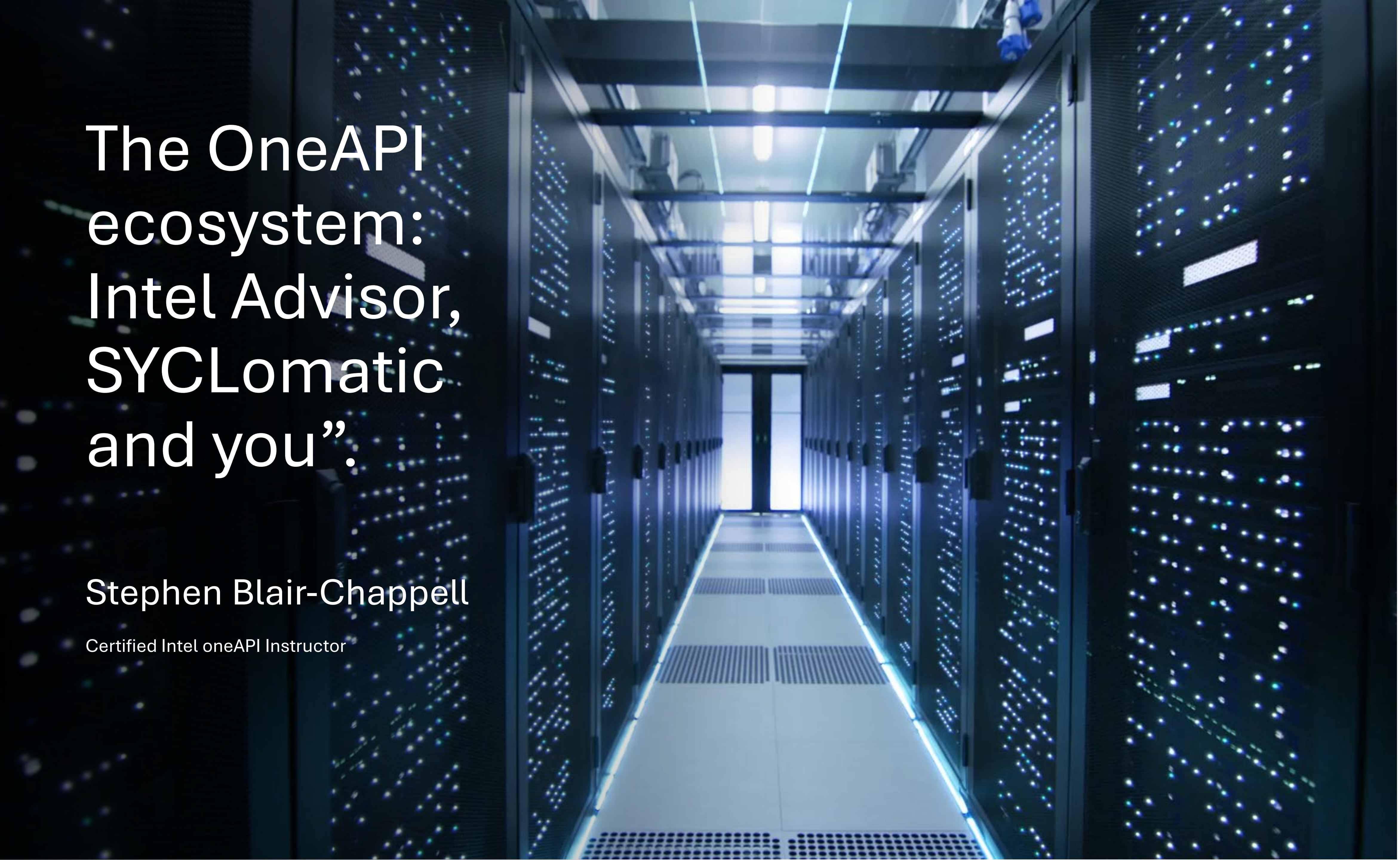


The OneAPI ecosystem: Intel Advisor, SYCLomatic and you”.

Stephen Blair-Chappell

Certified Intel oneAPI Instructor

About Me

Stephen Blair-Chappell

- Freelance software consultant
- Intel Certified oneAPI instructor.
- Formerly the Technical Director at Bayncore
- For 18 years he was a Technical Consulting Engineer at Intel
- Author of the book "Parallel Programming with Intel Parallel Studio XE".
- Sponsored by Lenovo to do some work at their Centre’s of Excellence.

When I’m not working

- Trustee of two charities involved with mental health & young people.
- Adventurous DIY
- Play the Pipe Organ

oneAPI Spec Elements

The Spec is made of 7 core elements.

oneDPL

oneAPI Data Parallel C++ Library

A companion to the DPC++ Compiler for programming oneAPI devices with APIs from C++ standard library, Parallel STL, and extensions.

oneDNN

oneAPI Deep Neural Network Library

High performance implementations of primitives for deep learning frameworks.

oneCCL

oneAPI Collective Communications Library

Communication primitives for scaling deep learning frameworks across multiple devices.

Level Zero

oneAPI Level Zero

System interface for oneAPI languages and libraries.

oneDAL

oneAPI Data Analytics Library

Algorithms for accelerated data science.

oneTBB

oneAPI Threading Building Blocks

Library for adding thread-based parallelism to complex applications on multiprocessors.

oneMKL

oneAPI Math Kernel Library

High performance math routines for science, engineering, and financial applications.

https://uxlfoundation.org/

Intel® oneAPI Toolkits

Intel® oneAPI Base Toolkit	A core set of high-performance libraries and tools for building C++, SYCL, C/OpenMP, and Python applications
Add-on **Domain-specific** Toolkits	For HPC developers **Intel® oneAPI Tools for HPC** Deliver fast Fortran, OpenMP & MPI applications that scale For visual creators, scientists & engineers **Intel® oneAPI Rendering Toolkit** Accelerate visual compute, deliver high-performance, high-fidelity visualization applications. For edge & IoT developers **Intel® oneAPI Tools for IoT** Build efficient, reliable solutions that run at network's edge
Toolkits **powered by oneAPI**	For AI developers & data scientists **Intel® AI Analytics Toolkit** Accelerate machine learning & data science pipelines end-to-end with optimized DL & ML frameworks & high-performing Python libraries For deep learning inference developers **Intel® OpenVINO™ toolkit** Deploy high performance inference & applications from edge to cloud

Download at **intel.com/oneAPI**
Or visit Intel® DevCloud for oneAPI

Intel® oneAPI Base Toolkit

Accelerate Data-centric Workloads

A core set of core tools and libraries for developing high-performance applications on Intel® CPUs, GPUs, and FPGAs.

Who Uses It?

- A broad range of developers across industries
- Add-on toolkit users since this is the base for all toolkits

Top Features/Benefits

- Data Parallel C++ compiler, library and analysis tools
- SYCLomatic / DPC++ Compatibility tool helps migrate CUDA code to C++ with SYCL
- Python distribution includes accelerated scikit-learn, NumPy, SciPy libraries
- Optimized performance libraries for threading, math, data analytics, deep learning, and video/image/signal processing

[Learn More & Download]

Intel® oneAPI Base Toolkit

Direct Programming

- Intel® oneAPI DPC++/C++ Compiler
- Intel® DPC++ Compatibility Tool
- Intel® Distribution for Python
- Intel® FPGA Add-on for oneAPI Base Toolkit

API-Based Programming

- Intel® oneAPI DPC++ Library oneDPL
- Intel® oneAPI Math Kernel Library - oneMKL
- Intel® oneAPI Data Analytics Library - oneDAL
- Intel® oneAPI Threading Building Blocks - oneTBB
- Intel® oneAPI Collective Communications Library oneCCL
- Intel® oneAPI Deep Neural Network Library - oneDNN
- Intel® Integrated Performance Primitives - Intel® IPP

