In Situ Analysis and Visualization with Ascent and ParaView Catalyst

[Conduit Mesh Representation Introduction and Hands-on]

SC23 Tutorial
Monday November 13th, 2023

Cyrus Harrison, Lawrence Livermore National Laboratory (LLNL)

Jean M. Favre, Swiss National supercomputing Centre (CSCS)

A heat-diffusion mini-app to demonstrate in-situ visualization with Ascent and Catalyst

Jean M. Favre

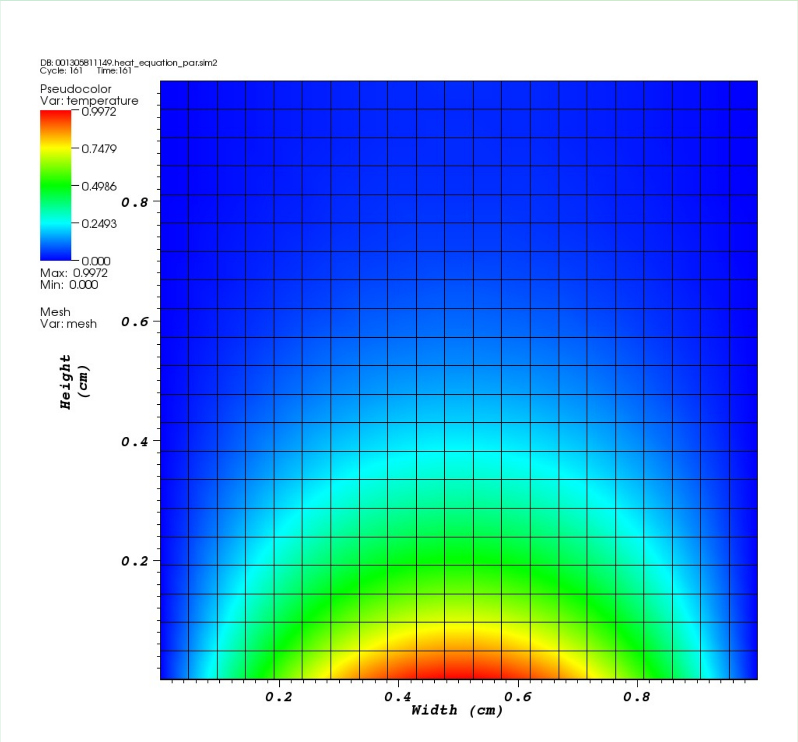
Swiss National supercomputing Centre (CSCS)

Motivations

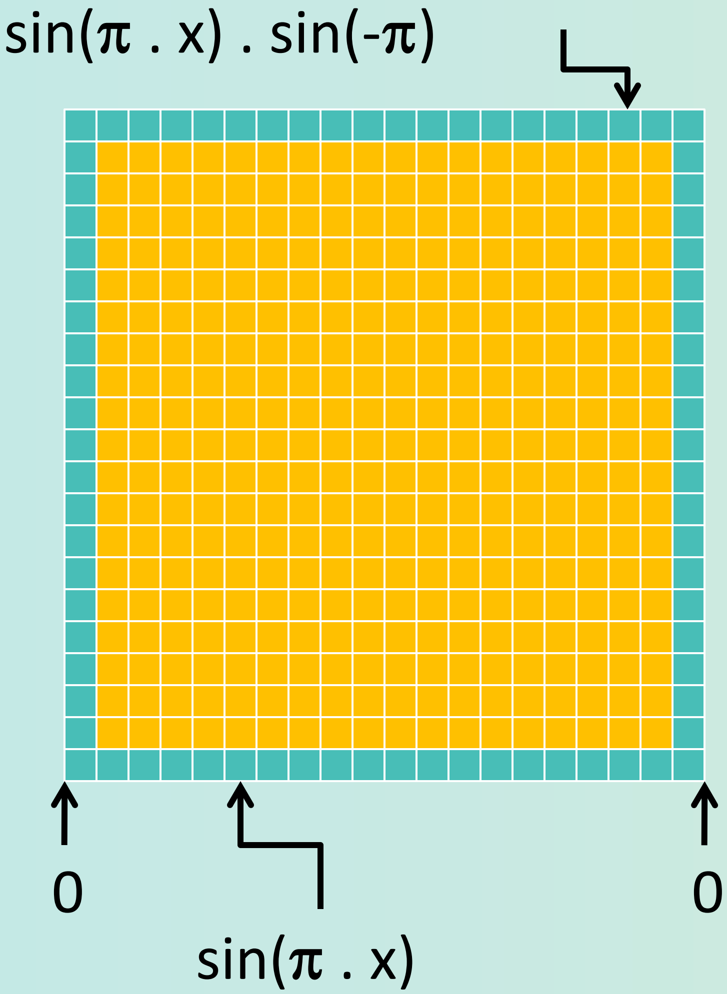
- A simple demonstrator for in-situ couplings based on Conduit + {Ascent, Catalyst}
  - in C++ and Python
  - with MPI (optional)
  - with four different grid types (uniform, rectilinear, structured and unstructured)

