CUDA ON ARM PLATFORM—错误检测与事件

NVIDIA企业级开发者社区 何琨

AGENDA

错误检测与事件

- CUDA进阶之路
- CUDA应用程序运行时的错误检测
- CUDA中的事件
- 利用事件进行计时

CUDA进阶之路

指令

存储

硬件

工具库

CUDA的运行时检测函数

```
__host____device__const char*     cudaGetErrorName ( cudaError_t error )
Returns the string representation of an error code enum name.

__host____device__const char*     cudaGetErrorString ( cudaError_t error )
Returns the description string for an error code.

__host____device__cudaError_t     cudaGetLastError ( void )
Returns the last error from a runtime call.

__host____device__cudaError_t     cudaPeekAtLastError ( void )
Returns the last error from a runtime call.
```

CUDA的运行时检测函数

```
__global__ void hello( )
{
    printf("blockIdx.x=%d/%d blocks, threadIdx.x=%d/%d threads\n",
                          blockIdx.x,  gridDim.x,
                          threadIdx.x, blockDim.x);
}

int main()
{
   hello<<< 1, 1025 >>>( );     // Error: #threads >= 1024 !!!

   printf("I am the CPU: Hello World ! \n");

   cudaDeviceSynchronize();

   return 0;
}
```

- **CUDA execution error** are **often *not* visible** when you run the program !!!

```
   /home/cs355001/demo/CUDA/1-intro/hello-error

  Output:

     I am the CPU: Hello World !
     (No error message from the GPU execution !!!)
```

CUDA的运行时检测函数

```
__global__ void hello( )
{
    printf("blockIdx.x=%d/%d blocks, threadIdx.x=%d/%d threads\n",
                        blockIdx.x,  gridDim.x,
                        threadIdx.x, blockDim.x);
}

int main()
{
   hello<<< 1, 1025 >>>( );    // Error !!!

   cudaError_t err = cudaGetLastError();        // Get error code

   if ( err != cudaSuccess )
   {
      printf("CUDA Error: %s\n", cudaGetErrorString(err));
      exit(-1);
   }

   printf("I am the CPU: Hello World ! \n");

   cudaDeviceSynchronize();

   return 0;
}
```

```
/home/cs355001/demo/CUDA/1-intro/hello-error2

Output:

   CUDA Error: invalid configuration argument
```

CUDA的运行时检测函数

```
#pragma once
#include <stdio.h>

#define CHECK(call)                                                \
do                                                                 \
{                                                                  \
    const cudaError_t error_code = call;                           \
    if (error_code != cudaSuccess)                                 \
    {                                                              \
        printf("CUDA Error:\n");                                   \
        printf("    File:       %s\n", __FILE__);                  \
        printf("    Line:       %d\n", __LINE__);                  \
        printf("    Error code: %d\n", error_code);                \
        printf("    Error text: %s\n",                             \
            cudaGetErrorString(error_code));                       \
        exit(1);                                                   \
    }                                                              \
} while (0)
```

CUDA的运行时检测函数

```
#pragma once
#include <stdio.h>

#define CHECK(call)                                     \
do                                                      \
{                                                       \
    const cudaError_t error_code = call;                \
    if (error_code != cudaSuccess)                      \
    {                                                   \
        printf("CUDA Error:\n");                        \
        printf("    File:       %s\n", __FILE__);       \
        printf("    Line:       %d\n", __LINE__);       \
        printf("    Error code: %d\n", error_code);     \
        printf("    Error text: %s\n",                  \
            cudaGetErrorString(error_code));            \
        exit(1);                                        \
    }                                                   \
} while (0)
```

```
CHECK(cudaMemcpy(d_b, h_b, sizeof(int)*n*k, cudaMemcpyHostToDevice));
```

CUDA的事件(event)

之前的课程中，我们已经讨论过，如何利用CUDA加速矩阵相乘的例子。
那么，我们如何判断加速比，如何计时呢？

CPU TIMER?

CUDA的事件(event)

CUDA event 本质是一个GPU时间戳，这个时间戳是在用户指定的时间点上记录的。由于GPU本身支持记录时间戳，因此就避免了当使用CPU定时器来统计GPU执行时间时可能遇到的诸多问题。

CUDA的事件(event)

__host__cudaError_t cudaEventCreate (cudaEvent_t* event)
Creates an event object.

__host____device__cudaError_t cudaEventCreateWithFlags (cudaEvent_t* event, unsigned int flags)
Creates an event object with the specified flags.

__host____device__cudaError_t cudaEventDestroy (cudaEvent_t event)
Destroys an event object.

__host__cudaError_t cudaEventElapsedTime (float* ms, cudaEvent_t start, cudaEvent_t end)
Computes the elapsed time between events.

__host__cudaError_t cudaEventQuery (cudaEvent_t event)
Queries an event's status.

__host____device__cudaError_t cudaEventRecord (cudaEvent_t event, cudaStream_t stream = 0)
Records an event.

__host__cudaError_t cudaEventRecordWithFlags (cudaEvent_t event, cudaStream_t stream = 0, unsigned int flags = 0)
Records an event.

__host__cudaError_t cudaEventSynchronize (cudaEvent_t event)
Waits for an event to complete.

CUDA的事件(event)

声明:
cudaEvent_t event;

创建:
cudaError_t cudaEventCreate(cudaEvent_t* event);

销毁:
cudaError_t cudaEventDestroy(cudaEvent_t event);

添加事件到当前执行流:
cudaError_t cudaEventRecord(cudaEvent_t event, cudaStream_t stream = 0);

等待事件完成，设立flag:
cudaError_t cudaEventSynchronize(cudaEvent_t event);//阻塞
cudaError_t cudaEventQuery(cudaEvent_t event);//非阻塞

当然，我们也可以用它来记录执行的事件:
cudaError_t cudaEventElapsedTime(float* ms, cudaEvent_t start, cudaEvent_t stop);

cudaEventRecord()视为一条记录当前时间的语句，并且把这条语句放入GPU的未完成队列中。因为直到GPU执行完了在调用cudaEventRecord()之前的所有语句时，事件才会被记录下来。且仅当GPU完成了之前的工作并且记录了stop事件后，才能安全地读取stop时间值。

更多资源:

https://developer.nvidia-china.com

何琨-Ken

北京 密云

扫一扫上面的二维码图案，加我微信

https://www.nvidia.cn/developer/community-training/


NVIDIA