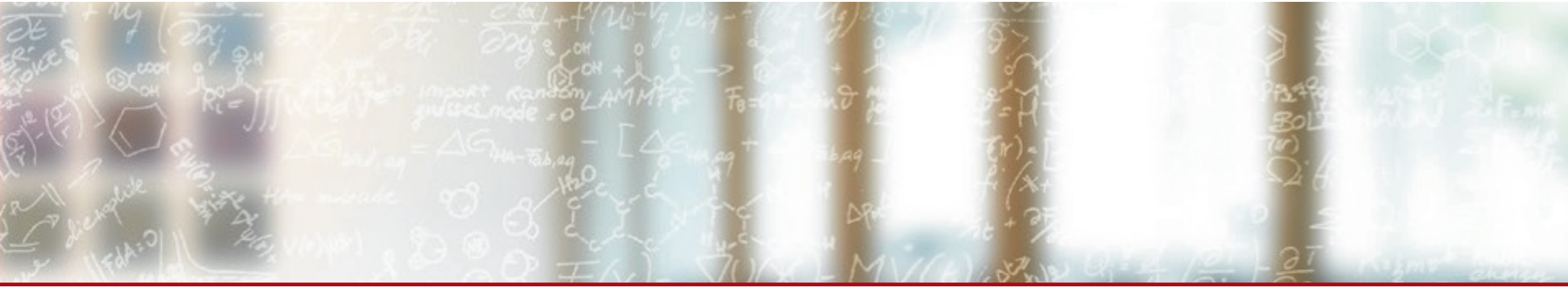


Scientific Visualization

User Lab Day 2018

Jean M. Favre, CSCS

September 11th 2018

Outline

- Recommendations about which file formats to use for 3D visualization
- Overview of the three most important Scientific Visualization Applications
  - ParaView
  - VisIt
  - VMD
- Usage scenarios
  - Desktop usage
  - Client-server
  - Batch-mode, server-side execution
- Listening to your wishes

Scientific Visualization and I/O libraries

The two topics are closely linked.

- Samuel Omlin has introduced the topic of parallel I/O

- I will follow with issues related to Scientific Visualization

Custom file formats for Visualization?

- Whenever possible, we recommend to **not** use custom file formats
- Why so?
  - It takes a considerable effort to make plugins for Visualization Applications
  - Portability across platforms would be an issue
  - Making the I/O parallel is non-trivial
  - You will most-often have to compile from source

Standard file formats

- Our three Scientific Visualization applications, ParaView, VisIt and VMD support several hundreds of file formats.

Use any one of the adopted file formats:

- Our task is to always have the most recent versions of the apps, finely-tuned for Piz Daint so that you can view the I/O problem as „Problem solved“
- The community is large. Bug reports and bug fixes are common.
- The web is full of resources, mailing lists, forums, etc.

- Our task is to guide you to select the best file formats for writing and reading your simulation data.

Libraries and Conventions

- Beware of the difference between using a library, and using an adopted convention.

- Said in other words: „Simply using HDF5 or NetCDF does not guarantee the Viz Applications will be able to read the data *out-of-the-box*“
  - Conventions, such as the CF convention for Climate Data would be the right choice.
  - Using XDMF to give the semantics of the data organization is one way to get access to HDF5 data
- In both of these cases, we are here to help you and guide you in making the right decisions.

Overview of three Scientific Visualization Applications

ParaView and VisIt

Similar paradigms of use:

- Desktop, client-server (parallel) and batch-only(parallel) execution
- Python-driven
- Many supported file formats in common

- Documentation:

https://user.cscs.ch/computing/visualisation/paraview/ Gallery

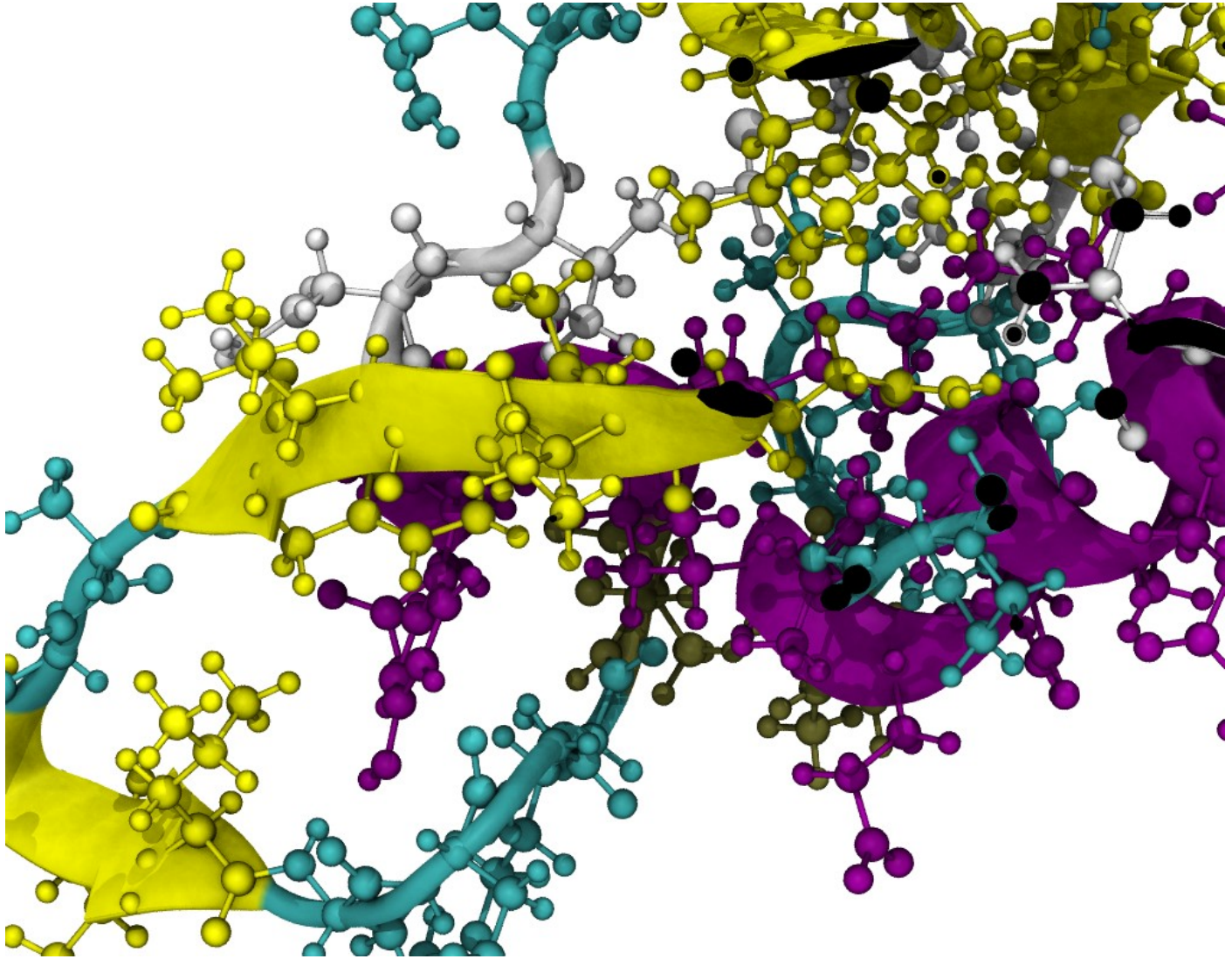
https://user.cscs.ch/computing/visualisation/visit/ Gallery

