PIConGPU: 10 years of living with the HPC hardware zoo

Michael Bussmann

19/05/2020 | JSC MSA Seminar

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

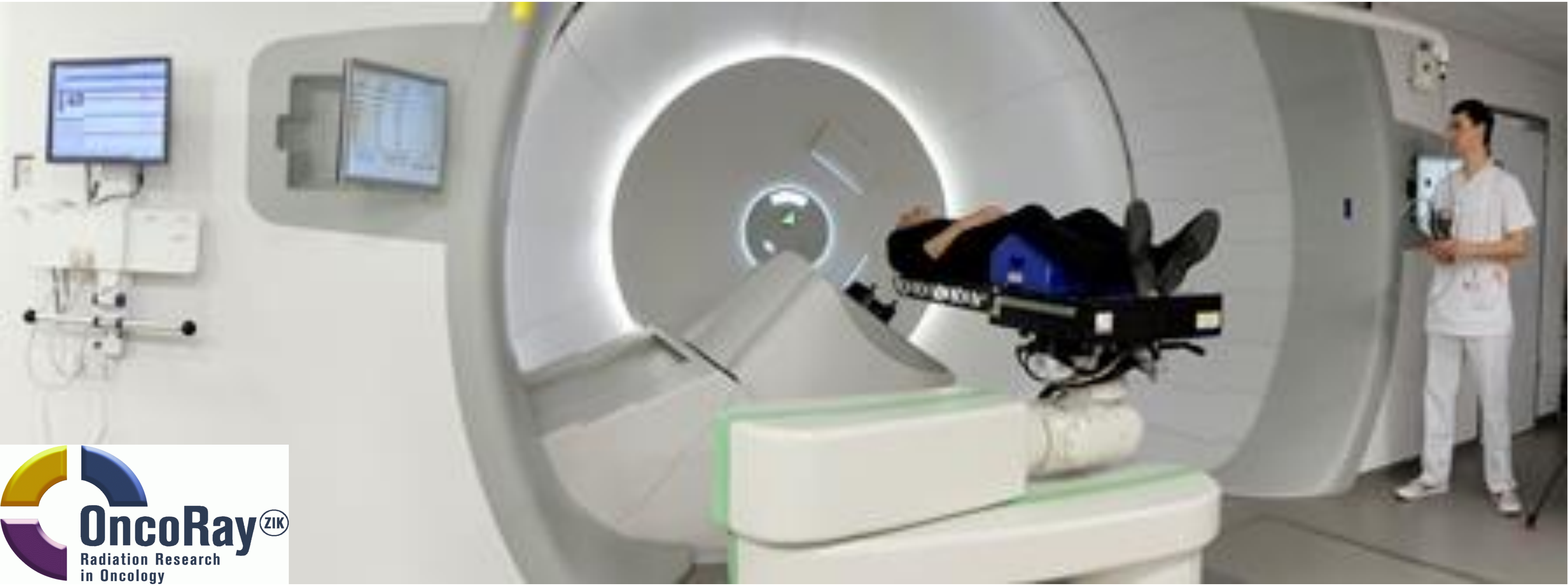
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Zentrum für Umweltforschung
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Plasma accelerators

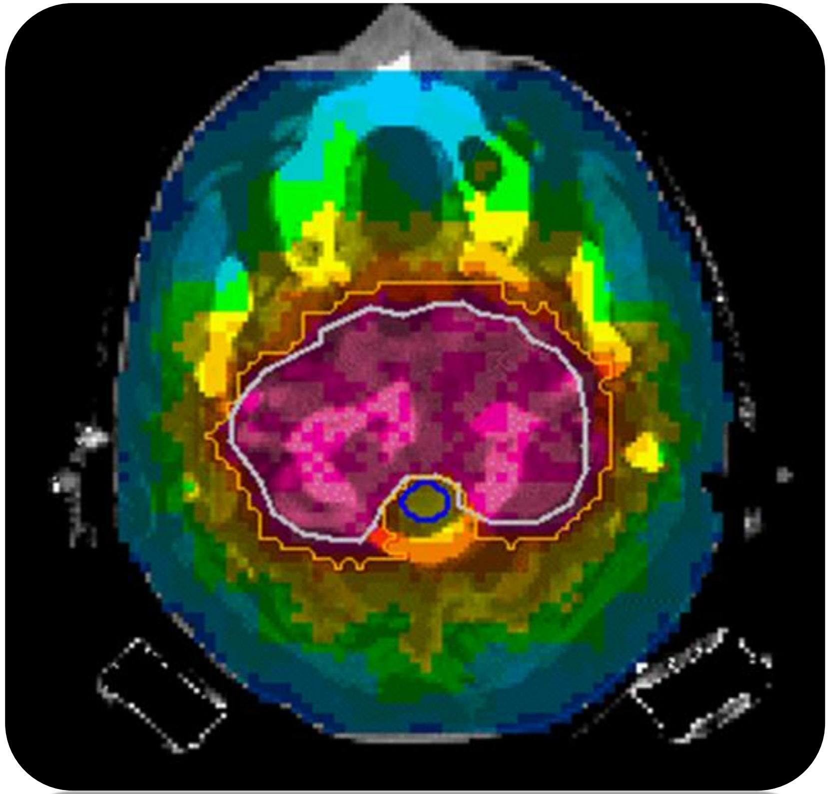
Radiation Tumor Therapy

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

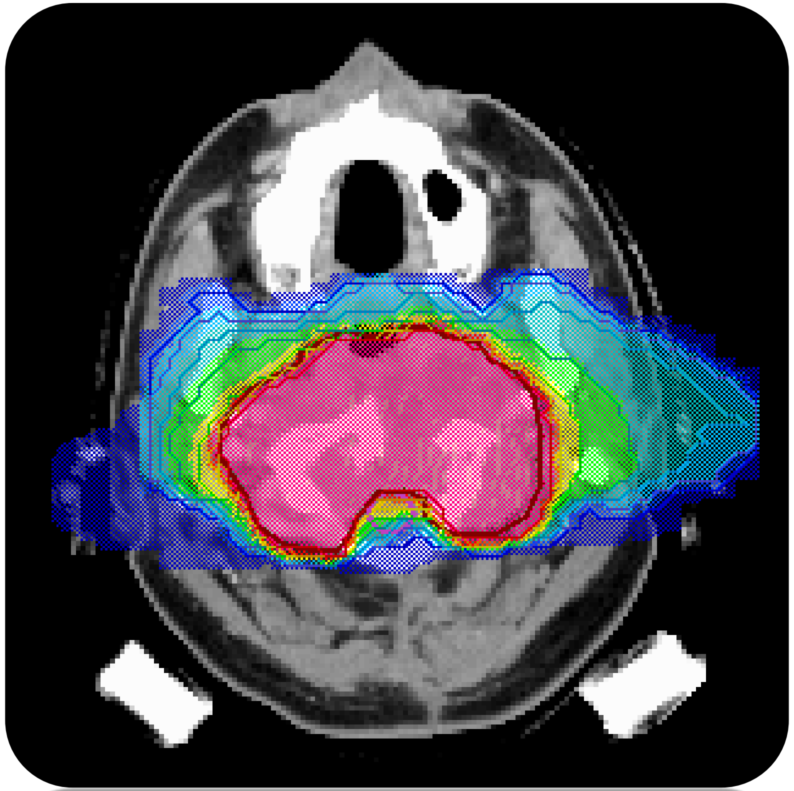


Plasma accelerators

Radiation Tumor Therapy with Ions



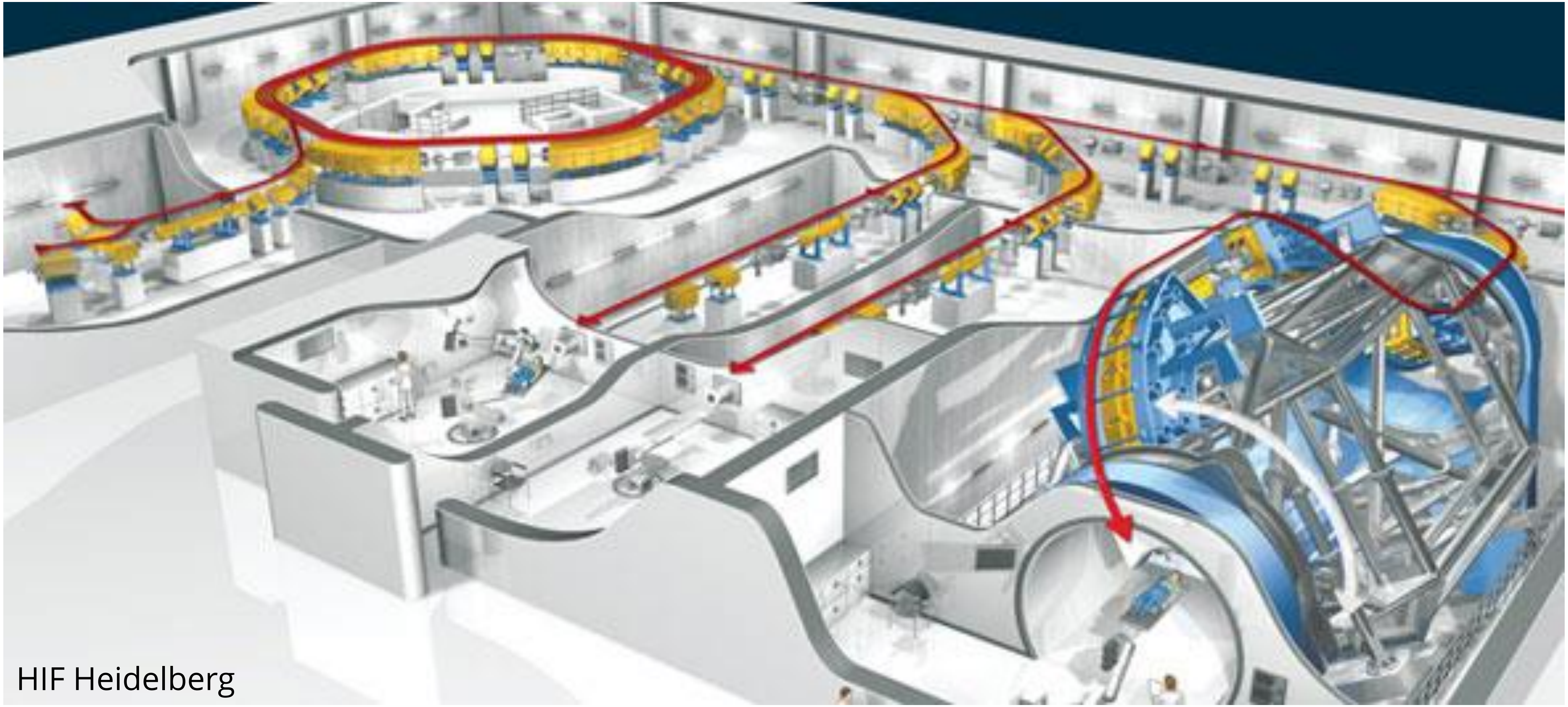
8 X-Ray Beams



2 Ion Beams

Plasma accelerators

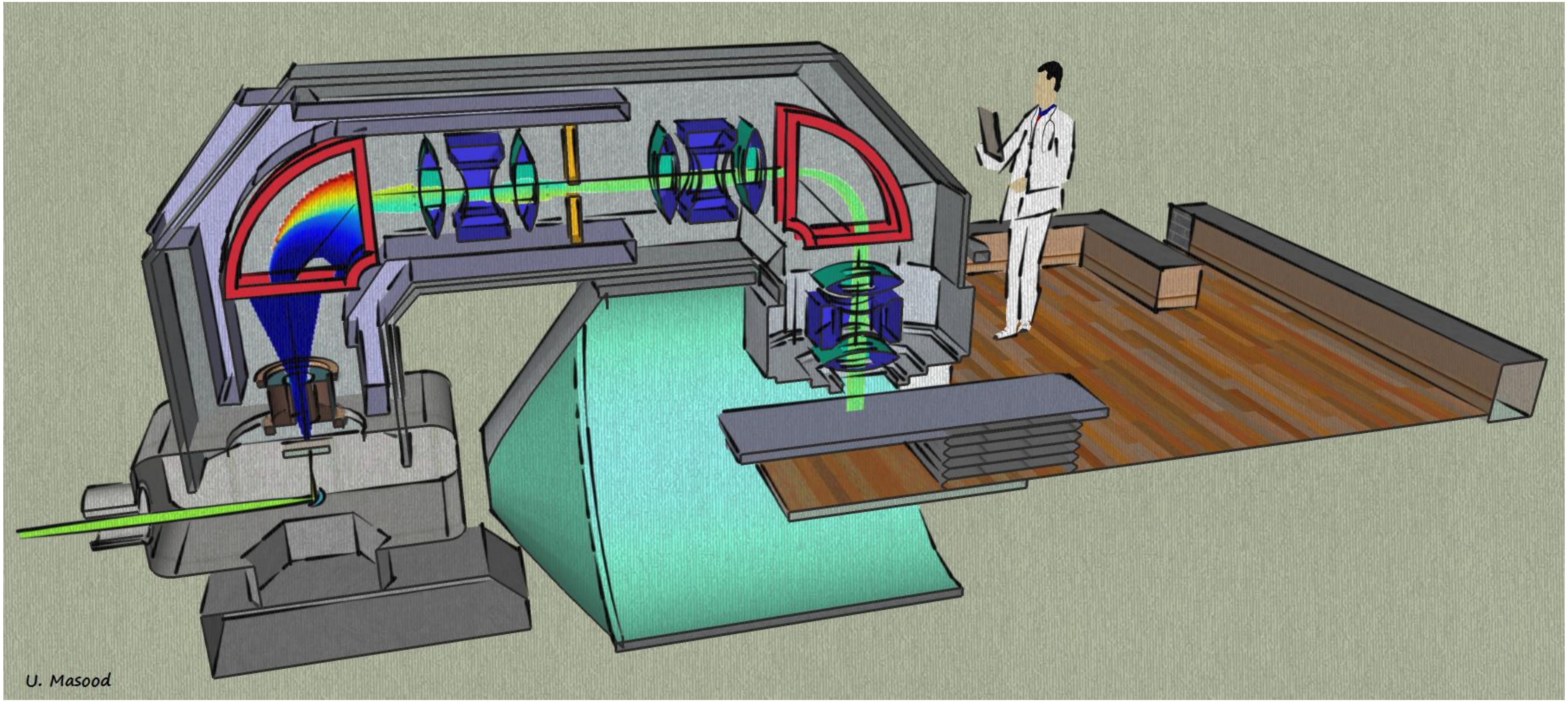
Accelerators can become quite big machines



HIF Heidelberg

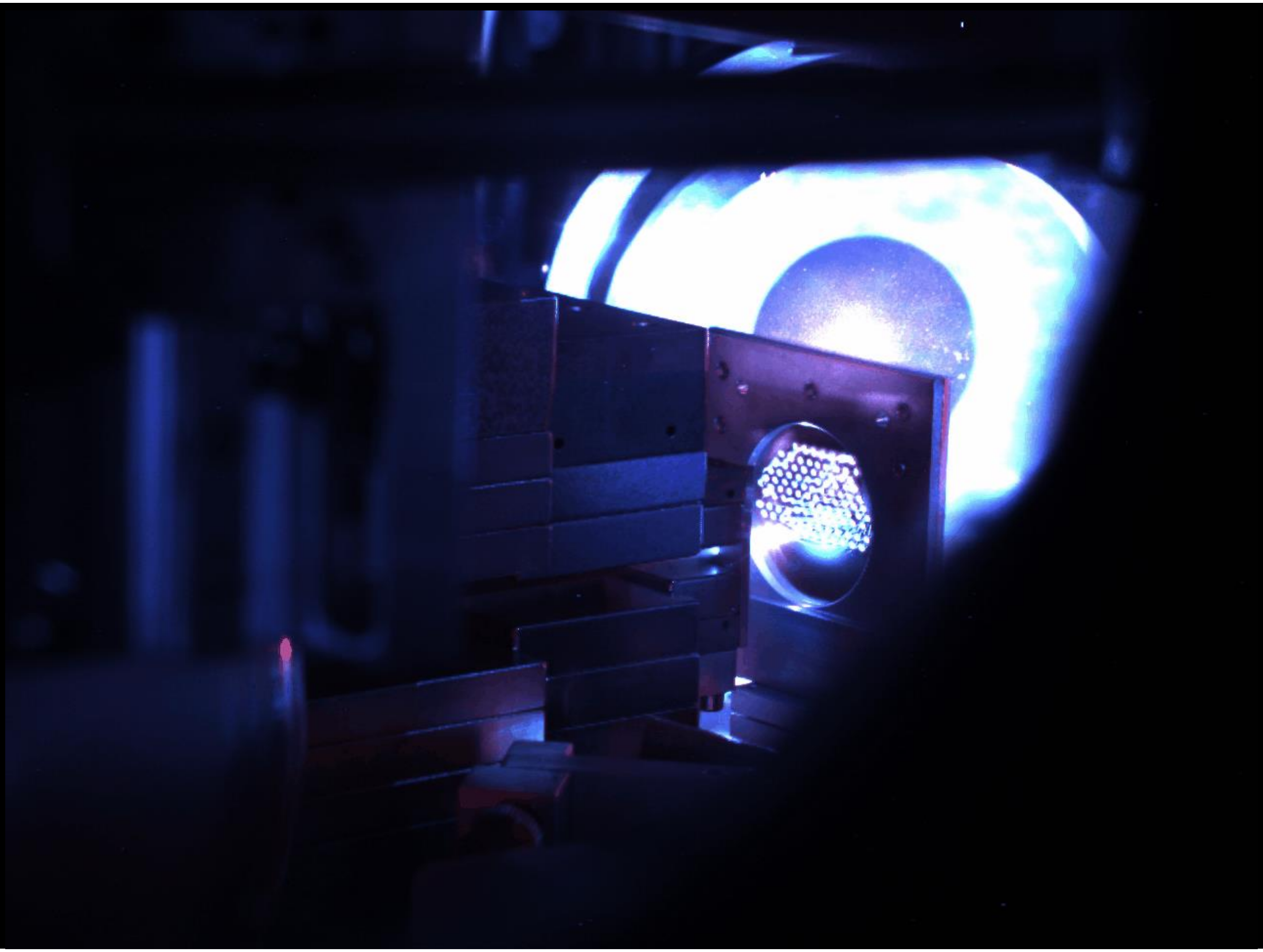
Plasma accelerators

Can we make them smaller?



