4η Εργασία – Παράλληλα και Διανεμημένα Συστήματα

Ραφαήλ Μπουλογεώργος 9186 – rafampou@gmail.com

Η εργασία βρίσκεται στο Github: **https://github.com/rafampou/PSD_ex4**
download link: **https://github.com/rafampou/PSD_ex4/archive/master.zip**

ΘΕΜΑ 1ο : Παράλληλη υλοποίηση του Vantage-Point tree

Υλοποίηση του VP-Tree στην Cuda

Η υλοποίηση του vantage point tree βασίστηκε στην πρώτη εργασία αλλά διαφέρει αρκετά εξαιτίας της ανάγκης για παραλληλοποίηση. Επιπλέον η λογική της αναδρομής δεν λειτουργεί στην CUDA (τουλάχιστον για τις κοινές κάρτες γραφικών που έχουμε).

Το πρόβλημα μελετήθηκε ως εξής:

Βήμα 1ο: Υπολογισμός των αποστάσεων όλων των σημείων από την ρίζα (από εδώ και πέρα ορίζουμε ως «head») που ανήκουν.
Βήμα 2ο: Για κάθε head εύρεση του μεσαίου στοιχείου και «σειριακή ταξινόμηση» των στοιχείων που ανήκουν στο ίδιο head (ρίζα)
Βήμα 3ο: Επανάληψη για όλα τα επίπεδα. Δηλαδή deep φορές όπου $deep = \frac{\log(n)}{\log(2)}$ με n ο αριθμός των σημείων.

Για τα βήματα 1,2 υλοποιήθηκαν διάφοροι εναλλακτικοί αλγόριθμοι που περιέχονται με μορφή σχολίων, αλλά εδώ θα αναλύσω σύντομα τους πλέων αποτελεσματικούς.

Για την περιγραφή της λύσης πρέπει να εξηγήσουμε την σημασία των πινάκων που χρησιμοποιήσαμε

Συνολικά χρησιμοποιήθηκαν 5 πίνακες:

1. dataArr: (double) διαστάσεις n x d και περιέχει όλα τα σημεία
2. X_indexes: (int) διάστασης n, και περιέρχει τα indexes των σημείων.
3. X_heads: (int) διάστασης n, περιέχει την θέση της ρίζας (head) κάθε σημείου εάν είναι θετικός ή εάν είναι αρνητικός περιέχει την θέση του σημείου στο δένδρο σε ενδοδιατεταγμένη μορφή. (Παράδειγμα παρακάτω)
4. Distances: (double) διάστασης n, περιέχει τις αποστάσεις των σημείων από την ρίζα τους.
5. array_n_dev: (int) διάστασης n, περιέχει των αριθμό των σημείων στο υποδέντρο που ανήκει κάθε σημείο.

Υπολογισμός αποστάσεων

Οι αποστάσεις υπολογίζονται με την global συνάρτηση distanceWithIndexesGPU στο αρχείο eucleidean_distance.cu. Δέχεται ως όρισμα τον πίνακα των σημείων, τα indexes, την ρίζα κάθε σημείου, τον πίνακα επιστροφής, τον αριθμό των σημείων και την διάσταση.

Ο υπολογισμός κάθε σημείου γίνεται από ένα thread με βάση την ευκλείδεια απόσταση. Αυτό που διαφέρει είναι ότι για κάθε σημείο η απόσταση υπολογίζεται όχι από το τέλος του πίνακα αλλά από το σημείο που δηλώνει ο πίνακας head, δηλαδή από την ρίζα του. Έτσι παράλληλα ανεξάρτητα εάν τα σημεία ανήκουν σε διαφορικά δέντρα υπολογίζονται όλες οι αποστάσεις.

Εάν η ρίζα δείχνει σε αρνητικό αριθμό δεν υπολογίζεται καμιά απόσταση για το σημείο αυτό.

Ταξινόμηση – Διαχωρισμός

Στην συνέχεια η συνάρτηση my_sort στο αρχείο my_select.cu υπολογίζει και ταξινομεί τα στοιχεία.

