Εθνικό Μετσόβειο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών – 8ο Εξάμηνο
3η Άσκηση – Ακ. Έτος 2021-2022

Κυριακόπουλος Γεώργιος – el18153

Εισαγωγή

Η άσκηση ασχολείται με τους μηχανισμούς συγχρονισμού και τα πρωτόκολλα συνάφειας κρυφής μνήμης σε σύγχρονες πολυπύρηνες αρχιτεκτονικές. Χρησιμοποιείται το πρόγραμμα προσομοίωσης Sniper Multicore Simulator για να μετεληθούν και να αξιολογηθούν διάφορα τέτοια συστήματα, με τη βοήθεια και του εργαλείου PIN.

Αρχικά, θα μελετηθούν διάφορες υλοποιήσεις του συστήματος με χρήση του προσομοιωτή, αναφορικά με το συνολικό χρόνο εκτέλεσης της περιοχής ενδιαφέροντος σε σχέση με τον αριθμό των νημάτων, αλλά και την κατανάλωση ενέργειας. Στη συνέχεια, θα γίνει παρόμοια μελέτη σε πραγματικό σύστημα. Τέλος, θα γίνει μία ακόμα μελέτη σχετικά με το συνολικό χρόνο εκτέλεσης και την κατανάλωση ενέργειας, αλλά αυτή τη φορά για διάφορες τοπολογίες νημάτων.

Σύγκριση υλοποιήσεων

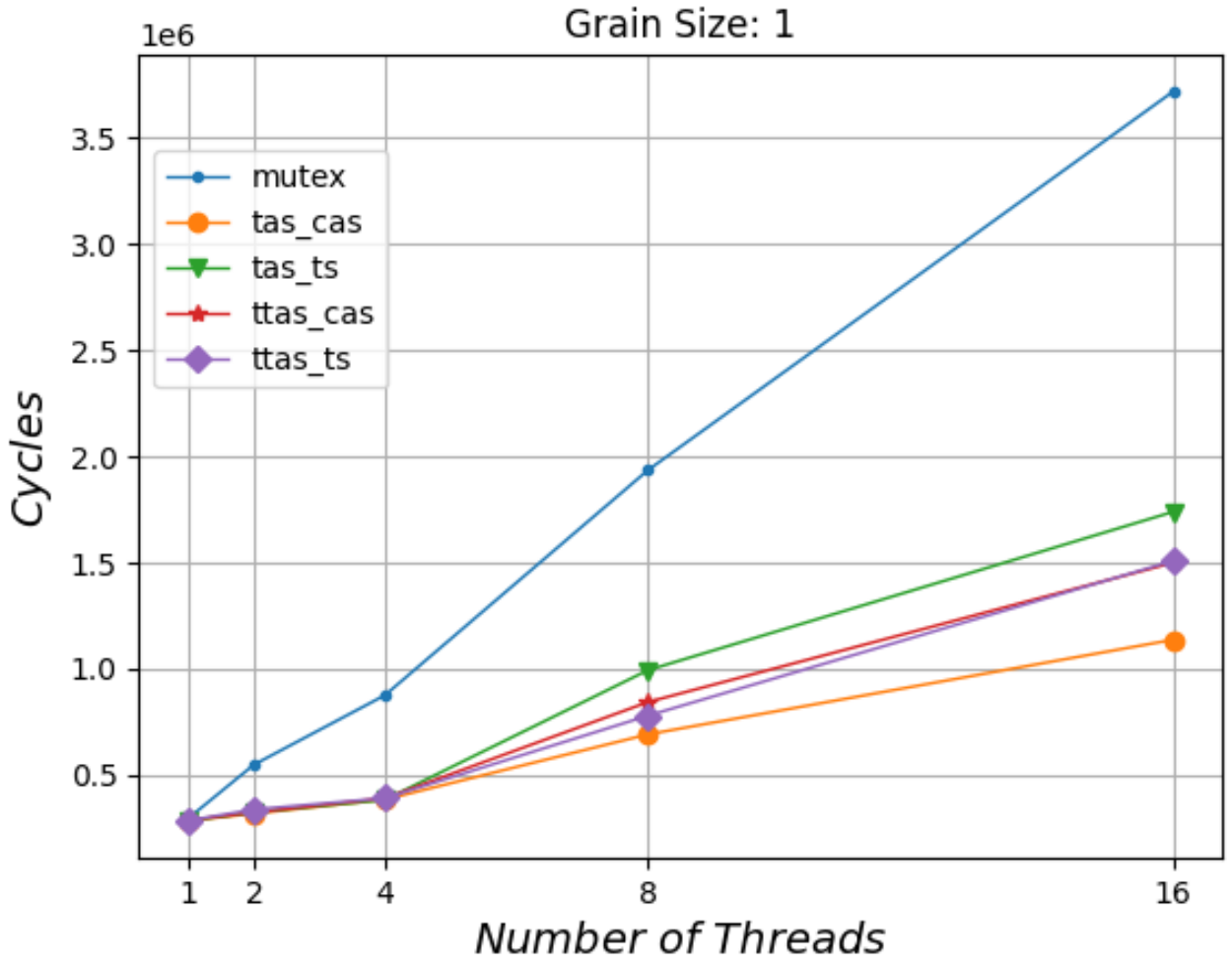
Συνολικός χρόνος εκτέλεσης

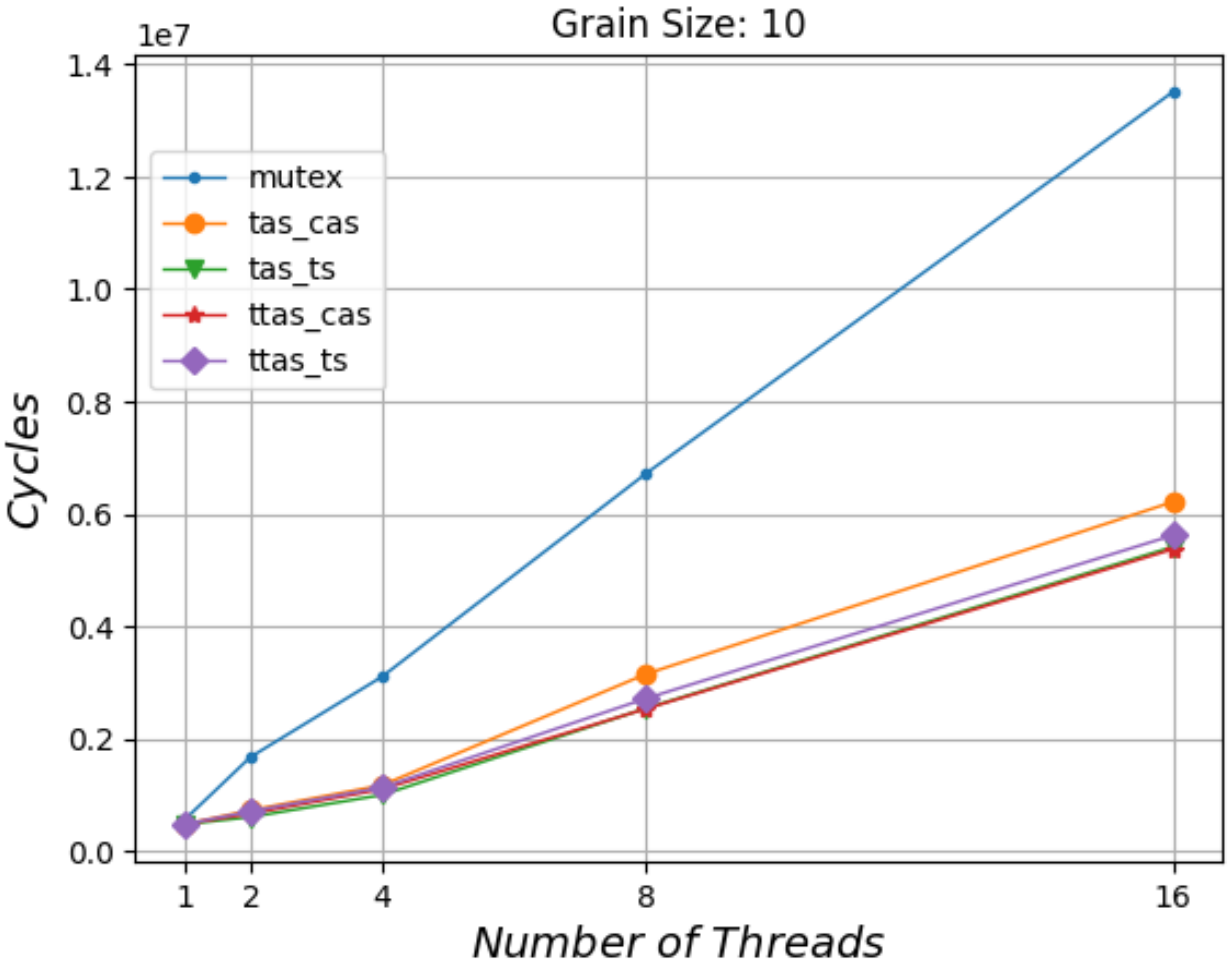
Για αυτό το ζητούμενο, εξετάζουμε με βάση το grain size (1, 10, 100) το συνολικό χρόνο εκτέλεσης της περιοχής ενδιαφέροντος σε σχέση με τον αριθμό των νημάτων.

