ac6

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.

Depending on the hardware environment, exercises will be run on either multi-core CPUs, nVidia or ATI GPUs.

Course environment

- One PC under Windows for two trainees, with either
  - Intel OpenCL SDK (needs a recent CPU, core i3 or better, and Windows 7)
  - nVidia SDK (needs a recent workstation-class nVidia graphic interface)
  - ATI SDK (needs a recent workstation-class ATI graphic interface)

Exercise : *For on-site training sessions, contact us to check the needed configuration for PC used during hands-on labs.*

Pre-requisites

- Good knowledge of the C language

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours.
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions
  - Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
  - Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.

- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
  - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
  - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
  - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

Plan

First day

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*

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

Second day

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*

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)*

Third day

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

Renseignements : 3 jours