OctoPOS: An Operating System for Invasive Computing

Benjamin Oechslein

Friedrich-Alexander-Universität Erlangen-Nürnberg
oechslein@cs.fau.de

Invasive Computing is a form of resource-aware computing specifically targeted towards massively-parallel systems that is currently being investigated in the Transregional Collaborative Research Center (TCRC) *Invasive Computing*[1]. The main idea behind Invasive Computing is that applications express their resource demand to the system and subsequently compete with each other at run-time for the available resources. Hence, applications need to be able to adapt themselves to over- or underfulfilment of their resource demands, but on the other side, once resources are claimed on behalf of an application, they are at their full disposal.

OctoPOS strives to provide a lightweight and efficient execution environment for invasive-parallel programs. It runs on a FPGA-based multi-core prototype specifically designed to support invasive computing that is being developed as part of the TCRC. This hardware comprises multiple, so-called tiles — clusters of processing cores with some local memory — that are interconnected via a Network-on-Chip. Cache coherency is only maintained for accesses originating from the same tile. Hence, OctoPOS is organised as a group of closely-coupled OS instances that provide the basic services for invasive computing: Acquisition and release of resources, possibly spanning multiple tiles and efficient execution of parallel programs on these resources.

Programs are split up in small fragments of execution that mostly adhere to run-to-completion semantics resulting in a very efficient implementation in terms of both memory footprint and execution time. Thus, OctoPOS is able to support large amounts of execution fragments and this enables application programmers to leverage even very fine-grained parallelism. To further bring down operating-system overhead the execution model is designed with hardware support in mind that assists OctoPOS in executing and synchronising control flows across tile boundaries. This leads to an interruptless implementation of all dispatching functionality even across the cache-cohererent tile boundary where common synchronisation primitives and data structures in shared memory cannot be used.

The talk will give an overview of the design and implementation of OctoPOS as well as of the integration and co-design with the surrounding hard- and software ecosystem.

[1] http://www.invasic.de