ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ Η/Υ
Αρχιτεκτονική Παράλληλων και Κατανεμημένων Υπολογιστών (HPY418)
Εαρινό Εξάμηνο 2018

Άσκηση 1: Χρήση OpenMP και pthreads

Ομάδα Εργασίας LAB41835594:
Νικόλαος Κυπαρισσάς, 2012030112
Σοφία Μαραγκού, 20120300078

1. Εισαγωγή

Ο σκοπός της 1ης άσκησης ήταν να επιταχύνουμε τον υπολογισμό αποστάσεων Hamming μεταξύ δύο συνόλων συμβολοσειρών με χρήση του OpenMP (Open Multi-Processing) API και ενός υποσυνόλου των συναρτήσεων pthreads (POSIX threads standard).

2. Ο Σειριακός Αλγόριθμος και ο Παραλληλισμός

Ο αλγόριθμος συγκρίνει συμβολοσειρές μήκους l μεταξύ των δύο συνόλων A και B μεγέθους m και n αντίστοιχα. Ο σειριακός αλγόριθμος συγκρίνει ακολουθιακά σε κάθε βήμα του έναν προς έναν τους χαρακτήρες μίας συμβολοσειράς του πίνακα A με τους χαρακτήρες μίας συμβολοσειράς του πίνακα B.

Προτού προχωρήσει σε επόμενες συμβολοσειρές, αποθηκεύει τον αριθμό των διαφορών που υπολόγισε σε έναν τρίτο πίνακα *Hamming Table*, π.χ. Hamming Table (3, 7) = αριθμός διαφορών μεταξύ των συμβολοσειρών A(3) και B(7).

2.1 Τρόποι Παραλληλισμού

Υπάρχουν τουλάχιστον 3 τρόποι παραλληλισμού του παραπάνω αλγορίθμου:

1. Ένα task μπορεί να αναλαμβάνει τη σύγκριση μόνο ενός μέρους του ζευγαριού των συμβολοσειρών.
2. Ένα task μπορεί να αναλαμβάνει τη σύγκριση μιας συμβολοσειράς του συνόλου A με μια συμβολοσειρά του συνόλου B. Ο *Hamming Table* αποτελεί μια “σκακιέρα” με τόσα χρώματα όσα και ο συνολικός αριθμός των threads.
3. Ένα task μπορεί να αναλαμβάνει τη σύγκριση μιας συμβολοσειράς του συνόλου A με όλες τις συμβολοσειρές του συνόλου B.

Και οι τρεις τρόποι παραλληλισμού που προαναφέραμε μοιράζουν τον φόρτο εργασίας μεταξύ των threads ανάλογα με το *thread ID* του κάθε thread.

Το πλεονέκτημα του δεύτερου και του τρίτου τρόπου είναι ό,τι δεν υπάρχει κάποιου είδους εξάρτηση μεταξύ των αποτελεσμάτων τους, αφού κάθε thread παράγει το αποτέλεσμα διαφορετικής σύγκρισης, δηλαδή για διαφορετικό κελί του *Hamming Table*. Αντιθέτως, ο πρώτος τρόπος απαιτεί τη χρήση locks αφού κάθε thread υπολογίζει μέρους του ίδιου αθροίσματος με τα υπόλοιπα threads.

2.2 Επαλήθευση Αποτελεσμάτων

Οι συμβολοσειρές που χρησιμοποιήσαμε αποτελούνται από ψευδοτυχαίους συνδυασμούς των συμβόλων "0" και "1". Ο χρήστης μπορεί να αρχικοποιήσει τη γεννήτρια τυχαίων αριθμών με όποια τιμή θέλει.

Η παραλληλοποίηση με βάση το *thread ID* απαιτεί προσοχή, ιδίως σε περιπτώσεις που ο φόρτος δεν μπορεί να μοιραστεί ισόποσα.

Για να είμαστε σίγουροι ότι οι επεμβάσεις μας στο σειριακό αλγόριθμο με σκοπό την παραλληλοποίηση δεν επηρεάζουν την εγκυρότητα του αποτελέσματός του, συγκρίναμε το άθροισμα όλων των τιμών που υπήρχαν στο *Hamming Table* μετά από κάθε παράλληλη εκτέλεση με το αντίστοιχο του σειριακού αλγορίθμου.

