Direct programming model
Data Parallel C++

Agenda

- What is Data Parallel C++?
- Anatomy of a DPC++ Application
- Memory model
- Compilation and Execution
- Device Selection
- Unified Shared Memory
- Intel® Compilers

What is Data Parallel C++?

The language is:

C++

+

SYCL* khronos.org/sycl/

+

Additional Features tinyurl.com/dpcpp-ext

Khronos® is a registered trademark and SYCL is a trademark of the Khronos Group, Inc.

What is Data Parallel C++?

The implementation is:

Clang

+

LLVM https://github.com/intel/llvm

+

Runtime https://github.com/intel/compute-runtime

Code samples:

tinyurl.com/dpcpp-tests

tinyurl.com/oneapi-samples

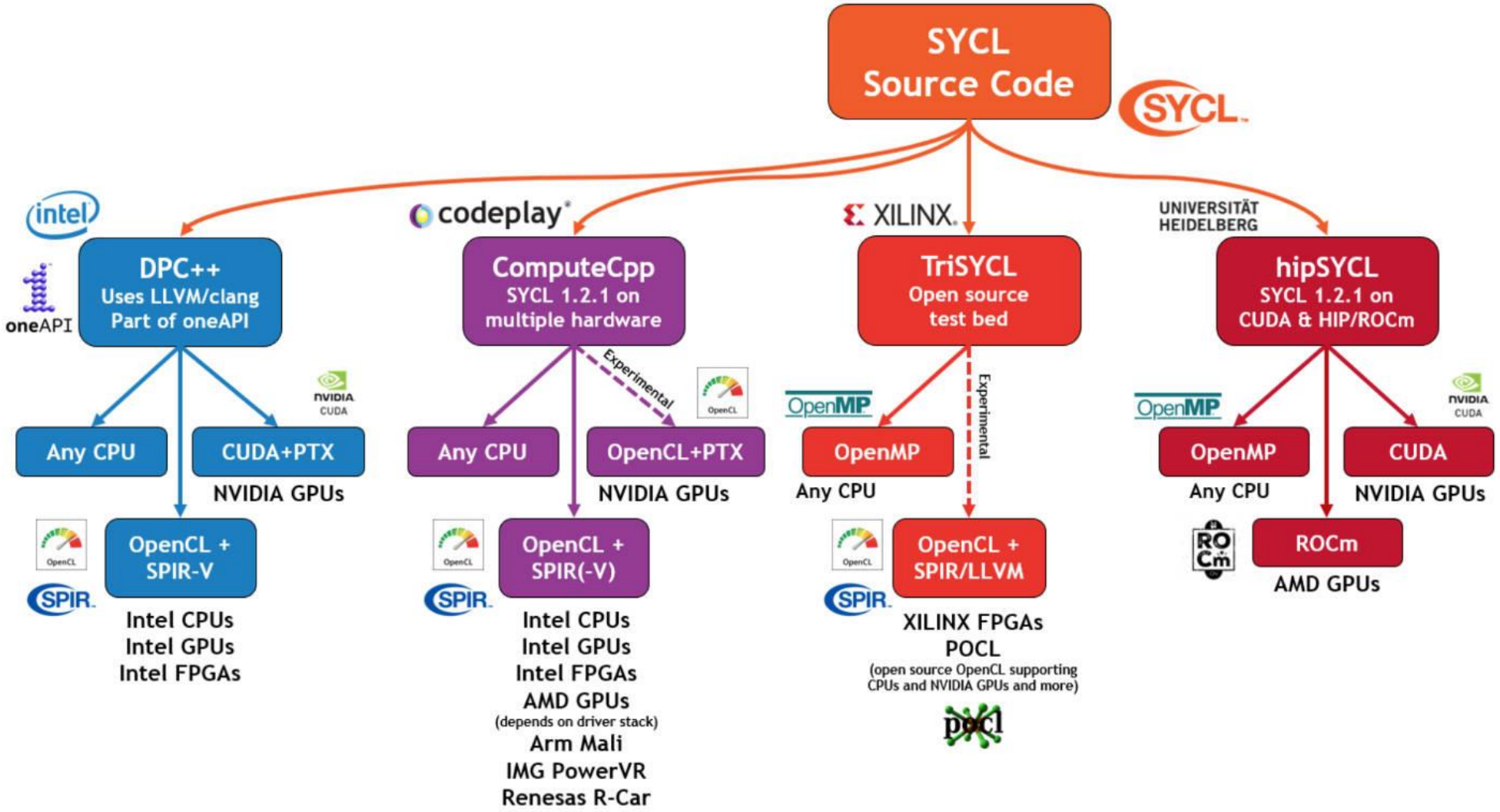
DPC++ Extensions

tinyurl.com/dpcpp-ext

tinyurl.com/sycl2020

Extension	Purpose	SYCL 2020
USM (Unified Shared Memory)	Pointer-based programming	✓
Sub-groups	Cross-lane operations	✓
Reductions	Efficient parallel primitives	✓
