Project: Παράλληλη Επεξεργασία

Υλοποιήσεις παράλληλου κώδικα με τη βοήθεια των εργαλείων: OpenMP OpenMP\_Tasks MPI Υβριδικό μοντέλο OpenMP-MPI

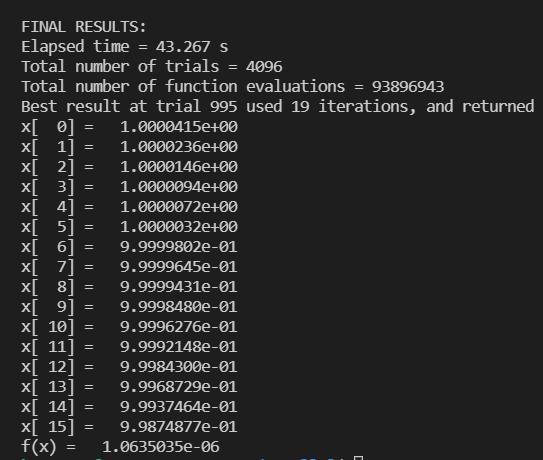
**Η ΟΜΑΔΑ:**

Βλαχογιάννης Δημήτρης, ΑΜ:1067371 Μητροπούλου Κατερίνα,ΑΜ:1067409 Στεφανίδης Μάριος, ΑΜ:1067458

**Σημείωση:** αναφέρουμε ότι λόγω προβλημάτων hardware μειώνουμε το πλήθος των δειγμάτων που λαμβάνουμε σε 4\*1024, καθώς στην αρχική περίπτωση το πρόγραμμα χρειαζόταν αρκετή ώρα για να βγάλει αποτέλεσμα στην περίπτωση του μη παράλληλου προγράμματος (γύρω στα 45 λεπτά), ενώ αργούσε σχετικά και στους παράλληλους κώδικες. Οπότε για λόγους ευκολίας μειώνουμε τα δείγματα για τον υπολογιστή στον οποίο τρέχουμε το πρόγραμμα, όμως οι ίδιοι κώδικες μπορούν να χρησιμοποιηθούν για τα αρχικά δείγματα σε κάποιο μηχάνημα που οι προδιαγραφές του να του επιτρέπουν να ανταπεξέλθει καλύτερα.

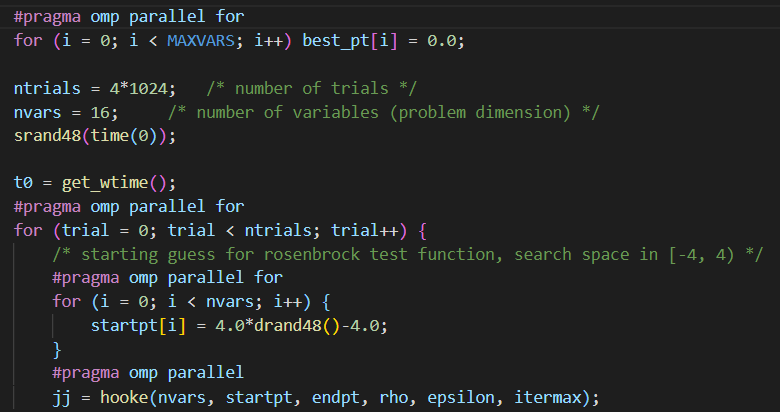


Πρώτη μας κίνηση είναι να τρέξουμε τον αρχικό κώδικα που μας δόθηκε, ώστε να λάβουμε τον αρχικό μας χρόνο (στο μη παράλληλο πρόγραμμα) και λαμβάνουμε τα παρακάτω αποτελέσματα σε χρόνο 43.267 δευτερόλεπτα:

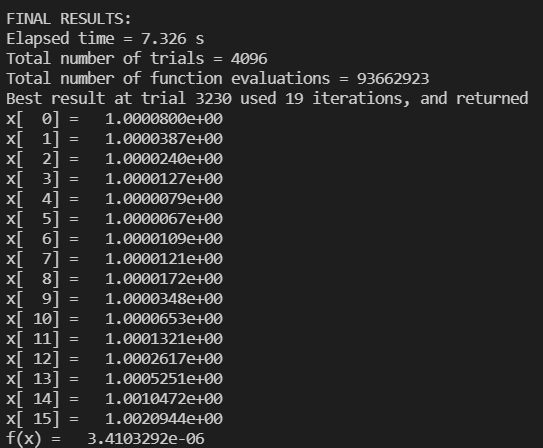


**OpenMP:**

Προχωράμε παρακάτω και ξεκινάμε να κάνουμε την υλοποίηση του κώδικα με τη βοήθεια του εργαλείου του OpenMP, τροποποιώντας τον κώδικα στο κομμάτι της συνάρτησης main:

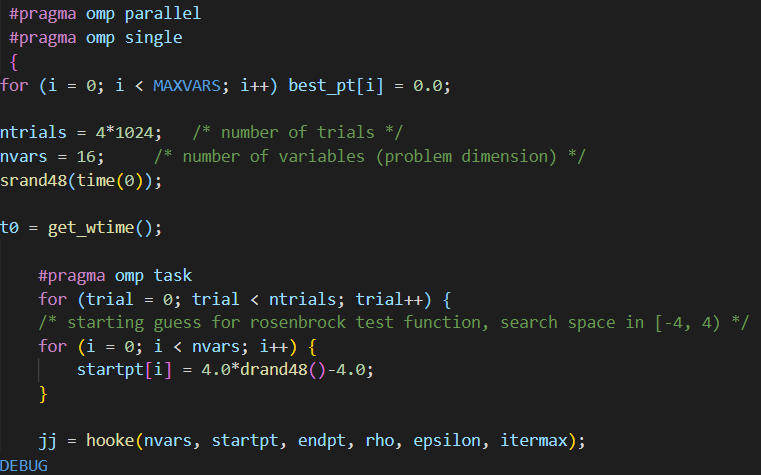


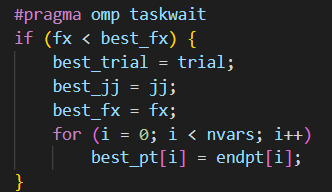
και παρατηρούμε σημαντική διαφορά στο χρόνο που χρειάστηκε για να λάβουμε τα αποτελέσματα, τα οποία φαίνονται παρακάτω:



**OpenMP\_Tasks:**

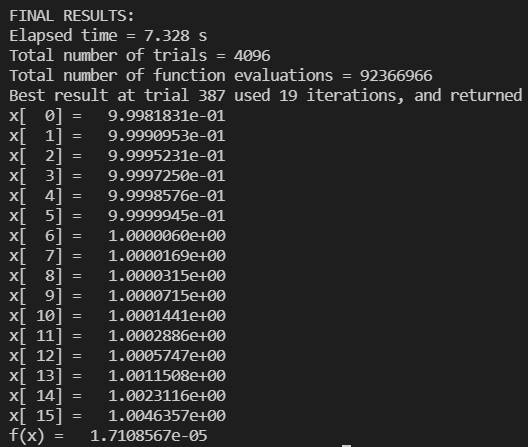
Τροποποιούμε ξανά τον κώδικά μας, αυτή τη φορά, προσπαθώντας να κάνουμε υλοποίηση με το OpenMp\_tasks





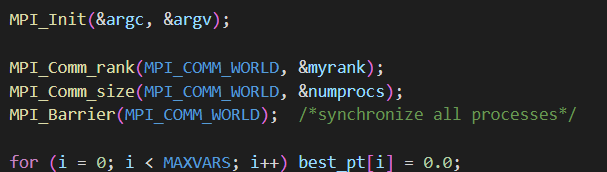
Προσέχουμε διαδικασίες που χρειάζονται προτεραιότητα να δρομολογούνται πριν από κάποιες που χρειάζονται τα προηγούμενα αποτελέσματα για να τρέξουν και γι αυτό το λόγο το κομμάτι του κώδικα που είναι υπεύθυνο για την εύρεση των ελαχίστων τιμών το «βάζουμε σε αναμονή» μέχρι τα πρώτα tasks να ολοκληρωθούν.

Παρατηρούμε ότι χρονικά δεν έχουμε μεγάλη απόκλιση με την προηγούμενη διαδικασία που ακολουθήσαμε για το OpenMp πρότυπο, όπως βλέπουμε και παρακάτω:

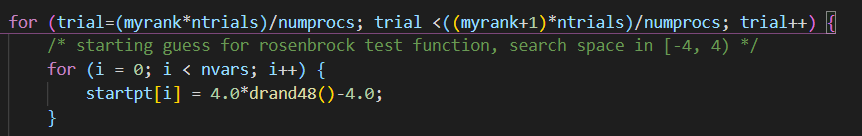


**MPI:**

Τροποποιούμε ξανά τον κώδικα στο κομμάτι της συνάρτησης main για άλλη μια φορά χρησιμοποιώντας το εργαλείο του MPI και συγκεκριμένα δουλεύουμε με το MPI\_REDUCE όπως ζητήθηκε με την προέκταση του MPI\_MIN καθώς χρειαζόμαστε να βρεθεί το ελάχιστο της συνάρτησης.

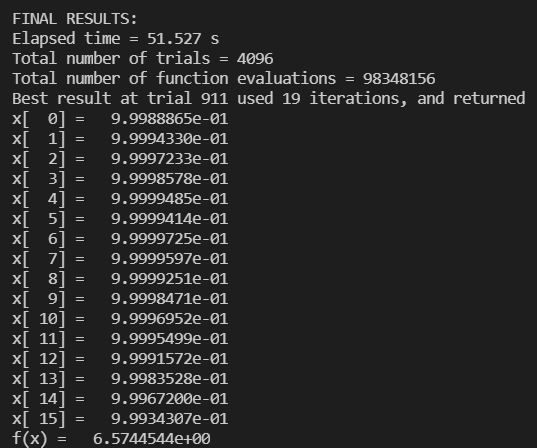


Δηλώνουμε τις μεταβλητές που χρειαζόμαστε για το ποιες διεργασίες χρησιμοποιούνται αλλά και το πόσες.

