Advances in the MultiController model: Programming heterogeneous systems in a homogeneous way

Ana Moreton-Fernandez, Eduardo Rodriguez-Gutiez,
Yuri Torres, Arturo Gonzalez-Escribano, Diego R. Llanos

Trasgo Group, University of Valladolid (Spain)

February 22, 2018

Abstract

Current HPC clusters are composed by several machines with different computation capabilities and different kinds and families of accelerators. Programming efficiently for these heterogeneous systems has become an important challenge. Generating coordination codes using different vendor specific programming models and languages to obtain the best possible efficiency is time consuming, error-prone, and demands a good knowledge of several types of architectures. There are many proposals to simplify the programming and management of accelerator devices and their hybrid programming, mixing accelerators and CPU cores. However, in many cases, portability compromises the efficiency, and there are details concerning the coordination of different types of devices that should still be tackled by the programmer.

In this paper we summarize the recent research advances of our group designing runtime and code generation solutions in the context of the Multi-Controller model. The Multi-Controller is an abstract entity implemented in a library that coordinates the management of several heterogeneous devices, including different types of accelerators and sets of CPU-cores, in an homogeneous way.

The Multi-Controller transparently manages both the kernel launching and the data-transfers between the host machine and different types of devices. It allows the programming of generic kernels portable across devices, that are efficiently adapted by generating a proper granularity and by selecting proper launching parameters. It also allows to build specialized kernels, specific for a device type, which are automatically selected by the runtime system during the program execution when the target platform includes the corresponding devices. To obtain a good efficiency, it internally exploits, and allows the exploitation in the kernels, of vendor specific programming models and languages (such as CUDA for NVIDIA GPUs, or OpenMP for sets of CPU-cores and XeonPhi platforms).

These features are used, for example, to create a really portable interface for specialized libraries, such as BLAS. The Controller internally links and selects at runtime the proper vendor implementation for each different type of device (such as cuBLAS for NVIDIA, MKL for Intel CPUs and XeonPhi, or different versions of MAGMA). This introduces a new level of application portability in highly heterogeneous systems.

Finally, the Multi-Controller presents an extensible system to apply data distribution policies, creating a context across devices with different architectures and features. It allows the execution of kernels using these distributed data structures in a transparent way.

We present practical examples, and results of experimental studies that indicate that this abstraction allows the development of flexible and highly efficient programs that adapt to the heterogeneous environment. It reduces the development effort and increases the code portability when it is compared with using the native or vendor programming models directly, while it achieves similar efficiency.

Most of the material presented has been published in recent conference and journal papers. We also provide hints about the future developments related to this work.