# ЛЕКЦІЯ 2
# ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ

## 1.1. Мотиви паралелізму

- Паралельність підвищує продуктивність системи через ефективнішу витрату системних ресурсів. Наприклад, під час очікування появи даних по мережі обчислювальна система може використовуватися для вирішення локальних завдань.
- Паралельність підвищує чутливість програми. Якщо один потік зайнятий розрахунком або виконанням якихось запитів, інший потік може реагувати на дії користувача.
- Паралельність полегшує реалізацію багатьох програм. Безліч додатків типу «клієнт-сервер», «виробник-споживач» мають внутрішній паралелізм. Послідовна реалізація таких додатків є більш трудомісткою, ніж опис функціональності кожного учасника окремо.

## 1.2. Класифікація обчислювальних систем

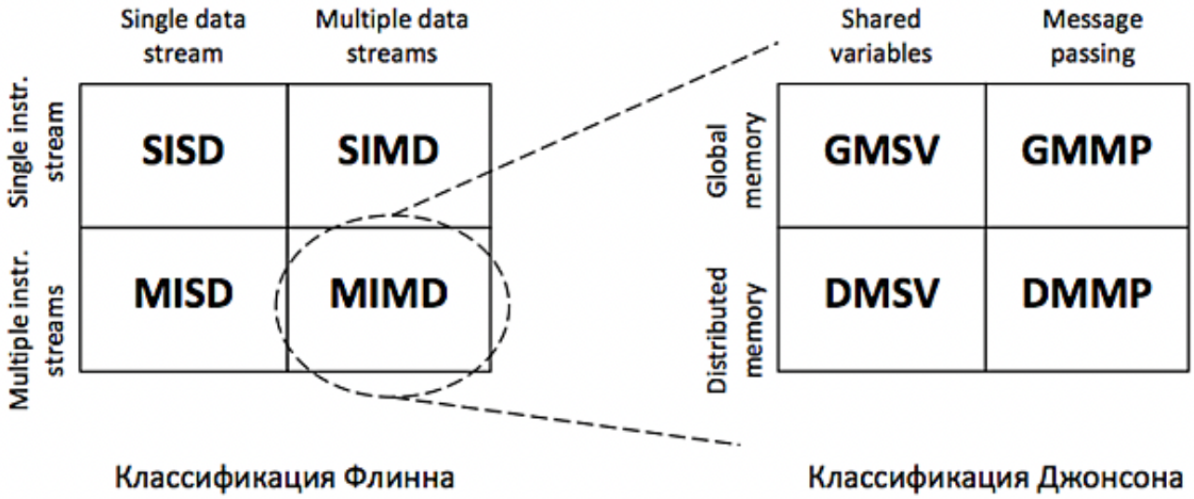
Однією з найпоширеніших класифікацій обчислювальних систем є класифікація Флінна. Чотири класи обчислювальних систем виділяються у відповідність із двома вимірами – характеристиками систем: потік команд, які дана архітектура здатна виконати в одиницю часу (одиночний чи множинний) та потік даних, які можуть бути оброблені в одиницю часу (одиночний чи множинний).

- SISD (Single Instruction, Single Data) – системи, в яких існує одиночний потік команд та одиночний потік даних. Кожного часу процесор обробляє одиночний потік команд над одиночним потоком даних. До цього типу систем належать послідовні персональні комп’ютери з одноядерними процесорами.
- SIMD (Single Instruction, Multiple Data) – системи з одиночним потоком команд та з множинним потоком даних; подібний клас складають

багатопроцесорні системи, в яких у кожний момент часу може виконуватися та сама команда для обробки кількох інформаційних елементів. Така архітектура дозволяє виконувати одну арифметичну операцію елементів вектора. Сучасні комп'ютери реалізують деякі команди типу SIMD (векторні команди), що дозволяють обробляти декілька елементів даних за один такт.

- MISD (Multiple Instructions, Single Data) – системи, у яких існує множинний потік команд та одиночний потік даних; до цього класу відносять систолічні обчислювальні системи та конвеєрні системи;
- MIMD (Multiple Instructions, Multiple Data) – системи з множинним потоком команд та множинним потоком даних; до цього класу належить більшість паралельних обчислювальних систем.

