Lecture 1

SETTING UP OPENCL PLATFORMS

Some notes on setting up OpenCL

- We will provide some instructions for setting up OpenCL on your machine for a variety of major platforms and device types
  - AMD CPU, GPU and APU
  - Intel CPU
  - NVIDIA GPU
- We assume you are running 64-bit Ubuntu 12.04 LTS

Many platforms!

- OpenCL runs on many different platforms
- In order to co-ordinate which library each device uses, the runtime uses the ICD
- If you compile your code against a generic runtime, the ICD will load the correct platform runtime when required
- These runtimes are listed in the /etc/OpenCL/vendors directory

Running OSX?

- OpenCL works out of the box!
- Just compile your programs with

```
-framework OpenCL -DAPPLE
```

Setting up with AMD GPU

- Install some required packages:
  - `sudo apt-get install build-essential linux-headers-generic debhelper dh-modaliases execstack dkms lib32gcc1 libc6-i386 opencl-headers`
- Download the driver from amd.com/drivers
  - Select your GPU, OS, etc.
  - Download the .zip
  - Unpack this with unzip filename.zip
- Create the installer
  - `sudo sh fglrx*.run --buildpkg Ubuntu/precise`
- Install the drivers
  - `sudo dpkg -i fglrx*.deb`
- Update your Xorg.conf file
  - `sudo amdconfig --initial --adapter=all`
- Reboot!
  - Check all is working by running `fglrxinfo`

* Fglrx is the name of AMD's graphics driver for the GPUs

Setting up with AMD CPU

- Download the AMD APP SDK from their website
- Extract with `tar -zxf file.tar.gz`
- Install with
  - `sudo ./Install*.sh`
- Create symbolic links to the library and includes
  - `sudo ln -s /opt/AMDAPP/lib/x86_64/* /usr/local/lib`
  - `sudo ln -s /opt/AMDAPP/include/* /usr/local/include`
- Update linker paths
  - `sudo ldconfig`
- Reboot and run `clinfo`
  - Your CPU should be listed

Setting up with AMD APU

- The easiest way is to follow the AMD GPU instructions to install fglrx.
- This means you can use the CPU and GPU parts of your APU as separate OpenCL devices.
- You may have to force your BIOS to use integrated graphics if you have a dedicated GPU too.

Setting up with Intel CPU

- NB: requires an Intel® Xeon™ processor on Linux
- Download the Xeon Linux SDK from the Intel website
- Extract the download
  - `tar -zxf download.tar.gz`
- Install some dependancies
  - `sudo apt-get install rpm alien libnuma1`
- Install them using alien
  - `sudo alien -i *base*.rpm *intel-cpu*.rpm *devel*.rpm`
- Copy the ICD to the right location
  - `sudo cp /opt/intel/<version>/etc/intel64.icd /etc/OpenCL/vendors/`

Setting up with Intel GPU

- NB: requires an Intel® Xeon™ processor on OS X

- This works out of the box!
- Just select the Intel® GPU device

- Intel® also have a driver for Windows: https://software.intel.com/en-us/articles/opencl-drivers

Setting up with Intel® Xeon Phi™

- Intel® Xeon Phi™ coprocessor are specialist processor only found in dedicated HPC clusters.
- As such, we expect most users will be using them in a server environment set up by someone else – hence we wont discusses setting up OpenCL on the Intel® Xeon Phi™ coprocessor in these slides

Setting up with NVIDIA GPUs

- Blacklist the open source driver (IMPORTANT)
  - `sudo nano /etc/modprobe.d/blacklist.conf`
  - Add the line: blacklist nouveau
- Install some dependencies
  - `sudo apt-get install build-essential linux-header-generic opencl-headers`
- Download the NVIDIA driver from their website and unpack the download
- In a virtual terminal (Ctrl+Alt+F1), stop the windows manager
  - `sudo service lightdm stop`
- Give the script run permissions then run it
  - `chmod +x *.run`
  - `sudo ./*.run`
- The pre-install test will fail – this is OK!
- Say yes to DKMS, 32-bit GL libraries and to update your X config
- Reboot!

Installing pyopencl

- Make sure you have python installed
- Install the numpy library
  - `sudo apt-get install python-numpy`
- Download the latest version from the pyopencl website
  - Extract the package with `tar -zxf`
- Run to install as a local package
  - `python setup.py install --user`

C/C++ linking (gcc/g++)

- In order to compile your OpenCL program you must tell the compiler to use the OpenCL library with the flag: `-l OpenCL`
- If the compiler cannot find the OpenCL header files (it should do) you must specify the location of the CL/ folder with the `-I` (capital “i”) flag
- If the linker cannot find the OpenCL runtime libraries (it should do) you must specify the location of the lib file with the `-L` flag
- Make sure you are using a recent enough version of gcc/g++ - at least v4.7 is required to use the OpenCL C++ API (which needs C++11 support)