HY-486

Αρχές Κατανεμημένου Υπολογισμού

Χειμερινό εξάμηνο 2022-2023

1η Προγραμματιστική Εργασία

Προθεσμία Παράδοσης: 5/12

1. Γενική Περιγραφή

Στην πρώτη προγραμματιστική εργασία καλείστε να υλοποιήσετε ένα διαμοιραζόμενο σύστημα διακίνησης προϊόντων, πάνω στο οποίο θα εκτελούνται ταυτόχρονα διάφορες λειτουργίες.

Η προγραμματιστική εργασία θα πρέπει να υλοποιηθεί με τη γλώσσα C και τη χρήση της βιβλιοθήκης pthreads.

Σημείωση: Η εργασία αποτελείται από φάσεις και είναι, εν μέρει, διαφορετική για τους προπτυχιακούς και τους μεταπτυχιακούς φοιτητές. Κάθε φάση περιγράφεται αναλυτικά παρακάτω.

2. Υλοποίηση

Στην εργασία αυτή θα πρέπει να υλοποιήσετε ένα διαμοιραζόμενο σύστημα διακίνησης προϊόντων. Οι λειτουργίες που θα πραγματοποιούνται στο σύστημα είναι:

- Η δημιουργία μιας διαμοιραζόμενης λίστας, η οποία θα περιέχει όλα τα προϊόντα του συστήματος που είναι προς πώληση. Υπεύθυνοι γι' αυτό είναι τα νήματα παραγωγοί.
- Η ταυτόχρονη δημιουργία διαμοιραζόμενων πινάκων κατακερματισμού (όπως περιγράφεται στην εκφώνηση της πρώτης σειράς θεωρητικών ασκήσεων) από τα νήματα *πωλητές*, οι οποίοι είναι υπεύθυνοι να πουλήσουν τα προϊόντα στους καταναλωτές. Οι πίνακες κατακερματισμού αναπαριστούν τους καταναλωτές και τα προϊόντα που έχει αγοράσει ο κάθε ένας.
- Τα νήματα *διαχειριστές*, μαζεύουν τα ελαττωματικά προϊόντα που έχουν επιστραφεί από τους καταναλωτές και τα αποθηκεύουν σε μια διαμοιραζόμενη στοίβα. Η στοίβα αυτή περιέχει όλα τα προϊόντα που πρόκειται να περάσουν υπό επεξεργασία.
- Όταν τα προϊόντα επισκευαστούν, επιστρέφονται πίσω στη διαμοιραζόμενη λίστα και είναι έτοιμα να πουληθούν ξανά.

Κατά την εκκίνηση του, το πρόγραμμα λαμβάνει ως είσοδο από την γραμμή εντολών (command line) έναν ακέραιο αριθμό **N**, ο οποίος καθορίζει το πλήθος των νημάτων. Αρχικά όλα τα νήματα **N** θα παίζουν το ρόλο των παραγωγών και στη συνέχεια, θα παίξουν το ρόλο των πωλητών και των διαχειριστών. Για λόγους απλότητας της άσκησης υποθέτουμε ότι ο αριθμός **N** είναι πολλαπλάσιο του 3.

Η πρώτη φάση περιλαμβάνει αρχικά τη δημιουργία των προϊόντων προς πώληση, από τα νήματα πωλητές, αποθηκεύοντάς τα σε μια διαμοιραζόμενη, ταξινομημένη και διπλά συνδεδεμένη λίστα, που λέγεται *λίστα πωλήσεων*.

