

Παράλληλη Επεξεργασία

Function approximation with k-Nearest Neighbors

Φοιτητές

ΑΜ:

Χρήστος Μεραντζής	1070936
Γιώργος Μητρομάρας	1070907
Χριστίνα Κόκκαλη	1072617
Αγγελική Σταυροπούλου	1080423

Περιεχόμενα

Περιεχόμενα	1
I Εισαγωγή	2
II Υλοποίηση	3
III Πειραματική αξιολόγηση	10
IV MPI	10
V OpenMP	11
VI OpenMP Task Model	13
VII Σύνοψη	14

I Εισαγωγή

Ο κώδικας και οι μετρήσεις υλοποιήθηκαν σε εικονική μηχανή (VM) Debian Linux με κατανομή 4 πυρήνων από επεξεργαστή Intel(R) Core(TM) i5 8ης γενιάς. Το σύστημα υποδοχής χρησιμοποιεί αρχιτεκτονική Coffee Lake στα 2,80 GHz και μνήμη cache 9MB. Ωστόσο, καθώς πρόκειται για ένα εικονικοποιημένο περιβάλλον, ενδέχεται να υπάρχουν κάποιες διαφορές στη συμπεριφορά εκτέλεσης και τα χαρακτηριστικά απόδοσης σε σύγκριση με ένα σύστημα bare-metal.

Το VM εκτελεί το Debian Linux ως λειτουργικό σύστημα, παρέχοντας ένα αξιόπιστο και προσαρμόσιμο περιβάλλον για ανάπτυξη και εκτέλεση λογισμικού. Έχει εκχωρηθεί 12 GB μνήμης DDR3, η οποία θα πρέπει να παρέχει επαρκή χωρητικότητα μνήμης για τις περισσότερες εφαρμογές.

k-NN Regression Algorithm

Ο αλγόριθμος παλινδρόμησης k-NN είναι μια μέθοδος που χρησιμοποιείται για την επίλυση προβλημάτων ταξινόμησης και παλινδρόμησης. Λειτουργεί με training και query points. Για κάθε query point, ο αλγόριθμος εντοπίζει τα k πλησιέστερα σημεία από το σύνολο εκπαίδευσης. Οι αποστάσεις μεταξύ των σημείων υπολογίζονται συνήθως με μια μετρική απόστασης, όπως η ευκλείδεια απόσταση. Στη συνέχεια, επιλέγονται οι k πλησιέστεροι γείτονες με βάση αυτές τις αποστάσεις. Ο αλγόριθμος αναθέτει βάρη στους γείτονες, συνήθως αντιστρόφως ανάλογα με τις αποστάσεις τους, και υπολογίζει τον μέσο όρο των τιμών τους. Αυτός ο μέσος όρος γίνεται η προβλεπόμενη τιμή για το query point. Η διαδικασία επαναλαμβάνεται και η απόδοση του μοντέλου αξιολογείται με μετρικές όπως το μέσο τετραγωνικό σφάλμα, ο αντίστροφος σταθμισμένος μέσος όρος, το μέσο απόλυτο σφάλμα κ.α.

Ένα σημαντικό σημείο που πρέπει να ληφθεί υπόψη κατά τη χρήση του k-NN είναι η υπολογιστική του πολυπλοκότητα. Για κάθε σημείο ερώτησης, ο αλγόριθμος πρέπει να υπολογίσει την απόστασή του από όλα τα δεδομένα εκπαίδευσης. Όσο αυξάνεται το μέγεθος του συνόλου εκπαίδευσης, ο υπολογιστικός χρόνος αυξάνεται σημαντικά. Για να αντιμετωπιστεί αυτό, μπορούν να χρησιμοποιηθούν τεχνικές παραλληλοποίησης για την κατανομή του φόρτου εργασίας σε πολλούς επεξεργαστές ή νήματα, επιτρέποντας ταχύτερα αποτελέσματα.

Συνολικά, ο αλγόριθμος παλινδρόμησης k-NN είναι μια ευέλικτη προσέγγιση για προβλήματα παλινδρόμησης. Είναι καλός στο να ανιχνεύει τοπικά μοτίβα και να παρέχει συνεχείς προβλέψεις. Ωστόσο, η απόδοση και η κλιμάκωσή του μπορούν να βελτιωθούν μέσω της χρήσης μεθόδων παραλληλοποίησης, ειδικά για μεγάλα σύνολα δεδομένων.

Επισκόπηση παραλληλοποίησης

Για την παραλληλοποίηση του αλγόριθμου k-NN οι υπολογισμοί για διαφορετικά ερωτήματα μπορούν να εκτελεστούν παράλληλα. Αυτό επιτρέπει την ταυτόχρονη επεξεργασία του εξωτερικού βρόχου που είναι υπεύθυνος για το χειρισμό των ερωτημάτων και στις τρεις εκδόσεις της παραλληλοποίησης. Επιπλέον, χρησιμοποιούνται τεχνικές μείωσης για τη συγκέντρωση διαφόρων αποτελεσμάτων, όπως το άθροισμα των χρόνων υπολογισμού και των σφαλμάτων.

Στην προσέγγιση MPI (Message Passing Interface), μια διεργασία διαβάζει όλα τα απαραίτητα δεδομένα και τα μεταδίδει στις υπόλοιπες διεργασίες. Στη συνέχεια, κάθε διεργασία εκτελεί τους υπολογισμούς στα δεδομένα που της έχουν δοθεί, αξιοποιώντας τον παραλληλισμό που παρέχεται από πολλαπλές διεργασίες. Τα αποτελέσματα που λαμβάνονται από κάθε διεργασία συλλέγονται στη συνέχεια και συνδυάζονται για να ληφθεί το τελικό αποτέλεσμα. Από την άλλη πλευρά, στο OpenMP, χρησιμοποιούνται παράλληλες πρακτικές για την επίτευξη

παραλληλισμού μεταξύ πολλαπλών νημάτων. Καθορίζοντας σωστά private ή shared μεταβλητές, μπορούμε να ελέγχουμε την προσβασιμότητά τους και να διασφαλίσουμε τη σωστή παράλληλη εκτέλεση. Τα κατάλληλα τμήματα του κώδικα μπορούν να παραλληλιστούν χρησιμοποιώντας οδηγίες OpenMP για την κατανομή του φόρτου εργασίας μεταξύ διαφορετικών νημάτων. Συνοπτικά, τόσο το MPI όσο και το OpenMP μπορούν να χρησιμοποιηθούν για να παραλληλίσουν τον αλγόριθμο. Το MPI εστιάζει στον παραλληλισμό μεταξύ των διεργασιών, ενώ το OpenMP στοχεύει τον παραλληλισμό εντός ενός περιβάλλοντος κοινής μνήμης χρησιμοποιώντας πολλαπλά νήματα. Αυτές οι τεχνικές παραλληλοποίησης επιτρέπουν αποτελεσματικούς υπολογισμούς και μπορούν να βελτιώσουν σημαντικά την απόδοση του k-NN.

