UCAR Software Engineering Assembly
University Corporation for Atmospheric Research

NCAR | National Center for
UCAR | Atmospheric Research

Proceedings of the **2020** **Improving Scientific Software Conference**

Editors
Weiming Hu
Davide Del Vento
Shiquan Su

NCAR Technical Notes
NCAR/TN-564+PROC

National Center for Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

NCAR/TN-564+PROC
NCAR Technical Note

2020-05

Proceedings of the 2020 Improving Scientific Software Conference

Editors
Weiming Hu
Davide Del Vento
Shiquan Su

NCAR Laboratory
NCAR Division

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
P. O. Box 3000
BOULDER, COLORADO 80307-3000
ISSN Print Edition 2153-2397
ISSN Electronic Edition 2153-2400

How to Cite this Document:

Hu, Weiming, Davide Del Vento, Shiquan Su, (Eds.). (2020). Proceedings of the 2020 Improving Scientific Software Conference (No. NCAR/TN-564 +PROC). doi:10.5065/P2JJ-9878

UCAR Software Engineering Assembly
University Corporation for Atmospheric Research

Information about future workshops and other SEA news can be found on our website, --> https://sea.ucar.edu/sea

The website of the 2020 workshop is https://sea.ucar.edu/conference/2020

To be added to the workshop mailing list, please send an email to taysia@ucar.edu.

Proceedings of the 2020 Improving Scientific Software Conference

Table of Contents

Foreword by the ISS 2020 Conference Chairs

Organizing the ISS 2020 Conference

The Software Engineering Assembly (SEA) is a loosely structured group of software engineers and scientists who write scientific software, mostly but not only at NCAR and mostly but not only in the field of atmospheric sciences. The mission of the SEA is to foster a sense of community for software professionals within UCAR, to facilitate effective participation through cross-cutting interactions in UCAR's science mission, to enhance communication with management in matters of concern to software matters, and to engage UCAR software professionals as full partners in setting priorities for the design, development and maintenance of the software infrastructure needed to realize UCAR's mission.

Among many initiatives to accomplish its mission, the SEA organizes a conference series, the Improving Scientific Software (ISS) Conference, formerly called simply SEA conference, which started in 2012. For the 2020 edition, the solicitation called for the following topics:

- Influence of computer architecture on scientific software design, including but not limited to:
    - Modern architectures (such as GPU, ARM, etc)
    - Post-modern architectures (such as quantum computing, neuromorphic, etc.)
    - Extreme heterogeneity (intra-node and inter-system)
    - Edge computing
    - Communication/computation overlap
    - Performance, portability & productivity
- Modern tools for
    - Data analysis, processing and visualization
    - Scientific workflows: purpose and product review.
- Machine learning for scientific computing
- Containers in scientific software
- Leveraging cloud computing resources for HPC development and operations

- Any other topic relevant to Improving Scientific Software

Moreover, for the first time in the conference history, it was decided to ask the speakers to provide an accompanying paper, which would have been peer reviewed and published as part of the proceedings. Being a first, the request was optional, and the organizing committee decided to publish it through the NCAR Library rather than through a higher-profile organization such as ACM or IEEE.

Response by the Community and Conference Cancellation

By the hard work of the conference committee, and by the engagement of the community, a very interesting program was put together, consisting of 21 accepted talks and three hands-on tutorials. Everything was in place and we were ready to start, when COVID-19 struck Colorado in mid-March. The organizers were already watching the COVID-19 situation unfolding elsewhere in the world and discussing contingency plans when UCAR decided to close its facilities to the public and to non-essential staff until further notice. In fact, everybody in the conference committee had just started the mandatory working from home when a decision needed to be made about the conference. Rather than trying to make a decision from the top while trying to navigate these uncharted waters, a survey was sent to all the speakers. The results showed 70% of respondents preferred canceling or rescheduling the conference as a first choice and 36% said it was their second choice. Many said they would withdraw from an online-only event and that the in-person networking is the most important aspect of this event. **Based on that, we decided to cancel the ISS 2020 conference.** Rescheduling anytime before the planned date for next year presented several obstacles. *For next year's program, we will give priority to authors whose talks and tutorials were accepted for 2020.*

While the conference itself was canceled, the organizing committee decided to go ahead with peer-reviewed proceedings publication for this year. Given the circumstances, we worked with authors to facilitate this as much as possible, including relaxing the deadlines as much as reasonable. Even so, only eight of the planned speakers agreed to submit papers for the proceedings. Eventually, because of challenges caused by the pandemic, only three papers were fully submitted by the deadline and underwent rigorous blind peer review by two reviewers per paper. They represent the best examples of the state of the practice in scientific software development, and disseminating them will contribute to improving scientific software, which is the mission of the conference.

Even if these proceedings contain only three papers, the organizing committee is proud of the work done by everybody involved in the process, starting obviously from the authors, continuing to the peer reviewers, who should remain anonymous, the NCAR Library, the admins and all the members of the committee.

Organizing Committee

Conference Chairs

Davide Del Vento, National Center for Atmospheric Research
Shiquan Su, National Center for Atmospheric Research

Program Committee Chairs

Davide Del Vento, National Center for Atmospheric Research
Shiquan Su, National Center for Atmospheric Research
Weiming Hu, Penn State University

Proceedings Committee

Weiming Hu, Penn State University
Davide Del Vento, National Center for Atmospheric Research
Shiquan Su, National Center for Atmospheric Research
Michael Flanagan, National Center for Atmospheric Research
Taysia Peterson, National Center for Atmospheric Research

Steering Committee

Davide Del Vento, National Center for Atmospheric Research
Shiquan Su, National Center for Atmospheric Research
Weiming Hu, Penn State University
Maggie Sleziak, University Corporation for Atmospheric Research
Andrew Younge, Sandia National Laboratories
Pat Nichols, Los Alamos National Lab
Srinath Vadlamani, Arm
Joseph Schoonover, Fluid Numerics
Anderson Banihirwe, University Corporation for Atmospheric Research
Guido Cervone, Penn State University
Sanjiv Pradhanang, CIRES/NOAA
Keith Maull, University Corporation for Atmospheric Research

Workshop Administrator

Taysia Peterson, National Center for Atmospheric Research

This page is intentionally left blank.

PARALLEL ANALOG ENSEMBLE – THE POWER OF WEATHER ANALOGS

Weiming Hu[1], Guido Cervone[1], Laura Clemente-Harding[2], and Martina Calovi[3]

1 Department of Geography, Institute for Computational and Data Sciences
2 Geospatial Research Laboratory, Engineer Research and Development Center
3 Department of Ecosystem Science and Management, Earth and Environmental Systems Institute,
Institute for Computational and Data Sciences, The Pennsylvania State University

The Parallel Analog Ensemble (PAnEn) is an open-source, flexible, and scalable library to generate accurate ensemble forecasts from a single deterministic weather forecast and a set of corresponding observations. The Analog Ensemble (AnEn) technique has been shown to generate accurate and reliable weather forecasts. This package implements the Analog Ensemble technique with the goals of scalability and usability, while preserving its accuracy and reliability. Profiling results are presented to test the scalability of the PAnEn implementation.
PAnEn is implemented in C++ using the OpenMP library and the MPI standard for performance and portability purposes, and it is best suited for a multi-node and multi-core environment. Additionally, an R interface is provided along with various helper functions to facilitate tasks like data preparation, forecast verification, and visualization. Results have shown that the Analog Ensemble technique is highly parallelizable and that the PAnEn implementation is able to achieve over 95% parallelization on a single node. When multiple nodes are used, the code has been optimized to maintain a low overhead. Tutorials and complete documentation can be accessed from the project website.

CCS Concepts: • **General and reference → General conference proceedings**; • **Applied computing → Earth and atmospheric sciences**.

Additional Key Words and Phrases: ensemble forecasts, parallel computing, high-performance computing

Suggested citation:
Weiming Hu, Guido Cervone, Laura Clemente-Harding, and Martina Calovi. 2020. Parallel Analog Ensemble – The Power of Weather Analogs. In *Proceedings of NCAR Software Engineering Assembly's Improving Scientific Software Conference and Tutorials 2020 (2020 SEA ISS).* Boulder, CO, USA, 14 pages. https://dx.doi.org/10.5065/P2JJ-9878

1 INTRODUCTION

Weather analogs are weather phenomena that share similar main features. Analog forecasting usually involves searching for historical weather analogs and then predicting future weather based on historically similar cases [38]. Upon its emergence back in the 1970s, analog-based approaches to weather forecasting did not receive much notice. The main reasons include: 1) the best analog using a short historical sequence (e.g. five years) is usually only mediocre [22]; 2) older computers could not handle large computing tasks, and model and observation data were insufficient [35]. After 1990, there has been a slow rejuvenation of the analog-based approach in weather forecasting [3, 20, 25], and it was mainly used as a Model Output Statistics (MOS) method to improve weather model predictions.

Ensemble forecasts are becoming widely known recently. They are found to yield more accurate predictions than the counterpart deterministic model prediction even when the forecast ensembles are summarized with a single value [17, 40]. Forecast ensembles also provide a series of possible future conditions of the variable of interest and they can be used as an agent for forecast uncertainty [7]. Examples of current atmospheric ensemble forecast models include the Global Ensemble Forecast System, the Short Range Ensemble Forecast, and the European Center for Modeling Medium-Range Weather Forecasts Ensemble Prediction System. These models are usually categorized as dynamical ensemble models because their generation typically requires running either multiple models, the same model with different parameterization, or a combination of both. Dynamically generated ensembles generally require supercomputers as they have large computational consumption.

Another type of forecast ensembles is the statistical ensemble. Statistical ensembles are usually generated on top of existing dynamical ensembles. Ensemble Model Output Statistics [15, 30] is a calibration method to improve the ensemble statistical consistency. It builds a calibrated Probability Distribution Function (PDF) from the existing ensemble members and the predictive PDF can then be used to inform the future atmospheric state with uncertainty. Random samples are usually drawn from the built PDF to generate statistical ensembles. *ensembleMOS* [41] provides an open source implementation in R. Another type of statistical ensembles is the daughter ensemble [28]. It refers to an ensemble generated around an individual member of a dynamical ensemble using forecast error statistics. The individual member can be a "best guess" out of the dynamical ensemble members.

The AnEn [11] is the first analog-based approach to directly generate weather forecast ensembles from a deterministic prediction model. This technique has been successfully applied not only to weather variables [14, 18, 33], but also to economic variables, like solar photovoltaic energy [2, 4] and wind power [1, 31]. The AnEn shows great success as an analog-based weather forecasting approach with the current advancement on both the computing and data availability. One of the key differences between the AnEn and previous versions of analog-based approaches [35, 36, 38, 39] is that instead of finding analogs within a large spatial and temporal window, weather analogs are sought with a much finer spatial and temporal resolution, usually at a single grid level and within several forecast lead times. This allows high quality weather analogs to be found even with a short period of historical dataset, for example with one or two years.

