ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

4η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2019-2020, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τσαμπάζη Μαρία
Α.Μ. : 031 15716

Εισαγωγή

Στην παρούσα άσκηση μελετήσαμε διαφορετικούς μηχανισμούς συγχρονισμού σε σύγχρονες πολυπύρηνες αρχιτεκτονικές, καθώς και τα cache coherence protocols. Ειδικότερα, παρατηρήσαμε την απόδοσή τους ως προς τη (χρονική) κλιμακωσιμότητα (scalability) και την κατανάλωση ενέργειας, δλδ., ως προς της ενεργειακή συμπεριφορά τους. Για να παρατηρήσουμε αποτελεσματικά την απόδοσή τους, κάθε φορά αλλάζαμε τις παραμέτρους : μέγεθος κρίσιμου τμήματος, πλήθος νημάτων και την τοπολογία. Μελετάμε σύγχρονες πολυπύρηνες αρχιτεκτονικές, γι'αυτό και χρησιμοποιούμε για προσομοιωτή τον sniper simulator, ενώ για τη μελέτη της "ενεργειακής συμπεριφοράς" χρησιμοποιούμε το McPAT. Επιπλέον, κάθε φορά, ενεργοποιούσαμε κλήσεις κατάλληλα στους μηχανισμούς συγχρονισμού TAS, TTAS και Pthread MUTEX. Στο Β' Μέρος μελετάμε τον Αλγόριθμο Tomasulo.

