FAKULTET TEHNIČKIH NAUKA
UNIVERZITET U NOVOM SADU

Monte Carlo simulacija

Izveštaj za projektni zadatak

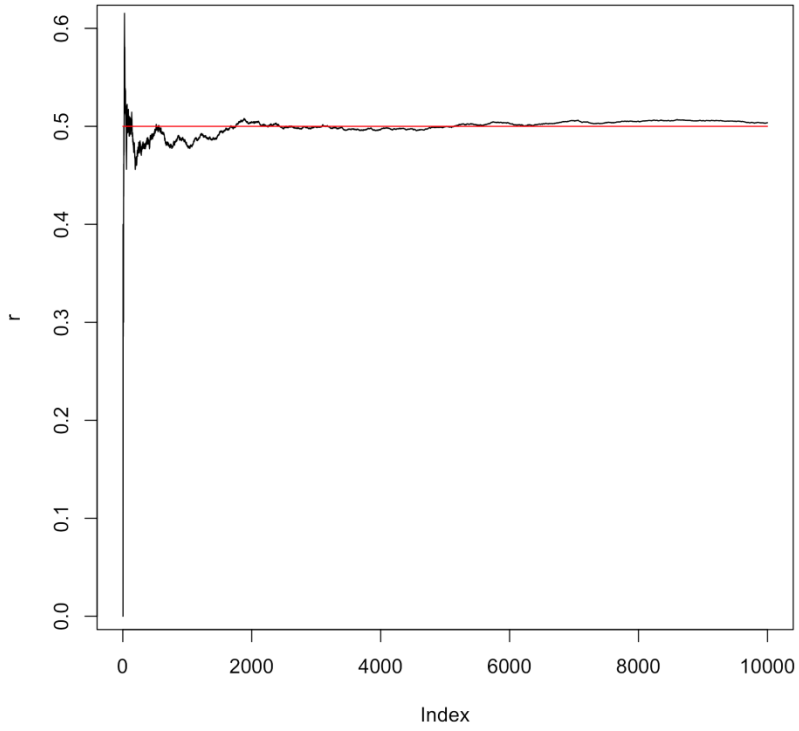
Dušan Stević

9/5/2020

Sadržaj

Motivacija problema

Monte Carlo simulacija spada u kategoriju probabilističkih aproksimativnih tehnika za simulaciju i predikciju. Zasnovana je na teoriji velikih brojeva i teoriji slučajnog uzrokovanja. Osnovna ideja koja se krije iza Monte Carlo simulacije je da sa porastom broja simulacija eksperimentalna verovatnoća se približava (konvergira) teorijskoj verovatnoći (npr.kada bacamo novčić ako imamo jako puno pokušaja bacanja onda će odnos pismo glava biti pibližno 50%-50%).



Slika 1 Teorija velikih brojeva na primeru bacanja novčića

Projektni zadatak

Monte Carlo simulacija ima širok dijapazon primene. U ovo radu Monte Carlo simulacija je primenjena u dve obalsti:

1. Simulacija izračunavanja broja π
2. Simulacija predikcije cene finansijske aktive (akcije, obveznice, derivati, kripto valute, itd.)

Osnovni zadatak u ovom projektu bio je razvoj serijske i paralelne verzije softverskog rešenja koje će omogućiti matematičku simulaciju izračunavanja broja π kao i simulaciju izračunavanja buduće vrednosti finansijske aktive. Serijske i paralelne verzije programa ostvarene su u Python i Golang programskim jezicima. Pored razvoja serijske i paralelne verzije programa posebna pažnja je posvećena i vizuelizaciji dobijenih rešenja. Vizuelizacija rešenja zasnovana je na programskom jeziku Pharo uz korišćenje graphic

engine-a Roassal. Takođe veoma važnu stavku u projektu čine i eksperimenti skaliranja. Kako je reč o projektu koji obuhvata paralelno programiranje i računarstvo visokih performansi bilo je potrebno sprovesti detaljne eksperimente kako bi se utvrdio uticaj broja procesnih jedinica i veličine posla na ubrzanje paralelnog programa u odnosu na serijski program. Eksperimenti skaliranja obuhvataju eksperimente jakog i slabog skaliranja. Cilj eksperimenata jakog i slabog skaliranja je da pokažemo kako se ovi algoritmi ponašanja na stvanom hardveru.

Simulacija izračunavanja broja π

Osnovni pojmovi

Monte Carlo simulacija može da posluži u izračunavanju broja π. Ako imamo kvadrat i u kvadratu imamo krug odnos površina kruga i kvadrata je π/4. Ovaj odnos približno možemo da dobijemo ako površinu kvadrata "izbombardujemo" tačkama i onda napravimo odnos tačke koje su upale u krug/ukupan broj tačaka. Množenjem ovog odnosa sa 4 dobija se približna vrednost broja π koja je sve tačnija i tačnija kako povećavamo broj simulacija.

