ΑΣΚΗΣΗ 4

Με θέμα

ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΟΥΣ ΜΗΧΑΝΙΣΜΟΥΣ ΣΥΓΧΡΟΝΙΣΜΟΥ ΚΑΙ ΤΑ ΠΡΩΤΟΚΟΛΛΑ ΣΥΝΑΦΕΙΑΣ ΚΡΥΦΗΣ ΜΝΗΜΗΣ

για το μάθημα

Προηγμένα Θέματα
Αρχιτεκτονικής Υπολογιστών

Γεώργιος Σκουρτσίδης (03114307)

ΜΕΡΟΣ Α

ΕΙΣΑΓΩΓΗ

Στόχος της άσκησης αυτής είναι η εξοικείωση με τους μηχανισμούς συγχρονισμού και τα πρωτόκολλα συνάφειας κρυφής μνήμης (cache coherence protocols) σε σύγχρονες πολυπύρηνες αρχιτεκτονικές.Για τις απαιτήσεις της άσκησης,υλοποιήθηκαν οι μηχανισμοί κλειδώματος Test-and-Set (TAS) και Test-and-Test-and-Set (TTAS), σύμφωνα με τις διαφάνειες του μαθήματος.

Σε κάθε προσομοίωση που εκτελέστηκε με χρήση του εργαλείου sniper υπήρξε το ακόλουθο πρόβλημα :

```
Thread 0 stopped with dummy result: 1.000000
[SIFT_RECORDER] emulation.cc:41: void handleCpuid(LEVEL_VM::THREADID, LEVEL_VM::PIN_REGISTER*,
 LEVEL_VM::PIN_REGISTER*, LEVEL_VM::PIN_REGISTER*, LEVEL_VM::PIN_REGISTER*): Assertion `emulat
ed' failed.
Pin app terminated abnormally due to signal 6.
```

Πρόκειται για το ίδιο error που αναφέρεται και στην εκφώνηση της εργασίας πως είναι πιθανό να προκύψει.Όμως στην πλειοψηφία των περιπτώσεων τα αρχεία sim.out δημιουργήθηκαν κανονικά και παρά τον μικρό χρόνο προσομοίωσης σε ορισμένες περιπτώσεις, τα αποτελέσματα είναι λογικά. Συνεπώς,η άσκηση εκτελέσθηκε κανονικά,όπως άλλοστε διαβάζουμε και στις σχετικές οδηγίες που μας δίνονται.

ΕΡΩΤΗΜΑ 1.1

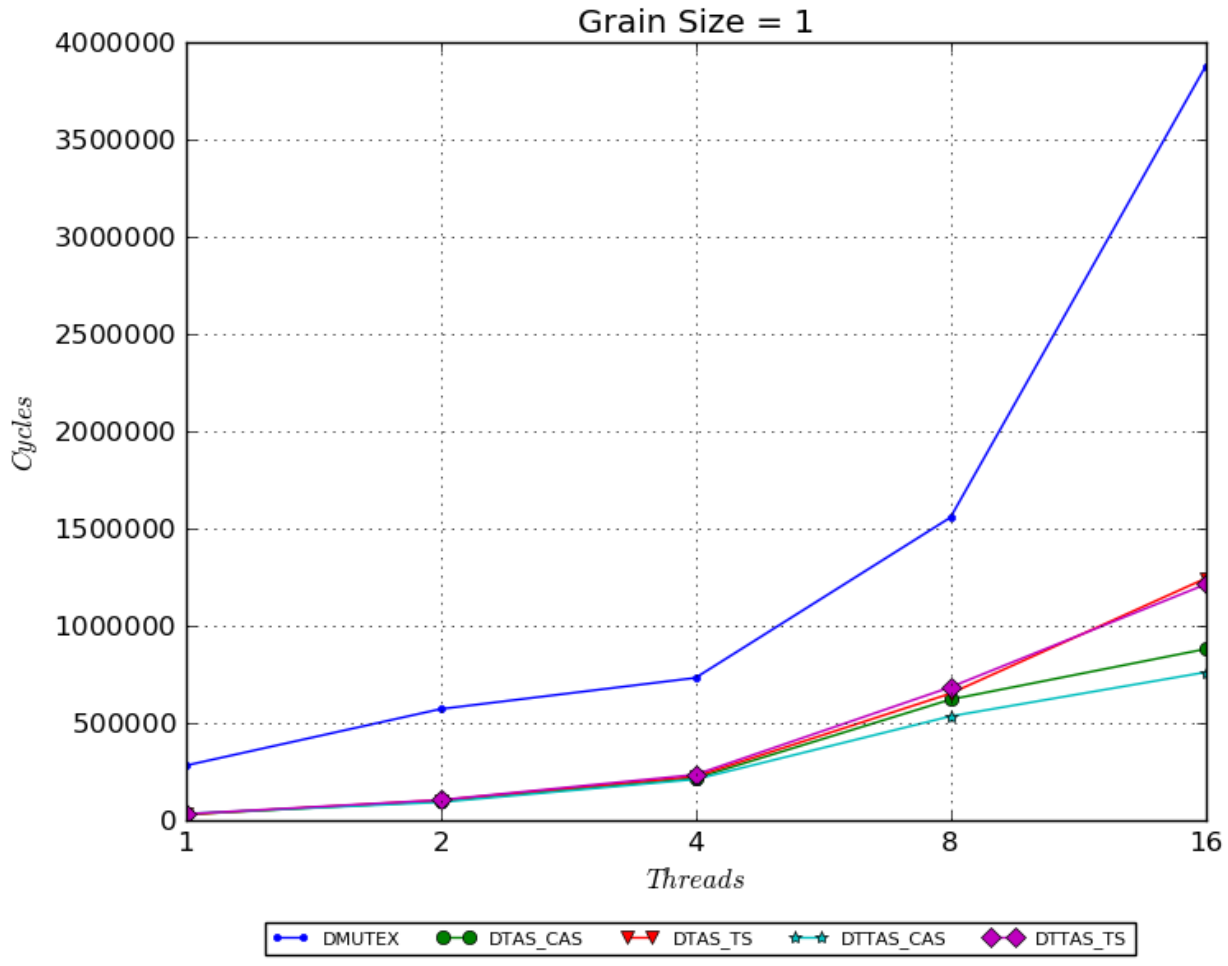
Εκφώνηση

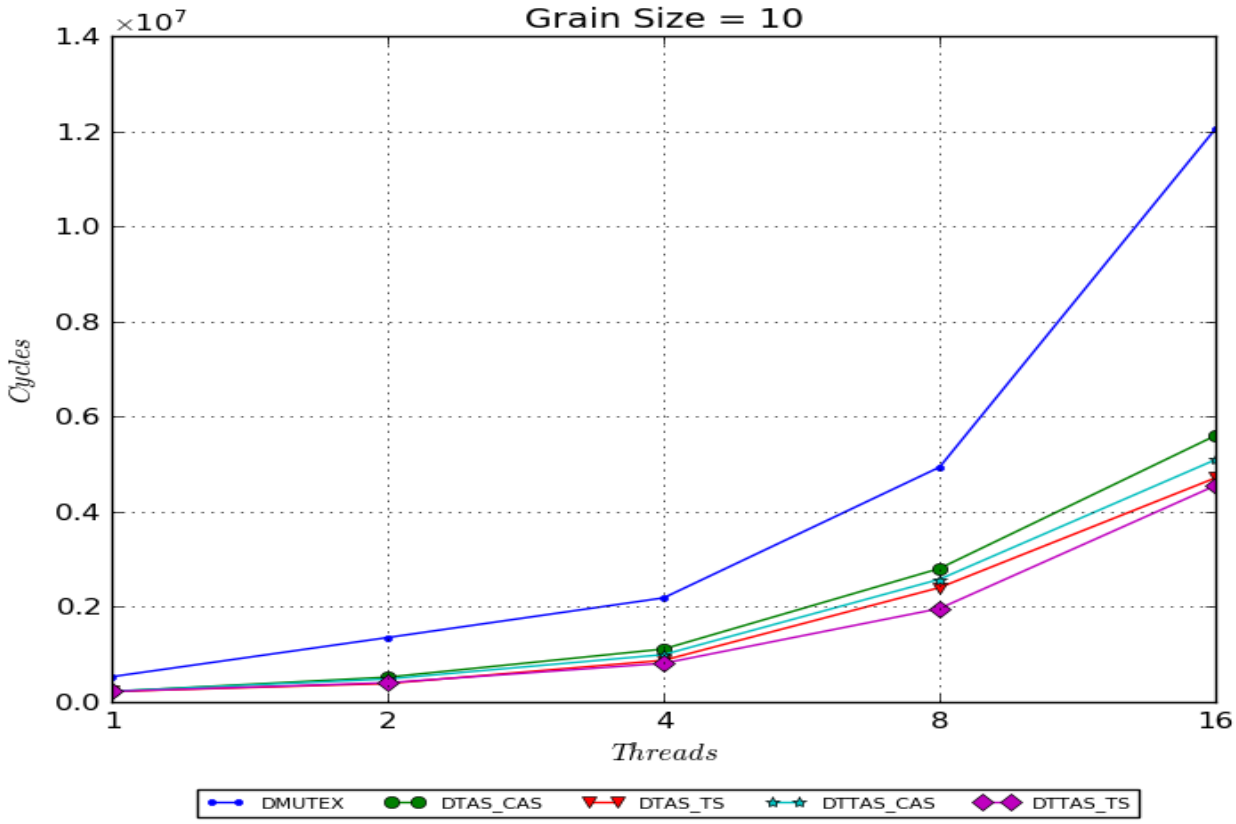
Για κάθε grain size, δώστε το διάγραμμα της κλιμάκωσης του συνολικού χρόνου εκτέλεσης της περιοχής ενδιαφέροντος σε σχέση με τον αριθμό των νημάτων. Συγκεκριμένα, στον x-άξονα θα πρέπει να έχετε τον αριθμό των νημάτων και στον