VMD

Users of CP2K, Quantum Espresso, LAMMPS, GROMACS, VASP, NAMD, NWChem, Amber, can take advantage of an optimized installation of VMD on daint.

Runs on compute nodes with GPU acceleration, in batch mode, or inside a remote VNC desktop.

https://user.cscs.ch/computing/visualisation/vmd/



Scenarios of usage

- We advocate the use of python scripts for all visualization activities, for desktop or parallel client-server runs

- Reproducibility
- Sharing – re-using
- Automation
- Fine-tuning for more effective resource allocation
- Many modules:
  - *import numpy as np* …… and more

Python-driven scripting for ParaView and VisIt

- The applications can save their state in Python.
  - It covers 100% of the *normal* usage
  - It covers 80-90% of the *advanced usage*

- Our contribution is often to augment python scripts with missing functionality.

New: ParaView for the multi-core partition!

- Huge improvements have been made for software rendering using Mesa
- LLVM
  - “The LLVM Core libraries provide a modern source- and target-independent **optimizer**, along with code generation support for many popular CPUs”

We compile Mesa with an LLVM backend for multi-core support and to fully utilize the modern instruction sets of Intel® SSE4, AVX, AVX2

Headless rendering, GPU-accelerated or LLVM-OSMesa rendering

Definitions to help your understanding:

- Headless rendering
  - On the server side, ParaView does not need to open a graphics window to do its rendering. No need for an X-server running on the compute nodes
- GPU-accelerated rendering
  - OpenGL on NVIDIA hardware with EGL
- OSMesa rendering
  - off-screen Mesa on the multi-core CPU

Resource allocation will differ

With GPU

- #SBATCH –constraint=gpu

module load ParaView/5.5.2-CrayGNU-18.08-EGL

Without GPU

- #SBTACH –constraint=mc
- #SBATCH --cpus-per-task=72
- #SBATCH --ntasks-per-core=2
-
- # Can use high-memory nodes
- #SBATCH –mem=122G

module load ParaView/5.5.2-CrayGNU-18.08-OSMesa

Resource allocation is automatic if using the documented procedures

With VisIt

- Choose host=daint to connect to the login node
- Select a file to open
- Choose the allocation needed
- The VisIt server process creates an executable submitted to srun
- Connection takes place and more files can be opened with the existing session.

With ParaView

- First, Open a connection with a resource allocation request
- Connection takes place when resource is allocated
- One or more files can be opened

Remote connection with python clients (VisIt)

With VisIt

```
from visit import *

Launch()

args = ("-sshtunneling", "-np", "24", "-nn", "1", "-l", "srun", "-dir",
"/apps/daint/UES/jenkins/6.0.UP04/mc/easybuild/software/Visit/2.13.0-CrayGNU-
17.08", "-t", "00:05:00", "-la", "--cpu_bind=sockets -C mc -p debug")

host = "daint101.login.cscs.ch"

OpenComputeEngine(host, args)

OpenGUI()
```

Remote connection with python clients (ParaView)

First Step, put the client in waiting mode

```
from paraview.simple import *

ReverseConnect(“1100”)
```

>>> Connection (csrc://hostname:1100)

(use your own id instead of, my private id (1100)

Second Step, execute the remote script creating the SLURM job

```
ssh -l jfavre -R 1100:localhost:1100
daint103.login.cscs.ch "/apps/daint/UES/
ParaView/rc-submit-pvserver.sh
pvserver 00:19:59 1 3 1100
daint103.login.cscs.ch GNU-5.5 debug;
sleep 180"
```

Remote connection with python clients (ParaView)

```
Temporary FileName is : /tmp/tmp.ZqTUSq2MsD
#!/bin/bash
#SBATCH --job-name=pvserver
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=3
#SBATCH --ntasks=3
#SBATCH --time=00:19:59
#SBATCH --partition=debug
#SBATCH --constraint=gpu
module load daint-gpu
module load ParaView/5.5.1-CrayGNU-17.08-EGL


srun -n 3 -N 1 --cpu_bind=sockets pvserver -rc
-ch=daint103.login.cscs.ch -sp=1100

Submitted batch job 9565038
```

