Αναφορά εργασίας 1 : Υπολογισμός αριθμού τριγώνων γράφου σειριακά και παράλληλα.

ΤΗΜΜΥ ΑΠΘ Παράλληλα και Διανεμημένα Συστήματα

Νικοπολιτίδης Εμμανουήλ-Γεώργιος

ΑΕΜ:9474

Κώδικας :
https://drive.google.com/drive/folders/16yBpqqXSJp4NrXUx15EEIk1femTlqfeB?usp=sharing

Εισαγωγή

Σκοπός της εργασίας είναι ο υπολογισμός των τριγώνων που σχηματίζουν οι κόμβοι μεγάλων γράφων χρησιμοποιώντας διαφορετικές προγραμματιστικές τεχνικές εφαρμόζοντας τόσο σειριακό όσο και παράλληλο κώδικα και συγκρίνοντας τα αποτελέσματα από άποψη χρόνου και απόδοσης. Οι δομές δεδομένων αντλούνται από το Διαδίκτυο και πρόκειται στην πραγματικότητα για αραιούς συμμετρικούς πίνακες σε CSC Format στους οποίους είναι αποθηκευμένοι οι γράφη. Η υλοποίηση γίνεται στις γλώσσες προγραμματισμού MATLAB και C.

Μαθηματικό υπόβαθρο

Σε κάθε περίπτωση βάση του αλγορίθμου αποτελεί η μαθηματική σχέση C=A.*(A*A) όπου A ο αρχικός αραιός πίνακας που λαμβάνουμε από το διαδίκτυο και C πίνακας του οποίου τα στοιχεία ερμηνεύονται ως το πλήθος των πιθανών διαδρομών-τριγώνων μεταξύ του στοιχείου στήλη και του στοιχείου σειρά. Για παράδειγμα αν το στοιχείο Cij=3 σημαίνει πως υπάρχουν τρία διαφορετικά τρίγωνα που εμπλέκουν τους κόμβους i και j. Στην συνέχεια μετράμε τα τρίγωνα αυτά.

Υλοποίηση

Θα υλοποιήσουμε έναν αλγόριθμο υπολογισμού του ζητούμενου κάθε φορά με ελάχιστες αλλαγές με τους εξής 5 τρόπους:

1. MATLAB Sequential
2. C Sequential
3. C Parallel with OpenCilk
4. C Parallel with OpenMP
5. C Parallel with Pthreads

Και για τους εξής 3 γράφους:

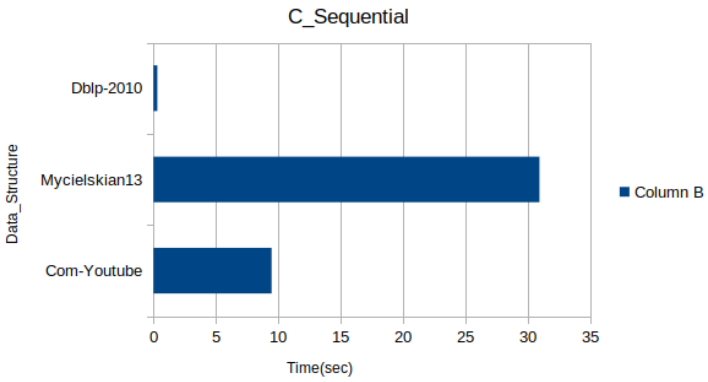
1. com-Youtube(3056386 τρίγωνα)
2. Mycielskian13(0 τρίγωνα)
3. dblp-2010(1676652 τρίγωνα)

MATLAB Sequential

Εδώ επιχειρούμε μια πρώτη απλή υλοποίηση σε μια υψηλού επιπέδου γλώσσα προγραμματισμού . Οι χρόνοι είναι αρκετά καλοί για δομές με μικρό μέγεθος

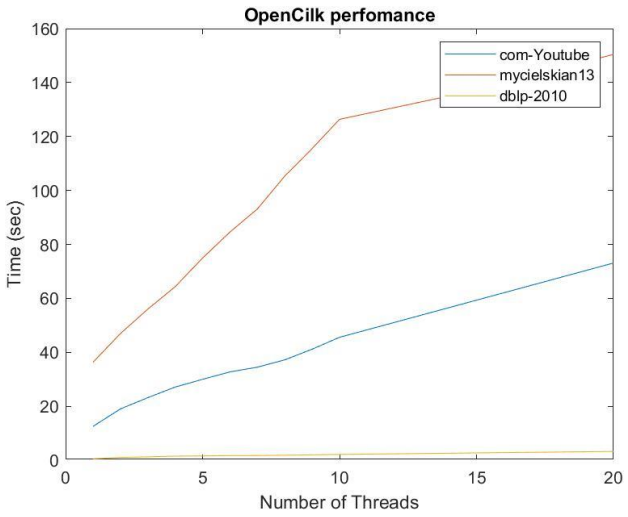
C Sequential

Μία συνηθισμένη σειριακή υλοποίηση του αλγορίθμου που περισσότερο χρησιμεύει ως αναφορά για την παραλληλοποίηση. Οι μέσοι χρόνοι (για τρεις εκτελέσεις σε κάθε γράφο είναι)



