання операції додавання векторів $\vec{x} = (x_1, x_2, \ldots, x_n), \vec{y} = (y_1, y_2, \ldots, y_n)$ потрібен час:

$$T_{конв} = (s + l + n)\Delta t, \qquad (1)$$

де $s\Delta t$ – фіксований час запуску конвеєра;
$l\Delta t$ – час «розгону» конвеєра, тобто його заповнення.

Після запуску конвеєра та його заповнення він видає результат через кожен такт, тривалістю Δt.

Із наведеної формули (1) випливає, що **середній час** одержання одного результату (тобто виконання однієї операції додавання) дорівнює

$$T_R = \left(\frac{s+l}{n} + 1 \right) \Delta t . \qquad (2)$$

Із формули (2) випливає, що необхідно працювати з достатньо довгими векторами, щоб амортизувати по багатьох результатах витрати на запуск та заповнення конвеєра.

Продуктивність конвеєра будемо характеризувати кількістю результатів, які він видає за одиницю часу. Ця кількість виражається формулою

$$R = (T_R)^{-1} = \frac{n}{(s+l+n)\Delta t} . \qquad (3)$$

Якщо знехтувати $s\Delta t$, $l\Delta t$ (це є можливим, коли n є набагато більшим за s, l), то із (3) одержуємо

$$R_\infty = \frac{1}{\Delta t} .$$

Ця величина називається **асимптотичною продуктивністю**. Вона дорівнює (недосяжній) максимальній швидкості видачі результатів конвеєром, яка одержується унаслідок ігнорування часових витрат на запуск і заповнення конвеєра. При цьому вважається, що **відсутні конфлікти** під час звертання до пам'яті.

Аналогічна ситуація виникає для конвеєрів будь-яких операцій. Умовно прийнято говорити, що **асимптотична продуктивність** конвеєра операцій досягається на **векторах нескінченної довжини**.

Під час виконання етапів конвеєра в строго послідовному режимі, очевидно, що максимальна швидкість видачі результатів дорівнює

$$(R_\infty)_{посл} = \frac{1}{l\Delta t} .$$

Отже, конвеєрна обробка збільшує продуктивність обчислювальної системи в l разів (на векторах нескінченної довжини).

2. Методику оцінки **продуктивності векторнопаралельних і багатопроцесорних (MIMD)** систем також розглянемо на прикладі операції додавання векторів $\vec{x} = (x_1, x_2, \ldots, x_n)$, $\vec{y} = (y_1, y_2, \ldots, y_n)$ на N-процесорній системі. Час виконання цієї процедури на обох типах обчислювальних систем можна оцінити за формулою

$$T_{пар} = T_{com} + T_{cal} ,$$

де $T_{com} = \mathrm{O}(d \lfloor n/N \rfloor \nu)$ – час комунікацій;

$T_{cal} = \lfloor n/N \rfloor \tau$ – час обчислень;

d – діаметр комунікаційної мережі системи;

$\lfloor A \rfloor$ – найближче ціле, більше за A;

ν – продуктивність (пропускна здатність) каналів міжпроцесорного обміну;

τ – час виконання операції додавання двох чисел на одному процесорі системи.

Якщо знехтувати часом на комунікації, то за час виконання операції додавання компонент x_i, y_i на N процесорах можна прийняти

$$T_R^{nap} = \frac{T_{cal}}{n} = \frac{\lfloor n/N \rfloor \tau}{n}.$$

За одиницю часу N-процесорна векторнопаралельна або багатопроцесорна система видає таку кількість результатів:

$$R^{nap} = (T_R^{nap})^{-1} = \frac{n}{\lfloor n/N \rfloor \tau}. \qquad (4)$$

Якщо вважати, що величина n є кратною N і нехтування часу на комунікації передбачає, що немає конфліктів під час звертання до пам'яті, то одержимо із (4), що

$$R_\infty^{nap} = \frac{N}{\tau}.$$

Ця величина називається **асимптотичною продуктивністю** для векторнопаралельних та багатопроцесорних систем. Вона дорівнює (недосяжній) максимальній швидкості видачі результатів обчислювальною системою за названих вище припущень стосовно часу комунікацій та кратності n.

Під час додавання векторів $\bar{x}, \bar{y}$ на одному процесорі системи **максимальна швидкість** видачі результатів, очевидно, дорівнюватиме

$$(R_\infty)^{посл} = \frac{1}{\tau}.$$

Отже, паралельне додавання векторів збільшує продуктивність векторнопаралельних і багатопроцесорних систем максимум в N раз. Аналогічною є ситуація під час виконання на цих системах будь-яких бінарних операцій.

3. Важливою характеристикою **векторних** обчислювальних систем є величина $n_{1/2}$ – довжина векторів, для якої досягається **половина асимптотичної продуктивності** системи (інакшими словами, **довжина напівпродуктивності**). Зміст асимптотичної продуктивності і довжини напівпродуктивності є різним.

