Άσκηση 1
Παράλληλα και Διανεμημένα Συστήματα

Θωμάς Πλιάκης tpliakis@ece.auth.com
ΑΕΜ: 9018

Link κώδικα :
https://github.com/thpliakis/thpliakis-PDS_Exercise_1.git

Σειριακός κώδικας:

Η σειριακή υλοποίηση για την εύρεση των τριγώνων που συμμετέχει κάθε κόμβος βασίζεται αρκετά στην δομή του CSC προτύπου. Ουσιαστικά ο αλγόριθμος κάνει 2 for loops για να βρει το διάνυσμα C3, ενώ ταυτόχρονα μπορεί να απαριθμεί τα τρίγωνα του γράφου μία φορά το καθένα λόγω του ότι είναι αποθηκευμένος μόνο ο μισός πίνακας γειτνίασης του κάθε γράφου. Ο αλγόριθμος ακολουθάει τα εξής βήματα:

1. Για κάθε κόμβο (1 for) βρες με ποιους άλλους κόμβους έχει ακμή (1 λίστα)

2. Για κάθε έναν από αυτούς (2 for) βρες τους κόμβους με τους οποίους αυτός συνδέεται (2 λίστα)

3. Σύγκρινε τις 2 λίστες και για κάθε κοινό στοιχείο που υπάρχει αύξησε των αριθμό των τριγώνων και κάθε τα αντίστοιχες θέσεις των 3 κόμβων στο σιάνυσμα C3.

Παραληλοποίηση:

1. Με **phtreads** έγινε η παραλληλοποίση του πρώτου **for** ώστε να μετράνε παράλληλα τρίγωνα τα διάφορα νήματα, δηλαδή κάθε νήμα μετράει σε διαφορετική ομάδα κόμβων. Αρκετές μεταβλητές γίνανε global ώστε να έχουν πρόσβαση όλα τα νήματα. Φυσικά για τις μεταβλητές που τα νήματα γράφουν χρησιμοποιήθηκαν κλειδιά με τις *mutex_lock, mutex_unlock.* Επίσης όρισα μια δομή *stuct* για να περνούνται τα κατάλληλα ορίσματα σε κάθε νέο νήμα ώστε να υπαρχει έλεγχος και καταγραφή τους.
2. Στην **Cilk** έγινε το εσωτερικό του πρώτου for νέα συνάρτηση ώστε να χρησιμοποιηεί την εντολή *cilk_for* στο εξωτερικό και να δημιουργείται κάθε φορά καινούργιο νήμα για μια ομάδα κόμβων. Προσθέθηκαν και οι *__cilkrts_end_cilk, __cilkrts_set_param* για να θέσουμε τον αριθμό νημάτων που θέλουμε κάθε φορά, όπως επίσης και τα κλειδιά *mutex_lock, mutex_unlock* για να διασφαλιστεί ότι το C3 υπλογίζεται σωστά.
3. Ομοίως με την **Cilk** έγινε η παραλληλοποίηση στην **OpenMP** όπου χρησιμοποίησαμε τις εντολές #pragma *omp parallel , #pragma omp* for για την παραλληλοποίηση του πρώτου **for** και #pragma_omp_critical στον υπολογισμό του C3 για να μην διαβάζουν και γράφουν τα νήματα παράλληλα και να γίνεται σωστός υπολογισμός του διανύσματος.

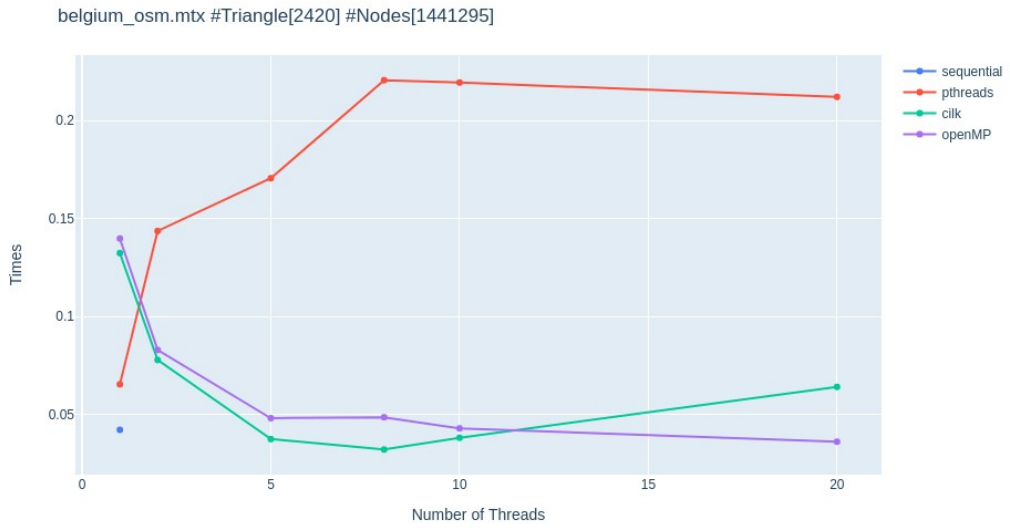
Έλεγχος αποτελεσμάτων: Οι υλοποιήσεις δίνουν σωστό αριθμό τριγώνων, αλλά τις δοκίμασα και σε έναν μικρό γράφο που δημιούρησα ώστα να ελέγξω αν υπολογίζεται σωστά το C3.

