Drugi projektni zadatak

U ovom zadatku su vršene četiri različite vrste optimizacija nad algoritmom Quicksort.

Za pivota se uvijek bira algoritam koji se nalazi skroz desno, jer po različitim istraživanja takav odabir pivota daje nešto bolje rezultate nego u slučaju kada se bira srednji element, medijana tri elementa ili prvi element. Takođe ovakav pristup čini sam algoritam jednostavnijim, jer u slučaju da se pivot bira na drugi način, potrebna su dodatna proračunavanja i premještanja izabranog pivota na posljednju poziciju u nizu, jer u tom slučaju ne moramo da uračunavamo pivot u poređenjima i zamjenama, i možemo samo u završnom koraku da ga prebacimo na željenu poziciju u nizu. Nedostatak ovakvog izbora pivota su loše performanse u slučajevima kada je niz već sortiran, bilo u opadajućem ili rastućem obliku, ali u ovom slučaju zbog slučajno generisanih nizova, i mogućnosti da vrijednosti elemenata budu pozitivne vrijednosti od 0 do max integer vrijednosti, vjerovatnoća da je ulazni niz već sortiran je zanemarljivo mala.

Paralelni Quicksort predstavlja paralelizaciju Quicksort algoritma na višejezgarnom procesoru. Ovakav pristup u većem dijelu slučajeva daje očekivano ubrzanje, međutim postoje slučajevi kada su zbog elemenata niza particije različitih veličina, i pokreće se veći paralelnih tokova, kod kojih promjena konteksta može biti skuplja nego da se niz sortira običnim Quicksortom. Ovdje treba uzeti u obzir i broj jezgara, tj. veća je vjerovatnoća da će veći broj jezgara omogućiti veći nivo paralelizacije.

Dual pivot Quicksort je varijanta kod koje postoji više manjih particija te je veća šansa da pristup istim elementima bude brži jer se oni već nalaze u kešu. Ovakav Quicksort ne daje mnogo veliko ubrzanje, tačnije do 5% u nekim slučajevima. Najgori slučaj je svakako složenosti $O(n^2)$ kao kod varijante sa jednim pivotom, kada je ulazni niz već sortiran. Varijante sa tri do pet pivota daju još bolje performanse. Iz rezultata vidimo da korištenje kompajlerskih optimizacija bolje djeluje na Quicksort sa jednim pivotom.

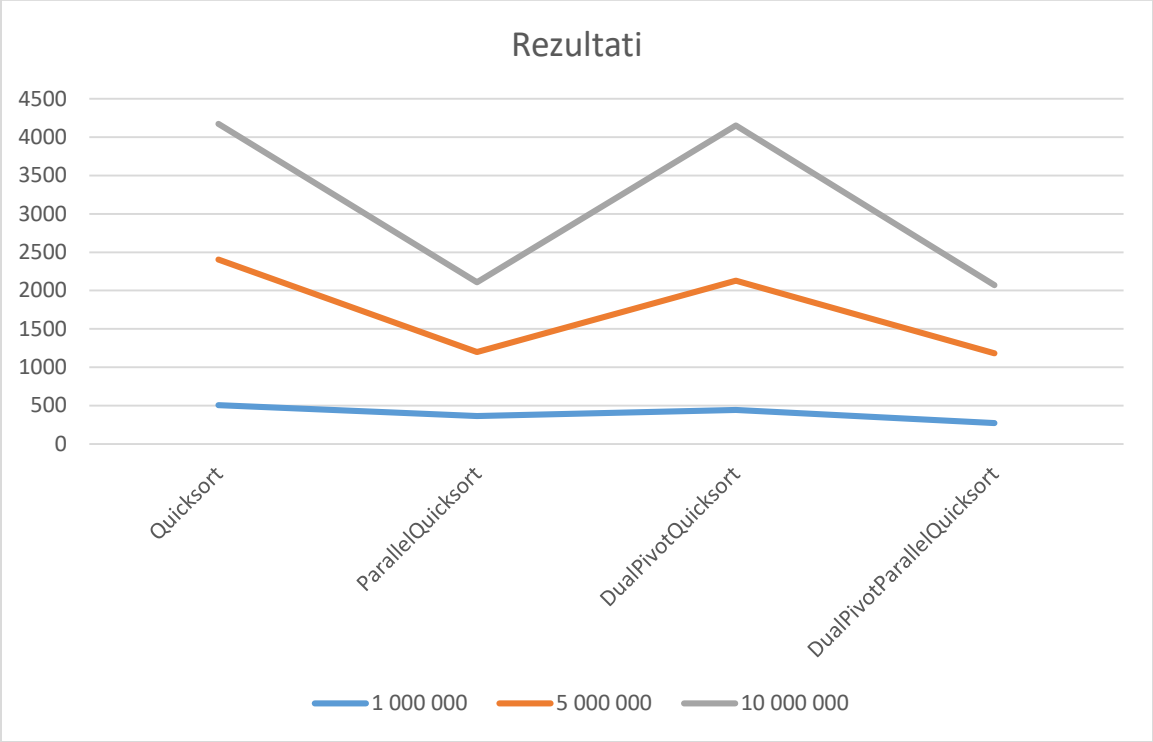
Dual pivot zajedno sa paralelnim izvršavanjem daje najbolje performanse, pogotovo kada imamo veći broj jezgara.

