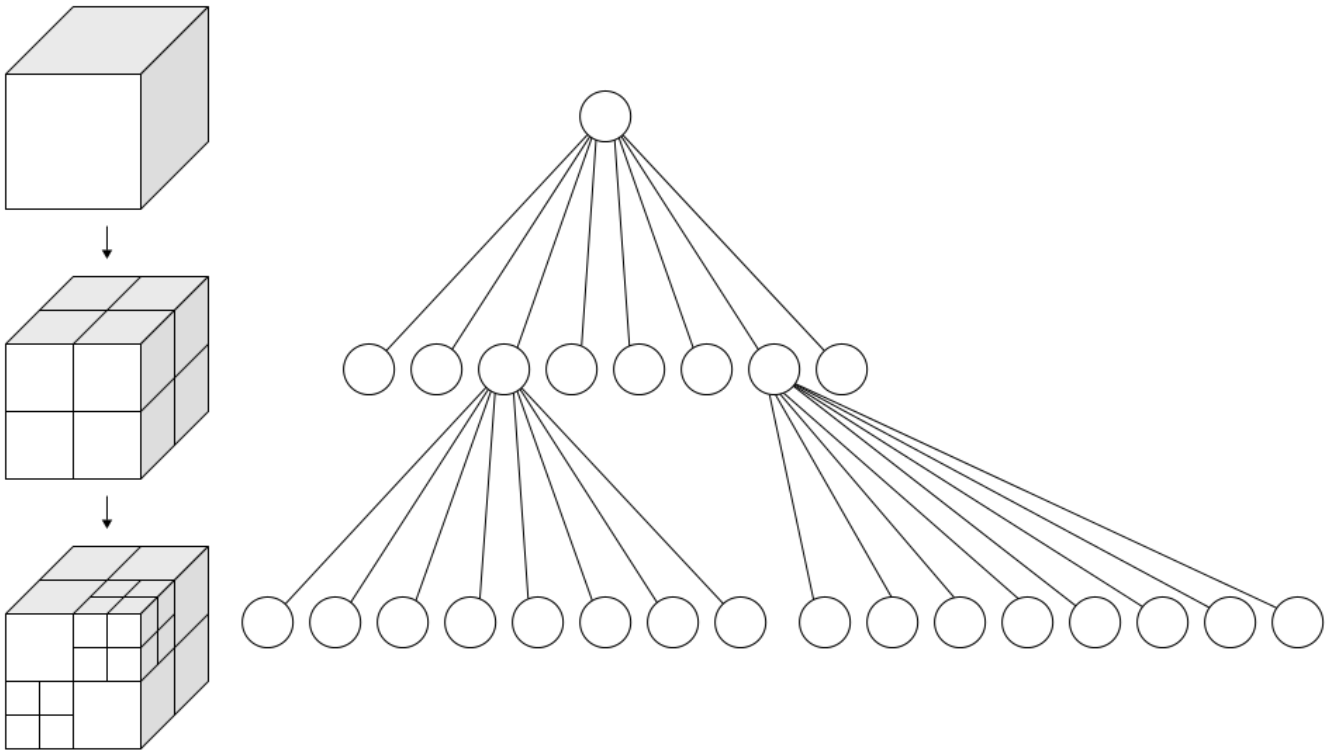


Παραλληλοποίηση αλγορίθμου κατασκευής οκταδικών δέντρων με κωδικοποίηση Morton

Μαμαλάκης Ευάγγελος

mamalakis@auth.gr

ΑΕΜ: 8191

Περιγραφη αλγορίθμου

Ο αλγόριθμος που καλούμαστε να παραλληλοποιήσουμε, αποτελείται από τέσσερα τμήματα, τα οποία βρίσκονται σε διαφορετικά αρχεία και αξιολογούνται ξεχωριστά, οπότε μπορούμε να παραλληλοποιήσουμε ανεξάρτητα το κάθε ένα.

