ΗΥ486: Αρχές Κατανεμημένου Υπολογισμού

Εαρινό Εξάμηνο 2025

Δεύτερη Προγραμματιστική Εργασία

Προθεσμία Παράδοσης: 14/6/2025

1. Γενική Περιγραφή

Στη δεύτερη προγραμματιστική εργασία καλείστε να υλοποιήσετε ένα κατανεμημένο σύστημα διαχείρισης βιβλιοθηκών, το οποίο θα διαχειρίζεται γεγονότα εισόδου και εξόδου από τους δανειστές. Η προγραμματιστική εργασία θα πρέπει να υλοποιηθεί στη γλώσσα C με τη χρήσητου Message Passing Interface (MPI), το οποίο υπάρχει εγκατεστημένο στα μηχανήματα της σχολής.

2. Υλοποίηση

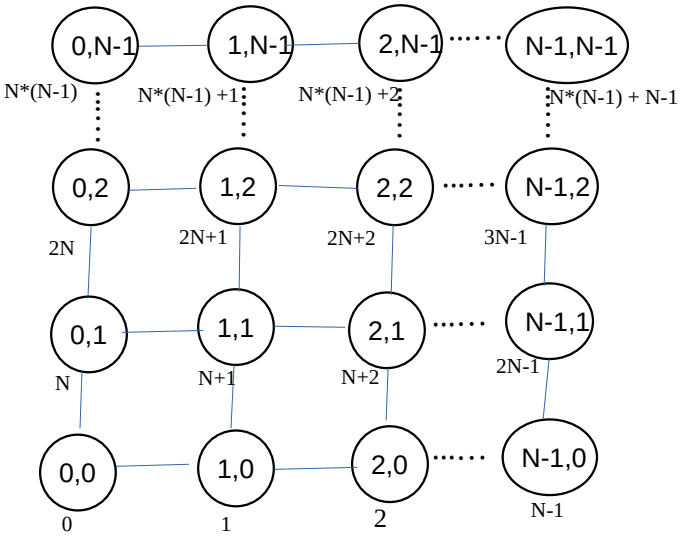
Στην εργασία αυτή θα πρέπει να υλοποιήσετε ένα σύστημα διαχείρισης βιβλιοθηκών. Το σύστημα αποτελείται από διεργασίες **βιβλιοθήκες** (servers) και διεργασίες **δανειστές** (clients). Κάθε διεργασία δανειστής μπορεί να αιτείται την καταχώρηση ή την παρακολούθηση γεγονότων σε μια συγκεκριμένη διεργασία βιβλιοθήκης στην οποία αντιστοιχεί. Οι διεργασίες βιβλιοθήκες είναι υπεύθυνες για την πραγματοποίηση αυτών των λειτουργιών. Για λόγους απλότητας θεωρούμε ότι κάθε διεργασία δανειστής (client) έχει αντιστοιχιστεί σε μια μόνο βιβλιοθήκη.

Με βάση τα παραπάνω, το σύστημα θα πρέπει να διαμορφωθεί ως εξής:

- Στο σύστημα θα υπάρχουν διεργασίες δανειστές που ζητούν την εκτέλεση αιτημάτων και διεργασίες βιβλιοθήκες που τα εξυπηρετούν.
- Η διεργασία με αναγνωριστικό (MPI rank) 0 θα παίζει το ρόλο του **συντονιστή** (coordinator), του οποίου κύριος σκοπός είναι να διαβάζει ένα testfile που θα περιέχει την περιγραφή των διαφόρων γεγονότων που πρέπει να προσομοιώνουν οι υπόλοιπες διεργασίες και να τις ενημερώνει για αυτό. Άρα, η διεργασία συντονιστής δεν αποτελεί ούτε διεργασία βιβλιοθήκης, ούτε διεργασία δανειστή. Η δομή του testfile, καθώς και τα γεγονότα περιγράφονται αναλυτικά στη συνέχεια.

