UV-CDAT Spatio-Temporal Parallel Processing Tools

Intended Audience and Purpose

The UV-CDAT Spatio-Temporal Parallel Processing Tools are designed to dramatically reduce the running time of parallel data analysis jobs under certain conditions. These conditions include:

1. The data is composed of a large number of time steps.
2. For temporal parallelism:
    a. The data analysis operations are to be applied to each time step, and time steps are independent of each other (no data exchange between time steps are required). Examples include generating an image of each time step and creating a contour of each time step.

    OR

    b. The data analysis operations require a reduction operation to be applied over a certain time period. An example would be generating monthly temperature values from daily temperature output.
3. Optionally, for spatial parallelism the data for a single time step does not fit in a single core's available memory while utilizing all cores on the processor or node.

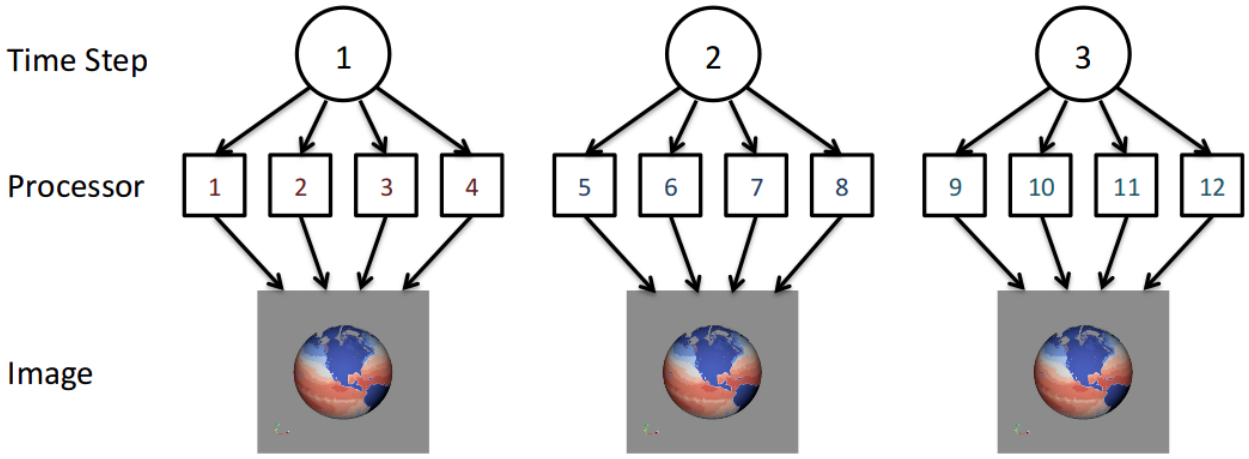
Introduction and problem definition

In the age of "big data", dealing with the massive amount of data can be a enormous challenge. For our definition of big data with respect to scientific computing, we are going by the working definition of data that requires an inordinate amount of time to analyze to obtain salient information from an appropriate computing hardware. Often when the analysis is separate from the computation (i.e. the simulation code producing the data), disk IO can be a significant factor. According to the DOE Exascale Initialize Roadmap, Architecture and Technology Workshop in San Diego in December 2009, there is expected to be a 500 fold increase in the FLOP/s between 2010 and 2018 but only a 100 fold increase in the IO bandwidth in that same time. Because of this we need to concentrate on ways to efficiently post-process simulation data generated on these leading edge supercomputers.

Approach and solutions

To this end we have approached this problem by reducing the amount of global operations that need to be done to process the data. We begin with the assumption that the simulation is time dependent and will be outputting data for multiple time steps. When processing this data, similar operations will need to be done for each time step but we look to group the amount of processes that work on a time step together to a reasonable size instead of using all of the processes available. We call the set of processes working together on a time step as the *time compartment*. By having multiple time compartments we can ensure that there is an

appropriate amount of work being done by each process. Additionally, the communication between processes is generally done within the time compartment and not over the entire global set of processes. In fact, for certain situations such as creating an image for each time step there will be no global inter-process communication. An example of this is shown in the figure below for 3 time steps and 3 time compartments of size 4. Note that for more time steps each time compartment would process multiple time steps for this example.

