Programowanie w CUDA na NVIDIA GPU (PKG) rok akademicki 2019/2020,
studenci Informatyki dziennej 3 rok nauczanie zdalne

Projekt 2 z Laboratorium z przetwarzania równoległego

Termin oddania: 1 czerwca 2020

Grupy projektowe: projekt jest wykonywany i zaliczany w grupach maksymalnie 2 osobowych (jeśli skład grupy ulega zmianie w stosunku do grup z projektu wcześniejszego proszę zgłosić do 11.05.2020)

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 sposób przetwarzania wymaganych wersji programów,
- wykonać eksperyment dla zadanego zakresu instancji i parametrów uruchomienia przy użyciu programu oceny efektywności przetwarzania lub bez w zależności od możliwości,
- zmierzyć czas przetwarzania/ odczytać w programie oceny efektywności informacje o uruchomieniach,
- przygotować sprawozdanie wg wymagań.

Różnorodność sprawozdań i dostępność opisu zadania
Ze względu na jednakowe zadania dla wszystkich grup, podobieństwa w realizacji projektu przez poszczególne grupy są nieuniknione, 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 4. 05.2020. W przypadku potrzeby konsultacji proszę o kontakt przez oprogramowanie Emeeting w poniedziłki terminie od godz. 16 do 17 pod adresem https://emeeting.put.poznan.pl/eMeeting/raf-wqr-mr2 lub w innym terminie ustalonym podczas kontaktu przez email.

Temat projektu

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

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).

Program CPU – metoda 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 opracowania komputerze z procesorem wielordzeniowym. Wyniki tego przetwarzania są punktem odniesienia przy porównaniu z efektywnością przetwarzania na karcie.

Program GPU –
Modyfikacja udostępnionego kodu mnożenia macierzy (Nvidia) pod kontem zrównoleglenia operacji pobierania danych do pamięci współdzielonej i obliczeń wykonywanych przez wątki. Eksperymenty proszę wykonać dla bloków o rozmiarach 8x8, 16x16 i 32x32 wątki (jeśli w danej karcie graficznej jest to możliwe).
Praca realizowaną przez blok wątków w poszczególnych etapach polega na:

- sprowadzaniu danych do pamięci współdzielonej bloku wątków,
- obliczaniu w oparciu o dane w pamięci współdzielonej wyników częściowych,

- zrównolegleniu pracy wątków bloku: możliwość wykonania etapu sprowadzania danych równocześnie z obliczeniami dzięki zapisowi danych sprowadzanych do oddzielnych tablic (lub rejestrów) i podmianie obszarów danych przez zamianę wskaźników (lub przepisanie wartości z rejestrów do pamięci współdzielonej) sprowadzonych z wykorzystywanymi do obliczeń.
- należy zbadać wpływ kolejności wystąpienia w kodzie etapów (równocześnie realizowanych) obliczeń i pobierania danych.
- Analizie eksperymentalnej i logicznej proszę poddać następujące wersje kodu:
    - oryginalna bez zrównoleglenia pobierania i obliczeń,
    - równoległe obliczenia i pobieranie do rejestrów (obliczenia jako pierwsze w kodzie),
    - równoległe obliczenia i pobieranie do rejestrów (pobieranie jako pierwsze w kodzie),
    - równoległe obliczenia i pobieranie do pamięci współdzielonej (obliczenia jako pierwsze w kodzie),
    - równoległe obliczenia i pobieranie do pamięci współdzielonej (pobieranie jako pierwsze w kodzie).

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
- przyspieszenie w stosunku do przetwarzania równoległego (openMP) za pomocą CPU
- CGMA – obliczony podczas analizowania przebiegu przetwarzania
- 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 (lub NVIDIA Visual Profiler – starsza wersja oprogramowania CUDA).

Parametry uruchomień proszę przedstawiać w sprawozdaniu w sposób zwarty (łatwy do porównań wartości) w jednej tabeli wraz z parametrami uruchomienia N, BLOK (liczba wątków bloku) i czytelnym wariantem wersji kodu. Porównanie efektywności rozwiązań CPU i GPU może oprócz prędkości uwzględniać dodatkowo (jeśli są dostępne) koszt obliczeń mierzony za pomocą:

- parametru energetycznego **TDP** (ang. *Thermal Design Power*),
- technologii (Lithography) i
- powierzchnię krzemu (Si area in mm^2) jako parametry kosztu produkcji i użycia układu obliczeniowego.

