Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 6: Αρχιτεκτονικές Μνημών UMA, NUMA

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

- Η κατανόηση των βασικών αρχιτεκτονικών διαμοιραζόμενης μνήμης παράλληλων συστημάτων.
- Η παρουσίαση παραδειγμάτων παραλληλοποίησης.

Οργάνωση μνήμης σε κατανεμημένα συστήματα

- **Λογικά κατανεμημένη μνήμη:** κάθε μνήμη έχει δικιά της διεύθυνση και μπορεί να προσπελαστεί μόνο από τους επεξεργαστές στους οποίους ανήκει (αν κάποιος εξωτερικός επεξεργαστής απαιτεί κάτι από τη μνήμη, πρέπει να το ζητήσει από τον τοπικό επεξεργαστή).
- **Κοινόχρηστη κατανεμημένη μνήμη (Distributed Shared Memory):** Όλες οι μνήμες είναι χωρικά κατανεμημένες, αλλά αποτελούν ένα ενιαίο χώρο διευθύνσεων. Έτσι, ένας απομακρυσμένος επεξεργαστής, μπορεί να γράψει άμεσα σε μια μνήμη που βρίσκεται στην τοπική μνήμη ενός άλλου επεξεργαστή.

Κατηγοριοποίηση των συστημάτων μοιραζόμενης μνήμης

- **Uniform Memory Access** (UMA) – **Ομοιόμορφη προσπέλαση μνήμης.**
- **NonUniform Memory Access** (NUMA) – **Μη ομοιόμορφη προσπέλαση μνήμης.**
- **Cache-Only Memory Architecture** (COMA).

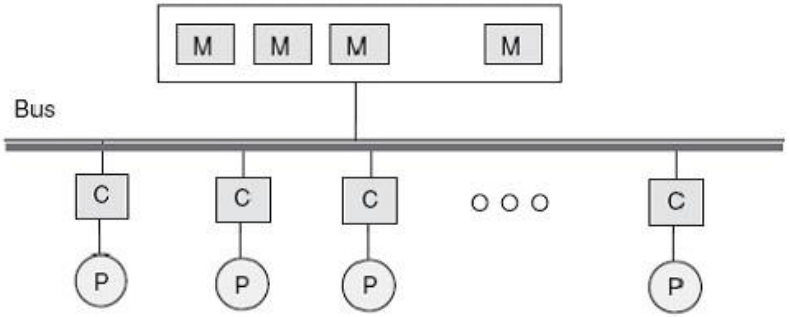
Η αρχιτεκτονική μνήμης Uniform Memory Access (UMA)

Uniform Memory Access (UMA) – Ομοιόμορφη προσπέλαση μνήμης (1/2)

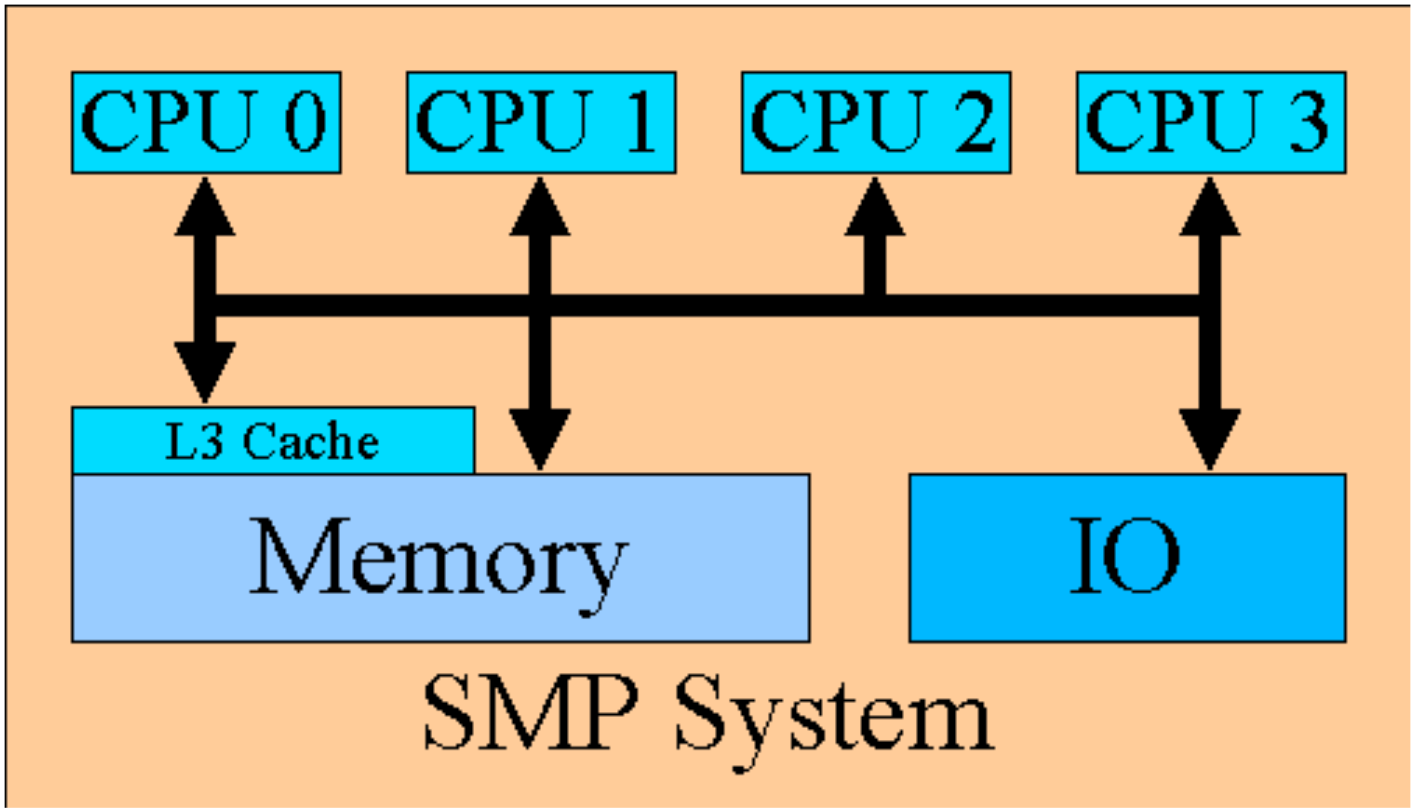
- Η κοινή μνήμη είναι **προσβάσιμη** από όλους τους επεξεργαστές μέσω ενός δικτύου διασύνδεσης **με τον ίδιο τρόπο** που ένας και μόνο επεξεργαστής έχει πρόσβαση στην μνήμη.
- Το δίκτυο δίκτυο διασύνδεσης μπορεί να είναι: **απλός δίαυλος**, **πολλαπλός δίαυλος**, **διασταύρωσης** (single bus, multiple buses, crossbar).
- Κάθε επεξεργαστής **έχει ίση ευκαιρία** για εγγραφή/ανάγνωση στη μνήμη καθώς και ίση ταχύτητα πρόσβασης.
- **SMP** systems (Symmetric Multiprocessor systems): Επειδή η πρόσβαση στη μνήμη είναι ισορροπημένη.

Uniform Memory Access (UMA) – Ομοιόμορφη προσπέλαση μνήμης (2/2)

- Η UMA οργάνωση Μνήμης είναι από τις πιο δημοφιλείς στα συστήματα διαμοιραζόμενης μνήμης.
- Εμπορικά παραδείγματα παραδείγματα: Sun Starfire Servers, HP V series, Compaq AlphaServer GS.

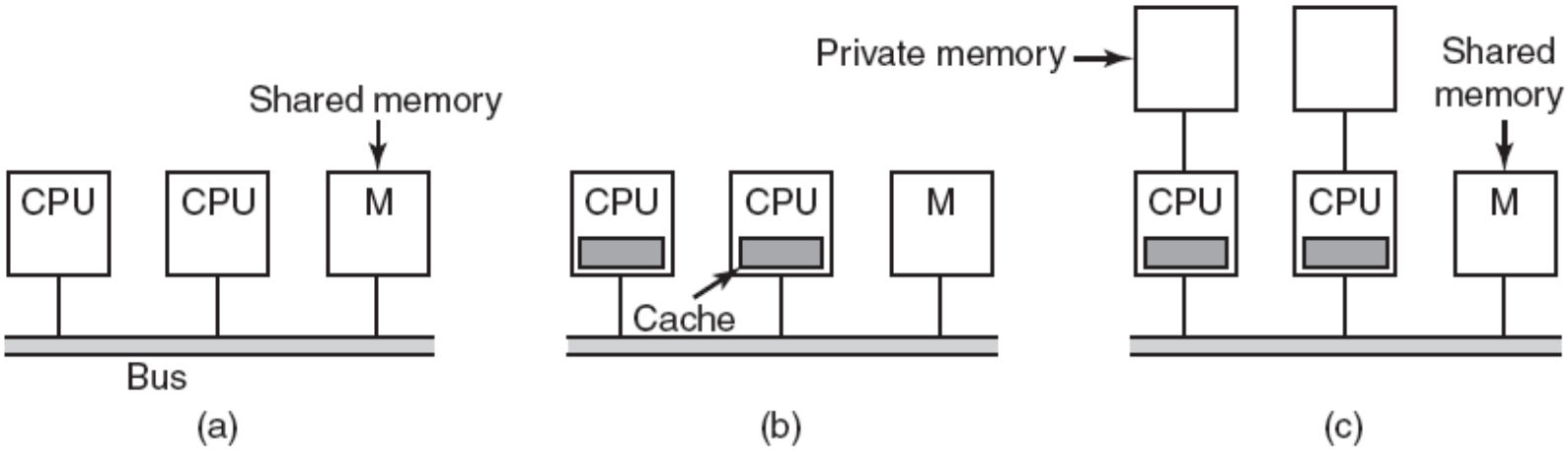


Ένα τυπικό σύστημα SMP



- Καθυστέρηση L1,L2,L3,memory.

