oneAPI State of the Union

Tony Mongkolsmai
Software Architect

oneAPI DevSummit
June 13, 2023

Welcome

oneAPI Ecosystem

oneAPI

Google · Microsoft · accenture · SIPEARL · RISC-V® · Julia computing · ArrayFire · Durham University

MathWorks® · FUJITSU · Adobe · PHILIPS · SIEMENS Healthineers · SAMSUNG MEDISON · mxnet · 瀚博半导体 POWERED BY VASTSTREAM

cerebras · stellar science · RIKEN · HUAWEI · EURECOM Sophia Antipolis · Sandia National Laboratories · UNIVERSITY of WASHINGTON · Argonne NATIONAL LABORATORY · Lawrence Livermore National Laboratory

KIT Karlsruher Institut für Technologie · THE UNIVERSITY OF TENNESSEE KNOXVILLE · Brookhaven National Laboratory · ANACONDA · MANCHESTER 1824 · UNIVERSITY of VIRGINIA · nag

CoEs

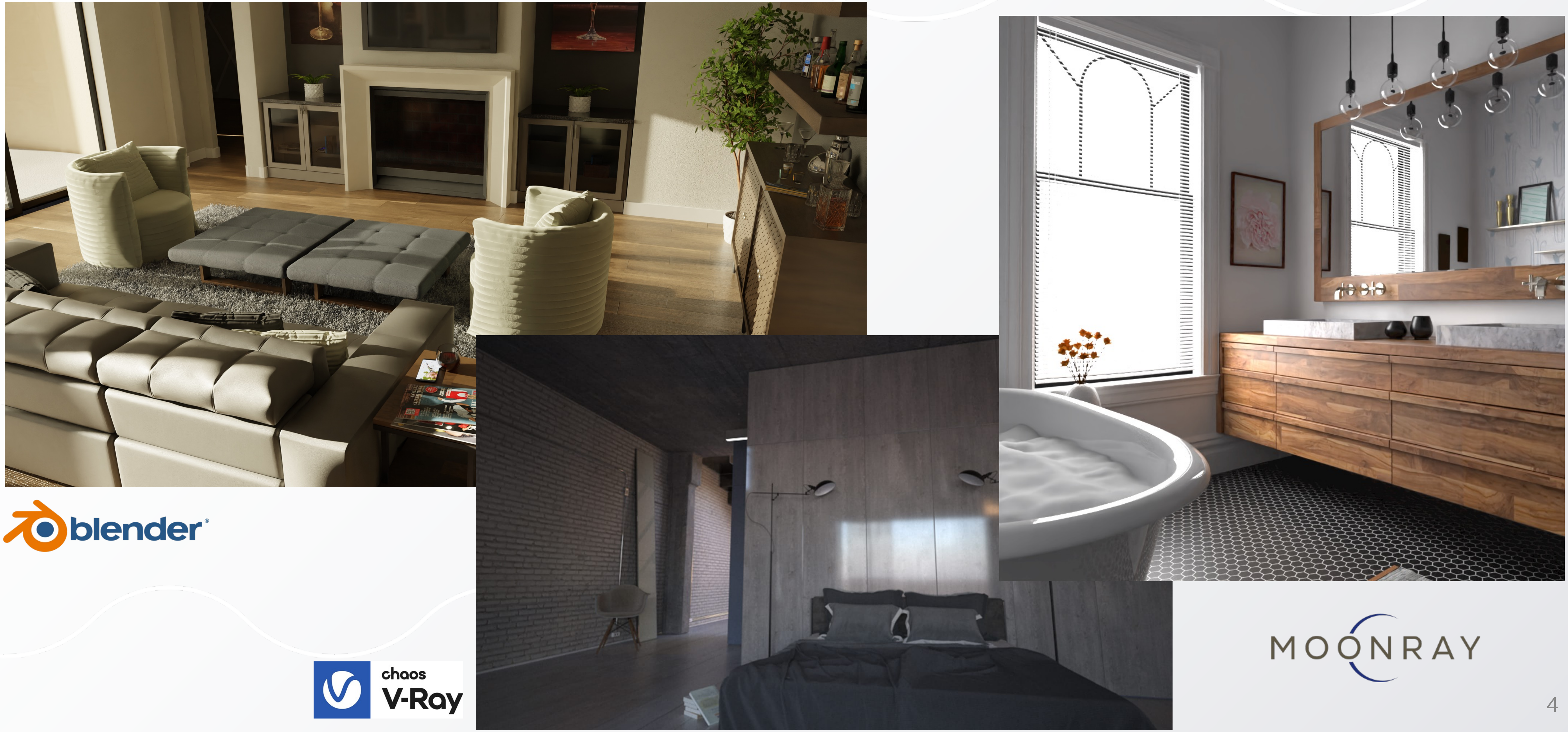
Argonne NATIONAL LABORATORY · University of BRISTOL · सीडैक CDAC · Computer Graphics Charles University · Heidelberg UNIVERSITY · 中國科學院 CHINESE ACADEMY OF SCIENCES · LOBACHEVSKY UNIVERSITY · Northern Illinois University NIU · OAK RIDGE National Laboratory

OLD DOMINION UNIVERSITY · Stockholm University · KTH · TECHNISCHE UNIVERSITÄT DARMSTADT · Berkeley UNIVERSITY OF CALIFORNIA · UC DAVIS · UNIVERSITY OF CAMBRIDGE · Durham University · ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