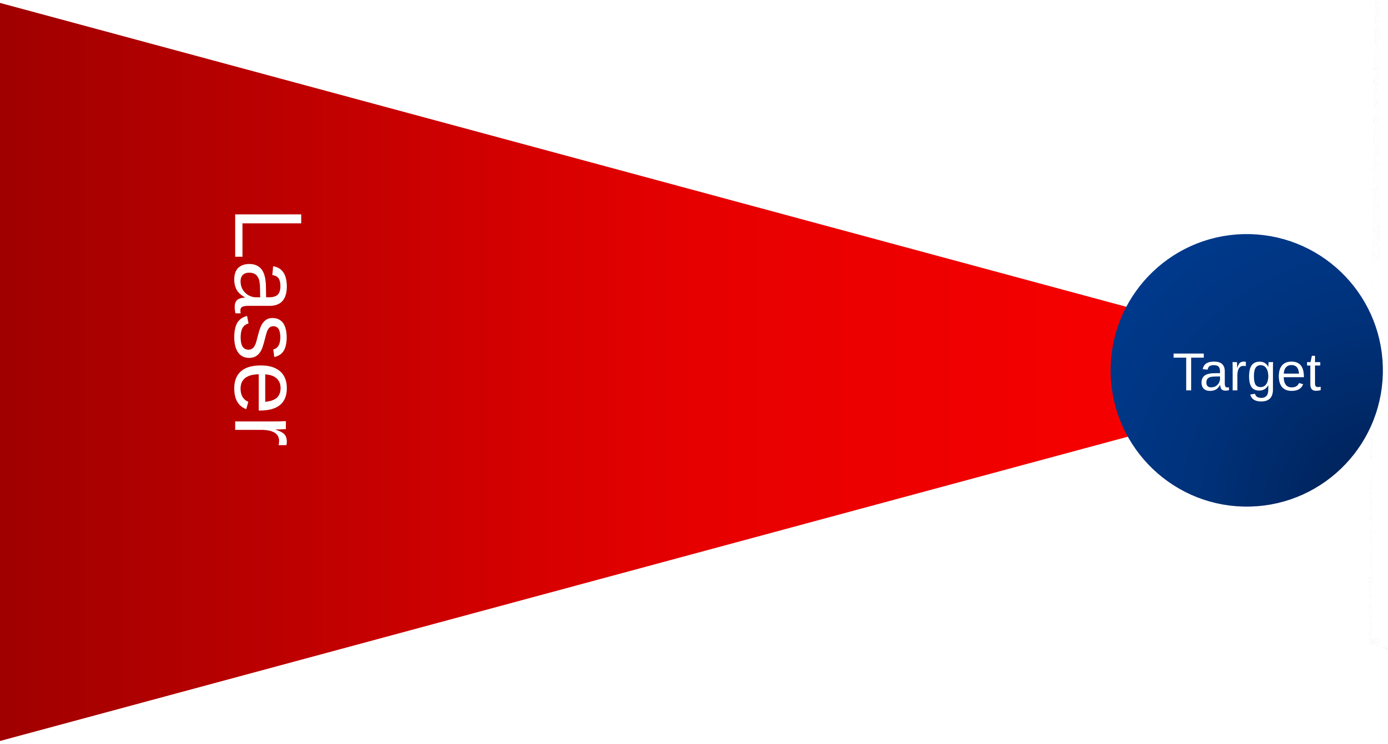
Plasma accelerators

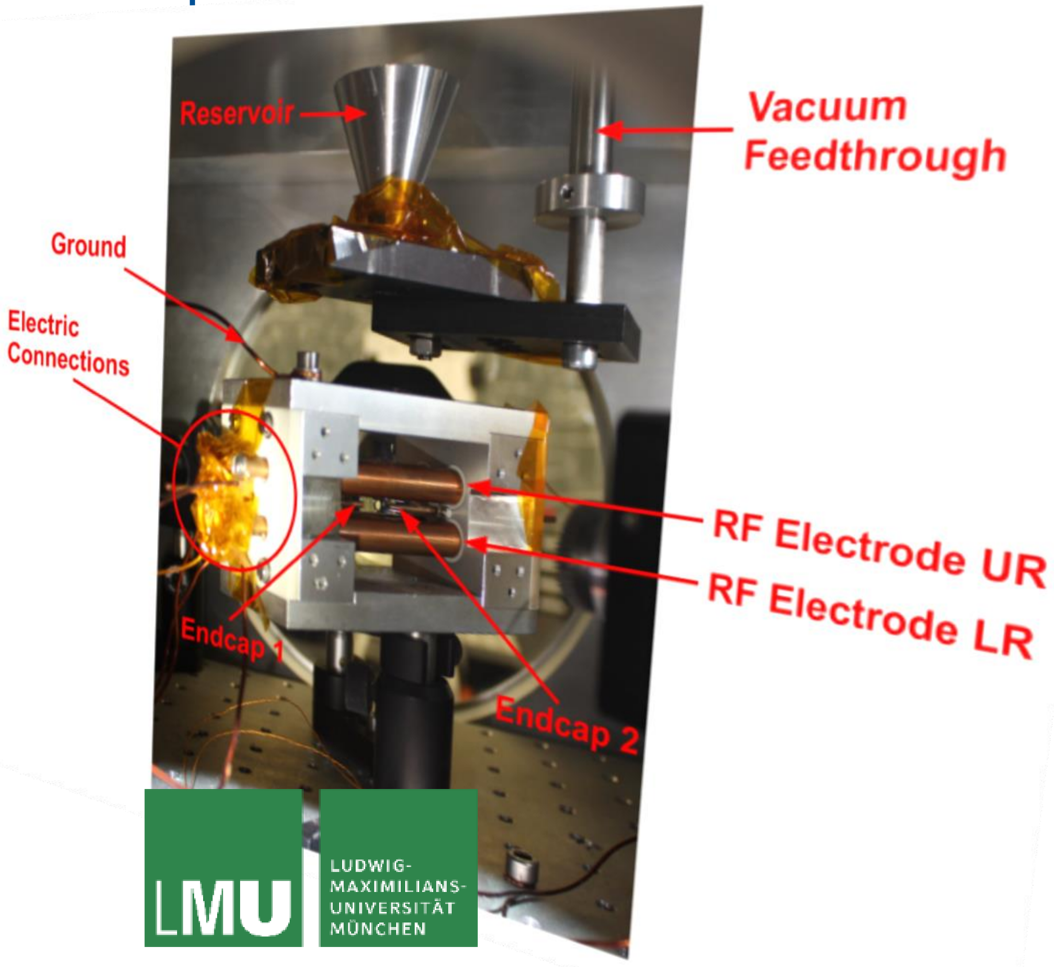
Lasers FTW!



Plasma accelerators

Making everything very easy with levitating platic spheres



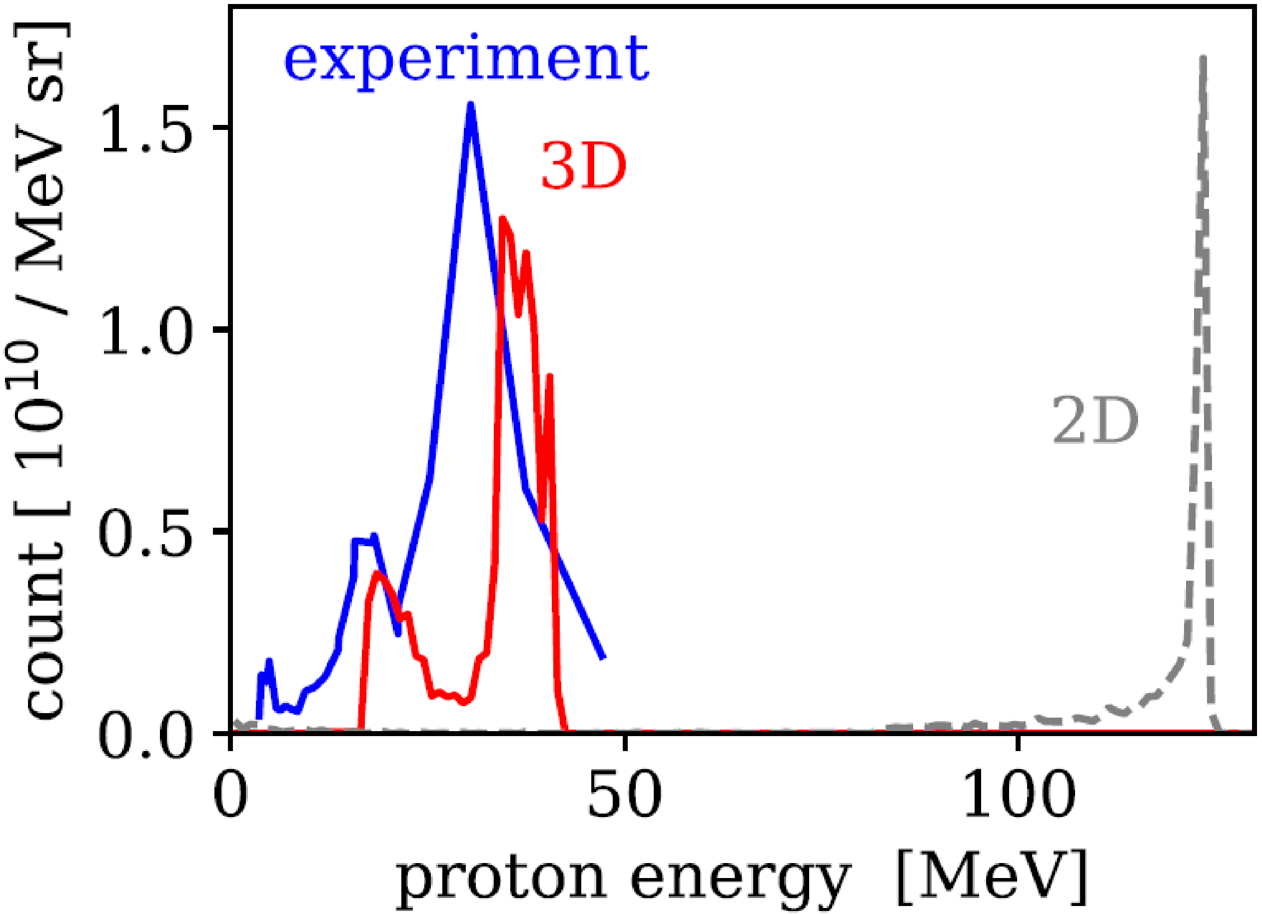


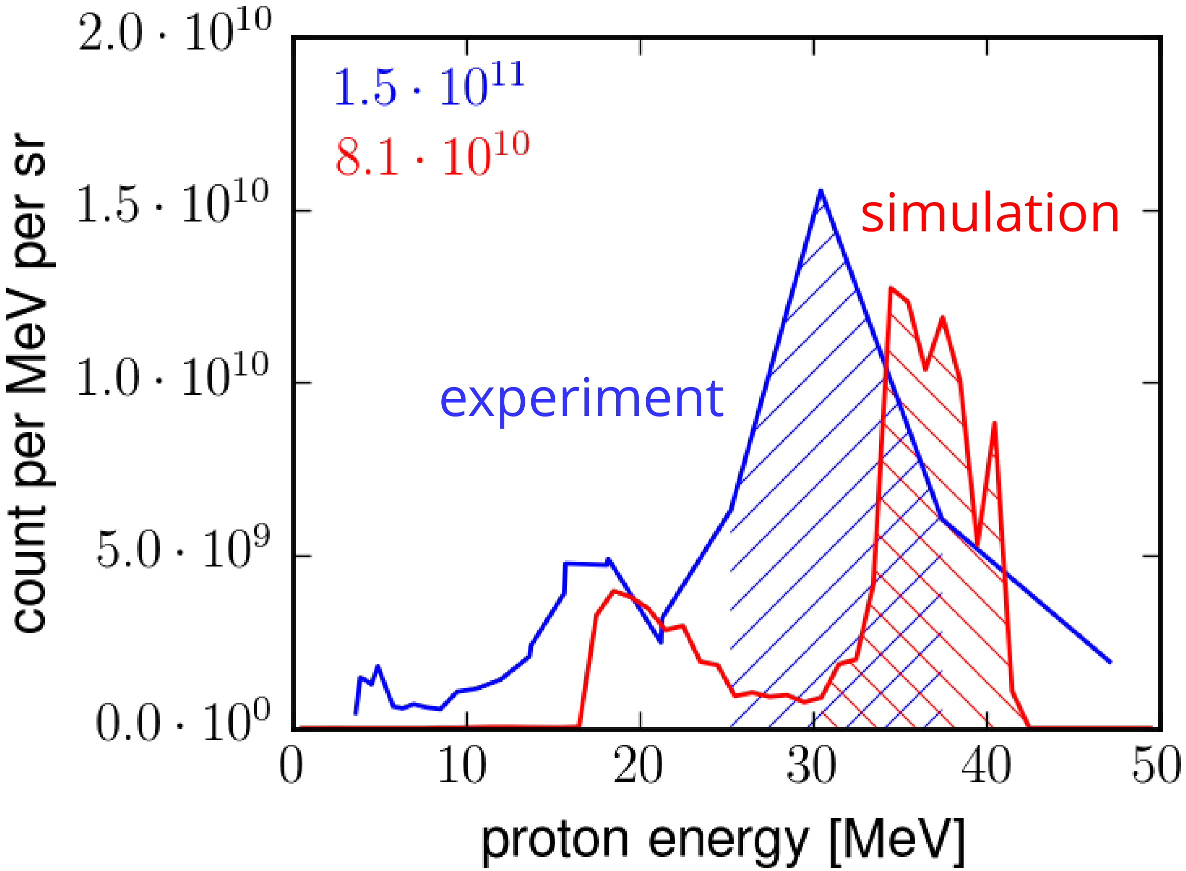
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

PIConGPU

Plasma accelerators

In theory, theory is easy



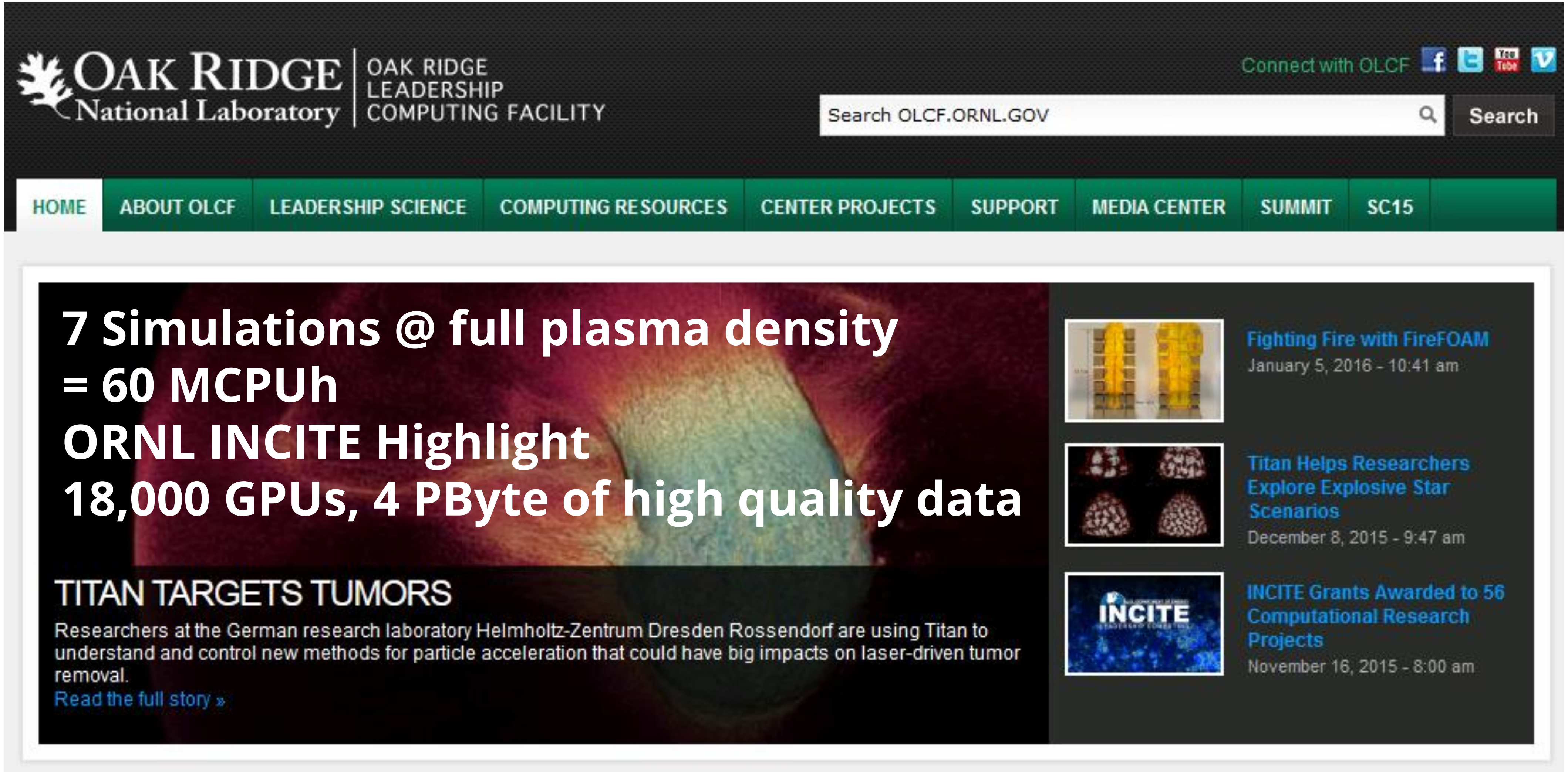


PIConGPU

Plasma accelerators (2015 on ORNL TITAN / #1 Top 500)

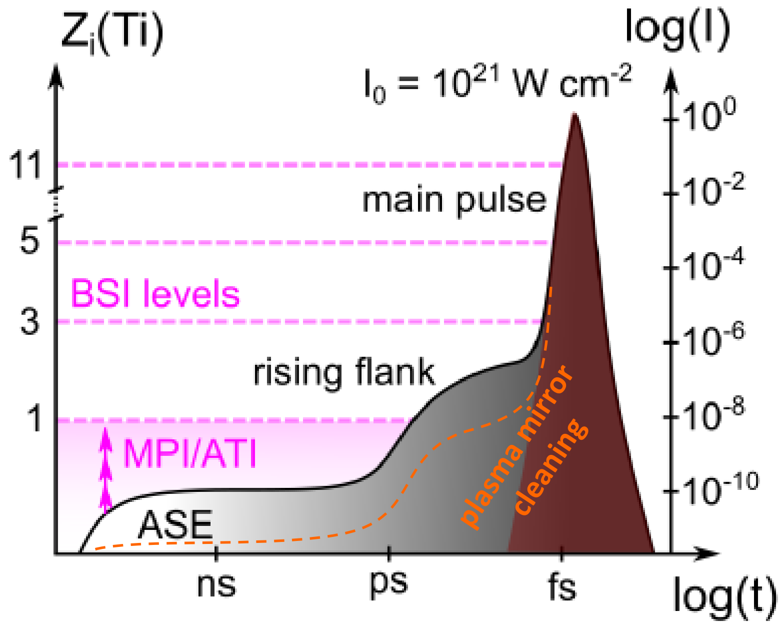
From 4 PByte to 100 kByte

7 Simulations @ full plasma density
= 60 MCPUh
ORNL INCITE Highlight
18,000 GPUs, 4 PByte of high quality data



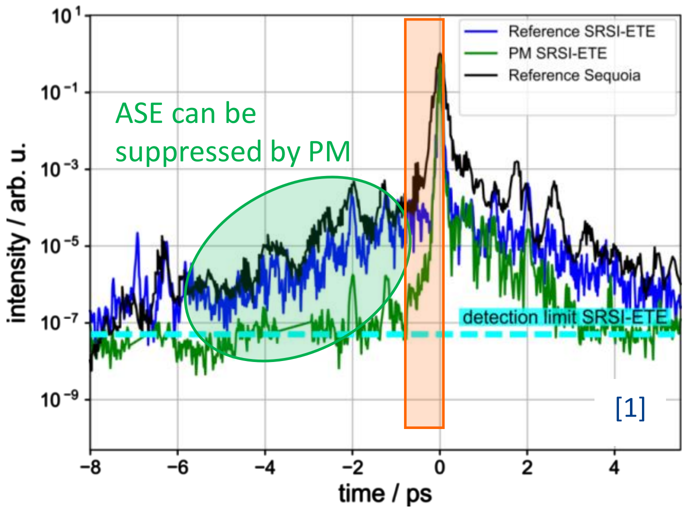
Plasma accelerators

Spanning 6 orders of magnitude in time



- “cleaning” of temporal contrast with plasma mirror techniques

Features on sub-ps scale remain!

