IWOCL / SYCLcon 2020

University of BRISTOL

Evaluating the performance of HPC-style SYCL applications

Tom Deakin
and Simon McIntosh-Smith

uob-hpc.github.io

Introduction

- SYCL was first released in 2014.
- Recent development of different implementations providing support for devices used in the HPC space.

- Platforms:
  - Intel Xeon Skylake and Iris Pro GPUs
  - NVIDIA RTX 2080 Ti GPU
  - AMD Radeon VII GPU

- Try out three different compilers:
  - Codeplay's ComputeCpp
  - Intel's oneAPI DPC++
  - Heidelberg University's hipSYCL

Platforms

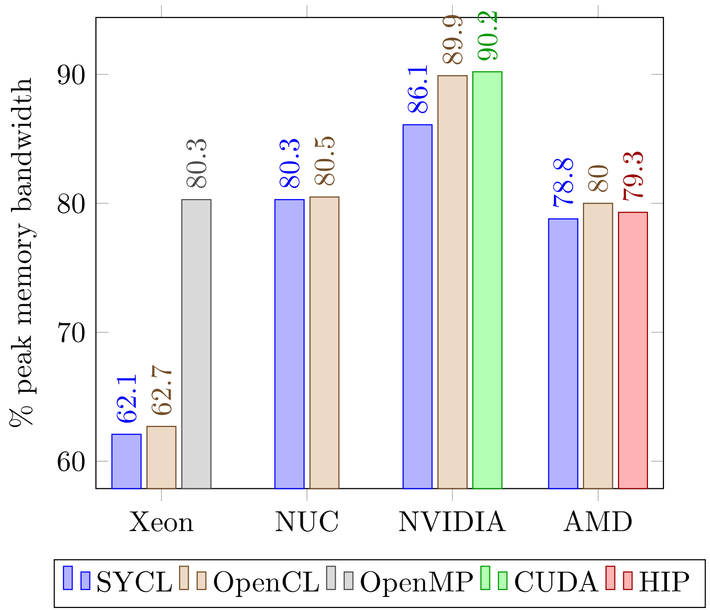
Name	Architecture	Device Type	Mem. BW (GB/s)
Intel Xeon Gold 6126 (12-core)	Skylake	HPC CPU (1 socket)	119.21
Intel NUC i7-6770HQ with Iris Pro 580 Graphics	Skylake/Gen9	CPU + Integrated GPU	34.1
AMD Radeon VII	Vega 20	Discrete GPU	1024
NVIDIA RTX 2080 Ti	Turing	Discrete GPU	616

Applications

- Three applications:
  - BabelStream
    - Copy kernel: `c[i] = a[i];`
    - Triad kernel: `a[i] = b[i] + scalar * c[i];`
    - Dot kernel: `sum += a[i] * b[i];`
  - Heat
    - Simple explicit finite difference solve.
    - 5-point stencil.
  - CloverLeaf
    - 2D structured grid Lagrangian-Eulerian hydrodynamics code.
- All are main memory bandwidth bound, like many other HPC applications today.

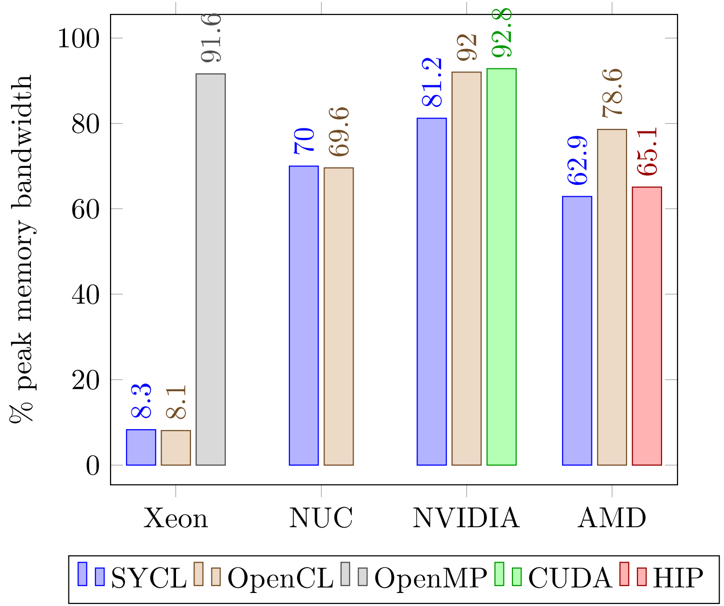
BabelStream: Triad

- Results are shown as percentage of theoretical peak bandwidth, so higher is better.
- SYCL shows little overhead over direct implementations in the underlying models, particularly on the GPUs.
- Intel OpenCL runtime still showing known performance gap with OpenMP on Xeon platforms.



