Multiprocesorski sistemi

Domaći zadatak 1
Školska godina 2025/2026
OpenMP – paralelizacija direktivama
(10 poena)

Uvod

Cilj prvog domaćeg zadatka je da studentima približi osnovne koncepte rada sa **`OpenMP`** tehnologijom koja omogućava paralelizaciju direktivama na sistemima sa deljenom memorijom. Domaći zadaci se rade **samostalno** ili **u paru**. Rešenja domaćih zadataka i izveštaj svaki student predaje **samostalno** u svoj svn repozitorijum.

Podešavanje okruženja

Za rad sa OpenMP tehnologijom koristiti **`gcc`**/**`g++`** na računaru **`rtidev5.etf.rs`** ili instalirati **`gcc`**/**`g++`** prevodilac na lokalnu Windows mašinu korišćenjem Cygwin, MinGW alata ili u okviru Linux subsystem for Windows. Za rešavanje domaćeg zadatka je potrebno imati **`gcc`**/**`g++`** prevodilac verzije 4.4.0 ili noviji koji podržava OpenMP standard 3.0.

Izveštaj

Uz predati domaći zadatak (izvorne kodove) treba napisati i priložiti kratak izveštaj o izvršenoj paralelizaciji i dobijenim ubrzanjima u odnosu na sekvencijalnu verziju koda. Za svaki rešeni zadatak treba kratko opisati uočena mesta koja je moguće paralelizovati i način paralelizacije. Takođe, potrebno je dati logove izvršenog koda za sve test primere koji se izvršavaju i nalaze se u **`run`** datoteci i nacrtati grafike ubrzanja u odnosu na sekvencijalnu verziju. Na graficima je potrebno dati i rezultate poređenja različitih načina paralelizacije za isti broj niti, ukoliko postoje takvi zahtevi u okviru teksta zadatka. Šablon za pisanje izveštaja se nalazi u okviru sekcije za domaće zadatke predmetnog sajta.

Zadaci

Svaki od programa treba napisati i paralelizovati tako da može biti izvršen sa bilo kojim brojem niti **`N`** u dostupnom OpenMP izvršnom okruženju. Za programe koji će biti izvršavani na samo jednom računaru, smatrati da **`N`** neće biti više od **`N=16`**. Preporučuje se testiranje zadataka sa 1, 2, 4, 8 i 16 niti. Bodovi koje svaki zadatak nosi dati su u zagradama na početku teksta zadatka.

Kod zadataka gde je to zahtevano, korisnik zadaje samo dimenzije problema/nizova/matrica, a sve potrebne ulazne podatke generisati u operativnoj memoriji uz pomoć generatora pseudoslučajnih brojeva iz biblioteke jezika C. Generisani brojevi treba da budu odgovarajućeg tipa u opsegu od **`-MAX`** do **`+MAX`**, gde **`MAX`** ima vrednost 1024. Za sve zadatke je potrebno napisati ili iskoristiti zadatu sekvencijalnu implementaciju odgovarajućeg problema. U cilju postizanja najboljih performansi, dozvoljeno je menjati izvorni sekvencijalni kod. Izvorni kod, ili njegova modifikacija koja postiže najbolje performanse, je potrebno koristiti kao referentnu (*gold*) implementaciju prilikom testiranja programa.

Svaki program treba da:

- Generiše ili koristi već obezbeđene ulazne test primere.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Izvrši paralelnu implementaciju nad zadatim test primerom.
- Ispište vreme izvršavanja sekvencijalne i paralelne implementacije problema.
- Uporedi rezultat sekvencijalne i OpenMP implementacije problema.
- Ispiše "**`Test PASSED`**" ili "**`Test FAILED`**" u zavisnosti da li se rezultat izvršavanja OpenMP implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Poređenje rezultata OpenMP i sekvencijalne implementacije problema izvršiti na kraju sekvencijalnog dela programa. Kod zadataka koji koriste realne tipove (**`float`**, **`double`**) tolerisati maksimalno odsupanje od **`±ACCURACY`** prilikom poređenja rezultata sekvencijalne i OpenMP implementacije. Smatrati da konstanta **`ACCURACY`** ima vrednost 0.01 ili usvojiti razumnu vrednost u skladu sa problemom koji se rešava. **Prilikom rešavanja zadataka voditi računa da se postigne maksimalni mogući paralelizam.** Dozvoljeno je ograničeno preuređivanje dostupnih sekvencijalnih implementacija prilikom paralelizacije. **Ukoliko u nekom zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku i da nastavi da izgrađuje svoje rešenje na temeljima uvedene pretpostavke.**

Dostupne sekvencijalne implementacije se nalaze u arhivi koje se mogu preuzeti na adresi sa predmetnog sajta. Na **`rtidev5.etf.rs`** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: **`wget http://mups.etf.rs/dz/2025-2026/MPS_DZ_2025_2026.zip`**

Raspakivanje: **`unzip MPS_DZ_2025_2026.zip`**

1. **[1]** Paralelizovati program koji formira sliku tačaka koje pripadaju Julia skupu tačaka (https://en.wikipedia.org/wiki/Julia_set). Neka se posmatra skup tačaka *(x, y)* u na pravougaonom domenu $x, y \in [-1{,}5, 1.5]$ i neka važi $z = x+yi$. Julia skup je skup tačaka za koji iteracija $z = z^2 + c$ ne divergira za određene zadate početne uslove. U zadatom programu početni uslov odgovara $c=-0.8+0.156i$. Ukoliko u bilo kom trenutku važi $1000 < |z|$, smatra se da tačka z ne pripada Julia skupu. Program formira sliku u *Targa* (*.tga*) formatu koja se može otvoriti u nekom od namenskih pregledača slika. Program se nalazi u datoteci **`julia.c`** u arhivi koja je priložena uz ovaj dokument, dok se primeri izlaznih datoteka nalaze u direktorijumu **`output`**. 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 **`run`** skripti.
2. **[2]** Prethodni program paralelizovati korišćenjem direktiva za podelu posla (*worksharing direktive*). Program testirati sa parametrima koji su dati u **`run`** skripti.
3. **[1]** 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 **`run`** skripti.
4. **[2]** Rešiti prethodni problem korišćenjem koncepta poslova (*tasks*).

   Program testirati sa parametrima koji su dati u **`run`** skripti.
5. **[4]** Paralelizovati jednostavan program koji se bavi molekularnom dinamikom. Simulacija se bavičesticama (molekulima) i njihovom međusobnom interakcijom u funkciji distance. Čestice nisu prostorno ograničene, a algoritam iterativno rešava sistem diferencijalnih jednačina u diskretnim vremenskim koracima. Na osnovu zadate pozicije i brzine čestice u jednom trenutku, algoritam određuje poziciju i brzinu u narednom trenutku. Kod koji treba paralelizovati se nalazi u datoteci **`md.c`** u arhivi koja je priložena uz ovaj dokument. Analizirati dati kod i obratiti pažnju na funkciju **`compute`** za izračunavanje sila i energija. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti dostupne OpenMP sinhornizacione direktive. Obratiti pažnju na efikasnost paralelizacije.

   Program testirati sa parametrima koji su dati u **`run`** skripti.