Асимптотична продуктивність насамперед характеризує **технологію** виготовлення ЕОМ, а **довжина напівпродуктивності** є **критерієм ступеня** паралельності системи.

Розглянемо величину:

$$\mu = \frac{n_{1/2}}{n_0},$$

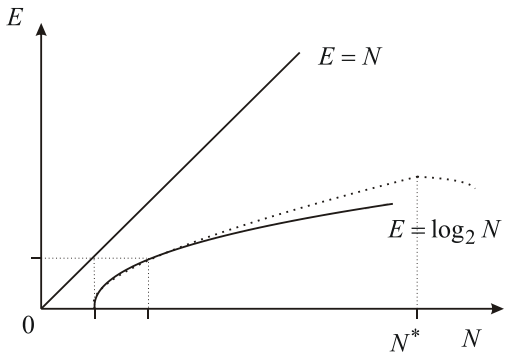
де n_0 – середня довжина оброблюваних векторів (кількість їх компонент). Тоді, якщо $\mu \approx 0$, то це означає, що алгоритм може бути ефективно розпаралелений для виконання на даній обчислювальній системі, а якщо $\mu \approx 1$, то згаданий алгоритм не можна ефективно реалізувати на цій системі. Отже, чим більшою є величина μ, тим меншим є ефект від розпаралелювання.

4. Наведені вище **оцінки продуктивності** паралельних обчислювальних систем є більше **теоретичними** і не завжди досяжними. Реальну продуктивність паралельних систем перевіряють на тестах. В обчислювальній практиці найчастіше застосовують **стандартні тести**. Оскільки більшу частину часу виконання програм зазвичай займають цикли, то часто якраз вони застосовуються у якості тестів. У даний час найбільш відомим тестом продуктивності є набір тестів **Linpack**, що є фактично набором програм для розв'язання СЛАР методом виключення Гауса. Головним параметром тестів Linpack є порядок системи n. Зазвичай використовуються тести з $n=1000$. Оскільки відомою є кількість операцій (як функція розмірності СЛАР n), які необхідно виконати для розв'язання СЛАР методом виключення Гауса, тому, знаючи час розв'язання задачі, легко знайти продуктивність обчислювальної системи в Mflops. Відомий список top500, який містить 500 найбільш високопродуктивних систем світу, складається саме на підставі тестування з допомогою тестів Linpack.

Для **MPP-систем** більш цікавим є тест **HP Linpack (Linpack's Highly Parallel Computing)** (тому він і найчастіше для них використовується), в якому продуктивність вимірюється при великих значеннях n і кількості процесорів.

Для високопаралельних **суперсистем** останнім часом все більше використовуються тести **NAS parallel benchmark**, які є набором алгоритмів розв'язання деяких задач обчислювальної газо- та гідродинаміки.

<u>Гіпотеза Мінського (Марвін Лі Мінський (M.L. Minsky)):</u> в N-процесорній векторнопаралельній обчислювальній системі або MIMD-системі, в якій продуктивність кожного процесора дорівнює 1, загальна продуктивність системи E росте, як $\log_2 N$ (див. рисунок).



У **перших паралельних системах**, коли кількість процесорів була невеликою, гіпотеза Мінського **підтверджувалась**. У сучасних системах з великою кількістю процесорів продуктивність залежить від кількості процесорів таким чином, як показано на рисунку штриховою лінією. Головними причинами такої залежності є те, що:

- із збільшенням кількості процесорів збільшуються комунікаційні затрати (унаслідок збільшення діаметра комунікаційної мережі);
- із збільшенням кількості процесорів більш незбалансованим стає їх завантаження корисною роботою.

Отже, судячи з рисунку, якщо кількість процесорів системи N досягає величини N^*, то немає сенсу далі нарощувати цю кількість.

5. Відомими **мірами складності** для **послідовних** алгоритмів є **часова** $T(n)$ складність алгоритмів, яка вимірюється кількістю кроків (операцій), які необхідно виконати, і **просторова** складність $S(n)$, що вимірюється кількістю пам'яті, що є необхідна для роботи алгоритму. Тут n – це розмір вхідних даних, що інтерпретуються відповідно до предметної області розв'язуваної задачі.

Для паралельних алгоритмів часова і просторова складності також залишаються актуальними, але мають свої особливості. По-перше, вони стають залежними від кількості процесорів p, на яких виконується алгоритм розв'язання задачі: $T_p(n)$, $S_p(n)$. По-друге, одиницею виміру в $T_p(n)$ тепер є **паралельна операція**, тобто така, що подає одночасно виконувані обчислювальні кроки як одну операцію. І, по-третє, сама просторова складність тепер визначається як кількість процесорів (зазвичай максимальна), необхідних для виконання алгоритму розв'язання задачі.

