

# Λειτουργικά συστήματα

## Εργασία 1η

Γεωργιάδης Γεώργιος , A.M. : 3199

Κουτρούδης Ιωάννης , A.M. : 3258

Η γενική ιδέα της άσκησης από ότι καταλάβαμε είναι η εξής:

Αρχικά πρέπει να υλοποιήσουμε την διαδικασία χρησιμοποιώντας πολυνηματισμό και αφού το κάνουμε αυτό θα προσθέσουμε μια ακόμα δυνατότητα στην οποία θα μπορούμε να διαβάζουμε και να γράφουμε ταυτόχρονα όπου αυτό είναι εφικτό.(writer-readers)

### Ερώτημα α)

Όλα ξεκινάνε από το bench.c αρχείο όπου εκεί μέσα υπάρχει η βασική main και ανάλογα με το τί ορίσματα θα δώσει ο χρήστης κάνει την ανάλογη διαδικασία.Πιο συγκεκριμένα αν δώσει ως όρισμα το write και έναν αριθμό θα εκτελεστεί το ανάλογο if-statement.Δηλαδή εκεί μέσα θα εκτελεστούν πρώτα βασικές συναρτήσεις στις οποίες τυπώνονται κάποια βασικά χαρακτηριστικά όπως πχ.(ημερομηνία,πληροφορίες του επεξεργαστή κλπ.).Στη συνέχεια θα καλεστεί η συνάρτηση \_write\_test η οποία υλοποιείται στο αρχείο kiwi.c.Μέσα στη συνάρτηση αυτή καλείται αρχικά η συνάρτηση db\_open η οποία ανοίγει το database και στην πορεία σύμφωνα με το όρισμα αριθμό(count ) που έχει δώσει ο χρήστης εκτελείται η λειτουργία db\_add τόσες φορές όσο είναι το count και τέλος κλείνει την βάση και τυπώνει το κόστος(δηλαδή τον χρόνο που έκανε να εκτελεστεί).

Μέσα στην συνάρτηση db\_add αρχικά τσεκάρει αν το memtable είναι γεμάτο και αν όντως είναι κάνει merge τα στοιχεία του memtable στον sst και κάνει reset το memtable.Στη συνέχεια καλεί την συνάρτηση memtable\_add.Σε αντίθετη περίπτωση αν το memtable δεν είναι γεμάτο καλεί κατευθείαν την memtable\_add η οποία προσθέτει μία καινούργια εγγραφή στο memtable.

Στην περίπτωση που ο χρήστης δώσει ως όρισμα το read και έναν αριθμό θα εκτελεστεί και πάλι το ανάλογο if-statement και οι βασικές συναρτήσεις που εκτελέστηκαν και στην περίπτωση του write. Στην συνέχεια θα καλεστεί η συνάρτηση `_read_test` η οποία υλοποιείται εξίσου στο `kiwi.c`. Μέσα στη συνάρτηση αυτή καλείται αρχικά η συνάρτηση `db_open` η οποία ανοίγει το database και στην πορεία σύμφωνα με το όρισμα αριθμό(count ) που έχει δώσει ο χρήστης εκτελείται η λειτουργία `db_get`, η οποία επιστρέφει τον αριθμό 1 σε περίπτωση που έχει βρει την εγγραφή και αν όντως την έχει βρει αυξάνει έναν counter κατά 1 ο οποίος counter δηλώνει το πλήθος των εγγραφών που βρέθηκαν στην βάση ενώ σε αντίθετη περίπτωση επιστρέφει ένα μήνυμα ότι δεν βρέθηκαν τα κλειδιά, τόσες φορές όσο είναι το count και τέλος κλείνει την βάση και τυπώνει το κόστος (δηλαδή τον χρόνο που έκανε να εκτελεστεί).

Μέσα στην συνάρτηση `db_get` τσεκάρει αρχικά αν η εγγραφή υπάρχει μέσα στο memtable καλώντας την συνάρτηση `memtable_get` και αν ναι επιστρέφει έναν 1 ενώ αν δεν υπάρχει μέσα εκεί τσεκάρει μέσα στον sst καλώντας την `sst_get`.

Όσον αφορά την εκτέλεση του κώδικα αρχικά εκτελέσαμε την εντολή `make all` μέσα στον φάκελο `kiwi-source` ώστε να γίνει ολικό compile και να δημιουργηθεί το εκτελέσιμο το οποίο είναι το `kiwi-bench` που βρίσκεται μέσα στον φάκελο `bench`. Επίσης παρατηρήσαμε ότι μέσα στον φάκελο αυτό το αρχείο κεφαλίδας `bench.h` γίνεται `include` σε περισσότερα από ένα αρχεία.c και επομένως για εξοικονόμηση χρόνου βάλαμε μια συνθήκη `ifndef ... endif` ώστε να εκτελεστεί μια φορά το αρχείο κεφαλίδας και όχι για κάθε αρχείο που γίνεται `include` σε αυτό.

Αρχικά θελήσαμε να κατανοήσουμε τον τρόπο με τον οποίο παράγονται τα κλειδιά όταν κάνουμε `write` και αντίστοιχα τον τρόπο που διαβάζονται τα κλειδιά εκτελώντας `read`. Για αυτό αρχικά εκτελέσαμε την εντολή `./kiwi-bench write 100000` και στην συνέχεια την εντολή `./kiwi-bench read 100000` και είχαμε τα εξής αποτελέσματα:

Για το write:

```

largest: key-9999
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 2 [ 1 files, 286 KiB ]---
[1466] 20 Mar 17:33:55.115 . sst.c:60 Metadata filenum:0 smallest: key-0 large
st: key-999
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . log.c:46 Removing old log file testdb/si/23.log
[1466] 20 Mar 17:33:55.115 . sst.c:170 Merge successfully completed. Releasing
the skiplist
[1466] 20 Mar 17:33:55.115 . skiplist.c:57 SkipList refcount is at 0. Freeing
up the structure
[1466] 20 Mar 17:33:55.115 . sst.c:422 Waiting the merger thread
[1466] 20 Mar 17:33:55.115 - sst.c:176 Exiting from the merge thread as user r
equested
[1466] 20 Mar 17:33:55.115 - file.c:170 Truncating file testdb/si/manifest to
332 bytes
[1466] 20 Mar 17:33:55.118 . log.c:46 Removing old log file testdb/si/24.log
+-----+-----+-----+-----+-----+-----+
|Random-Write (done:100000): 0.000020 sec/op; 50000.0 writes/sec(estimated);
cost:2.000(sec);
myy601@myy601lab1:~/kiwi/kiwi-source/bench$

