Υ07 – Παράλληλα Συστήματα
2011-12

5/3/2012

Εισαγωγή στα Παράλληλα Συστήματα (Οργάνωση-Προγραμματισμός)

Β. Δημακόπουλος

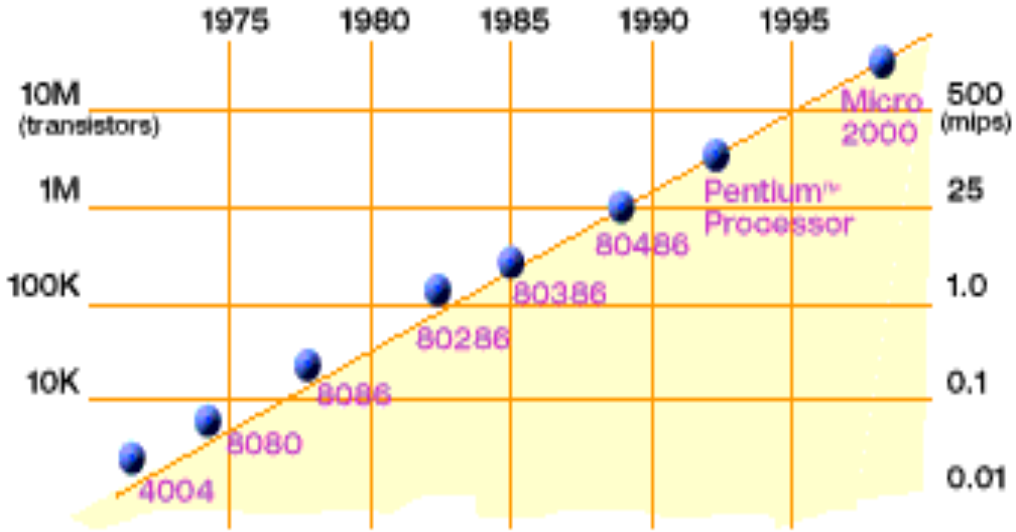
Α. Ευθυμίου

Τι περιλαμβάνει το ***σημερινό*** μάθημα;

- Εισαγωγή στα παράλληλα συστήματα
  - Τι είναι;
  - Πώς φτάσαμε ως εδώ;
  - Τι σημαίνει παράλληλο πρόγραμμα;
  - Τι είναι η κοινή και η κατανεμημένη μνήμη;

- Οργανωτικά / διαδικαστικά του μαθήματος
  - Πώς είναι οργανωμένο (π.χ. βαθμολόγηση) το μάθημα;
  - Τι ύλη καλύπτει το μάθημα;
  - Υπάρχει κάποιος λόγος να το πάρω (ή να μην το πάρω) αυτό το μάθημα;

Τεχνολογία – ο νόμος του Moore



2X transistors/Chip Every 1.5 years

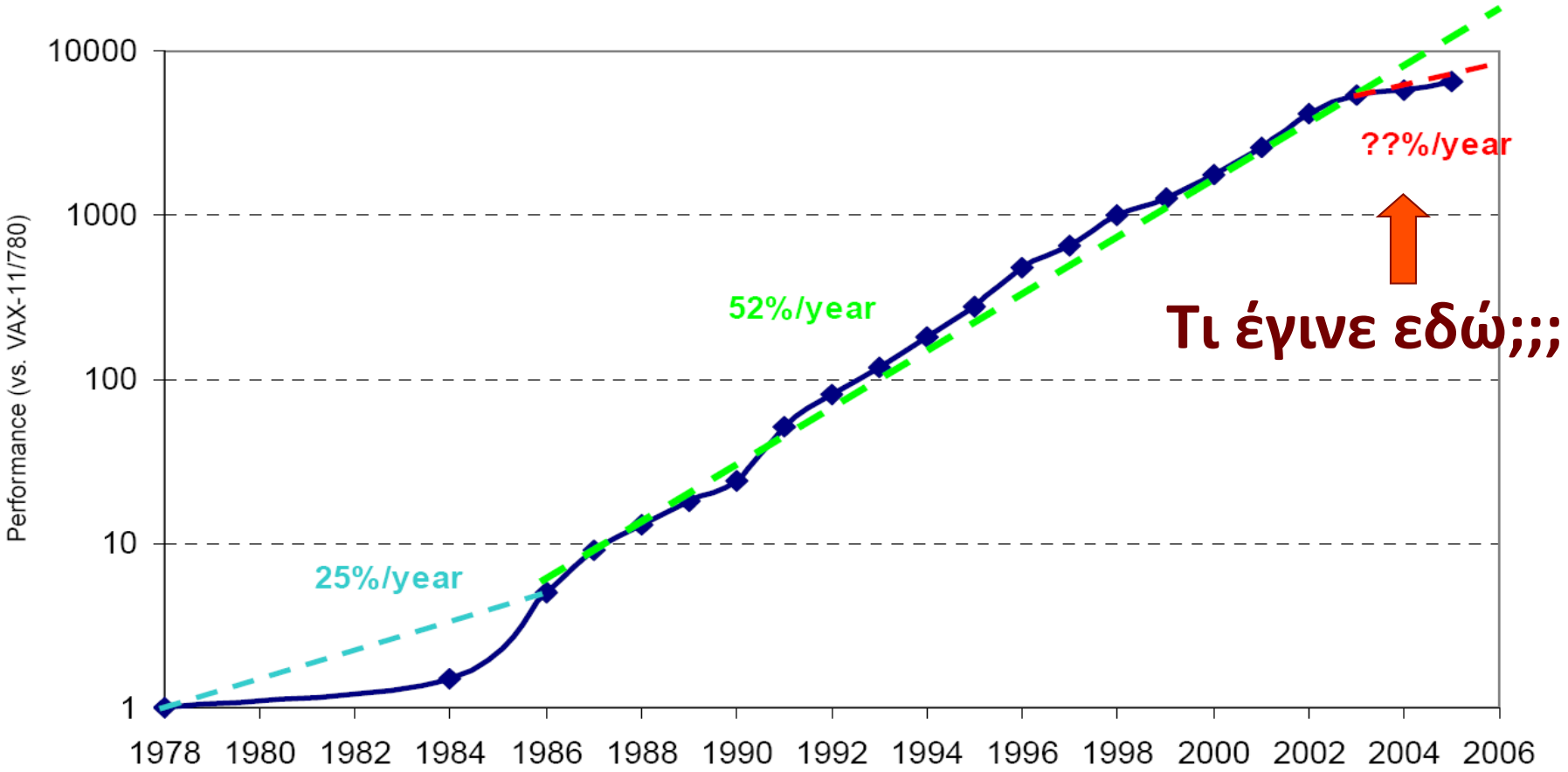
Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

Τι σημαίνει αυτό;

