Live, steerable in-situ Visualization for high performance computing

Alexander A. W. Matthes

7th March 2016

High Performance Computing in data analysis

Live, steerable in-situ Visualization for high performance computing

- Increasing amount of high data rate sources
  - Need high performance computing to analyze this data

Where?

- On up to thousands of parallel compute nodes
- Connected with high performance network (e.g. Infiniband)
- More and more use of computation accelerators like
  - Nvidia GPUs (CUDA)
  - Intel XEON Phis

High Performance Computing: Hypnos

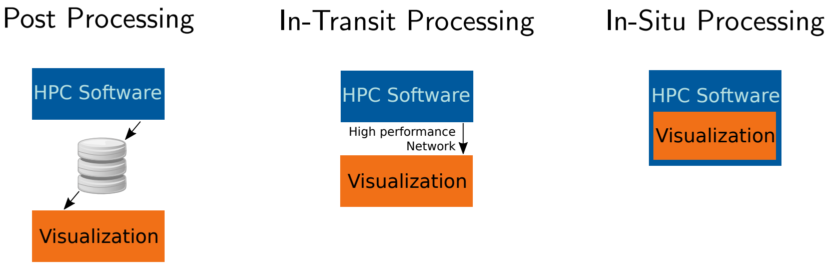
Live, steerable in-situ Visualization for high performance computing

- High performance cluster at the Helmholtz-Zentrum Dresden - Rossendorf
- 169 Compute nodes with
  - 552 CPUs
    (with $\sim$ 8.000 cores)
  - 164 NVidia GPUs as computation accelerators
    (with $\sim$ 360.000 cores)
- Whole peak performance: $\approx$ 597 TFLOPS
- 64 GB RAM per CPU
- $\sim$ 35 TByte total
- 5 GB RAM per GPU
- 820 GByte total

Why Visualization?

Live, steerable in-situ Visualization for high performance computing

- At some point the data have to be interpreted
- Much easier for humans to use pictures
  - instead of starring at pure data
- Also easier to explain phenomena



Advantages of In-Situ

Live, steerable in-situ Visualization for high performance computing

- Data processing can be as parallel as data analysis
- No network I/O is needed
- No storage I/O is needed
- No data copy is needed at all!
- Live view of the analysis data
  - Important for high rate data sources

New requirement: Steerable

Live, steerable in-situ Visualization for high performance computing

Needed for

- Steering the data analysis process
- Controlling the visualization

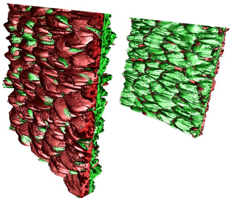
The same data channel can be used for:

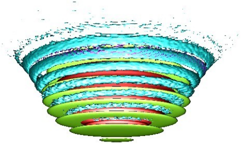
Meta data

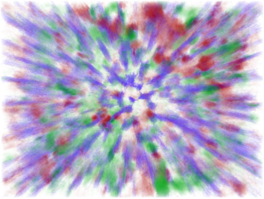
- From data analysis process (e.g. parameters)
- From visualization (camera position, render settings, etc.)

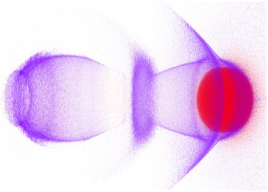
Our Solution: ISAAC

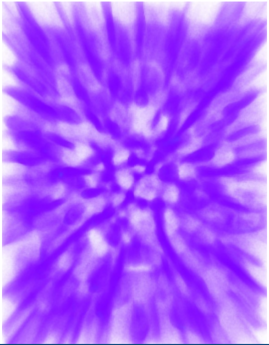
In Situ Animation of Accelerated Computations