Input/output simulacije izračunavanja broja π

1. Input: Pseudo-slučajno generisane koordinate tačaka (x i y) kojima se “bombarduje” površina kvadrata
2. Output: Aproksimirana vrednost broja π

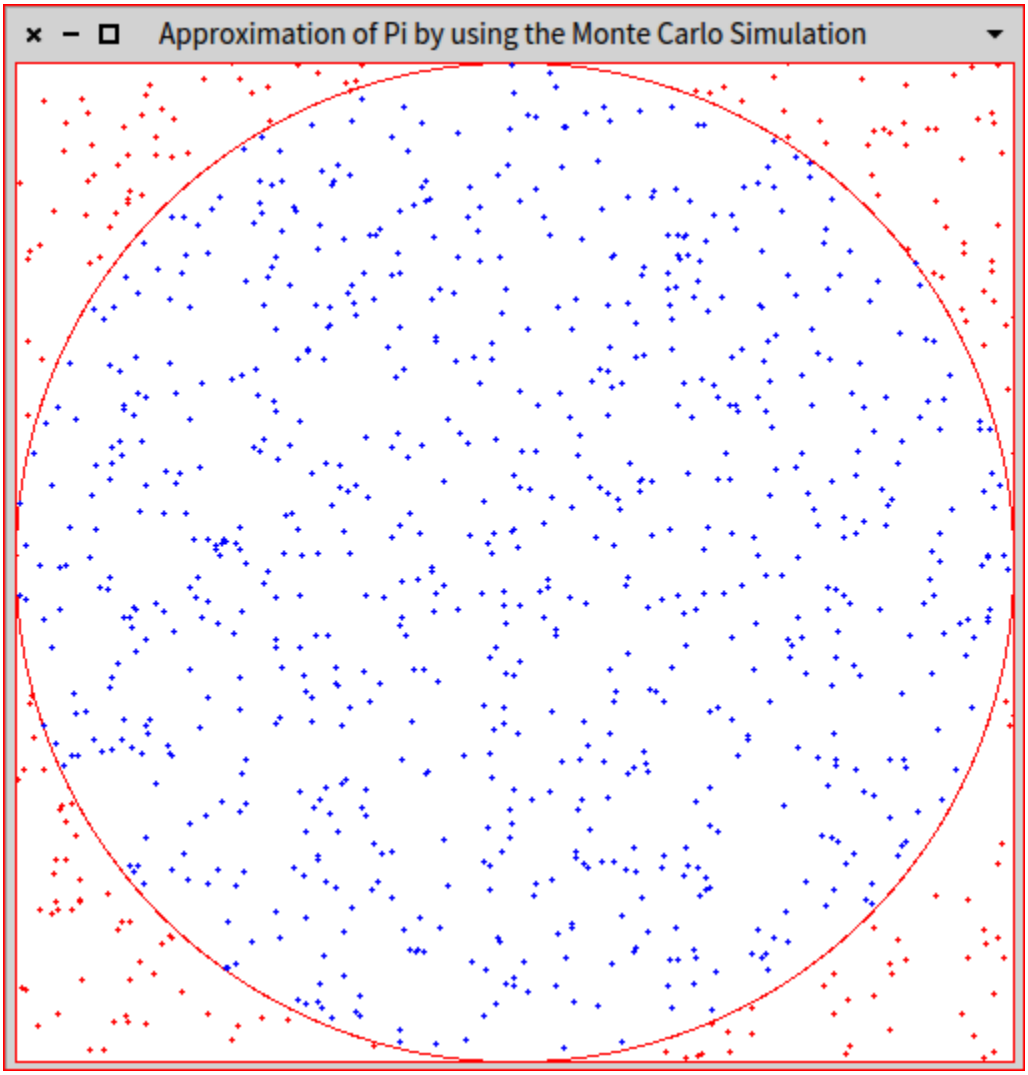
Serijski pristup simulacije izračunavanja broja π

1. Na pseudo-slučajan način odredimo koordinatu x na interval [0,1]
2. Na pseudo-slučajan način odredimo koordinatu y na interval [0,1]
3. Dobili smo tačku sa koordinatama (x,y)
4. Radimo sa jediničnom kružnicom čiji je centar u koordinatnom početku (0,0), a radijus r = 1
5. Proveravamo da li je udaljenost novo formirane tačke manja od 1
    a. $x^2+y^2<r^2$ odnosno u slučaju jedinične kružnice $x^2+y^2<1$ (tačka se nalazi unutar kružnice)
    b. $x^2+y^2=r^2$ odnosno u slučaju jedinične kružnice $x^2+y^2=1$ (tačka se nalazi na kružnici)
    c. $x^2+y^2>r^2$ odnosno u slučaju jedinične kružnice $x^2+y^2>1$ (tačka se nalazi van kružnice)
6. Ukoliko se tačka nalazi unutar kružnice uvećavamo brojač tačaka unutar kružnice
7. Ukupan broj simulacija predstavlja ukupan broj tačaka kojima se “bombarduje” površina kvadrata opisanog oko jedinične kružnice
8. Zbog odnosa površine kruga i površine kvadrata koja iznosi π/4 odnos broja tačaka unutar kružnice i ukupnog broja tačaka množimo sa 4 kako bismo dobili aproksimiranu vrednost broja π
9. $\frac{P_{kruga}}{P_{kvadrata}} \approx \frac{Broj \text{ tačaka u krugu}}{Ukupan\ broj \text{ tačaka}} \approx \frac{\pi}{4}$
10. Izračunali smo približno vrednost broja π koja je sve tačnija i tačnija kako povećavamo broj simulacija odnosno broj tačaka sa kojim “bombardujemo” površinu kvadrata

