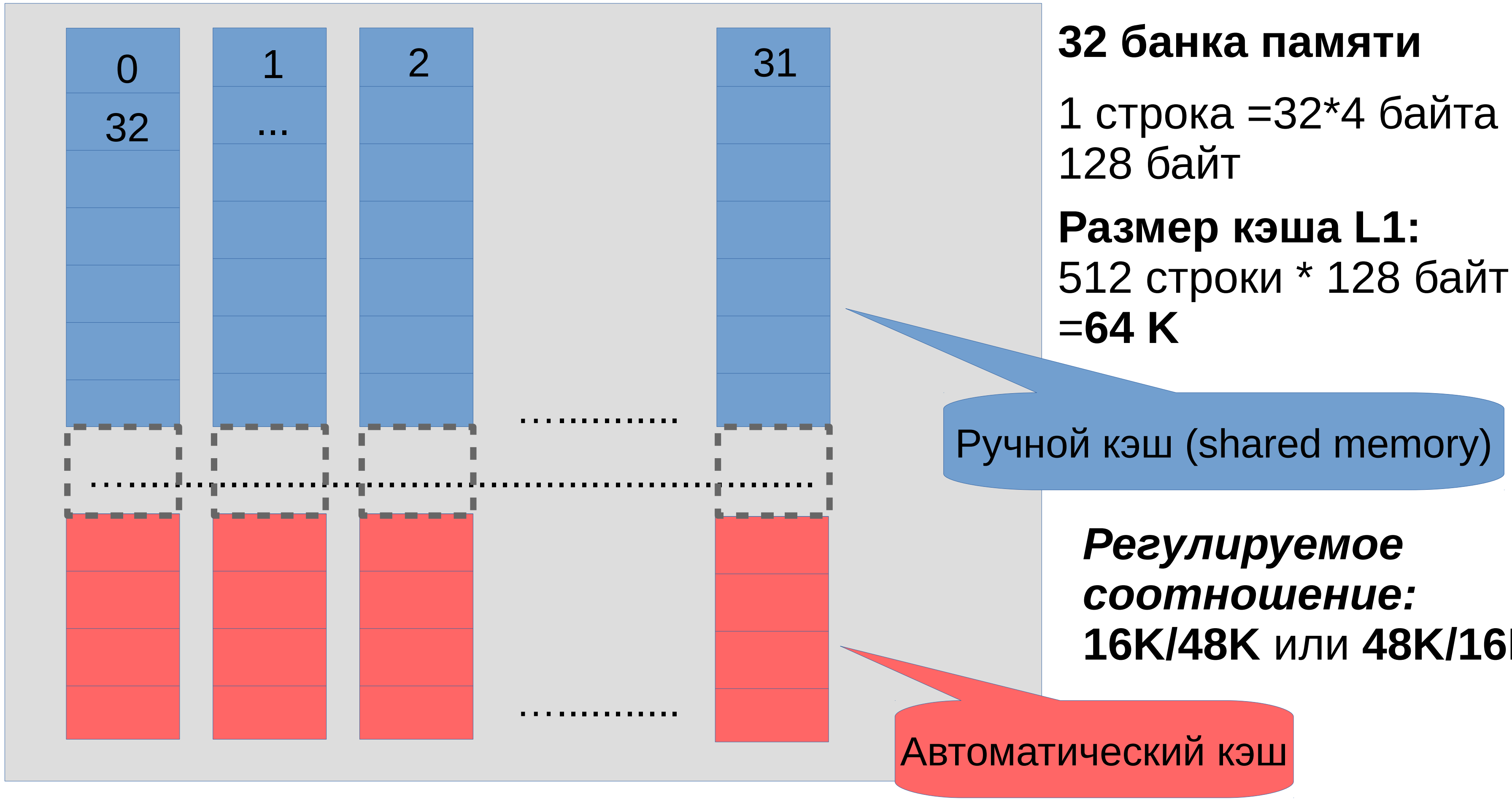
Лекция 3

СОДЕРЖАНИЕ

- ***разделяемая память (shared memory)***

L1 кэш память



32 банка памяти

1 строка =32*4 байта = 128 байт

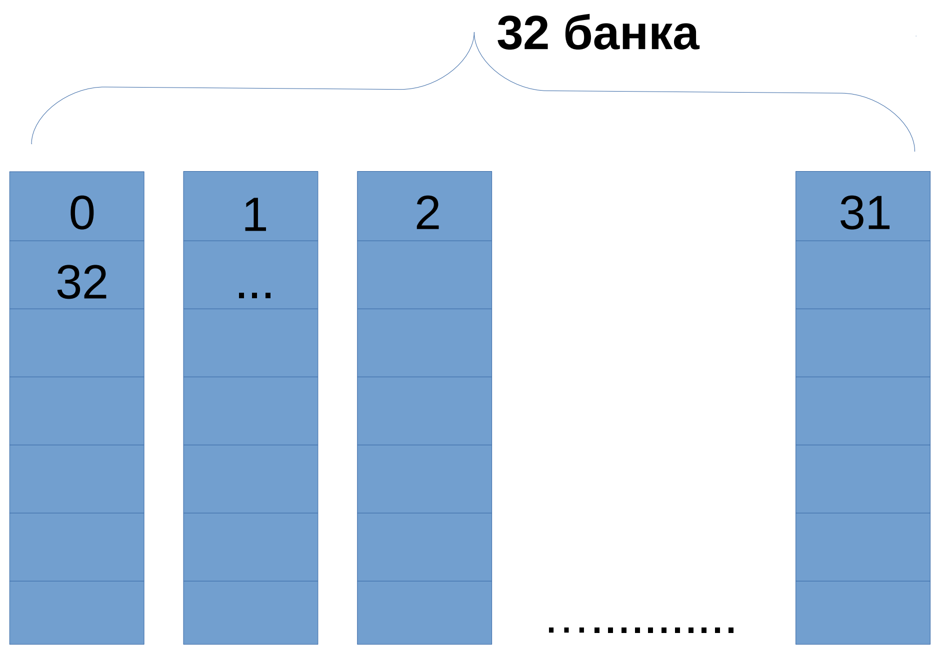
Размер кэша L1:
512 строки * 128 байт =**64 K**

Регулируемое соотношение:
16K/48K или **48K/16K**

Разделяемая память (shared memory)

Разделяемая память CUDA – память с низкой латентностью и высокой пропускной способностью.

Высокая пропускная способность обеспечивается параллельным выполнением запросов, благодаря разделению памяти на отдельные модули, банки памяти.



Если более одной нити варпа обращаются к одному и тому же банку, то происходит конфликт, который разрешается сериализацией выполнения запроса.

Выделение разделяемой памяти

Разделяемая память выделяется (статически или динамически) только на устройстве. Область видимости – нити одного блока. Время жизни – время выполнения ядра.

Статическое выделение:

```
#define N 3
#define M 512
__global__ void gTest(){
    __shared__ float s[N][M];
…......................................
}
```

Динамическое выделение:

```
extern __shared__ float s[];
__global__ void gTest2(){
        float* a=(float*)s;
        float* b=(float*)&s[512];
        float* c=(float*)&s[1024];
…......................................
}
```

```
gTest2<<<100,32,N*M>>>();
```

3-й параметр – размер разделяемой памяти.

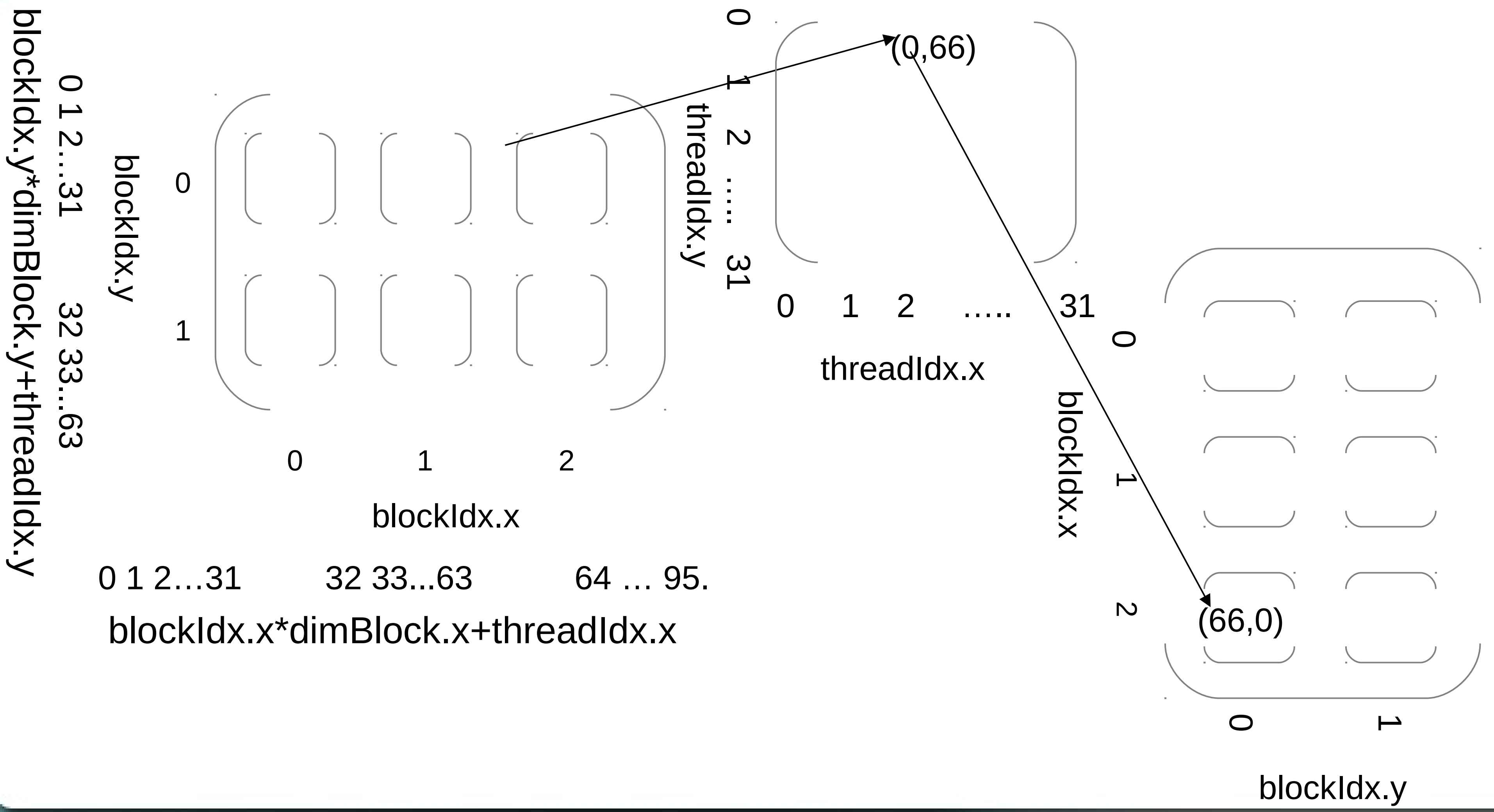
Задание (лабораторная 4)

Оптимизировать операцию транспонирования матриц используя *coalescing* и *shared memory*.