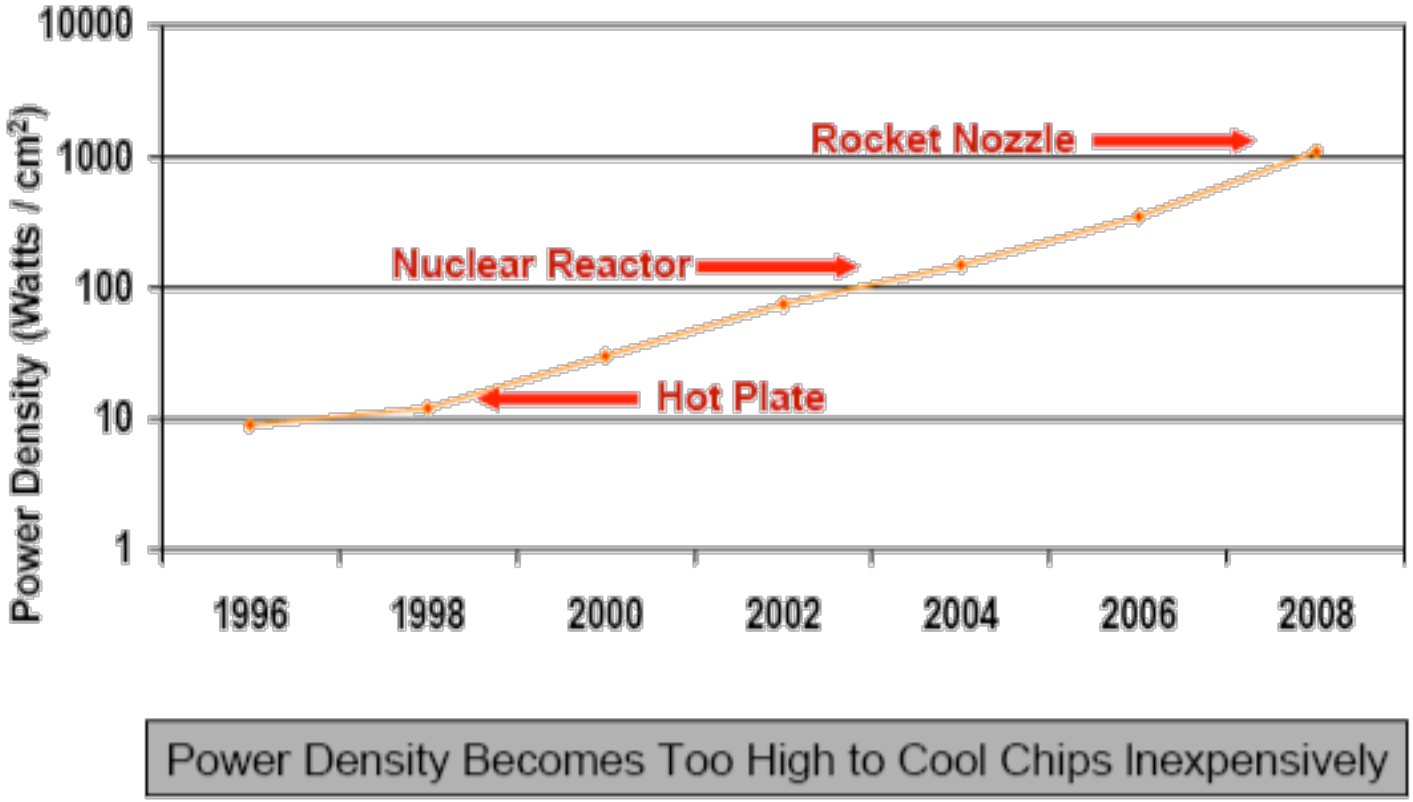
Απάντηση: πολυπλοκότητα!

Ο νόμος του Bill **Joy** τις επιδόσεις



- Based on integer SPEC benchmarks
- Επιδόσεις εκθετικά αυξανόμενες

Πυκνότητα ισχύος



Power Density Becomes Too High to Cool Chips Inexpensively

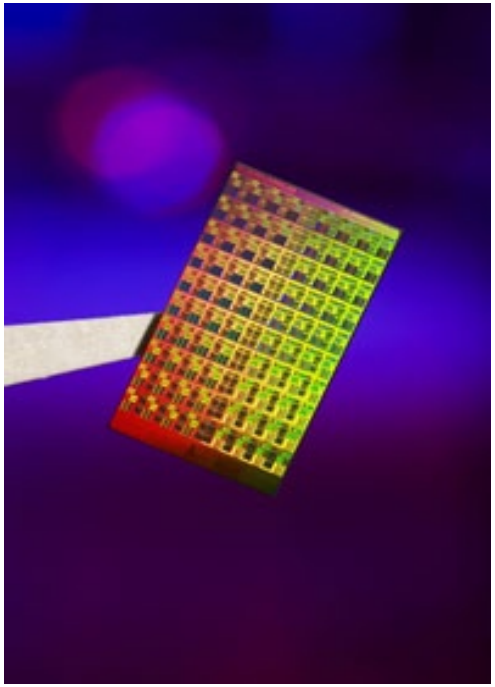
Πρόσφατο παρελθόν

- SMPs (symmetric multiprocessors)
  - 2 – 4 επεξεργαστές (μονοπύρηνοι) συνηθισμένοι, μέχρι 8 επεξεργαστές σε εμπορικά συστήματα
  - Πανάκριβα συστήματα με 12 - 16 επεξεργαστές ελάχιστα
  - Κοινή μνήμη

- Τμήμα Πληροφορικής:
  - Πολλά Sun με 2 UltraSparc
  - Πολλά PC με 2 Pentium / Athlon
  - **atlantis** με 4 Pentium III Xeon
    - 700.000 δρχ/CPU !
  - **paraguay** με 4 x Intel Xeon 7000 Paxville @ 3GHz
    - 2 cores per CPU / 2 threads per core (hyperthreaded)
  - **paragon** με 2 x AMD Opteron 6128 @ 2GHz
    - 8 cores per CPU

Dual core / Quad core / Multicore / Manycore ?

- Πλέον και τα φτηνότερα PC έχουν 1 επεξεργαστή τουλάχιστον διπύρηνο
- Διπύρηνοι αρχικά, τώρα 4/6/8/12πύρηνοι, (Intel, AMD)
  - T1 (Sun Niagara): 8πύρηνοι (με 4-way multithreaded πυρήνες) από τον 12/2005!
  - T3 (2010): 16πύρηνοι, 8-way multithreaded
  - Πολλαπλών πυρήνων (multicore) γενικά …
- **Manycore** (πολλών πυρήνων)??
  - Μιλάμε για *πολλούς πυρήνες*
  - Τριψήφιο νούμερο (> 64)
- Πότε?
  - Τώρα!
    - Intel 80-πύρηνο πρωτότυπο είναι 5 ετών!
    - Φεβ. 2007
    - 80 απλά cores
    - 2 floating point engines /core
    - "network-on-a-chip" τύπου πλέγματος
    - 100 million transistors, >1 Teraflop peak!



Clusters

- Παντού κι αυτά!
  - Συλλογή από διασυνδεδεμένους «κόμβους»
    - Φτηνοί / ευρέως διαθέσιμοι επεξεργαστές (π.χ. Clusters από PCs)
  - Ο μόνος τρόπος να φτιάξουμε «οικονομικούς» υπερ-υπολογιστές (πολλά Teraflops)
  - Sandia Laboratories Red Storm (Cray, 2004)
    - 13000 AMD Opterons (basically PC nodes), 75 Terabytes of memory
    - > 100 Teraflops (peak)
    - Linux
    - Κόστος: $90.000.000
  - Σε αντιδιαστολή με τον Earth Simulator (NEC, Ιαπωνία, 2002)
    - Διανυσματικός υπερυπολογιστής
    - 5120 διανυσματικοί επεξεργαστές, 10 Terabytes of memory
    - 35 Teraflops (sustained)
    - Κόστος: $400.000.000 (2002, most expensive computer ever built)

Clusters

- Πανεπιστήμιο Ιωαννίνων
  - Κέντρο προσομοιώσεων: 200 κόμβοι (κάθε κόμβος pc με 2 επεξεργαστές)
  - Τμήμα Πληροφορικής: 16 κόμβοι, κάθε κόμβος 2 CPUS, κάθε CPU διπύρηνη
  - Και τα δύο με gigabit ethernet

- Βελτιωμένες επιδόσεις με δίκτυα χαμηλής καθυστέρησης
  - Π.χ. Myrinet
  - Πολύ ακριβότερα όμως
  - Κάρτα δικτύο gigabit: 10-20 ευρώ
  - Cluster τμήματος Υλικών (Myrinet, πριν 6 χρόνια περίπου)
    - Περίπου 100.000 ευρώ για τα PC
    - … και άλλες 100.000 ευρώ για τις κάρτες δικτύου τους!

- Το μέλλον:
  - Clusters από πολυπύρηνους κόμβους

GPUs, GPGPUs, Cells, κλπ.

