

Лекция 6

Потоки CUDA (*CUDA Streams*)

- pinned-память - закрепленные страницы памяти;
- CUDA Streams – очереди команд;
- Мульти GPU

Возможность одновременного копирования

Device 0: "GeForce GTX 560 Ti"

CUDA Driver Version / Runtime Version 7.5 / 7.5

CUDA Capability Major/Minor version number: 2.1

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



Потоки CUDA и разрешение зависимостей при распараллеливании копирования и выполнения

(I)

Очередь копирования Очередь выполнения

stream0, copy a	
stream0, copy b	
блокировка	kernel0
stream0, copy c	
stream1, copy a	
stream1, copy b	
блокировка	kernel1
stream1, copy c	

(II)

Очередь копирования Очередь выполнения

stream0, copy a	
stream0, copy b	
stream1, copy a	kernel0
stream1, copy b	
stream0, copy c	kernel1
stream1, copy c	

```

#define N (1024*1024)
#define FULL_DATA_SIZE (N*20)

__global__ void kernel(int* a, int* b, int* c){
    int idx=threadIdx.x+blockIdx.x*blockDim.x;
    if(idx<N){
        int idx1=(idx+1)%256;
        int idx2=(idx+2)%256;
        float as=(a[idx]+a[idx1]+a[idx2])/3.0f;
        float bs=(b[idx]+b[idx1]+b[idx2])/3.0f;
        c[idx]=(as+bs)/2;
    }
}

int main(){
    cudaDeviceProp prop;
    int whichDevice;

    cudaGetDevice(&whichDevice);
    cudaGetDeviceProperties(&prop, whichDevice);
    if(!prop.deviceOverlap){
        printf("Device does not support overlapping\n");
        return 0;
    }

    cudaEvent_t start, stop;

```

.....

Sanders J., Kandrot E.
*CUDA by Example, an introduction to
 general-purpose GPU programming*,
 Addison-Wesley, 2013.

Выделение закрепленной (*paged-locked*) памяти

```
cudaHostAlloc( (void**)&host_a, FULL_DATA_SIZE*sizeof(int),  
              cudaHostAllocDefault);
```

```
cudaHostAlloc( (void**)&host_b, FULL_DATA_SIZE*sizeof(int),  
              cudaHostAllocDefault);
```

```
cudaHostAlloc( (void**)&host_c, FULL_DATA_SIZE*sizeof(int),  
              cudaHostAllocDefault);
```

Константа ***cudaHostAllocDefault*** означает эквивалентность функций

```
__host__ cudaError_t cudaHostAlloc ( void** pHost, size_t size, unsigned int flags )
```

и

```
__host__ cudaError_t cudaMallocHost ( void** ptr, size_t size )
```

```
.....
cudaStream_t stream;
cudaStreamCreate(&stream );

cudaEventRecord(start,0 );
for(int i=0; i<FULL_DATA_SIZE; i+=N){

    cudaMemcpyAsync(dev_a, host_a+i, N*sizeof(int), cudaMemcpyHostToDevice, stream );
    cudaMemcpyAsync(dev_b, host_b+i, N*sizeof(int), cudaMemcpyHostToDevice, stream );

    kernel<<<N/256, 256, 0, stream>>>(dev_a, dev_b, dev_c);

    cudaMemcpyAsync(host_c+i, dev_c, N*sizeof(int), cudaMemcpyDeviceToHost, stream );
}

cudaStreamSynchronize( stream );

cudaEventRecord(stop,0);
cudaEventSynchronize(stop);
.....
```

(I)

```
.....
cudaEventRecord( start, 0 );
for(int i=0; i<FULL_DATA_SIZE; i+=N*2){
////////// П Е Р В Ы Й   П О Т О К   //////////
    cudaMemcpyAsync(dev_a0, host_a+i, N*sizeof(int), cudaMemcpyHostToDevice, stream0 );
    cudaMemcpyAsync(dev_b0, host_b+i, N*sizeof(int), cudaMemcpyHostToDevice, stream0 );

    kernel<<<N/256, 256, 0, stream0>>>(dev_a0, dev_b0, dev_c0);

    cudaMemcpyAsync(host_c+i, dev_c0, N*sizeof(int), cudaMemcpyDeviceToHost, stream0 );

////////// В Т О Р О Й   П О Т О К   //////////
    cudaMemcpyAsync(dev_a1, host_a+i+N, N*sizeof(int), cudaMemcpyHostToDevice, stream1);
    cudaMemcpyAsync(dev_b1, host_b+i+N, N*sizeof(int), cudaMemcpyHostToDevice, stream1);

    kernel<<<N/256, 256, 0, stream1>>>(dev_a1, dev_b1, dev_c1);

    cudaMemcpyAsync(host_c+i+N, dev_c1, N*sizeof(int), cudaMemcpyDeviceToHost, stream1);
}

cudaStreamSynchronize( stream0 );
cudaStreamSynchronize( stream1 );

cudaEventRecord(stop,0);
cudaEventSynchronize(stop);
.....
```

