Αναφορά Πρώτης Άσκησης Λειτουργικά Συστήματα :

Μέλη της Ομάδας:
Δημήτρης Αντωνίου ΑΜ:4027
Λάζαρος Κοσμίδης ΑΜ: 4085

Στην αναφορά μας θα βρείτε της αλλαγές που εχούμε κάνει στο κώδικα μας το τι εχούμε προσθέσει καθώς και στιγμιότυπα οθόνης όπου ενημερώνουμε τις μετρήσεις της αποδοσης και εκτυπωνουμε τα αποτελεσματα στην οθονης

Εχουμε προσπαθησει να υλοποιησουμε το 2 βημα της ασκησης δηλαδη (πολλαπλους αναγνωστες και ενας γραφεας σε ολοκληρη τη βαση)

Παρακατω εχουμε τα στιγμιοτυπα οθονης με το κωδικα οπου τα αριθμησουμε πανω απο την εικονα (Στιγμιοτυπο(1),(2) …..(19)) που γραψαμε σε ολη την εργαστηριακη ασκηση και εξηγουμε λεπτομερως το οτι κανει ο κωδικας μας

Κατανομη Στιγμιοτυπων
Τα Στιγμιοτυπο(1,2) → db.c
Τα Στιγμιοτυπο(3,4,5,6,7) → kiwi.c
Το Στιγμιοτυπο(8) → bench.h
Τα Στιγμιοτυπο(9,10,11) → bench.c
Τα Στιγμιοτυπα(12-19) → σταστικα δεδομενα στην γραμμη εντολων
Στιγμιοτυπο(1):

```
8  //----------------------------------------------PROSTHETOS CODE-----------------------------------------------
9  //prototypa gia ta mutex
10 pthread_mutex_t bd_add_mutex = PTHREAD_MUTEX_INITIALIZER;
11 pthread_mutex_t bd_gets_mutex = PTHREAD_MUTEX_INITIALIZER;
12 //----------------------------------------------END OF PROSTHETOS CODE ----------------------------------------
```

Αυτες ειναι οι αλλαγες και οι προσθεσεις που γινανε στο αρχειο db.c
Στο στιμιοτυπο(1) δηλωνουμε και αρχικοποιουμε στατικα 2 mutex τα οποια τα χρησιμοποιηουμε για να κανουμε κλειδαριες στο db_add(write) και την db_get(read).

Στιγμιοτυπο(2):

```
52 int db_add(DB* self, Variant* key, Variant* value)
53 {
54     pthread_mutex_lock(&bd_add_mutex );//kleidaria gia to add (lock)
55 
56     if (memtable_needs_compaction(self->memtable))
57     {
58         INFO("Starting compaction of the memtable after %d insertions and %d deletions",
59              self->memtable->add_count, self->memtable->del_count);
60         sst_merge(self->sst, self->memtable);
61         memtable_reset(self->memtable);
62     }
63     pthread_mutex_unlock(&bd_add_mutex ); //kleidaria gia to add (unlock)
64     return memtable_add(self->memtable, key, value);
65 }
66 
67 int db_get(DB* self, Variant* key, Variant* value)
68 {
69     pthread_mutex_lock(&bd_gets_mutex); //kleidaria gia to get (lock)
70     if (memtable_get(self->memtable->list, key, value) == 1)
71         return 1;
72     pthread_mutex_unlock(&bd_gets_mutex);//kleidaria gia to get (unlock)
73     return sst_get(self->sst, key, value);
74 }
75 
```

Στο στιγμιοτυπο(2) εφαρμοζουμε lock και unlock για την add(54,63) και για την get(69,72) για να προστατεψουμε τις κρισιμες περιοχες ωστε να πετυχουμε ενας γραφεας και πολλους αναγνωστες σε ολη την βαση.

Αυτες ειναι οι αλλαγες και οι προσθεσεις που γινανε στο αρχειο kiwi.c

Στιγμιοτυπο(3):

```c
double cost; //global metavlhth
//Polynhmatikh ylopoihsh ths _write_test
void *_write_test(void *arg)
{
        int i;
        long long start,end;
        Variant sk, sv;
        DB* db;
        struct th *f = (struct th *) arg;
        char key[KSIZE + 1];
        char val[VSIZE + 1];
        char sbuf[1024];

        memset(key, 0, KSIZE + 1);
        memset(val, 0, VSIZE + 1);
        memset(sbuf, 0, 1024);

        db = db_open(DATAS);

        start = get_ustime_sec();
        for (i = 0; i < (f->counter); i++) {
                if (f->a)
                        _random_key(key, KSIZE);
                else
                        snprintf(key, KSIZE, "key-%d", i);
                fprintf(stderr, "%d adding %s\n", i, key);
                snprintf(val, VSIZE, "val-%d", i);

                sk.length = KSIZE;
                sk.mem = key;
                sv.length = VSIZE;
                sv.mem = val;

                db_add(db, &sk, &sv);
                if ((i % 10000) == 0) {
                        fprintf(stderr,"random write finished %d ops%30s\r",
                                        i,
                                        "");

                        fflush(stderr);
                }
        }

        db_close(db);

        end = get_ustime_sec();
        cost = end -start;
```

