CATEGORY: **PARALLEL PROGRAMMING, LANGUAGES & COMPILERS**

POSTER **PP12** | CONTACT NAME Rengan Xu: uhxrg@cs.uh.edu

GPU TECHNOLOGY CONFERENCE

OpenACC Programming Experiences using Scientific Applications

Rengan Xu, Cheng Wang, Sunita Chandrasekaran, Barbara Chapman
Department of Computer Science, University of Houston
Email: {uhxrg, cheng, sunita, chapman}@cs.uh.edu

Introduction

- GPU architecture poses several programming challenges. One of the emerging and portable models is OpenACC; this programming interface describes a collection of compiler directives in standard C/C++ and Fortran to allow regions of code to be offloaded from a host CPU to an attached accelerator offering portability across OSs, CPUs and accelerators. OpenACC provides higher abstraction level and a more incremental porting approach.
- We created a validation suite that is being used to check OpenACC implementations for conformance to standard. This validations suite is integrated to the official harness testsuites of Titan (#1 in Top 500) for production testing.
- We evaluated OpenACC model using applications from several domains. We targeted GPU Kepler cards and compared our results to other similar models such as OpenMP targeted for multicore CPUs. We observed that OpenACC achieved several factors of performance improvement over OpenMP codes.
- Scientific applications are becoming more and more complex and we need productive and portable programming models to exploit the large number of hardware resources. To address this challenge, we explored a combination of OpenMP and OpenACC as a plausible solution to port scientific applications to heterogeneous architecture especially when there is more than one GPU on single node to port to.

OpenACC Validation Suite

The primary goal of the OpenACC validation suite is to validate implementations of the OpenACC API. Our testing infrastructure provides a set of short feature tests whenever possible and check whether the feature being tested has been implemented correctly. Therefore, each feature test must have a single meaning according to the OpenACC specification. The validation suite contains two types of tests: (A)Normal test: We compare the value with an "oracle" for the test, which will fail if the result mismatches with the pre-defined oracle. (B) Cross test: To gain more confidence of the test result, a deeper test methodology, namely cross test, is designed to validate only the directive under consideration. The basic idea is that if we remove the directive being tested from the test code, the cross test should yield an "incorrect" result. Another common use of cross tests would be intentionally replace the being tested directive with another one.

The test is based on the statistics and each test is repeated multiple times. In order to eliminate the probability that a test is passed accidently we take the following approach: if n_f is the number of failed cross tests and M the total number of iterations, the probability of the test to fail is $p=n_f/M$. Thus the probability that in incorrect implementation passes the test is $P_a = (1-p)^M$, and the certainly of test is $P_c = 1- P_a$, which means the probability that a directive is validated.

OpenACC Code Samples

We are constructing a suite of OpenACC code samples, this suite consists of applications from several well-known benchmarks such as NAS, PARBOIL, Rodinia and so on. The applications for the suite were chosen based on several domains and include a variety of computational demands.

The evaluation platform consists of NVIDIA Kepler20 GPU and an Intel Xeon CPU with 8 cores. We performed a number of code optimizations as well as code restructuring to achieve the competitive performance. These include function inline, array linearization, loop unrolling, loop fission, loop fusion and manually privatize large arrays. OpenACC specification is evolving and the members are working to better define the directives based on the feedback from the users like us. Figure 1 show the speedup of OpenACC code over CUDA, OpenMP and serial codes.

