Parallel Visualization

The Visualization Pipeline

ParaView and VisIt

Dr. Jean M. Favre
Scientific Computing Research Group

30-09-2011

Les outils installes au CSCS

ParaView
VisIt

VMD, Molekel

Matematica
Matlab
Tecplot360

HDF5, NetCDF, ADIOS, Silo

Agenda

- Motivation by examples
- System architecture
- VTK Data Streaming
- ParaView and VisIt architectures

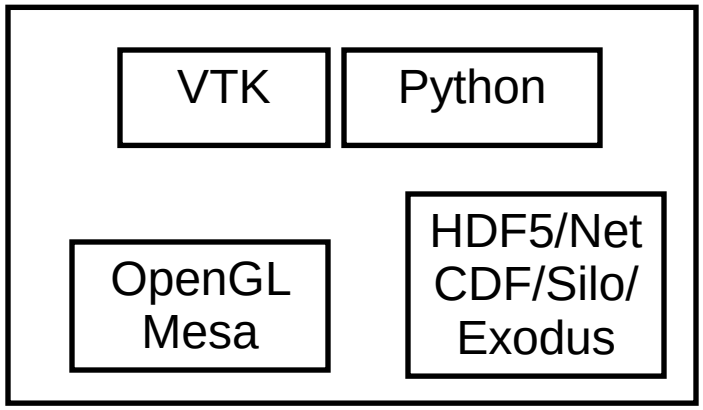
Resources:

- http://paraview.org/Wiki/ParaView/Users_Guide/Introduction
- http://visitusers.org/index.php?title=Main_Page

VTK (now version 5.8) is the *de-facto* standard

The Visualization ToolKit (VTK) is an open source, freely available software system for 3D computer graphics, image processing, and visualization.

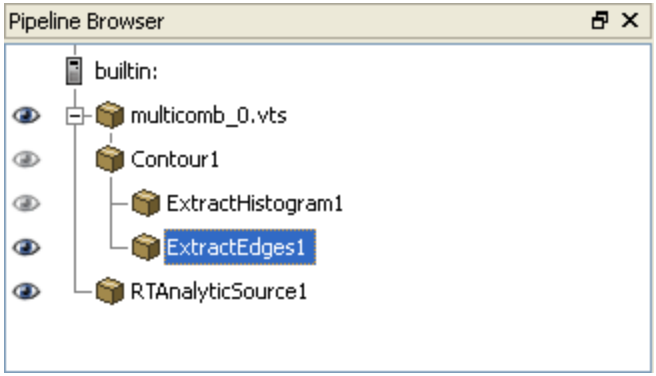
ParaView & Visit are end-user applications based on VTK, with support for:

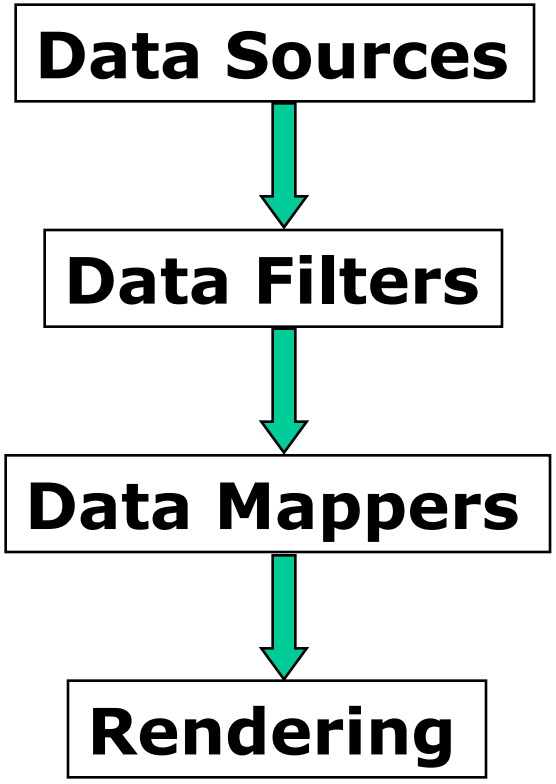


- Parallel Data I/O
- Parallel Processing
- Parallel Rendering
- Single node, client-server, MPI cluster rendering

The VTK visualization pipeline, lesson 1

VTK's main execution paradigm is the *data-flow*, i.e. the concept of a downstream flow of data






```
Filter.SetInputConnection(Source.GetOutputPort())
Mapper.SetInputConnection(Filter.GetOutputPort())
```

ParaView's Filters


Contour


Cut


Clip


Threshold


Extract grid


Warp vector


Stream lines


Integrate flow


Surface vectors


Glyph

etc…

VisIt's Operators


Elevate


IsoVolume


ThreeSlice


Coord Swap


Onion Peel

Reflect

InverseGhostCells

Create Bonds

Dual Mesh

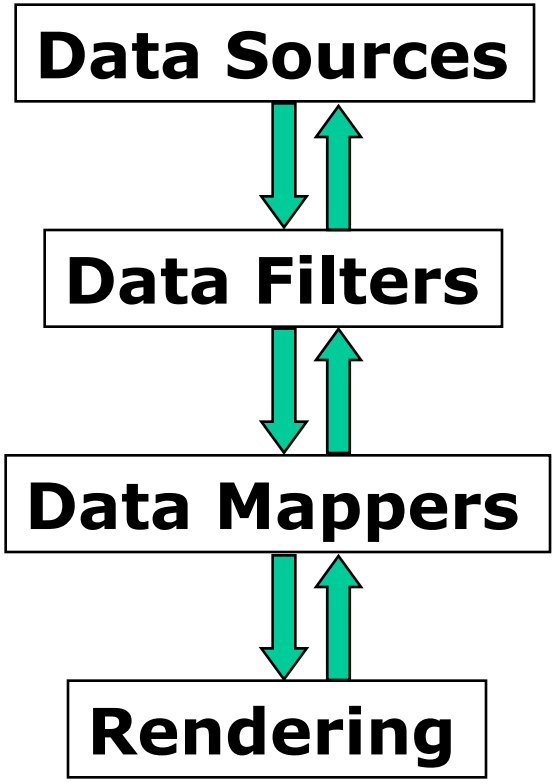
etc…

The VTK visualization pipeline, lesson 2

- VTK extends the paradigm of *data-flow*
- VTK acts as an *event-flow* environment, where data flow downstream and events (or information) flow upstream

The Rendering drives the execution.
Similar to a *load-on-demand*.

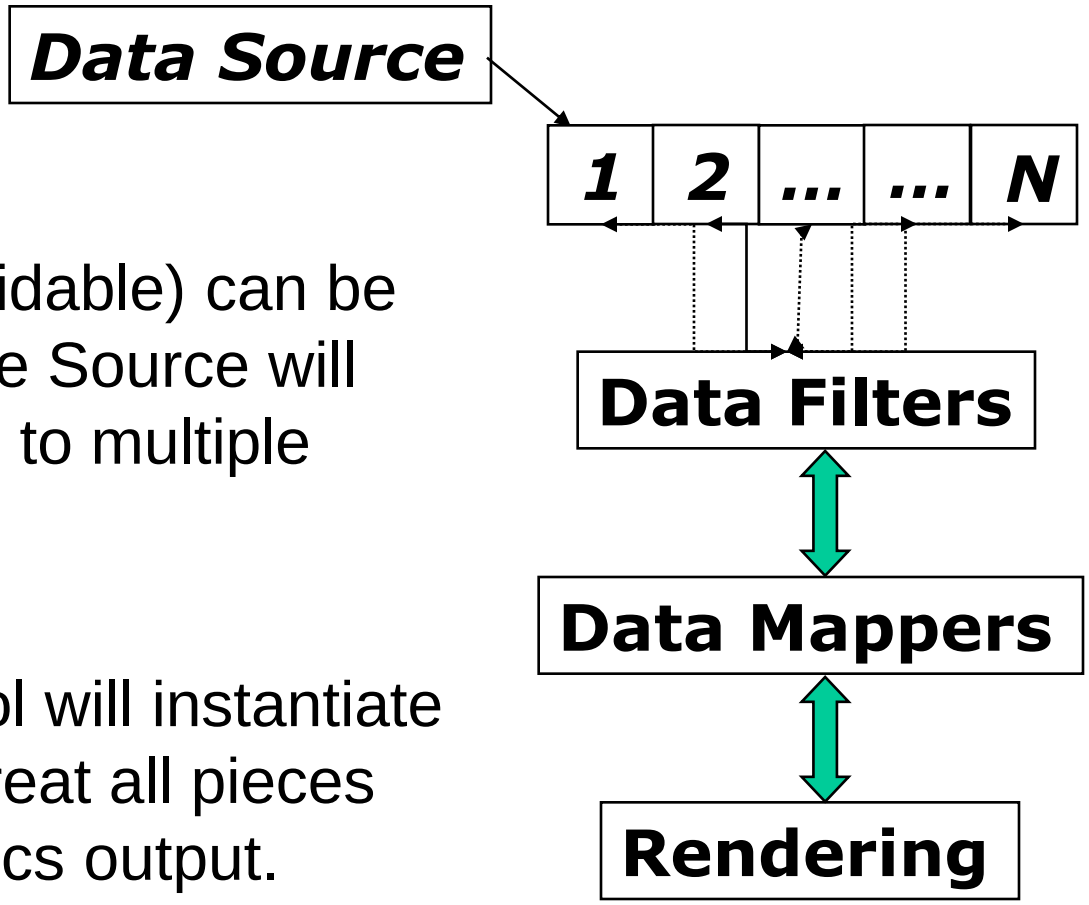
view.StillRender() will trigger the execution.



