GPU Programming with Google Colaboratory

Steps

Step1: Getting Started: Use google Colab with your gmail id

Step 2: Using GPU (Accelerator)

Step 3: Tour of the User Interface

Step 4: Install GCC 10: Required for OpenMP 5.0

Step 5: Use (not install) CUDA 10.0

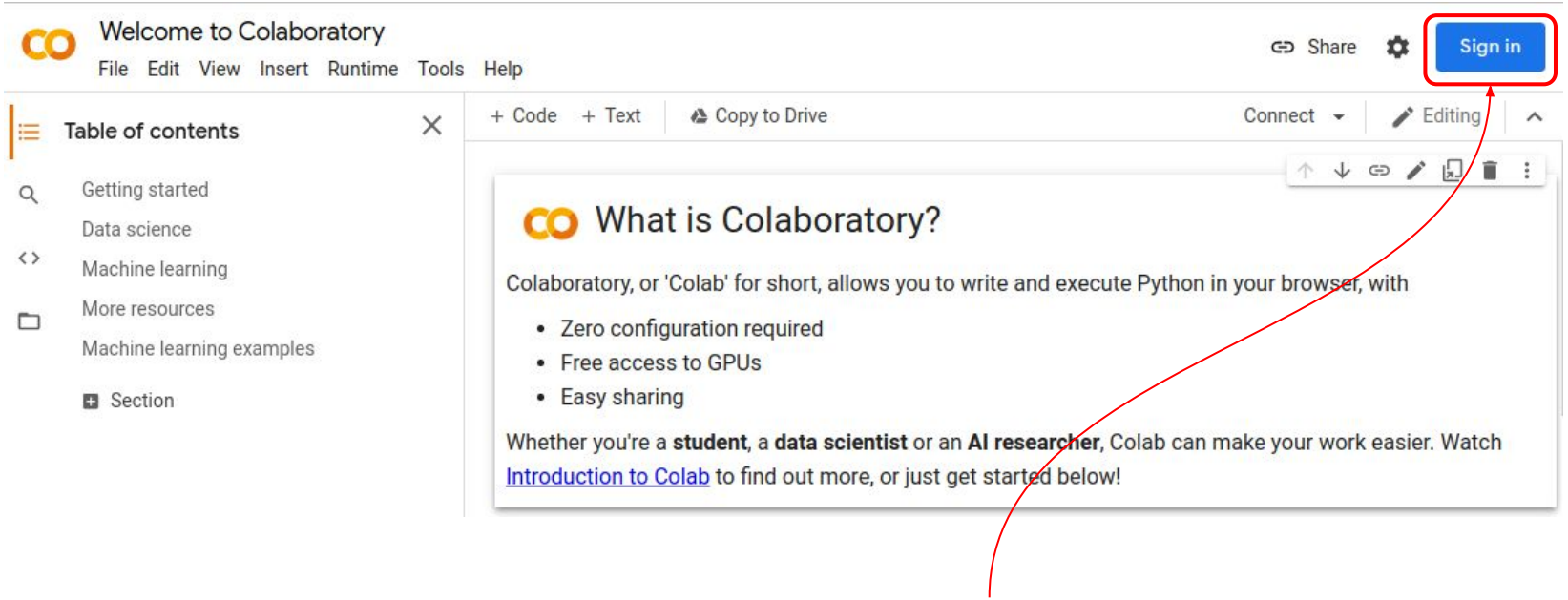
Step 6: Compiling OpenMP program : Example

Step 7: Profiling GPU with nvprof.

A Demo at the end.

