Aplikacja równoległa MPI - Mnożenie macierzy - algorytm Cannon'a

Łukasz Radzik, Konrad Walas

SRIR 2022

Opis Budowy

Program znajduje się w jednym pliku parallel_matrix_multiplication.c. Program używa jednego procesu głównego do obsługi ładowania danych, oraz składania obliczonych podmacierzy w całość oraz procesów slave, które równolegle obliczają poszczególne macierze. Składa się on z 5 części:

1. **Inicjalizacja**
   Na tym etapie przygotowywane są zmienne, alokowana jest pamięć dla macierzy, oraz wywoływane są funkcje inicjalizujące biblioteki MPI.

2. **Wczytanie danych oraz przesłanie ich do procesów**
   Na tym etapie wczytywane są dane z plików (A.txt, B.txt) i ładowane są do macierzy A oraz B. Etap ten odbywa sie tylko w procesie głównym. Następnie dane wysyłane są do procesów slave, które liczyć będą poszczególne części tych macierzy.

3. **Obliczenie wartości każdego bloku**
   W tym miejscu obliczane są poszczególne podmacierze. Każdy proces oblicza osobną podmacierz w sposób równoległy. Następnie obliczone macierze przesyłane są z powrotem do procesu głównego. Etap ten wykonywany jest jedynie w procesach slave.

4. **Złączenie poszczególnych bloków w wynikową macierz**
   Etap ten wykonywany jest w procesie głównym. Odbierane są poszczególne podmacierze, a następne są one łączone w jedna macierz wynikową. Na końcu wynik wypisywany jest na standardowe wyjście.

5. **Finalizacja**
   Zwalniana jest pamięć, oraz wykonywane są funkcje finalizujące z biblioteki MPI.

Opis działania

W programie wykorzystywany jest algorytm Cannona. Polega on na podziale macierzy na podmacierze oraz serii przesunięć, dzięki którym poszczególne procesy mogą policzyć mniejsze bloki niezależnie od siebie. Następnie podmacierze zostają połączone w macierz wynikową.

Algorytm:

1. Consider two n × n matrices A and B partitioned into p blocks.
2. A(i , j) and B(i , j) ($0 \leq i,j \leq \sqrt{p}$) of size $(n/\sqrt{p})\times(n/\sqrt{p})$ each.
3. Process P(i , j) initially stores A(i , j) and B(i , j) computes block C(i , j) of the result matrix.
4. The initial step of the algorithm regards the alignment of the matrixes.
5. Align the blocks of A and B in such a way that each process can independently start multiplying its local submatrices.
6. This is done by shifting all submatrices A(i , j) to the left (with wraparound) by i steps and all submatrices B(i , j) up (with wraparound) by j steps.
7. Perform local block multiplication.
8. Each block of A moves one step left and each block of B moves one step up (again with wraparound).
9. Perform next block multiplication, add to partial result, repeat until all blocks have been multiplied.

Pseudokod algorytmu:

```
forall i=0 to √p -1
    CShift-left A [i; :] by i
forall j=0 to √p -1
    Cshift-up B[: , j] by j
for k=0 to √p -1
    forall i=0 to √p -1 and j=0 to √p -1
        C[i,j] += A[i,j] * B[i,j]
        CShift-leftA[i; :] by 1
        Cshift-up B[: , j] by 1
    end forall
end for
```

Sposób obsługi programu

Wraz z programem dostarczony jest również plik Makefile. Ma on dostępne następujące komendy:

- make all - służy do kompilacji programu
- make nodes - służy do wygenerowania listy dostępnych nodów
- make run - służy do uruchomienia skompilowanego programu używając pliku nodes
- make runall - służy do uruchomienia wszystkich powyższych komend w odpowiedniej kolejności
- make clean - służy do przywrócenia zawartości katalogu do stanu wyjściowego