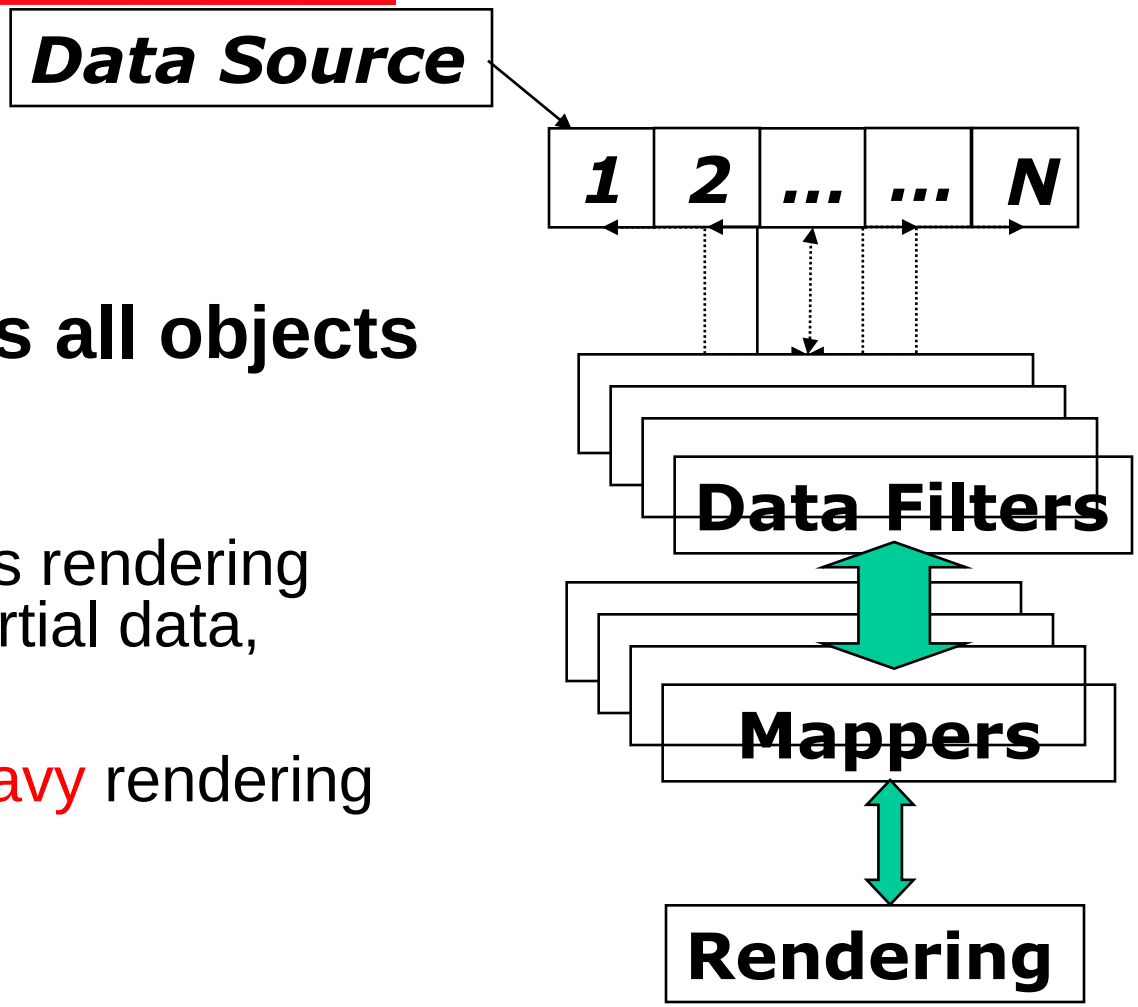
The VTK visualization pipeline, lesson 3

- Large data (when dividable) can be treated by pieces. The Source will distribute data pieces to multiple execution engines

- The Visualization Tool will instantiate parallel pipelines to treat all pieces and create the graphics output.



First rendering option



1) The client collects all objects to be rendered

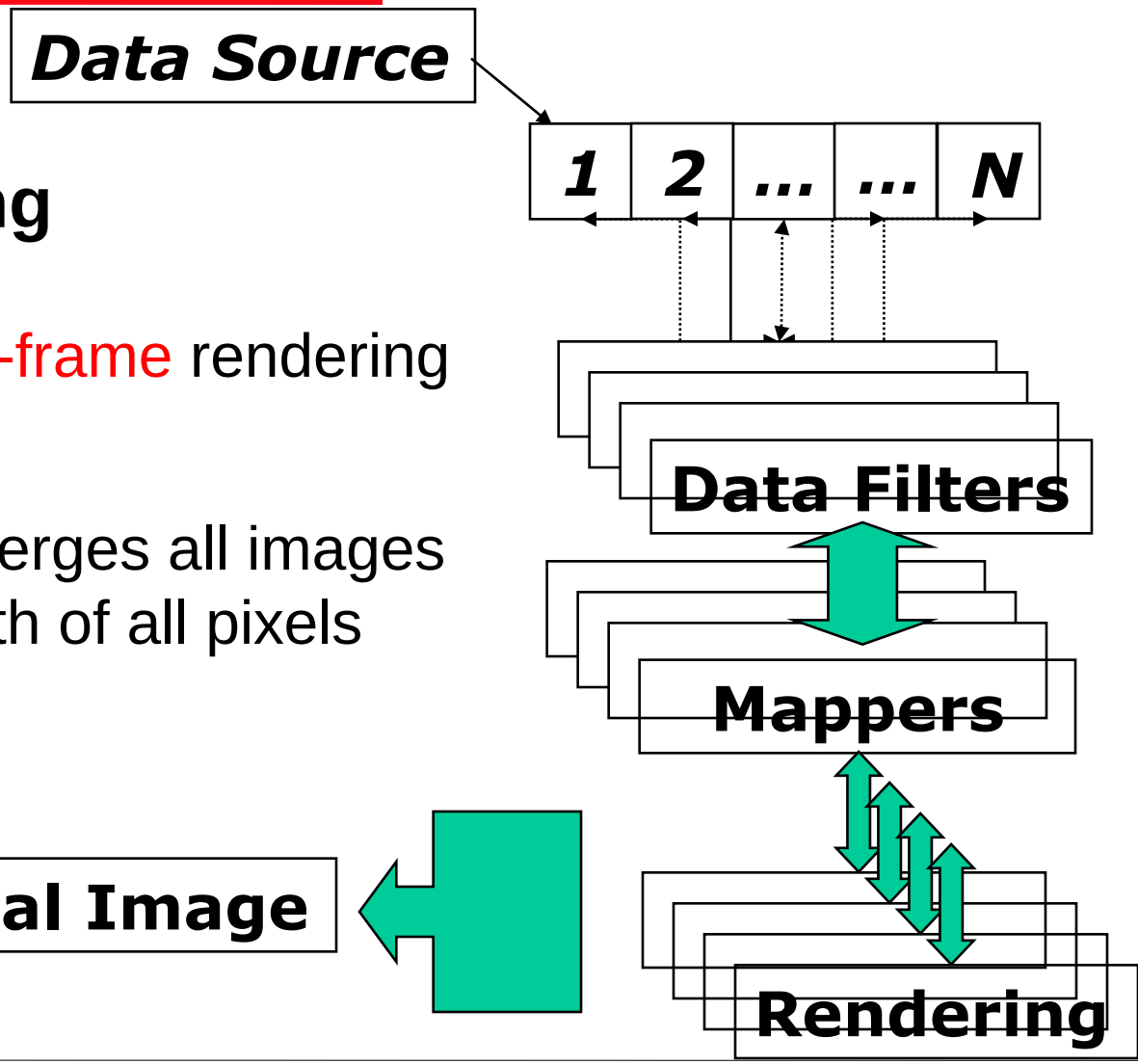
- Each pipeline creates rendering primitives from its partial data,
- The client does a heavy rendering

