14/3/2023
Εισαγωγή στα συστήματα υψηλών επιδόσεων

Οργάνωση-Προγραμματισμός

Λ8
Συστήματα & Λογισμικό Υψηλών Επιδόσεων

Οργάνωση του μαθήματος

- Διδάσκων:
  - Β. Δημακόπουλος (Β33 – dimako@[cse.]uoi.gr)
- Διαλέξεις
  - Τρίτες, 10:00 – 13:00
  - Ως μεταπτυχιακοί φοιτητές, δεν επιτρέπεται καμία απουσία, χωρίς συνεννόηση με τους διδάσκοντες ΑΠΟ ΠΡΙΝ.

Τι θα κάνετε εσείς:

- Μελέτη papers, προετοιμασία παρουσίασης και συζήτηση στην τάξη
  - Μετράει η συμμετοχή
- Εργασίες (στο χαρτί και προγραμματιστικές)
  - Κάθε 2 εβδομάδες περίπου (6-7 σύνολο)
- Projects
  - Μελέτη θέματος/περιοχής που δεν καλύφθηκε [mini/reading project]
  - Μελέτη προβλήματος και ανάπτυξη εφαρμογής [full project]
  - 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: projects και παραπάνω ασκήσεις / αποφυγή εξετάσεων [το πιθανότερο]
  - Εξαρτάται βέβαια και από το πώς ορίζεται το «πολλοί», «αρκετοί» και «λίγοι» ☺
- Βαθμολογία
  - Πάλι, θα εξαρτηθεί από τον αριθμό των φοιτητών
  - Την επόμενη εβδομάδα θα καθοριστούν τα τελικά ποσοστά των ασκήσεων, παρουσιάσεων, project (και τυχόν εξετάσεων). Αν πάμε με το «Λίγοι», το πιθανότερο σχήμα είναι:
    - 30% ασκήσεις + 20% mini project + 40% full project + 10% συμμετοχή

Θέματα που θα καλύψουμε

- Οργάνωση και προγραμματισμός συστημάτων κοινής μνήμης
  - Κύρια έμφαση (πολυπύρηνα)
  - Νήματα, OpenMP, κλπ
- Οργάνωση και προγραμματισμός συστημάτων κατανεμημένης μνήμης
  - Δίκτυα διασύνδεσης και clusters
  - MPI και ίσως map-reduce
- Προχωρημένα θέματα (ανάλογα με το χρόνο / ενδιαφέροντα)
  - Σύγχρονες τάσεις (GPUs, accelerators, OpenCL, ετερογένεια)
  - Λογισμικό συστήματος
  - Αξιοπιστία, κατανάλωση ενέργειας
  - …

Προαπαιτούμενα

1. Βασικές γνώσεις στην οργάνωση των υπολογιστών
   - Επεξεργαστής, μνήμη, caches και ιεραρχία μνήμης κλπ.
2. Άνεση στον προγραμματισμό
   - Πολύ καλή γνώση και χρήση της C
3. Όρεξη
   - Αρκετή!

Το μάθημα στο γενικότερο τοπίο

ΔΕΥΤΕΡΑ	ΤΡΙΤΗ	ΤΕΤΑΡΤΗ	ΠΕΜΠΤΗ	ΠΑΡΑΣΚΕΥΗ
Δ8 (Γ. Μανής): **Ανάλυση και Επεξεργασία Βιοϊατρικών Δεδομένων** **10:00-13:00** ΑΙΘΟΥΣΑ Α2	**Λ8** (Β. Δημακόπουλος): **Συστήματα και λογισμικό υψηλών επιδόσεων** **10:00-13:00** ΑΙΘΟΥΣΑ Β2	**Δ0, Εισαγωγή στην Ανάλυση και Επεξεργασία Δεδομένων** **9:00-12:00** ΑΙΘΟΥΣΑ Β2	**Υ5** (Κ. Βλάχος): **Ρομποτικά Συστήματα** **10.00-13.00** ΑΙΘΟΥΣΑ Α2	**Α2** (Λ.Γεωργιάδης): **Αλγόριθμοι Επιστήμης Δεδομένων** **9:00- 12:00** ΑΙΘΟΥΣΑ Γ2
	Δ5 (Χ. Νίκου): **Υπολογιστική Οραση** **13:00- 14:30** ΑΙΘΟΥΣΑ Β2	**Υ0, Εισαγωγή στα Συστήματα Υλικού** **12:00-15.00 ΑΙΘΟΥΣΑ Β2**	**Δ5** (Χ. Νίκου): **Υπολογιστική Οραση** **13:00- 14:30** ΑΙΘΟΥΣΑ Β2	
	Υ1 (Α. Ευθυμίου): **Σύγχρονη Αρχιτεκτονική Υπολογιστών** **17:30-20:30** ΑΙΘΟΥΣΑ Β2		**Δ1** (Α. Λύκας/Κ. Μπλέκας): **Μηχανική Μάθηση** **15.30-18.30** ΑΙΘΟΥΣΑ Α2	

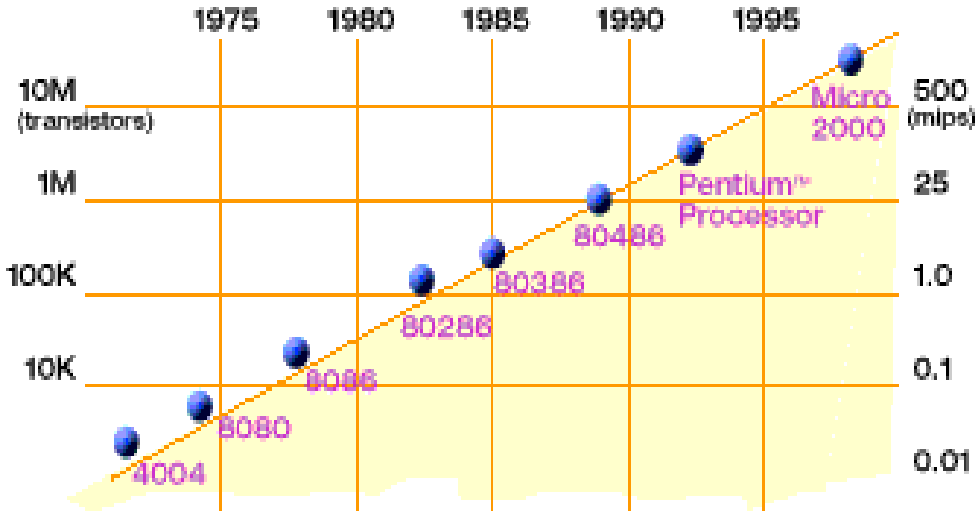
Το μάθημα στο γενικότερο τοπίο

Δευτέρα	20-2-2023	Έναρξη Μαθημάτων - Δηλώσεων - Εγγραφών
Παρασκευή	3-3-2023	Λήξη Δηλώσεων/Εγγραφών
Παρασκευή	24-3-2023	Λήξη παραιτήσεων από μάθημα
Παρασκευή	2-6-2023	Λήξη μαθημάτων
Δευτέρα	5-6-2023	Εναρξη Εξετάσεων
Παρασκευή	16-6-2023	Λήξη Εξετάσεων
Παρασκευή	2-6-2023	Λήξη προθεσμίας αιτήσεων για εξέταση Μεταπτυχιακής Εργασίας (M2)
Παρασκευή	23-6-2023	Παράδοση Βαθμολογίας

Προ-εισαγωγικά

Pre-introductory

Τεχνολογία – ο «νόμος» του Moore



2X transistors/Chip Every 1.5 years

Called “<u>Moore’s Law</u>”

<u>Τι σημαίνει αυτό;</u>

Απάντηση: πολυπλοκότητα!

