

Лекция 2

СОДЕРЖАНИЕ

- модель выполнения CUDA
- иерархия памяти;
- объединение запросов к глобальной памяти (coalescing);

Оптимальное количество блоков и нитей в блоках (заполняемость мультипроцессоров)

Для архитектуры Fermi:

- максимальное кол-во варпов, выполняемых одновременно на одном мультипроцессоре равно 48;
- максимальное кол-во блоков выполняемых одновременно на одном мультипроцессоре равно 8;

Пример.

32 нити в блоке, 512 блоков в гриде:

Кол-во одновременно выполняемых нитей= 32×8 ,

Максимальное кол-во= 48×32 ,

Заполняемость= $32 \times 8 / 48 \times 32 = 1/6 = 16.(6)\%$.

512 нитей в блоке, 32 блока в гриде:

Кол-во одновременно выполняемых нитей= 512×8 ,

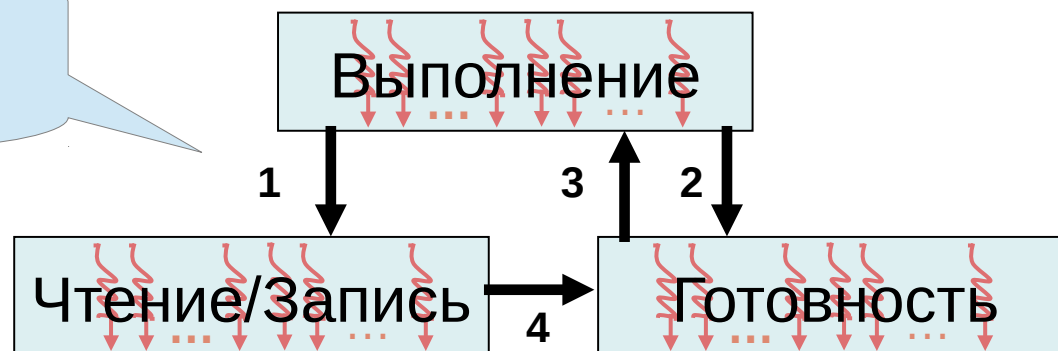
Максимальное кол-во= 48×32 ,

Заполняемость= $512 \times 8 / 48 \times 32 = 266.(6)\%$.

Оптимальное кол-во нитей в блоке (для Fermi): $48 \times 32 / 8 = 192$

Оптимальное количество блоков и нитей в блоках (сокращение латентности)

Преимущества
перекрывания/многозадачности:



Кол-во варпов $\% 32 == 0$ && Кол-во варпов $/ 32 > 1$
Кол-во блоков \geq Кол-во мультипроцессоров

Вычислительные возможности архитектур Fermi и Kepler

	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K	16K	16K	16K
	48K	48K	32K	32K
			48K	48K

Compute capabilities (вычислительные возможности)

Compute capabilities (вычислительные возможности) представляют спецификацию GPU. Особенности “вычислительных возможностей” включают допустимость операций с плавающей точкой, допустимость атомарных операций, возможность синхронизации нитей, кэшируемость глобальной памяти и т.д. Описание различных версий Compute capabilities можно найти, например, в CUDA C Programming Guid – руководстве по CUDA C компании NVIDIA.

Архитектура GPU	Compute capabilities	Версия CUDA
Tesla	1.*	CUDA 2.*-3.*
Fermi	2.*	CUDA 4.*-5.*
Kepler	3.*	CUDA 5.*
Maxwell	5.*	CUDA 6.*-7.*
Pascal	6.*	CUDA 8
Volta	7.*	

Характеристики GPU: GTX 560 Ti(Fermi), GTX 650(Kepler)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 560 Ti"

CUDA Driver Version / Runtime Version 5.0 / 5.0

CUDA Capability Major/Minor version number: 2.1

Total amount of global memory: 2048 MBytes (2147024896 bytes)