Second rendering option

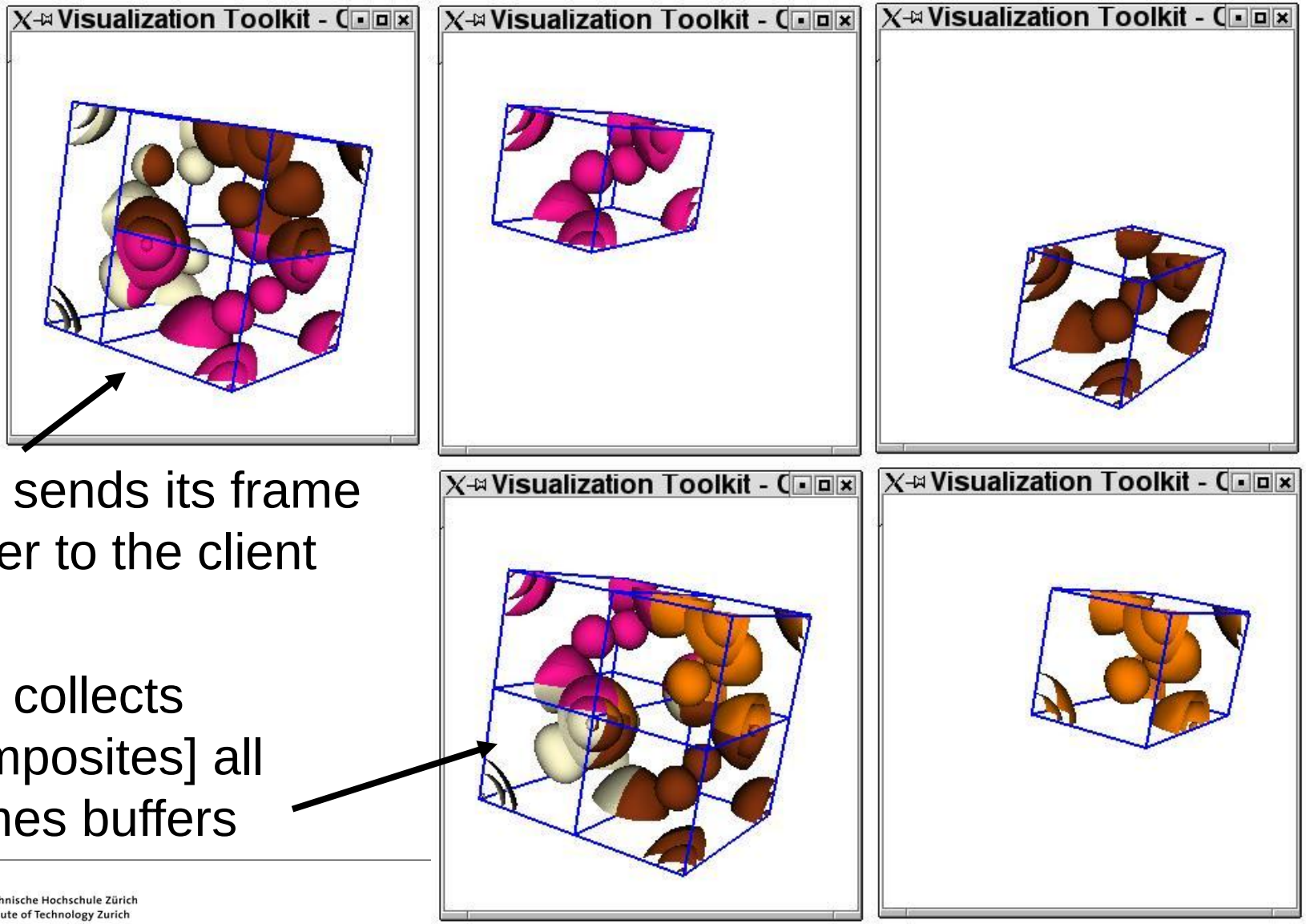
2) Sort-last rendering

Each pipeline does a full-frame rendering of its partial data

An image compositor merges all images by comparing Z-depth of all pixels



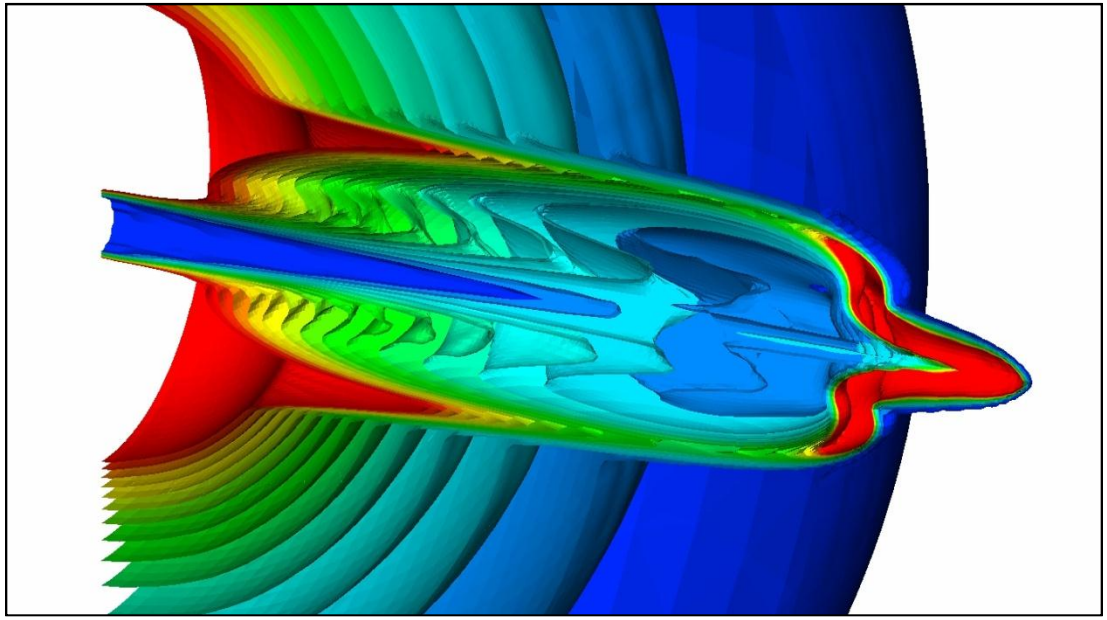
Sort-last rendering [pixel compositing]



Node 0 sends its frame buffer to the client

Node 0 collects [composites] all frames buffers

Arbitrary (or adaptive) 3-D data partitioning



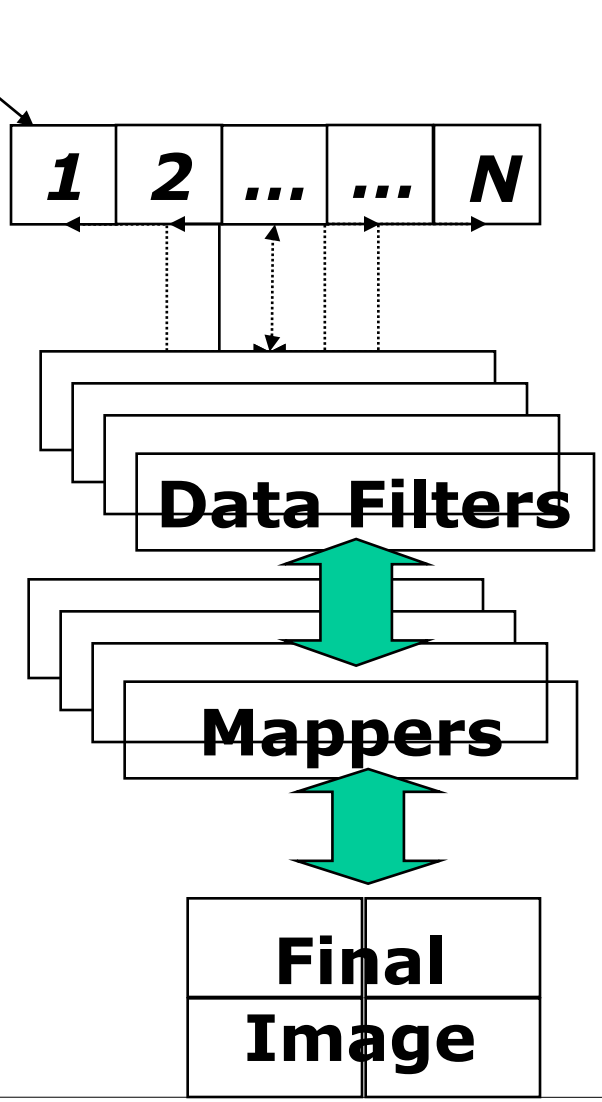
Sort-last rendering is great, fast, order-independent,…

