Univerzitet u Beogradu - Elektrotehnički fakultet
Multiprocesorski sistemi (13S114MUPS, 13E114MUPS)

Domaći zadatak 1 - OPENMP

Izveštaj o urađenom domaćem zadatku

Predmetni saradnici:
doc. dr Marko Mišić
dipl. ing. Pavle Divović

Studenti:
Iva Potkonjak 2019/0301
Lazar Premović 2019/0091

Beograd, novembar 2022.

Sadržaj

1 PROBLEM 1 - PRIME

1.1 Tekst problema

Paralelizovati program koji vrši određivanje ukupnog broja prostih brojeva u zadatom opsegu. Program se nalazi u datoteci **`prime.c`** u arhivi koja je priložena uz ovaj dokument. Prilikom paralelizacije nije dozvoljeno koristiti direktive za podelu posla (*worksharing* direktive), već je iteracije petlje koja se paralelizuje potrebno raspodeliti ručno. Obratiti pažnju na ispravno deklarisanje svih promenljivih prilikom paralelizacije. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]

1.2 Delovi koje treba paralelizovati

1.2.1 Diskusija

Delovi koda, koje je moguće paralelizovati, predstavljaju dve for petlje, od kojih je jedna ugnježdena i while petlja. Paralelizovana je spoljna for petlja, koja iterira kroz brojeve u zadatom opsegu. Unutrašnja for petlja nije paralelizovana, a njen zadatak je da proveri da li je trenutno posmatrani broj iz opsega deljiv nekim od brojeva manjih od njega. Razlog za ovo je što se prednost uvek daje paralelizaciji spoljne petlje, a korišćenje ugnježdenih paralelnih regiona nije preporučljivo (postoji mogućnost da ova opcija nije podržana od strane okruženja). While petlja se nalazi u delu koda koji niije eksplicitno deo algoritma, već pripada delu koji se odnosi na testiranje rada programa, zbog ovoga nije rađena paralelizacija.

1.2.2 Način paralelizacije

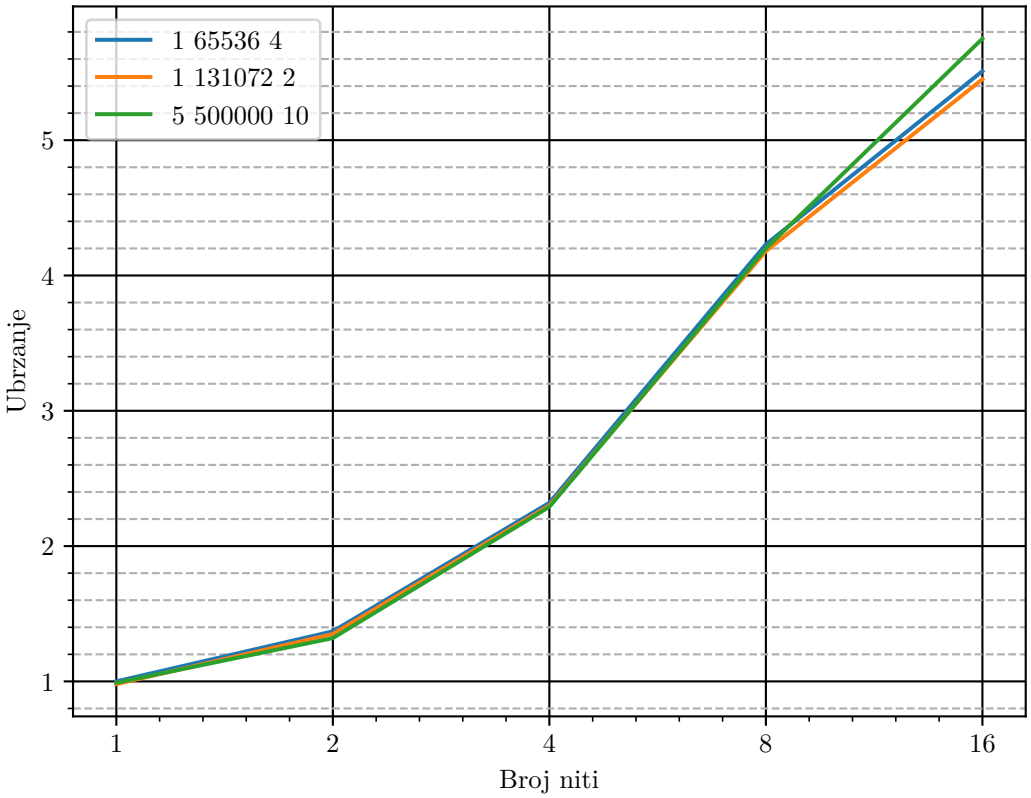
Paralelizacija je rađena korišćenjem paralelnog regiona, koji obuhvata dve navedene for petlje. Bitno je napomenuti da je na promenljivoj total primenjena redukcija po operaciji sabiranja. Razlog za ovo je što total predstavlja deljenu promeljivu, koja predstavlja broj prostih brojeva i kao takvoj pristupaće joj sve niti.
Podela posla je rađena ručno, kao što je traženo u tekstu zadatka, vodeći računa da bude maksimalno ravnomerna. Ovo je garantovano tako što je podela urađena tako da razlika u broju iteracija koje bilo koje dve niti obrađuju maksimalno jedan.