3. OpenMP

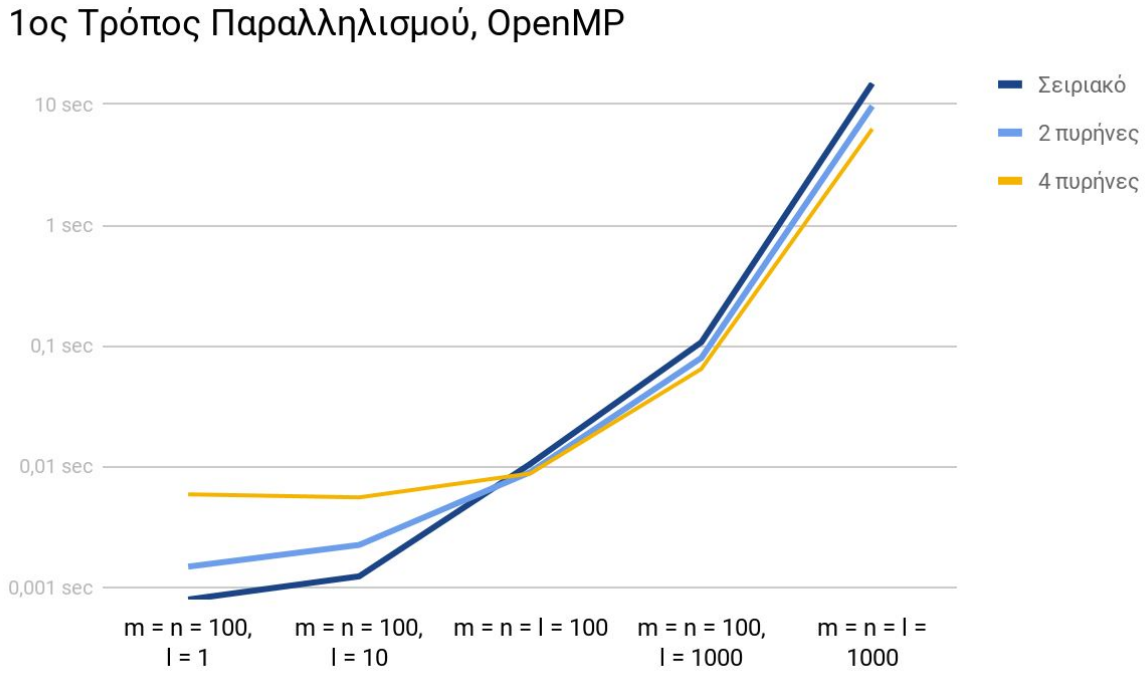
Το OpenMP API είναι ένας εύχρηστος τρόπος για να παραλληλοποιηθεί ένα πρόγραμμα, αφού συνήθως απαιτεί ελάχιστες αλλαγές στον αρχικό, σειριακό κώδικα.

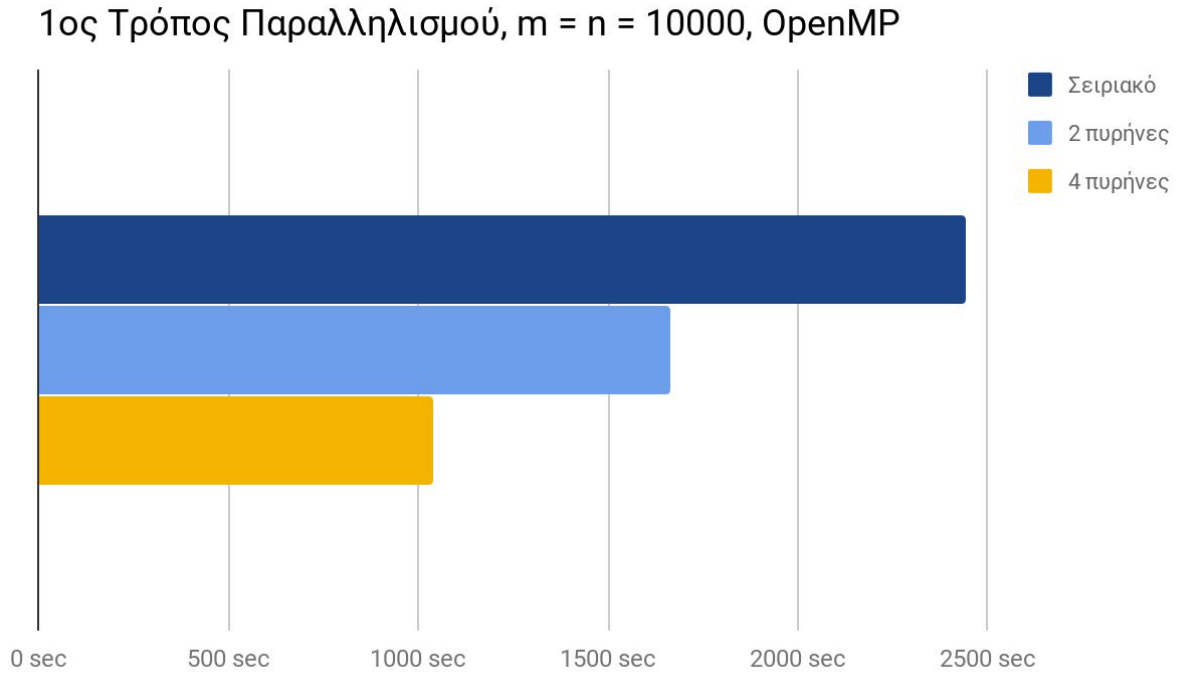
3.1 Αποτελέσματα και Συμπεράσματα

Ακολουθούν τα αποτελέσματα των εκτελέσεων των OpenMP υλοποιήσεων. Τα αποτελέσματα είναι σε δευτερόλεπτα εκτέλεσης, και παρουσιάζονται σε λογαριθμική κλίμακα.

Παρουσιάζονται επίσης σε κανονική κλίματα τα αποτελέσματα των εκτελέσεων για $m = n = 10000$ και $l = 1000$, τον μεγαλύτερο συνδυασμό που μας ζητήθηκε από την εκφώνηση.

