# Μάθημα: Συστήματα Υπολογισμού Υψηλών Επιδόσεων (ΜΔΕ 646)

Ονοματεπώνυμο: Θεοδώρου Γεώργιος

ΑΕΜ: 0497

**Report**

1) Ακολουθεί πίνακας και σχετικό διάγραμμα μετρήσεων χρόνων, βάση του κώδικα της προηγούμενης άσκησης (με padding), με ενεργοποιημένες και μη των cache L1 και L2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tile\_Width | Filter Radius | Image Size | GPU Time | dlcm=cg | dlcm=cs / dscm=wt |
| 16x16 | 16 | 64 x 64 | 0.000022 | 0.000026 | 0.000021 |
| 16x16 | 16 | 128 x 128 | 0.000029 | 0.000035 | 0.000048 |
| 16x16 | 16 | 256 x 256 | 0.000065 | 0.000109 | 0.000174 |
| 16x16 | 16 | 512 x 512 | 0.000237 | 0.000412 | 0.000593 |
| 16x16 | 16 | 1024 x 1024 | 0.000971 | 0.001748 | 0.003047 |
| 16x16 | 16 | 2048 x 2048 | 0.005957 | 0.006355 | 0.021371 |
| 16x16 | 16 | 4096 x 4096 | 0.01682 | 0.026898 | 0.050627 |
| 16x16 | 16 | 8192 x 8192 | 0.060491 | 0.100153 | 0.16653 |

Παρατηρούμε αρχικά ότι όταν απενεργοποιούμε την L1 cache οι χρόνοι εκτέλεσης των kernel αυξάνονται αισθητά, δηλαδή έχουμε μια αύξηση περίπου 65%. Ενώ όταν απενεργοποιούμε και την L2 cache τότε οι χρόνοι εκτοξέυονται, με μια αύξηση περίπου 175%.

3) Κάθε kernel με Thread Block = 16x16 και Tile = 16x16 χρησιμοποιεί 10-12 registers, 3072 bytes shared memory και 56 bytes constant memory (Επιπλέον έχουμε δεσμεύσει και 132 bytes constant memory για τον πίνακα με το φιλτρο.).Το μέγιστο μέγεθος thread block που βορούμε να υποστηρίξουμε ετσι και αλλιώς ειναι 32χ32=1024 threads per block. Επομένως αφού θέλουμε το tile να είναι ίσο με το thread block , τότε βορούμε να χρησιμοποιήσουμε μέχρι και 32χ32 tile.( 32χ32 ειναι η «καθαρή» διάσταση διότι στον row kernel θα πρέπει να προσθέσουμε ένα padding δεξία-αριστερά στο tile για να αποφύγουμε το divergence και αντίστοιχα στον column kernel πάνω-κάτω).

4) Ακολουθεί πίνακας και σχετικό διάγραμμα μετρήσεων χρόνων (σταθερή εικόνα 8192χ8192) για διάφορα μεγέθη tiles.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Filter Radius | Image\_Width | Thread Block | Tile\_W1 | Tile\_H1 | Tile\_W2 | Tile\_H2 | GPU Time |
| 16 | 8192 | 32x32 | 32 | 32 | 32 | 32 | 0.030857 |
| 16 | 8192 | 32x32 | 64 | 64 | 64 | 64 | 0.023221 |
| 16 | 8192 | 16x16 | 16 | 16 | 16 | 16 | 0.024366 |
| 16 | 8192 | 16x16 | 32 | 32 | 32 | 32 | 0.032905 |
| 16 | 8192 | 16x16 | 64 | 64 | 64 | 64 | 0.035246 |
| 16 | 8192 | 16x16 | 64 | 16 | 16 | 64 | 0.025193 |
| 16 | 8192 | 16x16 | 128 | 16 | 16 | 128 | 0.024573 |
| 16 | 8192 | 16x16 | 256 | 16 | 16 | 256 | 0.027638 |
| 16 | 8192 | 16x16 | 16 | 16 | 16 | 128 | 0.021172 |

Ο καλύτερος χρόνος επιτυγχάνεται με thread block 16x16 και tile ο πρώτος kernel 16x16, ενώ ο δεύτερος 16x128 . Σύμφωνα με το occupancy calculator ο πρώτος kernel έχει 100% occupancy, ενώ ο δεύτερος έχει 67%, διότι χρησιμοποιεί περισσότερους registers. Παρόλο όμως που το occupancy δεν ειναι 100% πετυγχάνουμε καλύτερους χρόνους.