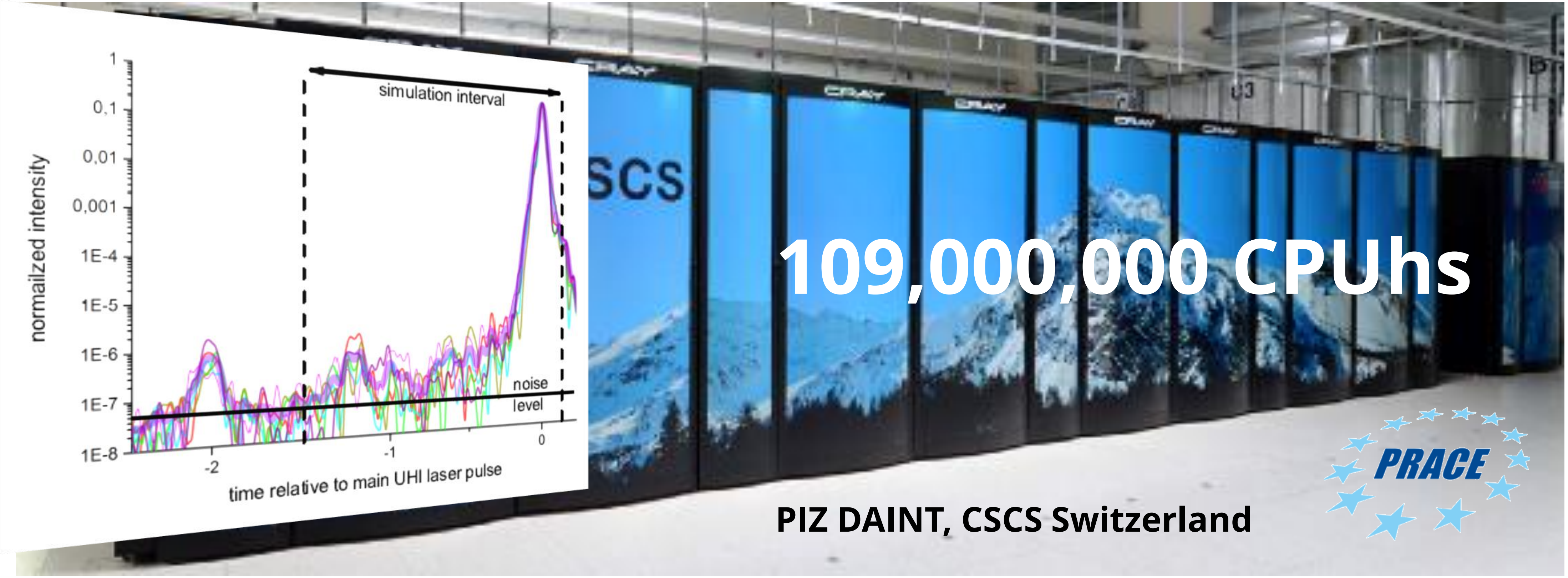


- Shot-to-shot fluctuation
- Experimentally accessible but still challenging to measure

Plasma accelerators (2018 on CSCS Piz Daint, #3 Top 500)

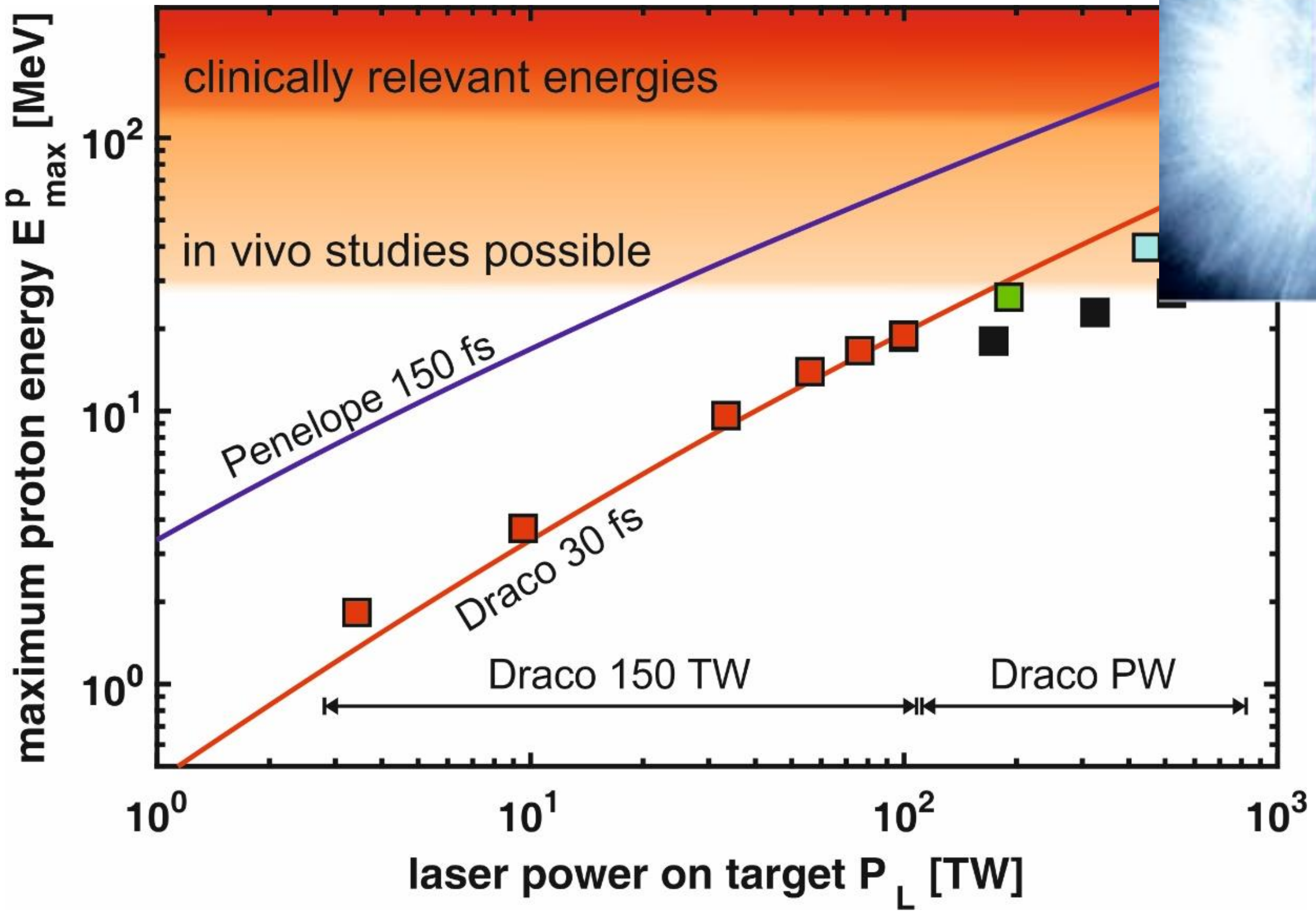
That was in the olden days – things are surely much better now!

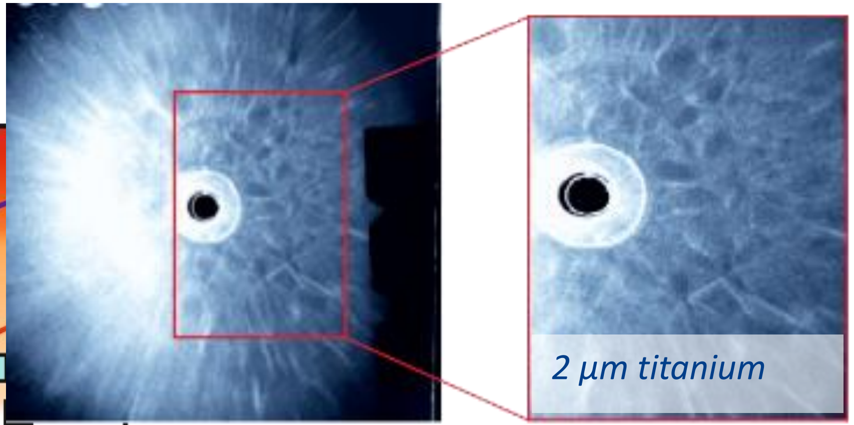
"Overall , this is an outstanding proposal. The High Performance Computing resources requested are appropriate. The PIs should try to ***reduce the data requirements and try to find a solution that is technically possible for CSCS****."*



Validating codes on the stomic scale

Towards higher energies



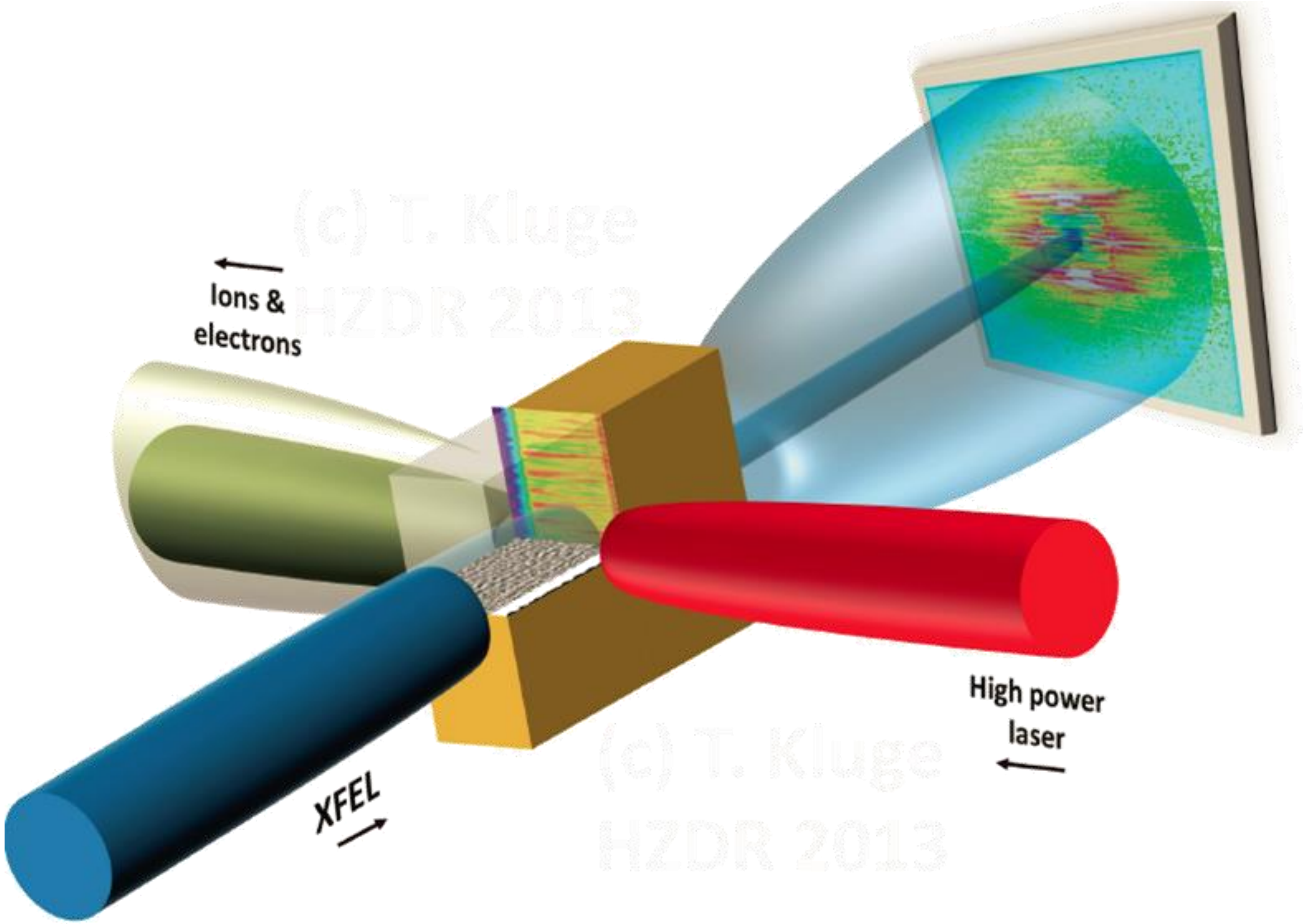


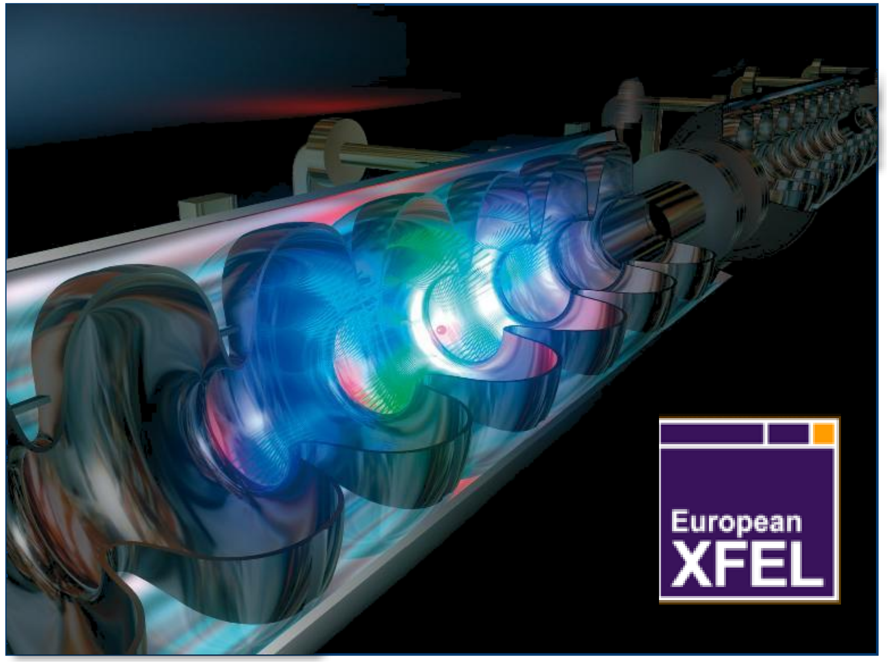
Plasma-Instabilities may degrade ion beam quality.

Clearly seen in experiment & simulation, but only simulations can provide atomic resolution of plasma dynamics….

Testing codes on the atomic scale

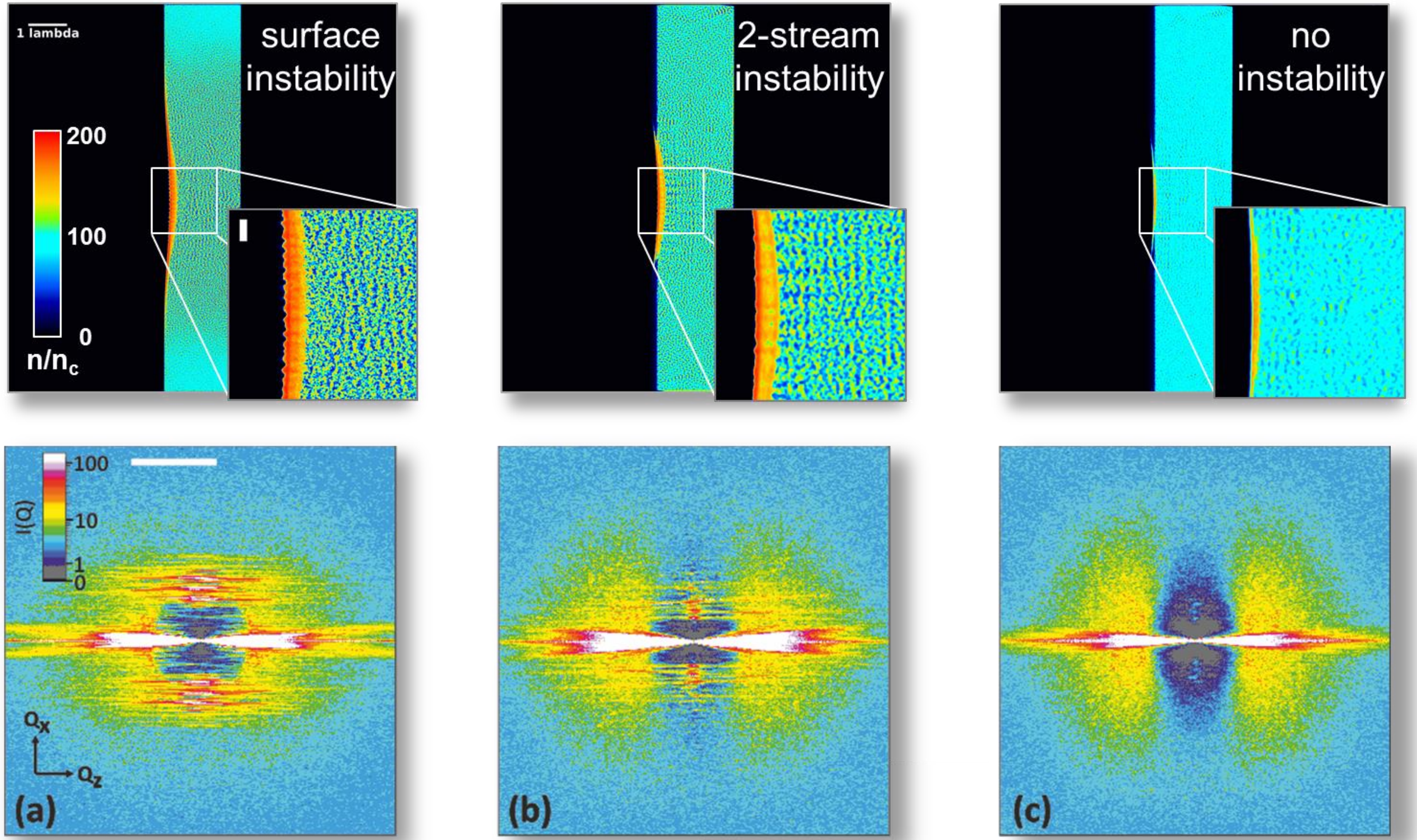
Looking at plasma dynamics at atomic resolution @ HIBEF / EU-XFEL





