

HLRS Workshop September 2023

Intel[®] Distribution for GDB*

A Cross-Architecture Application Debugger

Pascal Bähr

pascal.rene.baehr@intel.com

The Intel logo, consisting of the word "intel" in a lowercase, sans-serif font, with a small registered trademark symbol (®) to its right. The logo is positioned in the bottom left corner of the slide, partially overlapping a decorative graphic of overlapping squares in shades of blue.

Agenda

- Why Intel® Distribution for GDB*?
- Key features
- System Requirements Overview
- DPC++ Linux* Demo
- Debugging Multi-Tile GPU
- C++: Debugging OpenMP* offload
- Other Debug Capabilities
- Demo: CLI on Linux
- Demo: Visual Studio Code via SSH

Why Intel® Distribution for GDB*?

Overview

- Companion tool to Intel compilers and libraries
- Cross-architecture debugging
- Unified debugging experience for oneAPI ecosystem
 - C, C++, SYCL, OpenMP, or Fortran
- Debug parallel and threaded applications
 - Single session for CPU and GPU code
 - Capable of handling thousands of threads simultaneously

Key features

- Command line debugging on the same machine: `gdb-oneapi`
- IDE Integration – Visual Studio, Visual Studio Code
 - 2 machines required: CPU host and GPU target
- Device support:

Multi-node debugging	MPI applications	Not supported
Multi-thread debugging	On the same GPU	Supported
Multi-user debugging	On the same GPU	Not supported; GPU is blocked by the debugger
Multi-target debugging	debug GPU and CPU code in the same session	Supported

Windows*

Language Support

Data Parallel C++ (DPC++)

C \ C++

Fortran

OpenMP

IDE Support

Microsoft Visual Studio 2022*

Visual Studio Code *

OS Support

Windows* 10, 64-bit

Windows* 11, 64-bit

GPUs

Intel® Arc™ Series

CPUs

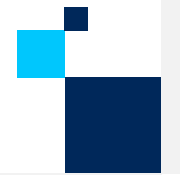
Intel® Core™ Processor family

Intel® Xeon® Processor family

Intel® Xeon® Scalable
Performance processors

FPGA

Emulation device only



Language Support

Data Parallel C++ (DPC++)

C \ C++

Fortran

OpenMP

IDE Support

Eclipse * (native)

Visual Studio Code *

OS Support

Ubuntu* 20.04, 22.04

SLES* 15

RHEL* 8, 9

GPUs

Intel® Arc™ Series

Intel® Data Center GPU Flex Series

Intel® Data Center GPU Max

CPUs

Intel® Core™ Processor family

Intel® Xeon® Processor family

Intel® Xeon® Scalable Performance processors

FPGA

Emulation device only



Other Debug Capabilities

oneAPI Debug Tools and Variables

- Specified level of tracing for SYCL Plugin Interface:
 - `SYCL_PI_TRACE={1, 2, -1}`
- GPU backends:
 - Profiling Tools Interfaces for GPU (PTI GPU) - [Level Zero Tracer ze_tracer](#)
 - Intercept Layer for OpenCL - [How to Use the Intercept Layer for OpenCL™ Applications](#)
- OpenMP Offload:
 - `LIBOMPTARGET_DEBUG={-1, 1, 2, 3}`
- Compiler options – more options are available Fortran!
- Clang Sanitizers, valgrind etc

Useful Links

■ Basic:

- [Documentation & Code Samples](#)
- [Intel® Distribution for GDB* Release Notes](#)
- [Intel® Distribution for GDB* System Requirements](#)

■ Advanced:

- [oneAPI Debug Tools at Intel® oneAPI Programming Guide](#)
- [Get Started with OpenMP* Offload to GPU for the Intel® oneAPI DPC/C++ Compiler and Intel® Fortran Compiler](#)

DPC++ Linux* Demo (Command Line)

Array Transform Sample

- Prerequisites:

- [Get Started Guide](#) to configure the debugger

- Clone [oneAPI-samples](#):

```
git clone https://github.com/oneapi-src/oneAPI-samples.git  
cd oneAPI-samples/Tools/ApplicationDebugger/array-transform
```

- Set oneAPI environment:

```
source /opt/intel/oneapi/setvars.sh
```

Array Transform Sample

- Enable i915 debug support in kernel persistently:

