Лекция 9

Python+CUDA

- CUDA Python:
  https://developer.nvidia.com/how-to-cuda-python
- [PyCUDA: https://mathema.tician.de/software/pycuda/]

Глоссарий

Anaconda — (Free Open Source Software) дистрибутив python для научных вычислений.

NumPy — библиотека python для поддержки численных расчетов.

Numba — оптимизирующий компилятор python для CPU и GPU.

Matplotlib — графическая библиотека python.

PyLab — процедурный интерфейс Matplotlib.

Module — файл .py

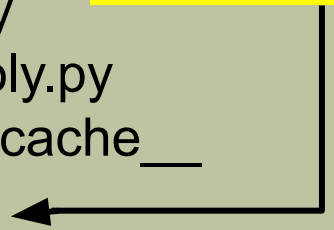
Package — коллекция модулей со структурированным пространством имён.

Пакеты и модули

```
...ws/numsch> ls -l
total 16
-rw-r--r-- 1 malkov users 2 Apr 2 13:07 __init__.py
drwxr-xr-x 3 malkov users 4096 Apr 2 13:15 interpol
drwxr-xr-x 2 malkov users 4096 Apr 2 13:09 __pycache__
drwxr-xr-x 3 malkov users 4096 Apr 2 13:09 weno

...ws/numsch> ls -l interpol
total 16
-rw-r--r-- 1 malkov users 2 Apr 2 13:08 __init__.py
-rw-r--r-- 1 malkov users 2458 Apr 2 13:14 interpoly.py
drwxr-xr-x 2 malkov users 4096 Apr 2 13:18 __pycache__
-rw-r--r-- 1 malkov users 150 Apr 2 13:18 test.py
```

```
def tt(s):
  print(s)
  return
```



```
...ws> python
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul 2 2016, 17:53:06)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numsch.interpol.test as t
>>> t.tt(34.8)
34.8
>>>
```

Сравненние производительности кодов, генерируемых компилятором Numba

```
import numpy as np
from pylab import imshow, show
from time import  perf_counter_ns  as timer

def mandel(x, y, max_iters):
  c = complex(x, y)
  z = 0.0j
  for i in range(max_iters):
    z = z*z + c
    if (z.real*z.real + z.imag*z.imag) >= 4:
      return i
 return max_iters
```

