4η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2015-2016, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τελική Ημερομηνία Παράδοσης: **10/07/2016**

1. Εισαγωγή

Στόχο της άσκησης αυτής αποτελεί η εξοικείωση με τους μηχανισμούς συγχρονισμού και τα πρωτόκολλα συνάφειας κρυφής μνήμης (cache coherence protocols) σε σύγχρονες πολυπύρηνες αρχιτεκτονικές. Για το σκοπό αυτό σας δίνεται ένας πολυνηματικός κώδικας με τον οποίο θα υλοποιήσετε διάφορους μηχανισμούς συγχρονισμού, τους οποίους στη συνέχεια θα αξιολογήσετε σε ένα πολυπύρηνο σύστημα με τη βοήθεια του προσομοιωτή Sniper.

2. Υλοποίηση μηχανισμών συγχρονισμού

Καλείστε να υλοποιήσετε τους μηχανισμούς κλειδώματος *Test-and-Set (TAS)* και *Test-and-Test-and-Set (TTAS)*, όπως τους διδαχτήκατε στις αντίστοιχες διαφάνειες του μαθήματος.

Συγκεκριμένα, στο αρχείο ***locks_scalability.c*** δίνεται έτοιμος κώδικας C ο οποίος χρησιμοποιεί τη βιβλιοθήκη Pthreads (Posix Threads) για τη δημιουργία και διαχείριση των νημάτων λογισμικού. Ο κώδικας δημιουργεί έναν αριθμό από νήματα, καθένα εκ των οποίων εκτελεί μια κρίσιμη περιοχή για ένα συγκεκριμένο αριθμό επαναλήψεων. Για όλα τα νήματα, η είσοδος στην κρίσιμη περιοχή ελέγχεται από μία κοινή μεταβλητή κλειδιού. Πριν την είσοδο στην κρίσιμη περιοχή, το κάθε νήμα εκτελεί την κατάλληλη ρουτίνα για την *απόκτηση του κλειδιού* (*lock aqcuire*), ώστε να εισέλθει σε αυτήν κατ' αποκλειστικότητα. Μέσα στην κρίσιμη περιοχή, το κάθε νήμα εκτελεί έναν dummy cpu-intensive υπολογισμό. Κατά την έξοδο από αυτήν εκτελεί την κατάλληλη ρουτίνα για την *απελευθέρωση του κλειδιού* (*lock release*). Δίνοντας τα κατάλληλα flags κατά τη μεταγλώττιση, μπορείτε να παράξετε κώδικα που χρησιμοποιεί κλήσεις σε κλειδώματα TAS, TTAS ή Pthread mutex (MUTEX). Ο τελευταίος από τους μηχανισμούς είναι ήδη υλοποιημένος στην βιβλιοθήκη Pthreads. Εσείς καλείστε να υλοποιήσετε τις ρουτίνες απόκτησης και απελευθέρωσης κλειδιού για τους μηχανισμούς TAS και TTAS.

2.1 Υλοποίηση κλειδωμάτων TAS και TTAS

Πιο συγκεκριμένα, οι ρουτίνες που θα πρέπει να υλοποιήσετε είναι οι ***spin_lock_tas_cas***, ***spin_lock_tas_ts***, ***spin_lock_ttas_cas*** και ***spin_lock_ttas_cas***. Οι ορισμοί τους δίνονται ημιτελείς στο αρχείο ***lock.h*** και εσείς πρέπει να υλοποιήσετε το κυρίως σώμα τους. Στο ίδιο αρχείο δίνεται ο ορισμός του τύπου για τη μεταβλητή κλειδιού (spinlock_t), ο ορισμός των σταθερών που σηματοδοτούν αν η κρίσιμη περιοχή είναι κλειδωμένη ή ελεύθερη (LOCKED, UNLOCKED), καθώς

και ο ορισμός της συνάρτησης αρχικοποίησης της μεταβλητής κλειδιού η οποία καλείται πριν τη δημιουργία των νημάτων.

