ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Παράλληλες Αρχιτεκτονικές Υπολογισμού για Μηχανική Μάθηση

Ε.ΔΕ.Μ²

Ακαδημαϊκό Έτος 2019-20

http://www.cslab.ece.ntua.gr/courses/parml

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3
Παράλληλη Επεξεργασία σε GPUs με CUDA

1 Εισαγωγικά

Στα πλαίσια αυτής της εργασίας θα εξοικειωθείτε με τον παράλληλο προγραμματισμό σε GPUs μέσα από το προγραμματιστικό περιβάλλον NVIDIA CUDA. Ο βοηθητικός κώδικας της εργασίας βρίσκεται στον scirouter στον κατάλογο `/home/parml/shared/lab3_cuda`. Ο κατάλογος περιέχει 3 υποκαταλόγους, έναν για κάθε άσκηση: `ex1`, `ex2`, `ex3`. Για κάθε άσκηση, θα βρείτε και τα αντίστοιχα torque scripts για τη μεταγλώττισή και εκτέλεσή τους. Η μεταγλώττιση και εκτέλεση θα γίνει στα μηχανήματα `termi`, που ανήκουν στην ουρά `termis`.

2 Ζητούμενα

2.1 "Hello World!"

Στον κατάλογο `ex1` σας δίνεται το αρχείο `hello_world.cu`. Στόχος της άσκησης είναι:

1. να συμπληρώσετε τον κώδικα ενός νήματος (device code) ώστε να τυπώνει το καθολικό αναγνωριστικό του μέσα στο πλέγμα (global thread id)
2. να ορίσετε εκτέλεση με 64 νήματα (host code)
3. να μεταγλωττίσετε τον κώδικα
   - `$ nvcc -O3 hello_world.cu -o helloworld`
4. και να τρέξετε το πρόγραμμα.
   - `$ ./helloworld`

Ερώτηση: Ποιος ο ρόλος του cudaDeviceSynchronize();

2.2 Πρόσθεση Διανυσμάτων

Στον κατάλογο `ex2` σας δίνεται υλοποίηση για CPU σε C στο αρχείο `vector_add.c`. Επιπλέον, σας δίνεται αρχική υλοποίηση σε CUDA στο αρχείο `vector_add.cu`. Αρχικός στόχος της άσκησης είναι:

1. Να μεταγλωττίσετε τους κώδικες
    - `$ gcc -O3 vector_add.c -o vecadd_cpu`
    - `$ nvcc -O3 vector_add.cu -o vecadd_gpu`
2. Να τρέξετε τα προγράμματα
    - `$ time ./vecadd_cpu 4096`
    - `$ time ./vecadd_gpu 4096`

Ερώτηση 1: Ποιά η έξοδος του προγράμματος; Τι συνέβη; Τροποποιήστε κατάλληλα τον κώδικα ώστε να εκτελείται σωστά. Συγκεκριμένα, διαχειριστείτε τη μεταφορά δεδομένων από/προς τη GPU.

Ερώτηση 2: Τι παρατηρείτε ως προς τον χρόνο εκτέλεσης σε CPU και GPU; Γιατί ο χρόνος εκτέλεσης σε GPU είναι απογοητευτικά αργός; Τροποποιήστε κατάλληλα τον κώδικα ώστε να βελτιώσετε την επίδοση στη GPU.

Ερώτηση 3: Εκτελέστε τα προγράμματα για διαφορετικά μεγέθη διανυσμάτων στο εύρος 128-8192. Τι παρατηρείτε ως προς την επίδοση σε CPU και GPU; Υπάρχει επιτάχυνση στη GPU;

2.3 Εσωτερικό Γινόμενο Διανυσμάτων

Στον κατάλογο `ex3` σας δίνεται ελλιπής υλοποίηση σε CUDA στο αρχείο `dot_product.cu` Στόχος της άσκησης είναι:

1. να υλοποιήσετε το εσωτερικό γινόμενο χωρίς χρήση της κοινής μνήμης (shared memory)
2. να υλοποιήσετε το εσωτερικό γινόμενο κάνοντας χρήση της κοινής μνήμης
3. και να συγκρίνετε επιδόσεις.

Ερώτηση: Υπάρχει βελτίωση της επίδοσης με χρήση της κοινής μνήμης;

3 Υποδείξεις και διευκρινίσεις

3.1 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, μπορείτε να εκτελέσετε το πρόγραμμα `deviceQuery` ποιυ βρίσκεται στον κάταλογο `/usr/local/cuda/samples/1_Utilities/deviceQuery` 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. Ενδεικτικά, για την υποβολή ενός script στην ουρά `termis` χρησιμοποιούμε:

```
$ qsub -q termis -l nodes=termi3:ppn=1:cuda compile_and_run_on_termis.sh
```