Московский Государственный Университет им. М.В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Суперкомпьютеров и Квантовой Информатики

Практикум на ЭВМ
Отчёт № 2
Оптимизация свёртки изображения с использованием CUDA

Работу выполнил

Сайбель Т. А.

Москва 2021

Постановка задачи

Реализовать программу с использованием CUDA, осуществляющую свёртку изображения с 3 фильтрами, и протестировать на 2 типах изображений: 2000x2000 и 300x300.

Описание алгоритма

Рассматривалось 3 фильтра:

Edge detection = $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$

Gaussian blur = $\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$

Sharpen = $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Каждый элемент нового изображения вычислялся по следующей формуле:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} * \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} x_{(m-i)(n-j)}\, y_{(1+i)(1+j)}$$

Пример работы

Оригинал

Edge detection

Gaussian blur

Sharpen

Описание программы

Программы использует make и CImg(для чтения и записи изображения).
В качестве аргументов командной строки программа получает название фильтра (gaussian/edge/sharpen) и название файла с изображением (все изображения хранятся в image/, формат .jpg).
Программа выводит название фильтра и 2 времени выполнения: выполнение CUDA-ядер и выполнения CUDA-ядер + копирование данных, и сохраняет изображение в res/filename + filtername + “.jpg”.

Класс Solver: основной класс.
Класс Args: парсит командную строку.
Класс Reader: читает изображение
Класс Writer: сохраняет изображение
Класс Image: хранит изображение (Pixel*)

```
void Solver::solve(int filter, const std::string &inFilename, const
std::string &outFilename) - получает фильтр и название файла и
сохраняет результат solve<filterName>().

Image Solver::solve<filterName>(Image) - Выделяет необходимую память
на GPU и вызывает Kernel call.

__global__
void applyFilter(Pixel *image, Pixel* filtere, const double * kernel,
int kernelCenter, int width, int height) - осуществляет свёртку
одного пикселя входного изображения
```

Оптимизация

Были реализованы следующие улучшения:

1) **Развертка массива структур Pixel* -> в массив unsigned char***, хранящий изображение в следующем формате: rrr...ggg...bbb… Позволяет улучшить шаблон доступа к глобальной памяти. **Ускоряет программу.**
2) **Использование более быстрой разделяемой памяти для доступа к данным и 3-х нитей для обработки rgb компонент**.
3) **Выбор оптимального блока**. Размер оптимального блока = 16. Позволяет эффективнее работать с памятью. **Ускоряет программу.**
4) **Оптимизация применения фильтра**: развёртка циклов + использование отдельных функций для каждого фильтра + использование константных матриц. Позволяет снизить ветвление и обращение к глобальной памяти. **Ускоряет программу.**
5) **Оптимизация для маленьких изображений**: выделение памяти под обрабатываемые изображения один раз, обработка нескольких изображений за раз одним ядром.