Η υλοποίηση των ζητούμενων ρουτίνων θα πραγματοποιηθεί χρησιμοποιώντας τα atomic intrinsics του gcc και συγκεκριμένα τις εξής 2 συναρτήσεις:

1. *int __sync_lock_test_and_set(int *ptr, int newval);*
2. *int __sync_val_compare_and_swap(int *ptr, int oldval, int newval);*

Η πρώτη συνάρτηση υλοποιεί μία ατομική λειτουργία ανταλλαγής (exchange), δηλαδή:

ατομικά {
*γράψε τη newval στον *ptr και επέστρεψε την παλιά τιμή του *ptr*
}

Η δεύτερη συνάρτηση υλοποιεί μία ατομική λειτουργία compare-and-swap, δηλαδή:

ατομικά {
*αν η τρέχουσα τιμή του *ptr είναι oldval, τότε γράψε τη newval στον *ptr.*
*σε κάθε περίπτωση επέστρεψε την παλιά τιμή του *ptr*
}

2.2 Περιγραφή του προγράμματος

Το πρόγραμμα locks_scalability.c δέχεται τα εξής ορίσματα γραμμής εντολών:

- ***nthreads***: ο αριθμός των threads που θα δημιουργηθούν
- ***iterations***: ο αριθμός των επαναλήψεων που θα εκτελεστεί η κρίσιμη περιοχή από κάθε νήμα
- ***grain_size***: καθορίζει τον όγκο των dummy, cpu-intensive υπολογισμών, και κατ' επέκταση το μέγεθος της κρίσιμης περιοχής

Στον κώδικα του προγράμματος υπάρχουν κατάλληλα compile-time directives τα οποία καθορίζουν τη συμπερίληψη ή όχι κατά το χρόνο μεταγλώττισης ενός τμήματος του κώδικα. Με αυτόν τον τρόπο μπορούμε να ενσωματώσουμε σε ένα μόνο αρχείο διαφορετικές εκδόσεις του προγράμματος. Τα directives που χρησιμοποιούνται, και η σημασία τους, είναι τα εξής:

- SNIPER | REAL: ενεργοποιούν τα κατάλληλα τμήματα κώδικα για εκτέλεση του προγράμματος στον Sniper ή σε πραγματικό σύστημα, αντίστοιχα
- TAS_CAS | TAS_TS | TTAS_CAS | TTAS_TS | MUTEX: ενεργοποιούν τις κλήσεις στους μηχανισμούς συγχρονισμού TAS, TTAS και Pthread Mutex, αντίστοιχα
- DEBUG: ενεργοποιεί την εκτύπωση debugging μηνυμάτων

Σε αρχικό στάδιο, στη συνάρτηση main γίνονται οι απαραίτητες αρχικοποιήσεις, όπως η δέσμευση δομών και η αρχικοποίηση της μεταβλητής κλειδιού. Τα δεδομένα εισόδου που προορίζονται για κάθε νήμα ξεχωριστά αποθηκεύονται στη δομή ***targs_t***. Συγκεκριμένα, η δομή περιλαμβάνει το

πεδίο my_id που εκφράζει την ταυτότητα ενός νήματος, και αρχικοποιείται σε μια τιμή ξεχωριστή για κάθε νήμα (0 έως nthreads-1).

Σε δεύτερη φάση, ακολουθεί η δημιουργία των νημάτων με χρήση της ***pthread_create***. Κάθε νήμα που δημιουργείται εκτελεί τη συνάρτηση ***thread_fn***. Μέσα σε αυτήν, το νήμα αρχικά θέτει το *cpu affinity* του, ορίζει δηλαδή σε ένα bitmask (mask) το σύνολο των cpus του συστήματος ή του simulator όπου μπορεί να εκτελεστεί. Για τους σκοπούς της άσκησης θέλουμε μια "1-1" απεικόνιση ανάμεσα στα νήματα του προγράμματος και τις διαθέσιμες cpus, επομένως κάθε νήμα θέτει το affinity του σε μία και μόνο cpu, αυτή που έχει το ίδιο id με το my_id του. Δηλαδή, το νήμα 0 θα εκτελεστεί στη cpu 0, το νήμα 1 στη cpu 1, κ.ο.κ..

