Programowanie w CUDA na NVIDIA GPU (PKG) rok akademicki 2021/2022,
studenci Informatyki dziennej 3 rok 6 semestr

Projekt 2 z Laboratorium z przetwarzania równoległego

Termin oddania: 4 tygodnie od 5 terminu zajęć laboratoryjnych; dla zajęć 11.05.2022 (18.05.2022) termin mija 15.06.2022 (22.06.2022) o godz. 23.59. **Nieuzasadnione opóźnienie** w oddaniu sprawozdania obniża ocenę w stosunku do oceny przyznanej za jakość sprawozdania. Obniżają ocenę o ½ stopnia następujące opóźnienia - po terminie, po tygodniu, po 2 tygodniach, po 3 tygodniach). Przekroczenie terminu oddania o 4 tygodnie (lub terminu 20 lipca 2022) powoduje wystawienie oceny na podstawie wcześniej oddanych opracowań z przedmiotu. Wystawienie oceny ndst (<50 % możliwych punktów) z pierwszego terminu zaliczenia przedmiotu powoduje konieczność indywidualnego wystąpienia Studenta o wydanie nowego zadania zaliczeniowego.

Grupy projektowe: projekt jest wykonywany i zaliczany w grupach maksymalnie 2 osobowych. Skład grupy projektowej proszę zgłaszać podczas zajęć dnia 11.05.2022 (18.05.2022) .

Koncepcja projektu

Celem ćwiczenia jest:

- zapoznanie Studentów praktyczne z zasadami programowania równoległego procesorów kart graficznych (PKG)
- zapoznanie z zasadami optymalizacji kodu dla PKG,
- tworzenie funkcji dla PKG dotyczących problemu mnożenia macierzy oraz
- ocena prędkości przetwarzania przy użyciu PKG oraz poznanie czynników warunkujących realizację efektywnego przetwarzania.

W ramach projektu należy:

- przygotować i wyjaśnić przebieg przetwarzania wymaganych wersji programów,
- wykonać eksperyment obliczeniowy dla przygotowanych kodów z pomiarem czasu przetwarzania dla zadanego zakresu instancji i parametrów uruchomienia,
- wykonać eksperyment pomiaru efektywności przetwarzania przy użyciu programu oceny efektywności przetwarzania Night Compute
- wykorzystać uzyskane wyniki obliczeń i miary efektywności (wskazane w opisie projektu/wybrane przez autorów opracowania i kodu) do porównania jakości przetwarzania i udowodnienia, że wymagany zakres projektu został wykonany,
- przygotować sprawozdanie wg opisanych poniżej wymagań.

Różnorodność sprawozdań i dostępność opisu zadania

Ze względu na wąski zakres różnorodności tematów zadań, pewne podobieństwa w realizacji projektu przez poszczególne grupy realizujące ten sam temat są nieuniknione. Ze względu na szeroki zakres docelowego sprzętu, badanych instancji problemu i możliwych rozwiązań implementacyjnych liczę na różnorodne opracowania w ramach poprawnego wyjaśniania przyczyn i uzasadniania uzyskanych wyników. Tą drogą zwracam się z apelem o samodzielną pracę nad zadaniami, gdyż pozwoli ona na zrozumienie zagadnień programowania równoległego na PKG. Proszę odpowiednio wcześnie podjąć pracę nad projektem, aby można ją było wykonać w wymaganym terminie samodzielnie.

Opis zadania został opublikowany 11.05.2020. W przypadku potrzeby konsultacji proszę o kontakt przez email (konsultacje zdalne w poniedziałki godz. 16.45)

Temat projektu i warianty

Mnożenie macierzy - Porównanie efektywności przetwarzania dla programów CPU i GPU, optymalizacja i porównanie przebiegu przetwarzania na procesorach kart graficznych.

Program CPU – równoległe przetwarzanie metodą zagnieżdżonych 3 pętli - kolejność zagnieżdżenia zmiennych IKJ (oznaczenia wg wykładu z analizy jakości kodu mnożenia macierzy w systemach z pamięcią współdzieloną) uruchamiana na posiadanym przez autora projektu komputerze z procesorem wielordzeniowym. Wyniki tego przetwarzania są punktem odniesienia przy porównaniu z efektywnością przetwarzania na karcie graficznej.

