ΜΥΕ-023
ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ & ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
2023-24

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ

Διδάσκων:

Β. Δημακόπουλος
dimako@cse.uoi.gr,
dimako@uoi.gr

Στοιχεία για το μάθημα

- Αντικείμενο του μαθήματος:
  1. Η δομή των παράλληλων υπολογιστών (πολυπύρηνα, SMPs, clusters, GPUs κλπ)
  2. Ο προγραμματισμός τους (threads, OpenMP, MPI κλπ)

- Απευθύνεται κυρίως σε ≥ 4ο έτος φοιτητές, με εμπειρία σε
  - Προγραμματισμό συστημάτων σε C
  - Λειτουργικά συστήματα
  - Αρχιτεκτονική υπολογιστών

Στοιχεία για το μάθημα

- Ημέρα/ώρα:
  - **Δευτέρα**, **09:00–12:00**
- Διδακτικό υλικό:
  - Βιβλία (Εύδοξος):
    1. «Εισαγωγή στον παράλληλο προγραμματισμό» - P. Pacheco (2015)
    2. «Εισαγωγή στον παράλληλο υπολογισμό» - Πάντζιου, Μάμαλης, Τομαράς (2013)
    3. «Προγραμματισμός και αρχιτεκτονική συστημάτων παράλληλης επεξεργασίας» – Παπαδάκης & Διαμαντάρας (2012)
    4. «Προγραμματισμός μαζικά παράλληλων επεξεργαστών» - D. B. Kirk, W-m. W. Hwu (2010)
  - Πρόσθετο Διδακτικό Υλικό (Εύδοξος):

    «Παράλληλα Συστήματα και Προγραμματισμός» – Β. Δημακόπουλος (2017)
    - 1η αναθεωρημένη έκδοση, ελεύθερο σύγγραμμα σε PDF
    - http://repository.kallipos.gr/handle/11419/3209
    - ***Κύριο σύγγραμμα***
- Ιστοσελίδα μαθήματος:
  - https://www.cse.uoi.gr/~dimako/teaching/

Εργασίες και βαθμολόγηση μαθήματος

- Επιλογή τρόπου βαθμολόγησης εξατομικευμένα:
  - **Πρώτη επιλογή:**
    Μικρές προγραμματιστικές ασκήσεις + τελικές εξετάσεις
    - Ασκήσεις εκμάθησης παράλληλου προγραμματισμού
    - Συνολικά 2-3 σε όλο το εξάμηνο
    - Κάθε μία πιάνει 10% στην τελική βαθμολογία
  - **Δεύτερη επιλογή:**
    Μόνο τελική εξέταση
    - Ως τελευταία (και με λιγότερη ουσία) επιλογή
  - **Τελικός βαθμός:**
    - max { βαθμός-με-εργασίες, βαθμός-μόνο-τελικής-εξέτασης }

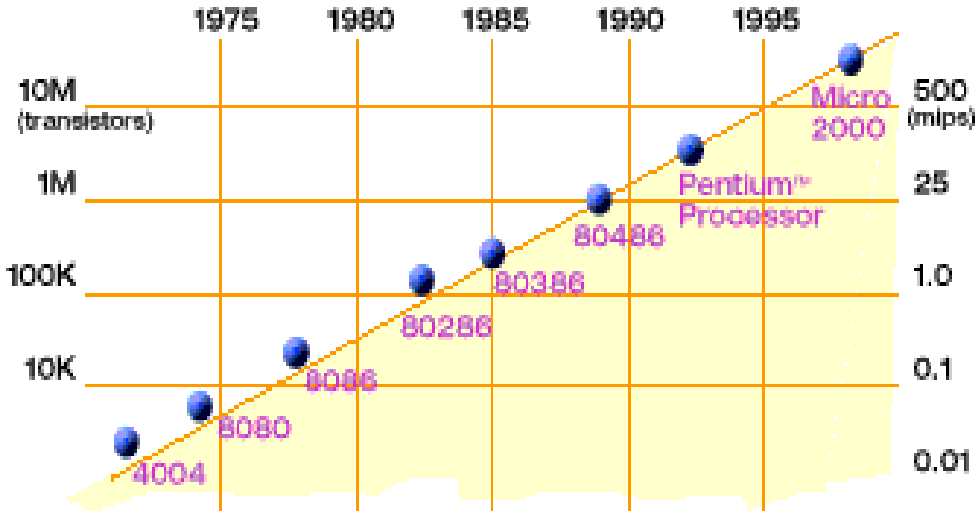
MYE023

Προ-εισαγωγικά

Pre-introductory

PARALLEL
PROCESSING
GROUP

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!