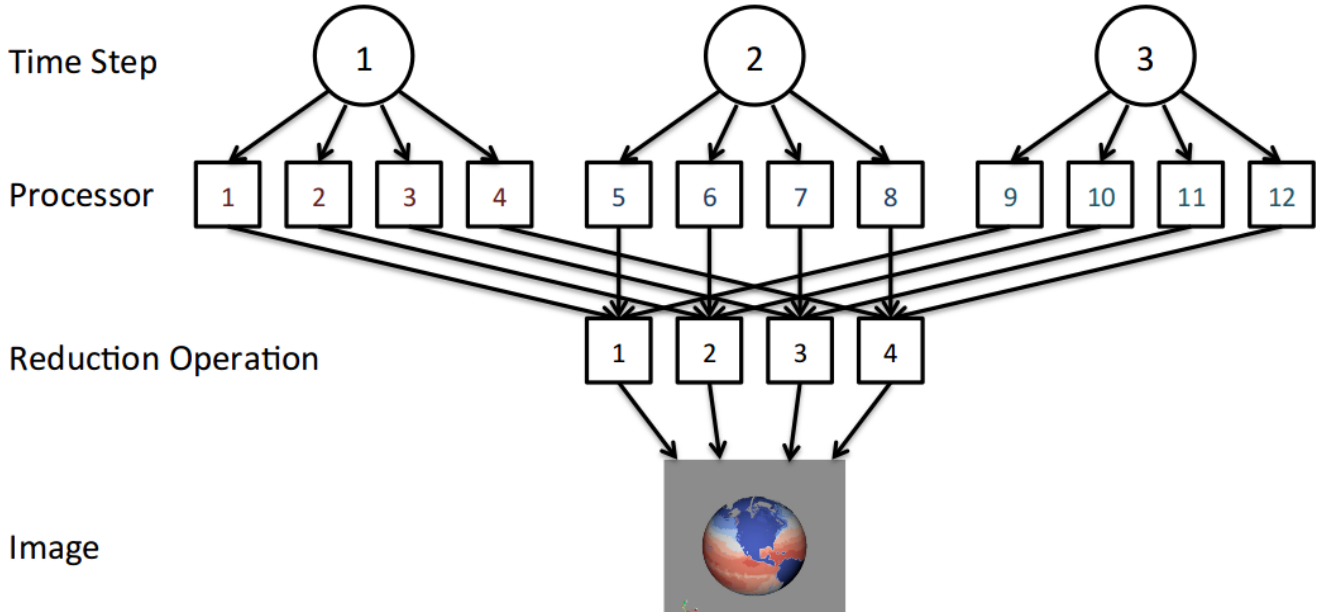


Note that the main advantage of this approach is that it escapes the typical limits of strong scaling for large amounts of processes where the computation per process gets overwhelmed by the communication per process. Similar results are obtained for weak scaling when the number of time steps analyzed is increased with process count while maintaining the same amount of work per time step. This is shown in the table below for results from the Parallel Ocean Program code (POP). Note that 1.4 GB of data is read in per time step.

Number of Timesteps	Number of Processes (P)	Time Compartment = P (seconds)	Time Compartment = 8 (seconds)
2	16	46.96	21.76
4	32	81.84	21.47
8	64	159.77	21.16
16	128	235.61	26.85
32	256	1,103.00	23.13
64	512	2,365.89	25.02
128	1024	8,128.92 (~2 hrs)	30.15
256	**2048**	**28,862.55 (~8 hrs)**	**62.83**

This was done on the Mustang supercomputer at LANL, using 8 cores per node. This is a reasonable time compartment size for this machine as long as the data fits on a single node. The reason for this is that it limits the inter-process communication to intra-node

communication. While many analyses will be time step independent (i.e. time step A and time step B don't need to share any information), we will also have cases where there is a global reduction operation. Computing temporal statistics (e.g. average temperature over all time steps) is an example of this. In this case each time compartment can compute a local set of statistics for all of its input time steps and then a global reduction operation is performed to get the statistics over the entire set of time steps. This is shown in the figure below.



Accessing UV-CDAT spatio temporal pipeline

1. One can access the UV-CDAT spatio-temporal pipeline by either downloading the UV-CDAT binary from http://sourceforge.net/projects/cdat/files/Releases/UV-CDAT/ or building UV-CDAT by following instructions posted at https://github.com/UV-CDAT/uvcdat/wiki/Development
2. Launch ParaView from the UV-CDAT install directory. The instructions to create and run spatio-temporal scripts using ParaView are described in the next section.