Assumptions

- This 2D heat-diffusion solver relies on a very simple 5-point stencil to update the nodes of a mesh.
- No focus on computational performance
- A very simple partitioning grid for parallel runs
- The underlying grid points form a uniform, Cartesian grid, defined solely with an **origin**, a fixed **spacing** between points (in X and Y), and a grid **resolution**
- For didactic purposes only, the same grid can also be viewed as:
  - a rectilinear grid (special case with equally spaced grid points along both axis)
  - A structured grid (special case with the points’ coordinates explicitly given)
  - An unstructured grid of quadrilateral cells (with an explicit connectivity list)
- The physical global domain is [0.,0.]->[1.,1.]



The domain fixed boundary conditions



Update grid with solver

Fixed boundary conditions

Laplace equation: $\Delta u = 0$

Generic run-time options

usage: heat_diffusion_insitu_*.py [-h] [-t TIMESTEPS] [--res RES] [-m {uniform,rectilinear,structured,unstructured} [-n] [-v]

-h, --help   show this help message and exit

  -t TIMESTEPS, --timesteps TIMESTEPS

                        number of timesteps to run the miniapp (default: 1000)

  --res RES   resolution in each coordinate direction (default: 64)

  -m {uniform,rectilinear,structured,unstructured}, --mesh {uniform,rectilinear,structured,unstructured}

                        mesh type (default: uniform)

-n, --noinsitu   toggle the use of the in-situ vis coupling

-v, --verbose   toggle printing of the conduit nodes

API-specific run-time options

With Ascent

-f FREQUENCY, --frequency FREQUENCY

How often should the Ascent script be executed

-d DIR, --dir DIR   path to a directory where to dump the Blueprint output (final step)

