ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:03

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

1. Σκοπός της άσκησης 4
2. Παραδοτέα 4
3. Ρουτίνες ελέγχου αρχικοποίησης 4
4. Αποστολή και λήψη μηνυμάτων (send, receive) 4
5. Χρήση του rank στις Send, Recv ως παράμετρος 6

1. Σκοπός της άσκησης

- Έλεγχος συνάρτησης με MPI_Init.
- Αποστολή και λήψη απλών μηνυμάτων.
- Χρήση του rank ως παράμετρο στο Send, Recv.

2. Παραδοτέα

(A) 1 ερώτηση
(C) 7 ασκήσεις

3. Ρουτίνες ελέγχου αρχικοποίησης

1. Αντιγράψτε το αρχείο **c4.c** στο αρχείο **c1L3.c** χρησιμοποιώντας την εντολή **`cp .`**
2. **(C1)** Τροποποιήστε το πρόγραμμα **c1L3.c** ώστε να περιέχει έλεγχο αρχικοποίησης της MPI_Init, ώστε αν αποτύχει να καλεί την MPI_Abort και να τυπώνει ένα μήνυμα λάθους.
3. Κάντε compile και δημιουργήστε το αρχείο **c1L3**, το οποίο να το εκτελέσετε σε όλους τους κόμβους της συστοιχίας σας.
4. Αντιγράψτε το αρχείο **c1L3.c** στο αρχείο **c2L3.c** χρησιμοποιώντας την εντολή cp.
5. **(C2)** Τροποποιήστε το πρόγραμμα **c2L3.c** ώστε να σε περίπτωση που υπάρχει σφάλμα στην αρχικοποίηση του MPI_Init και πριν από την MPI_Abort να εκτυπώνει το όνομα του υπολογιστή στον οποίο δημιουργήθηκε το σφάλμα.
6. Κάντε compile και δημιουργήστε το αρχείο **c2L3** το οποίο να το εκτελέσετε σε όλους τους κόμβους της συστοιχίας σας.
7. Επιβεβαιώστε την ορθή εμφάνιση των μηνυμάτων από όλους τους κόμβους. **.**

4. Αποστολή και λήψη μηνυμάτων (send, receive)

1. Δημιουργήστε ένα αρχείο **send_receive.c**, επαναχρησιμοποιώντας τον κώδικα από το c2L3.c *(include, αρχικοποίηση, έλεγχος αρχικοποίησης, τερματισμός)*.
2. Τοποθετήστε μια δομή ελέγχου if η οποία θα κάνει το εξής:

1. Αν αυτή είναι η πρώτη διεργασία *(rank 0)* θα στέλνει το μήνυμα xx με τη συνάρτηση MPI_send. Αμέσως μετά θα καλείται η συνάρτηση MPI_receive για να λαμβάνει το μήνυμα που θα στέλνει κάποια άλλη διεργασία. Μέσα στο block εντολών τοποθετείστε μετά την τελευταία κλήση μια εκτύπωση *(printf)* που να εμφανίζει το μήνυμα που στάλθηκε και το μήνυμα που έγινε λήψη, όπως παρακάτω:
   `Send: xx Receive: xx`

2. Αν αυτή είναι η δεύτερη διεργασία *(rank 1)* θα λαμβάνει το μήνυμα που στάλθηκε και αμέσως μετά το ίδιο το μήνυμα θα το στέλνει με την MPI_Send. Μέσα στο block εντολών τοποθετείστε μετά την τελευταία κλήση μια εκτύπωση *(printf)* που να εμφανίζει το μήνυμα που στάλθηκε και το μήνυμα που έγινε λήψη, όπως παρακάτω:
   `Send: xx Receive: xx`

3. Αν αυτή είναι μια άλλη διεργασία, τότε θα εκτυπώνει το μήνυμα ότι δεν έχει να κάνει κάτι και επίσης θα εμφανίζει το όνομα του υπολογιστή που εκτελέστηκε, όπως παρακάτω
   `Nothing to do at node openmpi-2398.dyndns-ip.com`

4. **(C3 το αρχείο)** Μεταγλωττίστε το πρόγραμμα. Διορθώστε τα λάθη που ίσως εμφανιστούν.