Γενικά στην GTX 480 παρατηρήσα καλύτερους χρόνους με thread block 16x16 αντί για 32x32.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filter Radius | Block\_Tile | Image Size | GPU Time (Non-tiled) | GPU Time (Tiled) |
| 16 | 16x16 | 128 x 128 | 0.000029 | 0.000015 |
| 16 | 16x16 | 256 x 256 | 0.000065 | 0.000033 |
| 16 | 16x16 | 512 x 512 | 0.000237 | 0.000093 |
| 16 | 16x16 | 1024 x 1024 | 0.000971 | 0.000343 |
| 16 | 16x16 | 2048 x 2048 | 0.005957 | 0.001427 |
| 16 | 16x16 | 4096 x 4096 | 0.01682 | 0.005391 |
| 16 | 16x16 | 8192 x 8192 | 0.060491 | 0.021172 |

Παρατηρούμε ότι στην tile έκδοση των kernel έχουμε σημαντική βελτίωση του χρόνου εκτέλεσης των kernel, της τάξης του 65% . Η αύξηση αυτή είναι λογική εφόσον πλέον δεν κάνουμε load τα δεδομένα από την global memory κάθε φορά που τα χρησιμοποιούμε, αλλά τα κάνουμε μία φορά load στην shared memory και μετά τα παίρνουμε από την shared για να κάνουμε όλες τις πράξεις που χρειάζονται.

5)

* Για τον πρώτο kernel που διατρέχει την εικόνα κατά γραμμές, τα loads από global memory είναι ***image\_size2 \* 3***, ενώ για τον δεύτερο kernel που διατρέχει την εικόνα κατά στήλες είναι ***image\_size2 \* 1.25***. Τα loads από shared memory είναι ίδια και για τους 2 kernel ίσα με ***image\_size2 \* filter\_length*** και από constant memory ***image\_size2 \* filter\_length.***

Όσο αυξάνουμε το tile τόσο αυξάνονται και τα στοιχεία που αποθηκεύουμε στην shared memory. Βασικά για να μειώσουμε το λόγο των αναγνώσεων της shared προς την global θα πρέπει να αυξήσουμε τα οφέλιμα 16χ16 κομάτια που φορτώνουμε στην shared σχετικα με το padding που χρειάζεται για το φίλτρο, δηλαδή όπως κάναμε στον δεύτερο kernel να έχει μεγάλο ύψος.

* Ο λόγος προσπελάσεων μνήμης για τον πρώτο kernel ειναι

**image\_size2 \* 3 / image\_size2 \* 2 \* filter\_length,**

ενώ για τον δεύτερο kernel είναι

**image\_size2 \* 1.25 / image\_size2 \* 2 \* filter\_length.**

* Ο λόγος προσπελάσεων μνήμης ήτανε

image\_size2 \* 2 \* filter\_length / image\_size2 \* 2 \* filter\_length = 1.

Η διαφορά στους λόγους ειναι

image\_size2 \* 2 \* filter\_length - image\_size2 \* 3 για τον πρώτο kernel και image\_size2 \* 2 \* filter\_length - image\_size2 \* 1.25 για τον δεύτερο kernel,

δηλαδή με filter\_length = 33 ειναι ίση με 66 \* image\_size2 – 3 \* image\_size2 και

66 \* image\_size2 – 1.25 \* image\_size2 αντίστοιχα.