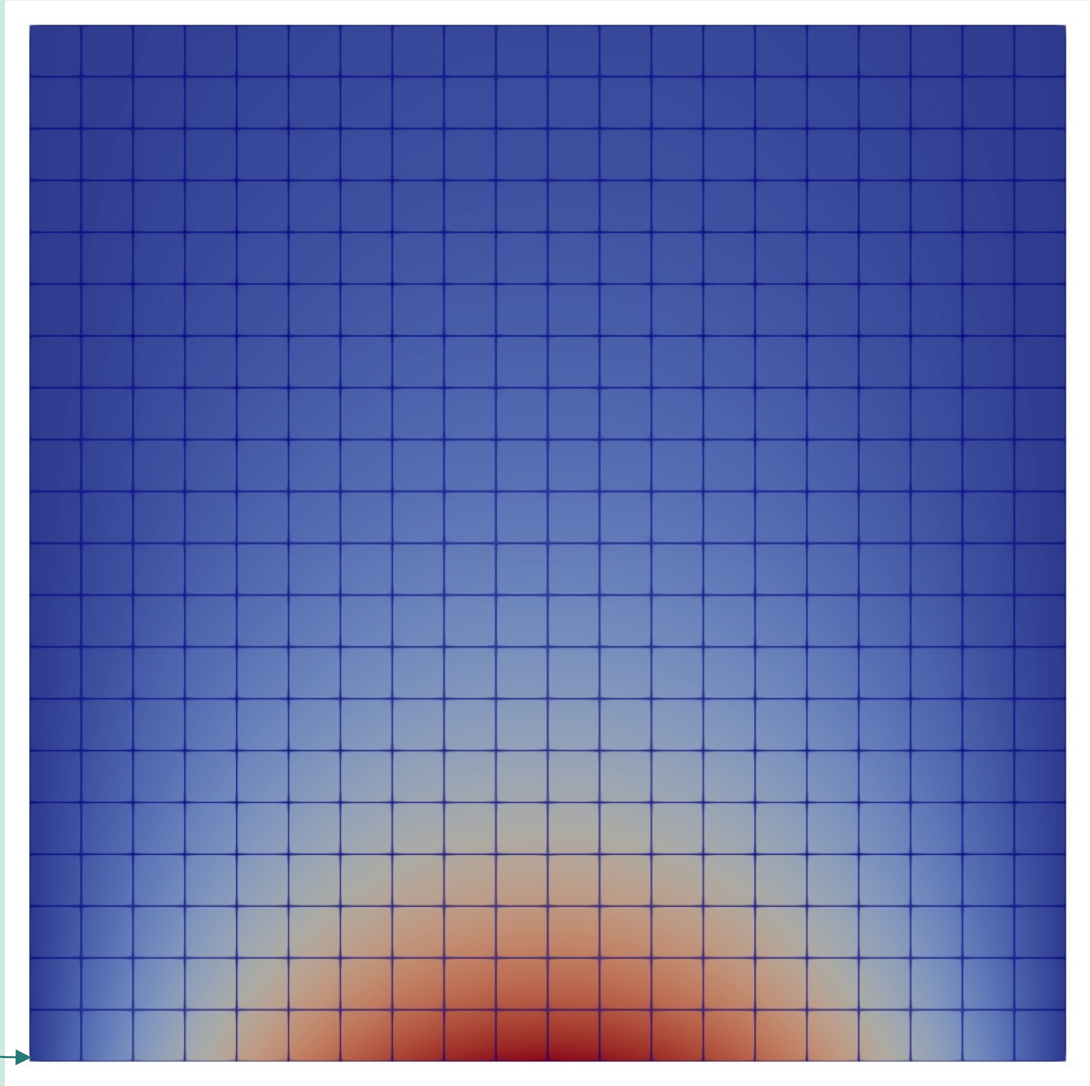
With Catalyst

-s SCRIPT, --script SCRIPT

path to the Catalyst script to use for in situ processing.

