oneAPI Level Zero Technical Advisory Board

Welcome and Thanks

- A unique opportunity to steer the parallel programming ecosystem
- A problem worth solving
  - Multi-architecture, avoiding lock-in to 1 specific hardware architecture
  - Direct and library-based programming
  - Extending existing models
  - Performant
- Your leadership, input, and feedback is critical

Rules of the Road

- DO NOT share any confidential information or trade secrets with the group
- DO keep the discussion at a High Level
  - Focus on the specific Agenda topics
  - We are asking for feedback on features for the oneAPI specification (e.g. requirements for functionality and performance)
  - We are NOT asking for feedback on any implementation details
- Please submit any feedback in writing on Github in accordance with the Contribution Guidelines at spec.oneapi.com. This will allow Intel to further upstream your feedback to other standards bodies, including The Khronos Group SYCL* specification.

oneAPI

Agenda

- oneAPI Welcome & Introduction – Paul Petersen, Intel
- Level Zero:
  - Specification & How to Participate -- Zack Waters, Intel
- Discussion Topic:
  - Separation of Sysman from core Level Zero APIs -- Ben Ashbaugh, Intel
- Wrap up, Question & Answer – All

oneAPI

oneAPI

A unified programming model to simplify development across diverse architectures

Common developer experience across architectures

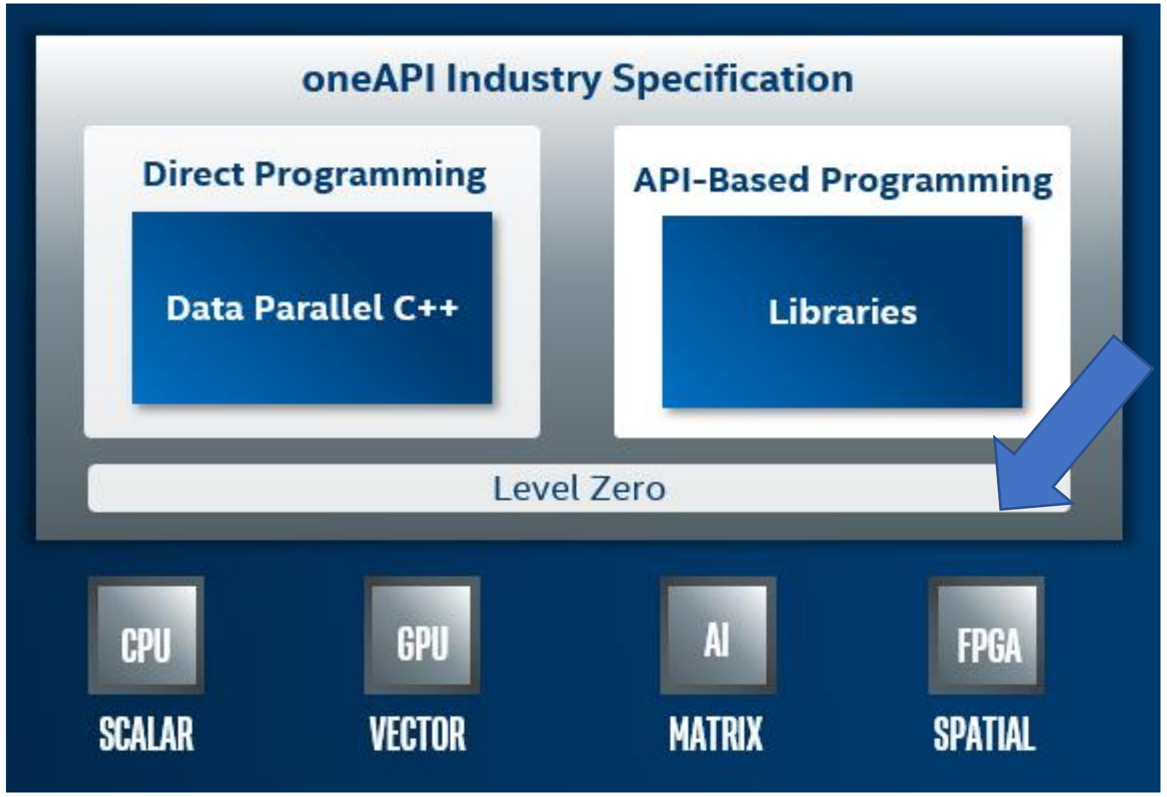
Unified and simplified language and libraries for expressing parallelism

