CUDA® C++ in Jupyter: Adding CUDA Runtime Support to Cling

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

S. Ehrig[1] and A. Huebl[1,2]

[1] Helmholtz-Zentrum Dresden – Rossendorf, Germany
[2] Lawrence Berkeley National Laboratory, USA

NVIDIA GPU Technology Conference 2020

San Jose, March 26th 2020

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

Accelerated Simulations & Data Analytics

Novel Particle Accelerator Research

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

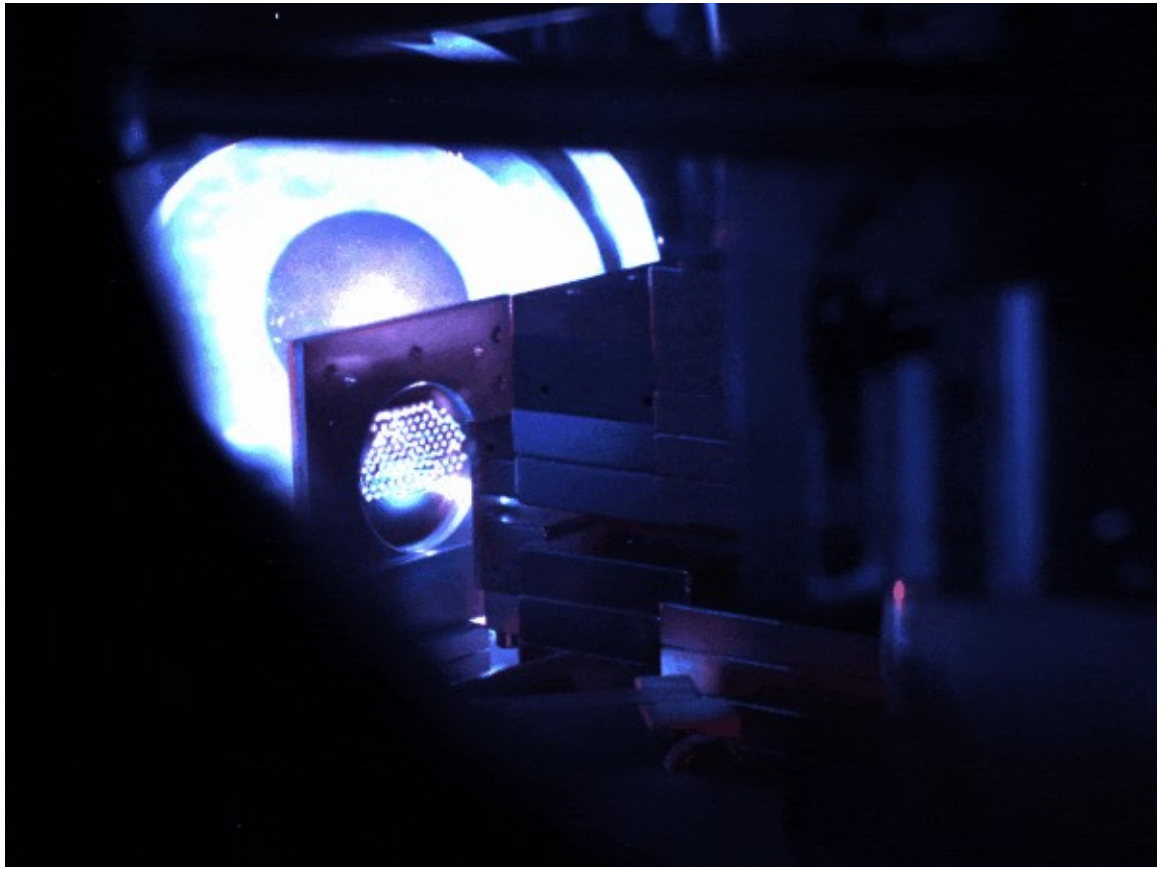
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ
Uniwersytet Wrocławski
NVIDIA GPU CENTER OF EXCELLENCE
DRESDEN concept excellence-dresden.de

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

Accelerated Simulations & Data Analytics

Novel Particle Accelerator Research

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science



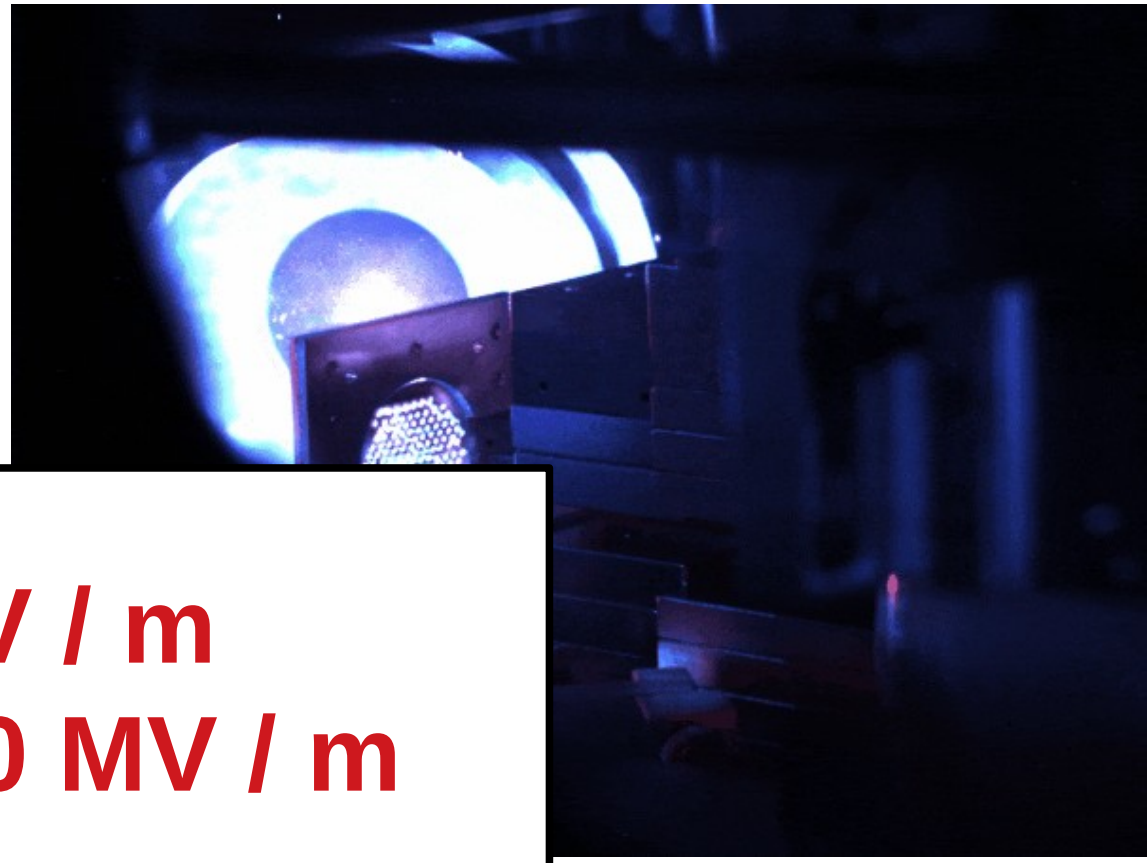
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ
Uniwersytet Wrocławski
NVIDIA GPU CENTER OF EXCELLENCE
DRESDEN concept excellence-dresden.de

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

Accelerated Simulations & Data Analytics

Novel Particle Accelerator Research

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science



~100 MV / m

→ 1 000 000 MV / m

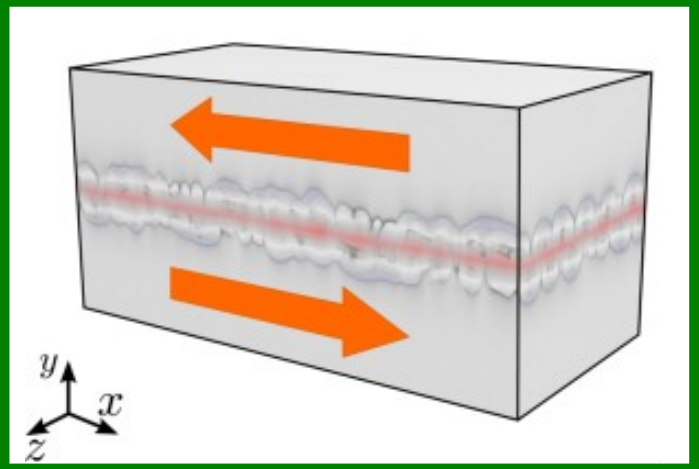
Laser-Plasma Simulations

Modeling Matter under Extreme Conditions

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

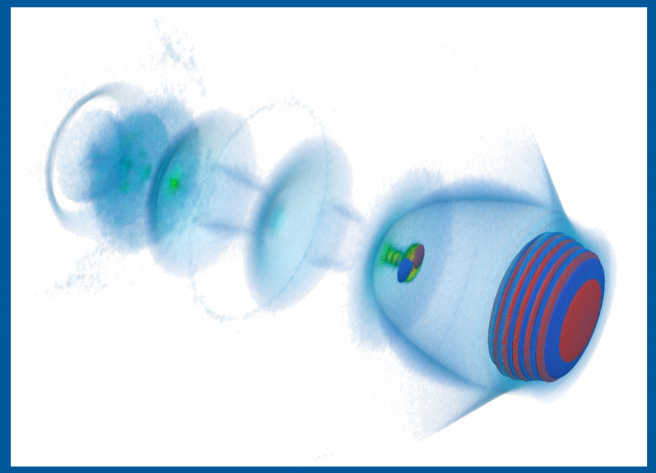
Astrophysics in the Lab

- Astrophysics
- Plasma instabilities



Novel Particle Accelerators

- Compact X-ray, electron and ion beam sources
- Pushing the Energy Frontier
- Matter at extreme conditions



Massively Parallel HPC

- Leadership-scale supercomputing
- Novel algorithms

69 PFlop/s (SP)

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

Data Driven Science at Exascale

The Widening Gap between Flop/s and I/O Bandwidth

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

SYSTEM SPECS	TITAN	SUMMIT	FRONTIER
Peak Performance	27 PF	200 PF	> 1.5 EF
		1/3x	1/3x
Storage	32 PB, 1 TB/s, Lustre Filesystem	250 PB, 2.5 TB/s, GPFS™	2-4x performance and capacity of Summit's I/O subsystem. Frontier will have near node storage like Summit.

Data Driven Science at Exascale

The Widening Gap between Flop/s and I/O Bandwidth

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

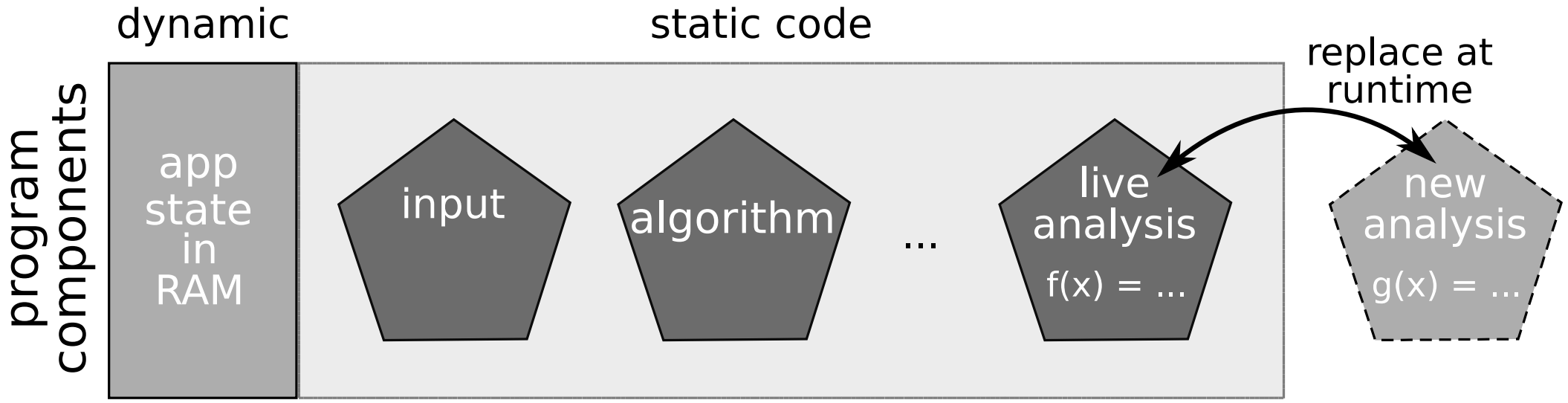
SYSTEM SPECS	TITAN	SUMMIT	FRONTIER
Peak Performance	27 PF	200 PF	> 1.5 EF
		1/3x	1/3x
Storage	32 PB, 1 TB/s, Lustre Filesystem	250 PB, 2.5 TB/s, GPFS™	2-4x performance and capacity of Summit's I/O subsystem. Frontier will have near node storage like Summit.