Paralelni pristup simulacije izračunavanja broja π

1. Podelimo ukupan broj simulacija tj. ukupan broj tačaka sa kojim "bombardujemo" površinu kvadrata na npr. četvrtine
2. Svaki task obrađuje jednu četvrtinu tačaka
3. Svi taskovi se obavljaju u paraleli
4. $\frac{P_{kruga}}{P_{kvadrata}} \approx \frac{Rezultat_{task1}+Rezultat_{task2}+Rezultat_{task3}+Rezultat_{task4}}{Ukupan\ broj\ \text{tačaka}}$
5. Zbog odnosa površine kruga i površine kvadrata koja iznosi π/4 odnos broja tačaka unutar kružnice i ukupnog broja tačaka množimo sa 4 kako bismo dobili aproksimiranu vrednost broja π
6. Izračunali smo približno vrednost broja π koja je sve tačnija i tačnija kako povećavamo broj simulacija odnosno broj tačaka sa kojim "bombardujemo" površinu kvadrata

Vizuelizacija simulacije izračunavanja broja π



Slika 2 Vizuelizacija simulacije izračunavanja broja π u programskom jeziku Pharo n = 1000

Simulacija predikcije cene finansijske aktive

Osnovni pojmovi

Kako bi decision maker-i doneli ispravnu investicionu odluku u pogledu kupovine ili prodaje finansijske aktive neophodno je da izvrše korektnu predikciju cene finansijske aktive. Monte Carlo simulacija može da pomogne u predikciji cena finansijske aktive. Povlačenjem podatke o kretanju cena akcija sa berze dolazimo do istorijskih podataka. Buduća cena akcija može da se izračuna preko formule za eksponencijalni rast koja uključuje volatilnost i kontinuelnu stopu prinosa. Volatilnost i kontinuelna stopa prinosa se računaju na osnovu istorijskih podataka.

$$\text{Periodic Daily Return} = \ln\left(\frac{\text{Day's Price}}{\text{Previous Day's Price}}\right)$$

Formula 1 Kontinualna stopa prinosa

$$\text{Drift} = \text{Average Daily Return} - \frac{\text{Variance}}{2}$$

Formula 2 Finansijski drift

$$\text{Random Value} = \sigma \times Z_{\text{score}}(\text{Rand}())$$

Formula 3 Slučajna vrednost

$$\text{Next Day's Price} = \text{Today's Price} \times e^{(\text{Drift}+\text{Random Value})}$$

Formula 4 Buduća cena finansijske aktive

Sa što većim brojem simulacija dobijamo familiju krivih koje formiraju zvonastu (normalnu) distibuciju. Svaka kriva predstavlja jednu simulaciju tj. predikciju buduće cene finansijske aktive. Sa velikim brojem simulacija imamo aparat sa kojim možemo sa određenom verovatnoćom da tvrdimo kako će se cena akcija kretati u budućnosti. Simulacije formiraju fen dijagram gde jasno vidimo koja od simulacija je najverovatnija da se desi, a koje su manje verovatne da se dese. Najverovatnije predikcije su one koje se nalaze u koridoru +/- jedna standardna devijacija (σ), dok su manje verovatne predikcije koje se nalaze u koridoru +/- dve standardne devijacije (2σ) i +/- tri standardne devijacije (3σ).

Input/output simulacije predikcije cene finansijske aktive

1. Input: povučeni istorijski podaci o kretanju cena finansijske aktive za određeni vremenski interval sa specijalizovanog sajta za praćenje finansijske aktive Yahoo finance
2. Output: Predikcije budućih cena finansijske aktive za određeni vremenski interval tj. za određeni prediction window

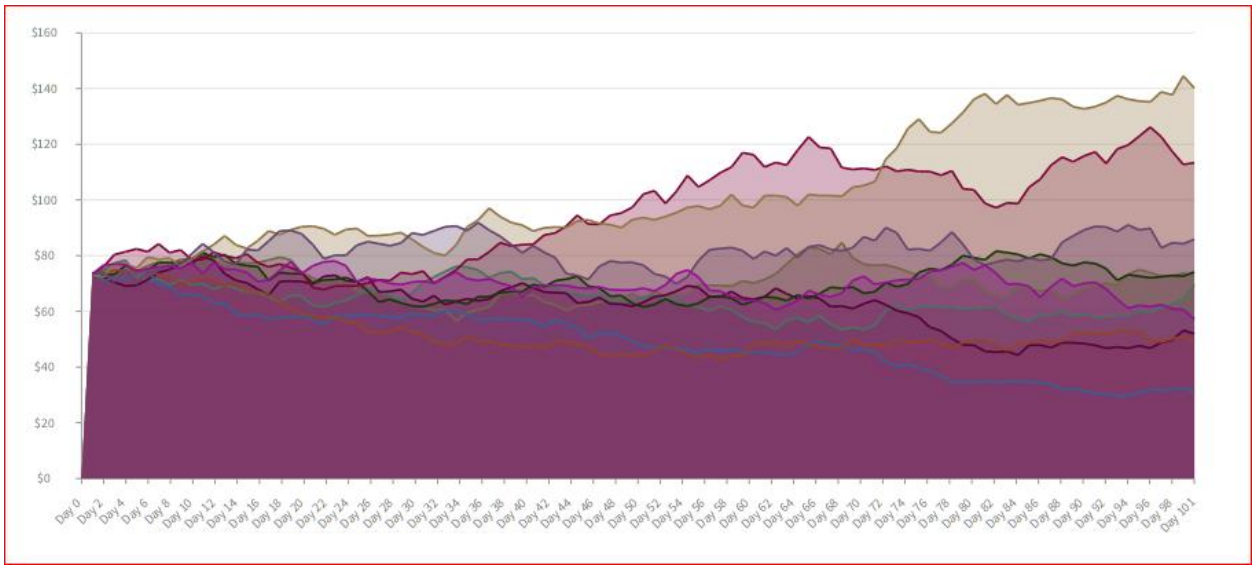
Serijski pristup predikcije cene finansijske aktive