```

Για το read:

```

99992 searching key-99992
99993 searching key-99993
99994 searching key-99994
99995 searching key-99995
99996 searching key-99996
99997 searching key-99997
99998 searching key-99998
99999 searching key-99999
[1469] 20 Mar 17:36:27.142 . db.c:31 Closing database 0
[1469] 20 Mar 17:36:27.142 . sst.c:415 Sending termination message to the deta
ched thread
[1469] 20 Mar 17:36:27.142 - sst.c:176 Exiting from the merge thread as user r
equested
[1469] 20 Mar 17:36:27.142 . sst.c:422 Waiting the merger thread
[1469] 20 Mar 17:36:27.142 - file.c:170 Truncating file testdb/si/manifest to
81 bytes
[1469] 20 Mar 17:36:27.150 . log.c:46 Removing old log file testdb/si/0.log
[1469] 20 Mar 17:36:27.150 . skiplist.c:57 SkipList refcount is at 0. Freeing
up the structure
+-----+-----+-----+-----+-----+-----+
|Random-Read (done:100000, found:100000): 0.000020 sec/op; 50000.0 reads /s
ec(estimated); cost:2.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$

```

Έτσι καταλάβαμε ότι τα κλειδιά στο write παράγονται σειριακά και όχι τυχαία ανάλογα με την for loop στην `_write_test` και `_read_test` μέσα στο `kiwi.c`.

Στη συνέχεια εκτελέσαμε και πάλι την εντολή `./kiwi-bench write 100000` χωρίς να κάνουμε `make clean` για να καθαρίσουμε το database μας και επίσης την `./kiwi-bench read 200000`

```

largest: key-9999
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 2 [ 1 files, 286 KiB ]---
[1466] 20 Mar 17:33:55.115 . sst.c:60 Metadata filenum:0 smallest: key-0 large
st: key-999
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[1466] 20 Mar 17:33:55.115 . log.c:46 Removing old log file testdb/si/23.log
[1466] 20 Mar 17:33:55.115 . sst.c:170 Merge successfully completed. Releasing
the skiplist
[1466] 20 Mar 17:33:55.115 . skiplist.c:57 SkipList refcount is at 0. Freeing
up the structure
[1466] 20 Mar 17:33:55.115 . sst.c:422 Waiting the merger thread
[1466] 20 Mar 17:33:55.115 - sst.c:176 Exiting from the merge thread as user r
equested
[1466] 20 Mar 17:33:55.115 - file.c:170 Truncating file testdb/si/manifest to
332 bytes
[1466] 20 Mar 17:33:55.118 . log.c:46 Removing old log file testdb/si/24.log
+-----+
+-----+
|Random-Write (done:100000): 0.000020 sec/op; 50000.0 writes/sec(estimated);
cost:2.000(sec);
myy601@myy601lab1:~/kiwi/kiwi-source/bench$

```

```

199996 searching key-199996
[1473] 20 Mar 17:38:07.510 . kiwi.c:93 not found key#key-199996
199997 searching key-199997
[1473] 20 Mar 17:38:07.510 . kiwi.c:93 not found key#key-199997
199998 searching key-199998
[1473] 20 Mar 17:38:07.510 . kiwi.c:93 not found key#key-199998
199999 searching key-199999
[1473] 20 Mar 17:38:07.510 . kiwi.c:93 not found key#key-199999
[1473] 20 Mar 17:38:07.510 . db.c:31 Closing database 0
[1473] 20 Mar 17:38:07.510 . sst.c:415 Sending termination message to the deta
ched thread
[1473] 20 Mar 17:38:07.510 - sst.c:176 Exiting from the merge thread as user r
equested
[1473] 20 Mar 17:38:07.510 . sst.c:422 Waiting the merger thread
[1473] 20 Mar 17:38:07.510 - file.c:170 Truncating file testdb/si/manifest to
81 bytes
[1473] 20 Mar 17:38:07.517 . log.c:46 Removing old log file testdb/si/0.log
[1473] 20 Mar 17:38:07.517 . skiplist.c:57 SkipList refcount is at 0. Freeing
up the structure
+-----+
+-----+
|Random-Read (done:200000, found:100000): 0.000020 sec/op; 50000.0 reads /s
ec(estimated); cost:4.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$

```

Και αυτό που παρατηρήσαμε είναι ότι βρέθηκαν τα 100000 κλειδιά άρα καταλάβαμε ότι κάθε φορά που εκτελούμε την εντολή write ξεκινάει να γραφεί από το κλειδί 0 και όχι από το τελευταίο κλειδί που είχε δημιουργηθεί από την προηγούμενη write εντολή.

## Ερώτημα β)

- bench.h:

Αρχικά κάναμε τις απαραίτητες δηλώσεις στο αρχείο bench.h:

Δηλώσαμε την στατική μεταβλητή THREADSNUMBER η οποία καθορίζει τον αριθμό νημάτων που θέλει ο χρήστης να χρησιμοποιήσει στο πρόγραμμα.Έπειτα δημιουργήσαμε μια struct στην οποία δηλώσαμε όλα τα ορίσματα που θα στείλει το κάθε νήμα και τέλος δηλώσαμε κάποιες κλειδαριές που χρησιμοποιήσαμε στα αρχεία kiwi.c και bench.c.

- bench.c:

Αρχικά μέσα στην main δηλώσαμε κάποιες μεταβλητές που θα χρησιμοποιήσουμε.Την id η οποία είναι τύπου pthread\_t και δηλώνει τον αριθμό νημάτων που θα χρησιμοποιήσουμε ,βάσει του THREADNUMBER που θα δώσει ο χρήστης.Επίσης αρχικοποιούμε την argTh η οποία είναι τύπου struct σύμφωνα με την struct που δηλώσαμε στο αρχείο bench.h.Στη συνέχεια στο if-statement της write περνάμε τις μεταβλητές count , r και untilNumber στην struct που έχουμε δηλώσει και αρχικοποιούμε μια μεταβλητή startNumTh την οποία χρησιμοποιούμε για να καθορίσουμε τον αριθμό κλειδιού από τον οποίο θα ξεκινήσει να παράγει καινούργια κλειδιά και μια μεταβλητή untilNumber η οποία καθορίζει τα πόσα κλειδιά θα επεξεργαστεί το κάθε νήμα(π.χ. αν το count είναι 1000 και τα νήματα είναι 10 τότε το κάθε νήμα θα επεξεργαστεί 1000/10 keys ).Μετά από αυτό ξεκινάμε την δημιουργία των νημάτων όπου κάθε φορά σε κάθε επανάληψη κάνουμε ένα lock(threadArgNumberSafe) ώστε να περαστούν σωστά τα ορίσματα σε κάθε νήμα και στην συνέχεια αυξάνουμε την μεταβλητή startNumTh ανάλογα με τον αριθμό κλειδιού που θέλουμε να ξεκινήσει το κάθε νήμα και περνάμε την μεταβλητή αυτή στην struct και δημιουργούμε τα νήματα βάζοντας σαν όρισμα των συναρτήσεων την struct.Τέλος καλούμε την pthread\_join για κάθε νήμα ώστε να περιμένουμε να τερματίσει.

