Introduction to OpenCL

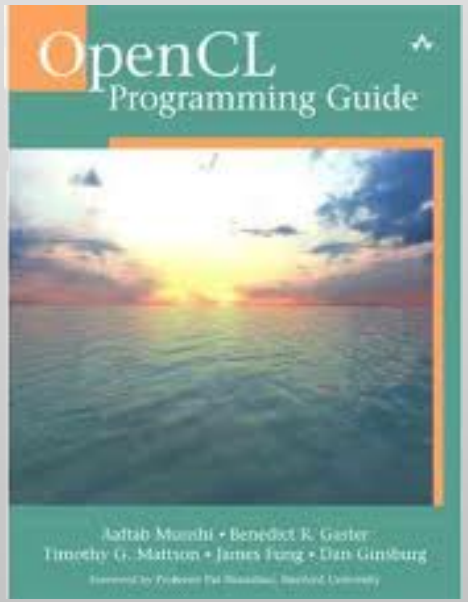
Paul Springer

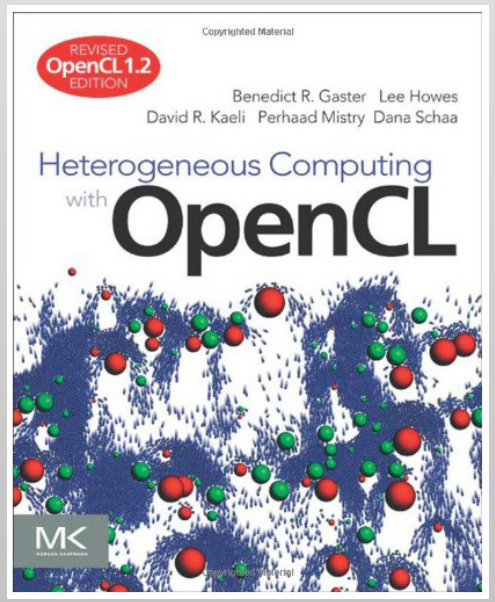
Aachen Institute for Advanced Study in
Computational Engineering Science

Aachen, 24.05.13

Resources

- http://www.khronos.org/opencl/
- NVIDIA CUDA 4.2 SDK
- https://developer.nvidia.com/gpu-computing-webinars
- Intel OpenCL documentation
  - Users Guide
  - Optimization Guide





- Open standard for heterogeneous systems
- Developed by AMD, IBM, Intel, NVIDIA and Apple
- Maintained by Khronos Group
- Available for ...
  - ...NVIDIA/ AMD GPUs
  - ...Intel/ AMD CPUs
  - ...Intel Xeon Phi
  - ...Altera's FPGAs
  - ...

- Single Program Multiple Data (SPMD)
  - The program (*kernel*) is executed multiple times
  - A running instance of a kernel is called *work-item*
- Execution Model
  - Index space
  - Work-group
  - Work-item

