Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 4: Το κόσκινο του Ερατοσθένη. Ο νόμος του Amdahl. Ο νόμος του Grosch. Ο νόμος των Gustafson-Barsis. Επιτάχυνση. Αποδοτικότητα. Κλιμάκωση.

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοπός της Ενότητας

- Η παράθεση βασικών νόμων των παράλληλων συστημάτων.
- Η αναλυτική εκτίμηση απόδοσης συστημάτων.

Έννοιες Σχετικές με τη διασωλήνωση

Έννοιες σχετικές με τη διασωλήνωση – Κύκλοι

- Ο χρόνος ολοκλήρωσης κάθε σταδίου μιας διασωληνωμένης αρχιτεκτονικής καλείται “**κύκλος σταδίου**”.
- Ο χρόνος ολοκλήρωσης όλων των σταδίων μιας εντολής καλείται “**κύκλος εντολής**”.
- Η διασωλήνωση υλοποιεί την τεχνική του “παραλληλισμού εντολών”, δηλαδή εκτελούνται σχεδόν ταυτόχρονα πολλές εντολές.

3 είναι τα είδη παραλληλισμών

- Παραλληλισμός Εντολών.
- Παραλληλισμός Δεδομένων.
- Παραλληλισμός Ελέγχου.

Παραλληλισμός δεδομένων

- Εκτός από τον παραλληλισμό εντολών, μπορούμε να έχουμε και παραλληλισμό δεδομένων.
- Παραλληλισμός δεδομένων (data parallelism) είναι η χρήση πολλών λειτουργικών μονάδων (π.χ. Πολλαπλών ALU) προκειμένου να εφαρμοσθεί η ίδια λειτουργία ταυτόγχρονα στα στοιχεία ενός συνόλου δεδομένων.
- Είναι διαφορετικός από τον παραλληλισμό εντολών.

Προβλήματα του παραλληλισμού δεδομένων

- Τρόπο επικοινωνίας και συγχρονισμού των μονάδων επεξεργασίας.
- Τον έλεγχο της ταυτόχρονης πρόσβασης στην κοινή μνήμη ή τις συσκευές εισόδου-εξόδου.
- Τον τρόπο ανάθεσης των προγραμμάτων στις μονάδες επεξεργασίας.
- Τον τρόπο περιγραφής παράλληλων προγραμμάτων.
- Τη διαδικασία ανάπτυξης και συντήρησης παράλληλου λογισμικού.

Υπάρχει και ο παραλληλισμός ελέγχου (control parallelism)

- Επιτυγχάνεται με την εφαρμογή διαφορετικών λειτουργιών σε διαφορετικά σύνολα δεδομένων.
- Η διασωλήνωση μπορεί να θεωρηθεί ως ειδική περίπτωση του παραλληλισμού ελέγχου.
- Ο συνδυασμός όλων των ειδών παραλληλισμού αποτελεί την καλύτερη λύση.

Το κόσκινο του Ερατοσθένη (εύρεση πρώτων αριθμών)

Ο αλγόριθμος "Το κόσκινο του Ερατοσθένη"

- Πρόκειται για τον κλασικό αλγόριθμο εύρεσης πρώτων αριθμών από τον Ερατοσθένη (275-195πΧ).
- Επιλύει το πρόβλημα της εύρεσης των πρώτων αριθμών, οι οποίοι είναι μικρότεροι ενός θετικού και ακέραιου αριθμού **Ν.**
- Το κόσκινο αρχίζει με μια γραμμική παράθεση των φυσικών αριθμών από **2, 3, …., Ν.**
- Εξαλείφει πολλαπλάσια του μεγαλύτερου πρώτου, ο οποίος είναι **μικρότερος ή ίσος της τετραγωνικής ρίζας** του **Ν.**

Παράδειγμα: “Το κόσκινο του Ερατοσθένη” (1/6)

- Έστω οι αριθμοί από 1 έως 100.
- Ξεκινάμε από το 2. Είναι πρώτος αριθμός.
- Διαγράφουμε όλα τα επόμενα πολλαπλάσια του 2.
- Το επόμενο που δεν έχει διαγραφεί είναι το 3.
- Ξεκινάμε από το 3. Είναι πρώτος αριθμός.
- Διαγράφουμε όλα τα επόμενα πολλαπλάσια του 3.
- Το επόμενο που δεν έχει διαγραφεί είναι το 5
- Ξεκινάμε από το 5. Είναι πρώτος αριθμός.
- Διαγράφουμε όλα τα επόμενα πολλαπλάσια του 5.
- …

Παράδειγμα:
“Το κόσκινο του Ερατοσθένη” (2/6)

1	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~
31	~~32~~	33	~~34~~	35	~~36~~	37	~~38~~	39	~~40~~
41	~~42~~	43	~~44~~	45	~~46~~	47	~~48~~	49	~~50~~
51	~~52~~	53	~~54~~	55	~~56~~	57	~~58~~	59	~~60~~
61	~~62~~	63	~~64~~	65	~~66~~	67	~~68~~	69	~~70~~
71	~~72~~	73	~~74~~	75	~~76~~	77	~~78~~	79	~~80~~
81	~~82~~	83	~~84~~	85	~~86~~	87	~~88~~	89	~~90~~
91	~~92~~	93	~~94~~	95	~~96~~	97	~~98~~	99	~~100~~

Διαγράφουμε τα πολλαπλάσια του 2.

Παράδειγμα: “Το κόσκινο του Ερατοσθένη” (3/6)

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Διαγράφουμε τα πολλαπλάσια του 3.

Παράδειγμα: “Το κόσκινο του Ερατοσθένη” (4/6)

Διαγράφουμε τα πολλαπλάσια του 5.

Παράδειγμα: “Το κόσκινο του Ερατοσθένη” (5/6)

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Διαγράφουμε τα πολλαπλάσια του 7.

Ο επόμενος αριθμός 11 έχει τετράγωνο μεγαλύτερο από 100 οπότε σταματάμε.

Παράδειγμα: "Το κόσκινο του Ερατοσθένη" (6/6)

- Υπάρχει δυσκολία στον υπολογισμό του τετραγώνου του προηγούμενου πρώτου, για μεγάλους αριθμούς.
- Αν θέλουμε να το υλοποιήσουμε απαιτείται:
  - Ένας μονοδιάστατος μπουλιανός πίνακας (δηλαδή με τιμές 0 ή 1) του οποίου τα στοιχεία αντιστοιχούν στους φυσικούς αριθμούς.
  - Ένας ακέραιος ο οποίος αντιστοιχεί στον πλέον προσφάτως ευρεθέντα πρώτο.
  - Ένας ακέραιος, ο οποίος χρησιμοποιείται ως δείκτης βρόχου, αυξανόμενος καθώς πολλαπλάσια του τρεχούμενου πρώτου εξαλείφονται.

2 διαφορετικές υλοποιήσεις παραλληλισμού

- Μπορεί να υλοποιηθεί με δύο τρόπους:
  - Παραλληλισμό ελέγχου.
  - Παραλληλισμό δεδομένων

Υλοποίηση του κόσκινου του Ερατοσθένη με παραλληλισμό ελέγχου

Παραλληλισμός ελέγχου στο κόσκινο του Ερατοσθένη

- Υπάρχει διαμοιραζόμενη μνήμη.
- Κάθε επεξεργαστής έχει ένα δείκτη βρόχου.
- Κάθε επεξεργαστής απαλείφει τα πολλαπλάσια ενός πρώτου (π.χ. Ο p1 τα πολλαπλάσια του 2, ο p2 τα πολλαπλάσια του 3 κ.ο.κ.).
- Μπορεί να εμφανιστούν τα εξής **προβλήματα**:
  - Δύο επεξεργαστές μπορεί να απαλείφουν τον ίδιο πρώτο.
  - Ένας επεξεργαστής μπορεί να κοσκινίζει πολλαπλάσια ενός σύνθετου αριθμού.
- Μπορούν με κατάλληλες μεθόδους να αποφευχθούν.

Επιτάχυνση σε 2 και 3 επεξεργαστές (1/2)

- Αν χρησιμοποιήσουμε 1 επεξεργαστή για 1000 αριθμούς τότε θα ολοκληρωθεί η εργασία σε 1411 μονάδες χρόνου.
- Αν χρησιμοποιήσουμε 2 επεξεργαστές για 1000 αριθμούς τότε ο πρώτος επεξεργαστής θα ολοκληρώσει την εργασία του σε 706 μονάδες χρόνου και ο δεύτερος σε 705. (επιτάχυνση ~ 2).
- Για τρεις επεξεργαστές: ο πρώτος 499, ο δεύτερος 420, ο τρίτος 410. Άρα επιτάχυνση ~2.83.
- Είναι άνω φράγμα γιατί δε μπορεί να μειωθεί ο χρόνος 499, αφού είναι ο χρόνος απαλοιφής των πολλαπλασίων του 2.

