NVIDIA IndeX for ParaView Plugin

User's Guide

5 September 2018
Version 2.2

NVIDIA

Copyright Information

Document build number unknown

Contents

1 Introduction

The NVIDIA® IndeX™ for ParaView® Plugin enables large-scale and high-quality volume data visualization capabilities of the NVIDIA IndeX library inside Kitware's ParaView.

This document is intended for a Paraview user who is new to the IndeX plugin and wants to explore the features of the IndeX library supported by the plugin. The following sections will explain the installation procedure and various features the plugin supports, followed by a section of frequently asked questions and useful links for further reference.

If you haven't downloaded the plugin yet, you can do so from this URL:

http://www.nvidia.com/object/index-paraview-plugin.html

2 Licensing

The NVIDIA IndeX for ParaView plugin comes with a free license that enables exploiting the capabilities of a single GPU. If you aim to use plugin on a cluster of multiple hosts and/or with multiple NVIDIA GPUs, then please contact us for appropriate licensing via email to `indexintegration@nvidia.com`. Users will be notified via email whenever a new version of the plugin is released.

3 Installation

NVIDIA IndeX for ParaView plugin is delivered with ParaView v5.5.0 and supports both Linux and Windows x86-64 platforms. Follow the installation instructions specific to your platform and install ParaView.

If you have installed ParaView v5.5.0 or later from binaries, the plugin is already included and you can continue to section 3.2, Loading the plugin (page 3).

3.1 Building the plugin

Please follow these steps to build the plugin matching your build environment.

1. You can download the ParaView source code from ParaView website[1]
2. Run `cmake` on your ParaView source tree and the plugin sources will be added to the ParaView plugins list to be compiled. Make sure you have the cmake option set, enabled by default in the plugin source code.

```
mkdir paraview_build
mkdir paraview_install
cd paraview_build
```

```
ccmake ../paraview-source-root
```

```
PARAVIEW_BUILD_PLUGIN_pvNVIDIAIndeX=ON
```

[1] `https://www.paraview.org/download/`

3. Run `make` or `make install` from your ParaView source tree. The plugin will be compiled together with ParaView.

   ```
   make paraview-binary-root
   ```

4. The plugin requires the NVIDIA IndeX libraries package. The package is not distributed with ParaView sources but can be downloaded from the ParaView dependency repository: https://www.paraview.org/files/dependencies/

   Linux
   : Download `nvidia-index-libs-2.1.20180314-linux.tar.bz2` and uncompress it to a folder of your choice. Update your `LD_LIBRARY_PATH` environment with the library path of your newly created folder: *new-folder*`/lib`

   Windows
   : Download `nvidia-index-libs-2.1.20180314-vs2013-x64.tar.bz2` and uncompress it to a folder of your choice. Update your `PATH` environment with the library path of your newly created folder: *new-folder*`\lib`

For additional information about compiling ParaView please refer to the instructions on the ParaView website[2].

3.2 Location of the plugin

In a ParaView installation, the plugin binary is located in the following directories:

Linux
: *your-paraview-installation-directory*`/lib/paraview-`*version*`/plugins/pvNVIDIAIndeX/`

Windows
: *your-paraview-installation-directory*`\bin\plugins\pvNVIDIAIndeX\`

The plugin binary will automatically show up in the [Tools ▸ Manage Plugins] menu.

[2] `https://www.paraview.org/Wiki/ParaView:Build_And_Install`

3.3 Loading the plugin

Follow the following steps to load the plugin in ParaView.

1. Start ParaView client and navigate to [Tools ▸ Manage Plugins] option from the menu bar.

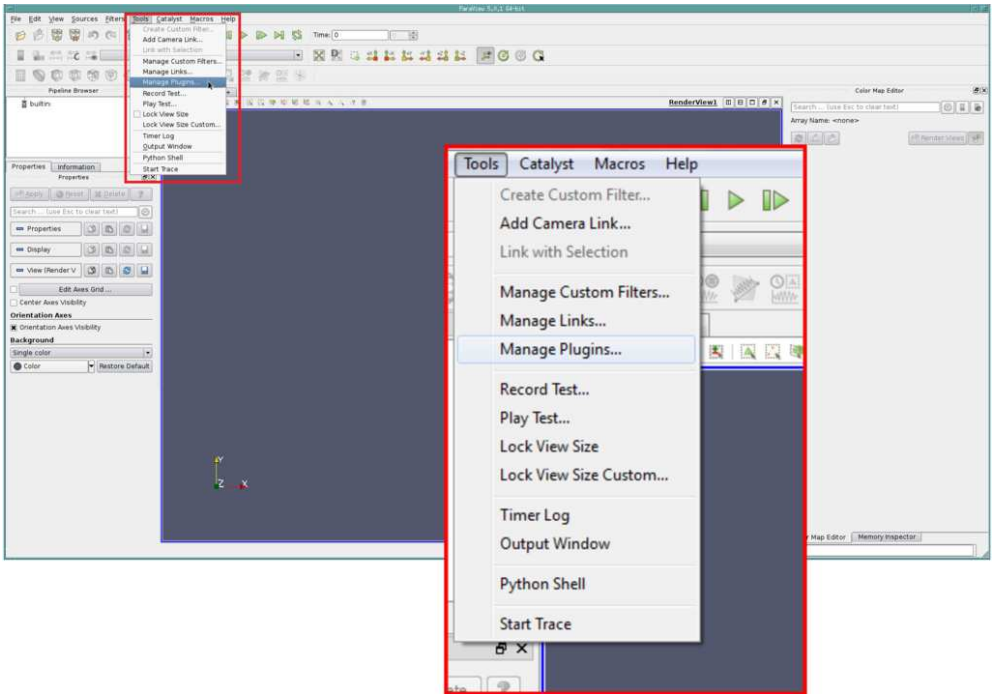

Fig. 3.1 – [Tools ▸ Manage Plugins] from ParaView menu

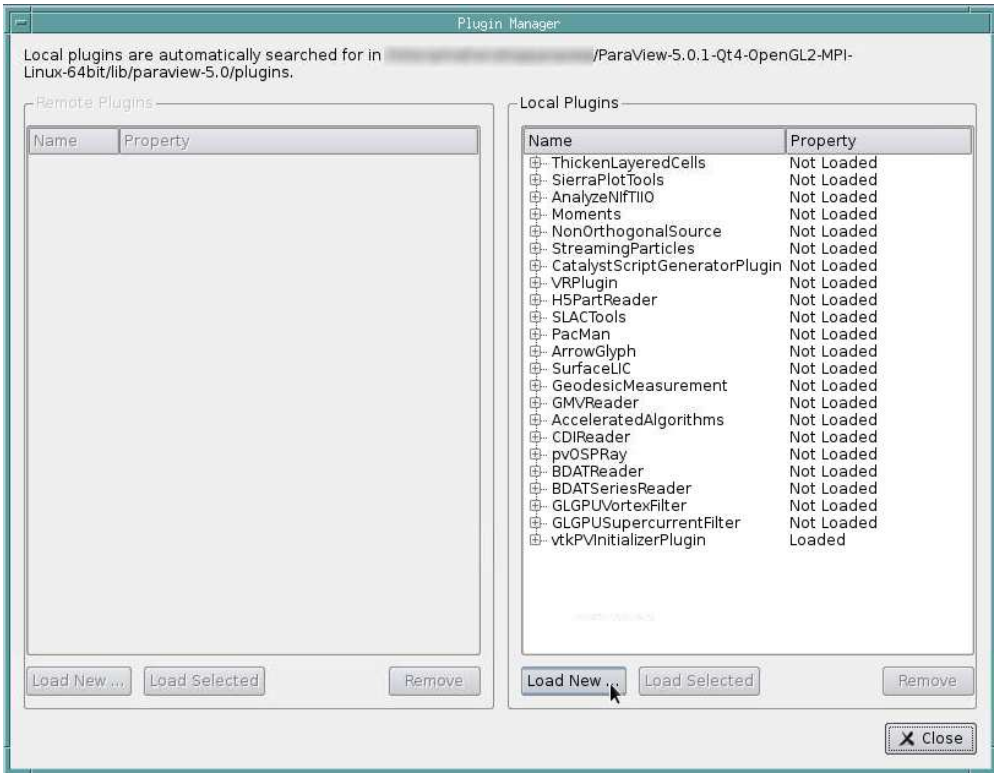

Fig. 3.2 – Click Load New option and locate the plugin

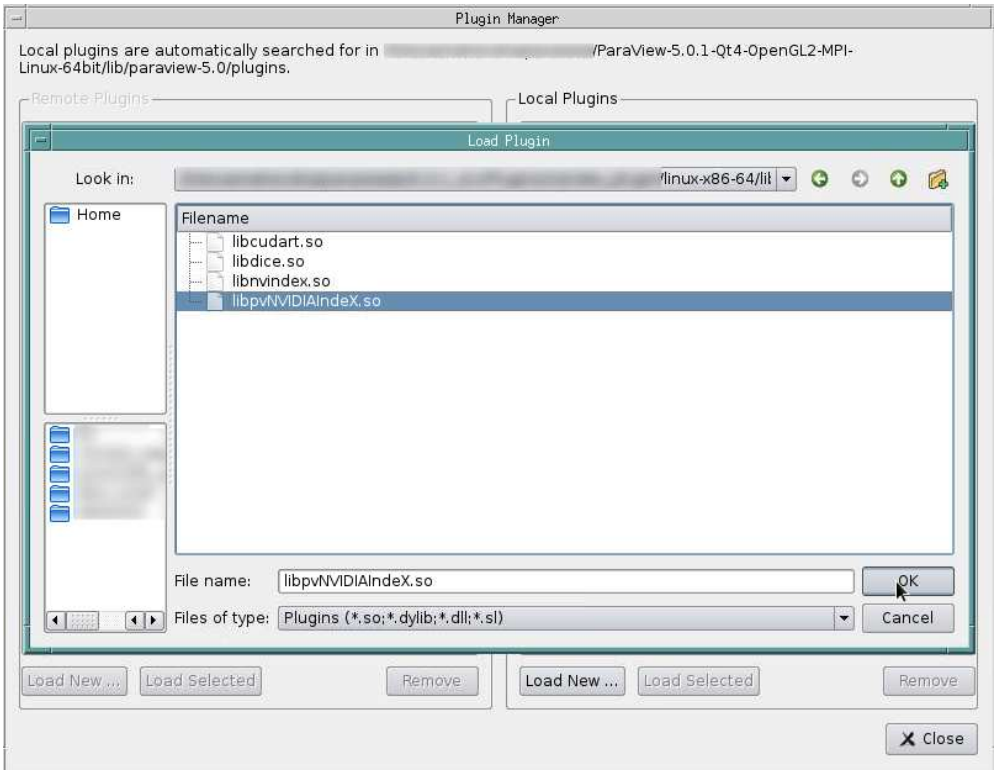

Fig. 3.3 – Load NVIDIA IndeX for ParaView plugin

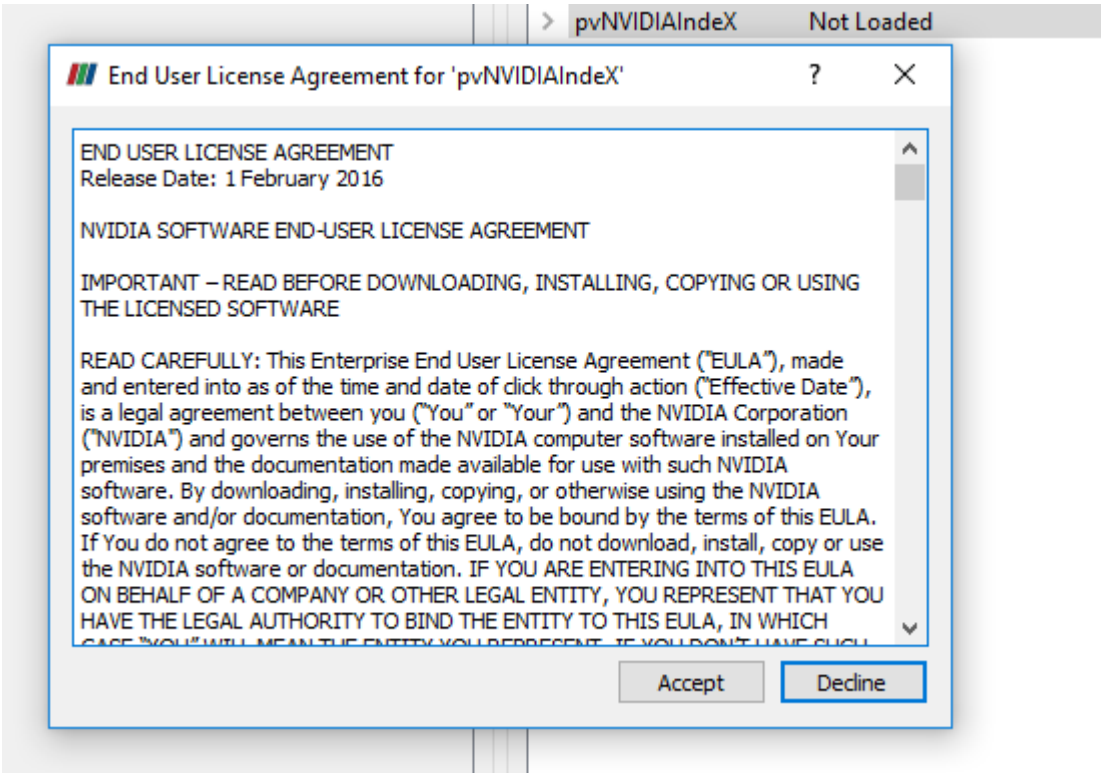

Fig. 3.4 – Read the EULA and click Accept

2. Once the plugin is loaded, plugin name shows up in the [Tools ▸ Manage Plugins] dialog box with status changed as *loaded*. Make sure there no errors in the terminal or on ParaView's console. Also, when using the plugin in client-server mode, be sure to load the plugin in both client and server side of the [Tools ▸ Manage Plugins] window.

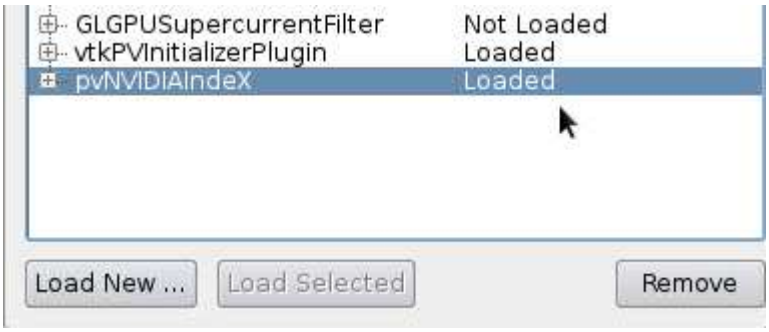

Fig. 3.5 – Status is shown as Loaded when no errors

4 Getting started

This section will describe instructions on how to use the NVIDIA IndeX for ParaView plugin in both client-only and client-server mode.

