https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020
© Author(s) 2020. CC BY 4.0 License.

Geoscientific Model Development
Discussions
Open Access

A New End-to-End Workflow for the Community Earth System Model (version 2.0) for CMIP6

Sheri Mickelson, Alice Bertini, Gary Strand, Kevin Paul, Eric Nienhouse, John Dennis, and Mariana Vertenstein

The National Center for Atmospheric Research, Boulder, CO, USA

Correspondence: Sheri Mickelson (mickelso@ucar.edu)

Abstract. The complexity of each Coupled Model Intercomparison Project grows with every new generation. The Phase 5 effort saw a large increase in the number of experiments that were performed and the number of variables that were requested compared to its previous generation, Phase 3. Many centers were not prepared for the large demand and this stressed the resources of several centers including at the National Center for Atmospheric Research. During Phase 5, we missed several deadlines and we struggled to get the data out to the community for analysis. In preparation for the current generation, Phase 6, we examined the weaknesses in our workflow and addressed the performance issues with new software tools. Through this investment, we were able to publish approximately six times the amount of data to the community compared to the volumes we produced in the previous generation and we were able to accomplish this within one-third of the time, providing an 18 times speedup. This paper discusses the improvements we have made to accomplish this success for Phase 6 and further improvements we hope to make for the next generation.

1 Introduction

The Coupled Model Intercomparison Project Phase 6 (CMIP6) (Eyring et al., 2016) is a large international project that consists of many centers around the world running the same simulations, in order to seek a better understanding of Earth processes under different scenarios. This includes, but not limited to, studying different mitigation strategies, paleo climate analysis, and different land mitigation strategies. Centers commit to running a core (or DECK) set of experiments along with different tiers of experiments that can be compared against the DECK runs. The National Center for Atmospheric Research (NCAR) committed to running most tier 1 experiments from almost all of the different Model Intercomparison Project (MIP) groups. In total, this included running 130 unique experiments with many having multiple ensemble members. This commitment required over one thousand different model runs, simulating over 37,000 years of climate, which consumed over 190 million CPU hours and produced over 2 PB of post-processed data.

During the Coupled Model Intercomparison Project Phase 5 (CMIP5) (Taylor et al., 2012), the post-processing of the data became a large problem for NCAR. During that process, NCAR used the Community Earth System Model (CESM) version 1 (Hurrell et al., 2013) to generate roughly 2.5 PB of raw output in about 18 months. It then took NCAR an additional 18 months to post-process and publish the data. While CESM ran relatively quickly compared to the other climate models that participated, the post-processing took longer to run than at other centers. Due to inefficiencies in both the software and the post-processing workflow orchestration, NCAR was only able to publish about 165 TB of data. To help ease the process of running the CMIP6 experiments and post-processing the data, NCAR invested resources to improve the scientific workflow to ensure everything would be published to the community efficiently. These changes were required to work with the new version of the model, the Community Earth System Model version 2 (Danabasoglu et al., 2020), to be as efficient as possible, and they needed to reduce the burdens caused by running such experiments.

In order to improve our end to end workflow, we needed to focus on three areas. The first step was to improve the performance of the data workflow by creating a set of new tools that would allow us to parallelize each of the operations and streamline the publication process. This work is discussed in Section 2. Second, we needed to automate the process workflow in order to remove the expertise needed to run the different tasks and to have tasks run continuously without intervention. This is discussed in Section 3. Finally, we needed a better way to track simulation progress and document the experiments. The improvements that were made in this area are discussed within Section 4.

2 Data Workflow

The first task in creating a new data workflow for CMIP6 was to evaluate the methods used in CMIP5 in order to find where improvements needed to be made. The life cycle of the data consists of the multiple stages shown in Figure 1. First, the model is ran and raw model output is generated. As the model runs, diagnostics are generated in order to track the simulation's scientific progress. For CMIP5, this was a manual process that was not often done because it was time consuming. When the model run is complete, the raw output is transformed into a time-series format. For CMIP5, this process did not contain any parallelism and it was slow to run because of the amount of data that was required to be post-processed. The time-series formatted data are then used to generate a new set of data that complies to the specific MIP standards. For CMIP5, this was also a time consuming process because it lacked parallelism, it was difficult to run, and it required expert knowledge to ensure the data met the correct standards. After the standardized data are verified, it is then published to the Earth System Grid Federation (ESGF) (Cinquini et al., 2014).

Fundamentally, the post-processing steps involved opening a set of files and reading the data, performing one or more simple operations on the data, and then writing out the results. While the post-processing steps were straight forward, they were very time consuming due to the number of files and total data volume on which they operated. For example during CMIP5, which had data volumes in the several tera-byte range, post-processing calculations would take several days to run for each experiment. For CMIP6, which involved a significantly larger number of files and total data volumes in the peta-byte range, a better solution was needed. In particular we needed tools with flexible interfaces that could write compressed NetCDF files in parallel while minimizing the number of times output files were opened and closed for writing,

There are a number of existing software package that can be used to perform the post-processing steps including; the NetCDF Operators (NCO) (Zender, 2008), the Ultrascale Visualization Climate Data Analysis Tools (UVCDAT) (Williams, 2014), the Climate Data Operators (CDO) (Kornblueh et al., 2019), and Pagoda (Daily, 2013). While these packages provide a diverse