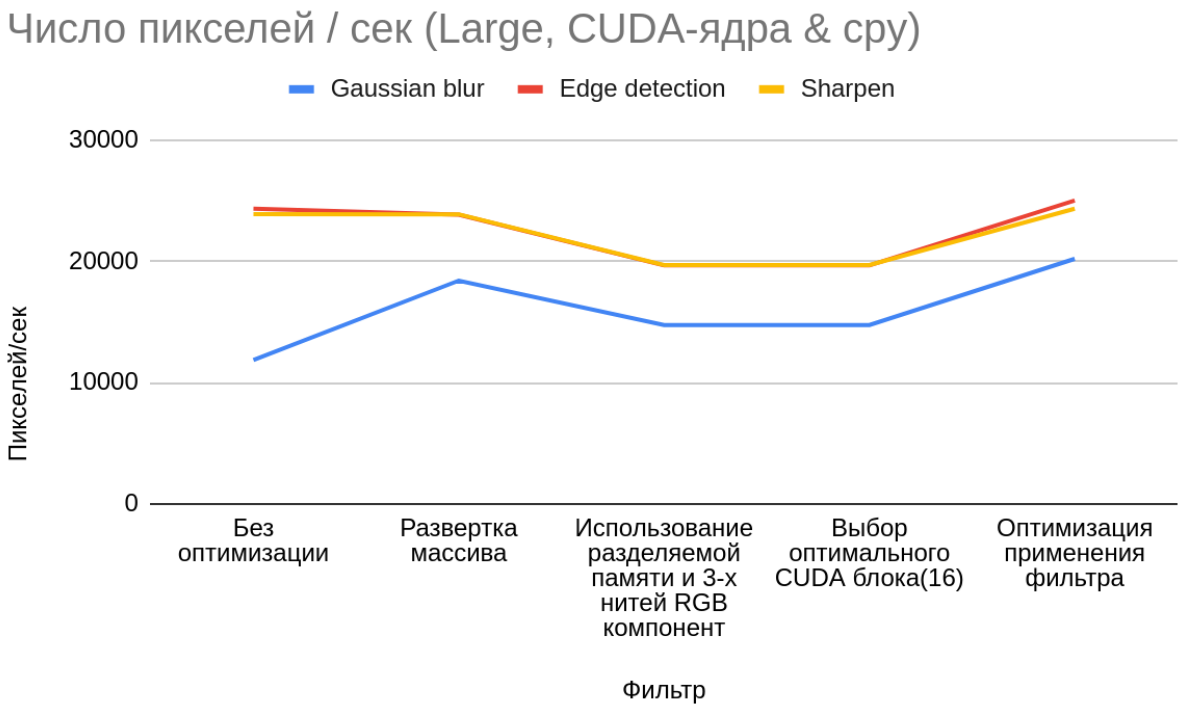
Класс Solver для каждой оптимизации хранится в пространстве имен Opt<номер оптимизации>.

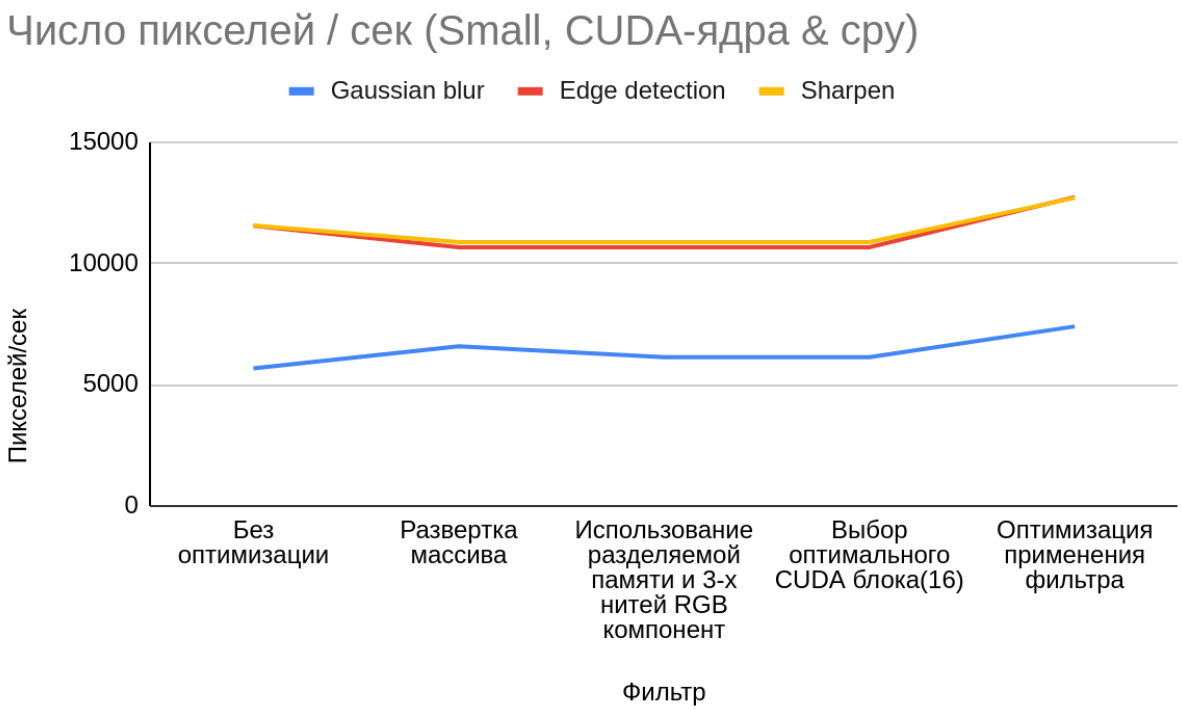
Результаты

Время работы (мс)