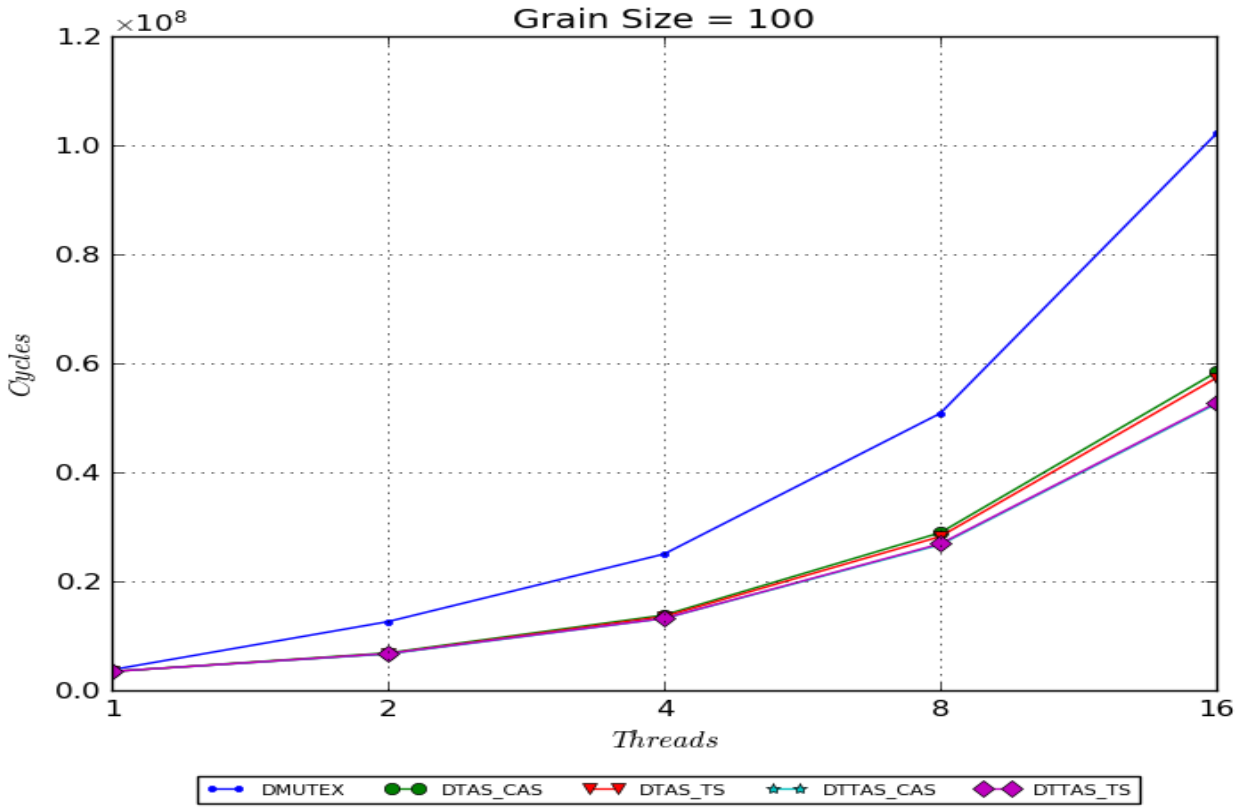
y-άξονα τον χρόνο εκτέλεσης σε κύκλους. Στο ίδιο διάγραμμα θα πρέπει να συμπεριλάβετε τα αποτελέσματα και για τις 5 εκδόσεις.

Λύση

Τα γραφήματα παρουσιάζονται παρακάτω.Στον x άξονα μερικά νούμερα είναι πολλαπλασιασμένα με τη δύναμη του δέκα που αναγράφεται πάνω αριστερά στην κορυφή του άξονα.




Grain Size = 10
×10^7
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
1
2
4
8
16
Threads
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS


Grain Size = 100
×10^8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Cycles
1
2
4
8
16
Threads
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS

ΕΡΩΤΗΜΑ 1.2

Εκφώνηση

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.

Απάντηση

Καθώς αυξάνεται το μέγεθος του grain size παρατηρείται αύξηση των κύκλων,ανεξάρτητα από το κλείδωμα που χρησιμοποιείται.Δεκαπλασιασμός του grain size συνεπάγεται περίπου δεκαπλασιασμό και στους εκτελούμενους κύκλους,όπως βλεπουμε στα διαγράμματα.

Σχετικά,με τον αριθμό των threads,παρατηρούνται μικρές διαφορές όταν τα threads είναι λίγα στον συνολικό αριθμό των κυκλων για τα κλειδώματα TAS και TTAS.Καθώς όμως ο συνολικός αριθμός των νημάτων αυξάνεται,οι TAS υλοποιήσεις έχουνε μειωμένη αποδοτικότητα.Αυτό συμβαίνει διότι κάθε φορά που ένα thread προσπαθεί να πάρει το "κλειδί"(lock),γράφει στη θέση μνήμης του κλειδιού.Όταν έχουμε πολλά threads,τότε κάνουν ανεπάλληλα invalidations της cache line που περιέχει το lock.Συνεπώς,μεγαλύτερος αριθμός νημάτων οδηγεί σε αυξημένη κίνηση στον δίαυλο.

Στην TTAS υλοποίηση,το κάθε thread διαβάζει την τιμή του lock αλλά δεν γράφει στη θέση μνήμης του κλειδιού,δηλαδή δεν επιχειρεί να το πάρει.Μόνο όταν το thread που βρίσκεται μέσα στο critical section απελευθερώσει το lock (δηλαδή βγει από την κρίσιμη περιοχή),μόνο τότε το εκάστοτε thread-διεκδικητής επιχειρεί να πάρει το lock.Τα thread-διεκδηκητές διαβάζουν συνεχώς τις τιμές του lock που βρίσκεται στις caches τους και δεν εισάγουν περιττή κίνηση στο δίαυλο.Όσο τα νήματα αυξάνουν,τόσο περισσότερα γίνονται τα νήματα που θα είναι κάθε φορά εκτός κρίσιμης περιοχής και άρα θα προκαλούν καθυστερήσεις στις TAS υλοποιήσεις.

Συγκρίνοντας τις υλοποιήσεις 'test-and-set' και 'compare-and-swap,βλέπουμε την δεύτερη να χρειάζεται λιγότερους κύκλους για την ολοκλήρωσή της.Η εξήγηση έγκειται στο ότι το test-and-set τροποποιεί το περιεχόμενο μιας θέσης μνήμης και επιστρέφει την παλιά του τιμή σε κάθε επανάληψη,ενώ το compare-

and-swap συγκρίνει ατομικά τα περιεχόμενα μιας θέσης μνήμης με μια δεδομένη τιμή και, μόνο εάν είναι ίδια, τροποποιεί το περιεχόμενο αυτής της θέσης μνήμης σε μια νέα τιμή.