Kada se uključe kompajlerski flegovi za optimizaciju rezultati su brži u svim slučajevima.

Svi rezultati su dobijeni kao aritmetička sredina tri izvršavanja. Za svako od tri izvršavanja korišten je isti ulazni niz slučajno generisanih vrijednosti, za svaki od varijanti Quicksort-a. Vrijednosti su date u milisekundama.

Algoritam	1 000 000	5 000 000	10 000 000
Quicksort	504	2403	4175
ParallelQuicksort	365	1199	2109
DualPivotQuicksort	444	2130	4152
DualPivotParallelQuicksort	274	1181	2071
Quicksort+kompajlerski flegovi za optimizaciju	215	906	1756
ParallelQuicksort+kompajlerski flegovi za optimizaciju	264	761	1206
DualPivotQuicksort+kompajlerski flegovi za optimizaciju	194	905	1824

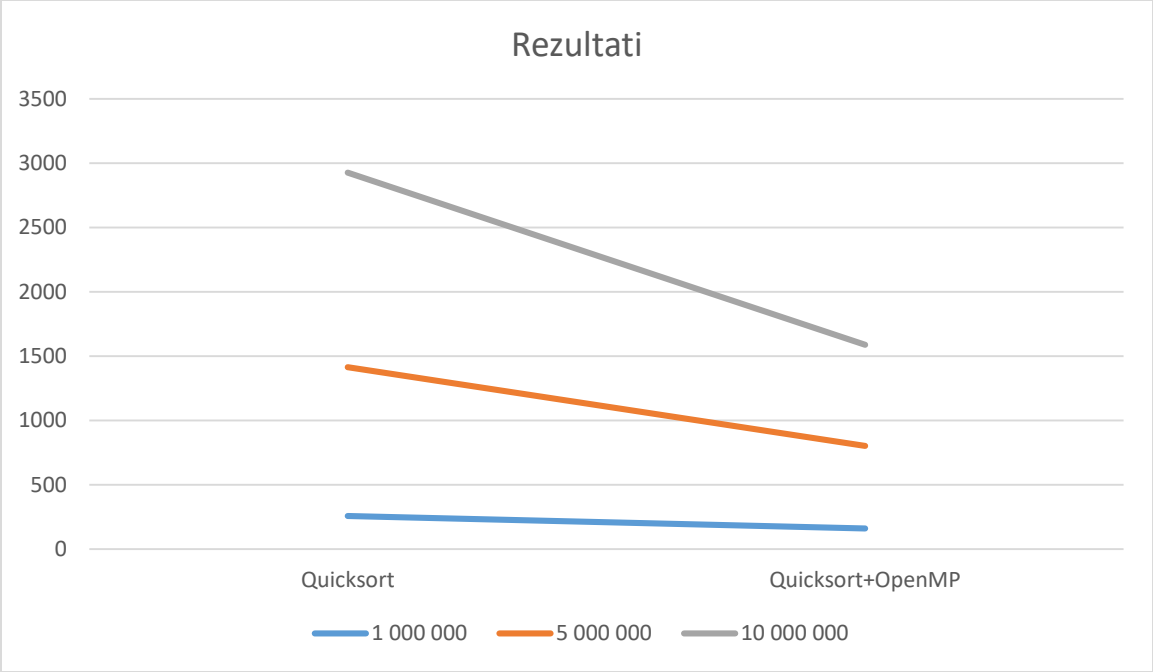
DualPivotParallelQuicksort+ kompajlerski flegovi za optimizaciju	139	674	1099



U programskim jezicima C/C++ postignuta je dodatna optimizacija korištenjem OpenMP platforme.

Testiranje je izvršeno na Linux operativnom sistemu. Korišteni su isti nizovi kao ulaz za sortiranje korištenjem običnog i optimizovanog algoritma. Rezultati su dati u milisekundama, za tri uzastopna izvršavanja.

Algoritam	1 000 000	5 000 000	10 000 000
Quicksort	258	1415	2928
Quicksort+OpenMP	161	803	1588



Dodatno je urađeno, poređenja radi, sortiranje korištenjem Merge sort algoritma, koje je i dodatno optimizovano korištenjem OpenMP platforme. Iz rezultata možemo vidjeti da je Merge sort sporiji od Quicksort algoritma, što je i očekivano.

Algoritam	1 000 000	5 000 000	10 000 000
Merge sort	296	1632	3497
Merge sort+OpenMP	218	1064	2210