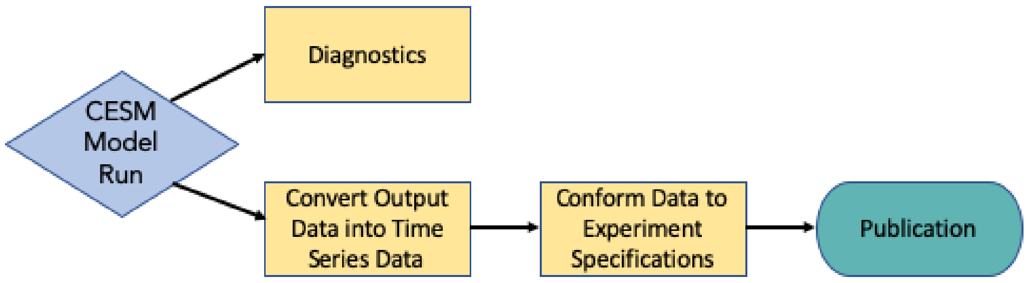
Diagnostics
CESM Model Run
Convert Output Data into Time Series Data
Conform Data to Experiment Specifications
Publication

Figure 1. This flowchart describes the tasks that are executed within the CESM workflow in order to generate data for CMIP. The diagnostics task can be executed several times while the CESM model run is executed. The remainder of the tasks are each executed once when the CESM model run completes.

set of operations, none of them satisfied all of the necessary requirements. For example while CDO minimized the number of times output files were opened and closed it did not easily enable parallel execution. Conversely while Pagoda offered parallel execution it did not minimize the number of open and closes. The XML IO Server (XIOS) (Meurdesoif, 2020) is an IO library that is able to write publication ready output directly from the model. While XIOS provides excellent performance, implementing this method would have required us to rewrite the IO interface within all of the modeling components and this would have taken more FTEs than were allotted for this project. We therefore decided to develop our own tools based on Python and the Message Passing Interface library (MPI) (Gropp et al., 1999) to enable parallelism. We choose to use Python because of its flexibility, available libraries, and quick prototyping ability (Perez et al., 2011; Oliphant, 2007) and MPI4Py (Dalcin, 2019) library to enable parallelism.

The publication of CMIP5 data contributions to the ESGF was also a bottle neck within the data workflow. During CMIP5, the ESGF software stack was stressed when large amounts of data were trying to be published by multiple organizations at the same time. Over the past few years, a team of individuals from around the world have been improving the ESGF software stack (Abdulla, 2019). These process improvements, along with the post-processing tools we developed are described in the following subsections.

2.1 Time Series Generation

The first step within our post-processing workflow involved a transformation of the raw CESM output data from time slice into time series. This operation is represented in the “Convert Output Data into Time Series Data” task within Figure 1. Each of the CESM components produces output files that contain multiple variables in one time slice chunk. Unfortunately this is not an ideal format for distribution, because scientists are typically interested in evaluating a handful of variables at multiple time steps. Instead the data are reformatted into a time-series format, where each file contains one or more time slices of a single variable.

Interestingly, the conversion of time-slice to time-series data was the single most expensive component of the CMIP5 workflow. While this operation is embarrassingly parallel due to the lack of data dependencies between each variable, the serial CMIP5 workflow used individual NCO commands that opened, read and wrote each individual time slice. Consider the number of file operations necessary to convert an entire data-set which contains num_{TS} (number of time slices) and num_{var} (number of variables) from time-slice to time-series. Using the serial CMIP5 workflow, the execution consisted of $2 \times num_{TS} \times num_{var}$ open and close operations and $num_{TS} \times num_{var}$ read and write operations. We were able to significantly reduce the number of these expensive disk I/O operations through the creation the PyReshaper (Paul et al., 2015, 2018). We next describe the PyReshaper tool which was adopted into the CESM post-processing framework (Bertini and Mickelson, 2019)

The approach used by PyReshaper is illustrated in Figure 2. An MPI rank is assigned one or more fields to read from the time-slice file and write to the time-series file. Each MPI rank i operates independently and performs $num^{i}_{var} * num_{TS} + 1$ open and close operations and $num^{i}_{var} * num_{TS} + 1$ read and write operations where num^{i}_{var} is the number of fields assigned to MPI rank i. Given a sufficient amount of memory, it is possible to further reduce the number of write operations by writing multiple time slices to the filesystem in a single call. This task base parallelism supports execution on as many MPI ranks as there are fields in the input data set. Ideally if all the input fields were the same size and the cost to read the data from and write the data to the filesystem was negligible it would be possible to achieve a maximum speedup of num_{var}. Unfortunately the size of all input fields are not the same and the cost of read data from and write data to the filesystem is not negligible. We next describe the actual speedup the PyReshaper approach enables.

In the performance evaluation of PyReshaper, we evaluated the time it took to convert 10 years of monthly atmospheric data into the time-series format. This test configuration represents the conversion of approximately 180 GBytes of input data. The conversion took approximately 5 ½ hours using the existing serial method on NCAR's Cheyenne (Cheyenne, 2017) supercomputer. Figure 3 illustrates the performance improvements of the PyReshaper tool over the existing method. Note that using 144 MPI ranks we achieve the same conversion in approximately 4 ½ minutes.

The large improvement seen between 144 ranks and 72 ranks is an indication of a load-imbalance in the partitioning of fields to MPI ranks within the PyReshaper tool. Because the algorithm does not take into account any difference in processing cost between variables, some ranks can end up with more expensive three-dimensional variables to process while others may get only two-dimensional variables.