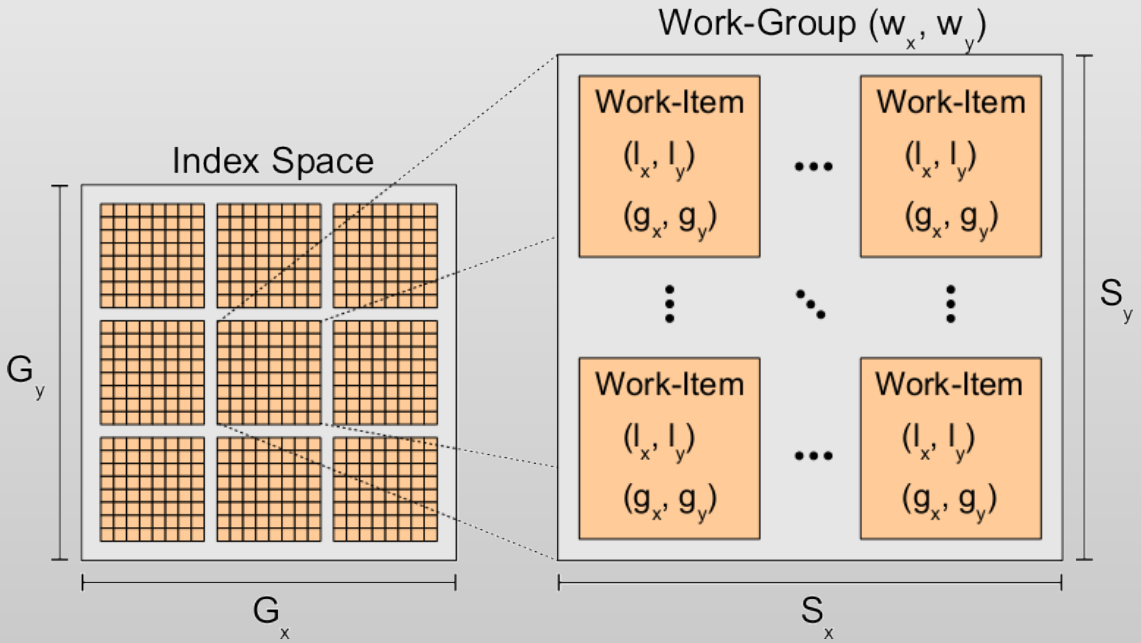


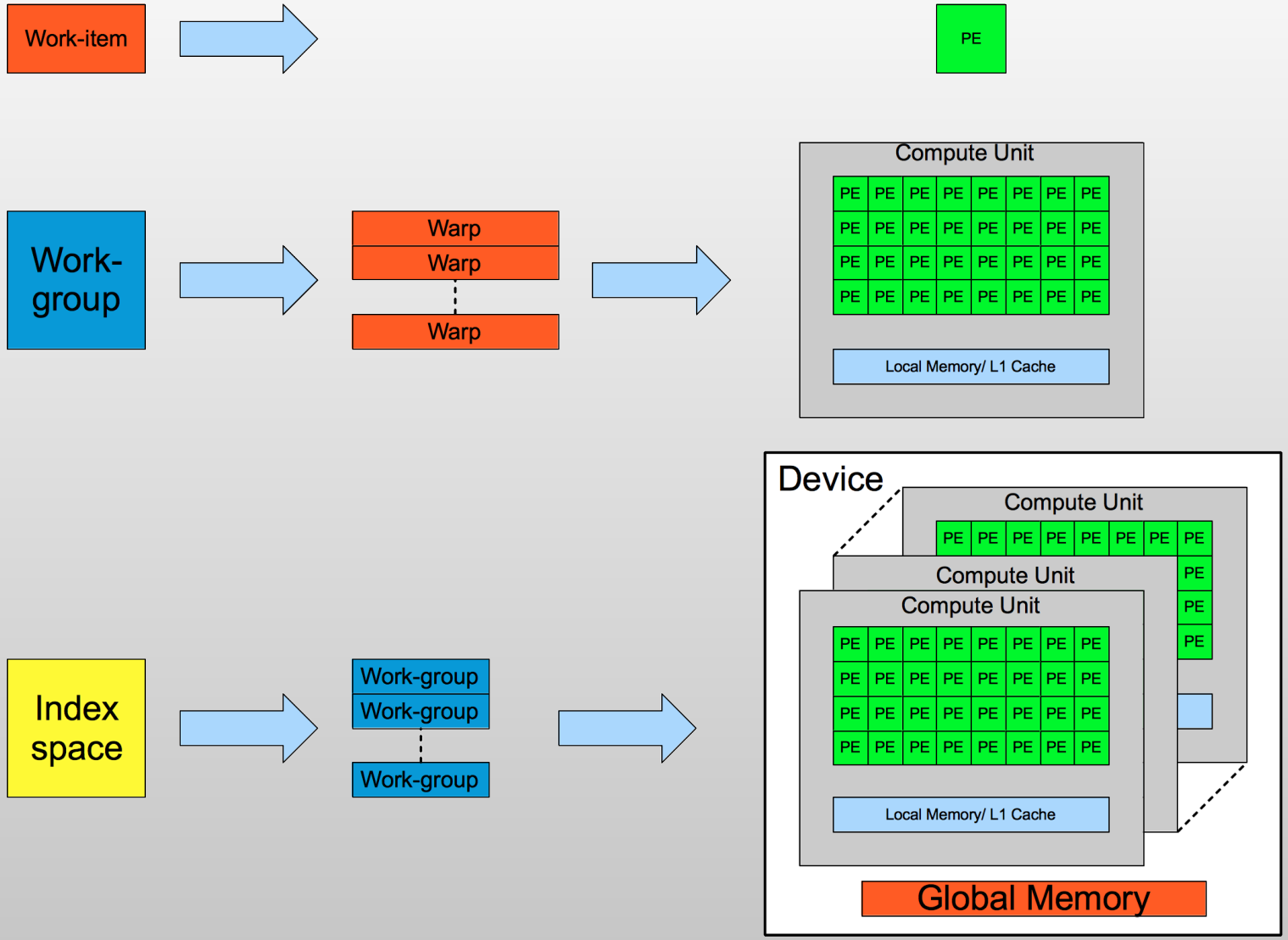
Figure: OpenCL Execution Model.

