Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2367 – 2375

Procedia Computer Science

2013 International Conference on Computational Science

ParCAT: Parallel Climate Analysis Toolkit

Brian Smith[a*], Daniel M. Ricciuto[a], Peter E. Thornton[a], Galen Shipman[a], Chad A. Steed[a], Dean Williams[b], Michael Wehner[c]

[a]Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge TN 37830
[b]Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550
cLawrence Berkeley National Laboratory, 1 Cyclotron Rd. Berkeley, CA 94720

Abstract

Climate science is employing increasingly complex models and simulations to analyze the past and predict the future of Earth's climate. This growth in complexity is creating a widening gap between the data being produced and the ability to analyze the datasets. Parallel computing tools are necessary to analyze, compare, and interpret the simulation data. The Parallel Climate Analysis Toolkit (ParCAT) provides basic tools to efficiently use parallel computing techniques to make analysis of these datasets manageable. The toolkit provides the ability to compute spatio-temporal means, differences between runs or differences between averages of runs, and histograms of the values in a data set. ParCAT is implemented as a command-line utility written in C. This allows for easy integration in other tools and allows for use in scripts. This also makes it possible to run ParCAT on many platforms – from laptops to supercomputers. ParCAT outputs NetCDF files so it is compatible with existing utilities such as Panoply and UV-CDAT. This paper describes ParCAT and presents results from some example runs on the Titan system at ORNL.

Keywords: Parallel Climate Analysis ParCAT

1. Introduction

Climate science is employing increasingly complex models and simulations to analyze the past and predict the future of Earth's climate [1,2]. This growth in data is creating a widening gap between the volumes of data being produced and the tools necessary to analyze the large, high dimensional datasets. Single model runs generate tens and hundreds of gigabytes of data for relatively small time scales. Extended time model runs have the potential to generate terabytes of data. Multiple model ensemble runs for comparisons between models are

* Corresponding author. Email address: smithbe@ornl.gov

doi:10.1016/j.procs.2013.05.408

also common. These ensemble runs can generate terabytes of data as well. Because of the explosion of data, parallel computing tools are becoming a necessity to analyse, compare, interpret, and verify the simulation data. Typical preliminary analysis operations are things like spatio-temporal averages, differencing two model runs, and generating frequency histograms.

Spatio-temporal averages are useful for "sanity checking" a given run. For example, it is expected that the warmest temperatures will be found in the summer months in the northern hemisphere. If an average of the ground temperature for a multi-year model run shows the warmest temperatures in the northern hemisphere in the summer months, the scientist can be reasonably certain that at least the temperature was calculated correctly by the model. Similarly, frequency histograms can be also used to check a given model run. It is expected that the highest temperatures will be clustered around the tropics so the distribution of higher temperatures should be skewed towards the tropics as well. Likewise, if scientist wants to see the effect of an increase in CO2 in the atmosphere, differencing a control run and the experimental run and looking at temperature might be a good place to start.

Fortunately, these operations are theoretically "trivial parallel" in multiple dimensions. For example, calculating a spatio-temporal average map is parallelizable in time, space, and per variable. Some variables even add a fourth parallelizable dimension – height or depth. For many analysis operations, all of those dimensions are independent. A given set of nodes could operate on a single variable and single time step calculating averages in small regions, such as continents or evenly divided ranges of latitude and longitude. Alternatively, a given set of nodes could operate on multiple variables where each process calculates the difference of two model runs for a given time stamp over the entire range of latitude and longitude. Taken to an extreme, a parallel analysis toolkit could parallelize calculations over latitudes, longitudes, elevations, variables, and times all over multiple input files. However, because these calculations require a significant amount of I/O (relative to compute), such extremes are unlikely to yield excellent results.

Because many clusters, supercomputers, and even desktop computers have multiple cores per processor, it makes more sense to limit the parallelism to two dimensions – one spread over a node and one spread over the cores. For example, it might make sense to have each node read files in parallel and each core on the node process variables in parallel out of the files read by the node.

