ESiWACE Summerschool
Data Visualization using ParaView

Niklas Röber, Michael Böttinger,
Karin Meier Fleischer, Dela Spickermann, Florian Ziemen
Deutsches Klimarechenzentrum (DKRZ)
Wednesday, August 26th – Afternoon

Data Visualization – Niklas Röber DKRZ
- 13.30 – 15.00 Data Visualization using ParaVie
  - Introduction + visualization examples
  - Handling large data in visualization
  - Hands-on examples using ParaView

Data Analytics – Donatello Elia CMCC
- 15.15 – 16.45 Introduction to Data Analytics
- 16.45 – 17.15 Lab Tutorial (Ophidia)
Visualization Workshops at DKRZ

- Hands-on tutorials for ParaView, NCL, VaPOR from 2 and 5 days
- Some online tutorials available at www.dkrz.de/up/services/analysis
- Online video tutorials coming soon
- ESiWACE2: dedicated data analytics and visualization workshop
ParaView

ParaView is an open source visualization package that reads a variety of different data formats and lattices and implements the most common visualization techniques. More specifically, ParaView also reads netCDF files and supports different grids, so that it can be used to visualize climate and earth science data sets.

ParaView 4.1 is installed on all visualization nodes of Halo and can be started from the command line via ‘paraview’. Older versions of ParaView can be started by appending the version number, such as ‘paraview3.98’.

ParaView has come a long way and is used and developed by a very large community from a variety of different sciences. It is installed on DKRZ’s Halo nodes since the end of 2012, and we have now prepared a little tutorial that will teach you how to use ParaView for the visualization of your own climate research data.

More general information on ParaView, along with some tutorial data can also be found online on the ParaView website.

The above example shows a complex visualization of an ICON ocean data set using ParaView. The viewpoint on the right displays the data, the selection made, as well as the Earth’s topography. The three viewports on the left hand side are used to specify the selection, based on a scatterplot matrix and parallel coordinates. These techniques are especially well suited for an in-depth data analysis and exploration.

ParaView Tutorial

The final tutorial document will comprise 8 chapters and will be released at the end of the summer in 2014. Alongside, we will provide courses to teach ParaView in a hands-on setting. The first course will already start in December 2013.

Here is a glimpse of the content from the tutorial:

- Chapter 1 ‘Introduction and Overview’ — The first chapter starts with an overview of ParaView and briefly explains the underlying visualization toolkit pipeline. The second part of this chapter concentrates on an introduction of the...
Visualization Work at DKRZ

- Looking at ways to work and interact with LARGE data
  - In-situ visualization with ParaView/Catalyst
  - Compression and progressive data visualization using wavelets and Vapor
  - Batch visualization on MISTRAL using ParaView and NCL

- Compression, especially *lossy*, as it has always been done
  (precision, variables, temporal/spatial resolution, model error, GRIB)

- Visualization of uncertainty
- Multivariate data visualization
- Machine learning & online feature tracking
See, understand, learn, communicate ...

- Confirmatory visualization
- Exploratory visualization
-Creating animations & stills for **communication**
ICON DYAMOND R2B10 2.5km Resolution
01.08.2016 at 00:00
Spilhaus Projection
Canary Islands

DYAMOND R2B10 - 2D Wind Visualization
(3 Minute Output - 10m Height)
ICON Earth System Model (D++ Setup)
- 21 GPU nodes (two Haswell/Boardwell, 256/512/1024 GB memory)
- 4 GPUs per node (two dual Kepler/Maxwell)
- Software: NCL, ParaView, VaPOR, IDL, Python
## Visualization Software on Mistral

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>URL</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-specific</td>
<td>NCL</td>
<td><a href="http://www.ncl.ucar.edu/">http://www.ncl.ucar.edu/</a></td>
<td>2D script-based</td>
</tr>
<tr>
<td></td>
<td>IDV</td>
<td><a href="http://www.unidata.ucar.edu/software/idv/">http://www.unidata.ucar.edu/software/idv/</a></td>
<td>2D/3D interactive GUI</td>
</tr>
<tr>
<td></td>
<td><strong>Vapor</strong></td>
<td><a href="https://www.vapor.ucar.edu/">https://www.vapor.ucar.edu/</a></td>
<td>3D interactive GUI</td>
</tr>
<tr>
<td></td>
<td>UV-CDAT</td>
<td><a href="http://uvcdat.llnl.gov/">http://uvcdat.llnl.gov/</a></td>
<td>Collection: 2D /3D tools</td>
</tr>
<tr>
<td></td>
<td>GrADS</td>
<td><a href="http://cola.gmu.edu/grads/">http://cola.gmu.edu/grads/</a></td>
<td>2D script-based</td>
</tr>
<tr>
<td></td>
<td>GMT</td>
<td><a href="http://gmt.soest.hawaii.edu/">http://gmt.soest.hawaii.edu/</a></td>
<td>2D script-based</td>
</tr>
<tr>
<td></td>
<td><strong>PyNGL / PyNIO</strong></td>
<td><a href="https://www.pyngl.ucar.edu/Download/">https://www.pyngl.ucar.edu/Download/</a></td>
<td>2D script-based</td>
</tr>
<tr>
<td>General-purpose</td>
<td>ParaView</td>
<td><a href="http://www.paraview.org/">http://www.paraview.org/</a></td>
<td>3D interactive GUI</td>
</tr>
<tr>
<td></td>
<td>Visit</td>
<td><a href="https://visit.llnl.gov/">https://visit.llnl.gov/</a></td>
<td>3D interactive GUI</td>
</tr>
<tr>
<td></td>
<td>Avizo</td>
<td><a href="https://www.fei.com/software/avizo3d/">https://www.fei.com/software/avizo3d/</a></td>
<td>3D interactive GUI</td>
</tr>
<tr>
<td></td>
<td>IDL</td>
<td><a href="http://www.harrisgeospatial.com/">http://www.harrisgeospatial.com/</a></td>
<td>2D script-based</td>
</tr>
<tr>
<td></td>
<td>Python / matplotlib</td>
<td><a href="http://matplotlib.org/">http://matplotlib.org/</a></td>
<td>2D script-based</td>
</tr>
</tbody>
</table>


