Εργασία Παράλληλα Και Διανεμημένα Συστήματα 7ο Εξάμηνο

Μπεκιάρης Θεοφάνης
ΑΕΜ:8200

Παραλληλοποίηση με **OpenMP**

Συνάρτηση **Test_Octree**:
Οι αλλαγές που έχουν γίνει στην συνάρτηση είναι να εισάγουμε στην global μεταβλητή numthreads τον αριθμός των threads μέσω των ορισμάτων της main,και η κλίση της omp_set_num_threads για τον καθορισμό του πλήθους των νημάτων κατά την εκτέλεση του προγράμματος.

Συναρτήσεις **Hash_code**,**Morton_enconding** και **Data_rearrangement:**
Στις συναρτήσεις αυτές οι παραλληλοποίση έχει γίνει με την ίδια λογική.Οι συναρτήσεις περιέχουν for loops μεγέθους N(αριθμός σωματιδίων) χωρίς να χρειάζεται κάποιος συχρονισμός καθώς τα δεδομένα εγγράφονται σε κάθε επανάληψη σε διαφορετικές θέσεις μνήμης.Επομένως το μόνο που χρειάζεται είναι η εισαγωγή των δηλώσεων #pragma omp parallel for πρίν απο την εκτέλεση αυτών των foor.Να τονιστεί οτι σε σημεία με loop λίγων επαναλήψεων η παραλληλοποίση δημιουργεί καθυστέρηση στο πρόγραμμα, επειδή ο κώδικας μιας παράλληλης περιοχής αντιγράφεται τόσες φορές όσα τα νήματα και δίνεται σε αυτά για να τον εκτελέσουν χρησιμοποιώντας πόρους του συστήματος.Αρα η σειριακή εκτέλεση μικρών loop σε αυτή την περίπτωση καθίσταται πιο γρήγορη. Επομένως μικρά loops και σημεία του κώδικα με λίγες εντολές δεν έχουν παραλληλοποιηθεί.

Συνάρτηση **Radix_sort:**
Τα κομμάτια του κώδικα της συνάρτησης radix_sort που έχουν επιλεχθεί για παραλληλοποίση είναι η πρώτη for των N επαναλήψεων και το κομμάτι της αναδρομικής κλίσης της truncated_radix_sort.Η foor των 8 επαναλήψεων είναι υπερβολικά μικρή για παραλληλοποίηση.Η δεύτερη foor των N επαναλήψεων σε κάθε επανάληψη κάνει προσπέλαση σε ένα απο τα 8 στοιχεία του πίνακα BinCursor[] και αφού κάνει τις αντίστοιχες μεταρτοπές στους πίνακες permutation_vector[] και sorted_morton_codes[] τότε αυξάνεται κατα 1.Αυτή η αύξηση σε περίπτωση παραλληλοποίησης θα πρέπει να γίνεται αντιληπτή αμέσως απο τα νήματα που τρέχουν παράλληλα στο κομμάτι της foor επειδή καθορίζει και τα αντίστοιχα αποτελέσματα.Με λίγα λόγια,αυτο σημαίνει

οτι όλα τα νήματα θα πρέπει να έχουν πρόσβαση στην ίδια θέση μνήμης(για την αρκίβεια σε 8 όσα και τα στοιχεία του πίνακα αλλά συγκριτικά με τον μεγάλο αριθμό των N σωματιδίων δεν υφίσταται διαφόρα) στην οποία θα γράφουν και θα διαβάζουν σε κάθε επανάληψη,κάτι που απαιτεί την εισαγωγή κρίσιμης περιοχής (#pragma omp critical) και που στην ουσία ποτέ δεν θα αφήνει τα νήματα να τρέξουν παράλληλα αφού σε κάθε επανάληψη θα πρέπει να γράφουν ένα ένα στην ίδια μεταβλητή.Συμπερασματικά,η παραλληλοποίηση με κρίσιμη περιοχή θα αυξήσει απαγορευτικά τον χρόνο εκτέλεσης και αρα αποφεύγεται.

Η λογική με την οποία παραλληλοποιούμε την πρώτη foor είναι η εξείς:
Απο έναν σύνολο με N στοιχεία πρέπει να μετρήσουμε πόσα παίρνουν τιμή 0,πόσα την τιμή 1,και ούτω καθεξης μέχρι και την τιμή 7.Για την μέτρηση χρησιμοποιείται ο πίνακας BinSizes[] στον οποίο αποθηκεύεται το αντίστοιχο πλήθος των σωματιδίων για την κάθε περίπτωση.Μπορούμε να παραλληλοποιήσουμε το συγκεκριμένο κομμάτι ορίζοντας νήματα τα οποία θα κάνουν καταμέτρηση παράλληλα σε κομμάτια του συνόλου των N στοιχείων.Το κάθε νήμα θα έχει τον δικό του πίνακα για την καταμέτρηση αντίστοιχο του BinSizes[] για να αποφευχθούν οι συγκρούσεις λόγο της εισαγωγή κρίσιμων περιοχών.Στο τέλος κάθε μέτρησης τα αποτελέσματα απο όλους του πίνακες θα αποθηκεύονται τελικά στον πίνακα BinSizes[] μέσα σε μία κρίσιμη περιοχή της οποία ο αριθμός εμφάνησης τελικά εξαρτάται απο το πλήθος των νημάτων και οχι το πλήθος N.Περεταίρω σχόλια για την ακριβή υλοποιήση φαίνονται στον κώδικα.Να σημειωθεί οτι η συγκεκριμένη υλοποίηση πραγματοποιείται μόνο για μεγάλο-κατάλληλο αριθμό στοιχείων ωστέ να έχουμε βελτίωση στην υλοποιήση,να ανοίγουν νέα threads μόνο αν χρειάζεται,δηλαδή υπάρχει πολύ δουλειά για αυτό που τρέχει,ώστε να μην εμφανίζονται ανεπιθύμητες-μεγάλες καθυστερίσεις,ενώ για μικρά N η εκτέλεση γίνεται σειριακά.Επιπλέον το συγκεκριμένο κομμάτι μας ενδιαφέρει κυρίως για την πρώτη φορά που θα καλέσουμε την radix αφού ο αριθμός των σωματιδίων θα είναι μεγάλος.Μετα την αναδρομή θα ανόιξουν νήματα τα οποία θα τρέχουν παράλληλα για μεγάλες δουλειές και η δημιουργία νέων νημάτων μέσα στα ίδια τα νήματα για καταμέτρηση μάλλον θα επιβαρύνει το σύστημα,εκτος και αν οι αριθμοί δεν έιναι ομοιόμορφα κατανεμημένοι και για παράδειγμα μετά την πρώτη αναδρομή βρεθούμε τα τρέχουμε μόνο 2 νήματα για επίσης μεγάλο αριθμό σωματιδίων N.

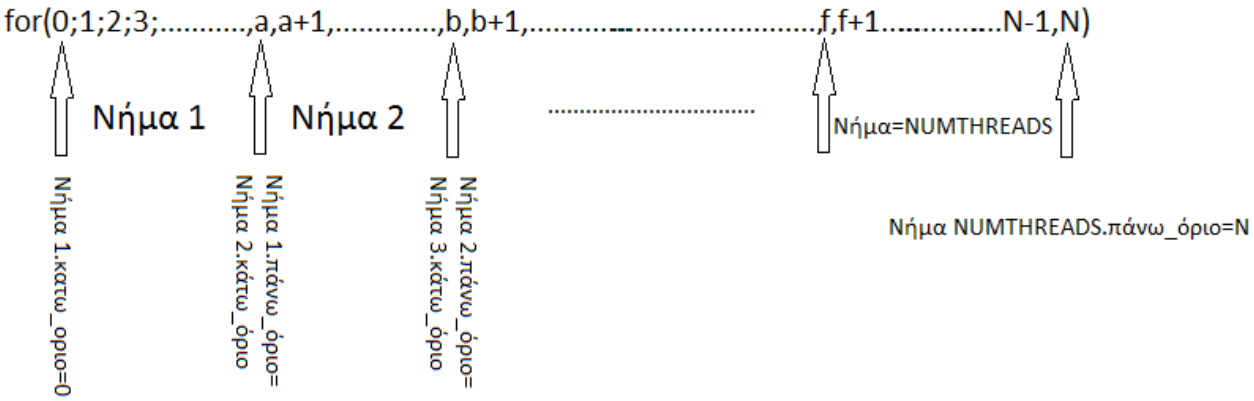
Το κομμάτι της αναδρομής το παραλληλοποιούμε με το εξείς σκεπτικό:
Ορίζουμε μια global μεταβλητή count την οποία χρησιμοποιούμε για να γνωρίζουμε πόσα νήματα είναι ανοιχτά κάθε στιγμή.Το πρόγραμμα συγκρίνει την μεταβλητή numthreads με την μεταβλητή count και αν υπάρχουν ελεύθερα νήματα τα οποία μπορούν να ανοίξουν τότε για όσα γίνεται κάνει την αναδρομή παράλληλα ενώ για τις υπόλοιπες αναδρομές τις κάνει σειριακά μέχρι πάλι να ελευθερωθούν νέα νήματα και τότε ξάνα κάνει παράλληλες αναδρομές.Να

σημειωθεί ότι ο αριθμός των στοιχείων N αποτελεί όπως και πρίν συνθήκη για την δημιουργία παράλληλων περιοχών.Δεν θα πρέπει να ανοίγουν νέα νήματα απλά επειδή υπάρχουν διαθέσημα για να ανοιξουν,διότι όταν εκτελείται ενα νήμα αν το N ειναι μικρό τότε θα ανοίξουν νέα νήματα για μικρές περιοχές(κάτι το οποίο απο μόνο του προσθέτει καθυστέρηση) ενώ σε κάποιο αλλό σημείο του προγράμματος που τρέχει παράλληλα μπορεί να υπάρχουν μεγάλες περιοχές που να μην μπορούν να ανοίξουν παράλληλα λόγο έλειψης νημάτων.

Παραλληλοποίηση με **Pthreads**

Συνάρτηση **Test_Octree**:
Όπως και στην προηγούμενη περίπτωση δέν έχουν γίνει πολλές αλλαγές σε αυτή την συνάρτηση πέρα απο τον καθορισμό του αριθμού των νημάτων.

Συναρτήσεις **Hash_code**,**Morton_enconding** και **Data_rearrangement:**
Για τα μεγάλα σημεία του κώδικα που είναι οι for μήκους N,δημιουργούμε νήματα τα οποία τρέχουν παράλληλα κομμάτια των foor loop με την λόγικη που φαίνεται στο παρακάτω σχήμα.



Αρχικά διαιρούμε το μήκος N της for σε NUMTHREADS(=αριθμός νημάτων) κομμάτια,υπολογίζουμε τα όρια (πάνω-κάτω) των foor loops και καλούμε την αντίστοιχη συνάρτηση για να εκτελέστει παράλληλα για τα αντίστοιχα όρια.Ετσι δημιουργούμε πολλές μικρότερες foor που τρέχουν παράλληλα.Έχει προστεθεί επιπλέον έλεγχος για τον αριθμό του N ωστε να τρέχει παράλληλα μόνο μετά απο το σημείο για το οποίο έχουμε βελτίωση στον χρόνο.Λεπτομέρειες για τον τρόπο υλοποίησης και περεταίρω σχόλια φαίνονται στους αντίστοιχους κώδικες.
Συνάρτηση **Radix_sort:**
Η συνάρτηση παραλληλοποιείται στα ίδια σημεία όπως και στην περίπτωση της openMP.Στο πρώτο κομμάτι της παραλληλοποίησης της πρώτης for,η λογική είναι η ίδια με την παραπάνω ανάλυση(παραπάνω σχήμα).Ορίζουμε μια global μεταβλητή count για των έλεγχο του αριθμού νημάτων και αν υπάρχουν ελεύθερα νήματα τότε μαζί με το νήμα(αρχηγό) που τα καλεί χωρίζουν τα N στοιχεία σε υποτμήματα και τα ελέγχουν παράλληλα.Για να δημιουργήσουμε τα νήματα χρησιμοποιούμε την δομή Binsworkers και την συνάρτηση

parallel_first_step. Στο τέλος αποθηκέυουν τα αποτελέσματα στον πίνακα BinSizes[].
Στο κομμάτι της αναδρομής ελέγχουμε τα ελέυθερα νήματα και τον αριθμό των στοιχείων για τα οποία γίνεται η αναδρομή και αν είναι ικανοποιητικός τότε ανοίγουμε νέο νήμα αλλίως η αναδρομή γίνεται απο το νήμα αρχηγό.Χρησιμοποιούμε την συνάρτηση parallelHelpFunction για να καλέσουμε τα νέα νήματα και την δομή radixworkers για να περάσουμε μέσα σε αυτήν τα απαραίτητα δεδομένα.
Θα πρέπει να παρατηρηθεί ότι σε σχέση με την υλοποίηση της OpenMP και της cilk (παρακάτω) οι οποίες μπορούν και ανοίγουν ταυτοχρονά πολλά νήματα και μέσα σε αυτά περιέχεται και το νήμα αρχηγός,η υπολοποίηση σε Pthread θέλει περισσότερο προσοχή διότι το νήμα αρχηγός ανοίγει ένα ένα νήμα και δεν θα πρέπει να ανήγει νήματα και ο ίδιος να μήν έχει δουλεία να κάνει,επειδή σε αυτή την περίπτωση απλά θα ανοίγει ενα νήμα και ο ίδιος θα περιμένει.Στην υλοποίηση του κώδικα αυτό έχει ληφθεί υπόψη και δίνεται πάντα δουλεία στο νήμα αρχηγό.