Testing codes on the atomic scale

Different instabilities create different scattering images

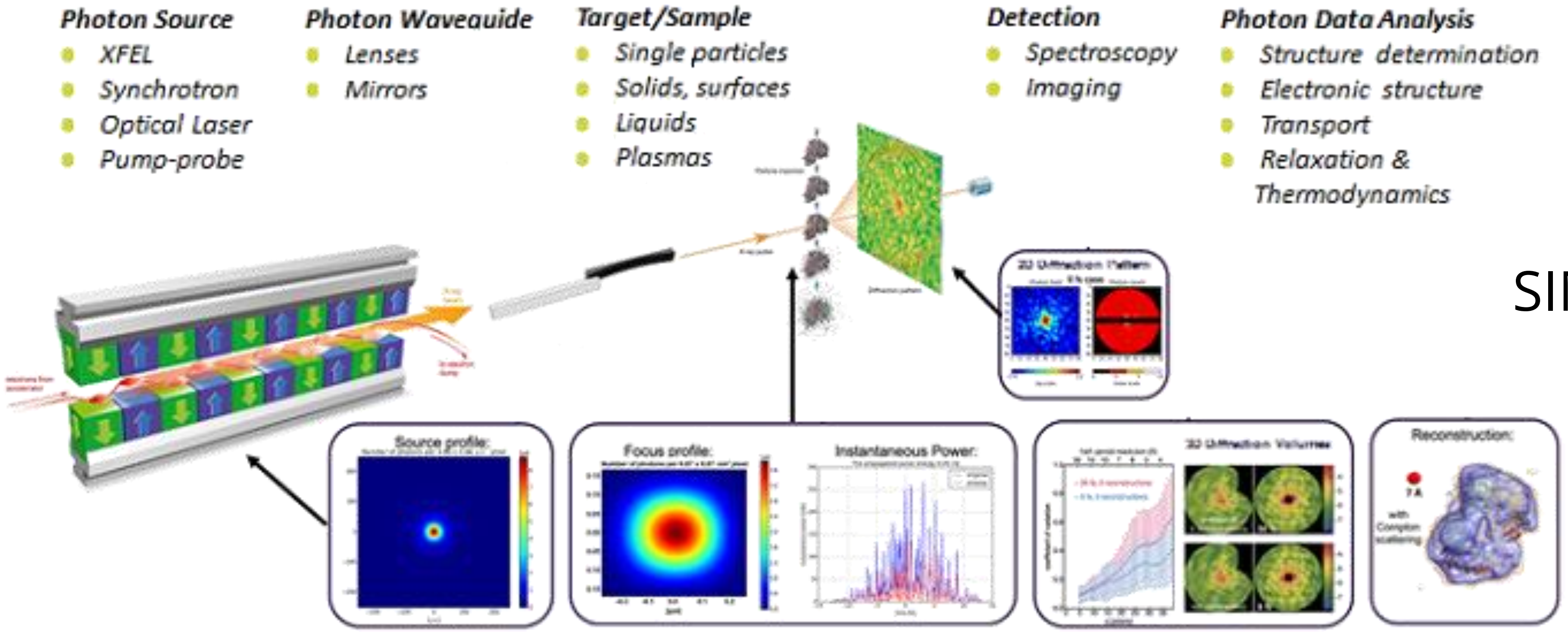


Simulated Plasma Density

Small Angle X-Ray Scattering

Testing codes on the atomic scale

Inversion of data is hard – Learning from CERN

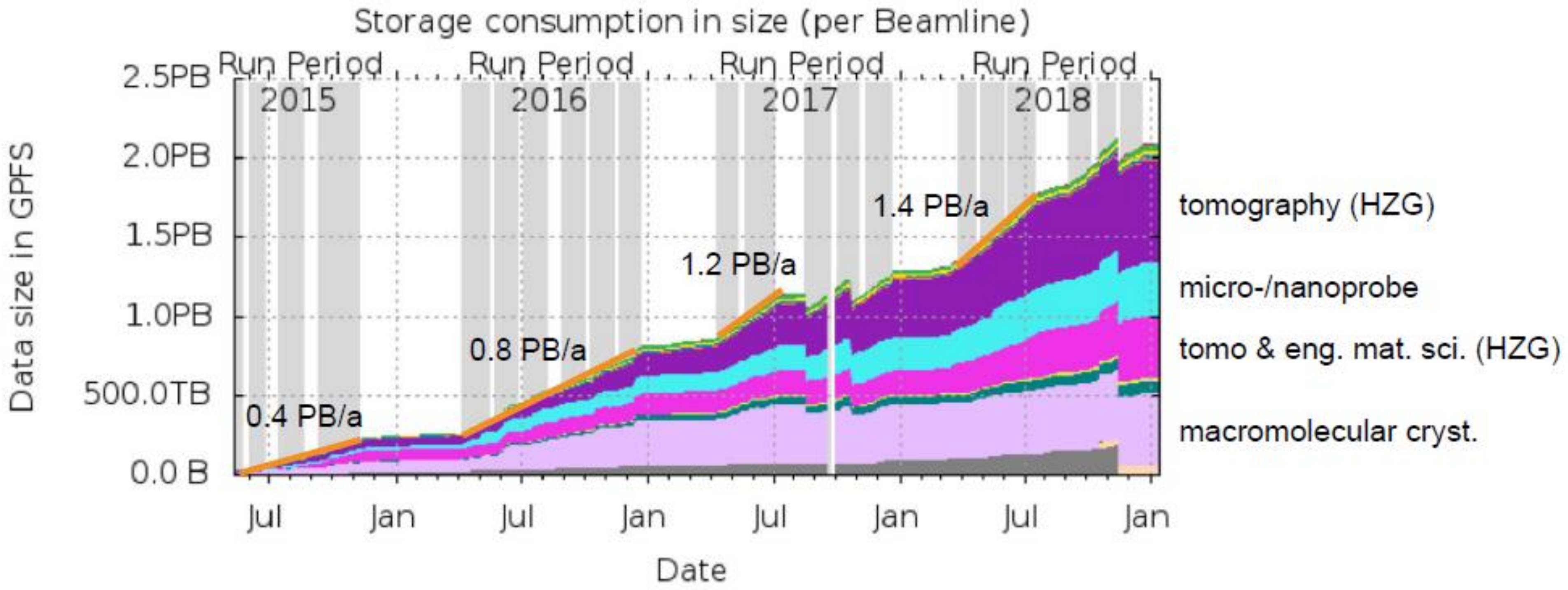


SIMEX_PLATFORM

Each system imaged is a full High Performance Computing simulation

Testing codes on the atomic scale

Data is coming



4 Beamlines generate 80 % of the data on GPFS

What we will (at least) need in the upcoming years

Data-intensive computing & Human in the Loop

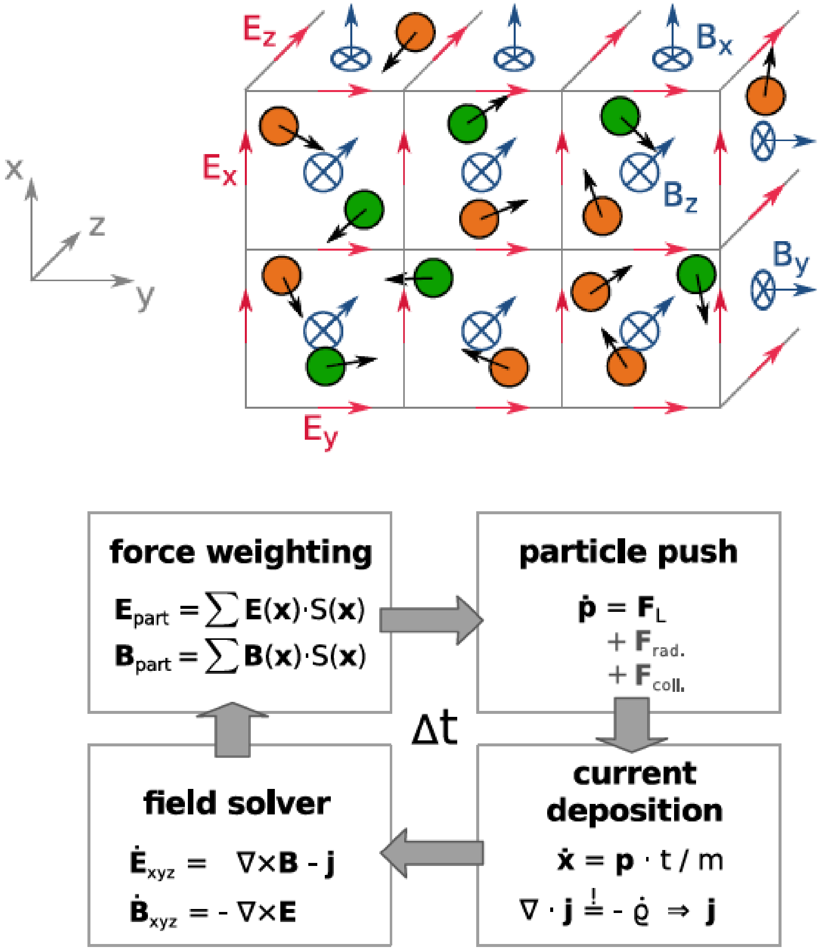
- Low-level software stacks for heterogeneous computing
- Data dependency and data flow descriptions
- Abstraction of communication and communication topologies
- A new way of thinking domain decomposition
- In-Memory workflow coupling
- Visual analytics combined with immersive UI interfaces, Machine Learning & Feedback
- Real time data fusion of experimental & simulation data and surrogate modeling

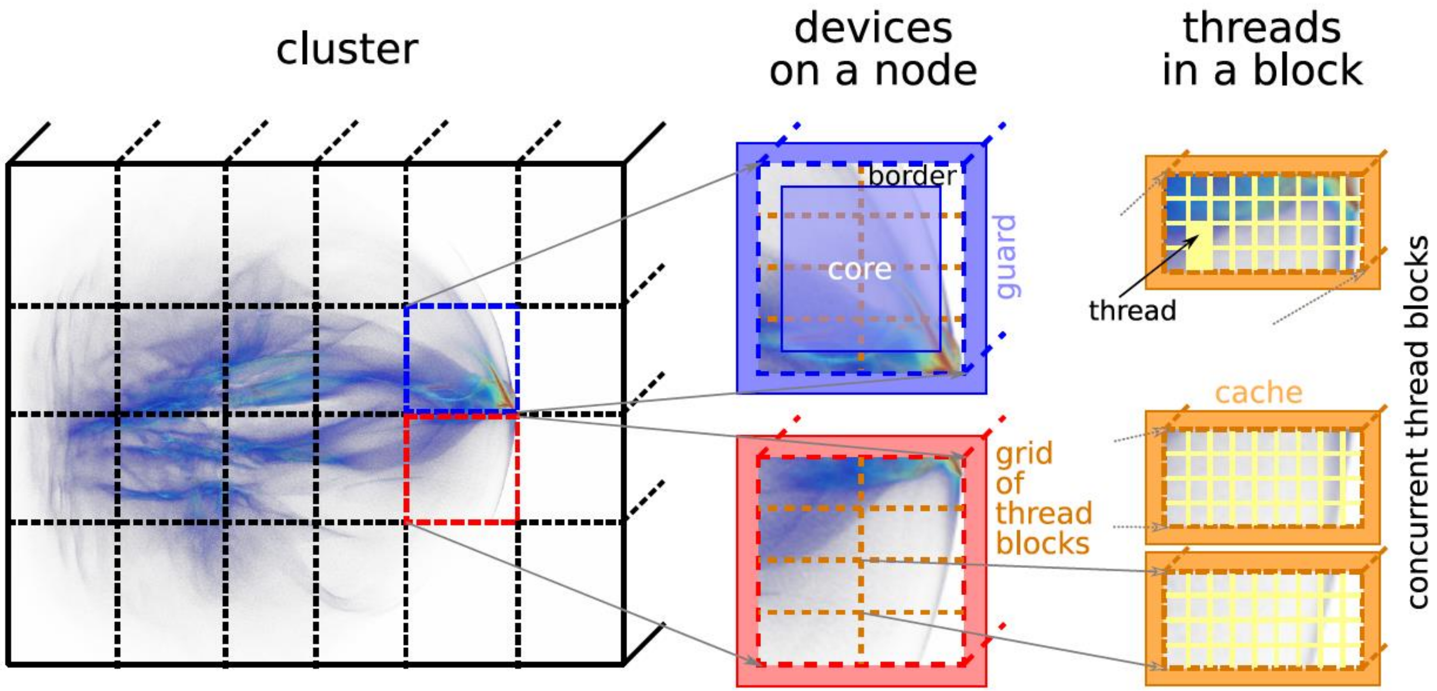
Data-intensive Software Stack

Human in the Loop

The Particle-in-Cell algorithm

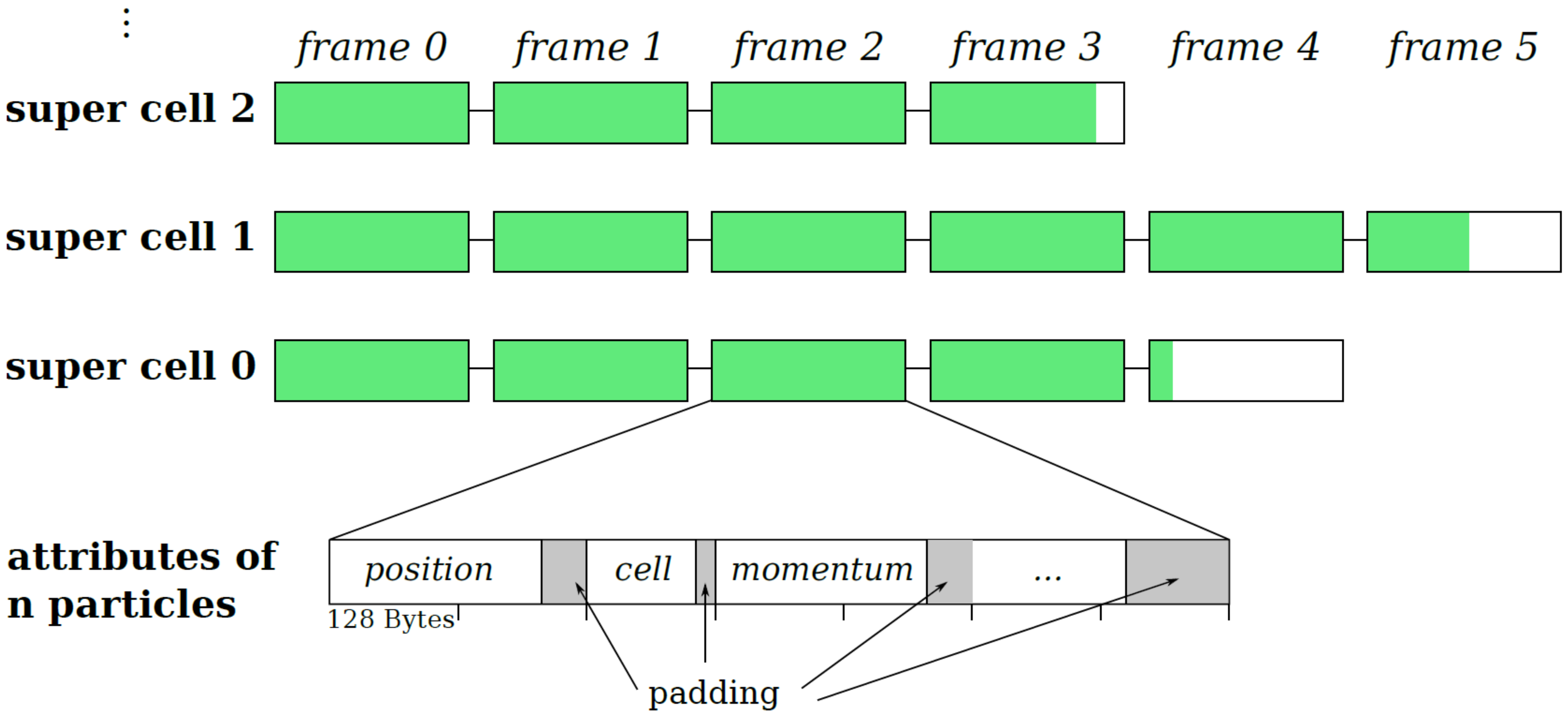
Domain decomposition in Super Cells





The Particle-in-Cell algorithm

Particle caching via Particle Frames



Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

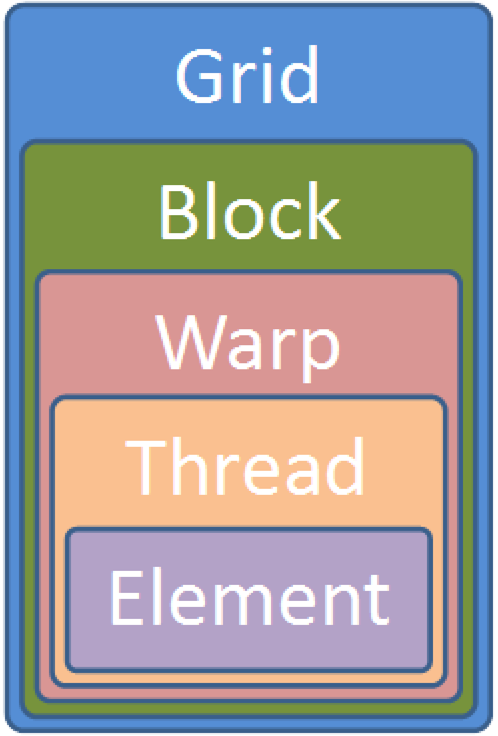
Single source heterogeneous many-core programming in C++

