



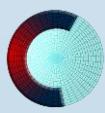
In-Situ Visualization with ParaView Catalyst



For ParaView Catalyst 1.0

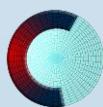
Agenda

- Introduction to ParaView Catalyst
- Catalyst for Users
- Catalyst for Developers

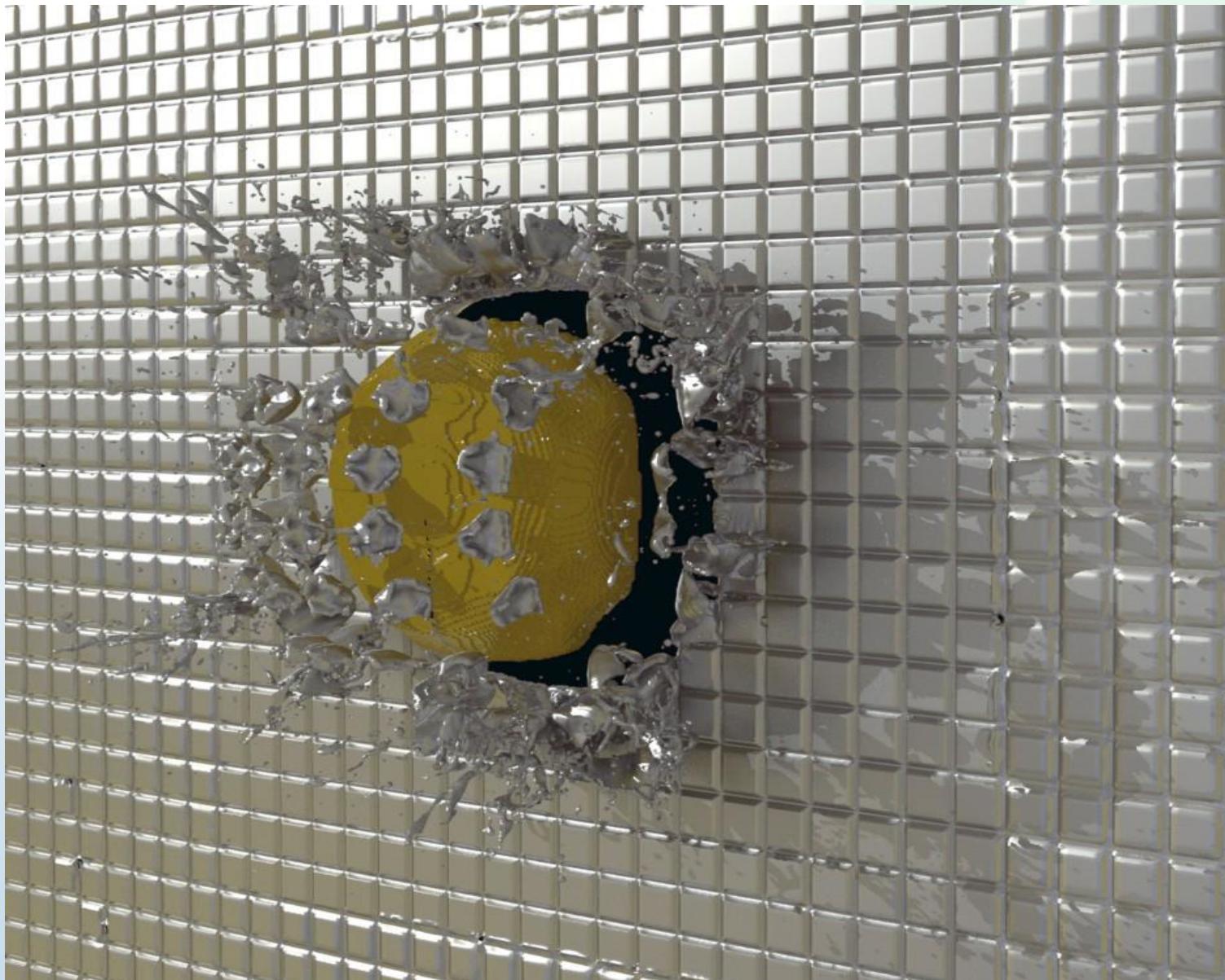


Online Help

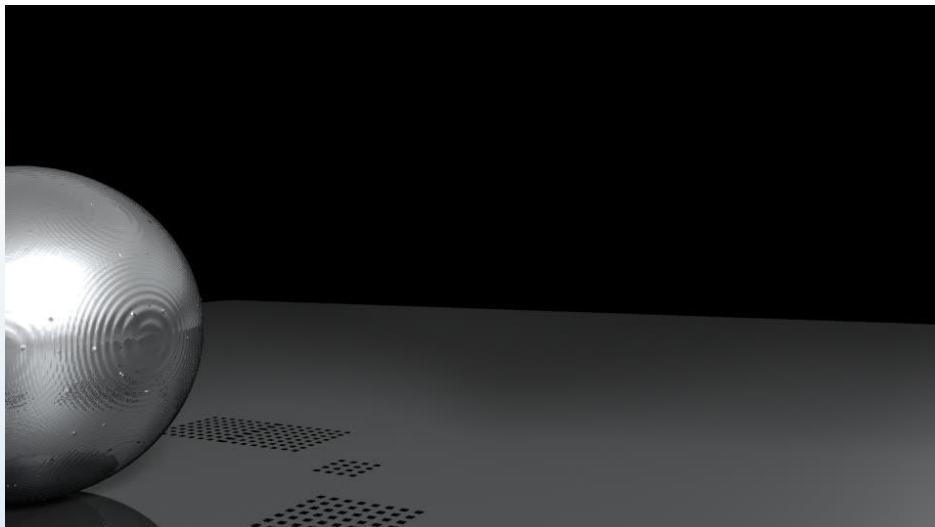
- Email list: paraview@paraview.org
- Doxygen:
 - <http://www.vtk.org/doc/nightly/html/classes.html>
 - <http://www.paraview.org/ParaView3/Doc/Nightly/html/classes.html>
- Sphinx:
 - <http://www.paraview.org/ParaView3/Doc/Nightly/www/py-doc/index.html>
- Websites:
 - <http://www.paraview.org>
 - <http://catalyst.paraview.org>
- Examples:
 - <https://github.com/acbauer/CatalystExampleCode>



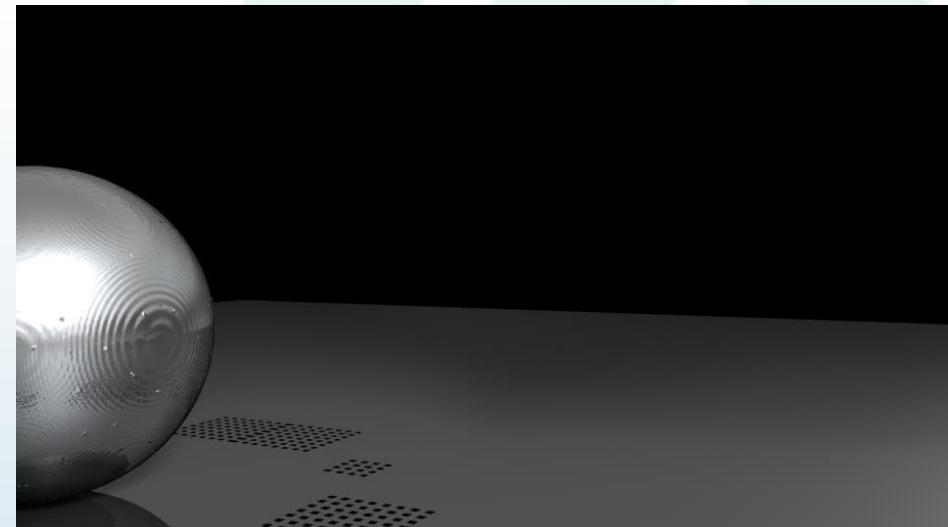
What is Catalyst



Access to More Data



Post-processing



In situ processing

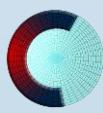
Roughly equal data stored at simulation time

Reflections and shadows added in post-processing for both examples

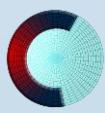
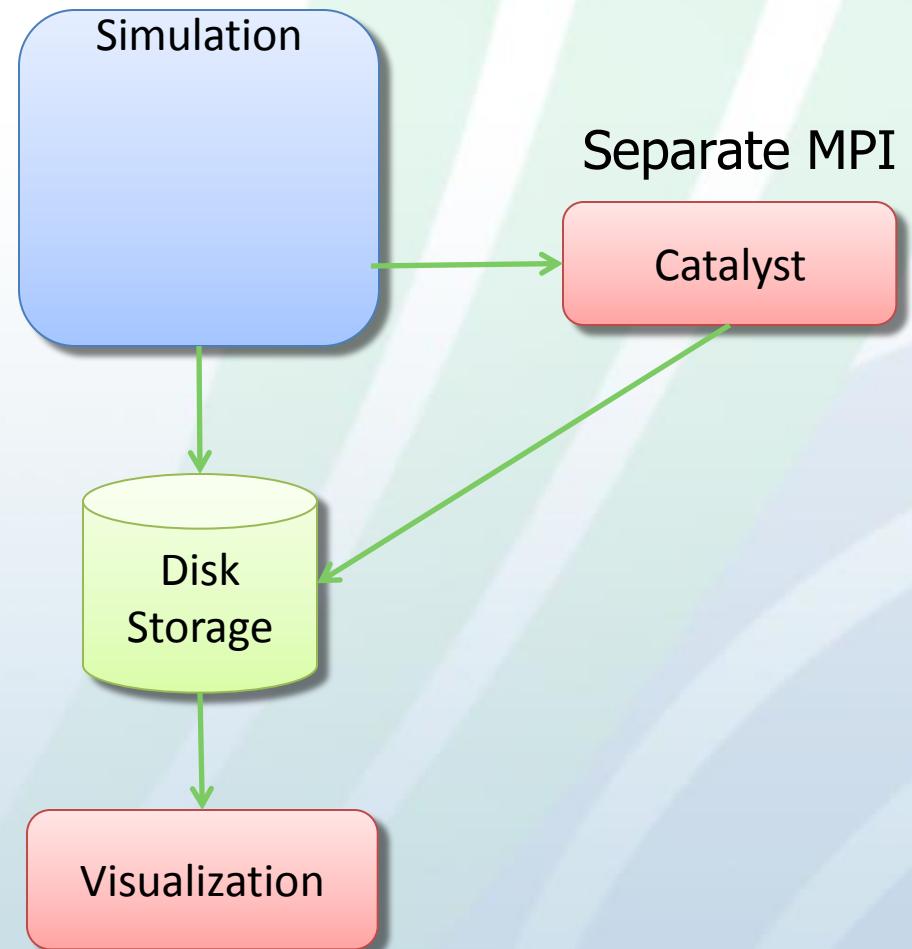
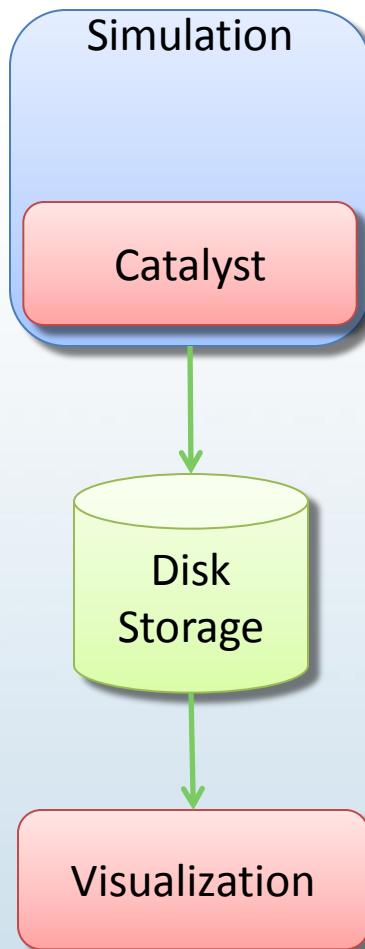
Why *In Situ*?

	2010	2018	Factor Change
System peak	2 Pf/s	1 Ef/s	500
Power	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 Gf/s	10 Tf/s	80
Node Memory BW	25 GB/s	400 GB/s	16
Node Concurrency	12 cpus	1,000 cpus	83
Interconnect BW	1.5 GB/s	50 GB/s	33
System size (nodes)	20 K nodes	1 M nodes	50
Total Concurrency	225 K	1 B	4,444
Storage	15 PB	300 PB	20
Input/Output Bandwidth	0.2 TB/s	20 TB/s	100

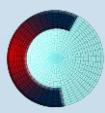
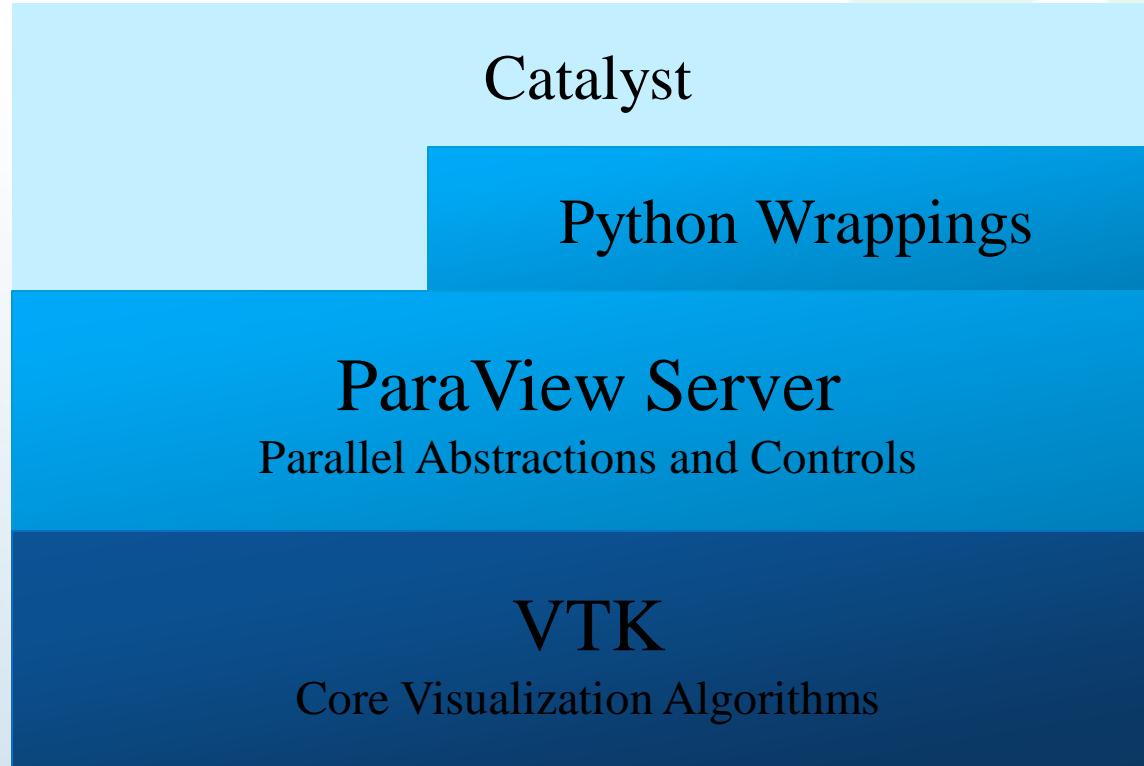
DOE Exascale Initiative Roadmap, Architecture and Technology Workshop, San Diego, December, 2009.



Two ways to run



Catalyst Architecture



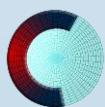
High Level View

Simulation Users

- Knowledge of ParaView as a post-processing/analysis tool
 - Basic interaction with GUI co-processing script generator plugin
 - Incremental knowledge increase to use the co-processing tools from basic ParaView use
- Programming knowledge can be useful to extend the tools

Simulation Developers

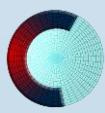
- Pass necessary simulation data to ParaView
- Need sufficient knowledge of both codes
 - VTK for grids and field data
 - ParaView Catalyst libraries
- Transparent to simulation users
- Extensible



User Perspective

Simulation

Catalyst

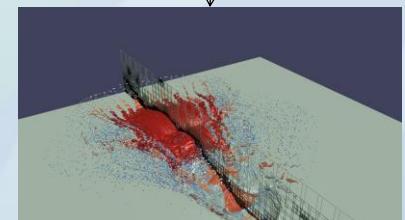


User Perspective

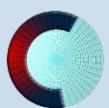
Simulation

Catalyst

Output
Processed
Data



Rendered Images



User Perspective

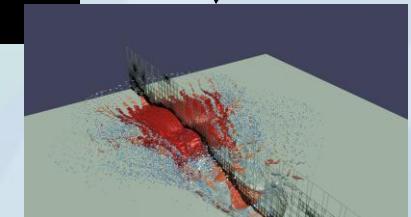
Simulation

Catalyst

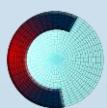
Output
Processed
Data



with Field Data

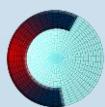
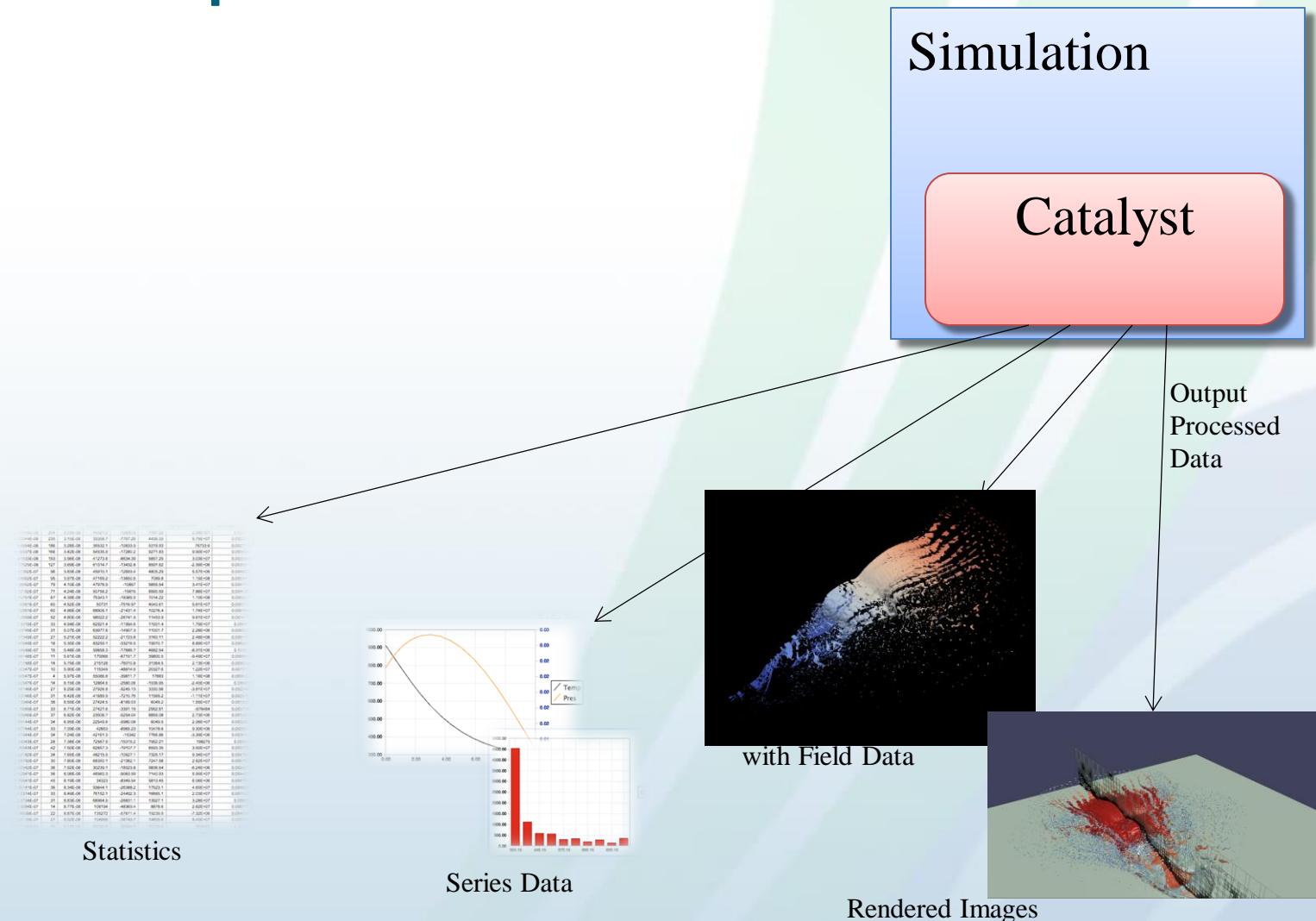


