Przetwarzanie współbieżne. Programowanie równoległe i rozproszone
Laboratorium 5

Cel: Nabycie umiejętności tworzenia i implementacji programów równoległych w środowisku Pthreads

Zajęcia:

1. Utworzenie katalogu roboczego (np. lab_5), skopiowanie ze strony przedmiotu paczki pthreads_suma.tgz, rozpakowanie w podkatalogu (np. zad_1), uruchomienie i przetestowanie działania dla różnej liczby watków (w miejsce dostarczonej biblioteki libpomiar_czasu.a należy umieścić własną bibliotekę utworzoną w ramach laboratorium „pomiar_czasu") - sprawdzenie poprawności wyniku i czasu działania (wnioski powinny znaleźć się w sprawozdaniu)

2. Opracowanie programu wykorzystującego bibliotekę wątków Pthreads i obliczającego na kilka sposobów przybliżoną wartość całki oznaczonej z zadanej funkcji metodą trapezów:

a) program powinien zawierać funkcję obliczającą całkę sekwencyjnie (można użyć np. kod:

```
x1 = a; c=0.0; dx = (b-a)/N;
for(i=0;i<N;i++){
  x2=x1+dx;  c+=0.5*(f(x1)+f(x2))*dx; x1=x2;
}
```

uwaga: w kodzie każda wartość funkcji (z wyjątkiem skrajnych) jest liczona dwa razy, a mnożenie przez stałą podstawę odbywa się wielokrotnie w sumowaniu, czy można zmodyfikować kod, tak żeby liczyć każdą wartość tylko raz, a dodatkowo zmniejszyć liczbę operacji wyciągając mnożenie przed sumowanie?)

b) poza wariantem obliczania całki sekwencyjnie (w celu sprawdzenia poprawności kodu równoległego) program powinien zawierać dwa warianty równoległego obliczania całki: pierwszy – realizujący zrównoleglenie pętli i drugi – związany z dekompozycją w dziedzinie problemu

- w wariancie pierwszym każdy wątek otrzymuje swój identyfikator, a dekompozycja odbywa się w sposób analogiczny jak w przykładzie sumowania w pthreads_suma.c; ostateczny wynik jest uzyskiwany przez redukcję wielowątkową (uwaga na poprawne stosowanie zmiennych prywatnych i wspólnych)

- w pierwszym wariancie można skorzystać z pliku pthreads_suma.c – należy (najlepiej po skopiowaniu do nowego pliku pthreads_calka.c) zamienić procedury suma_w... wykonywane przez utworzone wątki na procedury (np. o analogicznych nazwach calka_w...) realizujące równoległe obliczanie całki, pozostawiając schemat tworzenia wątków i uzyskiwania końcowego wyniku bez zmian

- w wariancie drugim każdy wątek otrzymuje jako daną wejściową strukturę zawierającą parametry sobie przydzielonego przedziału – struktura powinna także zawierać pole do wpisania uzyskanego lokalnie wyniku; ostateczny wyniki jest uzyskiwany przez watek główny – jednowątkowo

3. Uruchomienie procedury sekwencyjnej; sprawdzenie poprawności otrzymanego wyniku dla różnych dokładności zadawanych przez użytkownika (poprzez wielkość podstawy trapezów czyli także całkowitą liczbę trapezów w zadanym (globalnym) przedziale)

- należy zanalizować jaka jest dokładność całkowania (podstawa trapezów), dla każdego z wariantów całkowania – **rezultaty opisać we wnioskach sprawozdania**

4. W przypadku zrównoleglenia pętli opracowanie sposobu umożliwiającego równomierny podział iteracji między wątki. Jak rozwiązać problem, w przypadku gdy liczba trapezów jest niepodzielna przez liczbę wątków (nie zmieniając granic całkowania i nie zaniżając dokładności numerycznej)?

5. Uruchomienie programu dla różnych liczb wątków; sprawdzenie poprawności działania.

Dalsze kroki dla podniesienia oceny:

1. Dodanie procedur pomiaru czasu i uzyskanie czasu wykonania dla 1,2,3,...,8 wątków oraz dla dwóch rozmiarów zadań: n=10000 (10^4) i n=10000000 (10^7). Stworzenie, dla obu rozmiarów zadania, wykresów zależności czasu wykonania od liczby wątków

2. Wykorzystanie zamiast zapisu do globalnej tablicy wyników, argumentu funkcji pthread_exit, który następnie odczytywany jest przez pthread_join (uwaga na rzutowanie typów).

Warunki zaliczenia:

Obecność na zajęciach i wykonanie kroków 1-5.

Oddanie sprawozdania z opisem zadania, kodem źródłowym programów i analizą wyników dla wszystkich wariantów oraz odpowiednimi wnioskami.