Except if the drawings are semi-transparent

Third rendering option

3) Tiled-Display

Each renderer does a partial-frame rendering of the full data

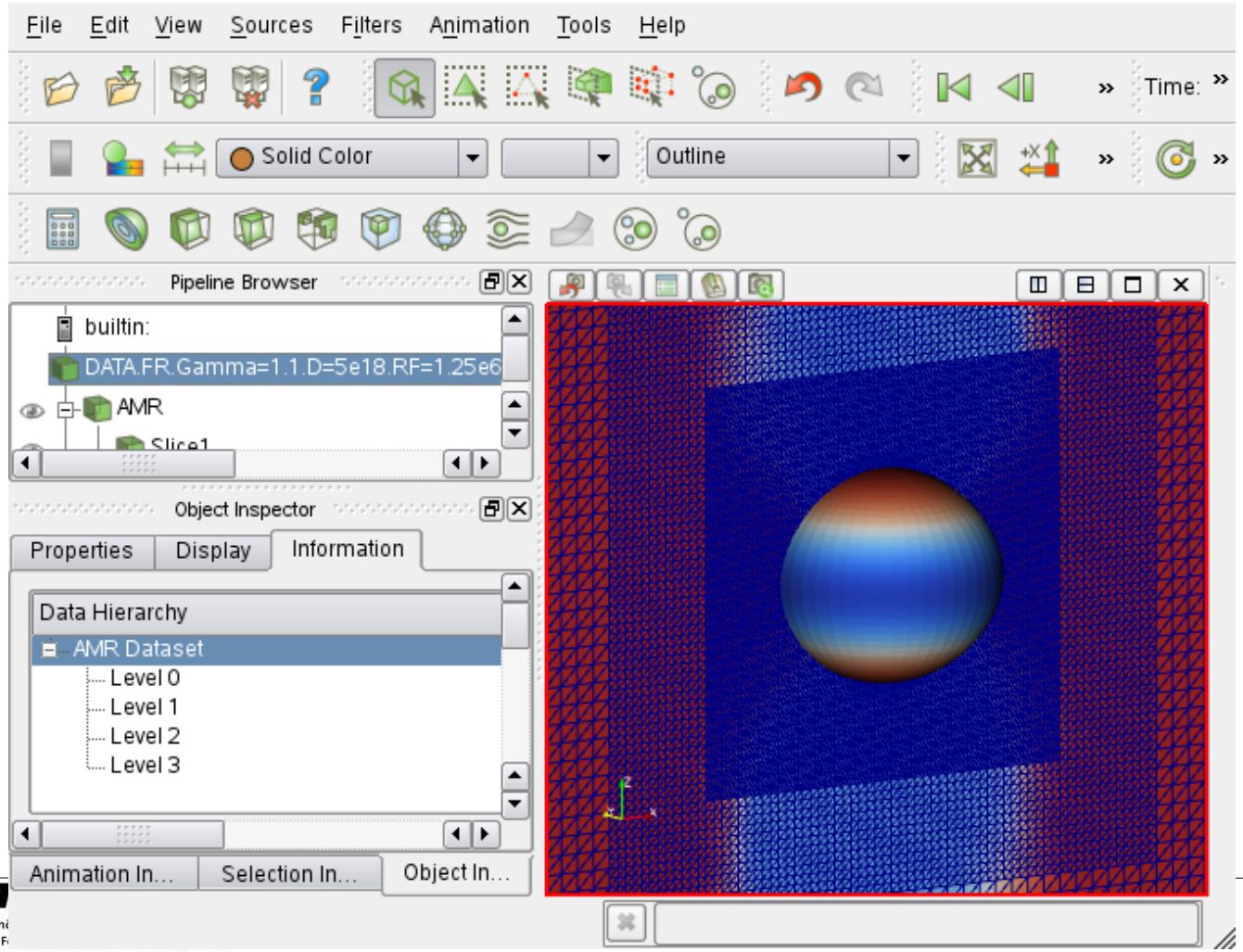


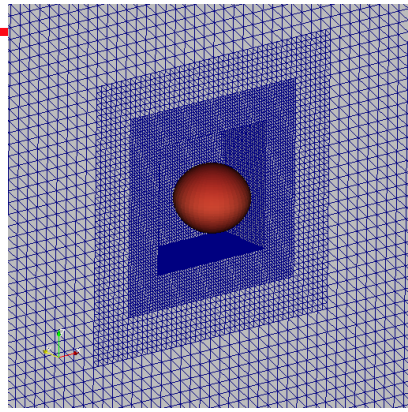
When very large data require distributed processing

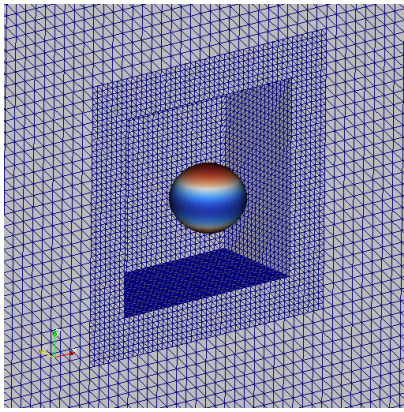
- Sub-sampling can help prototype a visualization
  - As long as the data format/reader supports it.
  - use the Xdmf format, or VisIt multires operator
- Piece-wise processing (on a single node)
  - Data streaming (when the whole visualization will not fit in memory)
  - See ParaView 3.12
- Distributed processing (on multiple nodes)
  - Parallel file I/O
  - Parallel processing
  - Parallel rendering

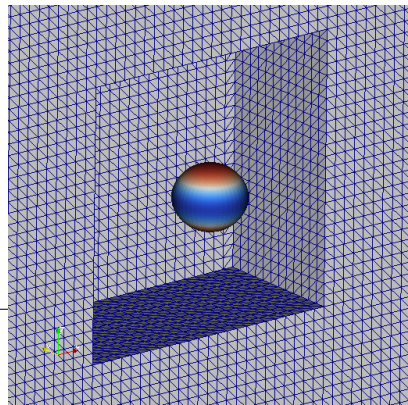
Hierachical data encoding are a plus!

- AMR datasets, or wavelet-encoded data



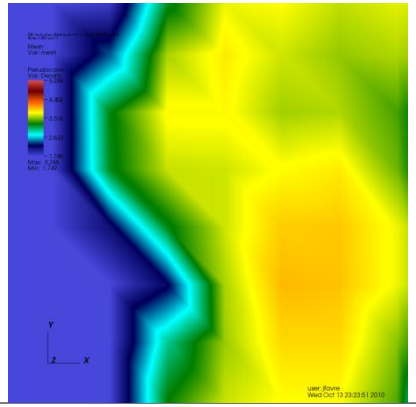
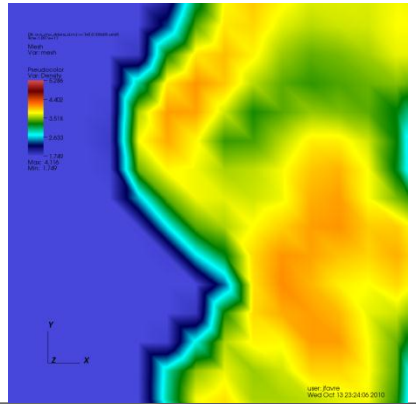
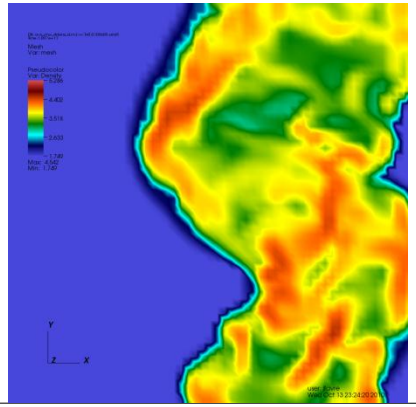
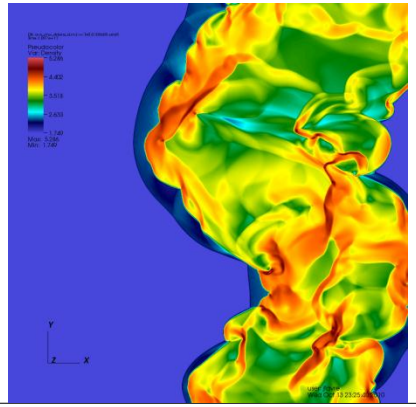






