alpaka Parallel Programming – Online Tutorial
Lecture 00 – Getting Started with alpaka
Lesson 03: Single-Source Applications for Many Ecosystems
Lesson 03: Single-Source Applications for Many Ecosystems

Single-source programming style

```cpp
impl.cpp

void myKernel(...) {
    // Implementation
}

int hostFunc() {
    myKernel(args);
}
```

Split-source programming style

```cpp
impl_device.ext

void myKernel(...) {
    // Implementation
}

impl_host.cpp

int hostFunc() {
    myKernel(args);
}
```
Lesson 03: Single-Source Applications for Many Ecosystems

**Single-source**

- Standard C++ programming style
- Kernel is embedded in application code
- Allows for easy reuse of common functionality
- Lower API complexity

**Split-source**

- Different from standard C++ style
- Kernel and application code are separate → No shared information
- Possibly leads to code duplication
- API requires additional mechanisms to load and execute kernel
Portability across ecosystems

- At compilation time: alpaka kernels are transformed into native kernels
  - NVIDIA: alpaka kernel → native CUDA kernel
  - AMD: alpaka kernel → native HIP kernel

- alpaka is fully compatible with the vendors’ ecosystems
  - Important for NVIDIA GPUs: can't debug / profile OpenCL or SYCL kernels with CUDA tools

- No extra cost!

- Does not hurt portability!
Lesson 03: Single-Source Applications for Many Ecosystems

[Diagram showing a system with various components and data flow.]
Lesson 03: Single-Source Applications for Many Ecosystems