Σύνδεση στο diades

Αρχικά συνδεόμαστε στο diades και τρέχουμε τις παρακάτω εντολές για να σετάρουμε το περιβάλλον.
ssh -p2288 mamalakis@diades.ee.auth.gr
export LC_CTYPE=en_US.UTF-8
source /export/opt/intel/bin/iccvars.sh intel64
chmod 755 -R ~

Για να κάνω mount τα αρχεία μου σε ένα τοπικό φάκελο
sshfs -p2288 mamalakis@diades.ee.auth.gr:/export/home/mamalakis ~/code/diades1
umount -f ~/code/diades1

Υλοποίηση

PThreads

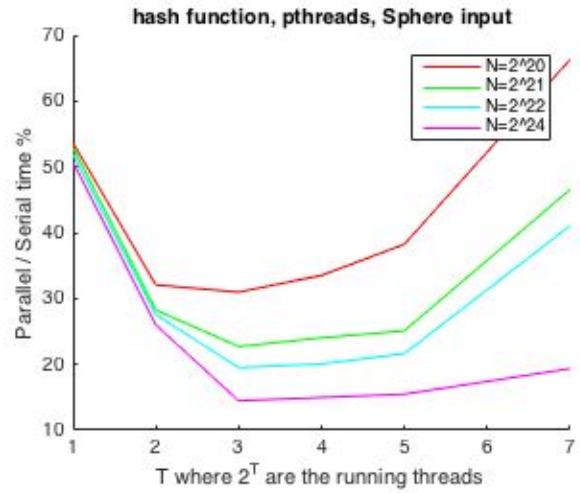
Οι συναρτήσεις hashcode, morton_encoding, data_rearangment είναι εύκολα παραλληλοποιήσιμες γιατί η βασική δουλεία τους βρίσκεται σε μία for όπου ο κάθε βρόγχος είναι ανεξάρτητος. Έτσι η παραλληλοποίηση τους γίνεται με παρόμοιο τρόπο. Συγκεκριμένα, αφού προσδιορίσουμε τον επιθυμητό αριθμό από threads(δηλώνεται σαν extern μεταβλητή για να είναι ορατός σε όλα τα αρχεία κώδικα), χωρίζουμε τη συνολική δουλεία μεγέθους N σε N / nthreads τμήματα. Για κάθε ένα από αυτά προσδιορίζω τη θέση μνήμης από την οποία ξεκινάνε τα αντίστοιχα δεδομένα στα input arrays(offset) και την δίνω σαν όρισμα στο κάθε thread, μαζί με το καινούριο μέγεθος εισόδου N. Στο τέλος χρησιμοποιώ την pthread_join για να σιγουρέψω ότι όλα τα threads τελειώνουν τη δουλειά τους πριν προχωρήσω.

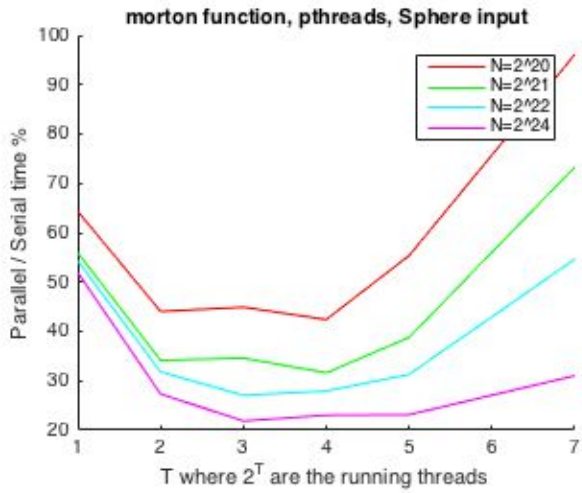
Η συναρτηση radix_sort παραλληλοποιήται πιο περίπλοκα καθώς έχει υλοποηιθεί με αναδρομή. Καταρχήν ορίζουμε μια global μεταβλητή open_threads για να μετράμε πόσα threads είναι ήδη ανοιχτά. Έτσι όταν έχουμε ανοίξει λιγότερα από τα max_threads, ανοίγουμε ένα καινούριο με τη νέα αναδρόμή, αλλιώς η αναδρομή καλείται σειριακά. Αν και υπάρχει αρκετό overhead λόγω της αναδρομής, παρατηρούμε μια βελτίωση της τάξης του 50% για 2^4 threads και μετα μια πολύ μεγάλη αύξηση χρόνο λόγω της επικράτησης του overhead.