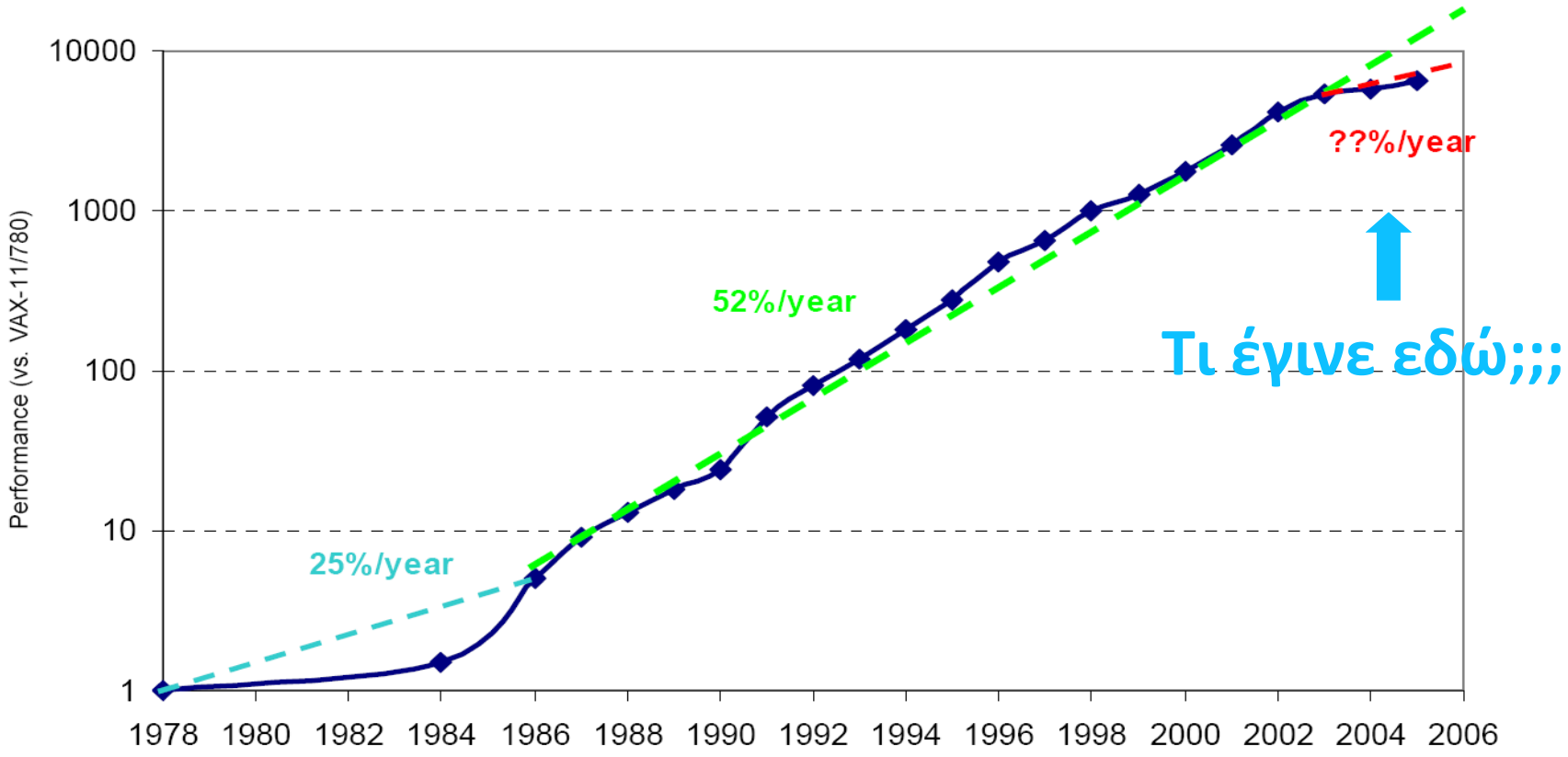
Τεχνολογία – η κλιμάκωση του Dennard

- Τα transistors που χρησιμοποιούνται στα ολοκληρωμένα κυκλώματα μικραίνουν («κλιμάκωση» προς τα κάτω) σε διαστάσεις με τη βελτίωση της τεχνολογίας όπως παρατήρησε και ο Moore. *Τι γίνεται αν πολλαπλασιαστούν με έναν παράγοντα α?*
- Απάντηση (Dennard): *η πυκνότητα ισχύος (ισχύς / επιφάνεια) μένει σταθερή ενώ η συχνότητα λειτουργίας αυξάνει.*

Μέγεθος	Επί	Σμίκρυνση κατά 30% (α = 0.7)
Μήκος, πλάτος, πάχος	**α**	↓ 30%
Επιφάνεια	**α^2**	↓ 50%
Τάση	α	
Ρεύμα	α	
Χωρητικότητα	α	
Καθυστέρηση διάδοσης	α	
Μέγιστη συχνότητα (F)	**1/α**	↑ 40%
Ενέργεια	α^3	
Ισχύς	**α^2**	↓ 50%

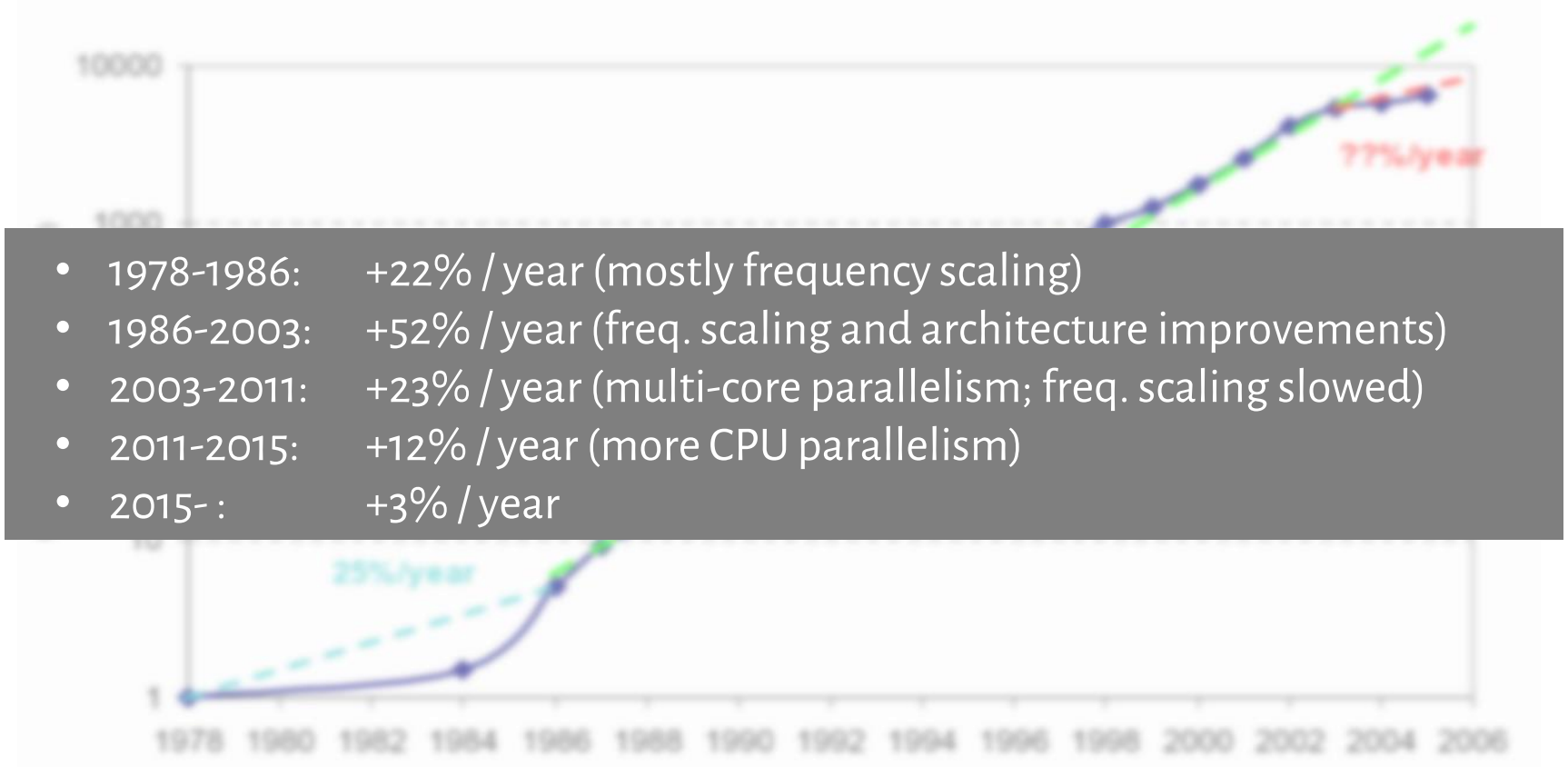
Άρα στον ίδιο χώρο πλέον:
- χωρούν 2 transistors
- καταναλώνεται συνολικά ΙΔΙΑ ισχύς
- είναι 40% ταχύτερα!

