KIWI SEARCH ENGINE MULTITHREADING

Υλοποίηση πολυνηματικής λειτουργίας σε μηχανή αποθήκευσης δεδομένων.

Team

Γκόβαρης Χρήστος-Γρηγόριος
Σπανού Μαρία

Χρονοδιάγραμμα 1ης Εργαστηριακής Άσκησης

Έκδοση	Ημερομηνία	Progress
1.0	13/3/2024	Ανακοίνωση 1ης εργαστηριακής άσκησης
2.0	17/3/2024	Υλοποίηση καθολικής κλειδαριάς (Ταυτοχρονισμός)
2.1	20/3/2024	Βάση: ένας γραφέας - πολλαπλοί αναγνώστες
3.0	25/3/2024	Δημιουργία πολλαπλών νημάτων (γραμμή εντολών)
3.1	27/3/2024	Λειτουργία READWRITE με ποσοστό γραφής
3.2	17/3 – 27/3	Ενημέρωση μετρήσεων και απόδοσης στην οθόνη
3.3	30/3/2024	Αποτελέσματα στο τερματικό
4.0	4/3/2024	Υλοποίηση Report 1ης εργαστηριακής άσκησης

Ανάλυση Χαρακτηριστικών

Η εργασία υλοποιήθηκε σε ένα μηχάνημα (Lenovo Ideapad 3), με τα εξής χαρακτηριστικά πυρήνα:

- AMD Ryzen 7 (7730U)
- 8 CPU cores
- 16 Threads
- Boost Clock up to 4.5GHz
- Base Clock 2.0GHz
- CPU Socket FP6
- Intergraded Graphics (Radeon Graphics)

Πραγματοποιθείσες αλλαγές και αρχεία που τροποποιήθηκαν

Ταυτοχρονισμός της READ και WRITE, ένας γραφέας στην βάση με πολλούς αναγνώστες, πολυνηματική υλοποίηση της READ και WRITE και υλοποίηση της εντολής READWRITE (db.c, kiwi.c, bench.c, bench.h).

2.0 Υλοποίηση καθολικής κλειδαριάς (Ταυτοχρονισμός)

Οι αλλαγές έχουν πραγματοποιηθεί στο αρχείο **db.c και στο **kiwi.c***

Σε αυτό το σημείο της εργασίας, μας ζητήθηκε να υλοποιήσουμε μία καθολική κλειδαριά, ώστε να επιτύχουμε τον **ταυτοχρονισμό** στο πρόγραμμα που μας δόθηκε.

Για να το πραγματοποιήσουμε αυτό, κάναμε χρήση του διάσημου **αλγορίθμου Peterson**, για τον ταυτοχρονισμό, ο οποίος επιτρέπει σε δύο διεργασίες να «μοιραστούν» μία κρίσιμη περιοχή μνήμης με ασφάλεια, χωρίς να παρουσιάζεται ανταγωνισμός. Μέσω της χρήσης αναδρομικής ανταλλαγής προτεραιοτήτων, οι διεργασίας επιτρέπονται να εκτελούνται με «ειρηνικό» τρόπο στο κοινό τμήμα μνήμης.

Αυτή η τεχνική χρησιμοποιήθηκε για την νέα υλοποίηση της **db_add,** όπως φαίνεται παρακάτω από την αρχικοποίηση, καθώς και τον νέο ολοκληρωμένο κώδικα της συνάρτησης:

```c
// mutex initializer
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

// Peterson's Algorithm Variables
int turn;
int flag[2];
```

```c
int db_add(DB* self, Variant* key, Variant* value) {

    // Peterson's Algorithm Variables Initialization
    int id = 0;
    int other = 1 - id;
    flag[id] = 1;
    turn = other;
    while(flag[other] == 1 && turn == other);

    if (memtable_needs_compaction(self->memtable)) {

        // locking for write access
        pthread_mutex_lock(&mutex);

        INFO("Starting compaction of the memtable after %d insertions and %d deletions",
        self->memtable->add_count, self->memtable->del_count);
        sst_merge(self->sst, self->memtable);
        memtable_reset(self->memtable);

        // unlocking for write access
        pthread_mutex_unlock(&mutex);
    }

    // Peterson's Algorithm Variables Initialization
    flag[id] = 0;

    return memtable_add(self->memtable, key, value);

}
```

Όσων αφορά την **db_get**, χρησιμοποιεί την τεχνική του **αμοιβαίου αποκλεισμού** με την χρήση ενός **mutex** το οποίο αρχικοποιήσαμε στην αρχή του κώδικα (όπως φαίνεται παραπάνω). Ο μηχανισμός αυτός εξασφαλίζει ότι μόνο ένα νήμα την φορά, θα έχει πρόσβαση στην κρίσιμη περιοχή του κώδικα που κλειδώνεται από τα mutex. Κατά την διάρκεια της εκτέλεσης db_get, το mutex κλειδώνεται πριν αρχίσει η πρόσβαση στις κοινές τοπικές μεταβλητές, όπως η μνήμη της βάσης δεδομένων. Έτσι επιτυγχάνεται ο αμοιβαίος αποκλεισμός, όπου όταν πολλά νήματα προσπαθούν να αποκτήσουν πρόσβαση ταυτόχρονα, να μην δημιουργούν καταστάσεις ανταγωνισμού.

Όταν η εκτέλεση της συνάρτησης ολοκληρώνεται, το mutex ξεκλειδώνεται επιτρέποντας σε άλλα νήματα να αποκτήσουν πρόσβαση στην κρίσιμη περιοχή του κώδικα. Έτσι, η χρήση του mutex προστατεύει τις κοινές τοπικές μεταβλητές και γενικότερα την κρίσιμη περιοχή, εξασφαλίζοντας της σωστή λειτουργία του προγράμματος, χωρίς ανταγωνισμό.

Η νέα μορφή της db_get, με την σωστή χρήση lock-unlock, φαίνεται παρακάτω:

```
int db_get(DB* self, Variant* key, Variant* value) {

    // locking for write access
    pthread_mutex_lock(&mutex);

    int ret = memtable_get(self->memtable->list, key, value);
    if (ret == 1)return 1;

    // unlocking for write access
    pthread_mutex_unlock(&mutex);

    return sst_get(self->sst, key, value);
}
```

Με χρήση lock-unlock, «περικυκλώσαμε» την **db_add** και **db_get** (στο αρχείο kiwi.c), που έχουν κληθεί στις συναρτήσεις _write_test και _read_test, όπως φαίνεται παρακάτω (για να αποφύγουμε τον ανταγωνισμό των νημάτων μεταξύ τους):

**Για την db_add:*

```
        // locking before write access
        pthread_mutex_lock(&mwrite);

        db_add(db, &sk, &sv);

        // unlocking after write access
        pthread_mutex_unlock(&mwrite);
```

*Για την db_get:

```
for (i = 0; i < count; i++) {

    // locking before modifying shared variables
    pthread_mutex_lock(&mread);


    memset(key, 0, KSIZE + 1);

    if (r) {
        _random_key(key, KSIZE);
    } else {
        snprintf(key, KSIZE, "key-%d", i);
    }

    fprintf(stderr,"%d searching %s\n", i, key);
    sk.length = KSIZE;
    sk.mem = key;
    ret = db_get(db, &sk, &sv);

    if (ret) {
        found++;
    } else {
        INFO("not found key#%s", sk.mem);
    }


    if ((i % 10000) == 0) {
        fprintf(stderr, "random read finished %d ops\n", i);
        fflush(stderr);
    }

    // locking before modifying shared variables
    pthread_mutex_lock(&mread);
}
```

2.1 Βάση: ένας γραφέας – πολλαπλοί αναγνώστες

Για την υλοποίηση **ενός γραφέα στην βάση** και **πολλαπλών αναγνωστών** δημιουργήσαμε τον ακόλουθο κώδικα, στον οποίο χρησιμοποιούμε την τεχνική του **αμοιβαίου αποκλεισμού** και των **συνθηκών αναμονής** (condition variables), για τον συγχρονισμό των νημάτων σε ένα πολυνηματικό περιβάλλον.

