Μιχάλης Μήτσιος
Α.Μ 1115201500096

Στο συμπιεσμένο αρχείο που έστειλα εμπεριέχει εκτός από τα 2 .c και 2 .h και τα 2 data1 και data2 αρχεία τα οποία είναι τα δοθέντα και τα ονόματά τους πρέπει να είναι έτσι σε περίπτωση που θελήσετε να τα τρέξετε με κάποιο δικό σας τεστ.

Τρόπος υλοποίησης :
make
./teliko #max_anafores #q #frames #k

Ο μοναδικός περιορισμός είναι επειδή στα PM1 και PM2 δίνεται σαν όρισμα το max/2 ο αριθμός αυτός δεν θα πρέπει να είναι περιττός.

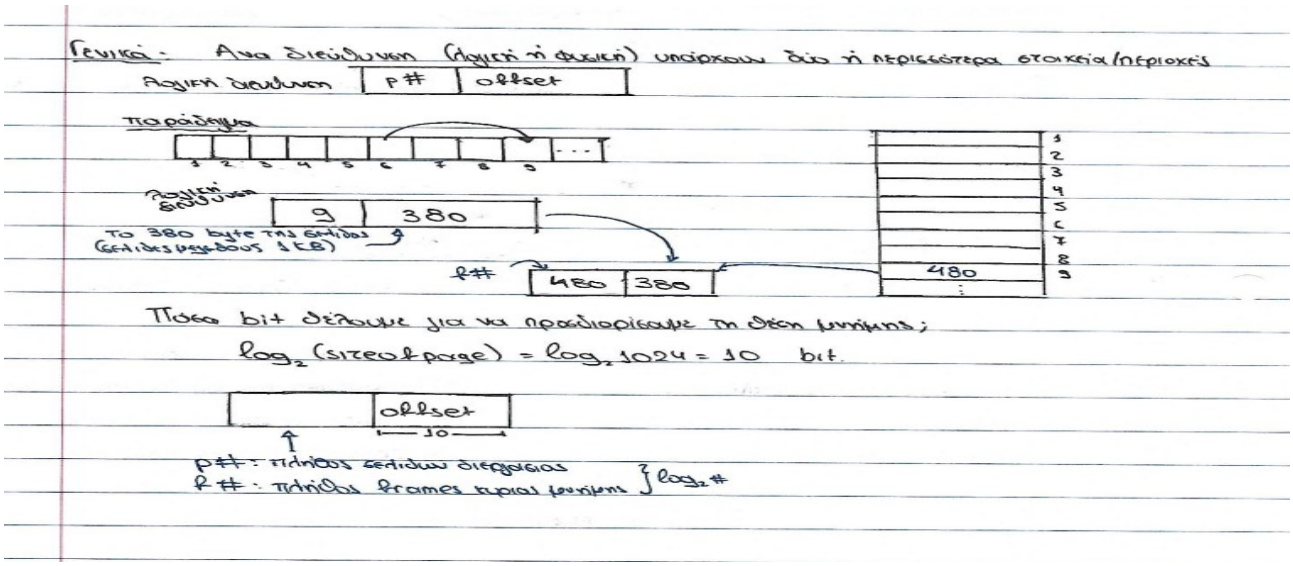
Το πρόγραμμα μου δημιουργεί 3 διεργασίες με 3 σημαφόρους και μια διαμοιραζομενη κοινή μνήμη.

Αρχικά οι 2 διεργασίες PM1 και PM2 κανουν την απλη δουλεία της ανάγνωσης του data1 και data2 αντίστοιχα μέσω μιας for η οποία γίνεται max/2 φορές.

Η επικοινωνία των 3 διεργασιών γίνεται με τους 3 σημαφόρους και την shm. Αρχικα αρχίζει η 1η η PM1 καθώς εχω κάνει τον αντίστοιχο σημαφόρο της , 0 , up για να μην μπλοκάρει ενώ η pm2 θα περιμένει για να συνεχίσει καθως ο 1 παραμένει down.

Και μετά αρχίζει η MM η οποία για την επικοινωνία χρησιμοποιεί ουσιαστικα 2 μεταβλητές την choice και την counter. Η counter για κάθε “ανάγνωση” που κάνει απο την shm αυξάνεται κατα 1 και όταν γίνει q τότε το choice θα γίνει από 0→1 και η counter θα μηδενιστεί. Έτσι για τις επόμενες q επαναλήψεις η PM2 θα γράφει κα ιη MM θα διαβάζει μέχρι να ξαναγίνει η αλλαγή.

