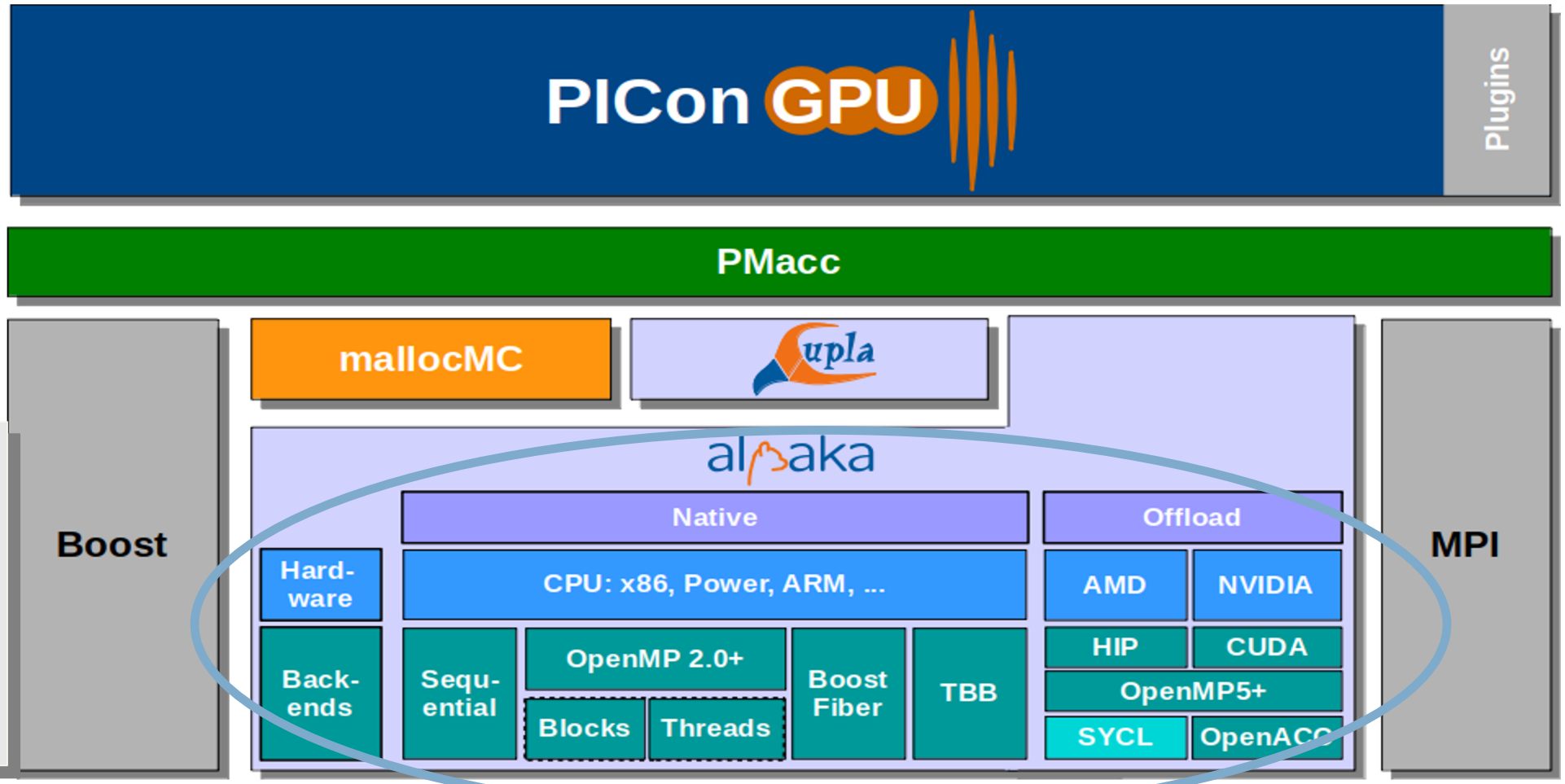
OpenMP Target Constructs Behind C++ Templates

Experiences of the PIConGPU CAAR application using Alpaka

Computational Science Department · Guido Juckeland · g.Juckeland@hzdr.de · www.hzdr.de/fwcc

PIConGPU Full Software Stack

```
    template< typename T_Acc
>
    ALPAKA_FN_ACC void
operator()(
        T_Acc const & acc,
        // ...
    ) const
    {
        // ...
    }
```



Huebl, Axel, et al. (2018) Zero Overhead Modern C++ for Mapping to Any Programming Model.
Software Stack updated by René Widera (2020)

What is alpaka?

- Header-only C++14 abstraction library for accelerator development
- Accelerator type passed to device kernels as backend handle

```
template<typename TAcc>
void kernel(const TAcc& acc, ...);
```

⇒ no conditional compilation required for backend selection

- API and feature set modelled after CUDA

host	devices, queues, events, memory management, ...
device	atomics, block-shared memory, block-sync, ...
lib	math, random, ...

- supported backends include:
  sequential, OpenMP, TBB, CUDA, HIP, ...

OpenMP target

```
1  // TaskKernel_Omp5::operator() (...) {
2  // copy members to local scope, e.g.:
3  auto args = m_args;
4  omp_set_num_threads( workdiv.threads );
5  #pragma omp target
6  {
7  # pragma omp teams distribute
8    for ( int b = 0; b < workDiv.blocks; ++b )
9    {
10     // OpenMP backend handle:
11     AccOmp5 ctx ( workdiv, b );
12 #    pragma omp parallel
13
14     {
15
16       apply([&ctx](auto ...args){
17           functor ( ctx, args... );
18         }, margs);
19 } } }
```

Issues in Standards

- types containing `static constexpr` data members were not mappable (OpenMP target (< 5.0))
  - probably result of a ban on static with no regard to const in C++

- mapping of `constexpr` variables with static lifetime (compile-time constants) not implicit (OpenMP target)
  - compiler knows which constants are used and there is no abiguity about sequence of copy ⇒ should be implicit

- `std::tuple` implementations are not required to be trivially copyable if all component types are (C++)

  ⇒ no `std::tuple` is formally mappable

Issues with Compilers

Main complication turned out to be a lack of tested compiler support:

- OpenMP 5.0 not fully supported anywhere. E.g:
  GCC types with static constexpr not mappable (very strict interpretation of OpenMP 4.5)
  ⇒ porting PIConGPU impossible
- Internal Compiler Errors (ICE) happen when directives meet C++
- Invalid use or not-implemented features can trigger ICE instead of compiler error
- Runtime errors, like incorrect data sharing, atomics not doing what they should

What runs now?

alpaka Test suite

- Suite of tests also used in alpaka's CI
- Battery of test cases for each aspect of a backend: kernels, memory, atomics, ...
- Using Catch2 ⇒ more TMP, harder for compilers to succeed.

	Clang Main		ROC Clang	GCC 11
	x86	hsa	hsa	x86
compile	✓	most	slow, linker hangs	most
run	✓	memory error		✗

- PIConGPU compiles and runs using Clang Main on x86