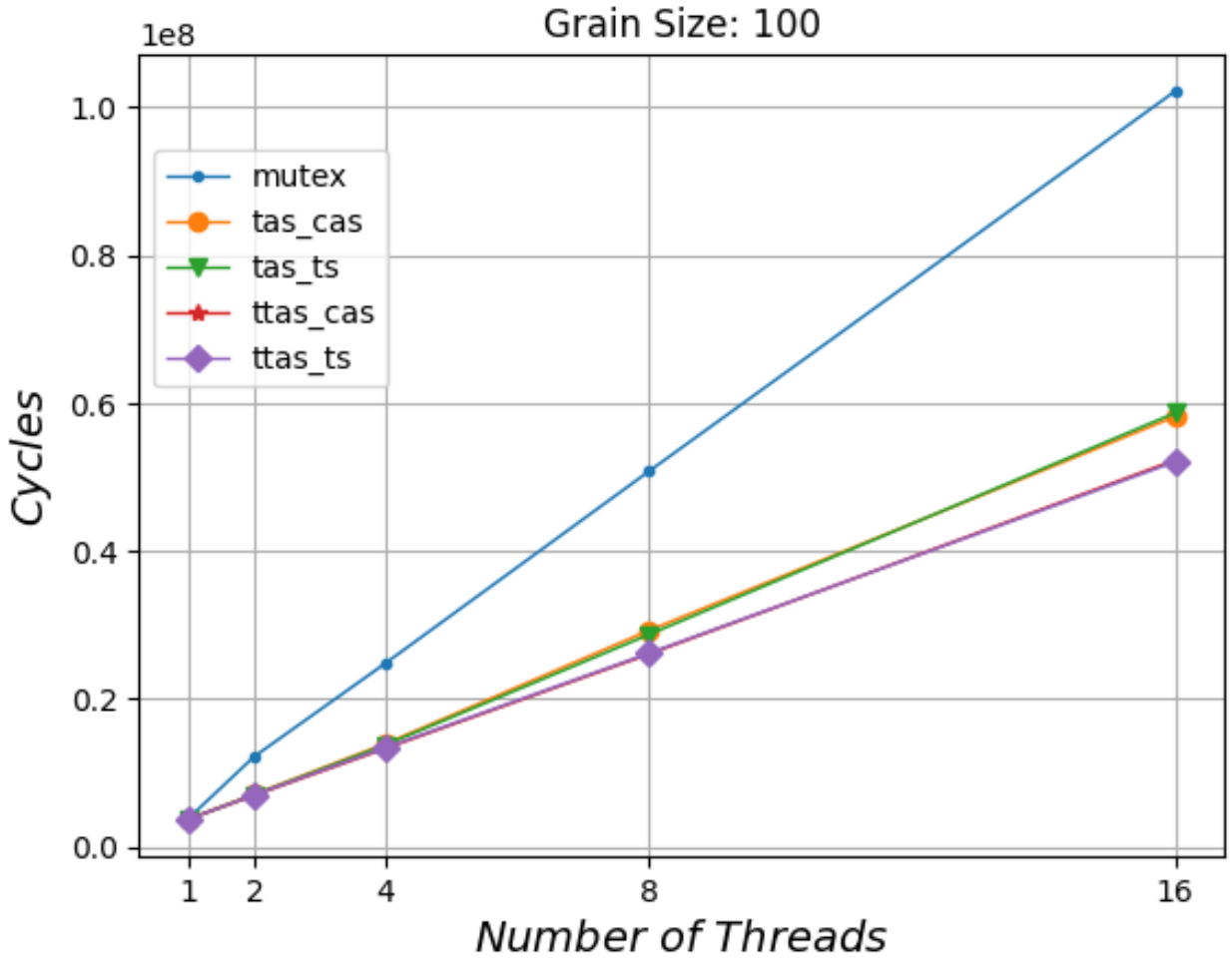
Συνολικά εκτελέστηκαν οι παρακάτω συνδυασμοί παραμέτρων για το σύστημα:

- 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 προκύπτουν, ένα για κάθε grain size.




Grain Size: 10
1e7
1.4
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

Σύγκριση υλοποιήσεων

Συμπεράσματα για συνολικό χρόνο εκτέλεσης

Αρχικά, έχουμε 3 μηχανισμούς συγχρονισμού, Mutex, Test-And-Set, Test-and-Test-And-Set. Οι δυό τελευταίοι χρησιμοποιούν μία διαφορετική υλοποίηση της Test-And-Set λογικής. Συγκεκριμένα, ο πρώτος προσπαθεί συνεχώς να γράψει πάνω στο lock, ώστε να λάβει άδεια να εκτελέσει την περιοχή ενδιαφέροντος του. Ο δεύτερος, δεν γράφει συνεχώς, αλλά διαβάζει την τιμή του lock και όταν εμφανιστεί ελεύθερο, τότε προσπαθεί να γράψει ώστε να πάρει άδεια εκτέλεσης. Έτσι, μειώνεται αρκετά το φορτίο στο δίαυλο, καθώς δε γίνονται συνεχώς εγγραφές, χωρίς να οδηγούν σε κάποιο αποτέλεσμα.

Με βάση τα διαγράμματα, παρατηρούμε γενικότερα μία γραμμική συμπεριφορά του συνολικού χρόνου εκτέλεσης, σε σχέση με τον αριθμό των νημάτων. Επίσης, διαπιστώνουμε ότι ο μηχανισμός MUTEX έχει σταθερά και με διαφορά τη χειρότερη επίδοση από όλους τους μηχανισμούς και για τα 3 grain size (1, 10, 100).

Για grain size 1, βλέπουμε ότι οι μηχανισμοί ttas_cas και ttas_ts έχουν παρόμοια απόδοση (αφού πρόκειται για ίδιο μηχανισμό με διαφορετική υλοποίηση, test-and-set vs. compare-and-swap), ενώ οι tas_cas και tas_ts, παρουσιάζουν μία μικρή διαφορά στην απόδοση.

Για grain size 10, οι 4 αυτοί μηχανισμοί έχουν παραπλήσια απόδοση.

Για grain size 100, βλέπουμε πάλι τους tas_cas και tas_ts να έχουν ίδια απόδοση και ομοίως τους ttas_cas και ttas_ts μεταξύ τους.

