

ΠΑΡΑΛΛΗΛΟΙ ΑΛΓΟΡΙΘΜΟΙ

ΕΔΙΠ Μαρία Λουκά

Τμήμα Πληροφορικής και Τηλεπικοινωνιών



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών

26 Ιουλίου 2025

Κεφάλαιο 1

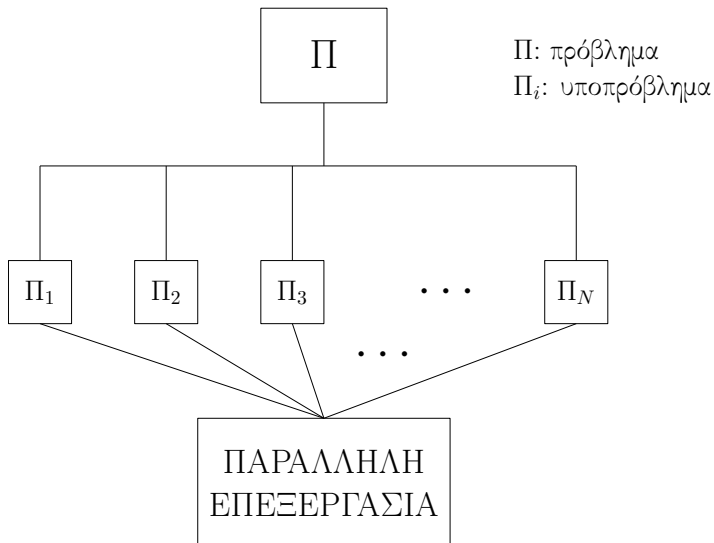
Εισαγωγή (Introduction)

Η ανάγκη για παράλληλους υπολογιστές

- Οι εφαρμογές απαιτούν χρονοβόρους υπολογισμούς:
 - ▶ Σχεδιασμός αεροπλάνων
 - ▶ Αριθμητική προσομοίωση σε προβλήματα Υπολογιστικής Δυναμικής των Ρευστών
 - ▶ Εξαγωγή πετρελαίου
 - ▶ Κρυπτανάλυση
 - ▶ Αστρονομία
 - ▶ Επίλυση μεγάλων συστημάτων Μερικών Διαφορικών Εξισώσεων που προκύπτουν από αριθμητικές προσομοιώσεις σε επιστημονικούς κλάδους τόσο διαφορετικούς μεταξύ τους όπως η σεισμολογία, η αεροδυναμική και η ατομική, πυρηνική και φυσική πλάσματος
- Η αύξηση των υπολογισμών θα κορεστεί.

Λύση: Χρήση της παραλληλίας

Η ανάγκη για παράλληλους υπολογιστές



Σχήμα: Διάγραμμα block παράλληλης επεξεργασίας

Η ανάγκη για παράλληλους υπολογιστές

Παράλληλος Αλγόριθμος: Μια μέθοδος λύσης για ένα δεδομένο πρόβλημα που πρόκειται να υλοποιηθεί σε ένα παράλληλο υπολογιστή

Παράλληλος Υπολογιστής: Ένας υπολογιστής με πολλές μονάδες επεξεργασίας ή επεξεργαστές.

Κάθε υπολογιστής, ακολουθιακός ή παράλληλος, λειτουργεί εκτελώντας εντολές στα δεδομένα. Ένα ρεύμα εντολών (ο αλγόριθμος) λέει στον υπολογιστή τι να κάνει σε κάθε βήμα. Ένα ρεύμα δεδομένων (η είσοδος του αλγορίθμου) επηρεάζεται από αυτές τις εντολές.

Διακρίνουμε τις ακόλουθες 4 περιπτώσεις ανάλογα με το πόσα ρεύματα έχουμε:

Μοντέλα Υπολογισμού

- | | |
|--------|----------------|
| • SISD | S: Single |
| • MISD | I: Instruction |
| • SIMD | D: Data |
| • MIMD | M: Multiple |

Η ανάγκη για παράλληλους υπολογιστές

SISD

- John von Neumann 1940
- Ακολουθιακός (σειριακός) Αλγόριθμος



Σχήμα: Single Instruction Single Data

Η ανάγκη για παράλληλους υπολογιστές

SISD

- Ένας υπολογιστής αυτής της κατηγορίας αποτελείται από μία μονάδα επεξεργασίας που λαμβάνει μία ροή εντολών που ενεργούν σε μία ροή δεδομένων, όπως φαίνεται στο σχήμα 2.
- Σε κάθε βήμα κατά τη διάρκεια του υπολογισμού, η μονάδα ελέγχου στέλνει μία εντολή που επεξεργάζεται ένα δεδομένο που προέρχεται από τη μονάδα μνήμης.
- Αυτή μπορεί να δώσει εντολή στον επεξεργαστή, για παράδειγμα, να εκτελέσει κάποια αριθμητική ή λογική λειτουργία στο δεδομένο και στη συνέχεια να το επαναφέρει στη μνήμη.
- Η ταχύτητα του οριοθετείται από δυο παράγοντες
 - 1 Ρυθμός εκτέλεσης εντολών
 - 2 Ταχύτητα ανταλλαγής πληροφορίας μεταξύ CPU και μνήμης

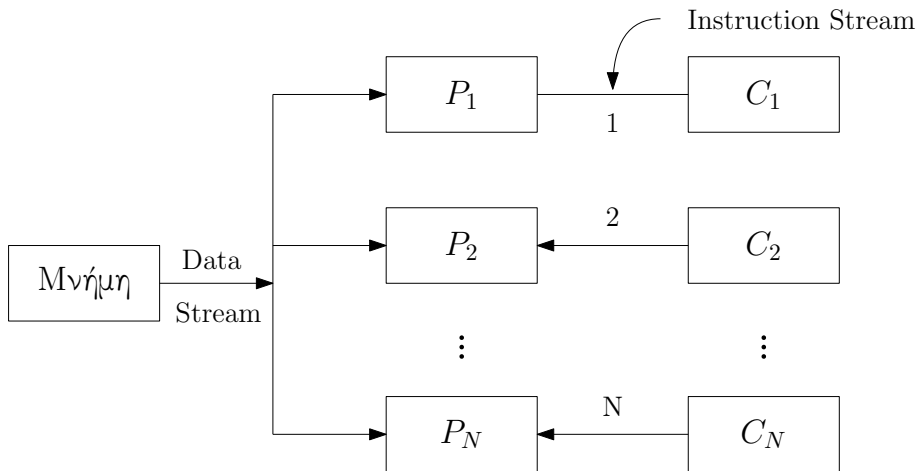
Η ανάγκη για παράλληλους υπολογιστές

SISD

Παράδειγμα

Για να υπολογίσει το άθροισμα n αριθμών, ο επεξεργαστής πρέπει να αποκτήσει πρόσβαση στη μνήμη n συνεχόμενες φορές και κάθε φορά να λαμβάνει έναν αριθμό. Υπάρχουν επίσης $n - 1$ προσθέσεις που εκτελούνται διαδοχικά. Επομένως, αυτός ο υπολογισμός απαιτεί συνολικά περίπου n πράξεις.

- Αυτό το παράδειγμα δείχνει ότι οι αλγόριθμοι για υπολογιστές SISD δεν περιέχουν κανένα παράλληλο στοιχείο.
- Ο λόγος είναι προφανής, υπάρχει μόνο ένας επεξεργαστής!
- Για να επιτευχθεί από έναν υπολογιστή ο τύπος παράλληλης λειτουργίας που ορίστηκε νωρίτερα, θα πρέπει να διαθέτει πολλούς επεξεργαστές. Αυτό παρέχεται από τις επόμενες τρεις κατηγορίες υπολογιστών. Σε κάθε μία από αυτές τις κατηγορίες, ένας υπολογιστής διαθέτει N επεξεργαστές όπου $N > 1$.



Σχήμα: Multiple Instruction Single Data

- Εδώ, N επεξεργαστές, ο καθένας με τη δική του μονάδα ελέγχου, μοιράζονται μια κοινή μονάδα μνήμης όπου βρίσκονται τα δεδομένα, όπως φαίνεται στο σχήμα 3.
- Υπάρχουν N ροές εντολών και μία ροή δεδομένων. Σε κάθε βήμα, ένα δεδομένο που λαμβάνεται από τη μνήμη υποβάλλεται σε επεξεργασία από όλους τους επεξεργαστές ταυτόχρονα, ο καθένας σύμφωνα με την εντολή που λαμβάνει από τον έλεγχο του.
- Έτσι, ο παραλληλισμός επιτυγχάνεται αφήνοντας τους επεξεργαστές να κάνουν διαφορετικά πράγματα ταυτόχρονα στο ίδιο δεδομένο.
- Αυτή η κατηγορία υπολογιστών προσφέρεται φυσικά για υπολογισμούς που απαιτούν μια είσοδο να υποβληθεί σε διάφορες λειτουργίες, κάθε μία από τις οποίες λαμβάνει την είσοδο στην αρχική της μορφή.

Παράδειγμα: Έλεγχος εάν ένας αριθμός z είναι πρώτος

Παράλληλος Αλγόριθμος

- Έστω ότι υπάρχουν τόσοι επεξεργαστές όσοι είναι και οι υποψήφιοι διαιρέτες του z .
- Έλεγχος από κάθε επεξεργαστή αν ο υποψήφιος διαιρέτης διαιρεί το z .

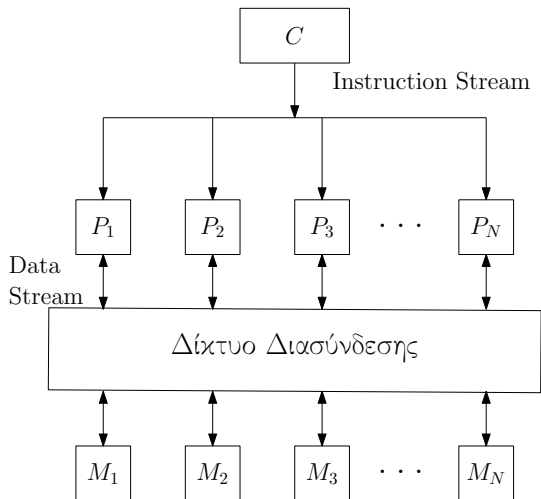
Παράδειγμα: Εύρεση του συνόλου (κατηγορίας) στην οποία ανήκει ένα αντικείμενο

Παράλληλος Αλγόριθμος

- Υποθέτουμε ότι υπάρχουν τόσοι επεξεργαστές όσες είναι οι κατηγορίες.
- Στέλνεται το αντικείμενο σε κάθε επεξεργαστή, ο οποίος ελέγχει αν ανήκει στην κατηγορία του.

Μειονέκτημα

- Ειδικής μορφής υπολογισμοί



Σχήμα: Single Instruction Multiple Data

- Σε αυτή την κατηγορία έχουμε ένα παράλληλο υπολογιστή με N πανομοιότυπους επεξεργαστές, όπως φαίνεται στο σχήμα 4.
- Κάθε ένας από τους N επεξεργαστές διαθέτει τη δική του τοπική μνήμη, όπου μπορεί να αποθηκεύει τόσο προγράμματα όσο και δεδομένα
- Όλοι οι επεξεργαστές λειτουργούν υπό τον έλεγχο μιας ενιαίας ροής εντολών που εκδίδεται από μια κεντρική μονάδα ελέγχου.
- Αντίστοιχα, μπορεί να θεωρηθεί ότι οι N επεξεργαστές διαθέτουν πανομοιότυπα αντίγραφα ενός ενιαίου προγράμματος, με το αντίγραφο κάθε επεξεργαστή να αποθηκεύεται στην τοπική μνήμη του.
- Υπάρχουν N ροές δεδομένων, μία για κάθε επεξεργαστή.
- Οι επεξεργαστές λειτουργούν συγχρονισμένα: Σε κάθε βήμα, όλοι οι επεξεργαστές εκτελούν την ίδια εντολή, ο καθένας σε διαφορετικό δεδομένο.
- Η εντολή μπορεί να είναι απλή (όπως η πρόσθεση ή ο υπολογισμός δύο αριθμών).
- Ομοίως, το δεδομένο μπορεί να είναι απλό (ένας αριθμός) ή σύνθετο (αρκετοί αριθμοί).

- Μερικές φορές, μπορεί να είναι απαραίτητο να εκτελεί μια εντολή μόνο ένα υποσύνολο του επεξεργαστή.
- Η πληροφορία μπορεί να κωδικοποιηθεί στην ίδια την εντολή, ενημερώνοντας έτσι έναν επεξεργαστή εάν πρέπει να είναι ενεργός (και να εκτελέσει την εντολή) ή ανενεργός (και να περιμένει την επόμενη εντολή).
- Υπάρχει ένας μηχανισμός, όπως ένα παγκόσμιο ρολόι, που εξασφαλίζει τη λειτουργία lock-step. Έτσι, οι επεξεργαστές που είναι ανενεργοί κατά τη διάρκεια μιας εντολής πριν από άλλους μπορούν να παραμείνουν αδρανείς μέχρι να εκδοθεί η επόμενη εντολή.
- Το χρονικό διάστημα μεταξύ δύο εντολών μπορεί να είναι σταθερό ή μπορεί να εξαρτάται από την εντολή που εκτελείται.

