# *Remote and in situ visualization: why is relevant?*

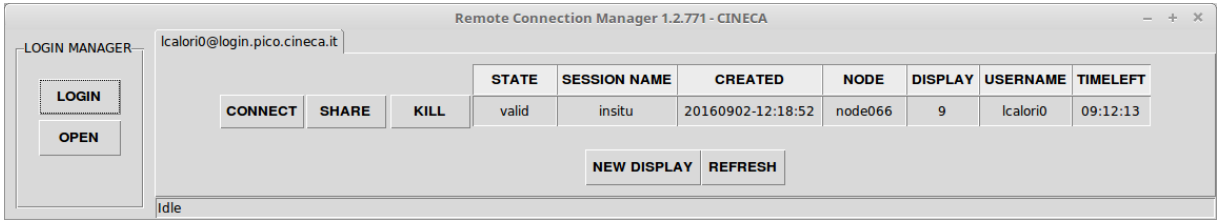
- researchers need to visualize the data produced by numerical simulations
  - simulation -> bulk data transfer -> post-processing -> visualization
  - simulation data sizes scale faster than available end - user transfer bandwidth
  - maximize interactivity to decrease insight turnaround
  - simplify the steps needed to effectively use visualization services and tools
- use available resources (viz nodes with GPUs) for interactive postprocessing
- needs to experiment the advanced coupling between simulation and visualization: in-situ visualization techniques;
- visualization services are still not well known or intimidating for the average user (SSH tunnelling, VNC client installation, graphics resource reservation, parallel rendering setup, simulation code instrumentation).
- no publicly available collection of detailed installation and deployment recipes.

# *VNC: Remote desktop on cluster visualization infrastructure*

- Protocol to transparently transfer control flow information from user workstation and to transfer back remotely rendered screen dumps.
- The protocol try to optimize available bandwidth by server side compressing of screen data updates  and corresponding de-compresing on user client device
- System should be (completely) application transparent allowing any GUI application to behave as if the user have been at the phisical screen intercting with a keyboard and mouse physically connected to the remote host.
- VNC handles 2D X11 GUI applications, for remote visualization of OpenGL based application, one techniques is to render on the host GPU, grab screen buffer and pass it through VNC protocol.
- One of the mot used Open Source stack is TurboVNC / VirtualGL

# VNC connection made simple: Desktop sessions on clusters

Due to large adoption amongst HPC centers and as well as within Prace partners, the open source stack VirtualGL / TurboVNC has been selected as basis for the prototype for a remote visualization service.
The basis for the current implementation is the RCM project, developed in Cineca, that wrap standard TurboVNC client, ssh tunneling into a single python executable implementing a GUI to help session bookkeeping
The prototype consist into a cross-platform python client that bundle TurboVNC client, ssh tunneling and a Python Tk GUI interface.



# *Remote Connection Manager: client components*

- **VNC client ( TurboVNC)**: is the (cross platform) client component that handle all the user interaction on the user device, it open a window on the device, forward all input events to the server and render the returning image stream.
- 
- **SSH tunneling transport** ( in latest version of TurboVNC is embedded )
- 
- **Thin python layer** that wraps components and auto-extract them on the fly, it does session book-keeping:
  - *Initialization and configuration*
  - *listing and reconnection*
  - *ending*
  - *sharing*

# RCM: server components

- **VNC server ( TurboVNC)**: is the process that run on the visualization node, handling input event forwarded by client:
  - *Implements a virtual X11 context for children applications.*
  - *It grabs the X11 virtual screen buffer, compress it and send the image stream back to the client ( VNC protocol )*
  -
- **Python RCM server:** this is a thin python wrapper around underlying queuing system. It provides to corresponding RCM client layer an abstracted view of
  - *Session information storage ( currently text files in user $HOME/.rcm)*
  - *Predefined session profiles ( site configurable with .ini files )*
  -
- **VirtualGL interposing library:** this component is needed when the visualization node has a graphics board providing 3D accelerated OpenGL:
  - *provides OpenGL accelerated context ( GFX ) to 3d applications using ld-preload technique and an interposer script ( vglrun )*
  - *handle interaction with the real accelerated X11 server on viz node*

# RCM: security

- **authentication policies to be supported**:
  - *user / password, (current, complete)*
  - *ssh keys (partial, planned)*
  - *Other ?*
  -
- **network restrictions**
  - *Need of Virtual Private Network (to be bundled with the client ?)*
  - *IP address white-list ( do we have to care ?)*
  -
- **Security assessment of RCM client components**
  - *Is it safe to rely on externally provided binary code in distributed client?*
  - *Build environment currently manual, some on Virtual Machines. Do we need to automate and version? ( early testing )*
  - *Official code signing ( MS windows ) for better user experience ( not done )*
-