1.3 Rezultati

1.3.1 Logovi izvršavanja

```
Running task: dz1z1
Running variant: prime_v0seq
Running test case: 1 of 3 (1 65536 4) N=1 3 times
    AVG Time: 0.2726
Running test case: 2 of 3 (1 131072 2) N=1 3 times
    AVG Time: 1.2808
Running test case: 3 of 3 (5 500000 10) N=1 3 times
    AVG Time: 12.3583
Running variant: prime_v1par
Running test case: 1 of 3 (1 65536 4) N=1 3 times
    AVG Time: 0.2724 AVG Speedup: 1.00 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=2 3 times
    AVG Time: 0.1993 AVG Speedup: 1.37 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=4 3 times
    AVG Time: 0.1173 AVG Speedup: 2.32 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=8 3 times
    AVG Time: 0.0644 AVG Speedup: 4.23 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=16 3 times
    AVG Time: 0.0495 AVG Speedup: 5.51 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=1 3 times
    AVG Time: 1.3045 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=2 3 times
    AVG Time: 0.9475 AVG Speedup: 1.35 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=4 3 times
    AVG Time: 0.5574 AVG Speedup: 2.30 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=8 3 times
    AVG Time: 0.3066 AVG Speedup: 4.18 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=16 3 times
    AVG Time: 0.2350 AVG Speedup: 5.45 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=1 3 times
    AVG Time: 12.5274 AVG Speedup: 0.99 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=2 3 times
    AVG Time: 9.3638 AVG Speedup: 1.32 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=4 3 times
    AVG Time: 5.3965 AVG Speedup: 2.29 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=8 3 times
    AVG Time: 2.9417 AVG Speedup: 4.20 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=16 3 times
    AVG Time: 2.1505 AVG Speedup: 5.75 ALL Tests PASSED
```

1.3.2 Grafici ubrzanja



Slika 1: Grafik zavisnosti ubrzanja od broja niti za različite test primere

1.3.3 Diskusija dobijenih rezultata

Za tumačenje rezultata veoma je važno napomenuti da server na kome se programi izvršavaju ima 16 logičkih jezgara. U skladu sa ovim dobijeno je da najveće ubrzanje izvršavanja upravo kada se paralelizacija radi sa 16 niti. Kako se broj niti smanjuje tako opada u ubrzanje. Takođe, ukoliko idemo preko 16 niti dešava se time sharing.
Na performanse utiče i sam problem, jer je poznato da raspored prostih brojeva nije uniforman, pa tako bilo koja statička raspodela posla ne daje najbolje performanse.

2 PROBLEM 2 - PRIME

2.1 Tekst problema

Prethodni program paralelizovati korišćenjem direktiva za podelu posla (*worksharing* direktive). Program testirati sa parametrima koji su dati u datoteci `run`. [1, N]

2.2 Delovi koje treba paralelizovati

2.2.1 Diskusija

Kako je program koji se paralelizuje isti kao u prethodnom zadatku ponovo su mogući delovi za paralelizaciju dve for i jedna while petlja. Kao i ranije while petlja nije paralelizovana, jer ne pripada kodu samog algoritma već učestvuje u testiranju rada programa. Paralelizovana je spoljna for petlja, dok unutrašnja i ovog puta nije i ako je kao odredbu for worksharing direktive moguće koristiti collapse. Razlog za ovo je što bi na ovaj način dobili zavisnost između iteracija.

2.2.2 Način paralelizacije

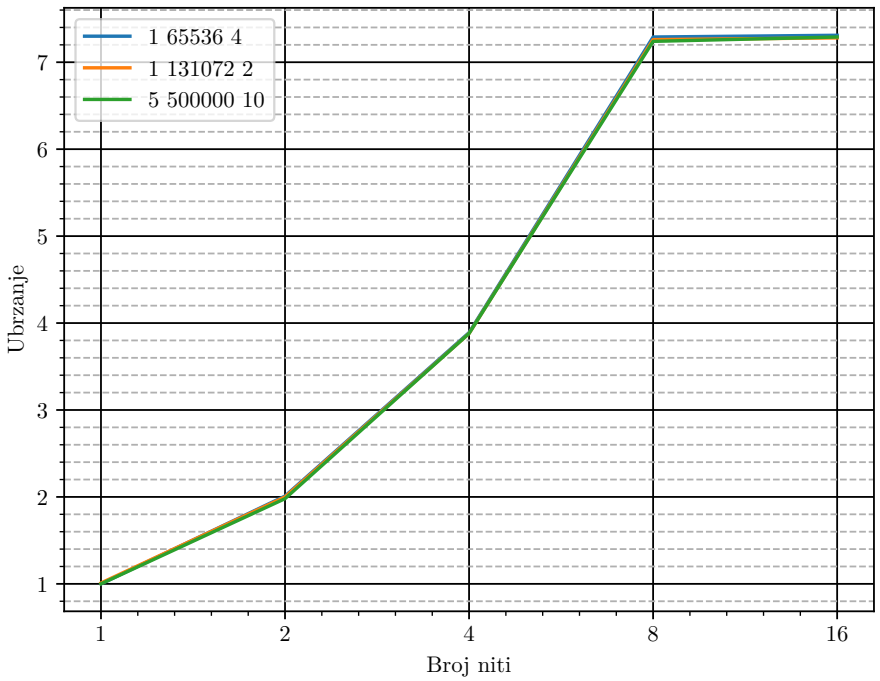
Za paralelizaciju korišćena je worksharing direktiva for, pri čemu je ponovo rađena redukcija po operaciji sabiranja za promenljivu total. Takođe, urađena je optimizacija promenom načina raspoređivanja posla sa podrazumevanog statičkog na dinamički pri čemu svaka nit u jednom trenutku uzima po jednu iteraciju petlje. Kako sam raspored prostih brojeva nije uniforman statička raspodela nam ne odgovara, pa promena načina raspodele donosi značajno ubrzanje.

