Προγραμματισμός Συστημάτων Υψηλών Επιδόσεων (ECE415)
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

Διδάσκων: Χρήστος Δ. Αντωνόπουλος

4η Εργαστηριακή Άσκηση

Στόχος: Βελτιστοποίηση προγράμματος CUDA, χρήση streams.

Υπόβαθρο:
Στα πλαίσια προηγούμενης άσκησης κληθήκατε να γράψετε κώδικα ο οποίος εφαρμόζει, με χρήση συνέλιξης, ένα διαχωρίσιμο δισδιάστατο φίλτρο πάνω σε ένα δισδιάστατο πίνακα (ο οποίος υποθέσαμε ότι αντιστοιχεί σε μια εικόνα).

Σε αυτή την άσκηση θα πρέπει να βελτιστοποιήσετε τον προηγούμενο κώδικα, να τον εκτελέσετε σε πίνακες πολύ μεγαλύτερου μεγέθους και να προσπαθήσετε να επικαλύψετε μεταξύ τους διαφορετικούς υπολογισμούς καθώς και υπολογισμό με μεταφορές δεδομένων.

Θα χρησιμοποιήσετε τον κώδικα από την προηγούμενη εργασία. Μπορείτε να χρησιμοποιήσετε είτε την έκδοση χωρίς padding του πίνακα, είτε την έκδοση με padding, η οποία εξαλείφει το φαινόμενο του divergence. Συνίσταται να χρησιμοποιήσετε την έκδοση με padding, εκτός αν στην προηγούμενη εργασία δεν καταφέρατε να αναπτύξετε αυτή την έκδοση με τρόπο που να δίνει ορθά αποτελέσματα και να έχει χρόνο καλύτερο (ή έστω όχι χειρότερο) της έκδοσης χωρίς padding. Θα εργαστείτε στην έκδοση του κώδικα που χρησιμοποιεί doubles.

Βήματα:
0) Τρέξτε το deviceQuery και καταγράψτε τα αποτελέσματα. Φροντίστε πάντα να μεταγλωττίζετε τον κώδικα για τη CPU με το μέγιστο βαθμό βελτιστοποιήσεων (-O4).

1) Βελτιστοποιήστε τον κώδικα της προηγούμενης εργασίας με όποιον τρόπο κρίνετε πρόσφορο (χρήση caches, registers κλπ). Μετά από κάθε αλλαγή βεβαιωθείτε ότι ο κώδικάς σας δουλεύει σωστά. Χρησιμοποιήστε εικόνα 8192x8192, ακτίνα φίλτρου 32, αλλά φροντίστε ο κώδικάς σας να λειτουργεί σωστά και με άλλα μεγέθη. Αν χρειαστεί να κάνετε κάποιες παραδοχές για το μέγεθος του φίλτρου (π.χ. εύρος επιτρεπτής ακτίνας, άλλοι περιορισμοί κλπ) καταγράψτε αυτές τις παραδοχές στην αναφορά σας.

Χρησιμοποιήστε τον profiler κατά τη διάρκεια της διαδικασίας. Καταγράψτε τις διαφορές στην επίδοση μεταξύ της αρχικής και της τελικής (μετά τις βελτιστοποιήσεις) έκδοση του κώδικά σας στη GPU.

2) Στη βελτιστοποιημένη έκδοση του κώδικά σας πειραματιστείτε με μεγαλύτερα και μικρότερα φίλτρα. Παρατηρήστε και καταγράψτε πώς το μέγεθος του φίλτρου επηρεάζει το χρόνο εκτέλεσης των kernels, καθώς και τη σχέση του χρόνου εκτέλεσης kernels / χρόνου μεταφορών μνήμης.

3) Επόμενός σας στόχος είναι να καταφέρετε να υποστηρίξετε πολύ μεγαλύτερες εικόνες πάνω στη GPU. Για το λόγο αυτό η εκτέλεση των kernels θα πρέπει να είναι blocked, με τέτοιο τρόπο ώστε όλα τα δεδομένα που απαιτούνται για την εκτέλεση κάθε βήματος να χωράνε στη GPU. Η υλοποίησή σας θα πρέπει να υποστηρίζει παραμετρικό μέγεθος block. Μπορείτε να κάνετε όποιες

παραδοχές θέλετε για τα επιτρεπτά μεγέθη block (π.χ δύναμη του 2, δύναμη του 2 + a, ή οτιδήποτε άλλο σας εξυπηρετεί).