Συστήματα UMA με σύνδεση απλού διαύλου (1/2)



- Three bus-based multiprocessors. (a) Without caching. (b) With caching. (c) With caching and private memories.
- UMA : Uniform access to the entire memory, same access times for all CPU's.

Συστήματα UMA με σύνδεση απλού διαύλου (2/2)

- Each CPU has to wait for the bus to be idle to read or write to the memory.
- For 2 or 3 computers, bus contention is manageable (a).
- Δημιουργείται **πρόβλημα συναγωνισμού** στη μνήμη αφού όλοι οι επεξεργαστές χρησιμοποιούν μια κοινή μνήμη.
- For larger number of CPU's, a cache is added to the CPU. Since reads can be satisfied by cache contents, there will be less traffic on the bus (b). **Επιλύει** εν μέρει το πρόβλημα του συναγωνισμού.
- Writing has to be managed!
- Some systems have private and shared memories (c).
- Mostly private memory is used. Shared memory is for shared variables between CPUs.
- Needs carefull programming!

Συστήματα UMA

- Η χρήση κρυφής μνήμης αντιμετωπίζει το πρόβλημα του ανταγωνισμού της μνήμης, αλλά δημιουργεί ένα άλλο πρόβλημα.
- Το πρόβλημα που δημιουργείται είναι της **συνάφειας μνήμης** (επίσης ονομάζεται **συνέπεια κρυφής μνήμης**), και εμφανίζεται όταν κάποιος επεξεργαστής γράφει σε μια θέση μνήμης, η οποία βρίσκεται στη κρυφή μνήμη άλλου επεξεργαστή. Ο άλλος επεξεργαστής θα έχει λοιπόν άκυρα δεδομένα.

Μια λύση στο πρόβλημα της συνέπειας κρυφής μνήμης

- Τεχνική **write-through.**
- Μόλις κάποιος επεξεργαστής γράψει, γίνεται broadcast της τιμής στον κοινό διάδρομο.
- Όλοι οι επεξεργαστές ακούν τον κοινό διάδρομο.
- Αν κάποιος κρατάει αντίγραφο τότε ενημερώνει ανάλογα την αντίστοιχη θέση.

UMA με πολλαπλά αρθρώματα

- Μια λύση στον ανταγωνισμό της μνήμης είναι η χρήση πολλαπλών τμημάτων ή αρθρωμάτων μνήμης.
- Απαιτείται δίκτυο διασύνδεσης που να επιτρέπει ταυτόχρονη πρόσβαση σε διαφορετικές μονάδες μνήμης.
- Ασφαλώς όλα τα τμήματα δημιουργούν μια κοινή μνήμη που όλοι οι επεξεργαστές έχουν ίδιο χρόνο πρόσβασης σε κάθε διεύθυνση.
- Μπορεί να εμφανιστεί συνωστισμός αν απαιτούνται πολλαπλές ταυτόχρονες προσβάσεις στο ίδιο τμήμα μνήμης.

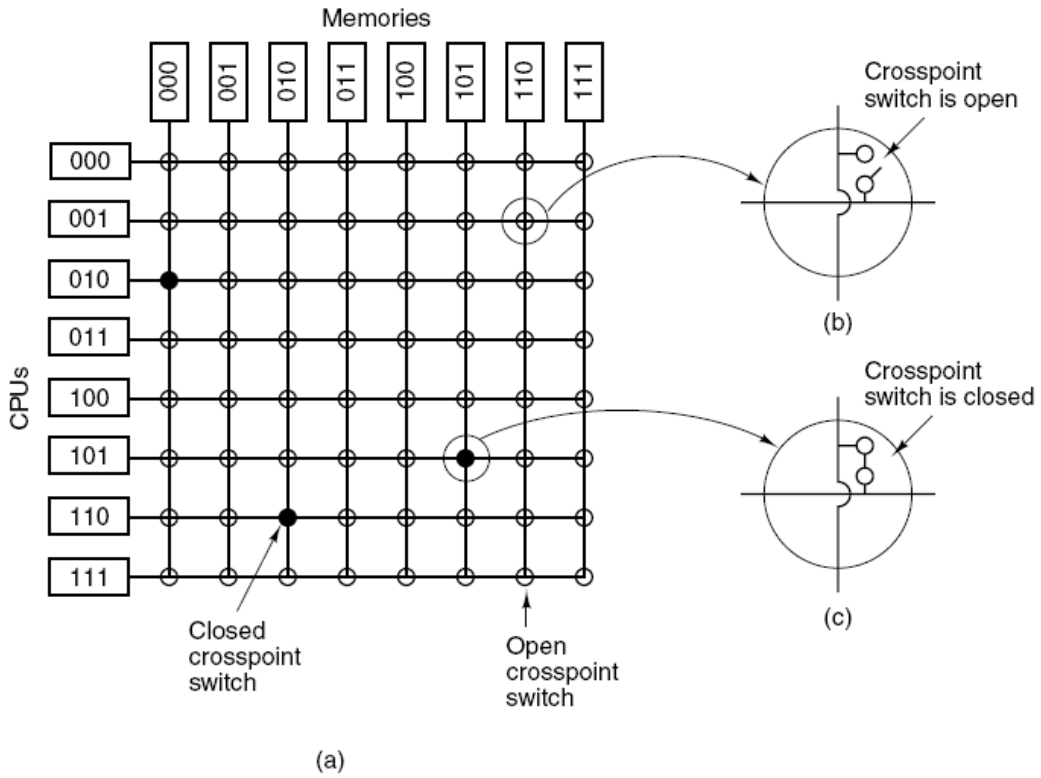
Τεχνική της μη διαδοχικής διευθυνσιοδότησης

- Μια τεχνική επίλυση είναι **να μην είναι συνεχόμενες οι διευθύνσεις** μνήμης σε κάθε τμήμα.

Μονάδα Μνήμης #1	Μονάδα Μνήμης #2	Μονάδα Μνήμης #3
Διεύθυνση 0	Διεύθυνση 1	Διεύθυνση 2
Διεύθυνση 3	Διεύθυνση 4	Διεύθυνση 5
Διεύθυνση 6	Διεύθυνση 7	Διεύθυνση 8
Διεύθυνση 9	Διεύθυνση 10	Διεύθυνση 11

- Έτσι αν οι επεξεργαστές ζητήσουν τις διευθύνσεις μνήμης 0,1,2,3,4,5 τότε δε θα παρουσιαστεί συνωστισμός.

Συστήματα UMA με σύνδεση δίαυλο διασταύρωσης (1/2)



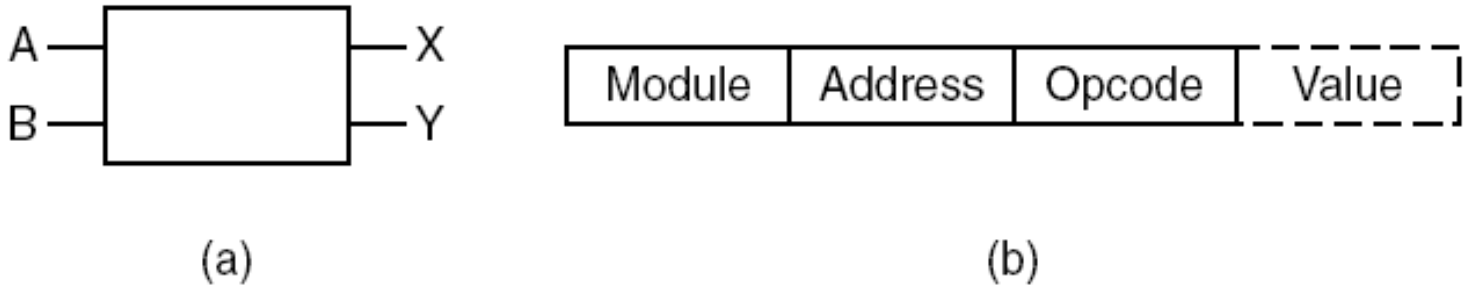
a) An 8 × 8 crossbar switch.

b) An open crosspoint. (c) A closed crosspoint.

Συστήματα UMA με σύνδεση δίαυλο διασταύρωσης (2/2)

- Use of a single bus limits (even with caches) the number of CPUs to about 16 or 32 CPUs.
- A **crossbar switch** connecting n CPUs to k memories may solve this problem.
- A **crosspoint** is a small electronic switch.
- Contention for memory is still possible if k < n. **Partitioning** the memory into n units may reduce the contention.