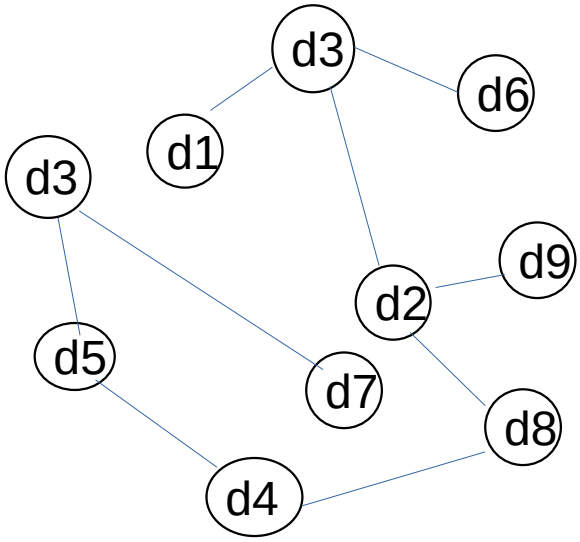
Οι κόμβοι των βιβλιοθηκών δημιουργούν ένα λογικό δίκτυο που έχει τη μορφή πλέγματος (Grid) μεγέθους NxN , όπως φαίνεται στο Σχήμα 1. Άρα, υπάρχουν N^2 διεργασίες βιβλιοθήκες στο σύστημα. Ο κόμβος με συντεταγμένες <x,y> στο πλέγμα έχει έναν ή δύο γείτονες στη γραμμή στην οποία ανήκει και έναν ή δύο γείτονες στη στήλη στην οποία ανήκει στο πλέγμα. Ο κόμβος με συντεταγμένες <0,y> δεν έχει αριστερό γείτονα, ενώ ο δεξιός του γείτονας έχει συντεταγμένες <1,y>. Παρομοίως, ο κόμβος με συντεταγμένες <N-1,y> δεν έχει δεξιό γείτονα, ενώ ο αριστερός του γείτονας έχει συντεταγμένες <N-2,y>. Επιπρόσθετα, οι κόμβοι με συντεταγμένες <x,0> δεν έχουν κάτω γείτονες και οι κόμβοι με συντεταγμένες <x,N-1> δεν έχουν πάνω γείτονες. Αν ο κόμβος έχει δύο γείτονες στη γραμμή στην οποία ανήκει, τότε ο αριστερός του γείτονας έχει συντεταγμένες <x-1, y>, ενώ ο δεξιός του γείτονας έχει συντεταγμένες <x+1, y>. Επίσης, αν ο κόμβος έχει δύο γείτονες στη στήλη στην οποία ανήκει, τότε ο πάνω γείτονας του κόμβου είναι ο κόμβος με συντεταγμένες <x,y+1> ενώ ο κάτω

γείτονας έχει συντεταγμένες <x,y-1>. Επιπρόσθετα, για κάθε i,j, $0 \le i,j \le N-1$, η βιβλιοθήκη στην θέση (i,j) αντιστοιχεί στη διεργασία βιβλιοθήκη με αναγνωριστικό l_id = N*j + i. Οι διεργασίες δανειστές είναι $N^3/2$ (Στρογγυλοποιήστε το $N^3/2$ προς τα κάτω) και έχουν αναγνωριστικά c_id = N^2+j, για κάθε j, $0 \le j \le (N^3/2)-1$. Κάθε βιβλιοθήκη έχει N/2 δανειστές που ανήκουν σε αυτήν. Συγκεκριμένα, η βιβλιοθήκη με αναγνωριστικό k, $0 \le k \le N^2-1$ έχει τους εξής δανειστές: N^2 + k*N/2,… , N^2 + (k+1)*N/2. (Ο ορισμός του N περιγράφεται στο κεφάλαιο 4.)



Βιβλιοθήκες

Σχήμα 1α



Δανειστές

Σχήμα 1β

Για να υλοποιηθεί αυτό το πλέγμα θα πρέπει ο κάθε κόμβος βιβλιοθήκης να κρατάει πληροφορίες για τους γειτονικούς του κόμβους στο πλέγμα. Σημαντικό μέρος της άσκησης είναι η υλοποίηση ενός αλγόριθμου δρομολόγησης μηνυμάτων στο πλέγμα των κόμβων βιβλιοθηκών. Επιπρόσθετα, κάθε διεργασία βιβλιοθήκης κρατάει πληροφορίες για τα βιβλία που έχει, ποια από αυτά έχει δανείσει και πόσες φορές το κάθε βιβλίο έχει δανειστεί. Για το λόγο αυτό, κάθε διεργασία βιβλιοθήκη l_id υλοποιεί ένα σύνολο βιβλίων (σειριακή δομή) που περιέχει όλα τα βιβλία που βρέθηκαν κάποια στιγμή στην βιβλιοθήκη (αποθηκεύτηκαν έστω για λίγο) ως π.χ., μια απλά συνδεδεμένη λίστα. Το κάθε βιβλίο έχει 2 χαρακτηριστικά: το αναγνωριστικό (id) του βιβλίου και το κόστος (cost) του βιβλίου.

Θεωρούμε ότι υπάρχουν $M = N^4$ βιβλία συνολικά και ότι στην κάθε μια από τις N^2 βιβλιοθήκες έχει από N διαφορετικά βιβλία, το κάθε ένα σε αριθμό N αντιτύπων. Κάθε βιβλιοθήκη έχει N^2 βιβλία αλλά μόνο N βιβλία είναι διακριτά μεταξύ τους. Η βιβλιοθήκη με αναγνωριστικό i, $0 \le i \le N^2-1$ έχει τα βιβλία με αναγνωριστικά b_id = i * N , …., (i+1) * (N) – 1. Η κάθε βιβλιοθήκη πρέπει να αρχικοποιηθεί με τα βιβλία που τις αντιστοιχούν σε σωστό αριθμό αντιτύπων. Για το κόστος (cost) χρησιμοποιήστε έναν τυχαίο αριθμό μεταξύ του 5 και του 100.