University of Stuttgart · THE UNIVERSITY OF TENNESSEE KNOXVILLE · TACC · SCI

oneAPI Going Mainstream

oneAPI

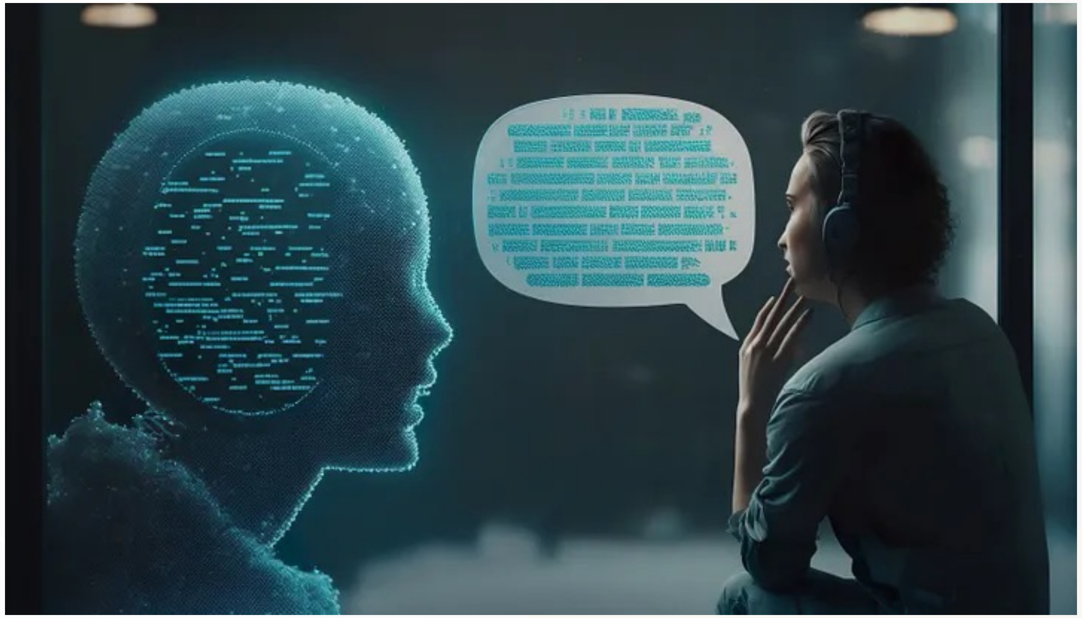


blender

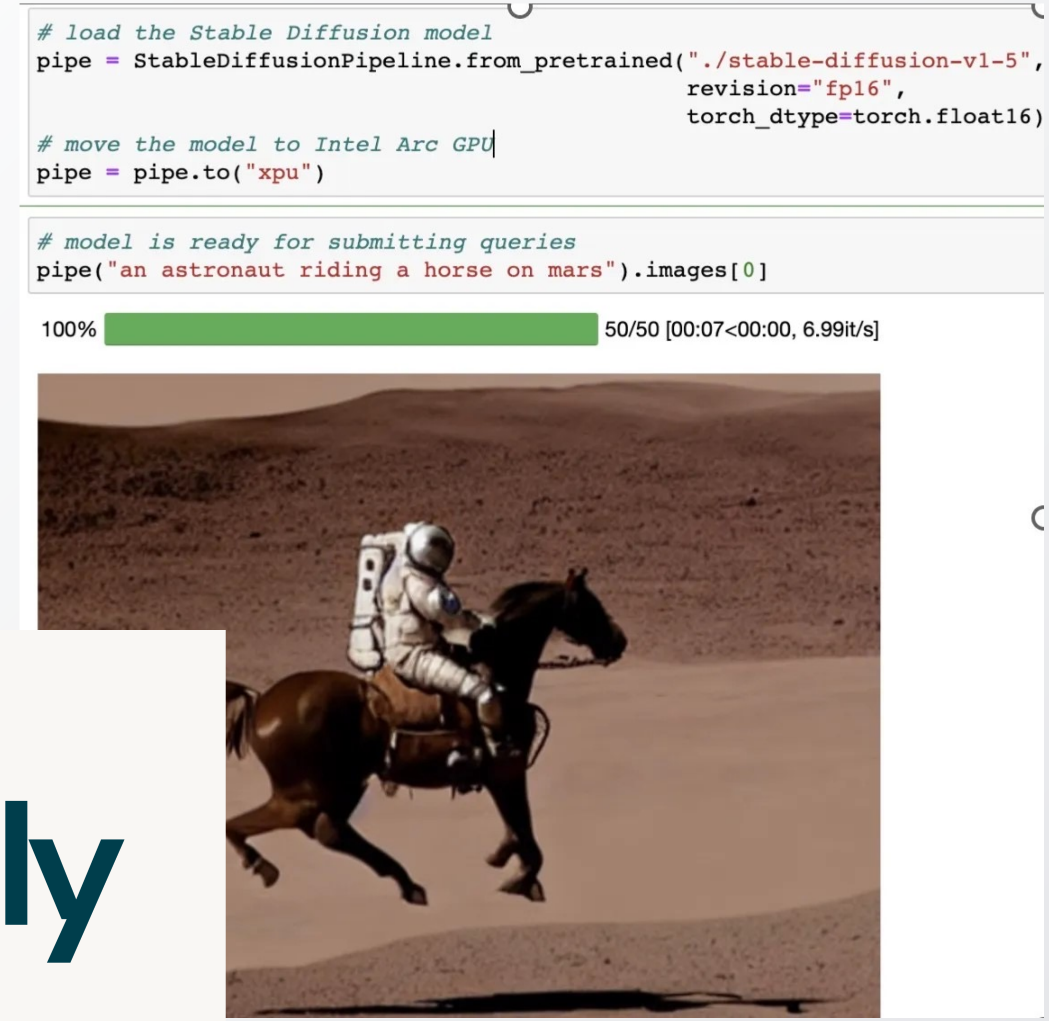
chaos V-Ray

MOONRAY

oneAPI for AI



Dolly



```
# load the Stable Diffusion model
pipe = StableDiffusionPipeline.from_pretrained("./stable-diffusion-v1-5",
                                                revision="fp16",
                                                torch_dtype=torch.float16)

# move the model to Intel Arc GPU
pipe = pipe.to("xpu")
```

```
# model is ready for submitting queries
pipe("an astronaut riding a horse on mars").images[0]
```

```
100% 50/50 [00:07<00:00, 6.99it/s]
```

