ΗΥ-486 – Αρχές Κατανεμημένου Υπολογισμού

Εαρινό Εξάμηνο 2019-2020

Πρώτη Προγραμματιστική Εργασία

Προθεσμία παράδοσης: Δευτέρα, 27/4 στις 23:59.

1. Γενική Περιγραφή

Στην πρώτη προγραμματιστική εργασία καλείστε να υλοποιήσετε ένα διαμοιραζόμενο σύστημα αναπαραγωγής μουσικής (music streaming system), πάνω στο οποίο θα εκτελούνται διάφορες λειτουργίες ταυτόχρονα.

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

Προσοχή: Η εργασία είναι, εν μέρει, διαφορετική για τους προπτυχιακούς και τους μεταπτυχιακούς φοιτητές. Κάθε μέρος αναφέρεται αναλυτικά στη συνέχεια.

2. Υλοποίηση

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

- Η δημιουργία ενός διαμοιραζόμενου δένδρου από ένα σύνολο από νήματα, τους *παραγωγούς*, στο οποίο θα συγκεντρώνονται όλα τα αρχεία μουσικής του συστήματος. Το δένδρο που πρέπει να υλοποιήσουν οι προπτυχιακοί φοιτητές θα έχει βαθμό 2, ενώ εκείνο που θα υλοποιήσουν οι μεταπτυχιακοί φοιτητές θα έχει βαθμό 4.
- Η ταυτόχρονη προσπέλαση του δένδρου από ένα άλλο σύνολο από νήματα, τους χρήστες.
- Η ταυτόχρονη δημιουργία ουρών αναπαραγωγής (play queues) από τους χρήστες.

Πρέπει να σημειωθεί, ότι κάθε νήμα παραγωγού «μεταφορτώνει» (uploads) τραγούδια με ένα συγκεκριμένο, μοναδικό id, και ο αριθμός των νημάτων θα δίνεται ως είσοδος στο πρόγραμμα (στην main).

Πιο συγκεκριμένα, στην πρώτη φάση, που είναι η φάση της «μεταφόρτωσης» (uploading) τραγουδιών, κάθε νήμα παραγωγός θα παράγει τραγούδια, κάθε ένα εκ των οποίων θα αναπαρίσταται από το ακόλουθο struct (που διαφοροποιείται μεταξύ των προπτυχιακών και των μεταπτυχιακών φοιτητών):

Προπτυχιακοί φοιτητές:	**Μεταπτυχιακοί φοιτητές:**

struct treeNode {
int songID;
struct node *lc;
struct node *rc;
pthread_mutex_t lock;
}
	struct treeNode {
	int songIDs[3];
	struct node *children[4];
	pthread_mutex_t lock;
	}

Στο παραπάνω struct, το πεδίο songID είναι το μοναδικό αναγνωριστικό του κάθε τραγουδιού, το πεδίο lc είναι δείκτης προς το αριστερό παιδί του κόμβου, το πεδίο rc είναι δείκτης προς το δεξιό παιδί του κόμβου και το πεδίο lock είναι το lock (από τη βιβλιοθήκη των pthreads) που έχει συσχετισθεί με τον κόμβο. Στην περίπτωση των μεταπτυχιακών φοιτητών ο πίνακας songIDs αποθηκεύει τις (έως) τρεις τιμές του κόμβου ενώ ο πίνακας children αποθηκεύει δείκτες προς τα (έως) τέσσερα παιδιά.

Το νήμα με αναγνωριστικό j (thread ID) θα πρέπει να μεταφορτώνει N τραγούδια με songIDs: **i*N + j,** όπου *N* είναι το πλήθος των παραγωγών, ενώ το *i* παίρνει τιμές $0 \leq i \leq N-1$.

Επομένως:

- Ο παραγωγός με αναγνωριστικό **j=0** ανεβάζει τραγούδια (struct treeNode) με songIDs: **0, N, 2*N, 3*N, …, (N-1)*N = N^2-N.**
- Ο παραγωγός με αναγνωριστικό **j=1** ανεβάζει τραγούδια με songIDs**: 1, N+1, 2*N+1, 3*N+1,..., (N– 1)*N + 1 = N^2-N+1**
- Ο παραγωγός με αναγνωριστικό **j=N-1** ανεβάζει τραγούδια με songIDs**: N-1, N + (N-1), 2*N+ (N-1),..., (N – 1)* N + N-1 = N^2 - 1.**

