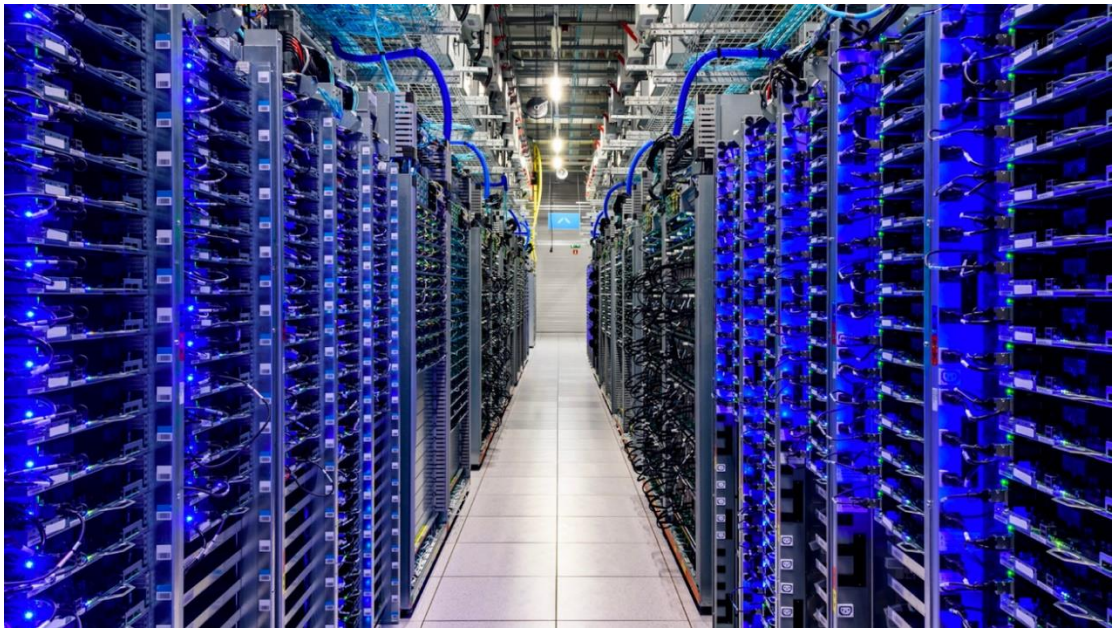
Ροή Υ :: Υπολογιστικά Συστήματα, ΕΜΠ

Βικέντιος Βιτάλης el18803

Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

8ο Εξάμηνο, 3η Σειρά

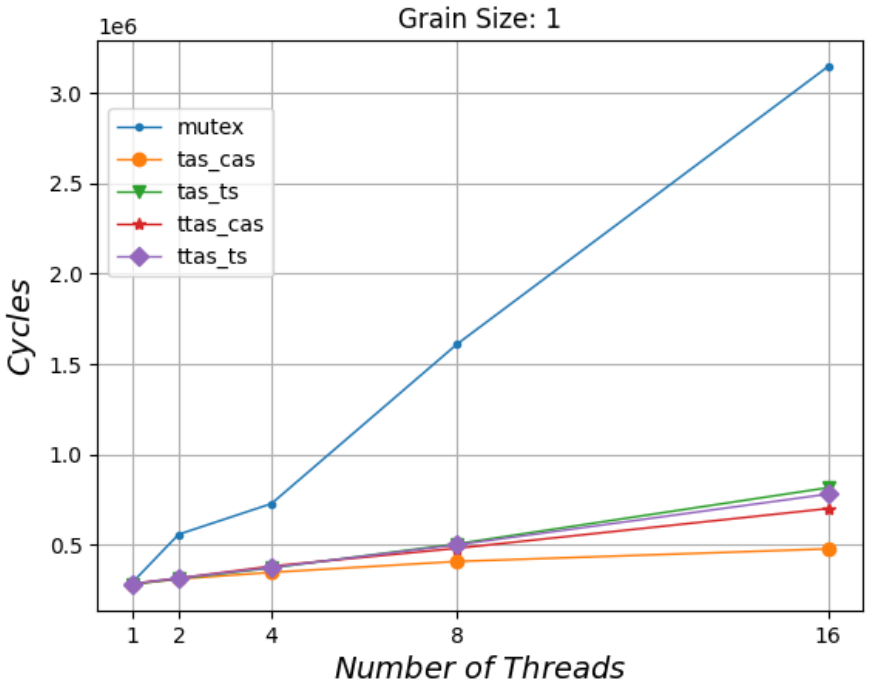


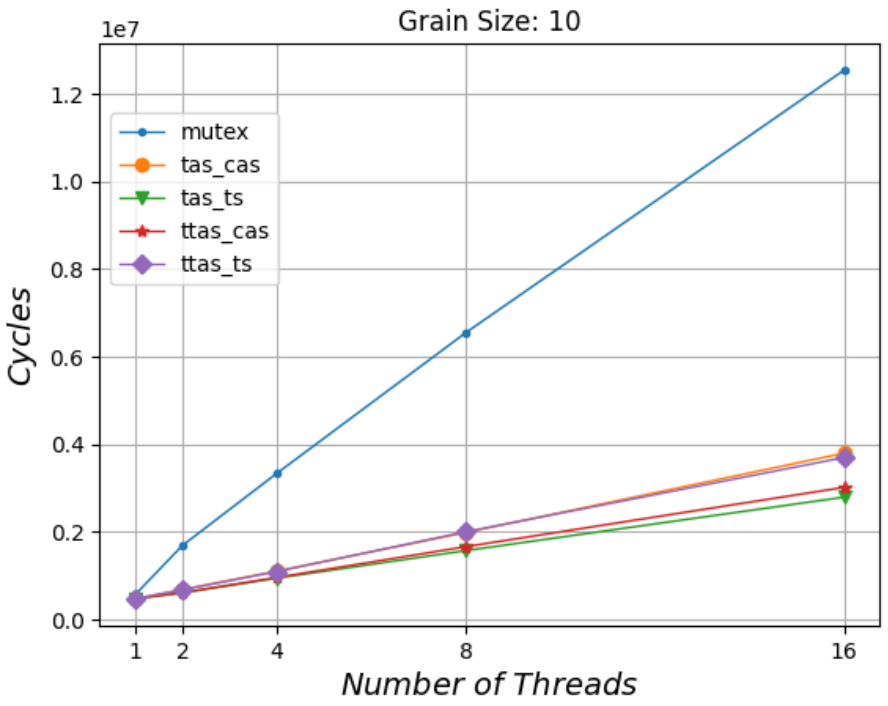
4.1.1. Για κάθε grain size, δώστε το διάγραμμα της κλιμάκωσης του συνολικού χρόνου εκτέλεσης της περιοχής ενδιαφέροντος σε σχέση με τον αριθμό των νημάτων. Συγκεκριμένα, στον x-άξονα θα πρέπει να έχετε τον αριθμό των νημάτων και στον y-άξονα τον χρόνο εκτέλεσης σε κύκλους. Στο ίδιο διάγραμμα θα πρέπει να συμπεριλάβετε τα αποτελέσματα και για τις 5 εκδόσεις.