Инициализация матрицы

```
__global__ void gInitializeStorage(float* storage_d){
        int i=threadIdx.x+blockIdx.x*blockDim.x;
        int j=threadIdx.y+blockIdx.y*blockDim.y;
        int N=blockDim.x*gridDim.x;

            storage_d[i+j*N]=i+j*N;
}
```

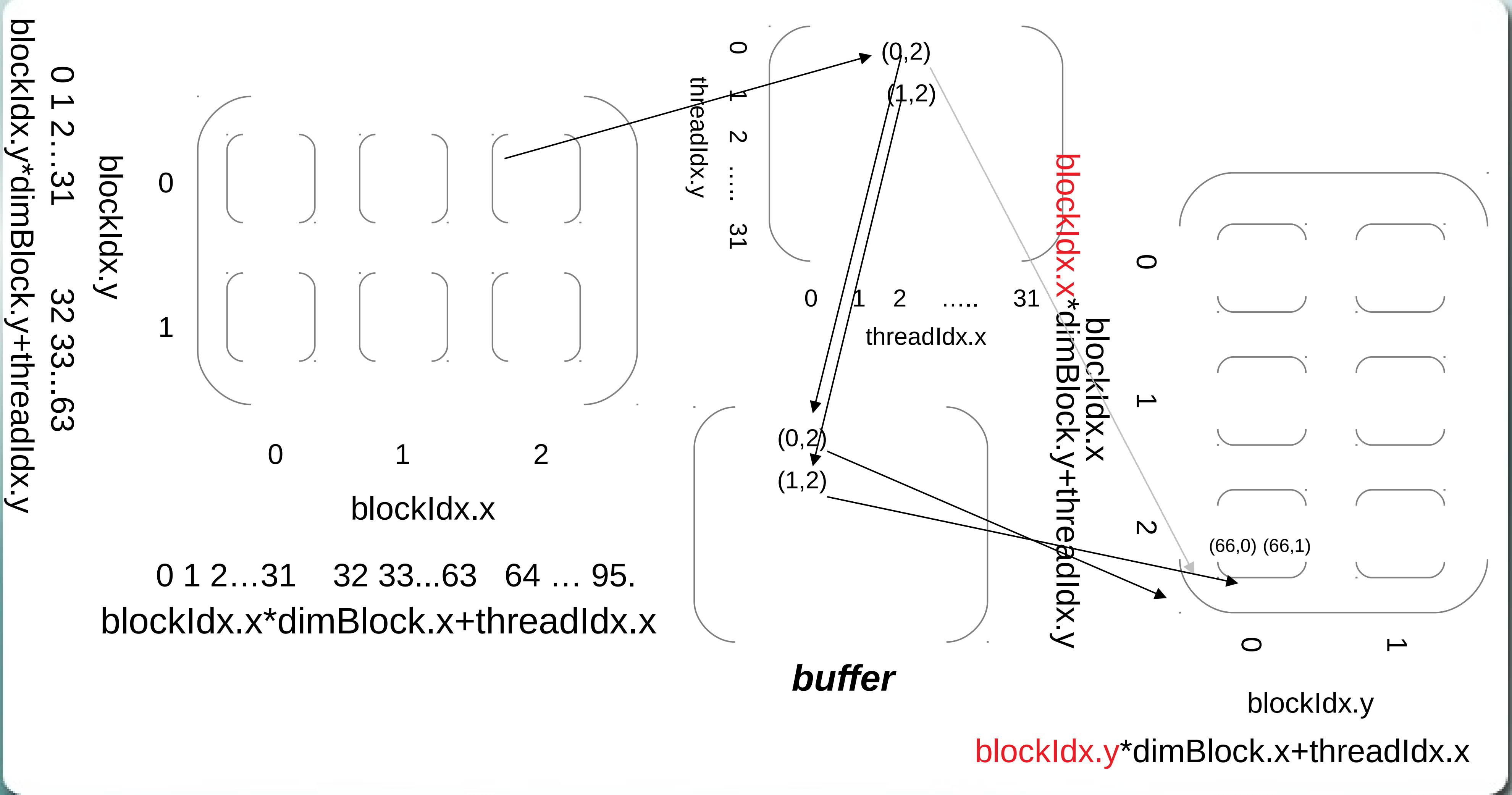
Простое транспонирование (схема)



Простое транспонирование

```
__global__ void gTranspose0(float* storage_d, float* storage_d_t){
        int i=threadIdx.x+blockIdx.x*blockDim.x;
        int j=threadIdx.y+blockIdx.y*blockDim.y;
        int N=blockDim.x*gridDim.x;

        storage_d_t[j+i*N]=storage_d[i+j*N];
}
```

Использование shared memory с конфликтом банков (схема)



Использование shared memory с конфликтом банков

```
#define SH_DIM 32

__global__ void gTranspose1(float* storage_d,float* storage_d_t){
        __shared__  float buffer[SH_DIM][SH_DIM];

        int i=threadIdx.x+blockIdx.x*blockDim.x;
        int j=threadIdx.y+blockIdx.y*blockDim.y;
        int N=blockDim.x*gridDim.x;

        buffer[threadIdx.y][threadIdx.x]=storage_d[i+j*N];
        __syncthreads();

        i=threadIdx.x+blockIdx.y*blockDim.x;
        j=threadIdx.y+blockIdx.x*blockDim.y;

        storage_d_t[i+j*N]=buffer[threadIdx.x][threadIdx.y];
}
```

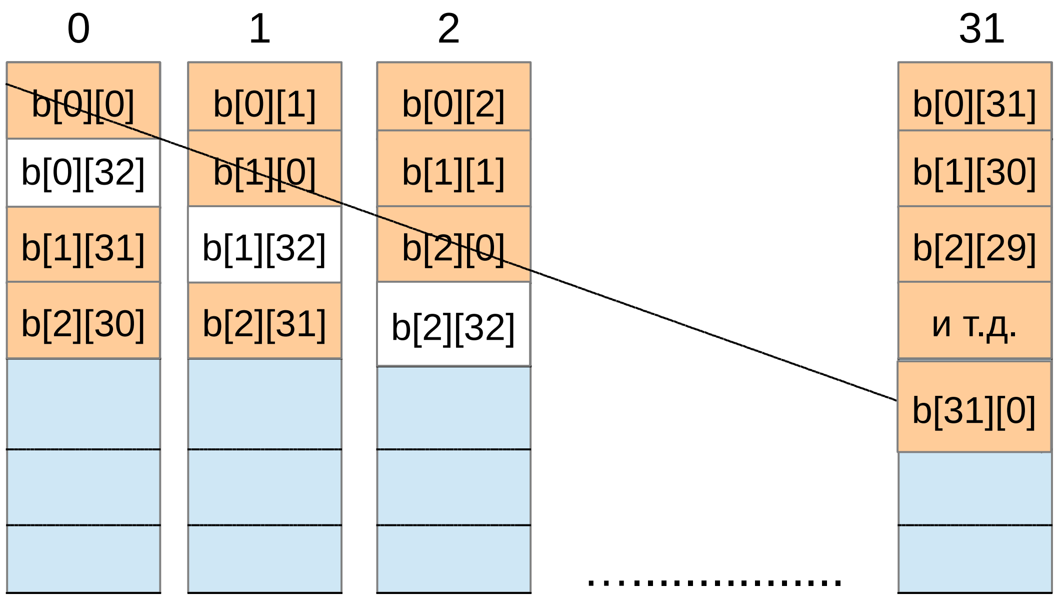
Использование shared memory с разрешением конфликтов банков

```
__global__ void gTranspose2(float* storage_d,float* storage_d_t){
        __shared__  float buffer[SH_DIM][SH_DIM+1];

        int i=threadIdx.x+blockIdx.x*blockDim.x;
        int j=threadIdx.y+blockIdx.y*blockDim.y;
         int N=blockDim.x*gridDim.x;

        buffer[threadIdx.y][threadIdx.x]=storage_d[i+j*N];
        __syncthreads();

        i=threadIdx.x+blockIdx.y*blockDim.x;
        j=threadIdx.y+blockIdx.x*blockDim.y;

        storage_d_t[i+j*N]=buffer[threadIdx.x][threadIdx.y];
}
```

Как избежать конфликта банков разделяемой памяти

__shared__ float buffer[SH_DIM][SH_DIM+1];

Размещение массива buffer в разделяемой памяти (shared memory):



Конфигурация нитей и запуск ядра

```
……………………………………………………………………………..........
int main(int argc, char* argv[]){
        if(argc<2){
                fprintf(stderr, "USAGE: matrix  <dimension of matrix>\n");
                return -1;
        }
        int N=atoi(argv[1]);

       const int max_size=1024;
       int dim_of_threads=SH_DIM;
       int size=N/dim_of_threads+(N%dim_of_threads>0);
        int dim_of_blocks=(size>max_size)?max_size:size;
        ……………………………………………………………….........
        float *storage_d, *storage_d_t, *storage_h;
```

Конфигурация нитей и запуск ядра

```
cudaMalloc((void**)&storage_d, N*N*sizeof(float));
cudaMalloc((void**)&storage_d_t, N*N*sizeof(float));
storage_h=(float*)calloc(N*N, sizeof(float));

gInitializeStorage<<<dim3(dim_of_blocks,dim_of_blocks),
                dim3(dim_of_threads,dim_of_threads)>>>(storage_d,N);
cudaDeviceSynchronize();

cudaMemcpy(storage_h,  storage_d, N*N*sizeof(float),
                                          cudaMemcpyDeviceToHost);

Output(storage_h, N);
```

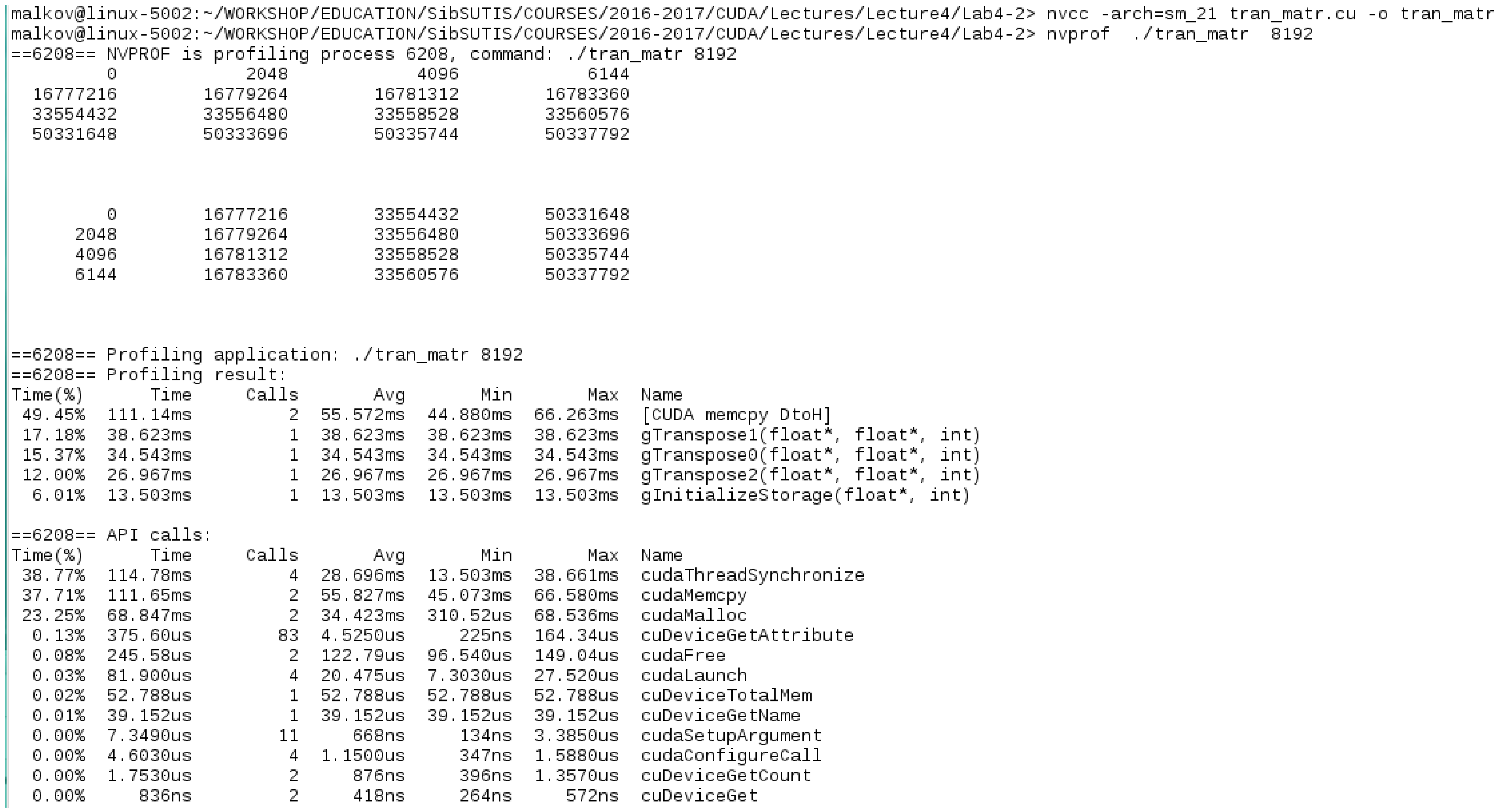
Конфигурация нитей и запуск ядра

```
gTranspose0<<<dim3(dim_of_blocks, dim_of_blocks),
                        dim3(dim_of_threads,dim_of_threads)>>>
                                                    (storage_d,storage_d_t,N);
cudaDeviceSynchronize();

gTranspose1<<<dim3(dim_of_blocks, dim_of_blocks),
                        dim3(dim_of_threads,dim_of_threads)>>>
                                                    (storage_d,storage_d_t,N);
CudaDeviceSynchronize();

gTranspose2<<<dim3(dim_of_blocks, dim_of_blocks),
                        dim3(dim_of_threads,dim_of_threads)>>>
                                                    (storage_d,storage_d_t,N);
cudaDeviceSynchronize();
```

Конфигурация нитей и запуск ядра

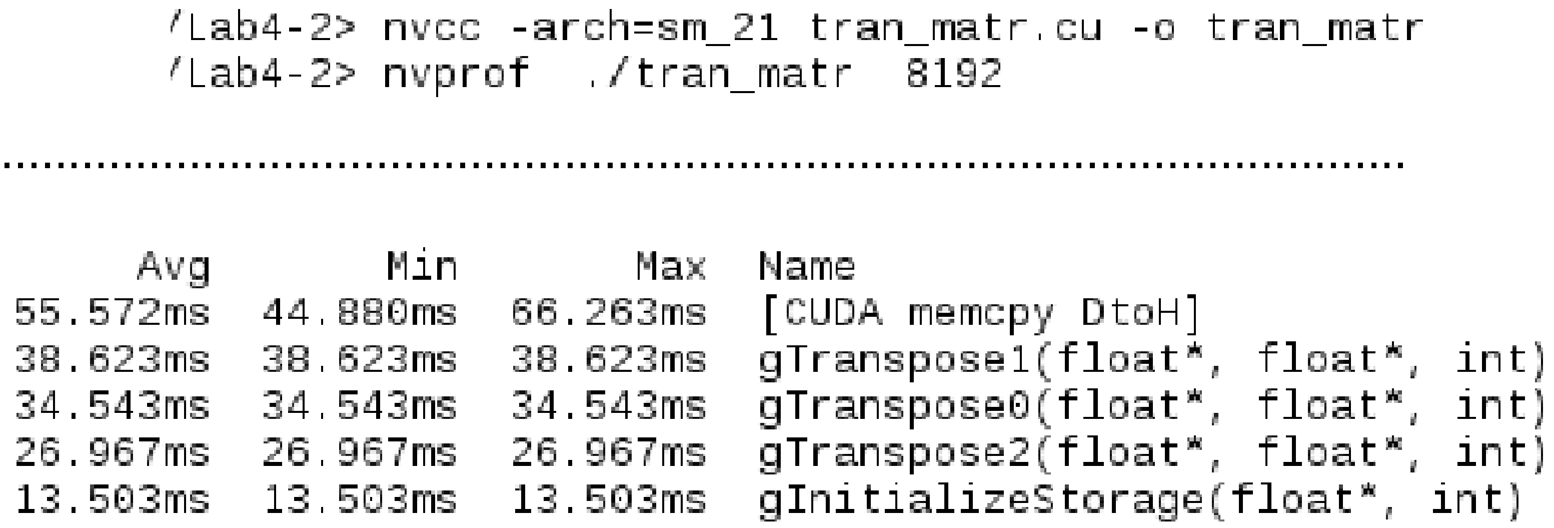
```
    cudaMemcpy(storage_h,  storage_d_t, N*N*sizeof(float),
                                              cudaMemcpyDeviceToHost);
    Output(storage_h, N);

    cudaFree(storage_d);
    cudaFree(storage_d_t);
    free(storage_h);

    return 0;
}
```

Время выполнения ядра с разными архитектурами CUDA (GeForce 560Ti, *Fermi*, CUDA 7.5, *compute capabilities 2.1*)



```
malkov@linux-5002:~/WORKSHOP/EDUCATION/SibSUTIS/COURSES/2016-2017/CUDA/Lectures/Lecture4/Lab4-2> nvcc -arch=sm_21 tran_matr.cu -o tran_matr
malkov@linux-5002:~/WORKSHOP/EDUCATION/SibSUTIS/COURSES/2016-2017/CUDA/Lectures/Lecture4/Lab4-2> nvprof  ./tran_matr  8192
==6208== NVPROF is profiling process 6208, command: ./tran_matr 8192
         0           2048           4096           6144
  16777216       16779264       16781312       16783360
  33554432       33556480       33558528       33560576
  50331648       50333696       50335744       50337792




         0       16777216       33554432       50331648
      2048       16779264       33556480       50333696
      4096       16781312       33558528       50335744
      6144       16783360       33560576       50337792




==6208== Profiling application: ./tran_matr 8192
==6208== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 49.45%  111.14ms         2  55.572ms  44.880ms  66.263ms  [CUDA memcpy DtoH]
 17.18%  38.623ms         1  38.623ms  38.623ms  38.623ms  gTranspose1(float*, float*, int)
 15.37%  34.543ms         1  34.543ms  34.543ms  34.543ms  gTranspose0(float*, float*, int)
 12.00%  26.967ms         1  26.967ms  26.967ms  26.967ms  gTranspose2(float*, float*, int)
  6.01%  13.503ms         1  13.503ms  13.503ms  13.503ms  gInitializeStorage(float*, int)

==6208== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 38.77%  114.78ms         4  28.696ms  13.503ms  38.661ms  cudaThreadSynchronize
 37.71%  111.65ms         2  55.827ms  45.073ms  66.580ms  cudaMemcpy
 23.25%  68.847ms         2  34.423ms  310.52us  68.536ms  cudaMalloc
  0.13%  375.60us        83  4.5250us     225ns  164.34us  cuDeviceGetAttribute
  0.08%  245.58us         2  122.79us  96.540us  149.04us  cudaFree
  0.03%  81.900us         4  20.475us  7.3030us  27.520us  cudaLaunch
  0.02%  52.788us         1  52.788us  52.788us  52.788us  cuDeviceTotalMem
  0.01%  39.152us         1  39.152us  39.152us  39.152us  cuDeviceGetName
  0.00%  7.3490us        11     668ns     134ns  3.3850us  cudaSetupArgument
  0.00%  4.6030us         4  1.1500us     347ns  1.5880us  cudaConfigureCall
  0.00%  1.7530us         2     876ns     396ns  1.3570us  cuDeviceGetCount
  0.00%     836ns         2     418ns     264ns     572ns  cuDeviceGet
```

Время выполнения ядра с разными архитектурами CUDA (GeForce 560Ti, *Fermi*, CUDA 7.5, *compute capabilities 2.1*)

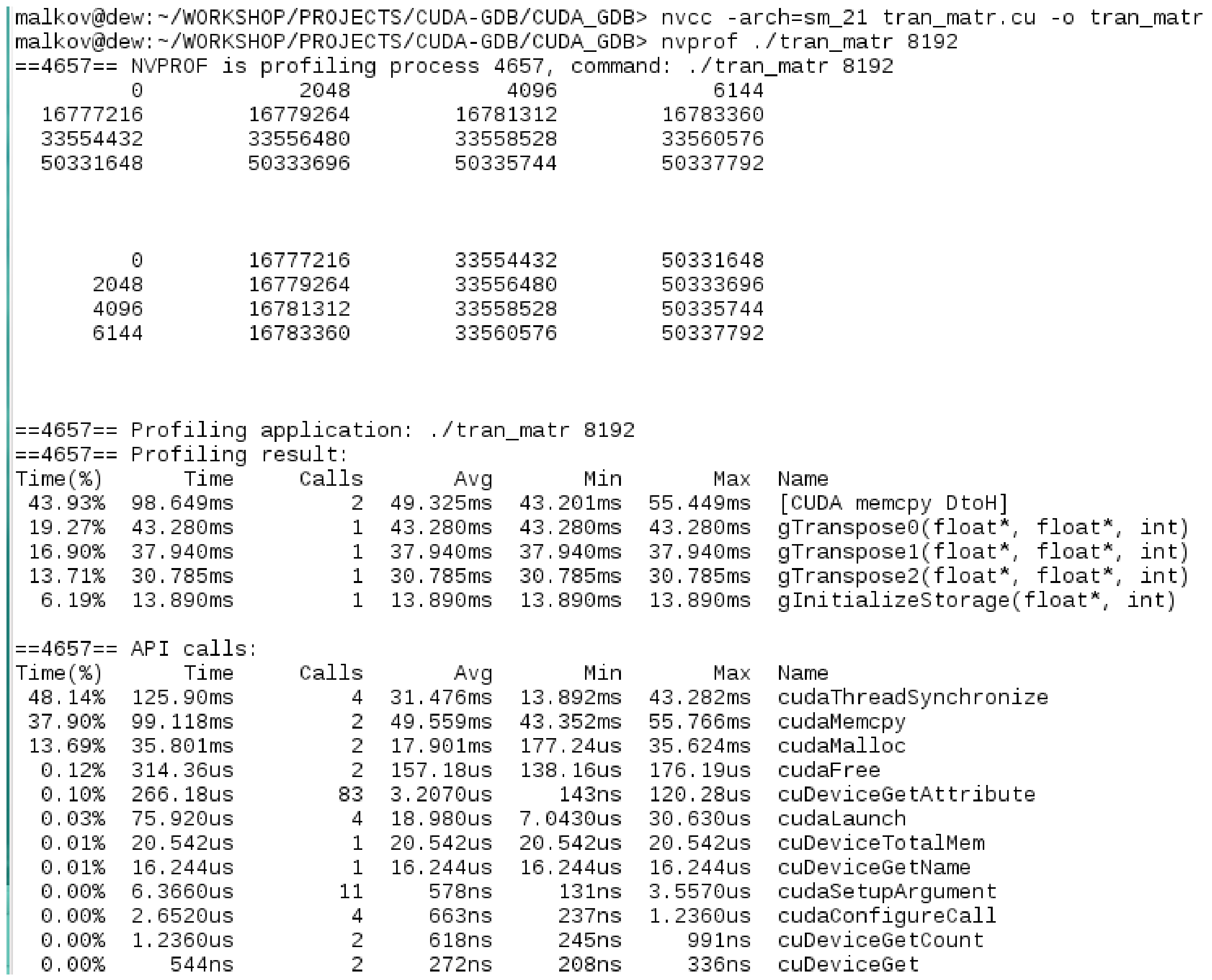


```
/Lab4-2> nvcc -arch=sm_21 tran_matr.cu -o tran_matr
/Lab4-2> nvprof  ./tran_matr  8192
```

………………………………………………………………………………………………

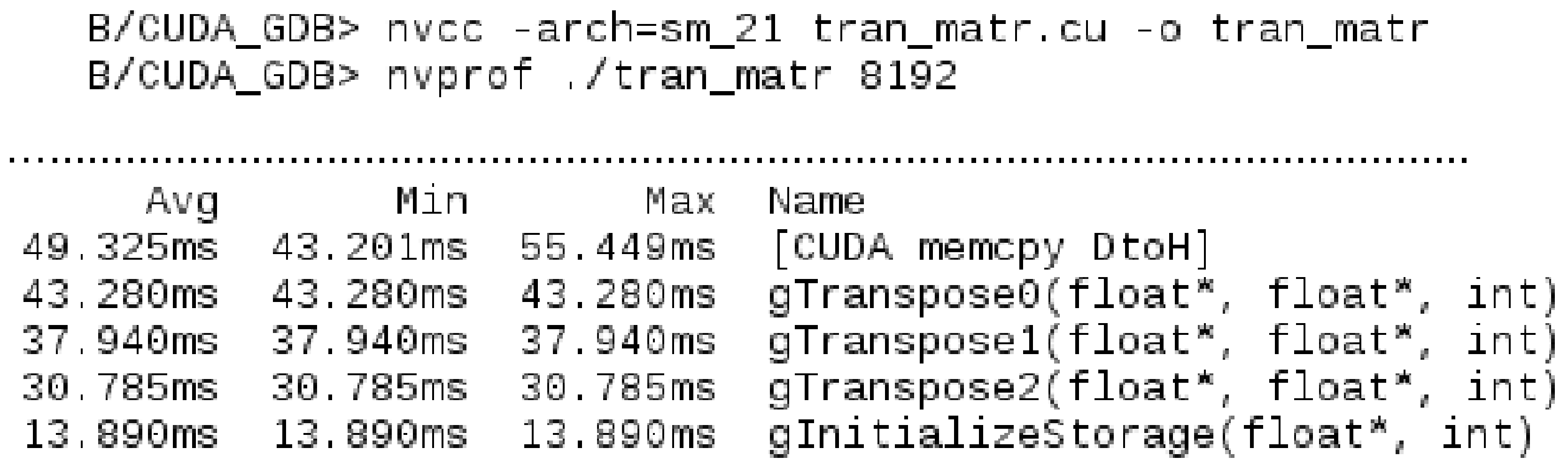
```
     Avg        Min        Max   Name
55.572ms   44.880ms   66.263ms   [CUDA memcpy DtoH]
38.623ms   38.623ms   38.623ms   gTranspose1(float*, float*, int)
34.543ms   34.543ms   34.543ms   gTranspose0(float*, float*, int)
26.967ms   26.967ms   26.967ms   gTranspose2(float*, float*, int)
13.503ms   13.503ms   13.503ms   gInitializeStorage(float*, int)
```

Время выполнения ядра с разными архитектурами CUDA (GeForce 650Ti, *Kepler*, CUDA 6.5, *compute capabilities 2.1*)



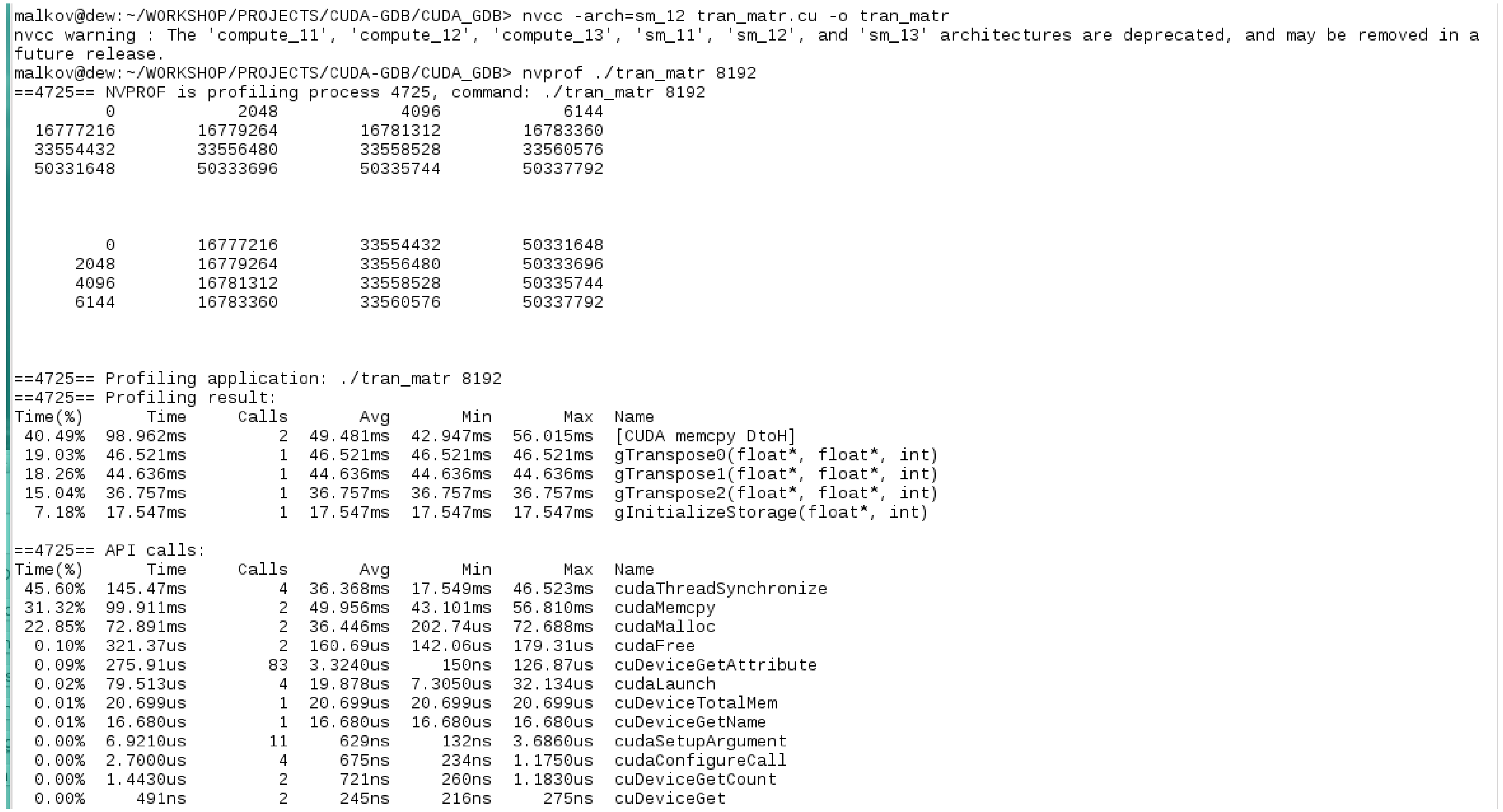
```
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-GDB/CUDA_GDB> nvcc -arch=sm_21 tran_matr.cu -o tran_matr
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-GDB/CUDA_GDB> nvprof ./tran_matr 8192
==4657== NVPROF is profiling process 4657, command: ./tran_matr 8192
         0         2048         4096         6144
  16777216     16779264     16781312     16783360
  33554432     33556480     33558528     33560576
  50331648     50333696     50335744     50337792




         0     16777216     33554432     50331648
      2048     16779264     33556480     50333696
      4096     16781312     33558528     50335744
      6144     16783360     33560576     50337792




==4657== Profiling application: ./tran_matr 8192
==4657== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 43.93%  98.649ms         2  49.325ms  43.201ms  55.449ms  [CUDA memcpy DtoH]
 19.27%  43.280ms         1  43.280ms  43.280ms  43.280ms  gTranspose0(float*, float*, int)
 16.90%  37.940ms         1  37.940ms  37.940ms  37.940ms  gTranspose1(float*, float*, int)
 13.71%  30.785ms         1  30.785ms  30.785ms  30.785ms  gTranspose2(float*, float*, int)
  6.19%  13.890ms         1  13.890ms  13.890ms  13.890ms  gInitializeStorage(float*, int)

==4657== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 48.14%  125.90ms         4  31.476ms  13.892ms  43.282ms  cudaThreadSynchronize
 37.90%  99.118ms         2  49.559ms  43.352ms  55.766ms  cudaMemcpy
 13.69%  35.801ms         2  17.901ms  177.24us  35.624ms  cudaMalloc
  0.12%  314.36us         2  157.18us  138.16us  176.19us  cudaFree
  0.10%  266.18us        83  3.2070us     143ns  120.28us  cuDeviceGetAttribute
  0.03%  75.920us         4  18.980us  7.0430us  30.630us  cudaLaunch
  0.01%  20.542us         1  20.542us  20.542us  20.542us  cuDeviceTotalMem
  0.01%  16.244us         1  16.244us  16.244us  16.244us  cuDeviceGetName
  0.00%  6.3660us        11     578ns     131ns  3.5570us  cudaSetupArgument
  0.00%  2.6520us         4     663ns     237ns  1.2360us  cudaConfigureCall
  0.00%  1.2360us         2     618ns     245ns     991ns  cuDeviceGetCount
  0.00%     544ns         2     272ns     208ns     336ns  cuDeviceGet
```

Время выполнения ядра с разными архитектурами CUDA (GeForce 650Ti, *Kepler*, CUDA 6.5, *compute capabilities 2.1*)



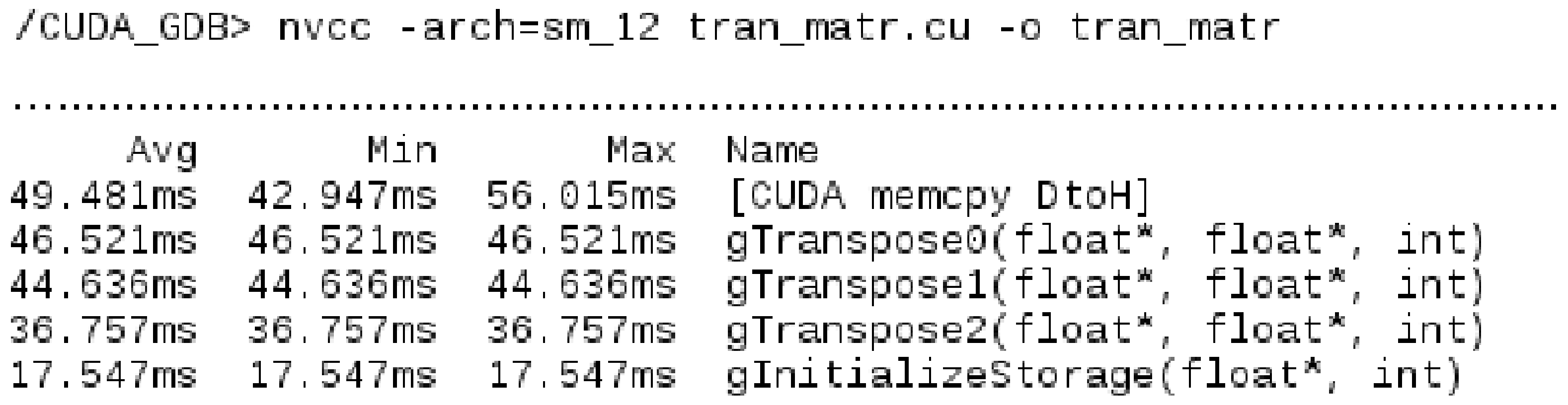
```
    B/CUDA_GDB> nvcc -arch=sm_21 tran_matr.cu -o tran_matr
    B/CUDA_GDB> nvprof ./tran_matr 8192

.............................................................................
      Avg         Min         Max    Name
 49.325ms   43.201ms   55.449ms   [CUDA memcpy DtoH]
 43.280ms   43.280ms   43.280ms   gTranspose0(float*, float*, int)
 37.940ms   37.940ms   37.940ms   gTranspose1(float*, float*, int)
 30.785ms   30.785ms   30.785ms   gTranspose2(float*, float*, int)
 13.890ms   13.890ms   13.890ms   gInitializeStorage(float*, int)
```

Время выполнения ядра с разными архитектурами CUDA (*Fermi*, CUDA 6.5, *compute capabilities 1.2*)



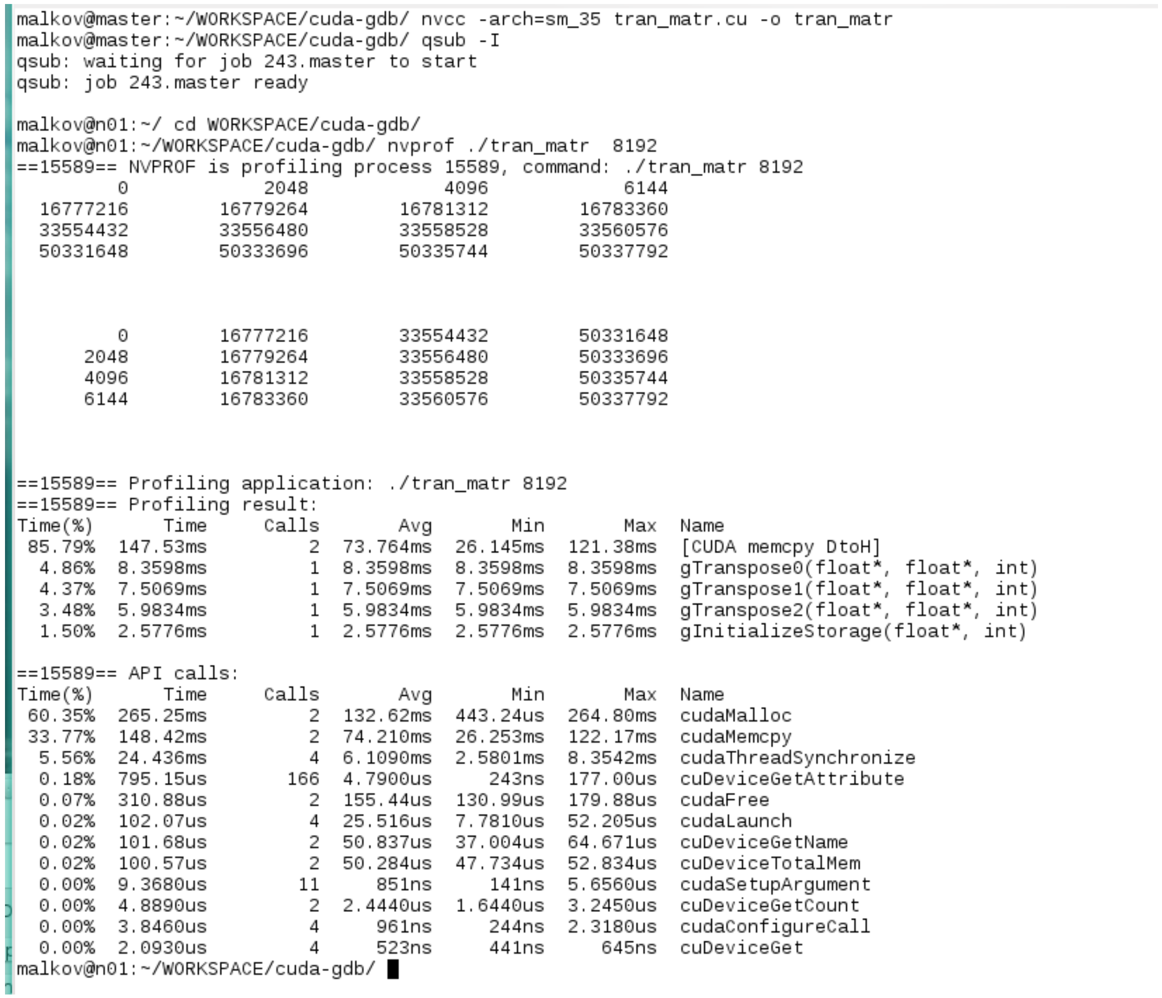
```
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-GDB/CUDA_GDB> nvcc -arch=sm_12 tran_matr.cu -o tran_matr
nvcc warning : The 'compute_11', 'compute_12', 'compute_13', 'sm_11', 'sm_12', and 'sm_13' architectures are deprecated, and may be removed in a
future release.
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-GDB/CUDA_GDB> nvprof ./tran_matr 8192
==4725== NVPROF is profiling process 4725, command: ./tran_matr 8192
         0         2048         4096         6144
  16777216     16779264     16781312     16783360
  33554432     33556480     33558528     33560576
  50331648     50333696     50335744     50337792



         0     16777216     33554432     50331648
      2048     16779264     33556480     50333696
      4096     16781312     33558528     50335744
      6144     16783360     33560576     50337792



==4725== Profiling application: ./tran_matr 8192
==4725== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 40.49%  98.962ms         2  49.481ms  42.947ms  56.015ms  [CUDA memcpy DtoH]
 19.03%  46.521ms         1  46.521ms  46.521ms  46.521ms  gTranspose0(float*, float*, int)
 18.26%  44.636ms         1  44.636ms  44.636ms  44.636ms  gTranspose1(float*, float*, int)
 15.04%  36.757ms         1  36.757ms  36.757ms  36.757ms  gTranspose2(float*, float*, int)
  7.18%  17.547ms         1  17.547ms  17.547ms  17.547ms  gInitializeStorage(float*, int)

==4725== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 45.60%  145.47ms         4  36.368ms  17.549ms  46.523ms  cudaThreadSynchronize
 31.32%  99.911ms         2  49.956ms  43.101ms  56.810ms  cudaMemcpy
 22.85%  72.891ms         2  36.446ms  202.74us  72.688ms  cudaMalloc
  0.10%  321.37us         2  160.69us  142.06us  179.31us  cudaFree
  0.09%  275.91us        83  3.3240us     150ns  126.87us  cuDeviceGetAttribute
  0.02%  79.513us         4  19.878us  7.3050us  32.134us  cudaLaunch
  0.01%  20.699us         1  20.699us  20.699us  20.699us  cuDeviceTotalMem
  0.01%  16.680us         1  16.680us  16.680us  16.680us  cuDeviceGetName
  0.00%  6.9210us        11     629ns     132ns  3.6860us  cudaSetupArgument
  0.00%  2.7000us         4     675ns     234ns  1.1750us  cudaConfigureCall
  0.00%  1.4430us         2     721ns     260ns  1.1830us  cuDeviceGetCount
  0.00%     491ns         2     245ns     216ns     275ns  cuDeviceGet
```

Время выполнения ядра с разными архитектурами CUDA (*Fermi*, CUDA 6.5, *compute capabilities 1.2*)



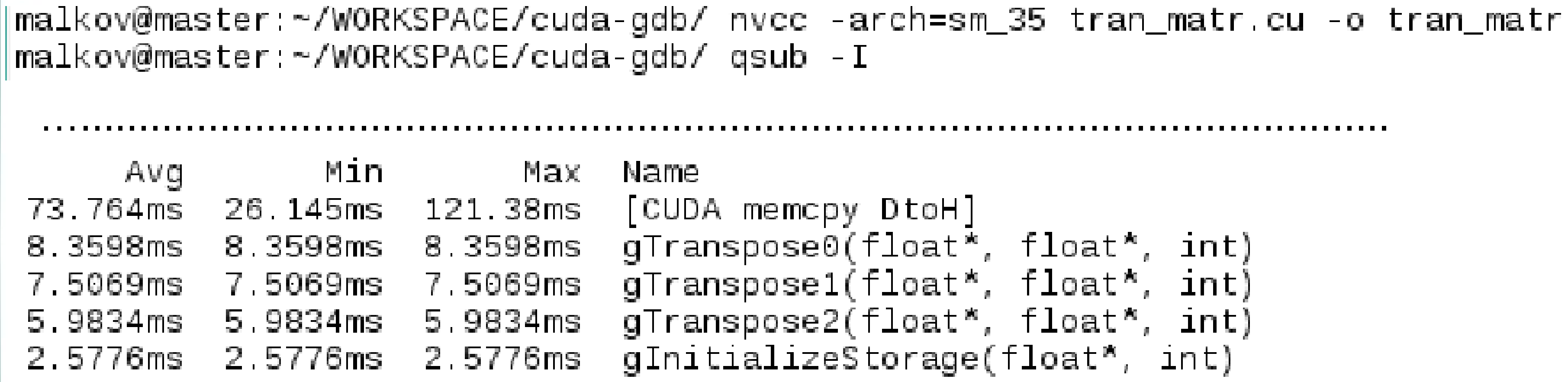
```
/CUDA_GDB> nvcc -arch=sm_12 tran_matr.cu -o tran_matr

..............................................................................
     Avg          Min          Max   Name
49.481ms    42.947ms    56.015ms   [CUDA memcpy DtoH]
46.521ms    46.521ms    46.521ms   gTranspose0(float*, float*, int)
44.636ms    44.636ms    44.636ms   gTranspose1(float*, float*, int)
36.757ms    36.757ms    36.757ms   gTranspose2(float*, float*, int)
17.547ms    17.547ms    17.547ms   gInitializeStorage(float*, int)
```

Время выполнения ядра с разными архитектурами CUDA (Tesla K40m, *Kepler*, CUDA 7.5, *compute capabilities 3.5*)



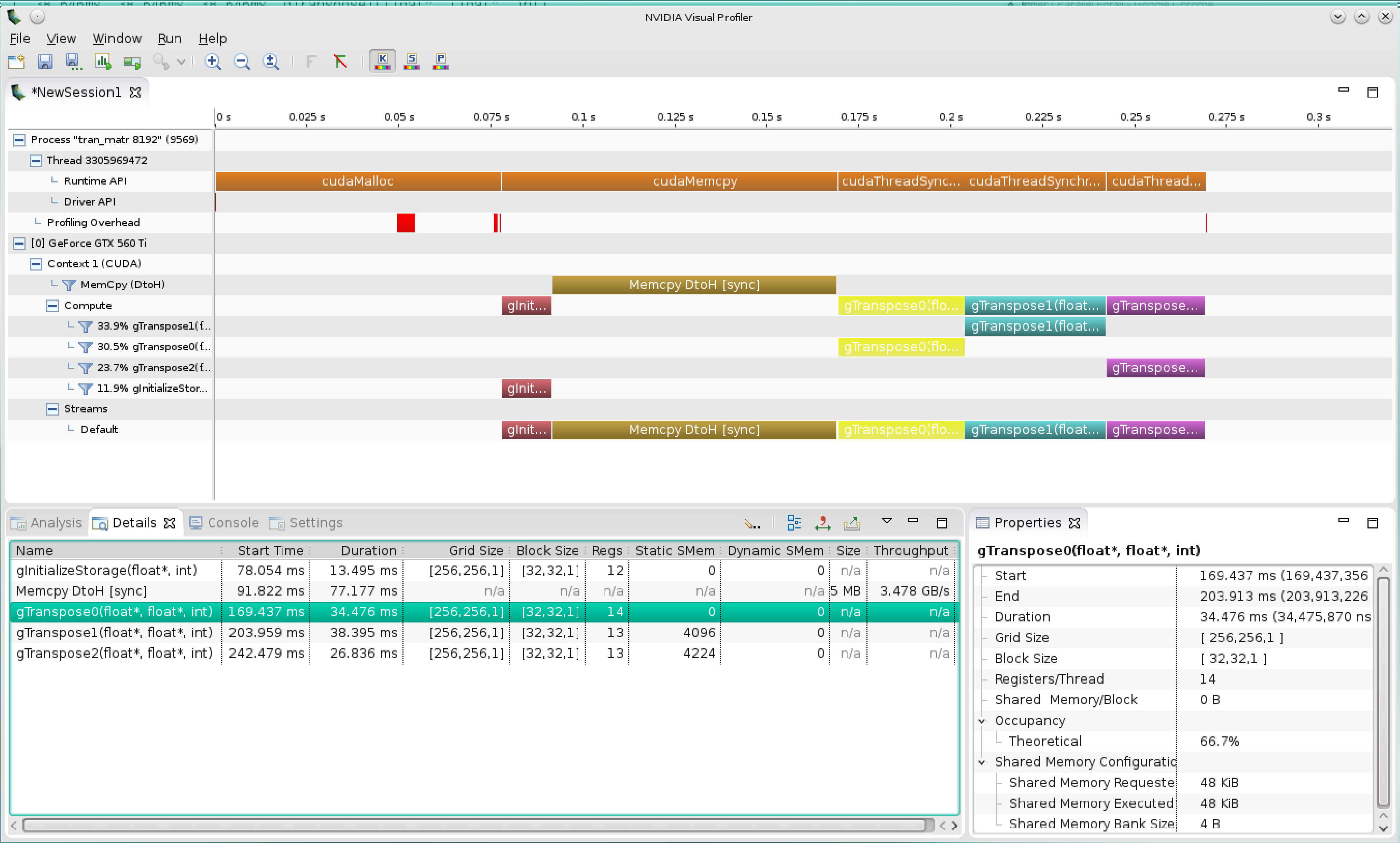
```
malkov@master:~/WORKSPACE/cuda-gdb/ nvcc -arch=sm_35 tran_matr.cu -o tran_matr
malkov@master:~/WORKSPACE/cuda-gdb/ qsub -I
qsub: waiting for job 243.master to start
qsub: job 243.master ready

malkov@n01:~/ cd WORKSPACE/cuda-gdb/
malkov@n01:~/WORKSPACE/cuda-gdb/ nvprof ./tran_matr  8192
==15589== NVPROF is profiling process 15589, command: ./tran_matr 8192
         0           2048           4096           6144
  16777216       16779264       16781312       16783360
  33554432       33556480       33558528       33560576
  50331648       50333696       50335744       50337792



         0       16777216       33554432       50331648
      2048       16779264       33556480       50333696
      4096       16781312       33558528       50335744
      6144       16783360       33560576       50337792



==15589== Profiling application: ./tran_matr 8192
==15589== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 85.79%  147.53ms         2  73.764ms  26.145ms  121.38ms  [CUDA memcpy DtoH]
  4.86%  8.3598ms         1  8.3598ms  8.3598ms  8.3598ms  gTranspose0(float*, float*, int)
  4.37%  7.5069ms         1  7.5069ms  7.5069ms  7.5069ms  gTranspose1(float*, float*, int)
  3.48%  5.9834ms         1  5.9834ms  5.9834ms  5.9834ms  gTranspose2(float*, float*, int)
  1.50%  2.5776ms         1  2.5776ms  2.5776ms  2.5776ms  gInitializeStorage(float*, int)

==15589== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 60.35%  265.25ms         2  132.62ms  443.24us  264.80ms  cudaMalloc
 33.77%  148.42ms         2  74.210ms  26.253ms  122.17ms  cudaMemcpy
  5.56%  24.436ms         4  6.1090ms  2.5801ms  8.3542ms  cudaThreadSynchronize
  0.18%  795.15us       166  4.7900us     243ns  177.00us  cuDeviceGetAttribute
  0.07%  310.88us         2  155.44us  130.99us  179.88us  cudaFree
  0.02%  102.07us         4  25.516us  7.7810us  52.205us  cudaLaunch
  0.02%  101.68us         2  50.837us  37.004us  64.671us  cuDeviceGetName
  0.02%  100.57us         2  50.284us  47.734us  52.834us  cuDeviceTotalMem
  0.00%  9.3680us        11     851ns     141ns  5.6560us  cudaSetupArgument
  0.00%  4.8890us         2  2.4440us  1.6440us  3.2450us  cuDeviceGetCount
  0.00%  3.8460us         4     961ns     244ns  2.3180us  cudaConfigureCall
  0.00%  2.0930us         4     523ns     441ns     645ns  cuDeviceGet
malkov@n01:~/WORKSPACE/cuda-gdb/
```

Время выполнения ядра с разными архитектурами CUDA (Tesla K40m, *Kepler*, CUDA 7.5, *compute capabilities 3.5*)



```
malkov@master:~/WORKSPACE/cuda-gdb/ nvcc -arch=sm_35 tran_matr.cu -o tran_matr
malkov@master:~/WORKSPACE/cuda-gdb/ qsub -I

 .....................................................................

      Avg         Min         Max    Name
73.764ms   26.145ms   121.38ms   [CUDA memcpy DtoH]
8.3598ms   8.3598ms   8.3598ms   gTranspose0(float*, float*, int)
7.5069ms   7.5069ms   7.5069ms   gTranspose1(float*, float*, int)
5.9834ms   5.9834ms   5.9834ms   gTranspose2(float*, float*, int)
2.5776ms   2.5776ms   2.5776ms   gInitializeStorage(float*, int)
```