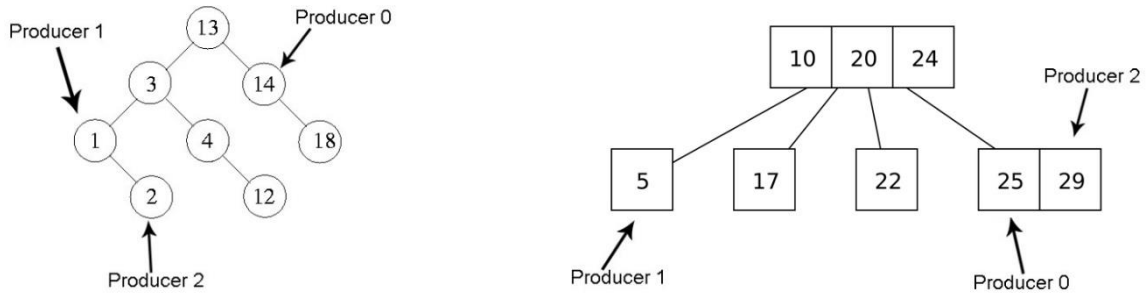
Κάθε τραγούδι που ανεβάζει ο κάθε παραγωγός, το εισάγει (καλώντας την BSTInsert) στο διαμοιραζόμενο δένδρο (concurrent tree) το οποίο θα πρέπει να υλοποιηθεί με τον εξής τρόπο:

Προπτυχιακοί φοιτητές:

- **Δένδρο δυαδικής αναζήτησης που υλοποιείται χρησιμοποιώντας fine-grained synchronization,** όπως σας ζητήθηκε να υλοποιήσετε στην 1η σειρά θεωρητικών ασκήσεων, Άσκηση 3. Το δένδρο είναι ταξινομημένο βάσει του πεδίου songID (έτσι ώστε αν πραγματοποιηθεί ενδοδιατεταγμένη διάσχιση πάνω στο δένδρο, τα songIDs να εμφανίζονται σε αύξουσα διάταξη).

Μεταπτυχιακοί φοιτητές:

- **Top-down (2,3, 4) δένδρο που υλοποιείται χρησιμοποιώντας fine-grained synchronization,** όπως σας ζητήθηκε να υλοποιήσετε στην 1η σειρά θεωρητικών ασκήσεων, Άσκηση 3.



Παράδειγμα διαμοιραζόμενου δένδρου. Αριστερά παρουσιάζεται το δυαδικό δένδρο αναζήτησης που έχουν να υλοποιήσουν οι προπτυχιακοί φοιτητές και δεξιά το (2-3-4)-δένδρο που έχουν να υλοποιήσουν οι μεταπτυχιακοί φοιτητές. Τα δένδρα είναι ταξινομημένα βάσει του πεδίου songID.

Πριν προχωρήσουν στην επόμενη φάση, όλα τα νήματα-παραγωγοί θα πρέπει να έχουν τελειώσει τη φάση της μεταφόρτωσης, δηλαδή την εισαγωγή των τραγουδιών που το καθένα μεταφορτώνει στο διαμοιραζόμενο δένδρο. Για να επιτευχθεί αυτό, θα χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier, από τη βιβλιοθήκη των pthreads) ώστε τα νήματα να συγχρονιστούν κατά το πέρας της φάσης της μεταφόρτωσης και να ξεκινήσουν συγχρονισμένα την επόμενη φάση. Η εκτέλεση κάθε νήματος θα φτάνει στο φράγμα συγχρονισμού (barrier) αφότου το νήμα τελειώσει την εισαγωγή όλων των τραγουδιών του στο διαμοιραζόμενο δένδρο.

Αφού ολοκληρωθεί η φάση της μεταφόρτωσης, το νήμα με αναγνωριστικό 0 θα πραγματοποιεί τους εξής δύο ελέγχους:

1. **Tree size check**: Το συνολικό πλήθος των τραγουδιών στο δένδρο πρέπει να ισούται με N^2. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:
   total size check passed (expected: N^2, found: Y)

   όπου N^2 είναι η προβλεπόμενη τιμή του συνολικού πλήθους των τραγουδιών και Y είναι ο συνολικός αριθμός τραγουδιών που βρέθηκαν στο δένδρο.

2. **Total keysum check**: Το άθροισμα των songIDs των τραγουδιών όλου του δένδρου πρέπει να ισούται με $N^2(N-1)(N+1)/2$. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:
   total keysum check passed (expected: $N^2(N-1)(N+1)/2$, found: Y)

   όπου X είναι η προβλεπόμενη τιμή του αθροίσματος των songIDs όλων των τραγουδιών και Y είναι το άθροισμα των songIDs των τραγουδιών που βρέθηκαν στο δένδρο.