With the broadening impact and growing application of the AnEn, this paper introduces a library that facilitates a complete workflow to generate an AnEn, the Parallel Analog Ensemble (PAnEn) [19]. This is the first open source library hosted on GitHub for the AnEn implementation. The AnEn is intrinsically a scalable algorithm suited for high-performance computing [4], and it is optimized to provide an efficient and scalable implementation.

In this paper, a brief description of the AnEn technique is first provided. Then it dives into the architectural design of the library. Multi-threading and multi-node profiling analysis are carried out to assess the performance of the library. Last, some typical usage of the library is introduced. Below is a list for additional documentation and tutorials for PAnEn:

- Package landing page: https://weiming-hu.github.io/AnalogsEnsemble
- Source code repository: https://github.com/Weiming-Hu/AnalogsEnsemble
- Full documentation: https://weiming-hu.github.io/AnalogsEnsemble/doc
- Tutorials: https://github.com/Weiming-Hu/AnalogsEnsemble/tree/master/RAnalogs/examples

2 LIBRARY ARCHITECTURE

The PAnEn is implemented with the 2011 C++ standard. It builds upon a list of dependencies including Boost C++ [29] and NetCDF libraries [21, 27]. The complete list can be found in the full documentation. The R [26] interface depends on Rcpp [12] which provides seamless integration between C++ and R codes. Tests are designed and implemented using the CppUnit [23] framework. Parallelization has been achieved with OpenMP [5, 6, 10] and the Message Passing Interface (MPI) standard [16].

The package has been successfully installed and tested on some of the popular operating systems, including CentOS, Ubuntu, Mac OS, and Windows. The tested compilers include GNU GCC (>= 4.9) with OpenMP (>= 4.0) and Intel C++ compilers (>= 18.0.0). The R interface has been tested with R versions later than 3.0.0. The MPI implementation has been tested with OpenMPI (>= 4.0.0) and Intel MPI (>= 18.0.0). Despite the limited list, the package is expected to work successfully with other types of environment setup.

The core technique is the AnEn which was originally proposed by Delle Monache et al.. It is an efficient approach to generate forecast ensembles from a single run of a deterministic weather model and a set of historical corresponding observations. A high-level overview of the technique is provided here to facilitate further reading. For an introductory primer on the AnEn technique, please review *An Introduction to the Analog Ensemble Technique* [8].

(1) A historical archive of operational weather model predictions and corresponding observations are required as input.
(2) A test period and a search period [1] are specified together with other configuration options including the number of ensemble members and predictor weights.
(3) A similarity metric is calculated between each test prediction and all search predictions. This step typically involves applying the predictor weights and calculating multi-variate Euclidean distances between pairs of test and search predictions. For details about the similarity metric, please refer to the original paper [11].
(4) After similarity calculation is complete for a test prediction, predictions with the lowest scores (being the most similar) are identified. Note that these predictions are historical and therefore, the corresponding observations are available.
(5) The corresponding observations are collected as members of the forecast ensemble generated for the particular test prediction.
(6) Repeat the steps 2 - 5 for all geographic locations and forecast lead times.

The rest of the section will introduce the implementation of the PAnEn in detail. The implementation guide first covers the data structure design to represent various components required in the technique, including *Forecasts* and *Observations* classes, followed by an overview of the AnEn implementation. Finally, it introduces R extensions designed for rapid research development and the broader scientific community.

2.1 Data Structure

Figure 1 demonstrates the Unified Modeling Language (UML) diagram for the data structure design. The overall data structure is built upon three abstract classes, shown in green, *Array4D*, *Forecasts*, and *Observations*, and the AnEn technique interface is built upon these abstract classes. An abstract class is an interface describing the behavior or capabilities of a C++ class without committing to a particular implementation of that class. Therefore, although these classes do not have actual implementation yet, they are of key importance that they describe what subsequent classes should be capable of doing to be able to be integrated into the AnEn technique. Classes *Observations* and *Forecasts* are designed for representing meta information associated with

[1]In the original paper by Delle Monache et al., this is also referred to as the "training period".

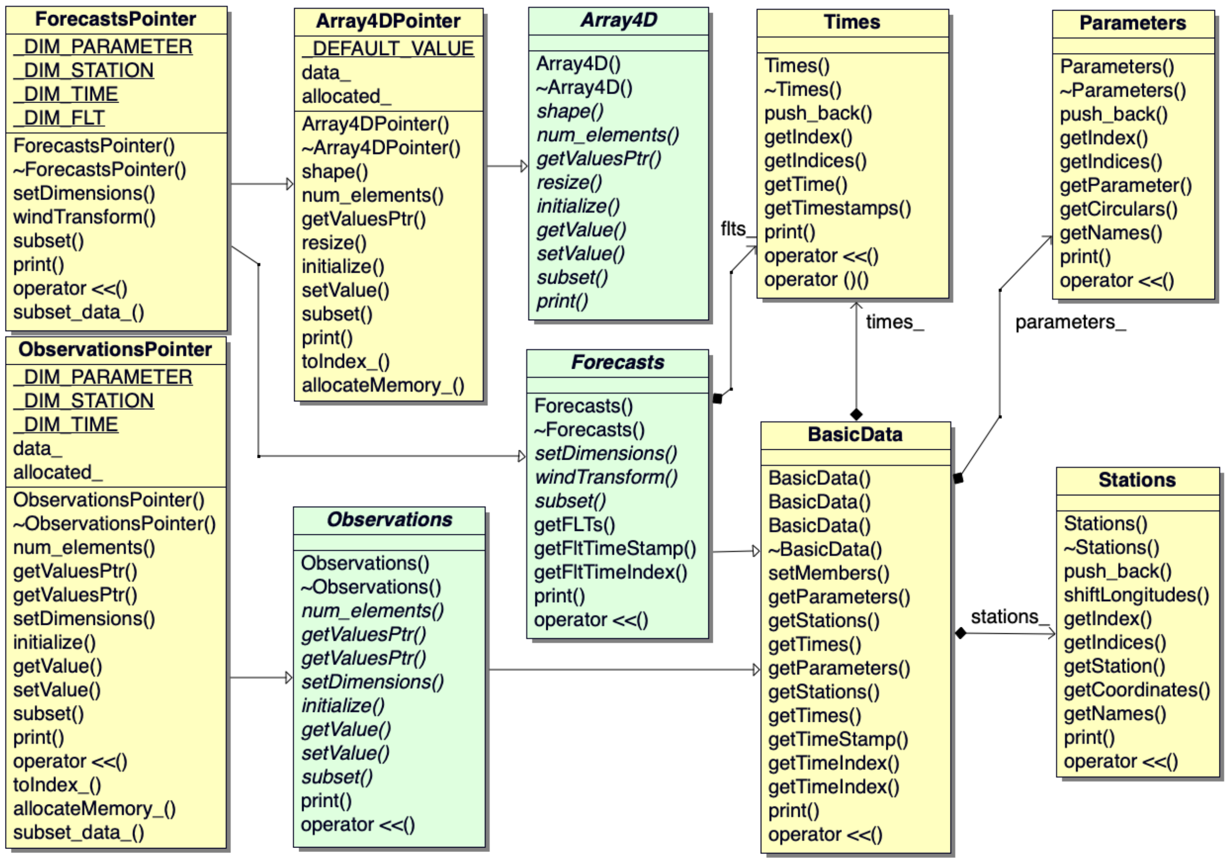


Fig. 1. UML diagram describing the design of the data structure. Abstract classes are shown in green and implementation classes are shown in yellow.

observations and forecasts. Forecasts are usually four-dimensional including predictors (e.g. weather variables), stations, forecast initialization times, and forecast lead times while the structure of observations is similar except that observations only have times of measurement rather than forecast initialization and lead times. Class *Array4D* is designed for storing a generalized four-dimensional data structure.

Classes *ForecastsPointer* and *ObservationsPointer* are derived from their parent abstract classes respectively. As the names suggest, these classes are implemented using C pointers for performance purposes. In fact, class *ForecastsPointer* is inherited from both *Forecasts* and *Array4DPointer* because *Forecasts* describes interfaces for meta information access and *Array4DPointer* describes how the high-dimensional data portion of forecasts are stored and accessed.

Forecasts and *Observations* are both inherited from class *BasicData* because they both have parameters, stations, and times except that class *Forecasts* has an extra dimension of forecast lead time. *Stations*, *Parameters*, and *Times* are implemented as bidirectional maps to carry out fast queries from indices to values and from values to indices.

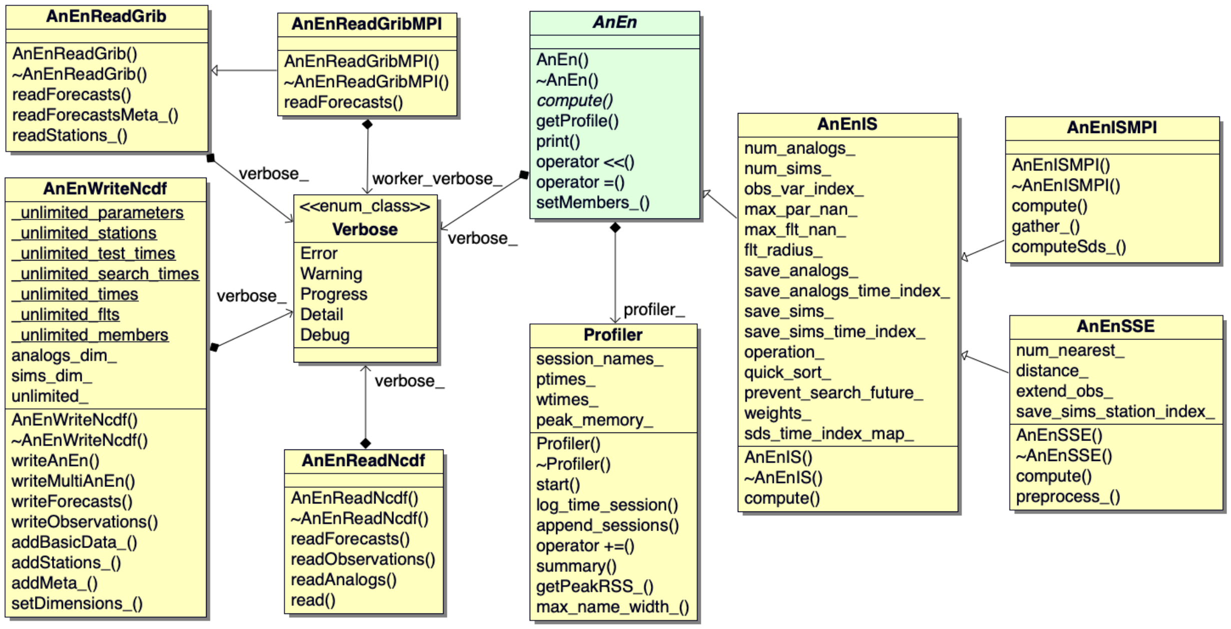


Fig. 2. UML diagram describing the design of the AnEn technique and the file I/O processes. Abstract classes are shown in green and implementation classes are shown in yellow.

