Michał Szczygieł	Aleksander Śmierciak	18/06/2014

Zadanie 7 - Test Millera-Rabina - MPI

Zadanie miało na celu implementację programu określającego dla każdej liczby w ciągu danych wejściowych czy jest ona liczbą pierwszą czy też złożoną. Wykorzystać w tym celu miano test pierwszości liczb Rabina-Millera, a sam program napisać w oparciu o architekturę MPI, umożliwiającą zrównoleglenie potrzebnych obliczeń.

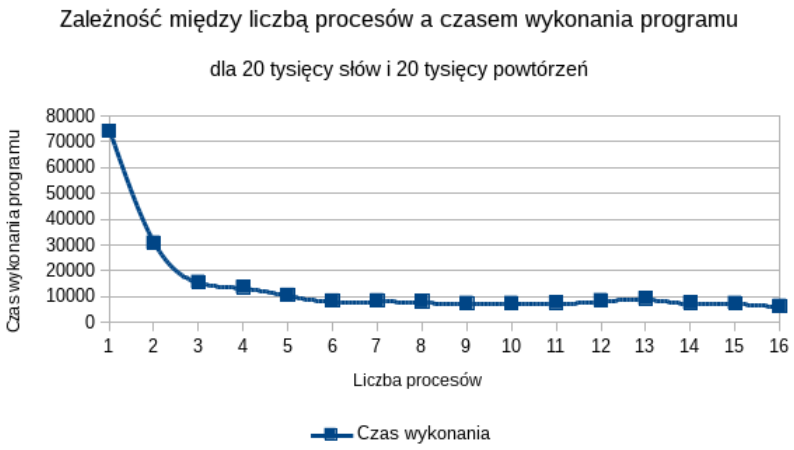
Aby zrealizować schemat komunikacyjny z polecenia (scentralizowana struktura), należało wykorzystać *MPI_Send* oraz *MPI_Recv* – blokujące funkcje wymiany danych między procesami.

Pseudokod dla rozwiązania odpowiadającego schematowi z polecenia prezentuje się następująco:

- dla każdej liczby wyznacz identyfikator węzła obliczającego
- jeśli węzeł jest węzłem głównym wyślij liczbę do wybranego węzła obliczającego
- jeśli nie, ale węzeł jest węzłem obliczającym sprawdź czy liczba jest liczbą pierwszą
- jeśli węzeł jest węzłem głównym odbierz sprawdzone liczby od węzła obliczającego

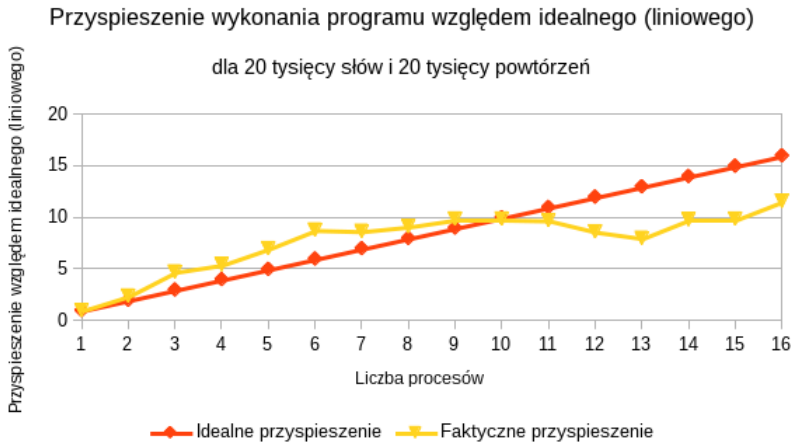
Faktyczny kod źródłowy temu odpowiadający znajduje się w pliku main.cpp, w funkcji searchForPrimesUsingMPI().

```
1 void searchForPrimes(const string &inputPath, const unsigned int
    repeatCount)
2 {
3 LongMultimap results;
4 const LongVector *numbers = readNumbersFromFile(inputPath);
5
6 searchForPrimesUsingMPI(numbers, results, repeatCount);
7
8 delete numbers;
9 }
```



Patrząc na wykresy wydajności tak napisanej implementacji należy mieć na uwadze, że jeden proces zawsze zajmował się jedynie wysyłaniem danych wejściowych i zapisywaniem danych wyjściowych – nie brał udziału w samych obliczeniach. Bezpośrednio wynika z takiego ograniczenia niemożność uruchomienia programu dla liczby procesów równej 1; w celu przedstawienia wydajności „dla 1 procesu" napisano osobny program, sekwencyjnie sprawdzający liczby po kolei.

Z uwagi na specyfikę komputera, na którym były wykonywane obliczenia (4-rdzeniowy procesor z HT), znaczące przyspieszenie obserwowane jest do liczby procesów równej 4 włącznie; zwiększając dalej (szczególnie przy liczbie procesów niebędącej wielokrotnością liczby rdzeni) zauważamy spadek przyrostu wydajności.



Wnioski

Środowisko MPI jest w naszym mniemaniu znacznie mniej wygodne dla komunikacji międzyprocesowej przy dokonywaniu zaawansowanych obliczeń równoległych. Wynika to ze sposobu zapisu tej komunikacji, dla żadnego z popularnych języków programowania niebędącego "natywnym", niejednokrotnych ograniczeń dotyczących typów zmiennych i sposobu ich przekazywania (wspomniany w kodzie brak typu boolean, "załatany" typem *MPI_BOOL* dopiero w nowszym standardzie), a także z mniejszej konfigurowalności architektury, którą można przy pomocy tego środowiska stworzyć.