Grupa 1	Kordian Kurdziel, Mateusz Maciejak	14/11/2018

Zadanie 6 - Liczby pierwsze - CUDA

Celem zadania było przetestowanie czy liczby podane w pliku są liczbami pierwszymi. W celu poprawy wydajności obliczeń program należało zrównoleglić z użyciem technologii CUDA.

Program przyjmuje na wejściu jeden argument: ścieżkę do pliku w którym znajduje się lista liczb do przetestowania. Pierwszą rzeczą jaką wykonuje program jest sprawdzenie czy została podana odpowiednia ilość argumentów. Następnie zostaje wczytany plik a wpisane liczby umieszczone są w tablicy.

```
__global__ void calculate(ll *Arr, bool *results, int sizeOfArray, int
    amountOfBlocks){

    int x = (blockIdx.x * blockDim.x) + threadIdx.x;

    if (amountOfBlocks >= sizeOfArray){
        results[x] += isPrime(Arr[x]);
    } else{
        int sizeOfPart = sizeOfArray / amountOfBlocks;
        int restOfDivide = sizeOfArray%amountOfBlocks;

        int startPart = sizeOfPart * x;
        int endPart = sizeOfPart * (x + 1);

        if (endPart <= sizeOfArray)
        {
            int restStart = sizeOfPart * amountOfBlocks;

            for (int i = startPart; i < endPart; i++){
                results[i] += isPrime(Arr[i]);
            }

            if (x < restOfDivide){
                results[restStart + x] += isPrime(Arr[restStart + x]);
            }


        }
    }
}
```

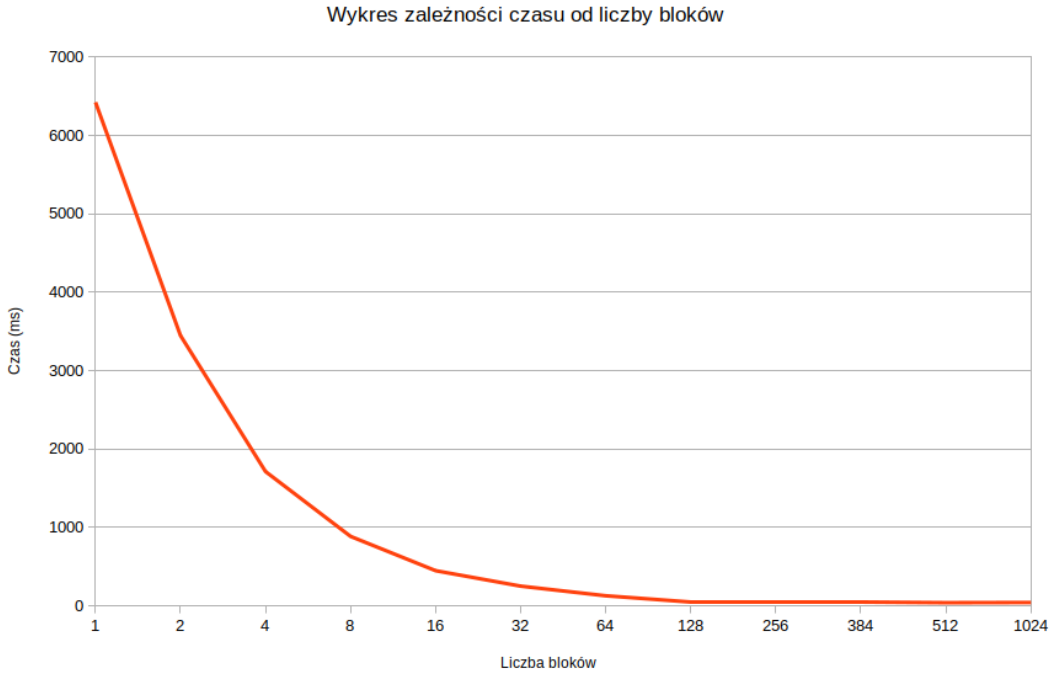
Powyższy funkcja jest wywoływana z hosta i odpowiada za odpowiedni podział przesłanej tablicy z liczbami do sprawdzenia. Dzięki zmiennym dostępnych w architekturze CUDA blockIdx.x, blockDim.x oraz threadIdx.x obliczany jest unikalny indeks bloku. Następnie jeżeli liczba bloków jest większa lub równa ilości liczb w tablicy w każdym bloku jest obliczana pojednyńcza liczba. W przeciwnym wypadku tablica jest rozdzielana na równe części między bloki i poszczególne oszacowanie jest przeprowadzane na tych partiach tablicy liczb.

```
__device__ bool isPrime(ll n)
{
    if(n<2)
        return false;
```

```
5
6       for(ll i=2;i*i<=n;i++)
7           if(n%i==0)
8               return false;
9
10      return true;
11  }
```

Powyższy listing przedstawia algorytm wyznaczania liczby pierwszej. Funkcja zostaje uruchomiona na urządzeniu z funkcji już działającej również na urządzeniu, dlatego przed zwracanym typem pojawiło się słowo kluczowe __device__. Został wybrany algorytm naiwny, ponieważ gwarantował poprawność wyników, a ilość i długość testowanych liczb nie była na tyle długa by powodowało to znaczące obniżenie wydajności obliczeń. Powyższy kod zostaje wywołany dla oszacowania każdej liczby znajdującej się w tablicy.

Poniższy wykres przedstawia wyniki uzyskane na serwerze CUDA. Czas obliczeń dla każdej ilości bloków przedstawionej na wykresie został przetestowany 5 razy a z wyników wyciągnięto średnią arytmetyczną.



Rysunek 1: Wykres zależności czasu wykonywania obliczeń od liczby bloków

Jak można zauważyć, czas obliczeń znacząco się zmniejsza wraz z rozpoczęciem zwiększania liczby bloków. Wraz z wzrostem liczby bloków różnice pomiędzy sąsiednimi czasami stają się coraz mniejsze. Siatka została ustawiona na wartość 1 gdyż taka konfiguracja dawała najlepsze rezultaty. Liczba bloków na jakich program uzyskuje najlepsze rezultaty zależy od ilości liczb do przetestowania. W naszym przypadku dla zbioru testowego składającego się z 500 liczb, liczba bloków powyżej tej wartości nie dawała przyśpieszenia, gdyż każda liczba zostawała przypisana do jednego bloku. Dlatego też końcowy ilość bloków jest ustalana na podstawie ilości liczb zawartych w testowym pliku i jest ustalana podczas uruchomienia programu. Jest to rozwiązanie optymalne dla naszego algorytmu. W kernelu odbywa się podział liczb między bloki oraz wyznaczanie czy przesłana liczba jest liczbą pierwszą.