- Πάρα πολλά και πολύ απλά επεξεργαστικά στοιχεία, κατάλληλα είτε για συγκεκριμένου τύπου υπολογισμούς (GPUs) είτε και για γενικότερους υπολογισμούς (GPGPUs, Cell).
- Πολύ «της μόδας»
- Πολύ γρήγορα
- «Ιδιαίτερος» προγραμματισμός

- Με μία λέξη:
  - ***Ετερογένεια***
  - Κλασικός ισχυρός πυρήνας/πυρήνες + «ειδικοί» (γρήγοροι, πολύ αλλά απλοί/ανίσχυροι) πυρήνες
  - Ετερογένεια και στον τρόπο προγραμματισμού

3 απλές ερωτήσεις

- Το excel θα τρέξει γρηγορότερα σε αυτά τα μηχανήματα;
  - Οχι!
- Αν είχα ένα από αυτά τα μηχανήματα σπίτι μου (ως PC), θα έβλεπα μεγαλύτερη ταχύτητα;
  - Ναι, κάποια (μικρή σχετικά) βελτίωση στην ταχύτητα θα υπήρχε
  - Γιατί όμως;

- "ManyCore" refers to many processors/chip
  - 64? 128? Hard to say exact boundary
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels

- Πρέπει η *εφαρμογή να έχει προγραμματιστεί παράλληλα ώστε να χρησιμοποιεί τους πολλαπλούς επεξεργαστες*

Σύντομη εισαγωγή στην κοινή μνήμη

Short intro to shared memory

«Οντότητες» εκτέλεσης κώδικα

- Σειριακό πρόγραμμα για υπολογισμό του π = 3.141592…

```
#define N 512
float pi = 0.0, W = 1.0/N;

main()
{
  int i;
  for (i = 0; i < N; i++)
    pi += 4*W / (1 + (i+0.5)*(i+0.5)*W*W);
  printf("π = %f\n", pi);
}
```

- Οταν φορτωθεί και εκτελείται το πρόγραμμα γίνεται **διεργασία (process)**.
- Κάθε διεργασία εκτελείται σε έναν επεξεργαστή
  - Το λειτουργικό σύστημα φροντίζει για αυτό

Διεργασίες & νήματα

- Μία διεργασία αποτελείται (κατ' ελάχιστο) από δεδομένα, κώδικα (εντολές), μία στοίβα (stack) και έναν μετρητή προγράμματος (program counter)
  - Ο μετρητής προγράμματος (PC):
    - δείχνει στην επόμενη εντολή του κώδικα που θα εκτελεστεί
  - Η στοίβα χρειάζεται για:
    - αποθήκευση των τοπικών μεταβλητών (τα «δεδομένα» της διεργασίας που είπαμε παραπάνω είναι οι global μεταβλητές)

```
#define N 512
float pi = 0.0, W = 1.0/N;

main()
{
  int i;
  for (i = 0; i < N; i++)
    pi += 4*W / (1 + (i+0.5)*(i+0.5)
*W*W);
  printf("π = %f\n", pi);
}
```

- Ο συνδυασμός PC + στοίβα λέγεται **νήμα εκτέλεσης (thread)**
  - Δηλαδή η διεργασία αποτελείται από (global) δεδομένα, από κώδικα και από ένα νήμα εκτέλεσης που περνάει από τις εντολές του κώδικα
  - Το νήμα είναι αυτό που εκτελείται όταν λέμε ότι εκτελείται η διεργασία!
  - Το νήμα αυτό λέγεται *αρχικό νήμα* της διεργασίας

Διεργασίες & νήματα

- Μία διεργασία μπορεί να δημιουργήσει κι άλλα πολλά νήματα εκτέλεσης:
  - Δημιουργώντας τίποτε άλλο εκτός από πολλές στοίβες και PCs
  - Ολα θα τρέχουν εντολές από τον *ίδιο* κώδικα (δεν δημιουργείται άλλο αντίγραφό του) και τέλος
  - Ολα θα έχουν τα *ίδια* global δεδομένα! Δηλαδή οι καθολικές μεταβλητές της διεργασίας είναι ΑΥΤΟΜΑΤΩΣ ΚΟΙΝΕΣ μεταξύ των νημάτων της.
  - Στην κάθε στοίβα το κάθε νήμα θα έχει τις δικές του τοπικές μεταβλητές
  - Κάθε νήμα εκτελείται σε έναν επεξεργαστή
    - Το λειτουργικό σύστημα φροντίζει για αυτό (εκτός από μερικές περιπτώσεις)

Παράλληλα προγράμματα

- Επομένως, για να εκμεταλλευτούμε πολλαπλούς επεξεργαστές, έχουμε δύο βασικές τεχνικές:
  - **Πολλαπλές διεργασίες**
    - Η διεργασία μας «γεννά» κι άλλες διεργασίες και κάθε μία εκτελείται στον δικό της επεξεργαστή (π.χ. fork())
  - **Πολλαπλά νήματα σε μία διεργασία**
    - Το αρχικό νήμα εκτέλεσης «γεννά» κι άλλα νήματα και κάθε ένα εκτελείται στον δικό του επεξεργαστή

- Υπάρχουν διαφορές σε ταχύτητα δημιουργίας, απαιτήσεις σε πόρους (π.χ. μνήμη) κλπ αλλά η πιο σημαντική διαφορά είναι ότι:
  - Ανάμεσα στις πολλαπλές διεργασίες *ΔΕΝ ΥΠΑΡΧΕΙ καμία κοινή μεταβλητή*. Πρέπει να δημιουργηθούν «με το χέρι», ενώ μεταξύ των πολλαπλών νημάτων όλες οι global μεταβλητές είναι κοινές είτε το θέλουμε είτε όχι.

Υλικό που θα εκτελέσει το παράλληλο πρόγραμμα

- Κάθε οντότητα εκτέλεσης (νήμα ή διεργασία) θα εκτελείται σε έναν επεξεργαστή, όπου «επεξεργαστής» είναι ότι το λειτουργικό σύστημα θεωρεί επεξεργαστή!

  - Πριν 3-4 χρόνια, ήταν μια CPU (μονοπύρηνη)
  - Πλέον είναι 1 πυρήνας από την CPU (γενικά, ένα αυτόνομο επεξεργαστικό στοιχείο)
  - Σε πολυνηματικούς επεξεργαστές (π.χ. Intel hyperthreading), κάθε ένα από τα hardware threads παρουσιάζεται στο λειτουργικό σύστημα ότι είναι διαφορετικός «επεξεργαστής»
    - Οι πολυνηματικοί επεξεργαστές εκτελούν 1 διεργασία (ή νήμα) τη φορά αλλά όταν βρίσκουν ελεύθερο χρόνο μπορούν να εκτελέσουν για λίγο εντολές από άλλη διεργασία (ή νήμα). Εκμεταλλεύονται κενούς κύκλους, αλλά αν υποστηρίζουν N hardware threads, δεν έχουν προφανώς τις επιδόσεις N ανεξάρτητων cores.

Οργάνωση των επεξεργαστών - SMPs

- SMPs – Symmetric Multiprocessors
  - Bus-based με όλους τους επεξεργαστές επάνω σε έναν δίαυλο στον οποίο βρίσκεται και η κύρια μνήμη
  - Ολοι προσπελαύνουν την ίδια μνήμη (κοινόχρηστη)