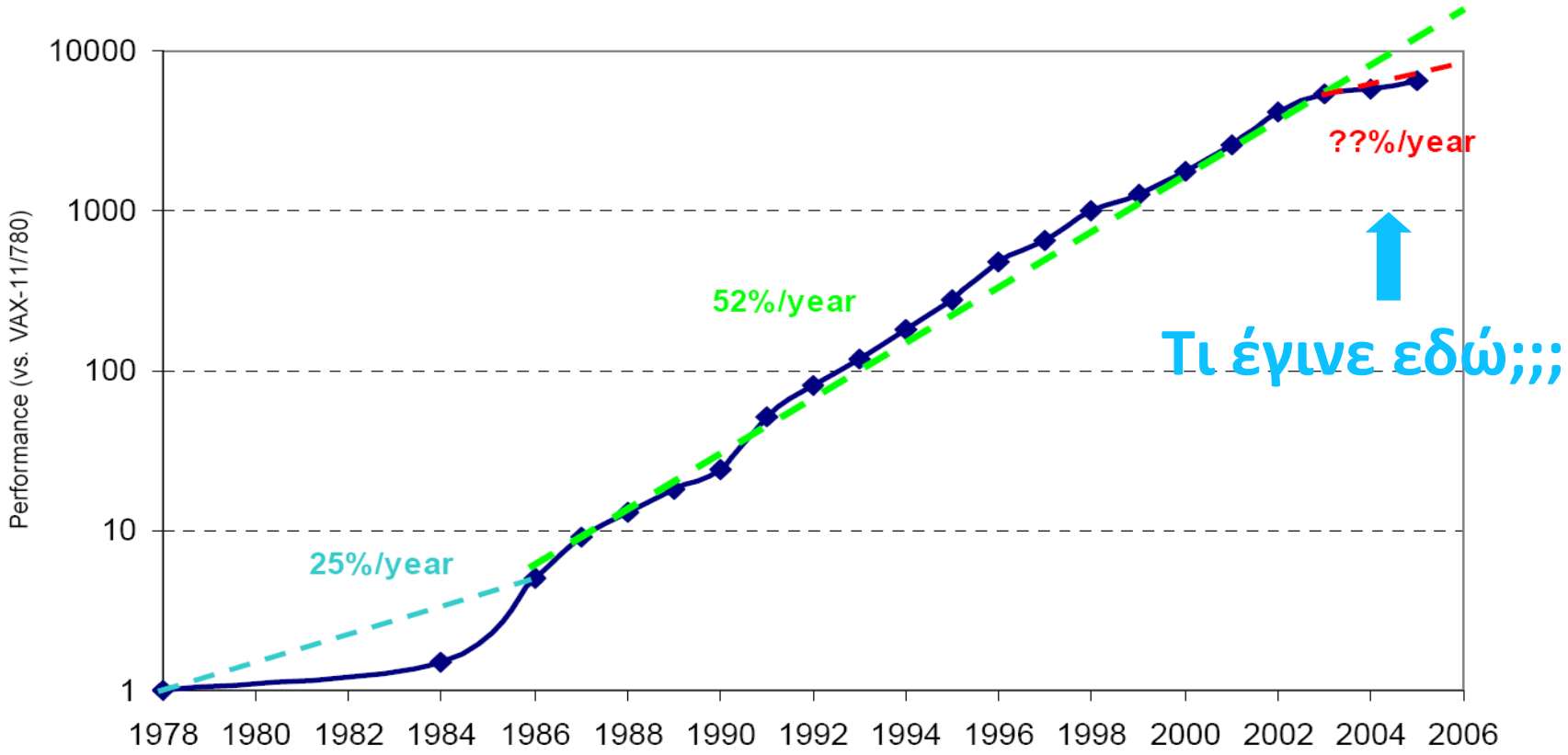
Τεχνολογία – η κλιμάκωση του Dennard

- 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%

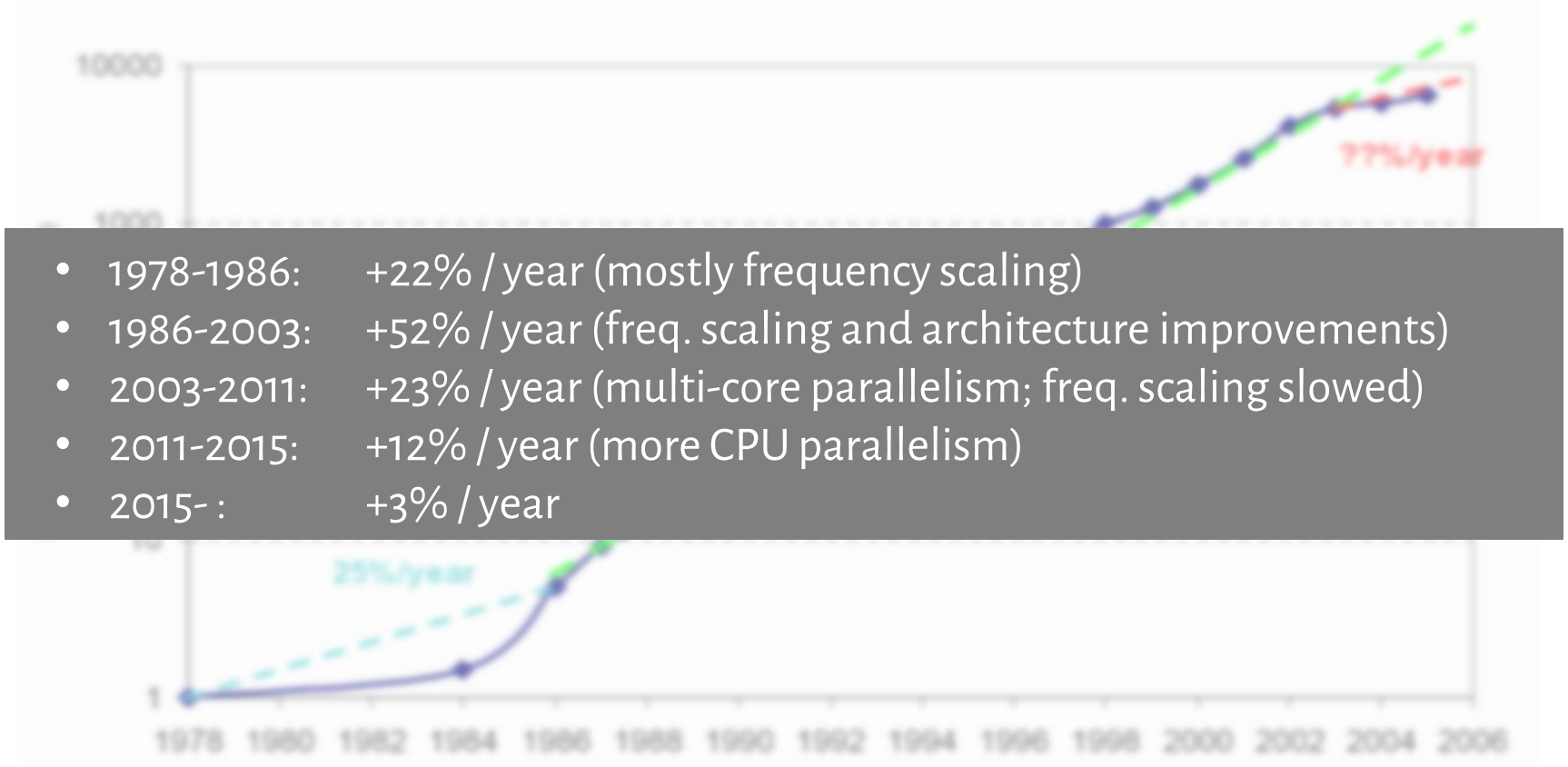
Άρα στον ίδιο χώρο πλέον:
- χωρούν 2 transistors
- καταναλώνεται συνολικά ΙΔΙΑ ισχύς
- είναι 40% ταχύτερα!

Επιδόσεις - ο «νόμος» του Bill Joy



- Based on integer SPEC benchmarks ("SPECint")

Εξέλιξη στις επιδόσεις (SPECint – single core)

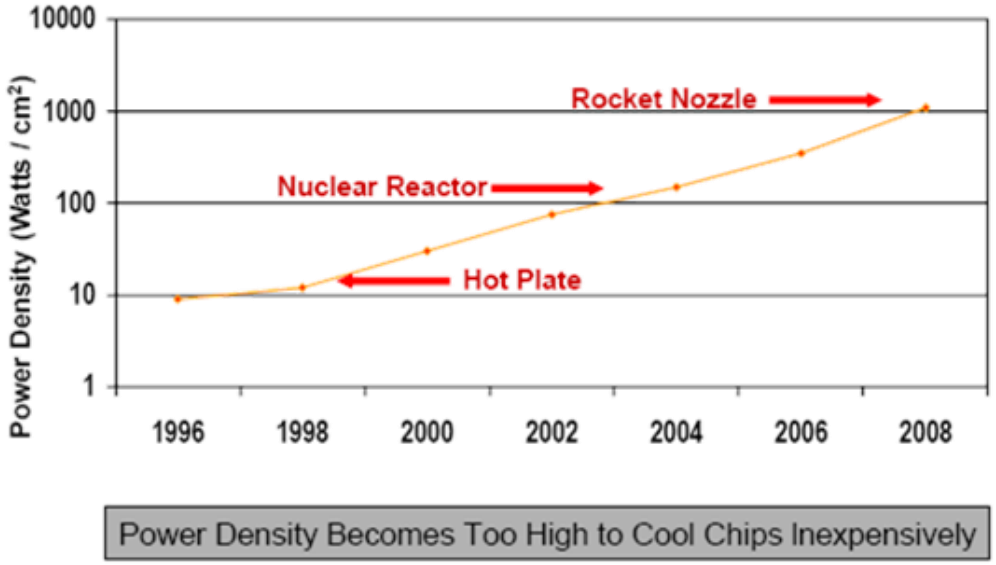


- 1978-1986: +22% / year (mostly frequency scaling)
- 1986-2003: +52% / year (freq. scaling and architecture improvements)
- 2003-2011: +23% / year (multi-core parallelism; freq. scaling slowed)
- 2011-2015: +12% / year (more CPU parallelism)
- 2015- : +3% / year

- Based on integer SPEC benchmarks (“SPECint”)

