Ιόνιο Πανεπιστήμιο

Σχολή Επιστημών της Πληροφορίας και Πληροφορικής

Τμήμα Πληροφορικής

Προγραμματιστική Εργασία #2

Μάθημα: Παράλληλος Προγραμματισμός
Διδάσκων: Μιχαήλ Στεφανιδάκης
Η' Εξάμηνο

Αλέξανδρος Ζερβόπουλος – Π2015111

1 Εισαγωγή

Στα πλαίσια της παρούσας εργασίας αναπτύχθηκε πρόγραμμα σε C που υλοποιεί τον αλγόριθμο Quicksort, χρησιμοποιώντας μία δεξαμενή από συγκεκριμένο αριθμό threads, ή **thread pool**. Τα threads αναλαμβάνουν πακέτα εργασίας αξιοποιώντας μία global **ουρά εργασιών**. Τόσο ο αριθμός των threads, το μέγεθος της ουράς εργασιών και το μέγεθος του πίνακα προς ταξινόμηση ορίζονται με define στην αρχή του κώδικα.

Στα υπόλοιπα κεφάλαια ακολουθεί η περιγραφή της υλοποίησης, καθώς και σύγκριση της απόδοσής της σε σχέση με τη σειριακή εκδοχή του προγράμματος.

2 Ουρά Μηνυμάτων

Η ουρά μηνυμάτων αναπαρίσταται από έναν πίνακα μεγέθους N από μεταβλητές τύπου *message*, που είναι ένα struct με τρία διαφορετικά μέλη: τον τύπο του μηνύματος *type*, τη θέση αρχής *start* και τη θέση τέλους *end*. Υπάρχουν τρεις ξεχωριστοί τύποι μηνύματος:

1. **WORK**, που χρησιμοποιείται για να δείξει ότι απαιτείται κάποια ενέργεια μεταξύ των θέσεων start και end.
2. **DONE**, που αξιοποιείται για την ενημέρωση main ότι έχουν ταξινομηθεί τα στοιχεία του πίνακα μεταξύ των θέσεων start και end.
3. **SHUTDOWN**, που λειτουργεί ως σήμα για την παύση της εκτέλεσης των threads.

Για την προσθήκη και την αφαίρεση μηνυμάτων από την ουρά χρησιμοποιούνται οι συναρτήσεις *send* και *recv*, αντίστοιχα. Οι συναρτήσεις αυτές χρησιμοποιούν τις δομές mutex και conditional variables, ώστε να αποφεύγονται προβλήματα συγχρονισμού μεταξύ των διαφόρων threads. Λεπτομέρειες για τον τρόπο, με τον οποίο αυτές αξιοποιούνται, περιγράφονται αναλυτικότερα στη συνέχεια για τη συνάρτηση recv, που λειτουργεί παρόμοια με τη send.

3 Thread Pool και Λειτουργία Threads

Το thread pool αποτελείται από έναν προκαθορισμένο αριθμό threads, που ορίζεται από τη σταθερά *THREADS*. Κατά τη δημιουργία τους, ως παράμετρος περνιέται ο πίνακας *a* προς ταξινόμηση, που έχει δημιουργηθεί δυναμικά από τη main. Στην αρχή της λειτουργίας του, κάθε thread ελέγχει την ουρά εργασιών για διαθέσιμα πακέτα, μέσω της συνάρτησης recv. Η συνάρτηση αυτή επιστρέφει με reference τα μέλη του struct που αναπαριστά τα πακέτα εργασίας, συγκεκριμένα τον τύπο του μηνύματος, και δύο δείκτες start, end προς θέσεις του πίνακα, που χρησιμοποιούνται στις διάφορες διεργασίες, απαραίτητες για την ταξινόμηση.

Η συνάρτηση recv, για να εξασφαλίσει ότι δεν θα προκύψουν θέματα συγχρονισμού, χρησιμοποιεί τις δομές mutex και conditional variable. Για την ακρίβεια, στην αρχή της recv, κάθε thread κλειδώνει το mutex και ελέγχει τον αριθμό μηνυμάτων *m_count*. Σε περίπτωση που δεν υπάρχει κανένα μήνυμα στην ουρά, ξεκλειδώνει ο mutex και το thread αναστέλλεται, μέχρι να ληφθεί σήμα για το conditional variable *msg_in*. Εναλλακτικά, εάν υπάρχουν, οι τιμές των μελών του μηνύματος ανατίθεται στις μεταβλητές που έχουν περαστεί ως ορίσματα στη συνάρτηση. Ο έλεγχος αυτός πραγματοποιείται με while, αντί για το πιθανώς αναμενόμενο if, καθώς είναι δυνατό να ξεκλειδώσουν ταυτόχρονα πολλά threads, εκ των οποίων το πρώτο

ίσως αδειάσει την ουρά, χωρίς τα υπόλοιπα να το γνωρίζουν. Στη συνέχεια, αποστέλλεται σήμα για το conditional variable *msg_out* και ξεκλειδώνει ο mutex, ώστε να ξυπνήσουν threads που πιθανώς έχουν μπλοκάρει, αν δεν υπάρχει διαθέσιμος χώρος για αποστολή στην ουρά εργασίας.

