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)

We would like to thank our two reviewers for their thorough review and thoughtful suggestions to make a stronger paper. We have included all of their comments below in blue and in italics We have included our responses after each comment. The line numbers we reference refer to the line numbers found within this response.

Reviewer #1:

This manuscript uses CMIP5 and 6 as a case study to show the improvements made to the workflow in order to support CMIP6 at NCAR/CESM. While this work is appreciated and does do a decent job of comparing the workflow from the CMIP5 era to the present, the information presented in the manuscript needs to be more efficiently conveyed so it's useful to the community as a whole. Information on CMIP5 workflow is not very elaborate to get an in-depth understanding and appreciation for the CMIP6 workflow efforts. Some parts of the manuscript can be much more than just an "internal documentation". Schematics like Figure 1 can be vastly improved. Information in the manuscript should serve as a motivation point for other labs to consider new workflow models. Several points are highlighted below. Most importantly, human time consumption could also be provided in this manuscript. Secondly, as we move towards a new computing era, the manuscript should also let the readers know what else is out there in order to develop a new workflow, inspired by this manuscript. A data agnostic model, processing workflow, cloud-optimized workflows, etc should be touched upon at least in the conclusions section. Overall, thanks for the manuscript and congratulations on publishing CMIP6 data on the ESGF.

Figure 1 was improved by changing the orientation to project a more natural flow of the data. The shapes were modified to reflect recommendations for flowcharts and colors were muted. It was also added that this is a simplistic representation of our workflows and we point to a Cylc workflow graph within the Appendix to show an example of a flowgraph we use in practice.

We tried to address the amount of human time where appropriate. We've added information about human time at lines 33, 56, 69, 72-74, and 326-334.

We addressed future analytic platforms within the conclusion and argued for the need of standardization of data to help make analysis easier on these types of platforms.

1. *Page 1 Ln 3: Statements such as "Many centers were not prepared.." needs to be rephrased to indicate the unexpected increase in complexity of CMIP6. Unless*

the relevant facts are cited and the data is provided, comments about other centers does not seem appropriate here.

This was a statement that was discussed informally amongst different centers that participated in CMIP5. This statement was removed (line 3) from the document as an appropriate citation does not exist.

2. *Page 1 Ln 7: It is nice that there is six times improvement. Please verify if the actual data volume is specified in the manuscript for CMIP5 and CMIP6.*

This was added starting at line 7 through line 10 within the Abstract.

3. *Page 1:Ln 20: For a diverse audience to follow, please indicate what is meant by postprocessed data when it's mentioned for the first time.*

This was addressed within the additions at line 23 and at line 48 within section 2.

4. *Page 1. Ln 24. What is the factual evidence to show CESM ran relatively quickly compared to the other climate models? Are you referring to models from NCAR or other modeling centers? Is this number from a CPMIP computational metrics calculation? If there is no factual evidence or appropriate citation, this statement does not seem to be appropriate.*

Again, this was a statement that was discussed informally amongst different centers that participated in CMIP5. This statement was removed from the document as an appropriate citation does not exist (lines 27-28).

5. *Page 2: Ln 1: What are the software inefficiencies? Workflow development tools? Please expand on this. Software seems very generic.*

This was modified at lines 28-29 to be more specific.

6. *Page 2: Ln 44, The line that begins with "For CMIP5..", please break this sentence into two or three and avoid using "it.." several times in the same sentence, for clarity.*

This sentence was modified starting at line 49.

7. *Page 2: Ln 45: What is different in CMIP6 versus CMIP5 in terms if requiring expert knowledge to ensure data met the correct standards? It does require expert knowledge in order to verify the correctness of scientific model output regardless for the phase of the CMIPs. If this is incorrect, please clarify and rephrase the sentence to avoid confusion.*

This statement is clarified within lines 50-52.

8. *Page 2 Ln 46: How is the standardized data verified? What is the Quality Assurance process in the workflow? What is meant by "standardized data"? Please spell out the conventions to be adhered to.*

The verification and quality assurance portion of this comment is addressed within the additions in line 52 and lines 238-239. The standardized data clarification is addressed within the additions in line 48.

9. *Page 2, Ln 51: Where are the simulations run? Information regarding the computational environment is completely missing. What is the name of the compute system? Is/Was there a batch workflow, job scheduler etc., etc.*

Information about our compute platforms was added at line 80 through line 86. Information about the scheduler was added within Process Workflow section (section 3) at line 335. These changes caused for the removal of the citation at line 116 and the additional clarification needed at line 231.

10. *Page 3. Ln 67: "The publication of CMIP5 data contributions to the ESGF was also a bottle neck within the data workflow" – This line needs more clarity to indicate what exactly is referred to as the bottle neck. Is ESGF the bottleneck? The tone of the sentence could be more constructive if a community developed federated framework is being criticized. Just as an example, thought not complete. Though there were performance issues with respect to the data publication onto the Earth System Grid Federation for a few reasons, the performance increased phenomenally due to …..*

This paragraph was modified as suggested. It can be found starting at line 75.

11. *Page 4, Ln 91 TYPO: task based parallelism*

This was changed at line 109. Thanks for catching that.

12. *Page 4, Ln 102: Try and avoid starting the sentence with "Because" .*

This was specifically addressed at line numbers 121 and 163.

13. *Page 4, Ln 110: How dependent is the diagnostics framework on the supported languages you've described? What is the potential to expand the supported languages to say Python, Ruby, etc. ?*

This line was modified in order to address this question and the response can be found at line numbers 136 through 140.

14. *Page 6, Ln 113: There is a mention of "this work". Please clarify what work this entails as part of this sentence for clarification. There is also a TYPO in "Specifically ..". Consider changing resulting to "results in" as you see fit.*

The clarification was added to lines 144-145 and, as a result, the typo was removed.

15. *Page 6, Ln 125: Was this a total re-write of the post-processing framework since CMIP5?*

A clarification was added to lines 162-163.

16. *Page 7 : Figure 5- Y axis units missing. Please check all figures as well. Figure 5 caption indicates "46 seconds". Is this in sync with what is shown in the actual figure? Please verify.*

Figure 5 is now Figure 6 and in it the Y axis was modified to be clearer that it's a speedup and the units are in "x". Since it's in speedup vs. actual time, the 46 seconds was added as a reference point to give readers an idea on how much time a task takes to complete. This time was verified and the plot correctly represents the data presented.

17. *Page 7 Ln 122. Figure 5 simply has info on pyAverager. If there is a single image that cross-compares the speedup time for the two tools, that'd be effective and in sync with the text in Ln 122.*

Figure 5 was added to address this comment. It shows a comparison that includes the time to create both the climatologies and all of the plots for three of the diagnostic packages. The paragraph starting at line 166 was written to describe these results. The paragraph starting at line 171 was modified slightly to help with the flow within the descriptions of Figures 5 and 6.

18. *Page 7, Ln 140. Paraphrase this sentence, especially "without any modifications". E.g. ..to allow seamless multi-model comparison based on uniform data standards to avoid less rewriting and error-prone transformations while doing so.. etc.*

This is clarified within lines 187-188.

19. *Page 7, Ln 142. "required" is a strong word here and not quite accurate. Please paraphrase this. Not all modeling centers used CMOR even for CMIP5.*

This was modified in line 190.