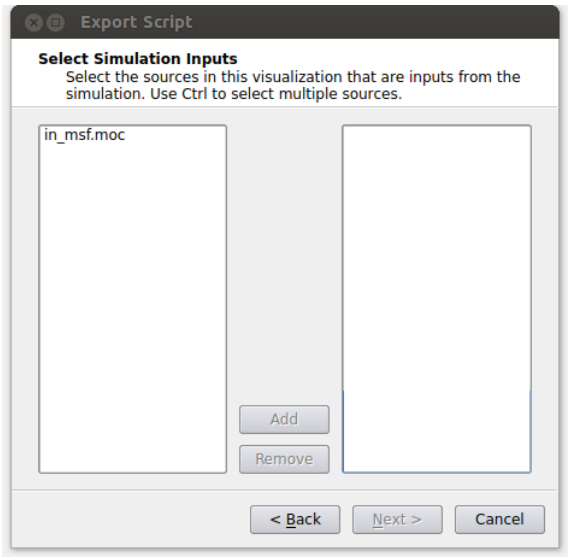
Also, we have been working towards extending uvcdat spatio-temporal environment to support ingestion of custom / CDAT ParaView workflow scripts. The central idea is to provide an environment when a user provides a serial script and some parameters as inputs to the spatio-temporal environment which then will run the script in parallel and produce the desired result.

Create Spatio-Temporal Pipeline Scripts in the ParaView GUI

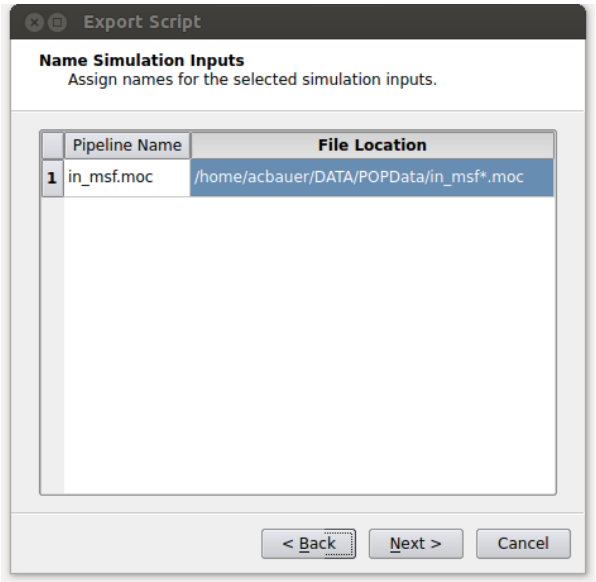
The goal of using spatio-temporal parallelism is to reduce the time for analyzing the simulation results. Thus, a key component in the workflow is being able to create and run the scripts quickly and easily. In order to do this, we've developed a ParaView plugin that can generate these spatio-temporal pipeline scripts. In this section we'll walk through the steps to create a script and then finally run it.

The easiest way to create a new spatio-temporal pipeline script is to use ParaView's GUI plugin. The directions are as follows:

1. From the UV-CDAT install directory, run ParaView with the Externals/bin/paraview command.
2. In the GUI, go to the Tools menu and select Manage Plugins. In the pop-up dialogue window, select TemporalParallelismPlugin and click on Load Selected. At this point it should list the plugin as loaded. To exit this dialog, click on Close.
3. Now, go through and create a pipeline to process the data as desired. The key point here is that any data files or images that we want to have output from our script needs to be included. For data files we don't need to store a local version, we just need to specify how to save the data when we run the script on the desired machine with the desired input files. We do this by creating a proxy to the writer that we want to use and specifying a file name to write to. Note that as we expect to write one file per time step we add in a wildcard "%t" which gets replaced with the time step index when creating the file. The steps to do this in the GUI are:
    a. Go to the Writers menu and select an appropriate writer. Writers that cannot be used with the data set will be grayed out and not accessible.
    b. Make sure that the writer proxy is highlighted. This is shown in the figure below where ParallelUnstructuredGridWriter is selected in the Pipeline Browser.
    c. In the Object Inspector, enter the name of the file. Note that %t will be replaced by the corresponding time step for each output file. In the figure below the output file name is popoutput_%t.pvtu. Note that the transform filter is used to obtain a better visual representation of the data but we would prefer to save the data in its original unscaled form.
4. At this point our script will read in a set of files, threshold out bad values (i.e. POP values that are set indicating that there is land) and write a file that contains that information. We may also want to output images based on our available view windows. This is set when we output the script. The steps to generate the script are:
    a. Click on the TemporalParallelism menu item and select Export State.
    b. This pops up a wizard to guide the user through exporting the spatio-temporal script. For the first window, click Next.
    c. At this point the window will look like the Figure below. It's possible that the user could have read in multiple files and this menu gives them the option of selecting which files they want to process in their spatio-temporal pipeline. Typically this will be a single input (in_msf.moc in this example) and the user can either double click on it or select it and click on the Add button. After this is done, click on the Next button.

