4η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ
Ακ. έτος 2018-2019, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Ονομ/νυμο: Χαρδούβελης Γεώργιος-Ορέστης
Α.Μ: 03115100
Εξάμηνο: 8ο

Στόχος της άσκησης αυτής είναι η εξοικείωση με τους μηχανισμούς συγχρονισμού και τα πρωτόκολλα συνάφειας κρυφής μνήμης (cache coherence protocols) σε σύγχρονες πολυπύρηνες αρχιτεκτονικές.

PIN

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήσαμε τα εργαλείο “PIN” (όπως και στις προηγούμενες ασκήσεις). Χρησιμοποιώντας το PIN μπορούμε και εισάγουμε κώδικας κατά τη διάρκεια της εκτέλεσης των μετροπρογραμμάτων ανάμεσα στις εντολές της εφαρμογής αυτής και συλλέγουμε την επιθυμητές πληροφορίες που αφορούν την εκτέλεση τους.

Ο προσομοιωτής που θα αξιοποιήσει το εργαλείο PIN είναι ο Sniper Multicore Simulator.

McPAT

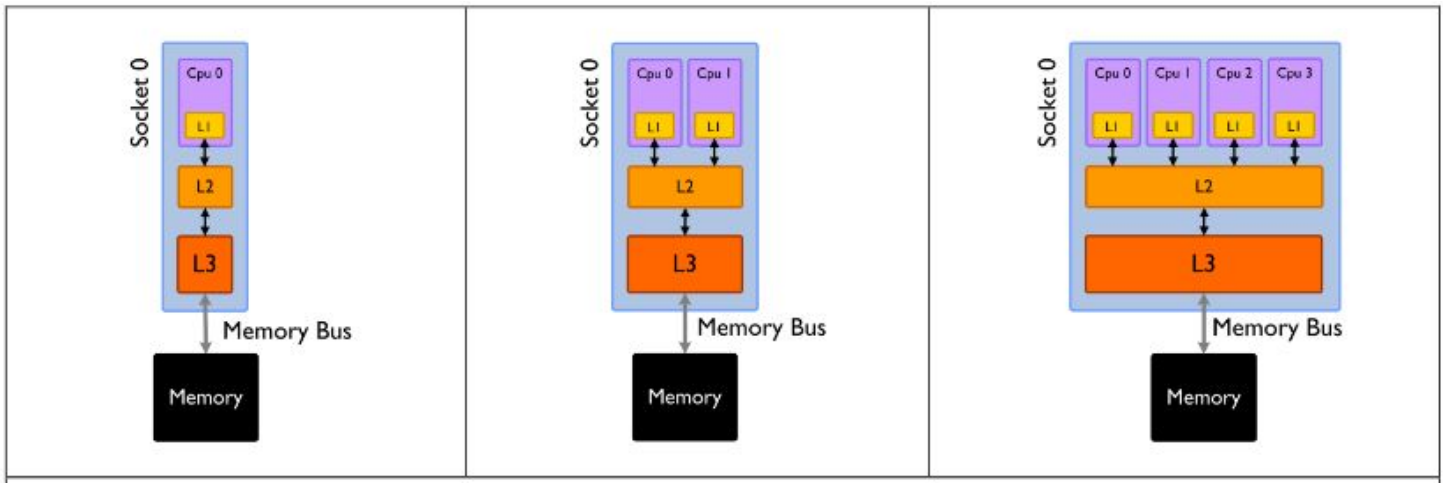
Το McPAT (Multi-core Power, Area, Timing) είναι ένα εργαλείο που συμπεριλαμβάνεται στο Sniper και θα το χρησιμοποιήσουμε με στόχο να πάρουμε τα ζητούμενα στατιστικά από τις προσομοιώσεις. Ουσιαστικά μοντελοποιεί τα χαρακτηριστικά ενός επεξεργαστή, όπως η κατανάλωση ενέργειας και το μέγεθος που καταλαμβάνουν στο τσιπ οι διαφορετικές δομικές μονάδες του επεξεργαστή

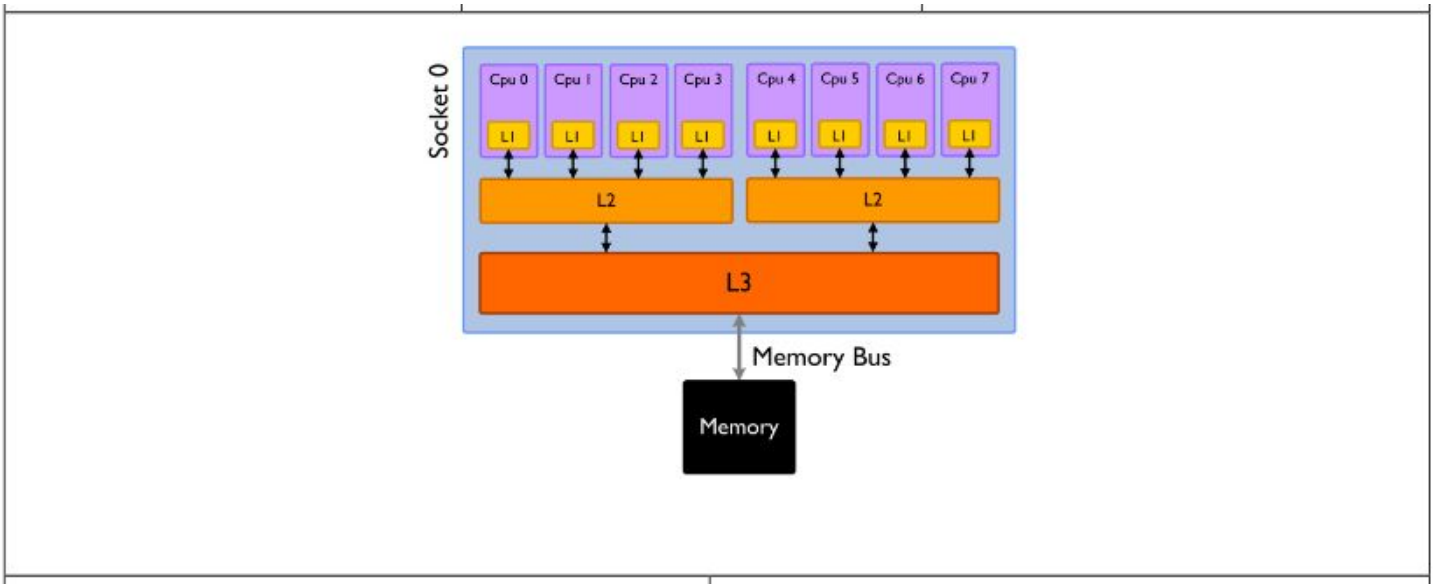
3.1 Σύγκριση υλοποιήσεων

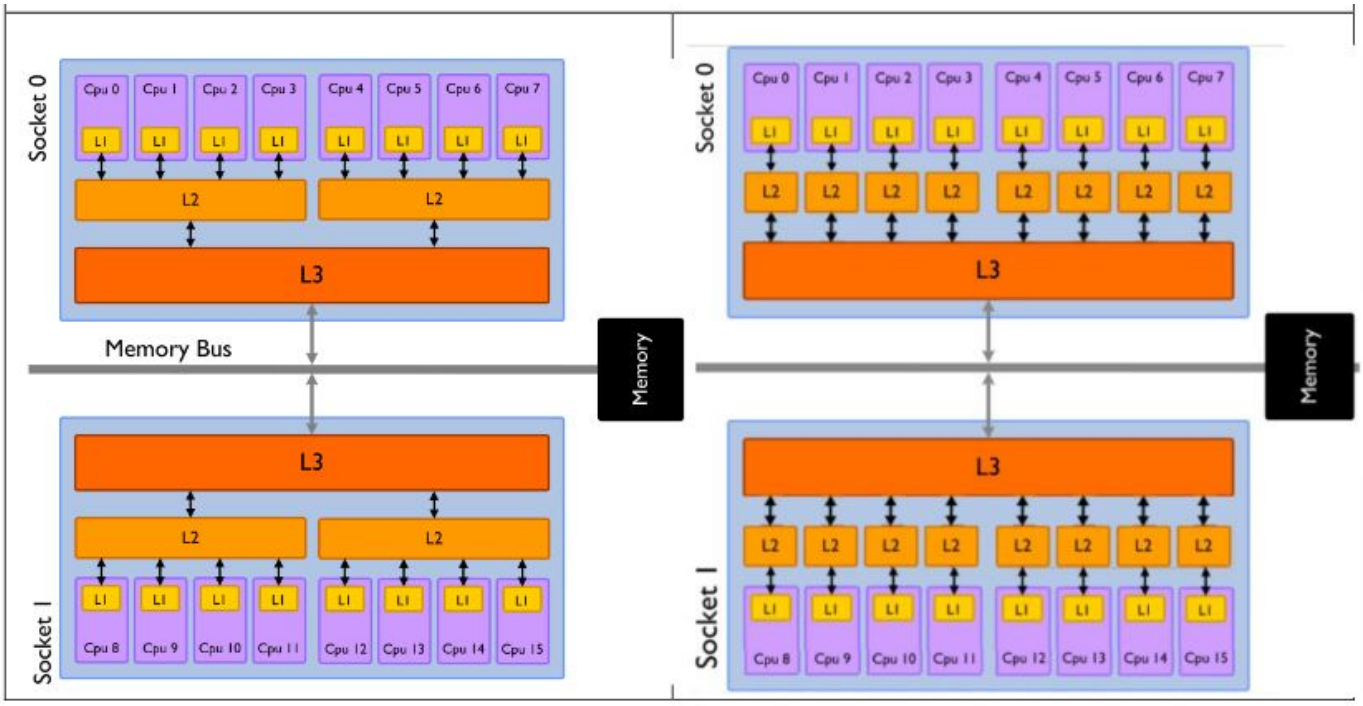
Στην άσκηση υλοποιήθηκαν οι μηχανισμοί κλειδώματος Test-and-Set (TAS) και Test-and-Test-and-Set (TTAS).

