Παραλληλισμός και Αρχιτεκτονική

25 / 04 / 2023

1

Ατζέντα

- Γιατί παραλληλισμός ?
- Διαφορετικά μοντέλα για παράλληλη εκτέλεση – σχετικές αρχιτεκτονικές
- Νέα προβλήματα και προκλήσεις που εισάγει ο παραλληλισμός
  - Επικοινωνία
  - Επεκτασιμότητα

2

Αιτίες για Παραλληλισμό

- Για οποιοδήποτε στοιχείο επεξεργασίας, εν γένη, περισσότερες μονάδες/στοιχεία $\Rightarrow$ μεγαλύτερες επιδόσεις

- Προβλήματα – Προκλήσεις
  - Ο παραλληλισμός περιορίζεται από την επικοινωνία, όπως λ.χ., κόστος επικοινωνίας μεταξύ chip >> κόστος επικοινωνίας εντός chip
  - Ποικιλία επιλογών / προκλήσεις σχεδίασης: N μικροί πυρήνες σε 1 chip ή N μεγάλοι πυρήνες σε N chip ?
  - N επεξεργαστές συχνά είναι $\neq$ N $\times$ επίδοση του 1 επεξεργαστή
  - Ο παράλληλος προγραμματισμός συχνά είναι δύσκολος
  - Μία εφαρμογή σε ποιόν τύπο παράλληλης αρχιτεκτονικής έχει καλύτερες επιδόσεις?

3

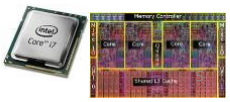
Παράλληλη Αρχιτεκτονική

- Ένας παράλληλος υπολογιστής είναι μία συλλογή από επεξεργαστικά στοιχεία που συνεργάζονται για να επιλύσουν γρήγορα μεγάλα προβλήματα
- Βασικά προβλήματα:
  - Δέσμευση – εκχώρηση πόρων: πόσα είναι τα στοιχεία, πόσο ισχυρά, πόση μνήμη...
  - Πρόσβαση στα δεδομένα, επικοινωνία, συγχρονισμός
    - Πως επικοινωνούν και συνεργάζονται τα επεργαστικά στοιχεία
    - Πως μεταφέρονται τα δεδομένα μεταξύ των στοιχείων
  - Επιδόσεις και επεκτασιμότητα
    - Πως μεταφράζονται όλα τα παραπάνω σε επιδόσεις
    - Πως επιδρά η επέκταση/πρόσθεση περισσότερων επεξεργαστικών στοιχείων

4

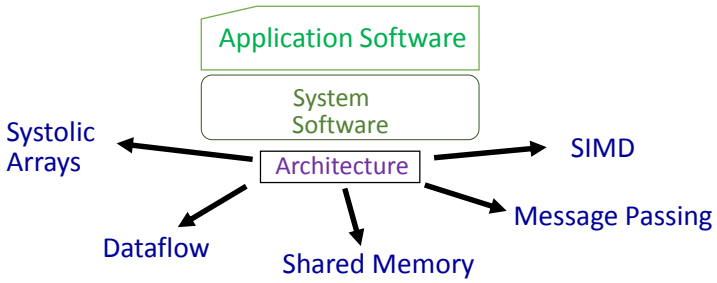
Προοπτικές Παράλληλης Αρχιτεκτονικής

- Πριν 15 χρόνια: επιτρέπει την αύξηση των επιδόσεων πέρα από ότι μπορεί η αύξηση της συχνότητας του ρολογιού
  - +30% freq/yr vs +40% transistors/yr—**10**× **πλεονέκτημα** σε 20 χρόνια
- Σήμερα: φαίνεται να αποτελεί τον καλύτερο τρόπο για υψηλότερες επιδόσεις στο προσεχές μέλλον
  - Ο ρυθμός του ρολογιού του επεξεργαστή δύσκολα αυξάνεται πλέον...
  - Ο παραλληλισμός στην εκτέλεση των εντολών δεν αυξάνεται πλέον...
- Δίχως παραλληλισμό (στοχευμένο), ή εξειδίκευση της αρχιτεκτονικής, δύσκολα θα βελτιωθούν οι επιδόσεις
  - Οι παράλληλες αρχιτεκτονικές μπορούν να απλοποιήσουν την αρχιτεκτονική
  - Και να μειωθεί η κατανάλωση ισχύος ($P = ½CV^2F$, $V \propto F \Rightarrow P \propto CF^3$)



Παράλληλες Αρχιτεκτονικές και Προγραμματιστικά Μοντέλα

- Αβεβαιότητα στην κατεύθυνση των παράλληλων αρχιτεκτονικών
- Σύνδεση μιας παράλληλης αρχιτεκτονικής με το προγραμματιστικό μοντέλο

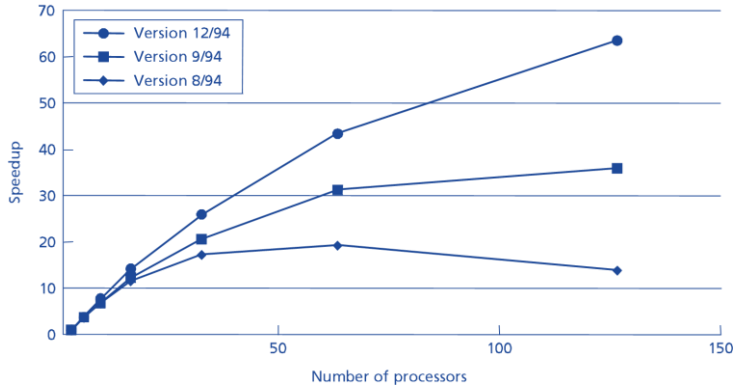


Ο παράλληλος προγραμματισμός παραμένει ένα μεγάλο πρόβλημα

6

Είναι η Παράλληλη Αρχιτεκτονική Αρκετή?

ΟΧΙ, οι παράλληλες αρχιτεκτονικές εξαρτώνται από τα προγράμματα για επιδόσεις



AMBER code for CRAY-1; ported to Intel Paragon

[Culler'99] 7

Τύποι Παραλληλισμού (1st: ILP)

- Παραλληλισμός σε επίπεδο εντολών
  - Διαφορετικές εντολές σε μια ακολουθία εντολών μπορούν να εκτελεστούν παράλληλα
  - Pipelining, out-of-order, speculative, VLIW
  - Dataflow

```
A:  LD R2, 0(R1)
    LD R3, 4(R1)
    SUBI R2, R2, #1
    SUBI R3, R3, #1
    BLTZ R2, B
    ST R2, 0(R1)
B:  BLTZ R3, C
    ST R3, 4(R1)
C:  ADDI R1, R1, #8
    SUB R5, R4, R1
    BGTZ R4, A
    RET
```

```
void decrement_all(
    int *array,
    int size) {
  for (int i = 0;
    i < size;
    i++) {
     int x = array[i] - 1;
     if (x > 0) {
       array[i] = x;
     }
  }
}
```