20. *Page 8, Ln 154. How are changes in dreqPy incorporated in the workflow? Was there a fixed version? How were corrections in the requirements considered and incorporated?*

This information was added into the new paragraph starting at line number 212.

21. *Page 9 Ln 158. CMOR may also have a Python interface, please double check and then change this sentence as needed.*

We modified lines 192-193 to reflect that we are referring to our code. CMOR does have a Python interface, but the problem resided within our software and not within CMOR.

22. *Page 10, Section 2.4. There is no mention of Data Quality Assurance which is extremely important though parts of the workflow may be automated. Please indicate the steps taken to quality control datasets. PrePARE could be accounted for metadata QA, but not quite for data and I am curious how that was incorporated. PrePARE also comes at a later stage when the heavy lifting of data processing and prep is almost complete. So, a bug revealed at a later stage may*

have its own cons. There was also a similar CMOR checker in some form available for CMIP5, though for CMIP6 it was more robust.

We have added a paragraph within the Data Publication section (2.4) starting at line 260 that talks about how we used PrePARE to verify the correctness of our data. We also added clarification to lines 238-239 indicating that the data was also verified by scientists visually inspecting the data before it was hand triggered to be published.

23. *Page 10, Ln 175: Not sure if it's a typo – under-development versus under-developed. This sentence needs to revised either way. A constructive tone would be great.*

This was changed in line 246.

24. *Page 10, Ln 181: Please cite CDNOT paper that was recently submitted if you haven't already. Ruth et al. 2020*

Thank you for bringing this to our attention. It is now cited at line 253.

25. *Page 10, Ln 183: Consider changing "harden" to strengthen or similar.*

This was modified at line 254.

26. *Page 10, Section 2.4. Versioning and ESDOC are two important components in data publication and the ESGF. These are not touched upon and would add immense value to the manuscript to include the process for these.*

Both of these concepts have been added within the two new paragraphs that are found within lines 267-274.

27. *Page 10. A schematic for Cylc, e.g. Cylc dependency graph/dashboard would add value to the manuscript.*

A Cyc workflow graph has been added as Appendix A1 and a citation has been included for where you can find the Cylc workflow code that was used to create that graph can be found. This is referenced from lines 297-299. The dashboard information can be found within the workflow documentation that is referenced within Mickelson, 2019b.

28. *Page 10. Ln 201, What is the internal DB implementation? How easy or difficult was it to get started with Cylc and is Cylc also used in other domains? Does the user have the ability to monitor the processes via Cylc and resubmit a job if needed?*

This information was added within the paragraph that starts at line 320. Information is also provided at lines 288-290. We also added training information that starts at line 326 that gives incite on how we made Cylc easier for our users.

29. *Page 11 Ln 207: Sample configuration files from Cylc would be helpful. A section to explain how reproducibility is achieved in the workflow with a schematic and a case study– would be helpful.*

A sample configuration file is cited at line 299. We have also published all of our workflow configurations in Mickelson, 2020 (see lines 318-319). Reproducibility is

further clarified with an addition within the Experiment Documentation section (section 4) at lines 372-374.

30. *Page 11, Ln 212: What are the setup steps? Is there an example of a definition file in the GitHub repository references?*

This is clarified with the additions to lines 311-312 and lines 314-315. As stated above, information on where to find example definition files can be found within Mickelson, 2020 (see lines 318-319).

31. *Page 11, Ln 218. How is troubleshooting and monitoring happening with Cylc and your workflow? Who manages that?*

Information in regards to Cylc automatically monitoring the status and resubmitting can be found at lines 303-307. Wording was modified in lines 300-302 to make that addition flow better. Monitoring by users is addressed though lines addition 320-321. Monitoring is also touched upon in the paragraph about training in lines 332-333.

32. *Page 11. Ln 225-227. This is nice. Was data publishing part of the automated pipeline? Please explain. In a fully automated workflow, what were the testing strategies, version control mechanisms, provenance capture mechanisms, etc.? There is little mention about a couple of things, but more the better to make the manuscript stronger and reachable.*

Publication was not part of the automated workflow and this is now specifically stated at line 284 and at lines 238-239. Publication was triggered through a manual step within our database because we wanted to ensure that at least one person reviewed each simulation. This step was an authorization step verifying the correctness of the data.

33. *Page 11, Section 4, Ln 229: Again, please be constructive. Be specific as to what experiments you're referring to, what model, what modeling Centre.*

This was clarified at line 356.

34. *Page 11, Ln 2356:How was information automatically harvested from the CESM experiment?*

This information was added at lines 363-364.

35. *Page 12, Ln 239: Please expand on what configurations and timing files mean here.*

This line was changed to add more clarity. The changes can be found at lines 366-367.

36. *Page 12: Ln 241: What is the "code" that's referred to here? Please elaborate*

This is clarified at lines 369-370.

37. *Page 12: Ln 244: Are there sample analysis figures that could be provided? Are there any collaborative work on the diagnostic package that needs to be acknowledged? Is there scope for collaborative efforts since some of the diagnostic packages can be helpful to the community as a whole.*

A citation was added to give an example of diagnostics we created for some of our PMIP4 experiments. This can be found at line 457 within the references and it is cited at lines 141 and 375. These packages are currently being deprecated and new versions are being created as suggested at lines 407-410 within the conclusion. These changes fall within our work towards Pangeo (https://pangeo.io/) and collaboration details can be found through the link to the Pangeo project.

38. *Page 12: Nice- HTML docs for viewing results.*

Thank you

39. *Page 12, Ln 250 This seems to speak about monitoring capabilities, although the information provided here is not very useful for readers to learn from this work.*

This line was removed as it was too detailed and did not add to the text (line 382-383). Lines 380 and 381 were modified as well.

40. *Page 12, Section 5. Line 260. By traditional tools, are serial tools referred to ? Since this seems to be the major difference in your workflow paradigm since CMIP5?*

Yes, this was clarified in line 393.

41. *Line 263: Are there citations to the datasets referred to here? Any information on how CMIP6 citations were processed for CESM.*

We have included two references at lines 259 and 396. The first is the location to download the data. The second contains the data citations that are automatically generated from the publication process.

Conclusion should have future work as well, because what is considered traditional today will not hold good for the years to come. Lessons learned from CMIP6 exercise needs to be magnified in order to move towards cloud-optimized workflows and flexible APIs. The manuscript should give some food for thought to the readers. Examples to show if (if not) CMIP may be the only style of experiments that the workflow processes should be clarified.

The conclusions section now contains some insights into cloud data analytic workflows. This is included within lines 408-409 and 411-414. The last paragraph of the conclusion was also modified to give readers a more precise take away message (lines 415-422). This includes the underlying themes that motivated us to design our tools the way we did and why evaluations and redesigns are necessary.

Reviewer #2:

This paper presents the work carried out to completely modify the CESM's postprocessing workflow. It's interesting and useful to get an overview of such a process, but I think some information are missing for the paper to serve as an example for other communities. During my reading I would have liked to know more information on the Cheyenne supercomputer. For example, do you have some restrictions on the storage (volume quota, inodes quota), is this supercomputer dedicated only for CMIP6

experiments ? Did you have some restrictions on you CPU allocation for post-treatment ? For each part, I think it can be useful to have an information on the human time and FTE necessaries to realize the tool from scratch to the production. It's really a great job to have created this workflow that can be used by a "normal" user, and that avoids the problem of knowing CMIP data that only relies on a few people.