Rendered Images

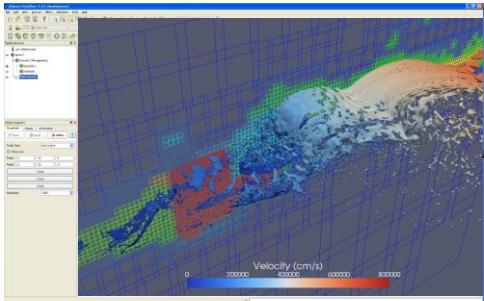


Kitware

User Perspective



User Perspective

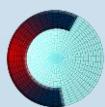
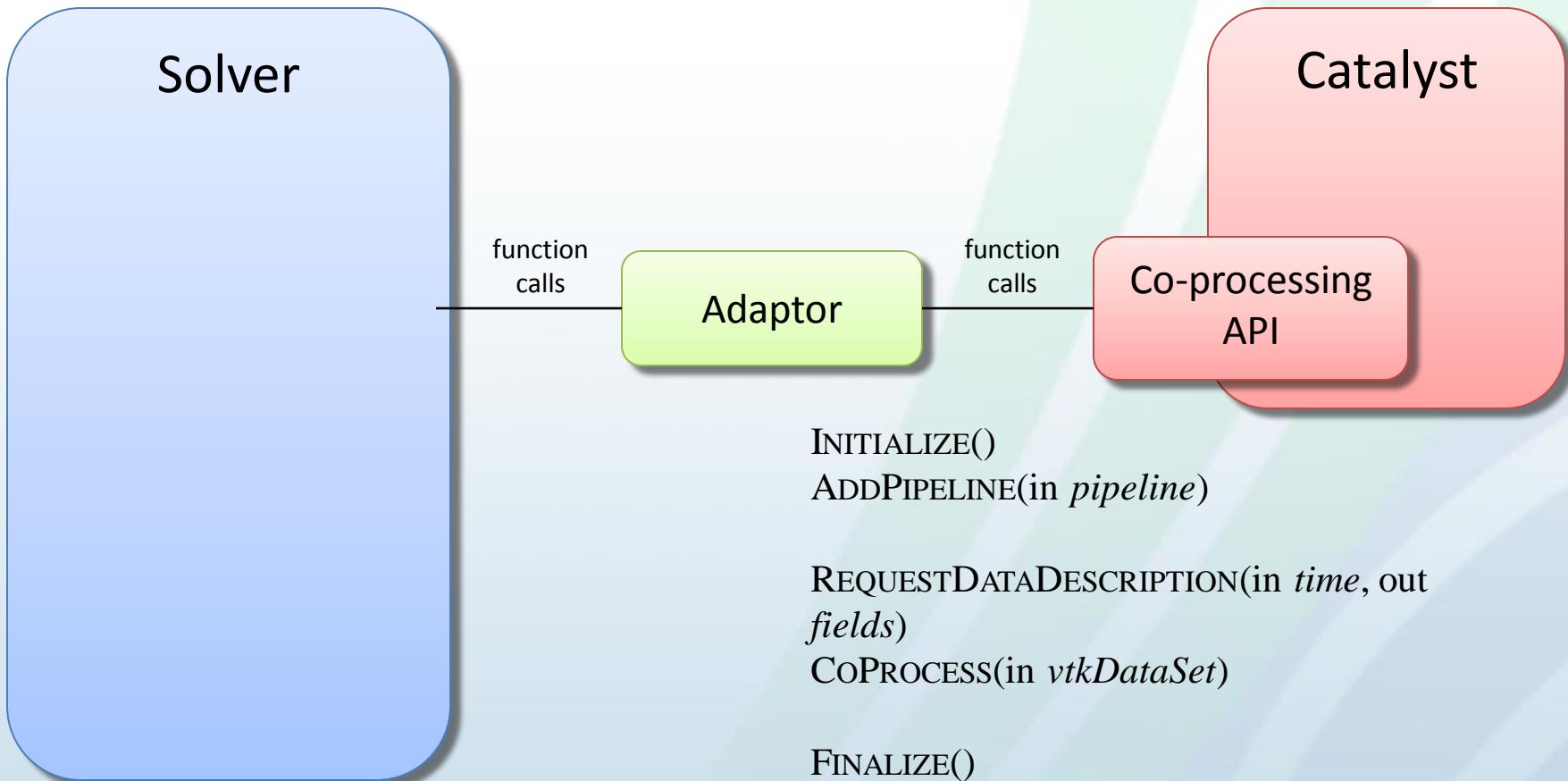


Script Export

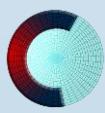
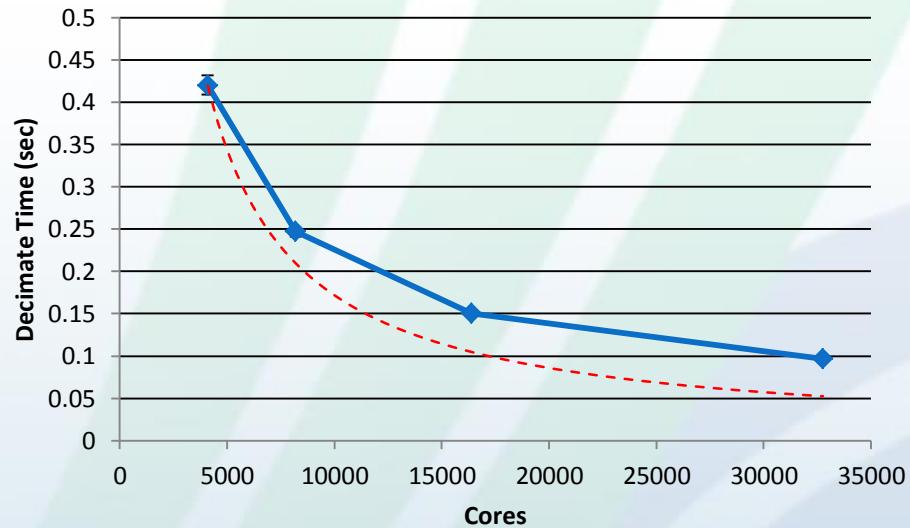
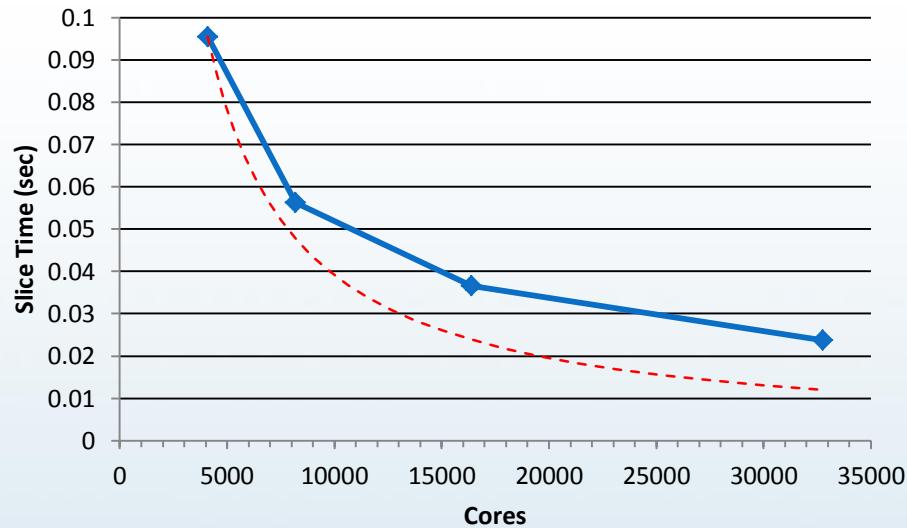
```
# Create the reader and set the filename.  
reader = servermanager.sources.Reader(FileNames=path)  
view = servermanager.CreateRenderView()  
repr = servermanager.CreateRepresentation(reader, view)  
reader.UpdatePipeline()  
dataInfo = reader.GetDataInformation()  
pInfo = dataInfo.GetPointDataInformation()  
arrayInfo = pInfo.GetArrayInformation("displacement9")  
if arrayInfo:  
    # get the range for the magnitude of displacement9  
    range = arrayInfo.GetComponentRange(-1)  
    lut = servermanager.rendering.PVLookupTable()  
    lut.RGBPoints = [range[0], 0.0, 0.0, 1.0,  
                    range[1], 1.0, 0.0, 0.0]  
    lut.VectorMode = "Magnitude"  
    repr.LookupTable = lut  
    repr.ColorArrayName = "displacement9"  
    repr.ColorAttributeType = "POINT_DATA"
```

```
111111.0 - 222.0 3.33E-09  
111111.0 - 188.0 3.29E-09  
111111.0 - 154.0 3.26E-09  
111111.0 - 120.0 3.23E-09  
111111.0 - 86.0 3.19E-09  
111111.0 - 52.0 3.16E-09  
111111.0 - 18.0 3.13E-09  
111111.0 - 14.0 3.10E-09  
111111.0 - 10.0 3.07E-09  
111111.0 - 6.0 3.04E-09  
111111.0 - 2.0 3.01E-09  
111111.0 - 1.0 2.98E-09  
111111.0 - 0.0 2.95E-09  
111111.0 - 1.0 2.92E-09  
111111.0 - 2.0 2.89E-09  
111111.0 - 6.0 2.86E-09  
111111.0 - 14.0 2.83E-09  
111111.0 - 22.0 2.80E-09  
111111.0 - 30.0 2.77E-09  
111111.0 - 38.0 2.74E-09  
111111.0 - 46.0 2.71E-09  
111111.0 - 54.0 2.68E-09  
111111.0 - 62.0 2.65E-09  
111111.0 - 70.0 2.62E-09  
111111.0 - 78.0 2.59E-09  
111111.0 - 86.0 2.56E-09  
111111.0 - 94.0 2.53E-09  
111111.0 - 102.0 2.50E-09  
111111.0 - 110.0 2.47E-09  
111111.0 - 118.0 2.44E-09  
111111.0 - 126.0 2.41E-09  
111111.0 - 134.0 2.38E-09  
111111.0 - 142.0 2.35E-09  
111111.0 - 150.0 2.32E-09  
111111.0 - 158.0 2.29E-09  
111111.0 - 166.0 2.26E-09  
111111.0 - 174.0 2.23E-09  
111111.0 - 182.0 2.20E-09  
111111.0 - 190.0 2.17E-09  
111111.0 - 198.0 2.14E-09  
111111.0 - 206.0 2.11E-09  
111111.0 - 214.0 2.08E-09  
111111.0 - 222.0 2.05E-09  
111111.0 - 230.0 2.02E-09  
111111.0 - 238.0 1.99E-09  
111111.0 - 246.0 1.96E-09  
111111.0 - 254.0 1.93E-09  
111111.0 - 262.0 1.90E-09  
111111.0 - 270.0 1.87E-09  
111111.0 - 278.0 1.84E-09  
111111.0 - 286.0 1.81E-09  
111111.0 - 294.0 1.78E-09  
111111.0 - 302.0 1.75E-09  
111111.0 - 310.0 1.72E-09  
111111.0 - 318.0 1.69E-09  
111111.0 - 326.0 1.66E-09  
111111.0 - 334.0 1.63E-09  
111111.0 - 342.0 1.60E-09  
111111.0 - 350.0 1.57E-09  
111111.0 - 358.0 1.54E-09  
111111.0 - 366.0 1.51E-09  
111111.0 - 374.0 1.48E-09  
111111.0 - 382.0 1.45E-09  
111111.0 - 390.0 1.42E-09  
111111.0 - 398.0 1.39E-09  
111111.0 - 406.0 1.36E-09  
111111.0 - 414.0 1.33E-09  
111111.0 - 422.0 1.30E-09  
111111.0 - 430.0 1.27E-09  
111111.0 - 438.0 1.24E-09  
111111.0 - 446.0 1.21E-09  
111111.0 - 454.0 1.18E-09  
111111.0 - 462.0 1.15E-09  
111111.0 - 470.0 1.12E-09  
111111.0 - 478.0 1.09E-09  
111111.0 - 486.0 1.06E-09  
111111.0 - 494.0 1.03E-09  
111111.0 - 502.0 1.00E-09  
111111.0 - 510.0 9.70E-10  
111111.0 - 518.0 9.40E-10  
111111.0 - 526.0 9.10E-10  
111111.0 - 534.0 8.80E-10  
111111.0 - 542.0 8.50E-10  
111111.0 - 550.0 8.20E-10  
111111.0 - 558.0 7.90E-10  
111111.0 - 566.0 7.60E-10  
111111.0 - 574.0 7.30E-10  
111111.0 - 582.0 7.00E-10  
111111.0 - 590.0 6.70E-10  
111111.0 - 598.0 6.40E-10  
111111.0 - 606.0 6.10E-10  
111111.0 - 614.0 5.80E-10  
111111.0 - 622.0 5.50E-10  
111111.0 - 630.0 5.20E-10  
111111.0 - 638.0 4.90E-10  
111111.0 - 646.0 4.60E-10  
111111.0 - 654.0 4.30E-10  
111111.0 - 662.0 4.00E-10  
111111.0 - 670.0 3.70E-10  
111111.0 - 678.0 3.40E-10  
111111.0 - 686.0 3.10E-10  
111111.0 - 694.0 2.80E-10  
111111.0 - 702.0 2.50E-10  
111111.0 - 710.0 2.20E-10  
111111.0 - 718.0 1.90E-10  
111111.0 - 726.0 1.60E-10  
111111.0 - 734.0 1.30E-10  
111111.0 - 742.0 1.00E-10  
111111.0 - 750.0 7.00E-11  
111111.0 - 758.0 4.00E-11  
111111.0 - 766.0 1.00E-11  
111111.0 - 774.0 1.00E-11  
111111.0 - 782.0 1.00E-11  
111111.0 - 790.0 1.00E-11  
111111.0 - 798.0 1.00E-11  
111111.0 - 806.0 1.00E-11  
111111.0 - 814.0 1.00E-11  
111111.0 - 822.0 1.00E-11  
111111.0 - 830.0 1.00E-11  
111111.0 - 838.0 1.00E-11  
111111.0 - 846.0 1.00E-11  
111111.0 - 854.0 1.00E-11  
111111.0 - 862.0 1.00E-11  
111111.0 - 870.0 1.00E-11  
111111.0 - 878.0 1.00E-11  
111111.0 - 886.0 1.00E-11  
111111.0 - 894.0 1.00E-11  
111111.0 - 902.0 1.00E-11  
111111.0 - 910.0 1.00E-11  
111111.0 - 918.0 1.00E-11  
111111.0 - 926.0 1.00E-11  
111111.0 - 934.0 1.00E-11  
111111.0 - 942.0 1.00E-11  
111111.0 - 950.0 1.00E-11  
111111.0 - 958.0 1.00E-11  
111111.0 - 966.0 1.00E-11  
111111.0 - 974.0 1.00E-11  
111111.0 - 982.0 1.00E-11  
111111.0 - 990.0 1.00E-11  
111111.0 - 998.0 1.00E-11  
111111.0 - 1006.0 1.00E-11  
111111.0 - 1014.0 1.00E-11  
111111.0 - 1022.0 1.00E-11  
111111.0 - 1030.0 1.00E-11  
111111.0 - 1038.0 1.00E-11  
111111.0 - 1046.0 1.00E-11  
111111.0 - 1054.0 1.00E-11  
111111.0 - 1062.0 1.00E-11  
111111.0 - 1070.0 1.00E-11  
111111.0 - 1078.0 1.00E-11  
111111.0 - 1086.0 1.00E-11  
111111.0 - 1094.0 1.00E-11  
111111.0 - 1102.0 1.00E-11  
111111.0 - 1110.0 1.00E-11  
111111.0 - 1118.0 1.00E-11  
111111.0 - 1126.0 1.00E-11  
111111.0 - 1134.0 1.00E-11  
111111.0 - 1142.0 1.00E-11  
111111.0 - 1150.0 1.00E-11  
111111.0 - 1158.0 1.00E-11  
111111.0 - 1166.0 1.00E-11  
111111.0 - 1174.0 1.00E-11  
111111.0 - 1182.0 1.00E-11  
111111.0 - 1190.0 1.00E-11  
111111.0 - 1198.0 1.00E-11  
111111.0 - 1206.0 1.00E-11  
111111.0 - 1214.0 1.00E-11  
111111.0 - 1222.0 1.00E-11  
111111.0 - 1230.0 1.00E-11  
111111.0 - 1238.0 1.00E-11  
111111.0 - 1246.0 1.00E-11  
111111.0 - 1254.0 1.00E-11  
111111.0 - 1262.0 1.00E-11  
111111.0 - 1270.0 1.00E-11  
111111.0 - 1278.0 1.00E-11  
111111.0 - 1286.0 1.00E-11  
111111.0 - 1294.0 1.00E-11  
111111.0 - 1302.0 1.00E-11  
111111.0 - 1310.0 1.00E-11  
111111.0 - 1318.0 1.00E-11  
111111.0 - 1326.0 1.00E-11  
111111.0 - 1334.0 1.00E-11  
111111.0 - 1342.0 1.00E-11  
111111.0 - 1350.0 1.00E-11  
111111.0 - 1358.0 1.00E-11  
111111.0 - 1366.0 1.00E-11  
111111.0 - 1374.0 1.00E-11  
111111.0 - 1382.0 1.00E-11  
111111.0 - 1390.0 1.00E-11  
111111.0 - 1398.0 1.00E-11  
111111.0 - 1406.0 1.00E-11  
111111.0 - 1414.0 1.00E-11  
111111.0 - 1422.0 1.00E-11  
111111.0 - 1430.0 1.00E-11  
111111.0 - 1438.0 1.00E-11  
111111.0 - 1446.0 1.00E-11  
111111.0 - 1454.0 1.00E-11  
111111.0 - 1462.0 1.00E-11  
111111.0 - 1470.0 1.00E-11  
111111.0 - 1478.0 1.00E-11  
111111.0 - 1486.0 1.00E-11  
111111.0 - 1494.0 1.00E-11  
111111.0 - 1502.0 1.00E-11  
111111.0 - 1510.0 1.00E-11  
111111.0 - 1518.0 1.00E-11  
111111.0 - 1526.0 1.00E-11  
111111.0 - 1534.0 1.00E-11  
111111.0 - 1542.0 1.00E-11  
111111.0 - 1550.0 1.00E-11  
111111.0 - 1558.0 1.00E-11  
111111.0 - 1566.0 1.00E-11  
111111.0 - 1574.0 1.00E-11  
111111.0 - 1582.0 1.00E-11  
111111.0 - 1590.0 1.00E-11  
111111.0 - 1598.0 1.00E-11  
111111.0 - 1606.0 1.00E-11  
111111.0 - 1614.0 1.00E-11  
111111.0 - 1622.0 1.00E-11  
111111.0 - 1630.0 1.00E-11  
111111.0 - 1638.0 1.00E-11  
111111.0 - 1646.0 1.00E-11  
111111.0 - 1654.0 1.00E-11  
111111.0 - 1662.0 1.00E-11  
111111.0 - 1670.0 1.00E-11  
111111.0 - 1678.0 1.00E-11  
111111.0 - 1686.0 1.00E-11  
111111.0 - 1694.0 1.00E-11  
111111.0 - 1702.0 1.00E-11  
111111.0 - 1710.0 1.00E-11  
111111.0 - 1718.0 1.00E-11  
111111.0 - 1726.0 1.00E-11  
111111.0 - 1734.0 1.00E-11  
111111.0 - 1742.0 1.00E-11  
111111.0 - 1750.0 1.00E-11  
111111.0 - 1758.0 1.00E-11  
111111.0 - 1766.0 1.00E-11  
111111.0 - 1774.0 1.00E-11  
111111.0 - 1782.0 1.00E-11  
111111.0 - 1790.0 1.00E-11  
111111.0 - 1798.0 1.00E-11  
111111.0 - 1806.0 1.00E-11  
111111.0 - 1814.0 1.00E-11  
111111.0 - 1822.0 1.00E-11  
111111.0 - 1830.0 1.00E-11  
111111.0 - 1838.0 1.00E-11  
111111.0 - 1846.0 1.00E-11  
111111.0 - 1854.0 1.00E-11  
111111.0 - 1862.0 1.00E-11  
111111.0 - 1870.0 1.00E-11  
111111.0 - 1878.0 1.00E-11  
111111.0 - 1886.0 1.00E-11  
111111.0 - 1894.0 1.00E-11  
111111.0 - 1902.0 1.00E-11  
111111.0 - 1910.0 1.00E-11  
111111.0 - 1918.0 1.00E-11  
111111.0 - 1926.0 1.00E-11  
111111.0 - 1934.0 1.00E-11  
111111.0 - 1942.0 1.00E-11  
111111.0 - 1950.0 1.00E-11  
111111.0 - 1958.0 1.00E-11  
111111.0 - 1966.0 1.00E-11  
111111.0 - 1974.0 1.00E-11  
111111.0 - 1982.0 1.00E-11  
111111.0 - 1990.0 1.00E-11  
111111.0 - 1998.0 1.00E-11  
111111.0 - 2006.0 1.00E-11  
111111.0 - 2014.0 1.00E-11  
111111.0 - 2022.0 1.00E-11  
111111.0 - 2030.0 1.00E-11  
111111.0 - 2038.0 1.00E-11  
111111.0 - 2046.0 1.00E-11  
111111.0 - 2054.0 1.00E-11  
111111.0 - 2062.0 1.00E-11  
111111.0 - 2070.0 1.00E-11  
111111.0 - 2078.0 1.00E-11  
111111.0 - 2086.0 1.00E-11  
111111.0 - 2094.0 1.00E-11  
111111.0 - 2102.0 1.00E-11  
111111.0 - 2110.0 1.00E-11  
111111.0 - 2118.0 1.00E-11  
111111.0 - 2126.0 1.00E-11  
111111.0 - 2134.0 1.00E-11  
111111.0 - 2142.0 1.00E-11  
111111.0 - 2150.0 1.00E-11  
111111.0 - 2158.0 1.00E-11  
111111.0 - 2166.0 1.00E-11  
111111.0 - 2174.0 1.00E-11  
111111.0 - 2182.0 1.00E-11  
111111.0 - 2190.0 1.00E-11  
111111.0 - 2198.0 1.00E-11  
111111.0 - 2206.0 1.00E-11  
111111.0 - 2214.0 1.00E-11  
111111.0 - 2222.0 1.00E-11  
111111.0 - 2230.0 1.00E-11  
111111.0 - 2238.0 1.00E-11  
111111.0 - 2246.0 1.00E-11  
111111.0 - 2254.0 1.00E-11  
111111.0 - 2262.0 1.00E-11  
111111.0 - 2270.0 1.00E-11  
111111.0 - 2278.0 1.00E-11  
111111.0 - 2286.0 1.00E-11  
111111.0 - 2294.0 1.00E-11  
111111.0 - 2302.0 1.00E-11  
111111.0 - 2310.0 1.00E-11  
111111.0 - 2318.0 1.00E-11  
111111.0 - 2326.0 1.00E-11  
111111.0 - 2334.0 1.00E-11  
111111.0 - 2342.0 1.00E-11  
111111.0 - 2350.0 1.00E-11  
111111.0 - 2358.0 1.00E-11  
111111.0 - 2366.0 1.00E-11  
111111.0 - 2374.0 1.00E-11  
111111.0 - 2382.0 1.00E-11  
111111.0 - 2390.0 1.00E-11  
111111.0 - 2398.0 1.00E-11  
111111.0 - 2406.0 1.00E-11  
111111.0 - 2414.0 1.00E-11  
111111.0 - 2422.0 1.00E-11  
111111.0 - 2430.0 1.00E-11  
111111.0 - 2438.0 1.00E-11  
111111.0 - 2446.0 1.00E-11  
111111.0 - 2454.0 1.00E-11  
111111.0 - 2462.0 1.00E-11  
111111.0 - 2470.0 1.00E-11  
111111.0 - 2478.0 1.00E-11  
111111.0 - 2486.0 1.00E-11  
111111.0 - 2494.0 1.00E-11  
111111.0 - 2502.0 1.00E-11  
111111.0 - 2510.0 1.00E-11  
111111.0 - 2518.0 1.00E-11  
111111.0 - 2526.0 1.00E-11  
111111.0 - 2534.0 1.00E-11  
111111.0 - 2542.0 1.00E-11  
111111.0 - 2550.0 1.00E-11  
111111.0 - 2558.0 1.00E-11  
111111.0 - 2566.0 1.00E-11  
111111.0 - 2574.0 1.00E-11  
111111.0 - 2582.0 1.00E-11  
111111.0 - 2590.0 1.00E-11  
111111.0 - 2598.0 1.00E-11  
111111.0 - 2606.0 1.00E-11  
111111.0 - 2614.0 1.00E-11  
111111.0 - 2622.0 1.00E-11  
111111.0 - 2630.0 1.00E-11  
111111.0 - 2638.0 1.00E-11  
111111.0 - 2646.0 1.00E-11  
111111.0 - 2654.0 1.00E-11  
111111.0 - 2662.0 1.00E-11  
111111.0 - 2670.0 1.00E-11  
111111.0 - 2678.0 1.00E-11  
111111.0 - 2686.0 1.00E-11  
111111.0 - 2694.0 1.00E-11  
111111.0 - 2702.0 1.00E-11  
111111.0 - 2710.0 1.00E-11  
111111.0 - 2718.0 1.00E-11  
111111.0 - 2726.0 1.00E-11  
111111.0 - 2734.0 1.00E-11  
111111.0 - 2742.0 1.00E-11  
111111.0 - 2750.0 1.00E-11  
111111.0 - 2758.0 1.00E-11  
111111.0 - 2766.0 1.00E-11  
111111.0 - 2774.0 1.00E-11  
111111.0 - 2782.0 1.00E-11  
111111.0 - 2790.0 1.00E-11  
111111.0 - 2798.0 1.00E-11  
111111.0 - 2806.0 1.00E-11  
111111.0 - 2814.0 1.00E-11  
111111.0 - 2822.0 1.00E-11  
111111.0 - 2830.0 1.00E-11  
111111.0 - 2838.0 1.00E-11  
111111.0 - 2846.0 1.00E-11  
111111.0 - 2854.0 1.00E-11  
111111.0 - 2862.0 1.00E-11  
111111.0 - 2870.0 1.00E-11  
111111.0 - 2878.0 1.00E-11  
111111.0 - 2886.0 1.00E-11  
111111.0 - 2894.0 1.00E-11  
111111.0 - 2902.0 1.00E-11  
111111.0 - 2910.0 1.00E-11  
111111.0 - 2918.0 1.00E-11  
111111.0 - 2926.0 1.00E-11  
111111.0 - 2934.0 1.00E-11  
111111.0 - 2942.0 1.00E-11  
111111.0 - 2950.0 1.00E-11  
111111.0 - 2958.0 1.00E-11  
111111.0 - 2966.0 1.00E-11  
111111.0 - 2974.0 1.00E-11  
111111.0 - 2982.0 1.00E-11  
111111.0 - 2990.0 1.00E-11  
111111.0 - 2998.0 1.00E-11  
111111.0 - 3006.0 1.00E-11  
111111.0 - 3014.0 1.00E-11  
111111.0 - 3022.0 1.00E-11  
111111.0 - 3030.0 1.00E-11  
111111.0 - 3038.0 1.00E-11  
111111.0 - 3046.0 1.00E-11  
111111.0 - 3054.0 1.00E-11  
111111.0 - 3062.0 1.00E-11  
111111.0 - 3070.0 1.00E-11  
111111.0 - 3078.0 1.00E-11  
111111.0 - 3086.0 1.00E-11  
111111.0 - 3094.0 1.00E-11  
111111.0 - 3102.0 1.00E-11  
111111.0 - 3110.0 1.00E-11  
111111.0 - 3118.0 1.00E-11  
111111.0 - 3126.0 1.00E-11  
111111.0 - 3134.0 1.00E-11  
111111.0 - 3142.0 1.00E-11  
111111.0 - 3150.0 1.00E-11  
111111.0 - 3158.0 1.00E-11  
111111.0 - 3166.0 1.00E-11  
111111.0 - 3174.0 1.00E-11  
111111.0 - 3182.0 1.00E-11  
111111.0 - 3190.0 1.00E-11  
111111.0 - 3198.0 1.00E-11  
111111.0 - 3206.0 1.00E-11  
111111.0 - 3214.0 1.00E-11  
111111.0 - 3222.0 1.00E-11  
111111.0 - 3230.0 1.00E-11  
111111.0 - 3238.0 1.00E-11  
111111.0 - 3246.0 1.00E-11  
111111.0 - 3254.0 1.00E-11  
111111.0 - 3262.0 1.00E-11  
111111.0 - 3270.0 1.00E-11  
111111.0 - 3278.0 1.00E-11  
111111.0 - 3286.0 1.00E-11  
111111.0 - 3294.0 1.00E-11  
111111.0 - 3302.0 1.00E-11  
111111.0 - 3310.0 1.00E-11  
111111.0 - 3318.0 1.00E-11  
111111.0 - 3326.0 1.00E-11  
111111.0 - 3334.0 1.00E-11  
111111.0 - 3342.0 1.00E-11  
111111.0 - 3350.0 1.00E-11  
111111.0 - 3358.0 1.00E-11  
111111.0 - 3366.0 1.00E-11  
111111.0 - 3374.0 1.00E-11  
111111.0 - 3382.0 1.00E-11  
111111.0 - 3390.0 1.00E-11  
111111.0 - 3398.0 1.00E-11  
111111.0 - 3406.0 1.00E-11  
111111.0 - 3414.0 1.00E-11  
111111.0 - 3422.0 1.00E
```