http://nbviewer.jupyter.org/gist/harrism/f5707335f40af9463c43

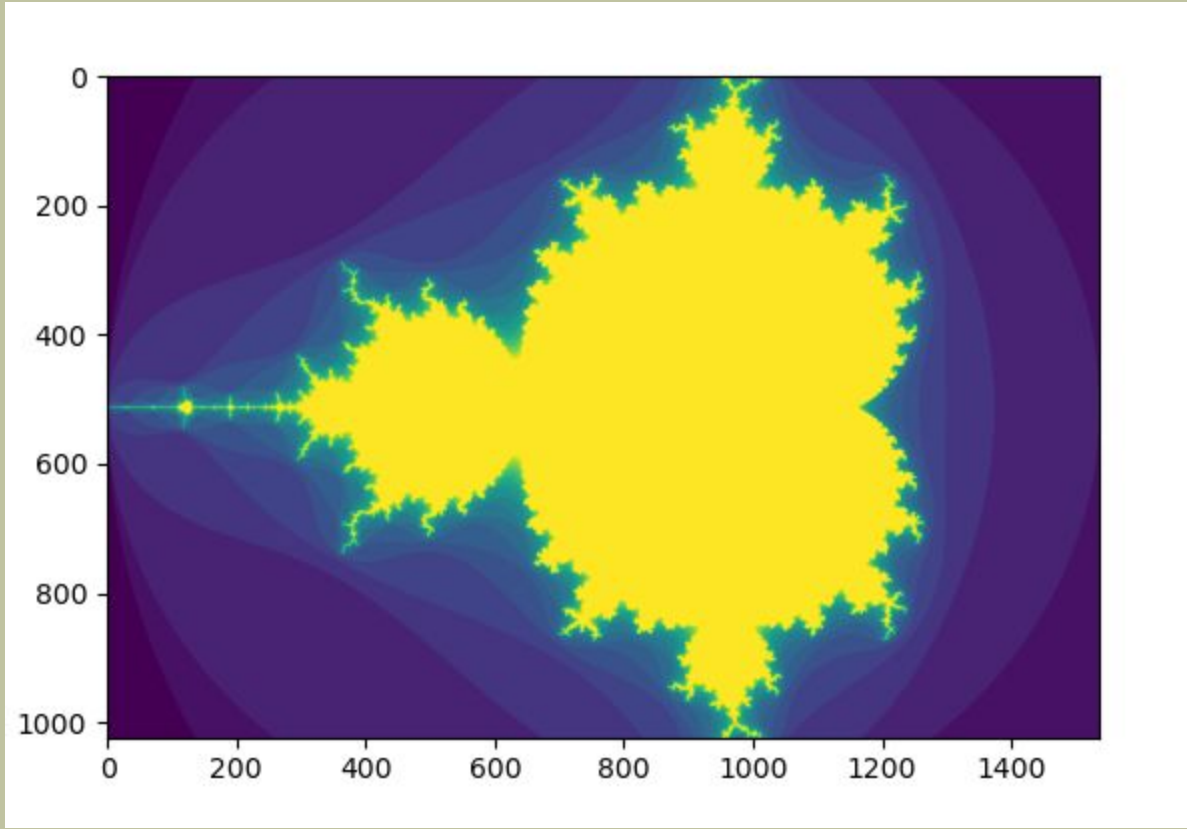
```
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
  height = image.shape[0]
#размерности двумерного массива
  width = image.shape[1]
  pixel_size_x = (max_x - min_x) / width
  pixel_size_y = (max_y - min_y) / height
#задание размеров пикселя
  for x in range(width):
    real = min_x + x * pixel_size_x
    for y in range(height):
      imag = min_y + y * pixel_size_y
      color = mandel(real, imag, iters)
      image[y, x] = color
```

```
#задание цвета пикселя
image = np.zeros((1024, 1536), dtype = np.uint8)

start = timer()
create_fractal(-2.0, 1.0, -1.0, 1.0, image, 20)
dt = timer() - start
dt/=1000000
print ("Mandelbrot created in %f ms" % dt)

imshow(image)
show()
```

```
/Lab10> python mandel.py
Mandelbrot created in 3332.904220 ms
```



```
import numpy as np
from pylab import imshow, show
from time import  perf_counter_ns  as timer
from numba import jit
@jit
def mandel(x, y, max_iters):
  c = complex(x, y)
  z = 0.0j
  for i in range(max_iters):
 z = z*z + c
 if (z.real*z.real + z.imag*z.imag) >= 4:
   return i
return max_iters
```

```
@jit
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
  height = image.shape[0]
  width = image.shape[1]
  pixel_size_x = (max_x - min_x) / width
  pixel_size_y = (max_y - min_y) / height
  for x in range(width):
    real = min_x + x * pixel_size_x
    for y in range(height):
      imag = min_y + y * pixel_size_y
      color = mandel(real, imag, iters)
      image[y, x] = color
```

```
image = np.zeros((1024, 1536), dtype = np.uint8)

start = timer()
create_fractal(-2.0, 1.0, -1.0, 1.0, image, 20)
dt = timer() - start

dt/=1000000
print ("Mandelbrot created in %f ms" % dt)

imshow(image)
show()
```

```
Lab10> python mandel_numba.py
Mandelbrot created in 250.150089 ms
```

```
import numpy as np
from pylab import imshow, show
#from timeit import default_timer as timer
from time import  perf_counter_ns  as timer
from numba import cuda
from numba import *

@cuda.jit('f8, f8, uint32', device=True)
def mandel(x, y, max_iters):
 c = complex(x, y)
 z = 0.0j
 for i in range(max_iters):
  z = z*z + c
  if (z.real*z.real + z.imag*z.imag) >= 4:
    return i
 return max_iters
```

```
@cuda.jit('f8, f8, f8, f8, uint8[:,:], uint32')
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
  height = image.shape[0]
#размерности двумерного массива
  width = image.shape[1]
  pixel_size_x = (max_x - min_x) / width
  pixel_size_y = (max_y - min_y) / height

  startX, startY = cuda.grid(2) #threadIdx.x+blockDim.x*blockIdx.x,...
  gridX = cuda.gridDim.x * cuda.blockDim.x;
  gridY = cuda.gridDim.y * cuda.blockDim.y;

  for x in range(startX, width, gridX): #если width>gridX
    real = min_x + x * pixel_size_x
    for y in range(startY, height, gridY):
      imag = min_y + y * pixel_size_y
      image[y, x] = mandel(real, imag, iters)
```

```
image = np.zeros((1024, 1536), dtype = np.uint8)
blockdim = (32, 8)
griddim = (32,16)

d_image = cuda.to_device(image)
create_fractal[griddim, blockdim](-2.0, 1.0, -1.0, 1.0, d_image, 20)
cuda.synchronize()

start = timer()
d_image = cuda.to_device(image)
create_fractal[griddim, blockdim](-2.0, 1.0, -1.0, 1.0, d_image, 20)
cuda.synchronize()
dt = timer() - start
dt/=1000000

print ("Mandelbrot created in %f ms" % dt)
imshow(d_image)
show()
```

```
/Lab10> python mandel_cuda.py
Mandelbrot created in 2.216281 ms
```

```
/Lab10> nvprof python mandel_cuda.py
    Type  Time(%)      Time     Calls       Avg       Min       Max  Name
GPU activities:   93.77%  4.3107ms         2  2.1554ms  2.1450ms  2.1657ms
cudapy::__main__::create_fractal$242(double, double, double, double,
Array<unsigned char, int=2, A, mutable, aligned>, unsigned int)
```