We have added lines 10-13 and 415-422 in hopes to clarify the motivation for this paper and to help frame our intentions for the community as they read this paper.

We have added lines 80-86 to describe our compute platforms and lines 335-342 to talk about our queueing system.

We have not included details on our volume quotas, but CISL was generous and gave us enough disk space that our experiments were able to run on our system without running out of space. We developed a specific data plan that influenced where the workflow would create files. For example, the raw files created directly from the model where put within our 1 PB scratch allocation and it was allowed to be purged off. Our post-process files that we kept were put into a more permanent space. This space was purchased specifically for CMIP6 and we determined the space we needed based on the CMIP6 data request for the experiments we were committed to.

We have also included information on human cost where appropriate and noted in your comments and from the comments of the reviewer #1.

We've added information about human time at lines 33, 56, 69, 72-74, and 326-334.

1. *Introduction – lines 24 & 25 : Can you add a graph in order to visualize calcul and post-treatment performances for NCAR and other climate models Data Workflow.*

   This was a statement that was discussed informally amongst different centers that participated in CMIP5. This statement was removed (lines 27-28) from the document as an appropriate citation does not exist.

2. *Line 41 : "it was time consuming" : can you precise if you are talking about "human time" (find the script, launch it, check it etc.) or CPU time ? -*

   This is clarified within lines 44-45.

3. *Line 63 : can you explicite "FTE" before to use it for the first time ? How did you make the FTE estimation for the implementation of XIOS and for the development of your own new tools ?*

   This is clarified within line 69.

4. *line 96 to 104 : Can you precise in the text how many Time-series (493) are created by your evaluation. Why did you stop the test to 144 MPI ranks and don't test with more MPI ranks ? Did you try with 493 MPI ranks ? Can you explain how finally you make your choice for the MPI ranks repartition you will use, I imagine there is a reflexion between the human time (5 1 2 hours with you previous workflow and now 4 1 2 minutes), the total CPU time (4 1 2 minutes * 144 = 10,8 hours), and your CPU allocation on Cheyenne. (this specific comment is done also for the other parts of your workflow)*

   This is now explained starting at line 124 through 127.

5. *Line 102 : did you try to improve the way you done the variables distribution on MPI ranks ?*

We did not, but an explanation is now provided at lines 128-132.

6. *Figure 3 : can you add the "ideal speedup" line on it ?*

This plot was modified to contain an ideal speedup line and it was changed to a line plot instead of a bar plot to show this information better.

7. *line 117 to 122 : can you add information on how the choice of subcommunicators's number was done, and of the MPI rank distribution on each subcommunicator. -*

This information was added starting at line 155 though line 159.

8. *Line 128 to 130 : can you explain on which criterion was done the climatologies distribution on MPI ranks ? -*

This information was added within the same paragraph as about at lines 158-159.

9. *Line 135 : can you re-run the experiment on 32 MPI ranks, to fixed the distribution problem. -*

This suggestion is addressed within lines 179-183.

10. *Figure 5 : can you add the "ideal speedup" line on it ?*

This plot was modified to contain an ideal speedup line and it was changed to a line plot instead of a bar plot to show this information better. For reference, this plot is now Figure 6 because another figure was suggested from reviewer #1.

11. *line 147 : can you explain what you mean by "flexible interface" ? -*

This line was specifically modified at lines 193-197 to add clarity. We also added two paragraphs that addresses the flexibility starting at line 217 though 227.

12. *Line 148 : can you describe the "task-parallel approach" you choose to implement ? -*

This line was removed at lines 199-200 and it was moved to lines 206-207 with a better explanation.

13. *Lines 152 a 153 : how users that are not experts on CMIP6 (as it's tell several times in the paper for example lines 218 & 219) can know which functionalities need to be create ?*

Through our process we had scientists define these for us. We listed the CMOR variables names along with the descriptions, requested units, and grids and they would note how CESM data would be used to derive the new CMIP6 variables. Their answers were then turned into a text file. An example of a definition text file can be found within the citation found within lines 203-204.

14. *Data Publication – As far as I know PrePARE will check the correspondence between output metadata and what is wait by CMIP6. But it will not check outputs quality (for example : no missing time step on a time-series). Can you present how you manage the quality control of your cmip6 outputs files ? – What happen if PrePARE return problems on outputs cmip6 files ?*

This is now addressed in several places throughout the paper. We describe how we used PrePARE within lines 260-266. We now address time step verification within lines 94-96. We also address quality control within lines 238-239. If problems were found, our procedure is now briefly described within lines 225-227.

15. *Process workflow – can you explain if learning how to use Cylc was easy or not ? Can you estimate time and FTE necessaries for this implementation ?Did you hesitate with another software ? – Maybe it can be useful to add a graphic showing how Cylc is incorporated to your workflow, with the call tree of all your tools. -*

We now address training on our workflow at lines 326-334. We address another software option (Rocoto) at lines 286-287. We also included a workflow graph (or call tree) within Appendix A1.

16. *Line 213 & 215 : I don't understand the difference between "the users set the default values" and "users only needed to set experiment specific information". And if it's "default values" why users need to modified them ? -*

This is clarified within lines 311-312 and lines 314-315.

17. *Is Cylc workflow can solve all errors ? Or is there a need for human intervention from time to time?*

Not all and some information about this at lines 316-319.

18. *Line 229 : "The experiments that . . . no provenance was obtained" : can you precise if it's only for NCAR simulations or for all groups' simulations ? –*

This is now clarified at line 356 to indicate that it was NCAR that did not have a formal way to record for provenance.

19. *Line 251 : can you precise how are managed "simulations that ran into problems" ?*

This was clarified at line 384.

Technical corrections

1. *Line 54 : it's finish by a "," instead of a "." -*

This is corrected at line 60.

2. *Line 55 : "steps including;" need to be modified by "steps including:" –*

This is corrected at line 61.

3. *Line 77 : “Instead the data”, I’m not sure that you want to tell “instead”, maybe “by consequences” or something like this. -*

This is corrected at line number 92.

4. *Line 91 : “this task base parallelism” need to be modified by “this task based parallelism”*

This is corrected at line number 109.

5. *Line 187 : “CMIP6”, I think you want to write “CMIP5” -*

This is corrected at line number 258.

6. *Line 200 : “in order keep track of the statues of all of the running tasks. In order to track the status of all of the tasks …”, maybe you can avoid to write two time “in order . . . tasks”*

The first sentence was re-written and it can be found at lines 288-290.

Author’s Changes

The authors have made the following modifications:

Line 2: large was changed to dramatic in order to emphasize the increase in complexity.

Line 19: run was changed to experiments to be consistent with the language used throughout the paper.

Line 22: the sentence was split in order to accommodate for the extra text that was requested.

Line 60: Grammar fix.

Line 186: Change was needed in order to match the label that was used in Figure 1.

Text in Figure 8: Text was added for clarification purposes.

Line 294-295: Clarify what “This” refers to.