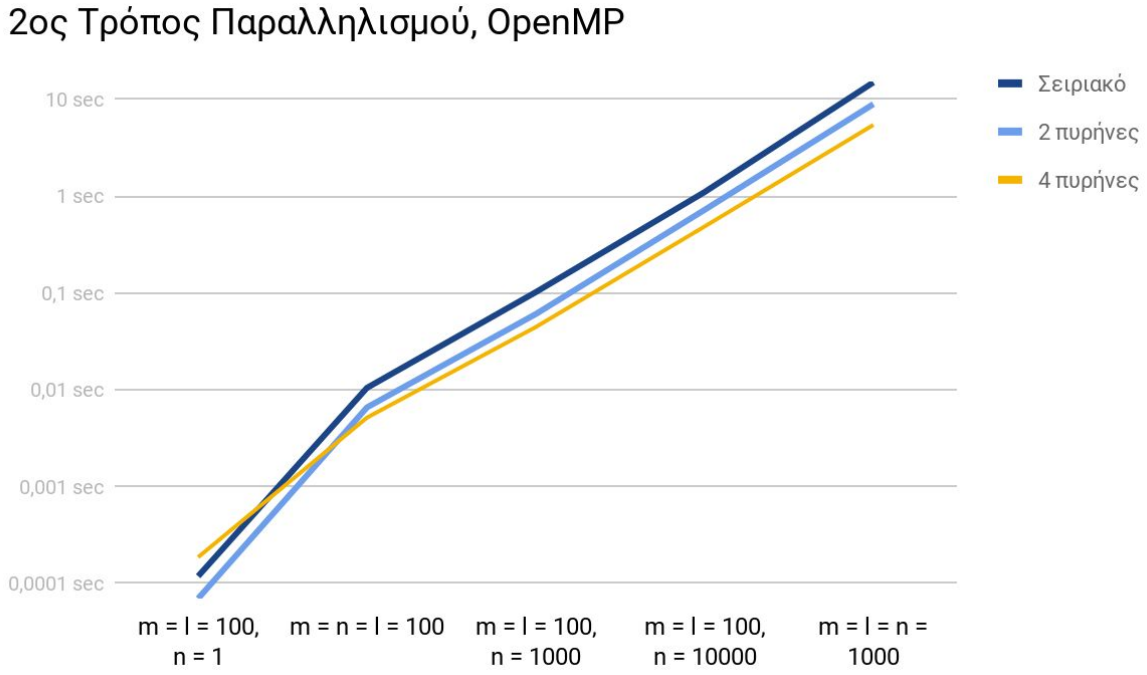
Τέλος το speedup που λαμβάνουμε σε κάθε περίπτωση καθώς και τα συμπεράσματα που εξάγουμε από τα αποτελέσματα.