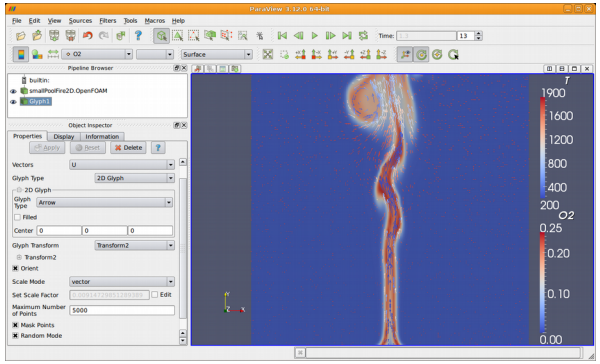
# RCM: resources allocation

- Policies of resource allocation for interactive usage
  - **interactive use generate largely variable load**:
    - *Almost no load in an inactive session*
    - *Medium load for heavy compilation sessions*
    - *Up to full node load when post-processing on viz node*
  - **job schedulers** tuning and **OS requirements**:
    - *Interactive sessions requires fast response and usually not much resources, so they likely requires different parameters and possible over-allocation of cores.*
    - *If running on graphics accelerated nodes, VirtualGL currently require ability to open X session on the X11 accelerated server.*

# RCM: Desktop applications

- **Window manager** ( Gnome, KDE, Fluxbox): is the process that handle the desktop management, usually one is already available as system package on viz nodes. Lightweight ones like Fluxbox could be preferred as consume less resources on the Viz nodes hosting sessions.
- 
- **Pre-installed applications of common usage**
  - Requiring OpenGL acceleration ( must be launched with vglrun )
    - *Open source :* ***Paraview, Visit, VMD, Blender***
    - *Independently licensed:* ***Tecplot, Matlab***
    - *Postprocessing tools of larger suites:* ***Starccm,Pointwise,Abaqus.***.
  - Not OpenGL based ( do not require vglrun )
    - *Open Source*
      - *Dev tools:* ***Qt creator, PyCharm***
      - *GIS tools:* ***Qgis***
    - *Debuggers:* ***TotalView***
    -

# ParaView



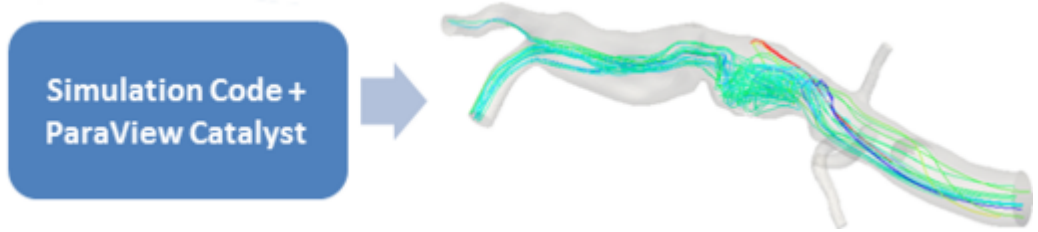
- **ParaView** www.paraview.org from kitware is one of the most used general purpose parallel visualization tools, is a based on VTK toolkit, implement a simplified data-flow paradigm, allowing for variety of file format readers, data processing filters and visualization algorithms
  - Can be used as a desktop Qt application, scripted in python
  - If MPI enabled can exploit data processing and rendering parallelism on HPC clusters
  - Support web access
  - Allow In-situ work-flow by the Catalyst library.
  -