oneAPI for Everyone

oneAPI Innovator Program

István Z Reguly
oneAPI Innovator

Student Ambassadors for oneAPI

Harvey Johnson
oneAPI Student Ambassador

Joshua Shiells
oneAPI Student Ambassador

Melbin Martin
oneAPI Student Ambassador

Liftoff for Startups

DIGITAL CORTEX

sitemana

NayaOne

Success Stories

(don’t take our word for it!)

Ginkgo Project

- High-performance linear algebra library for many core systems focused on sparse linear systems
- Modern C++ library that supports GPU kernels in CUDA, HIP, and oneAPI SYCL
- Extensible and Open Source

Dr. Hartwig Anzt
University of Tennessee

DPEcho

- Leibniz-Rechenzentrum (LRZ) Project
- **D**ata **P**arallel **E**ulerian **C**onservative **H**igh **O**rder (DPEcho) for General-Relativity-Magneto-Hydrodynamic simulation (GR-MHD) to model turbulence, wave propagation, stellar winds and processes around black holes

Rafael Lago
Intel

Beesearch

- Collaboration between Beewant/Weaviate
- Unlocking the potential of large amounts of Unstructured Data
- Use AI to accurately identify unstructured information
- Use Vector Databases and Similarity search to scale up

Ahmed Joudad
CEO Beewant

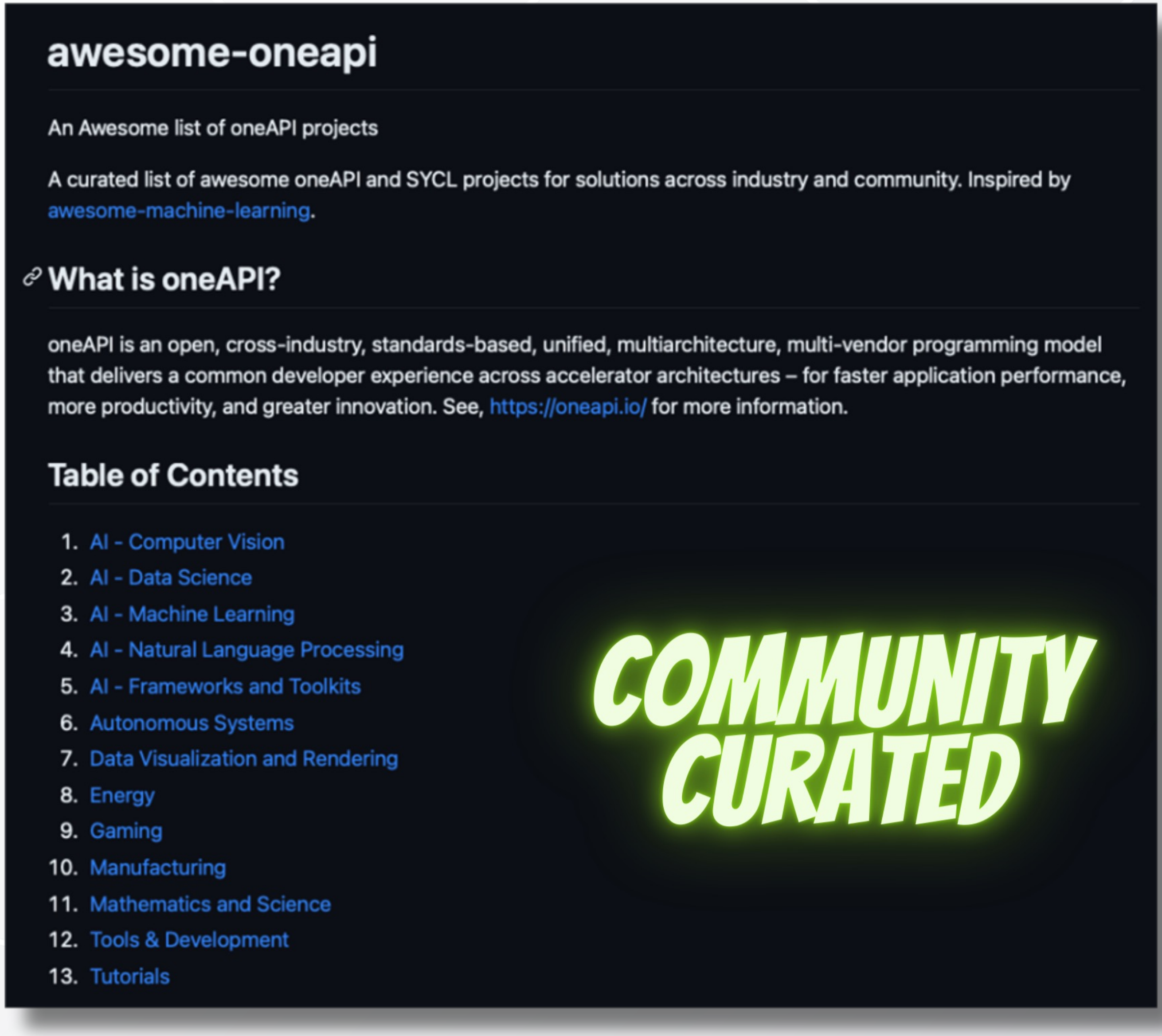
Sebastian Witalec
Weaviate

Get Inspired!

