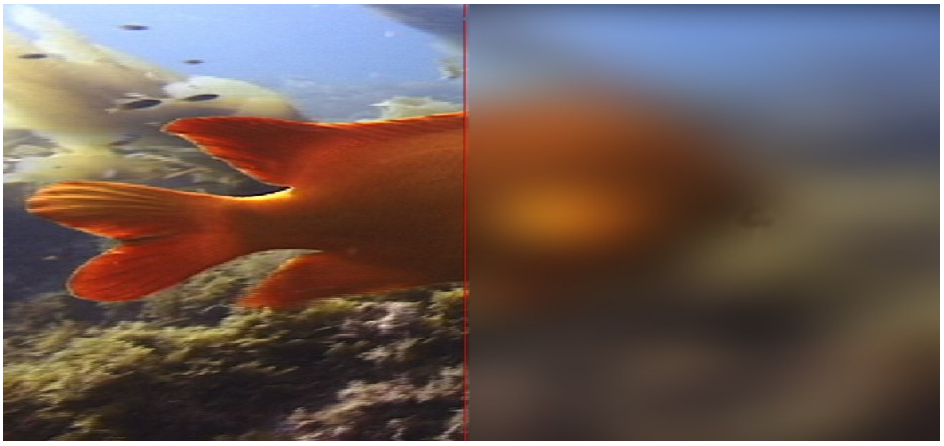
Προγραμματισμός Συστημάτων Υψηλών Επιδόσεων (ECE 415)
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

Διδάσκων: Χρήστος Δ. Αντωνόπουλος

3η Εργαστηριακή Άσκηση

Στόχος: Εξοικείωση με το περιβάλλον μεταγλώττισης CUDA, συγγραφή απλού προγράμματος CUDA, πειραματισμός με γεωμετρίες block και grid, προβλήματα πλήθους threads, προβλήματα ακρίβειας.

Υπόβαθρο:
Η συνέλιξη (convolution) βρίσκει εφαρμογή σε πλήθος εφαρμογών της μηχανικής και των μαθηματικών. Για παράδειγμα, πολλά φίλτρα μετασχηματισμού εικόνας στην ουσία υλοποιούνται ως συνελίξεις. Στο δεξί κομμάτι της παρακάτω εικόνας βλέπετε το αποτέλεσμα της εφαρμογής ενός φίλτρου Gaussian blur.



Εικόνα 1: Εφαρμογή gaussian blur σε εικόνα
(http://www.borisfx.com/images/bcc3/gaussian_blur.jpg)

Με μαθηματικούς όρους η συνέλιξη ποσοτικοποιεί το αποτέλεσμα της «επικάλυψης» δύο συναρτήσεων. Μπορούμε να τη σκεφτούμε σαν μια πράξη «ανάμιξης» η οποία ολοκληρώνει το αποτέλεσμα του σημείο-προς-σημείο πολλαπλασιασμού του ενός συνόλου δεδομένων με το άλλο.

$$r(i) = (s * k)(i) = \int s\,(i-n) * k(n)dn$$

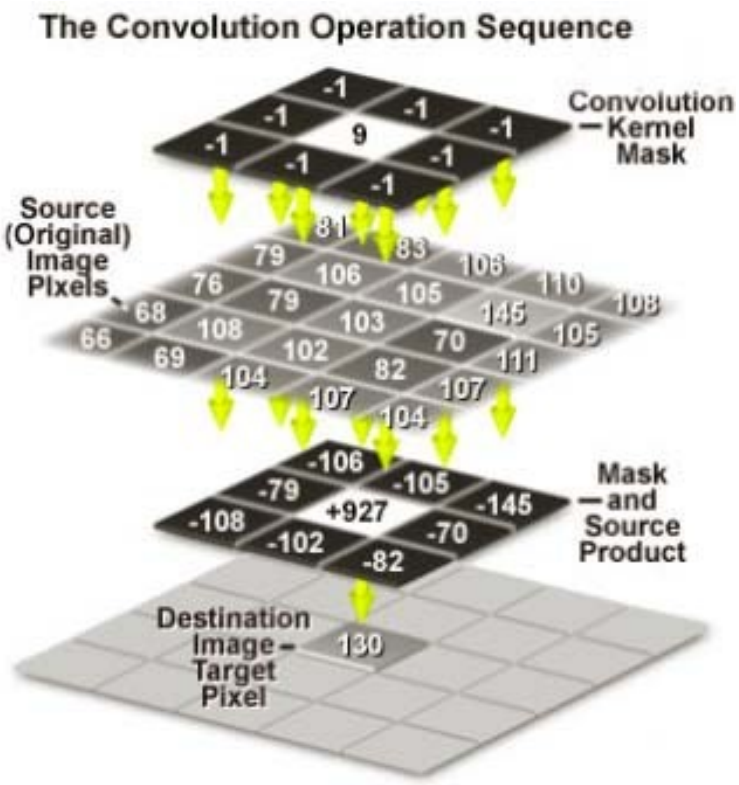
Στον διακριτό κόσμο, μπορεί να γραφτεί ως:

$$r(i) = (s * k)(i) = \sum_{n} s\,(i-n)k(n)$$

Η συνέλιξη μπορεί να επεκταθεί και σε 2 διαστάσεις:

$$r(i) = (s * k)(i,j) = \sum_{n}\sum_{m} s\,(i-n, j-m)k(n,m)$$

Στο πεδίο της επεξεργασίας εικόνας, ένα φίλτρο συνέλιξης είναι απλά το σημείο-προς-σημείο γινόμενο των βαρών του φίλτρου με τα pixels της εικόνας εισόδου, εντός ενός παραθύρου που περικλείει κάθε pixel εξόδου. Η παρακάτω εικόνα αποσαφηνίζει τη διαδικασία εφαρμογής μίας μάσκας 3x3 πάνω σε μια εικόνα εισόδου για να προκύψει ένα pixel της εικόνας εξόδου. Στο επόμενο βήμα, η μάσκα θα μετακινηθεί πάνω στην εικόνα εισόδου ώστε να υπολογιστεί ένα νέο pixel της εικόνας εξόδου κ.ο.κ.



Εικόνα 1: Εφαρμογή 2D συνέλιξης με μάσκα 3x3 σε εικόνα για την παραγωγή ενός pixel της εικόνας αποτελέσματος. (http://www.biomachina.org/courses/structures/01.html)

Τυπικά, η εφαρμογή ενός δισδιάστατου φίλτρου συνέλιξης για τον υπολογισμό μιας τιμής εξόδου (pixel) απαιτεί n*m πολλαπλασιασμούς, όπου n και m οι διαστάσεις του φίλτρου. Διαχωρίσιμα λέγονται τα δισδιάστατα φίλτρα τα οποία μπορούν να εκφρασθούν ως η σύνθεση δύο μονοδιάστατων φίλτρων, ένα από τα οποία εφαρμόζεται στις γραμμές τις εικόνας και ένα στις στήλες. Για παράδειγμα, η εφαρμογή του παρακάτω Sobel φίλτρου
$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$ισοδυναμεί με την εφαρμογή του $\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}$και ακολούθως του $[-1 \quad 0 \quad 1]$.
Στα πλαίσια αυτής της άσκησης καλείστε να γράψετε τον κώδικα που θα εφαρμόσει ένα διαχωρίσιμο, δισδιάστατο φίλτρο πάνω σε ένα δισδιάστατο πίνακα (ο οποίος ας υποθέσουμε ότι αντιστοιχεί σε μια εικόνα).

Βήματα:
0) Τρέξτε το deviceQuery και καταγράψτε τα αποτελέσματα. Αυτό το βήμα θα πρέπει να το επαναλαμβάνετε σε όλες τις ασκήσεις.

1) Δείτε προσεκτικά τις παραμέτρους εκτέλεσης του compiler nvcc, γράφοντας nvcc --help σε ένα τερματικό. Φροντίστε πάντα να μεταγλωττίζετε τον κώδικα για τη CPU με το μέγιστο βαθμό

βελτιστοποιήσεων (-O4).

2) Γράψτε κώδικα ο οποίος:

α) Εκτελεί συνέλιξη 2 διαστάσεων σε μια εικόνα NxN με στοιχεία τύπου float, με χρήση ενός φίλτρου που περιέχει τυχαίες τιμές, στη GPU. Το αποτέλεσμα θα πρέπει να αποθηκεύεται επίσης σε μία NxN εικόνα. Το μέγεθος N δίνεται από τη γραμμή εντολών κατά την εκτέλεση του προγράμματος, ενώ η εικόνα εισόδου αρχικοποιείται με τυχαίες τιμές. Ο τρόπος με τον οποίο πραγματοποιείται η συνέλιξη 2 διαστάσεων δίνεται σαν κώδικας που εκτελείται στη CPU και μέσω της ανάλυσης του μπορείτε να καταλάβετε την λειτουργία του και να την μεταφέρετε στη GPU. Το FILTER_RADIUS θα πρέπει επίσης να δίνεται από τη γραμμή εντολών ενώ το FILTER_LENGTH ορίζεται μέσα στον κώδικα (σε σχέση με το FILTER_RADIUS). Πρόκειται για την ακτίνα του φίλτρου συνέλιξης και το συνολικό του μήκος αντίστοιχα. Ο κώδικάς σας θα πρέπει να χρησιμοποιεί τα threads ενός μοναδικού block ώστε κάθε thread να υπολογίζει ένα στοιχείο της εικόνας-αποτελέσματος. Ο κώδικάς σας οφείλει να:

i) Δεσμέυει μνήμη στο device για την εικόνα εισόδου και αποτελέσματος, αλλά και για το φίλτρο που χρησιμοποιείται στη συνέλιξη. Η μνήμη αυτή θα πρέπει να αποδεσμεύεται με το πέρας της εκτέλεσης.

ii) Μεταφέρει τα δεδομένα στο device και τα αποτελέσματα πίσω στον host.

iii) Εκτελεί τον kernel.

iv) Ελέγχει συστηματικά και προσεκτικά για σφάλματα χρόνου εκτέλεσης.

β) Ελέγχει εάν το αποτέλεσμα από τη GPU συμφωνεί με αυτό από τη CPU δεδομένης μιας αποδεκτής ακρίβειας (σε αριθμό δεκαδικών ψηφίων). Εάν έστω και ένα στοιχείο βρεθεί να διαφέρει εκτός της ανεκτής ακρίβειας, η σύγκριση τερματίζεται και τυπώνεται κατάλληλο μήνυμα.

3) Πειραματική μελέτη:

α) Μέχρι ποιο μέγεθος εικόνας (πειραματιστείτε πάντα με μεγέθη εικόνας που είναι δυνάμεις του 2 και μεγαλύτερες από το μήκος του φίλτρου) μπορείτε να υποστηρίξετε χωρίς να συμβεί κάποιο σφάλμα χρόνου εκτέλεσης; Γιατί συμβαίνει αυτό το σφάλμα; (*μόνο για αυτό το ερώτημα χρησιμοποιήστε ακτίνα φίλτρου 4*)

β) Για το μέγιστο μέγεθος εικόνας που μπορείτε να υποστηρίξετε, ποια είναι η μέγιστη ακρίβεια (σε αριθμό ψηφίων), σε σχέση με το μέγεθος του φίλτρου, που μπορεί να υποστηριχθεί χωρίς να παρουσιάζονται σφάλματα σύγκρισης;

4) Γράψτε κώδικα ο οποίος λύνει το πρόβλημα που παρατηρήθηκε στο βήμα (3α), χρησιμοποιώντας πολλαπλά τετράγωνα blocks νημάτων οργανωμένα σε ένα τετράγωνο grid. Μέχρι τι μεγέθη εικόνας μπορείτε πλέον να υποστηρίξετε;

5) Πειραματική μελέτη:

α) Μελετήστε και σχεδιάστε σε ένα διάγραμμα τη μέγιστη ακρίβεια-στόχο (ως αριθμό δεκαδικών ψηφίων) που οδηγεί σε επιτυχείς συγκρίσεις σε συνάρτηση με το μέγεθος του φίλτρου. Χρησιμοποιείστε εικόνα 1024x1024 και πειραματιστείτε με φίλτρα των οποίων το η ακτίνα είναι δύναμη του 2. Τι παρατηρείτε; Γιατί συμβαίνει αυτό;

β) Μετρήστε το χρόνο εκτέλεσης στη CPU και στη GPU ως συνάρτηση του μεγέθους της εικόνας και σχεδιάστε τους σε ένα διάγραμμα. Για τις μετρήσεις χρόνου πειραματιστείτε με ακτίνα φίλτρου 16 και μέγεθος εικόνας έως 16384x16384 (ή χαμηλότερο εάν υπάρχει κάποιος περιορισμός στη CPU ή στη GPU που δε σας αφήνει να φτάσετε σε αυτό το μέγεθος). Στο χρόνο εκτέλεσης της GPU οφείλετε να συμπεριλάβετε και τον χρόνο εκτέλεσης για τις μεταφορές δεδομένων προς και αποτελεσμάτων από τη GPU, όχι όμως και τον χρόνο για τη

δέσμευση / αποδέσμευση της μνήμης για τις εικόνες και το φίλτρο πάνω στη συσκευή. Χρησιμοποιήστε τη μέθοδο μέτρησης χρόνου που σας φαίνεται καταλληλότερη για κάθε πλατφόρμα.

6) Αλλάξτε τον τύπο των στοιχείων σε doubles αντί για floats. Επαναλάβετε τα πειράματα του βήματος (5). Τι παρατηρείτε;

7) Απαντήστε στις ακόλουθες ερωτήσεις:

α) Πόσες φορές διαβάζεται κάθε στοιχείο της εικόνας εισόδου και του φίλτρου κατά την εκτέλεση του kernel;

β) Ποιός είναι ο λόγος προσπελάσεων μνήμης προς πράξεις κινητής υποδιαστολής; Θεωρήστε τους πολλαπλασιασμούς και τις προσθέσεις ως ξεχωριστές πράξεις και αγνοείστε την αποθήκευση του αποτελέσματος. Μετρήστε μόνο τις αναγνώσεις από την global memory της GPU ως προσπελάσεις μνήμης.

8) Ο κώδικας που γράψατε στα βήματα (3α) και (4), όπως μπορείτε εύκολα να παρατηρήσετε, πάσχει από το φαινόμενο του divergence. Το πρόβλημα το δημιουργεί το if που ελέγχει αν τμήμα του φίλτρου βρίσκεται εκτός του πίνακα. Ξεκινώντας από τον κώδικα του βήματος (4), προσπαθήστε να χρησιμοποιήσετε την τεχνική padding ώστε να λύσετε το πρόβλημα του divergence εντός των warps. Ποια η διαφορά ανάμεσα στους χρόνους του εν λόγω ερωτήματος με τους χρόνους που παρατηρήθηκαν στο ερώτημα (5);
Hint: Ο βαθμός του padding που απαιτείται εξαρτάται από την ακτίνα του φίλτρου συνέλιξης.

Παράδοση:
Πρέπει να παραδώσετε:

- Τον κώδικα των βημάτων (2), (4), (6) και (8).
- Αναφορά με τις απαντήσεις στα ερωτήματα, αντίστοιχα διαγράμματα και εξηγήσεις (καθώς και το αποτέλεσμα του deviceQuery).

Τρόπος παράδοσης:
Δημιουργήστε ένα αρχείο .tar.gz με τα παραπάνω περιεχόμενα και όνομα <όνομα1>_<AEM1>_<όνομα2>_<AEM2>_<όνομα3>_<AEM3>_lab3.tar.gz. Ανεβάστε το εμπρόθεσμα στο e-class.

Προθεσμία παράδοσης:
Τρίτη 25/11/2025 23:59.

Παρατηρήσεις:

- Όλη η ανάπτυξη θα γίνει στο σύστημα csl-venus (10.64.82.60) ή σε δικό σας μηχάνημα, αν υποστηρίζεται CUDA, ενώ οι τελικές μετρήσεις στο csl-venus.
- Το csl-venus διαθέτει 2 κάρτες Tesla K80. Η κάθε κάρτα έχει 2 GK210 GPU chips (αρχιτεκτονικής Kepler).
- Εάν δουλεύετε σε δικό σας μηχάνημα και χρησιμοποιείτε την ίδια GPU για την εκτέλεση κώδικα CUDA αλλά και για να "οδηγήσετε" την οθόνη σας, ο μέγιστος επιτρεπτός χρόνος εκτέλεσης οποιουδήποτε kernel ενδέχεται να περιορίζεται σε περίπου 5". Μετά το χρόνο αυτό το σύστημα ενδέχεται να τερματίζει βίαια τον kernel.
- Σε συστήματα που έχουν πολλαπλές κάρτες γραφικών, μπορείτε να ελέγχετε ποια/ποιες είναι ορατές και χρησιμοποιούνται. Ένας τρόπος να γίνει αυτό είναι να έχετε θέσει τη μεταβλητή περιβάλλοντος CUDA_VISIBLE_DEVICES στο shell από το οποίο εκτελείτε το

πρόγραμμά σας. Π.χ.

export CUDA_VISIBLE_DEVICES = 1 (είναι ορατή η GPU 1 μόνο)

export CUDA_VISIBLE_DEVICES=0,3 (είναι ορατές οι GPUs 0 και 3 μόνο)

unset CUDA_VISIBLE_DEVICES (είναι ορατές όλες οι GPUs)

Μπορείτε να αξιοποιήσετε αυτή την ιδιότητα και να κάνετε δεσμεύσεις χρόνου για συγκεκριμένη GPU.

- Φροντίστε να απελευθερώνετε όλη τη δυναμικά δεσμευμένη μνήμη και να κάνετε reset το device στο τέλος του προγράμματός σας, ανεξαρτήτως του αν τερμάτισε κανονικά ή απέτυχε κάποιος ενδιάμεσος έλεγχος και τερμάτισε πρόωρα.