Η κατάρρευση της κλιμάκωσης του Dennard

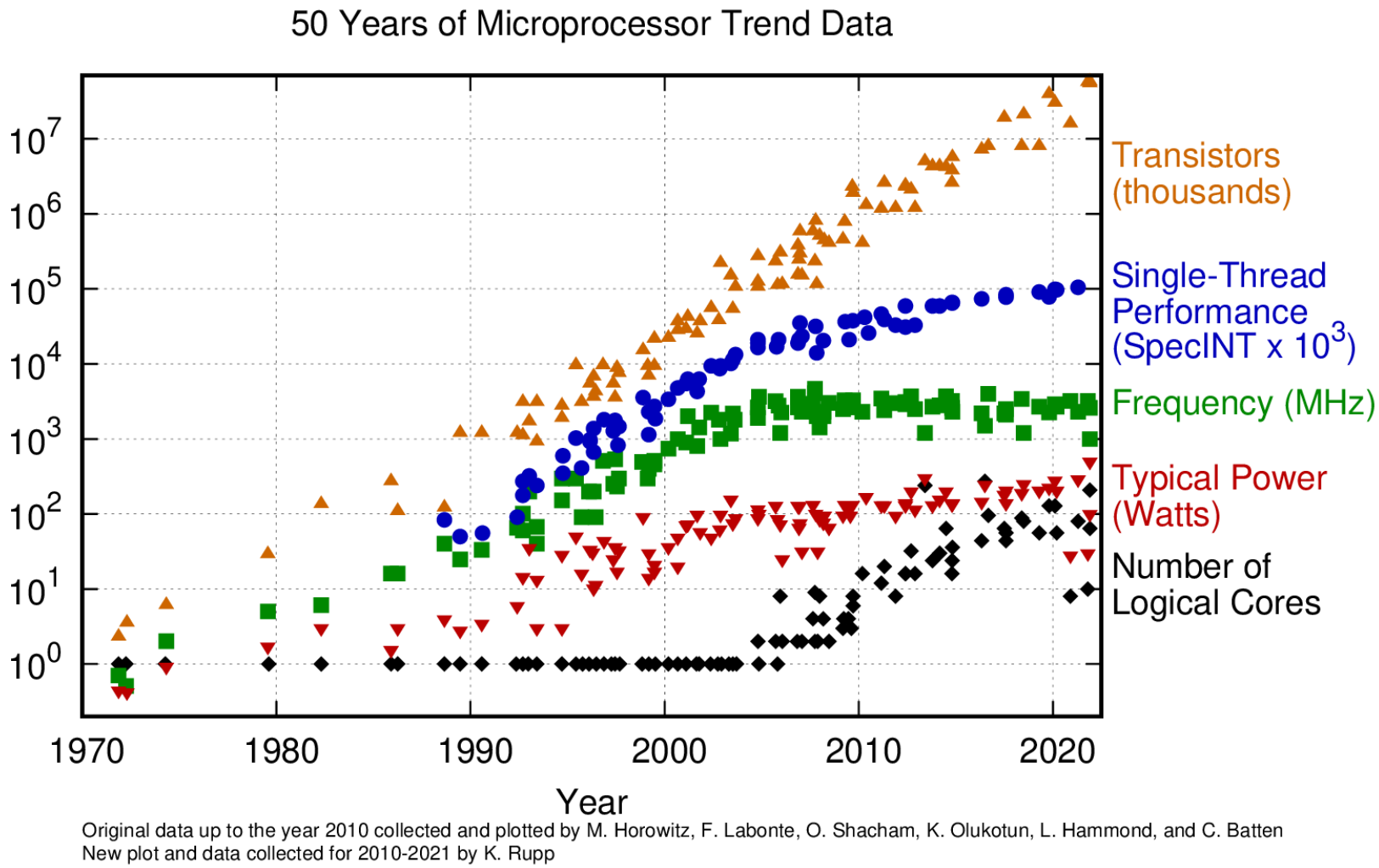
- Η κλιμάκωση και οι αναλογίες που κατέγραψε ο Dennard υπέθεταν ότι το λεγόμενο *ρεύμα διαρροής* είναι αμελητέο.
  - **Έπαψε να είναι αμελητέο γύρω στο 2005 όπου η τεχνολογία ήταν στα 65nm.**
- Περαιτέρω σμίκρυνση αυξάνει εκθετικά το ρεύμα διαρροής και κατά συνέπεια την καταναλισκόμενη ισχύ (αντί να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



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

MYE023

Που υπάρχουν παράλληλοι υπολογιστές;

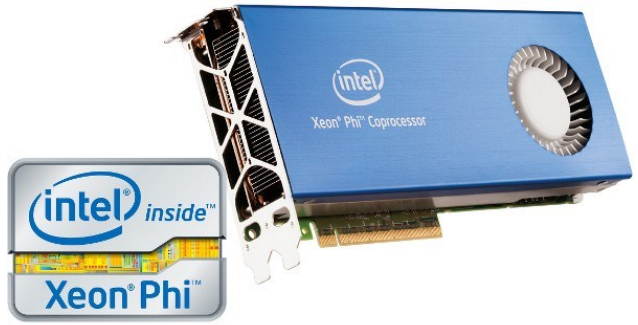
Πρόσφατο (σχετικά) παρελθόν

- 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;

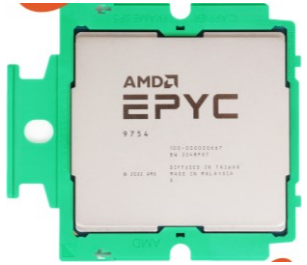
- Πλέον και τα φτηνότερα PC έχουν επεξεργαστή τουλάχιστον διπύρηνο
- Το κινητό μου είναι 8-πύρηνο
- Διπύρηνοι αρχικά, τώρα 4/6/8/12πύρηνοι (Intel, AMD)
  - T1 (Sun Niagara): 8πύρηνοι (με 4-way multithreaded πυρήνες) από 12/2005!
  - T3 (2010): 16πύρηνοι, 8-way multithreaded
- Πολλαπλών πυρήνων (multicore) 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



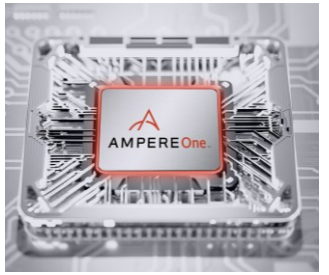
Manycore: Μάλλον είναι ήδη παρόν…

- Μερικοί πρόσφατοι εμπορικοί επεξεργαστές
  - Intel Xeon Platinum 8490H : **60 cores** (2023)
    - Νέα σειρά “Sierra Forest” : **144 and 288 cores** (2024)

  - AMD EPYC 9754 (Bergamo) : **128 cores** (Sept. 2023)



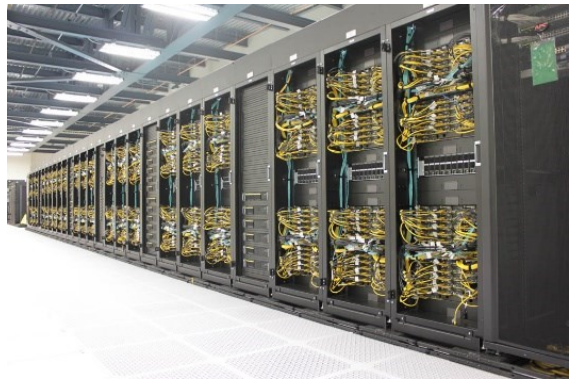
  - Ampere AmpereOne : **192 cores** (May 2023, ARM-based)



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



Top500, Θέση #1

- FRONTIER (November 2023, first installed in 2021)
  - OAK Ridge National Laboratory (USA)
  - **9472 κόμβοι** (1 CPU + 4 GPU)
  - Συνολικός # **cores:** **8,941,568**
  - Συνολική **μνήμη:** **1,212,416 GB**
  - Επεξεργαστές: AMD Epyc 3rd gen (64-cores each) + Radeon MI250X GPUs (220 cores each)
  - ~ 23 MW
  - Κόστος: ~ $600.000.000

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 (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