Для оцінки складності послідовних обчислень використовується **модель RAM** (random access memory) – машини з пам'яттю довільного доступу. Основними припущеннями цієї моделі є:

1) модель складається із вхідної стрічки (лише для читання), вихідної стрічки (лише для запису), процесора та пам'яті;
2) пам'ять є нескінченною множиною регістрів, здатних зберігати числа довільної розмірності;
3) операції RAM-машини становлять функціонально повний набір (завантаження регістрів, запис результату, бінарні числові операції тощо).

Отже, RAM-машина є універсальною моделлю машини Тьюрінга.

Модель для виконання паралельних обчислень **PRAM** є узагальненням моделі RAM завдяки:

1) збільшенню кількості ідентичних процесорів до $N > 1$, що є параметром моделі і визначається як функція від розміру вхідних даних;
2) наданню нескінченній множині регістрів властивості спільного доступу з боку процесорів;
3) розширенню множини команд командами читання та запису для спільної пам'яті.

Така модель, звичайно, є абстракцією паралельних обчислювальних систем, в якій не враховані важливі особливості реальних систем. Так, у моделі **немає** спеціальних **засобів синхронізації**, але передбачається певна стратегія для розв'язання конфліктів при одночасному звертанні до спільної пам'яті кількох процесорів. Крім того, в моделі PRAM **не враховуються втрати часу на обміни** даними. Модель PRAM хоч і має спільну пам'ять, проте це не означає, що вона застосовується лише до відповідних архітектур зі спільною пам'яттю.

В ідеальному випадку алгоритм складності T_1 в RAM-моделі мусив би мати складність $T_p = T_1 / p$ в PRAM, де p – це кількість процесорів. Тобто прискорення

обчислень на мультипроцесорній системі мало б визначатись кількістю процесорів. Проте, очевидно, це не завжди так відбувається і визначається характеристиками паралелізму для конкретних алгоритмів обчислень.

Для **оцінки паралельних алгоритмів** загалом використовують такі величини:

- середній ступінь паралелізму;
- прискорення паралельного алгоритму;
- ефективність паралельного алгоритму.

Середнім ступенем паралелізму алгоритму називається відношення загальної кількості операцій алгоритму до висоти його ярусно-паралельної форми. Наприклад, для алгоритму додавання m чисел за схемою повного двійкового дерева середній ступінь паралелізму дорівнює

$$(m-1)/\log_2 m,$$

тобто є $O(m/\log_2 m)$.

Ідеальний паралельний алгоритм (наприклад, алгоритм додавання двох векторів, які мають по l компонент) має ступінь паралелізму l.

Середній ступінь паралелізму характеризує лише сам алгоритм. З цієї характеристики ніяк не випливає ефективність виконання алгоритму на конкретній паралельній обчислювальній системі.

Прискорення паралельного алгоритму є його найбільш інформативною характеристикою, яка показує, у скільки разів застосування паралельного алгоритму зменшує час розв'язання задачі порівняно з послідовним алгоритмом. Прискорення паралельного алгоритму визначається величиною

$$S_p = \frac{T_1}{T_p} \le p.$$

Ця величина (S_p) залежить від кількості процесорів. Крім того, на практиці є низка факторів, які можуть знижувати S_p, а саме:

- спільні ресурси (пам'ять, канали зв'язку), що можуть знижувати продуктивність паралельної системи;
- паралельні процеси можуть втрачати час, очікуючи виконання умов синхронізації;
- послідовна частина обчислень, які не можуть мати паралельної реалізації.

Ефективність паралельного алгоритму визначається величиною

$$E_p = S_p / p.$$

З цього визначення випливає, що $0 < E_p \le 1$, отже ефективність характеризує **питоме прискорення**, що припадає на один процесор.

6. «Парадокс паралелізму» полягає у досягненні величинами прискорення і ефективності паралельного алгоритму значень, які **перевищують** відповідно ***p*** і **1**. Іншими словами «парадокс» полягає у «суперлінійному» зростанні продуктивності паралельної обчислювальної системи із збільшенням кількості її процесорів.

Насправді цей «парадокс» за своєю суттю не є парадоксом, оскільки прискорення, що перевищує ***p***, можливе з двох головних причин:

1) сумарна оперативна пам'ять паралельної системи перевищує оперативну пам'ять послідовної ЕОМ, з якою проводиться порівняння, а це дозволяє на паралельній системі розмістити всю програму в оперативній пам'яті; на послідовній ЕОМ доводиться використовувати вивантаження програми на накопичувач на жорстких магнітних дисках;

2) використання на паралельній обчислювальній системі апріорі паралельного алгоритму, особливості якого не можуть бути ефективно використані під час виконання обчислень на послідовній ЕОМ.

Курс «Паралельні обчислення та засоби їх реалізації».

Лекція 5. Компоненти комунікаційного середовища паралельних обчислювальних систем.

План лекції.