Στη συνέχεια, τα νήματα που έχουν δημιουργηθεί συγχρονίζονται (***pthread_barrier_wait***) ώστε ένα από αυτά (το πρώτο) να ενεργοποιήσει την λεπτομερή προσομοίωση και την καταγραφή των στατιστικών στην περιοχή που μας ενδιαφέρει (SimRoiStart()), αν η εκτέλεση γίνεται στον Sniper, ή να αρχίσει τη μέτρηση του χρόνου εκτέλεσης, αν η εκτέλεση γίνεται σε πραγματικό σύστημα (gettimeofday). Ακολουθεί ο βρόχος εντός του οποίου εκτελείται η κρίσιμη περιοχή, προστατευόμενη από την εκάστοτε υλοποίηση κλειδώματος. Ομοίως, αφού ολοκληρωθεί η εκτέλεση τα νήματα συγχρονίζονται και πάλι ώστε ένα από αυτά να απενεργοποιήσει τη λεπτομερή προσομοίωση ή την μέτρηση του χρόνου.

2.3 Μεταγλώττιση προγράμματος για προσομοίωση στον Sniper

Στο tarball που σας δίνεται υπάρχουν τα αρχεία locks_scalability.c, lock.h καθώς και το Makefile. Το τελευταίο μπορεί να παράξει είτε κώδικα για προσομοίωση στον Sniper ή για εκτέλεση σε πραγματικό μηχάνημα αναλόγως με την τιμή του IMPLFLAG. Μπορείτε να θέσετε το IMPLFLAG είτε ίσο με –DSNIPER είτε –DREAL. Για την παραγωγή κώδικα που κάνει χρήση των μηχανισμών TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS ή Pthread Mutex, αρκεί να θέσετε στο Makefile το LFLAG σε –DTAS_CAS, -DTAS_TS, -DTTAS_CAS, -DTTAS_TS ή -DMUTEX, αντίστοιχα. Τέλος, αφού έχετε θέσει το SNIPER_BASE_DIR ώστε να δείχνει στο path στο οποίο έχετε μεταγλωττίσει τον sniper, δίνοντας την εντολή make στο τερματικό θα παραχθεί το εκτελέσιμο με το όνομα locks.

3. Αξιολόγηση επίδοσης

Σε αυτό το μέρος καλείστε να αξιολογήσετε τις επιδόσεις των υλοποιήσεών σας, τόσο ως προς την κλιμακωσιμότητά τους, όσο και ως προς την τοπολογία των νημάτων.

3.1 Σύγκριση υλοποιήσεων

Με τον όρο *κλιμακωσιμότητα* (scalability) εννοούμε πόσο καλά αποδίδει ένα παράλληλο πρόγραμμα καθώς αυξάνεται ο αριθμός των νημάτων από τα οποία αποτελείται. Ακόμα και αν ένα παράλληλο πρόγραμμα είναι σχεδιασμένο για να κλιμακώνει ιδανικά (π.χ. ο χρόνος εκτέλεσης με N νήματα να ισούται με 1/N του χρόνου με 1 νήμα), υπάρχουν διάφοροι εξωγενείς παράγοντες που μπορούν να

περιορίσουν την κλιμακωσιμότητά του πολύ κάτω του ιδανικού. Τέτοιοι είναι το overhead που σχετίζεται με το πρωτόκολλο συνάφειας, το περιορισμένο bandwidth πρόσβασης στην κύρια μνήμη, κ.ο.κ.. Σε τέτοιες περιπτώσεις, το κόστος μιας λειτουργίας ενός νήματος (π.χ. προσπέλαση μνήμης) είναι πολύ μεγαλύτερο συνήθως όταν στην εκτέλεση συμμετέχουν πολλά νήματα, παρά όταν συμμετέχουν λίγα.

