U programskom jeziku C i C# vršena je optimizacija MergeSort i QuickSort algoritma. Zbog ograničenja VisualStudio frameworka, optimizacija upotrebom paralelizacije na višejezgarnom procesoru (OpenMP) je realizovana u programskom jeziku C pri cemu je program kompajliran i pokretan u LinuxOS(testiranje je vršeno u virtuelnoj mašini) dok su sve ostale optimizacije realizovane u C# u VisualStudio-u.

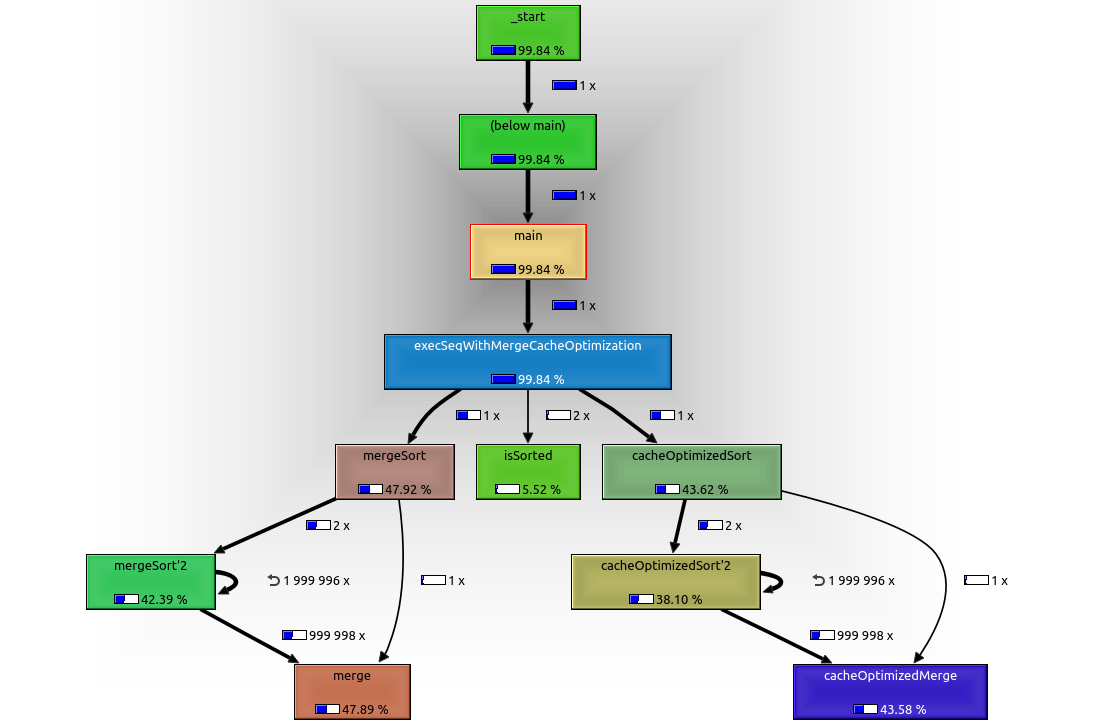
*Parallel Invoke Merge Sort* optimizacija podrazumijeva paralelizaciju Merge Sort algoritma upotrebom Parallel.Invoke metode programskog jezika C#.

*Task Merge Sort* podrazumijeva optimizaciju Merge Sort algoritma korištenjem Task.Factory.StartNew() metode programskog jezika C#.

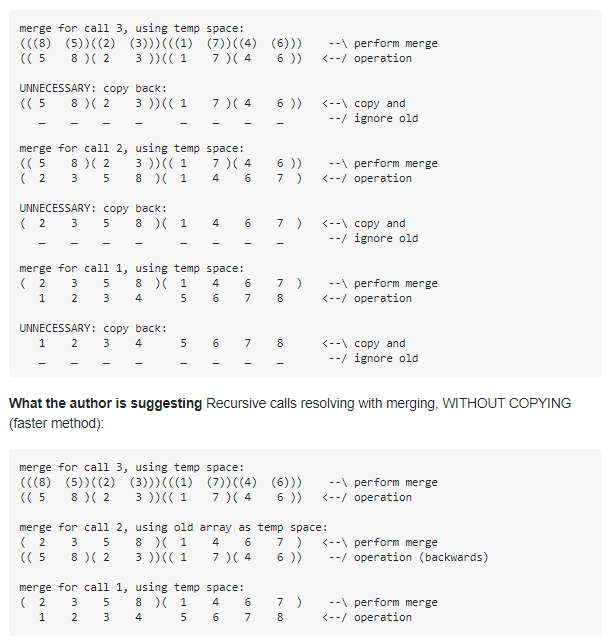
Obje navedene optimizacije dio posla raspoređuju na više (hardverskih) niti, i na taj način značajno ubrzavaju izvršavanje algoritma, u poredjenju sa serijskm izvršavanjem.