Класифікація Флінна відносить майже всі паралельні обчислювальні системи одного класу – MIMD. Для виділення різних типів паралельних обчислювальних систем застосовується класифікація Джонсона, у якій подальший поділ багатопроцесорних систем ґрунтується на використовуваних методах організації оперативної пам'яті в цих системах. Даний підхід дозволяє розрізняти два важливі типи багатопроцесорних систем: multiprocessors (мультипроцесорні або системи із загальною пам'яттю, що розділяється) і multicomputers (мультикомп'ютери або системи з розподіленою пам'яттю).




Классификация Флинна Классификация Джонсона

Класифікація Джонсоном заснована на структурі пам'яті (global - глобальна або distributed - розподілена) і механізмі комунікацій та

синхронізації (shared variables - змінні, що розділяються, або message passing - передача повідомлень). Системи GMSV (global-memory-shared-variables) часто називаються також мультипроцесорами з пам'яттю, що розділяється (shared-memory multiprocessors). Системи DMMP (distributed-memory-message-passing) також називають мультикомп'ютерами з розподіленою пам'яттю (distributed-memory multicomputers).

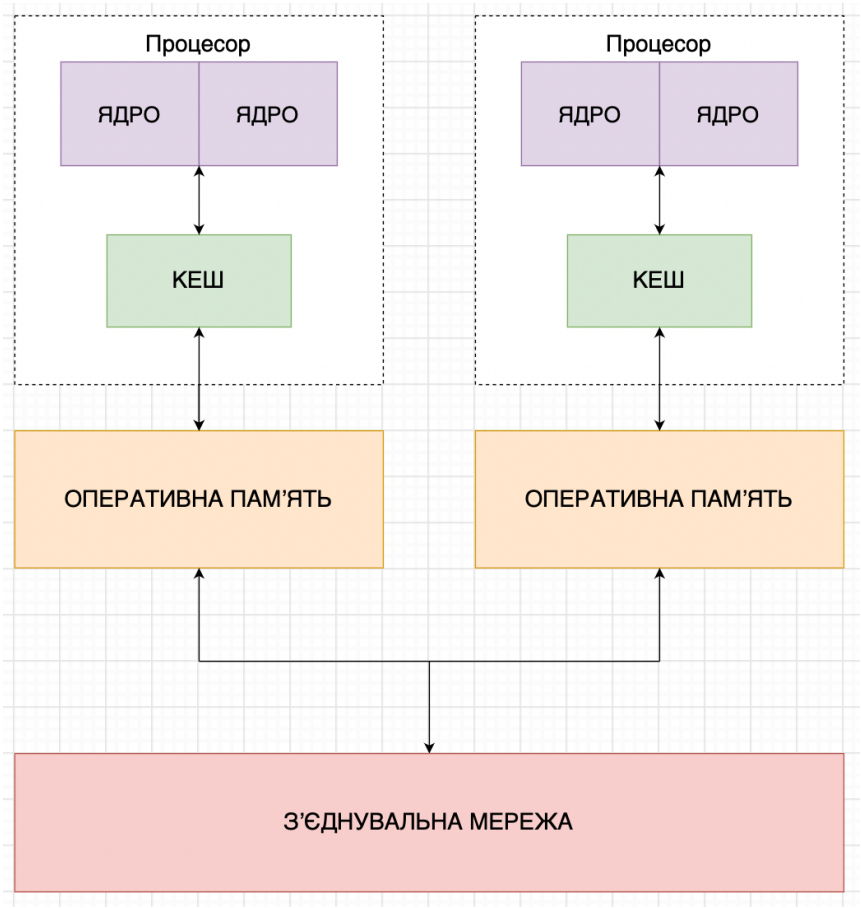
### 1.3. Архітектура однопроцесорної машини *(трішечки улюбленого АОС ☺)*

Сучасна однопроцесорна машина складається з кількох компонентів: центрального процесорного пристрою (ЦПУ), первинної пам'яті, одного або кількох рівнів кеш-пам'яті (кеш), вторинної (дискової) пам'яті та набору периферійних пристроїв (дисплей, клавіатура, миша, модем, принтер і т.д.). Основними компонентами для виконання програм є ЦПУ, кеш та пам'ять.