Work-group collectives	Efficient parallel primitives	✓
Pipes	Spatial data flow support	
Argument restrict	Optimization	
Optional lambda name for kernels	Simplification	✓
In-order queues	Simplification	✓
Class template argument deduction and simplification	Simplification	✓

SYCL implementation ecosystem



Anatomy of a DPC++ Application

```
#include <CL/sycl.hpp>
using namespace sycl;

int main() {
std::vector<float> A(1024), B(1024), C(1024);
// some data initialization
    {
      buffer bufA {A}, bufB {B}, bufC {C};
      queue q;
      q.submit([&](handler &h) {
          auto A = bufA.get_access(h, read_only);
          auto B = bufB.get_access(h, read_only);
          auto C = bufC.get_access(h, write_only);
          h.parallel_for(1024, [=](auto i){
              C[i] = A[i] + B[i];
          });
      });
    }
for (int i = 0; i < 1024; i++)
       std::cout << "C[" << i << "] = " << C[i] << std::endl;

}
```

Host code

Accelerator device code

Host code

Memory model

- SYCL abstractions for managing memory
  - Buffers
  - Images

SYCL* 1.2.1

- Powerful and elegantly expresses data dependences, but need to replace all pointers and arrays with buffers

- Unified Shared Memory (USM)

SYCL 2020

Khronos® is a registered trademark and SYCL is a trademark of the Khronos Group, Inc.

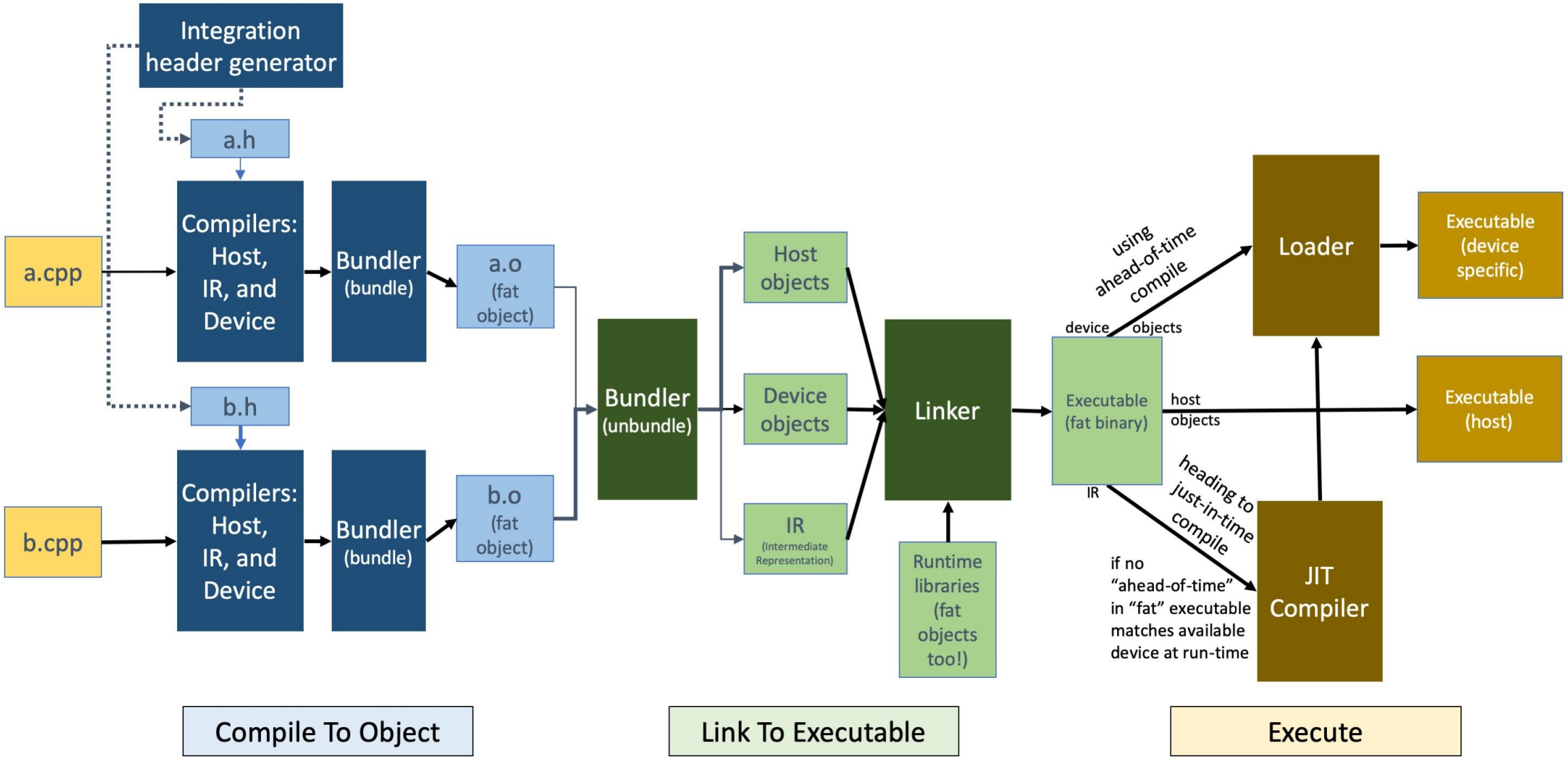
Buffers

```
std::vector<float> A(1024), B(1024), C(1024);
      {
        buffer bufA {A}, bufB {B}, bufC {C};
        queue q;
        q.submit([&](handler &h) {
            auto A = bufA.get_access(h, read_only);
            auto B = bufB.get_access(h, read_only);
            auto C = bufC.get_access(h, write_only);
            h.parallel_for(1024, [=](auto i){
                C[i] = A[i] + B[i];
            });
        });
      }
 for (int i = 0; i < 1024; i++)

        std::cout << "C[" << i << "] = " << C[i] << std::endl;

 }
```