Από τη στιγμή που έχει εντοπιστεί ένα μήνυμα, ελέγχεται ο τύπος του και ανάλογα με αυτόν, εκτελούνται οι ανάλογες ενέργειες, μετά την ολοκλήρωση των οποίων, γίνεται εκ νέου έλεγχος για καινούρια μηνύματα. Συγκεκριμένα, για τύπο:

1. **WORK**. Εάν το μήκος του πίνακα, που υποδηλώνεται από τη διαφορά των δεικτών $end - start$, είναι μικρότερο της σταθεράς *THRESHOLD*, τότε ταξινομείται με τη χρήση της συνάρτησης *ins_sort*, που όπως υποδεικνύει και το όνομά της, είναι μία υλοποίηση του αλγορίθμου insertion sort, μετά την εκτέλεση της οποίας τοποθετείται στην ουρά πακέτο, με τύπο *DONE*, με το εύρος των θέσεων των τιμών του πίνακα που ταξινομήθηκαν. Διαφορετικά, ο πίνακας μοιράζεται σε δύο μέρη, γύρω από το στοιχείο pivot στη θέση *p*, μέσω της συνάρτησης partition, και στη συνέχεια τοποθετούνται στην ουρά εργασίας τα πακέτα για τα δύο νέα υποσύνολα του πίνακα.

2. **DONE**. Κάθε thread που λαμβάνει ένα τέτοιο μήνυμα, το επανατοποθετεί στην ουρά εργασίας, αφού αυτός ο τύπος μηνύματος προορίζεται για το main thread, ώστε να ενημερωθεί για την κατάσταση της εκτέλεσης.

3. **SHUTDOWN**. Εάν το μήνυμα αυτό ληφθεί, το thread σταματάει να ελέγχει για νέα μηνύματα, αφότου το τοποθετήσει στην ουρά, ώστε να προωθηθεί στα υπόλοιπα threads.

4 Λειτουργία main

Η συνάρτηση main αρχικά δεσμεύει μνήμη για τον πίνακα a προς ταξινόμηση, τύπου double και μεγέθους που καθορίζεται από τη σταθερά *SIZE*. Αν η δέσμευση δεν ολοκληρωθεί επιτυχώς, η εκτέλεση του προγράμματος τερματίζει. Ο πίνακας αρχικοποιείται με τυχαίες τιμές $r \in [0, 1]$, που προκύπτουν διαιρώντας το αποτέλεσμα της συνάρτησης *rand* με τη μέγιστη τιμή *RAND_MAX* που μπορεί αυτή να επιστρέψει. Στη συνέχεια δημιουργείται το thread pool, που αναπαρίσταται ως ένας πίνακας από μεταβλητές τύπου pthread_t, με μέγεθος THREADS. Αν η δημιουργία ενός thread δεν είναι επιτυχής, ελευθερώνεται η μνήμη για τον πίνακα a και παύει η εκτέλεση.

Έπειτα, εισάγεται στην ουρά εργασίας το πρώτο πακέτο με δείκτες 0 και SIZE, για την αρχή και το τέλος του πίνακα a, αντίστοιχα. Μετά την ολοκλήρωση αυτού, αρχίζει ο επαναλαμβανόμενος έλεγχος της ουράς για μηνύματα. Σε αντίθεση με τα threads, η main προωθεί όλα τα μηνύματα εκτός από αυτά με τύπο DONE, τα οποία χρησιμοποιεί για να καταμετρήσει τον αριθμό των θέσεων του πίνακα που έχουν ταξινομηθεί. Όταν ο αριθμός αυτός γίνει ίσος με το μέγεθος του πίνακα a, βγαίνει από το loop, όπου λαμβάνονται νέα πακέτα και αποστέλλει μήνυμα SHUTDOWN στην ουρά, ώστε να τελειώσουν τη λειτουργία τους τα threads. Η main περιμένει μέχρι να ολοκληρώσουν όλα τα threads, μέσω της συνάρτησης *pthread_join* και στη συνέχεια ελέγχει εάν υπήρξε κάποιο λάθος κατά την ταξινόμηση. Σε περίπτωση που συνέβη κάτι τέτοιο, εμφανίζεται σχετικό μήνυμα. Εναλλακτικά, εμφανίζεται μήνυμα επιτυχίας, αποδεσμεύονται οι πόροι που χρειάστηκαν και τελειώνει η εκτέλεση του προγράμματος.