Portable Mapping of Data Parallel Programs
to OpenCL for Heterogeneous Systems

Dominik Grewe, Zheng Wang and Michael O’Boyle

School of Informatics, University of Edinburgh

UK Many-Core developer conference (UKMAC 2012)
Problem and Objective

• GPU computing has become mainstream
  – OpenCL emerges as an industry-wide standard
• Programming with OpenCL is non-trivial
  – Applications may need to be rewritten or retuned when targeting new processors

• **Goal:** use OpenMP to ease of programming and use OpenCL to exploit GPU performance
  – Automatically generate optimised OpenCL code for GPUs
Task Mapping - CPU vs GPU

**Parallelism**
- CPU: Sequential / modestly parallel tasks
- GPU: Highly parallel tasks

**Memory Accesses**
- CPU: Irregular / indirect access
- GPU: Regular / aligned access

**Control Flow**
- CPU: Irregular, many branches
- GPU: Regular control flow
Our Approach

• OpenMP data parallelism
  – Convert OpenMP parallel loops into OpenCL kernels

• Generate efficient OpenCL code
  – Optimise for memory behaviour

• Determine the best device (CPU or GPU)
  – at runtime
  – If it is not profitable to run the OpenCL code on the GPU, run the OpenMP version on the CPU
The Compiler Framework

OpenMP

OpenCL Code Generation

OpenCL

Collect Code Features

Program Features

Code Merge

Output program

Machine learning model lib

Features

Program Features

Output program

OMP

OCL
OpenCL Code Generation

• Extract loop parallelism from OpenMP programs

• GPU Code Optimisation
  – Loop interchange
  – Array index reordering
  – Memory Load Reordering
  – Register Promotion
  – Vectorisation
  – Change data structures
  – ...

08/12/2012
for (i=1; i < .. ; i ++)
    for (j=1; j< ..; j ++)   [i, j, k]
        for (k=1; k< ..; k++) {
            ...
            lhs[i][j][k][0][0][0] = ...;
            lhs[i][j][k][0][0][1] = ...;
            ...
        }

Loop interchange &
index reordering

#pragma omp parallel for
for (k=1; k<...; k++)
    for (j=1; j<...; j++)   [k, j, i]
        for (i=1; i<..; i++){
            ...
            lhs[0][0][0][i][j][k] = ... ;
            lhs[0][0][1][i][j][k] = ... ;
        }

08/12/2012
Application performance depends on optimisation and program input. Select the right device is important.

(a) Naive OpenCL Translation

(b) After Optimisation

BT (NAS Parallel Benchmark)
Program Mapping

• Determine the best device according to
  – Program characteristics
  – Runtime program input

• Machine learning based approach
Code Features

• **Static OpenCL program features**
  – Computation - communication ratio
  – Computation - global memory ratio
  – % coalesced memory accesses
  – ...

08/12/2012
Create a Model

Model is automatically created offline using training benchmarks.
Use the Model

Compiler Time
• Unknown feature values are represented as symbolic pre-computation

Execution Time:
• Insatiate unknown feature values at runtime
Experimental Setup

• Platforms
  – Intel Core i7 (6-core) + NVIDIA GeForce
  – Intel Core i7 (6-core) + AMD Radeon

• Benchmarks
  – Full suite NAS benchmarks – up to 66 kernels

• Comparison to
  – closest related work: OpenMPC (Lee et al. PPoPP’09)
  – hand-written OpenCL implementation (Seo et al., IISWC ’11)
GeForce: 98% of ‘oracle’ (the best) performance

OpenMP on CPU: 2.78x
OpenCL on NVIDIA GPU: 1.19x
Our approach: 4.70x
Radeon: 99% of oracle performance

OpenCL on AMD GPU: 0.71x
OpenMP on CPU: 2.74x
Our approach: 4.81x
Compare to State-of-the-Art

- **OpenMPC**: OpenMP to CUDA
  - (Lee et al. PPoPP’09)
- **SNU NPB**
  - Hand-written OpenCL implementation (Seo et al., IISWC’11)

1.63x faster than hand-coded version
Conclusion

• A compiler framework for CPU-GPU heterogeneous systems

• Loop and data transformations to improve code quality

• Predictive modelling for task mapping
  – The model is automatically trained offline
Thank You

To appear in 2013 International Symposium on Code Generation and Optimization (CGO)

EPSRC PreMapp Project: http://groups.inf.ed.ac.uk/PreMapp/