

IWOCL 2026



SYCL Everywhere: An Open Source Ecosystem for Parallel Computing

Ben Ashbaugh, Intel Corporation

With contributions from many...



“Why do we think about GPU compute, and not just compute?”

SYCL

“Why do we think about ~~CPU~~
~~compute~~, and not just ~~compute~~?”
programming programming

Claims and Motivations:

SYCL is still a niche technology:

Need to download special compilers

Need to download special runtimes

Niche-ness is hurting adoption

SYCL everywhere is a desirable goal

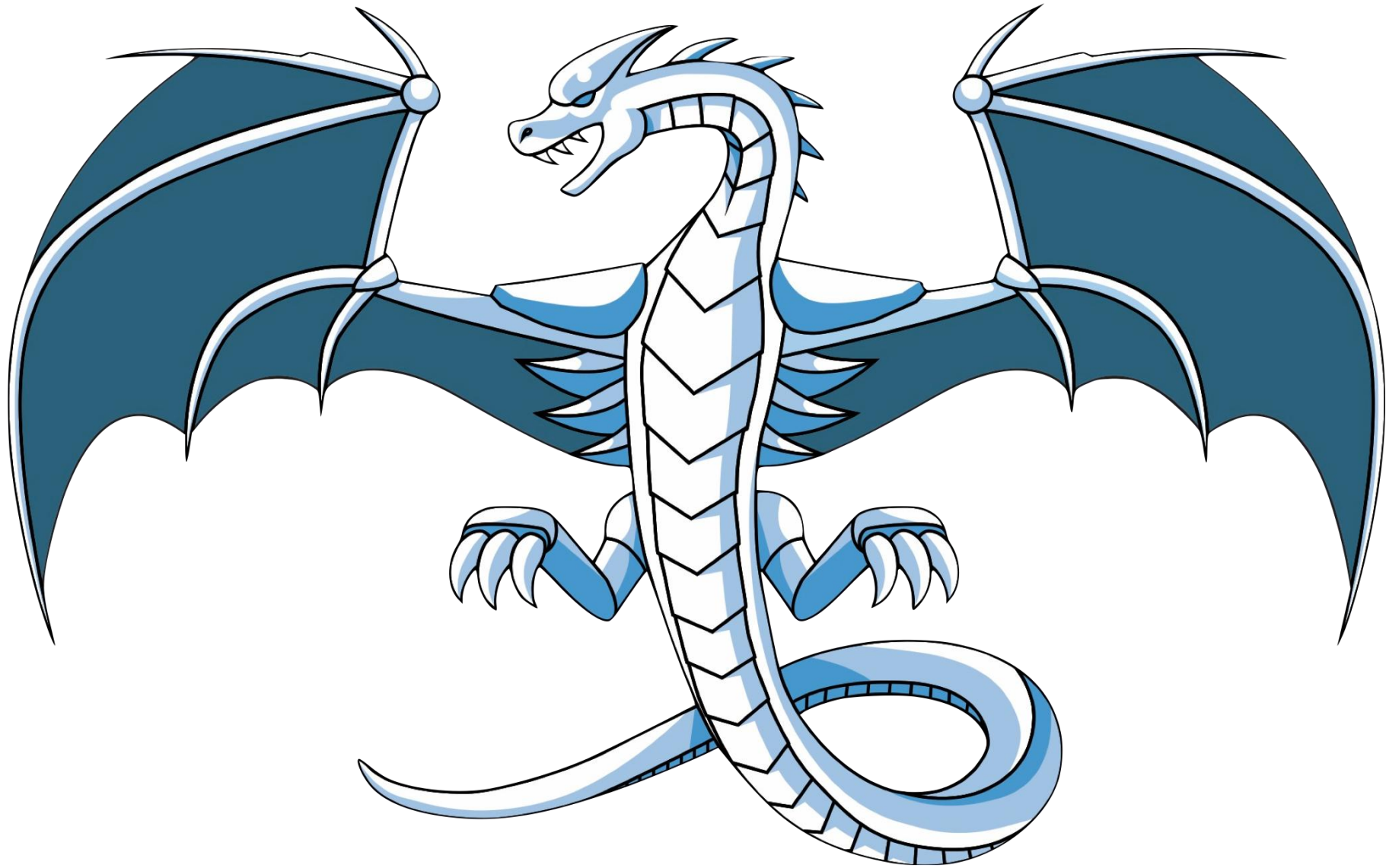
SYCL everywhere is an attainable goal!