2.3 Rezultati

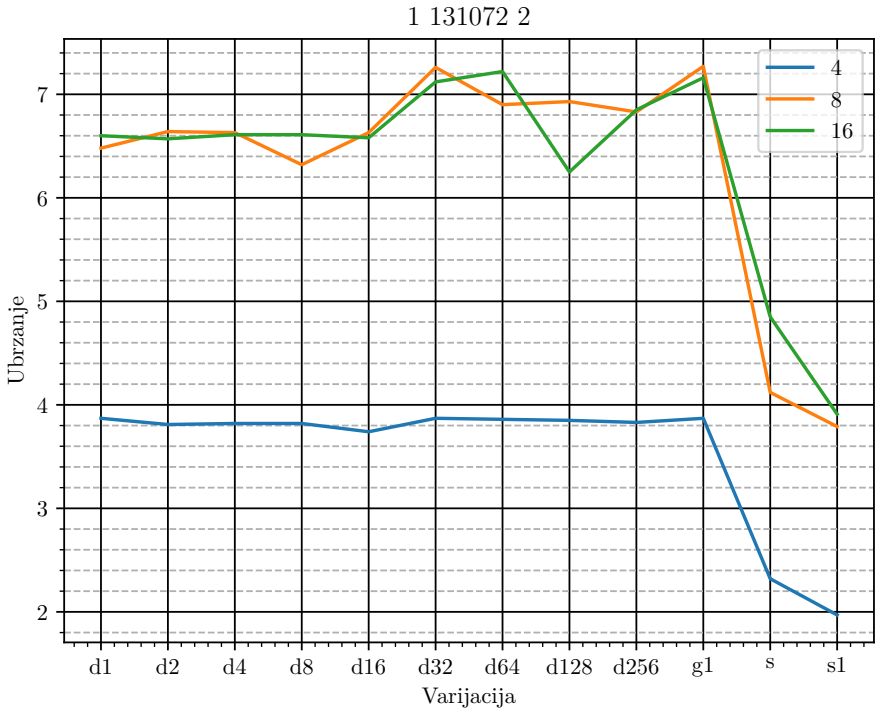
2.3.1 Logovi izvršavanja

```
Running task: dz1z2
Running variant: prime_v0seq
Running test case: 1 of 3 (1 65536 4) N=1 3 times
    AVG Time: 0.2743
Running test case: 2 of 3 (1 131072 2) N=1 3 times
    AVG Time: 1.2934
Running test case: 3 of 3 (5 500000 10) N=1 3 times
    AVG Time: 12.4910
Running variant: prime_v1par
Running test case: 1 of 3 (1 65536 4) N=1 3 times
    AVG Time: 0.2742 AVG Speedup: 1.00 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=2 3 times
    AVG Time: 0.1367 AVG Speedup: 2.01 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=4 3 times
    AVG Time: 0.0706 AVG Speedup: 3.89 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=8 3 times
    AVG Time: 0.0376 AVG Speedup: 7.29 ALL Tests PASSED
Running test case: 1 of 3 (1 65536 4) N=16 3 times
    AVG Time: 0.0375 AVG Speedup: 7.31 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=1 3 times
    AVG Time: 1.2859 AVG Speedup: 1.01 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=2 3 times
    AVG Time: 0.6472 AVG Speedup: 2.00 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=4 3 times
    AVG Time: 0.3336 AVG Speedup: 3.88 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=8 3 times
    AVG Time: 0.1780 AVG Speedup: 7.26 ALL Tests PASSED
Running test case: 2 of 3 (1 131072 2) N=16 3 times
    AVG Time: 0.1777 AVG Speedup: 7.28 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=1 3 times
    AVG Time: 12.4938 AVG Speedup: 1.00 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=2 3 times
    AVG Time: 6.3018 AVG Speedup: 1.98 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=4 3 times
    AVG Time: 3.2176 AVG Speedup: 3.88 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=8 3 times
    AVG Time: 1.7247 AVG Speedup: 7.24 ALL Tests PASSED
Running test case: 3 of 3 (5 500000 10) N=16 3 times
    AVG Time: 1.7135 AVG Speedup: 7.29 ALL Tests PASSED
```

2.3.2 Grafici ubrzanja



Slika 2: Grafik zavisnosti ubrzanja od broja niti za različite test primere



Slika 3: Grafik zavisnosti ubrzanja od algoritma raspoređivanja

2.3.3 Diskusija dobijenih rezultata

Promenom podele posla sa statičke, na dinamičku primećeno je značano poboljšanje u performansama, što odgovara prirodi problema. Za testiranje sa 1 do 8 niti možemo da primetimo linearno ubrzanje u skladu sa brojem korišćenih jezgara, dok za 16 niti vidimo da ubrzanje ulazi u zasićenje. Takođe, ovo je potvrđeno u drugom grafiku gde poredimo ponašanje programa pri različitim raspodelama posla. Treba napomenuti da pri izvršavanju programa sa 16 niti pozadinski procesi imaju veliki uticaj na vreme izvršavanja, kako sam server ima 16 logičkih jezgara. Sa gorepomenutog grafika se vidi da je optimalna raspodela dynamic sa chunksize-om 32 ili 64, kao i guided. Kako su prosti brojevi koncentrisani na početku intervala ima smisla koristiti i guided raspored.

