Multiprocesorski sistemi

Domaći zadatak 4
CUDA – osnove
(10 poena)

Uvod

Cilj zadatka je da studente obuči da samostalno razvijaju osnovne CUDA programe.

Podešavanje okruženja

Detaljna uputstva za instaliranje, podešavanje i prvo izvršavanje CUDA programa se mogu naći na adresi http://developer.nvidia.com/nvidia-gpu-computing-documentation ili na sajtu predmeta pod nazivom CUDA Getting Started Guide (Windows) ili CUDA Getting Started Guide (Linux) u zavisnosti koji operativni sistem se koristi. Po tom uputstvu podesiti okruženje za razvoj i kontrolisano izvršavanje (engl. debugging) CUDA programa na lokalnom računaru. Alternativno, koristiti CUDA (**`nvcc`**) na računaru **`rtidev5.etf.rs`**. Prevodilac se nalazi u direktorijumu: **`/usr/local/cuda/bin/`**.

Zadaci

Svi programi treba da koriste GPU za bilo koju obradu. Smatrati da je broj GPU niti na nivou jednog bloka niti određen konstantom **`NUM_OF_GPU_THREADS`**, čija je vrednost za sve zadatke 256. Obezbediti da niti koje u nekom koraku nemaju posla na korektan način stignu do kraja tela CUDA jezgra.

Korisnik zadaje samo dimenzije nizova/matrica, a sve potrebne ulazne podatke generisati u operativnoj memoriji uz pomoć generatora slučajnih brojeva iz biblioteke jezika C, a zatim prebaciti u GPU memoriju. Generisani brojevi treba da budu odgovarajućeg tipa u opsegu od **`-MAX`** do **`+MAX`**, gde **`MAX`** ima vrednost 1024. Za sve zadatke je potrebno napisati ili iskoristiti zadatu sekvencijalnu (CPU) implementaciju odgovarajućeg problema koja će biti korišćena kao referentna (*gold*) implementacija prilikom testiranja programa. Dostupne sekvencijalne implementacije se nalaze u arhivi **`MPS_DZ4_CUDA.zip`** ili **`MPS_DZ4_CUDA.tar`** koje se mogu preuzeti na adresi http://mups.etf.rs/dz/2013-2014/. Dozvoljeno je ograničeno preuređivanje dostupnih sekvencijalnih implementacija prilikom paralelizacije. Svaki program treba da:

- Generiše ulazne test primere.
- Kopira test primere u GPU memoriju i rezultat iz GPU memorije.
- Izvrši CUDA jezgro nad zadatim test primerom.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Uporedi rezultat CUDA i sekvencijalne implementacije problema.
- Ispiše vreme izvršavanja CUDA i sekvencijalne implementacije problema.
- Ispiše "**`Test PASSED`**" ili "**`Test FAILED`**" u zavisnosti da li se rezultat izvršavanja CUDA implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Kod zadataka koji koriste realne tipove (**`float`**, **`double`**) tolerisati maksimalno odsupanje od **`±ACCURACY`** prilikom poređenja rezultata CPU i GPU implementacije. Smatrati da konstanta **`ACCURACY`** ima vrednost 0.01.

1. Paralelizovati program koji vrši računanje aditivne perzistencije za svaki element niza celih brojeva. Aditivna perzistencija nekog celog broja je jednaka broju koraka koji je potreban da se početna vrednost tog broja svede na jednocifren broj uzastopnim računanjem zbira cifara (http://en.wikipedia.org/wiki/Persistence_of_a_number). Sekvencijalni program se nalazi u datoteci **`persistence.c`** u arhivi koja je priložena uz ovaj dokument.
2. Izmeniti prethodni program tako da se umesto niza celih brojeva koristi dvodimenzionalna matrica celih brojeva proizvoljnih dimenzija. Prilikom zadavanja izvršne konfiguracije jezgra, koristiti 2D rešetku (*grid*).
3. Paralelizovati program koji vrši množenje dve realne matrice proizvoljnih dimenzija. Izvršiti proveru da li se zadate dve matrice mogu pomnožiti. Sekvencijalni program koji radi samo sa kvadratnim matricama se nalazi u datoteci **`matMul.c`** u arhivi koja je priložena uz ovaj dokument.

4. Paralelizovati program koji računa skalarni proizvod dva realna vektora. Sekvencijalni program se nalazi u datoteci **`dotProduct.c`** u arhivi koja je priložena uz ovaj dokument. Prilikom rešavanja zadatka koristiti deljenu memoriju za smeštanje međurezultata.
5. Paralelizovati program koji vrši množenje zadate matrice vektorom. Sekvencijalni program se nalazi u datoteci **`matVecMul.c`** u arhivi koja je priložena uz ovaj dokument. Optimizovati program korišćenjem deljene memorije na nivou računanja skalarnog proizvoda jedne vrste matrice i zadatog vektora.

Važno: Ukoliko u nekom zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku i da nastavi da izgrađuje svoje rešenje na temeljima uvedene pretpostavke.