Awesome

oneAPI

Industry and community showcase of projects



awesome-oneapi

An Awesome list of oneAPI projects

A curated list of awesome oneAPI and SYCL projects for solutions across industry and community. Inspired by awesome-machine-learning.

What is oneAPI?

oneAPI is an open, cross-industry, standards-based, unified, multiarchitecture, multi-vendor programming model that delivers a common developer experience across accelerator architectures – for faster application performance, more productivity, and greater innovation. See, https://oneapi.io/ for more information.

Table of Contents

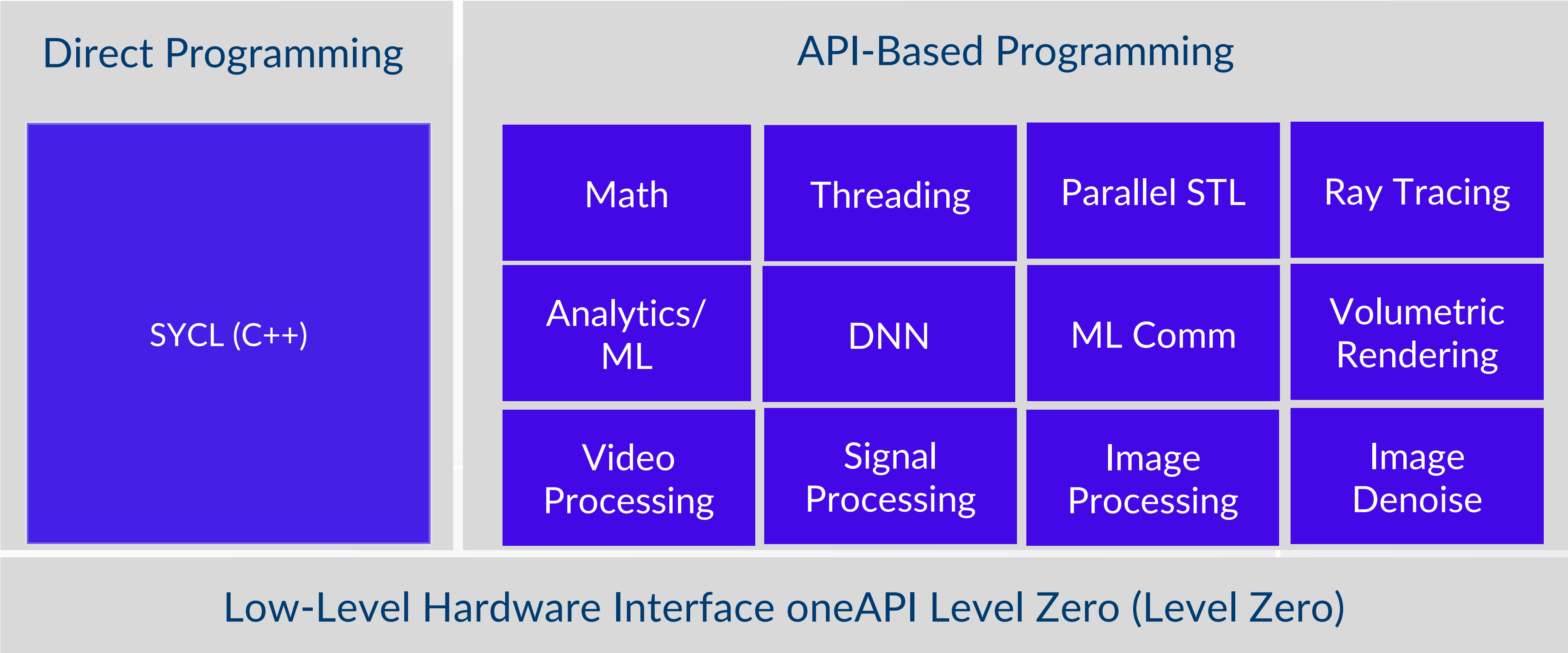
SYCLomatic

- Open source tool to help migrate CUDA code to oneAPI/SYCL
- 2832 contributors
- Just added support for 100+ CUDA APIs
- Working on SYCL 2020 support

https://github.com/oneapi-src/SYCLomatic

What’s New?

oneAPI Industry Specification



Direct Programming

SYCL (C++)

API-Based Programming

Math

Threading

Parallel STL

Ray Tracing

Analytics/ ML

DNN

ML Comm

Volumetric Rendering

Video Processing

Signal Processing

Image Processing

Image Denoise

Low-Level Hardware Interface oneAPI Level Zero (Level Zero)

oneAPI Math Kernel Library (oneMKL) Specification

Features

- APIs for Dense/Sparse Linear Algebra, Fast Fourier Transforms, Vector Math, Vector Random Number Generation, Summary Statistics
- Open source oneMKL interfaces project - support for multiple hardware backends

What's New

- AMD Support
- NVIDIA DFT Support
- Intel Data Center Max CPU/GPU Support

Domain	Backend Support
Basic Linear Algebra Systems (BLAS)	• Intel oneMKL • NVIDIA cuBLAS • AMD rocBLAS • SYCL-BLAS
Linear Algebra Package (LAPACK)	• Intel oneMKL • NVIDIA cuSOLVER • AMD rocSOLVER
Random Number Generation	• Intel oneMKL • NVIDIA cuRAND • AMD rocRAND
Discrete Fast Fourier Transforms	• Intel oneMKL • NVIDIA cuFFT

oneAPI Deep Neural Network (oneDNN) Specification

Features

- Open source library implementation
- Supports key data type formats, including 16- and 32-bit floating points, bfloat16, and 8-bit integers