Технология	Время выполнения *мс*	Ускорение
Python интерпретатор	3332.90	1
Numba jit	250.15	13.32
CUDA	2.16	**1543.01**

```
import numpy as np
from numba import jit
from numba import cuda
from time import  perf_counter_ns  as timer
from numba import cuda, float32

@cuda.jit#(argtypes=[float32[:,:], float32[:,:], float32[:,:]])
def matmul(A, B, C):
   m,n = cuda.grid(2)
   if m < C.shape[0] and n < C.shape[1]:
       acc = 0.
       for k in range(A.shape[1]):
           acc += A[m, k] * B[k, n]
       C[m, n] = acc
```

```
M=1024
K=1024
N=1024

A=np.arange(M*K)
A=A.reshape(M,K)
B=np.ones((K,N))
#C=np.zeros((M,N))
np.set_printoptions(formatter={'float': '{: 0.3g}'.format})

start = timer()
C=np.matmul(A,B)
dt = timer() - start
dt=dt/1000000.0
print ("np.matmul time %f ms" % dt)
print(C)
```

```
A=np.float32(A)
B=np.float32(B)
C=np.float32(C)

dA=cuda.to_device(A)
dB=cuda.to_device(B)
dC=cuda.to_device(C)
```

```
start = timer()
matmul[(64,64),(16,16)](dA,dB,dC)
cuda.synchronize()
dt = timer() - start

#dC.to_host()
dt=dt/1000000.0
print ("matmul time %f ms" % dt)
print(dC)
```

```
/Lab10> python cuda-gemm3.py
np.matmul time 20.236182 ms
[[ 5.24e+05  5.24e+05  5.24e+05 ...  5.24e+05  5.24e+05  5.24e+05]
 [ 1.57e+06  1.57e+06  1.57e+06 ...  1.57e+06  1.57e+06  1.57e+06]
 ...
 [ 1.07e+09  1.07e+09  1.07e+09 ...  1.07e+09  1.07e+09  1.07e+09]]
matmul time 30.242027 ms
[[ 5.24e+05  5.24e+05  5.24e+05 ...  5.24e+05  5.24e+05  5.24e+05]
 [ 1.57e+06  1.57e+06  1.57e+06 ...  1.57e+06  1.57e+06  1.57e+06]
 ...
 [ 1.07e+09  1.07e+09  1.07e+09 ...  1.07e+09  1.07e+09  1.07e+09]]
```

```
/Lab10> nvprof python cuda-gemm3.py
```

```
Type
Time(%)   Time      Calls       Avg             Min          Max          Name
GPU activities:
49.23%  62.324ms   2       31.162ms    30.458ms    31.866ms
cudapy::__main__::matmul$242(Array<float, int=2, A, mutable, aligned>,
Array<float, int=2, A, mutable, aligned>, Array<float, int=2, A, mutable, aligned>)
```