Open source!
↙

Caveats and Disclaimers

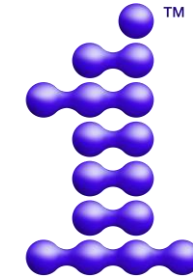
- More SYCL implementations is “a good thing”
 - This will not describe the only way to implement SYCL!
- Focus will be on Khronos technologies
 - This is what I know most about 😊
- These activities are happening now
 - However, it is still early, things can change
 - More participation and voices are welcome!





Clang and LLVM

- Clang and LLVM is a natural place to add SYCL support
- Has a vibrant community and broad availability
- Both mainstream SYCL compilers are Clang and LLVM based



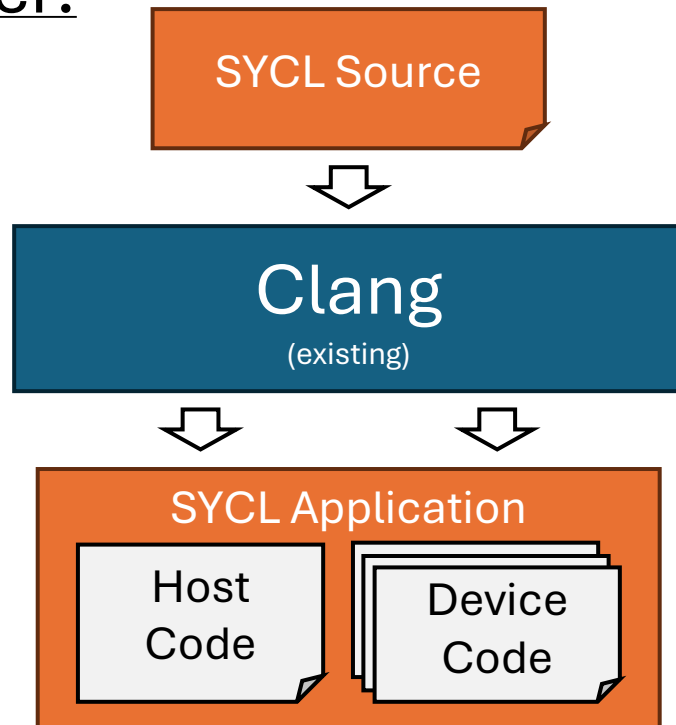
oneAPI

AdaptiveC++

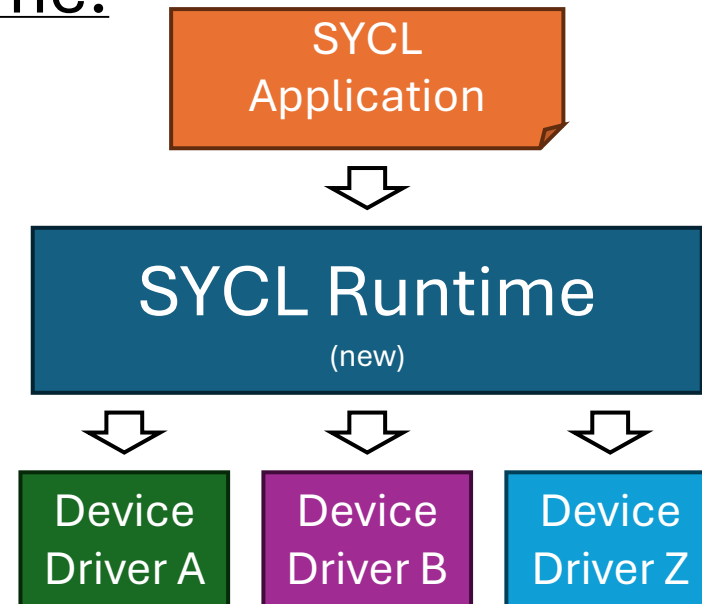
SYCL Clang and LLVM Goals

- SYCL compiler and runtime support in the upstream LLVM repo
 - No forks needed!

Compiler:



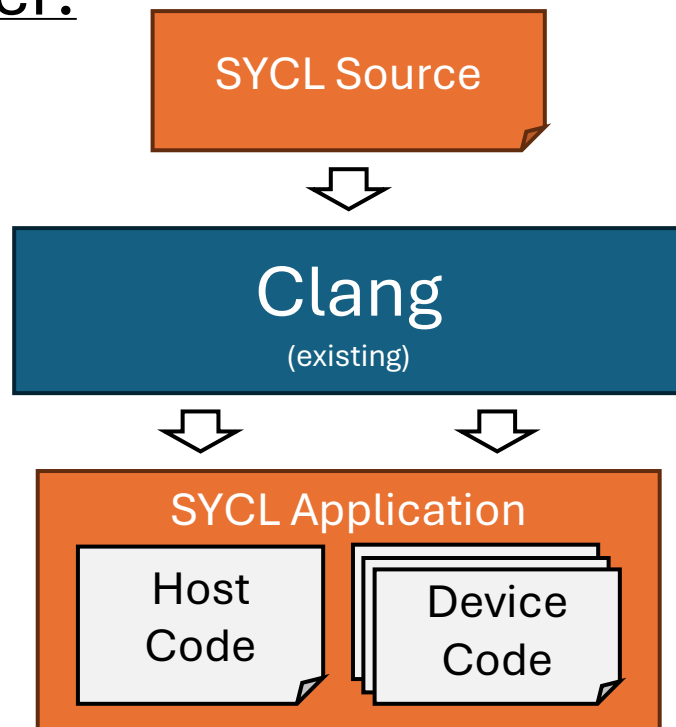
Runtime:



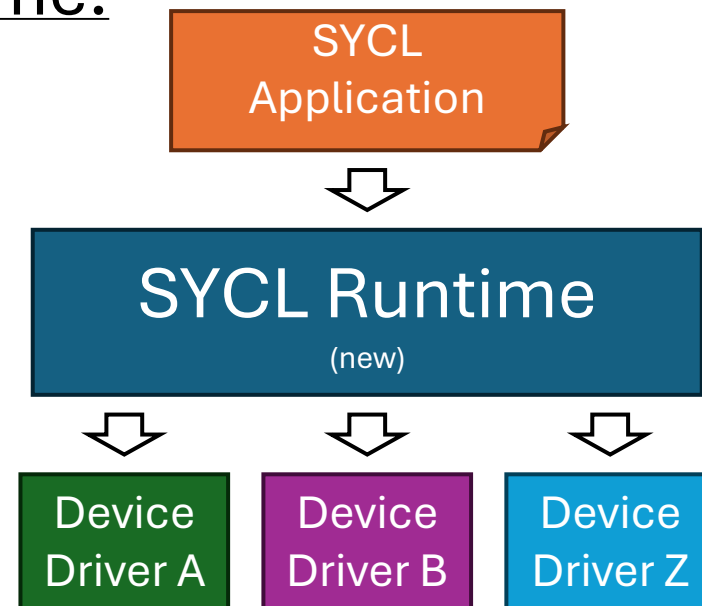
SYCL Clang and LLVM Goals

- Let's start by looking at the SYCL compiler...