Στο στιγμιοτυπο(3) εχουμε κανει την cost global μεταβλητη γιατι την χρησιμοποιουν ολες οι συναρτησεις.Εχουμε

μετατρεψει την _write_test σε void συναρτηση οπου δεχεται ενα struct(τυπου th) το οποιο περιεχει τα δεδομενα των νηματων. Η υλοποιηση της ειναι ιδια με της μονες διαφορες να ειναι αυτες που παιρνουν τα δεδομενα του struct φτιαχνοντας ενα βοηθητικο αντικειμενο του.

Στιγμιοτυπο(4):

```
56          printf(LINE);
57          printf("|Random-Write   (done:%ld): %.6f sec/op; %.1f writes/sec(estimated); cost:%.3f(sec);\n"
58                  ,f->counter, (double)(cost /f->counter)
59                  ,(double)(f->counter / cost)
60                  ,cost);
61          return 0;
62  }
63  //Polynhmatikh ylopoihsh ths  _read_test
64  void *_read_test(void *arg)
65  {
66          struct th *g = (struct th *) arg;
67          int i;
68          int ret;
69          int found = 0;
70          long long start,end;
71          Variant sk;
72          Variant sv;
73          DB* db;
74          char key[KSIZE + 1];
75
76          db = db_open(DATAS);
77          start = get_ustime_sec();
78          for (i = 0; i < g->counter; i++) {
79                  memset(key, 0, KSIZE + 1);
80
81                  /* if you want to test random write, use the following */
82                  if (g->a)
83                          _random_key(key, KSIZE);
84                  else
85                          snprintf(key, KSIZE, "key-%d", i);
86                  fprintf(stderr, "%d searching %s\n", i, key);
87                  sk.length = KSIZE;
88                  sk.mem = key;
89                  ret = db_get(db, &sk, &sv);
90                  if (ret) {
91                          //db_free_data(sv.mem);
92                          found++;
93                  } else {
94                          INFO("not found key#%s",
95                                          sk.mem);
96          }
97
98                  if ((i % 10000) == 0) {
99                          fprintf(stderr,"random read finished %d ops%30s\r",
100                                         i,
```

Εχουμε μετατρεψει την _read_test σε void συναρτηση οπου δεχεται ενα struct(τυπου th) το οποιο περιεχει τα δεδομενα των νηματων. Η υλοποιηση της ειναι ιδια με τις μονες διαφορες να ειναι αυτες που παιρνουν τα δεδομενα του struct φτιαχνοντας ενα βοηθητικο αντικειμενο του.

Στιγμιοτυπο(5):

```
                                        "");

                        fflush(stderr);
                }
        }

        db_close(db);

        end = get_ustime_sec();
        cost = end - start;
        printf(LINE);
        printf("|Random-Read    (done:%ld, found:%d): %.6f sec/op; %.1f reads /sec(estimated); cost:%.3f(sec)\n",
                g->counter, found,
                (double)(cost / g->counter),
                (double)(g->counter / cost),
                cost);
        return 0;
}

//------------------------------------PROSTHETOS CODE-------------------------------------------------------
//Synarthsh readwrite gia na kanei taftoxrona kai read kai writes
void *readwrite_test(void *arg){
        long long start,end;
        struct arguments *d = (struct arguments *) arg;
        int i,ret;
        int countread,countwrite;
        Variant sk, sv;
        DB* db;
        int found = 0;
        char key[KSIZE + 1];
        char val[VSIZE + 1];
        char sbuf[1024];
        memset(key, 0, KSIZE + 1);
        memset(val, 0, VSIZE + 1);
        memset(sbuf, 0, 1024);

        double x;
        x = (d->percent)*0.01;
        db = db_open(DATAS);
        countread = (d->counter)*x;
        countwrite = (d->counter) - countread;
        start = get_ustime_sec();



        for (i = 0; i < countread; i++) {
                memset(key, 0, KSIZE + 1);
```

