UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

MULTIPROCESORKI SISTEMI (13S114MUPS, 13E114MUPS)

DOMAĆI ZADATAK 2 – MPI

Izveštaj o urađenom domaćem zadatku

Predmetni asistent:

doc. dr Marko Mišić

Studenti:

Bogdan Bebić 2017/0011

Uroš Krstić 2017/0228

Beograd, decembar 2020.

SADRŽAJ

1. PROBLEM 1 – RAČUNANJE BROJA Π (PI)

1.1. Tekst problema

Paralelizovati program koji izračunava vrednost broja PI korišćenjem formule:

$$\pi = 4 * \sum_{k=1}^{n} \frac{(-1)^{k+1}}{2k-1}$$

Tačnost izračunavanja direktno zavisi od broja iteracija, a zbog malog radijusa konvergencije serija konvergira veoma sporo. Program se nalazi u datoteci **piCalc.c** u arhivi koja je priložena uz ovaj dokument. Proces sa rangom 0 treba da učita ulazne podatke, raspodeli posao ostalim procesima, na kraju prikupi dobijene rezultate i ravnopravno učestvuje u obradi. Za razmenu podataka, koristiti rutine za kolektivnu komunikaciju. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

1.2. Delovi koje treba paralelizovati

1.2.1. Diskusija

Jedino mesto u programu koje se može paralelizovati je glavna for petlja koja radi sumu faktora koji čine vrednost broja pi.

1.2.2. Način paralelizacije

Paralelizacije je rađena ručno – rutinama za kolektivnu komunikaciju. Ovo je urađeno podelom posla prema id-u niti, svakoj niti je dodeljen određeni chunk posla – ovo je implementacija poznatog SPMD pattern-a paralelnog programiranja.

1.3. Rezultati

Paralelizacijom ovog programa na više hardverskih niti se dobija ubrzanje koje je prikazano na grafiku (Slika 1) – ovo ubrzanje se vidi u slučaju većeg posla.

1.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

```
$ ./z1 1000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000 terms
   Our estimate of pi = 3.14159165358977
   Ref estimate of pi = 3.14159265358979
-----------------------Parallel execution-----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000 terms
   Our estimate of pi = 3.14159165358978
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.004619s
Parallel elapsed time: 0.001044s
Test PASSED

$ ./z1 10000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000 terms
   Our estimate of pi = 3.14159255358979
   Ref estimate of pi = 3.14159265358979
-----------------------Parallel execution-----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000 terms
   Our estimate of pi = 3.14159255358974
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.035823s
Parallel elapsed time: 0.015870s
Test PASSED

$ ./z1 100000000
----------------------Sequential execution----------------------
```

