OpenCL lab time

SPD course 2013-14

Massimo Coppola

27/05/2014

Exercise 1

- Start up your environment
- Write down a simple program computing the square root of all elements in an array
  - Insert proper includes and code for compilation
  - Detect your device type
  - Allocate an array
    - smaller than your GPUs memory! Say, 1M-32M floats?
  - Write a Kernel that squares the elements
  - Use the command queue
    - Transfer the array to the device
    - Fire the kernel repeatedly (parameter)
    - Transfer back the results
  - Measure performance
    - Play with the parameters: workgroup size / local work size, number of repeats

Exercise 2

- Compute K-means with OpenCL
  - Start up from the sequential version of the code used with MPI
  - Add init code and device initialization
    - Check that the array of data will fit on you GPU
  - Is the type of the array supported? Can you reuse sequential code as it is?
    - You may need to reorganize the data, change the types
  - Choose the core function of K-means and turn it into a kernel
    - You will need to transfer the data to the GPU
    - Are all transfers needed? Can you devise a way to reduce the number of transfers host – device and back?

Example 2 – continued

- First step will likely have *reduced* performance
- Choose the set of functions to turn into kernels
  - Operations which are executed often
- Optimizations for OpenCL
  - Reduce data transfers
    - Parallelization with MPI moves very little data, can you better with OpenCL?
  - Rearrange data structures
    - Exploit native types for faster operations
    - Separate arrays of structures into distinct arrays of simpler types
  - Move most of the computation to the GPU side
    - Host will need to check for continuation… or not?
    - Loop unrolling: Can you imagine to apply something similar?
  - Reduce the number of separate kernels
    - Merge kernels to reduce operations
    - What are the tradeoffs ?