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

#elif OPENMP_ENABLE
    // OpenMP implementation
    // …

#else
    // Sequential CPU implementation
    // …

#endif
```

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , **IBM** DOE/SC/Oak Ridge National Laboratory **United States**	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , **IBM / NVIDIA / Mellanox** DOE/NNSA/LLNL **United States**	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , **NRCPC** National Supercomputing Center in Wuxi **China**	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , **NUDT** National Super Computer Center in Guangzhou **China**	4,981,760	61,444.5	100,678.7	18,482
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , **Cray Inc.** Swiss National Supercomputing Centre (CSCS) **Switzerland**	387,872	21,230.0	27,154.3	2,384
6	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , **Cray Inc.** DOE/NNSA/LANL/SNL **United States**	979,072	20,158.7	41,461.2	7,578
7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , **Fujitsu** National Institute of Advanced Industrial Science and Technology (AIST) **Japan**	391,680	19,880.0	32,576.6	1,649
8	**SuperMUC-NG** - ThinkSystem SD530, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , **Lenovo** Leibniz Rechenzentrum **Germany**	305,856	19,476.6	26,873.9	
9	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , **Cray Inc.** DOE/SC/Oak Ridge National Laboratory **United States**	560,640	17,590.0	27,112.5	8,209
10	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , **IBM** DOE/NNSA/LLNL **United States**	1,572,864	17,173.2	20,132.7	7,890

Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

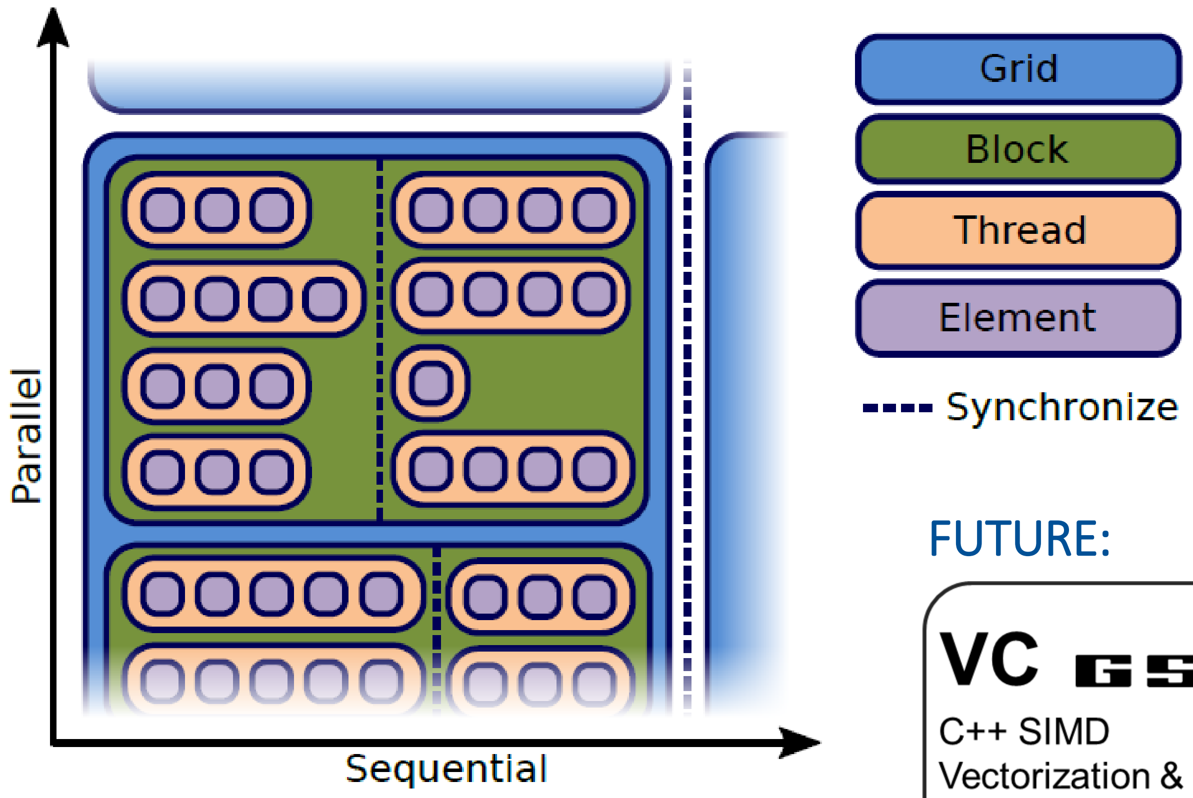
Parallel, redundant hierarchy (CUDA, OpenCL, HIP)



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

Abstraction **L**ibrary for **Pa**rallel **K**ernel **A**cceleration

Parallel, redundant hierarchy (CUDA, OpenCL, HIP)



Hierarchy Level	Parallelism	Synchronizable
grid	sequential / parallel	❌ / ✔
block	parallel	❌
warp	parallel	✔
thread	parallel / lock-step	✔
element	sequential	❌

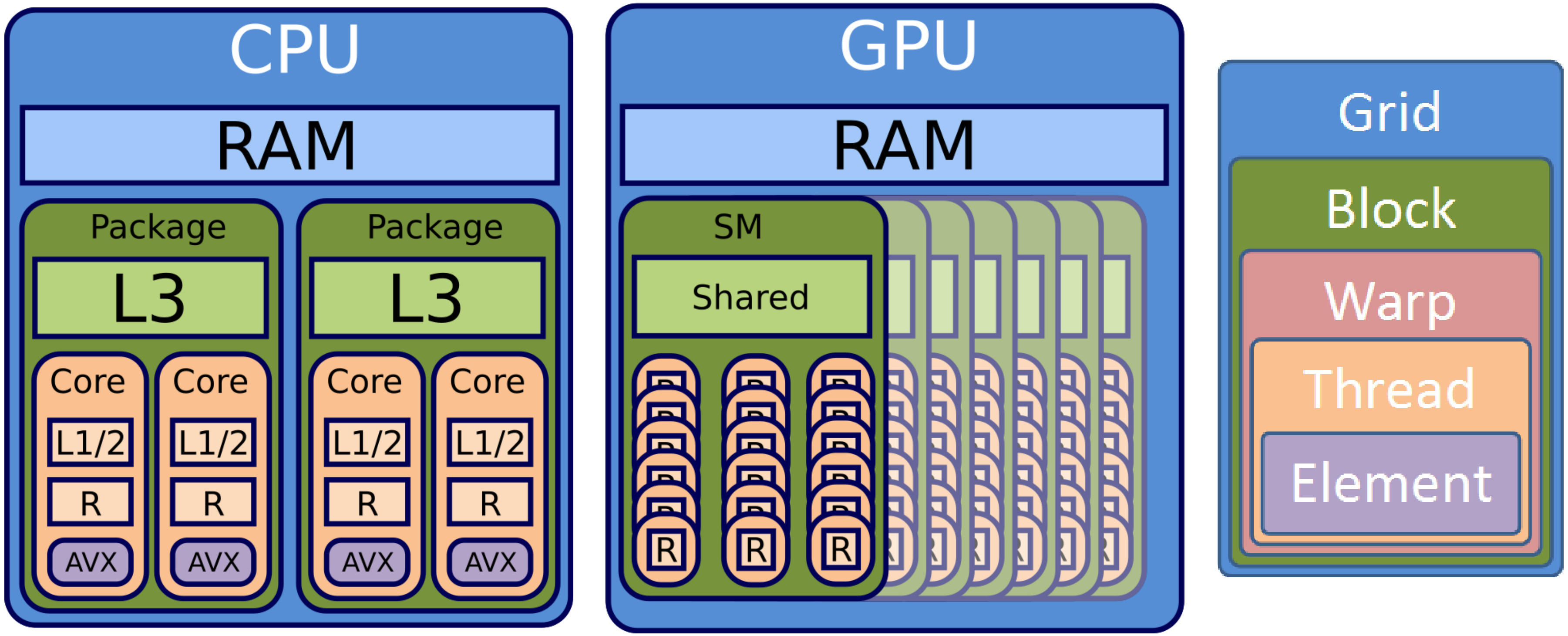
FUTURE:

VC GSI

C++ SIMD Vectorization & Packing

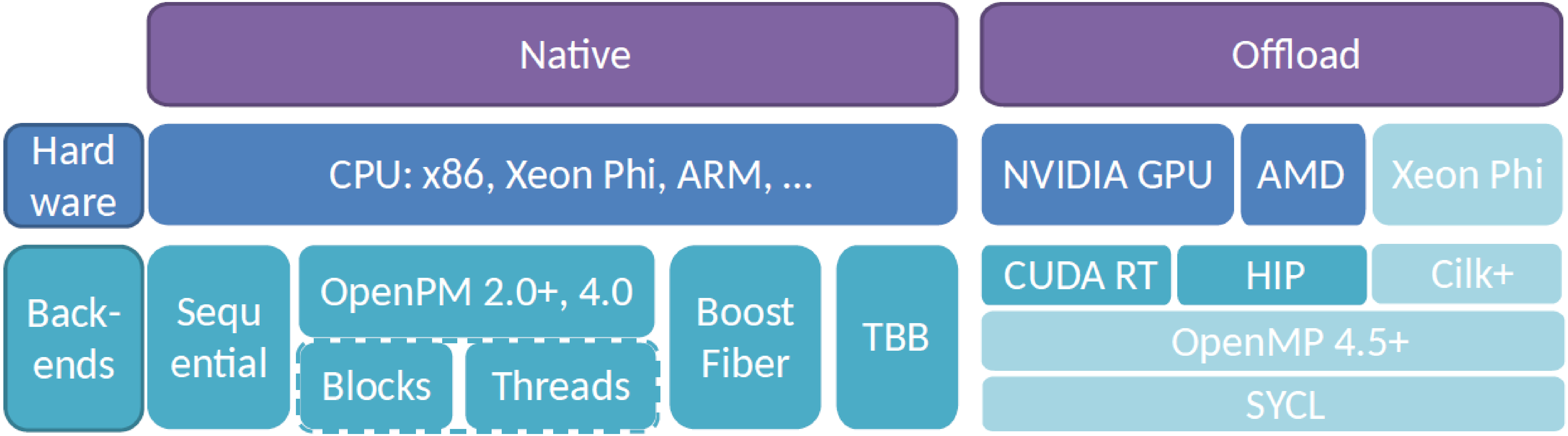
Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

Mapping the abstract hierarchy to real hardware



Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

Alpaka Backends



Abstraction Library for Parallel Kernel Acceleration

Memory allocation and kernel call

```
// Init Host
using Host = alpaka::acc::AccCpuSerial< Dim, Size >;
using DevHost = alpaka::dev::Dev< Host >;
using PltfHost = alpaka::pltf::Pltf< DevHost >;

// Memory allocation
auto X_h = alpaka::mem::buf::alloc<float, Size>( devHost, extent );
auto X_d = alpaka::mem::buf::alloc<float, Size>( devAcc, extent );

// Copy from host to device
alpaka::mem::view::copy(stream, X_d, X_h, extent);

// Kernel creation and execution
VectorAdd kernel;
auto const exec( alpaka::exec::create< Acc >(
    workDiv,
    kernel,
    numElements,
    alpaka::mem::view::getPtrNative(X_d),
    alpaka::mem::view::getPtrNative(Y_d)
));
alpaka::stream::enqueue( stream, exec );
```

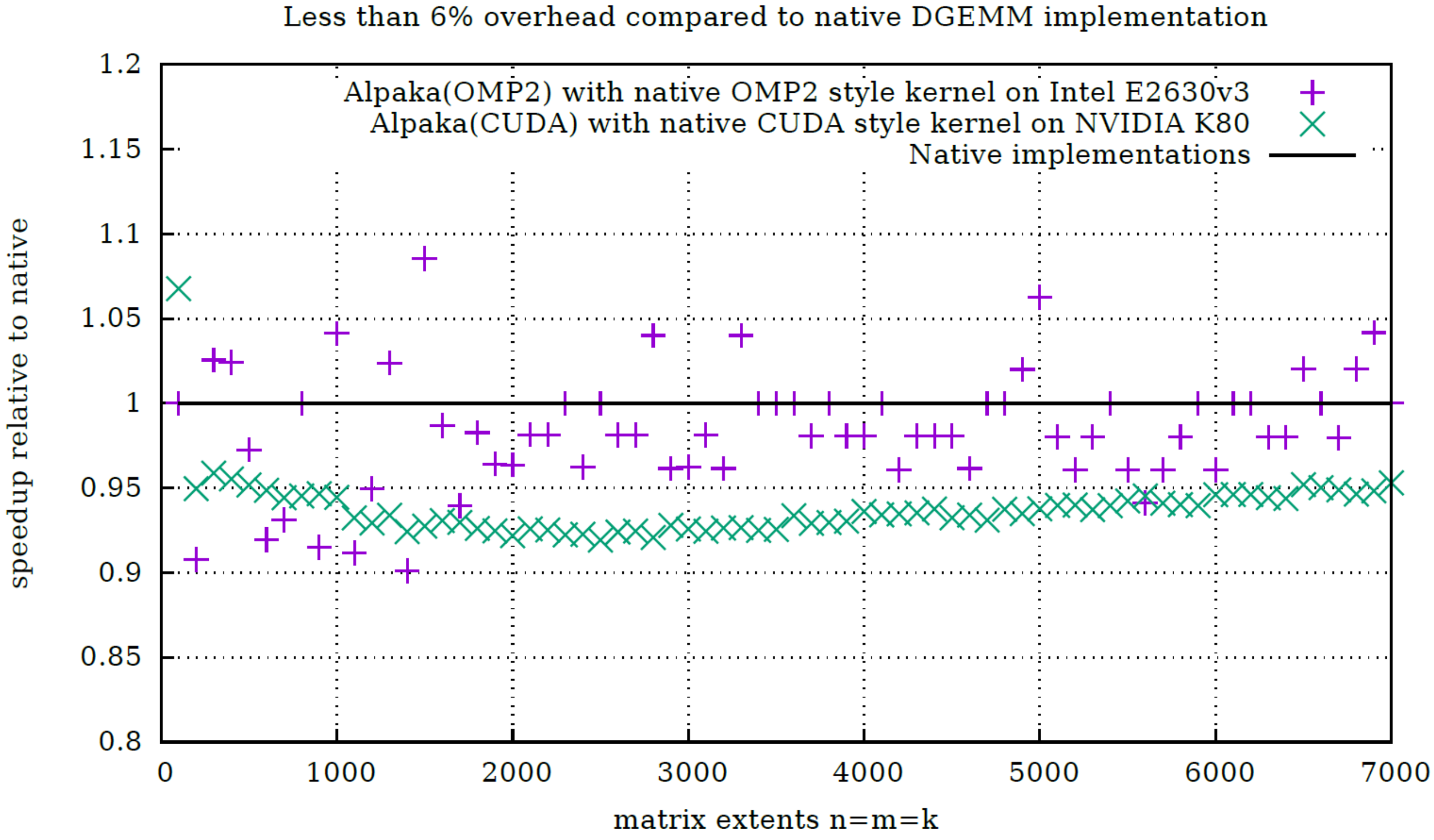
Abstraction Library for Parallel Kernel Acceleration

SIMD optimized vector addition

```
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
    }
};
```

Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

Zero overhead (DGEMM)



Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

Zero overhead (Vector Addition)

Alpaka CUDA PTX

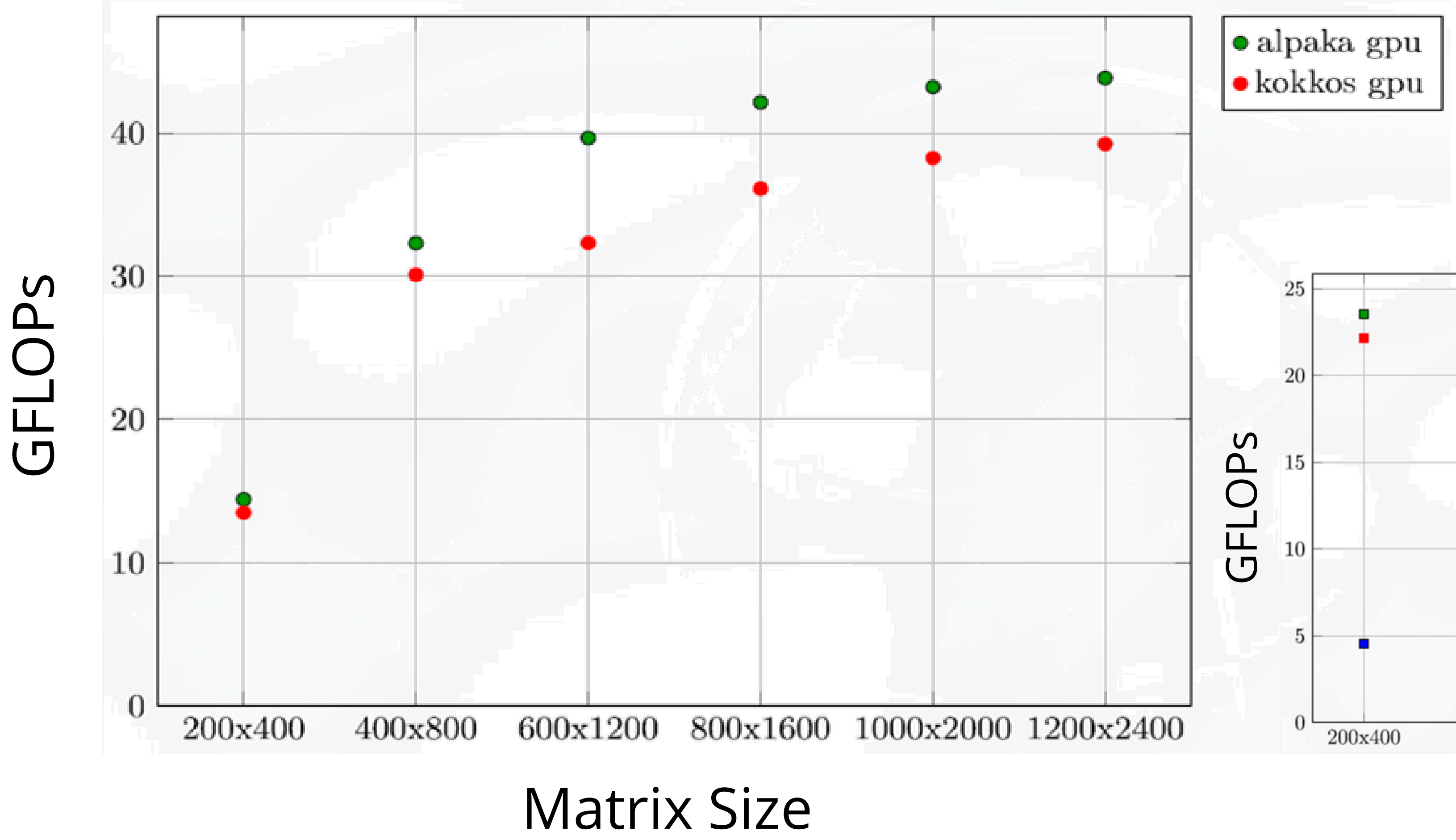
```
mov.u32      %r3, %ctaid.x;
mov.u32      %r4, %ntid.x;
mov.u32      %r5, %tid.x;
mad.lo.s32   %r1, %r4, %r3, %r5;
setp.ge.s32 %p1, %r1, %r2;
@%p1 bra   BB6_2;


cvta.to.global.u64  %rd3, %rd2;
cvta.to.global.u64  %rd4, %rd1;
mul.wide.s32        %rd5, %r1, 8;
add.s64             %rd6, %rd4, %rd5;
ld.global.f64       %fd2, [%rd6];
add.s64             %rd7, %rd3, %rd5;
ld.global.f64       %fd3, [%rd7];
fma.rn.f64          %fd4, %fd2, %fd1, %fd3;
st.global.f64      [%rd7], %fd4;
```

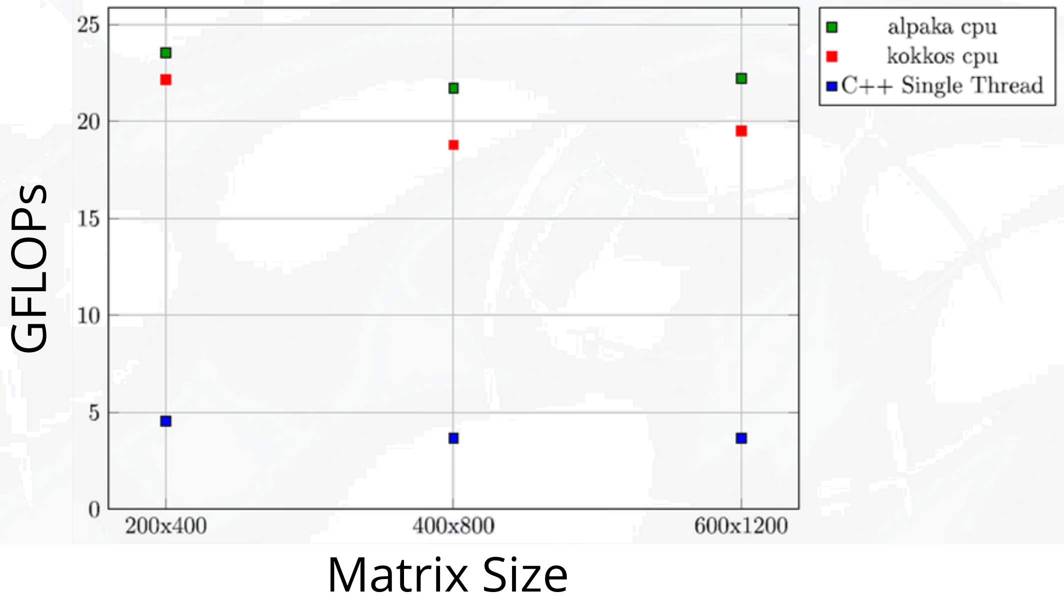
Native CUDA PTX

```
mov.u32      %r3, %ctaid.x;
mov.u32      %r4, %ntid.x;
mov.u32      %r5, %tid.x;
mad.lo.s32   %r1, %r4, %r3, %r5;
setp.ge.s32 %p1, %r1, %r2;
@%p1 bra   BB6_2;


cvta.to.global.u64  %rd3, %rd2;
cvta.to.global.u64  %rd4, %rd1;
mul.wide.s32        %rd5, %r1, 8;
add.s64             %rd6, %rd4, %rd5;
ld.global.nc.f64    %fd2, [%rd6];
add.s64             %rd7, %rd3, %rd5;
ld.global.f64       %fd3, [%rd7];
fma.rn.f64          %fd4, %fd2, %fd1, %fd3;
st.global.f64      [%rd7], %fd4;
```

Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

Heat diffusion simulation





Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

CUPLA – CUDA2ALPAKA

```
#include <cuda_runtime.h>
```

↓

```
#include <cuda_to_cupla.h>
```

```
kernel<<< blocks, threads >>>( elems, n, x_d, y_d );
```

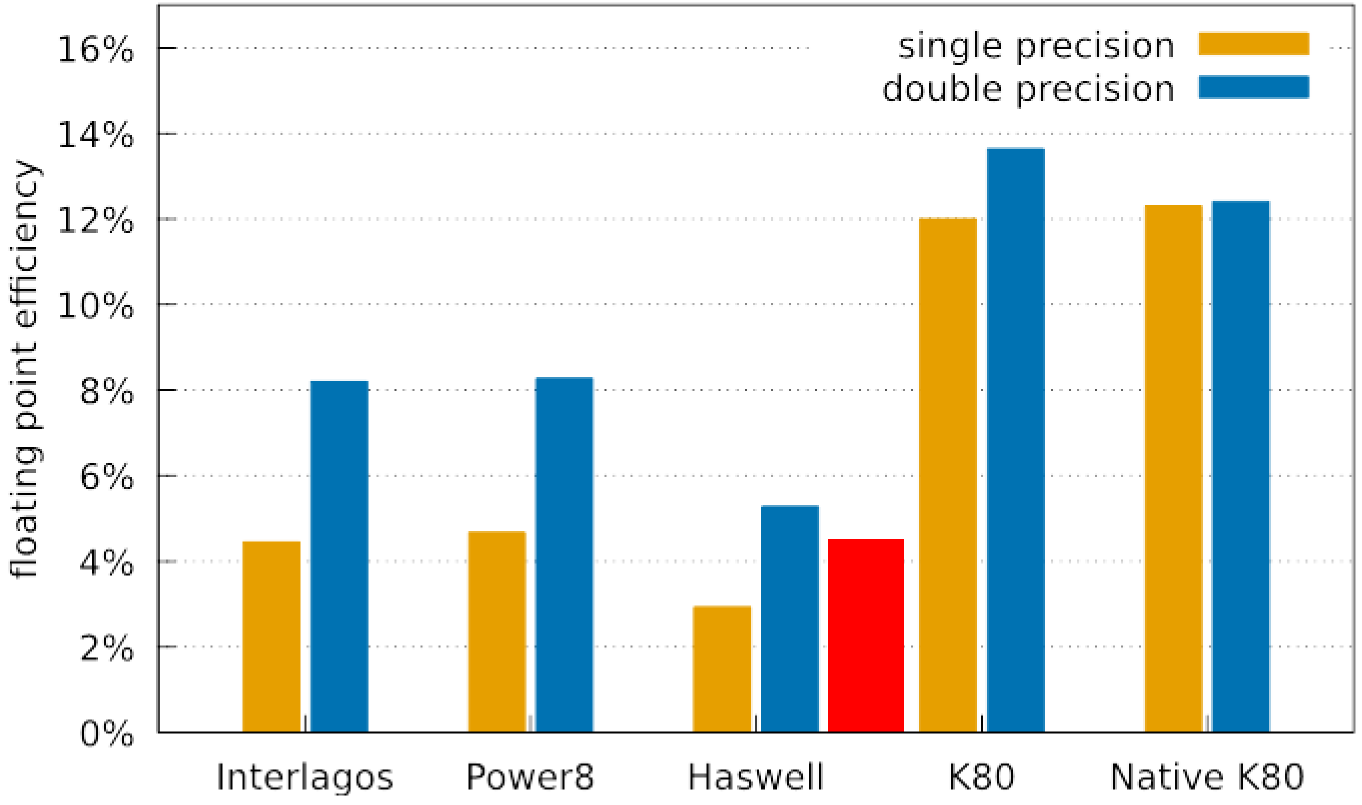
↓

