PyOpenCL

蘇育生

Source by 董孝倫

PyOpenCL?

PyOpenCL = Python + OpenCL

- wrap every OpenCL API construct into a Python class
- PyOpenCL objects are garbage collected
- kernels ('the device code') are given as a string
- errors are translated into Python exceptions

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

What is OpenCL?

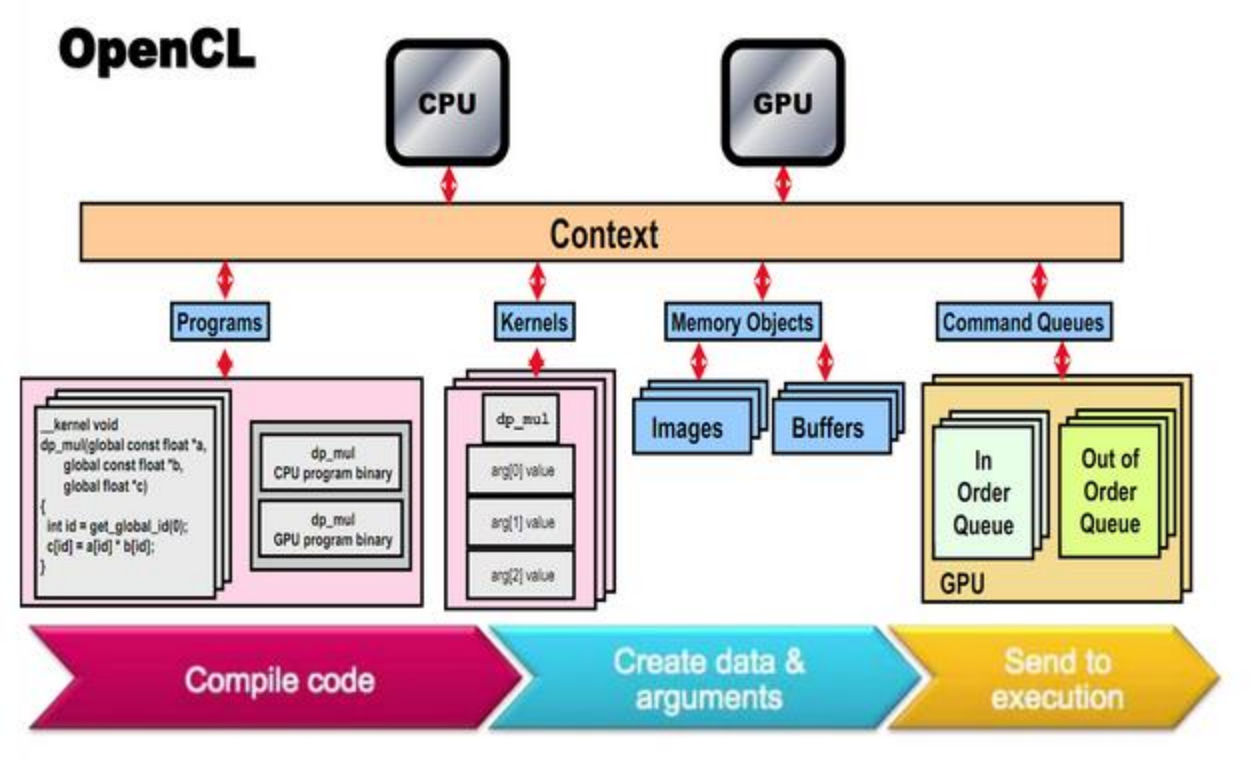
OpenCL（Open Computing Language，開放計算語言）是一個為異構平台編寫程式的框架，此異構平台可由CPU，GPU或其他類的處理器組成。