1. Структурна схема комунікаційного середовища.
2. Мережеві адаптери.
3. Комунікаційні мережі: адресація в мережах, мережеві протоколи, топологія комунікаційної мережі, мережеві концентратори.
4. Основні типи топологій комунікаційних мереж, діаметр комунікаційної мережі, мережі з фіксованою топологією та реконфігуровні, регулярні та нерегулярні мережі.
5. Мережеві комутатори: прості та складні комутатори.
6. Основні характеристики комунікаційних мереж.

1. Структурно **комунікаційне середовище** паралельної обчислювальної системи складається із **трьох** таких **компонент**:

- мережеві адаптери;
- комунікаційна мережа;
- мережеві комутатори.

Схематично це виглядає так, як подано на рис. 1. Зауважимо, що **комутатор** – це пристрій для комутації (встановлення зв'язку) двох і більше процесорів (або обчислювальних вузлів).

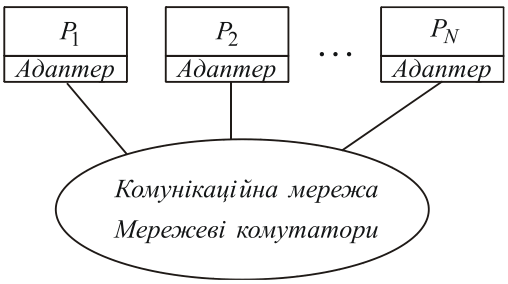


Рис. 1

Тут P_i – i-й вузол мережі (процесор або декілька процесорів, які утворюють обчислювальний вузол).

2. Під час обміну даними між вузлами мережі, що не є ближніми сусідами, відбувається передача (транспортування) даних через проміжні вузли. Очевидно, що у цих вузлах повинні знаходитися якісь **апаратні засоби**, що звільняють процесори вузлів від участі в передачі (транспортуванні) даних. Однією із основних функцій **мережевих адаптерів** є передавальна, тобто забезпечення обмінів даними в мережі.

Мережевий адаптер складається із:

- приймальнопередавальної частини, зв’язаної з вузловим процесором мережі;
- приймальнопередавальної частини, зв’язаної з мережею;
- узгоджувальних буферів (пристроїв, що використовуються для узгодження швидкостей обміну, розмірів блоків даних тощо).

Схематично структуру мережевого адаптера зображено на рис. 2.

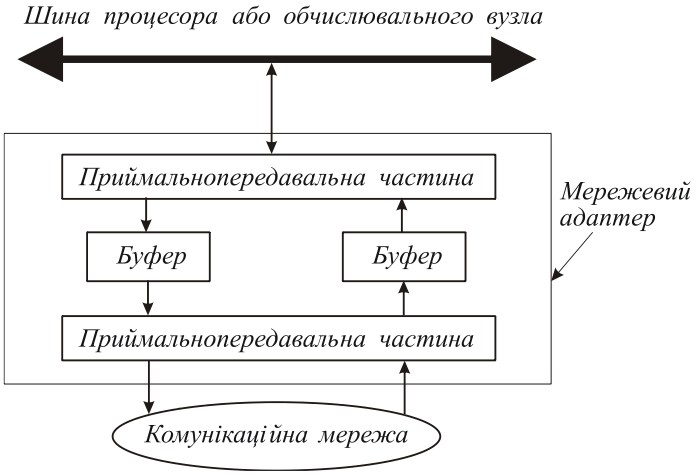


Рис. 2

Основною функцією **приймальнопередавальної** частини адаптера, зв’язаної з **комунікаційною мережею**, є реалізація протоколу мережі, а головною функцією такої ж частини адаптера, зв’язаної з **шиною вузла**, є реалізація протоколу цієї шини.

Функцією мережевого адаптера може бути також забезпечення **когерентності пам’яті** вузла мережі.

Якщо на вузловому процесорі одночасно функціонують декілька прикладних процесів, то мережевий адаптер може стати проблемою (**вузьким місцем**) для системи, оскільки за доступ до нього можуть змагатися прикладні процеси і процеси операційної системи. Тому інколи вузловий процесор забезпечують **двома мережевими адаптерами**: один – для процесів операційної системи; другий – для прикладних процесів.

3. Комунікаційні мережі діляться на **широкомасштабні** комунікаційні мережі (Wide Area Networks) – WAN-мережі та **локальні** комунікаційні мережі (Local Area Networks) – LAN-мережі.

Адресація в мережах. Множина адрес, допустимих у межах деякої схеми адресації, називається **адресним простором** комунікаційної мережі. Адресний простір може мати **лінійну або ієрархічну** організацію.

