Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №2

з дисципліни «Технології паралельних обчислень в енергетичних комплексах»

Тема «Паралельні обчислення для мультипроцесорів на основі технології OpenMP»

Варіант №19

Студента 3-го курсу НН ІАТЕ гр. ТР-12
Ковальова Олександра
Перевірив: ас., Софієнко А. Ю.

КИЇВ 2024

Мета роботи. Опанувати техніку розроблення паралельних програм у мультипроцесорному середовищі.

Завдання: напишіть паралельну програму в середовищі OpenMP для обчислення потрійного інтеграла

$$I = \int_0^1 \int_0^1 \int_0^1 x^4 * y^4 * z^4 \, dxdydz = \frac{1}{125}$$

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

Хід роботи

Програмний код:

```c
#include <stdio.h>
#include <omp.h>

double f(double x, double y, double z) {
    return x*x*x*x * y*y*y*y * z*z*z*z;
}


int main() {
    const int ni = 600, nj = 400, nk = 800;
    const double a = 0.0, b = 1.0;
    const double dx = (b - a) / ni;
    const double dy = (b - a) / nj;
    const double dz = (b - a) / nk;
    double sum = 0.0;

    #pragma omp parallel for reduction(+:sum)
    for (int i = 0; i < ni; ++i) {
        for (int j = 0; j < nj; ++j) {
            for (int k = 0; k < nk; ++k) {
                double x = a + i * dx;
                double y = a + j * dy;
                double z = a + k * dz;
                double result = f(x, y, z) * dx * dy * dz;
                sum += result;
            }
        }
    }

    double integral = sum;
    printf("Approximate integral value: %lf\n", integral);

    return 0;
}
```

Результати роботи програми:

```
xairaven@host:~/PCT/Lab2$ sudo perf stat ./nonparallel
Approximate integral value: 0.007892

 Performance counter stats for './nonparallel':

            909.25 msec task-clock                #    0.999 CPUs utilized
                 5      context-switches          #    5.499 /sec
                 0      cpu-migrations            #    0.000 /sec
                60      page-faults               #   65.989 /sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.910109478 seconds time elapsed

       0.905725000 seconds user
       0.004007000 seconds sys


xairaven@host:~/PCT/Lab2$ sudo perf stat ./parallel
Approximate integral value: 0.007892

 Performance counter stats for './parallel':

            964.41 msec task-clock                #    1.996 CPUs utilized
                28      context-switches          #   29.033 /sec
                 0      cpu-migrations            #    0.000 /sec
                73      page-faults               #   75.694 /sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.483065325 seconds time elapsed

       0.964830000 seconds user
       0.000000000 seconds sys
```

Контрольні запитання:

1) **Що таке OpenMP?**

   OpenMP (Open Multi-Processing) – це стандарт для паралельного програмування на багатоядерних та багатопроцесорних системах. Він надає набір директив компілятора, функцій та змінних середовища для спрощення розробки паралельних програм. Основна ідея OpenMP – це вставляти директиви у вихідний код програми, які вказують компілятору, як розподіляти обчислювальні завдання між різними потоками виконання. OpenMP дозволяє розпаралелювати програми на мовах програмування, таких як C, C++, і Fortran. Завдяки його використанню, розробники можуть зручно використовувати потоки для виконання обчислень паралельно, що сприяє покращенню продуктивності програм на багатоядерних системах. Директиви можуть бути використані для створення паралельних областей коду, розподілу обчислювальних завдань між потоками, синхронізації потоків, управління оточенням виконання та іншого.

2) **З яких компонентів складається OpenMP?**
   OpenMP складається з директив компілятора, бібліотечних функцій та змінних середовища. Директиви компілятора вбудовуються в програмний код і вказують компілятору, як розподіляти обчислення між потоками. Бібліотечні функції дозволяють керувати паралельним виконанням, а змінні середовища надають інформацію про оточення виконання.

3) **Опишіть структуру OpenMP-програми.**
   Програма має заголовок, де включаються бібліотеки та визначаються головні функції програми. Паралельні області коду визначаються за допомогою директиви *#pragma omp parallel*. Все, що знаходиться між цією директивою та відповідною фігурною дужкою, вважається паралельною областю, де потоки виконують код паралельно. Також важливо враховувати декларації та ініціалізацію змінних, а також можливе використання бібліотечних функцій та директив для керування паралельним виконанням.