As a result, the above architecture achieves the following:

(1) Interface stays agnostic of the underlying implementation. *Forecasts*, *Observations*, and *Array4D* are currently represented as C pointers, but this behavior can be easily swapped to other types of implementation.
(2) Time and space alignment is critical for the AnEn and this design ensures that, once forecasts and observations are created, the objects are only mutable through public interfaces and the alignment is handled internally to prevent runtime errors.

2.2 Analog Ensemble Implementation

Figure 2 is the UML diagram describing the design of file I/O and the AnEn. The abstract class *AnEn* is shown in green. Being the interface of the technique, class *AnEn* provides a method, *AnEn::compute*, which takes *Forecasts* and *Observations* as input to generate weather analogs. Additional arguments can be passed to the class constructor to fine-tune the technique performance.

Building on top of *AnEn*, *AnEn IS* provides the multi-threading implementation of the AnEn with OpenMP for Independent Search (IS). IS refers to the concept that similar historical forecasts are only sought at the current forecast location. It is implemented in the class method *AnEnIS::compute*. There are various member variables of *AnEnIS* to configure the behavior of the method. For example, the number of analogs specifies the number of ensemble members to generate and the toggle switch of whether to use operational mode. When operational mode is turned on, the search period becomes variable and it grows with the progression of test period. Test predictions will be compared with all historical predictions available from the immediate past predictions. Class *AnEnIS* enables two other class implementations, *AnEnSSE* and *AnEnISMPI*. Class *AnEnISMPI* is the parallelization of *AnEnIS* using

MPI for distributed memory systems. Class *AnEnSSE* is a variant of *AnEnIS* for Search Space Extension (SSE). This version of AnEn searches similar predictions from nearby stations in the spatial domain to increase the search repository size and have more available historical predictions. Class *AnEnSSE* provides additional arguments to configure the method including the number of nearest neighbors to search and the distance threshold.

Currently, PAnEn has NetCDF and WMO GRIB2 support since these two formats are widely used in Geoscience. To be consistent with the data structure interface, I/O functions accept abstract class in the interface. GRIB and NetCDF files can be read using classes *AnEnReadGrib* and *AnEnReadNcdf* while the output of the PAnEn can be written to NetCDF files with class *AnEnWriteNcdf*. File I/O can potentially be a bottleneck for computationally expensive tasks. In the case of numerical weather simulation, forecasts are usually saved to GRIB files and each GRIB file contains simulation results for the entire domain at a certain cycle time, initialization time, and forecast lead time. This often leads to a enormously large number of GRIB files to read. Class *AnEnGreadGribMPI* is designed to solve this problem. It is implemented with ecCodes from European Center for Medium Weather Forecasting (ECMWF) to decode binary GRIB messages. The parallelization is enabled through MPI to facilitate scaling.

3 PERFORMANCE ANALYSIS

This section shows results from performance analysis of PAnEn. Some key questions that are addressed in this section are:

(1) How are time and memory consumed during the execution of the program?
(2) How efficient is the multi-threading and multi-node parallelization implementation?

To assess the library performance, numerical weather model simulation has been collected from North American Mesoscale Model (NAM) [13, 37]. The three-year time period spans from January 1, 2016, to December 31, 2018. The archived dataset exceeds 3 Tb in size and is stored on National Center for Atmospheric Research (NCAR) Cheyenne [9] supercomputer. This dataset covers North America with 262,792 grids. It includes 395 weather variables on different vertical layers and forecast lead times are subset from the original 84 hours down to 36 hours.

Multi-threading analysis was carried out on a Dell Precision 7920 workstation with 64 GB of memory and 16 physical cores (equivalent to 32 CPUs with hyper-threading). The operating system was Ubuntu 18.04.1 LTS Bionic. PAnEn was compiled with GCC 7.3.0 with OpenMP support. Multi-node analysis was carried out on NCAR Cheyenne supercomputers.

While numerous profiling tools are available, e.g. GProf [8] and TAU [32], profiling information is generated and gathered internally by the program for simplicity. The execution runtime is measured based on the frequency clock of processors and the wall time functionality from OpenMP. Information on the memory consumption is collected by reading from the system log file. All profiling results are averaged from three runs of the program given the same analysis condition.

3.1 Resource Consumption Analysis

Figure 3 shows the itemized profiling for PAnEn. Figure 3a shows the time profiling of the master process for generating 10-member ensembles over 5,000 stations for one year as test using two years as search. This was carried out on Cheyenne requesting one node with 36 MPI processes. The three stages that take up most of the runtime are forecast and observation reading and analog generation. It is expected that analog generation takes up a significant amount of runtime but the fact that file I/O can also consume a big chunk of runtime should be alarming. In this example, the three-year forecast data, having multiple lead times and cycle times per day, amount to a total of 59,317 files to read. Additional to the sheer number of total files to read, reading a subset (5,000 out of 262,792) of the files can also lead to significant overhead because the file I/O process needs to query and locate a specific

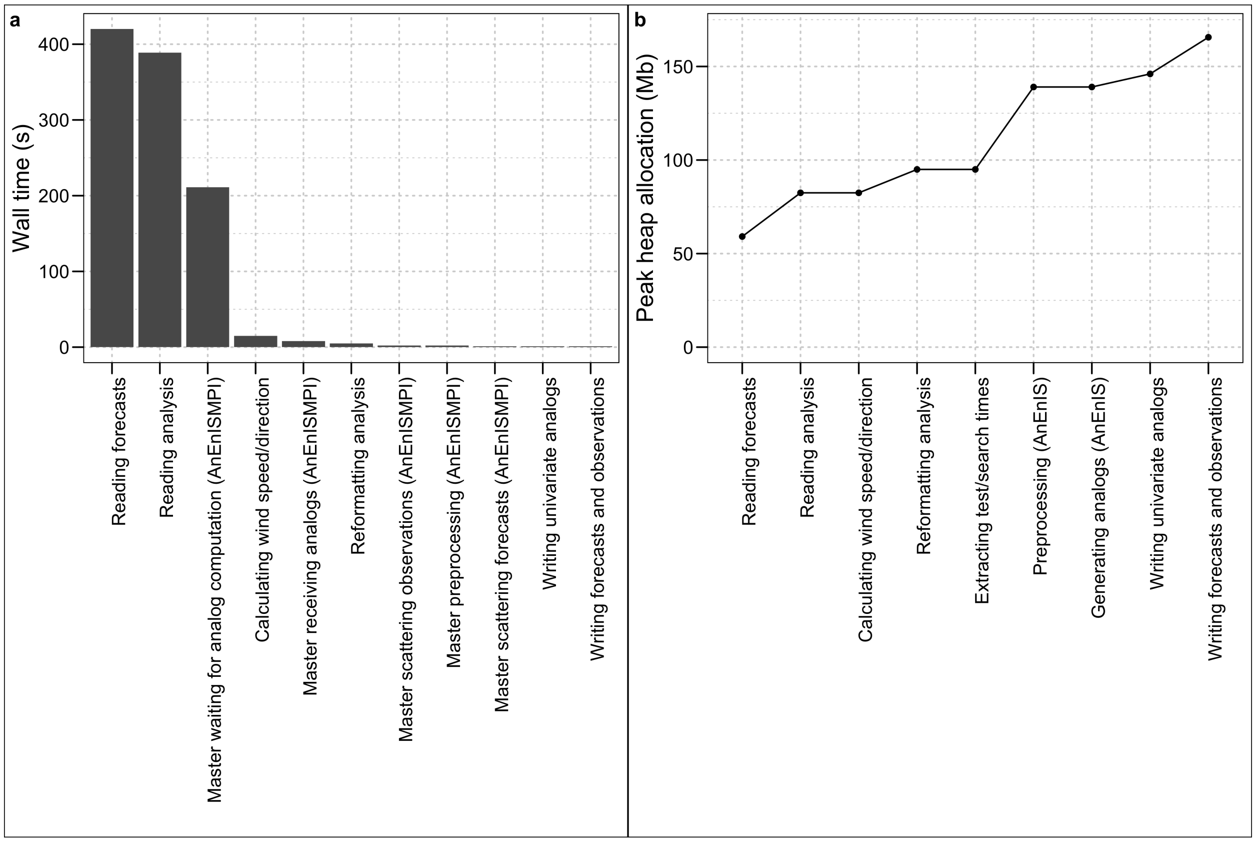


Fig. 3. (a) Wall time profiling of the master process sorted by the amount of time spent in each execution stage; (b) Peak heap allocation profiling organized in the order of runtime execution of each stage

message from the file and most of the time, messages are not contiguous to each other which means the I/O process needs to jump back and forth.

Two options are offered to bypass the file I/O bottleneck stage. GRIB data can be subset during the data preprocessing stage to significantly shrink the size of the dataset. Fortunately, this only needs to be done once, thus remain a constant factor, albeit sometimes a large constant. In cases where the constant becomes too large to manage, file I/O can be parallelized with MPI with high efficiency. MPI profiling is shown in the next section.

Figure 3b shows the peak heap allocation after each execution stage. This metric is used to profile the memory usage of the program. 10-member analog ensembles are generated for 1 month of test and 3 months of search using 1 nodes. This experiment was not run on multiple nodes because multi-node memory profiling for MPI can be inaccurate. As Figure 3b suggests, three significant jumps of memory usage occur during the file reading process (reading forecasts and analysis), analog generation preprocessing, and writing analog results. The last stage of the execution outputs forecasts and observations in a NetCDF format from the test period. This is a feature of PAnEn which enables further verification and analysis.

