In-situ, steerable, hardware-independent and data-structure agnostic visualization with ISAAC

A. Matthes[1,2], A. Huebl[1,2], R. Widera[1], S. Grottel[2], S. Gumhold[2], M. Bussmann[1]

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

June 23th 2016

EUCALL
NVIDIA GPU CENTER OF EXCELLENCE
DRESDEN concept
TECHNISCHE UNIVERSITÄT DRESDEN
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Problem of recent In-Situ visualizations

Why in-situ at all?

- I/O bottleneck: Too slow for post and in-transit processing
  - max. $\sim 55\,\frac{\text{MB}}{\text{s}}/\text{node}$ to hard disk (Titan)
  - max. $\sim 6\,\frac{\text{GB}}{\text{s}}/\text{node}$ interconnect (Titan)

In-situ bottleneck?

- $\sim 16\,\frac{\text{GB}}{\text{s}}/\text{PCIe bus}$
  $\Rightarrow$ Especially with hardware accelerators

Real life problem? $\Rightarrow$ PIConGPU

- On GPUs with up to $10\,\frac{\text{timesteps}}{\text{s}}$ or even more
  - $> 60\,\frac{\text{GB}}{\text{s}}/\text{Node}$ on K20X

Related work

With C API

- libSim (VisIt)
- Equalizer
- Paraview Catalyst

Hdf5 compatible API

- ICARUS (Paraview)

In-Transit solutions

- Paraview Coprocessing

Why another in-situ library?

- Often data conversion needed
  - Many times with the need for deep copy
- PCIe or network speed as bottleneck
  - Especially for data on hardware accelerators (like Nvidia GPUs)

In Situ Animation of Accelerated Computations

ISAAC

In-Situ raycasting library

- Glowing gas and iso surface rendering
- Arbitrary clipping planes
- ...

Open Source (LGPL)

- https://github.com/ComputationalRadiationPhysics/isaac

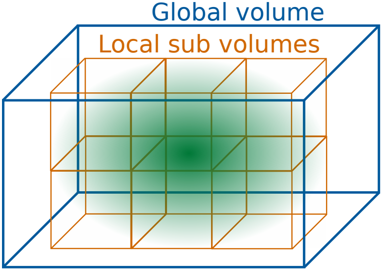
No deep copy needed

- Working on the simulation data structures
- Data conversion on the fly
- Works on arbitrary hardware accelerators

Abstraction of the high rate data source

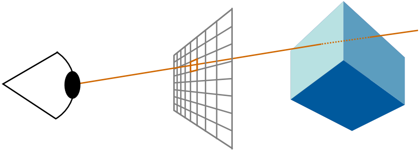
Simulation

- Describes a 3D volume
- Data in volume described with arbitrary amount of sources
- Every compute node has its own disjoint sub volume



Rendering

- Raycasting in local volume
- Blending of partial images via IceT



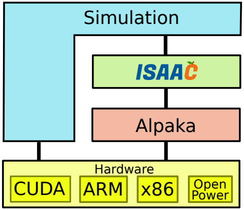
Source description and data access

Minimal interface for one source

```
struct TSource
{
  static const size_t f_dim = 1;
  static const bool has_guard = false;
  static const bool persistent = true;

  static std::string getName()
  {
    return std::string("Test␣Source");
  }

  void update(bool enabled, void* pointer) {}

  float_vec<f_dim> operator[] (const int_vec<3>& nIndex) const
  {
    float_vec<f_dim> result = { 42 };
    return result;
  }
};
```

Ray casting over unknown data

- Per sample step iterating over sources with template metaprogrammierung
- 2^n kernel for all combinations of activated sources
- Description of the render kernel with Alpaka [1]
  - Open source template library
  - CUDA-like C++ abstraction of arbitrary hardware accelerators

