ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

Συστήματα Παράλληλης Επεξεργασίας
2018–2019

Άσκηση 4:
Παράλληλος προγραμματισμός σε επεξεργαστές γραφικών

Προθεσμίες παράδοσης

Επίδειξη προγραμμάτων	Φεβ. 2019
Τελική αναφορά	Φεβ. 2019[1] [2]

1 Πολλαπλασιασμός πίνακα με πίνακα

Ο πολλαπλασιασμός πίνακα με πίνακα (Dense Matrix-Matrix multiplication, DMM) είναι ένας από τους πιο σημαντικούς υπολογιστικούς πυρήνες αλγεβρικών υπολογισμών που συναντάται σε πληθώρα επιστημονικών εφαρμογών. Έστω οι μήτρες (δισδιάστατος πίνακας) εισόδου **A** και **B**. Ο πυρήνας DMM υπολογίζει τη μήτρα εξόδου $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$. Σε αναλυτική μορφή θεωρώντας τετραγωνικούς $N \times N$ πίνακες εισόδου, το γινόμενο γράφεται ως

$$C_{i,j} = \sum_{k=1}^{N} A_{ik} \cdot B_{kj}, \forall k \in [1, N]$$

2 Ζητούμενα

2.1 Υλοποίηση για επεξεργαστές γραφικών (GPUs)

Στην άσκηση αυτή θα πρέπει να υλοποιήσετε τον αλγόριθμο DMM για τους επεξεργαστές γραφικών του εργαστηρίου χρησιμοποιώντας το προγραμματιστικό περιβάλλον CUDA. Ξεκινώντας από μία απλοϊκή αρχική υλοποίηση, στο τέλος της άσκησης θα έχετε πετύχει μία αρκετά αποδοτική υλοποίηση του αλγορίθμου DMM για τους επεξεργαστές γραφικών.

Βασική υλοποίηση

Υλοποιήστε τον αλγόριθμο DMM, αναθέτοντας σε κάθε νήμα εκτέλεσης τον υπολογισμό ενός στοιχείου του πίνακα εξόδου.

1. Πειραματιστείτε με διάφορα μεγέθη μπλοκ νημάτων και καταγράψετε την χρησιμοποίηση των πολυεπεξεργαστικών στοιχείων (SMs) και την επίδοση του κώδικά σας.
2. Ποιες από τις προσβάσεις στην κύρια μνήμη συνενώνονται;

[1] filename a4-parlabXX-final.pdf
[2] mail to: athena@cslab.ece.ntua.gr, cc: goumas@cslab.ece.ntua.gr

Συνένωση των προσβάσεων στην κύρια μνήμη (memory access coalescing)

Τροποποιήστε κατάλληλα τον κώδικά σας ή/και τον τρόπο αποθήκευσης του πίνακα **A**, ώστε να επιτύχετε συνένωση των προσβάσεων στην κύρια μνήμη για τον πίνακα **A**.

1. Πειραματιστείτε με διάφορα μεγέθη μπλοκ νημάτων και καταγράψετε την χρησιμοποίηση των πολυεπεξεργαστικών στοιχείων (SMs) και την επίδοση του κώδικά σας.
2. Συγκρίνετε την επίδοση με αυτή της βασικής έκδοσης.
3. Ποιες από τις προσβάσεις στην κύρια μνήμη συνενώνονται;

Χρήση της τοπικής on-chip μνήμης

Χρησιμοποιείστε την τοπική μνήμη των πολυεπεξεργαστικών στοιχείων (shared memory), ώστε να προφορτώνετε τμηματικά τους πίνακες εισόδου **A** και **B** σε αυτή και στην συνέχεια να εκτελείτε τους υπολογισμούς σε αυτό. Θα πρέπει να προσέξετε, οι προσβάσεις στην κύρια μνήμη να μπορούν να συνενωθούν, ώστε να μπορέσετε να αποκομίσετε οφέλη από αυτή την βελτιστοποίηση.