When there is too much data…

- multi-resolution, on-demand

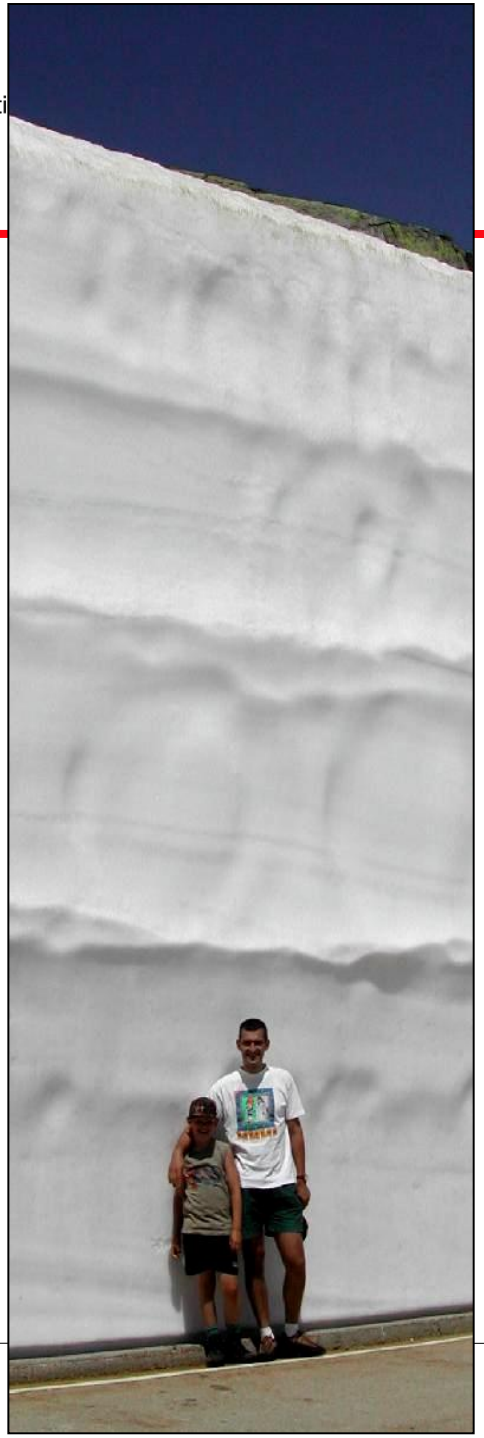





Sub-sampling, streaming or multi-pass…

- The snow removal was done in about 5 passes

  Data Streaming = Divide and conquer

- Load datasets of **any size** by splitting the volumes in pieces
- Process the split data



Example: Digital Elevation Model

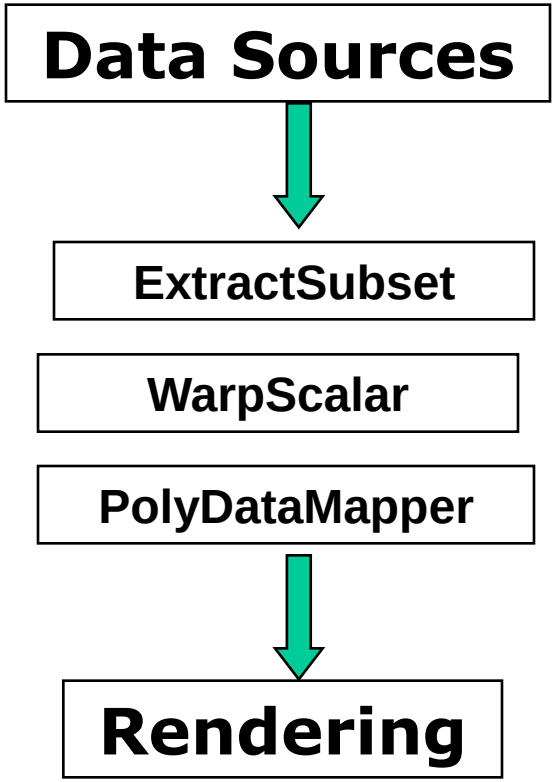
The VTK file header =>

```
# vtk DataFile Version 3.0
European DEM File
BINARY
DATASET STRUCTURED_POINTS
DIMENSIONS 8319 7638 1
ORIGIN 0 0 0
SPACING 1 1 1
POINT_DATA 63540522
```

Use sub-sampling when data are too big

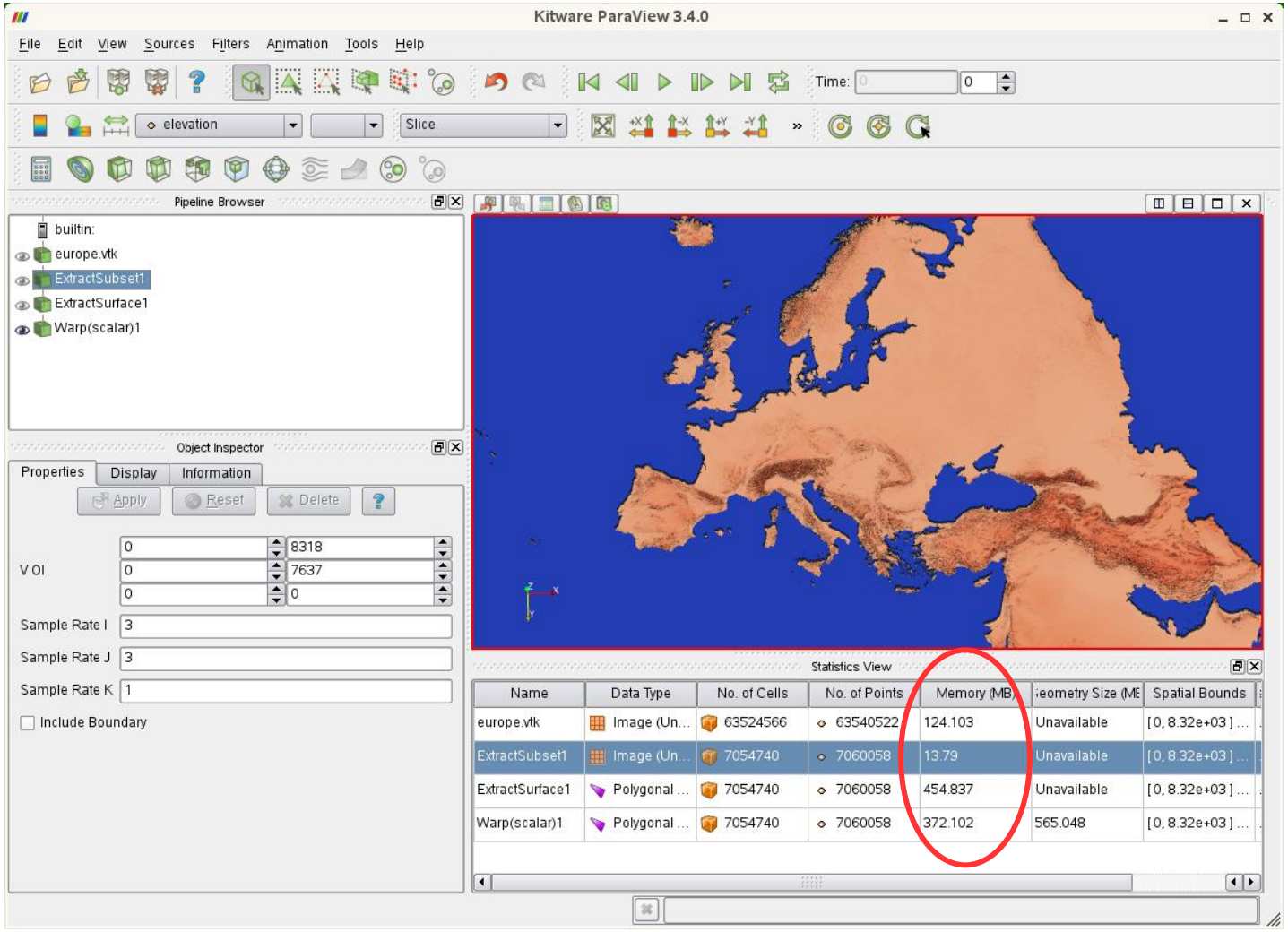
Warning: 64 millions points are first read in memory, then sub-sampled

The memory footprint can still be huge



http://paraview.org/Wiki/ParaView/UsersGuide/Recommendations

Memory usage blows-up down the pipeline...

