Séminaire GPU

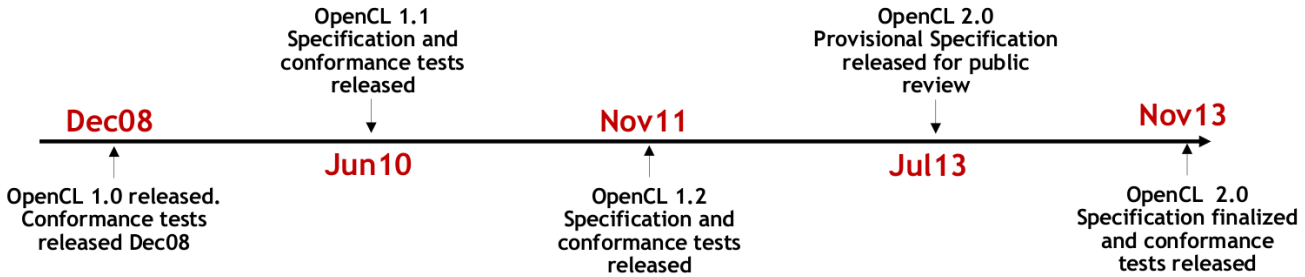
OpenCL

Eric Lombardi, LIRIS

Plan

Origine

- besoin d'un standard pour exploiter les plate-formes hétérogènes (CPU, GPU, DSP, FPGA)
- initié par Apple, standardisé par Khronos Group (OpenGL) en Décembre 2008
- historique



- implémenté par AMD, Intel, Nvidia ... avec plus ou moins de décalage dans le temps (support OpenCL 1.2 par Nvidia en 2015)

