Computational Science on Many-Core Architectures

360.252

Karl Rupp

Institute for Microelectronics
Vienna University of Technology
http://www.iue.tuwien.ac.at

Zoom Channel 95028746244
Wednesday, December 2, 2020

Agenda for Today

Exercise 6 Recap

OpenCL

Exercise 7

Freakin’ Fast Friday

Exercise 6 Recap

Feedback Time

- How was your experience?

Exercise 6 Recap

Feedback Time

- How was your experience?
- Points for Exercise 5 will be provided within 24 hours.

History of OpenCL

2008

- OpenCL working group formed at Khronos Group
- OpenCL specification 1.0 released

2010

- OpenCL 1.1 (multi-device, subbuffer)

2011

- OpenCL 1.2 (device partitioning)

2013

- OpenCL 2.0 (shared virtual memory, SPIR, etc.)

2020

- OpenCL 3.0 (back to the roots)

OpenCL

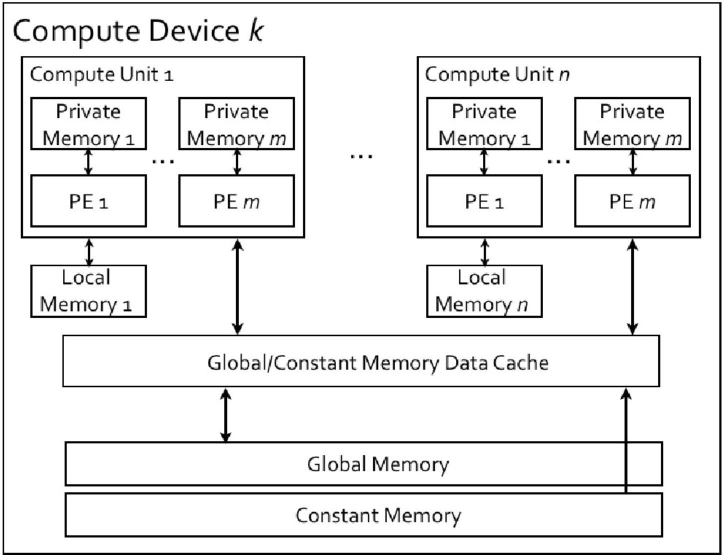
Similar to CUDA

- Kernel language is a subset of C
- Explicit memory management, host-device transfers
- Memory model: local, shared, global

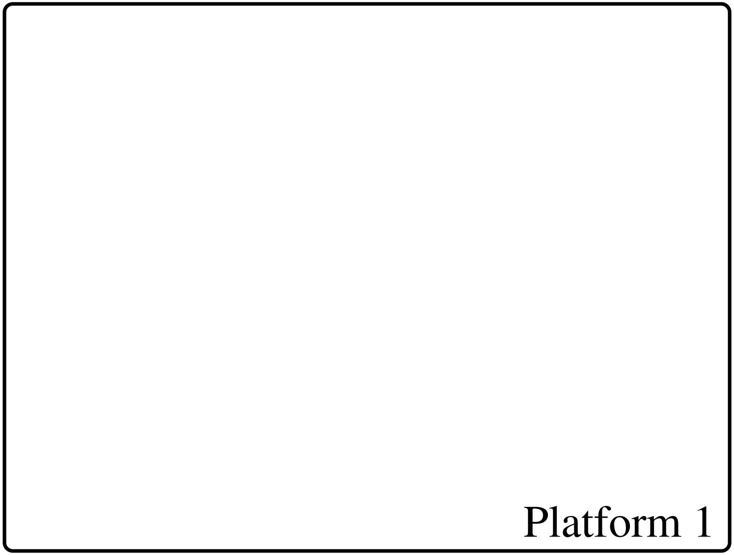
Different from CUDA

- Support by many vendors
- No compiler-wrapper, only a shared library
- Kernel compilation usually at runtime