What's New

- Experimental Graph API and Sparsity Support
- Math Mode API to manage down converting to low precision data types
- Updated quantization scheme supporting int8
- Performance improvements on Intel/ARM CPUs and AMD/NVIDIA/Intel GPUs

Category	Functions
Compute intensive operations	• (De-)Convolution • Inner Product • RNN (Vanilla, LSTM, GRU) • GEMM
Memory bandwidth limited operations	• Pooling • Batch Normalization • Local Response Normalization • Layer Normalization • Elementwise • Binary elementwise • Softmax • Sum • Concat • Shuffle
Data manipulation	• Reorder

oneAPI Data Parallel C++ Library (oneDPL) Specification

Features

- Optimized C++ standard algorithms – parallel algorithms (C++17) and utilities
- Custom Utilities and Algorithms
- Execution Policies semantically aligned to C++ standard
- Built on underlying SYCL

Category	Functions

Buffer Wrappers	begin()
	end()

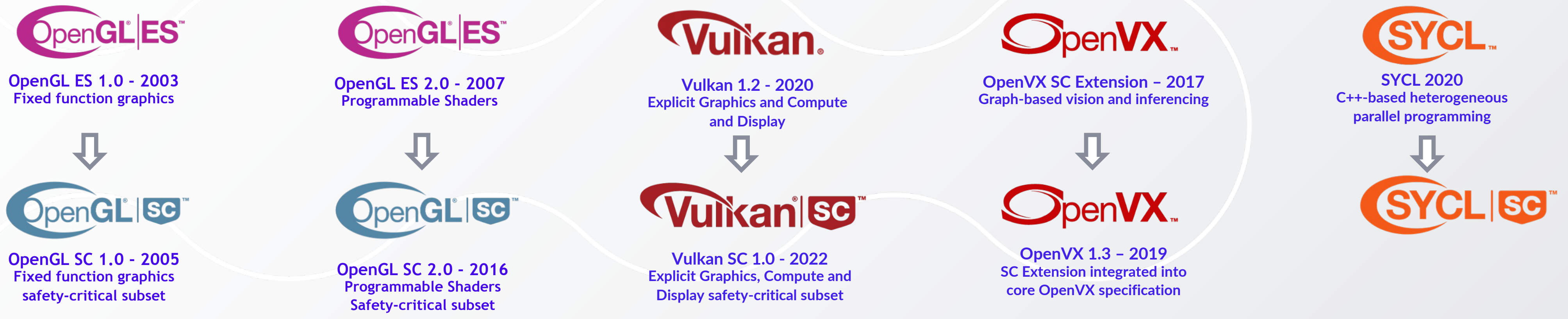
Iterators	counting_iterator discard_iterator permutation_iterator transform_iterator zip_iterator
Parallel Algorithms	exclusive_scan_by_segment inclusive_scan_by_segment reduce_by_segment binary_search lower_bound upper_bound

SYCL Compiler Advancements

- DPC++/C++ Compiler
  - Codeplay oneAPI for NVIDIA® GPUs and Codeplay oneAPI for AMD GPUs
  - Implementations for Proposed Extensions to SYCL
    - Joint Matrix
    - SYCL Graph
    - Bindless Texture
  - Updated to use SYCL 2020 Specification by default
- hipSYCL Compiler
  - Single pass compilation
  - Convenience of generating universal binaries -> --hipsycl-targets=generic
  - Intel GPU support no longer considered experimental
  - Unified device code representation for all backends

SYCL for Safety Critical Applications

- March 2023 - SYCL SC Working Group announced to develop C++-based heterogeneous parallel compute programming framework for safety-critical systems
- Will align with safety certification standards in avionics, automotive, industrial and medical fields



OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL SC 1.0 - 2005
Fixed function graphics
safety-critical subset