# *In situ: background*

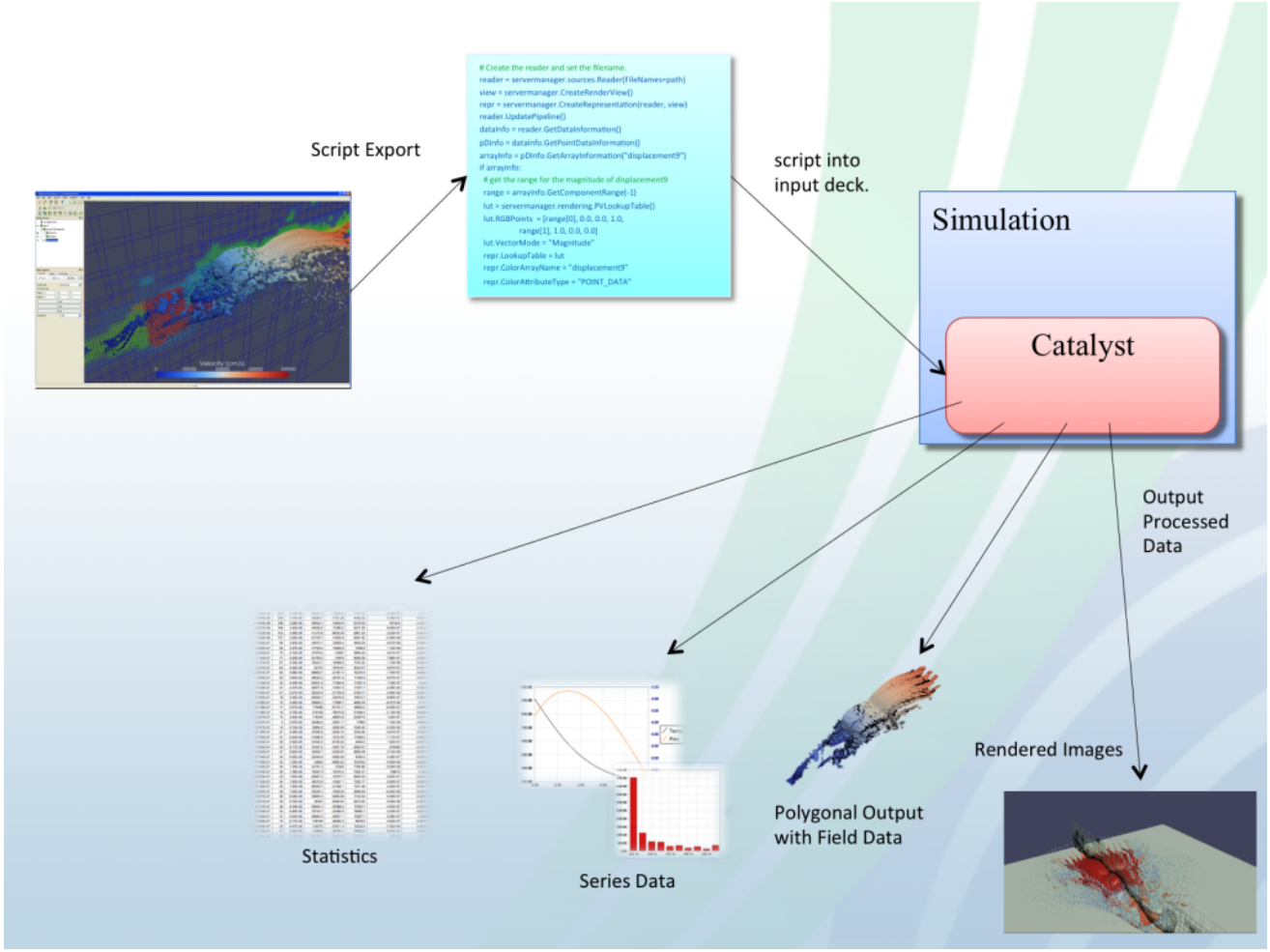
- Computing power increase rate is much higher than Input/Output
- Data size increase
- Saving / reloading simulation  data could become a bottleneck
- Shift in work-flow:  move (part) of post--processing inside simulation
-