2.2 Diagnostics

One of the main ways the NCAR scientists evaluate the output of CESM is to run the component diagnostic packages. This task is represented by the "Diagnostics" task within Figure 1. They consist of four separate packages which are used to evaluate atmosphere, ocean, ice, and land model output. Each of these packages creates a set of average files, or climatology files, plots the data against observations or another model run, and then creates an HTML document that links all of the plot image files. To do this, each of these packages used a combination of shell scripts, NCO, and NCL (NCL, 2019) to analyze the data. Each

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

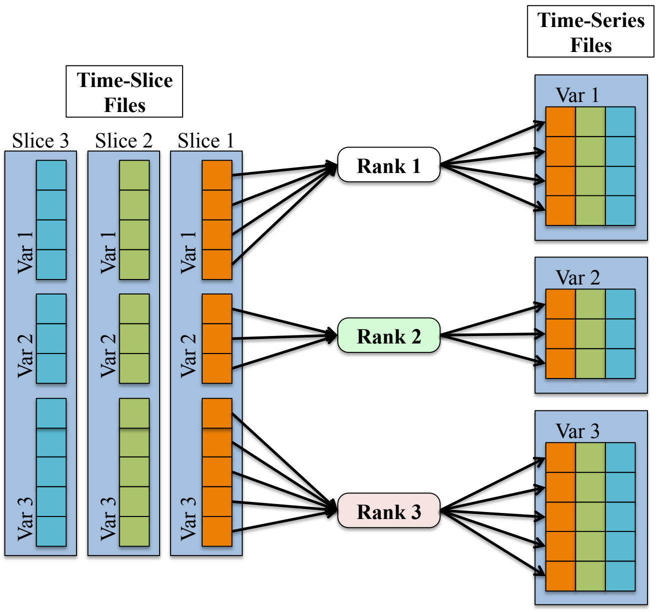

Figure 2. This figure shows the process of converting the data from a time-slice format to a time-series format in parallel within the PyReshaper. Each MPI rank is responsible for taking a particular variable from each time-slice file and writing it to the time-series file.

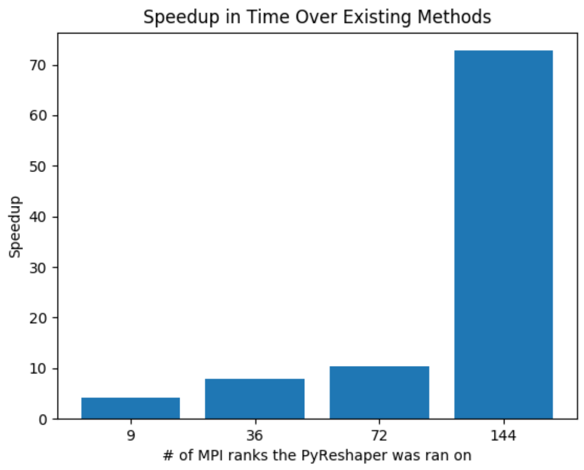

Figure 3. The comparative speedup in time of creating time-series files from ten years of monthly atmospheric data. In all cases, 493 time-series variable files were created. For comparison, the existing methods took approximately 5 ½ hours to complete. With 144 MPI ranks we were able to bring the time to do this same conversion down to approximately 4 ½ minutes.

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

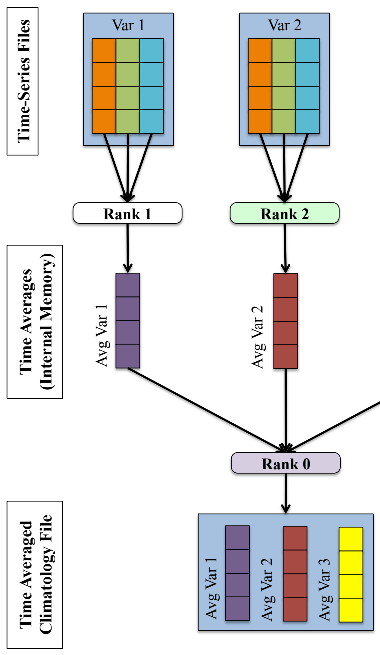

Figure 4. The parallelism strategy for the PyAverager for writing each climatology file. This figure describes the processes in one subcommunicator.

package requires different types of climatologies and plot types which creates unique performance characteristics for each of the packages. While previous efforts have enabled parallelism in the workflow (Woitaszek et al., November 2011; Jacob et al., 2012), this work presented it own set of issues. Specifically this approach resulting in poor performance for multiple file operations, and it had a steep learning curve for users. In order to create the climatology files in parallel and to reduce the expensive disk I/O operations, we developed the tool PyAverager (Paul et al., 2015; Mickelson et al., 2018). We also chose to call the NCL plotting scripts in parallel in order to improve performance further.

The parallelism strategy the PyAverager uses is illustrated in Figure 4. When the application begins, the pool of MPI ranks are partitioned into subcommunicators and the climatologies to be computed are partitioned across all subcommunicators. One MPI rank in each subcommunicator is assigned to be the writer of the given climatology file. Then, the field list is partitioned across the remainder of MPI ranks within the subcommunicator. Each of these ranks is responsible for retrieving its assigned field, computing the correct climatology, and then sending the result to the writer. After all fields have been written, the subcommunicator group begins computing the next climatology file it was assigned.

The second part of the diagnostics involves creating plots from the climatologies that were created. The plotting scripts individually can take a long time to run and run times vary among the plotting scripts. In order to improve the performance further, the CESM post-processing framework calls the individual NCL scripts in parallel. Because there are no data dependencies

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

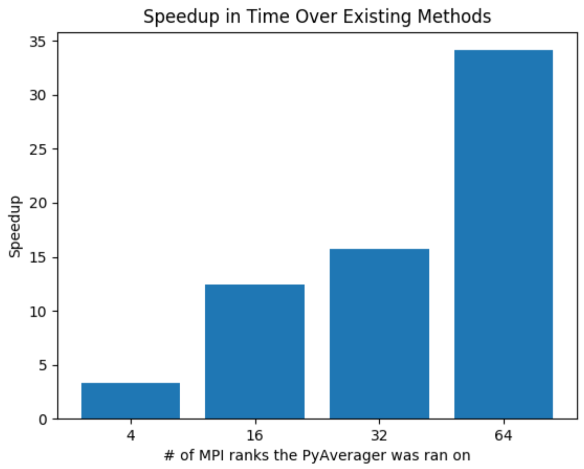

Figure 5. The comparative speedup of creating climatology files from ten years of monthly atmospheric data. Four seasonal and twelve monthly climatology files were created. For comparison, the original methods took approximately 26 ½ minutes to generate the climatologies. The PyAverager took approximately 46 seconds to create the same climatologies with 64 MPI ranks.

within the scripts, we are able to execute them in parallel. Therefore, if we have as many MPI ranks available as we do plotting scripts, the performance is limited to the longest running script.