Αρχικά, το νήμα που εκτελεί τον κώδικα, προσπαθεί να αποκτήσει πρόσβαση για εγγραφή στην κρίσιμη περιοχή. Χρησιμοποιείται ένα mutex (**mwrite** - το οποίο αρχικοποιήθηκε στην αρχή του αρχείου) για να εξασφαλίσει ότι μόνο ένα νήμα κάθε φορά θα μπορεί να εκτελεί τον κώδικα. Ένα υπάρχουν ήδη αναγνώστες ή γραφείς που περιμένουν να γράψουν (**pending_writers** – οι οποίοι έχουν ήδη αυξηθεί κατά ένα στο προηγούμενο βήμα) στο σύστημα, το νήμα περιμένει, χρησιμοποιώντας μία condition variable (**can_write**). Όταν η πρόσβαση είναι διαθέσιμη, το νήμα αυξάνει τον αριθμό των γραφεών (**writers**) και μειώνει των pending_writers, καταχωρεί το δικαίωμα του να γράψει και έπειτα ξεκλειδώνει το mutex. Στην συνέχεια, περιμένει την υλοποίηση του υπόλοιπου κώδικα και ενδέχεται να περιμένει ώστε κάποιο από τα νήματα ανάγνωσης να εκτελέσεις τις δικές του ενέργειες. Όταν ολοκληρώνει την εγγραφή του, μειώνει τους writers, ελέγχει αν υπάρχουν περισσότεροι pending_writers σε αναμονή και αν ναι, συνεχίζει να ειδοποιεί έναν εκ των γραφέων. Διαφορετικά, ειδοποιεί τους αναγνώστες με την χρήση του condition variable **can_read**.

Όσων αφορά τα νήματα ανάγνωσης, το νήμα που εκτελεί την **_read_test**, αποκτά πρόσβαση για ανάγνωση χρησιμοποιώντας ένα διαφορετικό mutex (**mread**). Εκτελεί αντίστοιχα με την **_write_test** και περιμένει μέχρι να είναι διαθέσιμη η πρόσβαση για ανάγνωση, αυξάνει τον αριθμό των αναγνωστών (**readers**) και στην συνέχεια ξεκλειδώνει το mutex. Όταν τελειώνει με την ανάγνωση, μειώνει τον αριθμό των αναγνωστών και ενδεχομένως ειδοποιεί έναν εκ των γραφόντων για να αποκτήσει πρόσβαση.

Το όλο σύστημα, επιτυγχάνει συγχρονισμό μεταξύ γραφέων και αναγνωστών, αποτρέποντας την εμφάνιση κολλήματος και εξασφαλίζοντας την ορθή κοινού πόρου πρόσβαση.

Στην παρακάτω εικόνα, φαίνονται οι αρχικοποιήσεις όλων των μεταβλητών που χρησιμοποιήθηκαν, για την υλοποίηση του ζητουμένου:

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο kiwi.c*

```
// mutex initializer
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;

// number of current readers
int readers = 0;

// number of current writers
int writers = 0;

// number of pending writers
int pending_writers = 0;

// mutex for writers
pthread_mutex_t mwrite = PTHREAD_MUTEX_INITIALIZER;

// mutex for readers
pthread_mutex_t mread = PTHREAD_MUTEX_INITIALIZER;

// condition variable for readers
pthread_cond_t can_read = PTHREAD_COND_INITIALIZER;

// condition variable for writers
pthread_cond_t can_write = PTHREAD_COND_INITIALIZER;
```

Στο παρακάτω σημείο του κώδικα, φαίνεται η συνάρτηση **_write_test**, με την λειτουργία που αναφέραμε άνωθι:

```
    // locking for write access
    pthread_mutex_lock(&mwrite);

    // increment pending writers counter
    pending_writers++;

    while (readers > 0 || writers > 0) {
        // wait until no readers or writers
        pthread_cond_wait(&can_write, &mwrite);
    }

    // decrement pending writers counter
    pending_writers--;

    // increment writers counter
    writers++;

    // unlocking after write access
    pthread_mutex_unlock(&mwrite);
```

```
// locking before modifying shared variables
pthread_mutex_lock(&mwrite);

// decrement writers counter
writers--;

// if there are pending writers
if (pending_writers > 0) {
    // signal one pending writer
    pthread_cond_signal(&can_write);
} else {
    // if no pending writers, signal readers
    pthread_cond_broadcast(&can_read);
}

// unlocking after modifying shared variables
pthread_mutex_unlock(&mwrite);
```

Στο παρακάτω σημείο του κώδικα, φαίνεται η συνάρτηση **_read_test**, με την λειτουργία που αναφέραμε άνωθι:

```
// locking for read access
pthread_mutex_lock(&mread);

while (writers > 0 || pending_writers > 0) {
    // wait until no writers or pending writers
    pthread_cond_wait(&can_read, &mread);
}

// increment readers counter
readers++;

// unlocking after read access
pthread_mutex_unlock(&mread);
```

```
// locking before modifying shared variables
pthread_mutex_lock(&mread);

// decrement readers counter
readers--;

// if no more readers
if (readers == 0) {
    // signal a pending writer
    pthread_cond_signal(&can_write);
}

// unlocking after modifying shared variables
pthread_mutex_unlock(&mread);
```

3.0 Δημιουργία πολλαπλών νημάτων (γραμμή εντολών)

Ο παρακάτω κώδικας χρησιμοποιεί την τεχνική του **διαχωρισμού διεργασιών σε πολυνηματικές εφαρμογές** για την εκτέλεση εργασιών, εντός πολλαπλών νημάτων. Η συνάρτηση **thread_maker** διαιρεί τον συνολικό αριθμό διεργασιών, που πρέπει να εκτελεστούν μεταξύ των διαθέσιμων νημάτων και καθορίζει τις παραμέτρους για κάθε νήμα που θα δημιουργηθεί. Αυτό επιτυγχάνεται, με την χρήση της μεταβλητής **num_threads** για να καθορίσει πόσα νήματα θα χρησιμοποιηθούν για την εκτέλεση των εργασιών, καθώς και με την χρήση των μεταβλητών **nth** και **remain** για τον διαχωρισμό των εργασιών μεταξύ των νημάτων.

Για κάθε νήμα που δημιουργείται, καθορίζονται οι παράμετροί του, όπως ο αριθμός των εργασιών που πρέπει να εκτελεστεί (**count**), η λειτουργία που πρέπει να εκτελέσει (είτε **εγγραφή** είτε **ανάγνωση**) και η περιοχή των δεδομένων που πρέπει να χειριστεί. Έπειτα, δημιουργείται το νήμα, χρησιμοποιώντας της συνάρτηση **pthread_create()** που καλείται να εκτελέσει μία συγκεκριμένη συνάρτηση (_write_test ή _read_test – ανάλογα με την λειτουργία που ζητήθηκε) με τις παραμέτρους που ορίζονται στο **struct πεδίο threads**. Τέλος, τα νήματα, συγχρονίζονται, ώστε να περιμένουν την ολοκλήρωση της εκτέλεσής τους με την χρήση της συνάρτησης **pthread_join()**, που εξασφαλίζει ότι η εκτέλεση του κυρίως νήματος θα περιμένει μέχρι να ολοκληρωθούν όλα τα υπόλοιπα νήματα.

Τέλος, για την δημιουργία του struct δυναμικού πίνακα threads, ο οποίος έχει θέσεις όσες είναι τα **available_threads** που έχουν αρχικοποιηθεί στο αρχείο **bench.h**, ελευθερώνεται στο τέλος με την χρήση της συνάρτησης **free()**. Με αυτόν τον τρόπο, η τεχνική του διαχωρισμού εργασιών σε πολλαπλά νήματα, επιτρέπει την αποτελεσματική εκτέλεση πολυνηματικών εφαρμογών, βελτιώνοντας την απόδοση και την απόκριση του συστήματος.