SIMD Παράδειγμα

αν ($B = 0$)
 $C = A$
αλλιώς
 $C = A/B$

A	5
B	0
C	0

Επεξεργαστής 0

A	4
B	2
C	0

Επεξεργαστής 1

A	1
B	1
C	0

Επεξεργαστής 2

A	0
B	0
C	0

Επεξεργαστής 3

Αρχικές τιμές

Αδρανής

A	5
B	0
C	5

Επεξεργαστής 0

A	4
B	2
C	0

Επεξεργαστής 1

A	1
B	1
C	0

Επεξεργαστής 2

A	0
B	0
C	0

Επεξεργαστής 3

Βήμα 1

Αδρανής

A	5
B	0
C	5

Επεξεργαστής 0

A	4
B	2
C	2

Επεξεργαστής 1

A	1
B	1
C	1

Επεξεργαστής 2

A	0
B	0
C	0

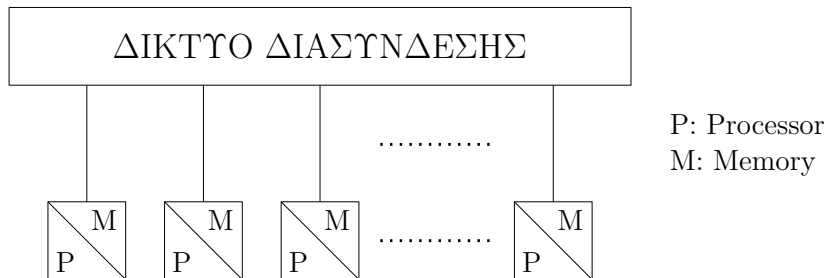
Επεξεργαστής 3

Βήμα 2

Αδρανής

- Κάθε επεξεργαστής έχει τη δική του τοπική μνήμη (πρόγραμμα και δεδομένα)
- Οι επεξεργαστές λειτουργούν σύγχρονα: σε κάθε βήμα, όλοι οι επεξεργαστές εκτελούν την ίδια εντολή, ο κάθε ένας σε διαφορετικά δεδομένα
- Κατάλληλοι για παράλληλα-δεδομένα (data-parallel)

- Στα περισσότερα ενδιαφέροντα προβλήματα που επιθυμούμε να λύσουμε σε έναν υπολογιστή SIMD, είναι επιθυμητό να υπάρχει δυνατότητα επικοινωνίας μεταξύ τους κατά τη διάρκεια του υπολογισμού, προκειμένου να ανταλλάσσονται δεδομένα ή ενδιάμεσα αποτελέσματα.
- Αυτό μπορεί να επιτευχθεί με δύο τρόπους, δημιουργώντας δύο υποκατηγορίες:
 - 1 Υπολογιστές SIMD όπου η επικοινωνία γίνεται μέσω κοινής μνήμης και
 - 2 Υπολογιστές όπου η επικοινωνία γίνεται μέσω δικτύου διασύνδεσης.



Σχήμα: Μία τυπική αρχιτεκτονική διαβίβασης μηνυμάτων (message passing)

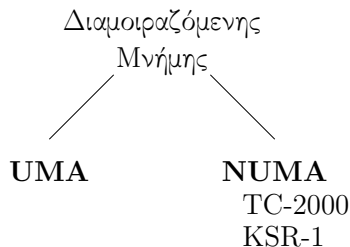
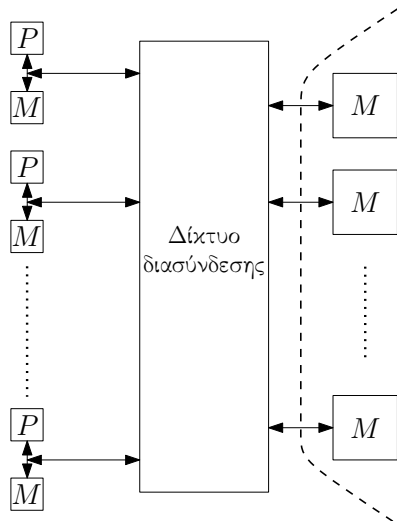
Διαμοιραζόμενης Μνήμης Υπολογιστές

- C. mmp
- NYU Ultracomputer

Μειονεκτήματα

- μεγάλο εύρος δικτύου διασύνδεσης, γιατί σε κύκλο εντολής κάθε επεξεργαστής ίσως χρειάζεται να προσπελάσει μια λέξη από τη διαμοιραζόμενη μνήμη μέσω του δικτύου διασύνδεσης
- αργή προσπέλαση μνήμης, καθώς μια αίτηση για ανάγνωση ή εγγραφή πρέπει να περάσει μέσα από πολλές φάσεις (stages) στο δίκτυο

Θεραπεία → τοπική μνήμη, η οποία αποθηκεύει το πρόγραμμα που εκτελείται στον επεξεργαστή και όχι στις μη διαμοιραζόμενες δομές δεδομένων.

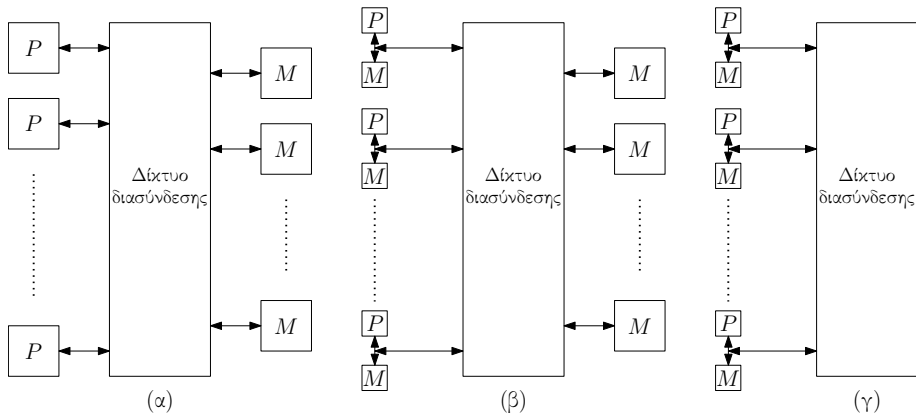


Σχήμα: Υπολογιστής Διαμοιραζόμενης Μνήμης

Διαμοιραζόμενη Μνήμη Υπολογιστές

Κατηγορίες

- UMA: Ο χρόνος που χρειάζεται ένας επεξεργαστής για να προσπελάσει μια λέξη μνήμης είναι ο ίδιος. Όλοι οι επεξεργαστές βλέπουν ταυτόχρονα την κοινή μνήμη
- NUMA: Ο χρόνος προσπέλασης ενός μακρινού memory bank είναι μεγαλύτερος από το χρόνο προσπέλασης ενός τοπικού
 - 1 NUMA με τοπική και ολική μνήμη: Κάθε επεξεργαστής έχει τη δική του τοπική μνήμη και διατηρείται και η ολική διαμοιραζόμενη μνήμη
 - 2 NUMA με τοπική μνήμη: Δεν υπάρχει καθολική μνήμη γιατί έγινε τοπική σε κάθε επεξεργαστή. Ο κάθε επεξεργαστής επικοινωνεί με την τοπική του μνήμη και αν δεν βρει αυτό που θέλει μπορεί να έχει μέσω δικτύου προσπέλαση σε οποιαδήποτε άλλη μνήμη.



Σχήμα: Κοινή Μνήμη: (α) UMA, (β) NUMA με τοπική και ολική μνήμη, (γ) NUMA με τοπική μνήμη μόνο

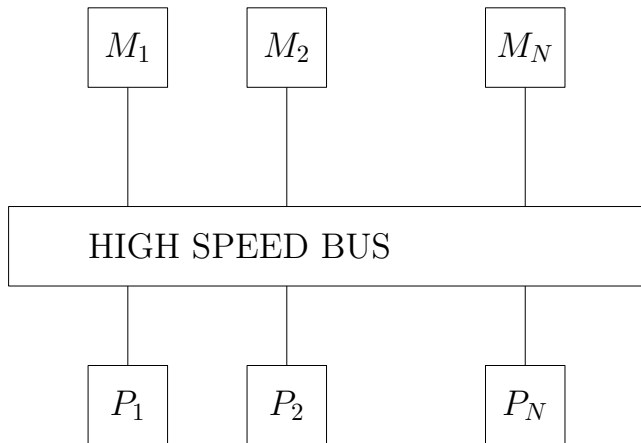
Κατηγορίες Δικτύων Διασύνδεσης

- Στατικά Δίκτυα: Αποτελούνται από point-to-point συνδέσμους επικοινωνίας μεταξύ των επεξεργαστών.
- Δυναμικά Δίκτυα: Φτιάχνονται χρησιμοποιώντας switches και συνδέσμους επικοινωνίας. Οι σύνδεσμοι επικοινωνίας συνδέουν το ένα το άλλο δυναμικά μέσω των διακοπών προκειμένου να φτιάξουν μονοπάτια μεταξύ επεξεργαστών και memory banks.

Διαμοιραζόμενη Μνήμη (PRAM) SIMD

- EREW: Σε αυτή την κατηγορία, η πρόσβαση σε μια θέση μνήμης είναι αποκλειστική. Δεν μπορούν ταυτόχρονα δύο επεξεργαστές να διαβάσουν ή να γράψουν στην ίδια τοποθεσία μνήμης. Αυτό είναι το πιο αδύναμο μοντέλο PRAM, σύμφωνα με την ελάχιστη ταυτόχρονη πρόσβαση στη μνήμη.
- CREW: Σε αυτήν την κατηγορία, επιτρέπεται σε πολλούς επεξεργαστές να διαβάσουν από την ίδια θέση μνήμης. Ωστόσο, οι πολλαπλές προσβάσεις εγγραφής σε μια θέση μνήμης είναι σειριακές.
- ERCW: Σε αυτήν την κατηγορία, επιτρέπεται σε πολλούς επεξεργαστές να γράψουν στην ίδια θέση μνήμης. Ωστόσο, οι πολλαπλές προσβάσεις διαβάσματος σε μια θέση μνήμης είναι σειριακές.
- CRCW: Σε αυτήν την κατηγορία, επιτρέπεται σε πολλούς επεξεργαστές να διαβάσουν και να γράψουν στην ίδια θέση μνήμης. Αυτό είναι το πιο δυνατό μοντέλο που αναφέρεται σε αρχιτεκτονική κοινής μνήμης.

Διαμοιραζόμενης Μνήμης (PRAM) SIMD



Διαμοιραζόμενης Μνήμης (PRAM) SIMD

Παράδειγμα

Να προσδιοριστεί αν ένα δεδομένο στοιχείο x βρίσκεται σε ένα αρχείο με n θέσεις.

Ακολουθιακός αλγόριθμος $\longrightarrow$ απαιτεί n βήματα.

Παράλληλος αλγόριθμος:

EREW SM SIMD με $N \leq n$ επεξεργαστές $P_1, P_2, \dots, P_N$.

Διαμοιραζόμενη Μνήμη (PRAM) SIMD

- Μετάδοση (Broadcasting) του x στους επεξεργαστές
 - 1 P_1 διαβάζει το x και το μεταδίδει στον P_2
 - 2 Οι P_1 και P_2 μεταδίδουν ταυτόχρονα το x στους P_3 και P_4 αντίστοιχα, κ.ο.κ.
 - 3 Οι P_1, P_2, P_3 και P_4 μεταδίδουν ταυτόχρονα το x στους P_5, P_6, P_7 και P_8 αντίστοιχα, κ.ο.κ.

Η μετάδοση του x σε όλους τους επεξεργαστές απαιτεί $\log_2 N$ βήματα

- Χωρισμός του αρχείου σε υποαρχεία με n/N στοιχεία
 P_i αναζητεί σε n/N στοιχεία άρα n/N βήματα

Σύνολο: $\log N + n/N$ βήματα

Διαμοιραζόμενη Μνήμη (PRAM) SIMD

Αν χρησιμοποιηθεί μια λογική μεταβλητή F για τη δήλωση ότι βρέθηκε το x , τότε

$$\log N + (n/N) \log N$$

γιατί σε κάθε βήμα της αναζήτησης απαιτείται η γνώση της F σε όλους τους επεξεργαστές (broadcasting $\log N$).

Για CREW SM SIMD:

- 1 βήμα για μετάδοση του x
- 1 βήμα για μετάδοση της F (τέλος κάθε αναζήτησης)

Άρα μόνο n/N βήματα για αναζήτηση.

Δυναμικά Δίκτυα Διασύνδεσης

Σε έναν EREW PRAM SM υπολογιστή με p επεξεργαστές και μια καθολική μνήμη m λέξεων, οι επεξεργαστές συνδέονται με τη μνήμη μέσω ενός συνόλου διακοπών οι οποίοι προσδιορίζουν τη λέξη της μνήμης που θα προσπελαστεί από κάθε επεξεργαστή. Σε έναν τέτοιο υπολογιστή η κάθε λέξη ΔΕΝ μπορεί να προσπελαστεί από περισσότερους από έναν επεξεργαστές. Προκειμένου να διασφαλιστεί η σύνδεση, ο συνολικός αριθμός διακοπών πρέπει να είναι της τάξης $\Theta(mp)$. Για ένα λογικό μέγεθος μνήμης η κατασκευή ενός δικτύου διακοπών αυτής της πολυπλοκότητας είναι πολύ ακριβό. Προκειμένου να μειωθεί η πολυπλοκότητα του δικτύου διακοπών μπορούμε να μειώσουμε την τιμή του m οργανώνοντας τη μνήμη σε banks. Ένας επεξεργαστής αλλάζει θέση μεταξύ των banks και όχι σε μεμονωμένες λέξεις μνήμης.