1. Povlačimo finansijske podatke preko berzanskog API-a
2. Računamo prinos finansijske aktive
3. Računamo standardnu devijaciju finansijske aktive odnosno volatilnost
4. Računamo buduću cenu finansijske aktive koristeći Formula 4 Buduća cena finansijske aktive
5. Formiramo fen dijagram krivih
6. Odabiramo onu predikciju koja ima najveću verovatnoću da se desi
7. Ostale predikcije odbacujemo

Paralelni pristup predikcije cene finansijske aktive

1. Umesto da vršimo predikciju po predikciju sekvencijalno
2. Funkcija za izračunavanje buduće cene finansijske aktive se poziva simultano
3. Više taskova istovremeno vrši predikcije čime se za kraće vreme formira fen dijagram krivih
4. Svaki task računa buduće cene finansijske aktive sa različitim ulaznim parametrima čme se oslikava tržišna nestabilnost i slučajnost
5. Formiramo fen dijagram krivih
6. Odabiramo onu predikciju koja ima najveću verovatnoću da se desi
7. Ostale predikcije odbacujemo

Vizuelizacija predikcije cene finansijske aktive



Slika 3 Vizuelizacija predikcije cene finansijske aktive za 100 dana u programskom jeziku Pharo n = 10

Eksperimenti skaliranja

Eksperimenti skaliranja predstavljaju važan deo projektnog zadatka. U eksperimentima skaliranja bilo je neophodno ispitati kako ubrzanje koje se dobija paralelizacijom koda zavisi od broja procesnih jedinica u

slučaju jakog skaliranja odnosno kako ubrzanje zavisi od broja procesnih jedinica i obima posla koji se izvršava u slučaju slabog skaliranja. U Python programskoj realizaciji za procesne jedinice koristi se Pool procesa iz standardne multiprocessing biblioteke, dok se u Golang implementaciji koriste Go rutine. Tokom eksperimentisanja primećeno je da serijska i paralelna verzija simulacija u Golangu imaju kraće vreme izvršenja u odnosu na simulacije u Python-u. Brže izvršavanje Golang programa u odnosu na Python programe prepisuje se činjenici da procesi u Python-u ne dele istu memoriju i da gube vreme na prekljućivanje. Go rutine dele istu memoriju i ne gube vreme na prekljućivanje i zato je izvršavanje Golang programa nekoliko desetina puta brže u odnosu na izvršavanje programa u Python-u.

$$\text{speedup} = \frac{t_1}{t_N}$$

Formula 5 Ubrzanje paralelnih programa

Ubrzanje se računa kao odnos vremena izvršavanja programa na jednoj procesnoj jedinici (serijski program) i vremena izvršavanja programa na N procesnih jedinica za istu količinu posla. U slučaju da je problem savršeno idealan za paralelizaciju tj. da se kompletan program može paralelizovati tada je ubrzanje koje se postiže proporcionalno broj procesnih jedinica. Kako u eksperimentima jakog tako i u eksperimentima slabog skaliranja tri veličine igraju ključnu ulogu na ostvarene performanse:

1. s – Procenat ukupnog vremena izvršenja serijskog programa koje se ne može paralelizovati
2. p – Procenat ukupnog vremena izvršenja serijskog programa koje se može paralelizovati
3. N – Broj procesnih jedinica

$$s + p = 1$$

Formula 6 Proporcija ukupnog vremena izvršavanja programa

Kako je celokupan serijski program (simulacija broja π i simulacije u finansijama) moguće paralelizovati odatle sledi da je s = 0 i p = 1 odnosno teorijski maksimum ubrzanja jednak je broj procesnih jedinica N.

Arhitektura sistema nad kojom su vršeni eksperimenti skaliranja

- Model: Dell Inspiron 15 3000
- Procesor: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz, 1701 Mhz, 2 Core(s), 4 Logical Processor(s)
- RAM memorija: 4 GB, 1600MHz, DDR3L
- Grafička kartica: NVIDIA GeForce 920M
- Cache memorija: 3 MB
- Hard drive: 500 GB HDD
- Operativni sistem: Microsoft Windows 7 Professional Version 6.1.7601 Service Pack 1 Build 7601