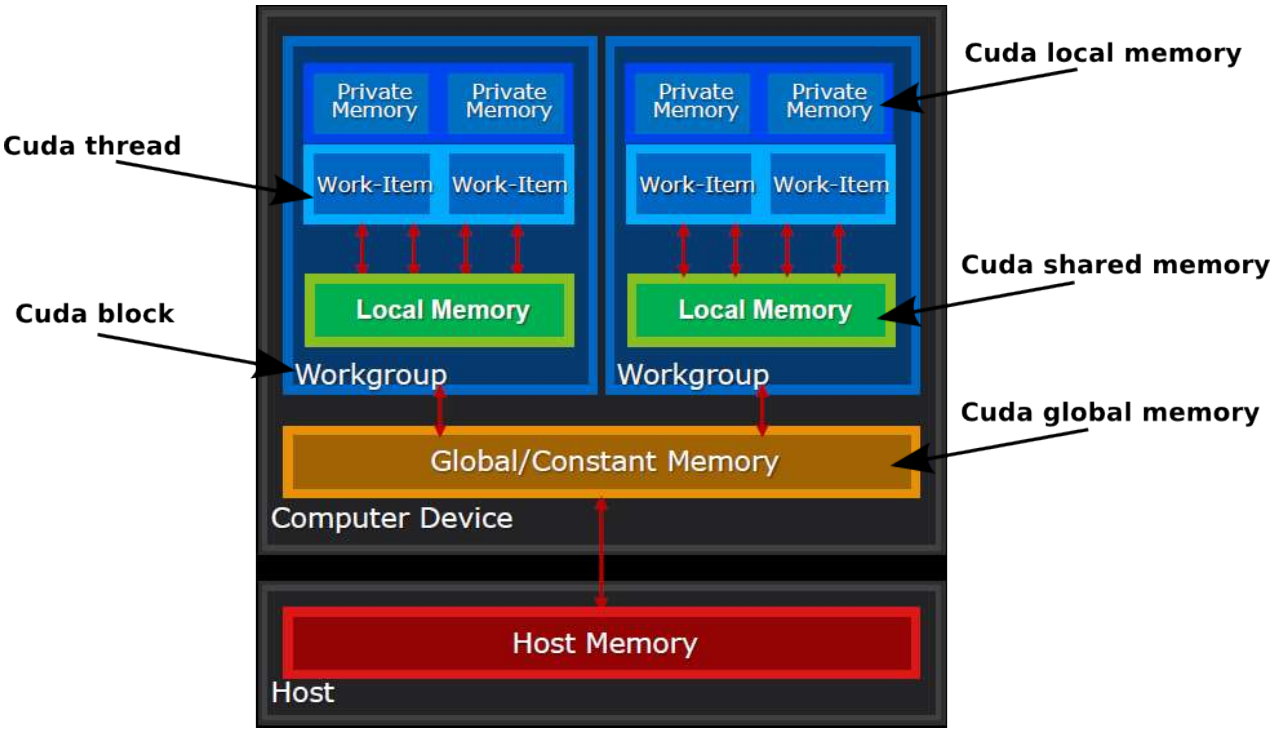
Comparaison avec CUDA

OpenCL	CUDA
Expression du parallélisme (fonction de calcul pour une valeur, grille de calcul)	
API -> compilateur classique (gcc)	Extension de langage -> compilateur dédié (nvcc)
Kernel compilé à l'exécution	Kernel compilé avec le programme principal
Multi-plateforme (GPU, CPU, DSP, FPGA)	Spécifique GPU Nvidia

Expression du parallélisme

OpenCL	CUDA
Work-item	Thread
Work-group	Thread block
Processing Element	Cuda core
Compute Unit	Streaming Multiprocessor

Modèle de mémoire



Définition de kernel

```
__kernel void vectadd(
    __global const float *a,
    __global const float *b,
    __global float *result)
{
    int gidx = get_global_id(0);
            // id dans la grille
    int lidx = get_local_id(0);
            // id dans le work group
    int gszx = get_global_size(0);
            // taille de la grille
    int lszx = get_local_size(0);
            // taille du work group
    ...
}
```

Thread index ...

OpenCL	CUDA
get_local_id(0)	threadIdx.x
get_local_id(1)	threadIdx.y
get_local_id(2)	threadIdx.z
get_global_id(0)	blockIdx.x*blockDim.x + threadIdx.x

API OpenCL : initialisation

```
clGetPlatformIDs(...);
    // detecte les platformes
    // (implémentation OpenCL
    // Intel/AMD/Nvidia)
clGetDeviceIDs(platform_id, ...);
    // detecte les devices
    // (CPU/GPU, GPUs)

clCreateContext(..., &device_id, ...);
clCreateCommandQueue(context, ...);
```

API OpenCL : compilation du kernel

```
program = clCreateProgramWithSource(
    context, ..., kernelSource, ...);
    // associe contexte et source
clBuildProgram(program, ..., options, ...);
    // compile et link le programme
clGetProgramBuildInfo(program, ...);
    // erreurs de compilation
cl_kernel kernel = clCreateKernel(
    program, kernel_name, ...);
    // un programme peut contenir
    // plusieurs kernels
clReleaseKernel(kernel);
clReleaseProgram(program);
    // libère les ressources
```

API OpenCL : allocation de mémoire

```
cl_mem gpuBuffer
    = clCreateBuffer(context, ...);
    // alloue la mémoire globale du GPU
    // depuis l'hôte
clReleaseMemObject(gpuBuffer);
    // libère la mémoire

clSetKernelArg(..., nb_bytes, NULL);
    // alloue dynamiquement la mémoire
    // locale du GPU depuis l'hôte
__local float foo[256];
    // alloue statiquement la mémoire
    // locale du GPU depuis le kernel
float bar[256];
    // alloue la mémoire privée du GPU
    // depuis le kernel
```

API OpenCL : transfert de données

```
clEnqueueWriteBuffer(queue, ...);
    // copie de l'hôte vers la mémoire
    // globale du GPU
clEnqueueReadBuffer(queue, ...);
    // copie de la mémoire globale
    // du GPU vers l'hôte
// les copies peuvent être bloquantes ou
// non-bloquantes
```

