Projekt MPI

Rafał Czarnecki

Optymalizacje i implementacja

Zaimplementowałem wszystkie optymalizacje opisane w pracy znajdującej się w treści zadania z tą różnicą, że od razu puszczam drugą wersję algorytmu.
Do sortowania używam rozproszonego odd-even sort.
Do odpowiadania na pytania używam następującego algorytmu:

1. wybieram pierwsze zapytanie nie swojego przedziału
2. uruchamiam binsearch, w którym
   - pytam inne wątki o wartość SA na obecnie przeglądanej pozycji `pos`
   - pytam o część genomu przechowywaną przez wątek zawierający `SA[pos]`
   - jeżeli ta część genomu wystarczy do porównania sekwencji z zapytania, to:
     - tak długo jak jakiś wątek wciąż porównuje swoje zapytanie i może potrzebować mojej części genomu, przesyłam mu ją
     - kiedy wszystkie wątki porównały swoje zapytania, kontynuuję binsearcha
   - jeżeli nie jestem ich porównać, to pytam o kolejną część genomu i powtarzam poprzedni krok
3. po dwukrotnym uruchomieniu binsearcha mam pierwszą i ostatnią pozycję sekwencji z zapytania. Zapisuję wynik i jeżeli mam jeszcze jakieś kolejne zapytania, to biorę kolejne zapytanie i wracam z nim do kroku 2
4. jeżeli odpowiedziałem już na wszystkie zapytania, to dopóki inne wątki nie dotarły do tego kroku, odpowiadam na ich zapytania o moje `SA` lub o moją część genomu

Powyższy algorytm jest wykonywany na każdym wątku, a zapytania są rozłożone jak najbardziej równomiernie między nimi.

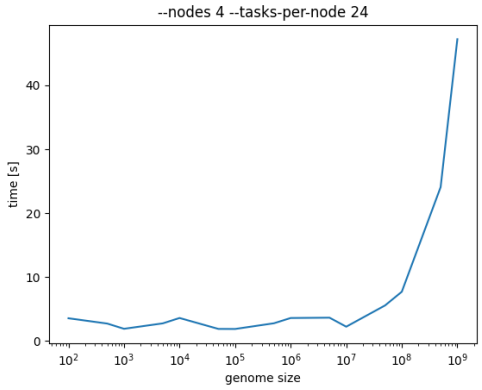
Kompilacja

Po wykonaniu `make` powstanie program `genome_index` uruchamiany zgodnie z opisem w treści zadania.

Testy

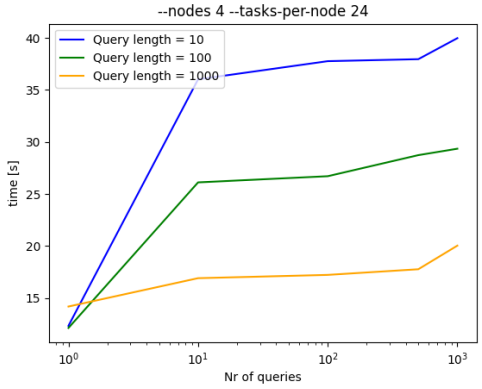
Rozmiar genomu

W tych testach zmieniał się rozmiar genomu. Użyte zostało pojedyncze zapytanie długości 11 oraz `--nodes 4 --tasks-per-node 24`



Rozmiar zapytań

W tych testach zmienia się ilość i rozmiar zapytań. Długość użytego genomu to 100000000, a parametry uruchomienia to `--nodes 4 --tasks-per-node 24`



Ilość node'ów

W tych testach zmienia się ilośc node'ów. Użyłem w nich 100 zapytań długości 100.