OpenGL ES 2.0 - 2007
Programmable Shaders

OpenGL SC 2.0 - 2016
Programmable Shaders
Safety-critical subset

Vulkan 1.2 - 2020
Explicit Graphics and Compute and Display

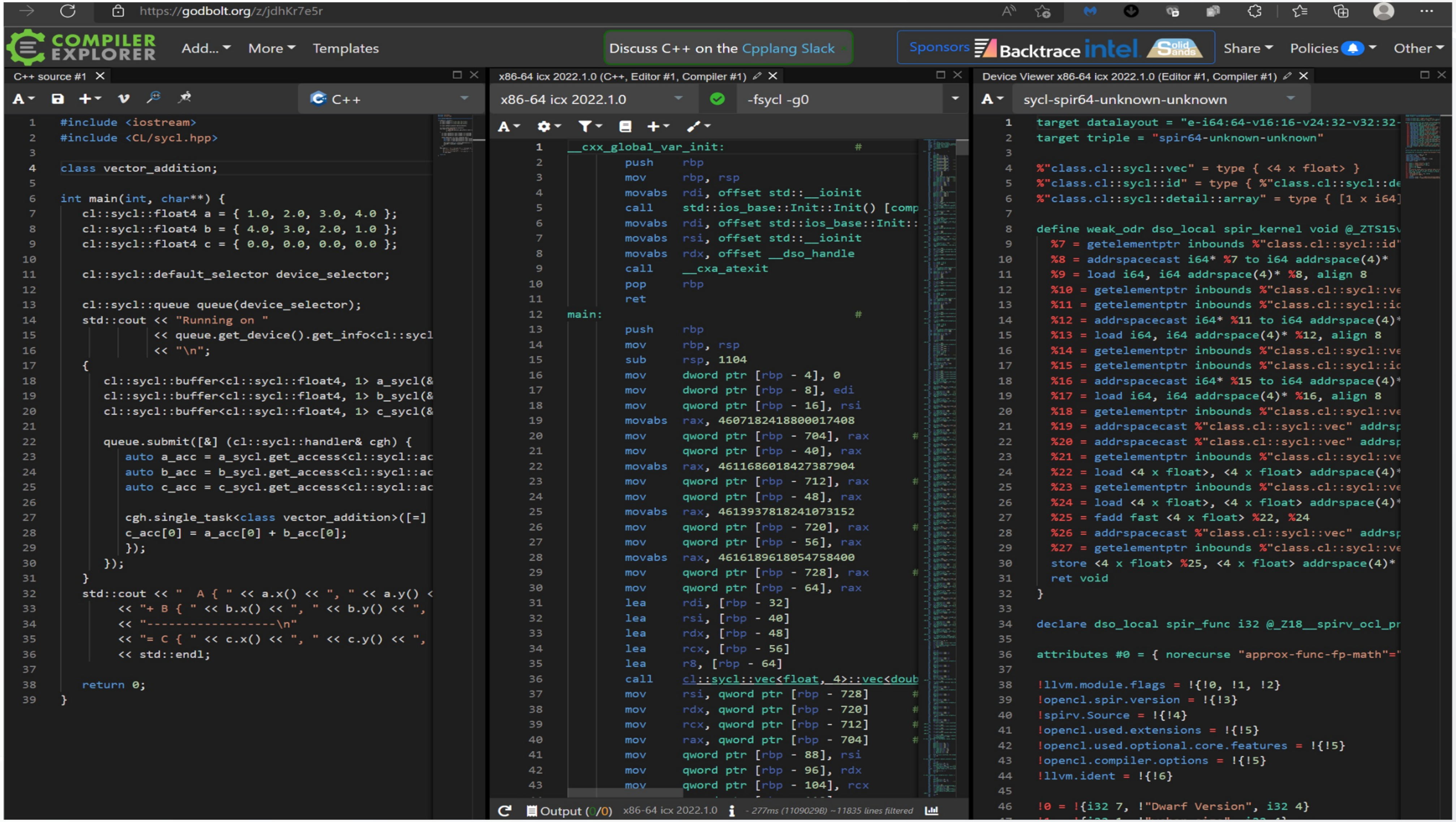
Vulkan SC 1.0 - 2022
Explicit Graphics, Compute and Display safety-critical subset

OpenVX SC Extension – 2017
Graph-based vision and inferencing

OpenVX 1.3 – 2019
SC Extension integrated into core OpenVX specification

SYCL 2020
C++-based heterogeneous parallel programming

SYCL in Compiler Explorer



C++ source #1

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

class vector_addition;

int main(int, char**) {
   cl::sycl::float4 a = { 1.0, 2.0, 3.0, 4.0 };
   cl::sycl::float4 b = { 4.0, 3.0, 2.0, 1.0 };
   cl::sycl::float4 c = { 0.0, 0.0, 0.0, 0.0 };

   cl::sycl::default_selector device_selector;

   cl::sycl::queue queue(device_selector);
   std::cout << "Running on "
             << queue.get_device().get_info<cl::sycl
             << "\n";
   {
      cl::sycl::buffer<cl::sycl::float4, 1> a_sycl(&
      cl::sycl::buffer<cl::sycl::float4, 1> b_sycl(&
      cl::sycl::buffer<cl::sycl::float4, 1> c_sycl(&

      queue.submit([&] (cl::sycl::handler& cgh) {
         auto a_acc = a_sycl.get_access<cl::sycl::ac
         auto b_acc = b_sycl.get_access<cl::sycl::ac
         auto c_acc = c_sycl.get_access<cl::sycl::ac

         cgh.single_task<class vector_addition>([=]
         c_acc[0] = a_acc[0] + b_acc[0];
         });
      });
   }
   std::cout << "  A { " << a.x() << ", " << a.y() <
        << "+ B { " << b.x() << ", " << b.y() << ",
        << "------------------\n"
        << "= C { " << c.x() << ", " << c.y() << ",
        << std::endl;

   return 0;
}
```

x86-64 icx 2022.1.0 (C++, Editor #1, Compiler #1)

x86-64 icx 2022.1.0 | -fsycl -g0