Παραλληλοποίηση με **Cilk**

Συνάρτηση **Test_Octree**:
Στην συνάρτηση ορίζουμε όπως και στις προηγούμενες περιπτώσεις τον αριθμό των νημάτων μέσα απο τα ορίσματα της main(με τύπο const char) και καλούμε την συνάρτηση __cilkrts_set_param() για να ορίσουμε τον αριθμό των νημάτων στίς παράλληλες περιοχές.Επιπλέον ορίζουμε reducer για τον μετρητή count.
Συναρτήσεις **Hash_code**,**Morton_enconding** και **Data_rearrangement:**
Στις συναρτήσεις αυτές οι υλοποίηση είναι όπως στην περίπτωση της OpenMP, παραλληλοποιούμε τις μεγάλες foor με την δήλωση cilk_for και λαμβάνουμε υπόψην και τον αριθμό των στοιχείων N ώστε να είναι μεγάλος.Ο αριθμός των νήματων που τρέχουν μέσα στις foor δεν χρειάζεται έλεχγο αφού εχεί οριστεί απο την __cilkrts_set_param().Επιπλέον ορίζουμε reducer τον οποίο περνάμε μέσα στην συνάρτηση RadixSort επειδή τον χρειαζόμαστε ως μετρητή count στην συνάρτηση RadixSort.
Συνάρτηση **Radix_sort:**
Το κομμάτι της πρώτης foor για να το παραλληλοποιήσουμε ορίζουμε δομές που περιέχουν τα όρια της foor στα οποία θα πάει κάθε νήμα να παραλληλοποιήσει και εναν πίνακα αντίστοιχο του BinSizes[] για την καταμέτρηση των στοιχείων απο κάθε νήμα.Στην συνέχεια ανόιγουμε μία cilk_foor μεγέθους ίσο με τον αριθμό των νημάτων που έχουμε ορίσει,για να τρέξουν παράλληλα το καθένα μέσα στα αντίστοιχα όρια που εχούν οριστεί.Δεν χρειάζεται να ελέγχουμε πόσα νήματα τρέχουν μέσα στην cilk_foor επειδή αυτόματα ανόιγουν απο μόνα τους όσα είναι διαθέσημα.
Στο κομμάτι της αναδρομής χρησιμοπούμε έναν μετρητή για τα νήματα και όσα είναι ελεύθερα τα ανοίγουμε παράλληλα με μία cilk_for ενώ αν δεν γίνεται τότε η κλίση εκτελείται σειριακά.

Διαγράμματα

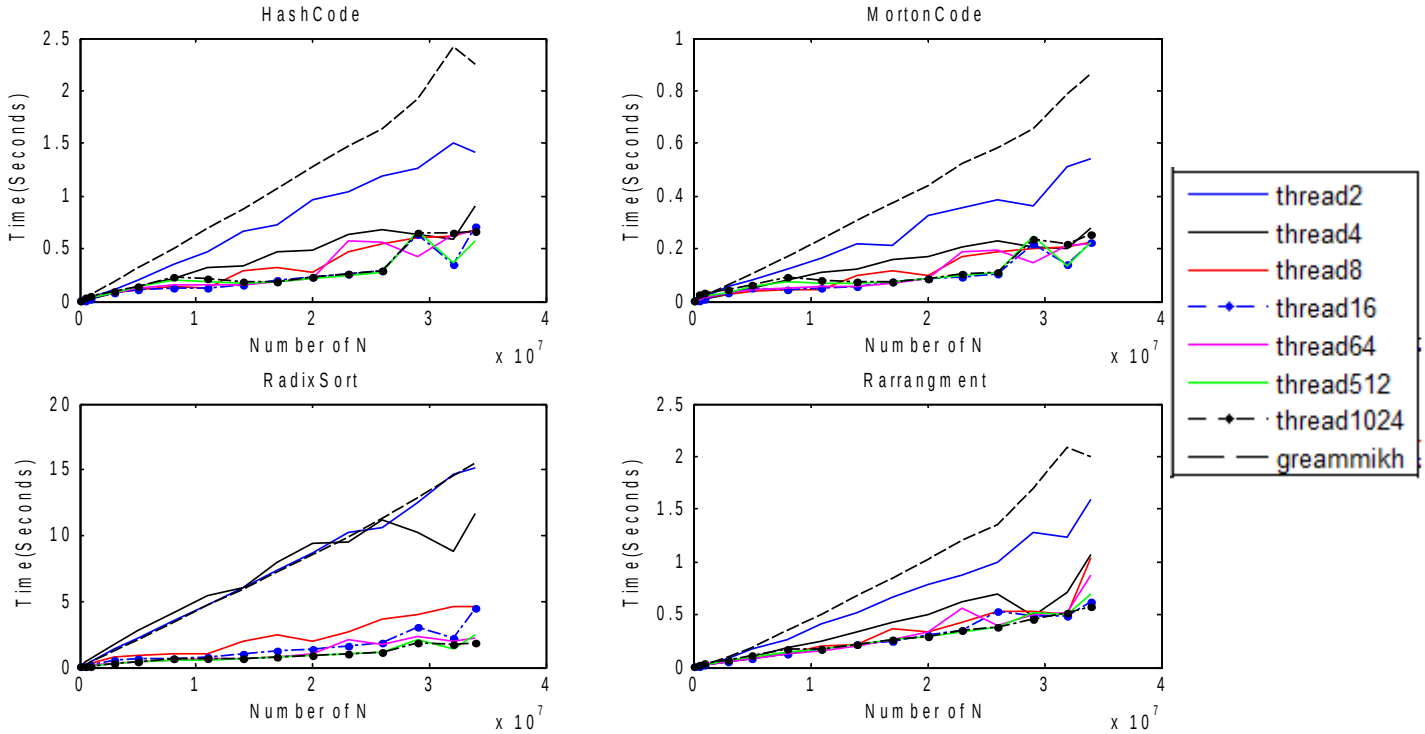
Σε αυτό το σημείο για να μπορέσουμε να μελετήσουμε και να βγάλουμε συμπεράσματα για την απόδοση των προγραμμάτων θα παρουσιάσουμε κάποιες γραφικές παραστάσεις συναρτήση του χρόνου εκτέλεσης των παραλληλοποιημένων προγραμμάτων σε σχέση με το γραμμικό μεταβάλλοντας διάφορες μεταβλητές των προγραμμάτων που επηρεάζουν τον χρόνο εκτέλεσης τους όπως τον αριθμό των νημάτων,τον αριθμό των στοιχείων,το μέγιστο βάθος και το όριο πληθυσμού.

Αρχικά θα δούμε την απόδοση τους συναρτήση του αριθμό των νημάτων για διάφορες τιμές του αριθμού των στοιχείων για την χειρότερη περίπτωση εκτέλεσης του γραμμικού προγράμματος,δηλαδή για βάθος L=18 και μέγιστο πληθυσμό S=1 και για τις δύο κατανομές(κύβου,1/8 του κύκλου).Συγκεκριμένα θα μεταβάλουμε των αριθμό στοιχείων στις τιμές 500 1000 50000 100000 500000 1000000 3000000 5000000 8000000 11000000 14000000 17000000 20000000 23000000 26000000 29000000 32000000 34000000 δηλαδή για τιμές 500 εώς 2^25 και τον αριθμό νημάτων σε τιμές πολλαπλάσια του 2.Η εκτέλεση των προγραμμάτων και η μέτρηση των χρόνων έχουν γίνει στο δικό μου 4πύρηνο προσωπικό σύστημα,όμως στο κομμάτι της cilk για να γίνει σύγκριση χρειάστηκε να εκτελέσω κάποια κομμάτια και στο σύστημα του Διαδή και επομένως σε αυτή την αναφορά θα παρουσιαστούν και κάποιες γραφικές παραστάσεις και απο το οκταπύρηνο σύστημα του Διαδη.

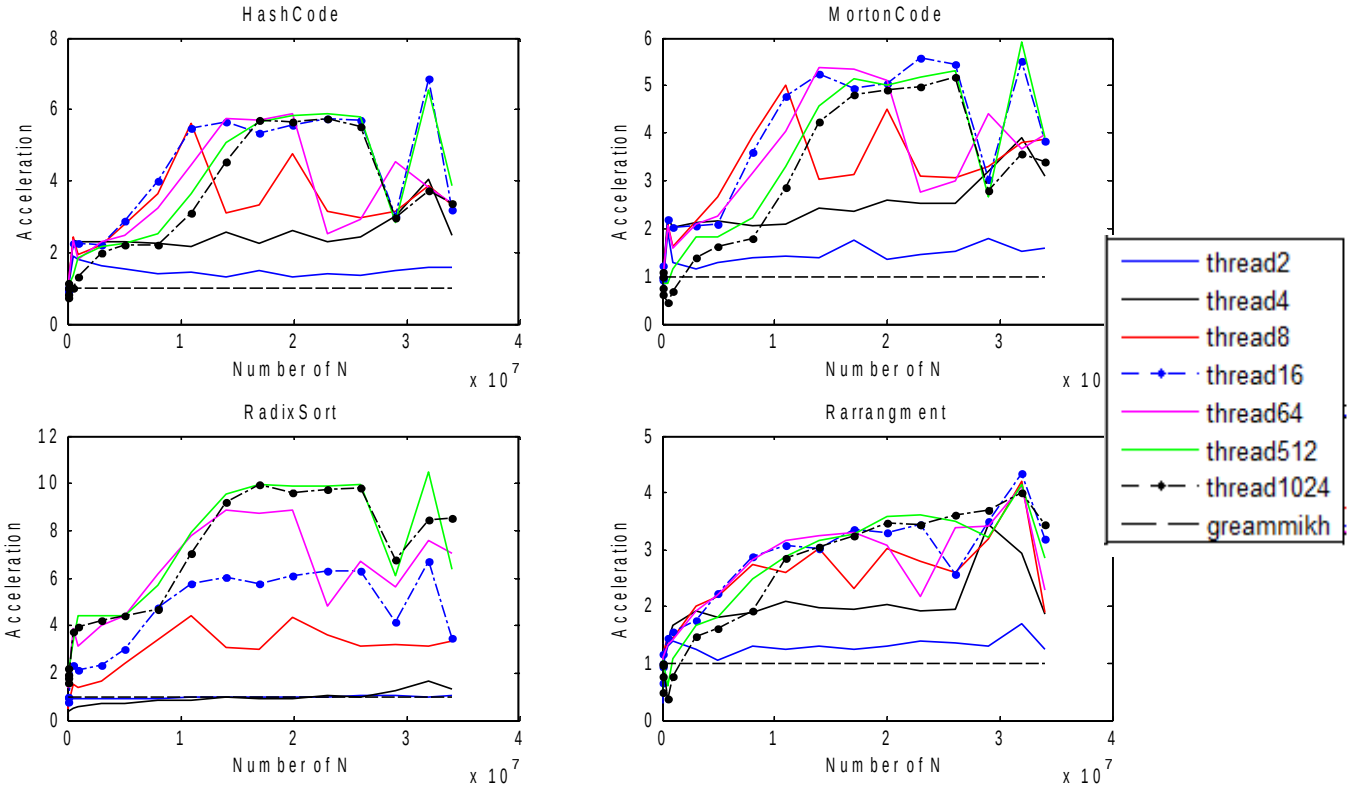
OpenMP

Περίπτωση 1/8 σφαίρας:

Διαγράμματα Χρόνου-ΑριθμούΣωμάτων μεταβάλλοντας των αριθμό μηνάτων



Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)



