Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 9: Συμφωνία μνημών CACHE.

Μοντέλα Συνέπειας Μνήμης.

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

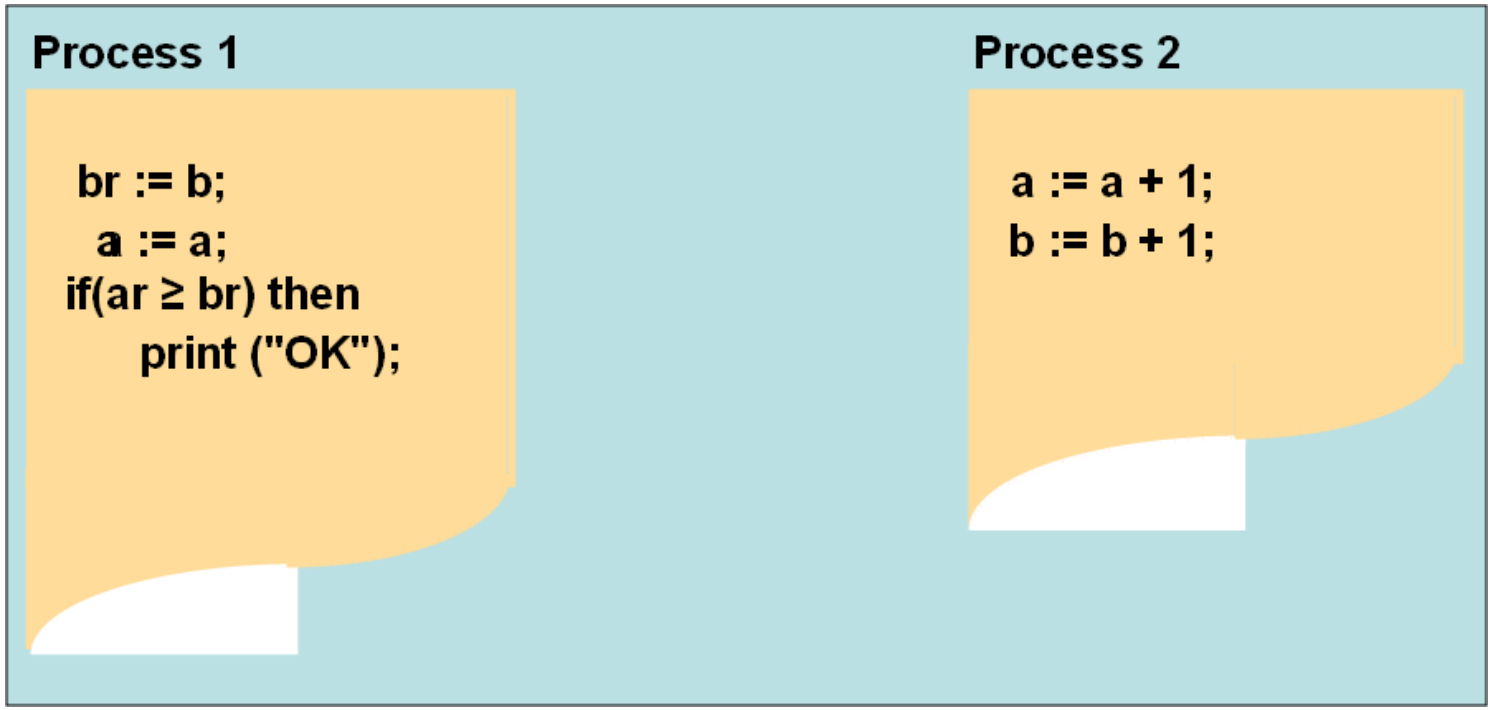
Σκοπός της Ενότητας

- Το πρόβλημα της συμφωνίας των κρυφών μνημών.
- Η παρουσίαση των βασικών μοντέλων συνέπειας μνήμης.

Ποιο είναι το πρόβλημα της συμφωνίας μνημών cache;

- Σε SMP συστήματα υπάρχουν μνήμες ιδιωτικές (μόνο για ένα επεξεργαστή), και κοινές (για ομάδες επεξεργαστών ή για όλους).
- Η οργάνωση είναι ουσιώδης για να πετύχουμε καλή λειτουργία.
- Δημιουργείται ένα πρόβλημα.
- Μπορεί να υπάρχουν πολλά αντίγραφα των **ίδιων δεδομένων** σε διαφορετικές μνήμες cache, και αν επιτραπεί στους επεξεργαστές να ενημερώνουν ελεύθερα τα αντίγραφά τους, τότε θα υπάρχει **αντιφατική άποψη** της μνήμης.

Παράδειγμα συνέπειας μνήμης



- Θα εκτυπωθεί το μήνυμα OK ή όχι (η a και η b είναι κοινές μεταβλητές);

Το πρόβλημα της συνάφειας της κοινής μνήμης

Χρόνος	Γεγονός	Τα περιεχόμενα της cache για τη CPU A	Τα περιεχόμενα της cache για τη CPU B	Τα περιεχόμενα της μνήμης για τη θέση X
0				1
1	Η CPU A διαβάζει τη θέση X	1		1
2	Η CPU B διαβάζει τη θέση X	1	1	1
3	Η CPU A αποθηκεύει το 0 στη θέση X	0	1	0

- **Το πρόβλημα της συνάφειας της cache για μία μονή θέση μνήμης (X) η οποία διαβάζεται και εγγράφεται από δύο επεξεργαστές (A και B).**
- Αρχικά, υποθέτουμε ότι καμία από τις δύο cache δε περιλαμβάνει κάποια τιμή και ότι στη θέση X βρίσκεται η τιμή 1. Επίσης, υποθέτουμε ότι έχουμε write-through cache (μία write-back cache θα πρόσθεται μερικά επιπλέον αλλά παρόμοια μπερδέματα). Μετά την εγγραφή της τιμής της θέσης X από την A, η cache της A αλλά και η μνήμη περιέχουν τη νέα τιμή ενώ η cache της B όχι, έχοντας ως αποτέλεσμα εάν η B διαβάσει τη τιμή της X να πάρει κι αυτή τη τιμή 1!
- **Η cache της A έχει άλλη τιμή από την cache της B για την τιμή του X.**

Η συνάφεια (ή συνέπεια ή συνεκτικότητα) της μνήμης

- Το πρόβλημα της συνάφειας (ή συνέπειας) της μνήμης **είναι ένα σημαντικό πρόβλημα** στα παράλληλα συστήματα.
- Είναι κοινό και στους πολυεπεξεργαστές (SMP) και στους πολυ-υπολογιστές.
- Υπάρχει εξαιτίας του γεγονότος ότι υπάρχουν ιδιωτικές μνήμες που μπορεί να έχουν διαφορετική εικόνα για τα ίδια τα δεδομένα.

Που εμφανίζεται το πρόβλημα της συνάφειας της μνήμης

- **Πολυεπεξεργαστές:**
  - Οι CPU έχουν ιδιωτικές caches, ενώ υπάρχει μια κοινή εσωτερική μνήμη με ένα μόνο χώρο διευθύνσεων, ο οποίος είναι ορατός από όλες τις CPU (Κατηγορία UMA).
- **Πολυ-υπολογιστές:**
  - Οι κόμβοι έχουν ιδιωτικές μνήμες και μπορεί να έχουν και κοινή μοιραζόμενη μνήμη η οποία είναι ορατή από όλες τις CPU, με κάποια καθυστέρηση (κατηγορία NUMA).

