 DOI: 10.5281/zenodo.1412537

Data Analysis and Simulations in Exascale Computing: Quō vādis?

A. Huebl[1,2], **S. Ehrig**[1,2], **and M. Bussmann**[1]

@ax3l

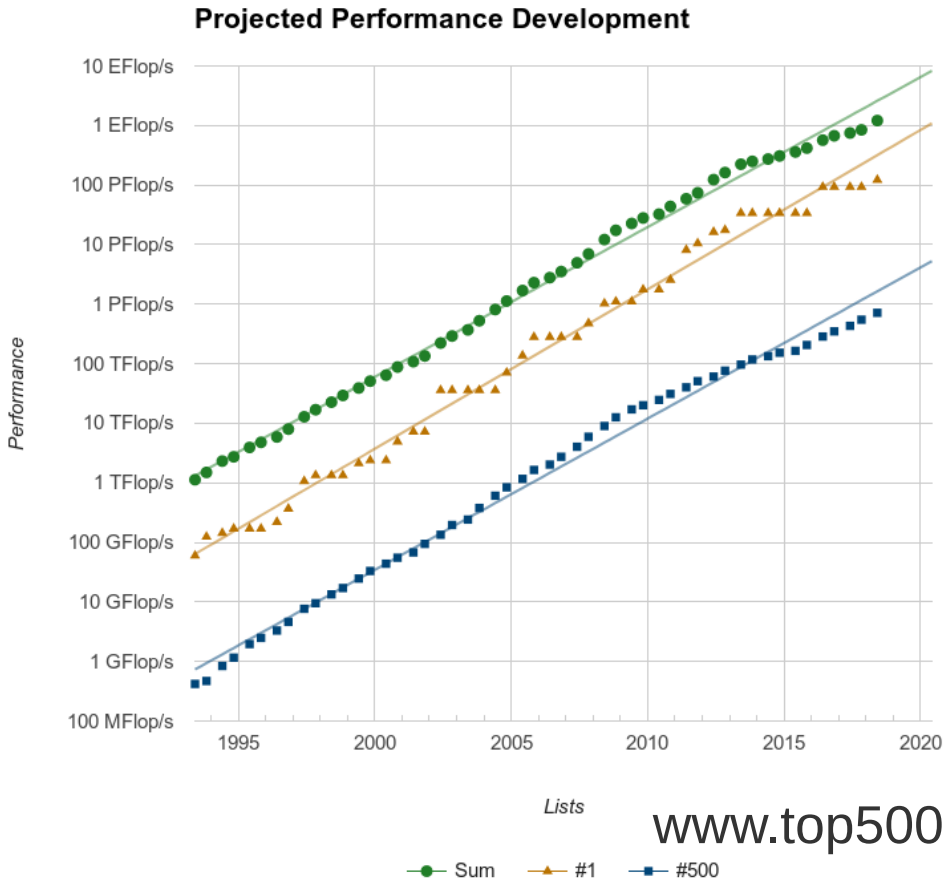
[1] Helmholtz-Zentrum Dresden - Rossendorf
[2] Technische Universität Dresden

ROOT Users' Workshop
Parallelism, Heterogeneity and Distributed Data Processing

Sarajevo, September 10th 2018

Data-Driven Computing
Where is Scientific C++ in Three Years?

- 2020/'21: Challenges towards **Exascale**
  - Maintainability
  - Scalability
  - Data & I/O
    - Reproducibility
  - Productivity