4.1 Client-only mode

To run the plugin in client-only mode simply launch the ParaView client and load the plugin as described in section 2.

4.2 Client-server mode on a single GPU

To run the plugin in client-server mode, start `pvserver` with `mpirun` as shown below.

```
mpirun -bynode -np 1 pvserver -display :0.0 --use-offscreen-rendering
```

Once `pvserver` process is launched, run the ParaView client and connect to the server where pvservers are running by using [File ▸ Connect ▸ Add Servers] option in ParaView's menubar. Typically the server address is printed out in the console where mpirun was executed, once the client-server is connected, console will update the status with "Client connected" message. Make sure to load the plugin on both client and server side as described in section 2.

To verify that the plugin is installed correctly and loaded successfully in ParaView, please create a *Wavelet* source by clicking the menu option [Sources ▸ Wavelet] in ParaView client.

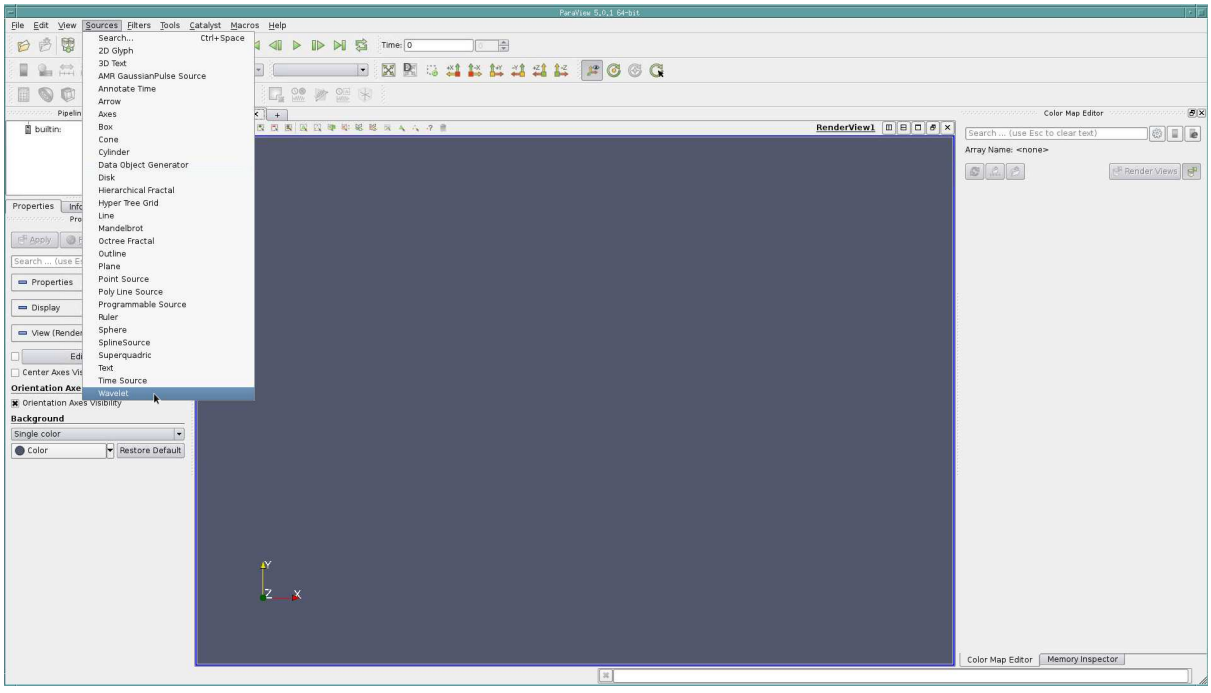

Fig. 4.1 – Create a Wavelet source

Once you click apply an *Outline* representation will be shown in the viewport. Select *RTData* as the scalar array from the dropdown box instead of *Solid Color* and *NVIDIA IndeX* instead of *Outline* as the representation as shown below.

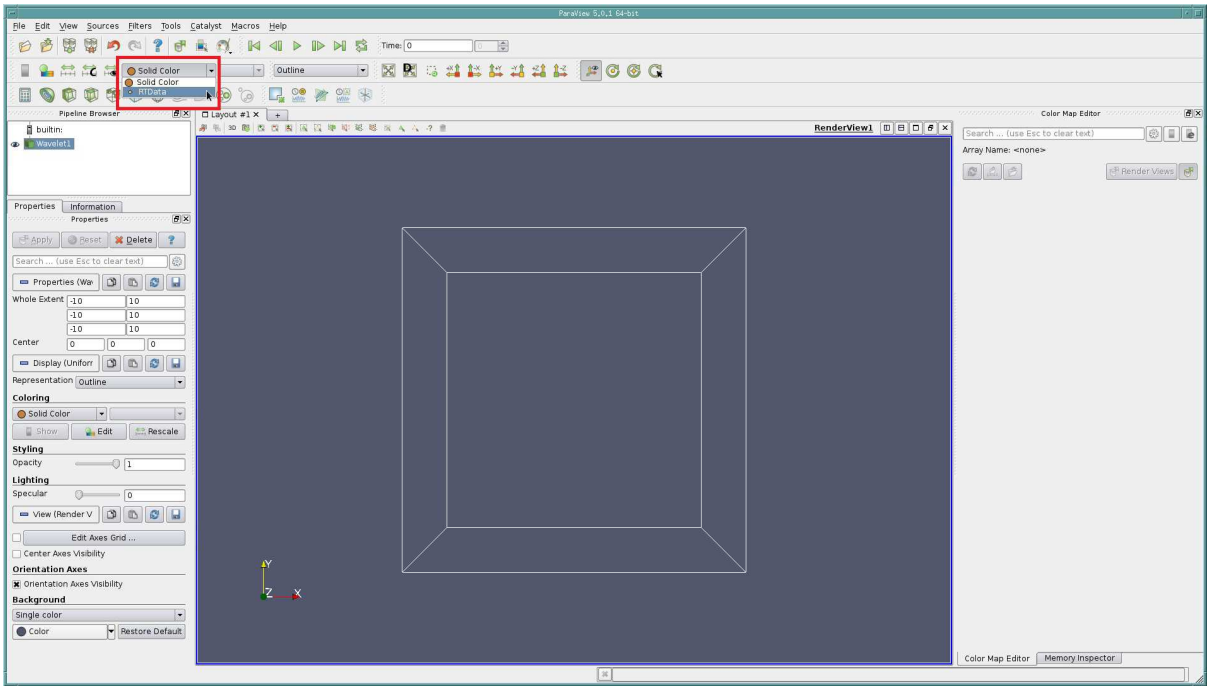

Fig. 4.2 – Outline is default representation

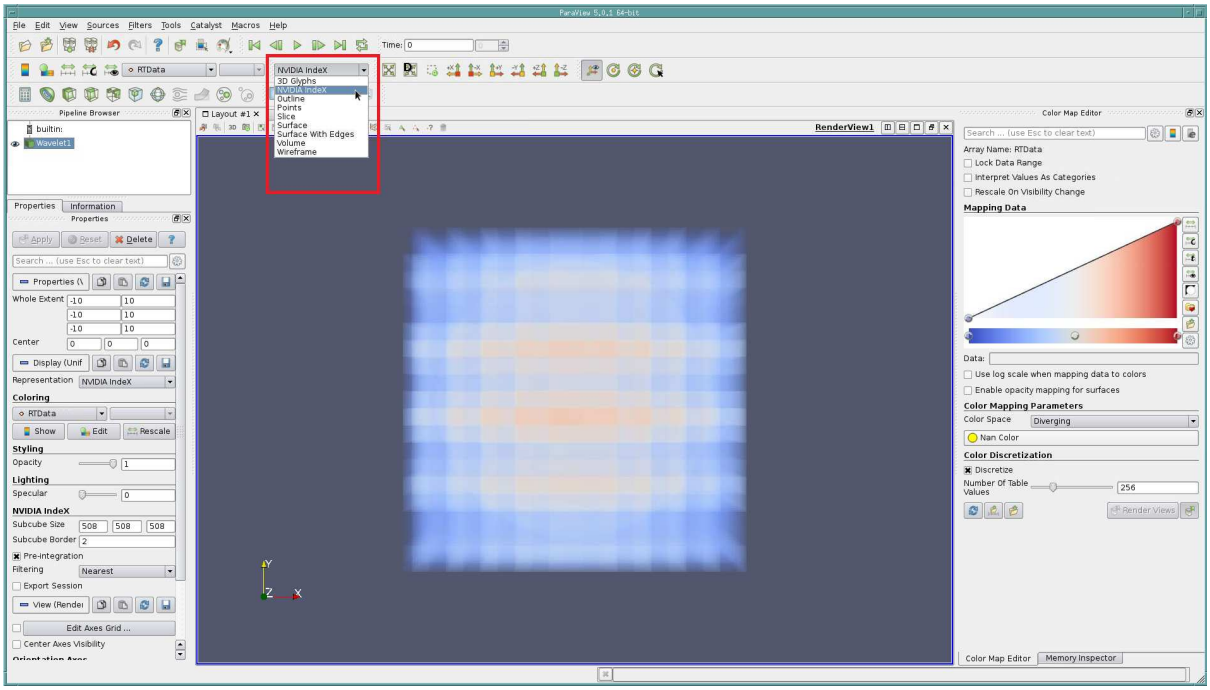

Fig. 4.3 – Wavelet source rendered by NVIDIA IndeX

4.2.1 Client-server mode on multiple GPUs

If you have acquired a valid license for the cluster version of the plugin, you can run the plugin in client-server mode on multiple GPUs. When using multiple GPUs on the same machine it is required to start one MPI process per GPU. For example, on a machine with 4 GPU's:

```
mpirun -bynode -np 1 pvserver -display :0.0 --force-offscreen-rendering \
 : -np 1 pvserver -display :0.1 --force-offscreen-rendering \
 : -np 1 pvserver -display :0.2 --force-offscreen-rendering \
```

```
  : -np 1 pvserver -display :0.3 --force-offscreen-rendering \
```

When using multiple GPUs on multiple machines in a cluster environment MPI's host-file functionality can be used. For example, to utilize two machines with two GPUs each:

```
mpirun --hostfile myhosts --bynode -np 2 pvserver -display :0.0 --force-↦
    offscreen-rendering \
  : -np 2 pvserver -display :0.1 --force-offscreen-rendering
```

Where *myhosts* is a text file with list of host names where MPI will spawn pvserver instances.

Before connecting pvservers with ParaView client, please disable IceT compositing. This can be done from ParaView client's settings menu [Edit ▸ Settings ▸ Render view] and restart ParaView client.

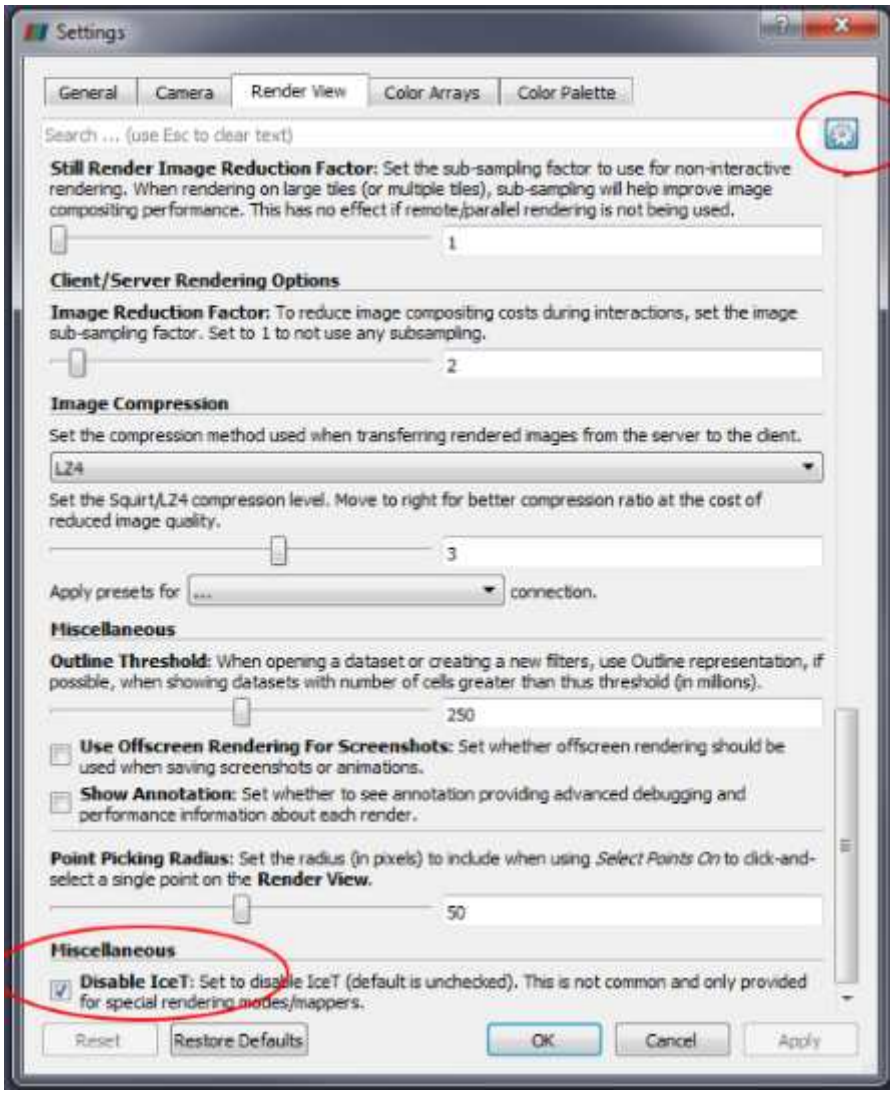

Fig. 4.4 – Disable IceT from Settings menu

Refer to this page[3] for more details about pvserver and running ParaView in client-server mode.