Процесор вибирає інструкції з пам'яті, декодує їх та виконує. Він містить керуючий пристрій (УУ), арифметико-логічний пристрій (АЛУ) та регістри. УУ виробляє сигнали, що керують діями АЛУ, системою пам'яті та зовнішніми пристроями. АЛУ виконує арифметичні та логічні інструкції, що визначаються набором інструкцій процесора. У регістрах зберігаються інструкції, дані та стан машини (включаючи лічильник команд).

## 1.4. Мультикомп’ютери з розподіленою пам’яттю

У мультикомп’ютерах із розподіленою пам’яттю існує сполучна мережу, але кожен процесор має власну пам’ять. Сполучна мережа підтримує надсилання повідомлень. Мультикомп’ютери (багатопроцесорні системи з розподіленою пам’яттю) не забезпечують загальний доступ до всієї наявної в системах пам’яті. Кожен процесор системи може використовувати лише свою локальну пам’ять, у той час як для доступу до даних, які розміщуються на інших процесорах, необхідно використовувати інтерфейси передачі повідомлень (наприклад, стандарт MPI). Цей підхід використовується при побудові двох важливих типів багатопроцесорних обчислювальних систем - масивно-паралельних систем (massively parallel processor or MPP) та кластерів (clusters).




Мультикомп’ютер (багатомашинна система) – мультипроцесор з розподіленою пам’яттю, в якому процесори та мережа розташовані фізично близько (в одному приміщенні). Також називають тісно пов’язаною

машинною. Вона одночасно використовується однією або невеликою кількістю додатків; кожен додаток задіює виділений набір процесорів. Сполучна мережа з великою пропускною здатністю надає високошвидкісний шлях зв'язку між процесорами.

Мережева система – це багатомашинна система з розподіленою пам'яттю, пов'язані з допомогою локальної мережі чи глобальної мережі Internet (слабко пов'язані мультикомп'ютери). Тут процесори взаємодіють також за допомогою передачі повідомлень, але час доставки більше, ніж у багатомашинних системах, й у мережі більше конфліктів.

З іншого боку, мережева система будується на основі звичайних робочих станцій та мереж, тоді як у багатомашинній системі часто є спеціалізовані компоненти, особливо у сполучній мережі.

Під кластером зазвичай розуміється безліч окремих комп'ютерів, об'єднаних у мережу, котрим за допомогою спеціальних апаратно-програмних засобів забезпечується можливість уніфікованого керування, надійного функціонування та ефективного використання.