Ο κάθε δανειστής πρέπει να αποθηκεύει πληροφορία για το ποια βιβλία έχει δανειστεί καθώς και για πόσες φορές έχει δανειστεί το κάθε ένα. Για να υλοποιηθεί αυτό, κάθε διεργασία δανειστή πρέπει να υλοποιεί ένα σύνολο με αντίστοιχο τρόπο όπως και οι διεργασίες βιβλιοθήκες. Ο γράφος

των δανειστών (Σχήμα 1β) είναι ένα δένδρο και σχηματίζεται βάσει των γεγονότων τύπου CONNECT που περιγράφονται στην ενότητα 2.2.

Το σύστημα θα υποστηρίζει τα ακόλουθα γεγονότα (events στο testfile):

1. Δημιουργία αντιστοίχισης δένδρου δανειστών (CONNECT).
2. Δανεισμός βιβλίου από δανειστή (takeBook).
3. Δωρεά βιβλίων από κάποιον δανειστή στην βιβλιοθήκη του (donateBook).
4. Εντοπισμός του πιο δημοφιλές βιβλίου (getMostPopularBook).
5. Έλεγχος ότι δεν έχει χαθεί κάποιο βιβλίο (checkNumberOfBooksLoaned).

2.1 Εκκίνηση

Κατά την εκκίνηση του συστήματος, εκτελείται μία ρουτίνα, που ονομάζεται MPI_init(), η οποία αρχικοποιεί όλες τις διεργασίες που θα εκτελεστούν στο σύστημα (δείτε λεπτομέρειες για την αρχικοποίηση του συστήματος στην Ενότητα 4).

2.2 Δημιουργία Δένδρου Διεργασιών Δανειστών

Οι διεργασίες δανειστές συνδέονται μεταξύ τους ώστε να σχηματίσουν ένα δένδρο (Σχήμα 1β). Το δένδρο καθορίζεται από τα γεγονότα τύπου CONNECT που περιγράφονται στο testfile. Η διεργασία συντονιστής ξεκινάει να διαβάζει το testfile, το οποίο αρχικά περιγράφει τη συνδεσμολογία των διεργασιών προκειμένου να δημιουργηθεί το δένδρο που τις συνδέει (χωρίς ωστόσο να καθοριστεί κάποιος κόμβος ως ρίζα). Συγκεκριμένα, τα γεγονότα αυτά έχουν την εξής μορφή:

CONNECT <client_id1> <client_id2>

Κατά το γεγονός αυτό, ο συντονιστής αρχικά στέλνει ένα μήνυμα τύπου CONNECT στη διεργασία με αναγνωριστικό <client_id1> ώστε να την ενημερώσει ότι ο γειτονικός της κόμβος στο δένδρο είναι η διεργασία με αναγνωριστικό <client_id2>. Η διεργασία με αναγνωριστικό <client_id1> καταγράφει τη διεργασία με <client_id2> ως γειτονική της και στη συνέχεια της στέλνει ένα μήνυμα (τύπου NEIGHBOR) ώστε να την ενημερώσει ότι είναι γειτονικός της κόμβος. Η διεργασία με αναγνωριστικό <client_id2> την καταγράφει ως έναν από τους γειτονικούς της κόμβους και της στέλνει πίσω ένα μήνυμα επιβεβαίωσης ACK. Όταν η διεργασία <client_id1> λάβει το ACK στέλνει με τη σειρά της ένα μήνυμα ACK στο συντονιστή για να τον ενημερώσει για την περάτωση της εκτέλεσης του γεγονότος αυτού.

Για κάθε ένα από τα παραπάνω μηνύματα τύπου **<CONNECT>**, η διεργασία συντονιστής θα πρέπει να περιμένει ένα μήνυμα τύπου **<ACK>** από τη διεργασία στην οποία έστειλε το μήνυμα, πριν προχωρήσει στην επεξεργασία του επόμενου γεγονότος του testfile. Με αυτόν τον τρόπο εξασφαλίζεται ότι όλες οι διεργασίες πελάτες γνωρίζουν το ρόλο τους και τη συνδεσμολογία τους στον δένδρο πελατών κι έτσι μπορεί να ξεκινήσει ομαλά η προσομοίωση του συστήματος. Η οποία περιγράφεται στο test-file μέσω τον γεγονότων που ακολουθούν τα γεγονότα τύπου <CONNECT>

2.3 Εκλογή Αρχηγού Μεταξύ Διεργασιών Βιβλιοθήκης

Στη συνέχεια, το test_file περιέχει ένα γεγονός τύπου <**START_LE _LIBR**>

Όταν ο συντονιστής διαβάσει ένα τέτοιο γεγονός στο testfile αποστέλλει ένα μήνυμα τύπου <START_LEADER_ELECTION> σε κάθε διεργασία βιβλιοθήκη που υπάρχει στο σύστημα και

έπειτα περιμένει να λάβει ένα μήνυμα τύπου <LE_LIBR_DONE> από τη διεργασία που θα εκλεγεί αρχηγός. Ακολουθεί η περιγραφή του αλγορίθμου εκλογής αρχηγού για τις διεργασίες βιβλιοθήκες.

