ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΥΠΟΛΟΓΙΣΜΟ

ΑΣΚΗΣΗ-Ι

Ζωβοΐλης Δημήτριος-Μάριος

ΑΜ:19390064

Τμήμα: Ε1 (Δευτέρα 12-2μμ)
18/12/21

ΠΕΡΙΕΧΟΜΕΝΑ

Πίνακας περιεχομένων

1. Εισαγωγή – περιγραφή της εργασίας

Σκοπός της εργασίας είναι η κατασκευή ενός προγράμματος MPI σε γλώσσα C δέχεται ως είσοδο ένα διανύσματος X (μήκους n στοιχείων x_i | i=0…n-1), να υπολογίζει παράλληλα σε περιβάλλον 'p' επεξεργαστών και να τυπώνει στην οθόνη (ως έξοδο) τα ακόλουθα:

α) τη μέση τιμή των στοιχείων του διανύσματος X: $m=\frac{x_0+x_1+x_2+\ldots+x_{n-1}}{n}$

β) τη διασπορά των στοιχείων του διανύσματος X: $var=\frac{\sum_{i=0}^{n-1}(x_i-m)^2}{n}$

γ) ένα νέο διάνυσμα Δ όπου κάθε στοιχείο δi θα ισούται με την ποσοστιαία σχέση του αντίστοιχου στοιχείου (x_i) του διανύσματος X με τη διαφορά μεγίστου-ελαχίστου των τιμών όλου του διανύσματος X: $\delta_i=\frac{x_i-x_{min}}{x_{max}-x_{min}}\cdot 100$

2. Τεκμηρίωση της εργασίας

Η λύση της εργασίας βασίζεται στην ιδέα ότι το σύνολο του απαιτούμενου υπολογιστικού φόρτου θα ισοκατανεμηθεί στους 'p' επεξεργαστές του παράλληλου περιβάλλοντός. Έτσι, κάθε επεξεργαστής θα λαμβάνει ένα μέρος του αρχικού διανύσματος για να εκτελέσει ότι πράξεις χρειάζονται. Επίσης, να σημειωθεί πως για την η είσοδος των δεδομένων στο πρόγραμμα θαγίνεται με την χρήση μόνο ενός 'κεντρικού' επεξεργαστή. Στην προκειμένη περίπτωση, ως 'κεντρικός' επεξεργαστής επιλέγχηκε αυτός που έχει `rank==0`. Τέλος, τα αποτελέσματα του προγράμματος συγκεντρώνονται στον 'κεντρικό' αυτό επεξεργαστή και μέσω αυτού παρουσιάζονται στο χρήστη.

2.1 Σχεδιασμός και πληροφορίες για την υλοποίηση του κώδικα

Η δομή της main συνάρτησης του προγράμματος είχε την εξής μορφή:

1. Αρχικά, διαβάζεται το διάνυσμα X και το μήκος του από τον επεξεργαστή με `rank==0` με την χρήση της συνάρτησης `read_data()`.
2. Έπειτα στέλνουμε το μήκος του διανύσματος και το μέρος του διανύσματος που αντιστοιχεί σε κάθε επεξεργαστή.
3. Κάθε πεξεργαστής υπολοπγίζει την τοπική μέση τιμή, ελάχιστη και μέγιστη τιμή με την χρήση της συνάρτησης `calculate_info()`.
4. Κάθε επεξεργαστής στέλνει στον κεντρικό επεξεργαστή τις τιμές που υπολογίστικαν και εκεί υπολογίζεται η τελική μέση, ελάχιστη και μέγιστη τιμή για το αρχικό διάνυσμα X.
5. Από τον κενρτικό επεξεργαστή στέλνονται σε όλους τους επεξεργαστές η τελική μέση, ελάχιστη και μέγιστη τιμή για το αρχικό διάνυσμα X.