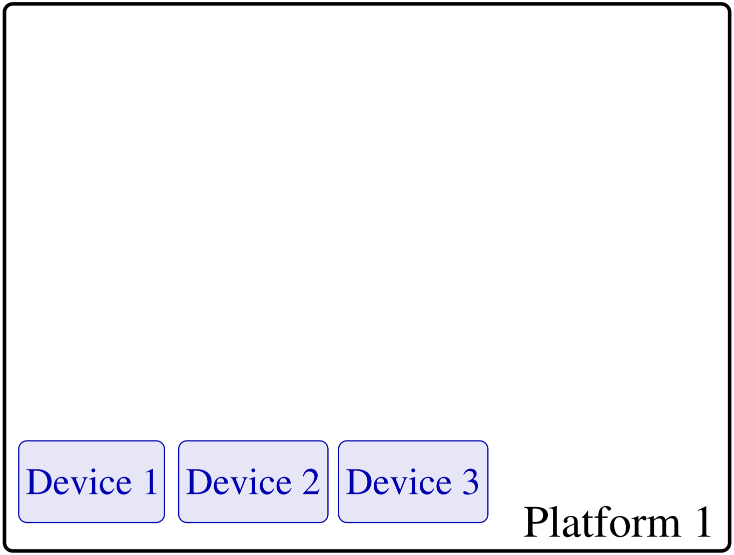
OpenCL Platform Model



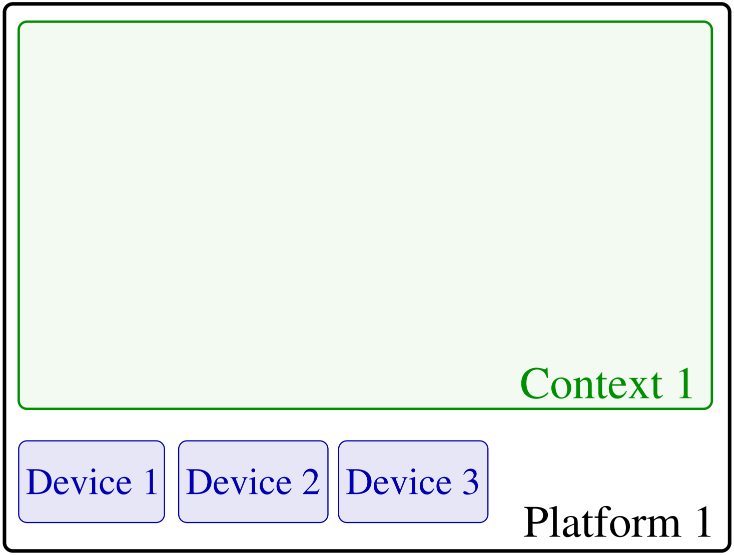
OpenCL Platform Model



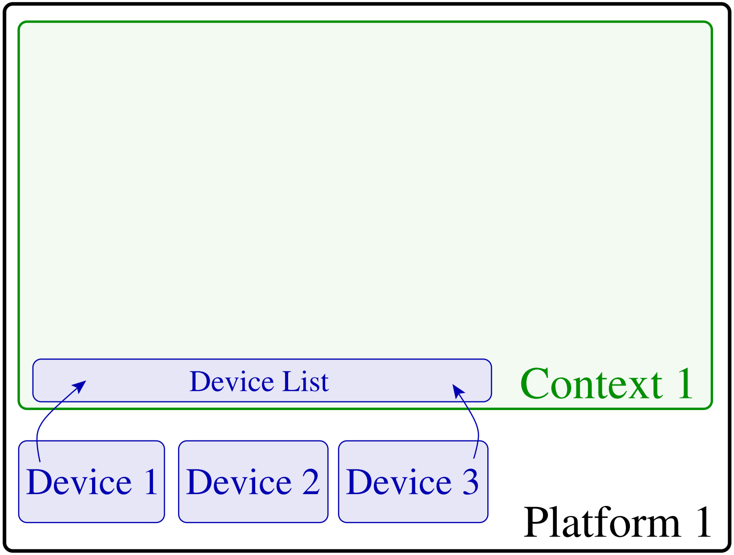
OpenCL Platform Model



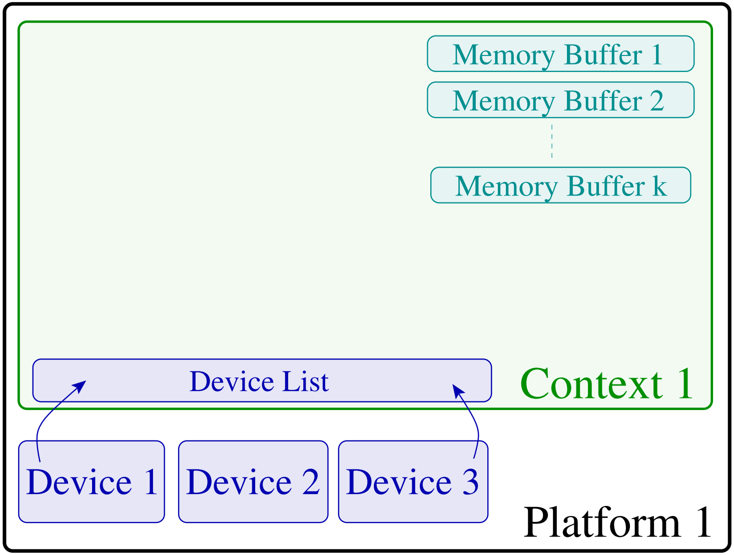
OpenCL Platform Model



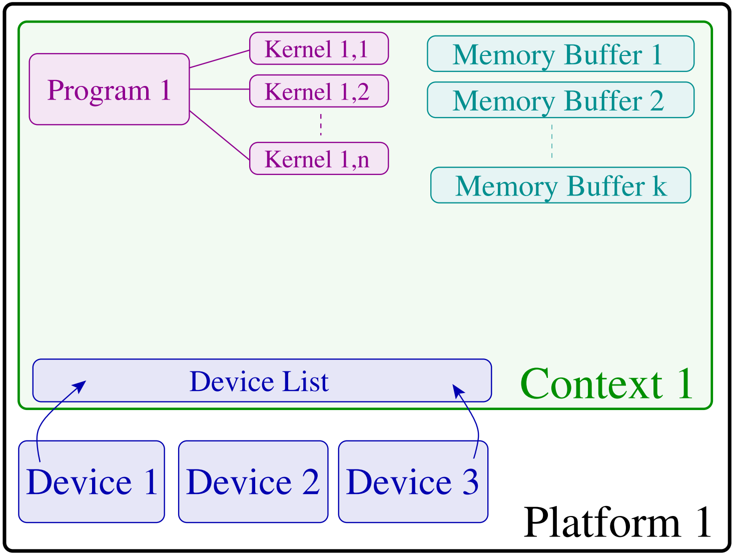
OpenCL Platform Model



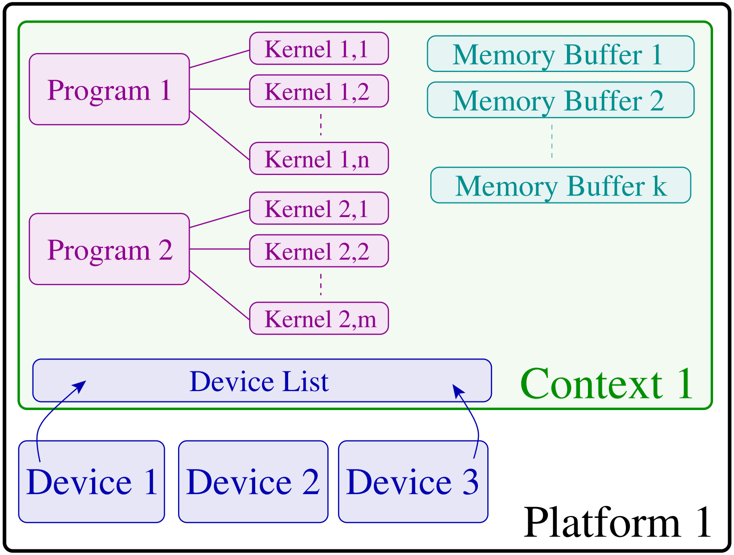
OpenCL Platform Model



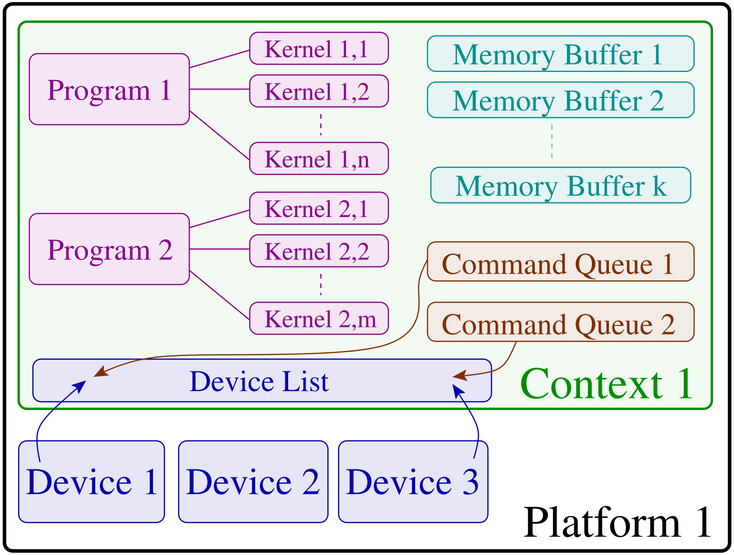
OpenCL Platform Model



OpenCL Platform Model



OpenCL Platform Model



OpenCL

OpenCL Thread Control (1D) vs. CUDA

Local ID in block: `get_local_id(0)`	`threadIdx.x`
Threads per block: `get_local_size(0)`	`blockDim.x`
ID of block: `get_group_id()`	`blockIdx.x`