Αν οποιοσδήποτε από τους παραπάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους (όπου θα φαίνεται ποιος έλεγχος απέτυχε και γιατί) και η εκτέλεση θα τερματίζει.

Στη συνέχεια, τα *N* νήματα παραγωγών θα γίνουν οι χρήστες, τα οποία θα διαβάζουν κόμβους από το δένδρο (εκτελώντας λειτουργίες BSTSearch) και θα εισάγουν τα τραγούδια σε *M = N/2* διαμοιραζόμενες ουρές (concurrent queues), καλώντας την enqueue. Μπορείτε να υποθέσετε ότι το N θα είναι άρτιος αριθμός.

Κάθε διαμοιραζόμενη ουρά θα πρέπει να υλοποιηθεί ως εξής:

Προπτυχιακοί φοιτητές:

- **Unbounded Total Queue με χρήση locks,** όπως έχετε διδαχθεί στο μάθημα (κεφάλαιο 5, διαφάνειες 2 και 3). Η δομή θα αναπαρίσταται από το struct:

```
struct queue {
    struct queueNode *Head;
    struct queueNode *Tail;
    pthread_mutex_t headLock;
    pthread_mutex_t tailLock;
}
```

Το struct που θα περιγράφει τον κάθε κόμβο της ουράς θα είναι της μορφής:

```
struct queueNode {
    int songID;
    struct queueNode *next;
}
```

Μεταπτυχιακοί φοιτητές:

- **Unbounded Lock-Free Queue χωρίς τη χρήση locks** (non-blocking algorithm of Michael and Scott), όπως έχετε διδαχθεί στο μάθημα (κεφάλαιο 5, διαφάνειες 4 και 5). Η δομή θα αναπαρίσταται από το struct:

```
struct queue {
    struct queueNode *Head;
    struct queueNode *Tail;
}
```

Το struct που θα περιγράφει τον κάθε κόμβο της ουράς θα είναι της μορφής:

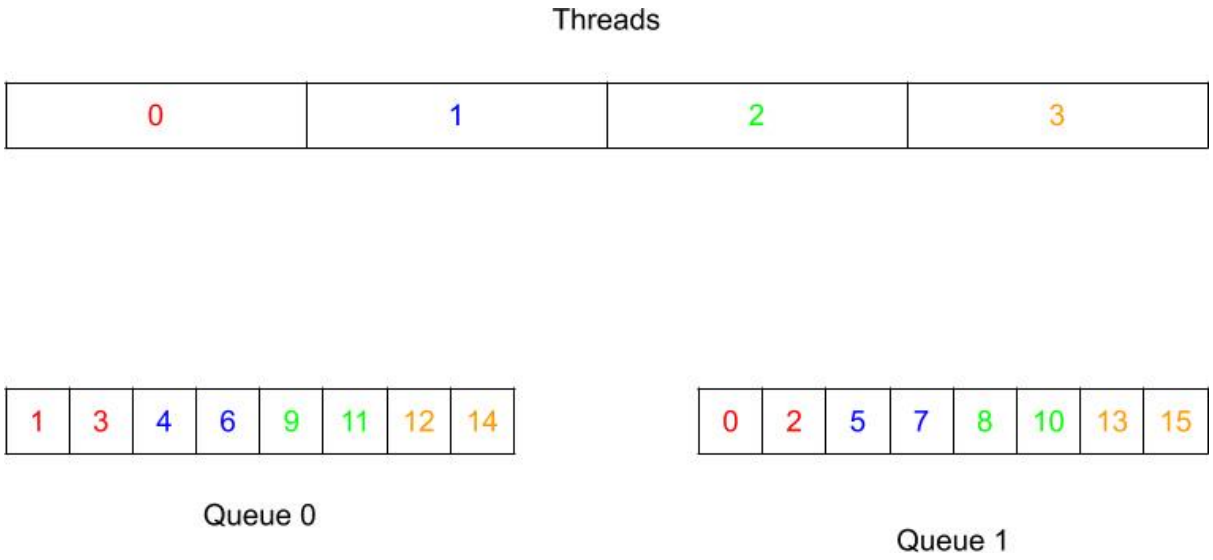
```
struct queueNode {
    int songID;
    struct queueNode *next;
}
```

Κάθε χρήστης i ($0 \leq i \leq N-1$) εισάγει *N* τραγούδια στις *M* ουρές, δύο στην κάθε μια, ξεκινώντας από την ((i+1) mod M) ουρά, όπου i το αναγνωριστικό του νήματος (thread ID), πηγαίνοντας προς τα δεξιά

