ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΕΜ 361: Παράλληλοι Υπολογισμοί

Ενότητα #3: Θεωρία Παράλληλου Προγραμματισμού - Απόδοση

Διδάσκων: Χαρμανδάρης Ευάγγελος
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο 3.0 Ελλάδα

(Attribution – Non Commercial – Non-derivatives 3.0 Greece)

R

[ή επιλογή ενός άλλου από τους έξι συνδυασμούς]

[και αντικατάσταση λογότυπου άδειας όπου αυτό έχει μπει (σελ. 1, σελ. 2 και τελευταία)]

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» 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.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΜ 361: Παράλληλοι Υπολογισμοί

Χαρμανδάρης Βαγγέλης, Τμήμα Εφαρμοσμένων Μαθηματικών

Πανεπιστήμιο Κρήτης, Χειμερινό Εξάμηνο 2010/11

Κεφάλαιο 3: Θεωρία Παράλληλου Προγραμματισμού - Απόδοση

- **Απόδοση Παράλληλων Προγραμμάτων.**
- **Νόμος Amdahl.**
- **Νόμος Gustafson.**
- **Karp-Flatt metric.**
- **Τεστ Αναφοράς: Linpack.**

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:

• Νόμος του Amdahl

• Νόμος του Gustafson

Απόδοση Παράλληλων Προγραμμάτων

• 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:

-- Μικρή επιμέριση: Μικρός φόρτος εργασίας σε κάθε επεξεργαστή, πολύ επικοινωνία.

-- Μεγάλη επιμέριση: Μεγάλος φόρτος εργασίας σε κάθε επεξεργαστή, λίγη επικοινωνία.

• Βασικός στόχος η μεγιστοποίηση του λόγου:

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

Speedup Factor (Λόγος Επιτάχυνσης)

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

$$\boxed{S(P) = \frac{\text{χρόνος εκτέλεσης σε ένα επεξεργαστή}}{\text{χρόνος εκτέλεσης σε } P \text{ επεξεργαστές}} = \frac{T_s}{T_p}}$$

- Απόλυτος λόγος επιτάχυνσης: αν T_s είναι ο χρόνος του καλύτερου σειριακού αλγόριθμου.
- Σχετικός λόγος επιτάχυνσης: αν T_s 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 (optimization) 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 ($S(P) << P$) τότε:

1) είτε το πρόβλημα δεν παραλληλίζεται,
2) είτε ο παράλληλος αλγόριθμος χρειάζεται βελτίωση!

Προσοχή:

• Ο μέγιστος λόγος επιτάχυνσης δεν έχει νόημα αν είναι $<< P$.

• Καθώς ο αριθμός των επεξεργαστών, n, αυξάνεται, το σειριακό κομμάτι του κώδικα αποτελεί ολοένα και μεγαλύτερο τμήμα του συνολικού χρόνου εκτέλεσης (Νόμος του Amdahl).

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:

$$\boxed{S(P) \equiv \frac{T_S}{T_P} = \frac{t_s}{f t_s + \frac{(1-f)t_s}{P}} = \frac{P}{1+(P-1)f}}$$