```
__cxx_global_var_init:                          #
        push    rbp
        mov     rbp, rsp
        movabs  rdi, offset std::__ioinit
        call    std::ios_base::Init::Init() [comp
        movabs  rdi, offset std::ios_base::Init::
        movabs  rsi, offset std::__ioinit
        movabs  rdx, offset __dso_handle
        call    __cxa_atexit
        pop     rbp
        ret
main:                                           #
        push    rbp
        mov     rbp, rsp
        sub     rsp, 1104
        mov     dword ptr [rbp - 4], 0
        mov     dword ptr [rbp - 8], edi
        mov     qword ptr [rbp - 16], rsi
        movabs  rax, 4607182418800017408
        mov     qword ptr [rbp - 704], rax       #
        mov     qword ptr [rbp - 40], rax
        movabs  rax, 4611686018427387904
        mov     qword ptr [rbp - 712], rax       #
        mov     qword ptr [rbp - 48], rax
        movabs  rax, 4613937818241073152
        mov     qword ptr [rbp - 720], rax       #
        mov     qword ptr [rbp - 56], rax
        movabs  rax, 4616189618054758400
        mov     qword ptr [rbp - 728], rax       #
        mov     qword ptr [rbp - 64], rax
        lea     rdi, [rbp - 32]
        lea     rsi, [rbp - 40]
        lea     rdx, [rbp - 48]
        lea     rcx, [rbp - 56]
        lea     r8, [rbp - 64]
        call    cl::sycl::vec<float, 4>::vec<doub
        mov     rsi, qword ptr [rbp - 728]       #
        mov     rdx, qword ptr [rbp - 720]       #
        mov     rcx, qword ptr [rbp - 712]       #
        mov     rax, qword ptr [rbp - 704]       #
        mov     qword ptr [rbp - 88], rsi
        mov     qword ptr [rbp - 96], rdx
        mov     qword ptr [rbp - 104], rcx
```

Output (0/0) x86-64 icx 2022.1.0 - 277ms (1109029B) ~11835 lines filtered

Device Viewer x86-64 icx 2022.1.0 (Editor #1, Compiler #1)

sycl-spir64-unknown-unknown

```
target datalayout = "e-i64:64-v16:16-v24:32-v32:32-
target triple = "spir64-unknown-unknown"

%"class.cl::sycl::vec" = type { <4 x float> }
%"class.cl::sycl::id" = type { %"class.cl::sycl::de
%"class.cl::sycl::detail::array" = type { [1 x i64]

define weak_odr dso_local spir_kernel void @_ZTS15v
  %7 = getelementptr inbounds %"class.cl::sycl::id"
  %8 = addrspacecast i64* %7 to i64 addrspace(4)*
  %9 = load i64, i64 addrspace(4)* %8, align 8
  %10 = getelementptr inbounds %"class.cl::sycl::ve
  %11 = getelementptr inbounds %"class.cl::sycl::ic
  %12 = addrspacecast i64* %11 to i64 addrspace(4)*
  %13 = load i64, i64 addrspace(4)* %12, align 8
  %14 = getelementptr inbounds %"class.cl::sycl::ve
  %15 = getelementptr inbounds %"class.cl::sycl::ic
  %16 = addrspacecast i64* %15 to i64 addrspace(4)*
  %17 = load i64, i64 addrspace(4)* %16, align 8
  %18 = getelementptr inbounds %"class.cl::sycl::ve
  %19 = addrspacecast %"class.cl::sycl::vec" addrsp
  %20 = addrspacecast %"class.cl::sycl::vec" addrsp
  %21 = getelementptr inbounds %"class.cl::sycl::ve
  %22 = load <4 x float>, <4 x float> addrspace(4)*
  %23 = getelementptr inbounds %"class.cl::sycl::ve
  %24 = load <4 x float>, <4 x float> addrspace(4)*
  %25 = fadd fast <4 x float> %22, %24
  %26 = addrspacecast %"class.cl::sycl::vec" addrsp
  %27 = getelementptr inbounds %"class.cl::sycl::ve
  store <4 x float> %25, <4 x float> addrspace(4)*
  ret void
}

declare dso_local spir_func i32 @_Z18__spirv_ocl_pr

attributes #0 = { norecurse "approx-func-fp-math"="

!llvm.module.flags = !{!0, !1, !2}
!opencl.spir.version = !{!3}
!spirv.Source = !{!4}
!opencl.used.extensions = !{!5}
!opencl.used.optional.core.features = !{!5}
!opencl.compiler.options = !{!5}
!llvm.ident = !{!6}

