do zdobycia: [30pkt]
czas realizacji: 2 tygodnie

Zestaw 5

Celem ćwiczenia jest stworzenie programu stanowiącego realizację farmy zadań (ang. task farm). Powstały program powinien działać w sposób następujący:

1. program powinien pobierać 3 parametry (wektor argumentów), są to kolejno:
   (a) całkowita liczba zadań do wykonania n_{tasks},
   (b) maksymalny rozmiar zadania t_{max},
   (c) ziarno s służące do inicjalizacji generatora liczb pseudolosowych,
2. wyodrębiony zostaje nadzorca (proces o randze 0) oraz robotnicy (pozostałe procesy),
3. w etapie dalszym nadzorca tworzy zbiór zadań do wykonania, w tym celu generuje n_{tasks} losowych liczb całkowitych (każda z zakresu od 1 do t_{max}) zapamiętując je, liczby te powinny być generowane zgodnie z rozkładem jednostajnym,
4. w kolejnym etapie następuje rozpoczęcie pracy, w szczególności:
   (a) nadzorca wysyła po jednym zadaniu do każdego z $n-1$ robotników (n oznacza całkowitą liczbę procesów), poprzez wysłanie zadania należy rozumieć wysłanie jednej z losowo wygenerowanych (na wstępie) liczb,
   (b) każdy z robotników odbiera, a następnie przetwarza swoje zadanie,
   (c) robotnik otrzymując zadanie o wartości t powinien:
      i. obliczyć wartość wyrażenia
      $$r = \frac{t!}{\sum_{i=1}^{t} i},$$
      ii. „odespać” t milisekund, odpoczywając po wysiłku związanym z obliczeniem wyjątkowo złożonego wyrażenia (czytaj: odczekać t milisekund nie robiąc nic),
      iii. odesłać do nadzorcy wynik realizacji zadania r,
   (d) po rozesłaniu wstępnych $n-1$ zadań nadzorca przechodzi w tryb „nasłuchu”, obserwuje czy któryś z robotników nie zgłosił faktu zakończenia realizacji zadania, jeżeli tak jest, odbiera od robotnika wynik realizacji zadania i przydziela mu nowe zadanie (jeżeli jest to możliwe, tj. jeżeli pula nieprzetworzonych zadań nie jest pusta),
   (e) robotnicy, każdorazowo po zakończeniu realizacji zadania, zgłaszają ten fakt nadzorcy (odsyłając również wynik jego realizacji), oczekując na przydzielenie nowego zadania,
   (f) nadzorca widząc, że pula nieprzetworzonych zadań jest pusta wysyła do wszystkich robotników komunikat o końcu pracy, nadzorca oraz robotnicy kończą pracę.
5. po zakończeniu pracy nadzorca zapisuje do pliku wyjściowego (teksowego) wszystkie wyniki przetwarzania, wyniki te powinny być zapisane w formacie kolumnowym, w każdej linii powinny być zawarte dwie liczby, jedna całkowita t (zadnie) oraz jedna rzeczywista r (wynik zadania), plik powinien zawierać dokładnie n_{tasks} linii, kolejność pojawiania się zadań powinna ściśle odpowiadać kolejności w jakiej zostały one wygenerowane,

6. dodatkowo każdy z robotników powinien w trakcie pracy obliczać całkowity czas „snu” (całkowitą nakład włożonej pracy), każdy z robotników powinien również obliczyć całkowitą liczbę przetworzonych zadań, wartości te (całkowity nakład włożonej pracy, całkowita liczba przetworzonych zadań) powinny zostać po zakończeniu wszystkich prac przesłane do nadzorcy, nadzorca powinien wypisać na ekranie (w sposób sformatowany) te wartości.

Uwagi/zastrzeżenia/wskazówki:

1. każde zadanie powinno zostać przetworzone dokładnie raz,
2. program powinien sprawdzać poprawność przekazanych argumentów,
3. należy założyć, że $n_{tasks} >= n$, gdzie $n \geqslant 2$ oznacza całkowitą liczbę procesów,
4. nie jest dozwolone stosowanie metod komunikacji grupowej, dozwolone jest jedynie stosowanie blokujących metod komunikacji punktowej (`MPI_Send` oraz `MPI_Recv`),
5. nie jest dozwolone gromadzenie na robotnikach informacji o zadaniach/wynikach,
6. robotnik odbierając zadanie musi znać jego numer identyfikacyjny, nadzorca odbierając wynik musi również znać numer identyfikacyjny zadania, któremu dany wynik odpowiada, warto do tego celu wykorzystać tag wiadomości,
7. „tagowanie” wiadomości jest w rozważanym problemie kluczowe, może zostać również wykorzystane do przesłania informacji o końcu wszystkich prac,
8. dozwolne jest rozesłanie na wstępie informacji o umownym sygnale zatrzymania,
9. do odbycia „precyzyjnego” snu można (czytaj: należy) wykorzystać funkcję `MPI_Wtime`.