Uncompromised native high-level language performance

Interoperates with existing languages and libraries

Support for CPU, GPU, AI and FPGA

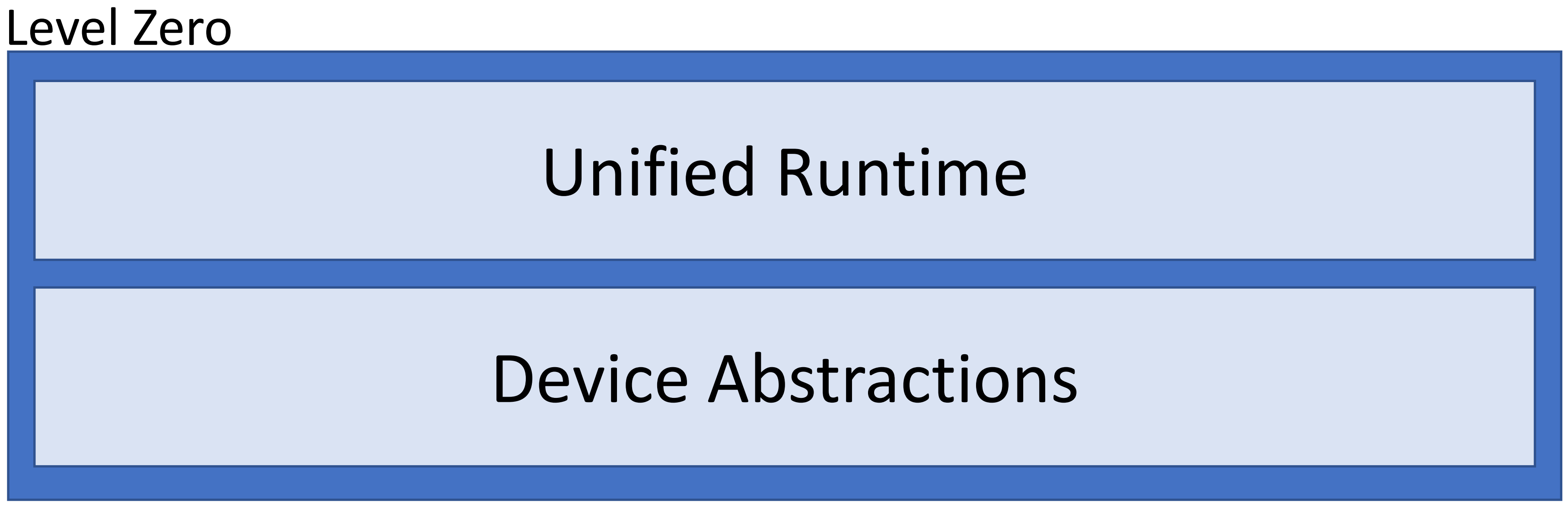
Based on industry standards and open specifications



