|  |  |  |
| --- | --- | --- |
| Łukasz Oprych  Gr. Lab. 5  Informatyka Techniczna | Podstawy komunikacji MPI | Data  23.12.2023 |

Cel ćwiczenia:

Zapoznanie się z podstawowym przesyłaniem komunikatów, składnią, kompilacją MPI.

Przebieg ćwiczenia:

Po przygotowaniu środowiska, struktury katalogowej, instalacji paczki mpicc zgodnie z poleceniem prowadzącego, przystąpiono do wykonania programu MPI\_Simple.c

Zdefiniowanie liczby procesów w Makefile, w tym przypadku 6:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Utworzenie tablicy hostname w celu przesłania do kolejnych procesów adresu nadawcy oraz receive\_hostname w celu odebrania przez proces owego adresu. Użyto funkcji gethostname() do pobrania adresu nadawcy.

Użyto MPI\_Send do przesłania ID procesu, oraz hostname’a który składa się z m.in. jego rozmiaru, typu zmiennej, docelowego procesu, oznaczenia wiadomości, komunikatora COMM\_WORLD

Użyto MPI\_Recv do odebrania ID procesu oraz hostname’a, który składa się z m.in. jego rozmiaru (tablica o wymiarze 100, ID o wymiarze 1), typu zmiennej, docelowego procesu, tagu wiadomości, aby rozróżnić procesy w przypadku przesłania więcej niż jednej wiadomości, komunikatora oraz statusu.

Na koniec wypisano ID procesu oraz hostname nadawcy.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Wynik:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Następnie utworzono katalog o nazwie sztafeta i utworzono plik sztafeta.c, gdzie skopiowano początkowo kod z pliku MPI\_simple.c i dostosowano do polecenia:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Początkowo program zaczyna się od inicjowania MPI oraz pobrania liczby procesów za pomocą MPI\_Comm\_Size oraz własnego identyfikatora (rangi) dla poszczególnego procesu.

W celu identyfikacji procesów użyliśmy zmiennej rank odpowiadającej za rangę procesu. Dostępne procesy podzielono na 3 grupy, proces początkowy, procesy środkowe oraz proces końcowy. Proces początkowy nr 0 odpowiada w tym przypadku jedynie za przesłanie zmiennej send\_value o nadanej wartości 10 do procesu nr 1 używając MPI\_Send. W skład procesów środkowych wchodzą wszystkie wątki poza początkowym i ostatnim, te procesy kolejno odbierają wartość zmiennej receive\_value od poprzedniego za pomocą MPI\_Recv, podbijają jej wartość o 1 i przesyłają dalej do kolejnego procesu przy użyciu MPI\_Send. Proces końcowy jedynie odbiera wartość receive\_value za pomocą MPI\_Recv, ostatni proces kończy sztafetę.

Wynik programu sztafeta:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wnioski:

Na podstawie programu sztafeta.c możemy zauważyć standardową komunikację typu punkt-punkt, która odnosi się do bezpośredniej wymiany informacji między parą konkretnych procesów. MPI\_Send oraz MPI\_Recv są funkcjami o blokującej charakterystyce polecenia. Program wstrzymuje wykonanie aż do wysłania/odebrania wiadomości, pomaga to nam łatwo obsługiwać synchronizację i sterować przebiegiem programu. Nieblokującymi zastępnikami owych funkcji są MPI\_Isend oraz MPI\_Irecv, które natychmiast przekazują sterowanie dalszym instrukcjom programu, pozwala to na komunikację asynchroniczną i program może być wykonywany natychmiastowo. Można również stosować mieszaną kombinację blokujących i nieblokujących procedur. Zależnie od postawionego przed nami rodzaju problemu oba rodzaje funkcji będą mieć swoje zastosowanie.