Η κλήση της thread_maker έγινε στο αρχείο **bench.c**, στο οποίο ελέγχουμε αν η είσοδος αποτελείται από 3 ή 4 ορίσματα και αν το 4° όρισμα είναι ίσο με 0. Τότε, αν τα ορίσματα είναι 3 ή το 4° είναι ίσο με 0, καλεί την **write_without_threads** ή την **read_without_threads**, αλλιώς καλεί την thread_maker, όπως φαίνεται παρακάτω:

*(οι συναρτήσεις **write_without_threads** και **read_without_threads** είναι ίδιες με τις αρχικές υλοποιήσεις των _write_test και _read_test, οπότε δεν χρήζουν επισύναψη στην αναφορά)*

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο kiwi.c*

**Για την thread_maker:*

```c
// function that creates threads for each operation
void thread_maker(char* operation, long int count, int num_threads) {

    // number of each thread operations
    long int nth = count / num_threads;

    // remainder of operations - added to last thread
    int remain = count % num_threads;

    // starting point of the first thread
    long int start = 0;

    // final destination of the first thread
    long int end = start + nth;

    // defining thread pointer - memory allocation for threads
    final* threads;
    threads = (final*)malloc(available_threads*sizeof(final));

    if (strcmp(operation, "write") == 0){
        if (num_threads<=available_threads){
            for (int i = 0; i < num_threads; i++){

                // passing count into threads[i]
                threads[i].count = count;

                // passing r into threads[i]
                threads[i].r = 1;

                // passing start into threads[i]
                threads[i].start = start;

                // passing end into threads[i]
                threads[i].end = end;

                // passing num_threads into threads[i]
                threads[i].num_threads = num_threads;
```

```
        // passing operation into threads[i]
        strcpy(threads[i].operation, "write");

        // passing write_percentage into threads[i]
        threads[i].write_percentage = 0;

        // creating a new thread, calls _write_test function, with the parameters of threads
        pthread_create(&threads[i].tid, NULL, _write_test, (void *) &threads[i]);

        // change start for the next thread
        start = end;

        // if the next thread, is the last thread
        if (i == num_threads-2) {
            end = end + nth + remain;
            threads[i].count =  nth + remain;
        } else {
            end = end + nth;
        }
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i].tid, NULL);
    }

    free(threads);

} else {
    fprintf(stderr,"Usage: db-bench (<write | read> <count> <random>) (<readwrite> <count> <writePercentage> <random>) \n");
    exit(1);
}
```

```
    } else if (strcmp(operation, "read") == 0) {
        if (num_threads <= available_threads) {
            for (int i = 0; i < num_threads; i++) {

                // passing count into threads[i]
                threads[i].count = count;

                // passing r into threads[i]
                threads[i].r = 1;

                // passing start into threads[i]
                threads[i].start = start;

                // passing end into threads[i]
                threads[i].end = end;

                // passing num_threads into threads[i]
                threads[i].num_threads = num_threads;

                // passing operation into threads[i]
                strcpy(threads[i].operation,"read");

                // passing write_percentage into threads[i]
                threads[i].write_percentage = 0;

                // creating a new thread, calls _read_test function, with the parameters of threads
                pthread_create(&threads[i].tid, NULL, _read_test, (void *) &threads[i]);

                // change start for the next thread
                start = end;
```

```
                // if the next thread, is the last thread
                if (i == num_threads-2) {
                    end = end + nth + remain;
                } else {
                    end = end + nth;
                }
            }

            for (int i = 0; i < num_threads; i++){
                pthread_join(threads[i].tid, NULL);
            }

            free(threads);

        } else {
            fprintf(stderr,"Usage: db-bench (<write | read> <count> <random>) (<readwrite> <count> <writePercentage> <random>) \n");
            exit(1);
        }
    }
}
```

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο bench.c*

```c
int main(int argc, char** argv) {

    // number of operations
    long int count;

    // number of threads
    int num_threads;

    srand(time(NULL));

    if (argc < 3) {
        fprintf(stderr,"Usage: db-bench (<write | read> <count> <random>) (<readwrite> <count> <writePercentage> <random>) \n");
        exit(1);
    }

    if (strcmp(argv[1], "write") == 0) {

        int r = 0;
        count = atoi(argv[2]);

        // if no threads
        if (argc == 3 || atoi(argv[3]) == 0) {
            _print_header(count);
            _print_environment();
            write_without_threads(count, r);
        }

        // if threads
        else if (argc == 4) {
            r = 1;
            num_threads = atoi(argv[3]);
            _print_header(count);
            _print_environment();
            thread_maker(argv[1], count, num_threads);
        }
```

```c
    } else if (strcmp(argv[1], "read") == 0) {

        int r = 0;
        count = atoi(argv[2]);

        // if no threads
        if (argc == 3 || atoi(argv[3]) == 0) {
            _print_header(count);
            _print_environment();
            read_without_threads(count, r);
        }

        // if threads
        else if (argc == 4) {
            r = 1;
            num_threads = atoi(argv[3]);
            _print_header(count);
            _print_environment();
            thread_maker(argv[1], count, num_threads);
        }
```

Όσων αφορά την δημιουργία νημάτων, είναι γνωστό ότι η συνάρτηση η οποία καλείται να εκτελέσει το thread, είναι τύπου **void *** (<όνομα συνάρτησης> (void* arg)), οπότε οι _write_test και _read_test, αναδιαμορφώθηκαν έτσι ώστε να πληρούν αυτές τις προϋποθέσεις. Τα ορίσματά τους ορίστηκαν ως τοπικές μεταβλητές μέσα στην συνάρτηση, αντιστοιχισμένες με τα πεδία του struct final, που ορίστηκε μέσα στο **bench.h**. Επίσης, υπάρχουν lock-unlock στις εντολές **db_add** και **db_get** αντίστοιχα

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο kiwi.c*

**Για την _write_test:*

```
// function that gets called to the function thread_maker for write purposes
void* _write_test(void * arg) {

    // creates a pointer of struct final type
    final* args = (final*) arg ;

    // initializes struct variables
    long int count = args->count;
    int r = args->r;
```

```
        // locking before write access
        pthread_mutex_lock(&mwrite);

        db_add(db, &sk, &sv);

        // unlocking after write access
        pthread_mutex_unlock(&mwrite);
```

**Για την _read_test:*

```
// function that gets called to the function thread_maker for read purposes
void* _read_test(void * arg) {

    // creates a pointer of struct final type
    final* args = (final*) arg ;

    // initializes struct variables
    long int count = args->count;
    int r = args->r;
```

```
    // locking for read access
    pthread_mutex_lock(&mread);

    ret = db_get(db, &sk, &sv);

    // unlocking for read access
    pthread_mutex_unlock(&mread);
```

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο bench.h*

**Για τα available_threads:*

```
// max threads of our machine
#define available_threads 16
```

**Για το struct final:*

```
// this struct initializes elements on each thread
typedef struct {

    // starting point of the first thread
    long int start;

    // final destination of the first thread
    long int end;

    // number of operations
    long int count;

    // random flag
    int r;

    // number of threads
    int num_threads;

    // name of operation
    char operation[10];

    // write percentage
    int write_percentage;

    // thread tid
    pthread_t tid;

} final;
```

3.1 Λειτουργία READWRITE με ποσοστό γραφής

Η συνάρτηση **_readwrite_test** λειτουργεί ως μία πολυνηματική διαχείριση για να δοκιμάσει την **απόδοση ανάγνωσης και εγγραφής** σε έναν κοινό πόρο, με **διαφορετικά ποσοστά εγγραφής**. Η συνάρτηση, δέχεται ως ορίσματα τον αριθμό των επιθυμητών λειτουργιών, τον τύπο της λειτουργίας (ανάγνωση-εγγραφή), τον αριθμό των νημάτων που πρέπει να δημιουργηθούν και το ποσοστό των εγγραφών στο συνολικό αριθμό επιθυμητών ενεργειών.

Η αρχικοποιήσεις των μεταβλητών, περιλαμβάνουν την δέσμευση μνήμης για την αποθήκευση πληροφοριών των νημάτων και τον υπολογισμό του αριθμού των εγγραφών και αναγνώσεων, που πρέπει να πραγματοποιηθούν από κάθε νήμα. Έπειτα, γίνεται η διαίρεση των εργασιών ανάμεσα στα νήματα και η εκχώρηση των παραμέτρων τους.

Η δημιουργία νημάτων, γίνεται εναλλάξ ανάλογα με τον τύπο λειτουργίας:

- Αν πρόκειται για εγγραφή, τα νήματα δημιουργούνται καλώντας την συνάρτηση _write_test, ενώ
- Αν πρόκειται για ανάγνωση, καλείται η _read_test.