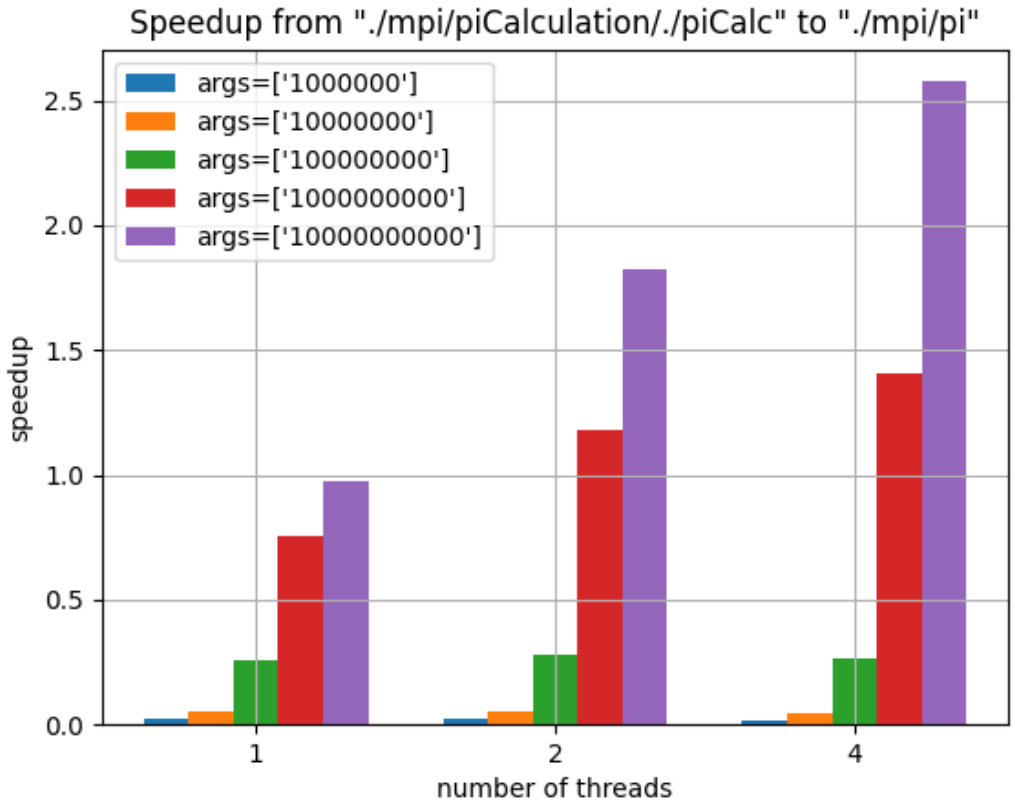
```
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 100000000 terms
   Our estimate of pi = 3.14159264358933
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 100000000 terms
   Our estimate of pi = 3.14159264358982
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.321339s
Parallel elapsed time: 0.079651s
Test PASSED

$ ./z1 1000000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000000 terms
   Our estimate of pi = 3.14159265258805
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000000 terms
   Our estimate of pi = 3.14159265258921
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 3.185368s
Parallel elapsed time: 0.795850s
Test PASSED

$ ./z1 10000000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000000 terms
```

```
   Our estimate of pi = 3.14159265348835
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000000 terms
   Our estimate of pi = 3.14159265348827
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 31.833506s
Parallel elapsed time: 7.963139s
Test PASSED
```

Listing 1. Logovi izvršavanja računanja pi

1.3.2. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

1.3.3. Diskusija dobijenih rezultata

Paralelizacija na više niti je donela ubrzanje za posao koji je dovoljno veliki – ovo ubrzanje se najbolje vidi kada imamo više od 1000000 iteracije petlje. Program je embarrassingly parallel i svaka nit dobija određeni chunk iteracija za računanje. Za pokretanje na jednoj niti se vidi vrlo malo usporenje – ovo je očekivano – to usporenje je uzrokovano režijskim troškovima razmena poruka.

2. PROBLEM 2 – *NEEDLEMAN-WUNSCH* ALGORITAM

2.1. Tekst problema

Paralelizovati program koji vrši poravnavanje bioloških sekvenci korišćenjem *Needleman-Wunsch* algoritma. Algoritam predstavlja primenu koncepta dinamičkog programiranja za globalno poravnavanje dve sekvence nukleotida ili aminokiselina (više o algoritmu na adresi: https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm). Program se nalazi u datoteci **needle.c** u arhivi koja je priložena uz ovaj dokument. Paralelizaciju pokušati podelom sekvence za poravnavanje na podsekvence. Ukoliko je moguće, koristiti rutine za neblokirajuću komunikaciju za razmenu poruka. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

2.2. Delovi koje treba paralelizovati

2.2.1. Diskusija

Usled zavisnosti po podacima preko granica iteracija for petlji koje rade obradu matrice (for petlje su do-across tipa), nemoguće je paralelizovati spoljašnje petlje obrade, slično tome ni obrade dve trougaone matrice koje čine matricu koja se obrađuje se ne mogu raditi u paraleli. Uočeno je da unutrašnje petlje obrade trouganih delova matrice rade iteriranje po dijagonalama matrice, a da svaki element na dijagonali ima nezavisnu obradu od ostalih. Ove dve unutrašnje petlje su zbog toga paralelizovane. Kako TRACEBACK deo programa radi sekvencijalnu pretragu sa zavisnostima od prošle iteracije, ovaj deo programa mora ostati sekvencijalan zbog korektnosti izvršavanja.

2.2.2. Način paralelizacije

Paralelizacija je vršena kombinacijom kolektivne i point-to-point komunikacije.

2.3. Rezultati

Paralelizacijom ovog programa se dobija ubrzanje koje je prikazano na grafiku (Slika 2).