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~~ dramatic 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~~ The large increase in data volume 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~~ 565 TB of compressed data to the community, with another 30 TB yet to be published. When compared to the volumes we produced in the previous generation~~and~~ , 165 TB of uncompressed data, we were able to provide six times the amount of data and we accomplish this within one-third of the time~~, providing an~~ . This provided us an approximate 18 times speedup. ~~This~~ While this paper discusses the improvements we have made to ~~accomplish this success for Phase 6 and further improvements~~ our own workflow for CMIP6, we hope to ~~make for the next generation~~encourage other centers to evaluate and invest in their own workflows in order to be successful in these types of modeling campaigns.

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~~experiments. 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~~ . This consumed over 190 million CPU hours and produced over 2 PB of ~~post-processed data~~ model output time-series data and 600 TB of requested formatted 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~~ post-processing software and 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 human 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~~would take several hours for users to setup and run a full set. 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 that are defined within Taylor et al. (2017) and within the CMIP6 data request (Juckes et al., 2020). For CMIP5, this ~~was also a time consuming process because it lacked parallelism , it was difficult to run , and~~ process also did not contain any parallelism and it was slow to run because of the amount of data that was required to be post-processed. In addition, our software for this process was difficult to run as it required expert knowledge to ensure the data that was generated met the correct MIP standards. After the standardized data are verified by the scientist, 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 to run due to the number of files and total data volume on which they operated. For example during CMIP5,

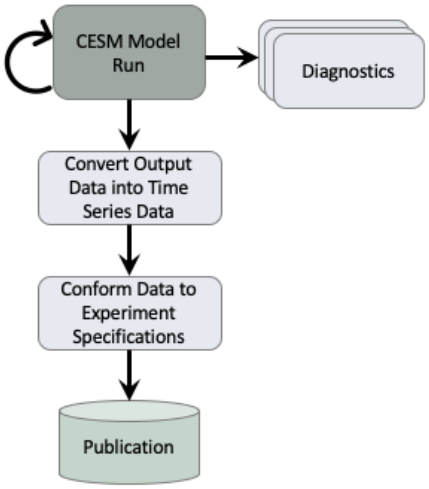

Figure 1. This flowchart describes the tasks that are executed within the CESM workflow in order to generate data for CMIP. The diagnostics task can be executed several times while the CESM model run is executed. The remainder of the tasks are each executed once when the CESM model run completes. This flow chart is a simplistic view of our general workflows. In practice, our workflows are more similar to the workflow depicted in Appendix Figure A1.

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~~ , and minimized 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 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~~ required more people to work on this option 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. These benefits of Python allowed two

full-time employees to create the post-processing tools presented in this section within the three year timeline we needed them completed by.

The We also saw performance issues during 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 The process improvements that were made to ESGF, along with the post-processing tools we developed are described in the following subsections.

Most of the performance improvements that are described in the following subsections were ran on the Cheyenne supercomputer (Cheyenne, 2017). Cheyenne is a 5.34-petaflop machine that contains 4,032 dual-socket nodes. Each node contains two Intel Broadwell processors that are clocked at 2.3-Ghz. For these tests, we used the standard nodes that contain 64 GB of memory per node.

We also provided timing results from the Yellowstone supercomputer (Yellowstone, 2017). Yellowstone was a 1.51-petaflop machine that contained 4,536 nodes, each containing dual Intel Sandy Bridge processors. Each node contained 32 GB of memory. Yellowstone was in operation from 2012 through 2017.

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 In order to increase the usability of the data, the data are reformatted into a time-series format, where each file contains one or more time slices of a single variable.

In addition to transforming the data, this process also needed to verify that all time slices were inserted correctly into each time series file. This involved sorting all of the time slices, verifying that the time values were all unique, ensuring there were no gaps in the time dimension, and correctly inserting the time slices into chronological order.

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 $num^i_{var} * num_{TS} + 1$

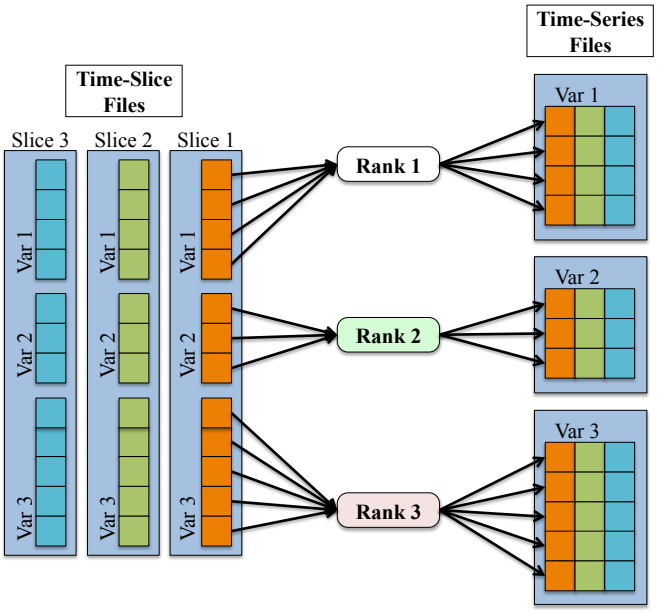

Figure 2. This figure shows the process of converting the data from a time-slice format to a time-series format in parallel within the PyReshaper. Each MPI rank is responsible for taking a particular variable from each time-slice file and writing it to the time-series file.

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~~ based 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.~~ 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~~ This behavior occurs 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.

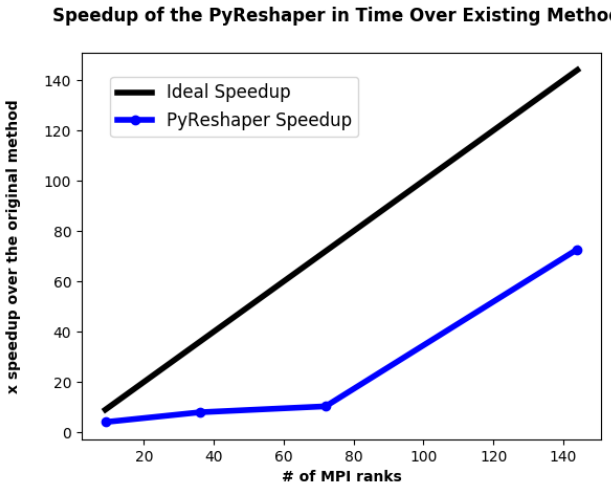

Figure 3. The comparative speedup in time of creating time-series files from ten years of monthly atmospheric data. In all cases, 493 time-series variable files were created. For comparison, the existing methods took approximately 5 ½ hours to complete. With 144 MPI ranks we were able to bring the time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

For this evaluation we did not scale above 144 ranks because this is what we had run in production. As noted above, the PyReshaper does not attempt to load balance between the ranks and as the PyReshaper completes, several ranks remain idle while others still complete their work. We found 144 ranks to be a good balance of resources per throughput based on the average number of three-dimensional and two-dimensional variables, which was verified by running throughput tests.