Wariant pierwszy projektu (kluczowe elementy: synchroniczna komunikacja host-device, optymalizacja liczby wyników obliczanych przez wątek, pamięć współdzielona bloku wątków)

Program GPU –
Modyfikacja udostępnionego kodu mnożenia macierzy (kody przykładowe Nvidia) pod kontem:
- liczby wyników obliczanych przez wątek obliczeniowy oraz
- zrównoleglenia operacji realizowanych przez jeden blok wątków: pobierania danych do pamięci współdzielonej i wykonywanie obliczeń.

Badanie wpływu jak liczba wartości (tablicy wynikowej) obliczanych przez każdy z wątków uruchomionych na karcie graficznej wpływa na prędkość przetwarzania. W kodzie przykładowym dostarczanym przez Firmę Nvidia każdy watek liczy jeden wynik w oparciu o bloki danych sprowadzane wcześniej do pamięci współdzielonej. W przypadku wzrostu liczby wyników obliczanych przez wątek wzrasta też liczba danych w pamięci współdzielonej bloku wątków (część tablicy – podtablica o rozmiarze BSxBS) niezbędnych do wykonania tych obliczeń. Wzrost liczby wyników obliczanych przez wątek powoduje konieczność wzrostu liczby danych pobieranych przez wątki w każdym etapie przetwarzania do pamięci współdzielonej. Zależność pomiędzy liczbą wartości obliczanych przez wątek i ilością danych sprowadzanych w każdym kroku do pamięci współdzielonej jest następująca:

1. Wątek oblicza jeden wynik, ilość danych: [BS,BS] podtablica A, [BS,BS] podtablica B (wersja oryginalna kodu)
2. Obliczenia przez wątek K*L wyników [K*BS,BS] podtablica A, [BS,L*BS] podtablica B

- Określane w ten sposób przypadki posiadają ograniczenie na wielkość sprowadzanych danych do pamięci współdzielonej polegające na tym, że liczba kolumn podtablicy A jest równa liczbie wierszy podtablicy B i jest równa rozmiarowi dwu-wymiarowego kwadratowego bloku wątków [BS,BS]
- Wzrost liczby wyników obliczanych przez każdy wątek powoduje niekorzystny spadek maksymalnego poziomu równoległości określanego liczbą równocześnie przetwarzanych bloków wątków (wymagania zasobowe) oraz korzystny wzrost wartości parametru CGMA (ponowne użycie danych raz sprowadzonych do pamięci współdzielonej bloku wątków). W konsekwencji zmienić może się prędkość przetwarzania ze względu na zmiany w organizacji przetwarzania – należy wskazać pozytywne i negatywne konsekwencje zmiany liczby wyników (wyznaczanych przez każdy wątek) dla efektywności przetwarzania na kartach graficznych. Podczas organizacji przetwarzania należy zadbać o efektywny sposób dostępu do danych zarówno z pamięci globalnej jak i pamięci współdzielonej.

Zadanie szczegółowe projektu: jaka liczba wyników liczona w powyżej określony sposób umożliwia uzyskanie maksymalnej prędkości obliczeń w dostępnym systemie obliczeniowym?
Ocena projektu będzie uzależniona od poprawności kodu i eksperymentu oraz zakresu analiz wyników przetwarzania dla różnych wartości parametrów K i L. Na zaliczenie projektu na ocenę dostateczną wystarczy zbadać i ocenić uruchomienia dla 2 par wartości parametrów K i L gdzie K<>L<>1.

Wariant drugi projektu (kluczowe elementy: użycie techniki dostępu do danych - **0copy**, pamięci globalnej karty graficznej jako bufor danych jednokrotnie pobieranych z PO CPU przez wątki gridu, asynchroniczna komunikacja HOST-DEVICE, porównanie jakości przetwarzania przy wykorzystaniu: a) pamięci globalnej i pamięci podręcznej karty oraz b) dodatkowo pamięci współdzielonej bloku wątków)