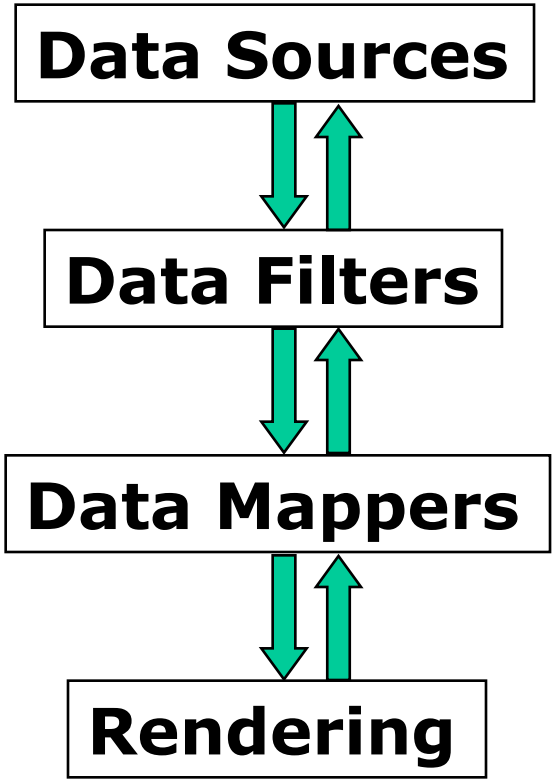


Data Streaming in VTK

- Data larger than memory can be easily treated
  - Piece by piece
  - Releasing or re-using memory after each subset
  - Optionally accumulating sub-object representations for the final image
- The upstream filters should be prepared to handle piece requests of any size
- Each filter can translate the piece request

Update the VTK pipeline in several steps

- The VTK pipeline enables a two-way exchange of data/information.
- The renderer drives the request for data updates.

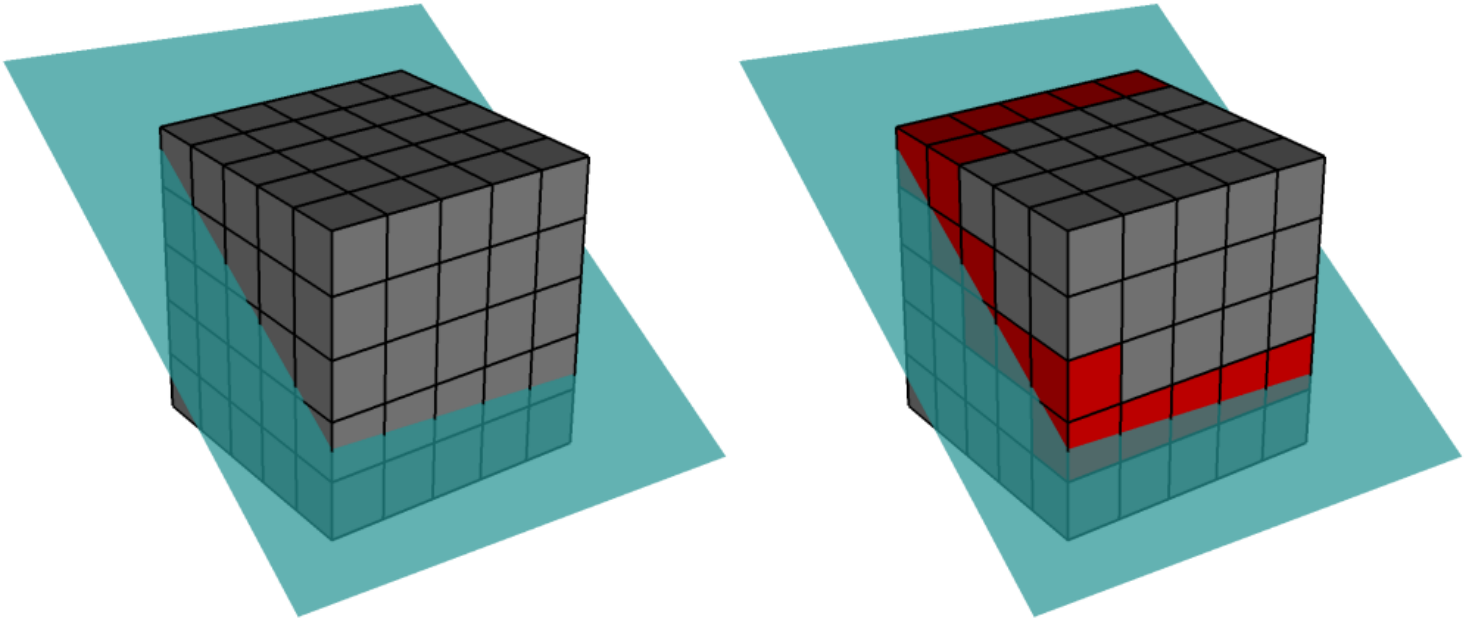


- First pass. Advertise Meta Data: Get general bounds information, without reading the data
- Second pass: Decide how to sub-divide and process pieces

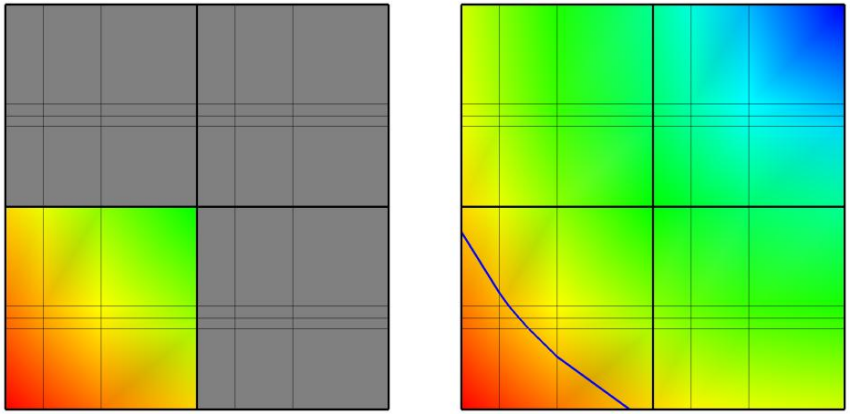
VisIt extends this notion even more (1)

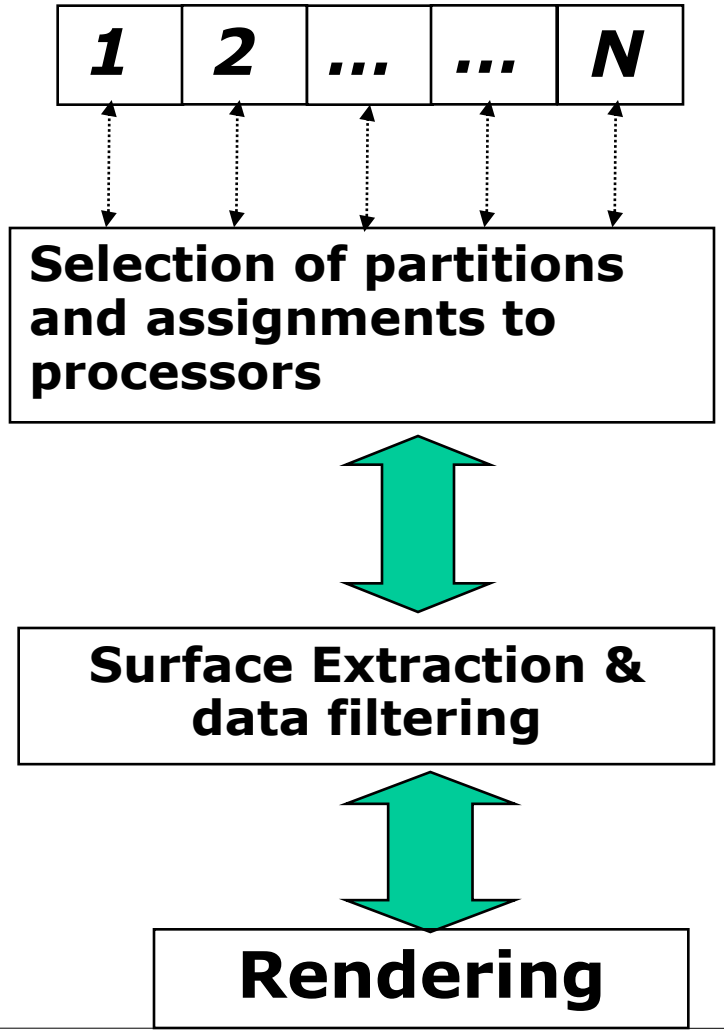
Spatial Extents can be assigned

If the block partitioner receives spatial hints, VisIt will not load the data in memory