Διάβασμα γραφημάτων: Κάθε γράφημα δείχνει τους χρόνους, τα νηματά και την μέθοδο παραλληλοποίησης που χρησιμοποίθηκε σε κάθε γράφο. Ο κάθετος άξονας δείχνει τον χρόνο που χρειάστηκε το καμμάτι του κώδικα για την μέτρηση των τριγώνων. Ο οριζόντιος άξονας αναγράφει το πλήθος των νημάτων που έχουν δημιουργηθεί.

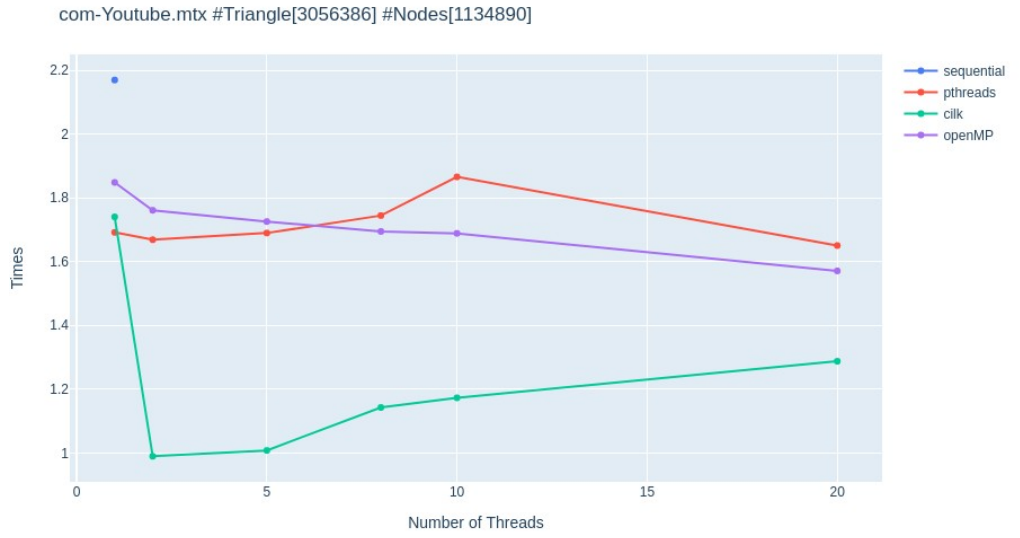
Παρατηρήσεις στα αποτελέσματα: Στην υλοποίηση με τα **Pthreads** φαίνεται από το *System Monitor* να μην αξιοποιούνται επαρκώς παράλληλα όλοι οι επεξεργαστές του υπολογιστή. Ειδικά σε μεγάλο αριθμό νημάτων εμφανίζονται και πιο αργοί χρόνοι καθώς πρέπει να περιμένουν τα νήματα το κλειδί για να γράψουν σε κοινές μεταβλητές οπότε έχουμε και έχουμε μεγάλο overhead. Φαίνεται να παίζει σε μεγάλο βαθμό και η δομή του γράφου στους χρόνους υπολογισμού αλλά δεν μου είναι ξεκάθαρο λόγω τον μικρών αποκλίσεων που έχουν οι ίδιοι οι χρόνοι. Γενικά είναι η πιο αργή από τις 4 υλοποιήσεις. Οι υλοποίησεις με **Cilk** και **openMP** έχουν παρόμοια απόδοση. Στους 2/5 από τους γράφους, είναι πιο αργές από την σειριακή, σε 1/5 την φτάνουν σε επίδοση χρησιμοποιόντας περισσότερα νήματα που εξαλείφουν τις καθιστερήσεις και στις άλλες 2 /5 είνα πολύ πιο γρήγορες από αυτήν. Επίσης δεν ξεκάθαρο κάποιο συγκεκριμένο συμπέρασμά αφού στην πρώτη περίπτωση και στην τρίτη έχουμε ταυτόχρονα γράφους με πολλούς κόμβους και πολλά τρίγωνα. Τέλος μεγάλο ρόλο έπαιξαν τα flags που δώθηκαν στον compiler.