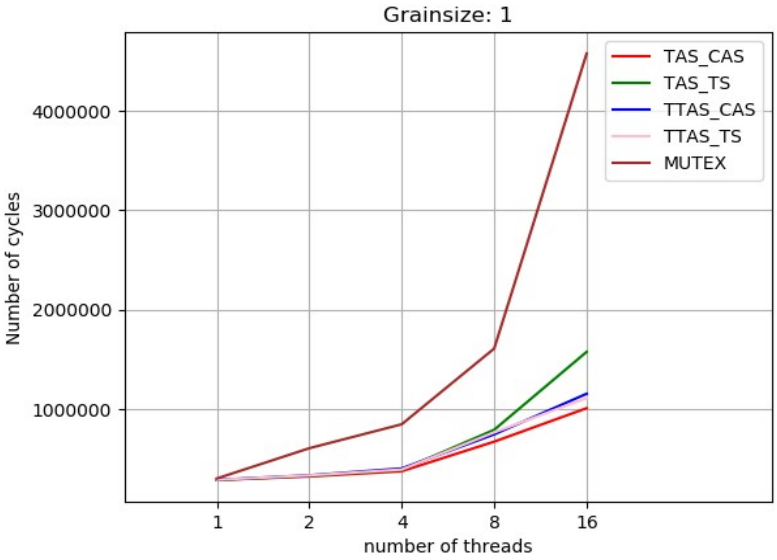
Α' ΜΕΡΟΣ

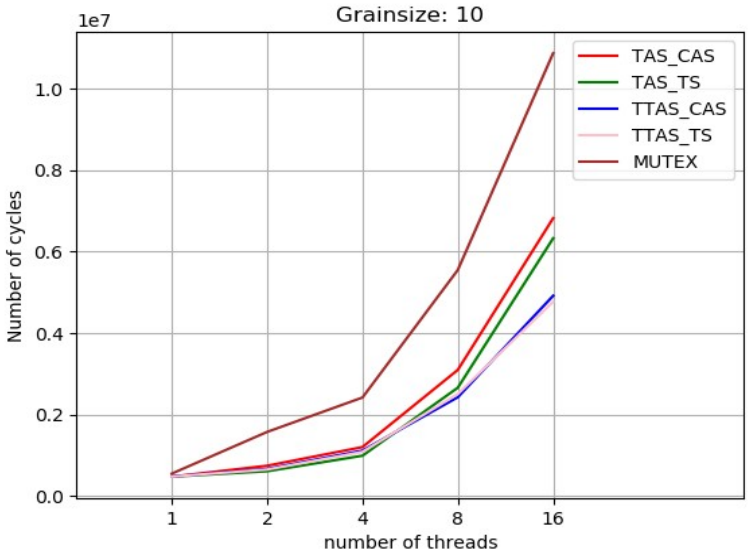
- Αξιολόγηση Επίδοσης

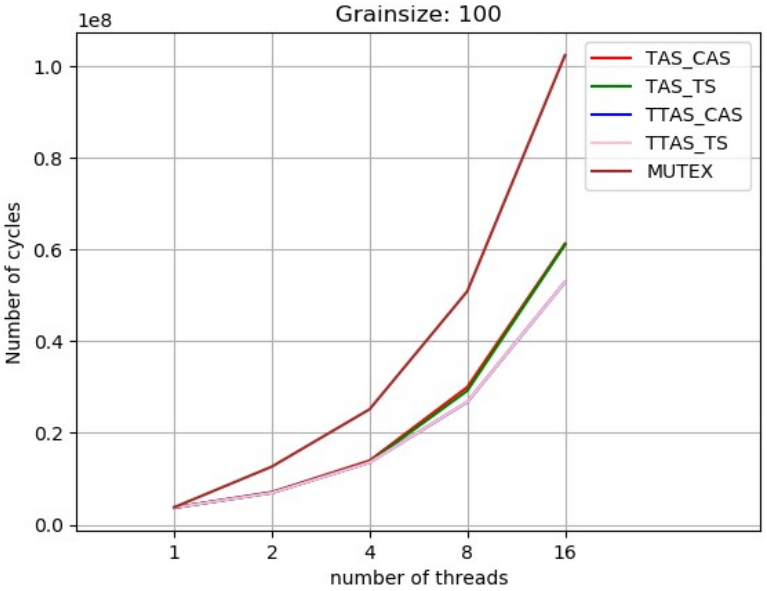
Αρχικά, καλούμαστε να υλοποιήσουμε τους μηχανισμούς κλειδώματος Test-and-Set (TAS) και Test-and-Test-and-Set (TTAS). Δίνοντας κατά τη μεταγλώττιση τα κατάλληλα flags, παράξαμε κώδικα που χρησιμοποεί κλήσεις σε κλειδώματα TAS, TTAS ή Pthread mutex (MUTEX). Επίσης, ορίσαμε σύμφωνα με την εκφώνηση:
nthreads: ο αριθμός των threads που θα δημιουργηθούν **= 1000**
iterations: ο αριθμός των επαναλήψεων που θα εκτελεστεί η κρίσιμη περιοχή από κάθε νήμα = **1, 2, 4, 8, 16** (σε σύστημα με ισάριθμους πυρήνες)
grain_size: καθορίζει τον όγκο των dummy, cpu-intensive υπολογισμών, και κατ' επέκταση το μέγεθος της κρίσιμης περιοχής = **1, 10, 100**

Στο ζητούμενο αυτό, αξιολογούμε και συγκρίνουμε την κλιμακωσιμότηταα των μηχανισμών TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS και Pthread mutex για αριθμούς νημάτων 1, 2, 4, 8, 16. Προσομοιώνουμε πολυπύρηνα συστήματα αντίστοιχου πλήθους πυρήνων και που υλοποιούν διαφορετικές πολιτικές διαμοιρασμού των caches. Κάθε τέτοιο σύστημα χρησιμοποιεί το MSI πρωτόκολλο συνάφειας κρυφής μνήμης.

