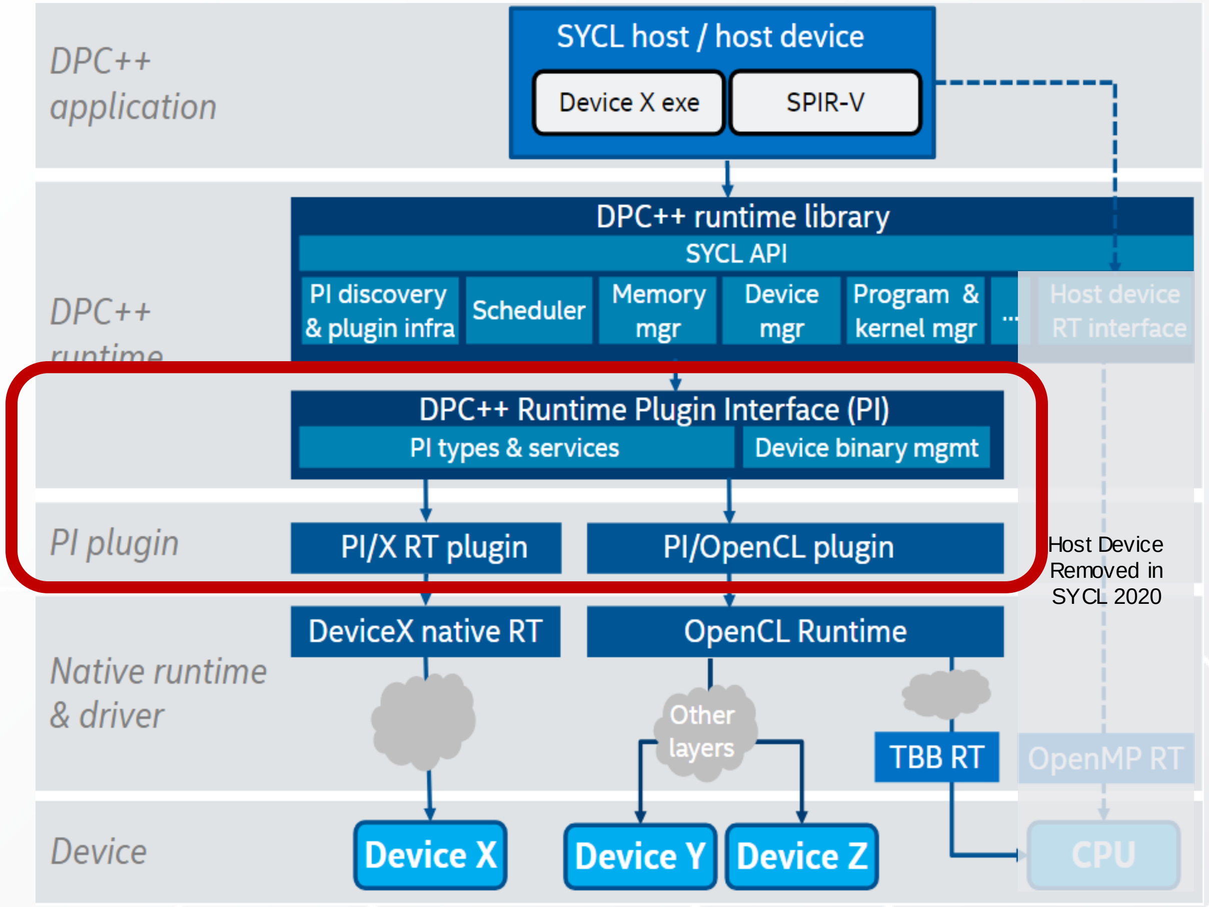
Unified Runtime

Topics

- Recap of Unified Runtime and goals
- Specification work
- Implementation work
- Unified Memory Architecture
- Future considerations