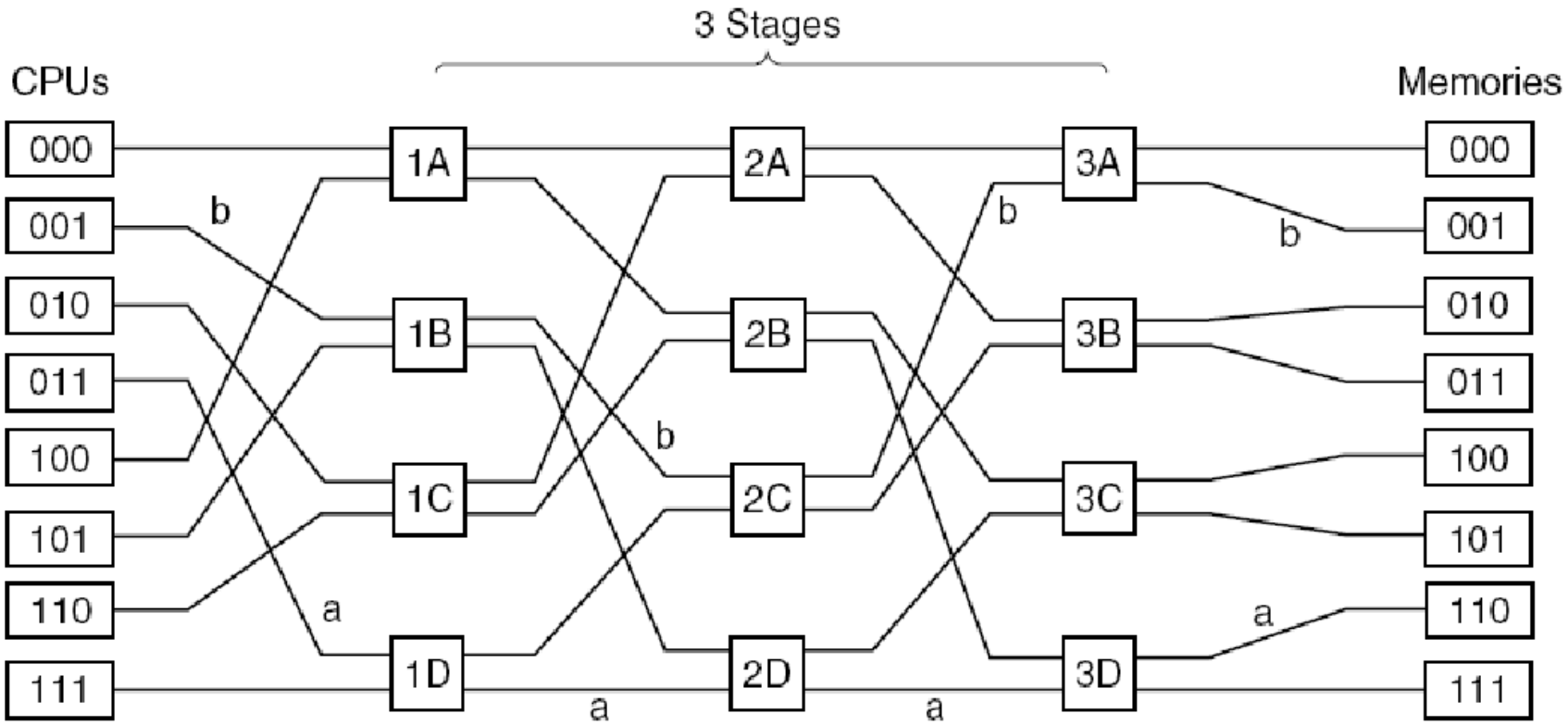
Συστήματα UMA με σύνδεση διακόπτη (switch) (1/2)



(a) A 2 × 2 switch with two input lines, A and B, and two output lines, X and Y. (b) A message format.

- Module: memory unit.
- Address: an address within a module.
- Opcode: Read or Write.
- Value: value to be written.

Συστήματα UMA με σύνδεση διακόπτη (switch) (2/2)



An omega switching network.

Η αρχιτεκτονική μνήμης NUMA

Κατηγοριοποιήσεις NUMA

- **Simple NUMA** cache coherence is not kept by the hardware (CM*,Cenju, T3D, RWC-1, Earth simulator).
- **CC (Cache Coherent)-NUMA** providing coherent cache. (DASH, Alewife, Origin, SynfinityNUMA, NUMA-Q).
- **COMA** (Cache Only Memory Architecture): No home memory (DDM,KSR-1).

Η αρχιτεκτονική μνήμης Simple NUMA

Non Uniform Memory Access (NUMA) –Μη ομοιόμορφη προσπέλαση μνήμης

- Κάθε επεξεργαστής έχει άμεση & ταχύτατη πρόσβαση σε **ένα τμήμα** της κοινής μνήμης.
- Υπάρχει ένας **κοινός χώρος διεύθυνσης** μνήμης (shared address space).
- Ο **χρόνος πρόσβασης** σε κάθε άρθρωμα μνήμης **εξαρτάται** από την απόσταση του κάθε επεξεργαστή από αυτό ==> Μη Ομοιόμορφη προσπέλαση μνήμης.
- Το δίκτυο διασύνδεσης μπορεί να είναι: ιεραρχικοί δίαυλοι, δένδρα, (hierarchical buses, tree).
- Η απόδοση εξαρτάται από την «**τοπικότητα** των δεδομένων» -- data locality δηλ για το αν οι αιτήσεις από τους επεξεργαστές για δεδομένα αφορούν τοπικές ή όχι θέσεις μνήμης.

Non Uniform Memory Access (1/2)

- Providing shared memory whose access latency and bandwidth are different by the address.
- Usually, its own memory module is easy to be accessed, but ones with other PUs are not.
- All shared memory modules are mapped into a unique logical address space, thus the program for UMA machines works without modification.
- Also called a machine with Distributed Shared Memory
- ⇔A machine with Centralized Shared memory (UMA).

Non Uniform Memory Access (2/2)

- A PU can access memory with other PUs/Clusters, but the cache coherence is not kept.
- Simple hardware.
- Software cache support functions are sometimes provided.
- Suitable for connecting a lot of PUs: Supercomputers: Cenju, T3D, Earth simulator, IBM BlueGene, Roadrunner.
- Why recent top supercomputers take the simple NUMA structure;
  - Easy programming for wide variety of applications.
  - Powerful interconnection network.

NUMA και SMP

- Η NUMA έχει σχεδιαστεί για να **ξεπεράσει τα όρια κλιμάκωσης** που υπάρχουν στα SMP (UMA).
- Ασφαλώς απαιτείται διασύνδεση υψηλής ταχύτητας.
- Και η αρχιτεκτονική NUMA και η UMA έχουν κοινό χώρο διευθύνσεων μνήμης.

Ορισμός της τοπικής μνήμης

- Τι ονομάζουμε **τοπική μνήμη**;
  - Η μνήμη που βρίσκεται στον ίδιο δίαυλο με τον επεξεργαστή που εκτελεί μια διεργασία. Κάθε μνήμη που δεν ανήκει σε αυτή την κατηγορία ονομάζεται απομακρυσμένη μνήμη.

Μη ομοιόμορφη προσπέλαση μνήμης

- Characteristics of NUMA machines:
  - There is a single address space visible to all CPUs (σημαντικό στοιχείο: Κοινός χώρος διευθύνσεων).
  - Access to remote memory is via LOAD and STORE instructions.
  - Access to remote memory is slower than access to local memory.

Η αρχιτεκτονική μνήμης ccNUMA

Αρχιτεκτονική ccNUMA (1/2)

- Σχεδόν όλοι οι υπολογιστές έχουν μια τοπική cache εκτός από την τοπική μνήμη για να εκμεταλλευτούν την τοπικότητα της αναφοράς.
- Απαιτείται ένας μηχανισμός για τη συνέπεια μνήμης.
- Η εγκατάσταση μιας cache απαιτεί ειδικό εξοπλισμό προκειμένου να διατηρηθεί η συνάφεια της μνήμης.
- Έτσι δημιουργούνται οι υπολογιστές NUMA με συνάφεια κρυφής μνήμης (Cache Coherant NUMA).

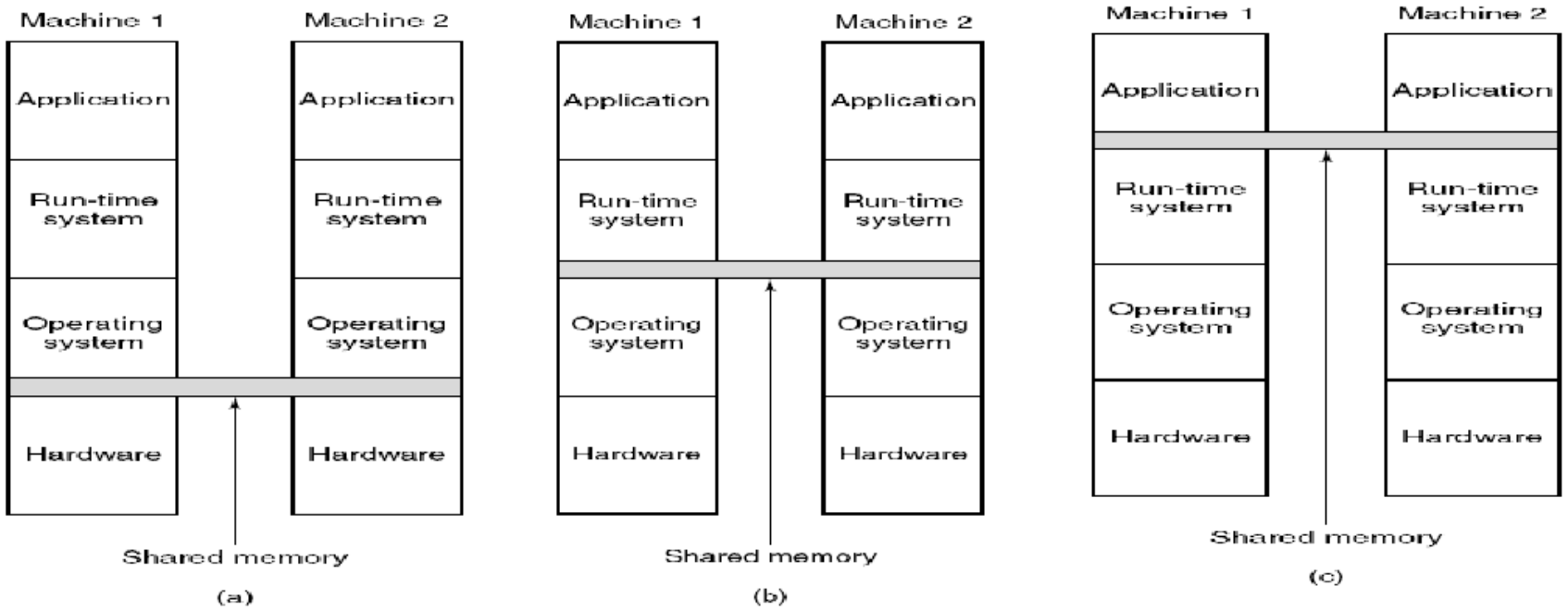
Αρχιτεκτονική ccNUMA (2/2)

- Για να διατηρηθεί η συνάφεια **χρησιμοποιείται δια-επεξεργαστική επικοινωνία** ανάμεσα στους ελεγκτές κρυφής μνήμης.
- Υπάρχει αρκετά μειωμένη απόδοση αν πολλαπλοί επεξεργαστές προσπαθούν να προσπελάσουν την ίδια περιοχή μνήμης σχεδόν την ίδια στιγμή.
- Τα λειτουργικά συστήματα που υποστηρίζουν NUMA προσπαθούν να ελαχιστοποιήσουν αυτές τις περιπτώσεις.
- Χρησιμοποιούνται ειδικά πρωτόκολλα συνάφειας μνήμης για να μειώνεται η επικοινωνία στο ελάχιστο.