3 PROBLEM 3 - FEYMAN

3.1 Tekst problema

Paralelizovati program koji vrši izračunavanje 3D Poasonove jednačine korišćenjem Feyman-Kac algoritma. Algoritam stohastički računa rešenje parcijalne diferencijalne jednačine krenuvši N puta iz različitih tačaka domena. Tačke se kreću po nasumičnim putanjama i prilikom izlaska iz granica domena kretanje se zaustavlja računajući dužinu puta do izlaska. Proces se ponavlja za svih N tačaka i konačno aproksimira rešenje jednačine. Program se nalazi u datoteci **`feyman.c`** u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]

3.2 Delovi koje treba paralelizovati

3.2.1 Diskusija

Delovi koda koji se mogu paralelizovati su tri ugnježdene for petlje nad kojima se može primeniti paralelizacija pomoću odredbe worksharing direktive collapse. Takođe, postoje dodatne petlje unutar njihovog tela, ali one nisu razmatrane zbog ugnježdenog paralelizma,

3.2.2 Način paralelizacije

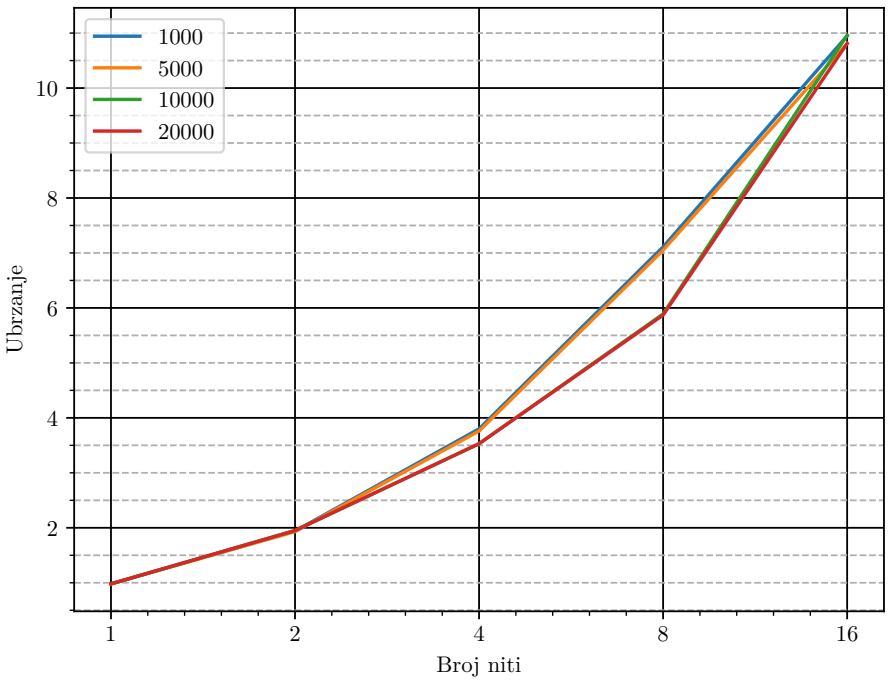
Paralelizacija je postignuta korišćenjem worksharing direktive for i njene odredbe collapse. Uz ovo primenjena je redukcija po operaciji sabiranja na promenljive err i n_inside. Kako bi se izbegao gubitak performansi pri pristupanju generatoru slučajnih brojeva svakoj niti je dodeljen njen privatni seed inicijalizovan na istu početnu vrednost. Testiranjem rešenja zaključeno je da dinamički raspored posla više odgovara ovom problemu.