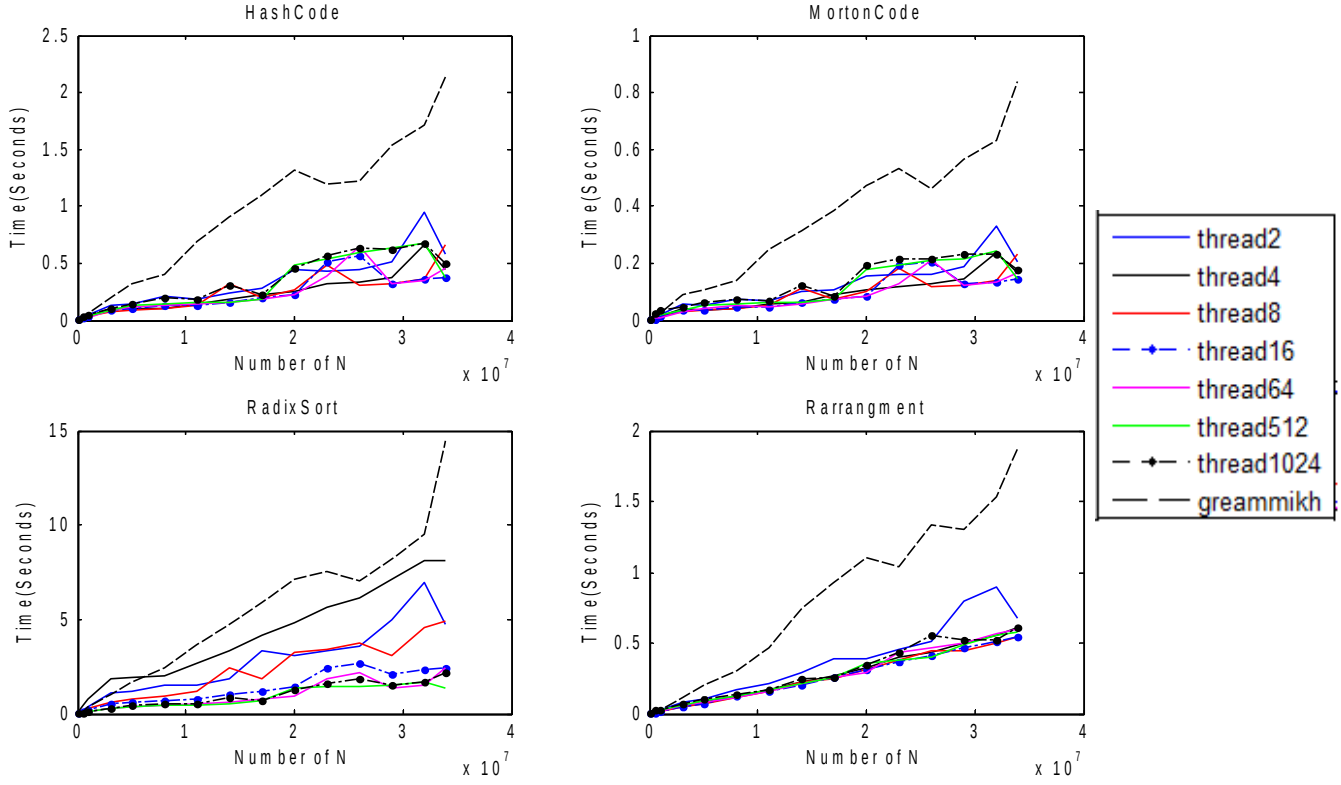
Η κατασκευή των γραφικών παραστάσεων έχει γίνει με την βοήθεια του προγράμματος MATLAB.Στα παραπάνω διαγράμματα οι μάυρες διακεκομένες γραμμές αντιστοιχούν στο αντίσοιχο σειριακό πρόγραμμα ενώ οι υπόλοιπες είναι για την παράλληλη υπολοποίηση για διάφορους αριθμούς νήματων και συγκεκριμένα για 2 4 8 16 64 512 1004 νήματα(δεν μου επιτρέπεται απο το σύστημα να ορίσω περισσότερα απο 1200 νήματα).Για να αποφύγουμε την περίπλοκη περιγραφή και την εκτενή παρουσίαση για ακριβή αντιστοίχηση των γραμμών με τα αντίστοιχα νήματα,μαζί με αυτήν την αναφορά επισυνάπτονται και τα αρχεία των γραφικών παραστάσεων για ευκολότερη παρατήρηση και μελέτη τους με την βοήθεια του MATLAB και του εργαλείου Desktop-> Plot Browser.Οι γενικές παρατηρήσεις απο τα διαγράμματα ειναι:

Α)Ο χρόνος εκτέλεσης βελτιώνεται ακρετά συμαντικά μετά απο τον αριθμό των 4 νημάτων και στις 4 συναρτήσεις καθώς αυξάνεται ο αριθμός των νημάτων και τελικά σταθεροποιείται κοντά σε μία τιμή,και για μεγάλο αριθμό νημάτων(1024) έχουμε μια μικρή χειροτέρευση σε σχέση με τις προηγουμενές τους όπως φαίνεται πιο έυκολα απο τα διαγράμματα επιτάχυνσης.Κανονικά θα περιμέναμε μετά απο κάποιο αριθμό νημάτων λόγο της έλλειψης φυσικών πόρων δηλαδή τον περιορισμένο αριθμό

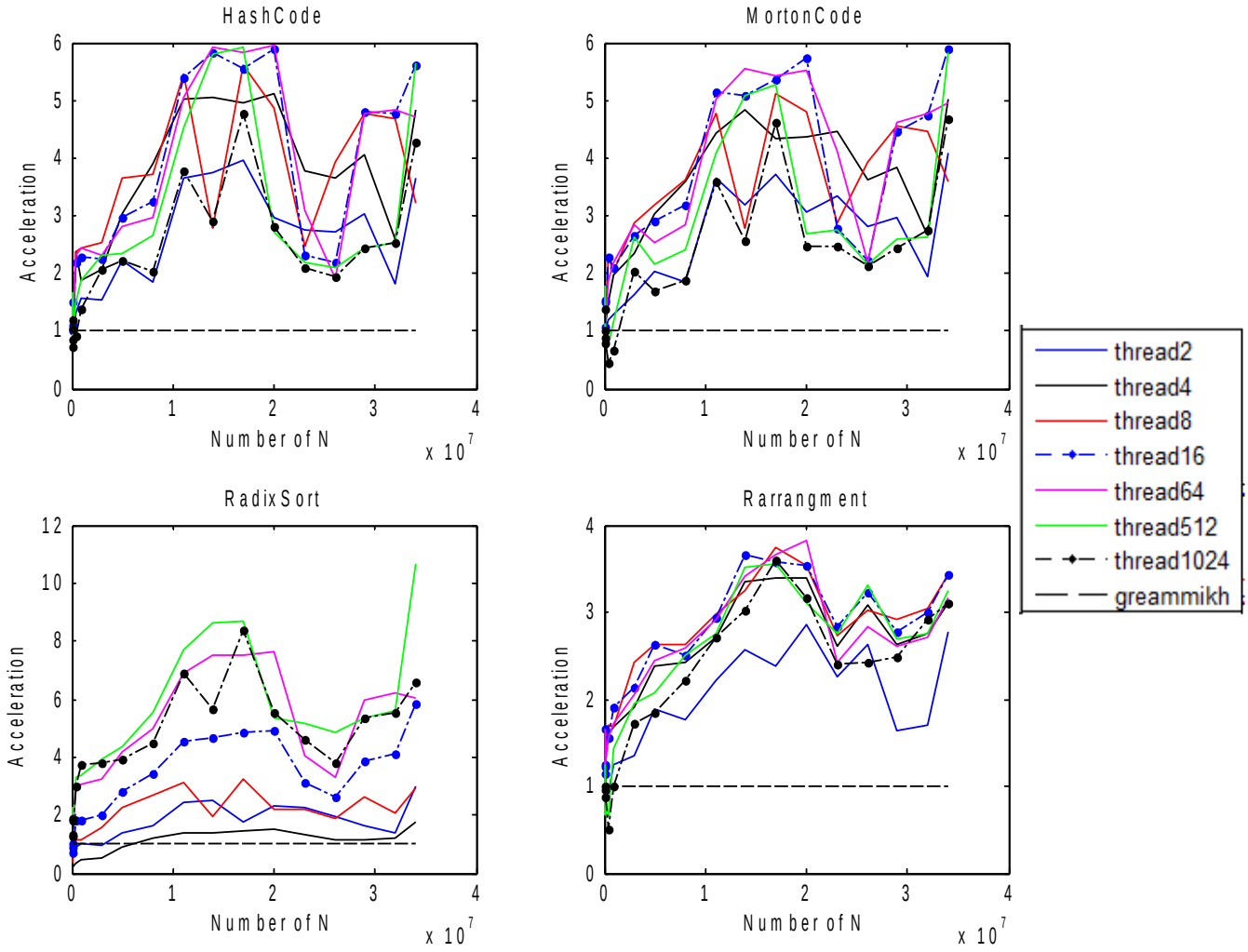
επεξεργαστών,ο χρόνος εκτέλεσης να αρχίζει να χειροτερεύει συμαντικά στην radix λόγο της ανάθεσης αναδρομών σε πολλά νήματα και να γίνεται ακόμα και χειρότερος απο τον σειριακό.Αυτό όμως έχει αποφευχθεί με την εισαγώγη των συνθηκών if για τον αριθμό των σωματιδίων,για να μην ανοίγουν νήματα για μικρές δουλειές.Θα ήταν λάθος σε ένα σύστημα με περιορισμένο αριθμό επεξεργάστων να ανοίγουν απρόβλεπτα συνέχεια νέα νήματα.Επομένως με αυτή την υλοποίηση που έγινε αν βρεθούμε σε ένα άλλο σύστημα ορίζοντας κατάλληλα τις συνθήκες μπορούμε να προσαρμόσουμε το πρόγραμμα πάνω στις απαιτήσεις του συστήματος (αριθμός επεξεργαστών) και να ανοίγουν ακριβώς τόσα νήματα όσα χρειαζονταί.Γενικά στις συναρτήσεις παρατηρούμε μια μικρή διακύμανση στις τιμές(μη γραμμικότητα)των χρόνων για τις οποίες μάλλον ευθύνεται η ΜΗ ισοσταθμισμένη κατανομή του 1/8 κύκλου.
Β)Επιπλέον οι χρόνοι για μικρό αριθμό σωματιδίων για την περίπτωση των παραλληλοποιημένων θα έπρεπε να έιναι χειρότεροι,όμως εχουν οριστεί και σε αυτό το κομμάτι συνθήκες if ώστε τα προγράμματα να εκτελούντε παράλληλα μετά απο κάποιο όριο για τον αριθμό N για το οποίο η παραλληλοποιήση βελτιώνει την απόδοδη του προγράμματος.Ετσι για μικρά N τα παραλληλοποιημένα τρέχουν σειριακά.