Compiler:



Runtime:



Clang Details

- Lots of important details are being designed and implemented...
 - Command line options
 - Identifying and separating host and device code
 - Packaging host code and device code into an executable
 - ... and more!
- Focus of this talk is SPIR-V
 - Specifically, the “SPIR-V LLVM backend”

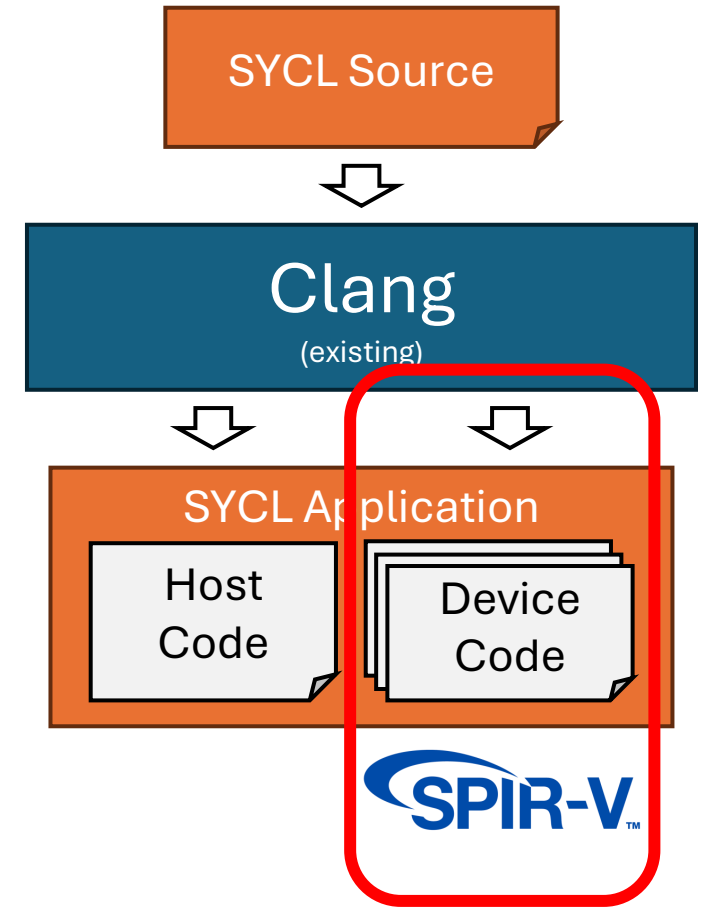


SPIR-V LLVM Backend: What is it?

- An upstream mechanism to generate **SPIR-V** from **LLVM IR**
 - Supported as an official LLVM target since January 2025
- **LLVM IR** is the IR used internally by Clang and LLVM
 - Useful and expressive, but not a stable distribution format
- **SPIR-V** is the **Sandard Portable Intermediate Representation**
- A Khronos standard IR to represent device code
- Stable, accepted by device compilers from many vendors

SPIR-V LLVM Backend

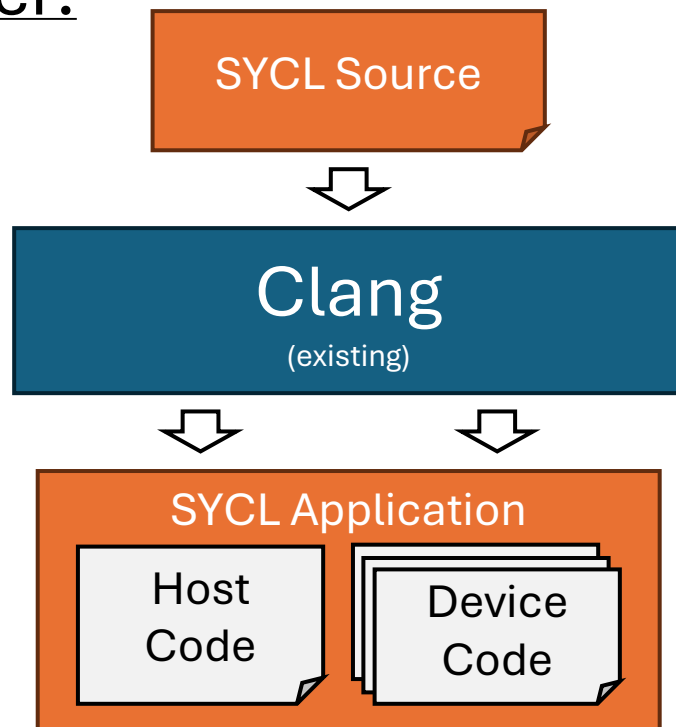
- Provides a mechanism to generate standard device code
- Notes:
 - **A** mechanism, not the only mechanism! Some devices may support other device code formats
 - The SPIR-V LLVM backend is useful for more than just SYCL!