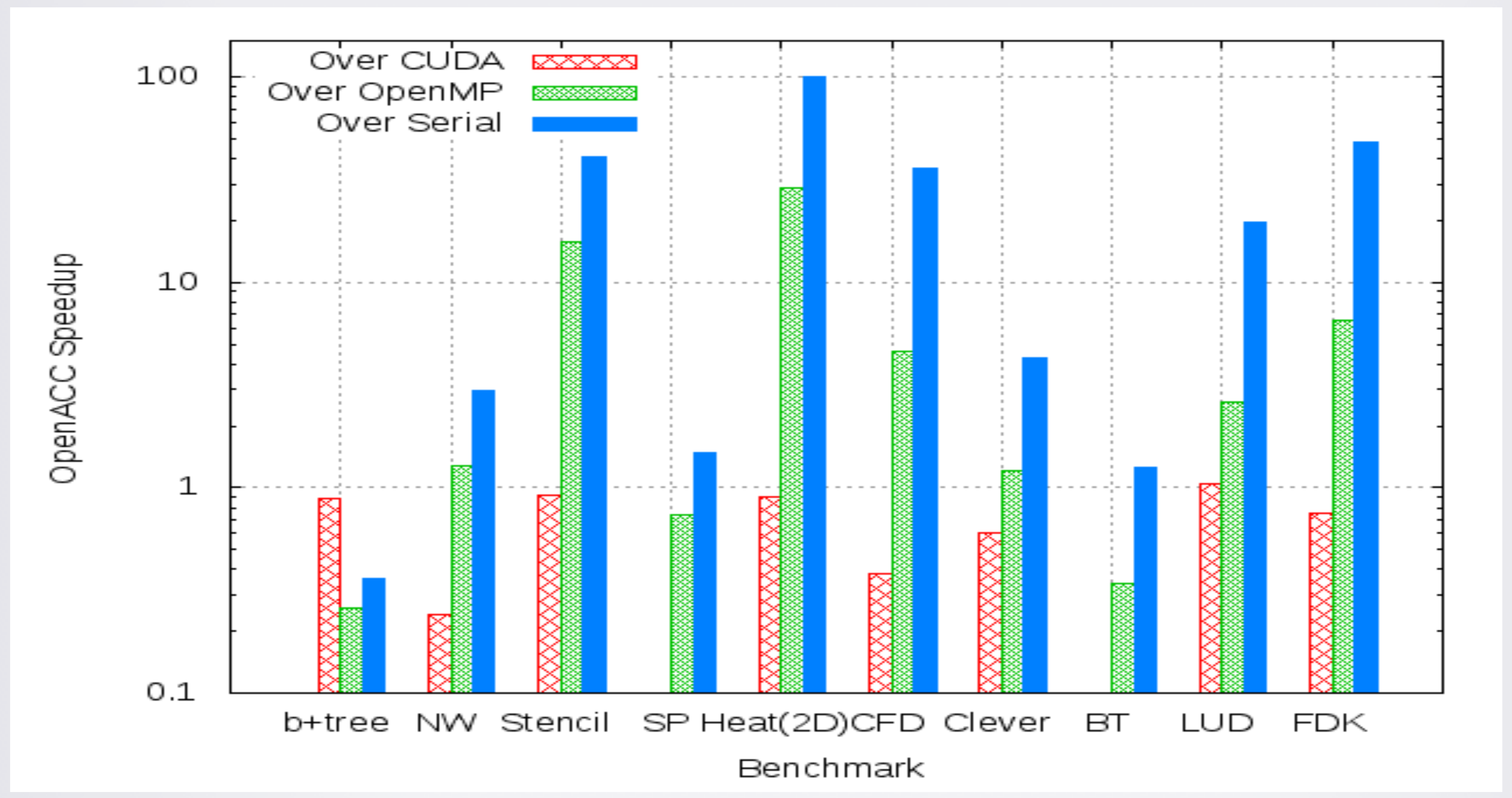

Fig 1: Performance Comparison of OpenACC and other Programming Models. All OpenMP implementations uses 8 threads. SP and BT do not have CUDA version.

Multi-GPU Programming - OpenACC & OpenMP

To enable the multi-GPU programming within a single node, we explored the feasibility using the hybrid model of OpenACC and OpenMP. Figure 2 (in the III column) gives a general idea of this hybrid model where we manually divide the problem among OpenMP threads, and then associate each thread with a particular GPU. The effectiveness of this approach is demonstrated by exploring three applications of different characteristics:

- S3D: It includes different independent kernels, each kernel is dispatched to one GPU.
- Matrix multiplication: It has a large workload that is decomposed into multiple small sub-workloads, after which each sub-workload is scheduled on one GPU.
- 2D Heat Conduction: Figure 3 shows the communication between different GPUs.