Ακριβώς την ίδια διαδικασία εκτελούμε και στο if-statement της read με την διαφορά ότι το κάθε νήμα καλεί την `_read_test` αντί της `_write_test` που καλείται παραπάνω.

- `kiwi.c`:

Αρχικά δηλώνουμε κάποιες global μεταβλητές:

Χρησιμοποιούμε δύο μεταβλητές για να τσεκάρουμε το άνοιγμα και το κλείσιμο της database(`checkOpenDb` και `checkCloseDb`).Επίσης κάναμε global μεταβλητή την `found` η οποία μετράει τον συνολικό αριθμό των κλειδιών που βρέθηκαν,το ίδιο κάναμε για τις `start,end` για να μετρήσουμε το συνολικό κόστος όλων των νημάτων μαζί και τέλος την `db` την οποία επίσης κάναμε καθολική.

#### 1. `_write_test`:

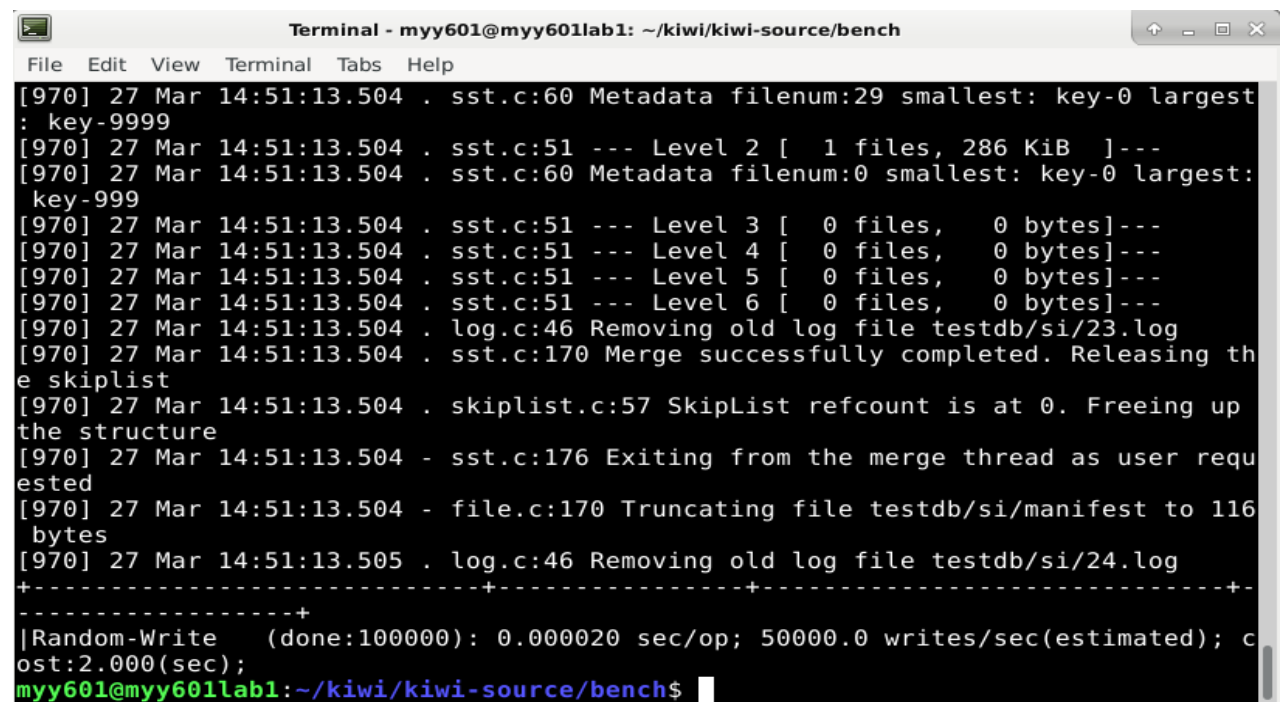
Αρχικά αρχικοποιούμε μια struct που περιέχει τα ορίσματα της `_write_test` ,επίσης τις μεταβλητές `count` , `r` ,`startNumTh` και `untilNumber` οι οποίες παίρνουν τιμές σύμφωνα με τα ορίσματα που είχε η struct.Στην συνέχεια χρησιμοποιούμε ένα if-statement στο οποίο μπαίνει μέσα μόνο το πρώτο νήμα και ανοίγει το database και ξεκινάει τον χρόνο.Αμέσως μετά την if ξεκλειδώνουμε την κλειδαριά `threadArgNumberSafe`(αφού πλέον έχουμε τελειώσει με το πέρασμα των ορισμάτων) που είχαμε βάλει μέσα στο αρχείο `bench.c` κατά την δημιουργία των νημάτων.Έπειτα ξεκινάμε,διατρέχουμε την for loop ανάλογα με την μεταβλητή `startNumTh` και `untilNumber` τις οποίες έχουμε πάρει ως όρισμα από την `bench.c`.(πχ αν έχουμε 2 νήματα το 1<sup>ο</sup> θα τρέξει από 0 έως  $\chi/2$  και το 2<sup>ο</sup> από  $\chi/2$  έως  $\chi$ ).Επίσης μέσα στην for προστατεύουμε την δομή της `db_add` κλειδώνοντας πριν καλεστεί και ξεκλειδώνοντας αφότου καλεστεί την `putSafe` κλειδαριά.Τέλος για να προστατέψουμε την `checkCloseDb` χρησιμοποιούμε μια κλειδαριά `numberSafe` και στο εσωτερικό της κλειδαριάς χρησιμοποιούμε ένα if-statement ώστε το τελευταίο νήμα που θα ολοκληρωθεί θα κλείσει την database και θα τυπώσει το ολικό κόστος.

## 2. `_read_test`:

Εκτελούμε σχεδόν την ίδια διαδικασία με την διαφορά ότι εδώ δεν βάζουμε κάποια κλειδαριά στην `db_get` γιατί δεν χρειάζεται να προστατευτεί αλλά βάζουμε μια κλειδαριά(`GlobalGetSafe`) στο `if-else-statement` όπου αυξάνεται η `global` μεταβλητή `found` και χρειάζεται να την προστατέψουμε.

