Παράλληλα και Διανεμημένα Συστήματα
Εργασία - 3 CUDA

Κωστινούδης Ευάγγελος ekostinou@auth.gr 8708
Κυριαζής Γεώργιος - Λέανδρος gkyriazt@ece.auth.gr 7711

January 25, 2018

1 Συμπεριλαμβανόμενα αρχεία

Link: Dropbox

Τα αρχεία που συμπεριλαμβάνονται στο αρχείο .zip είναι τα εξής:
φάκελος **Final** με όλα τα αρχεία κώδικα ήτοι:

- mean_shift_with_shared.cu και mean_shift_without_shared.cu με τους κώδικες
- Τα dataset σε binary

2 Περιγραφή προβλήματος

Μας δόθηκε ένας σειριακός κώδικας MatLAB ο οποίος υλοποιούσε τον αλγόριθμο Clustering mean-shift. Μας ζητήθηκε να κάνουμε τον αλγόριθμο παράλληλο σε CUDA. Έγιναν 2 υλοποιήσεις μία με χρήση της shared memory και μια χωρίς την χρήση αυτής.

3 Παρουσίαση Αλγορίθμου

3.1 Στρατηγική παραλληλοποίησης

Για την επιτάχυνση του προβλήματος χρησιμοποιούμε N block και N threads (οπου N ο αριθμός των σημείων). Κάθε μπλοκ υπολογίζει ένα νέο σημείο y. Σε κάθε νήμα ένα διαφορετικό όρισμα του Kernel δηλαδήσ ένα όρο του συνολικού αθροίσματος.

3.2 #define

Αρχικά δηλώνουμε ως σταθερές:

- τον αριθμό των σημείων: (**N**)
- τον αριθμό των παραμέτρων: (**D**)
- την τυπική απόκλιση της Gaussian συνάρτησης πυρήνα: (**STANDARD_DEVIATION**)
- το e το οποίο καθορίζει την σύγκλιση του αλγορίθμου: (**EPSILON**)
- το όνομα του αρχείου με τα δεδομένα (**POINTS_FILE**)

3.3 Main function

Αρχικά δεσμεύεται μνήμη για την αποθήκευση των σημείων εισόδου (**x**)και του τελικού διανύσματος εξόδου(**y**) μέσω της **malloc**. Έπειτα ανοίγεται και διαβάζεται το αρχείο των σημείων και δεσμεύεται μνήμη μέσω της **cudaMalloc** στην μεριά της κάρτας γραφικών για εισαγωγή και εξαγωγή των δεδομένων (**dev_x** , **dev_y**). Εν συνεχεία μεταφέρονται τα δεδομένα στην κάρτα και ξεκινάμε την μέτρηση του χρόνου της κλήσης kernel για την υλοποίηση του αλγόριθμου *mean_shift*. Στό τέλος της εκτέλεσης μετράμε τον χρόνο και τον εμφανίζουμε. Τέλος τα δεδόμενα από την κάρτα επιστρέφονται από την GPU.

3.4 __global__ void meanShift()

Αρχικά ορίζουμε μια μεταβλητή y η οποία αναφέρεται στο y του κάθε block και μια μεταβλητή x που αναφέρεται στο σημείο της εισόδου που επεξεργάζεται το κάθε νήμα. Έπειτα δεσμεύεται μνήμη στην shared memory για τον πίνακα των δεδομένων εισόδου (**x**) μαζί με τους πίνακες του αριθμητή και του παρανομαστή. Αντιγράφουμε τον πίνακα χ στην shared memory και συγχρονίζουμε όλα τα νήματα έτσι ώστε να έχουν αντιγραφεί όλα τα σημεία πριν συνεχίσουμε. Ύστερα αρχικοποιούμε το διανύσματα y(**y_prev**) και αρχίζει η επαναληπτική διαδικασία του αλγορίθμου σύμφωνα με τον σειριακό κώδικα του matlab και της εκφώνησης.

Πίο συγκεκριμένα αφαιρούμε το διάνυσμα χ από το y_prev υπολογίζουμε την δεύτερη νόρμα του. Μετά ελέγχουμε αν είναι αρκετά κοντά τα σημεία για να δούμε αν η εν λόγω νόρμα θα συμμετάσχει στον υπολογισμό

της (1) από την εκφώνηση. Σε περίπτωση που ισχύει παίρνουμε το τετράγωνο της νόρμας υπολογίζουμε τον πυρήνα και μετά το κάθε υποδιάνυσμα του αριθμητή. Αν η παραπάνω συνθήκη δεν ισχύει(αν τα 2 σημεία είναι πιο μακρία από το σ^2) τότε δίνουμε στον παρανομαστή και το διάνυσμα του αριθμητή την τιμή 0.