Τι προβλήματα δημιουργεί η ύπαρξη μεριζόμενης μνήμης;

- Συχνά, πολλές CPU επιχειρούν να διαβάσουν την ίδια θέση μνήμης.
- Συχνά, πολλές CPU επιχειρούν να γράψουν στην ίδια θέση μνήμης.
- Μερικά μηνύματα αιτήσεων παραλαμβάνονται με διαφορετική σειρά από εκείνη που υποβλήθηκαν.
- Κάποιες ενότητες μνήμης έχουν αντίγραφα σε ιδιωτικές μνήμες (π.χ. Cache).
- Απαιτούνται λοιπόν αυστηρά μέτρα/πρωτόκολλα για να μη δημιουργηθεί χάος.

Ποιες είναι οι 2 πιο κοινές τακτικές εγγραφής cache;

- **Εγγραφή προς τα πίσω** (write-back):
  - Οι πράξεις (ανανέωση/εγγραφή) γίνονται μόνο στη μνήμη cache. Η κύρια μνήμη ενημερώνεται μόνο όταν η αντίστοιχη γραμμή της cache πρόκειται να εκδιωχθεί.
- **Εγγραφή από μέσα** (write-through):
  - Όλες οι πράξεις εγγραφής γίνονται στην κύρια μνήμη καθώς και στη μνήμη cache. Πάντα η κύρια μνήμη έχει έγκυρες τιμές.

Δημιουργία ασυμφωνίας

- Η **write-back** εγγραφή δημιουργεί πρόβλημα, γιατί αν δυο διαφορετικές ιδιωτικές cache έχουν την ίδια γραμμή και ένας από τους επεξεργαστές ενημερώσει η γραμμή, τότε η άλλη cache θα έχει τιμή που δεν ισχύει.

- Η **write-through** εγγραφή ομοίως προκαλεί πρόβλημα, εκτός αν οι άλλες cache εποπτεύουν την κίνηση της μνήμης ή δέχονται απευθείας ειδοποίηση της ανανέωσης.

2 πιθανοί τρόποι για ενημέρωση

- **Write-update:** Κάθε ενημέρωση γίνεται multicast προς όλα τα προγράμματα. Οι αναγνώσεις γίνονται στα τοπικά αντίγραφα των δεδομένων.

- **Write-invalidate:** Ένα μήνυμα γίνεται multicast σε κάθε πρόγραμμα ακυρώνοντας τα αντίγραφα των δεδομένων τους πριν ενημερωθούν τα δεδομένα. Παρόλα αυτά, άλλα προγράμματα μπορούν να κάνουν αίτηση για τα ενημερωμένα δεδομένα.

Βασικά πρωτόκολλα διατήρησης συνέπειας: write-update

- Η εγγραφή σε ένα block **γίνεται σε όλα τα αντίγραφά του**, και επιτυγχάνει μόνον αφού έχουν ανανεωθεί όλα.
- Η συνέπεια μπορεί να επιτευχθεί με τη χρήση του μηχανισμού καθολικής διάταξης (total order) των εγγραφών σε όλο το ΚΣ.
- Ένας τρόπος είναι η χρήση μίας διεργασίας “global sequencer”.
  - Κάθε προτιθέμενη αλλαγή σε ένα block στέλνεται πρώτα στον sequencer,
  - αυτός της αναθέτει τον επόμενο-μοναδικό αριθμός μιας μονοτονικά αύξουσας ακολουθίας και
  - στέλνει την αλλαγή σε όλους τους κόμβους που έχουν αντίγραφο του block.
  - Αν κάποιος κόμβος λάβει μήνυμα ανανέωσης εκτός αναμενόμενης σειράς, ζητάει από τον sequencer να ξαναστείλει το μήνυμα αλλαγής που λείπει προφανώς ο sequencer κρατάει log των πρόσφατων αλλαγών σε όλα τα blocks.
- Η μέθοδος αυτή απαιτεί επικοινωνία στο δίκτυο, για κάθε εγγραφή στην κοινή μνήμη, γι΄ αυτό και **δεν χρησιμοποιείται ευρέως.**

Βασικά πρωτόκολλα διατήρησης συνέπειας: write-invalidate

- Πριν από μία εγγραφή σε τοπικό αντίγραφο block, όλα οι κόμβοι που έχουν αντίγραφό του λαμβάνουν μήνυμα απαξίωσης του block.
- Μετά την απαξίωση, το μοναδικό έγκυρο αντίγραφο βρίσκεται στον τοπικό κόμβο.
- Αν κάποιος άλλος κόμβος χρειαστεί το block, πρέπει να φέρει αντίγραφό του από τον κόμβο-ιδιοκτήτη.
- Έτσι διασφαλίζεται ακολουθιακή συνέπεια.

Επόπτευση αρτηρίας και ακύρωση γραμμής

Δραστηριότητα Επεξεργαστή	Δραστηριότητα Διαύλου	Τα περιεχόμενα της cache για τη CPU A	Τα περιεχόμενα της cache για τη CPU B	Τα περιεχόμενα της μνήμης για τη θέση X
				0
Η CPU A διαβάζει τη θέση X	Cache miss στη θέση X	0		0
Η CPU B διαβάζει τη θέση X	Cache miss στη θέση X	0	0	0
Η CPU A γράφει ένα 1 στη θέση X	Ακύρωση για τη θέση X	1		0
Η CPU B διαβάζει τη θέση X	Cache miss στη θέση X	1	1	1