Οι υλοποιήσεις έτρεξαν σε:
Dell latitude 7490
8 GB
Intel® Core™ i5-8350U CPU @ 1.70GHz × 8
Mesa Intel® UHD Graphics 620 (KBL GT2)

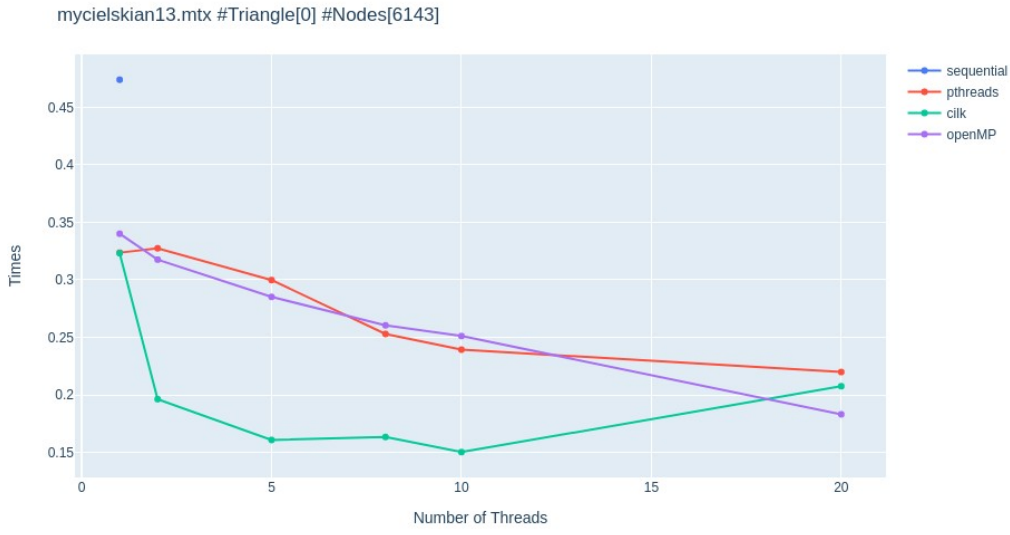
belgium_osm.mtx



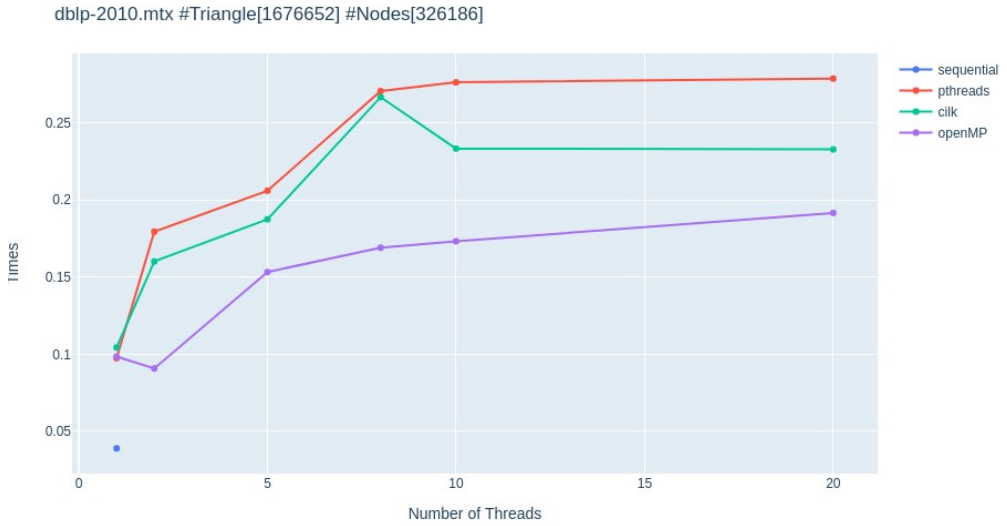
com-Youtube.mtx



mycielskian13.mtx



dblp-2010.mtx



NACA0015.mtx