Αρχικά από τον πίνακα array_n υπολογίζει που βρίσκεται το μεσαίο στοιχείο και στην συνέχεια κάθε thread μετράει με τον counter *lower* πόσα σημείο έχουν μικρότερη ή ίση απόσταση από αυτό. Το thread που θα έχει lower == low_num = n - 1 - (n-1)/2 όπου n ο αριθμός των σημείων μαζί με την ρίζα, είναι το μεσαίο στοιχείο. Αυτό το thread αναλαμβάνει την ταξινόμηση για το σύνολο των σημείων που ανήκουν στο ίδιο υποδένδρο. Ο διαχωρισμός των σημείων γίνεται με βάση το head, δηλαδή τα σημεία με ίδια ρίζα ανήκουν στο ίδιο υποδέντρο.

Η ταξινόμηση δεν διαφέρει από την σειριακή υλοποίηση. Εδώ πρέπει αν τονίσω ότι η απόδοση του αλγορίθμου θα ήταν καλύτερη για μεγάλα n εάν παραλληλοποιήσαμε και την διαδικασία αυτή.

Στην παρούσα υλοποίηση παραλληλοποιούμε μόνο την κατασκευή των υποδέντρων. Επομένως για την ρίζα του δέντρου ο αλγόριθμος εκτελείται σχεδόν σειριακά αλλά σε βάθος 10 εκτελούνται $2^{10} = 1024$ παράλληλες ταξινομήσεις με πολύ μικρό αριθμό σημείων.

Για την συνέχεια του αλγορίθμου κατά την ταξινόμηση εκτός από τα indexes αλλάζουμε τις ρίζες και τις αποστάσεις. Η ρίζα της μικρότερης απόστασης γίνεται 2φορές το index της ρίζα του σημείου δηλαδή το inner, ενώ το σημείο με την μέση απόσταση γίνεται το outer και η ρίζα είναι 2φορές το index +1. Δηλαδή ακολουθούμε την ενδοδιατεταγμένη αρίθμηση για ένα δυαδικό δέντρο.

Όπως έγινε σαφές όλο το δέντρο υλοποιείται στην GPU αλλάζοντας την θέση των indexes και ορίζοντας στον πίνακα heads την θέση κάθε σημείου στο δέντρο.

Ακολουθεί ένα δείγμα για 5 τυχαία σημεία όπως εκτυπώθηκαν από τον αλγόριθμο

Αρχικά δεδομένα

i	idx	head	n	distance	point	
0.	4	-1	5	0.000	13675.806	9738.971
1.	3	0	5	0.000	84427.438	8571.144
2.	2	0	5	0.000	24842.250	32745.551
3.	1	0	5	0.000	73995.647	48.932
4.	0	0	5	0.000	62467.515	8787.282

Ορίζουμε τα idx ανάστροφα ώστε να ταιριάζει το την σειριακή υλοποίηση όπου η ρίζα ήταν το τελευταίο στοιχείο του πίνακα.

Πρώτη επανάληψη

1η Επανάληψη - distanceWithIndexesGPU						
0.	4	-1	5	0.000	13675.806	9738.971
1.	3	0	5	70761.270	84427.438	8571.144
2.	2	0	5	25573.272	24842.250	32745.551
3.	1	0	5	61093.208	73995.647	48.932
4.	0	0	5	48800.990	62467.515	8787.282
1η Επανάληψη - my_sort						
0.	4	-1	5	48800.990	13675.806	9738.971
1.	2	-2	2	0.000	24842.250	32745.551
2.	0	1	2	0.000	62467.515	8787.282
3.	3	-3	2	0.000	84427.438	8571.144
4.	1	3	2	0.000	73995.647	48.932

Βλέπουμε ότι υπολογίστηκαν όλες οι αποστάσεις από τα σημεία 3,2,1,0 ως προς το 4.

Και σύμφωνα με τις αποστάσεις παραπάνω ταξινομήθηκαν στην θέση 1 το μικρότερο και στην 4 το μεγαλύτερο. Η απόσταση για την ρίζα ορίστηκε ως η μέση απόσταση δηλαδή το median.

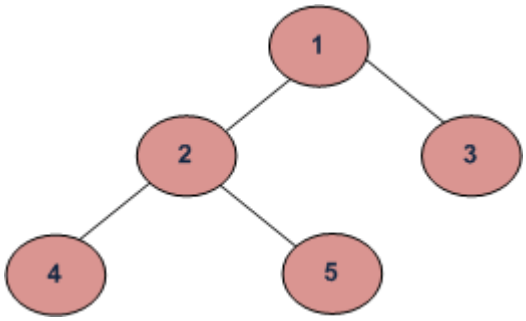
Οι τα 2 νέα δέντρα έχουν τώρα ρίζες τα σημεία 2,3.