Κάθε νήμα, εκτείνεται με τις παραμέτρους που έχουν οριστεί και αρχίζει την εκτέλεσή του.

Τέλος, η συνάρτηση αναμένει την ολοκλήρωση όλων των νημάτων πριν από την αποδέσμευση της μνήμης που δεσμεύτηκε για τα νήματα. Μέσω της **pthread_join()**, η κύρια λειτουργία περιμένει μέχρι να ολοκληρωθεί η εκτέλεση κάθε νήματος. Κατόπιν, η μνήμη που δεσμεύτηκε για τα νήματα, αποδεσμεύεται χρησιμοποιώντας την συνάρτηση **free()**.

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο kiwi.c*

```c
// function that is capable of simultaneous reads and writes using percentages
void _readwrite_test(long int count, char* operation, int num_threads, int writePercentage) {

    // defining thread pointer - memory allocation for threads
    final* threads;
    threads=(final*)malloc(num_threads*sizeof(final));

    // number of write operations
    int num_write = count * writePercentage/100;

    // number of read operations
    int num_read = count-num_write;

    // number of operations for each thread
    long int nth = count / num_threads;

    // number of remaining operations to be added to the last thread
    int remain = count % num_threads;
```

```
// char arrays to save the word 'readwrite'
char read[5];
char write[6];
memcpy(read, operation, 4);
read[4] = '\0';
memcpy(write, operation + 4, 5);
write[5] = '\0';

// starting point of the first thread
long int start = 0;

// final destination of the first thread
long int end = start + nth;

for(int i = 0; i < num_threads; i++){

    // passing r into threads[i]
    threads[i].r = 1;

    // passing start into threads[i]
    threads[i].start = start;

    // passing end into threads[i]
    threads[i].end = end;

    // passing num_threads into threads[i]
    threads[i].num_threads = num_threads;

    // passing write_percentage into threads[i]
    threads[i].write_percentage = writePercentage;
```

```
    if (strcmp(write, "write") == 0) {

        // passing count into threads[i]
        threads[i].count = num_write;

        // passing operation into threads[i]
        strcpy(threads[i].operation, "write");

        // creating a new thread, calls _write_test function, with the parameters of threads
        pthread_create(&threads[i].tid, NULL, _write_test, (void *) &threads[i]);

        // change start for the next thread
        start = end;

        // if the next thread, is the last thread
        if (i == num_threads-2) {
            end = end + nth + remain;
        } else {
            end = end + nth;
        }
```

```
        if (strcmp(read, "read") == 0) {

            // passing count into threads[i]
            threads[i].count = num_read;

            // passing operation into threads[i]
            strcpy(threads[i].operation, "read");

            // creating a new thread, calls _read_test function, with the parameters of threads
            pthread_create(&threads[i].tid, NULL, _read_test, (void *) &threads[i]);

            // change start for the next thread
            start = end;

            // if the next thread, is the last thread
            if (i == num_threads-2) {
                end = end + nth + remain;
            } else {
                end = end + nth;
            }
        }
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i].tid, NULL);
    }

    free(threads);
}
```

Η κλήση της συνάρτησης έγινε στο αρχείο **bench.c**, ελέγχοντας για το αν η πρώτη παράμετρος που δόθηκε κατά την εκτέλεση του προγράμματος είναι readwrite. Σε αυτήν την περίπτωση, πραγματοποιείται η απαραίτητη προεπεξεργασία για την δοκιμή απόδοσης, ανάγνωσης και εγγραφής σε έναν κοινό πόρο. Η παράμετρος **num_threads** λαμβάνεται από το 4ο όρισμα της γραμμής εντολών, ενώ το ποσοστό εγγραφής από το 3ο. Το πρόγραμμα εκτυπώνει την κεφαλίδα για την έναρξη της δοκιμής, καθώς και πληροφορίες σχετικά με το περιβάλλον εκτέλεσης. Σε περίπτωση που δόθηκαν λανθασμένες τιμές για το ποσοστό εγγραφής, εμφανίζεται ένα μήνυμα σφάλματος και το πρόγραμμα τερματίζει. Τέλος, καλείται η συνάρτηση _readwrite_test, για την πραγματοποίηση της δοκιμής, με την πέρασμα των αναγκαίων παραμέτρων.

**Οι αλλαγές πραγματοποιήθηκαν στο αρχείο bench.c*

```
} else if (strcmp(argv[1], "readwrite") == 0) {

    int r = 0;
    int num_threads = atoi(argv[4]);

    // gets write percentage from main argument
    int perWrite = atoi(argv[3]);

    count = atoi(argv[2]);
    _print_header(count);
    _print_environment();

    if (argc == 5) {
        r = 1;
    }

    // check for invalid write percentage
    if (perWrite< 1 || perWrite > 99) {
        printf("Write percentage should be in range [1-99]\n");
        exit(-1);
    }

    _readwrite_test(count,argv[1],num_threads,perWrite);

} else {
    // changed to accept more parameters
    fprintf(stderr,"Usage: db-bench (<write | read> <count> <random>) (<readwrite> <count> <writePercentage> <random>) \n");
    exit(1);
}

return 1;
}
```

3.2 Ενημέρωση μετρήσεων και απόδοσης στην οθόνη

Το παρακάτω τμήμα κώδικα, χρησιμοποιείται για την σωστή ενημέρωση των μετρήσεων απόδοσης και την εκτύπωσή τους στο αρχείο **output.txt**. Αρχικά, **κλειδώνεται** ένα **mutex** με σκοπό την ασφαλή πρόσβαση στο αρχείο εξόδου. Έπειτα, ανοίγει το αρχείο για **εγγραφή**, υπολογίζει τον **συνολικό αριθμό ενεργειών** που πραγματοποίησε το νήμα και εκτυπώνει τα αποτελέσματα στο αρχείο. Η εγγραφή περιλαμβάνει τον αριθμό του νήματος, τον συνολικό αριθμό ενεργειών του νήματος, τον μέσο χρόνο εκτέλεσης ανά ενέργεια, τον εκτιμώμενο αριθμό ενεργειών ανά δευτερόλεπτο, καθώς και τον συνολικό χρόνο εκτέλεσης του νήματος. Τέλος, το αρχείο κλείνει και το mutex ξεκλειδώνεται για την επόμενη ενημέρωση μετρήσεων απόδοσης.

**Οι παραπάνω αλλαγές πραγματοποιήθηκαν στο αρχείο kiwi.c και στις συναρτήσεις _write_test και _read_test αντίστοιχα.*

**Για την _write_test:*

```c
// locking before writing to the file
pthread_mutex_lock(&writeStatsToFile);

// open the output file
fp = fopen("output.txt", "a+");

// calculate the total number of actions performed by the thread
long int threadTotal = args->end-args->start;

// print the results to the file
fprintf(fp, "Thread %ld completed execution, %ld writes: %.6f sec/op; %.1f writes/sec(estimated); cost:%.3f(sec);\n"
    ,args->tid, threadTotal, (double) (cost/threadTotal), (double) (threadTotal/cost), cost);

// close the file
fclose(fp);

// unlocking after writing to the file
pthread_mutex_unlock(&writeStatsToFile);
```