Program GPU –
Modyfikacja udostępnionego kodu mnożenia macierzy (kody przykładowe Nvidia) pod kontem jednokrotnego asynchronicznego pobierania danych na żądanie z pamięci operacyjnej (obszary danych o stałej lokacji – bez realokacji).
Badanie wpływu wielkości macierzy i wielkości bloku wątków na prędkość przetwarzania. W kodzie przykładowym dostarczanym przez Firmę Nvidia każdy watek liczy jeden wynik w oparciu o bloki danych sprowadzane wcześniej do pamięci współdzielonej. Należy dodatkowo przygotować kod, w którym wykorzystywane będą dane pobierane przez wątki z pamięci globalnej bez użycia pamięć współdzielonej. Pobranie danych wejściowych wymaga synchronizacji wątków wykorzystujących te same dane w oczekiwaniu na sprowadzenie ich do pamięci globalnej.

Instancje: mnożenie tablic kwadratowych C=AxB o wymiarach o rozmiarach NxN : 1024x1024, 3072x3072 lub podobnych dostosowanych wielkością do prostoty i czytelności kodu (przy porównaniu efektywności dla wariantów kodu dokładna wielkość instancji nie ma znaczenia gdyż porównywane są prędkości przetwarzania macierzy i porównanie może dotyczyć instancji o podobnym rozmiarze). Eksperymenty proszę wykonać dla bloków wątków o rozmiarach 8x8, 16x16 i 32x32 wątki (jeśli w użytej karcie graficznej jest to możliwe).

Na podstawie parametrów uruchomienia i czasu należy obliczyć:

- prędkość obliczeń uwzględniającą liczbę operacji zmiennoprzecinkowych wg złożoności algorytmu mnożenia macierzy o wielkości NxN, prędkość=2*N*N*N/T gdzie T jest czasem jednokrotnego uruchomienia kernela dla danej konfiguracji, wariantu kodu i instancji,
- przyspieszenie w stosunku do najlepszego dostępnego przetwarzania równoległego (openMP) za pomocą CPU (lab 2.7.6),
- CGMA – obliczony w oparciu o analizę teoretyczną przygotowanego kodu i informację bazująca o wyniki pracy Night Compute,
- zajętość multiprocesora za pomocą CUDA Occupancy Calculator tool dostępnego w dystrybucji oprogramowania, przedstawiać przyczyny obniżonej wartości zajętości (wielkość instancji, liczba bloków, wymagania rejestrowe, wymagania na pamięć współdzieloną). Do wyznaczenia liczby rejestrów wykorzystywanych przez wątek proszę skorzystać z programu oceny efektywności przetwarzania NVIDIA Nsight Compute.

Parametry uruchomień proszę przedstawiać w sprawozdaniu w sposób zwarty (łatwy do porównań wartości) w jednej tabeli wraz z parametrami uruchomienia N (rozmiar kwadratowych NxN mnożonych tablic), BLOK (rozmiar tablicy wątków kwadratowego BLOKxBLOK bloku wątków) i czytelnym wariantem wersji kodu.
Porównanie efektywności rozwiązań CPU i GPU powinno oprócz prędkości uwzględniać dodatkowo koszt obliczeń mierzony za pomocą:

- parametru energetycznego **TDP** (ang. *Thermal Design Power*),
- technologii (Lithography) i
- powierzchni krzemu układu scalonego CPU/GPU (Si area in mm^2)

jako parametrów kosztu użycia i produkcji układu obliczeniowego.

Dokumentacja

Po wykonaniu zadania proszę dostarczyć materiały **POPRZEZ ZAKŁADKĘ PRZESYŁANIA OPRACOWAŃ** kursu Przetwarzanie Równoległe PROJEKT 2. Należy dostarczyć:

- plik 2P_NR_INDEKSU1_NR_INDEKSU2.zip - archiwum z kodami źródłowymi
- plik 2P_NR_INDEKSU1_NR_INDEKSU2.pdf - sprawozdanie zawierające:

