Multiprocesorski sistemi

Domaći zadatak 1
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.

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 iz opsega navedenog iza postavke zadatka. **`N`** označava maksimalan mogući broj procesa u trenutno dostupnom OpenMP izvršnom okruženju. Za programe koji će biti izvršavani na samo jednom računaru, smatrati da **`N`** neće biti više od **`N=8`**. Preporučuje se testiranje zadataka sa 1, 2, 4 i 8 niti.

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, koja će biti korišćena kao referentna (*gold*) implementacija prilikom testiranja programa.

Svaki program treba da:

- Generiše ili koristi već obezbeđene ulazne test primere.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Izvrši paralelnu, OpenMP 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. **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 **`MPS_DZ1_OpenMP.zip`** ili **`MPS_DZ1_OpenMP.tar.bz2`** koje se mogu preuzeti na adresi http://mups.etf.rs/dz/2021-2022/. Na **`rtidev5.etf.rs`** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: **`wget http://mups.etf.rs/dz/2021-2022/MPS_DZ1_OpenMP.tar.bz2`**

Raspakivanje: **`tar xjvf MPS_DZ1_OpenMP.tar.bz2`**

1. Paralelizovati program koji računa integral funkcije **`F`** na osnovu unutrašnjosti *simplex*-a (https://en.wikipedia.org/wiki/Simplex) u 20 dimenzija korišćenjem Monte Carlo metode. Program se naalazi u datoteci **`simplex.c`**. U izvornom kodu data je matrica eksponenata jednačine i ivica *simplex*-a. Ulazni parametar programa je broj iteracija aproksimacije. 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]

2. Prethodni program paralelizovati korišćenjem direktiva za podelu posla (*worksharing* direktive). Obratiti pažnju na raspodelu opterećenja po nitima i testirati program za različite načine raspoređivanja posla. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]

3. Paralelizovati program koji implementira simulaciju ćelijskog automata *Game of Life*. Simulacija je predstavljena dvodimenzionalnom matricom dimenzija *w x h*, a svaka ćelija *c* može uzeti vrednost 1 ukoliko predstavlja živu ćeliju, a 0 ukoliko je mrtva. Za svaku ćeliju se vrši izračunavanje vrednosti *n* koja predstavlja zbir živih ćelija u susedstvu posmatrane ćelije. Posmatra se osam suseda. Ćelije se rađaju i umiru prema pravilima iz sledeće tabele.

Vrednost C	**Vrednost N**	**Nova vrednost C**	**Komentar**
1	0, 1	0	Usamljena ćelija umire
1	4, 5, 6, 7, 8	0	Ćelija umire usled prenaseljenosti
1	2,3	1	Ćelija živi
0	3	1	Rađa se nova ćelija
0	0, 1, 2, 4, 6, 7, 8	0	Nema promene stanja

   Može se smatrati da su ćelije van opsega posmatrane matrice mrtve. Kod koji treba paralelizovati se nalazi u datoteci **`gameoflife.c`** u arhivi koja je priložena uz ovaj dokument. Program se može prevesti u dve konfiguracije: sa vizuelnim prikazom i bez vizuelnog prikaza, u zavisnosti da li je definisan makro **`LIFE_VISUAL`**. Prevođenje sa vizuelnim prikazom se može izvršiti naredbom **`make visual`**. Paralelizovati konfiguraciju bez vizuelnog prikaza, a vreme meriti na nivou cele simulacije i na nivou jednog izvršavanja funkcije **`evolve`**. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]

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

5. Paralelizovati program koji rešava problem promene temperature na čipu procesora u dvodimenzionalnom prostoru kroz vreme, ako su poznati početna temperatura i granični uslovi. Simulacija rešava seriju diferencijalnih jednačina nad pravilnom mrežom tačaka kojom se aproksimira površina procesora. Svaka tačka u mreži predstavlja prosečnu temperaturu za odgovarajuću površinu na čipu. Mreža tačaka je predstavljena odgovarajućom matricom koja opisuje trenutne temperature. Program se nalazi u direktorijumu **`hotspot`** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih je od interesa datoteka **`hotspot.c`**. Analizirati dati kod i obratiti pažnju na način izračunavanja temperatura. 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. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim temperaturama u poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **`run`**. [1, N]. Kao pomoćno sredstvo, data je i **`python`** skripta koja izlaznu datoteku formatira u **`heatmap`** sliku u PNG formatu.