Terminology

CUDA	OpenCL
Multiprocessor	Compute Unit
Scalar processor	Processing element
Global memory	Global memory
Shared memory	Local memory
Local memory	Private memory
Kernel	Program
Grid	Index space
Threadblock	Work-group
Thread	Work-item

AMD, "Porting CUDA Applications to OpenCL" http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/programming-in-opencl/porting-cuda-applications-to-opencl/

OpenCL Execution Model



Qualifiers

CUDA	OpenCL
__global__	__kernel
__shared__	__local

AMD, "Porting CUDA Applications to OpenCL" http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/programming-in-opencl/porting-cuda-applications-to-opencl/

Indexing

CUDA	OpenCL
dimGrid	get_num_groups()
blockDim	get_local_size()
blockIdx	get_group_id()
threadIdx	get_local_id()
	get_global_id()
	get_global_size()

AMD, "Porting CUDA Applications to OpenCL" http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/programming-in-opencl/porting-cuda-applications-to-opencl/

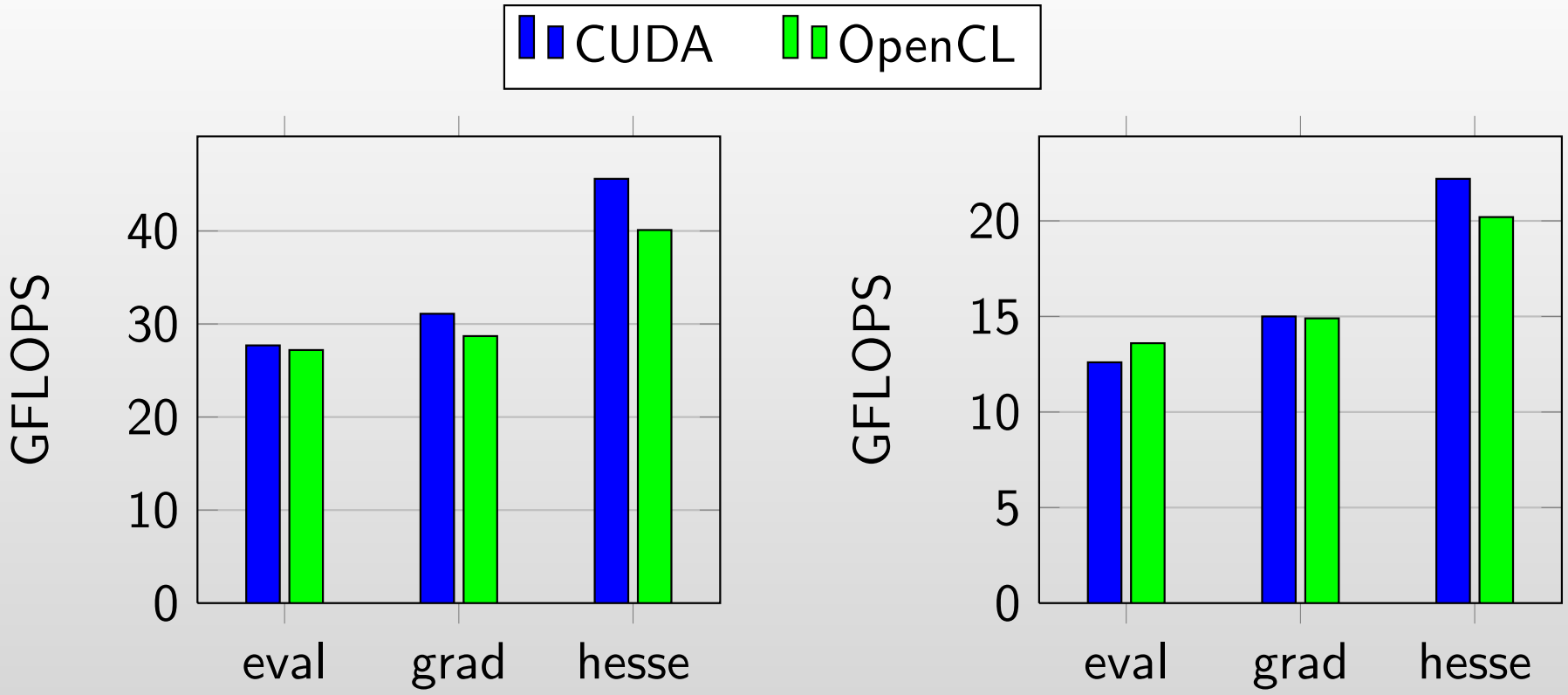
Example API Calls

CUDA	OpenCL
cudaMalloc()	clCreateBuffer()
cudaFree()	clReleaseMemObject()
cudaMemcpy()	clEnqueueReadBuffer()
cudaMemcpy()	clEnqueueWriteBuffer()
__syncthreads()	barrier()
<<<dimGrid,dimBlock>>>	clEnqueueNDRangeKernel()

AMD, "Porting CUDA Applications to OpenCL" http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/programming-in-opencl/porting-cuda-applications-to-opencl/

1. Create the computing context
   1. Choose the compute device
   2. Compile the kernel for that device
   3. Create kernel objects
   4. Create a command queue[1]
2. Transfer the data to the device
3. Initialize kernel arguments
4. Launch the kernel
5. Transfer data back to the host

[1] each CD can have multiple command queues



(a) Single Precision

(b) Double Precision

Figure: GFLOPS of some computational kernels for CUDA and OpenCL running on an NVIDIA Quadro 6000 GPU.