Επιτάχυνση σε 2 και 3 επεξεργαστές (2/2)

- Υπάρχει άνω φράγμα στο 499, γιατί όσους επεξεργαστές και να χρησιμοποιήσουμε δε βελτιώνεται ο χρόνος.

Υλοποίηση του κόσκινου του Ερατοσθένη με παραλληλισμό δεδομένων

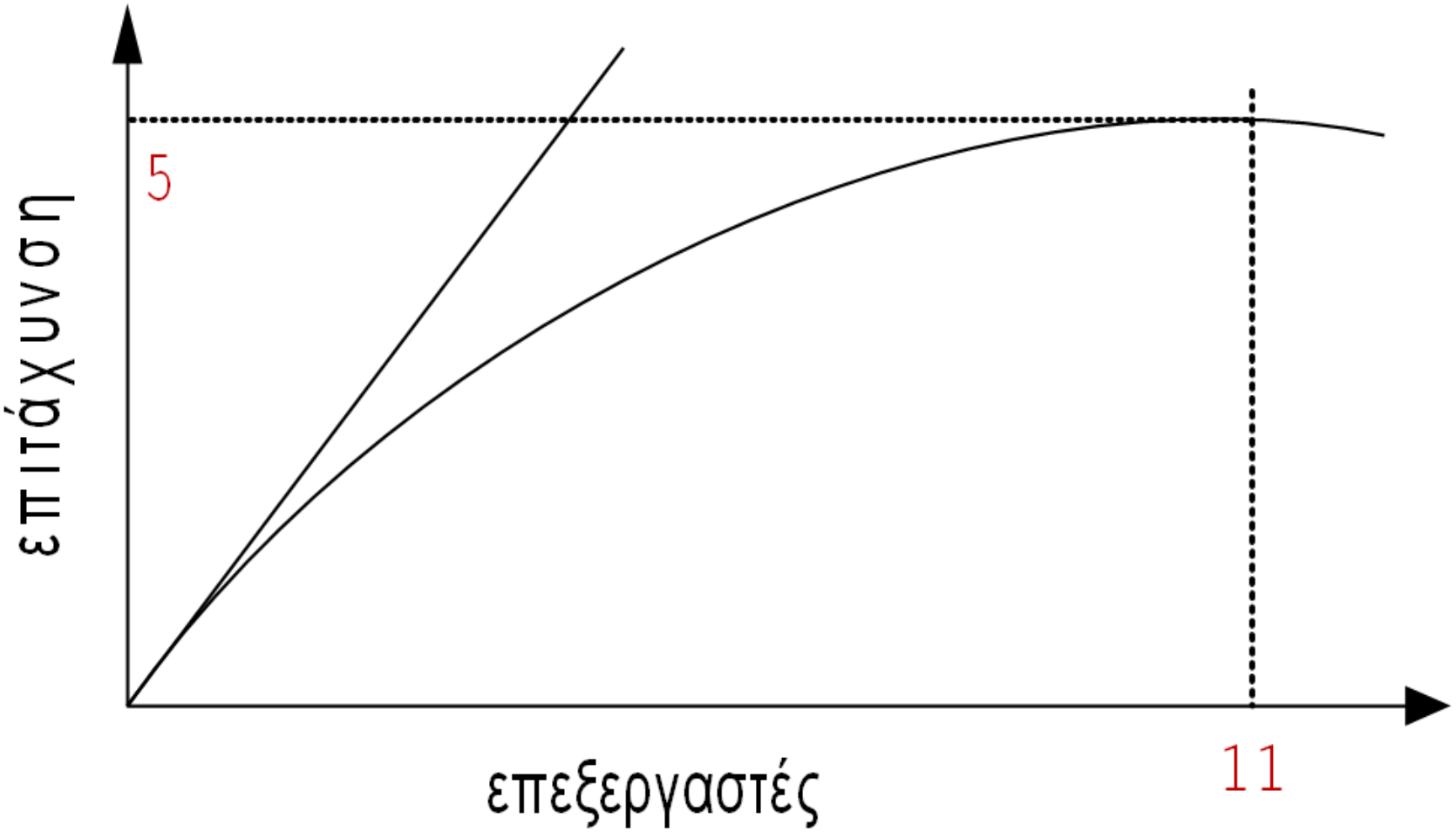
Παραλληλισμός δεδομένων στο κόσκινο του Ερατοσθένη

- Διαμοιράζουμε σε πλήθος p επεξεργαστών το διάστημα των αριθμών 0 έως N.
  - Ο 1ος επεξεργαστής παίρνει από 2 έως N/p.
  - Ο 2ος επεξεργαστής παίρνει από N/p+1 έως 2N/p.
  - ...Ο p επεξεργαστής παίρνει από (p-1)N/p+1 έως N.
  - δηλαδή ο n επεξεργαστής (n-1)N/(n+1) έως nN/p.
- Ο κάθε επεξεργαστής έχει τη δικιά του θέση μνήμης για τον πρόσφατο πρώτο και για το δείκτη βρόχου.
- Ο πρώτος επεξεργαστής θα εντοπίσει τον επόμενο πρώτο και θα μεταδώσει την τιμή του στους άλλους.

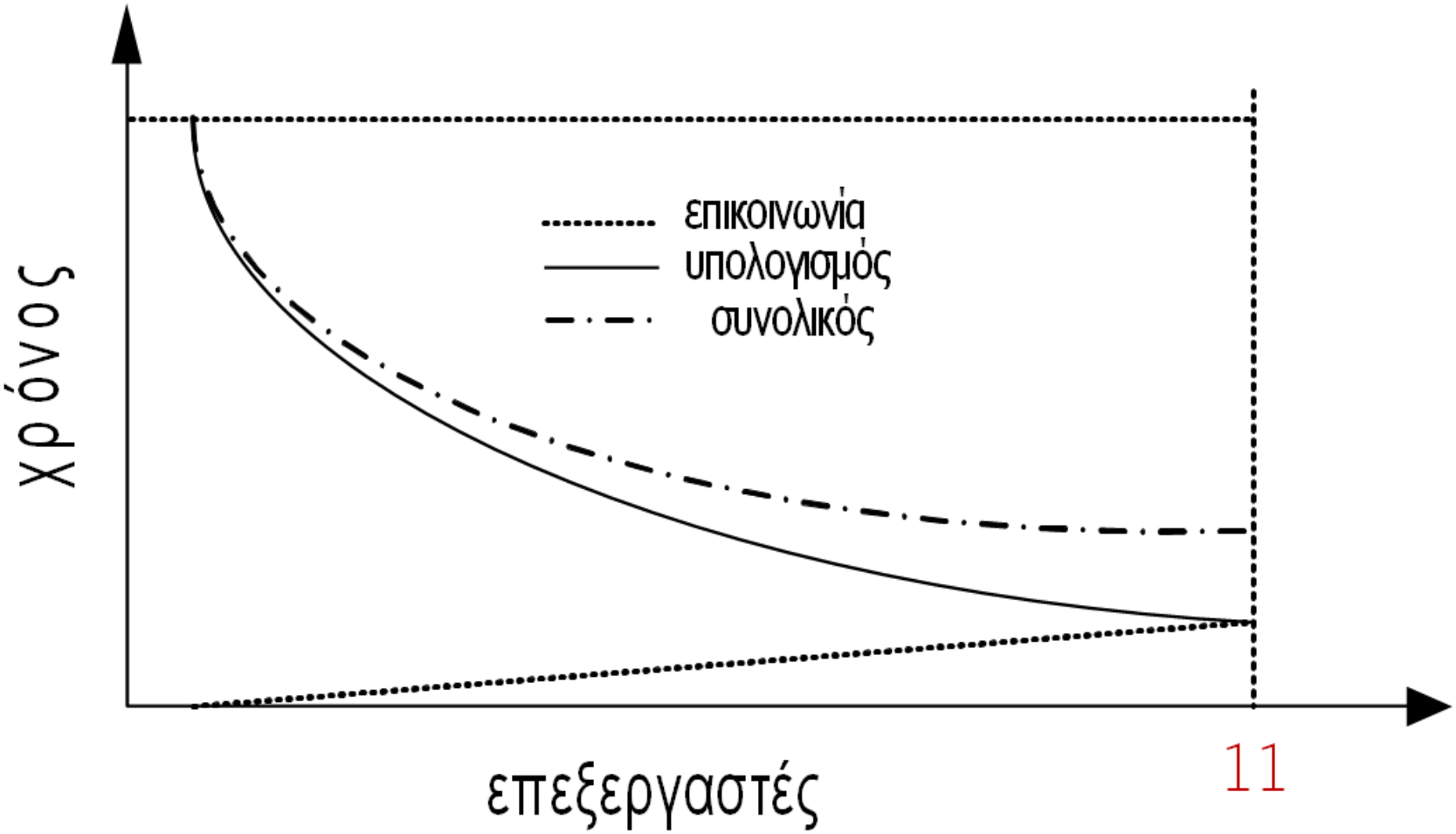
Επιτάχυνση παραλληλισμού δεδομένων στο Κόσκινο του Ερατοσθένη

- Για τον υπολογισμό της επιτάχυνσης απαιτείται να ληφθεί υπόψιν ο χρόνος μετάδοσης μηνυμάτων.
- Απαιτείται πέρασμα μηνύματος (πιθανό θέμα εξετάσεων εργαστηρίου).
- Αν κάνουμε το σχεδιάγραμμα με το χρόνο που απαιτείται για την εκτέλεση σε 1 2 κτλ επεξεργαστές θα δούμε ότι η επιτάχυνση είναι μέγιστη στους 11 επεξεργαστές (επόμενο σχήμα).