Περίπτωση Κύβου:

Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων



Διάγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)

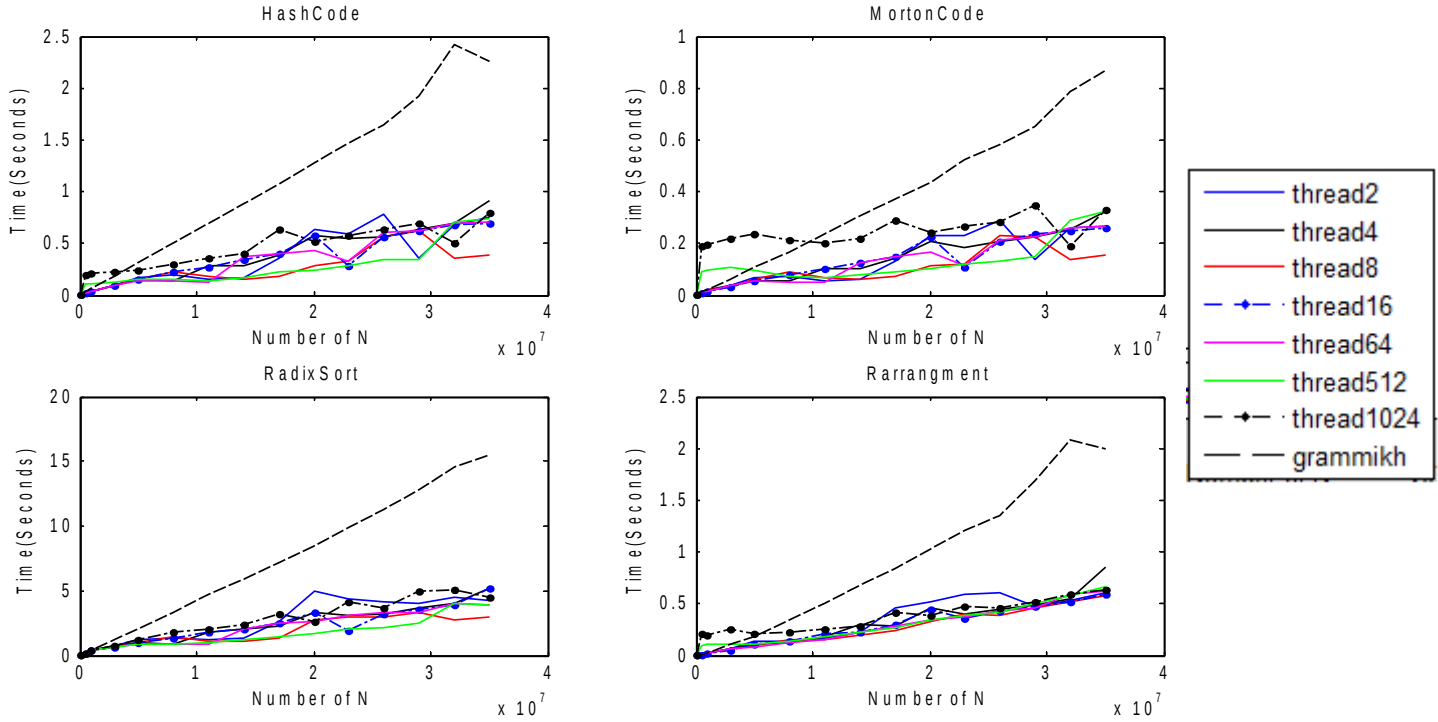


Παρατηρούμε και σε αυτή την περίπτωση σταδιακή μείωση στον χρόνο μέχρι εως ενα οριο και μετά μια μικρή αύξηση(βλέπε διάγραμμα επιτάχυνσης).Οι συναρτήσεις οπως και στην προηγούμενη περίπτωση φαίνεται να μην αποδείδουν καλά για πολυ μεγάλο αριθμό νημάτων,δηλάδη βλεπουμε ότι όσο αυξανεται ο οριθμός των νημάτων αυτό δεν σημαίνει οτι θα αυξάνεται και η απόδοση.Γενικα φαίνεται να προσεγγίζουν ενα κάτω όριο και στην συνέχεια να ξανα ανεβαίνουν.Επιπλέον έχουμε μεγαλύτερη επιτάχυνση όπως φαίνεται απο όλα τα παραπάνω διαγράμματα στο διάστημα μεταξύ 10 με 20 εκατομμύρια σωματίδια.Για την Radix φαίνεται οτι αποδίδει καλύτερα για αριθμό 512 νημάτων ενώ οι υπόλοιπες για την περίπτωση των 64 νημάτων.

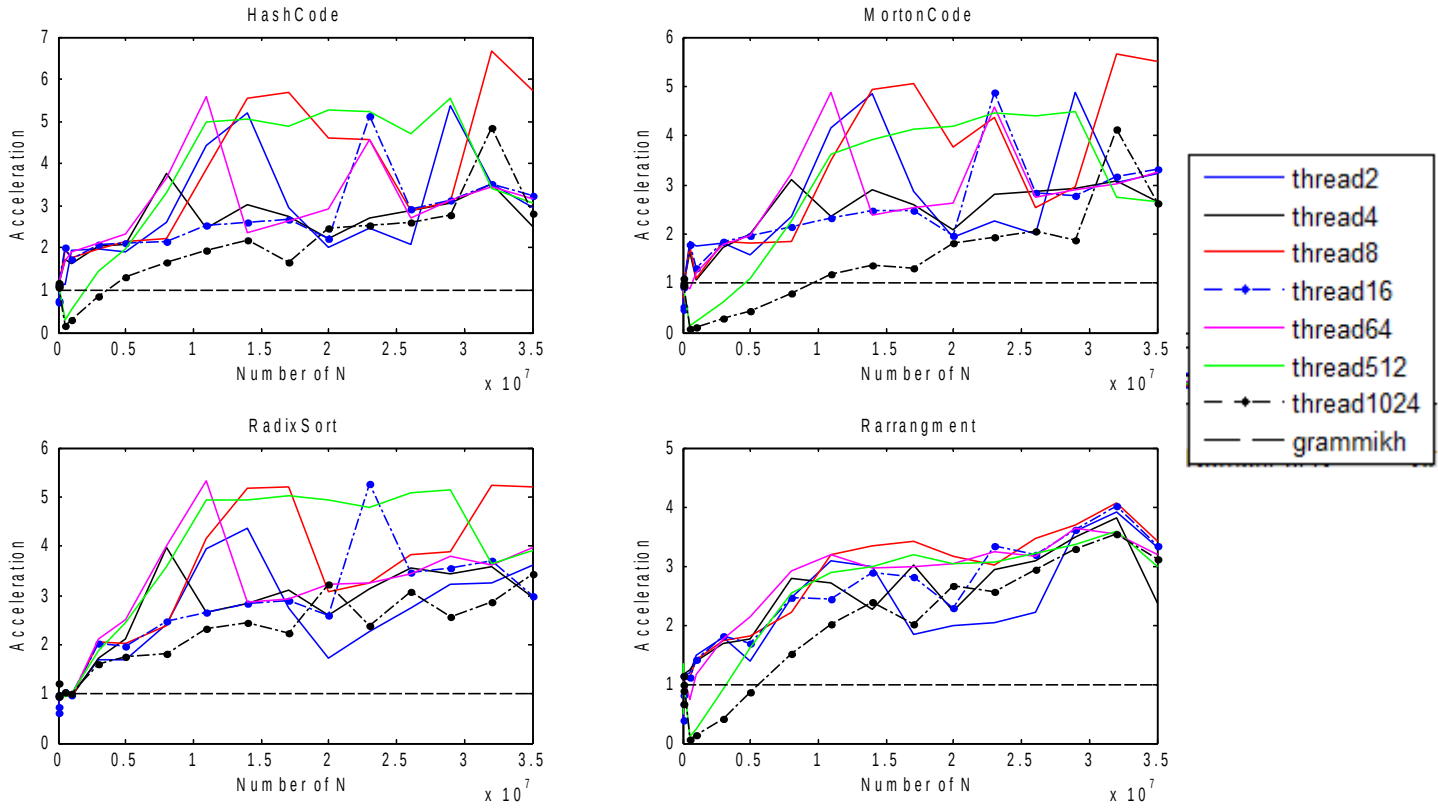
Pthreads

Περίπτωση 1/8 σφαίρας:

Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων

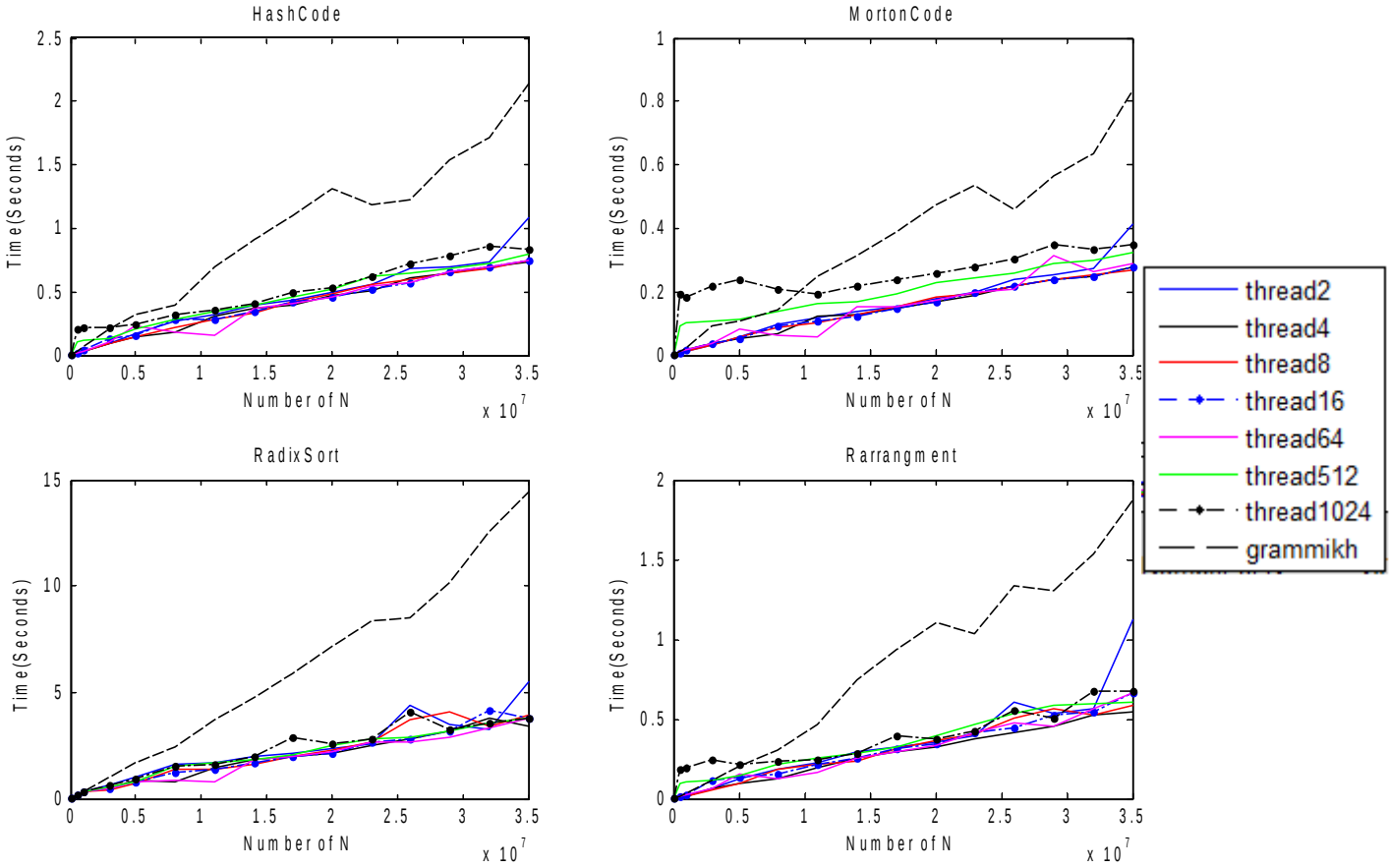


Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)

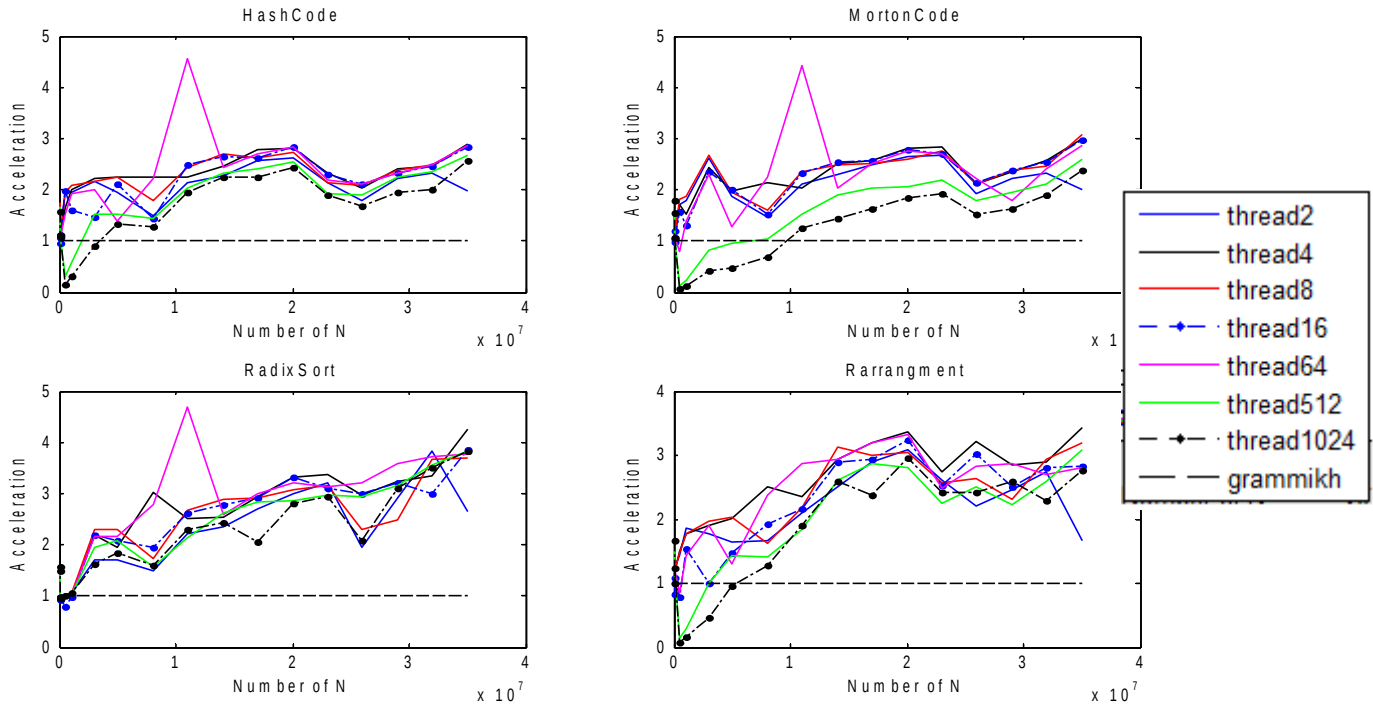


Περίπτωση Κύβου:

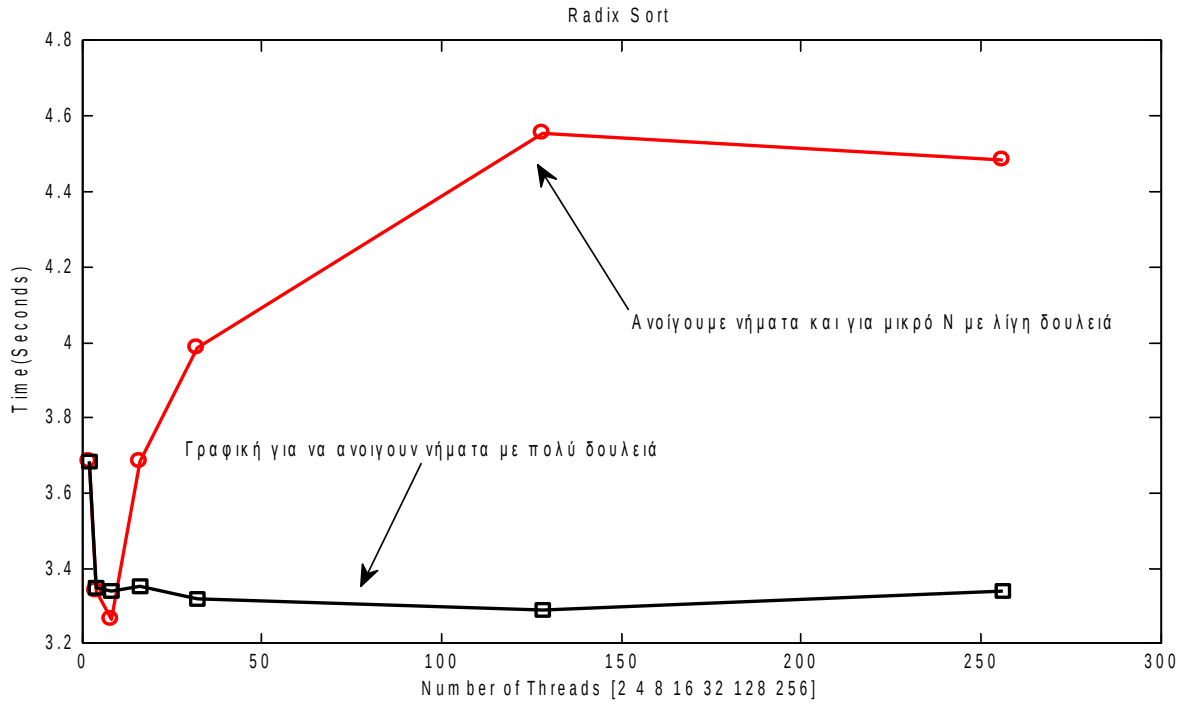
Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων



Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)



Συγρίνοντας την περίπτωση του κύβου και της 1/8σφαίρας στην περίπτωση των Pthread είναι περισσότερο εμφανής η σημασία της τυχαιότητας των αριθμών για τις μετρήσεις και η εξάρτηση τους απο τις αντίστοιχες κατανομές.Στα διαγράμματα επιτάχυνσης τα οπόια εχουν καλύτερη εστίαση στις αριθμητικές τιμές φαίνεται η γραμμικότητα στους χρόνους στην περίπτωση του κύβου σε σχέση με τις τιμές του χρόνου του 1/8σφαίρας οι οποίες εχουν μεγαλύτερη διακύμανση.Να σημειωθεί οτι οι χρόνοι του σειριακού οι οποίοι διαιρούνται με τους αντίστοιχους των παράλληλων για να κάνουμε τα διαγράμματα επιτάχυνσης είναι οι ίδιοι δηλαδή εχουν μετρηθεί μία φορά και χρησιμοποιουνται σε κάθε περίπτωση διοτί διαφορετικά θα είχαμε διαφορετικό σημείο αναφοράς για τις συγκρίσεις και επομένως σφάλματα απο τυχαιότητα στους χρόνους εκτέλεσης του σειριακού.
Απο τα παραπάνω διαγράμματα και σε αυτήν την περίπτωση υπάρχει ένα κάτω όριο στον χρόνο εκτέλεσης.Στην υλοποίση των Hash,Morton και Rearrangment η απόδοδη για πολύ μεγάλο αριθμό νημάτων(δηλαδή για 1024 νηματα)φαίνεται να χειροτερεύει αρκετά και να γίνεται χειρότερη(με όρο επιτάχυνσης) και απο την περίπτωση της OpenMP.Αυτο μάλλον συμβαίνει λόγο του οτι εχουμε ορίσει να δημιουργούνται πίνακες και να γίνονται επαναλλήψεις ανάλογες των αριθμών νημάτων,αλλά οσο αυξάνονται τα νήματα με βάση την υλοποίηση ο κώδικας πρέπει να δημιουργεί περισσότερα δεδομένα(δομές και πίνακες) ,δεσμεύονται θέσεις μνήμης και να αυξάνονται οι επάναλήψεις for που εξαρτώνται απο τον αριθμό νημάτων.Στην περίπτωση της OpenMp που είδαμε προηγουμένως επειδή τα νήματα ανοίγουν αυτόματα όλα μαζί σε μια παράλληλη περιοχή αποφεύγεται η διαδικασία που ακολουθήσαμε για τα Pthreads.Στην περίπτωση της Radix για το κάτω οριο ευθύνονται οι συνθήκες ελέγχου if τον αριθμών των σωματιδίων,επειδή όσα νήματα και να ορίσουμε να ανοίξουν θα ανοίξουν μόνο τα απαραίτητα,θα αλλάξουμε επομένως αυτες τις if (συγκεκριμένα if(N>500)) στο κομμάτι της αναδρομής(δέυτερη if μέσα στην for) ωστε να δημιουργούντε νέα νήμα και για πιό λίγα σωματίδια και θα κάνουμε την γραφική παράσταση χρόνου για σταθερο αριθμό N=32000000 σωματιδίων μεταβάλοντας των αριθμό των νημάτων και θα κάνουμε σύγκριση με την προηγούμενη περίπτωση.



Με βάση το παραπάνω σχήμα βλέπουμε την αξία του να μην ανοίγουν άσκοπα νήματα.Οταν ανοίγουν νεα νήματα για ταξινόμηση σωματιδίων μικρού μεγέθους N αυτά απασχολούνται με κάτι που θα μπορούσε να γίνει πιο γρήγορα σειριακά με αποτέλεσμα οταν το πρόγραμμα βρεθεί μπροστά στην ταξινόμηση πολλών σωματιδίων λόγο έλλειψης νημάτων η εκτελεί γίνεται "σειριακά" χωρίς ανάθεσει σε άλλο νήμα.Επιπλέον βλέπουμε οτι αρχικά ο χρόνος για νήματα αριθμου<=8 είναι ίδιος και στις 2 περιπτώσεις αφου στην πρωτή αναδρομή θα ανόιξουν μεχρι 8 νήματα πιθανόν με μεγάλο αριθμό N ενώ απο εκεί και πέρα ο χρόνος της κόκκινης καμπλύλης θα αυξηθεί και θα προσεγγίσει κάποιο όριο εξαρτώμενο και πάλι απο την νέα συνθήκη if.

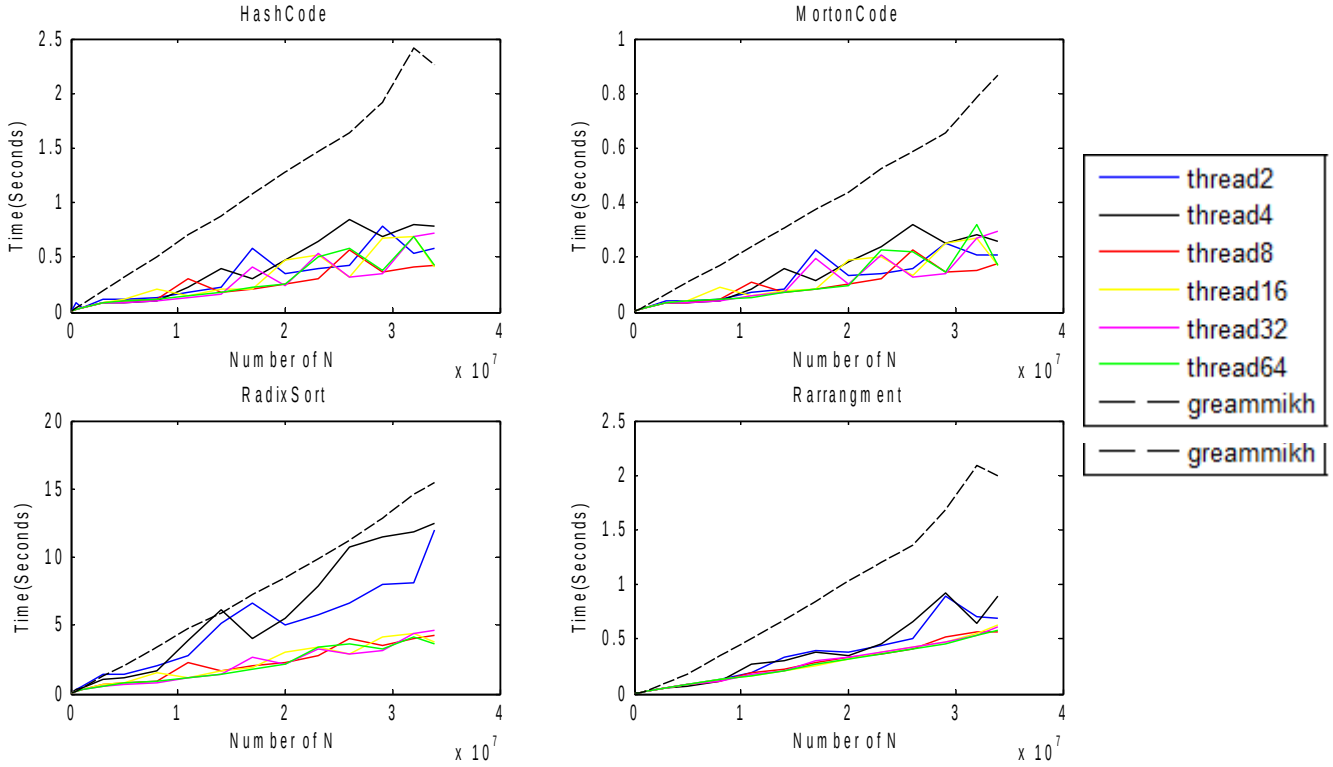
Cilk

Στο κομμάτι της Cilk για το σύστημα μου (4πυρήνο) μπορούσα να ορίσω να ανοίγουν μέχρι 64 νήματα,απο εκεί και πέρα γινόταν αυτόματα ορισμός στα 4 νήματα.Επομένως προσπάθησα να τρέξω στο Δίαδη για περισσότερα νήματα αλλά και πάλι δεν μου επιτρεπόταν να τρέξω περισσότερα απο 128 νήματα(βέπουμε μια σχέση αναλογίας αφου ο Διάδης εχει 8πύρηνο συστημα,δηλαδή διπλάσιοι επεξεργαστές αρα και όριο για διπλάσια νήματα).Τελικά έκανα μετρήσεις και απο τον Δίαδη μαζί με αυτές απο το σύστημα μου για να φανούν τυχόν διαφορές μεταξύ στυστημάτων με διαφορετικό αριθμό επεξεργαστών.