1. dane autorów (imiona i nazwiska, numery indeksu) i data oddania
2. nazwa przedmiotu i nazwa projektu,
3. **opis użytej karty graficznej** –
   - nazwa modelu karty i użyty w nim układ scalony,
   - nazwa technologii: Fermi, Maxwell, Volta itp,
   - parametr CC – możliwości obliczeniowe,
   - liczba jednostek wykonawczych przeznaczonych do obliczeń ogólnego przeznaczenia: liczba SM, rdzenie, jednostki zmiennoprzecinkowe SP i DP, jednostki całkowitoliczbowe, SFU,
   - wielkości i rodzaje pamięci karty używanej podczas obliczeń,
   - ograniczenia posiadanego przez kartę CC.
4. opis zakresu zrealizowanego zadania (przeprowadzone analizy realizowanego zadania)
5. kluczowe fragmenty kodów kerneli z **wyjaśnieniami** dotyczącymi:
   - znaczenia użytych instrukcji i zmiennych,
   - określenie i **uzasadnienie jakości występującego w kodzie dostępu do użytej pamięci** (odwołanie się w wyjaśnieniach do pojęć: łączenia dostępów do pamięci globalnej i minimalizacji konfliktów w dostępie do banków pamięci współdzielonej),
   - znaczenia poszczególnych synchronizacji występujących w kodzie
   - ilości przesyłanych danych pomiędzy pamięciami karty,
   - opis sposobu określania konfiguracji uruchomienia kernela.
6. rysunki z opisem określające:
   - miejsce dostępu i kolejność dostępu do danych realizowane przez poszczególne wątki, bloki i
   - wyznaczane przez wątki i bloki wartości wyników,
7. wzory zastosowane do obliczeń wszystkich prezentowanych miar efektywności przetwarzania wraz z wyjaśnieniem znaczenia tych miar,
8. wartości miar efektywności dostępne w ramach programów oceny efektywności i **informacje o znaczeniu** tych miar dostępnych w programie NVIDIA Visual Profiler / NsightCompute pozwalających na ocenę/porównanie efektywności przetwarzania. Szczególne znaczenie maja miary stopnia wykorzystania modułów SM, pamięci i zajętości teoretycznej oraz praktycznej multiprocesora.
9. wymagane wyniki jakości przetwarzania dla poszczególnych zbadanych instancji proszę przedstawić w postaci tabelarycznej (najlepiej jedna tabela - orientacja portrait – kolumny tabeli wzdłuż osi dłuższej kartki), tabele (i wykresy) należy ponumerować i podpisać w sposób unikalny definiujący zawartość obiektu,
10. wnioski z wykonanych eksperymentów z uzasadnieniem (przy pomocy takich pojęć jak CGMA, zajętość multiprocesora, stopień łączenia dostępów do pamięci globalnej, konflikty w dostępie do pamięci współdzielonej, synchronizacja wątków, ilość pracy wątków, liczba bloków wątków) obserwowanych wartości

(konieczne czytelne odwołanie we wnioskach do omawianej wielkości parametrów w tabeli wyników i jej wartości - jakiego uruchomienia - system, parametry, wersja kodu – dotyczą).

Zaliczanie projektu będzie wymagało wiadomości z zakresu przetwarzania równoległego na PKG, a w szczególności zagadnień związanych z projektem.