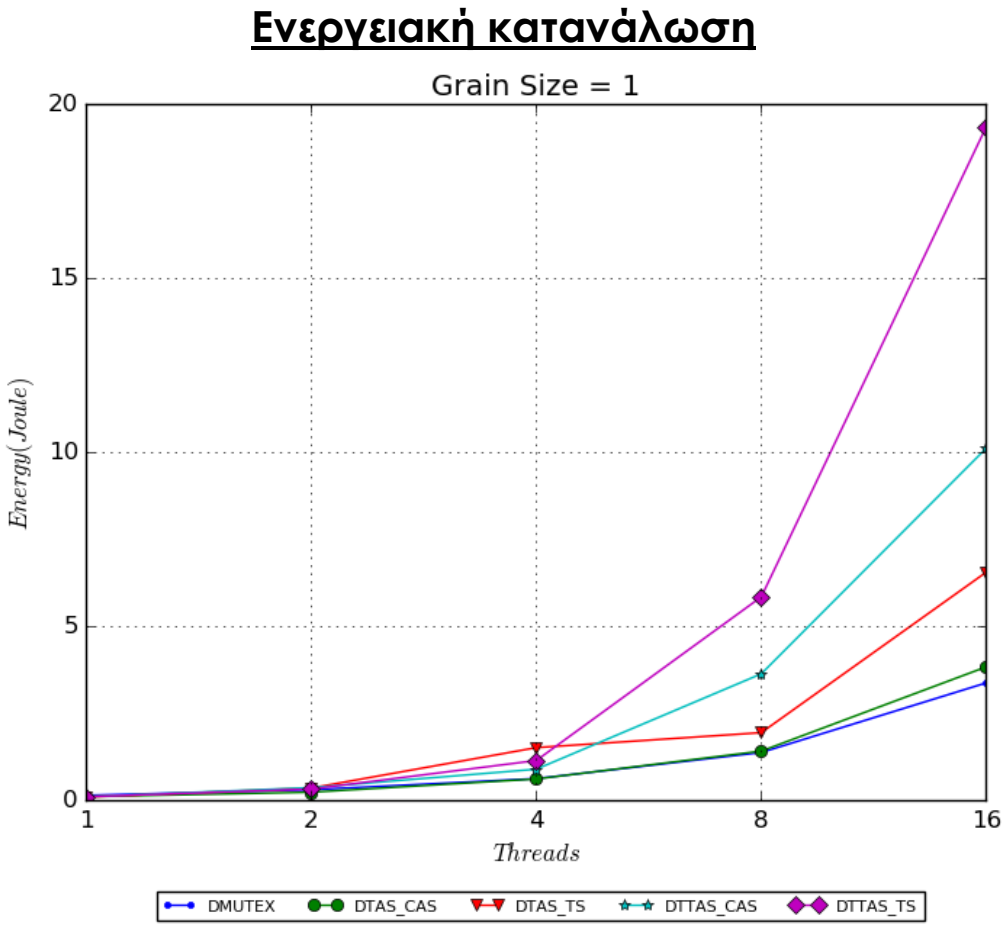
Από όλες τις υλοποιήσεις τα mutexes δίνουν τα χειρότερα αποτελέσματα.Αυτό συμβαίνει διότι όποτε προσπαθούμε να πάρουμε ένα lock σε ένα ήδη κλειδωμένο mutex, πρόκειται να προκαλέσουμε ένα context-switch μεταξύ νημάτων.

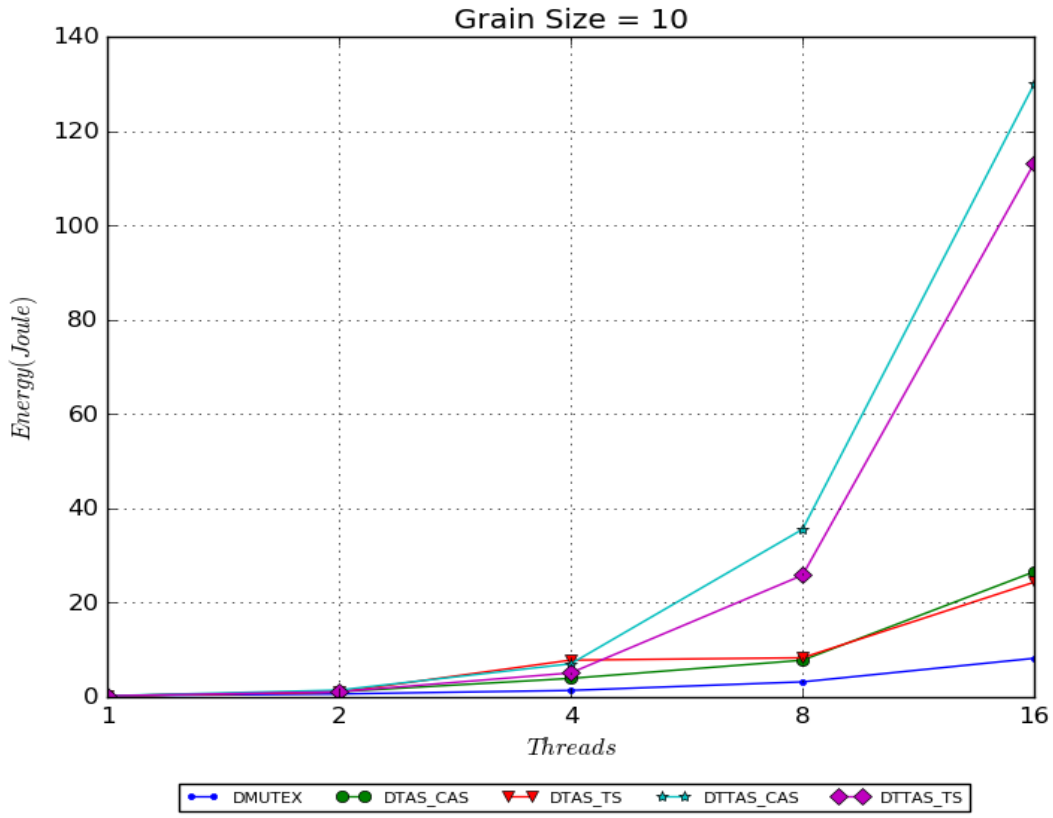
ΕΡΩΤΗΜΑ 1.3

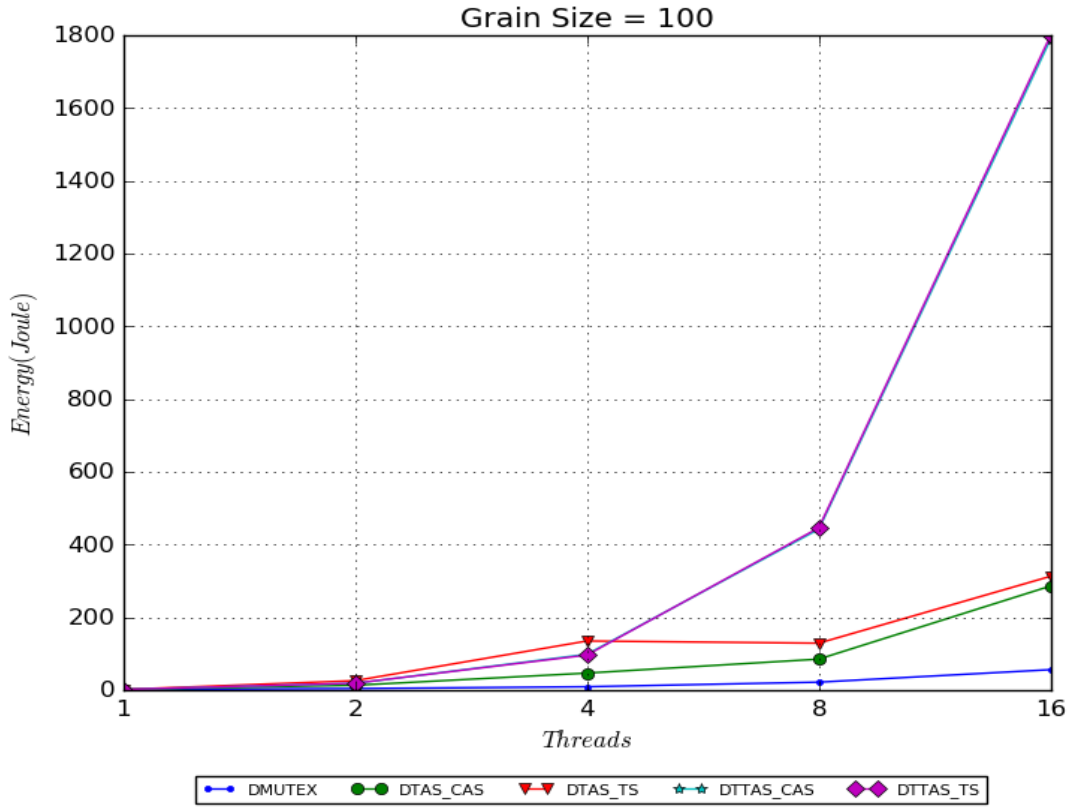
Εκφώνηση

Χρησιμοποιώντας όπως και στην προηγούμενη άσκηση το McPAT, συμπεριλάβετε στην ανάλυση σας εκτός από το χρόνο εκτέλεσης και την κατανάλωση ενέργειας (Energy, EDP κτλ.)

