

Target NVIDIA and AMD with oneAPI and SYCL

Rafal Bielski – Codeplay Software

14 September 2023



Enabling AI & HPC To Be Open, Safe & Accessible To All



Ŵ

Established 2002 in **Edinburgh, Scotland**.

Grown successfully to around 100 employees.

In 2022, we became a **wholly owned subsidiary** of Intel.



Committed to expanding the **open ecosystem** for heterogeneous computing.

Through our involvement in oneAPI and SYCL governance, we help to **maintain and develop** open standards. Developing at the forefront of **cutting-edge research**.

Currently involved in two research projects - **SYCLOPS** and **AERO**, both funded by the Horizon Europe Project.



Bringing oneAPI to NVIDIA and AMD GPUs



Codeplay provides oneAPI plugins enabling SYCL to run on NVIDIA and AMD GPUs







Write code using SYCL, and then run freely across Intel, NVIDIA and AMD GPUs





On NVIDIA GPU – SYCL Provides Comparable Performance to CUDA

Testing Date: Performance results are based on testing by Intel as of April 15, 2023 and may not reflect all publicly available updates.

Configuration Details and Workload Setup: Intel® Xeon® Platinum 8360Y CPU @ 2.46Hz, 2 socket, Hyper Thread On, Turbo On, 256GB Hynix DDR4-3200, ucode 0xd000363, CPU: Nvidia A100 PCIe 80GB GPU memory. Software: SYCL open source/CLANG to ND, CUDA SDK 120 with NVIDIA-NVCC 120.76, cuMath 120, cuDNN 120, Ubuntu 22.04.1, SYCL open source/CLANG compiler switches: -fscycl-targets=nvptx64-nvidia-cuda, NVIDIA-NVCC compiler switches: -03 –gencode arch=compute_80, code=sm_80. Represented workloads with Intel optimizations.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

SYCL performance on NVIDIA/AMD GPUs matches native CUDA/HIP for diverse workloads

See <u>our blog post</u> for more details on these benchmark results



No compromises on

performance.



On AMD GPU – SYCL Provides Comparable Performance to HIP

Relative Performance: AMD SYCL vs. AMD HIP on AMD Instinct MI250 Accelerator (HIP = 1.00) (Higher is Better)



Testing Date: Performance results are based on testing by Intel as of April 15, 2023 and may not reflect all publicly available updates.

Configuration Details and Workload Setup: AMD EPYC 7313 CPU @ 3.0GHz, 2 socket, AMD Simultaneous Multi-Threading Off, AMD Precision Boost Enabled, 512GB DDR4, ucode 0xa001144, GPU: AMD Instinct MI250 OAM, 128GB GPU memory. Software: SYCL open source/CLANG 17.0.0, AMD RoCm 5.3.0 with roc-5.3.0 22362, hipSolver 5.3.0, rocBLAS 5.3.0, Ubuntu 20.04.4. SYCL open source/CLANG compiler switches: -03 -fsycl -fsycHargets=amdgcn-amdhsa-XsycHarget-backend --offload-arch=gfx90a, AMD-ROCm compiler switches: -03. Represented workloads with Intel optimizations.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at <u>www.Intel.com/PerformanceIndex</u>. Your costs and results may vary.

SYCL performance on NVIDIA/AMD GPUs matches native CUDA/HIP for diverse workloads

See <u>our blog post</u> for more details on these benchmark results

() codeplay[®]

Q

Open, cross-industry collaboration on standards.

	[Dexter] Add timeout options	2 mont
	Revert "Revert "[Ci][Deps] Uplift CPU/FPGAEMU RT version to 2023	las
	[flang] fix fir.array_coor of fir.box with component references	3 wee
	[LIBC] Fix comments / name ofsched_cpu_count tests	3 wee
g (#9271)	였 Discussions ⓒ Actions 冊 Projects 1 때 Wiki ♡ Se	curity
hes 🔿 1,158	tags Go to	file
(CL][NFC] Remo	ove extra code. Fix spelling (#9271) • b9b7f9b 27 minutes ago 🕥 4	7 1,596 coi
	[CI] Skip Windows checks if linter fails (#9050)	3 wee
	[BOLT][NFC] Fix UB due to unaligned load in DebugStrOffsetsWriter	3 wee
	[SYCL] Build AMDGPU target if AMD libclc targets are enabled (#915	2 wee
	[clangd] Fix test failure in initialize-params.test	3 wee
	[SYCL][NFC] Remove extra code. Fix spelling (#9271)	27 minut
	[cmake] Add missing CMakePushCheckState include to FindLibEdit.c	6 mont
	Merge commit 'b9ba05360e585729458363a53f6cba71d1a8ebb6' i	2 wee
	[Dexter] Add timeout options	2 mont
	Revert "Revert "[CI][Deps] Uplift CPU/FPGAEMU RT version to 2023	las
	[flang] fix fir.array_coor of fir.box with component references	3 wee
	[LIBC] Fix comments / name ofsched_cpu_count tests	
g (#9271) ₁ 72	□ Discussions	