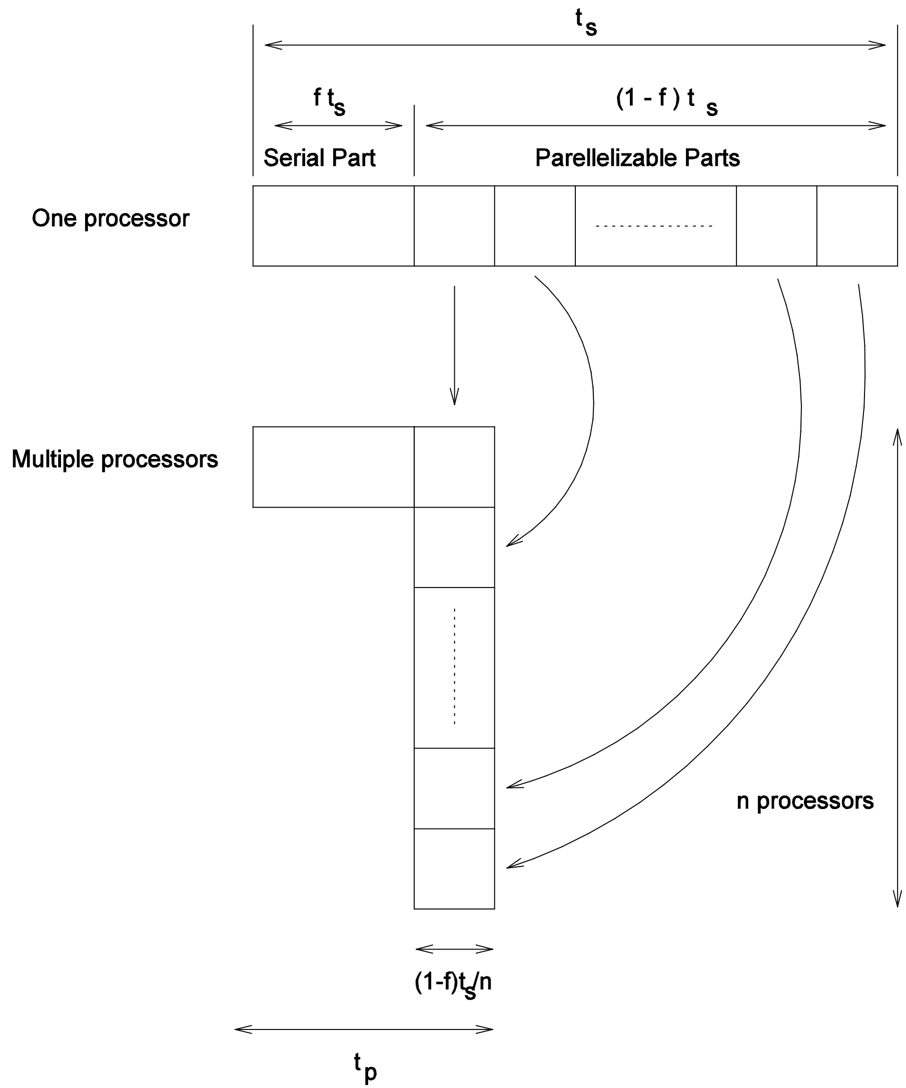
• Το όριο για $P \rightarrow \infty$ (μέγιστη επιτάχυνση) είναι :

