

alpaka Parallel Programming – Online Tutorial

Lecture 00 – Getting Started with alpaka

Lesson 03: Single-Source Applications for Many Ecosystems



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science



Lesson 03: Single-Source Applications for Many Ecosystems

Single-source programming style

```
impl.cpp

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

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

Split-source programming style

```
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

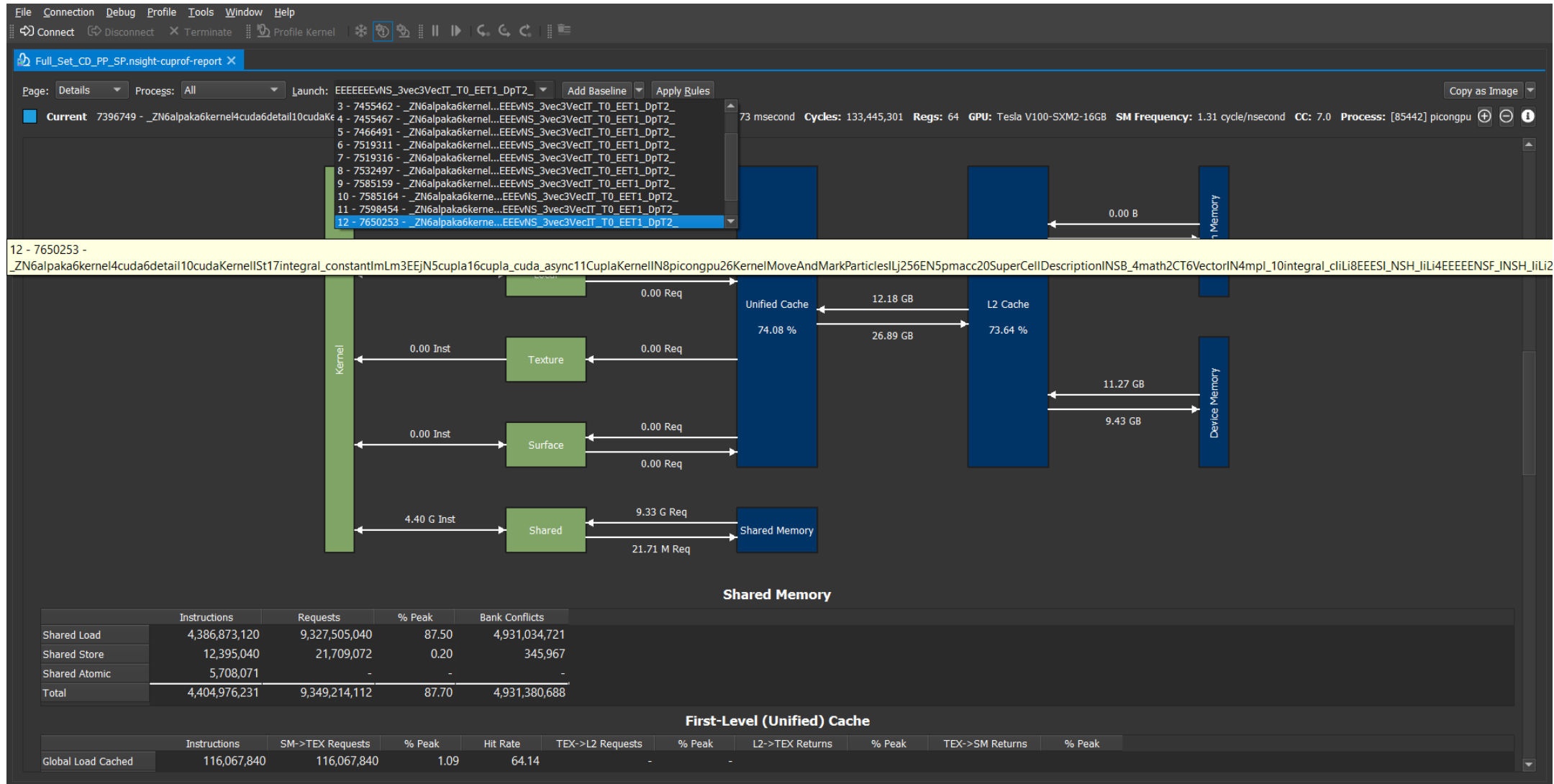
Lesson 03: Single-Source Applications for Many Ecosystems