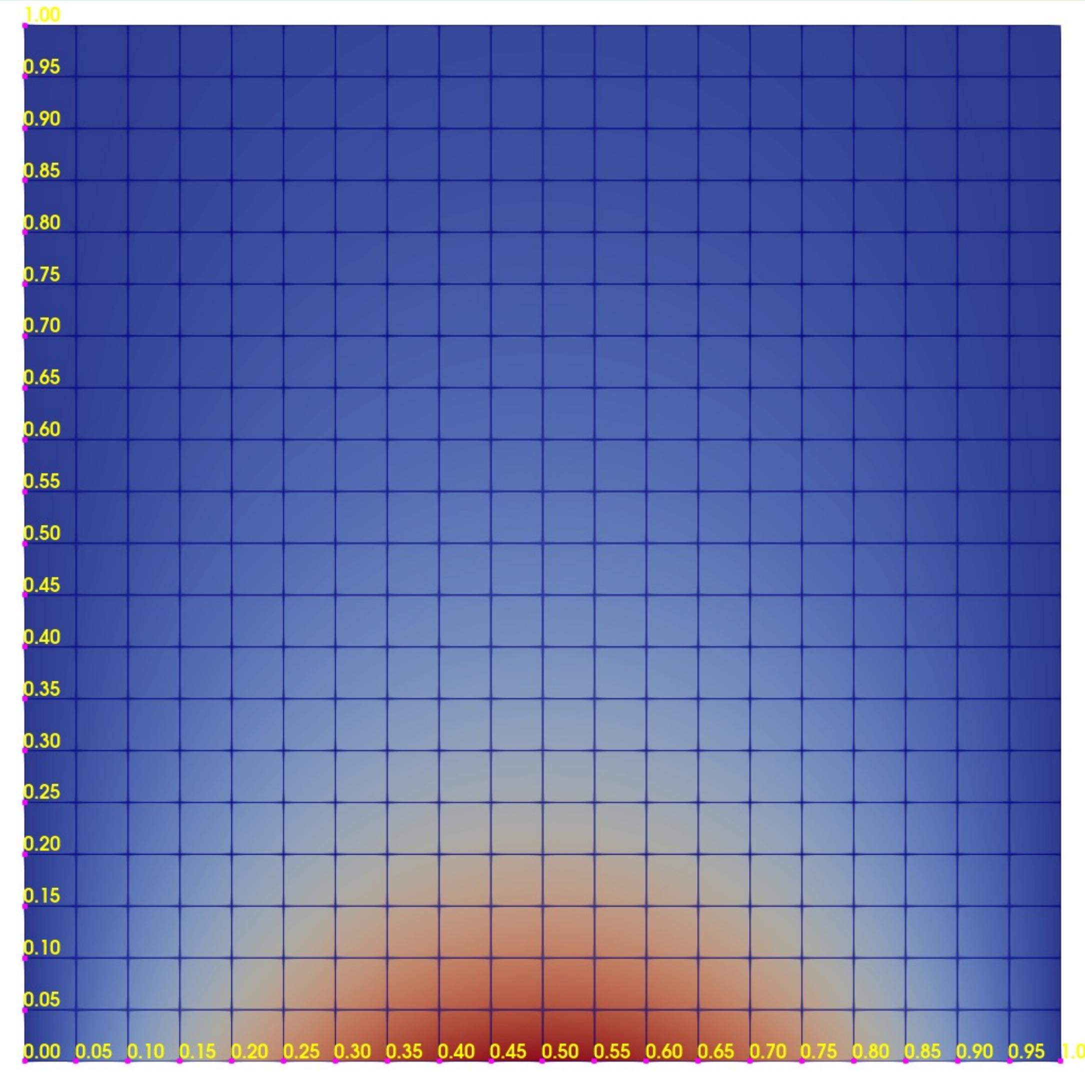
The default case: a uniform grid


```
coordsets:
  coords:
    type: "uniform"
    dims:
      i: 21
      j: 21
    origin:
      x: 0.0
      y: 0.0
    spacing:
      dx: 0.05
      dy: 0.05
topologies:
  mesh:
    type: "uniform"
    coordset: "coords"
fields:
  temperature:
    association: "vertex"
    topology: "mesh"
    Values: [0.0, 0.156, 0.309, ..., 0.006, 5.29e-18]
```