2.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

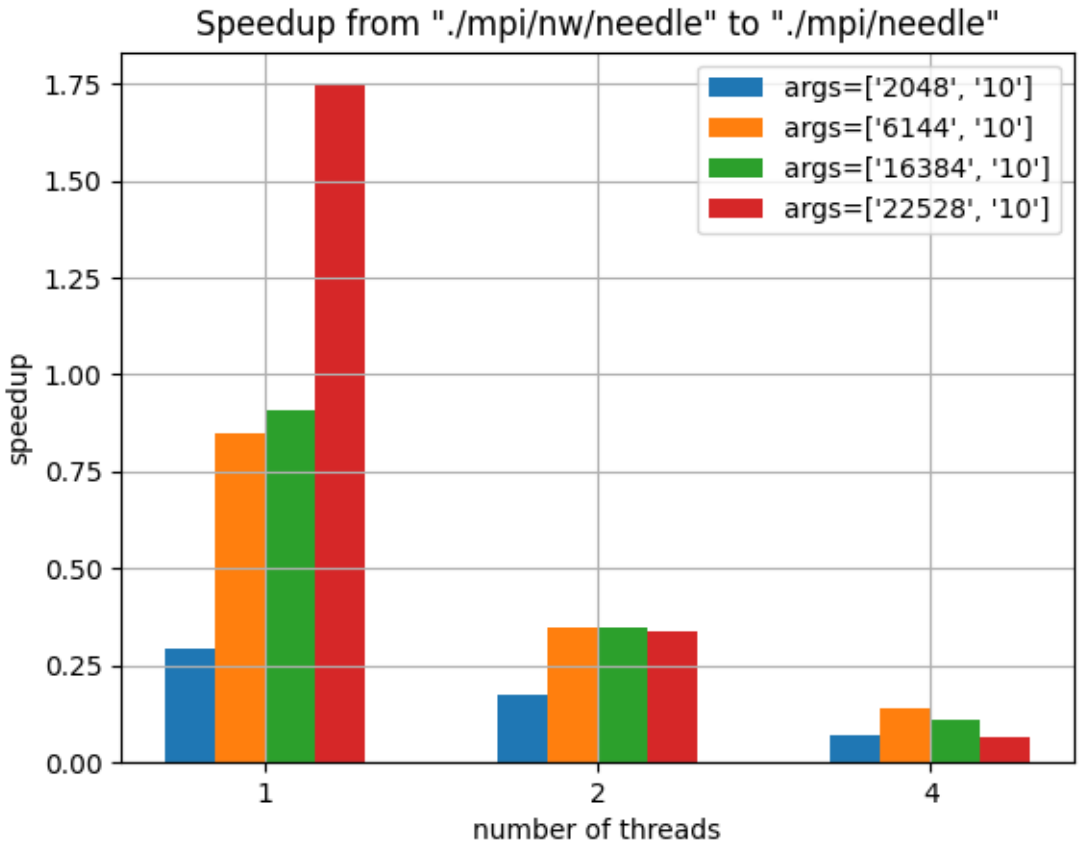
```
$ ./z2 2048 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Start Needleman-Wunsch
Processing top-left matrix
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
Processing bottom-right matrix

Sequential elapsed time: 0.076627s
Parallel elapsed time: 0.225329s
Test PASSED

$ ./z2 6144 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Start Needleman-Wunsch
Processing top-left matrix
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
Processing bottom-right matrix

Sequential elapsed time: 1.055449s
Parallel elapsed time: 2.749505s
Test PASSED
```

```
$ ./z2 16384 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Start Needleman-Wunsch
Processing top-left matrix
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
Processing bottom-right matrix

Sequential elapsed time: 8.248092s
Parallel elapsed time: 21.694646s
Test PASSED

$ ./z2 22528 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Start Needleman-Wunsch
Processing top-left matrix
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
Processing bottom-right matrix

Sequential elapsed time: 16.601136s
Parallel elapsed time: 42.786052s
Test PASSED
```

Listing 2. Logovi izvršavanja Needleman-Wunsch algoritma

2.3.2. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 2. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

2.3.3. Diskusija dobijenih rezultata

Problem sam po sebi nije preterano paralelan pošto koristi dinamičko programiranje koje unosi dosta zavisnoti po podacima preko granica iteracija svih petlji. Zbog ovih zavisnosti potreban je veliki broj relativno malih poruka, ovo unosi dodatnu sinhronizaciju i režijske troškove, pa se nažalost dobija veće usporenje za veći broj niti.