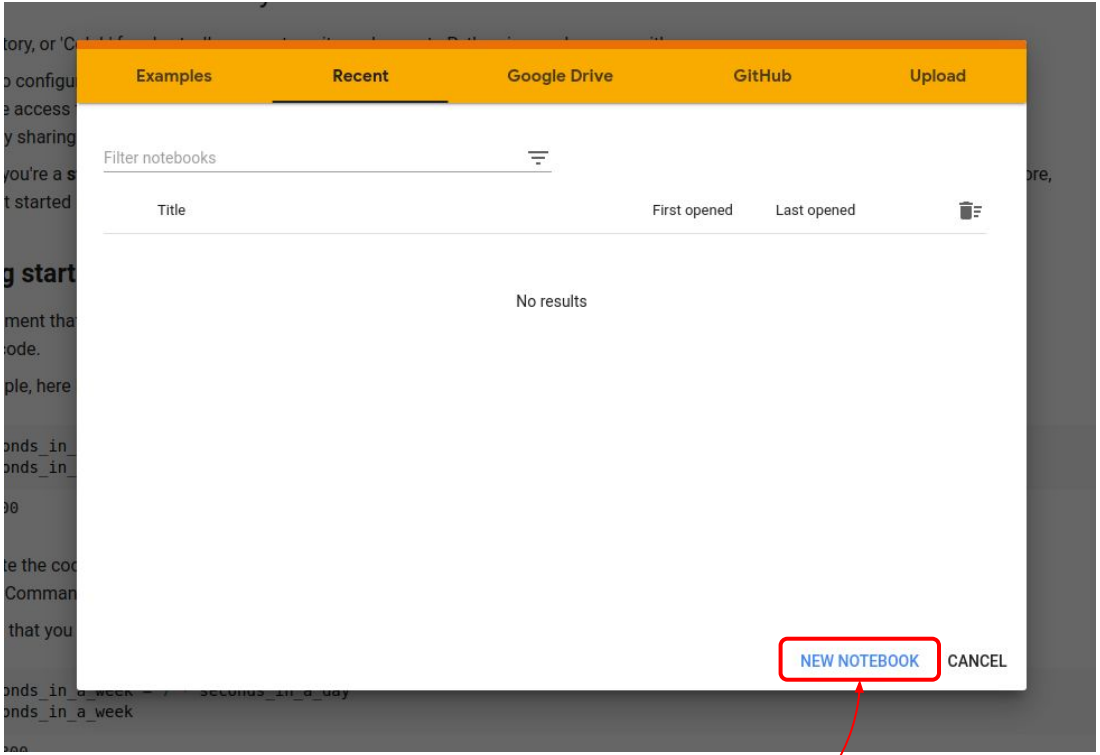
Step 1 - Getting Started

Go to Google Colab - https://colab.research.google.com



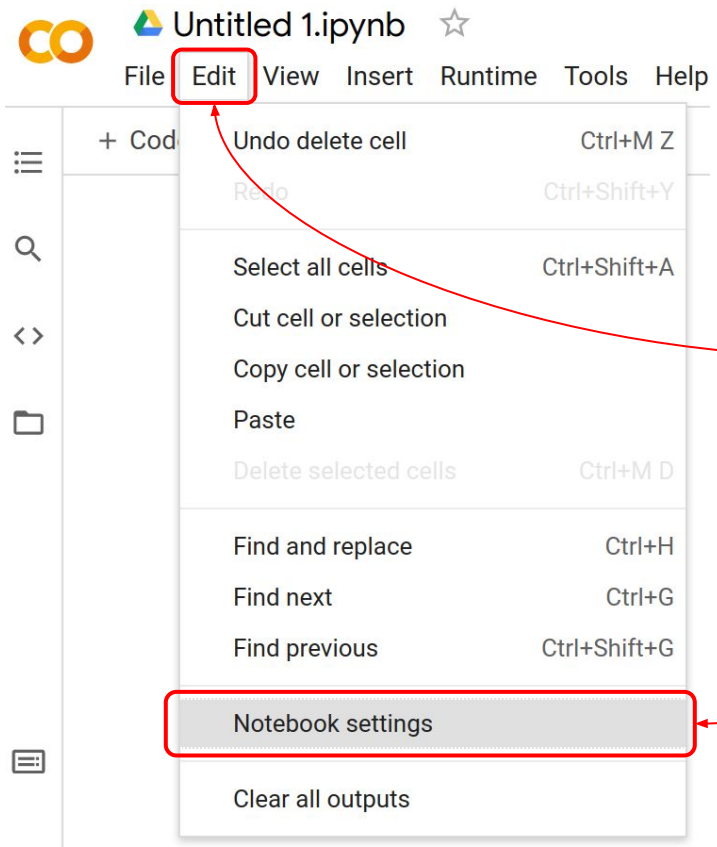
Login with your Google Account

Step 1 - Getting Started



Click on NEW NOTEBOOK

Step 2 - Choose GPU Accelerator

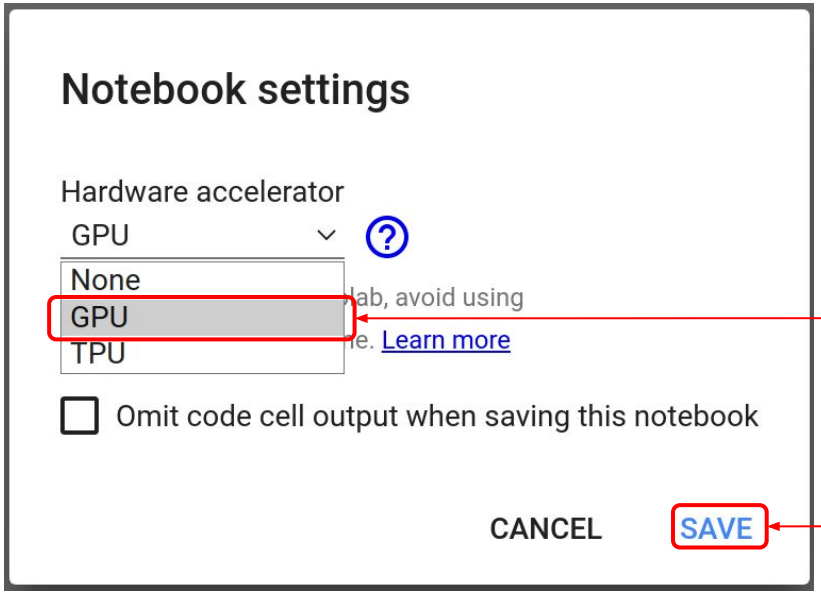


First, you'll need to enable GPUs for the notebook:

Navigate to Edit

Click on Notebook Settings

Step 2 - Choose GPU Accelerator



Select GPU in the Hardware accelerator drop-down menu

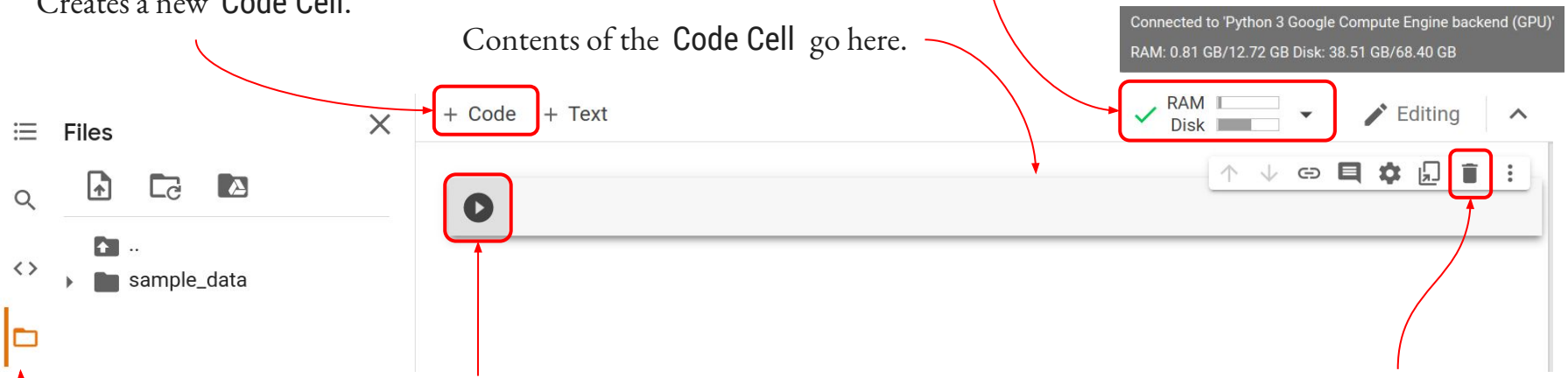
Click on SAVE

Step 3 - Tour

Hover over to see your allocated runtime.

Creates a new **Code Cell**.

Contents of the **Code Cell** go here.



Executes the contents of the **Code Cell**.

Deletes a **Code Cell**.

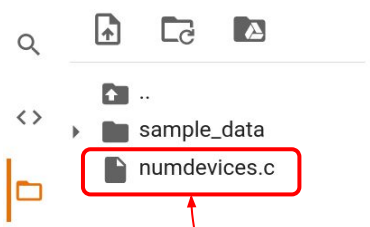
Click Here to see your files in the current session.

Step 3 - Tour

```
!ls /usr/local/cuda
```

```
bin     EULA.txt  lib64            nvml     Sanitizer  targets
compat  extras    libnvvp          nvvm     share      tools
doc     include   nsightee_plugins  samples  src        version.txt
```

shell commands are prefixed with !




```
%%writefile numdevices.c

#include <stdio.h>
#include <omp.h>

int main(void) {
  int NumDevices = omp_get_num_devices();
  printf ("Number of Devices: %d", NumDevices);
}
```

```
Writing numdevices.c
```

The writefile magic writes the contents of the cell into a named file, in our case, numdevices.c