Eksperimenti jakog skaliranja

Teorijske napomene

U eksperimentima jakog skaliranja ubrzanje koje se postiže paralelizacijom posmatra se u odnosu na promenu broja procesnih jedinica dok se količina posla drži fiksnom. U stručnoj literaturi ovaj tip eksperimenta poznat je kao Amdalov zakon. Ubrzanje koje se postiže prema Amdalovom zakonu računa se prema sledećoj formuli:

$$\text{Amdahl's law speedup} = \frac{1}{(s + \frac{p}{N})}$$

Formula 7 Ubrzanje prema Amdalovom zakonu

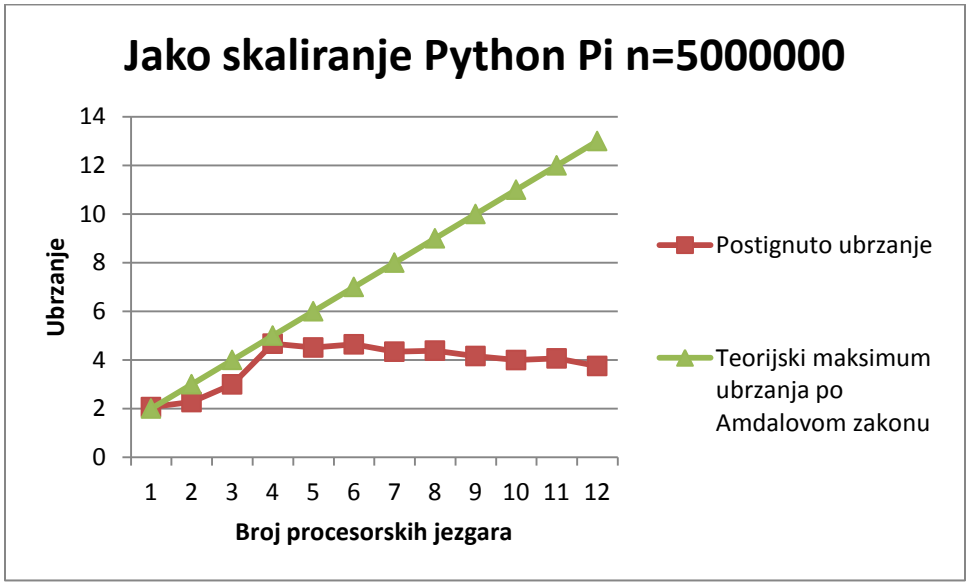
Kako je u konkretnom slučaju s = 0 i p = 1 formulu za ubrzanje prema Amdalovom zakonu možemo da korigujemo:

$$\text{Amdahl's law speedup} = \frac{1}{(s + \frac{p}{N})} = \frac{1}{(0 + \frac{1}{N})} = N$$

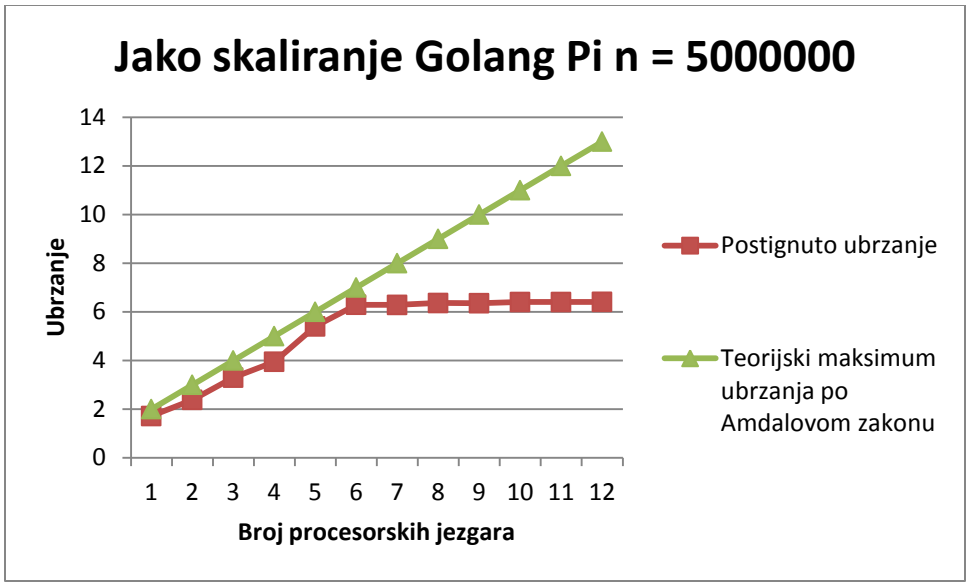
Formula 8 Teorijski maksimum ubrzanja po Amdalovom zakonu

Zaključujemo da je teorijski maksimum ubrzanja po Amdalovom zakonu jednak broju procesnih jedinica N.