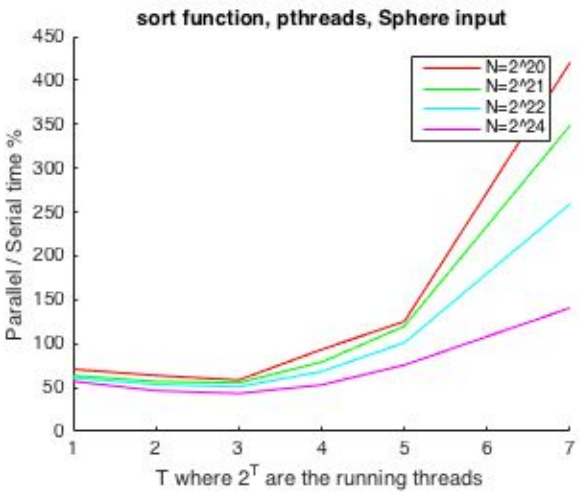
Τεχνική Υλοποίηση

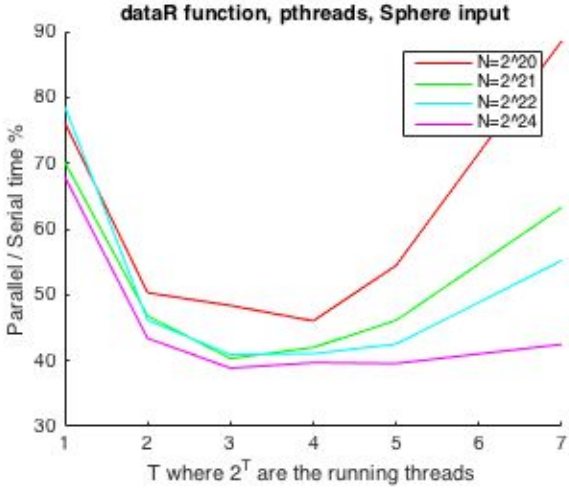
Χρησιμοποιήσαμε τη συνάρτηση pthreads_create() για να φτιάξουμε τα threads. Σαν arguments, δίνουμε τα αρχικά arguments της συνάρτησης, με τροποποιημένα τα arrays ώστε να δείχνουν σε ένα offcet και το N ώστε να αντιστοιχεί ακριβώς στο μέγεθος της προβλεπόμενης δουλείας του thread. Το μέγεθος της δουλειάς καθορίζεται ώς **N / nthreads**. Έτσι το κάθε thread μπορεί να εργαστεί ανεξάρτητα σε ενα τμήμα του input. Στο τέλος κάνουμε join τα threads με τη συνάρτηση pthread_join(). Ως προς το offcet, προσοχή χρειάζεται στο ότι κάποια input arrays έχουν DIM * N στοιχεία οπότε το offcet προσαρμόζεται ανάλογα.

