Ιόνιο Πανεπιστήμιο – Τμήμα Πληροφορικής
Παράλληλος Προγραμματισμός
2025-26

Εισαγωγή

(Βασικές έννοιες παράλληλου υπολογισμού)

https://mixstef.github.io/courses/parprog/

Μ.Στεφανιδάκης

Παράλληλη επεξεργασία

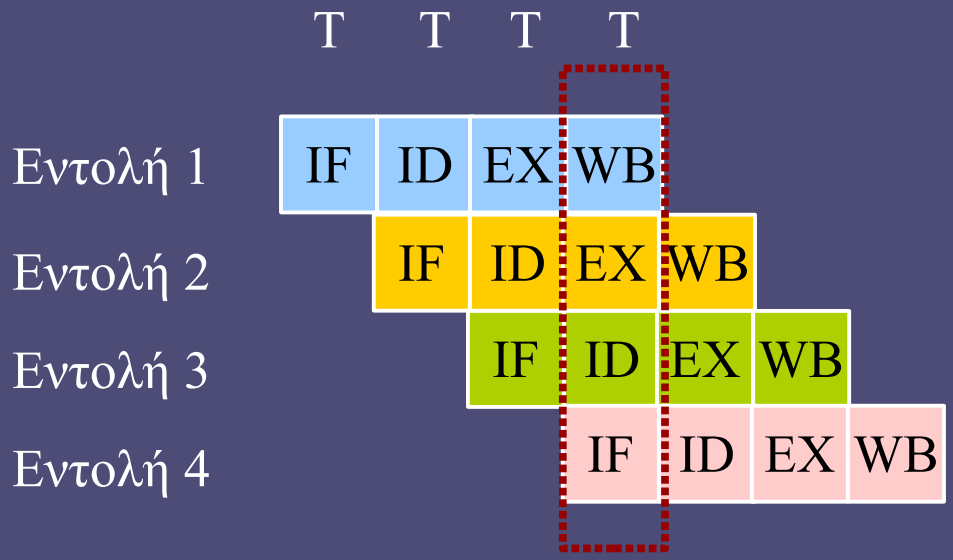
- Ταυτόχρονη εκτέλεση διεργασιών
  - Κώδικας που εκτελείται την ίδια στιγμή σε διαφορετικές υπολογιστικές μονάδες (πόρους επεξεργασίας)
- Γιατί είναι επιθυμητή;
  - Επίλυση υπολογιστικά δύσκολων προβλημάτων
  - Αλλά και απλούστερων προβλημάτων με πολύ μεγάλο όγκο δεδομένων εισόδου
    - Μοντελοποίηση φυσικών φαινομένων
    - Τεχνητή νοημοσύνη
    - Βιοιατρική
    - κ.λ.π.

Η αναγκαιότητα της παράλληλης επεξεργασίας

- Το τέλος της «κούρσας των GHz»
  - Στις αρχές της δεκαετίας του 2000
  - Εμπόδια στα οφέλη από την αύξηση της συχνότητας του ρολογιού
    - Υπέρμετρη κατανάλωση ενέργειας – αδυναμία απαγωγής θερμότητας
    - Οι αλληλοεξαρτήσεις μεταξύ εντολών τονίζονται – μείωση της προσδοκώμενης αύξησης της απόδοσης
  - Το σειριακό πρόγραμμα δεν γίνεται πλέον γρηγορότερο «αυτόματα» με την πάροδο του χρόνου
    - “Free ride is over!”
- Πώς θα χρησιμοποιηθεί η αφθονία τρανζίστορ;
  - Παράλληλη επεξεργασία σε χαμηλότερες συχνότητες

Παρεχόμενη παραλληλία: pipelines

- **Παραλληλισμός σε επίπεδο εντολών (ILP)**
  - Μια βασική τεχνική παράλληλης επεξεργασίας
    - Την ίδια στιγμή εκτελούνται λειτουργίες πολλαπλών εντολών μηχανής
    - Ιδανικά, σε κάθε κύκλο ρολογιού (περίοδος Τ) ολοκληρώνεται μια εντολή



