Αρχιτεκτονική υπολογιστών

4η Άσκηση μέρος Α

Αθανάσιος Δελής 03117103

ΓΕΝΙΚΑ

Μελέτη πολυνηματικού κώδικα σε σύστημα προσωμοίωσης και στον υπολογιστή μας με ταυτόχρονη μελέτη ζητημάτων συγχρονισμού και προτοκώλλων συνάφεια της κρυφής μνήμης.

Τιμές ενδιαφέροντος:grain size τεχνική συγχρονισμού , αριθμός νημάτων

Θέλαμε να εξετάσουμε πως επηρεάζαν την χρονική και ενεργειακή απόδοση στο σύστημά μας

ΑΡΧΕΙΟ lock.h

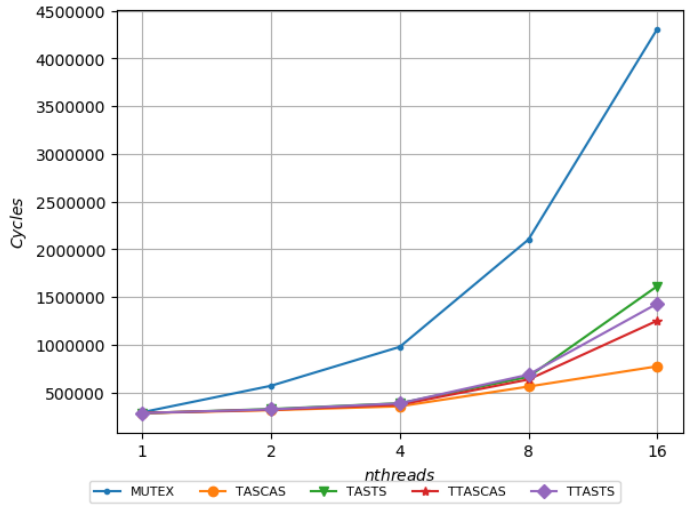
```
#ifndef LOCK_H_
#define LOCK_H_

#include <stdio.h>

typedef volatile int spinlock_t;

#define UNLOCKED  0
#define LOCKED    1

static inline void spin_lock_init(spinlock_t *spin_var)
{
   *spin_var = UNLOCKED;
}

static inline void spin_lock_tas_cas(spinlock_t *spin_var)
{
   while (__sync_val_compare_and_swap(spin_var, UNLOCKED, LOCKED) != UNLOCKED) {}
   // spin until previous was UNLOCKED and then set it to LOCKED
```

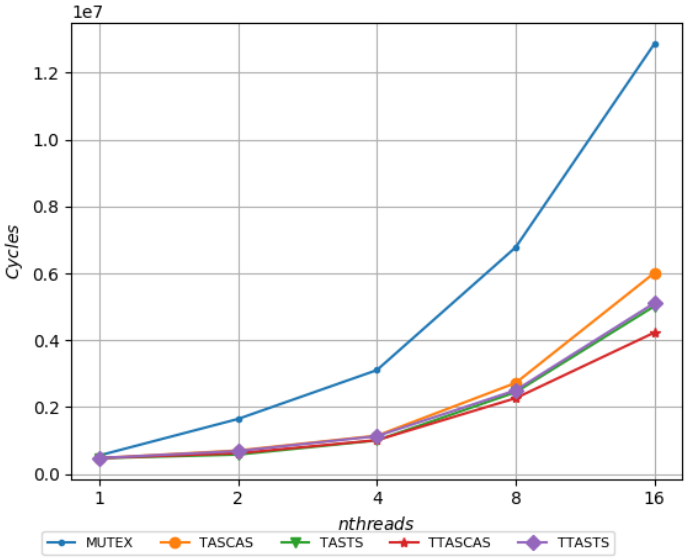
```
}

static inline void spin_lock_ttas_cas(spinlock_t *spin_var)
{
   do {
      while (*spin_var != UNLOCKED) {} // spin until it's UNLOCKED
   } while (__sync_val_compare_and_swap(spin_var, UNLOCKED, LOCKED) != UNLOCKED);
    // if someone else locked it before me start again
}

static inline void spin_lock_tas_ts(spinlock_t *spin_var)
{
   while (__sync_lock_test_and_set(spin_var, LOCKED) != UNLOCKED) {}
   // spin until previous was UNLOCKED, setting it to locked each time
}

static inline void spin_lock_ttas_ts(spinlock_t *spin_var)
{
   do {
      while (*spin_var != UNLOCKED) {} // spin until it's UNLOCKED
   } while (__sync_lock_test_and_set(spin_var, LOCKED) != UNLOCKED);
    // if someone else locked it before me start again
}

static inline void spin_unlock(spinlock_t *spin_var)
{
   __sync_lock_release(spin_var);
}
```

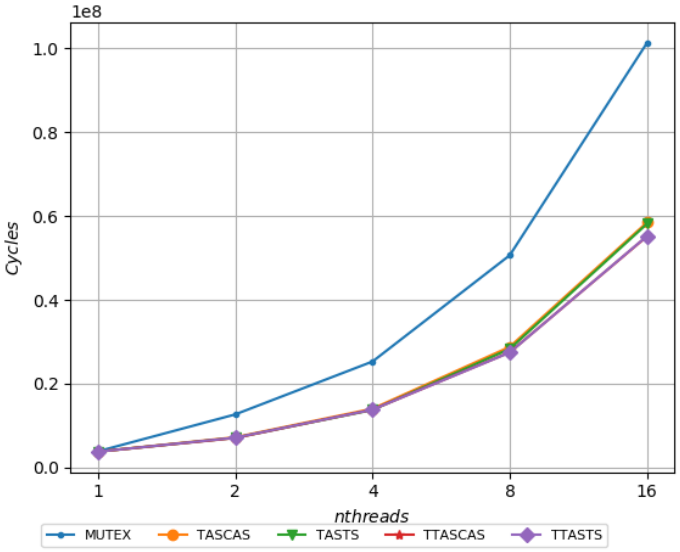
#endif

Χρείαστηκαν 10 μεταβολές του makefile για να παραχθούν τα αντίστοιχα lock αρχεία(εντολές make και make clean). Τα παρακάτω διαγράμματα και οι προσωμοιώσεις έγιναν με την βοήθεια μη δοσμένων και κατασκευασμένων προγραμμάτων(βλέπε ενδεικτικά παράρτημα)

3.1.1


4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
Cycles
1
2
4
8
16
nthreads
MUTEX
TASCAS
TASTS
TTASCAS
TTASTS


1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
1
2
4
8
16
nthreads
MUTEX
TASCAS
TASTS
TTASCAS
TTASTS

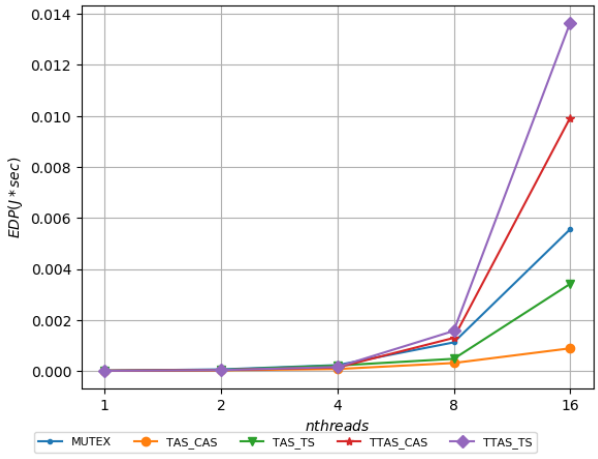

1e8
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
1
2
4
8
16
nthreads
MUTEX
TASCAS
TASTS
TTASCAS
TTASTS

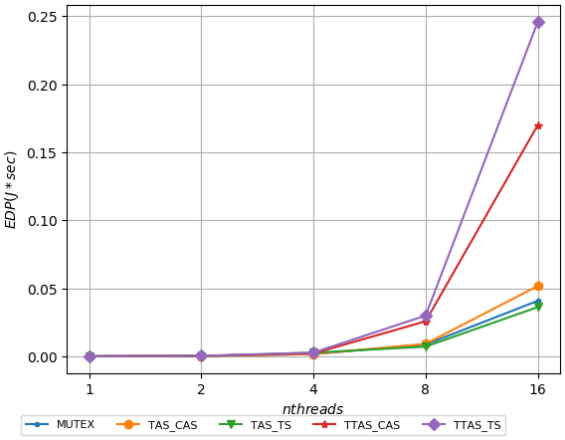
3.1.2

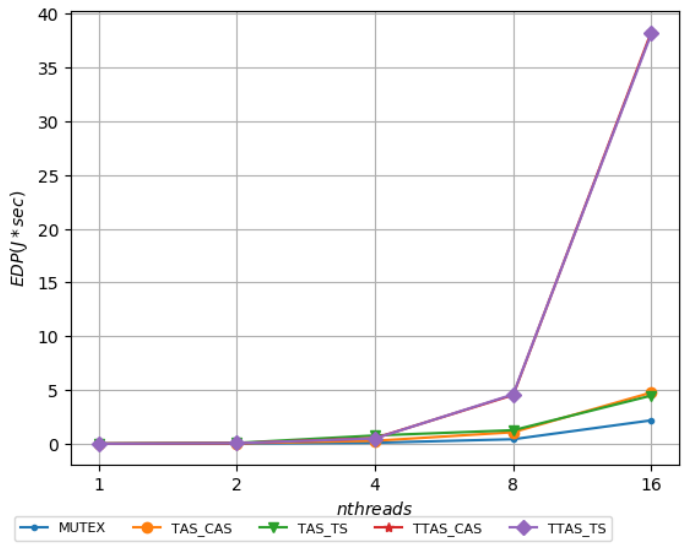
Ο χρόνος εκτέλεσης(σε κύκλους εκτέλεσης)όπως ήταν αναμενόμενο αυξάνεται με την αύξηση των νημάτων, και μάλιστα μη γραμμικά. Η τεχνική MUTEX απαιτούσε το μεγαλύτερο χρόνο εκτέλεση ,ενώ στη θέση του ελάχιστου είδαμε για grain size 1 την TASCAS, 10 την TTASCAS και για 100 την TTASTS. Ωστόσο όλες πλην της Mutex συγκλίναν στις ίδιες τιμές όσο ανέβαινε το grain size και όσο λιγότερα τα threads.

Το grain size επηρέασε το χρόνο εκτέλεσης ως εξής:Από τη μετάβαση 1 σε 10 παρατηρήσαμε διπλασιασμό και από τη μετάβαση 10 σε 100 παρατηρήσαμε δεκαπλασιασμό του χρόνου(τάξη μεγέθους διαγραμμάτων)

3.1.3





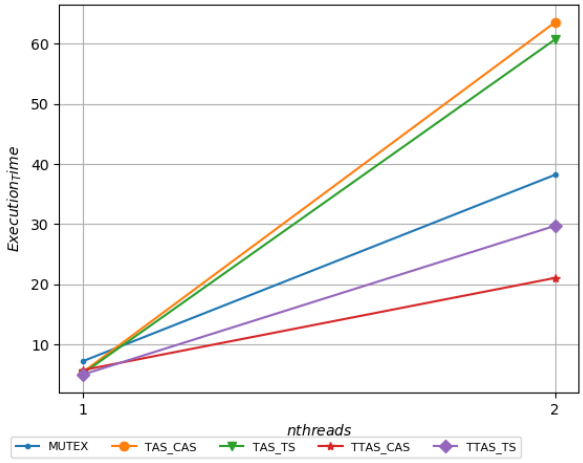


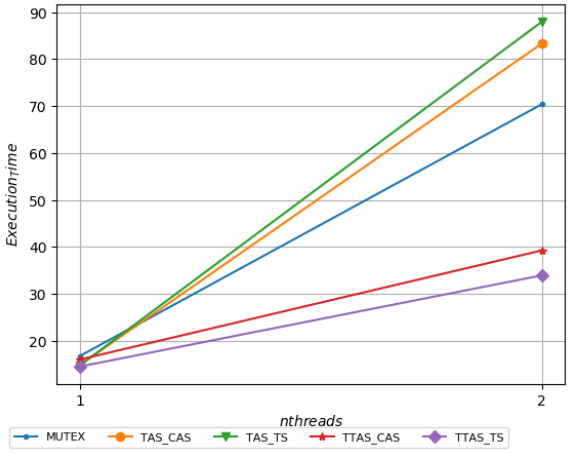
Για αριθμό threads μικρότερο ή ίσο του 4 το Energy Delay Product παρουσιάζει μιδαμινή αύξηση και εν συνεχεία για τιμές μεγαλύτερες του 8 αυξάνει εκθετικά.

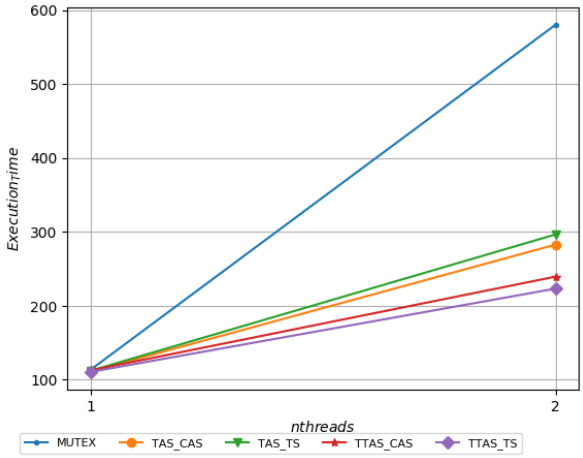
Παρατηρούμε διαφορετικές αναλογίες μεταξύ τεχνικών με την TTAS_TS να έχει το μεγαλύτερο EDP σε όλες τις περιπτώσεις και την MUTEX για grain size 100 τη μικρότερη,TASCAS για 1 και TASTS για 10

3.1.4

Ο χρόνος εκτέλεσης(σε μονάδες χρόνου καθώτι αυτήν την πληροφορία έδινε το σχετικό πρόγραμμα όχι κύκλους) αυξήθηκε γραμμικά με την αύξηση των threads





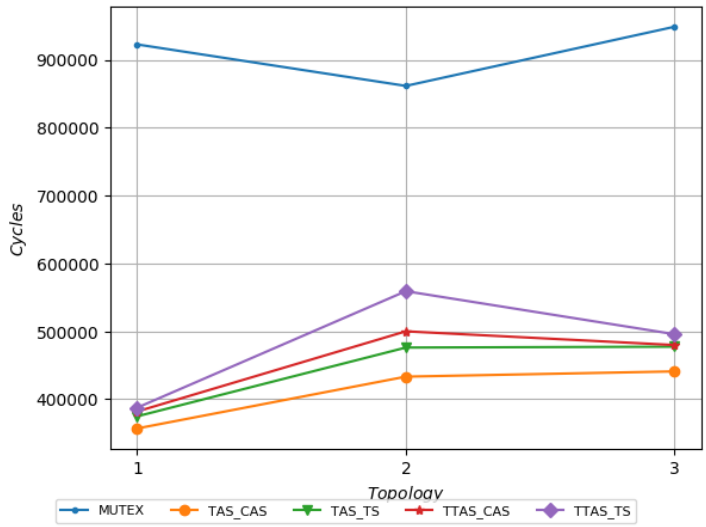


Οι προσομοιώσεις έγιναν για έως 2 threads (όπως αναγέραμε και στην άσκηση 3 διαθέτει Intel core 2 cpu cpu 4400 2.00GHz(ακόμα να βρω πληροφορίες window size για το συγκεκριμένο))

Παρατηρήθηκε γραμμική αύξηση στο χρόνο με την αύξηση των threads και για αλλάγές grain size συντελεστή αλλαγής x1.5 και x50 για 1 σε 10 και 10 σε 100 αντίστοιχα.

3.2.1

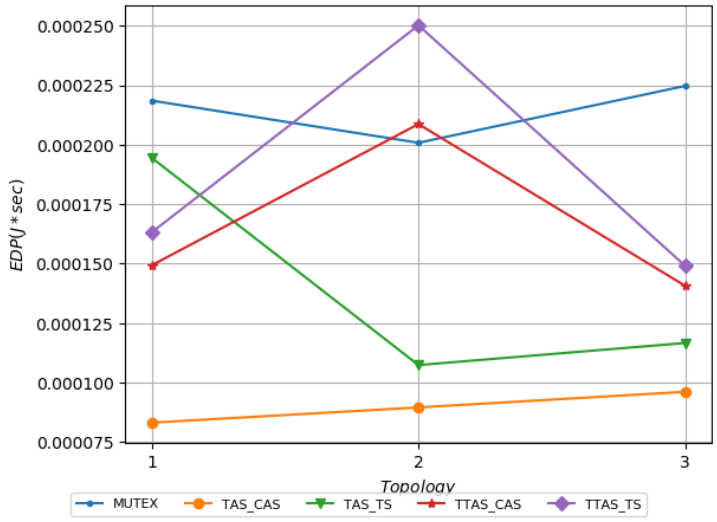
Topologies: share-all, share-L3, share-nothing(σε αντιστοιχία με τα διαγράμματα)



Από share-all σε share-L3:

Όλες οι τεχνικες πλην της mutex οδηγούν σε αύξηση χρόνου καθώς δεν μοιραζόμαστε πλέον την L2

Γενικά παρατηρώντας τα ανά τοπολογία βλέπω ότι η share-all είναι η καλύτερη και η share-L3 η χειρότερη στην κατανάλωση χρόνου



Χειρότερο από άποψη edp είναι σαν τοπολογία η share-L3.

Δεν βρίσκω άλλο μοτίβο με τα παρόντα διαγράμματα αν βρω χρόνο μέχρι την εξέταση θα δοκιμάσω και bar plots μήπως βρω κάποιο εξτρά μοτίβο στα δεδομένα