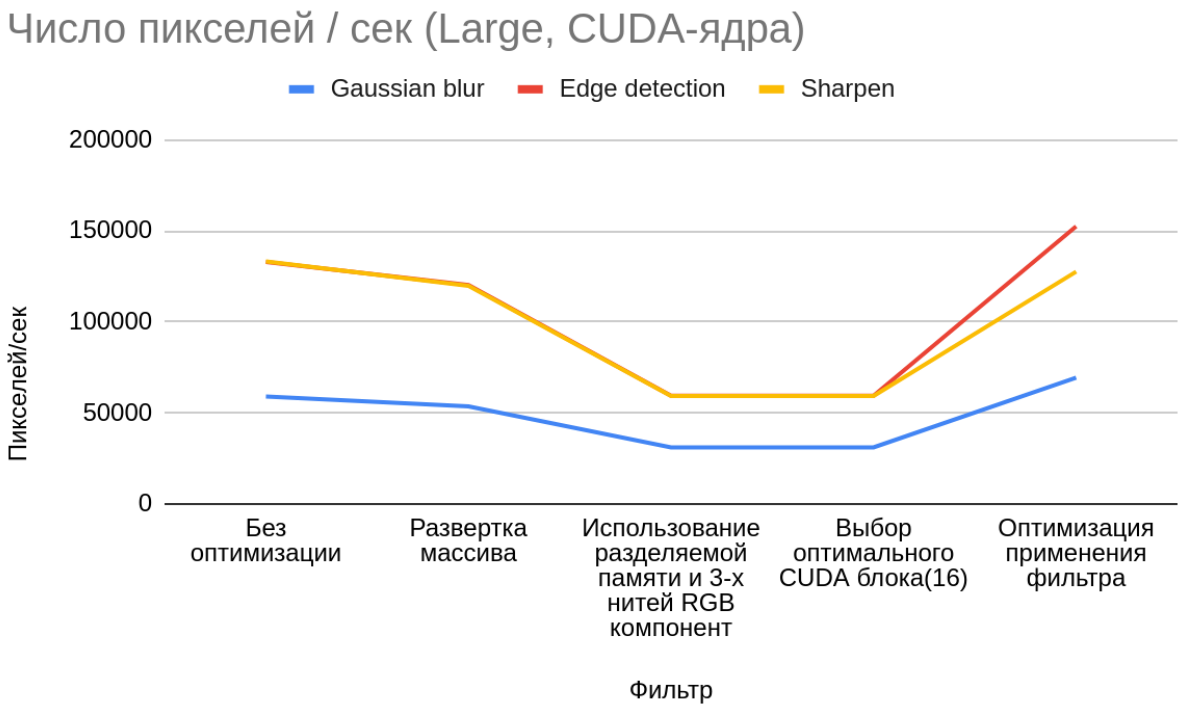
Без оптимизации	Без копирования			С копированием	
	Фильтр	***Big image***	***Small image***	***Big image***	***Small image***
	Gaussian blur	2.34013	0.071744	11.657	0.264608
	Edge detection	1.03939	0.032384	5.67971	0.129888
	Sharpen	1.03776	0.031488	5.78701	0.12976
Развертка массива	Без копирования			С копированием	
	Gaussian blur	2.57894	0.074368	7.51075	0.227936
	Edge detection	1.14918	0.032544	5.80029	0.127808
	Sharpen	1.1536	0.03168	5.79098	0.124704
Использование разделяемой памяти и 3-х нитей RGB компонент	Без копирования			С копированием	
	Gaussian blur	4.44336	0.092064	9.37821	0.244768
	Edge detection	2.32627	0.044544	7.0319	0.140736
	Sharpen	2.32934	0.04448	7.02202	0.137984
Выбор оптимального CUDA блока(16)	Без копирования			С копированием	
	Gaussian blur	4.44336	0.092064	9.37821	0.244768
	Edge detection	2.32627	0.044544	7.0319	0.140736
	Sharpen	2.32934	0.04448	7.02202	0.137984
Оптимизация применения фильтра	Без копирования			С копированием	
	Gaussian blur	1.99088	0.062752	6.84301	0.202752
	Edge detection	0.90672	0.0296	5.52736	0.117856
	Sharpen	1.08346	0.030848	5.68182	0.118144