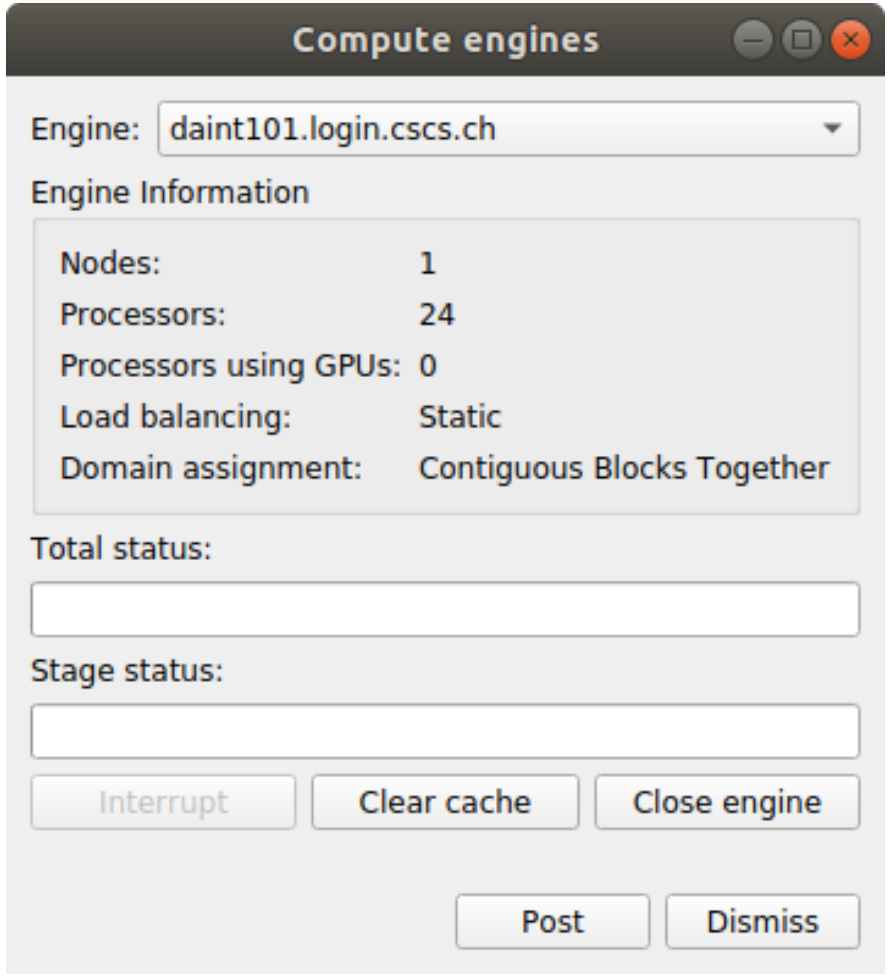
Testing from your client:

```
Sphere()

pid = ProcessIdScalars()
Show()
Render()
```

Once connected to a remote session...