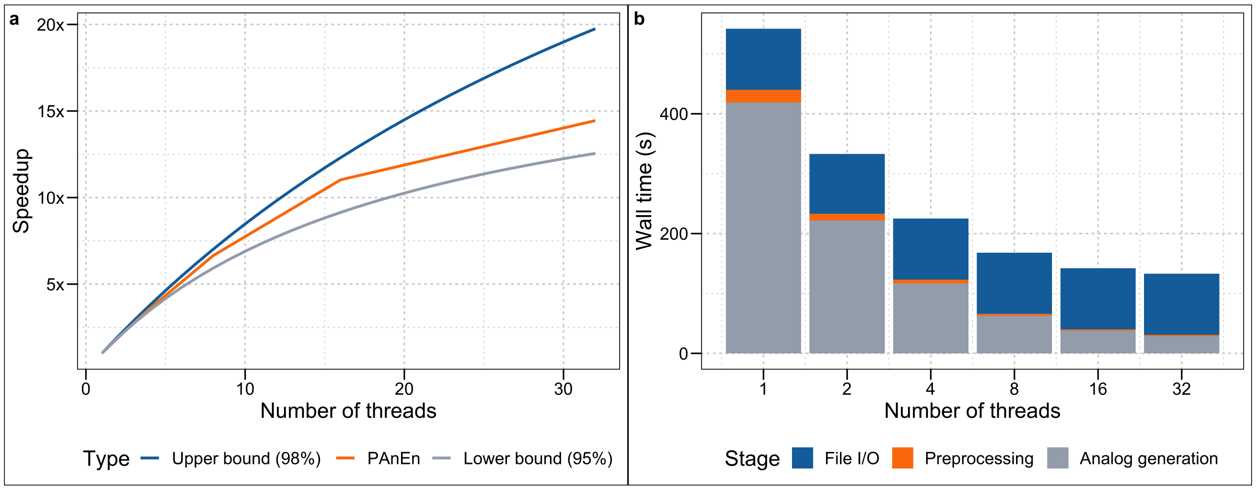


Fig. 4. (a) Multi-threading profiling of speedup; (b) multi-threading profiling of runtime

4 MULTI-THREADING AND MULTI-NODE PROFILING

Section 3.1 identifies two potential bottlenecks of the PAnEn. The first bottleneck is analog generation which is the core of the technique; the second bottleneck is the file reading stage when a long period of forecasts are required. The PAnEn solves these problems with a hybrid approach of OpenMP and MPI. When the simulation can be carried out on a single computing node, multi-threading can be used to speed up the generation of weather analogs. For large problems where weather analogs are sought from a multi-year forecast repository for over thousands of stations, MPI can be used on top of OpenMP to support large scale computation.

Figure 4 shows multi-threading profiling carried out on the Dell workstation. To fit the problem onto a single computing node without reading too many separate forecast files, 10-member analog ensembles are generated for 10 forecast lead times and only 10 days using 1 month as search. The spatial domain is increased to the entire domain including 262,792 grids to ensure enough amount of workload to parallelize. Figure 4a shows the speedup of analog generation compared to the theoretical speedup defined by the Amdahl's Law as the following:

$$S(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

where

- S is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

The speedup ratio is well bounded between 95% and 98% which shows the majority of the execution has been parallelized. Speedup of the PAnEn further improves by about 20% when hyper-threading is used (the data point where 36 threads are used). This suggests that extra performance can be obtained on supercomputers with the same amount of computational budget because computation is usually charged by the number of nodes or cores, not threads or processes.

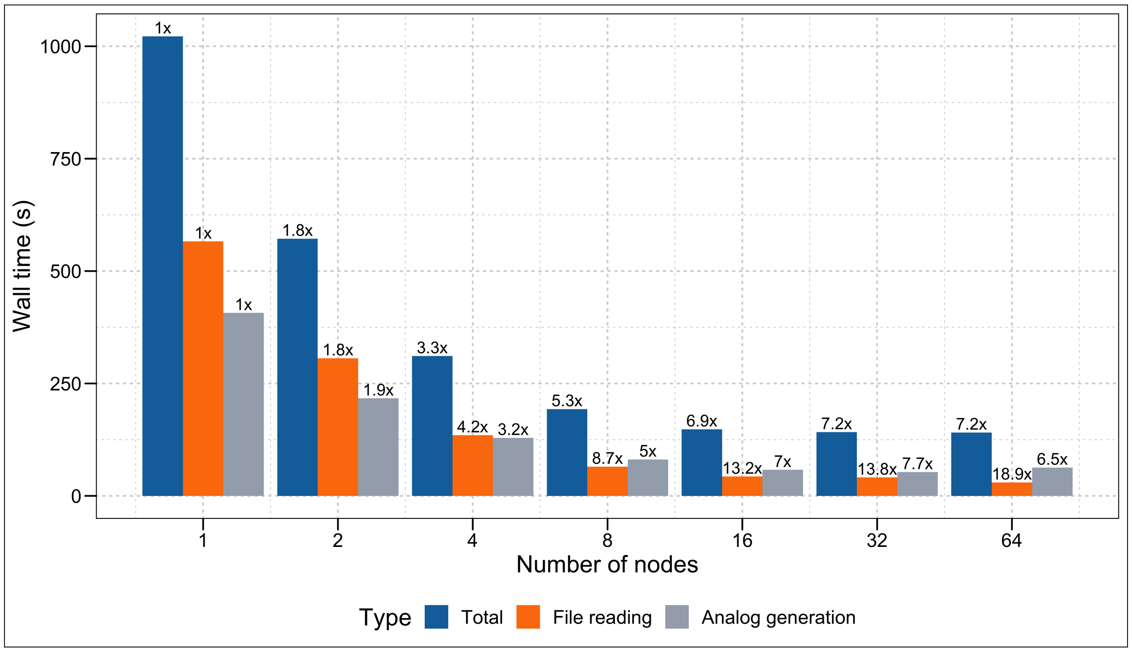


Fig. 5. Multi-node profiling of program runtime

Figure 4b shows the itemized time consumption with different numbers of threads. The stages are categorized into file I/O, preprocessing, and analog generation. File I/O is currently implemented serially because the dependent packages, ecCodes and netCDF, are currently not thread-safe and multi-threading implementation is discouraged. As a result, the time consumption remains constant across experiments when different numbers of threads are used. Preprocessing and analog generation are parallelized with multi-threading and the runtime decreases proportionally as the number of threads increases.

Figure 4b suggests when the execution runtime is below 200 seconds, more than half of the time is spent on serial file I/O. As mentioned in Section 3.1, this can be avoided by converting GRIB files to NetCDF files ahead of time. This solution, however, is not scalable and applicable for analog generation requiring a long search repository. Figure 5 shows the multi-node profiling of speedup for the two most computationally expensive stages, the reading stage and the analog generation stage. This experiment was carried out to generate 10-member analog ensembles for one year using two years as the search period. There are 18 forecast lead times and the total number of stations is 5,000. The experiment was completed on Cheyenne and each computing node has 36 cores. Each core is assigned with one MPI process. When the total runtime exceeds 250 seconds, the speedup of the reading stage and the analog generation stage is about 10% slower than the theoretical optimum. When the total runtime is reduced to under 250 seconds, the speedup reaches a plateau. This is because, in these cases, the workload distributed to each process becomes too small such that each process only needs to read several files and generate analogs for a handful of stations, but the communication between processes become significant. It is demonstrated in the case of analog generation going from 32 nodes to 64 nodes. When the number of nodes increases, the wall time of analog generation increases because, although the numbers of stations for each

```r
library(RAnEn)
config <- new(Config)
config$num_analogs <- 11
AnEn <- generateAnalogs(forecasts, observations, test_times, search_times, config, 'IS')
```

Listing 1. Code snippet for using a *RAnEn::Config* class object for analog generation

process decrease from $5000/(32 * 36) \approx 5$ to $5000/(64 * 36) \approx 2$, the number of total processes increase from $32 * 36 = 1152$ to $64 * 36 = 2304$.

The plateau cases shown in Figure 5 is a contrived example of how communication overhead overtakes the benefit from multi-node parallelization. An obvious solution to have better statistics is to increase the total workload. This shows that the program can achieve high parallelization efficiency when the total runtime is above 250 seconds. It is generally not suggested to request more nodes after this critical line is hit.

5 TYPICAL USAGE

This section provides typical usages of the PAnEn. The complete C++ interface documentation can be found at https://weiming-hu.github.io/AnalogsEnsemble/CXX; the complete R package documentation can be found at https://weiming-hu.github.io/AnalogsEnsemble/R.

5.1 RAnEn Package

The R package *RAnEn* provides the interface to the underlying C++ libraries together with preprocessing and file I/O functions. This package is usually used in conjunction with *RAnEnExtra* which provides extensive functions for forecast verification and analog visualization. *RAnEnExtra* can be access from https://weiming-hu.github.io/RAnEnExtra/.

A typical workflow of using *RAnEn* includes:

(1) Preparing forecasts and observations. Forecasts and observations are represented as R lists with members including data, times, and locations. A complete list of accepted members can be found at https://weiming-hu.github.io/AnalogsEnsemble/2019/01/16/NetCDF-File-Types.html;
(2) Specifying the test and search periods and the number of analogs to generate;
(3) Calling the main function, *RAnEn::generateAnalogs*, to generate analogs.

There are several ways to prepare forecasts and observations in the required format. If forecasts are originally stored in GRIB files, the command line utilities, *grib_convert* and *grib_convert_mpi*, can be used to convert GRIB files to NetCDF files with the required format. Then, these NetCDF files can be read into R using *RAnEn::readForecasts* and *RAnEn::readObservations*. These two functions are actually designed to read NetCDF files that already have the required variables and they are not designed to read any arbitrary NetCDF files. Observations are typically stored as multivariate time series format, for example, ground level ozone measurement from 2017 to 2019. This type of format can generally be read into R as data frames, and then converted to the required format using *RAnEn::formatObservations*. For cases that are not covered, users need to manually format forecasts and observations.

```
# Data folders
forecasts-folder = /path/to/forecast/data
analysis-folder = /path/to/observation/data

# Tagged regular expressions for parsing file names
forecast-regex = ^.*nam_218_(?P<day>\d{8})_(?P<cycle>00)\d{2}_(?P<flt>000)\.grb2$
analysis-regex = ^.*nam_218_(?P<day>\d{8})_(?P<cycle>00)\d{2}_(?P<flt>000)\.grb2$

# Test and search time periods in standard format
search-start = 2016-01-01 00:00:00
search-end = 2017-12-31 23:59:59
test-start = 2018-01-01 00:00:00
test-end = 2018-12-31 23:59:59

# Which parameters to read from GRIB files as forecast predictors
#
# Parameter: Downward short-wave radiation flux
pars-name = DownwardShortwaveRadiation
pars-id = 260087
levels = 0
level-types = surface

# Parameter: 2 metre temperature
pars-name = Temperature_2m
pars-id = 167
levels = 2
level-types = heightAboveGround
```

Listing 2. Example configuration file for *anen_grib* and *anen_grib_mpi*

To generate analogs, *RAnEn::generateAnalogs* accepts six arguments including forecasts, observations, test times, search times, the name of the technique to use, and a detailed configuration object. The supported names are “IS” for independent search and “SSE” for search space extension. A *RAnEn::Config* object provides additional arguments for the analog generation function. A typical use of *RAnEn::Config* is provided in Listing 1.

5.2 PAnEn on Supercomputers

PAnEn is designed and implemented to solve complex problems that require sizeable computation with state-of-the-art supercomputers. Although it is possible to run R scripts on supercomputers, it is suggested to use the command line utilities built on top of the C++ libraries on supercomputers for maximum performance. There are two groups of command line utilities, *grib_convert* and *anen_grib*. Each group has its MPI variant. *grib_convert* is designed for converting and subseting a large number of GRIB files to NetCDF files. *anen_grib* is designed to generate analogs from GRIB input. All utilities accept a regular text file as its configuration file, typically named with the extension ".cfg". A minimal example of the configuration file is provided in Listing 2.

