εθνικο και καποδιστριακο πανεπιστημιο αθηνων

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

**Παράλληλα Υπολογιστικά Συστήματα**

Βασιλικάκης Χαρίδημος M1478

Κολιοπούλου Κωνσταντίνα Μ1551

Αθήνα

Σεπτέμβριος 2019

INDEX

[ΕΙΣΑΓΩΓΗ 2](#_Toc20699589)

[MPI 3](#_Toc20699590)

[Το πρόβλημα 3](#_Toc20699591)

[Αρχική Υλοποίηση 3](#_Toc20699592)

[Βελτιωμένη Υλοποίηση 3](#_Toc20699593)

# ΕΙΣΑΓΩΓΗ

Η παρούσα εργασία είναι το αποτέλεσμα της υλοποίησης ενός προγράμματος το οποίο προσομοιώνει τη μεταφορά θερμότητας σε μια 2D επιφάνεια.

Για την υλοποίησή του χρησιμοποιήσαμε το περιβάλλον του **MPI**. Σκοπός αυτής της εργασίας είναι η όσο το δυνατόν μεγαλύτερη επιτάχυνση των υπολογισμών η οποία μπορεί να επιτευχθεί με τον κατάλληλο διαμοιρασμό των δεδομένων σε **blocks**στις διεργασίες, την **ασύγχρονη** επικοινωνία και με την **παραλληλοποίηση** τους. Για την παραλληλοποίηση των διεργασιών δημιουργήθηκε ένα υβριδικό πρόγραμμα το οποίο αποτελεί ενα συνδυασμό **MPI** και **OpenMP**. Για κάθε μία από τις υλοποιήσεις θα παρουσιάσουμε τη δομή της και ενδεικτικά αποτελέσματα μετρήσεων τα οποία και θα χρησιμοποιηθούν για να συγκρίνουμε κάθε μέθοδο/τεχνολογία ξεχωριστά. Για τις μετρήσεις χρησιμοποιήθηκαν εργαλεία/βιβλιοθήκες όπως η **mpiP**(MPI Profiling) και το **Mpptest**.

# MPI

## Το πρόβλημα

Το πρόβλημα το οποίο καλούμαστε να επιλύσουμε αφορά τον **Heat2D** αλγόριθμο. Ο αλγόριθμος αυτός αποτελεί μια απλοποιημένη αναπαράσταση της μεταφοράς θερμότητας σε δύο διαστάσεις. Έστω λοιπόν ότι έχουμε έναν πίνακα. Η τιμή της αρχικής θερμοκρασίας υπολογίζεται ότι είναι υψηλότερη στο μέσο και μηδέν στα άκρα. Σκοπός είναι η επιτάχυνση της εναλλαγής παλαιών και νέων δεδομένων μεταξύ των διεργασιών των workers, ώστε να γίνει ο υπολογισμός της θερμοκρασίας για κάθε σημείο. Σε κάθε βήμα κάθε διεργασία πρέπει να ανταλλάξει δεδομένα με τις γειτονικές της αφού για να υπολογιστεί η τρέχουσα θερμοκρασία για ένα σημείο στον πίνακα απαιτείτε να γνωρίζουμε την προηγούμενη χρονικά τιμή της συν τις τιμές των γειτονικών σημείων.

## Αρχική Υλοποίηση

Στην αρχική υλοποίηση του προγράμματος μας έχει δοθεί ένα MPI πρόγραμμα το οποίο υλοποιεί τα παραπάνω χωρίς όμως να χρησιμοποιεί ασύγχρονη επικοινωνία αλλά **blocking** επικοινωνία μεταξύ των διεργασιών, διαμοιράζοντας τα δεδομένα κατά **ζώνες/σειρές** σε μια διάσταση.

## Βελτιωμένη Υλοποίηση

Για τη βελτιστοποίηση του αρχικού προγράμματος χρειάστηκε να κάνουμε πολλές αλλαγές, τόσο στην επικοινωνία μεταξύ των διεργασίων όσο και στον τρόπο με των οποίων πραγματοποιείται ο διαχωρισμός των δεδομένων. Ακολουθούν αναλυτικότερα παρακάτω.

Διαμοιρασμός Δεδομένων σε Blocks

Αποτελεί μια από τις σημαντικότερες αλλαγές στη βελτιστοποίηση του αρχικού προγράμματος. Στην περίπτωση της υλοποίησης μας τα δεδομένα κατατμίζονται και αποστέλλονται σε **blocks** εν αντιθέσει του διαχωρισμού σε ζώνες όπως είχε δοθεί αρχικά. Σύμφωνα με τη μεθοδολογία Foster αυτός είναι και ο πιο αποδοτικός τρόπος προκειμένου η λύση του προβλήματος να παραλληλοποιηθεί.