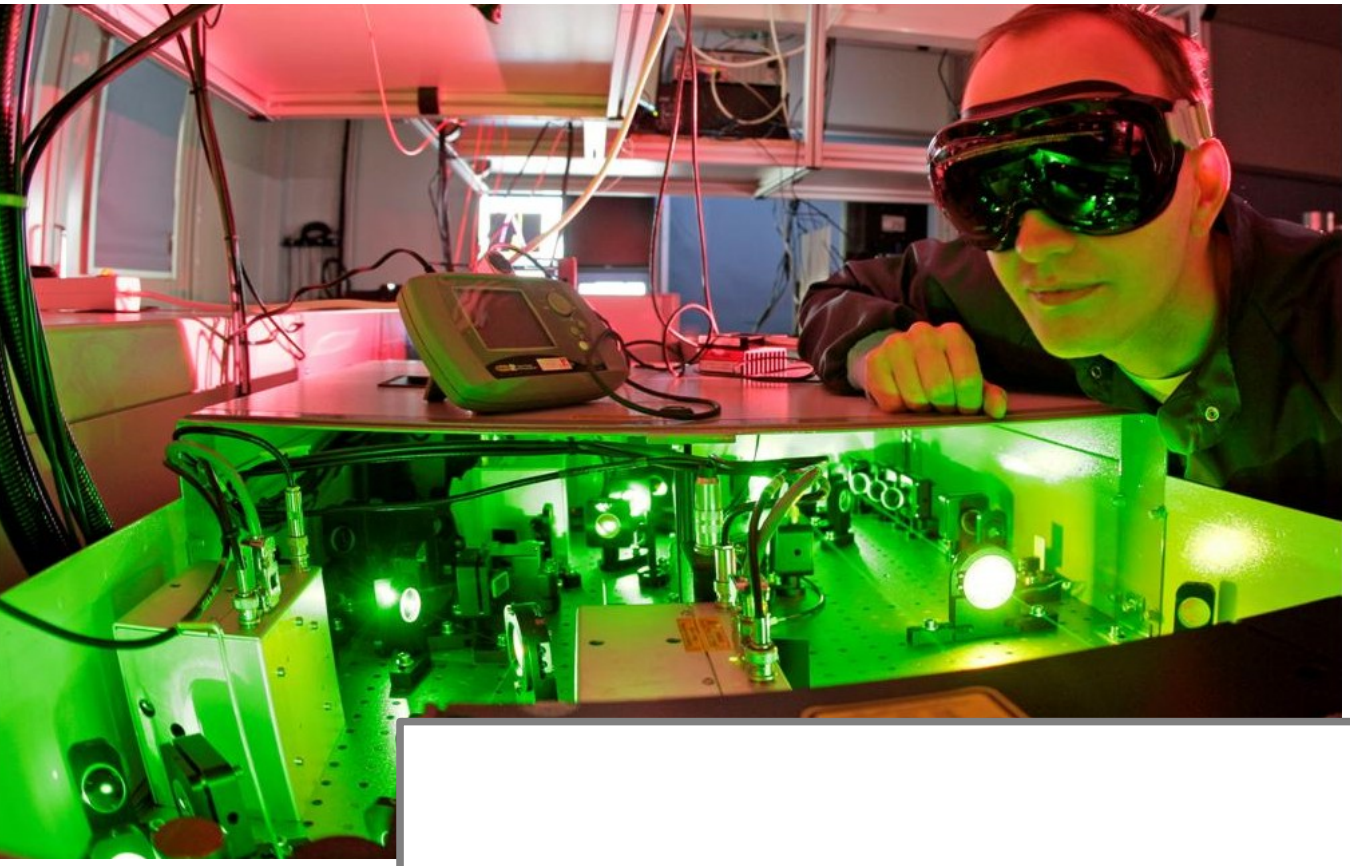


Our Background

Conventional Particle Acceleration

~100 MV / m

(Laser-) Plasma Acceleration



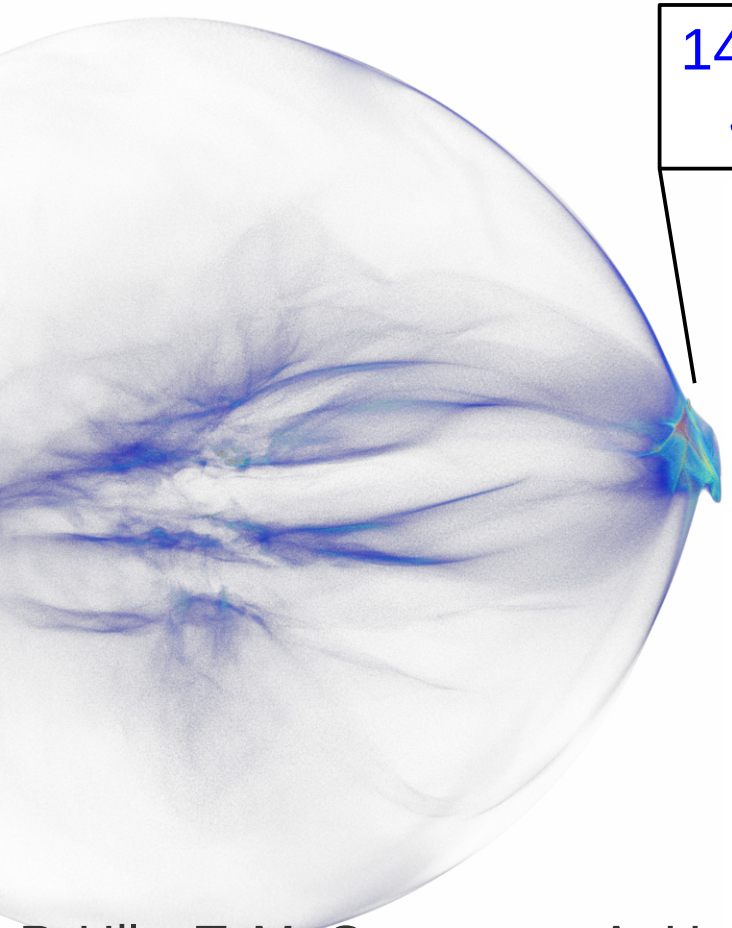
30 – 500 fs
800 – 1053 nm
45 – 200 J

$$I = \frac{PW}{cm^2}$$

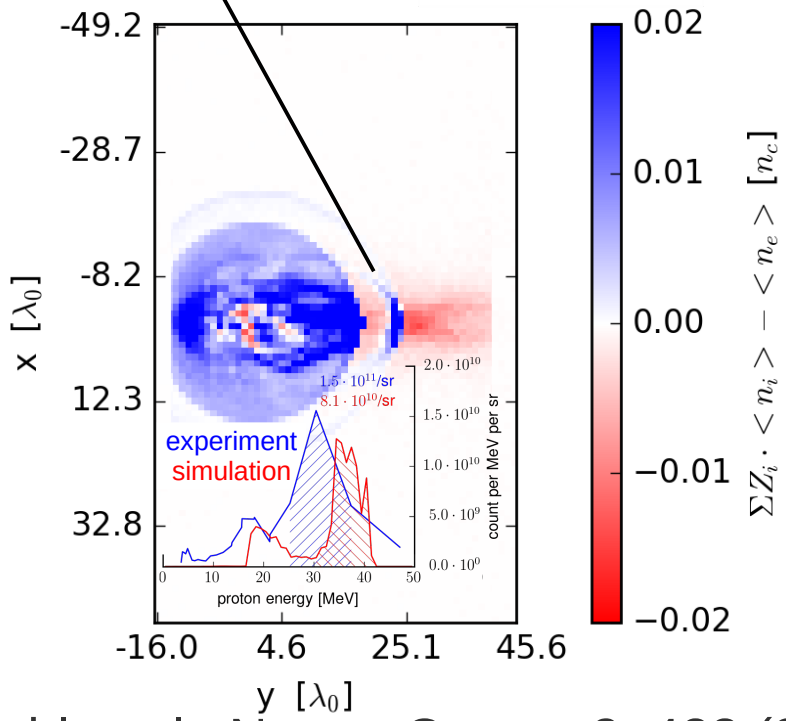
50 nm
-
10 mu

1 000 000 MV / m

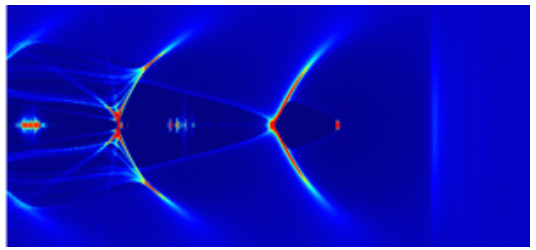
Laser Plasma Acceleration



14% $N_{p,0}$
4 µm



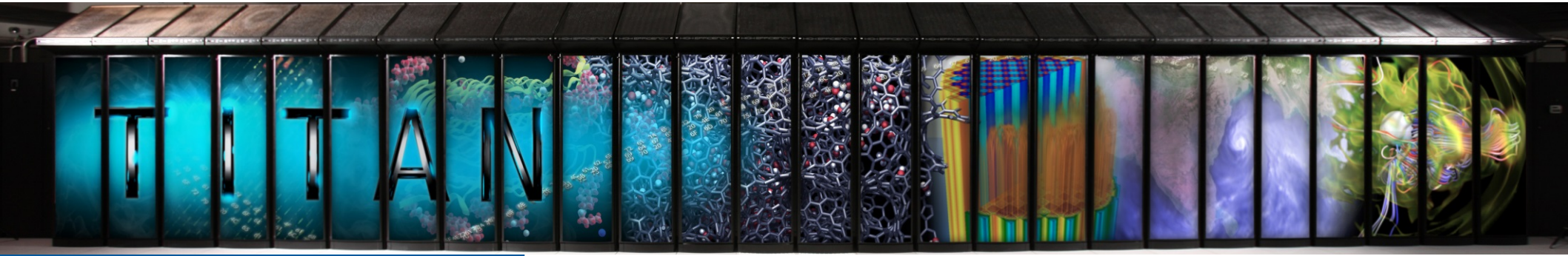
e-: **4.2 GeV**
in **9 cm**



LBNL (2014)