Παρακάτω έχουμε μερικά τεστ που επιχειρήσαμε για να τεστάρουμε αν λειτουργεί σωστά η διαδικασία που υλοποιήσαμε:

Εκτέλεση της `write` για `count 100000` με 1 νήμα:



```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
[970] 27 Mar 14:51:13.504 . sst.c:60 Metadata filenum:29 smallest: key-0 largest: key-9999
[970] 27 Mar 14:51:13.504 . sst.c:51 --- Level 2 [ 1 files, 286 KiB ]---
[970] 27 Mar 14:51:13.504 . sst.c:60 Metadata filenum:0 smallest: key-0 largest: key-999
[970] 27 Mar 14:51:13.504 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[970] 27 Mar 14:51:13.504 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[970] 27 Mar 14:51:13.504 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[970] 27 Mar 14:51:13.504 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[970] 27 Mar 14:51:13.504 . log.c:46 Removing old log file testdb/si/23.log
[970] 27 Mar 14:51:13.504 . sst.c:170 Merge successfully completed. Releasing the skiplist
[970] 27 Mar 14:51:13.504 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
[970] 27 Mar 14:51:13.504 - sst.c:176 Exiting from the merge thread as user requested
[970] 27 Mar 14:51:13.504 - file.c:170 Truncating file testdb/si/manifest to 116 bytes
[970] 27 Mar 14:51:13.505 . log.c:46 Removing old log file testdb/si/24.log
+-----+
+-----+
|Random-Write (done:100000): 0.000020 sec/op; 50000.0 writes/sec(estimated); cost:2.000(sec);
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

Εκτέλεση της `write` για `count 100000` με 10 νήματα και αυτό που παρατηρήσαμε είναι ότι μειώθηκε ο χρόνος:

```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
[991] 27 Mar 14:53:16.876 . sst.c:60 Metadata filenum:19 smallest: key-1000 largest: key-999
[991] 27 Mar 14:53:16.876 . sst.c:51 --- Level 2 [ 1 files, 287 KiB ]---
[991] 27 Mar 14:53:16.876 . sst.c:60 Metadata filenum:0 smallest: key-0 largest: key-99
[991] 27 Mar 14:53:16.876 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[991] 27 Mar 14:53:16.876 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[991] 27 Mar 14:53:16.876 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[991] 27 Mar 14:53:16.876 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[991] 27 Mar 14:53:16.876 . log.c:46 Removing old log file testdb/si/23.log
[991] 27 Mar 14:53:16.876 . sst.c:170 Merge successfully completed. Releasing the skiplist
[991] 27 Mar 14:53:16.876 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
[991] 27 Mar 14:53:16.876 - sst.c:176 Exiting from the merge thread as user requested
[991] 27 Mar 14:53:16.876 - file.c:170 Truncating file testdb/si/manifest to 332 bytes
[991] 27 Mar 14:53:16.877 . log.c:46 Removing old log file testdb/si/24.log
+-----+
+-----+
|Random-Write (done:100000): 0.000010 sec/op; 100000.0 writes/sec(estimated); cost:1.000(sec);
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

Εκτέλεση της read για count 100000 με 1 νήμα:

```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
99992 searching key-99992
99993 searching key-99993
99994 searching key-99994
99995 searching key-99995
99996 searching key-99996
99997 searching key-99997
99998 searching key-99998
99999 searching key-99999
[973] 27 Mar 14:52:14.408 . db.c:31 Closing database 0
[973] 27 Mar 14:52:14.408 . sst.c:415 Sending termination message to the detached thread
[973] 27 Mar 14:52:14.408 - sst.c:176 Exiting from the merge thread as user requested
[973] 27 Mar 14:52:14.409 . sst.c:422 Waiting the merger thread
[973] 27 Mar 14:52:14.409 - file.c:170 Truncating file testdb/si/manifest to 117 bytes
[973] 27 Mar 14:52:14.415 . log.c:46 Removing old log file testdb/si/0.log
[973] 27 Mar 14:52:14.415 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+-----+
+-----+
|Random-Read (done:100000, found:100000): 0.000020 sec/op; 50000.0 reads /sec (estimated); cost:2.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

Εκτέλεση της read για count 100000 με 10 νήματα και πάλι βλέπουμε ότι ο χρόνος μειώθηκε:



```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
89992 searching key-89992
89993 searching key-89993
89994 searching key-89994
89995 searching key-89995
89996 searching key-89996
89997 searching key-89997
89998 searching key-89998
89999 searching key-89999
[1003] 27 Mar 14:53:55.133 . db.c:31 Closing database 0
[1003] 27 Mar 14:53:55.133 . sst.c:415 Sending termination message to the detach
ed thread
[1003] 27 Mar 14:53:55.133 - sst.c:176 Exiting from the merge thread as user req
uested
[1003] 27 Mar 14:53:55.134 . sst.c:422 Waiting the merger thread
[1003] 27 Mar 14:53:55.134 - file.c:170 Truncating file testdb/si/manifest to 81
bytes
[1003] 27 Mar 14:53:55.140 . log.c:46 Removing old log file testdb/si/0.log
[1003] 27 Mar 14:53:55.145 . skiplist.c:57 SkipList refcount is at 0. Freeing up
the structure
+-----+-----+-----+-----+-----+-----+
+-----+
|Random-Read      (done:100000, found:100000): 0.000010 sec/op; 100000.0 reads /se
c(estimated); cost:1.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

### Ερώτημα γ)

Αρχικά να σημειωθεί ότι χρειάστηκε να αλλάξουμε την κλειδαριά που προστατεύουμε την db\_add την οποία έχουμε βάλει εντός σχολίων και αυτήν την διαδικασία την εκτελέσαμε πιο βαθιά μέσα στον κώδικα(θα σας εξηγήσουμε στην συνέχεια ακριβώς πώς το κάναμε και που).

- bench.c:

Πρώτα από όλα να εξηγήσουμε τον τρόπο με τον οποίο δουλεύουμε:

Η λογική μας είναι η εξής, έχοντας τις λειτουργίες read και write και στην διάθεση μας κάποια νήματα μοιράσαμε τις λειτουργίες μισές-μισές,δηλαδή τα μισά νήματα υλοποιούν λειτουργίες get και τα άλλα μισά put(πχ. Αν το πρόγραμμα μας τρέχει με 10 νήματα και count 100 τότε τα πρώτα 5 θα κάνουν put 100 τιμές και τα άλλα 5 θα κάνουν get 100 τιμές).

Αρχικά προσθέσαμε μια επιπλέον λειτουργία την readwrite στην οποία εκτελούμε ταυτόχρονα read και write λειτουργίες. Στη συνέχεια στο if-statement της readwrite περνάμε τις μεταβλητές count , r και untilNumber στην struct που έχουμε δηλώσει και αρχικοποιούμε δύο

μεταβλητές `startNumThWr` και `startNumThRe` τις οποίες χρησιμοποιούμε για να καθορίσουμε τον αριθμό κλειδιού από τον οποίο θα ξεκινήσουν να επεξεργάζονται καινούργια κλειδιά ανάλογα αν το νήμα εκτελεί `put` ή `get` λειτουργία, ώστε και οι `put` και οι `get` λειτουργίες να ξεκινάνε την επεξεργασία από το μηδέν. Στην συνέχεια μέσα στην `for loop` κλειδώνουμε την `threadArgNumberSafe` ώστε να πάρει το κάθε νήμα σωστά τα ορίσματα του και μετά με ένα `if-else-statement` καλούμε με τα πρώτα μισά νήματα λειτουργίες `put` και με τα υπόλοιπα λειτουργίες `get` θέτοντας κάθε φορά στο όρισμα της `struct(startNumTh)` την κατάλληλη μεταβλητή δηλαδή `startNumThWr` για τις `put` λειτουργίες και την `startNumThRe` για τις `get` λειτουργίες. Τέλος καλούμε την `pthread_join` για κάθε νήμα ώστε να περιμένουμε να τερματίσει.