(8) Multiprocessors x (48) CUDA Cores/MP: 384 CUDA Cores

GPU Clock rate: 1645 MHz (1.64 GHz)

Memory Clock rate: 2004 Mhz

Memory Bus Width: 256-bit

L2 Cache Size: 524288 bytes

Max Texture Dimension Size (x,y,z) 1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)

Max Layered Texture Size (dim) x layers 1D=(16384) x 2048, 2D=(16384,16384) x 2048

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 32768

Warp size: 32

Maximum number of threads per multiprocessor: 1536

Maximum number of threads per block: 1024

Maximum sizes of each dimension of a block: 1024 x 1024 x 64

Maximum sizes of each dimension of a grid: 65535 x 65535 x 65535

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 1 copy engine(s)

Run time limit on kernels: Yes

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled

Device supports Unified Addressing (UVA): Yes

Device PCI Bus ID / PCI location ID: 1 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 650"

CUDA Driver Version / Runtime Version 5.5 / 5.5

CUDA Capability Major/Minor version number: 3.0

Total amount of global memory: 2048 MBytes (2147155968 bytes)

(2) Multiprocessors x (192) CUDA Cores/MP: 384 CUDA Cores

GPU Clock rate: 1110 MHz (1.11 GHz)

Memory Clock rate: 2500 Mhz

Memory Bus Width: 128-bit

L2 Cache Size: 262144 bytes

Max Texture Dimension Size (x,y,z) 1D=(65536), 2D=(65536,65536), 3D=(4096,4096,4096)

Max Layered Texture Size (dim) x layers 1D=(16384) x 2048, 2D=(16384,16384) x 2048

Total amount of constant memory: 65536 bytes

Total amount of shared memory per block: 49152 bytes

Total number of registers available per block: 65536

Warp size: 32

Maximum number of threads per multiprocessor: 2048

Maximum number of threads per block: 1024

Maximum sizes of each dimension of a block: 1024 x 1024 x 64

Maximum sizes of each dimension of a grid: 65535 x 65535 x 65535

Maximum memory pitch: 2147483647 bytes

Texture alignment: 512 bytes

Concurrent copy and kernel execution: Yes with 1 copy engine(s)

Run time limit on kernels: Yes

Integrated GPU sharing Host Memory: No

Support host page-locked memory mapping: Yes

Alignment requirement for Surfaces: Yes

Device has ECC support: Disabled

Device supports Unified Addressing (UVA): Yes

Device PCI Bus ID / PCI location ID: 1 / 0

Compute Mode:

< Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

Установить оптимальную конфигурацию нитей для своего устройства (задание 1).