Διαγράμματα και Σχολιασμός

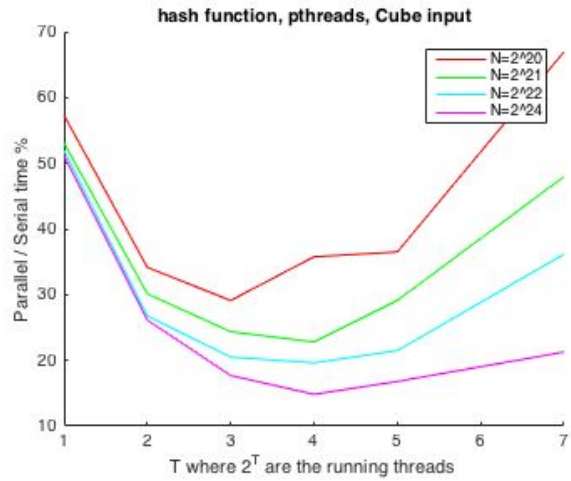


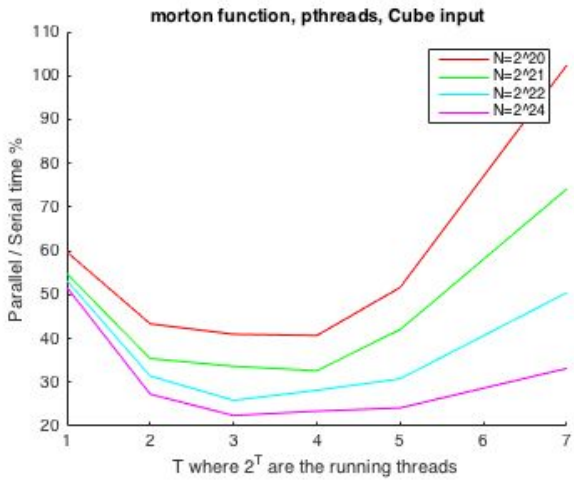


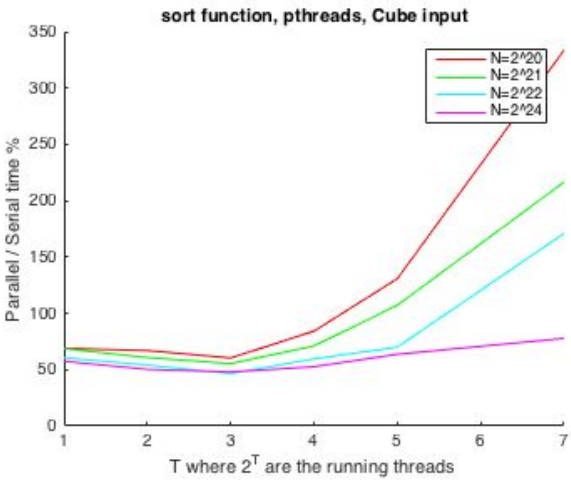


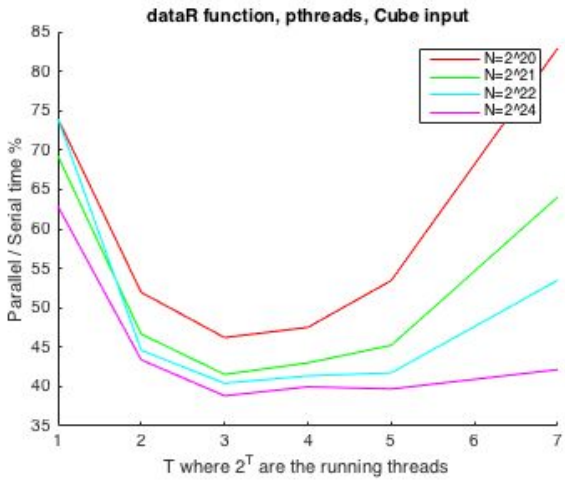


Παρουσιάζουμε σε διαγράμματα για τις 4 συναρτήσεις την επί τις εκατό βελτίωση που παρουσιάζει ο χρόνος του παράλληλου αλγορίθμου σε σχέση με το σειριακό για διάφορες τιμές από threads. Παρατηρούμε ότι ήδη για 2 threads έχουμε μια μείωση του χρόνο κοντά στο 50% η οποία για T=3 (8 threads) φτάνει στο 20% για μεγάλα N. Η συνάρτηση radix παρουσιάζει σχεδον σταθερή μείωση στο 50% λόγω της αναδρομής ενώ η dataR μειώνεται για λίγο μεγαλύτερες τιμές threads. Για τιμες T>4 (threads>16) δηλαδή 2 * 8 που είναι οι πυρήνες του συστήματος παρατηρούμε ότι ο χρόνος αυξάνεται.









