Bitonic merge-sort

Jakub Sawicki

14 stycznia 2016

1 Algorytm

Jako podstawę pod algorytm wykorzystałem opis i kod bitonicznego merge-sorta z [1]. Został on zmodyfikowany tak, aby przyjmował jako wejście rekordy wygenerowane przez gensort i zapisywał je w plikach weryfikowalnych przez valsort [2].

Algorytm ten oparty jest na architekturze hiper-kostki. Musi być uruchomiony na ilości procesorów równej potędze dwójki. Każdy procesor wykonuje merge-sort na swojej części danych. Następnie procesory wymieniają się danymi ze swoimi partnerami sortując powstające kawałki. Ta część wymaga $\frac{\log_2^2(n_p)}{2}$ iteracji, gdzie n_p to ilość dostępnych procesorów.

2 Wyniki

Testy wydajnościowe zostały wykonane dla czterech wielkości danych wejściowych: 640MB, 1280MB, 2560MB oraz 5120MB. Co odpowiada odpowiednio 6,4M, 12,8M, 25,6M oraz 51,2M rekordów (każdy po 100 bajtów).

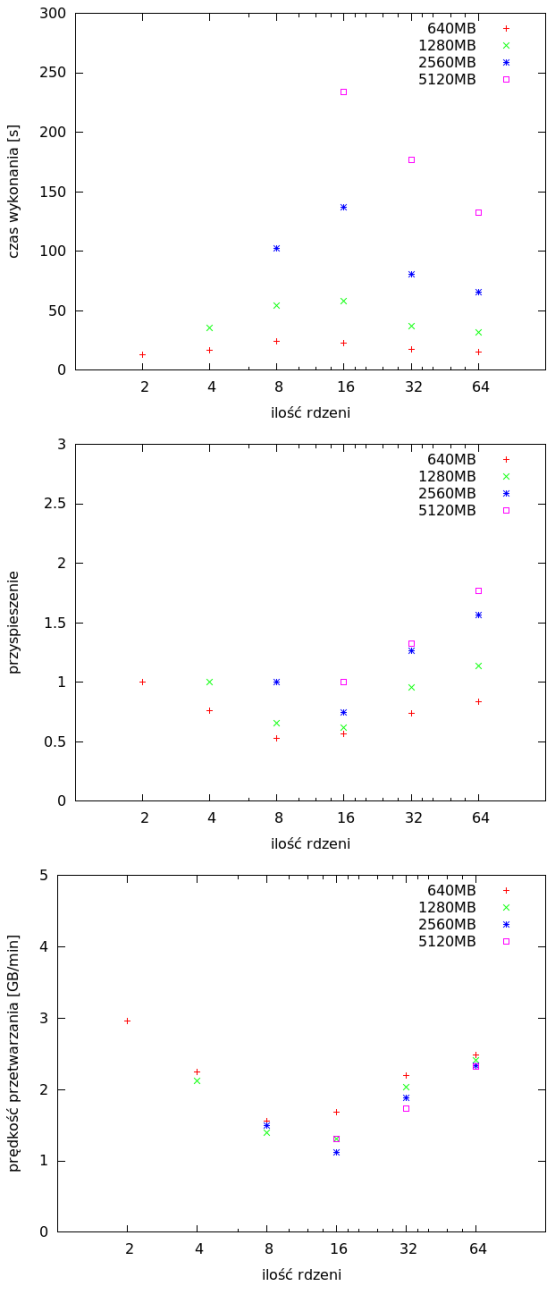
Nie udało się uruchomić algorytmu dla dużych problemów i niewielkiej ilości węzłów. Wynikało to z użycia typu int, do określania wielkości alokowanego bufora, powodując błąd jeśli był on większy niż około 2GB.

Wyniki pokazane są na Rys. 1. Nie ma przyspieszenia, jeżeli obliczenia wykonywane są na pojedynczym węźle, do 8 rdzeni. Dla większej ich ilości wydajność zaczyna wzrastać.

Literatura

[1] *Divide-and-Conquer Parallelization Paradigm* http://cacs.usc.edu/education/cs653/02-3DC.pdf dostęp 2016.01.14

[2] *Sort algorithms benchmark* http://sortbenchmark.org/ dostęp 2016.01.14



Rys. 1: Wykresy czasu wykonania, przyspieszenia oraz tempa przetwarzania danych dla różnych wielkości danych wejściowych. Dane wejściowe miały wielkości 640MB, 1280MB, 2560MB oraz 5210MB.