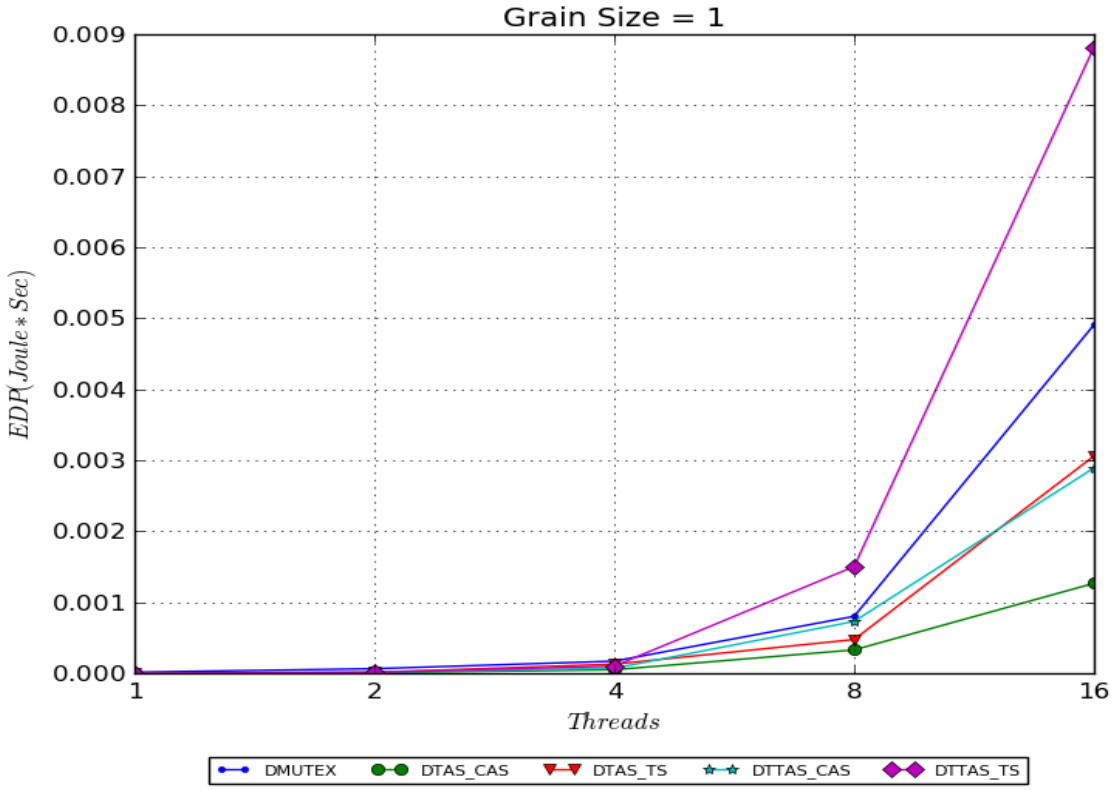
Απάντηση

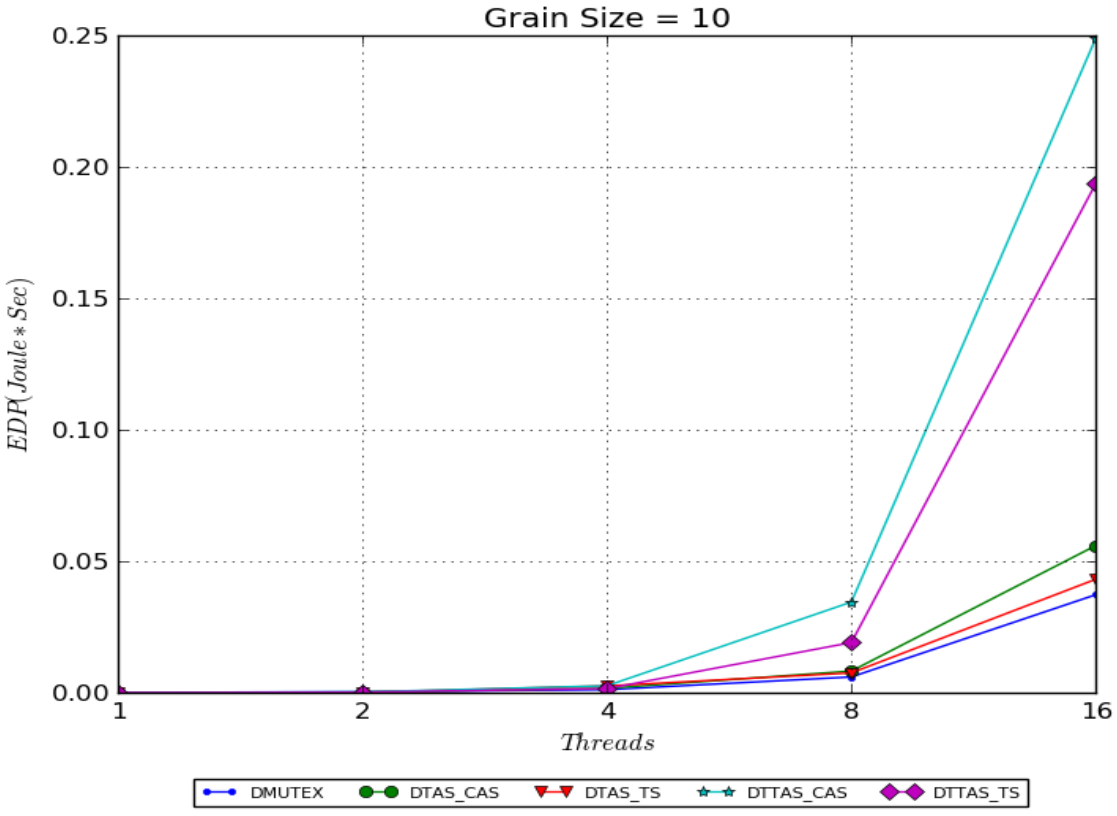


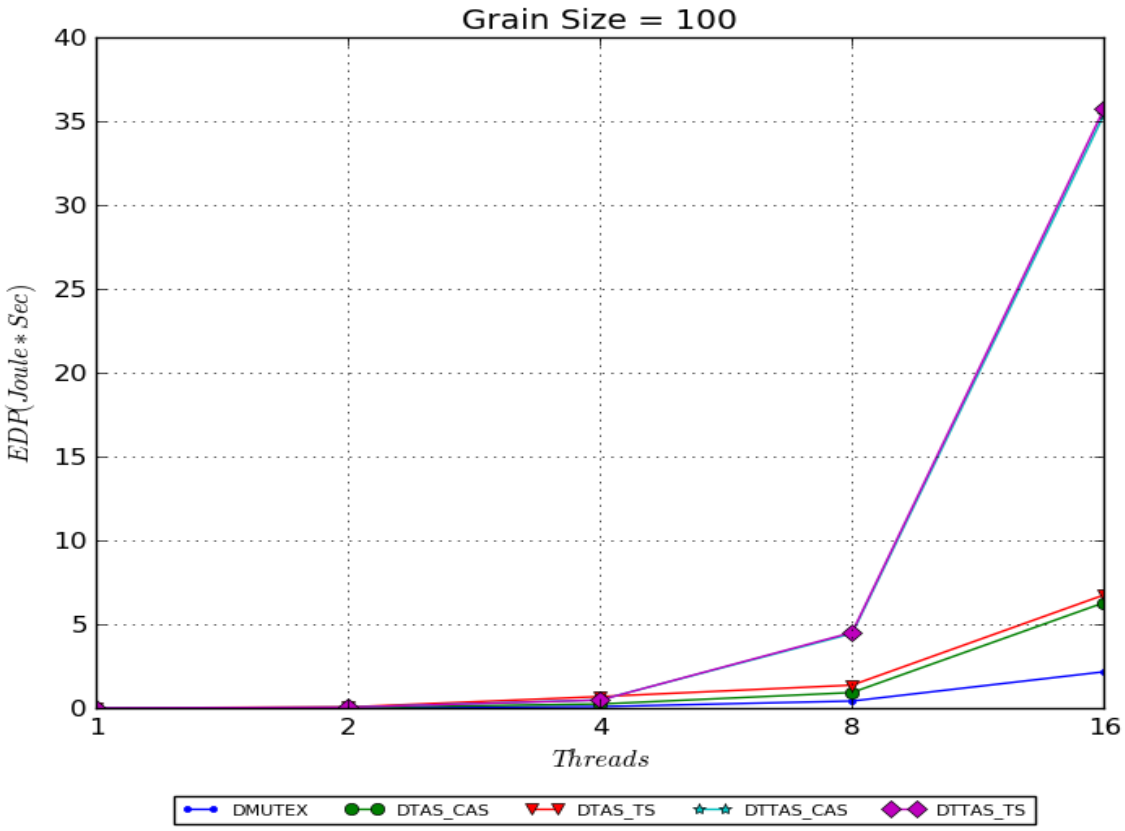

Grain Size = 10
Energy(Joule)
Threads
0
20
40
60
80
100
120
140
1
2
4
8
16
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS


Grain Size = 100
Energy(Joule)
Threads
0
200
400
600
800
1000
1200
1400
1600
1800
1
2
4
8
16
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS

EDP


Grain Size = 1
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
EDP(Joule * Sec)
1
2
4
8
16
Threads
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS


Grain Size = 10
0.25
0.20
0.15
0.10
0.05
0.00
EDP(Joule * Sec)
1
2
4
8
16
Threads
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS



Σχολιασμός αποτελεσμάτων

Χειρότερα αποτελέσματα μας δίνει το DTTAS_TS , με εξαίρεση το grain size = 10 όπου κυριαρχεί το DTTAS_CAS.Το κλείδωμα TTAS έχει τη μεγαλύτερη ενεργειακή κατανάλωση καθώς αποσχολεί συνεχώς τον επεξεργαστή με busy wait.Η μικρότερη κατανάλωση παρατηρείται στο mutex κλείδωμα.Στο EDP βλέπουμε πως το mutex έχει (συγκριτικά με τα energy διαγραμμάτα) υψηλότερες τιμές ως προς τα άλλα κλειδώματα,εξαιτίας του υψηλότερου χρόνου εκτέλεσής του.Και πάλι όμως η χαμηλή του κατανάλωση υπερνικά τον παραπάνω χρόνο εκτέλεσης.

