Eksperimenti jakog I slabog skaliranja   
Kanonov algoritam množenja matrica

### Tehničke karakteristike sistema

Operativni sistem: Windows 10

Dodatne biblioteke: numpy 1.21.2, mpi4py 3.1.1

# Analiza koda

Postupak: Procenat koda koji se može paralelizovati (p) dobijen je tako što se vreme izvršavanja ovog dela koda podeli ukupnim vremenom izvršavanja sekvencijalne verzije. Procenat koda koji ne može biti paralelizovan je onda 1 – p.

Merenjem vremena izvršavanja sekvencijale verzije za ulazne matrice dimenzija 200x200 dobijen je procenat sekvencijalnog dela koda koji se može paralelizovati i iznosi 99%, dok preostalih 1% predstavlja procenat koda koji ne može biti paralelizovan. Ovo važi u oba programska jezika, jer kod implementiran tako da ima istu kompleksnost i u Golang-u i Pythonu.

## Jako skaliranje, Amdalov zakon

Po Amdalovom zakonu maksimalno ubrzanje uslovljeno je delom koja koji se ne može paralizovati.

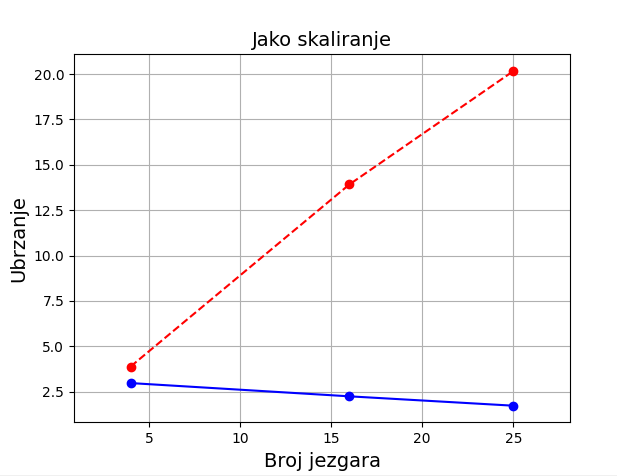
Jednačina maksimalnog ubrzanja: , gde s predstavlja procenat vremena ižvršavanja nužno sekvencijalnog dela koda, p procenat vremena izvršavanja koda koji može biti paralelizovan, n broj procesorskih jezgara.

Primenom Amdalovog zakona za problem Kanonovog algoritma na mašini sa 4 procesorska jezgra, dobije se maksimalno ubrzanje od 3.88 puta.

Eksperiment jakog skaliranja izveden je pokretanjem paralelne verzije koda sa ulaznim matricama dimenzija 200x200 na različitom broju procesorskih jezgara, 30 puta radi bolje preciznosti. Kako je ideja Kanonovog algoritma da u svakom koraku množi i šiftuje podblokove dimenzija n/sqrt(p), paralelna verzija algoritma radi sa najmanje 4 procesa, jer je to najmanji broj p koji može da se korenuje. Za potrebe iscrtavanja grafika i samog eksperimenta, razmotren je rad algoritma i sa 16 i 25 procesorskih jezgara iako arhitektura sistema ne podržava taj broj procesora.

### Python

Dobijeni rezultati predstavljeni su na grafiku ispod, x osa označava broj procesora, y osa postignuto ubrzanje. Ubrzanje je izračunato po formuli ts/tp gde je ts srednje vreme ižvršavanja sekvencijalne verzije, dok je tp srednje vreme izvršavanja paralelne verzije. Rezultati svakog od 30 pokretanja nalaze se u fajlovima “paralell4.txt”, “parallel16.txt” I “paralell25.txt”. Na grafiku crvenom bojom je označeno maksimalno ubrzanje dobijeno Amdalovim zakonom, zbog korišćenja 16 i 25 procesora, izračunato je maksimalno ubrzanje i za taj broj procesora, mada jasno je da je to ubrzanje na mašini od 4 procesora nemoguće. Maksimalno ubzanje dobijeno u ovom projektu postignuto je za 4 procesora i iznosi 2,97. Zbog kreiranja i pokretanja većeg broja procesa kada to tehnički nije moguće (16 i 25 procesora), ubrzanje je manje nego sa 4 procesora. Za 16 procesora ubrzanje je 2.24, dok za 25 iznosi 1.72. Odnos maksimalnog ubrzanja i ubrzanja postignutog u ovom projektu najbolje vidimo na prvoj tački grafa, koja predstavlja ubrzanje za 4 procesorska jezgra.