Analysis & debug Tools

- Intel® VTune™ Profiler
- Intel® Advisor
- Intel® Distribution for GDB

Intel Advisor

Intel® Advisor

Configure Your Accelerated Computing Solution

Offload Advisor

Estimate performance of offloading to an accelerator

Roofline Analysis

Optimize CPU/GPU code for memory and compute

Vectorization Advisor

Add and optimize vectorization

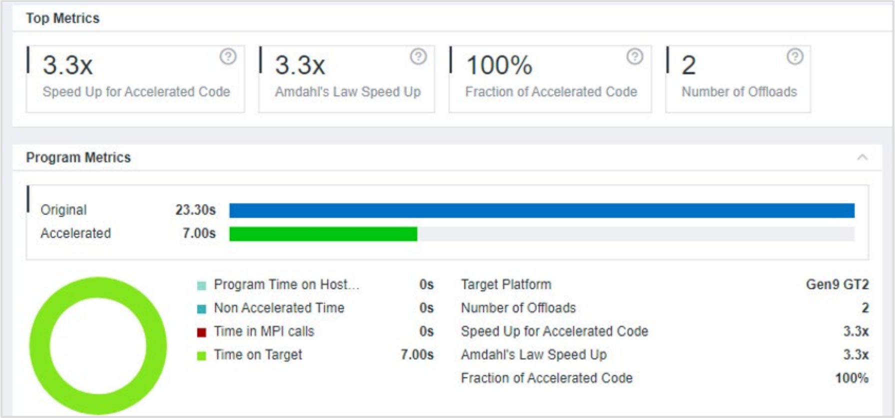
Threading Advisor

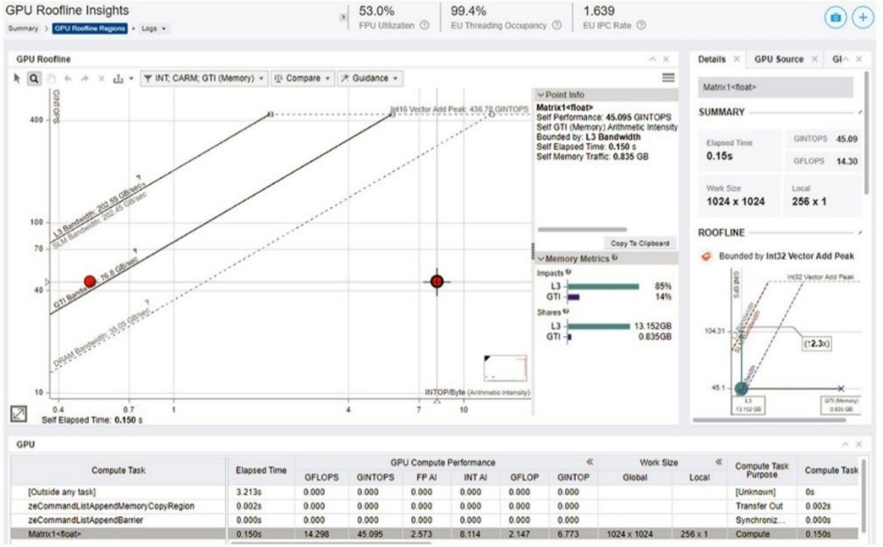
Add effective threading to unthreaded applications

Flow Graph Analyzer

Create and analyze efficient flow graphs

[Learn More & Download]





Know Before You Offload

Advisor Offload Modelling

Offload Modelling – doing it by hand

1. Run code on CPU and find hotspots — CPU
2. Examine results – decide which hotspots are suitable for offloading
3. Implement offload using SYCL or OpenMP
4. Run code on CPU and GPU and see if there is a speed up — CPU — GPU

Potentially time-consuming.
Possibly no return-on-investment

Offload Modelling – with Advisor

1 Run code on CPU using Advisor (multiple stage profiling collection)

CPU

2 Examine results – decide which hotspots are suitable for offloading

3 Implement offload using SYCL or OpenMP

4 Run code on CPU and GPU and see if there is a speed up

CPU — GPU

Advisor Simulates all these stages

CPU

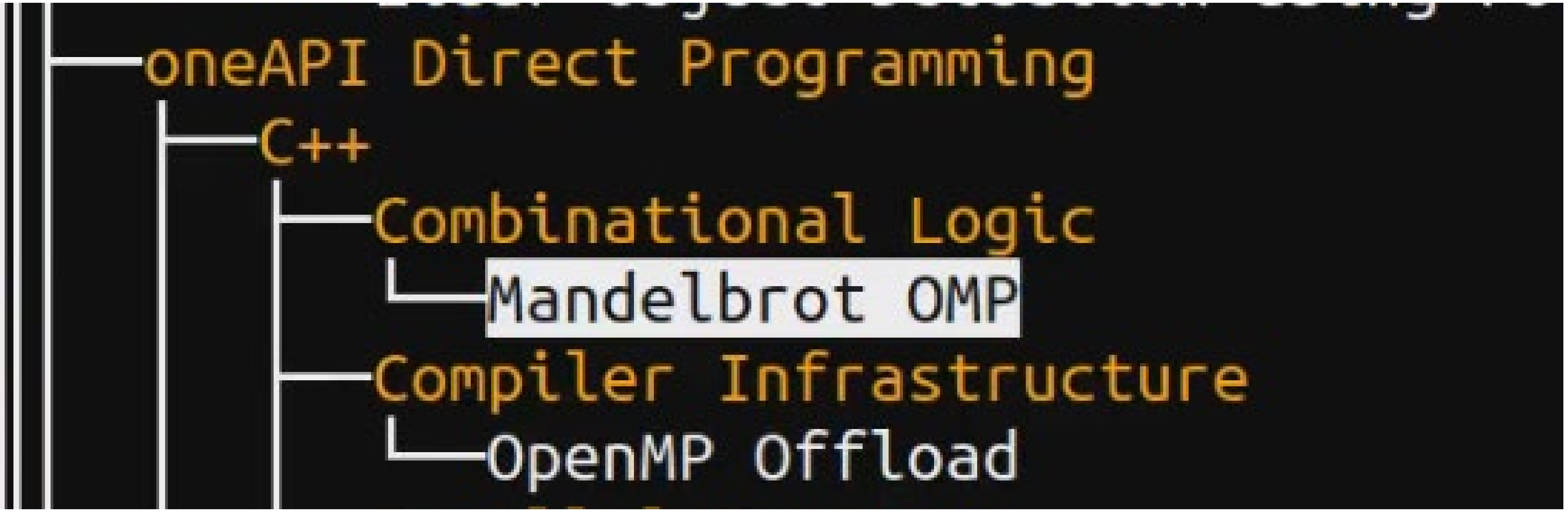
Report

[Generate Pre-configured Command Lines (intel.com)]

intel

Demo – An example you could experiment with

- git clone https://github.com/oneapi-src/oneAPI-samples.git

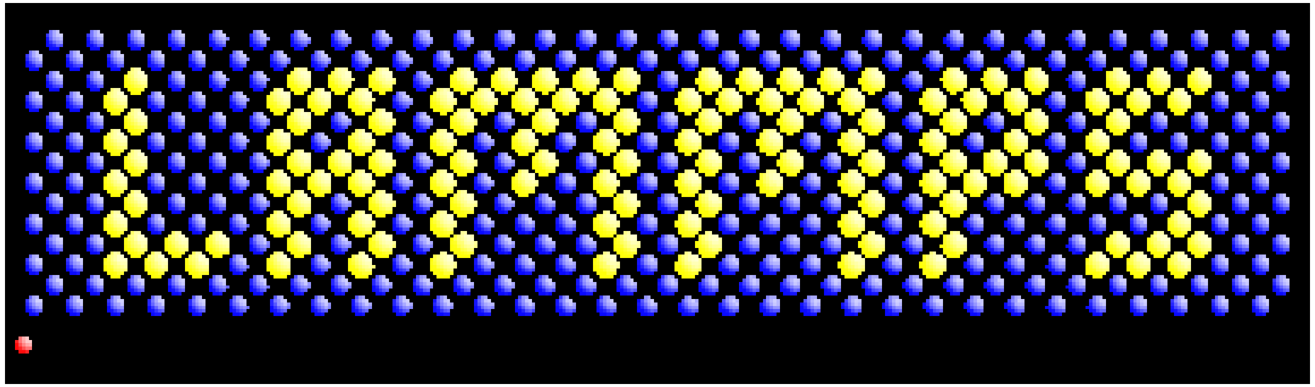


make CXX=icpx EXTRA_CFLAGS=-g

advisor --collect=offload -- ./release/Mandelbrot

OUR EXAMPLE

LAMMPS Molecular Dynamics Simulator



https://www.lammps.org/

- DO THE DEMO NOW!

Using accuracy presets to control modelling

- Default (Medium Accuracy)
```
advisor --collect=offload --config=gen12_tgl
--project-dir=./cpu2gpu_offload_modeling --
./release/Mandelbrot 1
```

- Low Accuracy
```
advisor --collect=offload -accuracy=low
--config=gen12_tgl --project-
dir=./cpu2gpu_offload_modeling --
./release/Mandelbrot 1
```

- Getting list of steps
```
advisor --collect=offload --dry-run
--config=gen12_tgl --project-
dir=./cpu2gpu_offload_modeling --
./release/Mandelbrot 1
```

Low	Medium	High
5-10x overhead	15-50x overhead	50-80x overhead
Survey Trip Count Offload Modelling	Survey Trip Count Offload Modelling	Survey Trip Count Dependency analysis Offload Modelling
L1 Cache	L1 Cache + Host-Device data	L1 Cache + Host-Device data

Steps to Offload Projection with Advisor

Expensive Steps

1. Run a **Survey**: get a list of hotspots

```
advisor -collect survey …
```

- Sampling
- Binary Static Analysis
- Compiler & debug info

2. Run a **Trip Count**: count loop iteration

```
advisor -collect=tripcounts -target-device=gen9_gt2 .
```

- Trip count
- Cache simulation

3. Perform a **dependency analysis** [optional for quick modelling]

```
advisor -collect dependencies . . .
```

- Check memory accesses
- Loop selection heuristic

4. **Model** the Performance

```
advisor -collect projection -no-assume-dependencies . . .
```