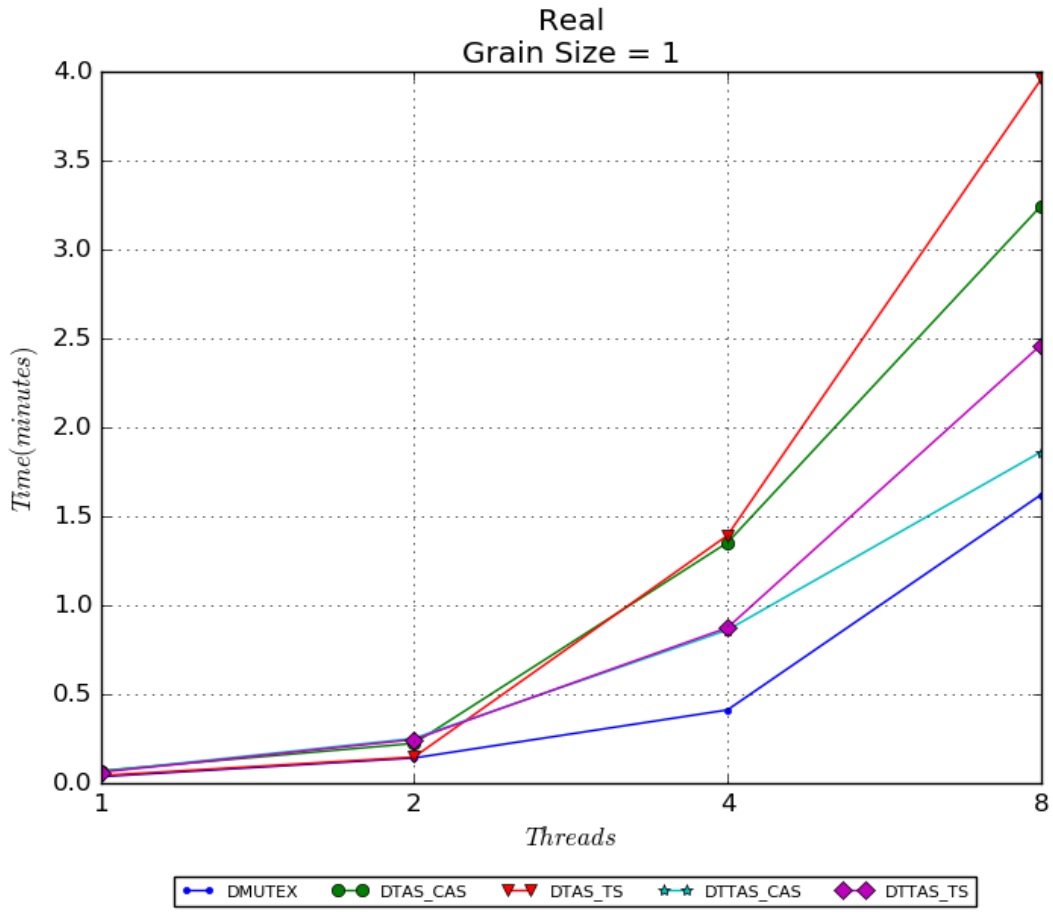
ΕΡΩΤΗΜΑ 1.4

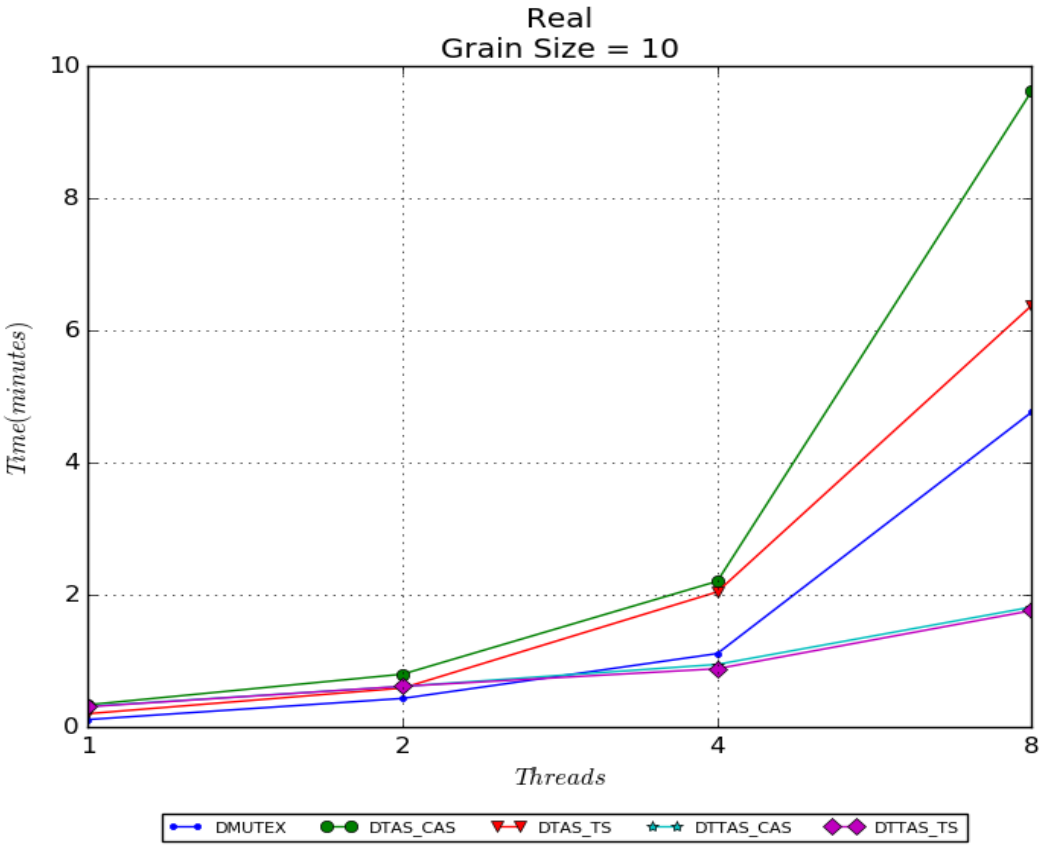
Εκφώνηση

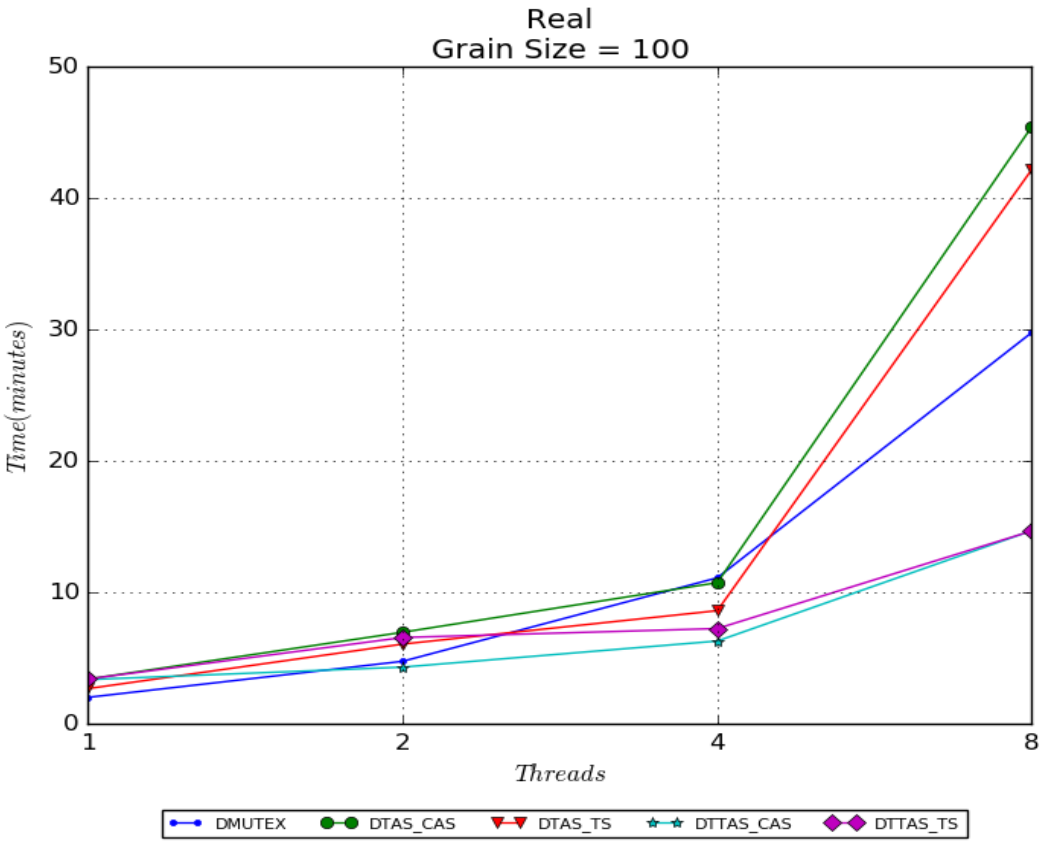
Μεταγλωττίστε τις διαφορετικές εκδόσεις του κώδικα για πραγματικό σύστημα. Εκτελέστε τα ίδια πειράματα με πριν σε ένα πραγματικό σύστημα που διαθέτει πολλούς πυρήνες. Χρησιμοποιήστε τους ίδιους αριθμούς νημάτων (με μέγιστο αριθμό νημάτων ίσο με τον αριθμό των πυρήνων που διαθέτει το μηχάνημά σας) και τα ίδια grain sizes με πριν.

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

Απάντηση




Real
Grain Size = 10
Time(minutes)
Threads
0
2
4
6
8
10
1
2
4
8
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS


Real
Grain Size = 100
Time(minutes)
Threads
0
10
20
30
40
50
1
2
4
8
DMUTEX
DTAS_CAS
DTAS_TS
DTTAS_CAS
DTTAS_TS

Σχολιασμός αποτελεσμάτων

Το σύστημα στο οποίο εκτελέστηκαν οι προσομοιώσεις ήταν ένα 4-πύρηνο VM με 8 threads (hyperthreading).Περισσότερες πληροφορίες φαίνονται παρακάτω.

```
giorgosskourtsidis@askisi4:~$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      2
Core(s) per socket:      4
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   63
Model name:              Intel(R) Xeon(R) CPU @ 2.30GHz
Stepping:                0
CPU MHz:                 2300.000
BogoMIPS:                4600.00