With VisIt

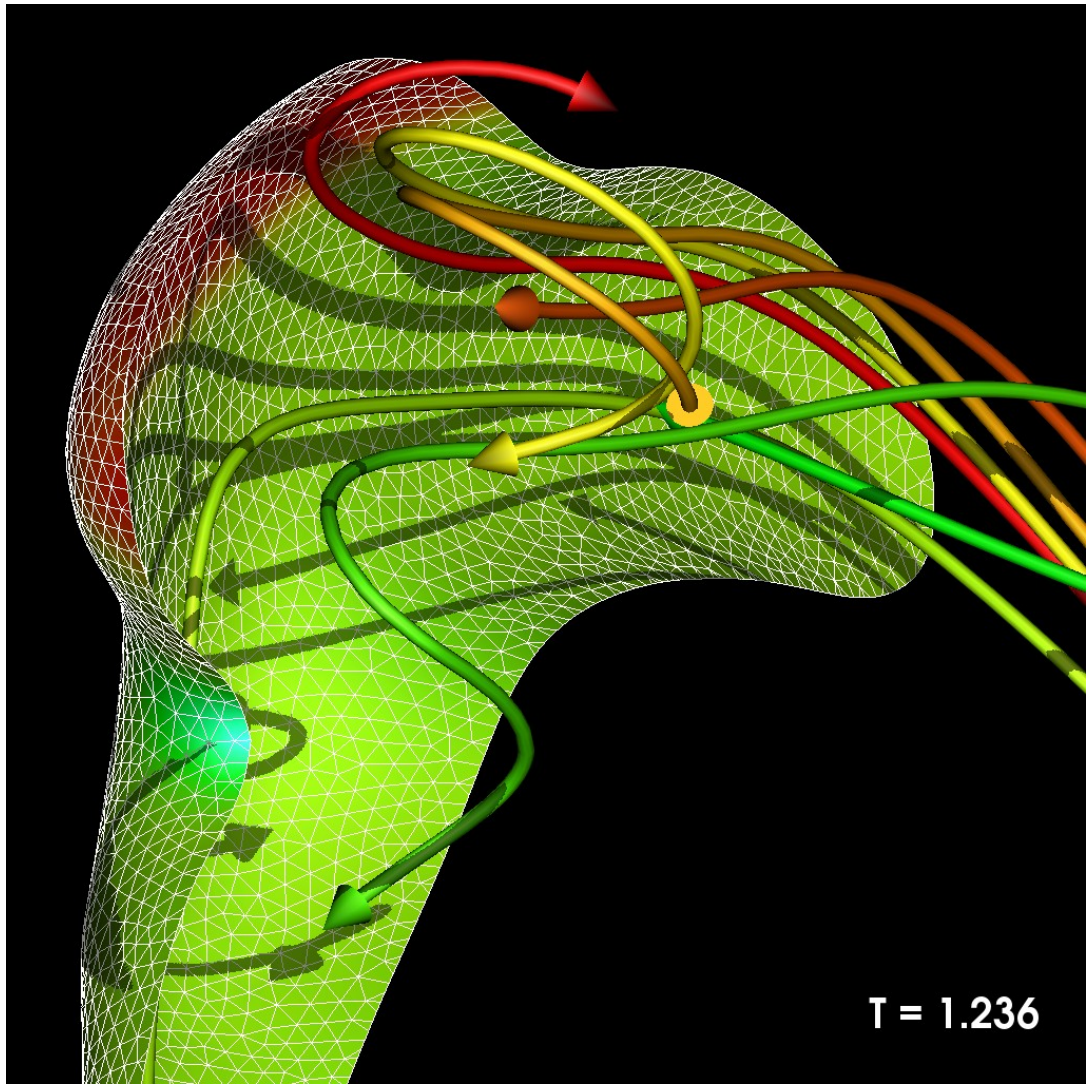


With ParaView

Motivational Examples

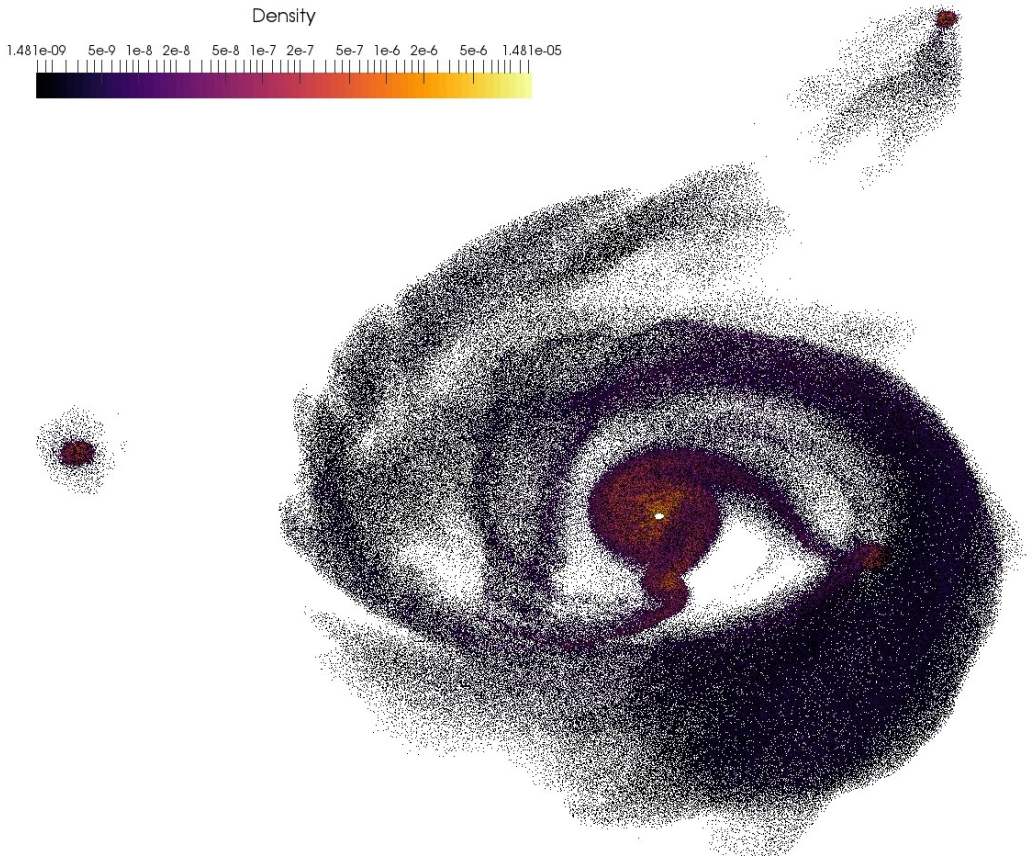
- From raw data to interactive visualization
-
- Simply loading the raw data results in in an image with little appeal.

Motivational Examples



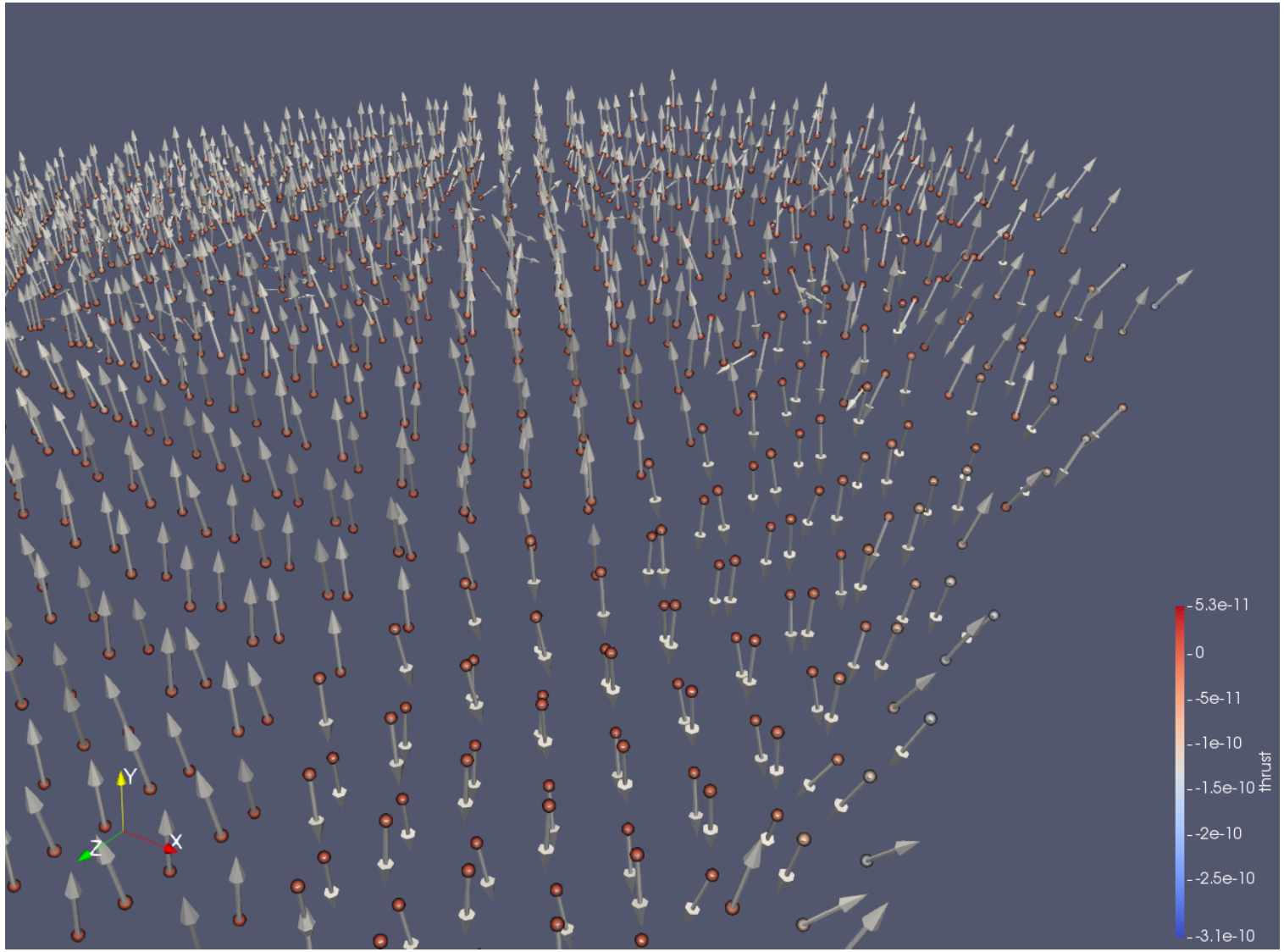
- Increased perception with tricks and advanced rendering