Kao „dubina“ do koje se vrši optimizacija (odnosno maksimalni broj niti na koji se rasporedjuje posao) korišten je broj fizičkih jezgara.

*Cache optimized MergeSort* predstavlja keš optimizaciju. Nakon testiranja i graficke reprezentacije upotrebom Valgrind i KCachegrind alata, optimizacija keša (iako je konstantna) nije značajnija.



Sljedeća slika pokazuje način rada Cache optimiyed Merge Sort algoritma. (Preuzeto sa Stack overflow - a)



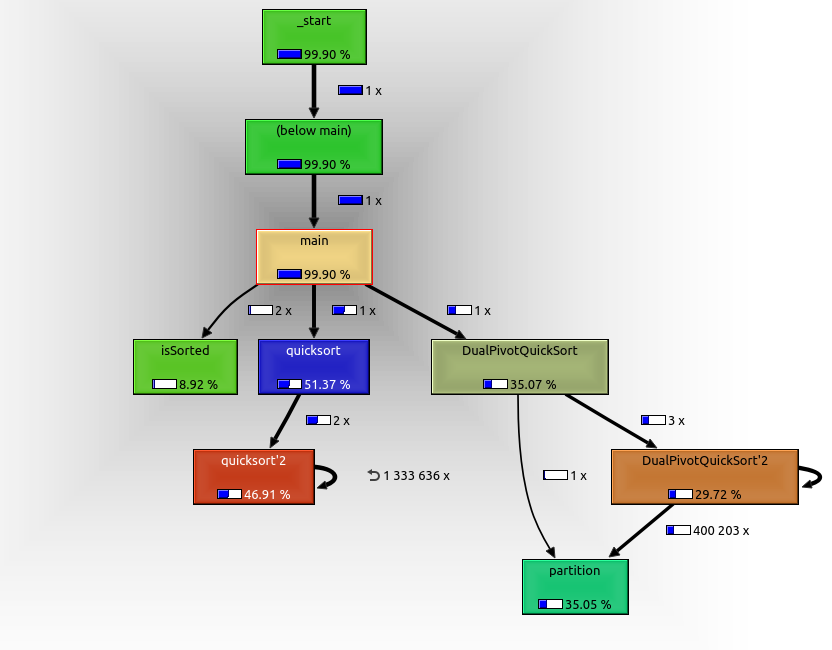
*Parallel Invoke Cache Optimized Merge Sort* predstavlja kombinaciju dva gore opisana načina optmizacije. Vrijeme sortiranja se značajno smanjuje, što je i očekivano.

Pored Merge Sort optimizacije uradjena je i optimizacija QuickSort algoritma. Glavni cilj je demonstracija keš optimizacije, što je predstavljeno *Dual Pivot Quick Sort* algoritmom optimizacije, dok su *Parallel Invoke I Task Quick Sort* navedeni radi usporedba a glavna ideja je ista kao I kod Merge Sort-a.

*ParallelInvoke DualtPivot QuickSort* predstavlja kombinaciju DualPivot (kes optimizacije) i paralelizacije na visejezgarnom procesoru.

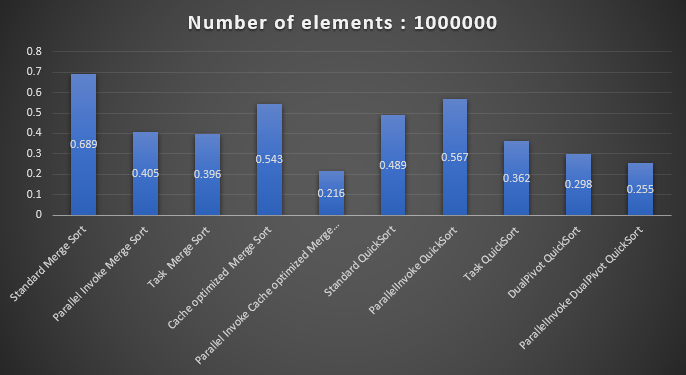
U svim primjerima mozemo primjetiti primjenu Amdahl-ovog zakona, jer maksimalno poboljanje performansi je ograniceno dijelom algoritma koji se ne moze paralelizovati.