II Υλοποίηση

MPI

Για να επιτευχθεί η παραλληλοποίηση στην έκδοση MPI, χρησιμοποιήθηκε ένας κυκλικός διαχωρισμός των εργασιών. Με τον τρόπο αυτό εξασφαλίστηκε η εξισορρόπηση του φορτίου μεταξύ των διεργασιών, όπου σε κάθε διεργασία ανατέθηκε ένα υποσύνολο ερωτημάτων προς επεξεργασία. Η εξισορρόπηση φορτίου είναι ζωτικής σημασίας για την ομοιόμορφη κατανομή του υπολογιστικού φόρτου εργασίας στις διαθέσιμες διεργασίες και στη μεγιστοποίηση της αποδοτικότητας. Στην έκδοση MPI του παραλληλοποιημένου αλγορίθμου k-NN, τα δεδομένα ομαλοποιούνται πριν από τη διανομή τους στις διεργασίες. Αυτό το βήμα είναι απαραίτητο επειδή οι ρουτίνες MPI, όπως η ρουτίνα εκπομπής (MPI_Bcast), απαιτούν τα δεδομένα να είναι συνεχόμενα στη μνήμη. Ετοι, εξασφαλίζεται ότι οι πληροφορίες για κάθε ερωτημα αποθηκεύονται συνεχόμενα στη μνήμη, επιτρέποντας την αποτελεσματική διανομή δεδομένων και την επικοινωνία μεταξύ των διεργασιών MPI. Με αυτή τη διαδικασία μετατρέπεται η πολυδιάστατη δομή των δεδομένων σε μονοδιάστατο πίνακα ή διανυσματική αναπαράσταση, όπου κάθε στοιχείο αντιστοιχεί σε ένα συγκεκριμένο ερώτημα. Συνεπώς η αρχική διεργασία μοιράζεται αποτελεσματικά τα δεδομένα με όλες τις άλλες διεργασίες, επιτρέποντάς τους να εκτελούν τους αντίστοιχους υπολογισμούς στα υποσύνολα ερωτημάτων που τους έχουν ανατεθεί.

```
1   double* flattened = (double*)malloc(TRINELEMS * PROBDIM *  
2                               sizeof(double));  
3   if (rank == 0) {  
4       int k = 0;  
5       for (int i = 0; i < TRINELEMS; i++) {  
6           for (int j = 0; j < PROBDIM; j++) {  
7               flattened[k++] = xdata[i][j];  
8           }  
9       }  
10      MPI_Bcast(flattened, TRINELEMS * PROBDIM, MPI_DOUBLE, 0,  
11                  MPI_COMM_WORLD);
```

Κατά τη διαδικασία μέτρησης του παραλληλοποιημένου αλγορίθμου k-NN, λαμβάνεται υπόψη μόνο ο χρόνος που δαπανάται για την επεξεργασία των ερωτημάτων. Ο χρόνος για τις λειτουργίες ανάγνωσης και εγγραφής δεδομένων εξαιρείται από τον συνολικό υπολογισμό

του χρόνου. Εστιάζοντας αποκλειστικά στον χρόνο επεξεργασίας των ερωτημάτων, μπορούμε να λάβουμε μια πιο ακριβή εκτίμηση της απόδοσης και της αποδοτικότητας του ίδιου του παράλληλου αλγορίθμου. Αυτό μας επιτρέπει να αναλύσουμε συγκεκριμένα τον αντίκτυπο των τεχνικών παραλληλοποίησης και των μεταβολών στον αριθμό των διεργασιών ή των νημάτων στον χρόνο υπολογισμού. Ο αποκλεισμός του χρόνου ανάγνωσης και εγγραφής δεδομένων διασφαλίζει ότι ο μετρούμενος χρόνος αντικατοπτρίζει τον πραγματικό υπολογιστικό φόρτο εργασίας και τη στρατηγική παραλληλοποίησης που χρησιμοποιείται. Παρέχει σαφέστερη κατανόηση της αποδοτικότητας που επιτυγχάνεται μέσω της παραλληλοποίησης και βοηθά στη βελτιστοποίηση της απόδοσης του αλγορίθμου χωρίς την επίδραση των λειτουργιών εισόδου/εξόδου.

Ακολουθεί απόσπασμα του κώδικα

```
1   for (int i = rank; i < QUERYELEMS; i += size) { /* requests
2       */
3       for (int k = 0; k < PROBDIM; k++)
4           x[k] = xx[i][k];
5
6       t0 = MPI_Wtime();
7       double yp = find_knn_value(x, PROBDIM, NNBS);
8       t1 = MPI_Wtime();
9       t_sum += (t1-t0);
10      if (i == rank) t_first = (t1-t0);
11
12      local_sse += (y[i]-yp)*(y[i]-yp);
13
14      local_err = 100.0*fabs((yp-y[i])/y[i]);
15
16      local_err_sum += local_err;
17
18  }
19
20  MPI_Reduce(&local_sse, &sse, 1, MPI_DOUBLE, MPI_SUM, 0,
21             MPI_COMM_WORLD);
22  MPI_Reduce(&local_err_sum, &err_sum, 1, MPI_DOUBLE, MPI_SUM, 0,
23             MPI_COMM_WORLD);
24
25  MPI_Reduce(&t_first, &t_first_max, 1, MPI_DOUBLE, MPI_MAX, 0,
26             MPI_COMM_WORLD);
27  MPI_Reduce(&t_sum, &t_sum_max, 1, MPI_DOUBLE, MPI_MAX, 0,
28             MPI_COMM_WORLD);
```

