матеріальних витрат і трудомісткості, довгострокові вкладення в дані засоби окупляться.

Різноманітні модифікації RAD в основному розраховані на швидку розробку прототипів, а не на зручність експлуатації, безвідмовність та продуктивність. Вони різняться ступенем залучених користувачів у сам процес утворення прототипів та по використанню інструментальних засобів. На практиці ж ці методи дуже часто застосовуються разом для розробки прототипів систем. Тож відзначимо, що методи RAD, як будь-які інші, неспроможні вважатись універсальними, вони хороші в, першу чергу, стосовно невеликих проектів, що розроблюються для конкретних замовників.

Список використаних джерел та літератури

1. Моделі і методи проектування інформаційних систем. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/233994/index.html.
2. Методологія RAD розробки інформаційних систем. URL: https://ua-referat.com/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_RAD_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.
3. Методологія RAD. URL: http://um.co.ua/7/7-5/7-5898.html.
4. Методологія RAD - Rapid Application Development. URL: http://ni.biz.ua/3/3_17/3_170895_osnovnie-osobennosti-metodologii-RAD.html.

Наконечна Оксана,
кандидат технічних наук,
доцент кафедри комп'ютерних наук та інформаційних технологій,
Якимчук Богданна,
кандидат технічних наук,
старший викладач кафедри комп'ютерних наук та інформаційних технологій,
Ярмоленко Тетяна,
асистент кафедри комп'ютерних наук та інформаційних технологій,
Житомирський державний університет імені Івана Франка,
м. Житомир, Україна

ОСНОВНІ ПРИНЦИПИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Можливості подальшого нарощування продуктивності комп'ютерів в рамках послідовних принципів обробки даних практично вичерпали себе, що обумовлено в основному скінченою швидкістю розповсюдження сигналів. Пошук вирішення проблеми підвищення продуктивності йде в напрямку розвитку принципів паралельної обробки інформації. Отримати суттєвий приріст в продуктивності можна лише у використанні принципово нових комп'ютерних архітектур, які основані на паралельній обробці даних.

Основні принципи паралелізму були впроваджені ще в перші експериментальні паралельні машини, які з'явились в 60-70 роках минулого століття. У векторній CRAY-1, матричній ILLIAC-4, ортогональній OMEN та в інших комп'ютерах тих часів були започатковані основні напрямки паралельних обчислень: конвеєризація, процесорні матриці, асоціативна адресація [1].

В наші дні принципи паралелізму використовуються в більшості обчислювальних пристроїв, найбільш поширеними паралельними комп'ютерами є кластерні системи та багатоядерні персональні комп'ютери [2]. Прогрес в обчислювальній техніці викликав інтенсивний розвиток відповідного програмного забезпечення. Утворився окремий розділ в програмуванні – паралельне програмування.

До основних питань, якими займається паралельне програмування, відносяться розробка паралельних алгоритмів та їх реалізація мовами паралельного програмування для конкретних паралельних архітектур. Основна проблема сучасного паралельного програмування полягає у складності побудови схеми паралельних обчислень. Велику допомогу програмісту можуть надати технології паралельних обчислень, які вже стали визнаними стандартами: OpenMP, MPI, CUDA.

Суть паралельної обробки даних полягає в розподілі всієї обчислювальної роботи на окремі частини і їх одночасному виконанні, що в підсумку має дати виграш в часі виконання всієї роботи. Використовуючи математичну термінологію, кажуть про декомпозицію початкової обчислювальної задачі.

Виділяють такі способи декомпозиції:

– за даними,
– за функціями (підзадачами),
– за часом.

Відповідно можна розрізняти паралелізм за даними, паралелізм за функціями (підзадачами) та паралелізм за часом.

Основна ідея паралелізму за даними полягає в тому, що одна операція виконується одночасно над всіма елементами масиву даних, наприклад, «помножити всі елементи масиву на задану константу». В програмах, де використовується паралелізм за даними, використовується глобальний простір імен на основі єдиного блоку пам'яті та багатьох процесорних блоків (ядер). Різні фрагменти масиву обробляються на різних процесорах (ядрах) паралельної машини. Таким чином, такий спосіб розпаралелювання виконується на машинах з архітектурою SIMD[2].

Характерною особливістю таких обчислень є їхня слабка синхронізація, тобто процесори працюють незалежно і немає гарантії, що в заданий момент часу на всіх процесорах виконується одна і та ж команда. Розподіл даних між процесорами задається в програмі. Роль програміста зводиться лише до розбиття початкових даних на рівні за величиною блоки D1, D2 , … , Dn (рис.1) та задання відповідних директив (опцій), а власне векторизація чи розпаралелювання виконується на етапі компіляції – під час переведення початкового тексту програми в машинні коди.