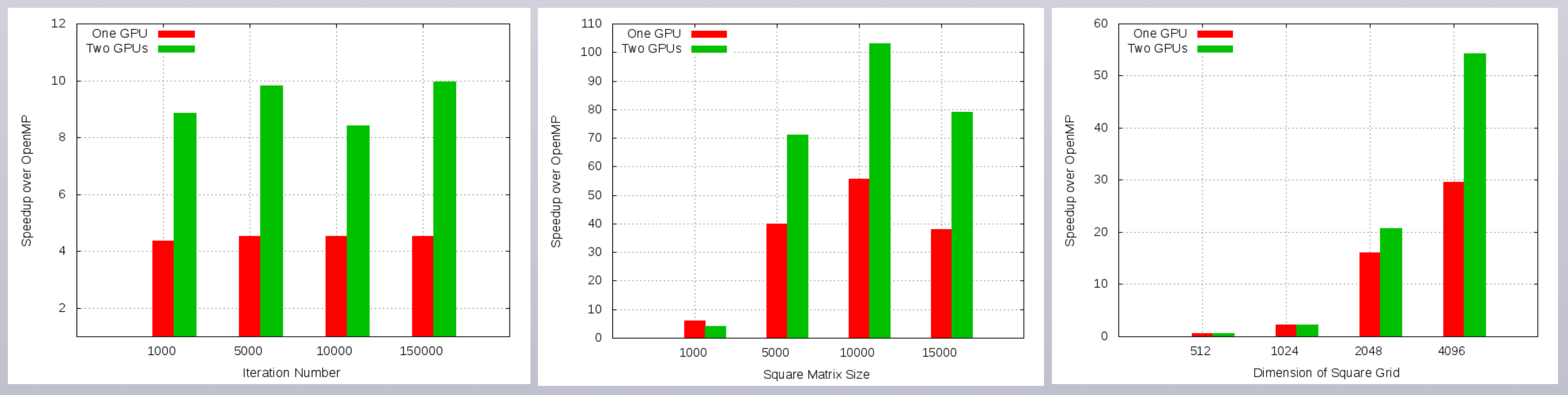

Fig. 4: Performance Comparison between Single GPU and Multi-GPUs Implementation. The performance shown here is the speedup compared to OpenMP version.

Results

Figure 4 shows the results of performance compared using single and multi-GPU implementations of S3D, matrix multiplication and 2D heat conduction. Some observations made are:

- In almost all the cases, the performance of a single GPU is much better to the OpenMP model; Multi-GPU implementation naturally performs better than single GPU
- S3D: Every iteration has the same amount of workload, hence the multi-GPU execution takes approximately half of the time taken by a single GPU.
- Matrix Multiplication: When the square matrix size is 1000, the speedup achieved by two GPUs was a little less than that achieved by a single GPU, possibly due to the overhead of host threads creation and GPU context setup.
- 2D heat conduction: The multi-GPU implementation shows significant performance increase when the problem size becomes larger primarily because the computation/communication ratio becomes larger.