1. Πειραματιστείτε με διάφορα μεγέθη μπλοκ νημάτων και καταγράψετε την χρησιμοποίηση των πολυεπεξεργαστικών στοιχείων (SMs) και την επίδοση του κώδικά σας.
2. Συγκρίνετε την επίδοση με αυτή της προηγούμενης έκδοσης.
3. Είναι χρήσιμο τα στοιχεία του C να προφορτώνονται στην τοπική μνήμη? Αν όχι, πού;

Χρήση της βιβλιοθήκης cuBLAS

Τροποποιήστε κατάλληλα τον κώδικά σας ώστε να χρησιμοποιήσετε την συνάρτηση `cublasSgemm()` της βιβλιοθήκης cuBLAS για την υλοποίηση του ζητούμενου πολλαπλασιασμού. Η βιβλιοθήκη cuBLAS αποτελεί υλοποίηση της BLAS για τις κάρτες γραφικών της NVIDIA. Χρησιμοποιήστε τις κατάλληλες παραμέτρους για τους βαθμωτούς `alpha` και `beta` που απαιτεί η συνάρτηση. Επιπλέον, διαβάστε προσεκτικά πως θεωρεί η βιβλιοθήκη cuBLAS ότι είναι αποθηκευμένα τα μητρώα στην μνήμη για να καθορίσετε την σωστή τιμή της παραμέτρου `trans`.

1. Καταγράψετε την επίδοση και συγκρίνετέ την με την επίδοση του κώδικα που αναπτύξατε στα προηγούμενα ερωτήματα της εργασίας.

3 Υποδείξεις και διευκρινίσεις

3.1 Δομή κώδικα

Για την διευκόλυνσή σας, αλλά και για να υπάρχει ένας κοινός τρόπος μέτρησης του χρόνου εκτέλεσης, σας δίνεται πλήρης και λειτουργικός σκελετός του κώδικα της άσκησης, καθώς και η ρουτίνα της σειριακής εκτέλεσης του DMM. Ο κώδικας βρίσκεται στον scirouter, στο φάκελο `/home/parallel/pps/2018-2019/a4/dmm-skeleton` και αποτελείται από δύο υποφακέλους `/common` και `/cuda`. Εσείς θα κάνετε προσθήκες στον υποφάκελο `/cuda` ο οποίος περιέχει:

Κυρίως πρόγραμμα: Πρόκειται για το αρχείο `dmm_main.cu`, το οποίο περιέχει την συνάρτηση `main()` του προγράμματος, η οποία είναι υπεύθυνη (α′) για την ανάγνωση των ορισμάτων της γραμμής εντολών, (β′) για την δημιουργία των πινάκων εισόδου/εξόδου, (γ′) για την αρχικοποίηση (παραχωρήσεις μνήμης, παράμετροι πυρήνα), εκτέλεση και χρονομέτρηση του πυρήνα στην GPU, και (δ′) για τον έλεγχο της εγκυρότητας των αποτελεσμάτων.

Υλοποιήσεις για CPU: Πρόκειται για το αρχείο `dmm.c`, το οποίο περιέχει την σειριακή υλοποίηση του DMM για τις CPUs.

Υλοποιήσεις για GPU: Πρόκειται για το αρχείο `dmm_gpu.cu`, το οποίο περιέχει τις υλοποιήσεις των πυρήνων για GPUs.

Βοηθητικά αρχεία: Πρόκειται για τα αρχεία `mat_util.c` και `gpu_util.cu`, τα οποία περιέχουν βοηθητικές συναρτήσεις για το χειρισμό δισδιάστατου πίνακα και για το χειρισμό των GPUs αντίστοιχα.

Σας παρέχονται επιπλέον και τα κατάλληλα `Makefiles` για την μεταγλώττιση και την σύνδεση του κώδικά σας. Πληκτρολογείστε '`make help`', ώστε να δείτε τις διάφορες επιλογές που σας δίνονται κατά την μεταγλώττιση. Η προεπιλεγμένη ρύθμιση για την μεταγλώττιση είναι σε λειτουργία debug (όλες οι βελτιστοποιήσεις του μεταγλωττιστή απενεργοποιημένες).