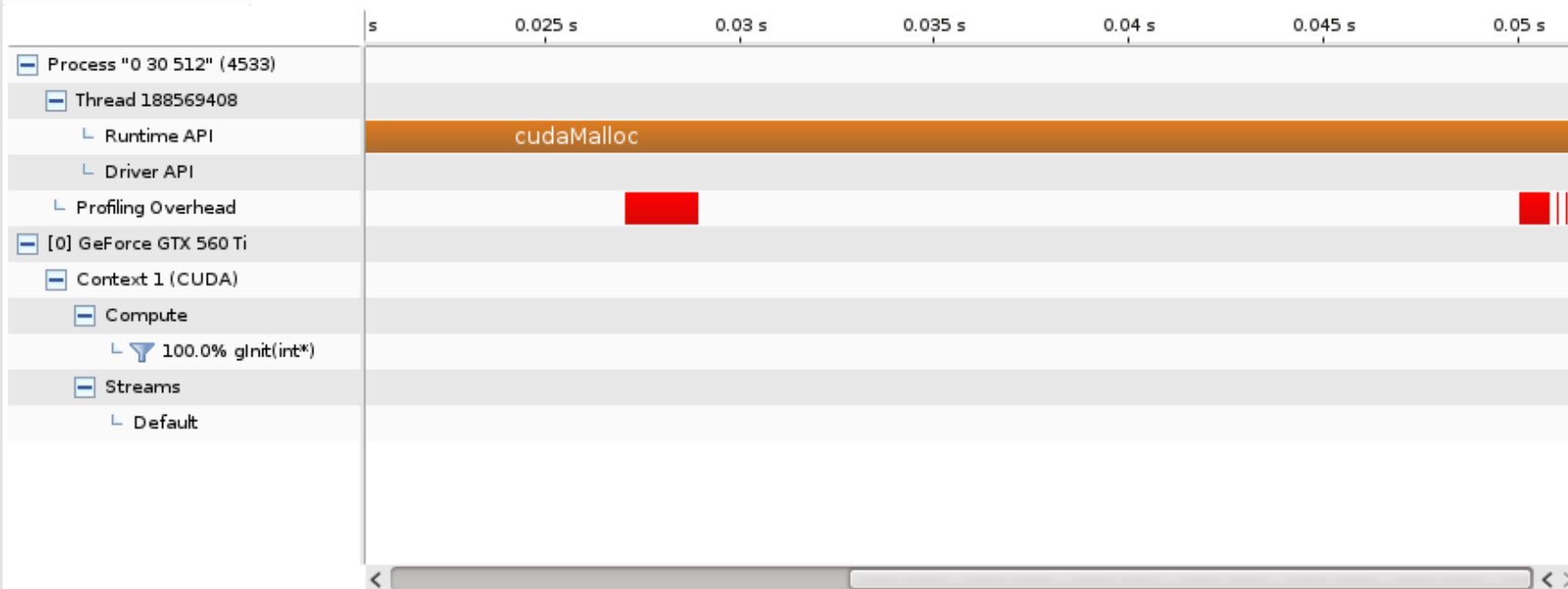
NVIDIA_CUDA-7.5_Samples/1_Uutilities/deviceQuery/**deviceQuery.cpp**

```
cudaSetDevice(dev);
cudaDeviceProp deviceProp;
cudaGetDeviceProperties(&deviceProp, dev);

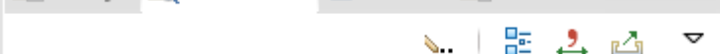
.....
printf(" Total amount of constant memory: %lu bytes\n", deviceProp.totalConstMem);
printf(" Total amount of shared memory per block: %lu bytes\n",
                                             deviceProp.sharedMemPerBlock);
printf(" Total number of registers available per block: %d\n", deviceProp.regsPerBlock);
printf(" Warp size: %d\n", deviceProp.warpSize);
printf(" Maximum number of threads per multiprocessor: %d\n",
                                             deviceProp.maxThreadsPerMultiProcessor);
printf(" Maximum number of threads per block: %d\n", deviceProp.maxThreadsPerBlock);
.....
```



*NewSession1



Analys Detail Conso Settin



Name	Start Time	Duration	Grid Size	Block S	R
glInit(ir	51.414 ms	7.235 μ s	[30,1,1]	[512,1,1]	
glInit(ir	51.384 ms	10.385 μ s	[512,1,1]	[30,1,1]	