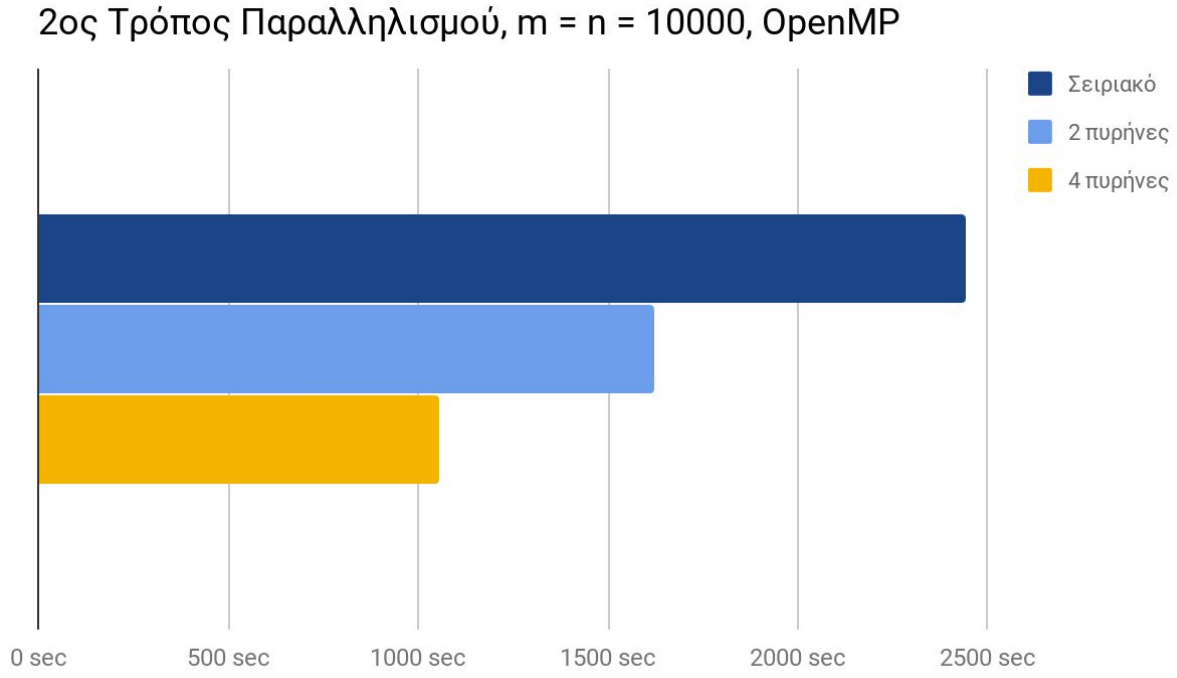




Speedup με 2 πυρήνες = 1.47x Speedup με 4 πυρήνες = 2.35x

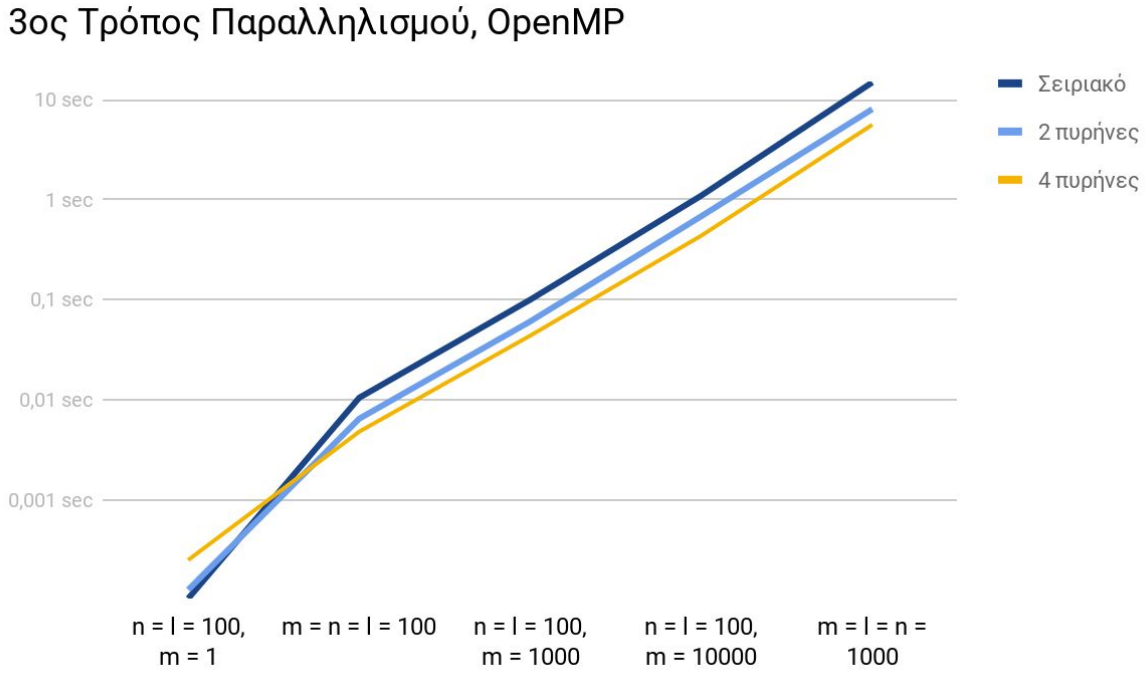
Παρατηρούμε ότι για πολύ μικρές συμβολοσειρές οι παράλληλες εκτελέσεις αποδίδουν χειρότερα από τον σειριακό αλγόριθμο. Αυτό οφείλεται στην ελάχιστη έως μηδαμινή (l=1) παραλληλία του προβλήματος, και έχει ως συνέπεια να κοστίζει τελικά η ίδια η εγκαθίδρυση της παραλληλίας στο σύστημα.

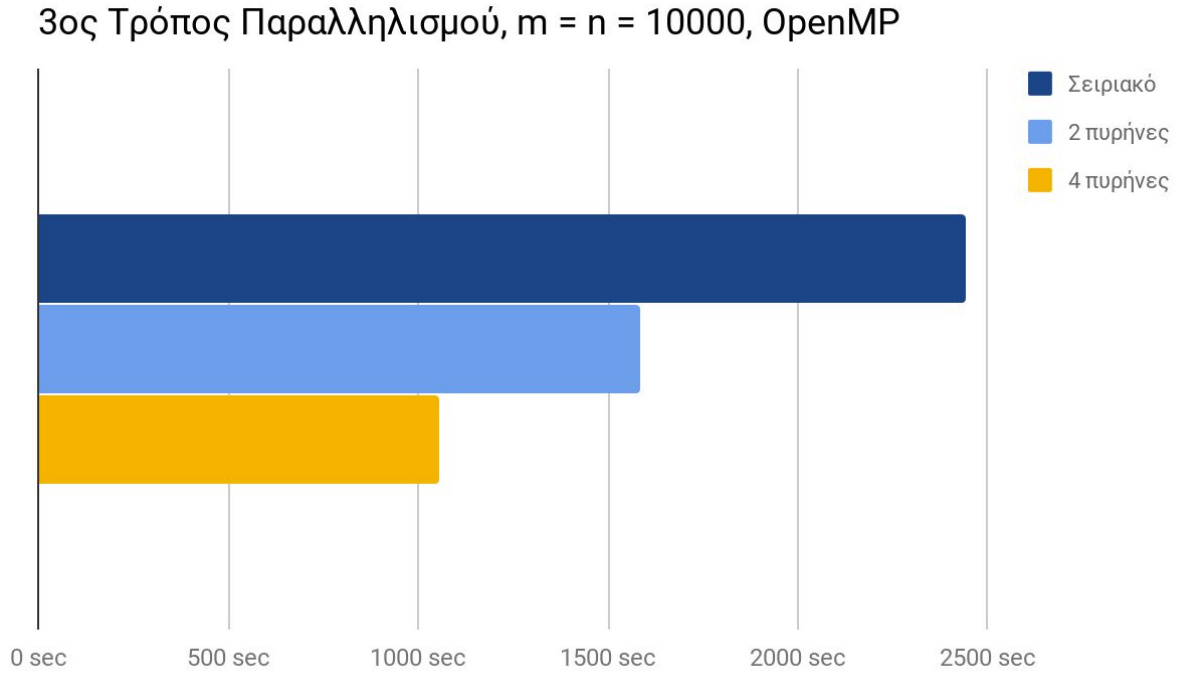




Speedup με 2 πυρήνες = 1.51x Speedup με 4 πυρήνες = 2.31x

Παρατηρούμε ότι για πολύ λίγες συμβολοσειρές η παράλληλη εκτέλεση αποδίδει χειρότερα για 4 πυρήνες από ότι για 2. Πιθανόν να κοστίζει πιο πολύ στον αλγόριθμο να "χρωματίσει" με 4 χρώματα μία μονοδιάστατη σκακιέρα από τη δουλειά που απαιτείται για τον υπολογισμό των αποστάσεων Hamming.