4) **Як створюються потоки в OpenMP-програмі?**
   Потоки створюються автоматично при входженні в паралельну область коду за допомогою директиви *#pragma omp parallel*. Кількість створених потоків може бути налаштована залежно від конфігурації системи або задана явно програмістом. Кожен потік отримує свій власний ідентифікатор, який можна отримати в коді за допомогою функції *omp_get_thread_num()*. Потоки розподіляють обчислювальні завдання між собою, виконуючи їх паралельно та сприяючи покращенню продуктивності програм на багатоядерних системах.

5) **Яким чином відбувається розподіл обчислень в OpenMP?**
   Розподіл обчислень відбувається шляхом автоматичної розпаралелювання паралельних областей коду. Коли програма входить в паралельну область, компілятор автоматично розподіляє обчислення між створеними потоками. Розподіл може включати паралельне виконання циклів, фрагментів коду чи інших обчислювальних завдань між потоками.

6) **В який спосіб відбувається в OpenMP взаємодія між потоками?**
   Взаємодія між потоками може відбуватися через спільний доступ до спільної області пам'яті. Потоки можуть звертатися до одних і тих же змінних та даних, що може призводити до конфліктів і неправильної роботи програми. Для управління цією взаємодією і уникнення проблем в OpenMP використовуються такі засоби: приватні змінні, взаємовиключення (mutex), атомарні операції та інші механізми синхронізації.

7) **Які класи даних слід розрізняти у паралельній ділянці?**
   У паралельній ділянці важливо розрізняти спільні та приватні дані. Спільні дані – це ті, до яких може звертатися більше одного потоку і які потенційно

можуть призвести до конфліктів. Приватні дані – це ті, які кожен потік обробляє незалежно, маючи свої власні копії, що уникне конфліктів.

8) **Чим визначається кількість потоків за замовчуванням у паралельній ділянці?**
Кількість потоків за замовчуванням у паралельній ділянці визначається числом процесорів або ядер на системі. Зазвичай OpenMP автоматично використовує доступні процесорні ресурси для створення потоків і розпаралелювання обчислень.

9) **Як можна змінити кількість потоків у паралельній ділянці в явний спосіб?**
Кількість потоків у паралельній ділянці можна явно вказати за допомогою спеціальної змінної оточення виконання (environment variable). Найчастіше використовується змінна *OMP_NUM_THREADS*, яку можна встановити перед запуском програми. Наприклад, в командному рядку перед викликом програми: *OMP_NUM_THREADS=4 ./your_program*. Це призведе до виконання паралельної ділянки коду з чотирма потоками. Також, кількість потоків можна встановити програмно за допомогою функції *omp_set_num_threads()*.

10) **Яким чином здійснюється синхронізація потоків?**
Синхронізація потоків здійснюється за допомогою різних механізмів. До них входять взаємовиключення (mutex), атомарні операції, директиви для синхронізації та інші засоби. Це дозволяє контролювати взаємодію між потоками, уникати гонок за ресурсами та забезпечувати правильне виконання паралельних операцій.

11) **Назвіть найважливіші змінні оточення OpenMP.**
Найважливіші змінні оточення включають *OMP_NUM_THREADS* для вказівки кількості потоків, *OMP_SCHEDULE* для вибору графіка розподілу ітерацій, *OMP_PROC_BIND* для контролю прив'язки потоків до ядер, та *OMP_DYNAMIC* для включення/виключення динамічного розподілу потоків.

12) **Опишіть призначення вбудованих функцій OpenMP.**
Вбудовані функції використовуються для отримання інформації про виконання паралельної області коду. Наприклад, *omp_get_thread_num()* повертає ідентифікатор потока, а *omp_get_num_threads()* повертає загальну кількість потоків в паралельній області. Ці функції дозволяють програмістам отримувати і контролювати характеристики потоків під час виконання паралельного коду.

13) Як запустити OpenMP-програму на виконання?

OpenMP-програму можна запустити на виконання шляхом компіляції її відповідного вихідного коду за допомогою компілятора, який підтримує стандарт. При цьому, важливо включити опції компілятора, які дозволяють використовувати функціонал OpenMP. Наприклад, для компіляції на C або C++ може використовуватися опція *-fopenmp*, а на Fortran – *-openmp*. Після компіляції і лінкування програма готова до виконання, і вона використовуватиме паралельні можливості OpenMP при виконанні на багатоядерних або багатопроцесорних системах.