Loop unrolled x2

8

Τύποι Παραλληλισμού (1st: ILP) - 2

- Παραλληλισμός σε επίπεδο εντολών
  - Διαφορετικές εντολές σε μια ακολουθία εντολών μπορούν να εκτελεστούν παράλληλα
  - Pipelining, out-of-order, speculative, VLIW
  - Dataflow

```
A:  LD R2, 0(R1)
    LD R3, 4(R1)
    SUBI R2, R2, #1
    SUBI R3, R3, #1
    BLTZ R2, B
    ST R2, 0(R1)
B:  BLTZ R3, C
    ST R3, 4(R1)
C:  ADDI R1, R1, #8
    SUB R5, R4, R1
    BGTZ R4, A
    RET
```

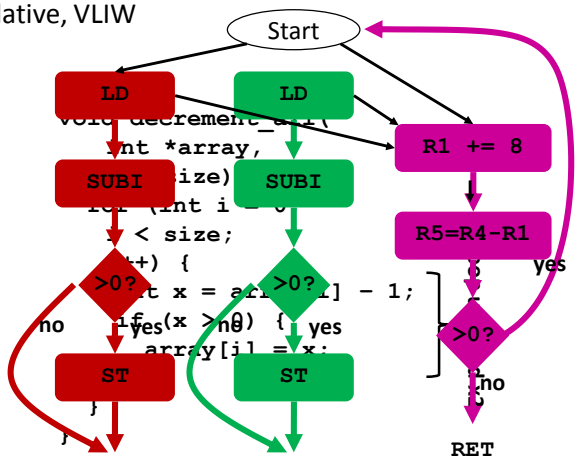
```
void decrement_all(
    int *array,
    int size) {
  for (int i = 0;
    i < size;
    i++) {
     int x = array[i] - 1;
     if (x > 0) {
       array[i] = x;
     }
  }
}
```

Loop unrolled x2

Τύποι Παραλληλισμού (1st: ILP) - 3

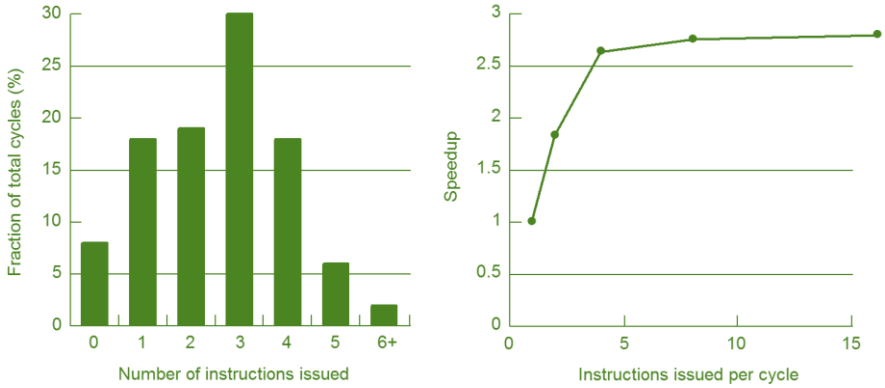
- Παραλληλισμός σε επίπεδο εντολών
  - Διαφορετικές εντολές σε μια ακολουθία εντολών μπορούν να εκτελεστούν παράλληλα
  - Pipelining, out-of-order, speculative, VLIW
  - Dataflow

```
A:  LD R2, 0(R1)
    LD R3, 4(R1)
    SUBI R2, R2, #1
    SUBI R3, R3, #1
    BLTZ R2, B
    ST R2, 0(R1)
B:  BLTZ R3, C
    ST R3, 4(R1)
C:  ADDI R1, R1, #8
    SUB R5, R4, R1
    BGTZ R4, A
    RET
```



Τύποι Παραλληλισμού (1st: ILP) - 4

- Πραγματικά προγράμματα με ρεαλιστικές καθυστερήσεις cache και pipeline
- Χωρίς περιορισμό στους πόρους ($\rightarrow\infty$)



Πάνω όριο παραλληλισμού → 4 εντολές / κύκλο ρολογιού

11

Τύποι Παραλληλισμού (2nd:Παρ/μός Δεδομένων)

- Παραλληλισμός Δεδομένων
  - Διαφορετικά δεδομένα μπορούν να επεξεργαστούν παράλληλα
  - Vector processing, array processing
  - Systolic arrays, streaming processors

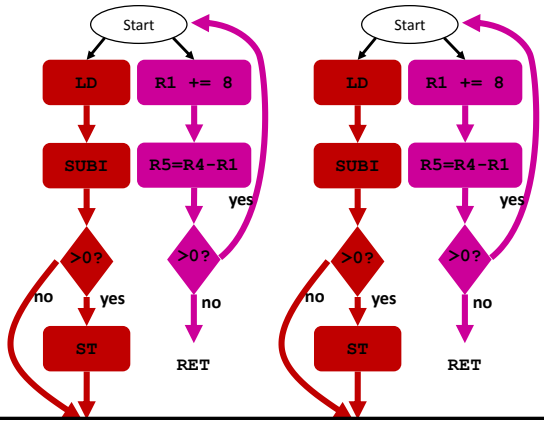
12

Τύποι Παραλληλισμού (3rd:Παραλληλισμός Task)

- Παραλληλισμός διεργασιών
  - Διαφορετικά tasks / threads μπορούν να εκτελεστούν παράλληλα
  - multithreading
  - Multiprocessing, multi-core

Adjust R1, R5 per thread…

```
A:  LD R2, 0(R1)
    SUBI R2, #1
    BLTZ R2, #0
    ST R2, 0(R1)
    ADDI R1, R1, #4
    SUB R5, R4, R1
    BGTZ R4, A
    RET
```



13

Παράλληλη Αρχιτεκτονική

- Επέκταση της «αρχιτεκτονικής υπολογιστών» για υποστήριξη επικοινωνίας και συνεργασίας
  - Παρελθόν: Αρχιτεκτονική Σετ Εντολών
  - Παρόν: Αρχιτεκτονική Επικοινωνίας

- Η αρχιτεκτονική επικοινωνίας ορίζει:
  - Όρια, βασικές λειτουργίες επικοινωνίας (interfaces)
  - Οργάνωση, δομές και σχεδίαση (hw, sw)
- Οι συμβολομεταφραστές, το λειτουργικό και οι βιβλιοθήκες έχουν καθοριστικό ρόλο
- Σύγκλιση κατανεμημένων συστημένων και παράλληλων αρχιτεκτονικών

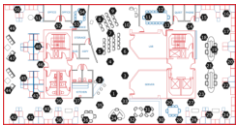
14

Παράλληλα (?) Συστήματα

Embedded-Physical Distributed

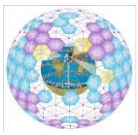
Claytronics

Sensor Networks



Geographically Distributed

Internet



Power Grid

Cloud Computing

EC2
Azure

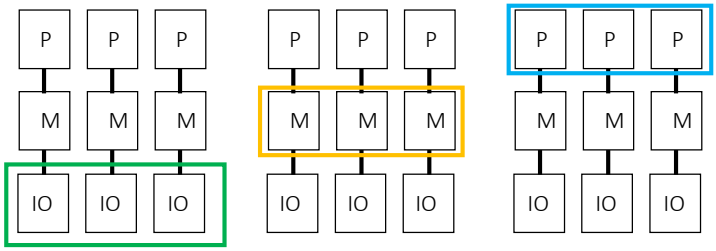
15

Προγραμματιστικό Μοντέλο

- Τι είναι? → Τεχνικές που χρησιμοποιεί ο προγραμματιστής για παράλληλα εφαρμογές
  - Λειτουργίες, ονομασίες, σειριοποίηση: για επικοινωνία και συνεργασία
- Παραδείγματα
  - Διαμοιρασμός χώρου διευθύνσεων: απαιτεί συγκεκριμένες λειτουργίες συγχρονισμού (λ.χ. ατομικές λειτουργίες)
  - Επικοινωνία με μηνύματα: συγκεκριμένα μηνύματα σημείου-σε-σημείο που ενεργούν ταυτόχρονα για επικοινωνία και συγχρονισμό
  - Παραλληλισμός δεδομένων: (δεδομένα οργανωμένα σε μια δομή ανατίθεται σε ένα σύνολο tasks) λειτουργίες σε δεδομένα σφαιρικές, υλοποιούνται και με διαμοιρασμό χώρου διευθύνσεων και με επικοινωνία με συγκεκριμένα μηνύματα

16

Που Υφίσταται η Επικοινωνία?