Future work will be needed to handle the load-balancing within the PyReshaper. Load-balancing techniques that could be implemented include a master-slave task assignment method. Another naive implementation would involve assigning work based on evenly dividing the three-dimensional and two-dimensional variables amongst the ranks. Either method would create a more predictable scaling that would reduce the need to study performance tests based on different problem sizes in order to achieve desired performance.

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~~ used a combination of shell scripts, NCO, and NCL (NCL, 2019) to create a set of average files, or climatology files, ~~plots~~ plotted the data against observations or another model run, and then ~~creates~~ created an HTML document that ~~links~~ linked 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.~~ While NCL was the preferred language to create these plots, with a few modifications, any of the packages could create plots in other languages. An example set of diagnostic plots from our CMIP6/PMIP4 experiments can be found on the web (CESM Diagnostics Results).

Each 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~~ diagnostic packages (Woitaszek et al., November 2011; Jacob et al., 2012), this ~~work presented it own set of issues. Specifically this approach resulting~~ approach resulted 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 number of MPI ranks within a subcommunicator was set to four, unless the total number of ranks was less than four or there were less than four variables that needed to be operated on. If either of those conditions were true, the number of ranks within a subcommunicator was set to two. The total number of subcommunicators was computed by dividing the total of MPI ranks by the number of ranks within a subcommunicator. Once the MPI ranks were evenly distributed to their corresponding subcommunicators, the averages where then assigned evenly amongst the subcommunicators.

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 existing individual NCL scripts and some newly created Python plotting scripts in parallel. ~~Because~~ We are able to execute them in parallel because there are no data dependencies 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 of our improvements, we ~~compared~~ ran original versions of the diagnostic packages and compared them to the time it took ~~to create~~ our new version to create the same climatology files and the same NCL plots. We ran these comparisons on the Yellowstone supercomputer and we used 16 MPI ranks for all PyAverager/NCL in parallel timings. Figure 5 shows that we were able to achieve a 5.8 times speedup for the atmospheric diagnostics, a 6 times speedup for the ice diagnostics, and a 4.6 times speedup for the ocean diagnostics.

In order to evaluate the scalability of the PyAverager, 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 6 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 sub-

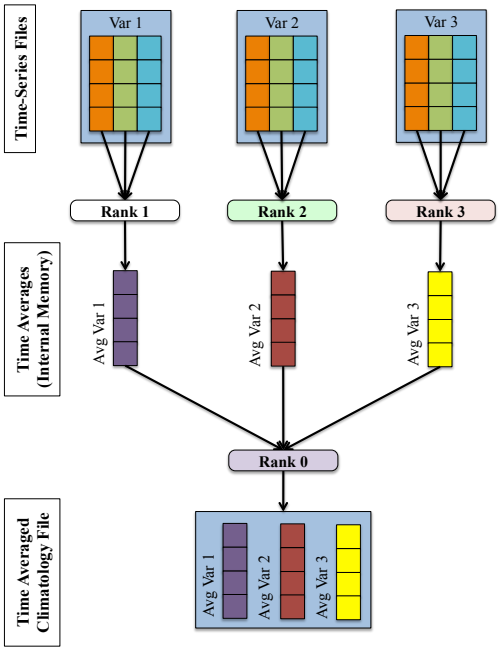

Figure 4. The parallelism strategy for the PyAverager for writing each climatology file. This figure describes the processes in one subcommunicator.

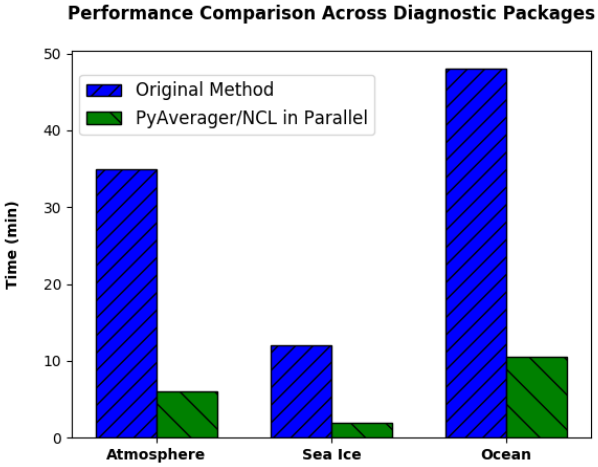

Figure 5. The performance comparison across different diagnostic packages from 10 years of monthly CESM data. These timings include the total time to create all of the required climatology files and to run each of the NCL plotting scripts. The PyAverager/NCL in Parallel timings were all computed using 16 MPI ranks.

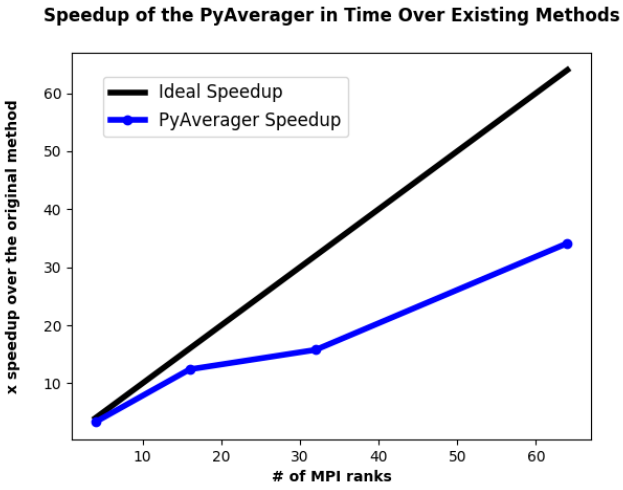

Figure 6. The comparative speedup of creating climatology files from ten years of monthly atmospheric data. Four seasonal and twelve monthly climatology files were created. For comparison, the original methods took approximately 26 ½ minutes to generate the climatologies. The PyAverager took approximately 46 seconds to create the same climatologies with 64 MPI ranks.

communicators 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. This was unavoidable because of the order in which the tasks were assigned. To improve the performance on 32 tasks, we would have to evaluate the problem size before assignment and evenly distribute the work among the subcommunicators. This can be difficult to predict because some calculations can become more expensive under different variable sizes. We chose to avoid this complexity because we were content with the improvements we had seen, but this is something we can improve on in the future.

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 Specifications" task within Figure 1. This requirement is done in order to enable scientists to directly compare the data from different centers without ~~any modifications~~having to perform data transformations that can be error prone. 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~~ recommended method to create the specified data requires users to write code to make required data transformations and to call the Climate Model Output Rewriter (CMOR) (Taylor et al., 2006) library to check for compliance and to add file attributes. ~~This was usually~~ The software we had written and used for CMIP5 used this recommended method, but it was 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~~It was also difficult to extend this software to include the many additional variables added for CMIP6. 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 adapt to changes in requirements more easily, that could create variable output in parallel, and still produced data that met specification requirements.