In order to evaluate the performance improvements, we compared the time it took to create twelve monthly and four seasonal climatology files with the PyAverager against the NCO tools ran in serial. We chose to operate on the same data that was used to evaluate the performance of the PyReshaper in the previous section and all timings were performed on Cheyenne.

You can see from Figure 5 that the PyAverager is able to scale better than the PyReshaper. This is because the problem size is more load balanced. As you recall, the PyAverager distributes the number of averages to be done amongst the available subcommunicators and the number of variables are distributed amongst the ranks within the subcommunicator. For this particular problem, the work is more evenly distributed because the problem sizes were all similar and this lead to the better scaling.

The lack of improvement seen between ranks 16 and 32 is because the work wasn't evenly distributed and a subcommunicator ended up with slightly more work to do.

2.3 Conforming Data to Meet Specifications

The final step before publishing the data involves conforming the data to meet experiment specifications. This is represented as the "Conform Data to Experiment" task within Figure 1. This requirement is done in order to enable scientists to directly compare the data from different centers without any modifications. Some examples include renaming model variables, combining fields (e.g., adding or subtracting) to create one output field, converting units, verifying the data resides on the specified grid, and checking that the correct attributes are attached to the files. The method used for CMIP5 required users to write code to make required data transformations and to call the Climate Model Output Rewriter (CMOR) (Taylor et al., 2006) library to

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

check for compliance and to add file attributes. This was usually written as serial code and it took a long time to execute on a large data set. Also, generating the data was an error prone process that required expert knowledge to ensure data integrity. In order to meet the demands of CMIP6, we developed the tool PyConform (Paul et al., 2016, 2019) because we needed a tool with a flexible interface, that could create variable output in parallel, and still produced data that met specification requirements.

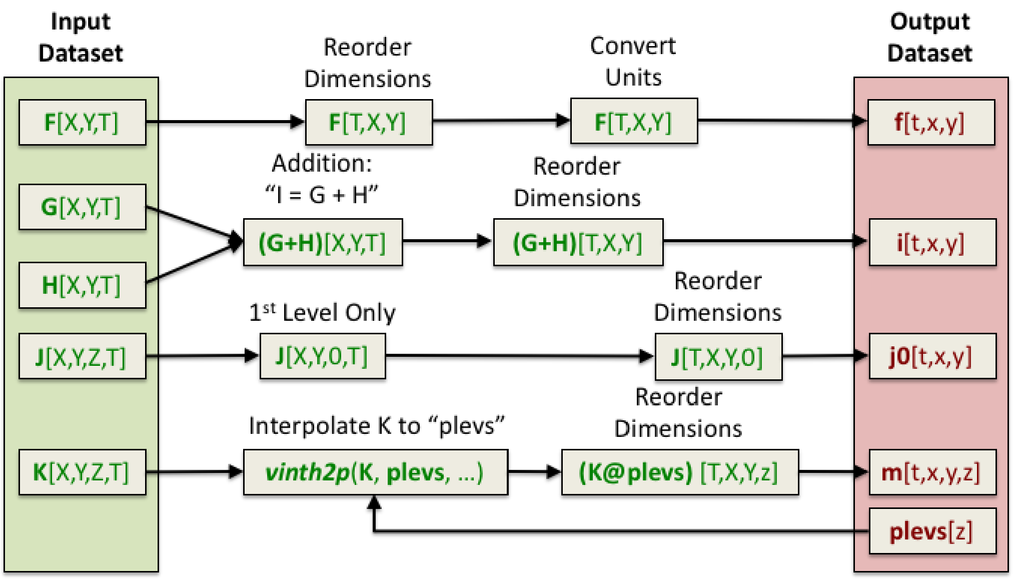

Figure 6. An example of a PyConform job. Each MPI rank is responsible for creating a particular output data set. Its job is to retrieve the variable data it needs, map operations, execute these operations, and then write the data.

An example of a PyConform job is shown in Figure 6. For this application, we again relied on a task-parallel approach, parallelizing across output files. The input fields are found on the left side of the figure. These fields are operated on as they are fed through the system in order to produce the output fields on the right. There are a variety of operations that can be performed on the data and this figure only shows a small subset. Several common functions and arithmetic operations are provided with the tool, but we could not account for all functions users may need. If more functionality is needed, we provide a framework in which users can create their own functions in Python and plug them into the framework.

PyConform depends on the CMIP6 data request Python API, `dreqPy` (Juckes et al., 2020). This package interfaces with the CMIP6 data request database which contains information regarding all of the fields within the request. This includes field names, descriptions, units, coordinates, and other specific information. Experiment information is also contained within the data request, specifying experiment descriptions and which fields are being requested for that experiment.

In order to evaluate the performance of the PyConform tool, we chose to compare it against the performance of the Fortran code that we used for CMIP5. In this example we were limited to generating only fifty variables because this was the union of variables that matched between CMIP5 and CMIP6 for the atmosphere model.