Σημείο επικοινωνίας:	I/O (Network)	Memory	Processor
Προγραμματιστική Τεχνική:	Message Passing	Shared Memory	Dataflow/Systolic

17

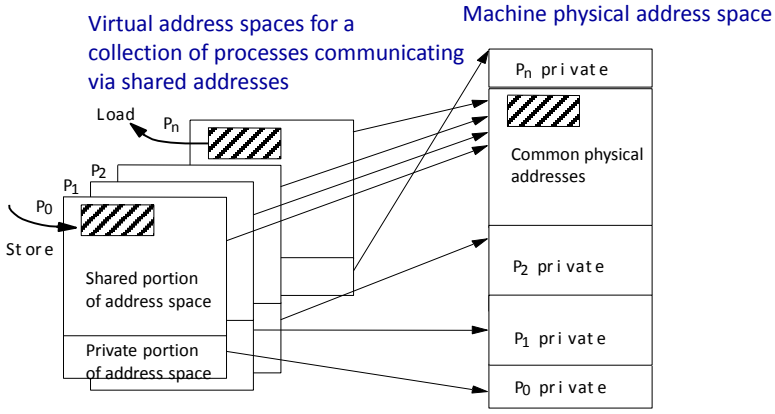
Αρχιτεκτονικές Διαμοιρασμένης Μνήμης (Shared Address Space)

- Όλοι οι επεξεργαστές έχουν άμεση πρόσβαση σε όποια θέση μνήμης
  - Η επικοινωνία γίνεται έμμεσα όταν γίνεται το load / store στη μνήμη
- Πλεονεκτήματα
  - Διαφανής πρόσβαση σε όποια μνήμη
  - Ίδιο προγραμματιστικό μοντέλο με ενός μονο-επεξεργαστικού συστήματος
- Ιστορικά ονομάζεται μοντέλο διαμοιρασμένης μνήμης
  - Μπορεί να είναι φυσικά συνδεδεμένη/κατανεμημένη στους επεξεργαστές

18

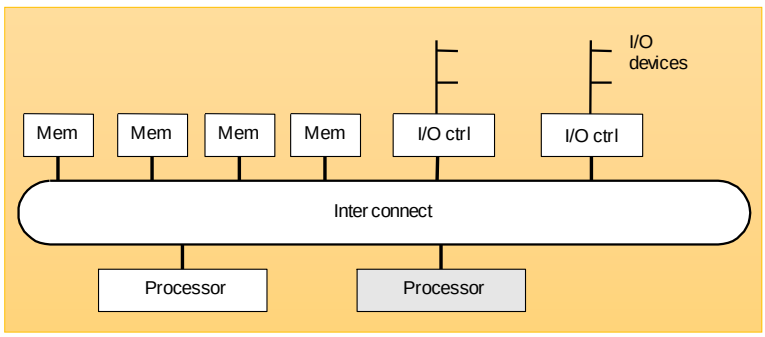
Προγραμματιστικό Μοντέλο SAS

- Εγγραφές στη διαμοιρασμένη μνήμη είναι ορατές από τα άλλα threads
- Επέκταση ενός μονο-επεξεργαστικού συστήματος – συνήθεις λειτουργίες μνήμης, ειδικές ατομικές λειτουργίες για συγχρονισμό
- Το Λ.Σ. Χρησιμοποιεί τη διαμοιρασμένη μνήμη για να συντονίσει τις διεργασίες



Τμήματα του χώρου διευθύνσεων των διεργασιών είναι διαμοιρασμένα

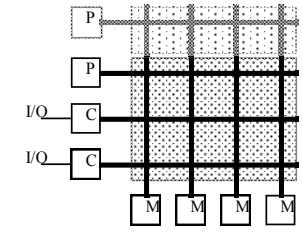
Επικοινωνία σε ένα SAS Σύστημα

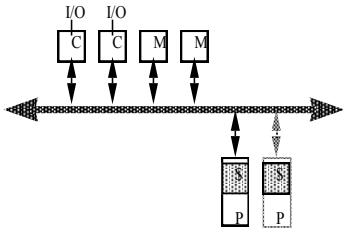


- Η μνήμη αυξάνεται προσθέτοντας στοιχεία μνήμης
- Οι εξωτερικές συσκευές επεκτείνονται μέσω προσθήκης ελεγκτών I/O
- Οι επιδόσεις βελτιώνονται προσθέτοντας επεξεργαστές

SAS Αρχιτεκτονικές

- A. Crossbar-based
  - η επεκτασιμότητα εξαρτάται από την επεκτασιμότητα του crossbar
  - Multi-stage crossbar
- B. Bus-based/symmetric multiprocessing (SMP)
  - Καθυστέρηση μεγαλύτερη από μονο-επεξεργαστικό σύστημα
  - Αδύνατο σημείο: ο δίαυλος επικοινωνίας
  - Η κρυφή μνήμη είναι μια υποσχόμενη λύση, αλλά…
    - Cache coherence ?

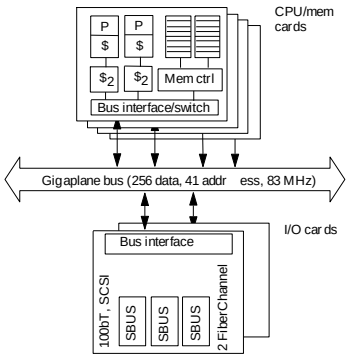




21

Παράδειγμα

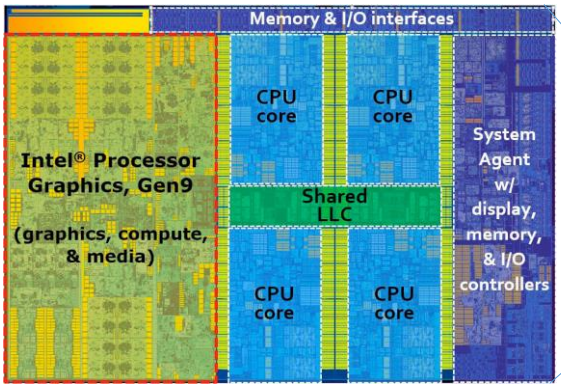
- 16 κάρτες διαφόρων τύπων: processors + memory, I/O
- Up to 30 processors
- Όλη η μνήμη ήταν προσβάσιμη μεσω του διαύλου, άρα συμμετρική
- Higher bandwidth, higher latency bus