This paper describes a new utility called ParCAT that was developed from a collaborative effort of climate scientists and computer scientists to facilitate parallel data analysis. ParCAT provides the ability to perform spatio-temporal averages, differencing multiple datasets, and generating histograms with algorithms that take advantage of the "trivially parallel" nature of the operations while respecting realistic computing architectures and I/O systems. Results from some sample runs on a high-end supercomputer (ORNL's Titan) are provided.

2. Parallel Climate Analysis Toolkit (ParCAT)

The Parallel Climate Analysis Toolkit (ParCAT) provides basic tools to efficiently use parallel computing techniques to make analysis of large datasets manageable. Because of the collaborative effort, the tools are highly focused on the needs of climate scientists. ParCAT targets model output generated by the Community Climate System Model (CCSM) [1], and has specifically been focused on land (CLM) and atmospheric (CAM) model runs. However, the framework is meant to be fairly flexible and could conceivably provide parallel analysis tools for any large NetCDF dataset.

Currently, ParCAT provides the ability to compute spatio-temporal means, differences between runs or differences between averages of runs, and histograms of the values in a data set. The ParCAT framework also

tries to make it fairly easy to add additional functions that can be parallelized in space, time, or per variable.

ParCAT is designed as a backend tool for processing large, multidimensional datasets. The toolkit does not focus on performing the final visualizations or interpretation of the data, but rather it can reduce the large datasets to smaller, more manageable summaries for visualization, analysis, interpretation or even presentation in other tools. ParCAT's focus is on datasets with many variables or large number of timesteps rather than extremely fine spatial resolution. The toolkit is implemented as a command-line utility written in C and utilizing parallel NetCDF (pNetCDF) [3] and MPI [4]. This allows for easy integration in other tools (e.g. UV-CDAT [5] or EDEN [6]) and allows for scripting (e.g. recreating much of the NCAR model diagnostics [7]). Because ParCAT is a command-line utility written in C with fairly common libraries, it is also possible to run it on any platform from a laptop to a supercomputer such as ORNL's Titan system.

ParCAT processes NetCDF files as input and provides NetCDF files as output. Output files from ParCAT are therefore compatible with existing analysis tools, such as Panoply or UV-CDAT, and libraries that process NetCDF files in C, Python, Perl, etc.

2.1. ParCAT Operation

The minimal invocation requires a path to the files to process and the desired operations to perform. A more typical invocation specifies a subset of time (years/months), a list of variables of interest, an output file name, and a hint for the type of dataset (CLM or CAM are currently supported). For functions that take multiple input file sets (e.g. difference) a second path is required. A second set of time specifiers is also allowed. This allows things like a comparison between summer and winter months in one model run or between two separate models.

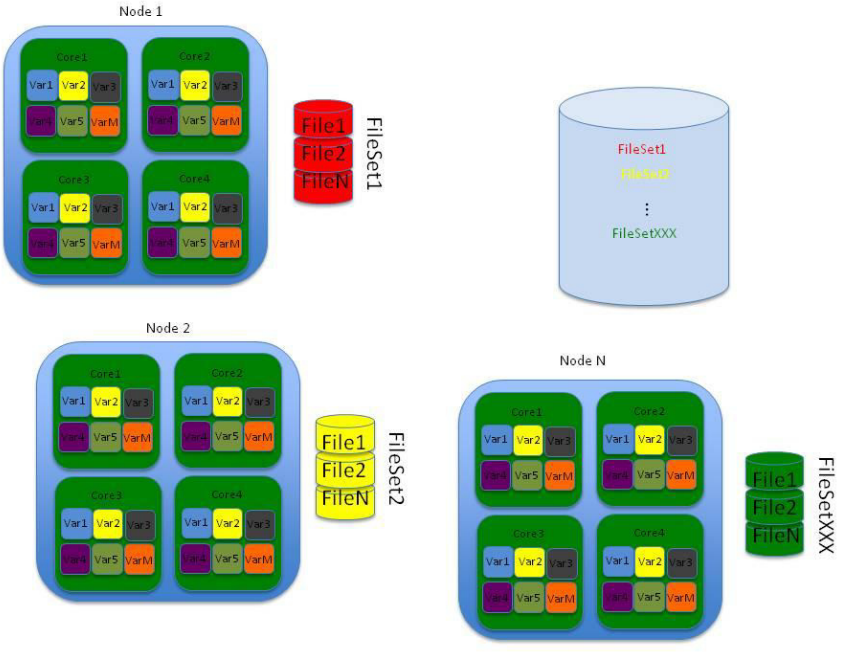
Currently, ParCAT supports three different types of model output:

