|  |  |
| --- | --- |
| Akademia Nauk Stosowanych w Nowym Sączu  Wydział Nauk Inżynieryjnych | |
| Imię i nazwisko: | Michał Bernardy |
| Grupa: | P1 |
| Ocena: |  |

**1. Opis problemu**

W niniejszym sprawozdaniu przedstawiono implementację równoległego algorytmu obliczania całki oznaczonej metodą prostokątów z wykorzystaniem biblioteki **MPI (Message Passing Interface)**. Przykładem jest całkowanie funkcji sin(x) na przedziale [0,π][0, \pi][0,π], której dokładna wartość wynosi **2.0**.

Całkowanie numeryczne to fundamentalny problem w analizie numerycznej, szczególnie istotny w przypadkach, gdy:

* całka nie ma rozwiązania analitycznego,
* lub jego uzyskanie jest trudne.

**Metoda prostokątów** to jedna z najprostszych metod przybliżania całki oznaczonej. Polega na:

* podziale przedziału całkowania na małe podprzedziały,
* aproksymacji pola pod wykresem funkcji za pomocą sumy pól prostokątów.

Ten problem doskonale nadaje się do **równoległego przetwarzania**, ponieważ:

* obliczenia dla poszczególnych podprzedziałów są niezależne,
* a ich wyniki mogą zostać zsumowane do uzyskania ostatecznej wartości.

**2. Implementacja rozwiązania**

**2.1 Metoda prostokątów**

W zaimplementowanym rozwiązaniu zastosowano następujące kroki:

* Przedział całkowania [0,π][0, \pi][0,π] podzielono na **n = 1 000 000** równych podprzedziałów.
* Dla każdego z nich obliczono pole prostokąta o:
  + szerokości: h=b−anh = \frac{b-a}{n}h=nb−a​,
  + wysokości: równej wartości funkcji w **lewym końcu** podprzedziału.
* Suma wszystkich pól daje przybliżenie wartości całki.

**2.2 Równoległa implementacja w MPI**

Kluczowe elementy implementacji:

1. **Podział zadania** – liczba prostokątów jest dzielona między procesy. Reszta przydzielana jest procesom o niższych rangach.
2. **Lokalne obliczenia** – każdy proces oblicza część całki dla przypisanych mu podprzedziałów.
3. **Agregacja wyników** – wykorzystano funkcję kolektywną MPI\_Reduce do zsumowania częściowych wyników.
4. **Pomiar czasu** – wykorzystano MPI\_Wtime() dla precyzyjnego pomiaru czasu wykonania.

**3. Wyniki eksperymentów**

**Wyniki z różnych konfiguracji:**

**Dla 4 procesów:**

Przybliżona wartość całki: 1.999999999998359

Dokładna wartość całki sin(x) od 0 do π: 2.000000000000000

Błąd: 0.000000000001641

Czas wykonania: 0.0114606 s

**Dla 13 procesów:**

Przybliżona wartość całki: 1.999999999998352

Dokładna wartość całki sin(x) od 0 do π: 2.000000000000000

Błąd: 0.000000000001648

Czas wykonania: 0.0213425 s

**Dla 100 procesów:**

Przybliżona wartość całki: 1.999999999998355

Dokładna wartość całki sin(x) od 0 do π: 2.000000000000000

Błąd: 0.000000000001645

Czas wykonania: 0.0868805 s

**4. Analiza wyników**

**1. Dokładność obliczeń**

* Uzyskano **bardzo dużą dokładność** (błąd rzędu 10−1210^{-12}10−12).
* Dokładność nie zależy od liczby procesów — algorytm wykonuje te same obliczenia.

**2. Wydajność obliczeniowa**

* Najlepszy czas uzyskano dla **4 procesów**.
* Przy 13 procesach czas wzrósł o ok. **86%**.
* Przy 100 procesach czas wzrósł ponad **7-krotnie** względem wersji 4-procesowej.

**3. Efektywność zrównoleglenia**

* Przy dużej liczbie procesów pojawia się zjawisko **spowolnienia równoległego** (parallel slowdown).
* Narzut komunikacyjny zaczyna dominować nad korzyściami z podziału pracy.

**5. Wnioski**

1. **Efektywne wykorzystanie MPI**  
   MPI umożliwia efektywną implementację algorytmów równoległych, m.in. dzięki MPI\_Reduce.
2. **Optymalna liczba procesów**  
   Dla rozmiaru danych n = 1 000 000 najlepszą wydajność uzyskano dla **4 procesów**.
3. **Skalowanie problemu**  
   Dla większych wartości n optimum może przesunąć się w stronę większej liczby procesów.
4. **Narzut komunikacyjny**  
   Wzrost liczby procesów skutkuje wzrostem narzutu komunikacyjnego — szczególnie widoczne przy **100 procesach**, gdzie każdy proces wykonuje relatywnie niewielką pracę.
5. **Zastosowania praktyczne**  
   Metoda może być użyta do całkowania dowolnych funkcji — zwłaszcza:
   * kosztownych obliczeniowo,
   * lub z bardzo dużym przedziałem całkowania.

Równoległa implementacja obliczania całki numerycznej skutecznie ilustruje zarówno **potencjał**, jak i **ograniczenia** programowania równoległego. Kluczem do sukcesu jest odpowiedni **balans pomiędzy podziałem pracy a narzutem komunikacyjnym**.