Tomasz Ligęza

Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Sprawozdanie z laboratorium 9.

Cel zajęć:

Nabycie umiejętności tworzenia i implementacji programów równoległych z wykorzystaniem OpenMP.

W ramach zajęć zrealizowałem następujące kroki:

1. Utworzyłem katalog roboczy lab_9,
2. Skopiowałem pliki ze strony przedmiotu, skompilowałem je oraz uruchomiłem.
3. Zrównolegliłem wskazaną pętlę za pomocą dyrektywy parallel for bez klauzuli schedule z liczbą wątków sterowaną za pomocą zmiennej środowiskowej:

```
int main() {
   int LICZBA_WATKOW = omp_get_num_threads();
   printf("liczba watkow: %d\n", LICZBA_WATKOW);

[...]

// pętla do modyfikacji - docelowo równoległa w OpenMP
   double suma_parallel = 0.0;
#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(LICZBA_WATKOW) schedule(static, 3) ordered
//#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(LICZBA_WATKOW) schedule(static) ordered
//#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(LICZBA_WATKOW) schedule(dynamic, 2) ordered
//#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(LICZBA_WATKOW) schedule(dynamic) ordered
   for (i = 0; i < WYMIAR; i++) {
       int id_w = omp_get_thread_num();
       // ...
       suma_parallel += a[i];
       // ...
#pragma omp ordered
       printf("a[%2d]->W_%1d  \n", i, id_w);
   }
```

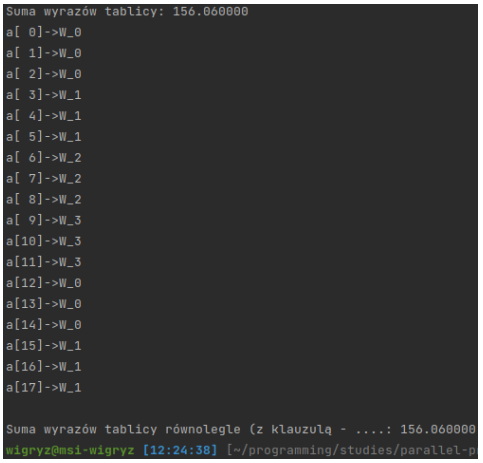
Niestety napotkałem tutaj problem z ustawianiem zmiennej środowiskowej:

```
b_9] [master *]
-> % set OMP_NUM_THREADS = 10
wigryz@msi-wigryz [12:19:48] [~/programming/studies/parallel-pro
b_9] [master *]
-> % ./openmp_petle_simple

libgomp: Invalid value for environment variable OMP_NUM_THREADS
liczba watkow: 1
Suma wyrazów tablicy: 156.060000
a[ 0]->W_0
a[ 1]->W_0
```

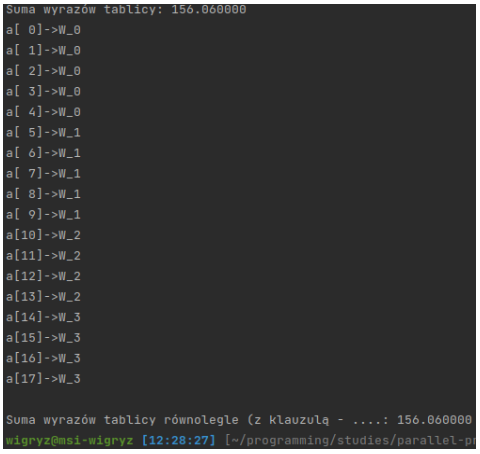
Z tego powodu pominąłem aspekt ustawiania tej zmiennej z poziomu terminala, i ustawiałem ją na “sztywno” w kodzie, co niestety wydłużyło czas pracy przez potrzebę ciągłego kompilowania kodu.

4. Następnie przeprowadziłem test dla kolejnych 4 wersji klauzuli schedule:
    a. static, rozmiar porcji=3:



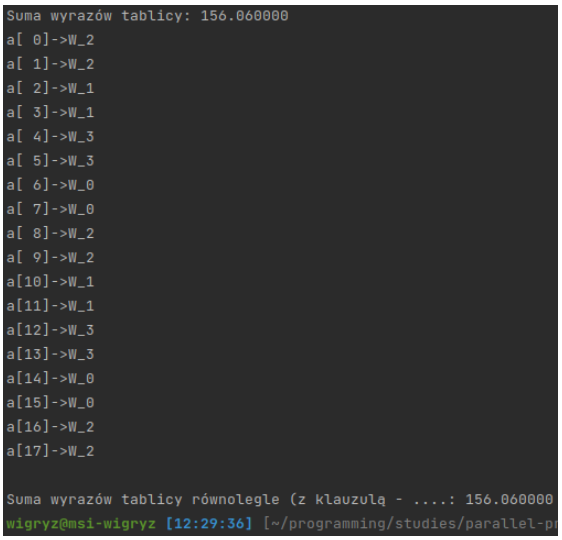
Każdy wątek otrzymuje po 3 iteracje, oraz za każdym razem wątki otrzymują te same iteracje.