- Generate HTML report

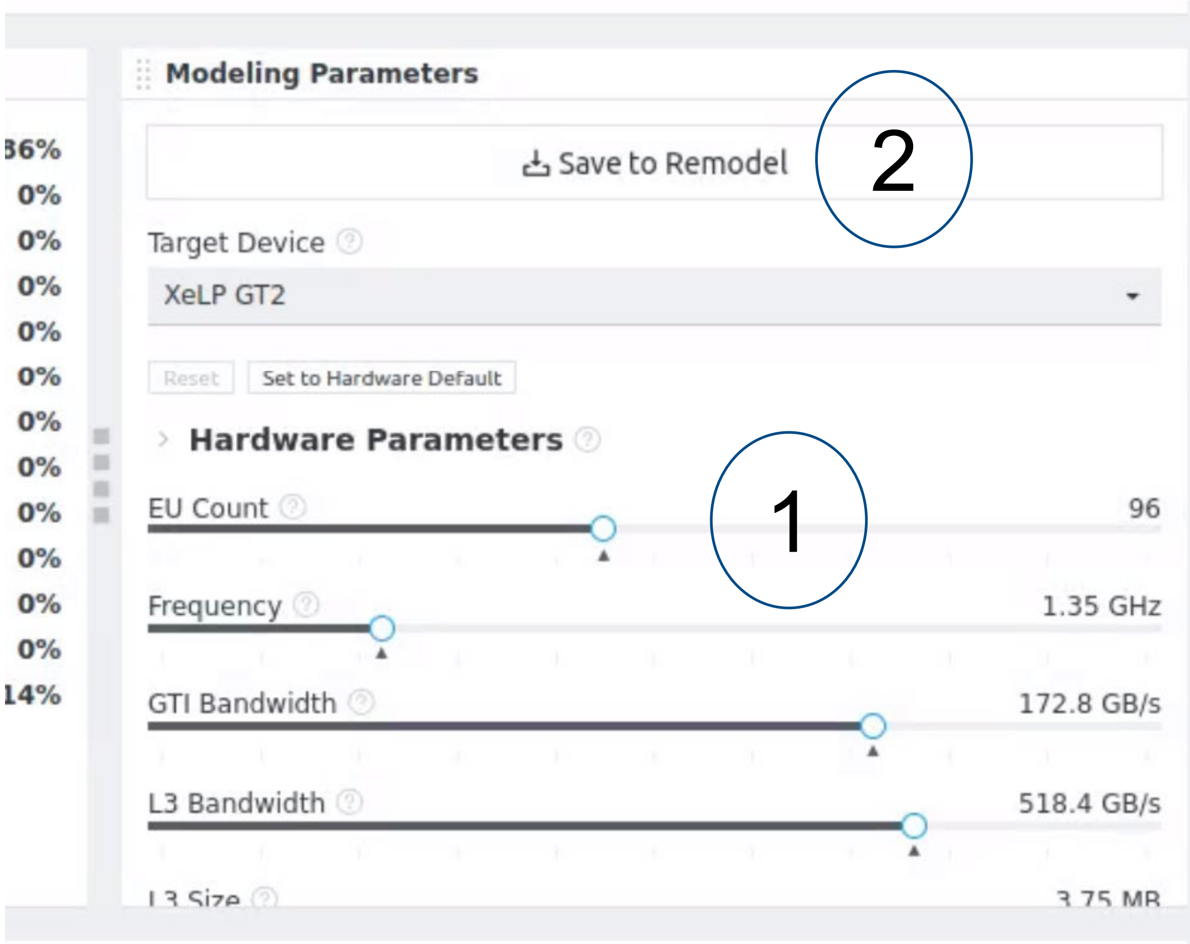
GPU-GPU modelling

- Add the –gpu flag
- Runs code on ACTUAL GPU and models against new GPU.
- Use when upgrading from one GPU to another one.
- NB: use –accuracy=low

Comparison of CPU-GPU and GPU-GPU modelling

CPU-GPU	GPU-GPU
Survey	Survey (on GPU)
Trip Count	Trip Count (Characterization - num Floats and Integer operations)
Dependency Check	x
Model Offloading	Model Offloading

Re-modelling for a different GPU
- once you already have a set of results



1. Adjust Values
2. Save parameters
3. Re-run modelling

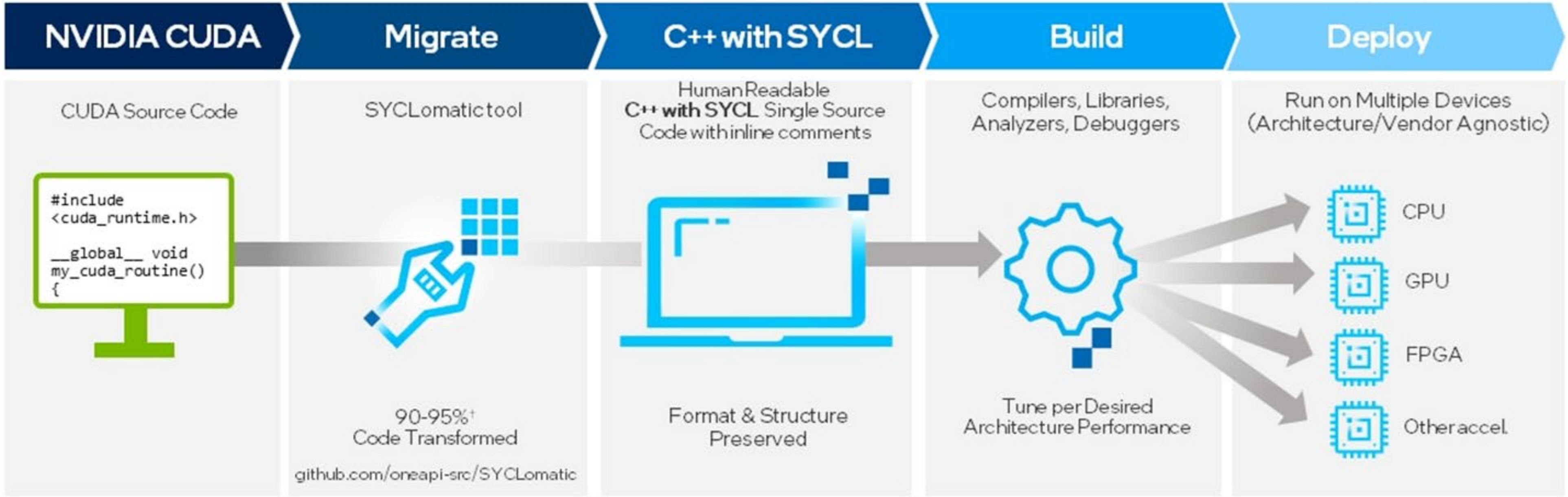
e.g.: advisor -c=projection --custom-config=config.toml --config=gen12_tgl

Advantage: quicker than doing a completely new modelling

SYCLomatic

CUDA to SYCL Migration Made Easy

Open Source SYCLomatic Tool Reduces Code Migration Time



Assists developers migrating code written in CUDA to C++ with SYCL, generating **human readable** code wherever possible

~90-95% of code typically migrates automatically[1]

Inline comments are provided to help developers finish porting the application

Intel® DPC++ Compatibility Tool is Intel's implementation, available in the Base Toolkit

[1]Intel estimates as of September 2021. Based on measurements on a set of 70 HPC benchmarks and samples, with examples like Rodinia, SHOC, PENNANT. Results may vary.

Intel® DPC++ Compatibility Tool

Migration of Large Code Bases

Prepare Intercept-Build

Migrate dpct

Review Verify & Manually Edit

1. Create a compilation database file

   intercept-build make

2. Migrate your source to DPC++

   dpct -p compile_commands.json -in-root=$PROJ_DIR -out-root=dpcpp_out *.cu

3. Verify the source for correctness and fix not migrated parts

BlackScholes Demo

SYCLOMATIC on BlackScholes NVIDIA CUDA example

- `git clone` https://github.com/NVIDIA/cuda-samples.git
- `cd Samples/5_Domain_Specific/BlackScholes`
- `make clean`
- `intercept-build make`
- `mkdir ../migration`
-
```
dpct --cuda-include-path /usr/local/cuda-12/include \
-p compile_commands.json --in-root=. --out-root=../migration
```
- `cp Makefile ../migration`

Changes to make command line

- `make: *** No rule to make target 'BlackScholes.cu', needed by 'BlackScholes.o'. Stop.`

```
BlackScholes.o:BlackScholes.cu  =>
BlackScholes.o:BlackScholes.dp.cpp
```

- `BlackScholes.dp.cpp:34:10: fatal error: sycl/sycl.hpp: No such file or directory`

```
make "NVCC=icpx -fsycl“
```

- `icpx: error: unsupported option '--threads'; did you mean '-mthreads’?`
  `icpx: error: unknown argument: '-maxrregcount=16’`
  `icpx: error: unknown argument: '-gencode’`

```
make "NVCC=icpx -fsycl" ALL_CCFLAGS="" GENCODE_FLAGS="-g -O3“
```

Changes to `BlackScholes.dp.cpp`

- `BlackScholes.dp.cpp:106:3: error: use of undeclared identifier 'findCudaDevice'`
  ```
  checkCudaErrors(DPCT_CHECK_ERROR(
  findCudaDevice(argc, (const char **)argv);
  getLastCudaError("BlackScholesGPU() execution failed\n");
  ```
- add to `BlackScholes.dp.cpp`
  ```
  void checkCudaErrors(int err){}
  void findCudaDevice(int argc,const char** argv){}
  void getLastCudaError(char * strErr){}
  ```

Different Performance

CUDA

- Executing Black-Scholes GPU kernel (512 iterations)...
- Options count : 8000000
- BlackScholesGPU() time : 0.158816 msec
- Effective memory bandwidth: 503.726277 GB/s
- Gigaoptions per second : 50.372628

- BlackScholes, Throughput = 50.3726 GOptions/s, Time = 0.00016 s, Size = 8000000 options, NumDevsUsed = 1, Workgroup = 128

SYCL

- Executing Black-Scholes GPU kernel (512 iterations)...
- Options count : 8000000
- BlackScholesGPU() time : 1.463035 msec
- Effective memory bandwidth: 54.680848 GB/s
- Gigaoptions per second : 5.468085

- BlackScholes, Throughput = 5.4681 GOptions/s, Time = 0.00146 s, Size = 8000000 options, NumDevsUsed = 1, Workgroup = 128

Checking app is running on correct GPU

- Check what accelerators are available

  [opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2023.16.6.0.22_223734]
  [opencl:cpu:1] Intel(R) OpenCL, 13th Gen Intel(R) Core(TM) i9-13900HX 3.0 [2023.16.6.0.22_223734]
  [opencl:gpu:2] Intel(R) OpenCL HD Graphics, Intel(R) Graphics [0xa788] 3.0 [22.24.23453]
  [ext_oneapi_level_zero:gpu:0] Intel(R) Level-Zero, Intel(R) Graphics [0xa788] 1.3 [1.3.23453]
  [ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA GeForce RTX 4090 Laptop GPU 8.8 [CUDA 12.0]

- Run app with SYCL_PI_TRACE

  SYCL_PI_TRACE=1 ./Blackscholes
  SYCL_PI_TRACE[all]: Selected device: -> final score =
  SYCL_PI_TRACE[all]:   platform: Intel(R) Level-Zero
  SYCL_PI_TRACE[all]:   device: Intel(R) Graphics [0xa788]
  [./BlackScholes] - Starting...

