Παράλληλος Προγραμματισμός 2019
Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Ιωάννης Αγγέλης
Α.Μ: Π2015006

Αναφορά Εργασίας

Στην συγκεκριμένη εργασία μας ζητήθηκε να υλοποιήσουμε τον αλγόριθμο quicksort με την χρήση των thread από ένα thread pool. Στο εργαστήριο του μαθήματος είχαμε υλοποίηση τον quicksort με thread . Κάθε φορά που έσπαγε ο πίνακας που ταξινομούσαμε δημιουργώντας έτσι δύο νέα threads όπου με την σειρά τους υλοποιούσανε τον αλγόριθμο quicksort. Έπεισες σε ένα άλλο εργαστήριο είχαμε υλοποιήσει ένα thread pool όπου διαθέταμε ορισμένο αριθμό από threads τα οποία μόλις τελείωναν μια εργασία αναλάμβαναν την επόμενη μέσα από μια λίστα η οποία περιείχε εργασίες (jobs). Κάθε φόρα που εμφανίζονταν μια καινούρια δουλεία πήγαινε στο τέλος της λίστας και κάθε φόρα που τελείωνε μια εργασία αφαιρούνταν μέσα από την λίστα . Μόλις τελειώσουν όλες οι εργασίες μέσα από την λίστα κλείνουμε όλα τα threads από το thread pool. Για να το υλοποιήσουμε αυτήν την τεχνική χρησιμοποιήσαμε δυο συνάρτησης την send και την recv οι οποίες πρόσθεταν και αφαιρούσαν εργασίες αντίστοιχα από την ουρά των εργασιών.

Η δική μας σειρά ήταν να ενώσουμε αυτές τις δυο τεχνικές. Στο πρόγραμμα μας υπάρχει μια καθολική ουρά όπου την βλέπουν όλοι και περιέχει σε κάθε θέση της τα στοιχεία για την εργασία που θα πάρει το thread. Πιο συγκεκριμένα περιέχει τον τύπο του μηνύματος το οποίο παίρνει δίφορες τιμές. Το WORK λέει στο thread ότι πρέπει να κάνει κάποια εργασία με τα δεδομένα του πακέτου. Το FINISH λέει στο thread ότι η συγκεκριμένη εργασία έχει τελειώσει και το SUTDOWN λέει στο thread ότι πρέπει να τερματιστεί. Κάθε εργασία που προκύπτει την αποθηκεύουμε στη λίστα με τις εργασίες όπου λέει τον τύπο της εργασίας , από πιο σημείο του πίνακα θα ταξινομήσει και σε πιο σημείο θα τελειώσει.
Αν η ουρά με τις εργασίες για κάποιο λόγο γεμίσει το thread που θα προσπαθήσει να προσθέσει την επόμενη εργασία στην send θα κλειδωθεί με την βοήθεια του mutex . Για να ξεκλειδωθεί θα πρέπει ένα άλλο thread να τραβήξει μια εργασία από την ουρά με την χρήση της recv. Η send προσθέτει εργασίες στην ουρά η recv αφαίρει εργασίες.

Στην thread_func κάθε φόρα που καλείτε περνούμε την τελευταία εργασία μέσα από την ουρά . Αν ο τύπος της εργασίας είναι WORK . Βρίσκουμε πιο είναι το μέγεθος του πίνακα που επεξεργαζόμαστε αν είναι μικρότερο από CUTOFF όπου είναι μια σταθερά που χρησιμοποιούμε τον **Insertion sort** για την ταξινόμηση αυτού του υπό πίνακα και στέλνουμε ένα μήνυμα στην ουρά ότι αυτό το κομμάτι του πίνακα έχει ταξινομηθεί. Αν το μέγεθος δεν είναι μικρότερο χρησιμοποιούμε την partision για να σπάσουμε τον πίνακα σε αλλά δυο μέρη και στέλνουμε στην ουρά άλλες δυο εργασίες με το αριστερό και το δεξί μέρος του πίνακα. Αν το μήνυμα έχει τύπο SUTDOWN στέλνουμε ένα νέο μήνυμα με τύπο SUTDOWN όπου εκτελούμε pthread_exit και τερματίζουμε το τρέχον thread επιστέφοντα την τιμή NULL. Στην περίπτωση που ο τύπος είναι FINISH στέλνουμε ένα νέο μήνυμα στην ουρά. Τέλος ελέγχουμε την ουρά για νέα μηνύματα.

Στην main μας κατασκευάζουμε έναν πίνακα από threads όπου αντιστοιχούν στο thread pool. Αρχικοποιούμε έναν πίνακα με τυχερές τιμές και τον περνάμε στην thread_func του κάθε thread. Για αρχίσουν οι δούλες στέλνουμε ένα μήνυμα που περιέχει το type WORK, την αρχή και το τέλος του πίνακα. Ελέγχουμε για νέες εργασίες μέσα σε ένα ατέρμονα βρόχο και μετράμε της εργασίες που έχουν τελείωση. Αν όλα τα στοιχεία του πίνακα έχουν ταξινομηθεί δίνουμε εντολή να κλείσουν τα thread. Σε κάθε άλλη περίπτωση προωθούμε το μήνυμα που λάβαμε.

Τέλος κάνουμε τον ελέγχω στο πίνακα στον οποίο ταξινομήσαμε και τον αποδεσμεύουμε του πόρους του συστήματος.

Αν στο πρόγραμμα μας δώσουμε μεγαλύτερο μέγεθος πίνακα το οποίο είναι να ταξινομηθεί σε σχέση με το μέγεθος της ουράς το πρόγραμμα δεν μπορεί να τρέξει.