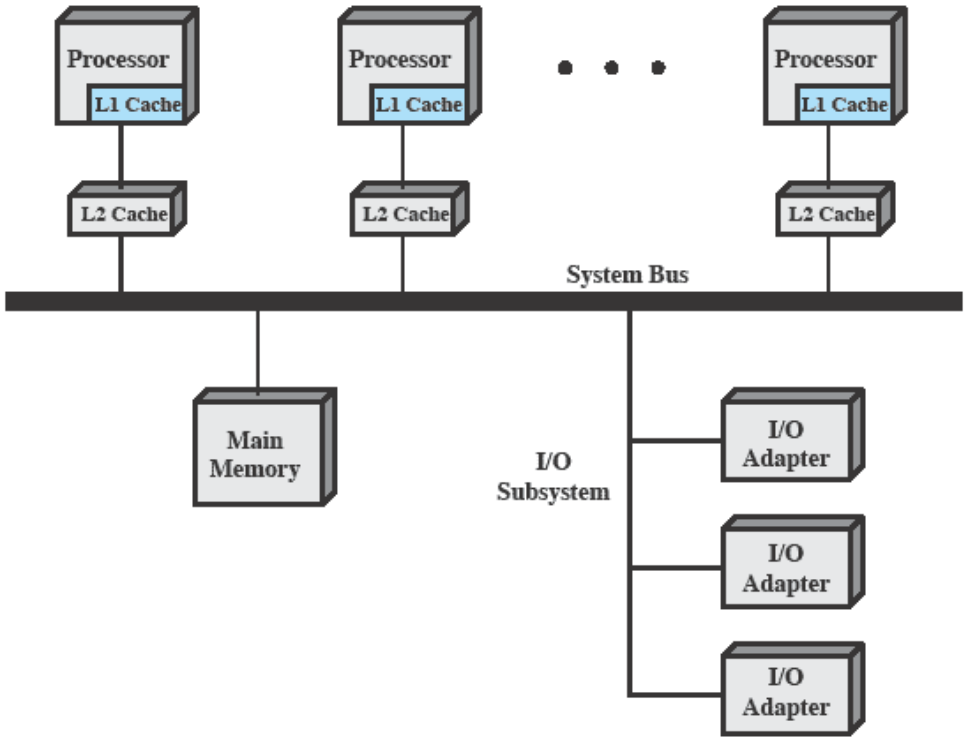


Figure 4.9 Symmetric Multiprocessor Organization

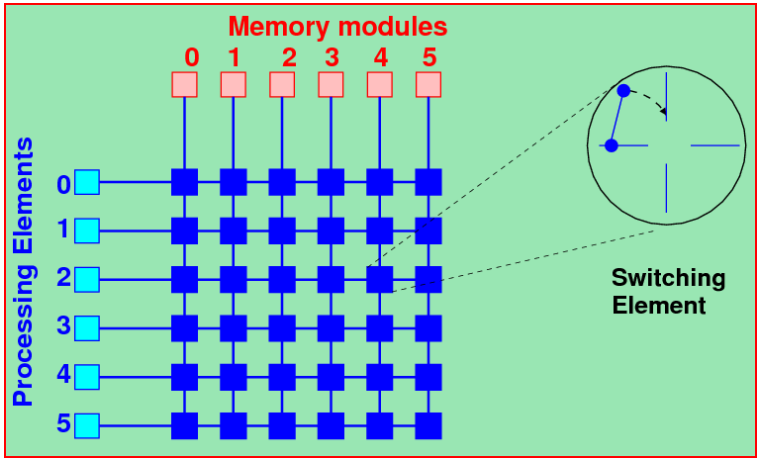
SMPs

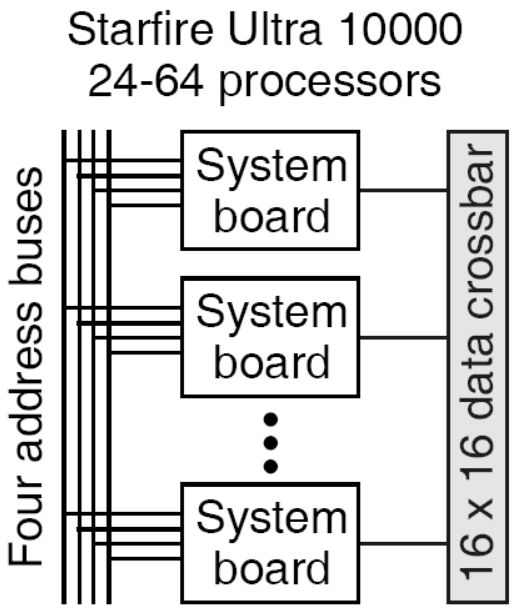
- Οικονομική λύση, επιτυχημένα εμπορικά
- Λίγοι επεξεργαστές (έως 10+, το πολύ)
- Ο δίαυλος είναι το bottleneck

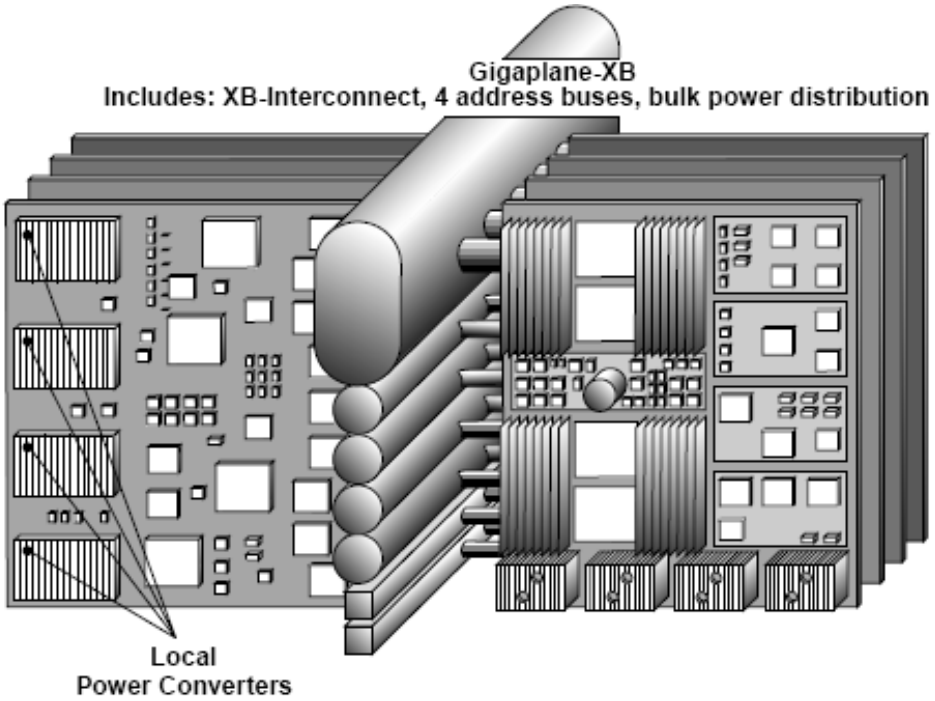
- Γενικά, οι επεξεργαστές όλοι βλέπουν και απευθύνονται στη κύρια μνήμη σαν να είναι ενιαία και εξ ολοκλήρου προσπελάσιμη
- **UMA** (= Uniform Memory Access)
  - Ολοι αντιλαμβάνονται την ίδια καθυστέρηση στην προσπέλαση οποιουδήποτε δεδομένου

Switch-based διασύνδεση

- Αντί για δίαυλος, διακοπτικό δίκτυο
  - **crossbar**
    (Sun Enterprise E10000, Earth Simulator)
  - Αλλα δίκτυα (π.χ. πεταλούδας, clos κλπ)
  - UMA
  - Καλύτερη κλιμακωσιμότητα, ακριβό