**Για την _read_test:*

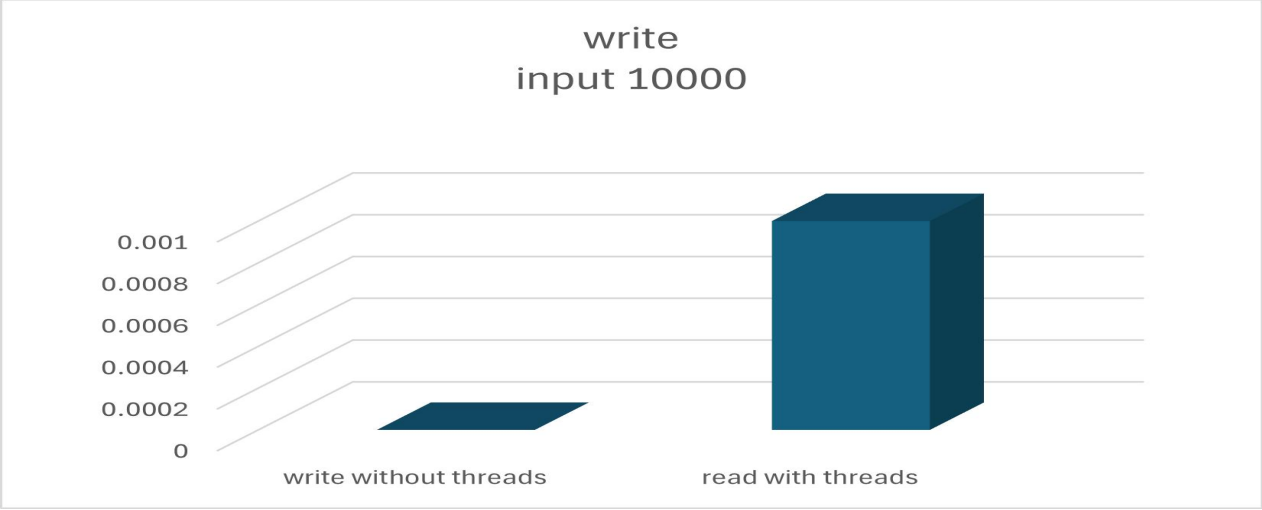
```c
// locking before writing to the file
pthread_mutex_lock(&writeStatsToFile);

// open the output file
fp = fopen("output.txt", "a+");

// calculate the total number of actions performed by the thread
long int threadTotal = args->end-args->start;

// print the results to the file
fprintf(fp, "Thread %ld completed execution, found %d/%ld reads: %.6f sec/op; %.1f reads/sec(estimated); cost:%.3f(sec);\n"
    ,args->tid, found, threadTotal, (double) (cost/threadTotal), (double) (threadTotal/cost), cost);

// close the file
fclose(fp);

// unlocking before writing to the file
pthread_mutex_unlock(&writeStatsToFile);
```

ΤΑ ΠΡΩΤΌΤΥΠΑ ΟΛΩΝ ΤΩΝ ΠΑΡΑΠΆΝΩ ΣΥΝΑΡΤΉΣΕΩΝ, ΈΧΟΥΝ ΟΡΙΣΤΕΊ ΣΤΟ ΑΡΧΕΊΟ bench.h

```c
// function that reads without using any threads
void read_without_threads(long int count, int r);

// function that writes without using any threads
void write_without_threads(long int count, int r);

// function that gets called to the function thread_maker for write purposes
void _write_test(void* arg);

// function that gets called to the function thread_maker for read purposes
void* _read_test(void* arg);

// function that creates threads for each operation
void thread_maker(char* operation, long int count, int num_threads);

// function that is capable of simultaneous reads and writes using percentages
void _readwrite_test(long int count, char* operation, int num_threads, int writePercentage);
```

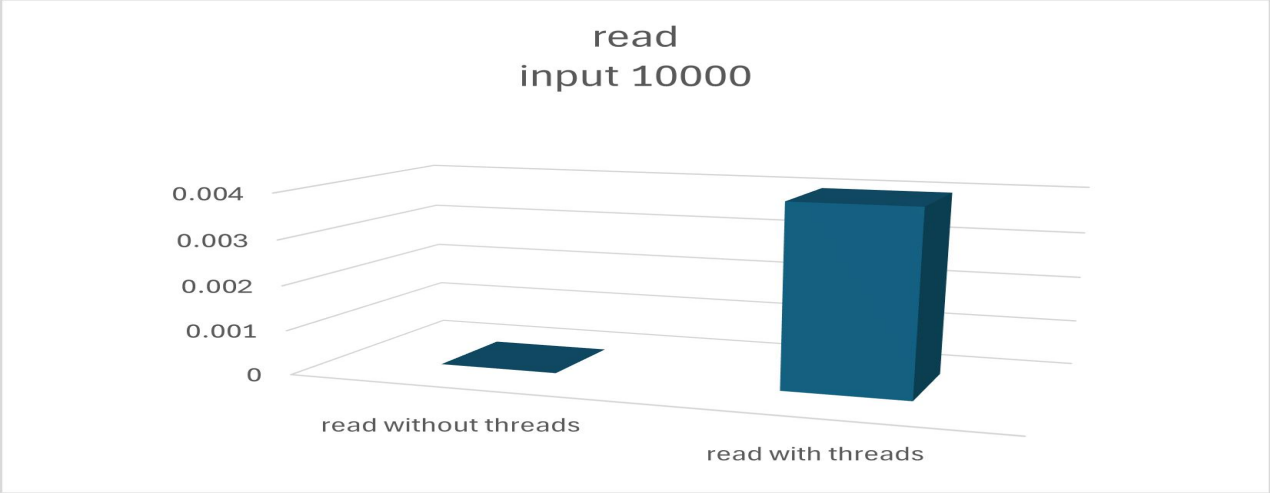
Δημιουργήσαμε, λοιπόν, το αρχείο output.txt που φαίνεται παρακάτω:

Για την εντολή **write:*




```
 1 Thread 140415480235712 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 2 Thread 140415471843008 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 3 Thread 140415463450304 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 4 Thread 140415365805760 completed execution, 2500 writes: 0.000400 sec/op; 2500.0 writes/sec(estimated); cost:1.000(sec);
 5 Thread 139704697616064 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 6 Thread 139704680830656 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 7 Thread 139704689223360 completed execution, 2500 writes: 0.000400 sec/op; 2500.0 writes/sec(estimated); cost:1.000(sec);
 8 Thread 139704672437952 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
 9 Thread 140027887613632 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
10 Thread 140027879220928 completed execution, 2500 writes: 0.000400 sec/op; 2500.0 writes/sec(estimated); cost:1.000(sec);
11 Thread 140027870828224 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
12 Thread 140027745007296 completed execution, 2500 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
```

Για την εντολή **read:*




```
13 Thread 140625710151360 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
14 Thread 140625684973248 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
15 Thread 140625693365952 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
16 Thread 140625701758656 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
17 Thread 140130467575488 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
18 Thread 140130450790080 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
19 Thread 140130459182784 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
20 Thread 140130442397376 completed execution, found 0/2500 reads: 0.000400 sec/op; 2500.0 reads/sec(estimated); cost:1.000(sec);
21 Thread 139880704374464 completed execution, found 0/2500 reads: 0.000800 sec/op; 1250.0 reads/sec(estimated); cost:2.000(sec);
22 Thread 139880712767168 completed execution, found 0/2500 reads: 0.000800 sec/op; 1250.0 reads/sec(estimated); cost:2.000(sec);
23 Thread 139880695981760 completed execution, found 0/2500 reads: 0.000800 sec/op; 1250.0 reads/sec(estimated); cost:2.000(sec);
24 Thread 139880687589056 completed execution, found 0/2500 reads: 0.000800 sec/op; 1250.0 reads/sec(estimated); cost:2.000(sec);
```

Για την εντολή **readwrite:*

```
25 Thread 140100427183808 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
26 Thread 140100410398400 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
27 Thread 140100385220288 completed execution, 1250 writes: 0.000800 sec/op; 1250.0 writes/sec(estimated); cost:1.000(sec);
28 Thread 140100281296576 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
29 Thread 140100247725760 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
30 Thread 140100264511168 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
31 Thread 140100368434880 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
32 Thread 140100230940352 completed execution, 1250 writes: 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
33 Thread 140100385220288 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
34 Thread 140100247725760 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
35 Thread 140100427183808 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
36 Thread 140100264511168 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
37 Thread 140100410398400 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
38 Thread 140100230940352 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
39 Thread 140100281296576 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
40 Thread 140100368434880 completed execution, found 0/1250 reads: 0.000000 sec/op; inf reads/sec(estimated); cost:0.000(sec);
```

Η μεγάλη διαφορά στους χρόνους, οφείλεται στην ύπαρξη των threads, οι οποίοι ανάλογα με το μηχάνημα, είτε αυξάνονται είτε μειώνονται. Η χρονική τους διαφορά, είναι αναμενόμενη εφόσον έχει να κάνει με το μηχάνημα και τις δυνατότητές του. Σε ένα πιο γρήγορο μηχάνημα, θα υπήρχαν διαφορετικοί χρόνοι.