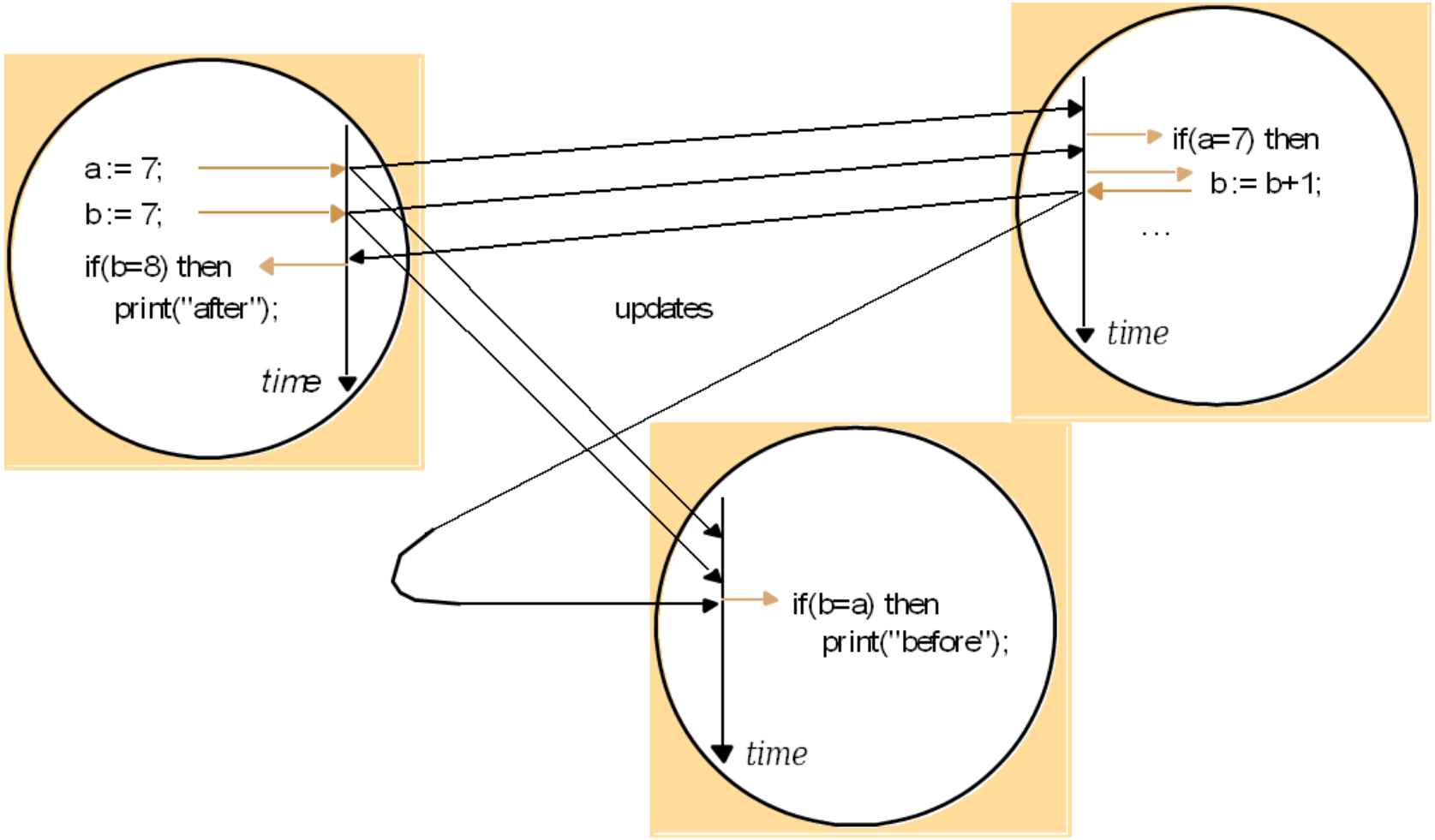
- **Παράδειγμα ενός πρωτόκολλου ακύρωσης που τρέχει σε έναν snooping δίαυλο για ένα block (X) μονής cache με write-back caches.**
- Αρχικά, υποθέτουμε ότι καμία από τις δύο cache δε περιλαμβάνει το X και ότι η τιμή του X στη μνήμη είναι 0. Τα περιεχόμενα της μνήμης και της CPU δείχνουν τη τιμή αφότου έχουν ολοκληρωθεί και η δραστηριότητα του επεξεργαστή και του διαύλου. Ένα κενό δείχνει ότι δεν υπήρξε δραστηριότητα ή ότι δεν αντιγράφτηκε η cache. Όταν γίνεται το δεύτερο miss από τη B, η CPU A ανταπαντάει με τη τιμή, ακυρώνοντας την ανταπόκριση της μνήμης. Επιπλέον, και τα περιεχόμενα της cache της B αλλά και της μνήμης ενημερώνονται. Αυτή η ενημέρωση της μνήμης, η οποία συμβαίνει όταν ένα block γίνεται διαμοιραζόμενο, είναι χαρακτηριστικό σε όλα τα πρωτόκολλα και τα απλοποιεί, όπως θα δούμε σύντομα.

Επόπτευση αρτηρίας και εκπομπή γραμμής

Δραστηριότητα Επεξεργαστή	Δραστηριότητα Διαύλου	Τα περιεχόμενα της cache για τη CPU A	Τα περιεχόμενα της cache για τη CPU B	Τα περιεχόμενα της μνήμης για τη θέση X
				0
Η CPU A διαβάζει τη θέση X	Cache miss στη θέση X	0		0
Η CPU B διαβάζει τη θέση X	Cache miss στη θέση X	0	0	0
Η CPU A γράφει ένα 1 στη θέση X	Εκμπομπή της εγγραφής της θέσης X	1	1	1
Η CPU B διαβάζει τη θέση X		1	1	1

- **Παράδειγμα ενός write update ή πρωτόκολλου εκπομπής που τρέχει σε έναν snooping δίαυλο για ένα block (X) μονής cache με write-back caches.**
- Αρχικά, υποθέτουμε ότι καμία από τις δύο cache δε περιλαμβάνει το X και ότι η τιμή του X στη μνήμη είναι 0. Τα περιεχόμενα της μνήμης και της CPU δείχνουν τη τιμή αφότου έχουν ολοκληρωθεί και η δραστηριότητα του επεξεργαστή και του διαύλου. Ένα κενό δείχνει ότι δεν υπήρξε δραστηριότητα ή ότι δεν αντιγράφτηκε η cache. Όταν η CPU Α εκπέμπει την εγγραφή, και τα περιεχόμενα της cache της Β αλλά και της μνήμης ενημερώνονται.

Η χρήση του write-update



Σύγκριση write-invalidate με write-update

- **Έχει επικρατήσει το write-invalidate:**
  - **Πολλές εγγραφές στην ίδια λέξη:** απαιτείται μόνο μια ακύρωση, ενώ στην άλλη περίπτωση απαιτούνται πολλαπλές ενημερώσεις.
  - **Κάθε μπλοκ κρυφής μνήμης αποτελείται από πολλές λέξεις**. Αν γράφονται πολλές λέξεις στο ίδιο μπλοκ, απαιτείται μόνο μια ακύρωση, ενώ στην άλλη περίπτωση απαιτούνται πολλαπλές ενημερώσεις.
  - **Υπάρχει καθυστέρηση στην ανάγνωση (για miss)** αν υπάρχει write-invalidate, γιατί η λέξη δε βρίσκεται στην κρυφή μνήμη, κάτι που δεν ισχύει στο write-update.

Μοντέλα Συνέπειας Μνήμης

- Ένα μοντέλο συνέπειας (consistency model) αναφέρεται στο βαθμό συνέπειας που πρέπει να διατηρούν τα δεδομένα της κοινόχρηστης μνήμης, προκειμένου η μνήμη να δουλεύει σωστά για ένα ορισμένο σύνολο εφαρμογών.
- Κάθε μοντέλο έχει ένα **σύνολο κανόνων** στους οποίους πρέπει να υπακούν οι εφαρμογές, εάν θέλουν η κοινή μνήμη να προσφέρει το βαθμό συνέπειας που εγγυάται το αντίστοιχο μοντέλο.
- Τα βασικά μοντέλα θα παρουσιασθούν στην συνέχεια.

- *Ο τομέας αυτός αποτελεί αντικείμενο συνεχιζόμενης έρευνας (ιδιαίτερα για παράλληλα ΛΣ).*
- **(τι ονομάζεται μοντέλο συνέπειας μνήμης;)**

Παράδειγμα μοντέλου συνέπειας

- Η CPU 0 γράφει τιμή 1 σε κάποια θέση μνήμης.
- Η CPU 1 γράφει τιμή 2 στην ίδια θέση μνήμης.
- Η CPU 2 διαβάζει τη θέση μνήμης και βρίσκει την τιμή 1.

- **Υπάρχει ασυνέπεια;**
- => Εξαρτάται από το μοντέλο συνέπειας, τι δηλαδή είχε υποσχεθεί η μνήμη να κάνει.

Συνέπεια Μνήμης (memory coherence) Ορισμός & Μοντέλα συνέπειας

- Η μνήμη είναι **συνεπής** αν η επιστρεφόμενη από ανάγνωση τιμή είναι ότι αναμενόταν από τον προγραμματιστή.