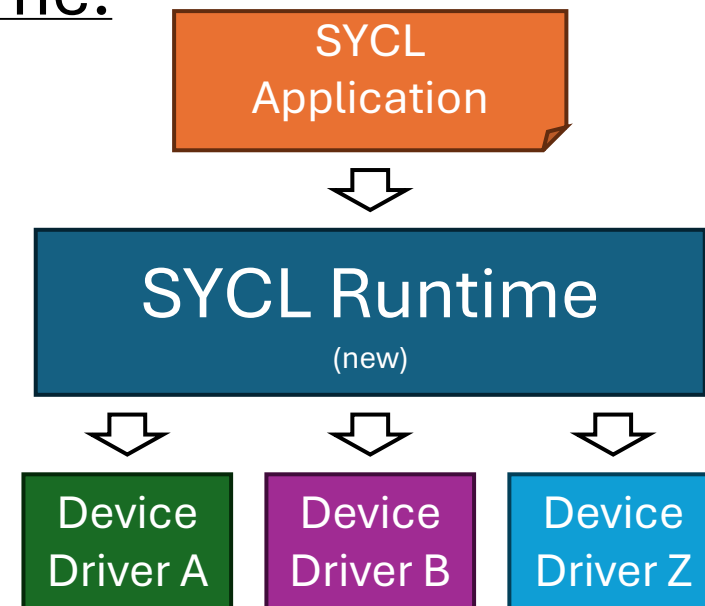
SYCL Clang and LLVM Goals

- Let's shift focus to the SYCL runtime...

Compiler:



Runtime:



Runtime Details

- LLVM has an offload runtime!
 - Used for OpenMP offload - **libomptarget**
 - Supports devices from multiple vendors
 - Can this be repurposed to execute SYCL applications?

