ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:06

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.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

1. Σκοπός της άσκησης

- Υπολογισμός Αθροίσματος στοιχείων πίνακα.
- Υπολογισμός του π.
- Συναρτήσεις MPI_Scatter() και MPI_Reduce().
- Πολλαπλασιασμός πίνακα επί διάνυσμα και χρήση της παραμέτρου tag.

2. Παραδοτέα

(A) 1 ερώτηση
(C) 5 ασκήσεις

(συνέχεια από το προηγούμενο εργαστήριο…)

1. **(C1)** Αντιγράψτε το τελευταίο πρόγραμμα του προηγουμένου εργαστηρίου για τον παράλληλο υπολογισμό αθροίσματος στοιχείων (sum_parallel.c) στο **c1.c**. Τροποποιήστε το παραπάνω πρόγραμμα ώστε να χρησιμοποιεί τη συνάρτηση MPI_Scatter για την αποστολή των δεδομένων στις υπόλοιπες διαδικασίες. Επίσης να χρησιμοποιείτε τη συνάρτηση MPI_Reduce από τη διεργασία 0, για τη συλλογή του επιμέρους αθροίσματος και την εύρεση του συνολικού αθροίσματος.

2. **(A1)** Εκτελέστε τα 2 προγράμματα που έχετε φτιάξει για τιμές από N=1 έως N=11 και διαπιστώστε αν υπολογίζουν τα ίδια δεδομένα. Αν είναι διαφορετικά, δικαιολογήστε την απάντησή σας.

3. Υπολογισμός π με τη συνάρτηση MPI_Scatter

- Να αντιγράψετε το προηγούμενο αρχείο στο αρχείο **c2.c**.
- **(C2)** Να τροποποιήσετε το πρόγραμμα κατάλληλα, ώστε να χρησιμοποιεί τη συνάρτηση MPI_Scatter και τη συνάρτηση MPI_Reduce.
  - Η συνάρτηση MPI_Scatter στέλνει τμήματα από πίνακες σε διεργασίες.
  - Εμείς δεν έχουμε πίνακα αλλά εύρος τιμών.
  - Μπορούμε όμως, να κατασκευάσουμε έναν πίνακα N στοιχείων *(όσα βήματα θέλουμε να έχουμε)* ο οποίος θα έχει την παρακάτω μορφή:
    - στη θέση 0 θα έχει την τιμή 0
    - στη θέση 1 θα έχει την τιμή 1
    - ...
    - στη θέση N θα έχει την τιμή N

- Στη συνέχεια χρησιμοποιούμε το MPI_Scatter και στέλνουμε αυτόν τον πίνακα στις διεργασίες. Η πρώτη διεργασία θα πάρει αυτόματα τα πρώτα 0 N/size στοιχεία (0,1,....,(N/size-1)), η δεύτερη διεργασία θα πάρει αυτόματα τα επόμενα N/size (N/size,...,2N/size-1) κ.ο.κ.
- Θα γίνει ο υπολογισμός του p, σύμφωνα με το προηγούμενο εργαστήριο.
- Θα χρησιμοποιηθεί η MPI_Reduce από τη rank=0 για να παραλάβει και να αθροίσει τα μερικά αθροίσματα.
- Θα εμφανιστεί το τελικό αποτέλεσμα του π.

4. Πολλαπλασιασμός πίνακα επί διάνυσμα

Στη συνέχεια θα κατασκευάσουμε ένα πρόγραμμα για να υλοποιηθεί ο πολλαπλασιασμός πίνακα επί διάνυσμα. Όπως γνωρίζουμε ο πολλαπλασιασμός ενός πίνακα A[M][J] με ένα διάνυσμα B[J] έχει ως συνέπεια τη δημιουργία ενός διανύσματος C[M] στοιχείων, όπου κάθε στοιχείο c(i) του διανύσματος C[M] υπολογίζεται από την έκφραση:

$$c(i) = \sum_{j=1}^{n} A(i, j) * b(j)$$

$$i = 1,2,..., m$$

Ο σειριακός υπολογισμός του αθροίσματος μπορεί να γίνει με το παρακάτω τμήμα κώδικα:

```
for (i=1 to m) {
     c[i] = 0;
     for (j=1 to n) {
          c[i] = c[i] + A[i][j]*b[j];
     }
```

Ο παραλληλισμός μπορεί να γίνει είτε με ταυτόχρονο παραλληλισμό και των δυο βρόχων i,j μαζί είτε με παραλληλισμό του βρόχου i με διάσπαση.

Για λόγους απλότητας θα επιλέξουμε να διασπάσουμε το βρόχο i σε τμήματα. Αν έχουμε συνολικά **size** διεργασίες τότε θα διασπαστεί το τμήμα i σε m/size. Με παρόμοιο τρόπο με το προηγούμενο εργαστήριο η κάθε διεργασία **rank** θα πρέπει να επεξεργάζεται από `(rank-1)m/size` έως `(rank)m/size` .