$$\rho\frac{D\vec{v}}{Dt} = -\nabla p + \mu\nabla^2\vec{v}$$

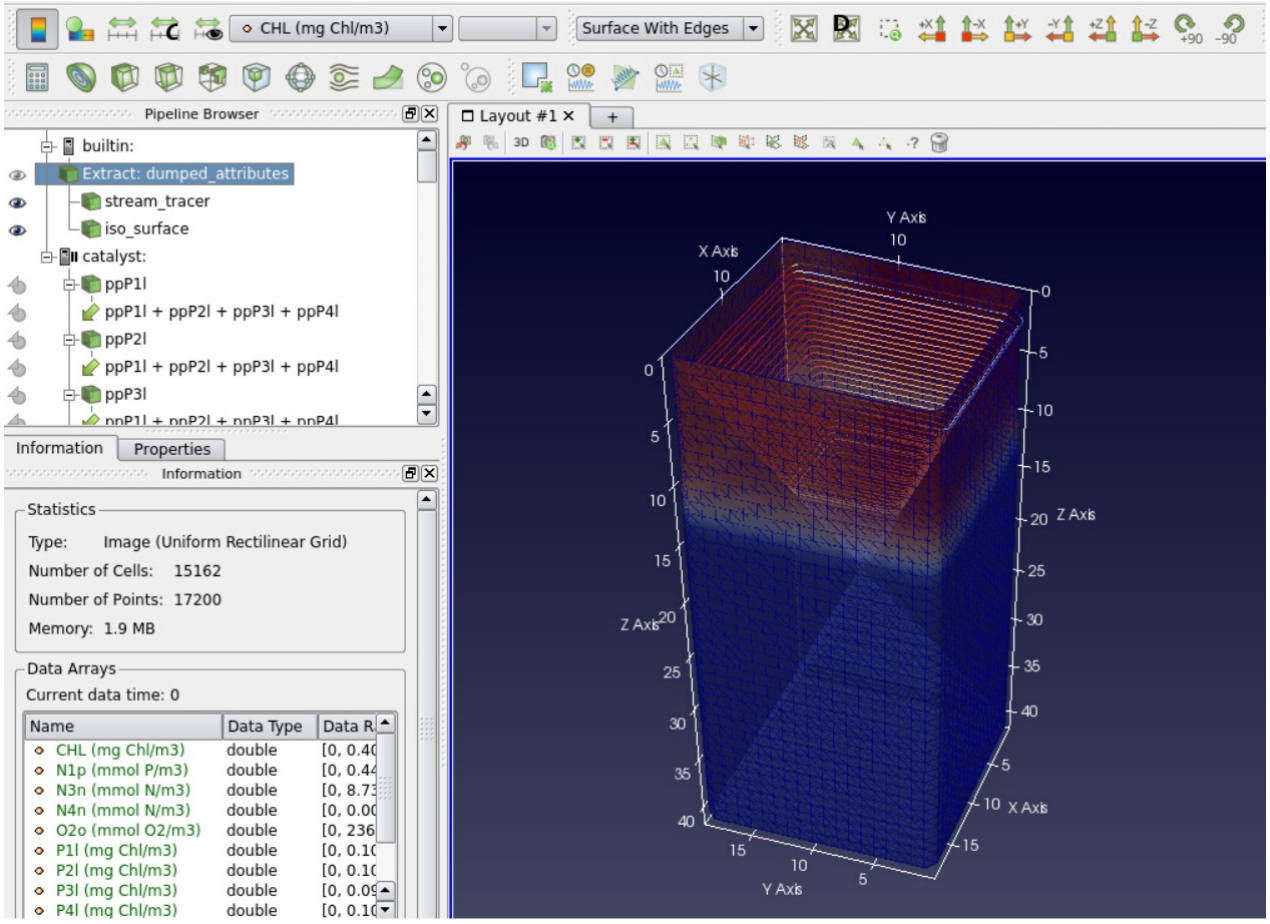

$$\rho\frac{D\vec{v}}{Dt} = -\nabla p + \mu\nabla^2\vec{v}$$



# Paraview Catalyst: structure



# Paraview with Catalyst



Links www.paraview.org

Webinars and videos:
http://www.paraview.org/webinars/
https://vimeo.com/120504264

In Situ and Cztalyst overview
http://www.paraview.org/in-situ/
http://www.paraview.org/catalyst-adaptors/
http://www.paraview.org/Wiki/ParaView/Catalyst/Overview
https://blog.kitware.com/paraview-catalyst-enabling-in-situ-analysis-and-visualization/

Examples
https://gitlab.kitware.com/paraview/paraview/tree/master/Examples/Catalyst
https://blog.kitware.com/anatomy-of-a-paraview-catalyst-python-script/

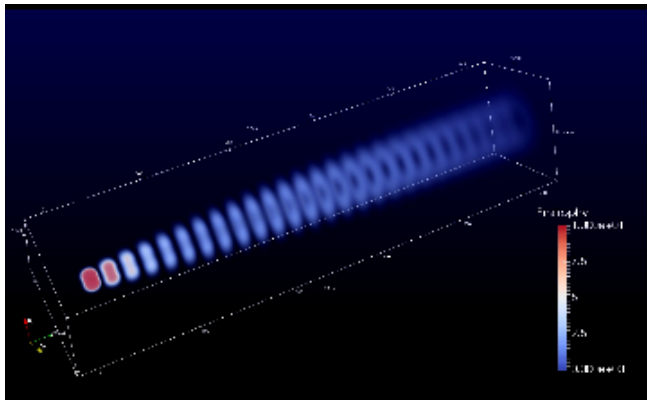
User Guide
http://www.paraview.org/files/catalyst/docs/ParaViewCatalystUsersGuide_v2.pdf