Παρεχόμενη παραλληλία: superscalar CPUs

- Εκκίνηση περισσότερων από μια εντολή σε κάθε κύκλο ρολογιού
  - Την εποχή της «κούρσας των GHz»
  - Υπάρχουν πολλαπλά pipelines
    - Η επιλογή γίνεται αυτόματα από την ΚΜΕ που παρακολουθεί τις εντολές σε ορισμένο βάθος χρόνου (“instruction window”)
  - Τεχνικές για την αύξηση των εντολών που μπορούν να εκτελεστούν παράλληλα
    - Εκτέλεση εκτός σειράς (out of order execution)
    - Μετονομασίες καταχωρητών (register renaming) – οι μοντέρνες ΚΜΕ έχουν περισσότερους «εσωτερικούς/φυσικούς» καταχωρητές (150+)
    - Πρόβλεψη διακλαδώσεων (branch prediction)

Παρεχόμενη παραλληλία: vector instructions

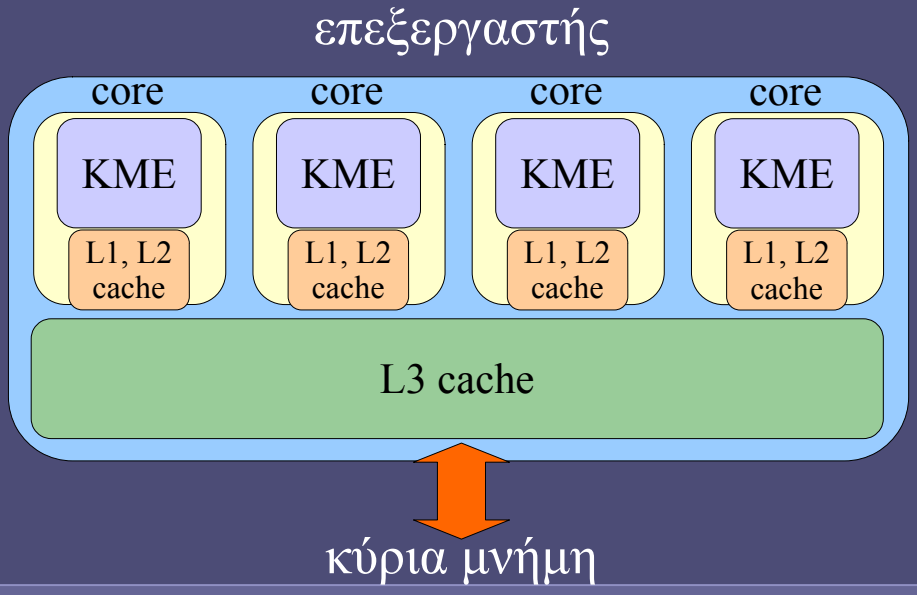
- Εντολές με πολύ μεγάλο εύρος δεδομένων
  - Την εποχή της «κούρσας των GHz»
  - Η ίδια λειτουργία σε πολλαπλά δεδομένα (SIMD)
    - Μονάδες εκτέλεσης πράξεων μεγάλου εύρους (π.χ. 512 bits)
  - “Streaming instructions”
    - Αρχικά για δεδομένα multimedia

Παρεχόμενη παραλληλία: SMT

- Simultaneous Multithreading
  - Το ξέρουμε καλύτερα με τον όρο marketing: “hyperthreading”
  - Παραλληλισμός σε επίπεδο thread (TLP)
    - Η «κούρσα των GHz» φτάνει στο τέλος της
  - Η ΚΜΕ μοιράζει τις μονάδες εκτέλεσης μεταξύ 2 (ή 4 ή 8..) διεργασιών
    - Κρατώντας ξεχωριστή κατάσταση (καταχωρητές) ανά διεργασία
    - Στο λειτουργικό σύστημα φαίνονται ως ανεξάρτητοι «λογικοί» πυρήνες

Παρεχόμενη παραλληλία: multicore

- Περισσότεροι πυρήνες (cores) στον επεξεργαστή
  - Η «κούρσα των GHz» έχει τελειώσει οριστικά
  - Παραλληλισμός σε επίπεδο thread (TLP)
  - Αυξάνεται η πίεση στη σύνδεση με την κύρια μνήμη – προσθήκη μεγαλύτερης ιεραρχίας κρυφών μνημών