У **лінійному** адресному просторі комунікаційної мережі множина адрес не є структурованою, всі адреси є рівноправними. Прикладом лінійного адресного простору є простір MAC (Media Access Control)-адрес, які призначені для однозначної ідентифікації мережевих інтерфейсів у локальних мережах. Ці адреси розподіляються між виробниками спеціальним комітетом IEEE (Інститут інженерів з

електротехніки і електроніки) і вбудовуються в апаратуру (адаптери та інше мережеве обладнання) під час її виготовлення. Такі адреси ще називають апаратними, формат їх – 6 байт і позначаються вони двійковим або шістнадцятковим кодом (наприклад, 11 A0 17 3B FD 01).

Типовим представником **ієрархічного** адресного простору комунікаційної мережі є простір мережевих 4-байтових IP (Internet Protocol)-адрес, що використовуються для адресації вузлів у WAN-мережах. IP-адреси є дворівневими: адреса ділиться на старшу частину (номер мережі) і молодшу частину (номер вузла в мережі). Така організація IP-адрес дозволяє передавати повідомлення між мережами лише на підставі номера мережі. Номер вузла використовується після доставки повідомлення в середині мережі. IP-адреса однозначно ідентифікує окремий вузол, групу вузлів або цілу мережу та подається у десятковій (ОС), двійковій (адміністрування) або шістнадцятковій (дуже рідко) формі. Десяткова IP-адреса 102.56.187.5 означає вузол з номером 56.187.5, який знаходиться в мережі з номером 102.

Мережеві протоколи. Правила, що визначають послідовність і формат повідомлень, якими обмінюються вузли мережі, називаються **мережевим комунікаційним протоколом**.

У **WAN-мережах** зазвичай використовуються протоколи **комутації пакетів.** Комутація пакетів: у пункті відправлення повідомлення розбивається на порції (пакети, декілька кілобайт); пакети висилаються за адресою призначення незалежно один від одного; у пункті призначення здійснюється збір повідомлення із пакетів. Кожен пакет має свій заголовок, в якому вказується, як мінімум, адреса вузла-отримувача і номер пакета.

У паралельних обчислювальних системах використовуються LAN-мережі і як **протоколи комутації пакетів**, так і протоколи **комутації каналів**. Зв’язок з комутацією каналів – це зв’язок між двома вузлами мережі, між якими спочатку встановлюється з’єднання, перш ніж вони почнуть обмін інформацією (на протязі всього сеансу це з’єднання використовується лише тими вузлами).

Зазвичай локальні мережі будують на базі найбільш поширеної технології Ethernet. Технології Ethernet, Fast Ethernet, Gigabit Ethernet, 10GE забезпечують швидкість передачі даних відповідно 10, 100, 1000 і 10000 Mbit/s. Оскільки комунікаційне середовище є розділюваним ресурсом, то можливі ситуації, коли два вузли намагаються одночасно передати кадр даних. Така ситуація призводить до «викривлення» інформації і називається **комунікаційною колізією**. У технології Ethernet передбачено механізм для вирішення таких ситуацій.

Головний **недолік мереж**, побудованих за технологією **Ethernet**, полягає в тому, що в них за одиницю часу може передаватися лише **один пакет** даних. Для забезпечення високої продуктивності таких мереж їх не слід завантажувати більш, ніж на 30–40 % від їх максимальної пропускної здатності.

Для побудови високошвидкісних комунікаційних мереж використовували, наприклад, технології

100 Gigabit Ethernet,
InfiniBand (100 Gbit/s, після 2023 р. – 250 Gbit/s),

Intel Omni-Path (100 Gbit/s, у 2019 р. відміна створення технології з продуктивністю 200 Gbit/s).

Важливою характеристикою комунікаційної мережі є її **топологія**, під якою розуміють спосіб з'єднання процесорних вузлів між собою каналами передачі даних. Топологію комунікаційної мережі прийнято зображати у вигляді графу, вершини якого відповідають процесорним вузлам, а ребра – каналам зв'язку.

Практично кожна паралельна обчислювальна система має свою **оригінальну топологію** комунікаційної мережі. Це пов'язано з тим, що неможливо з'єднати багато вузлів швидкими каналами зв'язку за типом «кожен з кожним». Тому в паралельних системах швидкий обмін інформацією здійснюється не між усіма вузлами, а лише між деякими. Всі інші обміни здійснюються відносно повільно.

Вказаний розподіл зв'язків на швидкі та повільні, по-перше, породжує **різноманіття топологій** комунікаційних мереж, а, по-друге, є однією із головних причин обмеження класу алгоритмів, які можуть бути ефективно реалізовані на заданій паралельній обчислювальній системі.

Швидкі обміни здійснюються по некомутованих лініях зв'язку, а **повільні** – або через мережеві комутатори, або з допомогою послідовних швидких обмінів.

Під час побудови LAN-мереж широко використовують також мережеві **концентратори** (concentrator, hub). Концентратор на один із своїх портів приймає сигнали від одного із вузлів мережі і синхронно передає їх на всі свої решту портів, з'єднаних з іншими вузлами мережі (див. рис. 3).