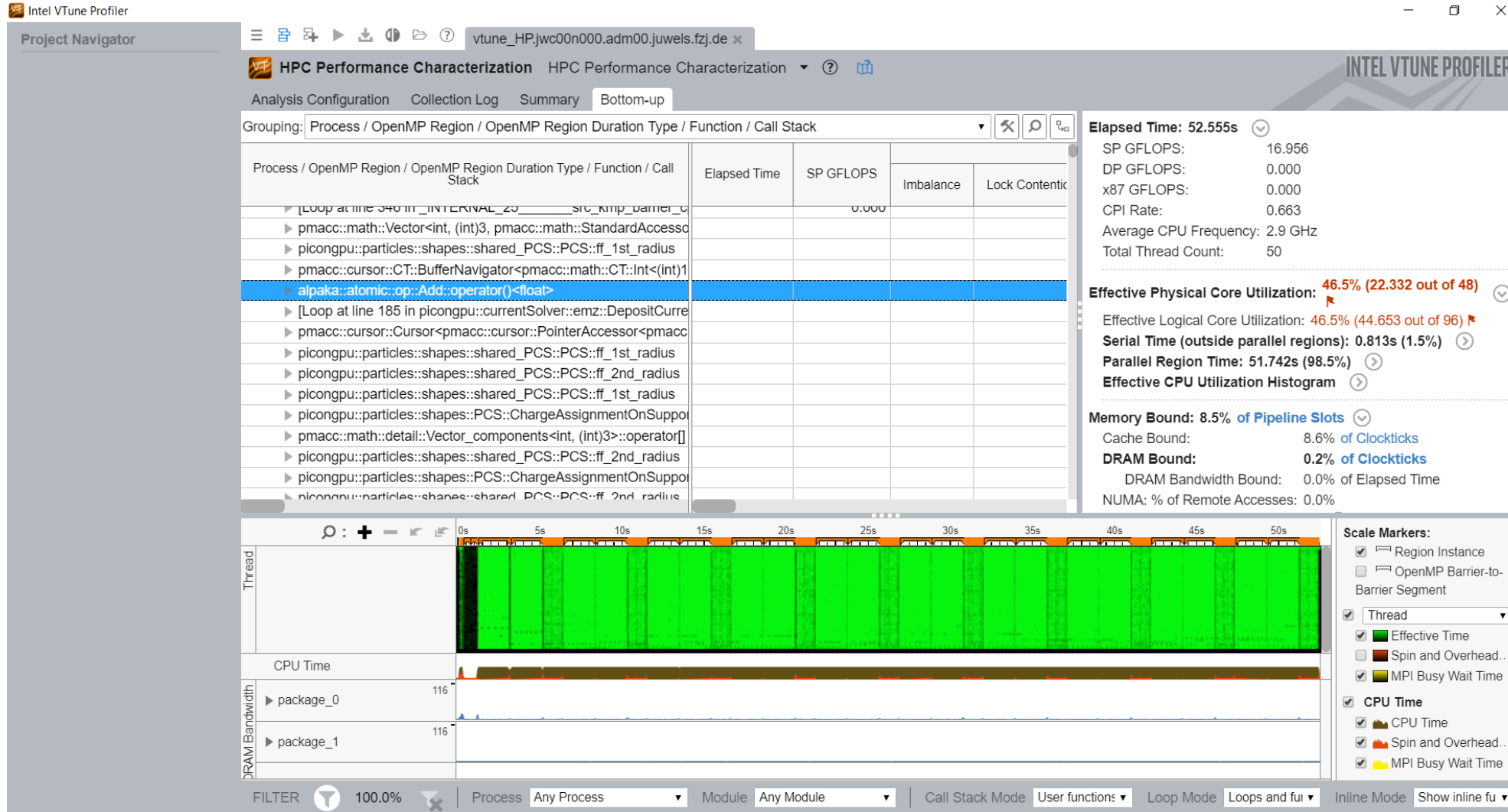
Portability across ecosystems

- 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



Lesson 03: Single-Source Applications for Many Ecosystems



The screenshot displays the Intel VTune Profiler interface for HPC Performance Characterization. The main window shows a call stack table with the following columns: Process / OpenMP Region / OpenMP Region Duration Type / Function / Call Stack, Elapsed Time, SP GFLOPS, Imbalance, and Lock Contention. The selected function is `alpaka::atomic::op::Add::operator()<float>`.

Process / OpenMP Region / OpenMP Region Duration Type / Function / Call Stack	Elapsed Time	SP GFLOPS	Imbalance	Lock Contention
[Loop at line 340 in INTERNAL_ZO...]		0.000		
pmacc::math::Vector<int, (int)3, pmacc::math::StandardAccesso				
picongpu::particles::shapes::shared_PCS::PCS::ff_1st_radius				
pmacc::cursor::CT::BufferNavigator<pmacc::math::CT::Int<(int)1				
alpaka::atomic::op::Add::operator()<float>				
[Loop at line 185 in picongpu::currentSolver::emz::DepositCurre				
pmacc::cursor::Cursor<pmacc::cursor::PointerAccessor<pmacc				
picongpu::particles::shapes::shared_PCS::PCS::ff_1st_radius				
picongpu::particles::shapes::shared_PCS::PCS::ff_2nd_radius				
picongpu::particles::shapes::shared_PCS::PCS::ff_1st_radius				
picongpu::particles::shapes::PCS::ChargeAssignmentOnSuppo				
pmacc::math::detail::Vector_components<int, (int)3>::operator[]				
picongpu::particles::shapes::shared_PCS::PCS::ff_2nd_radius				
picongpu::particles::shapes::PCS::ChargeAssignmentOnSuppo				
picongpu::particles::shapes::shared_PCS::PCS::ff_2nd_radius				

Summary Metrics:

- Elapsed Time: 52.555s
- SP GFLOPS: 16.956
- DP GFLOPS: 0.000
- x87 GFLOPS: 0.000
- CPI Rate: 0.663
- Average CPU Frequency: 2.9 GHz
- Total Thread Count: 50
- Effective Physical Core Utilization: 46.5% (22.332 out of 48)
- Effective Logical Core Utilization: 46.5% (44.653 out of 96)
- Serial Time (outside parallel regions): 0.813s (1.5%)
- Parallel Region Time: 51.742s (98.5%)
- Effective CPU Utilization Histogram
- Memory Bound: 8.5% of Pipeline Slots
- Cache Bound: 8.6% of Clockticks
- DRAM Bound: 0.2% of Clockticks
- DRAM Bandwidth Bound: 0.0% of Elapsed Time
- NUMA: % of Remote Accesses: 0.0%

Scale Markers:

- Region Instance
- OpenMP Barrier-to-Barrier Segment
- Thread
- Effective Time
- Spin and Overhead...
- MPI Busy Wait Time
- CPU Time
- CPU Time
- Spin and Overhead...
- MPI Busy Wait Time

Filters: FILTER 100.0% | Process: Any Process | Module: Any Module | Call Stack Mode: User functions | Loop Mode: Loops and fu | Inline Mode: Show inline fu



CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science