Παρεχόμενη παραλληλία: accelerators

- Μονάδες υπολογισμού πέρα από τον τυπικό επεξεργαστή
  - GPUs (παράλληλοι υπολογισμοί)
  - Tensor cores (εφαρμογές AI)
  - και λοιπές μονάδες επιτάχυνσης υπολογισμών
- Διασύνδεση μέσω διαύλων μεταφοράς δεδομένων υψηλής ταχύτητας
- Πληθώρα εξειδικευμένων αρχιτεκτονικών
  - Στην εποχή του AI

Η ταξινόμηση κατά Flynn (1972)

- Single Instruction Single Data (SISD)
  - Ο παραδοσιακός υπολογιστής χωρίς κανένα είδος παραλληλίας
  - Το κλασσικό «μοντέλο von Neumann»
  - Κάθε εντολή εκτελείται σειριακά σε μια μοναδιαία (single) ποσότητα δεδομένων
  - Μία και μοναδική λειτουργία σε κάθε χρονική στιγμή
  - Η απόδοση των εφαρμογών εξαρτάται από την ταχύτητα της επεξεργασίας

Η ταξινόμηση κατά Flynn (συνέχεια)

- **Single Instruction Multiple Data (SIMD)**
  - Η ίδια (single) λειτουργία (εντολή) εκτελείται σε πολλαπλά (multiple) δεδομένα παράλληλα
  - Το υλικό διαθέτει έναν και μοναδικό Program Counter και πολλαπλές μονάδες εκτέλεσης πράξεων
  - Τα περισσότερα εμπορικά συστήματα σήμερα διαθέτουν κάποια χαρακτηριστικά SIMD (αλλά όχι μόνον)
    - 4, 8 ή 16-πλες μονάδες υπολογισμού (εντολές streaming σε συμβατικές ΚΜΕ)
    - Χιλιάδες μονάδες υπολογισμού (streaming cores σε GPUs)
  - Υπερυπολογιστές (υψηλό κόστος)
    - Εξειδικευμένοι vector processors (ευρείς αγωγοί δεδομένων, από τη μνήμη έως τους καταχωρητές και τις μονάδες υπολογισμού)

Η ταξινόμηση κατά Flynn (συνέχεια)

- **Multiple Instruction Multiple Data (MIMD)**
  - Ξεχωριστές ακολουθίες εντολών εκτελούνται σε ξεχωριστές ομάδες δεδομένων
    - Πολλαπλές ΚΜΕ που εκτελούν ανεξάρτητα προγράμματα
  - Πολλαπλοί επεξεργαστικοί κόμβοι
    - Επεξεργαστές πολλών πυρήνων (Multicores)
    - Συνδυασμοί CPU + GPU/άλλων συνεπεξεργαστών στο ίδιο σύστημα
    - Κατανεμημένα συστήματα, συνδεδεμένα με κάποιο είδος δικτύου

Συστήματα κοινής μνήμης

- **Shared Memory Systems**
  - Συστήματα MIMD όπου όλοι οι επεξεργαστικοί κόμβοι «βλέπουν» μια κοινή και ενιαία μνήμη
    - Οι επεξεργαστές βρίσκονται μέσα σε μοναδικό ενιαίο σύστημα
  - Οι εκτελούμενες παράλληλες διεργασίες
    - Έχουν πρόσβαση στα διαμοιραζόμενα δεδομένα
    - Και χρησιμοποιούν την κοινή μνήμη για **συγχρονισμό**
  - Οι κρυφές μνήμες μπορούν να δημιουργήσουν πρόβλημα στη **συνοχή** των δεδομένων
    - Όταν διαφορετικοί επεξεργαστικοί κόμβοι (με διαφορετικές κρυφές μνήμες) τροποποιούν τα ίδια δεδομένα

Είδη κοινής μνήμης

- Uniform Memory Access (UMA)
  - Η κοινή μνήμη είναι φυσικά ενιαία
    - Όλοι οι επεξεργαστικοί κόμβοι την προσπελαύνουν με το ίδιο κόστος (σχήμα Symmetric Multiprocessor, SMP)
    - Η σύνδεση με την κύρια μνήμη αποτελεί σημείο συνωστισμού
- Non-Uniform Memory Access (NUMA)
  - Κάθε επεξεργαστής του συστήματος έχει τη δική του τοπική μνήμη
    - Για τις υπόλοιπες μνήμες βασίζεται στην ενδο-επικοινωνία μεταξύ επεξεργαστών
    - Υπάρχει και εδώ θέμα διατήρησης της συνοχής των δεδομένων μεταξύ κρυφών μνημών
    - Όσο κάθε επεξεργαστής επικοινωνεί με τη «δική» του μόνο τοπική μνήμη, ο συνωστισμός είναι ελάχιστος