Για να λειτουργήσουν όμως ταυτόχρονα όλες οι λειτουργίες προστατεύσαμε τις δομές τους με `mutexes` και `conditions`:

- `db.h`:

Δηλώνουμε δύο κλειδαριές και 1 `condition` που θα μας χρησιμεύσει στο `db.c`.

- `db.c`:

Αρχικά για να λειτουργήσει σωστά το πρόγραμμα μας έπρεπε να το παγώνουμε για κάθε νήμα κάθε φορά που γεμίζει το `memtable` και χρειάζεται `merge`. Για να το κάνουμε αυτό μέσα στην `db_add` χρησιμοποιήσαμε μια κλειδαριά `mergeSafe` η οποία κλειδώνει κάθε φορά πριν κάνουμε τον έλεγχο για το αν η `memtable` έχει γεμίσει και ξεκλειδώνει αφού έχει τελειώσει ο έλεγχος είτε έχει μπει μέσα στην `if` είτε όχι. Να σημειώσουμε ότι πριν γίνει το ξεκλείδωμα στέλνει ένα σήμα σε όλα τα νήματα που περίμεναν να τελειώσει αυτή η διαδικασία. Επίσης χρησιμοποιούμε και μία `global` μεταβλητή `test` την οποία χρησιμοποιούμε ώστε να ξέρουμε ανά πάσα στιγμή αν εκτελείτε η διαδικασία του `merge`, δηλαδή γίνεται 1 όταν ξεκινάει η διαδικασία και γίνεται και πάλι 0 όταν τελειώνει.

Όσον αφορά το `db_get` αρχικά ελέγχουμε αν τρέχει η διαδικασία του `merge` δηλαδή (`test = 1`) και αν όντως τρέχει καλούμε την εντολή `pthread_cond_wait(&mergeWait , &mergeNumber)` ώστε να περιμένουμε μέχρι να τελειώσει το `merge` και αυτόν τον έλεγχο τον προστατεύουμε με μια κλειδαριά `mergeNumber` διότι η μεταβλητή `test` είναι `global`.

- `memtable.h`:

Σε αυτό το αρχείο κεφαλίδας δηλώνουμε 2 κλειδαριές και μια `condition` που θα χρησιμοποιήσουμε στο `memtable.c` , για την λειτουργία του ενός γραφέα και πολλών αναγνωστών.

- `memtable.c`:

Αρχικά κατα την κλήση της `memtable_edit`, η οποία καλείται από την `memtable_add` βάζουμε μια κλειδαριά στα δύο άκρα της την `lockWriteRead` η οποία επιτρέπει την πρόσβαση σε μόνο έναν γραφέα κάθε φορά.Επίσης πριν το ξεκλείδωμα χρησιμοποιούμε ένα σήμα `waitPut` το οποίο ενημερώνει όλα τα νήματα ότι έχει τελειώσει ο γραφέας.Επιπλέον χρησιμοποιούμε μια `global` μεταβλητή `checkPutOrGet` την οποία αρχικοποιούμε στο 0 και όταν μπαίνει ένας γραφέας στην βάση γίνεται 1 και αντίστοιχα όταν τελειώσει γίνεται ξανά 0 ώστε να γνωρίζουμε ανά πάσα στιγμή αν υπάρχει γραφέας μέσα στην βάση.

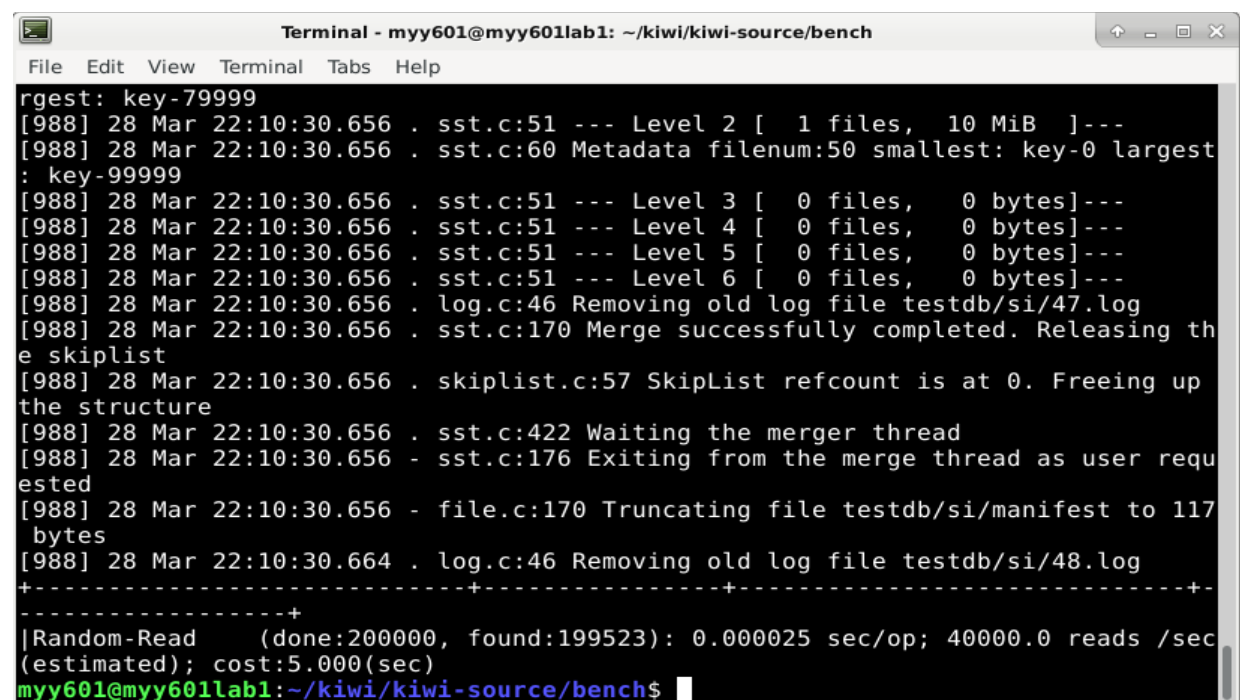
Στην περίπτωση των αναγνωστών όπου καλείται η `memtable_get` χρησιμοποιούμε ένα `if-statement` όπου δοκιμάζουμε να κλειδώσουμε το `lockWriteRead` και αν αυτό επιτευχθεί συνεχίζουμε κανονικά και ξεκλειδώνουμε μετά το ψάξιμο μέσα στο `SkipList`.Σε περίπτωση είναι ήδη κλειδωμένο ελέγχουμε εάν έχει κλειδωθεί από τον γραφέα(με τον έλεγχο `checkPutOrGet == 1` ) και αν αυτό ισχύει τότε καλούμε την συνάρτηση `pthread_cond_wait(&waitPut , &lockWriteRead)` , ώστε να περιμένουμε μέχρι να τελειώσει ο γραφέας.Αν όμως δεν έχει κλειδωθεί από κάποιον γραφέα αλλά από κάποιον αναγνώστη τότε συνεχίζουμε κανονικά για ώστε να επιτευχθεί η λειτουργία των πολλαπλών αναγνωστών.Να σημειώσουμε ότι αυτόν τον έλεγχο τον προστατεύουμε