κυκλικά. Για παράδειγμα εάν N = 4, το νήμα με αναγνωριστικό 0 θα εισάγει το πρώτο στην ουρά 1, το δεύτερο στην ουρά 0, το τρίτο στην 1 και το τέταρτο στην 0, με αυτή τη σειρά. Παρομοίως, το νήμα με αναγνωριστικό 1 θα εισάγει στην ουρά 0, στην 1, στην 0 και στην 1 (επειδή η αρίθμηση ξεκινάει από το 0), με αυτή τη σειρά. Τα τραγούδια που θα εισάγει θα είναι αυτά με αναγνωριστικά στο διάστημα [N * i, (N * i) + (N-1)].

Επομένως:

- Ο χρήστης με αναγνωριστικό **i=0** εισάγει τραγούδια (struct queueNode) με songIDs: **0, 1, 2, 3, …, N-1=N.**
- Ο χρήστης με αναγνωριστικό **i=1** εισάγει τραγούδια με songIDs**: N, N+1, N+2, N+3,..., 2N-1=N**
- Ο χρήστης με αναγνωριστικό **i=N-1** εισάγει τραγούδια με songIDs: **N*(N-1), N*(N-1)+1, N*(N-1)+2,..., N*(N-1)+N-1= N.**



Παράδειγμα για N=4. Οι 2 ουρές έχουν συνολικά: $\frac{N^2}{2}$*,=8 στοιχεία, και το άθροισμα των αναγνωριστικών (songIDs) της κάθε ουράς είναι 60.*

Πριν προχωρήσουν στην επόμενη φάση, όλα τα νήματα θα πρέπει να έχουν τελειώσει τη φάση της εισαγωγής στις ουρές. Για να επιτευχθεί αυτό, θα χρησιμοποιήσετε δύο ακόμη φράγματα συγχρονισμού (barriers) ώστε τα νήματα να συγχρονιστούν με το πέρας αυτής της φάσης και να ξεκινήσουν συγχρονισμένα την επόμενη φάση.

Αφού ολοκληρωθεί η φάση της εισαγωγής στις ουρές, το νήμα με αναγνωριστικό 0 θα πραγματοποιήσει τους εξής δύο ελέγχους:

1. **Queue size check:** Το συνολικό πλήθος των τραγουδιών της **κάθε** ουράς πρέπει να ισούται με 2**N*. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

   queues' total size check passed (expected: N^2, found: Y)

   όπου N^2 είναι η προβλεπόμενη τιμή του συνολικού πλήθους των τραγουδιών και Y είναι ο συνολικός αριθμός τραγουδιών που βρέθηκαν στις ουρές.

2. **Total keysum check**: Το άθροισμα των songIDs των τραγουδιών όλων των ουρών πρέπει να ισούται με $N^2(N-1)(N+1)/2$. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

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

   όπου Y είναι το άθροισμα των songIDs των τραγουδιών που βρέθηκαν στις ουρές.

Αν οποιοσδήποτε από τους παραπάνω ελέγχους αποτύχει, το πρόγραμμα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους (όπου θα φαίνεται ποιος έλεγχος απέτυχε και γιατί) και η εκτέλεση θα τερματίζει.

Αφού ολοκληρωθούν οι έλεγχοι, τα N νήματα χρηστών θα γίνουν οι διαχειριστές του συστήματος. Λόγω αλλαγών στις συμφωνίες που έχουν κάνει κάποιοι μουσικοί με την εταιρεία της εφαρμογής για τα πνευματικά δικαιώματα, κάποια τραγούδια πρέπει να φύγουν προσωρινά από την εφαρμογή και το σύστημα. Αυτό σημαίνει ότι πρέπει να διαγραφούν από το δένδρο του συστήματος και από τις λίστες αναπαραγωγής των χρηστών και να αποθηκευτούν σε μια άλλη δομή που θα είναι μια διαμοιραζόμενη λίστα (concurrent list) στην οποία εισαγωγές θα πραγματοποιούνται καλώντας μια συνάρτηση LL-Insert. Οι διαγραφές από το δένδρο και τις ουρές, καθώς και οι εισαγωγές στη λίστα, θα πρέπει να γίνονται ταυτόχρονα.

Κάθε διαμοιραζόμενη λίστα θα πρέπει να υλοποιηθεί ως εξής:

Προπτυχιακοί φοιτητές: Linked List with optimistic synchronization, όπως διδάχθηκε στο μάθημα (κεφάλαιο 5).
Μεταπτυχιακοί φοιτητές: Linked List with lazy synchronization, όπως διδάχθηκε στο μάθημα (κεφάλαιο 5).