Εκλογή Αρχηγού μεταξύ των διεργασιών βιβλιοθηκών (Library Leader Election). Ο αλγόριθμος εκλογής αρχηγού που πρέπει να υλοποιήσετε είναι ο εξής:

Constructing a DFS Spanning Tree without a Specified Root (σελίδα 14, Ενότητα 7 διαφανειών).

Μια περίληψη του αλγορίθμου παρατίθεται στη συνέχεια:

- Κάθε διεργασία προσπαθεί να δημιουργήσει ένα DFS spanning tree με τον εαυτό της σαν ρίζα, χρησιμοποιώντας ένα διαφορετικό αντίγραφο του δένδρου.
- Αν δύο δέντρα προσπαθήσουν να συνδεθούν στον ίδιο κόμβο, ο κόμβος θα συνδεθεί με το δένδρο του οποίου η ρίζα έχει το μεγαλύτερο αναγνωριστικό (rank).
- Άρα, τελικά, θα δημιουργηθεί ένα δένδρο με ρίζα τη διεργασία με το μεγαλύτερο αναγνωριστικό όλων, η οποία θα είναι και ο αρχηγός.

Η διεργασία βιβλιοθήκης που εκλέγεται αρχηγός, θα πρέπει να ενημερώσει όλες τις άλλες διεργασίες βιβλιοθήκης ότι εκλέχτηκε αρχηγός και στη συνέχεια να αποστείλει στη διεργασία συντονιστή ένα μήνυμα τύπου **<LE_LIBR_DONE>** για να την ενημερώσει για τον τερματισμό της φάσης υπολογισμού του αρχηγού και για το ποιος είναι ο αρχηγός των βιβλιοθηκών. Ο συντονιστής αποθηκεύει τον αρχηγό αυτό.

2.4 Εκλογή Αρχηγού Μεταξύ Διεργασιών Δανειστών

Αμέσως μετά το γεγονός τύπου <**START_LE_LIBR**>, το test_file περιέχει ένα γεγονός τύπου **<START_LE_LOANERS>**. Για την επεξεργασία ενός τέτοιου γεγονότος, ο συντονιστής αποστέλλει ένα μήνυμα τύπου <START_LE_LOANERS> σε κάθε διεργασία δανειστή που υπάρχει στο σύστημα και έπειτα περιμένει να λάβει ένα μήνυμα τύπου <LE _LOANERS_DONE> από τη διεργασία που θα εκλεγεί αρχηγός. Ακολουθεί η περιγραφή του αλγορίθμου εκλογής αρχηγού για τις διεργασίες δανειστές.

Εκλογή Αρχηγού μεταξύ των διεργασιών δανειστών (Loaner Leader Election). Ο αλγόριθμος εκλογής αρχηγού που πρέπει να υλοποιήσετε είναι ο **STtoLeader** (σελίδα 6, Ενότητα 9 διαφανειών). Όταν μία διεργασία δανειστής με αναγνωριστικό client_id, λάβει ένα μήνυμα τύπου <START_LE_LOANERS> από το συντονιστή, ξεκινά τον αλγόριθμο εκλογής αρχηγού (αποστέλλοντας ένα μήνυμα τύπου <ELECT> προς κάθε έναν από τους γειτονικούς της κόμβους). Μια περίληψη του αλγορίθμου εκλογής αρχηγού παρατίθεται στη συνέχεια:

Αλγόριθμος STtoLeader

• Τα φύλλα του δένδρου (δηλαδή οι κόμβοι που έχουν ένα μόνο γειτονικό κόμβο), ξεκινούν ένα convergecast, χρησιμοποιώντας μηνύματα <ELECT>.

  – Κάθε φύλλο, αρχικά, μπορεί να στείλει ένα μήνυμα <ELECT> στον μοναδικό του γείτονα.

  – Κάθε κόμβος που έχει λάβει μηνύματα <ELECT> από όλους τους γείτονες του εκτός από έναν, μπορεί να στείλει ένα μήνυμα <ELECT> στον υπολειπόμενο γείτονα.

• Τελικά, μια από τις ακόλουθες συνθήκες θα ισχύει:

  1. Κάποια διεργασία λαμβάνει μηνύματα <ELECT> από όλους τους γείτονες της, πριν στείλει η ίδια ένα μήνυμα <ELECT>. Η διεργασία, στην οποία συγκλίνουν τα μηνύματα, εκλέγεται αρχηγός.