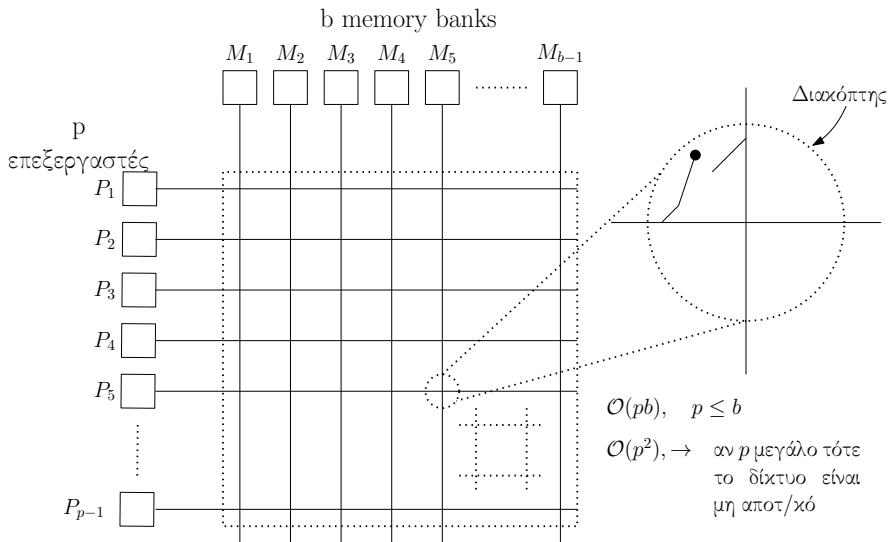
Είδη δικτύων διασύνδεσης

- Crossbar Switching Networks
- Bus-Based Networks
- Multistage Interconnection Networks

Crossbar Δίκτυα Διασύνδεσης

- Χρησιμοποιούνται για τη σύνδεση p επεξεργαστών σε b memory banks
- Η σύνδεση ενός επεξεργαστή σε ένα memory bank δεν μπορεί να εμποδίσει τη σύνδεση ενός άλλου επεξεργαστή σε κάποιο άλλο memory bank.
- Πρέπει ο κάθε επεξεργαστής να μπορεί να συνδεθεί σε οποιοδήποτε memory banks και αν όλοι θέλουν να το κάνουν, να μπορούν. Γί αυτό πρέπει $p \leq b$.

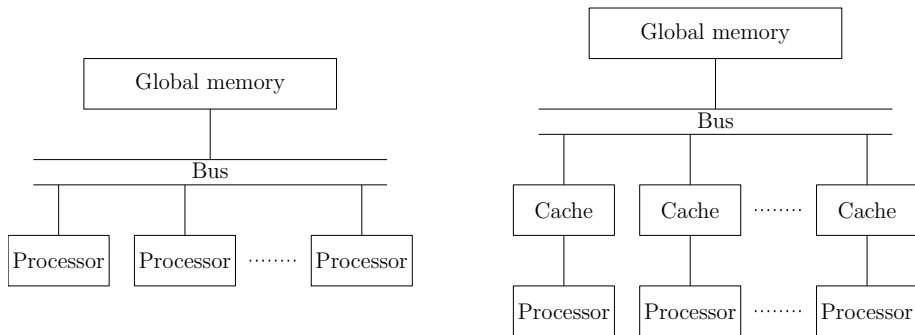
Crossbar Δίκτυα Διασύνδεσης



Σχήμα: Crossbar Switch: Cray Y-MP, Fujitsu VDD 500

- **Τυπική αρχιτεκτονική bus:** Όποτε ένας επεξεργαστής θέλει να προσπελάσει την κύρια μνήμη, κάνει αίτηση στο δίαυλο. Τα δεδομένα τότε αποσπώνται από τη μνήμη του διαύλου. Λόγω της απλότητας του είναι πολύ ελκυστικός, αλλά ο δίαυλος μπορεί να μεταφέρει μόνο ένα περιορισμένο ποσό δεδομένων μεταξύ μνήμης και επεξεργαστών. Αν αυξήσουμε τον αριθμό των επεξεργαστών, κάθε επεξεργαστής ξοδεύει αυξητικά κάποιο χρόνο αναμονής για να προσπελάσει τη μνήμη τη στιγμή που ο άλλος χρησιμοποιείται από άλλους επεξεργαστές.
- Επομένως η αποδοτικότητα του μπορεί να κορεστεί σε μικρό αριθμό επεξεργαστών.
- Η Λύση προκειμένου να ξεπεραστεί το πρόβλημα είναι σε κάθε επεξεργαστή να παρέχουμε τοπική μνήμη. Η τοπική κρυφή μνήμη μειώνει το χρόνο προσπελάσεων στην κύρια μνήμη.

Bus-based Δίκτυα Διασύνδεσης



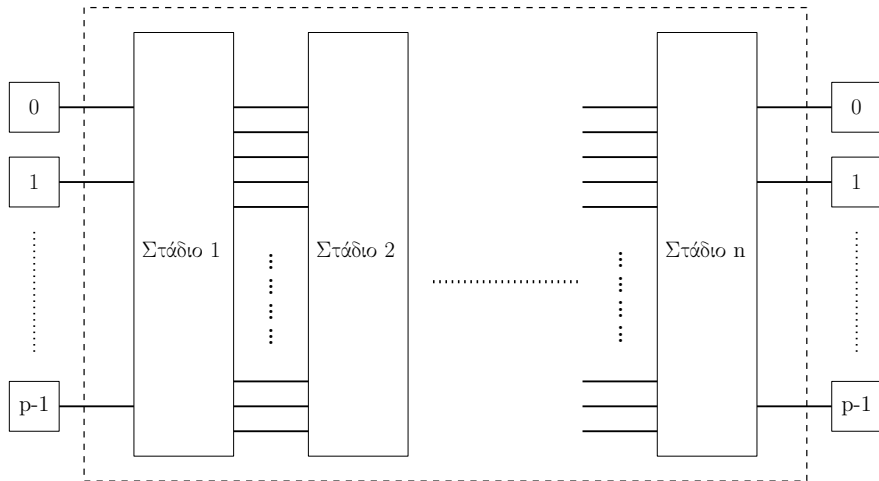
Σχήμα: Αρχιτεκτονική με bus: Symmetry, Multimax - κορεσμός με μικρό πλήθος επεξεργαστών

Bus-based Δίκτυα Διασύνδεσης

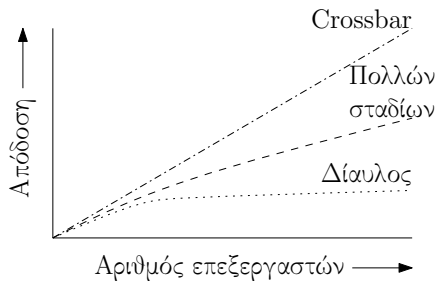
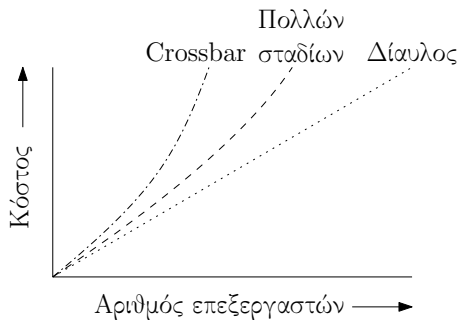
Επεξεργαστές

Δίκτυο διασύνδεσης πολλών σταδίων

Συστοιχίες μνήμης



Bus-based Δίκτυα Διασύνδεσης



Διαφορές

- Bus-based Δίκτυα Διασύνδεσης: Δεν συμφέρουν από άποψη αποδοτικότητας αλλά συμφέρουν από άποψη κόστους
- Crossbar Δίκτυα Διασύνδεσης: Δεν συμφέρουν από άποψη κόστους αλλά συμφέρουν από άποψη αποδοτικότητας
- Multistage Δίκτυα Διασύνδεσης: Ενδιαμεσα των άλλων δυο

Πολυφασικά Multistage Δίκτυα Διασύνδεσης

- Δίκτυο ωμέγα: Αποτελείται από $\log p$ φάσεις. Κάθε φάση αποτελείται από ένα πρότυπο ενδοεπικοινωνίας που συνδέει p εισόδους i και p εξόδους j , όπου

$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p, & p/2 \leq i \leq p - 1 \end{cases}$$

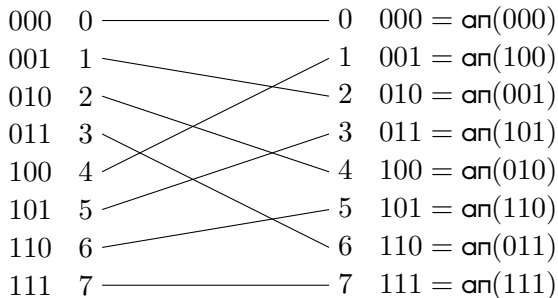
για να υπάρχει σύνδεση μεταξύ εισόδου i και εξόδου j .

Συμβολίζουμε με p τον αριθμό των επεξεργαστών, ο οποίος ισούται με τον αριθμό των memory banks.

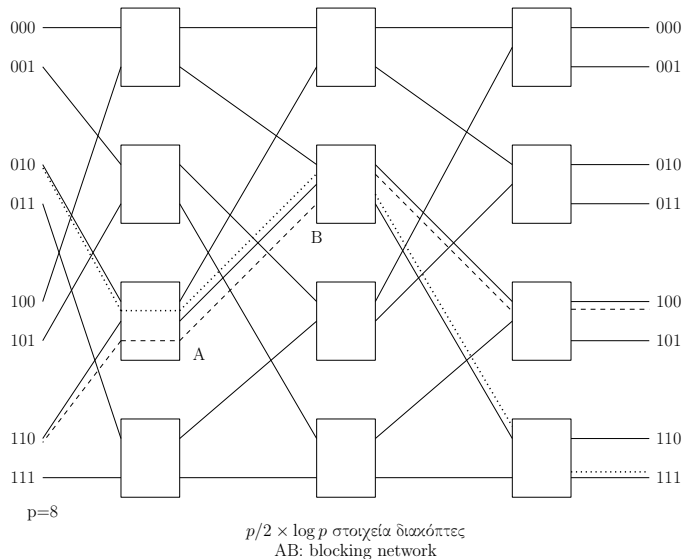
Η παραπάνω εξίσωση αναπαριστά την αριστερή περιστροφή στη δυαδική παράσταση i για να ληφθεί το j . Αυτό το σχέδιο διασύνδεσης λέγεται perfect shuffle.

Πολυφασικά Multistage Δίκτυα Διασύνδεσης

- Δίκτυο ωμέγα



Πολυφασικά Multistage Δίκτυα Διασύνδεσης



Σχήμα: Δίκτυο Ωμέγα

Πολυφασικά Multistage Δίκτυα Διασύνδεσης

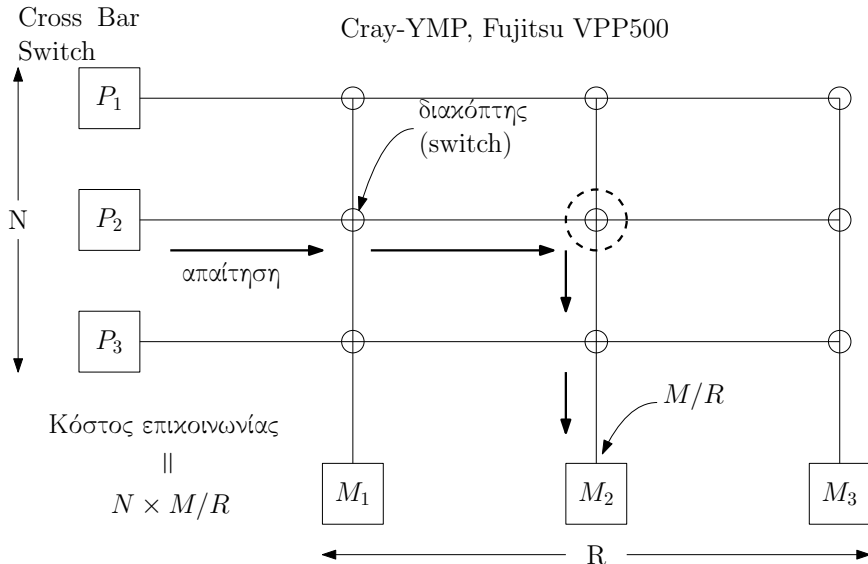
Στο σχήμα 11 φαίνεται το μήνυμα το οποίο κινείται σε ένα Ωμέγα δίκτυο 8 επεξεργαστών από τον επεξεργαστή 2 (010) στον 7 (111) και από τον επεξεργαστή 6 (110) στον 4 (100). Όταν ο επεξεργαστής 2 (010) επικοινωνεί με τον επεξεργαστή 7 (111) μπλοκάρει το μονοπάτι από τον επεξεργαστή 6 (110) στον 4 (100). Ο σύνδεσμος επικοινωνίας AB χρησιμοποιείται και από τα δυο μονοπάτια επικοινωνίας. Έτσι, σε ένα Ωμέγα δίκτυο η προσπάθεια σε ένα memory bank από έναν επεξεργαστή μπορεί να μην επιτρέψει την προσπάθεια σε κάποιο άλλο memory bank από κάποιο άλλο επεξεργαστή. Δίκτυα με αυτή την ιδιότητα λέγονται blocking δίκτυα.

Πολυφασικά Multistage Δίκτυα Διασύνδεσης

Παράλληλοι υπολογιστές: BBN Butterfly, IBM RP-3, NYU Ultracomputer

Παρατήρηση: Μόνο ένας επεξεργαστής μπορεί να διαβάσει ή να γράψει σε ένα τμήμα μνήμης.

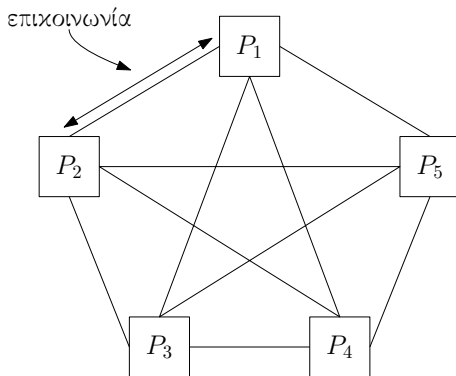
Πολυφασικά Multistage Δίκτυα Διασύνδεσης



- Οι μηνύες M_i είναι κατανεμημένες μεταξύ των N επεξεργαστών ($M_i = M/N$ θέσεις μνήμης)
- Διπλής κατεύθυνσης γραμμή επικοινωνίας μεταξύ των επεξεργαστών (μηνύματα)

Δίκτυο Διασύνδεσης

Κλίκα (clique)



Σχήμα: Κλίκα (clique)

Σύνολο $\frac{N(N-1)}{2}$ γραμμές.