[3] `http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server`

4.2.2 Networking parameters for NVIDIA IndeX

NVIDIA IndeX is a distributed renderer capable of utilizing multiple GPU's on a cluster, to configure networking features of NVIDIA IndeX an optional configuration file `nvindex_config.xml` is provided with the plugin. Copy this file under the default configuration directory of ParaView:

```
cp plugin-directory/nvindex_config.xml ~/.config/ParaView/
```

If the directory `/.config/ParaView/` is not accessible, you can set the following environment variable pointing to the `nvindex_config.xml` file:

```
export NVINDEX_PVPLUGIN_HOME=path-to-directory-with-config-file
```

Each networking option is enclosed within a pair of XML tags and the file is enclosed within `<index_config>` ... `</index_config>` tags.

4.2.2.1 Licensing

When you obtain a license for the cluster version of the plugin, you will receive a file named `license.lic`. It contains the keys `NVINDEX_VENDOR_KEY` and `NVINDEX_SECRET_KEY` that can be set as enviroment variables on all the hosts where NVIDIA IndeX is run.

```
export NVINDEX_VENDOR_KEY=vendor-key-here
export NVINDEX_SECRET_KEY=secret-key-here
```

Alternatively, copy paste those keys in the `<license>` section of the config file as show below.

```
<license>
   <vendor_key>vendor-key-here</vendor_key>
   <secret_key>secret-key-here</secret_key>
</license>
```

4.2.2.2 Networking options

All the networking configuration options for the plugin are specified under the `<network>` section of the config file.

Tag `<cluster_mode>` defines the networking mode of NVIDIA IndeX with modes `UDP` and `TCP` supported, with `UDP` being the preferred mode.

```
<cluster_mode>
   your-networking-mode
</cluster_mode>
```

Tag `<cluster_interface_address>` defines the Network Interface Card(NIC) that is used for communication between the nodes. On Linux, the `ifconfig` command gives the NIC address as `inet addr`. If not set, any address is valid. The string may end with a colon character (:

) and a port number to select which port to listen to for UDP and TCP. If no port is set and unicast only mode is set, port 10000 will be used.

```
<cluster_interface_address>
   172.161.123.0/24:10001
</cluster_interface_address>
```

Tag `multicast_address` defines the multicast address for the nodes to communicate. This is valid only when `cluster_mode` is set to UDP.

```
<multicast_address>
   224.1.3.2
</multicast_address>
```

Tag `discovery_address` defines the discovery address used for TCP `cluster_mode`.

```
<discovery_address>
   224.1.3.3:5555
</discovery_address>
```

Tag `<use_rdma>` can be used to switch `yes` or `no` to use RDMA mode for networking.

```
<use_rdma>
   no
</use_rdma>
```


5 Features

This section will provide a walk-through on individual plugin features.

5.1 Structured and Unstructred grids

NVIDIA IndeX for ParaView plugin enables volume rendering of both structured and unstructured grid types.

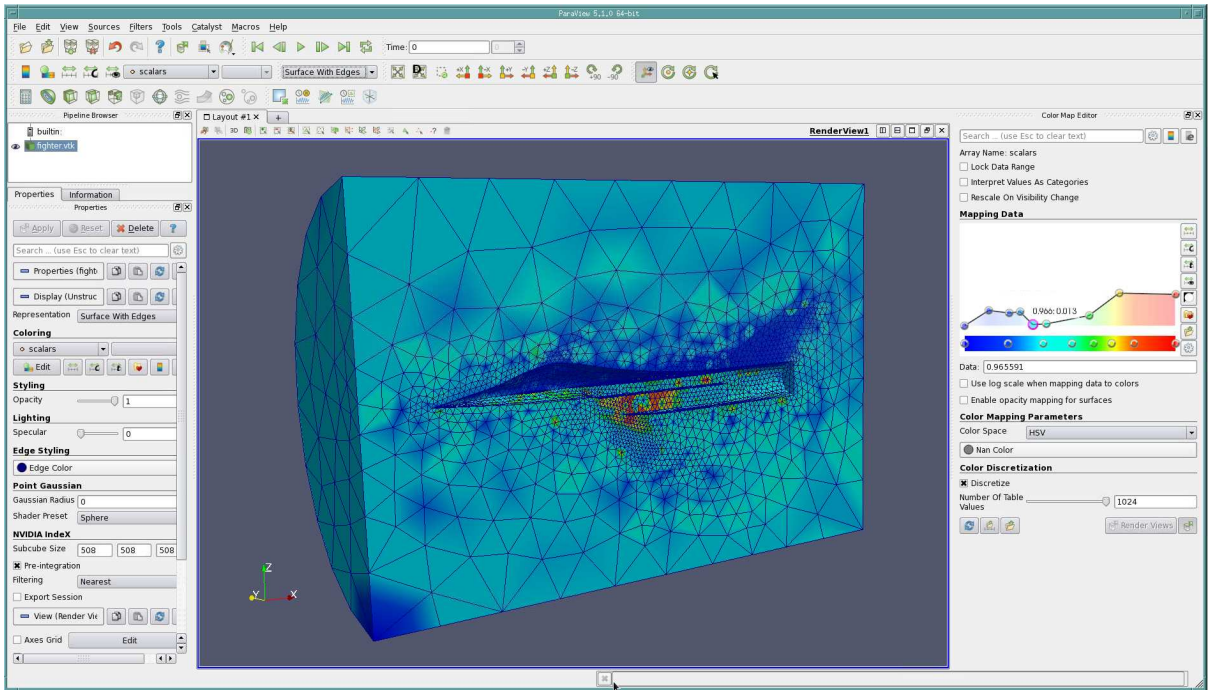

Fig. 5.1 – Grid rendered as a surface

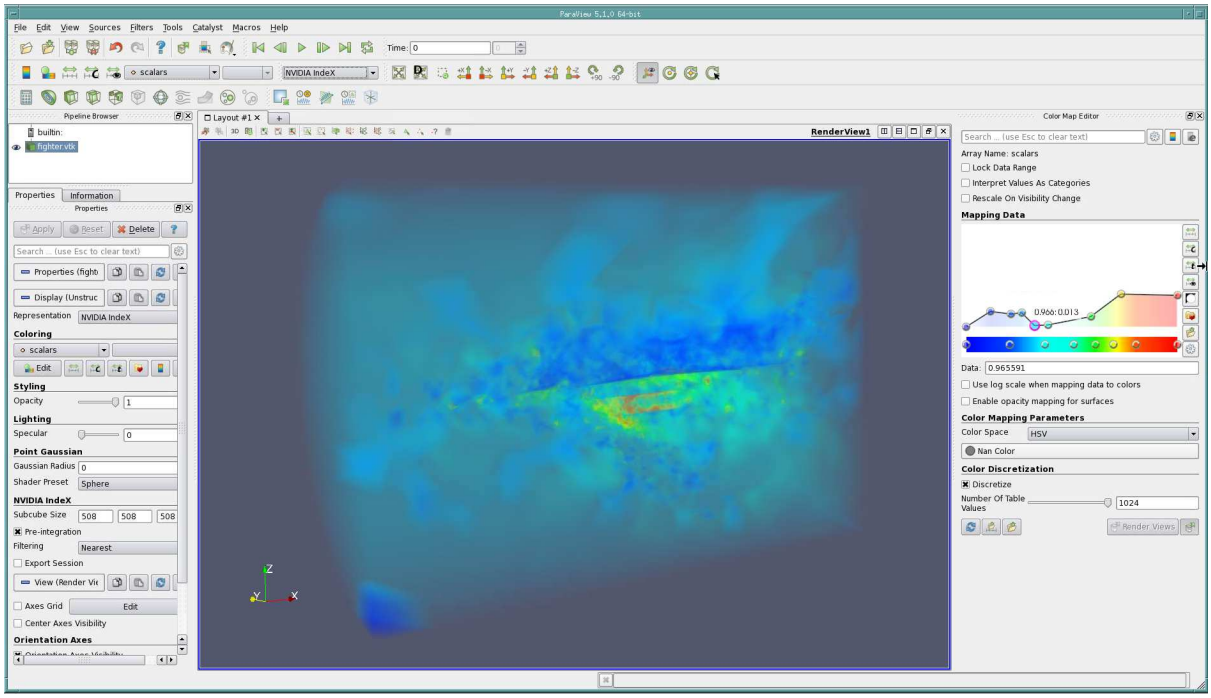

Fig. 5.2 – Grid rendered as a volume using NVIDIA IndeX

5.2 Datatypes

NVIDIA IndeX supports different datatype formats such as *unsigned char, unsigned short and floating point*. Make sure appropriate byte endianness is chosen when loading unsigned short and floating point datatypes otherwise your visualization might look like artifacts or even look completely random.

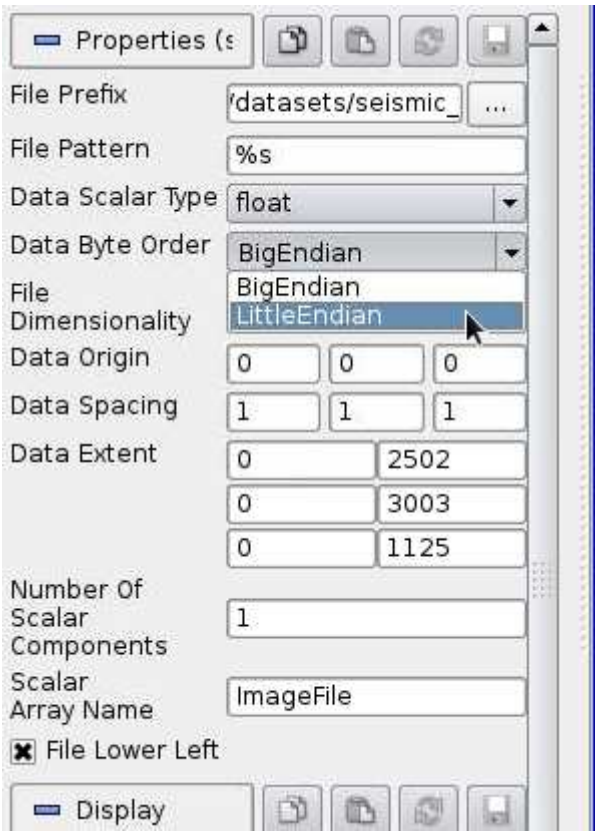

Fig. 5.3 – Choose endianness of the dataset

5.3 Transfer function and colormap changes

Transfer function changes can be done using ParaView's colormap editor. If the colormap editor is not open you can do so by using the menu option.

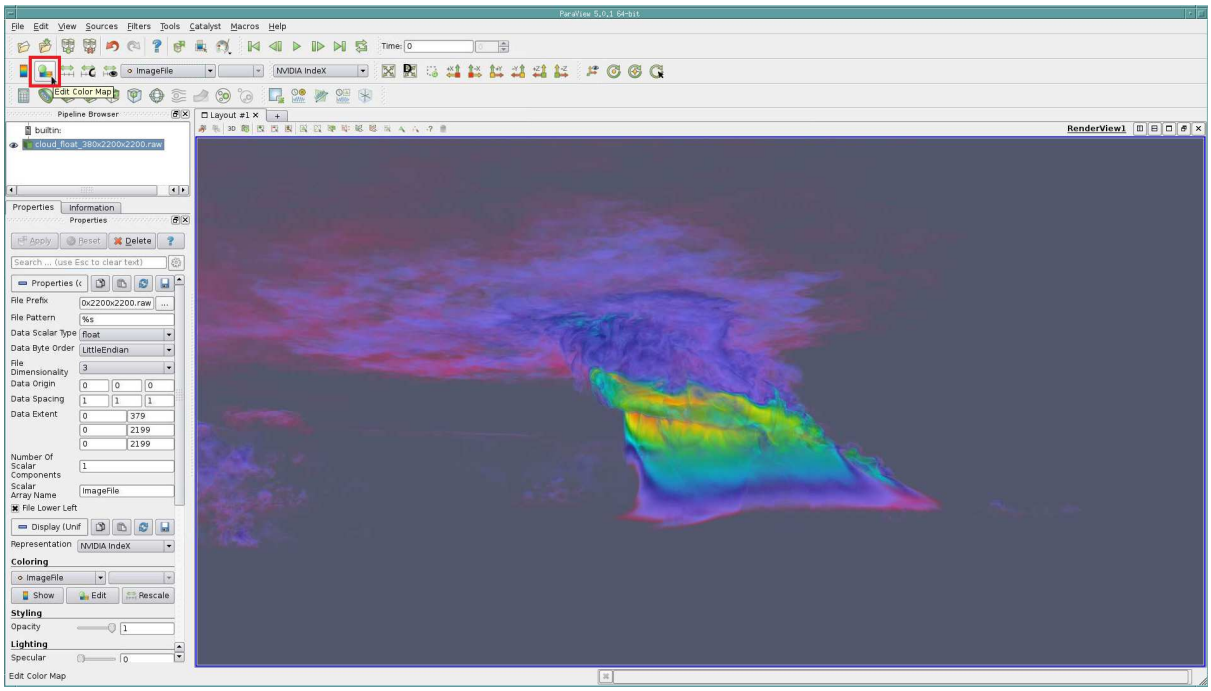

Fig. 5.4 – Click on the icon to open up Colormap Editor in ParaView

Using the colormap editor user interface you can visualize parts of the dataset that is interesting for you. This can be achieved by changing the colortable, by adjusting the transparency, or by setting custom domain range values to isolate parts of the dataset that is uninteresting.