3.3 Rezultati

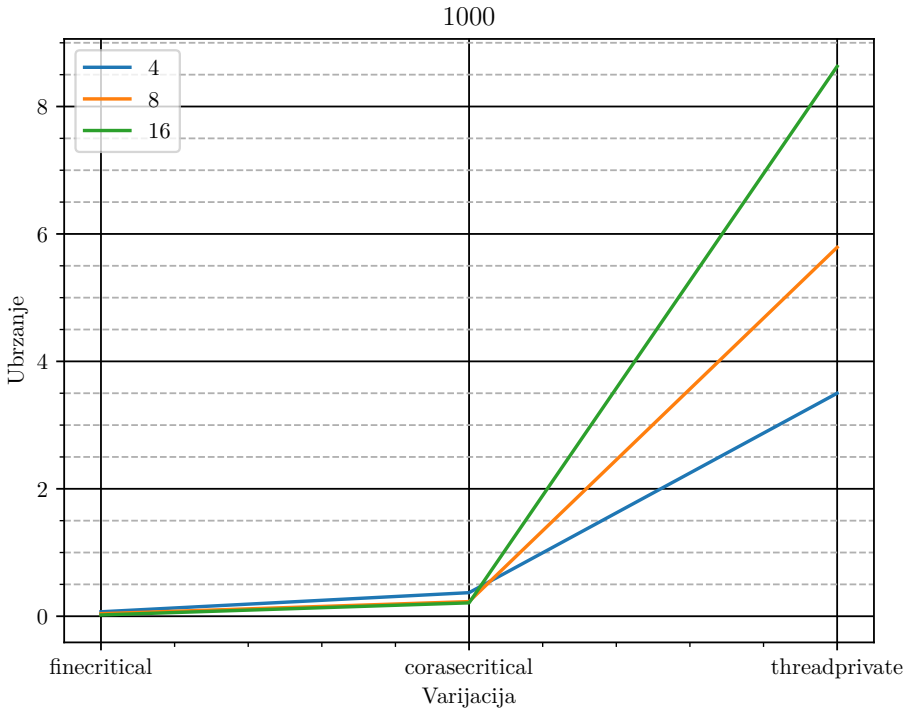
3.3.1 Logovi izvršavanja

```
Running variant: feyman_v0seq
Running test case: 1 of 4 (1000) N=1 3 times
    AVG Time: 3.0026
Running test case: 2 of 4 (5000) N=1 3 times
    AVG Time: 14.8497
Running test case: 3 of 4 (10000) N=1 3 times
    AVG Time: 30.1214
Running test case: 4 of 4 (20000) N=1 3 times
    AVG Time: 60.2822
Running variant: feyman_v1par
Running test case: 1 of 4 (1000) N=1 3 times
    AVG Time: 3.0764 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=2 3 times
    AVG Time: 1.5493 AVG Speedup: 1.94 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=4 3 times
    AVG Time: 0.7902 AVG Speedup: 3.80 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=8 3 times
    AVG Time: 0.4222 AVG Speedup: 7.11 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=16 3 times
    AVG Time: 0.2742 AVG Speedup: 10.95 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=1 3 times
    AVG Time: 15.2299 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=2 3 times
    AVG Time: 7.6995 AVG Speedup: 1.93 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=4 3 times
    AVG Time: 3.9482 AVG Speedup: 3.76 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=8 3 times
    AVG Time: 2.1063 AVG Speedup: 7.05 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=16 3 times
    AVG Time: 1.3731 AVG Speedup: 10.81 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=1 3 times
    AVG Time: 30.7732 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=2 3 times
    AVG Time: 15.4221 AVG Speedup: 1.95 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=4 3 times
    AVG Time: 8.5324 AVG Speedup: 3.53 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=8 3 times
    AVG Time: 5.1175 AVG Speedup: 5.89 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=16 3 times
    AVG Time: 2.7478 AVG Speedup: 10.96 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=1 3 times
    AVG Time: 61.7780 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=2 3 times
    AVG Time: 30.8633 AVG Speedup: 1.95 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=4 3 times
    AVG Time: 17.0766 AVG Speedup: 3.53 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=8 3 times
    AVG Time: 10.2720 AVG Speedup: 5.87 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=16 3 times
    AVG Time: 5.5769 AVG Speedup: 10.81 ALL Tests PASSED
```

3.3.2 Grafici ubrzanja



Slika 4: Grafik zavisnosti ubrzanja od broja niti za različite test primere



Slika 5: Grafik zavisnosti ubrzanja od načina pristupa promenljivoj seed

3.3.3 Diskusija dobijenih rezultata

Sa grafika se vidi približno linearno poboljšanje ubrzanja u skladu sa brojem iskorišćenih niti. Takođe, veoma važan aspekt rešavanja ovog problema je bila promenljiva seed. Prva varijanta bila je da se kritičnim regionom zaštiti pristup ovoj promenljivoj u samoj funkciji r8_uniform_01. Ovaj pristup dao je usporenje kao što se vidi sa drugog grafika, jer svaka nit ulazi u funkciju od 3 do 6 puta i funkcija postaje usko grlo. Druga varijanta je da se kritičnim regionom zaštite svi pristupi jedne niti pomenutoj funkciji. Tada konflikt u pristupu između niti postaje manji pa su performase u odnosu na prethodnu varijantu bolje. Međutim, i dalje se program izvršava sporije od sekvencijalnog. Treća varijanta je da svaka od niti dobije svoj privatni seed, pa da posao koji ta nit obrađuje deli promenljivu. Ovde se dobija ubrzanje na uštrb korektnosti logike, a treba napomenuti da ovo nema velikog uticaja jer je sekvencijalni kod deterministički po pitanju promene seed-a, dok bilo kakav paralelan kod to ne može da zadovolji.

4 PROBLEM 4 - FEYMAN

4.1 Tekst problema

Rešiti prethodni problem korišćenjem koncepta poslova (*tasks*). Obratiti pažnju na eventualnu potrebu za sinhronizacijom i granularnost poslova. Program testirati sa parametrima koji su dati u datoteci `run`. [1, N]

4.2 Delovi koje treba paralelizovati

4.2.1 Diskusija

Kako je u ovom zadatku potrebno koristiti task-ove za paralelizaciju odlučeno je da deo koda koji se nalazi u telu petlji paralelizovanih u prethodnom zadatku bude bude jedan posao. Ovo znači da same petlje nisu paralelizovane, što je urađeno kako bi de dobila adekvatna granularnost poslova. Odnosno ukoliko bi task bio jedna cela obuhvatajuća for petlja bilo bi premalo taskova. Takođe, ukoliko bi task smanjili na jednu iteraciju for petlje trenutno obuhvaćene task-om posao bi bio suviše mali.