Εξωπραγματικό για μεγάλο N

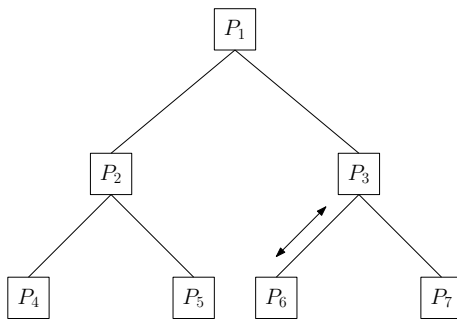
Γραμμικός πίνακας



Σχήμα: Γραμμικός πίνακας

Η επικοινωνία μεταξύ δυο επεξεργαστών γίνεται μόνο προς μια κατεύθυνση, είτε προς τα δεξιά είτε προς τα αριστερά. Εξαρτάται από το που βρίσκονται. Στο δακτύλιο η επικοινωνία μπορεί να γίνει είτε προς τη μια κατεύθυνση είτε προς την άλλη. Ο δρόμος που θα ακολουθηθεί εξαρτάται από την πολυπλοκότητα μετάδοσης του μηνύματος.

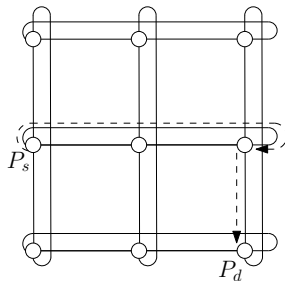
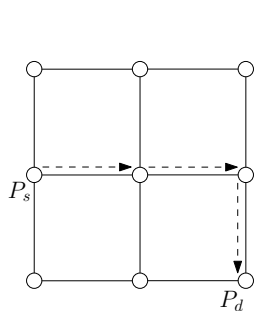
Δέντρο



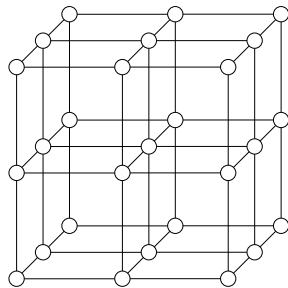
Σχήμα: Δέντρο

Η επικοινωνία γίνεται μέσω της ρίζας των υποδέντρων. Για παράδειγμα μπορούμε να πούμε ότι όλοι οι επεξεργαστές του αριστερού υποδέντρου μπορούν να επικοινωνήσουν με το δεξί υποδέντρο μέσω του P_1 .

Δίκτυο Διασύνδεσης



DAP, Paragon XP/S

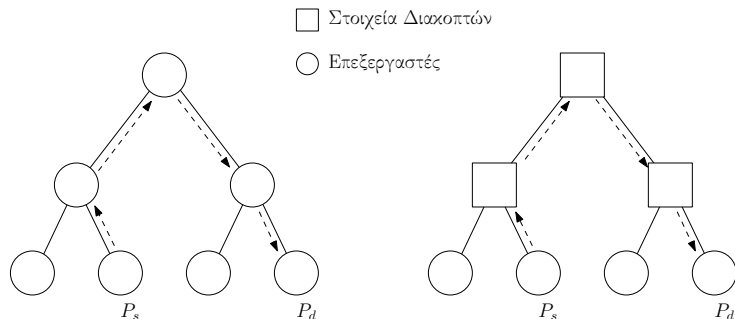


Cray T3D

- Mesh: Το μήνυμα μεταδίδεται προς τη μια διάσταση του πλέγματος μέχρι να συναντήσουμε συντεταγμένη επεξεργαστή προορισμού, οπότε αλλάζει κατεύθυνση προς τους επεξεργαστές.
- Wraparound mesh: Το διδιάστατο πλέγμα αποτελείται από δακτυλίους οριζόντιους και καθετους. Αν ο P_s θέλει να επικοινωνήσει με τον P_d θα το κάνει από τη συντομότερη διαδρομή (δεν θα περάσει από τον ενδιάμεσο επεξεργαστή).
- 3D mesh: Μπορεί να είναι είτε απλό mesh είτε Ωραπαρουνδ μεση.

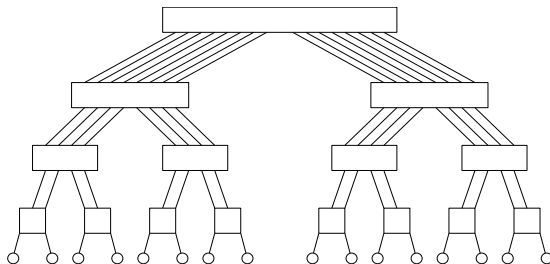
Δίκτυο Διασύνδεσης

Δίκτυο Δέντρου



- Δίκτυα Πλήρους Δυναμικού Δέντρου
- Διαδρομή Μηνύματος

Έχουμε τα στατικά δίκτυα δέντρου (αριστερό σχήμα) όπου υπάρχει μόνο ένα μονοπάτι μεταξύ κάθε ζεύγους επεξεργαστών και τα δυναμικά δίκτυα δέντρου όπου οι κόμβοι στα ενδιάμεσα επίπεδα είναι στοιχεία διακοπών και οι κόμβοι φύλλα είναι επεξεργαστές.



Σχήμα: fat tree network

Πρόβλημα: Όταν πολλοί επεξεργαστές του αριστερού υποδέντρου επικοινωνούν με επεξεργαστές του δεξιού υποδέντρου, ο κόμβος-ρίζα έχει να κρατήσει όλα τα μηνύματα.

Λύση: Αύξηση του αριθμού των συνδέσμων επικοινωνίας μεταξύ των επεξεργαστών που βρίσκονται πιο κοντά στη ρίζα.

Δίκτυο Διασύνδεσης

Υπερκύβος: Ο υπερκύβος είναι ένα πολυδιάστατο πλέγμα επεξεργαστών με ακριβώς 2 επεξεργαστές σε κάθε διασταση. Ένας d -διάστασης υπερκύβος αποτελείται από $p = 2^d$ επεξεργαστές. Ένας υπερκύβος αναδρομικά μπορεί να κατασκευαστεί ως εξής:

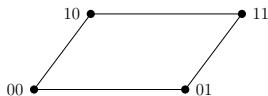
$$N = 2^q, \quad q \geq 1$$

0 •

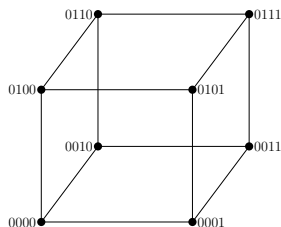
($q = 0$)

0 • ————— • 1

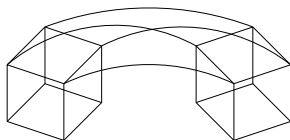
($q = 1$)



($q = 2$)



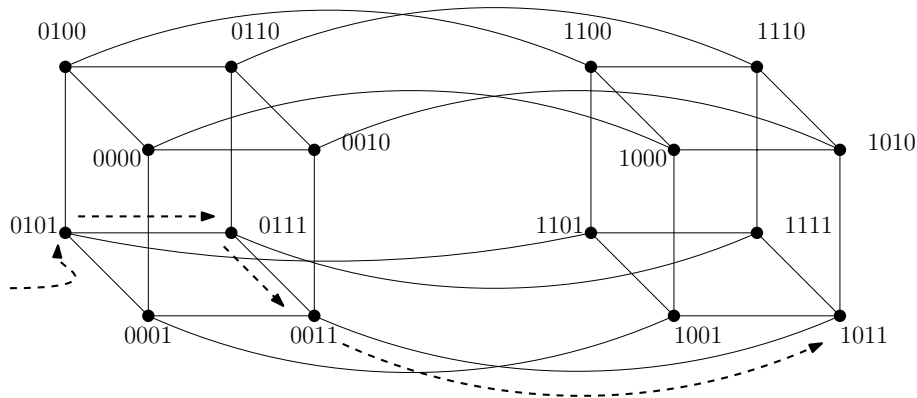
($q = 3$)



($q = 4$)

Δίκτυο Διασύνδεσης

Δίκτυο Υπερκύβου



Σχήμα: Δίκτυο Υπερκύβου

d-διάστασης υπερκύβος
 $p = 2^d$ επεξεργαστές

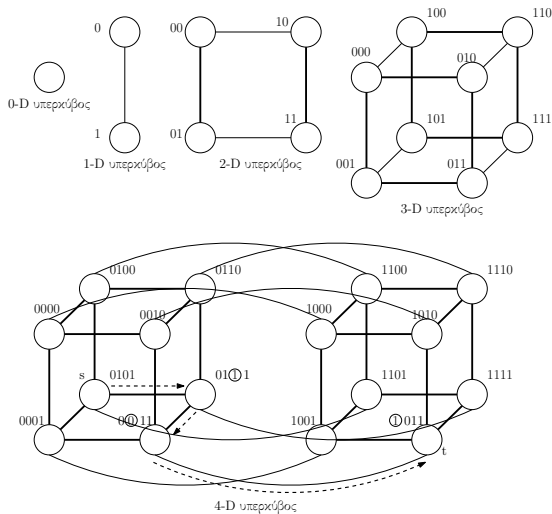
Ιδιότητες

- Δύο επεξεργαστές συνδέονται τότε και μόνο τότε αν διαφέρουν κατά ένα ακριβώς bit. Για παράδειγμα, οι επεξεργαστές 0111 και 0101 διαφέρουν κατά ένα ακριβώς bit.
- Σε έναν d -διάστασης υπερκύβο ο κάθε επεξεργαστής συνδέεται άμεσα με d άλλους επεξεργαστές. Για παράδειγμα, ένας $4D$ -διάστασης επεξεργαστής συνδέεται με 3 επεξεργαστές από τον κύβο του και έναν από τον διπλανό κύβο.
- Ο συνολικός αριθμός των bit θέσεων στα οποία διαφέρουν δύο επεξεργαστές λέγεται απόσταση του Hamming.

Παράδειγμα: Πάμε από τον $s = 0101$ στον $t = 1011$ γιατί $s \oplus t = 1110$, δηλαδή έχουμε 3 άσσους, δηλ. ξεκινάμε από την 3η θέση και η απόσταση Hamming είναι 3.

Δίκτυο Διασύνδεσης

Αρίθμηση των επεξεργαστών



Σχήμα: Απόσταση Hamming 0101 και 1011 = 3 ($s \oplus t = 1110$)

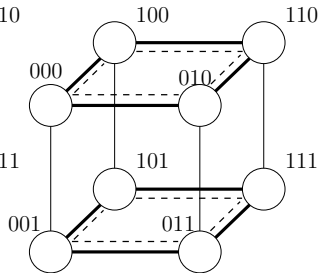
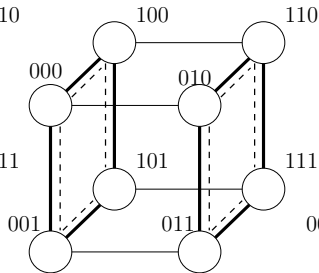
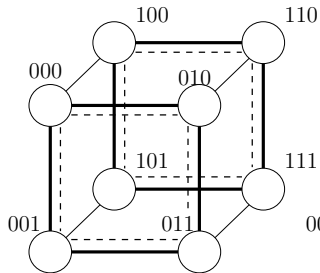
Όταν ένας υπερκύβος $d+1$ -διάστασης κατασκευάζεται με τη σύνδεση δύο υπερκύβων d -διάστασης, στις ετικέτες του ενός υπερκύβου τίθεται το πρόθεμα 0 και στον άλλο υπερκύβο το πρόθεμα 1.

Ο μόνος επεξεργαστής (γειτονικός) του s που έχει bit 1 στη θέση ένα είναι αυτός που δείχνεται στο Σχήμα, κ.ο.κ., οπότε και αυτόν τον επεξεργατή επιλέγουμε.

- Η απόσταση του Hamming μεταξύ δύο επεξεργαστών s και t είναι ο συνολικός αριθμός των bits τα οποία είναι 1 στην $s \oplus t$ (όπου $\oplus$ συμβολίζει xor).
- Ο αριθμός των συνδέσμων επικοινωνίας στο συντομότερο μονοπάτι μεταξύ δύο επεξεργαστών είναι ίσος με την απόσταση του Hamming. Ένα μήνυμα δρομολογείται από τον επεξεργαστή s στον επεξεργαστή t διαμέσου διαστάσεων οι οποίες αντιστοιχούν σε θέσεις bit που έχουν 1 στη δυαδική παράσταση του $s \oplus t$. Επειδή η δυαδική παράσταση του $s \oplus t$ μπορεί να περιέχει το πολύ d μονάδες, το συντομότερο μονοπάτι μεταξύ δύο επεξεργαστών σε ένα υπερκύβο δε μπορεί να έχει περισσότερους από d συνδέσμους.

Παράλληλοι Υπολογιστές βασισμένοι σε δίκτυο υπερκύβου είναι οι: η CUBE, Cosmic Cube και iPSC.