6. Κάθε επεξεργαστής υπολογίζει τη διασπορά των στοιχείων του διανύσματος που έλαβε από το αρχικό και ταυτόχρονα δημιουργεί μέρος του διανύσματος Δ.
7. Στέλνονται πίσω στον κεντρικό επεξεργαστή η τοπική διασπορά και μέρος του διανύσματος Δ στον κεντρικό επεξεργαστή, όπου και υπολιγίζεται η τελική διασπορά και ανακατασκευάζεται το διάνυσμα Δ.
8. Απελευθερώνεται ο χώρος που είχε δεσμευτεί δυναμικά μέσω της `malloc()`.
9. Τέλος, ο κεντρικός επεξεργαστής εμφανίζει στον χρήστη ένα μενού και στέλνει σε όλους τους επεξεργαστές την επιλογή του χρήστη ώστε στην περίπτωση που επιλέξει έξοδος να τερματιστεί ο βρόγχος σε όλους τους επεξεργαστές.

2.2 Προβλήματα που αντιμετωπίστηκαν

Ένα πρόβλημα στο οποίο έπεσα προσπαθώντας να λύσω την άσκηση ήταν ότι το μενού το εκτελούσαν όλοι οι επεξεργαστές με αποτέλεσμα να κολλάει το πρόγραμμα μου. Μετά όμως κατάλαβα πως για να δουλέψει το do while loop χρειαζόταν να εκτελείται το μενού από τον κεντρικό επεξεργαστή και η επιλογή του χρήστη να στέλνεται σε όλους τους επεξεργαστές και να τερματίζουν όλοι το do while loop. Να σημειωθεί επίσης πως απαντήθηκαν όλα τα ερωτήματα πλήρως.

3. Σχολιασμός – παράθεση εκτελέσων του προγράμματος

Όταν για είσοδος δόθηκε το διάνυσμα [8, 16, 5, 2, 3, 26, 13] σαν αποτελέσματα βγήκαν:

1. m = 10
2. var = 56.500000
3. d[0]=25.000000

   d[1]=58.333333

   d[2]=12.500000

   d[3]=0.000000

   d[4]=4.166667

   d[5]=100.000000

   d[6]=45.833333

   d[7]=20.833333

Που είναι και ταναμενόμενα αποτελέσματα.

Μετά για να δούμε τους χρόνους εκτέλεσης δόθηκαν διανύσματα διαφόρων μεγέθων και ελέγχηκε πόσο ήταν ο χρόνος εκτέλεσης του προγράμματος με διαφορετικούς αριθμούς επεξεργαστών.

- Διάνυσμα με 32 αριθμούς:
  - 1 επεξεργαστής: 0.000345 s
  - 2 επεξεργαστές: 0.000091 s
  - 4 επεξεργαστές: 0.004843 s
  - 8 επεξεργαστές: 0.072027
- Διάνυσμα με 1024 αριθμούς:
  - 1 επεξεργαστής: 0.000022 s
  - 2 επεξεργαστές: 0.044008 s
  - 4 επεξεργαστές: 0.025033 s
  - 8 επεξεργαστές: 0.116023 s
- Διάνυσμα με 1048576 αριθμούς:
  - 1 επεξεργαστής: 0.021976 s
  - 2 επεξεργαστές: 0.056615 s
  - 4 επεξεργαστές: 0.344049 s
  - 8 επεξεργαστές: 0.484108 s

Από τους χρόνους εκτέλεσης μπορούμε να δούμε ότι όσο περισσότερους επεξεργαστές χρησιμοποιούμε τόσο πιο πολύ αργεί το πρόγραμμά μας. Αυτό οφείλεται στο γεγον΄ζο ότι η χρήση πολλών επεξεργαστών δεν μικραίνει την πολυπλοκότητα αλλά μένει σταθερή. Έτσι, με την χρήση πολλών επεξαργαστών χάνεται χρόνος στα `MPI_Send` και `MPI_Recv`.