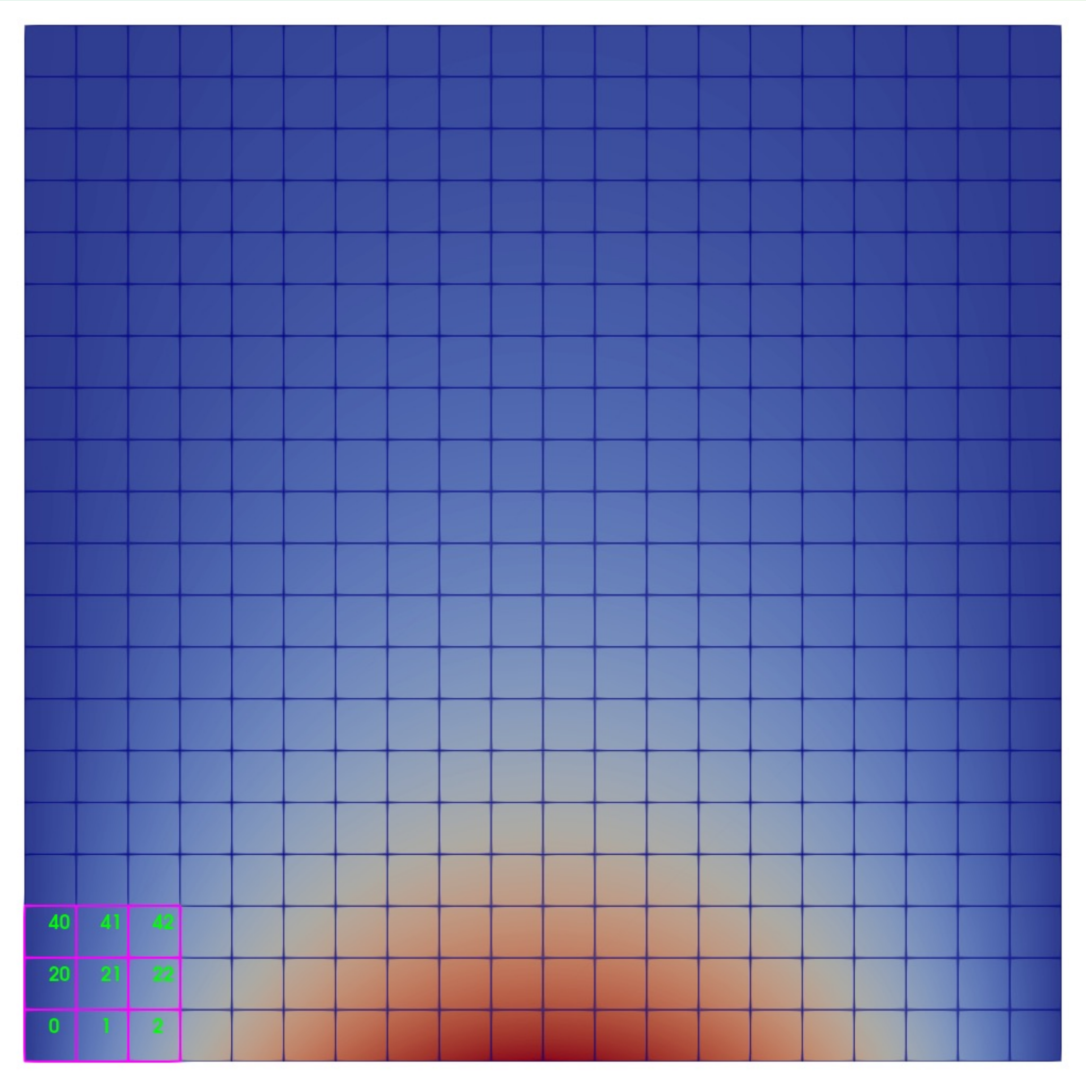
The rectilinear grid

coordsets:
  coords:
    type: “rectilinear"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
topologies:
  mesh:
    type: " rectilinear "
    coordset: "coords"
fields:
  temperature:
    association: "vertex"
    topology: "mesh"
    values: [0.0, 0.156, 0.309, ..., 0.006,
5.29e-18]