με μία κλειδαριά testLock για να προστατέψουμε την global μεταβλητή checkPutOrGet.

Παρακάτω έχουμε βάλει μερικά παραδείγματα που τεστάρουμε για να δούμε αν δουλεύει σωστά η λειτουργία readwrite:

Να σημειώσουμε ότι κάποιες φορές κατά την εκτέλεση του προγράμματος όταν χρησιμοποιούμε count μεγαλύτερο των 100000 μας βγάζει Segmentation Fault. Πιστεύουμε ότι για αυτό ευθύνεται η χρήση κάποιας κλειδαριάς σε λάθος σημείο είτε η μη χρήση της σε κάποιο σημείο που έπρεπε.

Για count 200000 με 10 νήματα:



```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
rgest: key-79999
[988] 28 Mar 22:10:30.656 . sst.c:51 --- Level 2 [ 1 files, 10 MiB ]---
[988] 28 Mar 22:10:30.656 . sst.c:60 Metadata filenum:50 smallest: key-0 largest
: key-99999
[988] 28 Mar 22:10:30.656 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[988] 28 Mar 22:10:30.656 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[988] 28 Mar 22:10:30.656 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[988] 28 Mar 22:10:30.656 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[988] 28 Mar 22:10:30.656 . log.c:46 Removing old log file testdb/si/47.log
[988] 28 Mar 22:10:30.656 . sst.c:170 Merge successfully completed. Releasing th
e skiplist
[988] 28 Mar 22:10:30.656 . skiplist.c:57 SkipList refcount is at 0. Freeing up
the structure
[988] 28 Mar 22:10:30.656 . sst.c:422 Waiting the merger thread
[988] 28 Mar 22:10:30.656 - sst.c:176 Exiting from the merge thread as user requ
ested
[988] 28 Mar 22:10:30.656 - file.c:170 Truncating file testdb/si/manifest to 117
bytes
[988] 28 Mar 22:10:30.664 . log.c:46 Removing old log file testdb/si/48.log
+-----+
+-----+
+-----+
|Random-Read (done:200000, found:199523): 0.000025 sec/op; 40000.0 reads /sec
(estimated); cost:5.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

Για count 200000 με 100 νήματα και παρατηρήσαμε ότι το κόστος είναι μικρότερο με 100 νήματα σε σχέση με τα 10:

```
Terminal - myy601@myy601lab1: ~/kiwi/kiwi-source/bench
File Edit View Terminal Tabs Help
argest: key-99999
[6620] 28 Mar 21:33:58.118 . sst.c:51 --- Level 2 [ 1 files, 10 MiB ]---
[6620] 28 Mar 21:33:58.118 . sst.c:60 Metadata filenum:48 smallest: key-0 largest: key-99999
[6620] 28 Mar 21:33:58.118 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]---
[6620] 28 Mar 21:33:58.118 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]---
[6620] 28 Mar 21:33:58.118 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]---
[6620] 28 Mar 21:33:58.118 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]---
[6620] 28 Mar 21:33:58.118 . log.c:46 Removing old log file testdb/si/47.log
[6620] 28 Mar 21:33:58.118 . sst.c:170 Merge successfully completed. Releasing the skiplist
[6620] 28 Mar 21:33:58.118 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
[6620] 28 Mar 21:33:58.118 . sst.c:422 Waiting the merger thread
[6620] 28 Mar 21:33:58.118 . sst.c:176 Exiting from the merge thread as user requested
[6620] 28 Mar 21:33:58.118 . file.c:170 Truncating file testdb/si/manifest to 189 bytes
[6620] 28 Mar 21:33:58.130 . log.c:46 Removing old log file testdb/si/48.log
+-----+
+-----+
|Random-Read (done:200000, found:199778): 0.000020 sec/op; 50000.0 reads /sec (estimated); cost:4.000(sec)
myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

## Ερώτημα δ)

Για το ερώτημα δ έχουμε προσθέσει κάποια πράγματα μόνο στο αρχείο bench.c.

Αρχικά δίνουμε την δυνατότητα στον χρήστη μέσω τέταρτου ορίσματος να επιλέξει το μείγμα λειτουργιών βάσει ποσοστού των put και get που θέλει να διατρέξει βάσει με το count.(πχ εάν έχουμε 100 count και ο χρήστης επιλέξει 60-40 στο τέταρτο όρισμα τότε θα εκτελεστούν 60 λειτουργίες put και 40 λειτουργίες get και σύμφωνα με το ποσοστό αυτό καθορίζεται και η αναλογία των νημάτων δηλαδή αν έχουμε 10 νήματα θα χρησιμοποιηθούν 6 νήματα στην put και 4 στην get).Αν τυχόν όμως δεν επιλέξει κάποιο ποσοστό και ουσιαστικά δεν βάλει τέταρτο όρισμα θα εκτελεστεί το 3<sup>ο</sup> ερώτημα.Να προσθέσουμε ότι υπάρχει διαφορά στην εκτέλεση του 3<sup>ου</sup> ερωτήματος και του 4<sup>ου</sup> ορίσματος αν είναι 50-50(πχ όταν εκτελούμε το 3<sup>ο</sup> ερώτημα και έχουμε το count 100 τότε θα εκτελεστούν 100 put και 100 get ενώ στο 4<sup>ο</sup> ερώτημα όταν έχουμε count 100 και ποσοστό 50-50 θα εκτελεστούν 50

put και 50 get).Τέλος δεν χρειάζεται να πειράζει κάτι ο χρήστης τον κώδικα απλώς να επιλέξει αν θα βάλει τέταρτο όρισμα.