Running on Intel GPU
not NVIDIA GPU !

Force app to use NVIDIA GPU using SYCL_DEVICE_FILTER

- SYCL_DEVICE_FILTER=cuda SYCL_PI_TRACE=1 ./BlackScholes

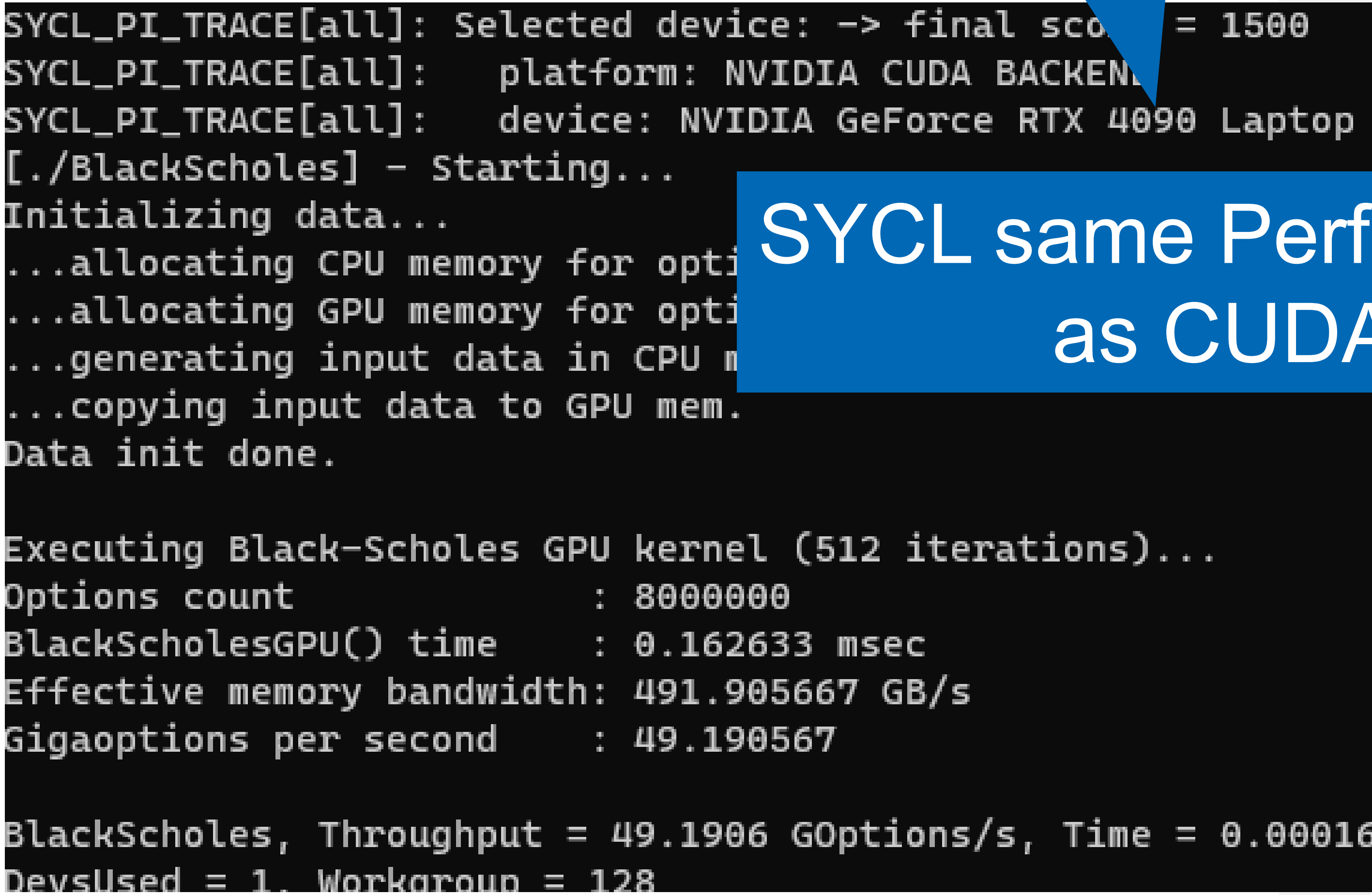
```
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_cuda.so [ PluginVersion: 12.27.1 ]
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Selected device: -> final score = 500
SYCL_PI_TRACE[all]:   platform: NVIDIA CUDA BACKEND
SYCL_PI_TRACE[all]:   device: NVIDIA GeForce RTX 4090 Laptop GPU
[./BlackScholes] - Starting...
Initializing data...
...allocating CPU memory for options.
...allocating GPU memory for options.
...generating input data in CPU mem.
...copying input data to GPU mem.
Data init done.

Executing Black-Scholes GPU kernel (512 iterations)...
terminate called after throwing an instance of 'sycl::_V1::runtime_error'
  what():  Native API failed. Native API returns: -42 (PI_ERROR_INVALID_BINARY) -42 (PI_ERROR_INVALID_BINARY)
Aborted
```

Invalid Binary: looks like wrong options were used in initial build!

Rebuilding and Running fixes the problem

NVIDIA GPU

- App should have been built with ‘option’ -fsycl-targets=nvptx64-nvidia-cuda
- make NVCC="icpx -fsycl" ALL_CCFLAGS="" GENCODE_FLAGS="-g -O3 -fsycl-targets=nvptx64-nvidia-cuda "



SYCL same Perf as CUDA

```
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_cuda.so [ PluginVersion: 12.27.1 ]
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Selected device: -> final score = 1500
SYCL_PI_TRACE[all]:   platform: NVIDIA CUDA BACKEND
SYCL_PI_TRACE[all]:   device: NVIDIA GeForce RTX 4090 Laptop GPU
[./BlackScholes] - Starting...
Initializing data...
...allocating CPU memory for options.
...allocating GPU memory for options.
...generating input data in CPU mem.
...copying input data to GPU mem.
Data init done.

Executing Black-Scholes GPU kernel (512 iterations)...
Options count            : 8000000
BlackScholesGPU() time    : 0.160705 msec
Effective memory bandwidth: 497.806309 GB/s
Gigaoptions per second    : 49.780631

BlackScholes, Throughput = 49.7806 GOptions/s, Time = 0.00016 s, Size = 8000000 options, NumDevsUsed = 1, Workgroup = 12
8
```

and
YOU

Backup

Lammps offloading

Stephen Blair-Chappell

Test Application

LAMMPS Molecular Dynamics Simulator

https://www.lammps.org/

Machine 1 Spec - Workstation

```
u58345@s011-n001:~/ILDevCON/lammps-offload-OneAPI/LAB3/1-Model-Offload$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          24
On-line CPU(s) list:             0-23
Thread(s) per core:              2
Core(s) per socket:              12
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           85
Model name:                      Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz
Stepping:                        7
CPU MHz:                         1200.315
CPU max MHz:                     4800.0000
CPU min MHz:                     1200.0000
BogoMIPS:                        6999.82
Virtualization:                  VT-x
L1d cache:                       384 KiB
L1i cache:                       384 KiB
L2 cache:                        12 MiB
L3 cache:                        19.3 MiB
NUMA node0 CPU(s):               0-23
Vulnerability Itlb multihit:     KVM: Mitigation: Split huge pages
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:        Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Mitigation; TSX disabled
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acp
                                 i mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_per
                                 fmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64
                                  monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic
                                  movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cp
                                 uid_fault epb cat_l3 cdp_l3 invpcid_single ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_s
                                 hadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms in
                                 vpcid cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt avx512cd
                                 avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cq
                                 m_mbm_local dtherm ida arat pln pts hwp hwp_act_window hwp_epp hwp_pkg_req avx512_vnni
                                 md clear flush l1d arch capabilities
```

```
 Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz
```

```
CPU(s):                          24
On-line CPU(s) list:             0-23
Thread(s) per core:              2
```

```
vx2 smep bmi2 erms in
wb intel_pt avx512cd
_llc cqm_mbm_total cq
_pkg_req avx512_vnni
```