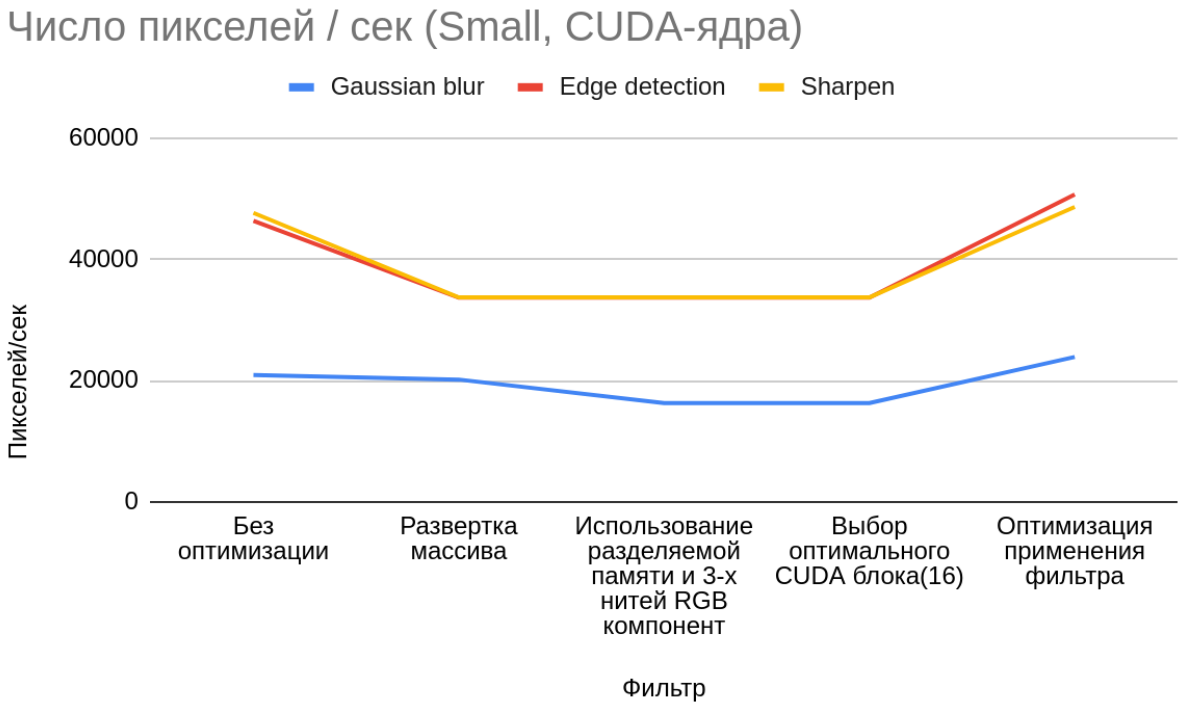
Количество обрабатываемых пикселей в секунду (CUDA-ядра + копирование)





Количество обрабатываемых пикселей в секунду (CUDA-ядра)





Профилировка

1) Задание 1, большое изображение:

```
==224977== NVPROF is profiling process 224977, command: ./ImageConvolution big/
==224977== Profiling application: ./ImageConvolution big/
==224977== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   44.17%  19.965ms         3  6.6551ms  6.5858ms  6.7055ms  [CUDA memcpy DtoH]
                   37.05%  16.747ms         3  5.5822ms  3.6827ms  9.3801ms  applyFilter(Pixel*, Pixel*, double const *, int, int, int)
                   18.77%  8.4848ms         6  1.4141ms  1.1200us  2.8735ms  [CUDA memcpy HtoD]
      API calls:   77.70%  175.80ms         9  19.533ms  68.129us  174.83ms  cudaMalloc
                   13.33%  30.164ms         9  3.3516ms  55.354us  7.1586ms  cudaMemcpy
                   7.47%  16.907ms        12  1.4089ms     902ns  9.3785ms  cudaDeviceSynchronize
                   1.15%  2.6059ms         9  289.54us  99.097us  573.13us  cudaFree
                   0.15%  343.90us         1  343.90us  343.90us  343.90us  cuDeviceTotalMem
                   0.08%  182.39us       101  1.8050us     190ns  78.509us  cuDeviceGetAttribute
                   0.04%  91.344us         3  30.448us  29.366us  31.911us  cudaLaunchKernel
                   0.03%  58.492us        12  4.8740us  2.0140us  12.634us  cudaEventRecord
                   0.02%  35.427us         1  35.427us  35.427us  35.427us  cuDeviceGetName
                   0.01%  21.832us         6  3.6380us  3.1160us  4.8190us  cudaEventSynchronize
                   0.01%  19.478us        12  1.6230us     471ns  7.6250us  cudaEventCreate
                   0.01%  13.676us         6  2.2790us     832ns  3.8970us  cudaEventElapsedTime
                   0.00%  7.4940us         1  7.4940us  7.4940us  7.4940us  cuDeviceGetPCIBusId
                   0.00%  2.0030us         3     667ns     330ns  1.1720us  cuDeviceGetCount
                   0.00%  881ns            3     293ns     280ns     310ns  cudaGetLastError
                   0.00%  792ns            2     396ns     210ns     582ns  cuDeviceGet
                   0.00%  321ns            1     321ns     321ns     321ns  cuDeviceGetUuid
```

