

# 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.
- Ispiše 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  $\pm$ **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/2022-2023/>. Na **rtidev5.etf.rs** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: `wget http://mups.etf.rs/dz/2022-2023/MPS_DZ1_OpenMP.tar.bz2`

Raspakivanje: `tar xjvf MPS_DZ1_OpenMP.tar.bz2`

1. Paralelizovati program koji vrši određivanje ukupnog broja prostih brojeva u zadatom opsegu. Program se nalazi u datoteci **prime.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. Prethodni program paralelizovati korišćenjem direktiva za podelu posla (*worksharing* direktive). Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]
3. Paralelizovati program koji vrši izračunavanje 3D [Poissonove jednačine](#) korišćenjem [Feynman-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 datoteci **run**. [1, N]
4. 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 datoteci **run**. [1, N]
5. Paralelizovati jednostavan program koji se bavi molekularnom dinamikom. Kod predstavlja simulaciju molekularne dinamike argonovog atoma u ograničenom prozoru (prostoru) sa periodičnim graničnim uslovima. Atomi se inicijalno nalaze raspoređeni u pravilnu mrežu, a zatim se tokom simulacije dešavaju interakcije između njih. U svakom koraku simulacije u glavnoj petlji se dešava sledeće:
  - Čestice (atomi) se pomeraju zavisno od njihovih brzina i brzine se parcijalno ažuriraju u pozivu funkcije **domove**.
  - Sile koje se primenjuju na nove pozicije čestica se izračunavaju; takođe, akumuliraju se prosečna kinetička energija (*virial*) i potencijalna energija u pozivu funkcije **forces**.
  - Sile se skaliraju, završava ažuriranje brzine i izračunavanje kinetičke energije u pozivu funkcije **mkekin**.
  - Prosečna brzina čestice se računa i skaliraju temperature u pozivu funkcije **velavg**.
  - Pune potencijalne i prosečne kinetičke energije (*virial*) se računaju i ispisuju u funkciji **prnout**.

Program se nalazi u datoteci direktorijumu **Moldyn** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **main.c** i **forces.c**, jer se u njima provodi najviše vremena. Analizirati dati kod i obratiti pažnju na redukcionu promenljivu unutar datoteke **forces.c**. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti kritične sekcije ili atomske operacije. [1, N]