3.3 Αποτελέσματα στο τερματικό

Σε αυτό το σημείο της αναφοράς μας, ακολουθούν τα **αποτελέσματα** που μας εμφανίζονται στο τερματικό (**screenshots**) μετά από την **εκτέλεση της κάθε εντολής**.

Εκτέλεσης της εντολής **make all**

Εκτέλεση της εντολής **./kiwi-bench write 10000**

```
[2817] 07 Apr 16:08:54.592 - sst.c:165 The merge thread received a MERGE job
[2817] 07 Apr 16:08:54.592 . sst.c:166 Merging inside compaction thread
[2817] 07 Apr 16:08:54.592 . sst.c:608 Compacting the memtable to a SST file
[2817] 07 Apr 16:08:54.592 - sst.c:877 Range [key-8234, key-9999] DOES NOT overlap in level 0. Checking others
[2817] 07 Apr 16:08:54.592 - sst.c:825 Extracted range: [key-0, key-9999]
[2817] 07 Apr 16:08:54.592 - sst.c:892 Range [key-8234, key-9999] DOES  overlap in level 2. Checking others
[2817] 07 Apr 16:08:54.592 . sst.c:929 Using level 1 for memtable compaction [key-8234, key-9999]
[2817] 07 Apr 16:08:54.592 . file.c:200 Creating directory structure: testdb/si/1
[2817] 07 Apr 16:08:54.592 . sst.c:633 Compaction of 1766 [1799554 bytes allocated] elements started
[2817] 07 Apr 16:08:54.594 - sst_builder.c:167 Index block @ offset: 0x1CC9D size: 8451
[2817] 07 Apr 16:08:54.594 - sst_builder.c:168 Meta block @ offset: 0x1CC55 size: 72
[2817] 07 Apr 16:08:54.594 - sst_builder.c:171 Bloom block @ offset: 0x1BBB9 size: 4252
[2817] 07 Apr 16:08:54.594 - file.c:170 Truncating file testdb/si/1/2.sst to 126440 bytes
[2817] 07 Apr 16:08:54.595 . file.c:65 Mapping of 126440 bytes for testdb/si/1/2.sst
[2817] 07 Apr 16:08:54.596 - sst_loader.c:183 Index @ offset: 117917 size: 8451
[2817] 07 Apr 16:08:54.596 - sst_loader.c:184 Meta Block @ offset: 117845 size: 72
[2817] 07 Apr 16:08:54.596 . sst_loader.c:201 Data size:        113593
[2817] 07 Apr 16:08:54.596 . sst_loader.c:203 Index size:       0
[2817] 07 Apr 16:08:54.596 . sst_loader.c:204 Key size:         28256
[2817] 07 Apr 16:08:54.596 . sst_loader.c:205 Num blocks size:  354
[2817] 07 Apr 16:08:54.596 . sst_loader.c:206 Num entries size: 1766
[2817] 07 Apr 16:08:54.596 . sst_loader.c:207 Value size:       1766000
[2817] 07 Apr 16:08:54.596 . sst_loader.c:210 Filter size:      4252
[2817] 07 Apr 16:08:54.596 . sst_loader.c:211 Bloom offset 113593 size: 4252
[2817] 07 Apr 16:08:54.596 . sst.c:635 Compaction of 1766 elements finished
[2817] 07 Apr 16:08:54.596 - file.c:170 Truncating file testdb/si/manifest to 116 bytes
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 0 [  0 files,   0 bytes]---
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 1 [  2 files, 411 KiB  ]---
[2817] 07 Apr 16:08:54.596 . sst.c:55 Metadata filenum:1 smallest: key-4117 largest: key-8233
[2817] 07 Apr 16:08:54.596 . sst.c:55 Metadata filenum:2 smallest: key-8234 largest: key-9999
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 2 [  1 files, 286 KiB  ]---
[2817] 07 Apr 16:08:54.596 . sst.c:55 Metadata filenum:0 smallest: key-0 largest: key-999
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 3 [  0 files,   0 bytes]---
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 4 [  0 files,   0 bytes]---
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 5 [  0 files,   0 bytes]---
[2817] 07 Apr 16:08:54.596 . sst.c:51 --- Level 6 [  0 files,   0 bytes]---
[2817] 07 Apr 16:08:54.596 . log.c:46 Removing old log file testdb/si/1.log
[2817] 07 Apr 16:08:54.596 . sst.c:170 Merge successfully completed. Releasing the skiplist
[2817] 07 Apr 16:08:54.596 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
[2817] 07 Apr 16:08:54.597 - sst.c:176 Exiting from the merge thread as user requested
[2817] 07 Apr 16:08:54.597 - file.c:170 Truncating file testdb/si/manifest to 116 bytes
[2817] 07 Apr 16:08:54.598 . log.c:46 Removing old log file testdb/si/2.log
+-------------------------+-------------+--------------------------+-----------------+
|Random-Write (done:10000): 0.000000 sec/op; inf writes/sec(estimated); cost:0.000(sec);
```

Εκτέλεση της εντολής **./kiwi-bench read 10000**

```
9965 searching key-9965
9966 searching key-9966
9967 searching key-9967
9968 searching key-9968
9969 searching key-9969
9970 searching key-9970
9971 searching key-9971
9972 searching key-9972
9973 searching key-9973
9974 searching key-9974
9975 searching key-9975
9976 searching key-9976
9977 searching key-9977
9978 searching key-9978
9979 searching key-9979
9980 searching key-9980
9981 searching key-9981
9982 searching key-9982
9983 searching key-9983
9984 searching key-9984
9985 searching key-9985
9986 searching key-9986
9987 searching key-9987
9988 searching key-9988
9989 searching key-9989
9990 searching key-9990
9991 searching key-9991
9992 searching key-9992
9993 searching key-9993
9994 searching key-9994
9995 searching key-9995
9996 searching key-9996
9997 searching key-9997
9998 searching key-9998
9999 searching key-9999
[2824] 07 Apr 16:11:12.979 . db.c:49 Closing database 0
[2824] 07 Apr 16:11:12.979 . sst.c:415 Sending termination message to the detached thread
[2824] 07 Apr 16:11:12.979 . sst.c:422 Waiting the merger thread
[2824] 07 Apr 16:11:12.979 - sst.c:176 Exiting from the merge thread as user requested
[2824] 07 Apr 16:11:12.980 - file.c:170 Truncating file testdb/si/manifest to 116 bytes
[2824] 07 Apr 16:11:12.983 . log.c:46 Removing old log file testdb/si/0.log
[2824] 07 Apr 16:11:12.983 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+--------------------------+--------------+----------------------------+------------------+
|Random-Read (done:10000, found:10000): 0.000000 sec/op; inf reads /sec(estimated); cost:0.000(sec)
```

