Αναφορά εργασίας 2 : Υλοποίηση αναδρομικής συνάρτησης διανομής δεδομένων με χρήση MPI.

ΤΗΜΜΥ ΑΠΘ Παράλληλα και Διανεμημένα Συστήματα

Νικοπολιτίδης Εμμανουήλ-Γεώργιος

ΑΕΜ:9474

Κώδικας : https://drive.google.com/drive/folders/15juSwxqAmsskFwO5VH3WkoJKPLyn7gRg

Εισαγωγή

Σκοπός είναι η δημιουργία συνάρτησης η οποία θα βρίσκει N δεδομένα d διαστάσεων αποθηκευμένα σε p processes και θα τα ανακατανέμει σύμφωνα με τις αποστάσεις τους από ένα τυχαίο σημείο (pivot) ώστε τελικά τα N/p σημεία με τις μικρότερες αποστάσεις να βρίσκονται στο πρώτο process , τα N/p σημεία με τις μεγαλύτερες αποστάσεις να βρίσκονται στο τελευταίο process και ανάλογη κατάσταση να επικρατεί στα ενδιάμεσα.

Βάση Δεδομένων

Το πρόγραμμα διαβάζει αρχεία δεδομένων με επέκταση .txt ή .data προερχόμενα από το αρχείο της UCI . Βασική προϋπόθεση για την σωστή διανομή των δεδομένων είναι η απουσία διπλών εγγραφών από την εκάστοτε βάση καθώς και το πλήθος των εγγραφών να είναι δύναμη του 2 όπως άλλωστε και ο αριθμός των processes.

Υλοποίηση

Ξεκινάμε με των διαμοιρασμό των δεδομένων στα processes (Στην πραγματικότητα τα δεδομένα θα ήταν υπερβολικά μεγάλα για τη μνήμη ενός υπολογιστή) . Έπειτα καλούμαι την συνάρτηση distributeByMedian όπου επιλέγουμε το πρώτο στοιχείο του πρώτου process ως pivot αλλά μόνο όταν στην συνάρτηση συμμετέχουν όλες οι διαδικασίες. Μετά υπολογίζουμε την ευκλείδια απόσταση μεταξύ pivot και κάθε εγγραφής και τις συλλέγουμε στην αρχηγό διαδικασία ώστε να βρούμε την διάμεση απόσταση που στην συνέχεια θα μεταδώσουμε σε όλους .Εκτελώντας preFixScan τοποθετούμε τα στοιχεία με απόσταση μεγαλύτερη της διάμεσης στην αρχή του πίνακα δεδομένων κάθε διαδικασίας. Έτσι κάθε διαδικασία ξέρει και ενημερώνει τον αρχηγό για το πόσα στοιχεία θα ανταλλάξει .Αυτός μέσω μιας άλλης υπορουτίνας δημιουργεί , συγκρίνοντας και μειώνοντας τον αριθμό των προς ανταλλαγή στοιχείων με τον αντίστοιχο των διαδικασιών από το αντίθετο μισό του συνολικού πίνακα δεδομένων ,τρισδιάστατο πίνακα με τις εξής πληροφορίες : Στοιχεία της μορφής finalArr[i][j][k] (k==0 or k==1) ερμηνεύονται ως η i διαδικασία θα στείλει (και αντίστοιχα θα λάβει) finalArr[i][j][1] στοιχεία στην finalArr[i][j][0] καθώς και έναν άλλον πίνακα που εκφράζει των αριθμό των ανταλλαγών που θα εκτελέσει κάθε process .

Κατόπιν μετατρέπουμε των 3D πίνακα σε μονοδιάστατο και γενικότερα προετοιμάζουμε το έδαφος ώστε να γίνουν με ασφάλεια οι ανταλλαγές .Τέλος τις εκτελούμε με non-blocking διαδικασίες και διαχωρίζουμε τις διαδικασίες σε μικρότερες ομάδες ώστε να περάσουμε στην επόμενη φάση της αναδρομής. Πριν ολοκληρωθεί το πρόγραμμα κάνουμε έναν αυτοέλεγχο εξετάζοντας αν η μεγαλύτερη απόσταση της κάθε διαδικασίας είναι μικρότερη από την μικρότερη της επόμενης.

Ο κώδικας θα παρουσιάσει προβλήματα στην περίπτωση που 2 ή παραπάνω εγγραφές έχουν ίδια απόσταση από το pivot και αυτή τυγχάνει να είναι και η διάμεση απόσταση σε κάποια από τις εκτελέσεις της υπορουτίνας . Η πρόβλεψη τέτοιων περιπτώσεων παραλείφθηκε ηθελημένα θεωρώντας αρκετά απίθανο το σενάριο (κάτι που στην πράξη αποδείχτηκε λάθος) Παρόλα αυτά ο σκοπός και τα αποτελέσματα της άσκησης δεν επηρεάζονται χρησιμοποιώντας συμβατά datasets