The code is entirely open source

Available as a free plugin download at <u>developer.codeplay.com</u> for selected platforms

Available to build from source for other platforms



Getting started – live demo

Get started



Install the CUDA or HIP development environment and drivers Get the toolkit from Intel intel.com/developer

Get the plugins from developer.codeplay.com



Live demo: installing the plugins

- Using the following setup:
 - A machine with an NVIDIA GPU and an Intel iGPU
 - Another machine with an AMD GPU
 - Both with Ubuntu 22.04, <u>Intel oneAPI Base Toolkit</u> 2023.2.1, drivers and toolkits (CUDA/ROCm) for the GPUs installed
- We'll show:
 - Checking the available GPUs with 1shw -c video
 - Checking GPU and driver details with nvidia-smi / rocm-smi
 - Using sycl-1s to find supported SYCL backends
 - Installing <u>NVIDIA</u> and <u>AMD</u> plugins for oneAPI



Compiling

Basic compilation: icpx -fsycl -fsycl-targets=nvptx64-nvidia-cuda sycl-app.cpp -o sycl-app Use the SYCL compiler The source file The binary

Run the app with optional environment variables switching SYCL runtime options: ONEAPI_DEVICE_SELECTOR=cuda:gpu SYCL_PI_TRACE=1 ./sycl-app



Documentation of the runtime environment variables:

https://intel.github.io/llvm-docs/EnvironmentVariables.html



Compiling

- Some features are not yet fully supported by the *icpx* compiler driver
- See <u>developer.codeplay.com/products/oneapi/nvidia/2023.2.0/guides/troubleshooting</u>
- If you wish to use them, it is possible to use the clang++ driver
- Set up the environment with: source /opt/intel/oneapi/setvars.sh --include-intel-llvm
- Use clang++ in the same way as icpx the aforementioned CLI options are identical between the two
- icpx provides Intel's extra proprietary optimisations



Live demo: compiling for a single target

- Example code:
 - github.com/rafbiels/HeCBench/tree/nbody-sycl-demo/nbody-sycl
 - See upstream repo for more details: github.com/zjin-lcf/HeCBench
- Compile for NVIDIA GPU:

icpx -fsycl -fsycl-targets=nvptx64-nvidia-cuda \
-Xsycl-target-backend --offload-arch=sm_86 \
-03 -o main main.cpp GSimulation.cpp

• Compile for AMD GPU:

```
icpx -fsycl -fsycl-targets=amdgcn-amd-amdhsa \
-Xsycl-target-backend --offload-arch=gfx1010 \
-03 -o main main.cpp GSimulation.cpp
```

• Run:

```
SYCL_PI_TRACE=1 ./main
```



Enable SYCL compilation

2

3

4

Set the SYCL target to either nvptx64-nvidia-cuda or amdgcn-amd-amdhsa

Optionally set the GPU architecture to make the most out of your hardware. See these pages for the right values:

- <u>https://developer.nvidia.com/cuda-gpus</u>
- https://llvm.org/docs/AMDGPUUsage.html

Add any other flags, e.g. the optimisation level

() codeplay[®]

Multi-Target Compilation

- Use a comma-separated list of targets for -fsycl-targets
- Specify a target for the backend options with -Xsycl-target-backend=<target>



Multi-Target Compilation

Executing the same binary on different target hardware



Select Intel GPU with Level Zero or OpenCL backend:

ONEAPI_DEVICE_SELECTOR=level_zero:gpu SYCL_PI_TRACE=1 ./sycl-app
ONEAPI_DEVICE_SELECTOR=opencl:gpu SYCL_PI_TRACE=1 ./sycl-app

() codeplay[®]

Live demo: multi-target compilation

• Compile for three targets:

icpx -fsycl -fsycl-targets=amdgcn-amd-amdhsa,nvptx64-nvidia-cuda,spir64 \
 -Xsycl-target-backend=amdgcn-amd-amdhsa --offload-arch=gfx1010 \
 -Xsycl-target-backend=nvptx64-nvidia-cuda --offload-arch=sm_86 \
 -03 -o main main.cpp GSimulation.cpp

- objdump -h shows the binary contains an offload section for each target
- The default selector chooses an available backend
- Choose a specific backend with ONEAPI_DEVICE_SELECTOR or with custom selectors in code



Debugging

- Use standard tooling for debugging
- Vendor-specific gdb extensions facilitate debugging device code: cuda-gdb for NVIDIA GPU and gdb-oneapi for Intel GPU
- rocgdb for AMD GPU currently not supported, discussion ongoing on upstreaming AMD support for device debug information to llvm
- GPU debuggers can be integrated with your favourite IDE just like the regular gdb or 11db



Debugging

- cuda-gdb might have <u>some issues</u> with certain calls generated by LLVM
- If you run into errors while debugging SYCL code, try starting cuda-gdb with CUDBG_USE_LEGACY_DEBUGGER=1



Live demo: debugging

- Compile for the NVIDIA target with -00 and -g
- Launch the program in cuda-gdb
- List code, set a breakpoint, run the program
- Investigate the device state at breakpoint
- Print kernel stack variables and step through instructions



