

**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΑΡΑΛΛΗΛΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ**

**Heat 2D (MPI-OpenMP)**

**Χρήστος Πατσούρας Μ1612**

**ΑΘΗΝΑ**

**ΜΑΡΤΙΟΣ 2019**

**ΠΕΡΙΕΧΟΜΕΝΑ**

[1. ΕΙΣΑΓΩΓΗ 6](#_Toc200090)

[2. ΣΧΕΔΙΑΣΜΟΣ ΔΙΑΜΟΙΡΑΣΜΟΥ ΔΕΔΟΜΕΝΩΝ ΣΤΙΣ ΔΙΕΡΓΑΣΙΕΣ 7](#_Toc200091)

[2.1 Partitioning 7](#_Toc200092)

[2.2 Communication 7](#_Toc200093)

[2.3 Agglomeration 7](#_Toc200094)

[2.3.1 Διαχωρισμός σε λωρίδες 8](#_Toc200095)

[2.3.2 Διαχωρισμός σε blocks 9](#_Toc200096)

[2.3.3 Σύγκριση διαχωρισμών 9](#_Toc200097)

[2.4 Mapping 9](#_Toc200098)

[3. ΣΧΕΔΙΑΣΜΟΣ ΚΩΔΙΚΑ 10](#_Toc200099)

[4. ΜΕΤΡΗΣΕΙΣ 11](#_Toc200100)

[4.1 MPI πρόγραμμα χωρίς έλεγχο σύγκλισης 11](#_Toc200101)

[4.2 MPI πρόγραμμα με έλεγχο σύγκλισης κάθε ν επαναλήψεις 11](#_Toc200102)

[4.3 Υβριδικό MPI με OpenMP 11](#_Toc200103)

[4.4 Σύκριση αναλυτικών υπολογισμών και πραγματικών μετρήσεων 11](#_Toc200104)

[5. ΣΥΓΚΡΙΣΗ ΜΕ ΑΡΧΙΚΟ ΠΡΟΓΡΑΜΜΑ 12](#_Toc200105)

[6. ΕΞΟΔΟΣ MpiP 13](#_Toc200106)

[7. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΑΡΑΤΗΡΗΣΕΙΣ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ 14](#_Toc200107)

**ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ**

Σχήμα 1: Υπόδειγμα διαμόρφωσης ράχης βιβλιόδετου τόμου 14

Σχήμα 2: Στοιχεία ανάπτυξης Ανατολής, Βορρά και Δύσης 21

**ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ**

[Εικόνα 1: Ερημικό τοπίο κατά το ηλιοβασίλεμασελ. 17](#_Toc220142383)

**ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ**

[Πίνακας 1: Συντομογραφίες χωρώνσελ. 15](#_Toc220142393)

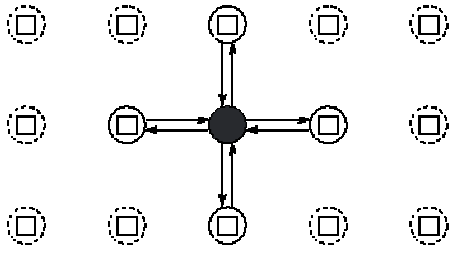
[Πίνακας 2: Πίνακας ορολογίας με τις αντιστοιχίσεις των ελληνικών και ξενόγλωσσων όρωνσελ. 21](#_Toc220142394)

[Πίνακας 3: Ακρωνύμια και ανάπτυξή τουςσελ. 22](#_Toc220142395)

# ΕΙΣΑΓΩΓΗ

Στόχος της εφαρμογής είναι η προσομοίωση μεταφοράς θερμότητας σε επιφάνεια. Στην ουσία έχουμε ένα πίνακα διάστασης , όπου κάθε στοιχείο του πίνακα είναι ένα σημείο του χώρου και έχει μια συγκεκριμένη θερμοκρασία. Η αρχική θερμοκρασία είναι υψηλή στο κέντρο και μηδέν στα ακριανά στοιχεία. Το σύστημα αλλάζει κατάσταση με την πάροδο του χρόνου. Αυτό συμβαίνει διότι κάθε σημείο επηρεάζει και επηρεάζεται από τα γειτονικά του. Γειτονικά στοιχεία θεωρούνται αυτά που βρίσκονται πάνω, κάτω, δεξιά και αριστερά σε ένα στοιχείο, όπως στην Εικόνα 1. Τα στοιχεία που βρίσκονται στα άκρα δεν αλλάζουν τιμές καθώς θεωρούνται στοιχεία που απορροφούν ή εκπέμπουν θερμότητα στο σύστημα.