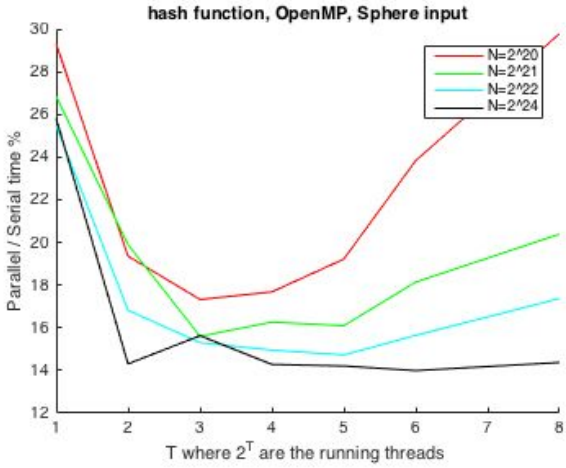
OpenMP

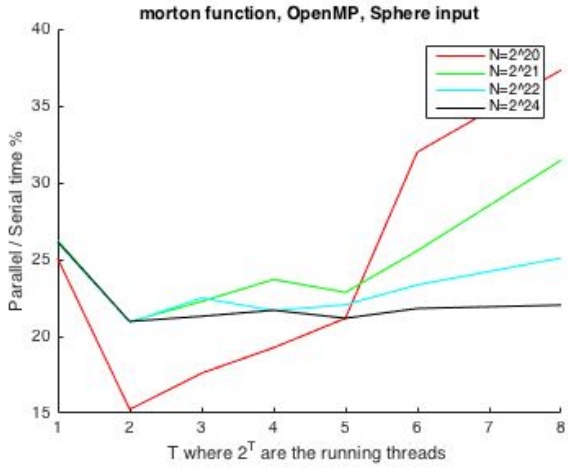
Η παραλληλοποίηση των for loops γίνεται εύκολα με την openmp με χρήση της εντολής **#pragma omp for**. Στην περίπτωση της radix είμαστε αναγκασμένοι να μετράμε χειροκίνητα τα ανοιχτά threads με τον ίδιο τρόπο που το κάναμε στα pthreads.

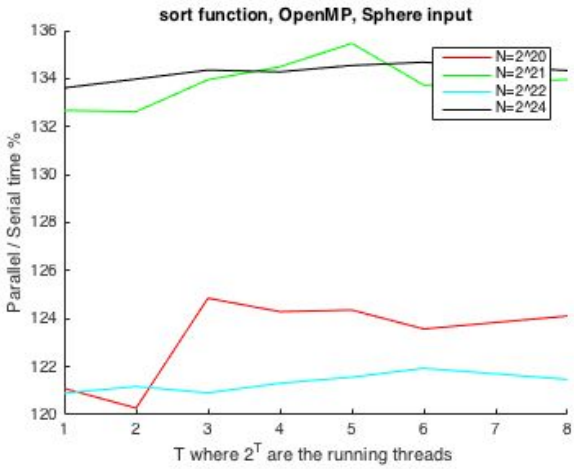
Τεχνική Υλοποίηση

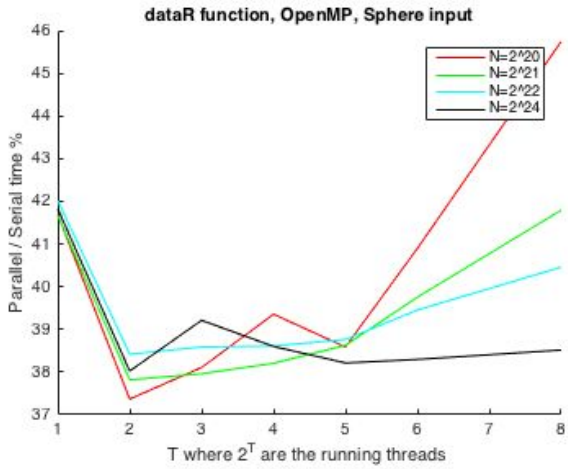
Για να καθορίσουμε τον αριθμό τον threads χρησιμοποιήσαμε τις εντολές:
omp_set_dynamic(0)
omp_set_num_threads(n)

