

Лекция 2

СОДЕРЖАНИЕ

- Обработка ошибок
- Профилирование с помощью объектов событий
- Версии CUDA и вычислительные возможности устройств

Макрос для определения ошибки

```
#include <cuda.h>
#include <stdio.h>

#define CUDA_CHECK_RETURN(value) {\
    cudaError_t _m_cudaStat = value;\
    if (_m_cudaStat != cudaSuccess) {\
        fprintf(stderr, "Error %s at line %d in file %s\n",\
            cudaGetErrorString(_m_cudaStat), __LINE__, __FILE__);\
        exit(1);\
    }\
}
```

`const char* cudaGetErrorString (cudaError_t error)` - возвращает сообщение с кодом ошибки `error`.

`__FILE__` и `__LINE__` - predefined макросы препроцессора для определения местоположения в коде программы - имени файла и номера строки.

Диагностика синхронных вызовов

```
__global__ void gTest(float* a){  
    a[threadIdx.x+blockDim.x*blockIdx.x]=(float)  
                                            (threadIdx.x+blockDim.x*blockIdx.x);  
}  
int main(){  
  
    float *da, *ha;  
    int num_of_blocks=10, threads_per_block= 1025;  
    int N=num_of_blocks*threads_per_block;  
  
    ha=(float*)calloc(N, sizeof(float));  
    CUDA_CHECK_RETURN(cudaMalloc((void**)&da,N*sizeof(float)));
```

Диагностика асинхронных вызовов

```
gTest<<<dim3(num_of_blocks), dim3(threads_per_block)>>>(da);
CUDA_CHECK_RETURN(cudaDeviceSynchronize());
CUDA_CHECK_RETURN(cudaGetLastError());
CUDA_CHECK_RETURN(cudaMemcpy(ha,da,N*sizeof(float),
                             cudaMemcpyDeviceToHost));

for(int i=0;i<N;i++)
    printf("%g\n",ha[i]);

free(ha);
cudaFree(da);

return 0;
}
```

Профилирование программ с помощью объектов событий

```
.....  
int main(){  
.....  
  
float elapsedTime;  
cudaEvent_t  start,stop; // встроенный тип данных – структура, для  
                        // фиксации контрольных точек  
cudaEventCreate(&start); // инициализация  
cudaEventCreate(&stop);  // событий
```

Синхронизация по событию

```
cudaEventRecord(start,0); // привязка (регистрация) события start
gTest<<<dim3(num_of_blocks), dim3(threads_per_block)>>>(da);
cudaEventRecord(stop,0); // привязка события stop
cudaEventSynchronize(stop); // синхронизация по событию
//CUDA_CHECK_RETURN(cudaDeviceSynchronize());
CUDA_CHECK_RETURN(cudaGetLastError());
cudaEventElapsedTime(&elapsedTime,start,stop); // вычисление
                                                    // затраченного времени

fprintf(stderr,"gTest took %g\n", elapsedTime);

cudaEventDestroy(start); // освобождение
cudaEventDestroy(stop); // памяти
.....
}
```

Характеристики GPU: GTX 560 Ti(Fermi)

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

.....

Характеристики GPU: GTX 560 Ti(Fermi) (продолжение)

.....

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

.....

Concurrent copy and kernel execution:	Yes with 1 copy engine(s)
Run time limit on kernels:	Yes

.....

Характеристики GPU: GTX 650(Keppler)

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

Характеристики GPU: GTX 650(Keppler) (продолжение)

.....

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

.....

Получение сведений об устройстве.

.../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);
```

.....

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.*	

Совместимость версий CUDA и архитектур устройств.

```
...:~> nvcc --version  
nvcc: NVIDIA (R) Cuda compiler driver  
Copyright (c) 2005-2015 NVIDIA Corporation  
Built on Tue_Aug_11_14:27:32_CDT_2015  
Cuda compilation tools, release 7.5, V7.5.17
```

```
...~>nvcc -arch=sm_20 file_name.cu -o file_name
```