Motivational Examples



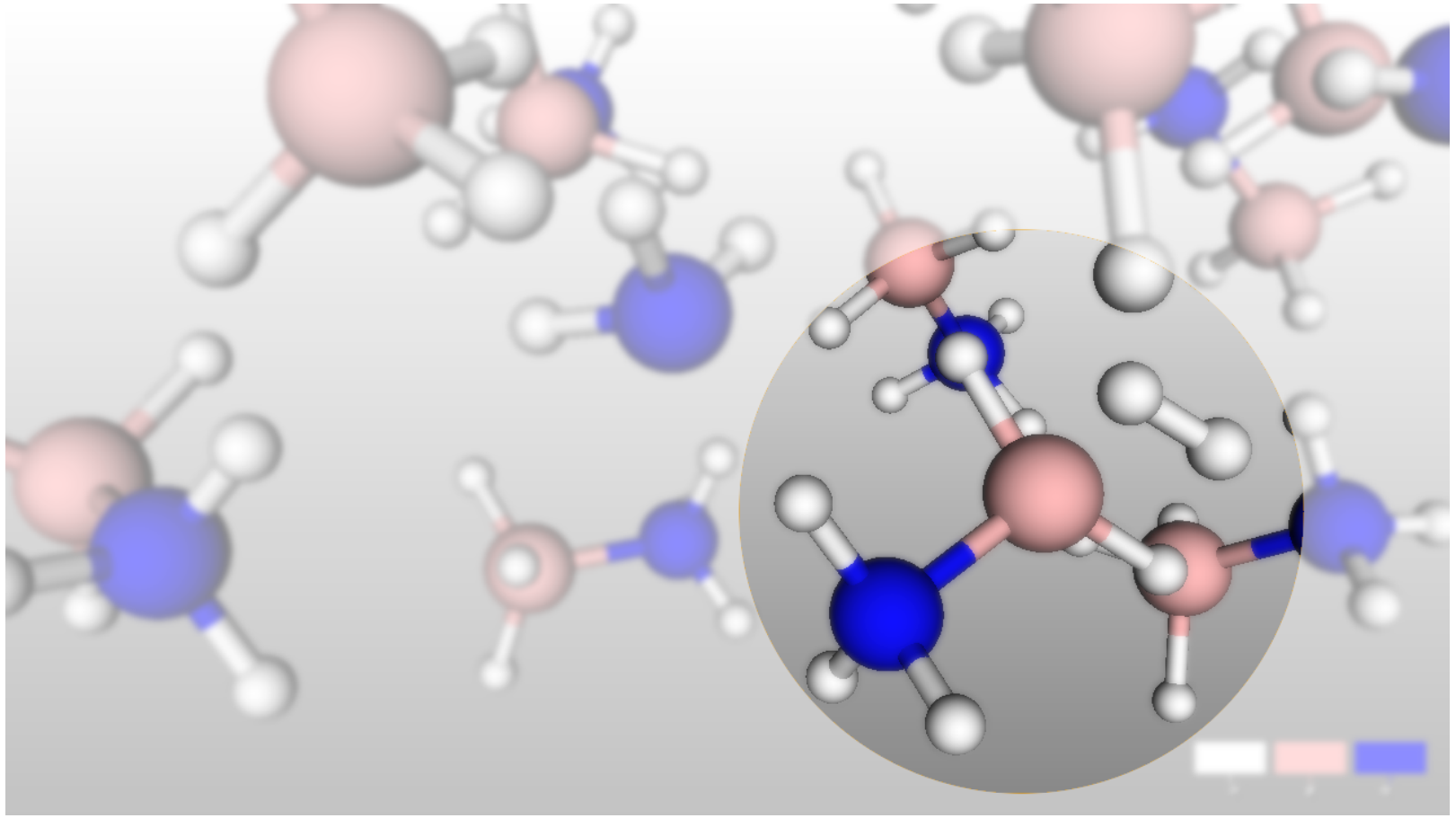
- Benchmarking of Point Cloud Processing and Rendering