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 και 32. Για το λόγο αυτό θα χρησιμοποιήσετε προσομοιωμένα πολυπύρηνα συστήματα αντίστοιχου πλήθους πυρήνων, με κοινό χαρακτηριστικό την εξής πολιτική διαμοιρασμού των caches:

L1 cache	ιδιωτική για κάθε πυρήνα
L2 cache	μοιραζόμενη ανά 4 πυρήνες σε κάθε socket (chip)
L3 cache	μοιραζόμενη από όλους τους πυρήνες σε κάθε socket (chip)

Κάθε σύστημα χρησιμοποιεί το MSI πρωτόκολλο συνάφειας κρυφής μνήμης. Η τοπολογία καθενός φαίνεται στις εικόνες που ακολουθούν. Στα συστήματα αυτά, καλείστε να εκτελέσετε προσομοιώσεις του προγράμματος για όλους τους ακόλουθους συνδυασμούς:

- εκδόσεις προγράμματος: TAS_CAS, TAS_TS, TTAS_CAS, TTAS_TS, MUTEX
- iterations: 1000
- nthreads: 1, 2, 4, 8, 16, 32 (σε σύστημα με ισάριθμους πυρήνες)
- grain_size: 1, 10, 100

Για να εκτελέσετε μια προσομοίωση, τρέχετε τον Sniper με όρισμα το επιθυμητό εκτελέσιμο:

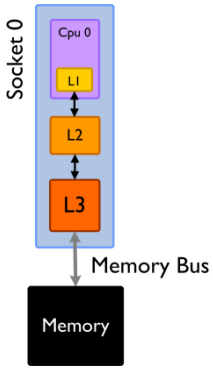
run-sniper -c <config_script> -n <ncores> --roi -- /path/to/binary/locks <nthreads> 1000 <grain_size>

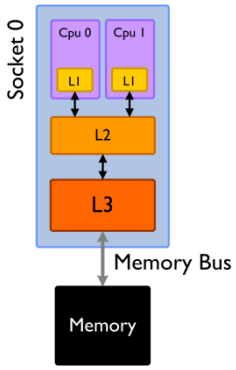
Το configuration script που θα χρησιμοποιήσετε είναι το ask3.cfg που δίνεται στο tarball.

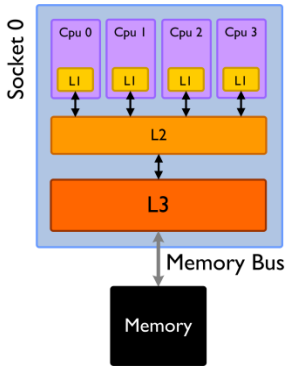
ΠΡΟΣΟΧΗ: ο αριθμός των νημάτων που δίνετε σαν 1^{ο} όρισμα στο εκτελέσιμο θα πρέπει να ταυτίζεται με τον αριθμό που δίνετε στο όρισμα -n του προσομοιωτή, ώστε το σύστημα που δημιουργεί ο Sniper να έχει ισάριθμους πυρήνες.

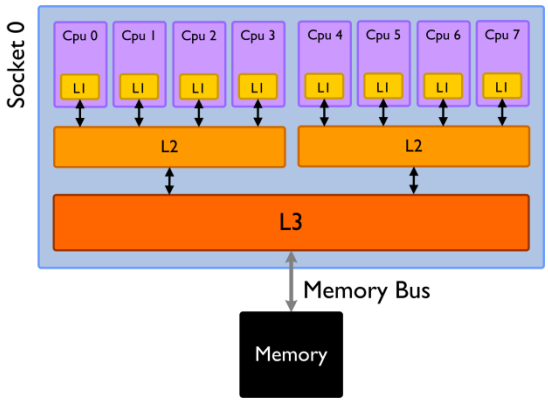
3.1.1. Για κάθε grain size, δώστε το διάγραμμα της κλιμάκωσης του συνολικού χρόνου εκτέλεσης της περιοχής ενδιαφέροντος σε σχέση με τον αριθμό των νημάτων. Συγκεκριμένα, στον x-άξονα θα πρέπει να έχετε τον αριθμό των νημάτων και στον y-άξονα τον χρόνο εκτέλεσης σε κύκλους. Στο ίδιο διάγραμμα θα πρέπει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