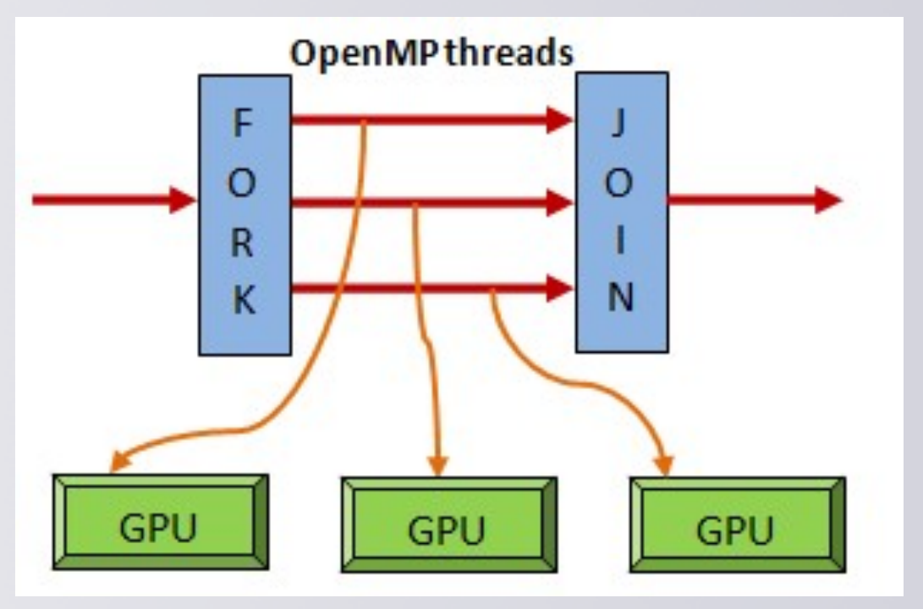

Fig. 2: A Multi-GPUs Solution using the hybrid OpenMP and OpenACC Model

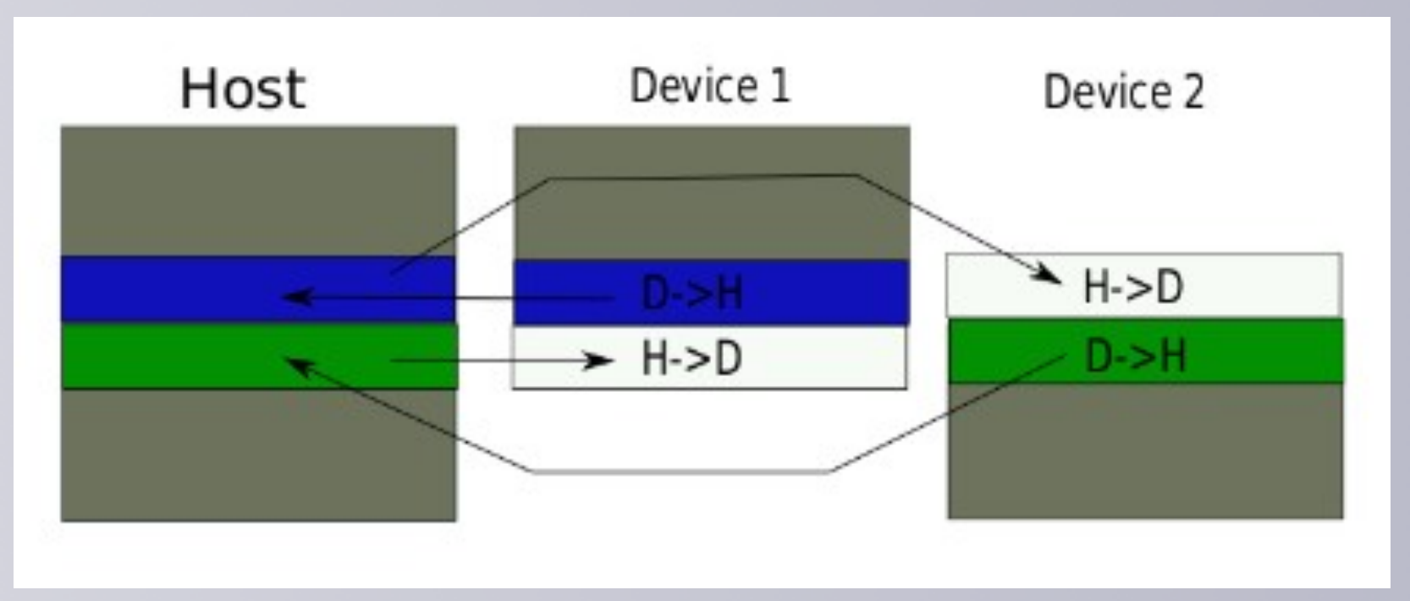

Fig. 3: GPU Communication in 2D heat conduction. Different GPUs exchange halo region data through the CPU host.

Conclusions

- OpenACC validation suite is critical to validate the correctness of OpenACC implementation in compilers.
- The performance of all the applications in the OpenACC code sample suite varies depending on the characteristics of applications chosen. We observed competitive speedup of OpenACC codes when compared against OpenMP and serial codes. We also observed that OpenACC is yet to achieve good speedup compared with that of the CUDA version of the applications.
- Multi-GPU programming can be achieved using a hybrid model – OpenACC and OpenMP.

References

1. Rengan Xu, Sunita Chandrasekaran, Barbara Chapman, and Christoph F. Eick. Directive-based Programming Models for Scientific Applications - A Comparison. Accepted to WOLFHPC'12 Workshop in conjunction with SC'12.(Under Publication)
2. Rengan Xu, Sunita Chandrasekaran, Barbara Chapman. Exploring Programming Multiple GPUs using OpenMP & OpenACC-based Hybrid Model. Accepted to PLC'13 Workshop in conjunction with IPDPS'13 (Under Publication).

Acknowledgement: This research is being supported by OpenACC standard committee.