Parallel Processing and Visualization
Large Data Visualization

In Situ Visualization
(ParaView/Catalyst)

Progressive Visualization
(VAPOR)

Simulation
Adaptor
ParaView/Catalyst
Results

Simulation
Decomposition
VAPOR
Results

HPC System
Workstation
From Post Visualization to In-Situ
ParaView
Visualization Pipeline generates Python Script

FORTRAN
ICON Model

Catalyst VTK/C++

Catalyst Adaptor

- Rendered images
- Cinema database
- Data reduction (par. I/O)
- Feature det./tracking (e.g. cloud classification)
- Live visualization
- Data decomp./comp.
Advantages

- Much less I/O
  -> Simulation faster / less disk
- Preview of data
- In situ feature tracking
- Analyze extremely large simulation “output”
- Time to knowledge shorter

Drawbacks

- Additional resources required
- A priori knowledge needed
- Need to run sim/vis again for new analysis/visualization
- Workflow complexity increases
- Statistical analysis more complex
Generating a Catalyst Script
Implementation & Status

- Started refactoring other in-situ code -> too complex
- Started fresh -> few hundred lines in FORTRAN and C++ with minimal changes to ICON
- Zero copy arrays FORTRAN -> C++
- Tightly coupled (in line) w/ even number of sim/vis processes
- Prototype available on Mistral for ICON
- Development of workflows
## Timings R2B10 – 2.5km global / 540 nodes

<table>
<thead>
<tr>
<th>name</th>
<th># calls</th>
<th>t_min</th>
<th>min r</th>
<th>t_avg</th>
<th>t_max</th>
<th>max r</th>
<th>total min (s)</th>
<th>total min r</th>
<th>total max (s)</th>
<th>total max rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>4305</td>
<td>06m48s</td>
<td>[6]</td>
<td>06m48s</td>
<td>408.010</td>
<td>[659]</td>
<td>408.027</td>
<td>[3919]</td>
<td>408.027</td>
<td>[3919]</td>
</tr>
<tr>
<td>L integrate_nh</td>
<td>344400</td>
<td>3.9458s</td>
<td>[34]</td>
<td>4.3407s</td>
<td>15.2784s</td>
<td>[256]</td>
<td>347.016</td>
<td>[3476]</td>
<td>347.308</td>
<td>[0]</td>
</tr>
<tr>
<td>L nh_solve</td>
<td>1722000</td>
<td>0.29028s</td>
<td></td>
<td>0.45025s</td>
<td>1.1131s</td>
<td>[216]</td>
<td>156.504</td>
<td>[2048]</td>
<td>190.621</td>
<td>[47]</td>
</tr>
<tr>
<td>L nh_hdiff</td>
<td>344400</td>
<td>0.09548s</td>
<td>[13]</td>
<td>0.13672s</td>
<td>0.44944s</td>
<td>[420]</td>
<td>8.426</td>
<td>[2111]</td>
<td>15.589</td>
<td>[1852]</td>
</tr>
<tr>
<td>L physics</td>
<td>344400</td>
<td>0.53099s</td>
<td>[4]</td>
<td>0.53099s</td>
<td>0.53099s</td>
<td>[418]</td>
<td>57.132</td>
<td>[421]</td>
<td>101.765</td>
<td>[2831]</td>
</tr>
<tr>
<td>L insitu_set_var</td>
<td>344400</td>
<td>0.01999s</td>
<td></td>
<td>0.02430s</td>
<td>0.05760s</td>
<td>[221]</td>
<td>1.663</td>
<td>[7]</td>
<td>2.037</td>
<td>[126]</td>
</tr>
<tr>
<td>L insitu_do_work</td>
<td>340095</td>
<td>0.00014s</td>
<td></td>
<td>0.00014s</td>
<td>0.00014s</td>
<td>[1095]</td>
<td>5.312</td>
<td>[2392]</td>
<td>10.067</td>
<td>[0]</td>
</tr>
<tr>
<td>L insitu_do_work1st</td>
<td>4305</td>
<td>1.5387s</td>
<td>[22]</td>
<td>1.6174s</td>
<td>2.033</td>
<td>[0]</td>
<td>1.539</td>
<td>[2239]</td>
<td>2.033</td>
<td>[0]</td>
</tr>
</tbody>
</table>

#### Additional Notes
- **06m48s**: Total time (min) for all calls.
- **408.027**: Maximum total time (max) across all calls.
- **0.02430s**: Minimum time (min) for the function `L insitu_set_var`.
- **0.06853s**: Average time (avg) for the function `L insitu_do_work`.
- **1.6174s**: Maximum time (max) for the function `L insitu_do_work1st`.
- **01m11s**: Total time (min) for the function `model_init`.
- **215.458**: Maximum total time (max) across all calls.
Hands-on Examples with ParaView

https://nextcloud.dkrz.de/s/LqDFNxyaLBMcycXc
The projects ESiWACE and ESiWACE2 have received funding from the European Union’s Horizon 2020 research and innovation program under grant agreements No 675191 and No 823988.

www.esiwace.eu