Eksperimenti jakog skaliranja simulacija izračunavanja broja π



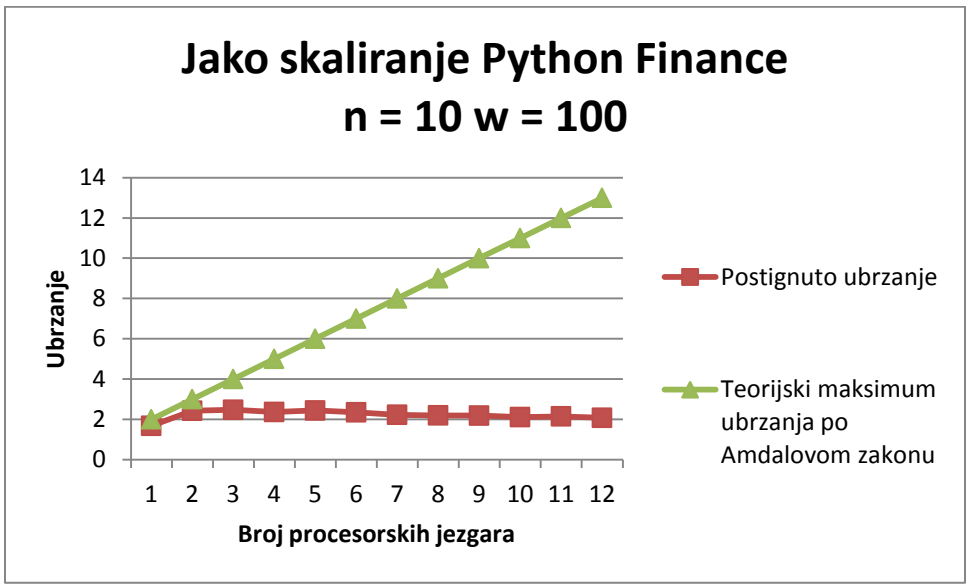
Slika 4 Jako skaliranje Python Pi n = 5000000



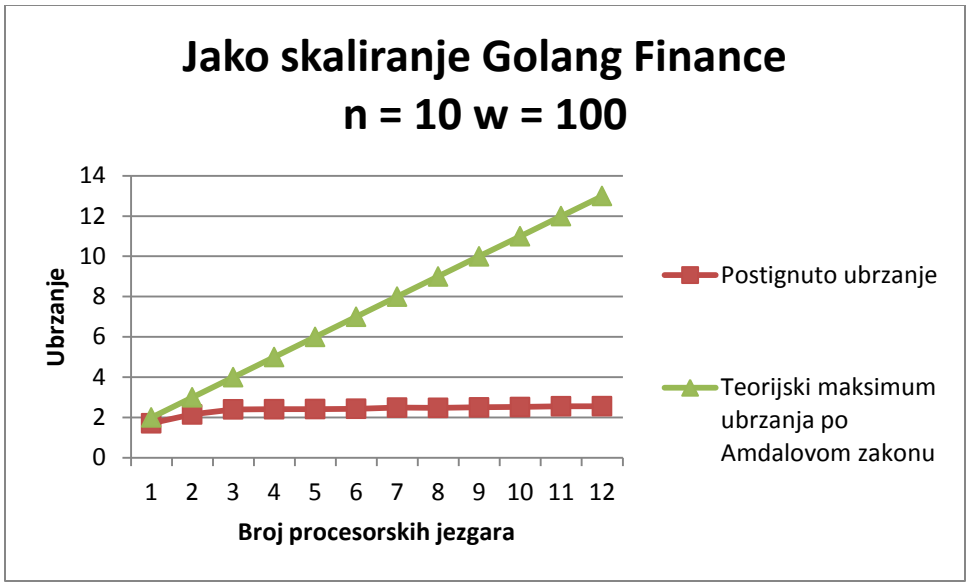
Slika 5 Jako skaliranje Golang Pi n = 5000000

Na x-osi se nalazi broj procesnih jedinica (broj procesorskih jezgara) dok se na y-osi nalazi ubrzanje. Zelena linija predstavlja teorijski maksimum ubrzanja po Amdalovom zakonu (idealno skaliranja) dok crvena linija predstavlja ostvareno ubrzanje. Go implementacija postiže bolje rezultate zato što go rutine dele istu memoriju i ne gube vreme na prekljućivanje.

Eksperimenti jakog skaliranja simulacija predikcije cene finansijske aktive



Slika 6 Jako skaliranje Python Finance n = 10 w = 100



Slika 7 Jako skaliranje Golang Finance n = 10 w = 100

Na x-osi se nalazi broj procesnih jedinica (broj procesorskih jezgara) dok se na y-osi nalazi ubrzanje. Zelena linija predstavlja teorijski maksimum ubrzanja po Amdalovom zakonu (idealno skaliranja) dok crvena linja predstavlja ostvareno ubrzanje. Go implementacija postiže bolje rezultate zato što go rutine dele istu memoriju i ne gube vreme na preključivanje. Simulacija cene finansijske aktive određena je brojem ponavljanja n i veličinom prediktivnog prozora w.

Eksperimenti slabog skaliranja

Teorijske napomene

U eksperimentima slabog skaliranja ubrzanje koje se postiže paralelizacijom posmatra se u odnosu na promenu broja procesnih jedinica i proporcionalnog povećanja količina posla. Rast broja procesorskih jezgara isparaćen je proporcionalnim rastom količine posla te se na taj način postiže konstantan posao po procesorskom jezgru. U stručnoj literaturi ovaj tip eksperimenta poznat je kao Gustafsonov zakon. Ubrzanje koje se postiže prema Gustafsonov zakonu računa se prema sledećoj formuli:

$$\text{Gustafson's law speedup} = \ s + p \ \times N$$

Formula 9 Ubrzanje prema Gustafsonovom zakonu

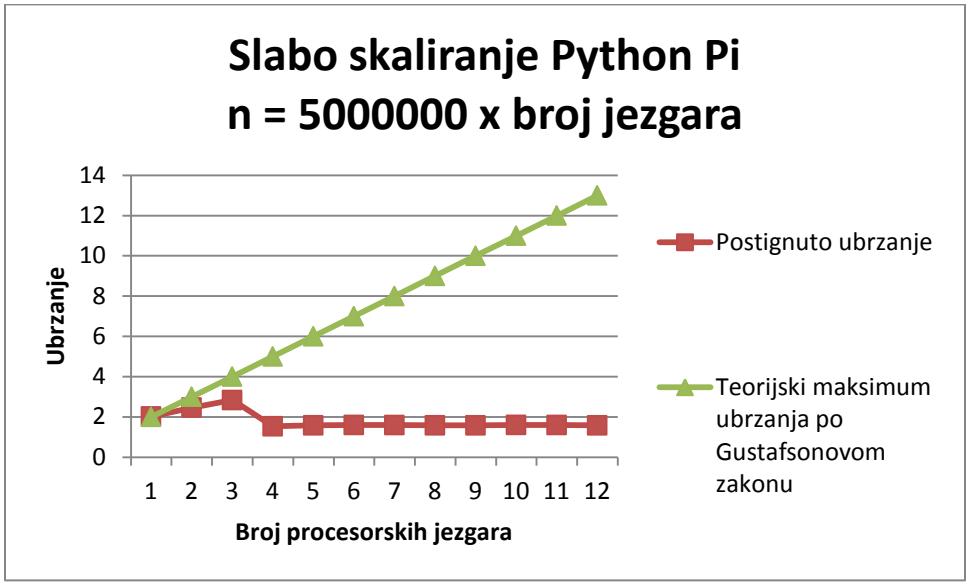
Kako je u konkretnom slučaju s = 0 i p = 1 formulu za ubrzanje prema Gustafsonovom zakonu možemo da korigujemo:

$$\text{Gustafson's law speedup} = \ s + p \ \times N = 0 + 1 \times N = N$$

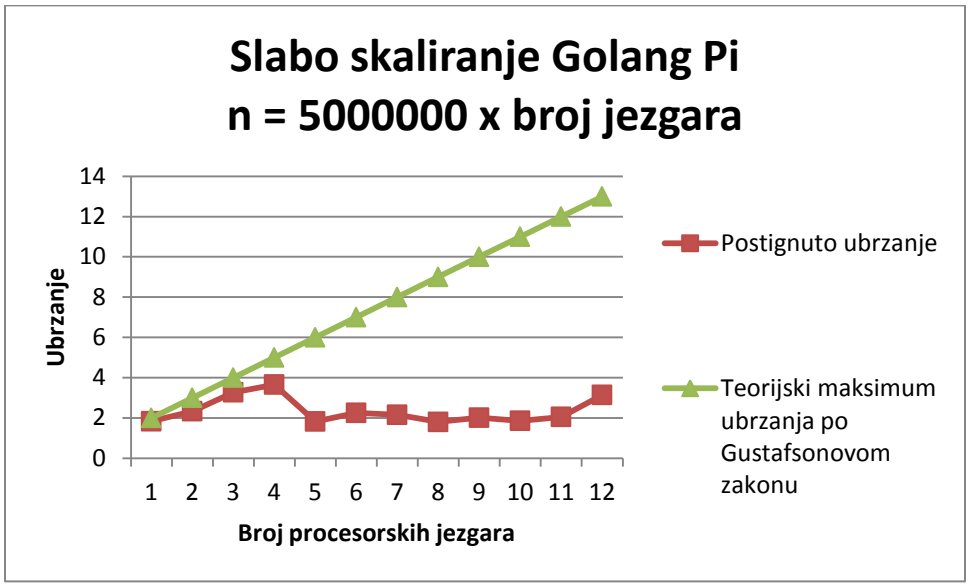
Formula 10 Teorijski maksimum ubrzanja po Gustafsonovom zakonu

Zaključujemo da je teorijski maksimum ubrzanja po Gustafsonovom zakonu jednak broju procesnih jedinica N.

Eksperimenti slabog skaliranja simulacija izračunavanja broja π



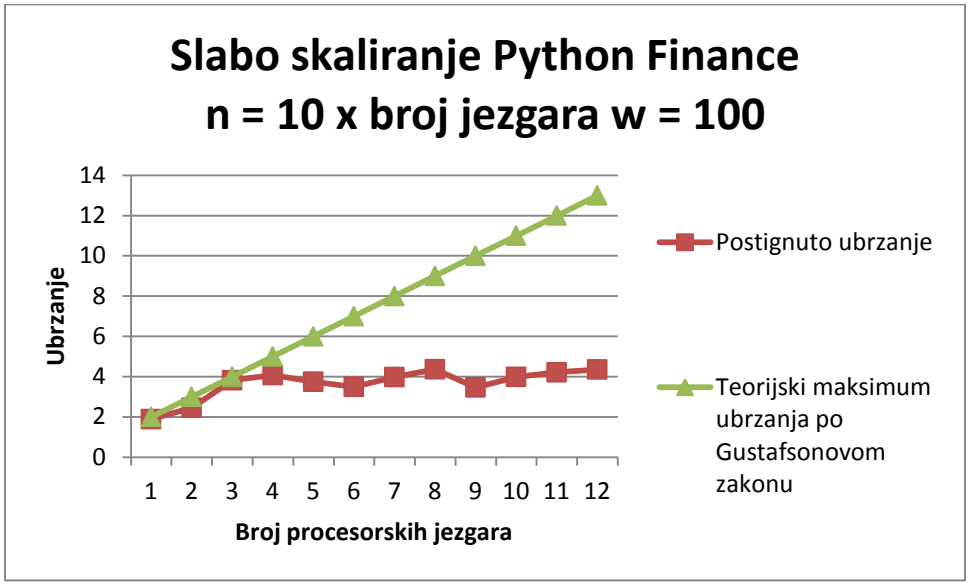
Slika 8 Slabo skaliranje Python Pi n = 5000000 x broj jezgara