Αφού γίνει η ανάγνωση απο την MM αυτό που κάνει είναι να παίρνει τον οχταψηφιο δεκαεξαδικό αριθμο και να το χωρίζει στα 5 πρώτα ψηφία #p και στα επόμενα 3 που είναι το offset



μονο που εφω πήρα τον log16(4096)=3=offset

Το hash έχει την μορφή ενός πίνακα από λίστες . Και η διαδικασία είναι η εξής:
Αφού πάρει τον #p τον μετατρέπει σε δεκαδικό και βρίσκει το υπόλοιπό του με το μισό μέγεθος του πίνακα. Εαν ανήκει στην 1η διεργασά θα πάει στο 1ο μισό του πίνακα και εαν ανήκει στην 2η διεργασία κατατάσσεται στο 2ο μισό του πίνακα. Η θέση της λίστας στον πίνακα που θα προσθεθεί

το στοιχείο θα είναι το υπόλοιπο της προηγουμενης διαίρεσης εαν ανήκει στο 1ο αλλίως το υπόλοιπο της διαίρεσης ΣΥΝ τον μισό πλήθος του πίνακα. Αφού βρούμε την θέση που ανήκει το στοιχείο το επόμενο βήμα είναι να δούμε εαν το στοιχείο υπάρχει ήδη. Εαν δεν υπάρχει τότε αυξάνονται τα page fault ΤΗΣ PM1 ΜΟΝΟ και τοποθετείται η σελίδα. Καταλαμβάνοντας ένα στρακτ με τα αντίστοιχα στοιχεία. Εάν υπάρχει και εάν το dirty bit είναι 0 και εάν είναι τύπου w τότε κάνει το dirty bit το μετατρέπει σε 1 για να δείξει ότι ξαναγράψαμε στο περιεχόμενο της. Αντίστοιχα θα συμβεί και με την PM2 .

Επίσης όσων αναφορά τα page fault είναι ξεχωριστός counter για την 1η και ξεχωριστός για την 2η διεργασία γτ ο έλεγχος της κάθε μιας πρέπει να είναι ανεξάρτητος και όταν κάποιος από τους 2 φτάσει το k+1 θα γίνει ο FWF ΓΙΑ ΤΟ ΑΝΤΙΣΤΟΙΧΟ ΜΙΣΟ ΠΙΝΑΚΑ.

Αντίθετα τα frames μετριούνται συνολικά καθώς πρέπει συνολικά να βλέπουμε πόσο χώρο έχουμε. Όταν γεμίσουν τα frames χωρις να έχει ικανοποιηθεί η συνθήκη του FWF για κάποια από τα 2 μισά τότε γίνεται FWF όλος ο πίνακας ώστε να ελευθερωθούν όλα τα framesώστε μετά να ξαναγεμίσουν.

Τελικά στατιστικά :
Αρχικά χρησιμοποιώ 2 μεταβλητές total_W και total_R για να μετρήσω πόσα συνολικά writes και reads εχω απο τον δίσκο. Το writes αυξάνεται όταν έχω page fault και η εγγραφή είνα ιτύπου W αλλά και όταν έχει γίνει κάποια αλλαγή και θα χρειαστεί να ξαναγραφτεί στον δίσκο δλδ όταν το dirty bit γίνεται 1 από 0.

Αντίστοιχα το total_R αυξάνεται όταν η εγγραφή είναι R και έχουμε page fault γτ εαν δεν είχαμε page fault σημαίνει ότι έεχουμε την εγγραφή ήδη στην εικονική μνήμη οπότεκ αι δεν χρειάζεται να την ξαναδιαβάσουμε από τον δίσκο.

Τα page fault αυξάνοτναι όταν η εγγραφή δεν υπάρχει στο hash table και χρειάζεται να την γράψουμε.(total_pag_fault)

Επίσης για τις καταχωρήσεις που εξετάστηκαν έχω μια 2η λίστα στην οποία εισάγωνται όλες αυτές οι καταχωρήσεις και εκτυπώνονται στην σειρά 465 σε περίπτωση που δεν επιθυμείτε την εκτύπωση.

Τέλος υπάρχει το fr1count+fr2count όπου αυτοί περιγράφουν το πόσα frames χρησιμοποιούνται από την 1η και 2η διεργασία αντίστοιχα.

Από τις διάφορες τιμές του κ παρατηρό ότι όσο μεγαλύτερο κ (ενώ τα υπόλοιπα κρατιούντια σταθερα και το κ είναι μικρότερο του frames) τα page fault μειώνονται το οποίο είναι και λογικό.

Για max_num=5000 q=10 frames=256

k=100
TOTAL WRITES TO DISK=2382
TOTAL READS FROM DISK=2460
TOTAL PAGE FAULT=4842
CURRNET FRAMES USED=124

k=20
TOTAL WRITES TO DISK=2387
TOTAL READS FROM DISK=2486
TOTAL PAGE FAULT=4873
CURRNET FRAMES USED=50

k=10
TOTAL WRITES TO DISK=2384
TOTAL READS FROM DISK=2475
TOTAL PAGE FAULT=4859
CURRNET FRAMES USED=93

k=200
TOTAL WRITES TO DISK=2378
TOTAL READS FROM DISK=2457
TOTAL PAGE FAULT=4835
CURRNET FRAMES USED=136

Εν ολίγης έχουμε μεγαλύτερο όριο για την εικονική μνήμη επομένως περισσότερες σελίδες μπορούν να μένουν ανοιχτές και όταν τις ξαναζητήσουμε θα υπάρχουν ήδη οπότε δεν θα χρειαστεί να τις ξαναφορτώσουμε στο hash άρα δεν θα χρειαστεί να αυξήσουμε τα page fault.