On measuring the maturity of SYCL implementations by tracking historical performance improvements

Wei-Chen Lin
and Tom Deakin, Simon McIntosh-Smith

uob-hpc.github.io
Introduction

Representative set of HPC style mini-apps
- Memory-bandwidth bound:
  - BabelStream
  - CloverLeaf (complex, high kernel count)
- Compute-bound
  - BUDE

Historical SYCL compilers
- ComputeCpp (Codeplay)
  - Jul. 2018 ~ Nov. 2020
- oneAPI DPC++ (Intel)
- hipSYCL (Heidelberg University)
  - Sep. 2019 ~ Jan. 2021

Alternative HPC frameworks
- CUDA
- OpenCL
- OpenMP
- Kokkos

Benchmark on:
- Intel Cascade Lake Xeon CPU
- AMD Rome CPU
- Intel Gen 9.5 GPU
- NVidia V100 GPU
SYCL landscape

<table>
<thead>
<tr>
<th></th>
<th>DPC++</th>
<th>ComputeCpp</th>
<th>hipSYCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP (CPU)</td>
<td>●</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>OpenCL SPIR (CPU/GPU)</td>
<td>●</td>
<td>●</td>
<td>○\textsuperscript{a}</td>
</tr>
<tr>
<td>Intel Level Zero (CPU/GPU)</td>
<td>●</td>
<td>○</td>
<td>○\textsuperscript{a}</td>
</tr>
<tr>
<td>Nvidia (GPU)</td>
<td>○\textsuperscript{b}</td>
<td>○\textsuperscript{c}</td>
<td>●\textsuperscript{d}</td>
</tr>
<tr>
<td>AMD ROCm (GPU)</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

\textsuperscript{a} Built on DPC++, experimental and work in progress
\textsuperscript{b} PTX, experimental
\textsuperscript{c} PTX, experimental, incomplete and discontinued
\textsuperscript{d} CUDA
# Hardware platform

<table>
<thead>
<tr>
<th>Name</th>
<th>Architecture</th>
<th>Short name</th>
<th>Device Type</th>
<th>Peak Mem. BW (GB/s)</th>
<th>Peak FP32 FLOP/s (GFLOP/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIDIA Tesla V100</td>
<td>Volta</td>
<td>v100-isambard</td>
<td>Discrete GPU</td>
<td>900</td>
<td>14000</td>
</tr>
<tr>
<td>Intel UHD P630 (Intel Xeon E2176G)</td>
<td>Gen9.5</td>
<td>uhdp630-devcloud</td>
<td>Integrated GPU + CPU</td>
<td>42.6</td>
<td>460</td>
</tr>
<tr>
<td>Intel Xeon Gold 6230 (20-cores)</td>
<td>Cascade Lake</td>
<td>cxl-isambard</td>
<td>HPC CPU (2-socket)</td>
<td>281.6</td>
<td>4096</td>
</tr>
<tr>
<td>AMD EPYC 7742 (64-cores)</td>
<td>Zen2 (Rome)</td>
<td>rome-isambard</td>
<td>HPC CPU (2-socket)</td>
<td>409.6</td>
<td>9216</td>
</tr>
</tbody>
</table>
BabelStream

- Port of the STREAM benchmark to many languages, SYCL included
- Memory-bandwidth bound
- Single kernel; repeated for consistency
- Measurements in GB/s
- Source code publicly available on GitHub
  - https://github.com/UoB-HPC/BabelStream

Algorithm 1 BabelStream Triad kernel

1: procedure TRIAD(a[], b[], c[], scalar, n) → a, b, c are arrays of size n
2:   for i ← 0, n do
3:     a[i] ← b[i] + scalar * c[i]
Results: BabelStream

- ComputeCpp: ~ 70% performance of alternative frameworks
- DPC++: Highly consistent
- hipSYCL: Major performance uplift after version 0.9
miniBUDE

- Proxy application of the Bristol University Docking Engine (BUDE)
- Compute bound
- Single kernel; repeated for consistency
- No hierarchical parallelism
- Measurements in GFLOPS/s
- Source code publicly available on GitHub
  - https://github.com/UoB-HPC/bude-portability-benchmark
Results: BUDE

- ComputeCpp: Performance regressions resolved recently
- DPC++: Highly consistent, competitive
- hipSYCL: Consistently low performance on the CPU end
CloverLeaf

- Proxy application for 2D hydrodynamics
- Memory-bandwidth bound; kernel traverses structured grid
- 170+ unique kernels, largest codebase (8kLOC)
- Measurements in total runtime for 2995 iterations
- Source code publicly available on GitHub
  - https://github.com/UoB-HPC/cloverleaf_sycl
Results: CloverLeaf

- **ComputeCpp**: Major improvements (~80%)
- **DPC++**: Highly consistent
- **hipSYCL**: Significant improvements (~40%)

Alternative frameworks

Note: normalised runtime per chart; values are not comparable across compilers
Summary

- All SYCL implementations approaching maturity
  - SYCL2020 alignment
  - Improved software and hardware platform support
  - Stabilised performance, trending upwards
- All mini-apps are on publicly available GitHub repositories