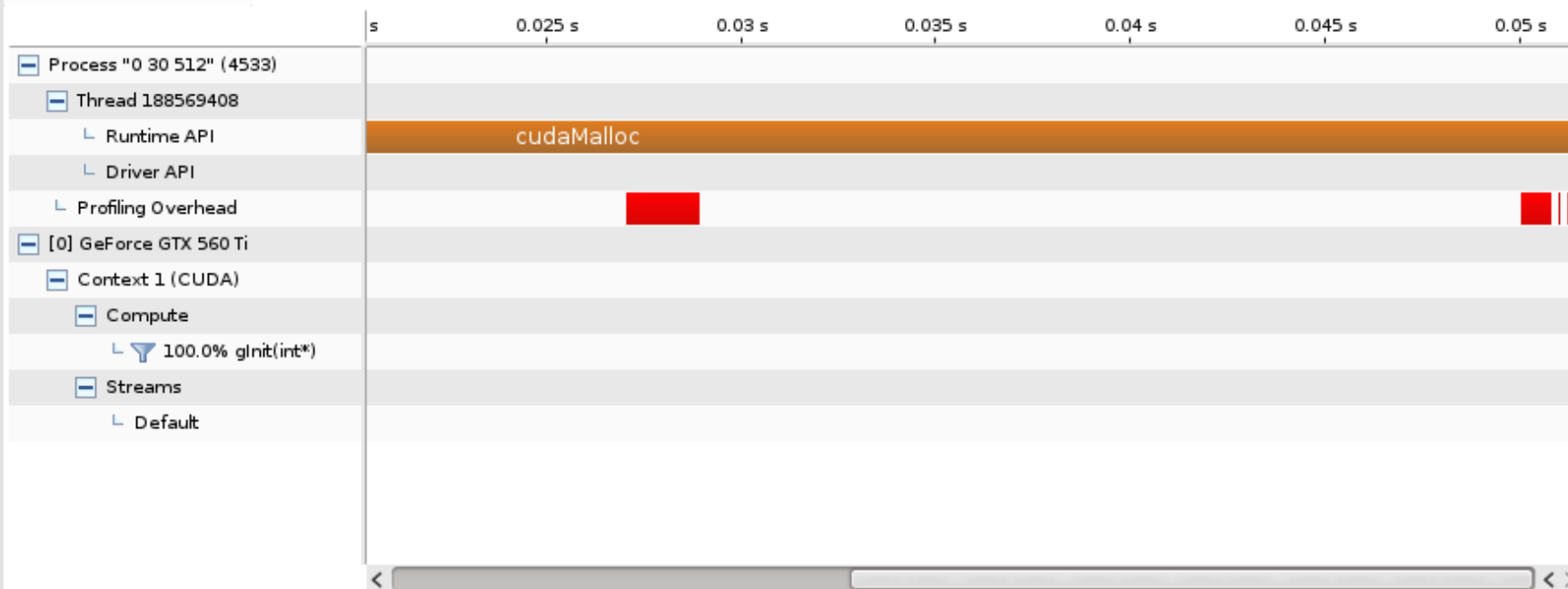
Properties

glInit(int*)

Start	51.384 ms (51,384,414)
End	51.395 ms (51,394,799)
Duration	10.385 μ s
Grid Size	[512,1,1]
Block Size	[30,1,1]
Registers/Thread	7
Shared Memory/Block	0 B
Occupancy	
Theoretical	16.7%
Shared Memory Configuration	
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B



*NewSession1



Analys Detail Conso Settin

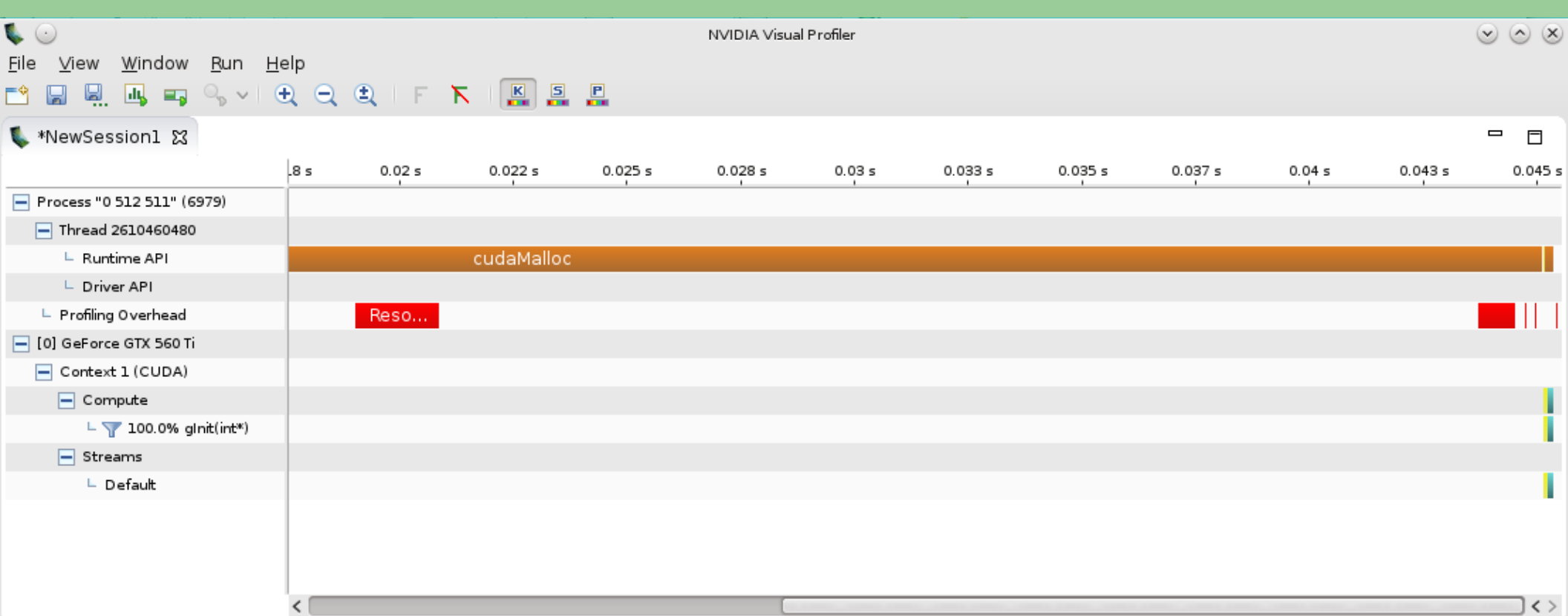


Name	Start Time	Duration	Grid Size	Block S	R
glnit(ir	51.414 ms	7.235 µs	[30,1,1]	[512,1,1]	
glnit(ir	51.384 ms	10.385 µs	[512,1,1]	[30,1,1]	

Properties

glnit(int*)

Start	51.414 ms (51,413,634)
End	51.421 ms (51,420,869)
Duration	7.235 µs
Grid Size	[30,1,1]
Block Size	[512,1,1]
Registers/Thread	7
Shared Memory/Block	0 B
Occupancy	
Theoretical	100%
Shared Memory Configuration	
Shared Memory Requested	48 KiB
Shared Memory Executed	48 KiB
Shared Memory Bank Size	4 B



Analysis Details Console Settings Properties

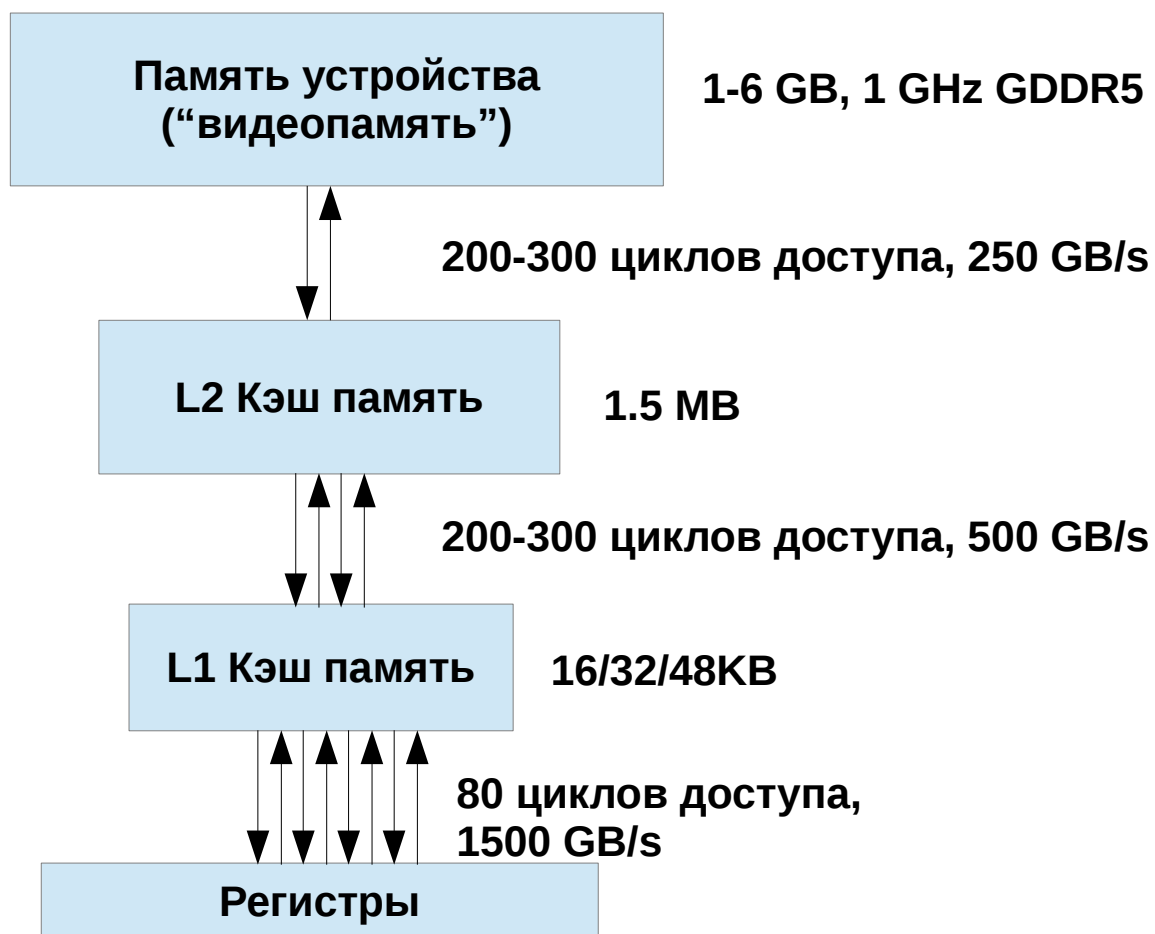
glInit(int*)

Name	Start Time	Duration	Grid Size	Block Size	Regs	Static S
glInit(ir	45.15 ms	86.363 μs	[511,1,1]	[512,1,1]	7	
glInit(ir	45.249 ms	129.647 μs	[512,1,1]	[511,1,1]	7	

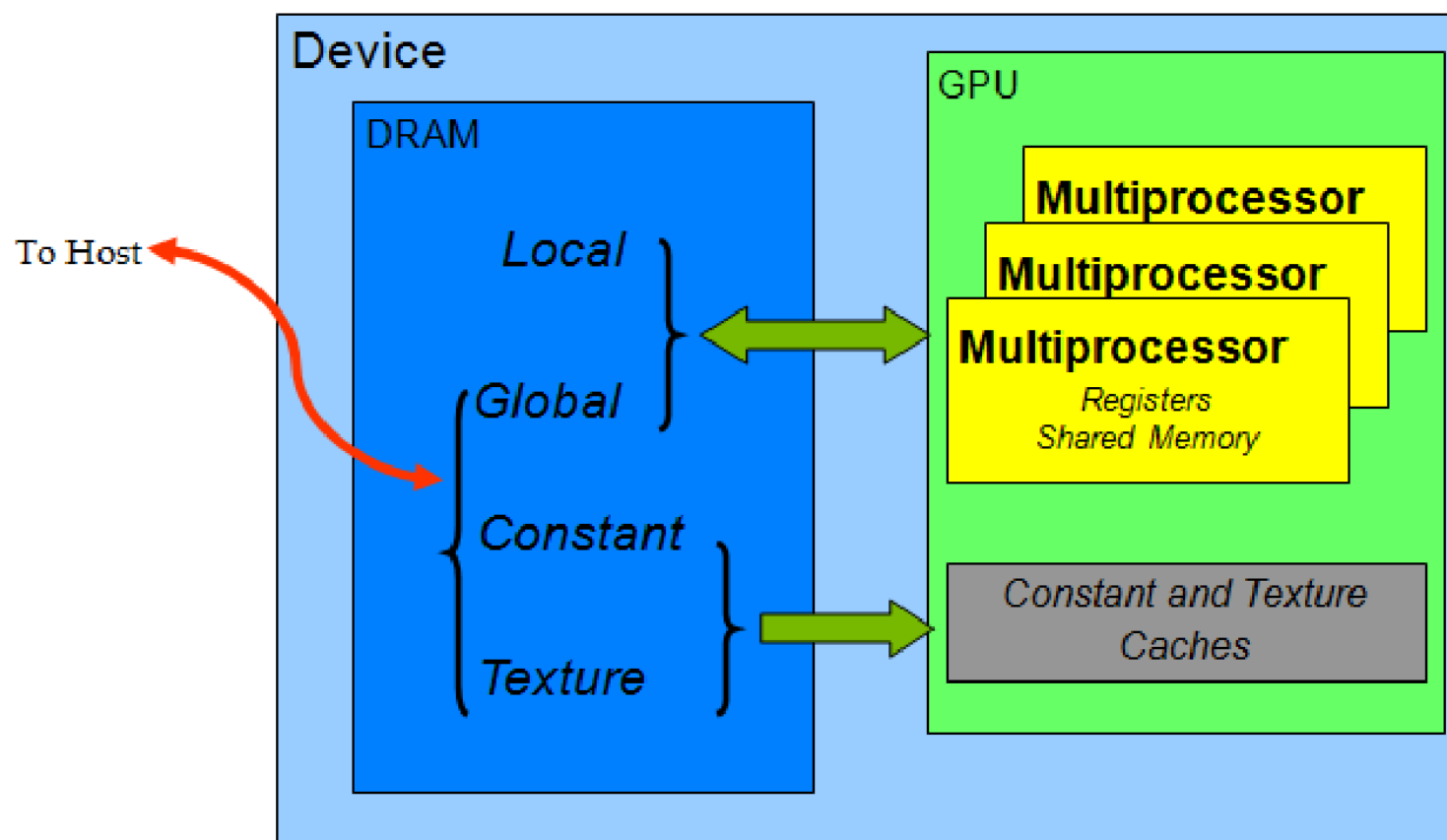
glInit(int*)

- Start: 45.15 ms (45,150,163 ns)
- End: 45.237 ms (45,236,526 ns)
- Duration: 86.363 μs
- Grid Size: [511,1,1]
- Block Size: [512,1,1]
- Registers/Thread: 7
- Shared Memory/Block: 0 B
- Occupancy
 - Theoretical: 100%
- Shared Memory Configuration
 - Shared Memory Requested: 48 KiB
 - Shared Memory Executed: 48 KiB
 - Shared Memory Bank Size: 4 B

Иерархия памяти графического процессора (Fermi, Kepler)