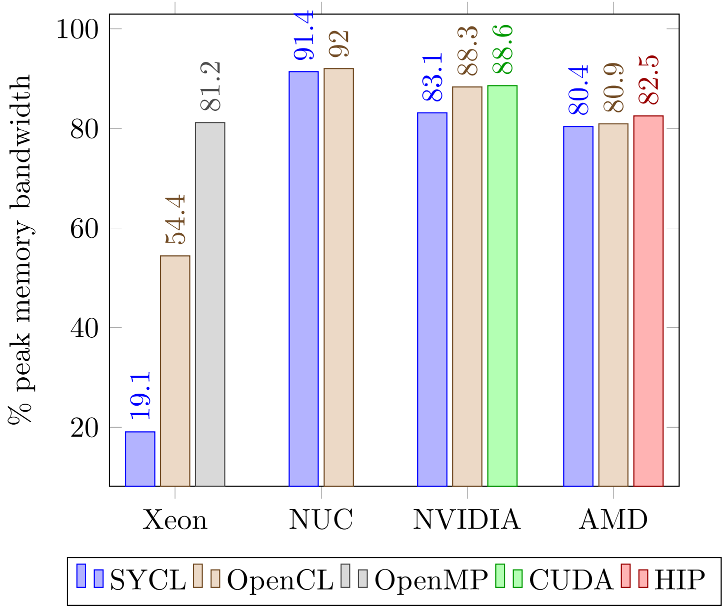
BabelStream: Dot

- For SYCL, OpenCL, CUDA and HIP, we implemented a global reduction by hand as they don't have one built in.
- Do see some performance loss in the SYCL version compared to what is possible on the platforms.
- SYCL performance matches underlying implementations in most cases.



BabelStream: Copy

- Memory copy kernel, with no floating point operations.
- Heat application should behave similarly to this kernel.
- See good and consistent performance on all the GPUs.
- Observe large range of performance on the Xeon CPU.