Dokumentacja

Po wykonaniu zadania proszę dostarczyć przez email z TEMATEM: 2P_NR_INDEKSU1_NR_INDEKSU2 sprawozdane zawierające :

- dane autorów (imiona i nazwiska, numery indeksu) i data oddania
- nazwa przedmiotu i nazwa projektu,
- opis wykorzystywanej karty graficznej (typ, CC – możliwości obliczeniowe, liczba SM i rdzeni, innych jednostek obliczeniowych, ograniczenia posiadanego przez kartę CC)
- szczegółowy opis zakresu zrealizowanego zadania (przeprowadzone analizy realizowanego zadania)
- kluczowe fragmenty kodów kerneli z **wyjaśnieniami** dotyczącymi znaczenia instrukcji, określenie i uzasadnienie jakości występującego w kodzie dostępu do pamięci, opis sposobu określania konfiguracji uruchomienia.
- rysunki z opisem określające:
    - miejsce dostępu i kolejność dostępu do danych realizowane przez poszczególne wątki, bloki i
    - wyznaczane przez wątki i bloki wartości wyników,
- wzory zastosowane do obliczeń wszystkich prezentowanych miar efektywności przetwarzania wraz z wyjaśnieniem znaczenia tych miar,
- wymagane wyniki jakości przetwarzania dla poszczególnych zbadanych instancji proszę przedstawić w postaci tabelarycznej (najlepiej jedna tabela w orientacja portrait), tabele (i wykresy) należy ponumerować i podpisać w sposób nie budzący wątpliwości co do zawartości,
- wnioski z wykonanych eksperymentów z uzasadnieniem (przy pomocy takich pojęć jak CGMA, zajętość multiprocesora, łączenie dostępów, lokalność dostępu do danych, synchronizacja wątków, ilość pracy wątków) obserwowanych wartości (czytelne odwołanie do omawianej wielkości i jej wartości - jakiego uruchomienia - system, parametry, wersja kodu – dotyczą).

Do przesłanej wersji elektronicznej sprawozdania proszę dołączyć pliki źródłowe kernela.

Zaliczanie projektu będzie wymagało wiadomości z zakresu przetwarzania równoległego na PKG, a w szczególności poniżej wymienionych zagadnień. W przypadku braku pomysłu na wnioski z przeprowadzonego eksperymentu proszę opracować poniższe zagadnienia i jeśli nie były zawarte już wcześniej zawrzeć w opracowaniu:

1. Możliwość / brak możliwości łączenia dostępów do pamięci globalnej - uzasadnienie.
2. Wyznaczenie CGMA, znaczenie wielkości CGMA dla poszczególnych wersji kodu – na podstawie kodu.
3. Analiza wpływu na prędkość przetwarzania takich czynników jak: wielkość bloku wątków, wielkość instancji problemu, zajętość multiprocesora, żądania zasobowe związane z wariantem kodu: liczba rejestrów i wymaganie na pamięć współdzieloną
4. Ocena liczby dostępów do pamięci globalnej (odczyt i zapis) w funkcji rozmiaru problemu dla poszczególnych wersji kodu.
5. Ocena liczby operacji zmiennoprzecinkowych w funkcji rozmiaru problemu.
6. Ocena wielkości żądania na wielkość pamięci współdzielonej wątków w ramach poszczególnych wersji kodu.
7. Ocena potrzeby, zakresu, zadań i skutków synchronizacji przetwarzania w poszczególnych wersjach kodu.

INFORMACJE UZUPEŁNIAJĄCE – PRZYGOTOWANIE SYSTEMU

1. Ze względu na możliwą obecnie tylko zdalną realizację zadań laboratoryjnych proszę wykonać eksperyment 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
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" i dołączonym również do niniejszego opisu. 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 NVIDIA Visual Profiler wymaga:

- utworzenia kodu wynikowego badanej aplikacji w VisualStudio,
- uruchomienia NVIDIA Visual Profiler,
- utworzenia sesji oceny efektywności (profilowania) i wskazania pliku uruchamianego kodu i
- 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: Czasu przetwarzania (Timeline View), widoku analiz (Analysis View), widoku szczegółów (miar efektywności) Details View, Detail Graphs View, Properties View, Console View (standardowe wyjście przetwarzania kodu) , Settings View (określenie ścieżki kodu i parametrów linii uruchomienia).
- Różnice (w stosunku do NVIDIA Visual Profiler) w użyciu programu NsightCompute (nowsze wersje CUDA SDK) zostały opisane w dokumentacji do oprogramowania.

Literatura do projektu
Wykłady z przedmiotu z zakresy PKG/GPU
Dokumentacja SDK CUDA https://docs.nvidia.com/cuda/ a w szczególności:
CUDA_C_Programming_Guide
CUDA_C_Best_Practices_Guide
NsightCompute
CC karty: https://developer.nvidia.com/cuda-gpus#compute
Programming Guides dla poszczególnych rodzin kart graficznych:
https://docs.nvidia.com/cuda/#programming-guides