Εκτέλεση της εντολής **./kiwi-bench write 10000 8**

**Όσων αφορά την συγκεκριμένη εντολή, μερικές φορές η ενημέρωση των ολικών εργασιών που πρέπει να κάνει, ενημερώνεται στο να βγάλει ως συνολικό αριθμό εργασιών, τον αριθμό εργασιών που εκτελεί κάθε thread. Π.χ. στο συγκεκριμένο παράδειγμα, θα μπορούσε να βγάλει 'done : 1250'.*

```
[2827] 07 Apr 16:14:12.186 - sst.c:870 Range [002ogxkl8941wasi, zztqaileg8idlskk] DOES overlap in level 0. Checking others
[2827] 07 Apr 16:14:12.186 . sst.c:929 Using level 0 for memtable compaction [002ogxkl8941wasi, zztqaileg8idlskk]
[2827] 07 Apr 16:14:12.186 . file.c:200 Creating directory structure: testdb/si/0
[2827] 07 Apr 16:14:12.186 . sst.c:633 Compaction of 1766 [1799554 bytes allocated] elements started
[2827] 07 Apr 16:14:12.187 - sst_builder.c:167 Index block @ offset: 0x227D6 size: 8459
[2827] 07 Apr 16:14:12.187 - sst_builder.c:168 Meta block @ offset: 0x2278E size: 72
[2827] 07 Apr 16:14:12.187 - sst_builder.c:171 Bloom block @ offset: 0x216F2 size: 4252
[2827] 07 Apr 16:14:12.187 - file.c:170 Truncating file testdb/si/0/24.sst to 149801 bytes
[2827] 07 Apr 16:14:12.188 . file.c:65 Mapping of 149801 bytes for testdb/si/0/24.sst
[2827] 07 Apr 16:14:12.188 - sst_loader.c:183 Index @ offset: 141270 size: 8459
[2827] 07 Apr 16:14:12.188 - sst_loader.c:184 Meta Block @ offset: 141198 size: 72
[2827] 07 Apr 16:14:12.188 . sst_loader.c:201 Data size:        136946
[2827] 07 Apr 16:14:12.188 . sst_loader.c:203 Index size:       0
[2827] 07 Apr 16:14:12.189 . sst_loader.c:204 Key size:         28256
[2827] 07 Apr 16:14:12.189 . sst_loader.c:205 Num blocks size:  354
[2827] 07 Apr 16:14:12.189 . sst_loader.c:206 Num entries size: 1766
[2827] 07 Apr 16:14:12.189 . sst_loader.c:207 Value size:       1766000
[2827] 07 Apr 16:14:12.189 . sst_loader.c:210 Filter size:      4252
[2827] 07 Apr 16:14:12.189 . sst_loader.c:211 Bloom offset 136946 size: 4252
[2827] 07 Apr 16:14:12.189 . sst.c:635 Compaction of 1766 elements finished
[2827] 07 Apr 16:14:12.189 - file.c:170 Truncating file testdb/si/manifest to 296 bytes
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 0 [  6 files,   1 MiB  ]---
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:19 smallest: 001mvzpcohr2qctw largest: zzd3e0oj0erqdzzf
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:24 smallest: 002ogxkl8941wasi largest: zztqaileg8idlskk
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:22 smallest: 00ctwonpfec60p23 largest: zzo1pj5hteqtafus
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:21 smallest: 00q85lnswirixwsr largest: zzll6jfkmmvudisv
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:23 smallest: 017w70618zosx5kf largest: zybt4psumeleekex
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:20 smallest: 01fyod6onmt14q0q largest: zzv9qjcuyfl7srsd
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 1 [  1 files,   3 MiB  ]---
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:18 smallest: 001nds6lax0ncabm largest: zzzuu318h6bzoamx
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 2 [  1 files, 339 KiB  ]---
[2827] 07 Apr 16:14:12.189 . sst.c:55 Metadata filenum:0 smallest: 01d99qxk1oa1gxmi largest: zzubyw93wyjl94xf
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 3 [  0 files,   0 bytes]---
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 4 [  0 files,   0 bytes]---
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 5 [  0 files,   0 bytes]---
[2827] 07 Apr 16:14:12.189 . sst.c:51 --- Level 6 [  0 files,   0 bytes]---
[2827] 07 Apr 16:14:12.189 . log.c:46 Removing old log file testdb/si/1.log
[2827] 07 Apr 16:14:12.189 . sst.c:170 Merge successfully completed. Releasing the skiplist
[2827] 07 Apr 16:14:12.189 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
[2827] 07 Apr 16:14:12.190 - sst.c:176 Exiting from the merge thread as user requested
[2827] 07 Apr 16:14:12.190 - file.c:170 Truncating file testdb/si/manifest to 296 bytes
[2827] 07 Apr 16:14:12.192 . log.c:46 Removing old log file testdb/si/2.log
+--------------------------+--------------+----------------------------+------------------+
|Random-Write (done:10000): 0.000100 sec/op; 10000.0 writes/sec(estimated); cost:1.000(sec);
```

Εκτέλεση της εντολής **./kiwi-bench read 10000 8**

Όσων αφορά την συγκεκριμένη εντολή, μερικές φορές αντιμετωπίσαμε πρόβλημα με το **bus error. Ωστόσο, αυτό το δεν γινόταν σε κάθε εκτέλεση της παραπάνω εντολής και συνήθως βελτιωνόταν αν ξανατρέξουμε τον κώδικα με τα ίδια ακριβώς χαρακτηριστικά, απλά με 1 thread.Αν ξανατρέξουμε έπειτα τον κώδικα με τον επιθυμητό αριθμό threads λειτουργεί κανονικά.*

```
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#inz4r89yuyxct6aj
9983 searching vcofbc2ixcyjsz10
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#vcofbc2ixcyjsz10
9984 searching s6a9l9dld03xw46y
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#s6a9l9dld03xw46y
9985 searching 6kjo2bw5ulecqmi8
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#6kjo2bw5ulecqmi8
9986 searching yyn9x6kgxu3zof30
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#yyn9x6kgxu3zof30
9987 searching 5deduh8e8dwo5k2t
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#5deduh8e8dwo5k2t
9988 searching 8v9brziujr0xd94o
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#8v9brziujr0xd94o
9989 searching s8sdvqxt9knkaf49
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#s8sdvqxt9knkaf49
9990 searching gjqe8yeywk1zkvd2
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#gjqe8yeywk1zkvd2
9991 searching av5vs9frzs2zecek
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#av5vs9frzs2zecek
9992 searching lv4zj93ljva9wu2w
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#lv4zj93ljva9wu2w
9993 searching fdixsnvhln7p5bgw
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#fdixsnvhln7p5bgw
9994 searching wq1m5vxvweuiy25j
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#wq1m5vxvweuiy25j
9995 searching 6dno68cx19dwqzyt
[1949] 07 Apr 16:31:25.282 . kiwi.c:326 not found key#6dno68cx19dwqzyt
9996 searching fplrqoccswlwpw61
[1949] 07 Apr 16:31:25.283 . kiwi.c:326 not found key#fplrqoccswlwpw61
9997 searching fjvbxdyochq9mus8
[1949] 07 Apr 16:31:25.283 . kiwi.c:326 not found key#fjvbxdyochq9mus8
9998 searching 9jp6xr8wtziolefq
[1949] 07 Apr 16:31:25.283 . kiwi.c:326 not found key#9jp6xr8wtziolefq
9999 searching 3h7qkck3zg2c00ag
[1949] 07 Apr 16:31:25.283 . kiwi.c:326 not found key#3h7qkck3zg2c00ag
[1949] 07 Apr 16:31:25.283 . db.c:49 Closing database 0
[1949] 07 Apr 16:31:25.283 . sst.c:415 Sending termination message to the detached thread
[1949] 07 Apr 16:31:25.283 - sst.c:176 Exiting from the merge thread as user requested
[1949] 07 Apr 16:31:25.283 . sst.c:422 Waiting the merger thread
[1949] 07 Apr 16:31:25.284 - file.c:170 Truncating file testdb/si/manifest to 188 bytes
[1949] 07 Apr 16:31:25.289 . log.c:46 Removing old log file testdb/si/0.log
[1949] 07 Apr 16:31:25.289 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+---------------------------+---------------+----------------------------+------------------+
|Random-Read (done:10000, found:0): 0.000400 sec/op; 2500.0 reads /sec(estimated); cost:4.000(sec)
```

Όσων αφορά τα πόσα στοιχεία βρήκε ('found : '), λόγω της τυχαιότητας, υπάρχουν πολύ μικρές πιθανότητες να βρει κάποιο από τα στοιχεία. Ωστόσο, αν γίνουν οι παρακάτω αλλαγές στον κώδικα, λειτουργεί σωστά:

**Για την _write_test:*

```
if (r) {
        snprintf(key, KSIZE, "key-%d", i);
        //_random_key(key, KSIZE);
} else {
        //snprintf(key, KSIZE, "key-%d", i);
        _random_key(key, KSIZE);
}
```

**Για την _read_test:*

```
if (r) {
        snprintf(key, KSIZE, "key-%d", i);
        //_random_key(key, KSIZE);
} else {
        //snprintf(key, KSIZE, "key-%d", i);
        _random_key(key, KSIZE);
}
```

