MPI - Raport

Władek Pałucki
wp418359@students.mimuw.edu.pl

28 listopada 2024

1 Kontrola programu

Mój program nie odbiega zbyt bardzo od tego co zostało opisane w podanej pracy. Na początku wczytuję dane wejściowe i obliczam optymalne wartości `p_n, p_m, p_k`. Następnie dzielę macierz na kawałki przydzielone do poszczególnych *k-task groups*. W obrębie każdej grupy dzielę przydzielony kawałek macierzy bardziej, tak żeby każdy proces miał jakiś kawałek macierzy do wygenerowania. Następnie każdy proces generuje swój kawałek macierzy `A` i `B`. Jeżeli `c`> 1 (czyli `p_n` $>$ `p_m` oznacza to, grupy Cannona w obrębie *k-task group* muszą się powymieniać swoimi kawałkami macierzy, co jest zaimplementowane za pomocą operacji `MPI_Allreduce`. Po tej operacji każdy proces ma już wszystkie potrzebne mu kawałki macierzy i rozpoczynany jest algorytm Cannona. Następnie, jeżeli `p_k` > 1, czyli jest więcej niż jedna *k-task group*, to grupy muszą zsumować wyniki tak, żeby dostać pełne macierze, tutaj znowu używam operacji `MPI_Allreduce`. Na koniec, zależnie od parametrów wejściowych cała macierz jest wypisywana, albo każdy proces zlicza wystąpienia elementów większych niż podana wartość w swoim, oddzielnym kawałku macierzy, a potem przy pomocy `MPI_Reduce` te informacje są sumowane w procesie 0.

2 Zastosowane Optymalizacje

- Generowanie macierzy `A` i `B`
  Macierze `A` i `B` generowane są przy pomocy wszystkich procesów. Dzięki temu faza generowania jest bardzo szybka. Ponieważ każdy proces generuje podmacierz macierzy której ostatecznie będzie potrzebował do obliczeń, aby zebrać całą macierz każdy proces musi porozumieć się tylko z podzbiorem innych procesów, które będą potrzebowały tych samych kawałków macierzy `A` i `B` a nie ze wszystkimi procesami.

- Padding
  Najprostszym rozwiązaniem tego, że wymiary macierzy nie muszą być podzielne przez `p_n, p_m, p_k` byłoby doklejenie do oryginalnych macierzy zer, tak, aby zawsze ich wymiary były podzielne. Mogłoby to jednak powodować znacznie większe zużycie pamięci. W moim programie wyliczam, podział macierzy na procesy tak, że nie jest to konieczne, co wiąże się z dużą oszczędnością pamięci (Procesy na samym "dole"i z

"prawej"strony macierzy mogą potrzebować paddingu w swoich kawałkach macierzy, ale jest on mniejszy niż ten który by powstał, przy paddowaniu całej macierzy).

- Komunikacja podczas algorytmu Cannona
  W moim programie procesy w każdej grupie Cannona są połączone przy użyciu komunikatora `MPI_Cart`. Dzięki temu procesy łatwo mogą znajdować swoich sąsiadów w Cannonie. Podczas samego wykonywania algorytmu wykorzystuję asynchroniczne wiadomości. Proces jednocześnie wysyła swoje macierze do sąsiadów, odbiera nowe macierze i oblicza wynik z macierz które miał w danym kroku.