In our evaluation we found that the original method took approximately 9 1/2 minutes to generate the CMIP compliant output and it took PyConform about 1 1/2 minutes to generate the same output using 16 MPI ranks. This provided us with over a six times speedup over existing methods. Since this was a smaller problem, we chose to run the timing tests on a smaller number of ranks. When PyConform was executed in a production mode for CMIP6, it generated thousands of variable files and we are able to scale out to more MPI ranks efficiently.

2.4 Data Publication

The final step in the CMIP workflow within Figure 1 is publication of reviewed experiments to the ESGF, which is the data distribution and access platform designated for sharing CMIP and related simulation data. NCAR operates an ESGF Data Node, which is a software application stack that includes tools for checking conformance to the CMIP6 metadata standards, serving NetCDF data files using a Thredds Data Server and Globus Transfer, replication services, automated citation generation and experiment lifecycle support, including data retraction and re-publication.

A significant challenge with CMIP5 data publication was managing the velocity and complexity of data publication using ad hoc communications, such as email. Given the challenges of post-processing noted above, each experiment was published incrementally. This led to multiple versions of experiments and added unnecessary complexity to the publication process. A separate challenge was managing an under-development and changing ESGF software stack during the production CMIP5 data publication. The burden of updating the ESGF node frequently coupled with changing metadata requirements led to further slow-downs in the overall process.

For CMIP6 the ESGF software components were significantly improved, due to the increase in diversity, complexity and volumes being managed, as well as the experiences of data managers and node operators during CMIP5. In addition, a number of new components were developed for CMIP6, including the PrePARE data QC tools, a data replication tool, the Errata Service, and the Citation Service. These components were tested through a series of five "Data Challenges", which NCAR participated in as a member of the CMIP Data Node Operations Team (CDNOT) from January to June 2018. These data challenges were performed in advance of the model data availability and served to harden and improve the ESGF software stack with a series of integration and other system level tests. The significant improvements to the ESGF software stack and related tools vastly improved the rate of data publication for CMIP6. These performance improvements are shown within Figure 7. In the first two months of the CMIP6 publication process, NCAR was able to smoothly publish 50 TB more than it had published in the full 25 months it took to publish data towards the CMIP6 campaign because of the improvements that were made.

3 Process Workflow

During the completion of the CMIP5 simulations, each of the processes illustrated in Figure 1 were independent tasks, and they were not automatically run in succession. Another problem was that each of the tasks were run by different individuals causing workflows to stop while they waited for someone to start the next task. For a run to have continuous forward progress,

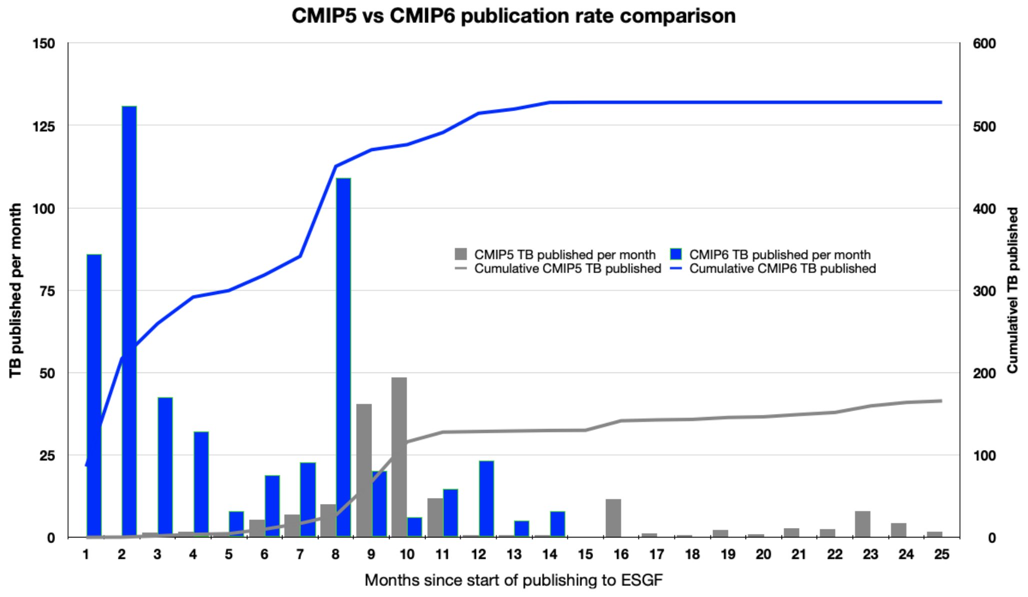

Figure 7. The cumulative and per month increases in the volume of data published to ESGF. During CMIP5, the ESGF software was stressed and problems arose. Despite those problems, NCAR was able to publish 165 TB of data. In preparation for CMIP6, the problems with the ESGF software stack were addressed and through these improvements, NCAR was able to publish 528 TB of data, a three times increase in volume. In the first two months of the CMIP6 publication process, NCAR smoothly published over 216 TB of data, over 50 TB more than it contributed towards CMIP5.

it needed to be monitored repeatedly at all hours and people needed to be on call continuously to post-process the data and this was not practical. There was also no fault-tolerance built into this workflow. If part of the simulation failed because of machine error, the simulation stopped and it wouldn't restart until someone did a manual check.

We adopted the use of Cylc (Oliver et al., 2018) for our CMIP6 experiments in order to coordinate the execution of all of the tasks within the end-to-end workflow of an experiment. Cylc is a workflow management tool developed at the National Institute of Water and Atmospheric Research (NIWA) and supported through NIWA and the UK Met Office. A Cylc workflow can be invoked through command line tools or through a graphical user interface (GUI) and runs daemons in the background in order keep track of the status of all of the running tasks. In order to track the status of all of the tasks within a workflow, Cylc updates its internal database that contains information about each of the tasks. This allowed the workflow to recover to a previous state if a problem was encountered on the machine.