- One timestamp per file with multiple variables per file. The CLM sample dataset used in section 3 is structured this way. The filenames have a regular pattern similar to rundescription-{year}-{month}.
- One variable per file with multiple timestamps per file. The CAM sample dataset is structured this way and was postprocessed from output similar to the CLM sample dataset by a widespread community practice. The filenames have a regular pattern such as {variable name}rundescription-{start year}-{end year}.
- An arbitrary listing of files and variables

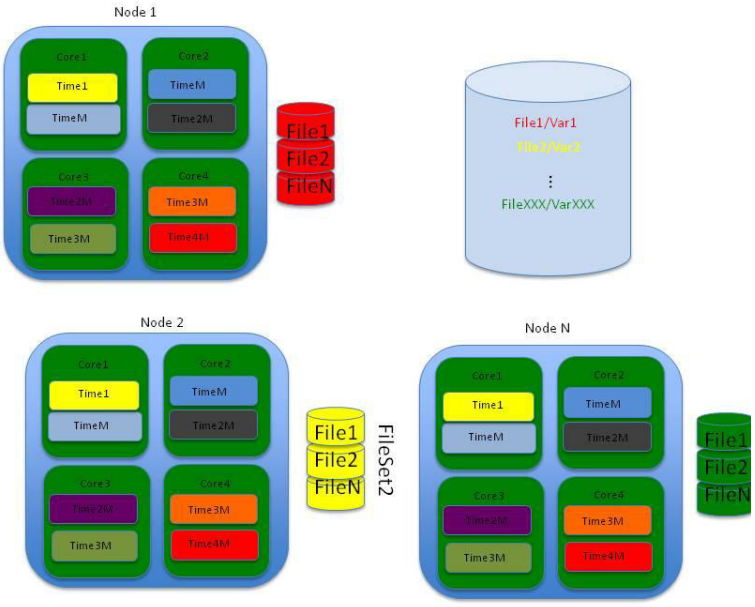
The first case will be referred to as "CLM" in this paper and the second case will be referred to as "CAM". However, CCSM allows the user to specify how output is generated independent of the model being run. As discussed earlier, even though the operations ParCAT supports are trivially parallelizable performance is unlikely to improve with more than one or two levels of parallelism given the I/O-bound nature of data processing. ParCAT supports up to two parallel dimensions when processing CAM and CLM-like datasets and one parallel dimension for an arbitrary listing of files and variables.

For CLM datasets, ParCAT distributes one or more files to each node in parallel. Every core on the node processes one or more variables in the file(s) the node has opened. This is shown in Figure 1. For the sample runs in section 3 there are twenty years of data with monthly snapshots (120 total files) and 349 variables. If there are four cores per node and thirty nodes, each node will open four different files. On each node, each core will process about eighty-eight of the 349 variables contained within each file. Once the data is read in on each core, the cores combine their data to get a local sum per variable on the node. This would be the sum of the data in the four files the node processed. After this is done, all nodes perform an allreduce operation. Every

node now has the total sum of all data points for all variables. One “master node” can have each of its cores write out a section of the output file. As the number of files increases, the number of nodes ParCAT can utilize also increases. As the number of variables increases, ParCAT can parallelize work among more cores.



For CAM datasets, ParCAT distributes one or more files (each of which contains one variable) to each node. Each core handles a subset of the timestamps within the file. This is less parallelizable than CLM datasets because the number of variables (and, thus files) in CAM runs tends to be smaller. However, the larger individual files can provide some advantages as discussed in section 3. This setup is shown in Figure 2. For the sample dataset utilized in section 3, there were twenty-eight variables and 624 timesteps. If there are four nodes, each node processes seven files (which is therefore seven variables). If there are sixteen cores, each core would processes thirty nine time steps.