Motivational Examples



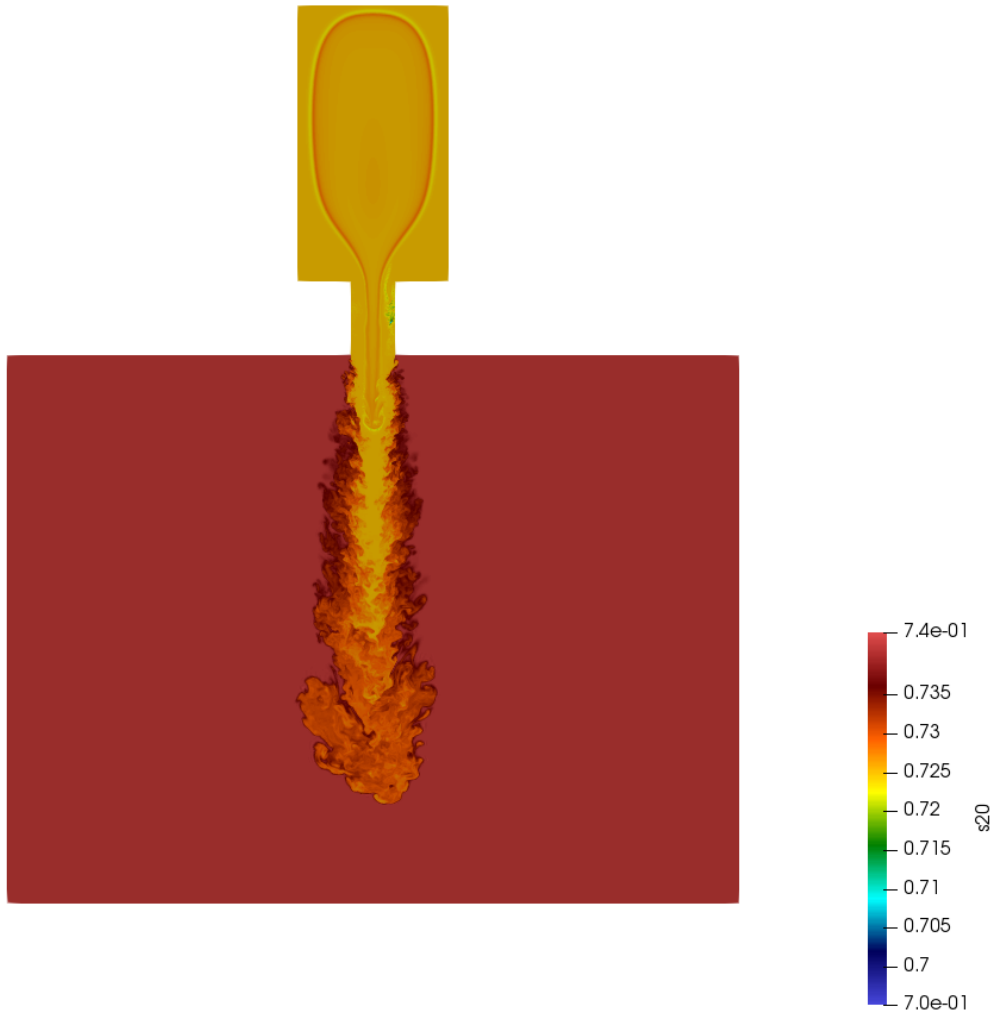
- Visualization as a debugging tool
- A process that has lasted several months during multiple versions of the code development

Motivational Examples



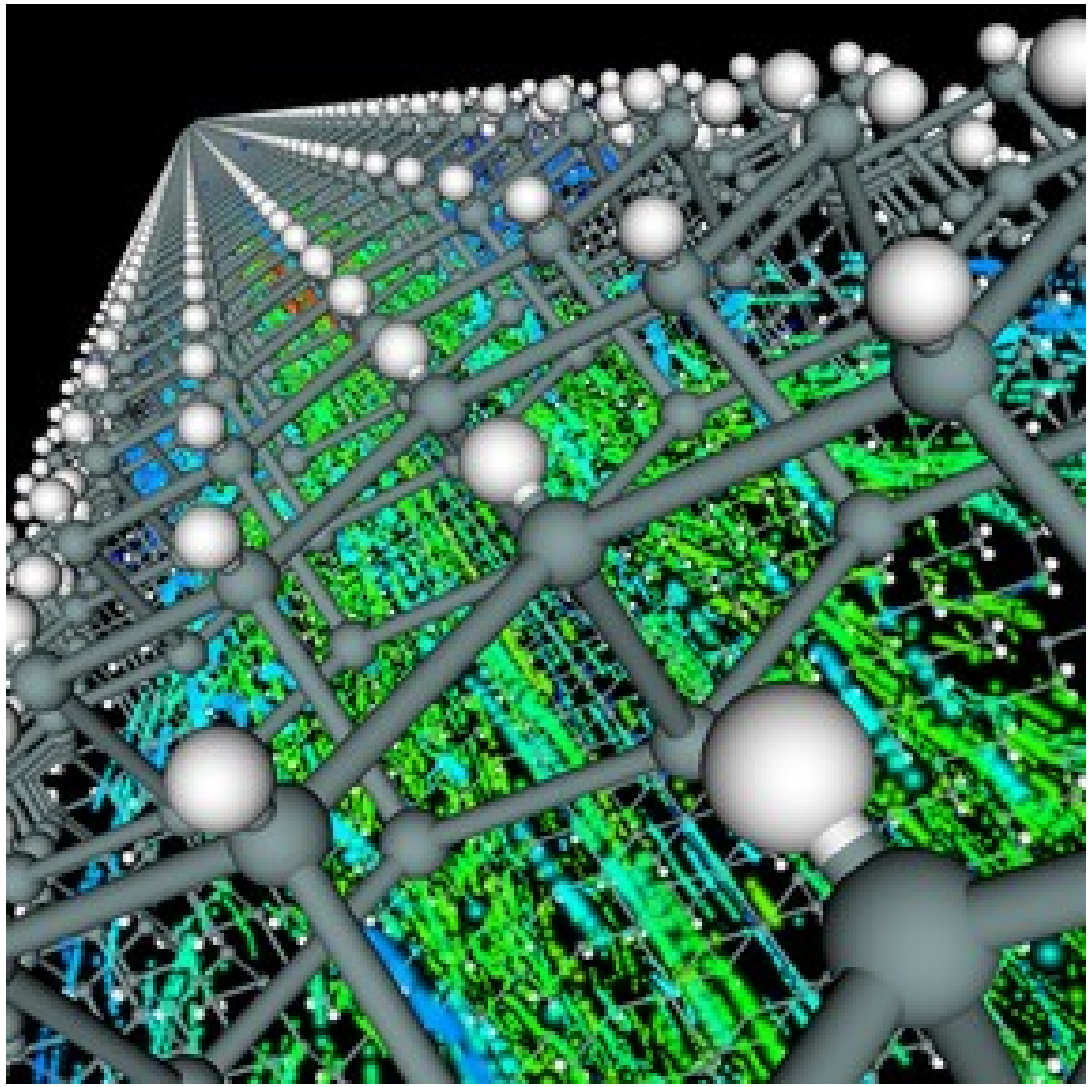
- Very simple, small dataset
- Very confusing to see in 3D
- Enhancing the traditional displays…
- Work in progress...