SUN Enterprise (1996)

22

Παράδειγμα



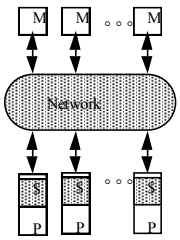
Intel, 2017

Intel's Core i7 7th generation

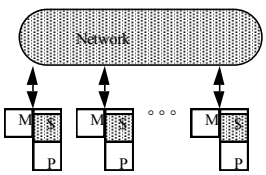
- Ολοκλήρωση σε ένα μοναδικό chip: low-latency, high-bandwidth communication via shared cache

23

Επεκτασιμότητα



"Dance hall"

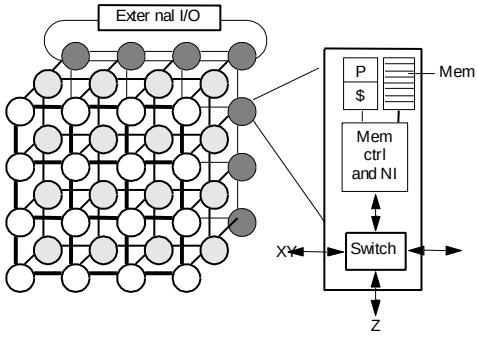


Distributed memory

- Βασικό πρόβλημα είναι το μέσο διασύνδεσης
  - Crossbar: κόστος, Bus: bandwidth
- Dance-hall: bandwidth still scalable, but lower cost than crossbar
  - Όμοια καθυστέρηση πρόσβασης στην μνήμη, αλλά η όμοια καθυστέρηση είναι μεγάλη...
- Distributed memory or non-uniform memory access (NUMA)
  - Κατασκευή ενός διαμοιρασμένου χώρου διευθύνσεων με απλά μηνύματα πάνω από ένα network γενικού σκοπού (e.g. read-request, read-response)
- Caching shared (particularly nonlocal) data?

24

Παράδειγμα

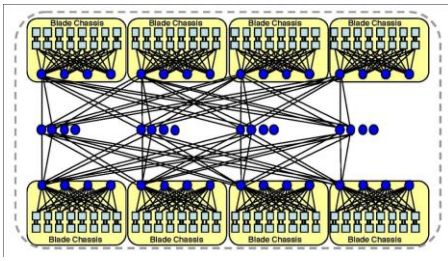


- Cray (1996)
  - Επεκτασιμότητα έως 1024 επεξεργαστές DEC ALPHA, 480MB/s links
  - Ο ελεγκτής μνήμης στέλνει μηνύματα επικοινωνίας για πρόσβαση σε μνήμες μη-τοπικές
  - Δεν υπάρχει μηχανισμός για coherence

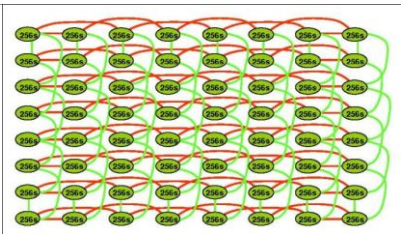
25

Παράδειγμα -2: SGI Altix UV'1000 (2009)

Blacklight at the PSC (4096 cores)



256 socket (2048 core) fat-tree
(this size is doubled in Blacklight via a torus)



8x8 torus

- Scales up to 131,072 Xeon cores
- 15GB/sec links
- Hardware cache coherence for blocks of 16TB with 2,048 cores

26

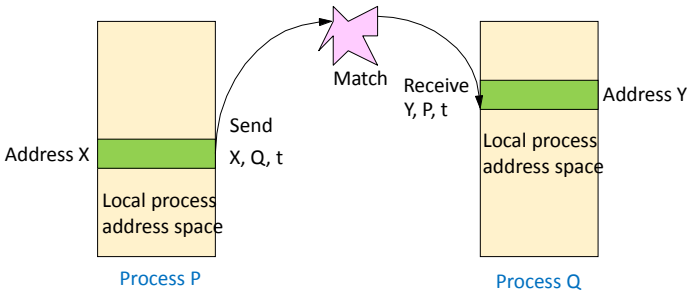
Αρχιτεκτονική με βάση τα Μηνύματα (Message Passing)

- Το δομικό στοιχείο είναι ένας πλήρης υπολογιστής (συμπεριλαμβανομένου και του I/O), η επικοινωνία γίνεται με μηνύματα I/O
- Προγραμματιστικό μοντέλο
  - Απ'ευθείας πρόσβαση μόνο στην τοπική μνήμη
  - Επικοινωνία με συγκεκριμένα μηνύματα (send/receive)
- Σε ψηλό επίπεδο παρόμοια αρχιτεκτονική με την SAS
  - Διαφέρει στην επικοινωνία: γίνεται μέσω I/O, δεν είναι ενσωματωμένη στο σύστημα μνήμης
  - Είναι πιο εύκολη η σχεδίαση συγκρινόμενη με ένα επεκτάσιμο SAS
- Το προγραμματιστικό μοντέλο στηρίζεται σε βιβλιοθήκες ή υποστήριξη από το Λ.Σ.

27

Πέρασμα Μηνυμάτων

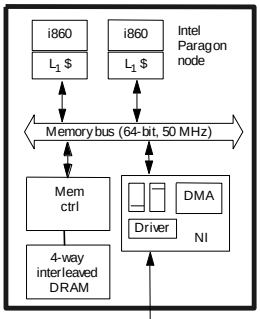
- Send: ορίζει τον buffer που πρέπει να σταλεί και σε ποια διεργασία
- Recv: ορίζει τη διεργασία που θα στείλει μήνυμα και τον χώρο που θα αποθηκευτεί το εισερχόμενο μήνυμα
- Memory-to-memory-copy: πρέπει να ορίζονται οι διεργασίες
- Προεραιτικά υποστηρίζονται Tag για την αποστολή και κανόνες για τα εισερχόμενα μηνύματα
- Στην απλή εκδοχή τα μηνύματα send/recv οριοθετούν και τα events συγχρονισμού
- Πολλά overheads: αντιγραφή στα μηνύματα, διαχείριση των buffers, προστασία

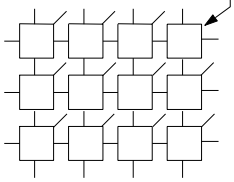


28

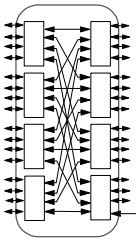
Παράδειγμα

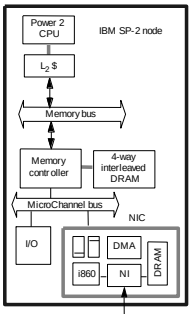
Sandia's Intel Paragon XP/S-based Supercomputer