Profiling

Use NVIDIA NSight Systems and NSight Compute to profile SYCL code

Elle Connection Debug Profile Tools Window Help Connect Connect Torminate Profile Kernel Connect Torminate Connect	Copy as Im Copy as Im the hardware' he ha	nage • • • • • * s in • • • • • • • • • • • • • • • • • •
Connect © Disconnect × Terminate © Profile Kernel Q d-syck-dbg ncu-rep × Page: Details • Result: 0 - 136 - compute • V Add Baseline Apply Rules © Occupancy Calculator Result: 0 - 136 - compute • V Add Baseline Apply Rules © Occupancy Calculator • CC Process • CC Process • Company Occupancy Is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencie overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads. Theoretical Occupancy [%] S Block Limit Registers [block] Achieved Active Warps Per SM [warp] 22 Block Limit SM glock] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM glock] This kernel's theoretical occupancy (50 0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and n achieved occupancy Limiters This kernel's theoretical occupancy (50 0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and n achieved occupancy Limiters This kernel's theoretical occupancy (50 0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and n achieved occupancy limiters This kernel's theoretical occupancy of Varying Register Count Per Thread 94	Copy as Im Copy as Im he hardware' he har	nage • © • 2's in 4 32 8 32
Bed-syd-dbg nou-rep × Page: Details Result O- 136 - compute Page: Details Result Time Cycles Regs GPU SM Frequency CC Process Current 136 - compute (24, 1, 1)x(256, 1, 1) 1.65 second 1.259,406,848 52 O-NVIDIA A100-PCIE-40GB 764.98 cycle/usecond 8.0 [3013141] main Occupancy Occupancy Occupancy CC Process oreral Inder of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of th ability to process warps that is actively in use Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to indicates highly imbalanced workloads. Theoretical Active Warps per SM [warp] S0 Block Limit Registers [block] Achieved Active Warps Per SM [warp] Tearet Active Warps Per SM [warp] Ta2:4 Block Limit SM glock] Achieved Active Warps Per SM [warp] Ta3: Block Limit SM glock] Achieved Active Warps Per SM [warp] Ta3: Block Limit SM glock] Achieved Active warps versultin a block as well as across blocks of the same krenel. See the G_UDA Brest Principes during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same krenel. See the G_UDA Brest Principes during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same krenel. See the G_UDA Brest Principes during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same krenel. See the G_UDA Brest Principes during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same krenel. See the G_UDA Brest Principes during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same krenel. Sev	Copy as Im © © the hardware' s, resulting i neasured an occur pancy.	nage • © • 2's in 4 32 8 32
Page: Details Result 0 - 136 - compute Image Apply Rules Occupancy Calculator Result Time Cycles Regs GPU SM Frequency CC Process Current 136 - compute (24, 1, 1)x(256, 1, 1) 1.65 second 1.259,406,848 52 0 - NVIDIA A100-PCIE-40GB 764 98 cycle/usecond 8.0 [3013141] main Occupancy Cocupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencie overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads. Theoretical Active Warps per SM [warp] 32 Block Limit Registers [block]	Copy as Im © © he hardware' s, resulting i neasured an occur pancy.	nage • © • 2's in 4 32 8 32
Result Time Cycles Reg d GPU SM Frequency CC Process Current 136 - compute (24, 1, 1)x(256, 1, 1) 1.65 second 1.259,406,848 52 0 - NVIDIA A100-PCIE-40GB 764.98 cycle/usecond 8.0 [3013141] main • Occupancy Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencie overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads. Theoretical Occupancy [%] 50 Block Limit Registers [block] Theoretical Active Warps per SM [warp] 32 Block Limit Warps [block] 4 Achieved Occupancy [%] 50 Achieved Active Warps Per SM [warp] 7.83 Block Limit Warps [block] 50 50.00 kLimit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] 50.00 kLimit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] 50.00 kLimit SM [block] A Occupancy Limiters The kernel texecution accupancy [%] 50.00 kLimit SM [block] 5	Arrow of the second secon	€ C P's in 4 32 8 32 32
Current 136 - compute (24, 1, 1)x(256, 1, 1) 1.65 second 1.259,406,848 52 0 - NVIDIA A100-PCIE-40GB 764.98 cycle/usecond 8.0 [3013141] main Cocupancy Cocupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to process warps that is actively in use. Higher occupancy dees not always result in higher performance, however, low occupancy always reduces the ability to proceed active warps per SM [warp] 50 Block Limit Registers [block] Achieved Occupancy [%] 7.83 Block Limit Mans [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] 7.84 Block Astive doccupancy (12.2%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can be tween warps within a block as well as across blocks of the same kernel. See the [\$ CLOA Breat Practices Glube for more details on optimizing occup Impact of Varying Register COAN Perf	he hardware es, resulting i measured an occur pancy.	Q ▲ a's in 4 32 8 32
Cocupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of tt ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencie overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads. Theoretical Active Warps per SM [warp]	he hardware es, resulting i measured an occur pancy.	© ▲ e's in 4 32 8 32
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the adult to process warps that is actively in use. Higher occupancy does not adways result in higher performance, however, fow occupancy adways reduces the ability to process warps that is actively in use. Higher occupancy does not adways result in higher performance, however, fow occupancy adways reduces the ability to high attended overall performance. However, fow occupancy adways reduces the ability to high attended occupancy during execution typically indicates highly imbalanced workloads. Theoretical Occupancy [%] 50 Block Limit Registers [block] Achieved Occupancy [%] 12.24 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] The oretical Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] This kernel's theoretical occupancy (50.0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and n achieved occupancy (12.2%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances or between warps within a block as well as across blocks of the same kernel. See the (20.00.4 Best Principes Guide for more details on optimizing occup Impact of Varying Register Cocun Per Thread 96<	he hardware es, resulting i measured an occur pancy.	e's in 32 8 32
Theoretical Occupancy [%] 50 Block Limit Registers [block] Theoretical Active Warps per SM [warp] 32 Block Limit Marea Mem [block] Achieved Occupancy [%] 12.24 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit Warps [block] A Occupancy Limiters This kernel's theoretical occupancy (50.0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and m balances during the kernel execution. Load imbalances are between warps within a block as well as across blocks of the same kernel. See the the CUDA Best Practices Guide for more details on optimizing occup Impact of Varying Register Count Per Thread 96 96	measured an occur pancy.	4 32 8 32
Theoretical Active Warps per SM [warp] 32 Block Limit Shared Mem [block] Achieved Occupancy [%] 12.24 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] This kernel's theoretical occupancy (15.0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and in achieved occupancy (12.2%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can between warps within a block as well as across blocks of the same kernel. See the (10.04) Alest Practices Guide for more details on optimizing occup Impact of Varying Register Count Per Thread 96. Per Thread	measured an occur pancy.	32 8 32
Achieved Occupancy [%] 12.24 Block Limit Warps [block] Achieved Active Warps Per SM [warp] 7.83 Block Limit SM [block] A Occupancy Limiters A Occupancy	measured an occur pancy.	32
A Difference between values reliant reality This kernel's theoretical occupancy (50.0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and r Cocupancy Limiters achieved occupancy (12.2%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances are between warps within a block as well as across blocks of the same kernel. See the the CUUA Best Practices Suide for more details on optimizing occup Impact of Varying Register Count Per Thread	measured an occur pancy.	52
This kernel's theoretical occupancy (50.0%) is limited by the number of required registers. The difference between calculated theoretical (50.0%) and f achieved occupancy (12.2%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances ca between warps within a block as well as across blocks of the same kernel. See the ∰ cubA Best Practices Guide for more details on optimizing occup Impact of Varying Register Count Per Thread	measured an occur pancy.	
6 72		
	+24 +24	725
- · · · · · · · · · · · · · · · · · · ·	0 0	6
Impact of Varying Block Size		
64 64 64 32 32 4 6 4 8 6 4 8 6 4 8 5 6 4 8 6 4 8 6 4 8 6 4 8 7 4 8 6 4 8 7 4 8 7 4 8 7 4 8 7 4 8 7 4 8 7 4 8 7 4 8 7 4 7 4		
-+-++++++++++++++++++++++++++++++++++		T ^{1,024}
Block Size		