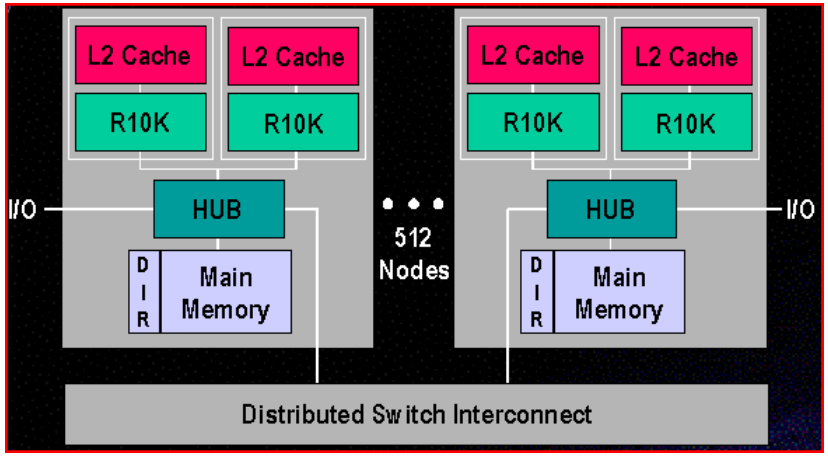






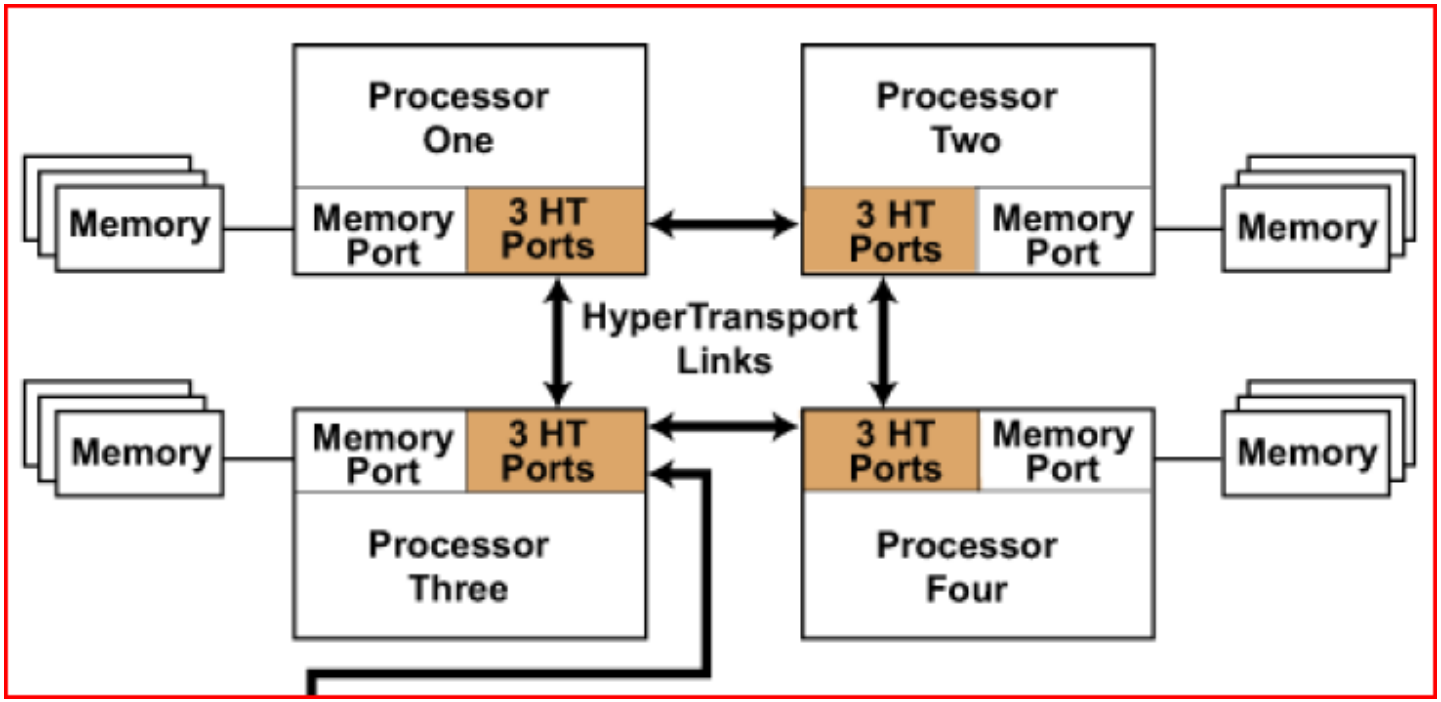
Ανομοιόμορφη προσπέλαση

- Για μεγάλη κλιμακωσιμότητα: *ιδιωτική μνήμη σε κάθε επεξεργαστή*
  - Θα πρέπει οι επεξεργαστές να συνδέονται μεταξύ τους για να μπορούν να προσπελάσουν ο ένας τη μνήμη του άλλου
- Πώς θα νομίσουν όλοι ότι η μνήμη είναι ενιαία, ενώ ο καθένας έχει μόνο ένα κομμάτι της;



  - Χρειάζεται ειδικός ελεγκτής ο οποίος ξεχωρίζει τις προσπελάσεις για τα τοπικά δεδομένα και τα απομακρυσμένα. Στη δεύτερη περίπτωση πάει και τα φέρνει από την μνήμη άλλου επεξεργαστή.
  - Επομένως, υπάρχει *ανομοιόμορφη προσπέλαση της μνήμης*
    - Ταχύτατα για δεδομένα που τυχαίνουν να είναι στην τοπική μνήμη
    - Με καθυστέρηση αν βρίσκονται στη μνήμη κάποιου άλλου
  - **NUMA** (= Non-Uniform Memory Access)

Παράδειγμα: AMD Opterons και hypertransport



- **NUMA factor**:
  - Πόσο (κατά μέσο όρο) πιο αργή είναι η προσπέλαση μνήμης άλλου επεξεργαστή απ' ότι της τοπικής μνήμης
  - Σε συστήματα με opterons είναι περίπου 1.3 – 1.5.

Opteron 8347HE (1,9GHz)

Access to...	Local node	Neighbor node	Opposite node
Read	83 ns	98 ns (x1.18)	117 ns (x1.41)
Write	142 ns	177 ns (x1.25)	208 ns (x1.46)

Τα multicore τι είναι;

- Μέχρι τώρα είναι παρόμοια με τα SMP μιας και είναι πολλαπλά cores που προσπελαύνουν μία κοινή μνήμη
- Αν και θυμίζουν SMPs, ανάλογα με το πώς διαμοιράζονται τις caches, πάνε όλο και πιο πολύ προς NUMA!
- Π.χ. Intel Xeon 5450 (quad core)
  - Ανά δύο οι πυρήνες έχουν κοινή cache δευτέρου επιπέδου
    - Αρα αν κάτι το πετύχουν εκεί, το παίρνουν πολύ γρήγορα
    - Αλλιώς πάνε στην κύρια μνήμη (που είναι πολύ αργή)
- Τα manycore θα είναι NUMA 99,999%
- «Βαθιές» ιεραρχίες cache
- Συνολική cache μικρότερη από SMPs (υπάρχει μοίρασμα ανάμεσα στα cores)
- Θα στοιχίζει πολύ το cache miss

Θέματα που έχουν επίπτωση στις επιδόσεις των προγραμμάτων

- SMPs & caches
  - Coherency (συνοχή)
  - False sharing

- NUMA & caches
  - ccNUMA & directory-based coherency (συνοχή με καταλόγους)
  - Πως το NUMA factor επηρεάζει τις επιδόσεις

Cache

- *Απαραίτητη* μιας και εξοικονομεί πολύ χρόνο
  - Πολύ γρηγορότερη από την κύρια μνήμη
  - Πολύ μικρότερη από την κύρια μνήμη
- Οτι δεδομένο χρησιμοποιήσει η διεργασία μας (δηλαδή ο επεξεργαστής),
  - Το φέρνει και το φυλάει
  - Φέρνει και τα γειτονικά του δεδομένα (φέρνει ένα ολόκληρο μπλοκ μνήμης)
- Εφόσον το πρόγραμμα έχει *τοπικότητα* (δηλαδή χρησιμοποιεί σχετικά λίγα και γειτονικά δεδομένα για αρκετή ώρα), έχουμε πολύ μεγάλη βελτίωση στην ταχύτητα προσπέλασης των δεδομένων.