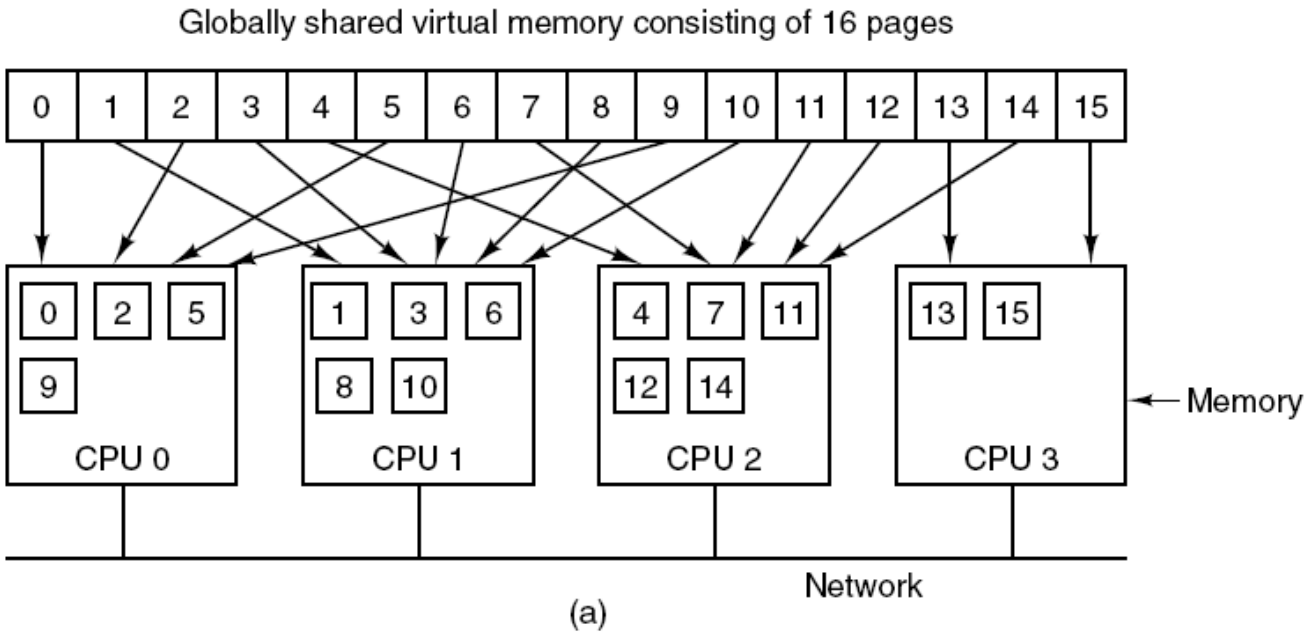
NUMA/συστοιχία υπολογιστών

- Η NUMA μπορεί να χαρακτηριστεί ως ένα είδος συστοιχίας υπολογιστών με πολύ ισχυρή ζεύξη.
- Η NUMA μπορεί να υλοποιηθεί εξολοκλήρου σε software ή σε hardware.
- Ασφαλώς η υλοποίηση με λογισμικό προκαλεί πάρα πολύ μεγάλη καθυστέρηση.

Η χρήση της διαμοιραζόμενης μνήμης μπορεί να γίνει σε διάφορα επίπεδα



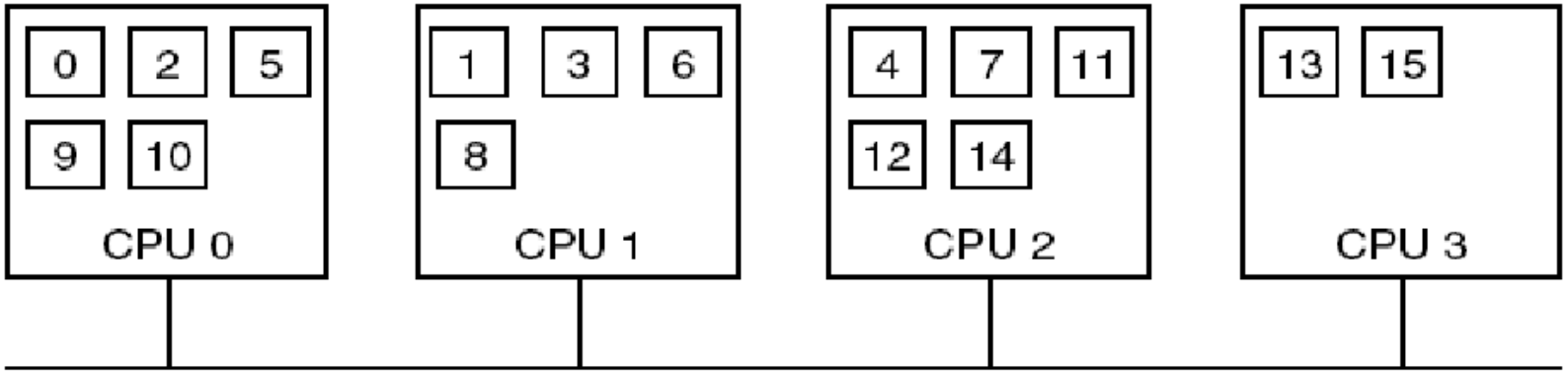
Σελιδοποίηση Διαμοιραζόμενης μνήμης (1/3)



(a) Pages of the address space distributed among four machines.

- When a CPU references an address that is not local, a trap occurs, and the DSM software fetches the page containing the address and restarts the faulting instruction.

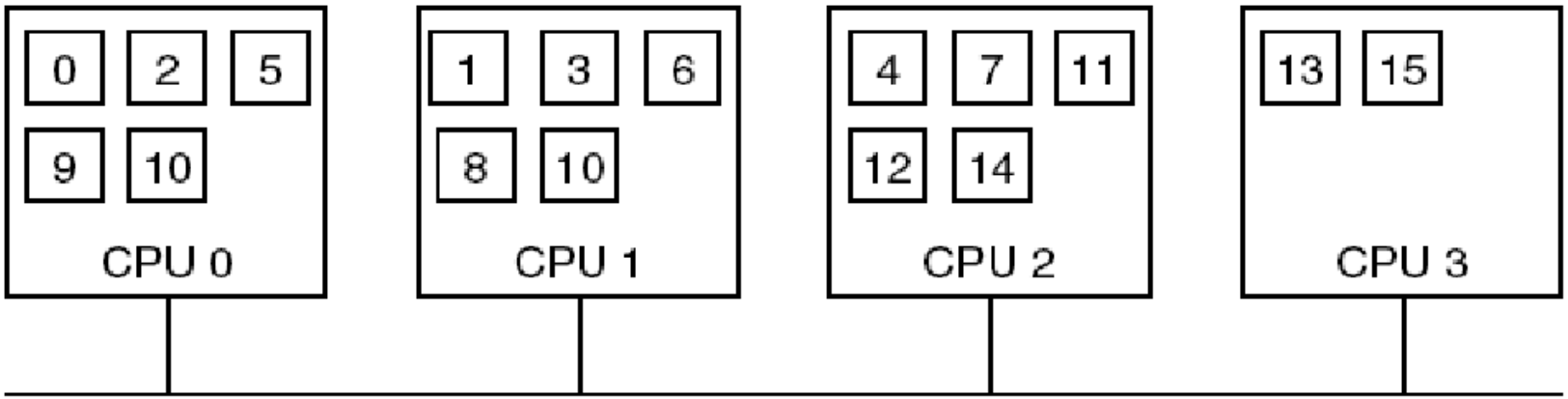
Σελιδοποίηση
Διαμοιραζόμενης μνήμης (2/3)



(b)

(b) Situation after CPU 0 references page 10 and the page is moved there.

Σελιδοποίηση
Διαμοιραζόμενης μνήμης (3/3)



(c)

(c) Situation if page 10 is read only and replication is used.

Cache - Only Memory Architecture (COMA)

- Σε αυτή την κατηγορία η κοινή μνήμη είναι η κρυφή (cache).
- Κάθε επεξεργαστής έχει άμεση πρόσβαση σε ένα τμήμα της κοινής μνήμης πιο αργή πρόσβαση στις απομακρυσμένες κρυφές μνήμες.
- Υπάρχει μια επιπλέον cache μνήμη (D) που συμβάλει την απομακρυσμένη πρόσβαση στις κρυφές.
- Υπάρχει ένας κοινός χώρος μνήμης (cache shared address).