General interconnection network formed from 8-port switches



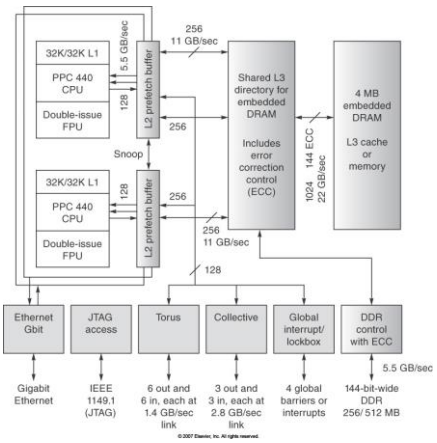


IBM SP-2 (1995):
Αποτελείται από ~complete RS6000 (IBM RISC) workstations
Network interface integrated in I/O bus (bw limited by I/O bus)

paragon (1993): Intel RISC processor (!!)- Up to 2048 nodes

29

Παράδειγμα

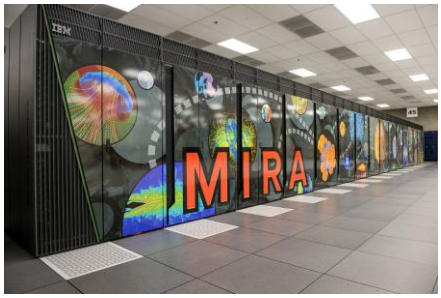


- IBM BLUE Gene/L (2004)
  - 2 low-power PowerPC 400s
  - Ολοκλήρωση, όλα σε ένα on-chip εκτός από DRAM
  - Έως 64K nodes
- Ο πιο ισχυρός supercomputer για 3.5 years (έως το 2008)
- First machine to achieve 100 TFLOPs on real application

30

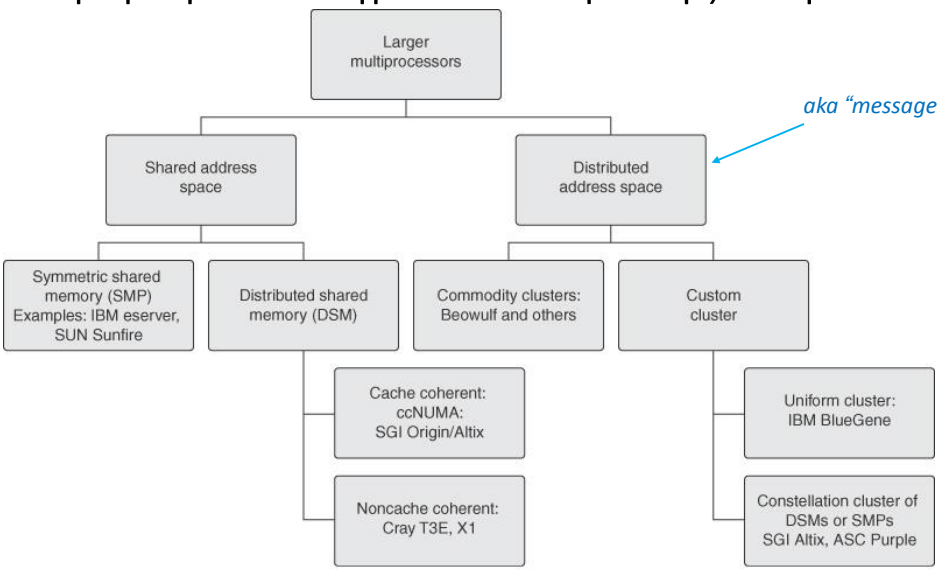
Παράδειγμα

- IBM BLUE Gene/Q (2011)
  - Node: 18 cores,
    4-way issue @ 1.6GHz,
    SIMD (vector) instructions,
    coherence within node
  - 16 user cores (1 for OS, 1 spare)
- Top of "green Top500" (2.1GFLOPS/W)
- Ο πρώτος που έφτασε τα 10PFLOPS με πραγματική εφαρμογή (100x BQ/L)



31

Ταξινόμηση: Συστήματα Μεγάλης Κλίμακας SAS, MP

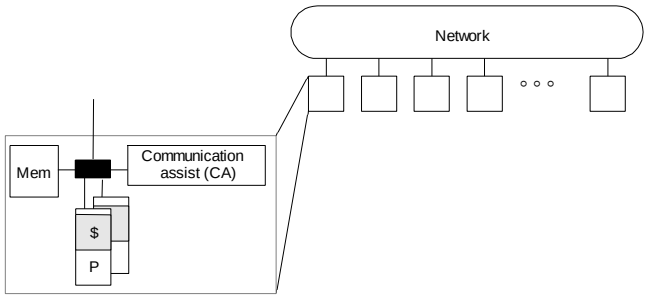


32

Σύγκλιση Αρχιτεκτονικής

- Η εξέλιξη και ο ρόλος του λογισμικού έχει "θολώσει" τα όρια
  - Send/recv υποστηρίζονται σε SAS με χρήση buffers
  - Συνολικός (global) χώρος διευθύνσεων είναι εφικτός σε MP μέσω hashing
  - Εικονική μνήμη κοινόχρηστη με υποστήριξη pages
- Σύγκλιση του Hardware έχει όμοια εξέλιξη
  - Στενή/άρρηκτη ολοκλήρωση διεπαφής δικτύου (NI) για MP (low-latency, high-bandwidth)
  - Σε χαμηλό επίπεδο, ακόμα και το hardware σε SAS χρησιμοποιεί μηνύματα σε hardware
- Ακόμα και clusters of workstations/SMPs γίνονται παράλληλα συστήματα
- Τα προγραμματιστικά μοντέλα είναι διαφορετικά, αλλά η οργάνωση των συστημάτων γίνεται παρόμοια
  - Οι κόμβοι συνδέονται μέσω δικτύων γενικού σκοπού και υποβοήθηση για επικοινωνία
  - Οι υλοποιήσεις επίσης συγκλίνουν σε μηχανές high-end

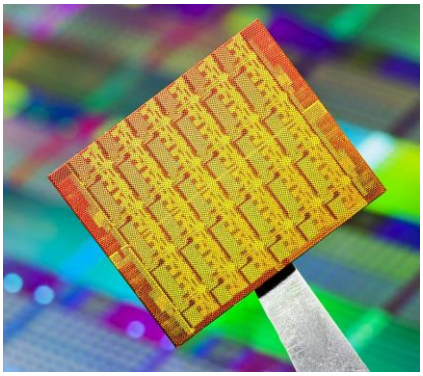
Μία Γενική Παράλληλη Αρχιτεκτονική



- Ένας κόμβος αποτελείται από επεξεργαστή, μνήμη και σύστημα επικοινωνίας
  - Σύστημα επικοινωνίας: διεπαφή δικτύου και ελεγκτής δικτύου
- Είναι σημαντικό το δίκτυο να είναι επεκτάσιμο
- Ανοιχτά πεδία: επίπεδο ολοκλήρωσης του συστήματος διασύνδεσης, λειτουργίες του ελεγκτή, καθυστέρηση και ρυθμός μετάδοσης, κ.α...