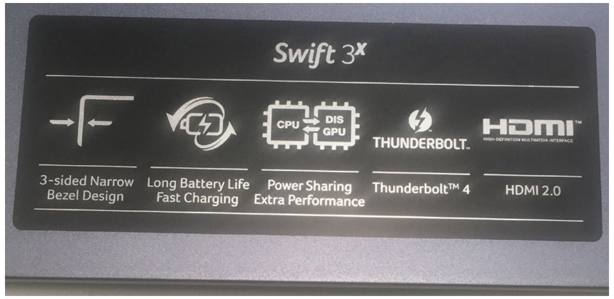
Machine 2 Spec - Laptop

```
stephen@stephen-Swift-SF314-510G:~/dv/OneAPI-Package-1/LAMMPS$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          8
On-line CPU(s) list:             0-7
Thread(s) per core:              2
Core(s) per socket:              4
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           140
Model name:                      11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
```

```
u58345@s011-n001:~/ILDevCON/lammps-offload-OneAPI/LAB3/1-Model-Offload$ lspci | grep VGA
1c:00.0 VGA compatible controller: Intel Corporation Device 4905 (rev 01)
6a:00.0 VGA compatible controller: Intel Corporation Device 4905 (rev 01)
```

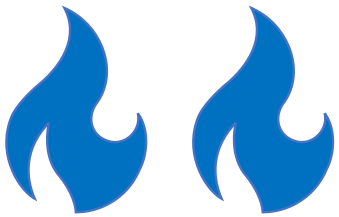
Graphics processor table

PCI IDs	Name	Architecture	Codename
4905	Intel® Iris® Xe MAX Graphics	Xe	DG1



Analyze

Goal of Analysis



1. Find the ‘hot spots’

2. if possible, predict benefit of offloading

Three Tools

APS – (Application Performance Snapshot)

Advisor - model offloading

VTune - Profiler

13

APS – (Application Performance Snapshot)

APS report

File | C:/Users/STEPHE~1/ONEDRI~1/DOCUME~1/MobaXterm/slash/stephenblaircha_desktopikul9t5/RemoteFiles/133800_2_1/aps_report_20210923_065855.html

Application Performance Snapshot

Application: lmp
Report creation date: 2021-09-23 06:58:02
Number of ranks: 16
Ranks per node: 16
HW Platform: Intel(R) microarchitecture code named Tigerlake
Frequency: 2.42 GHz
Logical Core Count per node: 8
Collector type: Event-based sampling driver,Event-based counting driver

7.87 s CPU
Elapsed Time
0.06 s GPU

1 IPC Rate

0 SP GFLOPS

1.43 DP GFLOPS

2.88 GHz Average CPU Frequency

Your application may underutilize the GPU.

Run a GPU Offload (Preview) or a GPU Compute/Media Hotspots (Preview) analysis with VTune Profiler to discover how to better utilize the GPU.

	Current run	Target	Tuning Potential
MPI Time	54.64%	<10%	
Memory Stalls	7.2%	<20%	
Vectorization	0%	>70%	
GPU Utilization when Busy	3.6%	>80%	

GPU Utilization when Busy

3.6%

EU State	% of EUs
Active	3.6%
Idle	83%
Stalled	13.4%

GPU Occupancy
9.1% of Peak Value

Memory Footprint

Resident
113.19 MB

Resident per Node
1811 MB

Virtual
2646.12 MB

Virtual Per Node
42338 MB

MPI Time

4.3 s
54.64% of Elapsed Time

MPI Imbalance
2.38 s
30.28% of Elapsed Time

TOP 5 MPI Functions	% of Elapsed Time
MPI_Bcast	28.84%
MPI_Wait	9.86%
MPI_Allreduce	7.87%
MPI_Init	5.4%
MPI_Sendrecv	1.69%

Memory Stalls

7.2% of Pipeline Slots

Cache Stalls
19.2% of Cycles

DRAM Stalls
7.5% of Cycles

DRAM Bandwidth

Average	9 GB/s
Peak	8.9 GB/s
Bound	0%

Vectorization

0%

Instruction Mix

SP FLOPs
0% of uOps

DP FLOPs
6.9% of uOps
Packed:
0% from DP FP
128-bit:
0%
256-bit:
0%
512-bit:
0%
Scalar:
100% from DP FP