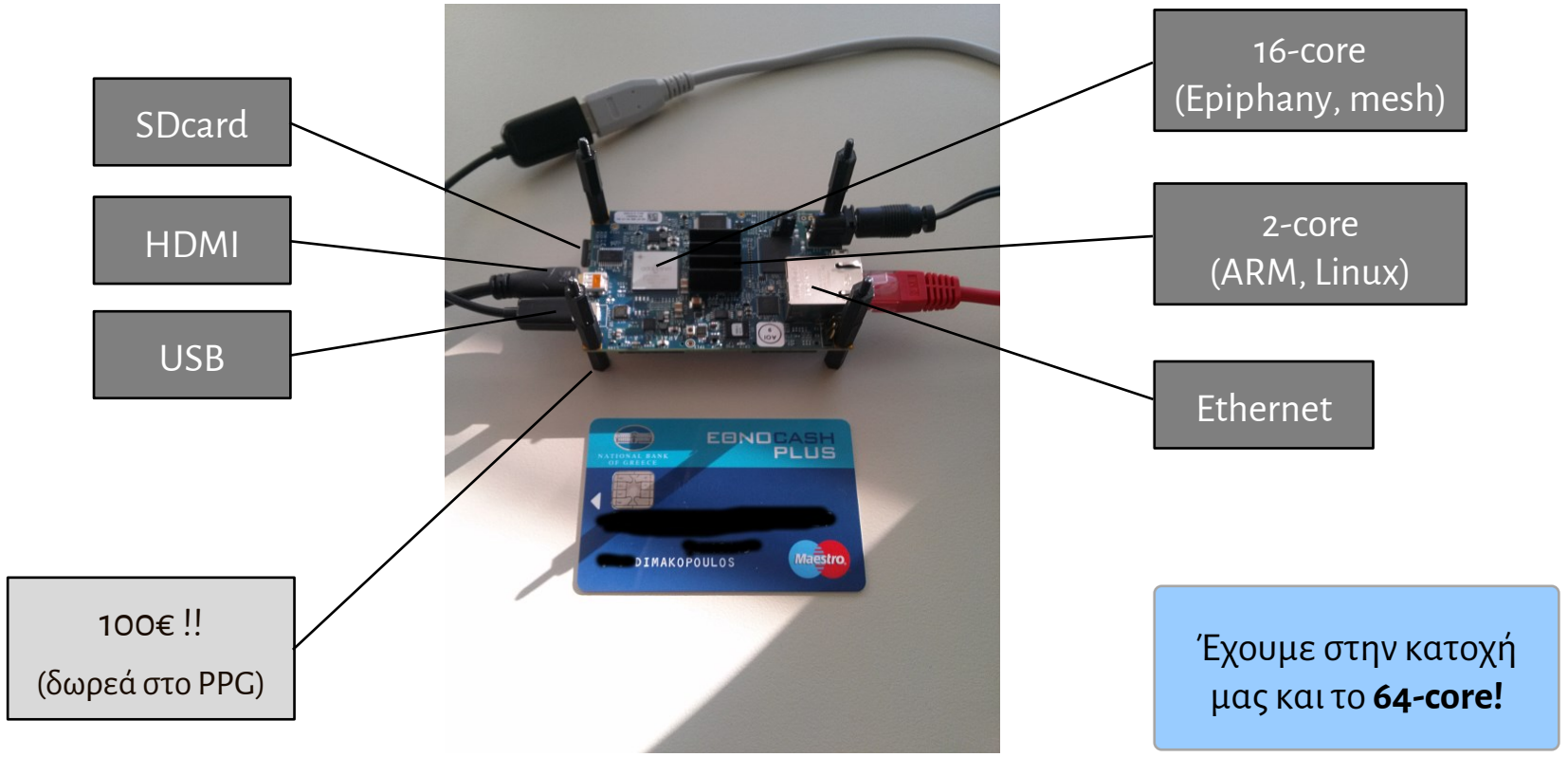
Mobile

- 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*:



Έχουμε στην κατοχή μας και το **64-core!**

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 (January 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*

Γιατί να παρακολουθήσει κανείς αυτό το μάθημα;

- Το μάθημα μιλάει για παράλληλους υπολογιστές
- Οι παράλληλοι υπολογιστές ΔΕΝ είναι πλέον κάτι εξωτικό / σπάνιο / κλπ. Τα πάντα πλέον είναι παράλληλα.
- Όμως παρότι τους έχουμε, ΔΕΝ ξέρουμε πως να τους προγραμματίζουμε
  - Ακόμα και multicore επεξεργαστή να έχετε, αν γράψετε ένα πρόγραμμα όπως τα γράφετε μέχρι τώρα, ΜΟΝΟ το ένα core θα δουλεύει.
- Περιζήτητη γνώση ο παράλληλος προγραμματισμός
- Λίγοι οι «καλοί» που έχουν γνώσεις και πείρα
- Ερευνητικές, εμπορικές και επιχειρηματικές ευκαιρίες

Ύλη του μαθήματος

- Εισαγωγή στους παράλληλους υπολογιστές και τον τρόπο χρήσης τους:
  - Οργάνωση
  - Προγραμματισμός (έμφαση)
- Ενότητες:
  - Εισαγωγή
  - Κλάδος οργάνωσης:
    - Πολυεπεξεργαστές/πολυπύρηνοι κοινόχρηστης μνήμης
    - Πολυεπεξεργαστές/πολυπύρηνοι κατανεμημένης μνήμης
    - Επικοινωνίες
  - Κλάδος προγραμματισμού:
    - Κοινές μεταβλητές (διεργασίες, νήματα)
    - Μεταβίβαση μηνυμάτων (MPI)
  - GPUs (οργάνωση + προγραμματισμός) 

  - Μετρικές και ανάλυση

MYE023

Εισαγωγή

Κεφάλαιο 1

PARALLEL
PROCESSING
GROUP

Η τεχνολογία βελτιώνεται συνεχώς

	ENIAC (1945)	Φτηνό laptop (2022, 199€)
Βάρος (kg)	30.000	1,05
Όγκος (m^3)	70	0,0009
Κατανάλωση (Watt)	140.000	15
Μνήμη (bytes)	200	4.294.967.296
Ρολόι (MHz)	0,005	έως 2.800

... και δεν χωράει σύγκριση!

Άρα, είναι και θα γίνουν ακόμα πιο γρήγοροι οι υπολογιστές …

- Ναι, είναι και θα είναι, για πολλές εφαρμογές.
- Όχι για όλες, όμως.
- Υπάρχουν εφαρμογές που δεν μπορούν να λυθούν ικανοποιητικά ακόμα και με τον πιο πρόσφατο και εξελιγμένο επεξεργαστή.
- Κλασσικό παράδειγμα εφαρμογής με μη ικανοποιητική λύση: *πρόγνωση καιρού* (μήπως χαλάνε μερικές φορές τα μηχανήματα της ΕΜΥ; ☺)
- Ικανοποιητική (δηλ. με καλή ακρίβεια) πρόγνωση καιρού *μπορεί να γίνει* αλλά απαιτεί την επίλυση ασύλληπτα μεγάλων συστημάτων εξισώσεων. Και τεράστιες ποσότητες μνήμης αλλά και τεράστιες υπολογιστικές ταχύτητες είναι αναγκαίες. Τόσες που δεν υπάρχει τυπικό υπολογιστικό σύστημα να τις καλύψει ..
- .. και άρα χρησιμοποιούμε λιγότερες εξισώσεις ⇒ λιγότερο ακριβές μοντέλο ⇒ λιγότερο ακριβής πρόβλεψη ⇒ γίνεται της ΕΜΥ!

Παράδειγμα με αριθμούς:

μελέτη της εξέλιξης του κλίματος της γης

- Το μοντέλο για 10 χρόνια μετά, περιλαμβάνει 10^{16} flops (θέλουμε 10 ημέρες με CPU των 10 Gflops)
- Παράγει 100 Gbytes δεδομένων
- Για καλύτερη μελέτη:

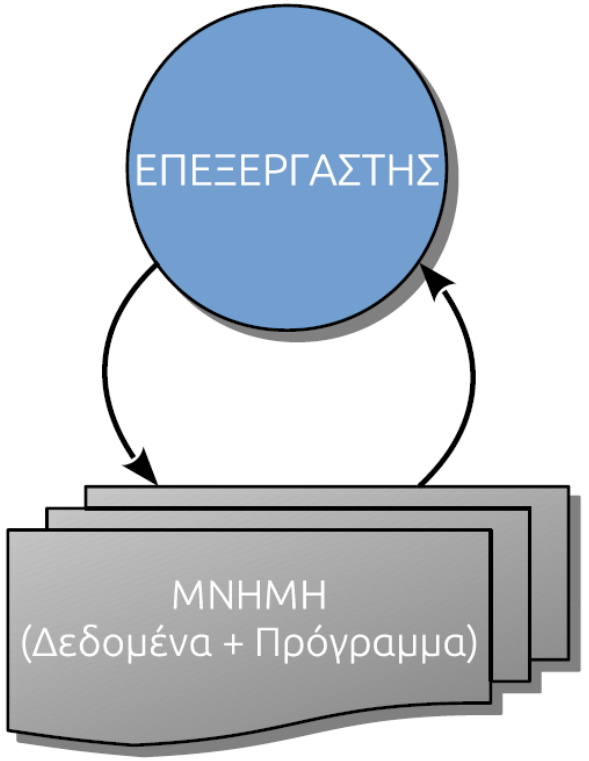
	Τρέχον μοντέλο	**Καλύτερο**	**Κόστος x**
Ανάλυση	100km	10km	x 100 – 1000
Περίοδος υπό μελέτη	10 χρόνια	100 χρόνια	x 10 – 100