2. Έχουν σταλθεί μηνύματα <ELECT> και στις δύο κατευθύνσεις μίας συγκεκριμένης ακμής. Η διεργασία με το μεγαλύτερο αναγνωριστικό (rank), από τις προσκείμενες διεργασίες στην ακμή αυτή, εκλέγεται αρχηγός.

Η διεργασία δανειστής που εκλέγεται αρχηγός, θα πρέπει να ενημερώσει όλες τις άλλες διεργασίες δανειστές ότι εκλέχτηκε αρχηγός με ένα μήνυμα τύπου **<LE_LOANERS>** που περιέχει το rank της διεργασίας που είναι αρχηγός τύπος δανειστών και στη συνέχεια να αποστείλει στη διεργασία συντονιστή ένα μήνυμα τύπου **<LE_LOANERS_DONE>** για να την ενημερώσει για τον τερματισμό της φάσης υπολογισμού του αρχηγού και για το ποιος είναι ο δανειστής αρχηγός. Ο συντονιστής αποθηκεύει τον αρχηγό αυτό.

3 Γεγονότα

Ο αρχηγός των βιβλιοθηκών είναι υπεύθυνος για την αναζήτηση βιβλίων όταν μια βιβλιοθήκη δεν έχει κάποιο βιβλίο αλλά το ζητάει κάποιος δανειστής της. Για να το καταφέρει αυτό αποθηκεύει σε έναν πίνακα N^4 στοιχείων το κάθε βιβλίο σε ποια βιβλιοθήκη αντιστοιχεί. Όταν του ζητηθεί να εντοπίσει ένα βιβλίο με αναγνωριστικό b_id προσβαίνει το αντίστοιχο στοιχείο του πίνακα για να βρει την βιβλιοθήκη που του αντιστοιχεί.

1. TakeBook <c_id> <b_id>

Γεγονός που σηματοδοτεί το δανεισμό ενός βιβλίου από τη διεργασία δανειστή με αναγνωριστικό c_id. Όταν ο συντονιστής διαβάσει ένα τέτοιο γεγονός στο test_file, ενημερώνει τη διεργασία δανειστή με αναγνωριστικό c_id. Η διεργασία δανειστής c_id ζητάει να δανειστεί το βιβλίο b_id από τη βιβλιοθήκη l_id στην οποία ανήκει (το l_id θα πρέπει να υπολογιστεί με αλγοριθμικό τρόπο). Για να γίνει αυτό, η διεργασία δανειστή με αναγνωριστικό c_id στέλνει ένα μήνυμα τύπου LEND_BOOK στη διεργασία βιβλιοθήκης με αναγνωριστικό l_id. Όταν η διεργασία βιβλιοθήκης με l_id λάβει το μήνυμα, εξετάζει αν το βιβλίο είναι στην δική της συλλογή. Αν ναι στέλνει ένα μήνυμα τύπου <GET_BOOK> στον δανειστή c_id το οποίο προσομοιώνει ότι την αποστολή του βιβλίο. Αν όχι, στέλνει μήνυμα τύπου

FIND_BOOK <b_id>

στον αρχηγό των διεργασιών βιβλιοθηκών, για να βρει το αναγνωριστικό l_id` της βιβλιοθήκης στην οποία βρίσκεται το βιβλίο

Όταν ο αρχηγός των διεργασιών βιβλιοθηκών λάβει το μήνυμα, επιστρέφει το αναγνωριστικό l_id` της διεργασίας βιβλιοθήκης που έχει το βιβλίο b_id. Συγκεκριμένα υπολογίζει το l_id` κάνοντας τις κατάλληλες πράξεις.

Όταν ο διεργασία δανειστής με αναγνωριστικό l_id λάβει την απάντηση του αρχηγού, στέλνει ένα μήνυμα στη διεργασία βιβλιοθήκης με αναγνωριστικό l_id` του ακόλουθου τύπου:

BOOK_REQUEST <b_id>, <c_id>, <l_id>

Η βιβλιοθήκη με αναγνωριστικό l_id` απαντάει στη βιβλιοθήκη l_id που έκανε το αίτημα με ένα μήνυμα τύπου **ACK_TB <bid>** πως βρήκε το βιβλίο. Και η βιβλιοθήκη με αναγνωριστικό l_id απαντάει στον δανειστή με c_id με ένα μήνυμα τύπου **ACK_TB <b_id>**. Αφού η διεργασία δανειστής λάβει το μήνυμα τύπου **ACK_TB <bid>** ενημερώνει τα στατιστικά του απ’την συλλογή

των βιβλίων του, δηλαδή ότι έχει δανειστεί το βιβλίο b_id και αυξάνει τον μετρήτη που δείχνει πόσες φορές έχει δανειστεί το συγκεκριμένο βιβλίο. Το ίδιο κάνει και η διεργασία βιβλιοθήκης με αναγνωριστικό l_id. Ακόμα, η διεργασία δανειστής τυπώνει το μήνυμα “Got book {b_id} from {l_id}“ οπου l_id το αναγνωριστικό της βιβλιοθήκης απ’την οποία το δανείστηκε

Τελικά η διεργασία δανειστής στέλνει στον συντονιστή ένα μήνυμα τύπου <DONE_FIND_BOOK> για την ολοκλήρωση του γεγονότος FIND_BOOK.

