Циганок Віталій Володимирович,

д.т.н., с.н.с.

«Методи та системи паралельного програмування »

галузь знань
"Інформатика та обчислювальна техніка“
напрям підготовки **"Комп'ютерні науки“**
освітньо-кваліфікаційного рівня «**бакалавр**»

Література

Основна:

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.
2. Эхтер Ш., Робертс Д. Многоядерное программирование. – СПб.: Питер, 2010. – 316 с.
3. Rauber T., G. Runger, Parallel Programming: for Multicore and Cluster Systems. Springer, 2010, 450 P.
4. Mattson T., Sanders B., Massingil Berna L. Patterns for Parallel Programming. Addison-Wesley, 2004, 355 P.
5. Дорошенко А.Е. Математические модели и методы организации высоко-производительных параллельных вычислений. - К.,"Наукова думка", 2000.- 177 с.
6. Корнеев В. Параллельные вычислительные системы.−М. Ноулидж. − 1999.

Література

Додаткова:

1. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие. – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. – 184 с.
2. Эндрюс, Грегори Р. Основы многопоточного, параллельного и распределенного программирования.: Пер. с англ. – М.: Издательский дом "Вильямс", 2003. – 505 с.
3. Дорошенко А.Ю. Лекції з паралельних обчислювальних систем. Київ: Видавничий дім “КМ Академія”, 2003. – 42 с.
4. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с.
5. Introduction to OpenMP (http://ci-tutor.ncsa.uiuc.edu/login.php).
6. Introduction to MPI (http://ci-tutor.ncsa.uiuc.edu/login.php).
7. MPI Forum (http://www.mpi-forum.org/)
8. http://parallel.ru/

Лекція №2: **Основи паралельних обчислювальних процесів**

1. Вимоги до паралельних програм.
2. Парадигми паралельного програмування.
    2.1. Паралелізм даних.
    2.2. Паралелізм завдань.
3. Моделі паралельного програмування
    3.1. Завдання/канал.
    3.2. Модель спільної пам'яті.
    3.3. Передача повідомлень
4. Продуктивність паралельних обчислень.
    4.1. Закон Амдала.
    4.2. Особливості закону Амдала.
    4.3. Закон Густавсона.

Паралельні обчислення

— спосіб організації комп'ютерних обчислень, при якому програми розробляються як набір взаємодіючих обчислювальних процесів, що працюють паралельно (одночасно).

Основна складність при проектуванні паралельних програм — забезпечити *правильну послідовність взаємодій між* різними обчислювальними ***процесами***, а також *координацію* ***ресурсів***, що розділяються між процесами.

Паралельна програма

– це множина взаємодіючих паралельних процесів.

Основною метою паралельних обчислень є прискорення вирішення обчислювальних завдань.

Особливості паралельних програм :

1) здійснюється *управління* роботою множини процесів;
2) організовується *обмін даними* між процесами;
3) втрачається *детермінізм* поведінки через асинхронність доступу до даних;
4) переважають нелокальні і динамічні помилки;
5) з'являється можливість *тупикових ситуацій*;
6) виникають проблеми *масштабованост*і програми і балансування *завантаження* обчислювальних вузлів.

Основні риси моделі паралельного програмування

- більш висока продуктивність програм,
- застосування спеціальних прийомів програмування та, як наслідок,
- більш висока трудомісткість програмування,
- проблеми з перенесенням програм.

- немає властивості унікальності.
- з'являються проблеми, незвичні для програміста, який звик займатися послідовним програмуванням:
  - керування роботою множини процесорів,
  - організація міжпроцесорних пересилок даних і т.п.