VisIt extends this notion even more (2)





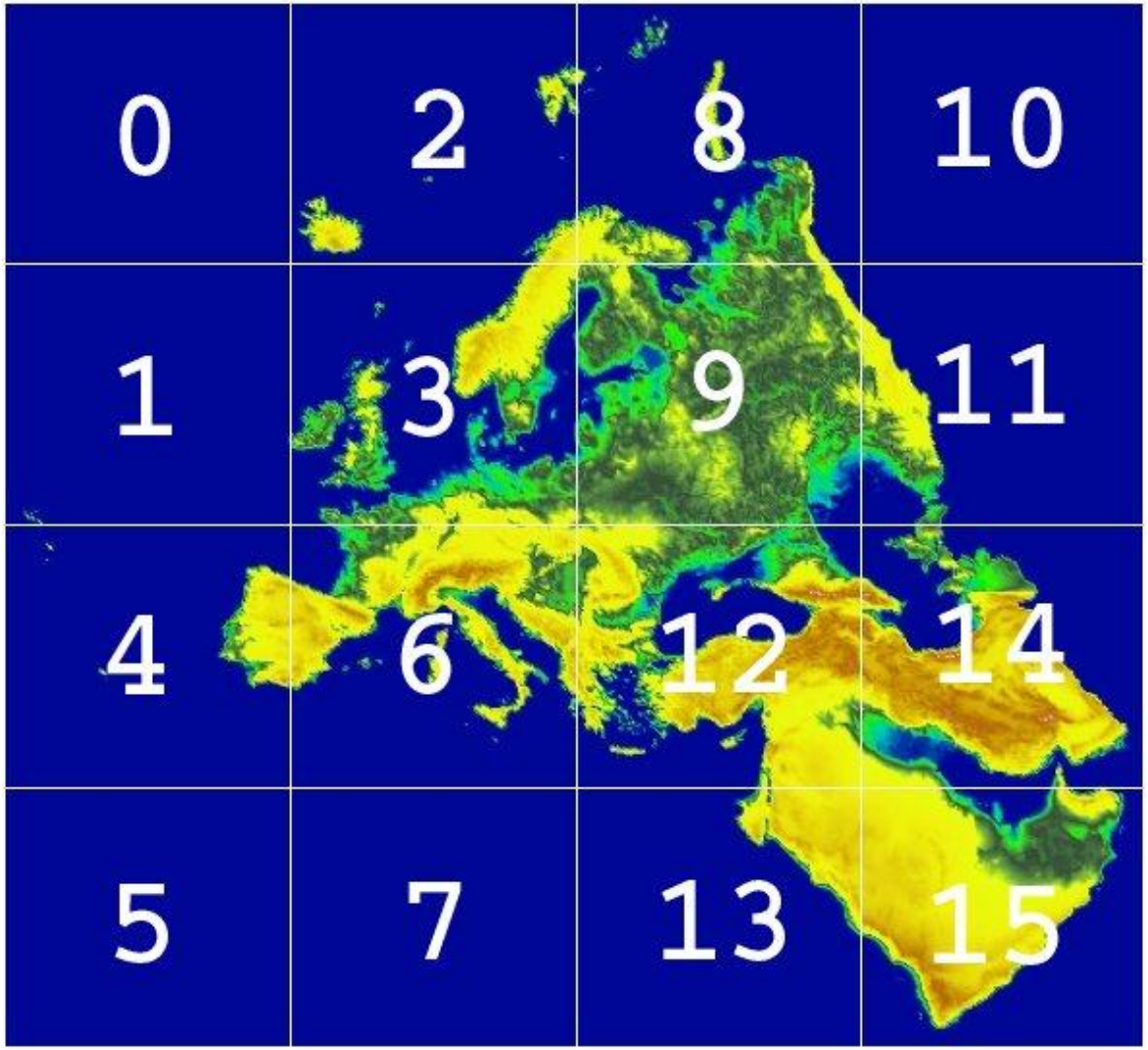
Data Extents can be given

Example: ADIOS lib

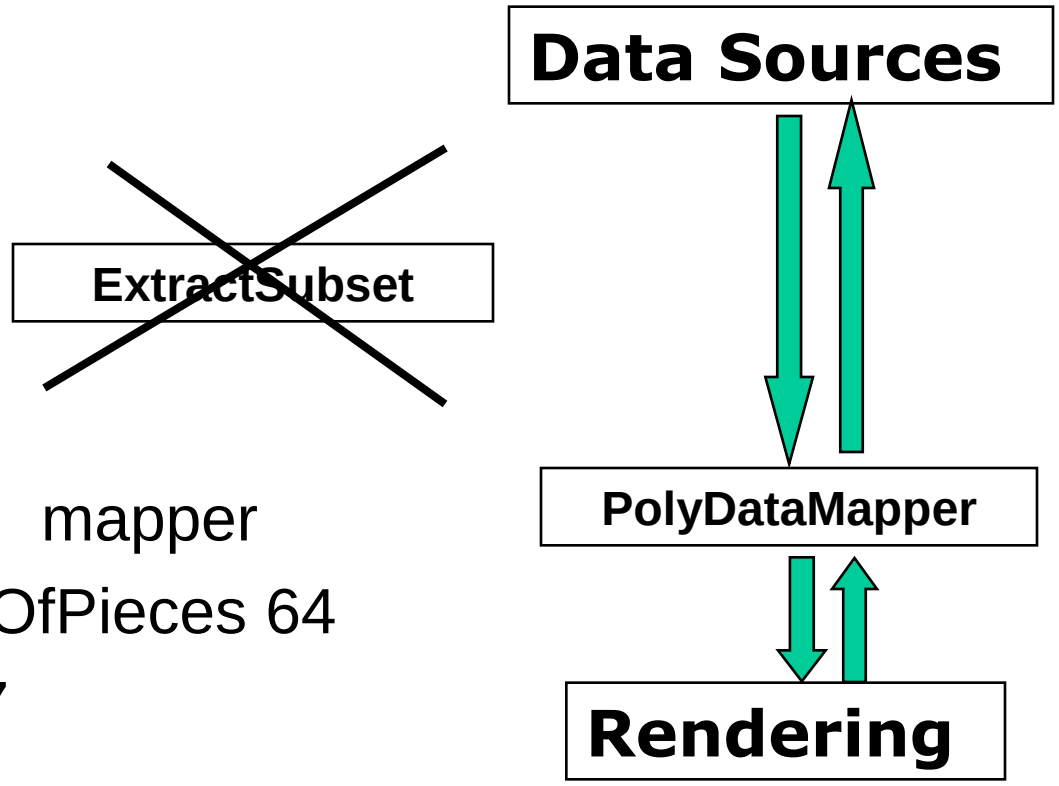
VisIt will not load the data block in memory

The Extent Translator

The Extent Translator does a binary subdivision of the data and let the user access pieces one at a time