Διαγράμματα και Σχολιασμός

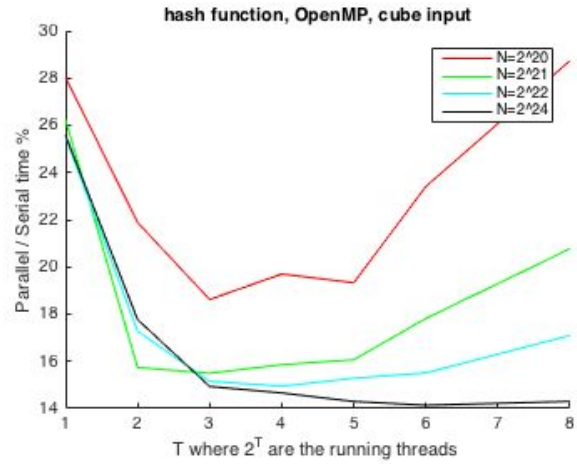


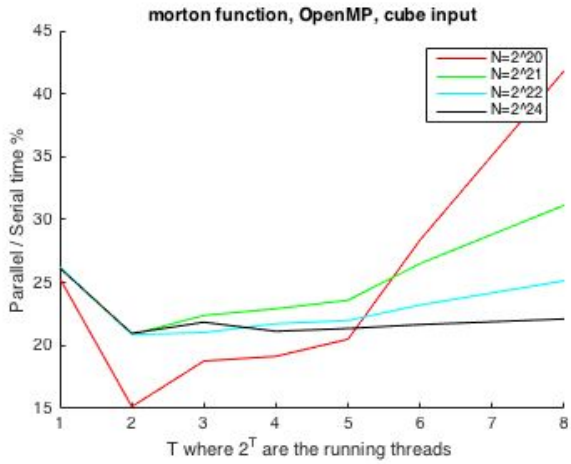


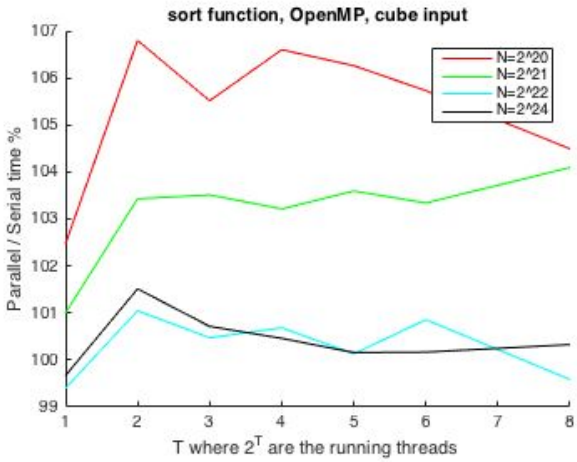


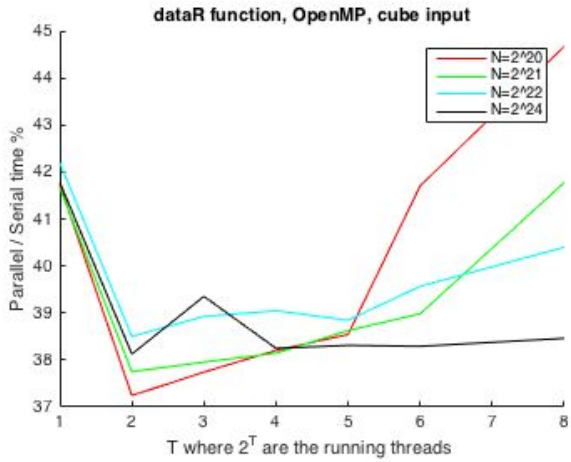


Τα αποτελέσματα που παίρνουμε με την openMP είναι επίσης ικανοποιητικά αφού πετυχαίνουμε μια μείωση του χρόνου στο 25%-40% για 2 threads και για περισσότερα threads φτάνουμε έως και το 14% της σειριακής υλοποίησης. Στα συγκεκριμένα διαγράμματα η παραλληλοποίηση της radix_sort δεν λειτουργεί σωστά καθως δεν περιορίζεται σωστά το άνοιγμα νεων threads με αποτέλεσμα να δημιουργείται overhead που ξεπερνάει την όποια βελτίωση. Στο παραδιδόμενο κώδικα αυτό έχει βελτιωθεί με τη χρήση ενός μετρητή open_threads με τον ίδιο ακριβώς τρόπο που έγινε στα pthreads, όμως δεν ήταν δυνατόν να πάρουμε αξιόπιστες νέες μετρήσεις αφού ο server του μαθήματος ήταν πεσμένος η χρησιμοποιούνταν κάθε φορά που προσπαθήσαμε. Η νέα radix sort πάντως συμπεριφέρεται παρόμοια με αυτή των pthreads.