- Θέλουμε x 10^3 – 10^5 flops, δηλαδή x 10^3 – 10^5 ημέρες ή περίπου 10.000 – 1.000.000 ημέρες (10.000 = 30 χρόνια!)
- Χωρίς να υπολογίσουμε τον ρυθμό δεδομένων …

Και τι πρέπει να γίνει;

- Πρώτη λύση: ***Βελτίωση της τεχνολογίας***.
  Άρα γρηγορότερα κυκλώματα. Άρα γρηγορότεροι επεξεργαστές.
  - Αυτή είναι ίσως η σημαντικότερη λύση μέχρι τώρα. Η τεχνολογία έχει προχωρήσει με ιλιγγιώδεις ρυθμούς, πέρα από κάθε πρόβλεψη.
  - Θα συνεχίσει έτσι, όμως, και του χρόνου;
    - Όχι - βλ. multicores!
  - Η πιο σωστή απάντηση είναι ότι δεν ξέρουμε ακριβώς τις λεπτομέρειες, αλλά τελικά θα *σταματήσει* αυτή η εξέλιξη, τουλάχιστον με αυτούς τους ρυθμούς. Ή θα αλλάξει ο τρόπος με τον οποίο φτιάχνουμε υπολογιστές (π.χ. κβαντικοί, βιολογικοί)

Το πρόβλημα: σειριακό μοντέλο (ικλασικός Η/Υ)



Γιατί τέτοια απαισιοδοξία;

- Φυσικοί νόμοι: με βάση την ειδική θεωρία της σχετικότητας του Αϊνστάιν, η ταχύτητα των ηλεκτρικών σημάτων είναι μικρότερη ή ίση αυτής του φωτός. Η ταχύτητα του φωτός είναι περίπου 0,3 m/nsec στο κενό, και αρκετά μικρότερη σε χάλκινα καλώδια.

- Άρα, αν η απόσταση μεταξύ CPU και μνήμης είναι 30 cm, θα χρειαστεί (τουλάχιστον) 1 nsec για να πάει η διεύθυνση στη μνήμη και άλλο 1 nsec για να φτάσει το δεδομένο στη CPU (υποθέτοντας ότι η μνήμη έχει ΜΗΔΕΝΙΚΗ καθυστέρηση).

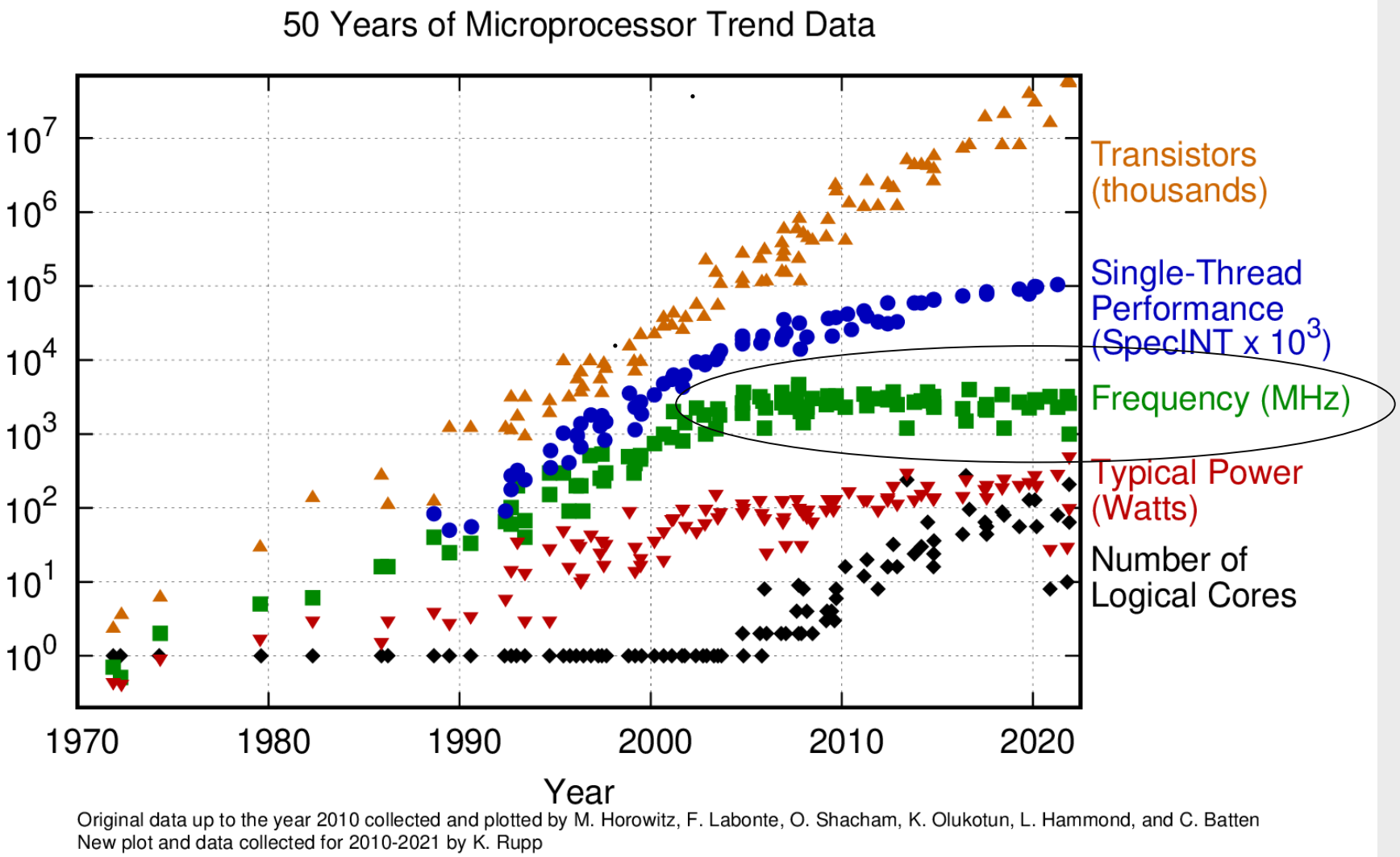
- Άρα η μόνη λύση για να κάνουμε ταχύτερους υπολογιστές είναι να τους κάνουμε εξαιρετικά μικροσκοπικούς!

Μικρότεροι επεξεργαστές

- Ήδη τα μεγέθη των τρανζίστορ και οι αποστάσεις μέσα στα chip είναι πολύ μικρές (2-3 μόρια απαρτίζουν κάποια τμήματα των τρανσίστορ).
  - Περαιτέρω σμίκρυνση θα συνοδευτεί από μοριακά / ατομικά / πυρηνικά φαινόμενα που δεν ξέρουμε ακόμα πώς να τα χειριστούμε.