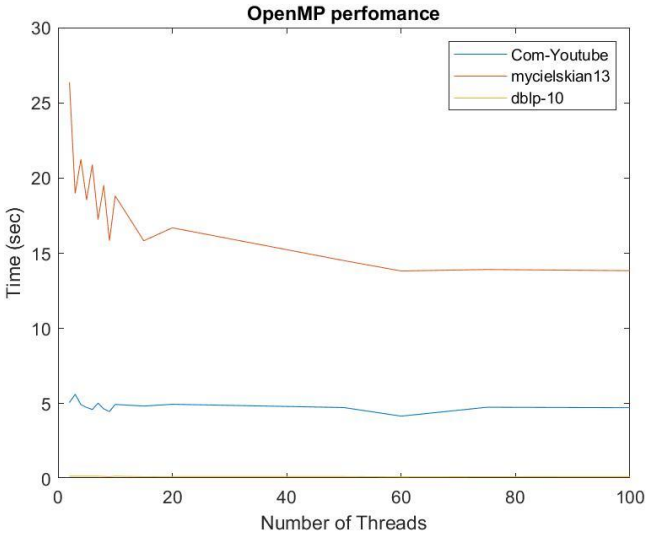
C Parallel with OpenCilk

Βλέπουμε πως η υλοποίηση με OpenCilk βγάζει χειρότερους χρόνους από την σειριακή και μάλιστα η κατάσταση επιδεινώνεται όσο ανεβαίνει ο αριθμός τον threads . Κάτι τέτοιο παρότι αναπάντεχο έχει εξήγηση αφού παίζει ρόλο ο αλγόριθμος αλλά και η φτωχή σχετικά αξιοπιστία του συγκεκριμένου add-on.



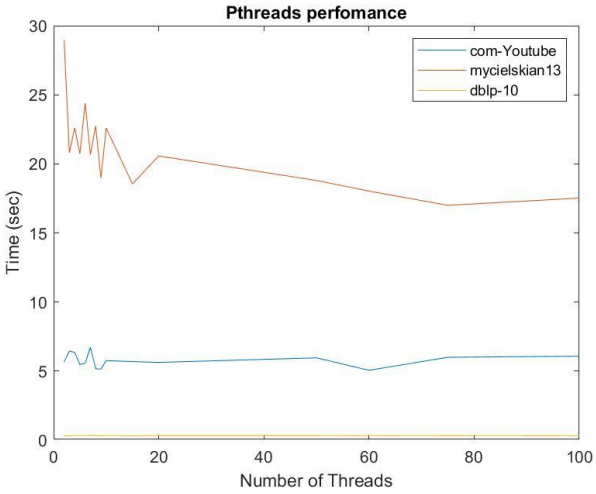
C Parallel with OpenMP

Χρησιμοποιώντας OpenMP παρατηρούμε πως όντως ο παράλληλος κώδικας επιταχύνει την έκδοση αποτελεσμάτων Ακόμα υπάρχει ένας αριθμός threads πέρα από τα οποία τα αποτελέσματα δεν εμφανίζουν σημαντικές αλλαγές και άρα είναι προτιμότερο να περιορίσουμε τα threads στον αριθμό αυτό αφού μας κοστίζουν υπολογιστικούς πόρους. Επίσης χρησιμοποιώντας μικρό αριθμό threads φαίνεται να υπάρχει μια αστάθεια στην απόδοση που είναι όμως σε κάθε περίπτωση καλύτερη του σειριακού.



C Parallel with Pthreads

Με τα pthreads βγάζουμε ανάλογα συμπεράσματα . Αξίζει να σημειωθεί πως τα pthreads φαίνεται να είναι λίγο πιο αργά από την OpenMP για τους γράφους που δοκιμάστηκαν ,αν και για μικρές δομές δεδομένων όπως η dbpl-2010 η διαφορά στους χρόνους είναι ασήμαντη.



Επιπλέον Συμπεράσματα

Με βάση τα παραπάνω είναι ξεκάθαρο πως η παραλληλοποίηση ενός κώδικα εξαρτάται τόσο από τον ίδιο τον κώδικα αλλά και από τον τρόπο και τον βαθμό παραλληλοποίησης που επιλέγουμε . Επίσης η απόδοση δεν εξαρτάται πάντα από το μέγεθος τον δεδομένων αλλά υπάρχουν περιπτώσεις όπου μικρότερα dataset αργούν σε σχέση με μεγαλύτερα. Τέλος η λογική του όσο περισσότερα threads τόσο το καλύτερο δεν ισχύει αφού κάποια στιγμή η δημιουργία threads γίνει πιο χρονοβόρα από την υλοποίηση του κώδικα με λιγότερα νήματα.