Motivational Examples



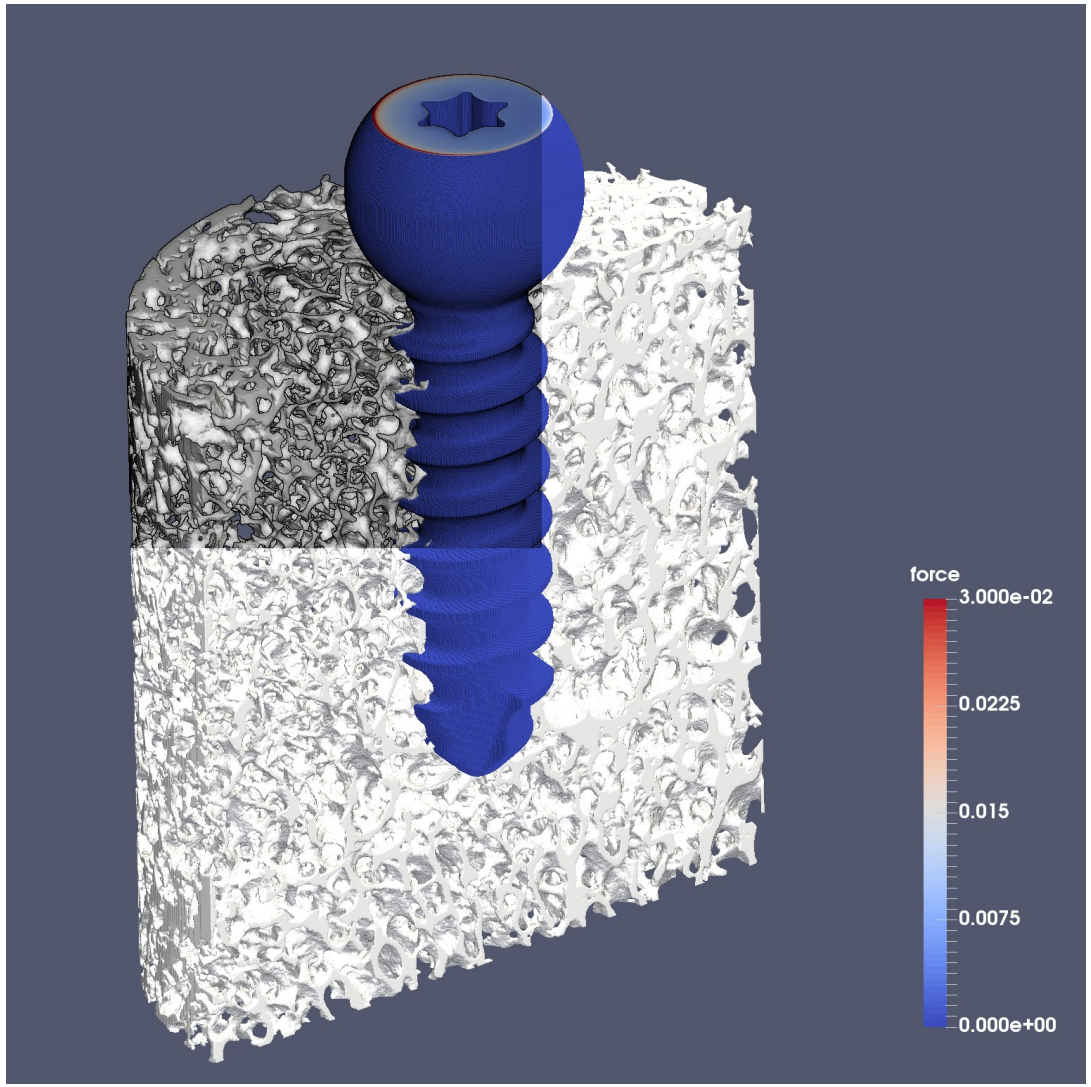
- A bug in the viz application was only discovered when testing with Very large data (near a billion cell)
- We found the bug, developed a patch, installed the patch on Piz Daint, contributed the patch to the open-source dev tree
-

Motivational Examples



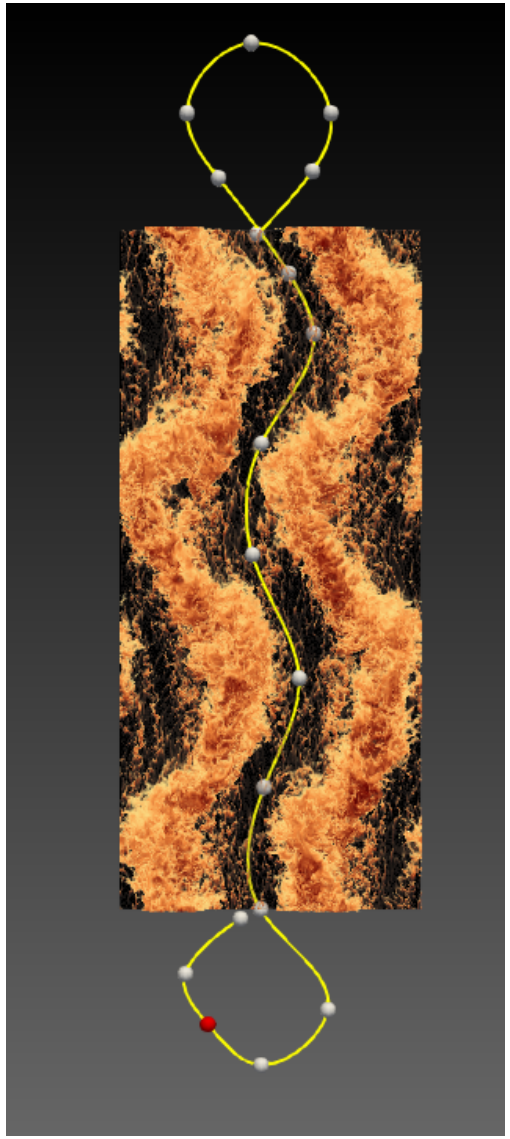
Mixing molecular rendering with traditional structured grid rendering called for a custom viz mini-app