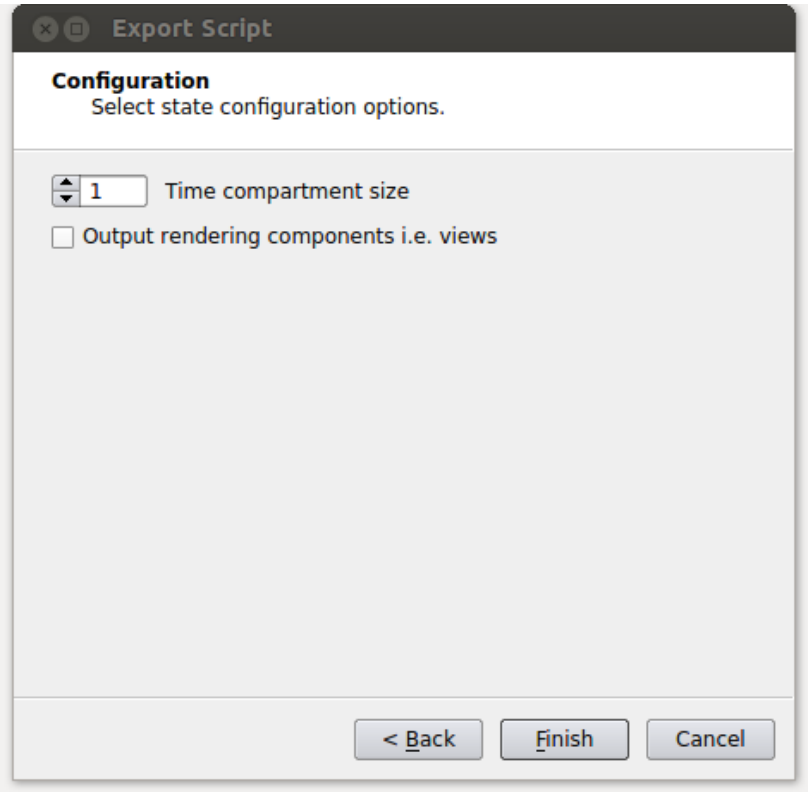


d At this point, we need to specify where the input is coming from on the machine we'll be running the script on and what that input corresponds to with our local pipeline. For our example, we want the in_msf.moc source in our pipeline to read in files located at "/home/acbauer/DATA/POPData/in_msf*.moc". The wildcard is included as we have multiple files to be read (e.g. in_msf0.moc, in_msf1.moc,...) for the machine we'll be running on. After this is done, click on the Next button.



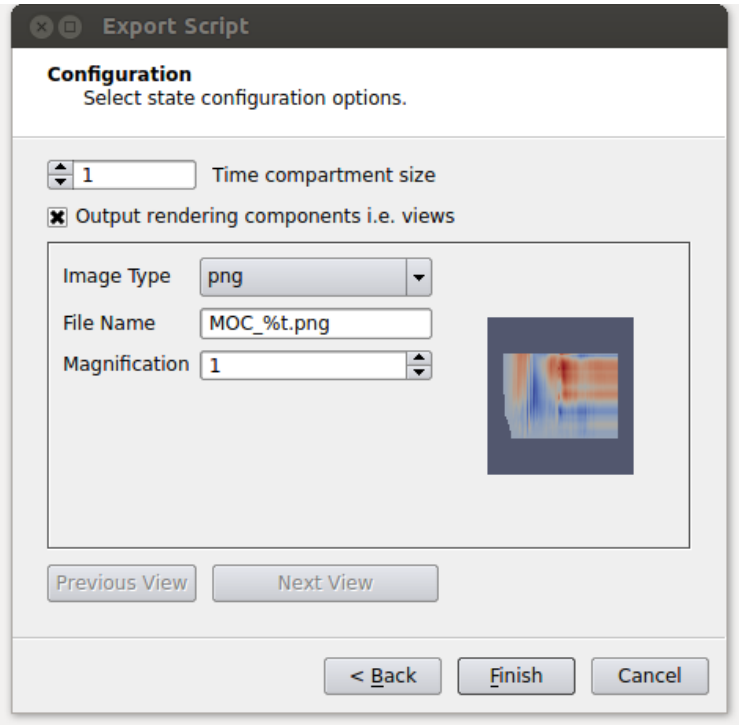
e At the next window, we specify the time compartment size to be used in our run. Setting a time compartment size of 1 results in each process working independently on separate time steps. A time compartment size of 1 will work well when the amount of data per time step is small. For typical supercomputers though, the amount of memory per core is relatively small such that we will also need to spatially decompose the data in order to efficiently use each core while also not running out of memory on the node. In this case the time compartment size should be the number of cores on a node