5. **(A1)**
   - Εκτελέστε το πρόγραμμα με 1 διεργασία. Τι παρατηρείτε;
   - Εκτελέστε το πρόγραμμα με 2 διεργασίες. Τι παρατηρείτε;
   - Εκτελέστε το πρόγραμμα με 4 διεργασίες. Τι παρατηρείτε;

6. **(C4)** Τροποποιήστε το παραπάνω πρόγραμμα, ώστε η διεργασία με rank=1 να προσθέτει τον χαρακτήρα '+' στο τέλος του μηνύματος που έχει ληφθεί και να στέλνει το νέο μήνυμα που δημιουργείται. Προσέξτε το μέγεθος του buffer να είναι σωστό.

   ******* Μπορείτε να χρησιμοποιήσετε τη συνάρτηση stpcpy προκειμένου να αντιγράψετε αυτό που δέχεστε ως μήνυμα σε ένα buffer, να προσθέσετε τον χαρακτήρα **"+"**, στη συνέχεια να προσθέσετε τον χαρακτήρα τερματισμού string *(δηλαδή το* **\0***)* και στη συνέχεια να στείλετε το καινούργιο buffer,

ΑΣΚΗΣΗ (C5)

Δημιουργήστε ένα πρόγραμμα **send_receive_ping_pong.c** στο οποίο δυο διεργασίες ανταλλάσσουν μηνύματα, αλλά κάθε φορά προστίθεται ένας χαρακτήρας στο τέλος. Η διεργασία με rank 0 προσθέτει τον χαρακτήρα ***** στο τέλος του μηνύματος και το στέλνει, και μόλις το λαμβάνει η διεργασία με rank 1 προσθέτει τον χαρακτήρα **+** στο τέλος του μηνύματος και το στέλνει. Κάθε φορά θα εκτυπώνεται ένα μήνυμα αποστολής/λήψης, όπως:

`Send: xx+ Receive: xx+*`

5. Χρήση του rank στις Send, Recv ως παράμετρος

8. Αντιγράψτε το αρχείο **send_receive.c** στο αρχείο **send_receive2.c**

9. Συμπληρώστε το πρόγραμμα ώστε να γίνεται καταμέτρηση των δεδομένων που ελήφθησαν από την MPI_Recv της κάθε διεργασίας και να τυπώνεται αμέσως μετά την `printf()`

10. **(C6)** Σε κάθε `printf()` να εκτυπώνεται ως πρώτο στοιχείο μέσα σε παρένθεση το rank της διεργασίας που εκτυπώνει το συγκεκριμένο μήνυμα και το όνομα του κόμβου που εκτελείται ως εξής:

    ```
    (rank0) (openmpi-2398.dyndns-ip.com) Send.....
    ```

11. Μεταγλωττίστε και εκτελέστε το αρχείο σε 1,2,3 κόμβους. Επιβεβαιώστε την ορθή λειτουργία του.

12. Αντιγράψτε το αρχείο send_receive2.c στο αρχείο send_receive3.c

13. **(C7)** Τροποποιήστε τις συναρτήσεις `MPI_Send()`, `MPI_Recv()` ώστε στην παράμετρο που εμφανίζεται σε ποια διεργασία θα γίνει η αποστολή ή ή λήψη να μην υπάρχουν αριθμοί 0 ή 1 ή 2 αλλά να υπάρχει μια αριθμητική σχέση ως προς τη μεταβλητή rank. Δηλαδή, αντί να γράψετε να σταλεί κάτι στην επόμενη διεργασία (π.χ. Τη διεργασία με rank 1), να τοποθετείτε κατευθείαν μέσα στην κλήση της συνάρτησης το 'rank + 1' . Ομοίως για κάθε κλήση `MPI_Send()`, `MPI_Recv()`.

14. Μεταγλωττίστε και εκτελέστε το αρχείο σε 1, 2, 3 κόμβους. Επιβεβαιώστε την ορθή λειτουργία του.