Hypervisor vendor:       KVM
Virtualization type:     full
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                46080K
NUMA node0 CPU(s):       0-7
Flags:                   fpu vme de pse tsc msr pae mce c
b rdtscp lm constant_tsc rep_good nopl xtopology nons
 popcnt aes xsave avx f16c rdrand hypervisor lahf_lm
invpcid xsaveopt arat md_clear arch_capabilities
```

Παρατηρούνται έντονες διαφορές συγκριτικά με την εκτέλεση σε προσομοιωμένο σύστημα.Δε γίνεται να εξάγουμε γρήγορα συμπεράσματα για την αποτελεσματικητα των κλειδωμάτων σε αυτήν την περίπτωση,διότι καθώς το grain sizei αυξάνει,η αποδοτικότητα κάθε κλειδώματος ποικίλλει.Πολύ καλά αποτελέσματα δίνουν τα κλειδώματα DDTAS_CAS και DDTAS_TS με εξαίρεση το grain size = 1 όπου το mutex είνα ελαφρώς καλύτερο του DDTAS_CAS.DTAS_CAS

κλείδωμα δίνει τα χειρότερα αποτελέσματα με εξαίρεση το grain size = 1 όπου το DTAS_TS είναι το χείριστο lock.Συνεπώς, σε αυτήν την περίπτωση η "Test and Test And Set" στρατηγική είναι σχεδόν μονόδρομος.

ΕΡΩΤΗΜΑ 2.1

Εκφώνηση

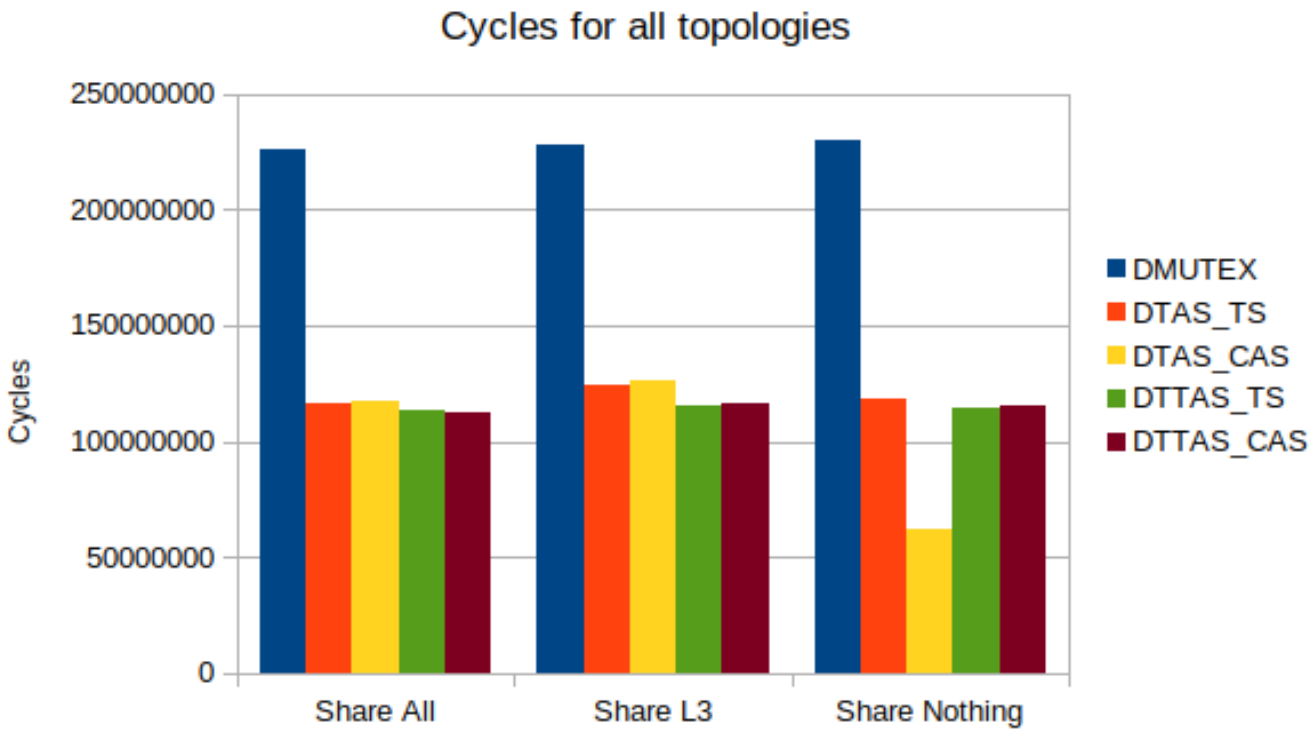
Στόχος του ερωτήματος αυτού είναι η αξιολόγηση της κλιμάκωσης των διαφόρων υλοποιήσεων όταν τα νήματα εκτελούνται σε πυρήνες με διαφορετικά χαρακτηριστικά ως προς το διαμοιρασμό των πόρων.