In situ approaches: **tightly** versus **loosely** coupled workflows

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

C++ *Feels* too Static

Data is Code is Data

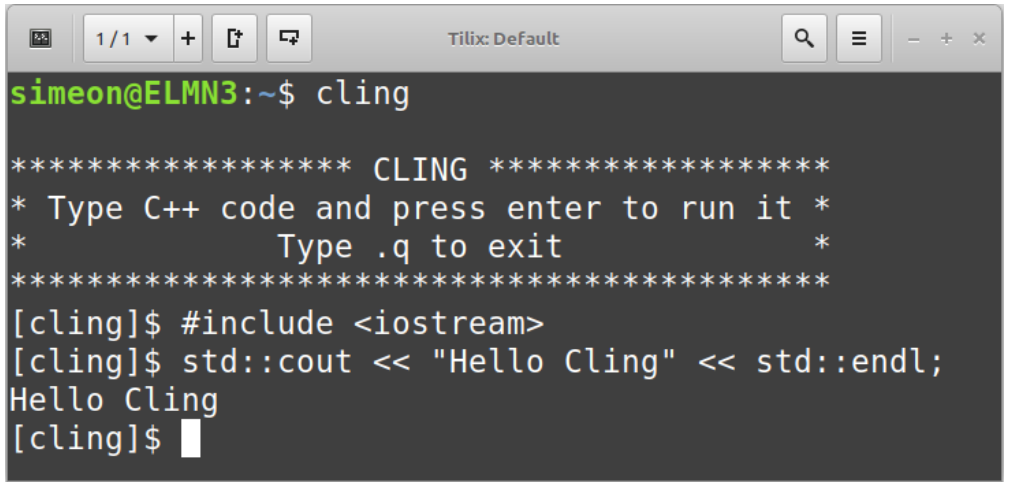


CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

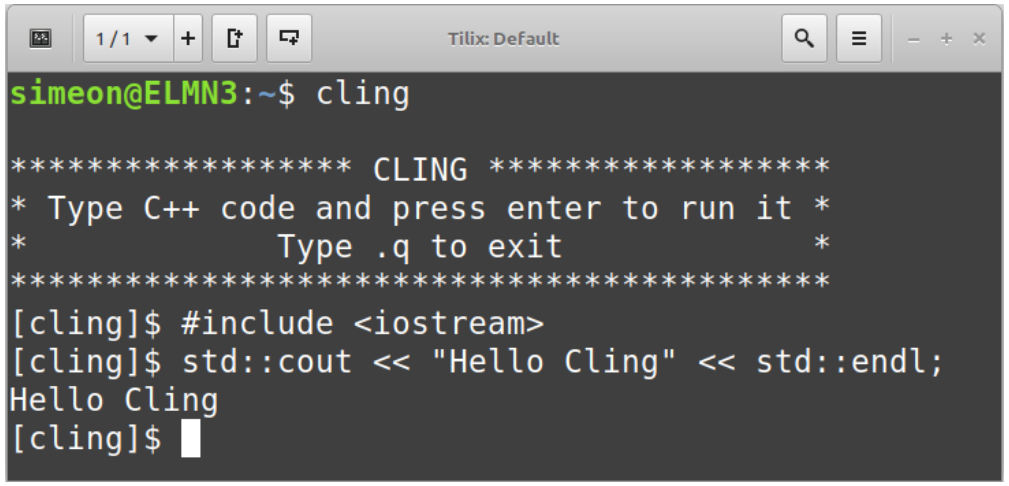
Interactive C++
An introduction to Cling

How to use Cling

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science



How to use Cling

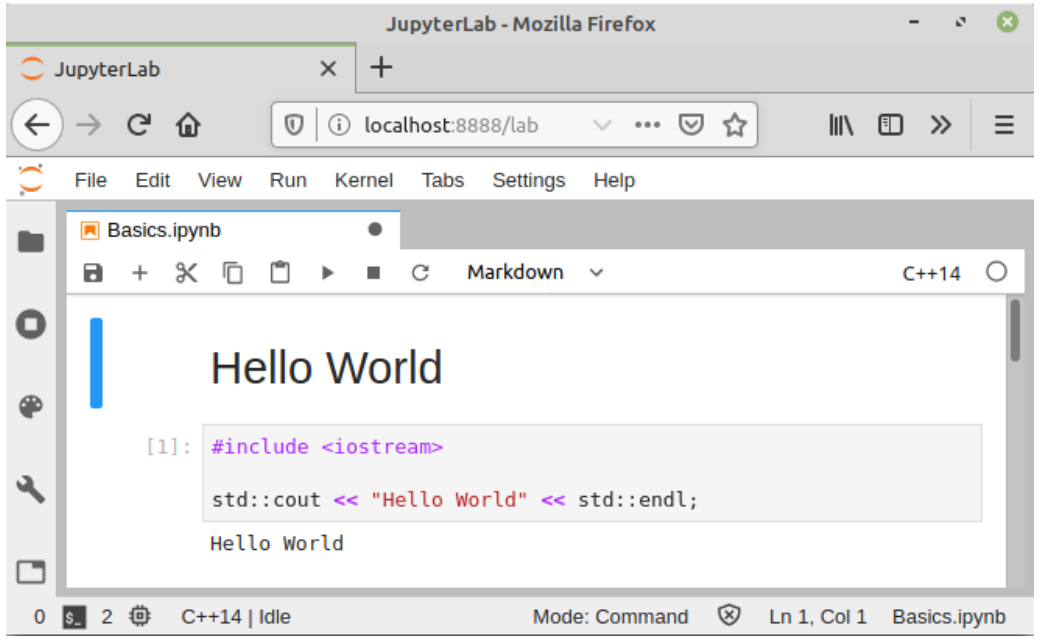



```
#include "cling/Interpreter/Interpreter.h"

int main(int argc, char *argv){
      auto cling = cling::Interpreter(argc, argv);
      return 0;
}
```

How to use Cling

```
simeon@ELMN3:~$ cling

****************** CLING ******************
* Type C++ code and press enter to run it *
*             Type .q to exit             *
*******************************************
[cling]$ #include <iostream>
[cling]$ std::cout << "Hello Cling" << std::endl;
Hello Cling
[cling]$
```

```
#include "cling/Interpreter/Interpreter.h"

int main(int argc, char *argv){
      auto cling = cling::Interpreter(argc, argv);
      return 0;
}
```



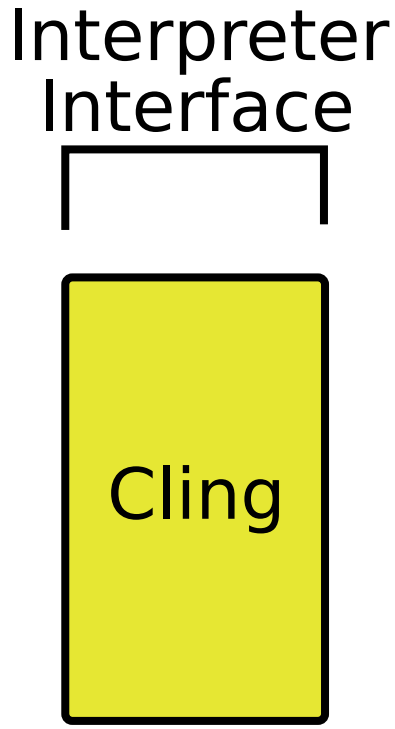
[Jupyter Live Demo]

