Παράλληλος Προγραμματισμός 2021
Προγραμματιστική Εργασία #2

(Προσοχή: η παράδοση της άσκησης θα γίνει μέσω opencourses. Διαβάστε τις οδηγίες στο τέλος της εκφώνησης)

Θέμα

Ο σειριακός αλγόριθμος Quicksort δέχεται ως είσοδο έναν πίνακα αριθμών, επιλέγει ένα στοιχείο (pivot) και τον διαμερίζει σε δύο μέρη, με τα μικρότερα και τα μεγαλύτερα του pivot στοιχεία. Στη συνέχεια εφαρμόζει αναδρομικά την ίδια διαδικασία σε καθένα από τα δύο τμήματα.

Εάν θελήσουμε να μετατρέψουμε τον αλγόριθμο Quicksort σε παράλληλη μορφή με τη βοήθεια των POSIX Threads (pthreads), η δυναμική χρήση της αναδρομής δυσκολεύει τη μετατροπή αυτή: η διαμέριση και οι αναδρομικές κλήσεις εξαρτώνται από τα δεδομένα του πίνακα. Αν π.χ. το "πατρικό" thread κατασκευάζει δυναμικά δύο νέα threads για να χειριστούν τους δύο υποπίνακες, η μέθοδος αυτή μπορεί να δημιουργήσει ανεξέλεγκτα μεγάλο αριθμό threads, γεγονός που θα επιβαρύνει σημαντικά την απόδοση της εφαρμογής.

Στην παρούσα άσκηση ζητείται να υλοποιήσετε τον αλγόριθμο Quicksort με μία **δεξαμενή threads** (thread pool). Στη δεξαμενή αυτή θα υπάρχει ένας **σταθερός αριθμός threads** (π.χ. 4), τα οποία θα δημιουργούνται στην αρχή του προγράμματος και θα τερματίζουν όταν θα έχει ολοκληρωθεί η ταξινόμηση. Τα threads θα αναλαμβάνουν **πακέτα εργασίας** από μια σφαιρική (global) **ουρά εργασιών**.

Ζητούμενο

α) Ξεκινήστε την ανάπτυξη χρησιμοποιώντας ως βάση τον κώδικα του σειριακού quicksort από το:

https://gist.github.com/mixstef/01237891de0728d1b8aae81c6374fd74#file-quicksort-simple-c

όπου ο βοηθητικός αλγόριθμος insertion sort χρησιμοποιείται όταν ένας πίνακας έχει μέγεθος μικρότερο από το όριο CUTOFF.

β) Χρησιμοποιήστε τον κώδικα όπου υλοποιείται μια κυκλική ουρά με Ν θέσεις για μηνύματα:

https://gist.github.com/mixstef/01237891de0728d1b8aae81c6374fd74#file-cv-example-queue-c

Την ουρά αυτή (με τις κατάλληλες μετατροπές) θα χρησιμοποιήσετε ως σφαιρική ουρά εργασιών.

γ) Αποφασίστε τα είδη των μηνυμάτων που θα στέλνετε στην ουρά και φτιάξτε την κατάλληλη δομή (C struct) για να τα φιλοξενήσει. Η ουρά θα χωράει QUEUE_SIZE τέτοιες δομές. Θέστε την κατάλληλη τιμή με #define για το QUEUE_SIZE, εκτιμώντας τον μέγιστο αριθμό μηνυμάτων που μπορούν να υπάρξουν ταυτόχρονα στην ουρά.

δ) Κατασκευάστε τον κώδικα του κάθε thread:

- Κάθε thread όταν εκτελεί ένα πακέτο εργασίας θα **διαμερίζει** (partition) τον πίνακα με τον οποίο δουλεύει σε μικρότερα και μεγαλύτερα του pivot στοιχεία και θα στέλνει στην ουρά δύο **μηνύματα με τα νέα πακέτα εργασίας** (αρχή του πίνακα και μέγεθος). Στη συνέχεια, το thread

δεν θα περιμένει την ολοκλήρωση, αντιθέτως θα αναζητά στην ουρά νέα πακέτα εργασίας.

- Όταν το μήκος του πίνακα είναι μικρότερο από ένα **όριο PARTITION_LIMIT** (ορίστε το με #define PARTITION_LIMIT 10000), δεν κάνετε νέες αναθέσεις αλλά ολοκληρώνετε την ταξινόμηση επιτόπου.
- Όταν ένα thread ολοκληρώσει ένα πακέτο ταξινόμησης θα στέλνει ένα **μήνυμα ολοκλήρωσης** τμήματος του πίνακα (με το πλήθος των στοιχείων του πακέτου που ολοκληρώθηκε).

ε) Κατασκευάστε τον κώδικα του main():

- Ξεκινήστε από το main() του σειριακού quicksort (δέσμευση και αρχικοποίηση πίνακα N στοιχείων double).
- Αρχικοποιήστε την ουρά εργασιών, δημιουργήστε τα threads της δεξαμενής (αριθμός ίσος με THREADS) και τοποθετήστε στην ουρά το πρώτο πακέτο εργασίας (όλος ο αρχικός πίνακας).
- Στη συνέχεια, παρακολουθήστε την ουρά για μηνύματα ολοκλήρωσης (αθροίζοντας τον αριθμό των ταξινομημένων στοιχείων).
- Όταν έχει ολοκληρωθεί η συνολική ταξινόμηση, στείλτε **μήνυμα shutdown** και περιμένετε στο join των threads.
- Τέλος, χρησιμοποιώντας και πάλι τη σειριακή εκδοχή του κώδικα, ελέγξτε την ορθότητα της ταξινόμησης, αποδεσμεύστε όλες τις δομές που χρησιμοποιήσατε και τερματίστε την εφαρμογή.

στ) Ετοιμάστε αναφορά σε μορφή pdf, η οποία θα περιέχει:

- Σύντομη περιγραφή των τύπων μηνυμάτων που χρησιμοποιείτε.
- Πίνακα χρόνου εκτέλεσης για διάφορα μεγέθη N (10.000 έως 10.000.000) και THREADS (2 έως 16) του σειριακού και του παράλληλου κώδικα.
- Σύντομο σχολιασμό των αποτελεσμάτων.
- Αναφορές σε πηγές που χρησιμοποιήσατε.

<u>Παραδοτέο</u>

Η παράδοση θα γίνει μέσω opencourses:

1. Στο μάθημα του Παράλληλου Προγραμματισμού στο opencourses επισκεφτείτε την ενότητα «**Εργασίες**».
2. Η κατάθεση του παραδοτέου σας θα γίνει στην εργασία «**Προγραμματιστική Εργασία #2**».
3. Τοποθετήστε **την αναφορά σας** (αρχείο pdf) και **τον κώδικά σας** (αρχείο C) σε **<u>ένα (και μοναδικό) αρχείο zip</u>**.
4. Ανεβάστε το αρχείο zip στο opencourses.

Η εργασία είναι αυστηρά ατομική.

Προθεσμία παράδοσης: 16/5/2021.