Δίκτυο Διασύνδεσης

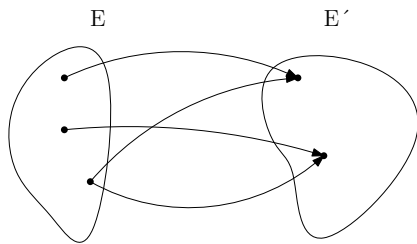
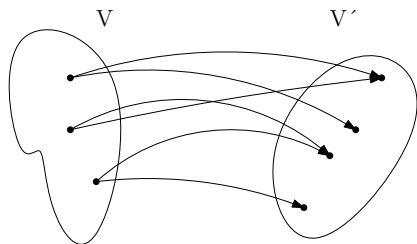


Δίκτυο Διασύνδεσης

Δίκτυο	Διάμετρος	Εύρος Διχοτόμησης	Συνδεσιμότητα Τόξου	Κόστος (αριθ. συνδέσεων)
Πλήρως συνδεδεμένο	1	$p^2/4$	$p - 1$	$p(p - 1)/2$
Αστέρας	2	1	1	$p - 1$
Πλήρες Δυαδικό Δέντρο	$2 \log((p+1)/2)$	1	1	$p - 1$
Γραμμικός πίνακας	$p - 1$	1	1	$p - 1$
Δακτύλιος	$\lfloor p/2 \rfloor$	2	2	p
2-D πλέγμα χωρίς κύρτωση	$2(\sqrt{p} - 1)$	$\sqrt{p}$	2	$2(p - \sqrt{p})$
2-D πλέγμα με κύρτωση	$2\lfloor \sqrt{p}/2 \rfloor$	$2\sqrt{p}$	4	$2p$
Υπερκύβος	$\log p$	$p/2$	$\log p$	$(p \log p)/2$
d-κύβος k-τάξης με κύρτωση	$d\lfloor k/2 \rfloor$	$2k^{d-1}$	$2d$	dp

Εμφύτευση Δικτύων στον Υπερκύβο

$$G(V, E) \xrightarrow{\text{εμφύτευση}} G'(V', E')$$



Εμφύτευση Δικτύων στον Υπερκύβο

Η εμφύτευση ενός γραφήματος εντός ενός άλλου είναι σημαντική διότι ένας αλγόριθμος που έχει σχεδιαστεί για ένα συγκεκριμένο δίκτυο διασύνδεσης μπορεί να είναι αναγκαίο να προσαρμοστεί σε ένα άλλο δίκτυο.

- συσσώρευση (congestion): Το μέγιστο πλήθος ακμών που απεικονίζονται σε μια ακμή στο E' . Δηλ. πολλές ακμές του E μπορούν να απεικονιστούν σε 1 ακμή του E'
- διαστολή (dilation): Το πλήθος συνδέσμων στο E' στους οποίους απεικονίζεται μία ακμή του E . Δηλ. 1 ακμή του E μπορεί να απεικονιστεί σε περισσότερες από 1 ακμές του E' .
- επέκταση (expansion): Επεξεργαστές στο V' / Επεξεργαστές στο V . Δηλ. κάθε επεξεργαστής του V αντιστοιχεί σε περισσότερους από έναν επεξεργαστές του V' .

Στη συνέχεια θα μελετηθούν οι εμφυτεύσεις διαφόρων δικτύων διασύνδεσης εντός του υπερκύβου. Η μελέτη θα περιοριστεί στην περίπτωση όπου το πλήθος των επεξεργαστών είναι το ίδιο στα δύο δίκτυα (επέκταση = 1). Επιπλέον το πολύ μία ακμή του E απεικονίζεται σε μία ακμή του E' (συσσώρευση = 1).

Εμφύτευση γραμμικού πίνακα σε υπερκύβο

Ένας γραμμικός πίνακας (ή ένας δακτύλιος) που αποτελείται από 2^d επεξεργαστές μπορεί να εμφυτευθεί σε ένα υπερκύβο d διάστασης απεικονίζοντας τον i – οστό επεξεργαστή του γραμμικού πίνακα στον επεξεργαστή $G(i, d)$ του υπερκύβου, όπου

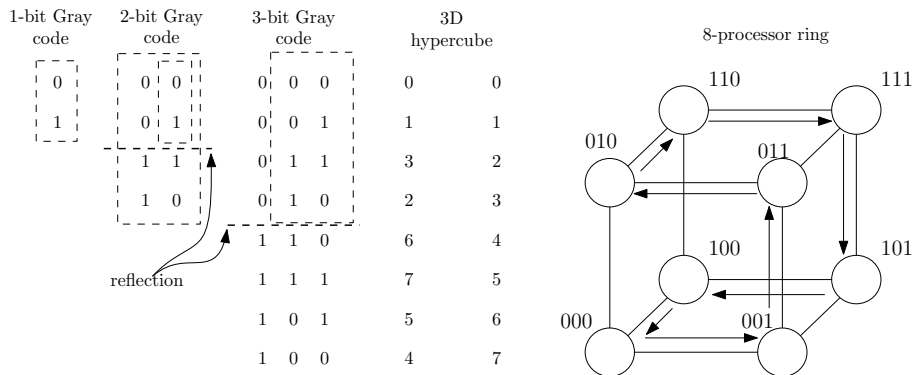
$$G(0, 1) = 0$$

$$G(1, 1) = 1$$

$$G(i, x + 1) = \begin{cases} G(i, x), & i < 2^x \\ 2^x + G(2^{x+1} - 1 - i, x), & i \geq 2^x \end{cases}$$

Εμφύτευση Δικτύων στον Υπερκύβο

Η συνάρτηση G λέγεται binary reflected Gray code (RGC). Η $G(i, d)$ συμβολίζει το i – οστό στοιχείο στην ακολουθία των κωδίκων Gray με d bits. Οι κώδικες Gray $d+1$ bits παράγονται από ένα πίνακα των κωδίκων Gray με d bits ως εξής: Παίρνοντας το συμμετρικό πίνακα και τοποθετώντας στα στοιχεία του συμμετρικού πίνακα το πρόθεμα 1 και στα αρχικά στοιχεία το πρόθεμα 0.



Σχήμα: Εμφύτευση ενός δακτυλίου οκτώ επεξεργαστών σε ένα υπερκύβο διάστασης 3

Εμφύτευση Δικτύων στον Υπερκύβο

- Τα στοιχεία $G(i, d)$ και $G(i + 1, d)$ διαφέρουν μόνο σε ένα bit.
- Ο επεξεργαστής i στο γραμμικό πίνακα απεικονίζεται στον επεξεργαστή $G(i, d)$ του υπερκύβου και ο επεξεργαστής $i + 1$ του γραμμικού πίνακα στον επεξεργαστή $G(i + 1, d)$ του υπερκύβου.
- Συνεπώς υπάρχει ένας άμεσος σύνδεσμος στον υπερκύβο, ο οποίος αντιστοιχεί σε κάθε άμεσο σύνδεσμο στο γραμμικό πίνακα (οι ετικέτες δύο γειτονικών επεξεργαστών διαφέρουν μόνο κατά ένα bit).
- Η απεικόνιση που ορίζεται από την G έχει συσσώρευση ένα και διαστολή ένα.

Εμφύτευση Δικτύων στον Υπερκύβο

Εμφύτευση διδιάστατου δικτύου (mesh) σε υπερκύβο

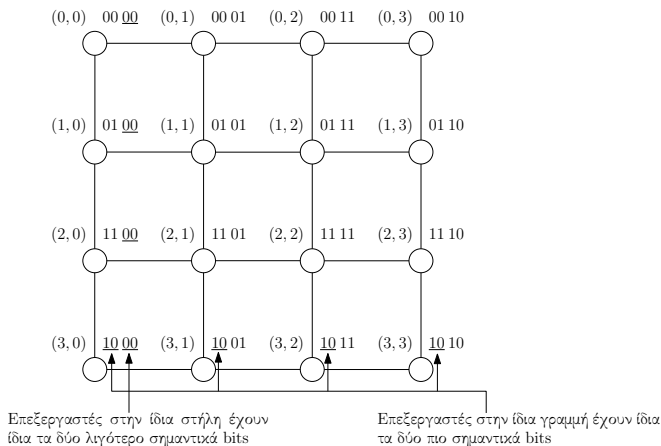
- Μπορούμε να εμφυτεύσουμε ένα $2^r \times 2^s$ wraparound mesh σε έναν 2^{r+s} -επεξεργαστών υπερκύβο απεικονίζοντας τον επεξεργαστή (i, j) του mesh στον επεξεργαστή $G(i, r) || G(j, s)$ του υπερκύβου, όπου $||$ συμβολίζει τη συνένωση των δύο κωδίκων Gray.
- Οι άμεσοι γείτονες στο mesh απεικονίζονται σε επεξεργαστές του υπερκύβου των οποίων οι επικέτες διαφέρουν μόνο σε ένα bit). Συνεπώς η απεικόνιση αυτή έχει μία διάδοση = 1 και μία συσσώρευση = 1.

Εμφύτευση Δικτύων στον Υπερκύβο

$$r = 1, s = 2, \quad (i, j) \longrightarrow G(i, 1) || G(j, 2)$$

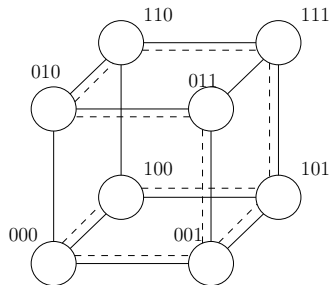
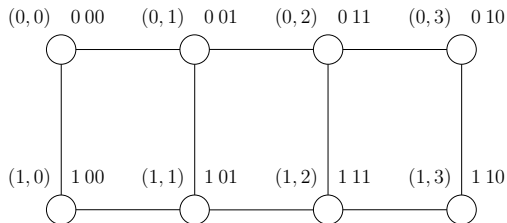
$$(0, 0) \longrightarrow G(0, 1) || G(0, 2) = 0 || 00 = 000$$

$$(0, 1) \longrightarrow G(0, 1) || G(1, 2) = 0 || 01 = 001$$



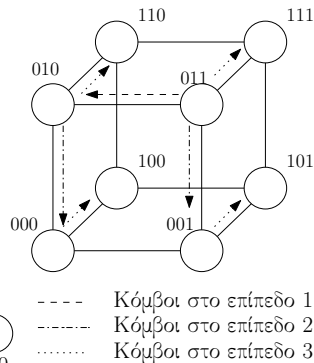
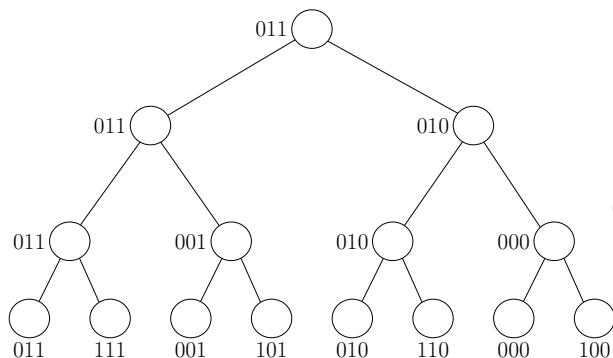
Σχήμα: Εμφύτευση 2×4 mesh

Εμφύτευση Δικτύων στον Υπερκύβο



Σχήμα: Εμφύτευση 2×4 mesh

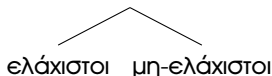
Εμφύτευση Δικτύων στον Υπερκύβο



Δρομολόγηση Μηνυμάτων

Ένας μηχανισμός δρομολόγησης καθορίζει το μονοπάτι που θα ακολουθήσει το μήνυμα για να πάει από τον πηγαίο επεξεργαστή P_s στον επεξεργαστή προορισμού P_d .

Μηχανισμοί Δρομολόγησης



- Οι ελάχιστοι ακολουθούν το συντομότερο δρόμο μεταξύ P_s και P_d .
- Οι μη ελάχιστοι δεν ακολουθούν τη συντομότερη αλλά την μακρύτερη διαδρομή προκειμένου να αποφευχθεί συμφόρηση του δικτύου.

Εμφύτευση Δικτύων στον Υπερκύβο

Δρομολόγηση Μηνυμάτων

Ένας μηχανισμός δρομολόγησης καθορίζει το μονοπάτι που θα ακολουθήσει το μήνυμα για να πάει από τον πηγαίο επεξεργαστή P_s στον επεξεργαστή προορισμού P_d .

Μηχανισμοί Δρομολόγησης

Προσδιοριστικοί Προσαρμοστικοί

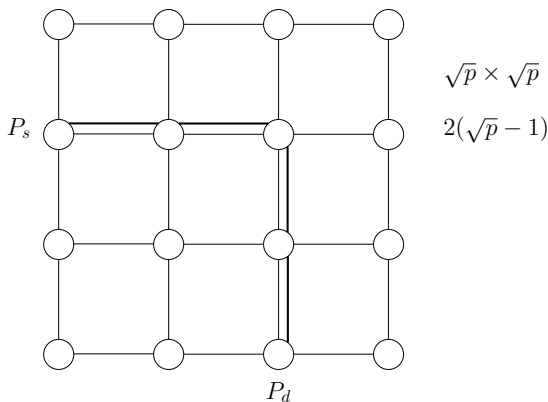
- Οι προσδιοριστικοί καθορίζουν ένα μοναδικό μονοπάτι για το μήνυμα βασισμένο στους κόμβους πηγής και προορισμού.
- Οι προσαρμοστικοί παίρνουν πληροφορία κοιτάζοντας την τρέχουσα κατάσταση του δικτύου για να καθορίσουν το μονοπάτι του μηνύματος

πηγή (source)

προορισμός (destination)



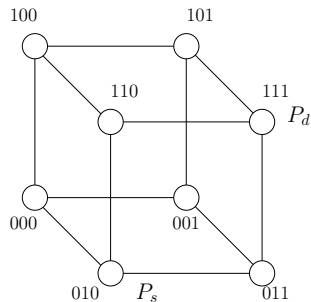
Εμφύτευση Δικτύων στον Υπερκύβο



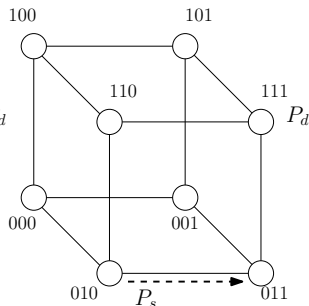
Σχήμα: XY - δρομολόγηση σε mesh

Το μήνυμα στέλνεται πρώτα στη Χ διάσταση μέχρι να φτάσει τη στήλη του επεξεργαστή προορισμού και στη συνέχεια πάει κατά μήκος της διάστασης Υ μέχρι να φτάσει στον προορισμό του.