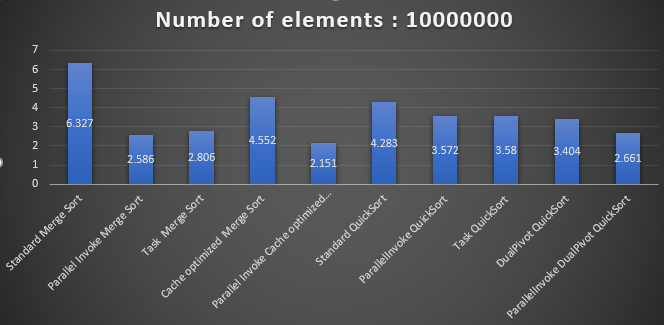
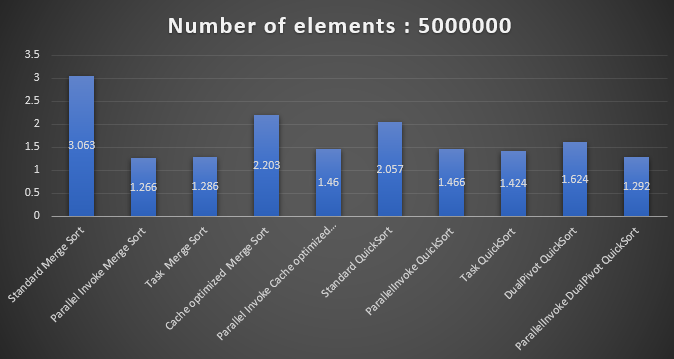
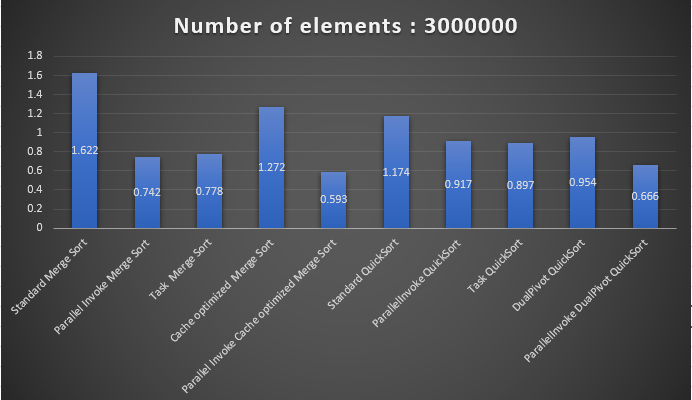
Optimizacija keša upotrebom DualPivot QuickSort-a je primjetno značajnija, što sljedeća slika pokazuje.

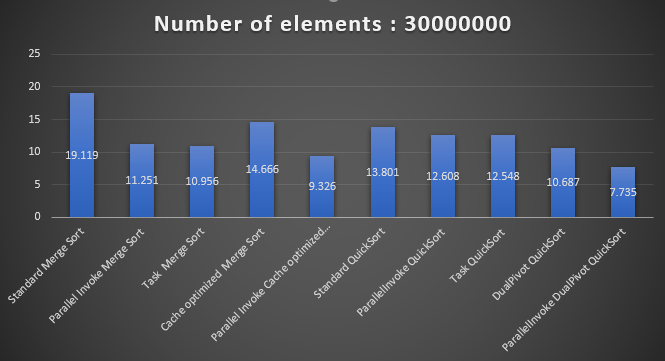


***Napomena : u oba slučaja je vršeno testiranje na nizu veličine 1000000, a rezultati se odnose na L1 Data Read Miss***

**Naredne slike predstavljaju rezultate mjerenja vremena izvršavanja algoritama sortiranja. VRIJEME JE MJERENO U SEKUNDAMA.**

****

****



OpenMP optimizacija Merge Sort algoritma je napisana u C-u, a pokretana u LinuxOS-u, pri cemu je program kompaliran gcc kompajlerom.

Za optimizaciju je korišten **#*pragma omp task.*** Slično kao i kod Parallel.Invoke i Task.Factory, I ovdje je maksimalni broj niti na koji se može rasporediti posao ograničen na broj jezgara vraćen f-jom *get\_nprocs().*

**U prilogu su rezultati mjerenja vremena izvršavanja sortiranja upotrebom “standardnog” MergeSort algoritma, te upotrebom OpenMP optimizacije MergeSort algoritma.**

**Rezultati su dati u sekundama.**