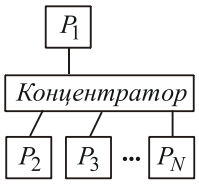


Рис. 3

4. Діаметром d комунікаційної мережі називається максимальна відстань між двома її довільними вузлами. **Відстань** між вузлами комунікаційної мережі – це кількість ребер, які утворюють найкоротший шлях між двома вузлами в графі мережі.

За топологією комунікаційні мережі **поділяються** на мережі з **фіксованою** топологією і на **реконфігуровні** мережі. З іншого боку ці мережі діляться на **регулярні і нерегулярні**. Регулярні мережі – це мережі, в яких кожен вузол (вершина) має однакову кількість зв'язків (ребер).

Найчастіше в паралельних системах використовують регулярні мережі з фіксованою топологією. Розглянемо найбільш розповсюджені топології мереж.

Комунікаційна мережа *з топологією типу* ***«шина»*** (див. рис. 4).

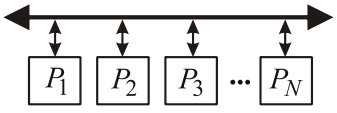


Рис. 4

Тут всі процесори з'єднані за допомогою (високошвидкісної) шини. Перевагою такої мережі є дуже мала кількість ліній зв'язку, але при цьому можливі затримки (конфлікти на шині) під час використання шини декількома процесорами. Це може стати серйозною проблемою під час збільшення кількості процесорів. **Діаметр** такої мережі $d = 1$. Тут і надалі N – це кількість процесорів у системі. Зазначимо, що **шину** можна інтерпретувати як **мережевий комутатор** з часовим розділенням.

Комунікаційна мережа *з топологією типу* ***«лінійка»*** (див. рис. 5).

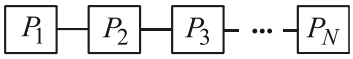


Рис. 5

У мережі з цією топологією кожен процесор зв'язаний з двома сусідами (за вийнятком кінцевих процесорів, які мають лише по одному з'єднанню). **Перевага** цієї топології – в її **простоті** (із кожного процесора виходить не більш як дві лінії зв'язку). З іншого боку, згадана топологія відповідає структурі передачі даних під час розв'язання багатьох обчислювальних задач. Серйозним **недоліком** є те, що дані, перш ніж досягти свого кінцевого призначення, можливо, повинні будуть пройти через декілька проміжних процесорів. **Діаметр** такої мережі $d = N - 1$.

Комунікаційна мережа *з топологією типу* ***«кільце» (коло)*** (див. рис. 6).

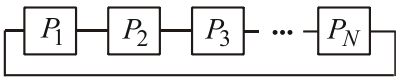


Рис. 6

Ця топологія одержується з топології типу «лінійка» унаслідок з'єднання між собою першого і останнього процесора, при цьому **діаметр** $d \approx N/2$.

Комунікаційна мережа *з топологією типу* ***«двовимірна решітка» (гратка)*** (див. рис. 7).

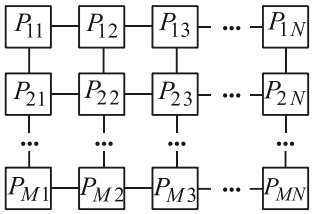


Рис. 7

Перевагою такої мережі є її **простота** реалізації, а **недолік** полягає в тому, що під час обміну між віддаленими процесорами дані повинні пройти через низку проміжних процесорів. **Діаметр** такої мережі $d = N + M - 2$. Із цієї топології легко одержати **топологію типу «тор»**. Для цього досить з'єднати між собою граничні процесори так: по «горизонталі» – $P_{11}, P_{1N}; P_{21}, P_{2N}; \ldots; P_{M1}, P_{MN}$ і по «вертикалі» – $P_{11}, P_{M1}; P_{12}, P_{M2}; \ldots; P_{1N}, P_{MN}$.

Комунікаційна мережа «повністю зв'язана» (див. рис. 8).
Розглянемо приклад такої мережі для $N = 6$.

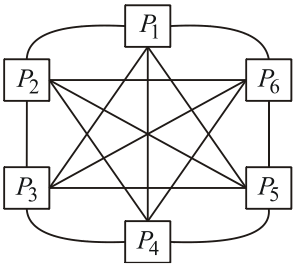


Рис. 8

У цій мережі між будь-якою парою процесорів існує безпосередній зв'язок. Повна зв'язаність мережі для *N* процесорів вимагає, щоб із кожного процесора виходили $N-1$ лінія зв'язку, що є **непрактичним** при великому *N*. Для цієї мережі $d = 1$.

Мережа *з топологією типу* ***«бінарне дерево»*** (див. рис. 9).