Μοντέλα συνέπειας:

Αυστηρό (strict consistency): γίνεται ανάγνωση της πιο πρόσφατης τιμής που είχε γίνει εγγραφή.(απαιτεί καθολική διάταξη των αιτημάτων ==> δαπανηρό).

Ακολουθιακό (sequential consistency): για παράδειγμα η σειριοποιησιμότητα.

Γενικό (general consistency): όλα τα αντίγραφα έχουν τελικά την ίδια τιμή.

Αδύνατο/Αποδεσμευτικό (weak/release consistency): (ala two phase locking) with.

Πρωτόκολλα συνέπειας μνήμης

- **Πρωτόκολλα για τη διατήρηση της συνέπειας της μνήμης:**
  - Write-invalidate.
  - Write-update.
- **Πλεονεκτήματα:**
- Το κύριο αντίγραφο ενημερώνεται πρώτο, μετά όλα τα αντίγραφα (all replicas) στο set.
- Ο εγγραφέας δεν μπλοκάρεται, εκτός και αν θέλει να κάνει ανάγνωση μιας θέσης που αναμένει ενημέρωση.
- Ευελιξία:
  - Το κλείδωμα χρησιμοποιείται σε ρύθμιση του αναγνώστη/εγγραφέα.
  - Συνδυάζει κλείδωμα με μεταφορά δεδομένων από και προς τον ιδιοκτήτη.

Write-Update Πρωτόκολλα συνέπειας

- **Πρωτόκολλο ιδιοκτησίας Berkeley (ala locking with modes):**
- Κάθε αντικείμενο έχει έναν ιδιοκτήτη.
- Ένα αντικείμενο σε έναν κόμβο μπορεί να είναι σε μία από αυτές τις **καταστάσεις:**
  - Άκυρο (invalid).
  - Χωρίς κάποιον ιδιοκτήτη αλλά με έγκυρα δεδομένα (unowned).
  - Με ιδιοκτήτη και έγκυρα δεδομένα-μονό αντίγραφο (owned exclusively).
  - Με ιδιοκτήτη και έγκυρα δεδομένα-πολλαπλά αντίγραφα (owned shared).
- Και ένα αντικείμενο σε έναν κόμβο μπορεί να κάνει τις εξής **λειτουργίες:**
  - Ανάγνωση πολλαπλών αντιγράφων (read-shared).
  - Ανάγνωση μονού αντίγραφου (read-exclusive).
  - Εγγραφή (write).

Προϋποθέσεις συνεκτικότητας

- Ένα σύστημα μνήμης είναι **συνεκτικό** όταν:
- **Η επικοινωνία γίνεται με κοινές θέσεις μνήμης.** Αν ένας επεξεργαστής γράψει στη θέση μνήμης X, και ένας άλλος **ύστερα από κάποιο χρονικό διάστημα** διαβάσει τη θέση X θα πάρει την ενημερωμένη τιμή.
- **Διατηρείται η σειρά των εντολών του προγράμματος**: αν ένας επεξεργαστής διαβάσει τη θέση X που είχε γράψει προηγουμένως και δεν έχει μεσολαβήσει άλλη εγγραφή, πρέπει να πάρει την τιμή που είχε γράψει.
- Διαφορετικές **εγγραφές** στην ίδια θέση μνήμης **εκτελούνται σειριακά** και αυτή η σειρά φαίνεται η ίδια σε όλους τους επεξεργαστές.

Μοντέλα Συνέπειας Μνήμης: Αυστηρό Μοντέλο (strict)

Μοντέλα Συνέπειας Μνήμης: Αυστηρό Μοντέλο (strict) (1/2)

- Μία κοινόχρηστη μνήμη υποστηρίζει το αυστηρό μοντέλο, εάν η τιμή που επιστρέφει η εντολή read από μία διεύθυνση, **είναι πάντα η ίδια με την τιμή που έγραψε στην ίδια διεύθυνση η τελευταία - χρονικά - εντολή write** ανεξάρτητα από τις τοποθεσίες των διεργασιών που εκτέλεσαν τις read, write.
- Δηλαδή, **όλες οι εγγραφές γίνονται αμέσως ορατές** σε όλες τις διεργασίες.
- Απαιτεί **απόλυτο καθολικό χρόνο**, ο οποίος δεν είναι εφικτός σε ΚΣ.
- Μπορεί να υλοποιηθεί αν δεν υπάρχει cache και υπάρχει μόνο ένα άρθρωμα μνήμης με μια πόρτα εισόδου/εξόδου (που θα προκαλούσε τεράστια συμφόρηση).
- Στην πράξη, η υλοποίηση του μοντέλου αυτού είναι **αδύνατη.**

Μοντέλα Συνέπειας Μνήμης: Αυστηρό Μοντέλο (strict) (2/2)

Παραδείγματα:

Αυστηρή συνέπεια

P1: W(x)1

P2: R(x)1

Μη αυστηρή συνέπεια

P1: W(x)1

P2: R(x)0 R(x)1

Η P2 μετά την εγγραφή της P1 της τιμής 1 διαβάζει το κελί και βλέπει την τιμή 0, δηλαδή δεν υπάρχει άμεση ενημέρωση.

Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential)

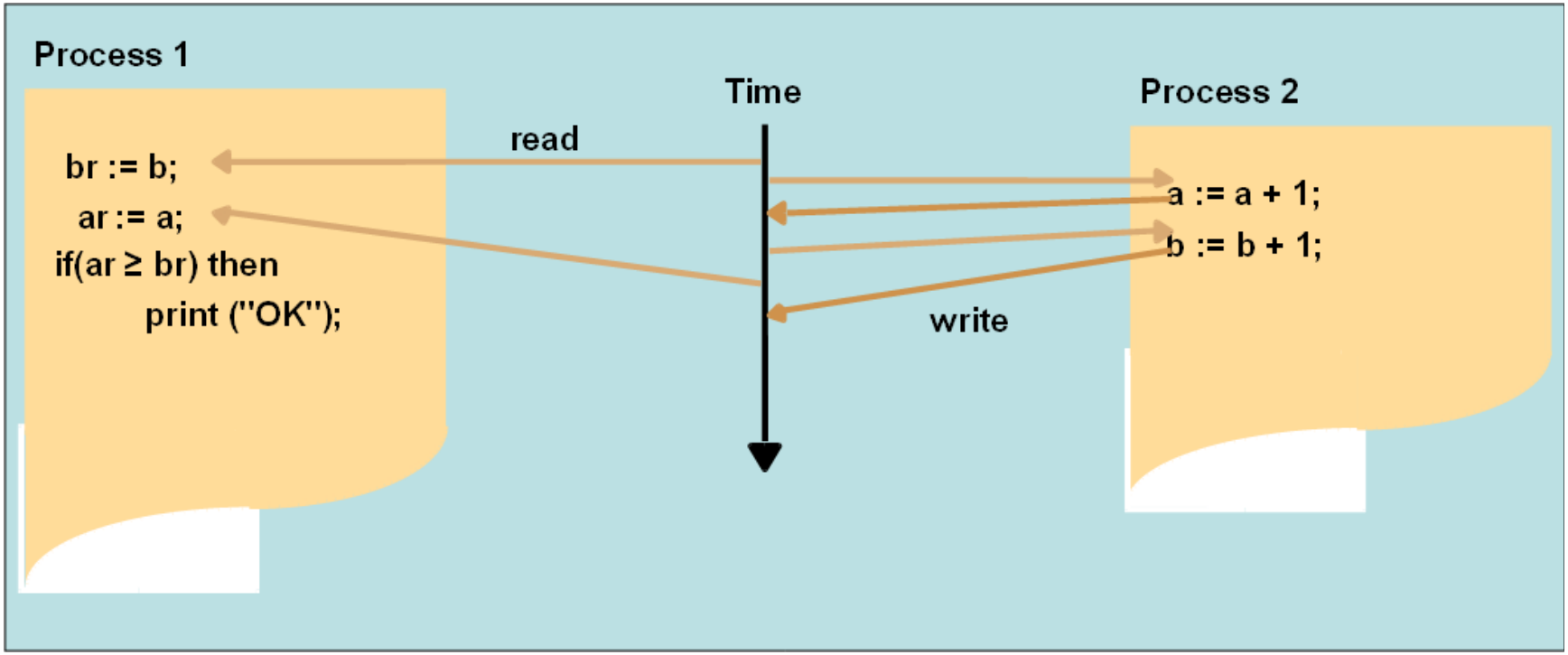
Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential) (1/5)