W. P. Leemans et al.,
PRL **113**, 245002

P. Hilz, T. M. Ostermayr, A. Huebl et al., Nature Comm. **9**, 423 (2018)



Particle-in-Cell Simulations

PIConGPU

- Ab initio, electro-magnetic plasmas
- Scaling to the full-size of Titan & Piz Daint
- Gordon Bell finalist 2013

Challenges
And How We Approach Them with C++

Maintainability

Heterogeneous Computing: Performance Portability

```
#ifdef CUDA_ENABLE
    // CUDA Kernel implementation
    // …

#elif OPENMP_ENABLE
    // OpenMP implementation
    // …

#else
    // Sequential CPU implementation
    // …

#endif
```

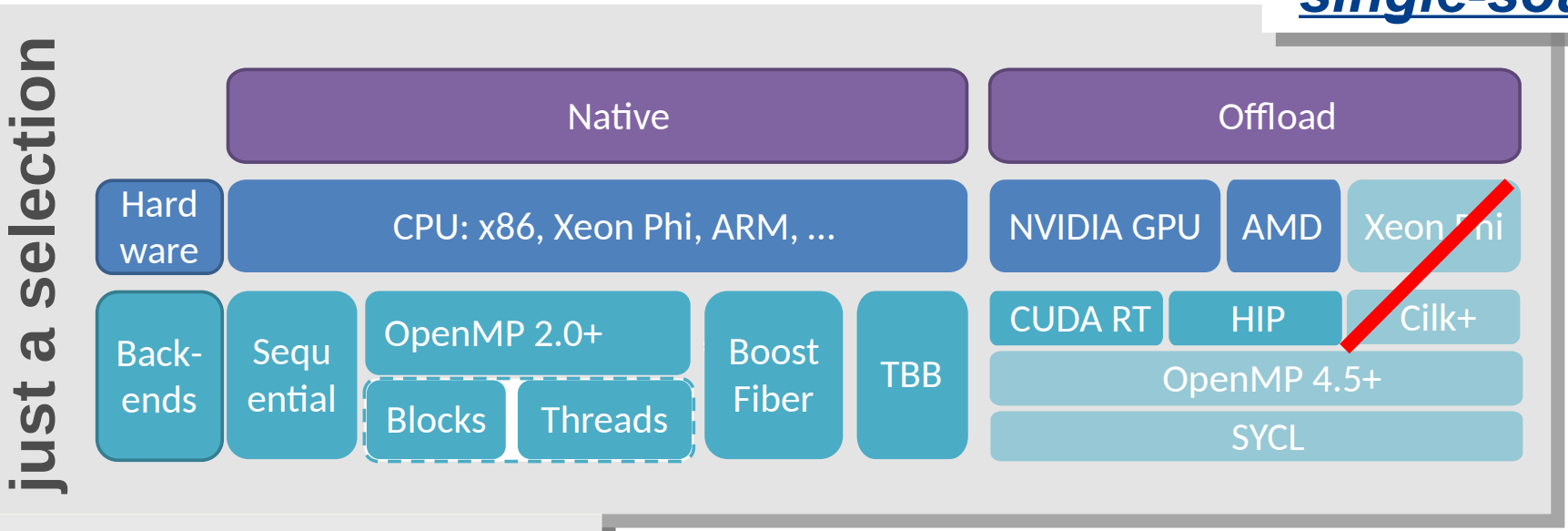
E. Zenker et al., ISC (2016), 10.1007/978-3-319-46079-6_21
A. Matthes et al., ISC (2017), 10.1007/978-3-319-67630-2_36