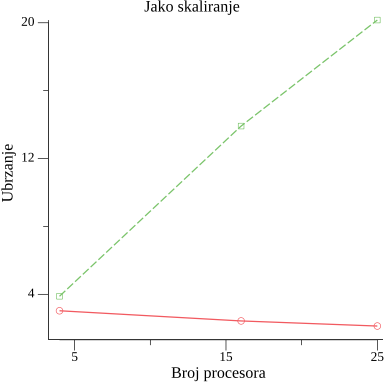


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj procesora | Serijalna sv | Paralelna sv | Serijalna std | Paralelna std | Ubrzanje |
| 4 | 39.25 s | 13.21 s | 3.65 | 3.14 | 2.97 |
| 16 | 39.25 s | 17.51 s | 3.65 | 3.47 | 2.24 |
| 25 | 39.25 s | 22.75 s | 3.65 | 3.99 | 1.72 |

U tabeli su za svaki od broja procesora, vidimo srednja vrednost izvršavanja sekvencijalne i paralelne verzije (oznaka sv), vrednosti standardne devijacije kod sekvencijalne i paralelne verzije kao i postignuto ubrzanje.

### Go

Kako je kompleksnost koda u Golang jeziku podjenaka kao u Pythonu, procenat fiksno sekvencijalnog koda je isti, i maksimalno ubrzanje dobijeno primenom Amdalovog zakona iznosi 3.88. Grafik prikazuje ubrzanje postignuto za 4, 16, 25 procesora kao i maksimalno ubrzanje za isti broj procesora uslovljeno Amdalovim zakonom. Realno ubrzanje postignuto u ovom projektu prikazano je crvenom linijom, dok isprekidana zelena linija predstavlja teorijski maksimum ubrzanja. Kako je eksperiment rađen na istom računaru, hardversko ograničenje uslovljeno brojem procesora, opravdava zašto algoritam ne ubrzava shodno Amdalovom zakonu za 16, 25 procesora. Objašnjenje je isto kao za eksperiment rađen u Python programskom jeziku.



U tabeli su za svaki od broja procesora, vidimo srednja vrednost izvršavanja sekvencijalne i paralelne verzije (oznaka sv), vrednosti standardne devijacije kod sekvencijalne i paralelne verzije kao i postignuto ubrzanje.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj procesora | Serijalna sv | Paralelna sv | Serijalna std | Paralelna std | Ubrzanje |
| 4 | 3.90 s | 1.27 s | 0.56 | 0.10 | 3.03 |
| 16 | 3.90 s | 1.60 s | 0.56 | 0.44 | 2.43 |
| 25 | 3.90 s | 1.82 s | 0.56 | 1.33 | 2.13 |

## Slabo skaliranje, Gustafsonov zakon

Gustafsonov zakon baziran je na aproksimacijima da se paralelni deo koda skalira linearno sa količinom resursa, i da se sekvencijalni deo ne povećava sa obzirom na veličinu problema. Formula po kojij se izračunava ubrzanje je sledeća:

scaled speedup = s + p × N

gde s i p respektivno označavaju procenat sekvencijalnog i paralelnog koda (kao u Amdalovom zakonu). Na osnovu ove formule skalirano ubrzanje se linearno povećava u odnosu na broj procesora, i ne postoji gornja granica za skalirano ubrzanje kao kod Amdalovog zakona. Ovo se naziva slabim skaliranjem, gde se skalirano ubrzanje računa na osnovu količine posla obavljenog za skaliranu veličinu problema.