NVidia Visual Profiler *(nvvp)*

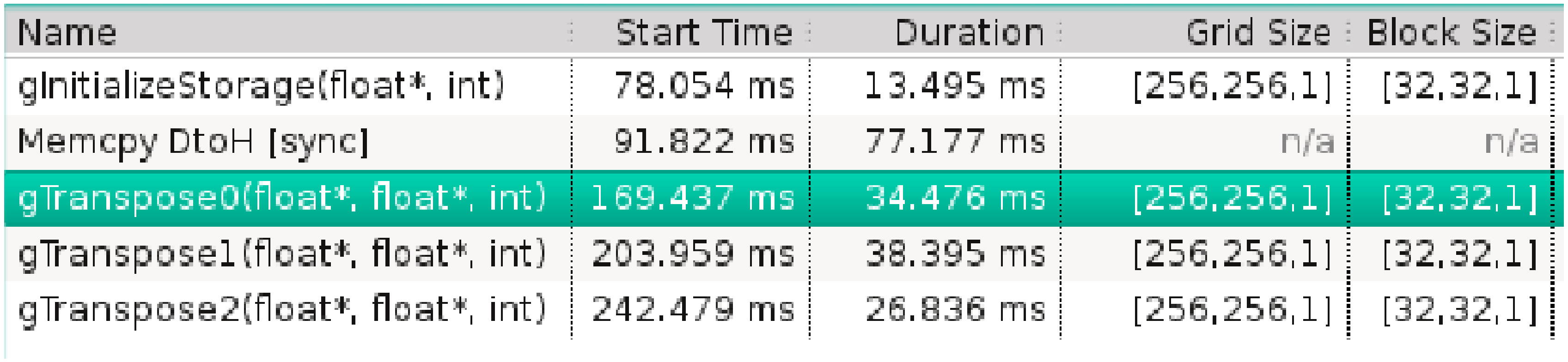


Name	Start Time	Duration	Grid Size	Block Size	Regs	Static SMem	Dynamic SMem	Size	Throughput
gInitializeStorage(float*, int)	78.054 ms	13.495 ms	[256,256,1]	[32,32,1]	12	0	0	n/a	n/a
Memcpy DtoH [sync]	91.822 ms	77.177 ms	n/a	n/a	n/a	n/a	n/a	5 MB	3.478 GB/s
gTranspose0(float*, float*, int)	169.437 ms	34.476 ms	[256,256,1]	[32,32,1]	14	0	0	n/a	n/a
gTranspose1(float*, float*, int)	203.959 ms	38.395 ms	[256,256,1]	[32,32,1]	13	4096	0	n/a	n/a
gTranspose2(float*, float*, int)	242.479 ms	26.836 ms	[256,256,1]	[32,32,1]	13	4224	0	n/a	n/a

gTranspose0(float*, float*, int)

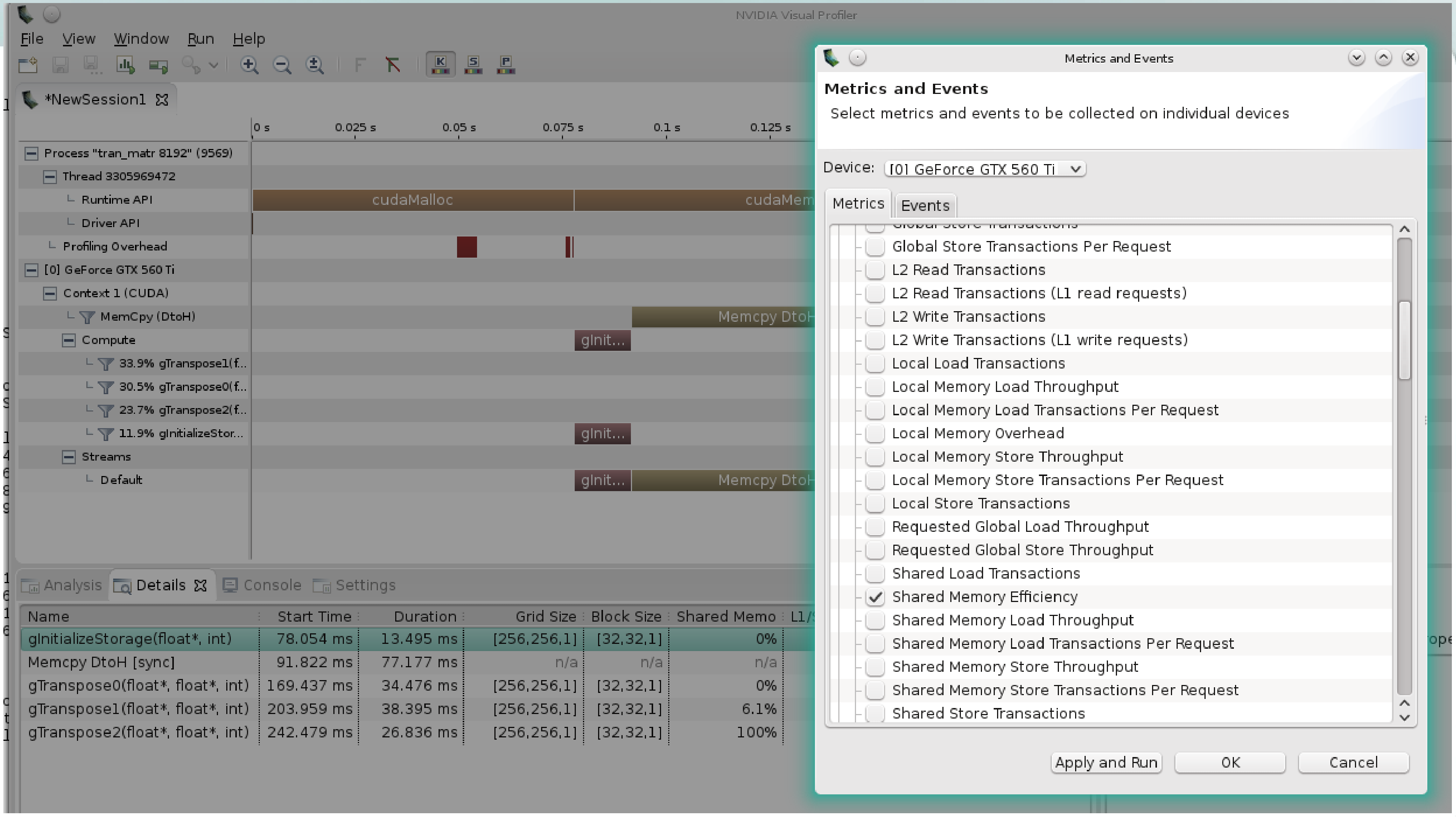
Start	169.437 ms (169,437,356
End	203.913 ms (203,913,226
Duration	34.476 ms (34,475,870 ns
Grid Size	[256,256,1]
Block Size	[32,32,1]
Registers/Thread	14
Shared Memory/Block	0 B
Occupancy	
Theoretical	66.7%
Shared Memory Configuratic	
Shared Memory Requeste	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B

NVidia Visual Profiler *(nvvp)*



Name	Start Time	Duration	Grid Size	Block Size
gInitializeStorage(float*, int)	78.054 ms	13.495 ms	[256,256,1]	[32,32,1]
Memcpy DtoH [sync]	91.822 ms	77.177 ms	n/a	n/a
gTranspose0(float*, float*, int)	169.437 ms	34.476 ms	[256,256,1]	[32,32,1]
gTranspose1(float*, float*, int)	203.959 ms	38.395 ms	[256,256,1]	[32,32,1]
gTranspose2(float*, float*, int)	242.479 ms	26.836 ms	[256,256,1]	[32,32,1]

Добавление событий и счетчиков (nvvp)



Name	Start Time	Duration	Grid Size	Block Size	Shared Memo	L1/
gInitializeStorage(float*, int)	78.054 ms	13.495 ms	[256,256,1]	[32,32,1]	0%	
Memcpy DtoH [sync]	91.822 ms	77.177 ms	n/a	n/a	n/a	
gTranspose0(float*, float*, int)	169.437 ms	34.476 ms	[256,256,1]	[32,32,1]	0%	
gTranspose1(float*, float*, int)	203.959 ms	38.395 ms	[256,256,1]	[32,32,1]	6.1%	
gTranspose2(float*, float*, int)	242.479 ms	26.836 ms	[256,256,1]	[32,32,1]	100%	

Metrics and Events

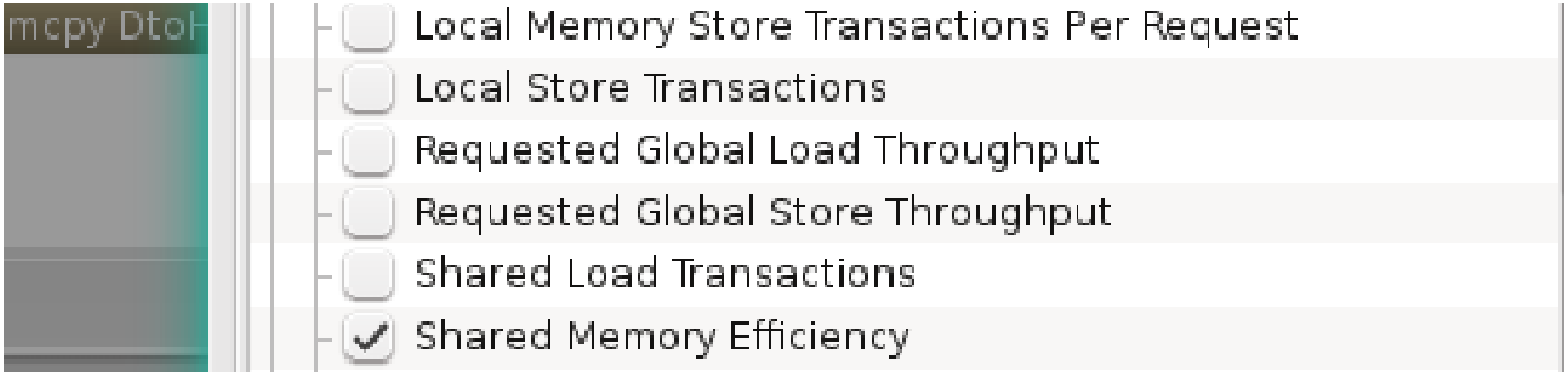
Select metrics and events to be collected on individual devices

