Hardware virtualisation for heterogeneous many-core systems

Grelck, C.; Poss, R.; Jesshope, C.

Publication date
2010
Document Version
Author accepted manuscript
Published in
Intel European Research and Innovation Conference (ERIC'10), Braunschweig, Germany

Hardware Virtualisation for Heterogeneous Many-Core Systems

Clemens Grelck, Raphael Poss, Chris Jesshope

[1] University of Amsterdam, Amsterdam, Netherlands

C.Grelck@uva.nl

Summary

The multi-core/many-core revolution has brought up a hardly precedented diversity in computer architecture. While parallelism id the common property, granularity of concurrent processing resources may easily span multiple orders of magnitude. This requires design decisions in the organisation of concurrent program execution to be made differently depending on the concrete execution platform. We propose a hardware virtualisation layer that separates the aspect of granularity from the expression (or detection) of concurrency and provides uniform access to concurrent computing resources.

1 Motivation

The multi-core/many-core revolution has not only changed the landscape of computing machinery from sequential to parallel. It also leads to an increasing diversity in hardware parameters that goes way beyond different instruction set architectures or operating systems from a limited number of vendors in previous hardware generations. Today the numbers of compute cores in a single system can almost span two orders of magnitude. We find modest numbers of very powerful cores next to large numbers of functionally restricted cores. We find all kinds of memory subsystems from distributed memory over non-consistent shared memory to consistent shared memory. Accelerator boards and GPGPUs add another layer of complexity.

In machine-level concurrent programming all these hardware characteristics require design choices to be made in certain ways and many architectural features require explicit coding (e.g. CUDA). Moreover, these hardware characteristics are in constant flux. Supporting a wide variety of different parallel architectures in an explicitly is neither desirable from a software engineering point of view nor economically viable in most application domains.

2 Solution

We propose SVP, a hardware virtualisation layer that decouples the notion of concurrency as expressed by the programmer from the capabilities of the target architecture, which may not even be known at system-design time. Our aim is to provide a write-once, deploy-anywhere philosophy. Consequently, application programmers do not need to be concerned with details of the target architecture when writing application code. Instead, they can rather focus on expressing as much concurrency as possible using the mechanisms available in the source language, e.g. high-level concurrent programming languages like the data-parallel array language SAC (Single Assignment C) or the asynchronous stream-based coordination language S-Net. It is then up to the virtualisation layer to aggregate this fine-grained concurrency into a grain size that matches the requirements of a specific target architecture for efficient execution. This aggregation may happen at compile time (if the target system is known), at deploy time or even at runtime to care for dynamic variations in the effectively available hardware (e.g. through hardware failure or external reconfiguration).

In essence, hardware virtualisation removes the aspect of granularity as a concern of concurrent programming and unifies access to computing resources. SVP acts as a mediator between different concurrent programming environments on one side and a variety of execution environments on the other side.

Publications

[1] C. Jesshope, “Operating Systems in Silicon and the Dynamic Management of Resources in Many-Core Chips“, *Parallel Processing Letters* **18(2)** (2008).

[2] T. Bernard, C. Grelck, and C. Jesshope, “On the Compilation of a Language for General Concurrent Target Architectures“, *Parallel Processing Letters* **20(1)** (2010).