Παράλληλη Επεξεργασία

Αθηνά Φουσέκη 1059623
Αικατερίνη Δέρβου 1054185
Αλέξανδρος Ξιάρχος 1059619

Εισαγωγή

Στόχος της εργασίας αυτής είναι η παραλληλοποίηση του λογισμικού ολικής βελτιστοποίησης με τη χρήση των OpenMP, OpenMP tasks, MPI και MPI σε συνδυασμό με OpenMP. Το υπολογιστικό σύστημα που χρησιμοποιήθηκε για τη εκτέλεση των παραπάνω ήταν ένα Virtual Machine, και συγκεκριμένα το Virtual Box της Oracle, με τα παρακάτω χαρακτηριστικά:
Λειτουργικό: Ubuntu, έκδοση 20.04
CPU: AMD Ryzen 3
N of cores: 2
RAM: 3 GB
Δευτερεύων δίσκος: SSD

Αναλυτική τεκμηρίωση παράλληλων υλοποιήσεων και τυχόν βελτιστοποιήσεων

OpenMP:

Η παραλληλοποίηση που επιλέξαμε να κάνουμε με το OpenMP είναι η βασική for του προγράμματος. Η τακτική που ακολουθείται για την επιλογή των σημείων παραλληλοποίησης στο OpenMP, είναι ότι επιλέγονται να παραλληλοποιηθούν διαδικασίες οι οποίες σε σειριακό κώδικα παίρνουν πάρα πολύ χρόνο, και όχι μικρές διαδικασίες. Έτσι, παραλληλοποιήθηκε η for που έχει ntrials (128*1024) επαναλήψεις. Πρώτα ορίστηκε η περιοχή παραλληλοποίησης με μια #pragma omp parallel με την ανάθεση εμβέλειας των μεταβλητών, και μέσα σε αυτή χρησιμοποιήθηκε μια #pragma omp for για την παραπάνω for.

OpenMP Tasks:

Για τον ορισμό του πεδίου το οποίο θα παραλληλοποιηθεί χρησιμοποιούμε τον κώδικα #pragma omp parallel. Στη συγκεκριμένη περίπτωση θα παραλληλοποιήσουμε τον βασικό κώδικα της main, στον οποίον ξεχωρίζουμε δύο σημεία: την αρχικοποίηση του πίνακα *startpt*, και την συνάρτηση *hook* που καλείται. Καθορίζουμε και τις δύο περιπτώσεις ως διαφορετικά tasks μέσω του κώδικα #pragma omp task, το οποίο έχει ως αποτέλεσμα την παράλληλη εκτέλεσή τους σε μέγιστο αριθμό threads, συντελώντας στην ζητούμενη χρονοβελτίωση. Τέλος ο κώδικας #pragma omp single επιτυγχάνει την σωστή χρονική σειρά εκτέλεση των εντολών παρά την παραλληλοποίηση.

MPI:

Για την παραλληλοποίηση με MPI χρησιμοποιήθηκε πάλι ο βασικός κώδικας της main. Αρχικά κλήθηκε η MPI_init για την έναρξη της παράλληλης περιοχής. Στη συνέχεια ορίσαμε τα ranks και size και χωρίσαμε τις ntrials επαναλήψεις στα ranks ισότιμα. Στο τέλος της δομής επανάληψης κλήθηκε η MPI_Finalize και επιλέξαμε να τυπωθούν τα αποτελέσματα του rank=0.

MPI+OpenMP:

Για αυτή την υλοποίηση συνδυάστηκαν οι κώδικες των OpenMP και MPI.