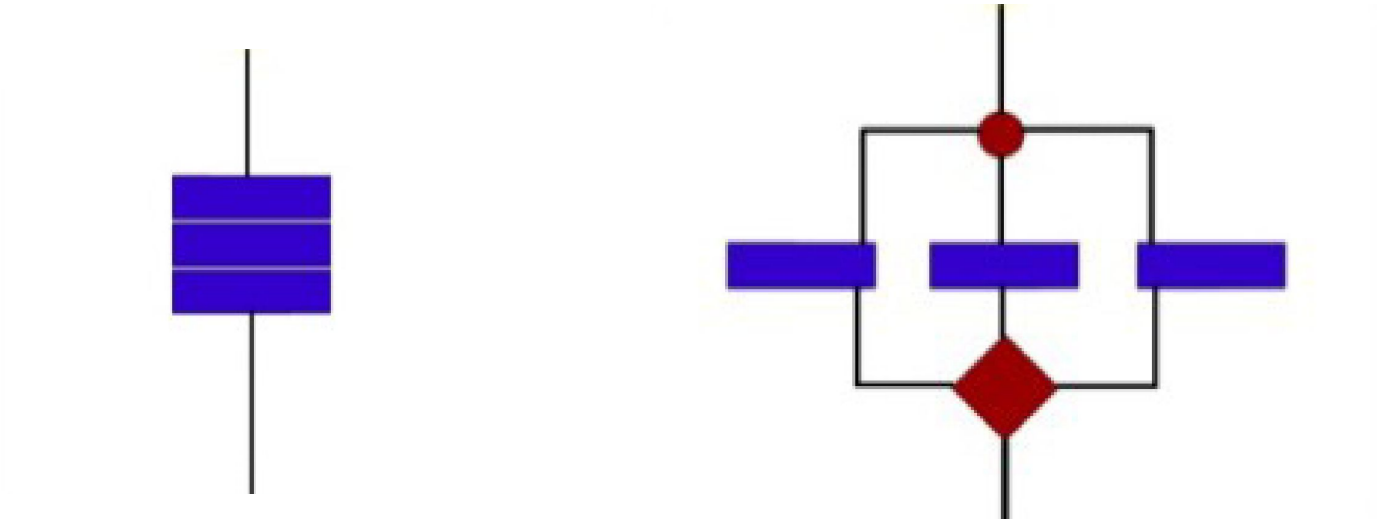
Вимоги до паралельних програм

1. паралелізм,
2. універсальність,
3. локальність,
4. модульність.

Паралелізм

Ідея розпаралелювання обчислень заснована на тому, що більшість завдань можуть бути розділені на набір менших завдань, які можуть бути вирішені одночасно.

Паралельна програма використовує множинність обчислювальних одиниць (процесорів, ядер, GPU і т. д.) для прискорення обчислень. Ми поділяємо роботу між одночасно працюючими обчислювачами в надії, що виграш від паралелізму перевершить додаткові витрати на нього. Прискорення обчислення — єдина мета паралелізму.



Універсальність

-- початкова орієнтація та налаштування на широкий клас задач предметної галузі, в якій працює прикладний фахівець, є однією з ключових особливостей ППП (пакетів прикладних програм).

Локальність

полягає в тому, що коли процес виконується, він рухається від однієї *локальності* до іншої.

Локальність - набір сторінок, які активно використовуються разом.

Програма складається з декількох різних *локальностей*, які можуть перекриватися. Наприклад, коли викликана процедура, вона визначає нову *локальність*, що складається з інструкцій процедури, її *локальних* змінних, і множини глобальних змінних. Після її завершення процес залишає цю *локальність*, але може повернутися до неї знову. Таким чином, *локальність* визначається кодом і даними програми.

Зауважимо, що модель локальності - принцип, покладений в основу роботи **кешу**. Якби доступ до будь-яких типів даних був випадковим, **кеш був би даремним**.

Модульність

Поєднання в одному пакеті різноманітних моделей і алгоритмів досягається шляхом використання принципу *модульної* організації функціонального наповнення пакету.

Модуль, який входить до складу ППП, видається, як правило, у вигляді автономної програмної одиниці, написаної традиційному мовою програмування, що забезпечує рішення деякої підзадачі.

Передбачається, що взаємодія *модулів* здійснюється тільки на рівні їх вхідних і вихідних параметрів.

Використання принципу *модульності* дозволяє замінити написання програми (в традиційному розумінні) її конструюванням з готових програмних блоків великого розміру.