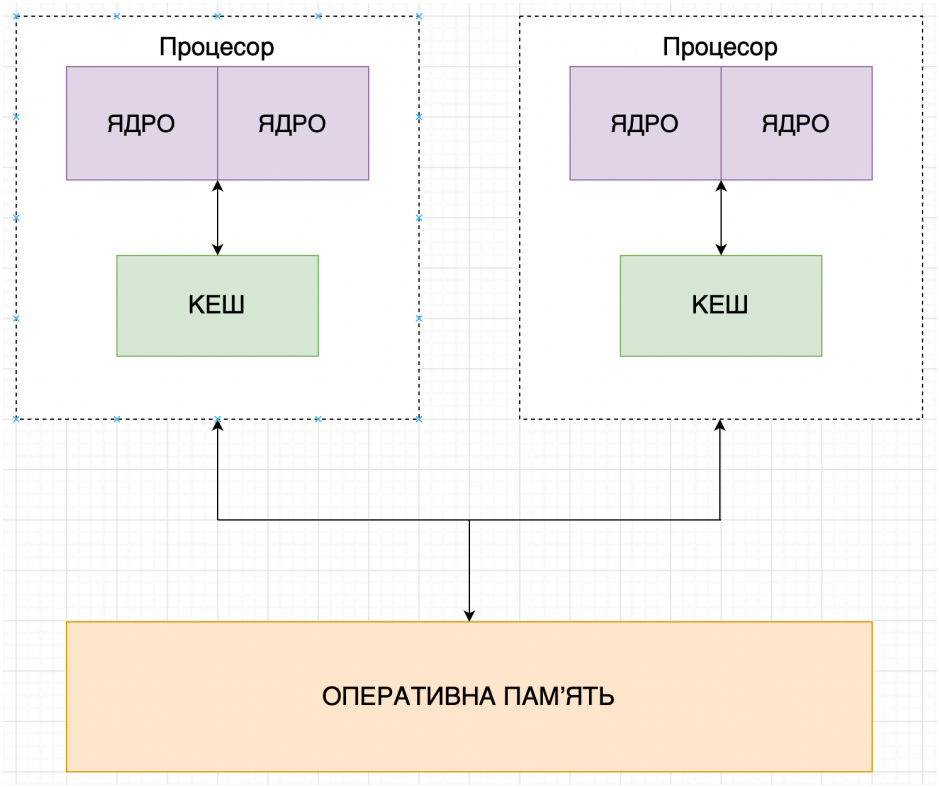
Кластери можуть бути утворені на базі вже існуючих у споживачів окремих комп'ютерів, або сконструйовані з типових комп'ютерних елементів, що зазвичай не вимагає значних фінансових витрат. Застосування кластерів може також певною мірою знизити проблеми, пов'язані з розробкою паралельних алгоритмів та програм, оскільки підвищення обчислювальної потужності окремих процесорів дозволяє будувати кластери з порівняно невеликої кількості (кілька десятків) окремих комп'ютерів (lowly parallel processing). Це призводить до того, що для паралельного виконання в алгоритмах вирішення обчислювальних завдань достатньо виділяти лише великі незалежні частини розрахунків (coarse granularity), що, у свою чергу, знижує складність побудови паралельних методів обчислень і зменшує потоки даних між комп'ютерами кластера.

Разом з цим слід зазначити, що організація взаємодії обчислювальних вузлів кластера за допомогою передачі повідомлень зазвичай призводить до

значних тимчасових затримок, що накладає додаткові обмеження на тип паралельних алгоритмів і програм, що розробляються.

## 1.5. Мультипроцесор з пам’яттю, що розділяється

У мультипроцесорі і в багатоядерній системі виконавчі пристрої (процесори і ядра процесорів) мають доступ до оперативної пам’яті, що розділяється. Процесори спільно використовують оперативну пам’ять.




Кожен процесор має власний кеш. Якщо два процесори посилаються на різні області пам’яті, їх вміст можна безпечно помістити в кеш кожного з них. Проблема виникає, коли два процесори звертаються до однієї області пам’яті. Якщо обидва процесори лише зчитують дані, у кеш кожного можна помістити копію даних. Але якщо один із процесорів записує на згадку, виникає проблема узгодженості кешу: у кеш-пам’яті іншого процесора тепер містяться невірні дані. Необхідно або оновити кеш іншого процесора або визнати вміст кешу недійсним. Забезпечення однозначності кешів реалізується на апаратному рівні - для цього після зміни значення загальної змінної всі копії цієї змінної в кешах відзначаються як недійсні і доступ до змінної вимагатиме обов’язкового звернення до основної пам’яті. Необхідність забезпечення

когерентності призводить до деякого зниження швидкості обчислень та ускладнює створення систем із досить великою кількістю процесорів.

Наявність загальних даних при виконанні паралельних обчислень призводить до необхідності синхронізації взаємодії потоків команд, що одночасно виконуються. Так, наприклад, якщо зміна загальних даних вимагає для свого виконання деякої послідовності дій, необхідно забезпечити взаємовиключення з тим, щоб ці зміни в будь-який момент часу міг виконувати тільки один командний потік. Завдання взаємовиключення і синхронізації відносяться до класичних проблем, і їх розгляд при розробці паралельних програм є одним з основних питань паралельного програмування.

## 1.6. Режими виконання незалежних частин програми

Під час розгляду проблеми організації паралельних обчислень слід розрізняти такі можливі режими виконання незалежних частин програми:

***1) Режим поділу часу (багатозадачний режим)***

Режим поділу часу передбачає, що число підзавдань (процесів або потоків одного процесу) більше, ніж кількість виконавчих пристроїв. Цей режим є псевдопараллельным, коли активним (виконуваним) може бути єдина підзавдання, проте інші процеси (потоки) перебувають у стані очікування своєї черги використання процесора; використання режиму поділу часу може підвищити ефективність організації обчислень (наприклад, якщо один з процесів не може виконуватися через очікування даних, процесор може бути задіяний для виконання іншого, готового до виконання процесу), крім того в даному режимі проявляються багато ефектів паралельних обчислень (Необхідність взаємовиключення та синхронізації процесів та ін.).