2. DonateBook <c_id>, <b_id>, <n_copies>

Γεγονός που σηματοδοτεί τη δωρεά ενός βιβλίου με αναγνωριστικό b_id από την διεργασία δανειστή με αναγνωριστικό c_id, με αριθμό αντιτύπων n_copies. Όταν ο συντονιστής διαβάσει ένα τέτοιο γεγονός στο test_file ενημερώνει τη διεργασία δανειστή με αναγνωριστικό c_id με ένα μήνυμα τύπου <DONATE_BOOKS>. Η διεργασία δανειστής με αναγνωριστικό c_id δωρίζει το βιβλίο b_id. Για να γίνει αυτό, η διεργασία δανειστή με αναγνωριστικό c_id στέλνει ένα μήνυμα τύπου <DONATE_BOOK> στη διεργασία αρχηγού των δανειστών και αυτός μοιράζει όλα τα αντίγραφα(n_copies) του βιβλίου στις βιβλιοθήκες. Τα μοιράζει εκ περιτροπής (round robin) διατηρώντας μια μεταβλητή donate_next, που αρχικά έχει την τιμή 0. Κάθε φορά που δέχεται ένα request τύπου donateBook θα μοιράσει στις βιβλιοθήκες με αναγνωριστικά

donate_next % N^2 , (donate_next + 1) % N^2 ,…, (donate_next + n_copies) % N^2

Η κάθε βιβλιοθήκη που αντιστοιχεί στα παραπάνω αναγνωριστικά λαμβάνει ένα αντίγραφο του βιβλίου. Αν υπάρχουν περισσότερα αντίγραφα(n_copies) από τις βιβλιοθήκες επαναλαμβάνεται το round robin διατηρώντας και πάλι την donate_next.

Στη συνέχεια κάθε βιβλιοθήκη απαντάει με ένα request τύπου ACK_DB**,** στη διεργασία αρχηγού τύπου δανειστή για να δείξει ότι πήρε το βιβλίο b_id. Επίσης, κάθε βιβλιοθήκη ενημερώνει τα στατιστικά της δηλαδή την συλλογή της, ότι υπάρχει καινούργιο βιβλίο με αναγνωριστικό b_id, διαθέσιμο για δανεισμό και τυπώνει το μήνυμα “Received donate book {b_id} from {l_id}“ οπου l_id το αναγνωριστικό της βιβλιοθήκης.

Τελικά η διεργασία δανειστής στέλνει στον συντονιστή ένα μήνυμα τύπου <DONATE_BOOKS_DONE> για την ολοκλήρωση του γεγονότος του DONATE_BOOK.

3. GetMostPopuralBook

Γεγονός που σηματοδοτει την εύρεση του πιο δημοφιλές βιβλίου ανα όμαδα δανειστών σε κάθε βιβλιοθήκη l_id, δηλαδή του βιβλίο που έχει δανειστεί τις περισσότερες φορές από έναν δανειστή, όχι αθροιστικά οι αριθμοί δανεισμών από όλους τους δανειστές. Για κάθε βιβλιοθήκη l_id αντιστοιχεί μια συγκεκριμένη ομάδα δανειστών που έχει περιγραφεί προηγουμένως. Όταν ο συντονιστής διαβάσει ένα τέτοιο γεγονός στο test_file, ενημερώνει τη διεργασία αρχηγό τύπου δανειστή με ένα μήνυμα τύπου <GET_MOST_POPULAR_BOOK>. Όταν η διεργασία αρχηγός τύπου δανειστή λάβει το μήνυμα θα πρέπει να βρει το πιο δημοφιλές βιβλίο σε επίπεδο δανειστών. Για να βρεθεί το πιο δημοφιλές βιβλίο θα πρέπει να εκτελεστεί ένα broadcast από την διεργασία αρχηγού δανειστή μέσω του σκελετικού δένδρου των δανειστών και μετά ένα convergecast για να ενημερωθεί ο αρχηγός δανειστών για το πιο δημοφιλές βιβλίο για κάθε βιβλιοθήκη. Ψάχνουμε να βρούμε το βιβλίο που έχει δανειστεί τις περισσότερες φορές μεταξύ των δανειστών της κάθε βιβλιοθήκης. Ο κάθε δανειστής κρατάει ποια βιβλία έχει δανειστεί και πόσες φορές έχει δανειστεί το κάθε ένα. Βρίσκει το βιβλίο b_id που έχει δανειστεί τις περισσότερες φορές και στέλνει πληροφορία αυτή στον αρχηγό των δανειστών με ένα μήνυμα τύπου