Maintainability

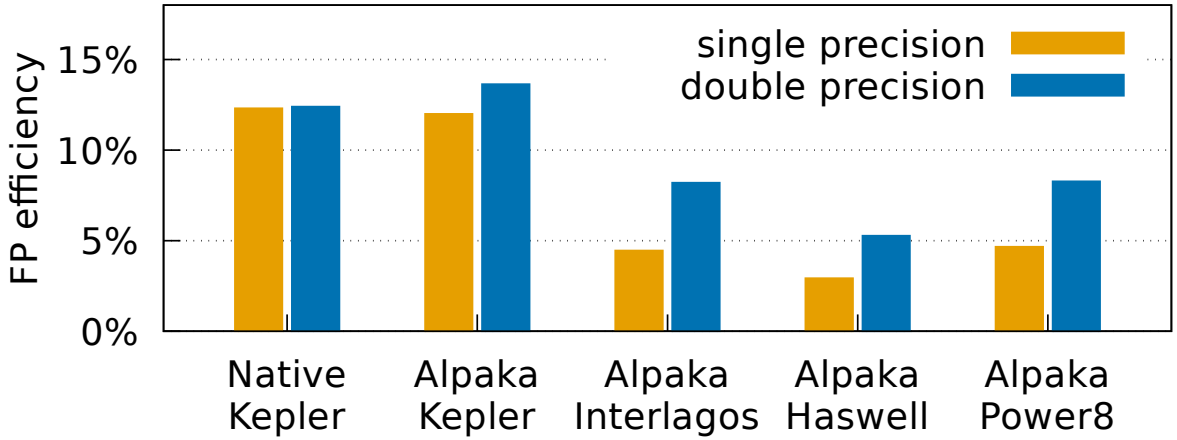
Performance Portability

C++ solutions:
Alpaka & cupla

single-source




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



E. Zenker et al., ISC (2016), 10.1007/978-3-319-46079-6_21
A. Matthes et al., ISC (2017), 10.1007/978-3-319-67630-2_36

EUCALL

NVIDIA GPU CENTER OF EXCELLENCE

DRESDEN concept

HZDR

Scalability

Algorithmic Challenges & Memory Bounds

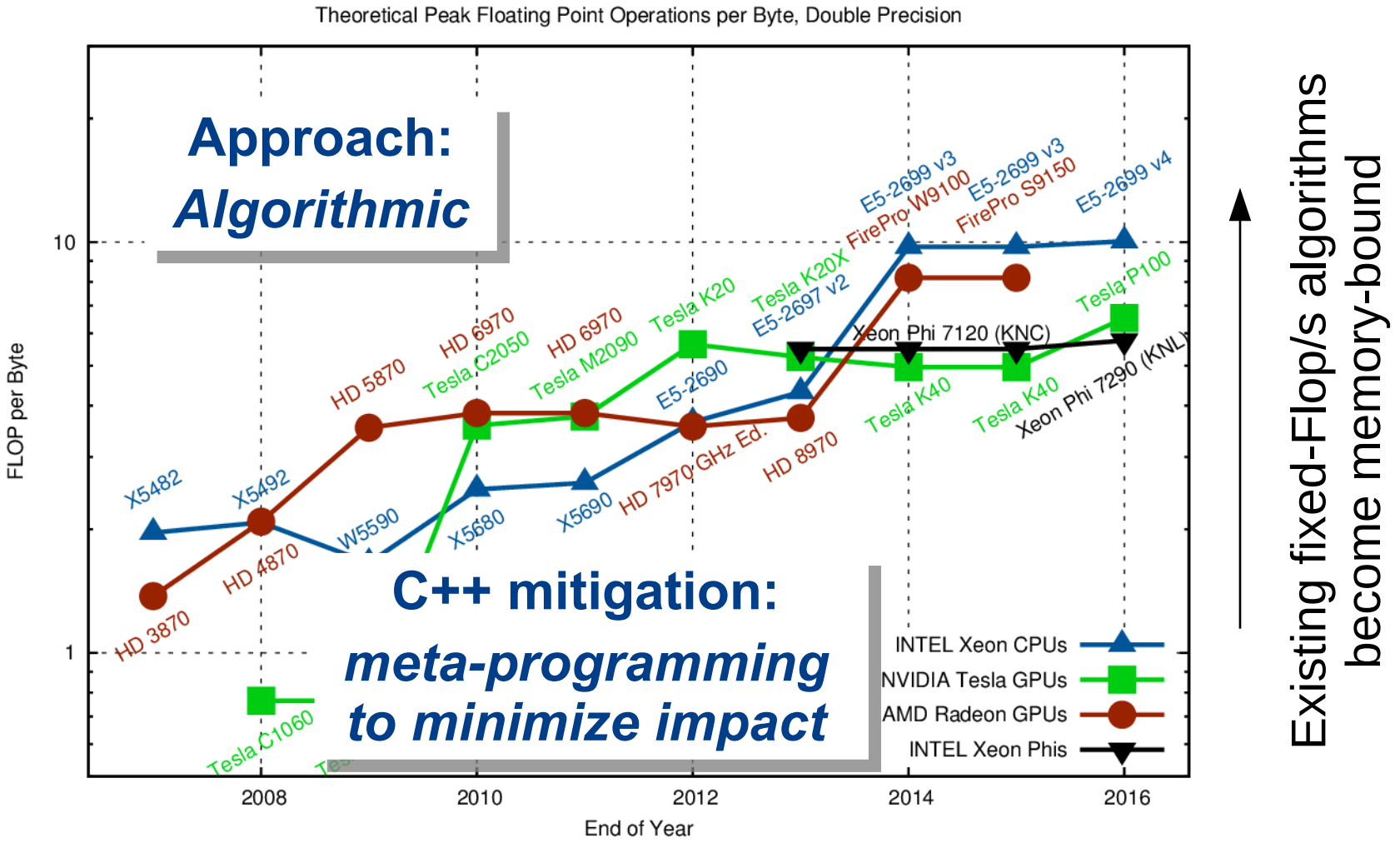
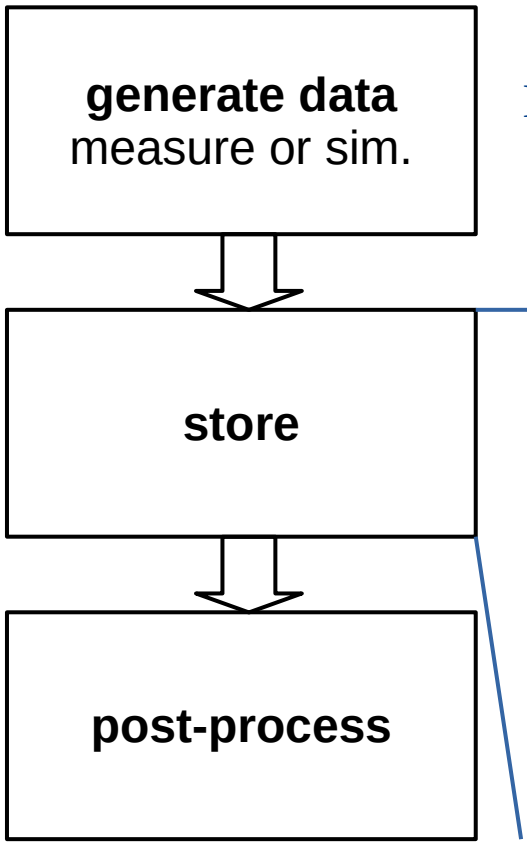


Image: K. Rupp. *CPU, GPU and MIC Hardware Characteristics over Time*, on karlrupp.net
Also: S. Williams. *Roofline: an insightful vis. perf. model for multicore architectures*, 10.1145/1498765.1498785 (2009)

(Cold) Data

Spoiler alert: We cannot afford it

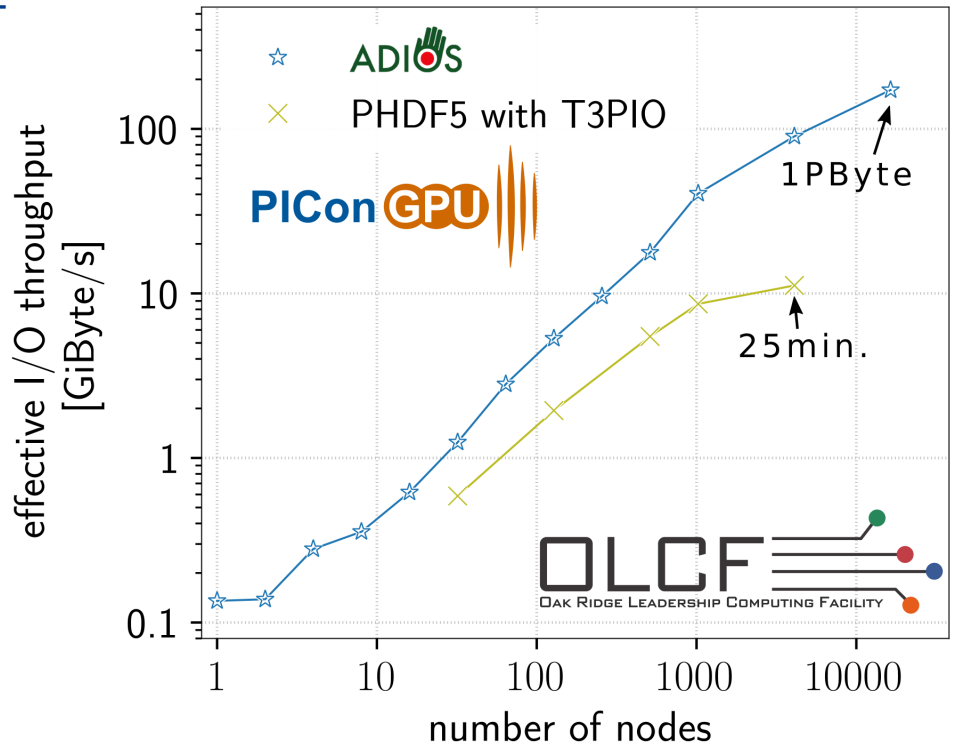
fun fact:
cloud has the same problem



K20x GPU (2013), **bw := 6 GByte · 10 Hz steps**

PCI-Express: **6 Gbyte/s 1 / 10 th bw**

I/O per node:
1 / 10 th · 1 / 200 th bw
– 42 MByte/s (Titan)
– 29 MByte/s (PizDaint)