Δεύτερη επανάληψη

2η Επανάληψη - distanceWithIndexesGPU						
0.	4	-1	5	48800.990	13675.806	9738.971
1.	2	-2	2	0.000	24842.250	32745.551
2.	0	1	2	44605.597	62467.515	8787.282
3.	3	-3	2	0.000	84427.438	8571.144
4.	1	3	2	13470.351	73995.647	48.932
2η Επανάληψη - my_sort						
0.	4	-1	5	48800.990	13675.806	9738.971
1.	2	-2	2	44605.597	24842.250	32745.551
2.	0	-4	1	0.000	62467.515	8787.282
3.	3	-3	2	13470.351	84427.438	8571.144
4.	1	-6	1	0.000	73995.647	48.932

Ο αλγόριθμος τερματίζει μετά από $\frac{log(5)}{log(2)} = 2$ δηλαδή όταν όλα τα heads γίνουν αρνητικά.

Επομένως τώρα το δέντρο είναι έτοιμο. Για να πάρει την μορφή των δεικτών όπως στην εργασία 1 αρκεί να ακολουθήσουμε τον αλγόριθμο για την μετατροπή από πίνακα σε δυαδικό δέντρο με βάση την ενδοδιατεταγμένη διάσχιση.



Δηλαδή το σημείο idx=4 έχει head=|-1|=1 άρα είναι η ρίζα με median =48800,990 και idx=4.

Το αριστερό παιδί θα έχει head=1*2=2=|-2| άρα θα είναι το idx=2 και median=44605,597

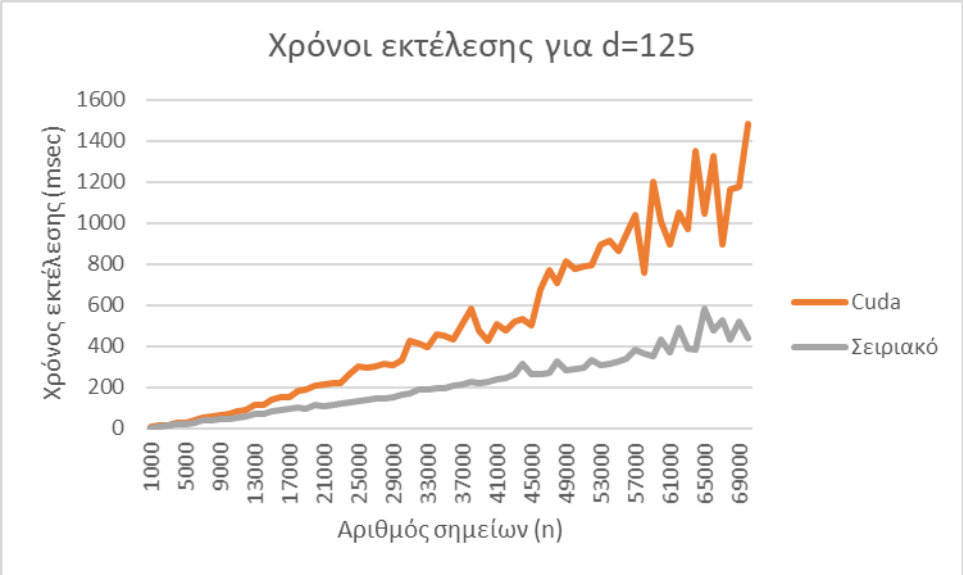
και το δεξί παιδί θα έχει head=1*2+1=3=|-3| άρα θα είναι το idx= 3 και median=13470,351

Αντίστοιχα στην θέση 4 αντιστοιχεί το idx=0 ενώ στην θέση 6 το idx=1 με median=0 αφού είναι φύλλα του δέντρου.

Η μετατροπή γίνεται στην CPU και δεν λαμβάνεται στο χρόνο υλοποίησης του δέντρου.