Πώς να καταστρέψετε την cache

```
for (i = 0; i < 10000; i++)
  for (j = 0; j < 10000; j++)
    a[i][j] = 2*b[i][j];
```

Χρόνος εκτέλεσης στο φορητό μου:

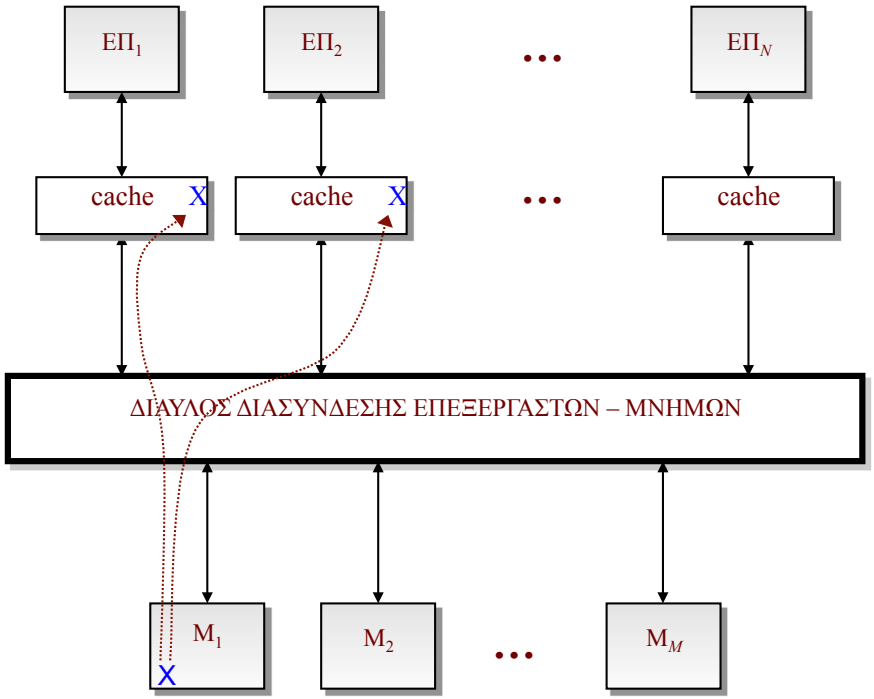
1.2 sec

```
for (j = 0; j < 10000; j++)
  for (i = 0; i < 10000; i++)
    a[i][j] = 2*b[i][j];
```

Χρόνος εκτέλεσης στο φορητό μου:

14 sec

Cache coherency



X = global μεταβλητή και δύο νήματα της διεργασίας εκτελούνται στους επεξεργαστές 1 και 2

Τι γίνεται αν το νήμα 1 αλλάξει το X (π.χ. κάνει X = X+1);

Cache coherency II

- Αυτό είναι το πρόβλημα της συνοχής της cache
  - Πρέπει όλες οι caches και η κύρια μνήμη να είναι ενημερωμένες με τις πιο πρόσφατες τιμές των δεομένων που έχουν
  - Το πρόβλημα στους SMPs λύνεται με ειδικά πρωτόκολλα στο hardware, π.χ. MESI (Intel), MOESI (AMD, Sparc). Τροποποιήσεις σε μία cache *ακυρώνουν* τυχόν αντίγραφο του δεδομένου σε μία άλλη
  - Οι τροποποιήσεις και ακυρώσεις δημιουργούν αρκετή κίνηση στο δίαυλο
    - Τα πρωτόκολλα λειτουργούν παρακολουθώντας ΤΑ ΠΑΝΤΑ ΠΟΥ ΣΥΜΒΑΙΝΟΥΝ ΣΤΟΝ ΔΙΑΥΛΟ, ΑΠΟ ΟΠΟΙΟΝ ΕΠΕΞΕΡΓΑΣΤΗ ΚΑΙ ΝΑ ΠΡΟΕΡΧΟΝΤΑΙ.
  - **ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΔΙΔΑΓΜΑ**: ***οι κοινές μεταβλητές που τροποποιούνται συχνά και από διαφορετικά νήματα προκαλούν υψηλή κίνηση στον δίαυλο και επιφέρουν μεγάλες καθυστερήσεις.***
  - Τα πρωτόκολλα προσπαθούν να μειώσουν την κίνηση στο ελάχιστο, όμως ένα απρόσεκτο πρόγραμμα μπορεί να καταστρέψει το όποιο όφελος.

NUMA & συνέπειες

- Ολοι οι επεξεργαστές έχουν caches. Τι γίνεται αν κάποια στιγμή ο επεξεργαστής φέρει ένα δεδομένο από έναν άλλο επεξεργαστή;
  - Το δεδομένο θα πάει στην cache
  - Το πρόβλημα της συνέπειας πώς λύνεται σε αυτή την περίπτωση (αφού δεν υπάρχει δίαυλος να «παρακολουθείται»);

- Δύο λύσεις:
  - ΑΠΑΓΟΡΕΥΕΤΑΙ τα απομακρυσμένα δεδομένα να μπαίνουν στην cache (π.χ. Cray T3D) ή
  - ΕΠΙΤΡΕΠΕΤΑΙ, αλλά επιπλέον χρησιμοποιούνται νέα πρωτόκολλα συνοχής, βασισμένα σε καταλόγους (**ccNUMA** – cache coherent NUMA)
    - Directory-based cache coherence

Τοποθέτηση δεδομένων / νημάτων

- Βασικό μέλημα το πού θα τοποθετηθούν τα νήματα και πού τα δεδομένα
  - Αν τα δεδομένα που χρησιμοποιεί πιο συχνά ένα νήμα πάνε σε άλλον επεξεργαστή από ότι το νήμα, τότε θα έχουμε καθυστερήσεις (NUMA factor)
  - Συνήθης τακτική στα λειτουργικά συστήματα:
    - **First touch** (όποιο νήμα προσπελάσει πρώτο κάποιο δεδομένο, τότε «τραβάει» το δεδομένο αυτό στην μνήμη του επεξεργαστή που το εκτελεί. Το δεδομένο δεν μετακινείται από εκεί και πέρα).
  - **ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΔΙΔΑΓΜΑ???**
    - Θέλει προσοχή στις *αρχικοποιήσεις.*
    - Δεν μπορεί τις αρχικοποιήσεις να τις κάνει μόνο ένα νήμα!!

8 cores – είναι τα ίδια;;