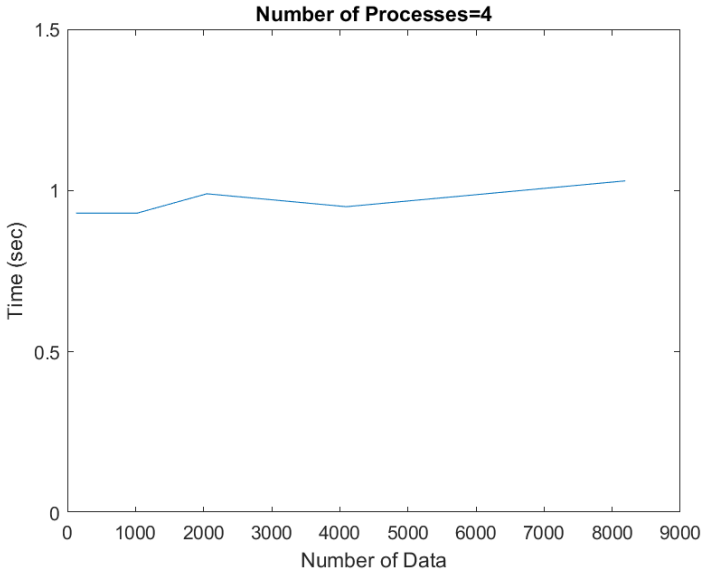
MPI

Χρησιμοποιήθηκαν εντολές MPI Send ,Receive ,Gather ,Broadcast ,Scatter καθώς και η δημιουργία communicators . Με ιδιαίτερη προσοχή σε

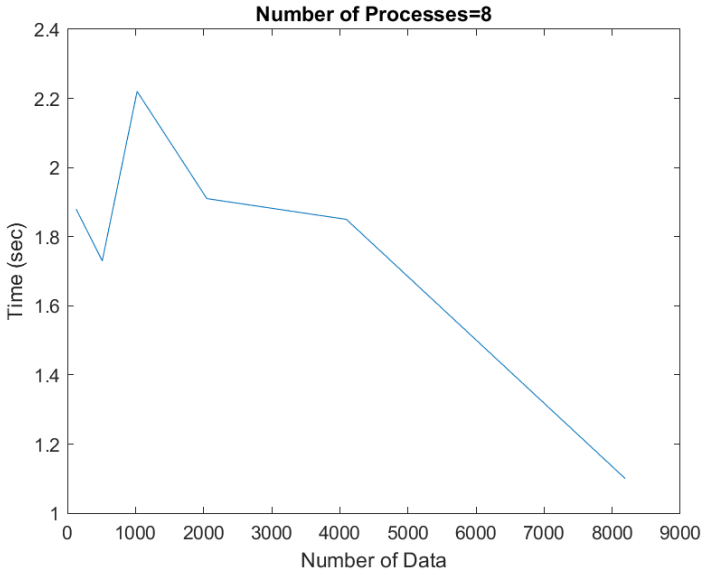
- Τα στοιχεία που θέλουμε να στείλουμε πρέπει να βρίσκονται σε συνεχόμενες θέσεις μνήμης
- Ο προορισμός να διαθέτει μνήμη άφιξης
- Το σωστό χρονισμό (μεγάλη βοήθεια προσφέρει η MPI_Barrier())

Αποτελέσματα

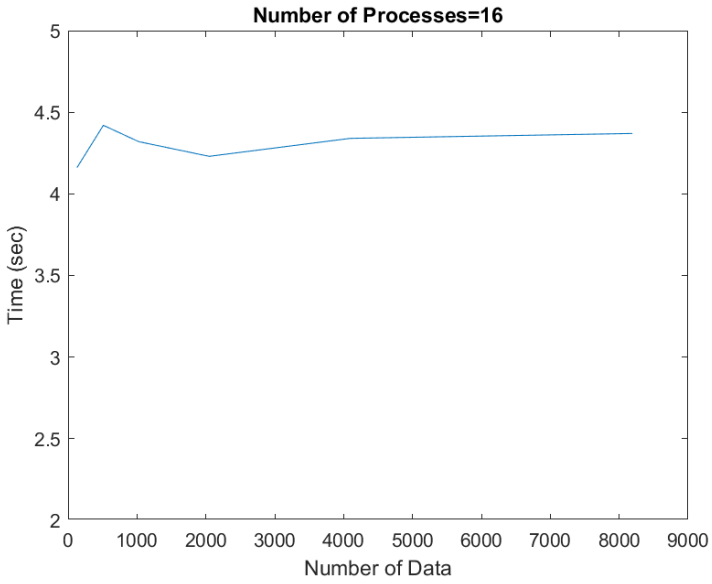
Η εφαρμογή δοκιμάστηκε σε ορισμένες κατάλληλες βάσεις δεδομένων (Dry_Bean_Dataset2.txt , wine.data) με τα αποτελέσματα που φαίνονται στα διαγράμματα. Καταλήγουμε έτσι στα εξής συμπεράσματα:



1.Για λίγες διαδικασίες ο χρόνος παραμένει πρακτικά σταθερός με τάση φυσιολογικής αύξησης που ακολουθεί την αύξηση των δεδομένων



2.Για μεσαίο αριθμό διαδικασιών ο χρόνος παρουσιάζει αρχικά μια αύξηση για λίγα δεδομένα αλλά στην συνέχεια εμφανίζει εντυπωσιακή πτώση .Αυτό συμβαίνει γιατί αρχικά έχουμε μια επιβράδυνση που οφείλεται στο γεγονός πως οι επικοινωνίες μεταξύ των processes είναι χρονοβόρες .Στη συνέχεια όμως το κόστος αυτό αντισταθμίζεται από τον διαμοιρασμό του φόρτου εργασίας.



3.Για τον μεγαλύτερο αριθμό διαδικασιών διαπιστώνουμε πως το κόστος των επικοινωνιών επικαλύπτει το όφελος τις παραλληλοποίησης, επιβαρύνει των κώδικα μας και τον κάνει αρκετά αργό (περίπου 4 φορές).

Έτσι είναι εύκολο να συμπεράνουμε πως καλούμαστε να διαχειριστούμε ένα trade-off μεταξύ κόστους επικοινωνίας και φόρτου εργασίας έχοντας πάντα ως περιορισμό τους πόρους μνήμης που διαθέτουμε.