Παράδειγμα

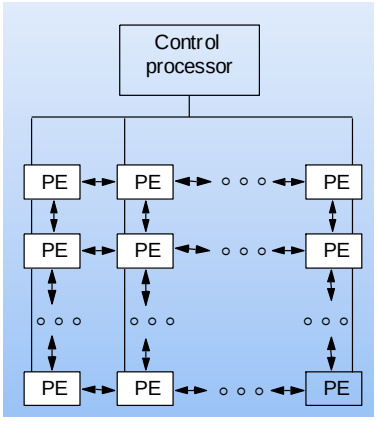
- Intel Cloud Computer (2009)
  - 48 cores
  - 2D mesh network
  - 24 tiles in 4x6 grid
  - 2 cores / tile
  - 16KB msg buffer / tile
  - 4 DDR3 controllers
  - No hardware coherence
  - Message passing hardware
  - Coherence available through software library



35

Παράλληλα Συστήματα Επεξεργασίας Δεδομένων (Data Parallel)

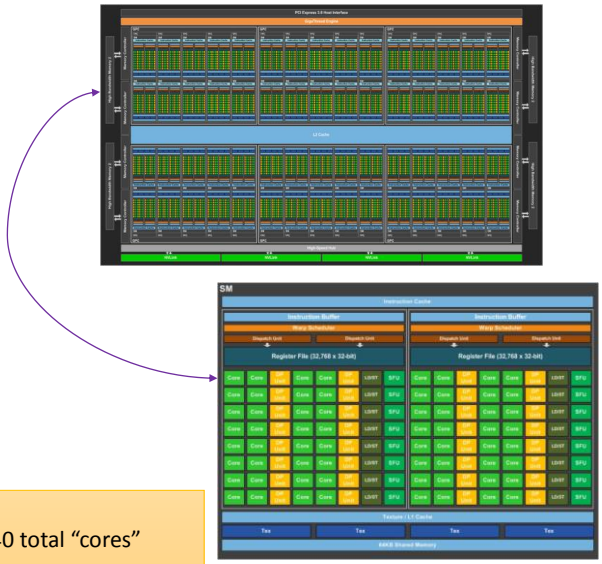
- Προγραμματιστικό μοντέλο
  - Λειτουργίες που εκτελούνται παράλληλα σε κάθε στοιχείο μιας δομής δεδομένων
  - Λογικά εκτελείται 1 thread ελέγχου
  - Γενικά, διατίθεται ένα επεξεργαστικό στοιχείο για κάθε στοιχείο της δομής δεδομένων
- Αρχιτεκτονικό μοντέλο
  - Μια συστοιχία από απλούς και φθηνούς επεξεργαστές με λίγη μνήμη
    - Οι επεξεργαστές δεν έχουν τη δυνατότητα να εκτελούν ακολουθία εντολών
  - Εξειδικευμένο και γενικού σκοπού σύστημα επικοινωνίας



36

Σύγχρονες Επιδράσεις της Αρχιτεκτονικής Data Parallel

- "Multimedia extensions" of ISAs (e.g., SSE)
  - Περιορισμένη παραλληλία τύπου SIMD: 4-8 δεδομένα
  - Εκδόσεις τύπου «εντολών διανυσμάτων» (vector instructions)
- Υπολογισμοί GPGPU
  - Προγραμματιστικό μοντέλο τύπου MIMD, αλλά πρακτικά είναι ένας επεξεργαστής που εκτελεί κώδικα σε πολλά threads, SIMD τύπου
  - Ονοματολογία GPGPU: vector lane == "core"
  - Στην πραγματικότητα: 16-64 multithreaded SIMD (vector) cores

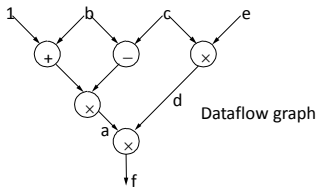


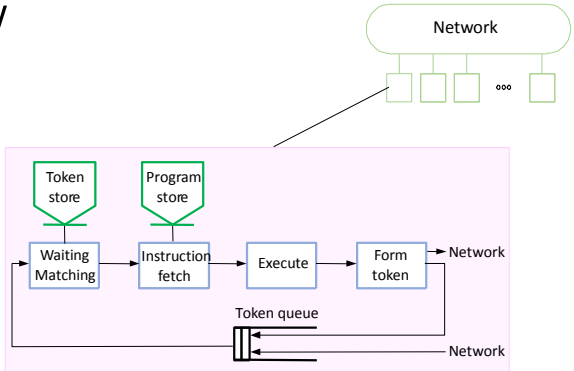
NVIDIA Pascal (2016)
60x streaming multiprocessors (SMs), 64 "CUDA cores" each => 3840 total "cores"
732 GB/s mem bw using 3D stacking technology
256KB registers / SM

37

Αρχιτεκτονικές Dataflow

$$a = (b + 1) \times (b - c)$$
$$d = c \times e$$
$$f = a \times d$$





- Αυτή η μοντελοποίηση βασίζεται στην αναπαράσταση των υπολογισμών ως έναν γράφο με τις εξαρτήσεις
  - Ένα επεργαστικό στοιχείο σε κάθε κόμβο υπολογίζει όταν εμφανιστούν οι είσοδοι
  - Μηνύματα (tokens), που περιέχουν ένα tag της επόμενης εντολής, στέλνονται στον επόμενο επεξεργαστή
  - Όταν ταιριάξει το tag τότε ενεργοποιείται η εκτέλεση

38

Αρχιτεκτονική Dataflow: Επιδράσεις...

- Εκτέλεση-εκτός-ακολουθίας (OoO execution)
  - Οι περισσότεροι επεξεργαστές σήμερα έχουν μία μηχανή dataflow εσωτερικά
  - OoO θεωρεί όρια σε περιοχές προγράμματος
- Ιδέες...
  - Ενσωμάτωση της επικοινωνίας με τη δημιουργία threads
  - Στενή σύνδεση επικοινωνίας και συγχρονισμού
  - Dataflow επεξεργασία σε περιοχές διαφόρων μεγεθών (granularity)

39

Κατανοώντας την Αρχιτεκτονική Παραλληλισμού

40

Προβλήματα

- Σχεδίαση διεπαφών σε αρκετά επίπεδα
  - Ο προγραμματιστής βλέπει το προγραμματιστικό μοντέλο
  - Συμβολομεταφραστής, βιβλιοθήκες, λειτουργικό σύστημα
  - Λογισμικό – υλικό
- Προβλήματα σχεδίασης
  - Μέγεθος-επίπεδο ολοκλήρωσης/κόμβο: πως να κατανεμηθούν επεξεργαστές, μνήμη και ελεγκτές δικτύου
  - Κόστος επικοινωνίας: καθυστέρηση, ρυθμός, επιβαρύνσεις, ...
  - Αντίγραφα: πως αντιγράφονται/κατανέμονται τα δεδομένα για να μειωθεί η επικοινωνία

41

Προγραμματιστικό Μοντέλο SAS

- Ονοματολογία: οποιοδήποτε process δίνει ότι όνομα θέλει σε μεταβλητή που βρίσκεται στον διαμοιρασμένο χώρο
- Λειτουργίες:
  - Loads, stores, και κάποιες επιπλέον για να εξασφαλιστεί η σειρά προσβάσεων
- Μοντέλο Σειριακής Πρόσβασης: Sequential Consistency
  - Εντός σε μία διεργασία/thread υπάρχει σειριακή εκτέλεση στο πρόγραμμα
  - Μεταξύ των threads: μερική/εκ περιτροπής εκτέλεση (όπως σε time-sharing)
  - Επιπλέον σειριοποίηση μέσω λειτουργιών συγχρονισμού
  - Συμβολομεταφραστές και hardware ενδέχεται να παραβιάζουν τη σειριακή εκτέλεση....