Αποτελέσματα

Seq: Elapsed time = 121.276 s

Speedup=$T_{sequential}/T_{parallel}$

OpenMP

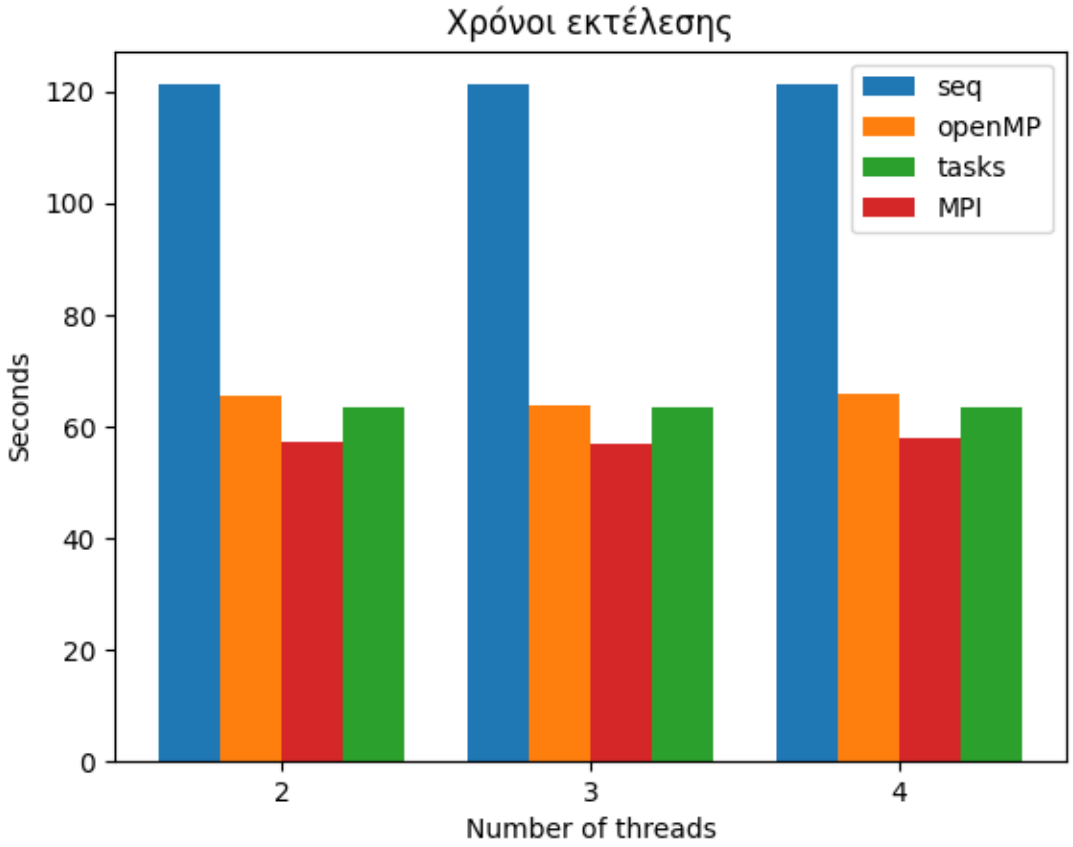
num_of_threads	2	3	4
Time	65.446 s	63.790 ss	65.749 s
Speedup	1.853	1.901	1.845
Τελικό Ελάχιστο	6.5408031e-08	1.5663865e-07	4.9250573e-08
Συνολικός Αρ. Κλήσεων	1616523648	1126436186	868416523

MPI

	2	3	4
Time	57.375 s	56.748 s	57.870 s
Speedup	2.114	2.137	2.096
Τελικό Ελάχιστο	1.6231246e-06	1.7702969e-07	7.1015495e-07
Συνολικός Αρ. Κλήσεων	1560413782	1034895284	788165621

OpenMP tasks

Time	63.294 s
Speedup	1.916
Τελικό Ελάχιστο	8.5414835e-09
Συνολικός Αρ. Κλήσεων	1598754845



OpenMP+MPI

omp	mpi	time	Speedup	Τελικό min	Total #κλήσεων
2	2	62.602 s	1.937	9.0239375e-08	806885316
2	3	69.405 s	1.747	1.7464885e-07	556536702
2	4	63.999 s	1.895	4.8957000e-08	417692781
3	2	63.676 s	1.905	1.8070253e-08	546679283
3	3	68.404 s	1.773	4.3536465e-07	380714864
3	4	66.742 s	1.817	6.1909449e-07	280427509
4	2	66.503 s	1.824	1.8070253e-08	417834054
4	3	64.633 s	1.876	5.3862889e-07	296082904
4	4	64.549 s	1.879	4.8957000e-08	219202235