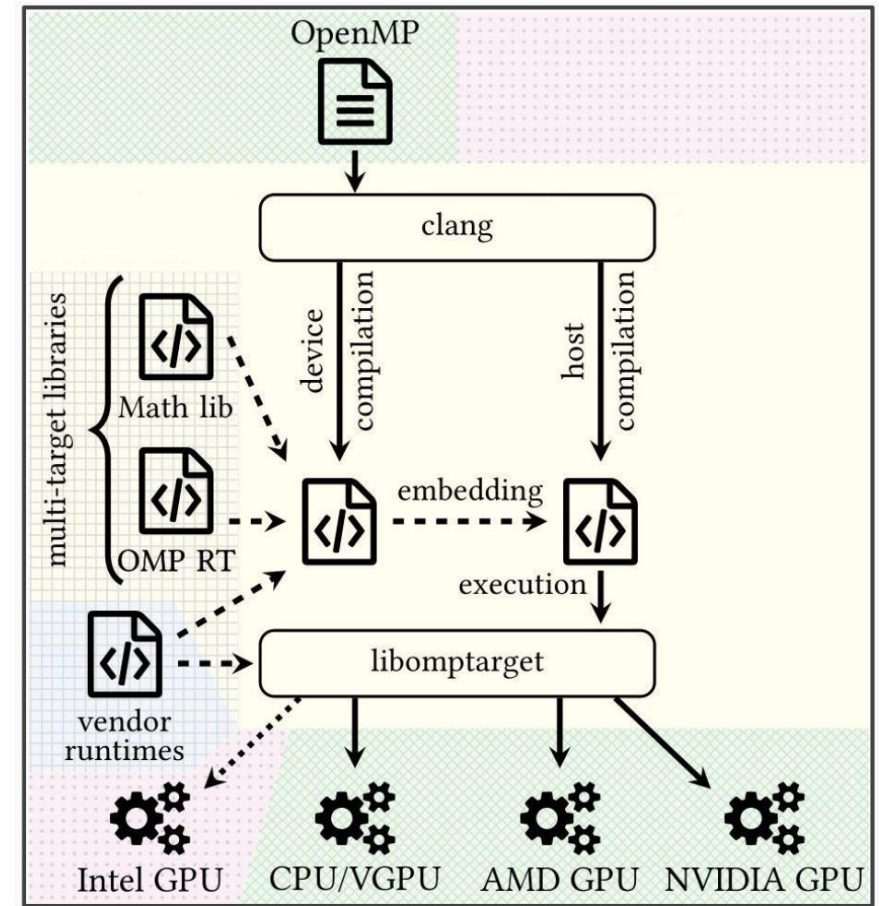
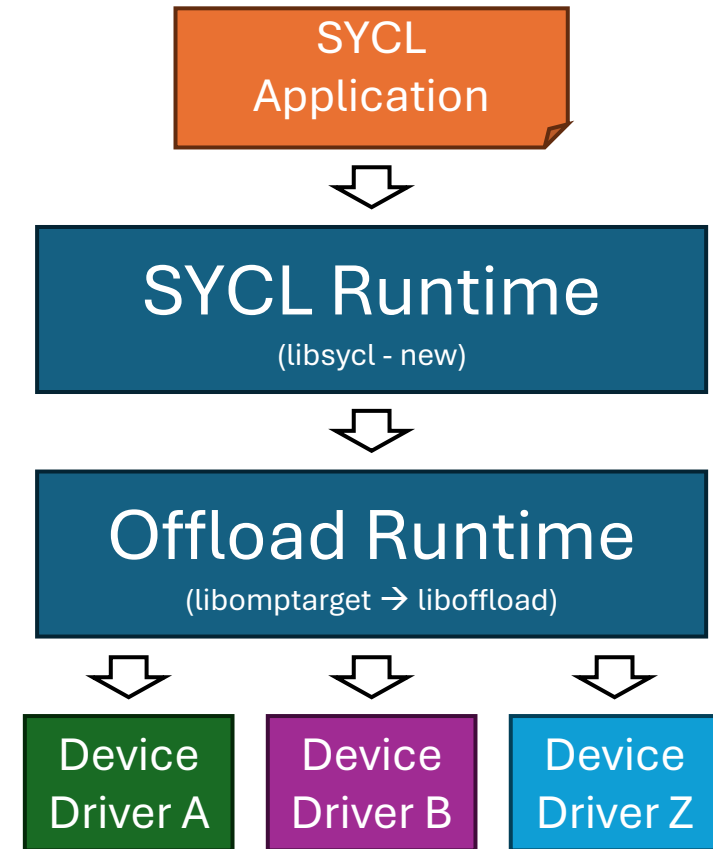


Diagram source: <https://llvm.org/devmtg/2023-05/slides/QuickTalks-May10/04 - EuroLLVM 23 - OpenMP Kernel Language.pdf>