Επιβεβαιώστε ότι η υλοποίησή σας δουλεύει σωστά για εικόνες μεγέθους 16384x16384.

Μετά από αυτό το βήμα δεν είναι απαραίτητο να εκτελείτε τον κώδικα στη CPU, να δεσμεύετε μνήμη στη CPU (πέρα από αυτή που χρειάζεται για τον πίνακα εισόδου και τον πίνακα με το αποτέλεσμα) και να κάνετε σύγκριση του αποτελέσματος της CPU με αυτό της GPU. Βάλτε τα αντίστοιχα κομμάτια κώδικα μέσα σε ζεύγη #ifdef / #endif και φροντίστε στο εξής να μην είναι ορισμένο το macro που ελέγχεται στο #ifdef.

Καταγράψτε τα αποτελέσματα του profiling ώστε να τα αξιοποιήσετε για σύγκριση σε επόμενα βήματα.

4) Παρατηρήστε στον profiler ότι, όπως ήταν αναμενόμενο, δεν υπάρχει καμία επικάλυψη ανάμεσα στον κώδικα που εκτελείται από διαφορετικούς kernels ή σε εκτέλεση κώδικα και μεταφορά δεδομένων. Μετατρέψτε τον κώδικά σας με χρήση streams ώστε να υπάρχει επικάλυψη στο μεγαλύτερο βαθμό που μπορείτε.

Επιβεβαιώστε την επιτυχία σας με τον profiler και καταγράψτε τα αποτελέσματα. Συγκρίνετέ τα με αυτά του προηγούμενου βήματος.

5) Ποιο είναι το μέγιστο μέγεθος εικόνας που μπορείτε να υποστηρίξετε; Ποιος είναι ο πόρος που σας περιορίζει;

Παράδοση:
Πρέπει να παραδώσετε:

- Τον κώδικα των βημάτων (1), (3) και (4).
- Αναφορά με τις απαντήσεις στα ερωτήματα (και το αποτέλεσμα του deviceQuery).

Τρόπος παράδοσης:
Δημιουργήστε ένα αρχείο .tar.gz με τα παραπάνω περιεχόμενα και όνομα <όνομα1>_<AEM1>_<όνομα2>_<AEM2>_lab4.tar.gz. Ανεβάστε το εμπρόθεσμα στο e-class.

Προθεσμία παράδοσης:
Τρίτη 22/12/2020 23:59.

Παρατηρήσεις:

- Όλη η ανάπτυξη θα γίνει στο σύστημα inf-mars1 (10.64.82.31) (ή σε δικό σας μηχάνημα, αν υποστηρίζεται CUDA), ενώ οι τελικές μετρήσεις στο csl-artemis (10.64.82.65). Οι κρατήσεις slots αφορούν, όπως και στην προηγούμενη εργασία, στο csl-artemis.
- Το csl-artemis διαθέτει κάρτα Tesla K80. Η κάρτα έχει 2 GK210 GPU chips (αρχιτεκτονικής Kepler). Το inf-mars1 διαθέτει 1 κάρτα GTX690, με επίσης 2 chips (επίσης αρχιτεκτονικής Kepler). Δώστε προσοχή γιατί οι απαντήσεις σε κάποια από τα ερωτήματα θα διαφέρουν!
- Προσέξτε ότι η μνήμη στο inf-mars1 (σε host και GPU) είναι πολύ μικρότερη από αυτή του csl-artemis, συνεπώς κατά την ανάπτυξη στο inf-mars1 θα πρέπει να περιοριστείτε σε μικρότερα μεγέθη. Μπορείτε να δείτε τη μνήμη της GPU με το deviceQuery, ενώ τη μνήμη του host με την εντολή cat /proc/meminfo. Ενδέχεται επίσης στο inf-mars1, ακόμα και για

μεγέθη εικόνων για τα οποία δεν έχετε πρόβλημα όταν τρέχετε μόνοι σας, να έχετε πρόβλημα σε ότι αφορά τη μνήμη του host όταν τρέχετε μαζί με άλλους.

- 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.