    b. static, rozmiar porcji domyślny:



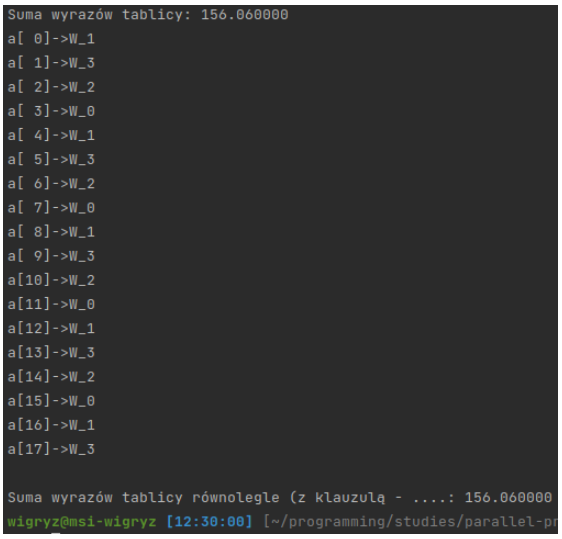
Każdy wątek otrzymuje po 5 iteracji, oraz za każdym razem wątki otrzymują te same iteracje.

c. dynamic rozmiar porcji=2:



Każdy wątek otrzymuje po 2 iteracje, ale są one przydzielane wątkom w sposób dynamiczny, to znaczy nie jest określone, że pierwszy wątek dostanie pierwsze dwie iteracje.

d. dynamic, rozmiar porcji domyślny:



Każdy wątek otrzymuje po jednej iteracji, ale są one przydzielane wątkom w sposób dynamiczny, to znaczy nie jest określone, że pierwszy wątek dostanie pierwszą iterację.

Statyczny podział oznacza ustaloną z góry kolejność wątków, które będą otrzymywały kolejne iteracje, natomiast dynamiczny oznacza “losowy” podział (pod spodem zapewne działa jakiś mechanizm, który zarządza wątkami biorąc pod uwagę to czy są obecnie zajęte).

Domyślnym podziałem jest podział statyczny z domyślną ilością porcji.

5. Skopiowałem plik openmp_petle.c, skompilowałem go oraz uruchomiłem z liczbą wątków 3.

6. Napisałem trzy wersje zrównoleglenia obliczania sumy wyrazów tablicy dwuwymiarowej:
    a. dekompozycja wierszowa (reduction + schedule(static, 2)):

```
#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(NUM_THREADS) private(i,j) schedule(static, 2) ordered
   for (i = 0; i < WYMIAR; i++) {
       int id_w = omp_get_thread_num();
       for (j = 0; j < WYMIAR; j++) {
           suma_parallel += a[i][j];
#pragma omp ordered
           printf("(%1d,%1d)-W_%1d ", i, j, omp_get_thread_num());
       }
       printf("\n");
   }
```

```
Suma wyrazów tablicy: 913.500000
(0,0)-W_0 (0,1)-W_0 (0,2)-W_0 (0,3)-W_0 (0,4)-W_0 (0,5)-W_0 (0,6)-W_0 (0,7)-W_0 (0,8)-W_0 (0,9)-W_0
(1,0)-W_0 (1,1)-W_0 (1,2)-W_0 (1,3)-W_0 (1,4)-W_0 (1,5)-W_0 (1,6)-W_0 (1,7)-W_0 (1,8)-W_0 (1,9)-W_0
(2,0)-W_1 (2,1)-W_1 (2,2)-W_1 (2,3)-W_1 (2,4)-W_1 (2,5)-W_1 (2,6)-W_1 (2,7)-W_1 (2,8)-W_1 (2,9)-W_1
(3,0)-W_1 (3,1)-W_1 (3,2)-W_1 (3,3)-W_1 (3,4)-W_1 (3,5)-W_1 (3,6)-W_1 (3,7)-W_1 (3,8)-W_1 (3,9)-W_1
(4,0)-W_2 (4,1)-W_2 (4,2)-W_2 (4,3)-W_2 (4,4)-W_2 (4,5)-W_2 (4,6)-W_2 (4,7)-W_2 (4,8)-W_2 (4,9)-W_2
(5,0)-W_2 (5,1)-W_2 (5,2)-W_2 (5,3)-W_2 (5,4)-W_2 (5,5)-W_2 (5,6)-W_2 (5,7)-W_2 (5,8)-W_2 (5,9)-W_2
(6,0)-W_0 (6,1)-W_0 (6,2)-W_0 (6,3)-W_0 (6,4)-W_0 (6,5)-W_0 (6,6)-W_0 (6,7)-W_0 (6,8)-W_0 (6,9)-W_0
(7,0)-W_0 (7,1)-W_0 (7,2)-W_0 (7,3)-W_0 (7,4)-W_0 (7,5)-W_0 (7,6)-W_0 (7,7)-W_0 (7,8)-W_0 (7,9)-W_0
(8,0)-W_1 (8,1)-W_1 (8,2)-W_1 (8,3)-W_1 (8,4)-W_1 (8,5)-W_1 (8,6)-W_1 (8,7)-W_1 (8,8)-W_1 (8,9)-W_1
(9,0)-W_1 (9,1)-W_1 (9,2)-W_1 (9,3)-W_1 (9,4)-W_1 (9,5)-W_1 (9,6)-W_1 (9,7)-W_1 (9,8)-W_1 (9,9)-W_1
Suma wyrazów tablicy równolegle: 913.500000
```