Summit (ORNL, 2018): ratio **4x "worse" - gap of 10^4**

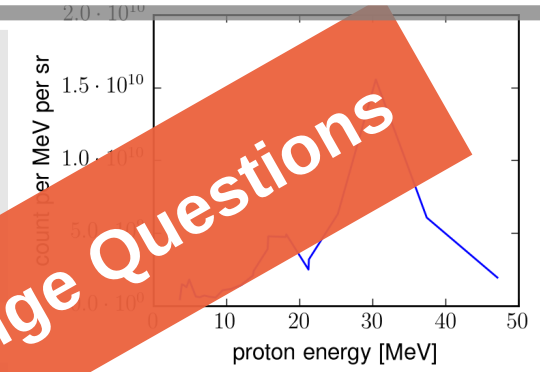
www.olcf.ornl.gov/summit
A. Huebl et al., DRBSD-1 - ISC'17 (2017),
DOI:10.1007/978-3-319-67630-2_2, arXiv:1706.00522

~~(Cold)~~ Hot Data Analysis

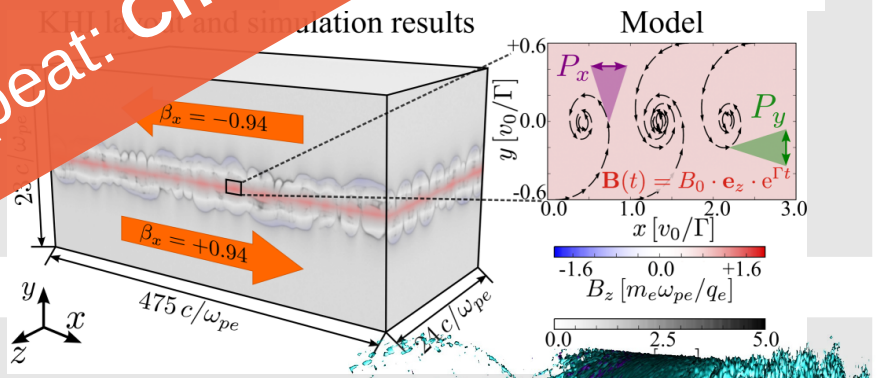
In Situ: Invert Workflows and Repeat

C++ in situ solutions:
openPMD + ADIOS,
"plugins": ISAAC, ...

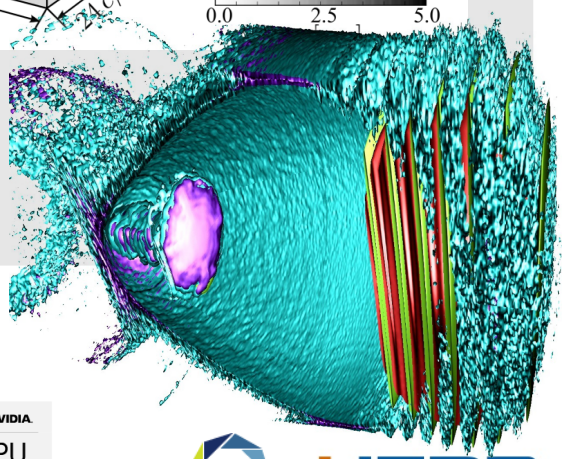
Binning of a **spectrogram**
Creation of a **phase space image**



In situ **radiation diagnostics**



Ray-cast or photo-realistic ray-trace
Lossy data **compression**



Observe, Correlate & Repeat: Change Questions

A. Huebl et al. (2014), DOI:10.1109/TPS.2014.2327392
R. Pausch et al. (2017), DOI:10.1103/PhysRevE.96.013316
A. Matthes, A. Huebl et al., ISC'16 (2016), DOI:10.14529/jsfi160403
A. Huebl et al., ISC'17 (2017), DOI:10.1007/978-3-319-67630-2_2

EUCALL
NVIDIA GPU CENTER OF EXCELLENCE
DRESDEN concept
HZDR

~~(Cold)~~ Hot Data Analysis

In Situ Analysis: Reproducibility

+ review
+ eductation

- Mandates **open science approach**
- **Repeat** (simulation, analysis) with:
  - newly measured data / computing time
  - new question (query)
- Imagine LHC High-Luminosity upgrade:
  - submit in situ **“queries”** before beam-time
  - just as in software: **review, improve,** … queries
  - repeat **modified** queries in next beam-time

explorative science!

EUCALL

NVIDIA GPU CENTER OF EXCELLENCE

DRESDEN concept

HZDR

Expressive, Productive Programming
Interactive

Use all mentioned interactively on HPC systems

- **Data analysis** in notebooks with GPUs
- Big, **interactive simulation** with GPUs
- **Teaching** GPU programming
- Easing **development** and debugging