4.2.2 Način paralelizacije

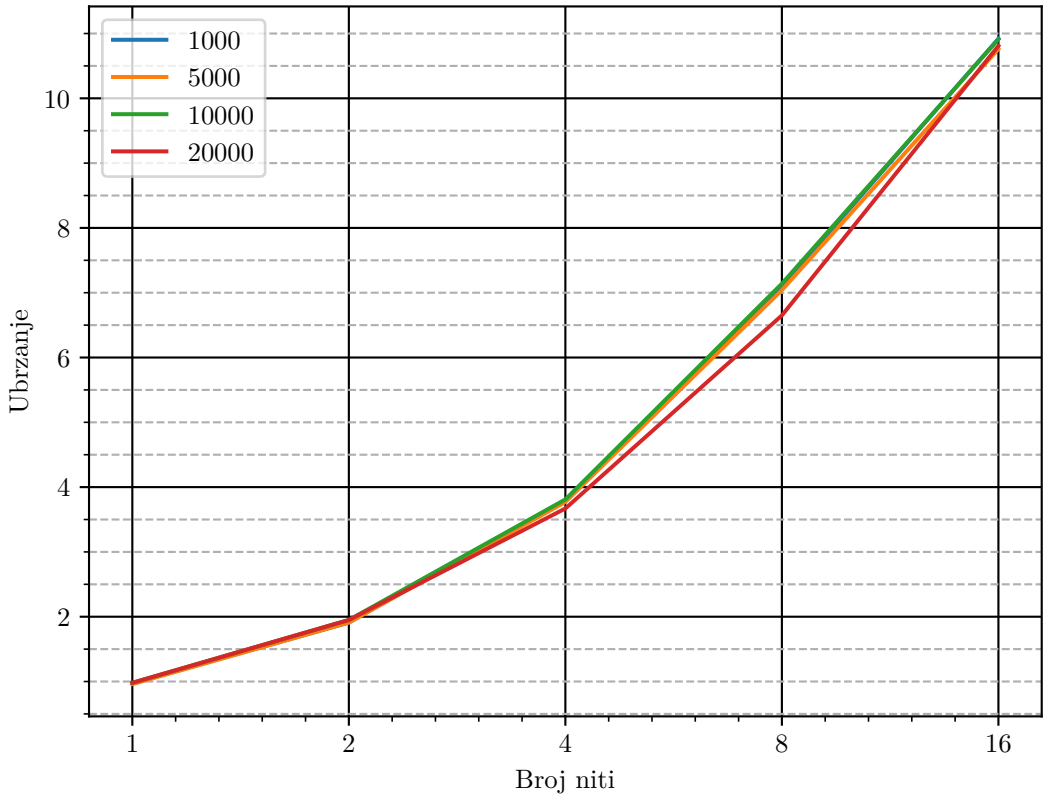
Primećeno da je većina iteracija petlje kratka zbog izvršavanja continue naredbe, pa je jedan posao redukovan na deo koda koji se nalazi ispod nje. Izdavanje poslova se radi iz jedne niti, a sam paralelni region se definiše iznad spoljne for petlje kako bi se smanjili režijski troškovi. Za generator brojeva primenjena je ista strategija kao u prethodnom zadatku, što u ovom slučaju zahteva da promenljiva seed bude threadprivate. Redukciona promenljiva err iz prošlog zadatka i dalje će biti deljena između niti, pa je za korektnost pristupa korišćena atomic direktiva, dok promenljiva n_inside više ne pripada paralelizovanom delu koda.

4.3 Rezultati

4.3.1 Logovi izvršavanja

```
Running variant: feyman_v0seq
Running test case: 1 of 4 (1000) N=1 3 times
    AVG Time: 3.0026
Running test case: 2 of 4 (5000) N=1 3 times
    AVG Time: 14.8497
Running test case: 3 of 4 (10000) N=1 3 times
    AVG Time: 30.1214
Running test case: 4 of 4 (20000) N=1 3 times
    AVG Time: 60.2822
Running variant: feyman_v1par
Running test case: 1 of 4 (1000) N=1 3 times
    AVG Time: 3.0912 AVG Speedup: 0.97 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=2 3 times
    AVG Time: 1.5744 AVG Speedup: 1.91 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=4 3 times
    AVG Time: 0.7892 AVG Speedup: 3.80 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=8 3 times
    AVG Time: 0.4228 AVG Speedup: 7.10 ALL Tests PASSED
Running test case: 1 of 4 (1000) N=16 3 times
    AVG Time: 0.2749 AVG Speedup: 10.92 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=1 3 times
    AVG Time: 15.4184 AVG Speedup: 0.96 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=2 3 times
    AVG Time: 7.7564 AVG Speedup: 1.91 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=4 3 times
    AVG Time: 3.9423 AVG Speedup: 3.77 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=8 3 times
    AVG Time: 2.1098 AVG Speedup: 7.04 ALL Tests PASSED
Running test case: 2 of 4 (5000) N=16 3 times
    AVG Time: 1.3802 AVG Speedup: 10.76 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=1 3 times
    AVG Time: 30.6522 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=2 3 times
    AVG Time: 15.4195 AVG Speedup: 1.95 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=4 3 times
    AVG Time: 7.9048 AVG Speedup: 3.81 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=8 3 times
    AVG Time: 4.2168 AVG Speedup: 7.14 ALL Tests PASSED
Running test case: 3 of 4 (10000) N=16 3 times
    AVG Time: 2.7609 AVG Speedup: 10.91 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=1 3 times
    AVG Time: 61.6934 AVG Speedup: 0.98 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=2 3 times
    AVG Time: 30.9420 AVG Speedup: 1.95 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=4 3 times
    AVG Time: 16.4067 AVG Speedup: 3.67 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=8 3 times
    AVG Time: 9.0703 AVG Speedup: 6.65 ALL Tests PASSED
Running test case: 4 of 4 (20000) N=16 3 times
    AVG Time: 5.5753 AVG Speedup: 10.81 ALL Tests PASSED
```

