

Antoni Pawlak

401480

Laboratorium 10: OpenMP zmienne

Cel ćwiczenia

- Tworzenie programów równoległych przy użyciu OpenMP
- Wykorzystanie dyrektyw zmiennych
- Analiza wykonania programu

Wykonanie

Przygotowania

1. Utworzenie katalogu i pobranie plików ze strony
2. Skompilowanie kodu i uruchomienie programu

3. Poprawienie czytelności wydruku

```
#pragma omp critical
{
    ... // umieszczamy wydruki wewnątrz sekcji krytycznej
}
```

4. Naprawienie a_shared

```
#pragma omp critical // umieszczamy zmienną w sekcji krytycznej
for(i=0;i<10;i++){
    a_shared++; //WAW
}
```

5. Naprawienie e_atomic

```
for(i=0;i<10;i++){
    #pragma omp atomic //używamy dyrektywy atomic
    e_atomic+=omp_get_thread_num(); //WAW
}
```

6. Naprawianie d_local_private

```
int tmp_a = a_shared; // wprowadzamy zmienną tymczasową przed obszarem równ.
...
int d_local_private;
d_local_private = tmp_a + c_firstprivate; //WAR //używamy zmiennej tymczasowej
```

7. Weryfikacja poprawności programu

```
przed wejściem do obszaru równoległego - nr_threads 1, thread ID 0
  a_shared      = 1
  b_private     = 2
  c_firstprivate = 3
  e_atomic      = 5

w obszarze równoległym: aktualna liczba wątków 10, mój ID 9
  a_shared      = 101
  b_private     = 0
  c_firstprivate = 93
  d_local_private = 4
  e_atomic      = 455

w obszarze równoległym: aktualna liczba wątków 10, mój ID 1
  a_shared      = 101
  b_private     = 0
  c_firstprivate = 13
  d_local_private = 4
  e_atomic      = 455

w obszarze równoległym: aktualna liczba wątków 10, mój ID 3
  a_shared      = 101
  b_private     = 0
  c_firstprivate = 33
  d_local_private = 4
  e_atomic      = 455

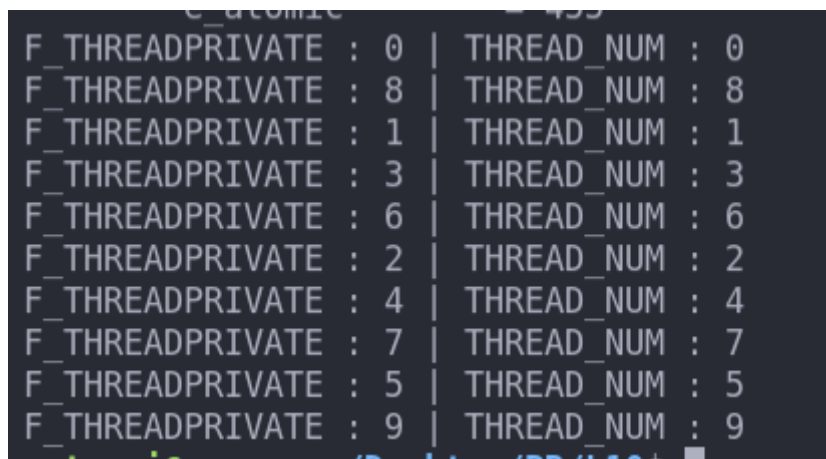
w obszarze równoległym: aktualna liczba wątków 10, mój ID 5
  a_shared      = 101
  b_private     = 0
  c_firstprivate = 53
  d_local_private = 4
  e_atomic      = 455

w obszarze równoległym: aktualna liczba wątków 10, mój ID 7
  a_shared      = 101
  b_private     = 0
  c_firstprivate = 73
  d_local_private = 4
  e_atomic      = 455
```

8. Napisanie drugiego obszaru równoległego

```
#pragma omp threadprivate(f_threadprivate) //ustalamy zmienną jako prywatną dla wątku
...
f_threadprivate = omp_get_thread_num();//zapisujemy numer do zmiennej
...
#pragma omp parallel default(none) num_threads(10) //drugi obszar równ.
{ // wyświetlamy wątki
printf("F_THREADPRIVATE : %d | THREAD_NUM : %d \n", f_threadprivate,
omp_get_thread_num());}
```

9. Weryfikacja poprawności

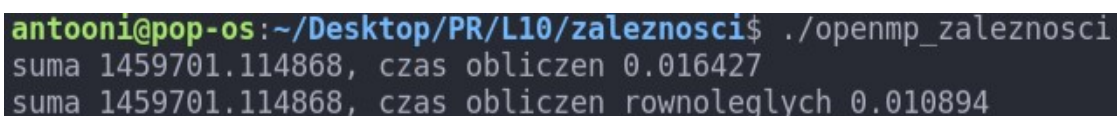


```
F_THREADPRIVATE : 0 | THREAD_NUM : 0
F_THREADPRIVATE : 8 | THREAD_NUM : 8
F_THREADPRIVATE : 1 | THREAD_NUM : 1
F_THREADPRIVATE : 3 | THREAD_NUM : 3
F_THREADPRIVATE : 6 | THREAD_NUM : 6
F_THREADPRIVATE : 2 | THREAD_NUM : 2
F_THREADPRIVATE : 4 | THREAD_NUM : 4
F_THREADPRIVATE : 7 | THREAD_NUM : 7
F_THREADPRIVATE : 5 | THREAD_NUM : 5
F_THREADPRIVATE : 9 | THREAD_NUM : 9
```

10. Zrównoleglenie petli poprzez wprowadzenie dodatkowej tablicy

```
double* A2 = malloc((N+2)*sizeof(double)); //utworzenie dodatkowej tablicy
for(i=0;i<N+2;i++) A2[i] = A[i]; //przepisanie wartości
// ustalenie A2 jako shared
#pragma omp parallel for default(none) num_threads(2) shared(A,B,A2) private(i)
for(i=0; i<N; i++){
#pragma omp atomic
A[i] += A2[i+2] + sin(B[i]);}
```

11. Uruchomienie programu za pomocą 2 wątków



```
antooni@pop-os: ~/Desktop/PR/L10/zalezności$ ./openmp_zalezności
suma 1459701.114868, czas obliczeń 0.016427
suma 1459701.114868, czas obliczeń równoległych 0.010894
```

Wnioski

RAW (Read After Write)

- Zależność rzeczywista
- Instrukcja druga próbuje odczytać zanim instrukcja 1 zdąży zapisać

```
i1. R2 <- R5 + R3
i2. R4 <- R2 + R3
```
- Nie da się przemianować

WAR (Write After Read)

- Anty zależność
- Instrukcja 2 zapisuje zanim 1 zdąży odczytać

```
i1. R4 <- R1 + R5
i2. R5 <- R1 + R2
```
- Rozwiązanie: wykorzystanie dodatkowej zmiennej

WAW (Write After Write)

- Anty zależność
- Jeden zapis może wyprzedzić drugi

```
i1. R4 <- R1 + R5
i2. R5 <- R1 + R2
```
- Rozwiązanie: wykorzystanie sekcji krytycznej lub zmiennej atomowej

Kroki do poprawnego zrównoleglania pętli

- Wykrycie zależności
- Przemianowanie anty-zależności
- Zmiana struktury w celu pozbycia się zależności

Wniosek

Równoległe programy niosą ze sobą pewne problemy, zależności. Zadaniem programisty jest identyfikacja takich miejsc oraz ich poprawienie.