Багатопотоковість додатків в операційних системах з поділом часу застосовується навіть в однопроцесорних системах. Наприклад, у Windows-додатках багатопотоковість підвищує чуйність програми - якщо основний потік зайнятий виконанням якихось розрахунків або запитів, інший потік

дозволяє реагувати на дії користувача. Багатопотоковість спрощує розробку програми. Кожен потік може плануватися та виконуватися незалежно. Наприклад, коли користувач натискає кнопку мишки персонального комп'ютера, надсилається сигнал процесу, що управляє вікном, в якому зараз знаходиться курсор миші. Цей процес може виконуватися і відповідати на клацання миші. Програми в інших вікнах можуть продовжувати своє виконання у фоновому режимі.

***2) Розподілені обчислення***

Компоненти виконуються на машинах, пов'язаних з локальною або глобальною мережею. Тому процеси взаємодіють, обмінюючись повідомленнями.

Такі системи пишуться для розподілу обробки (як у файлових серверах), забезпечення доступу до віддалених даних (як у базах даних та в Web), інтеграції та управління даними, розподіленими за своєю суттю (як у промислових системах), або підвищення надійності (як у відмовостійких системах). Багато розподілених систем організовано як системи типу клієнт-сервер. Наприклад, файловий сервер надає файли даних для процесів, які виконуються на клієнтських машинах. Компоненти розподілених систем часто є багатопотоковими.

***3) Синхронні паралельні обчислення.***

Їхня мета – швидко вирішувати це завдання або за той же час вирішити велике завдання. Приклади синхронних обчислень:

• наукові обчислення, які моделюють та імітують такі явища, як глобальний клімат, еволюція сонячної системи чи результат дії нових ліків;

• графіка та обробка зображень, включаючи створення спецефектів у кіно;

• великі комбінаторні чи оптимізаційні завдання, наприклад, планування авіаперельотів чи економічне моделювання.

Програми вирішення таких завдань потребують ефективного використання доступних обчислювальних ресурсів системи. Число підзавдань

має бути оптимізовано з урахуванням числа виконавчих пристроїв у системі (процесорів, ядер процесорів).

## 1.7. Рівні паралелізму у багатоядерних архітектурах

Паралелізм на рівні команд (InstructionLevelParallelism, ILP) дозволяє процесору виконувати кілька команд за один такт. Залежності між командами обмежують кількість доступних виконання команд, знижуючи обсяг паралельних обчислень. Технологія ILP дозволяє процесору впорядкувати команди оптимальним чином з метою унеможливити зупинення обчислювального конвеєра і збільшити кількість команд, що виконуються процесором за один такт. Сучасні процесори підтримують певний набір команд, які можуть виконуватись паралельно.

Паралелізм лише на рівні потоків процесу. Потоки дозволяють виділити незалежні потоки виконання команд у межах одного процесу. Потоки підтримуються лише на рівні операційної системи. Операційна система розподіляє потоки процесів по ядрам процесора з урахуванням пріоритетів. За допомогою потоків програма може максимально задіяти вільні обчислювальні ресурси.

Паралелізм на рівні додатків. Одночасне виконання кількох програм здійснюється у всіх операційних системах, що підтримують режим розподілу часу. Навіть на однопроцесорній системі незалежні програми виконуються одночасно. Паралельність досягається з допомогою виділення кожному додатку кванта процесорного часу.