For arbitrary file lists, ParCAT distributes one file to each node. There is no further parallelization.

2.2. ParCAT in other applications.

Because ParCAT is a command-line utility, other applications can make use of the ParCAT functionality. Currently, ParCAT support is enabled in the EDEN application and ParCAT has been used to dramatically speed up portions of the NCAR Land Diagnostics Package.

The Exploratory Data analysis Environment (EDEN) is a robust visual analytics framework that fosters interactive visual queries for tackling big data challenges in climate science. EDEN features a multi-faceted filter panel as well as a highly interactive visual data analysis canvas that integrates scatterplots, a correlation matrix, and a geographic scatter plot. Once a user has selected one or more datasets and specific years, months, and variables of interest, a real-time spatio-temporal average map is generated using ParCAT. This enables the user to determine quickly if the specified parameters seem viable for further study. An example screen shot is shown in Figure 3. This map was generated by EDEN running ParCAT after the user selected a time series and variable of interest. Additional functionality in EDEN might eventually be offloaded to ParCAT as well.

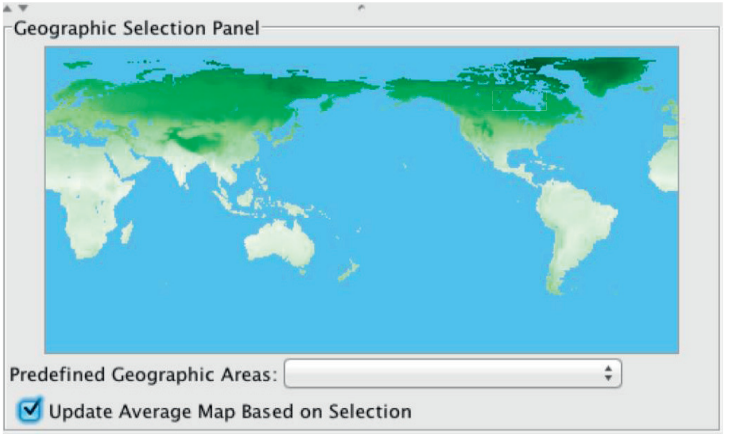


Figure 3 ParCAT generated average map from user-selection parameters inside EDEN

The NCAR Land Diagnostics Package can be used to compare two model simulations or compare a model simulation to observational data. The diagnostics package produces postscript plots of long-term trends and seasonal means, in regional, global, and globally averaged formats from CLM files. The postscript files are converted to .gif files and packaged in a web page for easy reference. As a proof of concept, several pages of the diagnostics were recreated with ParCAT output, a Python script utilizing pupynere (a NetCDF file reader for Python) and gnuplot. Results can be obtained very quickly after ParCAT has summarized model output.

3. Example Results

Two typical datasets were analyzed with ParCAT, one CLM and one CAM. The CLM dataset was a fifteen-year simulation with monthly snapshots (180 time steps total). There were 349 variables and the spatial resolution was ½ degree. The total size of the dataset was approximately 71GB. The CAM dataset was the result of a fifty-two year simulation with monthly snapshots (624 time steps total). Raw CAM output consisting of files containing all output variables for one snapshot time step was postprocessed into the CMIP5[8] file standards of one variable per file containing all 624 time snapshots. There were twenty-eight variables and the spatial resolution was one degree. The total size of the dataset was approximately 18GB. Both the CLM and CAM datasets are structured on a regular latitude-longitude horizontal grid. The ParCAT results were obtained with the Titan[9] supercomputer at Oak Ridge National Labs, a large Cray XK7 installation. None of the GPUs were available at the time of the runs, but ParCAT is not setup to take advantage of them anyway. Titan is connected to the Spider Lustre-based filesystem[10] which is a high performance parallel file system. Several other users were running on Titan at the same time the results were obtained, so each ParCAT run was done three times and the results are averaged. All runs are using the 'average' function of ParCAT and are for all variables (349 for the CLM dataset, twenty eight for the CAM dataset). Additionally, in an effort to minimize possible caching effects, sequential runs changed the number of nodes with the number of cores fixed, so each node (except node 0) should get different files to process. Lustre has a number of tunable parameters based on file size and access patterns. However, none of the parameters were adjusted for these runs. The Spider system is tuned for files larger than the individual NetCDF data files, so some further performance improvement might be achievable.