Η επιτάχυνση μεγιστοποιείται στους 11 επεξεργαστές



Ο χρόνος επικοινωνίας αυξάνεται γραμμικά



Μια πιο πραγματική υλοποίηση

- Δεν αποθηκεύονται τα αποτελέσματα.
- Απαιτείται συσκευή εισόδου/εξόδου.
- Απαιτείται δηλαδή να υπολογιστεί ο χρόνος που απαιτείται για είσοδο/έξοδο.
- Η επιτάχυνση είναι μικρότερη.

Η εύρεση των πρώτων αριθμών κρύβει σημαντικά βραβεία!!!

- http://w2.eff.org/awards/coop-prime-rules.php
- Official EFF Cooperative Computing Award Rules.
- EFF establishes four computation awards:
  - EFF will award $50,000 to the first individual or group who discovers aprime number with at least 1,000,000 decimal digits. **(δόθηκε 6/4/2000)**.
  - EFF will award $100,000 to the first individual or group who discovers a prime number with at least 10,000,000 decimal digits. **(δόθηκε 22/10/2009)**.
  - EFF will award $150,000 to the first individual or group who discovers aprime number with at least 100,000,000 decimal digits.
  - EFF will award $250,000 to the first individual or group who discovers a prime number with at least 1,000,000,000 decimal digits.

Μελέτη ορίων επιτάχυνσης σε παράλληλους αλγόριθμους

Μελέτη Ορίων Επιτάχυνσης Παράλληλων Εφαρμογών

- Υπολογιστικά Μοντέλα σε Παράλληλους Υπολογιστές.
- Παράγοντες Επιτάχυνσης και Αποδοτικότητας.
- Νόμος Amdahl.
- Νόμος του Grosch.
- Νόμος των Gustafson–Barsis.

2 είναι τα υπολογιστικά μοντέλα σε παράλληλους υπολογιστές

- Υπάρχουν δυο Υπολογιστικά Μοντέλα για Παράλληλους υπολογιστές με την υπόθεση ότι κάθε δοθέν υπολογισμός μπορεί να υποδιαιρεθεί σε παράλληλες υπό-εργασίες:
  - **Μοντέλο ίσης διάρκειας** (Equal Duration Model).
  - **Μοντέλο Παράλληλου Υπολογισμού με Σειριακά Τμήματα** (Parallel Computation with Serial Sections Model).

Μοντέλο ίσης διάρκειας (1/7)

- Υπόθεση στην οποία στηρίζεται το μοντέλο:
  - Κάθε δοθείσα εργασία μπορεί να διαιρεθεί σε **n** ίσες υπό-εργασίες, κάθε μια από την οποία μπορεί να ανατεθεί για εκτέλεση σε έναν επεξεργαστή.
- Αν θεωρήσουμε ότι:
  - $\mathbf{t_s}$ είναι χρόνος εκτέλεσης ολόκληρης της εργασίας (ενός προγράμματος) σε 1 επεξεργαστή.
  - $\mathbf{t_m}$ είναι ο χρόνος που απαιτείται ώστε ένας επεξεργαστής να εκτελέσει μια υπό-εργασία της ολικής εργασίας.

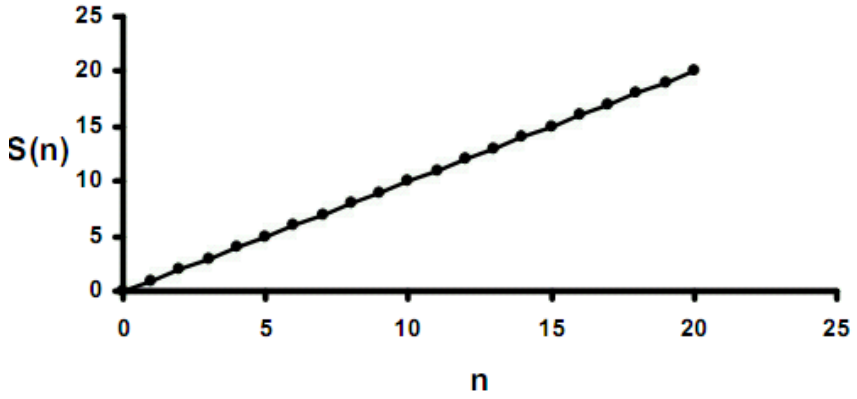
Μοντέλο ίσης διάρκειας (2/7)

- Επομένως ισχύει: $t_m = t_s / n$
- Διότι όλοι οι επεξεργαστές εκτελούν την υπό-εργασία τους ταυτόχρονα.
- Ορισμός:

  Παράγοντας επιτάχυνσης (speedup factor): S(n)
- Είναι ο λόγος του χρόνου που απαιτείται από έναν επεξεργαστή για την λύση ενός συγκεκριμένου προβλήματος προς το χρόνο που απαιτείται από ένα παράλληλο σύστημα που αποτελείται από **n** επεξεργαστές για να λύσει το ίδιο πρόβλημα.
- Επομένως: $S(n) = \mathbf{t_s} / \mathbf{t_m} = \mathbf{t_s} / (\mathbf{t_s} / \mathbf{n}) = \mathbf{n}$

Μοντέλο ίσης διάρκειας (3/7)

- S(n) = $\mathbf{t_s}$ / $\mathbf{t_m}$ = $\mathbf{t_s}$ / ($\mathbf{t_s}$ /**n**) = **n.**
- Η παραπάνω εξίσωση σύμφωνα με το μοντέλο ίσης διάρκειας, υποδηλώνει ότι ο συντελεστής επιτάχυνσης ισούται με το πλήθος των επεξεργαστών n που χρησιμοποιούνται.
- Επομένως περιμένουμε μια γραμμική αύξηση του S(n) συναρτήσει του **n**:



Μοντέλο ίσης διάρκειας (4/7)

- Στο μοντέλο αυτό δεν λήφθηκε υπόψιν ένας σημαντικός παράγοντας. Η **καθυστέρηση του δικτύου**, δηλαδή το χρόνο επικοινωνίας που απαιτείται για την επικοινωνία ή πιθανή ανταλλαγή δεδομένων μεταξύ των επεξεργαστών μέσω του δικτύου διασύνδεσης.
- Θεωρούμε ότι ο χρόνος επικοινωνίας είναι $\mathbf{t_c}$.
- Επομένως ο πραγματικός χρόνος για να επικοινωνήσουν 2 επεξεργαστές γράφεται:
  - $\mathbf{t_m} = (\mathbf{t_s}/\mathbf{n}) + \mathbf{t_c}$
- Επομένως, Speedup:
  - $S(n) = \mathbf{t_s} / \mathbf{t_m} = \mathbf{t_s} / [\ (\mathbf{t_s}/\mathbf{n}) + \mathbf{t_c}]$

Μοντέλο ίσης διάρκειας (5/7)

- Η S(n) = $\mathbf{t_s}$ / $\mathbf{t_m}$ = $\mathbf{t_s}$ / [($\mathbf{t_s}$ /**n**) + $\mathbf{t_c}$] μπορεί να γραφεί ως:

$$S(n) = \frac{n}{1 + n\frac{t_c}{t_s}}$$

- Αυτό φανερώνει ότι η επιτάχυνση ενός υπολογισμού εξαρτάται από:
  - Σειριακό χρόνο εκτέλεσης.
  - Καθυστέρηση λόγω επικοινωνίας.

Μοντέλο ίσης διάρκειας (6/7)

Μπορούν να διακριθούν 3 περιπτώσεις:

- **$t_c << t_s$:** το κόστος επικοινωνίας είναι αμελητέο.
  - Τότε η πιθανή επιτάχυνση είναι ίση με n.
- **$t_s << t_c$ => S(n) << 1 :** το κόστος επικοινωνίας είναι πολύ μεγάλο.
  - επομένως το κόστος επικοινωνίας είναι τόσο μεγάλο που έχουμε επιβράδυνση.
- **$t_c = t_s$:** το κόστος επικοινωνίας είναι συγκρίσιμο με το κόστος υπολογισμού.
  - Επομένως η επιτάχυνση είναι ~1 για n>>1.