Στοιχεία Παράλληλου Προγραμματισμού

Εύρεση παραλληλίας

- Σε προηγούμενη διάλεξη έγινε κατανοητό ότι πρέπει να βρεθεί ή να δημιουργηθεί η παραλληλία σε μια εφαρμογή προκειμένου να μπορεί να εκτελεστεί αποτελεσματικά σε ένα παράλληλο σύστημα.
- Πως βρίσκεται όμως η παραλληλία;

Που μπορεί να βρεθεί η παραλληλία σε μια εφαρμογή;

Αν είστε εξοικειωμένοι με την εφαρμογή, θα πρέπει να γνωρίζετε για το πού μπορούν να βρεθούν ανεξάρτητοι υπολογισμοί:

- Εμπλέκεται μεγάλο σύνολο δεδομένων? **(domain decomposition)**
- Υπάρχουν τμήματα κώδικα που μπορούν να εκτελεστούν σε διαφορετική σειρά? **(task decomposition)**
- Είναι ο υπολογισμός ένα σύνολο σταδίων που δεν αλληλεπιδρούν εκτός από την χρήση της εξόδου του ενός ως εισόδου για την επόμενη? **(pipeline decomposition)**

Μια πιο επίσημη μέθοδος της ανακάλυψης του παραλληλισμού χρησιμοποιεί γραφήματα εξάρτησης.

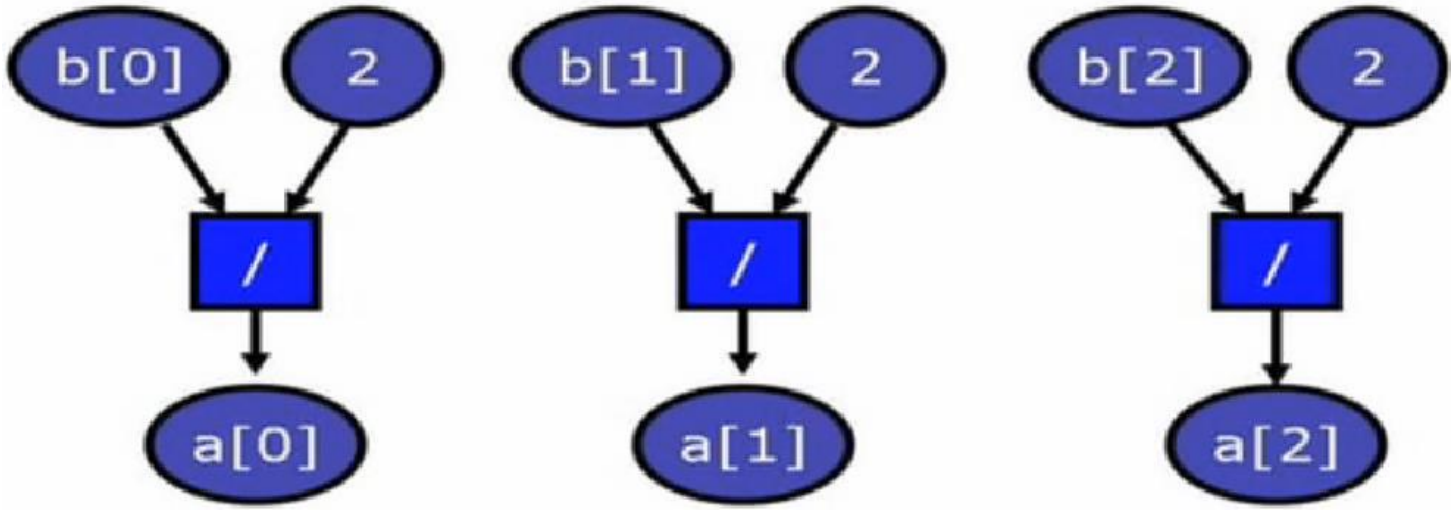
Τι είναι ένα γράφημα εξαρτήσεων;

- Γράφημα = (κόμβοι, βέλη).
- Για κάθε κόμβο:
- Εκχώρηση μεταβλητής (εκτός από τις μεταβλητές του δείκτη).
- Σταθερά.
- Χερισμός ή λειτουργία κλήσης.
- Τα βέλη δείχνουν τη χρήση των μεταβλητών και των σταθερών:
- Δεδομένα ροής.
- Έλεγχος ροής.

- **Τα γραφήματα μας επιτρέπουν την οπτική αναπαράσταση των εξαρτήσεων, καθοδηγώντας μας στις αποφάσεις παραλληλοποίησης.**

Παράδειγμα γράφου εξάρτησης 1

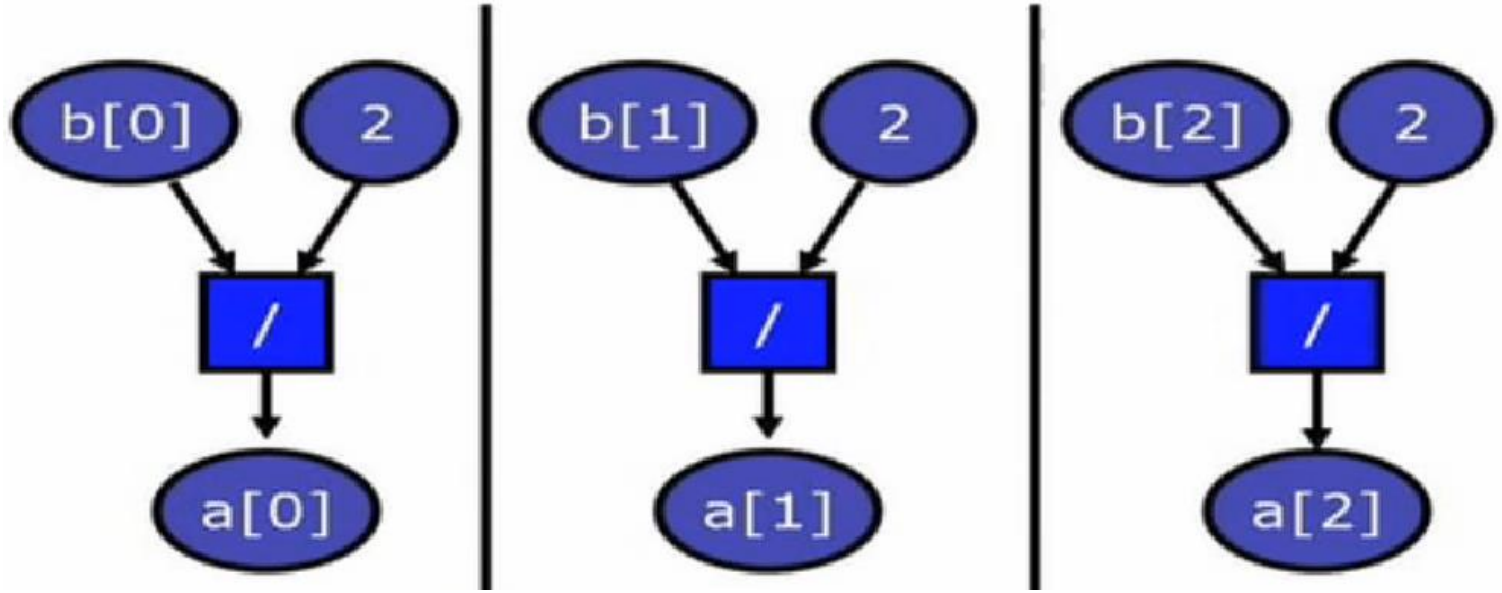
```
for ( i = 0; i < 3; i++)
    a[i] = b[i] / 2.0;
```



Εύκολα φαίνεται ότι μπορεί να παραλληλοποιηθεί με domain decomposition

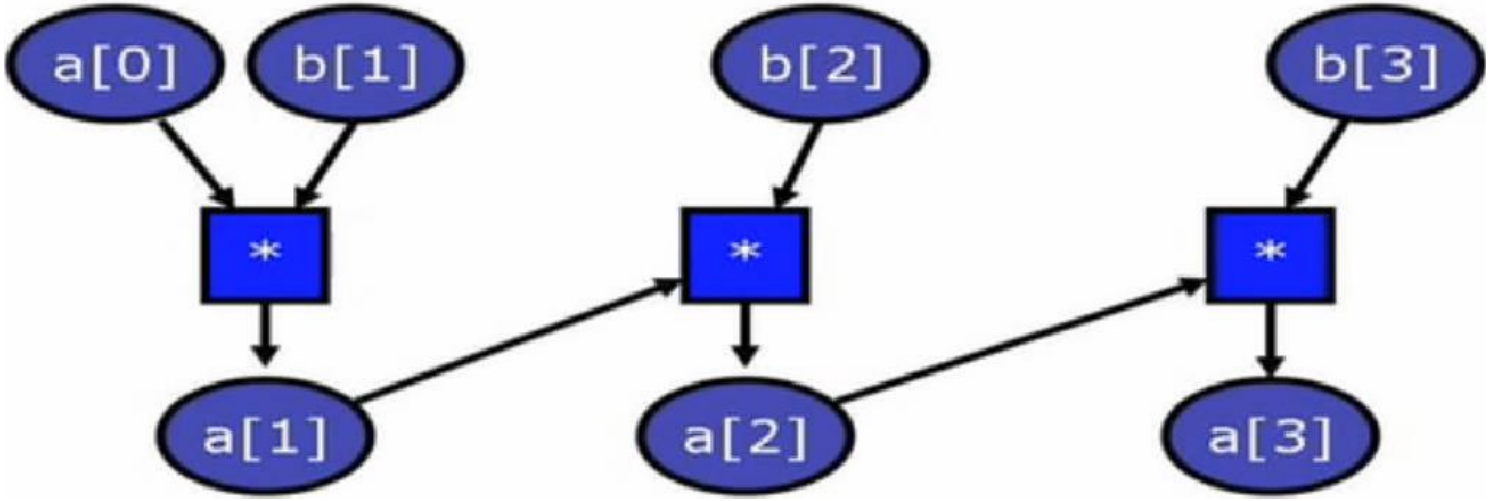
- For (i = 0; i < 3; i++)
- a[i] = b[i] / 2.0;