Specification

Recap: DPC++ Runtime Plugin Interface

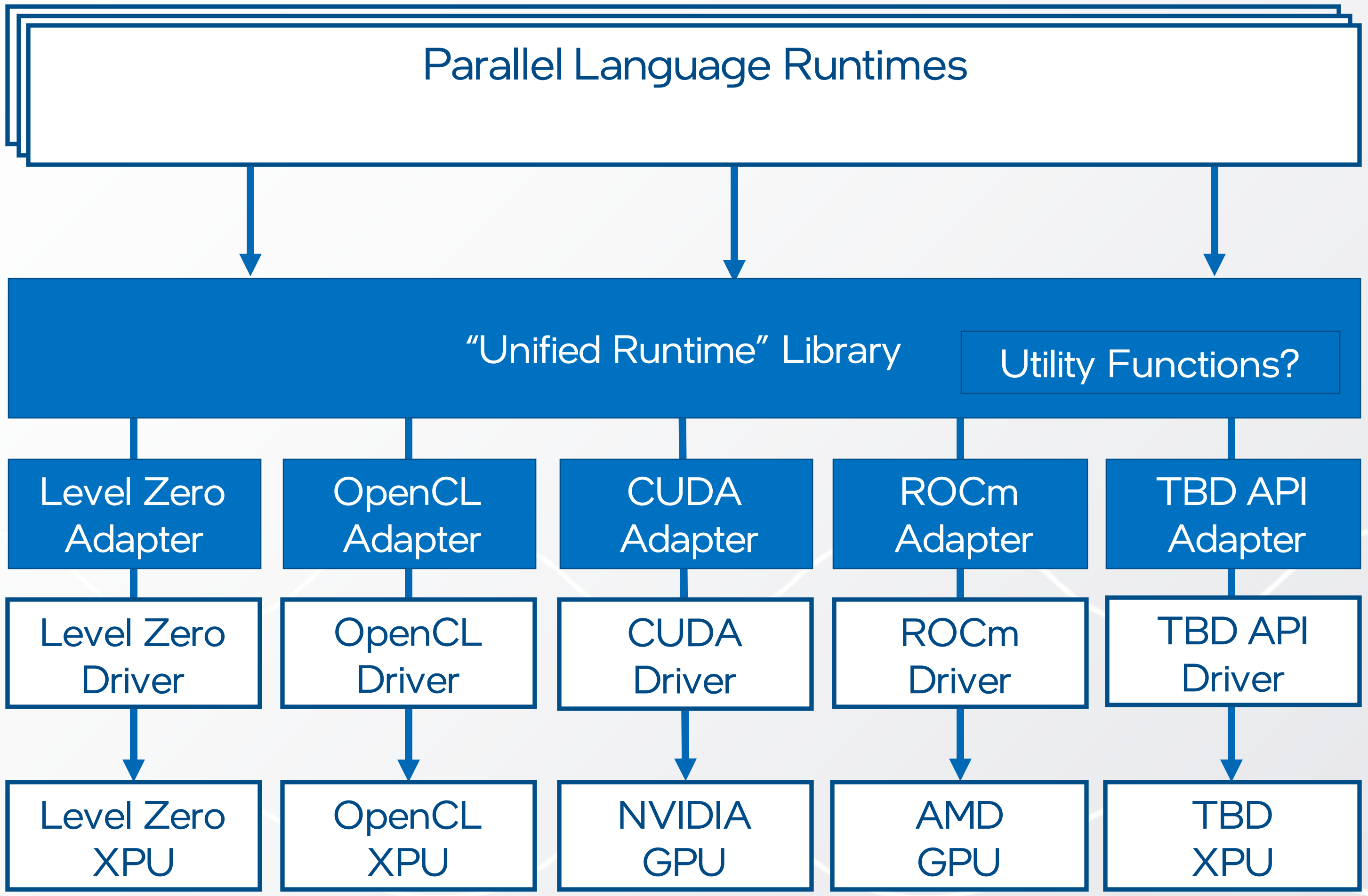


Problem Statement:

- Plugin Interface is an implementation detail
    - No specification
- Plugin Interface is only usable by the DPC++ Runtime
    - Other language runtimes end up duplicating the same functionality
    - Interop between languages on separate implementations requires a lot of work

Diagram Source: https://github.com/intel/llvm/blob/sycl/sycl/doc/design/PluginInterface.md