- Μικρά μεγέθη + αύξηση συχνότητας λειτουργίας (για ταχύτητα) οδηγεί σε αύξηση της κατανάλωσης.
  - Από τα 4Ghz περίπου πριν 15 χρόνια, οι συχνότητες έπεσαν στα 1.5-2.5GHz ...
  - ... και έγιναν πολυπύρηνοι (multicore)

Επεξεργαστές & συχνότητες



Υπάρχει άλλη λύση;

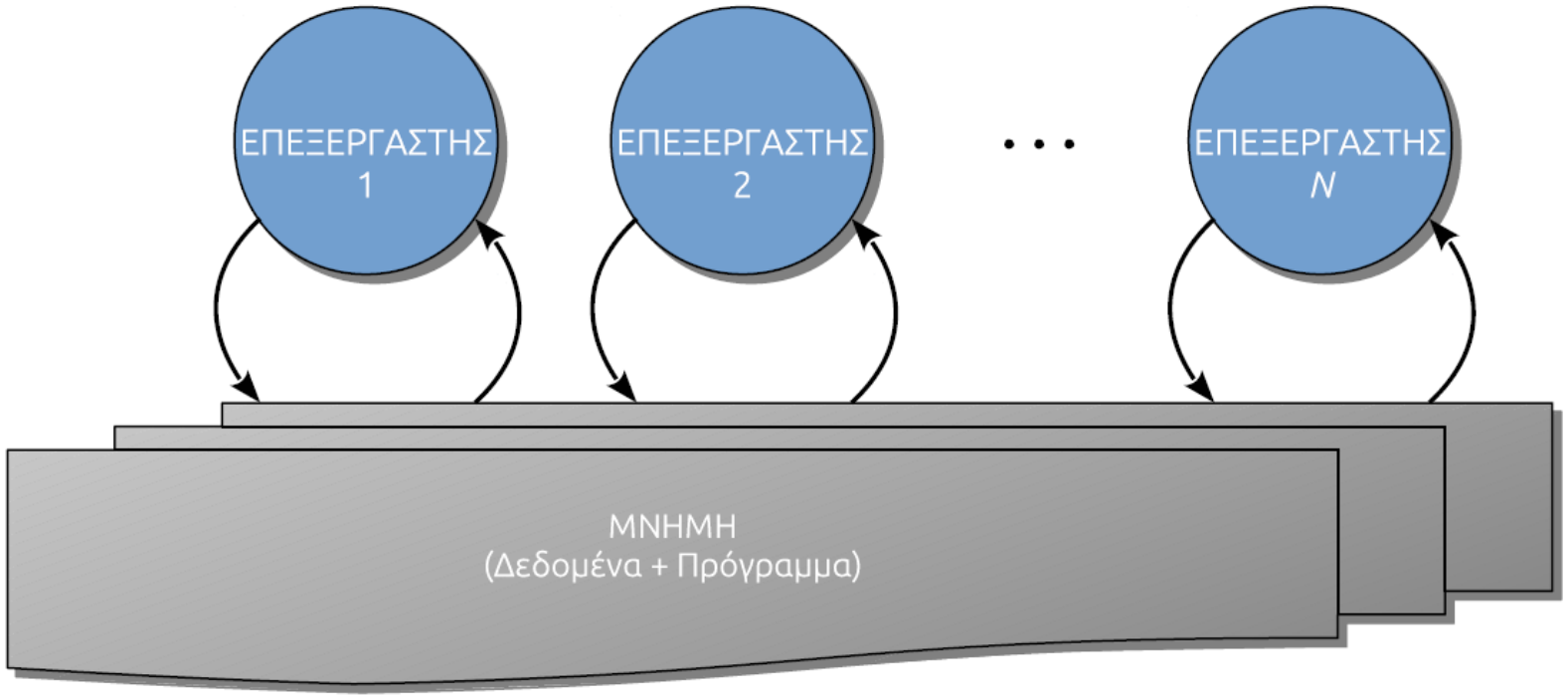
- Δεύτερη (και μοναδική) λύση: ***Παραλληλισμός***.
  Πολλαπλές εντολές εκτελούνται παράλληλη (ταυτόχρονα).
  Άρα γρηγορότερη ολοκλήρωση των προγραμμάτων.

  – Παραλληλισμός σε επίπεδο εντολών (ILP: διοχέτευση, υπερβαθμωτές αρχιτεκτονικές κλπ) καλός αλλά λίγος (αύξηση ταχύτητας x10, *το πολύ*)

  – *Μοναδική λύση είναι η χρήση καθαρά παράλληλων υπολογιστών*: υπολογιστές που διαθέτουν πολλούς επεξεργαστές, διασυνδεδεμένους μεταξύ τους, που συνεργάζονται για την ταυτόχρονη εκτέλεση τμημάτων μίας εφαρμογής.

  – Π.χ. σύστημα με 1000 επεξεργαστές έχει τη δυνατότητα να εκτελέσει ένα πρόγραμμα 1000 φορές πιο γρήγορα (από ότι ένα σύστημα με 1 επεξεργαστή).

Παράλληλος υπολογιστής



Βασικά ερωτήματα

- Πόσοι επεξεργαστές;
  - λίγοι (2 – 10)
  - πολλοί (εκατοντάδες έως δεκάδες χιλιάδες)
- Πώς συνδέονται / επικοινωνούν / συνεργάζονται;
  - κοινής μνήμης (πολυεπεξεργαστές)
  - κατανεμημένης μνήμης (πολυϋπολογιστές)
- Πως «σπάει» ένα πρόβλημα για να μοιραστεί στους επεξεργαστές;
  - λεπτόκκοκος / χονδρόκοκκος παραλληλισμός (fine/coarse grain)
  - ισοκατανομή φόρτου (load balancing)
- Πόσο διαφορετικός είναι ο προγραμματισμός τους;
  - τρία βασικά μοντέλα …

Που υπάρχει παραλληλισμός;

```
for (i = 0; i < k; i = i+1)
    c[i] = a[i]+b[i];
```

- Εφαρμογές (grand challenges)
  - συμπεριφορά σωματιδίων (π.χ. δυναμική των μορίων)
  - μελέτη των κυμάτων των ωκεανών και πρόβλεψη καιρού
  - σεισμικά μοντέλα
  - σχεδιασμός VLSI κυκλωμάτων με τη βοήθεια του υπολογιστή (CAD)
  - εξέλιξη των γαλαξιών
  - ...

Παράδειγμα: το πρόβλημα των Ν σωμάτων

- N-body problem: συλλογή από Ν σώματα (ηλεκτρόνια, μόρια, πλανήτες, σύνεφα, κλπ)
- Δυνάμεις μεταξύ τους (βαρυτικές, ηλεκτρικές, μαγνητικές, κλπ) που τους αλλάζουν την κατάσταση (π.χ. τη θέση τους στο χώρο)
- Μελέτη (προσομοιώσεις) για να δούμε σε ποια κατάσταση θα είναι μετά από κάποιο χρονικό διάστημα

- Παράδειγμα: *εξέλιξη γαλαξιών* (σώματα = πλανήτες / αστέρια κλπ., δυνάμεις = βαρυτικές, κατάσταση = η θέση των ουρανίων σωμάτων)

Εξέλιξη γαλαξιών

(απλά, γνωστά πράγματα… ☺)

$$\vec{f}_{ij} = \frac{G m_i m_j (\vec{x}_i - \vec{x}_j)}{\left| \vec{x}_i - \vec{x}_j \right|^3}$$

δύναμη μεταξύ σώματος *i* και *j* (1)

$$\vec{F}_i = \sum_{j=1}^{N} \vec{f}_{ij}.$$

συνολική δύναμη επάνω στο σώμα i (2)

$$\vec{F}_i = m_i \frac{d^2 \vec{x}_i}{dt^2}.$$

σχέση δύναμης-μετατόπισης (3)

Σειριακό πρόγραμμα

```
/* [0 .. MAX_TIME]: the time interval we want to study
 * f(i,j):          force between i and j (using (1))
 * xnew(k,F[k]):    new position of k (using (3))
 */
for (t = 0; t < MAX_TIME; t = t + deltat)
{
  for (i = 0; i < N; i++)
  {
    F[i] = zero();      /* calculate sum (using 2) */
    for (j = 0; j < N; j++)
      F[i] = add( F[i], f(i,j) );
  }
  for (k = 0; k < N; k++)
    xnew( k, F[k] );    /* calculate new positions */
}
```