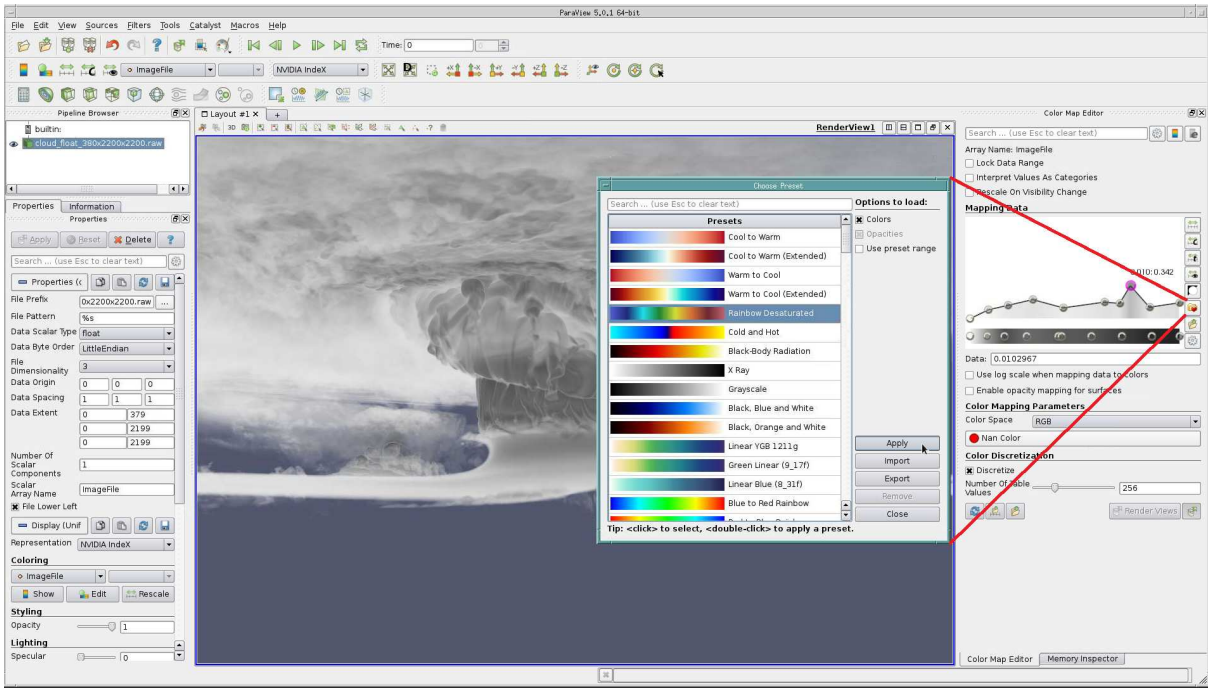

Fig. 5.5 – Choose a suitable colortable and click Apply

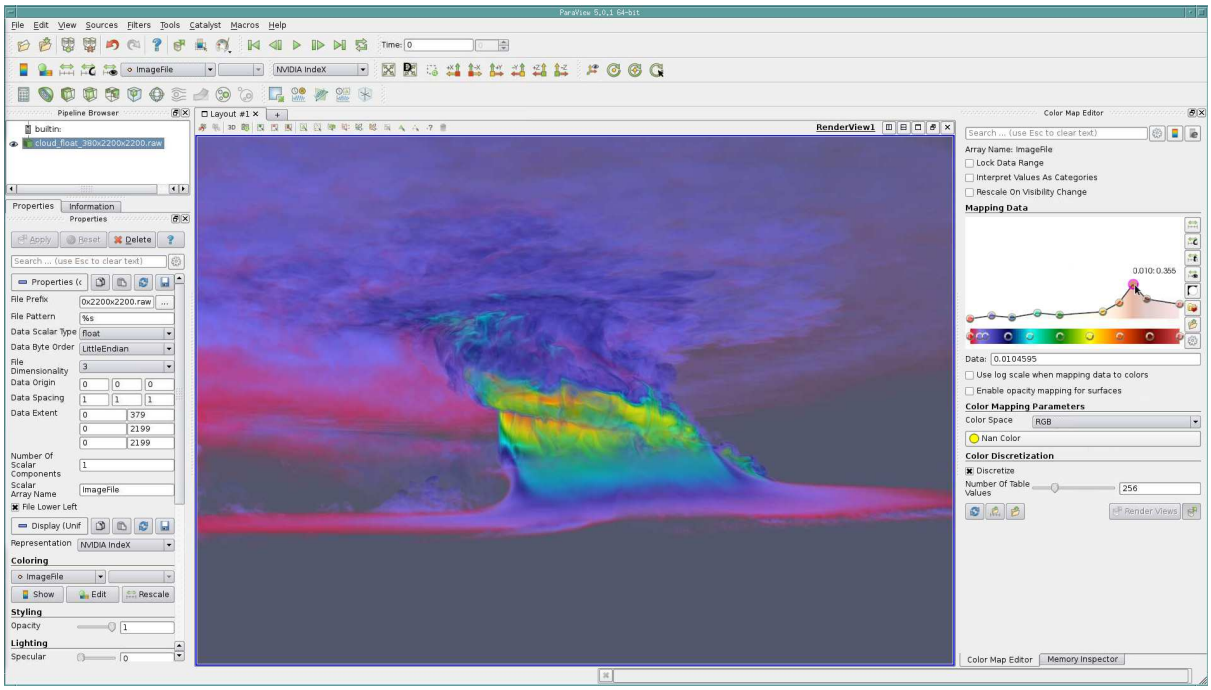

Fig. 5.6 – Colortable matching your dataset domain

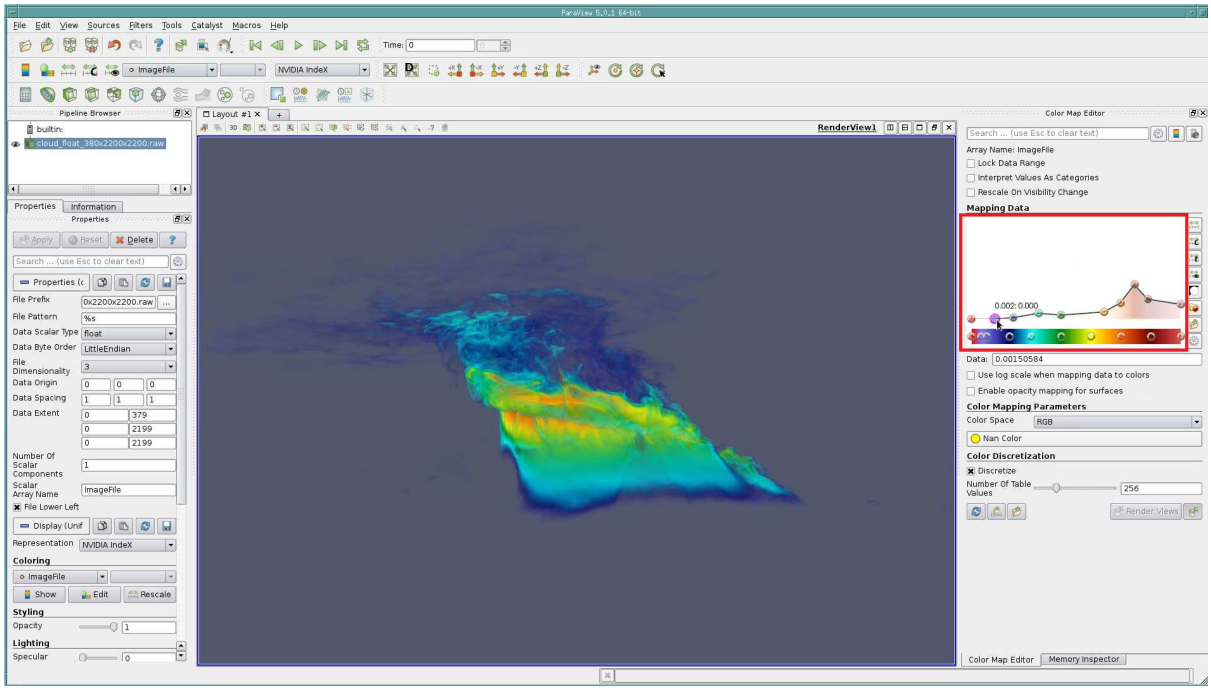

Fig. 5.7 – Changing the opacity values

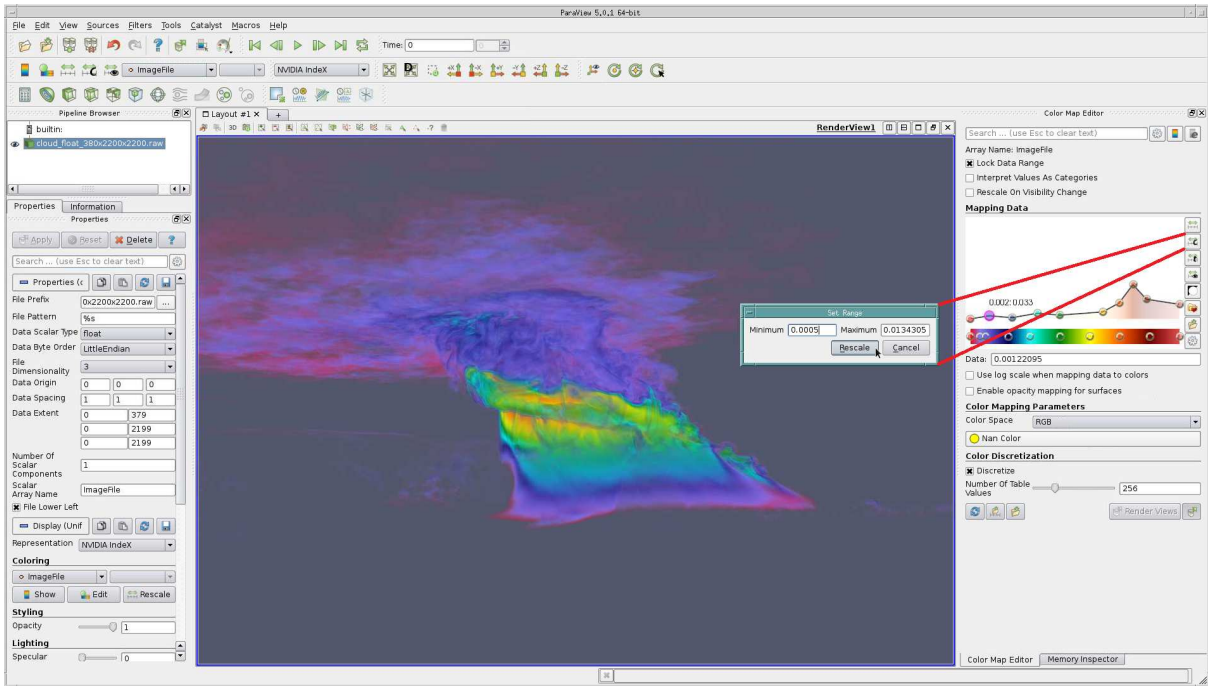

Fig. 5.8 – Custom data domain range

5.4 Region of interest changes

Users can select a custom region of interest to visualize specific sections of the dataset using the sliders in the properties panel. This is tagged as an experimental feature since changing region of interest is restricted to axis aligned directions.

