Multiprocesorski sistemi

Domaći zadatak 2
MPI – komunikacija, izvedeni tipovi, grupe i komunikatori
(10 poena)

Uvod

Cilj zadatka je da studente obuči da samostalno podese MPI okruženje i razvijaju osnovne MPI programe korišćenjem rutina za pojedinačnu i kolektivnu komunikaciju, izvedenih tipova podataka, grupa i komunikatora.

Podešavanje okruženja

Podešavanja okruženja izvršiti prema uputstvima koja se nalaze u dokumentu za laboratorijsku vežbu 2 - MPI. Obratiti pažnju na razlike koje postoje kod podešavanja za prevođenje na 32-bitnim i 64-bitnim računarskim sistemima. Alternativno, koristiti OpenMPI na računaru **`rtidev5.etf.rs`**.

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 tako da može biti izvršen sa bilo kojim od broja procesa iz opsega navedenog iza postavke zadatka. **`N`** označava maksimalan mogući broj procesa u trenutno dostupnom MPI okruženju. Za programe koji će biti izvršavani na samo jednom računaru, pretpostaviti da važi **`N=4`**. Svaki program treba da vrši proveru da li je broj procesa tekućeg izvršavanja odgovarajući postavci zadatka. U slučaju da to nije zadovoljeno, prekinuti izvršavanje korišćenjem MPI poziva **`Abort`**.

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 MPI implementaciju nad zadatim test primerom.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Ispište vreme izvršavanja sekvencijalne i paralelne implementacije problema.
- Uporedi rezultat MPI i sekvencijalne implementacije problema.
- Ispiše "**`Test PASSED`**" ili "**`Test FAILED`**" u zavisnosti da li se rezultat izvršavanja MPI implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Poređenje rezultata MPI i sekvencijalne implementacije problema izvršiti unutar procesa sa rangom 0. Kod zadataka koji koriste realne tipove (**`float`**, **`double`**) tolerisati maksimalno odsupanje od **`±ACCURACY`** prilikom poređenja rezultata CPU i MPI 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_DZ2_MPI.zip`** ili **`MPS_DZ2_MPI.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_DZ2_MPI.tar.bz2`**

Raspakivanje: **`tar xjvf MPS_DZ2_MPI.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 nalazi u datoteci **`simplex.c`**. U izvornom kodu data je matrica eksponenata jednačine i ivica *simplex*-a. Ulazni parametar programa je broj iteracija aproksimacije. Proces sa rangom 0 treba da učita ulazne podatke, raspodeli posao ostalim procesima, na kraju prikupi dobijene rezultate i ravnopravno učestvuje u obradi. Za razmenu podataka, koristiti rutine za kolektivnu komunikaciju. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]
2. 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. Proces sa rangom 0 treba da učita ulazne podatke, raspodeli posao ostalim procesima, na kraju prikupi dobijene rezultate i ravnopravno učestvuje u obradi. Svakom procesu dodeliti određen podskup matrice na obradu. Za slanje jednog dela matrice koristiti odgovarajući izvedeni tip. **Obratiti pažnju na nepotrebnu obradu mrtvih ćelija, i konvergiranih grupacija, kao i na međuprocesnu komunikaciju.** 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. Program testirati sa parametrima koji su dati u datoteci **`run`**. [1, N]

3. 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. 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.
4. Prethodni program paralelizovati korišćenjem *manager - worker* modela. Proces gospodar (master) treba da učita neophodne podatke, generiše poslove, deli posao ostalim procesima i ispiše na kraju dobijeni rezultat. U svakom koraku obrade, proces gospodar šalje procesu radniku na obradu jednu jedinicu posla čiji veličinu treba pažljivo odabrati. Proces radnik prima podatke, vrši obradu, vraća rezultat, signalizira gospodaru kada je spreman da primi sledeći posao i ponavlja opisani postupak dok ne dobije signal da prekine sa radom. Veličinu jedne jedinice posla prilagoditi karakteristikama programa. Ukoliko je moguće, koristiti rutine za neblokirajuću komunikaciju za razmenu poruka. Analizirati dati kod i obratiti pažnju na način izračunavanja temperatura. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim temperaturama u poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **`run`**. [1, N]