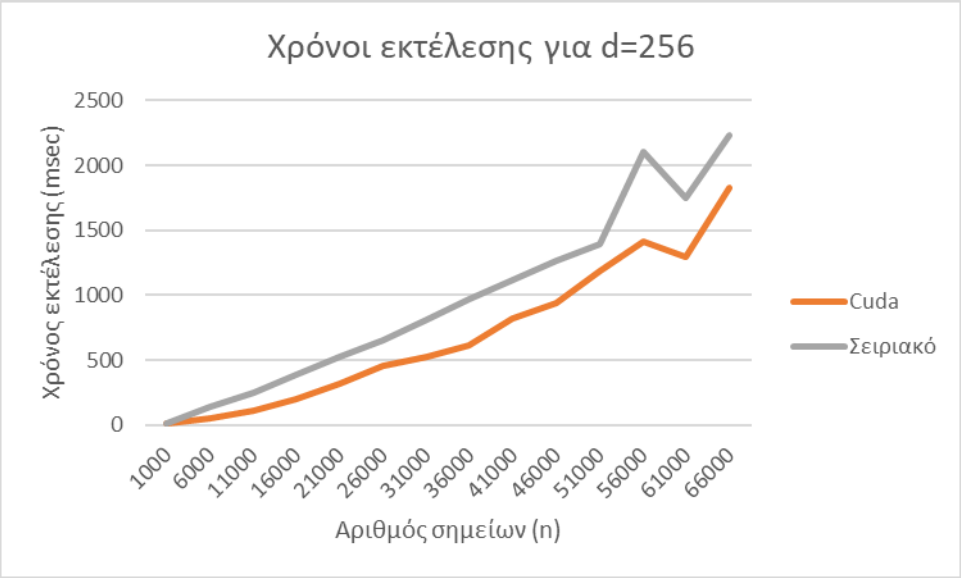
Στατιστικά και χρόνοι

Οι μετρήσεις έγιναν σε laptop με CPU i7-7500U @ 2.70GHz και GPU Nvidia GeForce 940MX 4gb.
Με τα παραπάνω τεχνικά χαρακτηριστικά είναι λογικό να μην δούμε επιτάχυνση της υλοποίησης σε σχέση με το σειριακό καθώς ο επεξεργαστής είναι ιδιαίτερα δυνατός σε σχέση με την κάρτα γραφικών. Παρακάτω βλέπουμε τα γραφήματα για μικρό και μεγάλο d.



Βλέπουμε ότι για d=125 η CPU μπορεί να εκτελεί μέχρι και 2φορές πιο γρήγορα τους υπολογισμούς ακόμα και για μεγάλα n.

Καθώς όμως αυξάνουμε το d βλέπουμε ότι υπάρχει επιτάχυνση της Cuda σε σχέση με τον σειριακό κώδικα.



Real datasets from UCI

Δεν ήταν δυνατό να συγκρίνω την υλοποίηση με kd-tree καθώς οι χρόνοι που υπάρχουν σε διάφορα papers αναφέρονται σε τυχαία δεδομένα και μάλιστα σε πολύ καλύτερες κάρτες γραφικών.

Από το UCI πήρα τα δεδομένα «Breast Cancer Wisconsin (Diagnostic) Data Set» μεγέθους 569*30 (double)

Για να διαβάσει ο tester δεδομένα από αρχείο θέλει 3 ορίσματα, 1. (το όνομα του αρχείου), 2.(το πλήθος των δεδομένων), 3.(την διάσταση του δείγματος). Έτσι πχ με την εντολή cuda.exe ./wdbc.data 569 30 παίρνουμε τους αντίστοιχους χρόνους.

$$n\ x\ d \ \ |\ Cuda\ |\ Sequential$$
$$560\ x\ \ 30|\ 3\ msec|\ 2\ msec$$

Περιεχόμενα Repository

Τα απαιτούμενα αρχεία για την εκτέλεση της συνάρτησης CUDABuildvp αντίστοιχα με την Buildvp είναι τα CUDABuildvp.cu eucleidean_distance.cu my_select.cu vptree_sequential.cu ενώ το αρχείο tester.cu περιέχει την main όπου επιστρέφει 0 εάν είναι σωστά ή 1 εάν είναι λάθος σύμφωνα με τον tester της πρώτης εργασίας.

Προτεινόμενη εντολή compile
nvcc -O3 .\my_select.cu .\euclidean_distance.cu .\vptree_sequential.cu .\CUDABuildvp.cu .\tester.cu -o cuda.exe
Περιέχεται ήδη το cuda.exe για windows 64bit που δέχεται ως ορίσματα 2 ακέραιους n d. Για το αρχείο αυτό υπάρχει και το script.bat το οποίο εκτελεί για σειρά από δοκιμές σε command line.