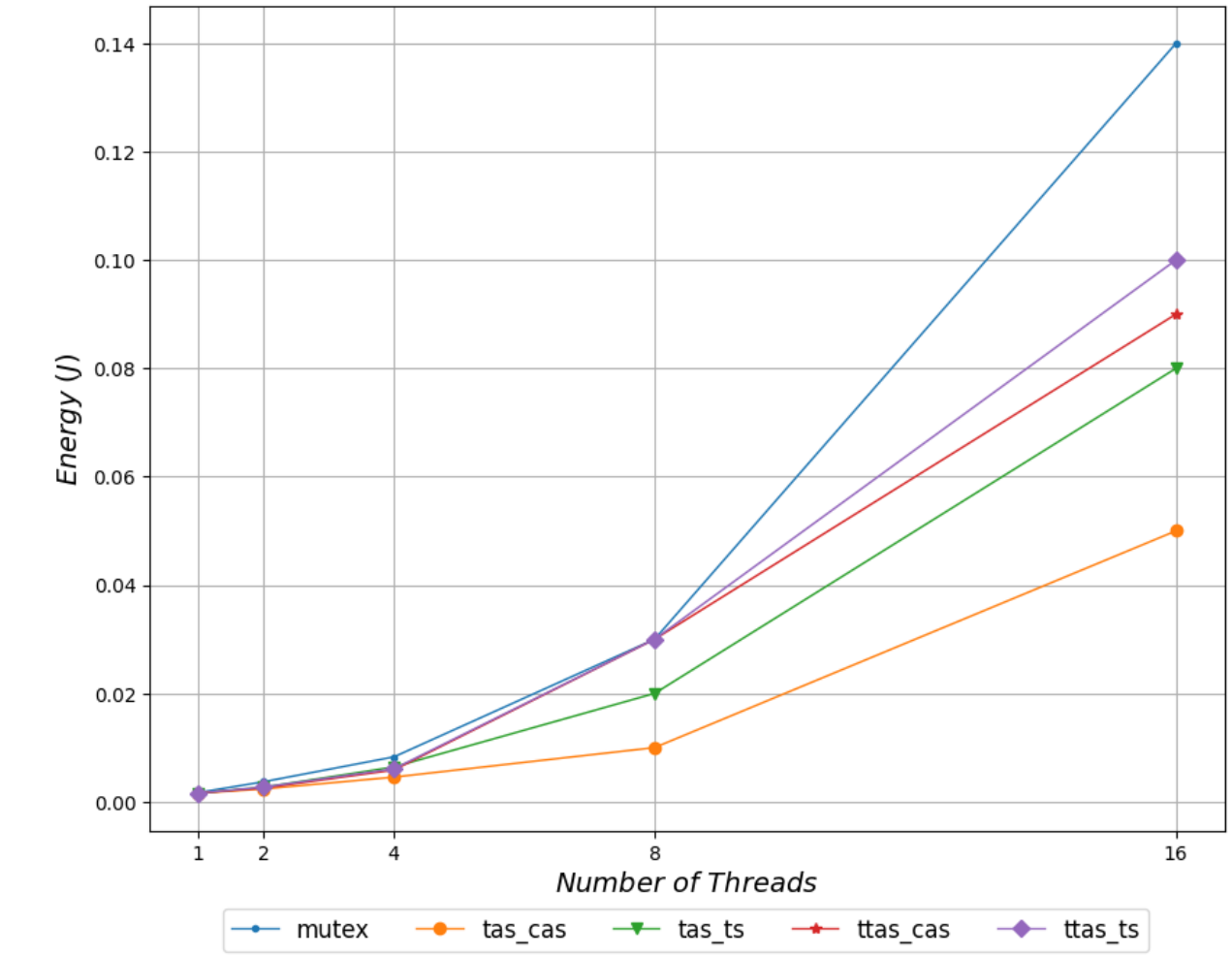
Μια ακόμα παρατήρηση είναι πως για κάθε αύξηση του grain size (x10), δεκαπλασιάζονται περίπου και οι κύκλοι που απαιτούνται για την ολοκλήρωση της εκτέλεσης.

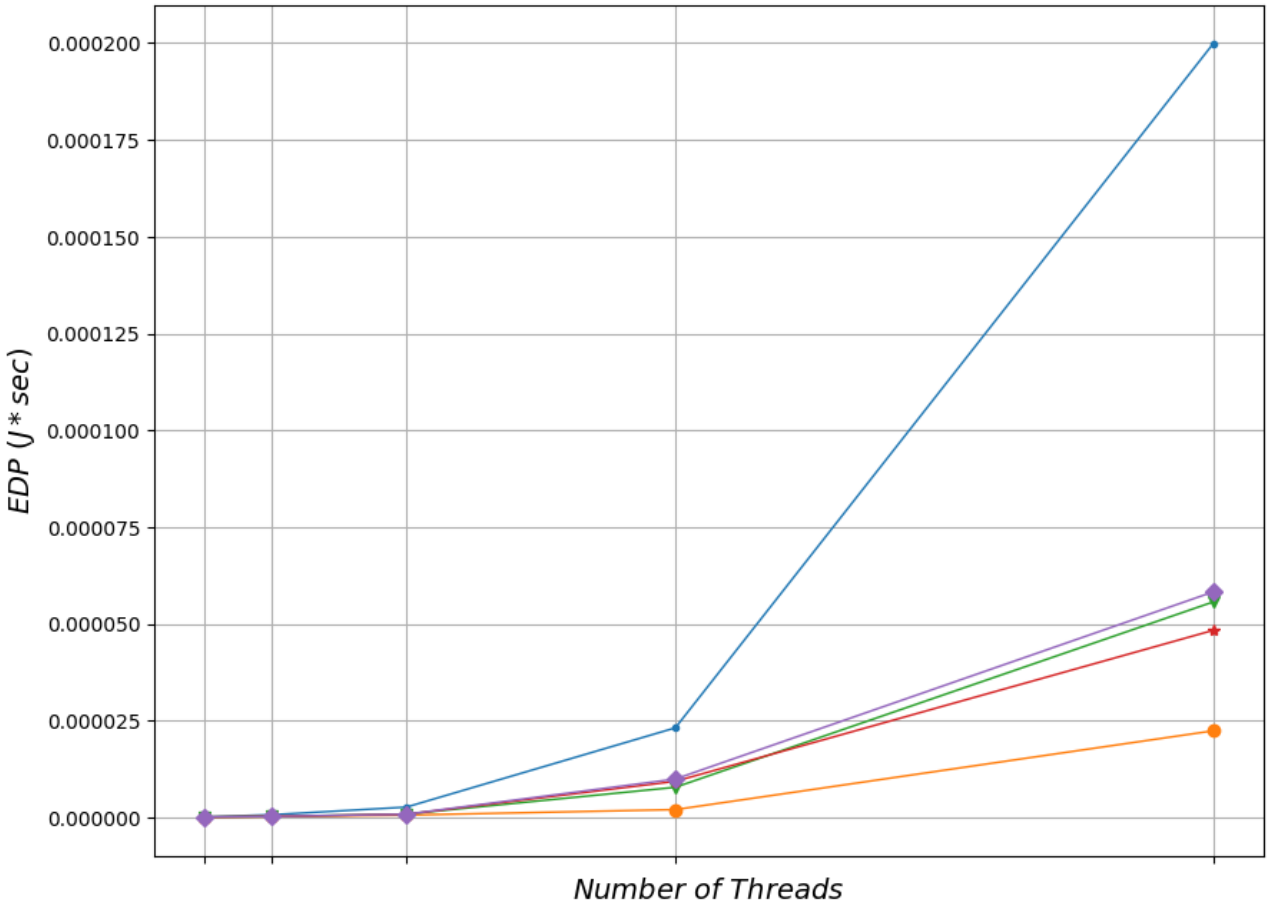
Επίσης, γενικότερα, διαπιστώνουμε ότι οι μηχανισμοί ttas_cas και ttas_ts, με εξαίρεση για grain size 1, όπου προστίθεται και ο tas_cas, έχουν την καλύτερη κλιμακωσιμότητα, κάτι που είναι αναμενόμενο, λόγω της καλύτερης υλοποίησης του πρωτοκόλλου ttas έναντι του tas και του mutex, όπως αναλύθηκε και νωρίτερα.