42

Σειριοποίηση – Μέθοδοι Συγχρονισμού

- Αμοιβαίος αποκλεισμός (locks): μέθοδος επίτρεψης μόνο ενός για να εκτελέσει μια λειτουργία
  - Acquire()/Release(): μόνο ένα thread τη φορά
  - Εξασφάλιση λειτουργιών σε δεδομένα από μονάχα ένα thread
  - Δεν υπάρχουν εγγυήσεις μεταξύ threads
- Διάφορες τεχνικές συγχρονισμού
  - Γεγονότα: αναμονή έως ότου η μεταβλητή S γίνει 1 από κάποιον
  - Barriers: αναμονή έως ότου N threads φτάσουν στο barrier
  - Read-write lock: επιτρέπονται πολλοί να διαβάσουν αλλά μόνο έναν να την αλλάξει
  - Semaphores: επιτρέπονται έως N concurrent threads
  - ….

43

Παράλληλες Αρχιτεκτονικές - Συμπεράσματα

- Επίδραση τεχνολογίας, κόστους και επιδόσεων εφαρμογών παρά τις εξωτικές αρχιτεκτονικές
  - Η βασική αρχιτεκτονική επεξεργαστή – μνήμης παραμένει η ίδια
  - Βασικό θέμα είναι η αρχιτεκτονική δικτύου διασύνδεσης
  - Οι αποφάσεις σχεδίασης καθοδηγούνται από τις επιδόσεις προγραμμάτων
- Ουσιαστικός στόχος είναι ένα σύστημα
  - Εύκολο στον προγραμματισμό
  - Να υποστηρίζει καλές επιδόσεις (αποδοτικότητα: σε επιδόσεις/watt, /euro…)
  - Να επεκτείνεται καλά (προσθέτοντας πόρους να αυξάνουν οι επιδόσεις)

Στόχος είναι η συνήθης περίπτωση / πρόβλημα να επεξεργάζεται γρήγορα!

44

Μετρικές Επιδόσεων

Επιτάχυνση Λόγω Παραλληλισμού

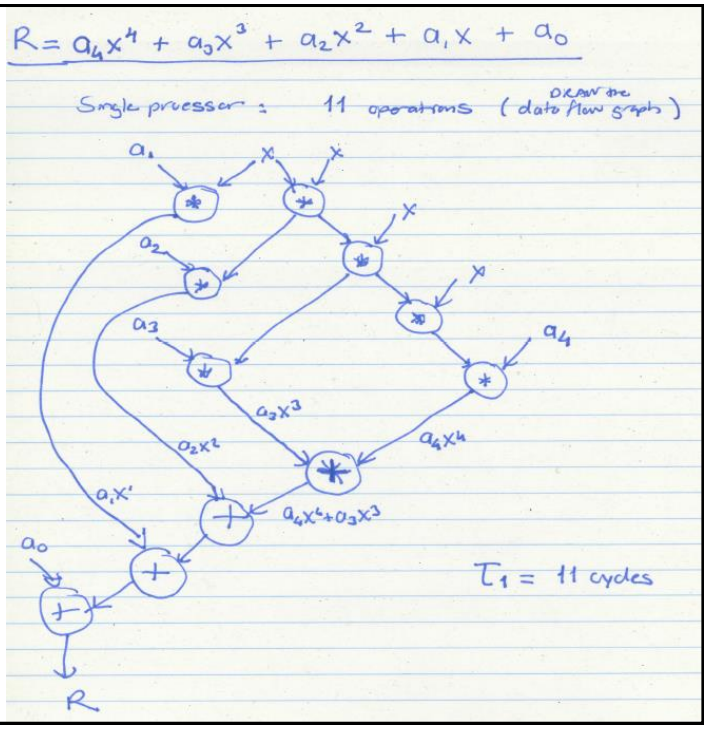
$$\frac{\text{χρόνος εκτέλεσης προγράμματος με 1 processor}}{\text{χρόνος εκτέλεσης προγράμματος με } N \text{ processors}}$$

Παράδειγμα Επιτάχυνσης

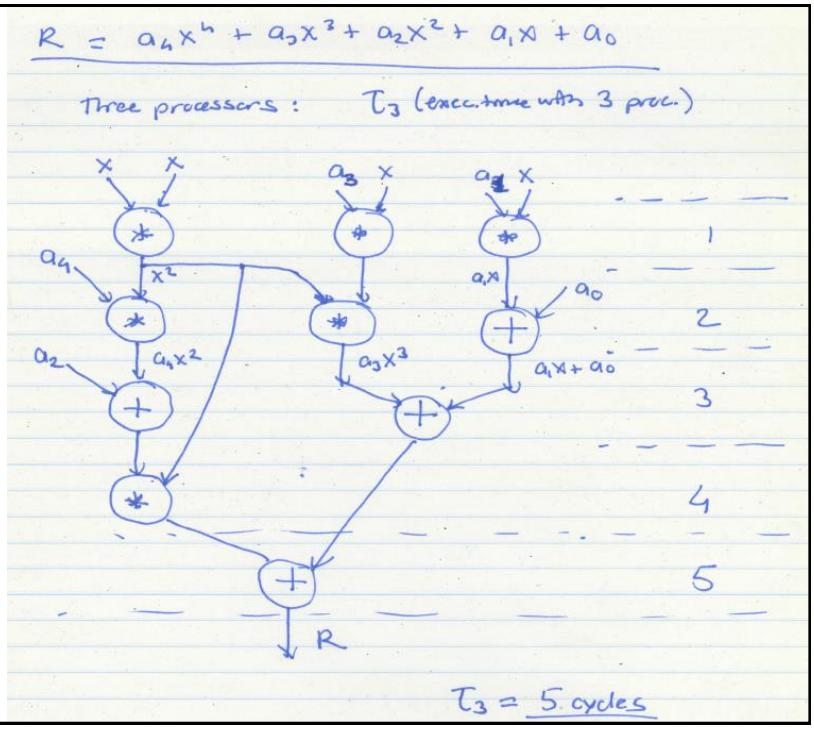
- Έστω ο υπολογισμός: $a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$
- Υπόθεση: 1 cycle ανά υπολογισμό, μηδενικό κόστος επικοινωνίας, κάθε λειτουργία μπορεί να εκτελεστεί σε ξεχωριστό επεξεργαστή
- Πόσο γρήγορο είναι το πρόγραμμα σε έναν επεξεργαστή?
  - Υποθέτουμε χωρίς pipelining ή παράλληλη εκτέλεση εντολών
- Πόσο γρήγορο είναι το πρόγραμμα σε τρεις επεξεργαστές?