3. PROBLEM 3 – INTERAKCIJA ČVRSTIH TELA (*N-BODY*)

3.1. Tekst problema

Paralelizovati program koji simulira problem interakcije čvrstih tela u dvodimenzionalnom prostoru (*n-body* problem). Tela interaguju putem gravitacione sile na osnovu sopstvene mase, pozicije u prostoru i trenutne brzine. Program se nalazi u direktorijumu **nbody** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih je od interesa datoteka **nbody.c**. Analizirati dati kod i obratiti pažnju na način izračunavanja sila i energija. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim energijama i poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **run**. [1, N]

3.2. Delovi koje treba paralelizovati

3.2.1. Diskusija

Usled zavisnosti po podacima preko granica iteracija for petlje koja radi glavnu obradu (for petlja je do-across tipa), nemoguće je paralelizovati spoljašnju petlju obrade. Unutrašnje petlje se mogu paralelizovati uz vođenje računa o sinhronizaciji pristupa promenljivama. Primećeno je da se funkcije Compute_force i Compute_energy mogu paralelizovati uz vođenje računa o redukciji promenljivih, ali se Compute_energy radi samo jednom, pa nije od interesa za paralelizaciju prosleđivanjem poruka.

3.2.2. Način paralelizacije

Paralelizacija je vršena kolektivom komunikacijom uz redukciju i preimenovanje niza forces.

3.3. Rezultati

Paralelizacijom ovog programa se dobija ubrzanje koje je prikazano na grafiku (Slika 3) – ovo ubrzanje se vidi u slučaju većeg posla, dok se na manjoj količini posla primećuje usporenje.

3.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

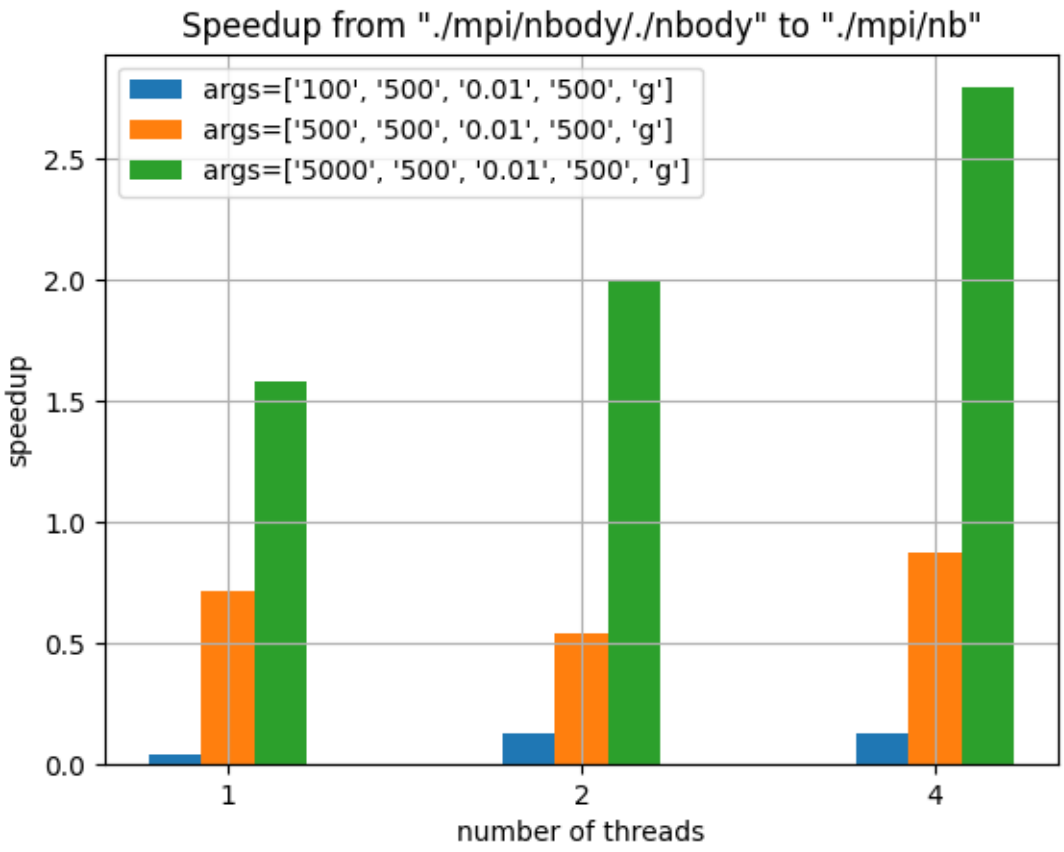
```
$ ./z3 100 500 0.01 500 g
--------------------Sequential execution--------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 3.328300e-02 seconds
----------------------Parallel execution----------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 8.792162e-03 seconds

Sequential elapsed time: 0.033291s
Parallel elapsed time: 0.008796s
Test PASSED

$ ./z3 500 500 0.01 500 g
--------------------Sequential execution--------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 8.138480e-01 seconds
----------------------Parallel execution----------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 1.848829e-01 seconds

Sequential elapsed time: 0.813864s
Parallel elapsed time: 0.184889s
Test PASSED

$ ./z3 5000 500 0.01 500 g
--------------------Sequential execution--------------------
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 8.016041e+01 seconds
----------------------Parallel execution----------------------
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 1.776071e+01 seconds
```

```
Sequential elapsed time: 80.160481s
Parallel elapsed time: 17.760760s
Test PASSED
```