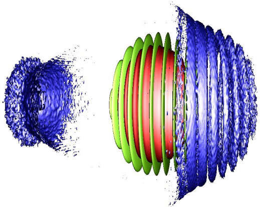












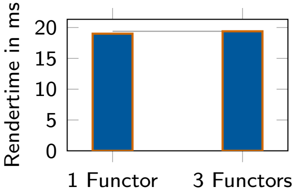
ISAAC: Key Features

In-Situ C++ Template Library

- Easy to add to existing high data rate sources
- With very application specific optimizations, because
- It works *on* the original data (instead of *with*)
- Works with accelerators like Nvidia GPUs or Intel Xeon Phis (using *ALPAKA*)
- Possible to transmit arbitrary meta data
  - from client to data source or visualization and vice versa
- Open Source (LGPL)

ISAAC: Direct Visualization Problem

- ISAAC works on original data
  - but maybe not suited for a direct visualization
- Some fast method needed for transforming the data



Functor Chain

- Functor: Small operator like addition, multiplication or length
- Can be concatenated to very simple transform functions
  - Example: length | mul(2) | add(0.5)

ISAAC: Distinction

Purpose

- Fast live visualization of big data
- Easy and expendable steering

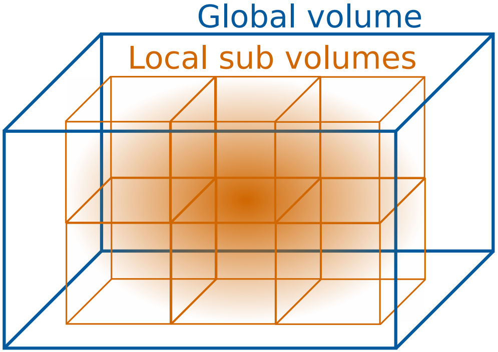
Not meant for

- Complex mathematical analytics
- High quality visualizations for publishing
- Transmitting big data out of the simulation

ISAAC Visualization: Abstraction

High data rate process

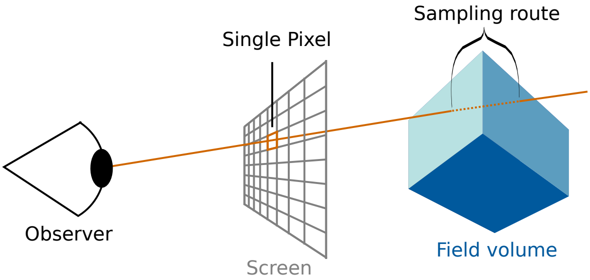
- Whole process describes a 3D volume
- Data in volume described by arbitrary amount of *Sources*, *which*
  - Define scalar or vector fields on this volume
- Every compute node has it's own disjoint sub volume
  - with local source data



ISAAC Visualization: Idea

Rendering

- Per compute node raycasting on the local sub volume
- Compositing of the partial images with IceT



ISAAC Visualization: Implementation

Compiletime settings

- High data rate process defines the sources as C++ classes
  - with some constant meta data, like the feature dimension
  - with a method to get data per volume position
- Used to precompile every needed accelerator kernel
- All functor chain combinations are precompiled, too

Runtime settings

- Abstraction: The global and local volumes
- Visualization: Camera settings, functor chains and so on...

ISAAC: Components

In-Situ Template Libary

- Draws images
- Sends and receives meta data to and from a server

ISAAC Server

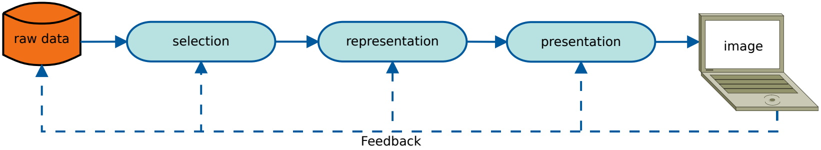
- Broker between library and clients
- Creates video streams from visualization images