INFORMACJE UZUPEŁNIAJĄCE – PRZYGOTOWANIE SYSTEMU

1. Eksperymentu można wykonać w laboratorium lub przy użyciu dowolnej karty NVIDIA posiadającej klasę CC>=1.3 W zależności od dostępnej karty proszę użyć **najnowszą wersję** systemu CUDA, który wspomaga dostępne urządzenie: lista wersji oprogramowania dostępna pod adresem: https://developer.nvidia.com/cuda-toolkit-archive oraz właściwe dla tego środowiska i karty programy uruchomieniowe.
2. **Tworzenie projektu CUDA dla Visual Studio** (przygotowanie środowiska testowego dla własnej karty). Sposób postępowania przy przygotowywaniu środowiska testowego:
    a. Dla posiadanej karty NVIDIA znaleźć współpracujący z nią najnowszy pakiet SDK CUDA,
    b. Wybrać najnowsze VISUAL STUDIO, które z wybranym pakietem SDK CUDA współpracuje.
    c. Tworzenie projektu dla PKG w Visual studio wymaga takiego skonfigurowania Visual studio, aby posiadało zasady kompilacji projektu CUDA (Build Customizations for CUDA Projects) dzieje się to automatycznie podczas instalacji pakietu CUDA współpracującego z posiadaną wersją Visual studio. Projekt w wersji dla karty graficznej powstaje przy wykorzystaniu opcji File-> New | Project... NVIDIA-> CUDA->, a następnie wybraniu wzorca CUDA Toolkit version - np. "CUDA versja Runtime". Wzorzec pozwoli na skonfigurowanie projekt do wykorzystania CUDA wersja Toolkit. Nowy projekt jest projektem C++ lecz skonfigurowanym do użycia NVIDIA's Build Customizations. Wszystkie własności projektów Visual Studio C++ projects będą nadal dostępne. Przy korzystaniu z starszych kart graficznych potrzebne będzie określenie w właściwościach projektu Visual Studio informacji o wersji generowanego kodu
3. Przy tworzenie kodu dla GPU proszę bazować na projekcie matrixMul dostępnym w ramach "CUDA SDK code samples". Po zrozumieniu zawartości udostępnionego kodu warto skorzystać z niego jako wzorca takich działań jak: pomiar czasu obliczeń, przygotowanie danych i sprawdzenie poprawności obliczeń.
4. Program oceny efektywności przetwarzania to obecnie NsightCompute a we wcześniejszych wersjach systemu Nvidia profiler. Dokumentacja do oprogramowania jest dostępna w pakiecie narzędzi CUDA SDK.

NVIDIA Visual Profiler / NsightCompute
Praca z programem oceny efektywności kodu wymaga:

- utworzenia kodu wynikowego badanej aplikacji w VisualStudio,
- uruchomienia **NVIDIA Visual Profiler /** NsightCompute
- utworzenia sesji oceny efektywności (profilowania), wskazania pliku uruchamianego kodu, wskazania pliku wyników profilowania, określenia zakresu zbieranych miar jakości przetwarzania.
- uruchomienia procesu oceny aplikacji.

Podczas wielokrotnych uruchomień programu zbierane będą statystyki i mierzone czasy realizacji poszczególnych funkcji przetwarzania dotyczącego GPU. Wyniki profilowania dostępne są w poszczególnych widokach prezentacji miar efektywności. Widoki wyników prezentowane są podczas zajęć laboratoryjnych dostępne w oddzielnym pliku materiałów oraz zostały opisane w dokumentacji do oprogramowania - Nsight Compute profiling guide

Podstawowe miary jakości przetwarzania z Nsight Compute

proponowane do wykorzystania podczas oceny jakości przetwarzania przygotowanych i uruchamianych kodów:
Czas obliczeń – Duration [sekundy]
Wydajność obliczeń - Performance [flop/s]
CGMA- arithmetic intensity [flop/byte] – uwzględnia kod i użycie pamięci współdzielonej
Przepustowość obliczeń SM – Compute (SM) throughput[%]
Przepustowość systemu pamięci – Memory throughput[%]
L1 hit rate
L2 hit rate
Wielkość transmisji z PG – odczyt [GB/ % rozmiaru instancji]
Wielkość transmisji z PG – zapis [GB/ % rozmiaru instancji]
Wielkość transmisji z PW – odczyt [GB/ % rozmiaru instancji]
Wielkość transmisji z PW – zapis [GB/ % rozmiaru instancji]
Liczba konfliktów w dostępie do PW – bank conflicts
Wykonane instrukcje – executed instructions
Grid_size
Block_size
Liczba rejestrów na wątek

Rozmiar PW użytej przez blok wątków[B]
Occupancy theoretical/achieved [%]
Ograniczenie na liczbę bloków [blok] – zależne od rejestrów, dostępnej PW, liczby wiązek, liczby SM
- przyczyna krytycznej wartości ograniczenia.

Literatura do przygotowania projektu

Wykłady z przedmiotu z zakresu PKG/GPU
Dokumentacja SDK CUDA https://docs.nvidia.com/cuda/ a w szczególności:
CUDA_C_Programming_Guide B.14 Atomic functions, B.5 Memory Fence Functions B.6 Synchronization functions
CUDA_C_Best_Practices_Guide
NsightCompute
CC karty: https://developer.nvidia.com/cuda-gpus#compute
Przewodniki programowania dla poszczególnych rodzin kart graficznych:
https://docs.nvidia.com/cuda/#programming-guides
Przygotowano 10.05.2022