Μοντέλο ίσης διάρκειας (7/7)

- Προκειμένου να θέσουμε την επιτάχυνση σε μια κλίμακα μεταξύ 0 και 1, εισάγουμε τον ορισμό της αποδοτικότητας (efficiency) ως τον λόγο της επιτάχυνσης προς το πλήθος των επεξεργαστών:

$$\xi = S(n) / n$$

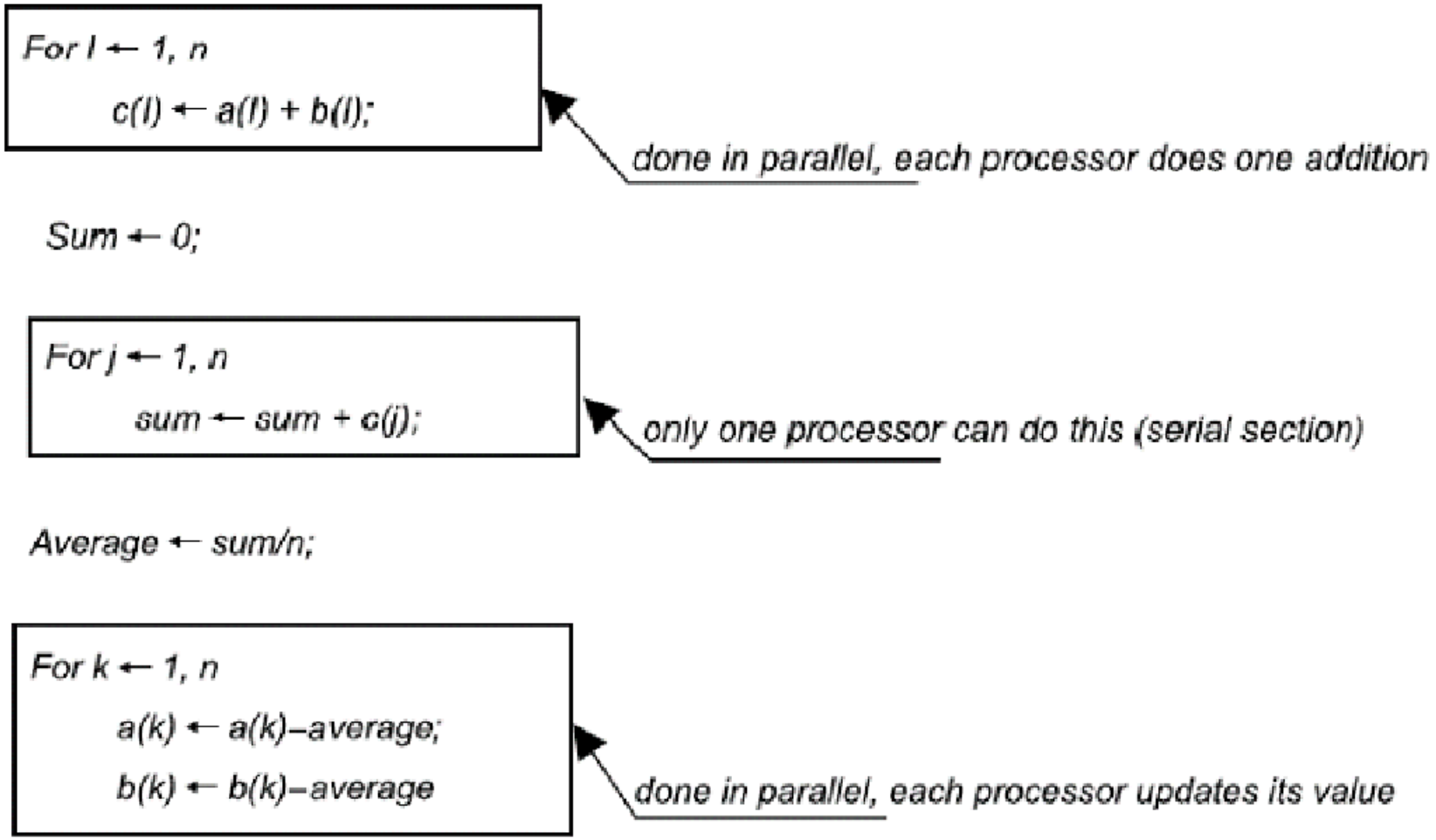
- Είναι ένα μέτρο της επιτάχυνσης που επιτυγχάνεται ανά επεξεργαστή.
- Σύμφωνα με το Μοντέλο ίσης διάρκειας, η αποδοτικότητα είναι ίση με 1 αν αγνοηθεί το κόστος επικοινωνίας. Διαφορετικά αν ληφθεί υπόψιν το κόστος επικοινωνίας, δίνεται από την σχέση:

$$\xi = \frac{1}{1 + n\frac{t_c}{t_s}}$$

Μοντέλο ίσης διάρκειας: Χαρακτηριστικά (1/2)

- Αν και απλό **δεν είναι ρεαλιστικό.**
- Αδυναμία: Βασίζεται στην υπόθεση ότι μια δοσμένη εργασία μπορεί να διαιρεθεί σε ένα σύνολο ίσων υπό-εργασιών οι οποίες μπορούν να εκτελεσθούν από ένα πλήθος επεξεργαστών παράλληλα.
- Οι πραγματικοί αλγόριθμοι περιέχουν (σειριακά) τμήματα τα οποία δεν μπορούν να διαιρεθούν σε άλλες υπό-εργασίες. Ο μόνος τρόπος είναι να εκτελεστούν μόνο σε έναν επεξεργαστή!
- Παράδειγμα:
  1. Άθροιση των αντίστοιχων στοιχείων 2 διανυσμάτων.
  2. Άθροισμα του προηγούμενου βήματος και υπολογισμός ΜΟ.
  3. Επικαιροποίηση των αρχικών τιμών των διανυσμάτων.

Παραδείγματα



Μοντέλο ίσης διάρκειας: Χαρακτηριστικά (2/2)

- Το παραπάνω παράδειγμα φανερώνει ότι ένα ρεαλιστικό υπολογιστικό μοντέλο πρέπει να λαμβάνει υπόψιν την ύπαρξη (σειριακών) τμημάτων σε ένα πρόγραμμα που δε μπορούν να διαιρεθούν σε υπο-εργασίες.
- Αυτό λαμβάνεται υπόψιν στο μοντέλο **Παράλληλου Υπολογισμού με Σειριακά Τμήματα.**

Ο νόμος του Amdahl

Μοντέλο Παράλληλου Υπολογισμού με Σειριακά Τμήματα (1/7)

- Υπόθεση στην οποία στηρίζεται το μοντέλο:
  - Κάθε δοθείσα εργασία έχει ένα τμήμα (ένα κλάσμα της ολικής εργασίας) το οποίο ΔΕ μπορεί να διαιρεθεί σε **n** ίσες υπο-εργασίες.
- Έστω **f** αυτό το τμήμα:
  - Επομένως το κλάσμα του προγράμματος που απομένει μπορεί να διαιρεθεί σε **n** ίσες υπο-εργασίες είναι **(1-f).**

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (2/7)

- Κι εδώ: t_s είναι χρόνος εκτέλεσης ολόκληρης της εργασίας (ολόκληρου του προγράμματος) σε 1 επεξεργαστή.
- Και: t_m θεωρούμε ότι είναι ο χρόνος που απαιτείται ώστε ένας επεξεργαστής να εκτελέσει μια υπο-εργασία της ολικής εργασίας.

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (3/7)

- Κάνοντας αντίστοιχους συλλογισμούς με αυτούς που εφαρμόσαμε στο Μοντέλο Ίσης Διάρκειας, παράγεται ο χρόνος t_m συναρτήσει του χρόνου εκτέλεσης t_s:
- Επομένως, η επιτάχυνση είναι $S(n)= t_s / t_m$.
- Επιτάχυνση:

$$t_m = f * t_s + (1-f)\frac{t_s}{n}$$

$$S(n) = \frac{t_s}{t_m} = \frac{t_s}{f * t_s + (1-f)\frac{t_s}{n}} = \frac{n}{1+(n-1)*f}$$

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (4/7)

- Σύμφωνα με την σχέση η πιθανή επιτάχυνση εξαρτάται από το πλήθος των επεξεργαστών καθώς και από το τμήμα του κώδικα που δεν μπορεί να παραλληλοποιηθεί.
- Το τελευταίο είναι γνωστό ως και **Νόμος του Amdahl**: **“Υπάρχει ένα όριο στον βαθμό που ένα πρόγραμμα μπορεί να παραλληλοποιηθεί**”.
- Στην περίπτωση που το πρόγραμμα είναι πλήρως σειριακό (f = 1), τότε δεν μπορεί να γίνει επιτάχυνση όσοι επεξεργαστές και να χρησιμοποιηθούν.

