

Лабораторна робота №5

Тема: Технологія OpenMP

Мета: Познайомитись з базовими директивами OpenMP та навчитись їх застосовувати для розпаралелювання послідовних програм.

Теоретичні відомості

OpenMP — це стандартна модель паралельного програмування для середовищ зі спільною пам'яттю, яка дозволяє реалізувати багатопоточні програми мовами C, C++ та Fortran.

OpenMP базується на прагмах (директивах компілятора), які легко інтегруються у наявний код і можуть бути деактивовані без видалення, повертаючи програму до послідовного виконання.

Основні директиви:

1. **#pragma omp parallel** — створює паралельну область:
 - num_threads(n) — кількість потоків;
 - private(list) — створює локальні копії змінних;
 - firstprivate(list) — локальні копії з ініціалізацією;
 - shared(list) — спільні змінні;
 - reduction(op: list) — агрегування з локальних копій.
2. **#pragma omp for** — розподіл ітерацій циклу:
 - schedule(type[, chunk]) — тип розподілу: static, dynamic, guided, auto, runtime;
 - collapse(n) — об'єднання вкладених циклів.
3. **#pragma omp sections** — виконує незалежні ділянки коду паралельно:
 - #pragma omp section — одна секція;
 - nowait — дозволяє уникнути бар'єра після виконання секцій.
4. **#pragma omp barrier** — бар'єрна синхронізація потоків.
5. **#pragma omp critical [(name)]** — критична секція, яку може виконувати лише один потік одночасно.
6. **#pragma omp single** — ділянку виконує лише один (будь-який) потік.
7. **#pragma omp master** — ділянку виконує тільки головний потік.

Хід роботи:

На основі прикладу з репозиторію <https://github.com/oboris/openMpDemoCPP> реалізовано програму на C++, яка демонструє використання технології OpenMP для розпаралелювання обчислень. Програма виконує такі завдання:

1. Обчислює суму всіх елементів двовимірного масиву.
2. Визначає номер рядка з мінімальною сумою елементів та саме значення цієї суми.

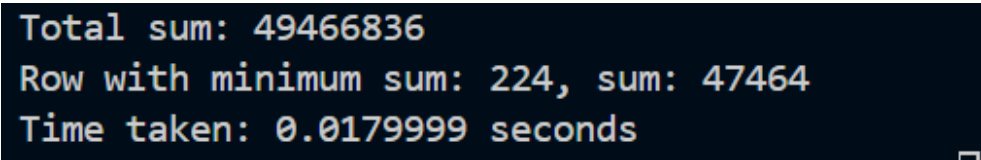
Кожне завдання реалізовано у окремій функції, що використовує директиви OpenMP (parallel, for, reduction) для прискорення обчислень.

Виклики функцій винесено в окремі паралельні секції за допомогою директиви #pragma omp sections, що дозволяє виконувати обидва завдання одночасно.

Для демонстрації ефективності реалізовано вимірювання часу виконання у двох режимах:

- послідовне виконання (без використання OpenMP);
- паралельне виконання (з використанням OpenMP).

Результат:



```
Total sum: 49466836
Row with minimum sum: 224, sum: 47464
Time taken: 0.0179999 seconds
```

Висновок:

У результаті виконання лабораторної роботи було ознайомлено з базовими директивами технології OpenMP та набуті практичні навички їх застосування для розпаралелювання обчислень у C++.

За допомогою директив parallel, for, reduction та sections було реалізовано ефективний розподіл обчислювальних завдань між потоками, що значно прискорило виконання програми порівняно з послідовним варіантом.

Розпаралелювання двох окремих функцій, які виконують підрахунок суми елементів масиву та пошук рядка з мінімальною сумою, у паралельних секціях дозволило одночасно виконувати обидва завдання, підвищуючи загальну продуктивність.

Отримані результати демонструють, що використання OpenMP є ефективним способом оптимізації ресурсів сучасних багатоядерних процесорів для задач з

Гаврилюк А.В.

розподіленими обчисленнями, а також показали зручність та простоту інтеграції OpenMP у існуючий код.

Посилання на репозиторій з вихідними кодом:

<https://github.com/AnnaHavryliuk4/ParallelProcessesCourse/tree/main/lab5>