Μέσα στην main προσθέσαμε στο σημείο που έχουμε σε σχόλια τέταρτο ερώτημα έχουμε υλοποιήσει τον κώδικα που αναγνωρίζουμε και αξιοποιούμε το τέταρτο όρισμα.Δηλαδή εάν έχει δώσει 4<sup>ο</sup> όρισμα ο χρήστης μπαίνουμε σε ένα if-statement και εκεί χωρίζουμε το όρισμα που είναι στην μορφή (πχ 60-40) και παίρνουμε τους 2 αριθμούς.Έπειτα ελέγχουμε εάν το άθροισμα τους είναι ίσο με 100 ώστε να μην δωθούν λάθος ορίσματα και αν όντως είναι παίρνουμε τον πρώτο αριθμό και τον μετατρέπουμε σε δεκαδικό ώστε να μας βοηθήσει παρακάτω.Στην περίπτωση που δεν είναι ίσο με 100 το άθροισμα τότε τυπώνουμε ένα error και βγαίνουμε από το πρόγραμμα.Εάν όμως δεν έχει δώσει 4<sup>ο</sup> όρισμα τότε αυτομάτως παίρνουμε τους αριθμούς σαν 50-50.Επιπλέον μέσα στο readwrite το όρισμα count που θα βάλουμε στα νήματα το πολλαπλασιάζουμε με το ποσοστό των put που έχει δώσει ο χρήστης όπως επίσης και μέσα στην for που δημιουργεί τα νήματα και μέσα στο if-statement που είναι υπεύθυνο για το πόσα put νήματα θα δημιουργηθούν πολλαπλασιάζουμε επίσης με το arg1.(πχ αν ο χρήστης δώσει 60-40 και έχουμε 100 count και 10 νήματα το arg1 θα γίνει  $60/100 = 0.6$  , το count θα γίνει  $100 * 0.6 = 60$  και τα νήματα θα γίνουν  $10 * 0.6 = 6$  και όλα αυτά για τις λειτουργίες put όλα τα περισσευούμενα θα γίνουν λειτουργίες get).Τέλος σε κάποια σημεία κάποιες μεταβλητές τις έχουμε πολλαπλασιάσει με την μεταβλητή erwthma3 η οποία είναι 2 εάν δεν έχουμε 4<sup>ο</sup> όρισμα ενώ 1 εάν έχουμε. Έτσι πετυχαίνουμε αυτήν την διαφορά μεταξύ τους ερωτήματος 3 και 4 που αναφέραμε παραπάνω.

Για count 20 και νήματα 10 και 4° όρισμα 60-40:

Για count 50000 και νήματα 10 και 4° όρισμα 60-40:

```
[2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 1 [ 1 files, 1 MiB ]--- [2004] 28 Mar 23:27:03.131 . sst.c:60 Metadata filenum:6 smallest: key-10632 largest: key-9573 [2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 2 [ 1 files, 287 KiB ]--- [2004] 28 Mar 23:27:03.131 . sst.c:60 Metadata filenum:0 smallest: key-0 largest: key-999 [2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 3 [ 0 files, 0 bytes]--- [2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 4 [ 0 files, 0 bytes]--- [2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 5 [ 0 files, 0 bytes]--- [2004] 28 Mar 23:27:03.131 . sst.c:51 --- Level 6 [ 0 files, 0 bytes]--- [2004] 28 Mar 23:27:03.131 . log.c:46 Removing old log file testdb/si/6.log [2004] 28 Mar 23:27:03.131 . sst.c:170 Merge successfully completed. Releasing the skiplist [2004] 28 Mar 23:27:03.131 . skiplist.c:57 Skiplist refcount is at 0. Freeing up the structure [2004] 28 Mar 23:27:03.131 . sst.c:176 Exiting from the merge thread as user requested [2004] 28 Mar 23:27:03.131 . file.c:170 Truncating file testdb/si/manifest to 152 bytes [2004] 28 Mar 23:27:03.132 . log.c:46 Removing old log file testdb/si/7.log +-----+ |Random-Read (done:30000, found:18538): 0.000033 sec/op; 30000.0 reads /sec(estimated); cost:1.000(sec) myy601@myy601lab1:~/kiwi/kiwi-source/bench$
```

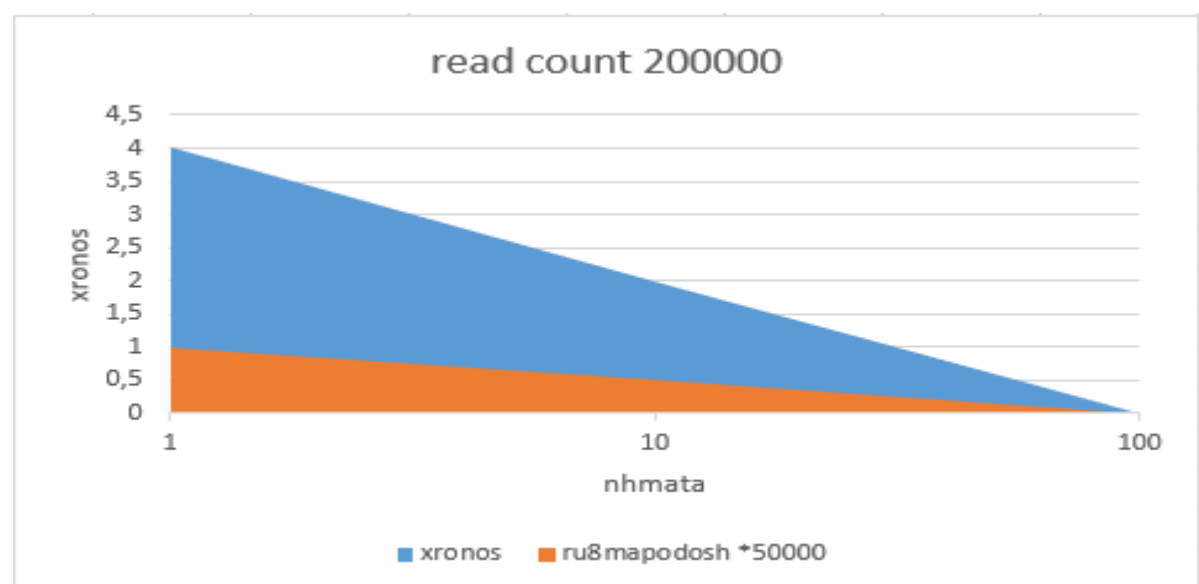
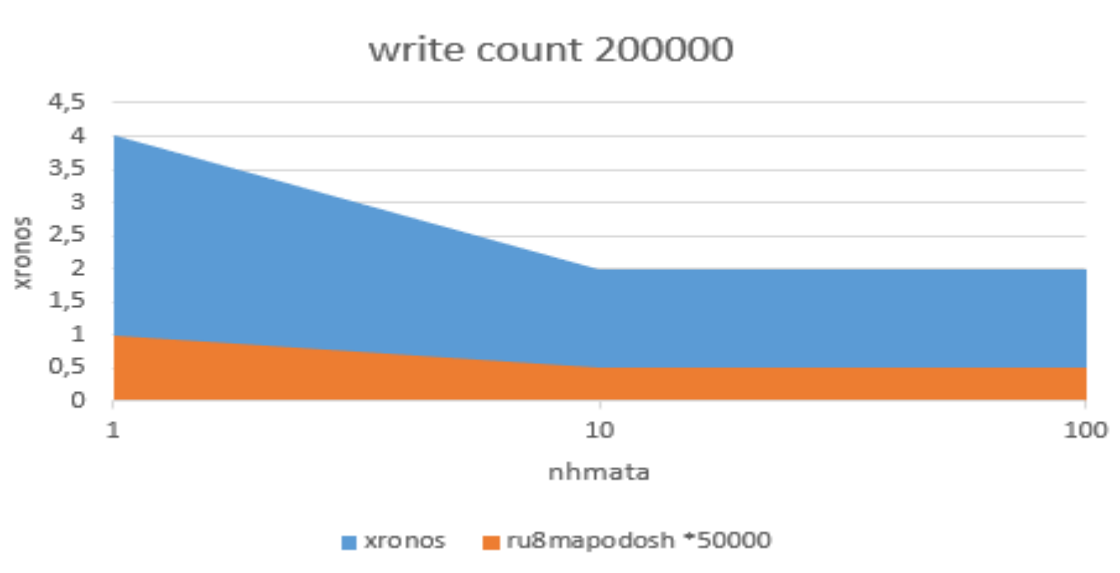
Για count 50000 και νήματα 100 και 4° όρισμα 60-40:

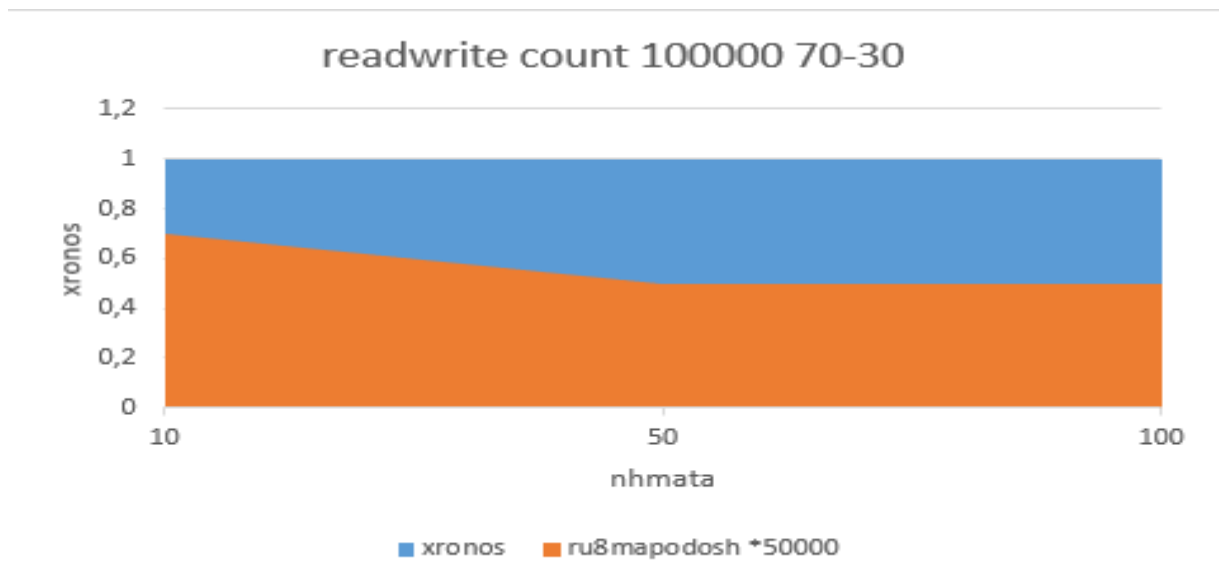
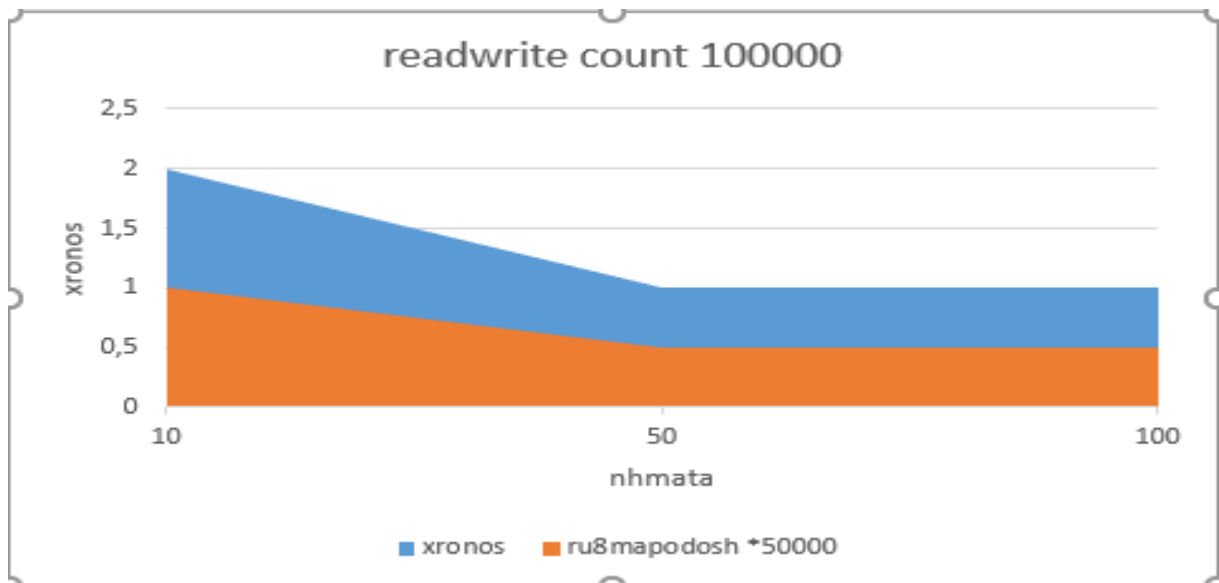




### Ερώτημα ε)

Παρακάτω βάλουμε τα γραφήματα των μετρήσεων που κάναμε για όλα τα ερωτήματα:





### ΑΡΧΕΙΑ ΠΟΥ ΤΡΟΠΟΠΟΙΗΘΗΚΑΝ

1. bench.h
2. bench.c
3. kiwi.c
4. db.h
5. db.c
6. memtable.h
7. memtable.c

