Παράλληλος Προγραμματισμός 2019
Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Μαρία Νεφέλη Νικηφόρου
ΑΜ: Π2015054

• *Συνοπτική περιγραφή των δύο παραλλαγών του κώδικα*

Στην παρούσα άσκηση υλοποιείται ο αλγόριθμος ταξινόμησης Quicksort με μία δεξαμενή threads (thread pool). Στη δεξαμενή αυτή υπάρχουν σταθερά τέσσερα threads, τα οποία δημιουργούνται στην αρχή του προγράμματος και τερματίζουν όταν ολοκληρώνεται η ταξινόμηση. Τα threads αναλαμβάνουν πακέτα εργασίας από μια σφαιρική (global) ουρά εργασιών *msg_queue*, μεγέθους *N* (1000000). Τα πακέτα εργασίας έχουν δομή μηνύματος (βλ. struct message) και αποτελούνται από τα *type*, *first* και *last*. Το *first* είναι η θέση του πρώτου στοιχείου του τμήματος του πίνακα προς ταξινόμηση. Το *last* είναι η θέση του τελευταίου στοιχείου του τμήματος του πίνακα προς ταξινόμηση. Το *type* είναι ένας ακέραιος αριθμός που προσδιορίζει τον τύπο του μηνύματος. Οι τιμές που μπορεί να πάρει είναι οι ακόλουθες:

- 0, που σημαίνει ότι χρειάζεται επεξεργασία (WORK)
- 1, που σημαίνει ότι έχει ολοκληρωθεί η επεξεργασία του (FINISH)
- 2, που είναι το σήμα τερματισμού που θα σταλθεί μόλις ολοκληρωθεί η επεξεργασία όλων των πακέτων εργασίας, δηλ. όταν έχει ταξινομηθεί όλος ο πίνακας (SHUTDOWN)

Οι συναρτήσεις *send()* και *recv()* διαχειρίζονται και συγχρονίζουν την αποστολή και τη λήψη των μηνυμάτων, αντίστοιχα, με τη χρήση mutex και condition variables, έτσι ώστε να αποφεύγονται τα overflow και underflow στην ουρά.

Η συνάρτηση *inssort()* υλοποιεί ταξινόμηση Insertion Sort, όταν το μέγεθος του τμήματος του προς ταξινόμηση πίνακα είναι μικρότερο από το κατώφλι που έχουμε ορίσει (CUTOFF 10).

Η συνάρτηση *partition()* είναι βοηθητική για τη συνάρτηση *quicksort()* και διαμερίζει τον πίνακα σε μικρότερα και μεγαλύτερα του *pivot* στοιχεία.

Η συνάρτηση *quicksort()* λαμβάνει το αποτέλεσμα της *partition ()*, εκτελείται αναδρομικά με βάση το αποτέλεσμα αυτό, και στέλνει στην ουρά δύο μηνύματα με τα νέα πακέτα εργασίας. Η συγκεκριμένη συνάρτηση έχει ένα επιπλεόν όρισμα πέρα από τον πίνακα και το μέγεθός του. Αυτό είναι ο τύπος του μηνύματος με το οποίο ενεργοποιήθηκε, και χρησιμοποιείται κατά τον έλεγχο για το εάν έχει σταλθεί σήμα τερματισμού, ώστε να σταματήσει η αναδρομική εκτέλεση της Quicksort.

Η συνάρτηση *thread_func()* εκτελείται για κάθε thread. Αναζητά συνεχώς νέο πακέτο εργασίας στην ουρά και εκτελεί το κομμάτι ταξινόμησης που περιέχεται σε αυτό. Μόλις το thread ολοκληρώσει ένα πακέτο ταξινόμησης, η συνάρτηση στέλνει ένα μήνυμα ολοκλήρωσης τμήματος του πίνακα, ενώ παρακολουθεί το main thread για σήμα τερματισμόυ ώστε να τερματίσει το thread.

Στη συνάρτηση *main()* δημιουργείται και αρχικοποιείται με τυχαίες τιμές ο πίνακας double αριθμών προς ταξινόμηση. Έπειτα, αρχικοποιήται η ουρά εργασιών,

δημιουργούνται τα threads της δεξαμενής και τοποθετείται στην ουρά το πρώτο πακέτο εργασίας. Στη συνέχεια, παρακολουθεί την ουρά για μηνύματα ολοκλήρωσης και αθροίζοντας κάθε φορά τον αριθμό των ταξινομημένων στοιχείων, ώστε να στείλει σήμα τερματισμού μόλις ταξινομηθούν όλα τα στοιχεία του πίνακα. Όταν συμβεί αυτό, γίνεται join των threads και ελέγχεται η ορθότητα της ταξινόμησης με σύγκριση των στοιχείων του πίνκα. Τέλος, αποδεσμεύονται όλες οι δομές που χρησιμοποιήθηκαν και η εφαρμογή τερματίζεται επιτυχώς.

• *Πηγές*

Sample quicksort implementation in C, Σύνδεσμος (τελευταία ανάκτηση: 31/05/2019)

Δομές συγχρονισμού pthreads (mutexes, condition variables, barriers), Σύνδεσμος (τελευταία ανάκτηση: 31/05/2019)