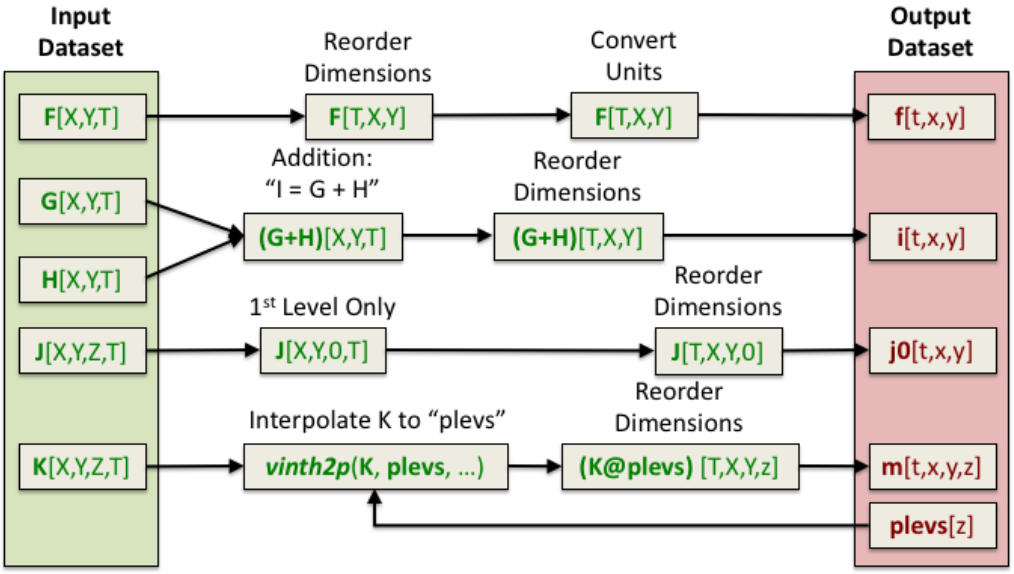

Figure 7. An example of a PyConform job. Each MPI rank is responsible for creating a particular output data set. Its job is to retrieve the variable data it needs, map operations, execute these operations, and then write the data.

An example of a PyConform job is shown in Figure 7. ~~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. We provide an example PyConform CESM model output to MIP compliance definition file (Mickelson, 2019a) to list the available functions and operations that PyConform provides. 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. For this application, we again relied on a task-parallel approach in which an MPI rank was assigned to create one output file. Once the file was written, the MPI rank was given another output file to create.

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.

During the development of PyConform, we chose to keep our `dreqPy` interface code as flexible as possible. `dreqPy` was intended to be an evolving database, adding new fields and experiments in time, and PyConform needed to be able to handle new information without any code modifications. To do this, PyConform queries the `dreqPy` package to obtain experiment and field information. This information is then used within the PyConform software to generate the requested field output files with the correct attributes attached to it.

Flexibility was also needed within the interface used to define how CESM data would be used to derive the variables that were being requested for CMIP6. We chose to use text files (Mickelson, 2019a) to define how these variables would be created. The variable definitions within the text files follow the simple format `cmipvariable=modelvariable`. These variable definitions were provided by and verified by many of the scientists who work on the CESM model. If we needed to make any modifications due to changes within the CESM model, changes within the CMIP6 data request, or if a variable was added, all that was required was to add a line to the text file or modify a line. This allowed us to make modifications quickly because we did not have to modify any Python code. Instead PyConform would see the updated information in the text definition file and automatically use the new definition to create the output file.

The flexibility we added in for the PyConform tool allowed us to fix data quickly if inconsistencies were found. Once we retracted data, we where able to republish data within a few days because we were able to make modifications quickly to a text definition file or simply just read in a new version of the data request and regenerate the data quickly.

In order to evaluate the performance of the PyConform tool, we chose to compare it against the performance of the ~~Fortran code~~ software 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 on the Cheyenne supercomputer, 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. This part of the workflow was not automated. Instead it was a step triggered by the lead scientist once the data had been visually inspected. 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~~ evolving 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.

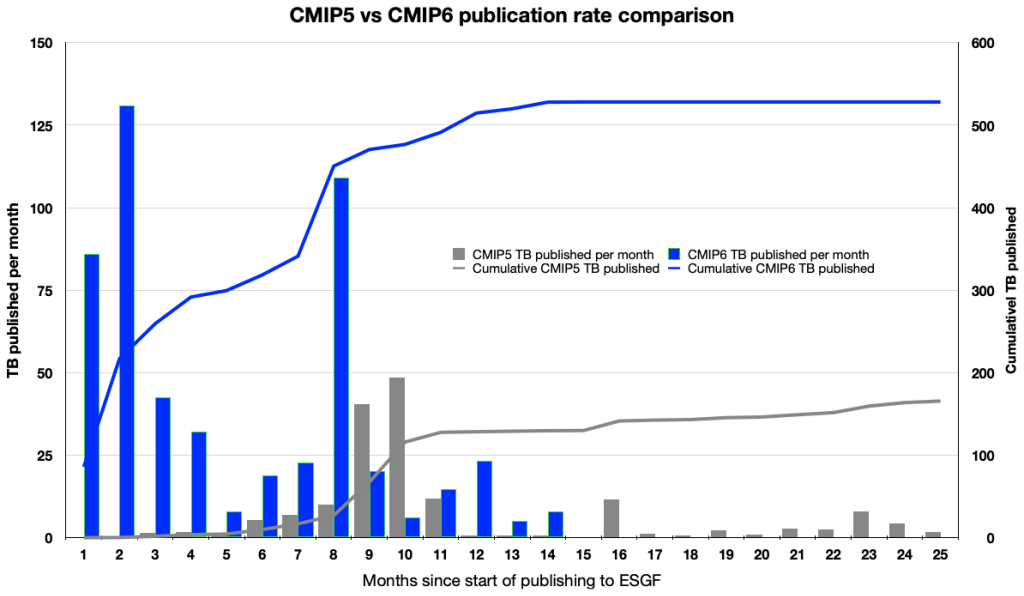

Figure 8. The cumulative and per month increases in the volume of data published to ESGF. During CMIP5, the ESGF software was stressed and problems arose. Despite those problems, NCAR was able to publish 165 TB of data. In preparation for CMIP6, the problems with the ESGF software stack were addressed and through these improvements, NCAR was able to publish 528 TB of data within 14 months, 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.

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)(Petrie et al., 2020) from January to June 2018. These data challenges were performed in advance of the model data availability and served to ~~harden~~ strengthen 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 8. 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~~ CMIP5 campaign because of the improvements that were made. This data (ESGF-NCAR; CMIP6 Data References) is available to download via ESGF.

Another reason why we were able to publish large volumes of data quickly is because we had used a stand alone version of PrePARE (PrePARE) to verify that our data contained all of the correct attributes before we started the publication process. The PrePARE package is part of the CMOR3 (Taylor et al., 2006) package produced by the Program for Climate Model Diagnostics and Intercomparison (PCMDI) group within the Lawrence Livermore National Laboratory. The small problems that PrePARE was able to find allowed us to make corrections before large quantities of data were generated. Once the errors were corrected, PrePARE allowed us to verify that the data that was being created from the PyConform tool met the standards and gave us the confidence that we would be able to pass the publishing quality verification checks.

Another improvement to ESGF involved data versioning. Each ESGF data set is allocated a version number. This allows any data set to be uniquely referenced. Versioning enables data managers to retract any data that may have errors and replace it with a new version without any interruption on any ancillary services. This method of versioning allows all end users to know which data set version was used in their analysis, making data versioning critical for reproducibility.