Το πρόγραμμα που δίνεται ως πρότυπο για τις απαιτήσεις της άσκησης είναι διαθέσιμο [εδώ](https://computing.llnl.gov/tutorials/mpi/samples/C/mpi_heat2D.c) και ζητούμενο είναι η αξιολόγηση, ο επανασχεδιασμός και η βελτίωση του με στόχο την καλύτερη κλιμάκωση και η σύγκριση χρονικά, σε speedup και σε efficiency του προγράμματος αυτού με αυτά που θα παραχθούν σε MPI και σε MPI+OpenMP (υβριδικό).



Εικόνα 1: Γείτονες σημείου

# ΣΧΕΔΙΑΣΜΟΣ ΔΙΑΜΟΙΡΑΣΜΟΥ ΔΕΔΟΜΕΝΩΝ ΣΤΙΣ ΔΙΕΡΓΑΣΙΕΣ

Partitioning

Εν γένει το partitioning μπορεί να γίνει ως προς τα δεδομένα και ως προς τους υπολογισμούς ή τη λειτουργία που πρέπει να επιτελεστεί. Σε αυτή την περίπτωση, το partitioning μπορεί να γίνει μόνο ως προς τα δεδομένα υπάρχει ένας μοναδικός υπολογισμός για όλα τα δεδομένα. Συγκεκριμένα με διάφορους τρόπους μπορούμε να αποφασίσουμε πως θα διαχωρίσουμε τα δεδομένα του πίνακα. Εφόσον ο φόρτος επεξεργασίας είναι ίδιος για όλο τον πίνακα δεν υπάρχει λόγος να γίνει κάποιος κυκλικός διαμοιρασμός. Οπότε τον πίνακα θα τον χωρίσουμε είτε σε γραμμές είτε σε στήλες είτε σε blocks. Στο δοσμένο πρόγραμμα ο διαχωρισμός γίνεται σε γραμμές. Εμείς θα αποδείξουμε ότι ο διαχωρισμός σε blocks είναι ορθότερος ως προς την κλιμάκωση και θα τον ακολουθήσουμε και στην υλοποίηση του νέου προγράμματος.

Communication

Όπως έχουμε αναφέρει, η τιμή ενός στοιχείου εξαρτάται από τα τέσσερα γειτονικά του σε διάταξη βοράς-νότος-ανατολή-δύση. Άρα σε περίπτωση διαχωρισμού του πίνακα, οι διεργασίες πρέπει να ανταλλάσσουν πληροφορίες των στοιχείων που χρειάζονται οι «γειτονικές» διεργασίες για τους υπολογισμούς που πραγματοποιούν. Αυτές θα είναι είτε οι ακραίες γραμμές είτε οι ακραίες στήλες στις περιπτώσεις διαχωρισμού σε οριζόντιες λωρίδες και σε κάθετες λωρίδες αντίστοιχα και συγκεκριμένα η πρώτη και η τελευταία κάθε υποπίνακα. Στην περίπτωση διαχωρισμού σε blocks θα είναι όλα τα ακραία στοιχεία, συγκεκριμένα η πρώτη και η τελευταία γραμμή και στήλη κάθε υποπίνακα. Θεωρούμε ότι η επικοινωνία μεταξύ των υπολογιστών είναι ίδιων τεχνικών χαρακτηριστικών και ότι δεν υπάρχει επικοινωνιακός φόρτος από άλλες εφαρμογές.

Η επικοινωνία στο συγκεκριμένο πρόβλημα είναι:

* Local: επικοινωνούμε με ένα μικρό τμήμα των διεργασιών, όχι με όλες
* Structured: Ακολουθούμε καρτεσιανή τοπολογία για να δομίσουμε την επικοινωνία και μιλάμε μόνο με τέσσερις γείτονες στην περίπτωση του διαχωρισμού σε blocks. Στην περίπτωση του διαχωρισμού σε γραμμές ή στήλες έχουμε μόνο δύο γείτονες, όπως στο παράδειγμα που δίνεται.
* Static: οι γείτονες είναι πάντα οι ίδιοι, δεν αλλάζουν με την πάροδο του χρόνου
* Synchronous: οι παραγωγοί και οι καταναλωτές των μηνυμάτων εκτελούνται συντονισμένα και συνεργάζονται για τη μεταφορά δεδομένων