The Cylc workflows were able to incorporate all of the tasks that a user wanted to execute. This included the model iterations, the moving of data, and the running of all of the Python tools discussed in this paper. This made the end-to-end workflow seamless and users did not have to worry about submitting any of the tasks by hand. This also eliminated the needed expertise to run the post-processing tools. Instead they were setup correctly and automatically ran as part of the workflow.

Cylc also provided fault-tolerance within the workflows. Cylc allows you to specify if you would like for it to try rerunning a particular executable if it fails. This was especially helpful when the compute system was unstable. If one of the model runs failed because of machine error, it was resubmitted to the queue and rerun without user intervention.

The process of setting up a CMIP6 workflow is complex because of the different tools that need to be setup for a particular experiment. This includes the Python tools discussed in this paper and the Cylc workflow file. In order to reduce the burden on the users, a Python setup script (Mickelson, 2020) performed many of the setup steps so users did not need any CMIP6 expertise. Once the users set the default values, the script created the CESM experiment, created a post-processing directory, set up the post-processing tools for the specified CMIP6 experiment, and created a Cylc workflow definition file based on known task dependencies between the different tasks that were to run. After the script completed, users only needed to set experiment specific information into the CESM model and to build the model. Then the users started the experiment through Cylc.

These modifications had the largest positive impact on our ability to complete our contributions towards CMIP6. Through this process, the users did not have to have expert knowledge on how to run any of the post-processing tools nor did they need to know how to format the published data. This eliminated the need to have an extra person run the post-processing tasks by hand. Also, having the workflow submit all tasks and resubmit failed tasks allowed us to complete experiments and publish data sooner because everything was continuously running. Finally, it made the process of completing complex experiments easier. As an example, for a particular MIP exercise, we were required to run eight different experiments, each containing one-hundred ensemble members (Deser and Sun, 2019; Smith et al., 2019). This would have required the user to build all eight-hundred experiments, run each experiment, create time-series files for each, and then create the standardized files for each experiment and all these steps would have been done by hand. This becomes a labor intensive process that requires extensive bookkeeping. With the workflow automation provided by Cylc, we were able to complete the eight experiments with each taking only a couple of days to complete and the user was only required to run a script that set up each case and to click on a start button.

4 Experiment Documentation

The experiments that were done for CMIP5 contained little documentation and no provenance was obtained. This made the simulations difficult to reproduce without having to contact the person who ran the original simulation. Another problem that was encountered was that it was difficult to track the progression of the simulations for CMIP5. During the process, only one individual knew which runs were in progress, the status of each of the simulations, and what was complete. To address these problems, the CESM experiment database was extended to provide the extra features that were needed.

The first task was to make it easier for the scientists to enter new experiments into the database. The previous version of the database required users to enter several pieces of information and this made it a cumbersome process. To improve this, known information was automatically harvested from the CESM experiment and was filled in from the experiment's case information and from the CMIP6 data request. This reduced the number of fields users had to fill in by hand and made the process more streamlined.

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

The next task was to allow for the experiments to upload their configuration and timing files to a subversion repository so that experiment provenance could be captured. This was done by adding a subversion commit call right after a run iteration completed. When the data archiving step was ran, the code gathered up all relevant files, created a new subversion directory with the current date stamp, and then uploaded the files. The database then gathered the differences and displayed them under the experiment's entry within the database. This allowed users to quickly identify changes that were made mid-run.

Another feature request was to link the diagnostic package results to the database. As discussed in Section 2.2, the diagnostic packages produce several plots that are linked within an HTML document. The workflow uploaded these HTML documents automatically to a web server so people could view the results as soon as they were produced. The links to these web pages are found within each experiment's entry in the database so others could easily locate all of the results at one location.

The final feature request was to provide run-time status. As stated previously, it was difficult to know the status of any given experiment. In order to automate the run-time status, we had each experiment's Cylc workflow email the database its status. The database then parsed the emails and updated the progress information. This new interface allowed management and scientists to monitor the status of all CMIP6 experiments and to identify simulations that ran into problems.

Collectively these enhancements allowed us to track the progress of the experiments and document model configurations, output, and diagnostics all within one utility. This work also lead itself to other research projects that allowed for analyzing the timing information that was collected from each model run in order to study the model performance over time. This allowed for the identification of a degradation of performance after a machine upgrade, users selecting imbalanced processor layouts for their model runs, and model performance degradations (Mannik, 2019). This information can be then used to improve model performance and allow for more efficient use of computational resources.

5 Conclusions

Every generation of MIP exercises introduces new layers of complexity. We learned in CMIP5 that we could no longer use traditional tools to post-process the required amount of data and still meet our deadlines. CMIP6 required us to develop a new tool chain and forced us to change our methodologies. These new methods, described in this paper, provided us with an 18-times speedup. This allowed us to meet our deadlines and we were able to publish more than half a million data sets on the ESGF for the CMIP6 project.

While Cylc has a learning curve, it was shown through this work to be extremely useful in coordinating all of the individual tasks of running a simulation, running diagnostics, and post-processing the data. It was shown to save both human time and time to simulation completion. Because of this success, Cylc is being more tightly integrated within CESM. This tighter integration now resides within the CESM infrastructure code and a Cylc workflow can now be generated with an option set within the CESM environment instead of it being a standalone Python script.

While we have shown that our new Python tools were successful, we believe these fundamental tasks should also be integrated more tightly within the CESM. This includes the time series and data conforming tasks. The current practices force multiple versions of the data to be on disk at a given time. As future MIP's grow more complex, their requested data volumes grow larger. This growth in data being requested makes it more difficult to carry multiple versions of the data around and the tighter integration of having the formatted data generated directly from the model simulation will allow us to save disk space.

