LARA Tutorial

5. MATLAB to C/OpenCL Compilation

Tiago Carvalho, Pedro Pinto, João Bispo, Ricardo Nobre, Luís Reis, and João M.P. Cardoso

University of Porto, FEUP, Porto, Portugal

April 20th, 2016

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

ANTAREX 10^{18}

SPeCS
Special-Purpose Computing Systems, languages and tools

Objectives

- Understanding MATLAB to C Compilation
- Using LARA to guide MATLAB-to-C compilation
- Understanding OpenCL Code Generation

Compiling MATLAB to C

- Many projects are prototyped in MATLAB, but deployed in C
  - So the code has to be translated between these two languages
- So we developed a C backend for MATISSE
  - Capable of handling a non-trivial subset of MATLAB
  - Generates efficient code
- How to compile?
  - Website has a demo
  - Types of the input variables must be specified by the user

Is it fast?

- We tested a MATLAB example from the San Diego Vision Benchmark Suite (Disparity, FullHD image)
  - Original MATLAB: >4 min
  - Optimized MATLAB: 12.2s
  - Original MATLAB compiled with MATISSE: 6.86s
  - Modified MATLAB to use MATISSE extensions: 4.19s
    - We’re improving MATISSE: We aim to reach 2.75s and we know how to do it
  - Original C version: 2.7s

5.1 subbandSingle

- Goal
  - Compile a simple MATLAB program
- File
  - subband.m
- Strategy
  - List the type of each variable
  - Define its type
- Notes:
  - Try removing all type information except for “z” and “m”.

5.2 Smart Define Types

- Goal
  - Automatic type specification
  - Only works for certain code styles
- File
  - test.m
- Strategy
  - Find all inputs
  - See which name format they have
  - Define the type based on that

5.3 Use Primitives

- Goal
  - Take advantage of MATISSE-specific extension functions
- File
  - subband.m
- Strategy
  - Find MATLAB calls to “zeros”
  - Replace with a call (“matisse_new_array_from_dims”) that does not initialize the values

5.4 Insert Directives

- Goal
  - Remove expensive runtime checks by adding MATISSE-specific directives
- File
  - subband.m
- Strategy
  - Find the function with the specified name
  - Add the “assumption” directives
- If all functions are safe, then the “Unchecked” mode that automatically does this.
  - Not available in the web demo.

5.5 Add ByRef Directive

- Goal
  - Reduce number of allocations in example
- File
  - need_for_byref_test.m
- Strategy
  - Modify function declaration
  - Move variable allocation
  - Adapt function calls

MATLAB to OpenCL Compilation

- OpenCL: Suitable for GPUs and multi-core CPUs
  - Early FPGA implementations as well
- Early stages
- Not available in web demo

MATLAB to OpenCL Compilation (2)

- How it works:
  - Most code is compiled by the C backend normally
  - Programmers insert directives (manually or with LARA) in sections to parallelize
  - The generator generates OpenCL for those sections, along with the necessary wrappers.
  - The code can then run on any OpenCL-capable device, such as modern GPUs.

Takeaway Points

- MATISSE features a MATLAB to C/OpenCL Compiler
- LARA can be used to preprocess the MATLAB file, as well as specify additional program properties (e.g. Types)
- Generated code can be improved by using MATISSE extensions
  - These can be placed manually, or through LARA