HTML Client

- Rudimentary reference implementation
- Shows live meta data and video stream

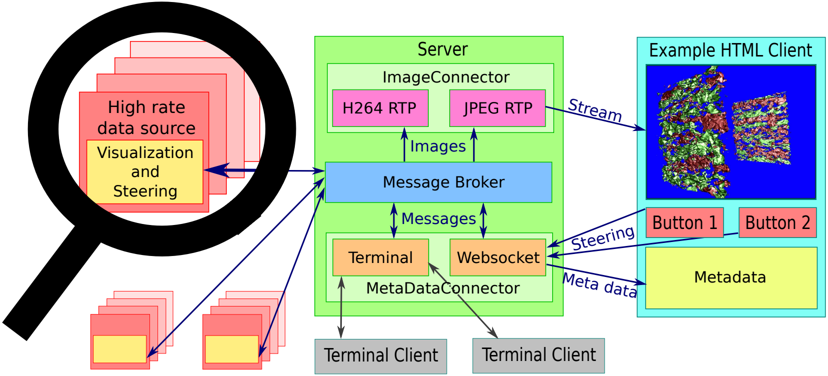
ISAAC: Visualization pipeline

- In-situ access of high rate data
- Source selection
- Functor chains
- Clipping
- Render settings
- Functor chains
- Camera
- Stream type
- Client type
- Arbitrary client



- Arbitrary steering information

ISAAC: Whole concept



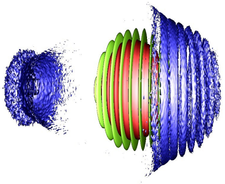
Evaluation with an Example

- Not many high rate data sources exist yet
  - Simulation used as example data source

PIConGPU

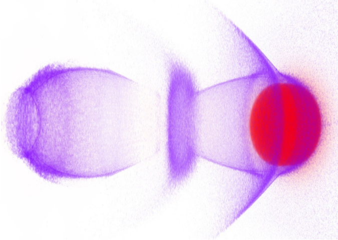
- Plasma simulation
- Developed at HZDR
- Distributed particle in cell algorithm running on GPUs
- Produces up to 5 GB (Hypnos) per time step
  - Per GPU!
  - Perfect high rate data source!