Επιδόσεις - ο «νόμος» του Bill **Joy**



- Based on integer SPEC benchmarks (“SPECint”)

Επιδόσεις - ο «νόμος» του Bill **Joy**

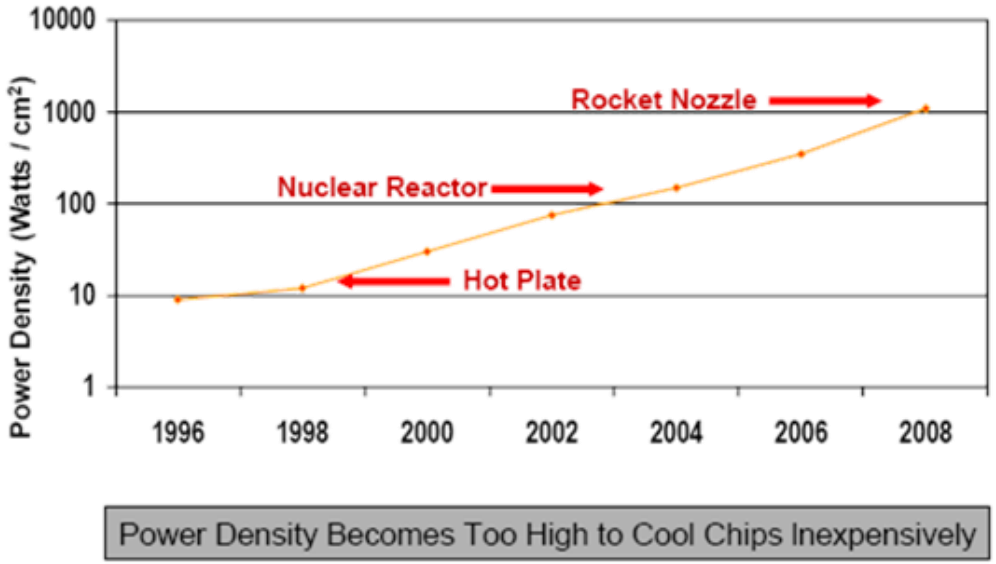


- 1978-1986: +22% / year (mostly frequency scaling)
- 1986-2003: +52% / year (freq. scaling and architecture improvements)
- 2003-2011: +23% / year (multi-core parallelism; freq. scaling slowed)
- 2011-2015: +12% / year (more CPU parallelism)
- 2015- : +3% / year

- Based on integer SPEC benchmarks (“SPECint”, single-core)

Η κατάρρευση της κλιμάκωσης του Dennard

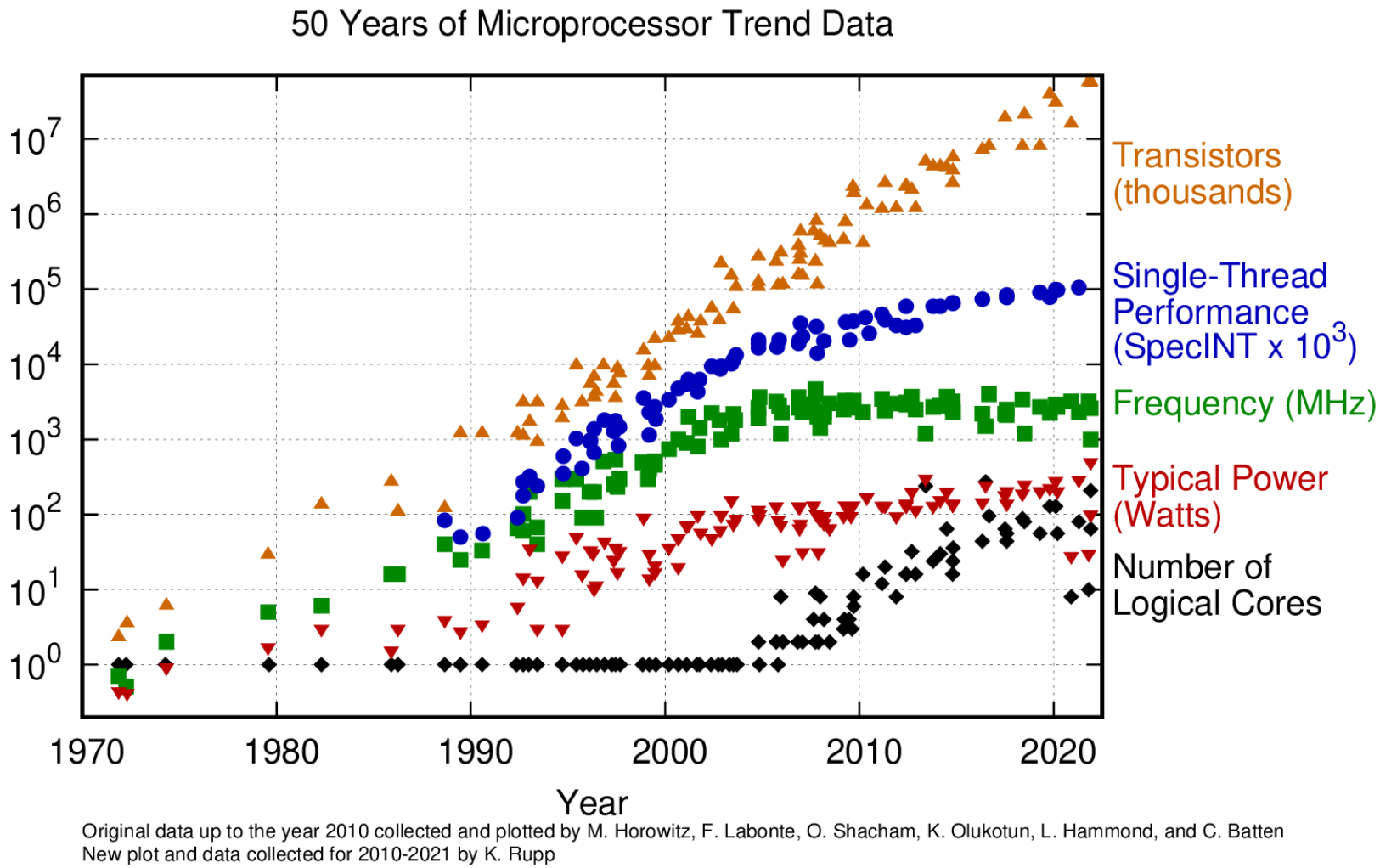
- Η κλιμάκωση και οι αναλογίες που κατέγραψε ο Dennard υπέθεταν ότι το λεγόμενο *ρεύμα διαρροής* είναι αμελητέο.
  - **Έπαψε να είναι αμελητέο γύρω στο 2005 όπου η τεχνολογία ήταν στα 65nm.**
- Περαιτέρω σμίκρυνση αυξάνει εκθετικά το ρεύμα διαρροής και κατά συνέπεια την καταναλισκόμενη ισχύ (αντί να την μειώνει)
  - **Κατάρρευση της κλιμάκωσης του Dennard.**



Τι κάνουμε;

- Όμως ο νόμος του Moore συνεχίζει να ισχύει…
  - Ερώτηση: Πως χειριζόμαστε την ισχύ των διπλάσιων transistors;
  - Απάντηση: **ρίχνουμε τη συχνότητα**
  - Ερώτηση: Και πως θα λέμε ότι ό νέος επεξεργαστής έχει βελτιωμένες επιδόσεις;
  - Απάντηση: **dual-core**. Χαμηλότερη συχνότητα μεν, δύο επεξεργαστές στον ίδιο χώρο δε!