4.3.2 Grafici ubrzanja



Slika 6: Grafik zavisnosti ubrzanja od broja niti za različite test primere

4.3.3 Diskusija dobijenih rezultata

Pri povećanju broja niti dobija se približno linearno poboljšanje performansi programa. Kao što je napomenuto u sekciji za način paralelizacije, task ne obuhvata deo koda iznad continue naredbe. Zbog same naredbe ovo nije ni moguće uraditi, ali i ako bi bilo korektno ova granularnost posla nije optimalna. Razlog je što se testiranjem dobija da za 1056 iteracija petlje continue ne izvrši samo njih 364.

5 PROBLEM 5 - MOLDYN

5.1 Tekst problema

Paralelizovati jednostavan program koji se bavi molekularnom dinamikom. Kod predstavlja simulaciju molekularne dinamike argonovog atoma u ograničenom prozoru (prostoru) sa periodičnim graničnim uslovima. Atomi se inicijalno nalaze raspoređeni u pravilnu mrežu, a zatim se tokom simulacije dešavaju interakcije između njih. U svakom koraku simulacije u glavnoj petlji se dešava sledeće:

- Čestice (atomi) se pomeraju zavisno od njihovih brzina i brzine se parcijalno ažuriraju u pozivu funkcije `domove`.
- Sile koje se primenjuju na nove pozicije čestica se izračunavaju; takođe, akumuliraju se prosečna kinetička energija (*virial*) i potencijalna energija u pozivu funkcije `forces`.
- Sile se skaliraju, završava ažuriranje brzine i izračunavanje kinetičke energije u pozivu funkcije `mkekin`.
- Prosečna brzina čestice se računa i skaliraju temperature u pozivu funkcije `velavg`.
- Pune potencijalne i prosečne kinetičke energije (*virial*) se računaju i ispisuju u funkciji `prnout`.

Program se nalazi u direktorijumu `MolDyn` u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke `main.c` i `forces.c`, jer se u njima provodi najviše vremena. Analizirati dati kod i obratiti pažnju na redukcione promenljive unutar datoteke `forces.c`. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti kritične sekcije ili atomske operacije. [1, N]

5.2 Delovi koje treba paralelizovati

5.2.1 Diskusija

U ovom zadatku za paralelizaciju su posmatrane samo funkcije main i forces. U main funkciji primećena je jedna for petlja, koja se ne može paralelizovati zbog zavisnosti po podacima. U forces funkciji primećene su dve ugnježdene for petlje. Primena collapse odredbe nije moguća, pa je doneta oodluka da se paralelizuje samo spoljna petlja.

5.2.2 Način paralelizacije

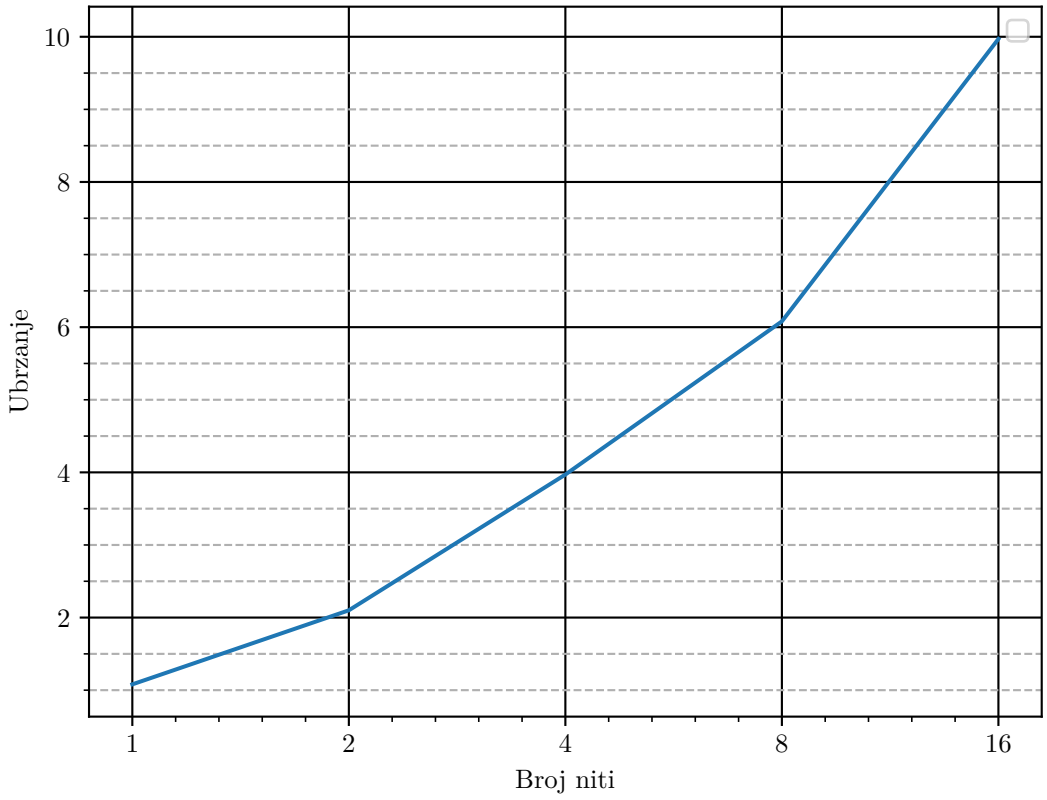
Paralelizacija je urađena primenom for worksharing direktive, pri čemu su promenljive vir i epot redukovane operacijom sabiranja. Takođe, kako za nizove nije podržana redukcija za sve pristupe elementima niza f korišćena je atomic direktiva. Testiranjem je utvrđeno da problemu više odgovara dinamički raspored posla, pri čemu jedna nit u jednom trenutku uzima po 20 iteracija petlje.