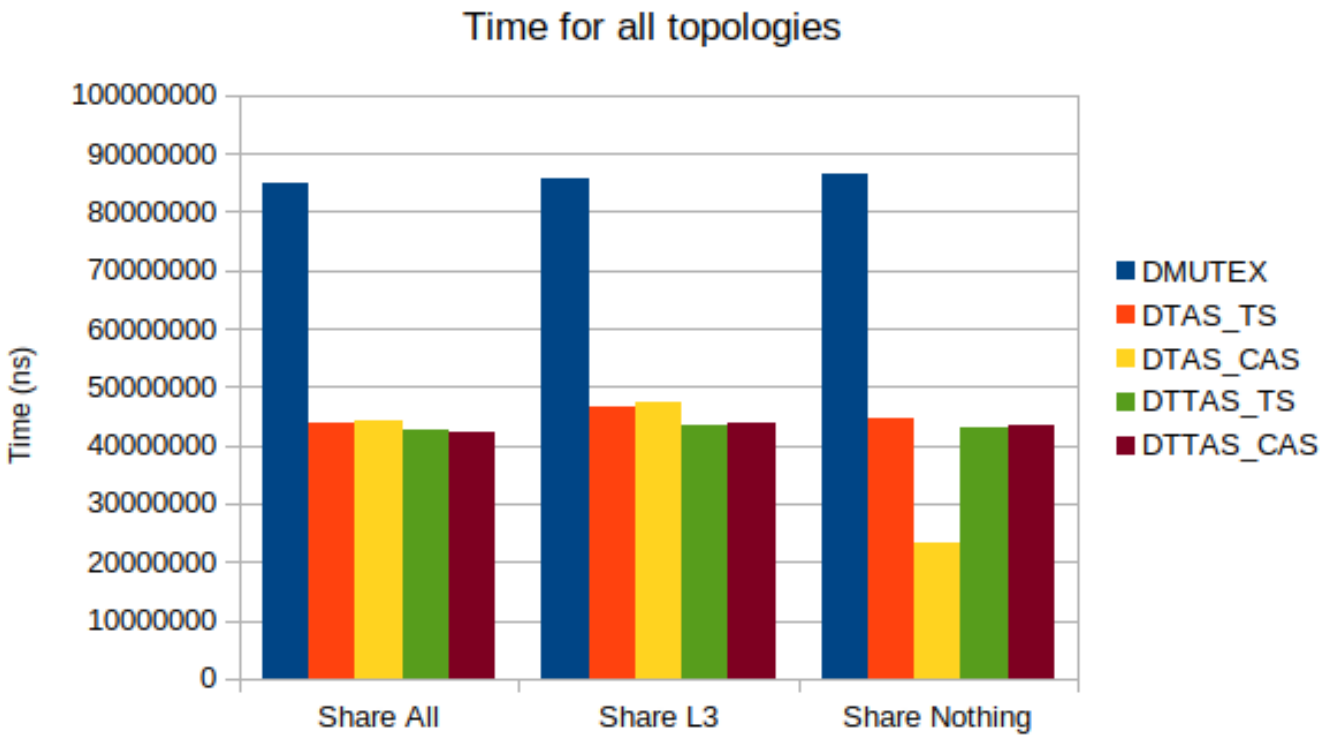
Εξετάζουμε τις ακόλουθες τοπολογίες:

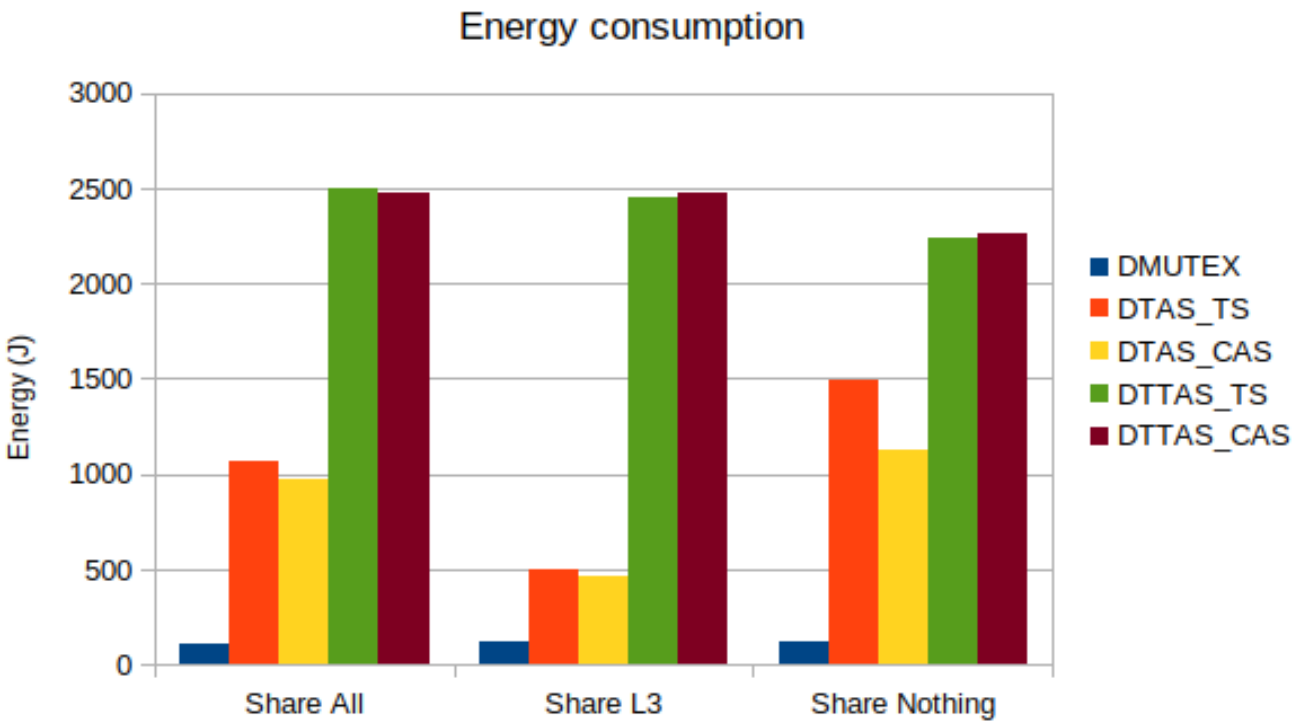
1. share-all: και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L2 cache
2. share-L3: και τα 4 νήματα βρίσκονται σε πυρήνες με κοινή L3 cache, αλλά όχι κοινή L2
3. share-nothing: και τα 4 νήματα βρίσκονται σε πυρήνες με διαφορετική L3 cache