47

Λύση για 1 Επεξεργαστή



Λύση για 3 CPUs



Επιτάχυνση με 3 Επεξεργαστές

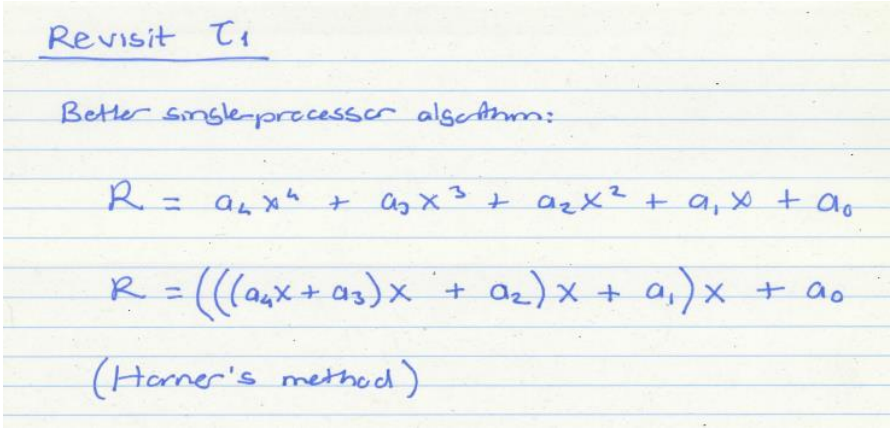
$\tau_3 = 5$ cycles

Speedup with 3 processors $= \frac{11}{5} = 2.2$

$\left(\frac{\tau_1}{\tau_3}\right)$

Είναι δίκαιη η σύγκριση ?

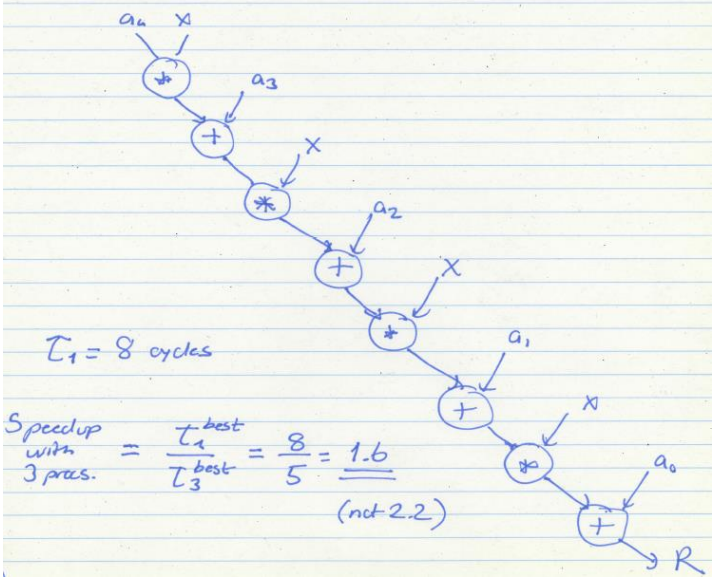
Αλγόριθμος για 1 Επεξεργαστή....



Horner, "A new method of solving numerical equations of all orders, by continuous approximation," Philosophical Transactions of the Royal Society, 1819.

51

Επιτάχυνση με 3 Επεξεργαστές



52

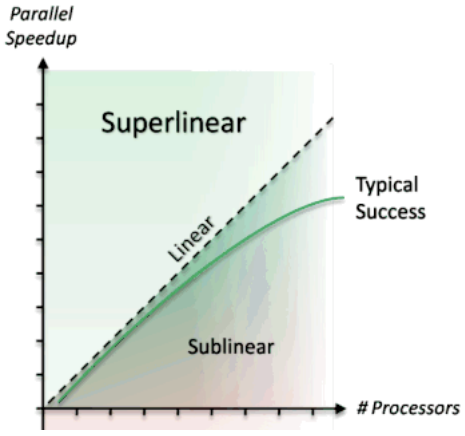
Συμπέρασμα

- Για έναν δίκαιο υπολογισμό της επιτάχυνσης πρέπει να γίνεται χρήση του βέλτιστου αλγορίθμου για καθέναν από τους N επεξεργαστές
- Αν δεν χρησιμοποιείται ο βέλτιστος τότε το αποτέλεσμα ονομάζεται superlinear speedup

53

Επιτάχυνση Υπερ-γραμμική

- Είναι δυνατόν να υπάρχει super-linear speedup (>P) όταν έχουμε P επεξεργαστές?
- Πρέπει να λαμβάνεται υπόψιν
  - Η επίδραση της cache
  - Οι καθυστερήσεις της μνήμης
  - Το μέγεθος των δεδομένων επεξεργασίας
  - Μπορεί να συμβεί όταν γίνονται ανόμοιες συγκρίσεις ή όταν δεν λαμβάνονται υπόψιν οι επιδράσεις του συστήματος μνήμης



54

Νόμος του Amhdal

- Amdahl's Law: ένα πρόγραμμα αποτελείται από ένα τμήμα που μπορεί να εκτελεστεί μόνο σειριακά.
- Έστω:
  - f: Parallelizable fraction of a program
  - P: Number of processors

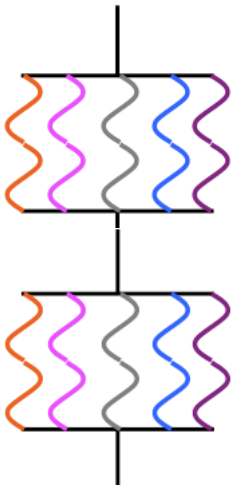
$$\text{Speedup} = \frac{1}{1 - f + \frac{f}{P}}$$

- Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," AFIPS 1967.

Η Μέγιστη Επιτάχυνση αναγκαστικά περιορίζεται λόγω του σειριακού τμήματος — aka the **Serial Bottleneck**

55

Σειριακό Bottleneck σε Παράλληλη Μηχανή



Βασική αιτία: λειτουργίες σε δεδομένα που δεν είναι παραλληλοποιήσιμες (e.g. non-parallelizable loops)

```
for ( i = 0 ; i < N; i++)
        A[i] = (A[i] + A[i-1]) / 2
```

Συνήθης μέθοδος είναι ένα μοναδικό thread προετοιμάζει τα data και δημιουργεί τα parallel tasks (τα οποία εσωτερικά είναι σειριακά)

56

Εμπόδια στην Παράλληλη Εκτέλεση

- Synchronization: Operations manipulating shared data cannot be parallelized
  - Locks, mutual exclusion, barrier synchronization
    - Suleman et al., "Accelerating Critical Section Execution with Asymmetric Multi-Core Architectures," ASPLOS 2009.
  - Communication: Tasks may need values from each other
    - Causes thread serialization when shared data is contended
- Load Imbalance: Parallel tasks may have different lengths
  - Due to imperfect parallelization or microarchitectural effects
    - Reduces speedup in parallel portion
- Resource Contention: Parallel tasks can share hardware resources, delaying each other
  - Replicating all resources (e.g., memory) expensive
    - Additional latency not present when each task runs alone

57

Βιβλιογραφία

- Πως μπορούν να επιλυθούν κάποια από τα προβλήματα στο σειριακό τμήμα και τα προβλήματα στο παράλληλο τμήμα ?
- Reading list:
  - Annavaram et al., "Mitigating Amdahl's Law Through EPI Throttling," ISCA 2005.
  - Suleman et al., "Accelerating Critical Section Execution with Asymmetric Multi-Core Architectures," ASPLOS 2009.
  - Joao et al., "Bottleneck Identification and Scheduling in Multithreaded Applications," ASPLOS 2012.
  - Ipek et al., "Core Fusion: Accommodating Software Diversity in Chip Multiprocessors," ISCA 2007.
  - Hill and Marty, "Amdahl's Law in the Multi-Core Era," IEEE Computer 2008.

58