Η λίστα πωλήσεων είναι μια διπλά συνδεδεμένη λίστα που υλοποιείται ως εξής:

- **Προπτυχιακοί φοιτητές**: fine synchronization
- **Μεταπτυχιακοί φοιτητές**: lazy synchronization

Η διαμοιραζόμενη λίστα αναπαρίσταται από το ακόλουθο struct:

```
struct DLLNode{
    int productID;
    pthread_mutex_tlok lock;
    struct DLLNode *next;
    struct DLLNode *prev;
}
```

```
struct LinkedList{
    struct DLLNode *head;
    struct DLLNode *tail;
}
```

Κάθε προϊόν περιγράφεται από έναν κόμβο struct DLLNode. Το πεδίο productID είναι το *μοναδικό* αναγνωριστικό του κάθε προϊόντος. Το πεδίο lock είναι κλείδωμα που έχει συσχετιστεί με τον εκάστοτε κόμβο (είναι από τη βιβλιοθήκη των pthreads). Τα πεδία next και prev είναι δείκτες προς το επόμενο και προηγούμενο στοιχείο της λίστας, αντίστοιχα.

Κάθε νήμα παραγωγός με αναγνωριστικό j (thread id) θα πρέπει να εισάγει στη λίστα πωλήσεων, N προϊόντα με τα ακόλουθα productIDs: **i*N+j** , όπου *N* είναι το πλήθος των παραγωγών, ενώ το *i* παίρνει τιμές $0 \leq i \leq N-1$.

Επομένως:

- Ο/Η παραγωγός με αναγνωριστικό j = 0 εισάγει στη λίστα πωλήσεων τα προϊόντα (struct DLLNode) με productIDs: 0, N, 2*N, 3*N, …, (N-1)*N = N^2 -N
- Ο/Η παραγωγός με αναγνωριστικό j = 1 εισάγει στη λίστα πωλήσεων τα προϊόντα με productIDs: 1, N+1, 2*N+1, 3*N+1, …, (N-1)*N+1 = N^2 -N+1
- Ο/Η παραγωγός με αναγνωριστικό j = N-1 εισάγει στη λίστα πωλήσεων τα προϊόντα με productIDs: N-1, N+(N-1), 2*N+(N-1), 3*N+(N-1), …, (N-1)*N+N-1 = N^2 -1

Η εισαγωγή των προϊόντων στην λίστα γίνεται με την συνάρτηση ListInsert(). Κάθε φορά που ένας/μια παραγωγός εισάγει ένα προϊόν στη λίστα πωλήσεων λέμε ότι ο/η παραγωγός δημιούργησε αυτό το προϊόν.

Αφού τελειώσουν όλες οι εισαγωγές στη λίστα πωλήσεων, το νήμα παραγωγός με αναγνωριστικό 0 θα επιβεβαιώσει ότι έχουν εισαχθεί όλα τα προϊόντα και έχουν σωστά, μοναδικά αναγνωριστικά. Για να εξασφαλιστεί ότι όλα τα νήματα παραγωγοί θα έχουν τελειώσει με την εισαγωγή των προϊόντων τους στη διαμοιραζόμενη λίστα, θα πρέπει να χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη pthreads.

Όταν όλα τα νήματα θα έχουν φτάσει στο πρώτο φράγμα συγχρονισμού, το νήμα παραγωγός με αναγνωριστικό 0 κάνει τις εξής δυο μετρήσεις για την επιβεβαίωση της ορθότητας:

- List size check: Το συνολικό πλήθος των προϊόντων στη λίστα πρέπει να ισούται με N^2. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  List size check (expected: N^2 , found: Y)

όπου Y είναι ο συνολικός αριθμός προϊόντων που βρέθηκαν στην λίστα.

- List keysum check: Το συνολικό άθροισμα των αναγνωριστικών (productID) των προϊόντων της λίστας, πρέπει να ισούται με $N^2(N^2-1)/2$. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  List keysum check (expected: $N^2(N^2-1)/2$, found: Y)

  όπου Y είναι ο αριθμός που εκτιμήθηκε.

Ωστόσο, αν οποιοσδήποτε από τους πιο πάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους, όπου θα φαίνεται ποιος έλεγχος απέτυχε και γιατί, ενώ η εκτέλεση θα πρέπει να τερματίζει.

