PARALELNO PROGRAMIRANJE – 2. LABORATORIJSKA VJEŽBA

U okviru ove laboratorijske vježbe bilo je potrebno razvili algoritam koji će protiv stvarnog igrača igrati igru 4 u nizu. Računalo će uz pomoć pretraživanja u dubinu određivati svoj sljedeći potez. Algoritam je bilo potrebno paralelizirati.

Podjela poslova sam izvršio na sljedeći način. Iz neke pozicije je moguće odigrati 7 poteza jer imamo 7 stupaca. Potez nakon toga je također moguće odigrati na 7 načina, dakle imamo ukupno 49 načina na koji se mogu odigrati sljedeća dva poteza. Za svaki od tih 49 poteza sam napravio jedan zadatak koji onda podijelim procesorima radnicima tako da svakom prvom pošaljem jedan zadatak, pa drugi i tako u krug. Naravno, ovdje nailazimo na problem kod paralelizacije ako imamo 50 ili više jezgri, no u trenutku pisanja ove laboratorijske vježbe (ljeto Gospodnje 2019.) 50 ili više jezgri je znanstvena fantastika za prosječnog studenta.

Komunikacija se odvija pomoću MPI specifikacije. Koristim dvije MPI funkcije, send i recv za slanje poslova i primanje rezultata između glavnog procesa i radnika.

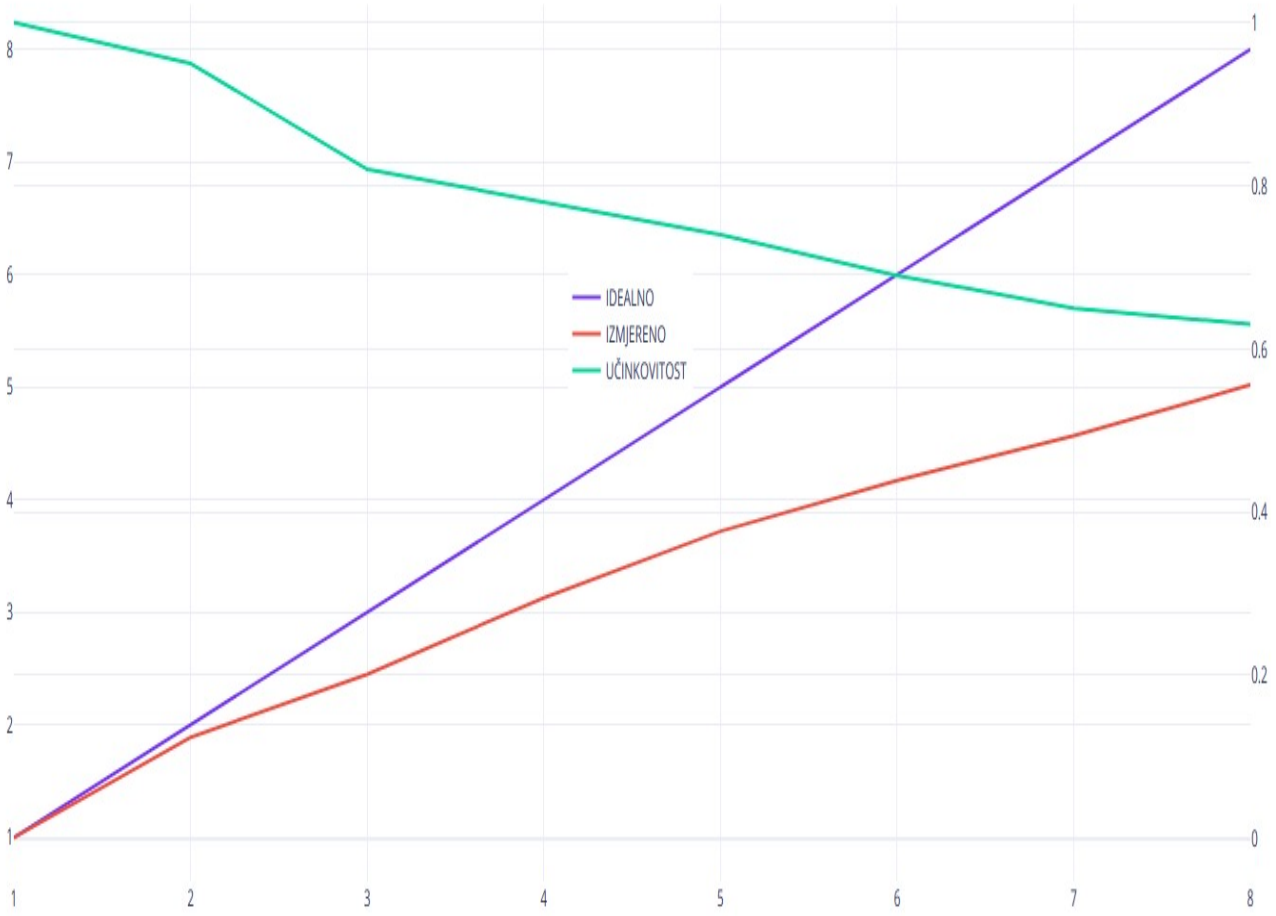
Nemam implementiranu aglomeraciju jer mi se sve odvija lokalno pa sam smatrao da nije potrebna.

Što se tiče pridruživanja, imam ujednačavanje opterećenja.

Rezultati mjerenja performansi algoritama su prikazani u nastavku. Mjerenja su obavljena na serveru zakupljenom na Digital Oceanu koji je imao 16 jezgri. Pretraživao sam 7 poteza u dubinu.

BROJ PROCESORA	IDEALNO VRIJEME	IZMJERENO VRIJEME	IDEALNO UBRZANJE	IZMJERENO UBRZANJE	UČINKOVITOST
1	26.85	26.85	1	1	1
2	13.43	14.18	2	1.89	0.95
3	8.95	10.95	3	2.45	0.82
4	6.71	8.58	4	3.13	0.78
5	5.37	7.22	5	3.72	0.74
6	4.48	6.44	6	4.17	0.69
7	3.84	5.88	7	4.57	0.65
8	3.36	5.35	8	5.02	0.63

Grafovi mjerenja prikazani su u nastavku.



Iz ovih podataka možemo vidjeti da je algoritam dobro skalabilan i učinkovit. Učinkovitost raste povećanjem broja procesora gotovo linearno. Algoritam je iznimno dobro prilagođen za paralelni rad.