Είναι δυνατόν με ***Domain decomposition***



Παράδειγμα γράφου εξάρτησης 2

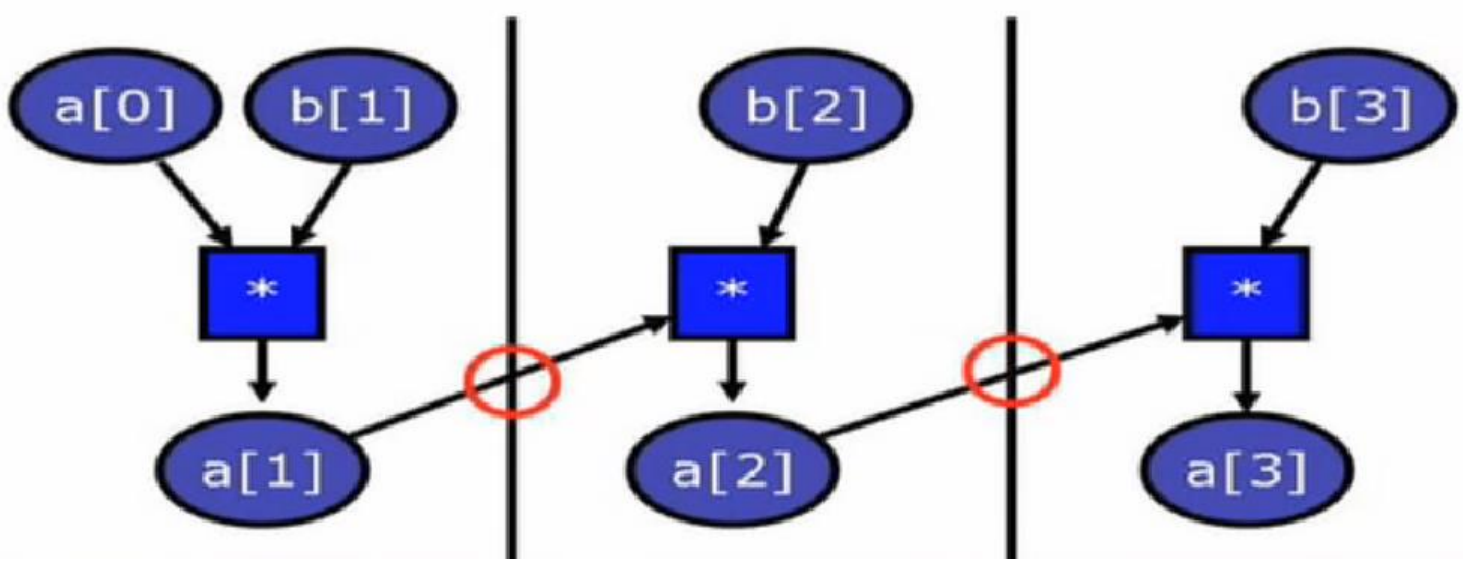
```
for (i = 1; i < 4; i++)
    a[i] = a[i – 1] * b[i];
```



Δε μπορεί να γίνει παραλληλοποίηση με domain decomposition

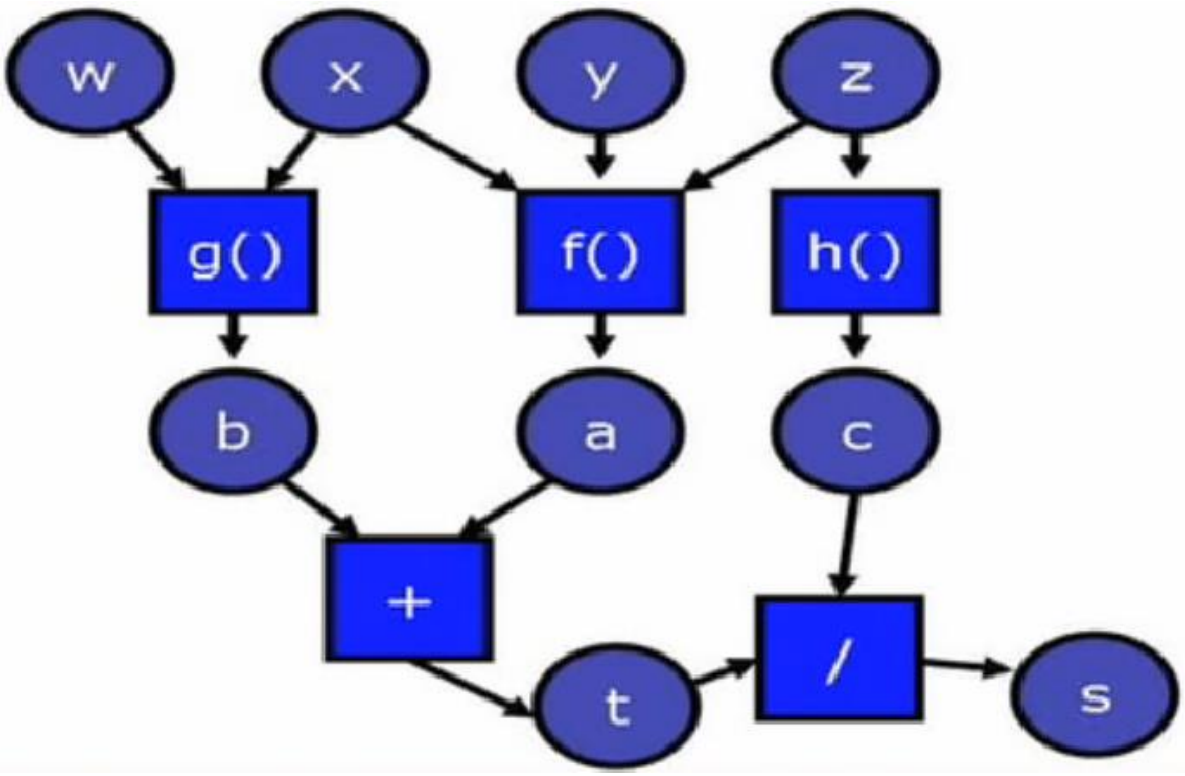
```
for (i = 1; i < 4; i++)
    a[i] = a[i –1] * b[i];
```

No domain decomposition



Παράδειγμα γράφου εξάρτησης 3

```
a = f(x, y, z);
b = g(w, x);
t = a + b;
c = h(z);
s = t / c;
```



Μπορεί να γίνει παραλληλοποίηση μέσω tasks

a = f(x, y, z);

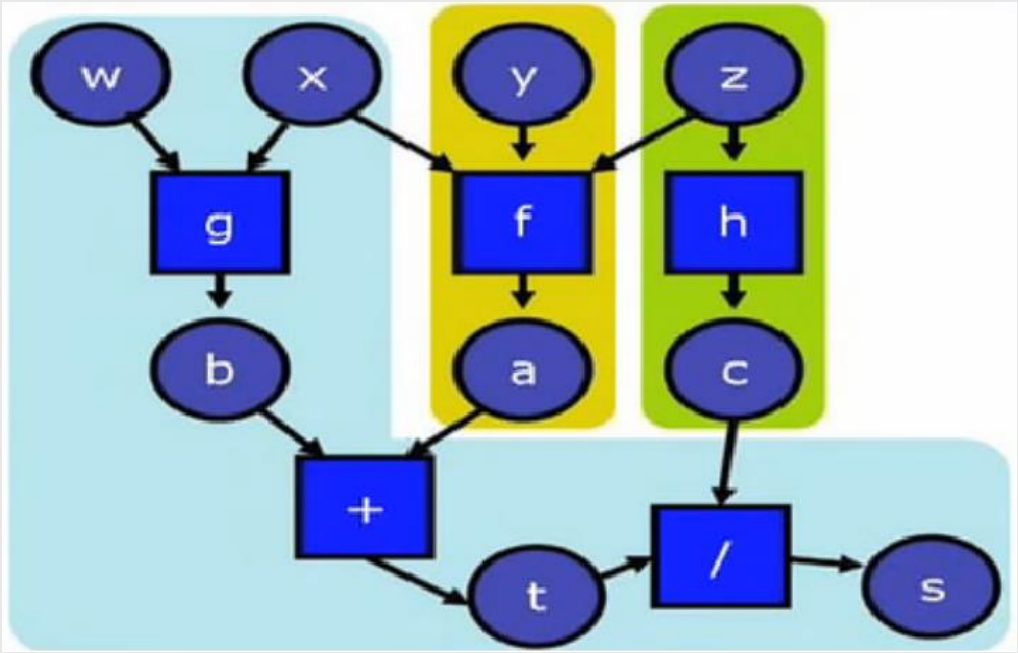
b = g(w, x);

t = a + b;

c = h(z);

s = t / c;

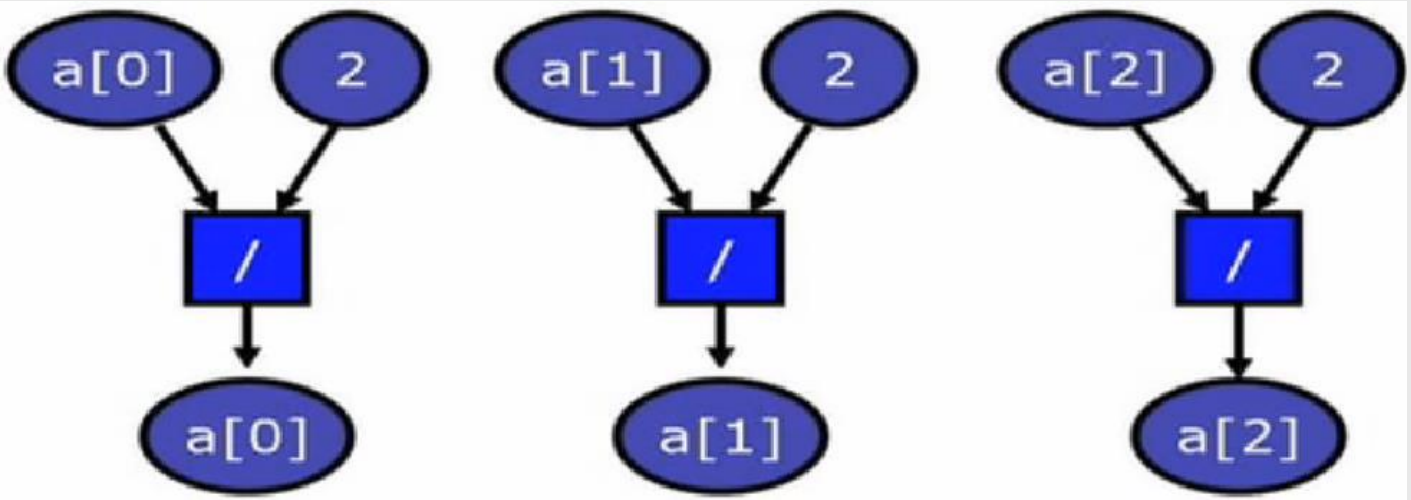
Task Decomposition με 3 πυρήνες.