OpenCL由一門用於編寫kernels（在OpenCL裝置上執行的函式）的語言（基於C99）和一組用於定義並控制平台的API組成。

OpenCL提供了基於**任務分割**和**資料分割**的平行計算機制。

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

OpenCL架構

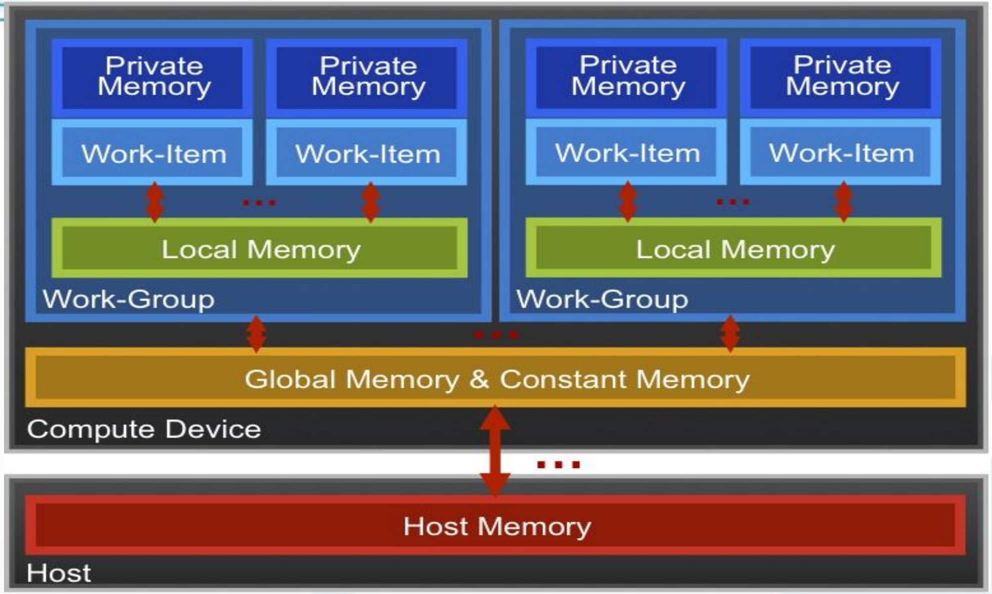


- Context：The environment within which kernels execute and in which synchronization and memory management is defined
- Programs：Collection of kernels and other functions (analogous to a dynamic library)
- Kernels： the code for a work-item (basically a C function)

高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

Memory

- Private Memory
  - Per work-item
- Local Memory
  - Shared within a work-group
- Global Memory/ Constant Memory
  - Visible to all work-groups
- Host memory
  - On the CPU

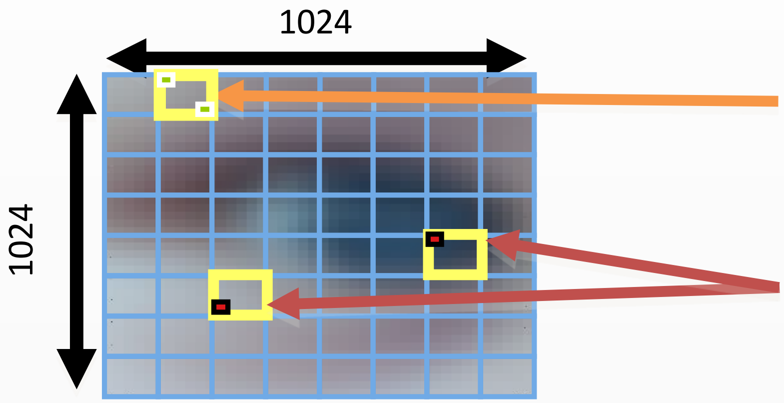


Memory management is **explicit**: You are responsible for moving data from host → global → local *and* back

高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

An N-dimensional domain of work-items

- Global Dimensions:
  - 1024x1024 (whole problem space)
- Local Dimensions:
  - 64x64 (work-group, executes together)



Synchronization between work-items possible only within work-groups:
barriers and memory fences

