Abstract

The arrival of manycore systems enforces new approaches for developing applications in order to exploit the available hardware resources. Developing applications for manycores requires programmers to partition the application into subtasks, consider the dependence between the subtasks, understand the underlying hardware and select an appropriate programming model. This is complex, time-consuming and prone to error.

In this thesis, we identify and implement abstraction layers in compilation tools to decrease the burden of the programmer, increase programming productivity and program portability for manycores and to analyze their impact on performance and efficiency. We present compilation frameworks for two concurrent programming languages, `occam-pi` and `CAL Actor Language`, and demonstrate the applicability of the approach with application case-studies targeting these different manycore architectures: STHorm, Epiphany and Ambric.

For `occam-pi`, we have extended the Tock compiler and added a backend for STHorm. We evaluate the approach using a fault tolerance model for a four stage 1D-DCT algorithm implemented by using `occam-pi`'s constructs for dynamic reconfiguration, and the FAST corner detection algorithm which demonstrates the suitability of `occam-pi` and the compilation framework for data-intensive applications. We also present a new `CAL` compilation framework which has a front end, two intermediate representations and three backends: for a uniprocessor, Epiphany, and Ambric. We show the feasibility of our approach by compiling a `CAL` implementation of the 2D-IDCT for the three backends. We also present an evaluation and optimization of code generation for Epiphany by comparing the code generated from `CAL` with a hand-written C code implementation of 2D-IDCT.