- Πιο “χαλαρό μοντέλο”, αλλά και το πιο διαδεδομένο, λόγω **ευκολότερης υλοποίησης.**
- Όλες οι διεργασίες βλέπουν την ίδια ακολουθία προσπελάσεων στην κοινή μνήμη, χωρίς να παίζει ρόλο ποια είναι αυτή.
- **Για παράδειγμα,** αν γίνουν 3 λειτουργίες: read (r1), write (w1), read (r2) στην ίδια θέση μνήμης με αυτή τη σειρά, οποιαδήποτε ακολουθία είναι αποδεκτή, αρκεί να ισχύει για όλες τις διεργασίες, π.χ.: (r1, w1, r2), (r1, r2, w1), (r2, r1, w1).

Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential) (2/5)

- Η σειριακή συνέπεια εγγυάται ότι υπάρχει μια και **μοναδική καθολική διάταξη όλων των εγγραφών**, η οποία είναι ορατή από όλες τις CPU. Αν μια CPU βλέπει ότι η τιμή XXX γράφτηκε πρώτα στη θέση W τότε όλες οι CPU πρέπει να βλέπουν την ίδια διάταξη.
- Δεν είναι τόσο ισχυρός κανόνας όσο η αυστηρή συνέπεια.
- Είναι πολύ χρήσιμη και υλοποιείται.

Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential) (3/5)



Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential) (4/5)

Το ακόλουθο είναι σωστό:

P1: W(x)1 P1: W(x)1

P2: R(x)0 R(x)1 P2: R(x)1 R(x)1

Δύο πιθανά αποτελέσματα που τρέχουν το ίδιο πρόγραμμα:

a=1; b=1; c=1;

print(b,c); print(a,c); print(a,b);

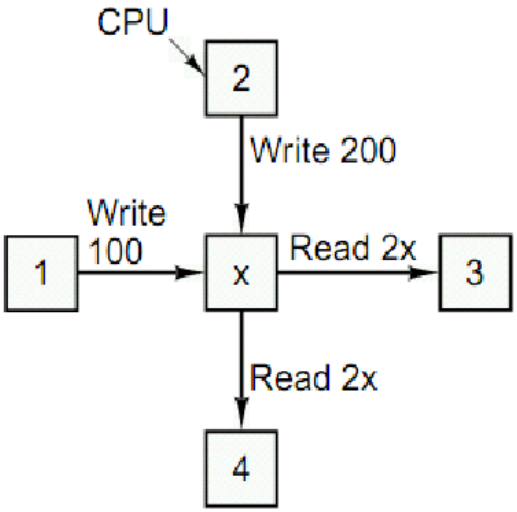
Τρεις παράλληλες διεργασίες: P1, P2, P3.

P1P2P3: 00 00 00 δεν επιτρέπεται.

P1P2P3: 00 10 01 δεν επιτρέπεται. (γιατί δεν ακολουθούν όλες οι διεργασίες την ίδια ακολουθία εγγραφών, π.χ. Η P1 βλέπει b=0,c=0 ενώ η P2 a=1,c=0).

Όλες οι διεργασίες βλέπουν όλες τις κοινές προσβάσεις στην ίδια διάταξη.

Μοντέλα Συνέπειας Μνήμης: Ακολουθιακό Μοντέλο (sequential) (5/5)



(a)

(b)	(c)	(d)
W100	W100	W200
W200	R3 = 100	R4 = 200
R3 = 200	W200	W100
R3 = 200	R4 = 200	R3 = 100
R4 = 200	R3 = 200	R4 = 100
R4 = 200	R4 = 200	R3 = 100

- (a) Δύο CPU's που κάνουν read και δύο που κάνουν write μιας κοινής μνήμης λέξη.
- (b – d) Τρεις πιθανοί τρόποι να γίνουν interleaved στο χρόνο τα δύο writes και τα τέσσερα reads.

Μοντέλα Συνέπειας Μνήμης: Αιτιατό Μοντέλο (causal)

Μοντέλα Συνέπειας Μνήμης: Αιτιατό Μοντέλο (causal) (1/3)

- Ακόμη πιο χαλαρό από το ακολουθιακό, προσφέρεται όμως για μεγαλύτερη παραλληλία.
- Όλες οι διεργασίες βλέπουν με την ίδια (σωστή) σειρά μόνον εκείνες τις λειτουργίες (read/write) οι οποίες πιθανώς σχετίζονται με **σχέση *αιτίου-αποτελέσματος*** (η μία δηλαδή επηρεάζει με οποιονδήποτε τρόπο την άλλη).
- Η υλοποίηση του μοντέλου απαιτεί την δημιουργία και συντήρηση ενός **dependency-graph** για τις λειτουργίες πρόσβασης κοινόχρηστης μνήμης.
- Κάτι τέτοιο είναι περίπλοκο, ενώ το μοντέλο απαιτεί επιπλέον συγχρονισμό από τις εφαρμογές εκείνες που απαιτούν ακολουθιακή συνέπεια.

Μοντέλα Συνέπειας Μνήμης: Αιτιατό Μοντέλο (causal) (2/3)

- Οι εγγραφές που σχετίζονται με **πιθανόν αιτιατή σχέση** πρέπει να μπορούν να τις βλέπουν όλες οι διεργασίες σε **μία ίδια διάταξη.** Σε διαφορετικές μηχανές μπορεί να βρεθούν ίσως τυχαίες εγγραφές σε διαφορετική διάταξη.

P1: W(x)1 W(x)3

P2: R(x)1 W(x)2

P3: R(x)1 R(x)3 R(x)2

P4: R(x)1 R(x)2 R(x)3

- Η ακολουθία αυτή επιτρέπεται στο αιτιατό μοντέλο συνέπειας μνήμης, αλλά όχι στο ακολουθιακό ή αυστηρό μοντέλο συνέπειας.

- Στο παράδειγμα έγινε η εγγραφή από την P1 W(x)1 και μετά από την P1 W(x)3, που σημαίνει ότι πρώτα θα εμφανιστεί το 1 και μετά το 3.