Streaming the data



vtkPolyDataMapper mapper

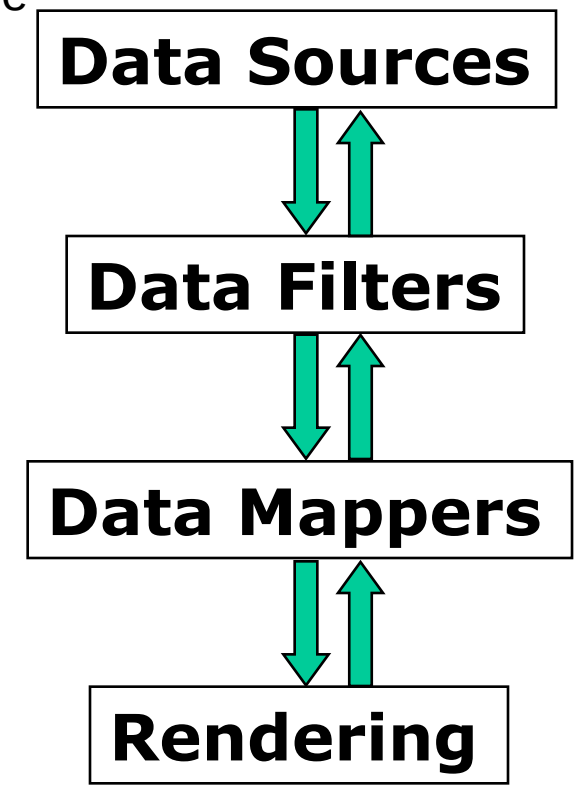
mapper SetNumberOfPieces 64

mapper SetPiece 27

Streaming enables interactive exploration

Rendering speed is linearly increasing according to the number of pieces

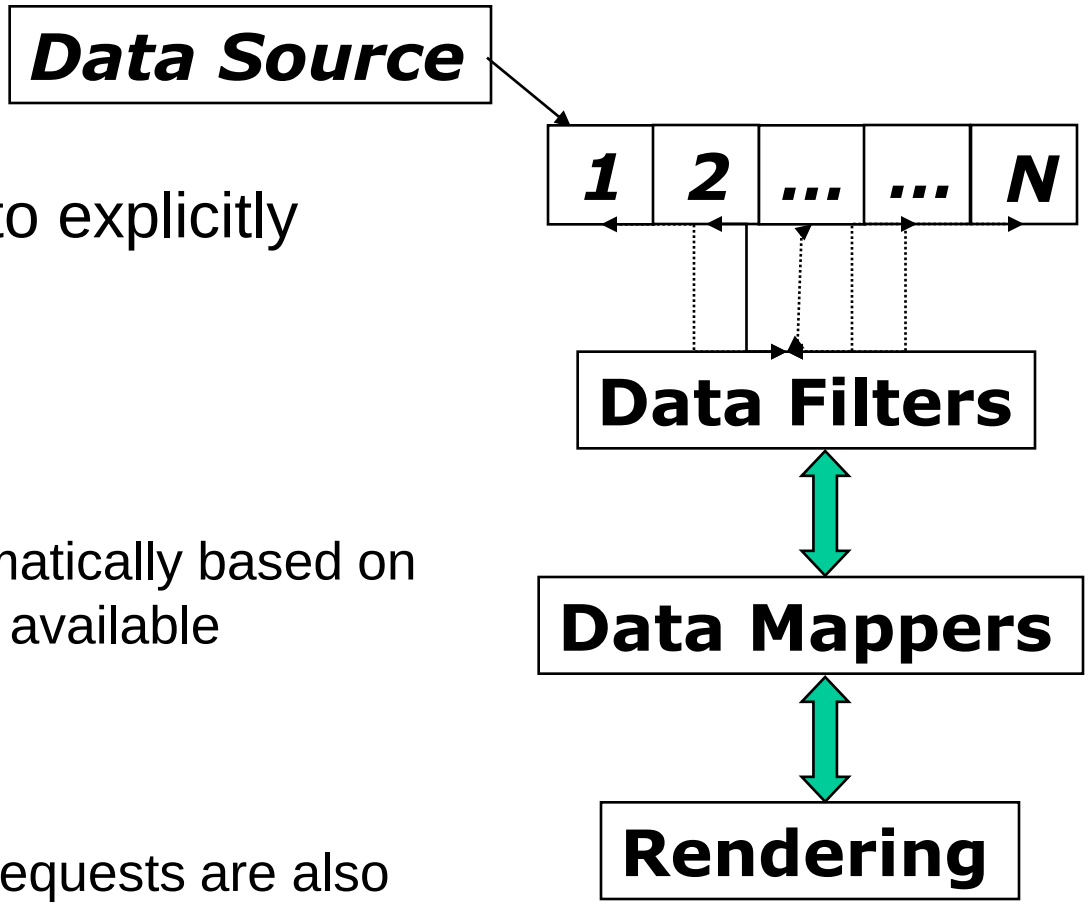
# of Pieces	Rendering Speed (sec./frame)
128	0.13
64	0.21
32	0.44
16	0.88
8	1.78
4	3.55
2	7.04
1	14.00



Pipeline management is hard

The User does not have to explicitly manage the pipeline

- Data Parallelism
  - data are divided automatically based on the number of servers available
- Transient Data
  - time-dependent data requests are also managed similarly via the two-way pipeline data exchange



ParaView & VisIt offer the state-of-the-art

Which one should you choose?

vi	or	emacs
CUDA	or	OpenCL
ParaView	or	VisIt

Can you read your data?

ParaView

- Exodus reader
- Line Integral Convolution
- Interaction Widgets are much nicer to use

VisIt

- Silo, NEK5000 reader
- Queries, expressions, data-level comparisons are much easier to operate
- Python interface is easier

My personal approach is to write data I/O interfaces which create VTK objects + 2 wrappers for ParaView and VisIt

Python

With ParaView:

```
di = Data.GetDataInformation()
ddi = di.DataInformation

ddi.GetBounds()
ddi.GetNumberOfPoints()
```

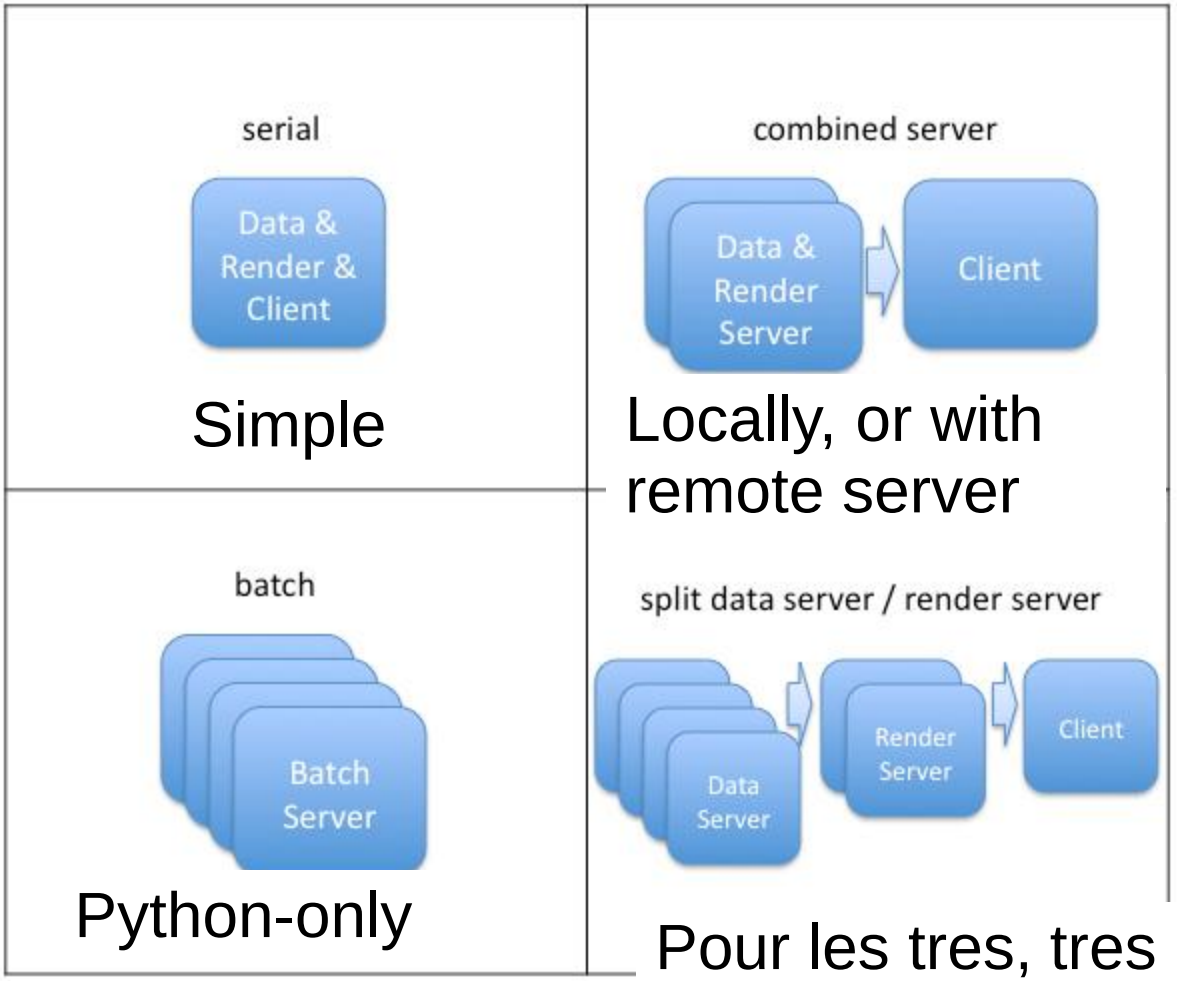
With VisIt:

```
Query("SpatialExtents")

Query("NumNodes")
```

Modi operandi

- Prototyper, avec la GUI, ou Python
- N'utiliser pas ssh –X, mais plustot les compressions et image transfer de Ice-T
- Attention aux I/O



Modi operandi

ParaView has 6 executables

- paraview
- pvbatch
- pvpython
- pvserver
- pvdataserver
- pvrenderserver

VisIt has 5 executables

- mdserver
- cli
- engine_ser
- engine_par
- viewer