<GET_POPULAR_BK_INFO> <b_id> <times_loaned> <l_id>

Το μήνυμα αυτό περιέχει το id του βιβλίου <b_id> πόσες φορές έχει δανειστεί <times_loaned> και από σε ποια βιβλιοθήκη <l_id> ανήκει αυτό το βιβλίο. Το l_id το υπολογίζετε αλγοριθμικά . Στη συνέχεια ο αρχηγός των δανειστών απαντάει με ένα request τύπου **ACK_BK_INFO** στον κάθε δανειστή. Όταν η διεργασία αρχηγού δανειστής πάρει μηνύματα από όλους τους υπόλοιπους δανειστές, βρίσκει το πιο δημοφιλές βιβλίο δηλαδή αυτό που έχει δανειστεί τις περισσότερες φορές μεταξύ των δανειστών που ανήκουν στην l_id βιβλιοθήκη. Αν υπάρχει ισοψηφία το πιο ακριβό βιβλίο βάση του πεδίου cost υπερτερεί. Η διεργασία αρχήγού τύπου δανειστή τυπώνει το πιο δημοφιλές βιβλίο ως “Popular book {b_id} for library {l_id}” για κάθε l_id.
Τελικά η διεργασία αρχηγός των δανειστών στέλνει στον συντονιστή ένα μήνυμα τύπου <GET_MOST_POPULAR_BOOK_DONE>
για την ολοκλήρωση του γεγονότος FIND_BOOK.

4. CheckNumBooksLoaned

Γεγονός για τον έλεγχο ορθότητας μεταξύ του αριθμού των δανεισθέντων βιβλίων από την μεριά των βιβλιοθηκών και του αριθμού βιβλίων που έχουν δανειστεί οι δανειστές. Όταν ο συντονιστής διαβάσει ένα τέτοιο γεγονός στο test_file, στέλνει στην διεργασία αρχηγού των βιβλιοθηκών καθώς και του αρχηγού των δανειστών ένα μήνυμα τύπου

<CHECK_NUM_BOOKS_LOAN>.

Ο αρχηγός των βιβλιοθηκών ψάχνει να βρει πόσα βιβλία έχει δανεισεί η κάθε βιβλιοθήκη. Για να το κάνει αυτό, στέλνει μήνυμα τύπου <CHECK_NUM_BOOKS_LOAN> στην βιβλιοθήκη 0 και εκείνη προωθεί το ίδιο μήνυμα στην επόμενη βιβλιοθήκη. Αυτο επαναλάμβανεται από την κάθε επόμενη βιβλιοθήκη μέχρι να λάβουν όλες οι βιβλιοθήκες το μήνυμα.

Πιο συγκεκριμένα η βιβλιοθήκη 0 θα εκκινήσει τη δρομολόγηση του μηνύματος και στις υπόλοιπες βιβλιοθήκες. Συγκεκριμένα, το μήνυμα θα πρέπει να διασχίσει το πλέγμα των βιβλιοθηκών με την ακόλουθη σειρά;

<0,0>, <1,0>, <2,0>, …, <N-1,0>, <N-1,1>,…,<2,1> , <1,1> ,<0,1>, <0,2>, <1,2>, <2,2>,..,<N-1,2>,... (δηλαδή το μήνυμα ταξιδεύει στο πλέγμα γραμμή ανά γραμμή). Κάθε κόμβος που λαμβάνει ένα μήνυμα που πρέπει να διασχίσει το πλέγμα, θα πρέπει να γνωρίζει σε ποιον κόμβο πρέπει να προωθήσει το μήνυμα αυτό.

Αυτές διατρέχουν τη λίστα με τα βιβλία και μετράνε τον συνολικό αριθμό δανεισμένων βιβλίων. Στην συνέχεια στέλνουν ένα μήνυμα στην διεργασία αρχηγός των βιβλιοθηκών τύπου

<NUM_BOOKS_LOANED> <num_books_loan>

που περιέχει πόσα βιβλία <num_books_loan> έχει δανείσει η συγκεκριμένη βιβλιοθήκη l_id. Αφού λάβει το μήνυμα η διεργασία αρχηγός των βιβλιοθηκών στέλνει ένα μήνυμα τύπου **<ACK_NBL>** στην l_id βιβλιοθήκης. Στην συνέχεια η διεργασία που είναι αρχηγός των βιβλιοθηκών αφού λάβει όλα τα μηνύματα τύπου <NUM_BOOKS_LOANED> αθροίζει τους αριθμούς δανεισμών για κάθε βιβλιοθήκη. Αφού το κάνει αυτό στέλνει στο συντονιστή ένα μήνυμα τύπου <CHECK_NUM_BOOKS_LOAN_DONE>. Με αντίστοιχο τρόπο η διεργασία αρχηγού των δανειστών κάνει τις αντίστοιχες ενέργιες από την μεριά των δανειστών. Θα πρέπει να εκτελέσει ένα broadcast μέσω του σκελετικού δέντρου των δανειστών για να βρεθεί ο συνολικός αριθμός δανεισμών όλων των βιβλίων από όλους τους δανειστές και μέτα ένα convergecast για να ενημερωθεί ο αρχηγός. Στο τέλος πρέπει ο αριθμός των δανεισθέντων φορών από την μεριά των δανειστών να ισούται με τον αριθμό δανεισμένων φορών απ’τις βιβλιοθήκες. Ακόμα, ο αρχηγός των βιβλιοθηκών και των δανειστών, εκτυπώνουν τον αριθμό των δανεισμένων βιβλίων ως “Loaner Books <b1>”, “Library Books <b2>”, ‘όπου b1 ο αριθμός των δανεισμών από τη μεριά των