OPENMP

Στην έκδοση OpenMP, παραλληλοποιήθηκε ολόκληρος ο βρόχος που είναι υπεύθυνος για την επεξεργασία των ερωτημάτων. Με την εφαρμογή κατάλληλων οδηγιών OpenMP, όπως parallel pragmas, ο φόρτος εργασίας της επεξεργασίας πολλαπλών ερωτημάτων κατανεμήθηκε σε πολλαπλά νήματα. Αυτή η προσέγγιση αξιοποίησε τον παραλληλισμό που είναι διαθέσιμος σε περιβάλλοντα κοινής μνήμης και επέτρεψε αποδοτικούς υπολογισμούς σε πολλαπλούς

πυρήνες ή επεξεργαστές. Με την υιοθέτηση της πρώτης προσέγγισης και την παραλληλοποίηση της επεξεργασίας των ερωτημάτων, τόσο οι εκδόσεις MPI και OpenMP του αλγορίθμου k-NN πέτυχαν παράλληλη εκτέλεση. Οι συγκεκριμένες στρατηγικές που χρησιμοποιήθηκαν, όπως ο κυκλικός διαχωρισμός εργασιών στο MPI και η παραλληλοποίηση ολόκληρου του βρόχου ερωτημάτων στο OpenMP, επέτρεψαν την αποτελεσματική αξιοποίηση των πόρων και την επιτάχυνση του υπολογισμού.

```
1 #pragma omp parallel
2 {
3     double x[PROBDIM];
4     #pragma omp for reduction(+:sse) reduction(+:err_sum
5         ) reduction(+:t_sum)
6     for (int i = 0; i < QUERYELEMS; i++) { /* requests
7         */
8         for (int k = 0; k < PROBDIM; k++)
9             x[k] = xx[i][k];
10
11         t0 = gettime();
12         double yp = find_knn_value(x, PROBDIM, NNBS)
13             ;
14         t1 = gettime();
15         t_sum += (t1-t0);
16         if (i == 0) t_first = (t1-t0);
17
18         sse += (y[i]-yp)*(y[i]-yp);
19
20         err = 100.0*fabs((yp-y[i])/y[i]);
21
22         err_sum += err;
23     }
24 }
```

OpenMP task model

Στην έκδοση OpenMP task model, OpenMP pragmas χρησιμοποιούνται για τον παραλληλισμό του αλγορίθμου βάσει διεργασιών. Κάθε ερώτημα επεξεργάζεται ως μεμονωμένη διεργασία και τα pragma omp atomic εξασφαλίζουν τη σωστή συσσώρευση των κοινών μεταβλητών. Αυτή η προσέγγιση επιτρέπει την αποτελεσματική παράλληλη εκτέλεση του αλγορίθμου, μειώνοντας τον χρόνο υπολογισμού και βελτιώνοντας τις επιδόσεις. Μπορεί να φανεί στον παρακάτω κώδικα:

```

1 #pragma omp parallel shared(sse) shared(err_sum) shared(t_sum)
2     private(err, t0, t1)
3     {
4         double x[PROBDIM];
5         #pragma omp single
6         {
7             for (int i = 0; i < QUERYELEMS; i++) {
8                 #pragma omp task
9                 {
10                    for (int k = 0; k < PROBDIM; k++)
11                        x[k] = xx[i][k];
12
13                    t0 = gettime();
14                    double yp = find_knn_value(x
15                                         , PROBDIM, NNBS);
16                    t1 = gettime();
17                    if (i == 0) t_first = (t1-t0
18                                         );
19                    err = 100.0*fabs((yp-y[i])/y
20                                     [i]);
21
22                    #pragma omp atomic
23                    t_sum += (t1-t0);
24
25                    #pragma omp atomic
26                    sse += (y[i]-yp)*(y[i]-yp);
27
28                    #pragma omp atomic
29                    err_sum += err;
30                }
31            }
32        }

```

Στιγμιότυπα εκτέλεσης

Ακολουθούν στιγμιότυπα εκτέλεσης των παραπάνω διαμορφώσεων για κάθε μοντέλο για 128, 256, 512 και 1024 data points