3.1.1. - 3.1.2

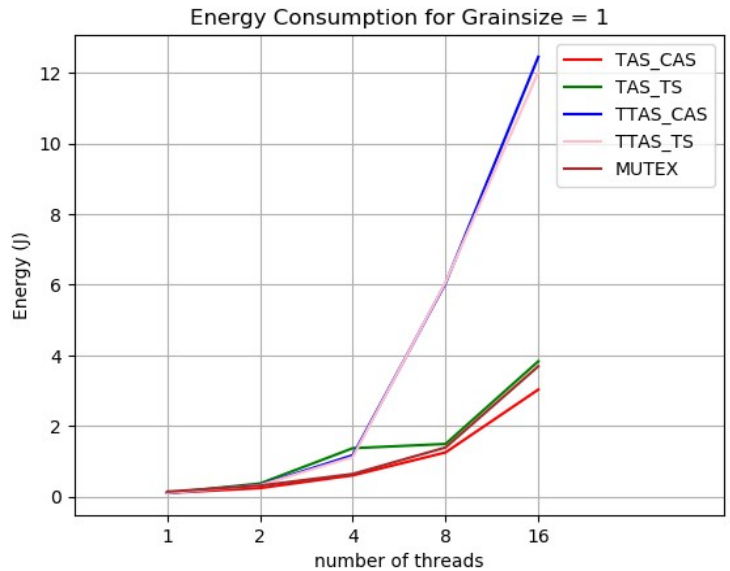


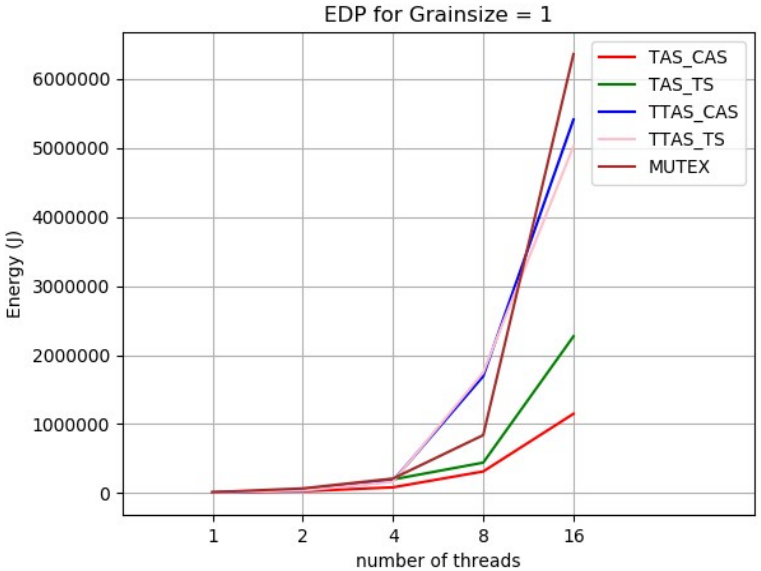


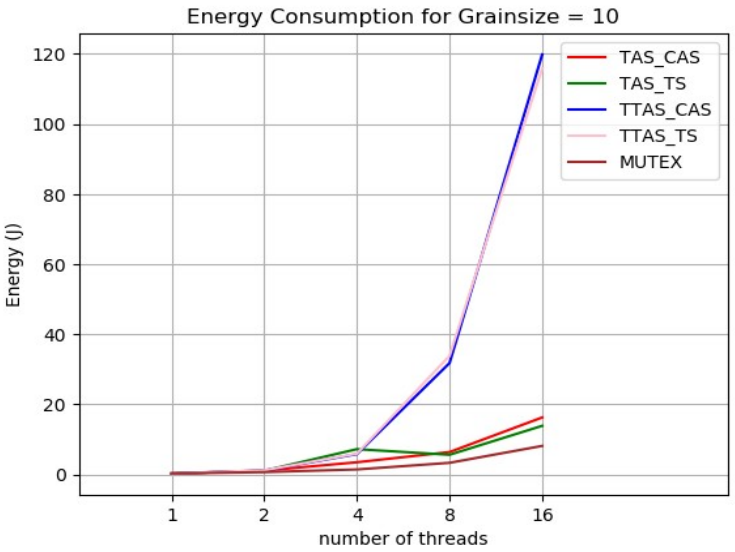


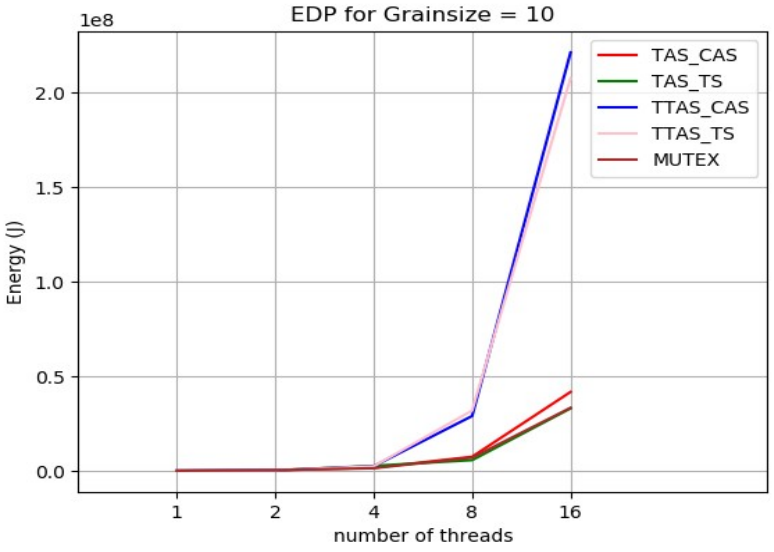
Ως προς την απόδοση, παρατηρούμε ότι το MUTEX αποδίδει αρκετά χειρότερα, δλδ. είναι πιο αργό, σε σχέση με τα spinlocks που εμείς υλοποιήσαμε, εξαιτίας του busy-waiting (switch μεταξύ των threads). Ακόμη, γενικά παρατηρούμε ότι τα αποτελέσματα των TTAS είναι πολύ κοντά, και έχουν σταθερά πάντα καλές επιδόσης, κάτι που φαίνεται καθώς αυξάνεται το grainsize, όπου αποδίδουν σχετικά καλύτερα από τα TAS, αφού γλιτώνουμε stores που δε χρειάζονται, επιτυγχάνουμε μείωση της κίνησης στο bus και συνεπώς καλύτερη αξιοποίηση της cache. Καθώς αυξάνεται το grainsize η απόδοση χειροτερεύει.