## 1.8. Аналіз ефективності паралельних обчислень

Нехай, обчислення виконуються на комп’ютері з числом процесорів, що дорівняє $p$. Таким чином за допомогою $T_p$ позначимо проміжок часу між

початком обчислень та кінецем. Аналіз часу затраченого на виконання обчислень базується на наступних поняттях:

- *Робота* – це кількістний показник елементарних операцій, що були виконанні кількістю процесорів $p$ для виконання обчислень. Позначається як $T_1$.
- *Проліт* – довжина найдовшої послідовності операцій, які повинні виконуватися послідовно. Проліт часто також називають критичною довжиною шляху або глибину розрахунку. Дуже важливо звести до мінімуму величину прольоту під час розробки паралельних алгоритмів, оскільки саме від величини прольоту залежить можливий максимально короткий термін виконання обчислень. Проліт також може позначатися як час $T_\infty$, що був затрачений для обчислень, що були виконані за умови використанням ідеалізованої машини з нескінченною кількістю процесорів.
- *Вартість обчисення* – показник загального часу витраченого процесорними одиницями для виконання обчислень.

З наведених вище понять та їх визначень випливає наступне:

- *Закон Роботи* – вартість обчислень завжди буде менша за роботу:

Це пов'язано з тим фактом, що певна кількість процесорів $p$ може виконувати не більше $n$ операцій паралельно.

- *Закону Прольоту* – кількість процесорів завжди буде скінченним числом, а отже

Опираючись на вищеподані визначення і закони можна виділити наступні атрибути:

- *Прискорення* – це збільшення швидкості паралельних обчислень в порівнянні з послідовними: $S_p = \frac{T_1}{T_p}$.

  Якщо прискорення $\Omega(n)$ для вхідного масиву даних з розміром $n$ $\epsilon$ лінійним, то випливає, що $\frac{T_1}{T_p} \leq p$. Ситуація коли $\frac{T_1}{T_p} = p$ називається

досконалим лінійним прискоренням. Алгоритм, який показує лінійне прискорення вважається так званим масштабованим алгоритмом.

- *Ефективність* – величина прискорення, що припадає на один процесор: $\frac{S_p}{p}$.
- *Паралельність* – відношення $\frac{T_1}{T_\infty}$. Являє собою максимально можливе прискорення на будь-якій кількості процесорів. Згідно закону прольоту: $p > \frac{T_1}{T_\infty}$, тоді $\frac{T_1}{T_p} \leq \frac{T_1}{T_\infty} < p$

Аналіз ефективності паралельних обчислень

$$S_p = \frac{T_1}{T_p}$$

Ефективність паралельного алгоритму визначається так:

$$E_p = \frac{T_1}{pT_p} = \frac{S_p}{p}$$

Ефективність показує, наскільки задіяно обчислювальні ресурси системи; ідеальне теоретичне значення ефективності дорівнює одиниці.

## 1.9. Межі паралелізму

Досягнення максимального прискорення може перешкоджати існування у обчисленнях послідовних розрахунків, які не можуть бути розпаралелені. Джин Амдал (Gene Amdahl) показав, що верхня межа для прискорення визначається часткою послідовних обчислень алгоритму:

$$S_p \leq \frac{1}{f + \frac{(1-f)}{p}} \leq S^* = \frac{1}{f}$$

$f$ – доля послідовних обчислень в використовуваному алгоритмі обробки даних, $p$ – число процесорів.

Наприклад, якщо частка послідовних обчислень становить 25%, то максимально досяжне прискорення для паралельного алгоритму дорівнює:

$$S^* = \frac{1}{f} = \frac{1}{0{,}25} = 4$$

У «Законі Амдала» є кілька припущень, які у реальних додатках може бути неправильними. Одне з припущень у тому, частка послідовних розрахунків є постійної величиною і залежить від обчислювальної складності розв'язуваного завдання. Однак, для більшості задач f є спадною функцією від n, де n –параметр складності задачі. У цьому випадку прискорення може бути збільшено зі збільшенням обчислювальної складності завдання.

Порушення «Закону Амдала» також може бути пов'язані з архітектурними особливостями паралельної обчислювальної системи. Наприклад, паралельний алгоритм зменшує обсяг даних, використовуваних кожним процесором, та підвищує ефективність використання кеш-пам'яті кожного процесора. Оптимальна робота з кеш-пам'яттю може сильно збільшити швидкодію алгоритму.

Паралельний алгоритм називається масштабованим, якщо при зростанні числа процесорів він забезпечує збільшення прискорення за збереження постійного рівня ефективності використання процесорів.