MPI

```
christieChris-PC>Desktop/MPI$ make clean
christieChris-PC>Desktop/MPI$ make
christieChris-PC>Desktop/MPI$ make
gcc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=128 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c gendata.c
gcc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=128 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
mpicc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=128 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
mpicc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=128 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
128 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 1 ./myknrMPI t.txt q.txt
Results for 128 query points
APE = 4.84 %
MSE = 1.318851
R2 = 1 - (MSE/Vari) = 0.934613
Total time = 1750.860157 ms
Time for 1st query = 14.884233 ms
Time for 2..N queries = 1741.208924 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ mplexec -n 2 ./myknrMPI t.txt q.txt
Results for 128 query points
APE = 4.84 %
MSE = 1.318851
R2 = 1 - (MSE/Vari) = 0.934613
Total time = 1750.860157 ms
Time for 1st query = 24.759293 ms
Time for 2..N queries = 1151.161394 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ mplexec -n 3 ./myknrMPI t.txt q.txt
Results for 128 query points
APE = 4.84 %
MSE = 1.318851
R2 = 1 - (MSE/Vari) = 0.934613
Total time = 1750.860157 ms
Time for 1st query = 33.185482 ms
Time for 2..N queries = 1157.331467 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ mplexec -n 4 ./myknrMPI t.txt q.txt
Results for 128 query points
APE = 4.84 %
MSE = 1.318851
R2 = 1 - (MSE/Vari) = 0.934613
Total time = 1851.689215 ms
Time for 1st query = 44.521332 ms
Time for 2..N queries = 517.166283 ms
Average time/query = 7.568348 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
128 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 1 ./myknrMPI t.txt q.txt
Results for 256 query points
APE = 4.84 %
MSE = 1.518757
R2 = 1 - (MSE/Vari) = 0.925572
Total time = 3824.688725 ms
Time for 1st query = 12.875319 ms
Time for 2..N queries = 2865.99907 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
256 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 2 ./myknrMPI t.txt q.txt
Results for 256 query points
APE = 4.84 %
MSE = 1.518757
R2 = 1 - (MSE/Vari) = 0.925572
Total time = 3824.688725 ms
Time for 1st query = 12.875319 ms
Time for 2..N queries = 2179.378271 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
256 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 3 ./myknrMPI t.txt q.txt
Results for 256 query points
APE = 4.84 %
MSE = 1.518757
R2 = 1 - (MSE/Vari) = 0.925572
Total time = 3824.688725 ms
Time for 1st query = 24.895140 ms
Time for 2..N queries = 1874.772549 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
256 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 4 ./myknrMPI t.txt q.txt
Results for 256 query points
APE = 4.84 %
MSE = 1.518757
R2 = 1 - (MSE/Vari) = 0.925572
Total time = 3824.688725 ms
Time for 1st query = 24.895140 ms
Time for 2..N queries = 1879.981838 ms
Average time/query = 7.568348 ms
```

```
christieChris-PC>Desktop/MPI$ make clean
christieChris-PC>Desktop/MPI$ make
christieChris-PC>Desktop/MPI$ make
gcc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=512 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c gendata.c
gcc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=512 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
mpicc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=512 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
mpicc -OPIBODIM=16 -DNBS=32 -DTRAINELEMS=1848576 -DQUERYELEM=512 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -c myknrMPI.c
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
512 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 1 ./myknrMPI t.txt q.txt
Results for 512 query points
APE = 4.84 %
MSE = 1.759052
R2 = 1 - (MSE/Vari) = 0.924778
Total time = 3556.528521 ms
Time for 1st query = 4087.079494 ms
Time for 2..N queries = 3845.495939 ms
Average time/query = 11.493327 ms
christieChris-PC>Desktop/MPI$ mplexec -n 2 ./myknrMPI t.txt q.txt
Results for 512 query points
APE = 4.84 %
MSE = 1.759052
R2 = 1 - (MSE/Vari) = 0.924778
Total time = 4087.079494 ms
Time for 1st query = 4087.079494 ms
Time for 2..N queries = 4072.305441 ms
Average time/query = 7.969287 ms
christieChris-PC>Desktop/MPI$ mplexec -n 3 ./myknrMPI t.txt q.txt
Results for 512 query points
APE = 4.84 %
MSE = 1.759052
R2 = 1 - (MSE/Vari) = 0.924778
Total time = 4087.079494 ms
Time for 1st query = 4085.349598 ms
Time for 2..N queries = 4085.349598 ms
Average time/query = 7.969287 ms
christieChris-PC>Desktop/MPI$ mplexec -n 4 ./myknrMPI t.txt q.txt
Results for 512 query points
APE = 4.84 %
MSE = 1.759052
R2 = 1 - (MSE/Vari) = 0.924778
Total time = 4087.079494 ms
Time for 1st query = 30.356686 ms
Time for 2..N queries = 4045.349598 ms
Average time/query = 7.969287 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
512 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 1 ./myknrMPI t.txt q.txt
Results for 1024 query points
APE = 4.84 %
MSE = 1.080333
R2 = 1 - (MSE/Vari) = 0.920949
Total time = 6189.668054 ms
Time for 1st query = 61.899447 ms
Time for 2..N queries = 11875.774145 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
1024 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 1 ./myknrMPI t.txt q.txt
Results for 1024 query points
APE = 4.84 %
MSE = 1.080333
R2 = 1 - (MSE/Vari) = 0.920949
Total time = 6189.668054 ms
Time for 1st query = 61.899447 ms
Time for 2..N queries = 11879.774145 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
1024 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 2 ./myknrMPI t.txt q.txt
Results for 1024 query points
APE = 4.84 %
MSE = 1.080333
R2 = 1 - (MSE/Vari) = 0.920949
Total time = 6189.668054 ms
Time for 1st query = 21.885872 ms
Time for 2..N queries = 7466.884851 ms
Average time/query = 10.884666 ms
christieChris-PC>Desktop/MPI$ ./gendata t.txt q.txt
1848576 data points written to t.txt!
1024 data points written to q.txt!
christieChris-PC>Desktop/MPI$ mplexec -n 3 ./myknrMPI t.txt q.txt
Results for 1024 query points
APE = 4.84 %
MSE = 1.080333
R2 = 1 - (MSE/Vari) = 0.920949
Total time = 6189.668054 ms
Time for 1st query = 33.835649 ms
Time for 2..N queries = 7666.339341 ms
Average time/query = 7.493992 ms
```

OpenMP

```

chris@chris-PC:~/Desktop/openMP$ make clean
chris@chris-PC:~/Desktop/openMP$ make
pc -O2 -fopenmp -fno-strict-aliasing -DTRAILERLEMS=1048576 -DQUEERYLEMS=1024 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c gendata.c
cc -O2 -fopenmp -fno-strict-aliasing -DTRAILERLEMS=1048576 -DQUEERYLEMS=1024 -DLB=0 -DUB=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c myknOMP.c
chris@chris-PC:~/Desktop/openMP$ ./gendata t.txt q.txt
chris@chris-PC:~/Desktop/openMP$ ./gendata t.txt q.txt x.txt
1024 data points written to x.txt

chris@chris-PC:~/Desktop/openMP$ ./myknOMP T.txt q.txt 1
[...]
UPE = 4.86 s
Total time = 8.929000 ms
UPE = 1 - (MEZ/var) = 0.929000
Time for 1 query = 11.895140 ms
Time for 2, N queries = 11144.866159 ms
Time for 2, N queries = 11144.866159 ms
chris@chris-PC:~/Desktop/openMP$ ./myknOMP T.txt q.txt 2
[...]
UPE = 4.86 s
Total time = 8.929000 ms
UPE = 1 - (MEZ/var) = 0.929000
Time for 1 query = 11.895140 ms
Time for 2, N queries = 11144.866159 ms
Time for 2, N queries = 11144.866159 ms
chris@chris-PC:~/Desktop/openMP$ ./myknOMP T.txt q.txt 3
[...]
UPE = 4.86 s
Total time = 8.929000 ms
UPE = 1 - (MEZ/var) = 0.929000
Time for 1 query = 11.895140 ms
Time for 2, N queries = 2864.999814 ms
Time for 2, N queries = 2864.999814 ms
chris@chris-PC:~/Desktop/openMP$ ./myknOMP T.txt q.txt 4
[...]
UPE = 4.86 s
Total time = 8.929000 ms
UPE = 1 - (MEZ/var) = 0.929000
Time for 1 query = 11.895140 ms
Time for 2, N queries = 1863.748729 ms
Time for 2, N queries = 1863.748729 ms
Average time/query = 1.823046 ms