or some multiple of that value as the data size per time step increases. This is shown in the figure below



f If an image is desired for the output, this is also where that needs to be specified. By enabling the “Output rendering components i.e. views” option, it will allow the user to set parameters for output each view in the script. The parameters are image type, the name of the file where again %t will be replaced with the corresponding time step and a magnification factor. In our example we only have a single view but if there were more the user could

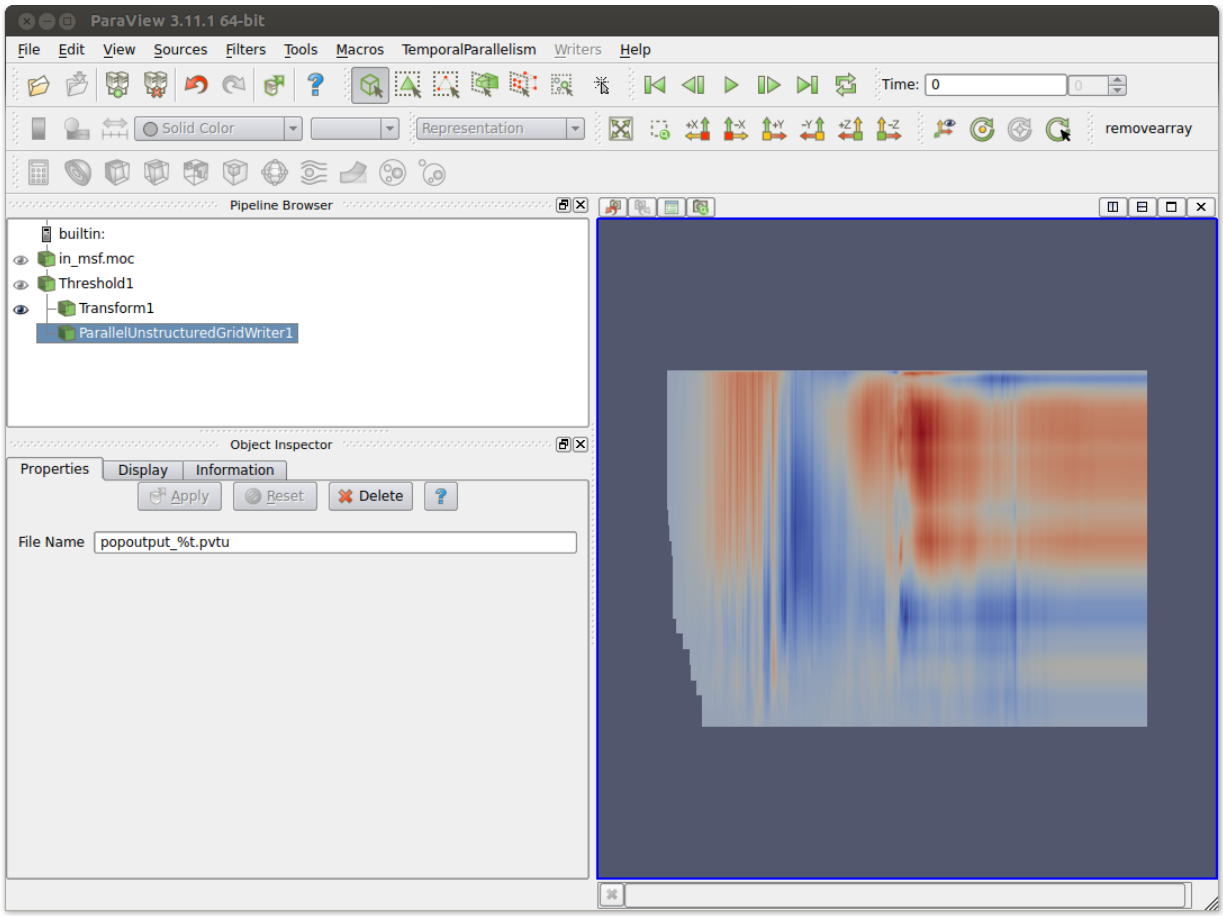
toggle through the views by clicking on the Previous View and Next View buttons.



g After this has been done, click on the Finish button which will open a dialogue to specify the name and location to save the spatio-temoral pipeline script.

