Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Συστήματα Παράλληλης Επεξεργασίας

3η Εργαστηριακή Άσκηση - Ενδιάμεση Αναφορά

9ο Εξάμηνο - Ακαδημαϊκό Έτος 2020-2021

Γιαννόπουλος Εμμανουήλ - 03117031

Παγώνης Γεώργιος - 03117030

11 Δεκεμβρίου 2021

1 Διάδοση θερμότητας σε δυο διαστάσεις - Μέθοδος Jacobi

1.1 Περιγραφή του προβλήματος

Στην ενδιάμεση αναφορά θα παραλληλοποίησουμε την εξίσωση θερμότητας. Το πρόβλημα στο οποίο θα εφαρμόσουμε την εξίσωση θερμότητας είναι η διάδοση θερμότητας σε επιφάνειας από το σύνορο στο εσωτερίκο. Ο τρόπος με τον οποίο θα αναπτυχθεί το παραλληλό πρόγραμμα είναι το μοντέλο ανταλλαγής μηνυμάτων με την βοήθεια της βιβλιοθήκης MPI. Η μέθοδος που θα χρησιμοποίησουμε για να προσομοίωσουμε την εξίσωση διάδοσης θερμότητας είναι η **Jacobi** .

Algorithm 1 Jacobi

for t, (0 to T) and (not $converged$) **do**
  for i, 1 to $X-1$ **do**
    for j, 1 to $Y-1$ **do**
      $U[t+1][i][j] = (1/4) * (U[t][i-1][j] + U[t][i][j-1] + U[t][i+1][j] + U[t][i][j+1]);$
    end for
  end for
  $converged = check\_convergence(U[t+1], U[t])$
end for

1.2 Παραλληλισμος σε αρχιτεκτονικές κατανεμημένης μνήμης

Παρατηρούμε ο,τι με την μέθοδο **Jacobi** ο παραλληλισμός είναι προφανής. Η τιμή κάθε σημείου του πίνακα κάθε χρονική στιγμή δεν εξαρτάται από καμία άλλη την ίδια χρονική στιγμή μόνο από τις γειτονικές της **North,South,West,East** την προηγούμενη χρονική στιγμή. Επομένως, μπορούμε να επεξεργαστούμε όλα τα στοιχεία του πίνακα παράλληλα κάθε χρονική στιγμή , αρκεί να έχουμε διασφαλίσει ότι έχουν υπολογιστεί όλα τα στοίχεια της προηγούμενης χρονικής στιγμής .

Για την παραλληλοποίησης του αλγορίθμου σε αρχιτεκτονικές κατανεμημένης μνήμης θα χρησιμοποίησουμε το μοντέλο ανταλλαγής μηνυμάτων με την βιβλιοθήκη **MPI**. Θα διαμοίρασουμε τον πίνακα σε ένα δισδιάστατο πλέγμα επεξεργαστών. Ο κάθε επεξεργαστής θα πάρει το κομμάτι του πίνακα που του αναλογεί και κάνει πραξείς πάνω στα δεδομένα του. Το μοντέλο ανταλλαγής μηνυμάτων θα χρησιμοποίηθει όταν για το υπολογισμό ένος από των στοιχείων του πίνακα που βρίσκεται σε έναν επεξεργαστή χρείαζεται τα δεδομένα από πίνακα ο οποίος βρίσκεται σε έναν άλλο επεξεργαστή. Αυτό συμβαίνει όταν χρείαζεται να υπολογίσουμε τα συνοριακά στοίχεια του υποπίνακα του κάθε υπολογιστή. Όταν ο κάθε επεξεργαστής χρείαζεται να υπολογήσει ένα τέτοιο στοιχείο ζητάει από τους γειτονικούς του να το δώσουν το κατάλληλο στοιχείο .

1.3 Ανάπτυξη του παράλληλου προγράμματος με MPI

1. Θα δημιουργήσουμε ένα δισδιάστατο πλέγμα επεξεργαστών το οποίο θα χρησιμοποίηθει για τον παραλληλίσμο.