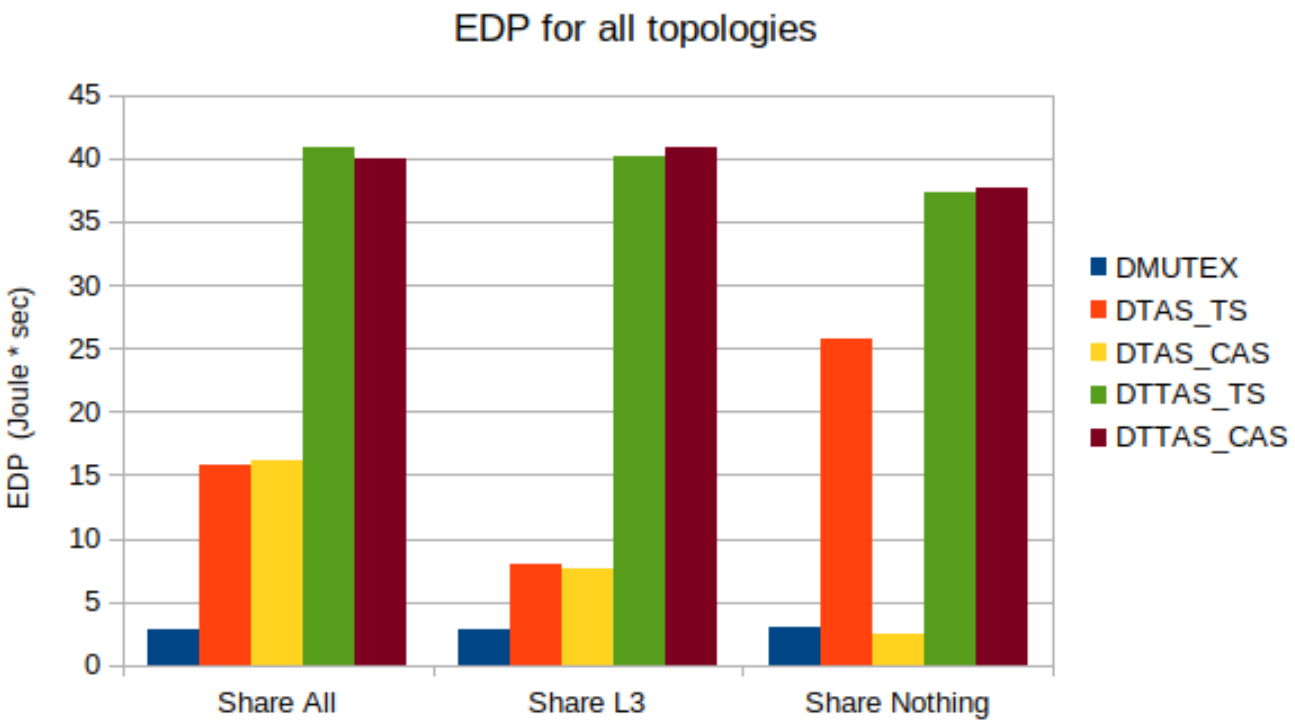
Για τις παραπάνω τοπολογίες, δώστε σε ένα διάγραμμα το συνολικό χρόνο εκτέλεσης της περιοχής ενδιαφέροντος για όλες τις υλοποιήσεις. Χρησιμοποιώντας το McPAT συμπεριλάβετε στην αξιολόγηση σας και την κατανάλωση ενέργειας (Energy, EDP κτλ.). Τι συμπεράσματα βγάζετε για την απόδοση των μηχανισμών συγχρονισμού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

Απάντηση


Cycles for all topologies
250000000
200000000
150000000
100000000
50000000
0
Cycles
Share All
Share L3
Share Nothing
DMUTEX
DTAS_TS
DTAS_CAS
DTTAS_TS
DTTAS_CAS


Time for all topologies
100000000
90000000
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
Time (ns)
Share All
Share L3
Share Nothing
DMUTEX
DTAS_TS
DTAS_CAS
DTTAS_TS
DTTAS_CAS


Energy consumption
Energy (J)
3000
2500
2000
1500
1000
500
0
Share All
Share L3
Share Nothing
DMUTEX
DTAS_TS
DTAS_CAS
DTTAS_TS
DTTAS_CAS


EDP for all topologies
EDP (Joule * sec)
45
40
35
30
25
20
15
10
5
0
Share All
Share L3
Share Nothing
DMUTEX
DTAS_TS
DTAS_CAS
DTTAS_TS
DTTAS_CAS

Σχολιασμός αποτελεσμάτων

Το γρηγορότερο topology είναι το share all,καθώς απαιτεί λιγότερους κύκλους εκτέλεσης,άρα και χρόνο εκτέλεσης (όπως άλλωστε φαίνεται και από τα αντίστοιχα διαγράμματα).Το αποτέλεσμα συμβαδίζει με αυτό που περιμέναμε,καθώς στις άλλες περιπτώσεις απαιτείται η μεταφορά του block όπου περιέχει το lock μεταξύ των caches των επεξεργαστών.

Στην share nothing τοπολογία οι επιμέρους caches θα πρέπει να επικοινωνήσουν μέσω της κύριας μνήμης για λάβουν οποιοδήποτε μπλοκ, πράγμα που σημαίνει πως θα υπάρχουν αυξημένες καθυστερήσεις καθώς η επικοινωνία μεταξύ των caches είναι ταχύτερη.

Σχετικά με την ενεργειακή κατανάλωση παρατηρούμε πως τα TTS κλειδώματα έχουν τη μεγαλύτερη κατανάλωση στην τοπολογία share all ενώ τα TS έχουν τη μεγαλύτερη κατανάλωση στην τοπολογία Share Nothing.Τη μικρότερη κατανάλωση απαιτεί το mutex κλείδωμα.

Το EDP γράφημα δεν μπορούμε να το εμπιστευτούμε 100% καθώς ο χρόνος εκτέλεσης ήταν πολύ μικρός εξαιτίας του error που περιγράφεται στην εισαγωγή της εργασίας.Συνεπώς δε βλέπουμε να επηρεάζει σημαντικά τα αποτελέσματα,παρά μόνο σε μία προσομοίωση που έχει τρέξει για τόσο λίγο χρόνο όπου το EDP παίρνει πολύ χαμηλές τιμές(share all topology,lock = DTAS_CAS).Σε όλες τις άλλες περιπτώσεις διατηρείται αντίστοιχο μοτίβο στο διάγραμμα με το energy γράφημα.

ΜΕΡΟΣ Β

OP	IS	EX	WR	CMT	COMMENTS
LD F0, 0(R1)	1	2-5	6	7	Miss A[0], fetch A[0],A[1] → block 0
ADDD F4,F4, F0	1	7-9	10	11	RAW(F0)
LD F1, 0(R2)	2	3-6	7	11	Miss B[0], fetch B[0],B[1] → block 1, LRU=0
MULD F4,F4, F1	2	11-15	16	17	RAW(F1,F4)
ANDI R9,R8, 0x2	3	4-5	8	17	CDB conflict
BNEZ R9,NEXT	3	9-10	11	18	RAW(R9), PRED=T, RES=NT
LD F5, 8(R1)	7	8	9	-	Load Queue full, cache hit, LRU=1, FLUSH@11
ADDD F4,F4, F5	7	-	-	-	RAW(R4,R5), FP RS full, FLUSH@11
ADDI R1,R1, 0x8	8	9-10	-	-	FLUSH@11
SUBI R8, R8,0x1	9	10-11	-	-	Integer RS full, FLUSH@11
LD F2,16(R2)	12	13-16	17	18	Miss B[2], fetch B[2],B[3] →block 1,LRU→0
MULD F2,F2,F5	12	18-22	23	24	FP FU busy
ADDD F4,F4,F2	13	24-26	27	28	RAW(F2)
LD F5, 8(R1)	13	14	15	28	Cache hit, LRU→1
ADDD F4,F4, F5	14	28-30	31	32	RAW(F4,F5), FP RS full
ADDI R1,R1, 0x8	14	15-16	18	32	CDB conflict, R1→A[1]
SUBI R8, R8,0x1	18	19-20	21	33	ROB is full
BNEZ R8,LOOP	18	22-23	24	33	RAW(R8), PRED = T, RES = NT
LD F0, 0(R1)	19	20	22	-	CDB conf, hit A[1], LRU→1 , FLUSH@24
ADDD F4,F4, F0	19	-	-	-	RAW(F0,F4), FP FU full, FLUSH@24
SD F4, 8(R2)	25	32-35	36	37	RAW(F4), miss B[1], fetch B[0],B[1] →block 1

Ύστερα από την εκτέλεση του κώδικα τα περιεχόμενα της cache είναι τα ακόλουθα:

Cache

Block	Περιεχόμενα	
0	A[0]	$A[1]$
1	B[0]	$B[1]$