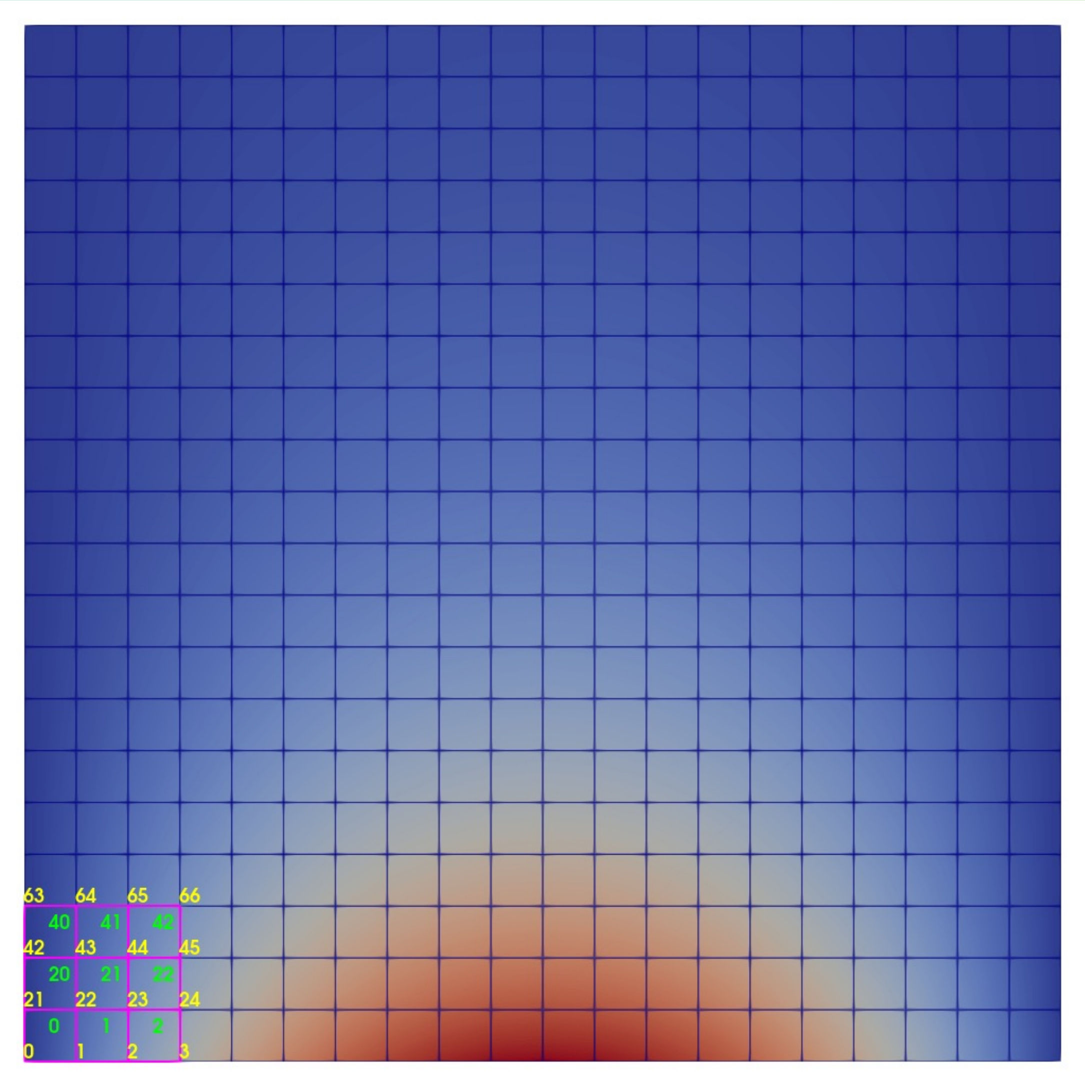
The structured grid

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: [0.0, 0.0, 0.0, ..., 1.0, 1.0]
topologies:
  mesh:
    type: "structured"
    coordset: "coords"
     elements:
      dims:
        i: 20
        j: 20
fields:
  temperature:
    association: "vertex"
    topology: "mesh"
    values: [0.0, 0.149, 0.294, ..., 0.006,
5.29e-18]
```



The unstructured grid

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: [0.0, 0.0, 0.0, ..., 1.0, 1.0]
topologies:
  mesh:
    type: "unstructured"
    coordset: "coords"
    elements:
      shape: "quad"
      connectivity: [0, 21, 22, 1,..., 440, 419]
fields:
  temperature:
    association: "vertex"
    topology: "mesh"
    values: [0.0, 0.156, 0.309, ..., 0.006,
5.29e-18]
```



Actions need to be defined

Ascent

```
actions = conduit.Node()
add_act = actions.append()
add_act["action"] = "add_scenes"

# declare a scene (s1) to render the dataset
scenes = add_act["scenes"]
scenes["s1/plots/p1/type"] = "pseudocolor"
scenes["s1/plots/p1/field"] = "temperature“
# add a second plot to draw the grid lines
scenes["s1/plots/p2/type"] = "mesh"
```

ParaView Catalyst (can be written by the ParaView client)

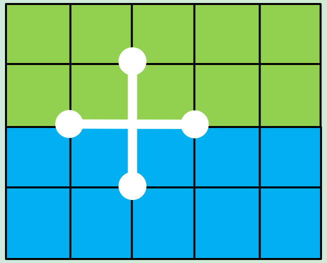
```
renderView1 = GetRenderView()
reader = TrivialProducer(registrationName='grid')

rep = Show(reader, renderView1)
rep.Representation = 'Outline'
ColorBy(rep, ['POINTS', 'temperature'])
temperatureLUT = GetColorTransferFunction('temperature')
temperatureLUT.RescaleTransferFunction(0.0, 1.0)

contour1 = Contour(Input=reader)
contour1.ContourBy = ['POINTS', 'temperature']
contour1.ComputeScalars = 1
contour1.Isosurfaces = [i*0.1 for i in range(11)]

png1 = CreateExtractor('PNG', renderView1)
png1.Trigger = 'TimeStep'
png1.Trigger.Frequency = 100
png1.Writer.FileName = 'view-{timestep:06d}{camera}.png'
```