ES-DOC was used to document climate models that participated in CMIP6 as well as to document the data sets the participating modeling centers published to ESGF (Pascoe et al., 2020). CESM2 has been extensively documented in ES-DOC (ES-DOC Model). Links to the unique ES-DOC documentation pages for each data set published are located within each netCDF file within the CMIP6 collection on ESGF. The link can be located via the global history attribute `further_info_url`.

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, 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 except for the publication task. 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. We also evaluated Rocoto (Harrop, 2017) as a workflow management tool. While Rocoto provided the basic functionality we required, we preferred Cylc's more robust interface and we valued its larger active development community.

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~~ . Both provide intuitive control of the workflow and the individuals 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~~ The ability to automate the submission of all of the tasks we needed to run 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. All of these tasks are shown within an example Cylc dependency workflow graph within Appendix Figure A1 and the Cylc workflow description file used to create this workflow can be found within the CESM-WF repository (Mickelson, 2020b).

Cylc also provided fault-tolerance within the workflows ~~. Cylc allows you~~ by allowing users to specify if ~~you~~ they would like for ~~it~~ Cylc to try rerunning a particular executable if it fails. ~~This was especially helpful when the compute system was unstable. If~~ For example, if one of the model runs failed because of machine error, it was resubmitted to the queue and rerun without user intervention. This became extremely useful when compute nodes on Cheyenne would become unresponsive due to network issues. In these cases, the CESM execution would fail and the non-zero failure exit code triggered Cylc to resubmit the task again. This allowed us to automatically continue our workflow during the many network issues that plagued Cheyenne while we executed these simulations. Without this Cylc feature, we would of had to resubmit the tasks to the queue by hand and this would have caused us to loose productivity.

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 setup of the Python tools discussed in this paper and the Cylc workflow description file. In order to reduce the burden on the users, a Python setup script (Mickelson, 2020a) performed many of the setup steps so users did not need any CMIP6 expertise. Once the users set the ~~default values,~~ run time option values, such as run length and post-processing options, 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, such as specific input file information and output variable names, into the CESM model and to build the model. Then the users started the experiment through Cylc. Human intervention was only needed if the Cheyenne login nodes went down or if the CESM simulation needed to be restarted from the beginning. In each case, users were able to restart the simulation from any point within the workflow. We have made our auto generated Cylc suite definition files that we used to run our CMIP6 experiments available on github (Mickelson et al., 2020).

~~These modifications~~ Once the experiment was started with Cylc, the user running the simulation was able to view the simulation's progress through the GUI or command line interface. Users were also able to pause or stop individual tasks, rerun tasks, or skip tasks. It was also possible to add and remove tasks from the workflow graph after Cylc had started to execute. Cylc also provided process status information for all of the tasks, including start and stop run times and job identification numbers given by the queuing system. This provided our users with the control they needed to run the full experiment and any post-processing task.

None of our workflow users had any experience with the new Python tools we developed nor did they have any experience with Cylc before starting their first CMIP6 experiment. Therefore each of our users needed to be trained. We provided each

of our twenty users a hands on individual training session that lasted roughly two hours. We also answered questions they had via direct email and through an NCAR CMIP6 email group that was setup to only contain the workflow users and a few individuals from the NCAR supercomputing user support group. An addition to this support, we provided documentation (Mickelson, 2019b). The documentation walks through the workflow setup instructions, provides instructions on how to run Cylc, and provides answers to common questions we would receive. As a final method of support, we helped monitor the status of their simulations along with them and provided direct help when needed. We found that these training methods provided the confidence most of our users needed to finish their first simulation and to try setting up their next experiment independently.

The Cylc workflow description file was setup to run each task as a separate job in the PBS scheduler on Cheyenne. Therefore, each CESM model iteration and post-processing task that needed to be performed were separate jobs within the PBS scheduler and they were submitted only when the task the proceeded it finished successfully. Throughout the duration of our CMIP6 experiments, we were able to achieve high throughput of our experiments with low wait times in the PBS queue. As a result, we did not suffer from lower job priorities by not pre-staging our jobs. Though our approach may have achieved slower performance on busier systems that give higher priority to jobs that have been queued for longer periods of time. For running on these systems, it is possible to configure the Cylc workflow description file to submit multiple jobs to the queue at once with dependencies on each other in order to increase job throughput.

The modifications described within this section 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~~ NCAR had 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~~ through CESM XML query commands 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.

The next task was to allow for the experiments to upload ~~their configuration and~~ the specific setup configuration files that were used and the CESM run-time 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~~ run, all of the experiment's configuration and timing files were gathered up and were uploaded to a new svn 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. Noting these changes and when they happen is critical to reproducing an experiment. These changes can be scripted into the experiment's Cylc workflow definition file and this would allow it to be reran exactly how the original was run.

Another feature request was to link the diagnostic package results (CESM Diagnostics 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~~ real-time experiment status within the database. As stated previously, it was difficult to know the status of any given experiment. In order to ~~automate~~ update the run-time status within the database, 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.~~ had stopped running and that had not finished.

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~~ serial methods 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 (ESGF-NCAR; CMIP6 Data References) 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 work we have done to improve the diagnostic packages has inspired new analysis workflows written by our scientists in Python. This work is being designed to run analytics on data that resides on our compute resources and on cloud platforms interchangeably. Current efforts are underway to combine these individual efforts in order to produce new versions of the diagnostic packages.

Having stronger data standards that the community conforms to will help ease the ability to perform intercomparisons across models. As more modeling centers move their data onto the cloud, the interest to compare results between models increases, and the community should make their data as easy to use as possible within these types of analytic workflows. We believe `dreqpy` is a great resource that moves the community towards that direction.

The complexity continues to grow with every generation of CMIP, 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~~ While we present a detailed description of the workflow we chose to use for CMIP6, we hope to encourage other centers to evaluate their own workflows. It is important to consider developing flexibility within these types of workflows as workflow tools should be able to adapt to changes easily. Other important considerations when evaluating workflows include a reduction in the data footprint and an increase of the model data throughput. CMIP exercises are resource-expensive and time-consuming to run and it is important to be prepared for the commitment involved with these types of campaigns.