!0 = !{i32 7, !"Dwarf Version", i32 4}
```

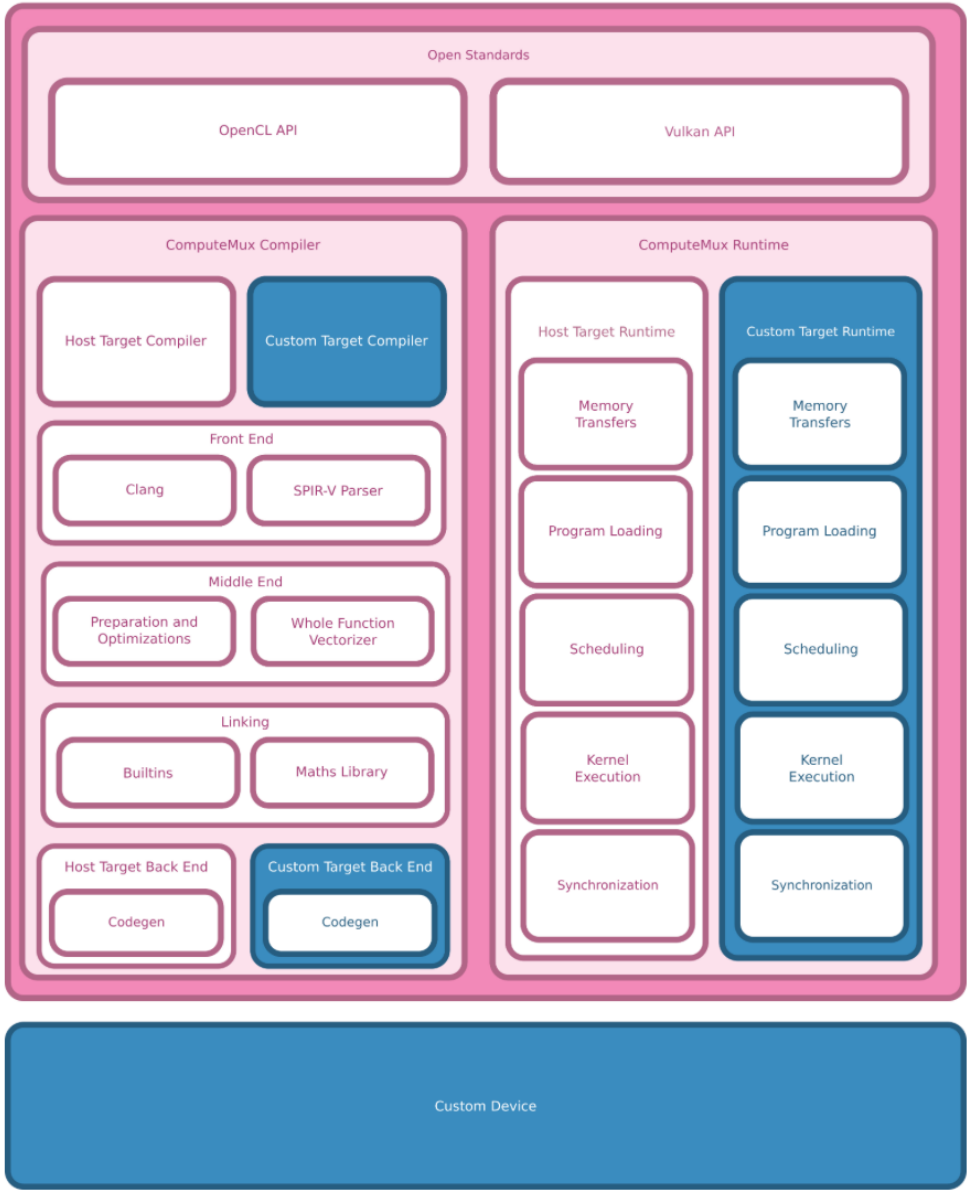
https://godbolt.org/z/jdhKr7e5r

oneAPI Construction Kit

- Simplifies process for hardware vendors to integrate with the oneAPI software stack
- For CPU + accelerator systems
- Ubuntu 20.04, Ubuntu 22.04, Windows
- RISC-V reference implementation

IWOCL 2020 Presentation

oneAPI Construction Kit Homepage



Community Matters

oneAPI Community Forum

For Software Developers

- Standards and industry defined libraries
- Future proof your software
- Enable an existing ecosystem of software and educational resources
- Develop with open standards for accelerator computing
- Enable software to run on multiple architectures

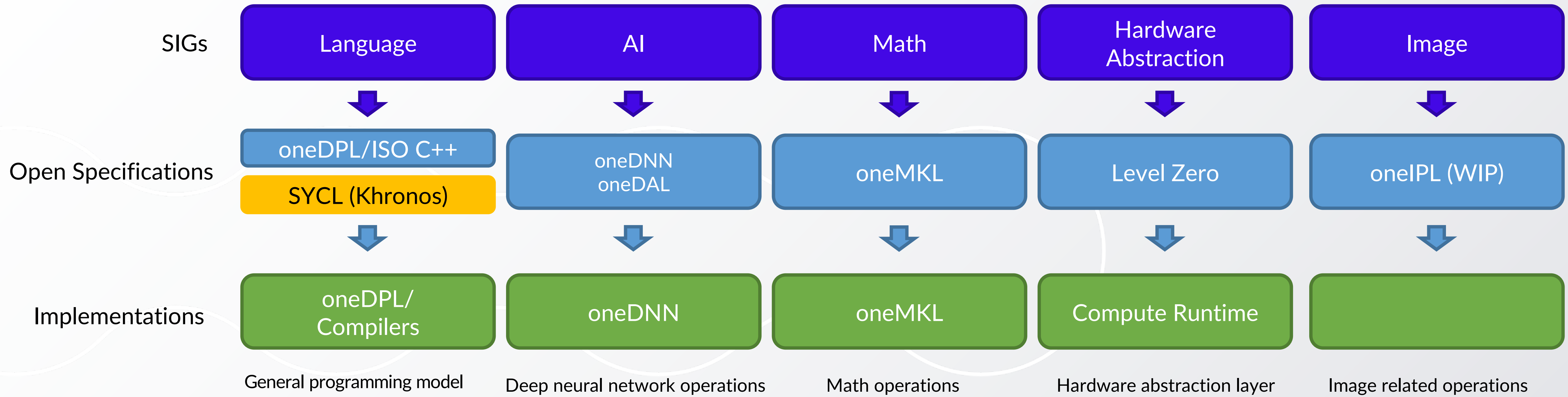
For Hardware Developers

- Fast track into market with an existing ecosystem
- Share development cost with open-source implementations
- Leverage an existing tested and optimized toolchain
- Enable an existing ecosystem of software and educational resources

Free and based on open standards

Special Interest Groups

Special Interest Groups influence the specifications and implementations



SIGs	Language	AI	Math	Hardware Abstraction	Image
Open Specifications	oneDPL/ISO C++ SYCL (Khronos)	oneDNN oneDAL	oneMKL	Level Zero	oneIPL (WIP)
Implementations	oneDPL/ Compilers	oneDNN	oneMKL	Compute Runtime	
	General programming model	Deep neural network operations	Math operations	Hardware abstraction layer	Image related operations

Get Involved!

Join

oneAPI Forum

Learn

Data Parallel C++ Book

Try

Intel® Developer Cloud (Beta)

Share

Awesome oneAPI GitHub

oneAPI

oneAPI Community Programs

Student Ambassadors

Innovator Program

Intel® Liftoff for Start-ups

Educator Program

Join our community!

Thank You