2. Θα αρχικοποιήσουμε 2 πίνακες για την **current** και για την **previous** στιγμή σε κάθε επεξεργαστή. Για την υλοποίηση της άσκησης θα προσθέσουμε και στους 2 πίνακες που έχει κάθε επεξεργαστής 2 επιπλέον γραμμές και 2 επιπλέον στήλες για να τις χρησιμοποίησουμε για τα

στοιχεία των γειτονικών πινάκων που θα χρειστούμε για τον υπολογισμό. Έπειτα θα διαμοίρασουμε και τους δύο πίνακες στο πλέγμα των επεξεργαστων.

3. Θα δημιουργήσουμε ένα MPI_Datatype column για να μπορούμε να μεταφέρουμε μία στήλη ενός πίνακα όταν αύτο είναι απαραίτητο.

4. Σε κάθε επεξεργαστή θα βρούμε τους γειτονικούς επεξεργαστές και ανάλογα με την τοποθεσία του θα προσαρμώσουμε καποίους δείκτες **i_min,i_max,j_min,j_max** για την σωστή ghost shells και boundary κελίων (μιας και στο boundary δεν κάνουμε πράξεις).

5. Επείτα ξεκινάμε την μέθοδο Jacobi. Κάθε επεξεργαστής επικοινώνει με τους γειτονικούς του , τους στέλνει τις γραμμές που χρειάζοντε από αυτόν και λαμβάνει τις τιμές που χρείαζεται. Στην συνέχεια, κάνει τους υπολογισμούς στα δικά του δεδομένα. Επαναλαμβάνουμε το βήμα αύτο μέχρι να φτάσουμε στον αριθμό επαναλήψεων που θέλουμε ή να συγκλίνει οι πίνακες previous και current.

6. Για να μετρήσουμε το συνολικό χρόνο , τον χρόνο των υπολογισμό των τιμών , τον χρόνο του υπολογισμού της σύγκλισης κάνουμε **MPI_Reduce** στον επεξεργαστή με $rank = 0$ για να κρατήσουμε τον μέγιστο χρόνο .

7. Για να παρουσιασούμε τα τελικά αποτελέσματα κάνουμε **MPI_Gather** στον επεξεργαστή με $rank = 0$ για να μπορέσουμε να τα τυπώσουμε.

1.4 Ενδιάμεση Αναφορά

Στο πλαίσιο της ενδιάμεσης αναφοράς θα τρέξουμε το παράλληλο πρόγραμμα μόνο για χωρίο 2048×2048, με 32 διεργασίες-επεξεργαστές χωρίς έλεγχο σύγκλισης για 256 επαναλήψεις:

```
parlab21@scirouter:~/a3/mpi$ cat test.out
Jacobi X 2048 Y 2048 Px 8 Py 4 Iter 256 ComputationTime 0.188703 TotalTime 1.032483 midpoint 0.000000
```

Παρουσίαζουμε και με 64 διεγρασίες-επεξεργαστές για να δούμε την βελτίωση στο computation time , αλλά και την καθυστέρη στον συνολικό χρόνο λόγω της υπέρμετρης επικοινωνίας ανάμεσα στους επεξεργαστές:

```
parlab21@scirouter:~/a3/mpi$ cat test.out
Jacobi X 2048 Y 2048 Px 8 Py 8 Iter 256 ComputationTime 0.087717 TotalTime 2.345719 midpoint 0.000000
```

Θα το συγκρίνουμε με τον σειριακό κώδικα :

```
parlab21@scirouter:~/a3/serial$ cat test.out
Jacobi X 2048 Y 2048 Iter 255 Time 7.471204 midpoint 0.000000
```

Βλέπουμε ο,τι τα αποτελέσματα είναι τα αναμενόμενα με το σειρικό να χρειάζεται αρκέτα περισσότερο χρόνο για να εκτελεστεί. Έχουμε ένα σχετικά καλό περίπου $speedup = 7.25$.