Στιγμιοτυπο(6):

```
150                 if (d->a)
151                         _random_key(key, KSIZE);
152                 else
153                         snprintf(key, KSIZE, "key-%d", i);
154                 fprintf(stderr, "%d searching %s\n", i, key);
155                 sk.length = KSIZE;
156                 sk.mem = key;
157                 ret = db_get(db, &sk, &sv);
158                 if (ret) {
159                         //db_free_data(sv.mem);
160                         found++;
161                 } else {
162                         INFO("not found key#%s",
163                                         sk.mem);
164         }
165
166                 if ((i % 10000) == 0) {
167                         fprintf(stderr,"random read finished %d ops%30s\r",
168                                         i,
169                                         "");
170
171                         fflush(stderr);
172                 }
173         }
174
175         for (i = 0; i < countwrite; i++) {
176
177                 if (d->a)
178                         _random_key(key, KSIZE);
179                 else
180                         snprintf(key, KSIZE, "key-%d", i);
181                 fprintf(stderr, "%d adding %s\n", i, key);
182                 snprintf(val, VSIZE, "val-%d", i);
183
184                 sk.length = KSIZE;
185                 sk.mem = key;
186                 sv.length = VSIZE;
187                 sv.mem = val;
188
189                 db_add(db, &sk, &sv);
190                 if ((i % 10000) == 0) {
191                         fprintf(stderr,"random write finished %d ops%30s\r",
192                                         i,
193                                         "");
```

Στιγμιοτυπο(7):

```
                        fflush(stderr);
                }
        }



        db_close(db);
        end = get_ustime_sec();
        cost = end -start;

        printf(LINE);
        printf("|Random-ReadWrite       (done:%ld): %.6f sec/op; %.1f readwrites/sec(estimated); cost:%.3f(sec);\n"
                ,d->counter, (double)(cost / d->counter)
                ,(double)(d->counter / cost)
                ,cost);
        printf("COUNT:%ld READCOUNT:%d WRITECOUNT:%d PERCENT:%f \n",d->counter,countread,countwrite,d->percent);

        return 0;
}

//------------------------------END OF PROSTHETOS CODE---------------------------------------------
```

Στα στιγμιοτυπα(5,6,7) γινεται η υλοποιηση της readwrite_test.
Η readwrite_test κανει ταυτοχρονα τις λειτουργιες καθε νηματος και ειναι μια void συναρτηση οπου δεχεται ενα struct(τυπου arguments) το οποιο περιεχει τα δεδομενα των νηματων.Στην συνεχεια η readwrite_test δεχεται εναν αριθμο που ειναι το counter(διεργασιες του καθε νηματος), το r και το percent(το ποσοστο των read και write που κανει το καθε νημα).Χρησιμοποιησαμε αυτο το ποσοστο(percent) για να δουμε ποσα countread και ποσα countwrite(δηλαδη read και write) θα κανει η readwrite_test.

Αυτες ειναι οι αλλαγες και οι προσθεσεις που γινανε στο αρχειο bench.h(αρχειο κεφαλιδας)

Στιγμιοτυπο(8):

```
//-------------------------------------PROSTHETOS CODE-----------------------------------------
//strucks poy kratane plhroforia gia ta nhmata
struct arguments{

        long int counter;
        int a;
        double percent;
};
struct th{
        long int counter;
        int a;
};

//-------------------------------------END OFF PROSTHETOS CODE---------------------------------


```

Στο στιγμιοτυπο(8) εχουμε φτιαξει 2 βοηθητικα structs οπου αποθηκευουν της πληροφοριες του καθε νηματος.Το arguments αντιστοιχει στα νηματα της readwrite_test και το th αντιστοιχει στα νηματα της _read_test και _write_test

Αυτες ειναι οι αλλαγες και οι προσθεσεις που γινανε στο αρχειο bench.c

Στιγμιοτυπο(9):

```
#include "bench.h"
#include <pthread.h>

//---------------PROSTHETOS CODE ----------------------
//Topothethsh prototypwn twn synarthsewn pou vriskontai sthn kiwi
void *readwrite_test(void *arg);
void *_write_test(void *arg);
void *_read_test(void *arg);
//--------------END OF PROSTHETOS CODE-----------------
```

Στο στιγμιοτυπο(9) οριζουμε τα πρωτοτυπα των συναρτησεων που χρειαζονται οταν της καλουμε μεσα απο τα νηματα