Generally, places where extra attention is needed are predictor definition and regular expressions. At least three attributes are needed to locate a specific parameter from a GRIB file: the parameter ID, the vertical level, and the vertical level type. Parameter ID can be found from the ECMWF GRIB parameters database at https://apps.ecmwf.int/codes/grib/param-db. *grib_ls* can be used to query the vertical levels and level types from a particular GRIB file. An example command for using *grib_ls* to print out meta information from a GRIB file is *grib_ls -p shortName,name,paramId,level,typeOfLevel nam_218_20190101_0000_000.grb2*. This command prints out the specified fields of all variables available in the GRIB file.

Regular expressions [24, 34] are used to extract time information from a particular file name because model simulation usually outputs all data for a particular cycle time and a particular lead time to a single file. This file contains simulation results for the entire domain for one timestamp and usually with all variables from all vertical levels. Therefore, time information can be identified from the file name. Different models, however, have different naming conventions and users need to provide the proper regular expression to assist parsing file names. For example, this is a typical file name from WRF NAM moel simulations, *nam_218_20190101_1200_084.grb2*. Separated by underscores, components are, in turns, the model type, the grid type, the forecast date in standard format, the simulation cycle time in military time, and the forecast lead time in hours. PAnEn needs help to understand which parts corresponds to the forecast date, the cycle time, and the lead time. An example of a regular expression is as follows, *nam_218_(?P<day>\d{8})_(?P<cycle>00)\d{2}_(?P<flt>\d{3})ġrb2*. The three pairs of parenthesis indicate parts of the string to match. Within the first pair, *?P<day>* specifies that the matched portion should be referenced as the date which is an eight-digit string. Similarly, the second pair of parenthesis should be referenced as the forecast cycle time and this expression specifically matches the cycle time "00". The third pair of parenthesis matches the forecast lead time from a three-digit string. If the regular expression is not properly formatted, PAnEn will raise an error and ask for changes.

6 SUMMARY

The PAnEn is an open-source library developed to generate forecast ensembles from a single run of a deterministic weather model and corresponding historical observations. It provides highly efficient implementation and a flexible interface for generating the AnEn. Current interface supports C++ and R usage and file I/O with GRIB and NetCDF. Multi-threading profiling shows over 95% parallelization of the code and multi-node profiling shows low overhead when the execution runtime exceeds 250 seconds. A hybrid architecture of OpenMP and MPI ensures the scaling capability of the programs. This library will continue to be maintained and improved with the addition of new analog generation algorithms and better parallelization implementation for large scale problems.

REFERENCES

[1] S. Alessandrini, L. Delle Monache, S. Sperati, and G. Cervone. 2015. An analog ensemble for short-term probabilistic solar power forecast. *Applied Energy* 157 (2015), 95 – 110. https://doi.org/10.1016/j.apenergy.2015.08.011

[2] S. Alessandrini, L. Delle Monache, S. Sperati, and J.N. Nissen. 2015. A novel application of an analog ensemble for short-term wind power forecasting. *Renewable Energy* 76 (2015), 768 – 781. https://doi.org/10.1016/j.renene.2014.11.061

[3] Mohammad Bannayan and Gerrit Hoogenboom. 2008. Weather analogue: a tool for real-time prediction of daily weather data realizations based on a modified k-nearest neighbor approach. *Environmental Modelling & Software* 23, 6 (2008), 703–713.

[4] Guido Cervone, Laura Clemente-Harding, Stefano Alessandrini, and Luca Delle Monache. 2017. Short-term photovoltaic power forecasting using Artificial Neural Networks and an Analog Ensemble. *Renewable Energy* (2017). https://doi.org/10.1016/j.renene.2017.02.052

[5] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. 2001. *Parallel programming in OpenMP.* Morgan kaufmann.
[6] Barbara Chapman, Gabriele Jost, and Ruud Van Der Pas. 2008. *Using OpenMP: portable shared memory parallel programming.* Vol. 10. MIT press.
[7] Laura Clemente-Harding. 2019. *Extension of the Analog Ensemble Technique to the Spatial Domain.* Ph.D. Dissertation. Pennsylvania State University, University Park, Pennsylvania. https://catalog.libraries.psu.edu/catalog/28948273
[8] Laura Clemente-Harding, Weiming Hu, and Guido Cervone. 2020. An Introduction to the Analog Ensemble Technique. ERDC/GRL SR-19-870 (2020).
[9] Computational and Information Systems Laboratory. 2019. Cheyenne: HPE/SGI ICE XA System (Climate Simulation Laboratory). Boulder, CO: National Center for Atmospheric Research. https://doi.org/10.5065/D6RX99HX
[10] Leonardo Dagum and Ramesh Menon. 1998. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering* 5, 1 (1998), 46–55.
[11] Luca Delle Monache, F.Anthony Eckel, Daran L. Rife, Badrinath Nagarajan, and Keith Searight. 2013. Probabilistic Weather Prediction with an Analog Ensemble. *Monthly Weather Review* 141 (2013), 3498–3516. https://doi.org/10.1175/MWR-D-12-00281.1
[12] Dirk Eddelbuettel, Romain François, J Allaire, Kevin Ushey, Qiang Kou, N Russel, John Chambers, and D Bates. 2011. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software* 40, 8 (2011), 1–18.
[13] National Weather Service NOAA U.S. Department of Commerce Environmental Modeling Center, National Centers for Environmental Prediction. 2015. NCEP North American Mesoscale (NAM) 12 km Analysis. http://rda.ucar.edu/datasets/ds609.0/
[14] Alessandro Fanfarillo, Behrooz Roozitalab, Weiming Hu, and Guido Cervone. 2019. Probabilistic Forecasting using Deep Generative Models. *arXiv preprint arXiv:1909.11865* (2019).
[15] Tilmann Gneiting, Adrian E Raftery, Anton H Westveld III, and Tom Goldman. 2005. Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation. *Monthly Weather Review* 133, 5 (2005), 1098–1118.
[16] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. 1996. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel computing* 22, 6 (1996), 789–828.
[17] T. M. Hopson. 2013. Assessing the Ensemble Spread–Error Relationship. *Monthly Weather Review* 142, 3 (2013), 1125–1142. https://doi.org/10.1175/mwr-d-12-00111.1
[18] Weiming Hu and Guido Cervone. 2019. Dynamically Optimized Unstructured Grid (DOUG) for Analog Ensemble of numerical weather predictions using evolutionary algorithms. *Computers & Geosciences* 133 (2019), 104299.
[19] Weiming Hu, Guido Cervone, Laura Clemente-Harding, and Martina Calovi. 2019. *Parallel Analog Ensemble.* https://doi.org/10.5281/zenodo.3384321
[20] Huang Jianping, Yi Yuhong, Wang Shaowu, and Chou Jifen. 1993. An analogue-dynamical long-range numerical weather prediction system incorporating historical evolution. *Quarterly Journal of the Royal Meteorological Society* 119, 511 (1993), 547–565.
[21] Jianwei Li, Wei-keng Liao, Alok Choudhary, Robert Ross, Rajeev Thakur, William Gropp, Robert Latham, Andrew Siegel, Brad Gallagher, and Michael Zingale. 2003. Parallel netCDF: A high-performance scientific I/O interface. In *Supercomputing, 2003 Acm/Ieee Conference.* IEEE, 39–39.
[22] Edward N Lorenz. 1969. Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric sciences* 26, 4 (1969), 636–646.
[23] Blake Madden. 2006. Using cppunit to implement unit testing. *Game Programming Gems* 6 (2006).
[24] Eric Niebler. 2007. Boost. xpressive.
[25] Charles Obled, Guillaume Bontron, and Rémy Garçon. 2002. Quantitative precipitation forecasts: a statistical adaptation of model outputs through an analogues sorting approach. *Atmospheric research* 63, 3-4 (2002), 303–324.
[26] R Core Team. [n.d.]. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org
[27] Russ Rew and Glenn Davis. 1990. NetCDF: an interface for scientific data access. *IEEE computer graphics and applications* 10, 4 (1990), 76–82.
[28] Mark S Roulston and Leonard A Smith. 2003. Combining dynamical and statistical ensembles. *Tellus A: Dynamic Meteorology and Oceanography* 55, 1 (2003), 16–30.
[29] Boris Schäling. 2011. *The boost C++ libraries.* Boris Schäling.
[30] Michael Scheuerer. 2014. Probabilistic quantitative precipitation forecasting using ensemble model output statistics. *Quarterly Journal of the Royal Meteorological Society* 140, 680 (2014), 1086–1096.
[31] M Shahriari, Guido Cervone, L Clemente-Harding, and L Delle Monache. 2020. Using the analog ensemble method as a proxy measurement for wind power predictability. *Renewable Energy* 146 (2020), 789–801.
[32] Sameer S Shende and Allen D Malony. 2006. The TAU parallel performance system. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 287–311.
[33] Simone Sperati, Stefano Alessandrini, and Luca Delle Monache. 2017. Gridded probabilistic weather forecasts with an analog ensemble. *Quarterly Journal of the Royal Meteorological Society* 143, 708 (2017), 2874–2885. https://doi.org/10.1002/qj.3137
[34] Ken Thompson. 1968. Programming techniques: Regular expression search algorithm. *Commun. ACM* 11, 6 (1968), 419–422.
[35] Zoltan Toth. 1989. Long-range weather forecasting using an analog approach. *Journal of climate* 2, 6 (1989), 594–607.
[36] Zoltan Toth. 1991. Estimation of atmospheric predictability by circulation analogs. *Monthly weather review* 119, 1 (1991), 65–72.
[37] University Corporation for Atmospheric Research Unidata, NOAA U.S. Department of Commerce National Centers for Environmental Prediction, National Weather Service, and European Centre for Medium-Range Weather Forecasts. 2003. Historical Unidata Internet Data Distribution (IDD) Gridded Model Data.

http://rda.ucar.edu/datasets/ds335.0/
[38] HM Van den Dool. 1989. A new look at weather forecasting through analogues. *Monthly weather review* 117, 10 (1989), 2230–2247.
[39] HM Van den Dool. 1994. Searching for analogues, how long must we wait? *Tellus A* 46, 3 (1994), 314–324.
[40] Jeffrey S. Whitaker and Andrew F. Loughe. 2002. The Relationship between Ensemble Spread and Ensemble Mean Skill. *Monthly Weather Review* 126, 12 (dec 2002), 3292–3302. https://doi.org/10.1175/1520-0493(1998)126<3292:trbesa>2.0.co;2
[41] RA Yuen, T Gneiting, TL Thorarinsdottir, and C Fraley. 2013. ensembleMOS: Ensemble model output statistics. *R package version 0.7* (2013).

