**Ομάδα:** Μάριος Γαβριήλ 4244,Μηνάς Ελευθερίου 4245

1η Εργαστηριακή άσκηση

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Για την δημιουργία τον πολλαπλών νημάτων υλοποιήσαμε δύο for loops δημιουργώντας και κλείνοντας αντίστοιχα τα νήματα από την γραμμή 128-131 για τους γραφείς και 147-150 για τουυς αναγνώστες . Για να καλέσουμε τα write\_test/read\_test μέσα από τα νήματα δημιουργήσαμε τις write\_test\_func / read\_test\_func και περάσαμε σαν παραμέτρους σε αυτά το r και τις τιμές που θα εγγράψει ή θα διαβάσει το κάθε νήμα μέσα από ένα struct input\_data το οποίο δηλώσαμε μέσα στο bench.h .Για την αποφυγή του άσκοπου ανοίγματος και κλεισίματος της βάσης δεδομένων μας την ανοίγουμε (125 και 144 ) και την κλείνουμε(132 και 151) σε κάθε κάλεσμα των write/read/readwrite. Μετά το τέλος της κάθε μιας τυπώνουμε τα αποτελέσματα του χρόνου που χρειάστηκε για να εκτελεστεί καλώντας μια νέα συνάρτηση printResults.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Για την καινούργια λειτουργία του συστήματος ορίσαμε δύο καινούργιες παραμέτρους για το ποσοστό των νημάτων που θα υλοποιούν γραφείς και αναγνώστες(169-170) υπολογίζουμε την αναλογία τους σε τιμές(171-172) και υπολογίζουμε τις τιμές που θα υλοποιήσει το κάθε νήμα για του αναγνώστες και τους γραφείς(173-174) περνάμε τις ανάλογες παραμέτρους και δημιουργούμε τα νήματα μέσα από for loops (186-189) .Δημιουργούμε τα νήματα αναγνώστες πριν να κάνουμε join τους γραφείς έτσι ώστε να δημιουργούνται και αναγνώστες και γραφείς ταυτόχρονα χωρίς να περιμένουν οι αναγνώστες τους γραφείς να ολοκληρώσουν . Κάνουμε join και τους αναγνώστες(190-191) και τους γραφείς(192-193) για να ολοκληρωθούν σίγουρα πριν ολοκληρωθεί η main. Στο τέλος καλείται η συνάρτηση printResults για τύπωμα των στατιστικών.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Στο kiwi.c δημιουργήσαμε δύο νέες συναρτήσεις open\_db/close\_db (12-17),όπου γίνεται το άνοιγμα και το κλείσιμο της βάσης αντίστοιχα , για να μπορέσουμε να καλέσουμε τις λειτουργίες αυτές μέσα από την bench.c . Καλούμε την write/read\_test αντίστοιχα και παίρνουμε το κόστος της υλοποίησης του κάθε test καλώντας την συνάρτηση calculate\_cost(18) και δημιουργώντας δύο mutex calculate\_write\_time/calculate\_read\_time(9-10) μέσα από τα οποία κλειδώνουμε το κάθε ένα ώστε μόνο ένα νήμα να προσθέτει στο συνολικό κόστος των read/write\_test το κόστος εκτέλεσης του κάθε νήματος παίρνοντας έτσι το συνολικό κόστος .

Για την υλοποίηση του 2ου βήματος ώστε να έχουμε πολλαπλούς αναγνώστες και ένα γραφέα αποφασίσαμε να υλοποιήσουμε τον αλγόριθμο αναγνωστών-γραφέων με προτεραιότητα στους αναγνώστες για μια πιο γρήγορη υλοποίηση απ’ότι θα είχαμε αν υλοποιούσαμε προτεραιότητα στους γραφείς. Γι’αυτό υλοποιήσαμε δύο mutex readwrite\_mutex/readcount\_mutex για έλεγχο τον γραφέων και αναγνωστών.