```

OpenMP Task Model

```

christieChris-PC:/Desktop/OpenMP_Task_Models
christieChris-PC:/Desktop/OpenMP_Task_Models make clean
christieChris-PC:/Desktop/OpenMP_Task_Models make
christieChris-PC:/Desktop/OpenMP_Task_Models make
g++ -fopenmp -O3 -fopenmp=32 -DTRAINLEMS=202144 -DQUERYLEMS=512 -DLB=0 -DBR=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c gendata.c
g++ -fopenmp -O3 -fopenmp=32 -DTRAINLEMS=202144 -DQUERYLEMS=512 -DLB=0 -DBR=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c myknOMPCT.c
g++ -fopenmp -O3 -fopenmp=32 -DTRAINLEMS=202144 -DQUERYLEMS=1024 -DLB=0 -DBR=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c myknOMPCT.c
g++ -fopenmp -O3 -fopenmp=32 -DTRAINLEMS=202144 -DQUERYLEMS=1024 -DLB=0 -DBR=2 -g -ggdb -O3 -DSURROGATES -Wall -fopenmp -c myknOMPCT.c
christieChris-PC:/Desktop/OpenMP_Task_Models ./gendata t.txt q.txt
202144 data points written to t.txt
512 data points written to q.txt
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 1
Results for 512 query points
APE = 5.41
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.887536
Total time = 1434.34949 ms
Time for 1st query = 1.000000 ms
Time for 2. N queries < 1431.979818 ms
Average time/query = 3.837728 ms
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 2
Results for 512 query points
APE = 5.41
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.887536
Total time = 1892.093108 ms
Time for 1st query = 8.188963 ms
Time for 2. N queries < 1891.904557 ms
Average time/query = 3.666781 ms
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 3
Results for 512 query points
APE = 5.41
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.887536
Total time = 2353.944691 ms
Time for 1st query = 2.988893 ms
Time for 2. N queries < 2353.755708 ms
Average time/query = 2.821853 ms
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 4
Results for 1024 query points
APE = 5.48
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.892314
Total time = 2886.755221 ms
Time for 2. N queries < 2886.755228 ms
Average time/query = 2.821853 ms
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 5
Results for 1024 query points
APE = 5.48
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.892314
Total time = 3733.929278 ms
Time for 1st query = 3.692959 ms
Time for 2. N queries < 3733.929281 ms
Average time/query = 3.643599 ms
christieChris-PC:/Desktop/OpenMP_Task_Models ./myknOMPCT t.txt q.txt 6
Results for 1024 query points
APE = 5.48
MSE = 2.399500
R2 = 1 - (MSE/Var) = 0.892314
Total time = 4697.681189 ms
Time for 1st query = 2.924919 ms
Time for 2. N queries < 4697.681196 ms
Average time/query = 4.589285 ms

```

Τεχνικές Βελτιστοποίησης

Κατά την ανάπτυξη και την εκτέλεση του προγράμματος χρησιμοποιήθηκαν διάφορες στρατηγικές βελτιστοποίησης για τη βελτίωση της απόδοσής του. Μια τέτοια στρατηγική περιελάμβανε τη χρήση της σημαίας Ο3 κατά τη μεταγλώττιση, η οποία επιτρέπει στον μεταγλωττιστή να εφαρμόζει επιθετικές βελτιστοποίησεις. Αυτή η σημαία βελτιστοποίησης συμβάλλει στη δημιουργία ιδιαίτερα βελτιστοποιημένου κώδικα, οδηγώντας σε βελτιωμένη ταχύτητα και αποδοτικότητα εκτέλεσης.

Για τη βελτιστοποίηση των λειτουργιών επικοινωνίας, χρησιμοποιήθηκαν τεχνικές συλλογικής επικοινωνίας αντί για επικοινωνίες σημείο προς σημείο. Reduction και broadcast, χρησιμοποιήθηκαν όποτε ήταν δυνατόν. Αυτές οι λειτουργίες επιτρέπουν την αποδοτικότερη ανταλλαγή δεδομένων μεταξύ διεργασιών, μειώνοντας την επιβάρυνση που σχετίζεται με την ατομική επικοινωνία από σημείο σε σημείο. Με τη χρήση συλλογικής επικοινωνίας, το πρόγραμμα μπορεί να εκμεταλλευτεί τον εγγενή παραλληλισμό και να βελτιστοποιήσει τη μεταφορά δεδομένων, οδηγώντας σε βελτιωμένη απόδοση. Στην υλοποίηση OpenMP, αποφεύχθηκαν κρίσιμα τμήματα για την ελαχιστοποίηση των σημείων συμφόρησης και τη μεγιστοποίηση του παραλληλισμού. Αντ' αυτού, χρησιμοποιήθηκε μια πράξη μείωσης για υπολογισμούς που αφορούσαν κοινές μεταβλητές. Με τη χρήση της αναγωγής, το πρόγραμμα μπορεί να συνδυάσει αποτελεσματικά τα αποτελέσματα των παράλληλων υπολογισμών χωρίς να βασίζεται σε ακριβούς μηχανισμούς συγχρονισμού. Αυτή η τεχνική βελτιστοποίησης συμβάλλει στην ενίσχυση της παράλληλης εκτέλεσης και της συνολικής απόδοσης του προγράμματος. Μια άλλη τεχνική βελτιστοποίησης περιελάμβανε τη χρήση τοπικών μεταβλητών για τις μονάδες επεξεργασίας. Αυτή η προσέγγιση βελτιώνει την απόδοση ελαχιστοποιώντας την ανάγκη συγχρονισμού και επιτρέποντας ταχύτερη επεξεργασία εντός κάθε μονάδας επεξεργασίας. Η αξιοποίηση των τοπικών μεταβλητών βοηθά στην αποτελεσματική εκμετάλλευση του παραλληλισμού και στη βελτιστοποίηση της ταχύτητας εκτέλεσης του προγράμματος.