Οι υπολογισμοί των F[ ] ανεξάρτητοι

```
/* [0 .. MAX_TIME]: the time interval we want to study
 * f(i,j):          force between i and j (using 1)
 * xnew(k,F[k]):    new position of k (using 3)
 */
for (t = 0; t < MAX_TIME; t = t + deltat)
{
  DO EACH i-ITERATION IN PARALLEL:
  for (i = 0; i < N; i++)
  {
    F[i] = zero();      /* calculate sum (using 2) */
    for (j = 0; j < N; j++)
      F[i] = add( F[i], f(i,j) );
  }
  for (k = 0; k < N; k++)
    xnew( k, F[k] );    /* calculate new positions */
}
```

Οι υπολογισμοί των x[ ] ανεξάρτητοι

```
/* [0 .. MAX_TIME]: the time interval we want to study
 * f(i,j):          force between i and j (using 1)
 * xnew(k,F[k]):    new position of k (using 3)
 */
for (t = 0; t < MAX_TIME; t = t + deltat)
{
  DO EACH i-ITERATION IN PARALLEL:
  for (i = 0; i < N; i++)
  {
    F[i] = zero();      /* calculate sum (using 2) */
    for (j = 0; j < N; j++)
      F[i] = add( F[i], f(i,j) );
  }
  DO EACH k-ITERATION IN PARALLEL:
  for (k = 0; k < N; k++)
    xnew( k, F[k] );    /* calculate new positions */
}
```

Οι υπολογισμοί των x[ ] εξαρτώνται από τα F[ ] !!

```
/* [0 .. MAX_TIME]: the time interval we want to study
 * f(i,j):          force between i and j (using 1)
 * xnew(k,F[k]):    new position of k (using 3)
 */
for (t = 0; t < MAX_TIME; t = t + deltat)
{
  DO EACH i-ITERATION IN PARALLEL:
  for (i = 0; i < N; i++)
  {
    F[i] = zero();      /* calculate sum (using 2) */
    for (j = 0; j < N; j++)
      F[i] = add( F[i], f(i,j) );
  }

  WAIT TILL ALL F[] ELEMENTS HAVE BEEN CALCULATED

  DO EACH k-ITERATION IN PARALLEL:
  for (k = 0; k < N; k++)
    xnew( k, F[k] );    /* calculate new positions */
}
```

Μεγάλη προσοχή!!

```
/* [0 .. MAX_TIME]: the time interval we want to study
 * f(i,j):          force between i and j (using 1)
 * xnew(k,F[k]):    new position of k (using 3)
 */
for (t = 0; t < MAX_TIME; t = t + deltat)
{
  DO EACH i-ITERATION IN PARALLEL:
  for (i = 0; i < N; i++)
  {
    F[i] = zero();      /* calculate sum (using 2) */
    for (j = 0; j < N; j++)
      F[i] = add( F[i], f(i,j) );
  }

  WAIT TILL ALL F[] HAVE BEEN CALCULATED

  DO EACH k-ITERATION IN PARALLEL:
  for (k = 0; k < N; k++)
    xnew( k, F[k] );    /* calculate new positions */

  WAIT TILL ALL x[] ELEMENTS HAVE BEEN CALCULATED
}
```

Καλύτερος βρόχος j

Λόγω συμμετρίας των δυνάμεων:

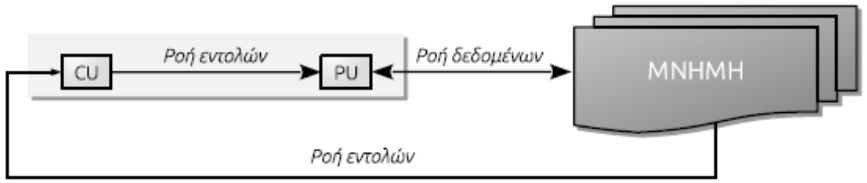
```
for (j = 0; j < i; j++)
{
  g = f(i,j);
  F[i] = add( F[i], g );
  F[j] = subtract( F[j], g );
}
```

Προβλήματα, πλέον!

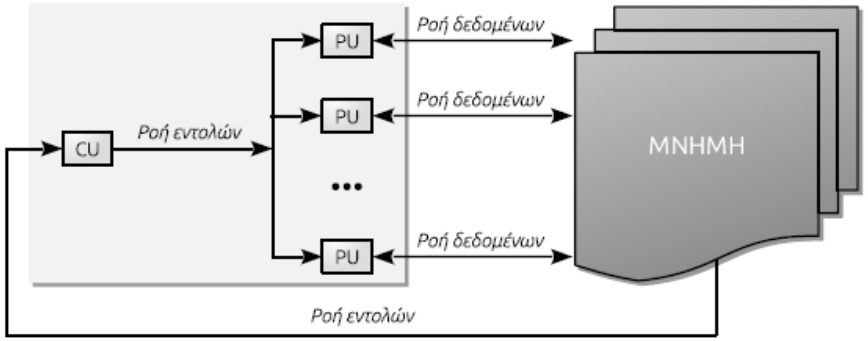
- Εξαρτήσεις μεταξύ επαναλήψεων
- Ανισοκατανομή φόρτου

Παράλληλες αρχιτεκτονικές

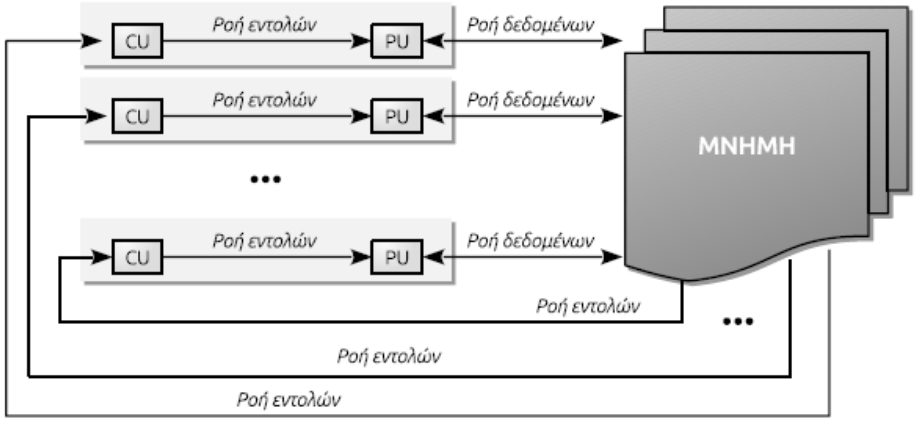
- Κατά Flynn (1972)
  - SISD (Single-Instruction, Single-Data)
  - SIMD (Single-Instruction, Multiple-Data)
  - MIMD (Multiple-Instruction, Multiple-Data)