Run the Spatio-Temporal pipeline script

Now that the script has been created, the user will need to transfer the script to the machine it is intended to be run on. If the path to the input and output files aren't known *a priori* and set in step 4d above, they may need to be changed to match where it is found on the supercomputer. As there are many different ways to run a program built with MPI (e.g. mpirun, mpiexec, aprun, etc.), the user will have to look at the machine documentation to figure that out. We assume here that mpirun is being used in which case "mpirun -np 32 <install path>/Externals/bin/pvbatch --symmetric <name and location of script>" is used to run the script. Performance on the leading edge supercomputers can be very tricky. Make sure that the proper queues are used as well as proper environment settings. In addition, run times can vary by a factor of 2 or more due to resource contention with other running jobs.



Computing Temporal Statistics

Currently the spatio-temporal scripts that are generated through ParaView's GUI plugin don't share information between time steps. Thus, filters that depend on information from multiple time steps won't work. An example of this is the Temporal Statistics filter. For climate data analysis, computing temporal statistics can be very useful but also very time consuming for large amounts of time steps. Because of this, we have developed a temporal statistics filter that can take advantage of spatio-temporal parallelism. The statistics that are computed are average, minimum, maximum and standard deviation. In addition, several climatologies are also supported. An example script that does this is shown below:

```
try: paraview.simple
except: from paraview.simple import *
paraview.simple._DisableFirstRenderCameraReset()

reader = NetCDFReader( \
  FileName=['/home/kitware/DATA/V_cam5.1994.nc', \
            '/home/kitware/DATA/V_cam5.1995.nc', \
            '/home/kitware/DATA/V_cam5.1996.nc'] )
reader.Dimensions = '(lev, lat, lon)'
```

```
MultiBlockTemporalStatistics1 = MultiBlockTemporalStatistics()
MultiBlockTemporalStatistics1.TimeStepType = 'Months'
MultiBlockTemporalStatistics1.SamplingMethod = 'Consecutive'
MultiBlockTemporalStatistics1.TimeSpan = 'Year'
MultiBlockTemporalStatistics1.TimeStepLength = 1
MultiBlockTemporalStatistics1.TimeCompartmentSize = 8

writer = XMLMultiBlockDataWriter()
writer.FileName = "stats.vtm"
writer.UpdatePipeline()
```

The script is rather simple in that it reads in a set of files (/home/kitware/DATA/V_cam5.1994.nc, /home/kitware/DATA/V_cam5.1995.nc and '/home/kitware/DATA/V_cam5.1996.nc), sets the temporal statistics to compute, and then writes out the data to a file called stats.vtm in the current directory. The parameters for each of the filters are listed below.

- NetCDFReader -- the file reader
    - FileName -- a list of strings for the names and locations of the files to be read in. Note that data from multiple time steps can be stored in each input data file.
    - Dimensions -- the NetCDF dimension variables that are used for specifying the grid as well as the attributes defined over the grid (e.g. fields such as velocity, temperature, etc.)
- MultiBlockTemporalStatistics -- the filter that computes temporal statistics
    - TimeStepType -- whether the time step is in days (‘Days’) or months (‘Months’). The default is ‘Days’.
    - SamplingMethod -- specifies how to group the statistics. ‘Climatology’ means that all of a type are grouped together (e.g. all Februaries), ‘Consecutive’ means a specific time period (e.g. February 2012). The default value is ‘Climatology’.
    - TimeSpan -- specifies what span of time to average together. The options are:
        - AllTimeSteps -- average over the entire set of time steps (default)
        - Month -- average over months
        - Season -- average over the standard seasons
        - Year -- average over each year
        - Decade -- average over each decade
    - TimeStepLength -- the number of days or months in a time step. The default value is 1.
    - TimeCompartmentSize -- the number of processes working together on a specific time step.
- XMLMultiBlockDataWriter -- the writer used to save the data. Note that the files will get written out in VTK’s XML multiblock data set format. The proper file extension is ‘.vtm’.

- FileName -- specifies the name to be used for storing the data. This can also include a relative or absolute path.
- UpatePipeline() -- the method used to execute the computation.

Support

For support with either installing, using this software and/or creating spatio-temporal scripts, please email uvcdat-support@llnl.gov.