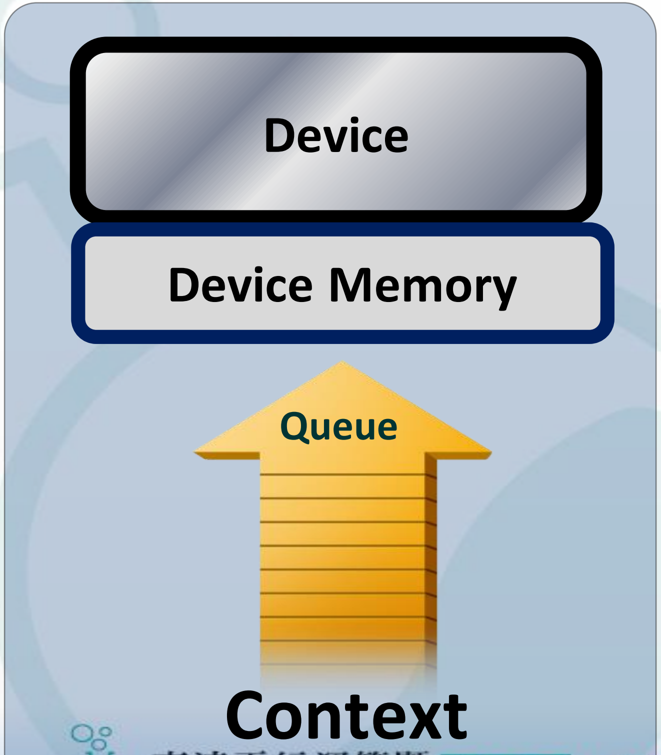
Cannot synchronize between work-groups within a kernel

- Choose the dimensions that are "best" for your algorithm

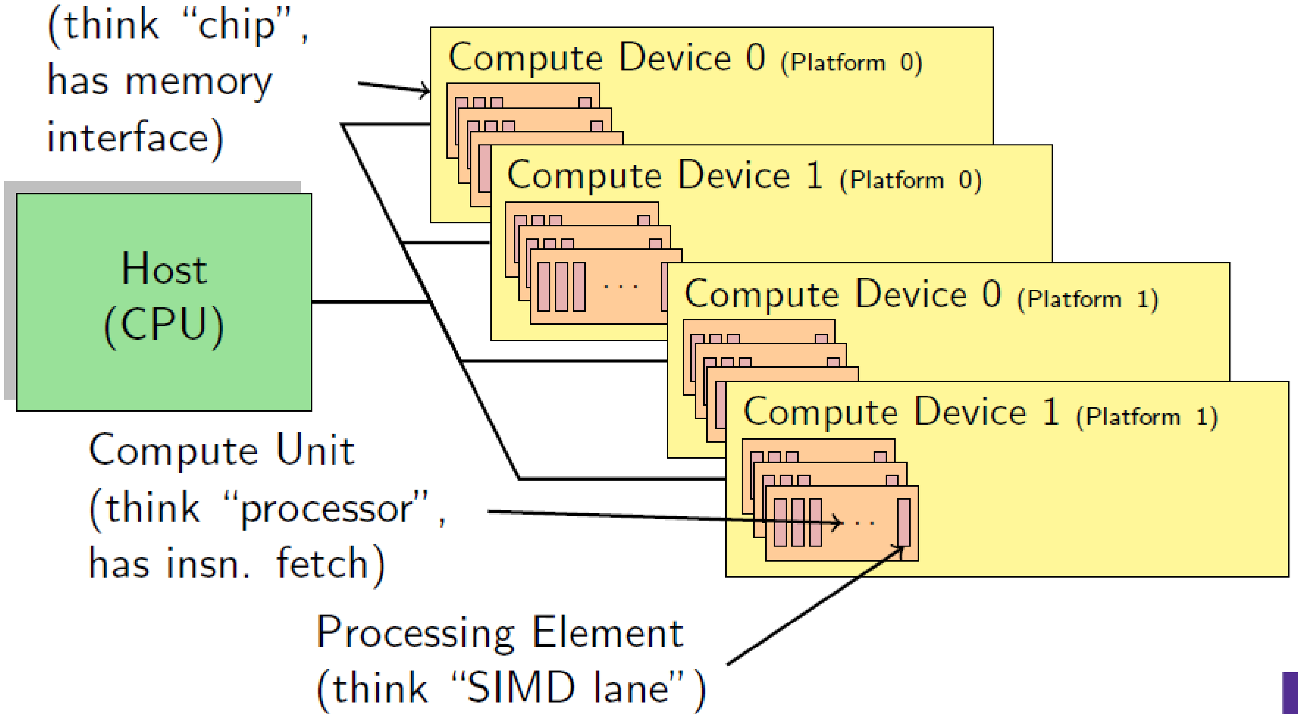
高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