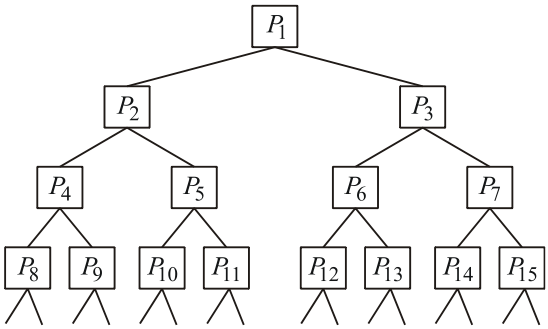


Рис. 9

Діаметр цієї мережі $d \approx 2(\log_2 N - 1)$.

Комунікаційна мережа *з топологією типу* ***«зірка»*** (див. рис. 10).

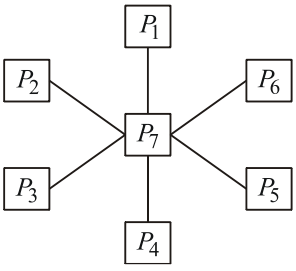


Рис. 10

Мережа з такою топологією є **ефективна**, наприклад, під час організації централізованих схем паралельних обчислень, при цьому $d = 2$.

Комунікаційна мережа *з топологією типу* ***«гіперкуб» (трьохвимірний гіперкуб)*** (див. рис. 11).

Діаметр такої мережі $d = \log_2 N$, $N = 2^p$, де $p = d$ – кількість вимірів гіперкуба. У цій мережі її діаметр **повільно** збільшується із збільшенням кількості процесорів.

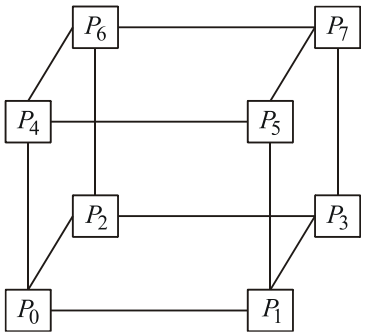


Рис. 11

Наприклад, для $p = 6;\ N = 2^6 = 64$ діаметр $d = 6$, а для $p = 10;\ N = 2^{10} = 1024$ – $d = 10$.

Зауважимо, що в p-вимірному гіперкубі кожен процесор безпосередньо зв'язаний рівно з p сусідами.

Відзначимо, що обчислювальні системи з **великою кількістю** процесорів зазвичай мають **ієрархічну** комунікаційну мережу. У такій мережі всі процесори розбиваються на групи з однаковою і відносно невеликою кількістю процесорів ($\approx 32 - 64$) в кожній. Фізично група процесорів монтується в **один блок**. Кожен з таких блоків має комунікаційну мережу однакової топології. Групи процесорів (блоки) об'єднуються між собою комунікаційною мережею такої ж або іншої топології.

Важливою **властивістю топології** комунікаційної мережі є можливість забезпечення **масштабованості** обчислювальної системи. З цього погляду **добре масштабуються** системи, що мають топологію мережі типу «лінійка», «кільце», «шина». Тут нарощування системи можливе по одному процесору. У мережі з топологією «повністю зв'язана» долучення одного процесора вимагає додавання N каналів зв'язку (N – кількість процесорів у вихідній мережі), що утруднює масштабування системи.

Нарощування кількості процесорів у обчислювальних системах з мережею топології «плоска $M \times N$ двовимірна решітка» або «тор» можливе лише квантами по M або N процесорів.

5. Мережеві комутатори поділяють на **прості і складні**. Перші потребують менше часу на комутацію, але можуть бути використані лише для побудови систем із малою кількістю вузлів. **Складні** комутатори будують на підставі **об'єднання простих** комутаторів.

Прості комутатори поділяють на комутатори з **часовим** розділенням і комутатори з **просторовим** розділенням.

Комутаторами з **часовим розділенням** є шини. Оскільки шина є розділюваним ресурсом, то процесорні вузли вимушені конкурувати за доступ до цього ресурсу. Для розв'язання **конфліктів** між вузлами використовують різні алгоритми **арбітражу**. Шина є доволі дешевим і надійним комутатором. Однак, під час комутації великої кількості коротких повідомлень накладні витрати на арбітраж стають значними, що суттєво обмежує пропускну її здатність.

Комутатор з **просторовим розділенням** ще називають координатним комутатором. Свою назву «**координатні комутатори**» прості комутатори одержали тому, що схему комутації з допомогою цього 3×3 комутатора можна подати так, як на рис. 12.

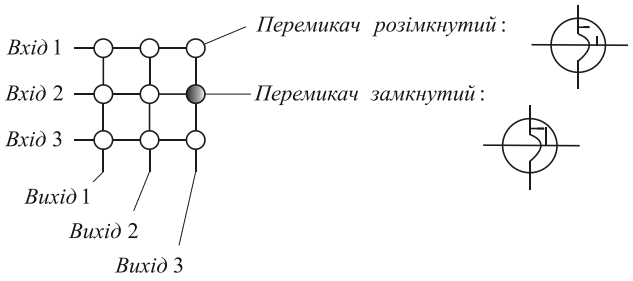


Рис. 12

Такі комутатори потребують мінімального часу на комутацію, але мають **високу складність**, а тому і **не** досить **високу надійність**. Кількість перемикачів, що дозволяють з'єднувати p процесорів з p пристроями пам'яті, в такому комутаторі дорівнює p^2, тому при великих p така схема з'єднань стає непрактичною.