- **Requires sudo!**

- `cat /etc/default/grub`

- **Make sure your GRUB_CMDLINE_LINUX_DEFAULT contains:**

```
i915.debug_eu=1 drm.debug=0xa i915.enable_hangcheck=0  
i915.debugger_timeout_ms=0
```

- Enable i915 debug support in Kernel:

- `cat /sys/class/drm/card*/prelim_enable_eu_debug`

- Make sure the output is **1**

Diagnostics Utility

- For the default oneAPI installation:

- `python3 /opt/intel/oneapi/diagnostics/latest/diagnostics.py --filter debugger_sys_check -force`

- Expected output:

```
Checks results:
=====
Check name: debugger_sys_check
Description: This check verifies if the environment is ready to use gdb (Intel(R) Distribution for GDB*).
Result status: PASS
Debugger found.
libipt found.
libiga found.
i915 debug is enabled.
Environmental variables correct.
=====
1 CHECK: 1 PASS, 0 FAIL, 0 WARNINGS, 0 ERRORS
```

Array Transform Sample on CPU

- **Build:**

```
icpx -fsycl -g -O0 array-transform.cpp -o array-transform
```

- **Run:**

```
ONEAPI_DEVICE_SELECTOR=*:cpu ./array-transform
```

- **Run under the debugger:**

```
ONEAPI_DEVICE_SELECTOR=*:cpu gdb-oneapi --args ./array-transform
```

Array Transform Sample on GPU

- **Build:**

```
icpx -fsycl -g -O0 array-transform.cpp -o array-transform
```

- **Run:**

```
ONEAPI_DEVICE_SELECTOR=level_zero:gpu gdb-oneapi ./array-transform
```

- **Enable debugging:**

```
export ZET_ENABLE_PROGRAM_DEBUGGING=1  
export IGC_EnableGTLocationDebugging=1
```

- **Run under the debugger:**

```
ONEAPI_DEVICE_SELECTOR=level_zero:gpu gdb-oneapi --args ./array-transform
```

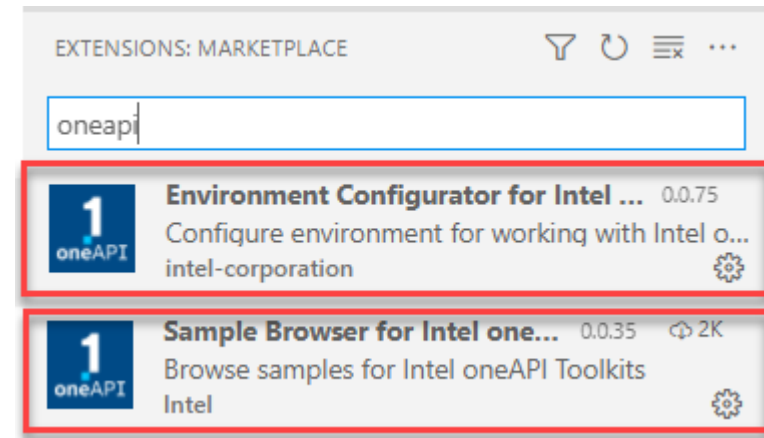

Debugging on GPU

- `info inferiors` - make sure you are on GPU now
- `info threads` - inspect threads
- `thread 2.<Thread_number>:<SIMD_lane>` - switching between threads
- `info locals` - print local threads variables
- `disassemble` - see disassemble
- `set scheduler-locking step` - step to the next

DPC++ Linux* Demo (Visual Studio Code - Remote)