Going parallel

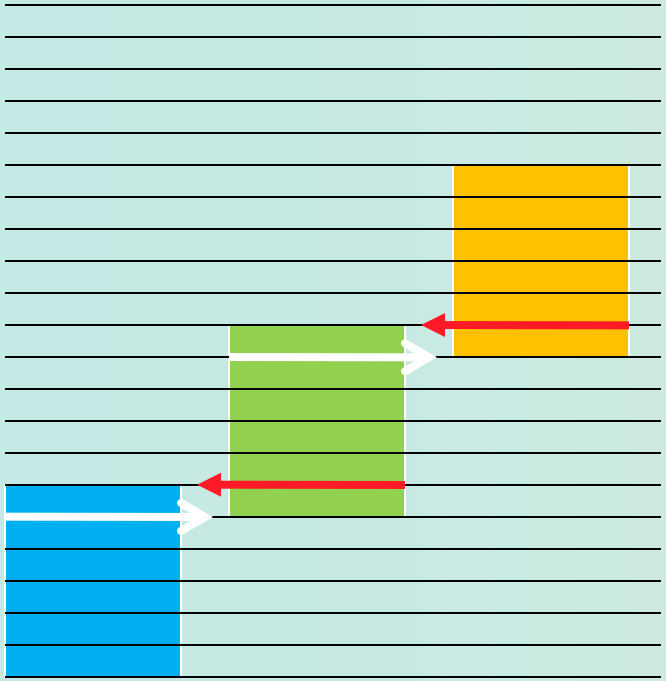
MPI-BASED EXECUTION

The Python example uses a 1D grid partitioning

- The grid is partitioned along the Y direction
- Boundary conditions are set
- A single line of ghost-nodes insure that the 5-point stencil is continuous across MPI task boundaries



Ghost data exchange for N MPI tasks



Overlap Send and Receive

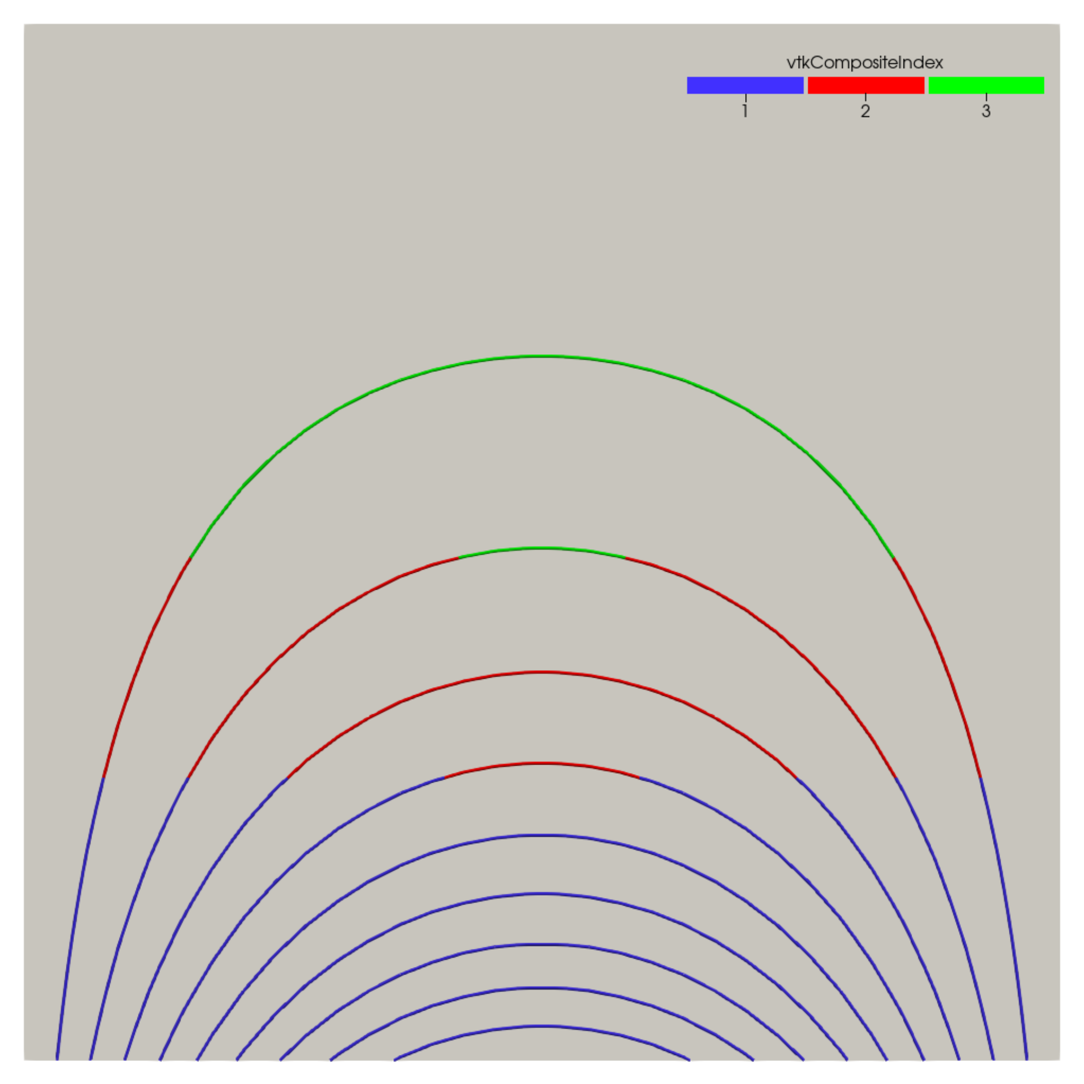
Proc. 0 does not receive data from “below”