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

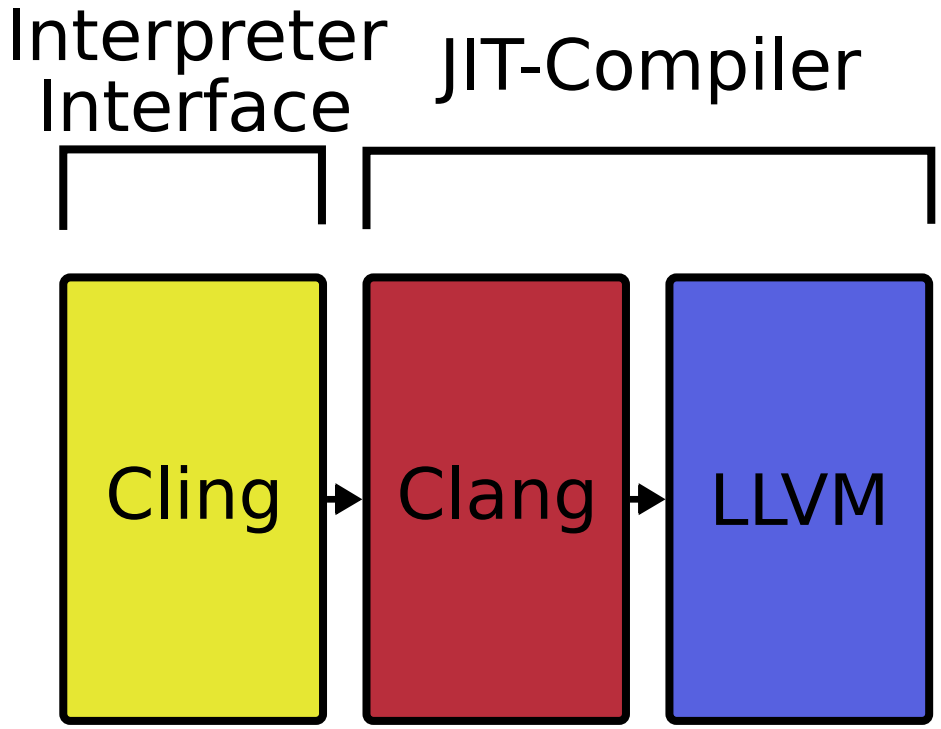
Architecture

Xeus-Cling Jupyter Architecture

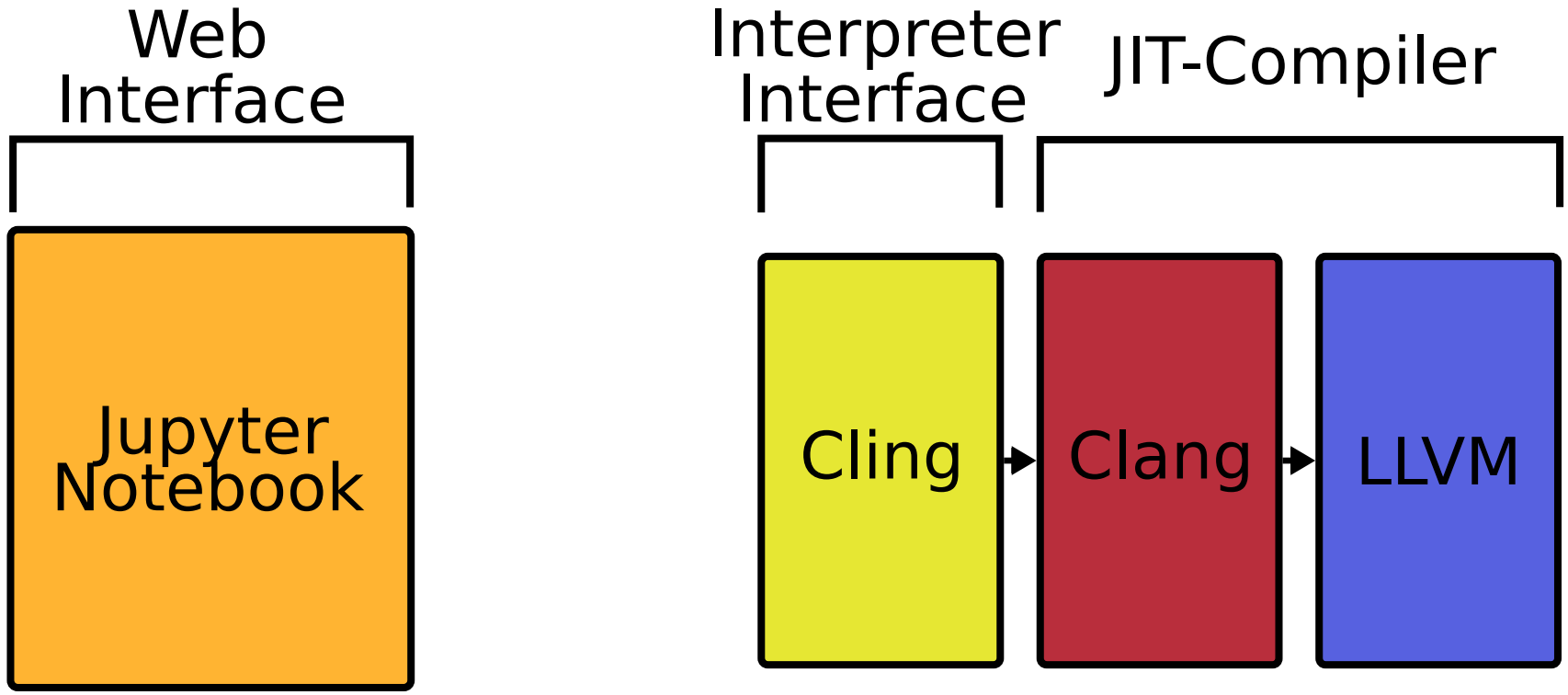
Xeus-Cling Jupyter Architecture



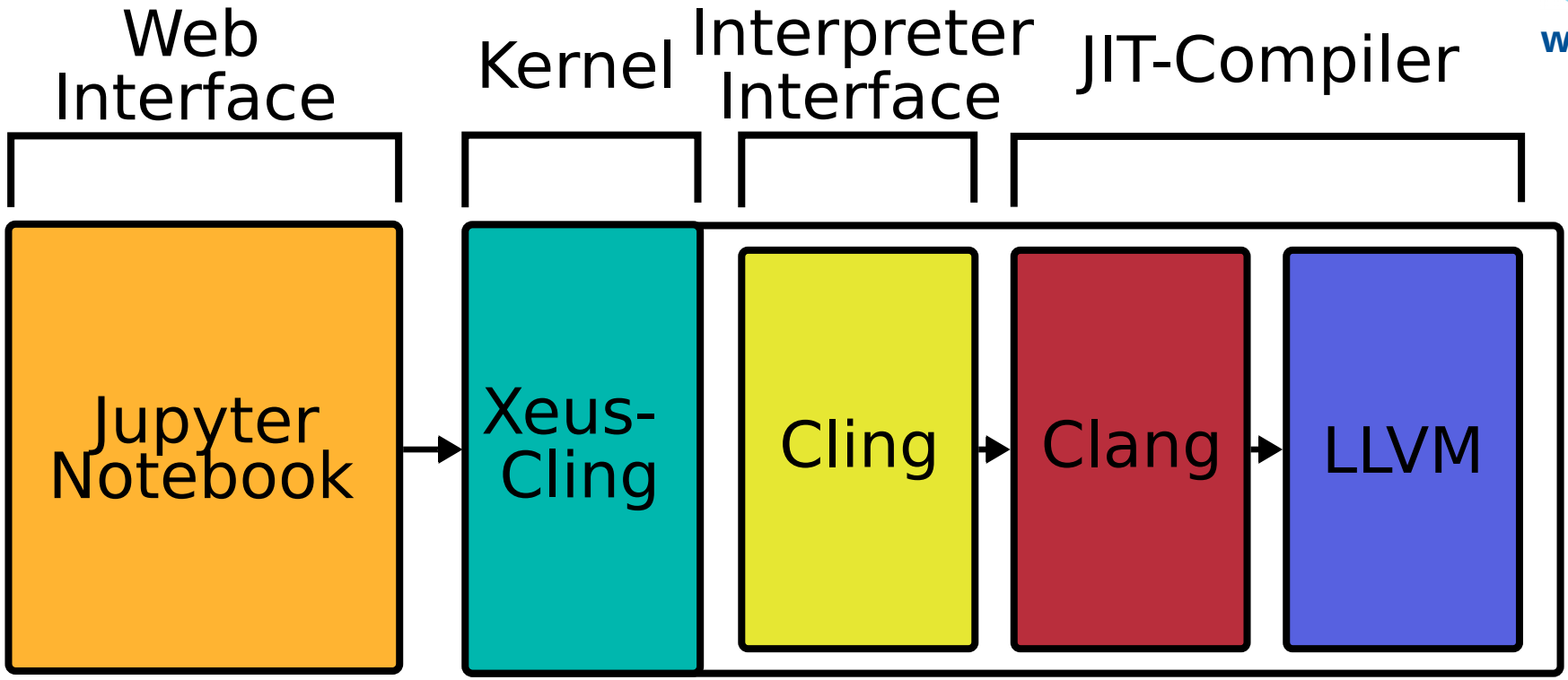
Xeus-Cling Jupyter Architecture



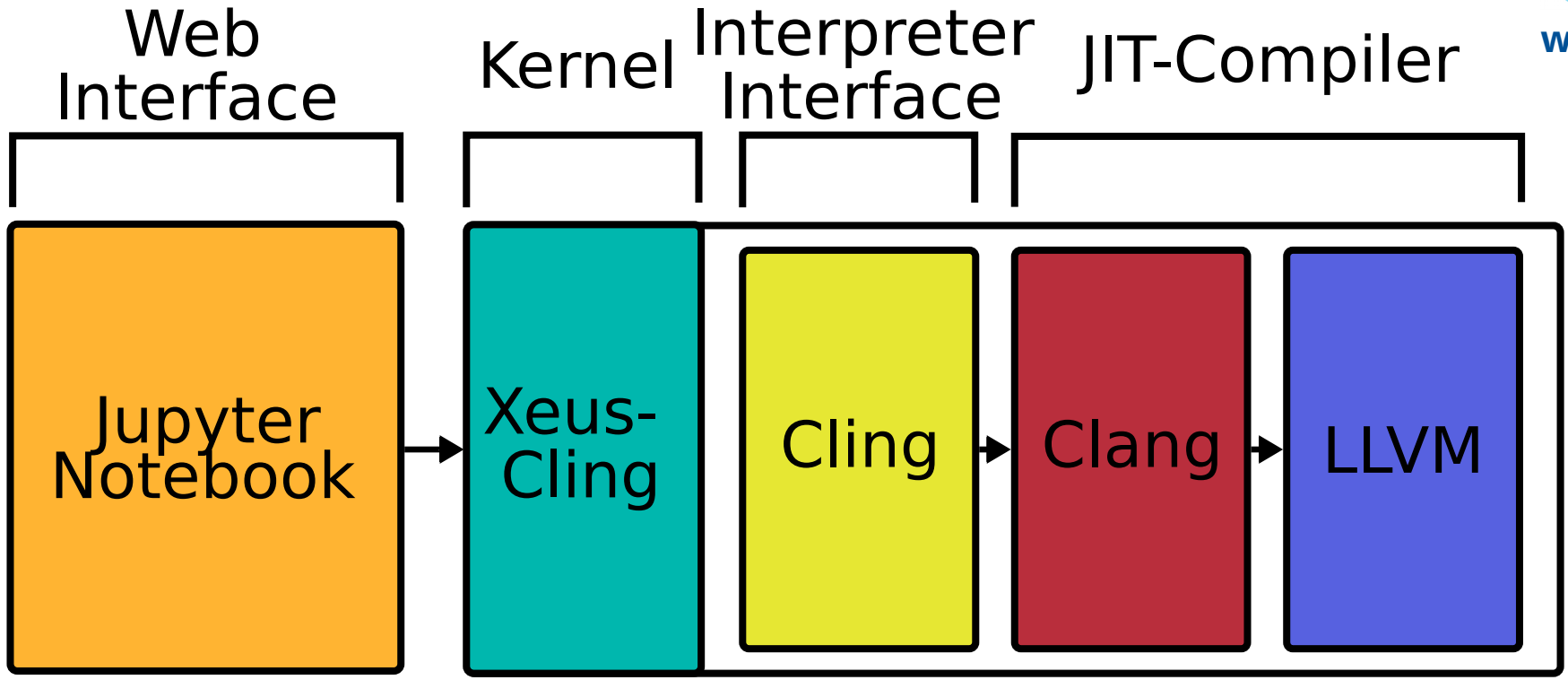
Xeus-Cling Jupyter Architecture



Xeus-Cling Jupyter Architecture



Xeus-Cling Jupyter Architecture



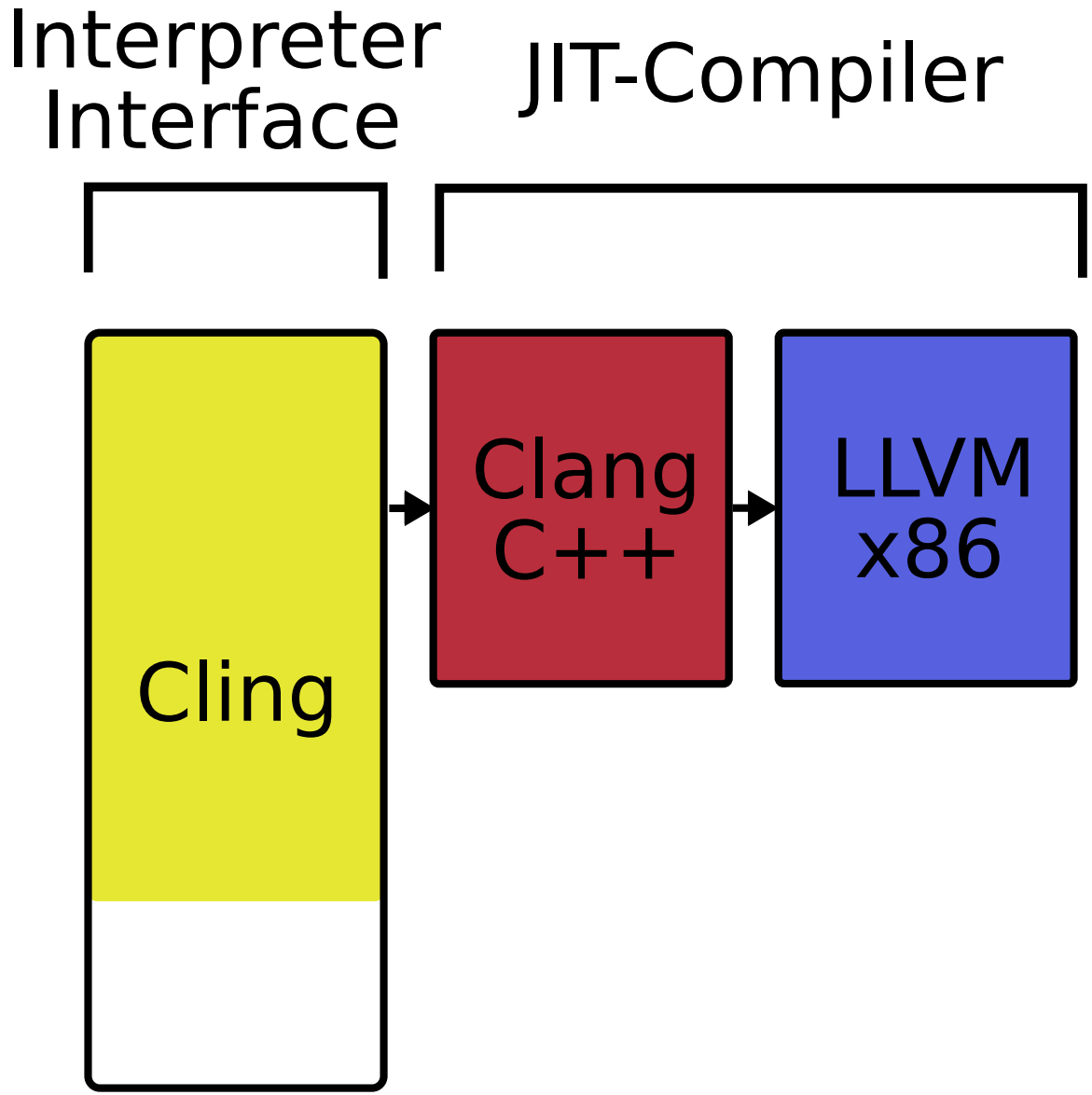
Jupyter

Quant-stack

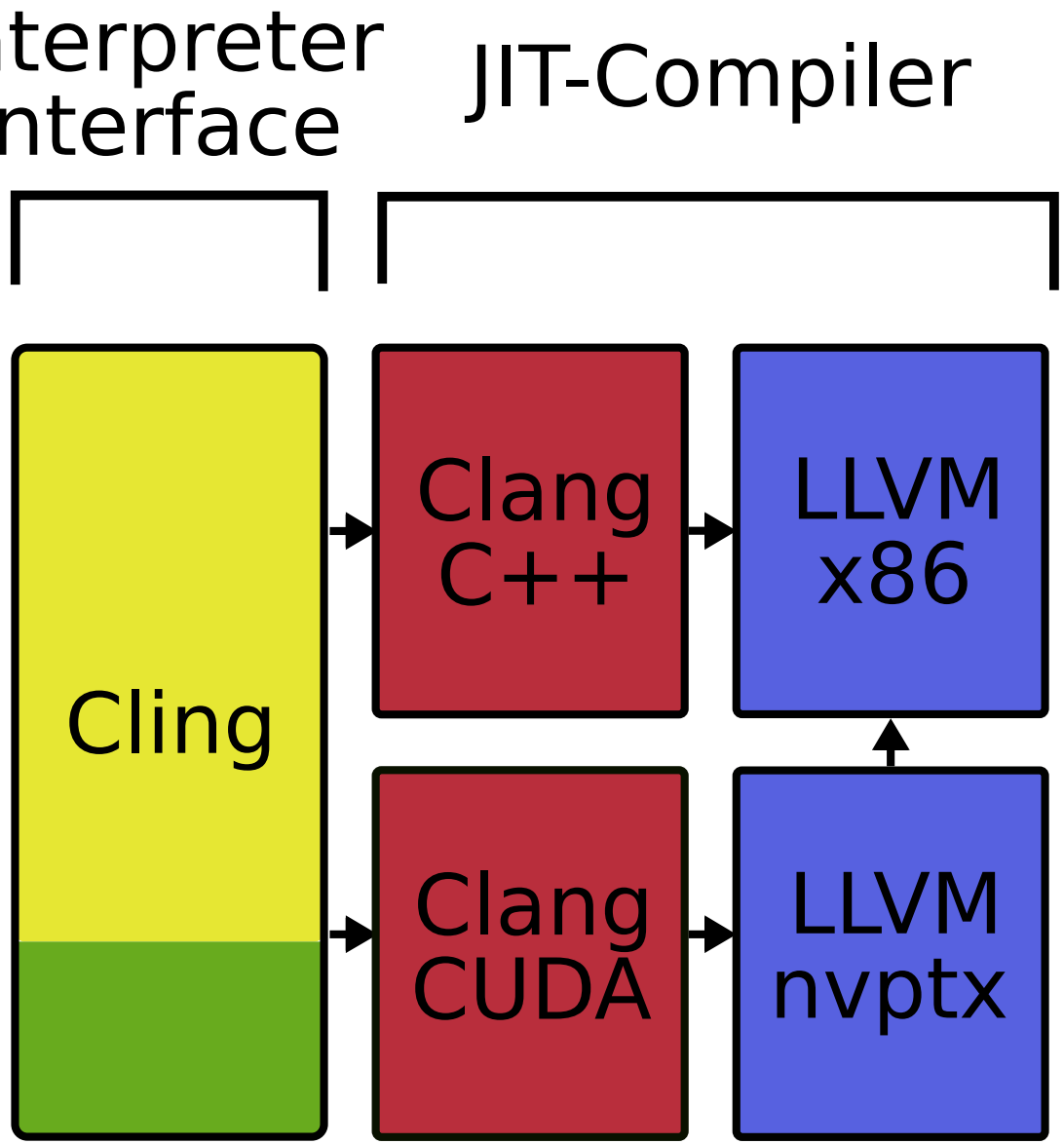
CERN

LLVM

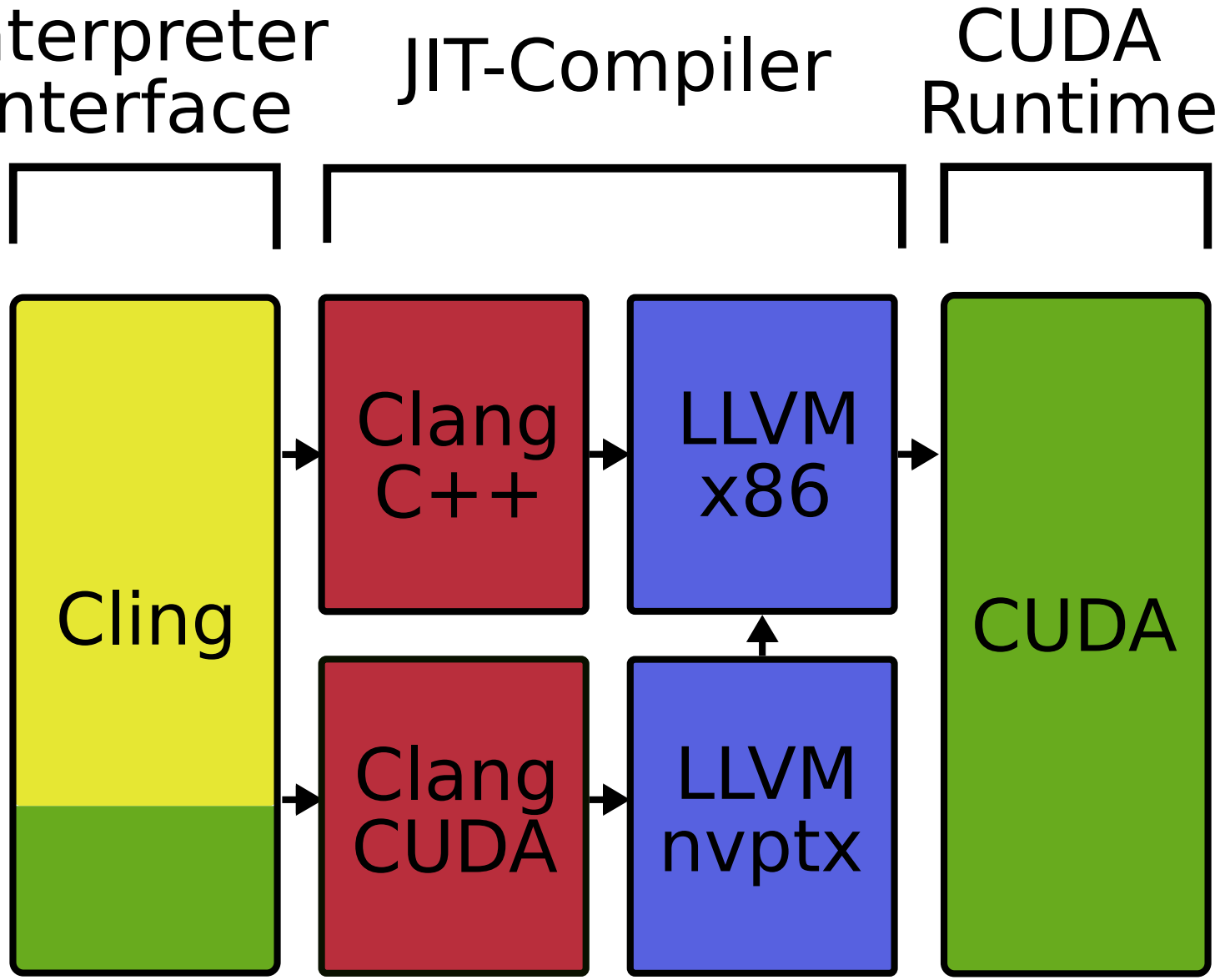
CUDA Extension



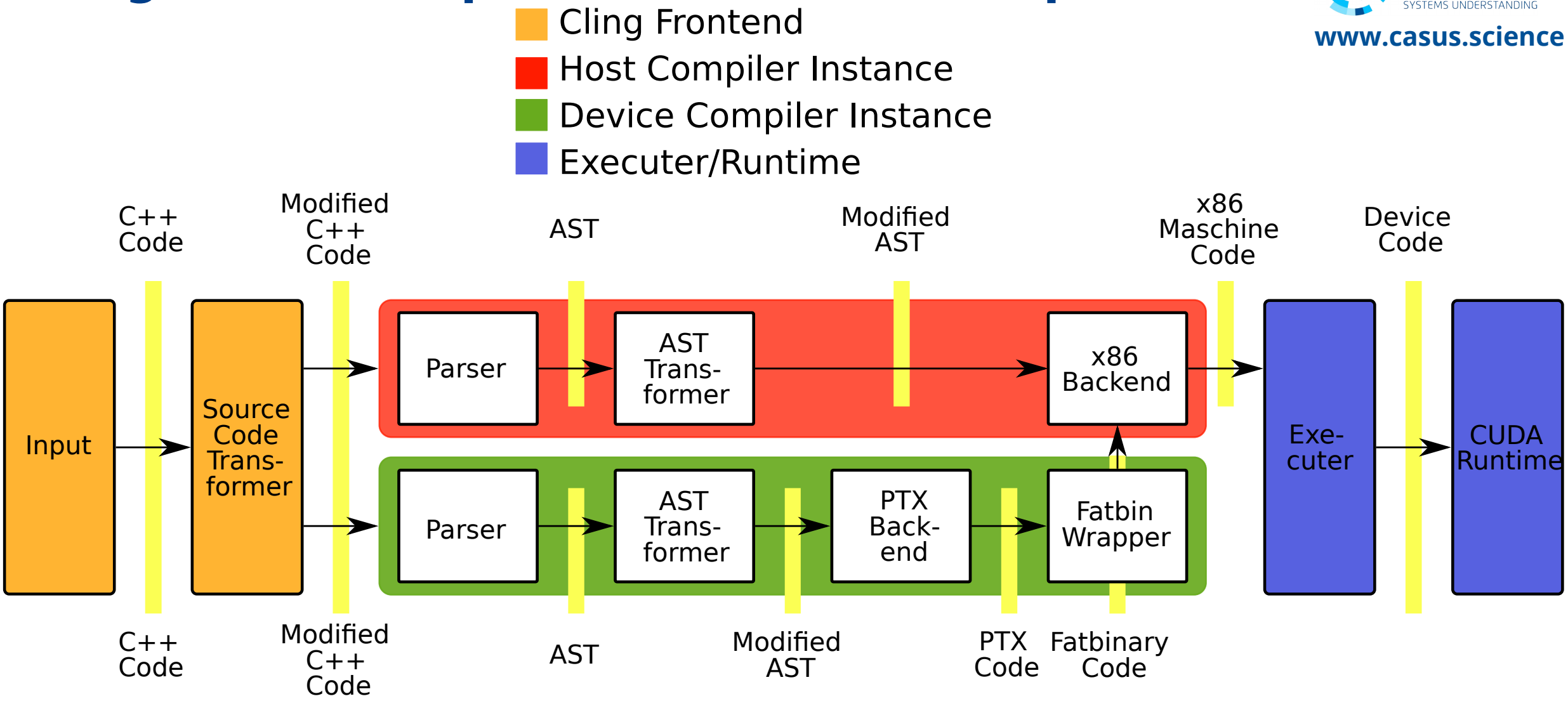
CUDA Extension



CUDA Extension