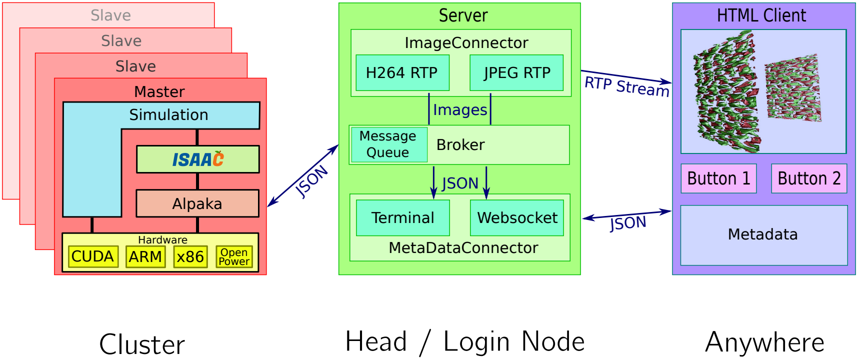


Zero overhead abstraction

Zero overhead abstraction

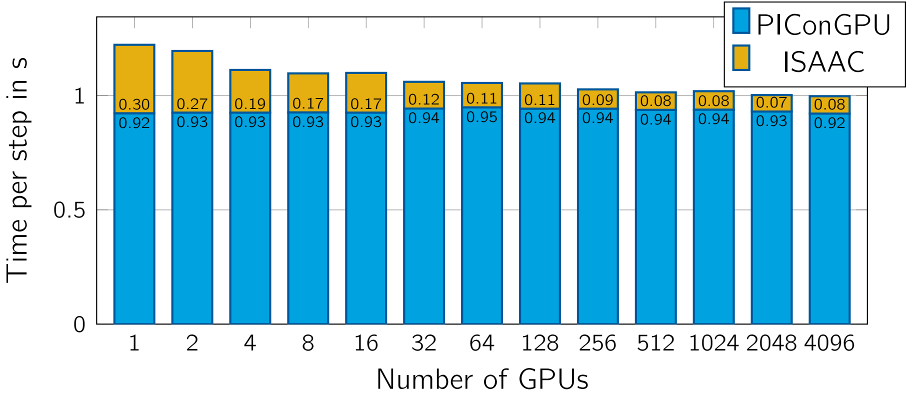
[1] E. Zenker et al. "Alpaka-An Abstraction Library for Parallel Kernel Acceleration." (2016)

Concept of ISAAC



Results

PIConGPU + ISAAC on Piz Daint

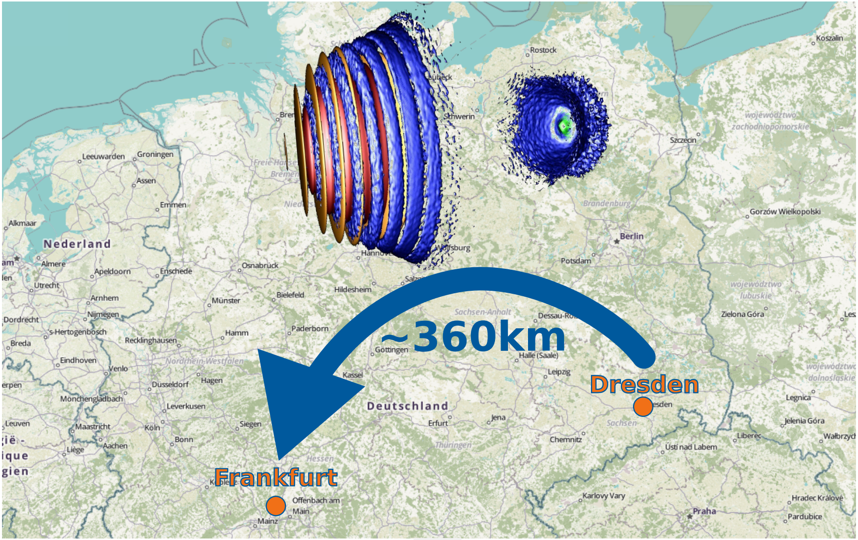


- Perfect weak scaling for PIConGPU [2]
- Strong scaling for ISAAC, because of shrinking sub images per node

[2] M. Bussmann et al. "Radiative signatures of the relativistic Kelvin-Helmholtz instability" (SC13)

Intel E5-2670, Nvidia K20X, 128^3 per node,
Kelvin-Helmholtz-instability, 2 sources, interpolation

Live demonstration

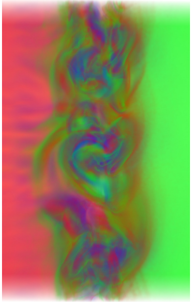


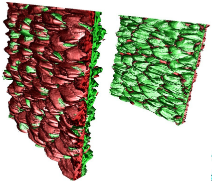
EUCALL | NVIDIA GPU CENTER OF EXCELLENCE | DRESDEN concept | TECHNISCHE UNIVERSITÄT DRESDEN | HZDR

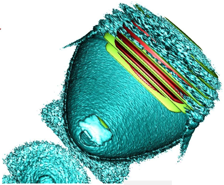
Summary

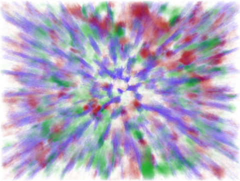
Features of ISAAC

- Live visualization
  - In-memory
  - Data agnostic
  - No deep copy needed
  - On arbitrary hardware accelerators
- Live steering
- Scales up to petascale



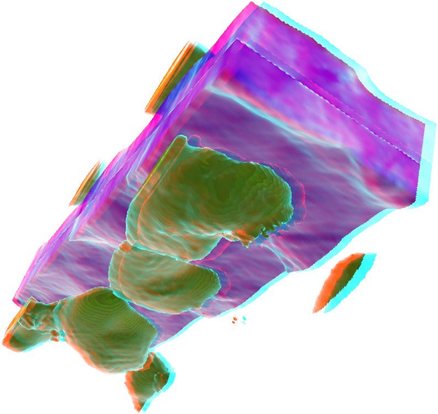






The end

Thanks for your attention!



Questions?