Στιγμιοτυπο(10):

```
78 int main(int argc,char** argv)
79 {
80         long int count;
81         struct arguments thread_args;
82         long int threads,threads1,threads2;
83         int i;
84         struct th writethreads;
85         struct th readthreads;
86         srand(time(NULL));
87         if (argc < 3) {
88                 fprintf(stderr,"Usage: db-bench <write | read | readwrite > <count>\n");
89                 exit(1);
90         }
91 //Eisagwgh neas leitourgias write me polynhmatismo
92         if (strcmp(argv[1], "write") == 0) {
93                 int i;
94                 writethreads.a = 0; //r
95                 threads1 = atoi(argv[3]);
96                 count = atoi(argv[2]);
97                 _print_header(count);
98                 _print_environment();
99                 if (argc == 5)
100                         writethreads.a = 1;
101                 pthread_t id[threads1];
102                 writethreads.counter = count/threads1;
103                 for(i = 0;i < threads1;i++){
104                         pthread_create(&id[i],NULL,_write_test,(void *) &writethreads);
105                 }
106                 for(i = 0;i < threads1;i++){
107                         pthread_join(id[i],NULL);
108                 }
109 //Eisagwgh neas leitourgias read me polynhmatismo
110         } else if (strcmp(argv[1], "read") == 0) {
111                 int i;
112                 readthreads.a=0;//r
113                 threads2 = atoi(argv[3]);
114                 count = atoi(argv[2]);
115
116                 _print_header(count);
117                 _print_environment();
118                 if (argc == 5)
119                         readthreads.a = 1;
120                 pthread_t id1[threads2];
121                 readthreads.counter = count/threads2;
122                 for(i = 0;i < threads2;i++){
123                         pthread_create(&id1[i],NULL,_read_test,(void *) &readthreads);
124                 }
125                 for(i = 0;i < threads2;i++){
```

Στο στιγμιοτυπο(10,11) οταν δινουμε στο τερματικο να κανει write του προσθεσαμε αλλο input που θα ειναι ο αριθμος των νηματων και αντιστοιχα στο read .Επισης εχουμε προσθεσει και στα 2 αυτα if(το πρωτο if ξεκιναει στην 92 σειρα και το αλλο if στην 110) την δημιουργια πολλαπλων νηματων

αναλογως με το input που θα δωσει ο χρηστης(πολυνηματικη υλοποιηση).Τελος δωσαμε την δυνατοτητα στο χρηστη να μπορει να κανει read και write ταυτοχρονα μεσω της τριτης if (ξεκιναει στην σειρα 132)που βαλαμε

Στιγμιοτυπο(11):

```
126                     pthread_join(id1[i],NULL);
127             }
128     }
129
130 //-------------------------------------PROSTHETOS CODE-----------------------------------------
131 //Eisagwgh neas leitourgia readwrite opou ektelountai parallhla put kai get me vash to input sth grammh entolwn argv
132     else if(strcmp(argv[1], "readwrite") == 0){
133             thread_args.a = 0; //r
134             count= atoi(argv[2]); //count
135             _print_header(count);
136             _print_environment();
137             if (argc == 6)
138                     thread_args.a = 1;
139             if(atoi(argv[4]) > 100){
140                     printf("Wrong input value for percent!");
141                     exit(1);
142             }
143             threads = atoi(argv[3]); //threadds
144             thread_args.percent = atoi(argv[4]); //percent
145             pthread_t tid[threads];
146             thread_args.counter = count/threads;//katanomh
147             for(i = 0;i < threads;i++){
148                     printf("------------------------------------------------------------");
149                     pthread_create(&tid[i],NULL,readwrite_test,(void *) &thread_args);
150             }
151             for(i=0;i<threads;i++){
152                     pthread_join(tid[i],NULL);
153             }
154             printf("all threads finished\n");
155
156     }
157 //-----------------------------------END OF PROSTHETOS CODE-----------------------------------
158
159     else {
160             fprintf(stderr,"Usage: db-bench <write | read | readwrite > <count> <random>\n");
161             exit(1);
162     }
163
164     return 1;
165 }
166
```

Παρακατω εχουμε στιγμιοτυπα οθονης οπου γινεται ενημερωση των μετρήσεων και εκτυπωσης τους οθονης σας στελνουμε την εντολη που εκτελουμε και απο κατω τα σταστικα της που βγαζει στο τελος :

Εδω ειναι που εκτελουμε την πολυνηματικη εντολης της write
Στιγμιοτυπο(12):

```
myy601@myy601lab1:~/kiwi/kiwi-source/bench$ ./kiwi-bench write 100000 1
```

Εδω ειναι τα αποτελεσματα των στατιστικων της write
Στιγμιοτυπο(13):

```
----+
|Random-Write   (done:100000): 0.000030 sec/op; 33333.3 writes/sec(estimated); cost:3.000(sec)
;
```