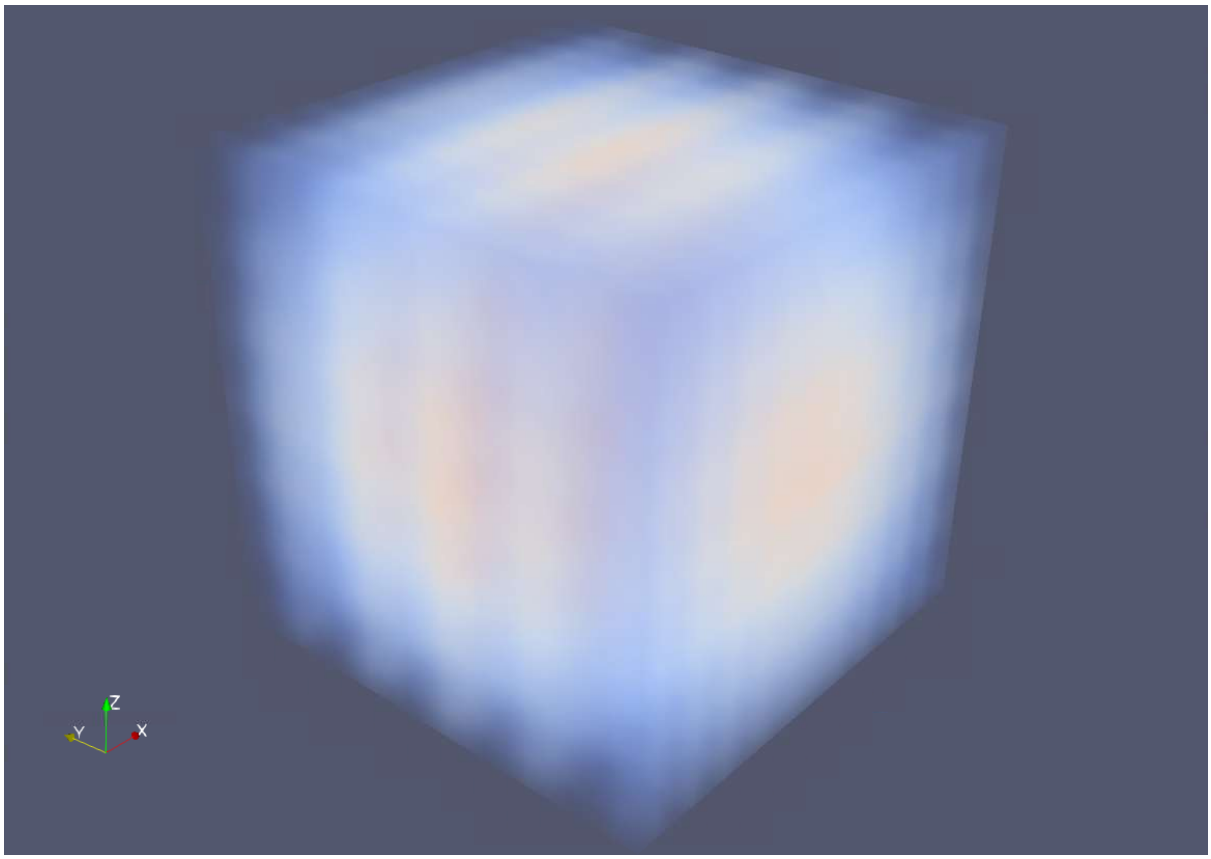

Fig. 5.9 – Wavelet volume

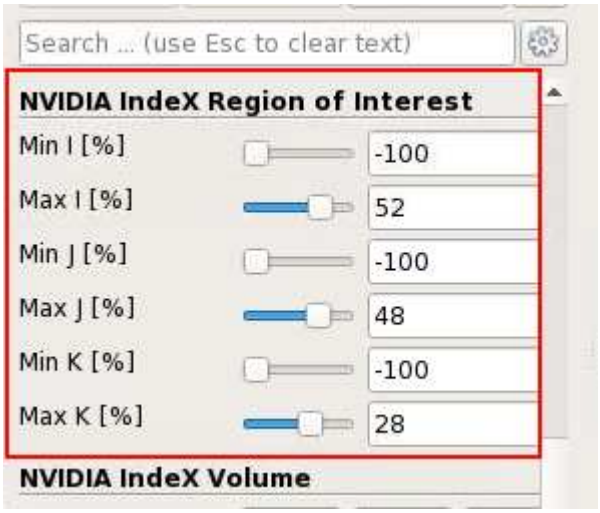

Fig. 5.10 – Changing region of interest

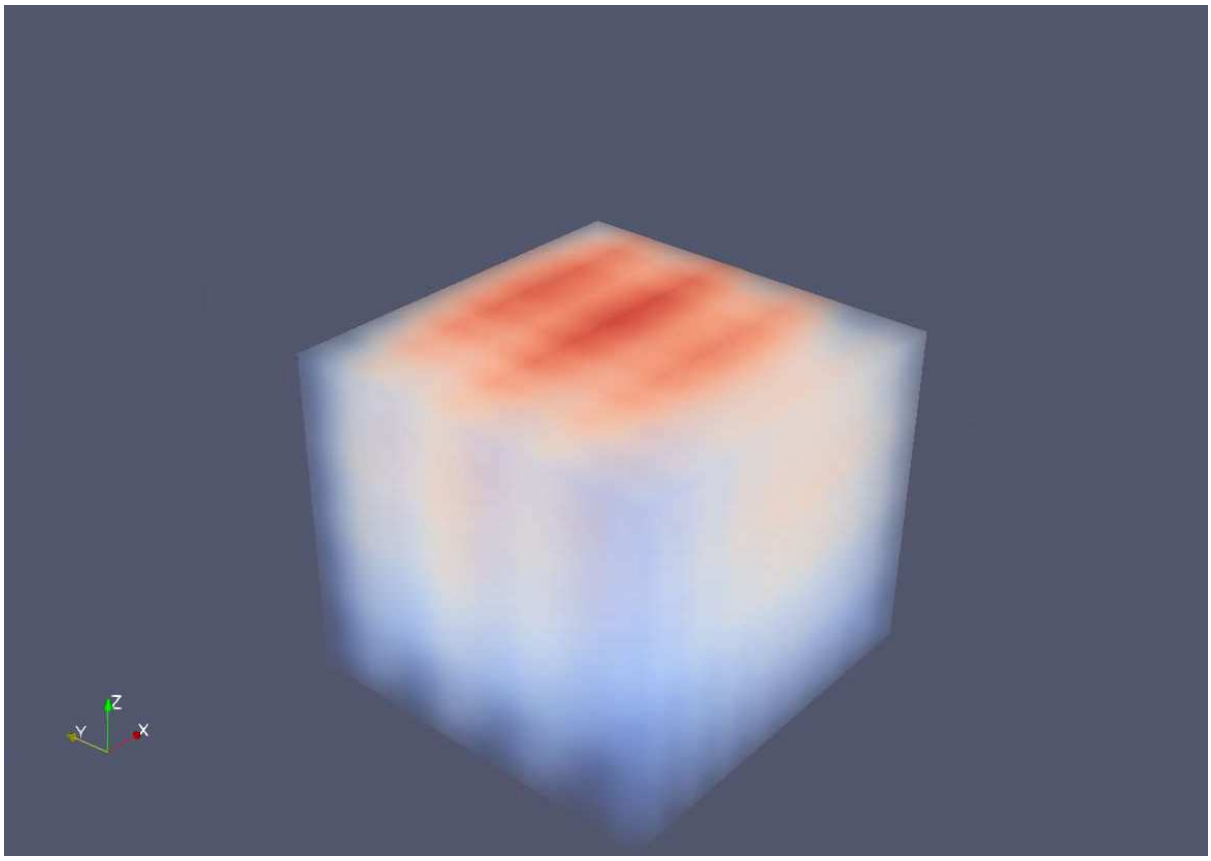

Fig. 5.11 – Wavelet volume with a custom region of interest

5.5 High-quality

High quality rendering can be achieved by using pre-integration and filtering techniques of NVIDIA IndeX exposed in the plugin, there is a known performance-quality trade off when using some of these filtering techniques.

These filtering options can be found in ParaView's **Properties** panel typically on the left hand side of the ParaView client user interface when the plugin is loaded.

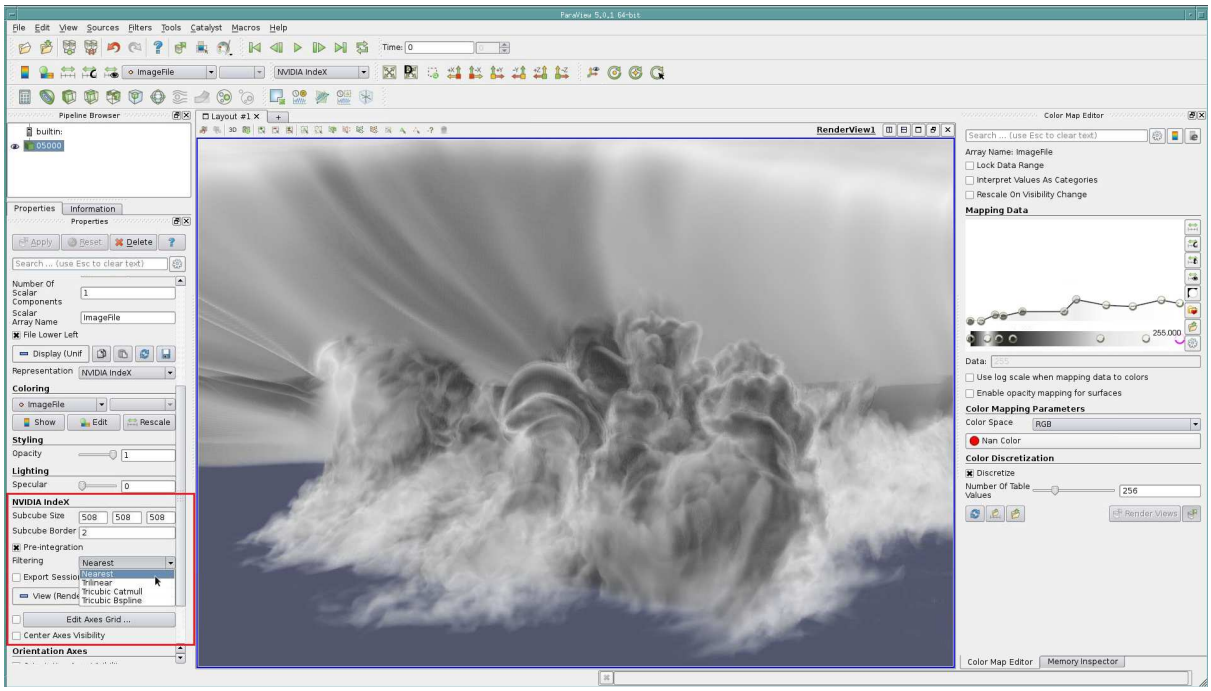

Fig. 5.12 – Properties panel in ParaView

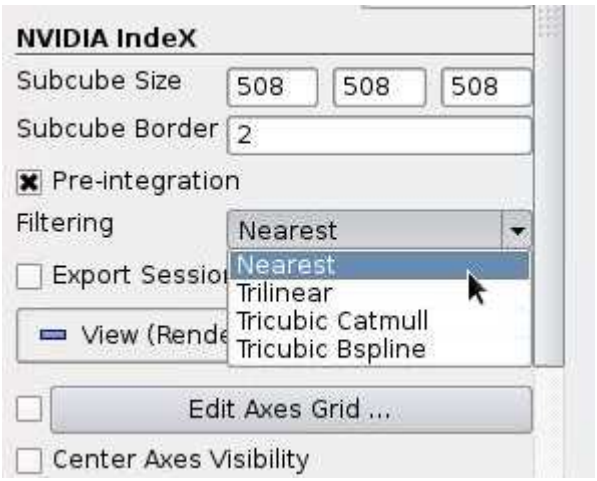

Fig. 5.13 – Filtering options in NVIDIA IndeX properties panel

There is no one filtering option optimal for all the datasets, each filtering option achieves different levels of quality with different datasets and transfer function combinations with nearest neighbor interpolation being the most basic one. Some example images comparing different filtering options are shown below.