CMIP exercises are resource-expensive and time-consuming to run. The complexity continues to grow with every generation, and focused efforts are needed to coordinate the improvements to the infrastructure code around these attempts. We believe we've made the correct steps in improving our process and we will continue to work towards more integrated development for future MIP exercises.

Code availability. The versions of the code that were used within our end to end workflow process for CMIP6 can be found at the following locations:

The version of the PyReshaper (version 1.0.6) that was used in this work can be downloaded from https://doi.org/10.5281/zenodo.3894842 (Paul et al., 2018). Further information can be found at https://github.com/NCAR/PyReshaper.

The version of the PyAverager (version 0.9.16) that was used in this work can be downloaded from https://doi.org/10.5281/zenodo.3894862 (Mickelson et al., 2018). Further information can be found at https://github.com/NCAR/pyAverager.

The version of the PyConform (version 0.2.8) that was used in this work can be downloaded from https://doi.org/10.5281/zenodo.3895009 (Paul et al., 2019). Further information can be found at https://github.com/NCAR/PyConform.

The version of the CESM post-processing framework (version 2.2.1) that was used in this work can be downloaded from https://doi.org/10.5281/zenodo.3895033 (Bertini and Mickelson, 2019). Further information can be found at https://github.com/NCAR/CESM_postprocessing.

The version of the CESM workflow generation tool set (version 1.0) that was used in this work can be downloaded from https://doi.org/10.5281/zenodo.3895058 (Mickelson, 2020). Further information can be found at https://github.com/NCAR/CESM-WF.

The CESM model (version 2) can be found at https://doi.org/10.5065/D67H1H0V (Danabasoglu et al., 2020). This work used the CESM versions

2.1.0 (https://doi.org/10.5281/zenodo.3895306),
2.1.1 (https://doi.org/10.5281/zenodo.3895315),
2.1.2 (https://doi.org/10.5281/zenodo.3895328).

For this work we used Cylc version 7.8.3. The source code for this version can be retrieved at https://doi.org/10.5281/zenodo.3243691 and it is referenced within Oliver et al. (2018).

Author contributions. SM led the development of the end-to-end CESM workflow for CMIP6. AB contributed to the development of the CESM post-processing framework and developed the CESM experiment database. GS managed the data for the project and was an advisor. KP contributed to the development of the Python post-processing tools. EN published NCAR's CMIP6 data onto ESGF. JD and MV were advisors for the project.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. The CESM project is supported primarily by the National Science Foundation (NSF). This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the NSF under Cooperative Agreement No. 1852977. Computing and data storage resources, including the Cheyenne supercomputer (doi:10.5065/D6RX99HX), were provided by the Computational and Information Systems Laboratory (CISL) at NCAR.

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

References

Abdulla, G.: Annual Earth System Grid Federation 2019 Progress Report, https://esgf.llnl.gov/esgf-media/pdf/2019-ESGF-Progress-Report.pdf, 2019.

Bertini, A. and Mickelson, S.: CESM Postprocessing (verison 2.2.1), https://doi.org/10.5065/4XV0-FG55, 2019.

Cheyenne: Computational and Information Systems Laboratory. Cheyenne: HPE/SGI ICE XA System (Climate Simulation Laboratory). Boulder, CO: National Center for Atmospheric Research, https://doi.org/doi:10.5065/D6RX99HX, 2017.

Cinquini, L., Crichton, D., Mattmann, C., Harney, J., Shipman, G., Wang, F., Ananthakrishnan, R., Miller, N., Denvil, S., Morgan, M., Pobre, Z., Bell, G. M., Doutriaux, C., Drach, R., Williams, D., Kershaw, P., Pascoe, S., Gonzalez, E., Fiore, S., and Schweitzer, R.: The Earth System Grid Federation: An open infrastructure for access to distributed geospatial data, Future Generation Computer Systems, 36, 400–417, https://doi.org/10.1016/j.future.2013.07.002, 2014.

Daily, J.: pagoda, https://github.com/jeffdaily/pagoda, 2013.

Dalcin, L.: MPI for Python, https://mpi4py.readthedocs.io/en/stable/, 2019.

Danabasoglu, G., Lamarque, J. F., Bachmeister, J., Bailey, D. A., DuVivier, A. K., Edwards, J., Emmons, L. K., Fasullo, J., Garcia, R., Gettelman, A., Hannay, C., Holland, M. M., Large, W. G., Lawrence, D. M., Lenaerts, J. T. M., Lindsay, K., Lipscomb, W. H., Mills, M. J., Neale, R., Oleson, K. W., Otto-Bliesner, B., Phillips, A. S., Sacks, W., Tilmes, S., van Kampenhout, L., Vertenstein, M., Bertini, A., Dennis, J., Deser, C., Fischer, C., Fox-Kember, B., Kay, J. E., Kinnison, D., Kushner, P. J., Long, M. C., Mickelson, S., Moore, J. K., Nienhouse, E., Polvani, L., Rasch, P. J., and Strand, W. G.: The Community Earth System Model version 2 (CESM2), Journal of Advances in Modeling Earth Systems, 12, https://doi.org/10.1029/2019MS001916, 2020.

Deser, C. and Sun, L.: Atmospheric circulation response to Arctic sea ice loss: sensitivity to background SSTs, in: AGU Fall Meeting Abstracts, vol. 2019, pp. A51A–03, https://ui.adsabs.harvard.edu/abs/2019AGUFM.A51A..03D, 2019.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geoscientific Model Development (Online), 9, https://doi.org/10.5194/gmd-9-1937-2016, 2016.