Όμως, απαιτείται συγχρονισμός.

Παράδειγμα γράφου εξάρτησης 4

```
for  ( i = 0;  i < 3;  i++)
    a[i]  =  a[i]  /  2.0;
```

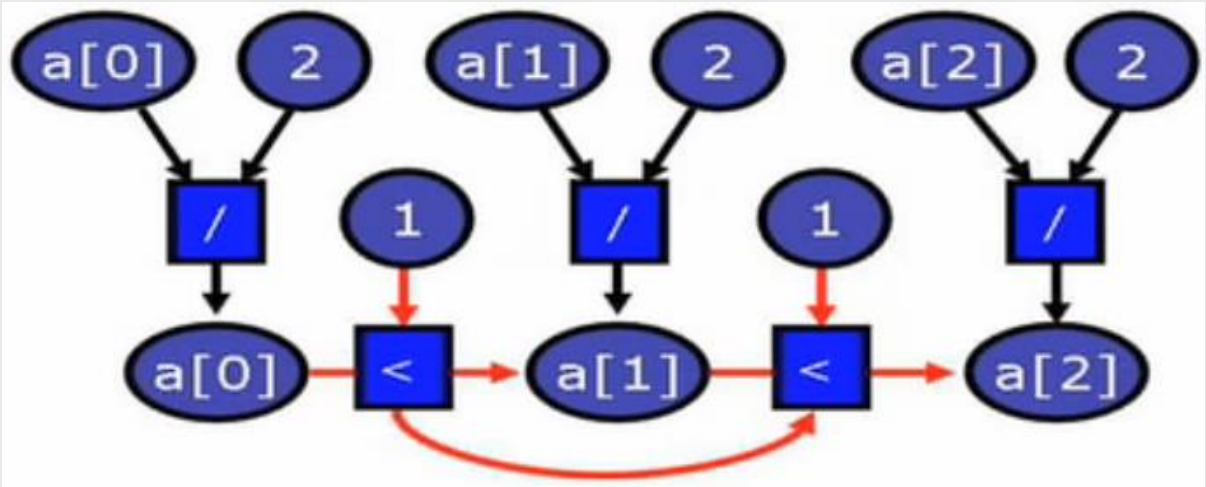


Μπορεί να χρησιμοποιηθεί domain decomposition.

Αν όμως έχουμε λίγο διαφορετικό κώδικα.

Παράδειγμα γράφου εξάρτησης 5

```
for ( i = 0; i < 3; i++) {
   a[i] = a[i]  /  2.0;
   if (a[i] < 1.0) break;
}
```



- Η εκτέλεση συνεχίζεται στην επόμενη επανάληψη, μόνο όταν δεν ισχύει η συνθήκη.
- **Υπάρχουν συνθήκες ή εξαρτήσεις ελέγχου στον κώδικα.**
- **Δε μπορούμε να τραβήξουμε κατακόρυφες γραμμές.**

Εύκολη/Δύσκολη παραλληλοποίηση

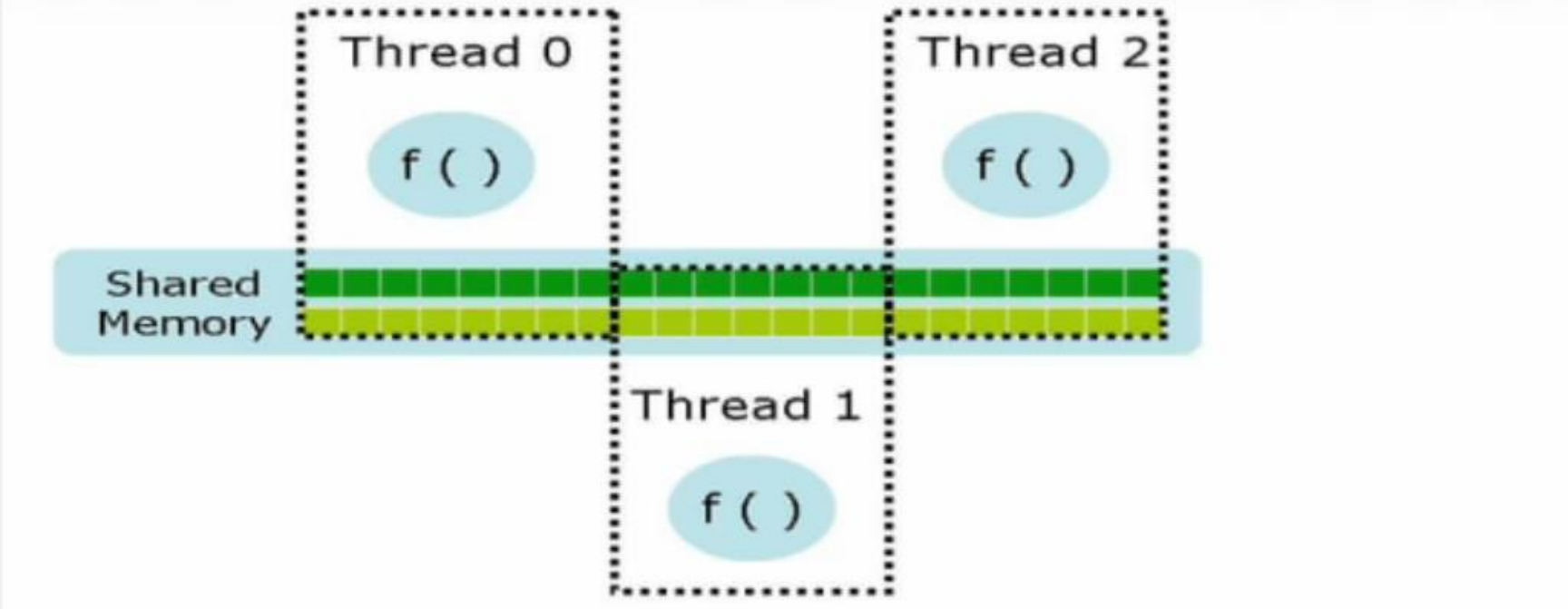
Ευκολότερη Λύση Παραλληλοποίησης	Πιο Δύσκολη ή Ακόμα και Αδύνατη Λύση Παραλληλοποίησης
Μεγαλύτερα σύνολα δεδομένων	Μικρότερα σύνολα δεδομένων
Πυκνές μήτρες	Αραιές μήτρες
Διαιρώντας το διάστημα μεταξύ των πυρήνων	Διαιρώντας τον χρόνο μεταξύ των πυρήνων

Η μεθοδολογία παραλληλοποίησης

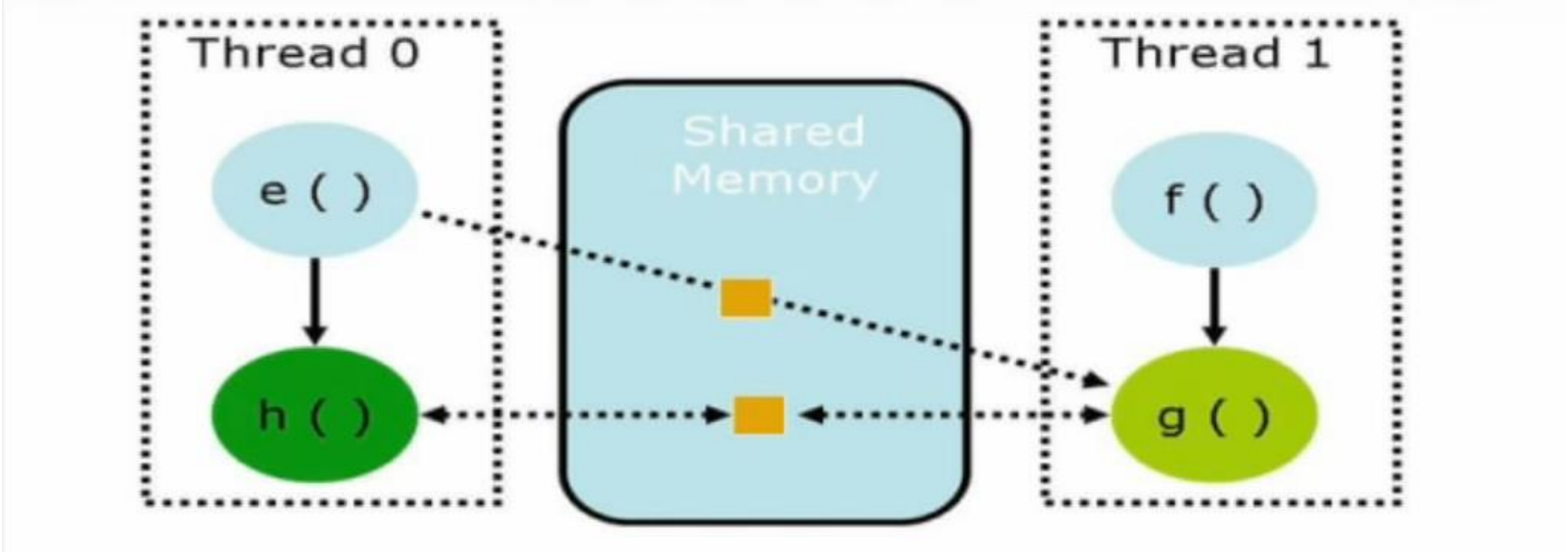
- Μελέτη προβλήματος, ακολουθιακό πρόγραμμα, ή τμήμα κώδικα.
- Αναζητήστε ευκαιρίες για παραλληλισμό.
- Χρησιμοποιήστε νήματα για να εκφράσετε τον παραλληλισμό.

