Title: Procedural code integration in streaming environments

Author: Mgr. Michal Brabec

Department: Department of Software Engineering

Supervisor: David Bednárek, Ph.D.

Abstract: Streaming environments and similar parallel platforms are widely used in image, signal, or general data processing as means of achieving high performance. Unfortunately, they are often associated with domain specific programming languages, and thus hardly accessible for non-experts. In this work, we present a framework for transformation of a procedural code to a streaming application. We selected a restricted version of the C# language as the interface for our system, because it is widely taught and many programmers are familiar with it. This approach will allow creating streaming applications or their parts using a widely known imperative language instead of the intricate languages specific to streaming.

The transformation process is based on the Hybrid Flow Graph – a novel intermediate code which employs the streaming paradigm and can be further converted into streaming applications. The intermediate code shares the features and limitations of the streaming environments, while representing the applications without platform specific technical details, which allows us to use well known graph algorithms to work with the code efficiently.

In this work, we present the entire transformation process from the C# code the complete streaming application. This includes the management of the control flow, data flow, processing of arrays, method integration and optimizations necessary to produce efficient applications. Control flow represents the main difference between procedural code, driven by control flow constructs, and streaming environments, driven by data. We transform the code directly into the Hybrid Flow Graph and then we optimize the graph to introduce a structure better suited for the streaming environments. Finally we transform the graph into a streaming application.

We use procedurally generated code to verify the framework's correctness, where we test methods containing all combinations of loops, branches and serial code nested in each other. We also evaluate the performance of the produced applications and since the use of a streaming platform automatically enables parallelism and vectorization, we were able to demonstrate that the streaming applications generated by our method may outperform their original C# implementation.

Keywords: code transformation; intermediate code; parallel programming; vectorization; streaming systems;