Gropp, W., Lusk, E., and Skjellum, A.: Using MPI: portable parallel programming with the message-passing interface, vol. 1, MIT press, 1999.

Hurrell, J. W., Holland, M., Gent, P., Ghan, S., Kay, K., Kushner, P., Lamrque, J.-F., Large, W., Lawrence, D., Lindsay, K., Lipscomb, W., Long, M., Mahowald, N., Marsh, D., Neale, R., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., Collins, W., Hack, J., Kiehl, J., and Marshall, S.: The Community Earth System Model: A Framework for Collaborative Research, Bulletin of the American Meteorological Society, pp. 1339–1360, https://doi.org/10.1175/BAMS-D-12-00121.1, 2013.

Jacob, R., Krishna, J., Xu, X., Mickelson, S., Tautges, T., Wilde, M., Latham, R., Foster, I., Ross, R., Hereld, M., Larson, J., Bochev, P., Peterson, K., Taylor, M., Schuchardt, K., Yin, J., Middleton, D., Haley, M., Brown, D., Huang, W., Shea, D., Brownrigg, R., Vertenstein, M., Ma, K., and Xie, J.: Abstract: Bringing Task and Data Parallelism to Analysis of Climate Model Output, in: 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, pp. 1493–1494, https://doi.org/10.1109/SC.Companion.2012.282, 2012.

Juckes, M., Taylor, K. E., Durack, P., Lawrence, B., Mizielinski, M., Pamment, A., Peterschmitt, J.-Y., Rixen, M., and Sénésis, S.: The CMIP6 Data Request (version 01.00.31), Geoscientific Model Development, 13, 201–224, https://doi.org/10.5194/gmd-13-201-2020, 2020.

Kornblueh, L., Mueller, R., and Schulzweida, U.: Climate Data Operators, https://code.mpimet.mpg.de/projects/cdo/, 2019.

https://doi.org/10.5194/gmd-2020-133
Preprint. Discussion started: 24 June 2020

Mannik, L.: Novel Database and Usage Analytics for the CESM2 Climate Model: First Steps to Tracking Configuration and Performance, https://www2.cisl.ucar.edu/siparcs-2019-mannik, 2019.

Meurdesoif, Y.: XIOS, https://forge.ipsl.jussieu.fr/ioserver, 2020.

Mickelson, S.: CESM Workflow (version 1.0), https://doi.org/10.5065/7we1-8k84, 2020.

Mickelson, S., Paul, K., and Dennis, J.: PyAverager (version 0.9.16), https://doi.org/10.5065/9zx1-jq74, 2018.

NCL: NCL, https://doi.org/10.5065/D6WD3XH5, https://www.ncl.ucar.edu/, 2019.

Oliphant, T.: Python for Scientific Computing, Computing in Science and Engineering, 9, 10–20, https://doi.org/10.1109/MCSE.2007.58, 2007.

Oliver, H., Shin, M., and Sanders, O.: Cylc: A Workflow Engine for Cycling Systems, Journal of Open Source Software, 3, 737, https://doi.org/10.21105/joss.00737, 2018.

Paul, K., Mickelson, S., Dennis, J. M., Xu, H., and Brown, D.: Light-weight parallel Python tools for earth system modeling workflows, 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, pp. 1985–1994, https://doi.org/10.1109/BigData.2015.7363979, 2015.

Paul, K., Mickelson, S., and Dennis, J. M.: A new parallel python tool for the standardization of earth system model data, 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, pp. 2953–2959, https://doi.org/10.1109/BigData.2016.7840946, 2016.

Paul, K., Mickelson, S., Dennis, J., and Hu, H.: PyReshaper (version 1.0.6), https://doi.org/10.5065/b92r-gt40, 2018.

Paul, K., Mickelson, S., and Dennis, J.: PyConform (version 0.2.8), https://doi.org/10.5065/9n3z-7x72, 2019.

Perez, F., Granger, B., and Hunter, J. D.: Python: An Ecosystem for Scientific Computing, Computing in Science and Engineering, 13, 13–21, https://doi.org/10.1109/MCSE.2010.119, 2011.

Smith, D. M., Screen, J. A., Deser, C., Cohen, J., Fyfe, J. C., García-Serrano, J., Jung, T., Kattsov, V., Matei, D., Msadek, R., Peings, Y., Sigmond, M., Ukita, J., and Yoon, J.-H.and Zhang, X.: The Polar Amplification Model Intercomparison Project (PAMIP) contribution to CMIP6: investigating the causes and consequences of polar amplification, Geoscientific Model Development, 12, 1139–1164, https://doi.org/10.5194/gmd-12-1139-2019, 2019.

Taylor, K., Doutriaux, C., and Peterschmitt, J.-Y.: Climate Model Output Rewriter (CMOR), https://pcmdi.github.io/cmor-site/media/pdf/cmor_users_guide.pdf, 2006.

Taylor, K. E., Stouffer, R. J., and Meehl, G. A.: An Overview of CMIP5 and the Experiment Design, Bulletin of the American Meteorological Society, 93, 485–498, https://doi.org/10.1175/BAMS-D-11-00094.1, 2012.

Williams, D. N.: Visualization and Analysis Tools for Ultrascale Climate Data, Eos: Earth and Space Science News, 95, 377–378, https://doi.org/10.1002/2014EO42000, 2014.

Woitaszek, M., Dennis, J. M., and Sines, T. R.: Parallel high-resolution climate data anslysis using swift, in: MTAGS'11: Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers, pp. 5–14, https://doi.org/10.1145/2132876.2132882, November 2011.

Zender, C.: Analysis of self-describing gridded geoscience data with netCDF Operators (NCO), Environmental Modelling and Software, 23, 1338–1342, https://doi.org/10.1016/j.envsoft.2008.03.004, 2008.