Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

ЗВІТ   
з виконання лабораторної роботи №2  
з дисципліни «Технології паралельних обчислень в енергетичних комплексах»

«Оброблення сигналів в багатопотокових застосунках»

Варіант 21

Виконав: студент групи ТР-12 Руденко В.І

Завдання:

Напишіть паралельну програму в середовищі OpenMP для обчислення потрійного інтеграла

за формулою прямокутників

на прямокутній сітці, де *xi = i · ∆x, yj = j · ∆y, zk = k · ∆z* за умови, що *ni = 300, nj = 200, nk = 400.* Для розподілу обчислення по процесорах використовуйте властивість адитивності інтеграла.

Хід Роботи:

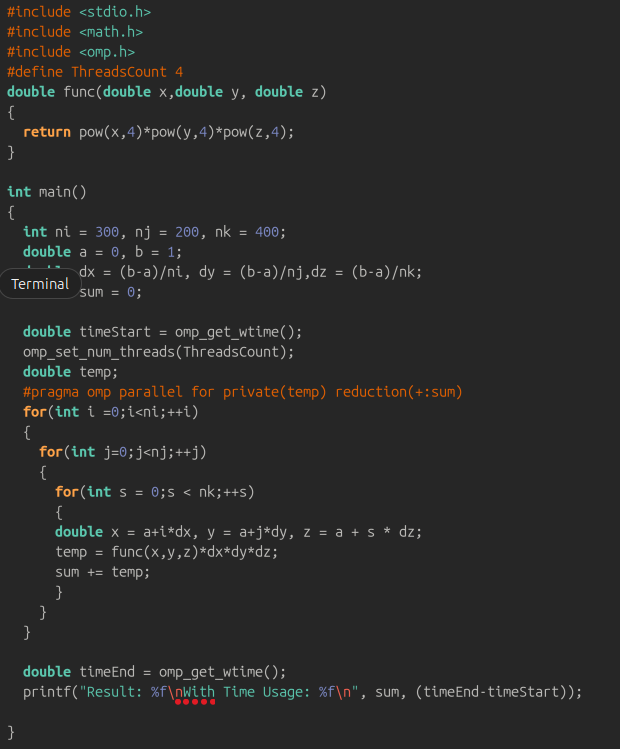


Table 1 Код паралельного виконання OMP

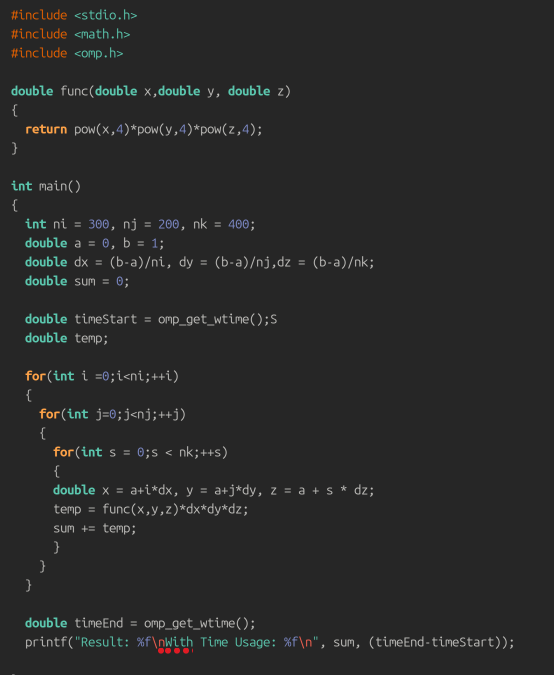


Table 2 Код виконання в однопоточному порядку

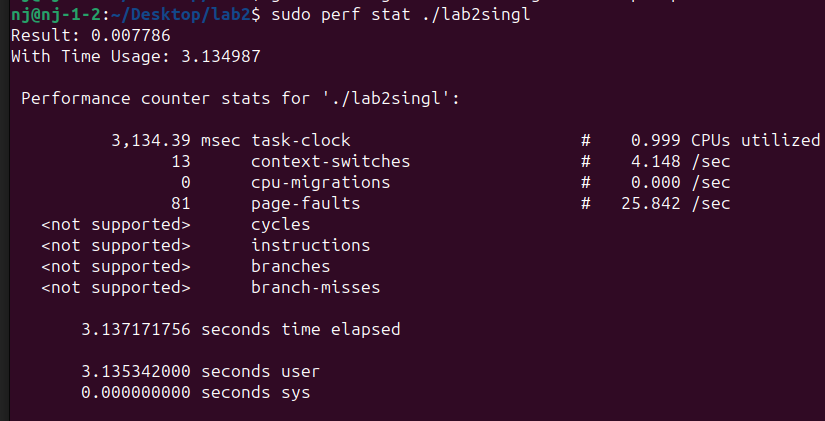


Table 3 Результат та статисктика виконання в однопоточному режимі

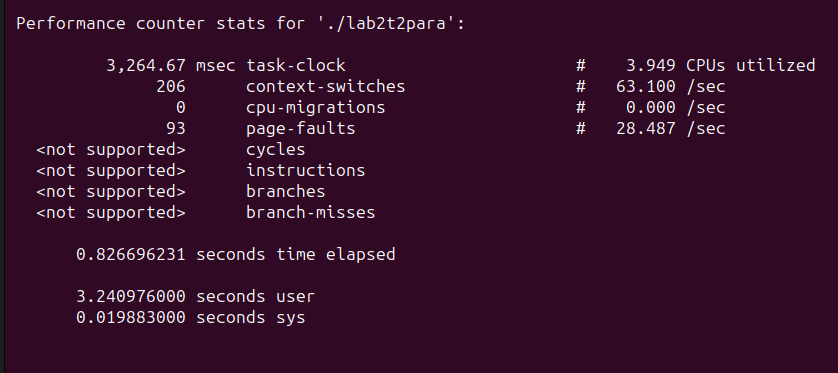


Table 4 Table 3 Результат та статисктика виконання в багатопоточному режимі

**Контрольні запитання**

1. Що таке OpenMP?

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.

2. З яких компонентів складається OpenMP?

Бібліотека OpenMP.

Бібліотека OpenMP містить набір функцій, які можна використовувати для створення та керування паралельними програмами. Ці функції можна використовувати для створення паралельних потоків, завдання паралельним потокам, управління доступом до спільних даних та синхронізації паралельних потоків.

Директиви OpenMP.

Директиви OpenMP є спеціальними директивами, які можна використовувати для позначення паралельних конструкцій у програмному коді. Ці директиви дозволяють компілятору розпізнавати паралельні конструкції та створювати відповідний код.

Компілятор OpenMP.

Компілятор OpenMP — це компілятор, який підтримує директиви OpenMP. Компілятор OpenMP використовується для компіляції програм, які містять директиви OpenMP, у машинний код.

3. Опишіть структуру OpenMP-програми.

Послідовні секції – це частини коду, які виконуються послідовно та одним потоком.

Паралельні секції – це блоки коду, які виконуються паралельно декількома потоками всередині паралельної області.

Директиви OpenMP:

#pragma omp parallel – розпочинає паралельну область, в якій кожен потік виконує свій набір інструкцій.

#pragma omp for – розподіляє ітерації циклу між потоками для паралельного виконання.

#pragma omp barrier – блокує виконання потоків до тих пір, поки всі потоки не дійдуть до цієї директиви.

#pragma omp critical – визначає критичну секцію коду, яка може бути виконана тільки одним потоком одночасно.

#pragma omp shared – задає змінні, які будуть спільними для всіх потоків.

#pragma omp private – вказує, що кожний потік має свою окрему версію цієї змінної.

4. Як створюються потоки в OpenMP-програмі?

У OpenMP потоки створюються при вході в паралельну секцію програми з директивою #pragma omp parallel. Коли програма досягає цієї директиви, OpenMP створює вказану кількість потоків, які можуть виконуватися одночасно на різних процесорах або ядрах. Кількість потоків може встановлюватись за замовчуванням, а може бути встановлена явно.

5. Яким чином відбувається розподіл обчислень в OpenMP?

У OpenMP розподіл обчислень зазвичай відбувається за допомогою розпаралелювання циклів або інших ітераційних структур даних. Це виконується за допомогою директиви #pragma omp for, яка автоматично розподіляє ітерації між потоками.

Коли виконується ця директива, OpenMP автоматично розбиває цикл на частини і розподіляє їх між потоками. Кожен потік виконує свій власний набір ітерацій, зазвичай приблизно однаковий за об'ємом обчислень, для досягнення більшої ефективності.

6. В який спосіб відбувається в OpenMP взаємодія між потоками?

У OpenMP взаємодія між потоками відбувається за допомогою кількох механізмів. Критичні секції обмежують доступ до певних частин коду лише для одного потоку одночасно, щоб уникнути гонок даних. Атомарні операції забезпечують неділимість операцій, запобігаючи гонкам даних. Синхронізація бар'єрів забезпечує синхронізацію потоків, чекаючи, поки всі потоки дійдуть до певної точки в коді. Замки блокують доступ до спільних ресурсів, дозволяючи лише одному потоку виконувати їх у певний момент часу. Захист даних визначає, які дані є приватними для кожного потоку та які є загальними для всіх потоків, управляючи доступом до спільних даних і запобігаючи конфліктам.

7. Які класи даних слід розрізняти у паралельній ділянці?

