**PRiR projekt**

Studenci:

*Mateusz Kutek*

*Wojciech Jacoszek*

Temat: Zrównoleglenie algorytmu Rabina Carpa służącego do wyszukiwania wzorca w tekście.

1. **OpenMP**
2. **OpenMPI**

W pierwszej kolejności należy odpowiednio rozdystrybuować pobrany tekst między wszystkie dostępne procesy. Czynność ta wykonywana jest przez proces 0.

*// Calculate size of each block*int size\_of\_each\_block = strlen(txt) / num\_procs;  
max\_size\_of\_each\_block = size\_of\_each\_block + num\_procs + 1;  
  
*// Calculate size of the division reminder*reminder\_count = strlen(txt) - (size\_of\_each\_block \* num\_procs);  
  
sending\_block\_sizes = get\_block\_sizes(reminder\_count, num\_procs, size\_of\_each\_block);  
sending\_block\_indexes = get\_block\_start\_indexes(num\_procs, size\_of\_each\_block);

Jak wiadomo w większości przypadków tekst nie zostanie rozdzielony równo między wszystkie procesy. Uwzględniono, że jeden z procesów otrzyma dodatkowo resztę ze wspomnianego dzielenia

.

W związku z tym, że przesyłane bloki będą różnej wielkości należy określić:

* tablicę zawierającą rozmiary bloków (sending\_block\_sizes)
* tablicę zawierającą indeksy rozpoczynania się poszczególnych bloków (sending\_block\_indexes)

Przed samym wysłaniem bloków należy wysłać niezbędne dane do każdego z procesów za pomocą broadcast. Wspomnianą akcję wykonuje proces 0:

* maksymalna ilość bloku

MPI\_Bcast(&max\_size\_of\_each\_block, 1, MPI\_INT, **ROOT\_RANK**, MPI\_COMM\_WORLD);

* długość szukanego wzorca

MPI\_Bcast(&pattern\_size, 1, MPI\_CHAR, **ROOT\_RANK**, MPI\_COMM\_WORLD);

- sam wzorzec

MPI\_Bcast(pat, pattern\_size, MPI\_CHAR, **ROOT\_RANK**, MPI\_COMM\_WORLD);

Po wspomnianym fakcie można rozesłać bloki tekstu między procesy:

MPI\_Scatterv(txt, sending\_block\_sizes, sending\_block\_indexes, MPI\_CHAR, received\_block, max\_size\_of\_each\_block, MPI\_CHAR, **ROOT\_RANK**, MPI\_COMM\_WORLD);

Następnie każdy z procesów szuka wzorca w swojej części tekstu:

found\_indexes = find\_pattern\_in\_text(pat, received\_block, &found\_indexes\_count);

W sytuacji, gdy dany proces wykona swoje zadanie musi swój rezultat zwrócić do głównego procesu. Rezultat jest reprezentowany jako indeksy miejsc w których rozpoczyna się znaleziony wzorzec. Jeżeli dany proces nie znalazł wzorca to we wspomnianej tablicy będzie znajdować się tylko wartość -1.

Każdy proces przed wysłaniem wyników musi przesłać do głównego procesu ilość znalezionych wzorców w tekście.

MPI\_Gather(&found\_indexes\_count, 1, MPI\_INT, received\_block\_sizes, 1, MPI\_INT, **ROOT\_RANK**, MPI\_COMM\_WORLD);

Analogicznie należy stworzyć dwie struktury przechowujące:

* ilość elementów w każdym z bloków (received\_block\_sizes)
* rozpoczynający się indeks bloku (received\_block\_indexes)

Dane są zwracane za pomocą metody Gatherv do procesu 0:

MPI\_Gatherv(found\_indexes, found\_indexes\_count, MPI\_INT, buff, received\_block\_sizes, received\_block\_indexes, MPI\_INT, **ROOT\_RANK**, MPI\_COMM\_WORLD);

Przed wykonaniem pomiaru czasu wywoływana jest dyrektywa MPI\_Barrier aby synchronizować zakończenie pracy wszystkich procesów.

**Rezultat**

Testy wykonano na pliku txt powieści Lalka zajmującym 1.7 MB pamięci. Wybrano wzorzec „Wokulski”.

Wykres zależności czasu od ilości wykorzystywanych procesorów:

Wykres zależności wartości przyśpieszenia od ilości wykorzystywanych procesorów:

1. **OpenMP + OpenMPI**