$$\boxed{S_{\max} = \lim_{P\to\infty} S(P) = \frac{1}{f}}$$

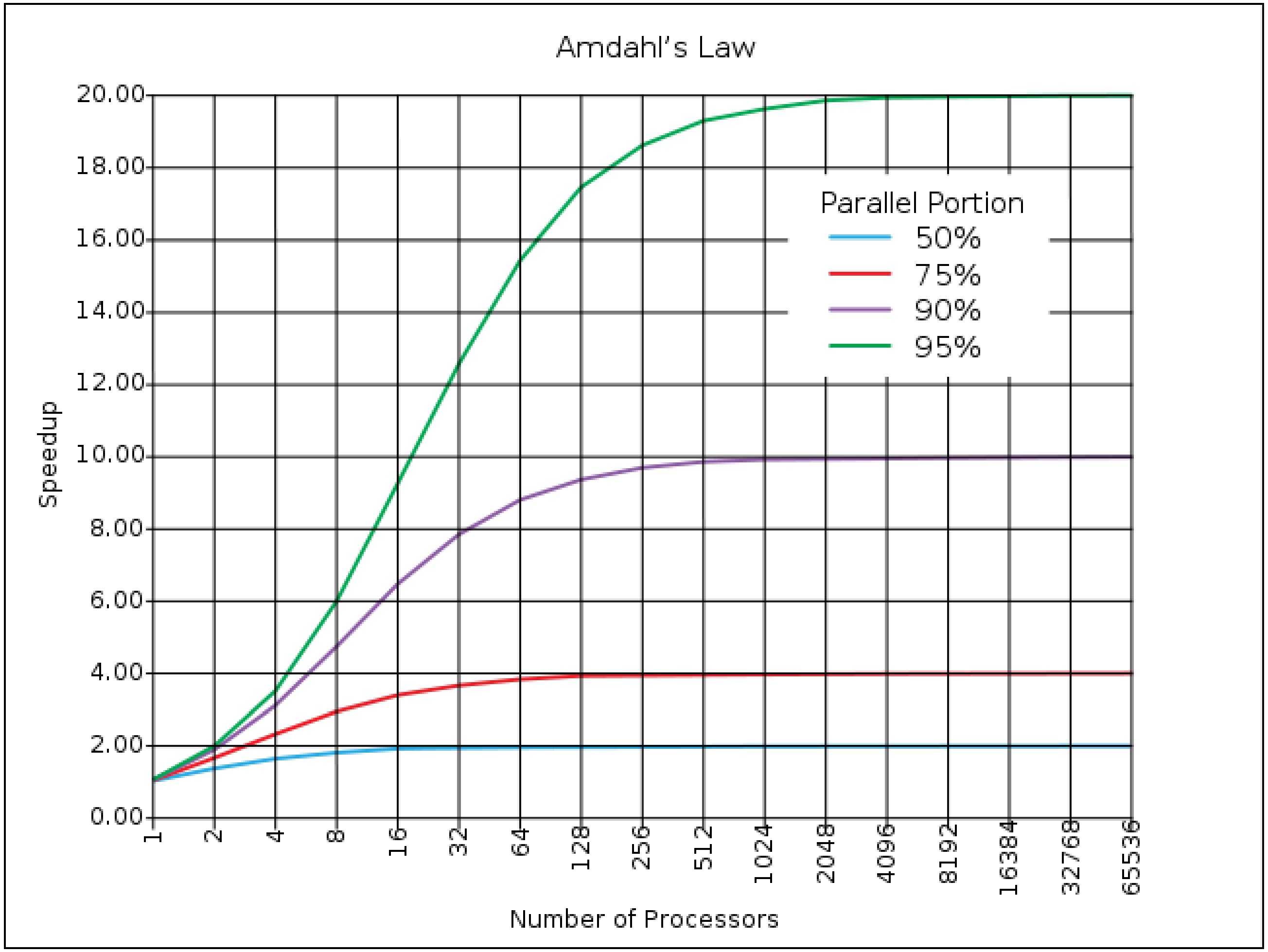
• **Προσοχή:** ακόμη και αν μόνο το 10% του προβλήματος είναι σειριακό τότε $S_{max} = 10$!

• Μεγάλο κομμάτι της παράλληλης επεξεργασίας αφορά την ελαχιστοποίηση του f.

Νόμος του Amdahl



Νόμος του Amdahl



Αποδοτικότητα, Επεκτασιμότητα

Αποδοτικότητα (Efficiency) :

$$E(P) = \frac{S(P)}{P} = \frac{T_s}{PT_p}$$

-- Αλγόριθμοι με σχεδόν γραμμική επιτάχυνση: $E \sim 1$

-- Πολλοί δύσκολα παραλληλίσιμοι αλγόριθμοι έχουν: $E \sim 1/\log(P)$, $E \rightarrow 0$ με $P \rightarrow \infty$!

• Επεκτασιμότητα υλικού είναι η δυνατότητα αναβάθμισης των συστημάτων (hardware) με αποτέλεσμα την αύξηση της απόδοσης.

-- Όμως περισσότεροι επεξεργαστές → μεγαλύτερο δίκτυο → αύξηση της επικοινωνίας → πτώση της απόδοσης!

• 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 αλγόριθμου.

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:

$$\boxed{S(P) = f(n) + P(1 - f(n)) = P - f(n)(P-1)}$$

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* → ∞ είναι :

$$\boxed{S_{\max} = \lim_{n \to \infty} S(P) = P}$$

- 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.

Karp-Flatt Metric (1990)

• Η μετρική Karp-Flatt είναι ένας δείκτης παραλληλισμού ενός κώδικα σε παράλληλα συστήματα χρησιμοποιώντας το σειριακό μέρος.