2) Задание 1, маленькие изображения:

```
==225253== NVPROF is profiling process 225253, command: ./ImageConvolution small/
==225253== Profiling application: ./ImageConvolution small/
==225253== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   60.71%  1.0339ms        15  68.926us  45.953us  115.04us  applyFilter(Pixel*, Pixel*, double const *, int, int, int)
                   20.66%  351.78us        30  11.725us     928ns  23.232us  [CUDA memcpy HtoD]
                   18.63%  317.22us        15  21.147us  21.056us  21.536us  [CUDA memcpy DtoH]
      API calls:   97.08%  184.41ms        45  4.0981ms  2.3150us  182.97ms  cudaMalloc
                   1.06%  2.0087ms        45  44.638us  22.162us  130.94us  cudaMemcpy
                   0.68%  1.2935ms        60  21.558us     781ns  116.21us  cudaDeviceSynchronize
                   0.53%  1.0049ms        45  22.330us  2.2650us  100.32us  cudaFree
                   0.24%  456.74us         1  456.74us  456.74us  456.74us  cuDeviceTotalMem
```

```
                0.11%  211.11us       101  2.0900us         230ns  90.221us
cuDeviceGetAttribute
                0.08%  161.42us       60  2.6900us  2.0540us  6.8330us  cudaEventRecord
                0.08%  154.71us       15  10.313us  8.0750us  22.943us  cudaLaunchKernel
                0.06%  120.15us       30  4.0050us  3.8170us  4.4180us  cudaEventSynchronize
                0.02%  46.210us       60       770ns  470ns  2.9450us  cudaEventCreate
                0.02%  37.441us       1  37.441us  37.441us  37.441us  cuDeviceGetName
                0.02%  35.791us       30  1.1930us  762ns  2.3340us  cudaEventElapsedTime
                0.00%  8.4260us       1  8.4260us  8.4260us  8.4260us  cuDeviceGetPCIBusId
                0.00%  5.6000us       2  2.8000us    310ns  5.2900us  cuDeviceGet
                0.00%  3.5260us       15       235ns  170ns  791ns  cudaGetLastError
                0.00%  2.7140us       3       904ns  350ns  1.5830us  cuDeviceGetCount
                0.00%  380ns          1       380ns  380ns  380ns  cuDeviceGetUuid
```