Τάσεις των επεξεργαστών



Homework #1

Δείτε τη διάλεξη, διαβάστε το paper και γράψτε μια περίληψη (το πολύ 2 σελίδες):

«A New Golden Age for Computer Architecture»
BY JOHN L. HENNESSY AND DAVID A. PATTERSON

-- Turing Award, 2019

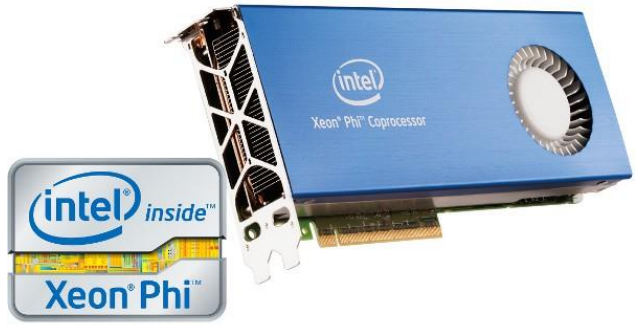
Πρόσφατο (σχετικά) παρελθόν

- SMPs (symmetric multiprocessors) – 10ετία του 2000
  - 2 – 4 επεξεργαστές (μονοπύρηνοι) συνηθισμένοι
  - Μέχρι 8 επεξεργαστές σε εμπορικά συστήματα
  - Πανάκριβα συστήματα με 12 - 16 επεξεργαστές, ελάχιστα
  - Κοινή μνήμη
- ΤΜΗΥΠ:
  - Πολλά Sun με 2 UltraSparc
  - Πολλά PC με 2 Pentium / Athlon και με διπύρηνους επεξεργαστές
  - **atlantis** με 4 Pentium III Xeon (700MHz) – 700.000 δρχ/CPU !
- ΤΜΗΥΠ: πολυεπεξεργαστές με πολυπύρηνους, ενδεικτικά:
  - **paraguay** με 4 x Intel Xeon 7000 Paxville (2008)
    - 2 cores per CPU / 2 threads per core, δωρεά της Intel USA στο Parallel Processing Group
  - **paragon** με 2 x AMD Opteron 6166 (2013)
    - 12 cores per CPU, custom-made by the Parallel Processing Group (PPG)
  - **parade** με 4 x Intel Xeon Gold 6130 (12/2019)
    - 16 cores per CPU, 64 cores total, 256 GiB RAM

Παρόν:
Dual core / Quad core / Multicore

Μέλλον;

- Πλέον και τα φτηνότερα PC έχουν επεξεργαστή τουλάχιστον διπύρηνο
- Το κινητό μου είναι 8-πύρηνο
- Διπύρηνοι αρχικά, τώρα 4/6/8/12πύρηνοι (Intel, AMD)
  - T1 (Sun Niagara): 8πύρηνοι (με 4-way multithreaded πυρήνες) από 12/2005!
  - T3 (2010): 16πύρηνοι, 8-way multithreaded
- Πολλαπλών πυρήνων (multicore) γενικά …
- **Manycore** (πάρα πολλών πυρήνων)
  - Μιλάμε για *πολλούς πυρήνες*
  - Τριψήφιο νούμερο (> 64)
- Πότε?
  - Εδώ και 16 χρόνια!
  - 80-πύρηνο πρωτότυπο από το 2007 (Intel)
  - Στην αγορά ως *Intel Xeon Phi* το 2012 με > 60 cores, ~5.000.000.000 transistors
  - Σταμάτησε η παραγωγή το 2018



Manycore: Μάλλον είναι ήδη παρόν…

- Μερικοί πρόσφατοι εμπορικοί επεξεργαστές
  - Intel Xeon Platinum 8490H : **60 cores** (Q1 2023)

  - AMD EPYC 9654 : **96 cores** (από 2022, θα φτάσει 128 cores μέσα στο 2023)

  - Ampere Altra Max : **128 cores** (από 2021, ARM-based)

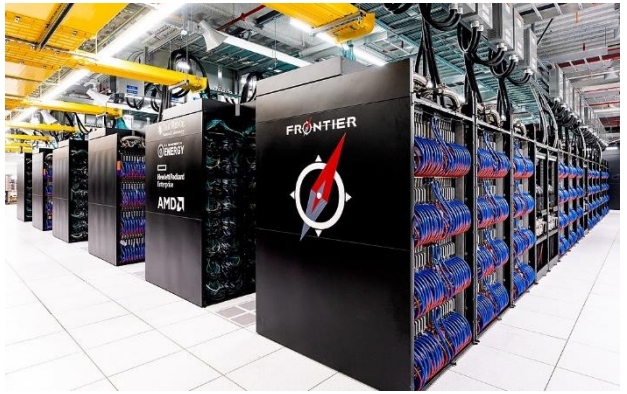
Clusters

- Παντού clusters
  - Συλλογή από διασυνδεδεμένους «κόμβους»
    - Φτηνοί / ευρέως διαθέσιμοι επεξεργαστές (π.χ. clusters από PCs)
  - Ο μόνος τρόπος να φτιάξουμε «οικονομικούς» υπερ-υπολογιστές
  - Με απλά PC + ethernet (“Beowulf” clusters)
  - Σε μεγάλα μεγέθη, δεν βολεύουν τα κουτιά των PC
    - πάμε σε «φέτες» (blades) και racks

Λεπτομέρεια: Top500, Θέση #1

(Νοε. 2022)

- FRONTIER – *1st Hexaflop Machine*
  - Oak Ridge National Laboratory (USA)
  - **9472 nodes** (74 καμπίνες, 64 blades each, 2 nodes per blade)
  - 1 AMD EPYC 64c CPU + 4 AMD Instinct 250X GPUs per node
  - Total number of **cores:** **8,730,112**
  - Interconnect: Slingshot (HPE/Cray) – 145 km καλωδίωση
  - Manufacturer: HPE
  - Performance: **1,102.00 PFlop/s (linpack)**
  - ~ 21 MW
  - Κόστος: ~ $600.000.000



Λεπτομέρεια: Top500, Θέση #2

- FUGAKU
  - RIKEN Center for Computational Science (Japan)
  - **158,976 nodes** (396 racks × 384 nodes/rack + some more)
  - 50-52 cores/node, 32 GiB memory/node
  - Total number of **cores:** **7,630,848**
  - Processor: A64FX 48C 2.2GHz (Arm V8.2-A)
  - Total **memory:** **5,087,232 GB**
  - Interconnect: TufuD (6D torus)
  - Manufacturer: Fujitsu
  - ~ 30 MW
  - Κόστος: ~ $1.000.000.000

Λεπτομέρεια: Top500, Θέση #3

- LUMI – *European Machine*
  - Kajaani, Finland
  - **2560 G-nodes** (1 x 64-core AMD CPU + 4 x AMD MI250X GPUs per node)
  - **1536 C-nodes** (2 x 64-core 3rd-genAMD EPYC CPUs per node).
  - Total number of **cores:** **2,220,288**
  - Interconnect: Slingshot 11 (HPE/Cray)
  - Manufacturer: HPE
  - Performance: **309 PFlop/s (linpack)**
  - ~ 6 MW
  - Κόστος: ~ $145.000.000

Clusters

- Πανεπιστήμιο Ιωαννίνων
  - Κέντρο προσομοιώσεων: 200 κόμβοι (κάθε κόμβος pc με 2 επεξεργαστές, *αιωνία του η μνήμη…*)
  - ΤΜΗΥΠ (παλαιό cluster): 16 κόμβοι, κάθε κόμβος 2 CPUS, κάθε CPU διπύρηνη (AMD Opteron).
  - ΤΜΗΥΠ (νέο, 2020!): 12 κόμβοι, σε κάθε κόμβο: 8 cores/16 threads και 8 GiB RAM (Intel Xeon E5-2620 v4)
  - Και τα τρία με δίκτυο gigabit Ethernet
- Βελτιωμένες επιδόσεις με δίκτυα χαμηλής καθυστέρησης
  - Π.χ. Myrinet, InfiniBand
  - Πολύ ακριβότερα όμως
  - Κάρτα δικτύο gigabit: 10-20€, 12-port switch ~50€
  - Κάρτα InfiniBand ~800€, 24-port switch ~35.000€
- Το μέλλον:
  - Clusters από πολυπύρηνους κόμβους
  - **Project ΔΙΩΝΗ!!**

GPUs, GPGPUs 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
  - Διαχείριση μνήμης με το «χέρι»