Cling-CUDA Compiler and Execution Pipeline



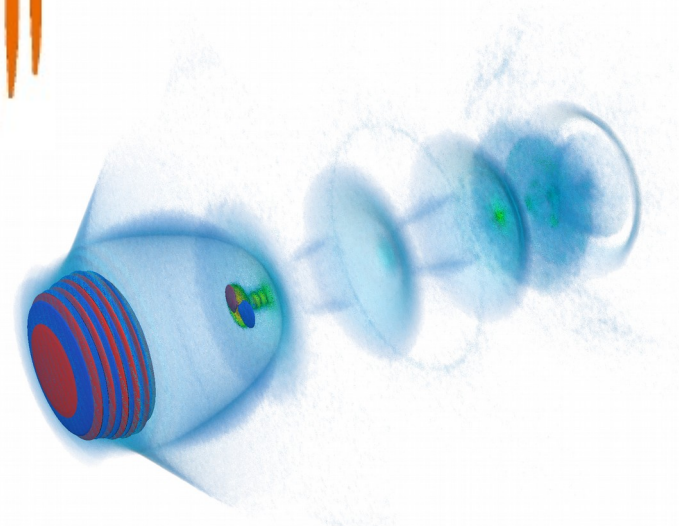
CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Our Vision for Interactive, GPU-driven Supercomputing

I/O Bound

PIConGPU

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING



A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

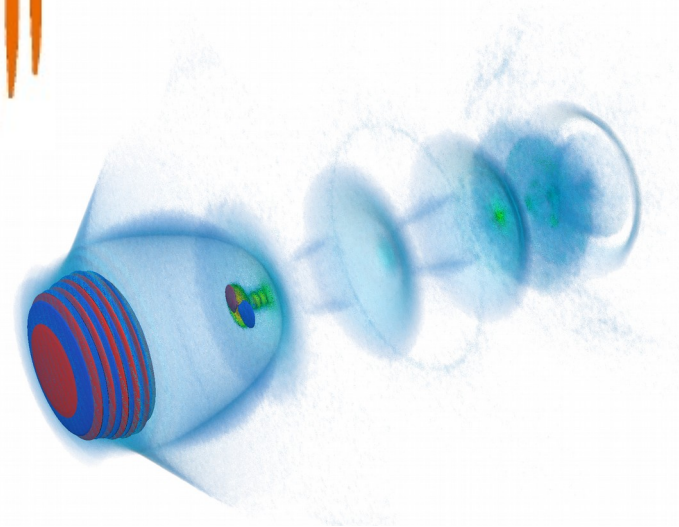
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ
Uniwersytet Wrocławski
NVIDIA GPU CENTER OF EXCELLENCE
DRESDEN concept excellence-dresden.de

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

I/O Bound

PIConGPU

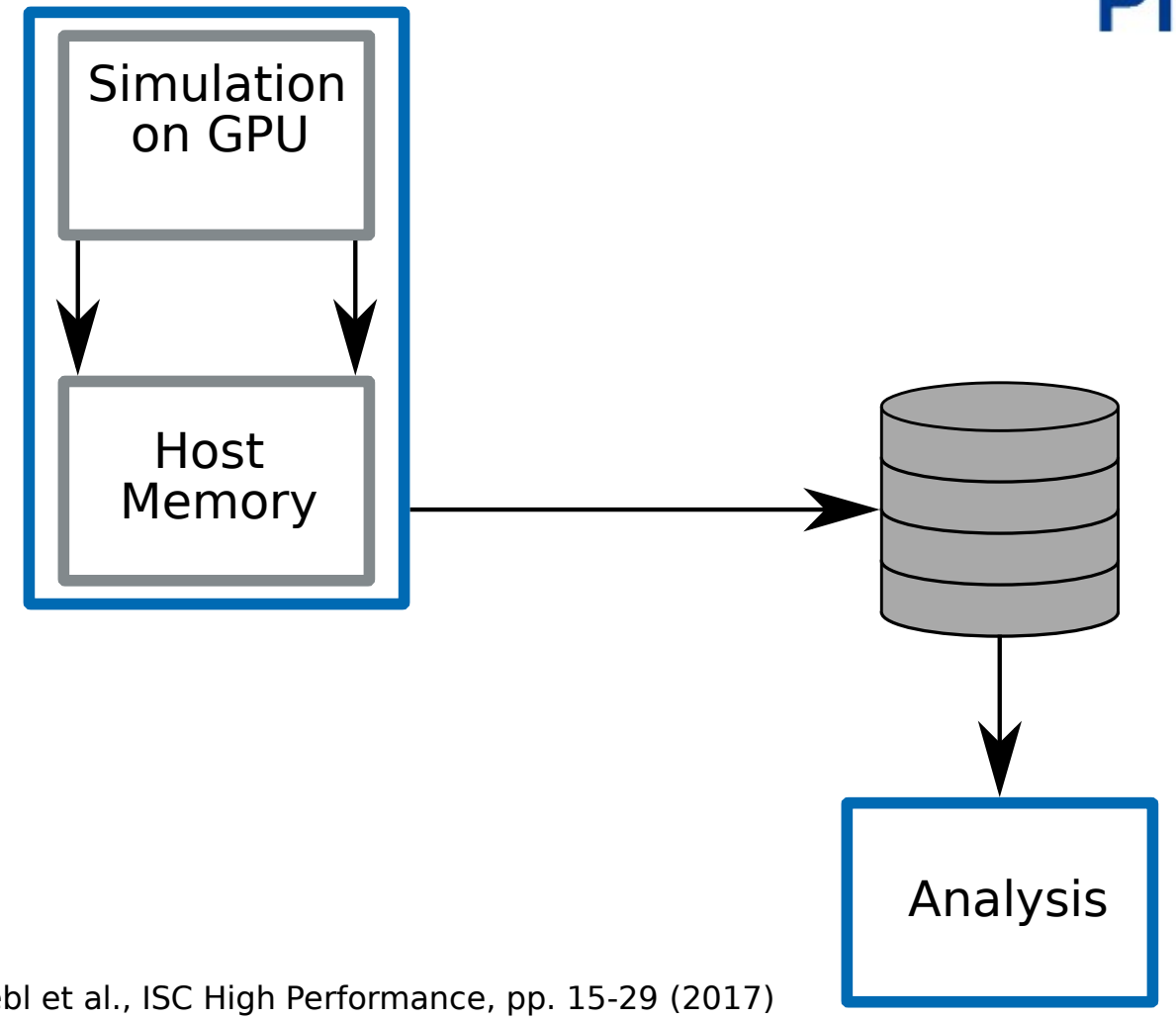
CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

