Παράλληλη Επεξεργασία

Εργασία Εξαμήνου

Μέλη Ομάδας (Ομάδα 48 eclass):

Γεωργόπουλος-Νίνος Νικόλαος	AM: 1054385	6ο εξ.
Κοσιώρης Νικόλαος	AM: 1054276	6ο εξ.
Παναγιωτόπουλος Γιώργος	AM: 1054377	6ο εξ.

• Ζητούμενο 1 (Static)

Στα πλαίσια του πρώτου ζητουμένου παραλληλοποιήσαμε τον σειριακό αλγόριθμο αναθέτοντας, με στατικό τρόπο, σε κάθε νήμα φόρτο εργασίας. Δηλαδή, χρησιμοποιήσαμε την εντολή omp parallel for, ενώ πρώτα γράψαμε τον βρόγχο for που διατρέχει τη λίστα των πολυγραμμών σε κανονική μορφή. Επιπλέον, χρησιμοποιήσαμε την συνάρτηση του openmp (omp_get_wtime) για να μετρήσουμε την ακριβή ώρα που είναι ενεργό το κάθε thread. Εφόσον δεν δώσαμε στην εντολή omp for παραμέτρους το πρόγραμμα χρησιμοποίησε τα defaults, άρα schedule(Static, chunk = είσοδος / αριθμός νημάτων).

Μετρώντας τους χρόνους εκτέλεσης παρατηρούμε ότι το τελευταίο νήμα παίρνει πολύ παραπάνω χρόνο από τα υπόλοιπα. Αυτό συμβαίνει διότι η λίστα των πολυγραμμών έχει τις μεγαλύτερες πολυγραμμές στο τέλος της, δηλαδή στο κομμάτι που ανατίθεται στο τελευταίο νήμα. Εφόσον το τελευταίο νήμα αργεί η παραλληλότητα του προγράμματος δεν είναι αποδοτική άρα παίρνει περισσότερο χρόνο

-O0	Threads	0	1	2	3	Total
	1	21.3672 secs	-	-	-	21.3672 secs
	2	0.212858 secs	21.1707 secs	-	-	21.590031 sec
	4	0.0677596 secs	0.208358 secs	0.800464 secs	20.6029 secs	21.447312 sec
-O3	Threads	0	1	2	3	Total
	1	6.87939 secs	-	-	-	6.87939 secs
	2	0.0622601 secs	6.8401 secs	-	-	7.073369 secs
	4	0.0278012 secs	0.0767265 secs	0.313281 secs	6.67559 secs	7.661156 secs

• Ζητούμενο 2 (Dynamic)

Σε αυτήν την υλοποίηση χρησιμοποιούμε ό,τι και στο πρόγραμμα Static αλλά προσθέτουμε την εντολή schedule(Dynamic) με το default chunk. Με την αλλαγή του chunk μπορεί να πάρουμε καλύτερο ή χειρότερο χρόνο.

Στο ερώτημα αυτό παραλληλοποιήσαμε τον αλγόριθμο Ramer-Douglas-Peucker με δυναμικό τρόπο ανάθεσης. Αναλυτικότερα, κάθε νήμα αναλαμβάνει ένα chunk στοιχείων από τη λίστα των πολυγραμμών, το απλοποιεί και στη συνέχεια προχωράει στο επόμενο chunk που δεν έχει πάρει άλλο thread. Με αυτήν τη μέθοδο υπάρχει καλύτερος διαμοιρασμός εργασίας, αφού δεν απλοποιεί ένα thread τις μεγάλες πολυγραμμές που βρίσκονται στο τέλος του αρχείου.

-O0	Threads	0	1	2	3	Total
	1	21.3778 secs	-	-	-	21.3778 secs
	2	10.7198 secs	10.3185 secs	-	-	11.098203 sec
	4	7.83062 secs	7.59667 secs	8.63166 secs	8.369secs	9.631749 sec
-O3	Threads	0	1	2	3	Total
	1	6.80995 secs	-	-	-	6.87939 secs
	2	3.50649 secs	3.37021 secs	-	-	3.681359 secs
	4	2.45516 secs	2.38691secs	2.70281 secs	2.59673secs	3.475880 secs

• Ζητούμενο 3 (Task1)

Προκειμένου να επιτευχθεί η παράλληλοποίηση του δοθέντος κώδικα με επεξεργασία μόνο της πρώτης πολυγραμμής από νέο task σε κάθε επανάληψη, χρησιμοποιήθηκε η εντολή omp task, δηλώνοντας ως first private (χρήση της αρχικοποιημένης τιμής κάθε μεταβλητής ανα task) τη μεταβλητή ανεπεξέργαστων πολυγραμμών firstline, και του συντελέστη απόκλισης epsilon, ενώ ως shared private (ενοποιημένη μεταβλητή - κοινή πρόσβαση μεταβλητής από κάθε νήμα) τη λίστα των επεξεργασμένων πολυγραμμών recResults1. Ακόμη, προστέθηκε η εντολή pragma omp taskwait, προκειμένου να τελείωσει η εκτέλεση των εντολών του task πριν συνεχιστεί η εκτέλεση του προγράμματος. Τέλος, στην πρώτη επανάληψη του αλγορίθμου (//main calculation) προστέθηκαν οι εντολές παραλληλοποίησης pragma omp parallel και pragma omp single με σκοπό την εκτέλεση του αλγορίθμου επεξεργασίας της αρχικής λίστας polyline από ένα μόνο νήμα.

Task1		
-O0	Threads	Total Time
	1	22.303608 sec
	2	15.519331 sec
	4	40.066184 sec
-O3	Threads	Total Time
	1	7.219.711
	2	5.024.788
	4	27.379595 sec

- ## Ζητούμενο 4 (Task2)

Προκειμένου να επιτευχθεί η παράλληλοποίηση του δοθέντος κώδικα με επεξεργασία μιας πολυγραμμής ανά task, χρησιμοποιήθηκε η εντολή omp task, δηλώνοντας ως first private (χρήση της αρχικοποιημένης τιμής κάθε μεταβλητής ανα task) τις μεταβλητές ανεπεξέργαστων πολυγραμμών firstline, lastline και του συντελέστη απόκλισης epsilon, ενώ ως shared private (ενοποιημένη μεταβλητή - κοινή πρόσβαση μεταβλητής από κάθε νήμα) τις λίστες των επεξεργασμένων πολυγραμμών recResults1, recResults2. Ακόμη, στην πρώτη επανάληψη του αλγορίθμου (//main calculation) προστέθηκαν οι εντολές παραλληλοποίησης pragma omp parallel και pragma omp single με σκοπό την εκτέλεση του αλγορίθμου επεξεργασίας της αρχικής λίστας polyline από ένα μόνο νήμα.

Ο λόγος που φαίνεται αυτή η μέθοδος να έχει κακό χρόνο είναι ότι ειδικά με πολλά thread δημιουργούνται αναδρομικά πολλά threads το οποίο έχει ως αποτέλεσμα overhead για το άνοιγμα και το κλείσιμο τους.

Task2		
-O0	Threads	Total Time
	1	23.140514 sec
	2	16.219487 sec
	4	31.928496 sec
-O3	Threads	Total Time
	1	7.278652 sec
	2	5.423790 sec
	4	14.302770 sec

Γραφήματα Χρονοβελτίωσης