Vapor Data Conversion for High Performance Parallel Systems
National Center for Atmospheric Research
Scott Pearse
July 21, 2015

Intro
Vapor now includes scripts that help users convert their data in parallel on computer clusters. Different schedulers (such as SGE, LSF, and SLURM) will have their own scripts to run. The scripts must be edited as described in this document before they can be used.

The scripts can be found in the *share/examples/parallelDataConversion* directory of $VAPOR_HOME.

Prerequisites

- *Change the "vaporBinDir" variable in vaporSGE.py, vaporLSF.py, or vapor***.py*

Before these parallel scripts can be run, they need to know where to find Vapor's binary path ($VAPOR_HOME/bin). This is applied in vaporSGE.py and vaporLSF.py through a global variable called "vaporBinDir". It needs to point at the directory that contains both the data converison routine to be run (wrf2vdf, mom2vdf, etc.), as well as the script named "vapor-setup.sh". The script for your environment (SGE or LSF) must be edited, and the value of the variable **vaporBinDir** must be changed appropriately.

If you wish to copy a batch script to a different location, make sure that the python script *vaporBatchFuncs.py* gets copied along with it, as this is a container for the data conversion dependency functions.

Usage
The scripts work on any of Vapor's data conversion routines such as wrf2vdf or grib2vdf. The data files to be converted are divided among a number of tasks (defined by the user), and then launched as an "array job" with each task operating on its subset of files. Please note the order of the arguments, and the delimiting ":" that separate them.

**Please note that the flags -numts and -startts should not be used in a parallel job submission. Doing so will result in adverse effects in the data conversion.*

[scheduler options] parallelScript vaporTool : [vaporToolFlags] [-files fileList] : [inputFiles vdfFile]

[scheduler options]	Options taken by the scheduler (SGE, LSF, etc) to set up the environment for the array job.
parallelScript	This simple script is tailored to user and scheduler settings. It uses these

	settings to allocate input files to individual tasks, and then launch Vapor's conversion routine on those files. (vaporSGE.py and vaporLSF.py are examples of the paralleScript argument)
vaporTool	One of Vapor's data conversion tools, such as wrf2vdf, grib2vdf, etc.
[vaporToolFlags]	Vapor's data conversion tools each have their own set of options that can be applied for data conversion (such as -numts to specify the number of timesteps to convert). These options may be also be applied here to be used by the parallel script.
[-files, fileList]	Users can specify a file that contains a list of data-files they want to perform their conversion on. If used, the .vdf file needs to be included as well. The file names in fileList must be delimited by newlines.
[-grid, gridFile]	For ROMS conversions only - ROMS conversions each require a grid file to specify the geospatial properties of the data to be converted. Specify the ROMS grid file here.
[inputFiles, .vdfFile]	If the -files flag is not applied, users may list their data files on the command line

Examples

SGE

qsub -t 1-4 ./vaporSGE.py wrf2vdf : -files wrfFiles.txt :

qsub -t 1-4	"qsub" informs SGE of the incurrent batch job, and "-t 1-4" indicates that this is also an array job that will be split into four tasks.
vaporSGE.py	This is the script that will allocate user files among the four tasks that were defined.
wrf2vdf	The vaporTool being used for data conversion.
-files wrfFiles.txt	In this case, we provide a list of files that contains our .vdf metadata file, as well as the wrf data files for conversion.

*qsub -t 1-4 ./vaporSGE.py wrf2vdf : -vars U:V:W : katrina.vdf wrfout_d02_2005-08-31_0**

Arguments used in the previous example have been omitted

-vars U:V:W	Only the variables U, V, and W within the WRF dataset will be converted in this batch job submission.

*katrina.vdf wrfout_d02_2005-08-31_0**	Instead of providing a file that lists our target data, we can list it on the command line after the second ":". Note that using wildcards can cause undesirable effects if there are a large number of files in a directory. It is recommended to use -files flag when possible.

LSF

bsub -W 00:60 -J "myJob[1-4]" -n 4 ./vaporLSF.py wrf2vdf : -files wrfFiles.txt :

bsub -W 00:60 -n 4 -J "myJob[1-4]"	*bsub* informs LSF of the incurrent batch job. *-W 00:60* assigns a wall clock time of 60 minutes per task. *-n 4* specifies the number of nodes to be used, and *-J "myJob[1-4]"* indicates that this is an array job named myJob that will be split into four tasks.
vaporLSF.py	This is the script that will allocate user files among the four tasks that were defined.
wrf2vdf	The vaporTool being used for data conversion.
-files wrfFiles.txt	In this case, we provide a list of files that contains our .vdf metadata file, as well as the wrf data files for conversion.

bsub -W 00:60 -J "myJob[1-4]" -n 4 ./vaporLSF.py
*wrf2vdf : -vars U : myWRF.vdf wrfout_d02_2005-08-31_0**

Arguments used in the previous example have been omitted

-vars U	Here we applied one of the data conversion tool's options, -vars. This means that our parallel conversion will only process the variable U, skipping all others in the supplied data.
*katrina.vdf wrfout_d02_2005-08-31_0**	Instead of providing a file that lists our target data, we can list it on the command line after the second ":". Note that using wildcards can cause undesirable effects if there are a large number of files in a directory. It is recommended to use -files flag when possible.

***Note that LSF may require a project code to associate jobs with users. The project code is specified with the "-P" flag.

Adapting for New Schedulers

These conversion scripts can be easily adapted for other schedulers by re-defining three global variables called *first, last, and myId*.

In an “array job”, each task that gets spawned receives its own set of environment variables that are set up by the scheduler. Among these environment variables are the three necessary for Vapor’s parallel scripts to allocate the user’s data files (*first, last, and myId)*.

All that needs to be done to expand these scripts to a new scheduler is the following:

1) Find the three scheduler-defined environment variables that identify the first task, last task, and current task’s numeric ID.
2) Apply these three environment variables into Vapor’s parallel script as the variables *first, last,* and *myId*.*
3) Ensure that the task ID’s are indexed starting at zero.

** If a scheduler does not have environment variables that define the first and last task numbers then they can be hard coded according to the user’s needs. However myId must always be defined as the current task number in the set.*