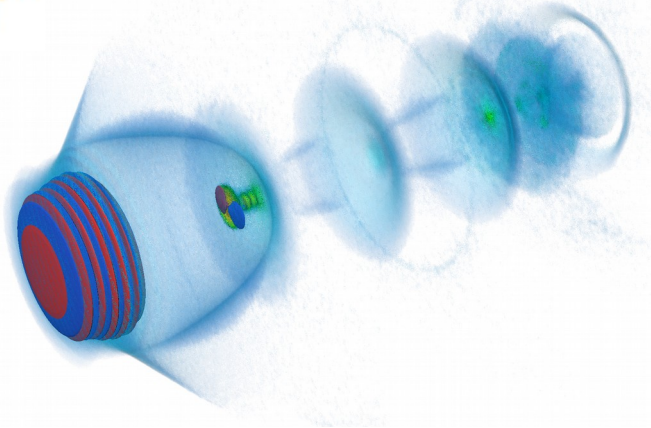


A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

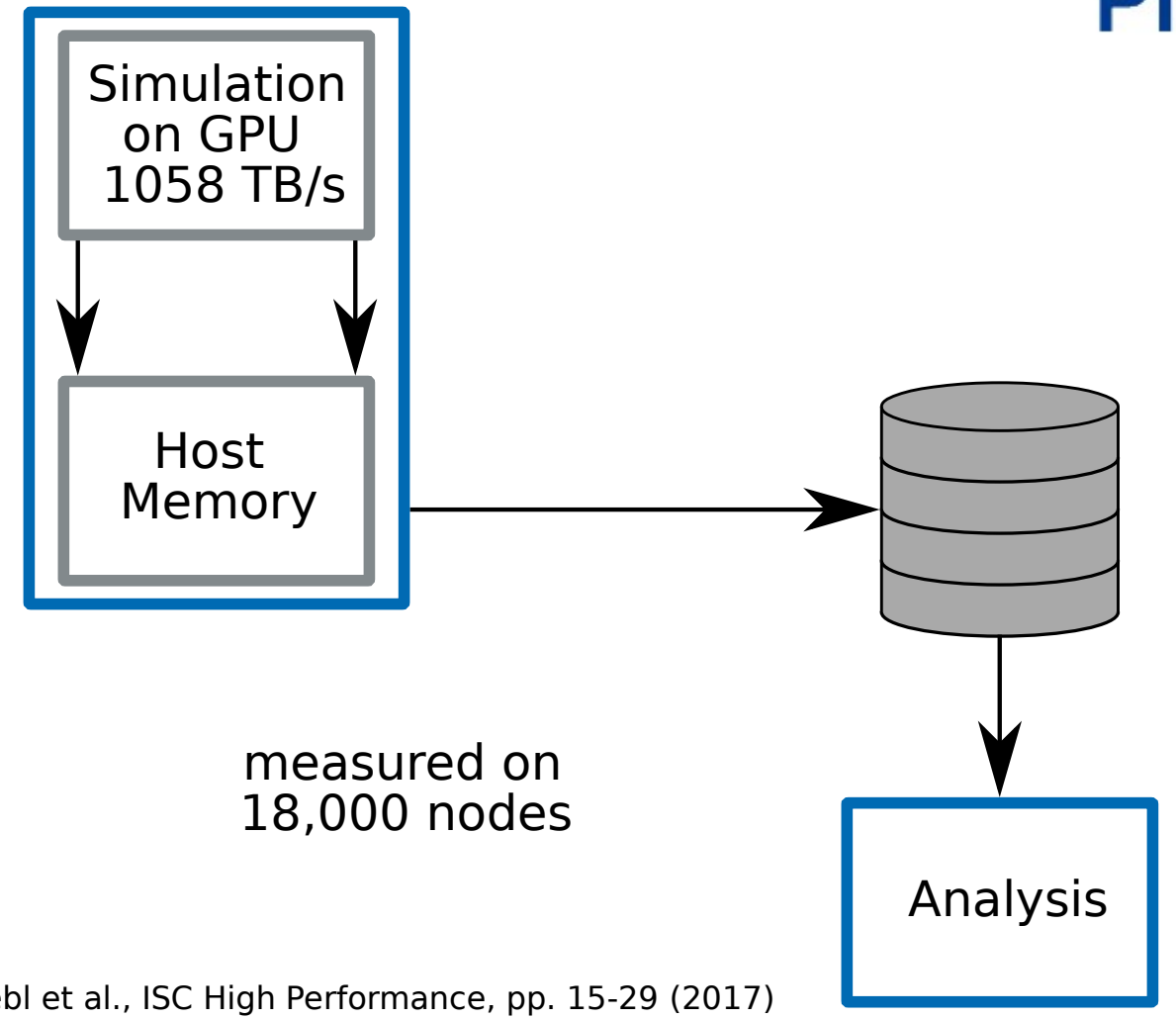
I/O Bound



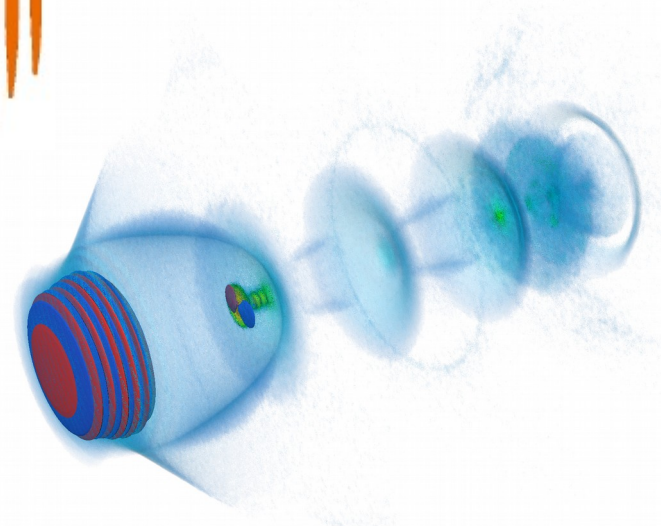


A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

I/O Bound

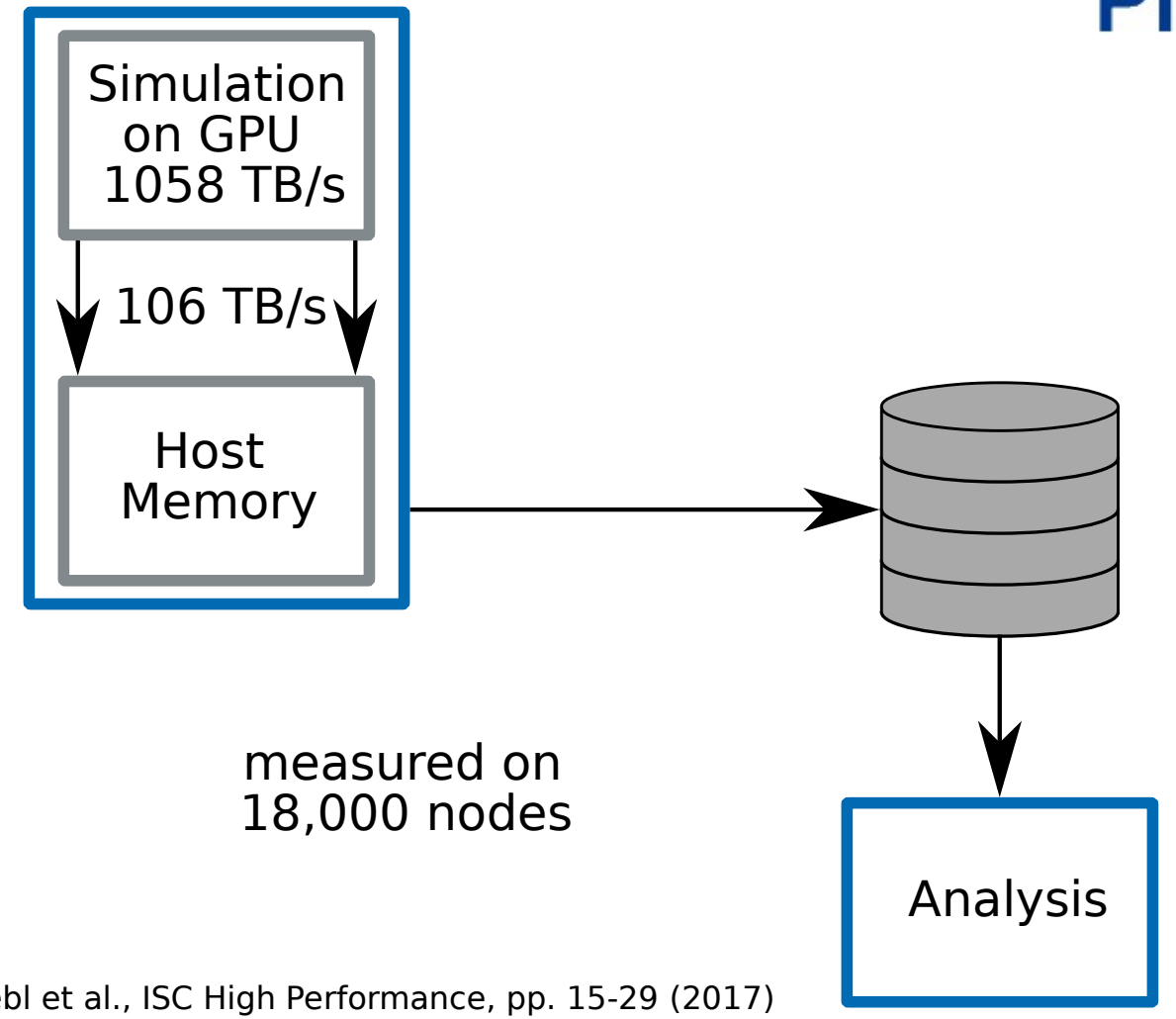


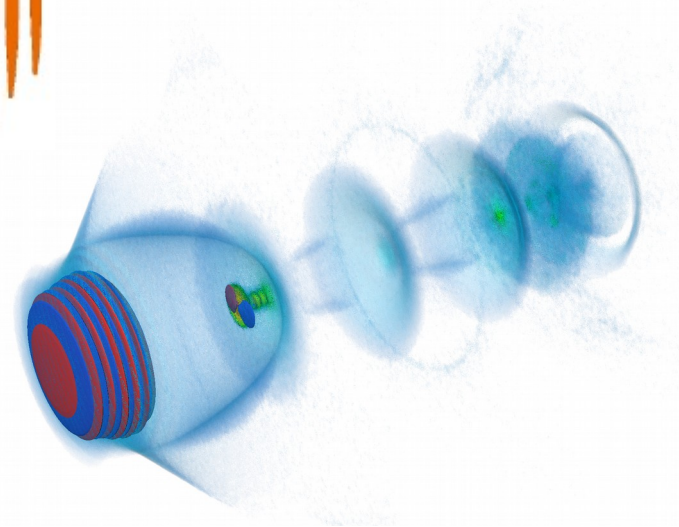
measured on
18,000 nodes



A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

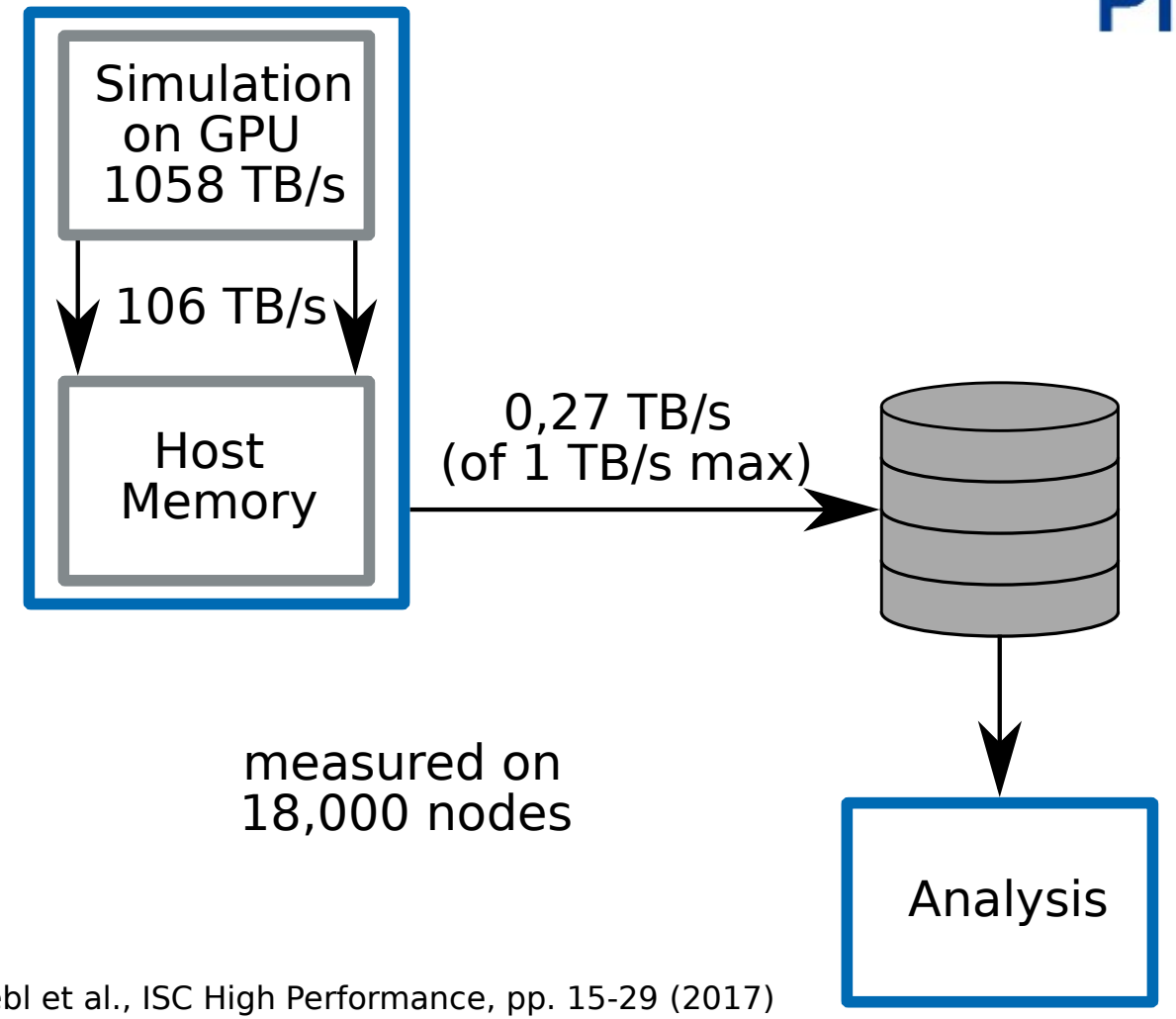
I/O Bound



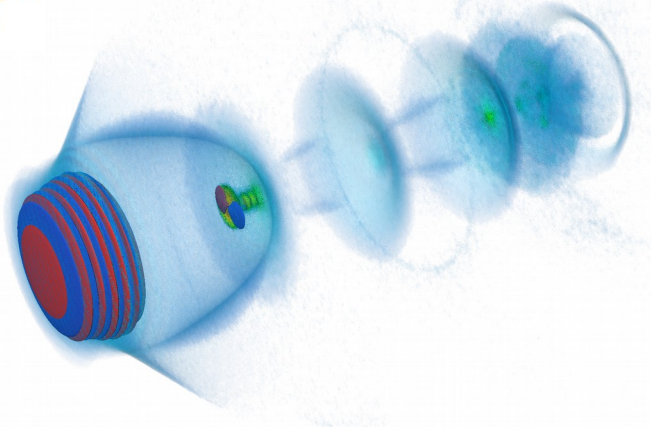


A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

I/O Bound



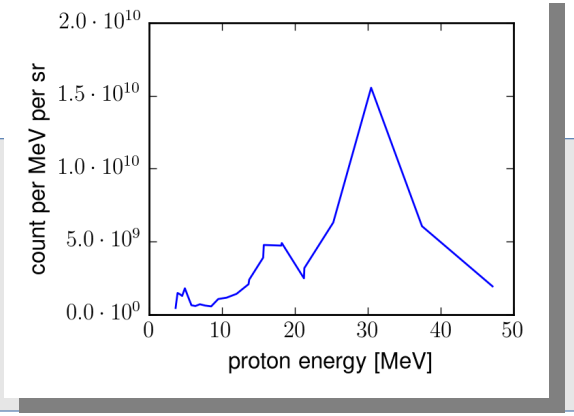
measured on 18,000 nodes



A. Huebl et al., ISC High Performance, pp. 15-29 (2017)
DOI:10.1007/978-3-319-67630-2_2

In Situ Data Analysis

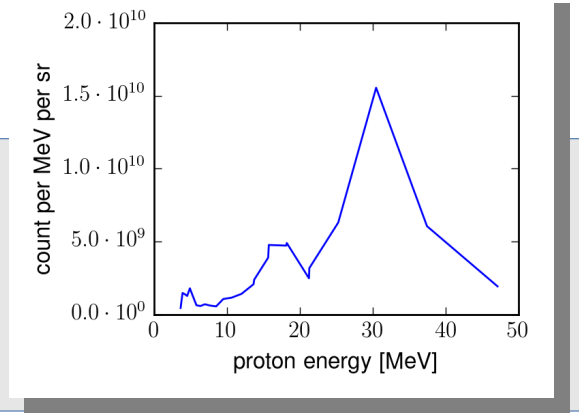
Binning of a **spectrogram**
Creation of a **phase space image**



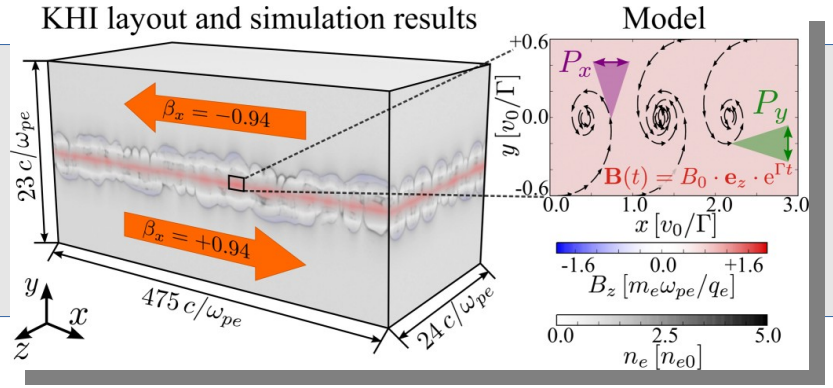
A. Huebl et al. (2014), DOI:10.1109/TPS.2014.2327392

In Situ Data Analysis

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



In situ **radiation diagnostics**

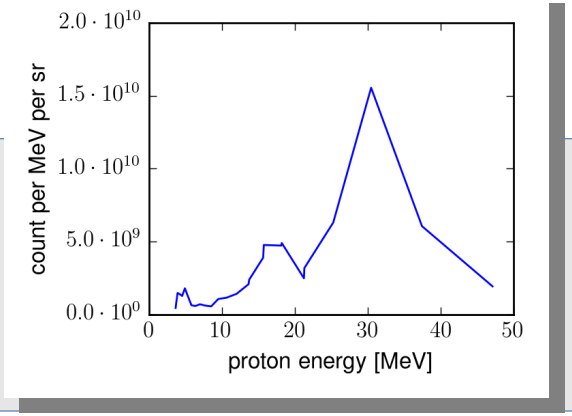


A. Huebl et al. (2014), DOI:10.1109/TPS.2014.2327392