δανειστών, και b2 ο αριθμός των δανεισμών από τη μεριά των βιβλιοθηκών. Αν δεν είναι ίσοι οι αριθμοί τότε τυπώνεται “CheckNumBooksLoanded FAILED”. Αλλιώς, τυπώνεται “ CheckNumBooksLoanded SUCCESS”.

4 Message Passing Interface (MPI)

Για την υλοποίηση της εργασίας θα πρέπει να χρησιμοποιήσετε τη βιβλιοθήκη Message Passing Interface (MPI). Η βιβλιοθήκη αυτή παρέχει τη δυνατότητα ανταλλαγής μηνυμάτων μεταξύ διεργασιών (που μπορεί να εκτελούνται στο ίδιο ή σε διαφορετικά μηχανήματα) μέσω ενός καθιερωμένου Application Programming Interface (API), ανεξαρτήτως γλώσσας και υλοποίησης. Για την προσομοίωση που ζητείται, κάθε κόμβος του συστήματος θα προσομοιώνεται από μία διεργασία MPI. Είναι αξιοσημείωτο ότι περισσότερες από μία διεργασίες MPI μπορεί να τρέχουν στο ίδιο μηχάνημα (παρότι προσομοιώνουν διαφορετικούς κόμβους του συστήματος). Το σύστημα MPI είναι εγκατεστημένο στα μηχανήματα του Τμήματος. Οι δύο βασικές εντολές που απαιτούνται για τη λειτουργία του είναι οι mpicc και mpirun. Η εντολή mpicc χρησιμοποιείται για την μεταγλώττιση προγραμμάτων που χρησιμοποιούν το API του MPI. Με την εντολή mpirun μπορείτε να τρέξετε το εκτελέσιμο ταυτόχρονα σε περισσότερα του ενός μηχανήματα.

Ένα παράδειγμα χρήσης παρατίθεται στη συνέχεια:

$ mpirun -np <count> --hostfile <file containing hostnames> <executable> <NUM_LIBS>

όπου η επιλογή <np> ισούται με το άθροισμα N^2 + N^3/2 + 1 και καθορίζει το συνολικό αριθμό των MPI διεργασιών που θα εκτελούν το εκτελέσιμο αρχείο που δίνεται σαν τελευταίο όρισμα στην παραπάνω γραμμή εντολών.

Στην επιλογή <hostfile> δίνεται το όνομα του αρχείου στο οποίο περιέχονται τα hostnames των μηχανημάτων όπου θα εκτελεστούν διεργασίες MPI. Κάθε διεργασία που εκκινείται με τη χρήση της εντολής mpirun έχει ένα ξεχωριστό αναγνωριστικό στο σύστημα MPI, το οποίο ονομάζεται βαθμός (rank) MPI. Επίσης, μπορεί να μάθει πόσες άλλες διεργασίες MPI τρέχουν στο σύστημα και να τους στείλει μηνύματα, χρησιμοποιώντας το rank τους σαν αναγνωριστικό. Είναι αξιοσημείωτο πως μία διεργασία μπορεί να στείλει μηνύματα μόνο σε διεργασίες που έχουν ξεκινήσει από την ίδια εντολή mpirun και είναι κατά συνέπεια μέλη του ίδιου “κόσμου” MPI. Για παραπάνω πληροφορίες μπορείτε να διαβάσετε το υλικό που βρίσκεται στον παρακάτω σύνδεσμο:

https://mpitutorial.com/tutorials

Επίσης, θα χρειαστεί να χρησιμοποιήσετε την συνάρτηση MPI_Type_create_struct, πληροφορίες μπορείτε να βρειτε στον παρακάτω σύνδεσμο.

https://www.mpich.org/static/docs/v3.1/www3/MPI_Type_create_struct.html

5 Παράδοση της Εργασίας

Η εργασία πρέπει να χτίζεται και να λειτουργεί ορθά στα μηχανήματα της σχολής.

Το παραδοτέο σας περιλαμβάνει τη συλλογή C source/header files στα οποία την υλοποιείτε, μαζί με το δικό σας Makefile για το χτίσιμό της. Επίσης, καλό θα ήταν να μοιράσετε την υλοποίηση σας σε πολλαπλά αρχεία.

Η παράδοση της εργασίας πραγματοποιείται μέσω του συστήματος elearn.