Στις επόμενες γραμμές γίνεται reduction δηλαδή παράλληλος τελικός υπολογισμός των αθροισμάτων του αριθμητή και του παρανομαστή. Συγχρονίζουμε τα νήματα για να βεβαιωθούμε ότι έχουν γίνει όλοι οι προηγούμενοι υπολογισμοί. Στο κομμάτι αυτό επιτυγχάνουμε να έχουμε υπολογίσει το άθροισμα του παρανομαστή στο πρώτο στοιχείο του παρανομαστή (**denominator[0]**) και το διάνυσμα του αριθμητή στα πρώτα D στοιχεία του.

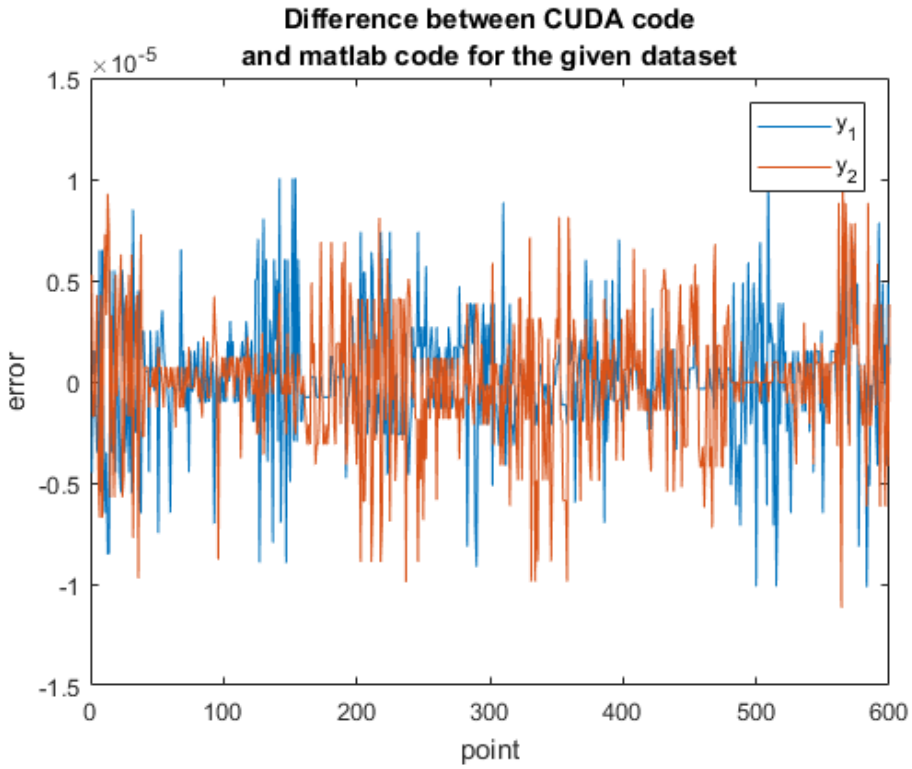
Τέλος υπολογίζουμε τα νέα διανύσματα y και m, ανανεώνουμε το y_prev, υπολογίζουμε την Frobenius νόρμα του m και ελέγχουμε την συνθήκη σύγκλισης και μεταφέρουμε το τελικό αποτέλεσμα στην global memory.

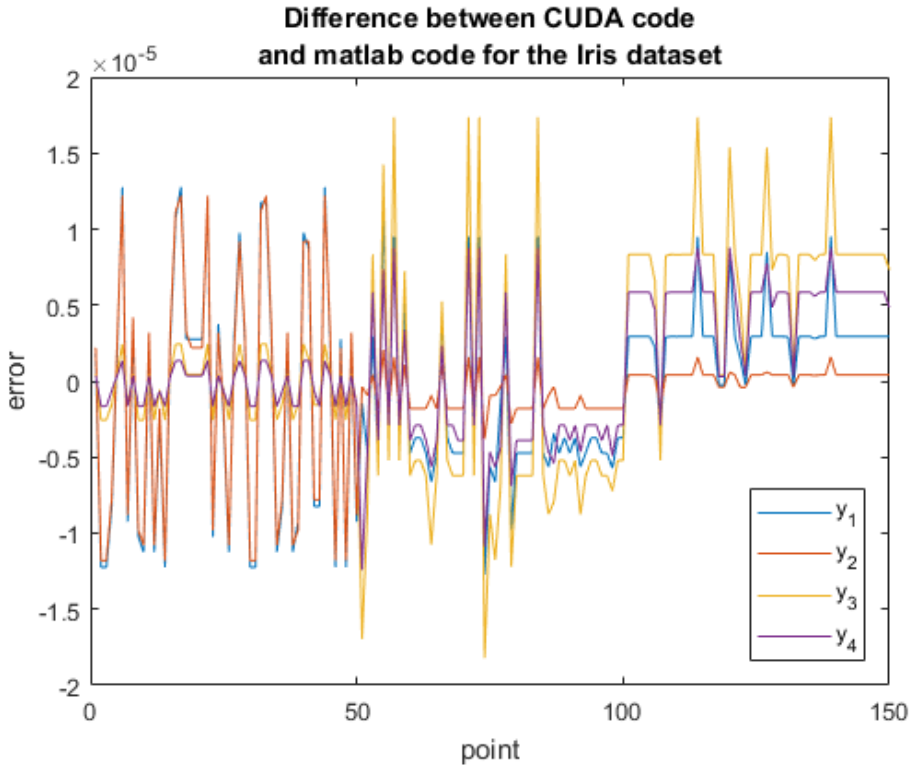
4 Έλεγχος ορθότητας και επίδοση αλγορίθμου

