Νικόλαος Σαγεώργης ΑΜ:2349

Εργαστήριο 1:Υλοποίηση ενός απλού πολύνηματικού διακομιστή αποθήκευσης ζευγών κλειδιού-τιμής

1. Περιγραφή λειτουργίας του πολυνηματικού διακομιστή αποθήκευσης

Στο συγκεκριμμένο εργαστήριο, θέλω ο server να εξυπηρετεί τις αιτήσεις που δέχεται από τον client ταυτόχρονα τη μία σε σχέση με τις άλλες, κατά τη διάρκεια της φάσης ανάγνωσης.
Η δομή που χρησιμοποιώ στον server περιέχει τον file descriptor της σύνδεσης και τους χρόνους tv1,tv2 και tv3 αντίστοιχα για το πότε ήρθε, πότε ξεκίνησε και πότε τελείωσε η εξυπηρέτηση της αίτησης. Το μοντέλο παραγωγού-καταναλωτή, με το κύριο νήμα ως παραγωγό και ως καταναλωτές τα νήματα που δημιουργώ ώστε να εξυπηρετούν ταυτόχρονα τις αιτήσεις, επικοινωνεί μέσο μιας στοίβας αποτελούμενης από τη παραπάνω δομή που αντιστοιχεί κάθε μία σε διαφορετική αίτηση. Ως κοινόχρηστη περιοχή τη στοίβα, βάζω κατάλληλα mutex_lock kai mutex_unlock ώστε να μην προκύψει πρόβλημα από την ταυτόχρονη προσπέλαση των κοινών πόρων. Με τον tablecount ως δείκτη της στοίβας, ελέγχω αν είναι άδεια ή γεμάτη και περιμένουν αντίστοιχα με το pthread_cond_wait(&notempty,&mutex) τα νήματα καταναλωτή και pthread_cond_wait(&notfull, &mutex) το κυρίος νήμα μέχρι να ειδοποιησούν με την pthread_cond_signal () ο ένας τον άλλο. Δημιουργώ τα νήματα που θα τρέχουν τον καταναλωτή πριν ξεκινήσει ο παραγωγός να δέχεται αιτήσεις. Η συνάρτηση που θα εκτελέσουν τα νέα νήματα, τρέχει συνεχώς ώστε να εξυπηρετεί αιτήσεις, με κατάλληλη πρόσβαση (ατομική) παίρνει την αίτηση και επιστρέφει τον χρόνο που βγήκε από την στοίβα. Έπειτα εξυπηρετείται καλώντας την process_request η σύνδεση με τον αντίστοιχο file desciptor και επιστρέφει τον χρόνο που τελείωσε η εξυπηρέτηση της αίτησης. Στην onemoreRequest() υπολογίζω κατάλληλα (κοινόχρηστες μεταβλητές) τον συνολικό χρόνο αναμονής, εξυπηρέτησης και το πόσες αιτήσεις εξυπηρετήθηκαν. Κλείνω πάντα τον file descriptor που δημιουργείται από κάθε νέα αίτηση, διότι δεν θα παίρνει σήμα ότι εξυπηρετήθηκε. Ορίζω τον χειριστή σήματος με την signal() ο οποίος καλεί την sigHandler όταν λάβει το σήμα SIGTSTP(^Z). Στη συνάρτηση τυπώνω τους αντίστοιχους χρόνους και αριθμό αιτήσεων που εξυπηρετήθηκαν και μετά τερματίζει την λειτουργία του server. Όλες οι παραπάνω αναφορές έγιναν/βρίσκονται στο server.c.
Για την υλοποίηση του (v) ερωτήματος και την μόνη αλλαγή στο client.c, χρειάστηκε ότι βρίσκονταν στην main() να το βάλω σε μία συνάρτηση childcode() και στην θέση του δύο λούπες στην οποία πρώτη θα κάνει αρχικά CLIENTS (#define 30) fork και στην δεύτερη τα αντίστοιχα wait() (για να μην γίνουν ζόμπιε). Δηλαδή δημιουργώ 30 παιδία που θα τρέξουν όλα την ίδια συνάρτηση και θα στείλουν αίτηση στον server. Αυτά τις αιτήσεις ο server θα τις αντιληφθεί ως αιτήσεις απο διαφορετικούς πελάτες, δηλαδή θα έχουν διαφορετικά file descriptor.

Πειράματα

Για τον πειραματισμό της λειτουργίας δοκίμασα 2 σενάρια. Σε κάθε σενάρια υπάρχει ενα σταθερό πλήθος απο πελάτες (30 διεργασίες) που ο καθένας κάνει 10 επαναλήψεις και

στέλνει 129 (MAX_ID_STATION) αιτήσεις. Για να γίνει κατάλληλη χρονομέτρηση επιλέχθηκε να εκτελεστούν ο server και ο client σε διαφορετικά μηχανήματα του εργαστηρίου. Ο λόγος για αυτή την επιλογή είναι για να τρέχει ο server στο μηχάνημα και τα διαφορετικά νήματα να μπορούν να αξιοποιούν τους επεξεργαστές χωρίς να παίρνουν χρόνο εκτέλεσης και οι διεργασίες του client

1ο Πείραμα

Στο πρώτο πείραμα δοκίμασα να δω πως επηρεάζει το πλήθος των καταναλωτών τους χρόνους εξυπηρέτησης και αναμονής.

STACK_SIZE	4	
CLIENTS	30	
THREADS	AVG_WAITING	AVG_SERVICE
1	0.00038709460	0.0001071458
2	0.00020808940	0.0002078690
4	0.00009939800	0.0002967880
8	0.00001823880	0.0003824251

Παρατηρούμε πως όσο αυξάνει το πλήθος των νημάτων καταναλωτών μειώνεται ο χρόνος που απαιτείται για την αναμονή στην στοίβα. Ο λόγος που συμβαίνει αυτό είναι γιατί τα νήματα καταναλωτές μπορούνε να εξυπηρετούν ταυτόχρονα περισσότερους πελάτες χωρίς να περιμενουν. Απο την άλλη όμως αυξάνει ο χρόνος για την εξυπηρέτηση ενός πελάτη. Αυτό συμβαίνει γιατί τα νήματα που τρέχουν είναι περισσότερα και υπάρχουν και σημεία που σειριοποιείται η εκτέλεση εξαιτίας των κλειδαριών

2ο Πείραμα

Σε αυτό το πείραμα δοκίμασα να δω πως επηρεάζει το μεγεθος της στοίβας τους χρόνους.

THREADS	4	
CLIENTS	30	
STACK_SIZE	AVG_WAITING	AVG_SERVICE
4	0.0001128152	0.0002871389
8	0.0001055535	0.0003018228
16	0.0001420096	0.0003072449
32	0.0001671476	0.0003000210

Παρατηρούμε πως η μεταβολή του μεγέθους της στοίβας δεν επηρεάζει ιδιαίτερα τους χρόνους.