(II)

```
.....  
cudaEventRecord(start,0)  
for(int i=0; i<FULL_DATA_SIZE; i+=N*2){  
////////// П Е Р Е М Е Ж А Е М Ы Е    П О Т О К И    //////////  
  cudaMemcpyAsync(dev_a0, host_a+i, N*sizeof(int),cudaMemcpyHostToDevice, stream0 );  
  cudaMemcpyAsync(dev_a1, host_a+i+N, N*sizeof(int),cudaMemcpyHostToDevice, stream1);  
  
  cudaMemcpyAsync(dev_b0, host_b+i, N*sizeof(int), cudaMemcpyHostToDevice, stream0 );  
  cudaMemcpyAsync(dev_b1, host_b+i+N, N*sizeof(int), cudaMemcpyHostToDevice, stream1);  
  
  kernel<<<N/256, 256, 0, stream0>>>(dev_a0, dev_b0, dev_c0);  
  kernel<<<N/256, 256, 0, stream1>>>(dev_a1, dev_b1, dev_c1);  
  
  cudaMemcpyAsync(host_c+i, dev_c0, N*sizeof(int),cudaMemcpyDeviceToHost, stream0 );  
  cudaMemcpyAsync(host_c+i+N, dev_c1, N*sizeof(int), cudaMemcpyDeviceToHost, stream1);  
}  
  
cudaStreamSynchronize( stream0 );  
cudaStreamSynchronize( stream1 );  
  
cudaEventRecord(stop,0 );  
cudaEventSynchronize(stop);  
.....
```


Device 0: "GeForce GTX 560 Ti"
CUDA Driver Version / Runtime Version 7.5 / 7.5
CUDA Capability Major/Minor version number: 2.1
Total amount of global memory: 2047 MBytes (2145927168 bytes)
(8) Multiprocessors, (48) CUDA Cores/MP: 384 CUDA Cores
GPU Max Clock rate: 1645 MHz (1.64 GHz)
Memory Clock rate: 2004 Mhz

.....
Concurrent copy and kernel execution: Yes with 1 copy engine(s)
Run time limit on kernels: Yes
Integrated GPU sharing Host Memory: No
.....

```
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/CUDA STREAMS> ./1  
Elapsed time: 44.1 ms  
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/CUDA STREAMS> ./2  
Elapsed time: 44.4 ms  
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/CUDA STREAMS> ./3  
Elapsed time: 41.6 ms
```

Device 0: "GeForce GTX 650"

CUDA Driver Version / Runtime Version 6.5 / 6.5

CUDA Capability Major/Minor version number: 3.0

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

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

GPU Clock rate: 1110 MHz (1.11 GHz)

Memory Clock rate: 2500 Mhz

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

Run time limit on kernels: Yes

Integrated GPU sharing Host Memory: No

malkov@dew:~/WORKSHOP/PROJECTS/CUDA-EXERCISE/CUDA_STREAMS> ./1

Elapsed time: 34.1 ms

malkov@dew:~/WORKSHOP/PROJECTS/CUDA-EXERCISE/CUDA_STREAMS> ./2

Elapsed time: 34.1 ms

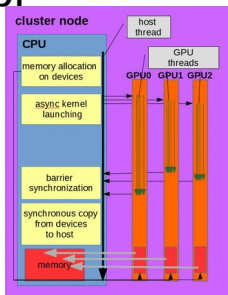
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-EXERCISE/CUDA_STREAMS> ./3

Elapsed time: 23.7 ms

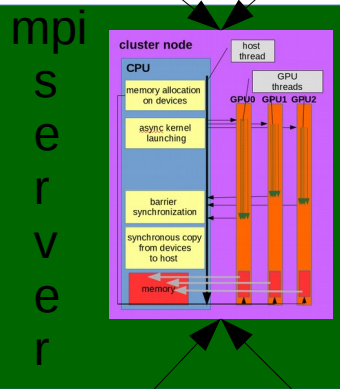
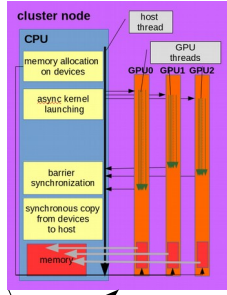
malkov@dew:~/WORKSHOP/PROJECTS/CUDA-EXERCISE/CUDA_STREAMS>

select=any:ncpus=1:
mpiprocs=1:gpus=1..3

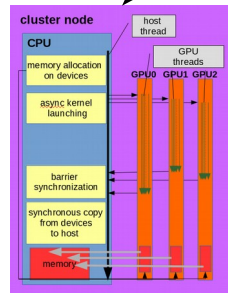
mpi
c
l
i
e
n
t



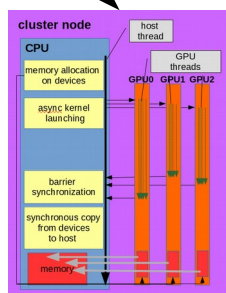
mpi
c
l
i
e
n
t



mpi
c
l
i
e
n
t



mpi
c
l
i
e
n
t



cluster node

CPU

memory allocation
on devices

async kernels
launching

barrier
synchronization

synchronous copy
from devices
to host

memory

host
thread

GPU
threads

GPU0 GPU1 GPU2

Выделение памяти

Выделение памяти на хосте:

```
float** Df_device=(float**)calloc(NG, sizeof(REAL*));  
float** St_device=(float**)calloc(NG, sizeof(REAL*));
```

Выделение памяти на устройствах:

```
for(int idev=0;idev< NG; idev++){  
    cudaSetDevice(assigned_devices[idev]);  
    cudaMalloc((void **) &Df_device[idev], size_of_Df/NG);  
    cudaMalloc((void **) &St_device[idev], size_of_Df/NG);  
}
```

NG – количество GPU.

idev – номер GPU.

assigned_devices[idev] – идентификатор GPU.

Df_device[idev], *St_device[idev]* – порции массивов, предназначенные для GPU с номером *idev*.

Асинхронный запуск ядер и барьерная синхронизация.

```
cudaEvent_t *mdEventStart=(cudaEvent_t*)calloc(NG, sizeof(cudaEvent_t));  
cudaEvent_t *mdEventStop=(cudaEvent_t*)calloc(NG, sizeof(cudaEvent_t));
```

```
for(int idev=0;idev< NG; idev++){  
    cudaSetDevice(assigned_devices[idev]);  
    cudaEventCreate(&mdEventStart[idev]);  
    cudaEventCreate(&mdEventStop[idev]);  
}
```

```
for(int idev=0;idev< NG; idev++){  
    cudaSetDevice(assigned_devices[idev]);  
    cudaEventRecord( mdEventStart[idev], 0 );  
    gStossCalc<<<dim3(nvx, nvy,nvz), dim3(SPACE_CELL_OFFSET)>>>  
        (Df_device[idev], St_device[idev], N/NG, l_offset);  
    cudaEventRecord( mdEventStop[idev], 0);  
}
```

```
for(int idev=0;idev< NG; idev++){  
    cudaSetDevice(assigned_devices[idev]);  
    cudaEventSynchronize(mdEventStop[idev]);  
}
```

Синхронное копирование с устройств на хост

```
for(int idev=0;idev< NG; idev++){  
    cudaSetDevice(assigned_devices[idev]);  
    cudaMemcpy(St+idev*size_of_DF/NG, St_device[idev],  
        size_of_Df/NG, cudaMemcpyDeviceToHost);  
}
```

(1 GPU)

NG=1 gCalculateParams Elapsed time: 1.55024

gClearStoss gClearStoss Elapsed time: 0.213664

gTransposeDf Elapsed time: 5.2569

gStossCalc Elapsed time: **246532**

gTransposeDfInverse Elapsed time: 7.51043

gClearStoss gClearStoss Elapsed time: 0.217504

gTransposeDf Elapsed time: 5.25312

gStossCalc Elapsed time: **246536**

gTransposeDfInverse Elapsed time: 7.53414

gClearStoss gClearStoss Elapsed time: 0.217088

gTransposeDf Elapsed time: 5.25322

gStossCalc Elapsed time: **246535**

gTransposeDfInverse Elapsed time: 7.51981

gClearStoss gClearStoss Elapsed time: 0.216192

gTransposeDf Elapsed time: 5.25834

gStossCalc Elapsed time: **246533**

gTransposeDfInverse Elapsed time: 7.46163

gClearStoss gClearStoss Elapsed time: 0.216544

gTransposeDf Elapsed time: 5.25523

gStossCalc Elapsed time: **246538**

gTransposeDfInverse Elapsed time: 7.49238

gClearStoss gClearStoss Elapsed time: 0.215904

gTransposeDf Elapsed time: 5.24982

gStossCalc Elapsed time: **246536**

gTransposeDfInverse Elapsed time: 7.52288

real 24m47.123s

user 24m45.149s

sys 0m0.944s

25,165,824
grid nodes
in phase space

every loop handles
4,194,304 grid nodes
in phase space

(2 GPU)

NG=2 gCalculateParams Elapsed time: 1.53939
NG=2 gCalculateParams Elapsed time: 1.49997
gClearStoss gClearStoss Elapsed time: 0.211264
gClearStoss gClearStoss Elapsed time: 0.19552
gTransposeDf Elapsed time: 5.2217
gTransposeDf Elapsed time: 5.23258
gStossCalc Elapsed time: **248925**
gStossCalc Elapsed time: **248926**
gTransposeDfInverse Elapsed time: 7.48387
gTransposeDfInverse Elapsed time: 7.48378
gClearStoss gClearStoss Elapsed time: 0.200736
gClearStoss gClearStoss Elapsed time: 0.209472
gTransposeDf Elapsed time: 5.2145
gTransposeDf Elapsed time: 5.22477
gStossCalc Elapsed time: **248722**
gStossCalc Elapsed time: **248723**
gTransposeDfInverse Elapsed time: 7.51037
gTransposeDfInverse Elapsed time: 7.51152
gClearStoss gClearStoss Elapsed time: 0.199392
gClearStoss gClearStoss Elapsed time: 0.208352
gTransposeDf Elapsed time: 5.21789
gTransposeDf Elapsed time: 5.22787
gStossCalc Elapsed time: **248926**
gStossCalc Elapsed time: **248926**
gTransposeDfInverse Elapsed time: 7.51379
gTransposeDfInverse Elapsed time: 7.51504
real 12m35.874s
user 12m33.123s
sys 0m1.628s

25,165,824
grid nodes
in phase space

every loop handles
4,194,304 grid nodes
in phase space

two-times speedup
vs. 1 GPU

(3 GPU)

NG=3 gCalculateParams Elapsed time: 1.47718
NG=3 gCalculateParams Elapsed time: 1.53101
NG=3 gCalculateParams Elapsed time: 1.5304
gClearStoss gClearStoss Elapsed time: 0.140192
gClearStoss gClearStoss Elapsed time: 0.155264
gClearStoss gClearStoss Elapsed time: 0.124384
gTransposeDf Elapsed time: 4.94176
gTransposeDf Elapsed time: 5.1377
gTransposeDf Elapsed time: 5.14787
gStossCalc Elapsed time: **164576**
gStossCalc Elapsed time: **164589**
gStossCalc Elapsed time: **164590**
gTransposeDfInverse Elapsed time: 7.12509
gTransposeDfInverse Elapsed time: 7.43171
gTransposeDfInverse Elapsed time: 7.44042
gClearStoss gClearStoss Elapsed time: 0.128416
gClearStoss gClearStoss Elapsed time: 0.13792
gClearStoss gClearStoss Elapsed time: 0.147904
gTransposeDf Elapsed time: 4.93661
gTransposeDf Elapsed time: 5.13046
gTransposeDf Elapsed time: 5.13981
gStossCalc Elapsed time: **164575**
gStossCalc Elapsed time: **164590**
gStossCalc Elapsed time: **164591**
gTransposeDfInverse Elapsed time: 7.11542
gTransposeDfInverse Elapsed time: 7.44173
gTransposeDfInverse Elapsed time: 7.4505
real 5m38.815s
user 5m36.017s
sys 0m1.800s

25,165,824
grid nodes
in phase space

every loop handles
4,194,304 grid nodes
in phase space

superlinear speedup:
aprox 5-times (not 3 !) speedup
vs. 1 GPU