Με τον όρο κλιμακωσιμότητα (scalability) εννοούμε πόσο καλά αποδίδει ένα παράλληλο πρόγραμμα καθώς αυξάνεται ο αριθμός των νημάτων από τα οποία αποτελείται.
Ζητούμενο του ερωτήματος αυτού είναι η αξιολόγηση και σύγκριση της κλιμακωσιμότητας των μηχανισμών TAS_CAS, TAS_TS, TTAS_CAS,

TTAS_TS και Pthread Mutex για αριθμούς νημάτων 1, 2, 4, 8, 16. Για το λόγο χρησιμοποιούμε προσομοιωμένα πολυπύρηνα συστήματα αντίστοιχου πλήθους πυρήνων, με διάφορες πολιτικές διαμοιρασμού των caches. Κάθε τέτοιο σύστημα χρησιμοποιεί το MSI πρωτόκολλο συνάφειας κρυφής μνήμης.
Οι τοπολογίες έχουν ως εξής:



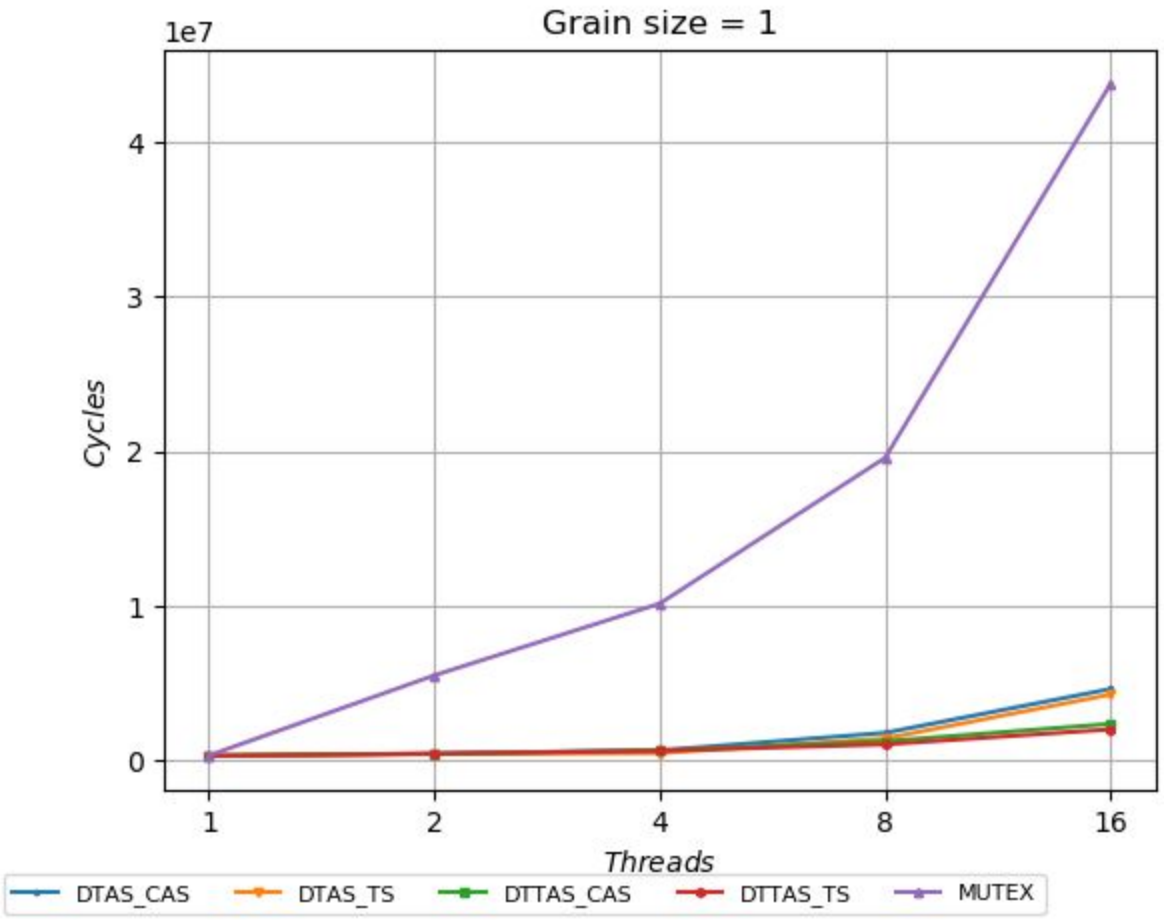


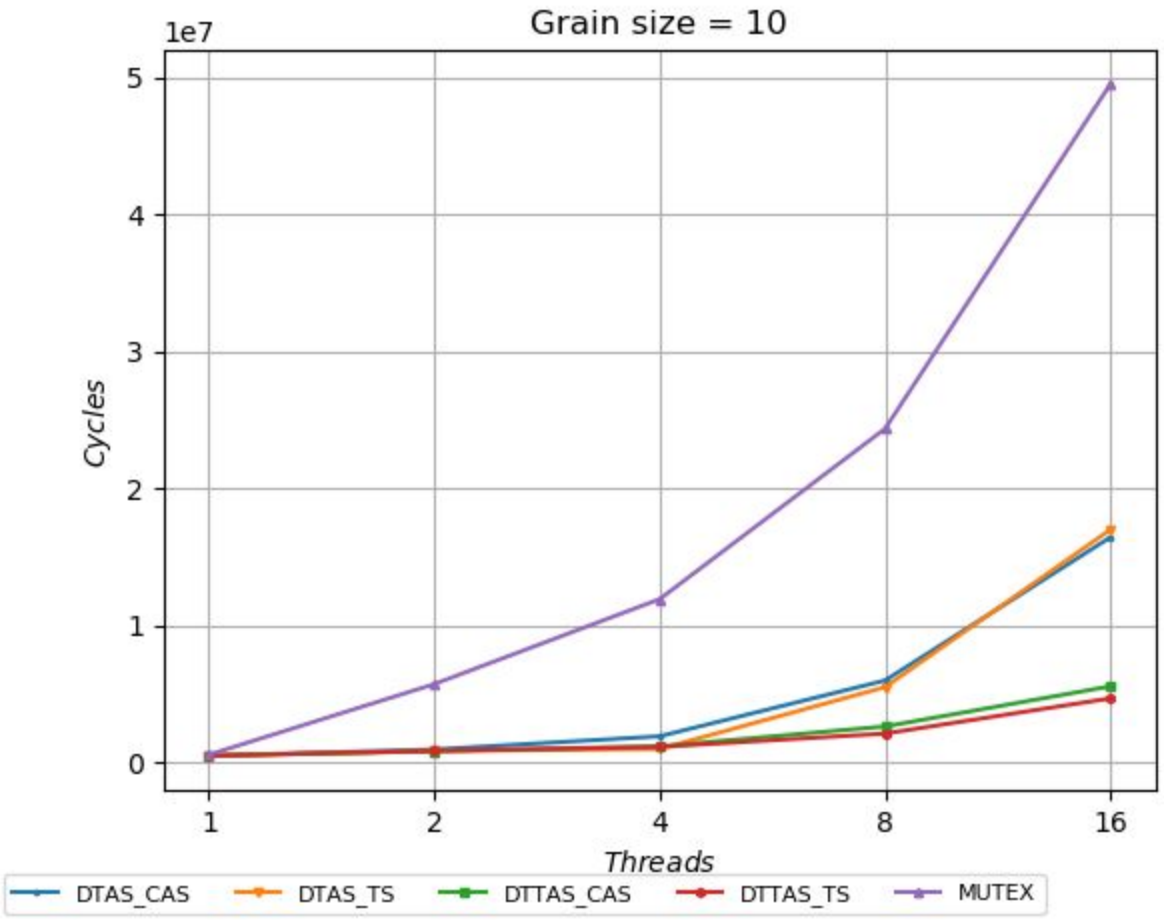


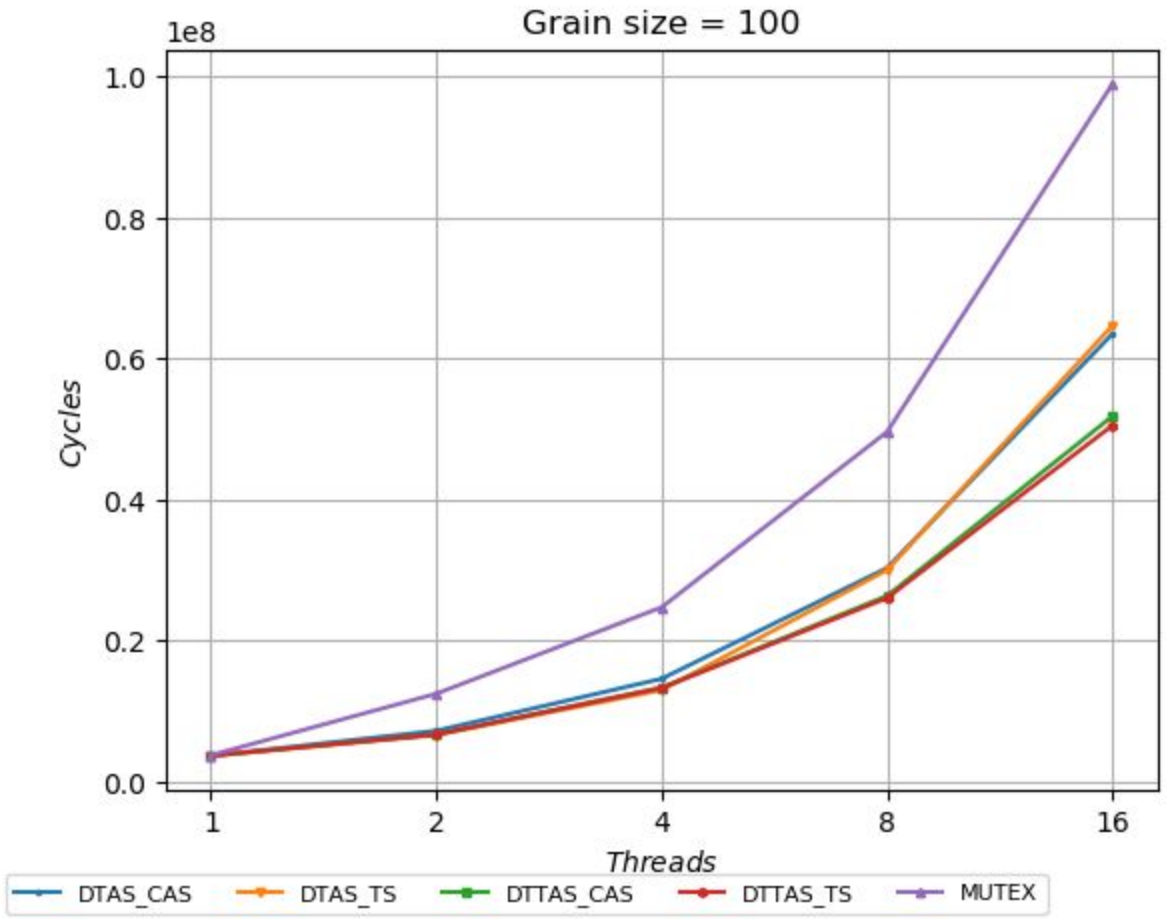
Για τα συστήματα αυτά, έχουν εκτελεστεί προσομοιώσεις του προγράμματος για τους ακόλουθους συνδυασμούς:

- 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

3.1.1. ***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.***


Grain size = 1
1e7
4
3
2
1
0
Cycles