Non-FP
93.1% of uOps

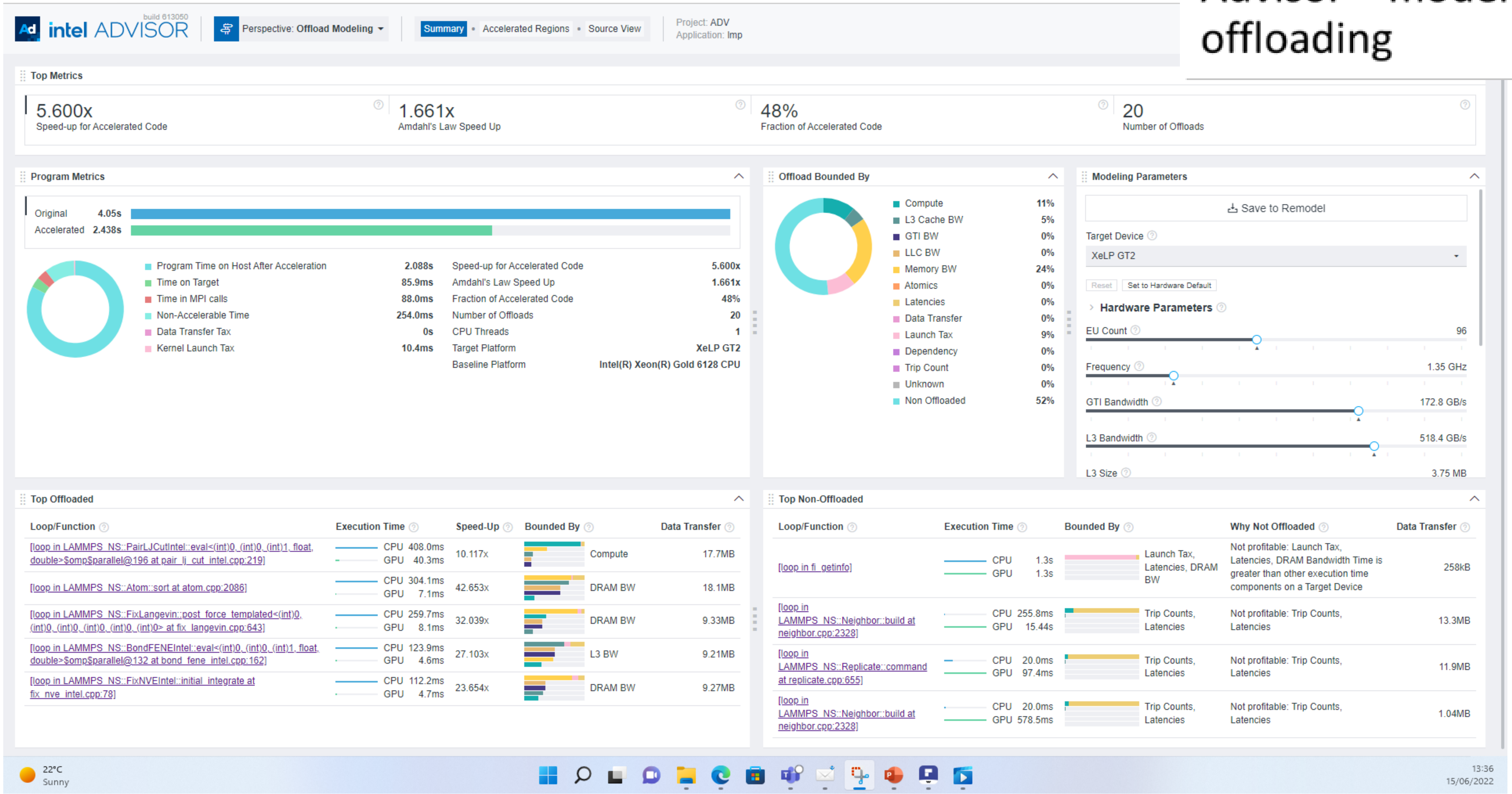
FP Arith/Mem Rd Instr. Ratio
0.21

FP Arith/Mem Wr Instr. Ratio
0.73

intel

LAB1

Advisor - model offloading



Steps to Offload Projection with Advisor

1. Run a **Survey**: get a list of hotspots

   `advisor -collect survey …`

   - Sampling
   - Binary Static Analysis
   - Compiler & debug info

2. Run a **Trip Count**: count loop iteration

   `advisor -collect=tripcounts -target-device=gen9_gt2 .`

   - Trip count
   - Cache simulation

3. Perform a **dependency analysis** [optional for quick modelling]

   `advisor -collect dependencies . . .`

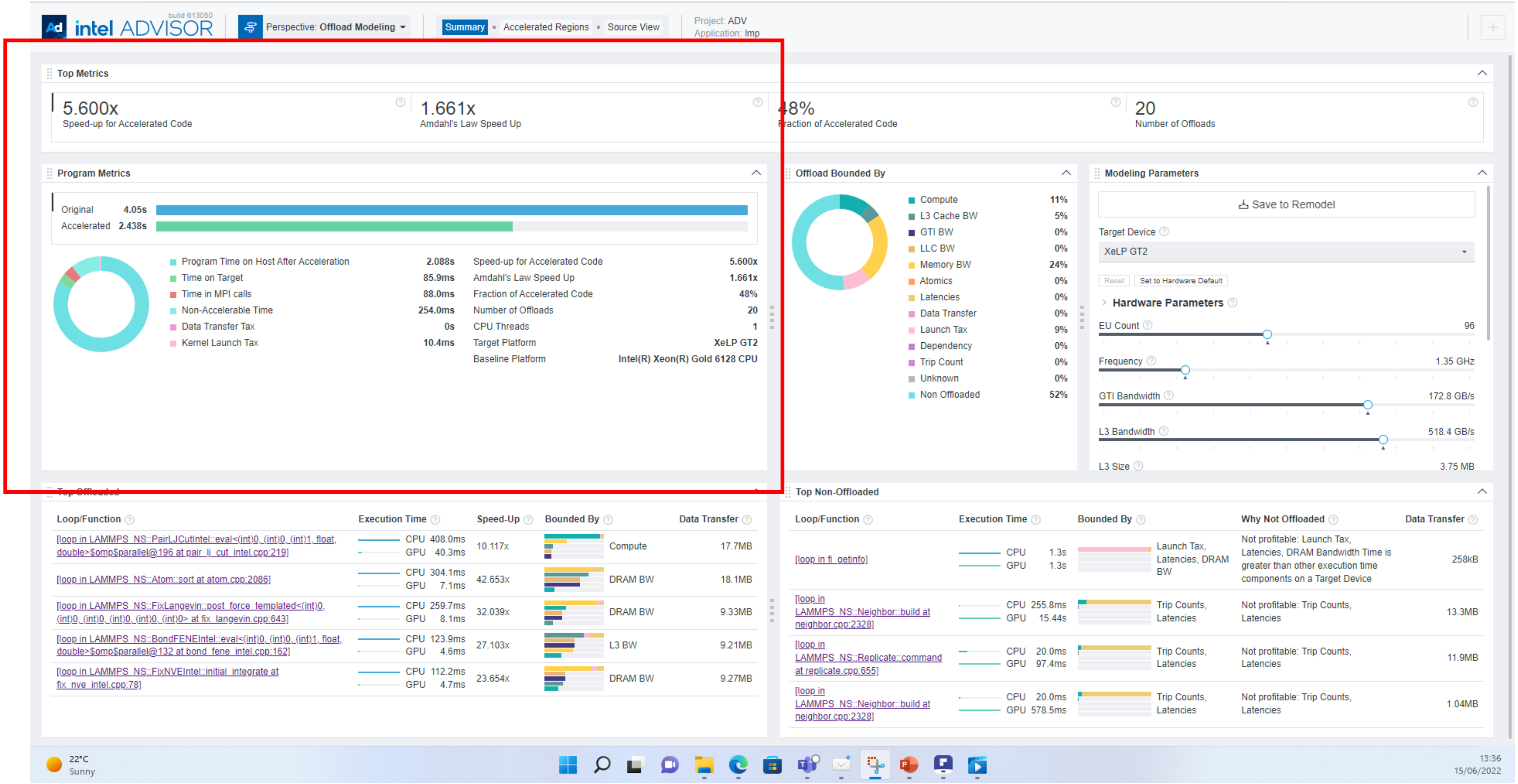
   - Check memory accesses
   - Loop selection heuristic

4. **Model** the Performance

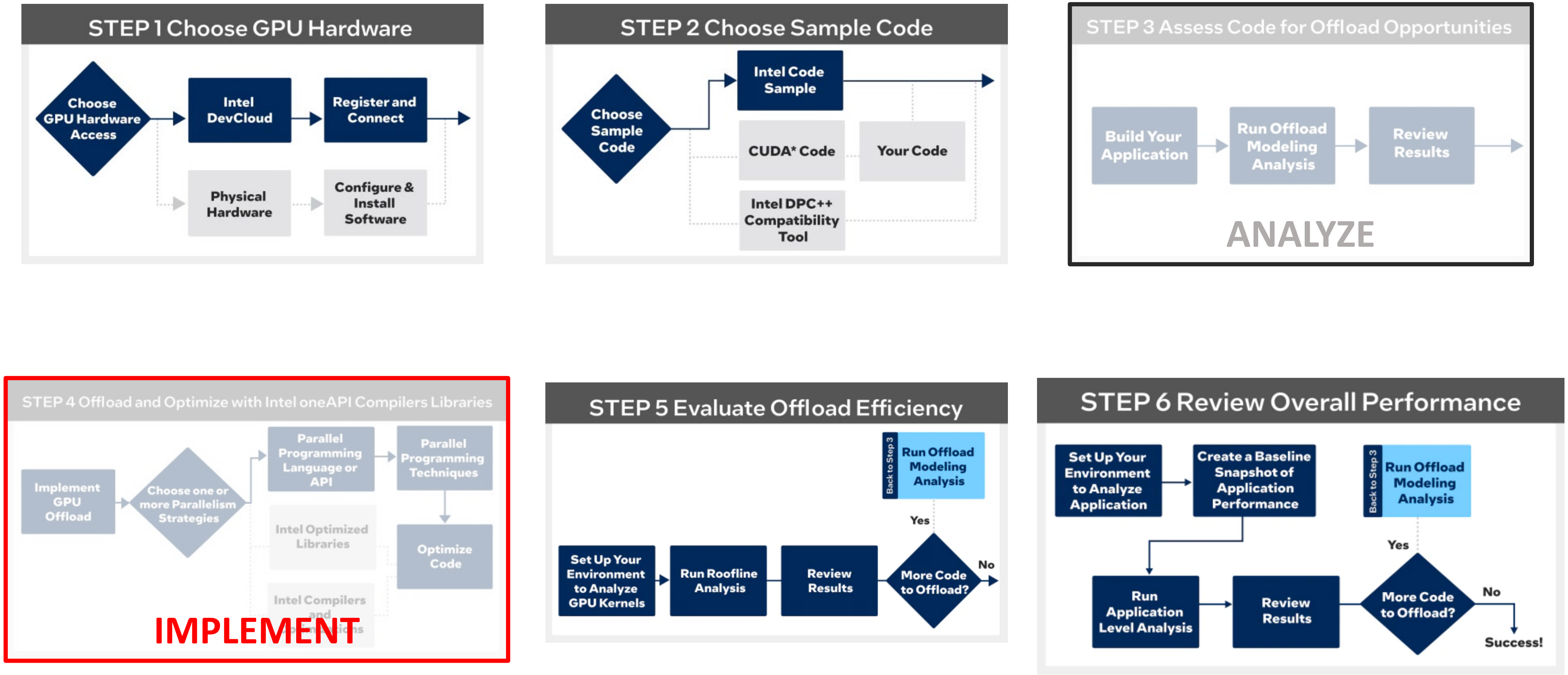
   `advisor -collect projection -no-assume-dependencies . . .`

   - Generate HTML report

Expensive Steps



intel ADVISOR
build 613050
Perspective: Offload Modeling
Summary
Accelerated Regions
Source View
Project: ADV
Application: lmp
Top Metrics
5.600x
Speed-up for Accelerated Code
1.661x
Amdahl's Law Speed Up
48%
Fraction of Accelerated Code
20
Number of Offloads
Program Metrics
Original 4.05s
Accelerated 2.438s
Program Time on Host After Acceleration 2.088s
Time on Target 85.9ms
Time in MPI calls 88.0ms
Non-Accelerable Time 254.0ms
Data Transfer Tax 0s
Kernel Launch Tax 10.4ms
Speed-up for Accelerated Code 5.600x
Amdahl's Law Speed Up 1.661x
Fraction of Accelerated Code 48%
Number of Offloads 20
CPU Threads 1
Target Platform XeLP GT2
Baseline Platform Intel(R) Xeon(R) Gold 6128 CPU
Offload Bounded By
Compute 11%
L3 Cache BW 5%
GTI BW 0%
LLC BW 0%
Memory BW 24%
Atomics 0%
Latencies 0%
Data Transfer 0%
Launch Tax 9%
Dependency 0%
Trip Count 0%
Unknown 0%
Non Offloaded 52%
Modeling Parameters
Save to Remodel
Target Device
XeLP GT2
Reset
Set to Hardware Default
Hardware Parameters
EU Count 96
Frequency 1.35 GHz
GTI Bandwidth 172.8 GB/s
L3 Bandwidth 518.4 GB/s
L3 Size 3.75 MB
Top Offloaded
Loop/Function
Execution Time
Speed-Up
Bounded By
Data Transfer
[loop in LAMMPS_NS::PairLJCutIntel::eval<(int)0, (int)0, (int)1, float, double>ompparallel@196 at pair_lj_cut_intel.cpp:219]
CPU 408.0ms GPU 40.3ms 10.117x Compute 17.7MB
[loop in LAMMPS_NS::Atom::sort at atom.cpp:2086]
CPU 304.1ms GPU 7.1ms 42.653x DRAM BW 18.1MB
[loop in LAMMPS_NS::FixLangevin::post_force_templated<(int)0, (int)0, (int)0, (int)0, (int)0, (int)0> at fix_langevin.cpp:643]
CPU 259.7ms GPU 8.1ms 32.039x DRAM BW 9.33MB
[loop in LAMMPS_NS::BondFENEIntel::eval<(int)0, (int)0, (int)1, float, double>ompparallel@132 at bond_fene_intel.cpp:162]
CPU 123.9ms GPU 4.6ms 27.103x L3 BW 9.21MB
[loop in LAMMPS_NS::FixNVEIntel::initial_integrate at fix_nve_intel.cpp:78]
CPU 112.2ms GPU 4.7ms 23.654x DRAM BW 9.27MB
Top Non-Offloaded
Loop/Function
Execution Time
Bounded By
Why Not Offloaded
Data Transfer
[loop in fi_getinfo]
CPU 1.3s GPU 1.3s Launch Tax, Latencies, DRAM BW
Not profitable: Launch Tax, Latencies, DRAM Bandwidth Time is greater than other execution time components on a Target Device
258kB
[loop in LAMMPS_NS::Neighbor::build at neighbor.cpp:2328]
CPU 255.8ms GPU 15.44s Trip Counts, Latencies
Not profitable: Trip Counts, Latencies
13.3MB
[loop in LAMMPS_NS::Replicate::command at replicate.cpp:655]
CPU 20.0ms GPU 97.4ms Trip Counts, Latencies
Not profitable: Trip Counts, Latencies
11.9MB
[loop in LAMMPS_NS::Neighbor::build at neighbor.cpp:2328]
CPU 20.0ms GPU 578.5ms Trip Counts, Latencies
Not profitable: Trip Counts, Latencies
1.04MB
22°C
Sunny
13:36
15/06/2022



https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html

TWO TYPES OF COMPILER

- **CLASSIC** **[Same as was in Parallel Studio]**

Deprecated and will be removed from product release in the second half of 2023

- icc
- icpc
- ifort

Offloading not supported

LLVM based [Totally new compilers]