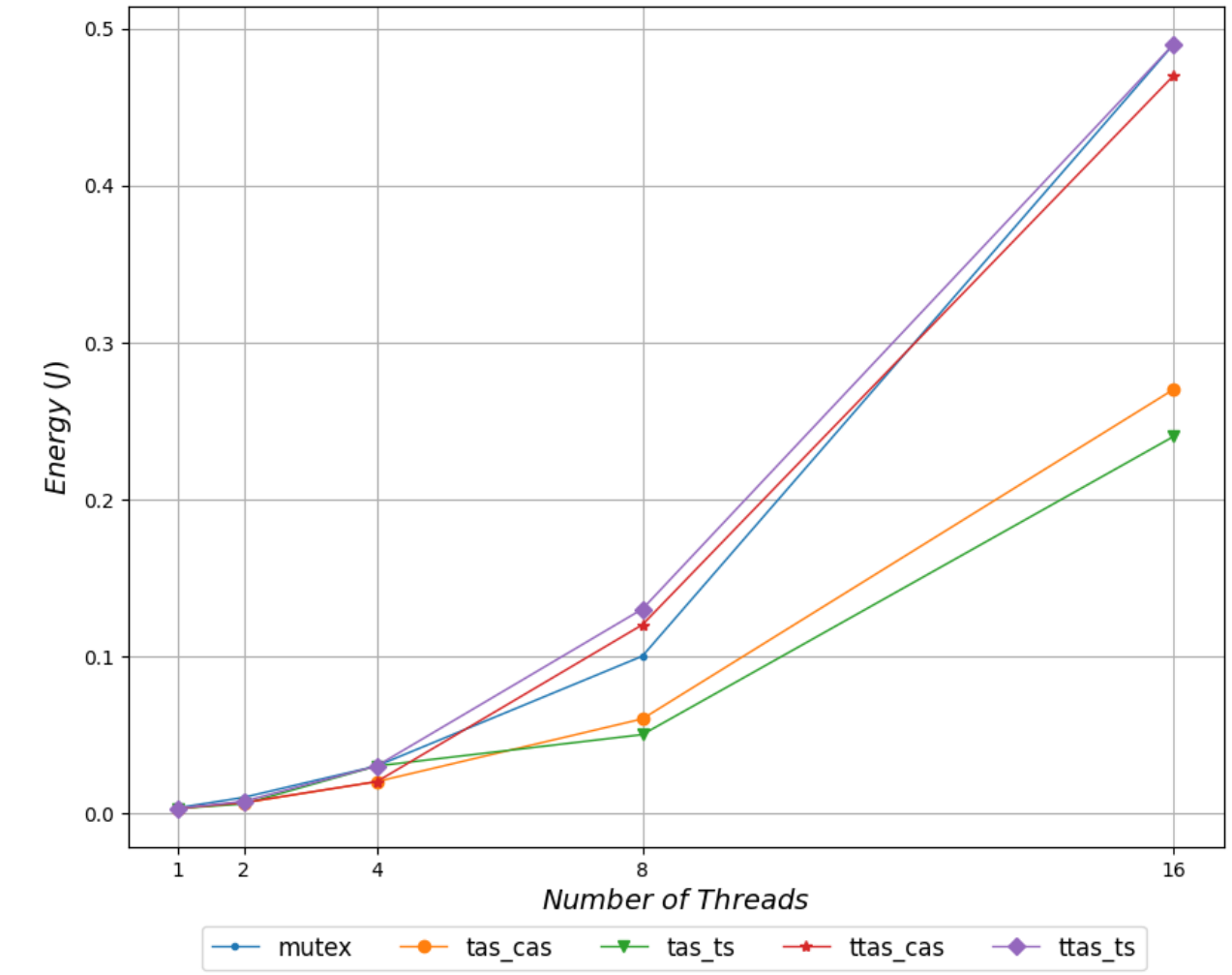
Σύγκριση υλοποιήσεων

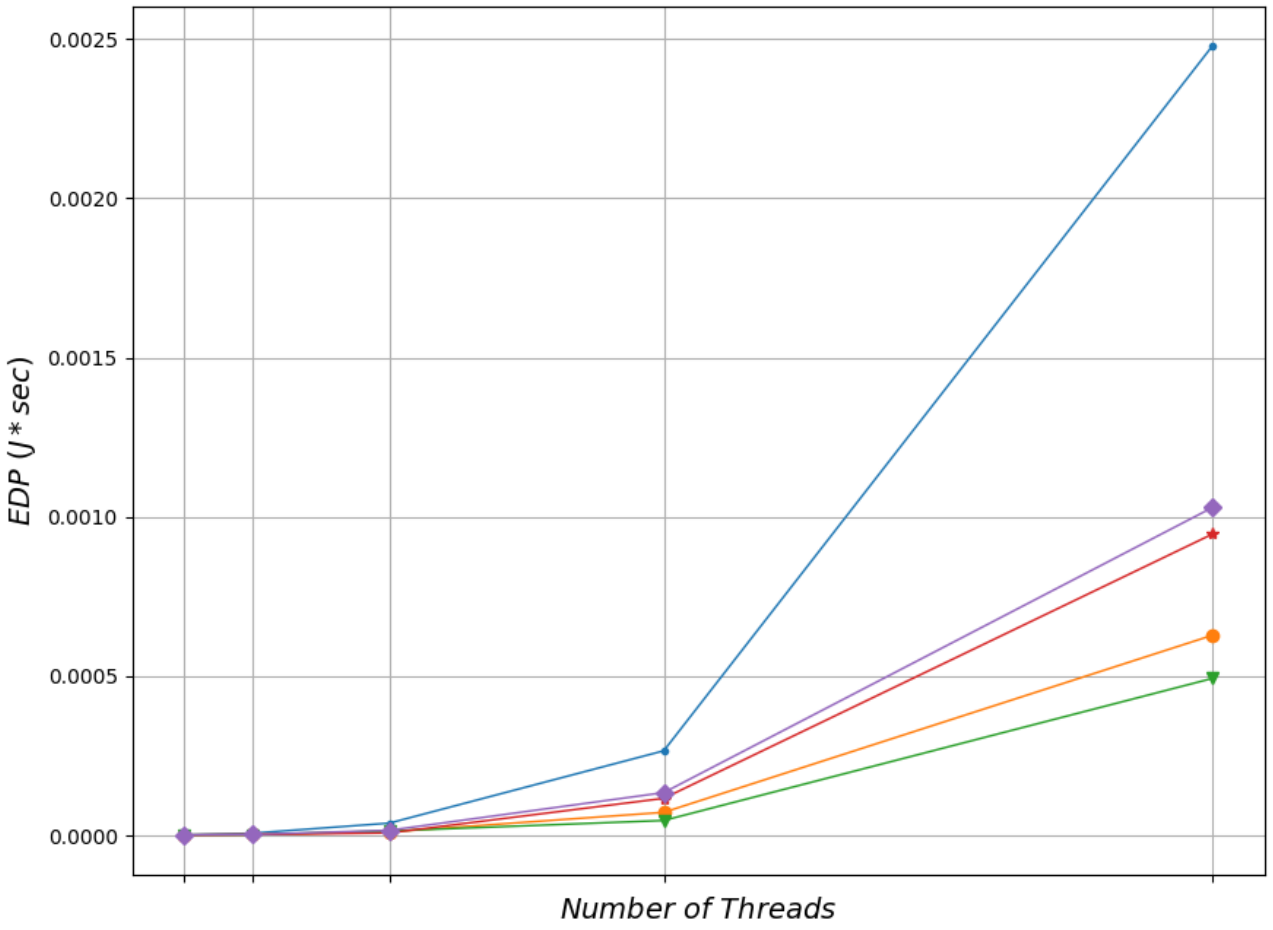
Κατανάλωση ενέργειας

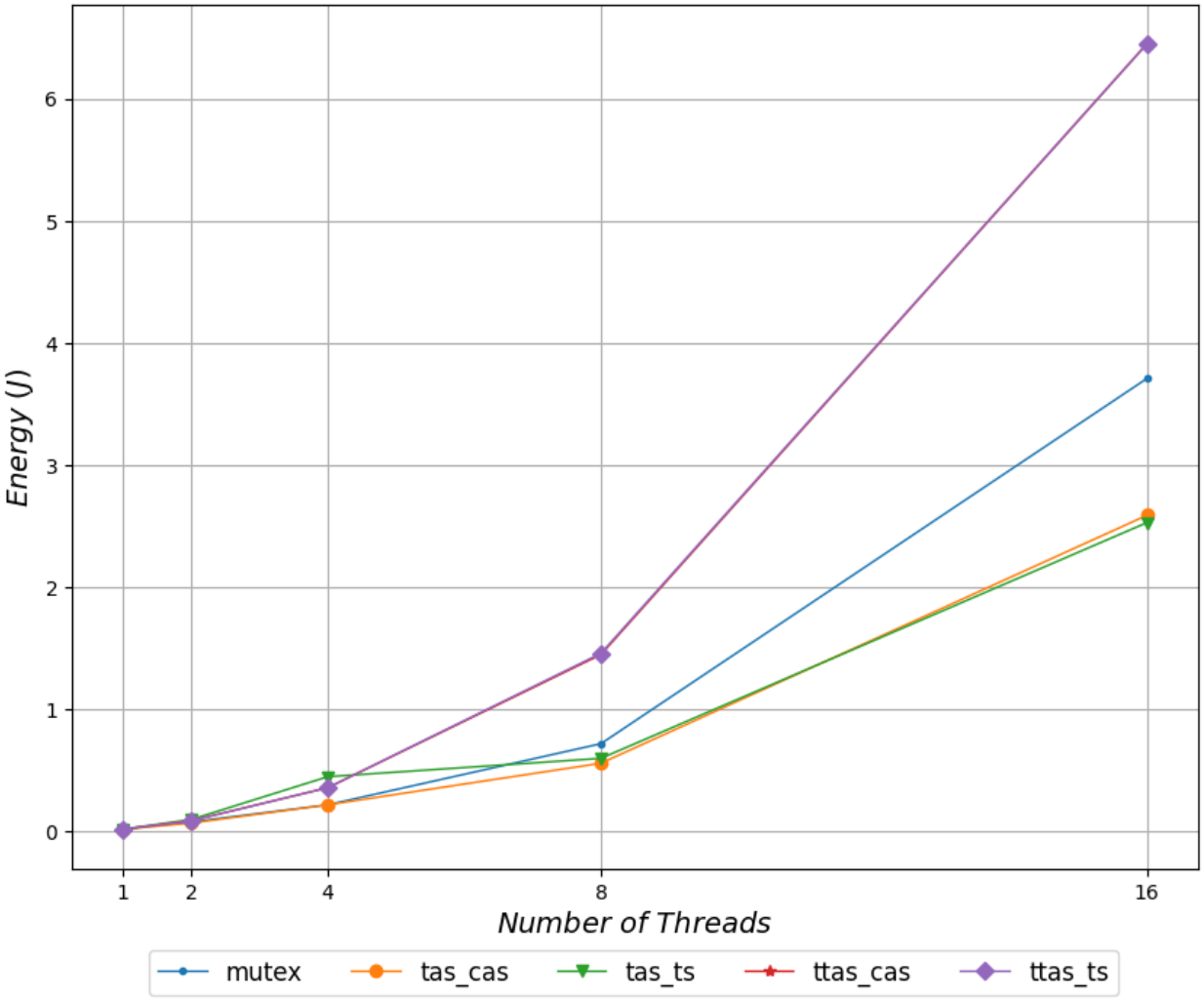
Ακολουθούν τα διαγράμματα για την κατανάλωση ενέργειας (τόσο Energy, όσο και EDP), με χρήση του εργαλείου McPAT. Είναι με τη σειρά για grain size 1, 10, 100.