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX


Grain size = 10
1e7
5
4
3
2
1
0
Cycles
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX



3.1.2. ***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.***

Από τα παραπάνω εύκολα βλέπουμε πως ο μηχανισμός mutex έχει την καλύτερη κλιμακωσιμότητα σε σχέση με τους υπόλοιπους. Αυτό συμβαίνει αφού δεν εκτελεί busy-wait αλλά ξυπνάει μόνο όταν ανοίγει το lock. Έτσι συνολικά απαιτούνται λιγότερα cpu. Παρόλα αυτά απαιτεί περισσότερο χρόνο από τους υπόλοιπους μηχανισμούς.

Παράλληλα, η tas είναι η χειρότερη και σε κλιμακωσιμότητα, αφού προσπαθώντας να γράφει το lock συνέχεια, αυξάνεται η κίνηση στο bus και

τα coherence misses, και οι cache lines του ενός νήματος ακυρώνονται από άλλο.

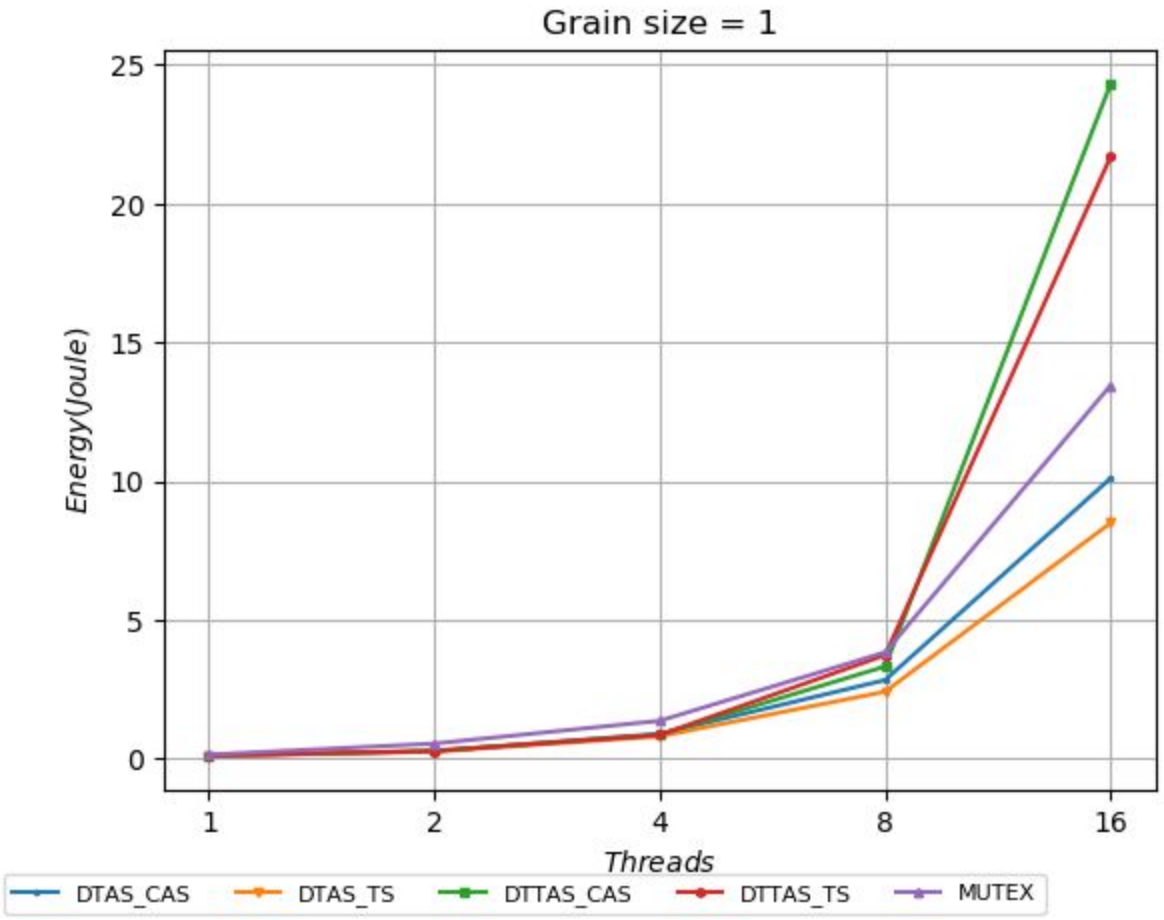
Συνολικά, τα παραπάνω είναι αναμενόμενα αποτελέσματα αφού το mutex επηρεάζεται από την ουρά εργασιών που δεν μεταβάλλεται με το grain size, ενώ όσο αυξάνεται το roi - region of interest-, αυξάνεται και το βάρος στις busy-wait τεχνικές.