Developer Perspective



Developer Perspective





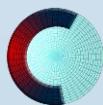
ParaView Catalyst for Simulation Users



Creating Catalyst Output

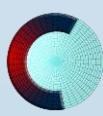
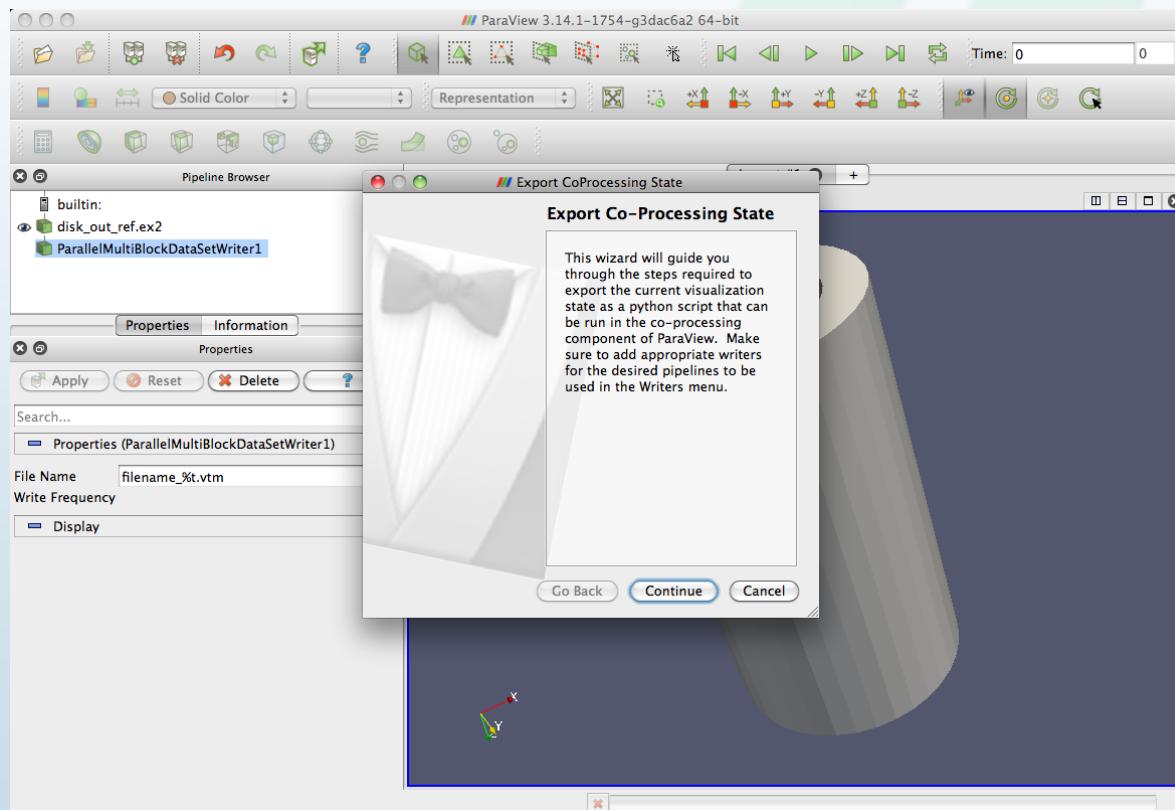
Two main ways:

- Create and/or modify Python scripts
 - ParaView GUI plugin to create Python scripts
 - Modification with knowledge of ParaView Python API
- Developer generated “canned” scripts
 - User provides parameters for already created Catalyst pipelines
 - User may not even need to know ParaView
 - See ParaView Catalyst User’s Guide



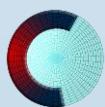
Create Python Scripts from ParaView

- Interact with ParaView normally
- Export a script that mimics that interaction
- Queries during each co-processing step
 - (one frame at a time)



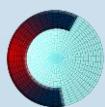
ParaView GUI Plugin

- Similar to using ParaView interactively
 - Setup desired pipelines
 - Ideally, start with a representative data set from the simulation
- Extra pipeline information to tell what to output during simulation run
 - Add in data extract writers
 - Create screenshots to output
 - Both require file name and write frequency



In Situ Demo

- Create a ParaView Catalyst Python pipeline script
 - Specify desired outputs from run
 - Export the script
- Run the script with a fictitious input
 - Time dependent grid and field data come from a file instead of from an actual simulation
- Examine results



In Situ Demo – Build Step

CMake 2.8.2 – /Users/ndfabia/Desktop/Work/ParaView_build

Where is the source code: /Users/ndfabia/Desktop/Work/ParaView

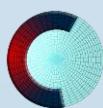
Where to build the binaries: /Users/ndfabia/Desktop/Work/ParaView_build

Search: Grouped Advanced

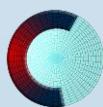
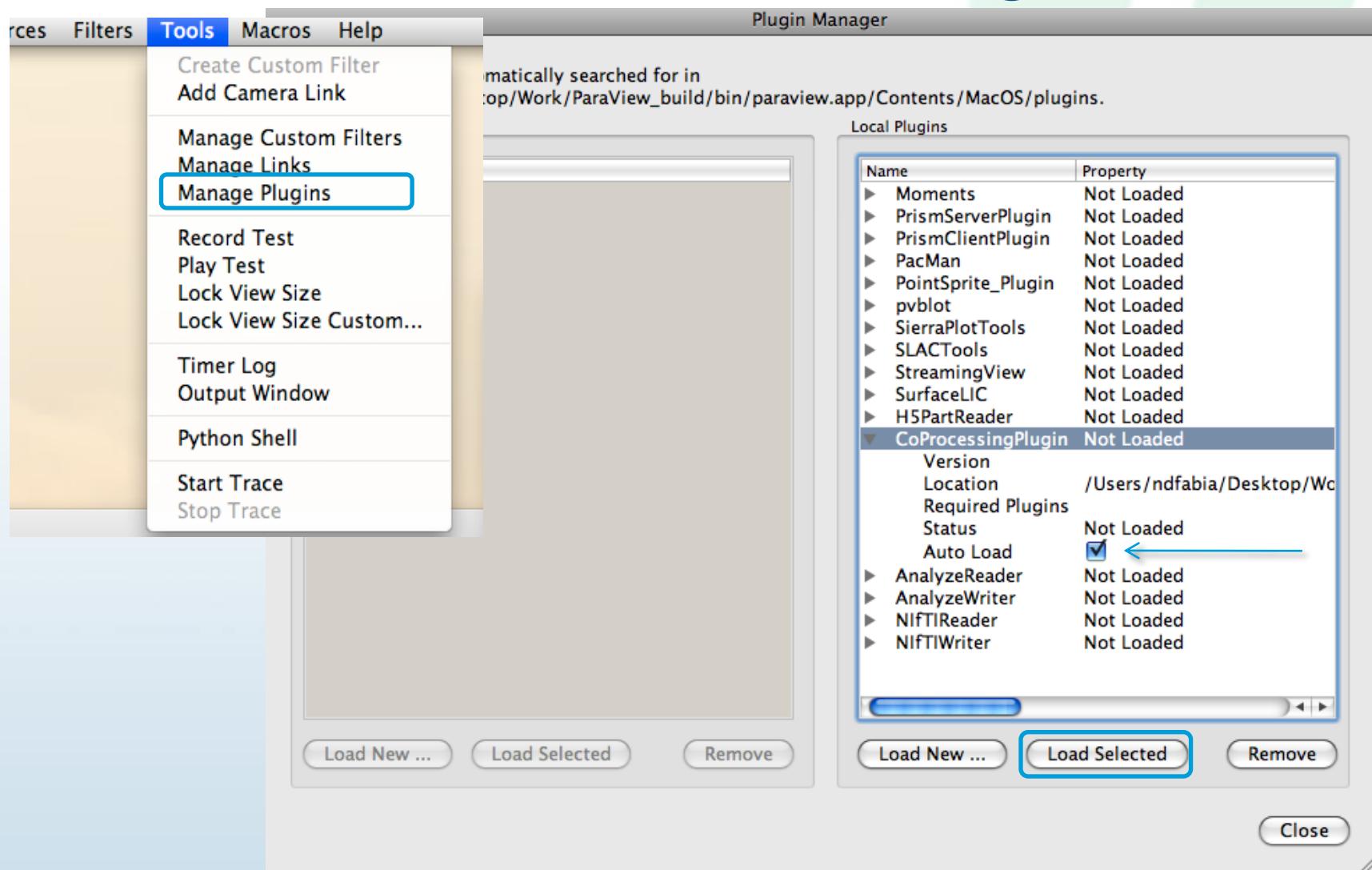
Name	Value
BUILD_COPROCESSING_ADAPTOR	<input checked="" type="checkbox"/>
BUILD_FORTRAN_COPROCESSING_ADAPTOR	<input checked="" type="checkbox"/>
BUILD_PARTICLE_COPROCESSING_ADAPTOR	<input type="checkbox"/>
PARAVIEW_BUILD_PLUGIN_CoProcessingScriptGenerator	<input checked="" type="checkbox"/>
PARAVIEW_ENABLE_COPROCESSING	<input checked="" type="checkbox"/>

