oL9 - OpenCL

Parallel programming with OpenCL

Objectives

- Learn parallel programming with OpenCL.
- Know what (not) to expect from parallel programming.
- Understand heavy multithreading and how it is mapped to the hardware.
- Measure OpenCL code performance, locate and solve bottlenecks.
- Write efficient OpenCL code.

Exercises will be run on multi-core CPUs with nVidia GPU running under Linux.

Pre-requisites

- Good knowledge of the C language

Course Environment

- Theoretical course
  - PDF course material (in English).
  - Course dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One Online Linux PC per trainee for the practical activities.
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Eclipse environment and GCC compiler.
  - QEMU Emulated board or physical board connected to the online PC (depending on the course).
  - Some Labs may be completed between sessions and are checked by the trainer on the next session.
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Duration

- Total: 20 hours
- 4 sessions, 5 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.

- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Session

Introduction to OpenCL

- History
  - OpenCL 1.2
  - OpenCL 2.2
  - OpenCP/EP (Embedded Profile)
- Design goals of OpenCL
  - CPUs, GPUs and GPGPUs
  - Data-parallel and Task-parallel
  - Hardware related and portable
- Terminology
  - Host / Device
  - Memory model
  - Execution Model

The OpenCL Architecture

- The OpenCL Architecture
  - Platform Model
  - Execution Model
  - Memory Model
  - Programming Model
- The OpenCL Software Stack
- Example

Exercise: *Installation and test of the OpenCL SDK*

Second Session

The OpenCL Host API

- Platform layer
  - Querying and selecting devices
  - Managing compute devices
  - Managing computing contexts and queues
  - The host objects: program, kernel, buffer, image

Exercise: *Write a platform discovery and analysis program (displaying CPUs, GPUs, versions...)*

- Runtime
  - Managing resources
  - Managing memory domains
  - Executing compute kernels

Exercise: *Write an image loader program, transferring image to/from compute devices*

- Compiler
  - The OpenCL C programming language

  - Online compilation
  - Offline compilation

The Basic OpenCL Execution Model

- How code is executed on hardware
  - Compute kernel
  - Compute program
  - Application queues
- OpenCL Data-parallel execution
  - N-dimensional computation domains
  - Work-items and work-groups
  - Synchronization and communication in a work-group
  - Mapping global work size to work-groups
  - Parallel execution of work-groups

Exercise: *Compile and execute a program to square an array on the platform computing nodes*

Third Session

The OpenCL Programming Language

- Restrictions from C99
- Data types
  - Scalar
  - Vector
  - Structs and pointers
  - Type-conversion functions
  - Image types

Exercise: *Rewrite the square program to use vector operations*

- Required built-in functions
  - Work-item functions
  - Math and relational
  - Input/output
  - Geometric functions
  - Synchronization
- Optional features
  - Atomics
  - Rounding modes

Exercise: *Write and execute an image manipulation program (Blur filter)*

Fourth Session

Advanced OpenCL Execution modes

- Profiling

Exercise: *Enhance the image manipulation program to measure kernel computation time*

- The OpenCL Memory Model
  - Global Memory
  - Local Memory
  - Private Memory
- OpenCL Task-parallel execution
  - Optional OpenCL feature
  - Native work-items

Exercise: *Simulate the N-Body problem, displaying data using OpenGL*

Efficient OpenCL

- When (not) to use OpenCL

- Code design guidelines
- Explicit vectorization

Exercise: *Explore vectorisation on an image rotation kernel*

- Memory latency and access patterns
  - ALU latency
  - Using local memory

Exercise: *Enhance the Blur filter program to investigate memory optimisations*

- Synchronizing threads
- Warps/Wavefronts, work groups, and GPU cores

Renseignements pratiques

Inquiry : 20 hours