```
CUPLA_KERNEL_ELEM(kernel)( blocks, threads, elems )( n, x_d, y_d );
```

Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

CUPLA — PIConGPU Plasma Simulation

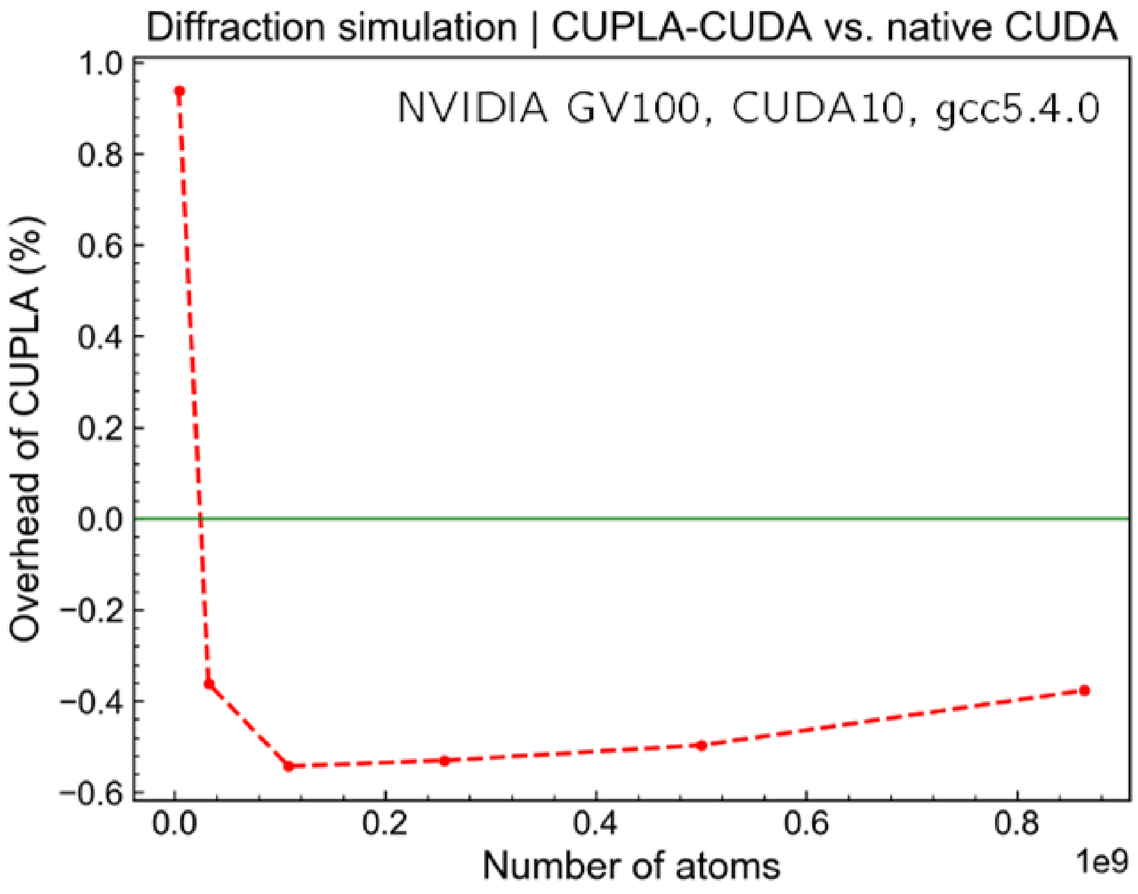
Before: PIConGPU + PMacc 80k LOC (20k in kernels)
After: 50k LOC (1 year)

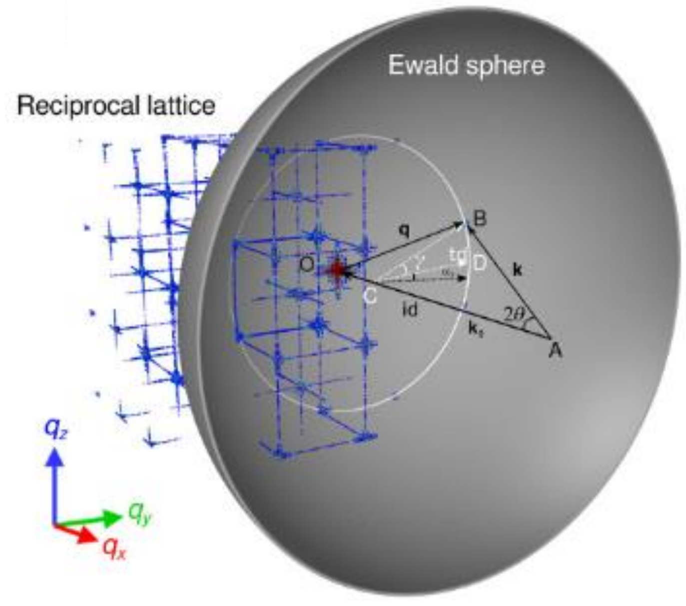


René Widera porting 80k LOC in 3 weeks

Abstraction Library for **Pa**rallel **K**ernel **A**cceleration

CUPLA ─ GAPD Diffraction Simulation





In-memory coupling of two Alpaka-fied codes

In-memory workflow coupling

openPMD Eco-System

github.com/openPMD/openPMD-projects

openPMD standard (1.0.0, 1.0.1, 1.1.0)
the underlying file markup and definition
A Huebl et al., doi: 10.5281/zenodo.33624

base standard
general description

extensions
domain-specific

e.g. ED-PIC, SpeciesType, BeamPhysics

native data tools
HDF5, ADIOS1/2, NetCDF, ...
e.g. h5ls, h5repack, h5dump, bpdump

writers & converters
simulations, frameworks, measurements
e.g. PIConGPU, Warp, SIMEX_Platform

HDF Compass
HDF5 & ADIOS file explorer
open and explore file trees

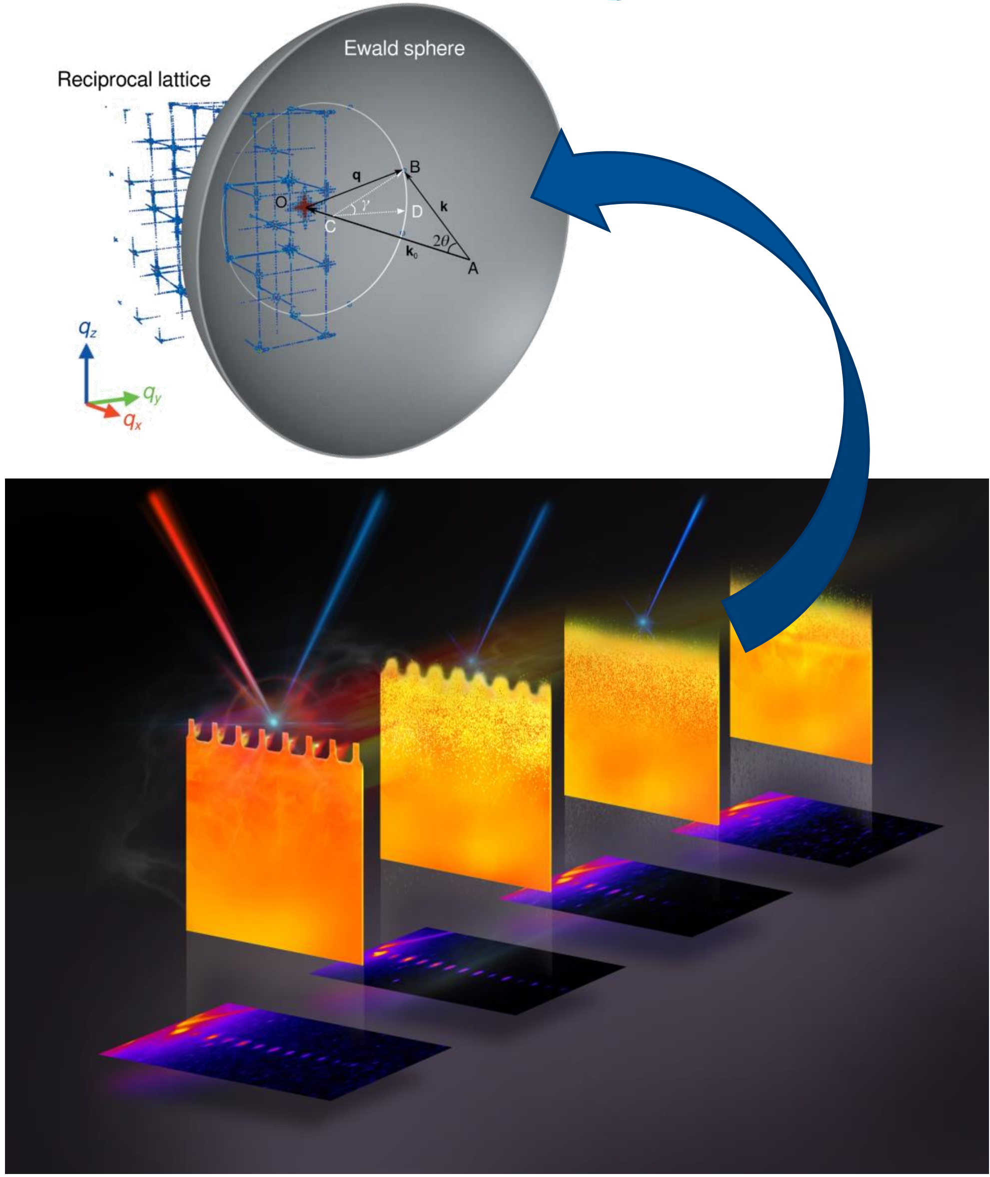
readers
coupled simulations, post-processing frameworks, ...
e.g. SIMEX_Platform, VisIt, yt-project, openPMD-viewer

openPMD-updater
update to new standard
edit in- or new file

openPMD-api
I/O library abstraction
file format agnostic

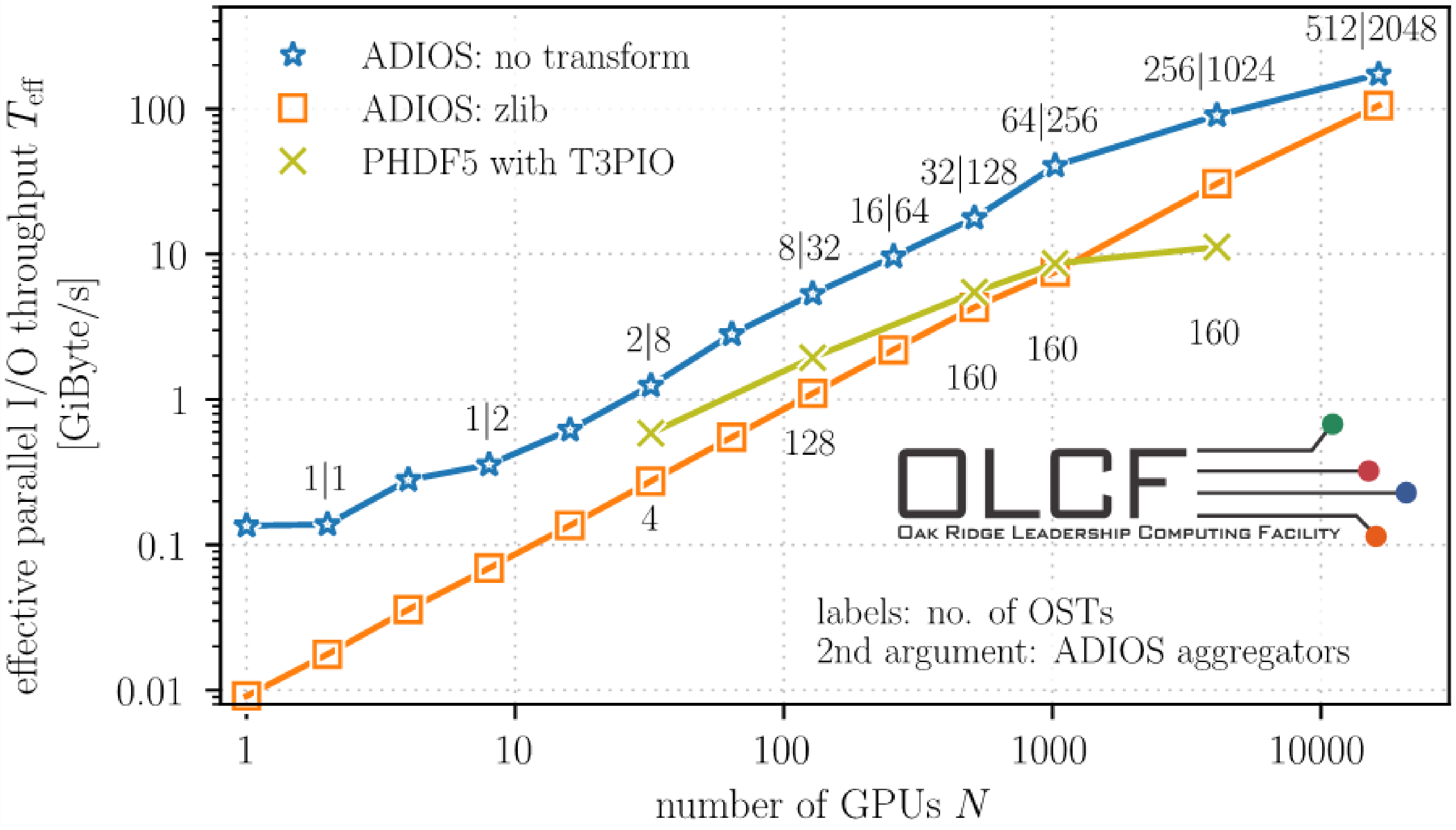
data repositories
exchange and long-time archival
e.g. Zenodo, RODARE (HZDR)

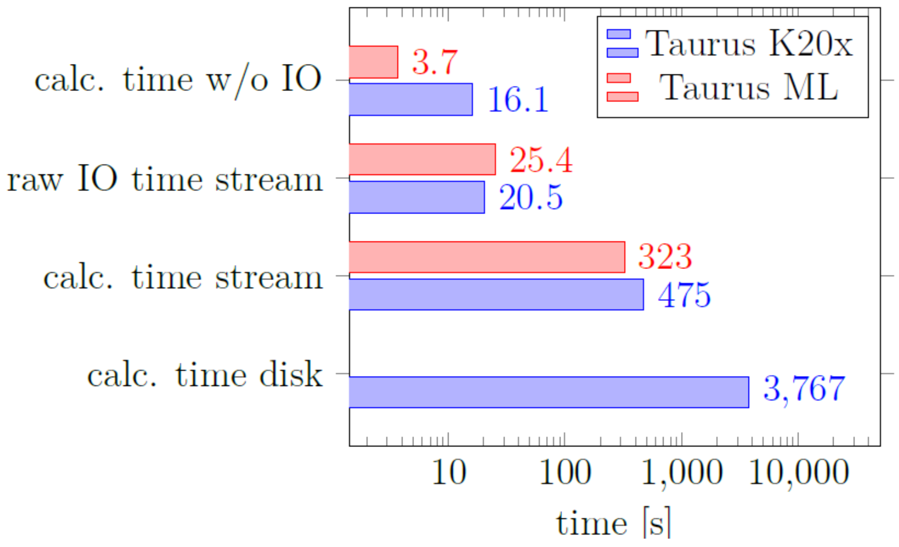
easy-to-use, fast, scalable, and portable I/O

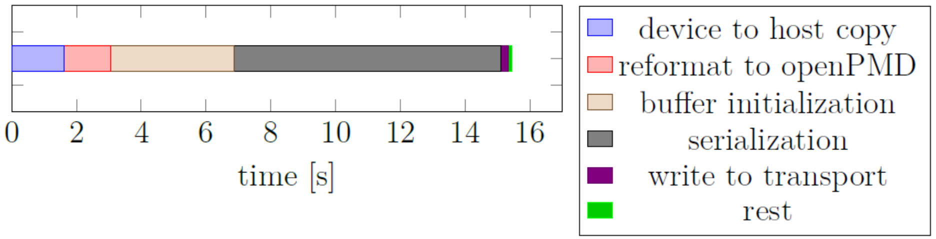


The bandwith hierarchy is killing us

In-memory workflow coupling



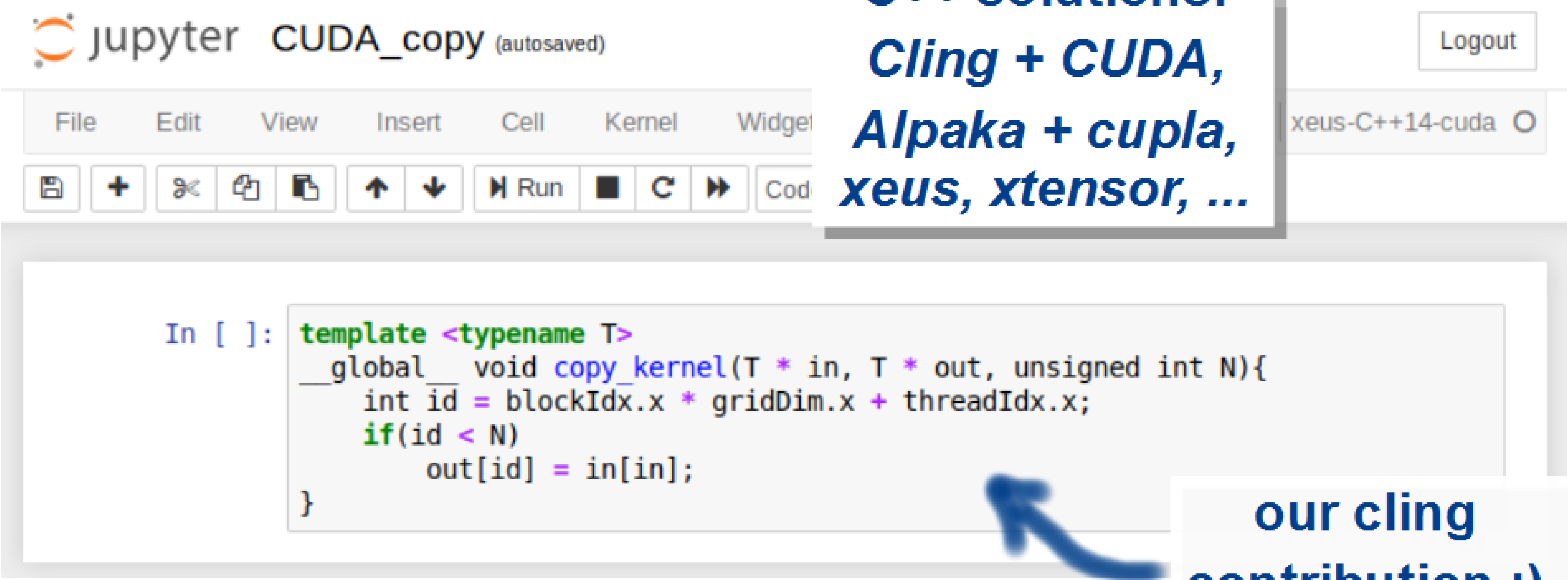




C++ JIT compilation and Jupyter Notebook integration

Cling and clang for Python-like C++ with GPUs and more

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



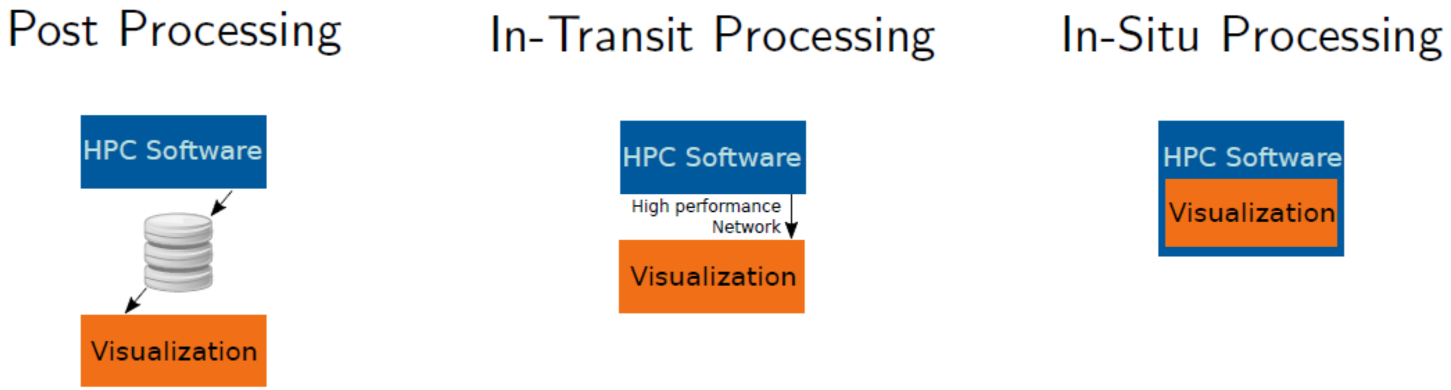
our cling contribution :)

https://developer.nvidia.com/gtc/2020/video/s21588

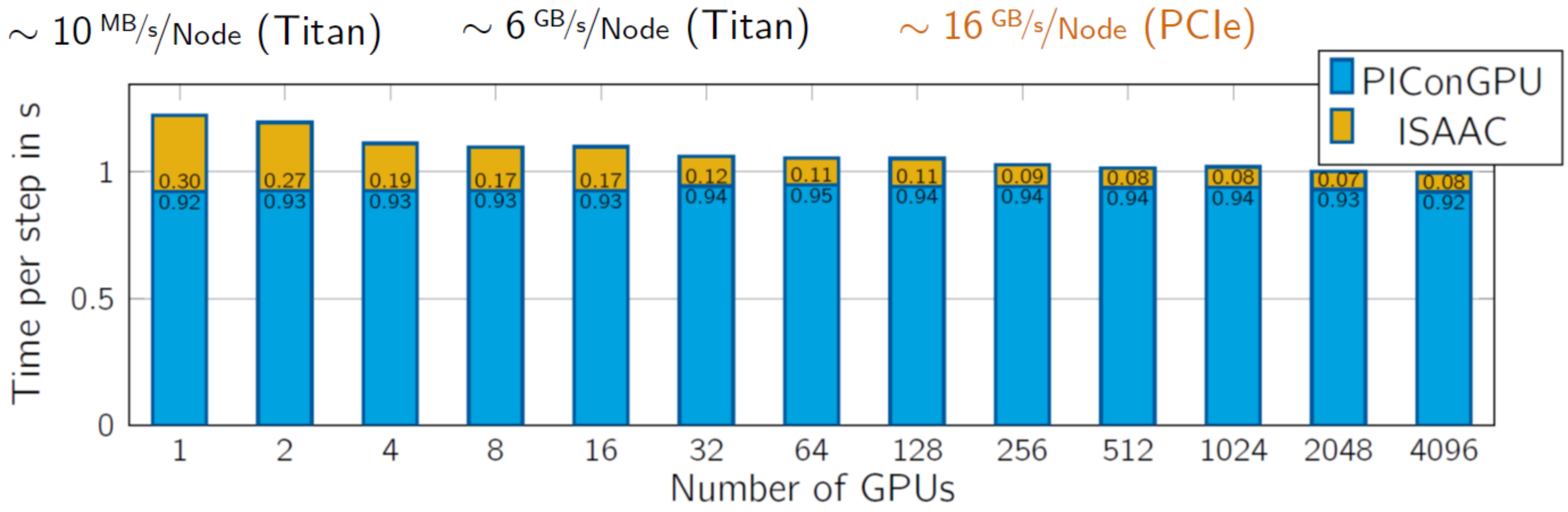
Strongly-coupled visualization of data with ISAAC

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

Visual analytics combined with immersive UI, ML & Feedback



ISAAC



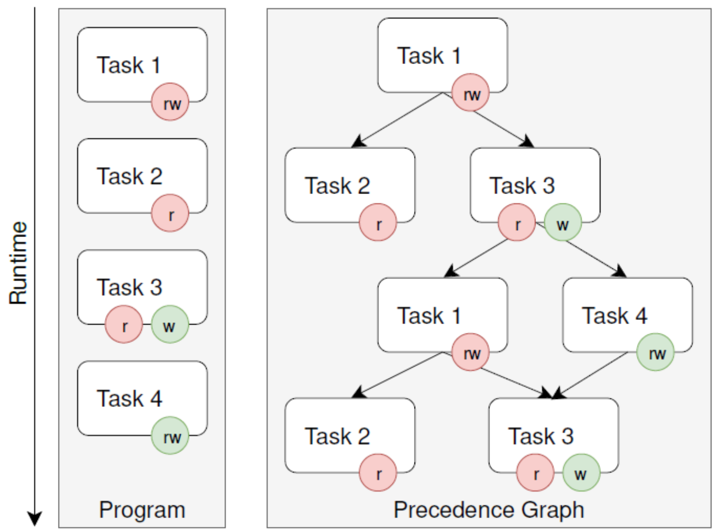
Next up: Creating task graphs from data dependencies

REsource-based, **D**eclarative task-**GRA**phs for **P**arallel, **E**vent-driven **S**cheduling

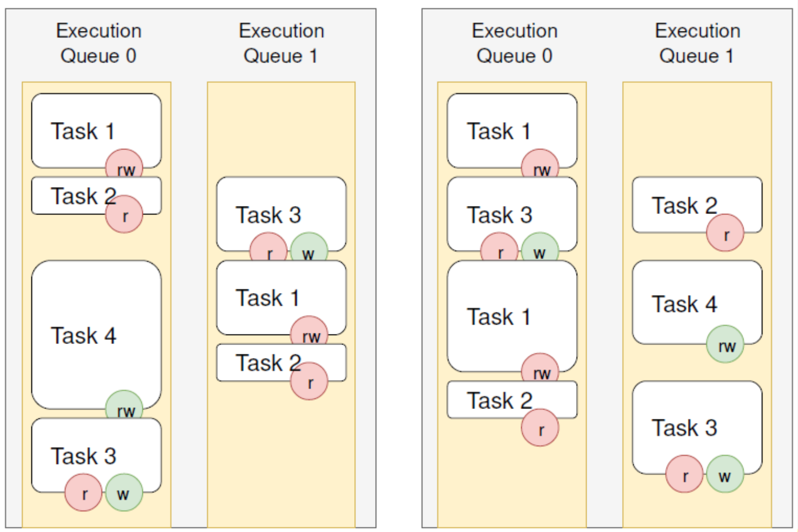
Example Code