Speedup με 2 πυρήνες = 1.54x Speedup με 4 πυρήνες = 2.31x

Παρατηρούμε ότι για πολύ λίγες συμβολοσειρές η παράλληλη εκτέλεση αποδίδει χειρότερα από το σειριακό πρόγραμμα. Για $m = 1$ δεν υπάρχει κάτι να μοιραστούν τα threads αφού το σύνολο A έχει μόνο μια συμβολοσειρά. Η εγκαθίδρυση της παραλληλίας στο σύστημα επιβαρύνει την εκτέλεση ενώ στην ουσία το πρόγραμμα υπολογίζει σειριακά τις διαφορές και δεν εκμεταλλεύεται παραλληλία.

4. pthreads

Οι συναρτήσεις pthreads (POSIX threads standard) αποτελούν παλαιότερη μέθοδο παραλληλοποίησης προγραμμάτων. Η χρήση τους είναι πιο περίπλοκη από το OpenMP, καθώς απαιτούν την κλήση συνάρτησης με σκοπό την παράλληλη εκτέλεσή της.

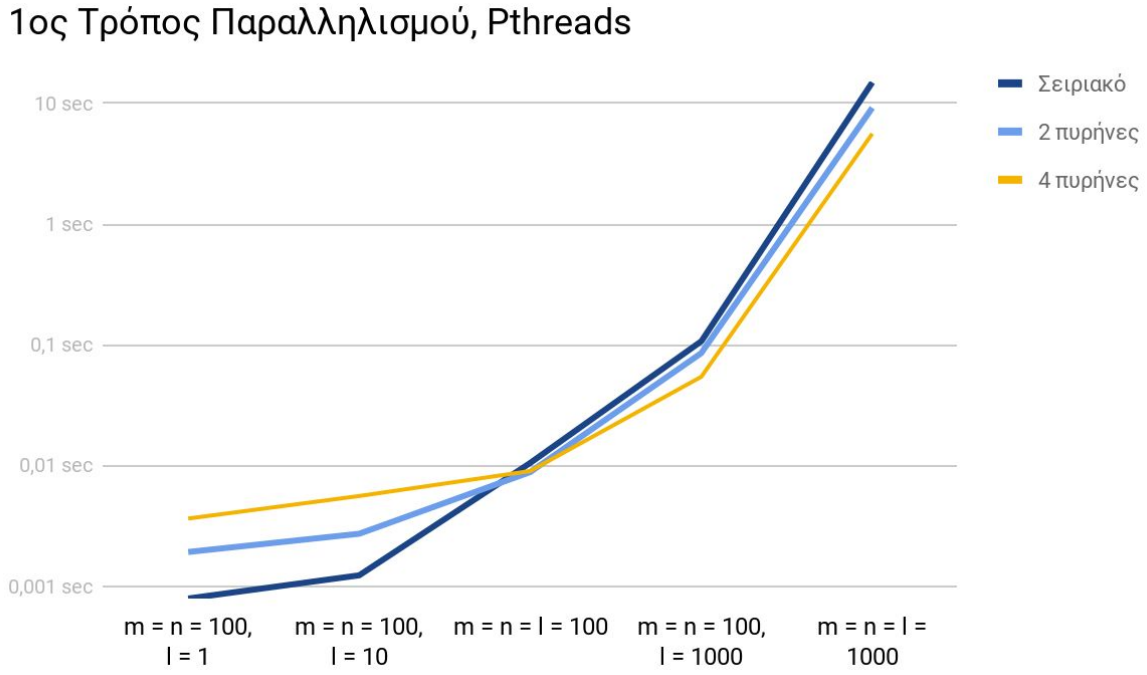
Για το λόγο αυτό δημιουργήσαμε μια δομή με σκοπό τη μεταφορά όλων των απαραίτητων για την εκτέλεση ορισμάτων. Επίσης, καθώς αντιμετωπίσαμε πρόβλημα με την απόκτηση του εσωτερικού *thread ID*, δημιουργήσαμε μια static μεταβλητή που τη χρησιμοποιεί το κάθε thread για να "δημιουργήσει" το ID του, αφού πρώτα την κλειδώσει.