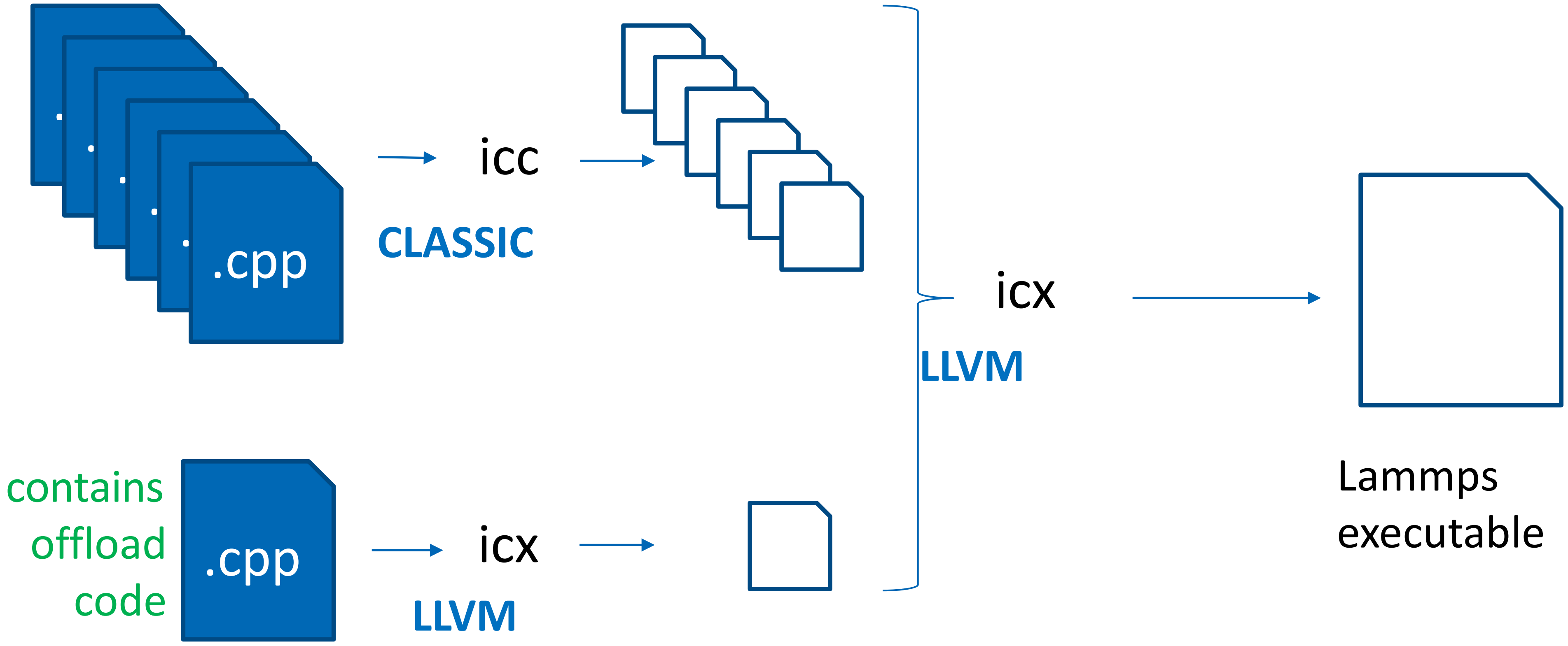
- icx
- dpcpp
- ifx

Offloading supported

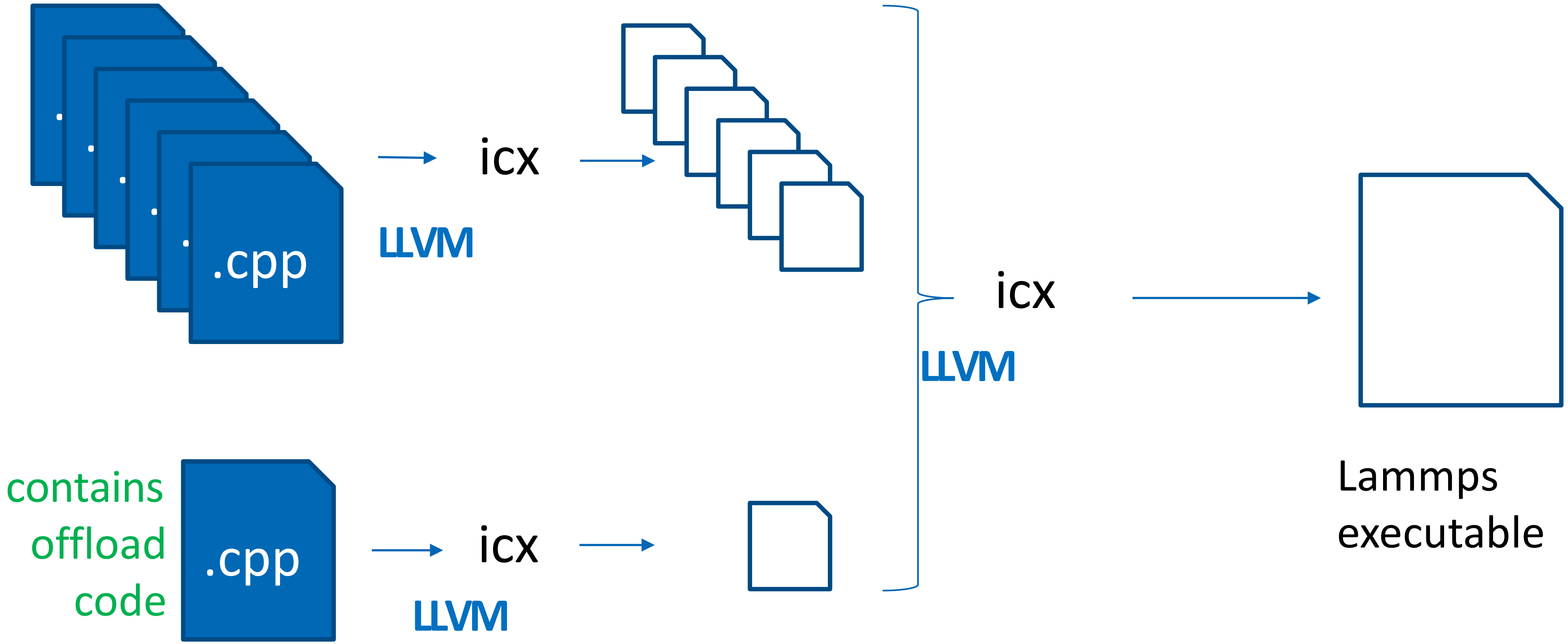
All objects are binary compatible

One approach…

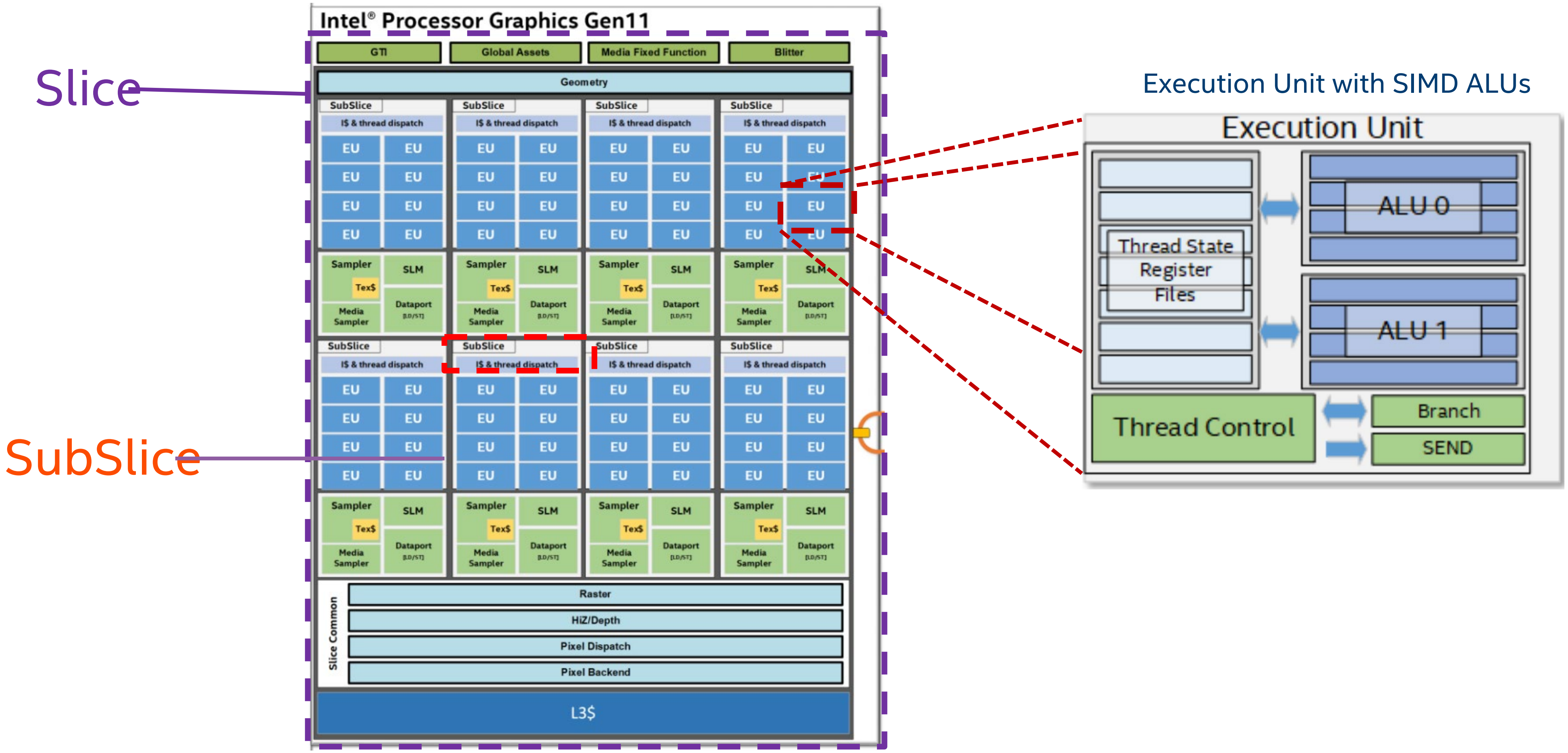
NB: icc deprecated mid 2023



. . . or alternatively



GPU Architecture



GPU Architecture


```
#pragma omp target                         parallel for
```

Sends Code to target
but only on one sub-slice

OpenMP GPU Offload and OpenMP Constructs

- OpenMP GPU offload support all “normal” OpenMP constructs
  - E.g. parallel, for/do, barrier, sections, tasks, etc.
  - Not every construct will be useful

- Full threading model outside of a single GPU subslice **not** supported
  - No synchronization among subslices
  - No coherence and memory fence between among subslice L1 caches

GPU Architecture

`#pragma omp target teams` `parallel for`

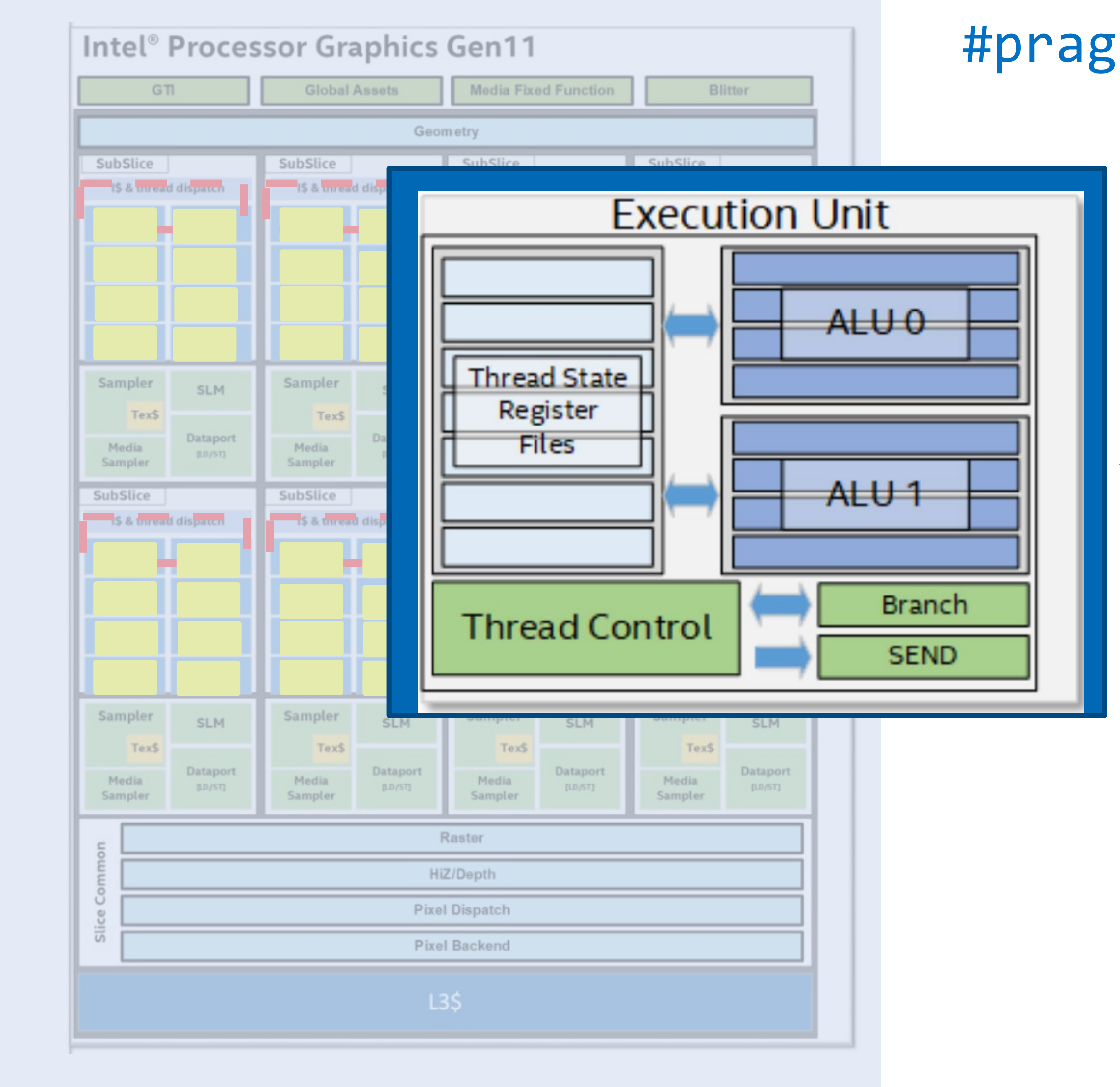
Share code across subslices but only on one EU

GPU Architecture

`#pragma omp target teams distribute parallel for`

Distributes loop iterations across all EUs

GPU Architecture




```
#pragma omp target teams distribute simd parallel
```

Uses SIMD wide registers
in execution units

Experiment in Lammps -Target

```
#if 1
      #pragma omp target map(to:ilist[iifrom:iito]) \
                         map(to:x[x_min:x_max]) \
                         map(tofrom:f[f_min:f_max]) \
                         map(from:ev_global[0:7]) \
                         map (tofrom:lj1,lj2,lj3,lj4,offset)
      #pragma omp teams distribute parallel for
 #endif
  // ------------------------ END SBC code -------------------


      for (int ii = iifrom; ii < iito; ii += iip) {
        const int i = ilist[ii];
        int itype, ptr_off;
        const FC_PACKED1_T * _noalias ljc12oi;
        const FC_PACKED2_T * _noalias lj34i;