Розширення класу задач, розв'язуваних пакетом, може досягатися за рахунок підключення до ППП новостворюваних *модулів*.

Масштабованість паралельних програм

Масштабованість – властивість програми, що визначає залежність її прискорення або ефективності, одержуваних на обчислювальній системі, від обсягу використаних для цього ресурсів і розміру задачі.

Масштабованість є однією з основних характеристик паралельної програми. Вона дозволяє отримати уявлення про роботу паралельної програми на даній обчислювальній системі.

Ідеальна паралельна програма має наступні властивості:

- паралельно виконувані гілки повинні мати приблизно однакову довжину;
- повністю виключені простої через
  - очікування даних,
  - передачі управління і
  - виникнення конфліктів при використанні загальних ресурсів;
- обмін даними повністю сумісний з обчисленнями.

Способи збільшення ступеня ефективності паралелізму (зменшення часових затрат на накладні витрати):

- укрупнення одиниць розпаралелювання;
- зменшення складності алгоритмів генерації паралельних процедур (підпрограм);
- початкова підготовка пакета різних варіантів вихідних даних;
- розпаралелювання алгоритмів генерації паралельних процедур (підпрограм).

Парадигми паралельного програмування

- **Паралелізм даних** — застосування однієї операції відразу до декількох елементів масиву даних.
- **Паралелізм завдань** — передбачає розбиття обчислювальної задачі на кілька відносно самостійних підзадач. Кожне завдання виконується на своєму процесорі. Комп'ютер при цьому являє собою MIMD-машину.

Основні особливості **паралелізму даних**

- паралельною обробкою даних управляє одна програма;
- простір імен є глобальним, тобто для програміста існує одна єдина пам'ять, а деталі структури даних, доступу до пам'яті і міжпроцесорного обміну даними від нього приховані;
- слабка синхронізація паралельних обчислень на процесорах, тобто виконання команд на різних процесорах відбувається, як правило, незалежно і тільки іноді проводиться узгодження виконання циклів або інших програмних конструкцій — їх синхронізація. Кожен процесор виконує один і той же фрагмент програми, але немає гарантії, що в заданий момент часу на всіх процесорах виконується одна і та ж машинна команда;
- паралельні операції над елементами масиву виконуються одночасно на всіх доступних даній програмі процесорах.

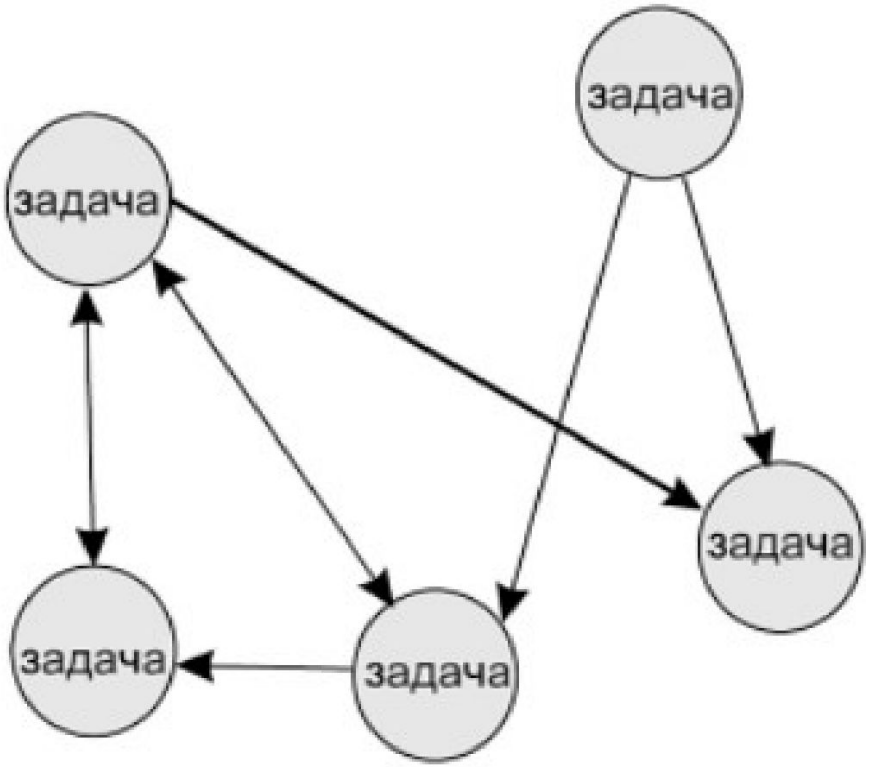
Проблеми, пов'язані з трудомісткістю **паралелізму завдань:**