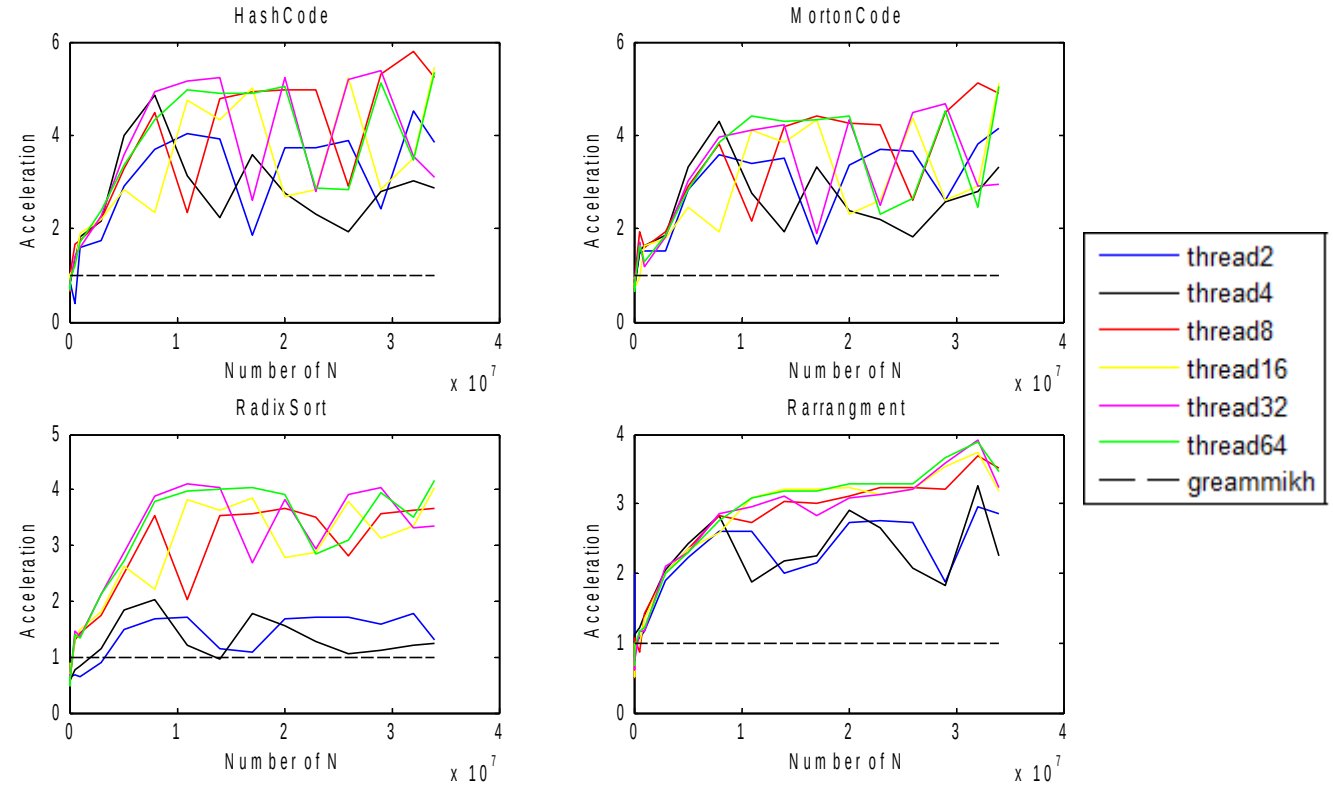
Τερταπύρηνο Σύστημα

Περίπτωση 1/8 σφαίρας:

Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων

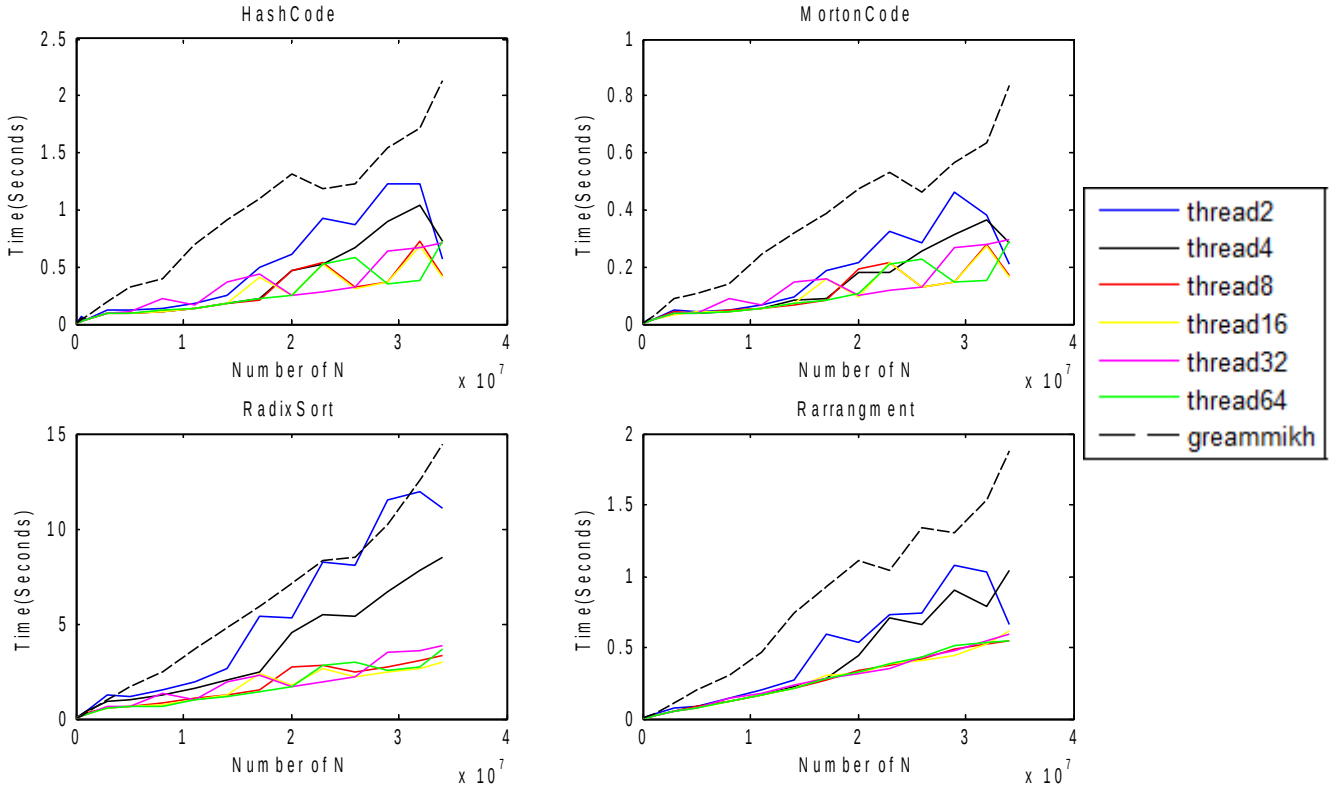


Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)

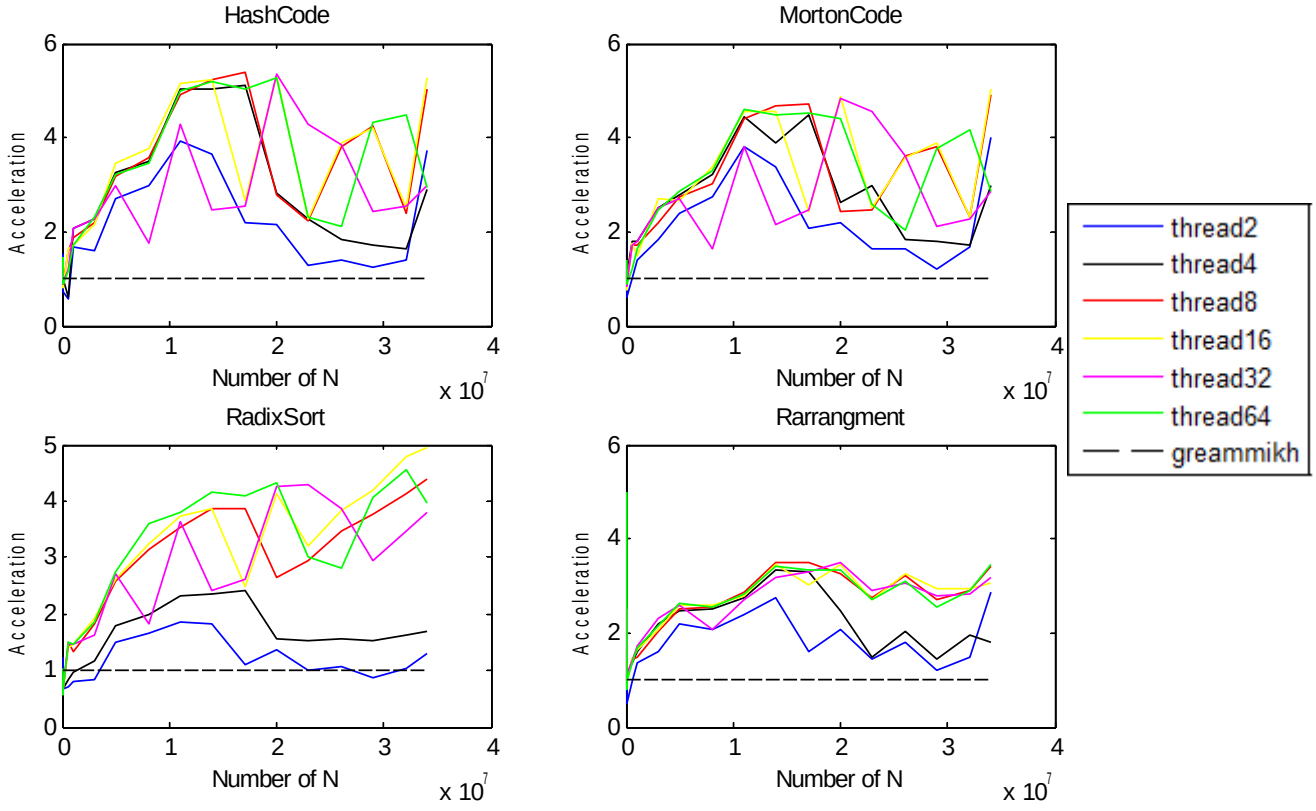


Περίπτωση Κύβου:

Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων

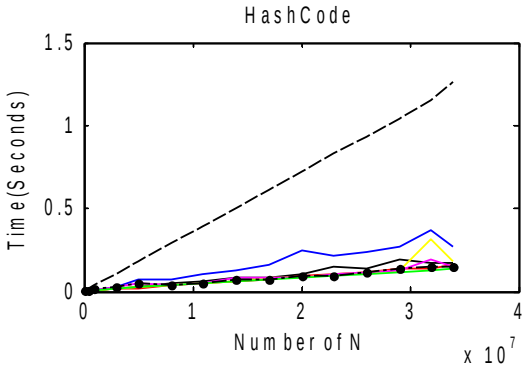


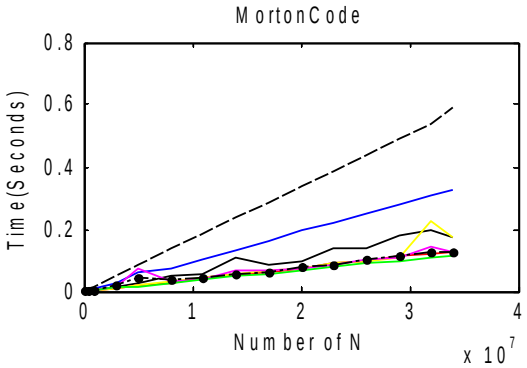
Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)

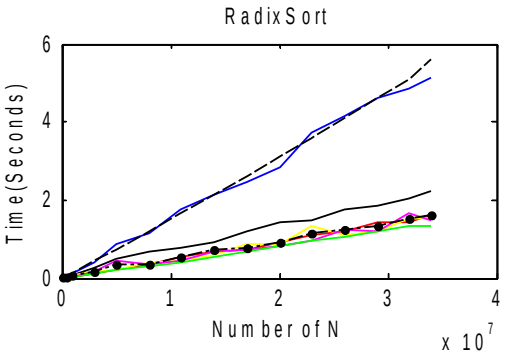


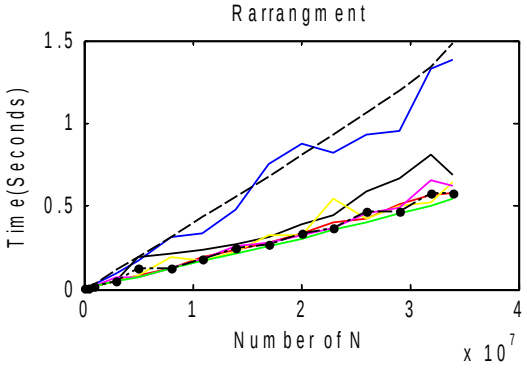
Μετρήσεις απο τον **ΔΙΑΔΗ** για στην περίπτωση του κύβου:

Διαγραμμα Χρόνου-ΑριθμούΣωματιδιων και Αριθμός νημάτων

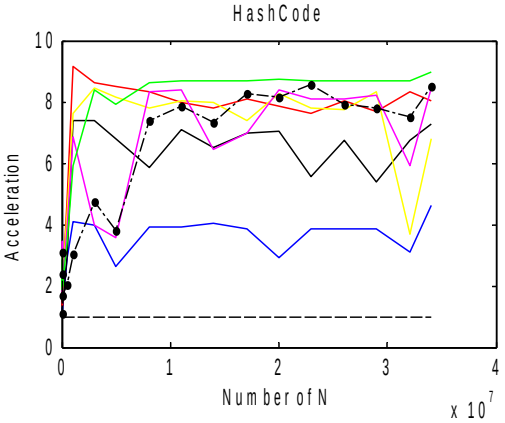


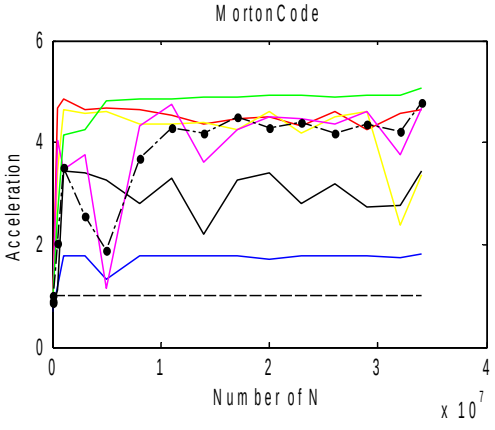


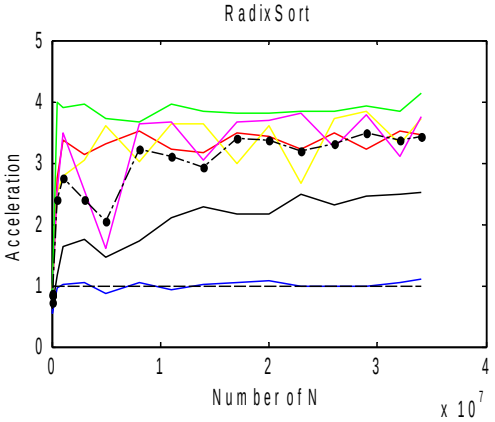


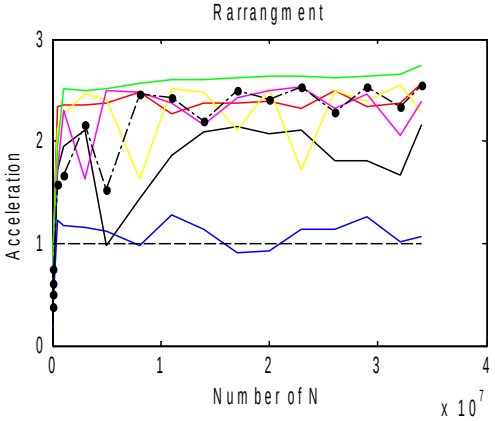


Διαγραμμα Επιτάχυνσης (ΧρόνοςΣειριακού/ΧρόνοςΠαράλληλου)





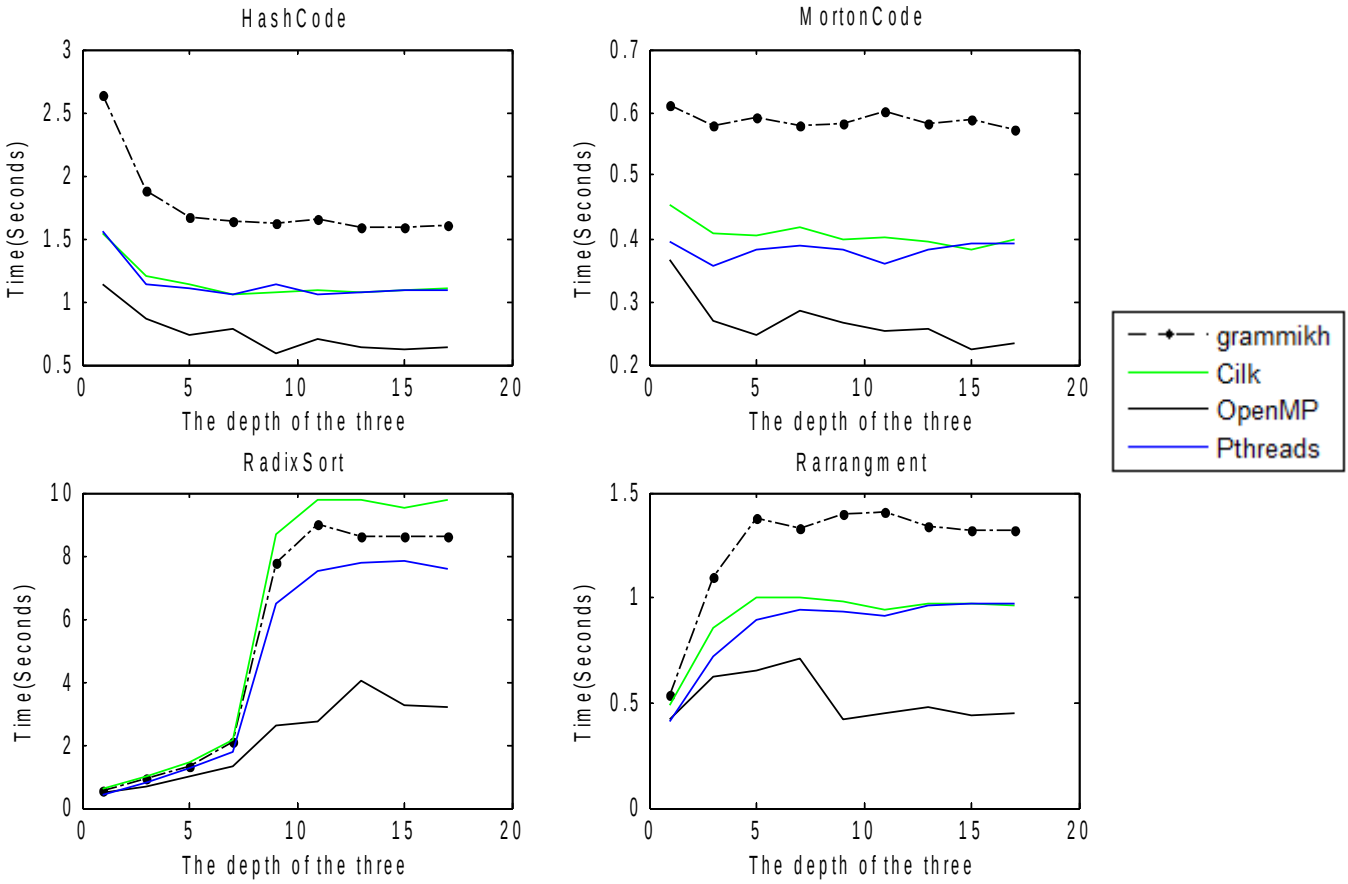




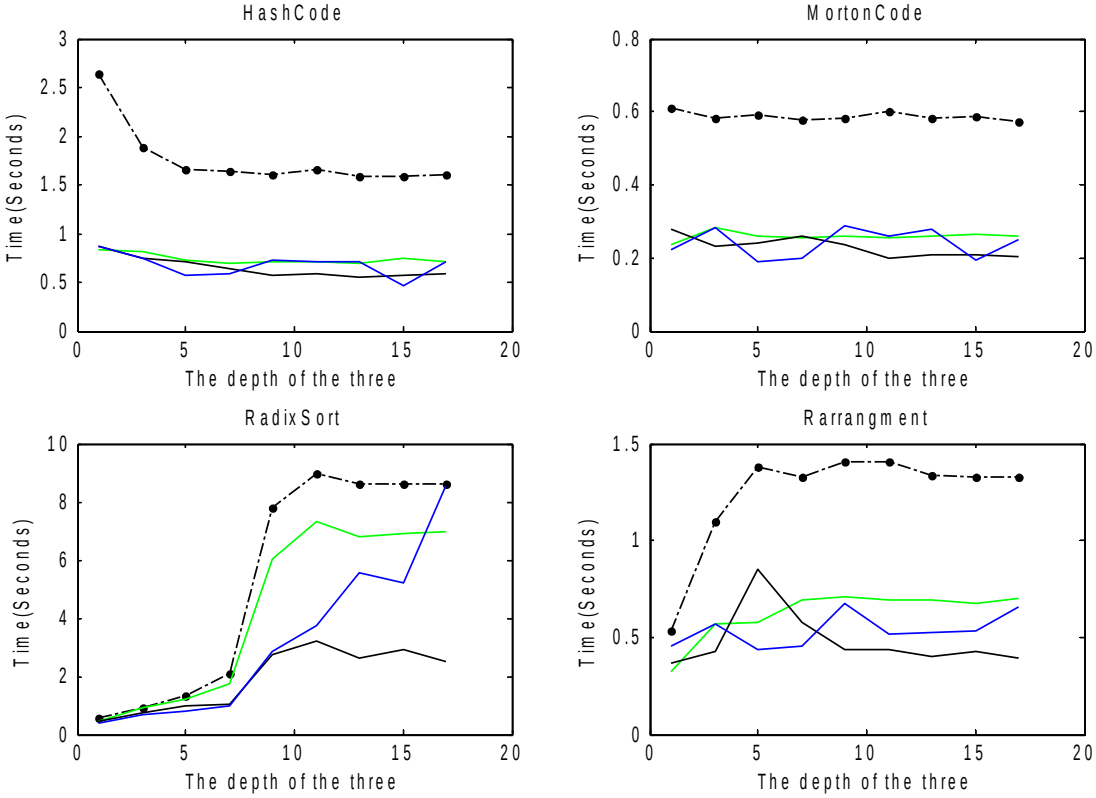
Σε γενικές γραμμές για το πρόγραμμα σε Cilk βλέπουμε οτι έχει παρόμοια συμπεριφορά με αυτη της περίπτωσης της OpenMp,λογικό εξάλλου αφού ακολουθούν και οι δύο υλοποιήσεις την ίδια λογική.Συγκριτικά τώρα με την εκτέλεση του προγράμματος σε διαφορετικά συστήματα φαίνεται απο τα πρώτα διαγράμματα ότι οι χρόνοι εκτέλεσης στο 8πύρηνο συστημα του Διάδη περίπου υποτριπλασιάζονται σε σχέση με το 4πύρινο σύστημα μου.Επιπλέον παρατηρούμε αύξηση της επιτάχυνσης άρα και τις απόδοσης για μεγάλα νήματα στο σύστημα του Δίαδη κάτι που περιμέναμε αφού έχουμε περισσότερους φυσικούς πόρους απο το σύστημα δηλαδή επεξεργαστές και για τα παραλληλοποιημένα προγράμματα αυτο σημαίνει οτι μπορούμε να έχουμε ακριβώς την ίδια στιγμή περισσότερα νήματα να εκτελούν κάποια δουλειά.

Σε αυτό το σημείο θα μελετήσουμε την σχέση/απόδοση των 3 παράλληλων προγραμμάτων μεταξύ τους και με το σειριακό πρόγραμμα για σταθερό αριθμό σωματιδίων ίσο με N=25000000 για διαφορετικό βάθος δέντρου και διαφορετικό αριθμό νημάτων.