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles



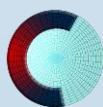
In Situ Demo – Load Plugin Step



In Situ Demo – New Plugin Menus

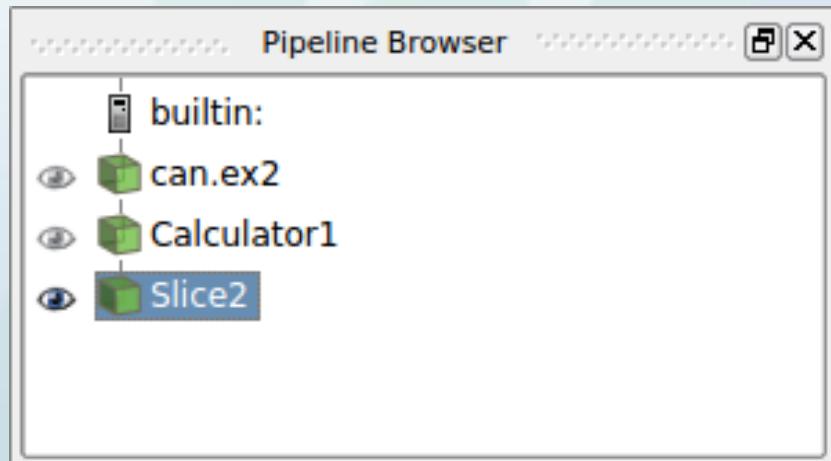
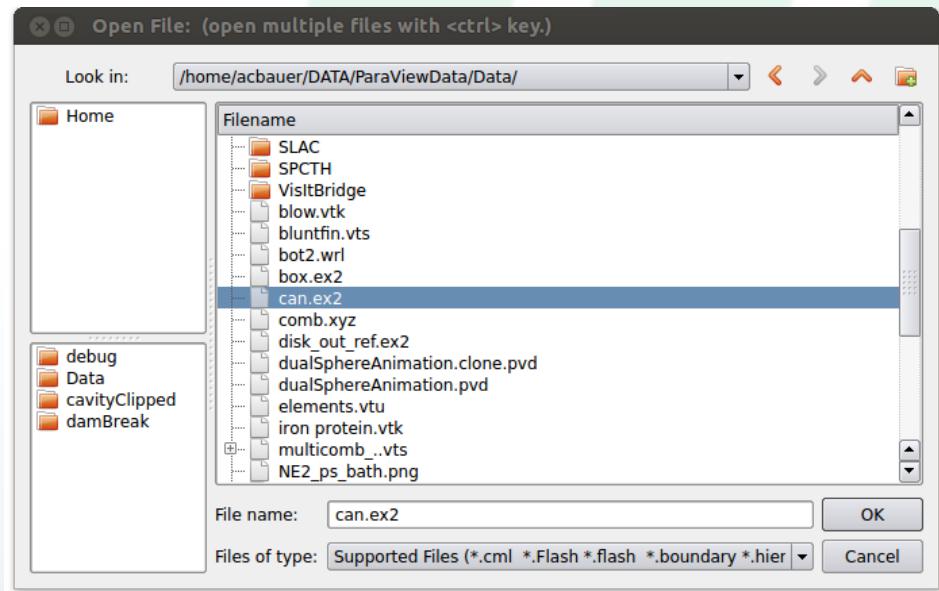
CoProcessir Writers

Export State	Parallel Hierarchical Box Data Writer Parallel MultiBlockDataSet Writer Parallel Image Data Writer Parallel PolyData Writer Parallel Rectilinear Grid Writer Parallel Structured Grid Writer Parallel Unstructured Grid Writer
---------------------	---



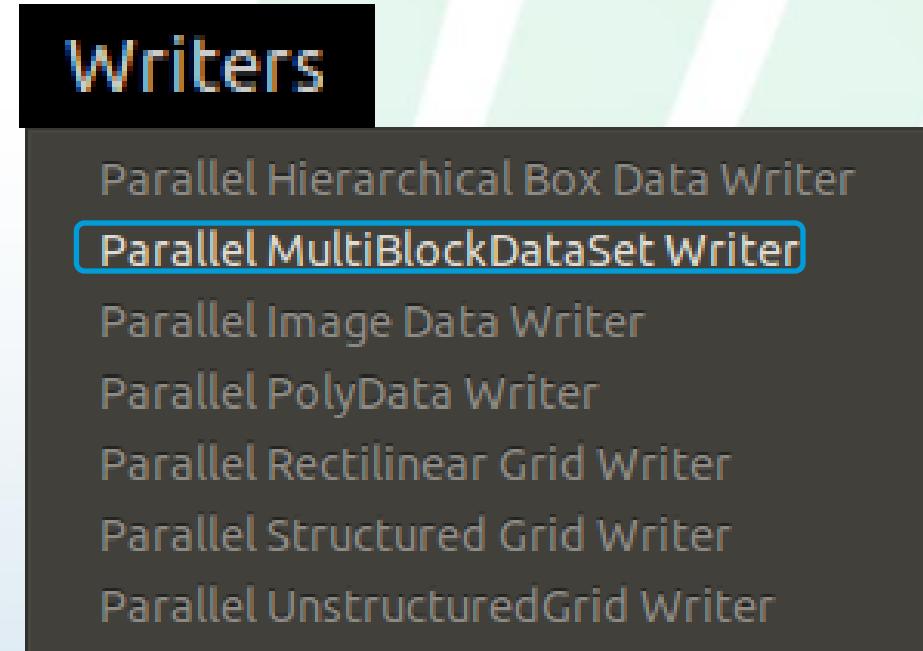
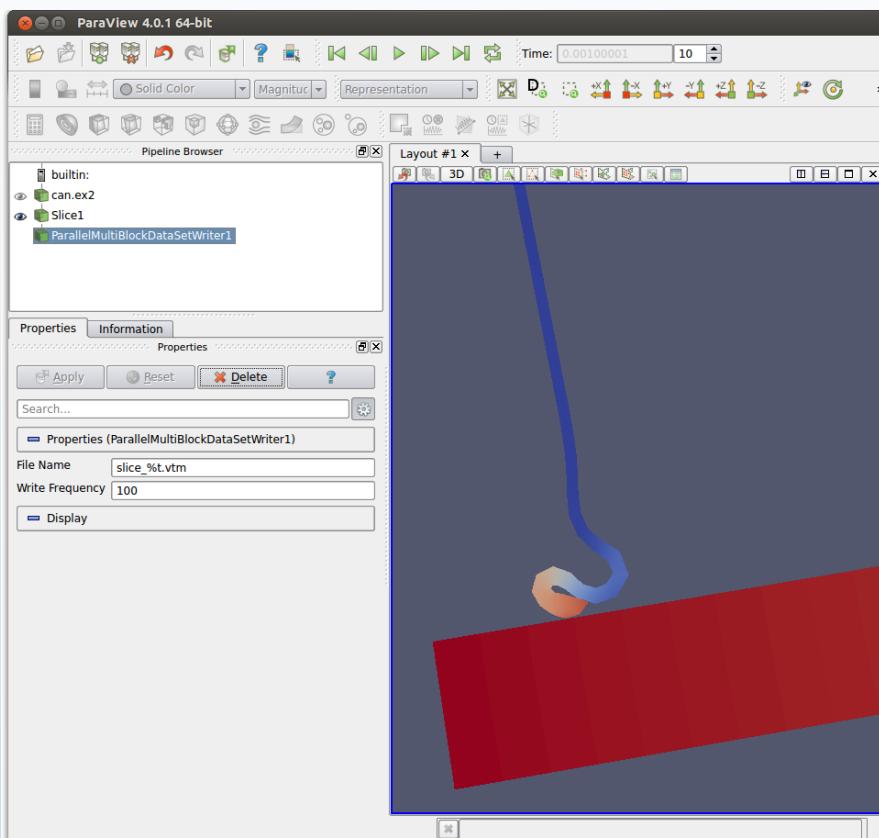
In Situ Demo – Creating a Catalyst Python Script

- Load can.ex2
 - Note that there are 44 time steps
- Create desired pipeline

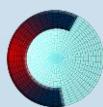
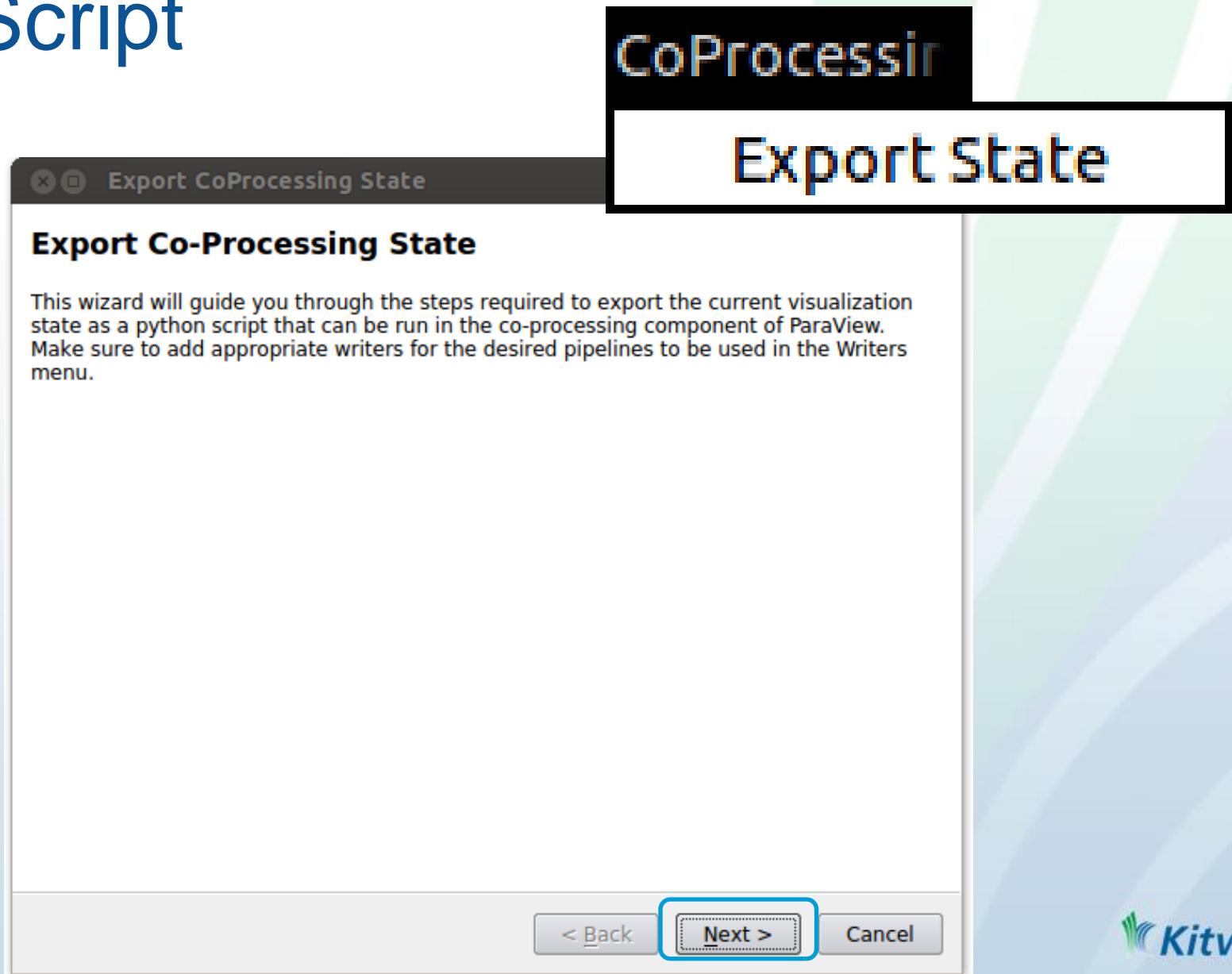