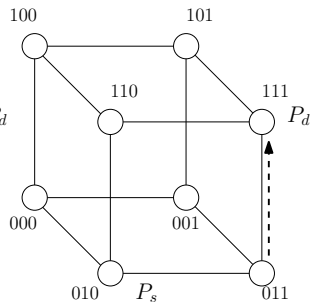
Εμφύτευση Δικτύων στον Υπερκύβο



$$\begin{aligned} 010 \oplus 110 &= 1\textcircled{0}\textcircled{1} \\ 010 \oplus 111 &= 101 \\ 010 \oplus 011 &= 001 \\ 010 \oplus 000 &= 010 \\ 010 \oplus 110 &= 100 \end{aligned}$$



$$\begin{aligned} \text{Βήμα 1 } (010 \rightarrow 01\textcircled{1}) \\ 010 \oplus 111 &= 0\textcircled{1}0 \\ 011 \oplus 111 &= 100 \end{aligned}$$



$$\begin{aligned} \text{Βήμα 2 } (011 \rightarrow 111) \\ 011 \oplus 111 &= \end{aligned}$$

Σχήμα: E-cube δρομολόγηση

Εμφύτευση Δικτύων στον Υπερκύβο

Κόστος Επικοινωνίας

- καθυστέρηση επικοινωνίας (*latency*)

Ο απαιτούμενος χρόνος για την επικοινωνία ενός μηνύματος μεταξύ δύο επεξεργαστών.

- χρόνος ξεκινήματος (*startup time*) (t_s)

Ο απαιτούμενος χρόνος για τη μεταχείριση του μηνύματος (προετοιμασία).

- *per-hop* χρόνος (t_h)

Ο απαιτούμενος χρόνος για να φθάσει η επικεφαλίδα του μηνύματος στον προορισμό της.

- χρόνος ανά *word* (t_w)

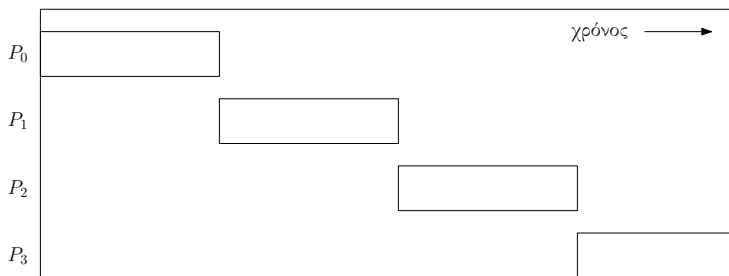
$$t_w = 1/r$$

όπου r = αριθμός words ανά δευτερόλεπτο

Εμφύτευση Δικτύων στον Υπερκύβο

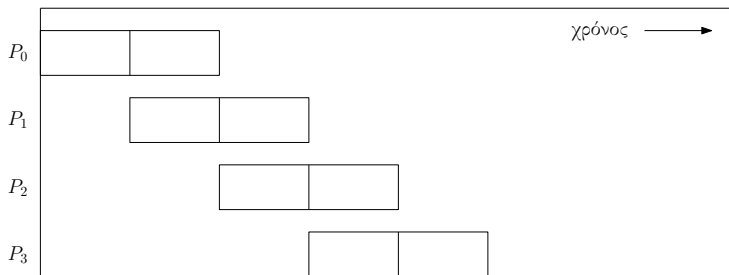
Καθυστέρηση Επικοινωνίας

τοπολογία δικτύου τεχνικές μεταφοράς μηνυμάτων



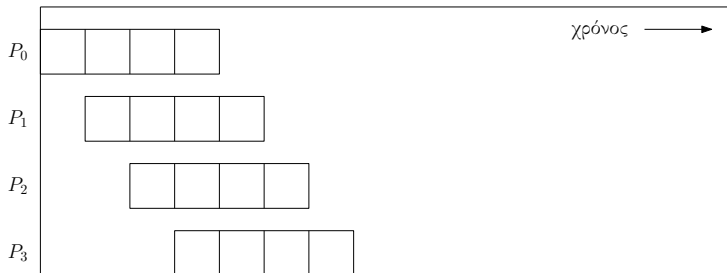
Σχήμα: Store and forward. Προώθηση μηνύματος μετά τη λήψη και την αποθήκευση

Εμφύτευση Δικτύων στον Υπερκύβο



Σχήμα: Cut through. Το μήνυμα κόβεται στα 2 ώστε ο P_0 όταν αρχίζει να λαμβάνει, το 2ο κομμάτι του θα το έχει πάρει ο P_1 για μετάδοση. Οπότε κόβουμε τον ανενεργό χρόνο του P_1 στη μέση.

Εμφύτευση Δικτύων στον Υπερκύβο



Σχήμα: Cut through. Το μήνυμα κόβεται στα 4. Όσο μικρότερα τα τμήματα, τόσο μικρότερος και ο ανενεργός χρόνος.

Υπολογισμός κόστους επικοινωνίας

$$t_{comm} = t_s + (m t_w + t_h) l$$

όπου m = πλήθος words, l = πλήθος συνδέσμων.

Ένα μήνυμα μεγέθους m μεταφέρεται μέσω ενός δικτύου. Έστω ότι διασχίζει ℓ συνδέσμους (επεξεργαστές). Σε κάθε σύνδεσμο, το μήνυμα έχει κόστος t_h για τη μεταφορά της επικεφαλίδας και $m \cdot t_w$ το κόστος για να περάσει το υπόλοιπο σώμα του μηνύματος άλλο σύνδεσμο. Αφού έχουμε ℓ τέτοιους συνδέσμους το κόστος επικοινωνίας ενός μηνύματος μεγέθους m για την store and forward δρομολόγηση θα είναι:

$$t_{comm} = t_s + m t_w l, \quad \mathcal{O}(ml) \quad \text{store and forward}$$

Το μήνυμα περνάει ℓ συνδέσμους. Με t_h συμβολίζουμε το χρόνο της επικεφαλίδας. Η επικεφαλίδα του μηνύματος θέλει $\ell \cdot h$ χρόνο να φτάσει στον επεξεργαστή P_d . Αν το μήνυμα έχει μέγεθος m λέξεις θα φτάσει σε χρόνο $m \cdot t_w$ μετά την άφιξη της επικεφαλίδας. Άρα για την cut through δρομολόγηση ο χρόνος επικοινωνίας θα είναι:

$$t_{comm} = t_s + l t_h + m t_w, \quad \mathcal{O}(l + m) \quad \text{cut through}$$

Βασικές Λειτουργίες Επικοινωνίας

- Οι σύνδεσμοι επικοινωνίας είναι διπλής κατεύθυνσης.
- Ο κάθε επεξεργαστής μπορεί να στείλει ένα μήνυμα σε ένα μόνο από τους συνδέσμους του κάθε φορά.
- Όμοια μπορεί να λάβει ένα μήνυμα από ένα σύνδεσμο κάθε φορά.
- Μπορεί να λάβει ένα μήνυμα ενώ στέλνει ένα άλλο την ίδια στιγμή στον ίδιο ή διαφορετικό σύνδεσμο.

SF δρομολόγηση

- $t_s + t_w m \lfloor p/2 \rfloor$ δακτύλιο
- $t_s + 2t_w m \lfloor p/2 \rfloor$ δίκτυο
- $t_s + t_w m \log p$ υπερκύβο

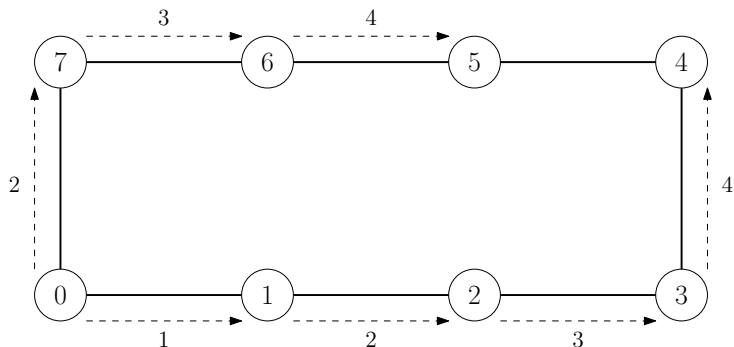
CT δρομολόγηση

$$t_s + m t_w + t_h l$$

Ένας προς όλους Μετάδοση

Δακτύλιος SF

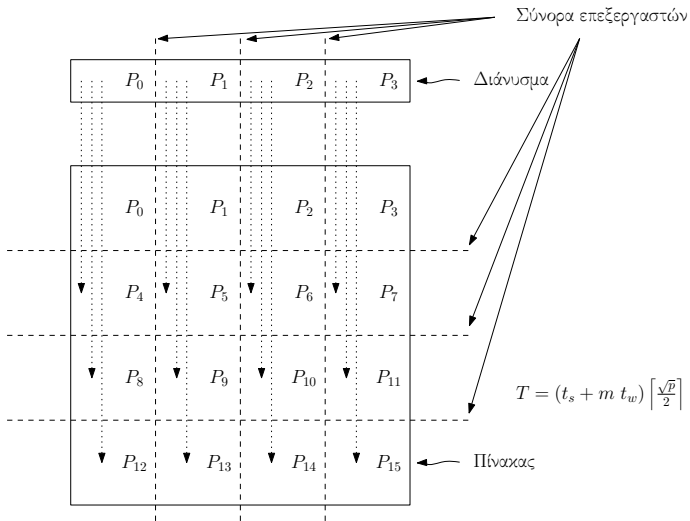
Βασικές Λειτουργίες Επικοινωνίας



Σχήμα: Δακτύλιος, $T = t_s + t_w m \lceil p/2 \rceil$

- Καθεμία από τις $\lceil p/2 \rceil$ επικοινωνίες με κοντινό γείτονα παίρνει $t_s + t_w m$ χρόνο.
- Ο επεξεργαστής 0 στέλνει στον 1, ο 1 στέλνει στον 2 και ο 0 στον 7, ο 2 στέλνει στον 3 και ο 7 στον 6. Τέλος ο 3 στέλνει στον 4 και ο 6 στον 5.

Βασικές Λειτουργίες Επικοινωνίας

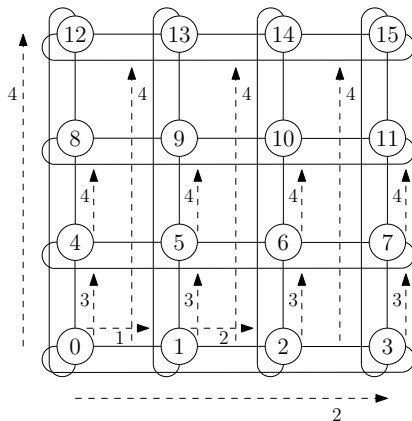


Σχήμα: Ένας προς όλους μετάδοση για τον πολλαπλασιασμό ενός 4×4 πίνακα με ένα 4×1 διάνυσμα

Βασικές Λειτουργίες Επικοινωνίας

$$T = 2(t_s + t_w m) \left\lceil \frac{\sqrt{p}}{2} \right\rceil$$

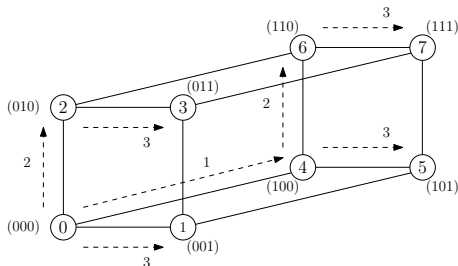
$$T = 3(t_s + t_w m) \left\lceil \frac{p^{1/3}}{2} \right\rceil \quad 30 \text{ mesh}$$



Σχήμα: Ένας προς όλους μετάδοση σε πλέγμα 16 επεξεργαστών με store and forward δρομολόγηση

Βασικές Λειτουργίες Επικοινωνίας

$$T = (t_s + t_w m) \log p$$

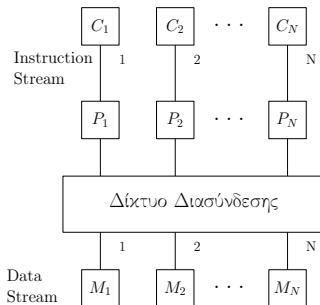


Σχήμα: Ένας προς όλους μετάδοση σε 3-D υπερκύβο

Στο σχήμα αυτό η επικοινωνία ξεκινάει κατά μήκος της υψηλότερης διάστασης (δηλ. της διάστασης που καθορίζεται από το περισσότερο σημαντικό *bit* στην δυαδική παράσταση της ετικέτας)

Κάθε ένα από τα $\log p$ βήματα θέλει $t_s + t_w m$ χρόνο για τη μεταφορά ενός απλού μηνύματος σε κάθε διάσταση.

MIMD Υπολογιστές



Σχήμα: Multiple Instruction Multiple Data

Είναι υπολογιστής όπου ο κάθε επεξεργαστής είναι ικανός να εκτελέσει διαφορετικά προγράμματα αναξάρτητα από τους άλλους επεξεργαστές. Αποθηκεύουν το πρόγραμμα και το λειτουργικό σύστημα σε κάθε επεξεργαστή.