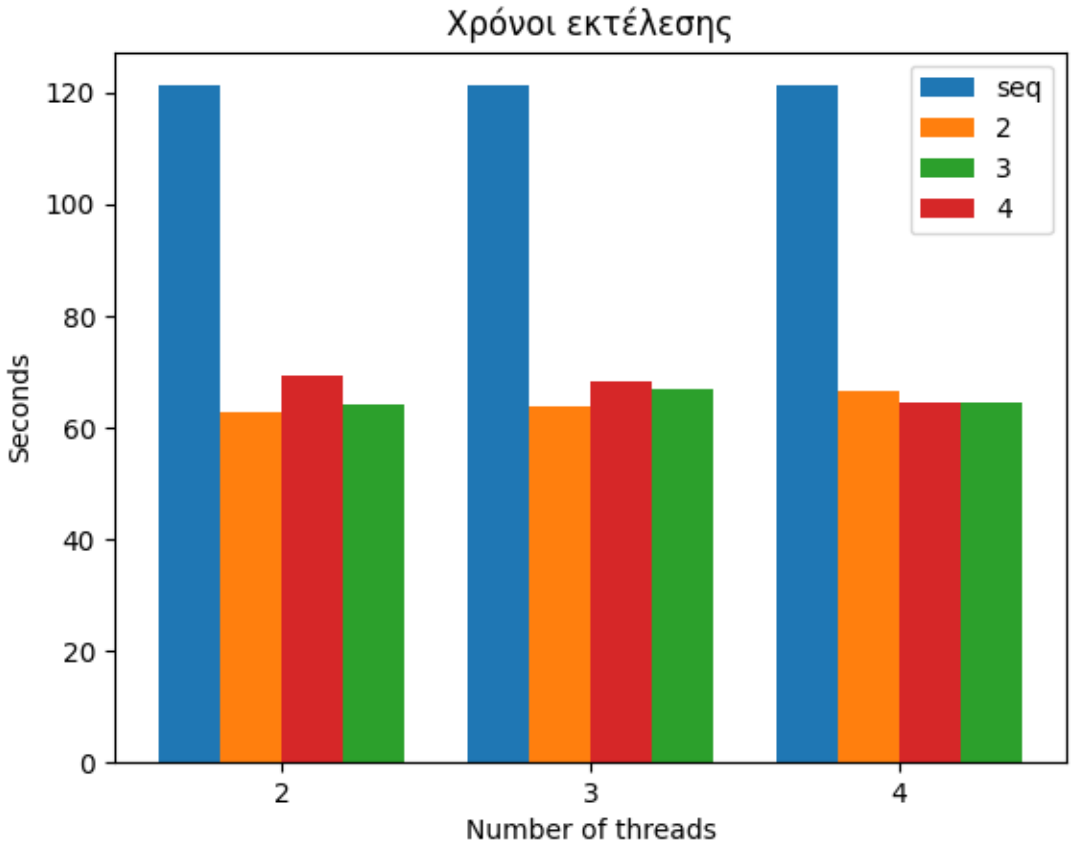


Figure 1: Με χρώμα εμφανίζονται οι διαφορετικές τιμές στο execution του MPI, ενώ στον άξονα εμφανίζονται τα νήματα του OpenMP.

Συμπεράσματα

Παρατηρώντας τα αποτελέσματα βλέπουμε ότι η παραλληλοποίηση με OpenMP είναι η λιγότερο αποδοτική μέθοδος. Όταν συνδυαστεί με το MPI είναι πιο αποδοτική, αλλά και πάλι οι άλλες μέθοδοι υπερτερούν. Η δέυτερη καλύτερη υλοποίηση είναι με OpenMP tasks και η καλύτερη είναι με τη χρήση των MPI.