Discussion from Previous TABs: Unified Runtime

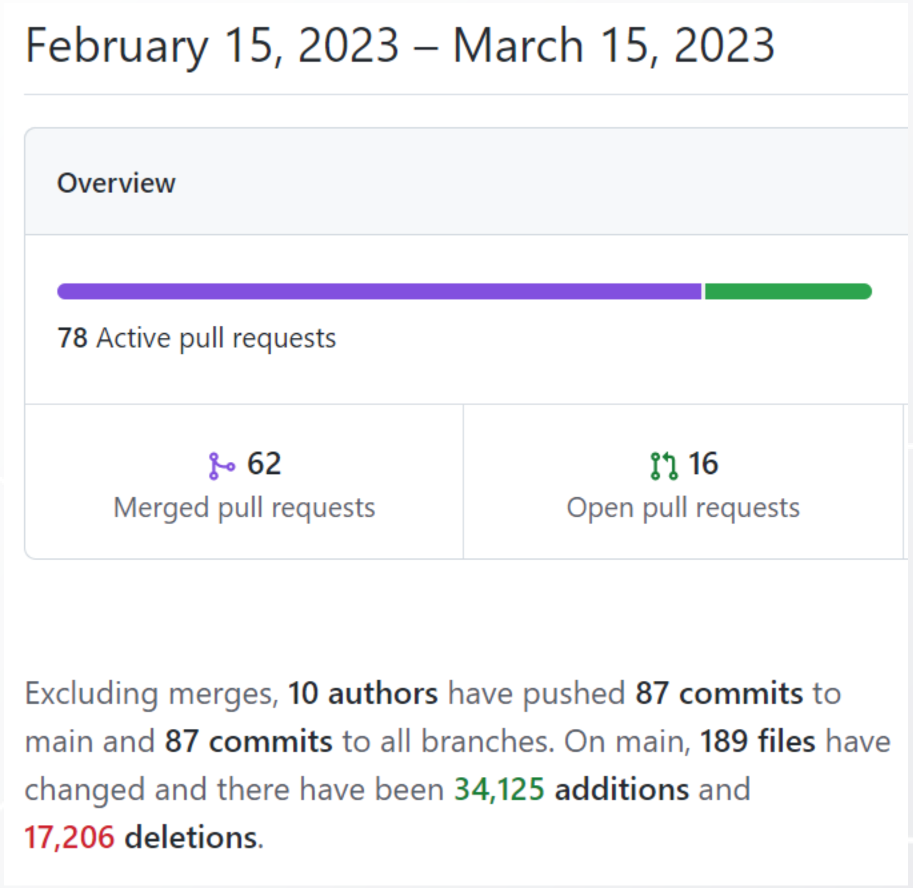


- Define a "Unified Runtime", usable by any Parallel Language Runtime, with a well-defined interface.
- Reviewed draft specification:
  - https://spec.oneapi.io/unified-runtime/latest/
  - Previously 0.5, now 0.6.

Goals of Unified Runtime

- Provide portable glue layer between language runtimes and close-to-the-metal native device APIs
- Allow efficient implementations regardless of native API
- Provide common ease-of-use features needed by language runtimes
- Support interoperability and access to native backends
- Have reasonable implementation cost for runtime users and adapter developers
- Tool and debug support
- Next steps:
    - Short-term, upgrade DPC++ SYCL runtime and experiment with OpenMP
    - Over time, possible to earn wide engagement by open-source community

