Лабораторна робота 2. Паралельні обчислення для мультипроцесорів на основі технології OpenMP

Мета роботи: опанувати техніку розроблення паралельних програм у мультипроцесорному середовищі.

Теоретичні відомості

У теперішній час технологія OpenMP (Open MultiProcessing) є одним з найпопулярніших засобів розроблення програм для комп'ютерів зі спільною пам'яттю. OpenMP – відкритий стандарт для розпаралелювання програм, написаних мовами C/C++ і Фортран. Він описує набір директив компілятора, бібліотечних функцій і змінних оточення, які призначені для розроблення багатопоточних застосунків на багатопроцесорних системах зі спільною пам'яттю. За основу береться послідовна програма, а для створення її паралельної версії в розпорядження користувача надається сукупність директив, функцій і змінних оточення. Передбачається, що створювана паралельна програма буде переносимою між різними комп'ютерами з поділюваною пам'яттю, що підтримують OpenMP API.

Головною метою технології OpenMP є те, щоб користувач мав один варіант програми для паралельного й послідовного виконання. Однак існує можливість створювати програми, які працюють коректно тільки в паралельному режимі або дають у послідовному режимі інший результат. Більше того, через накопичення помилок округлення результат обчислень із використанням різної кількості потоків може в деяких випадках різнитися.

Інтерфейс OpenMP був задуманий як стандарт для програмування на масштабованих SMP-системах зі спільною пам'яттю. OpenMP реалізує паралельні обчислення за допомогою багатопотоковості, в якій головний (master) потік створює набір підлеглих (slave) потоків, і обчислення розподіляються між усіма потоками. Передбачається, що потоки виконуються паралельно на машині з декількома процесорами (кількість процесорів не обов'язково повинна бути більшою або дорівнювати кількості потоків).

POSIX-інтерфейс для роботи з потоками (Pthreads) підтримується практично всіма UNIX-системами, однак з багатьох причин не підходить для практичного паралельного програмування: у ньому немає підтримки мови Фортран, занадто низький рівень програмування, немає підтримки паралелізму при роботі з даними, а сам механізм потоків з самого початку не мав на меті організацію паралельних обчислень. OpenMP можна розглядати як високорівневу надбудову над Pthreads (або аналогічними бібліотеками для роботи з потоками); в OpenMP використовується термінологія й модель програмування, близька до Pthreads, наприклад, динамічно породжувані потоки, спільні й поділювані дані, механізм блокувань для синхронізації. Згідно з термінологією Pthreads, будь-який процес в UNIX-системі складається з декількох потоків керування, які мають спільний адресний простір, але різні потоки команд і відокремлені стеки. У найпростішому випадку процес містить лише один потік. Потоки іноді називають також легковагими процесами (lightweight processes).

Чим приваблює користувача технологія OpenMP? Можна відзначити декілька моментів, серед яких варто особливо виділити два. По-перше, технологія з самого початку спроектована таким чином, щоб користувач міг працювати з єдиним текстом для паралельної й послідовної версій програми. Звичайний компілятор з мови програмування на послідовній машині просто “не помітить” директив OpenMP, оскільки вони оформлені у вигляді директив компілятора #pragma. Єдиним джерелом проблем можуть стати змінні оточення й спеціальні функції. Однак для них у специфікаціях стандарту передбачені спеціальні “заглушки”, що гарантують коректну роботу OpenMP-програми в послідовному випадку. Для доступу до ресурсів інтерфейсу OpenMP потрібно перекомпілювати програму, зазначивши прапорець -f і підключивши відповідну бібліотеку.

Іншою важливою позитивною якістю технології OpenMP є можливість реалізації так званого інкрементного програмування, коли програміст поступово знаходить ділянки в програмі, які можуть бути розпаралелені, і за допомогою надаваних механізмів робить їх паралельними, а потім переходить до аналізу наступних ділянок. Таким чином, послідовно виконувана частина програми поступово зменшується. Такий підхід значно полегшує процес адаптації послідовних програм до паралельних комп'ютерів, а також їх налагодження й оптимізацію.

Завдання, виконувані потоками паралельно, так само як і дані, необхідні для виконання цих завдань, описують за допомогою спеціальних директив препроцесора відповідної мови – прагм. Вона виглядає як директива препроцесора, але насправді не має до препроцесора майже жодного відношення. Ця конструкція для мов C/C++ має такий синтаксис:

#pragma [список параметрів]

Вона використовується для передачі вказівок компілятору, тобто для керування його роботою, а якщо компілятор їх не розуміє, то вказівки просто ігноруються. Гарний стиль написання програм полягає в тому, щоб програма працювала й видавала правильний результат незалежно від того, чи розуміє конкретний компілятор вказівки, які містить директива, або ні (тобто вказівки повинні впливати тільки на продуктивність, але не на працездатність програми). Нижче ми розглянемо директиви, що керують розпаралелюванням програми, і якщо компілятор їх не розуміє, програма просто виходить послідовною (тобто без розпаралелювання).

Наприклад, ділянці коду, що написана мовами C/C++, яка повинна виконуватися декількома потоками, кожен з яких має свою копію змінної N, передує така директива:

#pragma omp parallel private(N)

Кількість створюваних потоків може регулюватися як самою програмою за допомогою виклику бібліотечних функцій, так і ззовні, за допомогою змінних оточення.

Контрольні запитання

1. Що таке OpenMP?
2. З яких компонентів складається OpenMP?
3. Опишіть структуру OpenMP-програми.
4. Як створюються потоки в OpenMP-програмі?
5. Яким чином відбувається розподіл обчислень в OpenMP?
6. В який спосіб відбувається в OpenMP взаємодія між потоками?
7. Які класи даних слід розрізняти у паралельній ділянці?
8. Чим визначається кількість потоків за замовчуванням у паралельній ділянці?
9. Як можна змінити кількість потоків у паралельній ділянці в явний спосіб?
10. Яким чином здійснюється синхронізація потоків?
11. Назвіть найважливіші змінні оточення OpenMP.
12. Опишіть призначення вбудованих функцій OpenMP.
13. Як запустити OpenMP-програму на виконання?

Індивідуальні завдання

Напишіть паралельну програму в середовищі OpenMP для обчислення потрійного інтеграла

$$I = \int_0^1 \int_0^1 \int_0^1 x^4 \cdot y^4 \cdot z^4 dxdydz = 1/125$$

за формулою прямокутників $I \approx \sum_{i=0}^{n_i-1} \sum_{j=0}^{n_j-1} \sum_{k=0}^{n_k-1} f\left(x_i, y_j, z_k\right) \cdot \Delta x \Delta y \Delta z$ на прямокутній сітці, де $x_i = i \cdot \Delta x,\ y_j = j \cdot \Delta y,\ z_k = k \cdot \Delta z$ за умови, що $n_i = 300,\ n_j = 200,\ n_k = 400$. Для розподілу обчислення по процесорах використовуйте властивість адитивності інтеграла.