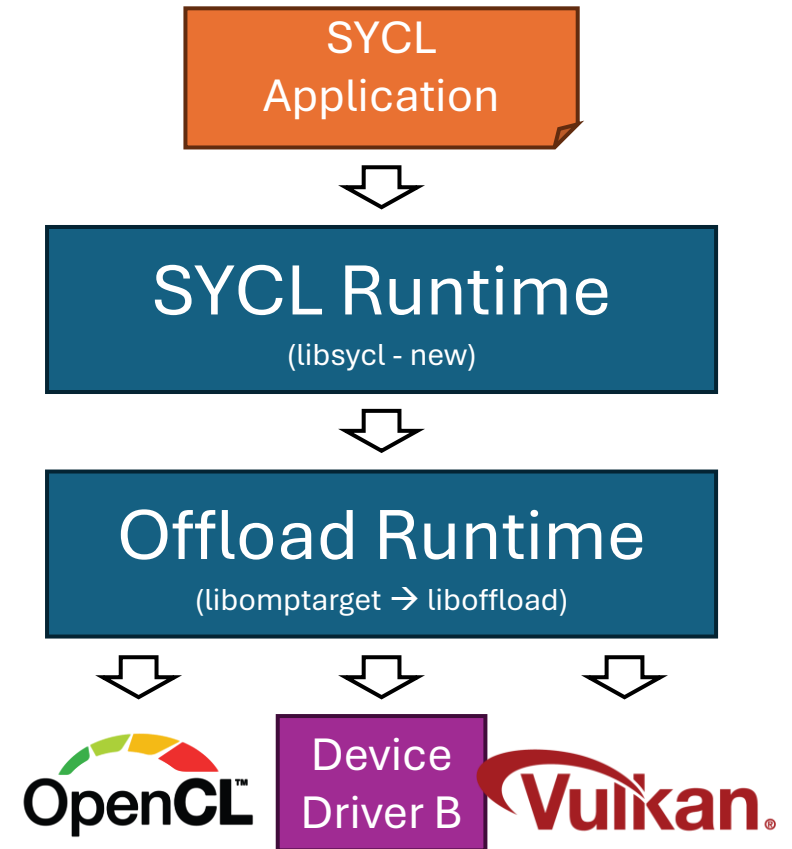
Runtime Details

- Short answer: Yes!
- **libomptarget** becomes **liboffload**
 - A shared offload runtime for OpenMP, SYCL, and more
 - Continues to support devices from multiple vendors
 - Continues to support multiple device types (GPUs, CPUs, FPGAs, and more)
 - Via multiple driver APIs



Runtime Details: Khronos APIs

- There are several “obvious” Khronos APIs that liboffload could target
 - OpenCL – likely future target
 - Vulkan – possible future target?
- Note:
 - Both OpenCL and Vulkan need extensions for full SYCL support
 - Both APIs consume SPIR-V! (kind of)



Current Status

Current Status (as of April 2026):

- Have community agreement, development has started
- Enabling SYCL “hello, world”
 - Compiling to SPIR-V
 - Running on an Intel GPU
 - On both Windows and Linux
- Not *quite* working yet...
 - But making lots of progress!
 - Expecting to work “soon”

```
#include <iostream>
#include <sycl/sycl.hpp>

int main() {
    // Create a queue to enqueue work for the default device
    sycl::queue myQueue;

    // Allocate memory for the device associated with the queue
    int* data = sycl::malloc_shared<int>(1024, myQueue);

    myQueue.parallel_for(1024, [=](sycl::id<1> idx) {
        // Initialize each element with an index starting at zero
        data[idx] = idx;
    });

    // Explicitly wait for kernel execution to complete
    myQueue.wait();

    // Print results
    for (int i = 0; i < 1024; i++)
        std::cout << "data[" << i << "] = " << data[i] << std::endl;

    // Free memory
    sycl::free(data, myQueue);
    return 0;
}
```

What's next?

- Add more SYCL features!
 - Prioritizing features used by real-world applications ([HeCBench](#))
- Evaluate performance
 - Especially through liboffload
- Eventually, possibly:
 - SYCL Conformance?