Μοντέλα Συνέπειας Μνήμης: Αιτιατό Μοντέλο (causal) (3/3)

Μια παράβαση αιτιατής μνήμης:

P1: W(x)1

P2: R(x)1 W(x)2

P3: R(x)2 R(x)1

P4: R(x)1 R(x)2

Μια σωστή ακολουθία των γεγονότων σε αιτιατή μνήμη:

P1: W(x)1

P2: W(x)2

P3: R(x)2 R(x)1

P4: R(x)1 R(x)2

Όλες οι διεργασίες βλέπουν όλες τις αιτιατά συσχετιζόμενες κοινές προσβάσεις σε μία ίδια διάταξη.

Μοντέλα Συνέπειας Μνήμης: Συνέπειας Επεξεργαστή

Μοντέλα Συνέπειας Μνήμης: Συνέπειας Επεξεργαστή (1/3)

  - Χαλαρό μοντέλο συνέπειας.
  - Ευκολότερο να υλοποιηθεί στους μεγάλους πολυεπεξεργαστές.
- **Ιδιότητες**:
  - Οι εγγραφές από οποιαδήποτε δεδομένη CPU φαίνονται από όλες τις CPU με τη σειρά που υποβλήθηκαν.
  - Για κάθε δεδομένη λέξη μνήμης, όλες οι CPU βλέπουν όλες τις εγγραφές που γίνονται σε αυτή με την ίδια σειρά.

Μοντέλα Συνέπειας Μνήμης: Συνέπειας Επεξεργαστή (2/3)

- **Επεξήγηση ιδιοτήτων:**
  - Αν η CPU1 υποβάλει τις πράξεις εγγραφής 1A,1B,1C σε κάποια θέση μνήμης, τότε όλοι οι άλλοι επεξεργαστές τις βλέπουν επίσης με αυτή τη σειρά.
  - Σε κάθε λέξη μνήμης υπάρχει μια μονοσήμαντη τιμή.
  - Όλες οι CPU πρέπει να συμφωνούν ποια πράξη εγγραφής έγινε τελευταία.
- **Παράδειγμα:**
  - Έστω ταυτόχρονα CPU1: 1A,1B,1C και CPU2 2A,2B,2C στην ίδια θέση μνήμης.
  - Και τα δυο είναι έγκυρα: 1A,1B,2A,2B,2C,1C ή
  - 2A,1A,2B,2C,1B,1C.

Μοντέλα Συνέπειας Μνήμης: Συνέπειας Επεξεργαστή (3/3)

- Δεν εγγυάται ότι όλες οι CPU θα βλέπουν την ίδια διάταξη.
- Είναι απόλυτα νόμιμο να συμπεριφέρεται το υλικό με τέτοιον τρόπο ώστε, μερικές CPU να βλέπουν την πρώτη από τις παραπάνω διατάξεις και μερικές να βλέπουν άλλες.
- Καμία CPU δε θα βλέπει την 1Β όμως να προηγείται της 1Α.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak)

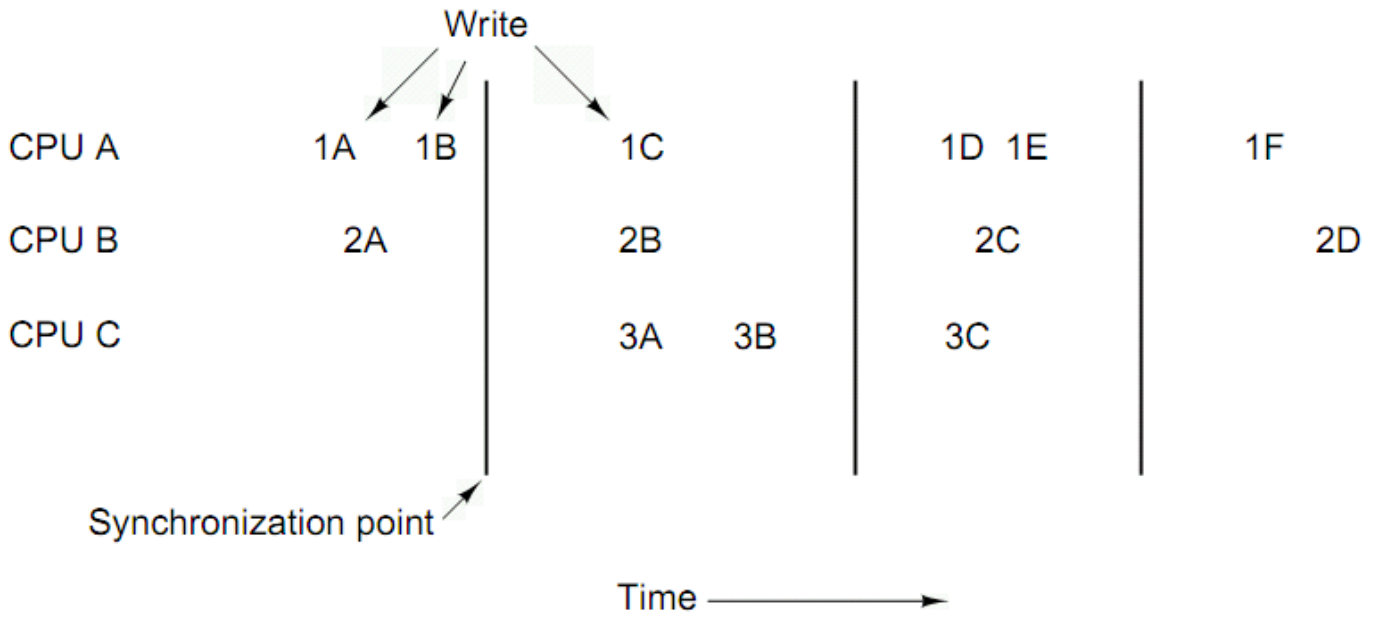
Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (1/9)

- **Βασίζεται στις παρατηρήσεις ότι:**
- Συνήθως δεν απαιτείται η γνωστοποίηση σε όλες τις διεργασίες κάθε μεμονωμένης αλλαγής σε μία θέση μνήμης, **αλλά η γνωστοποίηση ομαδοποιημένων αλλαγών**, σε όποιες διεργασίες το ζητήσουν.
- **Η μεμονωμένη προσπέλαση κοινόχρηστων μεταβλητών είναι σπάνια.** Συνήθως έχουμε αρκετές προσπελάσεις σε ένα σύνολο μεταβλητών και μετά καθόλου προσπέλαση για ένα μεγάλο χρονικό διάστημα.
- Η **συνέπεια επιβάλλεται σε ομάδα προσπελάσεων μνήμης** και όχι σε μεμονωμένες προσπελάσεις.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (2/9)

- Δεν εγγυάται ούτε καν ότι οι πράξεις εγγραφής από μια CPU θα εμφανίζονται με τη σειρά.
- Μπορεί μια CPU να βλέπει την 1Α εγγραφή πριν από την 1Β και μια άλλη CPU την 1Α μετά την 1Β.
- Διαθέτει **μεταβλητές συγχρονισμού** ή μια πράξη συγχρονισμού.
- Όταν εκτελείται ένας συγχρονισμός όλες οι εγγραφές που είναι σε αναμονή τελειώνουν (αδειάζει η γραμμή διοχέτευσης).
- Οι πράξεις συγχρονισμού είναι **σειριακά συνεπείς**, δηλαδή όταν τις υποβάλλουν πολλές CPU, επιλέγεται μια διάταξη και όλες οι CPU βλέπουν αυτή τη διάταξη.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (3/9)



- Το αδύνατο μοντέλο συνέπειας μνήμης χρησιμοποιεί λειτουργίες συγχρονισμού για να διαιρέσει το χρόνο σε περιόδους.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (4/9)

- **Χαλαρή διάταξη προγραμμάτων μεταξύ όλων των λειτουργιών στη μνήμη.**
- Ανάγνωση/Εγγραφές σε διαφορετικές λειτουργίες της μνήμης μπορούν να επαναδιαταχθούν.