(α) SISD



(β) SIMD

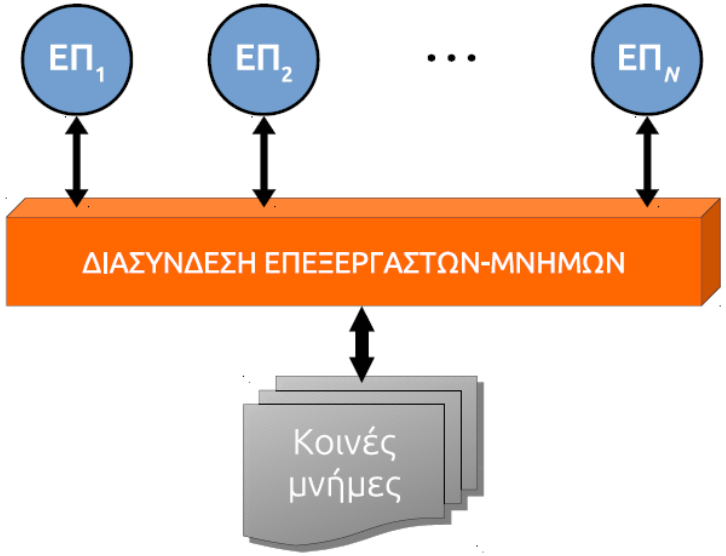


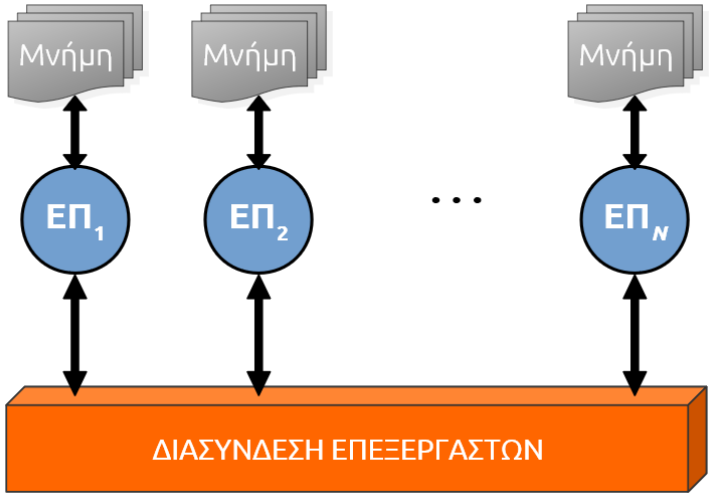
(γ) MIMD

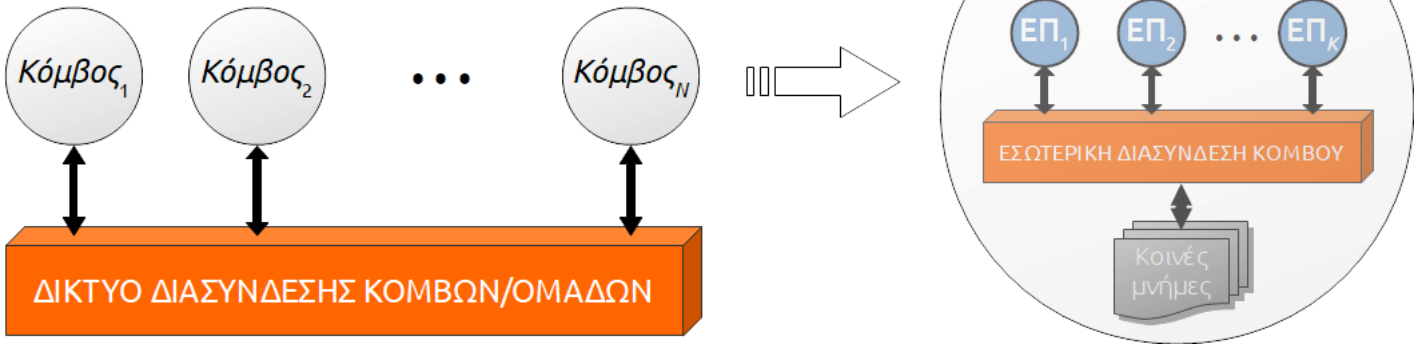
Υπολογιστές SIMD

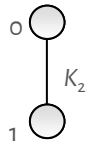
- Πρωτοπόρα συστήματα (’70 & ’80)
  - Ακριβοί οι αυτόνομοι επεξεργαστές
  - Κυρίως ερευνητικά, εξειδικευμένα μηχανήματα
  - Illiac IV
  - Επηρέασαν τα κατοπινά συστήματα
  - **«επεξεργαστές πίνακα» (array processors)**
- Πλέον μόνο για ειδικού σκοπού συστήματα (π.χ. systolic arrays)
- Επανεμφάνιση ως:
  - SWAR (SIMD-Within-A-Register: MMX, SSE, κλπ)
  - GPUs!

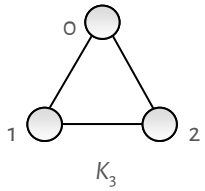
Πολυεπεξεργαστές κοινής & κατανεμημένης μνήμης (MIMD)

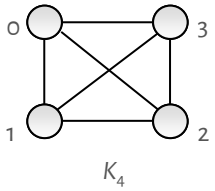




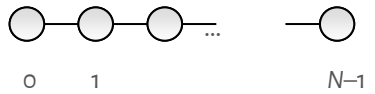




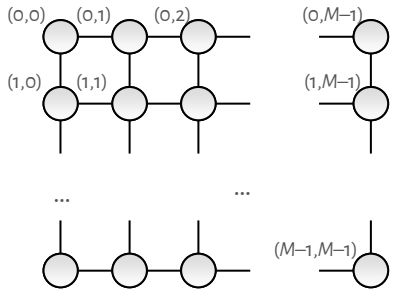




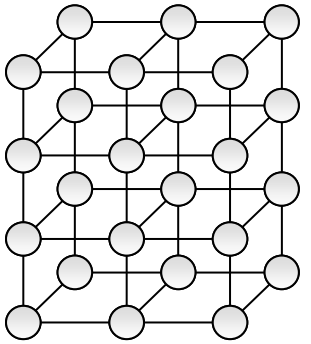
Πλήρεις γράφοι



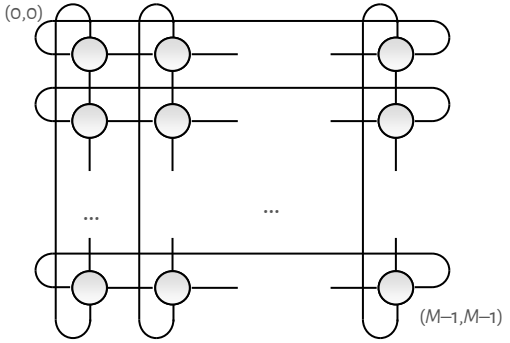
Γραμμικός γράφος



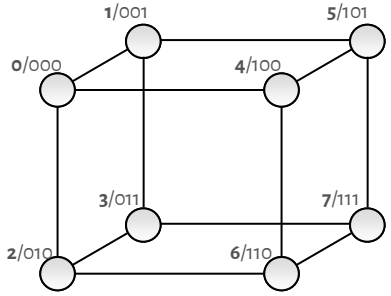
Πλέγμα $M \times M$



Πλέγμα $4 \times 3 \times 2$



Torus $M \times M$



Τρισδιάστατος κύβος

Παράλληλος προγραμματισμός (έμμεσος / άμεσος)

- Ο *έμμεσος* δεν δίνει καλά αποτελέσματα
  - «Παραλληλοποιητικοί» μεταφραστές που από σειριακό πρόγραμμα παράγουν αυτόματα (ή με κάποια βοήθεια από τον προγραμματιστή) ένα ισοδύναμο παράλληλο πρόγραμμα.
  - Μόνο για loops με αυστηρά όρια
- Ο *άμεσος* είναι που μας ενδιαφέρει, αλλά πιο «δύσκολος»:
  - Εξάρτηση από αρχιτεκτονική
  - Ασυμβατότητα μηχανών
  - Πολλά προγραμματιστικά μοντέλα
    - Παραλληλισμού δεδομένων (data parallelism)
    - Κοινού χώρου διευθύνσεων (shared address space)
    - Μεταβίβασης μηνυμάτων (message passing)

Βασική μεθοδολογία παραλληλο-ποίησης

- Διαχωρισμός (διάσπαση + ανάθεση) + τοποθέτηση (ενορχήστρωση + αντιστοίχιση)

- Διαχωρισμός (partitioning)
  - Διάσπαση (decomposition) σε εργασίες (tasks)
    - *κόκκος παραλληλίας*
  - Ανάθεση (assignment) σε νήματα/διεργασίες (processes)
    - *ισοκατανομή φόρου, μείωση επικοινωνιών*

- Τοποθέτηση (placement)
  - Ενορχήστρωση διεργασιών
    - *καθορισμός σειράς εκτέλεσης, προγραμματιστικό μοντέλο*
  - Αντιστοίχιση (mapping) σε επεξεργαστές
    - *στατική, δυναμική*

Διάσπαση και ανάθεση

Διάσπαση:

```
1.    F[i] = add( F[i], f(i,j) );

2.    for (j = 0; j < N; j = j+1)
        F[i] = add( F[i], f(i,j) );
```

Ανάθεση σε διεργασία:

```
Process-k :
{
  for (i = (k-1)*N/P; i < k*N/P; i++)
  {
    F[i] = zero();
    for (j = 0; j < N; j = j+1)
      F[i] = add( F[i], f(i,j) );
  }
}
```