Εικόνα που περιέχει κείμενο, οθόνη, εσωτερικό, στιγμιότυπο οθόνης

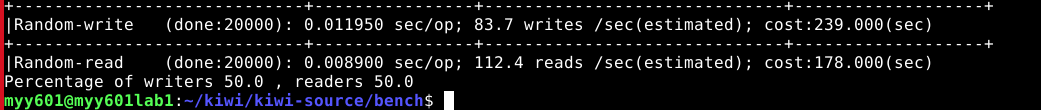
Περιγραφή που δημιουργήθηκε αυτόματα

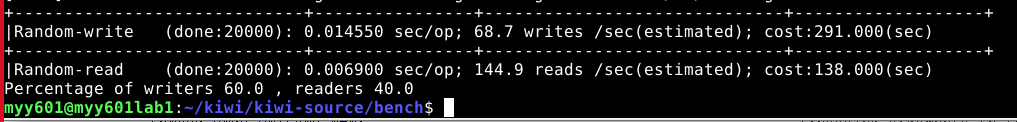
Ορίζοντας όλο τον κώδικα του db\_add (51-58) ως κρίσιμη περιοχή επιτυγχάνουμε να έχουμε μόνο ένα γραφέα κάθε φορά μέσα στο σύστημα μας. Λόγο του ότι το if επέστρεφε έναν ακέραιο αριθμό που χρησιμοποιείται αντίστοιχα στο write\_test και για να αποφύγουμε το error του να γίνεται return πριν το readwrite\_mutex να ξεκλειδώσει το περνάμε σε μια μεταβλητή και επιστρέφουμε την μεταβλητή αυτή μετά το unlock.

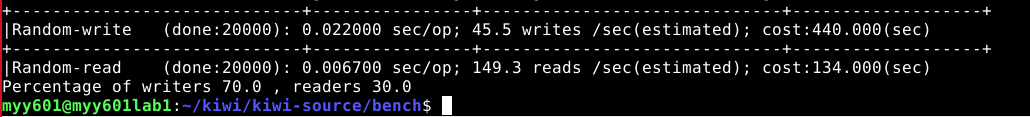
Εικόνα που περιέχει κείμενο

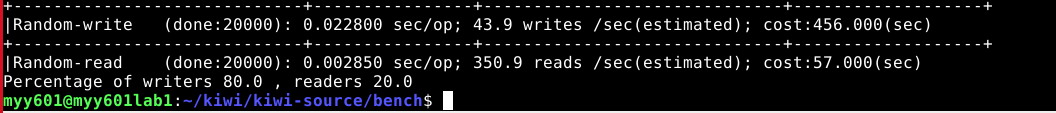
Περιγραφή που δημιουργήθηκε αυτόματα

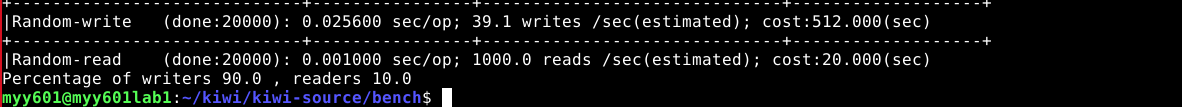
Αντίστοιχα για το db\_get(64) κλειδώσαμε την μεταβλητή readcount που είχαμε αρχικοποίηση στην αρχή του αρχείου db.c . Αυξάνοντας την κατά ένα και εφαρμόζοντας μια συνθήκη if αν το readcount ισούται με ένα τότε κλειδώνουμε το readwrite\_mutex εξασφαλίζοντας έτσι πως όσο θα υπάρχουν συνεχόμενοι αναγνώστες δεν μπορεί να υπάρξει γραφέας (66-72). Συνεχίζεται η εκτέλεση των συνεχόμενων αναγνωστών και εφόσον τελειώσουν οι αναγνώστες το readcount είναι ίσο με μηδέν τότε το readwrite\_mutex ξεκλειδώνει και συνεχίζει η κανονική εκτέλεση του συστήματος(78-82) . Όπως και στο db\_add για να αποφύγουμε return πριν από το unlock οι ακέραιοι αριθμοί που επέστρεφε το db\_get αποθηκεύονται σε μια μεταβλητή return\_of\_get και επιστρέφονται μετά το unlock.

