

CUDA_examples REPOSITORY TUTORIAL:

Author do recommend such logical sequence at development of CUDA program:

- get familiar with C programming language,
- read <https://en.wikipedia.org/wiki/CUDA>
- buy book „CUDA by examples” et. al. Sanders,
- write few dozens of your own programs,
- get an online Youtube course „Udacity: intro to parallel programming”,
- write few dozens of your own programs,
- buy book „Profesional CUDA programming” et. al. Cheng.

This step by step tutorial is intendent for executing repository CUDA programs. It consist of tip dashes:

1) obtain computer/laptop/tablet/phone with included NVidia GPU graphics (programs are written in Nvidia corporation C extensions language with some C++11 capabilities - Compute Unified Device Architecture CUDA). If you have not any, I will provide you some new-parts proposition in 2018y:

- CPU: Intel Celeron G3900
- MB: Gigabyte GA-B150-HD3 (note PCIe 3.0 x16 bandwidth ~12.6GBps)
- RAM: 1x GOODRAM PLAY Blue 2 / 4GB,
- HDD: 120GB SSD GOODRAM, or two pendrives 16GB with Lubuntu desktop 64bit live session on first pendrive, power supply,
- GPU: Nvidia GT730 4GB GDDR5 / GT 1030 2GB
 - tip: consider operating system distribution booting to RAM (f.e. modified Ubuntu),
 - tip: consider good quality monitors, chair and keyboard (for my best knowledge mouse is unnecessary and slows work),
 - tip: for parallel cluster consider adding SSD RAID 0 (#mdadm are not so computationally expensive, as you think, especially for multicore CPU) and sufficient network connection,
 - tip: theoretical lower price – any used Personal Computer with PCIe1.0x16 (f.e. Core2Duo s775 + 1x1GB RAM + NVS300 + 2xpendrive 16GB). I did not check that configuration by myself, but should work,
 - tip: if you disable graphical server (#sudo service lightdm stop), you will get additional free space of +0.5GB RAM +0.34GB GPU memory, (author do recommend vim + screen in two windows), additionally some programs might execute faster,
 - tip: making software RAID 0 on hardware SSD RAID 0 disk array controller for big-data efficient streaming processing.
 -

via .DEB PACKAGE

2) download cudaXX.deb package (if you have at least 6.x cuda-computation capable device - newer than GT1030):

<https://developer.nvidia.com/cuda-toolkit>

othercase (GTX 2/4/5/6/7/9xx) download CUDA-Toolkit 8.0 from:

<https://developer.nvidia.com/cuda-80-ga2-download-archive>

3) in terminal:

- switch to terminal view:
 - #Ctrl+Alt+F1 (to switch on terminal view)
 - #Ctrl+Alt+F7 (to switch from terminal view to graphical server)
- at terminal view (Ctrl+Alt+F1) log in:
 - #login: admin
 - #pass: admin1
 - #sudo chmod +x cudaXX.deb
 - #sudo dpkg -i cudaXX.deb
 - #sudo apt-get update

- 4) enable GPU driver at: System Settings->Additional Drivers-> Nvidia Driver xx
- 5) install CUDA Toolkit:


```
#sudo aptitude install cuda g++-4.9 gcc-4.9
#sudo ln -s /usr/bin/g++-4.9 /usr/local/cuda/bin/g++
#sudo ln -s /usr/bin/gcc-4.9 /usr/local/cuda/bin/gcc
```
- 6) restart PC, check whether CUDA programs are working properly – start with CUDA_examples/02_deviceFeatures


```
#!/RUN_COMMANDS.sh
```
- 7) if something went wrong, you can reinstall cuda package:


```
#dpkg -l | grep cuda- | awk '{print $2}' | xargs -n1 sudo dpkg --purge
#dpkg -i cuda**.deb
#sudo apt-get update
#sudo apt-get install cuda-X-X
```

Post Scriptum: Mostly I have been developing it on Dell Inspiron 7746 with i7 5500U, 16GB RAM 1666MHz, Nvidia GM108M GeForce 845M PCIe 3.0x4 2GB and SSD 1TB (OS: LINUX Ubuntu Desktop 16.04 64bit; LINUX Ubuntu Desktop 16.10 64bit). Also it was checked on PC's: Intel i7 2660k, 16GB 2 channel RAM 1333MHz, Nvidia GK110 GTX780 3GB PCIe 2.0x16, SSD 0.24TB (OS: LINUX Ubuntu 16.04 64bit) and Intel Celeron G3900, 12GB 2 channel RAM DDR3 1333MHz, Nvidia GK110 GTX780 3GB PCIe 3.0x16, Nvidia GP108 GT1030 2GB PCIe 3.0x4.