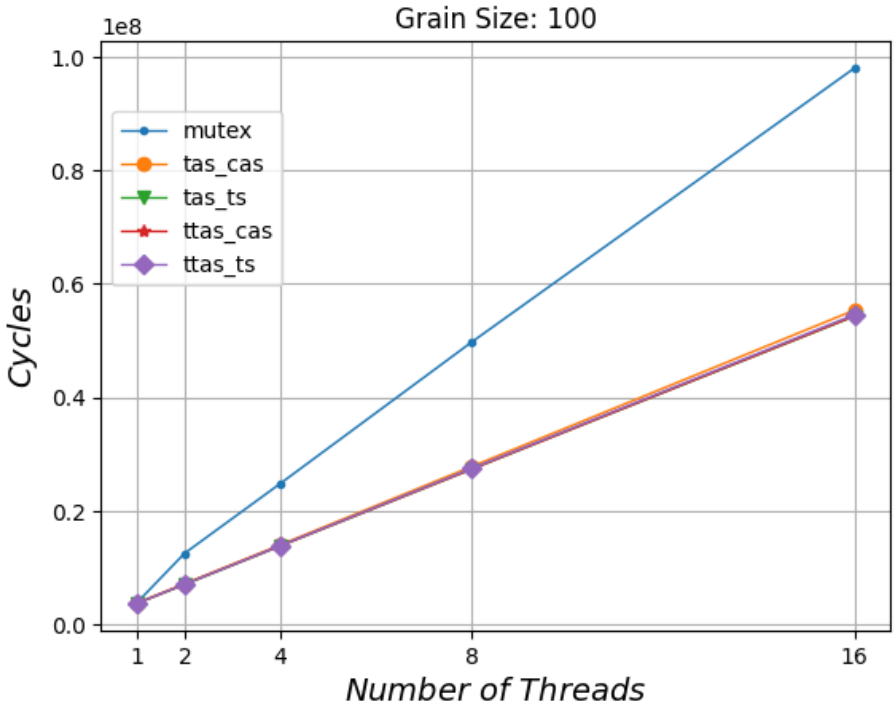
Στο τμήμα αυτό εκτελέστηκαν οι προσομοιώσεις του προγράμματος για τις παρακάτω εκδόσεις προγράμματος:

- TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS, MUTEX
- iterations: 1000
- nthreads: 1, 2, 4, 8, 16 (σε σύστημα με ισάριθμους πυρήνες)
- grain_size: 1, 10, 100

Ακολουθούν τα διαγράμματα που προέκυψαν με βάση τα αρχεία sim.out για τις διαφορετικε παραμέτρους προσομοίωσης:



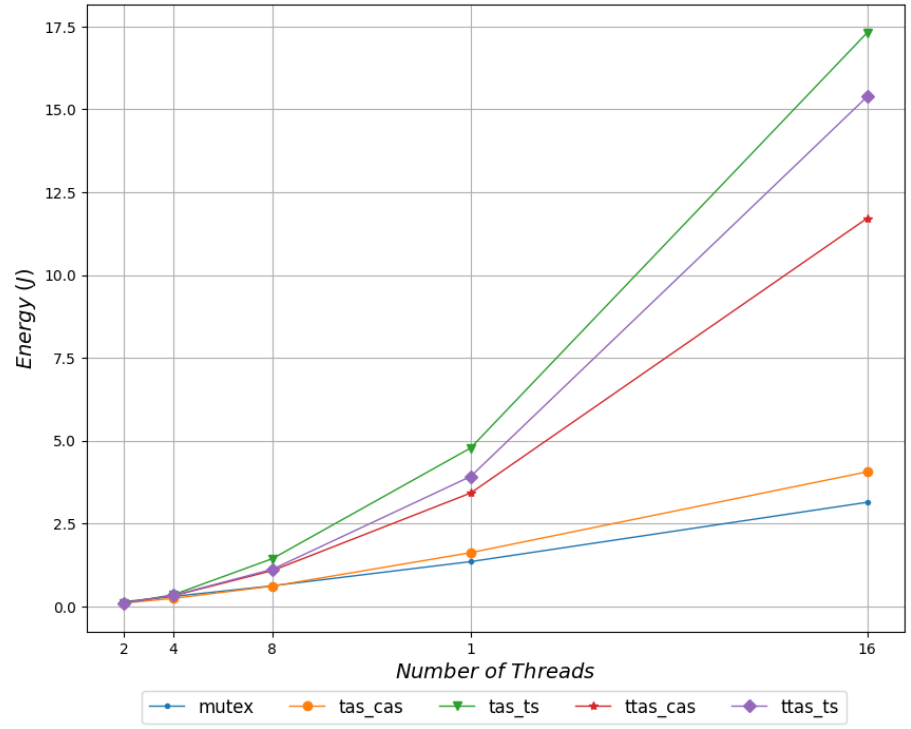

Grain Size: 10
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
1
2
4
8
16
Number of Threads

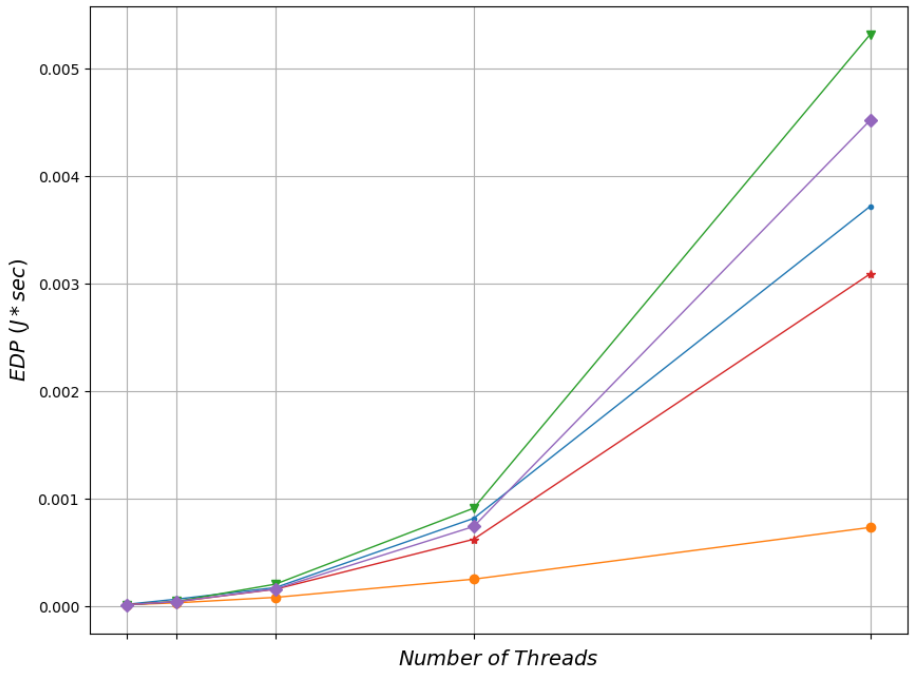

Grain Size: 100
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
1
2
4
8
16
Number of Threads