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (5/7)

- Σύμφωνα με τον Νόμο του Amdahl η βελτίωση στην απόδοση ενός παράλληλου αλγόριθμου δεν περιορίζεται από τον αριθμό των επεξεργαστών, αλλά κυρίως από το τμήμα του αλγορίθμου που δεν μπορεί να παραλληλοποιηθεί.
- Υπάρχει ένα εσωτερικό όριο παραλληλοποίησης και αυτό είχε αντίκτυπο στην ερευνητική κοινότητα για τις παράλληλες μηχανές.
- ΚΑΙ σε αυτό το μοντέλο δεν λήφθηκε υπόψιν η καθυστέρηση του δικτύου.

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (6/7)

- Αν λάβουμε υπόψιν την καθυστέρηση λόγω επικοινωνίας:

$$t_m = f * t_s + (1-f)\frac{t_s}{n} + t_c$$

- Η επιτάχυνση λοιπόν γράφεται ως:

$$S(n) = \frac{t_s}{f * t_s + (1-f)\frac{t_s}{n} + t_c} = \frac{n}{f*(n-1)+1+n*(t_c/t_s)}$$

- Η μέγιστη επιτάχυνση δίνεται από:

$$\lim_{n\to\infty} S(n) = \frac{n}{f*(n-1)+1+n*(t_c/t_s)} = \frac{1}{f+(t_c/t_s)}$$

Μοντέλου Παράλληλου Υπολογισμού με Σειριακά Τμήματα (7/7)

- Η τελευταία σχέση φανερώνει ότι και σε αυτό το μοντέλο η μέγιστη επιτάχυνση ΔΕΝ καθορίζεται από το πλήθος των επεξεργαστών ΑΛΛΑ από το τμήμα του υπολογισμού που δεν παραλληλοποιείται και από τον χρόνο επικοινωνίας.

Αποδοτικότητα

- Παρατήρηση: Πρέπει να γίνεται ορθή χρήση των πόρων (επεξεργαστών) μιας παράλληλης μηχανής ώστε να είναι αποδοτική.
- Αποδοτικότητα (χωρίς χρόνος επικοινωνίας).
- Αποδοτικότητα (με χρόνο επικοινωνίας).

$$\xi = \frac{1}{1+(n-1)*f}$$

$$\xi = \frac{1}{f*(n-1)+1+n*(t_c/t_s)}$$

Ο νόμος του Grosch

Ο νόμος του Grosch (1/6)

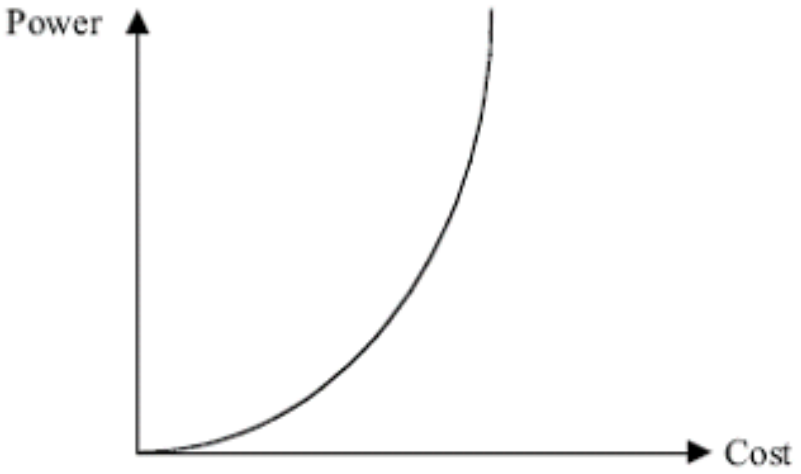
- Διατυπώθηκε το 1940 από τον H. Grosch και σχετίζει την απόδοση ενός παράλληλου συστήματος σε σχέση με την τιμή του.
- Υπέθεσε αξιωματικά ότι η ισχύς P ενός υπολογιστή αυξάνεται εκθετικά με το κόστος C κατασκευής του, και έδωσε τον τύπο:

$$P = K * C^{s}$$

- Όπου K, S θετικές σταθερές, με πολύ πιθανή για το S την τιμή 2.

Ο νόμος του Grosch (2/6)

- Για S=2, παίρνουμε την γραφική του παρακάτω σχήματος.
- Με άλλα λόγια: Για να αυξήσουμε την ταχύτητα υπολογισμού 2 φορές θα πρέπει να πληρώσουμε 4 φορές πιο πολλά χρήματα.
- Η εξέλιξη των υπολογιστών διέψευσε τον Grosch, αλλά εκείνη την εποχή ήταν έντονη η διαφωνία για το αν άξιζε ή όχι να κατασκευαστούν πιο γρήγορες μηχανές.



Ο νόμος του Amdahl (3/6)

- "Υπάρχει ένα όριο στο βαθμό που ένα πρόγραμμα μπορεί να παραλληλοποιηθεί".
- Η επιτάχυνση σε ένα παράλληλο σύστημα δίνεται από:

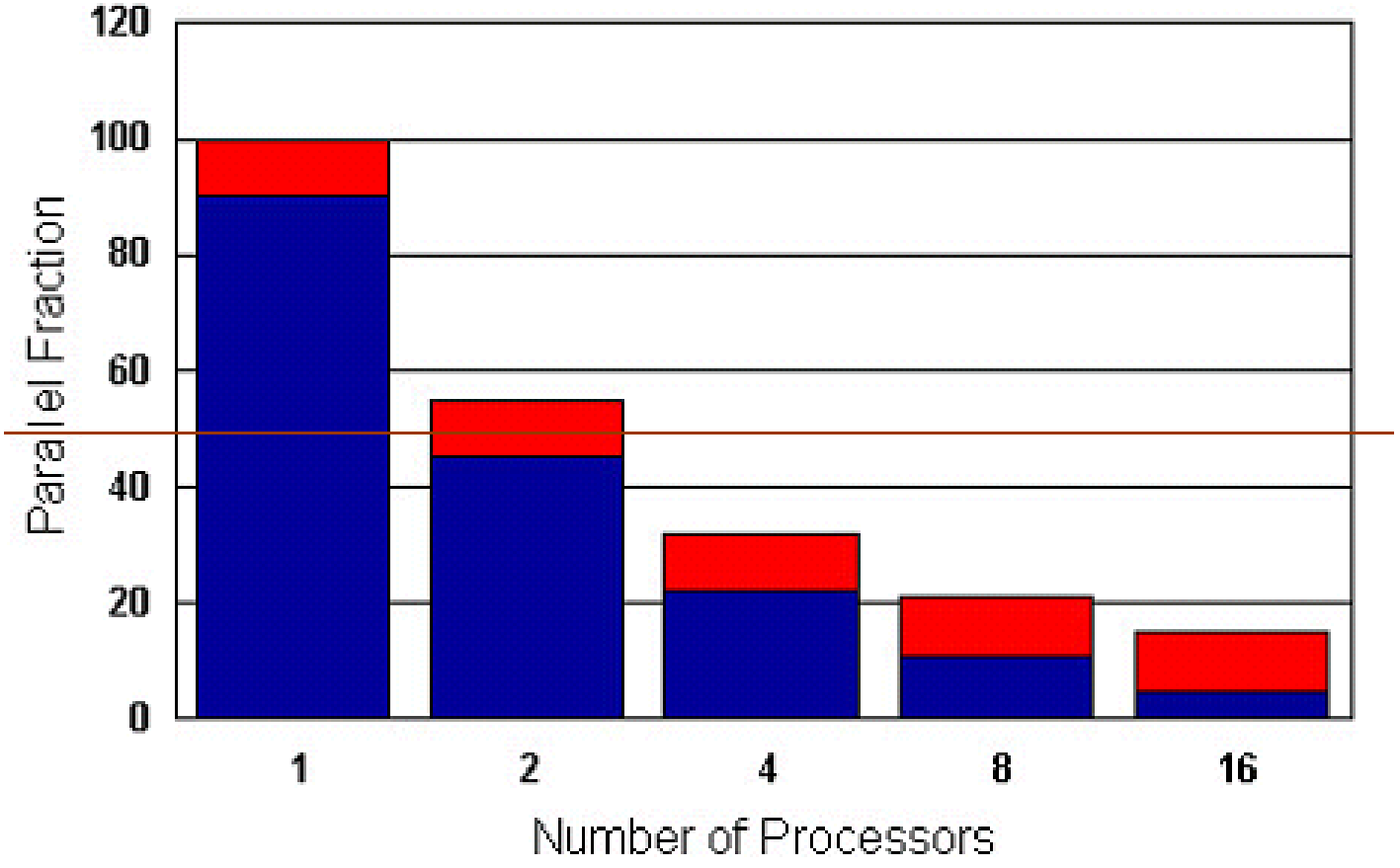
$$S(n) = \frac{t_s}{t_m} = \frac{t_s}{f * t_s + (1-f)\frac{t_s}{n}} = \frac{n}{1+(n-1)*f}$$

$$\lim_{n\to\infty} S(n) = \frac{1}{f}$$

Ο νόμος του Amdahl (4/6)

- Διατυπώθηκε το 1967 από τον G. Amdahl και κινείται στο ίδιο πεσιμιστικό πνεύμα με το Νόμο του Grosch, ότι δηλαδή λόγο των εγγενών σειριακών τμημάτων που υπάρχουν σε έναν αλγόριθμο δεν είναι βιώσιμη (επικερδής) η χρήση παράλληλων μηχανών.
- Ο Amdahl είχε καταλήξει στο συμπέρασμα αυτό με δεδομένο ότι το **f** είναι σταθερό και δεν μεταβάλλεται συναρτήσει του n.

