USING OPENMP OFFLOADING TO RUN CODE ON BLUE WATERS' GPU NODES

Allocation: Blue Waters Professor/100 Knh
PI: Daniel Bodony[1]
Collaborators: Matthias Diener[1], Sandeep Murthy[1]

[1]University of Illinois at Urbana–Champaign

EXECUTIVE SUMMARY

Heterogeneous systems containing a CPU+GPU pair on a single node (such as Blue Waters' XK nodes) are becoming more common. Programming such systems effectively is a difficult challenge, especially for applications that have not been developed from the ground up to support such systems. In this context, OpenMP has emerged recently as an interesting solution, by leveraging OpenMP offloading capabilities in existing code. In this work, OpenMP offloading was added to PlasCom2, a multiphysics simulation application, with the Hydra framework. Hydra enables concurrent execution of application code on a CPU+GPU pair, resulting in efficient resource usage and high-performance portability. Performance results on Blue Waters show gains of up to three times on a single XK7 node [2,3].

RESEARCH CHALLENGE

Programming heterogeneous systems is a challenging task, as few programming models support executing code on accelerators, leading to the use of specialized solutions such as CUDA, OpenCL, Legion, or Kokkos for these devices. Such specialized languages have the advantage of being able to provide the best performance in many cases since they can often provide support for special device features and offer good code generation for specific device types. However, existing application code can often not be reused and must be rewritten in a new language. Code is often not portable between devices (for example, between CPUs and GPUs), such that distinct devices may require different implementations, leading to duplicated code and an increased difficulty of code maintenance.

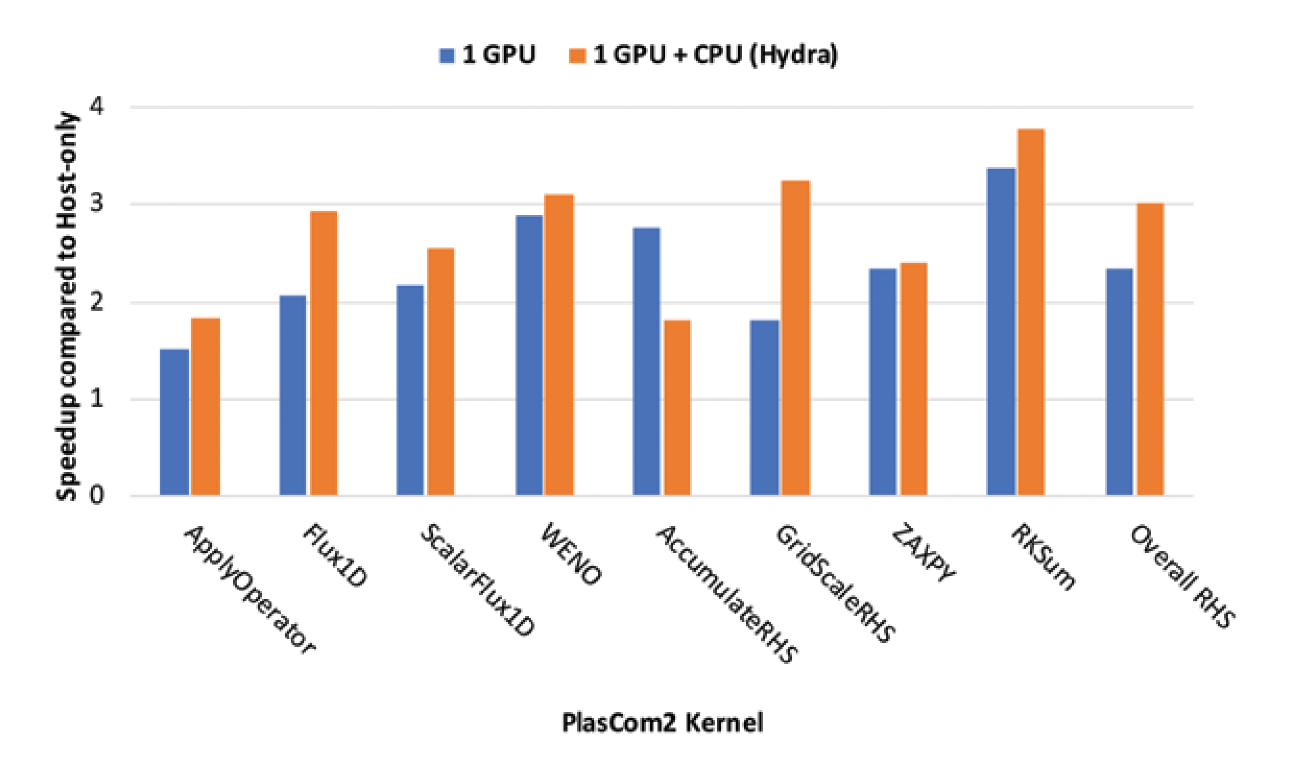

Figure 1: PlasCom2 kernel results on the Blue Waters XK7 system.

For PlasCom2, the research team needed an approach that was able to use the existing code (in C++ and Fortran) on several types of accelerators without having separate implementations for different device types. Furthermore, the team wanted to be able to support different hardware and software environments with this approach, and run concurrently on the host and offloading devices to use all available computing resources efficiently.

METHODS & CODES

Based on the offloading support available in recent OpenMP versions, the team developed Hydra, which is a library to support concurrent execution on host and accelerator devices. For PlasCom2, Hydra [1–3] measures the relative performance of the host and accelerator at startup and determines the best work distribution based on these data. During execution, Hydra handles the data movement between host and offloading device as well as the actual code execution [2,3]. Hydra requires an OpenMP 4.5 compiler and runtime but has no additional dependencies [1].

RESULTS & IMPACT

OpenMP offloading enables simple and efficient execution of a single code base on different types of devices, with minimal changes to existing code. Hydra builds on top of OpenMP offloading to add support for fully heterogeneous execution; that is, running parts of the problem concurrently on different device types. Using Hydra with PlasCom2 resulted in a speedup of three times compared to CPU-only execution on a Blue Waters XK7 node. Compared to running only on the GPU, performance from heterogeneous execution was improved by 17%. Some of the computationally intensive kernels of PlasCom2 showed a speedup of up to four times (Fig. 1). OpenMP offloading and Hydra show that existing codes can be enabled to run on heterogeneous systems with a low number of changes and high performance efficiency.

WHY BLUE WATERS

Blue Waters was essential to the research by providing a stable, high-performance platform with easy access to modern accelerators. Hydra and its integration into PlasCom2 could be developed directly on Blue Waters, allowing the team to evaluate and compare different implementation possibilities on a real system.