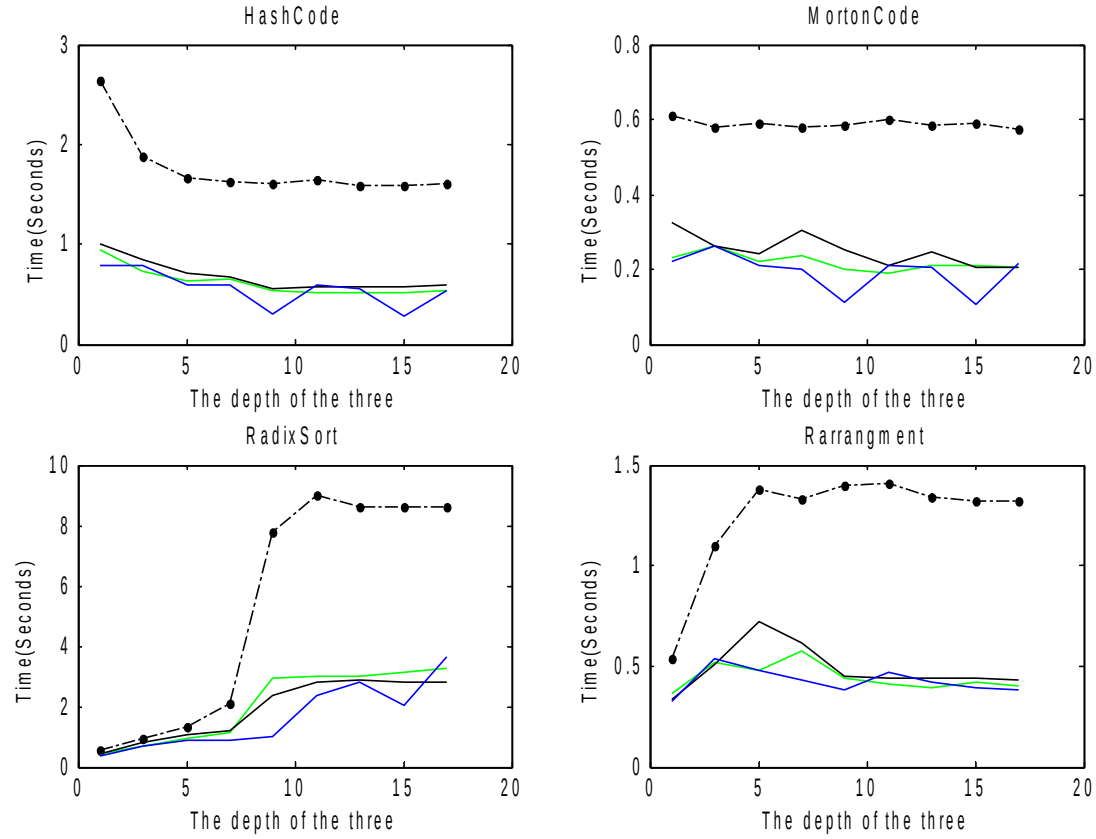
2 Νηματα



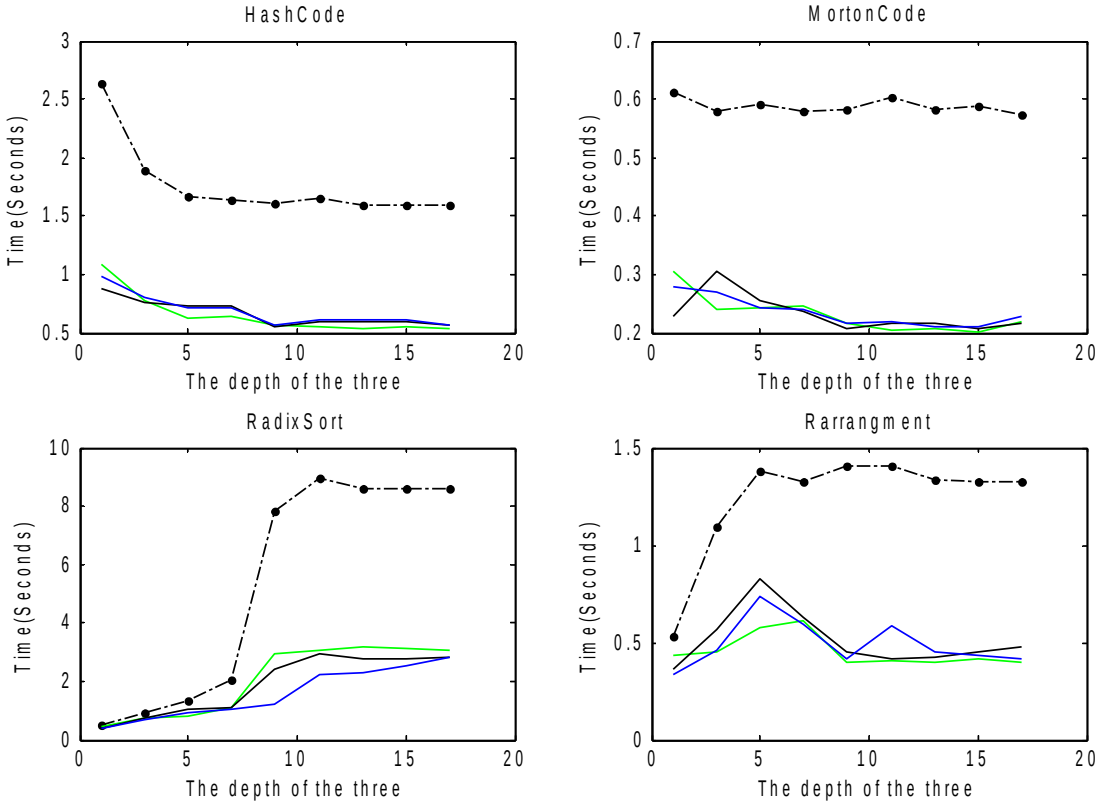
4 Νηματα



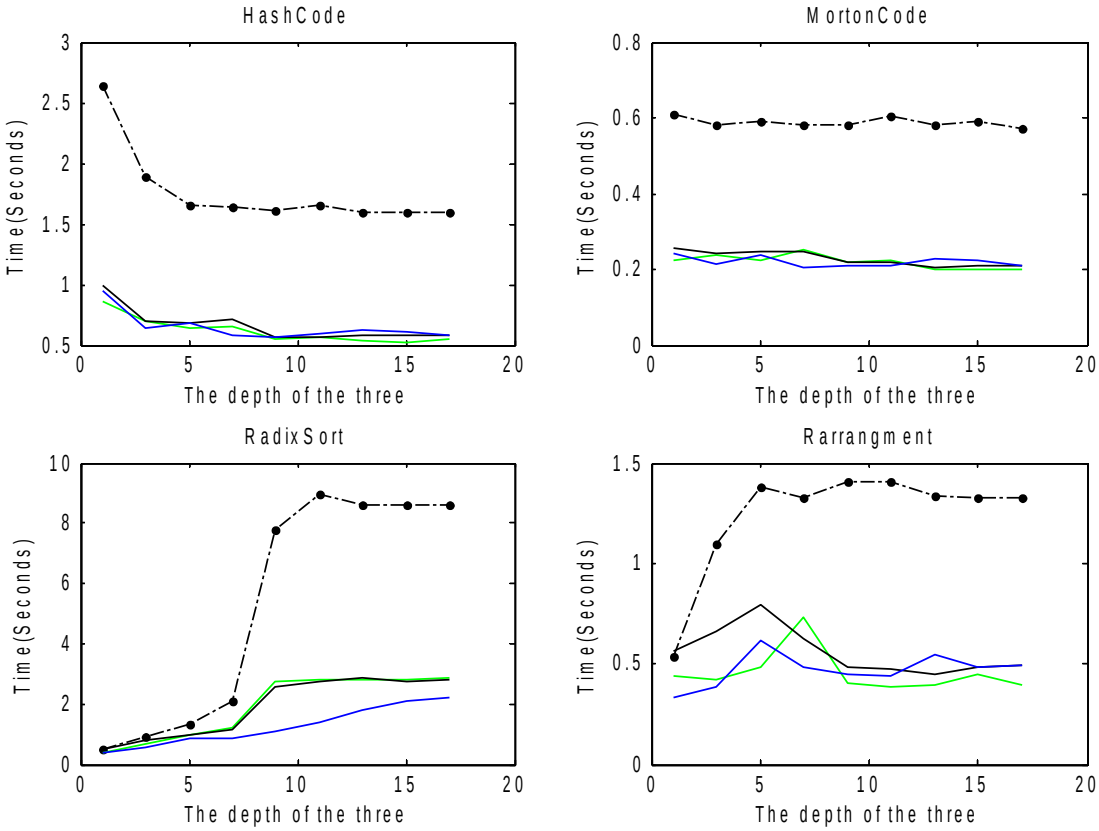
8Νηματα



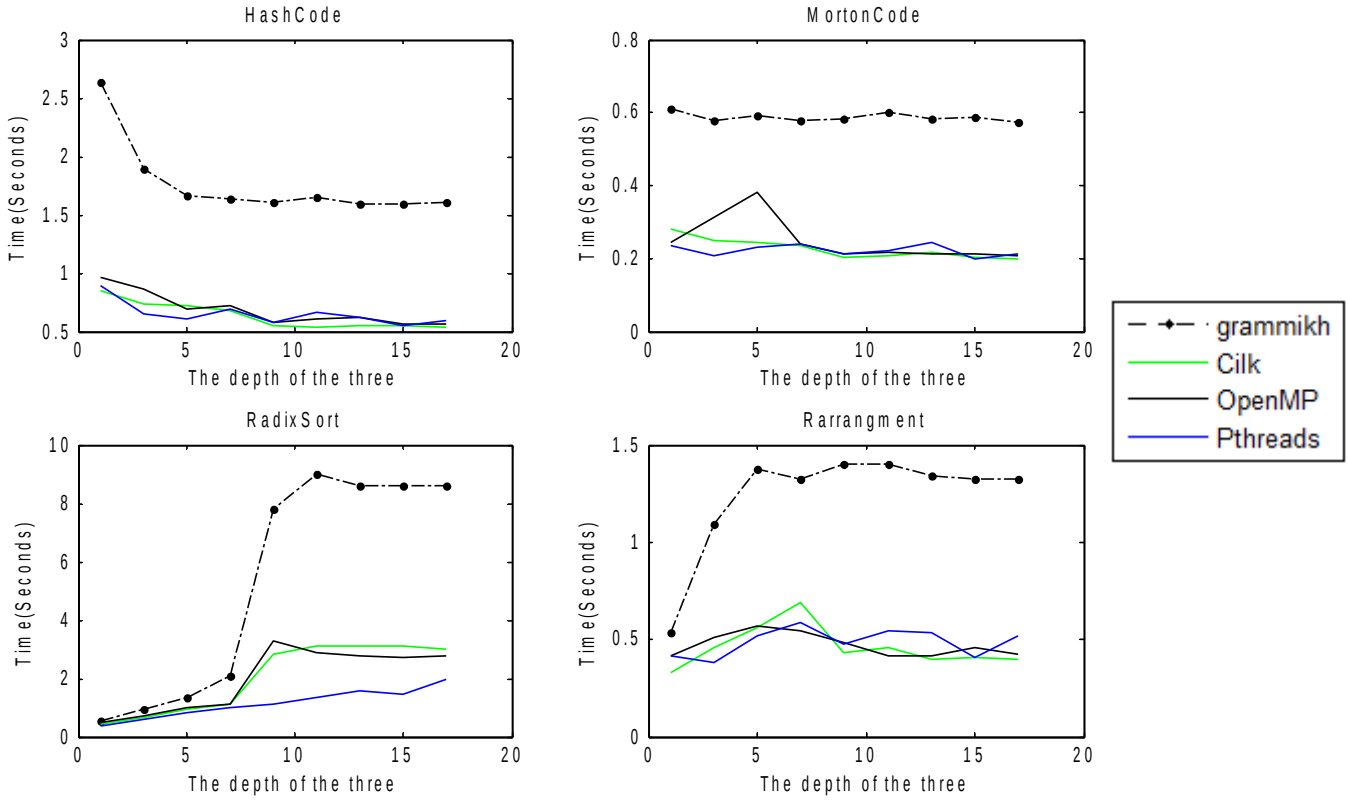
16 Νηματα



32 Νηματα



64 Νηματα



Βλέπουμε οτι αρχικά για μικρό αριθμό νημάτων την καλύτερη απόδοση την παρουσιάζει η OpenMP και την χειρότερη η Cilk για όλες τις συναρτήσεις.Οταν ομως αυξήσουμε τα νήματα για τις συναρτήσεις Hash,Morton και Rearrangment η συμπεριφορά και των τριών υλοποιήσεων είναι παρόμοια ενώ για την περίπτωση της RadixSort η υπολοίηση σε Pthreads δουλέυει πολύ καλύτερα.

Η περίπτωση για τον διαφορετικό αριθμό του πληθυσμού δεν θα επιρεάσει την απόδοση των παραλληλοποιημένων προγραμμάτων επειδή είναι μία συνιστώσα που εισέρχεται μόνο στην RadixSort και ορίζει τον ελάχιστο αριθμό N για τον οποίο σταματάνε οι αναδρομές για έναν αριθμό το πολύ μέχρι το 128.Εμείς με τις συνθήκες if για τόσο μικρό αριθμό σωματιδίων εκτελούμε το πρόγραμμα σειριακά και άρα όσο επιρεάζεται το σειριακό απο αυτή την συνθήκη θα επηρεάστούν αναλογικά και τα παράλληλα προγραμματα.

Τέλος