PhD Forum Extended Abstract Jose M Monsalve Diaz

JOSE M MONSALVE DIAZ, University of Delaware, USA

High performance computing systems are under constant stress from applications and users to achieve better results. There is a constant need of innovation that drives designers at all levels of HPC systems to constantly improve their systems and integrate new features. However, the complexity of the whole HPC ecosystem is so large that changes at one level needs to be supported, or at least adapted, at the other levels. One feature that has been widely adopted during the last decade is the integration of co-processors and accelerator devices to computer systems. For instance, an analysis of recent results of the Top 500 supercomputers shows that 15 of the systems are provisioned with some sort of co-processor, the most common one being GPGPUs.

Although the idea of including specialized hardware to accelerate computation is not new, and it dates from several decades back, the evolution of programming models and frameworks for parallel system is allowing new techniques to unburden the programmer's job of dealing with multiple target architectures within the same code. As a result and in response to the evolution of HPC systems, OpenMP, as one of the most popular programming frameworks for parallel systems, has recently introduced target offloading features for acceleration devices. Starting with OpenMP 4.0, users have access to offloading pragma-like constructs. The compiler will use these constructs to create kernel functions that will be executed on the devices (e.g. GPGPUs and Intel Xeon Phi) as well as to orchestrate data movement between host and device.

OpenMP specifications are a guideline for compiler developers to extend the functionality of the C, C++ and Fortran programming languages to support parallel execution of code. The specifications work as a contract between the programmer and the compiler. If a compiler wants to be compliant with the specifications, it must implement all the different constructs, clauses and API calls that describes the specifications. On the other hand, the user expect to be able to move from one compiler to another one without requiring the modification of a single line of code. Nevertheless, this is not always the case. While there is always the possibility to have implementation bugs in the compiler side, specially at an early stage. There are cases where the merely interpretation of the specifications can be an issue, as they might incur to ambiguities or misunderstanding of what the OpenMP committee had in mind when writing the specs. This could cause ambiguous behavior when moving from one compiler to another oner.

In particular, OpenMP 4.5 support is still an ongoing process for most of the compilers. Currently, GCC, Clang, XLC, and Cray are some of the compilers that have started to implement OpenMP 4.5 functionality. On the other hand, users and application developers are just starting to take into account in their codes. While there are still open questions regarding performance and portability of these programming models, there is another important question to be answered. What are the levels of readiness and compliance of the different available implementations? To this end, we are proposing to create a validation and verification tests suite that, based on the specifications and with the mindset of an OpenMP user, provides a way to asses the level of compliance of the different OpenMP 4.5 implementations that are available in different compilers. We have developed a well-defined methodology. Next, we have used this methodology to create multiple functional tests. Each test focuses on a particular construct or clause in accordance to the OpenMP 4.5 specification manual. Following, we have used these tests against multiple compilers, in three different state-of-the-art computer systems that are available to us at ORNL. This work summarizes this process, showing the methodology, examples of the tests and results. This project aims to attract attention to our test suite which is publicly available through our website.

ACM Reference Format:
Jose M Monsalve Diaz. 2018. PhD Forum Extended Abstract Jose M Monsalve Diaz. 1, (June 2018), 1 page.

Received June 2018

Author's address: Jose M Monsalve Diaz, University of Delaware, DuPont Hall, University of Delaware, Newark, DE, 19711, USA, josem@udel.edu.

2018. XXXX-XXXX/2018/6-ART
https://doi.org/