Πριν ξεκινήσει η επόμενη φάση του προγράμματος, όλα τα νήματα παραγωγοί θα πρέπει να περιμένουν το νήμα παραγωγό με αναγνωριστικό 0 να ολοκληρώσει τους ελέγχους. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα δεύτερο φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη pthread.

Στη επόμενη φάση, τα N νήματα παραγωγών θα παίξουν το ρόλο των πωλητών, και θα πουλάνε τα προϊόντα της λίστας στους καταναλωτές, οι οποίοι καταναλωτές θα αναπαρίστανται ως διαμοιραζόμενοι πίνακες κατακερματισμού. Επομένως, οι πωλητές όταν πουλάνε ένα προϊόν, θα το διαγράφουν από τη λίστα (με τη χρήση της συνάρτησης ListDelete()) και θα το εισάγουν στον αντίστοιχο πίνακα κατακερματισμού που αναπαριστά τον καταναλωτή που το αγόρασε (με τη χρήση της συνάρτησης HTInsert()).

Έχουμε **M=N/3** καταναλωτές, όπου το N θα είναι πολλαπλάσιος αριθμός του 3.

Οι πίνακες κατακερματισμού των καταναλωτών υλοποιούνται ως εξής:
Open addressing με double hashing, χρησιμοποιώντας την τεχνική ordered hashing (όπως περιγράφεται και στην $1^η$ θεωρητική άσκηση)

- **Προπτυχιακοί φοιτητές:** fine grain synchronization
- **Μεταπτυχιακού φοιτητές:** lazy synchronization

Και στις δυο περιπτώσεις, ο κάθε πίνακας κατακερματισμού θα αποτελείται από 4N θέσεις (buckets) οι οποίες θα αναπαρίστανται από το ακόλουθο struct:

```
struct HTNode{
    int productID;
    pthread_mutex_tlok lock;
}
```

Κάθε πωλητής i ($0 \le i \le N-1$) εισάγει N προϊόντα στους M πίνακες κατακερματισμού, τρία στον κάθε πίνακα κατακερματισμού, ξεκινώντας από το (i mod M) πίνακα κατακερματισμού, όπου i το αναγνωριστικό του νήματος πωλητή (threadID), πηγαίνοντας προς τα δεξιά κυκλικά. Για παράδειγμα εάν N=6 και M = 2, το νήμα με αναγνωριστικό 0 θα εισάγει το πρώτο στον πίνακα κατακερματισμού 0, το δεύτερο στον πίνακα κατακερματισμού 1 και το τρίτο στον πίνακα κατακερματισμού 0 (επειδή η αρίθμηση των πινάκων κατακερματισμού ξεκινάει από το 0). Τα προϊόντα που θα εισάγει θα είναι αυτά με αναγνωριστικά στο διάστημα [N*i, (N*i)+(N-1)]

Επομένως:

- Ο πωλητής με αναγνωριστικό i = 0 εισάγει στους κατάλληλους πίνακες κατακερματισμού τα προϊόντα (struct HTnode) με productIDs: 0, 1, 2, 3, … N-1
- Ο πωλητής με αναγνωριστικό i = 1 εισάγει προϊόντα με productids:
  N, N+1, N+2, N+3,..., 2N-1
- Ο πωλητής με αναγνωριστικό i = N-1 εισάγει προϊόντα με productids:
  N*(N-1), N*(N-1)+1, N*(N- 1)+2,..., N*(N-1)+N-1

Αφού τελειώσουν όλες οι εισαγωγές στους πίνακες κατακερματισμού, το νήμα πωλητής με αναγνωριστικό 0 θα πραγματοποιήσει κάποιους ελέγχους ορθότητας. Μη ξεχάσετε να χρησιμοποιήσετε τα δυο φράγματα (barriers) για τον συγχρονισμό των νημάτων, ένα πριν και ένα μετά από τους τους ακόλουθος ελέγχους:

- HT size check: Το συνολικό πλήθος των προϊόντων ανά πίνακα κατακερματισμού, πρέπει να ισούται με 3N. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  HT[0] size check (expected: 3N , found: Y_0)

  HT[1] size check (expected: 3N , found: Y_1)

  …

  HT[N/3 -1] size check (expected: 3N , found: $Y_{N/3-1}$)

  Όπου Y_k , $0 \le k \le N/3 - 1$ είναι ο συνολικός αριθμός προϊόντων που βρέθηκαν στον πίνακα κατακερματισμού HT[k]

- HT keysum check: Το συνολικό άθροισμα των αναγνωριστικών (productID) των προϊόντων των πινάκων κατακερματισμού, πρέπει να ισούται με $N^2(N^2-1)/2$. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  HT keysum check (expected: $N^2(N^2-1)/2$, found: Y)

  Όπου Y είναι ο αριθμός που εκτιμήθηκε.

Αν οποιοσδήποτε από τους πιο πάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους, όπου θα φαίνεται ποιος απέτυχε και γιατί, καθώς η εκτέλεση θα τερματίζει.

Αφού ολοκληρωθούν οι έλεγχοι, τα N νήματα πωλητών θα γίνουν οι διαχειριστές του συστήματος. Λόγω εισαγωγής ελαττωματικών προϊόντων, πολλά έχουν επιστραφεί από τους καταναλωτές ζητώντας πίσω τα χρήματά τους. Για το λόγο αυτό οι διαχειριστές έχουν να εκτελέσουν τις εξής λειτουργείες:

i) να μαζέψουν όλα τα ελαττωματικά προϊόντα από τους καταναλωτές και
ii) αφού τα επισκευάσουν, να τα επιστέψουν στην αγορά κάνοντάς τα ξανά διαθέσιμα προς πώληση

Αρχικά τα νήματα διαχειριστές θα μαζέψουν όλα τα ελαττωματικά προϊόντα από τους πίνακες κατακερματισμού και θα τα τοποθετήσουν σε μια άλλη δομή, η οποία είναι μια διαμοιραζόμενη στοίβα που περιέχει όλα τα προϊόντα προς επισκευή. Αυτό σημαίνει ότι πρέπει να διαγραφούν από τους πίνακες κατακερματισμού του συστήματος (με τη χρήση της συνάρτησης HTDelete()) και να τα εισάγουν στην διαμοιραζόμενη στοίβα (με τη χρήση της συνάρτησης push()).

Η στοίβα υλοποιείται ως εξής: Unbounded Lock-Free Stack με χρήση locks (όπως διδάχθηκε στην Ενότητα 5. Το πρόβλημα ABA που αναφέρει στα slide, δεν θα προκύψει αφού κάθε productid είναι μοναδικό). Η δομή θα αναπαρίσταται από το πιο κάτω struct

```
struct stackNode{
    int productID;
    struct stackNode *next;
}
```

Κάθε νήμα διαχειριστής διαγράφει N/3 προϊόντα από τους πίνακες κατακερματισμού, δηλαδή ένα προϊόν από το κάθε πίνακα κατακερματισμού. Ξεκινάει διαγράφοντας από το πίνακα κατακερματισμού (i mod M) , όπου το i αναγνωριστικό του νήματος (thread ID), συνεχίζοντας προς τα δεξιά κυκλικά. Για παράδειγμα εάν N=6 και M=2, το νήμα με αναγνωριστικό 0 θα αφαιρέσει ένα προϊόν από τα πίνακα κατακερματισμού 0 και μετά από τον 1. Το προϊόν που θα αφαιρεί το κάθε νήμα από ένα πίνακα κατακερματισμού, μπορείτε να το επιλέγεται εσείς randomly. Όσα προϊόντα αφαιρούνται από τους πίνακες κατακερματισμού θα εισάγονται στην διαμοιραζόμενη στοίβα.