```cpp
rg::IOResource< int > a, b;

for( ... ) {
    task([]( auto a ){ *a = 2; },
        a.write());
    task([]( auto a ){ printf("%d", *a); },
        a.read());
    task([]( auto a, auto b ){ *b = *a; },
        a.read(),
        b.write());
    task([]( auto b ){ *b += 1; },
        b.write());
}
```

Declarative Task Dependencies

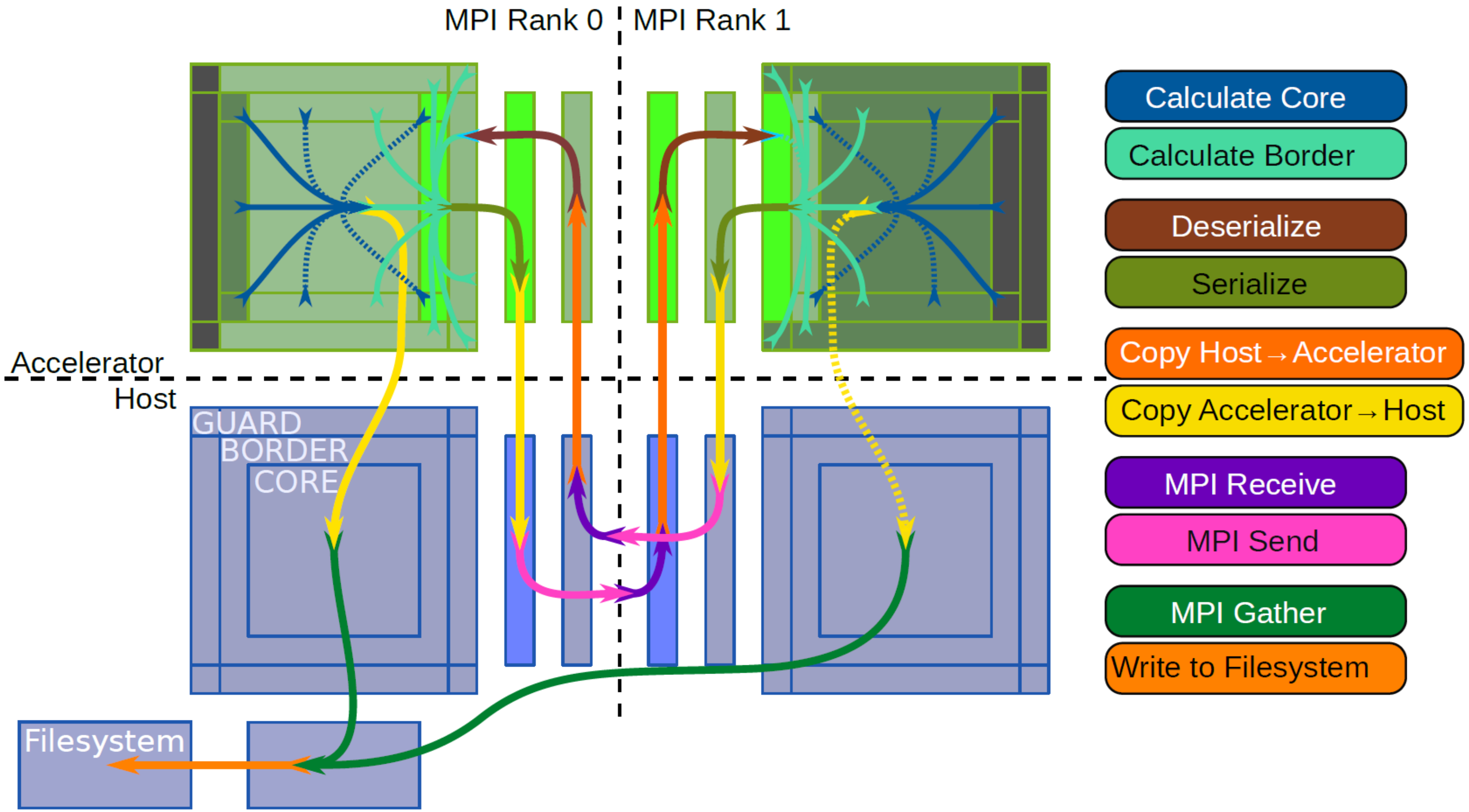


Possible Schedules



Next up: Creating task graphs from data dependencies

redGrapes — Data flow much more complex than data dependencies



Next up: Parallelism needs performant memory access

Low **L**evel **A**bstraction of **M**emory **A**ccess