- Σύγχρονη λειτουργία
- Ασύγχρονη λειτουργία

Αξιολόγηση MIMD Παράλληλων Συστημάτων

- Scheduling

MIMD

Διαμοιραζόμενης Μνήμης
(tightly coupled)

Κατανεμημένης Μνήμης
(loosely coupled)

Ανάλυση Παράλληλων Αλγορίθμων

$$T_p = T_{comp} + T_{comm}$$

Απόδοση Παράλληλου Προγραμματισμού

- Βασικός στόχος του παράλληλου αλγόριθμού είναι η ταχύτερη επίλυση ενός προβλήματος. Ο χρόνος εκτέλεσης του παράλληλου αλγόριθμου πρέπει να είναι πολύ μικρότερος από τον χρόνο εκτέλεσης του σειριακού αλγόριθμου.
- Η απόδοση παράλληλων αλγορίθμων – προγραμμάτων μπορεί να μετρηθεί με εμπειρικούς δείκτες όπως ο λόγος επιτάχυνσης.
- Θεωρητικοί νόμοι για διαφορετικούς τύπους προβλημάτων:
 - 1 Νόμος του Amdahl
 - 2 Νόμος του Gustafson

Απόδοση Παράλληλου Προγραμματισμού

- Η παράλληλη επεξεργασία προϋποθέτει τον καταμερισμό ενός προβλήματος (ή μιας εργασίας) σε μικρότερες διεργασίες που μπορούν να εκτελεστούν ταυτόχρονα (παράλληλα).
- Επιμέριση (Granularity): το μέγεθος της εργασίας (εντολών) κάθε διεργασίας.
- Στόχος είναι η επίτευξη μέσης επιμέρισης:
 - ❶ Μικρή επιμέριση: Μικρός φόρτος εργασίας σε κάθε επεξεργαστή, πολύ επικοινωνία.
 - ❷ Μεγάλη επιμέριση: Μεγάλος φόρτος εργασίας σε κάθε επεξεργαστή, λίγη επικοινωνία.
- Βασικός στόχος η μεγιστοποίηση του λόγου:

$$\frac{\text{Χρόνος Υπολογισμού}}{\text{Χρόνος Επικοινωνίας}} = \frac{T_{comp}}{T_{comm}}$$

Speedup Factor (Λόγος Επιτάχυνσης)

Λόγος επιτάχυνσης, S_p , είναι ένα μέτρο της απόδοσης ενός προγράμματος σε ένα παράλληλο υπολογιστικό σύστημα:

$$S_p = \frac{\text{χρόνος εκτέλεσης σε ένα επεξεργαστή}}{\text{χρόνος εκτέλεσης σε επεξεργαστές}} = \frac{T_1}{T_p}$$

- Απόλυτος λόγος επιτάχυνσης: αν T_1 είναι ο χρόνος του καλύτερου σειριακού αλγόριθμου.
- Σχετικός λόγος επιτάχυνσης: αν T_p είναι ο χρόνος του παράλληλου αλγόριθμου σε 1 επεξεργαστή.
- Βέλτιστη επιτάχυνση είναι η γραμμική: $S_p = p$
- Υπεργραμμική επιτάχυνση ($S_p > p$): μπορεί να εμφανιστεί σε μικρό αριθμό επεξεργαστών. Οφείλεται:
 - 1 είτε στην διαφορετική χρήση-ιεραρχία εσωτερικής μνήμης στους διαφορετικούς επεξεργαστές (cache effect),
 - 2 είτε σε αυτόματη βελτιστοποίηση (οπτιμίζαριον) του σειριακού κώδικα από τον compiler.

Speedup Factor (Λόγος Επιτάχυνσης)

Ο λόγος επιτάχυνσης, είναι ένα απλό και κατανοητό μέτρο απόδοσης.

- Αν ο λόγος επιτάχυνσης είναι μεγάλος (S_p είναι τάξης p) τότε ο παράλληλος αλγόριθμος είναι καλός.
- Αν ο λόγος επιτάχυνσης είναι μικρός ($S_p \ll p$) τότε:
 - 1 είτε το πρόβλημα δεν παραλληλίζεται,
 - 2 είτε ο παράλληλος αλγόριθμος χρειάζεται βελτίωση!

Προσοχή:

- Ο μέγιστος λόγος επιτάχυνσης δεν έχει νόημα αν είναι $\ll p$.
- Καθώς ο αριθμός των επεξεργαστών, n , αυξάνεται, το σειριακό κομμάτι του κώδικα αποτελεί ολοένα και μεγαλύτερο τμήμα του συνολικού χρόνου εκτέλεσης (Νόμος του Amdahl).

Speedup Factor (Λόγος Επιτάχυνσης)

Είναι φανερό ότι $S_p \geq 1$. Επίσης ένα βήμα ενός παράλληλου αλγορίθμου χρησιμοποιώντας p επεξεργαστές χρειάζεται p βήματα το πολύ όταν υλοποιηθεί σειριακά. Άρα

$$T_1 \leq p T_p$$

Συνεπώς

$$1 \leq S_p \leq p$$

Νόμος του Amdahl (1967)

- Έστω ότι για μια διεργασία απαιτείται σε ένα επεξεργαστή χρόνος t_s .
Έστω ακόμη ότι ένα μέρος (ποσοστό) f της διεργασίας είναι σειριακό ενώ το υπόλοιπο $1 - f$ παραλληλίζεται. Τότε για p επεξεργαστές ο λόγος επιτάχυνσης είναι:

$$S_p = \frac{T_1}{T_p} = \frac{t_s}{f t_s + \frac{(1-f)t_s}{p}} = \frac{p}{1 + (p-1)f} \leq \frac{1}{f}$$

$$T_1 = t_s, \quad T_p = T_{seq} + T_{par}, \quad T_{seq} = f T_1 \quad \text{και} \quad T_{par} = (1-f) T_1$$

- Το όριο για $p \rightarrow \infty$ (μέγιστη επιτάχυνση) είναι: $S_p^{max} = \lim_{p \rightarrow \infty} S_p = \frac{1}{f}$.
- Προσοχή: ακόμη και αν μόνο το 10% του προβλήματος είναι σειριακό τότε $S_p \leq 10$ για οποιοδήποτε p . Επομένως, $S_p^{max} = 10$!
- Μεγάλο κομμάτι της παράλληλης επεξεργασίας αφορά την ελαχιστοποίηση του f .

Αποδοτικότητα, Επεκτασιμότητα

Αποδοτικότητα (Efficiency) :

$$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}$$

όπου

$$0 \leq E_p \leq 1.$$

- Αλγόριθμοι με σχεδόν γραμμική επιτάχυνση: $E \simeq 1$
- Πολλοί δύσκολα παραλληλίσσιμοι αλγόριθμοι έχουν:

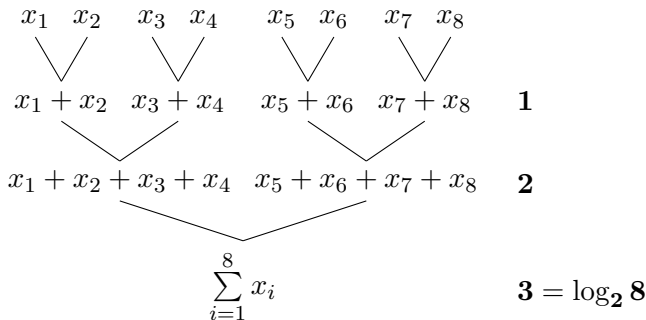
$$E = 1/\log p, E \rightarrow 0 \text{ με } p \rightarrow \infty$$

- 1 Επεκτασιμότητα υλικού είναι η δυνατότητα αναβάθμισης των συστημάτων (hardware) με αποτέλεσμα την αύξηση της απόδοσης.
- 2 Όμως περισσότεροι επεξεργαστές $\rightarrow$ μεγαλύτερο δίκτυο $\rightarrow$ αύξηση της επικοινωνίας $\rightarrow$ πτώση της απόδοσης!
- 3 Επεκτασιμότητα (Scalability) αλγορίθμου είναι η δυνατότητα του λογισμικού (software) να διαχειριστεί μεγαλύτερα προβλήματα και άρα μεγαλύτερο όγκο δεδομένων με σχετικά μικρή αύξηση στο κόστος του αλγορίθμου.

Αξιολόγηση MIMD Παράλληλων Συστημάτων

Παράδειγμα (SIMD)

Υπολογισμός του $\sum_{i=1}^N x_i$



Αξιολόγηση MIMD Παράλληλων Συστημάτων

Αριθμός Επεξεργαστών $p = \frac{N}{2}$ CREW
 $\log N$ βήματα

$$S_p = \frac{T_1}{T_p} = \frac{N-1}{\log N} \approx \frac{N}{\log N}$$

$$E_p \approx \frac{2}{\log N} \rightarrow 0 \quad \text{για} \quad N \rightarrow \infty$$

Ανάλογος αλγόριθμος για προβλήματα

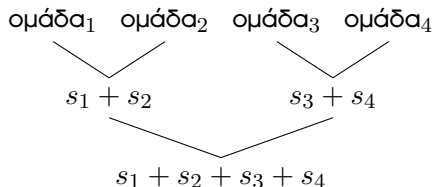
- εσωτερικο γινόμενο διανυσμάτων
- εύρεση μέγιστου - ελάχιστου αριθμού κ.α.

Αξιολόγηση MIMD Παράλληλων Συστημάτων

Παράδειγμα

Υπολογισμός του $\sum_{i=1}^N x_i$ με τη χρήση p επεξεργαστών $1 \leq p \leq N$.

- ❶ Χωρισμός των N αριθμών σε p ομάδες. Κάθε ομάδα περιέχει το πολύ $\left\lceil \frac{N}{p} \right\rceil$ αριθμούς
- ❷ Καταχώρηση μιας ομάδας σε κάθε ένα από τους p επεξεργαστές
- ❸ Υπολογισμός του αθροίσματος των αριθμών σε κάθε ομάδα σειριακά σε $\left\lceil \frac{N}{p} \right\rceil - 1$ βήματα
- ❹ Υπολογισμός του αθροίσματος των επιμέρους αθροισμάτων (σε μορφή δυαδικού δέντρου)



Αξιολόγηση MIMD Παράλληλων Συστημάτων

$$T_p = \left\lceil \frac{N}{p} \right\rceil - 1$$

$$S_p = \frac{\overbrace{N-1}^{T_1}}{\lceil N/p \rceil + \lceil \log p \rceil - 1} \quad \text{Av} \quad N = L p \log p$$

$$S_p = \frac{L p}{L + 1} \rightarrow \lim_{L \rightarrow \infty} S_p = p \quad \text{γραμμική στο } p.$$

Επίσης

$$E_p = \frac{1}{L + 1} \rightarrow \lim_{L \rightarrow \infty} E_p = 1$$

Αξιολόγηση MIMD Παράλληλων Συστημάτων

Παράδειγμα

Δίνονται τα $X = (x_1, x_2, \dots, x_N)$, $Y = (y_1, y_2, \dots, y_N)$. Να υπολογιστεί το

$$\sum_{i=1}^N x_i y_i.$$

Έστω ότι το p διαιρεί το N . Τότε

$$\sum_{i=1}^N x_i y_i = \sum_{i=1}^{N/p} x_i y_i + \sum_{i=N/p+1}^{2(N/p)} x_i y_i + \cdots + \sum_{i=(N/p)(k-1)+1}^{k(N/p)} x_i y_i + \cdots + \sum_{i=(N/p)(p-1)+1}^N x_i y_i$$

Άρα ο k επεξεργαστής υπολογίζει

$$\sum_{i=(N/p)(k-1)+1}^{k(N/p)} x_i y_i, \quad k = 1, 2, \dots, p$$

Αξιολόγηση MIMD Παράλληλων Συστημάτων

Για τον ανωτέρω υπολογισμό απαιτούνται

$$\frac{N}{p} \quad \text{πολ/σμοί}$$

$$\frac{N}{p} - 1 \quad \text{προσθέσεις}$$

Αξιολόγηση MIMD Παράλληλων Συστημάτων

Τα μερικά αθροίσματα προστίθενται με τη μορφή του δυαδικού δέντρου σε $\lceil \log p \rceil$ βήματα. Άρα

$$T_p = 2 \left(\frac{N}{p} - 1 \right) + \lceil \log p \rceil$$

και

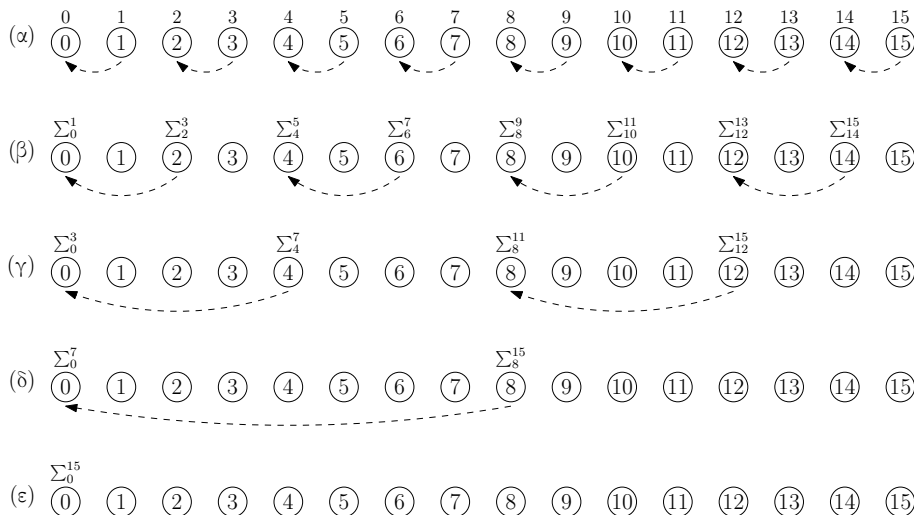
$$S_p = \frac{2N - 1}{\left(2 \frac{N}{p} - 1 \right) + \lceil \log p \rceil}$$

Αν $p = N$ τότε $S_p = \frac{2N - 1}{\lceil \log N \rceil} + 1$.