Type the contents of numdevices.c

numdevices.c after executing the above cell

Step 4 - Install GCC with OpenMP offload support

```
!true | add-apt-repository ppa:ubuntu-toolchain-r/test
 Toolchain test builds; see https://wiki.ubuntu.com/ToolChain

 More info: https://launchpad.net/~ubuntu-toolchain-r/+archive/ubuntu/test
Get:1 https://cloud.r-project.org/bin/linux/ubuntu bionic-cran40/ InRelease [3,626 B]
Ign:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  InRelease
Ign:3 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  InRelease
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64  Release
Get:5 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic InRelease [15.9 kB]
Hit:6 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64  Release
Hit:8 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:10 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:12 http://ppa.launchpad.net/cran/libgit2/ubuntu bionic InRelease
Hit:13 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu bionic InRelease
Get:14 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:15 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2,009 kB]
Get:16 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease [21.3 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2,165 kB]
Get:18 http://ppa.launchpad.net/ubuntu-toolchain-r/test/ubuntu bionic InRelease [15.4 kB]
Get:19 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main Sources [1,747 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [339 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [368 kB]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2,439 kB]
Get:23 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu bionic/main amd64 Packages [894 kB]
Get:24 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic/main amd64 Packages [49.4 kB]
Get:25 http://ppa.launchpad.net/ubuntu-toolchain-r/test/ubuntu bionic/main amd64 Packages [39.9 kB]
Fetched 10.4 MB in 8s (1,292 kB/s)
Reading package lists... Done
```

Add an extra repository to fetch newer GCC compiler.

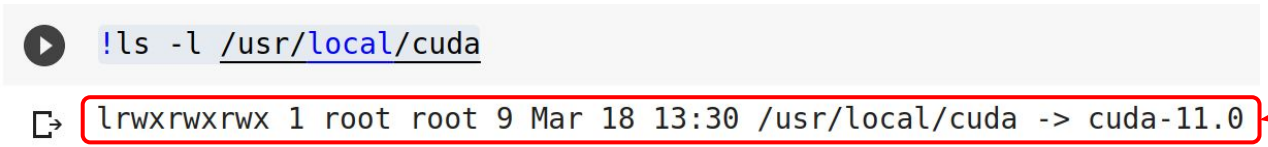
Step 4 - Install GCC with OpenMP offload support

```
!apt install gcc-10 g++-10 gcc-10-offload-nvptx libgomp1
```

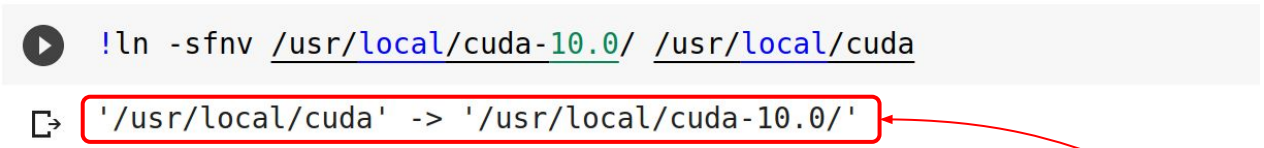
```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpp-10 gcc-10-base libasan6 libatomic1 libcc1-0 libgcc-10-dev libgcc-s1
  libgomp-plugin-nvptx1 libitm1 liblsan0 libquadmath0 libstdc++-10-dev
  libstdc++6 libtsan0 libubsan1 nvptx-tools
Suggested packages:
  gcc-10-locales g++-10-multilib gcc-10-doc libstdc++6-10-dbg gcc-10-multilib
  libgcc-s1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan6-dbg
  liblsan0-dbg libtsan0-dbg libubsan1-dbg libquadmath0-dbg libstdc++-10-doc
  nvidia-cuda-toolkit
The following NEW packages will be installed:
  cpp-10 g++-10 gcc-10 gcc-10-base gcc-10-offload-nvptx libasan6 libgcc-10-dev
  libgcc-s1 libgomp-plugin-nvptx1 libstdc++-10-dev libubsan1 nvptx-tools
The following packages will be upgraded:
  libatomic1 libcc1-0 libgomp1 libitm1 liblsan0 libquadmath0 libstdc++6
  libtsan0
8 upgraded, 12 newly installed, 0 to remove and 57 not upgraded.
```

Install GCC 10 C/C++ compiler, OpenMP Support Library, and NVPTX offload support.

Step 5 - Use CUDA 10.0



The default CUDA SDK in use is 11.0.



Google Colab supplies CUDA SDK 10.0, 10.1 and 11.0 in the allocated instance.

Only CUDA 10.0 works correctly with GCC 10 offloading compiler out of the box.