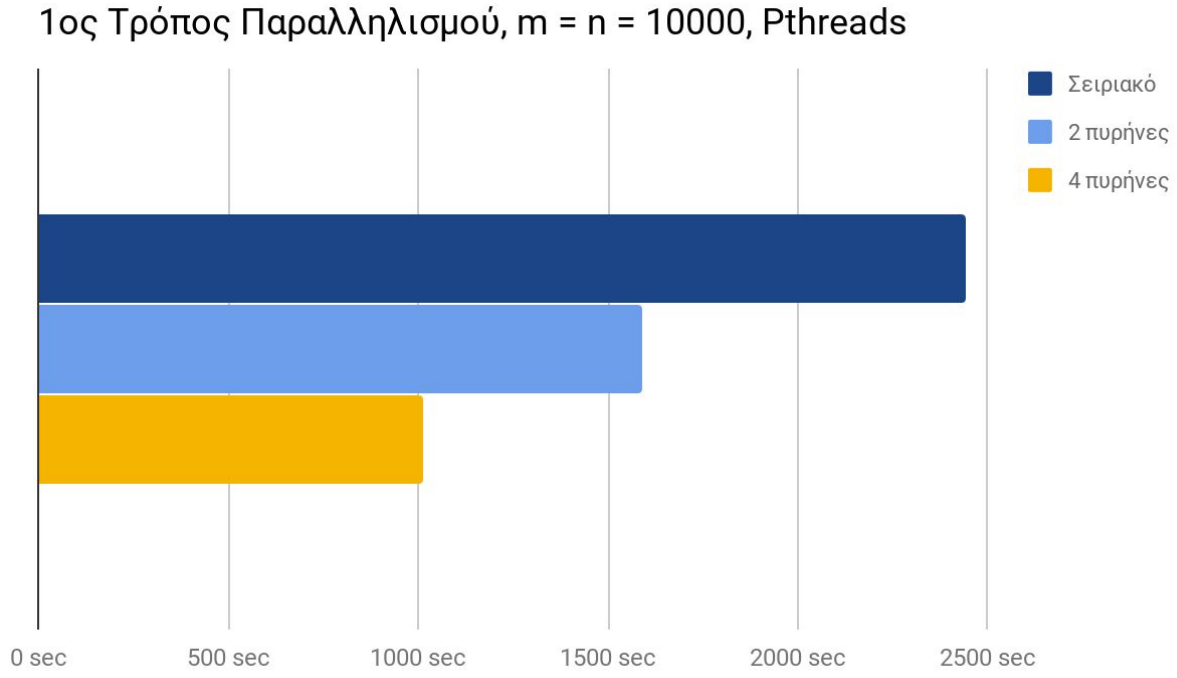
4.1 Αποτελέσματα και Συμπεράσματα

Ακολουθούν τα αποτελέσματα των εκτελέσεων των υλοποιήσεων με pthreads. Τα αποτελέσματα είναι σε δευτερόλεπτα εκτέλεσης, και παρουσιάζονται σε λογαριθμική κλίμακα.

Παρουσιάζονται επίσης σε κανονική κλίματα τα αποτελέσματα των εκτελέσεων για $m = n = 10000$ και $l = 1000$, τον μεγαλύτερο συνδυασμό που μας ζητήθηκε από την εκφώνηση.

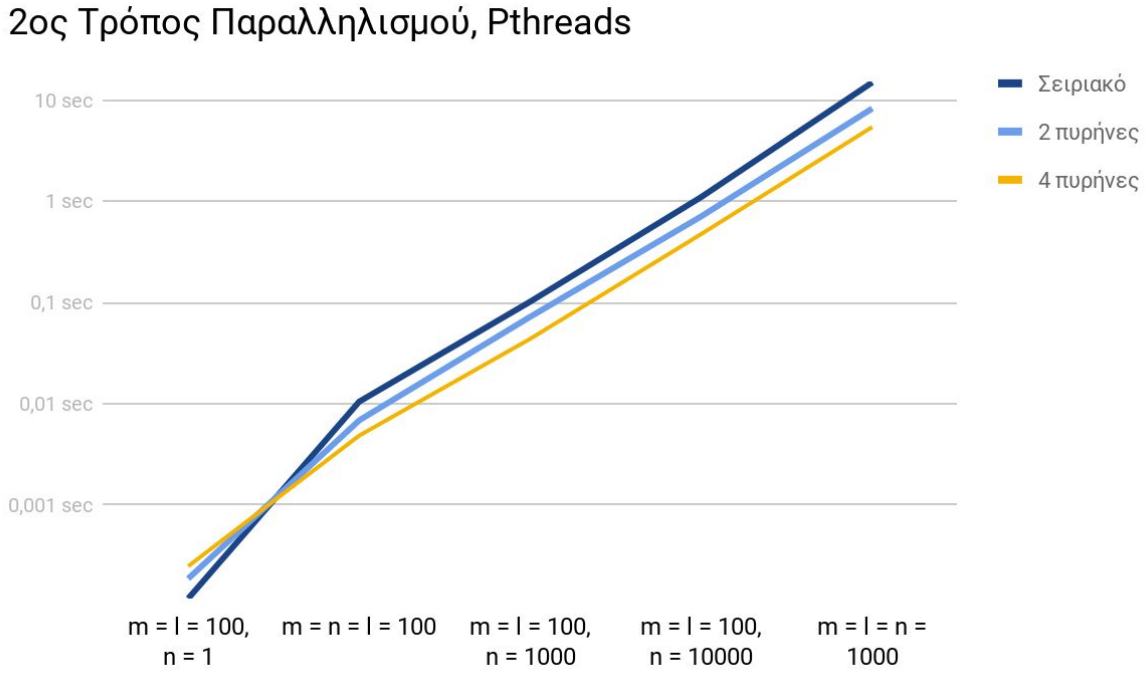
Τέλος το speedup που λαμβάνουμε σε κάθε περίπτωση καθώς και τα συμπεράσματα που εξάγουμε από τα αποτελέσματα.