At this time, ParCAT uses MPI for the intranode communication (as opposed to OpenMP or shared memory operations). This is because p-NetCDF shares file access via an MPI communicator. This also makes it easy to perform the collective operations (e.g. MPI_Allreduce) required for gathering node-wide results. However, the MPI standard provides no general, portable way to determine or guarantee how MPI processes are arranged on the nodes and cores of a given machine. The Cray runtime environment allows the user to specify how processes are arranged on the nodes but ParCAT has more generalized routines to try to determine node layout.

One trick that can be used on Titan and many clusters is to look at the process's hostname. MPI processes with the same hostname are likely multiple cores on a node. This trick works well on many machines. MPI 3.0 adds an MPI_Comm_split_type() function call which can split a communicator into subcommunicators that can create shared memory regions. Once MPI 3.0 is more widespread, this function call should allow for a more portable method of identifying cores on the same node.

Figure 4 shows results from processing the CLM dataset. The overall trend is as expected. More processing resources decreases runtime, and utilizing multiple cores to process variables helps as well. The times do not decrease linearly with increasing processor counts. This suggests that the problem is I/O bound, which is also somewhat expected given the quantity of data involved and the minimal amount of processing required. The results suggest that four processes per node is probably the best utilization given this dataset. ParCAT tries to minimize memory usage to ensure it can run on systems with minimal memory per core. On large memory nodes like those on Titan, performance might be improved by increasing memory usage. This could enable larger contiguous reads from the file system before processing data.