via **Jupyter Notebook.**

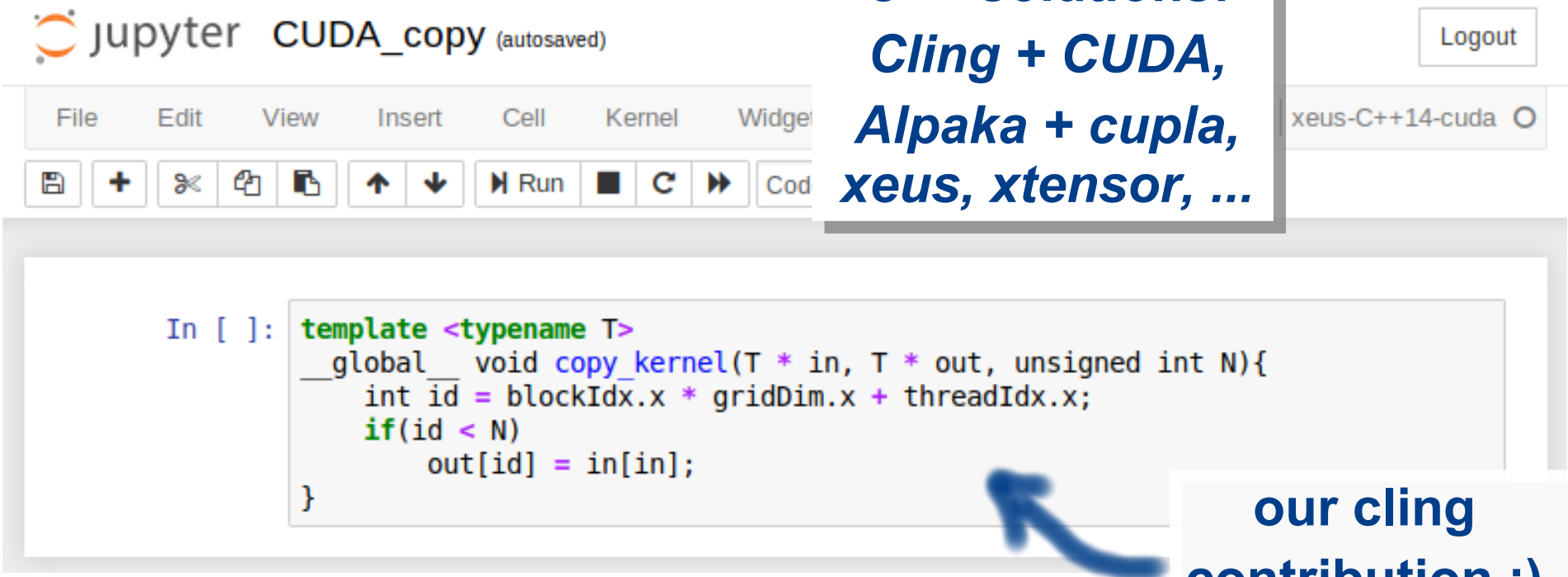
C++ meta-programming == *compile!*

Cling CUDA: S. Ehrig (HZDR, TU Dresden), Diploma Thesis (2018)

Expressive, Productive Programming

Interactive, Modern, JITed C++

C++ solutions:
Cling + CUDA,
Alpaka + cupla,
xeus, xtensor, ...



our cling contribution :)

- full **Alpaka** support → template change: CPU/GPU/...
- let's experiment: extend C++ for interpretation
  - store & restore full **Cling state**
  - fully **dynamic types**? e.g. __attribute__((variant_all)) or **re-definable** class / function

Cling CUDA: S. Ehrig (HZDR, TU Dresden), Diploma Thesis (2018)

Summary
Our Existing C++ & OpenData Projects

Our Scientific FOSS C++ Projects

Try, Reuse, Complain ;-) and Improve

ComputationalRadiationPhysics/

Alpaka

mallocMC

ISAAC

LLAMA

PIConGPU

PMacc

@ax3l

root-project/cling: -x CUDA

This project has been enabled by many people in open-source and open-science communities. Great thanks to the communities and developers of: PIConGPU, Alpaka, Jupyter, the SciPy ecosystem, ADIOS, HDF5, CMake, openPMD, Spack, Xeus, Cling, ...

This research used resources of the Oak Ridge Leadership Computing Facility located in the Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 654220.

Talk by **A. Huebl** (HZDR), a.huebl@hzdr.de

Backup Slides

Application Software Stack

Let's Draft an Ideal World

Application

helper

Containers and Algorithms

In-Node Acceleration

Message-Passing

Application C++ Software Stack

PIConGPU

PIConGPU

Plugins

Boost

PMacc

cupla

Alpaka

mallocMC

LLAMA*
*still in PMacc

MPI

CUDA, OpenMP, TBB, ...

NVIDIA GPU CENTER OF EXCELLENCE

DRESDEN concept

HZDR

Code Example: OpenMP

Mix of algorithm/data & parallelization strategy

- Pragma-based approaches, e.g. **OpenMP**:

```
auto axpy(
    double const & alpha,
    double const * const X,
    double * const Y,
    std::size_t const & n
) const
    -> void
{
    #pragma omp parallel for
    for( std::size_t i = 0u; i < n; ++i )
    {
        Y[i] = alpha * X[i] + Y[i];
    }
}
```

Code Example: Alpaka

Zero-Overhead, User-Specializable

- Parallelized depending on Hardware `TAcc const & acc`

```
struct DaxpyKernel
{
    template< typename T_Acc >
    ALPAKA_FN_ACC void operator()(
        T_Acc const & acc,
        double const & alpha,
        double const * const X,
        double * const Y,
        int const & numElements
    ) const
    {
        using alpaka;
        auto const i = alpaka::idx::getIdx< Grid, Threads >( acc )[0];
        Y[i] = alpha * X[i] + Y[i];
    }
};
```

Code Example: Alpaka II

SIMD Optimized

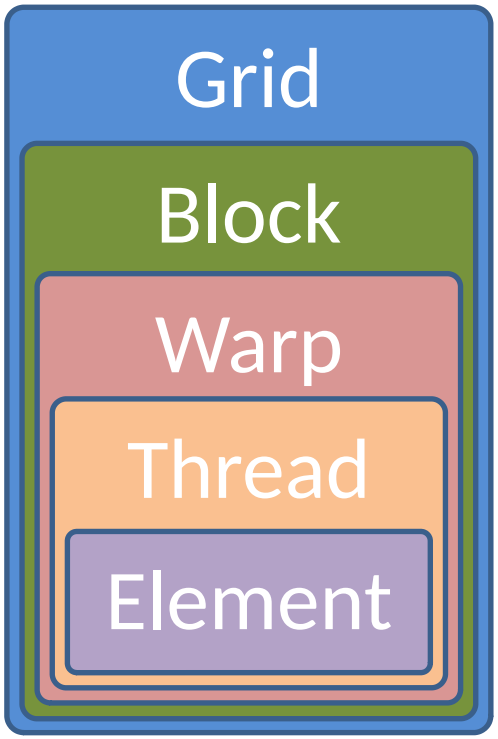
```
struct DaxpyKernel
{
    template< typename T_Acc >
    ALPAKA_FN_ACC void operator()(
        T_Acc const & acc,
        double const & alpha,
        double const * const X,
        double * const Y,
        int const & numElements
    ) const
    {
        using alpaka;
        auto const globalIdx = idx::getIdx< Grid, Threads >( acc )[0u];
        auto const elemCount = workdiv::getWorkDiv< Thread, Elems >( acc )[0u];

        auto const begin = globalIdx * elemCount;
        auto const end = min( begin + elemCount, numElements );

        for( TSize i = begin; i < end; i++ )
            Y[i] = X[i] + Y[i]; // Note difference between worker and data index
    }
};
```