Εδω ειναι που εκτελουμε την πολυνηματικη εντολης της read

Στιγμιοτυπο(14):

```
myy601@myy601lab1:~/kiwi/kiwi-source/bench$ ./kiwi-bench read 100000 1
```

Εδω ειναι τα αποτελεσματα των στατιστικων της Read
Στιγμιοτυπο(15):

```
----+
|Random-Read    (done:100000, found:100000): 0.000010 sec/op; 100000.0 reads /sec(estimated);
cost:1.000(sec)
```

Εδω ειναι που εκτελουμε την πολυνηματικη εντολης της ReadWrite
Στιγμιοτυπο(16):

```
myy601@myy601lab1:~/kiwi/kiwi-source/bench$ ./kiwi-bench readwrite 100000 1 50
```

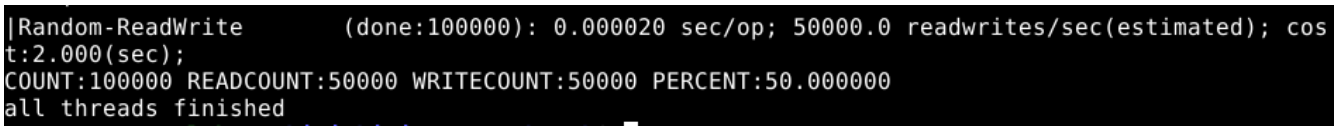
Εδω ειναι τα αποτελεσματα των στατιστικων της ReadWrite

Στιγμιοτυπο(17):

```
|Random-ReadWrite       (done:100000): 0.000020 sec/op; 50000.0 readwrites/sec(estimated); cos
t:2.000(sec);
COUNT:100000 READCOUNT:50000 WRITECOUNT:50000 PERCENT:50.000000
all threads finished
```

Το παρακατω στατιστικο προερχεται απο την εντολη
./kiwi-bench read 100000 5 (οπου 5 ειναι τα νηματα μας)

Στιγμιοτυπο(18):



./kiwi-bench read 100000 2 (οπου 2 ειναι τα νηματα μας)

Στιγμιοτυπο(19):

Παραπανω ηταν καποια ενδεικτικα στατιστικα που σας στειλαμε
μπορειτε και εσεις να τρεξετε της εντολες για να δειτε οτι βγαινουν ιδια τα αποτελεσματα οπως επισης μπορειτε να τρεξετε παρομοιες εντολες

Στιγμιοτυπο(20):

```
myy601@myy601lab1:~/kiwi/kiwi-source$ make
cd engine && make all
make[1]: Entering directory '/home/myy601/kiwi/kiwi-source/engine'
    CC db.o
    CC memtable.o
    CC indexer.o
    CC sst.o
    CC sst_builder.o
    CC sst_loader.o
    CC sst_block_builder.o
    CC hash.o
    CC bloom_builder.o
    CC merger.o
    CC compaction.o
    CC skiplist.o
    CC buffer.o
    CC arena.o
    CC utils.o
    CC crc32.o
    CC file.o
    CC heap.o
    CC vector.o
    CC log.o
    CC lru.o
    AR libindexer.a
make[1]: Leaving directory '/home/myy601/kiwi/kiwi-source/engine'
cd bench && make all
make[1]: Entering directory '/home/myy601/kiwi/kiwi-source/bench'
gcc -g -ggdb -Wall -Wno-implicit-function-declaration -Wno-unused-but-set-variable bench.c k
make[1]: Leaving directory '/home/myy601/kiwi/kiwi-source/bench'
```

Στο στιγμιοτυπο(20) συμπεριλαμβανεται η εξοδος της τελικης εντολης make στο kiwi-source

Στιγμιοτυπο(21):
στα παρακατω στιιγμιοτυπα συμπεριλαμβανουμε της εντολες make που χρησιμοποιησαμε

```
myy601@myy601lab1:~/kiwi/kiwi-source/bench$ make all
gcc -g -ggdb -Wall -Wno-implicit-function-declaration -Wno-unused-but-set-variable bench.c kiwi.
c -L ../engine -lindexer -lpthread -lsnappy -o kiwi-bench
```

```
myy601@myy601lab1:~/kiwi/kiwi-source/bench$ make clean
rm -f kiwi-bench
^[[3~rm -rf testdb
```

Σχολια σχετικα με την ευκολοτερη κατανοηση της εργασιας :

1) Εχουμε προσθεσει και σχολια στον κωδικα για να γινει πιο κατανοητος.
2) Η πολυνηματικη υλοποιηση της write δουλευει μονο για ενα νημα σε αλλες περιπτωσεις μας βγαζει (bus error) και αυτο εχει ως συνεπεια να δουλευει μονο για ενα νημα η readwrite_test