Δοθέντος ενός ΝxN πίνακα σε κάθε επανάληψη παράγουμε έναν καινούριο NxN πίνακα με θερμοκρασίες. Κάθε διεργασία έχει ένα block του πίνακα και για τον υπολογισμό κάθε νέου στοιχείου του πίνακα, όπως προαναφέρθηκε, χρειάζεται να γνωρίζουμε και τις τιμές των γειτόνων του. Οι υπολογισμοί γίνονται κατανεμημένα και κάθε διεργασία/worker έχει ένα block το οποίο σε κάθε επανάληψη πρέπει να στείλει την πληροφορία του στους γείτονες του και να λάβει την αντίστοιχη πληροφορία από αυτούς.

Καρτεσιανή Τοπολογία

Για τον καλύτερο διαμοιρασμό των διεργασιών θεωρήσαμε ως βέλτιστο τρόπο τη χρήση **Καρτεσιανής Τοπολογίας**(Virtual Topology). Με αυτό τον τρόπο η ίδια βιβλιοθήκη του MPI με βάση την αρχιτεκτονική του συστήματος αποφασίζει το βέλτιστο τρόπο κατανομής των διεργασιών. Ως αποτέλεσμα να ελαχιστοποιείται η επικοινωνία μεταξύ των διάφορων μηχανών στο δίκτυο και κατ’ επέκταση και ο χρόνος εκτέλεσης. Επιπλέον, η χρήση Καρτεσιανής Τοπολογίας παρέχει τη δυνατότητα μέσω της συνάρτησης **MPI\_Cart\_Shift** άμεσης εύρεσης των γειτονικών διεργασιών ώστε να μη χρειάζεται η υλοποίηση περιττών και χρονοβόρων υπολογισμών. Άλλο ένα πλεονέκτημα της συγκεκριμένης τοπολογίας το οποίο εκμεταλλευτήκαμε είναι η χρήση της συνάρτησης **MPI\_Dim**. Η MPI\_Dim υπολογίζει το βέλτιστο γινόμενο μεταξύ του αριθμού των διεργασιών και του μεγέθους του πίνακα, προκειμένου να χωριστεί ο πίνακας στο καρτεσιανό επίπεδο. Με αυτό τον τρόπο δεν υπάρχει κάποιος περιορισμός όσον αφορά τον αριθμό των διεργασιών που θα δοθούν ως είσοδος και για το αν αυτός ο αριθμός θα διαιρεί τέλεια το μέγεθος του πίνακα.

Επικοινωνία και αποστολή block

Σε αντίθεση με την αρχική υλοποίηση που μας δόθηκε η επικοινωνία μεταξύ των διεργασιών είναι ασύγχρονη (**non-blocking**). Αυτό επιτεύχθηκε χρησιμοποιώντας τις συναρτήσεις **MPI\_Send\_Init** και **MPI\_Recv\_Init** έναντι των **MPI\_Send** και **MPI\_Recv** που είναι **blocking**. Με αυτό τον τρόπο οι διεργασίες μπορούν συνεχίζουν τους εσωτερικούς τους υπολογισμούς χωρίς να χρειάζεται να περιμένουν την ολοκλήρωση της αποστολής των γειτονικών δεδομένων. Στη συνέχεια εφόσον έχουμε διασφαλίσει μέσω της **MPI\_WaitAll** πως έχουν ληφθεί όλα τα απαραίτητα δεδομένα και από τις γειτονικές διεργασίες πραγματοποιούνται και οι υπόλοιπο υπολογισμοί. Ο λόγος επιλογής των συγκεκριμένων συναρτήσεων έναντι των **MPI\_ISend** και **MPI\_IRecv** ,που είναι εξίσου non-blocking, υπάγεται στο γεγονός ότι η αρχικοποίηση του περιβάλλοντος για τα δεδομένα που θα σταλούν/ληφθούν γίνεται μια φορά, εφόσον δεν αλλάζουν οι διεργασίες που στέλνουν/λαμβάνουν δεδομένα, ώστε να μη χρειάζεται αυτό να γίνεται σε κάθε επανάληψη.

Για την αποστολή των δεδομένων μεταξύ των διεργασιών ορίσαμε δικά μας **MPI\_Datatype**. Με αυτό τον τρόπο επιτυγχάνεται η αποφυγή περιττών αντιγραφών δεδομένων στη μνήμη. Στην υλοποίηση μας όρισαμε δύο Datatypes μέσω της **MPI\_Type\_vector**, ένα για στήλες και ένα για γραμμές αντίστοιχα.

Τέλος, σε κάθε block έχουν προστεθέι επιπλέον δύο γραμμές και δύο στήλες προκειμένου να αποθηκεύονται τα δεδομένα που θα ληφθούν από τις γειτονικές διεργασίες και να διατηρούνται τα προηγούμενα δεδομένων καθώς είναι αναγκαία για την ολοκλήρωση των υπολογισμών.