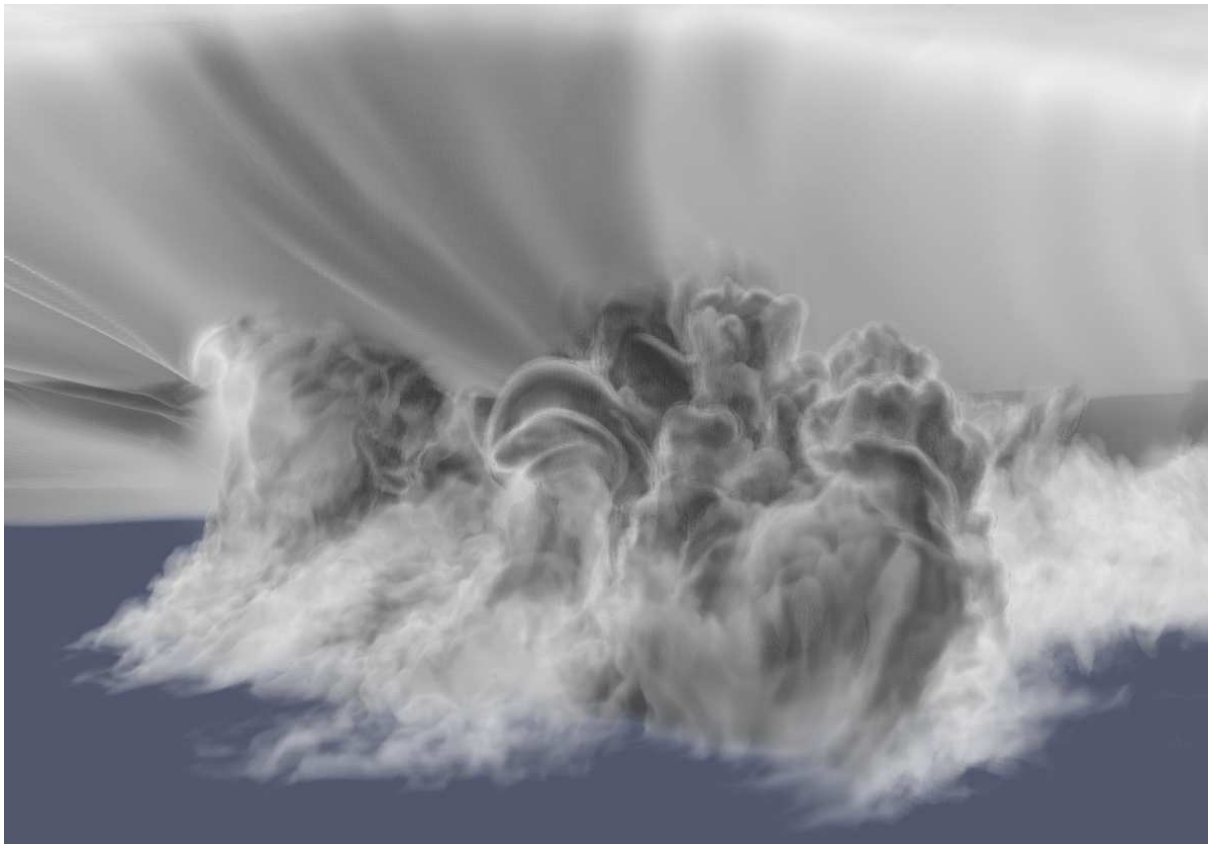

Fig. 5.14 – Base of an EF5 tornado cloud

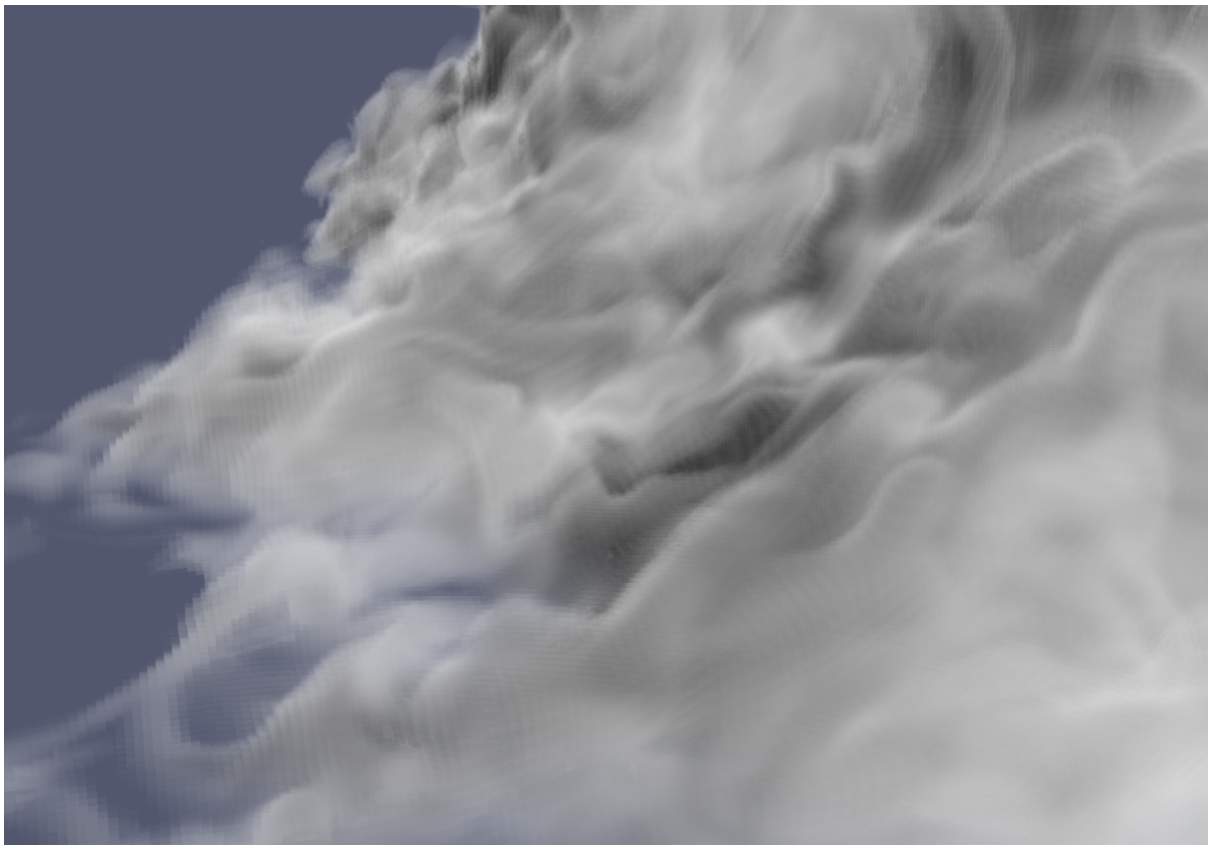

Fig. 5.15 – Nearest neighbor interpolation

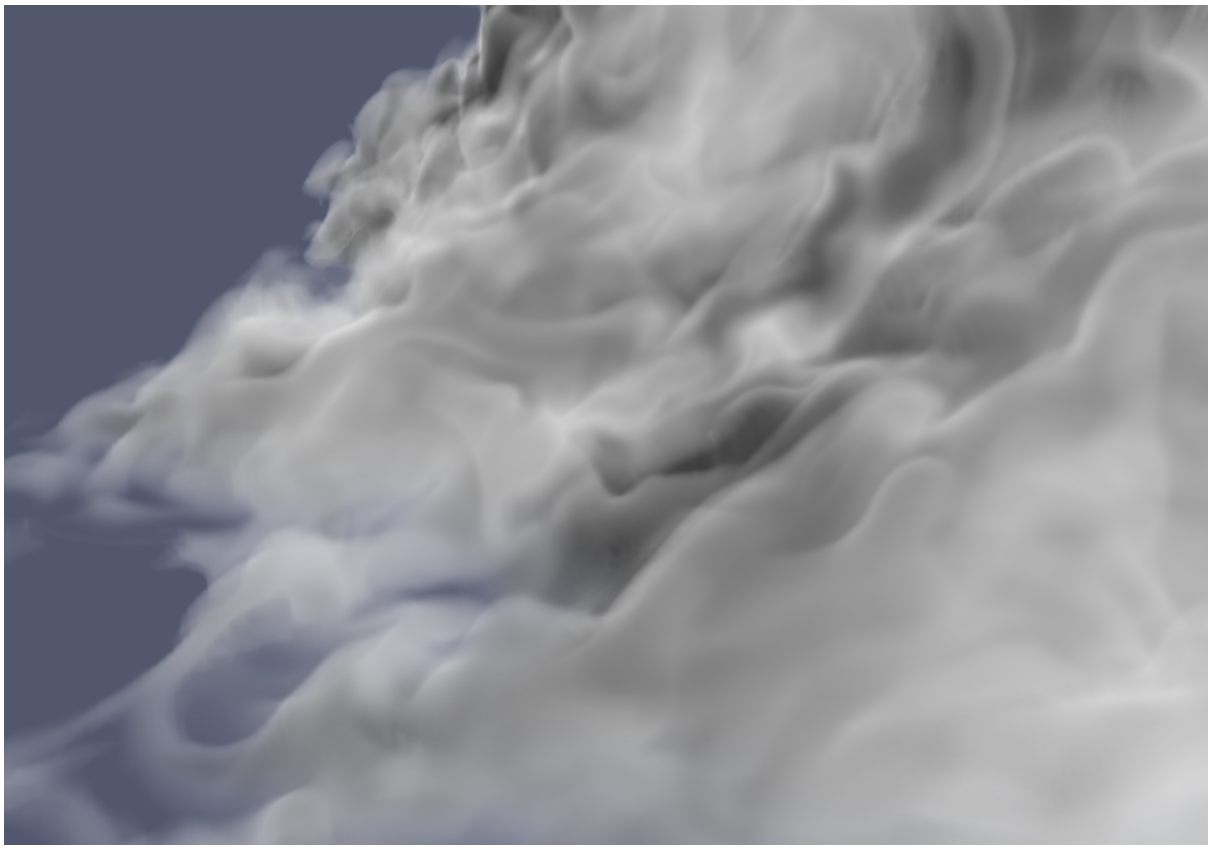

Fig. 5.16 – Trilinear interpolation

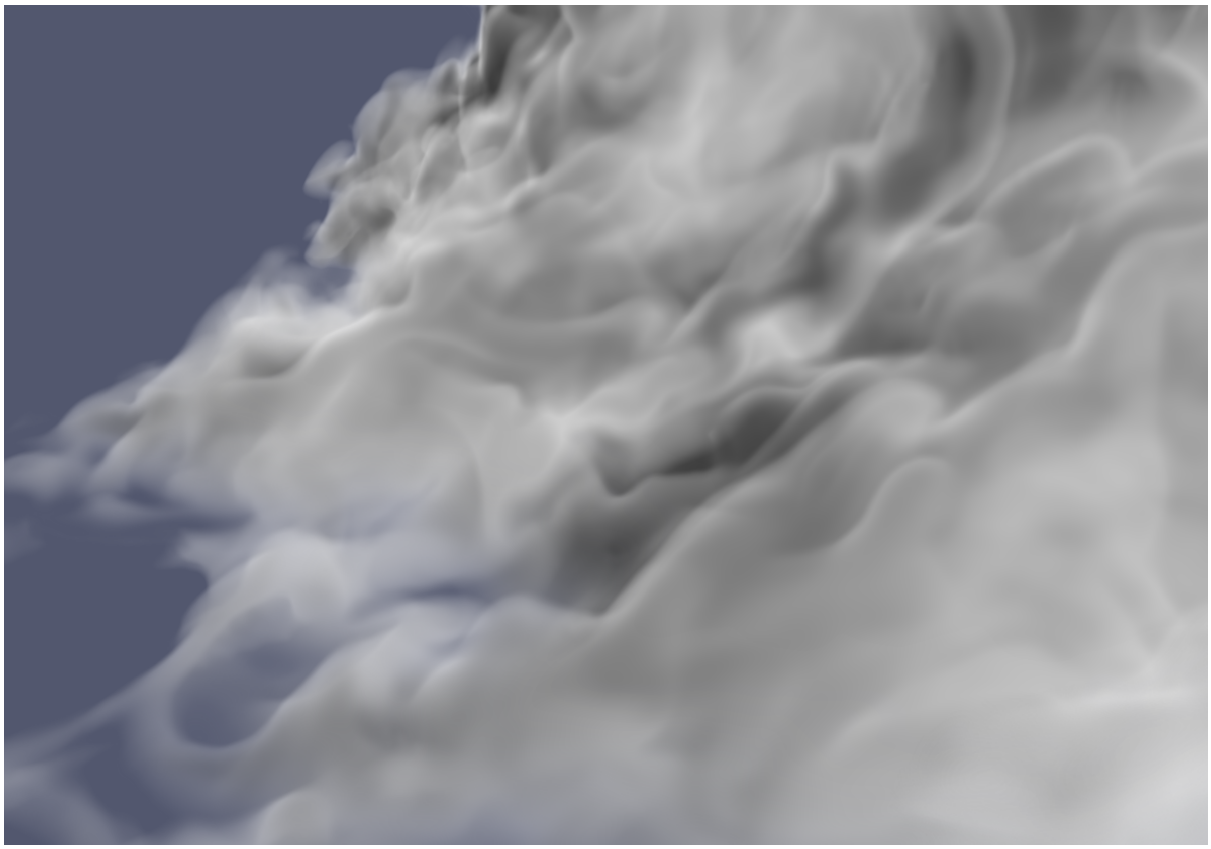

Fig. 5.17 – Tricubic Bspline interpolation

5.6 Slice rendering

Volume slices can be enabled from the GUI. Currently it is limited to a single axis aligned slice, native support with ParaView slices is coming soon.

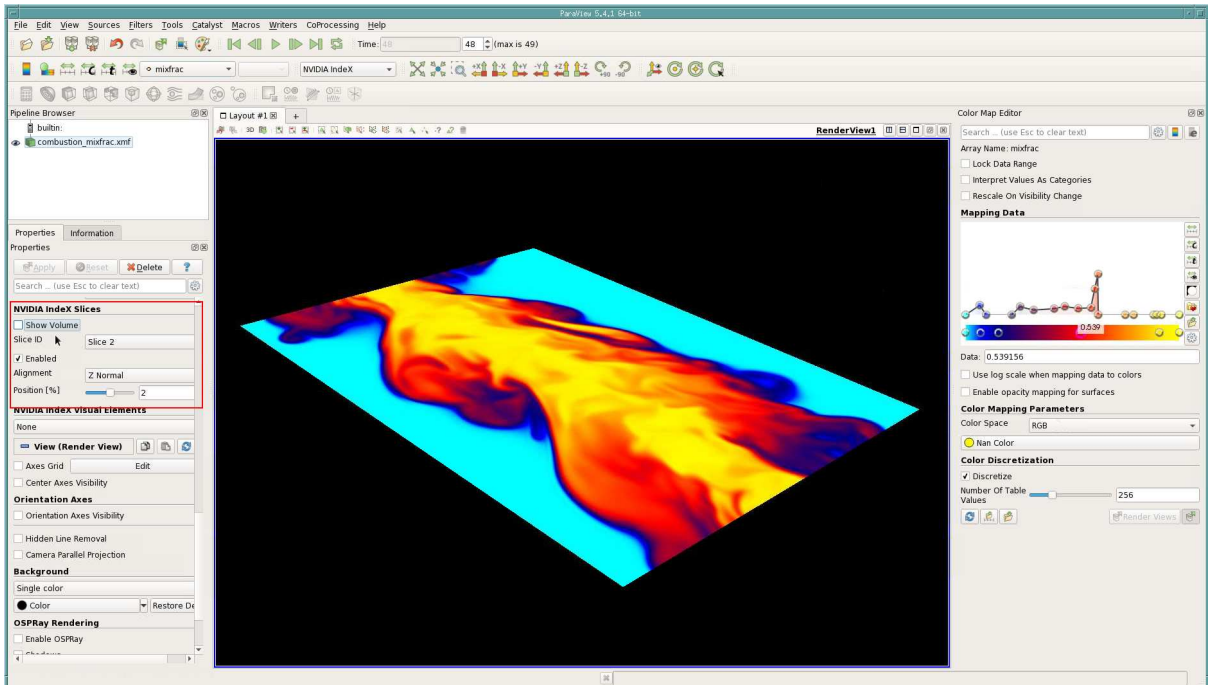

Fig. 5.18 – Properties panel in ParaView

5.7 Slice rendering

Volume slices can be enabled from the GUI. Currently it is limited to three axis aligned slices with ability to move individual slice positions, native support with ParaView slices is work in progress. This is tagged as an experimental feature since the slice rendering is not part of the ParaView slice rendering mechanism in the user interface.

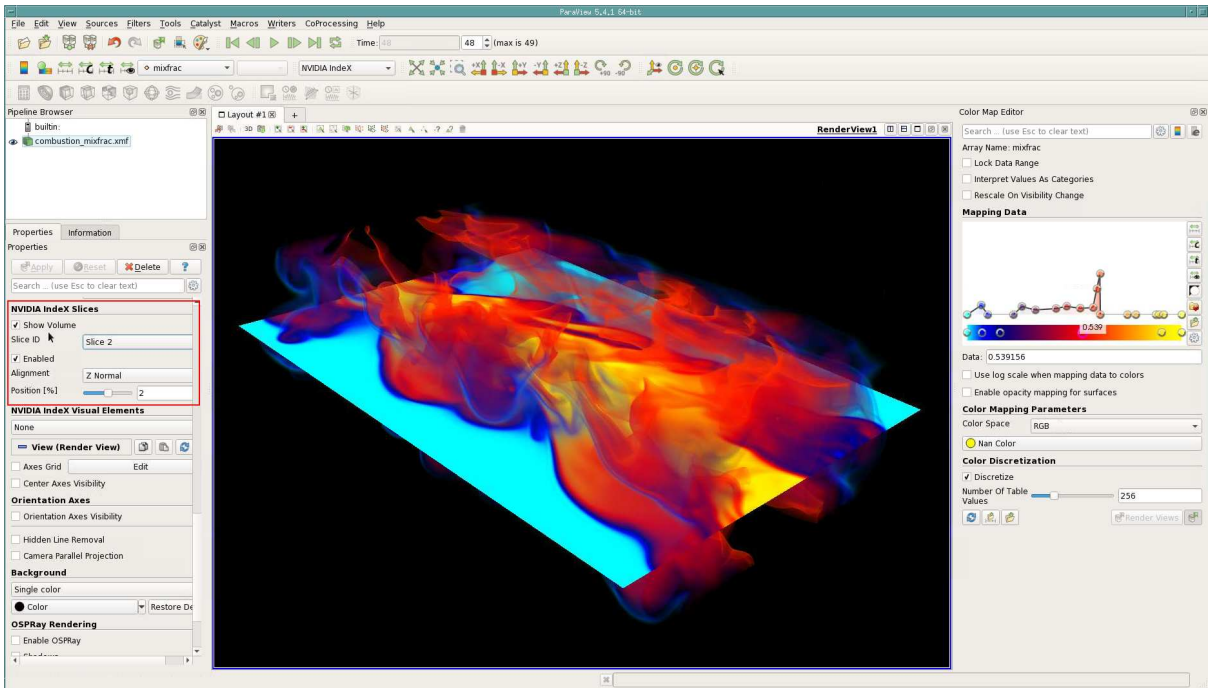

Fig. 5.19 – Slice rendering options through properties panel. Dataset is made available by Dr. Jackqueline Chen at Sandia Laboratories through US Department of Energy's SciDAC Institute for Ultrascale Visuaization.

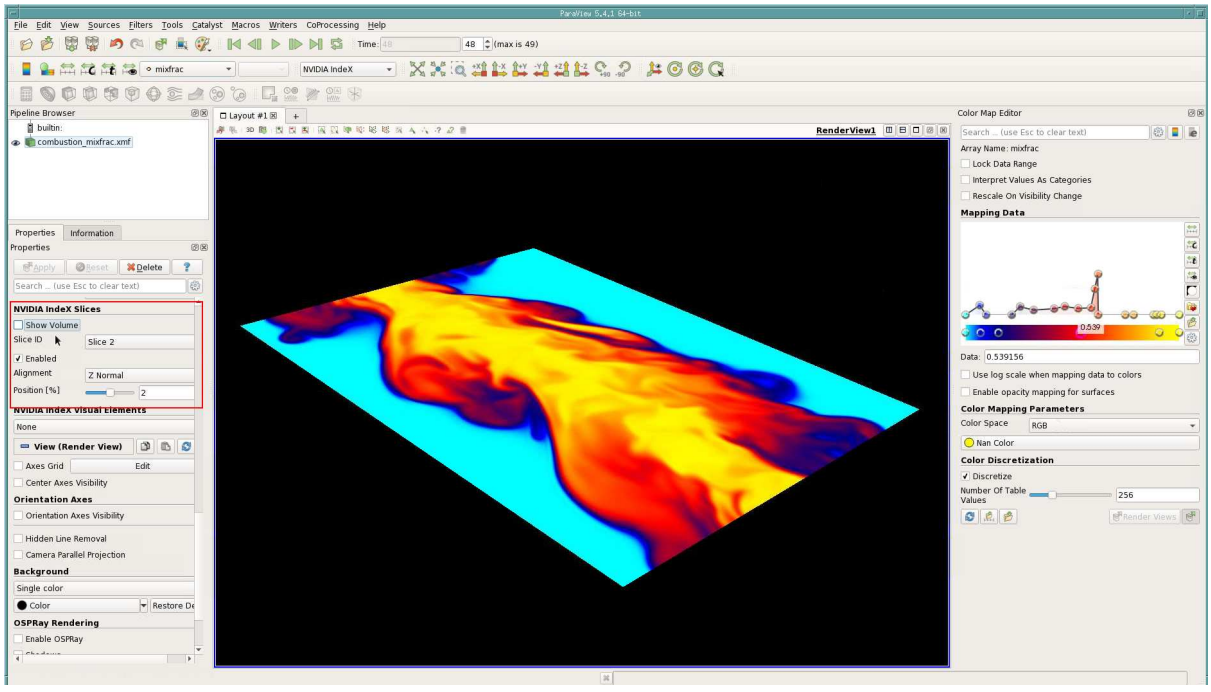

Fig. 5.20 – Slice rendering with volume disabled

5.8 NVIDIA IndeX visual elements

Visual elements feature of NVIDIA IndeX library enables you to enhance the visual cues in the dataset. In this version of the plugin there are three visual presets available, each preset has one ore more parameters for finer control over that visual element.