SYCL 1.2.1 vs SYCL 2020

```
std::vector<float> A(1024), B(1024), C(1024);
{
  buffer<float> bufA {A.data(), A.size()};
  buffer<float> bufB{B.data(), B.size()};
  buffer<float> bufC {C.data(), C.size()};

  queue q;
  q.submit([&](handler &h) {
    auto A = bufA.get_access<access::mode::read>(h);
    auto B = bufB.get_access<access::mode::read>(h);
    auto C = bufC.get_access<access::mode::write>(h);
    h.parallel_for <class vector_add>(range<1>{1024}, [=](id<1> i){
                C[i] = A[i] + B[i];
            });
        });
      }
 for (int i = 0; i < 1024; i++)
        std::cout << "C[" << i << "] = " << C[i] << std::endl;
 }
```

DPC++ Compilation and Execution

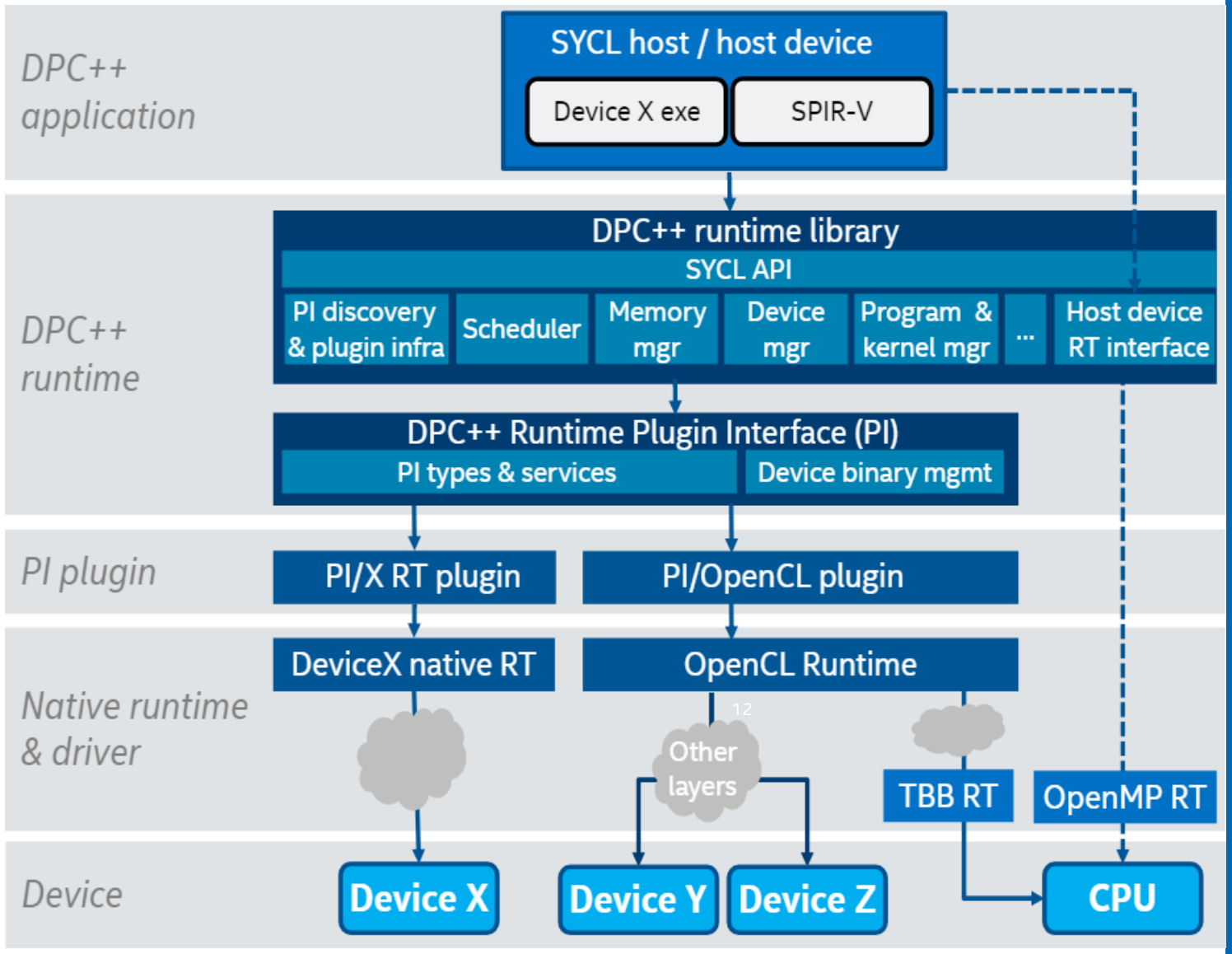


https://software.intel.com/en-us/oneapi-dpcpp-compiler-dev-guide-and-reference-ahead-of-time-compilation