The screenshot displays the GitHub repository for HeCBench. At the top, it shows the repository name 'HeCBench' with a 'Public' label, and statistics for 'Watch' (4), 'Fork' (107), and 'Star' (286). Below this, there's a navigation bar with 'master' branch selected, '2 Branches', and '0 Tags'. A search bar and 'Add file' and 'Code' buttons are also present.

The main content area shows a list of recent commits by 'zjin-lcf'. The commits include:

- .dvc: [DVC] Add DVC remote to pull data artifacts. (3 weeks ago)
- cmake: [cmake_build] uncompress .tar.bz2 file using tar as bzip2 w... (3 weeks ago)
- results: update the note with HIPCL (3 weeks ago)
- src: [vadd-hip] Use warpSize built-in instead of macro. (2 days ago)
- tools: clean up some misc files from cmake conversion (3 weeks ago)
- .dvcignore: Add the CLINK examples (3 weeks ago)
- .gitignore: Add CMake build system infrastructure (3 weeks ago)
- CMAKE_BUILD.md: [cmake_build] update the comment on the cmake configure f... (2 weeks ago)
- CMakeLists.txt: [cmake_build] Find NVHPC NCCL module with the PyTorch Fi... (3 weeks ago)
- CMakePresets.json: [cmake_build] address SYCL build errors (3 weeks ago)
- LICENSE: Revise the copyright statement (3 weeks ago)
- README.md: [readme] acknowledge the researcher for reporting memory r... (4 days ago)
- benchmarks.yaml: [frechet] delete the programs (3 weeks ago)
- convert_benchmarks.py: Add CMake support for 63 additional benchmarks (209 imple... (3 weeks ago)

Below the commit list, there's a 'README' section with the BSD-3-Clause license. The README content includes:

HeCBench

This repository contains a collection of heterogeneous computing benchmarks written with CUDA, HIP, SYCL/DPC++, and OpenMP-4.5 target offloading for studying performance, portability, and productivity.

Background, use cases and future work

Z. Jin and J. S. Vetter, "A Benchmark Suite for Improving Performance Portability of the SYCL Programming Model," 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Raleigh, NC, USA, 2023, pp. 325-327, doi: 10.1109/ISPASS57527.2023.00041. (<https://ieeexplore.ieee.org/document/10158214>)

On the right side of the repository page, there's an 'About' section with a description, tags for 'benchmark', 'performance', 'openmp', 'cuda', 'hip', 'portability', 'sycl', 'parallel-programming', and 'heterogeneous-computing'. It also lists 'Readme', 'BSD-3-Clause license', 'Activity', 'Custom properties', '286 stars', '4 watching', '107 forks', and 'Report repository'. Below this, there's a 'Releases' section (No releases published), a 'Packages' section (No packages published), and a 'Contributors' section (24 contributors). At the bottom, there's a 'Languages' section with a bar chart showing the distribution of languages used in the repository:

Language	Percentage
C++	43.1%
C	23.9%
Roff	1.0%
Other	0.9%
Cuda	24.9%
Makefile	5.7%
Shell	0.5%

One more thing...

Open Source Drivers

- Can't have a fully open source solution without open source drivers...
- To the vendors that have open sourced their drivers – thank you!
- To the developers working on Mesa and PoCL open source implementations – thank you!
- Even without full open source solutions, we can still have a *mostly* open source solution built upon standards (which is pretty good!)



Wrap Up

Summary and Calls to Action



- SYCL is coming to Clang and LLVM!
 - Not a science project, work is happening right now
- For device vendors:
 - Contribute to the SPIR-V LLVM backend and liboffload
 - Enabling SYCL on your device may be easier than you think!
- For the Khronos SPIR-V and OpenCL (and Vulkan?) working groups:
 - Prioritize features used by the SPIR-V LLVM backend and liboffload
 - High ROI development – will be useful for more than just SYCL
- For application developers and SYCL fans:
 - Review, test, and advocate for SYCL in the LLVM community!

Thank you!