|  |  |  |
| --- | --- | --- |
| Łukasz Oprych  Gr. Lab. 5  Informatyka Techniczna | T: OpenMP Zmienne | 15.12.2023r. |

Cel ćwiczenia:

Dalsze zapoznawanie się z pisaniem programów równoległych w OpenMP, zależnościami oraz wykorzystywaniem dyrektyw i klauzul.

Przebieg ćwiczenia:

Po przygotowaniu programu zgodnie z instrukcją prowadzącego wywołano domyślny program openmp\_watki\_zmienne dla ustawionych niedomyślnie 6 wątków:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Jak widać wynik, zmienne dla wątku ID3 wypisane wypisały się w wątku ID1, co jest niepożądanym zjawiskiem.

W celu poprawienia czytelności wydruku zastosowano dyrektywy bezpośrednio przed wypisaniem wyników poprzez dodanie bariery, aby przed wypisaniem wszystkie wątki wykonały swoje obliczenia, oraz dyrektywę critical, aby kod był wykonywany pojedynczo dla każdego wątku:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Jak widać zmienne wyświetlane są już prawidłowo dla każdego wątku:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Kolejnym etapem było poprawienie wypisywania wartości zmiennych globalnych, w tym wypadku a\_shared i e\_atomic, ponieważ przy każdym wywołaniu zmienne zmieniały swoją wartość:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Dodano klauzulę critical dla zmiennej a\_shared w celu ochrony sekcji krytycznej, czyli pętli z a\_shared.

Dodano klazulę atomic w pętli ze zmienną e\_atomic w celu zapewnienia poprawności działania przez użycie operacji atomowych.

Dodano klauzulę barrier, aby to co już zostało nadpisanie nie było zmienione przez inny wątek wykonujące.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W 33 linii za pomocą bariery eliminujemy zależność WAR

W 36 linii zależność WAR ze względu na zmienną a\_shared

W 45 linii zależność RAW i WAR ze względu na e\_atomic



W linii 34 zależność RAW i WAR ze względu na a\_shared

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W 58 i 62 linii zależność RAW ze względu na zmienne a\_shared i e\_atomic

Jak widać wartości zmiennych współdzielonych nie zmieniają się po kolejnym wykonaniu programu:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Kolejnym poleceniem było utworzenie kolejnego obszaru równoległego oraz przetestowanie w nim dyrektywy threadprivate poprzez utworzenie zmiennej f\_threadprivate oraz dodanie jej w pierwszym obszarze równoległym, która ma zachować swoją wartość w obu obszarach równoległych.

Definicja zmiennej:

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Umieszczenie zmiennej w pierwszym obszarze równoległym:



Zależność RAW i WAR w linii 34 ze względu na zmienną a\_shared.

Utworzenie drugiego obszaru równoległego, w którym zostaje wypisana wartość zmiennej f\_threadprivate:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wynik dla 5 wątków zdefiniowanych za pomocą funkcji bibliotecznej omp\_set\_num\_threads(5):

Obraz zawierający tekst, zrzut ekranu, menu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jak widać zmienne threadprivate zachowały swoje wartości dla poszczególnych wątków w drugim obszarze równoległym.

Następnie skonfigurowano program openmp\_zaleznosci i uruchomiono.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jak widać obliczenia sekwencyjne i równoległe różnią się wynikiem.

Analiza:

W wersji sekwencyjnej programu nie ma możliwości zmiany wartości elementów kolejnych zanim nie policzymy poprzednich.

W równoległej wersji jeżeli weźmiemy pod uwagę pierwsze 2 watki, to zakładając, ze pierwszy watek dostanie elementy 0 i 1 tablicy, drugi wątek dostanie elementy 2 i 3 tablicy. Problem pojawia się, gdy drugi watek wykonuje swoje instrukcje jako pierwszy, ponieważ obliczy on z podanego wzoru wartości 2 i 3 elementu w tablicy A, a dopiero po nim wątek pierwszy będzie liczył 0 i 1 element w tablicy, wówczas w oparciu o zmienione wartości tablicy A. W linii 37 zachodzi zależność WAR, A[i] odczytuje dane, do zapisu danych dochodzi w całej operacji A[i] += A[i+2] + sin(B[i]).

Wersja równoległa:

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

Modyfikacja:

Dodanie tablicy C



Uzupełnienie wersji równoległej:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Wnioski:

W powyższym ćwiczeniu zapoznano się z zależnościami danych RAW (Read After Write, zależność rzeczywista) oraz WAR (Write after Read, antyzależność). W RAW przykładowo, gdy mamy 2 instrukcje, instrukcja 2 próbuje odczytać dane zanim instrukcja 1 zdąży zapisać dane. Zależności rzeczywiste uniemożliwiają nam zrównoleglenie algorytmu, w celu poprawienia rozwiązania należy zmodyfikować kod, aby uzyskać wersję pozwalającą na zrównoleglenie. W WAR ponownie na przykładzie 2 instrukcji, instrukcja 2 zapisuje dane zanim instrukcja 1 zanim zdąży odczytać dane.W przypadku antyzależności rozwiązaniem problemu będzie przemianowanie zmiennych, wykorzystanie dodatkowej zmiennej.