Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Laboratorium 11.

Cel:

Opanowanie podstaw programowania z przesyłaniem komunikatów MPI.

Kroki:

1. Utworzenie katalogu roboczego (np. *lab11*) i podkatalogu (np. *simple*)
2. Skopiowanie paczki *MPI_simple.tgz*, rozpakowanie, modyfikacja pliku *Makefile* (jeśli konieczne, np. podstawienie właściwych ścieżek), kompilacja oraz uruchomienie – stosując różne liczby procesów (liczba musi być większa od 1)
   - zmianę liczby procesów można wprowadzać do Makefile (i uruchamiać przez *make run*) lub jawnie wywoływać właściwy *mpiexec* z odpowiednimi parametrami
3. Uzupełnienie programu (pozostawiając przesyłanie rang) o przesyłanie w tablicy znaków adresu internetowego węzła nadawcy i wypisywanie adresu u odbiorcy (adres można pobrać funkcją *gethostname*). **(ocena)**
4. Utworzenie podkatalogu (np. *sztafeta*)
5. Opracowanie (np. na podstawie kodu z p.2) programu propagującego komunikaty w konwencji pierścienia (sztafeta). Każdy proces ustala jednego poprzednika (od którego otrzymuje komunikaty, oraz jednego następcę, któremu przekazuje komunikaty (można w tym celu wykorzystać rangi procesów - rangę procesu, od którego odbiera komunikaty w ramach sztafety (pierścienia) proces może zapisać w zmiennej *prev*, a rangę procesu, któremu wysyła komunikaty w zmiennej *next*).
   1. Program rozpoczyna się, jak prawie każdy program w MPI, od inicjowania MPI oraz pobrania liczby procesów (rozmiaru komunikatora) i własnego identyfikatora (rangi)
   2. Program napisany jest w konwencji SPMD, należy utworzyć jeden kod, który będzie wykonywany przez wszystkie uruchomione procesy. Każdy proces na podstawie swoje rangi ustala jakie instrukcje ma wykonać. Pierwszym krokiem projektowania programu powinno być zastanowienie się nad następującą kwestią: jestem procesem o randze "rank" - jakie operacje muszę wykonać w ramach sztafety. Następnie można rozważyć pytanie czy wszystkie procesy wykonują dokładnie te same operacje. Ostatnim krokiem jest rozpisanie za pomocą odpowiednich instrukcji *if* , operacji dla każdego z procesów.
6. Komunikaty mogą zawierać dane, które są inicjowane przez rozpoczynający sztafetę proces o randze 0, a następnie modyfikowane (np. przez zwiększenie o 1) przez każdy kolejny proces – taka zawartość ułatwia sprawdzenie poprawności działania sztafety. Należy rozważyć dwa warianty:
   1. ostatni proces przesyła dane do procesu pierwszego (zamknięty pierścień)
   2. ostatni proces kończy sztafetę
7. Uruchomienie obu wariantów programu. Każdorazowo program powinien wyświetlić podstawowe informacje zgodnie z wzorem:

```
1. …
2. Proces ... odebrał liczbę ... od procesu ...
3. Proces ... wysłał liczbę ... do procesu ...
4. …
5. itp. (ocena)
```

8. Utworzenie podkatalogu roboczego (np. *struktura*).
9. Zaprojektowanie "bogatej" struktury danych języka C – każdy powinien zaprojektować indywidualną strukturę zawierającą co najmniej 3 typy danych, w tym co najmniej 1 tablicę znaków, w której przechowywane jest własne imię – tablica powinna mieć odpowiednio dobraną długość i przesyłane powinny być tylko znaki zawierające imię, plus ewentualnie znak końca napisu \0)
10. Napisanie programu równoległego (np. wykorzystując kod z p. 5), w którym dokonuje się przesłania zaprojektowanej struktury między procesami wykorzystując spakowany typ danych MPI (MPI_PACKED)
    - każdy proces uczestniczący w przesyłaniu powinien wypisać zawartość otrzymanej struktury, zmodyfikować zawartość i wypisać zawartość struktury wysyłanej do następnego procesu
11. Uruchomienie programu w środowisku MPI i przetestowanie poprawności działania **(ocena)**

Dalsze kroki dla podniesienia oceny:

1. Rozwinięcie programu tak, żeby realizować schemat przetwarzania potokowego
    1. Proces o randze 0 tworzy zbiór (tablicę, listę) struktur indywidualnie zaprojektowanego typu
        - można np. wykorzystać poprzednią strukturę lub strukturę, która będzie zawierała napis (jako tablicę znaków zakończoną \0) oraz kilka pól, takich jak np. liczba znaków w napisie, średnia długość słowa, liczba wystąpień 'a' (lub cały histogram), itp.
    2. Proces o randze 0 wysyła kolejne struktury do swojego następnika, ten do następnego procesu itd. (jak w schemacie sztafety)
    3. Każdy proces realizuję własne, indywidualnie zaprojektowane przetwarzanie danych w strukturze, np. obliczenie liczby znaków w napisie, zamianę wszystkich małych liter na duże lub na odwrót, itp.
    4. W efekcie każda struktura przechodzi przez szereg etapów przetwarzania, realizowanych przez kolejne procesy. W każdej chwili czasu przetwarzanych (i przesyłanych) jest wiele struktur, realizując równoległe przetwarzanie potokowe.
    5. Możliwe rozwiązanie z punktu widzenia pojedynczego procesu (procesy początkowy i końcowy mają swoją specyfikę):
        1. funkcje odbierania i wysyłania są realizowane w (nieskończonej) pętli po komunikatach
            - zakończenie przetwarzania (np. poprzez break) może następować kiedy przesłana struktura będzie zawierać specyficzną zawartość
        2. dla każdego komunikatu proces: odbiera go, modyfikuje na swój sposób i przesyła dalej
        3. modyfikacja jest realizowana na zasadzie SPMD – każdy proces wie jaką modyfikację ma wykonać na podstawie swojego ID (swojej rangi)
        4. tym razem kod dla każdego procesu jest trochę inny, każdy realizuje inne przetwarzanie
    6. Alternatywą sekwencyjną (do ewentualnego porównywania wydajności) jest realizacja przetwarzania przez jeden proces w podwójnej pętli – zewnętrznej po komunikatach, wewnętrznej po etapach przetwarzania (liczba etapów musi być równa liczbie procesów w wersji równoległej – w wersji sekwencyjnej zamiast pętli wewnętrznej (gdzie indeks będzie pełnił rolę rangi procesu z wersji równoległej) można także użyć sekwencji wywołań funkcji realizujących przetwarzanie)

---------- 4.0 ----------

2. Na podstawie slajdów z wykładu (W12, 14-16), opracowanie przesyłania struktury języka C za pomocą nowego typu danych MPI utworzonego specjalnie dla zaprojektowanej struktury przy użyciu funkcji *MPI_Type_create_struct* **(ocena)**

Warunki zaliczenia:

1.Obecność na zajęciach i wykonanie kroków 1-11.

2.Oddanie standardowego sprawozdania – zgodnie z regulaminem laboratoriów