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.

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 ili MinGW alata. Za rešavanje domaćeg zadatka je potrebno imati **`gcc`**/**`g++`** prevodilac verzije 4.4.0 ili noviji koji podržava OpenMP standard 3.0.

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/2015-2016/. Na **`rtidev5.etf.rs`** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: **`wget http://mups.etf.rs/dz/2015-2016/MPS_DZ1_OpenMP.tar.bz2`**

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

1. Paralelizovati program koji vrši množenje dve kvadratne matrice realnih brojeva u dvostrukoj preciznosti. Program se nalazi u datoteci **`matMul.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]
2. Paralelizovati program koji rešava *pathfinder* problem. *Pathfinder* problem koristi dinamičko programiranje da pronađe put sa najmanjom akumuliranom težinom u 2D mreži (matrici). Put započinje od početne i ide do krajnje vrste matrice krećući se preko elemenata sa najmanjom težinom. U svakom koraku, put se pomera ili pravo ili dijagonalno u odnosu na trenutni element. Program se nalazi u datoteci **`pathfinder.cpp`** u arhivi koja je priložena uz ovaj dokument. Paralelizaciju obaviti korišćenjem direktiva za podelu posla (*worksharing* direktive). Prilikom paralelizacije, obratiti pažnju na zavisnosti koje postoje između iteracija petlje. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]
3. Paralelizovati program koji vrši obilazak grafa po širini. Program se nalazi u datoteci **`hotSpot.c`** u arhivi koja je priložena uz ovaj dokument. Paralelizaciju obaviti korišćenjem direktiva za podelu posla (*worksharing* direktive). Program se nalazi u datoteci **`bfs.cpp`** u arhivi koja je priložena uz ovaj dokument. Obratiti pažnju na balansiranje opterećenja. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**. [1, N]
4. Rešiti prethodni problem korišćenjem koncepta poslova (*tasks*). Obratiti pažnju na zavisnosti koje mogu postojati među poslovima. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**. [1, N]
5. Paralelizovati program koji vrši *k-means* klasterizaciju podataka. Klasterizacija metodom *k-srednjih vrednosti* (eng. *k-means clustering*) je metod koji particioniše *n* objekata u *k* klastera u kojem svaki objekat pripada klasteru sa najbližom srednjom vrednošću. Objekat se sastoji od niza vrednosti - osobina (eng. *features*). Podelom objekata u podklastere, algoritam predstavlja sve objekte pomoću njihovi srednjih vrednosti (tzv. centroida podklastera). Inicijalni centroid za svaki podklaster se bira ili nasumično ili pomoću odgovarajuće heuristike. U svakoj iteratiji, algoritam pridružuje svaki objekat najbližem centroidu na osnovu definisane metrike. Novi centroidi za sledeću iteraciju se izračunavaju usrednjavanjem svih objekata unutar podklastera. Algoritam se izvršava sve dok se makar jedan objekat pomera iz jednog u drugi podklaster. Program se nalazi u direktorijumu **`kmeans`** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **`kmeans.c`**, **`cluster.c`** i **`kmeans_clustering.c`**. Analizirati dati kod i obratiti pažnju na generisanje novih centroida u svakoj iteraciji unutar datoteke **`kmeans_clustering.c`**. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti dostupne OpenMP konstrukte. Obratiti pažnju na efikasnost međusobnog isključenja niti i svesti ga na što je moguće manju meru uvođenjem pomoćnih struktura podataka. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**. [1, N]