R. Pausch et al. (2014, 2017, 2018), DOI:10.1103/PhysRevE.96.013316

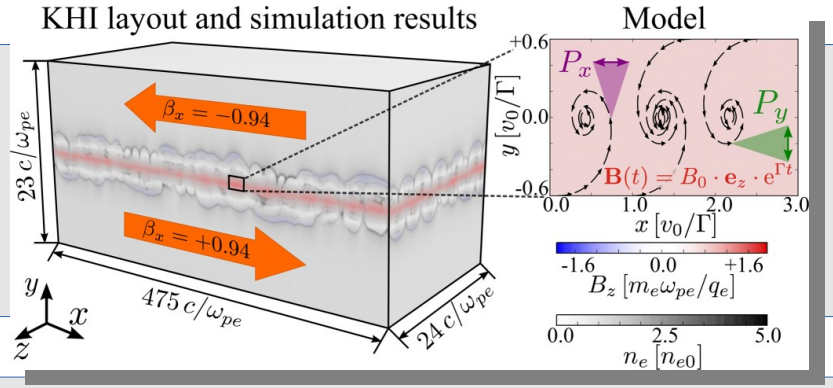
In Situ Data Analysis

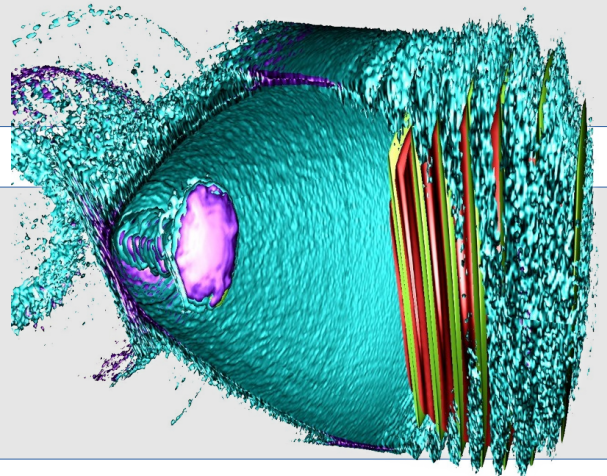
CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

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



In situ **radiation diagnostics**





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

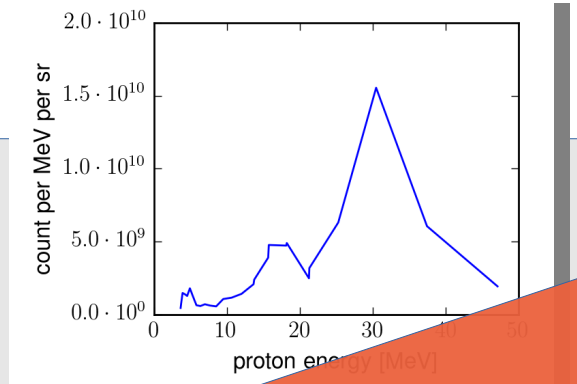
A. Huebl et al. (2014), DOI:10.1109/TPS.2014.2327392
R. Pausch et al. (2014, 2017, 2018), DOI:10.1103/PhysRevE.96.013316
A. Matthes, A. Huebl et al., ISC'16 (2016), DOI:10.14529/jsfi160403
A. Huebl et al., ISC'17 (2017), DOI:10.1007/978-3-319-67630-2_2

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

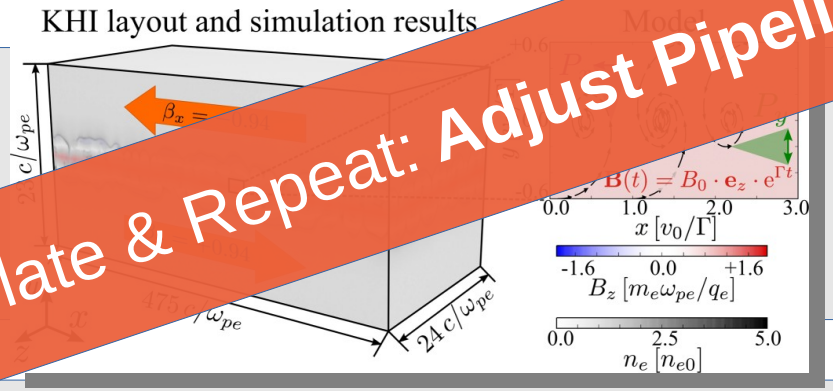
In Situ Data Analysis

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

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

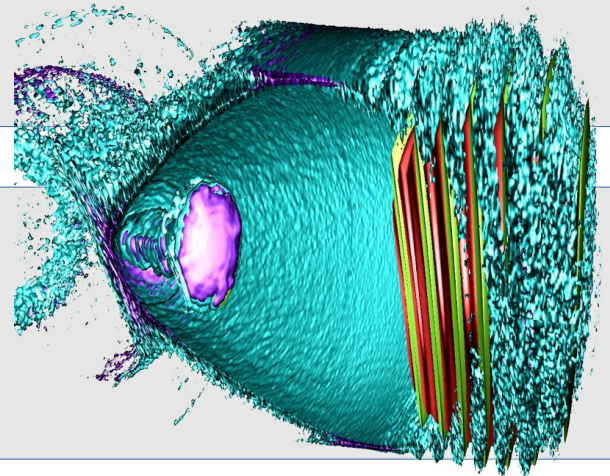


In situ **radiation diagnostics**



Observe, Correlate & Repeat: Adjust Pipeline

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



A. Huebl et al. (2014), DOI:10.1109/TPS.2014.2327392
R. Pausch et al. (2014, 2017, 2018), 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