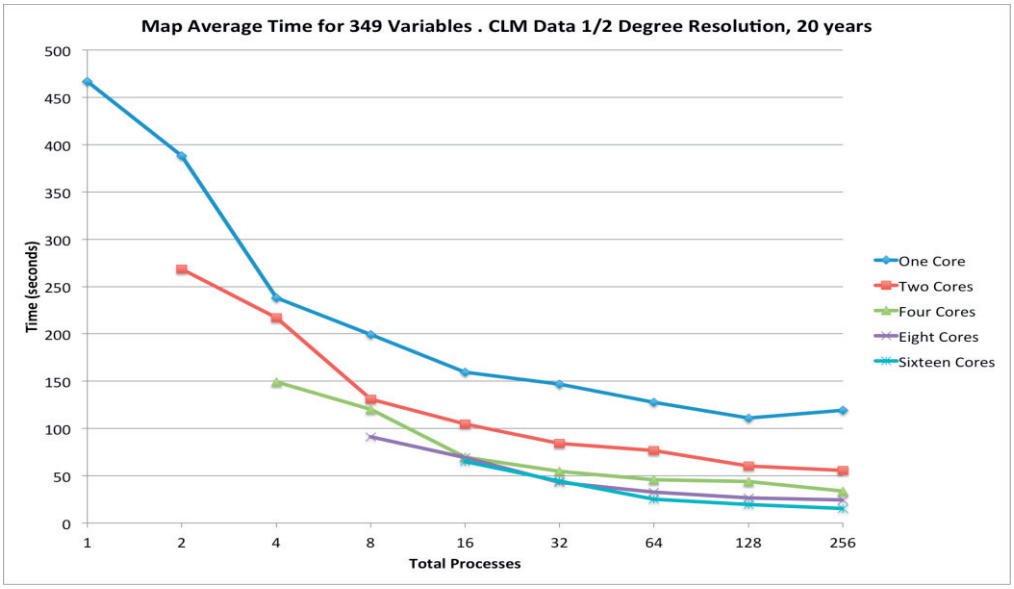


Figure 4 Map averaging time for CLM dataset

Figures 5 and 6 show run times for the postprocessed CAM dataset. Because each file contains only one variable, there is a sequential step to open each file, copy the dimensions to the output file, and setup the variable in the output file. This step is fairly expensive (approximately 60 seconds for most runs). However, the actual processing time (calculating the average among the participating cores) is lower per time step than for the one-file-per-time-step CLM dataset. This is likely because some of the CAM files are much larger than individual CLM files. Additionally, ParCAT uses a strided, non-blocking read function in pnetCDF. Reading strided data from these larger files is likely more efficient than reading blocks from individual files. Raw CAM output, consisting of files with all variables but only one snapshot, would be laid out in a similar manner as the

CLM output considered here, although larger in size due to a larger vertical dimension. It would be interesting to see individual variables per file CLM output or individual timestamp CAM output and see how the results compare to these sample datasets. It is also interesting to see that overhead increases when the processing is first distributed among multiple cores, then tapers off as the number of cores increases. There are likely optimizations in the pnetCDF implementation to minimize overhead when there is no parallel file access (e.g. fewer locks might be needed).