In Situ Demo – Adding in Writers

- Parameters:
 - File Name – %t gets replaced with time step
 - Write Frequency

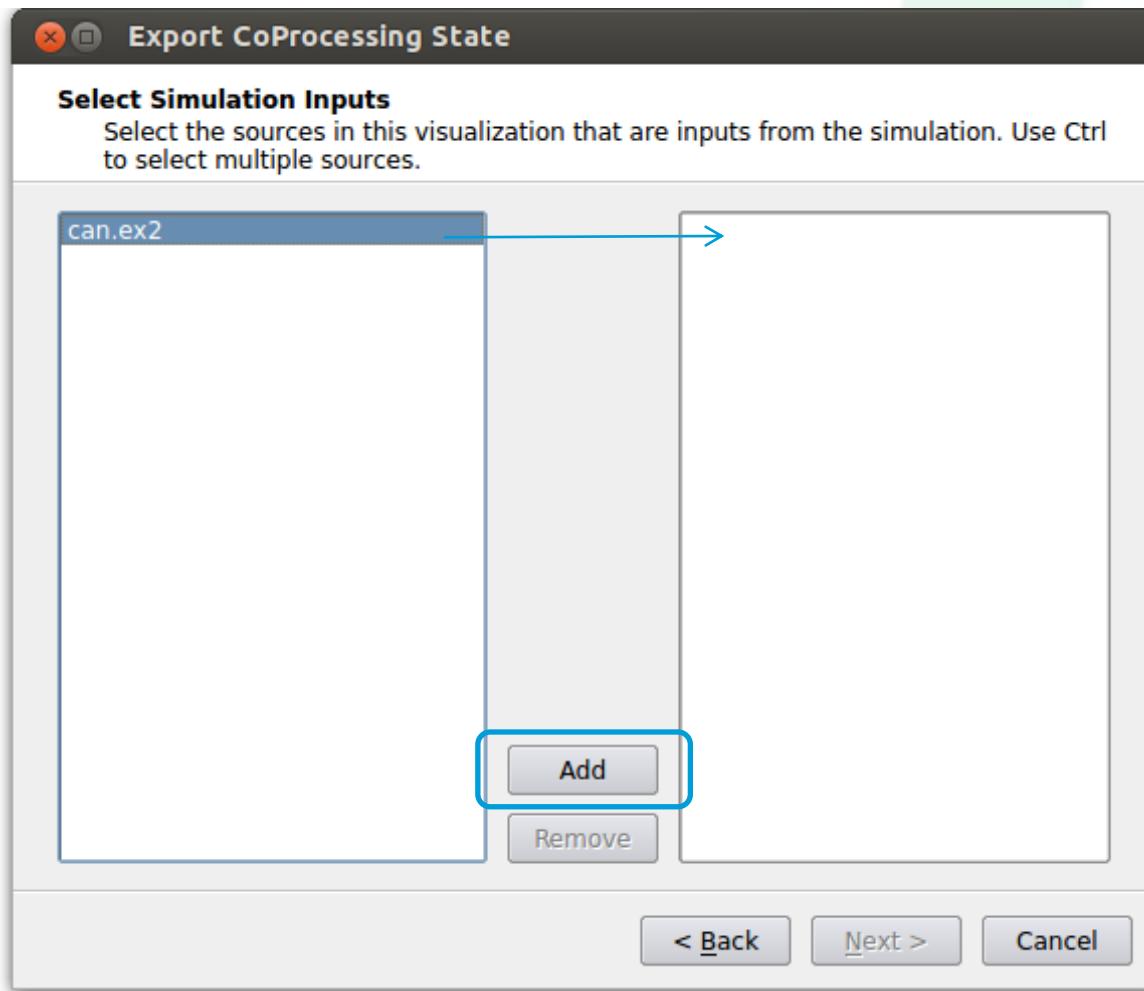


In Situ Demo – Exporting the Script



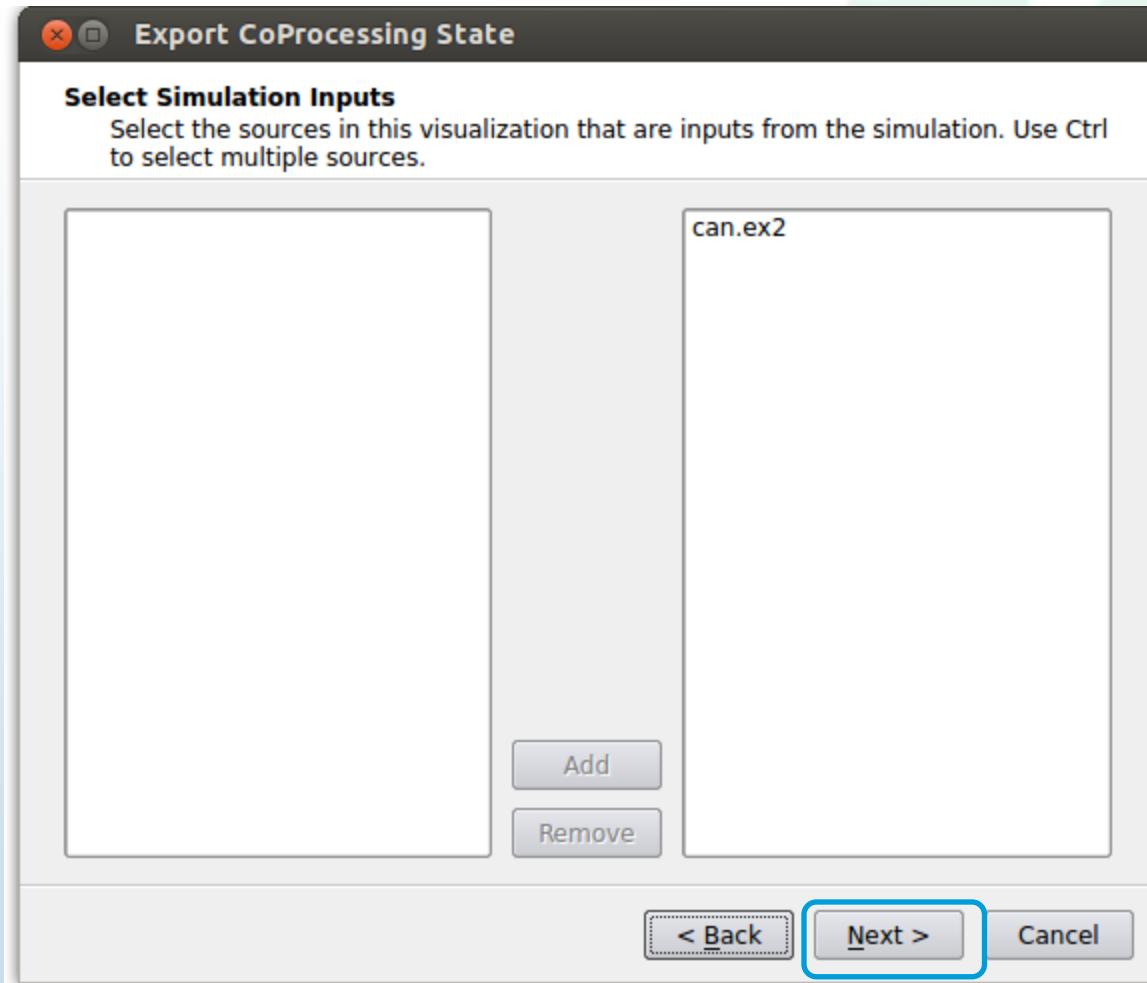
In Situ Demo – Select Inputs

- Usually only a single input but can have multiple inputs



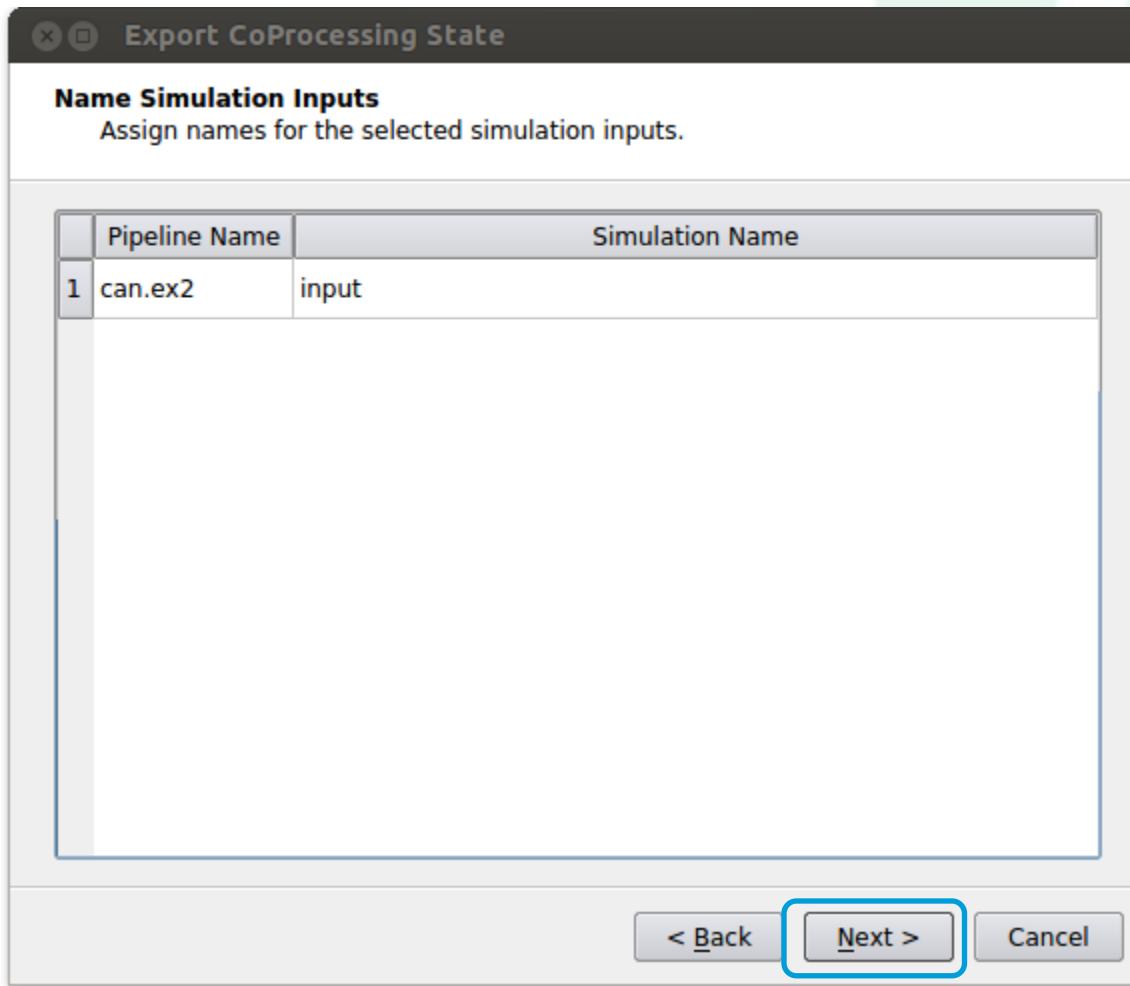
In Situ Demo – Select Inputs

- Each pipeline source is a potential input



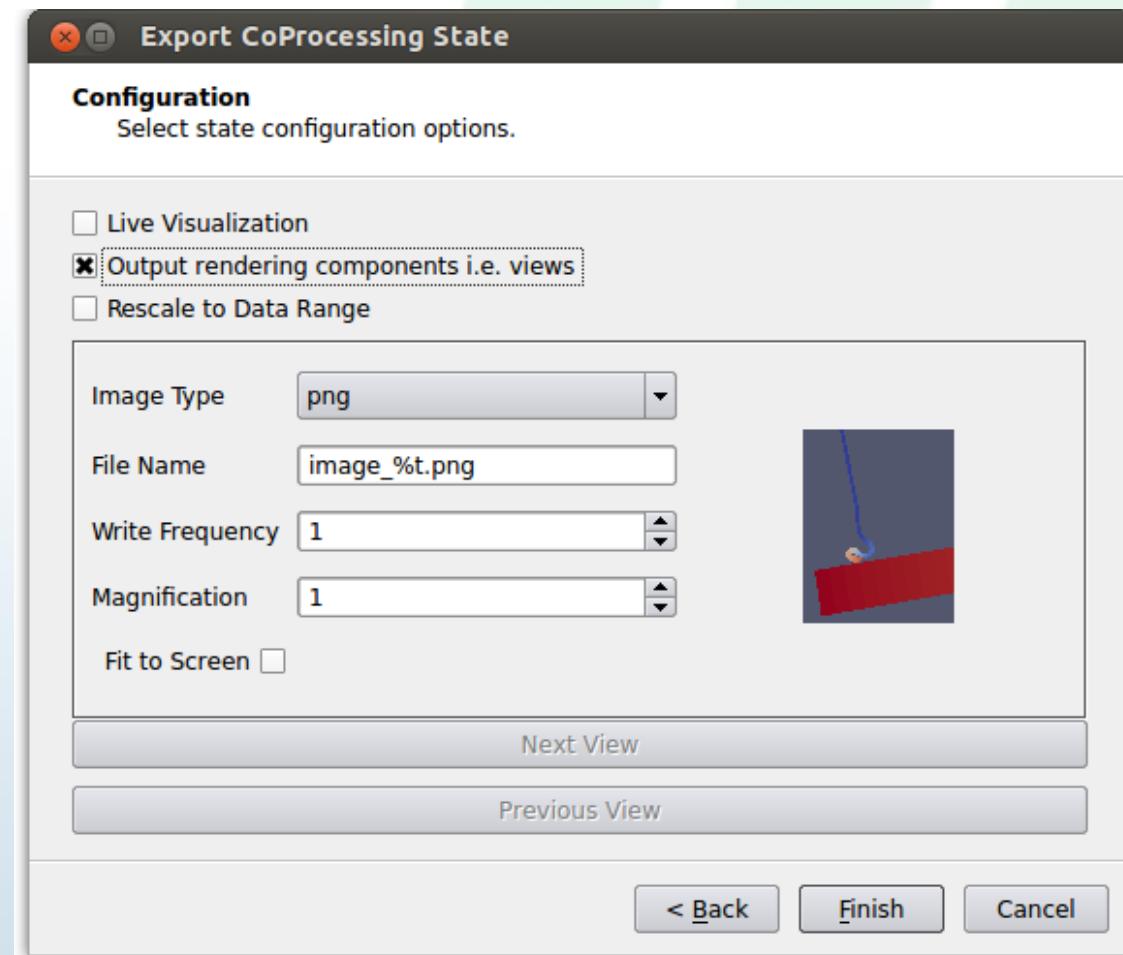
In Situ Demo – Match Up Inputs

- Source name (e.g. “can.ex2”) needs to be matched with string key in adaptor (e.g. “input”)



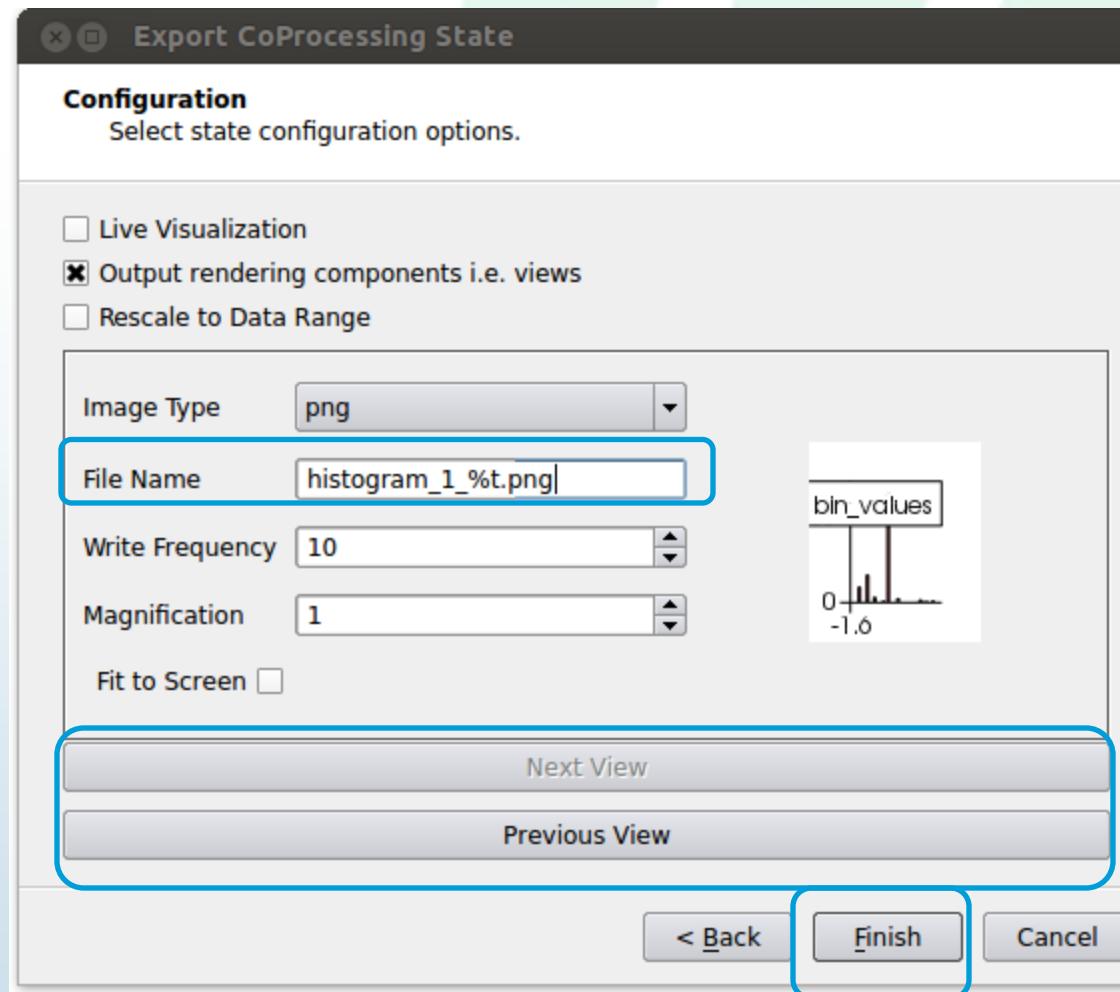
In Situ Demo – Generating an Image

- Parameters/Options:
 - Live visualization
 - Rescale to Data Range (all images)
 - Individual images
 - Image Type
 - File Name
 - Write Frequency
 - Magnification
 - Fit to Screen
- $\%t$ gets replaced with time step



In Situ Demo – Generating Two Images

- Parameters/Options:
 - Live visualization
 - Rescale to Data Range (all images)
 - Individual images
 - Image Type
 - File Name
 - Write Frequency
 - Magnification
 - Fit to Screen



In Situ Demo – Write Out the Script

- Generated script will look something like this

```
def DoCoProcessing (datadescription)

    input = CreateProducer ( datadescription "input"

    ParallelMultiBlockDataSetWriter1 = CreateWriter (
        XMLMultiBlockDataWriter "filename_%t.vtm" 1
```

```
try: paraview.simple
except: from paraview.simple import *

cp_writers = []

def RequestDataDescription(datadescription):
    "Callback to populate the request for current timestep"
    timestep = datadescription.GetTimeStep()

    input_name = 'input'
    if (timestep % 1 == 0):
        datadescription.GetInputDescriptionByName(input_name).AllFieldsOn()
        datadescription.GetInputDescriptionByName(input_name).GenerateMeshOn()
    else:
        datadescription.GetInputDescriptionByName(input_name).AllFieldsOff()
        datadescription.GetInputDescriptionByName(input_name).GenerateMeshOff()

def DoCoProcessing(datadescription):
    "Callback to do co-processing for current timestep"
    global cp_writers
    cp_writers = []
    timestep = datadescription.GetTimeStep()

    input = CreateProducer( datadescription, "input" )

    ParallelMultiBlockDataSetWriter1 = CreateWriter( XMLMultiBlockDataWriter, "filename_xt.vtm", 1 )

    for writer in cp_writers:
        if timestep % writer.cpFrequency == 0:
            writer.FileName = writer.cpFileName.replace("%t", str(timestep))
            ine()

            nager.GetRenderViews()

            renderviews):
            ome.replace("%w", str(view))
            ace("%t", str(timestep))
            renderviews[view])

    oxies -- we do it this way to avoid problems with prototypes
Delete()
:
esToDelete()

iter = servermanager.vtkSMProxyIterator()
iter.Begin()
tobedeleted = []
while not iter.IsAtEnd():
    if iter.GetGroup().find("prototypes") != -1:
        iter.Next()
        continue
    proxy = servermanager._getPyProxy(iter.GetProxy())
    proxygroup = iter.GetGroup()
    iter.Next()
    if proxygroup != 'timekeeper' and proxy != None and proxygroup.find("pq_helper_proxies") == -1:
        tobedeleted.append(proxy)

    return tobedeleted

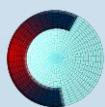
def CreateProducer(datadescription, gridname):
    "Create a producer proxy for the grid"
    if not datadescription.GetInputDescriptionByName(gridname):
        raise RuntimeError, "Simulation input name '%s' does not exist" % gridname
    grid = datadescription.GetInputDescriptionByName(gridname).GetGrid()
    producer = TrivialProducer()
    producer.GetClientSideObject().SetOutput(grid)
    producer.UpdatePipeline()
    return producer

def CreateWriter(proxy_ctor, filename, freq):
    global cp_writers
    writer = proxy_ctor()
    writer.FileName = filename
    writer.add_attribute("cpFrequency", freq)
    writer.add_attribute("cpFileName", filename)
    cp_writers.append(writer)
    return writer
```



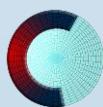
In Situ Demo – Run the Script

- Put candriver.py and the generated Python script in the same directory
- Linux and Mac from a terminal
 - <path>/pvpython candriver.py <generated script> <path>/can.ex2
- Windows from a command prompt
 - start “simple example” <path>/pvpython.exe candriver.py <generated script> <path>/can.ex2



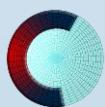
Live In Situ Analysis and Visualization

- Everything before this was “batch”
 - Preset information with possible logic
- “Beta” functionality for interacting with simulation data during simulation run
 - When exporting a Python script, select “Live Visualization”
 - During simulation run choose the “Tools->Connect to Catalyst” GUI menu item

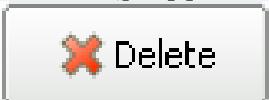


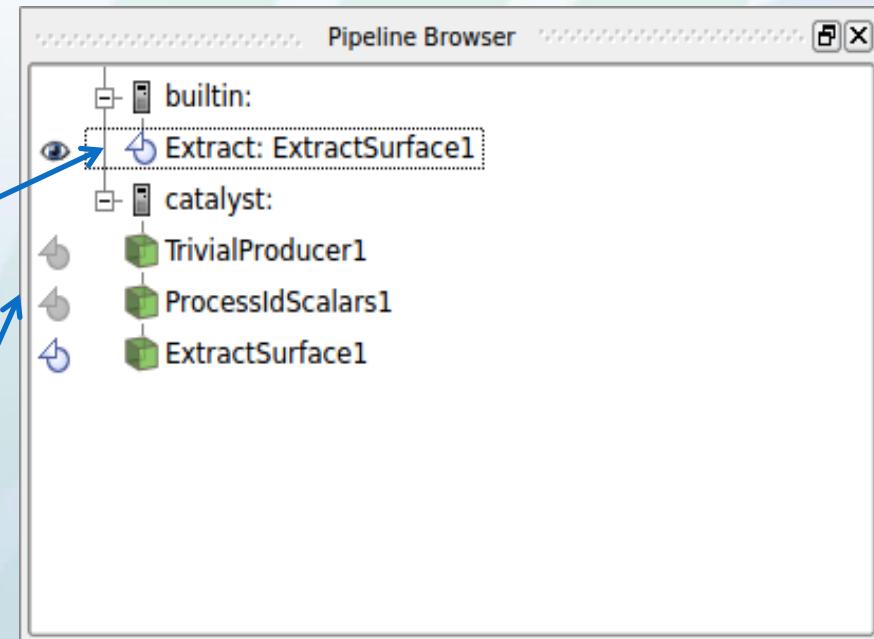
Live *In Situ* Example

- Linux and Mac from a terminal
 - <path>/pvpython livecoprocessing.py <path>/can.ex2
- Windows from a command prompt
 - start “simple example” <path>/pvpython.exe livecoprocessing.py <path>/can.ex2
- Start ParaView and select Tools→Connect to Catalyst
 - Select port (22222 is default)

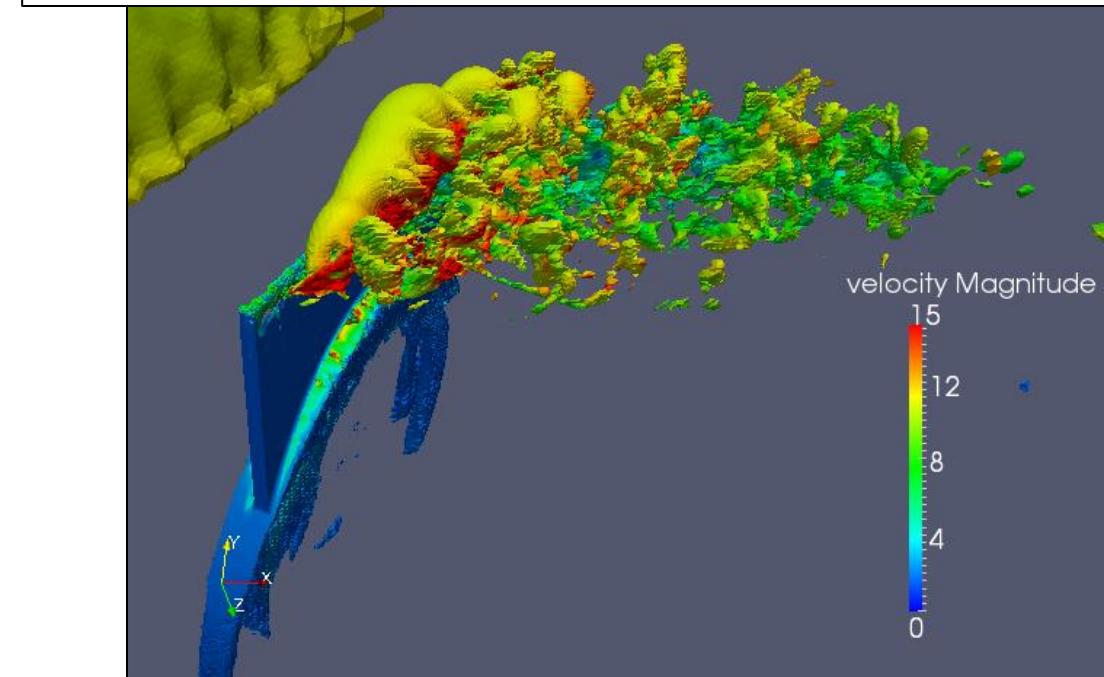
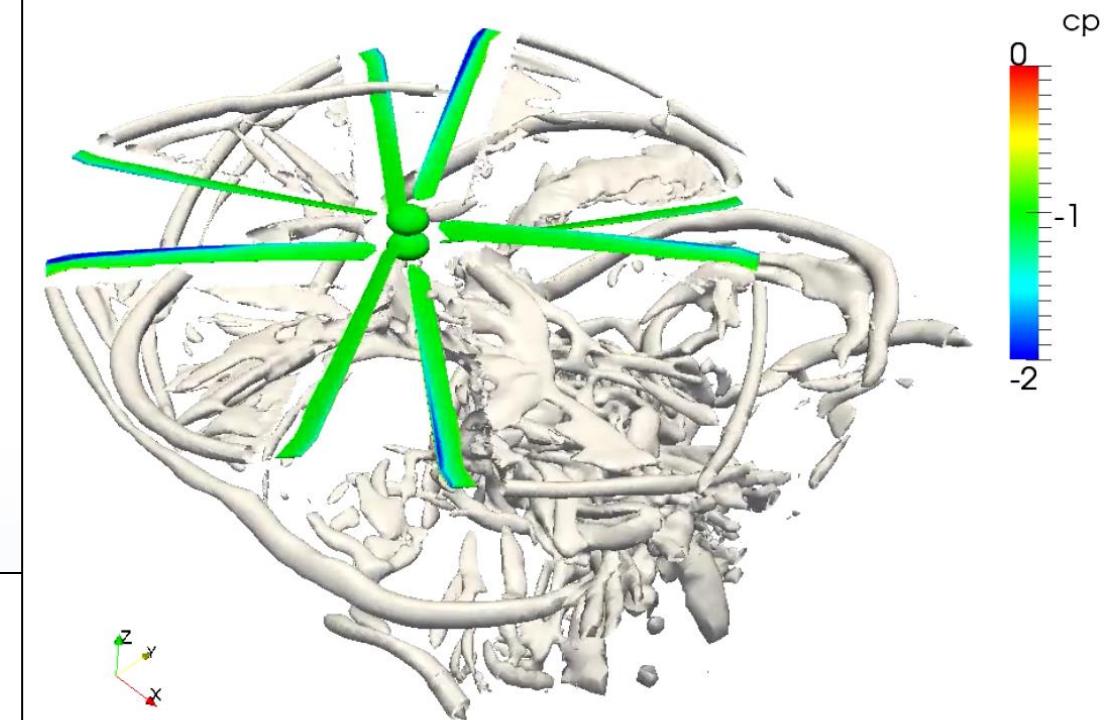
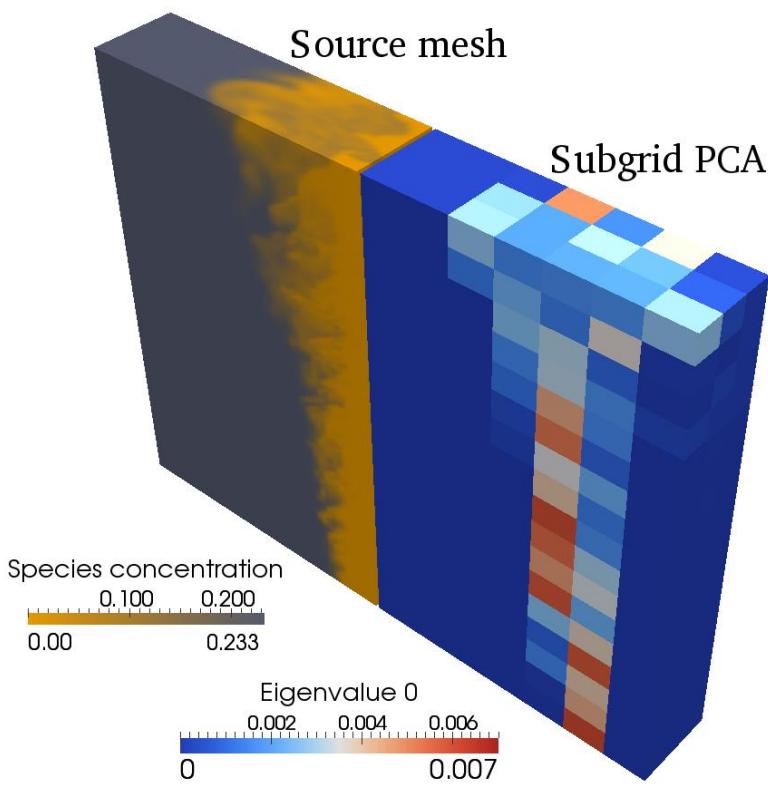