Jak widać każdy wątek otrzymał po dwa wiersze - stało się tak, bo użyliśmy klauzulę schedule z ilością porcji równą 2.

    b. dekompozycja kolumnowa (reduction + schedule(dynamic)):

```
   for (i = 0; i < WYMIAR; i++) {
       int id_w = omp_get_thread_num();
#pragma omp parallel for reduction(+:suma_parallel)
      num_threads(NUM_THREADS) private(j) schedule(dynamic) ordered
       for (j = 0; j < WYMIAR; j++) {
           suma_parallel += a[i][j];
#pragma omp ordered
           printf("(%1d,%1d)-W_%1d ", i, j, omp_get_thread_num());
       }
       printf("\n");
   }
```

```
Suma wyrazów tablicy: 913.500000
(0,0)-W_1 (0,1)-W_0 (0,2)-W_2 (0,3)-W_1 (0,4)-W_0 (0,5)-W_2 (0,6)-W_1 (0,7)-W_0 (0,8)-W_2 (0,9)-W_1
(1,0)-W_1 (1,1)-W_2 (1,2)-W_0 (1,3)-W_1 (1,4)-W_2 (1,5)-W_0 (1,6)-W_1 (1,7)-W_2 (1,8)-W_0 (1,9)-W_1
(2,0)-W_2 (2,1)-W_0 (2,2)-W_1 (2,3)-W_2 (2,4)-W_0 (2,5)-W_1 (2,6)-W_2 (2,7)-W_0 (2,8)-W_1 (2,9)-W_2
(3,0)-W_1 (3,1)-W_2 (3,2)-W_0 (3,3)-W_1 (3,4)-W_2 (3,5)-W_0 (3,6)-W_1 (3,7)-W_2 (3,8)-W_0 (3,9)-W_1
(4,0)-W_1 (4,1)-W_2 (4,2)-W_0 (4,3)-W_1 (4,4)-W_2 (4,5)-W_0 (4,6)-W_1 (4,7)-W_2 (4,8)-W_0 (4,9)-W_1
(5,0)-W_2 (5,1)-W_1 (5,2)-W_0 (5,3)-W_2 (5,4)-W_1 (5,5)-W_0 (5,6)-W_2 (5,7)-W_1 (5,8)-W_0 (5,9)-W_2
(6,0)-W_2 (6,1)-W_0 (6,2)-W_1 (6,3)-W_2 (6,4)-W_0 (6,5)-W_1 (6,6)-W_2 (6,7)-W_0 (6,8)-W_1 (6,9)-W_2
(7,0)-W_2 (7,1)-W_0 (7,2)-W_1 (7,3)-W_2 (7,4)-W_0 (7,5)-W_1 (7,6)-W_2 (7,7)-W_0 (7,8)-W_1 (7,9)-W_2
(8,0)-W_1 (8,1)-W_0 (8,2)-W_2 (8,3)-W_1 (8,4)-W_0 (8,5)-W_2 (8,6)-W_1 (8,7)-W_0 (8,8)-W_2 (8,9)-W_1
(9,0)-W_2 (9,1)-W_0 (9,2)-W_2 (9,3)-W_1 (9,4)-W_0 (9,5)-W_2 (9,6)-W_1 (9,7)-W_0 (9,8)-W_2 (9,9)-W_1
Suma wyrazów tablicy równolegle: 913.500000
```