Appendix A

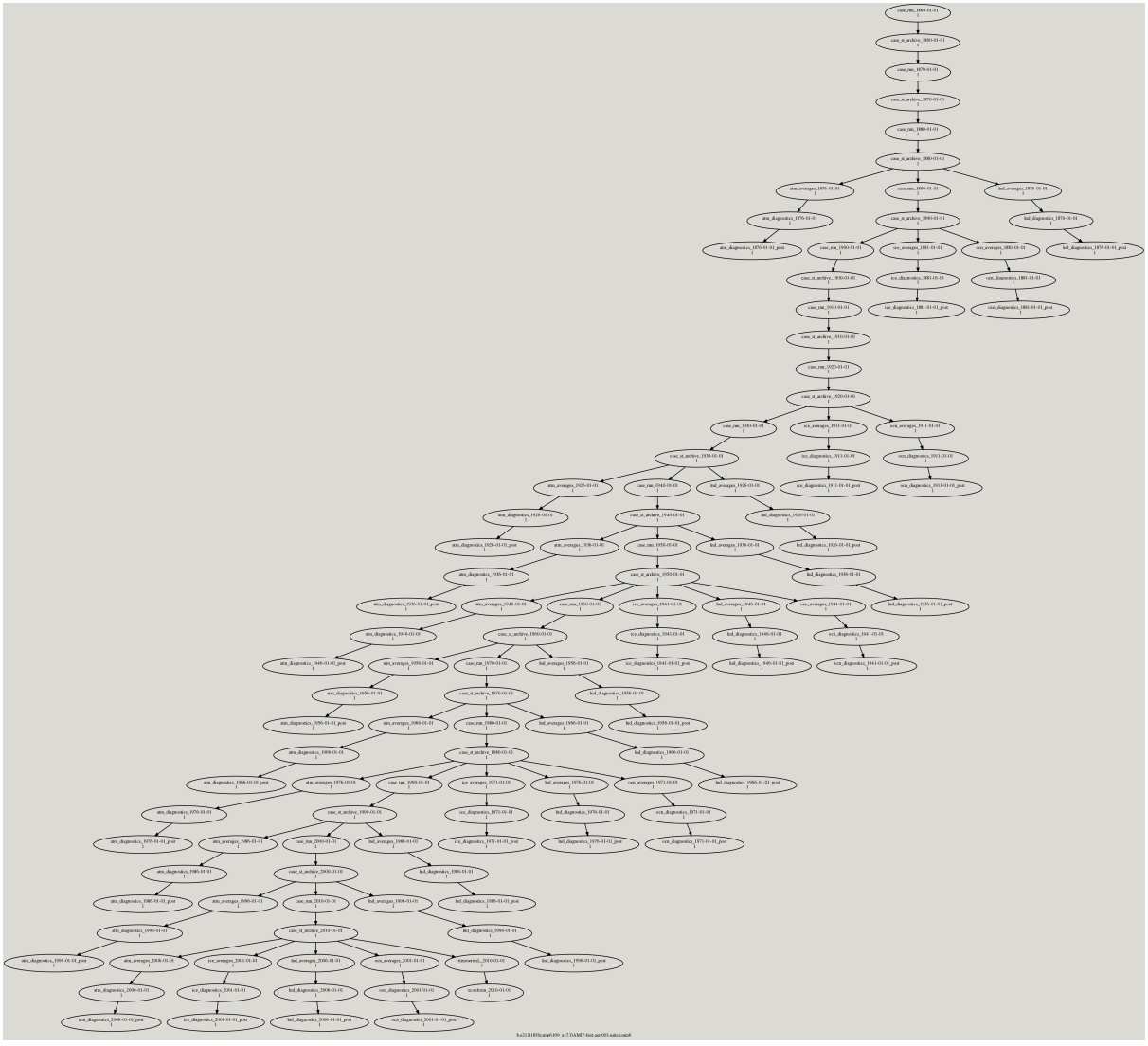

Figure A1. This figure shows a Cylc dependency workflow graph that was generated for an experiment we ran for CMIP6 which required us to simulate the climate from 1850 to 2015. The CESM model and its data movement utility were ran within this workflow. The CESM component diagnostic packages were also ran within this workflow. The ocean and ice model diagnostics were ran after every 30 years of simulation and the atmosphere and land model diagnostics were ran after every 10 years of simulation. The different diagnostic packages were ran as part of the many three chained tasks that are leaves within the workflow graph. The first task runs the PyAverager to generate the climatologies. The second task runs the NCL scripts in parallel to generate the plots and the web pages they are displayed on. The third task is a post command that copies the image and web files onto the web server. The PyReshaper and the PyConform tools were ran at the end of the simulation.

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, 2020a). 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.

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.

CESM Diagnostics Results: CESM Diagnostics, http://www.cesm.ucar.edu/experiments/cesm2.1/PMIP4/, 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.

CMIP6 Data References: CMIP6 Data References, http://bit.ly/CMIP6_Citation_Search, 2020.

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.

ES-DOC Model: ES-DOC Model, https://explore.es-doc.org/cmip6/models/ncar/cesm2, 2020.

ESGF-NCAR: CMIP6 Data Search, https://esgf-node.llnl.gov/search/cmip6/?institution_id=NCAR, 2020.

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.

Harrop, C.: Rocoto, https://doi.org/10.5281/zenodo.890939, hhttps://github.com/christopherwharrop/rocoto, 2017.

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.

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 PyConform Input Example, https://doi.org/10.5281/zenodo.3983646, https://github.com/NCAR/conform-input/blob/master/cmip6_input/CESM_MastList.def, 2019a.

Mickelson, S.: CESM Workflow Documentation, https://cesm-wf-documentation.readthedocs.io/en/latest/, 2019b.

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

Mickelson, S.: Example Cylc Description File, https://github.com/NCAR/CESM-WF/blob/master/example_cylc_wf/suite.rc, 2020b.

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

Mickelson, S., Altuntas, A., Bertini, A., Benedict, J., Coleman, D., Fasullo, J., Feng, R., Hannay, C., Lawrence, P., Lindsay, K., Medeiros, B., Mills, M., Oleson, K., Rosenbloom, N., Strand, G., Sun, L., Thayer-Calder, K., Tilmes, S., and Tomas, R.: CESM CMIP6 Cylc Suites, https://doi.org/10.5281/zenodo.3983653, https://github.com/NCAR/CESM_CMIP6_Cylc_Suites, 2020.

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.

Pascoe, C., Lawrence, B. N., Guilyardi, E., Juckes, M., and Taylor, K. E.: Documenting numerical experiments in support of the Coupled Model Intercomparison Project Phase 6 (CMIP6), Geoscientific Model Development, 13, 2149–2167, https://doi.org/10.5194/gmd-13-2149-2020, 2020.

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.

Petrie, R., Denvil, S., Ames, S., Levavasseur, G., Fiore, S., Allen, C., Antonio, F., Berger, K., Bretonnière, P.-A., Cinquini, L., Dart, E., Dwarakanath, P., Druken, K., Evans, B., Franchistéguy, L., Gardoll, S., Gerbier, E., Greenslade, M., Hassell, D., Iwi, A., Juckes, M., Kindermann, S., Lacinski, L., Mirto, M., Nasser, A. B., Nassisi, P., Nienhouse, E., Nikonov, S., Nuzzo, A., Richards, C., Ridzwan, S., Rixen, M., Serradell, K., Snow, K., Stephens, A., Stockhause, M., Vahlenkamp, H., and Wagner, R.: Coordinating an operational data

distribution network for CMIP6 data, Geoscientific Model Development Discussions, pp. 1–22, https://doi.org/10.5194/gmd-2020-153, 2020.

PrePARE: PrePARE, https://cmor.llnl.gov/mydoc_cmip6_validator/, 2020.

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., Juckes, M., Balaji, V., Cinquini, L., Denvil, S., Durack, P., Elkington, M., Guilyardi, E., Kharin, S., Lautenschlager, M., Lawrence, B., Nadeau, D., and Stockhause, M.: CMIP6 Global Attributes, DRS, Filenames, Directory Structure, and CVs, https://www.earthsystemcog.org/site_media/projects/wip/CMIP6_global_attributes_filenames_CVs_v6.2.6.pdf, 2017.

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.

Yellowstone: Yellowstone, https://www2.cisl.ucar.edu/supercomputer/yellowstone, 2017.

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.