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`**.

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. Dostupne sekvencijalne implementacije se nalaze u arhivi **`MPS_DZ2_MPI.zip`** ili **`MPS_DZ2_MPI.tar`** koje se mogu preuzeti na adresi http://mups.etf.rs/dz/2013-2014/. Dozvoljeno je ograničeno preuređivanje dostupnih sekvencijalnih implementacija prilikom paralelizacije.

Svaki program treba da:

- Generiše ulazne test primere.
- Izvrši MPI implementaciju nad zadatim test primerom.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- 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.**

1. Paralelizovati program koji određuje vrednost broja π. Proces sa rangom 0 treba da obavesti ostale procese o broju tačaka koje treba da obrade, prikupi podatke od ostalih procesa i ne treba da učestvuje u računanju. Vrednost broja π se može statistički odrediti na više načina uz pomoć generatora pseudoslučajnih brojeva uniformne raspodele. Jedan od načina je generisanje tačaka u ravni sa obema koordinatama u realnom opsegu [0,1]. Vrednost broja π tada može biti određena na osnovu odnosa broja tačaka koje se nalaze u delu kruga poluprečnika 1 sa centrom u koordinatnom početku i broja tačaka koje pripadaju kvadratu stranice 1 koji obuhvata sve generisane tačke. Sekvencijalni program se nalazi u datoteci **`piCalculation.c`** u arhivi koja je priložena uz ovaj dokument. Za

komunikaciju koristiti samo MPI pozive **`Send`** i **`Receive`**. Obezbediti da proces gospodar prihvati jedan rezultat izračunavanja čim neki od procesa-radnika izvrši slanje. [1,N]

2. Paralelizovati program koji određuje vrednost broja π korišćenjem principa *N-version* programiranja (http://en.wikipedia.org/wiki/N-version_programming). Računanje vrednosti broja π treba izvesti na način izložen u prethodnom zadatku, ali u grupama od po približno **`K`** procesa, gde je **`K`** manje od ukupnog broja procesa. Procesi treba da budu što ravnomernije raspoređeni po novim grupama i komunikatorima. Svaka grupa treba da izračuna vrednost broja π i pošalje je procesu sa rangom 0 u MPI svetu. Proces sa rangom 0 u MPI svetu treba da ispiše medijanu primljenih vrednosti (http://en.wikipedia.org/wiki/Median). [2,N]
3. Sastaviti program koji vrši cikličnu raspodelu elemenata niza po procesima. Proces gospodar treba da generiše niz celih brojeva, a zatim izvrši slanje elemenata ostalim procesima. Svi procesi treba da izračunaju sumu dobijenih elemenata i pošalju je natrag procesu gospodaru. Za slanje elemenata koristiti odgovarajući izvedeni tip podataka. Ako broj elemenata niza nije deljiv brojem procesa u MPI svetu, prekinuti program. [1,N]
4. Paralelizovati program koji izračunava površinu koju obuhvata Mandelbrotov skup (fraktal) korišćenjem MPI tehnologije. Detaljan opis problema izračunavanja površine Mandelbrotovog skupa je dat u Laboratorijskoj vežbi 2 (http://mups.etf.rs/lab/MPS%20-%20Lab2%20-%20MPI.pdf), a sekvencijalni program se nalazi u datoteci **`area.c`** u arhivi koja je priložena uz ovaj dokument. Zadatak paralelizovati korišćenjem *manager - worker* modela. Proces gospodar (master) treba da učita broj tačaka za obradu po *x* i *y* osi, maksimalni broj iteracija za ispitivanje uslova divergencije, deli posao ostalim procesima i ispiše na kraju dobijeni rezultat. U svakom koraku obrade, proces gospodar šalje procesu radniku jednu, neobrađenu vrednost iz skupa koordinata po *x* osi. Proces radnik prima dobijenu koordinatu po *x* osi, računa broj tačaka izvan Mandelbrotovog skupa za dobijenu koordinatu po *x* osi i sve moguće koordinate po *y* osi, šalje procesu gospodaru rezultate i ponavlja opisani postupak dok ne dobije signal da prekine sa radom. [2,N]
5. Paralelizovati program koji vrši sortiranje brojanjem (http://en.wikipedia.org/wiki/Counting_sort). Sekvencijalni program se nalazi u datoteci **`countingSort.c`** u arhivi koja je priložena uz ovaj dokument. Proces sa rangom 0 treba da generiše i popuni pseudoslučajnim vrednostima niz za testiranje, raspodeli elemente niza procesima i ravnopravno učestvuje u poslu sa ostalim procesima. Nakon izvršene operacije sortiranja brojanjem, sortirani niz treba da se nalazi u procesu sa rangom 0 koji treba da ga uporedi sa nizom sortiranim sekvencijalnom **`qsort`** funkcijom iz **`stdlib`** biblioteke. Ako broj elemenata niza nije deljiv brojem procesa u MPI svetu, prekinuti program. [1,N]

Važno: 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.

Napomena: Vrednost broja π se može statistički odrediti na više načina uz pomoć generatora pseudoslučajnih brojeva uniformne raspodele. Jedan od načina je generisanje tačaka u ravni sa obema koordinatama u realnom opsegu [0,1]. Vrednost broja π tada može biti određena na osnovu odnosa broja tačaka koje se nalaze u delu kruga poluprečnika 1 sa centrom u koordinatnom početku i broja tačaka koje pripadaju kvadratu stranice 1 koji obuhvata sve generisane tačke.