Improving code portability and optimization with Halide

Many embedded systems such as health-care monitoring applications, high tech production printers and digital cameras execute complex image processing pipelines. These applications are usually composed of different stencil stages and complex reductions. Some stages may also have global or data-dependent access patterns. Because of this complex structure, the performance difference between a naive C implementation of the application and an optimized one is often an order of magnitude. In order to meet the real-time constraints of the system, the software implementation of these applications must be highly optimized for a specific target platform which is usually either a multi-core CPU with an on-board GPU extension or a dedicated GPU architecture. Unfortunately, modern compilers ignore many of these architectural details during their optimization passes. As a result, optimizing for a target platform remains a mostly manual, error prone process where the final code has no relation to the initial algorithmic description. As a result, when changes are made to a specific stage of the application pipeline, or a new target platform is used, all optimization effort must be spent again.

Assignment

The Halide DSL\(^1\) and compiler, aims to address the optimization and portability problems by separating the algorithmic description of a pipeline from its optimization schedule. This enables both fast exploration of the optimization space and increased code portability. However, efficient implementations of a pipeline require optimization of both parallelism and locality, but due to the nature of stencils, there is a fundamental tension between parallelism, locality, and introducing redundant recomputation of shared values.

As a first step in this project you will implement an existing image processing pipeline used in the copypath of professional production printers in Halide. Next you will use the Halide compiler to optimize it for performance for GPUs, as well as multi-core, heterogeneous CPU-GPU architectures. To optimize the performance the trade-off between parallelism and locality must be explored systematically.

Requirements

Experience in C/C++ and preferably also with CUDA programming
Knowledge on Computer architectures (CPU,GPU)

Location

The assignment will take place at Océ Technologies in Venlo.

\(^1\) http://halide-lang.org/