1 **OpenCL**
- Crash Course

2 **Sources + slides**
- Known to work on Linux/OSX:
  - a C++ compiler (g++/clang++)
  - the OpenCL Header + libs cmake / OpenCV / Boost / GL / GLU / glut
  - Based on (Simple & Floyd-Warshall & Video):
    http://github.com/anirul/OpenCL_PA_2012.git
    http://github.com/anirul/OpenCL_Video.git
  - Warning! The Nvidia drivers on linux only support OpenCL 1.1!

3 **Plan**
- General overview (GPGPU -> OpenCL)
- Code dive (various examples)
- Conclusion (Optimisation tips)

4 **General Overview**
- GPGPU - Technology overview
- OpenCL
  - Language and API
  - Device Model
  - Objects

5 **GPGPU**
- Using Graphical Processing Unit to compute.
  - Shader languages (GLSL / DirectX)
  - CUDA (Nvidia proprietary)
  - DirectCompute (Windows)
  - OpenACC (no free compiler support yet)

6 **OpenCL**
- Khronos (OpenGL, Vulkan, COLLADA, etc…)
  - Intel, QUALCOMM, AMD, Altera Corporation, Vivante Corporation, Xilinx, Inc., MediaTek Inc, ARM Limited, Imagination Technologies, Apple, Inc., STMicroelectronics International NV, ARM, IBM Corporation, Creative Labs, NVIDIA, Samsung Electronics.

- Work on CPU / GPU / DSP / FPGA …
- Open Standard

7 **OpenCL**

- An API (run on the Host)
  - in C but with interface to many other languages
- A language (run on the Device)
  - vector oriented
  - C99 inspired (2.1 -> C++14)

8 **OpenCL**

- An API (run on the Host)
  - in C but with interface to many other languages
- A language (run on the Device)
  - vector oriented
  - C99 inspired (2.1 -> C++14)

9 **OpenCL**

- Khronos (OpenGL, Vulkan, COLLADA, etc…)
  - Intel, QUALCOMM, AMD, Altera Corporation, Vivante Corporation, Xilinx, Inc., MediaTek Inc, ARM Limited, Imagination Technologies, Apple, Inc., STMicroelectronics International NV, ARM, IBM Corporation, Creative Labs, NVIDIA, Samsung Electronics.
- Work on CPU / GPU / DSP / FPGA …
- Open Standard

10 **OpenCL**

- Forget about memory protection (no MMU)
- No recursion!
- memory limited (no swap)
- local memory bottleneck (64k typically)

11 **Code Dive**

- Khronos C++ wrapper

- Simple - (as it can be!)
- Floyd-Warshall - (memory coalescing)
- Histogram (reduce, local memory)
- Video (the full stack)

12 **Khronos C++ wrapper**
- Officially published by Khronos
- Template based
- Header only
- 100% portable
- Object Oriented

13 **Simple**
- Very simple example using the C++ wrapper
- Compute the product of 2 vectors
- Single file only OpenCL dependent
  - simple.cpp
- Not a practical example!

14 **Device code**
- Simple product of two vectors

15 **Host code (1)**
- Add exception support to OpenCL

16 **Host code (2)**
- Generate a context

17 **Host code (3)**
- Set the arguments of the kernel

18 **Host code (4)**
- Get the result

19 **Floyd Warshall**
- mat (whole)
- in_x = y[k]

- in_y = x[k]

20 **Device code**

- Simple product of two vectors

21 **Host code**

- Add exception support to OpenCL

22 **Remarks**

- Memory coalescing
- Loop on kernel
- local memory is typically 64k (on Nvidia)

23 **Performances**

- Very simple example using the C++ wrapper
- Compute the product of 2 vectors
- Single file only OpenCL dependent
  - simple.cpp
- Not a practical example!

24 **Histogram**

- mat (whole)
- in_x = y[k]
- in_y = x[k]

25 **Histogram**

- Split into 4 kernels:
  - compute luminosity (on image2d)
  - clean the partial histogram buffer
  - partial histogram (to avoid atomic hell)
  - collect all partial histogram (could be split more if needed)

26 **Device code (1)**

- Split into 2 kernel
  - partial histogram (to avoid atomic hell)
  - collect all partial histogram (could be split more if needed)

27 **Device code (2)**
- Split into 2 kernel
  - partial histogram (to avoid atomic hell)
  - collect all partial histogram (could be split more if needed)

28 **Device code (3)**
- Split into 2 kernel
  - partial histogram (to avoid atomic hell)
  - collect all partial histogram (could be split more if needed)

29 **Host code (1)**
- Create the different kernel from the same program

30 **Host code (2)**
- Create an image2d from a pointer

31 **Remarks**
- multiple kernel in a single program
- usage of image2d!
- map and collect technique
- atomic (warning atomic only work on int!)
  (there is ways to make it work with float)

32 **Performances**

33 **Video**
- Take a video

34 **Video**
- OpenCV to capture video
- OpenCL to modify it
  - copy to OpenGL could be avoided with OpenCL interop (OS / Hardware dependent)
- OpenGL to draw it

35 **Device code**
- simple code to copy from image_in to image_out

36 **Remarks**

- Kernel can be swap from the command line
- Direct feed back from the window

37 **Conclusion**

- Third part (conclusion)
  - Optimisation tips
  - OpenGL <-> OpenCL interop
  - Vulkan / OpenCL 2.1 / OpenACC / boost::compute
  - Questions?

38 **Optimisation Tips (1)**

- Overall
  - Maximizing parallel execution
  - Optimizing memory usage to achieve maximum memory bandwidth
  - Optimizing instruction usage to achieve maximum instruction throughput

39 **Optimisation Tips (2)**

- Don’t trust, test!
- Sometime using image is faster than using buffers
- Use barrier([memory type]) to synchronize inside a kernel
- Local memory don’t exist on CPU!
- The compiler is sometime (too) clever
- Watch out, branching can kill you (or not)
- For b power of 2 -> a % b -> a & (b -1)

40 **OpenCL / OpenGL**

- very platform dependent
- still a moving target
- usually drawing to the screen won’t be your bottleneck

41 **Vulkan**

- Vulkan is the fusion between OpenCL and OpenGL
  - access to the full pipeline in raw mode
  - OpenCL will compile to vulkan (SPIR-V)

42 **OpenCL 2.1**

- OpenCL 2.1 build on top of Vulkan (SPIR-V)
  - subset of the C++14 language
  - still compatible with C99 syntax

43 **OpenACC**

- OpenMP type optimisation (#pragma acc)
  - Should be supported in gcc 5

44 **boost::compute**

- A boost library for GPU programming?
  - Still under heavy development

45 **Questions?**