3) Оптимизация, большое изображение:

```
==225538== NVPROF is profiling process 225538, command: ./ImageConvolution big/
==225538== Profiling application: ./ImageConvolution big/
==225538== Profiling result:
            Type  Time(%)          Time    Calls    Avg     Min      Max  Name
 GPU activities:   44.34%  20.119ms         3  6.7064ms  6.6186ms  6.8561ms  [CUDA
memcpy DtoH]
                36.91%  16.750ms     3  5.5832ms  3.6838ms  9.3814ms  applyFilter(Pixel*,
Pixel*, double const *, int, int, int)
                18.75%  8.5071ms     6  1.4178ms  1.0880us  2.9332ms  [CUDA memcpy HtoD]
        API calls:   78.58%  186.20ms         9  20.689ms  68.550us  185.17ms  cudaMalloc
                12.82%  30.374ms     9  3.3749ms  55.345us  7.3527ms  cudaMemcpy
                7.13%  16.905ms      12  1.4088ms  902ns  9.3801ms  cudaDeviceSynchronize
                1.09%  2.5895ms      9  287.72us  94.078us  570.96us  cudaFree
                0.18%  435.08us      1  435.08us  435.08us  435.08us  cuDeviceTotalMem
                0.09%  212.63us       101  2.1050us         230ns  90.872us
cuDeviceGetAttribute
                0.04%  89.108us      3  29.702us  29.165us  30.006us  cudaLaunchKernel
                0.03%  63.489us      12  5.2900us  2.1940us  13.295us  cudaEventRecord
                0.02%  35.868us      1  35.868us  35.868us  35.868us  cuDeviceGetName
                0.01%  22.544us      6  3.7570us  3.3570us  4.3580us  cudaEventSynchronize
                0.01%  16.270us      6  2.7110us  1.3420us  4.6790us  cudaEventElapsedTime
                0.01%  14.898us      12  1.2410us  461ns  3.8980us  cudaEventCreate
                0.00%  5.0890us      1  5.0890us  5.0890us  5.0890us  cuDeviceGetPCIBusId
                0.00%  2.5140us      3       838ns  381ns  1.4720us  cuDeviceGetCount
                0.00%  1.0720us      2       536ns  291ns  781ns  cuDeviceGet
                0.00%  831ns         3       277ns  270ns  291ns  cudaGetLastError
                0.00%  411ns         1       411ns  411ns  411ns  cuDeviceGetUuid
```

4) Оптимизация, маленькие изображения:

```
==225625== NVPROF is profiling process 225625, command: ./ImageConvolution small/
==225625== Profiling application: ./ImageConvolution small/
==225625== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   24.74%  319.04us         3  106.35us  106.08us  106.75us  [CUDA memcpy HtoD]
                   23.74%  306.18us         3  102.06us  101.98us  102.18us  [CUDA memcpy DtoH]
                   23.15%  298.53us         1  298.53us  298.53us  298.53us  sharpenFilter(unsigned char*, unsigned char*, int, int)
                   14.70%  189.54us         1  189.54us  189.54us  189.54us  edgeFilter(unsigned char*, unsigned char*, int, int)
                   13.67%  176.32us         1  176.32us  176.32us  176.32us  gaussianFilter(unsigned char*, unsigned char*, int, int)
      API calls:   96.73%  128.67ms         6  21.445ms  67.658us  128.24ms  cudaMalloc
                    1.51%  2.0123ms         6  335.38us  109.25us  525.06us  cudaMemcpy
                    0.72%  960.48us        12  80.039us     851ns  366.64us  cudaDeviceSynchronize
                    0.40%  526.37us         6  87.728us  78.118us  99.338us  cudaFree
                    0.33%  433.75us         1  433.75us  433.75us  433.75us  cuDeviceTotalMem
                    0.16%  212.30us       101  2.1010us     231ns  89.961us  cuDeviceGetAttribute
                    0.05%  63.790us         3  21.263us  19.527us  24.526us  cudaLaunchKernel
                    0.03%  41.360us        12  3.4460us  2.0940us  8.6560us  cudaEventRecord
                    0.03%  36.369us         1  36.369us  36.369us  36.369us  cuDeviceGetName
                    0.02%  24.496us         6  4.0820us  3.7970us  4.5180us  cudaEventSynchronize
                    0.01%  13.995us        12  1.1660us     480ns  3.5360us  cudaEventCreate
                    0.01%  9.6480us         6  1.6080us  1.1420us  2.0640us  cudaEventElapsedTime
                    0.00%  5.0600us         1  5.0600us  5.0600us  5.0600us  cuDeviceGetPCIBusId
                    0.00%  2.8460us         3     948ns     642ns  1.4930us  cuDeviceGetCount
                    0.00%  1.2630us         2     631ns     321ns     942ns  cuDeviceGet
                    0.00%  1.0910us         3     363ns     270ns     531ns  cudaGetLastError
                    0.00%  421ns            1     421ns     421ns     421ns  cuDeviceGetUuid
```

Основные выводы

Применение перечисленных оптимизаций позволяет ускорить выполнения программы: ~1,7 раз.