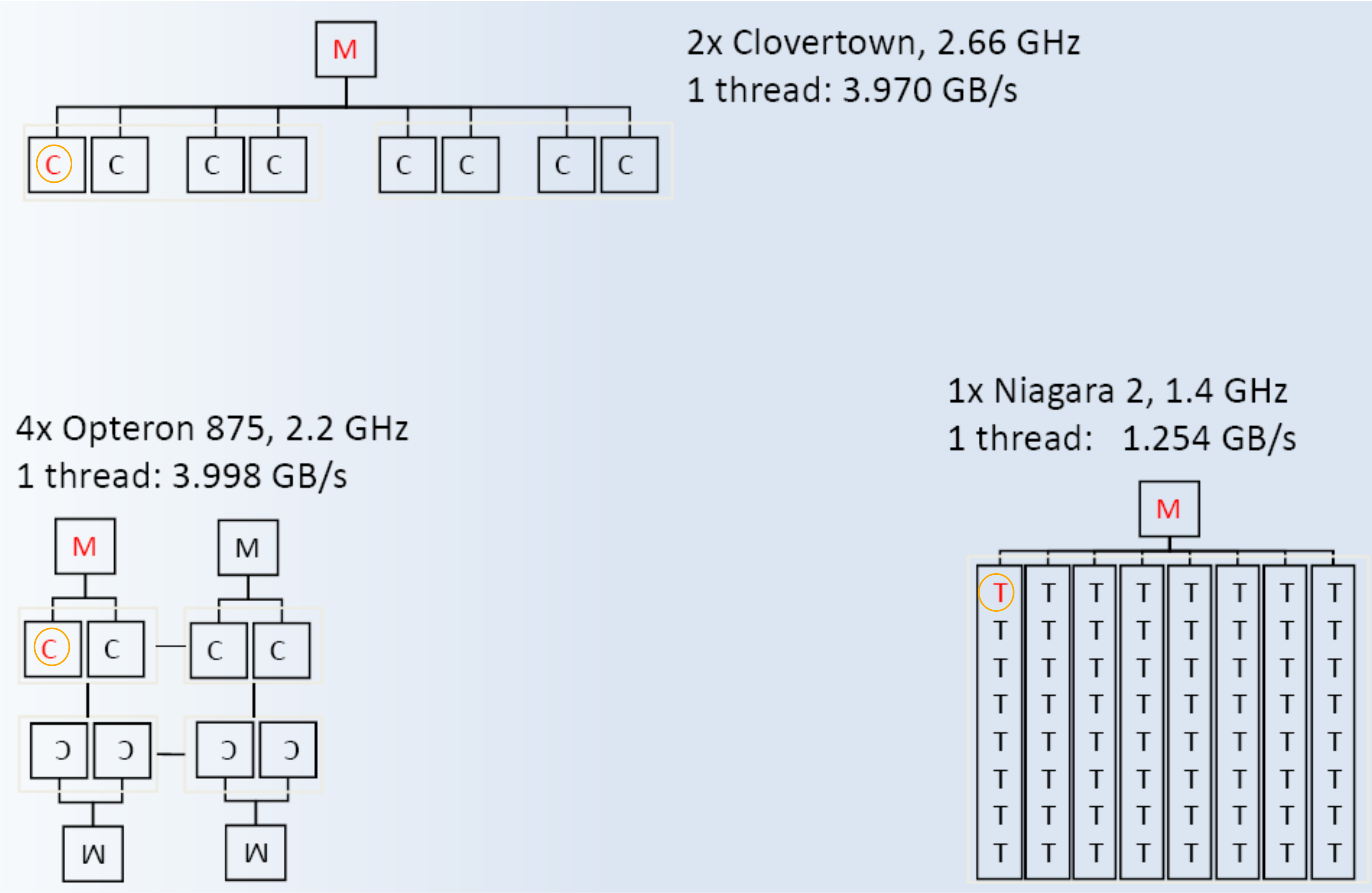
Metric \ Server	SF V40z	FSC RX200 S4	SF ???
Processor Chip	AMD Opteron 875 2.2 GHz	Intel Xeon 5450 3.00 GHz	UltraSPARC T2 1.4 Ghz
# sockets	4	2	1
# cores	8 (dual-core)	8 (quad-core)	8 (octo-core)
# threads	8	8	64
Accumulated L2 $	8 mb	16 mb	4 mb
L2 $ Strategy	Separate per core	Shared by 2 cores	Shared by 8 cores
Technology	90 nm	45 nm	65 nm
Peak Performance	35.2 GFLOPS	96 GFLOPS	11.2 GFLOPS
Dimension	3 units	1 unit	1 unit

Ταχύτητα προσπέλασης της μνήμης

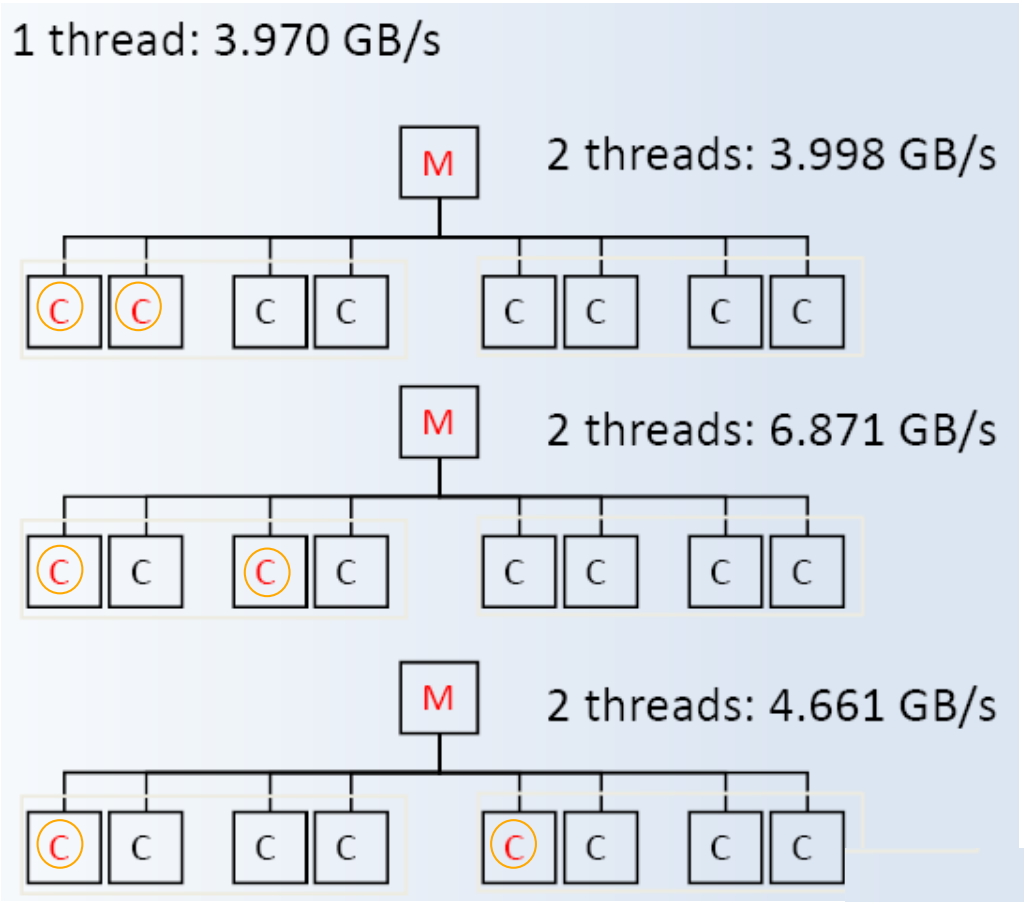
```
long long *x, *xstart, *xend, mask;
for (x = xstart; x < xend; x++) *x ^= mask;
```

- Κάθε επανάληψη κάνει μία ανάγνωση (load) και μία εγγραφή (store) στην μνήμη
- Παραλληλοποίηση με πολλά νήματα, το καθένα δούλευε μόνο στα δικά του δεδομένα (δηλαδή κάθε νήμα προσπελαύνει διαφορετικές διευθύνσεις – δεν υπάρχουν συγκρούσεις)
- Δοκιμές με διαφορετικές τοποθετήσεις των νημάτων