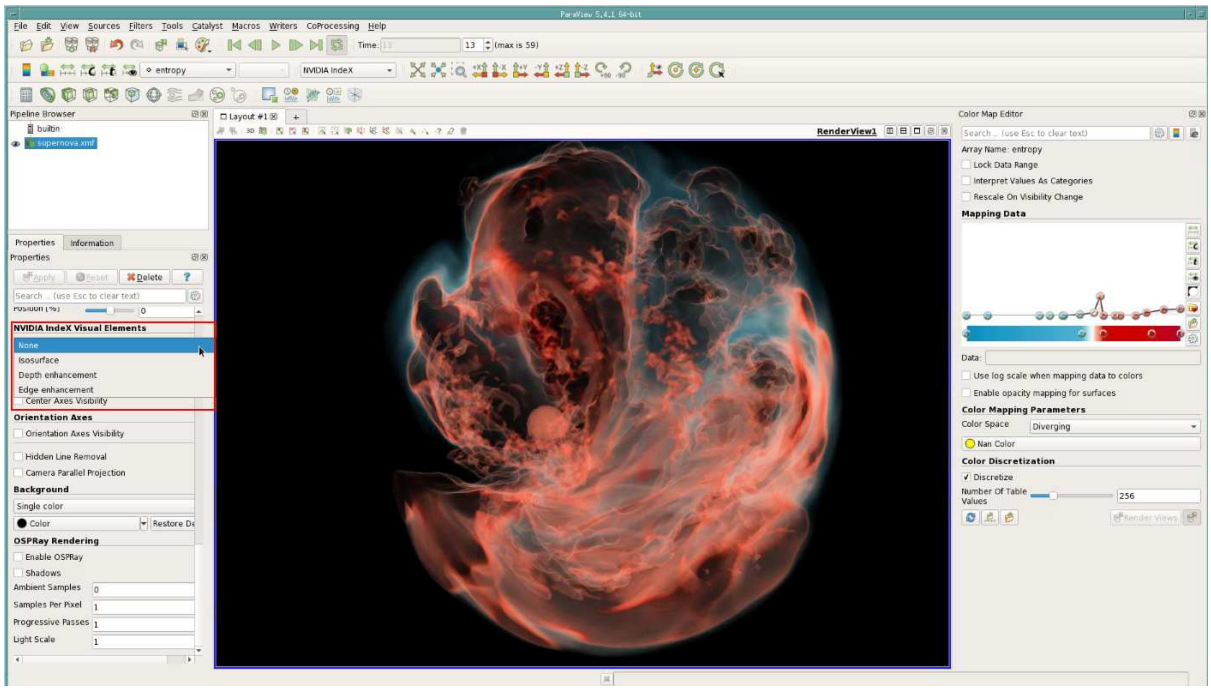

Fig. 5.21 – Supernova SASI visualized as a volume. Dataset is made available by Dr. John Blondin at the North Carolina State University through US Department of Energy's SciDAC Institute for Ultrascale Visuaization.

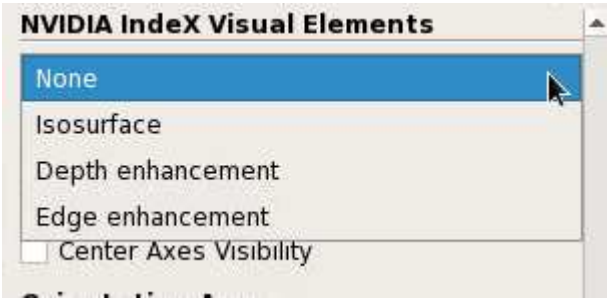

Fig. 5.22 – Visual element presets in the properties panel

5.8.1 Iso-surface preset

Iso-surface preset allows you to extract an iso-surface based on the min/max voxel range with different ways to map color values. Fill mode controls how the volume is rendered between the iso-min and iso-max values.

No Fill
: Volume samples are set to transparent.

Single Color
: Volume samples are set to Iso Min value.

Colormap
: Volume samples are taken from colormap.

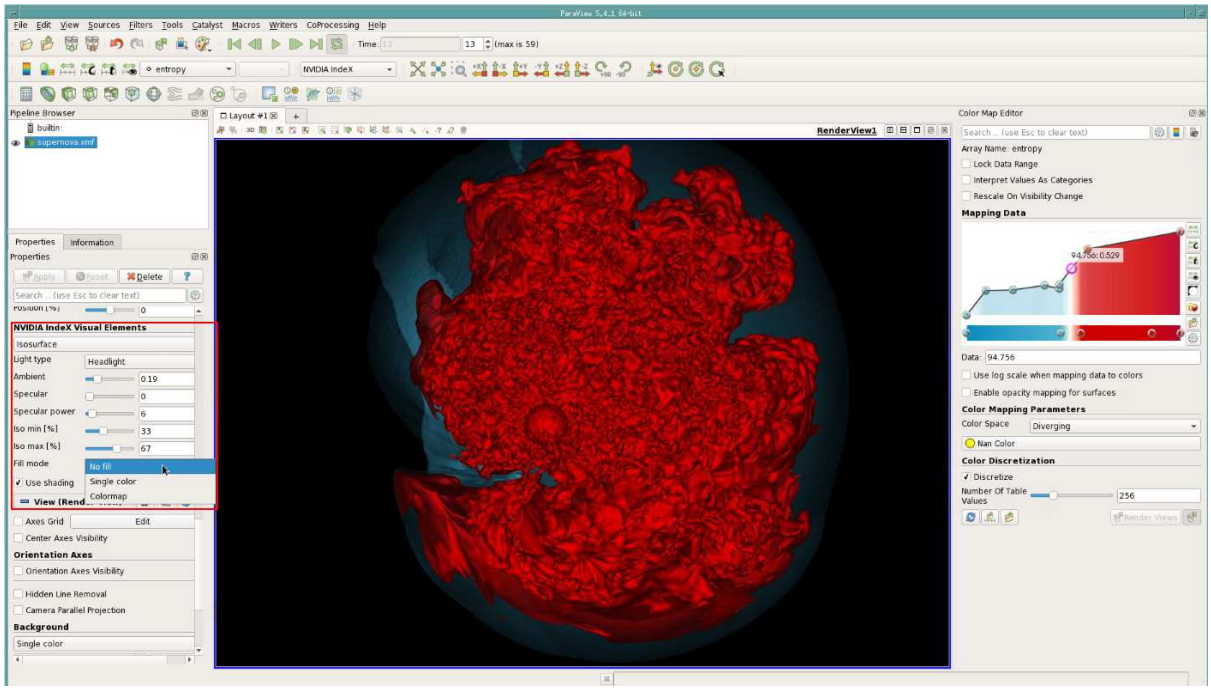

Fig. 5.23 – Supernova SASI visualized as an iso-surface

5.8.2 Depth enhancement preset

Depth enhancement preset allows you to enhance depth perception in the dataset by isolating features with high opacity values in the transfer function mapping. At the current volume position it accumulates colormap alpha values along a predefined ray segment and darken samples in regions with low alpha values.

Samples
: Number of samples taken along the predefined ray segment.

Gama
: Used to increase contrast.

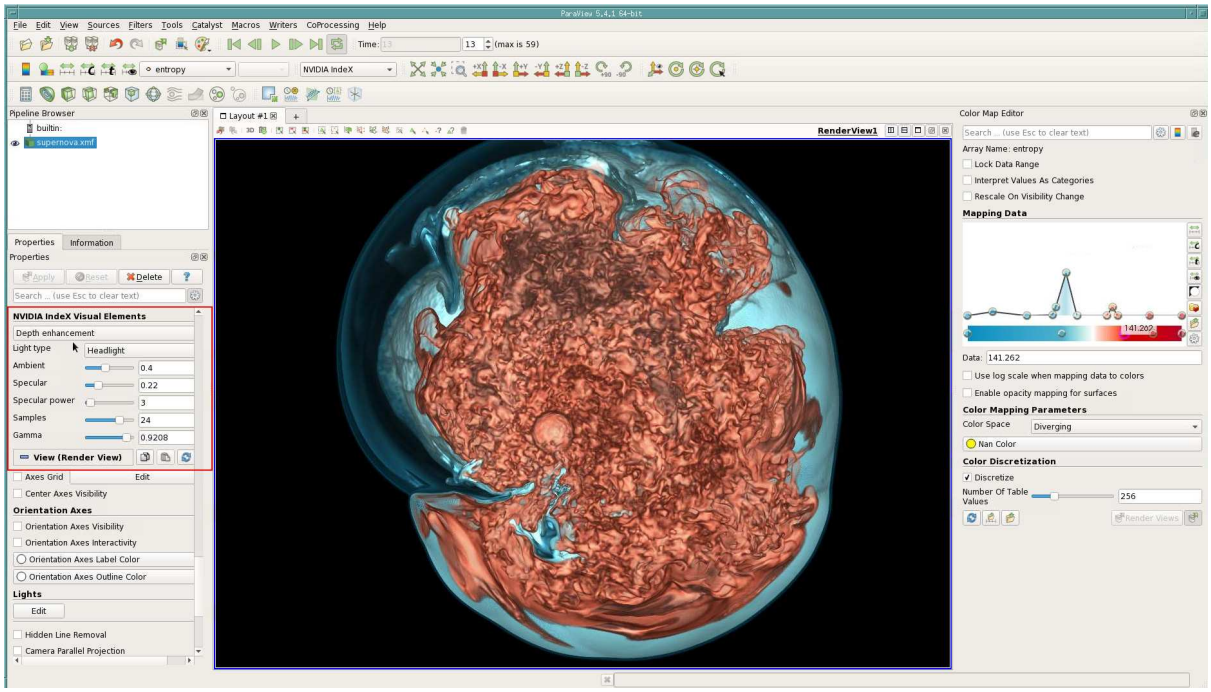

Fig. 5.24 – Supernova SASI visualized using depth enhancement preset

5.8.3 Edge enhancement preset

Edge enhancement preset allows you to enhance the edges or the “silhouettes” based on the transfer function setting. At the current volume position it calculates the colormap alpha gradient along a ray segment and darken samples that contain higher gradient magnitude.

Edge Range

The length of the ray segment where the gradient is calculated.

Samples

Number of samples used to calculate the gradient along the ray segment.

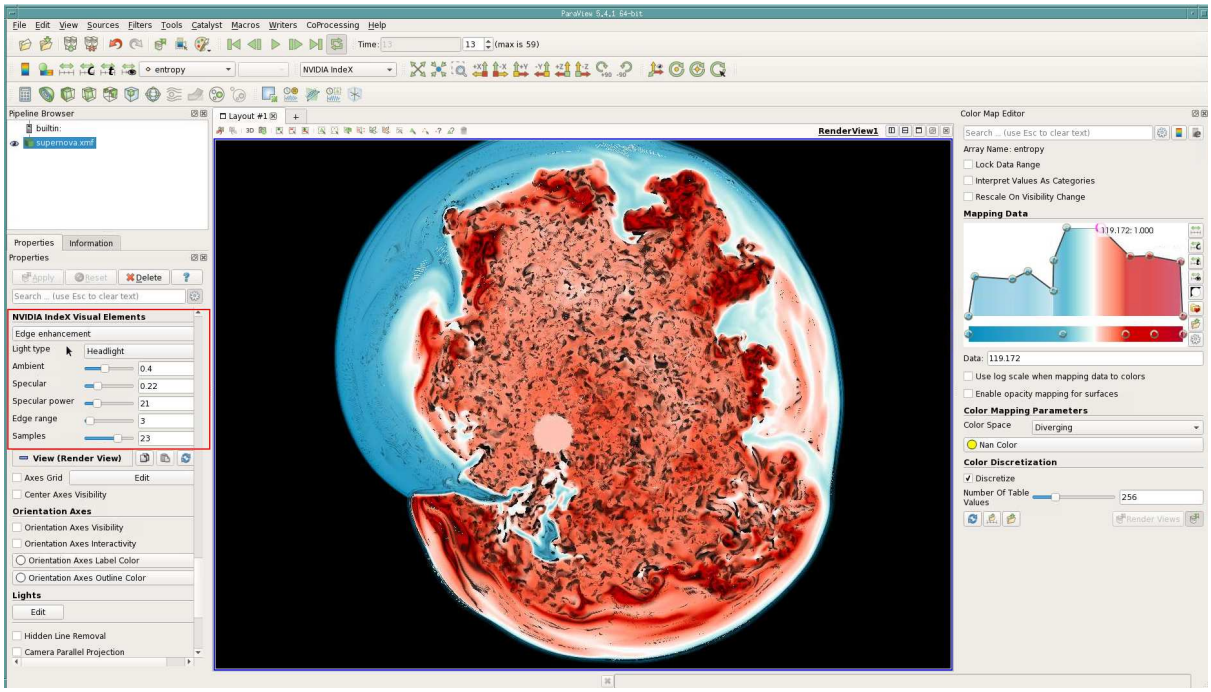

Fig. 5.25 – Supernova SASI visualized using edge enhancement preset

5.9 Time series animation

Time series animation feature allows you to render timesteps of a dataset in real-time. You can navigate, change colormaps and perform data operations as you would do with a static dataset. In order to have a smooth playback, please set the following setting from ParaView menu allowing you to cache geometry. Animation will be slower in the first cycle but once all the timesteps are loaded the playback should be smooth.