Explicit Programming Model

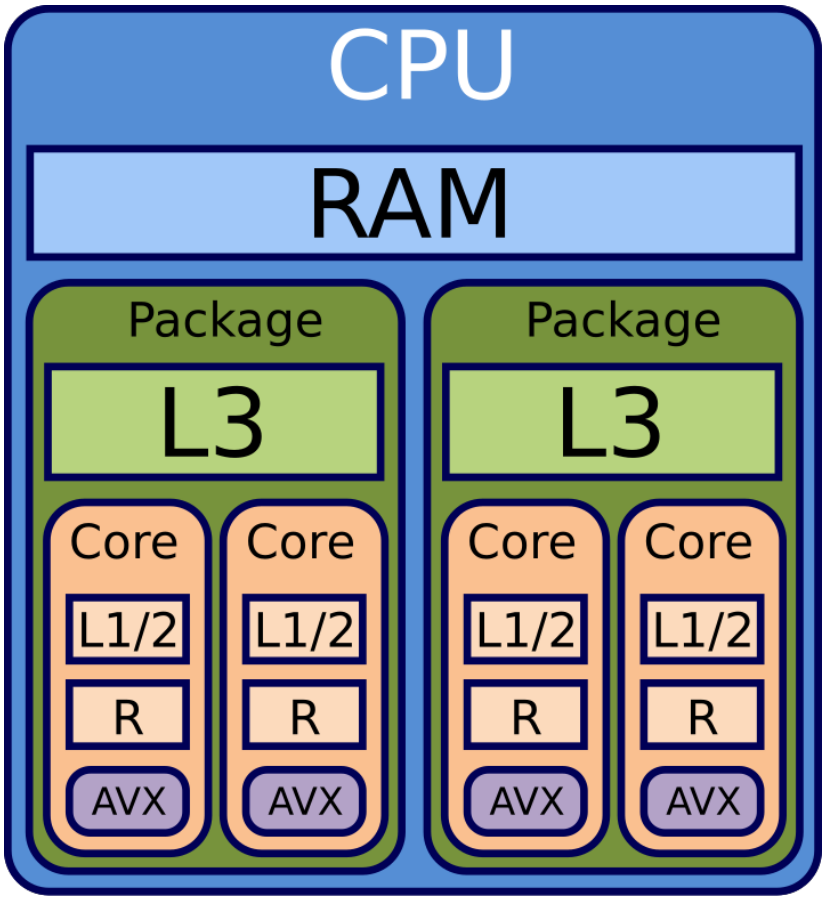
Alpaka

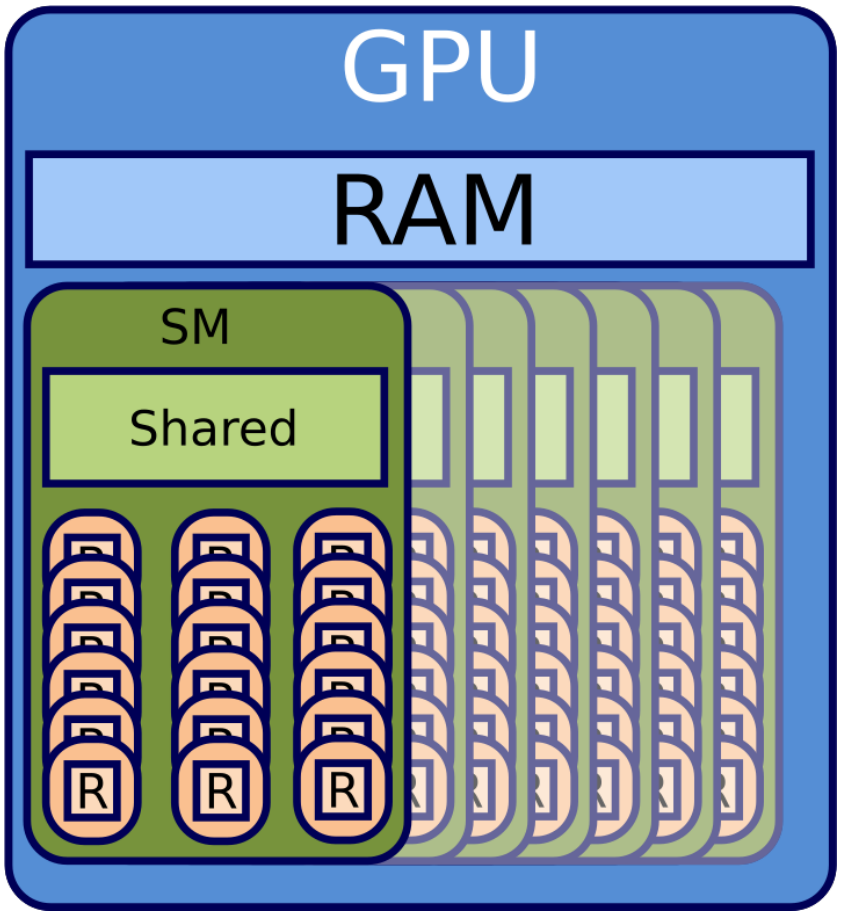


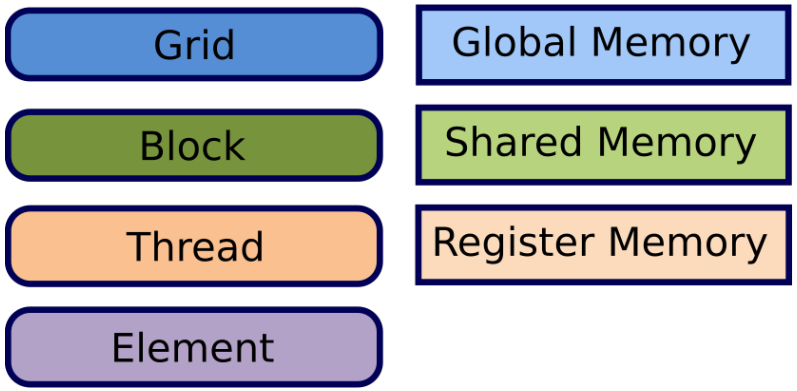
- **Grid** whole parallel task
- **Block** fully independent part of the grid
- **Warp** group of synchronous threads
- **Threads** executed concurrently
- **Elements** sub-thread, sequential lock-step

Hardware Mapping

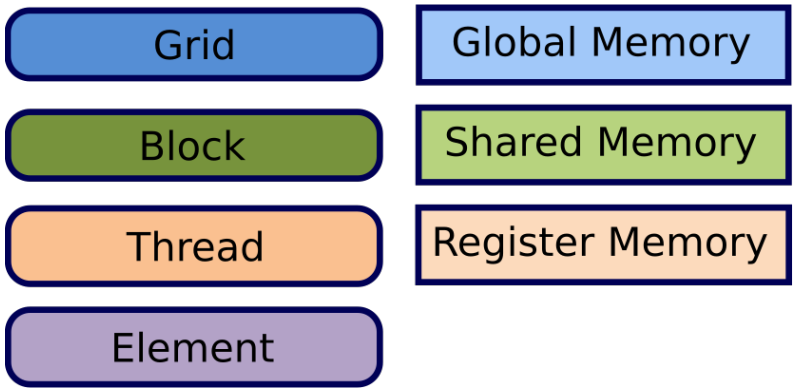
Alpaka on CPU, GPU, ...

