Deakin, T. J., Price, J., Martineau, M. J., & McIntosh-Smith, S. N. (2016). GPU-STREAM: now in 2D!. Poster session presented at 2016 International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, United States.

Peer reviewed version

GPU-STREAM: now in 2D!

Tom Deakin*, James Price, Matt Martineau and Simon McIntosh-Smith
Department of Computer Science
University of Bristol
Bristol, UK
Email: *tom.deakin@bristol.ac.uk

***Abstract*—We present a major update to the GPU-STREAM benchmark, first shown at SC'15. The original benchmark allowed comparison of achievable memory bandwidth performance through the STREAM kernels on OpenCL devices. GPU-STREAM v2.0 extends the benchmark to another dimension: the kernels are implemented in a wide range of popular state-of-the-art parallel programming models. This allows an intuitive comparison of performance across a diverse set of programming models and devices, investigating whether choice of model matters to performance and performance portability. In particular we investigate 7 parallel programming languages (OpenMP 4.x, OpenACC, Kokkos, RAJA, SYCL, CUDA and OpenCL) across 12 devices (6 GPUs from NVIDIA and AMD, Intel Xeon Phi (Knights Landing), 4 generations of Intel Xeon CPUs, and IBM Power 8).**

I. Measuring Memory Bandwidth

The STREAM benchmark is well known for measuring the achievable memory bandwidth on CPU architectures [1]. It consists of four kernels: copy, add, multiply and triad. Each kernel performs a simple floating point arithmetic operation per element from one or more arrays and stores the result in another array. It is simple to count the number of useful bytes moved and hence calculate the sustained memory bandwidth.

II. Programming Models

There has been recent interest in different programming models, with new versions of existing standards competing with new alternatives that are emerging. We have used OpenCL and CUDA to represent 'close-to-the-metal' parallel programming languages. OpenCL is a portable programming model and we previously demonstrated that it can achieve a good fraction of peak memory bandwidth on a range of GPUs for the STREAM kernels [2]. CUDA is available for NVIDIA GPUs, but also x86 CPUs via the PGI compiler. Both these models require the programmer to implement parallelism in a fine grained manner.

OpenMP 4 and OpenACC allow regions of code to be offloaded to a device through the use of preprocessor/compiler directives. Parallel code is identified typically at the loop level. OpenMP additionally allows native execution of parallel code through directives, which we use for the CPUs and Xeon Phi.

RAJA, Kokkos and SYCL are all C++ based programming models with loop bodies expressed as lambda functions and executed via a "parallel for" routine.

All these models can target different multi- and many-core devices. The programmer, therefore, surely hopes to write their code in their favourite model and achieve some degree of both performance *and* performance portability across a range of devices.

III. Results and Conclusions

Table I lists the 12 hardware devices we tested. These cover a wide range of different architectures but also represent the latest commercially available hardware. The theoretical peak memory bandwidth for each device as published by the respective vendors is listed. Note that Intel do not publish an exact figure for Xeon Phi MCDRAM bandwidth saying it is roughly five times that of DDR; we have taken this number as an approximation.

We take McCalpin STREAM as a baseline figure for the Intel x86 and IBM Power architectures. This is implemented in OpenMP 3 and so is unable to run on GPUs. We run the STREAM kernels, implemented in each of the 7 programming models, on each of the 12 devices, where supported. Note that some configurations are not available due to unavailability of compilers and we denote these results as 'N/A' in the upcoming figures. A few results we were not able to obtain because of library or system incompatibility: we denote these as 'X'. We use an array size of 2^{25} double precision elements and run 100 iterations of each kernel.

Figure 1 shows the fraction of theoretical peak performance obtained by each device, for each implementation in the different programming models of the triad kernel. Firstly note that C++ OpenMP running on CPUs takes a performance hit over the C OpenMP implementation of McCalpin STREAM. The C++ based approaches of RAJA and Kokkos demonstrate performance similar to the close-to-the-metal performance of CUDA and OpenCL on the NVIDIA GPUs, however neither support AMD GPUs. Both models use OpenMP for CPU support and demonstrate that there is little overhead using their abstractions over a directive based approach.