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

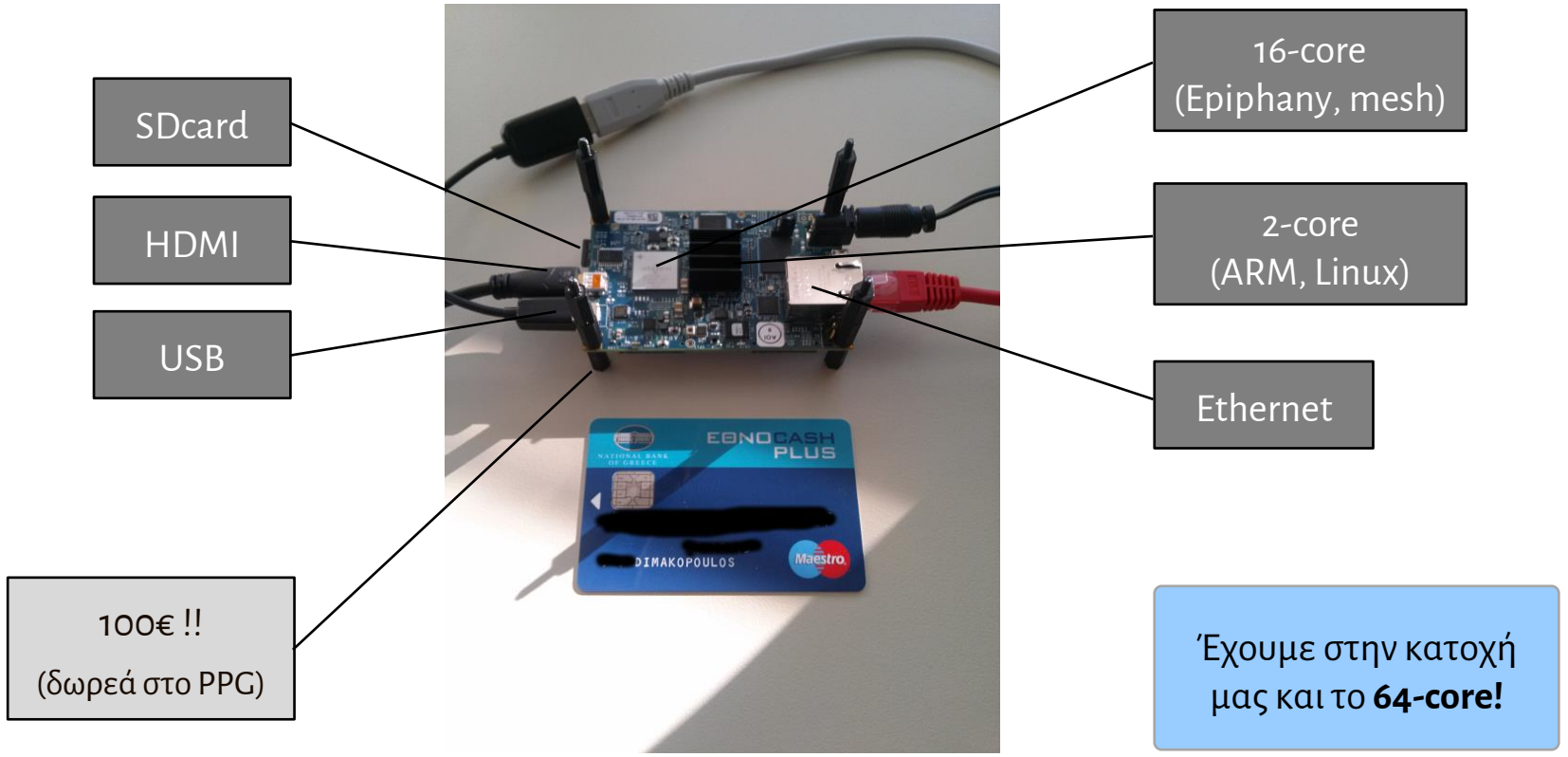
Mobile

- Πολυπύρηνες είναι πλέον ακόμα και οι μικρότερες συσκευές.
- Π.χ. κινητά τηλέφωνα (2/2022):

Octa core (45€)

Embedded

- Και συσκευές για ενσωματωμένα συστήματα.
- Π.χ. *parallella*:



Embedded – NVIDIA Jetson

- Jetson Nano 2GB

USB (2.0 & 3.0)

HDMI

Ethernet

WiFi

MicroSD

8cm x 10cm

COMPUTE

RAM: 2GB

CPU: Quad-core ARM® A57

GPU: 128-core NVIDIA Maxwell

2 Development Kits donated to PPG by NVIDIA (Jan. 2022)

Απλές ερωτήσεις

- Το excel θα τρέξει γρηγορότερα σε αυτά τα μηχανήματα;
  - Όχι!
- 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;
- OK, 2-8 cores ίσως τα απασχολώ ταυτόχρονα με πολλές ανοιχτές εφαρμογές (κάθε μία «κάθεται» και σε άλλο core, άρα τα εκμεταλλεύομαι)
  - 1 browser, 1 word, 1 excel, 1 για μουσική κλπ. κλπ.
- Αν όμως έχω τον Ampere Altra Max με τα 128 cores, τι κάνω;
  - Τα 120 cores ελέγχουν για ιούς;
- Πρέπει η *εφαρμογή να έχει προγραμματιστεί παράλληλα ώστε να χρησιμοποιεί τους πολλαπλούς επεξεργαστές*

Προγραμματισμός

- Πρέπει να μάθω νέες γλώσσες προγραμματισμού;
  - ΥΠΑΡΧΟΥΝ ιδιαίτερες γλώσσες προγραμματισμού, όντως.
  - Στην πράξη όμως συνεχίζουμε και χρησιμοποιούμε τις γνωστές γλώσσες (C, C++, Fortran)
  - Είτε με επιπλέον «εντολές»/οδηγίες είτε με νέες βιβλιοθήκες (κλήσεις συναρτήσεων)
- Διαφέρει από τον κλασικό σειριακό προγραμματισμό;
  - Σίγουρα το πρόγραμμα δομείται διαφορετικά
  - Σπάει σε «κομμάτια» όπου το καθένα το εκτελεί άλλος επεξεργαστής
  - Εξαρτάται από την οργάνωση του παράλληλου συστήματος
- Τι θα μάθω;
  - Κατ' ελάχιστον: Νήματα, OpenMP, MPI
  - Ανάλογα με τα ενδιαφέροντα, κάποια από OpenCL, CUDA, ή υψηλότερου επιπέδου μοντέλα (π.χ. Map-reduce)

Σύντομη εισαγωγή στην κοινόχρηστη 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);
}
```

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

Διεργασίες & νήματα