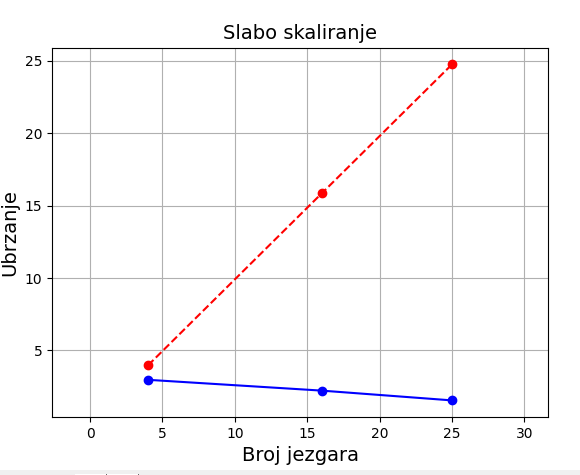
Primenom formule na problem ovog projekta za 4 procesora dobija se skalirano ubrzanje 3,97.

Procena posla za eksperimente slabog skaliranja rađena je na sledeći način. Početne dimenzije matrice su 100x100 za teorijski 1 procesor. Kako je Kanonov algoritam kompeksnosti, i kako se posao ya svaki od p procesa računa kao , dimenzije matrice za procesor p računaju se po formuli:

Tako za 4, 16, 25 procesora respektivno dobijamo 200,400,500 kao dimenzije matrica. Eksperimenti su rađeni pokretanjem programa 30 puta za svaki par parametara n I p, a zatim računanjem srednje vrednosti izvršavanja. Deljenjem dobijenih uprosečenih vrednosti sekvencijalne I paralelne verzije dobijamo ubrzanje.

### Python

Rezultati skaliranja u programskom jeziku Python prikazani su na grafiku ispod, realno ubrzanje programa jeste 2,97 jer je to maksimalni broj procesora koje mašina podržava. Zbog potrebe iscrtavanja grafika eksperiment je izvršen i na 16, 25 procesora. Kako postizanje ubrzanja definisanog Gustafsonovim zakonom za broj procesora veći od 4 nije moguće, logično je da sistem usporava. Usporenje je izazvano podelom posla i kreiranjem procesa, kao i stvarnim čekanjem i oslobađanjem 4 realna procesorka jezgra. Međutim vidimo da program svakako ubrzava u odnosu na sekvencijalnu verziju. Crvena boja predstavlja skalirano ubrzanje po zakonu, plava predstavlja realno ubrzanje postignuto u projektu.

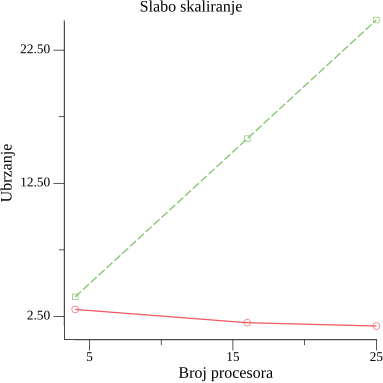


U tabeli su za svaki od broja procesora, vidimo srednja vrednost izvršavanja sekvencijalne i paralelne verzije (oznaka sv), vrednosti standardne devijacije kod sekvencijalne i paralelne verzije kao i postignuto ubrzanje.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj procesora | Serijalna sv | Paralelna sv | Serijalna std | Paralelna std | Ubrzanje |
| 4 | 39.25 s | 13.21 s | 3.65 | 3.14 | 2.97 |
| 16 | 308.42 s | 148.51 s | 7.21 | 6.69 | 2.08 |
| 25 | 419.45 s | 270.73 s | 11.12 | 10.72 | 1.55 |

### Go

Rezulati su prikazani na sledećem grafiku zelenom bojom označeno je teorijsko ubrzanje, crvenom realno ubrzanje, razlog izgleda grafa isti je kao kod Python izvršavanja. U go programskom jeziku vidimo nešto veće ubrzanje nego u Pythonu, ali odstupanje nije veliko.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Broj procesora | Serijalna sv | Paralelna sv | Serijalna std | Paralelna std | Ubrzanje |
| 4 | 3.90 s | 1.27 s | 0.56 | 0.10 | 3.03 |
| 16 | 8.75 s | 3.92 s | 0.40 | 0.50 | 2.23 |
| 25 | 10.19 s | 5.72 s | 1.04 | 0.611 | 1.78 |

U tabeli su za svaki od broja procesora, vidimo srednja vrednost izvršavanja sekvencijalne i paralelne verzije (oznaka sv), vrednosti standardne devijacije kod sekvencijalne i paralelne verzije kao i postignuto ubrzanje.