Για να προχωρήσουμε όλα τα νήματα διαχειριστές θα πρέπει να έχουν τελειώσει (χρήση των δύο barriers). Έπειτα, το νήμα διαχειριστής με αναγνωριστικό 0 θα πραγματοποιήσει τους εξής ελέγχους:

- HT size check: Το συνολικό πλήθος των προϊόντων ανά πίνακα κατακερματισμού, πρέπει να ισούται με 2N. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  HT[0] size check (expected: 2N , found: Y_0)

  HT[1] size check (expected: 2N , found: Y_1)

  …

  HT[N/3 -1] size check (expected: 2N , found: $Y_{N/3-1}$)

  Όπου Y_k , $0 \le k \le N/3 - 1$ είναι ο συνολικός αριθμός προϊόντων που βρέθηκαν στον πίνακα κατακερματισμού HT[k]

- Stack size check: Το συνολικό πλήθος των προϊόντων στη στοίβα πρέπει να ισούται με $N^2/3$. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  Stack size check (expected: $N^2/3$, found: Y)

  όπου Y είναι ο συνολικός αριθμός προϊόντων που βρέθηκαν στην λίστα.

Αν οποιοσδήποτε από τους πιο πάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους, όπου θα φαίνεται ποιος απέτυχε και γιατί, καθώς η εκτέλεση θα τερματίζει.

Αφού ολοκληρωθούν οι έλεγχοι, τα νήματα των διαχειριστών θα επισκευάσουν τα ελαττωματικά προϊόντα και θα τα επιστρέψουν στην αγορά κάνοντάς τα διαθέσιμα προς πώληση. Έτσι, σε αυτή τη φάση οι διαχειριστές θα αφαιρέσουν τα ελλαττωματικά προϊόντα από την στοίβα και θεωρώντας ότι έχουν επισκευαστεί, τα επανεισάγουν στη διαμοιραζόμενη λίστα.

Επομένως με την χρήση της συνάρτησης pop() θα αφαιρούμε τα προϊόντα από την στοίβα, και με την χρήση της συνάρτησης ListInsert() θα τα επανεισάγουμε στη διαμοιραζόμενη λίστα.

Για να προχωρήσουμε, όλα τα νήματα διαχειριστές θα πρέπει να έχουν τελειώσει (χρήση ενός barrier). Έπειτα, το νήμα διαχειριστής με αναγνωριστικό 0 θα πραγματοποιήσει τον εξής έλεγχο:

- List size check: Το συνολικό πλήθος των προϊόντων στη λίστα πρέπει να ισούται με $N^2/3$. Αφού ολοκληρωθεί αυτός ο έλεγχος θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

  List size check (expected: $N^2/3$, found: Y)

  Όπου Y είναι ο συνολικός αριθμός προϊόντων που βρέθηκαν στην λίστα

Αν ο πιο πάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους.

3. Αρχικοποίηση και Εκτέλεση Προγράμματος

Το πρόγραμμα θα πρέπει να δέχεται ως είσοδο ένα φυσικό αριθμό N, ο οποίος αναπαριστά το πλήθος των νημάτων και είναι πολλαπλάσιο του 3.

4. Παράδοση Εργασίας

Για την παράδοση της εργασίας θα πρέπει να χρησιμοποιήσετε το πρόγραμμα **turnin**, που υπάρχει εγκατεστημένο στα μηχανήματα του τμήματος. Συγκεκριμένα, η εντολή παράδοσης είναι:

turnin project1@hy486

Για να επιβεβαιώσετε ότι η υποβολή της εργασίας σας ήταν επιτυχής μπορείτε να χρησιμοποιήσετε την εντολή:

verify-turnin project1@hy486

Προσοχή, τα παραδοτέα σας θα πρέπει να περιέχουν ό,τι χρειάζεται και να είναι σωστά δομημένα ώστε να κάνουν compile και να εκτελούνται στα μηχανήματα της σχολής, όπου και θα γίνει η εξέταση της εργασίας.

Η προθεσμία παράδοσης της εργασίας είναι 5/12.