Σε κάθε περίπτωση, η διαμοιραζόμενη λίστα αναπαρίσταται από το struct:

```
struct list {
        struct listNode *head;
        struct listNode *tail;
}
```

Κάθε κόμβος της λίστας αναπαρίσταται από το struct:

```
struct listNode {
        int songID;
        struct listNode *next;
        bool marked;            //για τους μεταπτυχιακούς φοιτητές
        pthread_mutex_t lock;
}
```

Η ταξινόμηση των τραγουδιών θα γίνεται βάσει του songID.

Πιο συγκεκριμένα, κάθε νήμα-διαχειριστής διαγράφει $\frac{N}{2}$ τραγούδια από τις ουρές (με χρήση της dequeue), ένα από την κάθε μια ξεκινώντας από την ((i+1) mod M) ουρά, όπου i το αναγνωριστικό του νήματος (thread ID), πηγαίνοντας προς τα δεξιά κυκλικά. Για παράδειγμα εάν N = 4, το νήμα-διαχειριστής με αναγνωριστικό 0 θα αφαιρέσει ένα τραγούδι από κάθε μια από τις ουρές 1 και 0, με αυτή τη σειρά.

Κάθε φορά που ένα νήμα-διαχειριστής διαγράφει ένα τραγούδι από μια ουρά, αναζητά το τραγούδι βάσει του μοναδικού αναγνωριστικού του στο δένδρο (με χρήση της BSTSearch) και το διαγράφει (με χρήση της BSTDelete). Τέλος, το εισάγει στη διαμοιραζόμενη λίστα.

Συγκεντρωμένες όλες οι απαιτούμενες ενέργειες είναι:

1. Αφαίρεση (καλώντας την dequeue) ένα τραγούδι από την κατάλληλη ουρά
2. Αναζήτηση στο διαμοιραζόμενο δένδρο (καλώντας την BSTSearch) και διαγραφή (καλώντας την BSTDelete)
3. Εισαγωγή (καλώντας την LL-Insert) στη διαμοιραζόμενη λίστα.

Αφού ολοκληρωθεί η φάση της εισαγωγής στη διαμοιραζόμενη λίστα, το νήμα με αναγνωριστικό 0 θα πραγματοποιήσει τους εξής ελέγχους (θα χρειαστεί ακόμη ένα φράγμα συγχρονισμού):

1. **Tree size check:** Το συνολικό πλήθος των τραγουδιών στο δένδρο πρέπει να ισούται με $\frac{N^2}{2}$. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

   tree’s size check passed (expected: $\frac{N^2}{2}$, found: Y)

όπου $\frac{N^2}{2}$ είναι η προβλεπόμενη τιμή του μισού πλήθους των τραγουδιών και Y είναι ο συνολικός αριθμός τραγουδιών που βρέθηκαν στο δένδρο.

2. **Queue's size check**: Το συνολικό πλήθος των τραγουδιών της **κάθε** ουράς πρέπει να ισούται με N. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

   queues' total size check passed (expected: $\frac{N^2}{2}$, found: Y)

   όπου $\frac{N^2}{2}$ είναι η προβλεπόμενη τιμή του μισού του συνολικού πλήθους των τραγουδιών και Y είναι ο συνολικός αριθμός τραγουδιών που βρέθηκαν στις ουρές.

3. **List's size check**: Το συνολικό πλήθος των τραγουδιών στη λίστα πρέπει να ισούται με $\frac{N^2}{2}$. Αφού ολοκληρωθεί αυτός ο έλεγχος, θα πρέπει να τυπώνεται στην οθόνη το μήνυμα:

   list's size check passed (expected: $\frac{N^2}{2}$, found: Y)

   όπου $\frac{N^2}{2}$ είναι η προβλεπόμενη τιμή του μισού πλήθους των τραγουδιών και Y είναι ο συνολικός αριθμός τραγουδιών που βρέθηκαν στη λίστα.

3. Αρχικοποίηση κι Εκτέλεση Προγράμματος

Το πρόγραμμα θα πρέπει να δέχεται ως είσοδο έναν φυσικό αριθμό N, ο οποίος αναπαριστά το πλήθος της κάθε οικογένειας threads (N νήματα-παραγωγοί, N νήματα-χρήστες και N νήματα-διαχειριστές).

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 την **Δευτέρα 27/4 στις 23:59**.