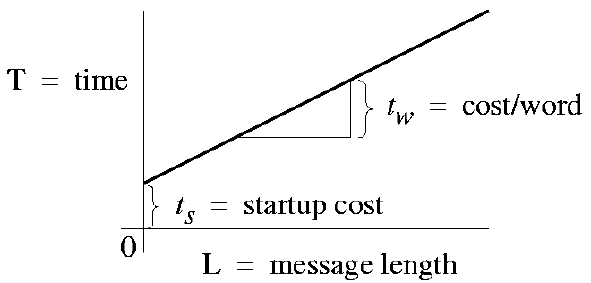
Στη δική μας υλοποίηση θα ακολουθήσουμε το διαχωρισμό σε blocks και τη διάταξη με τους τέσσερις γείτονες για κάθε διεργασία.

## Agglomeration

Θα εξετάσουμε και τις δύο προαναφερθείσες περιπτώσεις διαχωρισμού, δηλαδή σε λωρίδες (γραμμές ή στήλες) και blocks, και θα αποδείξουμε ότι ο διαχωρισμός σε blocks κλιμακώνει καλύτερα. Έστω ότι έχουμε ένα πίνακα . Χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι . Αφού έχουμε για να έχουμε λιγότερα στοιχεία για ανταλλαγή, ο διαχωρισμός σε λωρίδες θα είναι σε στήλες. Δε θα είχε διαφορά ο διαχωρισμός σε γραμμές. Ο συνολικός χρόνος εκτέλεσης ενός παράλληλου προγράμματος υπολογίζεται ως εξής:

όπου : χρόνος υπολογισμών, : χρόνος επικοινωνίας και : χρόνος αδράνειας/αναμονής.

Η διάρκεια του χρόνου αναμονής δε μπορεί να προβλεφθεί για αυτό και δε θα υπολογιστεί στο αποτέλεσμα. Αφού έχουμε ομοιογενή συστήματα, το για ένα στοιχείο θα είναι παντού ίδιο και δεν εξαρτάται από τον διαμοιρασμό, αλλά από τη φύση του προβλήματος. Επίσης για κάθε επικοινωνία που θα κάνει κάθε διεργασία το κόστος υπολογίζεται μέσω του τύπου



Εικόνα 2: Απλό μοντέλο επικοινωνίας

### Διαχωρισμός σε λωρίδες

Στην περίπτωση που γίνει ο διαχωρισμός σε λωρίδες, αφού , ο διαχωρισμός θα γίνει σε στήλες για να έχουμε λιγότερες ανταλλαγές στοιχείων. Άρα κάθε διεργασία θα έχει στοιχεία, αλλά οι πρώτες θα έχουν μια επιπλέον στήλη. Επίσης η πρώτη και η τελευταία διεργασία θα έχει μια λιγότερη στήλη. Άρα σύμφωνα με το παραπάνω ο χειρότερος χρόνος που μπορούμε να πετύχουμε είναι:

Για την επικοινωνία θα πάρουμε τη χειρότερη περίπτωση όπου μια διεργασία έχει κομμάτι του πίνακα που έχει δυο γείτονες. Από την ανταλλαγή των δυο στηλών που θα γίνει μεταξύ των δυο διεργασιών προκύπτει το εξής:

Συνεπώς:

* Execution time:
* Speedup:
* Efficiency:

### Διαχωρισμός σε blocks

Θεωρούμε ότι ο διαχωρισμός γίνεται σε P blocks και χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι ο P είναι τέλειο τετράγωνο και κάθε διάσταση μοιράζεται σε blocks. Σε αυτήν την περίπτωση σε κάθε μπλοκ θα έχουμε στοιχεία σε κάθε block. Οι πρώτες διεργασίες θα έχουν μια επιπλέον στήλη, ενώ οι πρώτες μια επιπλέον γραμμή. Επίσης όλες οι διεργασίες που θα περιλαμβάνουν ακραία στοιχεία θα έχουν μια λιγότερη γραμμή ή/και στήλη. Άρα σύμφωνα με το παραπάνω ο χειρότερος χρόνος που μπορούμε να πετύχουμε είναι:

Για την επικοινωνία θα πάρουμε τη χειρότερη περίπτωση όπου μια διεργασία έχει κομμάτι του πίνακα που έχει δυο γείτονες. Από την ανταλλαγή των δυο στηλών που θα γίνει μεταξύ των δυο διεργασιών προκύπτει το εξής:

Συνεπώς:

* Execution time:
* Speedup:
* Efficiency:

### Σύγκριση διαχωρισμών

Από μετρήσεις που έγιναν ανάμεσα σε δυο υπολογιστές του εργαστηρίου με το mpptest και από εκτέλεση του ανάλογου κώδικα σε ένα δοκιμαστικό πίνακα βρέθηκαν ότι = , = και = .