Ο νόμος του Amdahl (5/6)



Ο νόμος του Amdahl (6/6)

- Στην πράξη έχουν παρατηρηθεί περιπτώσεις με παράλληλους αλγόριθμους όπου το f, είναι συνάρτηση του πλήθους των επεξεργαστών n, δηλαδή η επιτάχυνση γράφεται:

$$S(n) = \frac{n}{1 + (n-1) * f(n)}$$

$$\lim_{n \to \infty} f(n) = 0$$

$$\lim_{n \to \infty} S(n) = n$$

Ο νόμος του Amdahl σε κάποιες περιπτώσεις δεν ισχύει

- Δηλαδή επιτυγχάνεται η πολυπόθητη γραμμική αύξηση, αυξανομένου του n, δηλαδή “παραβαίνει” το Νόμο του Amdahl.
- Ουσιαστικά δεν ανατρέπει αλλά συμπληρώνει το Νόμο του Amdahl.
- Τέτοια γραμμική αύξηση επιτεύχθηκε από ερευνητές του Sandia National Laboratories χρησιμοποιώντας ένα σύστημά 1024 επεξεργαστών.

Ο Νόμος των Gustafson - Barsis

Ο Νόμος των Gustafson - Barsis (1/2)

- Το 1988 οι Gustafson & Barsis παρατήρησαν ότι σε συστήματα με εκατοντάδες επεξεργαστές, αρκετά μεγάλου μεγέθους προβλήματα μπορούσαν να παραλληλοποιηθούν αποτελεσματικά.
- Ίσχυαν λοιπόν οι σχέσεις:

$$S(n) = \frac{n}{1 + (n-1) * f(n)}$$

$$\lim_{n \to \infty} S(n) = n$$

Ο Νόμος των Gustafson - Barsis (2/2)

- Κατέληξαν στην παρακάτω σχέση για την επιτάχυνση:

$$SS(n) = n - (n - 1)*f$$

- Πρέπει να πληρούνται και κάποιες προδιαγραφές, όπως π.χ. η διαθέσιμη μνήμη κάθε επεξεργαστή.
- Ο αντίκτυπος του συμπεράσματος των Gustafson – Barsis ήταν τεράστιος και ώθησε τους ερευνητές στην κατεύθυνση των Παράλληλων υπολογιστών.
- Εργάζονταν και οι δύο στο Sandia National Laboratories.

Ο Στόχος της παράλληλης επεξεργασίας είναι η απόδοση

- Ορισμός:
  - Ένας παράλληλος υπολογιστής αποτελείται από ένα πλήθος επεξεργαστικών στοιχείων που επικοινωνούν και συνεργάζονται για να επιλύσουν μεγάλα προβλήματα γρήγορα [Almasi and Gottlieb 1989].
- Ζητήματα:
  - Πόσο μεγάλο είναι το πλήθος των στοιχείων;
  - Πώς πραγματοποιείται η επικοινωνία και η συνεργασία τους;
  - **Πως μεταφράζεται αυτό σε απόδοση και ποια είναι αυτή**;

Τι είναι τα μέτρα απόδοσης

- Βασικό και απόλυτο μέτρο απόδοσης:
  - **Ο χρόνος εκτέλεσης.**
- Γενικά, αναφερόμαστε στον χρόνο που χρειάζεται ένα υπολογιστικό σύστημα για να ολοκληρώσει έναν υπολογισμό ή στον χρόνο που απαιτεί ένας αλγόριθμος ή μία διαδικασία για να ολοκληρωθεί.

Τι είναι ο χρόνος εκτέλεσης;

- Ακολουθιακός χρόνος εκτέλεσης:
  - εκτέλεση σε ακολουθιακό σύστημα (σε ένα επεξεργαστή).
- Βέλτιστος ακολουθιακός χρόνος εκτέλεσης:
  - ο ακολουθιακός χρόνος του ταχύτερου γνωστού αλγορίθμου (προγράμματος).
- Παράλληλος χρόνος εκτέλεσης:
  - εκτέλεση σε παράλληλο / πολυεπεξεργαστικό σύστημα.

Πως γίνεται η σύγκριση;

- Βασικό μέτρο σύγκρισης της απόδοσης: Η χρονοβελτίωση (**speedup**).
- Ορίζεται ως:
  - **speedup = sequential time / parallel time**
- **Απόλυτη χρονοβελτίωση**: Σύγκριση με το βέλτιστο ακολουθιακό χρόνο εκτέλεσης στο συγκεκριμένο σύστημα.
- **Χρονοβελτίωση** (σχετική): Σύγκριση με την εκτέλεση του παράλληλου αλγορίθμου ακολουθιακά (μια διεργασία/ένα νήμα).

Βασικές Μετρικές Λειτουργιών (1/2)

- **Χρόνος απόκρισης** (latency): Ο χρόνος που μεσολαβεί για την ολοκλήρωση μιας λειτουργίας.
- **Ρυθμός υπολογισμού ή ρυθμαπόδοση** (bandwidth): Ο ρυθμός με τον οποίο εκτελούνται οι λειτουργίες.
- **Κόστος** (cost): Το αντίκτυπο της εκτέλεσης των λειτουργικών στο χρόνο εκτέλεσης του προγράμματος.

Βασικές Μετρικές Λειτουργιών (2/2)

- Σε ένα ιδεατό κόσμο:
  - Ο ρυθμός υπολογισμού θα ήταν αντίστροφος του χρόνου απόκρισης.
  - Το κόστος θα ήταν το γινόμενο του χρόνου απόκρισης επί του αριθμού των λειτουργιών.

Παράδειγμα (1/2)

- Έστω παράλληλο σύστημα όπου:
  - Χρόνος απόκρισης μιας εντολής: 100 nsec.
    - Άρα ρυθμός 10 εκατομμύρια εντολές το δευτερόλεπτο
  - Αν όμως υπάρχει οργάνωση σε αγωγό 10 σταδίων.
    - τότε ο ρυθμός είναι 100 εκατομμύρια εντολές ανά δευτ.
  - Αν η εφαρμογή εκκινεί μια εντολή κατά μέσο όρο κάθε 200 nsec.
    - τότε ο ρυθμός είναι 5 εκατομμύρια εντολές ανά δευτ.

Παράδειγμα (2/2)

- Αν μια εφαρμογή χρησιμοποιήσει το συγκεκριμένο παράλληλο σύστημα για να εκτελέσει 100 εκατ. εντολές που κυμαίνεται το κόστος της εκτέλεσης τους;
  - Μέγιστο: 10 seconds.
  - Ελάχιστο: 1 second.
- Αν υπάρχουν εξαρτήσεις;

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (1/8)

Ακολουθιακός χρόνος εκτέλεσης:

- Αν υποθέσουμε ότι ένας ακολουθιακός αλγόριθμος απαιτεί **n** αριθμητικές πράξεις για να εκτελεστεί, και ότι η κάθε πράξη χρειάζεται μία χρονική μονάδα τ_α για να ολοκληρωθεί. Τότε ο ακολουθιακός χρόνος εκτέλεσης του αλγορίθμου θα είναι:

$$T_{seq}(n) = n^* \tau_\alpha$$

τ_α = μέσος χρόνος εκτέλεσης μίας πράξης

- Η τ_α είναι ανεξάρτητη του αλγορίθμου και εξαρτάται από τα χαρακτηριστικά του συστήματος. Ο αριθμός των πράξεων **n** αναφέρεται ως μέγεθος του προβλήματος ή και ως υπολογιστικός φόρτος.