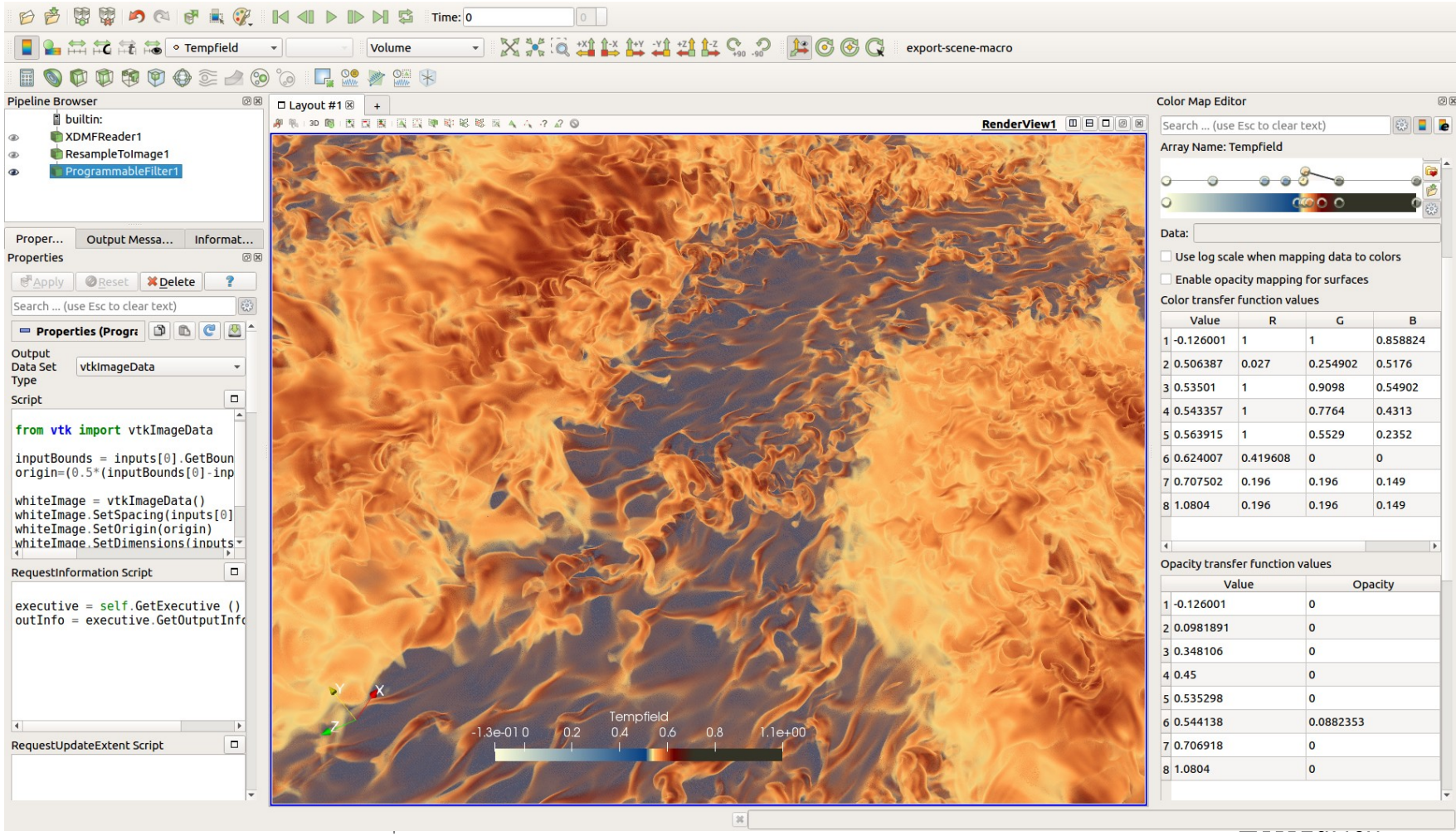
Motivational Examples



- Enhancing 3D perception...

Motivational Examples





CSCS

Soon available on Piz Daint

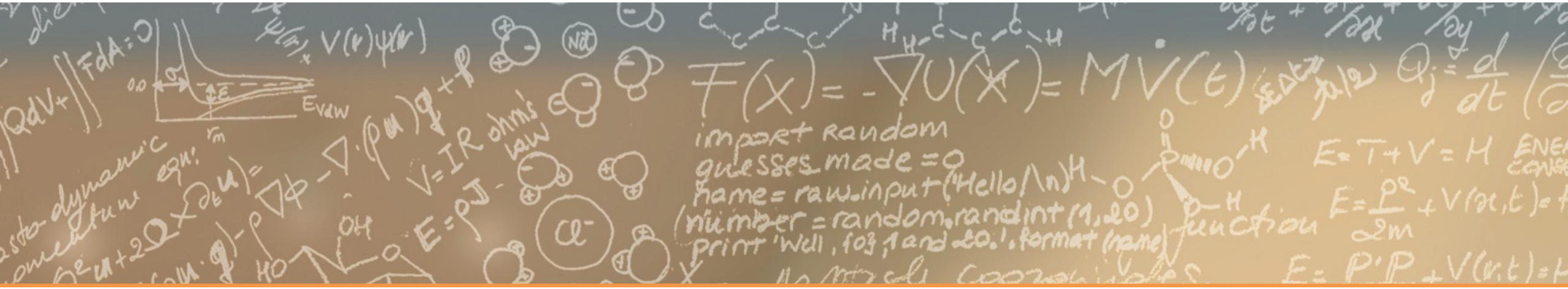
ParaView v5.5 with OSMesa for the MC partition

ParaView v5.6 for the GPU and MC partitions

Summary

- A great deal of experience has been collected on advanced and efficient use of data (file) storage to enable, and accelerate I/O for Visualization
- Likewise, daily practice with a few Visualization applications gives us an added advantage
- Advanced scripting knowledge is available to make the best of the tools
- Nearly fully-automatized client-server connections to viz tools (ParaView and VisIt) are offered as a service

- Consult with us
- We are here to assist you.
- vis-rt@cscs.ch


import random
guesses_made = 0
name = raw_input('Hello! What is your name?\n')
number = random.randint(1, 20)
print 'Well, ' + name + ', I am thinking of a number between 1 and 20.'