Listing 3. Logovi izvršavanja n-body problema

3.3.2. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 3. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

3.3.3. Diskusija dobijenih rezultata

Primećeno je ubrzanje za veći posao, dok se za manji posao vidi usporenje u odnosu na sekvencijalni program. Ovo se objašnjava time što su paralelizovane unutrašnje petlje obrade što prouzrokuje veliki broj kolektivnih operacija slanja poruka – zato se ovo isplati samo u slučajevima kada unutrašenje for petlje imaju puno posla.

4. PROBLEM 4 – INTERAKCIJA ČVRSTIH TELA (*N-BODY*)

4.1. Tekst problema

Prethodni program paralelizovati korišćenjem manager - worker modela u delu proračuna sila. Proces gospodar (master) treba da učita neophodne podatke, generiše poslove, deli posao ostalim procesima i ispiše na kraju dobijeni rezultat. U svakom koraku obrade, proces gospodar šalje procesu radniku na obradu jednu jedinicu posla čiji veličinu treba pažljivo odabrati. Proces radnik prima podatke, vrši obradu, vraća rezultat, signalizira gospodaru kada je spreman da primi sledeći posao i ponavlja opisani postupak dok ne dobije signal da prekine sa radom. Veličinu jedne jedinice posla prilagoditi karakteristikama programa. Ukoliko je moguće, koristiti rutine za neblokirajuću komunikaciju za razmenu poruka. Program se nalazi u direktorijumu **nbody** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih je od interesa datoteka **nbody.c**. Analizirati dati kod i obratiti pažnju na način izračunavanja sila i energija. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim energijama i poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **run**. [1, N]

4.2. Delovi koje treba paralelizovati

4.2.1. Diskusija

Usled zavisnosti po podacima preko granica iteracija for petlje koja radi glavnu obradu (for petlja je do-across tipa), nemoguće je paralelizovati spoljašnju petlju obrade. Unutrašnje petlje se mogu paralelizovati uz vođenje računa o sinhronizaciji pristupa promenljivama. Primećeno je da se funkcije Compute_force i Compute_energy mogu paralelizovati uz vođenje računa o redukciji promenljivih, ali se Compute_energy radi samo jednom, pa nije od interesa za paralelizaciju prosleđivanjem poruka.

4.2.2. Način paralelizacije

Paralelizacija je vršena kolektivom komunikacijom uz redukciju i preimenovanje niza forces, koristeći manager-worker model raspodele posla.

4.3. Rezultati

Paralelizacijom ovog programa se dobija ubrzanje koje je prikazano na grafiku (Slika 4) – ovo ubrzanje se vidi u slučaju većeg posla, dok se na manjoj količini posla primećuje usporenje. Kako se koristi manager-worker model, paralelni program se ne pokreće sa samo jednim procesom.