Συστήματα κατανεμημένης μνήμης

- Distributed memory systems
  - Αποτελούνται από ανεξάρτητα επεξεργαστικά συστήματα διασυνδεμένα μέσω ενός δικτύου
- Δεν υπάρχει η έννοια της «ενιαίας» και «κοινής» μνήμης
  - Τα διαμοιραζόμενα δεδομένα πρέπει να μεταφέρονται ανάμεσα στους κόμβους μέσω μηνυμάτων
    - Message passing APIs

Ο ρόλος του λογισμικού

- Η αποδοτική παράλληλη επεξεργασία βασίζεται στη συνεργασία
  - Λειτουργικού συστήματος
  - Μεταγλωττιστή
  - Αλγορίθμων και Δομών δεδομένων
  - Και του κώδικά μας ☺
- Και τη γνώση των χαρακτηριστικών του υλικού (hardware)
  - Εγκαταλείπουμε την αρχή «αποσύνδεσης του προγράμματος από το υλικό εκτέλεσης»;
  - Ή χρησιμοποιούμε νέα frameworks που κρύβουν τις λεπτομέρειες του παραλληλισμού στο υποκείμενο hardware;

Εμπόδια στη διαδικασία παραλληλισμού

- Μπορούμε πάντα να μετατρέψουμε αποδοτικά ένα σειριακό πρόγραμμα στο αντίστοιχο παράλληλο;
  - Αλληλεξαρτήσεις δεδομένων
    - Τα διάφορα στάδια εξαρτώνται από τιμές προηγούμενου υπολογισμού
    - Αναμονή για υπολογισμό εισόδων
  - Προσπέλαση μνήμης
    - Πολλά προγράμματα (και αλγόριθμοι) έχουν απόδοση που εξαρτάται από την επικοινωνία με τη μνήμη
    - Ο χρόνος μεταφοράς δεδομένων επισκιάζει κάθε όφελος παραλληλισμού
    - Ο τρόπος προσπέλασης μνήμης επηρεάζει τον χρόνο μεταφοράς

Απόδοση παράλληλων προγραμμάτων

$$\text{Speedup } S_P = \frac{\text{χρόνος σειριακής εκτέλεσης}}{\text{χρόνος παράλληλης εκτέλεσης}}$$

- S_p: επιτάχυνση (speedup) με p επεξεργαστικούς κόμβους
  - Στην καλύτερη περίπτωση $S_p = p$
  - Μερικές φορές, για ανεξάρτητους λόγους (περισσότεροι πόροι μνήμης, διαφορετικός αλγόριθμος…) προκύπτει $S_p > p$ (superlinear speedup) → δεν μοντελοποιούμε σωστά το σύστημα
- Επίσης: αποδοτικότητα (efficiency) $E_p = S_p / p$

Παράγοντες περιορισμού του speedup

$$\text{Speedup } S_P = \frac{t_s}{f \times t_s + (1-f)t_s/p} = \frac{1}{f + (1-f)/p}$$

- Κάθε αλγόριθμος περιέχει ένα ποσοστό εργασίας f που πρέπει να εκτελεστεί σειριακά
  - Επιβάρυνση παραλληλισμού (overhead)
  - Επικοινωνία και συγχρονισμός επεξεργαστικών κόμβων για την ανταλλαγή δεδομένων
  - $S_P \rightarrow 1/f$ όταν $p \rightarrow \infty$ («νόμος» του Amdahl)

Βιβλιογραφία

- Michael McCool, James Reinders, and Arch Robison. 2012. *Structured Parallel Programming: Patterns for Efficient Computation* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Peter Pacheco. 2011. An Introduction to Parallel Programming (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- John L. Hennessy and David A. Patterson. 2003. Computer Architecture: A Quantitative Approach (3 ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.