Складні комутатори. Головна **ідея** створення складних комутаторів полягає в **об'єднанні простих** комутаторів каналами типу «пункт-пункт» («інстанція-інстанція»). Складний комутатор потребує менше обладнання, ніж простий комутатор з такою ж кількістю входів-виходів. Однак складному комутатору при цьому потрібно більше часу на комутацію, який росте пропорційно до кількості рівнів комутації.

Найчастіше складні комутатори будуються на основі 2×2 простих комутаторів з просторовим розділенням.

Як складний комутатор розглянемо **комутатор Клоза**. Цей комутатор складається із трьох рівнів комутації: вхідного, проміжного та вихідного. Комутатор Клоза має однакову кількість входів та виходів $n=m\times l$ і складається із:

- m вхідних $l\times l$ простих комутаторів;
- m таких же вихідних комутаторів;
- l проміжних $m\times m$ комутаторів.

Схема такого комутатора для $m=3; l=4$ зображена на рис. 13.

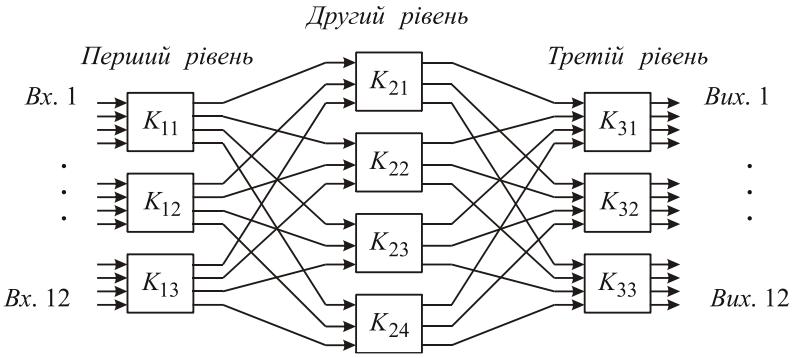


Рис. 13

Тут K_{ij} ($i=\overline{1,3}$; $j=\overline{1,3}$); K_{24} – прості комутатори. З'єднання простих комутаторів в комутаторі Клоза виконується за таким правилом (див. рис. 13):

- k-й вихід комутатора K_{1i} з'єднується з i-м входом комутатора K_{2k};
- k-й вхід комутатора K_{3j} з'єднується з j-м виходом комутатора K_{2k}.

Загальна кількість простих комутаторів у комутаторі Клоза дорівнює $2m+l$. Цей комутатор є **блокуючою** комунікаційною **мережею**. Вхід тут може одержати відмову від з'єднання із-за того, що який-небудь простий комутатор є зайнятий.

6. Продуктивність комунікаційної мережі визначається двома такими основними характеристиками: **латентністю** комунікаційної мережі та її **пропускною здатністю**. Латентність – це час підготовки до передавання інформації по каналу мережі. Пропускна здатність – це швидкість передачі інформації по каналу мережі або, більш строго, кількість інформації, що передається між вузлами мережі за одиницю часу.

Зазначимо одразу, що **наявність латентності** означає наступне: максимальна швидкість передачі по мережі не може бути досягнута на повідомленнях із невеликою довжиною. При цьому швидкості передачі даних на повідомленнях з великою і малою довжиною можуть відрізнятися на порядок. Звідси випливає важливий **практичний висновок**: під час розробки паралельних алгоритмів та їх програмної реалізації слід намагатися скоротити не лише загальний обсяг передаваних даних, але й кількість пересилань «коротких» даних.

Загалом для будь-якої комунікаційної технології справджується така **оцінка**: на реальних задачах швидкості передачі даних є на 20–40%, а інколи і на всі 100% гіршими, ніж максимально можливі.

Повний час, необхідний для передачі повідомлення довжиною L, визначається так:

$$T = S + L/R,$$

де S – латентність, L – довжина повідомлення (від декількох байт до мегабайт), а R – пропускна здатність каналу зв'язку.

Якщо паралельна обчислювальна система призначена для розв'язання задач, в яких є **великою частка комунікаційних операцій**, то «грубо» необхідну про-

пускну здатність каналів комунікаційної мережі можна визначити за таким **правилом**: «швидкість міжпроцесорних обмінів між двома вузлами, що вимірюється в Мбайт/с, повинна бути не меншою за 1/10 від пікової (теоретично максимально можливої) продуктивності обчислювального вузла, яка вимірюється в Mflops».