Device: [0] GeForce GTX 560 Ti

Metrics | Events

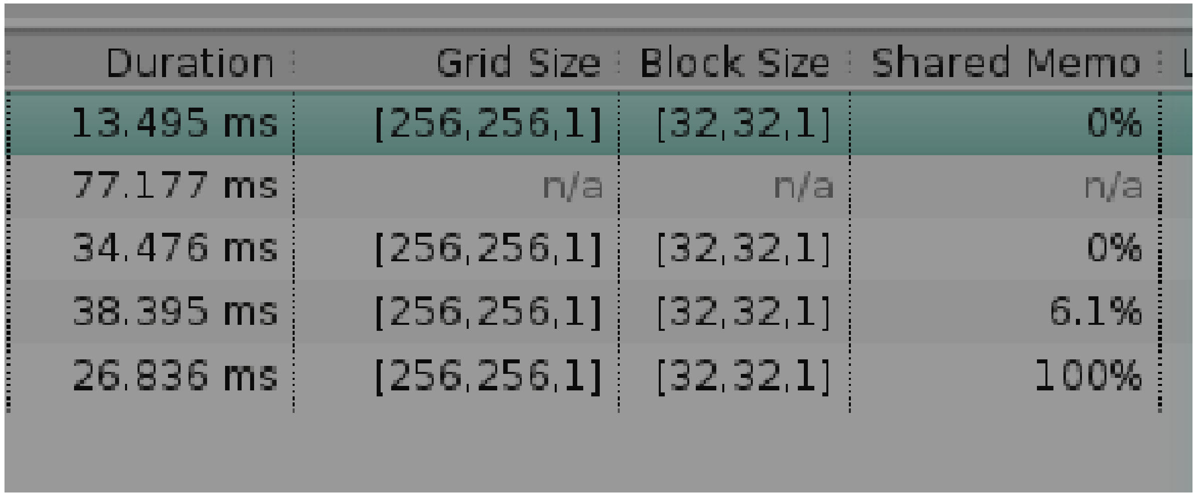
- [ ] Global Store Transactions Per Request
- [ ] L2 Read Transactions
- [ ] L2 Read Transactions (L1 read requests)
- [ ] L2 Write Transactions
- [ ] L2 Write Transactions (L1 write requests)
- [ ] Local Load Transactions
- [ ] Local Memory Load Throughput
- [ ] Local Memory Load Transactions Per Request
- [ ] Local Memory Overhead
- [ ] Local Memory Store Throughput
- [ ] Local Memory Store Transactions Per Request
- [ ] Local Store Transactions
- [ ] Requested Global Load Throughput
- [ ] Requested Global Store Throughput
- [ ] Shared Load Transactions
- [✓] Shared Memory Efficiency
- [ ] Shared Memory Load Throughput
- [ ] Shared Memory Load Transactions Per Request
- [ ] Shared Memory Store Throughput
- [ ] Shared Memory Store Transactions Per Request
- [ ] Shared Store Transactions

Добавление событий и счетчиков (nvvp)



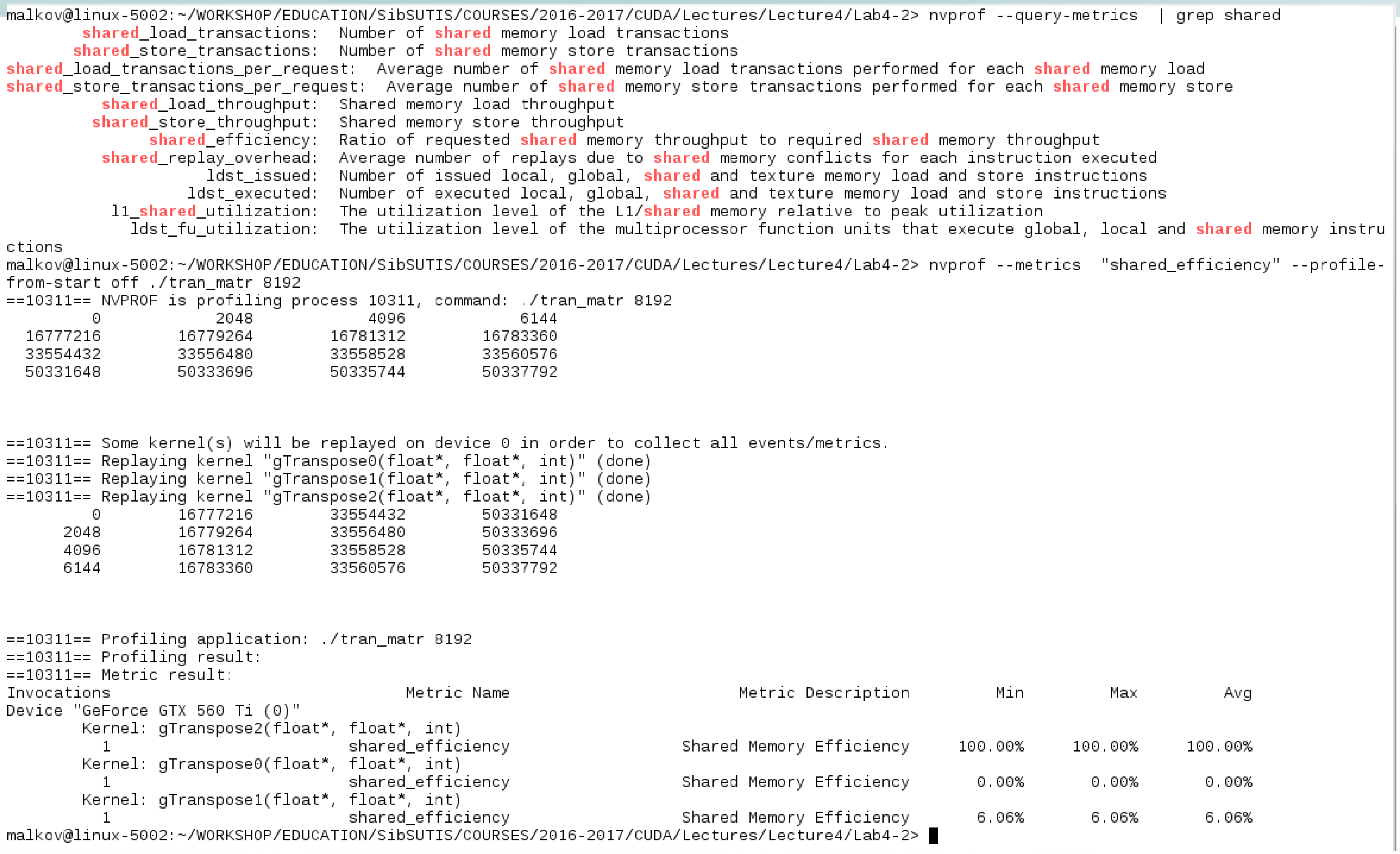
mcpy DtoH

- [ ] Local Memory Store Transactions Per Request
- [ ] Local Store Transactions
- [ ] Requested Global Load Throughput
- [ ] Requested Global Store Throughput
- [ ] Shared Load Transactions
- [✓] Shared Memory Efficiency



Duration	Grid Size	Block Size	Shared Memo	L
13.495 ms	[256,256,1]	[32,32,1]	0%	
77.177 ms	n/a	n/a	n/a	
34.476 ms	[256,256,1]	[32,32,1]	0%	
38.395 ms	[256,256,1]	[32,32,1]	6.1%	
26.836 ms	[256,256,1]	[32,32,1]	100%	

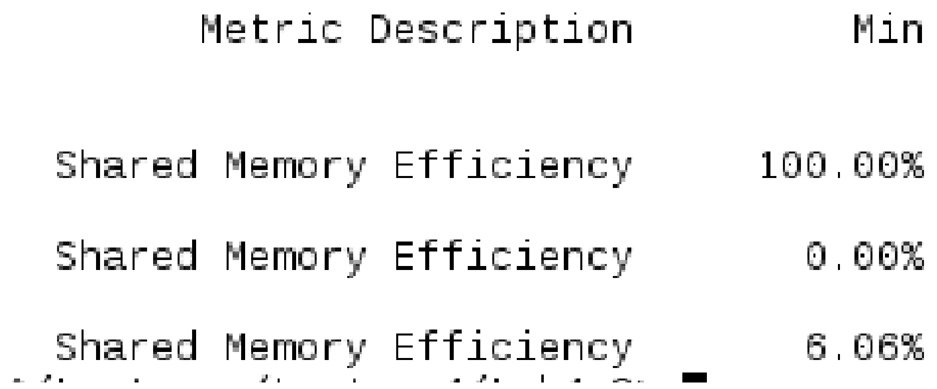
Добавление событий и счетчиков (nvprof)



```
malkov@linux-5002:~/WORKSHOP/EDUCATION/SibSUTIS/COURSES/2016-2017/CUDA/Lectures/Lecture4/Lab4-2> nvprof --query-metrics  | grep shared
        shared_load_transactions:  Number of shared memory load transactions
       shared_store_transactions:  Number of shared memory store transactions
shared_load_transactions_per_request:  Average number of shared memory load transactions performed for each shared memory load
shared_store_transactions_per_request:  Average number of shared memory store transactions performed for each shared memory store
          shared_load_throughput:  Shared memory load throughput
         shared_store_throughput:  Shared memory store throughput
               shared_efficiency:  Ratio of requested shared memory throughput to required shared memory throughput
          shared_replay_overhead:  Average number of replays due to shared memory conflicts for each instruction executed
                     ldst_issued:  Number of issued local, global, shared and texture memory load and store instructions
                   ldst_executed:  Number of executed local, global, shared and texture memory load and store instructions
           l1_shared_utilization:  The utilization level of the L1/shared memory relative to peak utilization
             ldst_fu_utilization:  The utilization level of the multiprocessor function units that execute global, local and shared memory instru
ctions
malkov@linux-5002:~/WORKSHOP/EDUCATION/SibSUTIS/COURSES/2016-2017/CUDA/Lectures/Lecture4/Lab4-2> nvprof --metrics  "shared_efficiency" --profile-
from-start off ./tran_matr 8192
==10311== NVPROF is profiling process 10311, command: ./tran_matr 8192
         0          2048          4096          6144
  16777216      16779264      16781312      16783360
  33554432      33556480      33558528      33560576
  50331648      50333696      50335744      50337792



==10311== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
==10311== Replaying kernel "gTranspose0(float*, float*, int)" (done)
==10311== Replaying kernel "gTranspose1(float*, float*, int)" (done)
==10311== Replaying kernel "gTranspose2(float*, float*, int)" (done)
         0      16777216      33554432      50331648
      2048      16779264      33556480      50333696
      4096      16781312      33558528      50335744
      6144      16783360      33560576      50337792



==10311== Profiling application: ./tran_matr 8192
==10311== Profiling result:
==10311== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 560 Ti (0)"
        Kernel: gTranspose2(float*, float*, int)
          1                         shared_efficiency                 Shared Memory Efficiency     100.00%     100.00%     100.00%
        Kernel: gTranspose0(float*, float*, int)
          1                         shared_efficiency                 Shared Memory Efficiency       0.00%       0.00%       0.00%
        Kernel: gTranspose1(float*, float*, int)
          1                         shared_efficiency                 Shared Memory Efficiency       6.06%       6.06%       6.06%
malkov@linux-5002:~/WORKSHOP/EDUCATION/SibSUTIS/COURSES/2016-2017/CUDA/Lectures/Lecture4/Lab4-2> █
```

Добавление событий и счетчиков (nvprof)

```
> nvprof –metrics “shared_efficiency”  ./tran_matr 8192
```



```
          Metric Description        Min

    Shared Memory Efficiency    100.00%

    Shared Memory Efficiency      0.00%

    Shared Memory Efficiency      6.06%
```