Post Post Scriptum: If one will use customized liveCD from LINUX_tips repo, he or she should take into consideration that minimal recommended RAM size should be 8GB.

Please do consider registers amount for some hardware:

GPR – General Purpose Register

FPR – Floating Point Register

Architecture, or processor	GPR	FPR
4004	1	0
16bit x86	8	8
x86 / IA-32 (80386)	8	8
x86-64(AMD 64)	16	16
IA-64 (Itanium)	128	128
Xeon Phi	16	32
CUDA	1	128*
Cray-1 GPR(data + addresses); FP(scalar * vector)	8+8=16	8*8=64
IBM POWER / POWERPC	32	32
IBM Cell	128	
SPARC	32	32
AVR MCU	32	0
ARM A32 / ARM T32	14	1-32
ARM A64	31	32
Epiphany	64**	
MIPS	32	32
SW26010***	264	16.896

*3584 CUDA cores at 2nd Maxwell generation GP102 processor (GTX1080ti) – there are 1.835.008 FPR's

**64 cores at E64G401 – there is 4096 register acting as GPR's or FPR's,

*** amount approximated

GPU + global memory type	GTX 780 GDDR5	GT 1030 GDDR5
max GFLOPs (float 32 bit)	4000	942
measured GFLOPs (double 64 bit)	198	42
measured GFLOPs (float 32 bit)	3747	1310
measured Giops (integer 32bit)	790	440
Size of global memory [GB]	3	2
Bandwidth [GBps]	288.4	48
Practical throughput [GBps]	95	19
GFLOPs / PLN	5.33	3.14
Price PLN (1PLN = 0.28USD)	750	300
2channel DDR3 1333MHz has practical throughput of ~6.8GBps and theoretical bandwidth of 20GBps		

for your RAM

details please refer to:

<http://manpages.ubuntu.com/manpages/xenial/man1/mbw.1.html>

MGPGPU (many GPU for the same price	-	3*GT730	-	-	4*GTX1080ti	-
GPU + global memory type	GT 730 GDDR5	GTX 780 GDDR5	GT 1030 GDDR5	GTX 1080ti GDDR5X	P6000 GDDR5X	Intel Xeon E-2660 (16 HT cores; 2.2GHz) DDR3 (4-channel)
Max. GFLOPs	690	4000	942	10600	10800	166
Size of global memory [GB]	4	3	2	11	24	~4 (GB / core)
Bandwidth [Tbps] (Throughput is about 50-60% of bandwidth)	0.04	0.2884	0.048	0.484	0.432	0.05
PCIe [2.0x16=x0.8; 3.0x16=x1; NVLink=x2]	0.8	1	1	1	2	1
Type [used=x1, brand-new=x1.2]	1.2	1	1.2	1.2	1.2	1
Slot size + Power (<45W)	1.2	1	1.2	1	1	1
GFLOPs / USD	0.8	1.5	0.9	0.9	0.3	0.1
Max power [W]	30	250	30	250	250	95
GFLOPs / W	23.00	16.00	31.4	42.40	43.20	1.75
Price USD	70	210	84	924	3360	109
normalized to P6000: GFLOPs*throughputGBps*memSizeGB / USD	23.74	62.02	6.75	191.53	62.71	0.28

Please do refer to:

https://asteroidsathome.net/boinc/cpu_list.php

<http://cuda-z.sourceforge.net/>

https://github.com/PiotrLenarczyk/CPlusPlus_examples/tree/master/78_linpackGFLOPS