Η διαφορά στους λόγους είναι ιδιαίτερα εμφανή, αφού οι λόγοι στην tiled έκδοση του κώδικα βελτιώθηκαν κατά **22 φορές** για τον πρώτο kernel και **52 φορές** για τον δεύτερο kernel αντιστοίχα, σε σχέση με τους λόγους στον non-tiled κώδικα.

6) Ακολουθεί πίνακας και σχετικό διάγραμμα μετρήσεων χρόνων, βάση του tiled κώδικα, με ενεργοποιημένες και μη των cache L1 και L2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Filter Radius | Block\_Tile | Shared\_Tile | Image Size | GPU Time | dlcm=cg | dlcm=cs / dscm=wt |
| 16 | 16x16 | 16x16 / 16x16 | 128 x 128 | 0.000015 | 0.000015 | 0.000016 |
| 16 | 16x16 | 16x16 / 16x16 | 256 x 256 | 0.000033 | 0.000031 | 0.000036 |
| 16 | 16x16 | 16x16 / 16x16 | 512 x 512 | 0.000093 | 0.000093 | 0.000105 |
| 16 | 16x16 | 16x16 / 16x16 | 1024 x 1024 | 0.000343 | 0.000342 | 0.000428 |
| 16 | 16x16 | 16x16 / 16x16 | 2048 x 2048 | 0.001427 | 0.001374 | 0.00223 |
| 16 | 16x16 | 16x16 / 16x16 | 4096 x 4096 | 0.005391 | 0.005425 | 0.007166 |
| 16 | 16x16 | 16x16 / 16x16 | 8192 x 8192 | 0.021172 | 0.021387 | 0.026841 |

Αυτή τη φορά που χρησιμοποιούμε tiled kernel παρατηρούμε ότι όταν απενεργοποιούμε την L1 cache ο χρόνος δεν μεταβάλεται, αφού πλέον τα περισσότερα loads τα κάνουμε από την shared μνήμη και όχι απο την gobal όπως κάναμε στον non-tiled κώδικα.

Όταν απενεργοποιούμε και την L2 cache οι χρόνοι αυξάνονται κατά 30%. Η διαφορά με τον non-tiled κώδικα είναι εμφανής αφου τα loads από την global memory μειώνονται πάρα πολύ.

7) Ακολουθεί πίνακας και σχετικό διάγραμμα μετρήσεων χρόνων, βάση του tiled κώδικα, με αντικατάσταση των floats με doubles.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Filter Radius | Block\_Tile | Shared\_Tile | Image Size | GPU Time (floats) | GPU Time (doubles) |
| 16 | 16x16 | 16x16 / 16x16 | 128 x 128 | 0.000015 | 0.00003 |
| 16 | 16x16 | 16x16 / 16x16 | 256 x 256 | 0.000033 | 0.000084 |
| 16 | 16x16 | 16x16 / 16x16 | 512 x 512 | 0.000093 | 0.00027 |
| 16 | 16x16 | 16x16 / 16x16 | 1024 x 1024 | 0.000343 | 0.001006 |
| 16 | 16x16 | 16x16 / 16x16 | 2048 x 2048 | 0.001427 | 0.003959 |
| 16 | 16x16 | 16x16 / 16x16 | 4096 x 4096 | 0.005391 | 0.015635 |
| 16 | 16x16 | 16x16 / 16x16 | 8192 x 8192 | 0.021172 | out of memory |

Εφόσον κάθε double αντιστοιχεί σε 8 byte, ενώ κάθε float αντιστοιχεί σε 4 byte είναι λογικό ότι το tile που χρησιμοποιούμε χρειάζεται διπλάσια shared memory για να αποθηκευτεί. Το μέγιστο μέγεθος tile που μπορούμε να χρησιμοποιήσουμε είναι 64χ64 (δηλαδή

( 64 \* ( 64 + 16 \* 2 )) \* 8 byte shared memory)

έτσι ώστε να χρησιμοποιήσουμε και τα 48Κ shared μνήμης.