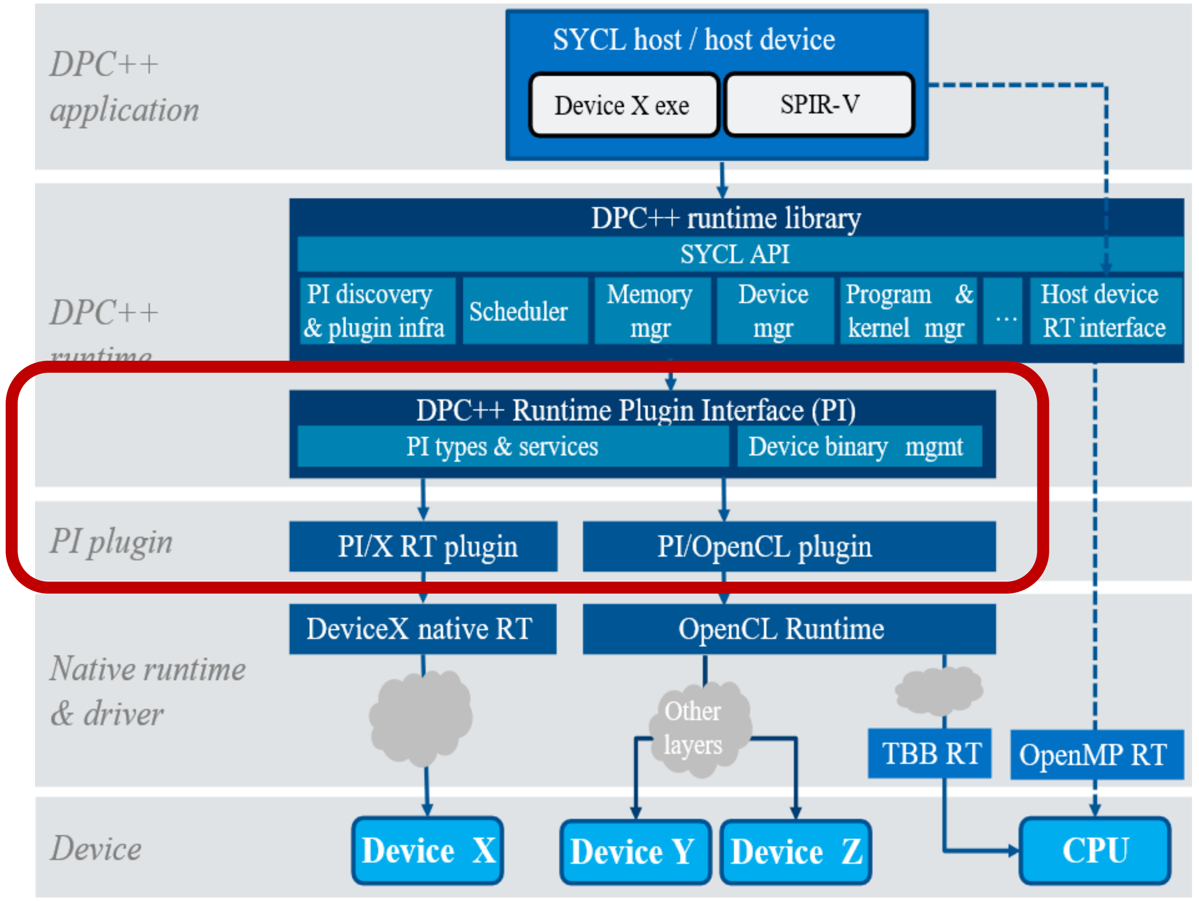
Does oneAPI need a Unified Runtime?

- Why is Level Zero as a defned not already sufficient?
  - Level Zero provides a low-level abstraction to a device.
  - Can we provide richer constructs for cross-platform and cross-API enabling?
- What are the Design principles?
  - Is it Deterministic – How would you support tuning heuristics?
  - Is it Stateless – How would you support callbacks to be registered?
- What features are missing to support your favorite language?
- Do we need special support for the CPU?
  - CPU device driver or resource management?
  - Leverage common components -- hwloc / memkind / numa?