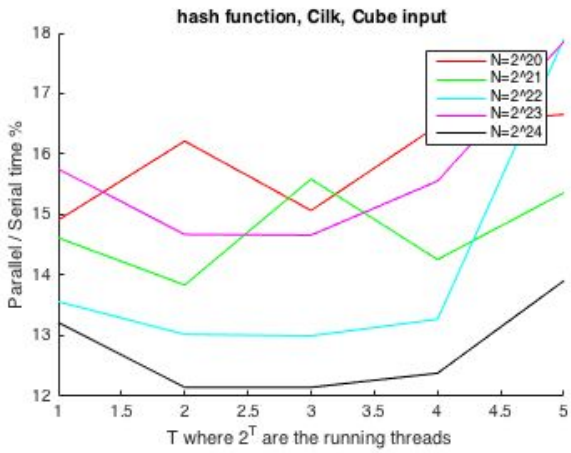
Cilk

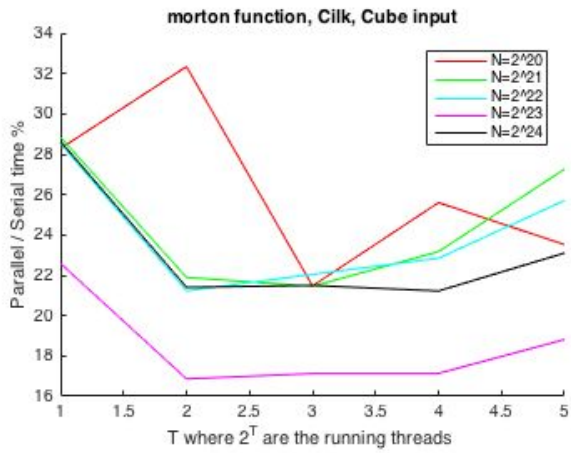
Η παραλληλοποίηση με τη cilk είναι άμεση και ταυτοχρονα αρκετά αποδοτική. Μπορούμε απλά να αντικαταστήσουμε τις κατάλληλες δομές for με cilk_for και η cilk αναλαμβάνει τα υπόλοιπα. Παρακάτω βλέπουμε τις μετρήσεις που πήραμε που είναι πολύ ικανοποιητικές.

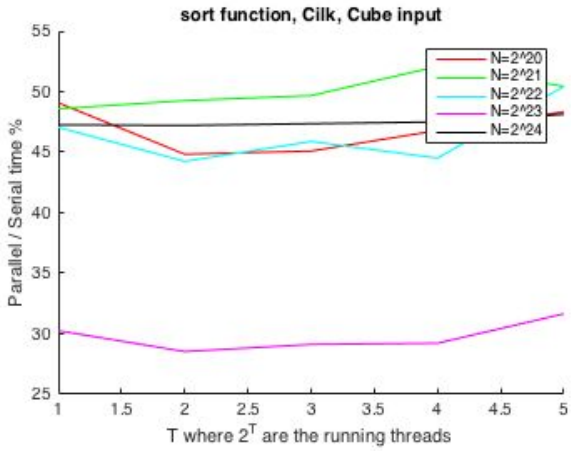
Τεχνική Υλοποίηση

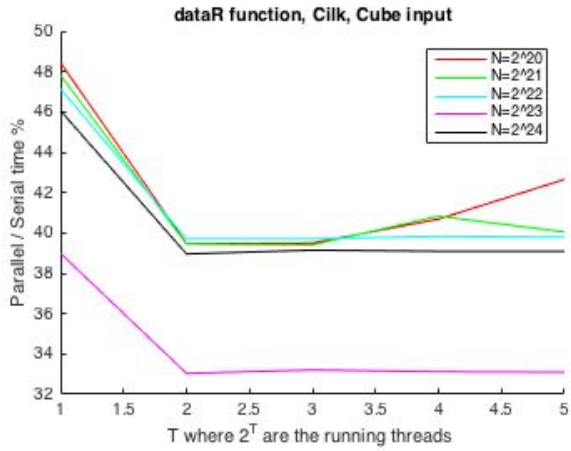
Για να θέσουμε τον συνολικό επιθυμητό αριθμό threads χρησιμοποιούμε την εντολή
__cilkrts_set_param("nworkers", nthreads);