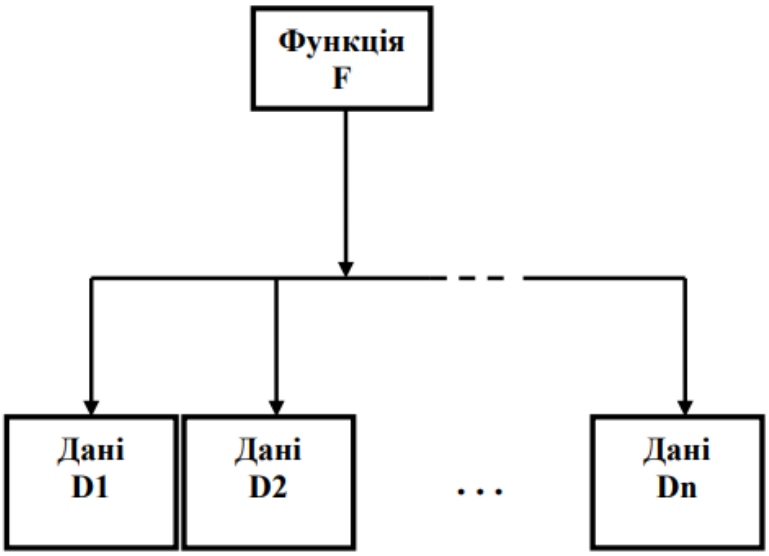

Рис. 1. Декомпозиція за даними

Стиль програмування, який базується на паралелізмі функцій (підзадач), полягає в розбитті всієї обчислювальної задачі на декілька відносно самостійних підзадач (методів або функцій F1, F2 , … , Fk), які виконуються в окремому процесорі або ядрі (рис. 2).

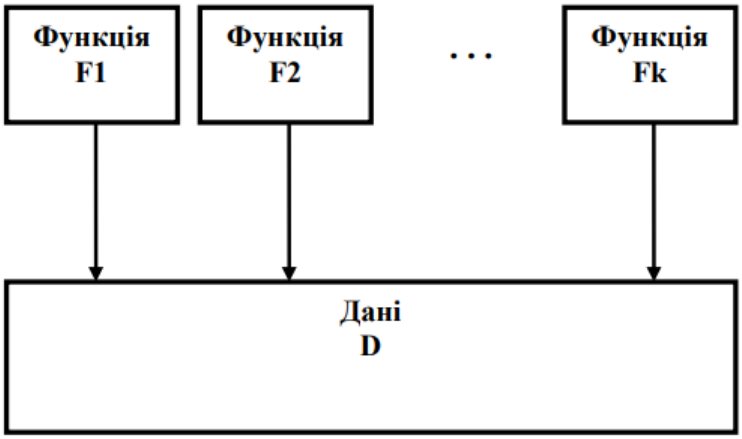

Рис. 2. Декомпозиція за підзадачами

Такому способу розпаралелювання обчислень теоретично відповідає архітектура MISD, але на практиці використовуються машини з Дані D1 Дані D2 Дані . . . Dn Функція F Функція . . . F1 Дані D Функція Fk Функція F2 10 архітектурою MIMD, оскільки різні підзадачі, як правило, використовують і різні дані. Для кожної підзадачі пишеться своя окрема функція чи програма, які виконуються на окремих процесорах і ядрах. Пам’ять може бути спільною або розподіленою. Характерною особливістю таких обчислень є обмін проміжними та кінцевими даними між підзадачами. У випадку розподіленої пам’яті такий обмін даними можливий лише як обмін повідомленнями між паралельними процесами.

Корисно порівняти між собою дві найпоширеніші технології паралельної обробки даних. По-перше, необхідно обов’язково враховувати особливості та обмеження практичної реалізації обчислень. Розмір блока даних і складність

підзадачі мають бути такими, щоб число допоміжних операцій в процесорі (ядрі) не перевищувало число основних операцій. Іншими словами, сумарний час створення і ліквідації потоків має бути меншим часу обчислень. Така вимога буде виконана, якщо, наприклад, процес чи потік буде виконувати роботу, яка за трудомісткістю буде не меншою, ніж 2000 операцій ділення чисел з рухомою комою. Звичайно, кожна задача має свою специфіку і точні оцінки ефективності паралельної обробки можна визначити лише експериментально. В цілому, паралелізм даних простіше та краще масштабується до дуже паралельного апаратного забезпечення, оскільки це зменшує або усуває спільні дані (тим самим зменшуючи проблему безпеки потоків). Крім того, паралелізм даних використовує той факт, що більше буває значень даних, ніж дискретних задач. Нарешті, корисно враховувати ступінь структурованості паралелізму. Паралелізм даних має кращу структуровану паралельність, тобто, паралельні процеси та потоки стартують і фінішують в одному місці в програмі. На противагу цьому, паралелізм підзадач має тенденцію бути неструктурованим, а це означає, що паралельні процеси і потоки можуть починатись і закінчуватись в різних місцях програми. Програми з низьким 11 ступенем структурного паралелізму складніші в налагоджуванні і більше піддаються помилкам. На практиці паралелізм даних та паралелізм функцій (підзадач) взаємно доповнюють один одного, тому у великих програмах часто застосовуються разом [3].

За останні роки було запропоновано різноманітні бібліотеки, компілятори, системні та сервісні програми, які допомагають програмісту у написанні та налагоджуванні паралельних програм. Вміння ефективно застосовувати ці програмні інструменти є важливим показником професіоналізму сучасного програміста.

Список використаних джерел та літератури

1. Семеренко В. П. Темпоральные модели параллельных вычислений. Austrian Journal of Technical and Natural Sciences. 2014. Vol. 1. P. 13–25.

2. Семеренко В. П. Теорія циклічних кодів на основі автоматних моделей: монографія. Вінниця : ВНТУ, 2015. 444 с. 10.

3. Шеховцов В. А. Операційні системи. Київ: Видавнича група BHV, 2005. 576 с.