Use AMD ROCProfiler to profile SYCL code



© codeplay°

Live demo: profiling

- Record standard NSight Systems profile: nsys profile ./main
- Analyse the profile in the GUI: nsys-ui report1.nsys-rep
- Record a selection of NSight Compute metrics: ncu --section ComputeWorkloadAnalysis --section LaunchStats --section MemoryWorkloadAnalysis --section Occupancy --section SchedulerStats --section SpeedOfLight --section SpeedOfLight_RooflineChart -o ncu-profile ./main 16000 1
- Analyse the profile in the GUI: ncu-ui ncu-profile.ncu-rep
- Record ROCProfiler metrics: **rocprof** --sys-trace ./main
- Print the output CSV files, analyse the trace (results.json) in the web UI at <u>https://ui.perfetto.dev</u>



Common Optimisations

- Choose an optimal work group size
 - Read more in our <u>blog post</u>
- Follow recommended indexing of multi-dimensional arrays
 - In SYCL the right-most index should vary the fastest
- Consider forcing the inlining for some functions
- Apply loop unrolling
- Avoid code path divergence between threads
- Ensure coalesced memory access

Our website contains guides for performance <u>developer.code</u>play.com



Live demo: common optimisations

- Example project:
 - github.com/rafbiels/sycl-crowd-simulation/tree/demo-optimisation
 - Crowd simulation with live display, but comes also with a headless
 -DPROFILING_MODE=ON
- Check for optimisation potential in NSight Systems
- Analyse detailed metrics for the "hot" kernel in NSight Compute
- Showcase an optimisation reducing branching and thread divergence



Support for our plugins



Enterprise Support (currently NVIDIA only)

Our highest level of support, for large teams.

Direct access to Codeplay's engineers and expertise via scheduled calls.

A custom support plan tailored to your requirements.



Priority Support (currently NVIDIA only)

Suited to small teams and individuals.

Access to a ticketed support desk.

Accelerated response time for questions and requests.



Forum Support (NVIDIA and AMD)

A public forum moderated by Codeplay engineers.

Available for free.

Engage with the oneAPI community and our engineers.

https://codeplay.com/company/contact/

https://support.codeplay.com



Summary

- Using open standard oneAPI and SYCL brings you choice
- Possible to achieve performance portability with oneAPI and SYCL
- You can use Codeplay's plugins to target NVIDIA and AMD GPUs today
- Possible to debug and profile SYCL code with NVIDIA and AMD tools



oneAPI Plugins for NVIDIA/AMD

Scan QR code or visit developer.codeplay.com





Social Media

Don't forget to follow us for the latest updates!

@codeplaysoft \mathbb{X} @codeplaysoftware codeplay-software codeplay-software in



Disclaimers

A wee bit of legal

Performance varies by use, configuration and other factors.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

© Codeplay Software Ltd.. Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.