Context and Command-Queues

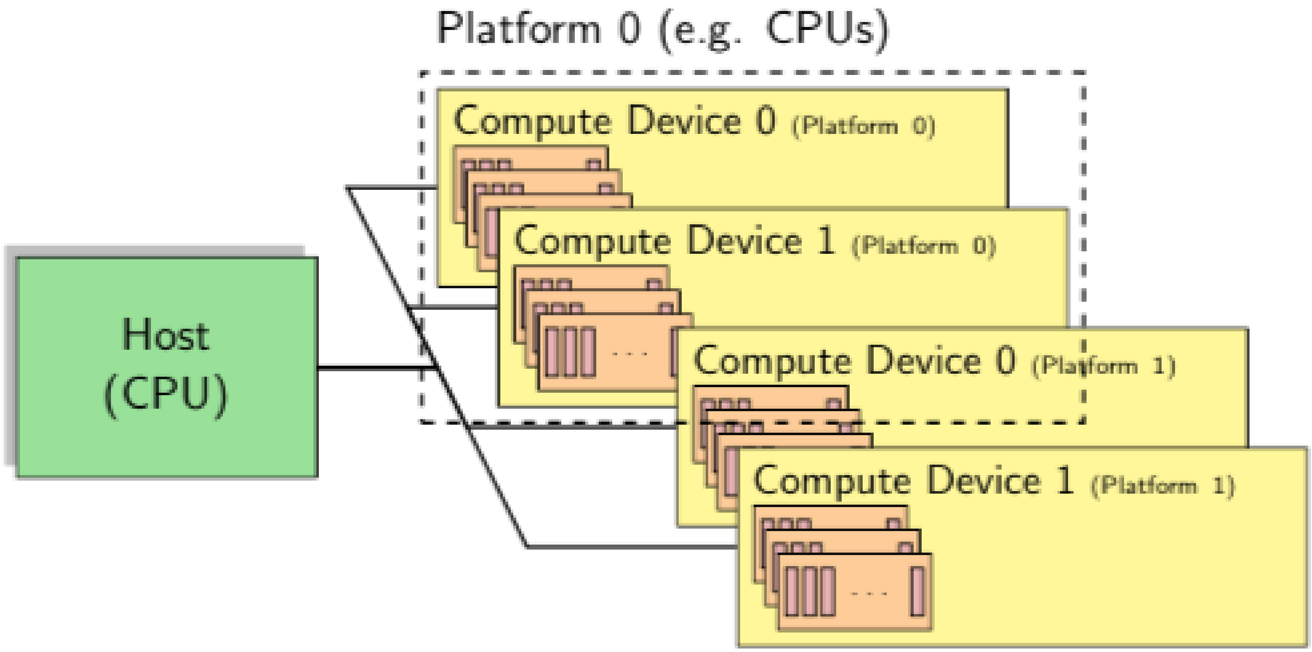
- Context:
  - The environment within which kernels execute and in which synchronization and memory management is defined.
- The context includes:
  - One or more devices
  - Device memory
  - One or more command-queues
- All commands for a device (kernel execution, synchronization, and memory transfer operations) are submitted through a command-queue.
- Each command-queue points to a single device within a context.