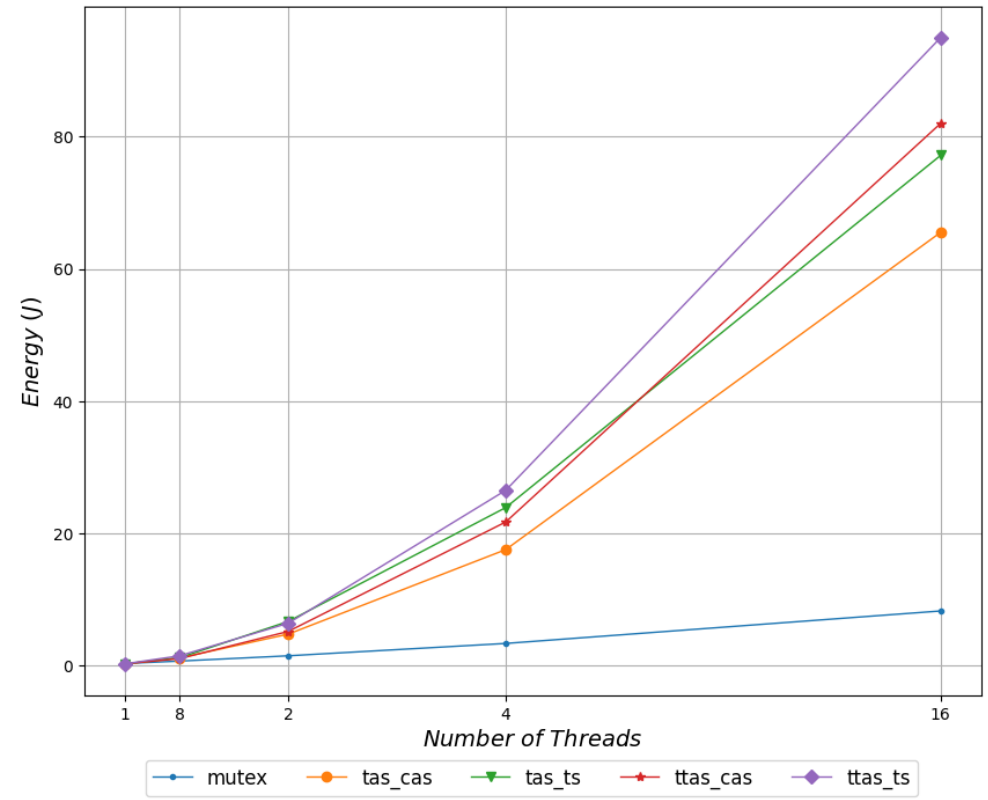
4.1.2. Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με τη φύση της εκάστοτε υλοποίησης; Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με το grain size; Δικαιολογήστε τις απαντήσεις σας.

Οι μηχανισμοί TAS_CAS και TAS_TS υλοποιούν, το Test-and-set πρωτόκολλο, αλλά χρησιμοποιούν διαφορετικό τρόπο υλοποίησης. Επομένως παρουσιάζουν κοντινές επιδόσεις. Όταν δύο ή περισσότεροι επεξεργαστές προσπαθούν να πάρουν το lock οδηγούνται σε κυκλικές εναλλαγές καταστάσεων (Modified – Invalid) της cache line που περιέχει το lock, δημιουργώντας αυξημένη περιττή κίνηση πάνω στο bus. Οι μηχανισμοί TTAS_CAS και TTAS_TS υλοποιούν το πρωτόκολλο Test-and-Test-and-set. Η διαφορά τους με τις Test-and-set υλοποιήσεις είναι ότι δεν γράφουν συνεχώς πάνω στο lock. Εκτελούν πρώτα ένα load για να δουν αν είναι ελεύθερο και μόνο τότε δοκιμάζουν το atomic instruction που το γράφει και προσπαθεί να το δεσμεύσει. Σε περίπτωση μη διαθέσιμου lock οι επεξεργαστές κάνουν spinning τοπικά στην cache τους αποφεύγοντας το άχρηστο broadcasting στον διάδρομο. Αναμένουμε οι μηχανισμοί Testand-Test-and-set να πετυχαίνουν καλύτερη κλιμακωσιμότητα και επίδοση σε σχέση με τους υπόλοιπους. Μελετώντας τα ανωτέρω διαγράμματα, παρατηρούμε πως ο χρόνος της εκτέλεσης (σε κύκλος) αυξάνεται σχεδόν γραμμικά με το πλήθος των νημάτων. Οι ευθείες ωστόσο έχουν διαφορετική κλίση για κάθε διαφορετικό μηχανισμό συγχρονισμού. Για grain size 1 και 100 την μεγαλύτερη κλίση εχει ο μηχανισμός MUTEX, που σημαίνει ότι αυξάνει πολύ πιο γρήγορα ο χρόνος εκτέλεσης σε σχέση με τους άλλους μηχανισμούς. Σε όλα τα grain sizes ωστόσο για το εύρος thread 1 έως 16, παρατηρούμε πως οι μηχανισμοί σε σειρά βελτιούμενης κλιμακωσιμότητας και επίδοσης από άποψη χρόνου εκτέλεσης οι TAS_TS & TAS_CAS και τέλος τα TTAS_TS & TTAS_CAS. Από όλα τα διαγράμματα γίνεται αντιληπτό ότι την βέλτιστη κλιμακωσιμότητα σε όλες τις περιπτώσεις επιτυγχάνει ο μηχανισμός Test-and-Test-and-SET (TTAS) υλοποιημένος είτε με τις ατομικές εντολές test-and-set ή compare-and-swap. Για το grain size παρατηρείται ότι για grain size 10, οι μηχανισμοί TAS για 16 threads πλησιάζουν στην επίδοση τον μηχανισμό MUTEX. Παρατηρούμε πως καθώς αυξάνεται το grain size, τόσο περισσότερο βελτιώνεται και γίνεται γραμμική με χαμηλότερη κλίση η κλιμακωσιμότητα των μηχανισμών. Αυτό γίνεται επειδή πλέον περισσότερος χρόνος για τους υπολογισμούς και λιγότερος χρόνος για τον ανταγωνισμό του lock, οπότε έχουμε καλύτερη επίδοση. Σαφώς βέβαια, αφού αυξάνει το πλήθος των dummy υπολογισμών οι χρόνοι στα αντίστοιχα διαγράμματα μεγαλώνουν κατά μία τάξη μεγέθους (x10) για κάθε δεκαπλασιασμό του grain size.

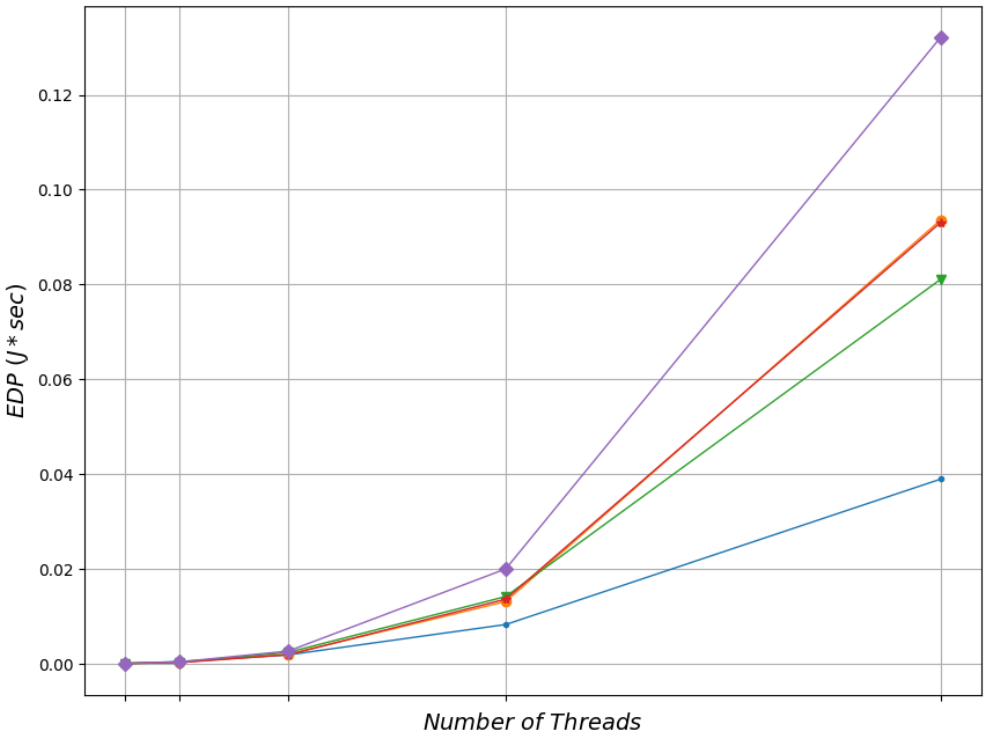
4.1.3. Χρησιμοποιώντας όπως και στην προηγούμενη άσκηση το McPAT, συμπεριλάβετε στην ανάλυση σας εκτός από το χρόνο εκτέλεσης και την κατανάλωση ενέργειας (Energy, EDP κτλ.) Ακολουθούν τα διαγράμματα της ενέργειας αλλά και της μετρικής EDP (Energy Delay Product) για τα προηγούμενα πειράματα:

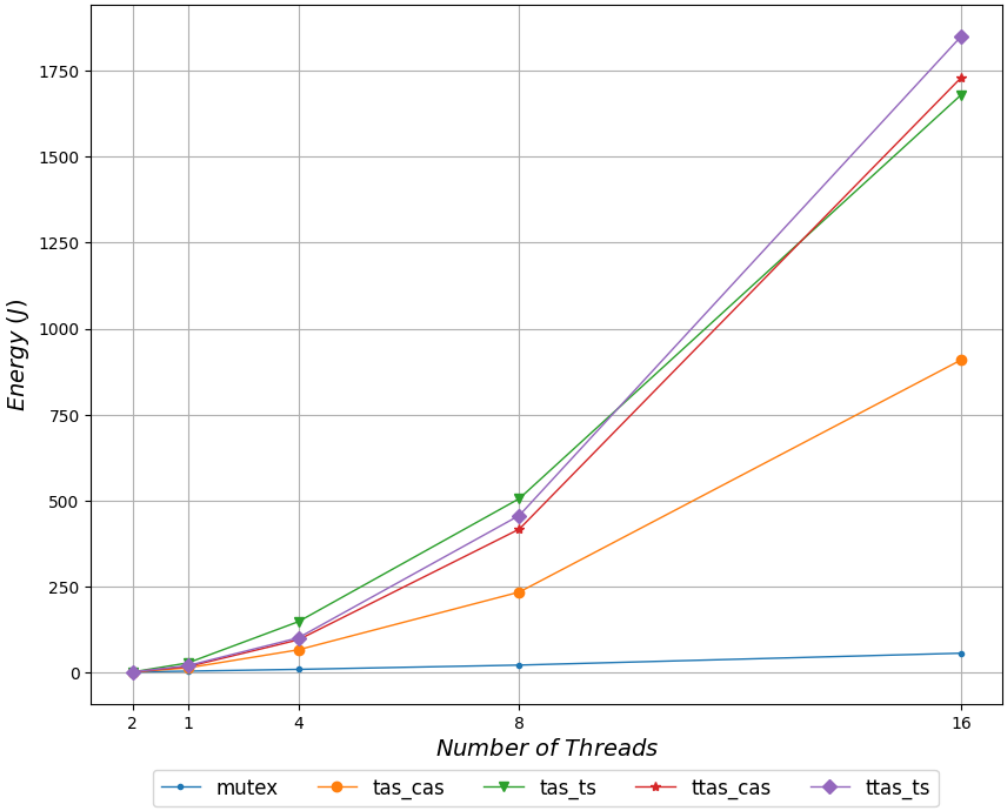




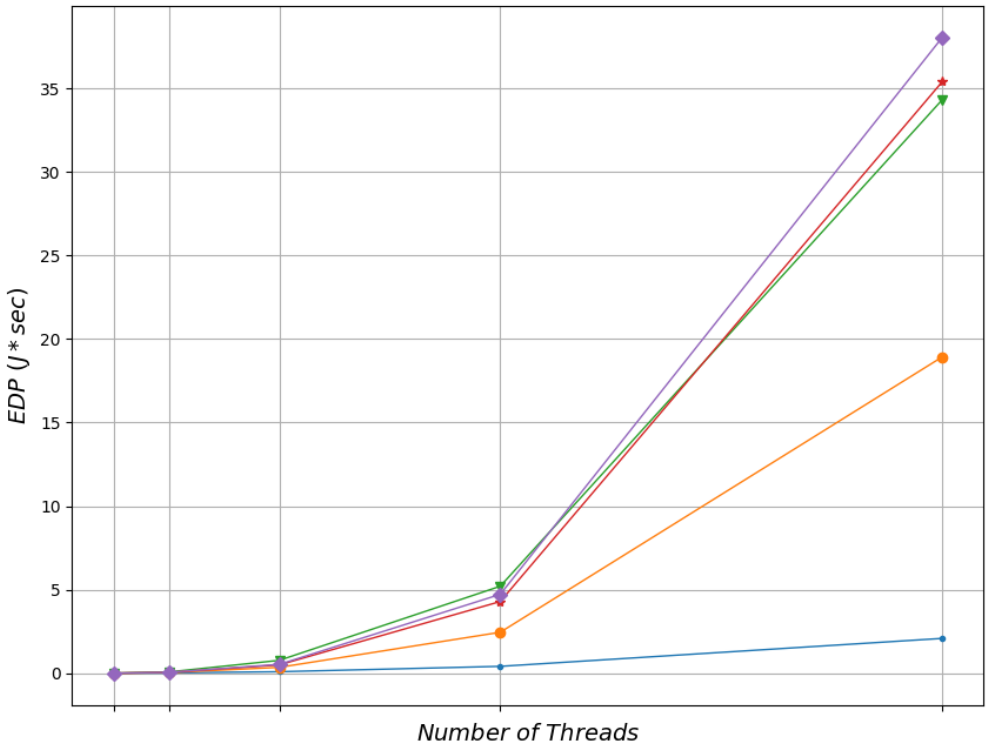


Grain Size: 10





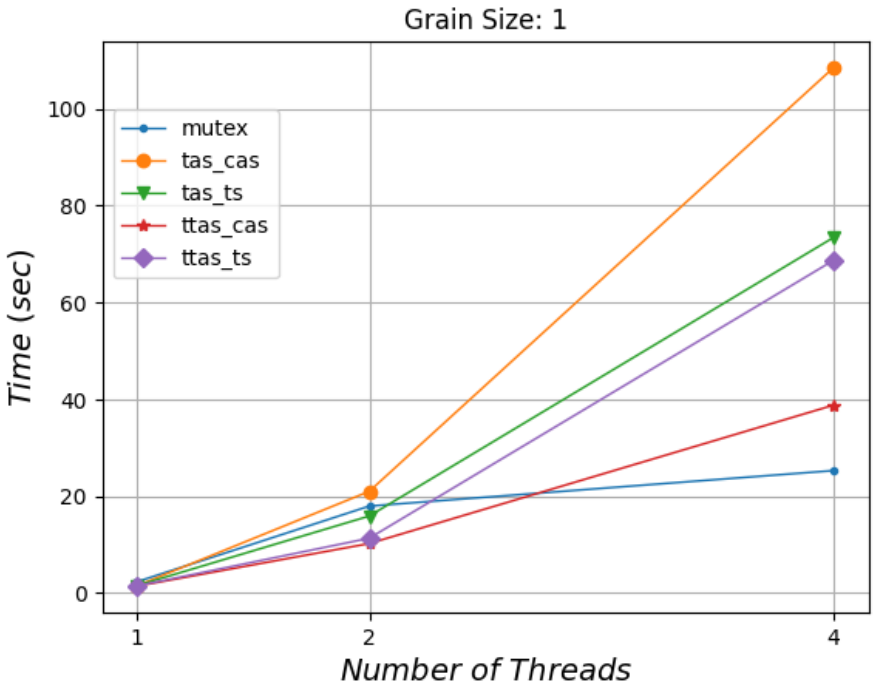
Grain Size: 100

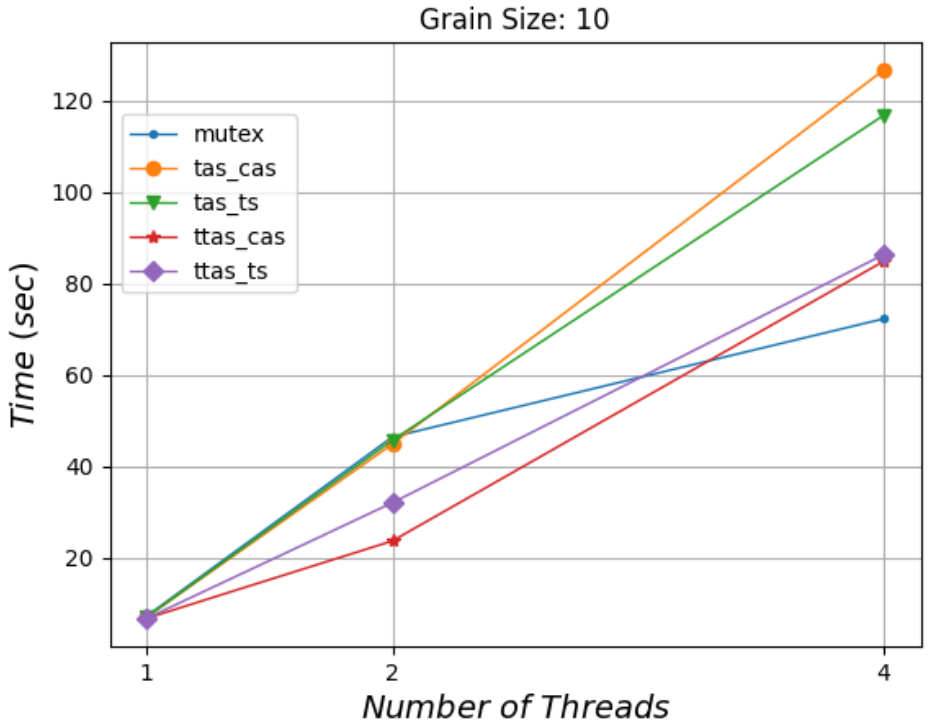


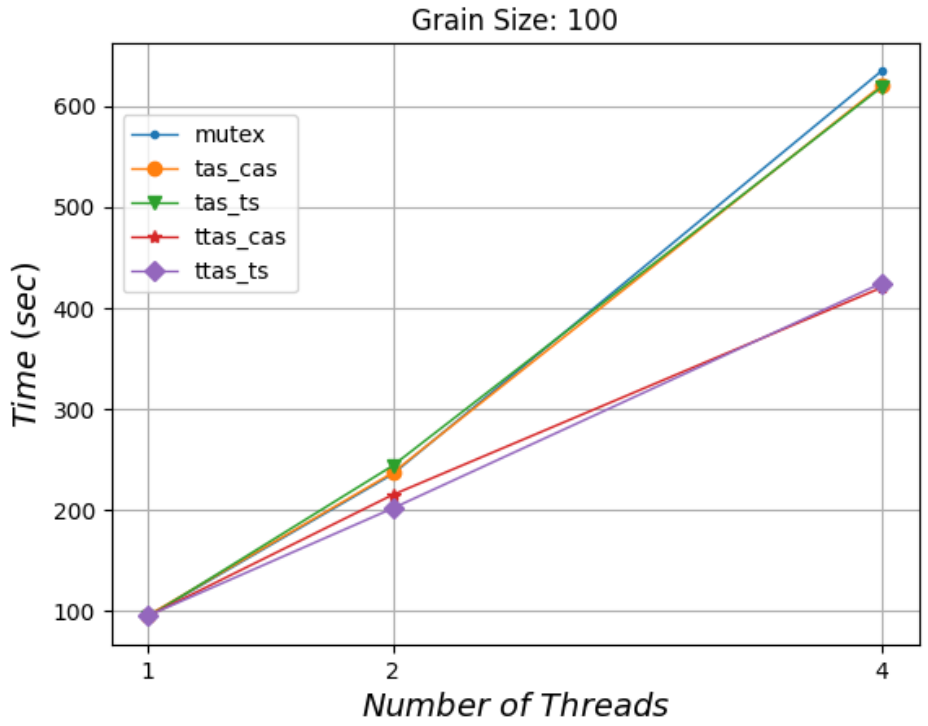
Παρατηρούμε πως η ενέργεια αυξάνεται σχεδόν εκθετικά με το πλήθος των νημάτων. Ειδικότερα όταν το πλήθος αυτό γίνει από 8 νήματα σε 16 η ενέργεια που καταναλώνεται είναι σημαντικά μεγάλη. Επίσης, βλέπουμε πως για μεγαλύτερα grain size 10 και 100, οι μηχανισμοί TTAS_CAS και TTAS_TS σημειώνουν πολύ υψηλότερη κατανάλωση ενέργειας για μεγάλο αριθμό νημάτων σε σχέση με τις υπόλοιπες υλοποιήσεις. Αυτό αιτιολογείται από το γεγονός ότι κάθε νήμα που προσπαθεί να πάρει το κατειλημμένο lock κάνει τοπικά sniping πάνω στην cache του με συνεχόμενα reads τα οποία κοστίζουν ενεργειακά πολύ περισσότερο συγκριτικά με τα τα stalls εξαιτίας cache miss ή κατειλημμένο bus που εμφανίζονται στην περίπτωση των Test-and-set υλοποιήσεων. Σε όλες τις παραπάνω περιπτώσεις φθηνότερο ενεργειακά φαίνεται να είναι ο μηχανισμός TAS. Αναφορικά με τη μετρική EDP, αξίζει να σημειώσουμε ότι για grain size 10 οι καμπύλες τείνουν να ταυτιστούν, και άρα οι μηχανισμοί μπορούν να θεωρηθούν εξίσου καλοί από πλευρά Energy Delay Product. Στη γενική περίπτωση όμως, με αυτό το κριτήριο κερδίζουν οι μηχανισμοί TAS-CAS και TAS-TS. Όσον αφορά το μηχανισμό MUTEX έχει τη χειρότερη επίδοση κρίνοντας με τη μετρική EDP για μεγάλο πλήθος νημάτων (>8), ανεξαρτήτως grain size.

4.1.4. Μεταγλωττίστε τις διαφορετικές εκδόσεις του κώδικα για πραγματικό σύστημα. Εκτελέστε τα ίδια πειράματα με πριν είτε σε ένα πραγματικό σύστημα, που διαθέτει πολλούς πυρήνες, είτε σε ένα πολυπύρηνο VM σε περίπτωση που έχετε πρόσβαση σε κάποιο. Χρησιμοποιήστε τους ίδιους αριθμούς νημάτων (με μέγιστο αριθμό νημάτων ίσο με τον αριθμό των πυρήνων που διαθέτει το μηχάνημά σας) και τα ίδια grain sizes με πριν. Αυτή τη φορά όμως δώστε έναν αρκετά μεγαλύτερο αριθμό επαναλήψεων ώστε ο χρόνος της εκτέλεσης να είναι επαρκώς μεγάλος για να μπορεί να μετρηθεί με ακρίβεια (π.χ. φροντίστε ώστε η εκτέλεση με 1 νήμα να είναι της τάξης των μερικών δευτερολέπτων). Δώστε τα ίδια διαγράμματα με το ερώτημα 4.1.1. Πώς συγκρίνεται η κλιμακωσιμότητα των διαφορετικών υλοποιήσεων στο πραγματικό σύστημα σε σχέση με το προσομοιωμένο; Δικαιολογήστε τις απαντήσεις σας.

Τα προηγούμενα πειράματα εκτελέστηκαν σε πραγματικό σύστημα 4 physical cores με τεχνολογία hyperthreading, συνεπώς με 8 logical cores. Ακολουθούν τα διαγράμματα του χρόνου:




Grain Size: 10
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Time (sec)
120
100
80
60
40
20
1
2
4
Number of Threads


Grain Size: 100
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Time (sec)
600
500
400
300
200
100
1
2
4
Number of Threads

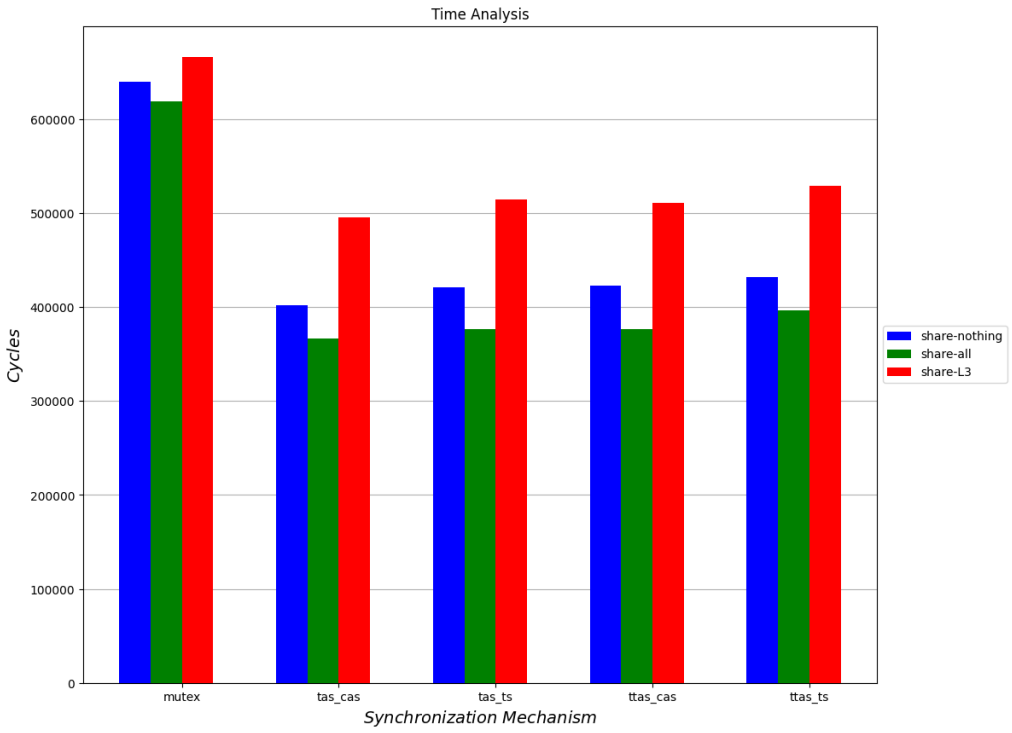
Βλέπουμε πως κι εδώ η κλιμάκωση είναι σχεδόν γραμμική, όπως και τα αποτελέσματα των προσομοιώσεων. Αυτό που είναι αξιοσημείωτο, είναι πως την βέλτιστη επίδοση επιτυγχάνει το TTAS_TS. Μάλιστα, καθώς αυξάνεται το πλήθος των πυρήνων σχεδόν ο χρόνος εκτέλεσης παραμένει σταθερός. Την επόμενη καλύτερη επίδοση σε όλα τα grain sizes παρουσιάζει το TTAS_CAS, δηλαδή ο ίδιος μηχανισμός υλοποιημένος με Compare and Swap atomic instructions. Είναι αξιοσημείωτο επίσης πως καθώς μεταβαίνουμε από τα 4 στα 8 νήματα, οι διαφορετικές υλοποιήσεις του μηχανισμού TAS αποκλίνουν σημαντικά στην επίδοση, με την υλοποιήση του TAS_CAS να εμφανίζει χειρότερη επίδοση από τον TAS_TS. Μάλιστα φαίνεται πως η επίδοση του TAS_CAS στις περισσότερες περιπτώσεις είναι παραπλήσια ή χειρότερη του MUTEX.

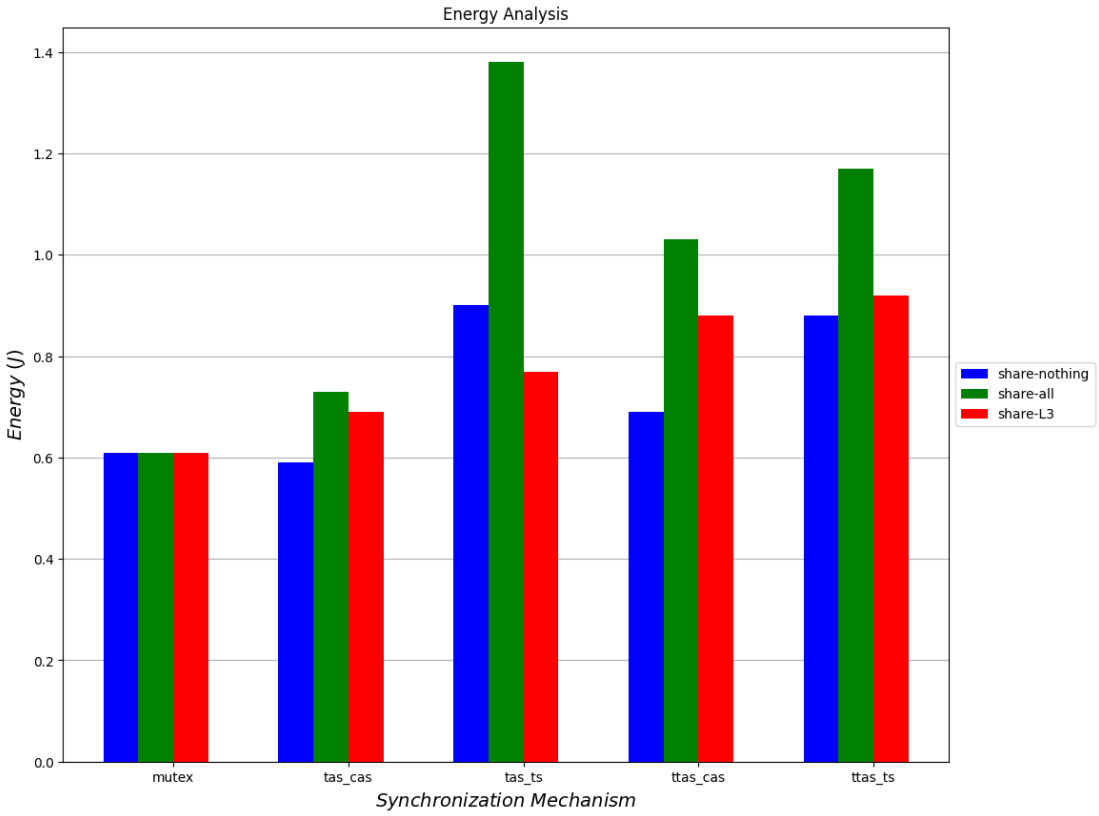
4.2.1. Για τις παραπάνω τοπολογίες, δώστε σε ένα διάγραμμα το συνολικό χρόνο εκτέλεσης της περιοχής ενδιαφέροντος για όλες τις υλοποιήσεις. Χρησιμοποιώντας το McPAT συμπεριλάβετε στην αξιολόγηση σας και την κατανάλωση ενέργειας (Energy, EDP κτλ.). Τι συμπεράσματα βγάζετε για την απόδοση των μηχανισμών συγχρονισμού σε σχέση με την τοπολογία των νημάτων; Δικαιολογήστε τις απαντήσεις σας.

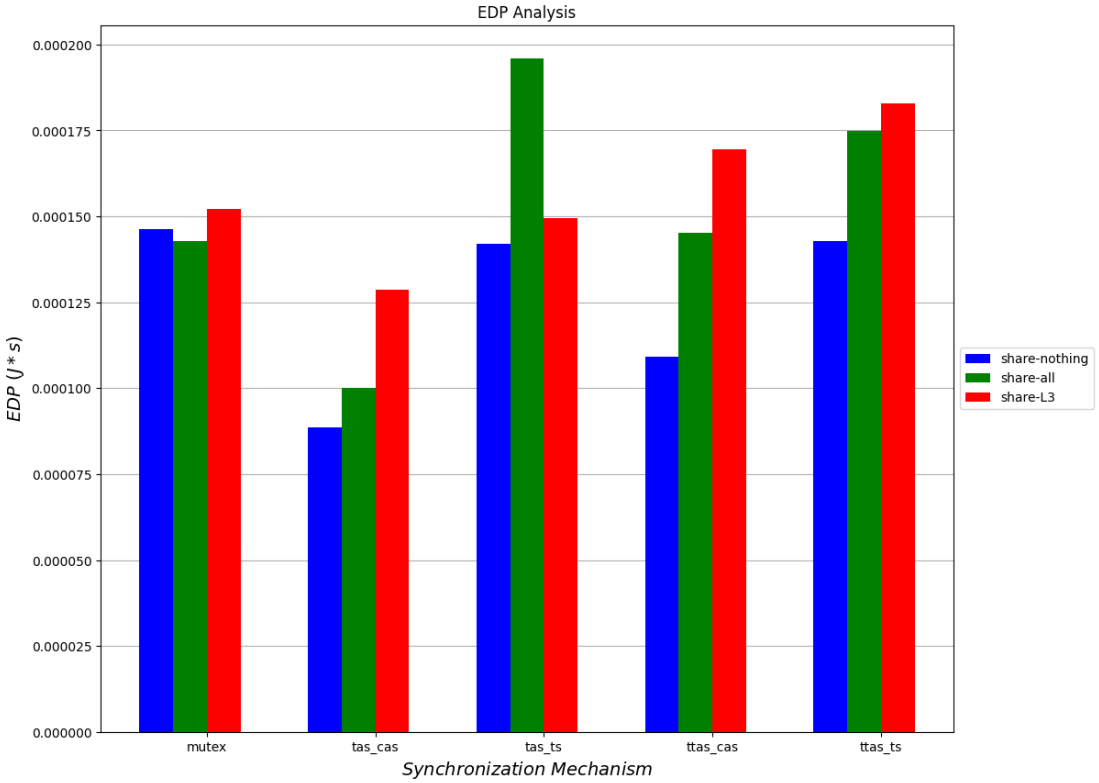
Σε αυτό το τμήμα της άσκησης εξετάζουμε την επίδοση των κάτωθι διαφορετικών τοπολογιών που διαφέρουν μεταξύ τους ως προς τον διαμοιρασμό των κρυφών μνημών. Συγκεκριμένα εξετάζουμε τις ακόλουθες τοπολογίες:

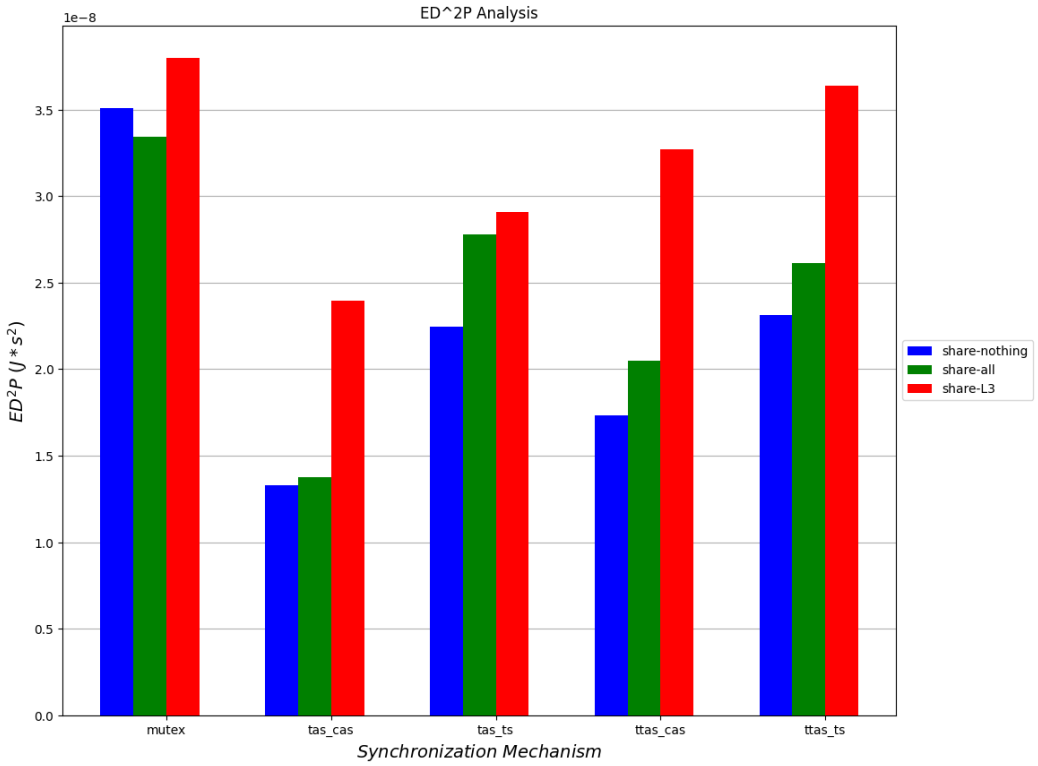
- share-all: και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L2 cache
- share-L3: και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L3 cache, μη κοινή L2
- share-nothing: και τα 4 νήματα βρίσκονται σε πυρήνες με διαφορετική L3 cache

Στη συνέχεια ακολουθούν ραβδογράμματα τα οποία παρουσιάζουν τους χρόνους εκτέλεσης, την κατανάλωση ενέργειας και την μετρική EDP για τα configurations αυτά:




Energy Analysis
Energy (J)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Synchronization Mechanism
share-nothing
share-all
share-L3


EDP Analysis
EDP (J * s)
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Synchronization Mechanism
share-nothing
share-all
share-L3



Όσον αφορά την επίδοση με βάση τη μετρική του χρόνου (ή αντίστοιχα κύκλος εκτέλεσης) για όλους τους υπο μελέτη μηχανισμούς συγχρονισμου, την καλύτερη επίδοση λαμβάνουμε στην τοπολογία share-all. Για τους μηχανισμούς συγχρονισμού MUTEX, TAS_TS, TAS_CAS οι τοπολογίες σε φθίνουσα επίδοση είναι οι share-all, share-L3, share-nothing. Για τους μηχανισμούς συγχρονισμού TTAS_TS, TTAS_CAS οι τοπολογίες σε φθίνουσα επίδοση είναι οι share-all, share-nothing, share-L3. Η βέλτιστη επίδοση του share-all αιτιολογείται από το γεγονός οτι αν επεξεργαστής ζητήσει ένα Invalid cache line θα ενημερωθεί η ιεραρχία μνήμης μέχρι την L2 και θα το διαβάσει από εκεί, ενώ στο share-L3 η invalid cache line θα ζητηθεί από την L3 και για share-nothing φτάνουμε μέχρι και την κύρια μνήμη για κάθε αυξάνοντας σημαντικά τον χρόνο που απαιτείται, αφού όσο πιο ”μακριά” ιεραρχικά βρίσκεται μία μνήμη από τον επεξεργαστή τόσο πιο αργή είναι. Ως προς την κατανάλωση ενέργειας την μεγαλύτερη κατανάλωση έχουμε στην τοπολογία share-nothing, ενώ την μικρότερη κατανάλωση στην τοπολογία share-all. Αντίστοιχα, με κριτήριο την EDP μετρική, την καλύτερη επίδοση έχει η τπολογία share-all.


VIKENTIOS VITALIS