Specification v0.6


February 15, 2023 – March 15, 2023

Overview

78 Active pull requests

62 Merged pull requests

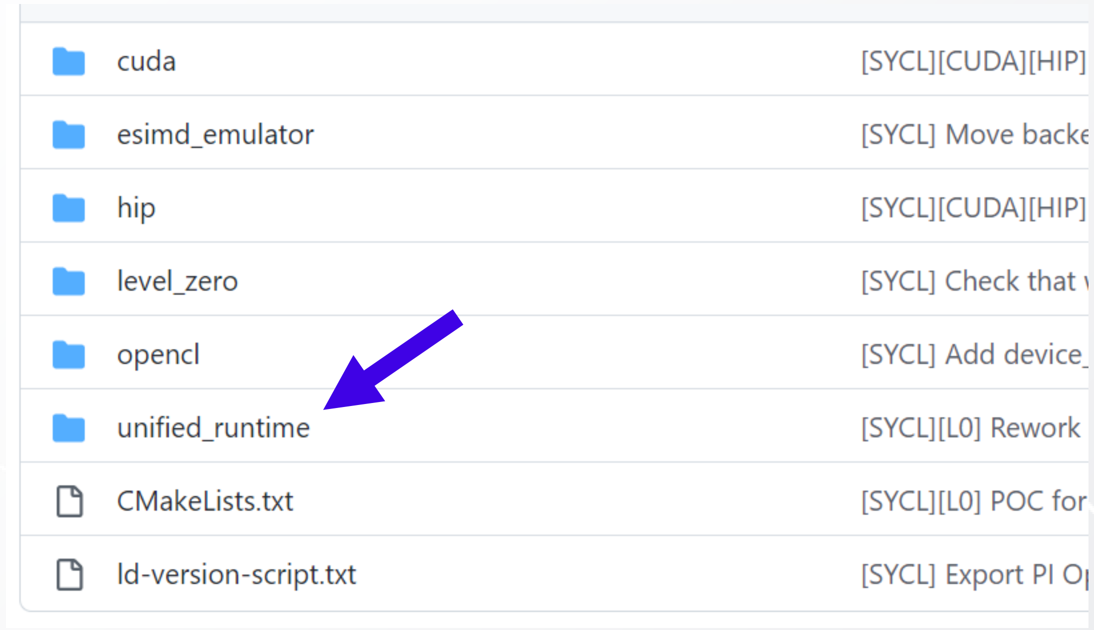
16 Open pull requests

Excluding merges, **10 authors** have pushed **87 commits** to main and **87 commits** to all branches. On main, **189 files** have changed and there have been **34,125 additions** and **17,206 deletions**.

- Specification work happens on GitHub directly
  - 192 PRs merged, 17 in review
  - 102 issues closed, 51 open
  - https://github.com/oneapi-src/unified-runtime
- Adapter and runtime porting in DPC++ codebase
  - https://github.com/intel/llvm
  - See PRs with "[UR]" tag
- A lot of specification changes are driven by keeping up with PI
  - DPC++, SYCL, and PI are always advancing, PI is not frozen, unified runtime matches each new PI change
- Other changes include mass rename to urFoo, large changes to compiler interface, changes to setting kernel arguments