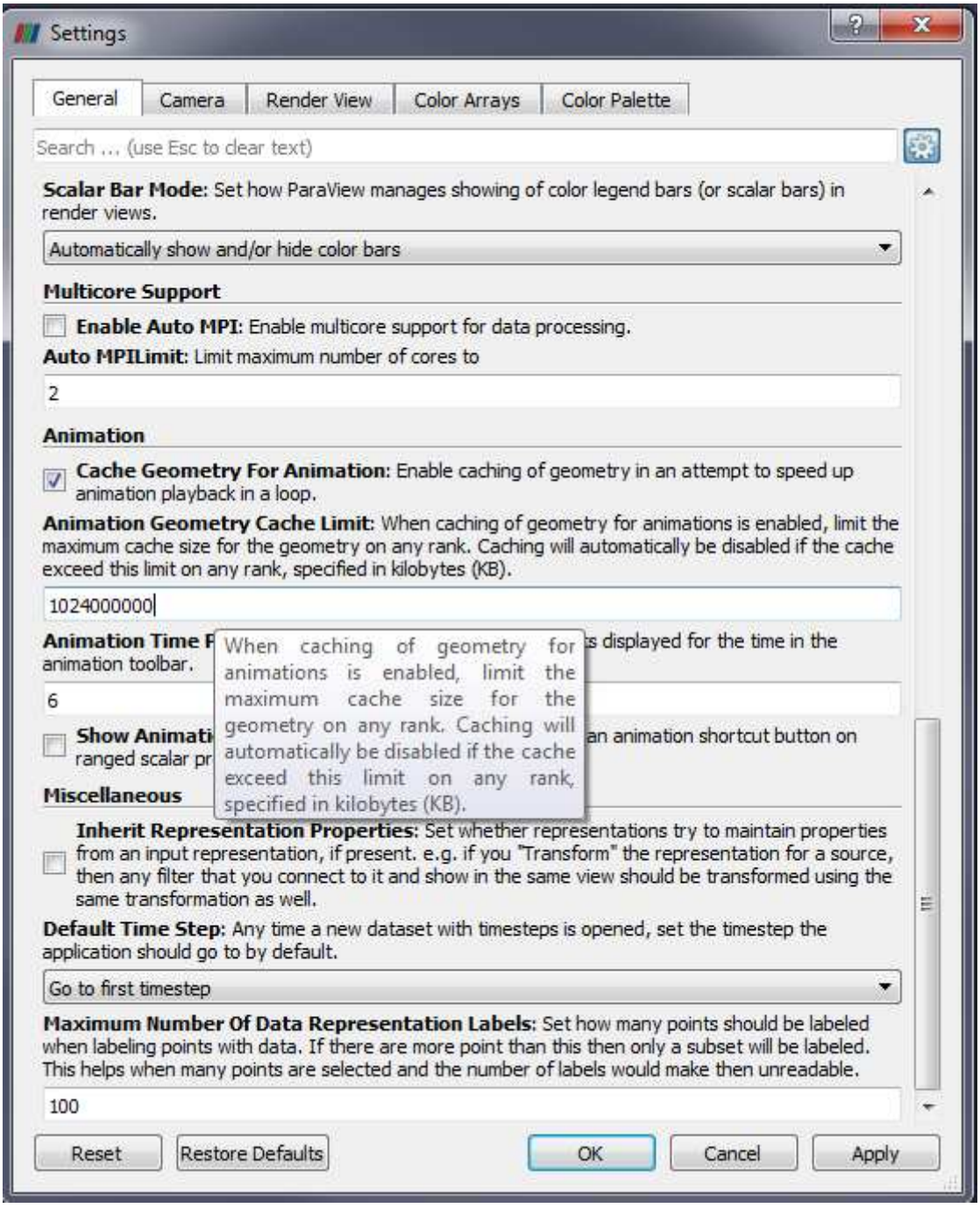

Fig. 5.26 – Eanble Cache Geometry for Animation and set a high value for cache limit

5.10 Catalyst and in-situ visualization

NVIDIA IndeX supports in-situ visualization, a user can run a simulation and visualize it in real-time without writing any data to disk. Catalyst[4] is the co-processing library that enables orchestration of simulation, analysis and visualization tasks together with VTK and ParaView. Catalyst can also be used to setup NVIDIA IndeX and ParaView to do live visualization of your simulation. Please visit the Catalyst website to learn how to write scripts to integrate your simulation and enable in-situ visualization.

[4] `https://www.paraview.org/in-situ/`

As an example, a simple wavlet source can be used to illustrate the Catalyst integration with NVIDIA IndeX rendering. Make sure you have compiled ParaView with Catalyst support before trying to do the live visualization.

You can start 50 iterations of a wavelet data source on a single process by running the following command. Both CatalystWaveletDriver.py CatalystWaveletCoprocessing.py scripts are under the directory `../Applications/ParaView/Testing/XML/` in ParaView source.

```
mpirun -np 1 ./pvbatch -sym CatalystWaveletDriver.py ↦
    CatalystWaveletCoprocessing.py 50
```

Next, start ParaView client and connect to the port where Catalyst is running from the menu [Catalyst ▸ Connect].

```
./paraview
```

Once ParaView connects to Catalyst, enable "input" and click "Extract input" from the pipeline browser. Once the input is extracted you can switch to NVIDIA IndeX representation from the menu.

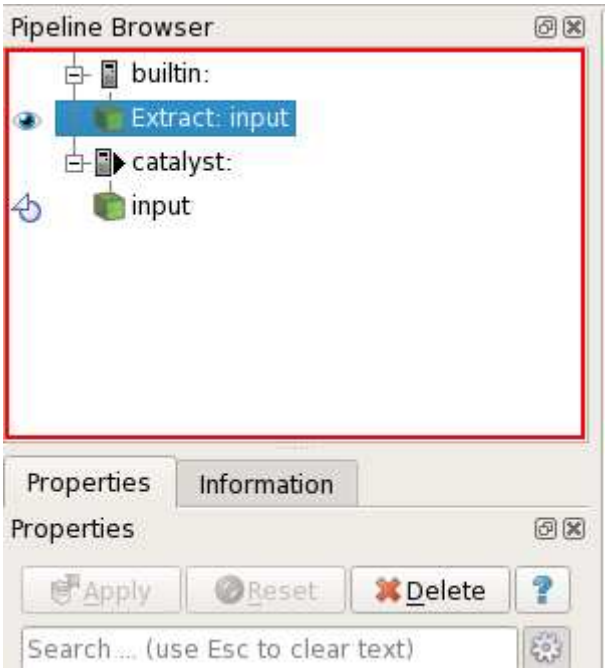

Fig. 5.27 – Enable input and extract input to visualize

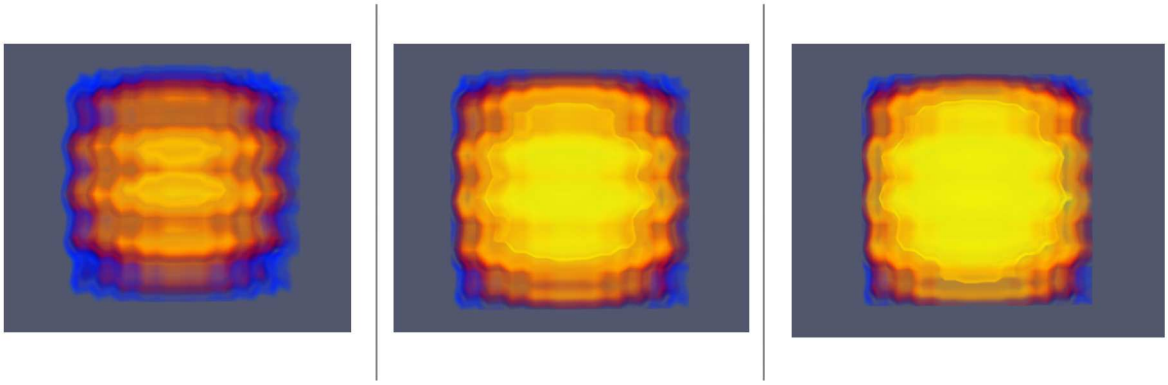

Fig. 5.28 – Wavelet example shown at different iterations

5.11 Mixing ParaView primitives

One of the unique features of the plugin is to mix volume rendering from NVIDIA IndeX along with other primitives such as Glyphs, Streamlines and Surfaces rendered by ParaView.

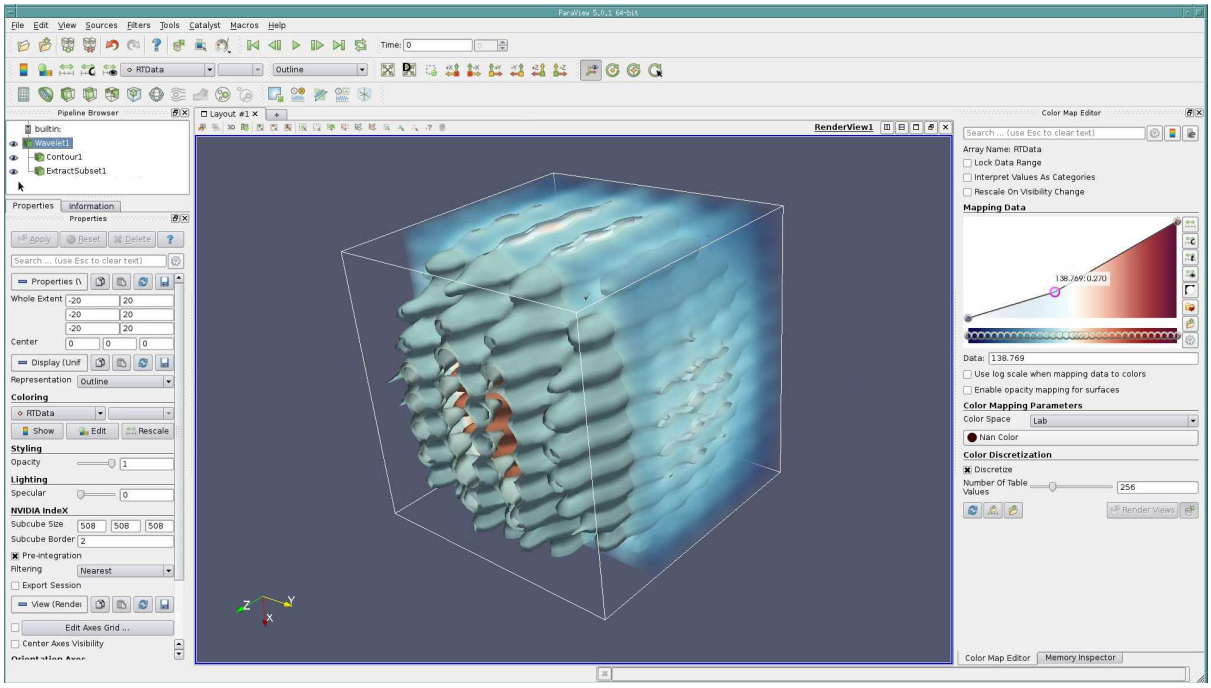

Fig. 5.29 – Wavelet data rendered as a Surface by ParaView and as a volume by NVIDIA IndeX

6 Frequently asked questions

Q: Do I need to install CUDA or any other libraries for using the plugin?

A: There is no need to install CUDA separately as the plugin package is bundled with all the required libraries. However, you need to have an appropriate NVIDIA display driver for your graphics card.

Q: When I load the plugin from ParaView's [Tools ► Manage Plugins] window, libpvNVIDIAIndeX or pvNVIDIAIndeX does not show up as loaded.

A: Make sure you have no errors in ParaView's console or on your terminal where you started ParaView from. These error messages will give you additional information about what the issue might be.

Q: Plugin is loaded successfully without any errors but *NVIDIA IndeX* as a representation does not show up in ParaView's representation dropdown box.

A: Make sure you have loaded a regular-grid volumetric data and it is selected in ParaView's scene graph, ParaView will automatically show representations based on the input data format.

Q: There is an error saying "Failed loading NVIDIA IndeX library" and viewport is empty.

A: This error message is usually printed when NVIDIA IndeX library (in `libnvindex.`) is not found. Make sure you have `libnvindex.` and `libdice.` in your `LD_LIBRARY_PATH` (or PATH on Windows). You can also copy all the libraries from the plugin directly into ParaView's library directories. Refer section-2 for more information.

Q: Viewport is blank when I choose *NVIDIA IndeX* as a representation.

A: Select appropriate *Scalar Array* for the dataset instead of *Solid Color*.

Q: Viewport is blank when I choose *NVIDIA IndeX* as a representation with a *Scalar Array* and not with *Solid Color*

A: This is most likely because of an old NVIDIA display driver, update your display drivers to the recommended versions.

Q: Why does my rendering look down-sampled when I interact?

A: NVIDIA IndeX does not down-sample the data and renders at full resolution. By default ParaView optimizes for high latency networks and enables compression and level of detail, you can disable this from [Edit ► Settings] option and turn off LOD Resolution, Image Reduction Factor and Image Compression.

Q: Can I render multiple volumes at once in the same scene graph in ParaView?

A: While the NVIDIA IndeX library itself supports multi-volume rendering, the ParaView plugin does not yet have this feature integrated so you can only render one volume at a given time.

Q: Can I use NVIDIA IndeX library in my own application without ParaView?

A: Sure you can, contact us for more details.

Q: What if I want to have a feature that is part of NVIDIA IndeX but not integrated in the ParaView plugin?

A: Full set of NVIDIA IndeX features are described on this webpage[5]. If there is a feature that is important for you please contact us, we are happy to take workflow and feature requests.

7 Useful links

- NVIDIA IndeX for ParaView plugin website[6]
- NVIDIA IndeX for ParaView plugin forum[7]
- NVIDIA IndeX website[8]
- ParaView binaries and source code download[9]
- ParaView documentation[10]
- ParaView user guide[11]
- Contact email: `indexintegration@nvidia.com`

[5] `https://developer.nvidia.com/index`
[6] `http://www.nvidia.com/object/index-paraview-plugin.html`
[7] `https://forum.nvidia-arc.com/forumdisplay.php?210-NVIDIA-IndeX-for-ParaView-Plug-in`
[8] `https://developer.nvidia.com/index`
[9] `http://www.paraview.org/download/`
[10] `http://www.paraview.org/documentation/`
[11] `http://www.paraview.org/paraview-guide/`


NVIDIA