III Πειραματική αξιολόγηση

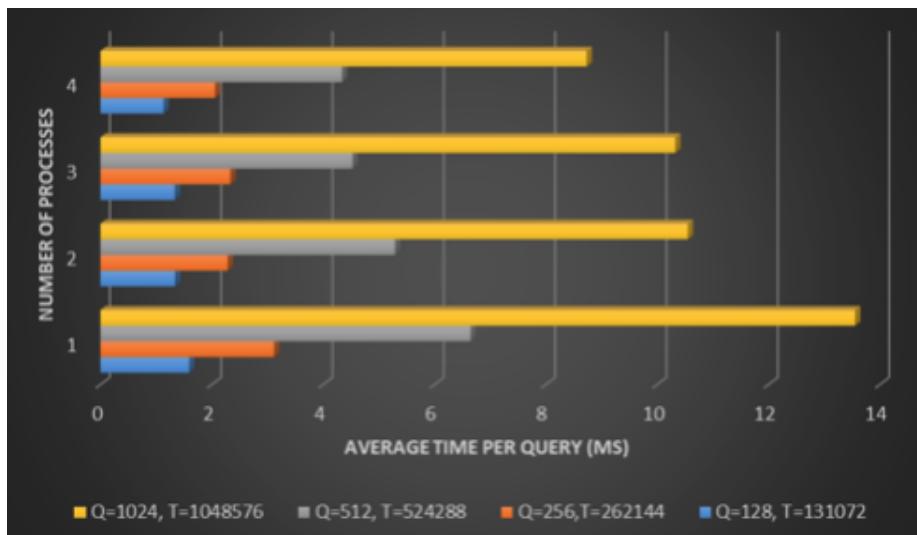
Τα πειράματα διεξήχθησαν σε προσομοιωμένη διανομή Linux βασισμένη στο Debian με τέσσερις εικονικούς πυρήνες και 12 GB μνήμης RAM. Το λειτουργικό σύστημα Linux χρησιμοποιήθηκε μαζί με τον μεταγλωττιστή gcc και την εφαρμογή OpenMPI του MPI. Οι πειραματικές δοκιμές πραγματοποιήθηκαν χρησιμοποιώντας διαφορετικά σύνολα δεδομένων και ποικίλους αριθμούς νημάτων, όπως φαίνεται στον παρακάτω πίνακα:

Query size	Training size
128	131072
256	262144
512	524288
1024	1048576

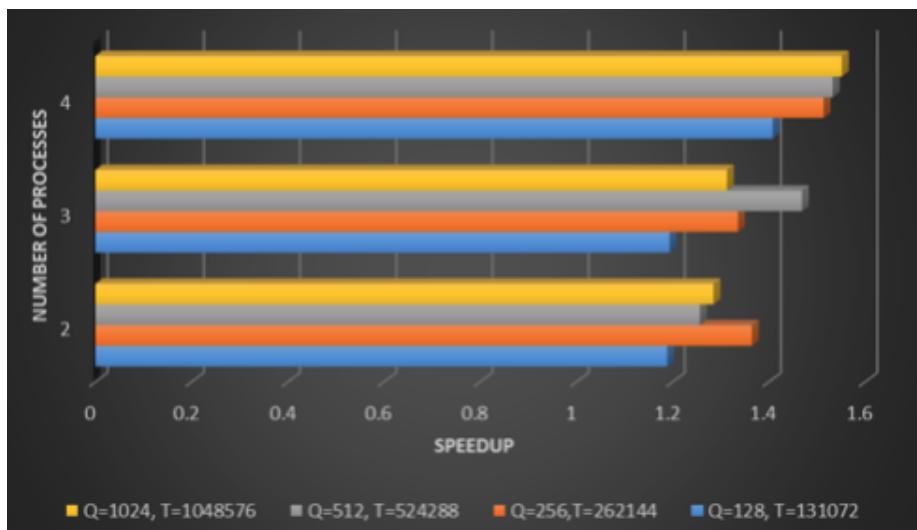
Επιπλέον, κατά τη διαδικασία αξιολόγησης, πειραματίστηκαν διαφορετικοί αριθμοί διεργασιών/νημάτων. Ο χρόνος εκτέλεσης και η επιτάχυνση (που είναι η αναλογία του χρόνου σειριακής εκτέλεσης προς τον παράλληλο χρόνο) μετρήθηκαν για την αξιολόγηση της απόδοσης. Ακολουθούν τα αποτελέσματα

IV MPI

Στα σχήματα 1 και 2, παρουσιάζονται τα αποτελέσματα που προέκυψαν από την έκδοση MPI (Message Passing Interface) του αλγορίθμου. Είναι σαφές από τα γραφήματα ότι όσο αυξάνεται ο αριθμός των διεργασιών, αυξάνεται και η επιτάχυνση. Αυτό δείχνει ότι η παραλληλοποίηση του αλγόριθμου χρησιμοποιώντας MPI μπορεί να βελτιώσει αποτελεσματικά την απόδοσή του. Επιπλέον, η επιτάχυνση δείχνει επίσης μια αύξηση με μεγαλύτερα σύνολα δεδομένων. Αυτό συμβαίνει επειδή τα μεγαλύτερα σύνολα δεδομένων παρέχουν περισσότερες ευκαιρίες για εκμετάλλευση του παραλληλισμού. Με περισσότερα σημεία δεδομένων, ο αλγόριθμος μπορεί να κατανείμει το φόρτο εργασίας σε πολλαπλές διεργασίες, οδηγώντας σε καλύτερη χρήση των υπολογιστικών πόρων και ταχύτερους χρόνους εκτέλεσης. Ωστόσο, είναι σημαντικό να σημειωθεί ότι η επιτάχυνση δεν αυξάνεται αναλογικά με την προστιθέμενη ισχύ του υπολογιστή ή τον αριθμό των διεργασιών. Αυτό οφείλεται σε παράλληλη κατανομή πόρων, οι οποίοι συμβάλλουν στην επικοινωνία, στον υπολογισμό και συγχρονισμό. Καθώς προστίθενται περισσότερες διεργασίες, το κόστος επικοινωνίας μεταξύ των διεργασιών μπορεί να αυξηθεί. Αυτό συμβαίνει επειδή η ανταλλαγή πληροφοριών και ο συντονισμός των υπολογισμών σε πολλαπλές διεργασίες απαιτεί επιπλέον χρόνο και πόρους. Συνεπώς, η αποτελεσματικότητα της παραλληλοποίησης μπορεί να μειωθεί όσο αυξάνεται ο αριθμός των διεργασιών. Αυτό το φαινόμενο αναφέρεται συχνά ως parallelization overhead.