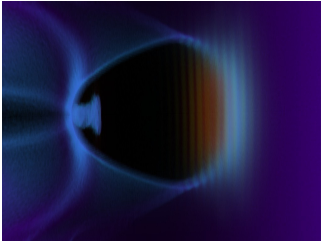
PIConGPU

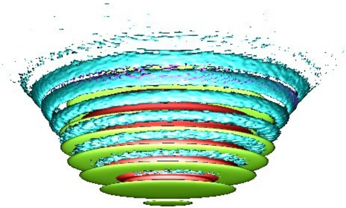


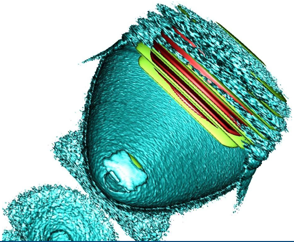
PIConGPU Live Visualizations

Laser Wakefield Accelerator



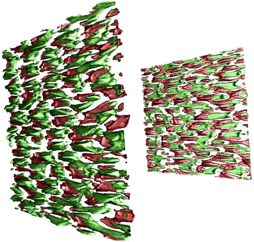


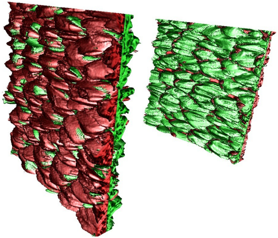


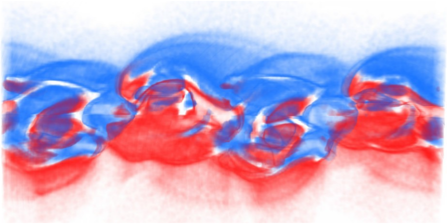


PIConGPU Live Visualizations

Kevin Helmholtz Instability

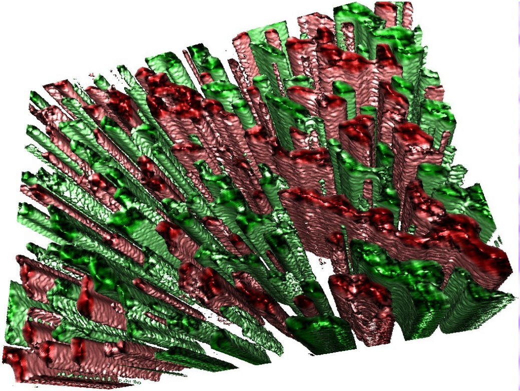


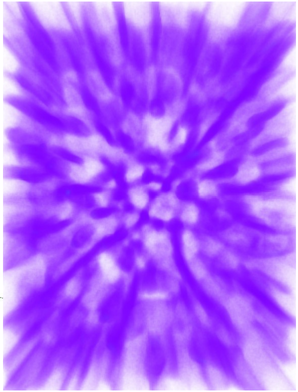




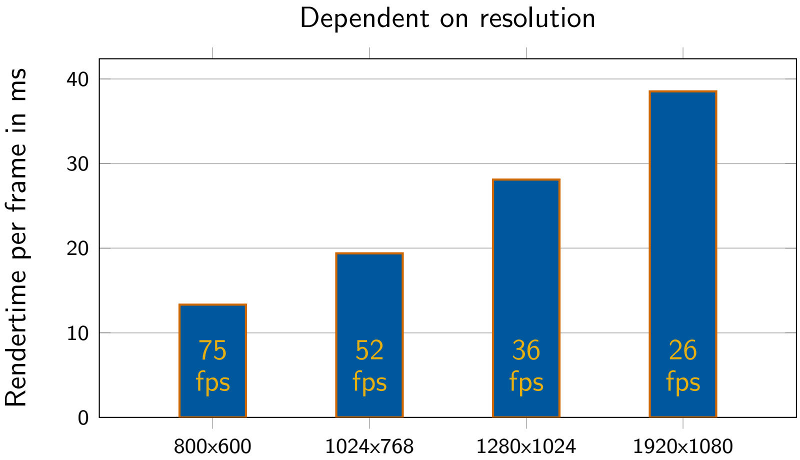
PIConGPU Live Visualizations

Two Stream Instability





ISAAC Performance



ISAAC Performance

Rendering (for PIConGPU)

- Laser Wakefield Simulation
- 64 GPUs
- 1.024×768
- $128 \times 64 \times 128$ volume per compute node
- No Interpolation

$\Longrightarrow \sim 20\ ms$

Scaling

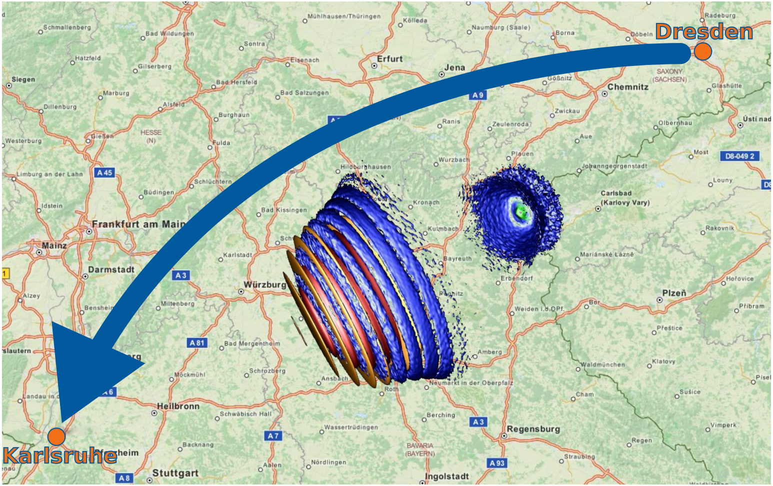
- Weak Scaling ✔
- Strong Scaling ✔

Latency

- HZDR Cluster $\longleftrightarrow$ Dresden

$\Longrightarrow \sim 175\ ms$

Live example



The End

Git repository:

https://github.com/ComputationalRadiationPhysics/isaac

Questions?