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.

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/2014-2015/. Na **`rtidev5.etf.rs`** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: **`wget http://mups.etf.rs/dz/2014-2015/MPS_DZ2_MPI.tar.bz2`**

Raspakivanje: **`tar xjvf MPS_DZ2_MPI.tar.bz2`**

1. Paralelizovati program koji izračunava skalarni proizvod dva vektora. 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 se nalazi u datoteci **`dotProduct.c`** u arhivi koja je priložena uz ovaj dokument. [1, N]
2. Paralelizovati prethodni program korišćenjem principa *N-version* programiranja (http://en.wikipedia.org/wiki/N-version_programming). Procese treba podeliti u dve približno jednake grupe i dva komunikatora. Proces sa rangom 0 ne treba da učestvuje ni u jednoj novostvorenoj grupi i ne učestvuje u izračunavanju, već samo učitava podatke i raspodeljuje ih procesima sa rangom 0 u novoformiranim grupama. U jednoj grupi, procesi treba da izračunaju skalarni proizvod uz korišćenje rutina za kolektivnu komunikaciju, kao u prethodnom zadatku. U drugoj grupi, procesi treba da izračunaju skalarni proizvod isključivo uz korišćenje rutina za *point-to-point* komunikaciju. Proces sa rangom 0 u MPI svetu treba da sakupi i uporedi dobijene vrednosti. Program se nalazi u datoteci **`dotProduct.c`** u arhivi koja je priložena uz ovaj dokument. [3, N]
3. Paralelizovati program koji rešava *simple heat equation* problem koji opisuje promenu temperature na nekom prostoru kroz vreme, ako su poznati početna temperatura i granični uslovi. Program se u praksi koristi za simulaciju temperature procesora, a 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. 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 broj kolona matrice na obradu. Za slanje jedne kolone matrice koristiti odgovarajući izvedenti tip. Program se nalazi u datoteci **`hotSpot.c`** u arhivi koja je priložena uz ovaj dokument. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**. [1, N]
4. Prethodni program rešiti korišćenjem rutina za neblokirajuću komunikaciju za razmenu graničnih elemenata. Dok razmenjuju granične elemente, procesi treba da računaju vrednost unutrašnjih elemenata u tekućoj iteraciji. Program se nalazi u datoteci **`hotSpot.c`** u arhivi koja je priložena uz ovaj dokument. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**. [1, N]
5. Paralelizovati program koji izračunava površinu koju obuhvata Mandlebrotov skup (fraktal) korišćenjem MPI tehnologije. Detaljan opis problema izračunavanja površine Mandlebrotovog 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 Mandlebrotovog 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]