Early Experiences with the OpenMP Accelerator Model*

Chunhua Liao[1], Yonghong Yan[2], Bronis R. de Supinski[1], Daniel J. Quinlan[1], and Barbara Chapman[2]

[1] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory
{liao6,dquinlan,desupinski1}@llnl.gov
[2] Department of Computer Science, University of Houston
{yanyh,chapman}@cs.uh.edu

Abstract. A recent trend in mainstream computer nodes is the combined use of general-purpose multicore processors and specialized accelerators such as GPUs and DSPs in order to achieve better performance and to reduce power consumption. To support this trend, the OpenMP Language Committee has approved a set of extensions to OpenMP (referred to as the OpenMP accelerator model). The initial version is the subject of Technical Report 1 (TR1) while OpenMP 4.0 Release Candidate 2 (RC2) further refines the extensions.

In this paper, we examine the newly released accelerator directives and create an initial reference implementation, referred to as HOMP (Heterogeneous OpenMP). Focused on targeting NVIDIA GPUs, our work is based on an existing OpenMP implementation in the ROSE source-to-source compiler infrastructure. HOMP includes extensions to parse the new constructs and to represent them in the AST and other compiler translation details. Further we provide initial runtime support. For our evaluation, we have adapted a few existing OpenMP codes to use the accelerator model directives and present preliminary performance results. Finally, we critique the accelerator model in terms of its impact on developers and compiler writers and suggest possible improvements.

1 Introduction

Heterogeneous computer architectures that combine general-purpose multicore CPUs with specialized accelerators have become a viable solution to build high performance supercomputers, as demonstrated by Titan at ORNL (NVIDIA GPGPUs) and Stampede at TACC (Intel Xeon Phi) in the recent top500 list. Multicore CPUs are good at processing coarse-grained, irregular tasks; while accelerators excel in certain workloads such as large-scale data parallel and finer-grained vector processing. However, to exploit their computation capabilities

* LLNL-CONF-636479. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work was also supported by the National Science Foundations Computer Research Infrastructure program under Award No. CNS-1205708.

A.P. Rendell et al. (Eds.): IWOMP 2013, LNCS 8122, pp. 84–98, 2013.

efficiently has required significant programmer effort to optimize an application program with respect to the specific hardware features of each type of accelerator.

Programming models such as OpenCL, CUDA and Brook provide mechanisms for an application to exploit the hardware capabilities of accelerators. High-level programming models, such as OpenACC [1], aim to provide an easier migration option from a sequential or parallel CPU version to the use of accelerators, typically GPGPUs. However, using these programming models to exploit their capabilities completely still poses significant challenges, even for expert programmers. Using multiple programming models in one application, as is likely with models that provide accelerator support that is distinct from CPU models, increases code complexity and decreases its portability. Mixing multiple programming models also complicates the compiler and runtime support due to the language complexity and to support runtime interoperability.

OpenMP has proven to be a productive solution for parallel programming with CPUs in shared memory systems. Recently, the OpenMP Language Committee has been working toward a single specification that supports heterogeneous computation nodes using both CPUs and accelerators. The committee has developed a set of extensions that they released first as a dedicated Technical Report 1 (TR1) and then as part of OpenMP 4.0 Release Candidate 2 (RC2) [2]. The extensions in this OpenMP accelerator model build on existing OpenMP concepts and constructs to provide a unified model for GPUs and CPUs. This model relies on compiler analysis and transformations to generate code that can execute on accelerators for specified source code regions, as well as runtime support to provide data movement and other support for hybrid execution.

In this paper, we review the OpenMP accelerator model and share our experiences of creating an initial implementation, the Heterogeneous OpenMP (HOMP) compiler. We have two goals: to provide early feedback on the usability of the OpenMP accelerator model and its impact on compiler writers; and to create a reference implementation for the extensions that the research community can leverage to explore further extensions.

The rest of the paper is organized as follow. Section 2 reviews the major accelerator extensions to OpenMP. Section 3 describes our initial implementation of those extensions. We present our preliminary results in Section 4 and critique the current model in Section 5. Section 6 presents related work. Section 7 concludes the paper and discusses our future work.

2 The OpenMP Accelerator Model

OpenMP 4.0 Release Candidate 2 [2] extends the execution model of the specification to support accelerators with device constructs. The OpenMP accelerator model assumes that a computation node has a host device connected with one or multiple accelerators as target devices. It uses a host-centric model in which a host device "offloads" code regions and data to accelerators for execution, specified using the `target` construct. This construct causes data and an executable to be copied (offloaded) to the accelerator before computation.