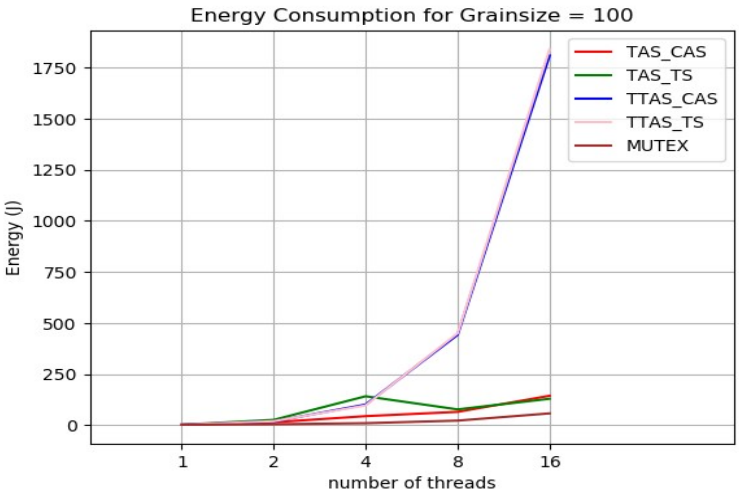
3.1.3

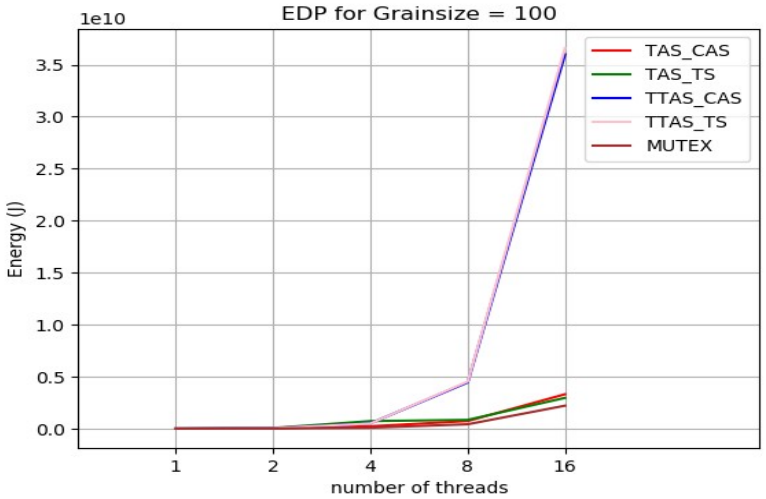










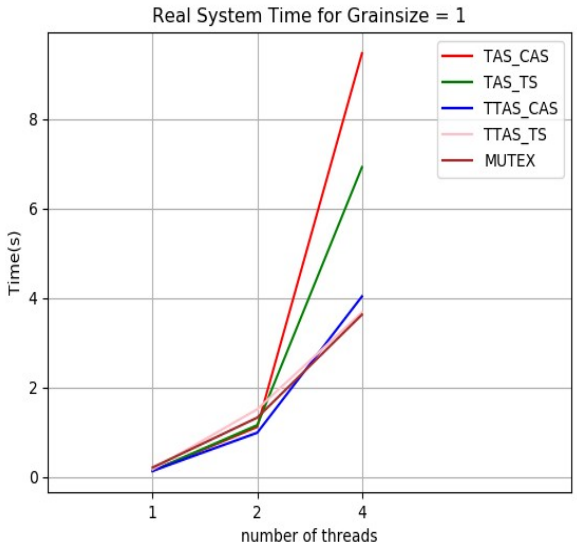


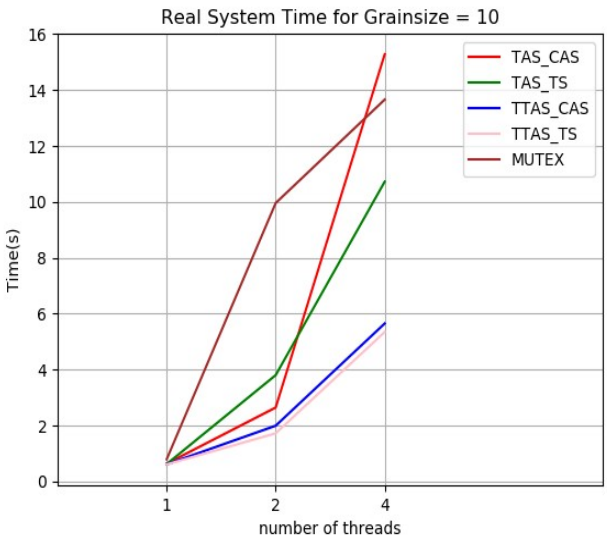
Μικρότερη κατανάλωση ενέργειας εμφανίζεται στα MUTEX, που είναι και τα πιο “αργά”. Οπότε παρατηρούμε το trade-off μεταξύ ταχύτητας-κατανάλωσης ενέργειας.

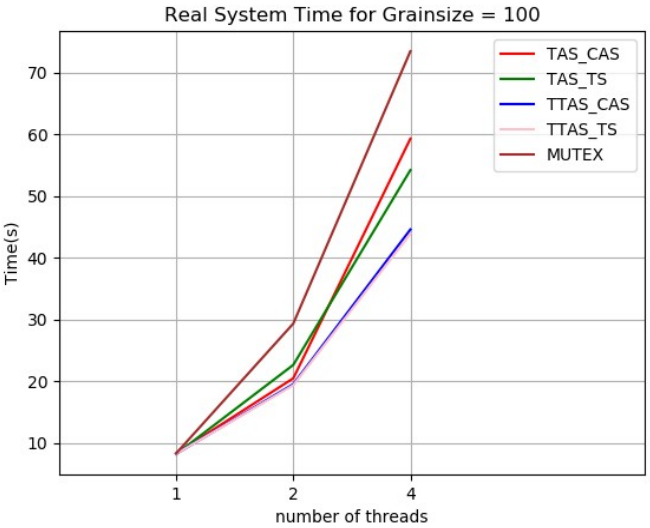
3.1.4

lscpu → CPUS(4), iterations =10000000.