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (2/8)

Ρυθμός παραγωγής αποτελεσμάτων, είναι **το πλήθος των παραγόμενων αποτελεσμάτων** από τις εκτελούμενες πράξεις **δια του χρόνου** που απαιτήθηκε για τους υπολογισμούς αυτούς. Πρόκειται δηλαδή για μέσο ρυθμό υπολογισμού. Στην περίπτωση του ακολουθιακού αλγορίθμου έχουμε:

$$r_{seq}(n) = n/T_{seq}(n) = n/n*\tau_a = 1/\tau_a$$

- Και σ' αυτή την περίπτωση το μέγεθος $\mathbf{r_{seq}}$ είναι ιδιότητα του συστήματος, ανεξάρτητη του αλγορίθμου και του μεγέθους του προβλήματος **n** ή του υπολογιστικού φορτίου.

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (3/8)

Ακολουθιακή εκτέλεση παράλληλου αλγορίθμου:

- Η εκτέλεση ενός παράλληλου αλγορίθμου σε παράλληλο σύστημα, **απαιτεί διάφορες χρονικές επιβαρύνσεις** που οφείλονται στην παρουσία πολλαπλών επεξεργαστικών στοιχείων PE>1 (Processing Elements).
- Τέτοιες επιβαρύνσεις μπορεί να οφείλονται για παράδειγμα, στην **ανάγκη ανταλλαγής δεδομένων** μεταξύ των επεξεργαστών προκειμένου να ολοκληρωθεί ένα σύνολο πράξεων. Εάν ο παράλληλος αλγόριθμος εκτελεστεί ακολουθιακά, τα επιπλέον στοιχεία που προκαλούν τις επιβαρύνσεις μπορούν να παραλειφθούν και να χρησιμοποιηθούν από τον αλγόριθμο μόνο οι απαραίτητες (ωφέλιμες) πράξεις. Θεωρούμε ότι ο αριθμός των ωφέλιμων πράξεων είναι $\mathbf{n}_{//}$. Έτσι η ακολουθιακή εκτέλεση ενός παράλληλου αλγορίθμου απαιτεί χρόνο:

$$T_1(n) = n_{//} * T_a$$

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (4/8)

Πλεονασμός,

είναι το πηλίκο που ισούται με το λόγο των δύο ακολουθιακών χρόνων $\mathbf{T_1}$ και $\mathbf{T_{seq}}$. Ισχύει επίσης ότι $\mathbf{n_{//} >= n}$, μπορεί δηλαδή ο αντίστοιχος παράλληλος αλγόριθμος να απαιτεί επιπλέον πράξεις από αυτές που απαιτούνται στον ακολουθιακό, για να πράξει το αντίστοιχο αποτέλεσμα.

$$\text{Πλεονασμός}: =\rho = n_{//}/n = T_1(n)/T_{seq}(n) = n_{//} * \tau_\alpha / n * \tau_\alpha \geq 1$$

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (5/8)

Παράλληλος χρόνος εκτέλεσης:

Για την εκτέλεση του παράλληλου αλγορίθμου σε ένα παράλληλο σύστημα με **P** επεξεργαστές, θα υπολογίσουμε εκτός από το χρόνο εκτέλεσης των "ωφέλιμων" πράξεων και το χρόνο που θα χρειαστεί για τις επιπλέον, απαραίτητες διαδικασίες συντονισμού του αλγορίθμου αλλά και τις διαδικασίες συντονισμού του παράλληλου συστήματος (Χρόνος Επιβάρυνσης). Έτσι ο παράλληλος χρόνος θα είναι το άθροισμα των δύο επιμέρους χρόνων.

T_{par} = (Ωφέλιμες πράξεις)*(μέσο χρόνο πράξεων) + Επιβαρύνσεις*(μέσο χρόνο επιβαρύνσεων)

$$T(n,P) = \Omega(n,P) * \tau_\alpha + E(n,P) * \tau_\varepsilon$$

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (6/8)

Παράλληλος χρόνος εκτέλεσης (συνέχεια…)

- Στην πραγματικότητα αν θελήσουμε να συγκρίνουμε τις δύο στοιχειώδεις μονάδες χρόνου τ_α και τ_ε, ισχύει ότι $\tau_\varepsilon > \tau_\alpha$. Αν και οι χρόνοι αυτοί δεν είναι απολύτως συγκρίσιμοι μπορούμε να πούμε, για παράδειγμα, ότι ο χρόνος που απαιτείται για τη μεταφορά ενός byte από ένα επεξεργαστή σε έναν άλλο είναι μεγαλύτερος από μία αριθμητική πράξη που μπορεί να εκτελεστεί μεταξύ δύο μεταβλητών του ενός Byte στο ίδιο υπολογιστικό σύστημα.
- Όταν εκτελούμε με P=1 τότε ισχύει ότι Ω(n,1):=n και E(n,1):=0 δηλαδή,

$T(n,1) = T_1$ =χρόνος ακολουθιακής εκτέλεσης παράλληλου αλγορίθμου

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (7/8)

Χρονοβελτίωση ή Επιτάχυνση (Speedup),

- Η χρονοβελτίωση δείχνει πόσες φορές πιο γρήγορος είναι ο παράλληλος αλγόριθμος από τον αντίστοιχο ακολουθιακό.

είναι το πηλίκο

$$S(n,P) = T_{seq}(n) / T(n,P),$$

ισχύει ότι

$$1 \leq S \leq P$$

Ορισμός Επιτάχυνσης (speedup) (1/2)

- Κλάσμα με:
  - Αριθμητή: χρόνο εκτέλεσης υπολογισμού ακολουθιακό.
  - Παρονομαστή: απαιτούμενο χρόνο εκτέλεσης σε παράλληλη μηχανή (δεδομένων ή εντολών).
- Όσο μεγαλώνει η επιτάχυνση τόσο μικραίνει ο χρόνος εκτέλεσης σε σύστημα παράλληλης επεξεργασίας.

$$\frac{\chi\rho\acute{o}\nu\text{ος ακολουθιακής εκτέλεσης}}{\chi\rho\acute{o}\nu\text{ος παράλληλης εκτέλεσης}}$$

Ορισμός Επιτάχυνσης (speedup) (2/2)

- Επιτάχυνση:

$$S_p = \frac{T_1}{T_p}$$

  - p αριθμός επεξεργαστών.
  - T_1 χρόνος ακολουθιακής εκτέλεσης.
  - T_p χρόνος παράλληλης εκτέλεσης.
- Αν S_p = p τότε έχουμε γραμμική επιτάχυνση ή ιδεατή επιτάχυνση (π.χ. διπλασιάζουμε τον αριθμό των επεξεργαστών και διπλασιάζεται η ταχύτητα εκτέλεσης).

Υπεργραμμική επιτάχυνση

- Υπάρχει (σπάνια) περίπτωση να έχουμε επιτάχυνση μεγαλύτερη από p.
- Αυτό μπορεί να οφείλεται στην ιεραρχία μνήμης και ιδιαίτερα στην λειτουργία της κρυφής μνήμης.
  - Περισσότεροι επεξεργαστές, περισσότερη κρυφή μνήμη.
  - Προκαλείται σημαντική μείωση της καθυστέρησης πρόσβασης στη μνήμη.
- Παρατηρείται επίσης όταν χρησιμοποιείται ένας αλγόριθμος οπισθοδρόμησης (backtrack) για την εύρεση λύσεων σε ένα πρόβλημα.

Κλιμάκωση (1/2)

- Η κλιμάκωση εκφράζει τη σταδιακή επέκταση.
- Ένας **αλγόριθμός** είναι κλιμακώσιμος (scalable) αν το επίπεδο παραλληλισμού του αυξάνεται τουλάχιστον γραμμικά με το μέγεθος του προβλήματος.
- Μια **αρχιτεκτονική** είναι κλιμακώσιμη αν εξακολουθεί να συνεπάγεται την ίδια απόδοση ανά επεξεργαστή, καθώς αυξάνεται ο αριθμός των επεξεργαστών, μολονότι χρησιμοποιήθηκε σε μεγαλυτέρου μεγέθους προβλήματα.

Κλιμάκωση (2/2)

- Η αλγοριθμική και η αρχιτεκτονική κλιμάκωση είναι σημαντικές έννοιες.
- Οι αλγόριθμοι που βασίζονται στον παραλληλισμό δεδομένων είναι πιο κλιμακώσιμοι από τους άλλους.