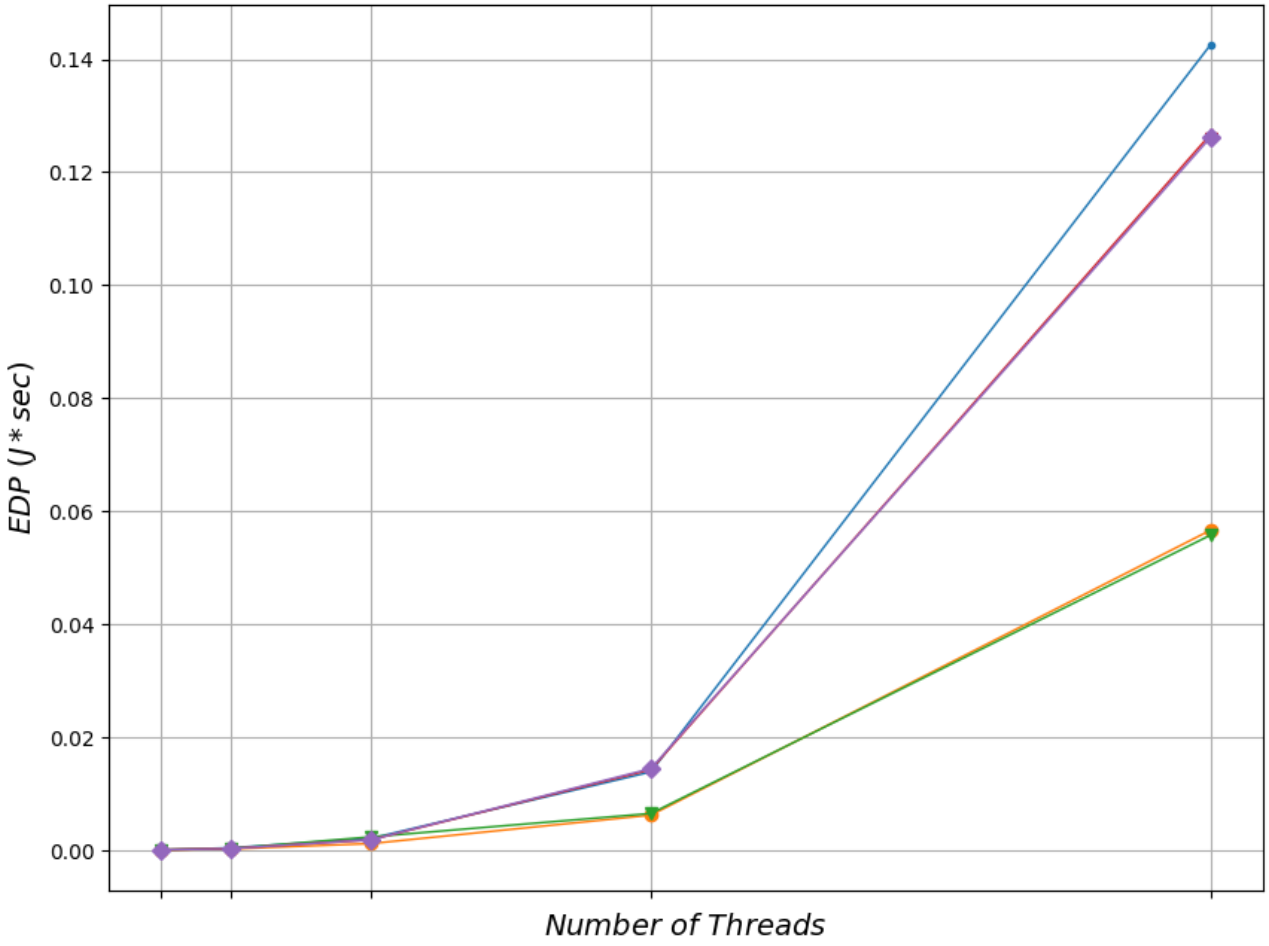

0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Energy (J)
1
2
4
8
16
Number of Threads
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts


0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
EDP (J * sec)
Number of Threads


0.5
0.4
0.3
0.2
0.1
0.0
Energy (J)
1
2
4
8
16
Number of Threads
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts


0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
EDP (J * sec)
Number of Threads


Energy (J)
0
1
2
3
4
5
6
1
2
4
8
16
Number of Threads
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts


EDP (J * sec)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Number of Threads

Σύγκριση υλοποιήσεων

Συμπεράσματα για κατανάλωση ενέργειας

Γενικότερα, παρατηρούμε μία εκθετική αύξηση της ενέργειας σε σχέση με τον αριθμό των νημάτων. Αυτό είναι έντονα φανερό και στο άλμα από 8 σε 16 νήματα, όπου οι τιμές της ενέργειας ανεβαίνουν αρκετά.

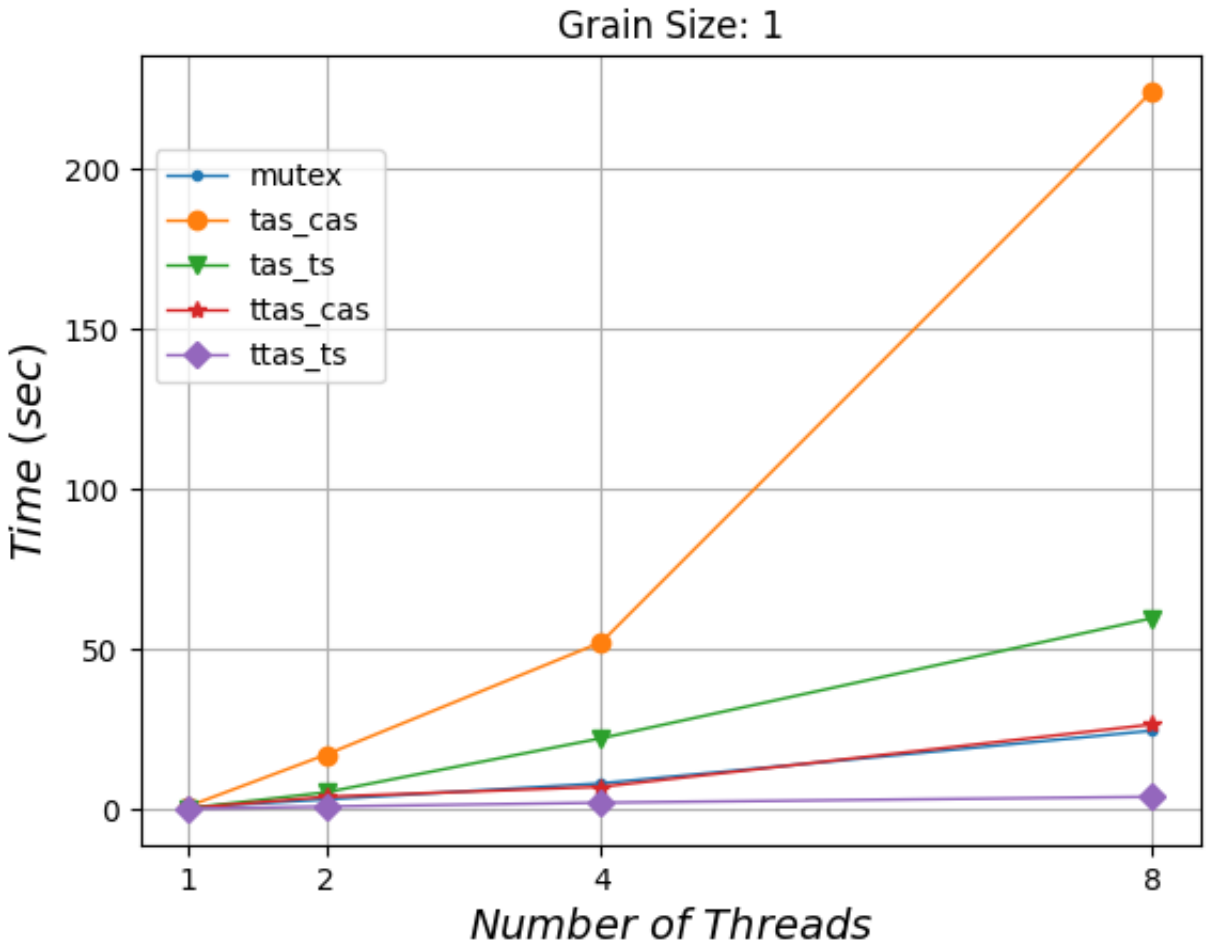
Και εδώ το mutex παρουσιάζει από τις χειρότερες συμπεριφορές (ειδικά στη μετρική EDP), ωστόσο οι μηχανισμοί ttas_cas και ttas_ts φαίνεται να σπαταλούν και αυτοί μεγάλα ποσά ενέργειας, σε σύγκριση και με τους tas_cas και tas_ts μηχανισμούς. Αυτό λογικά οφείλεται στο γεγονός ότι γίνονται πολλά κοστοβόρα reads στην περίπτωση του ttas, μέχρι να βρει ελεύθερο lock, ενώ σε αντίθεση περίπτωση το σκέτο tas πιθανόν να προκαλέσει κάποιο cache miss και stall, μειώνοντας την κατανάλωση ενέργειας.

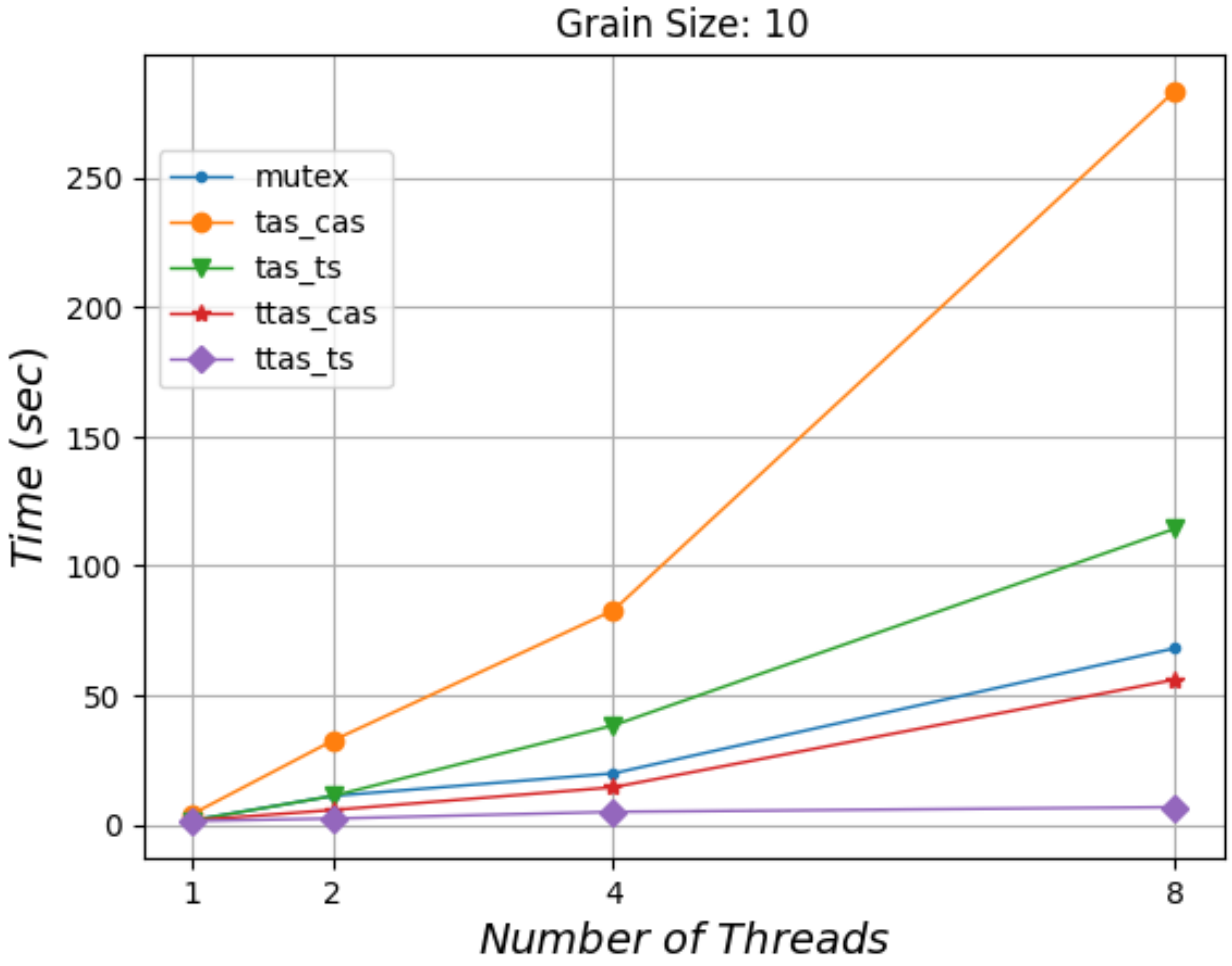
Εμφανές είναι, πάντως, ότι οι μηχανισμοί tas_cas και tas_ts αποτελούν τις καλύτερες ενεργειακά λύσεις, αφού καταναλώνουν σταθερά χαμηλότερα ποσά ενέργειας και έχουν καλές επιδόσεις και στη μετρική EDP, λόγω του πλεονεκτήματος που αναφέρθηκε.

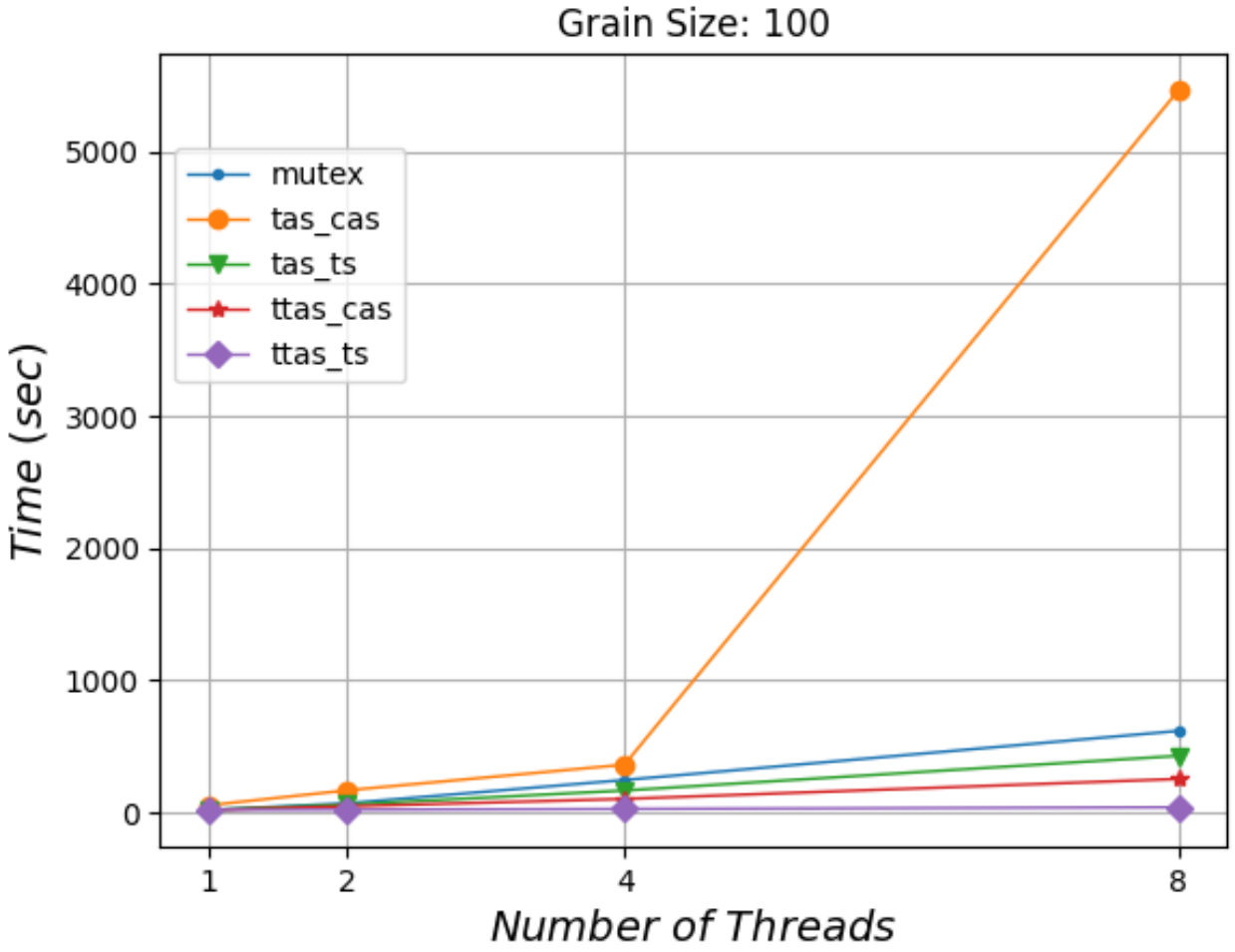
Σύγκριση υλοποιήσεων

Πραγματικό σύστημα

Ακολουθούν τα γραφήματα για την εκτέλεση σε πραγματικό σύστημα. Χρησιμοποιήθηκαν 8 πυρήνες μέσω virtual machine και αριθμός επαναλήψεων ίσος με 30.000.000.




Grain Size: 10
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Time (sec)
0
50
100
150
200
250
1
2
4
8
Number of Threads


Grain Size: 100
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Time (sec)
5000
4000
3000
2000
1000
0
1
2
4
8
Number of Threads

Σύγκριση υλοποιήσεων

Συμπεράσματα για πραγματικό σύστημα

Για τις εκτελέσεις στο πραγματικό σύστημα παρατηρούμε πως ο μηχανισμός tas_cas ακολουθούμενος από τον mutex, είναι με διαφορά ο χειρότερος σε θέμα χρόνου εκτέλεσης.

Καλύτερος σε αυτόν τον τομέα είναι ο ttas_ts ακολουθούμενος από τον ttas_ts που έχει απλά διαφορετική υλοποίηση στις ατομικές εντολές.

Παρατηρούμε επίσης ότι γενικά οι γραμμές ακολουθούν μια γραμμική τροχιά, ωστόσο παρεκκλίνει έντονα από αυτήν κυρίως ο μηχανισμός tas_cas (και λιγότερο αισθητά οι υπόλοιποι) κατά τη μετάβαση από τα 4 στα 8 νήματα.

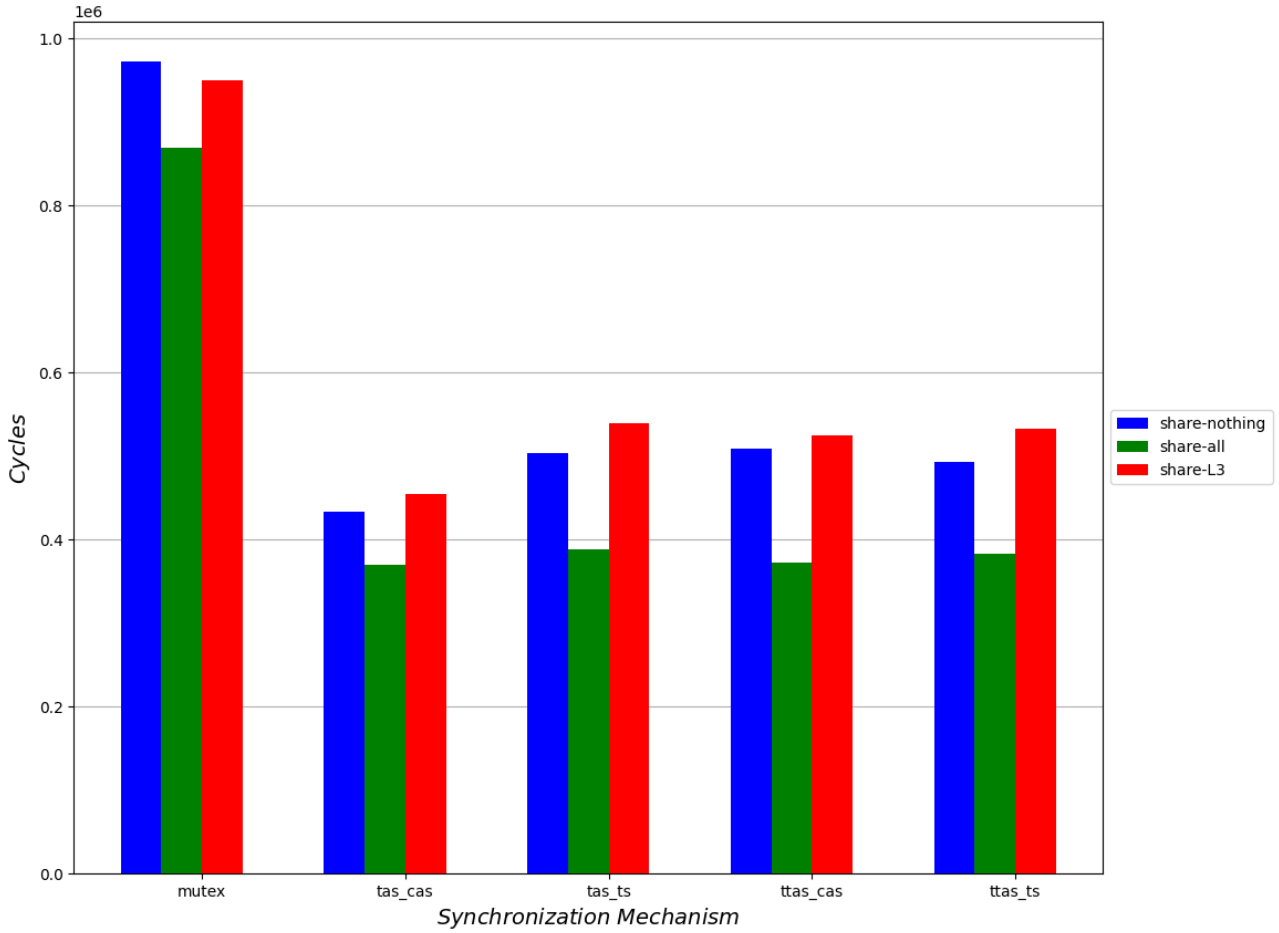
Τοπολογία νημάτων

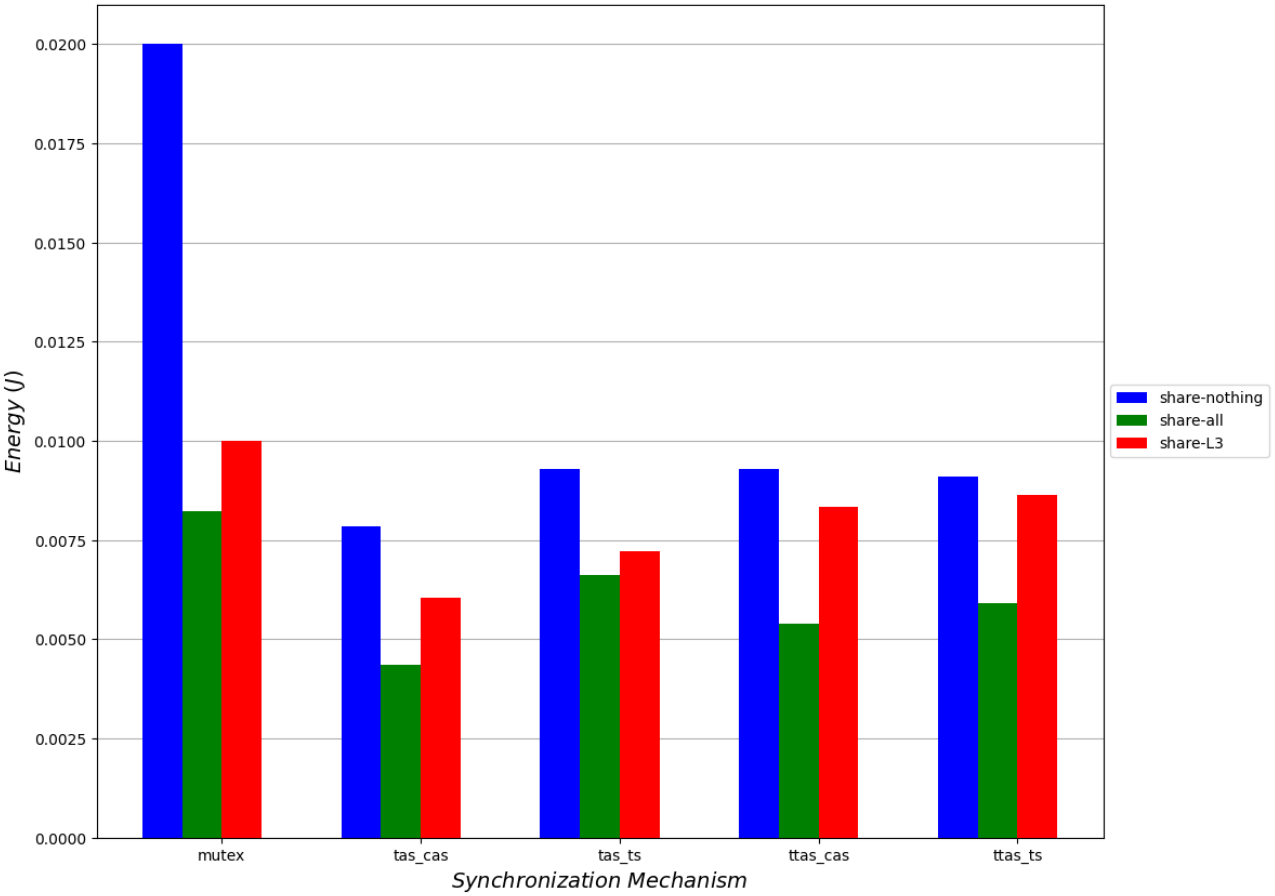
Συνολικός χρόνος εκτέλεσης και κατανάλωση ενέργειας

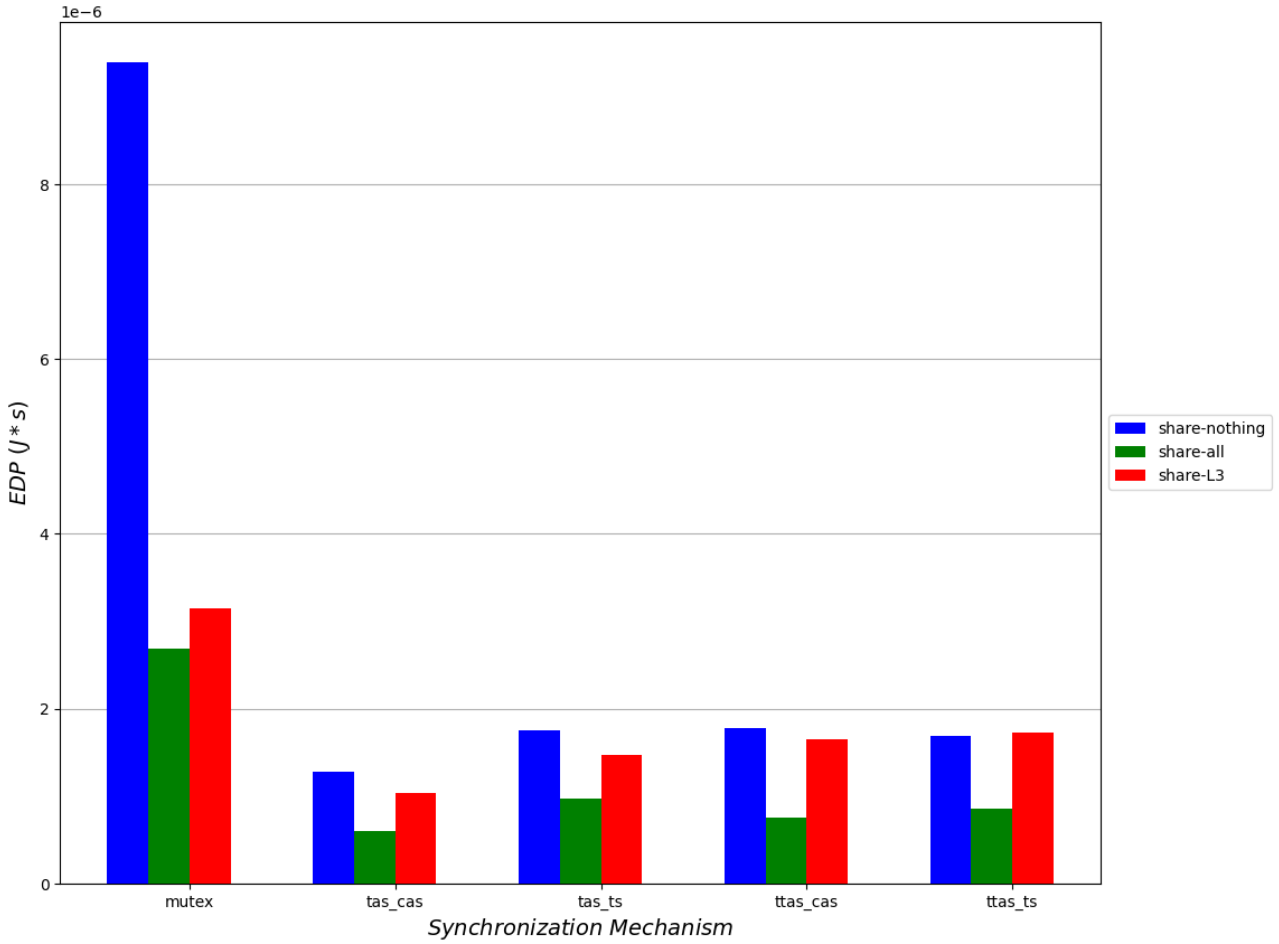
Ακολουθούν τα διαγράμματα για τις επιδόσεις διάφορων τοπολογιών νημάτων, ως προς τον διαμοιρσμό των κρυφών μνημών. Έχουμε τις παρακάτω 3 επιλόγές:

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

Τα 3 διαγράμματα αποτυπώνουν το συνολικό χρόνο εκτέλεσης, την ενέργεια και τη μετρική EDP για τις 3 διαφορετικές επιλογές και τους 5 διαφορετικούς μηχανισμούς.




0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Energy (J)
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Synchronization Mechanism
share-nothing
share-all
share-L3


1e−6
8
6
4
2
0
EDP (J * s)
mutex
tas_cas
tas_ts
ttas_cas
ttas_ts
Synchronization Mechanism
share-nothing
share-all
share-L3

Τοπολογία νημάτων

Συμπεράσματα για συνολικό χρόνο εκτέλεσης και κατανάλωση ενέργειας

Αρχικά, σχετικά με το συνολικό χρόνο εκτέλεσης φαίνεται καθαρά πως η καλύτερη επιλογή είναι η share-all. Αυτό πιθανόν να οφείλεται στη δυνατότητα να ενημερώνονται invalid blocks απευθείας από τη κοινή L2 και να μη χρειάζεται το σύστημα να ανεβαίνει πάνω στην αργότερα L3 η ακόμα και στην κύρια μνήμη σε περίπτωση share-nothing.

Οι μηχανισμοί tas_cas και ttas_cas φαίνεται να είναι οι καλύτεροι σε θέματα επιδόσεων, ακολουθούμενοι από τους tas_ts και ttas_ts και τελευταίο το mutex, φυσικά.

Σχετικά με την ενέργεια, όπως ήταν αναμενόμενο, η επιλογή share-nothing είναι αυτή που καταναλώνει την περισσότερη, ακολουθούμενη από την share-L3 και την οικονομικότερη share-all.

Όμοια φαίνονται τα αποτελέσματα και για τη μετρική EDP, με την share-nothing να έχει τεράστια διαφορά με τη δεύτερη share-L3 στο mutex μηχανισμό κυρίως, ενώ στα υπόλοιπα διατηρεί ένα μικρό overhead. Και πάλι η share-all φαίνεται να είναι η πιο αποδοτική επιλογή, με το μηχανισμό tas_cas να εμφανίζει την καλύτερη συμπεριφορά από όλους, ακολουθούμενος από τον tas_ts.