Ο υπολογιστής μου διαθέτει 4 πυρήνες → μέγιστος αριθμός νημάτων : 4.







Για μικρό grainsize το MUTEX αποδίδει καλύτερα σε σχέση με τα spinlocks. Ενώ, πάλι, παρατηρούμε την καλύτερη απόδοση να έχουν τα TTAS.

-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:

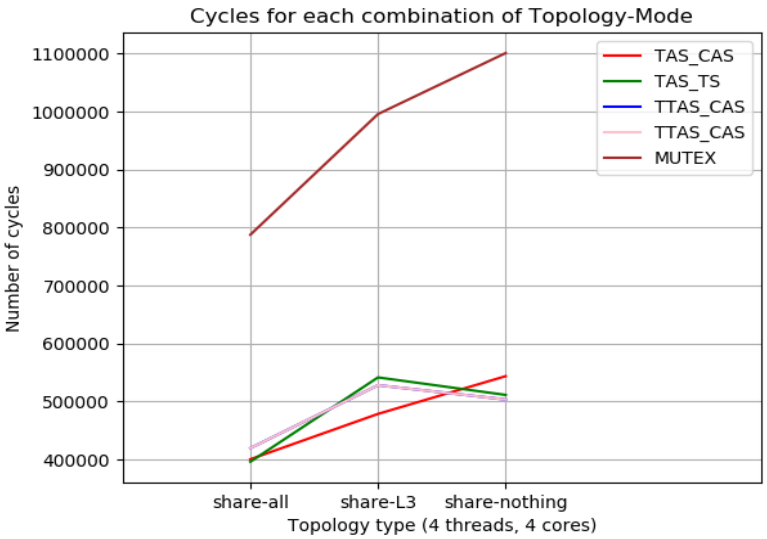
- εκδόσεις προγράμματος: TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS, MUTEX
- iterations: 1000
- nthreads: 4
- grain_size: 1

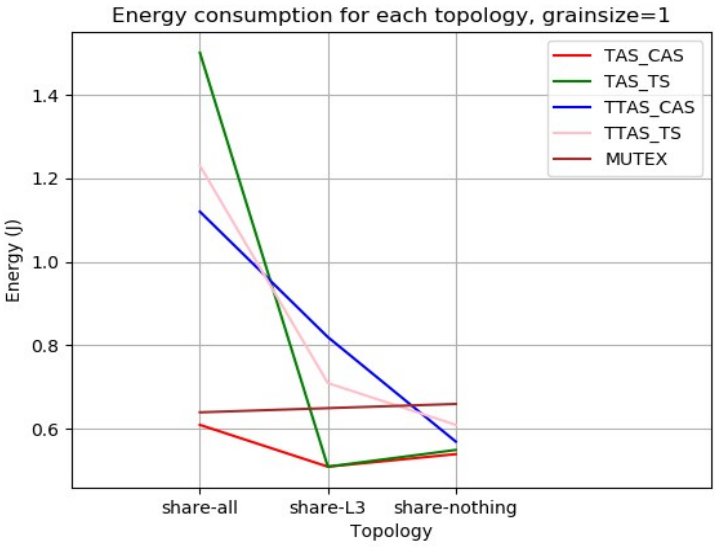
και εξετάζουμε τις ακόλουθες τοπολογίες:

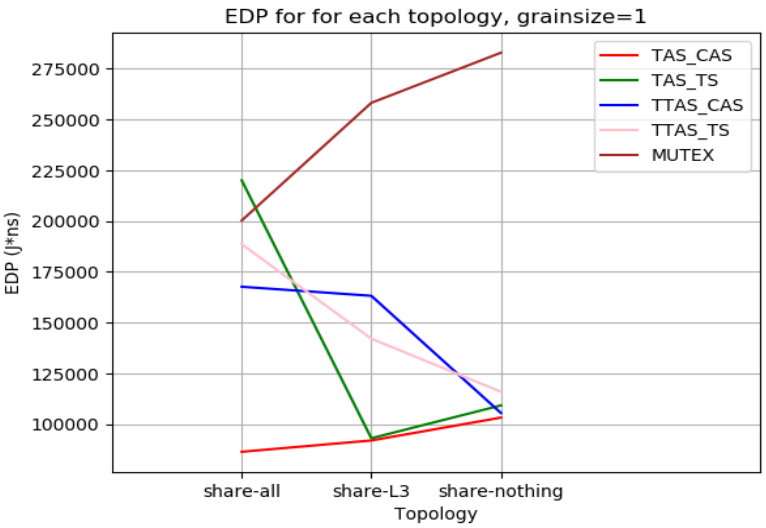
- ***share-all:*** και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L2 cache
- ***share-L3:*** και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L3 cache, αλλά όχι κοινή L2
- ***share-nothing:*** και τα 4 νήματα βρίσκονται σε πυρήνες με διαφορετική L3 cache

Για 4 πυρήνες και ισάριθμο πλήθος νημάτων με διαφορετική όμως τοπολογία, με τον sniper για τους μηχανισμούς TAS, TTAS και MUTEX, έχουμε:

3.2.1







Η τοπολογία share_all είναι η πιο γρήγορη, αλλά κοστίζει σε ενέργεια. Ο διαμοιρασμός των δεδομένων βοηθά στην αύξηση της ταχύτητας, αλλά το υλικό κοστίζει .

Β' ΜΕΡΟΣ

```
LOOP: LD    F0, 0(R1)     0
      ADDD  F4, F4, F0    1
      LD    F1, 0(R2)     2
      MULD  F4, F4, F1    3
      ANDI  R9, R8, 0x2   4
      BNEZ  R9, NEXT      5
IF:   LD    F2, 16(R2)    6
      MULD  F2, F2, F5    7
      ADDD  F4, F4, F2    8
NEXT: LD    F5, 8(R1)     9
      ADDD  F4, F4, F5    10
      ADDI  R1, R1, 0x8   11
      SUBI  R8, R8, 0x1   12
      BNEZ  R8, LOOP      13
      SD    F4, 8(R2)     14
```

OP	IS	EX	WR	CMT	COMMENTS
0	1	2-5	6	7	CACHE MISS
1	1	7-9	10	11	RAW F0
2	2	3-6	7	11	CACHE MISS
3	2	11-15	16	17	RAW F1, RAW F4
4	3	4-5	8	17	cdb

5	3	9-10	11	18	branch prediction: outcome = NT, prediction = T, RAW R9
9	7	8-8	9	-1	cache hit, full rs
10	7	-1--1	-1	-1	FU busy, RAW F4, RAW F5
11	8	9-10	-1	-1	
12	9	10-11	-1	-1	full rs
6	12	13-16	17	18	CACHE MISS
7	12	18-22	23	24	FU busy, RAW F2
8	13	24-26	27	28	RAW F2
9	13	14-14	15	28	cache hit
10	14	28-30	31	32	FU busy, RAW F4, RAW F5
11	14	15-16	18	32	cdb
12	18	19-20	21	33	full rob
13	18	22-23	24	33	branch prediction: outcome = NT, prediction = T, RAW R8
0	19	20-20	22	-1	cache hit, cdb
1	19	-1--1,	-1	-1	FU busy, RAW F0, RAW F4
14	25	32-35	36	37	CACHE MISS, RAW F4