COLLABORATIVE EXPLORATION OF SCIENTIFIC DATASETS USING IMMERSIVE AND STATISTICAL VISUALIZATION

Nicholas Burnhart-Lupo[1], Brian Bush[1], Kenny Gruchalla[1], Kristin Potter[1], Steve Smith[2]

1 National Renewable Energy Laboratory
2 Los Alamos Visualization Associates

We discuss the value of collaborative, immersive visualization for the exploration of scientific datasets and review techniques and tools that have been developed and deployed at the National Renewable Energy Laboratory (NREL). We believe that collaborative visualizations linking statistical interfaces and graphics on laptops and high-performance computing (HPC) with 3D visualizations on immersive displays (head-mounted displays and large-scale immersive environments) enable scientific workflows that further rapid exploration of large, high-dimensional datasets by teams of analysts. We present a framework, PlottyVR, that blends statistical tools, general-purpose programming environments, and simulation with 3D visualizations. To contextualize this framework, we propose a categorization and loose taxonomy of collaborative visualization and analysis techniques. Finally, we describe how scientists and engineers have adopted this framework to investigate large, complex datasets.

Additional Key Words and Phrases: datasets, collaborative visualization, statistical graphics, scientific workflow

1 INTRODUCTION

Immersive visualization is poised to advance analysis for certain classes of complex scientific and engineering data, through rapid advances in virtual reality (VR) and augmented reality (AR). In our daily usage of the large-scale immersive virtual environment at the National Renewable Energy Laboratory (NREL), we have observed how immersive visualizations enhance scientific workflows [27]. Many scientists and engineers across NREL are beginning to adopt commodity AR/VR head-mounted displays (HMDs). While these immersive displays may provide unique qualitative insights, they are relatively limited for quantitative examinations. To deepen our analyses, we have been blending statistical interfaces and statistical graphics on traditional displays with 3D visualizations in immersive displays, providing a collaborative visualization framework with the objective of gaining the best of both worlds.

Collaboration has long been named one of the grand challenges for visual analytics [63]. There has been significant research into distributed and co-located visualization and sense-making [5, 30, 41]. However, this collaborative visualization research has focused on a *shared* visual representation, either synchronously or asynchronously, among analysts. We are specifically interested in supporting workflows where analysts collaborate through heterogeneous imagery, as shown in Figure 1. The views into the data may be different, tailored to each analyst, but the views are directly linked together. For example, we describe use cases where some analysts have immersive views of a dataset, while other analysts have more quantitative statistical views of that same dataset. When one analyst introduces derived data or flags regions of interest, those actions are visible and available in all the views.

2 BACKGROUND

Increases in computational power has lead to growth in the size and complexity of scientific datasets. There has been a corresponding growth in both need and opportunity for teams, diverse in terms of location and expertise, to extract usable knowledge from

Authors' addresses: Nicholas Brunhart-Lupo, nicholas.brunhart-lupo@nrel.gov; Brian Bush, brian.bush@nrel.gov; Kenny Gruchalla, kenny.gruchalla@nrel.gov; Kristin Potter, kristi.potter@nrel.gov, National Renewable Energy Laboratory, 15013 Denver West Pky, Golden, CO, 80401; Steve Smith, sas@lava3d.com, Los Alamos Visualization Associates, 3 Bundy Rd, Otowi, NM, 87506.

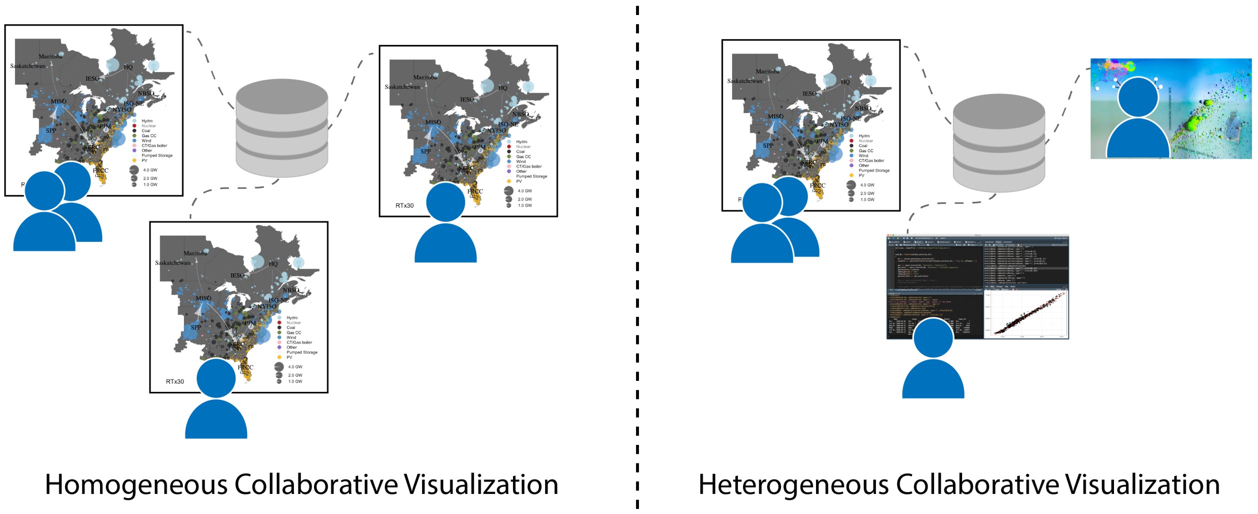

Homogeneous Collaborative Visualization
Heterogeneous Collaborative Visualization

Fig. 1. Left: In traditional collaborative visualization users share common imagery. Right: We propose linking different types of imagery of the same data, facilitating analysis by providing to stakeholders visualizations customized to their tasks and expertise, while directly linking and coordinating the data and actions of the team.

these sets. The technical progress in collaborative visualization has been likewise advancing; however, the challenges in creating compelling, useful, and accessible visualization environments hold ubiquitous collaborative visualization still at arm's length. These obstacles include technical challenges relating to factors like a limitation of network bandwidth [29], trade-offs between centralized or disparate data management [61], and the targeting of different types of display hardware. Several surveys have addressed the evolution of these challenges throughout the years [10, 19, 21, 26, 30, 57]; however, many of the software tools and environments have become outdated. Similarly, there has been research on human issues relating to how people operate in immersive spaces [69] and how they use shared visualization [68].

Experimentation and practice at NREL have revealed myriad considerations for designing workflows to explore datasets using collaborative, immersive visualization. A primary concern is whether collaborators are physically present in the same room and, if so, whether they jointly use the same immersive display or individually use separate displays. If they jointly use a display, then the collaborators may share the point of view with a single, privileged collaborator or have their own points of view for the shared display. (Multiple points of view for the same display pose technical challenges for walk-in immersive spaces, but not for individual AR or VR headsets.) Furthermore, with the joint use of a display, either a single collaborator or multiple collaborators might have control, being able to manipulate the view with user-input devices such as gamepads, joysticks, gloves, or wands. For display hardware that supports multiple points of view, each collaborator can be presented with a customized rendering of the model. For instance, some collaborators might view a simpler rendering of the dataset whereas others might see richer renderings that include uncertainty information or high-density details, but the basic geometric skeleton of the renderings likely would be common to all of the collaborators.

In situations where collaborators do not jointly use the same display, each collaborator might see and manipulate radically different renderings and subsets of the underlying raw dataset. For example, some renderings might be scatterplots, others might be maps, and still others might be abstract statistical graphics. Selections or highlighting of data records by one collaborator might automatically propagate to the views of the other collaborators; this linkage of data records across views might be total, targeted, or absent. Collaborators might work on subsets of the data, each having their own scene graph. The combination of disparate views with linked selections or highlights enables simultaneous, coordinated exploration among collaborators probing different aspects of the same dataset, essentially combining the insightfulness of the team and opening dialogs scrutinizing patterns, correlations, and hypotheses regarding the dataset.

Immersive collaboration can extend beyond the confines of a single room and a particular time. Moreover, recording, annotation, and bookmarking of visualization sessions allows for their revisiting and replay at later times, perhaps by different subsets of the collaborative team. These techniques further extend the exploratory dialog beyond single sessions and allow supplemental "offline", asynchronous exploration between the immersive sessions. Moreover, collaborators might be geographically distributed, so all of the aforementioned interactions can occur at distant locations.

Real-time statistical and data-analytic tools can supplement and drive immersive explorations. When collaborators find interesting paradigmatic features or form hypotheses regarding some aspect of a dataset, an analyst at a workstation or laptop might locate additional interesting features based on the paradigm or perform tests of the hypothesis, respectively, "pushing" the statistical results into the immersive display for scrutiny by the other collaborators. One might sometimes have several collaborators working in immersive spaces and several others working in statistical applications such as R, Python, or Julia, but with all of the collaborators' work linked in a common information space where results propagate among renderings. This approach combines the power of statistics and machine learning with the intuitive interactivity of immersive visualization.

2.1 Taxonomy

While a full taxonomy of work in collaborative visualization is out of scope for this publication, we present a loose classification of literature in the area to help establish the focus of previous works in this field and highlight directions for future work. We have broken our sampling of previous work into eight categories: location, imagery, viewpoint, data sources, number of display devices, type of display devices, interaction, and epoch. These categories are meant to summarize aspects of collaborative visualization and may not reflect details of those aspects.

The first category in our taxonomy is *location*, which refers to where participants are joining the collaboration. This category is loosely divided into remote and colocated participation. However, there has been a collection of research that focuses on mixed-presence [33], that is, support for both remote and in-person participation. Many of the challenges for remote and colocated participation are quite disparate: remote participation requires high-bandwidth, low-latency networking in models where data and imagery are shared from a centralized server, or for users to have high-powered computers to do that processing at each remote location. With the advances in new web-based technologies, some of these concerns are being alleviated; however, for large-scale scientific computing, these issues are still paramount. For colocated users, more prevalent issues include how to facilitate multiple user inputs and interactions, how to physically locate users in a space, and how to target imagery for different user tasks in a single location. There has also been work on representing remote participants as virtual avatars [59]. However, we did not capture that work in this taxonomy as the challenges to using avatars include technical issues related to motion capture of remote users and representational challenges of including avatars in a visualization environment, as well as issues relating to human interactions with avatars and other human factors. While this work is very relevant to collaborative visualization, the broad scope of such work is outside the focus of this paper.

The second category of our taxonomy is *imagery*: what the users are seeing while using a system. We classify the imagery as either *constant*, that is all users see the same representation of data, or *targeted* where imagery is designed for particular types of users or tasks. For example, the SAGE and SAGE2 frameworks [52, 61] aim at providing a workspace for multiple users to present windows from their individual machines to a powerwall. Thus, users can look at the same data but design disparate visualizations that investigate different aspects of that data and share those visualizations on a single powerwall. Similarly, systems could target volume rendering type displays for users whose main interest is the 3D aspects of the data, while also providing statistical charts and graphs for analysts looking at general 1 and 2D summaries of a data set.