Restriction of Analysis

- **I/O bandwidth** limits available data for analysis
- Analysis must be **integrated** into the simulation in-situ

- If you know **what to look** for: pre-configured, fixed analysis
- **Explorative** studies: require **feedback** to the simulation
  - **runtime-dependent** data analysis **kernels**
  - **optimize** kernels on runtime parameters
  - avoid explosion of **parameter space** at compile-time

[Game of Life Demo]

Current development

Cling-CUDA & Jupyter Extension Features

Feature	Status
CUDA Runtime support	✔ (see next slide)
Non-linear program flow	✔
Persistent memory	✔
Template specialization	✔
Includes and linking	✔
REPL Object Representation “`_repr_html_`”	✔
Reflection	✔
Redefinition of kernels	✘
Interaction with web elements	(✔)

Current Development

- Support additional CUDA features
  - `__constant__` device memory
  - global `__device__` variables
  - capture `printf` from kernels

- **Upstream**: features **merged**, follow-up patches submitted
- Implement **redefinition** support for **kernels**
- Support sophisticated C++ libraries in CUDA mode, e.g.
  - **xwidgets**: HTML buttons, sliders ...
  - **Alpaka**: our performance portability layer

Portable programming with Alpaka

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

- Alpaka is a C++14 header-only library, providing a GPU/CPU performance-portability layer

Portable programming with Alpaka

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

- Alpaka is a C++14 header-only library, providing a GPU/CPU performance-portability layer

alpaka

```
using Acc1 = alpaka::acc::AccCpuOmp2Blocks<Dim, Idx>;
using Acc2 = alpaka::acc::AccGpuCudaRt<Dim, Idx>;
// ...
auto func = [ ] ALPAKA_FN_ACC ( Acc const & acc) -> void {
                  printf("Hello World\n");
                };

// run Hello World parallel on CPU
alpaka::kernel::exec<Acc1>(queue, workDiv1, func);
// run Hello World parallel on GPU
alpaka::kernel::exec<Acc2>(queue, workDiv2, func);
```

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Try it yourself!
All our results are Open Source

Getting started

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

- Singularity container with full software stack and examples already available

Getting started

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

- Singularity container with full software stack and examples already available
- Requirements
  - Linux
  - Nvidia Driver > 375.26
  - Singularity > 3.3, see https://sylabs.io/guides/3.5/user-guide/
  - Web browser (Firefox or Chrome recommended)

Getting started

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

- Singularity container with full software stack and examples already available
- Requirements
  - Linux
  - Nvidia Driver > 375.26
  - Singularity > 3.3, see https://sylabs.io/guides/3.5/user-guide/
  - Web browser (Firefox or Chrome recommended)
- run singularity run --nv library://sehrig/default/gol-cling-cuda-example and enjoy interactive CUDA programming :-)

Getting started

- Singularity container with full software stack and examples already available
- Requirements
  - Linux
  - Nvidia Driver > 375.26
  - Singularity > 3.3, see https://sylabs.io/guides/3.5/user-guide/
  - Web browser (Firefox or Chrome recommended)
- run singularity run --nv library://sehrig/default/gol-cling-cuda-example and enjoy interactive CUDA programming :-)
- Nvidia Parallel For All blog post with details and example is work in progress

Source Code

- Cling:
  https://github.com/root-project/cling
- experimental Cling-CUDA features:
  https://github.com/SimeonEhrig/cling
- example notebook and container:
  https://github.com/ComputationalRadiationPhysics/Xeus-Cling-CUDA-Example/tree/master/GOL-function-presentation
- PIConGPU:
  https://github.com/ComputationalRadiationPhysics/picongpu
- Alpaka:
  https://github.com/ComputationalRadiationPhysics/alpaka
- Cling-CUDA diploma thesis (in German)
  https://doi.org/10.5281/zenodo.3713682

Acknowledgments

Funding and Collaborators

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING
www.casus.science

HZDR

HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

QuantStack
Scientific Computing

CERN

Group: M. Bussmann

S. Corlay, J. Mabille et al., *Jupyter-Xeus*

A. Naumann, et al. *Cling (ROOT)*

S. Ehrig acknowledges travel funding by CASUS, Germany. A. Huebl acknowledges travel funding by Berkeley Lab, USA.

Thank you to all LLVM developers, Nvidia for the nvptx backend in LLVM and to Google for Clang's CUDA frontend.

HZDR 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
NVIDIA GPU CENTER OF EXCELLENCE
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

Basics

March 19, 2020

1 About

This notebook demonstrates features of Cling, Xeus-Cling and Jupyter Notebook.

2 Hello World

[1]:
```
#include <iostream>

std::cout << "Hello World" << std::endl;
```

```
Hello World
```

- No main() necessary
- each statement is in global space
- some statements are forbidden in global space, like function calls ...
- ... but Cling handles such situations and transforms these statements internally

3 Global and Local Variables

[2]:
```
// global variable
int g1 = 1;
```

[3]:
```
// local variable
{
    int l1 = 2;
}
```

[4]:
```
std::cout << l1 << std::endl;
```

```
input_line_11:2:15: error: use of undeclared identifier
'l1'
 std::cout << l1 << std::endl;
              ^


        Interpreter Error:
```

[5]:
```
std::cout << g1 << std::endl;
{
    // hide global variable
    int g1 = 3;
    std::cout << g1 << std::endl;
}
std::cout << g1 << std::endl;
```

```
1
3
1
```

4 Standard C++ Features

[6]:
```
int fd1(int k){
    return k + 2;
}
```

[7]:
```
std::cout << fd1(3) << std::endl;
```

```
5
```

[8]:
```
class Cd1 {
  int a;
  int b;

public:
    Cd1(int a, int b) : a(a), b(b) {}
    int sum(){
        return a + b;
    }
};
```

[9]:
```
Cd1 cd1(4, 7);
std::cout << cd1.sum() << std::endl;
```

```
11
```

5 Non-Linear Program Flow

[15]:
```
std::cout << non_lin_var << std::endl;
```

```
4
```

[13]:
```
++non_lin_var;
```

[11]:
```
int non_lin_var = 3;
```

6 Persistent Memory

[16]:
```
int k = 0;
```

[20]:
```
for (int end = k + 5; k < end; ++k){
    std::cout << k << std::endl;
}
```

```
7
8
9
10
11
```

[19]:
```
k -= 3;
```

7 Template Specialization

[21]:
```
#include <chrono>
constexpr int dim = 512;
```

[22]:
```
float * A = new float[dim * dim];
float * B = new float[dim * dim];
float * C = new float[dim * dim];
```

[23]:
```
for(int i = 0; i < dim; ++i){
    A[i] = static_cast<float>(i);
    B[i] = static_cast<float>(i);
}
```

[24]:
```
void var_matmul(float const * const A, float const * const B, float * const C,␣
 ↪const int dim)
{
    float sum = 0.f;
    for (int i = 0; i < dim; ++i) {
      for (int j = 0; j < dim; ++j) {
        for (int k = 0; k < dim; ++k) {
          sum += A[i * dim + k] * B[k * dim + j];
        }
        C[ i * dim + j] = sum;
        sum = 0.f;
      }
    }
}
```

[25]:
```
template<int dim>
void t_matmul(float const * const A, float const * const B, float * const C)
{
    float sum = 0.f;
    for (int i = 0; i < dim; ++i) {
      for (int j = 0; j < dim; ++j) {
        for (int k = 0; k < dim; ++k) {
          sum += A[i * dim + k] * B[k * dim + j];
        }
        C[ i * dim + j] = sum;
        sum = 0.f;
      }
    }
}
```

[26]:
```
var_matmul(A, B, C, dim);
t_matmul<dim>(A,B,C);
```

[27]:
```
{
    std::chrono::time_point<std::chrono::high_resolution_clock> v_start, v_end,␣
 ↪t_start, t_end;

    v_start = std::chrono::high_resolution_clock::now();
    var_matmul(A, B, C, dim);
    v_end = std::chrono::high_resolution_clock::now();

    t_start = std::chrono::high_resolution_clock::now();
```

```
    t_matmul<dim>(A, B, C);
    t_end = std::chrono::high_resolution_clock::now();

    std::chrono::duration<double> v_diff = v_end - v_start;
    std::chrono::duration<double> t_diff = t_end - t_start;

    std::cout << "var_matmul: " << v_diff.count() << "s" << std::endl
              << "t_matmul: " << t_diff.count() << "s" << std::endl;
}
```

```
var_matmul: 0.540087s
t_matmul: 0.530416s
```

8 Including and Linking

8.1 Preparation: create a shared library

[28]:
```
%%file foo.hpp
#pragma once

namespace foo {
    int bar();
}
```

```
Overwriting foo.hpp
```

[29]:
```
%%file foo.cpp
#include "foo.hpp"

int foo::bar() { return 42; }
```

```
Overwriting foo.cpp
```

[30]:
```
!gcc -shared foo.cpp -o foo.so
```

8.2 Call Functionality of the Library

[31]:
```
foo::bar()
```

```
input_line_36:2:2: error: use of undeclared identifier
'foo'
 foo::bar()
 ^
```

```
Interpreter Error:
```

[32]:
```
#include "foo.hpp"
```

[33]:
```
foo::bar()
```

```
IncrementalExecutor::executeFunction: symbol '_ZN3foo3barEv' unresolved while
linking [cling interface function]!
You are probably missing the definition of foo::bar()
Maybe you need to load the corresponding shared library?
```

```
Interpreter Error:
```

[34]:
```
#pragma cling(load "foo.so")
```

[35]:
```
foo::bar()
```

[35]:
```
42
```

9 REPL Object Representation

[36]:
```
"Hello World"
```

[36]:
```
"Hello World"
```

[37]:
```
int i1 = 3;
```

[38]:
```
i1
```

[38]:
```
3
```

[39]:
```
int fi1(){
    return 42;
}
```

[40]:
```
fi1()
```

[40]:
```
42
```

10 Reflection

- values of variables
- type of a variable (cling kernel only)
- memory address
- enum completion (cling kernel only)
- interpreter environment

[41]:
```
struct S {
    int a = 3;
    float b = 6,f;
} s;
```

[42]:
```
s
```

[42]:
```
@0x7fa5267b9054
```

[43]:
```
s.a
```

[43]:
```
3
```

[44]:
```
#include "cling/Interpreter/Interpreter.h"
```

[45]:
```
gCling->getDefaultOptLevel()
```

[45]:
```
0
```

[46]:
```
gCling->setDefaultOptLevel(3)
```

11 Redefinition

[47]:
```
#include "cling/Interpreter/Interpreter.h"
gCling->allowRedefinition();
gCling->isRedefinitionAllowed();
```

[48]:
```
int func(){
    return 43;
}
```

[49]:
```
func()
```

[49]:
```
43
```

[50]:
```
int func(){
    return 42;
```

```
}
```

[51]:
```
func()
```

[51]:
```
42
```

[52]:
```
class class1 {
    int a = 3;
    int b = 4;
public:
    int func() {
        return a + b;
    }
};
```

[53]:
```
{
    class1 c;
    std::cout << c.func() << std::endl;
}
```

```
7
```

[54]:
```
class class1 {
    int a = 3;
    int b = 40;
public:
    int func() {
        return a + b;
    }
};
```

[55]:
```
{
    class1 c;
    std::cout << c.func() << std::endl;
}
```

```
43
```

12 I/O through Web Elements

[56]:
```
#include <string>
#include <fstream>

#include "xtl/xbase64.hpp"
#include "xeus/xjson.hpp"
```

[57]:
```
void display_image(const std::string filename){
    std::ifstream fin(filename, std::ios::binary);
    std::stringstream buffer;
    buffer << fin.rdbuf();
    // memory objects for output in the web browser

    xeus::xjson mine;

    xeus::get_interpreter().clear_output(true);

    mine["image/png"] = xtl::base64encode(buffer.str());
    xeus::get_interpreter().display_data(
        std::move(mine),
        xeus::xjson::object(),
        xeus::xjson::object());
}
```

[58]:
```
display_image("pictures/conclusion_basics.png");
```

Conclusion

Changed Behavior

- Simplified syntax
- Non-linear flow
- Persistent memory
- Template parameters at runtime
- linking at runtime

New Features

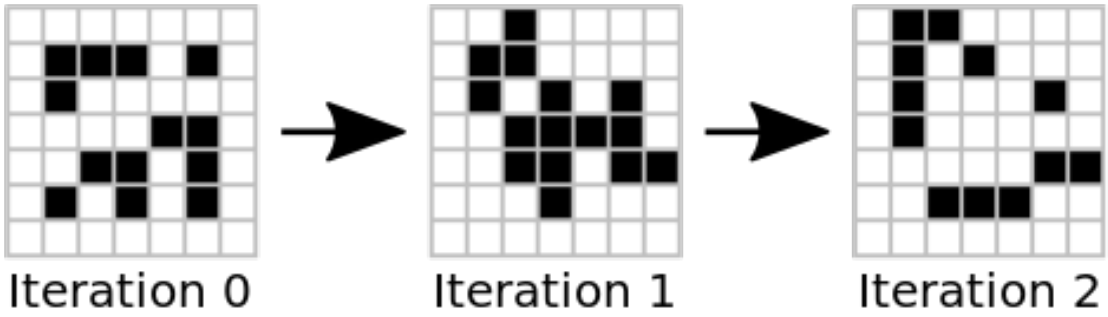
- REPL Object Representation
- Reflection
- Redefinition
- I/O with web elements
- Features of Jupyter Notebook

GameOfLife

March 20, 2020

1 Game of Life on GPU - Interactive & Extensible

- The example shows an interactive workflow of simulation and analysis.
- The simulation runs Conway's Game of Life on a GPU.