- **Λάβετε υπόψη ότι:**
- Για μια λειτουργία σε κρίσιμη περιοχή (κοινή).
- Και μια διεργασία που διαβάζει/γράφει.
- Κανείς άλλος δεν έχει πρόσβαση μέχρι η διεργασία να αφήσει την κρίσιμη περιοχή.
- Δεν υπάρχει ανάγκη για διάδοση εγγραφών ακολουθιακά ή καθόλου μέχρι η διεργασία να αφήσει την κρίσιμη περιοχή.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (5/9)

- Το αδύνατο μοντέλο συνέπειας έχει **τρεις ιδιότητες**:
  - Οι προσβάσεις σε μεταβλητές συγχρονισμού είναι ακολουθιακά συνεπείς.
  - Σε μία μεταβλητή συγχρονισμού δεν επιτρέπεται άδεια για να εκτελεστεί μέχρι να ολοκληρωθούν όλες οι προηγούμενες εγγραφές.
  - Δεν επιτρέπεται άδεια πρόσβασης σε δεδομένα (ανάγνωση και εγγραφή) μέχρις ότου όλες οι προηγούμενες προσβάσεις στις μεταβλητές συγχρονισμού να έχουν τελειώσει.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (6/9)

Μια έγκυρη ακολουθία γεγονότων για το αδύνατο μοντέλο συνέπειας:

P1: W(x)1 W(x)2 S

P2: R(x)1 R(x)2 S

P3: R(x)2 R(x)1 S

Μια μη έγκυρη ακολουθία γεγονότων για το αδύνατο μοντέλο συνέπειας:

P1: W(x)1 W(x)2 S

P2: S R(x)1

Η P2 πρέπει να πάρει 2 αντί για 1 επειδή έχει ήδη συγχρονιστεί.

Τα κοινά δεδομένα μπορούν να υπολογιστούν σαν συνεπή αφότου γίνει ο συγχρονισμός.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (7/9)

Μεταβλητή συγχρονισμού (barrier).

||

Λειτουργία για το συγχρονισμό της μνήμης.

- Όλες οι τοπικές εγγραφές μεταδίδονται.
- Όλες οι απομακρυσμένες εγγραφές μεταφέρονται στο τοπικό επεξεργαστή.
- Μπλοκάρισμα μέχρι να συγχρονιστεί η μνήμη.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (8/9)

- **Η πρόσβαση σε μεταβλητές συγχρονισμού είναι ακολουθιακά συνεπής.**
  - Όλες οι διεργασίες τις βλέπουν σε μία ίδια διάταξη.
  - Σε μία μεταβλητή συγχρονισμού δεν επιτρέπεται άδεια για να εκτελεστεί μέχρι να ολοκληρωθούν όλες οι προηγούμενες εγγραφές.
  - Δεν επιτρέπεται άδεια πρόσβασης σε δεδομένα (ανάγνωση και εγγραφή) μέχρις ότου όλες οι προηγούμενες προσβάσεις στις μεταβλητές συγχρονισμού να έχουν τελειώσει.
  - Η μνήμη ενημερώνεται κατά τη διάρκεια του συγχρονισμού.

Μοντέλα Συνέπειας Μνήμης: Αδύνατο Μοντέλο (weak) (9/9)

- **Προβλήματα με τη συνέπεια συγχρονισμού.**
- **Αναποτελεσματικότητα:**
  - Συγχρονισμός.
  - Επειδή μια διεργασία τελείωσε με τις προσπελάσεις της μνήμης ή τώρα πρόκειται να ξεκινήσει;
  - Χαμηλή αποδοτικότητα (πρέπει να τελειώνει όλες τις εκκρεμείς πράξεις & να εμποδίζει όλες τις καινούργιες).
- **Τα συστήματα πρέπει να επιβεβαιώνουν ότι:**
  - Όλες οι εγγραφές που ξεκίνησαν τοπικά θα πρέπει να ολοκληρωθούν.
  - Όλες οι απομακρυσμένες εγγραφές έχουν αποκτηθεί.

Μοντέλα Συνέπειας Μνήμης: Eager Release Consistency (ERC)

Μοντέλα Συνέπειας Μνήμης: Eager Release Consistency (ERC) (1/4)

- Η ιδέα της συνέπειας αποδέσμευσης είναι ότι όταν μια διεργασία εξέρχεται από μια κρίσιμη περιοχή, **δεν είναι απαραίτητο να αναγκάσει** όλες τις εγγραφές να ολοκληρωθούν αμέσως.
- Είναι απαραίτητο να εξασφαλίσει ότι **θα έχουν τελειώσει** πριν οποιαδήποτε διεργασία εισέλθει ξανά σε αυτή την κρίσιμη περιοχή.

Μοντέλα Συνέπειας Μνήμης: Eager Release Consistency (ERC) (2/4)

- Η ERC δεν αναγκάζει τις εκκρεμείς εγγραφές να ολοκληρωθούν, αλλά η ίδια δεν ολοκληρώνεται μέχρι να ολοκληρωθούν όλες οι εγγραφές που έχουν υποβληθεί προηγουμένως.
- Οι νέες πράξεις μνήμης δεν εμποδίζονται να αρχίσουν αμέσως.
- Όταν υποβληθεί η επόμενη acquire, γίνεται ένας έλεγχος για να διαπιστωθεί αν έχουν ολοκληρωθεί όλες οι προηγούμενες πράξεις release. Αν όχι, τότε μένει σε αναμονή.
- Αν η επόμενη acquire είναι αρκετά αργότερα, τότε θα έχουν ολοκληρωθεί η εγγραφές και δε θα χρειάζεται να περιμένει.
- Πιο πολύπλοκος μηχανισμός, αλλά δεν καθυστερεί τις εντολές τόσο συχνά.

Μοντέλα Συνέπειας Μνήμης: Eager Release Consistency (ERC) (3/4)

- **Μπορούμε να τα πάμε καλύτερα;**
- **Μπορούμε να κάνουμε το κάτι παραπάνω;**
  - **Διαχωρίζει τον συγχρονισμό σε δύο στάδια:**
    - **Απόκτηση πρόσβασης:**
      - Λήψη έγκυρων αντίγραφων των σελίδων (αποκλειστική πρόσβασή).
    - **Αποδέσμευση πρόσβασης:**
      - Στέλνει ακυρώσεις ή ενημερώσεις για κοινές σελίδες που έχουν τροποποιηθεί τοπικά σε κόμβους που έχουν αντίγραφα.

- **acquire(R) //αρχή της κρίσιμης περιοχής**

- ***Κάνει διάφορα πράγματα***

- **release(R)//τέλος της κρίσιμης περιοχής**

Μοντέλα Συνέπειας Μνήμης: Eager Release Consistency (ERC) (4/4)

- Το αποδεσμευτικό μοντέλο συνέπειας παρέχει προσβάσεις acquire και release. Οι προσβάσεις acquire συνήθως λένε πότε πρόκειται να μπουν κρίσιμες περιοχές στη μνήμη του συστήματος. Οι προσβάσεις release λένε πως μόλις βγήκαν από μία κρίσιμη περιοχή.
- **Μια έγκυρη ακολουθία γεγονότων για το αποδεσμευτικό μοντέλο συνέπειας.**

P1: Acq(L) W(x)1 W(x)2 Rel(L)
P2: Acq(L) R(x)2 Rel(L)
P3: R(x)1

- Η P3 δεν γίνεται acquire, οπότε το αποτέλεσμα δεν είναι **εγγυημένο**.

Μοντέλα Συνέπειας Μνήμης: Lazy Release Consistency (LRC)

Μοντέλα Συνέπειας Μνήμης: Lazy Release Consistency (LRC) (1/3)

- **Η αποδέσμευση απαιτεί:**
  - Αποστολή ακυρώσεων σε copyset κόμβους,
  - Και αναμονή όλων για επιβεβαίωση.
- **Μην κάνετε τροποποιήσεις καθολικά ορατές κατά την αποδέσμευση.**
- **Κατά την αποδέσμευση**:
  - Αποστέλλεται ακύρωση μόνο στον κατάλογο,
  - ή αποστέλλονται ενημερώσεις στον home κόμβο (κάτοχος σελίδας).
- **Κατά την acquire**: εκεί όπου οι τροποποιήσεις διεξάγονται:
  - Έλεγχος με τον κατάλογο για δει εάν χρειάζεται νέο αντίγραφο.
  - Οι πιθανότητες δεν είναι με το ότι κάθε κόμβος χρειάζεται να κάνει acquire.
- **Μειώνει την κίνηση των μηνυμάτων στις αποδεσμεύσεις.**

Μοντέλα Συνέπειας Μνήμης: Lazy Release Consistency (LRC) (2/3)

- Τα κοινά δεδομένα γίνονται συνεπή όταν εξέλθουν από μία κρίσιμη περιοχή.
- Στο Lazy Release μοντέλο συνέπειας, κατά τη στιγμή μιας αποδέσμευσης, τίποτα δεν στέλνεται πουθενά. Αντίθετα, όταν γίνεται μια acquire, **ο επεξεργαστής που προσπαθεί να κάνει την acquire** πρέπει να πάρει τις πιο πρόσφατες τιμές των μεταβλητών της μηχανής ή των μηχανών που τις κρατούν.

Μοντέλα Συνέπειας Μνήμης: Lazy Release Consistency (LRC) (3/3)

- **Παράδειγμα:** Home-based Lazy Release Consistency.
- **Κατά την αποδέσμευση:**
  - Υπολογίζονται τα Diffs.
  - Αποστέλλονται στον ιδιοκτήτη (home node).
- ***Home Node****: Εφαρμόζει τα diffs όσο πιο γρήγορα γίνεται με το που φτάνουν.*
- **Κατά την acquire:**
  - Οι αιτήσεις του κόμβου ενημέρωσαν τη σελίδα από τον home node.

2 οι πιο δημοφιλείς προσεγγίσεις

- **Οι προσεγγίσεις διαιρούνται σε 2 κατηγορίες:**
  - Προσεγγίσεις λογισμικού.
  - Προσεγγίσεις υλικού.

Πλεονεκτήματα & Μειονεκτήματα των λύσεων με λογισμικό

- Πιο απλές υλοποιήσεις.
- Δεν απαιτείται υλικό.
- Φθηνές λύσεις.
- Στηρίζονται στον compiler και στο λειτουργικό σύστημα.
- Το πρόβλημα μεταφέρεται από το χρόνο εκτέλεσης στο χρόνο στοιχειομετάφρασης.
- Όμως, λαμβάνονται συντηρητικές αποφάσεις, δηλαδή μη αποδοτική χρήση της cache.

Η βασική ιδέα των λύσεων με λογισμικό

- Ανάλυση του κώδικα.
- Εύρεση των αντικειμένων που ίσως γίνουν ανασφαλή για τοποθέτηση στη μνήμη cache.
- Κατάλληλη σημείωση αυτών των αντικειμένων.
- Το ΛΣ ή οι CPU δεν επιτρέπουν την τοποθέτηση στη cache αυτών των αντικειμένων.
- **Πολύ συντηρητικό, επειδή κοινά δεδομένα μπορούν να μη χρησιμοποιούνται ταυτόχρονα (διαφορετικές χρονικές περιόδους), αλλά αποκλειστικά κάθε φορά.**

Βελτίωση της βασικής ιδέας

- Προσθήκη επιπρόσθετου βήματος για εύρεση ασφαλών χρονικών περιόδων.
- Εισάγονται εντολές που δείχνουν τα ασφαλή (ή τα μη ασφαλή) τμήματα.
- Επιβάλλεται συμφωνία μνημών cache κατά τη διάρκεια μόνο των κρίσιμων χρονικών περιόδων.

Πλεονεκτήματα & μειονεκτήματα των λύσεων με υλικό

- Υπάρχει δυναμική αναγνώριση κατά το χρόνο εκτέλεσης.
- Το πρόβλημα αντιμετωπίζεται μόνο όταν εμφανιστεί.
- Βελτιωμένη λειτουργία σε σχέση με την προσέγγιση λογισμικού.
- Είναι διαφανείς για τον προγραμματιστή, ελαττώνει τον προγραμματιστικό φόρτο.
- Αποτελεσματική χρήση της cache.
- Δύσκολο στην υλοποίηση.

Λύσεις με υλικό τρόπο

- **Διαίρεση σε 2 κατηγορίες:**
  - Πρωτόκολλα καταλόγου.
  - Πρωτόκολλα Κρυφών μνημών (Cache protocols).

Μειονεκτήματα & πλεονεκτήματα της χρήσης καταλόγου

- Υποφέρουν από:
  - τα μειονεκτήματα μιας κεντρικής συμφόρησης.
  - Από την επιβάρυνση επικοινωνίας μεταξύ των διαφόρων ελεγκτών cache και του κεντρικού ελεγκτή.
- Είναι αποτελεσματικά σε συστήματα μεγάλης κλίμακας που περιέχουν πολλές αρτηρίες ή κάποια άλλη πολύπλοκη μορφή διασύνδεσης.

Λειτουργία των πρωτοκόλλων κρυφών μνημών

- Κατανέμεται η ευθύνη για διατήρηση της συμφωνίας μνημών cache μεταξύ όλων των ελεγκτών.
- Μια μνήμη cache θα πρέπει να αναγνωρίζει πότε μια γραμμή που έχει διαμοιράζεται με άλλες cache.
- Όταν εκτελείται μια ανανέωση/ενημέρωση σε διαμοιραζόμενη γραμμή θα πρέπει να ανακοινώνεται σε όλες τις άλλες μνήμες cache με ένα μηχανισμό μετάδοσης.
- Οι άλλες cache θα πρέπει να “κρυφακούουν” στο δίκτυο για να αντιλαμβάνεται αυτές τις ειδοποιήσεις.
- Ιδανικά για πολυεπεξεργαστή με βάση αρτηρία.

Ποιο μοντέλο εισάχθηκε στον Pentium 4

- Η κατάσταση της κάθε γραμμής σημειώνεται με δυο επιπλέον bit στην ετικέτα της μνήμης cache.

- **Υπάρχουν 4 καταστάσεις:**
  - Modified (τροποποιημένη).
  - Excluse (αποκλειστική).
  - Shared (διαμοιραζόμενη).
  - Invalid (άκυρη).
- =>M.E.S.I.

Τέλος Ενότητας