Both OpenMP and OpenACC show good performance on NVIDIA GPUs, with OpenACC also running well on the AMD S9150 (Hawaii architecture) GPU. OpenACC in general shows poor performance compared to OpenMP and McCalpin STREAM on the CPUs however.

Figure 2 shows the raw performance of the triad kernel. Note that the MCDRAM available to the KNL offers the highest achievable memory bandwidth of the devices we tested, with the AMD Fury X coming close in second (the only GPU tested with High Bandwidth Memory). GPUs still

TABLE I
LIST OF DEVICES

Name	Peak Memory BW (GB/s)
NVIDIA K20X GPU	250
NVIDIA K40 GPU	288
NVIDIA K80 GPU (1 GPU)	240
NVIDIA GTX 980 Ti GPU	224
AMD S9150 GPU	320
AMD Fury X GPU	512
Intel E5-2670 (Sandy Bridge) CPU	2×51.2=102.4
Intel E5-2697 v2 (Ivy Bridge) CPU	2×59.7=119.4
Intel E5-2698 v3 (Haswell) CPU	2×68=136
Intel E5-2699 v4 (Broadwell) CPU	2×76.8=153.6
Intel Xeon Phi (Knights Landing) 7210	∼5×102 = 510
IBM Power 8 CPU	2×192=384

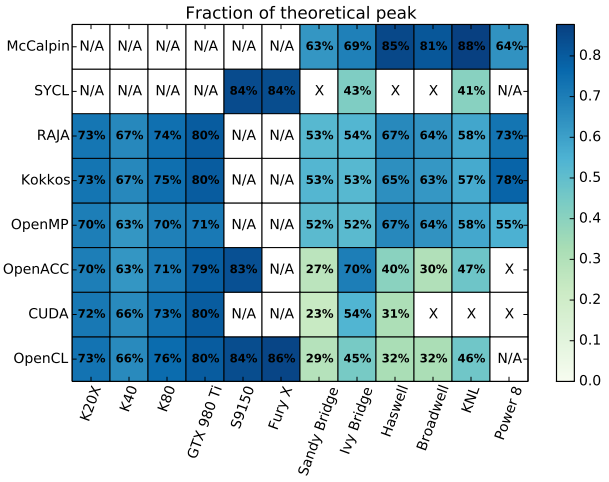

Fig. 1. Fraction of peak performance

offer an increased memory bandwidth over CPUs in general, however the IBM Power 8 CPU provides more bandwidth than any of the NVIDIA GPUs.

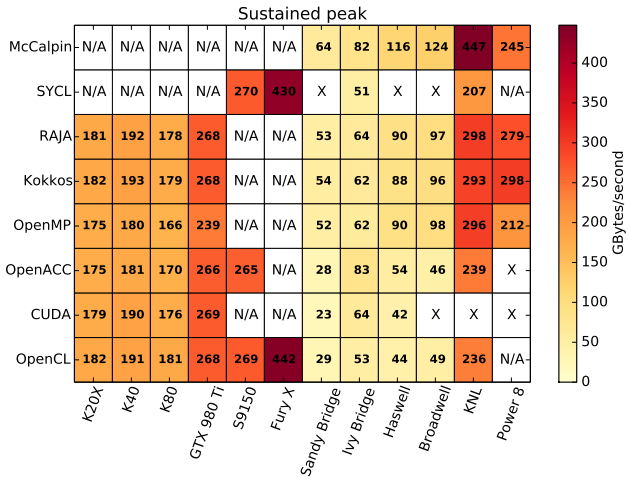

Fig. 2. Peak performance

GPU-STREAM is Open Source and available on GitHub at `github.com/UoB-HPC/GPU-STREAM`. The webpage maintains a repository of all our results and we encourage new submissions.

REFERENCES

[1] J. D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, dec 1995.

[2] T. Deakin and S. McIntosh-Smith, "GPU-STREAM: Benchmarking the achievable memory bandwidth of Graphics Processing Units (poster)," in *Supercomputing*, Austin, Texas, 2015.

[3] T. Deakin, J. Price, M. Martineau, and S. McIntosh-Smith, "GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models," in *P^3MA workshop at International Super Computing*, 2016.