4.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

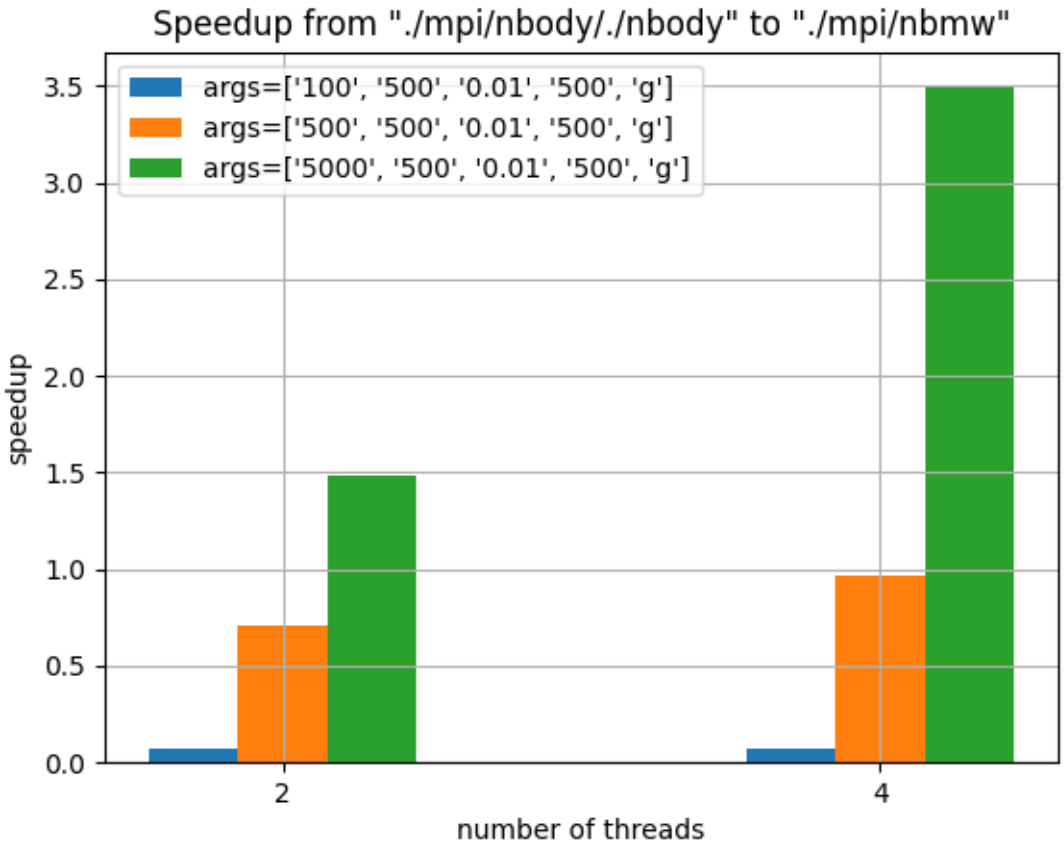
```
$ ./z4 100 500 0.01 500 g
----------------------Sequential execution----------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 3.979802e-02 seconds
----------------------Parallel execution----------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 1.493096e-02 seconds

Sequential elapsed time: 0.039808s
Parallel elapsed time: 0.014937s
Test PASSED

$ ./z4 500 500 0.01 500 g
----------------------Sequential execution----------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 8.155990e-01 seconds
----------------------Parallel execution----------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 1.526811e-01 seconds

Sequential elapsed time: 0.815616s
Parallel elapsed time: 0.152689s
Test PASSED

$ ./z4 5000 500 0.01 500 g
----------------------Sequential execution----------------------
```

```
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 8.031147e+01 seconds
----------------------Parallel execution----------------------
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 1.365096e+01 seconds

Sequential elapsed time: 80.311642s
Parallel elapsed time: 13.651001s
Test PASSED
```

Listing 4. Logovi izvršavanja n-body problema

4.3.2. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 4. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

4.3.3. Diskusija dobijenih rezultata

Primećeno je ubrzanje za veći posao, dok se za manji posao vidi usporenje u odnosu na sekvencijalni program. Ovo se objašnjava time što su paralelizovane unutrašnje petlje obrade što prouzrokuje veliki broj kolektivnih operacija slanja poruka – zato se ovo isplati samo u slučajevima kada unutrašenje for petlje imaju puno posla. Razmatrano je slanje poruka svakom procesu odvojeno point-to-point komunikacijom, ali je primećeno da svaki proces mora dobiti ceo niz, pa bi se ovime samo dodatno unosili režijski troškovi.