Live *In Situ* Example

- Only transfer requested data from server (simulation run) to client
 - ExtractSurface1 is already getting extracted
- Use  on client  to stop transferring to client
- Click on  to transfer to client from Catalyst



Gratuitous Catalyst Images



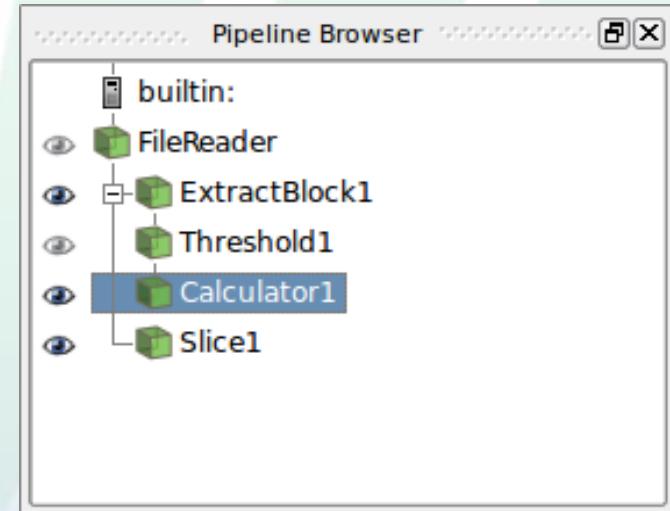


ParaView Catalyst for Developers



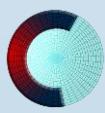
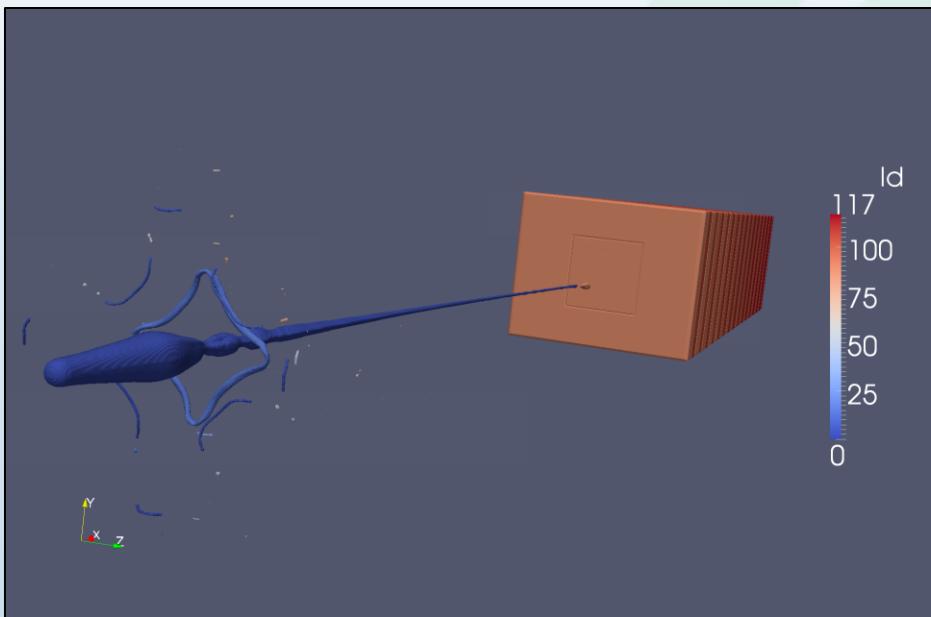
ParaView Pipeline Concept

- Fundamental concept in ParaView
- Directed acyclic graph specifying how to process information
- Filters are nodes in the graph
 - Perform a certain action on a data set (grid and fields)
 - Contours, streamlines, file IO, etc.
 - Do not modify input data set
- Catalyst executes user pipelines at specified times



Catalyst Pipelines

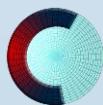
- User generated Python scripts from the ParaView plugin
 - Executed with `vtkCPPythonScriptPipeline`
- Hard-coded pipelines (“canned” output)
 - Executed with a class that derives from `vtkCPPipeline`



vtkCPPythonScriptPipeline

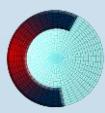
- Typically from ParaView script generator plugin
- Initialize from a path to a ParaView Catalyst Python script
 - `vtkCPPythonScriptPipeline::Initialize(const char* fileName)`

<http://www.paraview.org/ParaView3/Doc/Nightly/html/classvtkCPPythonScriptPipeline.html>



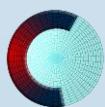
vtkCPPythonScriptPipeline (2)

- Advantages
 - Easily created through ParaView plugin
 - Output should be moderately readable
 - Encourage advanced users to modify
 - Can use ParaView's Python trace utility to see options
 - View/screenshot settings can be difficult to set
 - Camera angle, zoom, lighting, data representations, etc.
 - Takes care of parallel image compositing
 - Can modify without recompiling



vtkCPPythonScriptPipeline (3)

- Disadvantages
 - Slight overhead compared to C++ hard-coded pipeline
 - Roughly 10^{-5} seconds per time step
 - Simulation code must be linked with Python
 - Static build issues
 - More complex to minimize executable size



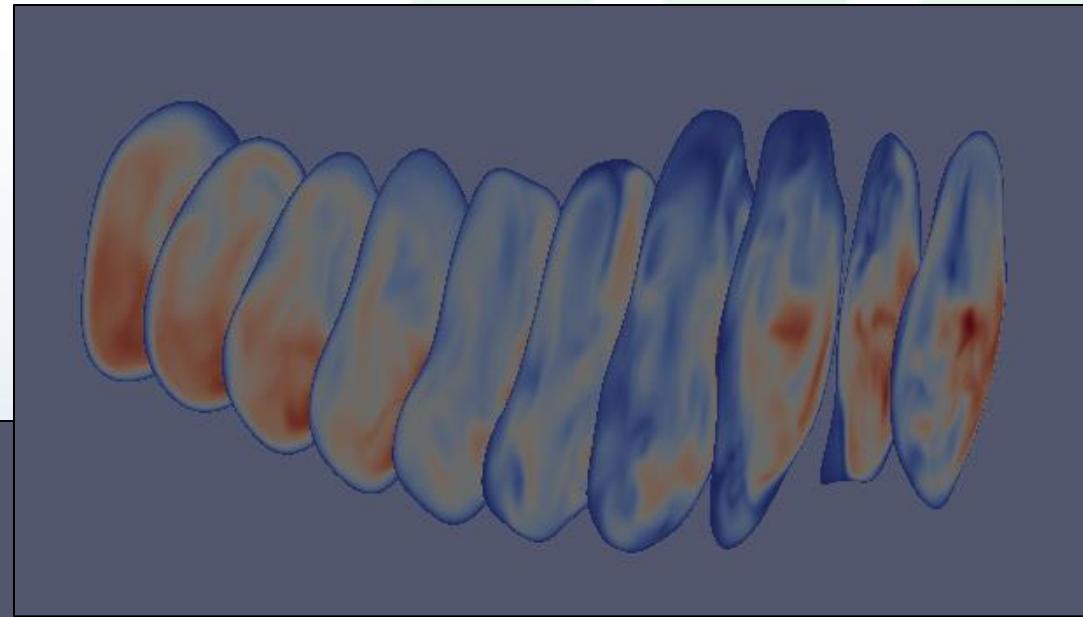
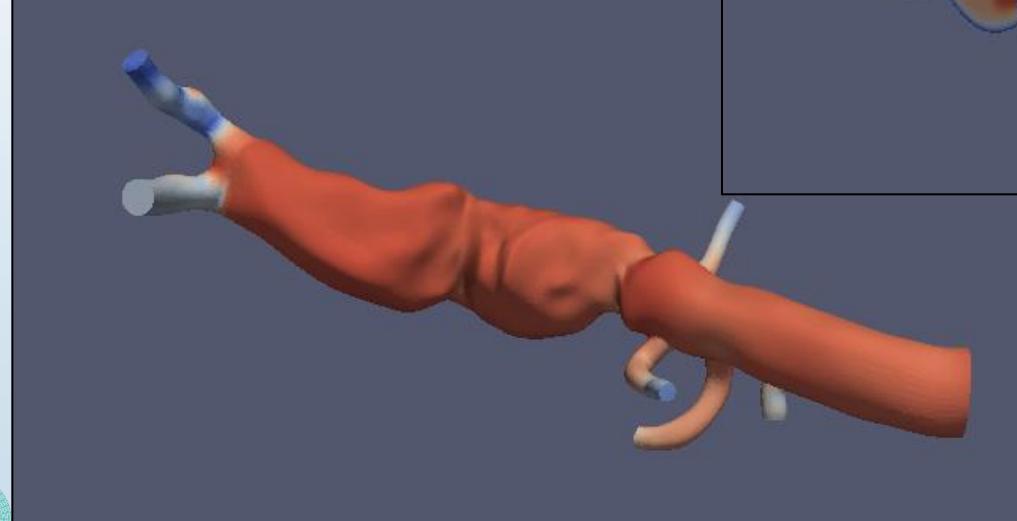
Hard-coded Pipelines

- Derives from `vtkCPPipeline`
- Generally C++ but could be Python code
- Most are done directly creating VTK filters and connecting them together
 - Creating screen shots from rendering pipeline with compositing can be daunting
- Possible to do using ParaView C++ proxies
 - Low level access is more complex
 - Simpler set up for rendering pipeline
- Less dependencies for compiling and linking

<http://www.paraview.org/ParaView3/Doc/Nightly/html/classvtkCPPipeline.html>

Catalyst

- Catalyst's job is to create and execute pipelines

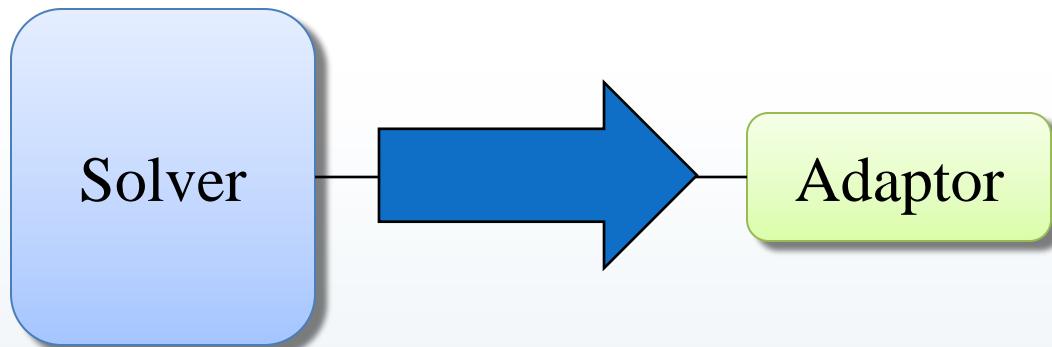


Data Structures

- Simulation has separate data structures from VTK data structures
- User an adaptor to bridge the gap
 - Try to reuse existing memory
 - Also responsible for other interactions between simulation code and Catalyst

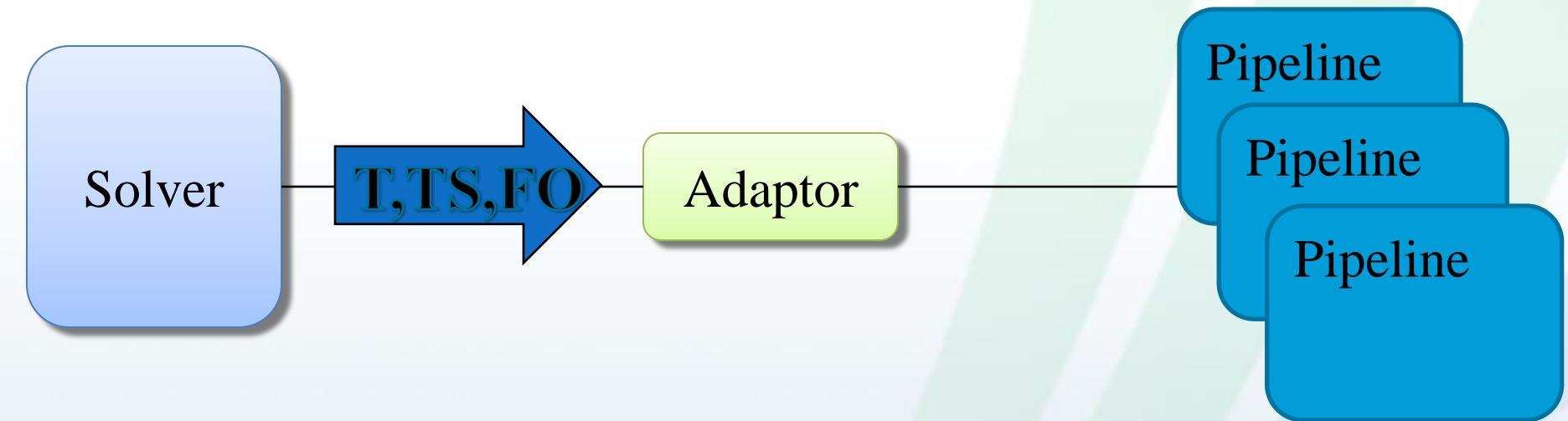


Information Flow



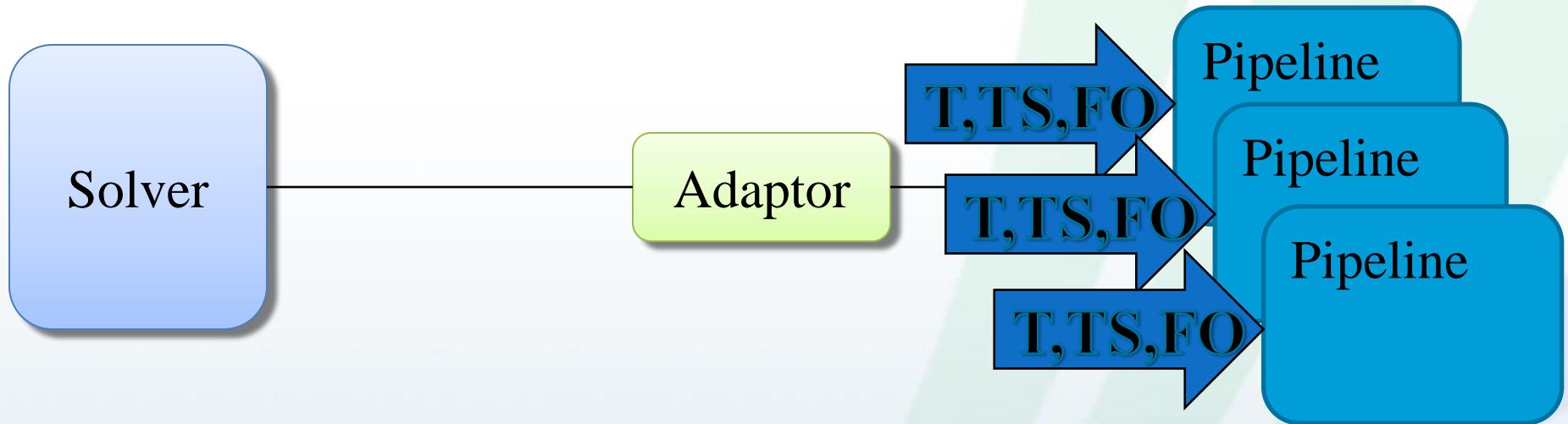
- Initialization
 - Information for creating pipelines

Information Flow

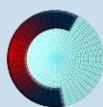


- After simulation completes time step update
 - Time, time step, force output flag
 - Information for creating grid and field information

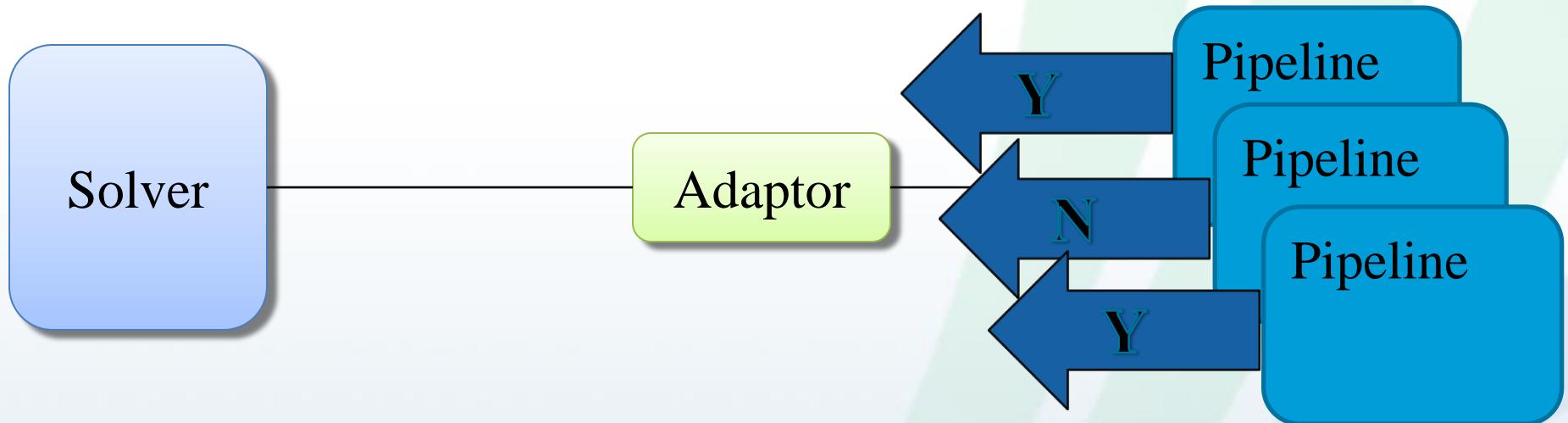
Information Flow



- After simulation completes time step update
 - Time, time step, force output flag passed to each pipeline