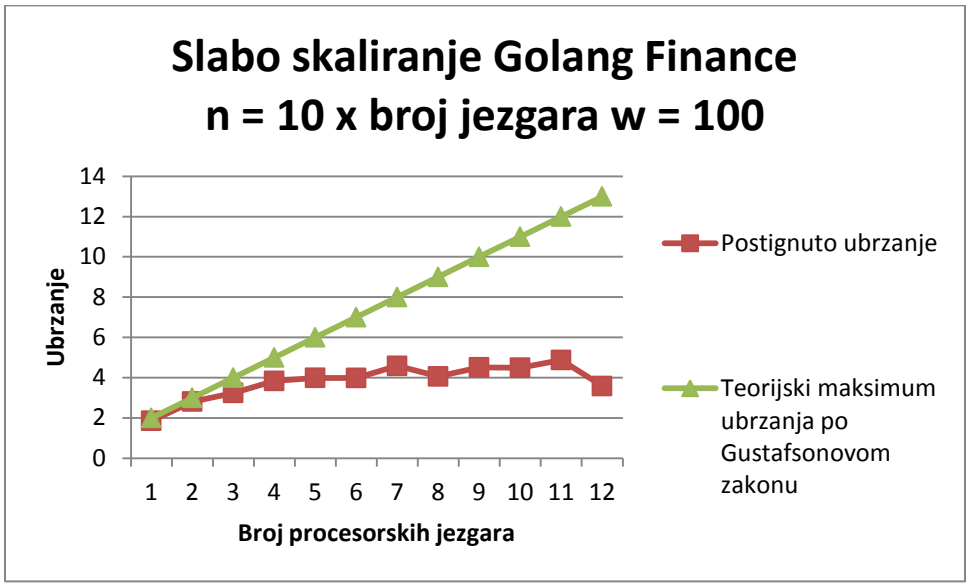
Slika 9 Slabo skaliranje Golang Pi n = 5000000 x broj jezgara

Na x-osi se nalazi broj procesnih jedinica (broj procesorskih jezgara) dok se na y-osi nalazi ubrzanje. Zelena linija predstavlja teorijski maksimum ubrzanja po Gustafsonovom zakonu (idealno skaliranja) dok crvena linja predstavlja ostvareno ubrzanje. Go implementacija postiže bolje rezultate zato što go rutine dele istu memoriju i ne gube vreme na prekljućivanje.

Eksperimenti jakog skaliranja simulacija predikcije cene finansijske aktive



Slika 10 Slabo skaliranje Python Finance n = 10 x broj jezgara w = 100



Slika 11 Slabo skaliranje Golang Finance n = 10 x broj jezgara w = 100

Na x-osi se nalazi broj procesnih jedinica (broj procesorskih jezgara) dok se na y-osi nalazi ubrzanje. Zelena linija predstavlja teorijski maksimum ubrzanja po Gustafsonovom zakonu (idealno skaliranja) dok crvena linja predstavlja ostvareno ubrzanje. Go implementacija postiže bolje rezultate zato što go rutine dele istu memoriju i ne gube vreme na preključivanje. Simulacija cene finansijske aktive određena je brojem ponavljanja n i veličinom prediktivnog prozora w.

Korisni linkovi i bibliografija

Oblast	Link
Python	http://www.igordejanovic.net/courses/tech/Python/
	https://github.com/vladaindjic/ntp-2020/blob/master/napredni-python/code/konkurentno_programiranje/pregled.md
	https://www.python.org
	http://www.igordejanovic.net/courses/ntp/napredni-python/
	https://www.youtube.com/watch?v=IEEhzQoKtQU
	https://www.youtube.com/watch?v=fKl2JW_qrso&t=622s
Go	http://www.igordejanovic.net/courses/tech/GoLang/index.html
	https://golang.org
	https://www.youtube.com/watch?v=C8LgvuEBraI
	https://www.youtube.com/watch?v=LvgVSSpwND8
	https://www.youtube.com/watch?v=kjr3mOPv8Sk&t=6s
Pharo	http://www.igordejanovic.net/courses/tech/Pharo/index.html
	https://pharo.org
	http://agilevisualization.com
	https://www.youtube.com/watch?v=-Pk4q5oMdLo
	https://www.youtube.com/watch?v=iXUZiFtnxK8&t=47s
Monte Carlo	https://www.investopedia.com/terms/m/montecarlosimulation.asp
	https://en.wikipedia.org/wiki/Monte_Carlo_method
Skaliranje	https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/
Finansijski podaci	https://finance.yahoo.com
Hardverske odlike	https://www.cnet.com/products/dell-inspiron-15-3000-series-non-touch-laptop-dncwc107s/