**ΜΥΕ023–Παράλληλα Συστήματα και Προγραμματισμός**

**Σετ Ασκήσεων #2**

Γεώργιος Παπαδάτος

ΑΜ: 3306

Email: [cs03306@uoi.gr](mailto:cs03306@uoi.gr)

Το 2ο σετ ασκήσεων αφορά τον παράλληλο προγραμματισμό μέσω του προτύπου MPI. Ζητείται η παραλληλοποίηση ενός προγράμματος πολλαπλασιασμού πινάκων με 2 τρόπους, καθώς και η παραλληλοποίηση ενός προγράμματος υπολογισμού του π με αυτοδρομολόγηση.

**Άσκηση 1**

Όλες η μετρήσεις έγιναν στα παρακάτω συστήματα:

|  |  |
| --- | --- |
| **Ονόματα υπολογιστών** | opti3060ws09  opti3060ws10  opti3060ws11  opti3060ws12  opti3060ws13  opti3060ws15  opti3060ws16 opti3060ws01 |
| **Επεξεργαστής** | Intel i3-8300 |
| **Πλήθος πυρήνων** | 4 |
| **Έκδοση OpenMPI** | 2.1.1 |
| **Έκδοση μεταφραστή** | gcc v7.3.0 |

**Το πρόβλημα**

Στην άσκηση αυτή ζητείται να παραλληλοποιηθεί ο πολλαπλασιασμός πινάκων με την μέθοδο διαχωρισμού γραμμών. Επίσης ζητείται η εκτέλεση του προγράμματος σε 3 διαφορετικά σετ υπολογιστών, το πρώτο πρέπει να περιέχει 2 υπολογιστές, το δεύτερο 4 και το τρίτο 8. Ο κάθε υπολογιστής πρέπει να χρησιμοποιεί όλους τους πυρήνες που διαθέτει.

**Μέθοδος παραλληλοποίησης**

Χρησιμοποιήθηκε το σειριακό πρόγραμμα από την ιστοσελίδα του μαθήματος. Αρχικά έγινε η αρχικοποίηση των απαραίτητων βοηθητικών μεταβλητών. Στην συνέχεια αρχικοποιήθηκε η λειτουργία του MPI με την εντολή

**MPI\_Init(&argc, &argv)**

Μετά την δημιουργία της «εικονικής μηχανής», παίρνουμε τα απαραίτητα στοιχεία για αυτή και το αναγνωριστικό της κάθε διεργασίας με τις εντολές

**MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myID)**

**MPI\_Comm\_size(MPI\_COMM\_WORLD, &nproc)**

Η κύρια διεργασία έχει myID ίσο με 0. Με βάση αυτό, ζητάμε μόνο από την κύρια διεργασία να διαβάσει και να φορτώσει τους πίνακες Α και Β. Από εδώ και πέρα ξεκινάει η διαδικασία υπολογισμού από το παράλληλο πρόγραμμα, οπότε κρατάμε αυτή την χρονική στιγμή με την εντολή

**startTime = MPI\_Wtime()**

Στην αρχή της «παράλληλης περιοχής» δημιουργούνται οι πίνακες myA και myC και υπολογίζεται ο αριθμός γραμμών για τις οποίες θα είναι υπεύθυνη η κάθε διεργασία. Οι παραπάνω πίνακες είναι τοπικοί πίνακες της κάθε διεργασίας και θα περιέχουν μόνο το κομμάτι του Α που χρειάζεται η διεργασία, καθώς και το κομμάτι του C το οποίο πρέπει να παράξει.

Σε αυτό το σημείο είναι όλα έτοιμα για την αποστολή των δεδομένων στις υπόλοιπες διεργασίες. Πριν την έναρξη των επικοινωνιών κρατάμε την χρονική στιγμή με την εντολή

**startCommTime = MPI\_Wtime()**

έτσι ώστε να μετρήσουμε τον χρόνο που αναλώθηκε στις επικοινωνίες. Η κλήση της παραπάνω εντολής γίνεται μόνο από την κύρια διεργασία (myID = 0), οπότε τοποθετείται μέσα σε έναν έλεγχο if. Η επικοινωνία επιτυγχάνεται αποτελεσματικά με τις εντολές