No. of blocks: `get_num_groups()`	`gridDim.x`
Global thread ID: `get_global_id()`	
No. of threads: `get_global_size()`	
Barrier: `barrier(CLK_GLOBAL_MEM_FENCE)`	`__syncthreads()`

OpenCL Example

```
// Multiplies A*x, leaving the result in y.
// A is a row-major matrix,
// meaning the (i,j) element is at A[i*ncols+j].
__kernel void matvec(__global const float *A,
                     __global const float *x,
                     uint ncols, __global float *y)
{
  size_t i = get_global_id(0);      // Global id, used as the row
      index
  __global float const *a = &A[i*ncols];// Pointer to the i-th row
  float sum = 0.f;                  // Accumulator for dot product
  for (size_t j = 0; j < ncols; j++) {
     sum += a[j] * x[j];
  }
  y[i] = sum;
}
```

Source: https://en.wikipedia.org/wiki/OpenCL

Exercises

Environment

- `https://gtx1080.360252.org/2020/ex7/`
- (Might receive visual updates and additional hints over the next days)
- Due: Tuesday, December 8, 2020 at 23:59pm

Hints and Suggestions

- Consider version control for locally developed code
- Please let me know of any bugs or issues

Freakin’ Fast Friday

Opportunity for Informal Chatting

- When? Friday, December 4, 17:00-18:00
- Where? ~~Wieden Bräu~~ This Zoom channel
- What? Preserving mental sanity during Lockdown

Hints and Suggestions

- Consider bringing a drink
- Will not change your course evaluation
- **Completely optional and no obligation to show up**