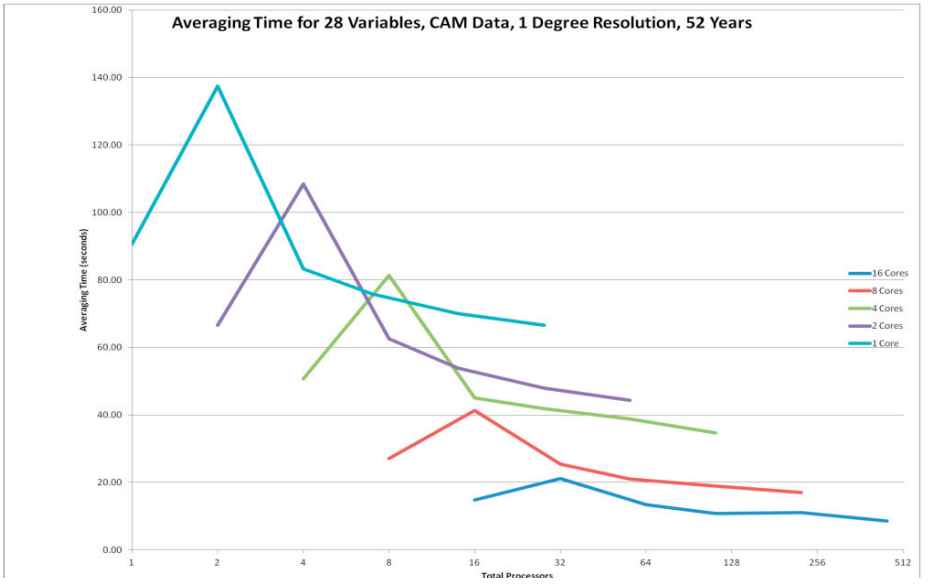


Figure 5 - Averaging calculation time for CAM Dataset

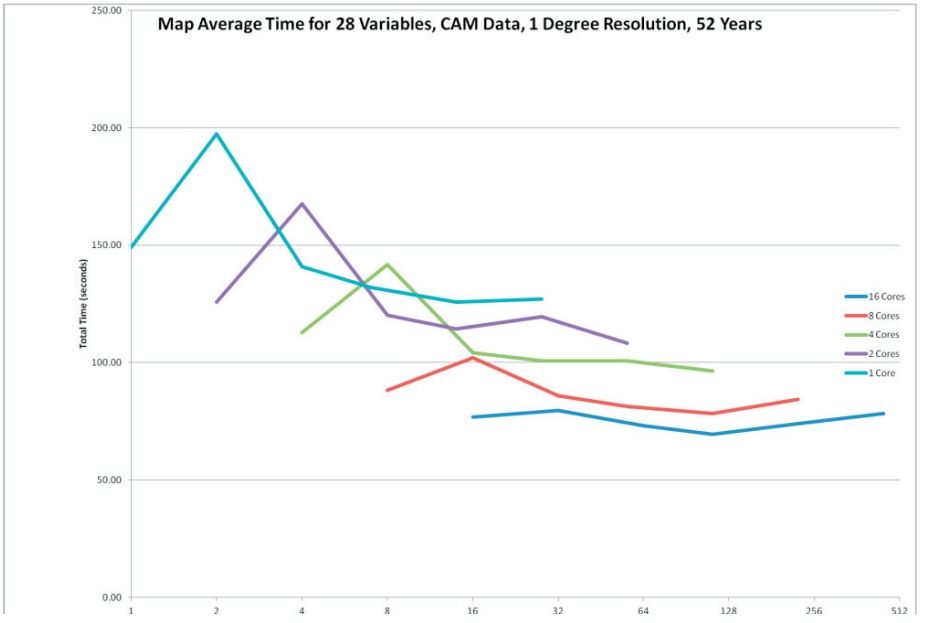


Figure 6 - Total time (setup plus averaging calculation) for CAM Dataset

4. Conclusions and Future Work

With the rapid increase in complexity of climate models, both from finer resolution simulations and more in-depth models comes a significant increase in output dataset sizes and complexity. This increase means parallel tools are now a requirement for meaningful analysis of the data. ParCAT provides basic statistical analysis tools to help accelerate discoveries and pursue new scientific inquiry in climate science. ParCAT can be embedded in existing applications or added to an existing workflow. It can run on ultra-high end supercomputers and basic desktop machines. Additional functionality can be added easily to expand the capabilities as well.

In the future, a Python wrapper or Python interface would increase the usefulness of ParCAT in many existing analysis tools. Releasing ParCAT to the open source community is also a goal. Additional parallelizable functions that are useful to the climate science community should also be explored.

Acknowledgements

This work is sponsored by the Office of Biological and Environmental Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. This research used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Thanks to Dave Dillow at Oak Ridge National Laboratory for discussing the Spider file system implementation and suggesting future optimization options.

References

[1] P.R. Gent, G. Danabasoglu, L.J. Donner, M. M. Holland, E.C. Hunke, S.R. Jayne, D. M. Lawrence, R. B. Neale, P.J. Rasch, M.Vertenstein, P.H. Worly, Zong-Liang Yang, and M. Zhang, "The Community Climate System Model Version 4", Journal of Climate, vol. 24, no. 19, pp. 4973-4991, 2011

[2] J.T. Overpeck, G.A.Meeghi, S.Bony, and D.R. Easterling, "Climate data challenges in the 21st Century", *Science* vol. 331, no 6018, pp 700-702, 2011.

[3] Jianwei Li, Wei-keng Liao, Alok Choudhary, Robert Ross, Rajeev Thakur, William Gropp, Rob Latham, Andrew Siegel, Brad Gallagher, and Michael Zingale. Parallel netCDF: A Scientific High-Performance I/O Interface. In the *Proceedings of Supercomputing Conference*, November, 2003.

[4] The Message Passing Interface Forum http://mpi-forum.org

[5] Ultrascale Visualization – Climate Data Analysis Tools http://uv-cdat.llnl.gov

[6] Chad A Steed, Galen Shipman, Peter Thornton, Daniel Ricciuto, David Erickson, and Marcia Branstetter. Practical Application of Parallel Coordinates in Climate Model Analysis. In *Proceedings of the International Conference on Computational Science,* pp 877-886, June 2012

[7] CCSM Land Model Diagnostics Package – http://www.cgd.ucar.edu/tss/clm/diagnostics/index.htm

[8] Taylor, K.E., R.J. Stouffer, and G.A. Meehl, 2012: The CMIP5 Experiment Design. *Bull. Amer. Meteorol. Soc.,* **93**, 485—498, doi: 10.1175/BAMS-D-11-00094.1

[9] Titan - https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/

[10] G. Shipman, D. Dillow, S. Oral, and F. Wang. The Spider Center Wide File System; From Concept to Production Deployment. In *Proceedings of the Cray Users Group Meeting*, 2009