Next, we look at *viewpoint*, which defines where in the scene a user is looking. The two classes in this category are *privileged* and *independent*. Privileged viewpoints constrain all users to see what the privileged user sees (WISIWYS—"what I see is what you see") with no support for multiple view points. Independent viewpoints allow users to individually explore the data, and oftentimes share their viewpoint with collaborators. Thus, one can imagine privileged viewing of a a volume rendering that is similar to a guided tour, versus allowing a user to actually navigate around the rendering. The challenges in disparate viewpoints include the computational expense of those different rendering viewpoints as well as how to indicate disparate viewpoints across all collaborators.

Interaction is related to viewpoint in that support for independent viewpoints often provides support for *asynchronous* interactions. However, asynchronous interactions may also refer to actions taken on a data set such as filtering, etc., and may be reflected back to all users in a privileged viewpoint environment, possibly through a queue of interactions or through sharing of a viewpoint. *Synchronous* interactions can be thought of as passing the baton such that only one user is performing an interaction and all other users are essentially an audience.

The *data sources* category considers whether the virtual environment provides methods for looking at different data sources simultaneously. Homogeneous environments support a single data source, while heterogeneous environments can handle disparate data sets. For example, a materials scientist may want to look at a volume rendering of a proposed molecule, whereas a techno-economist may want to look at cost sheets and graphs for the new material. The visualizations that support multiple data sets may actually use constant imagery across all users or can target visualizations for data set or user type. NREL is actively looking at how to support these sorts of heterogeneous data sets in collaborative spaces. The main challenges include how to visually connect disparate data sets and how to manage these data sets at scale.

The *number of displays* category simply defines how many display devices are targeted by a system. For example, is the collaboration environment a single cave-like immersive environment or powerwall, or is there a network of display devices, either remotely or co-located? Much of the early work on immersive displays focused on hardware and software issues in the design of single displays and not on the collaborative aspects of such spaces, and thus much of that work is out of scope of this taxonomy. However, the utility of collaboration quickly shifted the need to connect multiple users and thus multiple display devices. This category is not solely captured by the location category, specifically because multiple displays may be used in both remote and in-person collaborations.

The *type of devices* category is separated as its own category to capture targeting homogeneous or heterogeneous display types. For example, connecting two immersive cave-like displays is homogeneous, whereas connecting HMDs and laptops is a heterogeneous environment. This type of situation is becoming more prevalent as HMDs, table-top displays, and other state-of-the-art displays are becoming more main stream.

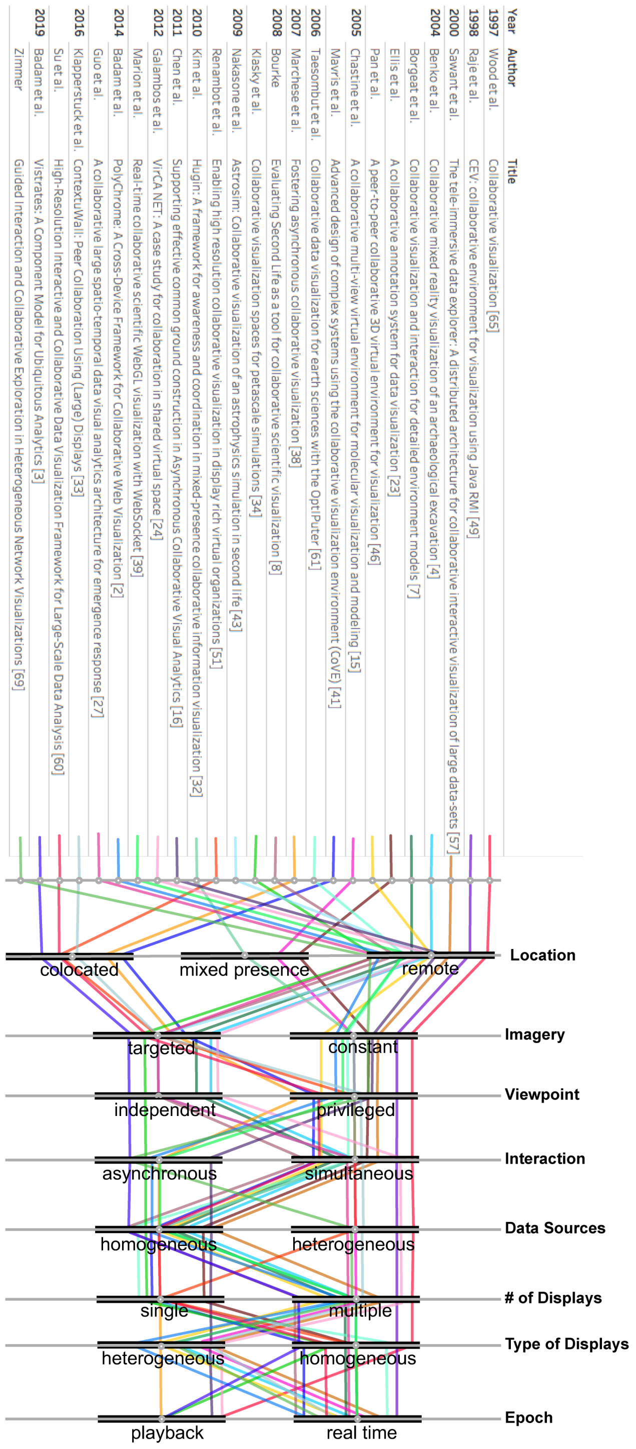
Finally, the last category, *epoch* relates to when a user is collaborating within a system. Most often, *realtime* collaboration is what is thought of in terms of immersive spaces, where people gather at the same time and supplement technological collaboration with personal interactions such as speech. However, there is also a group of work looking at *playback* collaborations where users can use the collaborative system independently, and share their work with others.

Year	Author	Title
1994	Anupam et al.	Distributed and collaborative visualization [1]
1995	Pang et al.	CSpray: A collaborative scientific visualization application [47]
	Wood et al.	CSCV-computer supported collaborative visualization [66]
1996	Rantzau et al.	Collaborative and interactive visualization in a distributed high performance software environment [50]
1997	Pang et al.	Collaborative 3D visualization with CSpray [48]
2000	Brewer et al.	Collaborative geographic visualization: Enabling shared understanding of environmental processes [9]
	Childers et al.	Access grid: Immersive group-to-group collaborative visualization [17]
2004	Koyamada et al.	VizGrid: collaborative visualization grid environment for natural interaction between remote researchers [35]
2005	Casera et al.	A collaborative extension of a visualization system [13]
2006	Song et al.	Paraview-based collaborative visualization for the grid [59]
2007	Ryu et al.	Collaborative object-oriented visualization environment [54]
2010	Dupont et al.	Collaborative scientific visualization: The COLLAVIZ framework [22]
	Filho et al.	An immersive and collaborative visualization system for digital manufacturing [21]
2015	Li et al.	High performance heterogeneous computing for collaborative visual analysis [36]

Table 1. Collaborative visualization taxonomy of systems with default configurations. Specifically, systems in this table support remote locations, constant imagery, privileged viewpoints, simultaneous interactions, homogeneous data sources and display types, multiple display devices, and a realtime epoch

The categorization of our taxonomy is fluid in that some categories are clearly disparate while others are closely related, but broken out based on findings in the literature. Some of this is due to the maturation of the technologies behind collaborative environments. Hardware challenges of the late 1990s, including networking and compute bandwidth, have, for the most part, been solved and replaced with challenges of new and novel display technologies, as well as expectations regarding speed of interactions and targeted support of certain devices, such as mobile. We expect this taxonomy to evolve and perhaps combine or divide categories along with the advances enabling this technology.

Tables 1 and 2 lay out a selection of related publications that target collaborative visualization systems. These works, for the most part, look at software systems to support collaboration, and papers discussing the use of these software for specific applications have not been included. Tables 1 and 2 are divided by the most common and novel categorizations. The most common configurations, shown in Table 1, consists of systems that support collaborations with remote locations, constant imagery, privileged viewpoints, simultaneous interactions, homogeneous data sources and display types, multiple display devices, and a realtime epoch. Many of these systems are older and reflect the state of the art for mid-1990s to mid-2000s systems. Table 2 shows a taxonomy for novel configurations that vary across all categories. Many of these systems reflect innovations in computational power and bandwidth, visualization hardware and software, and ways of using collaborative environments. Future work for this taxonomy includes researching applications in which collaboration systems have been used and understanding the success rate of those efforts. In addition, it will be useful to understand which systems support newer rendering clients (see Figure 2) such as virtual and augmented reality goggles, touch-panels, and web browsers.



Year	Author	Title
1997	Wood et al.	Collaborative visualization [65]
1998	Raje et al.	CEV: collaborative environment for visualization using Java RMI [49]
2000	Sawant et al.	The tele-immersive data explorer: A distributed architecture for collaborative interactive visualization of large data-sets [57]
2004	Benko et al.	Collaborative mixed reality visualization of an archaeological excavation [4]
	Borgeat et al.	Collaborative visualization and interaction for detailed environment models [7]
	Ellis et al.	A collaborative annotation system for data visualization [23]
	Pan et al.	A peer-to-peer collaborative 3D virtual environment for visualization [46]
2005	Chastine et al.	A collaborative multi-view virtual environment for molecular visualization and modeling [15]
	Mavris et al.	Advanced design of complex systems using the collaborative visualization environment (CoVE) [41]
2006	Taesombut et al.	Collaborative data visualization for earth sciences with the OptIPuter [61]
2007	Marchese et al.	Fostering asynchronous collaborative visualization [38]
2008	Bourke	Evaluating Second Life as a tool for collaborative scientific visualization [8]
	Klasky et al.	Collaborative visualization spaces for petascale simulations [34]
2009	Nakasone et al.	Astrosim: Collaborative visualization of an astrophysics simulation in second life [43]
	Renambot et al.	Enabling high resolution collaborative visualization in display rich virtual organizations [51]
2010	Kim et al.	Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization [32]
2011	Chen et al.	Supporting effective common ground construction in Asynchronous Collaborative Visual Analytics [16]
2012	Galambos et al.	VirCA NET: A case study for collaboration in shared virtual space [24]
	Marion et al.	Real-time collaborative scientific WebGL visualization with WebSocket [39]
2014	Badam et al.	PolyChrome: A Cross-Device Framework for Collaborative Web Visualization [2]
	Guo et al.	A collaborative large spatio-temporal data visual analytics architecture for emergence response [27]
2016	Klapperstuck et al.	ContextuWall: Peer Collaboration Using (Large) Displays [33]
	Su et al.	High-Resolution Interactive and Collaborative Data Visualization Framework for Large-Scale Data Analysis [60]
2019	Badam et al.	Vistrates: A Component Model for Ubiquitous Analytics [3]
	Zimmer	Guided Interaction and Collaborative Exploration in Heterogeneous Network Visualizations [69]

Table 2. Parallel Collaborative visualization taxonomy of novel system configurations.