Proc. (N-1) does not send data “above”

Conduit nodes: What’s new in parallel?

```
coordsets:
  coords:
    type: "uniform"
    dims:
      i: 21
      j: … => new for each MPI task
    origin:
      x: 0.0
      y: … => new for each MPI task
```

```
coordsets:
  coords:
    type: “rectilinear"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: … => new for each MPI task
```



Iso-contour lines, colored by their process-id

Conduit nodes:
What’s new in parallel?

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: … => new for each MPI task
topologies:
  mesh:
    type: "structured"
    coordset: "coords"
     elements:
      dims:
        i: 20
        j: … => new for each MPI task
```

```
coordsets:
  coords:
    type: "explicit"
    values:
      x: [0.0, 0.05, 0.1, ..., 0.95, 1.0]
      y: … => new for each MPI task topologies
  mesh:
    type: "unstructured"
    coordset: "coords"
    elements:
      shape: "quad"
      connectivity: … => new for each MPI task
```

N.B. the connectivity list can use local point ids (starting at 0)

The Blueprint output

- It is common for Blueprint data files to represent meshes that have been partitioned and must later be treated as a whole.
- Blueprint root files contain an index that facilitates reading in many individual Blueprint files => metadata about the overall contents of individual files
- The root file is a hierarchical index dataset created with Conduit that has been saved to a file using Relay
- https://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html#mesh-index-protocol

- Can be read by VisIt out-of-the-box
- Currently developing a reader for ParaView (WIP)

The Blueprint output

```
>>> import conduit

>>> import conduit.relay.io

>>> import numpy as np

>>> mesh = conduit.Node()

>>> conduit.relay.io.load(mesh,
"/dev/shm/mesh.cycle_001000.root", "hdf5")

>>> mesh
```


```
blueprint_index:
  mesh:
    state:
      number_of_domains: 4
      partition_pattern:
"mesh.cycle_001000/domain_{domain:06d}.hdf5:/"
      partition_map:
        file: [0, 1, 2, 3]
        domain: [0, 1, 2, 3]
    coordsets:
      coords:
        type: "uniform"
        coord_system:
            type: "cartesian"
      path: "mesh/coordsets/coords"
    topologies:
      mesh:
        type: "uniform"
        coordset: "coords"
        path: "mesh/topologies/mesh"
    fields:
      temperature:
        number_of_components: 1
        topology: "mesh"
        association: "vertex"
```

Demonstrations / exercises

```
python3 heat_diffusion_insitu_parallel_Ascent.py --help

python3 heat_diffusion_insitu_parallel_Ascent.py -v --mesh=uniform --res=64


mpiexec –n 4 python3 heat_diffusion_insitu_parallel_Ascent.py -v --mesh=uniform --res=64
```

```
python3 heat_diffusion_insitu_parallel_Catalyst.py --help

python3 heat_diffusion_insitu_parallel_Catalyst.py -v --mesh=uniform --res=64


mpiexec –n 4 python3 heat_diffusion_insitu_parallel_Catalyst.py -v --script=pvPythonScript.py
```