Τα σημεία του κώδικα, στα οποία θα πρέπει να επέμβετε είναι σημειωμένα με την επιγραφή '`FILLME:`' μαζί με μία σύντομη περιγραφή. Οι ζητούμενες προσθήκες θα γίνουν στα αρχεία `dmm_gpu.cu` και `dmm_main.cu`. Τέλος, για να δείτε τον τρόπο χρήσης του τελικού εκτέλεσιμου, εκτελέστε '`./dmm_main`' από την γραμμή εντολών και θα παρουσιαστεί ένα σύντομο μήνυμα βοήθειας για την σωστή χρήση του.

3.2 Περιβάλλον ανάπτυξης

Θα τρέξετε τον κώδικά σας σε μηχάνημα του εργαστηρίου (`termis`) με εγκατεστημένη κάρτα γραφικών γενιάς 2.0 (NVIDIA Tesla M2050, αρχιτεκτονική Fermi). Για περισσότερες πληροφορίες σχετικά με τα λεπτομερή τεχνικά χαρακτηριστικά της GPU, πληκτρολογείστε '`make query`' όντας σε ένα μηχάνημα termi. Η χρήση των υπολογιστών του εργαστηρίου (ουρά termis) για την εκτέλεση της άσκησης θα γίνεται μέσω του συστήματος υποβολής εργασιών Torque, όπως και στις προηγούμενες ασκήσεις. Για την εκτέλεση των μετρήσεων σε πραγματική GPU, θα πρέπει να δεσμεύσετε το κατάλληλο μηχάνημα από το σύστημα Torque ως εξής:

```
$ qsub -q termis -l nodes=1:ppn=24:cuda myjob.sh
```

4 Πειράματα και μετρήσεις επιδόσεων

4.1 Σενάριο μετρήσεων και διαγράμματα

Σκοπός των μετρήσεων είναι (α′) η μελέτη της επίδρασης στην επίδοση του μεγέθους του μπλοκ για τις υλοποιήσεις σε GPUs και (β′) η σύγκριση της επίδοσης των διαφόρων εκδόσεων του πυρήνα DMM για GPUs. Συγκεκριμένα, σας ζητούνται τα παρακάτω σύνολα μετρήσεων:

- Για κάθε έκδοση πυρήνα που υλοποιήσατε να καταγράψετε πώς μεταβάλλεται η επίδοση για διαφορετικές διαστάσεις του thread block για διαστάσεις πινάκων $M = N = K = 1024$.
- Να καταγράψετε την επίδοση για μεγέθη πινάκων $M, N, K \in [256, 512, 1024, 2048]$ για κάθε μία από τις τέσσερις εκδόσεις πυρήνων (naive, coalesced, shmem και cuBLAS) που υλοποιήσατε αφού επιλέξετε το καλύτερο thread block για κάθε υλοποίηση (με εξαίρεση την cuBLAS στην οποία δεν έχετε κανένα έλεγχο).

Για την διευκόλυνσή σας, σας δίνεται ένα πρωτότυπο script μετρήσεων `run_dmm.sh` το οποίο μπορείτε να χρησιμοποιήσετε ως βάση. Στην τελική αναφορά σας για την άσκηση, σας ζητείτε να ερμηνεύσετε (σύντομα) την συμπεριφορά της επίδοσης του DMM με βάση τον κώδικα σας σε GPUs.

ΠΡΟΣΟΧΗ: Στο Makefile που σας δίνεται, η προεπιλεγμένη ρύθμιση για την μεταγλώττιση είναι σε λειτουργία debug (όλες οι βελτιστοποιήσεις του μεταγλωττιστή απενεργοποιημένες). Για να μετρήσετε τις επιδόσεις των πυρήνων θα πρέπει να απενεργοποιήσετε αυτή την λειτουργία και να μεταγλωττίσετε εξαρχής ('`make clean`') τον κώδικά σας με '`make DEBUG=0`'.