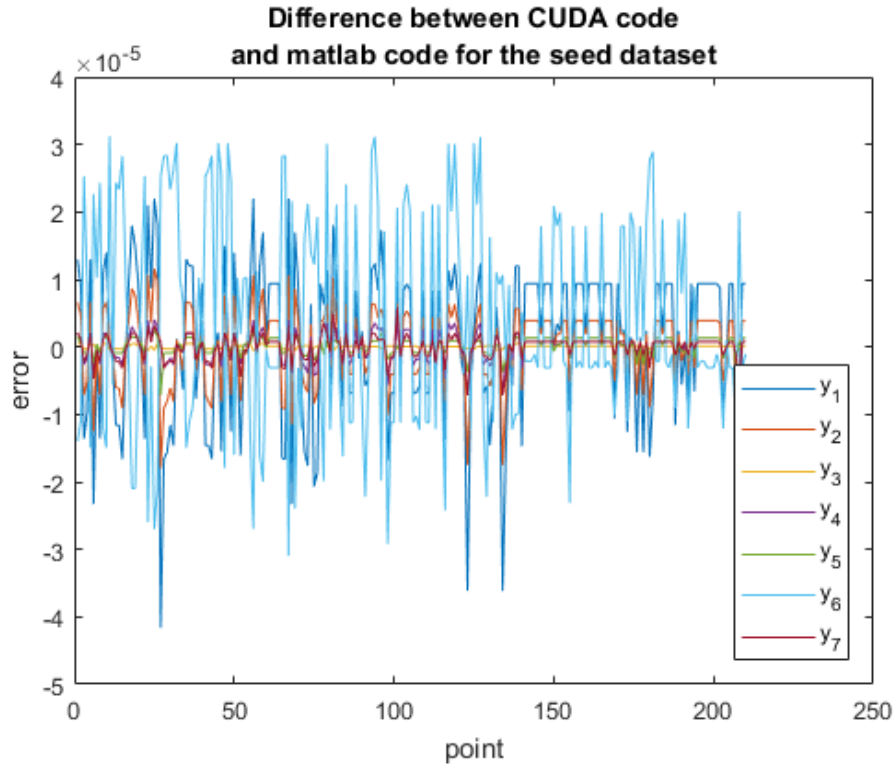
4.1 Έλεγχος ορθότητας

Για τον έλεγχο ορθότητας χρησιμοποιήσαμε 3 dataset. Το δωσμένο αρχικό το γνωστό Iris και ένα dataset με παραμέτρους σπόρων σιταριού. Υπολογίζουμε την διαφορά της εξόδου του δικού μας αλγορίθμου με των σειριακό του Matlab. Για το dataset με τα σιτάρια χρησιμοποιήσαμε $\sigma = 1.5$. Πληροφορίες για τα εν λόγω dataset υπάρχουν άφθονες στο διαδίκτυο δηλαδή τι συμβολίζει η κάθε παράμετρος κ.ο.κ.

$$Error = y_c - y_m$$







4.2 Χρόνοι αλγορίθμου

Προφανώς $1us = 10^{-6}s$

——	Given	Iris	Seeds
Shared mem	3898 us	1257 us	6436 us
Global mem	7080 us	1747 us	10944 us
Serial	4.91 s	6.59 s	10.99 s

Παρατηρούμε ότι με τον παράλληλο κώδικα CUDA επιτυγχάνουμε δραματική επιτάχυνση. Όπως ήταν

αναμενώμενο η χρήση της shared memory βελτιώνει την επίδοση του αλγορίθμου. Ο σειριακός κώδικας είναι αυτός του MatLab. Αυτοί χρόνοι προκύπτουν με ταυτόχρονη εμφάνιση των σχετικών εικόνων.