2 Include and Link

[1]:
```
// set the include path of PNGwriter
//   https://github.com/pngwriter/pngwriter/tree/dev
// (like -Ipngwriter/include for a compiler)
#pragma cling(add_include_path "pngwriter/include")

#include <fstream>
#include <vector>
#include <sstream>
#include <chrono>
#include <thread>

// include PNGwriter
#define NO_FREETYPE
#include <pngwriter.h>

// helper functions for displaying images
#include "xtl/xbase64.hpp"
#include "xeus/xjson.hpp"
```

```
// self-defined helper functions
#include "color_maps.hpp"
#include "input_reader.hpp"
#include "png_generator.hpp"
#include "helper.hpp"

// link PNGwriter (like -lPNGwriter for a compiler)
#pragma cling(load "pngwriter/lib/libPNGwriter.so")
```

3 Game of Life Setup

- setup world size
- allocate memory on CPU and GPU
- load initial world
- copy initial world to the GPU
- generate image of the initial world

[2]:
```
// size of the world
const unsigned int dim = 10u;
// two extra columns and rows for ghostcells
const unsigned int world_size = dim + 2u;
unsigned int iterations = 5;
unsigned int current_png = 0;

// pointers for host and device memory
int * sim_world;
int * d_sim_world;
int * d_new_sim_world;
int * d_swap;
// allocate memory on CPU and GPU
sim_world = new int[ world_size * world_size ];
cuCheck(cudaMalloc( (void **) &d_sim_world, sizeof(int)*world_size*world_size));
cuCheck(cudaMalloc( (void **) &d_new_sim_world,␣
 ↪sizeof(int)*world_size*world_size));

// read initial world from a file
if (int error = read_input("input.txt", sim_world, dim, dim, true))
    std::cout << "read input world failed - error code: " << error << std::endl;

// copy initial world to GPU
cuCheck(cudaMemcpy(d_sim_world, sim_world, sizeof(int)*world_size*world_size,␣
 ↪cudaMemcpyHostToDevice));

// allocate memory for the simulation images
std::vector< std::vector< unsigned char > > sim_pngs;
```

```
// create an image of the initial world
BlackWhiteMap<int> bw_map;
sim_pngs.push_back(generate_png<int>(sim_world, world_size, world_size,␣
 ↪&bw_map, true, 20));
```

4 CUDA Kernels

[3]:
```
// periodic boundary conditions: copy the first/last row/column
__global__ void update_boundaries(int dim, int *world) {
  int col = blockIdx.x * blockDim.x + threadIdx.x;
  int row = blockIdx.y * blockDim.y + threadIdx.y;

  // ignore the first two threads: only needed to copy ghost cells for columns
  if(col > 1) {
    if(row == 0) {
      // Copy first real row to bottom ghost row
      world[col-1 + (dim+2)*(dim+1)] = world[(dim+2)     + col-1];
    }else{
      // Copy last real row to top ghost row
      world[col-1]                   = world[(dim+2)*dim + col-1];
    }
  }
  __syncthreads();

  if(row == 0) {
    // Copy first real column to right most ghost column
    world[col*(dim+2)+dim+1] = world[col*(dim+2) + 1];
  } else {
    // Copy last real column to left most ghost column
    world[col*(dim+2)      ] = world[col*(dim+2) + dim];
  }

}
```

[4]:
```
// main kernel that calculates an iteration of the game of life
__global__ void GOL_GPU(int dim, int *world, int *newWorld) {
   int row = blockIdx.y * blockDim.y + threadIdx.y + 1;
   int col = blockIdx.x * blockDim.x + threadIdx.x + 1;
   int id = row*(dim+2) + col;

   int numNeighbors;
   int cell = world[id];

   numNeighbors =   world[id+(dim+2)]   // lower
      + world[id-(dim+2)]               // upper
```

```
      + world[id+1]                     // right
      + world[id-1]                     // left

      + world[id+(dim+3)]   // diagonal lower right
      + world[id-(dim+3)]   // diagonal upper left
      + world[id-(dim+1)]   // diagonal upper right
      + world[id+(dim+1)];  // diagonal lower left

   if (cell == 1 && numNeighbors < 2)
      newWorld[id] = 0;

    // 2) Any living cell with two or three living neighbors lives
    else if (cell == 1 && (numNeighbors == 2 || numNeighbors == 3))
      newWorld[id] = 1;

    // 3) Any living cell with more than three living neighbors dies
    else if (cell == 1 && numNeighbors > 3)
      newWorld[id] = 0;

    // 4) Any dead cell with exactly three living neighbors becomes alive
    else if (cell == 0 && numNeighbors == 3)
      newWorld[id] = 1;

    else
      newWorld[id] = cell;
}
```

5 Interactive Simulation: Main Loop

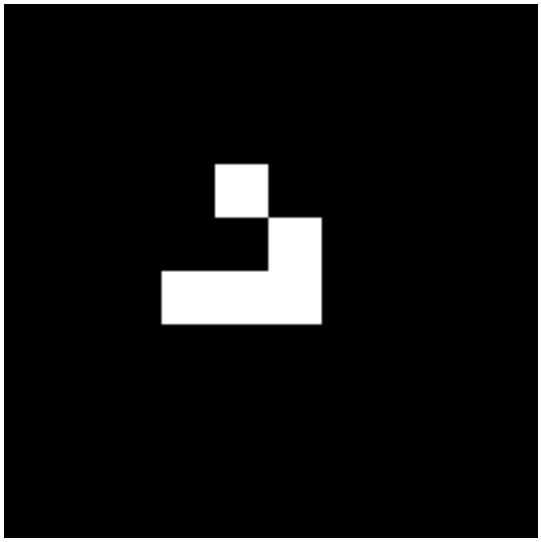
- calculate new iterations
- swap new world with the old one
- generate an image of the current iteration

[8]:
```
// main loop
for(unsigned int i = 0; i < iterations; ++i) {

    update_boundaries<<<1, dim3(dim+2, 2, 1)>>>(dim, d_sim_world);
    GOL_GPU<<<1, dim3(dim, dim, 1)>>>(dim, d_sim_world, d_new_sim_world);
    cuCheck(cudaDeviceSynchronize());

    d_swap = d_new_sim_world;
    d_new_sim_world = d_sim_world;
    d_sim_world = d_swap;
```

```
    cuCheck(cudaMemcpy(sim_world, d_sim_world,␣
 ↪sizeof(int)*world_size*world_size, cudaMemcpyDeviceToHost));
    sim_pngs.push_back(generate_png<int>(sim_world, world_size, world_size,␣
 ↪&bw_map, true, 20));
}
```

6 Display Simulation Images

[9]:
```
for(; current_png < sim_pngs.size(); ++current_png) {
    display_image(sim_pngs[current_png], true);
    std::cout << "iteration = " << current_png << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(800));
}
```



```
iteration = 8
```

6.1 Nonlinear Program Flow

[7]:
```
iterations = 3;
```

7 In-Situ Data Analysis

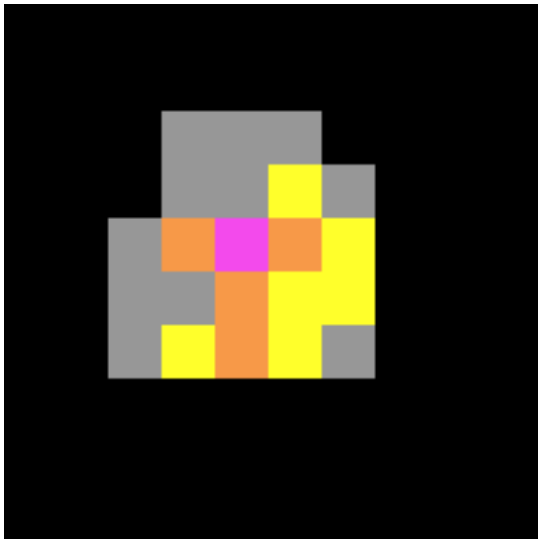
- heatmap of the living neighbors for each cell
- kernel uses the simulation result as its input and writes output to an extra buffer

[10]:
```
// counts the living neighbors of a cell
__global__ void get_num_neighbors(int dim, int *world, int *newWorld) {
   int row = blockIdx.y * blockDim.y + threadIdx.y + 1;
   int col = blockIdx.x * blockDim.x + threadIdx.x + 1;
   int id = row*(dim+2) + col;

   newWorld[id] =   world[id+(dim+2)]   // lower
      + world[id-(dim+2)]               // upper
      + world[id+1]                     // right
      + world[id-1]                     // left

      + world[id+(dim+3)]   // diagonal lower right
      + world[id-(dim+3)]   // diagonal upper left
      + world[id-(dim+1)]   // diagonal upper right
      + world[id+(dim+1)];  // diagonal lower left
}
```

- allocate memory for analysis on CPU and GPU
- run the analysis
- generate an image from the analysis result

[11]:
```
// allocate extra memory on the GPU to output the analysis
int * d_ana_world;
cuCheck(cudaMalloc( (void **) &d_ana_world, sizeof(int)*world_size*world_size));
// allocate memory on CPU for the images
std::vector< std::vector< unsigned char > > ana_pngs;
int * ana_world = new int[world_size*world_size];

// run the analysis
// use the simulation data as input and write the result into extra memory
get_num_neighbors<<<1,dim3(dim, dim, 1)>>>(dim, d_sim_world, d_ana_world);

// copy analysis data to the CPU
cuCheck(cudaMemcpy(ana_world, d_ana_world, sizeof(int)*world_size*world_size,␣
 ↪cudaMemcpyDeviceToHost));
// generate a heat map image
ana_pngs.push_back(generate_png<int>(ana_world, world_size, world_size,␣
 ↪&ch_map, true, 20));
```

[12]:
```
display_image(ana_pngs.back(), true);
```

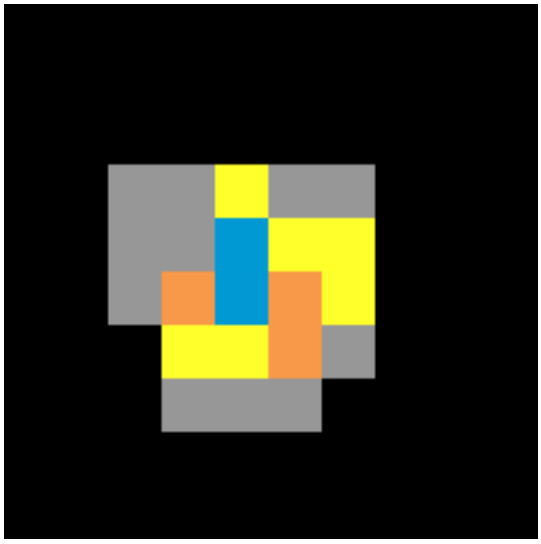


8 Run a single Simulation step with Analysis

[13]:
```
// one iteration of the simulation
update_boundaries<<<1, dim3(dim+2, 2, 1)>>>(dim, d_sim_world);
GOL_GPU<<<1, dim3(dim, dim, 1)>>>(dim, d_sim_world, d_new_sim_world);
cuCheck(cudaDeviceSynchronize());

// swap memory
d_swap = d_new_sim_world;
d_new_sim_world = d_sim_world;
d_sim_world = d_swap;
```

[14]:
```
// run analysis
get_num_neighbors<<<1,dim3(dim, dim, 1)>>>(dim, d_sim_world, d_ana_world);
cuCheck(cudaMemcpy(ana_world, d_ana_world, sizeof(int)*world_size*world_size,␣
 ↪cudaMemcpyDeviceToHost));
ana_pngs.push_back(generate_png<int>(ana_world, world_size, world_size,␣
 ↪&ch_map, true, 20));
```

[15]:
```
display_image(ana_pngs.back(), true);
```



9 Extras

9.1 Interactive Input

- Jupyter Notebook offers "magic" commands that provide language-independent functions
- magic commands starts with `%%`
- `%%file [name]` writes the contents of a cell to a file
  - the file is stored in the same folder as the notebook and can be loaded via C/C++ functions
- depends on the language kernel (xeus features)

Define the initial world for the Game of Life simulation. `X` are living cells and `0` are dead.

[ ]:
```
%%file input.txt
0 0 0 0 0 0 0 0 0 0
0 0 X 0 0 0 0 0 0 0
0 0 0 X 0 0 0 0 0 0
0 X X X 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

9.2 Reset the simulation without restarting the kernel

[ ]:
```
// load a new inital world into the host memory
read_input("input.txt", sim_world, dim, dim, true);
// copy the world to the device
cuCheck(cudaMemcpy(d_sim_world, sim_world, sizeof(int)*world_size*world_size,␣
 ↪cudaMemcpyHostToDevice));
// reset png print counter
current_png = 0;
// delete old images
sim_pngs.clear();
// create an image of the initial world
sim_pngs.push_back(generate_png<int>(sim_world, world_size, world_size,␣
 ↪&bw_map, true, 20));
```