```
#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);
}
```

- 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)

- 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
  - Όλα θα τρέχουν εντολές από τον *ίδιο* κώδικα (δεν δημιουργείται άλλο αντίγραφό του) και τέλος
  - Όλα 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 λίγα 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
  - Όλοι προσπελαύνουν την ίδια μνήμη (κοινόχρηστη)

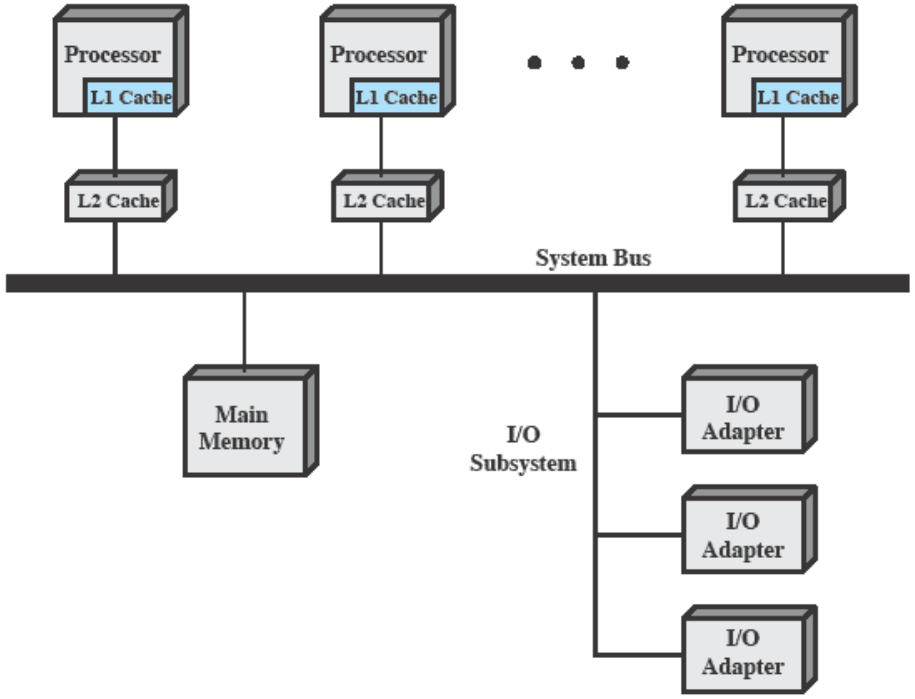


Figure 4.9 Symmetric Multiprocessor Organization

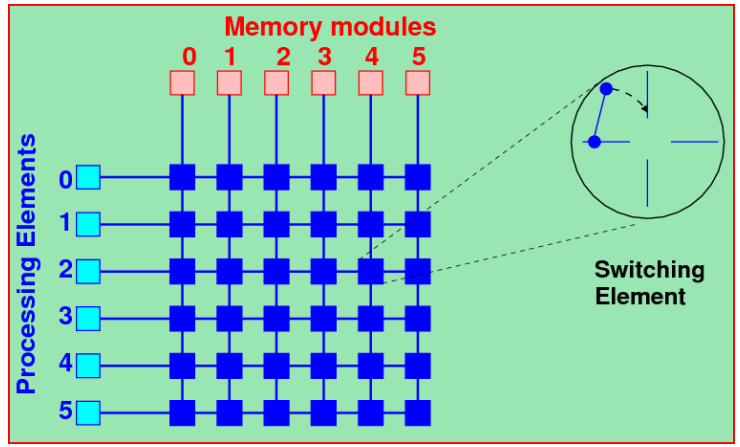
SMPs

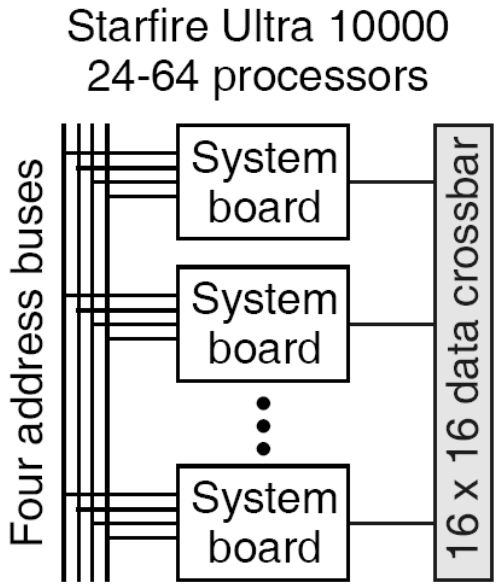
- Οικονομική λύση, επιτυχημένα εμπορικά
- Λίγοι επεξεργαστές (έως 10+, το πολύ)
- Ο δίαυλος είναι το bottleneck

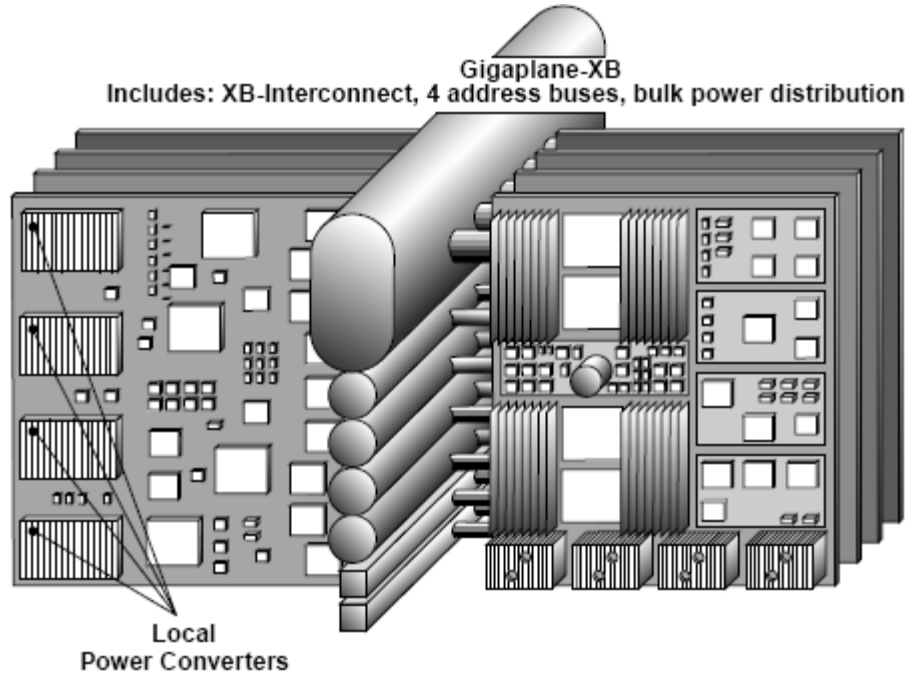
- Γενικά, οι επεξεργαστές όλοι βλέπουν και απευθύνονται στη κύρια μνήμη σαν να είναι ενιαία και εξ ολοκλήρου προσπελάσιμη
- **UMA** (= Uniform Memory Access)
  - Όλοι αντιλαμβάνονται την ίδια καθυστέρηση στην προσπέλαση οποιουδήποτε δεδομένου

Switch-based διασύνδεση

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

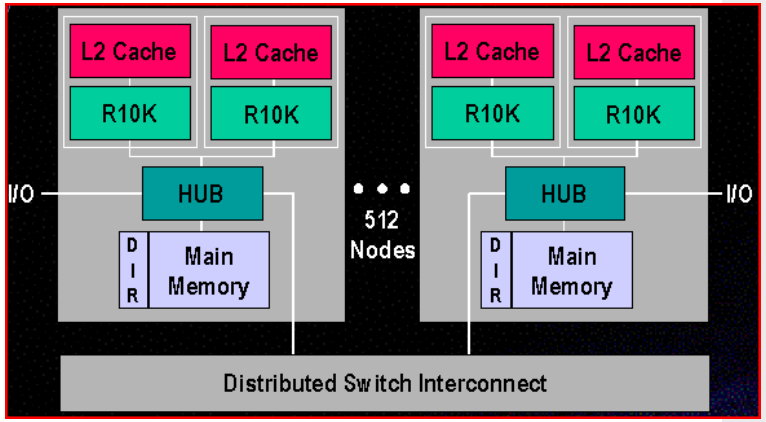






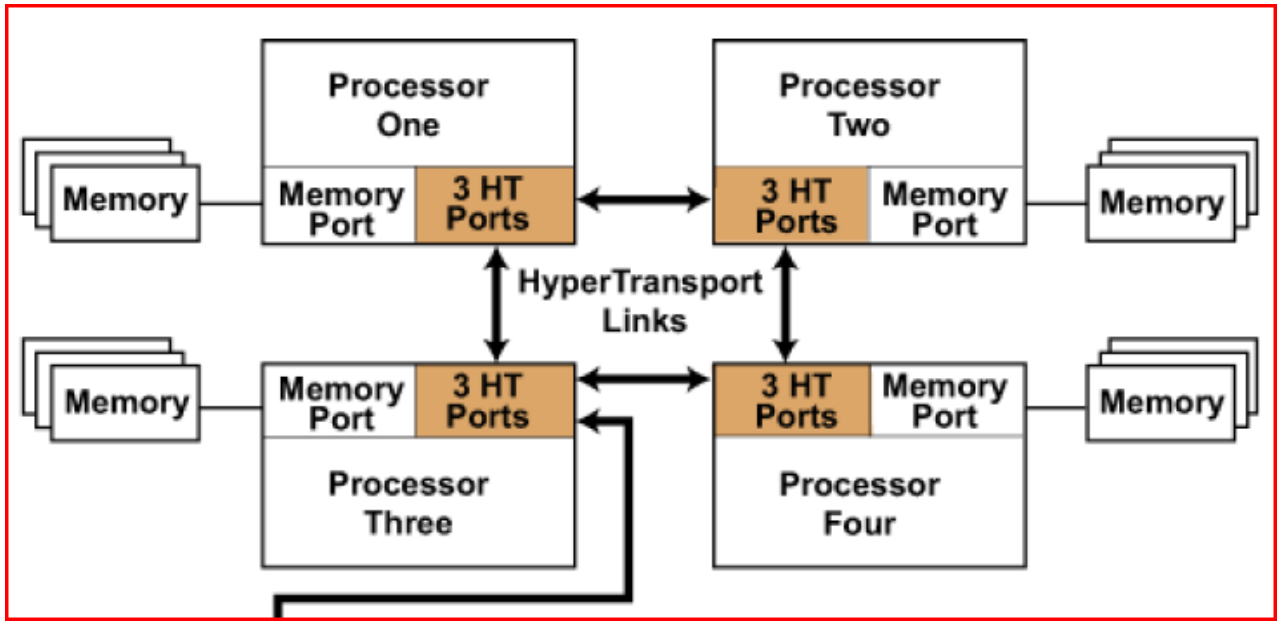
Ανομοιόμορφη προσπέλαση (NUMA)

- 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
    - Άρα 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
- Ότι 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 (παλαιότερο) φορητό μου:

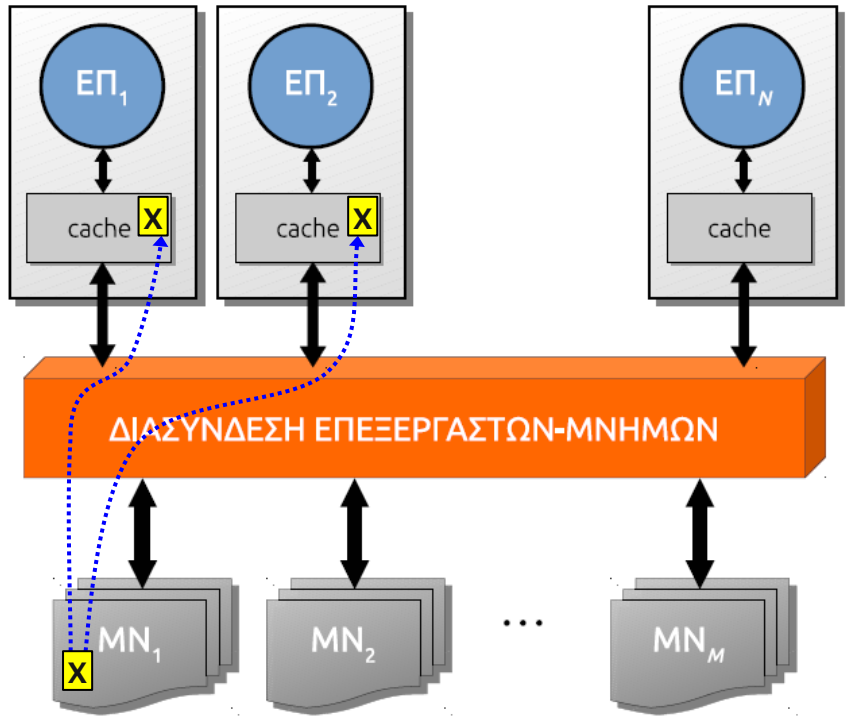
1.2 sec

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

Χρόνος εκτέλεσης στο (παλαιότερο) φορητό μου:

14 sec

Cache coherency



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

Τι γίνεται αν το νήμα 1 αλλάξει το Χ (π.χ. κάνει Χ = Χ+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

- Όλοι 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 αρκετά παλαιότερα 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;
```

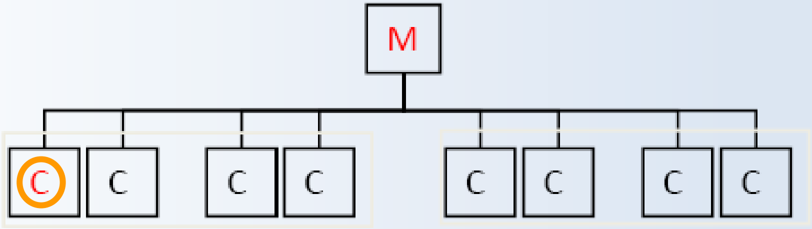
- Κάθε επανάληψη κάνει μία ανάγνωση (load) και μία εγγραφή (store) στην μνήμη
- Παραλληλοποίηση με πολλά νήματα, το καθένα δούλευε μόνο στα δικά του δεδομένα (δηλαδή κάθε νήμα προσπελαύνει διαφορετικές διευθύνσεις – δεν υπάρχουν συγκρούσεις)
- Δοκιμές με διαφορετικές τοποθετήσεις των νημάτων

For more information, see the ***STREAM Benchmark***

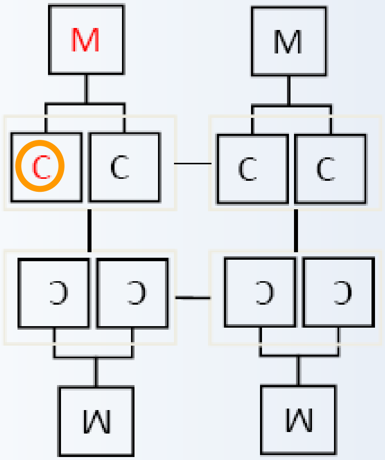
https://www.cs.virginia.edu/stream/ref.html

1 νήμα εκτελείται στο κάθε σύστημα

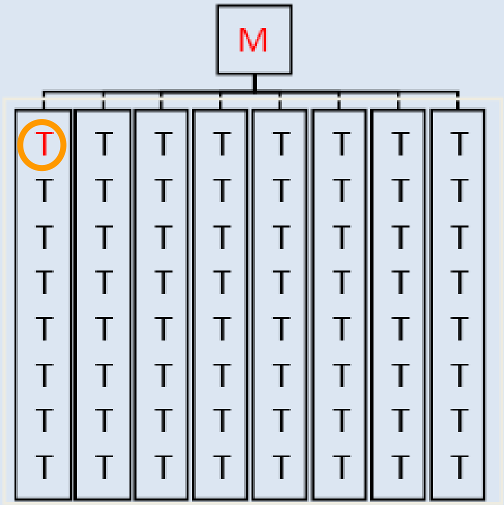
2x Clovertown, 2.66 GHz
1 thread: 3.970 GB/s



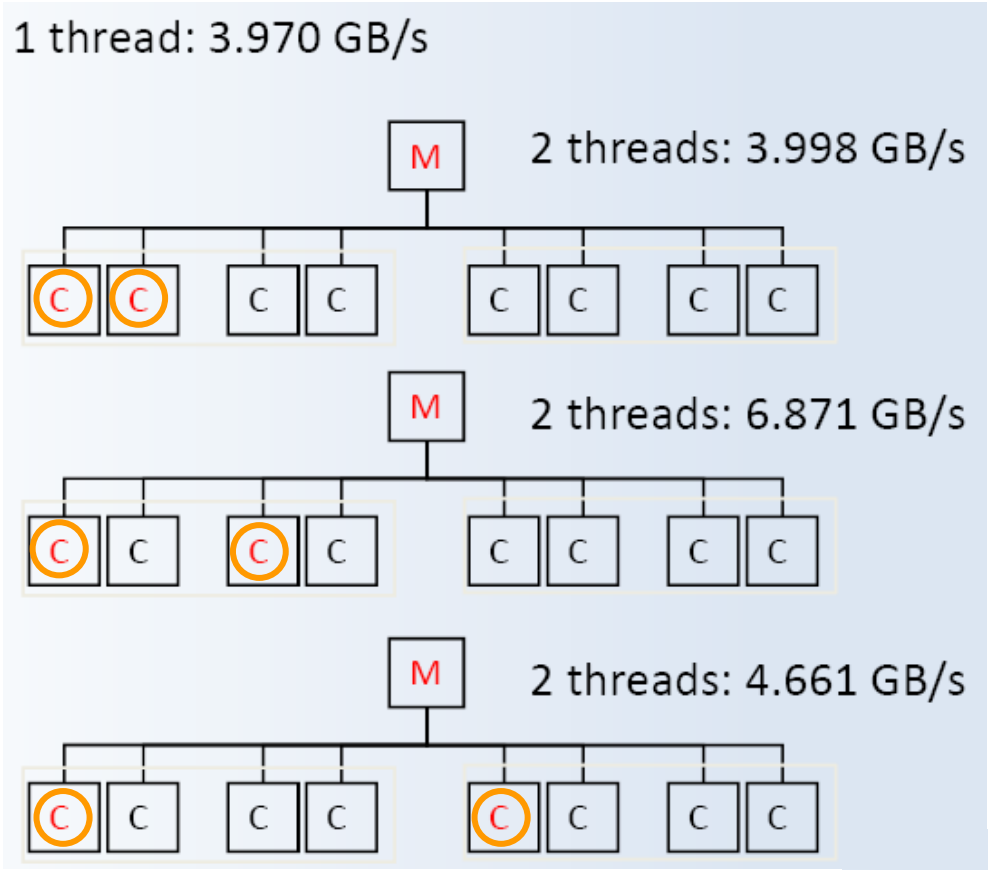
4x Opteron 875, 2.2 GHz
1 thread: 3.998 GB/s



