УДК 004.021

ЕТАПИ РОЗРОБКИ ПАРАЛЕЛЬНИХ АЛГОРИТМІВ

Жолубак Л.І., Бурак Н.Є.
Львівський державний університет безпеки життєдіяльності, м. Львів

У роботі розглянуто основні етапи розробки паралельних алгоритмів. Розв'язування задач за допомогою паралельних алгоритмів.
Ключові слова: паралельна обробка даних, алгоритм, міжпроцесорний обмін, топологія.

The main stages of development of parallel algorithms are considered in the work. Solve problems using parallel algorithms.
Keywords: parallel data processing, algorithm, interprocessor exchange, multiprocessor system, topology.

Поняття *паралельного алгоритму* (Parallel Algorithm) відноситься до фундаментальних в теорії обчислювальних систем. Це поняття, перш за все, асоціюється з обчислювальними системами з масовим паралелізмом. Паралельний алгоритм - це опис процесу обробки інформації, орієнтований на реалізацію в колективі обчислювачів. Такий алгоритм, на відміну від послідовного, передбачає одночасне виконання множини операцій в межах одного кроку обчислень і як послідовний алгоритм зберігає залежність подальших етапів від результатів попередніх.

Паралельний алгоритм рішення задачі складає основу паралельної програми. Паралельна програма у свою чергу, впливає на алгоритм функціонування колективу обчислювачів. Запис паралельного алгоритму на мові програмування, доступній колективу обчислювачів, називають паралельною програмою. Паралельні алгоритми і програми слід розробляти для складних або трудомістких завдань.

Вважатимемо, що відомо традиційний (послідовний) спосіб розв'язання деякої задачі і далі необхідно організувати її виконання з використанням паралельної обробки даних.

Загальна схема розробки паралельного алгоритму містить такі етапи:

– виконання декомпозиції задачі на складові частини одним із відомих способів;

– виявлення інформаційних залежностей;

– масштабування складових частин задачі;

– розподіл складових частин задачі між процесорами (ядрами).

Етап декомпозиції є першим етапом розробки паралельного алгоритму. Суть найвідоміших способів декомпозиції вже було детально розглянуто. Проблема може полягати у виборі того чи іншого способу.

Дуже часто вже наперед відома архітектура паралельної машини, де буде виконуватись задана обчислювальна задача. В цьому випадку архітектура машини визначатиме найоптимальніший варіант декомпозиції. Звичайно, можна використовувати одночасно декілька способів декомпозиції.

Після розбиття початкової задачі на складові частини виконується аналіз зв'язків між ними, тобто здійснюється ***етап виявлення інформаційних залежностей*.** При цьому необхідно розрізняти такі схеми взаємодії:

– локальні (на сусідніх процесорах) і глобальні (з участю всіх процесорів);

– структурні (відповідають типовим топологіям комунікацій згідно з вибраною на попередньому етапі архітектурою машини) і довільні;

– статичні (задаються на етапі проектування) та динамічні (визначаються під час роботи);

– синхронні (наступна операція виконується після закінчення попередньої всіма процесорами) і асинхронні (без очікування повного завершення всіх дій із передачі даних).

Дві підзадачі А та В вважаються інформаційно залежними, якщо результат виконання підзадачі А використовується як вхідні дані для підзадачі В або навпаки. Підзадачі А та В будуть також інформаційно залежними, якщо їх результати мають бути отримані одночасно для подальшого використання іншими підзадачами.

Етап масштабування складових частин задачі виконується в тому випадку, коли кількість наявних підзадач відрізняється від кількості наявних процесорів (ядер). Тут можливі два основних варіанти.

Якщо кількість k підзадач перевищує кількість n процесорів, то деякі підзадачі необхідно укрупнити, так щоб їх загальна кількість не перевищувала числа n.

Існує навіть спеціальний термін – «зернистість», – який характеризує рівень декомпозиції початкової задачі на окремі підзадачі. Дрібнозернистий паралелізм може оптимально завантажити всі наявні процесори, однак важко аналізувати паралельну програму. При цьому є межа складності підпрограм, коли загальний час виконання програм вже не буде зменшуватись внаслідок зростання числа допоміжних операцій зі створення нових процесів та потоків.

Таким чином, в багатьох випадках необхідний ще один етап розробки паралельних алгоритмів – ***етап розподілу підзадач між процесорами.***

Основний критерій успішності виконання цього етапу – ефективність використання процесорів, яка визначається як відносна частка часу, протягом якого процесори використовувались для обчислень, пов'язаних з виконанням поставленої задачі. Способи досягнення задовільних результатів в цьому напрямку базуються на таких самих принципах, як і в попередніх етапах: рівномірний розподіл обчислювального навантаження процесорів та мінімізація обмінів даних між ними.

Варто відзначити, що вимога мінімізації міжпроцесних обмінів може суперечити умові рівномірного завантаження. Можна розмістити всі підзадачі на одному процесорі і тим самим повністю ліквідувати міжпроцесний обмін, але завантаження процесорів в цьому випадку буде неоптимальним.

Вирішення питань балансування обчислювального навантаження значно ускладнюється, якщо схема обчислень може змінюватись під час розв'язання задачі. Для динамічного керування розподілом обчислювального навантаження часто використовується схема «менеджер– виконавці». Відповідно до такої схеми виділяється окремий процесор (менеджер), якому доступна вся інформація про стан виконання всіх підзадач на інших процесорах (виконавцях). Процесор-менеджер отримує результати виконання підзадач від процесорів-виконавців, формує нові завдання та необхідні ресурси для їх виконання.

Завершення обчислень відбувається тоді, коли процесори-виконавці завершили виконання всіх переданих їм підзадач, а процесор-менеджер не має більше нових завдань.

Отже, паралельні алгоритми досить важливі з огляду на постійне вдосконалення багатопроцесорних систем і збільшення числа ядер у сучасних процесорах. Зазвичай простіше сконструювати комп'ютер з одним швидким процесором, ніж з багатьма повільними з тією ж продуктивністю. Однак збільшення продуктивності за рахунок вдосконалення одного процесора натрапляє на фізичні обмеження, такі як досягнення максимальної щільності елементів та тепловиділення. Зазначені обмеження можна подолати лише шляхом переходу до багатопроцесорної архітектури, що виявляється ефективним навіть у малих обчислювальних системах.

Література

1. Гаврилова Т.А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем // Новости искусственного интеллекта. – №2, 2003. – С. 24-30.
2. Дорошенко А.Е. Математические модели и методы организации высокопроизводительных вычислений, Киев: Наукова думка, 2000.
3. Немнюгин С.А., Стесик О.Л.Параллельное программирование для многопроцессорных систем.- СПБ. БХВ Петербург, 2002.-400 с.
4. Цаленко М.Ш. Моделирование семантики в базах данных. – М.: Наука, 1989. –354с..