Na powyższym zrzucie ekranu widzimy wypis programu z użyciem klauzuli schedule typu dynamic z domyślną ilością porcji (= 1).

c. dekompozycja kolumnowa (atomic + schedule(static)):

```
#pragma omp parallel default(none) shared(suma_parallel, a) private(i, j)
      num_threads(NUM_THREADS)
   {
#pragma omp for ordered schedule(static)
       for (j = 0; j < WYMIAR; j++) {
           double suma_tmp = 0.0;
           for (i = 0; i < WYMIAR; i++) {
               suma_tmp += a[i][j];
#pragma omp ordered
               printf("(%1d,%1d)-W_%1d ", i, j, omp_get_thread_num());
           }
#pragma omp atomic
           suma_parallel += suma_tmp;
           printf("\n");
       }
   }
```

```
Suma wyrazów tablicy: 913.500000
(0,0)-W_0 (1,0)-W_0 (2,0)-W_0 (3,0)-W_0 (4,0)-W_0 (5,0)-W_0 (6,0)-W_0 (7,0)-W_0 (8,0)-W_0 (9,0)-W_0
(0,1)-W_0 (1,1)-W_0 (2,1)-W_0 (3,1)-W_0 (4,1)-W_0 (5,1)-W_0 (6,1)-W_0 (7,1)-W_0 (8,1)-W_0 (9,1)-W_0
(0,2)-W_0 (1,2)-W_0 (2,2)-W_0 (3,2)-W_0 (4,2)-W_0 (5,2)-W_0 (6,2)-W_0 (7,2)-W_0 (8,2)-W_0 (9,2)-W_0
(0,3)-W_0 (1,3)-W_0 (2,3)-W_0 (3,3)-W_0 (4,3)-W_0 (5,3)-W_0 (6,3)-W_0 (7,3)-W_0 (8,3)-W_0 (9,3)-W_0
(0,4)-W_1 (1,4)-W_1 (2,4)-W_1 (3,4)-W_1 (4,4)-W_1 (5,4)-W_1 (6,4)-W_1 (7,4)-W_1 (8,4)-W_1 (9,4)-W_1
(0,5)-W_1 (1,5)-W_1 (2,5)-W_1 (3,5)-W_1 (4,5)-W_1 (5,5)-W_1 (6,5)-W_1 (7,5)-W_1 (8,5)-W_1 (9,5)-W_1
(0,6)-W_1 (1,6)-W_1 (2,6)-W_1 (3,6)-W_1 (4,6)-W_1 (5,6)-W_1 (6,6)-W_1 (7,6)-W_1 (8,6)-W_1 (9,6)-W_1
(0,7)-W_2 (1,7)-W_2 (2,7)-W_2 (3,7)-W_2 (4,7)-W_2 (5,7)-W_2 (6,7)-W_2 (7,7)-W_2 (8,7)-W_2 (9,7)-W_2
(0,8)-W_2 (1,8)-W_2 (2,8)-W_2 (3,8)-W_2 (4,8)-W_2 (5,8)-W_2 (6,8)-W_2 (7,8)-W_2 (8,8)-W_2 (9,8)-W_2
(0,9)-W_2 (1,9)-W_2 (2,9)-W_2 (3,9)-W_2 (4,9)-W_2 (5,9)-W_2 (6,9)-W_2 (7,9)-W_2 (8,9)-W_2 (9,9)-W_2
Suma wyrazów tablicy równolegle: 913.500000
wigryz@msi-wigryz [13:17:50] [~/programming/studies/parallel-programming/lab_9/lab_9] [master *]
```

Dekompozycja kolumnowa z użyciem dyrektywy atomic.

Wnioski:

Używanie interfejsu OpenMP wydaje się być niewyobrażalnie pomocne. Za pomocą jednolinijkowych dyrektyw jesteśmy w stanie zapanować nad wielowątkowością w naszym programie. Nie znaczy to jednak, że interfejs ten wyręczy nas we wszystkim. Sam wykonując drugie ćwiczenie na zajęciach wpadłem na problem race condition, gdy nie oznaczyłem jednej ze zmiennych służących do iterowania jako private. Dobrym nawykiem może być ustawianie klauzuli “default(none)” co może pomóc w odkrywaniu takich błędów zanim jeszcze skompilujemy nasz program.