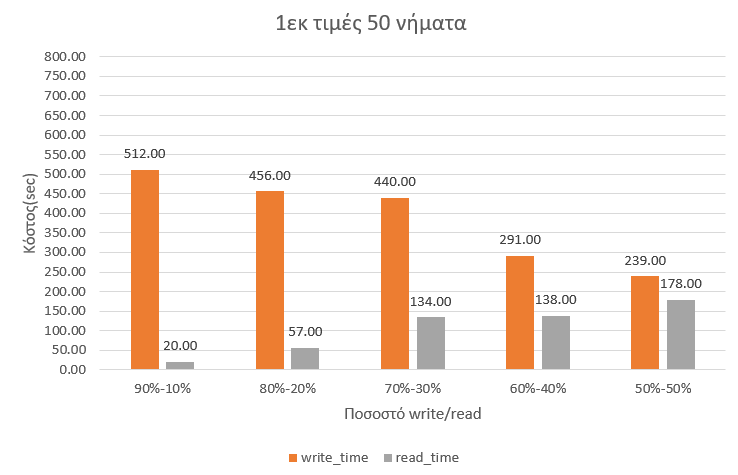












Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο

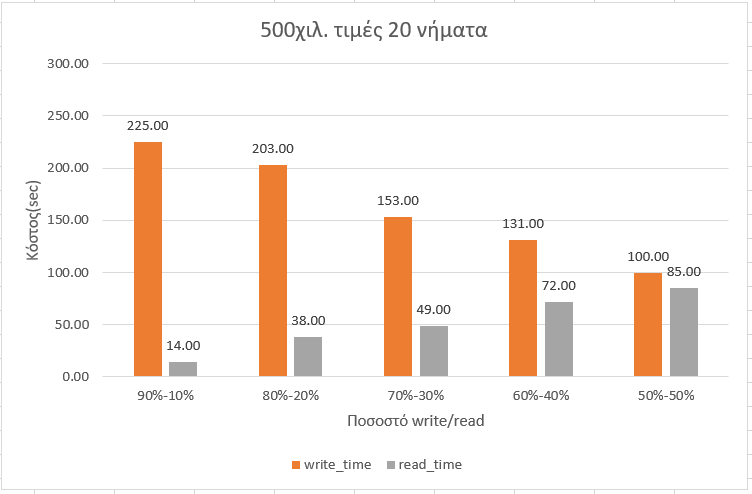
Περιγραφή που δημιουργήθηκε αυτόματα

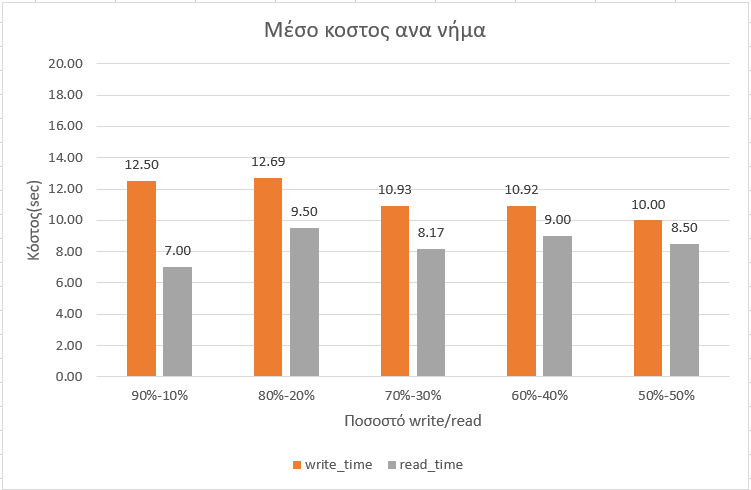
Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα





Από την πιο πάνω γραφική παράσταση παρατηρούμε ότι όσο μειώνετε το ποσοστό τον νημάτων που εκτελούν γραφείς μειώνετε σημαντικά ο χρόνος εκτέλεσης και συγκεκριμένα το κόστος εκτέλεσης των γραφέων μιας και μπορούμε να έχουμε μόνο ένα γραφέα κάθε φορά στο σύστημα μας .Αντίστοιχα το κόστος εκτέλεσης των αναγνωστών είναι σημαντικά μικρότερο σε σχέση με τους γραφείς. Επίσης παρατηρείται ότι όσο αυξάνεται το ποσοστό των γραφέων εκτελούνται λιγότερες εγγραφές ανα δευτερόλεπτο και αυξάνεται ο χρόνος που χρειάζεται για να υλοποιηθεί η κάθε εγγραφή. Ενώ οσο το ποσοστό τον αναγνωστών μειώνεται, μειώνεται σημαντικά και ο χρόνος που χρειάζεται να υλοποιηθεί η κάθε ανάγνωση , λόγο του ότι λιγότερα νήματα υλοποιούν τις αναγνώσεις, υπάρχουν περισσότερες συνεχόμενες αναγνώσεις και έχουμε περισσότερες αναγνώσεις ανα δευτερόλεπτο. Παρατηρείτε επίσης πως η μέση τιμή του κόστους ανά νήμα κυμαίνονται περίπου στις ίδιες τιμές για τους αναγνώστες μέσο όρο 8.434 s και για τους γραφείς 11.408 s.

Παρατήρηση!

Παρόλο που τα αποτελέσματα που πήραμε φαίνονται σωστά διαπιστώσαμε πως το κόστος που μας παρουσιάζει το σύστημα μας φαίνεται περίεργο.