1 νήμα εκτελείται στο κάθε σύστημα

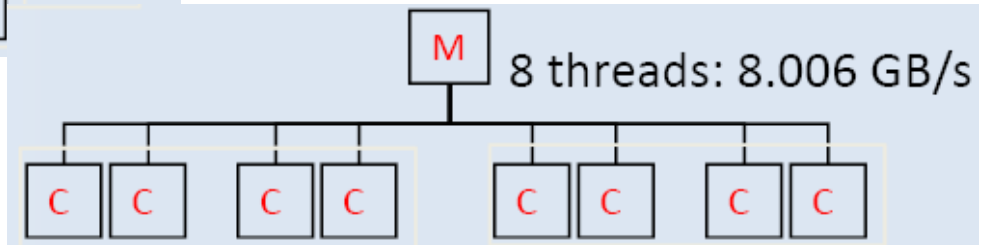


2 x 4core Intel Xeon



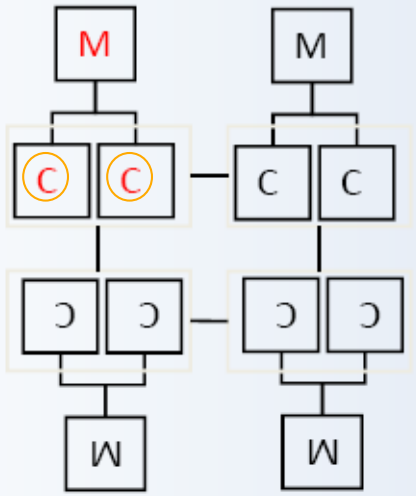
Scalability

☹

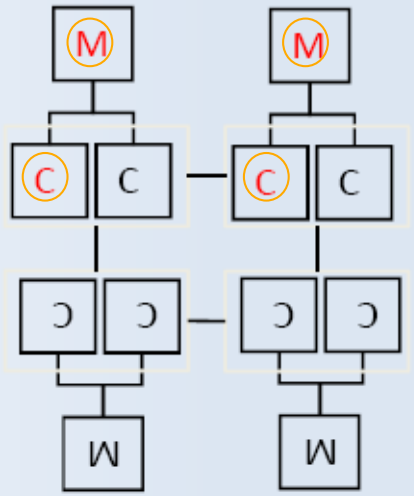


4 x 2core AMD Opteron

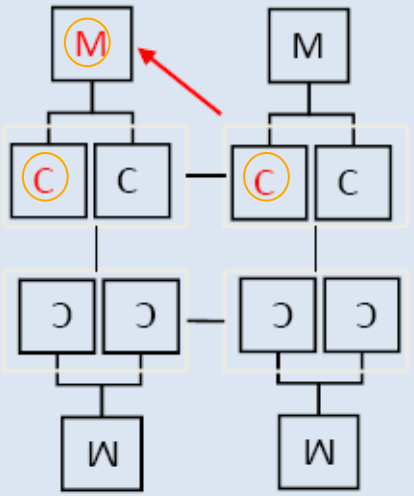
1 thread: 3.998 GB/s



2 threads: 4.674 GB/s



2 threads: 8.210 GB/s



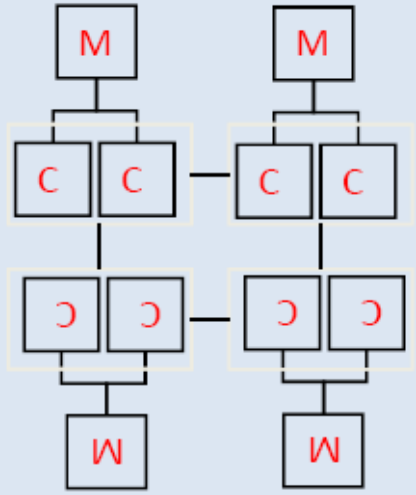
2 threads: 4.335 GB/s

Scalability

☺

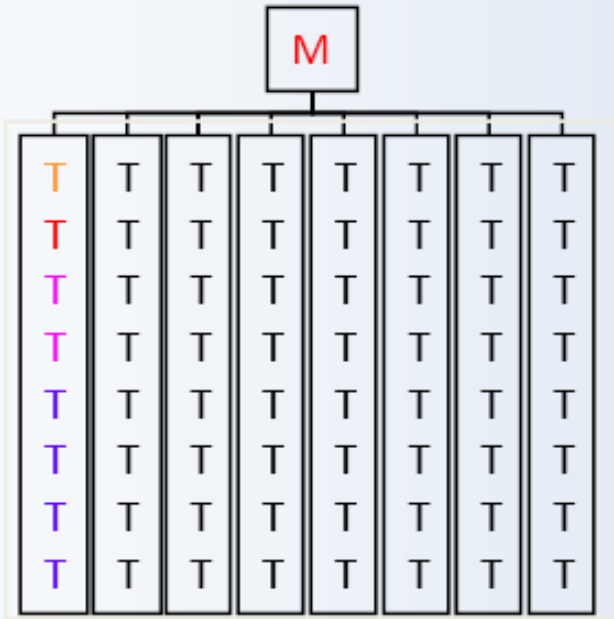
προσεκτικά

όμως



8 threads: 18.470 GB/s

1 x 8core Niagara T2 (με 8-way CMT ανά πυρήνα)



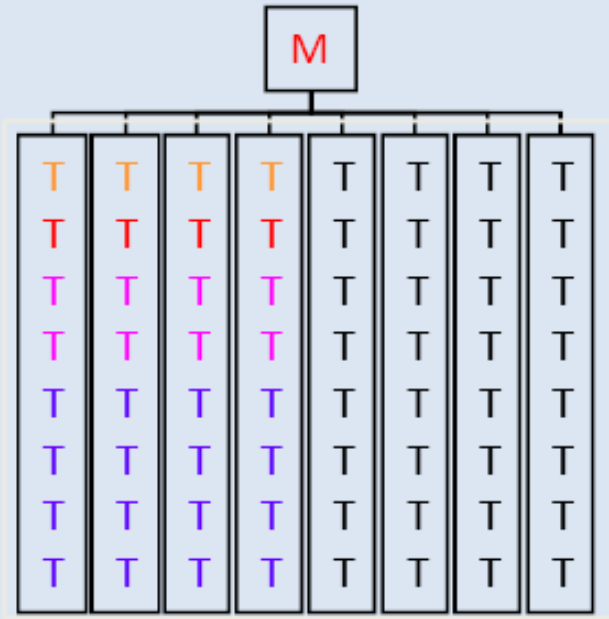
1 thread: 1.254 GB/s
2 threads: 2.405-2.455 GB/s
4 threads: 4.182-4.645 GB/s
8 threads: 7.367 GBs

Scalability

☺

Ομως αργός για 1 νήμα

4 threads: 4.997 GB/s
8 threads: 9.470-9.828 GB/s
16 threads: 12.459 GB/s
32 threads: 11.395 GB/s



Θέματα που θα καλύψουμε

- Οργάνωση και προγραμματισμός συστημάτων κοινής μνήμης
  - Κύρια έμφαση (πολυπύρηνα)
- Οργάνωση και προγραμματισμός συστημάτων κατανεμημένης μνήμης
- Προχωρημένα θέματα (ανάλογα με το χρόνο / ενδιαφέροντα)
  - Σύγχρονες τάσεις (GPUs, accelerators, OpenCL)
  - Αξιοπιστία, κατανάλωση ενέργειας
  - Προγραμματισμός συστήματος
  - Transactional memory
  - …

Οργάνωση του μαθήματος

- Διδάσκοντες (αλφαβητικά):
  - Β. Δημακόπουλος (Β33 – dimako@cs.uoi.gr)
  - Α. Ευθυμίου (Β29 – efthym@cs.uoi.gr)
- Διαλέξεις
  - Δευτέρες, 10:00 – 13:00
  - Ως μεταπτυχιακοί φοιτητές, δεν επιτρέπεται καμία απουσία, χωρίς συνεννόηση με τους διδάσκοντες ΑΠΟ ΠΡΙΝ.
- Μελέτη papers, προετοιμασία παρουσίασης και συζήτηση στην τάξη
  - Μετράει η συμμετοχή
- Εργασίες (προγραμματισμός ή πειραματισμός με προσομοιωτή)
  - Κάθε 2 εβδομάδες περίπου
- Project
  - Ανάπτυξη εφαρμογής και πλήρης μελέτη / τροποποίηση τμημάτων στον προσομοιωτή για βελτιστοποίηση επιδόσεων
  - Θα πούμε παραπάνω πράγματα σε λίγο καιρό

Οργάνωση του μαθήματος

- Πρόοδος / τελικές εξετάσεις (???)
  - Θα εξαρτηθούν από τον αριθμό των φοιτητών
    - Πολλοί: εξετάσεις και (αναγκαστικά) ομαδικά project
    - Αρκετοί: πρόοδος και projects, όχι τελικές εξετάσεις
    - Λίγοι: project και παραπάνω ασκήσεις για αποφυγή εξετάσεων
  - Εξαρτάται βέβαια και από το πώς ορίζεται το «πολλοί», «αρκετοί» και «λίγοι»
- Βαθμολογία
  - Πάλι, θα εξαρτηθεί από τον αριθμό των φοιτητών
- Σε 2 εβδομάδες θα καθοριστούν τα τελικά ποσοστά των ασκήσεων, παρουσιάσεων, project και τυχόν εξετάσεων

Ανακοίνωση

Τετάρτη, 7/3/2012

Ωρα 10:30

Αίθουσα σεμιναρίων

ΚΑΛΩΣΟΡΙΣΜΑ & ΕΝΗΜΕΡΩΣΗ ΤΩΝ ΝΕΩΝ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΦΟΙΤΗΤΩΝ ΤΟΥ ΤΜΗΜΑΤΟΣ