Μέτρα Απόδοσης
Χρόνοι Εκτέλεσης (8/8)

- Αποδοτικότητα (Efficiency), είναι ένα μέτρο της χρονοβελτίωσης, σε σχέση με το πλήθος των επεξεργαστών που απαίτησε για να επιτευχθεί:

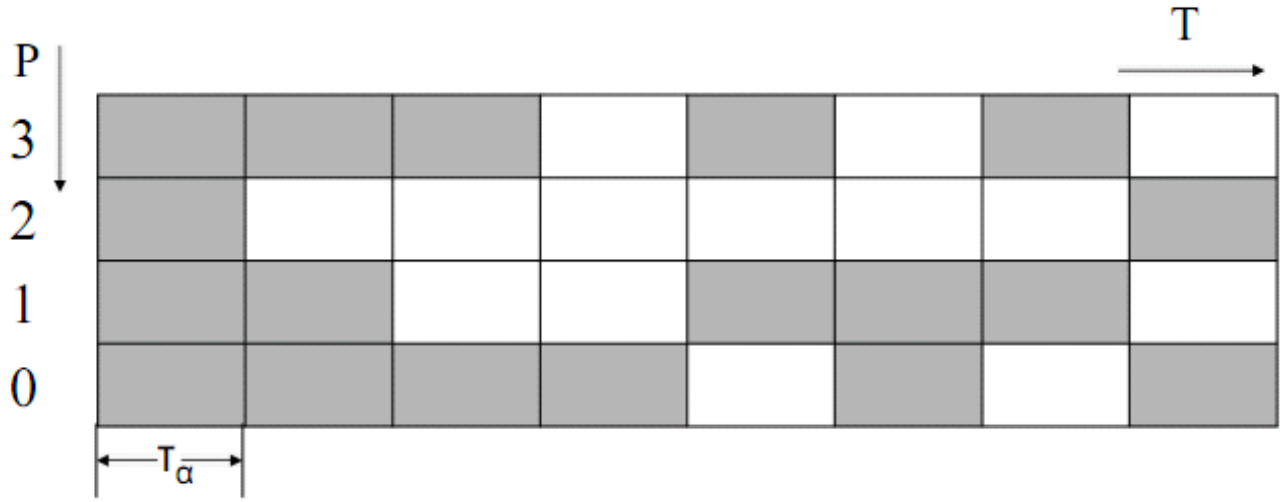
$$E = S(n,P)/P = T_{seq}(n)/PT(n,P) \leq 1$$

- Η ανισότητα ισχύει διότι η χρονοβελτίωση δεν μπορεί να υπερβαίνει την τιμή P (πλήθος επεξεργαστών). {**εκτός αν υπάρχει υπερ-γραμμική επιτάχυνση**}.

Μέτρα Απόδοσης - Αξιοποίηση (1/3)

- **Αξιοποίηση**,

το μέτρο αυτό εκφράζει το ποσοστό του χρόνου κατά το οποίο οι επεξεργαστές ασχολήθηκαν με ωφέλιμες πράξεις.



Μέτρα Απόδοσης - Αξιοποίηση (2/3)

- Το σκιασμένο μέρος του προηγούμενου σχήματος δείχνει την ενασχόληση των επεξεργαστών με ωφέλιμες πράξεις. Τέλεια αξιοποίηση θα είχαμε αν για όλο το χρονικό διάστημα όλοι οι επεξεργαστές έκαναν μόνο χρήσιμες πράξεις, δηλαδή αν όλο το ορθογώνιο εμβαδού **P*T** ήταν σκιασμένο. Επειδή όμως αυτό δεν συμβαίνει, το ποσοστό για το συγκεκριμένο παράδειγμα είναι:
- αξιοποίηση = (εμβαδόν σκιασμένου τμήματος) / (εμβαδόν ορθογωνίου = έργο).

$$18*\tau/P*T = 18/32$$

Μέτρα Απόδοσης - Αξιοποίηση (3/3)

- Δηλαδή αν ο Pj εκτελεί ωφέλιμες πράξεις για μέρος χρόνου qj*T, όπου qj <= 1, ενώ για τον υπόλοιπο χρόνο (1-qj)*T είναι ανενεργός ή επιβαρύνεται με δραστηριότητες που οφείλονται στην ύπαρξη P > 1 επεξεργαστών (π.χ. επικοινωνία), τότε:

$$\alpha\xi\iota o\pi o\acute{\iota}\eta\sigma\eta = \frac{1}{PT}\sum_{j=0}^{p-1} q_j T$$

Μέτρα Απόδοσης - Αξιοποίηση

$$PT = \text{έργο ή κόστος} = \sum_{j=0}^{P-1} q_j T + \sum_{j=0}^{P-1} (1 - q_j) T$$

- είναι το άθροισμα όλων ανεξαιρέτως των χρονικών
- διαστημάτων. Σημειώνεται ότι $\mathbf{q_j T}$= (χρόνος ακολουθιακού υπολογισμού για τον επεξεργαστή **j**) = $\mathbf{n_j \ \tau_\alpha}$, όπου $\mathbf{n_j}$ είναι ο αριθμός των πράξεων που εκτελεί ο επεξεργαστής **j**.

Μέτρα Απόδοσης - Αξιοποίηση: Παράδειγμα (1/2)

- Στο παρακάτω διάγραμμα εμφανίζεται η εκτέλεση ενός παράλληλου αλγόριθμου σε **4** επεξεργαστές. Κατά τα διαστήματα $\boldsymbol{\tau_\alpha}$ εκτελούνται χρήσιμες αριθμητικές πράξεις από τους επεξεργαστές για τους οποίους η αντίστοιχη περιοχή είναι σκιασμένη (οι υπόλοιπο επεξεργαστές είναι ανενεργοί) και κατά τα διαστήματα **τε** γίνεται επικοινωνία.
- Ο αντίστοιχος ακολουθιακός αλγόριθμος χρειάζεται **n=14** πράξεις. Αν $\boldsymbol{\lambda=\tau_\varepsilon/\tau_\alpha}$, συμπληρώστε τις παρακάτω τιμές συναρτήσει των λ, και $\boldsymbol{\tau_\alpha}$.

Μέτρα Απόδοσης - Αξιοποίηση: Παράδειγμα (2/2)

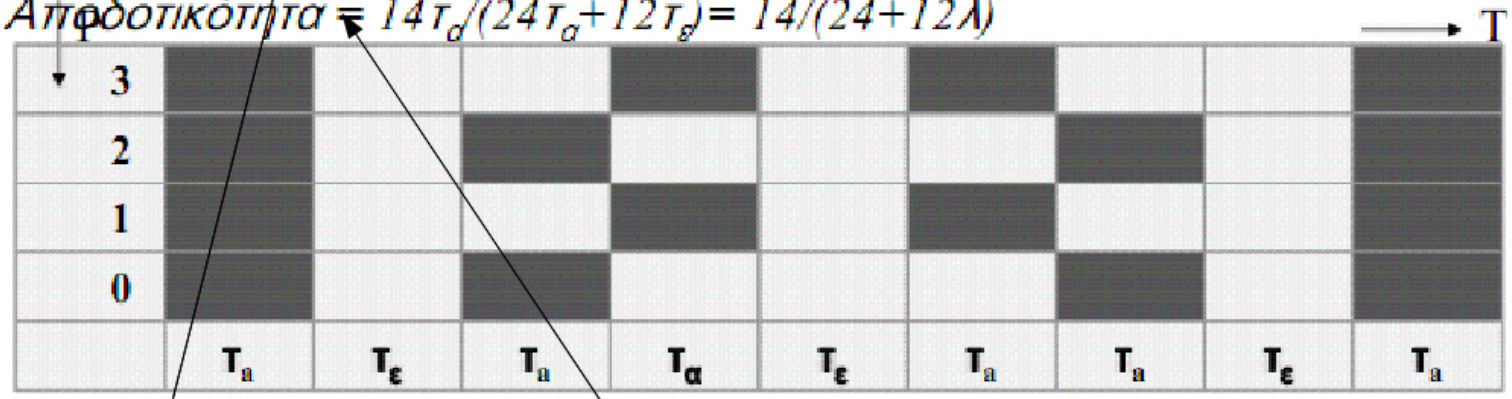
Μας δίνεται στην εκφώνηση

Αν σε κάποια στήλη μαύρο τότε ta αλλιώς τε

$Ακολουθιακός\ Χρόνος = 14\tau_a$ $Παράλληλος\ Χρόνος = 6\tau_a + 3\tau_\varepsilon = 6 + 3\lambda$

$Αξιοποίηση = 16\tau_a/(24\tau_a + 12\tau_\varepsilon) = 16/(24 + 12\lambda)$

$Απόδοτικότητα = 14\tau_a/(24\tau_a + 12\tau_\varepsilon) = 14/(24 + 12\lambda)$



Ακολουθιακός χρόνος προς συνολικό παράλληλο χρόνο

16 μαύρα κουτιά, προς 4γραμμές επί 6 (επεξεργασία) και 4γραμμες επί 3 (επικοινωνία)

Τέλος Ενότητας