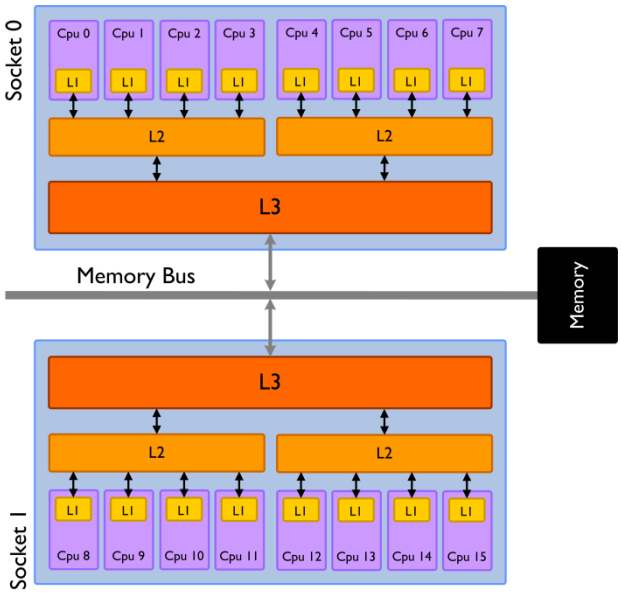
3.1.2. Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με τη φύση της εκάστοτε υλοποίησης; Τι συμπεραίνετε για την κλιμάκωση του χρόνου εκτέλεσης σε σχέση με το grain size; Δικαιολογήστε τις απαντήσεις σας.

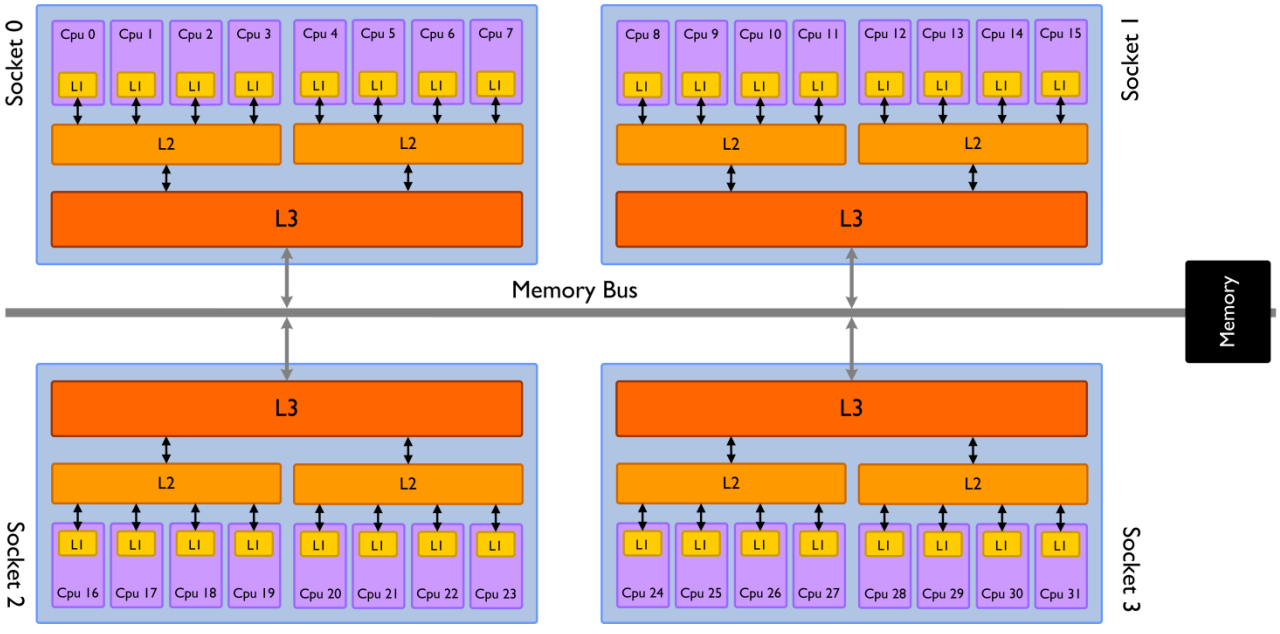

Socket 0
Cpu 0
L1
L2
L3
Memory Bus
Memory


Socket 0
Cpu 0
Cpu 1
L1
L2
L3
Memory Bus
Memory


Socket 0
Cpu 0
Cpu 1
Cpu 2
Cpu 3
L1
L2
L3
Memory Bus
Memory


Socket 0
Cpu 0
Cpu 1
Cpu 2
Cpu 3
Cpu 4
Cpu 5
Cpu 6
Cpu 7
L1
L2
L3
Memory Bus
Memory


Socket 0
Cpu 0
Cpu 1
Cpu 2
Cpu 3
Cpu 4
Cpu 5
Cpu 6
Cpu 7
L1
L2
L3
Memory Bus
Memory
Socket 1
Cpu 8
Cpu 9
Cpu 10
Cpu 11
Cpu 12
Cpu 13
Cpu 14
Cpu 15


Socket 0
Cpu 0
Cpu 1
Cpu 2
Cpu 3
Cpu 4
Cpu 5
Cpu 6
Cpu 7
Socket 1
Cpu 8
Cpu 9
Cpu 10
Cpu 11
Cpu 12
Cpu 13
Cpu 14
Cpu 15
L1
L2
L3
Memory Bus
Memory
Socket 2
Cpu 16
Cpu 17
Cpu 18
Cpu 19
Cpu 20
Cpu 21
Cpu 22
Cpu 23
Socket 3
Cpu 24
Cpu 25
Cpu 26
Cpu 27
Cpu 28
Cpu 29
Cpu 30
Cpu 31

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

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 αν έχετε στον ~okeanos. Χρησιμοποιήστε τους ίδιους αριθμούς νημάτων 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.

3.2 Τοπολογία νημάτων

Στόχος του ερωτήματος αυτού είναι η αξιολόγηση της κλιμάκωσης των διαφόρων υλοποιήσεων σε συστήματα 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

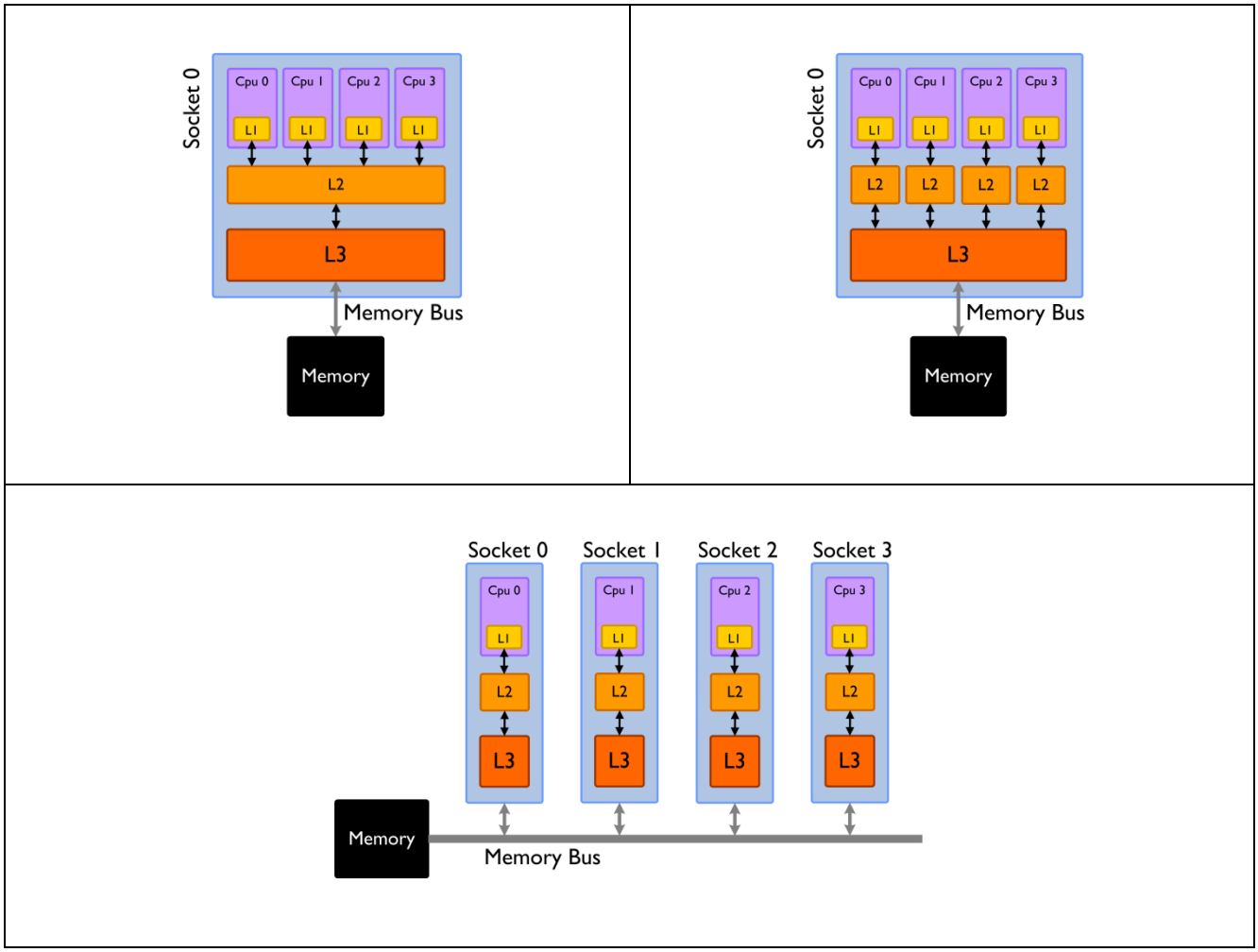
και εξετάζουμε τις ακόλουθες τοπολογίες:

- ***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

Οι τοπολογίες αυτές φαίνονται στα παρακάτω σχήματα. Για την εκτέλεση των προσομοιώσεων χρησιμοποιήστε και αυτή τη φορά το configuration script ask3.cfg, όπως δείξαμε στην ενότητα 3.1, επανακαθορίζοντας όμως συγκεκριμένες παραμέτρους του script που ορίζουν τον τρόπο διαμοιρασμού συγκεκριμένων επιπέδων της ιεραρχίας μνήμης. Συγκεκριμένα, για να μην αλλάζετε το configuration script, μπορείτε να εκτελέσετε τον sniper όπως δείχνουμε παρακάτω προκειμένου να περάσουν οι σωστές τιμές στις αντίστοιχες παραμέτρους:

- Για την τοπολογία share-all:

```
run-sniper -c ask3.cfg -n 4  --roi  -g --perf_model/l1_icache/shared_cores=1 \
    -g --perf_model/l1_dcache/shared_cores=1 -g --perf_model/l2_cache/shared_cores=4 \
    -g --perf_model/l3_cache/shared_cores=4 -g --perf_model/dram/controllers_interleaving=4  \
    -- /path/to/binary/locks 4 1000 1
```



- Για την τοπολογία share-L3:

  run-sniper -c ask3.cfg -n 4 –roi ***-g --perf_model/l1_icache/shared_cores=1*** ****

  -g --perf_model/l1_dcache/shared_cores=1 -g --perf_model/l2_cache/shared_cores=1 ****

  -g --perf_model/l3_cache/shared_cores=4 -g --perf_model/dram/controllers_interleaving=4 ****

  -- /path/to/binary/locks 4 1000 1

- Για την τοπολογία share-nothing:

  run-sniper -c ask3.cfg -n 4 --roi \

  -g --perf_model/l1_icache/shared_cores=1 -g --perf_model/l1_dcache/shared_cores=1 ****

  -g --perf_model/l2_cache/shared_cores=1 -g --perf_model/l3_cache/shared_cores=1 ****

  -g --perf_model/dram/controllers_interleaving=1 ****

  -- /path/to/binary/locks 4 1000 1

3.2.1. 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.

*Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (**pdf**, **docx** ή **odt**). Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας **(Όνομα, Επώνυμο, ΑΜ)**.*

Η άσκηση θα παραδοθεί ηλεκτρονικά στην ιστοσελίδα:

http://www.cslab.ece.ntua.gr/courses/advcomparch/submit

Δουλέψτε ατομικά. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην προσπαθήσετε να την αντιγράψετε από άλλους συμφοιτητές σας.

Μην αφήσετε την εργασία για το τελευταίο Σαββατοκύριακο, απαιτεί αρκετό χρόνο για την εκτέλεση όλων των προσομοιώσεων, ξεκινήστε αμέσως!