```
struct {
  float x,y;
} Pos;
Pos pos[8];
```

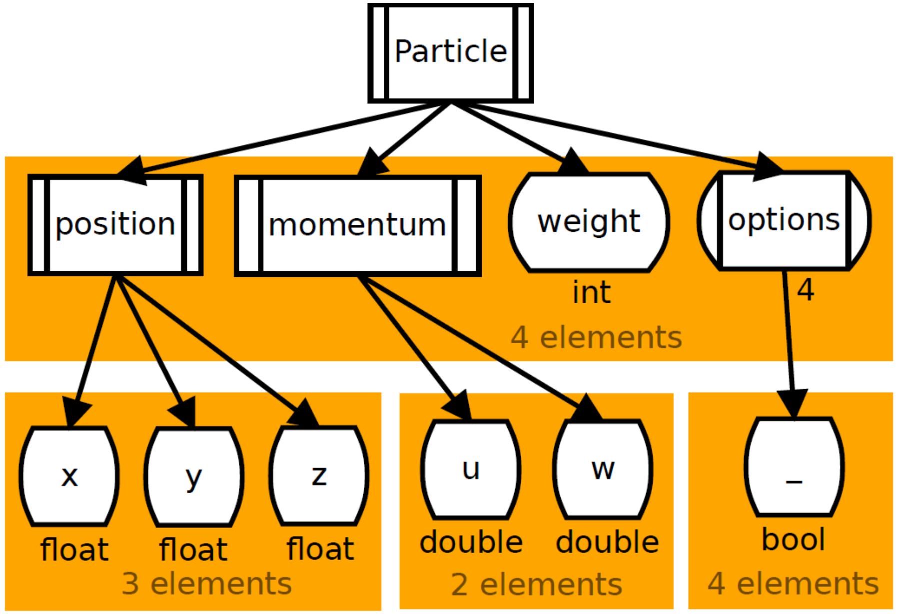
user code

x	y	x	y
x	y	x	y
x	y	x	y
x	y	x	y

user view

x	x	x	x
x	x	x	x
y	y	y	y
y	y	y	y

memory

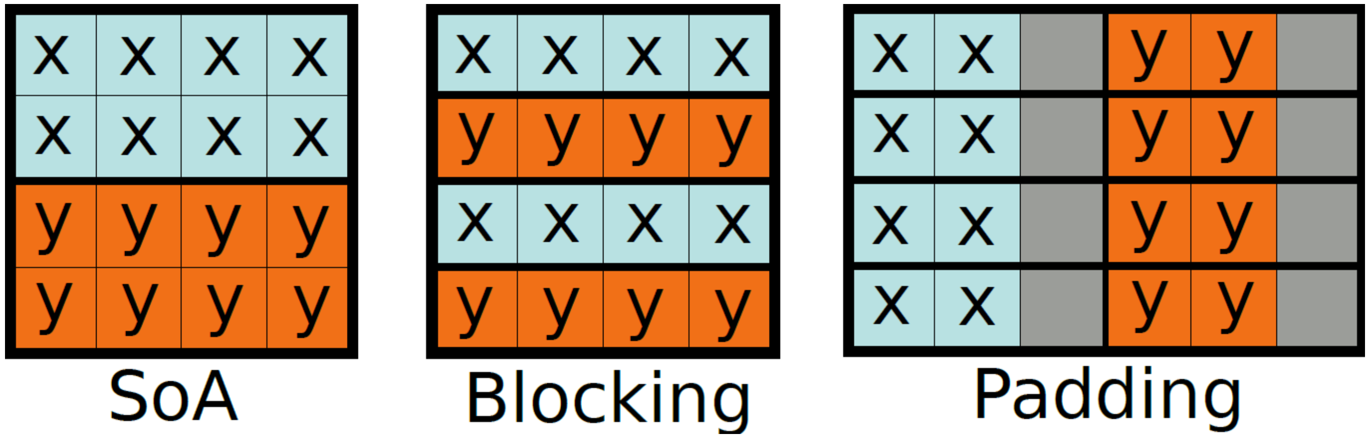


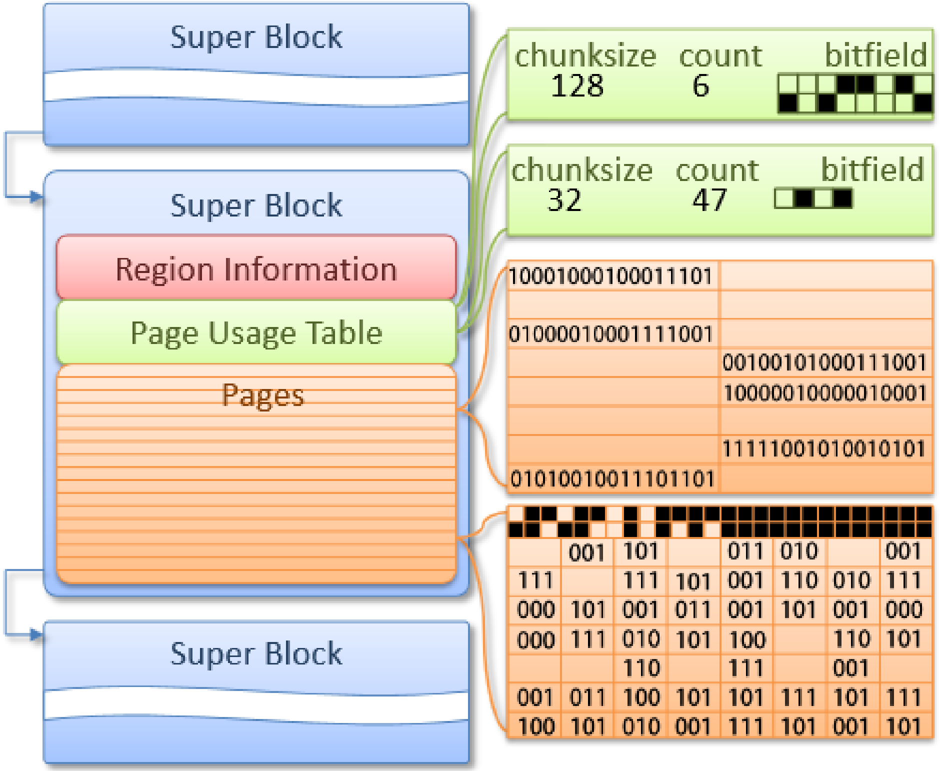
4 hierarchical Element Domains

ROOT
Data Analysis Framework

Next up: Parallelism needs performant memory access

Parallel object-like memory allocation & optimized deep copies

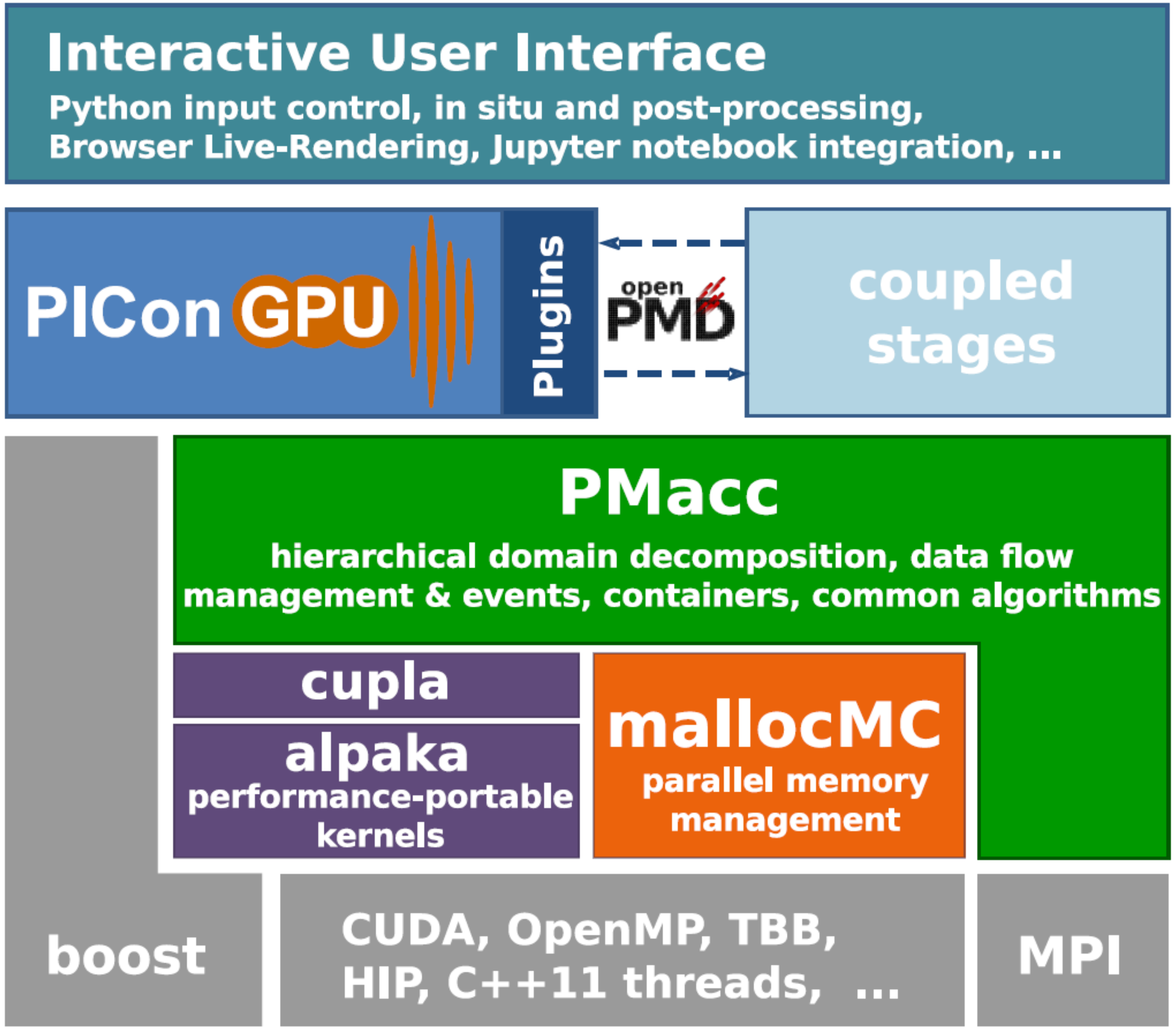




„mallocMC“

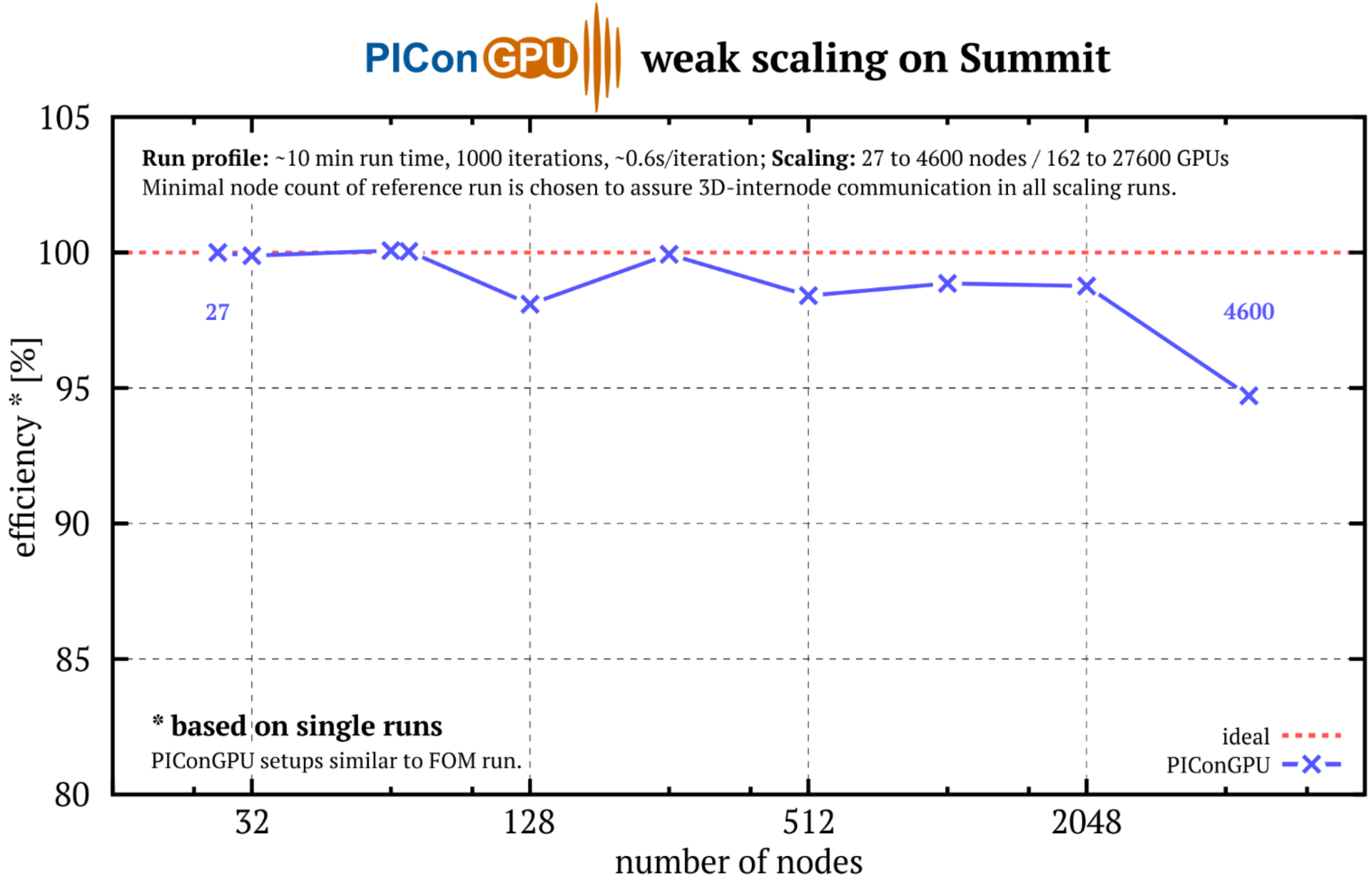
Modularizing code becomes more important

Exascale programming is not and should not be for everyone



When going to Exascale, take babysteps inbetween

Using Summit/ORNL as a testbed



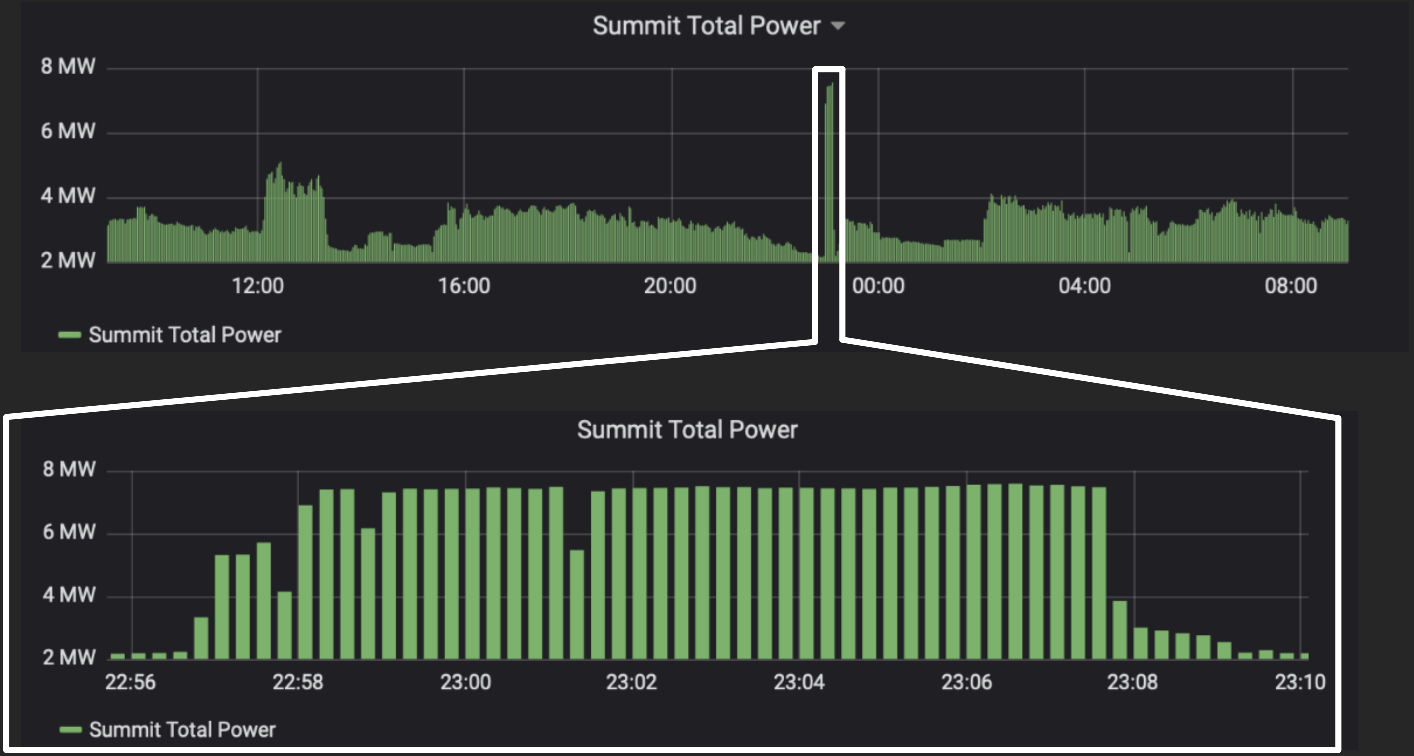
Weak scaling from 27 nodes to 4600 on the #1 HPC system Summit

When going to Exascale, take babysteps inbetween

But think before you simulate

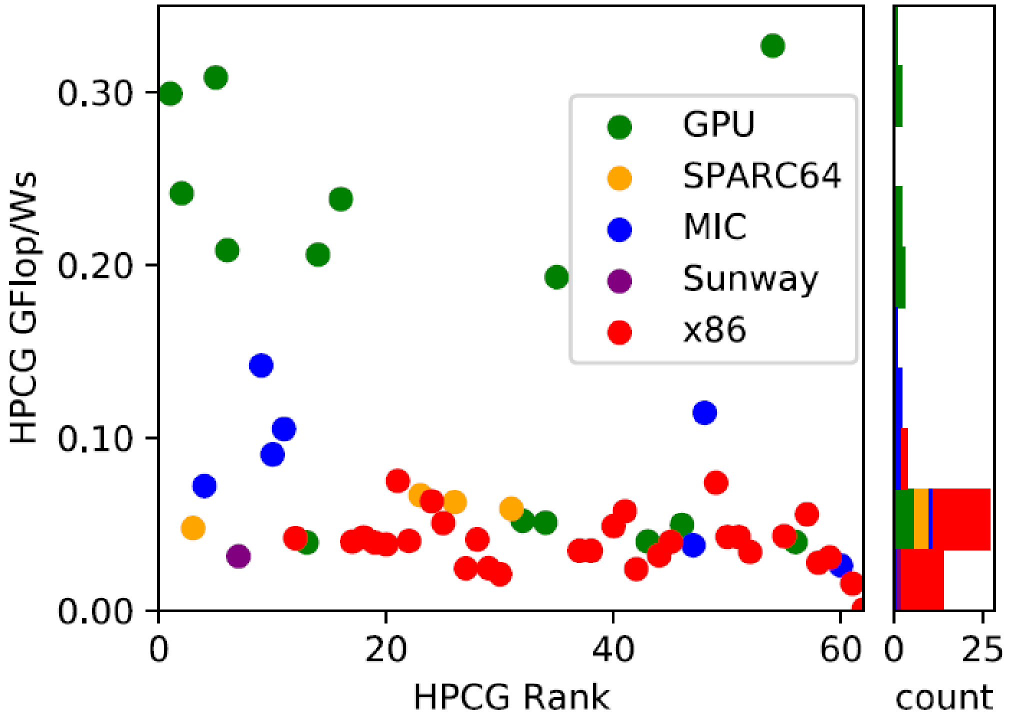
PIConGPU on FULL Summit
Peak power: **8MW**
Sustained power: **5.8MW**

```
# NODES        :    4600
# GPUS         :   27600
# particles    : 1.01e13
# cells        : 4.04e11
```



When going to Exascale, take babysteps inbetween

GPUs are pretty cool

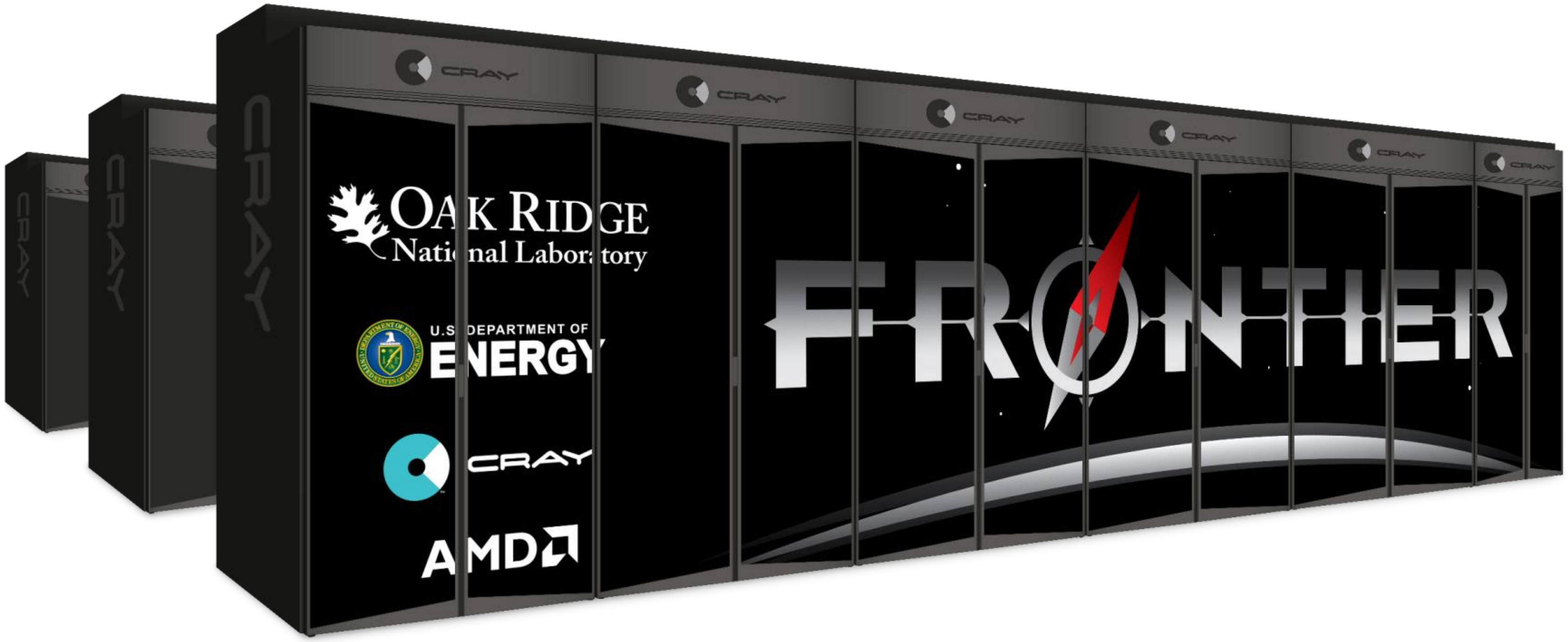


Specs	Volta	Ampére
FP32 cores	5120	6912
Memory BW	900 GB/s	1555 GB/s
VRAM	32 GB	40 GB
Interconnect	300 GB/s	600 GB/s

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Exascale System Readiness

ORNL Center for Accelerated Application Readiness (Exascale)



NVIDIA CUDA

GPUOpen

OpenMP

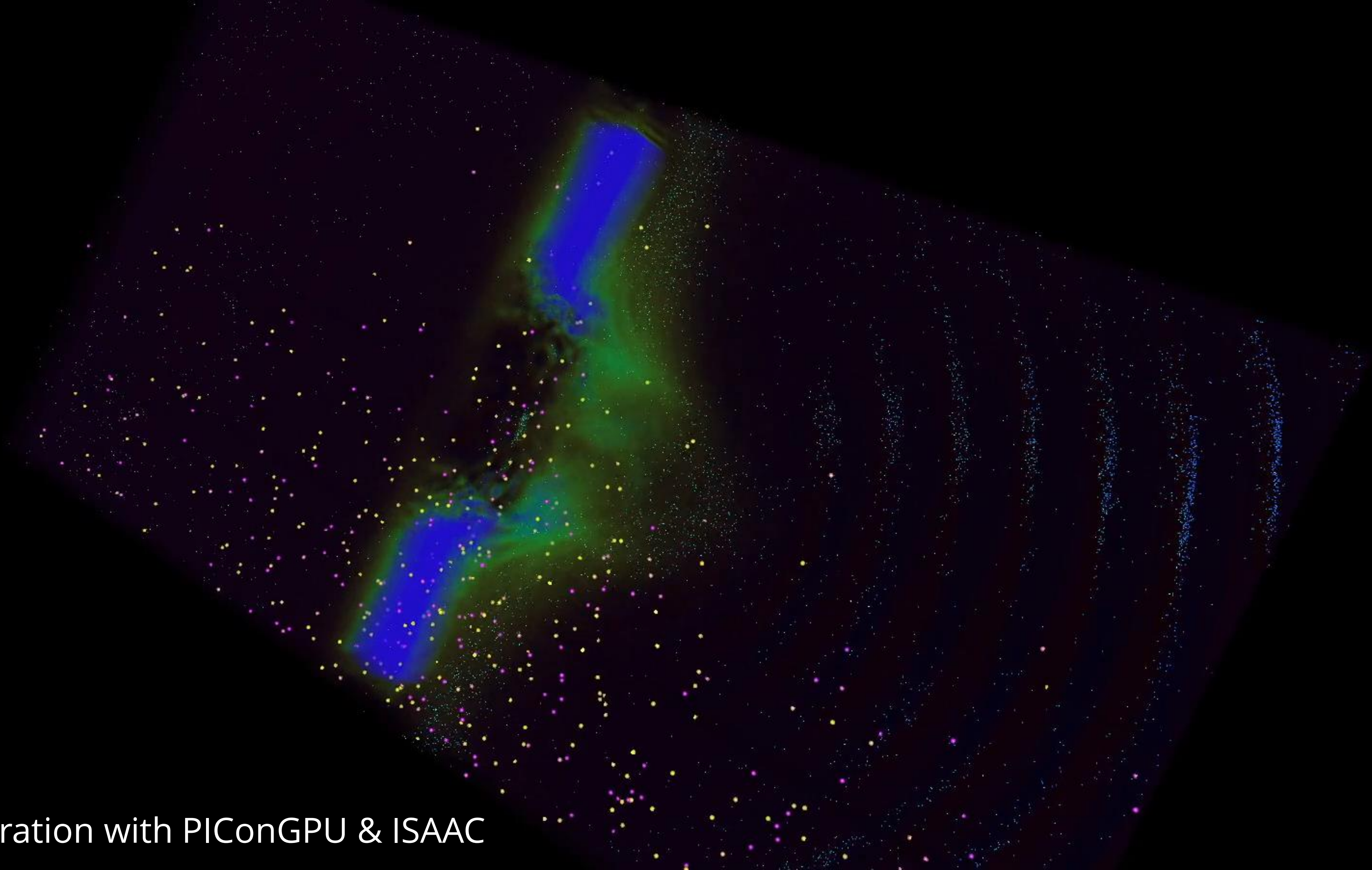
SYCL™

https://www.olcf.ornl.gov/caar/Frontier-CAAR/

PIConGPU alpaka

Exploring Petabytes in real time

Visual analytics combined with immersive UI, ML & Feedback



Laser-driven Ion Acceleration with PIConGPU & ISAAC

Make your code Exascale-ready

ISAAC

alpaka

cupla

LLAMA

openPMD

PIConGPU

Meet us on Github!

https://github.com/alpaka-group/alpaka

CASUS

CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN