Лекция 7

Библиотека ***thrust***

- Обобщенное программирование: *контейнеры, обобщенные алгоритмы, итераторы.*
- Контейнеры *host_vector* и *device_vector.*
- Алгоритмы *thrust.*
- Преобразование указателей и комбинированный код.
- Алгоритм *transform* и функторы.
- Кортежи и *zip*-итератор.

Контейнеры *host_vector* и *device_vector*

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
//#include <thrust/copy.h>

int main(void){
  thrust::host_vector<int> h(1 << 8);
  thrust::generate(h.begin(), h.end(), rand);
  thrust::device_vector<int> d=h;
  thrust::sort(d.begin(), d.end());

  //thrust::copy(d.begin(), d.end(), h.begin());
  h=d;

  for(int i=0;i<1<<8;i++)
    printf("%i\t%d\n",i, h[i]);

  return 0;
}
```

```
~> nvcc 0.cu -o 0
~> nvprof ./0 > tt
```

```
 1.19%  1.7920us         1  1.7920us  1.7920us  1.7920us  [CUDA memcpy DtoD]
 0.96%  1.4400us         1  1.4400us  1.4400us  1.4400us  [CUDA memcpy DtoH]
 0.51%    768ns         1    768ns    768ns    768ns       [CUDA memcpy HtoD]

==7688== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 98.77%  73.642ms         2  36.821ms  103.81us  73.539ms  cudaMalloc
  0.49%  366.17us        83  4.4110us     210ns  158.99us  cuDeviceGetAttribute
  0.24%  175.81us        21  8.3710us  7.0410us  21.062us  cudaLaunch
  0.20%  151.87us         2  75.935us  70.339us  81.532us  cudaFree
  0.07%  50.401us         1  50.401us  50.401us  50.401us  cuDeviceTotalMem
  0.06%  48.446us         3  16.148us  14.660us  18.604us  cudaMemcpyAsync
  0.05%  38.601us         1  38.601us  38.601us  38.601us  cuDeviceGetName
  0.05%  35.354us       140     252ns     197ns  3.3320us  cudaSetupArgument
  0.04%  28.573us        10  2.8570us  2.3560us  5.0370us  cudaFuncGetAttributes
  0.01%  6.3730us        21     303ns     276ns     524ns  cudaPeekAtLastError
  0.01%  6.0650us        21     288ns     227ns  1.1170us  cudaConfigureCall
  0.00%  3.4010us         6     566ns     345ns  1.1120us  cudaDeviceGetAttribute
  0.00%  2.4650us         2  1.2320us     942ns  1.5230us
                                                           cudaDeviceGetSharedMemConfig
  0.00%  1.7510us         2     875ns     377ns  1.3740us  cuDeviceGetCount
  0.00%  1.5600us         2     780ns     617ns     943ns  cudaGetDevice
```

Преобразование указателей и комбинированный код

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/fill.h>
#include <thrust/copy.h>
#include <cstdio>

__global__ void gTest(float* d){
  int idx=threadIdx.x+blockDim.x*blockIdx.x;
  d[idx]+=(float)idx;
}
```

```
int main(void){
  float *raw_ptr;
#ifdef H2D
  thrust::host_vector<float> h(1 << 8);
  thrust::fill(h.begin(), h.end(), 3.1415f);
  thrust::device_vector<float> d = h;
  fprintf(stderr, "Host to device\n");
#else
  thrust::device_vector<float> d(1<<8);
  thrust::fill(d.begin(), d.end(), 3.1415f);
  thrust::host_vector<float> h;
  fprintf(stderr, "Just on device\n");
#endif

  raw_ptr = thrust::raw_pointer_cast(&d[0]);//d.data());

  gTest<<<4,64>>>(raw_ptr);
  cudaDeviceSynchronize();

  h=d;                  //thrust::copy(d.begin(), d.end(), h.begin());
  for(int i=0;i<(1<<8);i++)
    printf("%g\n",h[i]);

  cudaFree(raw_ptr);
  return 0;
}
```

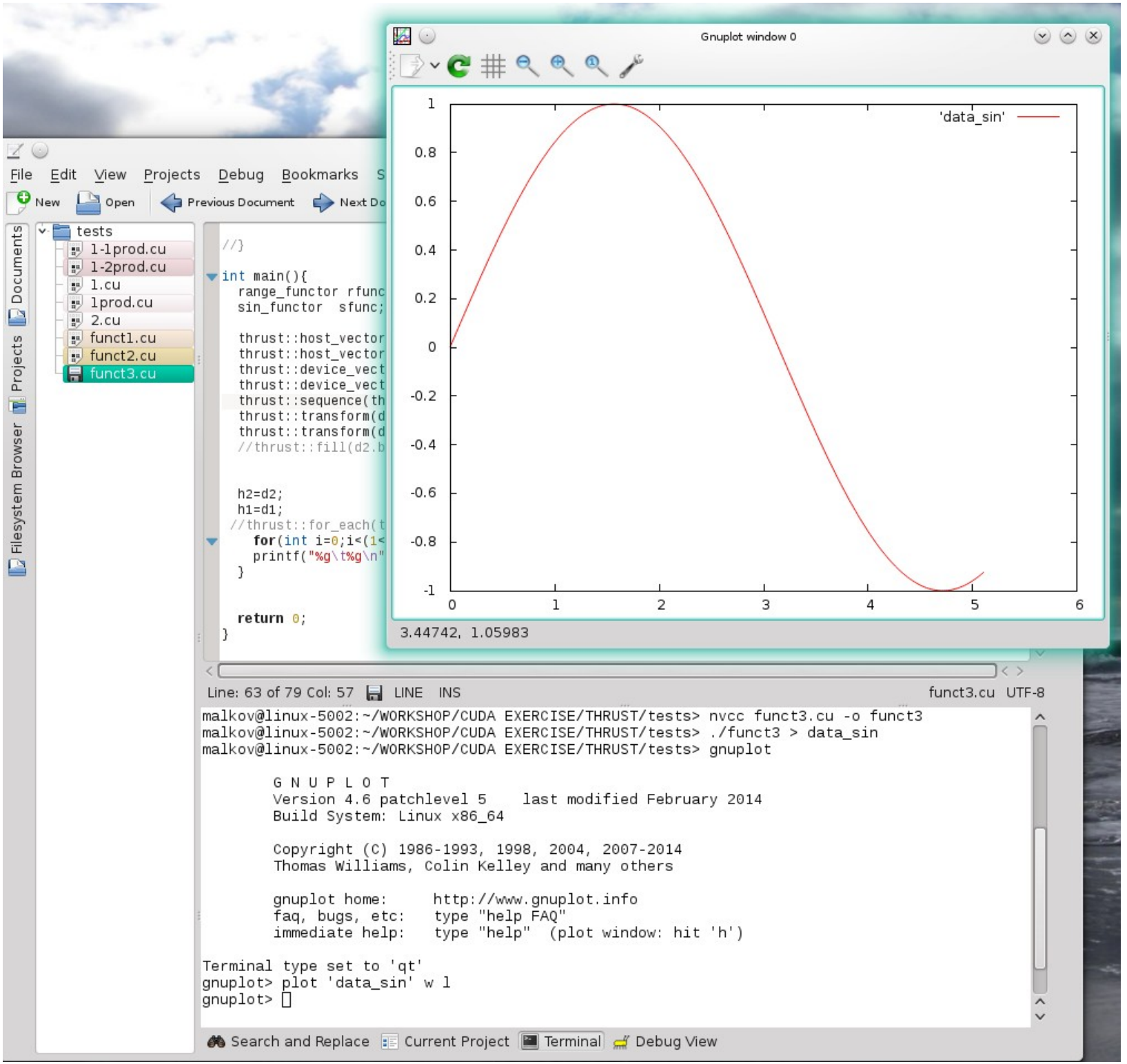
```
~> nvcc -D H2D 1n.cu -o 1n
~> 1n > tt
```

```
~> nvcc  1n.cu -o 1n
~> 1n > tt
```

Алгоритм *transform* и функторы

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/transform.h>
#include <thrust/sequence.h>
#include <cstdio>
#include <cmath>

struct range_functor{
  float h;
  range_functor(float _h):h(_h){}
  __host__ __device__
  float operator()(float x){
    return h*x;
  }
};

struct sin_functor
{
  __device__
  float operator()(float x){
    return __sinf(x);
  }
};
```

```
int main(){
  range_functor R(0.02);
  sin_functor  Sin;

  fprintf(stderr, "%g\n", R(30.0f));
//fprintf(stderr, "%g\n", Sin(3141592.0f/6.0f));

  thrust::host_vector<float> h1(1 << 8);
  thrust::host_vector<float> h2(1 << 8);
  thrust::device_vector<float> d1(1 << 8);
  thrust::device_vector<float> d2(1 << 8);
  thrust::sequence(thrust::device,d1.begin(), d1.end());
  thrust::transform(d1.begin(), d1.end(), d1.begin(), R);
  thrust::transform(d1.begin(), d1.end(), d2.begin(), Sin);

  h2=d2;
  h1=d1;

  for(int i=0;i<(1<<8);i++){
    printf("%g\t%g\n",h1[i], h2[i]);
  }

  return 0;
}
```


Gnuplot window 0
'data_sin'
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
3.44742, 1.05983
File Edit View Projects Debug Bookmarks
New Open Previous Document Next Do
Documents
Projects
Filesystem Browser
tests
1-1prod.cu
1-2prod.cu
1.cu
1prod.cu
2.cu
funct1.cu
funct2.cu
funct3.cu
//}
int main(){
range_functor rfunc
sin_functor sfunc;
thrust::host_vector
thrust::host_vector
thrust::device_vect
thrust::device_vect
thrust::sequence(th
thrust::transform(d
thrust::transform(d
//thrust::fill(d2.b
h2=d2;
h1=d1;
//thrust::for_each(t
for(int i=0;i<(1<
printf("%g\t%g\n"
}
return 0;
}
Line: 63 of 79 Col: 57 LINE INS
funct3.cu UTF-8
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> nvcc funct3.cu -o funct3
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> ./funct3 > data_sin
malkov@linux-5002:~/WORKSHOP/CUDA EXERCISE/THRUST/tests> gnuplot
G N U P L O T
Version 4.6 patchlevel 5 last modified February 2014
Build System: Linux x86_64
Copyright (C) 1986-1993, 1998, 2004, 2007-2014
Thomas Williams, Colin Kelley and many others
gnuplot home: http://www.gnuplot.info
faq, bugs, etc: type "help FAQ"
immediate help: type "help" (plot window: hit 'h')
Terminal type set to 'qt'
gnuplot> plot 'data_sin' w l
gnuplot>
Search and Replace
Current Project
Terminal
Debug View

Реализация *SAXPY*

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/transform.h>

struct saxpy_functor
{
  const float a;
  saxpy_functor(float _a) : a(_a) {}
  __host__ __device__
  float operator()(float x, float y){
    return a*x+y;
  }
};

void saxpy(float a, thrust::device_vector<float>& x,
                    thrust::device_vector<float>& y){
  saxpy_functor  func(a);

  thrust::transform(x.begin(), x.end(), y.begin(), y.begin(), func);
}
```

```
#include <thrust/fill.h>
#include <thrust/sequence.h>

int main(){
  thrust::host_vector<float> h1(1 << 24);
  thrust::host_vector<float> h2(1 << 24);
  thrust::sequence(h1.begin(), h1.end());
  thrust::fill(h2.begin(), h2.end(), 0.87);

  thrust::device_vector<float> d1 = h1;
  thrust::device_vector<float> d2 = h2;

  saxpy(3.0, d1, d2);

  h2=d2;
  h1=d1;

  for(int i=0;i<(1<<8);i++){
    printf("%d\t%g\t%g\n",i, h1[i], h2[i]);
  }

  return 0;
}
```

```
~> ./3n
0    0    0.87
1    1    3.87
2    2    6.87
3    3    9.87
4    4    12.87
5    5    15.87
6    6    18.87
7    7    21.87
……..................................
250 250 750.87
251 251 753.87
252 252 756.87
253 253 759.87
254 254 762.87
255 255 765.87
```

Кортежи

```
#include <thrust/tuple.h>
#include <cstdio>

int main(){
  thrust::tuple<int, float, const char *> test_tuple(23, 4.5, "thrust");
  printf("%d\t%g\t%s\n", thrust::get<0>(test_tuple),
                         thrust::get<1>(test_tuple),
                         thrust::get<2>(test_tuple));
  return 0;
}
```

```
~> ./tuple1
23      4.5     thrust
```

zip-итераторы

```
#include <thrust/tuple.h>
#include <thrust/device_vector.h>
#include <thrust/host_vector.h>
#include <thrust/transform.h>
#include <thrust/fill.h>
#include <thrust/iterator/zip_iterator.h>

#define N 32

struct rotate_tuple{
  __host__ __device__
    thrust::tuple<float,float,float>   operator()(thrust::tuple<float&,float&,float&>& t){
    float x = thrust::get<0>(t);
    float y = thrust::get<1>(t);
    float z = thrust::get<2>(t);

    float rx=0.36*x+0.48*y-0.80*z;
    float ry=-0.80f*x+0.60*y+0.00f*z;
    float rz=0.48f*x+0.64f*y+0.60f*z;

    return thrust::make_tuple(rx,ry,rz);
  }
};
```

```
int main(){
  thrust::device_vector<float> x(N), y(N), z(N);

  thrust::fill(x.begin(), x.end(), 2.0);
  thrust::fill(y.begin(), y.end(), 3.0);
  thrust::fill(z.begin(), z.end(), 5.0);

  thrust::transform(
                        thrust::make_zip_iterator(
                                thrust::make_tuple(x.begin(), y.begin(), z.begin() )),
                       thrust::make_zip_iterator(
                                thrust::make_tuple(x.end(),   y.end(),   z.end()   )),
                        thrust::make_zip_iterator(
                                thrust::make_tuple(x.begin(), y.begin(), z.begin() )),
                        rotate_tuple() );

  thrust::host_vector<float> hx(N), hy(N),hz(N);
  hx=x;  hy=y; hz=z;
      for(int i=0;i<N;i++)
        printf("%g\t%g\t%g\n",hx[i], hy[i], hz[i]);

  return 0;
}
```