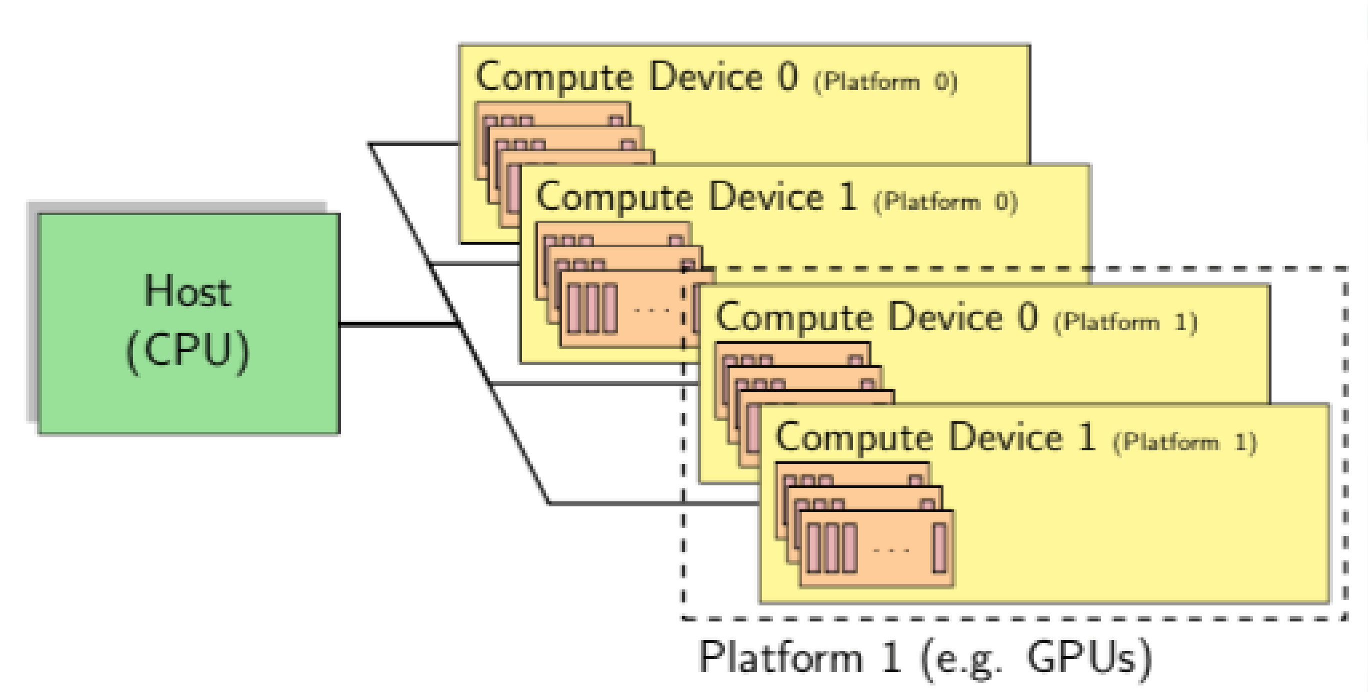


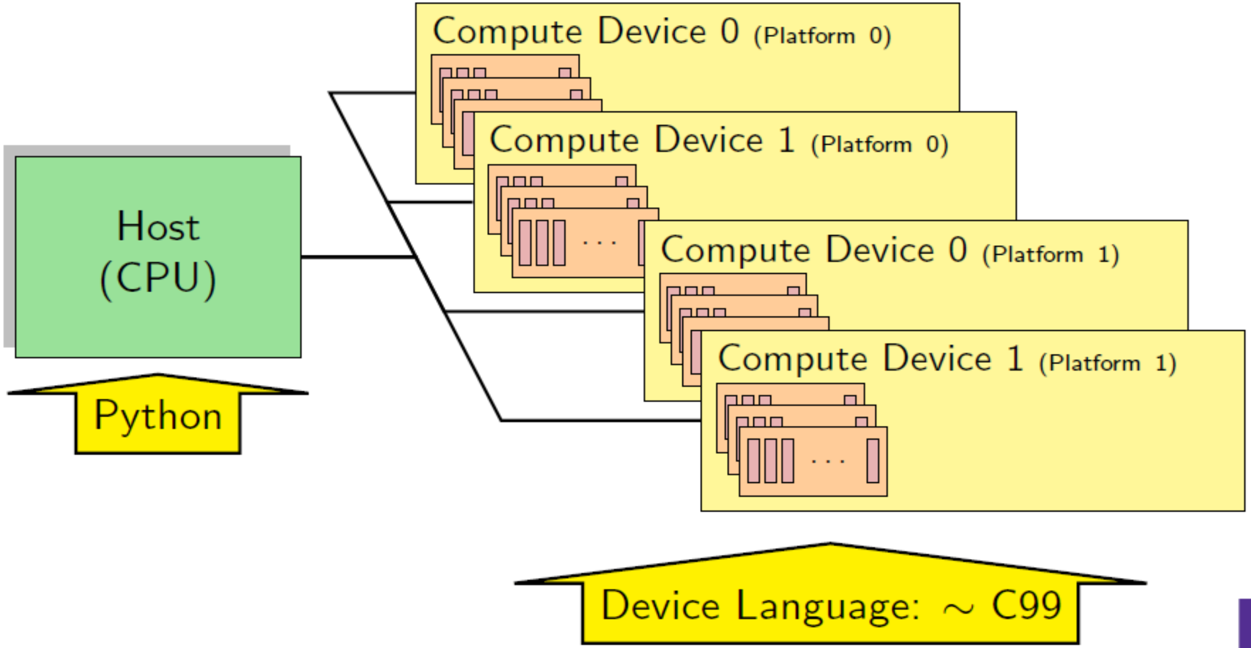
高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory



高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory


Platform 0 (e.g. CPUs)
Compute Device 0 (Platform 0)
Compute Device 1 (Platform 0)
Compute Device 0 (Platform 1)
Compute Device 1 (Platform 1)
Host
(CPU)


Compute Device 0 (Platform 0)
Compute Device 1 (Platform 0)
Host
(CPU)
Compute Device 0 (Platform 1)
Compute Device 1 (Platform 1)
Platform 1 (e.g. GPUs)



高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

How to install?

高速平行運算暨
巨量資料實驗室
BIGR LAB
big data research laboratory

環境安裝

1. 安裝python http://www.activestate.com/activepython/downloads
2. 套件安裝
http://www.lfd.uci.edu/~gohlke/pythonlibs/
a. numpy http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy
    (numpy‑1.9.2+mkl‑cp27‑none‑win_amd64.whl)
b. pyopencl http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopencl
    (pyopencl‑2015.1‑cp27‑none‑win_amd64.whl)

安裝方式：
    開啟命令提示字元，進入下載好的檔案目錄，輸入指令
    “”pip install some-package.whl “”
    Ex: numpy : pip install numpy‑1.9.2+mkl‑cp27‑none‑win_amd64.whl

高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

環境安裝

3. GPU driver
   a. NVIDIA:：
   http://www.nvidia.com.tw/Download/index.aspx?lang=tw
   b. AMD：http://support.amd.com/en-us/download

4. SDK
   a. AMD APP SDK：http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

How to start programing Vector Add?

1. create context and queue
2. alocate the memory objects and copy data from cpu to gpu
3. create and build program
4. execute kernel
5. copy data from gpu to cpu

```
import numpy as np
import pyopencl as cl
```


高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

create context and queue

```
ctx = cl.create_some_context()
queue = cl.CommandQueue(ctx)
```

```
platform = cl.get_platforms()[1]    # Select platform
device = platform.get_devices()[0]  # Select device
ctx = cl.Context([device])
```


高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

copy data from cpu to gpu

- READ_ONLY：表示 OpenCL kernel 只會對這塊記憶體進行讀取的動作
- WRITE_ONLY：表示 OpenCL kernel 只會對這塊記憶體進行寫入的動作
- READ_WRITE：表示 OpenCL kernel 會對這塊記憶體進行讀取和寫入的動作
- USE_HOST_PTR：表示希望 OpenCL 裝置直接使用指定的主記憶體位址。要注意的是，如果 OpenCL 裝置無法直接存取主記憶體，它可能會將指定的主記憶體位址的資料複製到 OpenCL 裝置上。
- ALLOC_HOST_PTR：表示希望配置的記憶體是在主記憶體中，而不是在 OpenCL 裝置上。不能和 USE_HOST_PTR 同時使用。
- COPY_HOST_PTR：將指定的主記憶體位址的資料，複製到配置好的記憶體中。不能和 USE_HOST_PTR 同時使用。