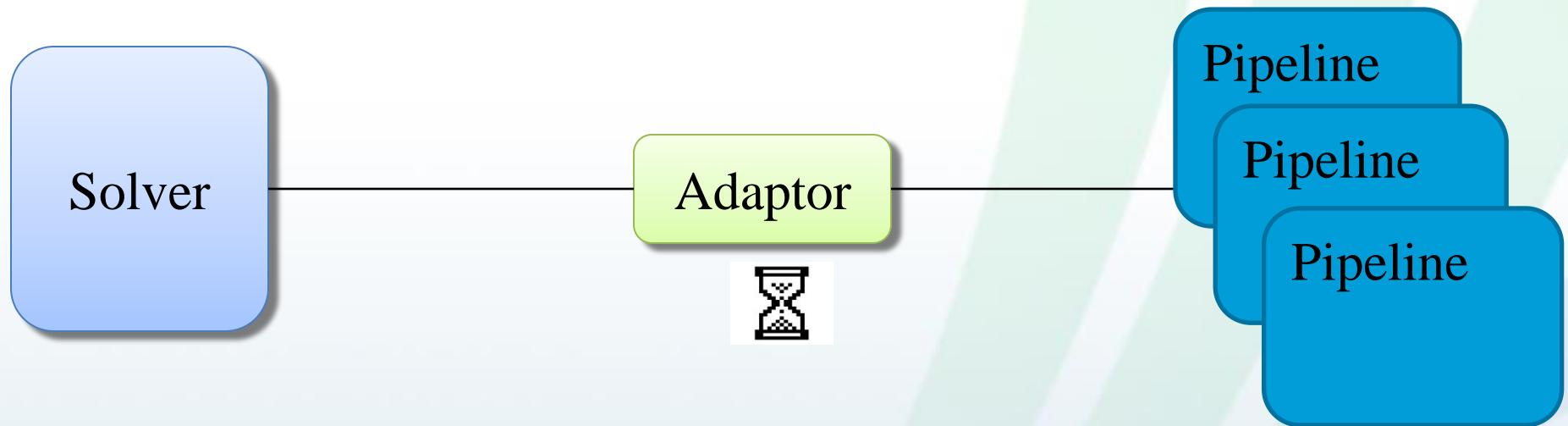


Information Flow



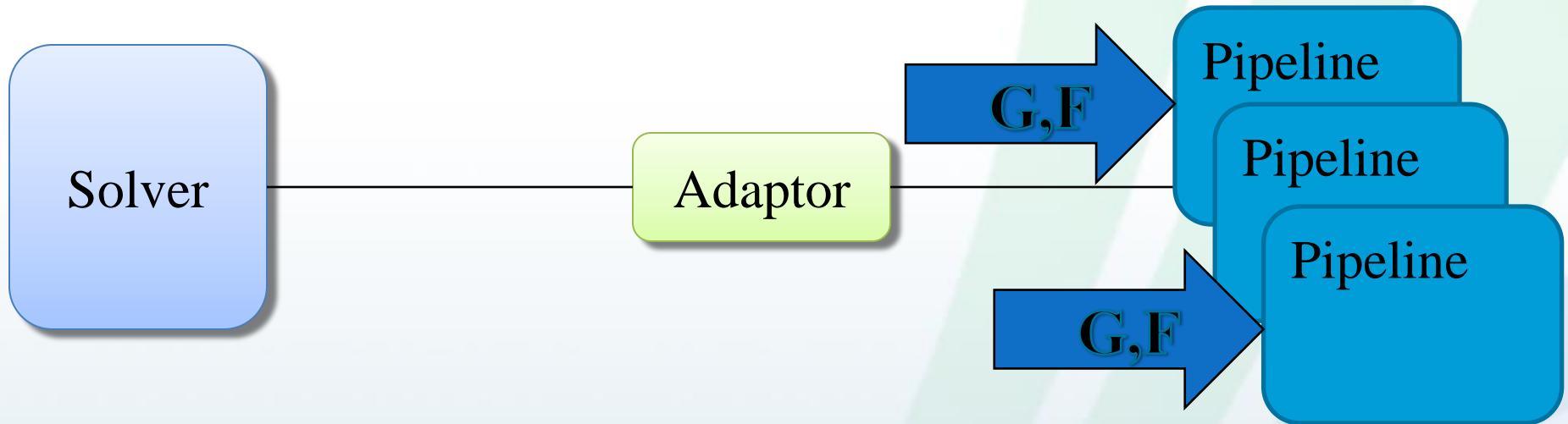
- After simulation completes time step update
 - Flag indicating which pipelines need to be executed/updated

Information Flow



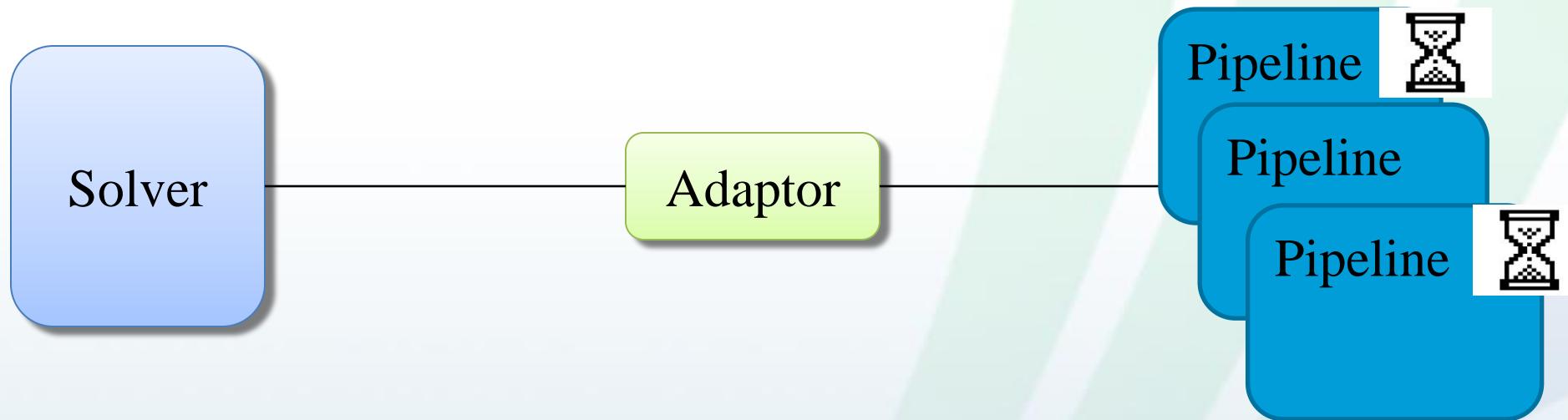
- After simulation completes time step update
 - If any pipeline needs to be executed
 - Adaptor creates VTK objects to represent grids and fields

Information Flow



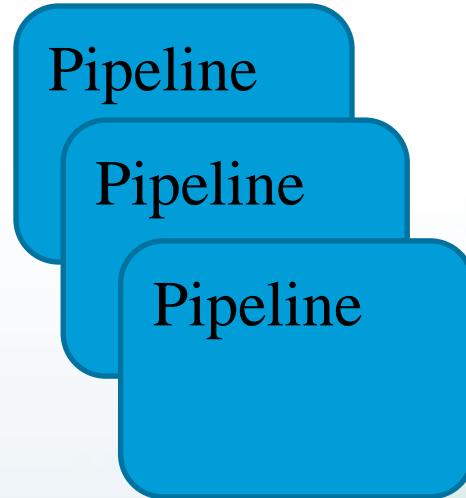
- After simulation completes time step update
 - Pass VTK data object representing grids and fields to pipelines that need to execute/update

Information Flow



- After simulation completes time step update
 - Pipelines execute and output desired information

What is Catalyst?

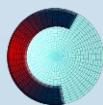


- Communication/synchronization routines
- Pipeline mechanics
- Data processing through filters
- Writers
- Compositing and rendering

Interfacing with Catalyst

Adaptor

- Catalyst calls should have minimal footprint in simulation code
 - Initialization call
 - Co-processing call
 - Finalize call
- VTK data structures usually not appropriate for simulation code data structures
- Adaptor's job is to provide interface between simulation code and Catalyst

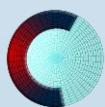


Adaptor Architectural View

Adaptor:

```
vtkCPPProcessor* Processor;  
vtkDataObject* CreateGrid();  
vtkCPPipeline* CreatePipeline();
```

- Initialization
 - Creates vtkCPPProcessor and initializes it
 - vtkCPPProcessor manages pipelines
 - vtkCPPProcessor::Initialize()
 - Creates vtkCPPipeline objects and adds them to vtkCPPProcessor
 - vtkCPPProcessor::AddPipeline(vtkCPPipeline*)



Adaptor Architectural View

Adaptor:

```
vtkCPPProcessor* Processor;  
vtkDataObject* CreateGrid();  
vtkCPPipeline* CreatePipeline();
```

- When called after update time step is completed
 - Creates a vtkCPDataDescription
 - Information needed by Catalyst to determine what pipelines need to execute
 - vtkCPDataDescription::SetTimeData(double time, vtkIdType timeStep)
 - vtkCPDataDescription::SetForceOutput(bool) *optional*
 - Queries pipelines to determine if co-processing is necessary
 - Processor->RequestDataDescription(vtkCPDataDescription*)
 - Returns 0 if no co-processing is needed that time step

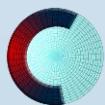


Adaptor Architectural View

Adaptor:

```
vtkCPPProcessor* Processor;  
vtkDataObject* CreateGrid();  
vtkCPPipeline* CreatePipeline();
```

- If Processor->RequestDataDescription() != 0
 - CreateGrid()
 - Creates grids and fields
 - Biggest part of the adaptor
 - vtkCPDataDescription::AddInput(const char* name)
 - Convention is that name := “input” for a single input
 - vtkCPDataDescription::GetInputDataDescriptionByName(const char* name)->SetGrid(CreateGrid())
 - Processor->CoProcess(vtkCPDataDescription*)

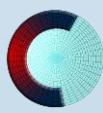


Adaptor Architectural View

Adaptor:

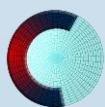
```
vtkCPPProcessor* Processor;  
vtkDataObject* CreateGrid();  
vtkCPPipeline* CreatePipeline();
```

- Finalize
 - Processor->Finalize()



Main Co-Processing Parts

- Initialize
- Check with each Catalyst pipeline if execution is necessary
 - Input is simulation time and time step
 - Output is if co-processing is needed along with needed grids and fields
- If co-processing is needed at specified time and time step:
 - Input the grid and field information
 - Execute the proper Catalyst pipelines
- Finalize



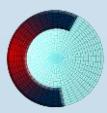
C++ Example

```
int main(int argc, char* argv[])
{
    MPI_Init(&argc, &argv);
    std::string cpPythonFile = argv[1];
    int nSteps = atoi(argv[2]);
    vtkCPPProcessor* processor = vtkCPPProcessor::New();
    processor->Initialize();
    vtkCPPythonScriptPipeline* pipeline =
        vtkCPPythonScriptPipeline::New();

    // read the coprocessing python file
    if(pipeline->Initialize(cpPythonFile.c_str()) == 0) {
        cout << "Problem reading the python script.\n";
        return 1;
    }
    processor->AddPipeline(pipeline);
    pipeline->Delete();

    if (nSteps == 0) {
        return 0;
    }

...
```



C++ Example (2)

```
...
double tStart = 0.0;
double tEnd = 1.0;
double stepSize = (tEnd - tStart) / nSteps;

vtkCPDataDescription* dataDesc = vtkCPDataDescription::New();
dataDesc->AddInput("input");

for (int i = 0; i < nSteps; ++i)
{
    double currentTime = tStart + stepSize*i;
    // set the current time and time step
    dataDesc->SetTimeData(currentTime, i);

    // check if the script says we should do coprocessing now
    if(processor->RequestDataDescription(dataDesc) != 0)
    {
        // create our vtkDataObject with the grids and fields
        vtkDataObject* dataObject = <generate grid>;
        dataDesc->GetInputDescriptionByName("input")->SetGrid(dataObject);
        processor->CoProcess(dataDesc);
    }
}

dataDesc->Delete();
processor->Finalize();
processor->Delete();

MPI_Finalize();

return 0;
}
```





Creating VTK Objects



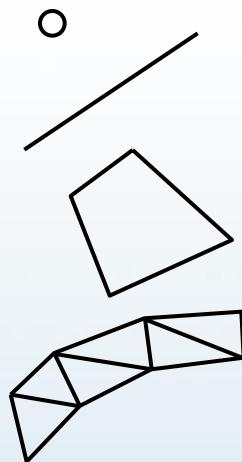
Getting Data Into Catalyst

- Main object will derive from vtkDataObject
 - Grids that derive from vtkDataSet
 - Multiblock grids that contain multiple vtkDataSets
- Field (aka attribute) information
 - Point data – information specified for each point in a grid
 - Cell data – information specified for each cell in a grid
 - Field data – meta-data not associated with either points or cells
- All object groups are 0-based/C/C++ indexing

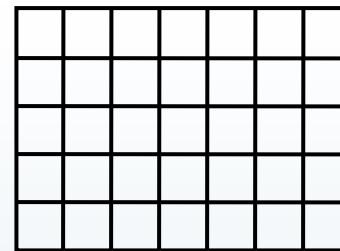


vtkDataSet Subclasses

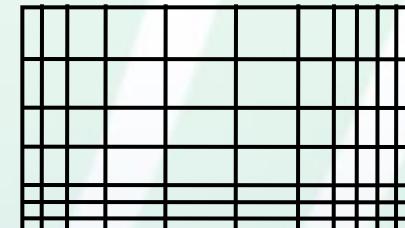
vtkPolyData



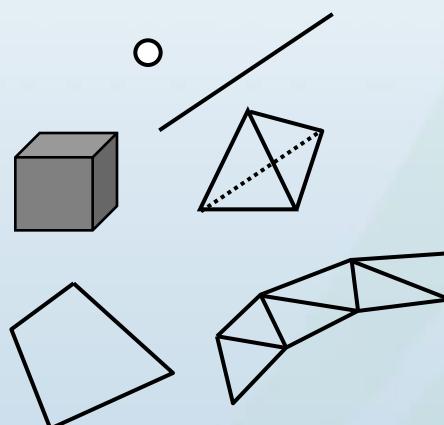
vtkImageData
vtkUniformGrid



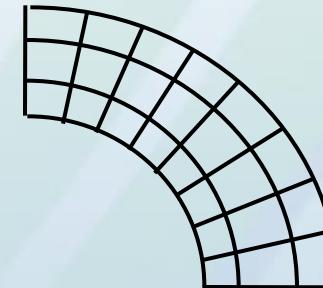
vtkRectilinearGrid



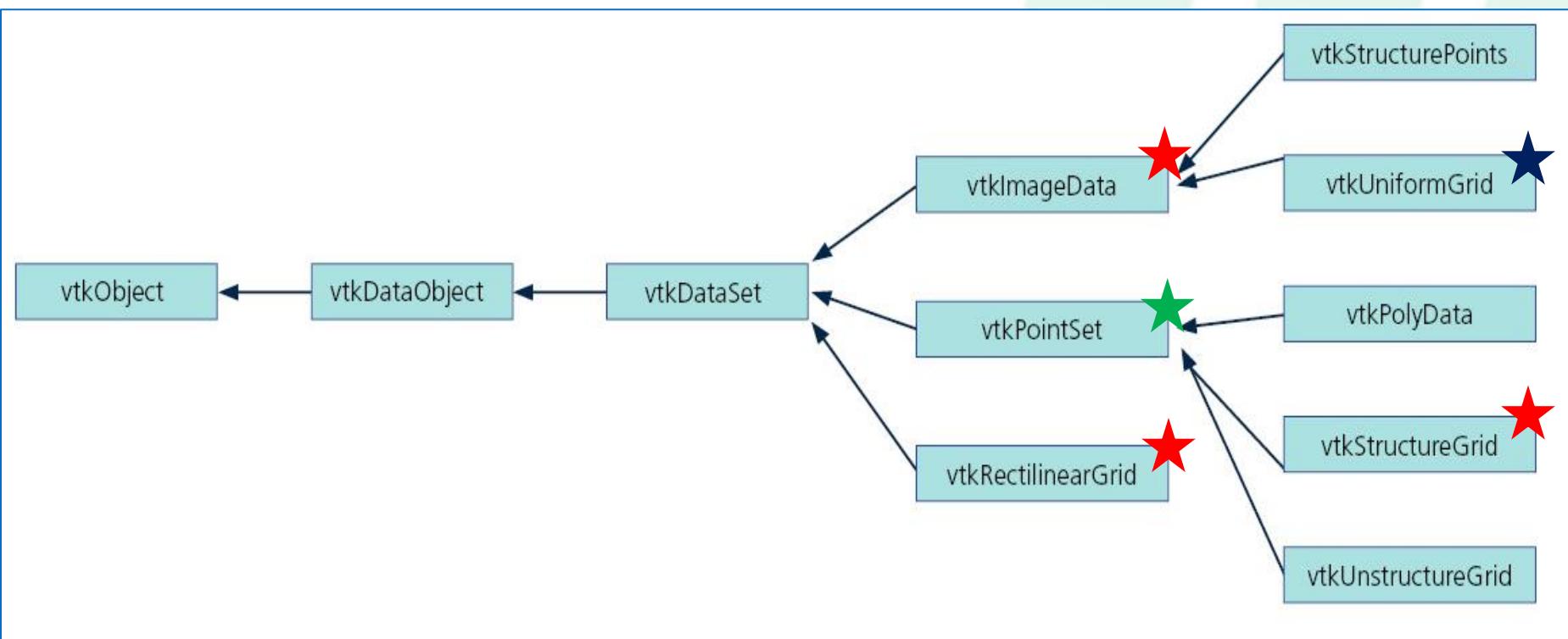
vtkUnstructuredGrid



vtkStructuredGrid



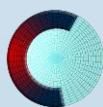
vtkDataSet Class Hierarchy



★ Topologically regular grid

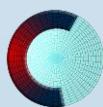
★ Irregular geometry

★ Supports blanking



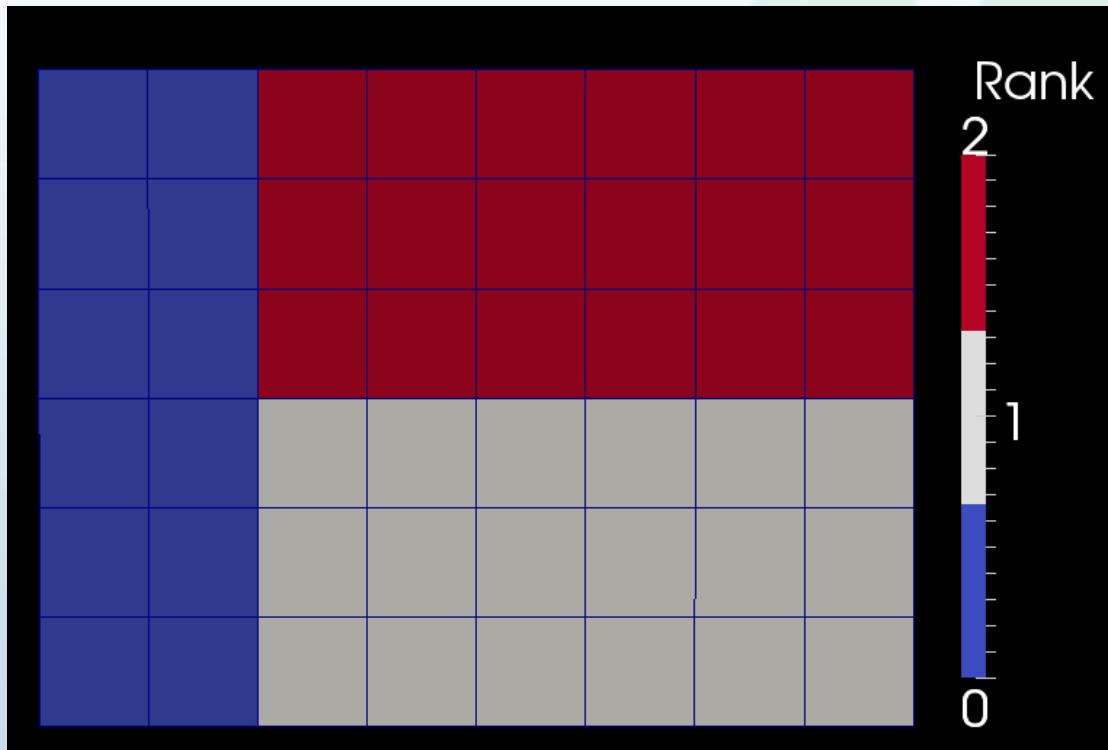
Topologically Regular Grids

- `vtkImageData/vtkUnstructuredGrid`,
`vtkRectilinearGrid` and `vtkStructuredGrid`
- Topological structure defined by whole extent
 - Gives first and last point in each logical direction
 - Not required to start at 0
- Partitioning defined by extents
 - First and last point in each logical direction of part of the entire grid
- Ordering of points and cells is fastest in logical x-direction, then y-direction and slowest in z-direction