Mapping

Υπάρχουν δύο γενικές στρατηγικές στο mapping:

* Οι λειτουργίες που μπορούν να εκτελεστούν ταυτόχρονα τοποθετούνται σε διαφορετικούς επεξεργαστές, ώστε να μπορούμε να εκμεταλλευτούμε αυτή τη δυνατότητα
* Οι διεργασίες που επικοινωνούν συχνά μεταξύ τους, τοποθετούνται στον ίδιο κόμβο για να αποφεύγεται όσο είναι δυνατό η interprocessor επικοινωνία

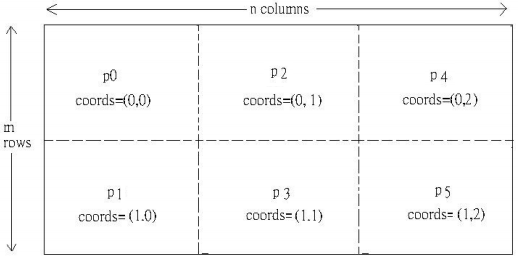
Εφόσον έχουμε να επιτελέσουμε μία μόνο διεργασία και υπάρχει ομοιογένεια στους επεξεργαστές και την επικοινωνία που έχουμε στη διάθεσή μας, για καλύτερη απόδοση θα αναθέσουμε μια διεργασία ανά επεξεργαστικό πυρήνα.

\*\*\* MPI\_Get\_processor\_name() και screenshot \*\*\*

# ΣΧΕΔΙΑΣΜΟΣ ΚΩΔΙΚΑ

Σχεδιασμός MPI προγράμματος

Σύμφωνα και με όσα αποδείχτηκαν στην ενότητα 2.3.3, ο ορθότερος ως προς την κλιμάκωση διαχωρισμός είναι σε blocks δύο διαστάσεων. Κάθε διεργασία θα αναλάβει έναν υποπίνακα. Για την επικοινωνία θα δημιουργήσουμε ένα δισδιάστατο καρτεσιανό πλέγμα διεργασιών στη μορφή που φαίνεται στην Εικόνα 3. Συγκεκριμένα, η πρώτη διεργασία θα πάρει το πάνω αριστερά κομμάτι και η ανάθεση θα συνεχιστεί πρώτα προς τα κάτω και μετα προς τα δεξιά. Η πρώτη βασική διαφοροποίηση με το δοσμένο πρόγραμμα είναι ότι θα χρησιμοποιήσουμε και τη master διεργασία για υπολογισμούς. Για αυτό η αρίθμιση στην Εικόνα 3 ξεκινάει απο το μηδέν. Κάθε διεργασία θα επικοινωνεί με τους τέσσερις γείτονες της (βοράς-νότος-ανατολή-δύση), όπως ορίζονται από το πλέγμα. Αν κάποιος γείτονας δεν υπάρχει (π.χ. ο p0 δεν έχει βορά και δύση στο παράδειγμά μας), θα τίθεται ίσος με MPI\_PROC\_NULL.



Εικόνα 3: Δισδιάστατο καρτεσιανό πλέγμα διεργασιών

Σχεδιασμός υβριδικού προγράμματος

# ΜΕΤΡΗΣΕΙΣ

MPI πρόγραμμα χωρίς έλεγχο σύγκλισης

Κείμενο

MPI πρόγραμμα με έλεγχο σύγκλισης κάθε ν επαναλήψεις

Κείμενο

Υβριδικό MPI με OpenMP

Κείμενο

Σύκριση αναλυτικών υπολογισμών και πραγματικών μετρήσεων

Κείμενο

# ΣΥΓΚΡΙΣΗ ΜΕ ΑΡΧΙΚΟ ΠΡΟΓΡΑΜΜΑ

# ΕΞΟΔΟΣ MpiP

# ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΑΡΑΤΗΡΗΣΕΙΣ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