Σχήμα 1: χρόνος εκτέλεσης MPI

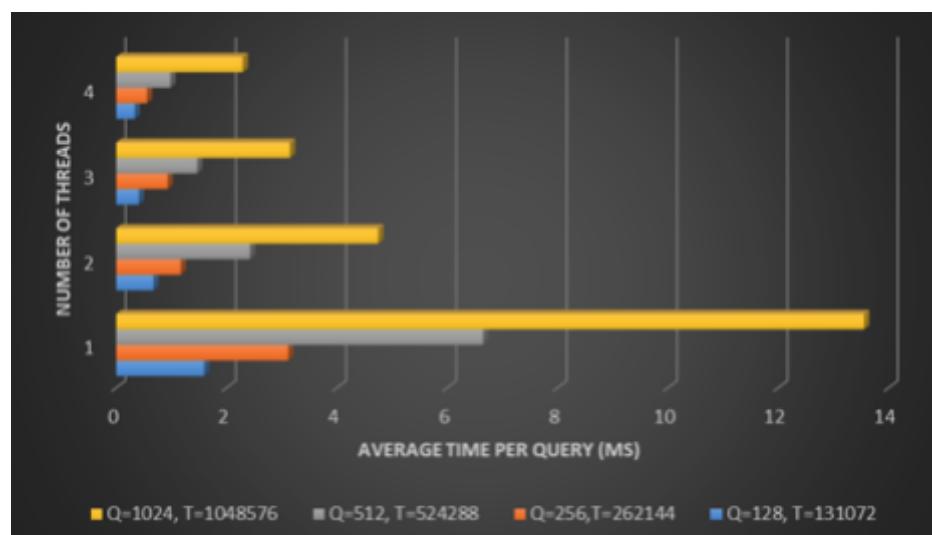


Σχήμα 2: Speedup MPI

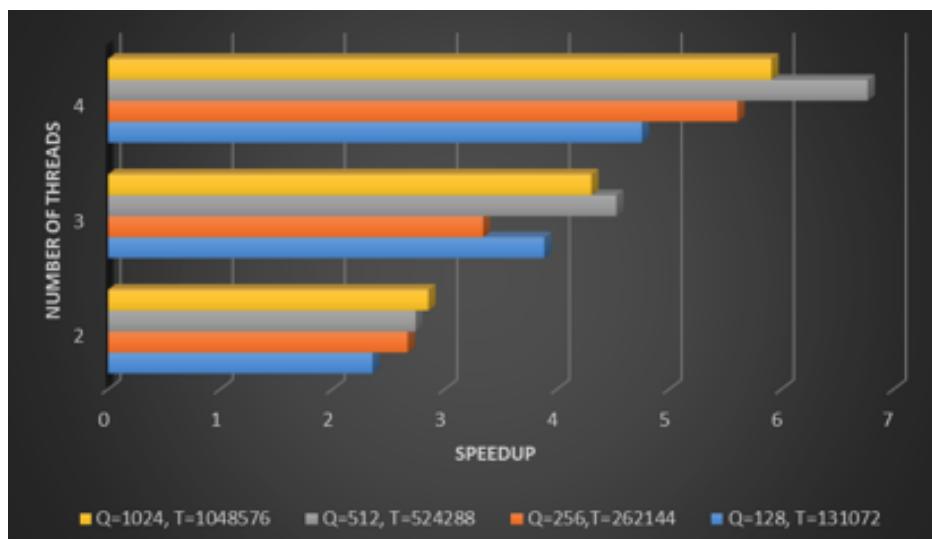
V OpenMP

Τα σχήματα 3 και 4 παρουσιάζουν τα αποτελέσματα που προέκυψαν από την έκδοση OpenMP του αλγορίθμου. Παρατηρείται ότι όσο αυξάνεται ο αριθμός των νημάτων, η επιτάχυνση βελτιώνεται. Αυτό υποδηλώνει ότι η παραλληλοποίηση του αλγόριθμου k-NN χρησιμοποιώντας το OpenMP μπορεί να βελτιώσει αποτελεσματικά την απόδοσή του. Ομοίως, με την έκδοση MPI, η επιτάχυνση παρουσιάζει επίσης αύξηση με μεγαλύτερα σύνολα δεδομένων. Αυτό συμβαίνει επειδή τα μεγαλύτερα σύνολα δεδομένων παρέχουν περισσότερες ευκαιρίες για την αξιοποίηση του παραλληλισμού. Με περισσότερα σημεία δεδομένων, ο αλγόριθμος

μπορεί να κατανείμει τον υπολογιστικό φόρτο εργασίας σε πολλαπλά νήματα, οδηγώντας σε βελτιωμένη χρήση των υπολογιστικών πόρων και ταχύτερους χρόνους εκτέλεσης. Μια αξιοσημείωτη διαφορά μεταξύ των εκδόσεων OpenMP και MPI είναι ότι δεν χρειάζεται επιπλέον χρόνος και πόροι για τη διαχείριση της επικοινωνίας μεταξύ παράλληλων διεργασιών. Σε αντίθεση με το MPI, το οποίο απαιτεί επικοινωνία μεταξύ των διεργασιών, το OpenMP αξιοποιεί τον παραλληλισμό κοινής μνήμης σε έναν μόνο κόμβο. Κατά συνέπεια, συμβάλει στην επίτευξη υψηλότερης ταχύτητας σε σύγκριση με την έκδοση MPI. Η υλοποίηση OpenMP μπορεί να χρησιμοποιήσει πιο αποτελεσματικά τους διαθέσιμους υπολογιστικούς πόρους, με αποτέλεσμα βελτιωμένη επεκτασιμότητα και απόδοση.