Διαγράμματα και Σχολιασμός

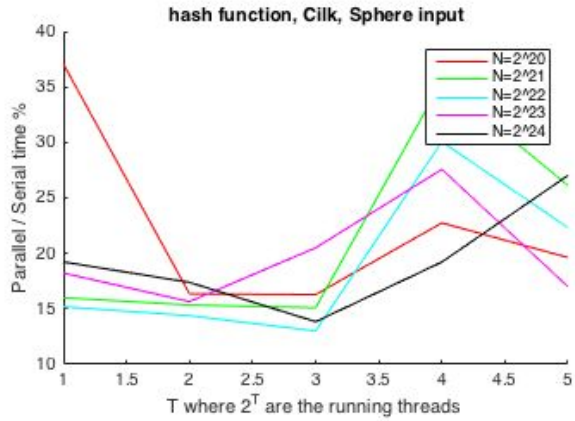


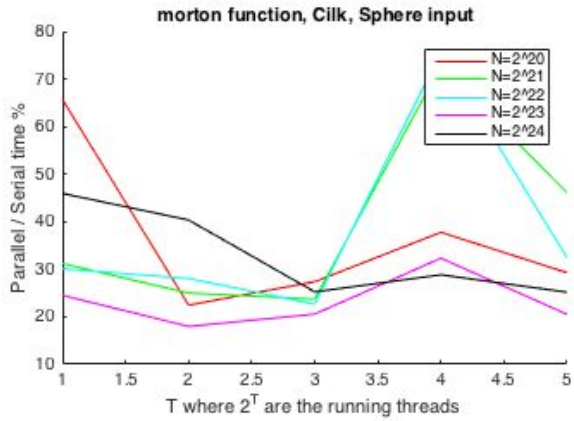


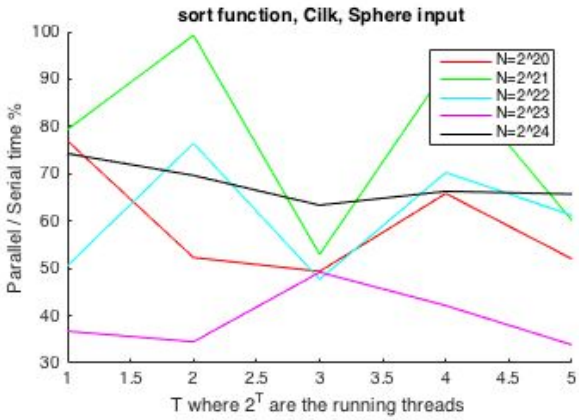


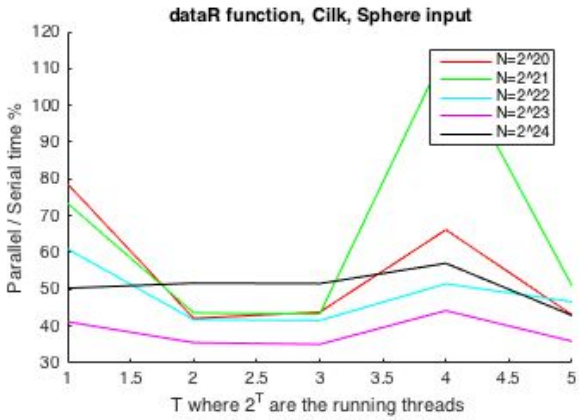


Βλέπουμε ότι η cilk πετυχαίνει πολύ καλή μείωση του χρόνου και συγκεκριμένα για τις συναρτήσεις hash και morton πλησιάζει το 100/8=12.5% του σειριακού χρόνου εκτέλεσης που είναι η θεωρητικά καλύτερη δυνατή βελτίωση. Στη radix η παραλληλοποίηση της cilk είναι περίπου 50% για τις διάφορες τιμές του T δηλαδή φαίνεται πως δεν
πετυχαίνουμε κάποια βελτίωση στον παραλληλισμό της αναδρομής για περισσότερα από 2 Threads. Τα διαγράμματα της sphere βασίζονταν σε μετρήσεις που έτρεξαν στο Διάδη ενώ έτρεχαν κι άλλοι συνάδελφοι μετρήσεις με αποτέλεσμα να παρατηρείται διακύμανση και μικρότερη βελτίωση του χρόνου εκτέλεσης σε σχέση με το σειριακό.









Παρουσίαση με Matlab

Για την παρουσίαση των διαγραμμάτων στη Matlab τροποποιήσαμε κατάλληλα την δοσμένη συνάρτηση read_data προσθέτοντας τον αριθμό τον threads και υλοποιήσαμε μια νέα συνάρτηση plot_data η οποία αναλαμβάνει να δημιουργήσει τα διαγράμματα για τις διάφορες τιμές του N και του T χρησιμοποιούντας και τη συνάρτηση subplot για να απεικονίσει τις 4 συναρτήσεις.

Βιβλιογραφία

https://en.wikipedia.org/wiki/OpenMP#Pros_and_cons
http://stackoverflow.com/questions/14082360/pthread-vs-intel-tbb-and-their-relation-to-openmp
https://en.wikipedia.org/wiki/Z-order_curve
https://software.intel.com/en-us/forums/intel-cilk-software-development-kit/topic/286954
https://amaral.northwestern.edu/resources/guides/mounting-remote-folder-os-x-over-ssh
https://software.intel.com/en-us/node/522586