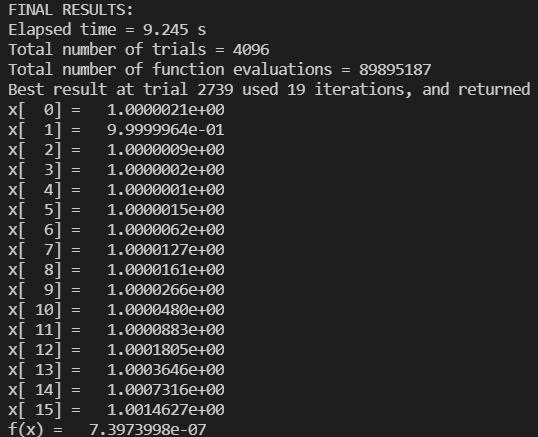




Θέλουμε τα αποτελέσματα για τα τοπικά ελάχιστα να προστίθενται και να τοποθετούνται κάθε φορά σε μια μεταβλητή sum την οποία στη συνέχεια «στέλνουμε» στη μεταβλητή best\_fx κι από κει να λαμβάνουμε το μικρότερο τοπικό ελάχιστο.

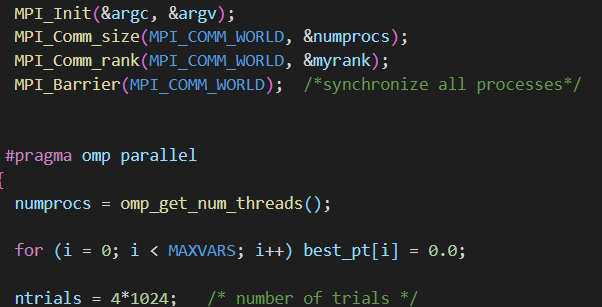


Παρατηρούμε ότι το χρονικό αποτέλεσμα είναι χειρότερο του αρχικού, κι όπως συνειδητοποιήσαμε στην πορεία συζητώντας το ζήτημα και με άλλους συμφοιτητές μας διαπιστώσαμε κατά πάσα πιθανότητα υπάρχει θέμα hardware, έτσι αποφασίσαμε να κάνουμε κι άλλες δοκιμές και παρατηρήσαμε ότι χωρίς τη χρήση του MPI\_Reduce λαμβάνουμε καλύτερα αποτελέσματα στο συγκεκριμένο υπολογιστή:

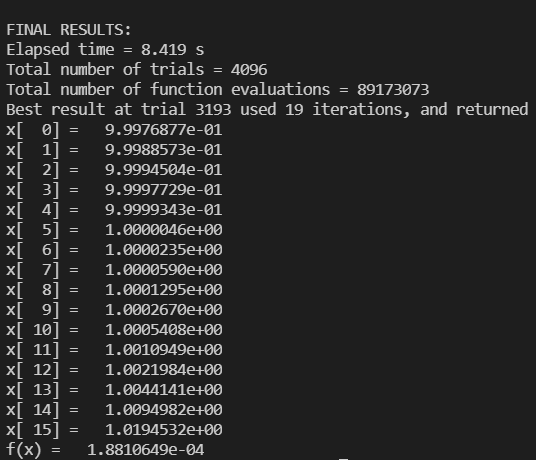


**Hybrid:**

Για μία ακόμη φορά κάνουμε τις απαραίτητες αλλαγές στον κώδικα κυρίως στο κομμάτι της main συνάρτησης.



Τα χρονικά αποτελέσματα που λαμβάνουμε παρατηρούμε ότι είναι τα καλύτερα κι απ’ τις δύο μας απόπειρες στο MPI, που ίσως όπως προείπαμε να οφείλεται στο υλικό:



**Συμπεράσματα:**

Συγκρίνοντας τις παράλληλες υλοποιήσεις είναι ευδιάκριτο ότι την καλύτερη χρονική απόδοση έχει η πρώτη, δηλαδή το OpenMP μοντέλο. Σημαντικό είναι επίσης να σημειώσουμε το πόσο καλύτερα αποτελέσματα μας δίνει ο παράλληλος κώδικας σε σχέση με την αρχική μας περίπτωση, όπου χρειάζεται να περιμένουμε σχεδόν ένα λεπτό για λάβουμε αποτελέσματα, ενώ η παραλληλοποίηση ρίχνει αυτόν τον αριθμό στα 9 περίπου δευτερόλεπτα στη χειρότερη περίπτωση (εξαιρείται η πρώτη υλοποίηση του MPI\_Reduce στην προκειμένη περίπτωση) , γεγονός που επιβεβαιώνει και στην πράξη όσα μαθαίνουμε τόσο καιρό στη θεωρία.