DPC++ Execution Flow

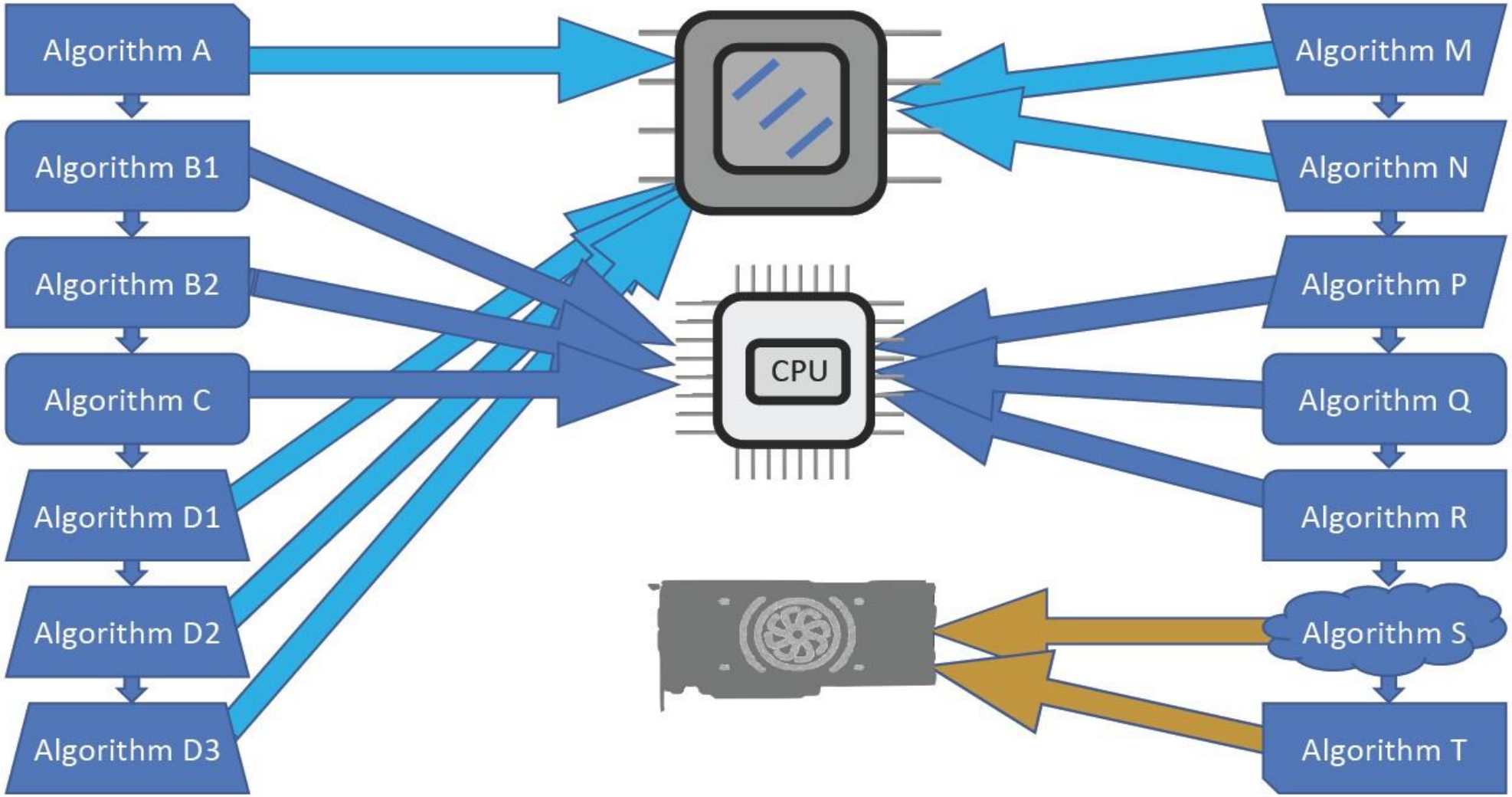
github.com/intel/llvm/tree/sycl/sycl/plugins

Controlled via SYCL_BE env var:
PI_OPENCL, PI_CUDA, PI_LEVEL0



More info: tinyurl.com/dpcpp-pi

DPC++ Device Selection



USM - Explicit Data Movement

```
queue q;
int hostArray[42];
int *deviceArray = (int*) malloc_device(42 * sizeof(int), q);

for (int i = 0; i < 42; i++) hostArray[i] = 42;
// copy hostArray to deviceArray
q.memcpy(deviceArray, &hostArray[0], 42 * sizeof(int));
q.wait();
q.submit([&](handler& h){
  h.parallel_for(42, [=](auto ID) {
    deviceArray[ID]++;
  });
});
q.wait();
// copy deviceArray back to hostArray
q.memcpy(&hostArray[0], deviceArray, 42 * sizeof(int));
q.wait();
free(deviceArray, q);
```