• Έστω ότι ένας παράλληλος κώδικας «τρέχει» σε P επεξεργαστές και ότι T_S είναι ο χρόνος εκτέλεσης του σειριακού μέρους και T_P ο χρόνος εκτέλεσης του παράλληλου μέρους σε ένα επεξεργαστή. Τότε ο συνολικός χρόνος εκτέλεσης του προγράμματος σε P επεξεργαστές, $T(P)$, είναι:

$$\boxed{T(P) = T_s + \frac{T_P}{P}}$$

• Ορίζοντας το «πειραματικό» σειριακό μέρος ως $e = T_s/T(1)$ έχουμε:

$$\boxed{T(P) = T(1)e + \frac{T(1)(1-e)}{P}}$$

Karp-Flatt Metric (1990)

• Χρησιμοποιώντας το λόγος επιτάχυνσης, $S=T(1)/T(P)$, τελικά:

$$\boxed{\frac{1}{S} = e + \frac{1-e}{P} \Rightarrow e = \frac{\frac{P}{S} - 1}{P - 1}}$$

• Η χρήση του «πειραματικού» σειριακού μέρους βοηθάει στο να έχουμε ένα εμπειρικό μέτρο που μπορεί να χρησιμοποιηθεί πρακτικά στη ανάλυση περίπλοκων αλγορίθμων.

• Όσο μικρότερο το *e* τόσο καλύτερος ο παραλληλισμός.

• Για δεδομένο μέγεθος προβλήματος *n* η αποτελεσματικότητα του παράλληλου προγράμματος συνήθως μειώνεται όσο ο αριθμός των επεξεργαστών αυξάνει. Χρησιμοποιώντας την μετρική Karp-Flatt μπορούμε να υπολογίσουμε αν η μείωση της αποτελεσματικότητας οφείλεται στο τύπου του προβλήματος ή στον αλγόριθμο.

Το Τεστ Αναφοράς LINPACK

• Το Τεστ Αναφοράς LINPACK (LINPACK Benchmarks) είναι ένα μέτρο της ταχύτητας εκτέλεσης πράξεων πραγματικών αριθμών ενός υπολογιστικού συστήματος.

• Αποτελείται από μια σειρά χαρακτηριστικών υπο-προγραμμάτων τα οποία λύνουν ένα σύστημα γραμμικών εξισώσεων. Τα προγράμματα είναι γραμμένα σε FORTRAN ή C και ο αλγόριθμος είναι ‘‘LU decomposition with partially pivoting”.

• Το LINPACK χρησιμοποιεί μία χαρακτηριστική βιβλιοθήκη γραμμικής άλγεβρας (BLAS, Basic Linear Algebra Subprograms) για εκτέλεση βασικών πράξεων πινάκων.

• Ο συνολικός αριθμός πράξεων του αλγόριθμου επίλυσης συστήματος n γραμμικών εξισώσεων είναι: $2/3\, n^3 + n^2 + O(n)$.

• R_∞: η απόδοση του αλγόριθμου σε flops για πολύ μεγάλο πρόβλημα ($n=N_{max}$).

• R_{Peak}: η θεωρητική απόδοση για κάθε σύστημα.

Το Τεστ Αναφοράς LINPACK

• Λίστα επίδοσης στο τεστ LINPACK των καλύτερων υπολογιστικών συστημάτων στον κόσμο (06/2010).

Name	Number of CPU’s	R_∞ (Gflops)	N_{max} (Order)	R_{Peak} (Gflops)
Jaguar, Cray XT	224256	1759000	5474272	2331000
Nebulae, , Intel X5650	120640	1271000	2359296	2984300
Roadrunner, IBM Cluster	122400	1042000	2249343	1375776
Kraken XT5, Cray XT	98928	831700	2078999	1028851
Juelich (FZJ), IBM Cluster	294912	825500	4043519	1002700

Βιβλιογραφία

- *Parallel Programming*, B. Wilkinson, M. Allen, Prentice Hall, 2nd Ed. 2005.
- *Parallel Computing: Theory and Practice*, M. J. Quinn, McGraw-Hill, 1994.
- *Scientific Computing: An introduction with Parallel Computing*, G. Golub, J. Ortega, Academic Press, 1993.
- http://en.wikipedia.org/wiki/Parallel_computing
- LINPACK: http://www.netlib.org/linpack/, http://www.top500.org/
- BLAS: http://www.netlib.org/blas/

Τέλος Ενότητας