ZRLMPI: A Unified Programming Model for Reconfigurable Heterogeneous Computing Clusters

Burkhard Ringlein*†‡, François Abel‡, Alexander Ditter‡, Beat Weiss‡, Christoph Hagleitner‡, and Dietmar Fey*†
IBM Research Europe, ‡Friedrich-Alexander University Erlangen-Nürnberg
*ngl, fab, wei, hle}@zurich.ibm.com, {burkhard.ringlein, alexander.ditter, dietmar.fey}@fau.de

Abstract—Over the past two decades, the Message Passing Interface (MPI) has evolved as the de-facto standard for programming High-Performance Computing (HPC) clusters. Its widespread utilization led to the rapid development of applications and high reusability. Meanwhile, energy- and compute-efficient devices such as Field-Programmable Gate Arrays (FPGAs) are stepping into modern data centers and HPC clusters to address the nearing end of technology scaling. This combination of traditional CPU servers and FPGA nodes leads to Reconfigurable Heterogeneous HPC (ReHPC) systems that are particularly cumbersome to program because of the absence of a standard programming model. This work advocates the use of MPI to program such ReHPC clusters and presents a proof of concept based on a cross-compiler, a High-Level Synthesis library, a C++ library, an FPGA- and a CPU-runtime environment. The result is a one-click solution, which compiles a standard MPI application for a ReHPC cluster.

I. PROGRAMMING REHPC CLUSTERS

Today’s High-Performance Computing (HPC) systems can be classified into three classes. The first and traditional HPC class solely consists of CPU servers, while the second class, typically referred to as Reconfigurable HPC (ReHPC), is only comprised of Field-Programmable Gate Arrays (FPGAs) nodes. The third class is named Reconfigurable Heterogeneous HPC (ReHPC) because it comprises a mixture of the CPU servers from the first class and the FPGA nodes from the second class. Unfortunately, despite many attempts, no standard has yet emerged for the programming of such heterogeneous clusters. This absence of agreement hinders the rapid development of applications using FPGAs in HPC, and motivated us to reconsider the use of the Message Passing Interface (MPI) for ReHPC platforms. MPI is widely adopted in the HPC community and we want to demonstrate that, with its standardized syntax and semantics, it also fits as a single programming model for ReHPC clusters. To avoid re-coding every application for every specific heterogeneous cluster, we propose a High-Level Synthesis (HLS) approach. This MPE is analogous to the affinity concept of MPI. To implement the MPI synchronization and collective routines via the underlying cluster communication protocol, we developed an HLS core called Message Passing Engine (MPE). This MPE is merged with the application HLS code by ZRLMPIcc and is synthesized to a partial bitstream. In parallel, the CPU specific parts are also emitted by ZRLMPIcc and compiled together with the ZRLMPI software runtime library (ZRLMPIlib). This ZRLMPIlib is the software counterpart of the MPE that synchronizes CPU and FPGA nodes. To distribute the partial bitfiles and software binaries as specified by the rankfile, we’ve developed a deployment framework (ZRLMPIrun) using the FPGA management runtime of platform [1].

II. ZRLMPI: MPI FOR REHPC

The goal of ZRLMPI is to bring CPUs and FPGAs to work together efficiently using a single source of code. As an example, consider the MPI code of Listing 1, which forwards a message around a ring of multiple nodes from a sender (rank 0) back to that same node. In such a programming approach, the user is not expected to annotate the MPI code or to use HLS tools her/himself in order to bring the program to a ReHPC cluster. This step is automated by our cross-compiler (ZRLMPIcc) that identifies the parts of the program that will be executed on FPGAs and transforms these parts from the original C code to synthesizable HLS code. To identify these parts, ZRLMPIcc uses a user-defined rankfile that maps every rank to a specific physical node. This is analogous to the affinity concept of MPI. To implement the MPI synchronization and collective routines via the underlying cluster communication protocol, we developed an HLS core called Message Passing Engine (MPE). This MPE is merged with the application HLS code by ZRLMPIcc and is synthesized to a partial bitstream. In parallel, the CPU specific parts are also emitted by ZRLMPIcc and compiled together with the ZRLMPI software runtime library (ZRLMPIlib). This ZRLMPIlib is the software counterpart of the MPE that synchronizes CPU and FPGA nodes. To distribute the partial bitfiles and software binaries as specified by the rankfile, we’ve developed a deployment framework (ZRLMPIrun) using the FPGA management runtime of platform [1].

REFERENCES


©2020 IEEE Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.