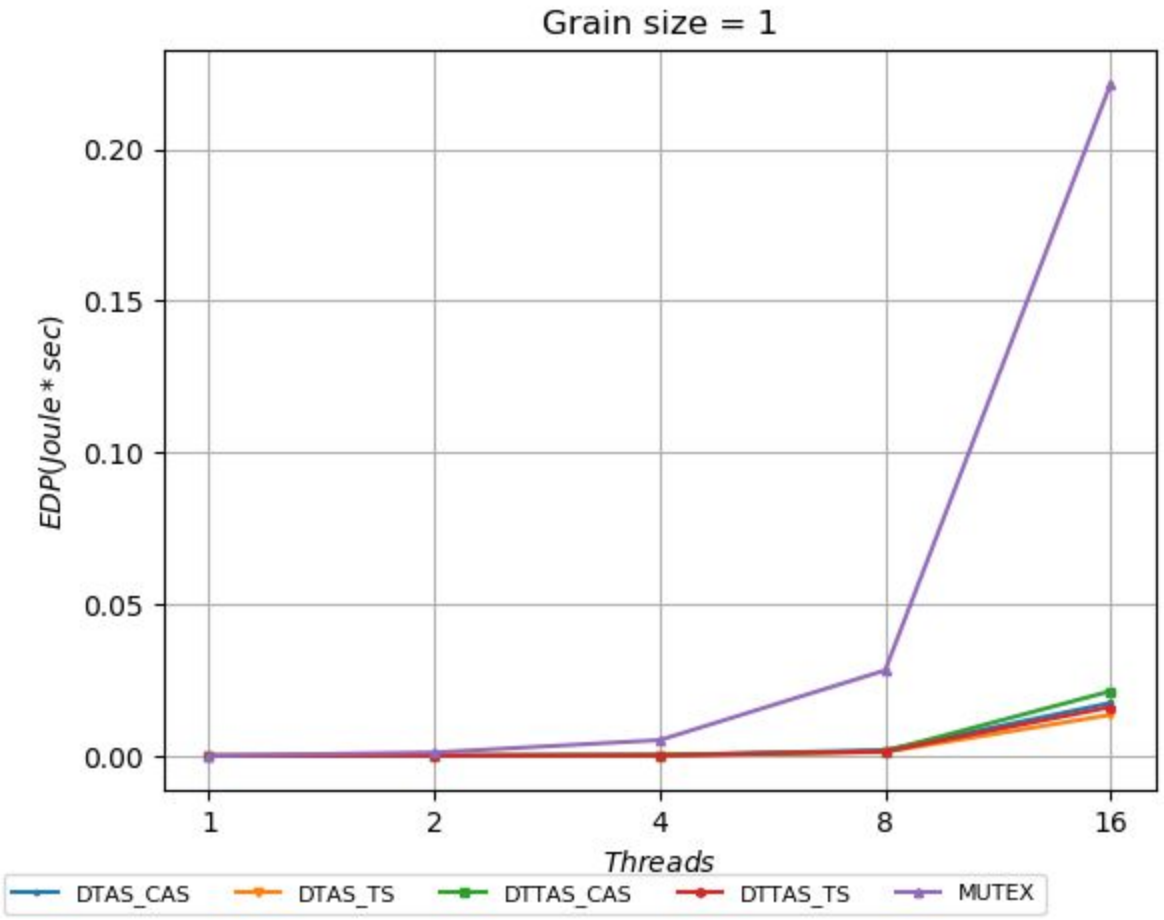
Σχετικά με τις busy-wait τεχνικές, οι TTAS τεχνικές εμφανίζουν ελαφρώς καλύτερα αποτελέσματα από τις TAS τεχνικές, οι διαφορά των οποίων αυξάνεται όσο αυξάνεται ο αριθμός των επαναλήψεων. Αυτό συμβαίνει αφού στην ttas τεχνική το πρόγραμμα επιστρέφει στην cache χωρίς να προκαλεί επιπρόσθετες μετακινήσεις στο bus.

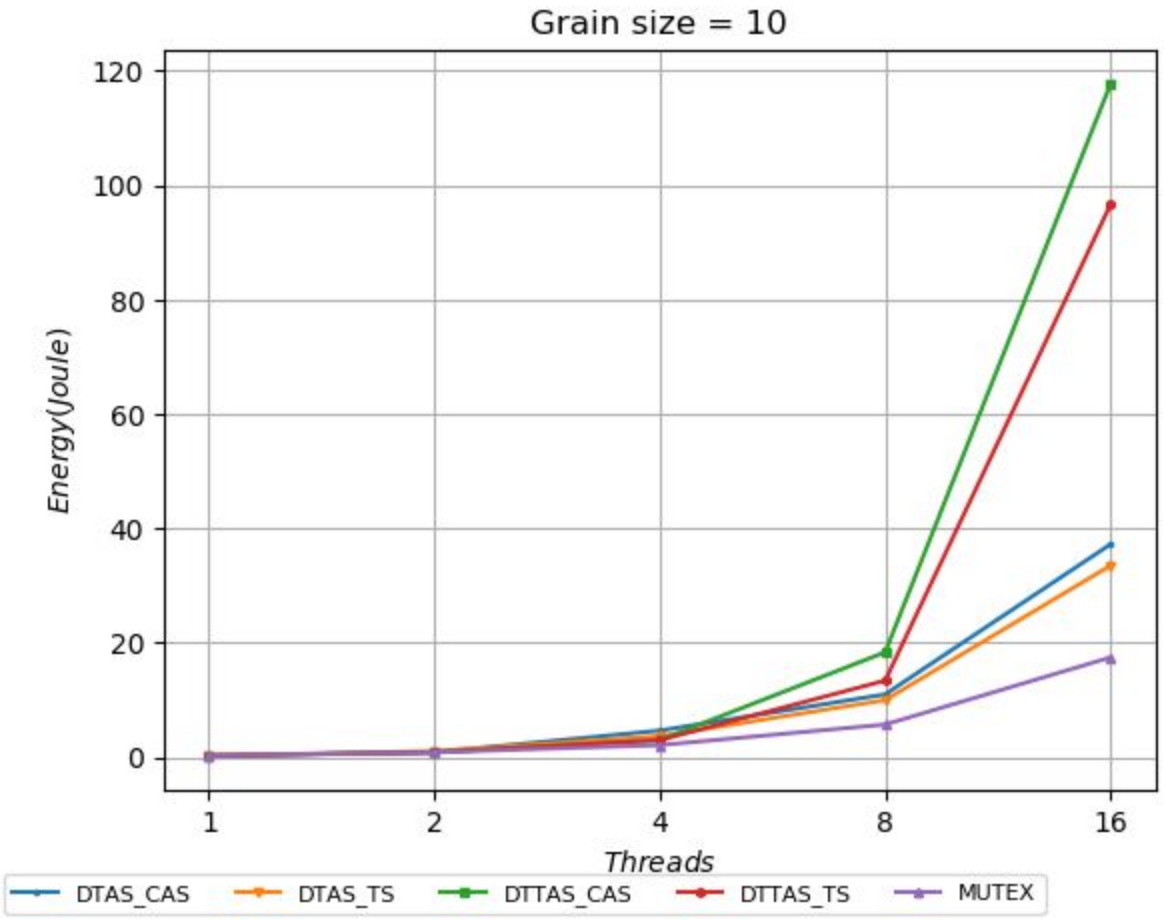
Τέλος, όσο αυξάνεται το grain size, η διαφορά χρόνου εκτέλεσης την mutex τεχνικής με τις υπόλοιπες μειώνεται. Η mutex άλλωστε είναι καλύτερη σε περιπτώσεις που έχουμε μεγάλο wait time, ώστε οι διεργασίες που περιμένουν έχει νόημα να κοιμηθούν.

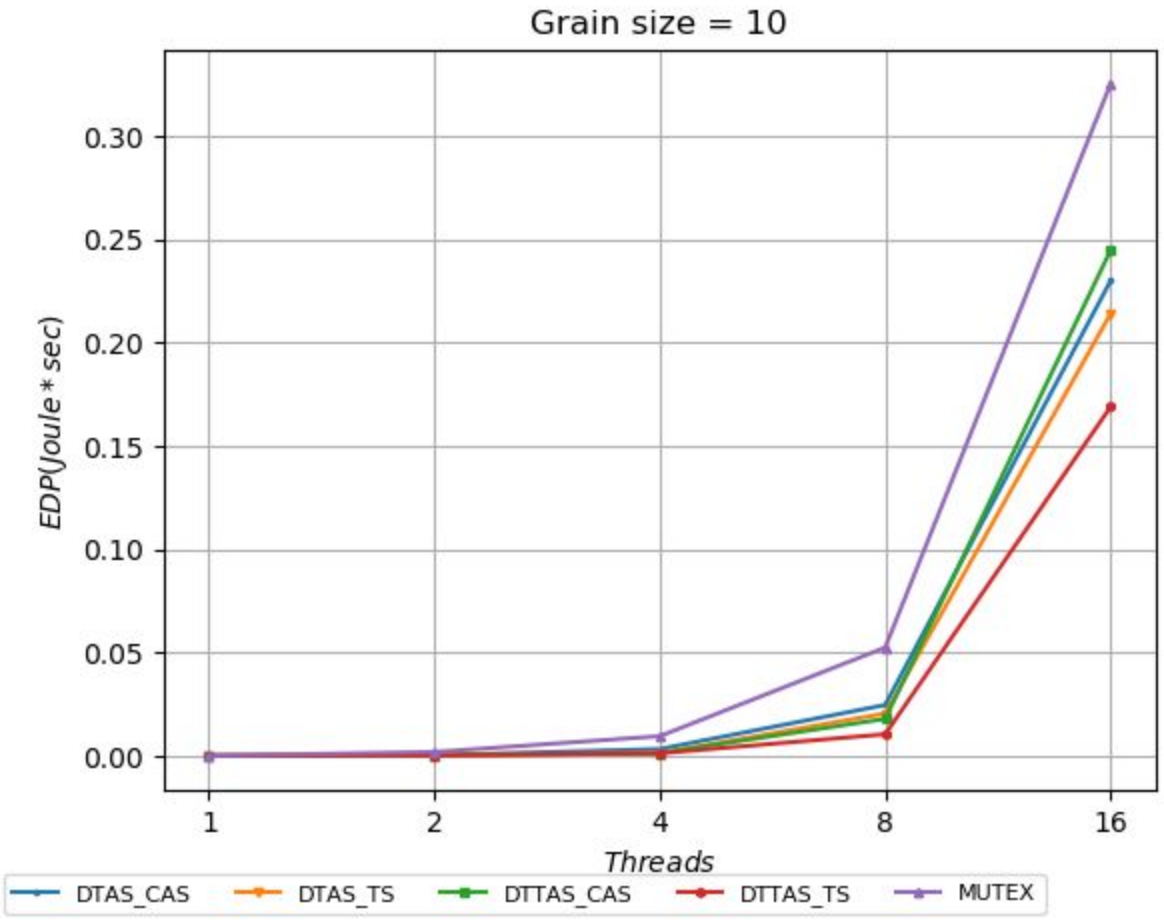
3.1.3. *Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με τη φύση της εκάστοτε υλοποίησης; Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με το grain size; Δικαιολογήστε τις απαντήσεις σας.*