Όλα τα αποτελέσματα *(κάθε* `c[i]`*)* θα στέλνονται σε μια διεργασία *(έστω rank 0)*. Όμως υπάρχει το πρόβλημα ότι κάθε διεργασία θα στείλει πολλά σε πλήθος `c[i]`.

Θα πρέπει λοιπόν να υπάρχει ένας τρόπος να ενημερώνουμε τη διεργασία που τα συλλέγει ποιο `c[i]` κάθε φορά στέλνεται.

Προκειμένου εκτός από την τιμή `c[i]` να στέλνουμε και σε ποια γραμμή i αντιστοιχεί θα πρέπει να χρησιμοποιήσουμε σε κάθε εντολή MPI_Send/Recv το πεδίο της ετικέτας (**tag**). Η ετικέτα είναι ένα πεδίο που ως τώρα το είχαμε μόνιμα σε τιμή 0. Σε αυτό το εργαστήριο η ετικέτα κάθε φορά θα έχει την τιμή i για να υποδηλώνει ότι η τιμή που μεταφέρεται αντιστοιχεί στο στοιχεί `c[i]`.

Μια τελευταία λεπτομέρεια είναι η εξής. Πως θα γνωρίζει κάθε διεργασία, εκτός από τη διεργασία με rank=0 τον πίνακα και το διάνυσμα. Ασφαλώς και δε γίνεται να τα γνωρίζει από την αρχή, οπότε θα πρέπει να τα στείλουμε από την διεργασία 0. Η διεργασία 0 θα αρχικοποιήσει τον πίνακα και το διάνυσμα και στη συνέχεια θα μοιράσει τα κατάλληλα στοιχεία σε κάθε διεργασία. Δε θα μοιράσει όλους τους πίνακες αλλά θα στείλει σε κάθε διεργασία μόνο τα στοιχεία που είναι απαραίτητα σε κάθε διεργασία.

Συγκεκριμένα στη διεργασία **rank** θα πρέπει να στείλει τις γραμμές από:

`(rank-1)m/size` έως **`(rank)m/size`** και όλο το **διάνυσμα**.

Λίστα Εργασιών Εργαστηρίου:

(C3) Να κατασκευάσετε το πρόγραμμα **c3.c** το οποίο υπολογίζει το γινόμενο ενός πίνακα με ένα διάνυσμα ΣΕΙΡΙΑΚΑ *(χωρίς openmpi)*. Ο πίνακας θα είναι 10*N στοιχείων διαστάσεις [10][N] και το διάνυσμα N θέσεων. Τοποθετήστε στην πρώτη γραμμή την εντολή **`#define N 5`**. Εσείς στο πρόγραμμά σας θα χρησιμοποιείτε το N και όχι το 5, ώστε αν θελήσουμε να εκτελέσουμε το πρόγραμμα για N=10 να μπορεί να γίνει εύκολα με μια τροποποίηση του define.

- Το πρόγραμμα θα ζητάει από το χρήση να πληκτρολογήσει N ακέραιες τιμές για το διάνυσμα.
- Το πρόγραμμα θα έχει μια συνάρτηση initialize_array_random() *(θα τη δημιουργήσετε εσείς)* η οποία θα χρησιμοποιεί τη συνάρτηση rand() για κάθε στοιχείο του πίνακα για να αρχικοποιηθεί από 0 έως 99 . Για να χρησιμοποιήσετε τη rand(), πρώτα καλείτε μια φορά για όλο το πρόγραμμα τη συνάρτηση αρχικοποίησης της rand() ως **`srand(time(NULL));`** και στη συνέχεια κάθε φορά που θέλετε ένα τυχαίο αριθμό από 0 έως 99 **`rand()%100;`** .
- Θα υπάρχει η κλήση της συνάρτησης print_results() *(θα τη δημιουργήσετε εσείς)* η οποία θα εκτυπώνει τον πίνακα, το διάνυσμα και το τελικό αποτέλεσμα. Να προσέξετε τη μορφοποίηση και τη στοίχιση των στηλών.

Επιβεβαιώστε την ορθή λειτουργία του προγράμματος.

(C4) Αντιγράψτε το προηγούμενο αρχείο, στο αρχείο **c4.c** στο οποίο θα αντικαταστήσετε τη συνάρτηση initialize_array_random() με τη συνάρτηση initialize_array_from_file() στην οποία θα τοποθετούνται αρχικές τιμές στον πίνακα, οι οποίες θα διαβάζονται από ένα αρχείο με όνομα **`array.dat`**

Επιβεβαιώστε την ορθή λειτουργία του προγράμματος.

(C5) Αντιγράψτε το προηγούμενο αρχείο, στο αρχείο **c5.c** το οποίο έχει μόνο τις συναρτήσεις αρχικοποίησης OpenMPI. Δε γίνεται διαχωρισμός των rank. Θα πρέπει αν εκτελεστεί σε μια διεργασία να μπορεί να λειτουργήσει σωστά.

Επιβεβαιώστε την ορθή λειτουργία του προγράμματος.

(συνεχίζεται σε επόμενο εργαστήριο…)