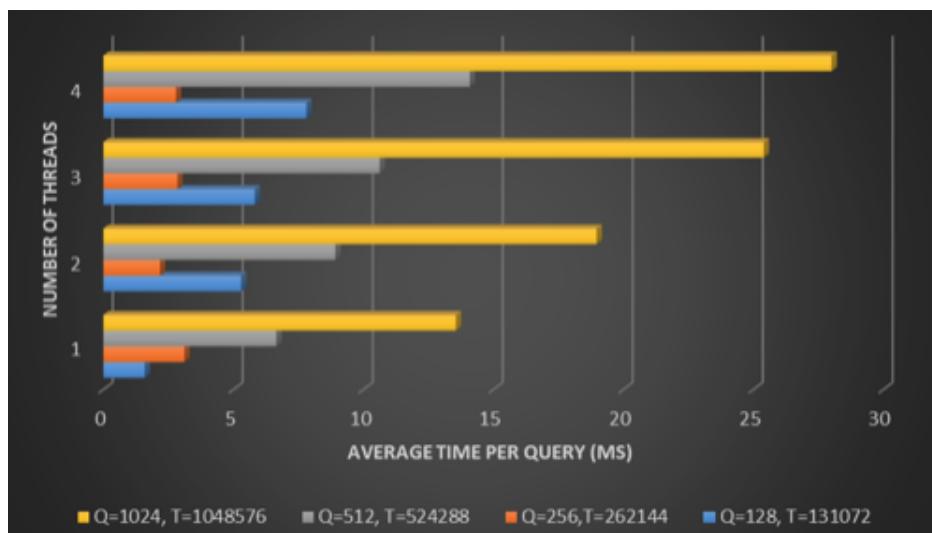
Σχήμα 3: χρόνος εκτέλεσης OpenMP



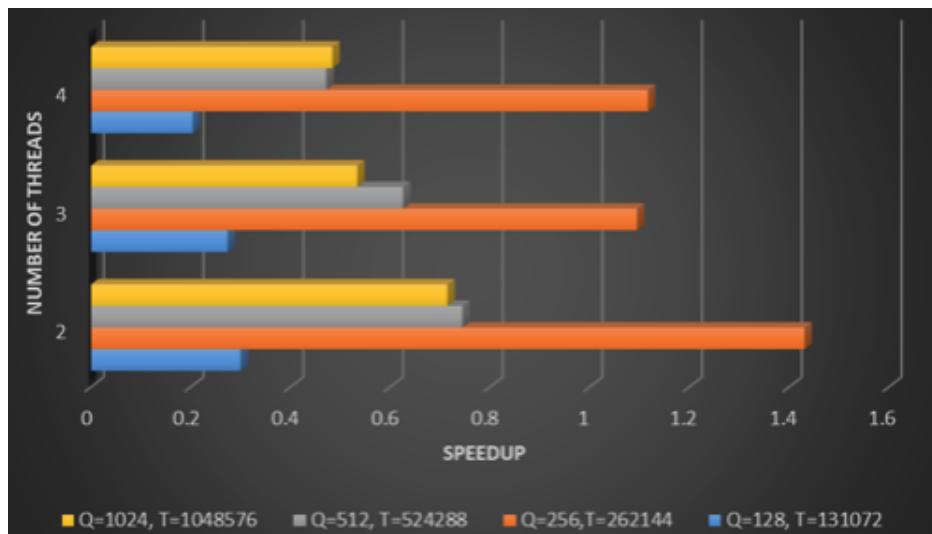
Σχήμα 4: Speedup OpenMP

VI OpenMP Task Model

Τα αποτελέσματα για την έκδοση του μοντέλου εργασιών OpenMP του αλγόριθμου απεικονίζονται στα Σχήματα 5 και 6. Είναι προφανές από τα γραφήματα ότι η επιτάχυνση που επιτυγχάνεται χρησιμοποιώντας την προσέγγιση μοντελοποίησης εργασιών δεν είναι ιδιαίτερα ενθαρρυντική σε σύγκριση με άλλες εκδόσεις. Η επιτάχυνση που παρατηρείται σε αυτά τα σχήματα υποδηλώνει ότι άλλες στρατηγικές παραληλοποίησης μπορεί να έχουν καλύτερη απόδοση από την υλοποίηση του μοντέλου εργασίας. Είναι σημαντικό να σημειωθεί ότι μια αξιοσημείωτη επιτάχυνση παρατηρείται μόνο όταν χρησιμοποιούνται δύο νήματα. Αυτό υποδηλώνει ότι η έκδοση του μοντέλου εργασιών ενδέχεται να μην εκμεταλλεύεται αποτελεσματικά τον παραλληλισμό πέρα από έναν ορισμένο αριθμό νημάτων.



Σχήμα 5: χρόνος εκτέλεσης OpenMP Task Model



Σχήμα 6: Speedup OpenMP Task Model

VII Σύνοψη

Συνοψίζοντας, η έκδοση MPI δείχνει αυξανόμενη ταχύτητα με μεγαλύτερο αριθμό διεργασιών, ενώ και οι δύο εκδόσεις OpenMP και OpenMP Task Model παρουσιάζουν βελτιωμένη επιτάχυνση με αύξηση του αριθμού των νημάτων. Επιπλέον, τα μεγαλύτερα σύνολα δεδομένων συμβάλλουν στην καλύτερη επιτάχυνση σε όλες τις εκδόσεις, εκτός από την έκδοση OpenMP Task Model, η οποία γενικά έχει χαμηλότερη απόδοση σε σύγκριση με τις άλλες προσεγγίσεις.