**MPI\_Scatter(A, work\*N, MPI\_INT, myA, work\*N, MPI\_INT, 0, MPI\_COMM\_WORLD)**

και

**MPI\_Bcast(B, N\*N, MPI\_INT, 0, MPI\_COMM\_WORLD)**

Αρχικά η scatter διαμοιράζει ένα διαφορετικό κομμάτι του πίνακα Α σε κάθε διεργασία, το οποίο αποθηκεύεται στην myA. Μετά η broadcast στέλνει ολόκληρο τον πίνακα Β σε κάθε διεργασία. Η διαφορετική συμπεριφορά προκύπτει γιατί η κάθε διεργασία δεν χρειάζεται όλες τις γραμμές του 1ου πίνακα για να κάνει τον πολλαπλασιασμό, αλλά μόνο αυτές που θα χρησιμοποιήσει. Αντίθετα, όλες οι διεργασίες χρειάζονται όλες τις στήλες του 2ου πίνακα για τον πολλαπλασιασμό, οπότε πρέπει να αποκτήσουν ολόκληρο τον B.

Εδώ ολοκληρώνονται προς το παρόν οι επικοινωνίες μεταξύ των διεργασιών, οπότε η κύρια διεργασία αποθηκεύει ξανά την τρέχουσα χρονική στιγμή και βρίσκει τον συνολικό χρόνο επικοινωνιών με τις εντολές

**endCommTime = MPI\_Wtime()**

**commTime += endCommTime - startCommTime**

Στη συνέχεια ακολουθεί ο πολλαπλασιασμός των πινάκων. Ο αλγόριθμος είναι παρόμοιος με του σειριακού προγράμματος, αλλά έχει μερικές αλλαγές. Αρχικά, έχει αλλαχθεί ο συνολικός αριθμός επαναλήψεων του εξωτερικού βρόγχου for

**for(i = 0; i < work; i++)**

καθώς πλέον ο 1ος πίνακας (myA) περιέχει λιγότερες γραμμές. Ο δεύτερος βρόγχος έχει μείνει ίδιος. Ο τρίτος βρόγχος έχει μείνει επίσης ίδιος, αλλά μέσα του δεν πολλαπλασιάζονται πλέον οι πίνακες Α και Β αλλά οι myA και Β. Τέλος, το άθροισμα που υπολογίζεται στον τρίτο βρόγχο αποθηκεύεται στον πίνακα myC

**myC[i][j] = sum**

Το επόμενο βήμα του προγράμματος είναι το μάζεμα των αποτελεσμάτων από όλες τις διεργασίες στην κύρια. Πριν την έναρξη των επικοινωνιών, η κύρια διεργασία κρατάει και πάλι την χρονική στιγμή με τον ακριβός ίδιο τρόπο όπως και πριν. Μετά, καλείται η εντολή

**MPI\_Gather(myC, work\*N, MPI\_INT, C, work\*N, MPI\_INT, 0, MPI\_COMM\_WORLD)**

η οποία μαζεύει τους πίνακες myC από όλες τις διεργασίες και τους ενώνει στον πίνακα C με την σωστή σειρά. Ολοκληρώνοντας το μάζεμα, η κύρια διεργασία υπολογίζει πάλι τον χρόνο που απαιτήθηκε για τις επικοινωνίες και τον προσθέτει στον προηγούμενο χρόνο. Επίσης, υπολογίζει και τον συνολικό χρόνο εκτέλεσης του προγράμματος μέχρι εδώ και τυπώνει τους δύο αυτούς χρόνους. Τέλος, γράφει τον τελικό πίνακα C σε ένα αρχείο και καλείται από όλες τις διεργασίες η

**MPI\_Finalize()**

για να κλείσει ομαλά το πρόγραμμα.

**Πειραματικά αποτελέσματα – μετρήσεις**

Τα προγράμματα εκτελέστηκαν στα συστήματα που αναφέραμε στην αρχή και η χρονομέτρηση έγινε με τη συνάρτηση MPI\_Wtime(). Συγκεκριμένα, αποθηκευόταν η αρχική χρονική στιγμή πριν την εκτέλεση της περιοχής ενδιαφέροντος και αφαιρούνταν από την τελική χρονική στιγμή μετά την εκτέλεση της περιοχής.

Στην εκτέλεση των παράλληλων κομματιών χρησιμοποιήθηκαν 4 πυρήνες σε κάθε υπολογιστή. Αυτό επιτεύχθηκε με την προσθήκη του αριθμού των slots για κάθε υπολογιστή στο αρχείο hostfile (π.χ. opti3060ws09 slots=4).