1x Niagara 2, 1.4 GHz
1 thread: 1.254 GB/s

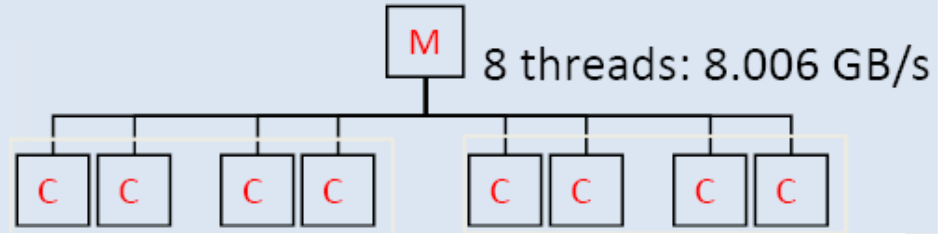


2 x 4core Intel Xeon

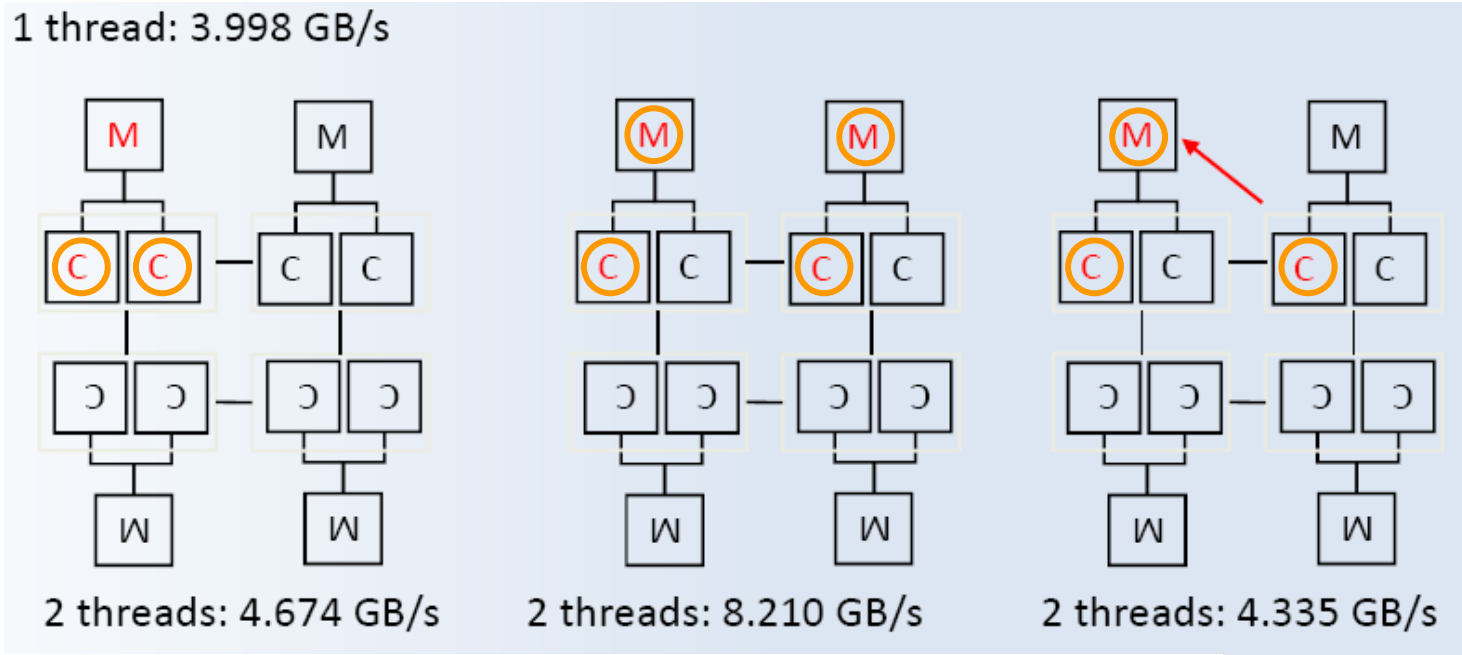


Scalability

☹



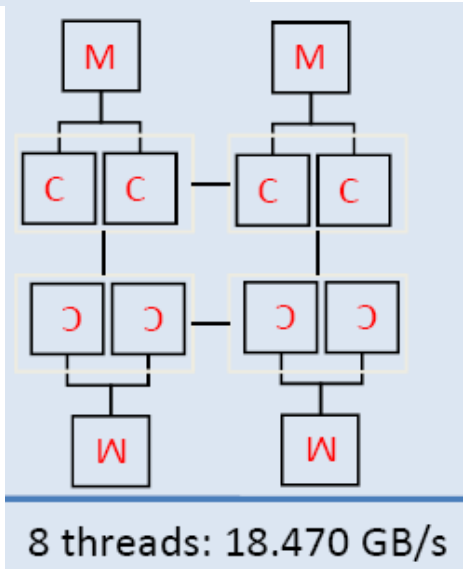
4 x 2core AMD Opteron



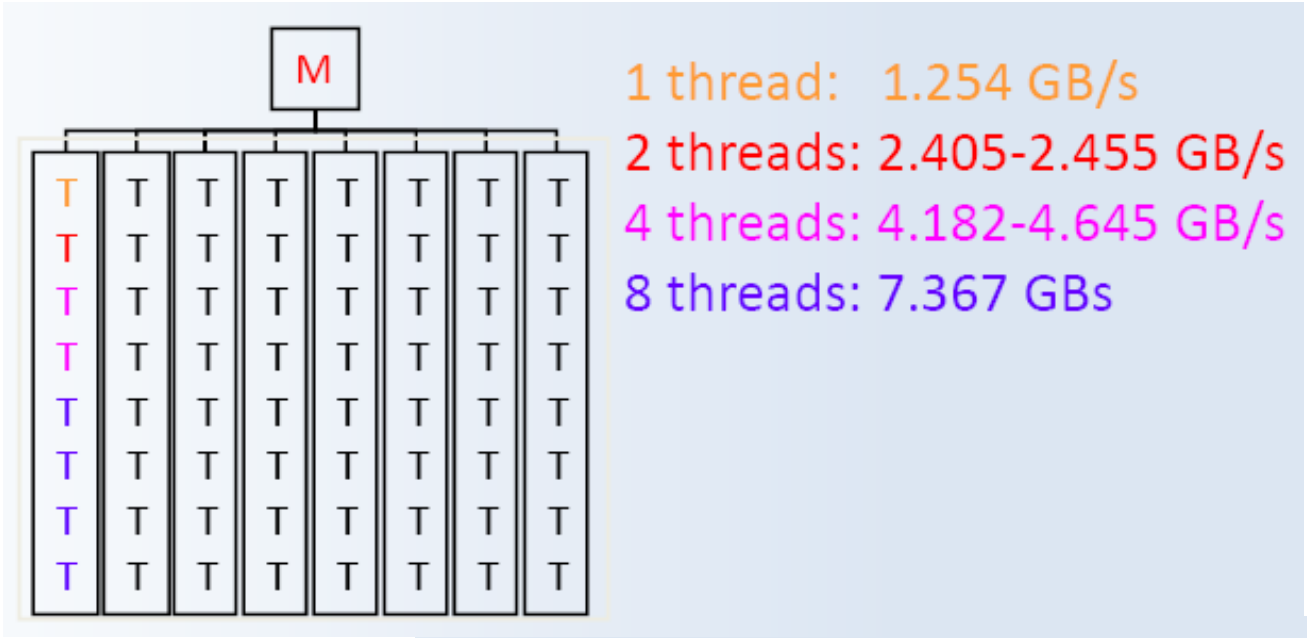
Scalability

☺

προσεκτικά όμως



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



Scalability

☺

Όμως αργός για 1 νήμα