        if (!ONETYPE) {
          itype = x[i].w;
          ptr_off = itype * ntypes;
```

Experiment in Lammps –Indirect Indexes

Start here | *pair_lj_cut_intel.cpp

```cpp
// ----------------------- SBC code ------------------
  #if 1
    // find min an max of indirect indexes
    int i_min=0;
    int i_max=0;
    int jlist_min=0;
    int jlist_max=0;
    int j_min =0;
    int j_max=0;
    int x_min=0;
    int x_max=0;
    int f_min=0;
    int f_max=0;

    for (int ii = iifrom; ii < iito; ii += iip) {
      int i=0, itype=0, sbindex=0;

      // get i_min and i_max
      i = ilist[ii];
      i_min=std::min(i,i_min);
      i_max=std::max(i,i_max);


       const int * _noalias const jlist = firstneigh[i];
       int jnum = numneigh[i];

      // we also need the j_min and j_max  as this is used in x[j]
      for (int jj = 0; jj < jnum; jj++) {
        int j=0, jtype=0, sbindex=0;
        if (!ONETYPE) {
          sbindex = jlist[jj] >> SBBITS & 3;
          j = jlist[jj] & NEIGHMASK;
         } else
           j = jlist[jj];

         j_min=std::min(j,j_min);
         i_max=std::max(j,j_max);
       }
     } // ii
     x_min = f_min = std::min(i_min,j_min);
     x_max = f_max = std::max(i_max,j_max);

    #if 0
    printf("x_min: %d, x_max: %d ",x_min,x_max);
    #endif
  #endif

#if 1
    #pragma omp target map(to:ilist[iifrom:iito]) \
```

evCon/lammps-offload-OneAPI/LAB2/6-ICX-PATCHED/pair_lj_cut_intel.cpp   C/C++

```cpp
// we also need the j_min and j_max  as this
for (int jj = 0; jj < jnum; jj++) {
  int j=0, jtype=0, sbindex=0;
  if (!ONETYPE) {
    sbindex = jlist[jj] >> SBBITS & 3;
    j = jlist[jj] & NEIGHMASK;
   } else
     j = jlist[jj];

   j_min=std::min(j,j_min);
   i_max=std::max(j,j_max);
 }
} // ii
x_min = f_min = std::min(i_min,j_min);
x_max = f_max = std::max(i_max,j_max);
```