Με τις παραπάνω αλλαγές, τα αποτελέσματα που εμφανίστηκαν στο terminal, είναι τα ακόλουθα:

```
9965 searching key-9965
9966 searching key-9966
9967 searching key-9967
9968 searching key-9968
9969 searching key-9969
9970 searching key-9970
9971 searching key-9971
9972 searching key-9972
9973 searching key-9973
9974 searching key-9974
9975 searching key-9975
9976 searching key-9976
9977 searching key-9977
9978 searching key-9978
9979 searching key-9979
9980 searching key-9980
9981 searching key-9981
9982 searching key-9982
9983 searching key-9983
9984 searching key-9984
9985 searching key-9985
9986 searching key-9986
9987 searching key-9987
9988 searching key-9988
9989 searching key-9989
9990 searching key-9990
9991 searching key-9991
9992 searching key-9992
9993 searching key-9993
9994 searching key-9994
9995 searching key-9995
9996 searching key-9996
9997 searching key-9997
9998 searching key-9998
9999 searching key-9999
[2011] 07 Apr 16:38:37.998 . db.c:49 Closing database 0
[2011] 07 Apr 16:38:37.998 . sst.c:415 Sending termination message to the detached thread
[2011] 07 Apr 16:38:37.998 . sst.c:422 Waiting the merger thread
[2011] 07 Apr 16:38:37.998 - sst.c:176 Exiting from the merge thread as user requested
[2011] 07 Apr 16:38:37.998 - file.c:170 Truncating file testdb/si/manifest to 188 bytes
[2011] 07 Apr 16:38:38.000 . log.c:46 Removing old log file testdb/si/0.log
[2011] 07 Apr 16:38:38.000 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+--------------------------+--------------+--------------------------+-----------------+
|Random-Read (done:10000, found:10000): 0.000200 sec/op; 5000.0 reads /sec(estimated); cost:2.000(sec)
```

Η δεύτερη αντιμετώπιση είναι, αν **δεν αφαιρέσουμε** τα στοιχεία που έχουν εισαχθεί στο **δίσκο (testdb)**, κάνουμε αυτές τις αλλαγές στον κώδικα, τόσο στην _write_test όσο και στην _read_test, λόγω του ότι υπάρχουν ήδη στοιχεία μέσα στον δίσκο, θα το βρει όσες φορές το βάλουμε να ψάξει μέσα στον δίσκο.

**Για την _write_test:*

```
for (i = 0; i < count; i++) {
        if (r) {
                strcpy(key,"whrqwrygtdu52tsd");
                _random_key(key, KSIZE);
        } else {
                snprintf(key, KSIZE, "key-%d", i);
        }
}
```

**Για την _read_test:*

```
sk.length = KSIZE;

// searching this specific key to the disk
strcpy(key,"whrqwrygtdu52tsd");

sk.mem = key;
```

Με τις παραπάνω αλλαγές στον κώδικα, το αποτέλεσμα στο terminal, είναι το παρακάτω:

```
9965 searching 3asu4x9r9ohormr2
9966 searching izpk18j51vnvbd35
9967 searching ul5o8k6oyt2v5z4t
9968 searching oj4gidr9y4uf73qr
9969 searching el6tbi7z1f1c4vvz
9970 searching 4p5s8mrcgritkyb5
9971 searching qn4rb1ximokwpllz
9972 searching hwifofiaxqunubya
9973 searching 4ss5zfdr93eou5d1
9974 searching s1767vnar73bosci
9975 searching aatgvcxu51o5x7xv
9976 searching za7wvkcsx5tbnbj4
9977 searching biackxcv5qr84utt
9978 searching u6vvhxd48wmmybgf
9979 searching kwyuj0wuxdsrxrqx
9980 searching oribul5snx4rzqx9
9981 searching c192rv2eelvhir4w
9982 searching 8cxsn9b1wlyli10k
9983 searching s0salle6cgdkx8nc
9984 searching aau4pbbc3z3bru19
9985 searching kk9bbu7tgbjj9wlp
9986 searching xmjs3kuwqox785my
9987 searching f2zw2xf8eohdr98u
9988 searching licu8xhork15fut0
9989 searching mimel7sp2z8je735
9990 searching vlq98xo5nv0tfjjr
9991 searching 8cbjptfxztm36fz7
9992 searching 7f75il1vm7ushkpf
9993 searching m74hq95v9h553a30
9994 searching f0b3ri4jgp1nzx8r
9995 searching u2yq29chxnc6nlc8
9996 searching rd182vy8qp1fsfws
9997 searching 719fgrmjk5fxghvy
9998 searching 12c9307kwepuzsdw
9999 searching zsilpavz5gmsongv
[1842] 07 Apr 16:50:47.022 . db.c:49 Closing database 0
[1842] 07 Apr 16:50:47.022 . sst.c:415 Sending termination message to the detached thread
[1842] 07 Apr 16:50:47.022 - sst.c:176 Exiting from the merge thread as user requested
[1842] 07 Apr 16:50:47.022 . sst.c:422 Waiting the merger thread
[1842] 07 Apr 16:50:47.022 - file.c:170 Truncating file testdb/si/manifest to 80 bytes
[1842] 07 Apr 16:50:47.024 . log.c:46 Removing old log file testdb/si/0.log
[1842] 07 Apr 16:50:47.024 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+--------------------------+--------------+--------------------------+-----------------+
|Random-Read (done:10000, found:10000): 0.000100 sec/op; 10000.0 reads /sec(estimated); cost:1.000(sec)
```

Εκτέλεση της εντολής **./kiwi-bench readwrite 10000 80 8**

Ίδιες αντιμετωπίσεις με προηγουμένως βοηθούν στο να «βρει» και η **readwrite κλειδιά*

```
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#sod9if4krkvbe7r9
1985 searching mj3rlfxn4m4t9ids
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#mj3rlfxn4m4t9ids
1986 searching cgr0m1qjbbkgo1v0
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#cgr0m1qjbbkgo1v0
1987 searching q4i1p5ujyo3xc6vv
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#q4i1p5ujyo3xc6vv
1988 searching td1lkiu2j4oycp4s
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#td1lkiu2j4oycp4s
1989 searching jc0yok7sfgvhdhic
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#jc0yok7sfgvhdhic
1990 searching 0anai72rhgvjvq2k
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#0anai72rhgvjvq2k
1991 searching 8s82imkn86vbdjd3
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#8s82imkn86vbdjd3
1992 searching j6jr3cpbyq0068ql
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#j6jr3cpbyq0068ql
1993 searching 6pdfhosfktx33gwc
[2328] 07 Apr 17:33:32.853 . kiwi.c:331 not found key#6pdfhosfktx33gwc
1994 searching d5amn53sluyyse95
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#d5amn53sluyyse95
1995 searching 9cawq82gs591rbja
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#9cawq82gs591rbja
1996 searching 7jn0uwi6gmufrtaq
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#7jn0uwi6gmufrtaq
1997 searching vqdsolemweddfnes
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#vqdsolemweddfnes
1998 searching c7jxt7tgkd1hchxd
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#c7jxt7tgkd1hchxd
1999 searching xgbrrg4tk7x5kh43
[2328] 07 Apr 17:33:32.854 . kiwi.c:331 not found key#xgbrrg4tk7x5kh43
[2328] 07 Apr 17:33:32.854 . db.c:49 Closing database 0
[2328] 07 Apr 17:33:32.854 . sst.c:415 Sending termination message to the detached thread
[2328] 07 Apr 17:33:32.854 . sst.c:422 Waiting the merger thread
[2328] 07 Apr 17:33:32.854 - sst.c:176 Exiting from the merge thread as user requested
[2328] 07 Apr 17:33:32.854 - file.c:170 Truncating file testdb/si/manifest to 152 bytes
[2328] 07 Apr 17:33:32.855 . log.c:46 Removing old log file testdb/si/0.log
[2328] 07 Apr 17:33:32.855 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+---------------------------+---------------+----------------------------+------------------+
|Random-Read (done:2000, found:0): 0.000000 sec/op; inf reads /sec(estimated); cost:0.000(sec)
[2328] 07 Apr 17:33:32.856 . log.c:46 Removing old log file testdb/si/0.log
[2328] 07 Apr 17:33:32.857 . skiplist.c:57 SkipList refcount is at 0. Freeing up the structure
+---------------------------+---------------+----------------------------+------------------+
|Random-Read (done:2000, found:0): 0.000000 sec/op; inf reads /sec(estimated); cost:0.000(sec)
```