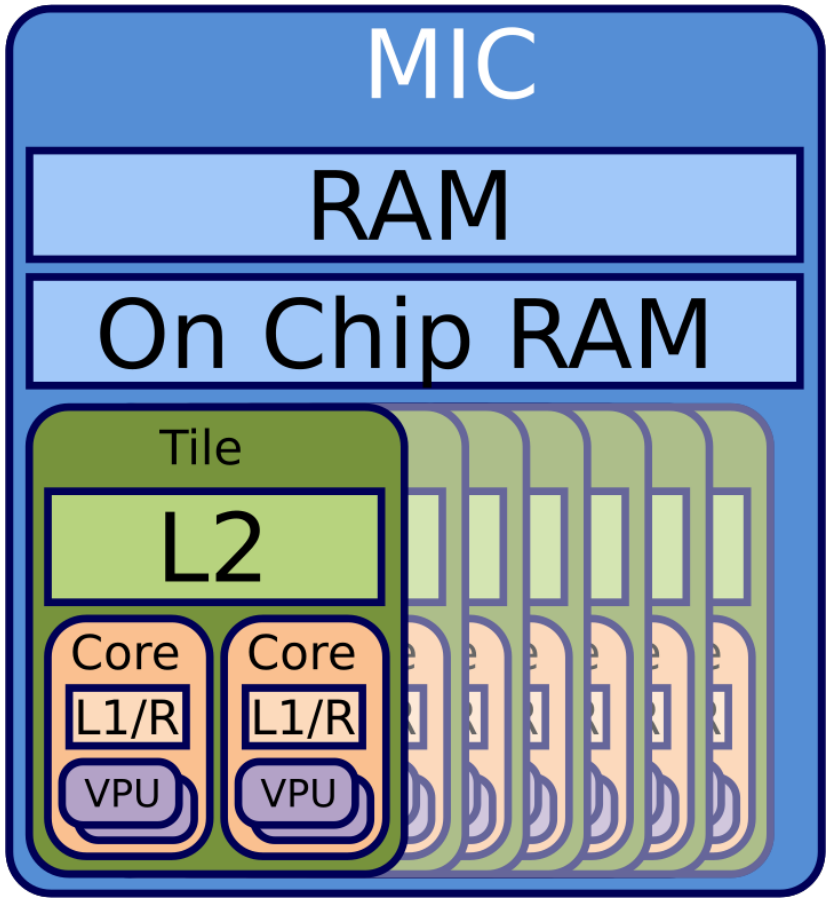


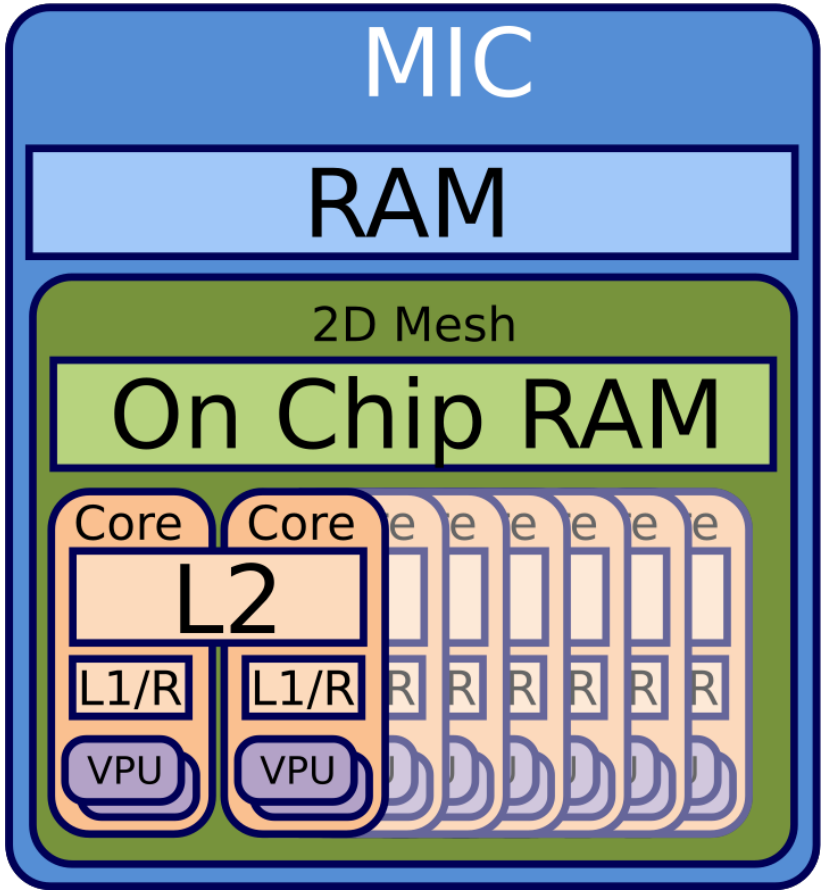




Hardware Mapping
…, MIC

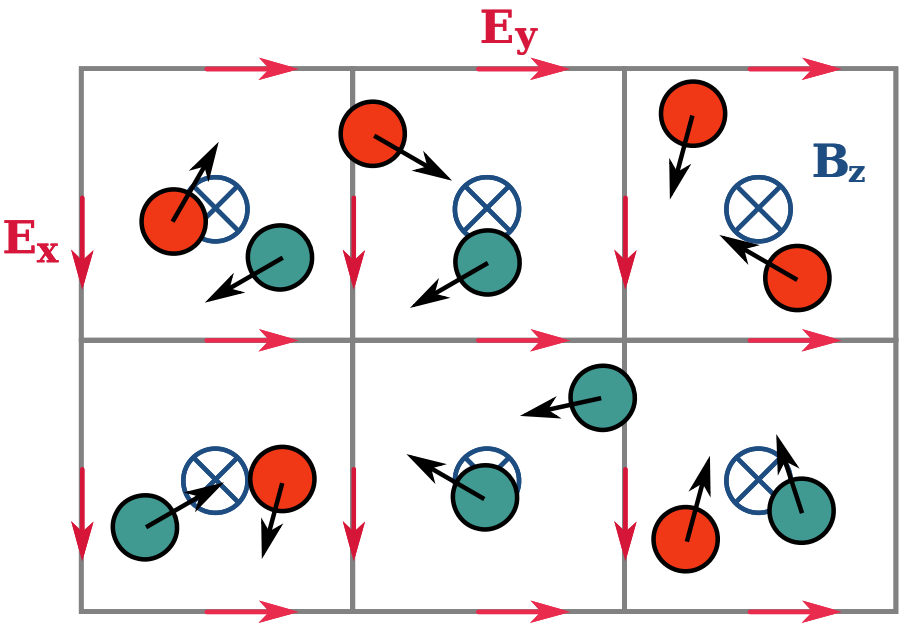






PIConGPU

- **Eulerian: electro-magnetic fields**
- **Lagrangian: particles in Vlasov-equation**



7.2 PFlop/s (DP)
+ 1.4 PFlop/s (SP)

96% weak scaling efficiency

M. Bussmann et al., SC'13, DOI:10.1145/2503210.2504564