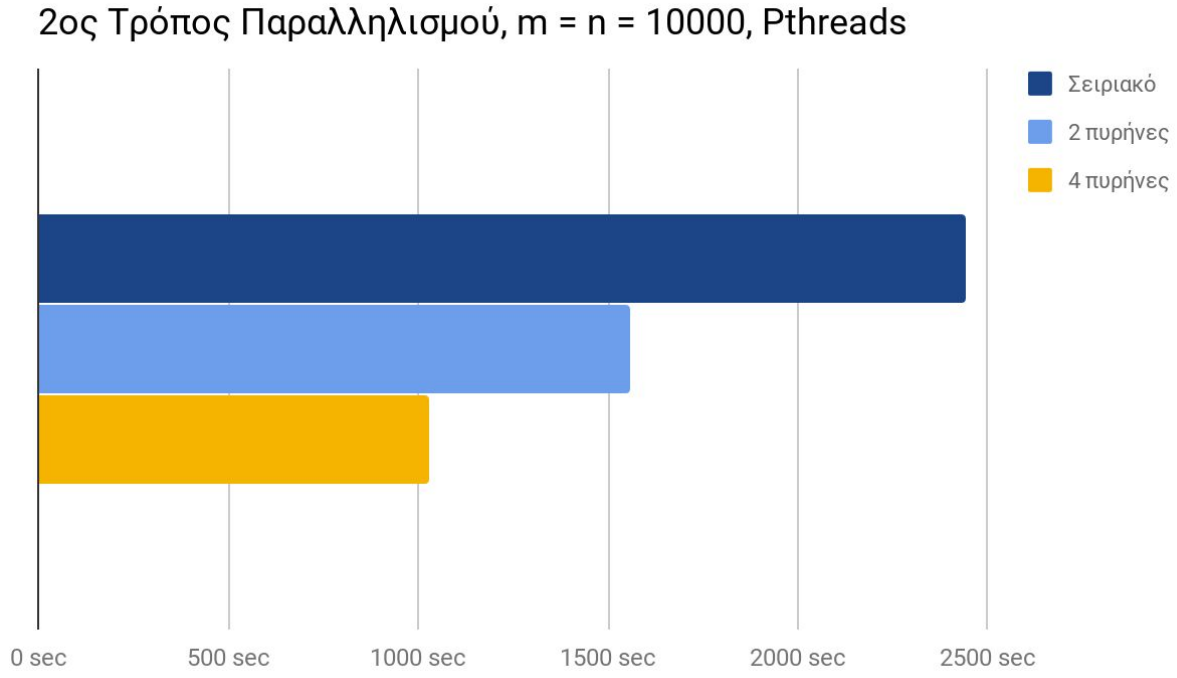




Speedup με 2 πυρήνες = 1.53x Speedup με 4 πυρήνες = 2.41x

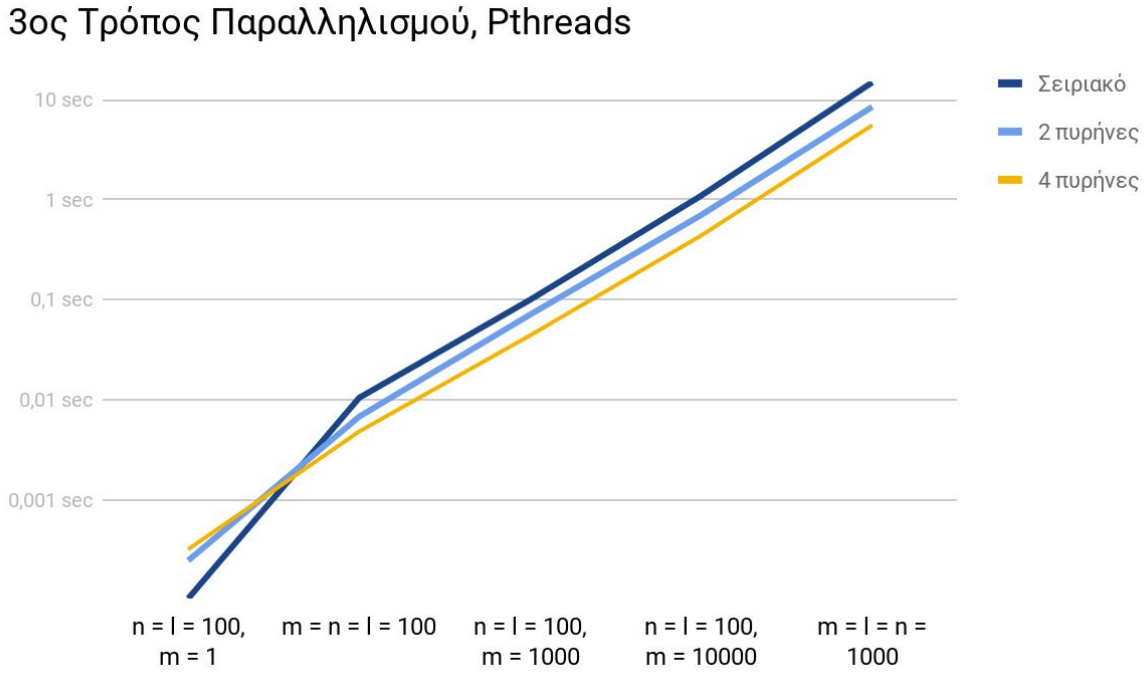
Παρατηρούμε ότι, όπως και με το OpenMP, για πολύ μικρές συμβολοσειρές οι παράλληλες εκτελέσεις αποδίδουν χειρότερα από τον σειριακό αλγόριθμο. Αυτό οφείλεται στην ελάχιστη έως μηδαμινή (l=1) παραλληλία του προβλήματος, και έχει ως συνέπεια να κοστίζει τελικά η ίδια η εγκαθίδρυση της παραλληλίας στο σύστημα.