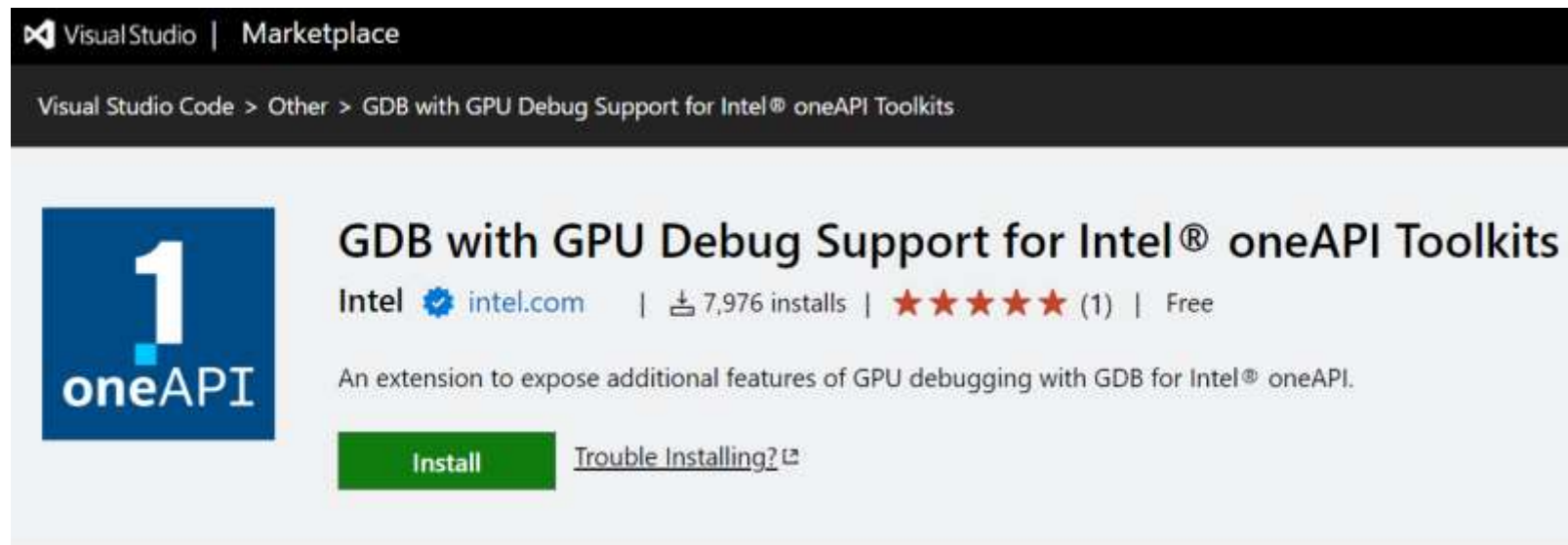
Setting up VS Code

- Prerequisites:
 - [Get Started Guide](#) to configure the debugger for remote debugging
 - Setup oneAPI environment on target machine
- Install oneAPI extensions for VSC on remote
- Install [oneAPI-samples](#) via sample browser:




Visual Studio Code – oneAPI GDB extension




- Install and [setup the oneAPI Debug extension for VS Code](#)



VisualStudio | Marketplace

Visual Studio Code > Other > GDB with GPU Debug Support for Intel® oneAPI Toolkits

 **GDB with GPU Debug Support for Intel® oneAPI Toolkits**

Intel  intel.com |  7,976 installs |  (1) | Free

An extension to expose additional features of GPU debugging with GDB for Intel® oneAPI.

[Install](#) [Trouble Installing?](#)

Visual Studio Code – oneAPI GDB extension

- Generate and setup a launch configuration
- Set environment variables
 - For debugging
 - For execution on GPU

```
"environment": [  
  {  
    "name": "ZET_ENABLE_PROGRAM_DEBUGGING",  
    "value": "1"  
  },  
  {  
    "name": "IGC_EnableGTLocationDebugging",  
    "value": "1"  
  },  
  {  
    "name": "ONEAPI_DEVICE_SELECTOR",  
    "value": "*:gpu"  
  }  
],
```

Visual Studio Code – oneAPI GDB extension

- Inspect variables

```
✓ VARIABLES
  ✓ Locals
    id0: 48
    element: -133246256
    result: -133247472
    ✓ this: 0xffffcfff80ec990
      > in
      > out
      ✓ index: {...}
        [0]: 48
```

```
50
51 // kernel-start
52 h.parallel_for(data_range, [=](id<1> index) {
53     size_t id0 = GetDim(index, 0);
54     int element = in[index]; // breakpoint-here
55     int result = element + 50;
56     if (id0 % 2 == 0) {
57         result = result + 50; // then-branch
58     } else {
59         result = -1; // else-branch
60     }
61     out[index] = result;
62 });
63 // kernel-end
64
65
```

Visual Studio Code – oneAPI GDB extension

- Inspect call stack





```
CALL STACK
> [12] PAUSED
v [13] PAUSED ON BREAKPOINT
  <in-memory@0x5b974c0-0x5f59c70>!main::{lambda(auto...
  <in-memory@0x5b974c0-0x5f59c70>!_ZTSZZ4mainENKU1RT
> [14] PAUSED
> [15] PAUSED
> [16] PAUSED
> [17] PAUSED
> [18] PAUSED
```

```
50
51 // kernel-start
52 h.parallel_for(data_range, [=](id<1> index) {
53     size_t id0 = GetDim(index, 0);
54     int element = in[index]; // breakpoint-here
55     int result = element + 50;
56     if (id0 % 2 == 0) {
57         result = result + 50; // then-branch
58     } else {
59         result = -1; // else-branch
60     }
61     out[index] = result;
62 });
63 // kernel-end
64 });
65
```

Visual Studio Code – oneAPI GDB extension

- Inspect GPU threads and SIMD Lanes

ONEAPI GPU THREADS

ThreadID	TargetID	Location	SIMD Lanes
5	Thread 1.1	main:: {lambda(auto:1...	
13	Thread 1.9	main:: {lambda(auto:1...	
37	Thread 1.33	main:: {lambda(auto:1...	
45	Thread 1.41	main:: {lambda(auto:1...	

```
50
51 // kernel-start
52 h.parallel_for(data_range, [=](id<1> index) {
53     size_t id0 = GetDim(index, 0);
54     int element = in[index]; // breakpoint-here
55     int result = element + 50;
56     if (id0 % 2 == 0) {
57         result = result + 50; // then-branch
58     } else {
59         result = -1; // else-branch
60     }
61     out[index] = result;
62 });
63 // kernel-end
64 });
65
```


Visual Studio Code – oneAPI GDB extension

- Inspect GPU threads and SIMD Lanes

```
▼ SELECTED LANE
Lane Number:      0
Thread Workgroup: x:0,y:0,z:0
Work item Global Id: x:48,y:0,z:0
Work item Local Id: x:48,y:0,z:0
Execution Mask:   0xffff
Hit Lanes Mask:  0xffff
SIMD Width:      16
```

```
50
51 // kernel-start
52 h.parallel_for(data_range, [=](id<1> index) {
53     size_t id0 = GetDim(index, 0);
54     int element = in[index]; // breakpoint-here
55     int result = element + 50;
56     if (id0 % 2 == 0) {
57         result = result + 50; // then-branch
58     } else {
59         result = -1; // else-branch
60     }
61     out[index] = result;
62 });
63 // kernel-end
64 });
65
```

Debugging Multi-Tile GPU

ZE_AFFINITY_MASK

Value	Behavior
0, 1	all devices and sub-devices are reported (same as default)
0	only device 0 is reported; with all its sub-devices
1	only device 1 is reported as device 0; with all its sub-devices
0.0	only device 0, sub-device 0 is reported as device 0
1.1	only device 1 is reported as device 0; with its sub-devices 1 and 2 reported as sub-devices 0 and 1, respectively
0.2, 1.3, 1.0, 0.3	both device 0 and 1 are reported; device 0 reports sub-devices 2 and 3 as sub-devices 0 and 1, respectively; device 1 reports sub-devices 0 and 3 as sub-devices 0 and 1, respectively; the order is unchanged.

Selecting Different Devices

- `$ gdb-oneapi --args ./array-transform`

```
(gdb) info devices
Location      Sub-device    Vendor Id     Target Id     Cores         Device Name
[3a:00.0]    -             0x8086       0x0bd5       1024         Intel(R) Graphics [0x0bd5]
* [9a:00.0]  -             0x8086       0x0bd5       1024         Intel(R) Graphics [0x0bd5]
```

- `$ ZE_AFFINITY_MASK=0.0 gdb-oneapi --args ./array-transform`

```
(gdb) info devices
Location      Sub-device    Vendor Id     Target Id     Cores         Device Name
* [9a:00.0]  -             0x8086       0x0bd5       512          Intel(R) Graphics [0x0bd5]
```

- `$ ZE_AFFINITY_MASK=1.0 gdb-oneapi --args ./array-transform`

```
(gdb) info devices
Location      Sub-device    Vendor Id     Target Id     Cores         Device Name
* [3a:00.0]  -             0x8086       0x0bd5       512          Intel(R) Graphics [0x0bd5]
```

Debugging OpenMP* Offload (C++)

Matmul build and run

■ Build:

- `icpx -O0 -g -fiopenmp -fopenmp-targets=spir64 matmul_offload.cpp -o matmul_debug`

■ Disable device optimizations:

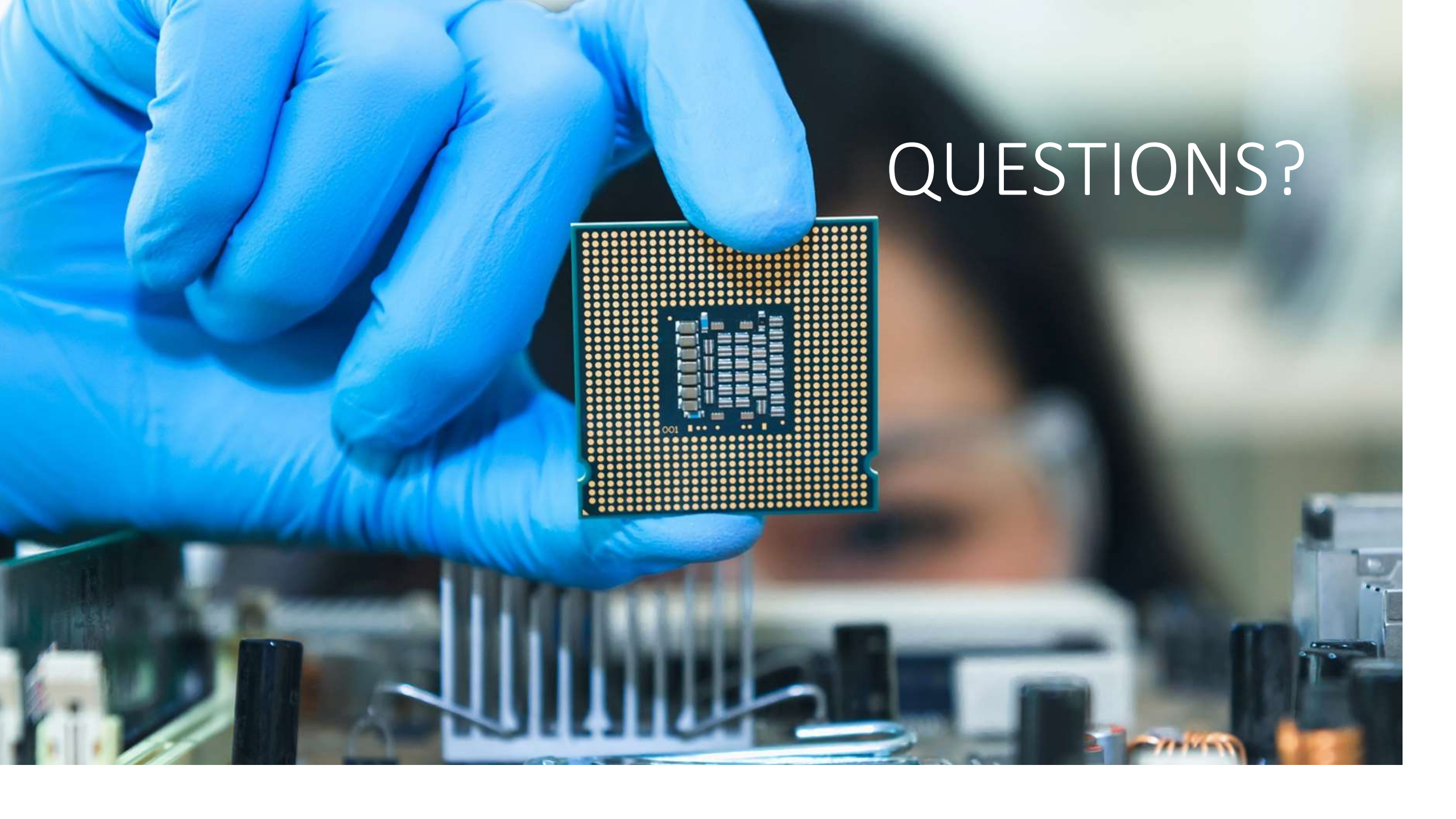
```
export ZET_ENABLE_PROGRAM_DEBUGGING=1
export IGC_EnableGTLocationDebugging=1
```

■ Set up offloading:

- `export OMP_TARGET_OFFLOAD="MANDATORY"`

■ Debug:

- `gdb-oneapi ./matmul_debug`



QUESTIONS?

Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

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.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

© Intel Corporation. Intel, the Intel logo, Xeon, Core, VTune, OpenVINO, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

intel®