Αν $p = 2^k$ και $N = L p \log p$ τότε

$$S_p = \frac{2L}{2L + 1} p \rightarrow \lim_{L \rightarrow \infty} S_p = p \quad !$$

Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



Σχήμα: Πρόσθεση n αριθμών σε ένα Υπερκύβο με n επεξεργαστές

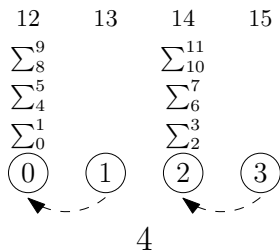
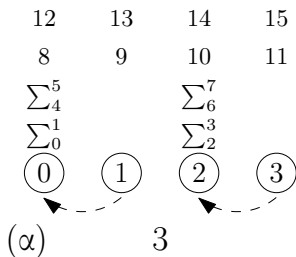
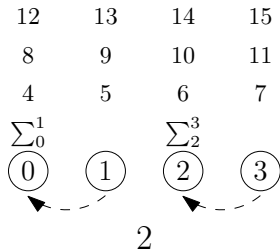
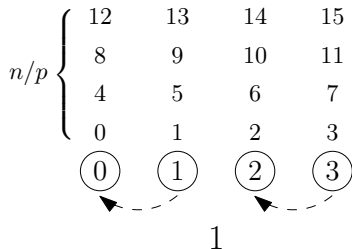
Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές

$$T_p = \Theta(\log n), \quad S_p = \Theta\left(\frac{n}{\log n}\right)$$

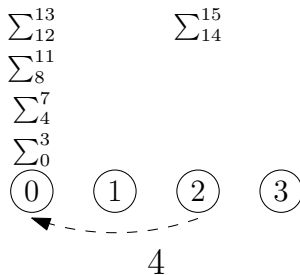
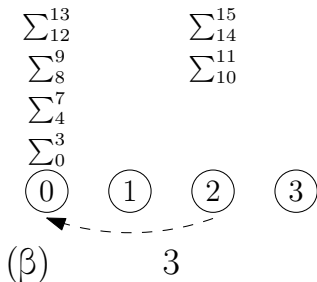
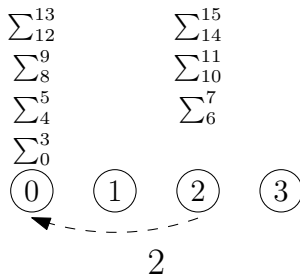
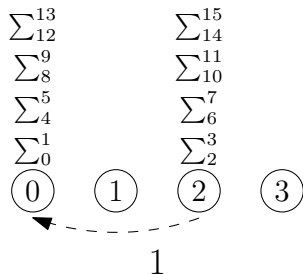
$$E_p = \Theta\left(\frac{1}{\log n}\right), \quad c_p = \Theta(n \log n)$$

όχι βέλτιστο κόστος

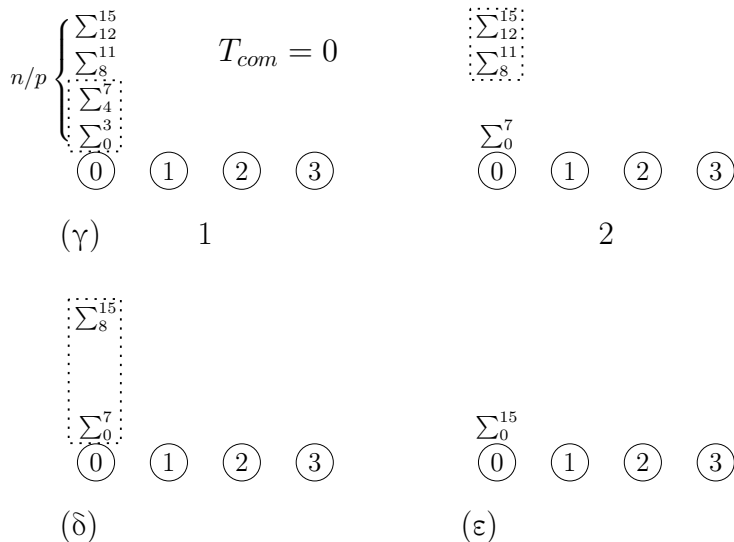
Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



Σχήμα: Πρόσθεση n αριθμών σε υπερκύβο με p επεξεργαστές, όπου $p < n$. Τα n, p είναι δυνάμεις του 2

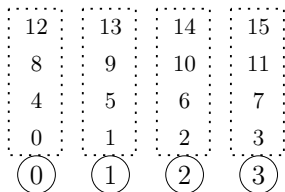
Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές

$$T_p = \Theta \left(\left(\frac{n}{p} \right) \log p \right)$$

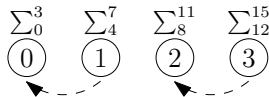
$$c_p = \Theta(n \log p) > \Theta(n) = c_1$$

όχι βέλτιστο κόστος

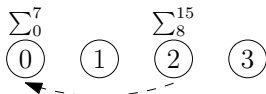
Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



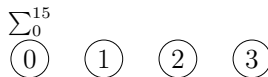
(α)



(β)



(γ)



(δ)

Σχήμα: Διαφορετικός τρόπος για την εύρεση του προηγούμενου αθροίσματος

Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές

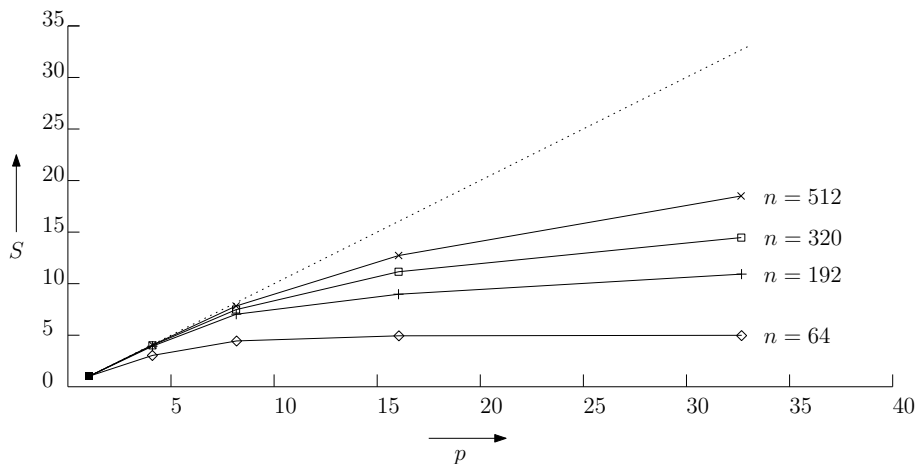
$$T_p = \Theta(n/p + \log p), \quad c_p = \Theta(n + p \log p)$$

$$\text{Αν } n = \Omega(p \log p) \text{ τότε } c_p = \Theta(n) !$$

$$T_p = \frac{n}{p} + 2t_{com} \log p, \quad S = \frac{np}{n + 2p \log p}$$

$$E_p = \frac{n}{n + 2p \log p} \quad \begin{cases} c_p = \mathcal{O}(n + 2p \log p) \\ n = \Omega(p \log p) \end{cases} \quad \text{βέλτιστο κόστος}$$

Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές



$$n = 8p \log p$$

Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές

n	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
64	1.0	<u>.80</u>	.57	.33	.17
192	1.0	.92	<u>.80</u>	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	<u>.80</u>	.62

Πίνακας: E_p

Πρόσθεση n αριθμών σε Υπερκύβο με n επεξεργαστές

- Η ταχύτητα δεν αυξάνει γραμμικά όταν αυξάνει το πλήθος των επεξεργαστών
- Για μεγαλύτερα προβλήματα η ταχύτητα και η αποδοτικότητα αυξάνουν

$$N = 2^n, n \geq 1. x_1, x_2, \dots, x_N \rightarrow M_1, M_2, \dots, M_N$$

Το τελικό αποτέλεσμα στην M_1 .

Νόμος του Gustafson (1988)

- Ο νόμος του Amdahl θέτει ένα πάνω όριο στην μέγιστη επιτάχυνση που μπορεί να επιτευχθεί σε ένα πρόγραμμα μέσω παραλληλισμού. Ο λόγος είναι ότι το σειριακό ποσοστό f είναι σταθερό.
- Ο νόμος του Gustafson προτείνει ότι ο χρόνος εκτέλεσης του σειριακού μέρους είναι σταθερός.
- Έστω n το μέτρο μεγέθους (*measure*) ενός προβλήματος. Έστω ακόμη ότι η εκτέλεση του σε ένα παράλληλο σύστημα με p επεξεργαστές μπορεί να γραφεί σαν $f(n) + (1 - f(n)) = 1$, όπου $f(n)$ είναι ο χρόνος εκτέλεσης του σειριακού μέρους του κώδικα και $(1 - f(n))$ ο χρόνος εκτέλεσης του παράλληλου μέρους του κώδικα.
- Τότε ο συνολικός χρόνος εκτέλεσης του προγράμματος σε ένα επεξεργαστή είναι: $f(n) + P(1 - f(n))$. Ο λόγος επιτάχυνσης είναι:

$$S_p = f(n) + p(1 - f(n)) = p - f(n)(p - 1)$$

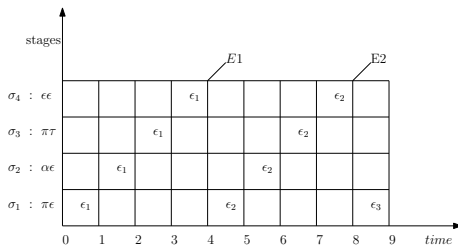
Νόμος του Gustafson (1988)

- Καθώς ο αριθμός των επεξεργαστών αυξάνει, αυξάνει και το παράλληλο τμήμα του κώδικα.
- Υποθέτοντας ότι το σειριακό μέρος $f(n)$ μειώνετε με το μέγεθος του προβλήματος, n , ο λόγος επιτάχυνσης αυξάνει.
- Το όριο για $n \rightarrow \infty$ είναι : $S_p^{max} = \lim_{n \rightarrow \infty} S_p = p$
- Πρακτικά ο νόμος του Gustafson θεωρεί ότι η επίδραση του σειριακού μέρους είναι η ίδια ακόμη και για πολύ μεγάλα παράλληλα συστήματα ενώ νόμος του Amdahl ότι επίδραση του σειριακού μέρους αυξάνει όσο μεγαλώνει ο αριθμός των επεξεργαστών!
- Τα περισσότερα ρεαλιστικά-πολύπλοκα προβλήματα ακολουθούν υβριδική συμπεριφορά μεταξύ του νόμου του Gustafson και του νόμου του Amdahl.

Η τεχνική της σωλήνωσης ή σωληνωμένος υπολογισμός pipelined computation

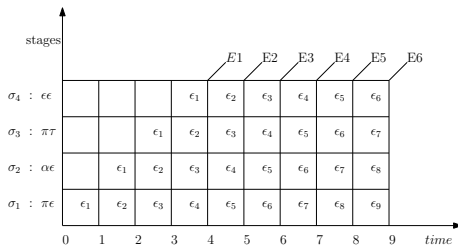
- Ένας σωληνωμένος υπολογισμός διαιρείται σε ένα αριθμό βημάτων τα οποία καλούνται στάδια (stages).
- Κάθε στάδιο αποτελεί ένα συγκεκριμένο τμήμα του υπολογισμού.
- Η έξοδος ενός σταδίου αποτελεί την είσοδο του επόμενου σταδίου.
- Αν θεωρήσουμε ότι κάθε στάδιο απαιτεί τον ίδιο χρόνο για να ολοκληρωθεί, τότε είναι εύκολο να παρατηρήσουμε ότι η βελτίωση του συνολικού χρόνου του υπολογισμού είναι ίση με τον πολλαπλασιαστικό παράγοντα που ισούται με τον αριθμό των σταδίων της σωλήνωσης.

Η τεχνική της σωλήνωσης ή σωληνωμένος υπολογισμός pipelined computation



Στο σχήμα απεικονίζονται τα 4 διαδοχικά στάδια $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ διέλευσης μιας εντολής e μέσα από έναν επεξεργαστή: Στο σ_1 πραγματοποιείται η προσκόμιση μιας εντολής e (πe), στο σ_2 η αποκωδικοποίηση της e (αe), στο σ_3 η προσκόμιση του τελεστέου ($\pi\tau$) και στο σ_4 η εκτέλεση της e (ϵe). Υποθέτοντας ότι τα χρονικά διαστήματα $[i, i + 1], i = 0, 1, 2, \dots$ είναι ίσα, στον ωρολογιακό παλμό 4 έχουμε την έξοδο E1 της πρώτης εντολής ϵ_1 από τον επεξεργαστή, στον ωρολογιακό παλμό 8 έχουμε την έξοδο E2 της δεύτερης εντολής ϵ_2 , κ.ο.κ.

Η τεχνική της σωλήνωσης ή σωληνωμένος υπολογισμός pipelined computation



Αν η διέλευση των εντολών ϵ_i , $i = 1, 2, \dots$ πραγματοποιείται από τον επεξεργαστή κατά το ανωτέρω σχήμα, ο επεξεργαστής χαρακτηρίζεται ως σωληνωμένος (pipelined), ενώ όλη η σχετική διεργασία ως σωλήνωση (pipelining). Ο χρόνος ολοκλήρωσης όλων των σταδίων μιας εντολής καλείται κύκλος εντολής (instruction cycle), ενώ ο χρόνος ολοκλήρωσης κάθε σταδίου της, κύκλος σωλήνα (pipeline cycle). Παρατηρούμε ότι στον ωρολογιακό παλμό 4 έχουμε την έξοδο $E1$ της πρώτης εντολής ϵ_1 από τον επεξεργαστή, στον παλμό 5 έχουμε την έξοδο $E2$ της δεύτερης εντολής ϵ_2 , στον παλμό 6 έχουμε την έξοδο $E3$ της τρίτης εντολής κ.ο.κ. Γενικεύοντας μετά τον παλμό 3 έχουμε σε κάθε παλμό την έξοδο μιας εντολής, δηλ. μετά τον πρώτο κύκλο μηχανής έχουμε την έξοδο τεσσάρων εντολών σε κάθε κύκλο μηχανής.

Η τεχνική της σωλήνωσης ή σωληνωμένος υπολογισμός pipelined computation

Η τεχνική αυτή έχει εφαρμογή είτε σε προβλήματα με επαναλαμβανόμενους υπολογισμούς ίδιας μορφής είτε σε απλούς υπολογισμούς που ωστόσο μπορούν να διαχωριστούν σε περισσότερα από ένα στάδια και μας ενδιαφέρει η αποδοτική επεξεργασία πολλαπλών στιγμιότυπων τους, ή σε προβλήματα που εμπεριέχουν έμφυτα τη λογική των διαδοχικών βημάτων υπολογισμού και επικοινωνίας.