Evolution of Level Zero



DPC++ Runtime Plugin Interface

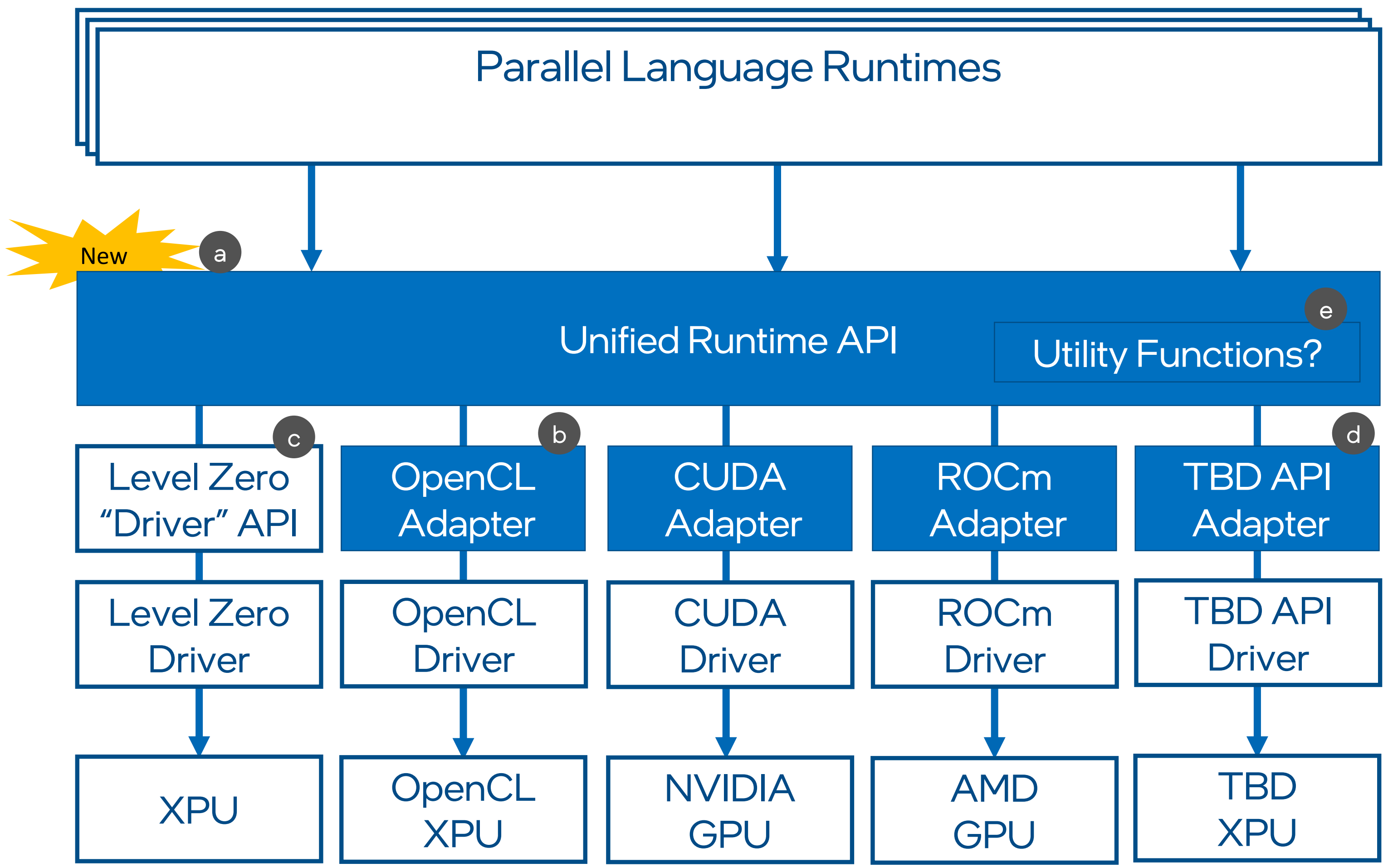


Problem Statement:

- Plugin Interface is an implementation detail
  - No formal specification
- Plugin Interface is only usable by the DPC++ Runtime
  - Other language runtimes must duplicate functionality
- Plugin Interface has deep OpenCL heritage

Diagram Source: https://github.com/intel/llvm/blob/sycl/sycl/doc/PluginInterface.md
(Note: slightly out-of-date!)

Proposal: Unified Runtime API



a. Refactor and formalize Plugin Interface into Unified Runtime APIs for use by multiple language runtimes

b. Refactor existing plugins into Unified Runtime API Adapters

c. Refactor, generalize, and modernize adapter interface

d. Enable new backends by implementing new Unified Runtime API Adapter

e. Eventually: Move common utility functionality to Unified Runtime API?

Request

“Our” initiative – help to frame it for your needs

Be brutally honest – good and bad

Invite your friends – and have fun

oneAPI

oneAPI™

Thank You!

http://oneapi.com