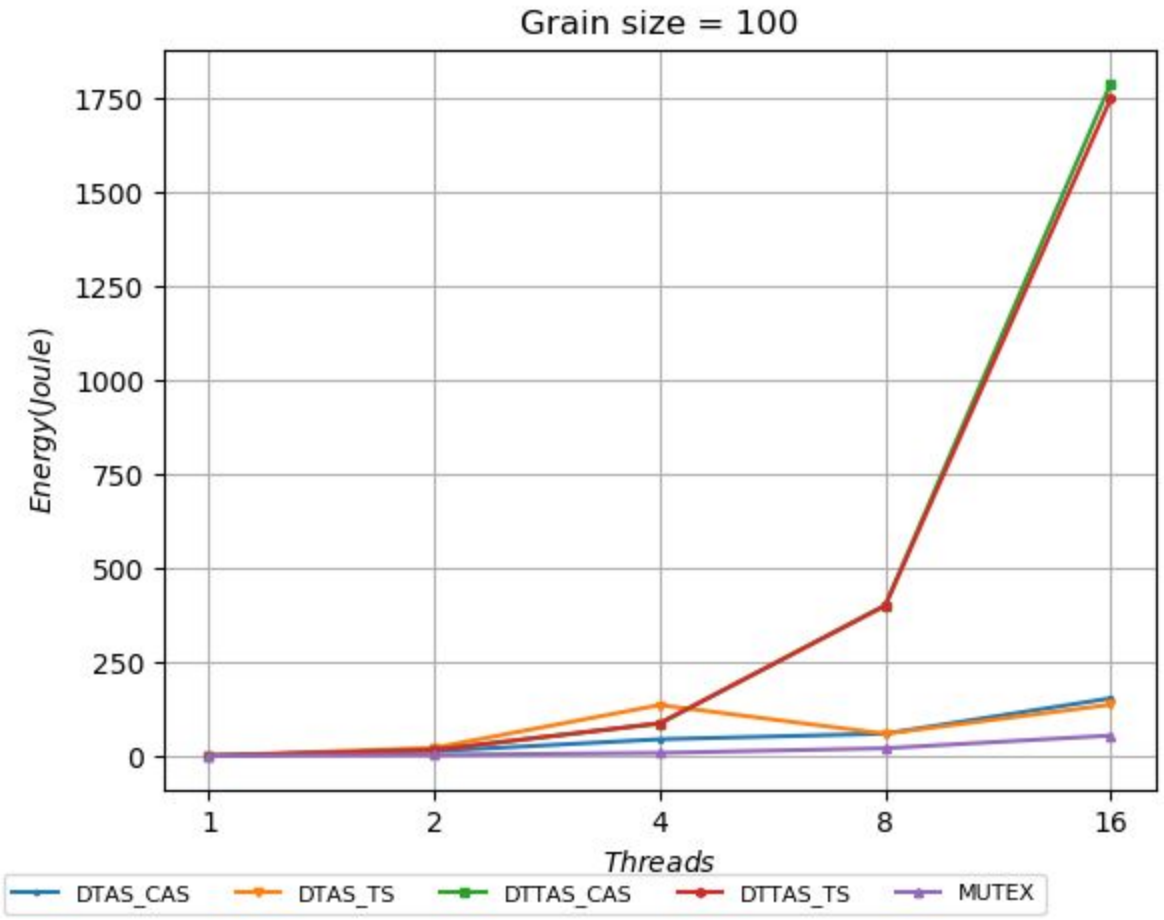
Τα διαγράμματα έχουν ως εξής:

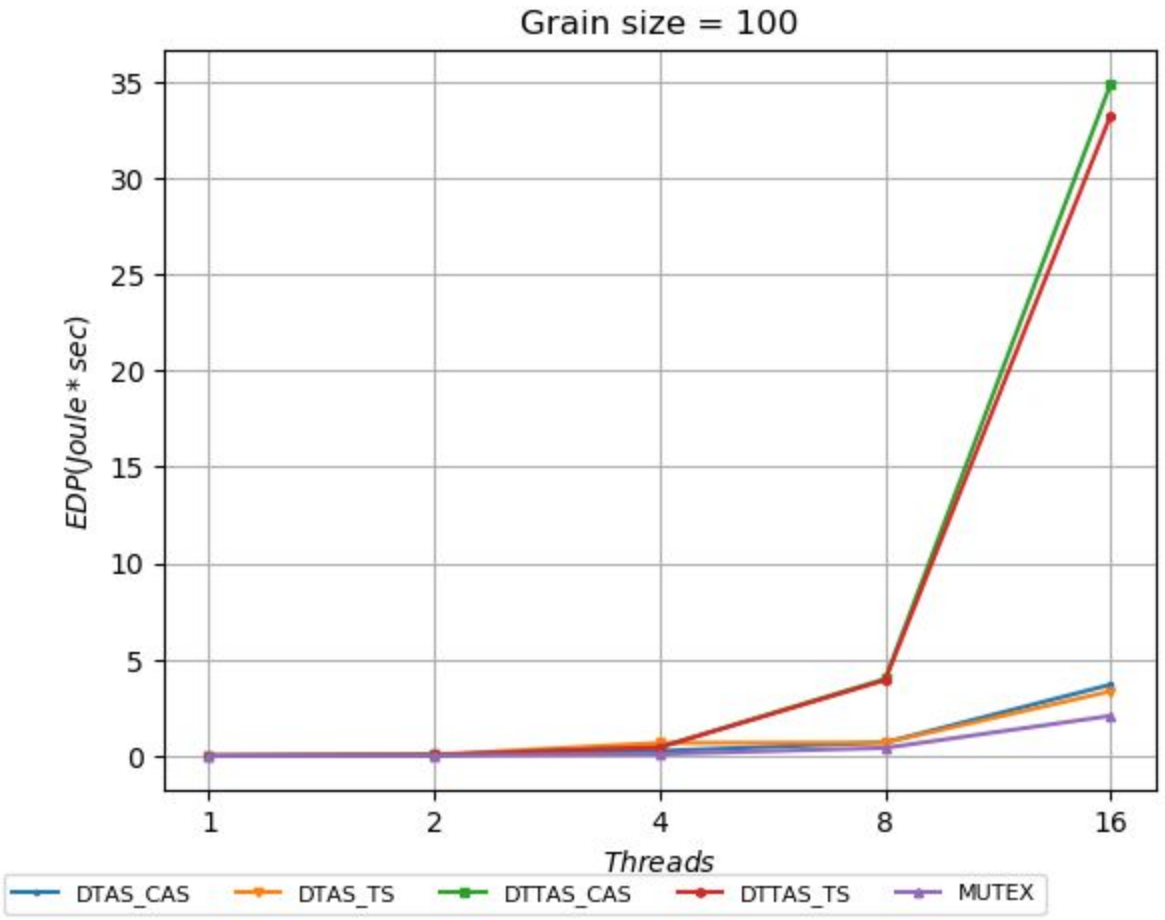

Grain size = 1
Energy(Joule)
25
20
15
10
5
0
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX


Grain size = 1
0.20
0.15
0.10
0.05
0.00
EDP(Joule * sec)
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX


Grain size = 10
120
100
80
60
40
20
0
Energy(Joule)
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX


Grain size = 10
0.30
0.25
0.20
0.15
0.10
0.05
0.00
EDP(Joule * sec)
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX


Grain size = 100
1750
1500
1250
1000
750
500
250
0
Energy(Joule)
1
2
4
8
16
Threads
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX



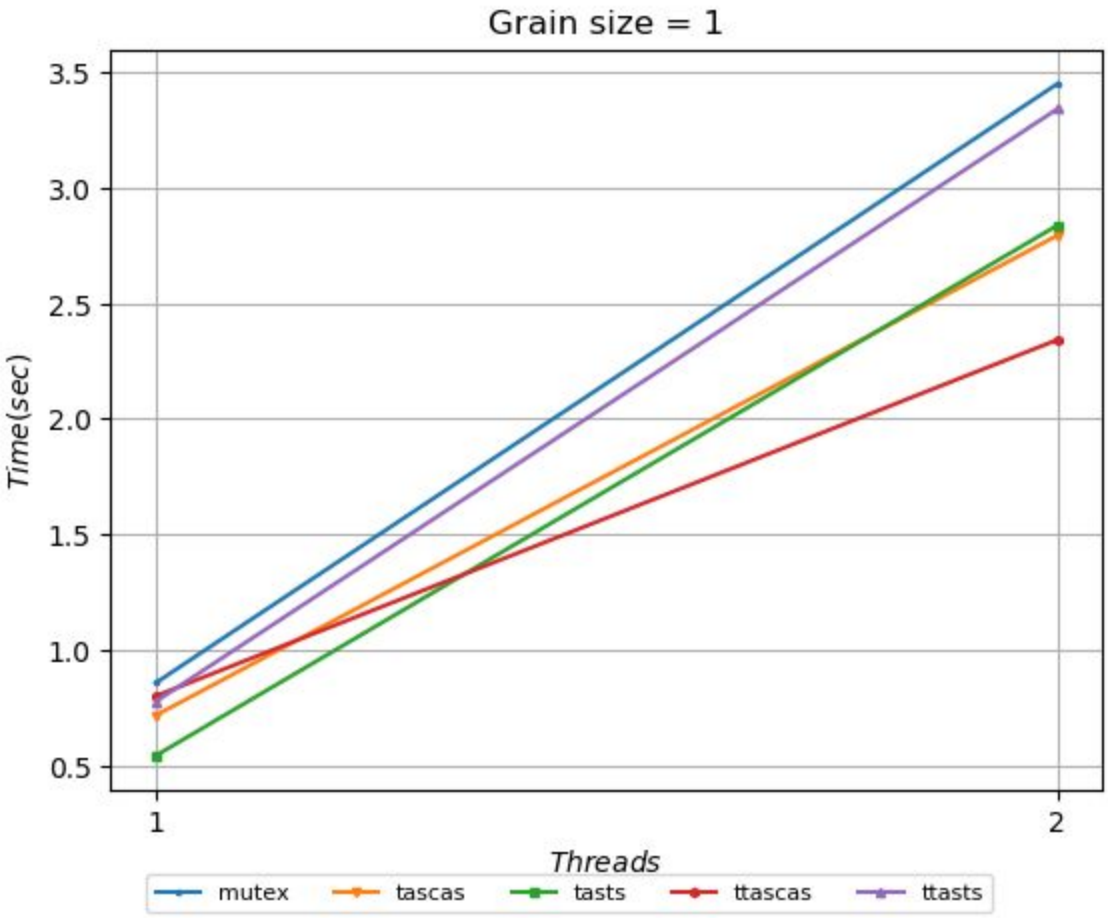
Μελετώντας τον MUTEX μηχανισμό, όσο αυξάνονται τα grains μειώνεται και η απαιτούμενη ενέργεια. Αυτό είναι αναμενόμενο αφου για grain 1 έχουμε πολύ μικρό region of interest και έτσι είναι άσκοπο να κοιμούνται οι εργασίες που περιμένουν την σειρά τους. Έτσι, ενώ για grain size = 1 είναι ο χείριστος μηχανισμός, καθώς το μέγεθος του grain αυξάνεται, η MUTEX αναδεικνύεται βέλτιστος μηχανισμός σε σχέση με τους busy-wait.