Step 5 - Compiling an OpenMP program with Offloading support

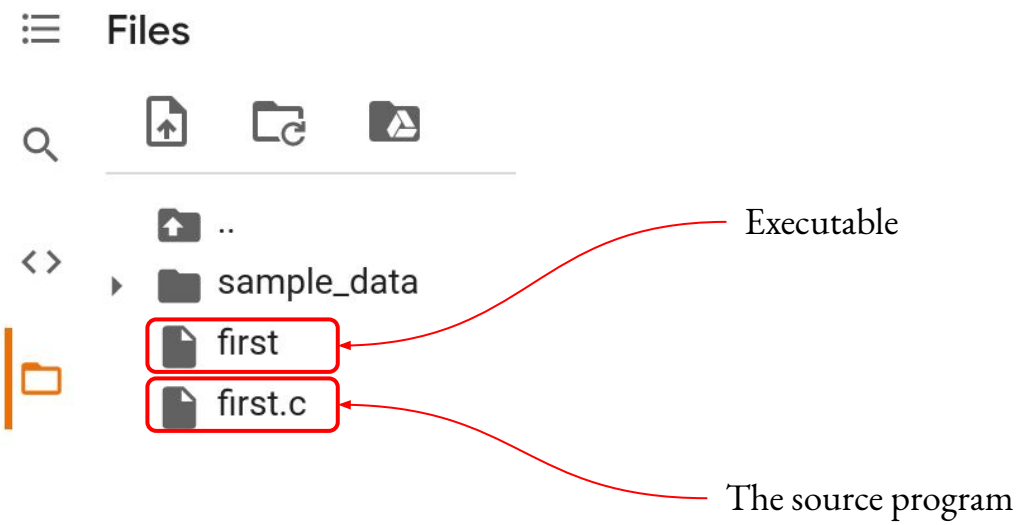
```
%%writefile first.c
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>

#define N  1024
int A[N][N], B[N][N], C[N][N];

int main() {
  srand(3);
  for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++) {
      A[i][j] = rand() % N + 1;
      B[i][j] = rand() % N + 1;
    }
  }

  printf("Total Devices = %d \n", omp_get_num_devices());
#pragma omp target data map(to: A, B) map(from: C)
{
#pragma omp target teams distribute parallel for
  for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++)
      C[i][j] = A[i][j] * B[i][j];
  }
}
  return 0;
}
```

```
Writing first.c
```

Write the example program into first.c

Step 5 - Compiling an OpenMP program with Offloading support

```
!g++-10  -fno-lto -fopenmp -foffload=nvptx-none -fstack-protector first.c -o first
```



Step 6 - Execution with nvprof

```
!nvprof ./first
```

```
==1689== NVPROF is profiling process 1689, command: ./first
Total Devices = 1
==1689== Profiling application: ./first
==1689== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   74.00%  1.5223ms         2  761.17us  747.71us  774.62us  [CUDA memcpy HtoD]
                   26.00%  534.83us         1  534.83us  534.83us  534.83us  [CUDA memcpy DtoH]
      API calls:   78.31%  208.05ms         1  208.05ms  208.05ms  208.05ms  cuCtxCreate
                   20.47%  54.390ms         1  54.390ms  54.390ms  54.390ms  cuCtxDestroy
                    0.72%  1.9177ms         2  958.83us  945.50us  972.15us  cuMemcpyHtoD
                    0.32%  855.92us         1  855.92us  855.92us  855.92us  cuMemcpyDtoH
                    0.07%  185.39us         1  185.39us  185.39us  185.39us  cuMemAlloc
                    0.06%  169.48us         1  169.48us  169.48us  169.48us  cuMemFree
                    0.03%  74.036us        16  4.6270us     141ns  68.491us  cuDeviceGetAttribute
                    0.00%  12.778us         1  12.778us  12.778us  12.778us  cuDeviceGetName
                    0.00%  6.0740us         3  2.0240us  1.1020us  3.1300us  cuMemGetAddressRange
                    0.00%  5.8330us         1  5.8330us  5.8330us  5.8330us  cuDeviceGetPCIBusId
                    0.00%  5.1620us         7     737ns     234ns  1.7030us  cuCtxGetDevice
                    0.00%  1.8710us         1  1.8710us  1.8710us  1.8710us  cuInit
                    0.00%  1.8590us         4     464ns     244ns  1.0720us  cuDeviceGetCount
                    0.00%  1.4280us         2     714ns     414ns  1.0140us  cuDeviceGet
                    0.00%     661ns         1     661ns     661ns     661ns  cuCtxGetCurrent
                    0.00%     255ns         1     255ns     255ns     255ns  cuDriverGetVersion
```