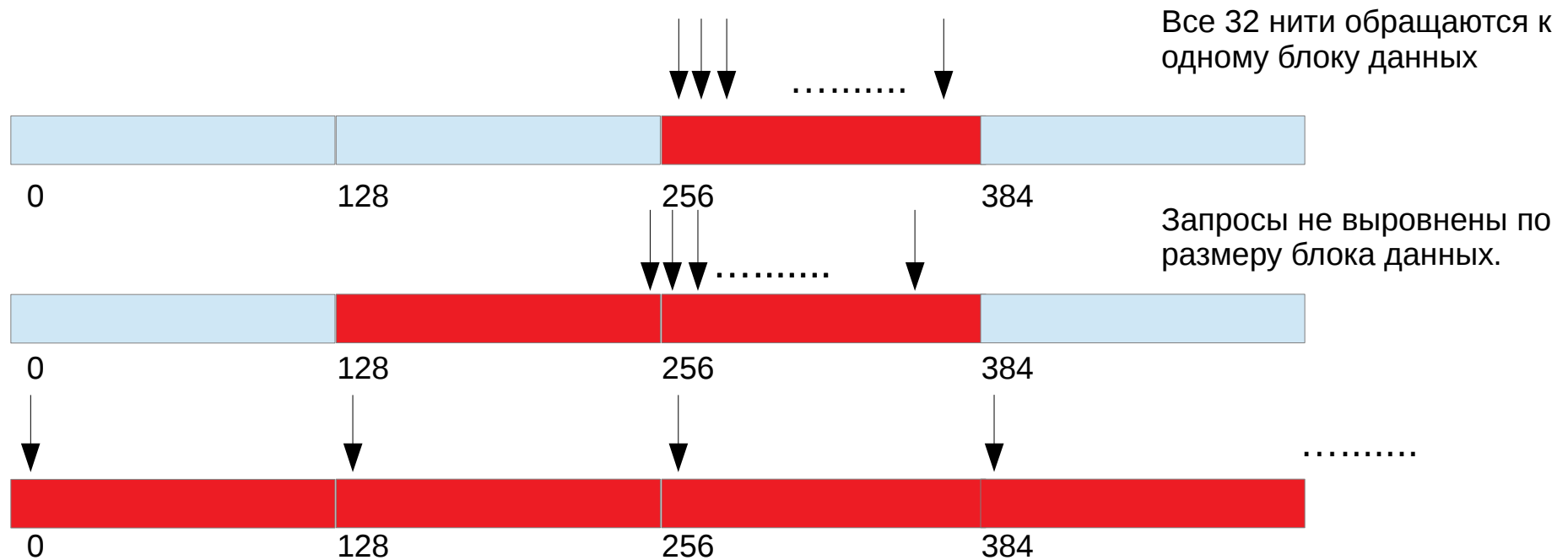


Иерархия памяти устройства CUDA



Объединение запросов к глобальной памяти (coalescing)

Запросы на чтение и запись к глобальной памяти нитями одного варпа (*a warp*) объединяются в транзакции, количество которых равно количеству необходимых для выполнения запросов блоков данных (*cache lines*) L1 кэша размером в 128 байт. **COMPUTE CAPABILITIES 2.* !**



Тестирование параллельных алгоритмов с использованием coalescing'a на GPU с архитектурой Tesla, Fermi и Kepler (задание 2)

Исследовать влияние coalescing'a на примере сложения векторов и инициализации матриц.

```
cudaMalloc((void**)&dVx, (N+offset)*sizeof(float));  
.....  
(dVx+offset)[threadIdx.x]=.....
```

Нарушение выравнивания на границы кратные 128, выполняемого cudaMalloc.

```
A) df[threadIdx.x+blockIdx.x*blockDim.x]=.....  
B) df[blockIdx.x+threadIdx.x*gridDim.x]=.....
```

Нарушение одновременного доступа нитей одного блока CUDA к одному блоку данных.

```
A) df[threadIdx.x+blockIdx.x*blockDim.x]=.....  
B) df[(blockDim.x-threadIdx.x-1)+blockIdx.x*blockDim.x]=.....
```

Нарушение порядка доступа к блоку данных полуварпа (CC 1.*).

Исследовать зависимость времени копирования от величины страйда массива.

```
df1[(threadIdx.x+blockIdx.x*blockDim.x)*stride]=  
df2[(threadIdx.x+blockIdx.x*blockDim.x)*stride];
```

Исследовать влияние выравнивания на операции с массивами структур.

```
struct __align__ (16){int a; float b; float c;};
```