Додаткове завдання
Обрахунок інтегралу заданої функції з використанням CUDA

Команда: Сень Іван, Пахолок Віктор, Семсічко Лідія

Опис імплементації:
Імплементація не сильно відрізняється від стандартного паралельного інтегралу. Головна різниця полягає в тому що CUDA ядра погано працюють з циклами та if-ками, тому їх стараємося оптимізувати. Кожне ядро рахує одну функцію функцію в точці і множить її на дельту. Це створює проблему коли точок стає більше ніж ядер, адже підвантаження з ram в пам'ять карти займає доволі багато часу.

Також наша відеокарта на якій тестувався код(Nvidia TITAN X) не підтримує атомік додавання для даблів, і тому ми змушенні використовувати своєрідний мютекс - __syncthreads()(cuda не має стандартного мютексу)

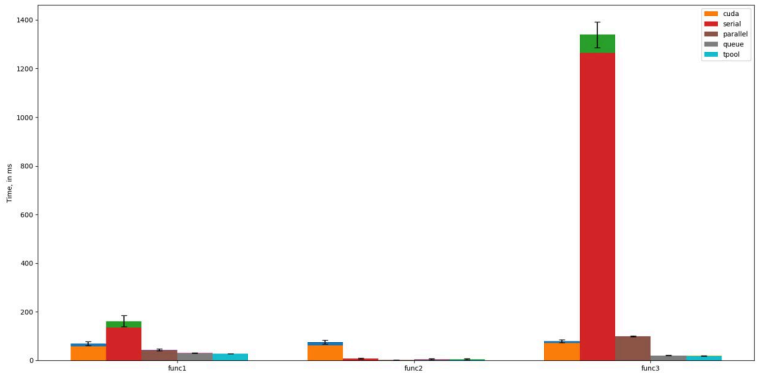
Також проблемою є те що багато стандартних cuda функцій працюють тільки для флоутів, а для даблів - ні. Саме тому в cuda похибка більша в нашому тестуванні.

Щоб дізнатися який шматок функції потоки рахують вони беруть свій індекс в блоці і індекс блоку потоків, і так знаходять свій блок функції:

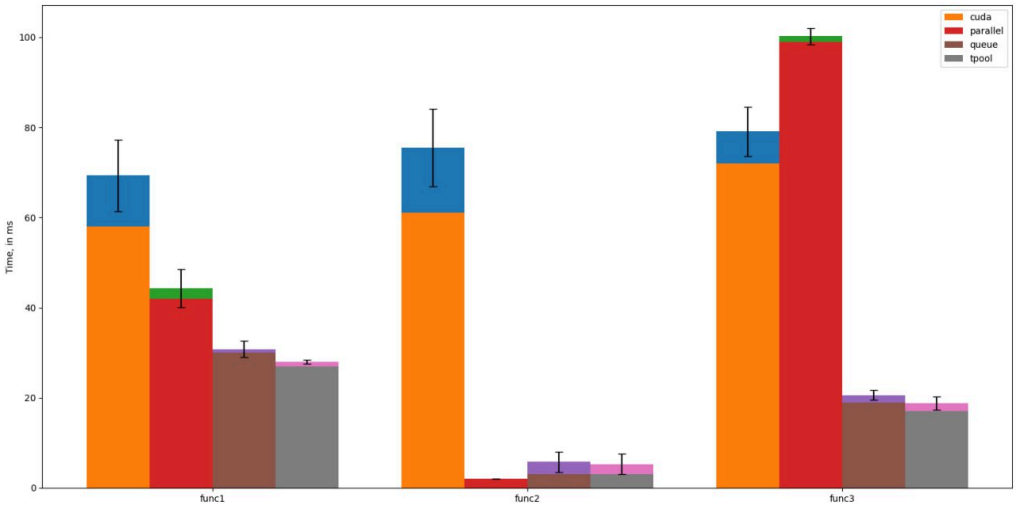
```
int idx = blockIdx.x * blockDim.x + threadIdx.x;
int idy = blockIdx.y * blockDim.y + threadIdx.y;
```

Графіки порівняння:
Спочатку  спробували побудувати графіки, щоб порівняти всі

наші лабораторні роботи на цю тему, проте послідовний інтеграл справляється набагато гірше на великих вимогах, що можна побачити на графіку нижче




Тому для показовості прибрали serial та порівнювали всі інші




**`func1`**:

- CUDA — найповільніше (≈70 ms), із великою похибкою. Похибка (error bar): досить висока — десь ~±7–8 мс, тобто близько 10%. Невигідно запускати `func1` на GPU через малий обсяг роботи.
- `parallel`, `queue`, `tpool` — вдвічі або навіть втричі швидші.
- `tpool` — найкращий результат (~28 ms), з найменшою похибкою.

**`func2`**:

- `cuda` знову найдовше (~76 ms), `parallel` — ≈3 ms (дуже добре). Похибка теж висока, на рівні ~±8–9 мс. Витрати на копіювання даних та ініціалізацію CUDA-ядра перевищують виграш від паралельності.
- `queue` і `tpool` — трохи повільніше (~10–13 ms), але стабільно.
- `parallel` тут найефективніше.

**`func3`**:

- `parallel` — *найгірший* (~100 ms).
- `cuda` трохи краще (~80 ms). Похибка вже менша. Все ще є значні витрати на перенесення даних та запуск ядер.
- `queue` і `tpool` — значно кращі (≈18–20 ms), з маленькою похибкою.

Можна сказати що при виконанні попередніх лаб, ми занадто добре все оптимізували).