USM – Implicit Data Movement

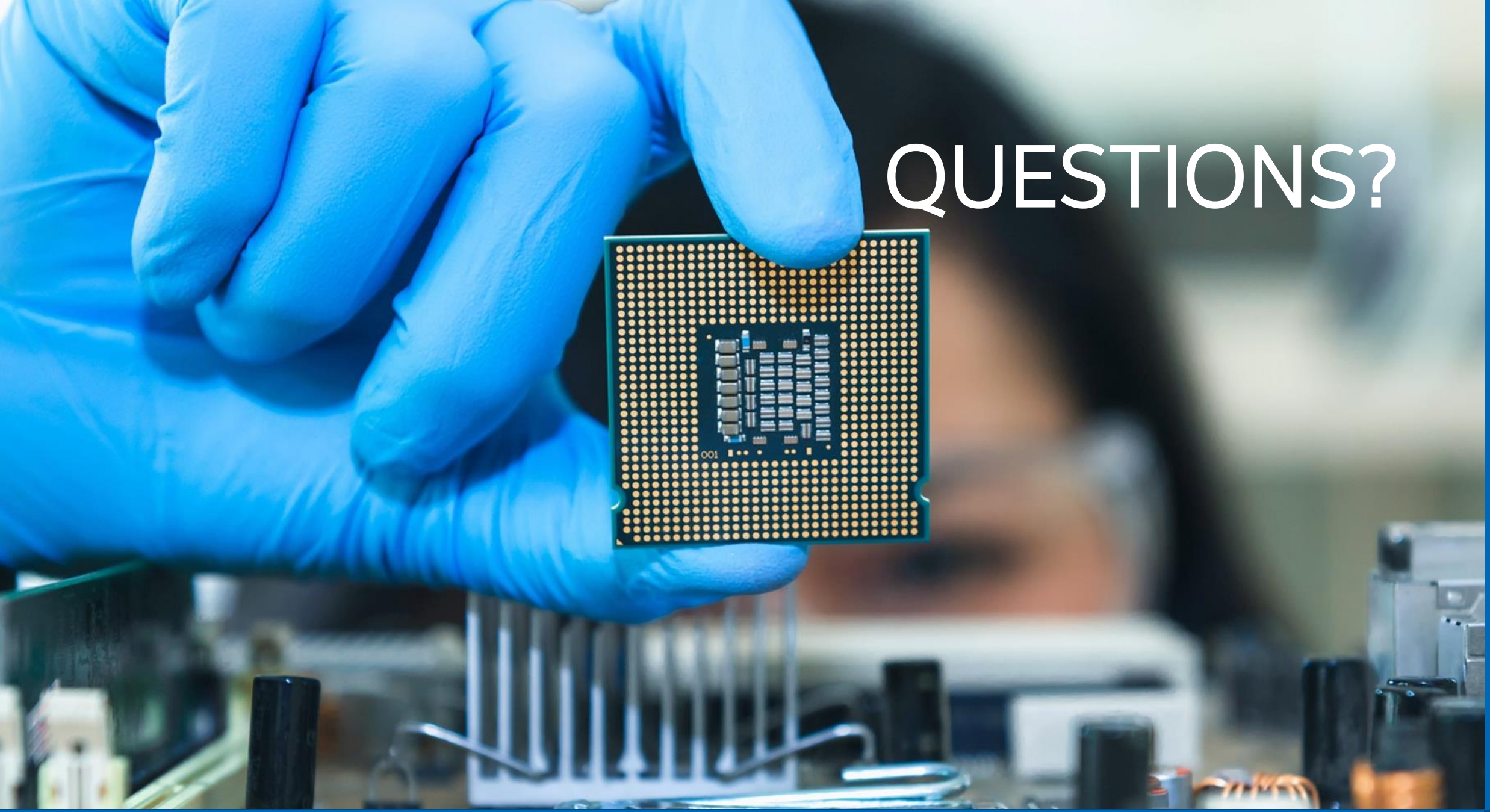
```
queue q;
int *hostArray = (int*) malloc_host(42 * sizeof(int), q);
int *sharedArray = (int*) malloc_shared(42 * sizeof(int), q);

for (int i = 0; i < 42; i++) hostArray[i] = 1234;
q.submit([&](handler& h){
  h.parallel_for(42, [=](auto ID) {
    // access sharedArray and hostArray on device
    sharedArray[ID] = hostArray[ID] + 1;
  });
});
q.wait();
for (int i = 0; i < 42; i++) hostArray[i] = sharedArray[i];
free(sharedArray, q);
free(hostArray, q);
```

Intel® Compilers

Intel Compiler	Target	OpenMP Support	OpenMP Offload Support	Included in oneAPI Toolkit
Intel® C++ Compiler, IL0 (icc)	CPU	Yes	No	HPC
Intel® oneAPI DPC++/C++ Compiler (dpcpp)	CPU, GPU, FPGA*	No	No	Base
Intel® oneAPI DPC++/C++ Compiler (ICX)	CPU GPU*	Yes	Yes	Base and HPC
Intel® Fortran Compiler, IL0 (ifort)	CPU	Yes	No	HPC
Intel® Fortran Compiler (ifx)	CPU, GPU*	Yes	Yes	HPC

Cross Compiler Binary Compatible and Linkable!

*Intel® Platforms
**PSXE 2020 Production+oneAPI HPC Toolkit(BETA)
*** IFX will remain in BETA in 2021


QUESTIONS?

Notices & Disclaimers

- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.
- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Copyright © 2020, Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Core, VTune, and OpenVINO are trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries. Khronos® is a registered trademark and SYCL is a trademark of the Khronos Group, Inc.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.
Notice revision #20110804


intel