Multiprocesorski sistemi

Domaći zadatak 1
Školska godina 2026/2027
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/2026-2027/MPS_DZ_2026_2027.zip`**

Raspakivanje: **`unzip MPS_DZ_2026_2027.zip`**

1. **[1]** Paralelizovati program koji rešava EP (*Embarrassingly Parallel*) problem iz skupa NAS Parallel Benchmarks (https://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks). Program generiše parove uniformnih pseudoslučajnih brojeva korišćenjem linearnog kongruentnog generatora i od njih metodom prihvatanja i odbacivanja (*acceptance-rejection*) formira parove Gausovih slučajnih brojeva. Program zatim računa sume koordinata dobijenih parova i broj parova koji upadaju u svaki od uzastopnih kvadratnih prstenova oko koordinatnog početka. Parovi se generišu u blokovima, a početna vrednost generatora (*seed*) za svaki blok računa se nezavisno, na osnovu rednog broja bloka. Veličina problema zadaje se klasom (**`S`**, **`W`**, **`A`**, **`B`** ili **`C`**) kao argumentom komandne linije, a program na kraju proverava dobijene sume u odnosu na referentne vrednosti za zadatu klasu. Program se nalazi u datoteci **`ep.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 **`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 Floyd-Warshall algoritmom određuje dužine najkraćih puteva između svih parova čvorova usmerenog težinskog grafa. Graf je predstavljen matricom **`path`** dimenzija $N \times N$, čiji element **`path[i][j]`** predstavlja cenu grane od čvora i do čvora j. U k-toj iteraciji spoljne petlje algoritam za svaki par čvorova (i, j) proverava da li je put preko čvora k kraći od do tada pronađenog puta i, ako jeste, ažurira odgovarajući element matrice. Program je preuzet iz skupa PolyBench/C i nalazi se u datoteci **`floyd-warshall.c`** u arhivi koja je priložena uz ovaj dokument. Dimenzija problema N zadaje se kao argument komandne linije. Prilikom paralelizacije obratiti pažnju na zavisnosti koje postoje između iteracija petlji.

   Program testirati sa parametrima koji su dati u **`run`** skripti.
4. **[2]** 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 **`run`** skripti.
5. **[4]** Paralelizovati program koji računa mapu elektrostatičkog potencijala koji stvaraju naelektrisani atomi raspoređeni u trodimenzionalnom prostoru (*Cutoff-limited Coulombic Potential*, CUTCP). Potencijal se računa u tačkama pravilne 3D rešetke sa korakom od 0,5 Å, pri čemu se uzima u obzir samo kratkodometna komponenta potencijala, odnosno doprinos atoma koji se nalaze na rastojanju manjem od 12 Å od posmatrane tačke. Atomi se najpre raspoređuju u ćelije prostorne mreže (*geometric hashing*), a zatim se, ćeliju po ćeliju, doprinos svakog atoma dodaje svim tačkama rešetke koje se nalaze unutar zadatog radijusa. Na kraju se vrednosti u tačkama rešetke koje su bliže od 1 Å nekom atomu postavljaju na nulu. Program je preuzet iz skupa Parboil i nalazi se u direktorijumu **`cutcp`** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **`main.c`**, **`cutcpu.c`** i **`excl.c`**. Analizirati dati kod i obratiti pažnju na način akumuliranja doprinosa

atoma u tačkama rešetke. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti dostupne OpenMP konstrukte. Obratiti pažnju na efikasnost međusobnog isključenja niti i po potrebi ga svesti na što je moguće manju meru uvođenjem pomoćnih struktura podataka. Ulazni test primeri (kocke molekula vode različitih veličina u *PQR* formatu) se nalaze u direktorijumu **`data`**. Verifikaciju paralelizovanog rešenja vršiti nad vrednostima potencijala u svim tačkama rešetke.

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