- підвищена трудомісткість як розробки програми, так і її налагодження;
- на програміста лягає вся відповідальність за рівномірне і збалансоване завантаження процесорів паралельного комп'ютера;
- програмісту доводиться мінімізувати обмін даними між завданнями, оскільки витрати часу на пересилку даних зазвичай відносно великі;
- підвищена небезпека виникнення тупикових ситуацій, коли надіслана однією програмою посилка з даними не приходить до місця призначення.

Моделі паралельного програмування.

- **Модель задача/канал -** найпростіша модель паралельного програмування, програма складається з декількох завдань(задач), які виконуються одночасно і пов'язані між собою каналами комунікації.
- **Модель спільної пам'яті -** завдання звертаються до загальної пам'яті, маючи спільний адресний простір і виконуючи операції зчитування/запису.
- **Модель передачі повідомлень –** взаємодія між завданнями здійснюється за допомогою відправки і прийому повідомлень.

Модель задача/канал.



Модель спільної пам'яті

Модель передачі повідомлень

Продуктивність паралельних обчислень

Процесорний час, необхідний для виконання програми, можна визначити за формулою:

$$T=Ni*CPI*t,$$

де Ni — кількість машинних команд у програмі, а t — тривалість такту.

Швидкодія процесора вимірюється в MIPS (Million Instructions Per Second). MIPS зворотно пропорційна CPI (Cycles per Instruction).

Нехай час виконання алгоритму на послідовній машині T1, причому Ts — час виконання послідовної частини алгоритму, а Tp — паралельної. Очевидно:

$$T1 = Ts + Tp.$$

При виконанні тієї ж програми на ідеальній паралельній машині, N незалежних гілок паралельної частини розподіляються по одній на N процесорів, тому час виконання цієї частини зменшується до величини Tp / N, а повний час виконання програми складе:

$$T2=Ts+ Tp/N.$$

Коефіцієнт прискорення, який показує, у скільки разів швидше програма виконується на паралельній машині, ніж на послідовній, визначається формулою:

$$X= T1/T2 = (Ts + Tp) / (Ts + Tp/N) = 1/(S + P/N),$$

де S= Ts / (Ts + Tp) і P= Tp / (Ts + Tp) — відносні частки послідовної і паралельної частин (S+ P= 1).