Extents

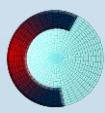
- Whole extent for all processes (0, 8, 0, 6, 0, 0)
- Extents different for each process
 - Rank 0: (0, 2, 0, 6, 0, 0), 21 points, 12 cells
 - Rank 1: (2, 8, 0, 3, 0, 0), 28 points, 18 cells
 - Rank 2: (2, 8, 3, 6, 0, 0), 28 points, 18 cells



Extents

- Point and cell indices for extent of (3, 8, 3, 6, 0, 0)
 - From rank 2 in previous slide
 - 28 points and 18 cells

21 (2,6,0)					27 (8,6,0)
12 (2,5,0)	13 (3,5,0)	14 (4,5,0)	15 (5,5,0)	16 (6,5,0)	17 (7,5,0)
6 (2,4,0)	7 (3,4,0)	8 (4,4,0)	9 (5,4,0)	10 (6,4,0)	11 (7,4,0)
0 (2,3,0)	1 (3,3,0)	2 (4,3,0)	3 (5,3,0)	4 (6,3,0)	5 (7,3,0)
0 (2,3,0)					6 (8,3,0)

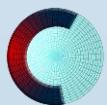
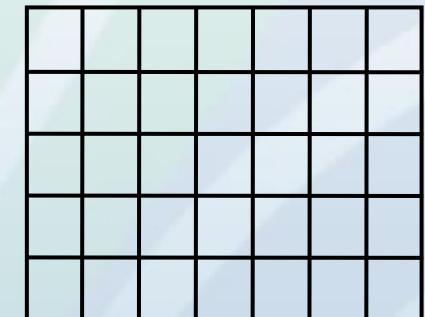


Kitware

vtkImageData/vtkUniformGrid

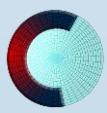
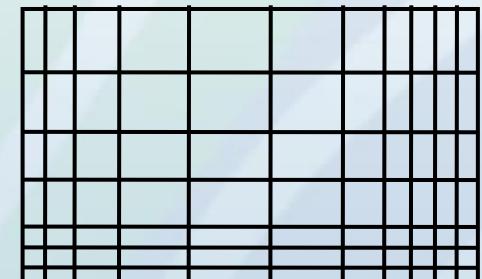
- `vtkCPInputDataDescription::SetWholeExtent()` – total number of points in each direction
- `SetExtent()` – a process's part of the whole extent
- `SetSpacing()` – cell lengths in each direction
- `SetOrigin()` – location of point 0 ($i=0, j=0, k=0$)

- `vtkUniformGrid`
 - Supports cell blanking
 - Currently written to file as `vtkImageData`



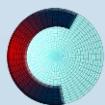
vtkRectilinearGrid

- vtkCPInputDataDescription::SetWholeExtent() – total number of points in each direction
- SetExtents() – a process's part of the whole extent
- Set<X,Y,Z>Coordinates() – point coordinates in each direction
 - Only values for process's extents
 - Index starting at 0



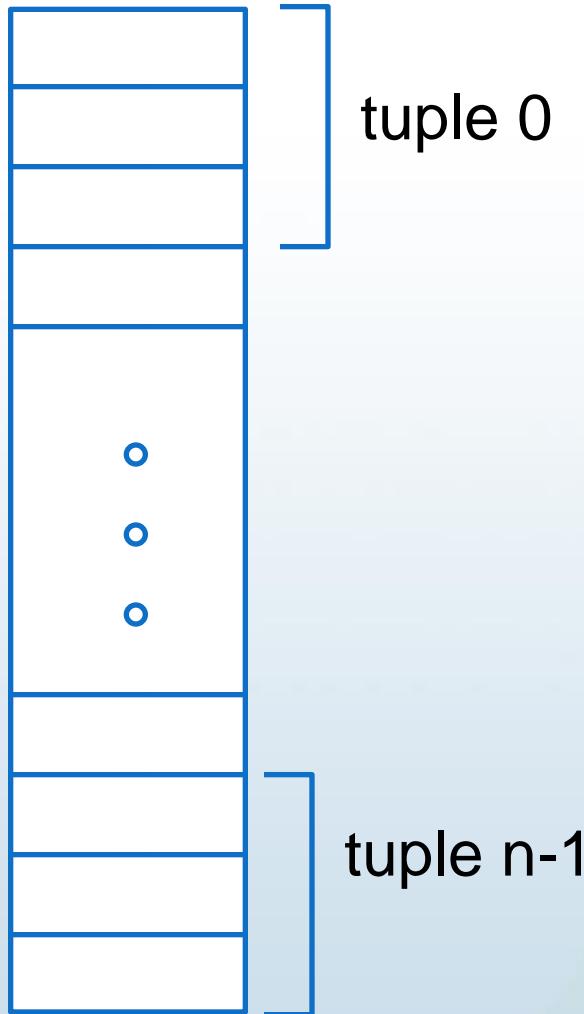
Irregular Geometry

- Data sets that derive from `vtkPointSet`
 - `vtkStructuredGrid` – still topologically regular
 - `vtkPolyData` – topologically irregular with 0D, 1D and 2D cell types
 - `vtkUnstructuredGrid` – topologically irregular with 0D, 1D, 2D and 3D cell types
- Uses `vtkPoints` to specify grid's point locations
 - `SetPoints(vtkPoints*)` – set the number and coordinates of a process's points



vtkdataArray – Basis for vtkDataObject

Contents



- An array of n tuples
- Each tuple has m components which are logically grouped together
- Internal implementation is a pointer to an $n \times m$ block of memory
- Data type determined by class
- Two APIs : generic and data type specific



vtkdataArray

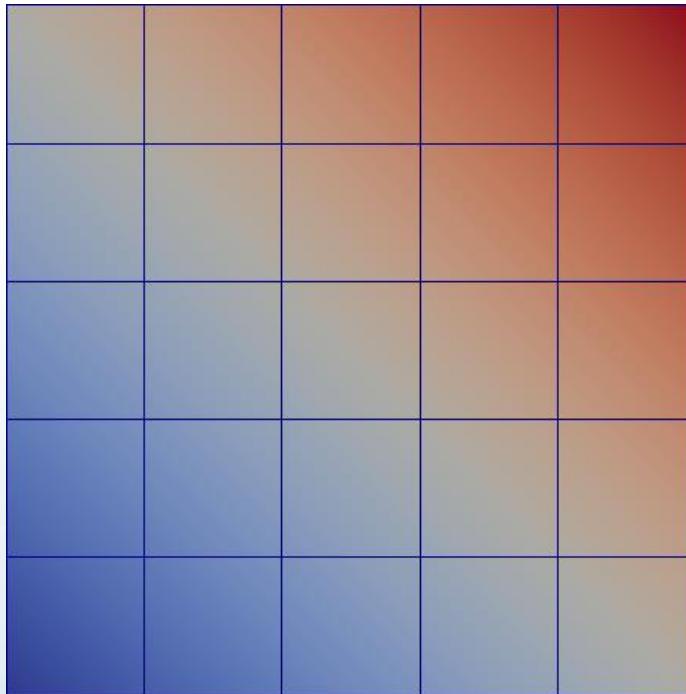
- SetNumberOfComponents() – call first
- SetNumberOfTuples()
 - For point data must be set to number of points
 - For cell data must be set to number of cells
- SetArray()
 - If flat array has proper component & tuple ordering use existing simulation memory – most efficient
 - Can specify who should delete
 - VTK uses pipeline architecture so Catalyst libraries will NOT modify array values
- SetTupleValue() – uses native data type
- SetValue() – uses native data type
- SetName() – array descriptor, e.g. velocity, density

vtkFieldData

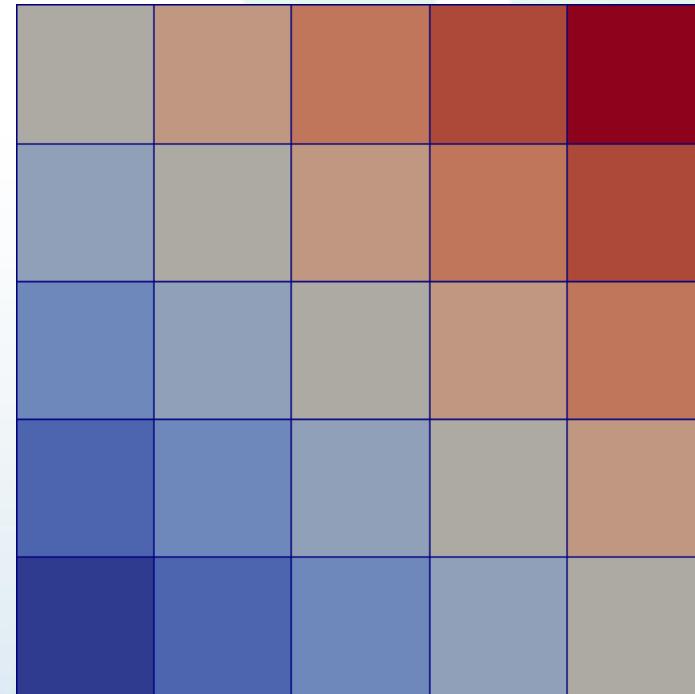
- Object for storing vtkDataArrays
- vtkDataSet::GetFieldData() – non-grid associated arrays
- Derived classes
 - vtkPointData – vtkDataSet::GetPointData()
 - vtkCellData – vtkDataSet::GetCellData()
- vtkFieldData::AddArray(vtkDataArray*)
- Specific arrays are normally retrieved by name from vtkFieldData
 - Uniqueness is not required but can cause unexpected results



Point Data and Cell Data



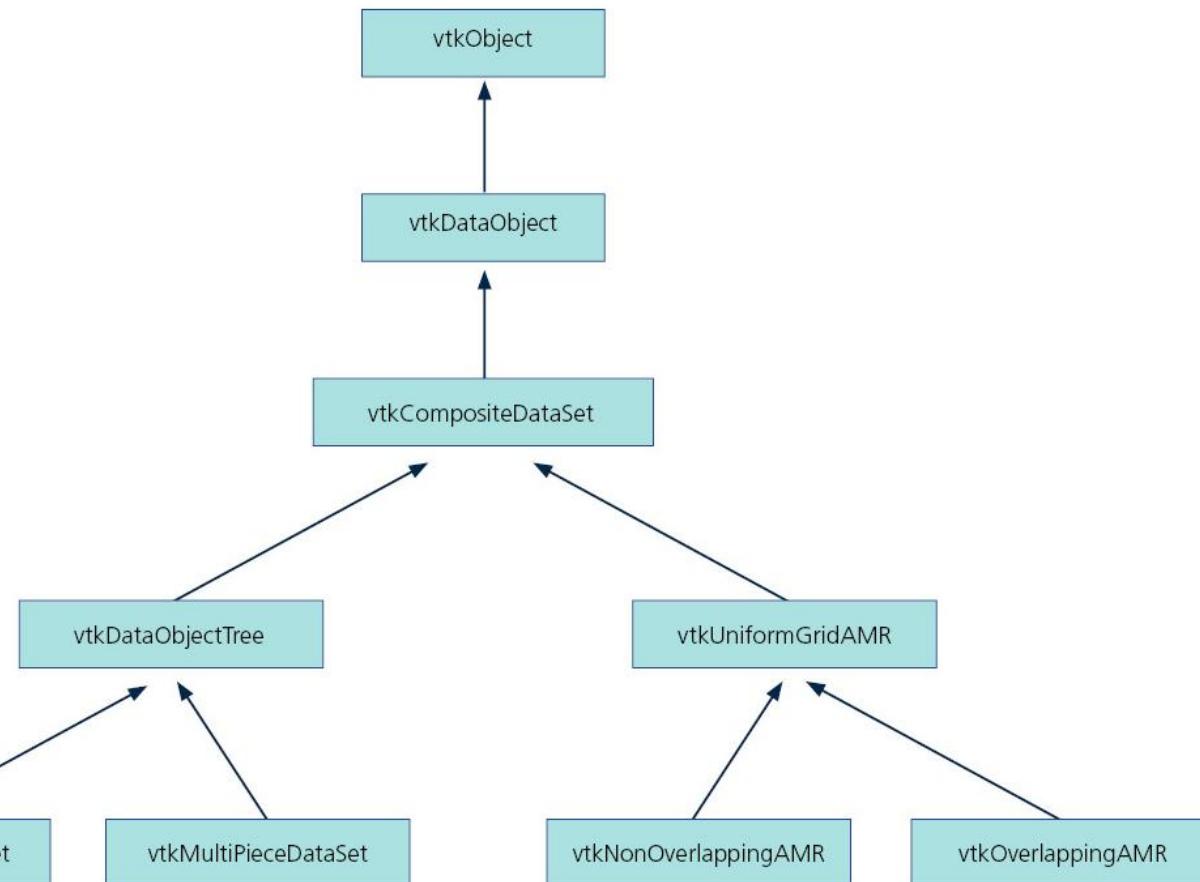
Point data – 36 values



Cell data – 25 values

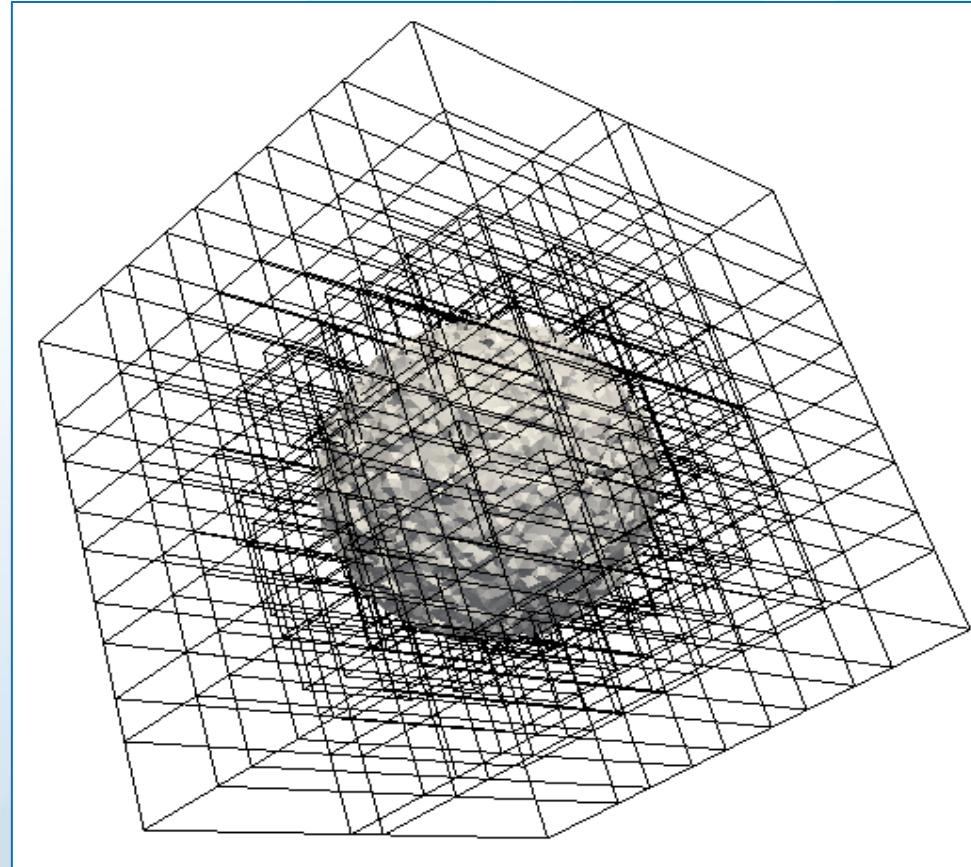
Multiple Grids

- Use something that derives from `vtkCompositeDataSet` to group multiple `vtkDataSets` together



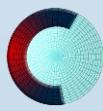
vtkMultiBlockDataSet

- Most general way of storing vtkDataSets and other vtkCompositeDataSets
- Block structure must be the same on each process
- Block is null if data set is on a different process
- SetNumberOfBlocks()
- SetBlock()



vtkMultiPieceDataSet

- Way to combine multiple vtkDataSets of the same type into a single logical object
 - Useful when the amount of pieces is unknown *a priori*
- Must be contained in a vtkMultiBlockDataSet
- SetNumberOfPieces()
- SetPiece()



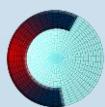
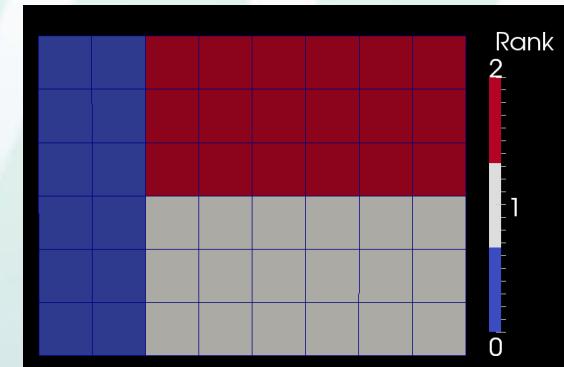
AMR Data Sets

- Classes derive from `vtkUniformGridAMR`
- Concept of grid levels for changing cell size locally
- Uses only `vtkUniformGrids`
 - `vtkNonOverlappingAMR` – no blanking used since no data sets overlap
 - `vtkOverlappingAMR` – `vtkUniformGrids` overlap and use blanking



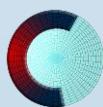
Grid Partitioning

- For unstructured grids and polydatas
 - Single data set per process – use as is
 - Multiple data sets per process – choice of combining or not depends on memory layout
- For topologically structured grids
 - Must be partitioned into logical blocks
 - SetExtent()
 - Index of first and last point in each direction
 - Will be different on each process
 - vtkCPInputDataDescription::SetWholeExtent() – same on each process



vtkUnstructuredGrid Example

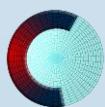
```
void CreateGrid(vtkCPDataDescription* dataDescription, int
numPoints, double* coordsArray, int numCells, int*
cellConnectivity, double* dofArray)
{
    vtkUnstructuredGrid* grid = vtkUnstructuredGrid::New();
    vtkPoints* nodePoints = vtkPoints::New();
    vtkDoubleArray* coords = vtkDoubleArray::New();
    coords->SetNumberOfComponents(3);
    coords->SetNumberOfTuples(numPoints);
    for(int i=0;i<*numPoints;i++)
    {
        double tuple[3] = {coordsArray[i],
                           coordsArray[i+numPoints],
                           coordsArray[i+numPoints*2]};
        coords->SetTupleValue(i, tuple);
    }
    nodePoints->SetData(coords);
    coords->Delete();
    grid->SetPoints(nodePoints);
    nodePoints->Delete();
    ...
}
```



vtkUnstructuredGrid Example (2)

- Only tetrahedra in this example
- Canonical ordering of tetrahedra is same in simulation data structures and VTK

```
...
vtkIdType      [4];
for(int iCell=0;iCell<numCells;iCell++)
{
    for(int i=0;i<4;i++)
    {
        pts[i] = cellConnectivity[iCell+i*numCells];
    }
    grid->InsertNextCell(VTK_TETRA, 4, pts);
}
dataDescription->GetInputDescriptionByName("input")
->SetGrid(grid);
grid->Delete();
...
```



vtkUnstructuredGrid Example (3)

```
...
vtkCPInputDataDescription* idd =
    dataDescription->GetInputDescriptionByName ("input") ;
if(idd->IsFieldNeeded ("velocity")) {
    vtkDoubleArray* velocity = vtkDoubleArray::New () ;
    velocity->SetName ("velocity") ;
    velocity->SetNumberOfComponents (3) ;
    velocity->SetNumberOfTuples (numPoints) ;
    for (vtkIdType idx=0; idx<numPoints; idx++) {
        velocity->SetTuple3 (idx, dofArray[idx] ,
            dofArray[idx+ numPoints], dofArray[idx+ 2*numPoints]) ;
    }
    grid->GetPointData ()->AddArray (velocity) ;
    velocity->Delete () ;
}
if(idd->IsFieldNeeded ("pressure")) {
    vtkDoubleArray* pressure = vtkDoubleArray::New () ;
    pressure->SetName ("pressure") ;
    pressure->SetArray (dofArray+3*numPoints, numPoints, 1) ;
    grid->GetPointData ()->AddArray (pressure) ;
    pressure->Delete () ;
}
}
```