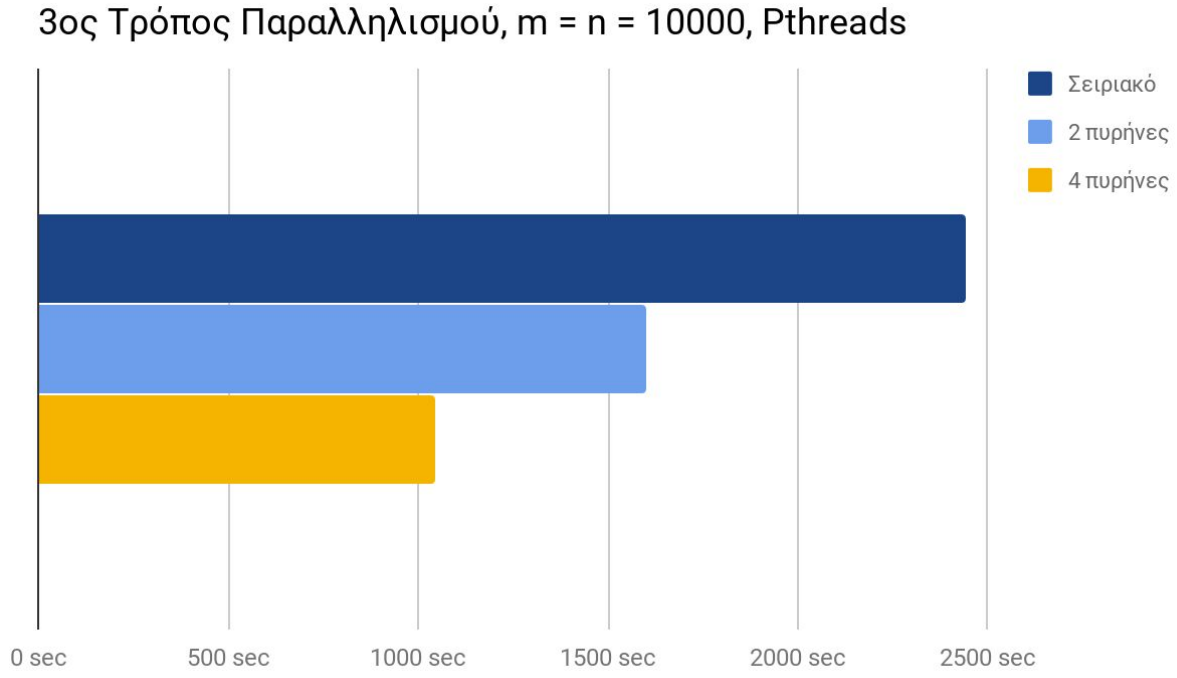




Speedup με 2 πυρήνες = 1.57x Speedup με 4 πυρήνες = 2.37x

Παρατηρούμε ότι όπως και με το OpenMP, για πολύ λίγες συμβολοσειρές οι παράλληλες εκτελέσεις αποδίδουν χειρότερα από το σειριακό πρόγραμμα. Πιθανόν να κοστίζει πιο πολύ στον αλγόριθμο να “χρωματίσει” μία μονοδιάστατη σκακιέρα από τη δουλειά που απαιτείται για τον υπολογισμό των αποστάσεων Hamming.





Speedup με 2 πυρήνες = 1.52x Speedup με 4 πυρήνες = 2.34x

Παρατηρούμε ότι και εδώ για πολύ λίγες συμβολοσειρές η παράλληλη εκτέλεση αποδίδει χειρότερα από το σειριακό πρόγραμμα. Για $m = 1$ δεν υπάρχει κάτι να μοιραστούν τα threads αφού το σύνολο A έχει μόνο μια συμβολοσειρά. Η εγκαθίδρυση της παραλληλίας στο σύστημα επιβαρύνει την εκτέλεση ενώ στην ουσία το πρόγραμμα υπολογίζει σειριακά τις διαφορές και δεν εκμεταλλεύεται παραλληλία.

5. Συμπεράσματα

Τα αποτελέσματα που παίρνουμε από τα δύο είδη παράλληλου προγραμματισμού είναι πάρα πολύ κοντά, με τα pthreads να αποδίδουν ελάχιστα καλύτερα από το OpenMP σε μερικές περιπτώσεις, αλλά ταυτόχρονα απαιτείται περισσότερος χρόνος για να προγραμματιστούν σωστά.

6. Παράρτημα

Πληροφορίες συστήματος μετρήσεων:
CPU: Intel Core i5 2410M (2.3 GHz, 2 πυρήνες ~ 4 threads, 3MB L3 Cache)
RAM: 8 GB DDR3, 1333 MHz
OS: Bash on Ubuntu on Windows 10 (16.04 xenial)

Παραδοχές:
Όταν η πολυπλοκότητα του προβλήματος είναι μικρή (αυθαίρετα ορισμένη στα 1.000.000 κελιά) τα αποτελέσματα της ίδιας εκτέλεσης μπορεί να έχουν μεγάλη απόκλιση μεταξύ τους. Για τον λόγο αυτόν παίρνουμε τον μέσο όρο 100 εκτελέσεων ως αποτέλεσμα αυτής της εκτέλεσης.

Για να εκτελεστεί το πρόγραμμα, εισάγετε την παρακάτω εντολή:
./script.sh m n l cores seed
Η σειριακή υλοποίηση αγνοεί την παράμετρο του αριθμού πυρήνων.