Модель передачі повідомлень

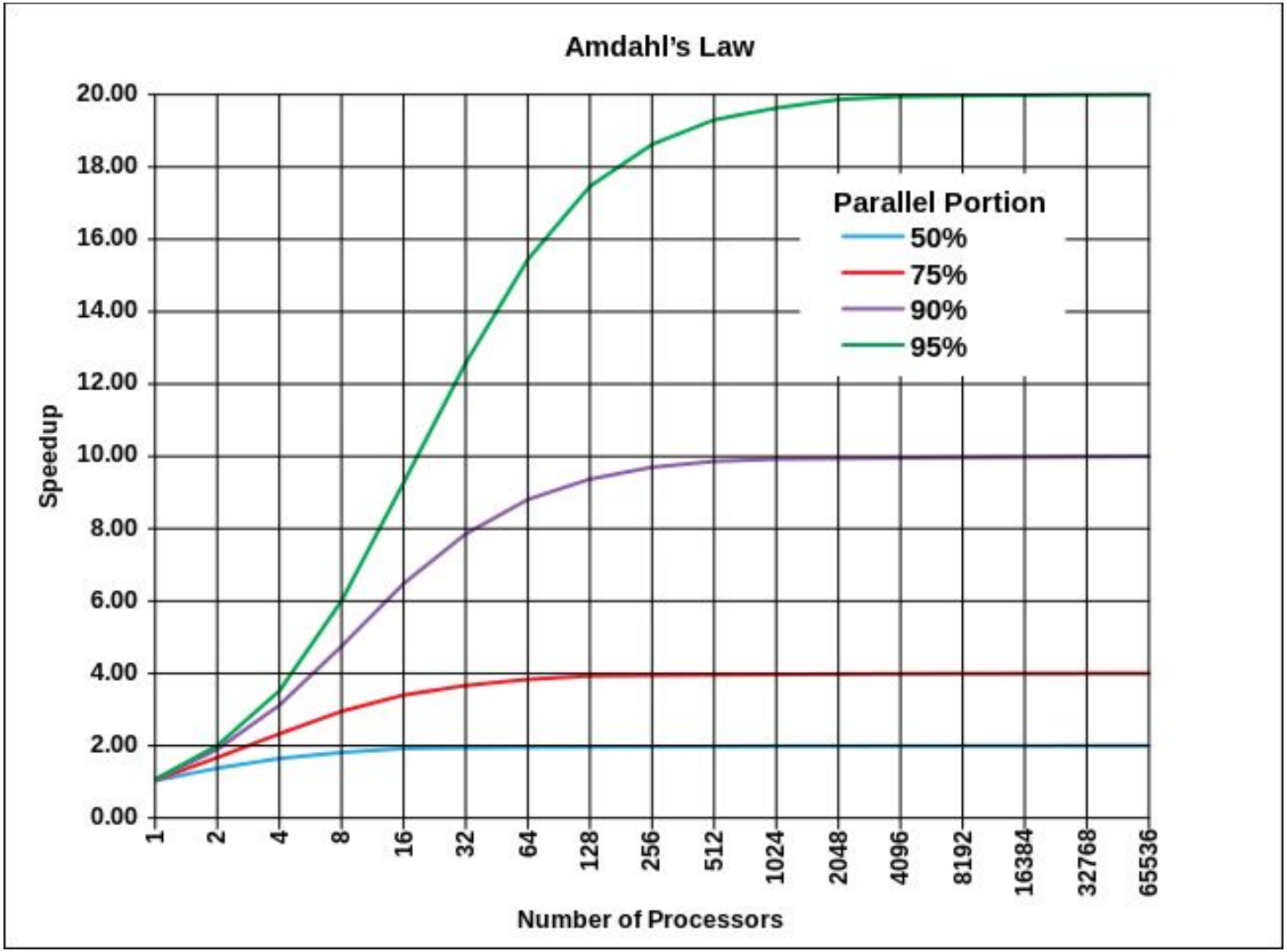
Закон Амдала

Джин Амдал сформулював закон в 1967 році, виявивши просте по суті, але непереборне за змістом обмеження на зростання продуктивності при розпаралелюванні обчислень: «У випадку, коли завдання поділяється на кілька частин, сумарний час його виконання на паралельній системі не може бути менше часу виконання самого довгого фрагмента».
Закон Амдала описує максимальний теоретичний виграш в продуктивності паралельного рішення по відношенню до кращого послідовного вирішення. Закон описується наступною математичною формулою:

$$S_n = \frac{1}{\alpha + \frac{1-\alpha}{n}}$$

Де S_n у скільки разів можна прискорити обчислення (прискорення), n - кількість процесорів (ядер), α - частка послідовно обчислюваного коду ($\alpha \neq 0$).

Ілюстрація закону Амдала. Прискорення програми з допомогою паралельних обчислень на декількох процесорах



Закон Густавсона

Закон Густавсона - Барсиса (1988р) оцінює максимально допустиме прискорення виконання паралельної програми, в залежності від кількості одночасно виконуваних потоків обчислень і частки послідовних розрахунків. Формула Густавсона – Барсиса виглядає наступним чином:

$$S_n = n + (1 - n)\alpha$$

Де α - частка послідовних розрахунків в програмі, n- кількість процесорів.