- **Τα νήματα είναι τμήματα των διεργασιών...**

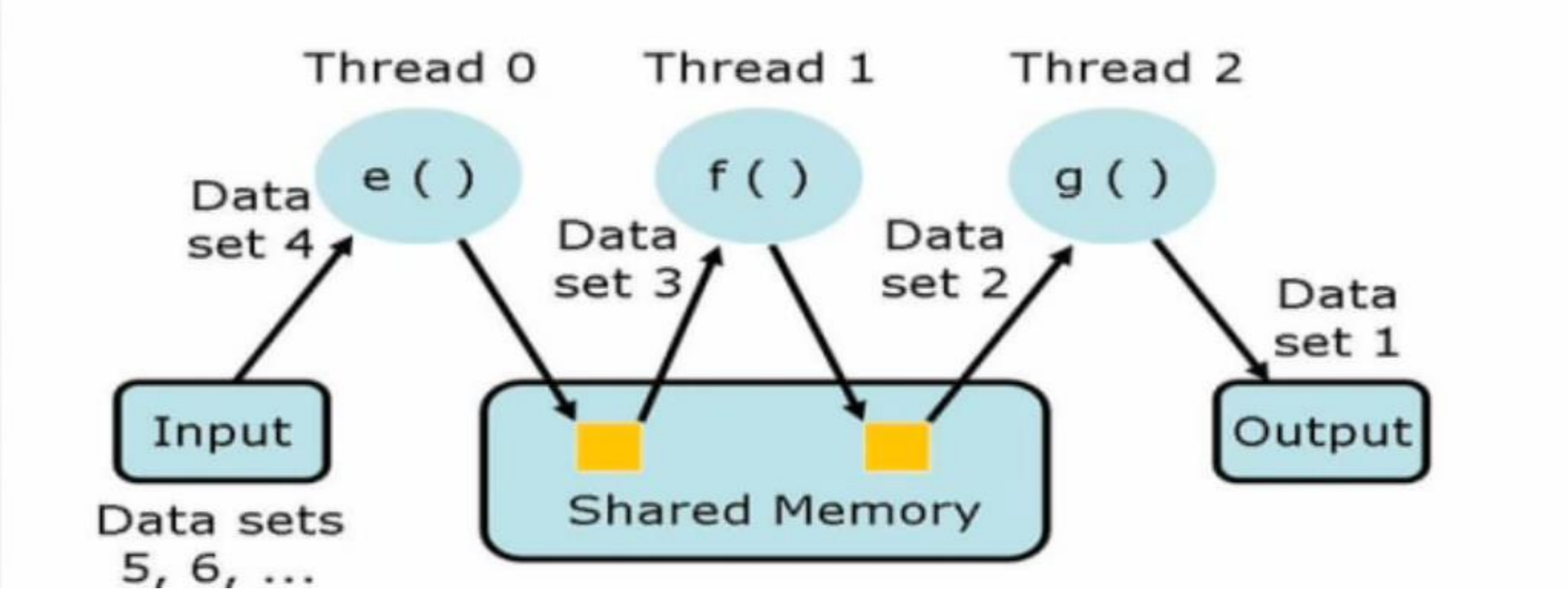
Domain Decomposition με νήματα



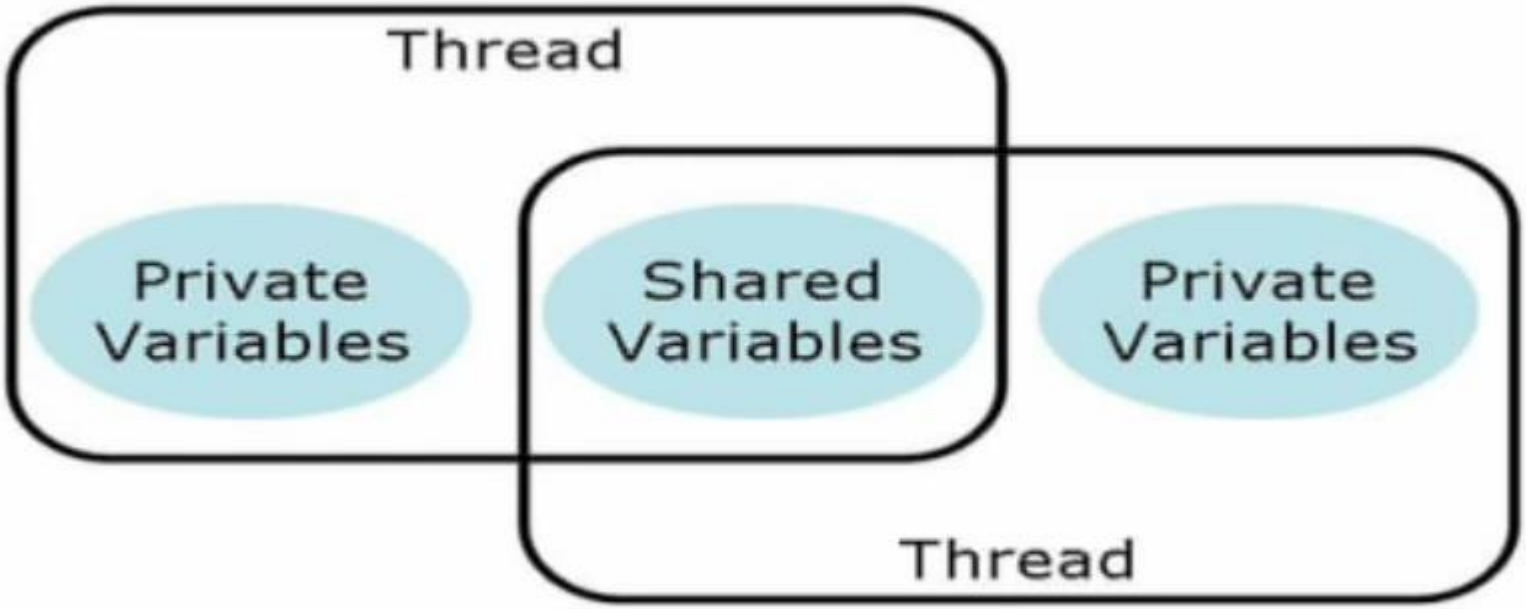
Task decomposition με νήματα



Pipeline και νήματα



Τα νήματα έχουν ιδιωτικές και κοινόχρηστες μεταβλητές



Παράδειγμα domain decomposition με νήματα (1/2)

- Ακολουθιακός Κώδικας:

```
int a[1000] , i;
for ( i = 0; i < 1000; i++) a[i] = foo(i);
```

Παράδειγμα domain decomposition με νήματα (2/2)

- Ακολουθιακός Κώδικας:

```
int a[1000] , i;
for ( i = 0; i < 1000; i++)  a[i] = foo(i);
```

- Νήμα 0:

```
for ( i = 0; i < 500; i++)  a[i] = foo(i);
```

- Νήμα 1:

```
for ( i = 0; i < 1000; i++) a[i] = foo(i);
```

- **Το a[ ] πρέπει να είναι κοινόχρηστη μεταβλητή.**
- **Το i πρέπει να είναι ιδιωτική μεταβλητή.**

Παράδειγμα task decomposition με νήματα (1/2)

```
int e;
main  ( )   {
    int x[10], j, k, m; j = f(x,k); m = g(x,
k);...
}
int  f (int  *x,  int  k)
{
     int  a; a = e * x[k] * x[k]; return  a;
}
int g(int *x,int k)
{
     int a; k = k - 1; a = e /x[k]; return a;
}
```

Παράδειγμα task decomposition με νήματα (2/2)

```
int e;
main ( ) {
   int x[10], j, k, m;  j = f (x, k); m = g (x, k)....
}

int f (int *x, int k)          Νήμα 0
{
   int a;   a = e * x[k] * x[k];  return a;
}

int g (int *x, int k)          Νήμα 1
{
   int a;   k = k – 1;   a = e / x[k];   return a;
}
```

- **Τα e, x[ ] πρέπει να είναι κοινόχρηστες μεταβλητές.**
- **Τα k, a πρέπει να είναι ιδιωτικές μεταβλητές.**
- **Ενδεχομένως, το e μπορεί να είναι και static (καλύτερη απόδοση).**

Σύνοψη: Ιδιωτικές και κοινόχρηστες μεταβλητές

- **Κοινόχρηστες μεταβλητές:**
  - Στατικές μεταβλητές.
  - Μεταβλητές σωρού.
  - Περιεχόμενα του χρόνου εκτέλεσης της στοίβας κατά τη στιγμή της κλήσης.
- **Ιδιωτικές μεταβλητές:**
  - Μεταβλητές δείκτη βρόχου.
  - Χρόνος εκτέλεσης των λειτουργιών της στοίβας που επικαλείται από το νήμα.

- Χρησιμοποιήθηκε υλικό από “Intro to Parallel Programming” της Intel.

Τέλος Ενότητας