Спільні (shared) змінні: Це змінні, які доступні для читання та запису всіма потоками в паралельній секції. Вони знаходяться в загальній пам’яті і можуть бути використані для обміну даними між потоками. Однак, при їх використанні необхідно забезпечити синхронізацію, щоб уникнути гонок за дані.

Індивідуальні (private) змінні: Це локальні змінні для кожного потока, які не діляться з іншими потоками. Кожен потік має свою копію таких змінних, і вони існують тільки в межах потока, що виконується. Це дозволяє уникнути конфліктів при доступі до даних і зменшує потребу в синхронізації.

8. Чим визначається кількість потоків за замовчуванням у паралельній ділянці?

Кількість потоків за замовчуванням у паралельній ділянці в OpenMP зазвичай визначається кількістю доступних процесорних ядер. Таким чином, якщо у системі доступно, скажімо, 4 ядра, то за замовчуванням буде створено 4 потоки для виконання паралельної ділянки програми.

9. Як можна змінити кількість потоків у паралельній ділянці в явний спосіб?

Використати директиву num\_threads, щоб явно вказати кількість потоків, які мають бути створені у паралельній області. Наприклад: #pragma omp parallel num\_threads(8).

Встановити кількість потоків за допомогою змінної середовища OMP\_NUM\_THREADS. Наприклад, перед виконанням програми в командному рядку можна встановити кількість потоків: export OMP\_NUM\_THREADS=8

Встановити кількість потоків за допомогою функцій OpenMP, таких як omp\_set\_num\_threads().

10. Яким чином здійснюється синхронізація потоків?

Синхронізація потоків у OpenMP здійснюється за допомогою різних механізмів, які дозволяють управляти взаємодією між потоками та уникати конфліктів при доступі до спільних ресурсів. Ці механізми включають в себе бар'єри, критичні секції, атомарні операції та замки. Бар'єри вимагають, щоб всі потоки дочекалися певної точки в коді перед продовженням виконання програми, тим самим забезпечуючи синхронізацію. Критичні секції блокують доступ до певних частин коду лише для одного потоку одночасно, щоб уникнути гонок даних. Атомарні операції дозволяють виконувати операції над спільними даними атомарно, забезпечуючи надійність при доступі до спільних ресурсів. Замки використовуються для блокування доступу до спільних ресурсів або кодових ділянок, дозволяючи лише одному потоку виконувати їх у певний момент часу.

11. Назвіть найважливіші змінні оточення OpenMP.

OMP\_NUM\_THREADS: Визначає кількість потоків, які будуть використовуватися в паралельній області.

OMP\_DYNAMIC: Дозволяє динамічно змінювати кількість потоків в паралельній області під час виконання програми.

OMP\_SCHEDULE: Визначає режим планування для розподілу ітерацій в паралельних циклах.

OMP\_NESTED: Дозволяє вкладати паралельні області одну в одну, якщо встановлено в true.

12. Опишіть призначення вбудованих функцій OpenMP.

omp\_get\_thread\_num(): Повертає номер потока, який виконує її в момент виклику.

omp\_get\_num\_threads(): Повертає загальну кількість потоків, які були створені для виконання паралельної області.

omp\_get\_max\_threads(): Повертає максимальну кількість потоків, яка може бути створена в паралельній області.

omp\_set\_num\_threads(): Встановлює кількість потоків, які мають бути створені для наступної паралельної області.

omp\_get\_wtime(): Повертає час в секундах від початку виконання програми до моменту виклику функції.

omp\_get\_num\_procs(): Повертає загальну кількість процесорних ядер у системи.

omp\_in\_parallel(): Перевіряє, чи виконується поточна функція в паралельній області OpenMP.

omp\_set\_dynamic(): Встановлює режим динамічного розподілу потоків у паралельній області. Коли цей режим увімкнено, кількість потоків може змінюватися відносно потреб програми.

omp\_get\_nested(): Повертає стан вкладених паралельних областей. Якщо вони ввімкнені, можна вкладати одну паралельну область всередину іншої.

omp\_set\_lock(): Встановлює блокування для поточного потоку.

omp\_unset\_lock(): Використовується для звільнення блокування, яке було встановлено за допомогою функції omp\_set\_lock().

omp\_test\_lock(): Перевіряє, чи встановлене блокування, і повертає true, якщо блокування доступне для поточного потоку.

omp\_init\_lock(): Ініціалізує блокування.

omp\_destroy\_lock(): Звільняє ресурси, пов'язані з блокуванням, і завершує його використання.

13. Як запустити OpenMP-програму на виконання?

Скомпілювати програму за допомогою компілятора, який підтримує OpenMP. Для цього зазвичай використовують флаг компілятора, наприклад, -fopenmp для GCC або /openmp для MSVC.