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

Allocate the memory objects and copy data from cpu to gpu

```
a_np = np.random.rand(50000).astype(np.float32)
b_np = np.random.rand(50000).astype(np.float32)


mf = cl.mem_flags
a_g = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=a_np)
b_g = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=b_np)


res_np = np.empty_like(a_np)
res_g = cl.Buffer(ctx, mf.WRITE_ONLY, a_np.nbytes)
```

create and build program

```
prg = cl.Program(ctx, """
__kernel void sum(__global const float *a_g, __global
const float *b_g, __global float *res_g) {
  int gid = get_global_id(0);
  res_g[gid] = a_g[gid] + b_g[gid];
}
""").build()
```


高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

execute kernel

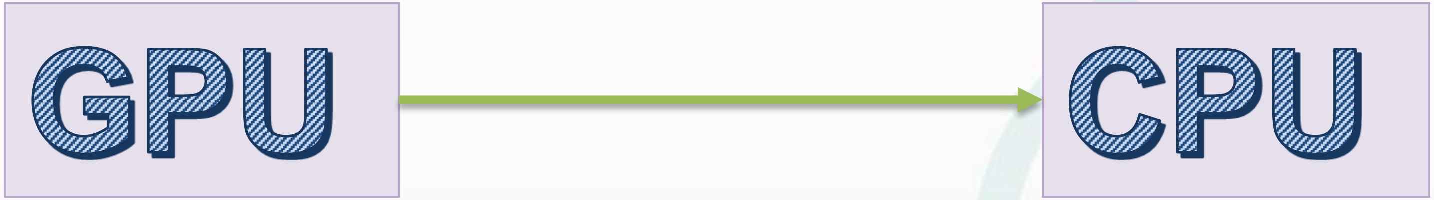
```
prg.sum(queue, a_np.shape, None, a_g, b_g, res_g)
```

Runing...

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

copy data from gpu to cpu

```
cl.enqueue_copy(queue, res_np, res_g)
```



Example Vector Add

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import pyopencl as cl

a_np = np.random.rand(50000).astype(np.float32)
b_np = np.random.rand(50000).astype(np.float32)

ctx = cl.create_some_context()
queue = cl.CommandQueue(ctx)

mf = cl.mem_flags
a_g = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=a_np)
b_g = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=b_np)

prg = cl.Program(ctx, """
__kernel void sum(__global const float *a_g, __global const float *b_g, __global float *res_g) {
  int gid = get_global_id(0);
  res_g[gid] = a_g[gid] + b_g[gid];
}
""").build()

res_g = cl.Buffer(ctx, mf.WRITE_ONLY, a_np.nbytes)
prg.sum(queue, a_np.shape, None, a_g, b_g, res_g)

res_np = np.empty_like(a_np)
cl.enqueue_copy(queue, res_np, res_g)

# Check on CPU with Numpy:
print(res_np - (a_np + b_np))
print(np.linalg.norm(res_np - (a_np + b_np)))
```

型態對應

numpy type	C type
int32	int
float32	float

高速平行運算暨巨量資料實驗室 BIGRLAB big data research laboratory

OpenCL CUDA Dictionary

OpenCL	CUDA
Grid	Grid
Work Group	Block
Work Item	Thread
__kernel	__global__
__global	__device__
__local	__shareed__
__private	__local
imagend_t	texture<type, n, ...>
barrier(LMF)	__syncthreads()
get_local_id(012)	hreadIdx.xyz
get_group_id(012)	blockIdx.xyz
get global_id(012)	– (reimplement)

References

1. opencl維基 http://zh.wikipedia.org/wiki/OpenCL
2. http://www.training.prace-ri.eu/uploads/tx_pracetmo/LinkSCEMM_pyOpenCL.pdf
3. http://www.kimicat.com/opencl-1/opencl-jiao-xue-yi
4. PyOpenCL's documentation http://documen.tician.de/pyopencl/

高速平行運算暨巨量資料實驗室 BIGR LAB big data research laboratory

We help you to understand new technologies and trends!