PROJECT ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ

ΜΕΛΗ:
ΔΙΟΝΥΣΙΟΣ ΒΥΝΙΑΣ 1054347
ΔΙΟΝΥΣΙΟΣ ΓΙΑΝΝΟΥΛΗΣ 1054302
ΚΩΝΣΤΑΝΤΙΝΟΣ-ΔΗΜΗΤΡΙΟΣ ΔΑΛΕΖΙΟΣ 1054323
ΜΑΡΙΟΣ ΑΛΕΞΑΝΔΡΟΣ ΜΟΡΦΟΠΟΥΛΟΣ 1058102

ΑΣΚΗΣΗ 1)

- Για αυτό το ερώτημα πρέπει να παραλληλοποιήσουμε το εσωτερικό της for η οποία καλεί την συνάρτηση RDP για όλες τις πολυγραμμές.
  Άρα θα χρησιμοποιήσουμε την εντολή #pragma omp for schedule με παραμέτρους static ώστε η παραλληλοποίηση να γίνει στατικά, και chunk που έιναι ο αριθμός των πολυγραμμών που θα πάρει κάθε thread. Επίσης πρέπει να επιτρέψουμε στο κάθε thread να προχωρήσει από το parallel κομμάτι χώρις να περιμένει τα υπόλοιπα thread να τελειώσουν με το nowait. Τέλος πρέπει να δηλώσουμε την εισαγωγή του αποτελέσματος στο vector SimplifiedAllPolylines σαν κρίσιμο σημείο με την εντολή #pragma omp critical.
- Οι χρόνοι των 4 threads έιναι:
  thread 0: 0.028004
  thread 1: 0.075923
  thread 2: 0.238155
  thread 3: 7.282558
  Παρατηρούμε σημαντική διαφορά στους χρόνους τέλεσης των 4 threads και ειδικά του 4^{ου}. Αυτό συμβαίνει γιατί και τα 4 thread θα επεξεργαστούν τον ίδιο αριθμό πολυγραμμών αλλά κάθε πολυγραμμή θέλει διαφορετικό χρόνο για να επεξεργαστεί, άρα οι συνολικοί χρόνοι των threads είναι πολύ διαφορετικοί.

ΑΣΚΗΣΗ 2)

- Για αυτό το ερώτημα αλλάξαμε σε σχέση με το προηγούμενο την παράμετρο του schedule σε dynamic με τις υπολοιπες αλλαγές από τον αρχικό κώδικα να μένον όπως έχουν.
- Η παράμετρος που πρέπει να καθοριστεί πειραματικά είναι το μέγεθος του chunk, δηλαδή ο αριθμός των πολυγραμμών που θα επεξεργαστεί το κάθε thread πριν ζητήσει να πάρει το επόμενο chunk.
- Παρατηρούμε ότι για μεγάλα chunk οι χρόνοι των thread διαφέρουν πολύ, ενώ όσο μικραίνουν τόσο πιο κοντά βρίσκονται .Για παράδειγμα για
  chunk = AllPolylines.size()/num_of_threads*10.000
  thread 0: 4.700602
  thread 1: 4.550206
  thread 2: 4.044856
  thread 3: 4.090328
  ενώ για chunk = AllPolylines.size()/num_of_threads*2.5
  thread 0: 1.646661
  thread 1: 0.454713
  thread 2: 1.040808
  thread 3: 9.391076

ΑΣΚΗΣΗ 3)

- Για αυτό το ερώτημα θα βάλουμε την εντολή #pragma omp task στο μέρος της συνάρτησης RDP που καλεί τον εαυτό της. Συγκεκριμένα το task που θα δημιουργείται θα εκτελεί την πρώτη κλήση, ενώ η δεύτερη θα συνεχίζει από το αρχικό task. Κάτω από αυτό θα χρησιμοποιήσουμε την εντολή #pragma omp taskwait ώστε να έχει τελειώσει τη διεργασία το task που δημιουργήθηκε πριν το αρχικό task ενώσει τα αποτελέσματα των 2 κλήσεων. Τέλος στη main θα βάλουμε το κομμάτι που καλείται η συνάρτηση για κάθε πολυγραμμή σε ένα parallel block και θα χρησιμοποιήσουμε την εντολή #pragma omp single ώστε να γίνει η κλήση μόνο από ένα thread.

ΑΣΚΗΣΗ 4)

- Για αυτό το ερώτημα θα χρησιμοποιήσουμε τις ίδιες εντολές με το προηγούμενο και θα προσθέσουμε ένα ακόμα task που θα εκτελεί την δεύτερη αναδρομική κλήση της συνάρτησης. Και τα 2 task θα πρέπει να ολοκληρωθούν πριν περάσουν από την εντολή #pragma omp taskwait.
- Στους χρόνους αυτού του ερωτήματος παρατηρούμε ότι είναι λίγο μεγαλύτεροι από τους χρόνους του προηγούμενου ερωτήματος και αρκετά μεγαλύτεροι από τους χρόνους της δυναμικής παραλληλοποίησης της δεύτερης άσκησης. Τέλος παρατηρούμε ότι για 2 threads ο χρόνος εκτέλεσης των πράξεων είναι καλύτερος σε σχέση με το 1 και με τα 4 threads. Μια λύση που θα προτείναμε είναι μια υβριδική παραλληλοποίηση όπου θα κάναμε χρήση των task στην αναδρομική κλήση της συνάρτησης και θα αναθέταμε με μια εντολή #pragma omp for schedule με δυναμικό τρόπο τις πολυγραμμές από το vector AllPolylines σε threads. Έτσι όταν αδειάσει το taskpool τα ελεύθερα threads μπορούν να τραβήξουν μια καινούργια πολυγραμμή από το vector.

Χρόνος εκτέλεσης αρχικού προγράμματος (-O0): 12.673315

(-O3):7.450119

Χρόνοι για κάθε άσκηση (παράμετροι polylines_large 0.1 0):

	O0			O3		
	1 THREAD	2 THREAD	4 THREAD	1 THREAD	2 THREAD	4 THREAD
STATIC	11.047782	10.919308	10.9128	7.572700	7.432236	7.284690
DYNAMIC	10.999941	6.512788	4.704313	7.616279	4.441965	2.881026
TASK1	11.827580	9.317051	11.215510	7.660573	5.960952	6.511222
TASK2	12.451343	9.558126	11.598672	7.703629	6.002369	6.651810

*το dynamic εκτελέστηκε για chunk = AllPolylines.size()/num_threads*10.000