Implementation

Experimental Implementation

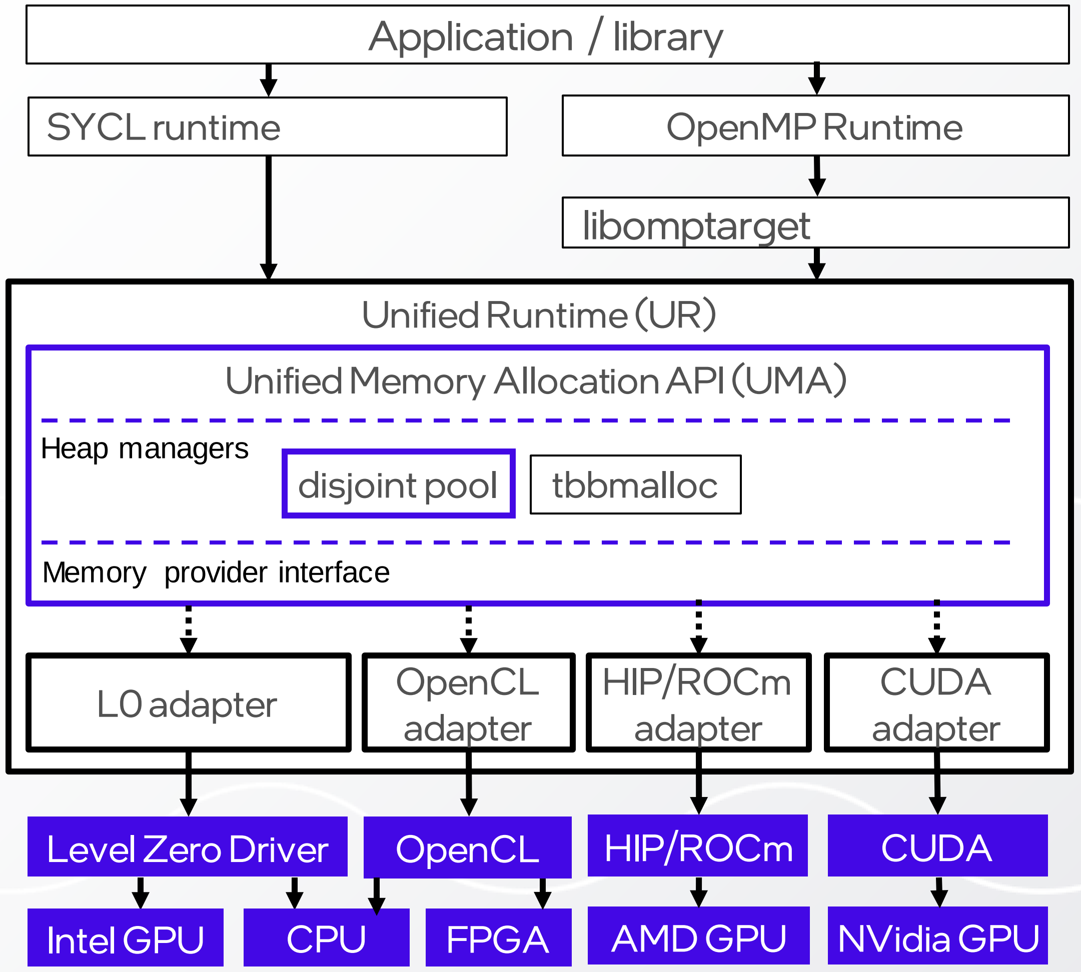


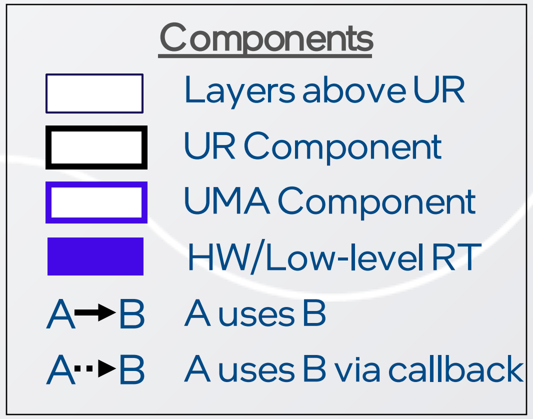
cuda	[SYCL][CUDA][HIP]
esimd_emulator	[SYCL] Move backe
hip	[SYCL][CUDA][HIP]
level_zero	[SYCL] Check that v
opencl	[SYCL] Add device_
unified_runtime	[SYCL][L0] Rework
CMakeLists.txt	[SYCL][L0] POC for
ld-version-script.txt	[SYCL] Export PI Op

https://github.com/intel/llvm/tree/sycl/sycl/plugins

- Implement Unified Runtime as a PI plugin initially
- Existing PI plugins get rewritten as UR adapters
  - Per-adapter choice, but so far all are planning to base PI and UR adapters on common code for transition
  - Currently porting L0 & CUDA adapters
  - Next comes OpenCL & HIP
- Finally, once PI and UR have parity, refactor SYCL-RT to use UR directly

Unified Memory Architecture





- UR provides interfaces for memory allocations
- UR employs UMA for memory pooling
- UMA provides a unified interface for heterogenous memory allocations
  - Decouple pool management from memory providers
- Adapters act as memory providers for coarse-grain allocations
- UMA is part of the upcoming UR release. Going to be a separate component in the next releases.

Future Considerations

License

- Current license on Unified Runtime repository is MIT
  - Covers common code and specification source, will cover adapters

- Project will involve taking code from DPC++ SYCL-RT, and wants long-term option of collaborating with upstream LLVM
  - All "Apache License v2.0 with LLVM Exceptions" license
  - So could change the license to match now, while project is small

- If this may negatively affect you please let us know
  - https://github.com/oneapi-src/unified-runtime/issues/363

Community Collaborations

- Previous TAB meetings asked us to look at OpenCL 3.0 and on-going work in LLVM libomptarget
  - Previous TAB in November covered OpenCL 3.0

- Short-term goal is to update SYCL runtime implementation, and investigate OpenMP
  - Learn more about how the API should look like through this
  - Improving the implementation and features of current compiler

- Bigger ambitions for the future
  - Produce widely used library and tooling for implementing language runtimes on a range of interfaces and hardware
  - Work with community on how best to achieve that

Summary

- Unified runtime specification and implementation are under current, active development

- Great time to give us any feedback!
  - Comments on recent specification changes: https://spec.oneapi.io/unified-runtime/latest/
  - Thoughts on building language runtimes on top of unified runtime
  - We're looking at SYCL and experimenting with OpenMP, but interested in language runtimes more broadly
  - Can file discussion points at https://github.com/oneapi-src/unified-runtime/issues

Questions