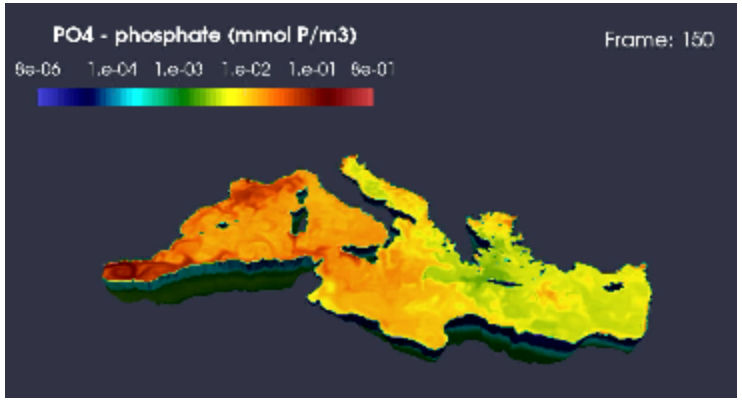
Source: https://gitlab.kitware.com/paraview/paraview

# Scale up visualization*: In-Situ experiments*

**Evaluation of ParaView Catalyst library for instrumenting two simulation codes with In-Situ visualization.**

•



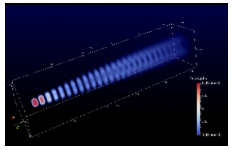


# *In-Situ experiments*

Two simulation codes have been selected, for adding In-Situ on-the fly and batch visualization capabilities, both were Fortran MPI codes with similar simulation grids.

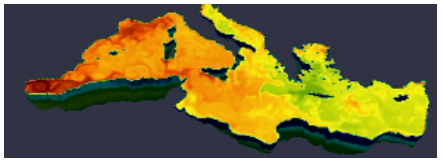
*Baseline CFD simulation of extropy effects possibly leading to tornados*

SoHPC video



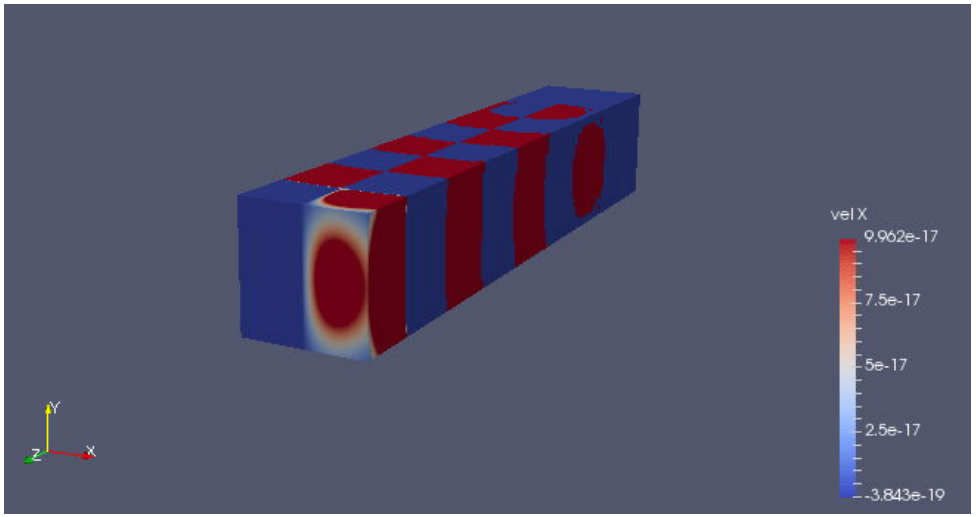
*Forecast of nutrient dispersion in the Mediterranean basin*
*Code in daily production in CINECA*

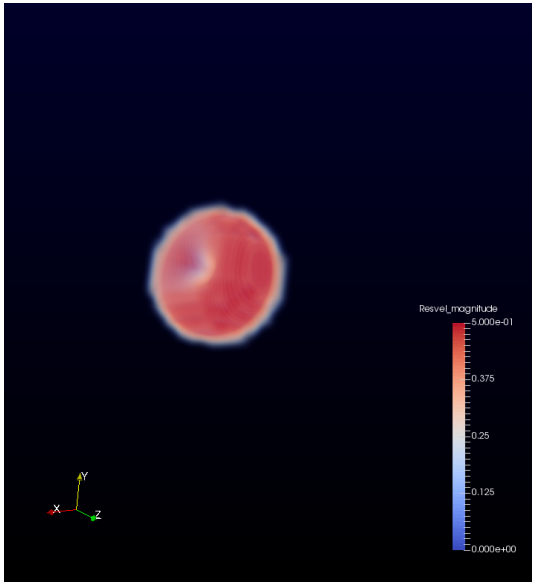
SoHPC video



Paraview Catalyst has been used for both experiments
No prior 3D visualization was available.
These In-Situ experiments were carried on under the framework of HPC-Summer.

# *Catalyst experiments: instrumentig simulation code… a bug in adaptor: wrong mapping to paraview*




blog SoHPC