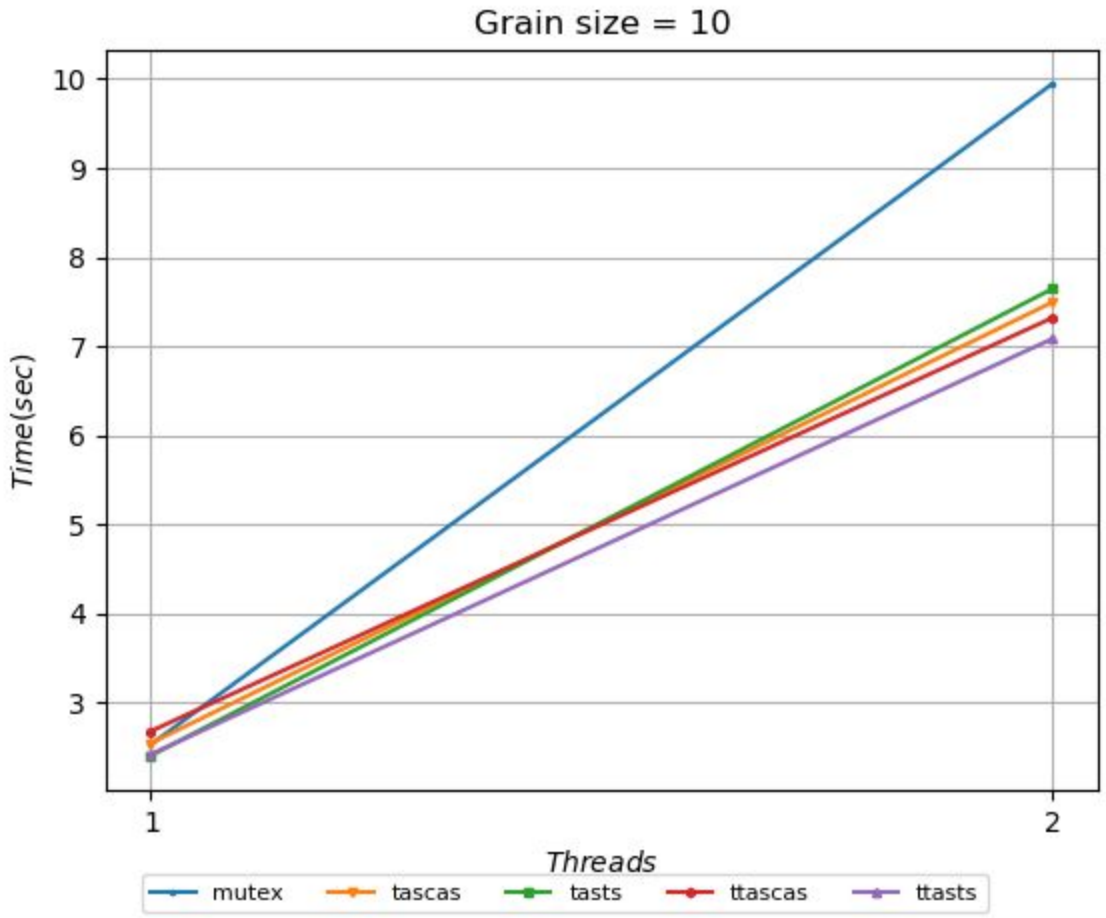
Εστιάζοντας τώρα στους busy-wait μηχανισμούς, συνολικά καταλήγουμε πως οι TAS καταναλώνουν λιγότερη ενέργεια από τους TTAS. Στο EDP για μεγάλο grain size πάλι υπερτερούν οι TAS μηχανισμοί, ενώ έχει και καλύτερη κλιμάκωση. Αυτό είναι λογικό αν σκεφτούμε πως στους TTAS είναι απαραίτητη η συνεχή πρόσβαση στην μνήμη, που καταναλώνει ενέργεια.

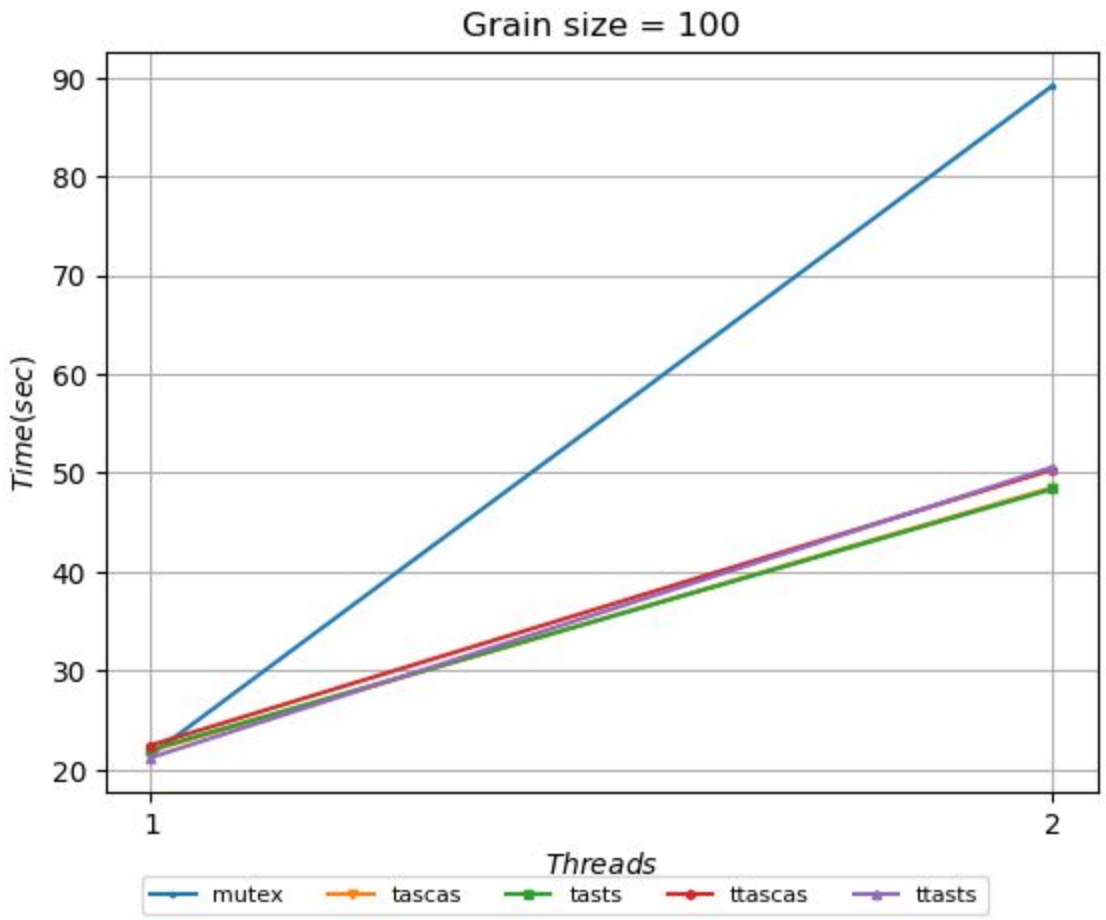
Για να επιλέξει κανείς τον βέλτιστο μηχανισμό εξαρτάται από το πρόβλημα καθώς και το πόσο σημαντική είναι η κατανάλωση ενέργειας για το παρόν πρόβλημα, πέρα από την χρονική απόδοση.

3.1.4. ***Μεταγλωττίστε τις διαφορετικές εκδόσεις του κώδικα για πραγματικό σύστημα. Εκτελέστε τα ίδια πειράματα με πριν σε ένα πραγματικό σύστημα, εφόσον αυτό διαθέτει πολλούς πυρήνες, ή σε ένα πολυπύρηνο vm που έχετε στον ~okeanos. 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 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 3.1.1. Πώς συγκρίνεται η κλιμακωσιμότητα των διαφορετικών υλοποιήσεων στο πραγματικό σύστημα σε σχέση με το προσομοιωμένο; Δικαιολογήστε τις απαντήσεις σας.***

Στο ερώτημα αυτό διεξάγαμε τις προσομοιώσεις σε σύστημα με 2 πυρήνες. Είναι συνεπώς εύλογο πως έγινε προσομοίωση για τιμές νημάτων 1 και 2 . Οι γραφικές παραστάσεις φαίνονται παρακάτω.


Grain size = 1
3.5
3.0
2.5
2.0
1.5
1.0
0.5
Time(sec)
1
2
Threads
mutex
tascas
tasts
ttascas
ttasts


Grain size = 10
Time(sec)
10
9
8
7
6
5
4
3
1
2
Threads
mutex
tascas
tasts
ttascas
ttasts



Δεδομένου του περιορισμού του συστήματος, και έχοντας αποτελέσματα μόλις για 2 νήματα, δεν μπορούμε να εξάγουμε καθοριστικά αποτελέσματα. Παρόλα αυτά δεν βλέπουμε μεγάλες αποκλίσεις από τα αρχικά μας αποτελέσματα.
Ο mutex μηχανισμος έχει τον μέγιστο χρόνο εκτέλεσης και η κλιμακωσιμότητα παρουσιάζει ίδιο μοτίβο όπως και πάνω για μεγάλα grain sizes.
Η διαφορά που παρατηρείται είναι πως υπάρχουν μικρότερες αποκλίσεις μεταξύ των διαφορετικών μηχανισμών.

3.2. Τοπολογία νημάτων

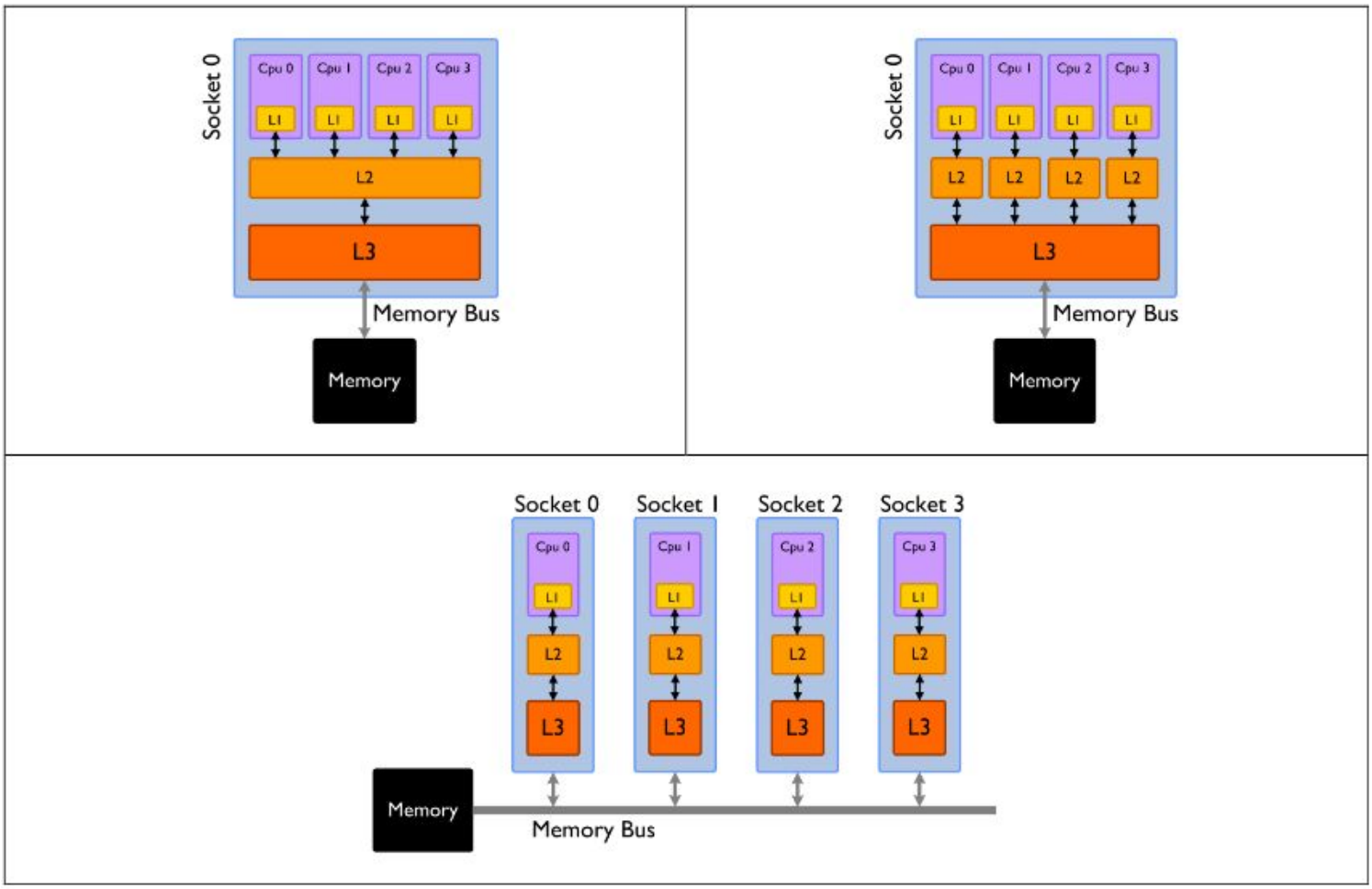
Στόχος του ερωτήματος αυτού είναι η αξιολόγηση της κλιμάκωσης των διαφόρων υλοποιήσεων όταν τα νήματα εκτελούνται σε πυρήνες με διαφορετικά χαρακτηριστικά ως προς το διαμοιρασμό των πόρων. Συγκεκριμένα, θεωρούμε τις εξής πειραματικές παραμέτρους:

- εκδόσεις προγράμματος: TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS, MUTEX
- iterations: 1000
- nthreads: 4
- grain_size: 1

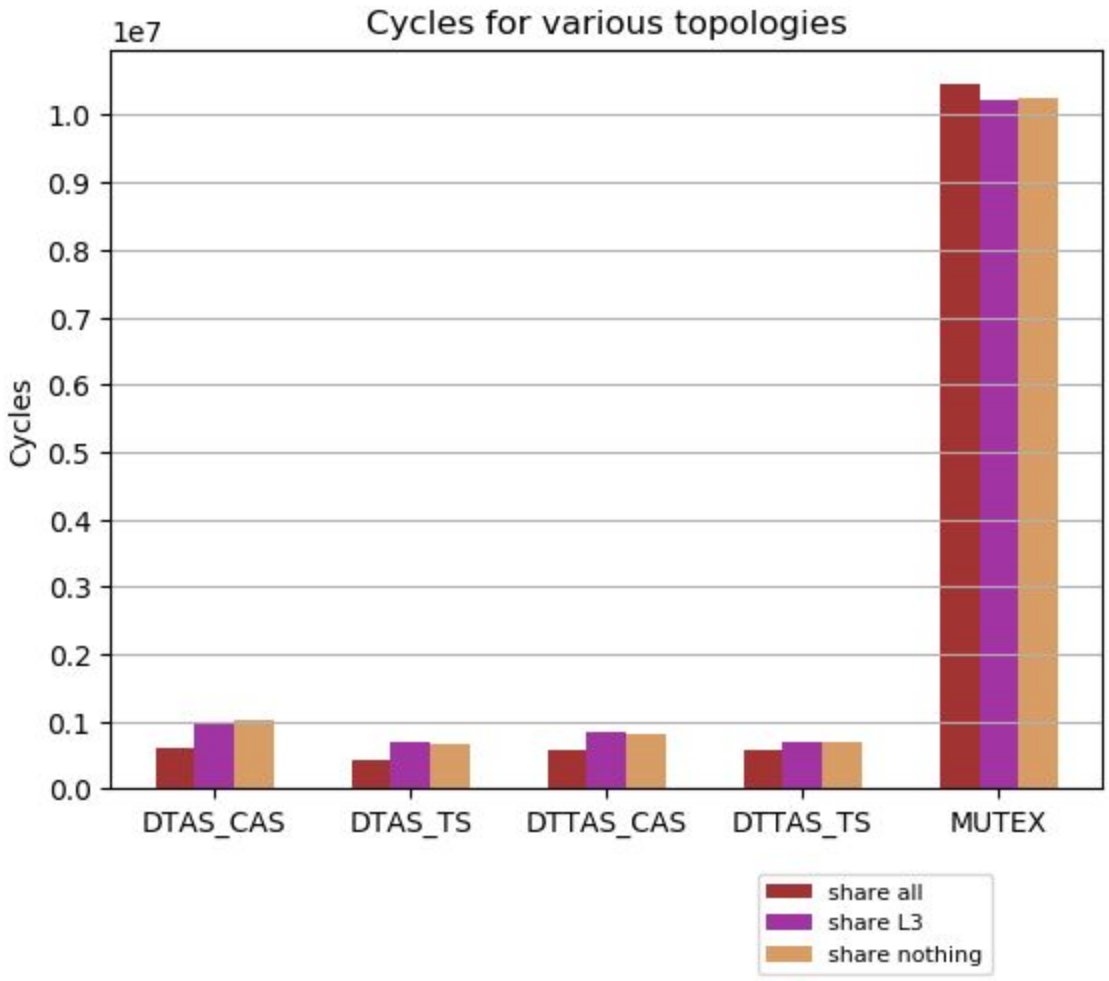
και εξετάζουμε τις ακόλουθες τοπολογίες:

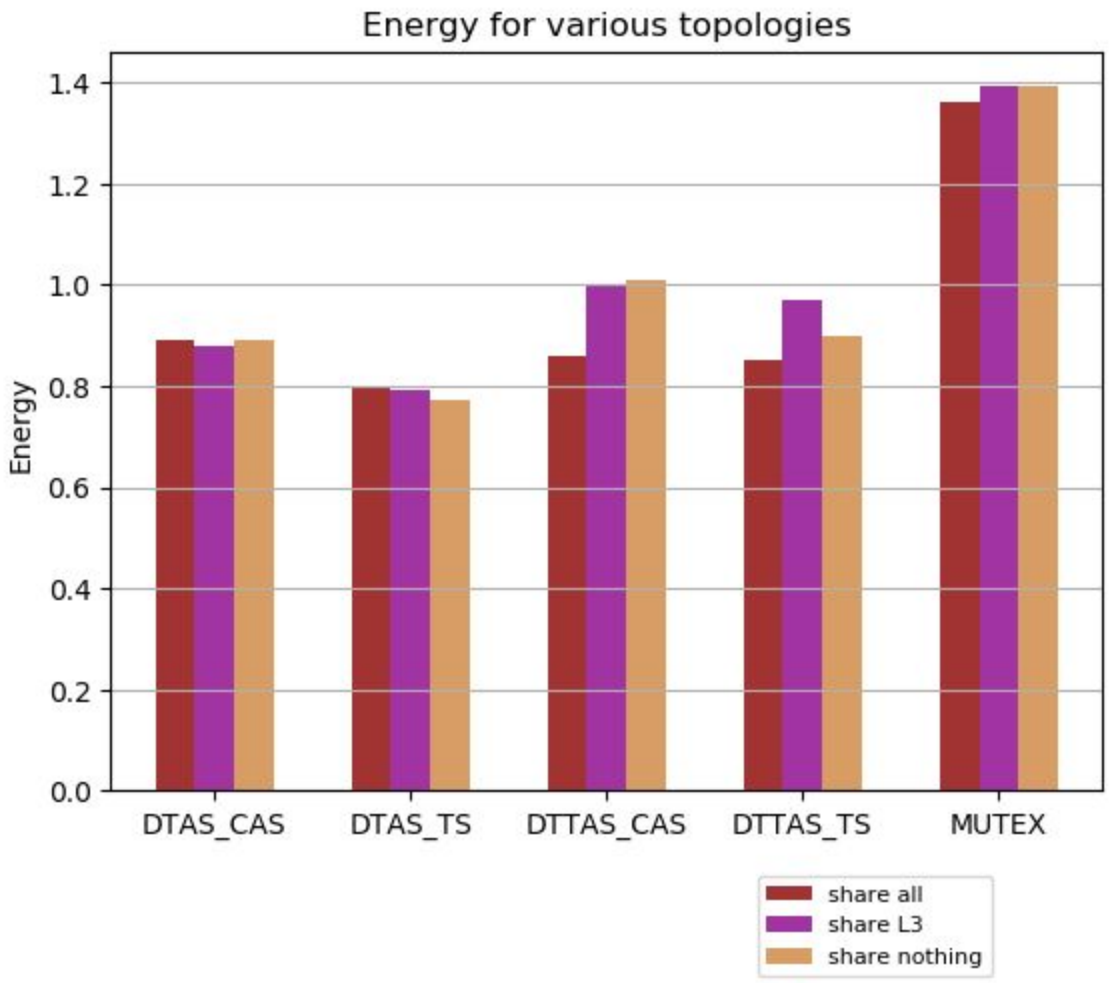
- share-all:και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L2 cache
- share-L3:και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L3 cache, αλλά όχι κοινή L2
- share-nothing:και τα 4 νήματα βρίσκονται σε πυρήνες με διαφορετική L3 cache

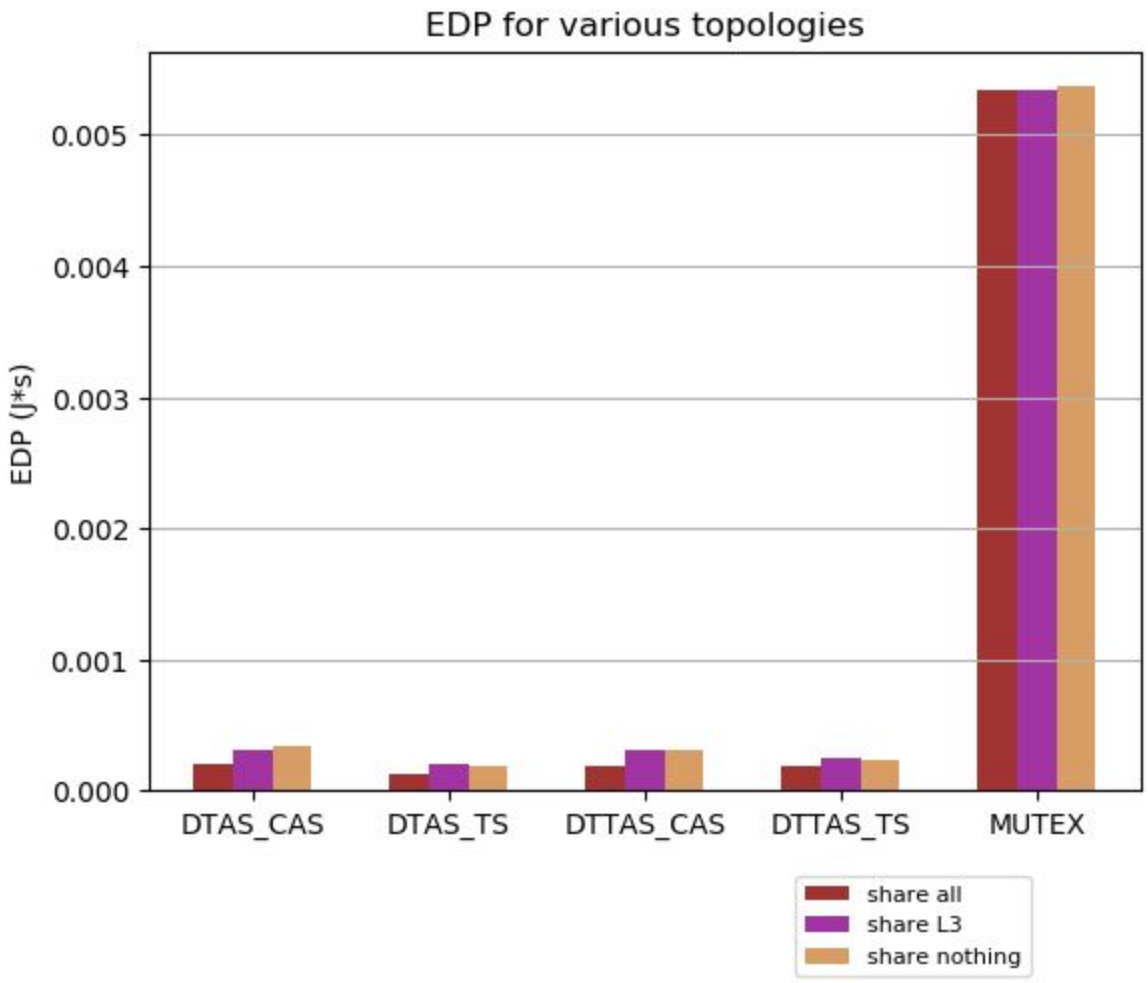
Σχηματικά, οι τοπολογίες έχουν ως εξής:



3.2.1. ***Για τις παραπάνω τοπολογίες -βλέπε εκφώνηση-, 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.***


Cycles for various topologies
1e7
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Cycles
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX
share all
share L3
share nothing


Energy for various topologies
Energy
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS
MUTEX
share all
share L3
share nothing



Μελετώντας την χρονική απόδοση, βλέπουμε πως ως βέλτιστη τοπολογία προκύπτει η share-all, με εξαίρεση τον μηχανισμό mutex όπου είναι η χειρότερη, και βέλτιστη είναι με μικρή διαφορά η share-L3.
Γενικά στον mutex, μιας και έχουμε αποτελέσματα για grain 1, δεν έχουμε κατατοπιστικά συμπεράσματα.

Σχετικά με την ενέργεια, τόσο στο mutex, όσο και στους TTAS μηχανισμούς έχουμε ελάχιστη ενέργεια στην share all τοπολογία. Μελετώντας παράλληλα το EDP, πάλι χαμηλότερες τιμές έχουμε στην share all τοπολογία.

Μικρότερο EDP παρουσιάζει ο TAS_TS μηχανισμός, αφού ελαχιστοποιούνται τα misses λόγω της κοινής cache.

Σε γενικές γραμμές όλες busy-wait τεχνικές εμφανίζουν καλύτερη απόδοση όταν υπάρχει κοινή cache, λόγω την μείωση των misses.