API OpenCL : exécution de kernel

```
clSetKernelArg(kernel, 0, sizeof(cl_mem),
    &gpuBuffer);
    // passe en argument l'adresse d'un
    // buffer en mémoire globale
clSetKernelArg(kernel, 1, nb_bytes, NULL);
    // si le paramètre est du type __local
    // nb_bytes sont alloués en mémoire locale

clEnqueueNDRangeKernel(queue, kernel, ...
    globalWorkSize, localWorkSize);
    // exécute le kernel, non-bloquant
clFinish(queue);
    // attend la fin du kernel
```

API OpenCL : interface C++

- mince surcouche C++ à l'API C
- gère les exceptions du C++
  - -> centralisation des erreurs
  - -> code plus compact et lisible
- fichier cl.hpp à récupérer sur le site de Khronos (doit correspondre à la version d'OpenCL installée)
- #include <cl.hpp> dans le programme principal

Exemple 1 : hello world (kernel)

```
__kernel void square(
        __global float* input,
        __global float* output,
        const unsigned int count)
{
   int i = get_global_id(0);

   if(i < count)
           output[i] = input[i] * input[i];
}
```

Exemple 1 : hello world (main)

```
std::vector<cl::Platform> platforms;
cl::Platform::get(&platforms);
cl::Platform platform = platforms[0];

std::vector<cl::Device> devices;
platform.getDevices(CL_DEVICE_TYPE_GPU, &devices);
cl::Device device = devices[0];

cl::Context context(devices);
cl::CommandQueue queue(context, device);

cl::Program::Sources sources;
sources.push_back(std::make_pair(kernel_source.c_str(),
        kernel_source.length()));
cl::Program program(context, sources);
program.build(devices);
```

```
cl::Kernel kernel(program, "square");

cl::Buffer gpu_input(context, CL_MEM_READ_ONLY,
        sizeof(float)*count);
cl::Buffer gpu_output(context, CL_MEM_WRITE_ONLY,
        sizeof(float)*count);

queue.enqueueWriteBuffer(gpu_input, CL_TRUE, 0,
        sizeof(float)*count, data);

kernel.setArg(0, gpu_input);
kernel.setArg(1, gpu_output);
kernel.setArg(2, count);

size_t work_group_size = ...;
size_t global_work_size = ...;
queue.enqueueNDRangeKernel(kernel, cl::NullRange,
```

```
        cl::NDRange(global_work_size),
        cl::NDRange(work_group_size));
queue.finish();

queue.enqueueReadBuffer(gpu_output, CL_TRUE, 0,
        sizeof(float)*count, results);
```

Compilation :

```
g++ hello_world_opencl.cpp -lOpenCL
```

Exemple 2 : SpMV-CSR (kernel)

```
__global__ void spmvCSR(int rNbr, float *values,
    int *col_ind, int *row_ptr, float *v, float *y)
{
    int r = blockIdx.x * blockDim.x + threadIdx.x;
    if( r < rNbr )
    {
        float dot = 0.0f;
        int row_beg = row_ptr[r];
        int row_end = row_ptr[r+1];
        for(int i = row_beg; i < row_end; i++)
            dot += values[i] * v[col_ind[i]];
        y[r] = dot;
    }
}
```

A vous de coder en OpenCL ...

Exemple 2 : SpMV-CSR

*** Solution de l'exercice ***

Exemple 3 : SpMV-CSRVect

- reprendre le code CUDA du kernel SpMV-CSR-Vect
- allouer la mémoire locale en OpenCL

```
// dans le kernel
__global__ void kernelFn(..., __local float *y)
// dans le programme principal
clSetKernelArg(kernel, 3, nb_bytes, NULL);
// ou
kernel.setArg(3, nb_bytes, NULL);
```

A vous de coder en OpenCL ...

Exemple 3 : SpMV-CSRVect

*** Solution de l'exercice ***

Cuda ou OpenCL ?

Comment choisir entre Cuda et OpenCL ?

	OpenCL	Cuda
GPU Nvidia		X
GPU AMD ou cpu	X	
Android	X ?	
iOS	X	
besoin d'une librairie existante		X