Κάθε πείραμα εκτελέστηκε 4 φορές και υπολογίστηκαν οι μέσοι όροι. Προσοχή δόθηκε οι χρόνοι να μην συμπεριλαμβάνουν την ανάγνωση και εγγραφή των αρχείων. Τα αποτελέσματα δίνονται στους παρακάτω πίνακες (οι χρόνοι είναι σε δευτερόλεπτα):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Πρόγραμμα | serial | 2 nodes | 4 nodes | 8 nodes |
| run 1 | 109.750823 | 25.194127 | 14.843630 | 24.411656 |
| run 2 | 110.368357 | 22.294424 | 12.456665 | 19.677093 |
| run 3 | 110.454618 | 22.731511 | 16.301924 | 12.769099 |
| run 4 | 110.505341 | 24.436533 | 14.550275 | 10.927721 |
| average | 110.269785 | 23.664149 | 14.538124 | 16.946392 |

Πίνακας 1. Χρόνοι εκτέλεσης

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Πρόγραμμα | serial | 2 nodes | 4 nodes | 8 nodes |
| run 1 | 0 | 1.343007 | 2.776257 | 18.656609 |
| run 2 | 0 | 0.932753 | 1.653708 | 14.369418 |
| run 3 | 0 | 1.080702 | 3.575596 | 7.150011 |
| run 4 | 0 | 1.227025 | 2.038049 | 5.614779 |
| average | 0 | 1.145872 | 2.510903 | 11.447704 |

Πίνακας 2. Χρόνοι επικοινωνιών

Με βάση τα παραπάνω προκύπτουν τα ακόλουθα γραφήματα.

Γράφημα 1. Μέσοι χρόνοι εκτέλεσης

Γράφημα 2. Μέσοι χρόνοι επικοινωνιών

**Σχόλια**

Παραλληλοποίηση 2 κόμβων

Με βάση τα αποτελέσματα παρατηρούμε ότι η παραλληλοποίηση 2 κόμβων βελτιώνει δραματικά την ταχύτητα εκτέλεσης του προγράμματος. Βέβαια δεν πλησιάζει την ιδανική μείωση χρόνου εκτέλεσης, καθώς πρόκειται για 8 διεργασίες. Επίσης παρατηρούμε πως ο χρόνος επικοινωνιών είναι μικρός αλλά όχι αμελητέος, καθώς είναι της τάξης του περίπου 5% του συνολικού χρόνου εκτέλεσης.

Παραλληλοποίηση 4 κόμβων

Στην περίπτωση της παραλληλοποίησης των 4 κόμβων τα αποτελέσματα είναι επίσης θετικά. Διπλασιάζοντας τις διεργασίες επιτυγχάνουμε σχεδόν τον υποδιπλασιασμό του χρόνου εκτέλεσης σε σχέση με την παραλληλοποίηση 2 κόμβων. Σε σχέση όμως με το σειριακό πρόγραμμα, παραμένουμε ακόμα στα ίδια περίπου επίπεδα βελτίωσης. Όσον αφορά το χρόνο επικοινωνιών, βλέπουμε μία αισθητή καθυστέρηση. Πλέον οι επικοινωνίες καταναλώνουν περίπου το 14% του χρόνου εκτέλεσης.

Παραλληλοποίηση 8 κόμβων

Τέλος, στην περίπτωση της παραλληλοποίησης 8 κόμβων τα αποτελέσματα δεν είναι θετικά. Σε σχέση με το σειριακό πρόγραμμα η βελτίωση παραμένει περίπου στα ίδια επίπεδα με την βελτίωση της παραλληλοποίησης 4 κόμβων, σε σχέση όμως με αυτή υπάρχει μια μείωση στην ταχύτητα εκτέλεσης. Παρατηρούμε ότι αυτή η μείωση οφείλεται στον τεράστιο χρόνο επικοινωνιών, ο οποίος καλύπτει το 65% του συνολικού χρόνου εκτέλεσης. Επίσης παρατηρούμε ότι ο χρόνος επικοινωνιών είναι ιδιαίτερα ευαίσθητος στον φόρτο του δικτύου εκείνη την χρονική στιγμή, καθώς υπάρχει πολύ μεγάλη απόκλιση από την μικρότερη και την μεγαλύτερη τιμή στις δοκιμές.