5.3 Rezultati

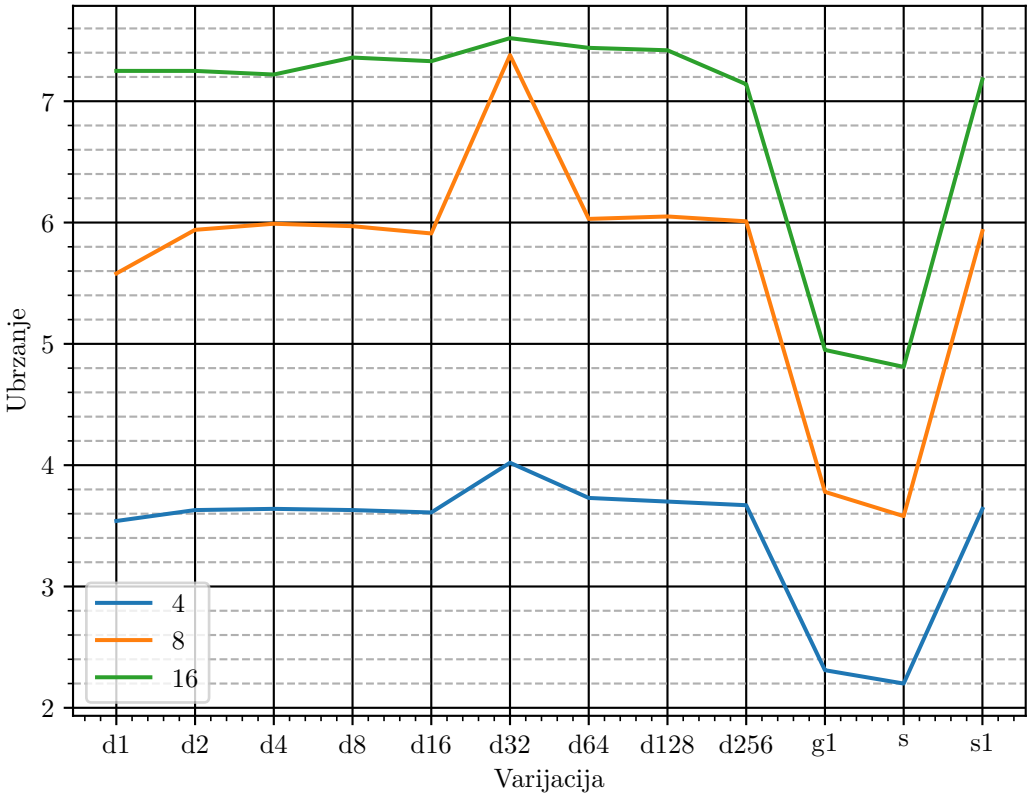
5.3.1 Logovi izvršavanja

```
Running task: dz1z5
Running variant: md_v0seq
Running test case: 1 of 1 () N=1 3 times
    AVG Time: 6.0875
Running variant: md_v1par
Running test case: 1 of 1 () N=1 3 times
    AVG Time: 5.6340 AVG Speedup: 1.08 ALL Tests PASSED
Running test case: 1 of 1 () N=2 3 times
    AVG Time: 2.8925 AVG Speedup: 2.10 ALL Tests PASSED
Running test case: 1 of 1 () N=4 3 times
    AVG Time: 1.5320 AVG Speedup: 3.97 ALL Tests PASSED
Running test case: 1 of 1 () N=8 3 times
    AVG Time: 1.0011 AVG Speedup: 6.08 ALL Tests PASSED
Running test case: 1 of 1 () N=16 3 times
    AVG Time: 0.6106 AVG Speedup: 9.97 ALL Tests PASSED
```

5.3.2 Grafici ubrzanja



Slika 7: Grafik zavisnosti ubrzanja od broja niti za različite test primere



Slika 8: Grafik zavisnosti ubrzanja od strategije raspoređivanja

5.3.3 Diskusija dobijenih rezultata

Dodavanje broja niti koje učestvuju u izvršavanju programa približno linearno poboljšava performanse. Testiranje je utvrđeno da su najbolje performanse dobijene za dinamički raspored posla sa chunksize-om 20, što potvrđuje u drugi grafik na kome su upoređeni različiti rasporedi.