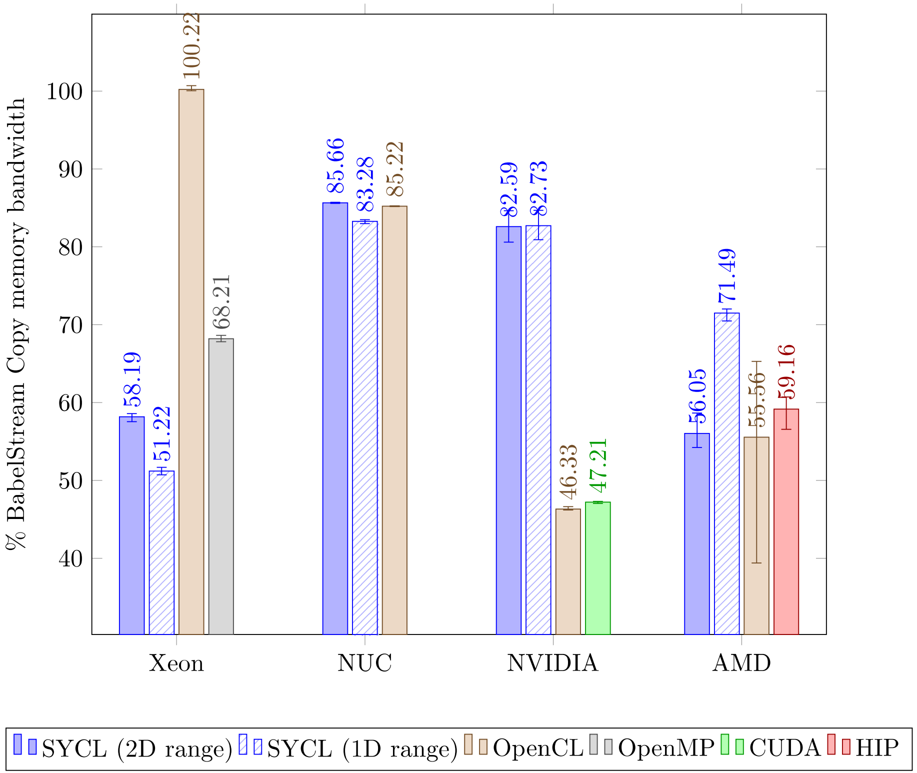
Heat: average performance

- Two SYCL versions:
  - 2D range:
    `parallel_for<…>(range<2>{n,n},…)`
    `acc[j][i]`
  - 1D range:
    `parallel_for<…>(range<1>{n*n},…)`
    `acc[j+i*n]`
- Consistent performance on NUC and AMD.
- Xeon performance mirrors that of BabelStream Copy.
- NVIDIA platform shows issues with underlying models, possibly driver related.

Platform	Model	Runtime (s)	Mem. BW (GB/s)
Xeon	SYCL (2D range)	77.17	13.27
	SYCL (1D range)	87.64	11.68
	OpenCL	15.71	65.04
	OpenMP	15.52	65.99
NUC	SYCL (2D range)	38.34	26.71
	SYCL (1D range)	39.44	25.97
	OpenCL	38.31	26.73
NVIDIA	SYCL (2D range)	2.28	449.50
	SYCL (1D range)	2.27	450.23
	OpenCL	4.06	252.13
	CUDA	3.97	257.80
AMD	SYCL (2D range)	2.23	461.13
	SYCL (1D range)	1.74	588.20
	OpenCL	2.26	460.32
	HIP	2.09	490.17

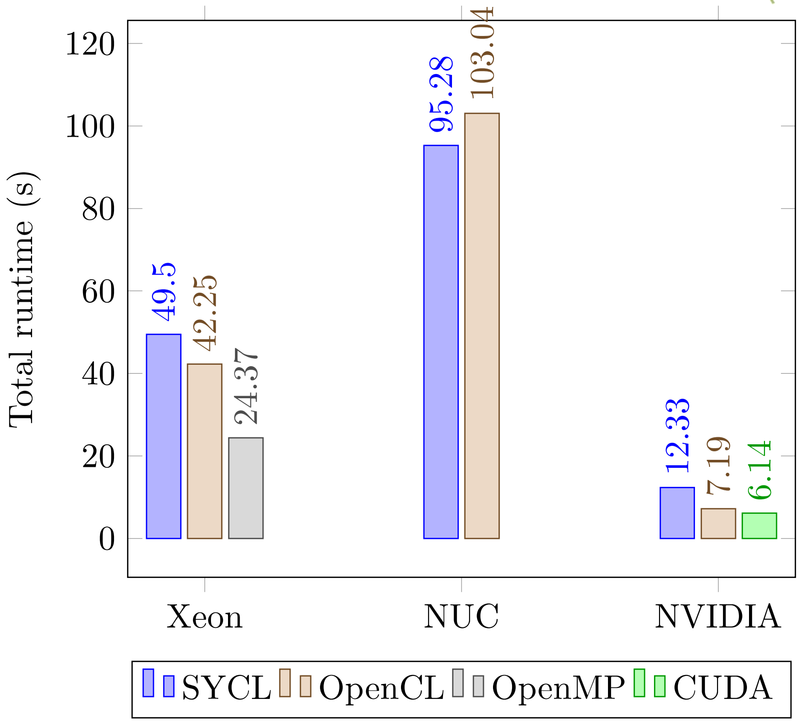
Heat: comparison to Copy

- Compare to performance of Copy as measured for each model.
- On Xeon see about 60% of attainable Copy bandwidth.
- Consistent performance on NUC.
- AMD shows high variability.
- This chart highlights the performance issues with CUDA and OpenCL on NVIDIA.



CloverLeaf

- Chart shows runtime, lower is better.
- SYCL within 10% of OpenCL performance.
- Reduction cause of performance gap on NVIDIA.
- The OpenCL runtime needs improvement on Xeon in order to SYCL to achieve it’s potential as a parallel programming model of choice.



Summary

- Often possible to write SYCL applications that get good performance across a number of platforms.
- SYCL performance close to lower level model such as OpenCL.
- All the source code is available online, at our GitHub page.
- Widespread and robust support from all vendors is needed now to ensure SYCL is a success for the HPC community.