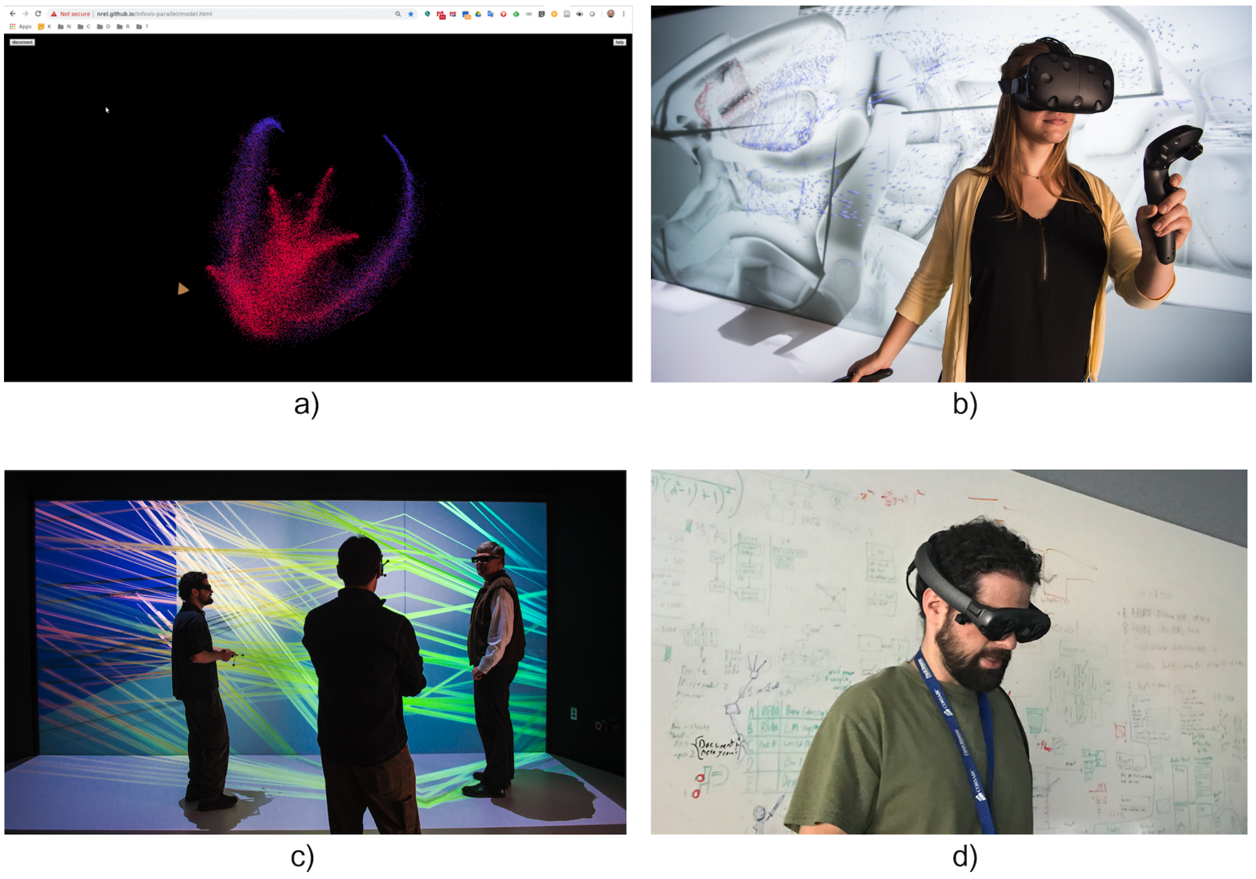


Fig. 2. Example 3D rendering clients: a) web browser with WebVR support; b) virtual reality (VR) headset; c) NREL's immersive visualization environment; d) augmented reality (AR) headset.

3 PLOTTYVR

PlottyVR represents a collection of tools to facilitate collaborative visualization. In terms of our taxonomy, this framework provides real-time, synchronous colocated or remote collaboration, with distinct imagery, distinct viewpoints, homogeneous data sources, on multiple heterogeneous displays. The primary objective is to combine the qualitative reasoning and intuition of immersive displays with the quantitative power of programming languages like R and Python. The toolset provides a bidirectional link between the Python or R programming language environments and an immersive 3D visualization, supporting both VR HMDs and large-scale, walk-in immersive environments. Using this software package, an analyst at a laptop can push data into an immersive visualization environment. The representation of that data can take the form of primitives, like points or line segments, or of more advanced objects like images and text. Developing immersive visualizations mirrors plot development in base R graphics [43] but with a third dimension. For example, R users can instantiate a 3D scatter plot in a connected immersive display by issuing a plot command with three arrays *x,y,z* for each data point, optionally providing point colors and diameters (scaling each point in cardinal directions to create ellipsoids). The immersive users can then interact with the visualization by selecting or querying data points. In our scatter plot example, textual annotations can also be linked to points so that the immersive users can, using this immersive system's interaction device, "click" on a point to quickly query relevant information, like a record identifier or notes about that data point. The immersive users can select regions of interest, which immediately become available to the R users for further quantitative analysis of the selected cluster.

One of our primary guiding design principle was to reduce the friction of incorporating an immersive space into statistical analyses. We have noticed that when the cost (even just the perception of cost) of using VR or other technologies is too high (whether that be financial, intellectual, or technical), researchers will not make use of it. This principle itself is expressed in several forms: (1) Critically, we cannot demand that analysts discard their existing workflows. Because of this, we chose to avoid a stand-alone, foundational framework. A common complaint for existing visualization solutions, such as ParaView or other large software packages, is that the researcher must then answer the question "how do I get my data and decision flow *into* there?" and re-engineer their entire pipeline. The user must now figure out the acceptable formats for the package, possibly write scripts to translate data, and then figure out how to extract features in the data and translate that back into discoveries. This is in addition to the challenges of learning to navigate an immersive space. In light of this, we chose to integrate with the existing analysis pipelines, i.e., to augment, not replace. (2) This augmentation must also be friction-free. Even if the package does not require replacing a researcher's workflow, it can still be excessively intrusive; researchers may quickly become disaffected when required to spend tens of minutes setting up a run-time environment on a server, verifying connections, and copying data for five minutes of exploration. Turnaround and setup times must be kept to a minimum.

To meet the aforementioned requirements, we provide PlottyVR to users as a library for R, Python, and Julia[1]. All that a researcher must do is to download and install these libraries for the environment of their choice, using the language's standard package manager[2], and issue a plot command. If, for example, the user is at a cave-like installation (such as the one at NREL), the server will already be running and the time from setup to use is on the order of seconds. If they are on their own private system, there is the additional step of launching the server first (automatic launching is currently being explored).

These libraries connect via WebSocket to the immersive platform. To reduce friction further, we also provide the most common plot types in a form that mirror the base graphics in R (see Fig. 3). 3D scatter plots and 3D line plots are direct and straightforward. From just the point (an ellipsoid in 3D) and the line (a tube in 3D) primitives, we can construct a wide variety of complex plot

[1]These are the most common environments in use at NREL. There is, however, no technical limitation to supporting other languages or systems.
[2]These packages are currently internal-only. They will be published in the future; see Section 5.

```
# Load and connect to PlottyVR
plotty_init()

# Plot the data
plot_tech("Coal Capacity [GW]", "Wind Capacity [GW]", "Utility Solar Capacity [GW]", ids=ids[1:70])

#---------------------------------------------------------------------------
#' plot_tech
#'
#' @description An immersive plot to compare real vs predicted of three
#'             technologies. Real trajectories are plotted as lines,
#'             and the error between real and pred is represented as
#'             an ellipsoid, scale by the error in the three cardinal
#'             directions.
#'
#' @param techx Metric string for x-axis
#' @param techy Metric string for y-axis
#' @param techz Metric string for z-axis
#' @param ids vector of run ids to plot
#'
tech_pts = function(techx, techy, techz, ids=unqiue(test$ID))
{
  xlim = range(c(data$Real[data$Metric==techx], data$Pred[data$Metric==techx]))
  ylim = range(c(data$Real[data$Metric==techy], data$Pred[data$Metric==techy]))
  zlim = range(c(data$Real[data$Metric==techz], data$Pred[data$Metric==techz]))

  # initialize the plot axes
  iplot(c(0),c(0),c(0),xlim=xlim,ylim =ylim,zlim=zlim,xlab=techx,ylab=techy,zlab=techz)

  plot_traj = function(ID)
  {
    col =randomcoloR::randomColor()
    xr = data$Real[data$ID==ID & data$Metric==techx]
    yr = data$Real[data$ID==ID & data$Metric==techy]
    zr = data$Real[data$ID==ID & data$Metric==techz]
    xp = abs(xr-data$Pred[data$ID==ID & data$Metric==techx])/(xlim[2]-xlim[1])
    yp = abs(yr-data$Pred[data$ID==ID & data$Metric==techy])/(ylim[2]-ylim[1])
    zp = abs(zr-data$Pred[data$ID==ID & data$Metric==techz])/(zlim[2]-zlim[1])

    id = ipoints(xr,yr,zr,col=col) # plot the points
    iscale(xp/2,yp/2,zp/2, id) # scale the points into ellipsoids
    ilines(xr,yr,zr,w_size=0.002,col=col) # draw the trajectories
  }

  invisible(lapply(ids, plot_traj))
}
```

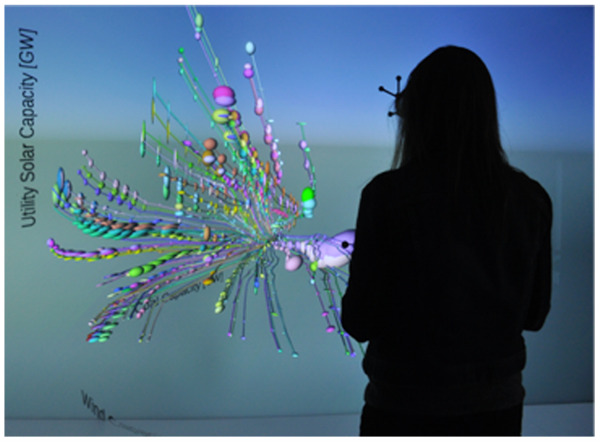


Fig. 3. R-code listing that generates the inset immersive visualization.

types, such as plot trajectories and 3D parallel coordinate plots, such as parallel planes [12]. We support images and text, providing context like geospatial plots and custom annotations. These API function calls are transmitted to a server application, running an interface to the graphics engine that is on the immersive platform. For the immersive environment at NREL, this engine is *Isopach*, a custom immersive scene graph library. For HMDs, we have developed a server application in the Unity engine. The server library is tasked with transforming the WebSocket messages to the engine's graphics representation, and relaying selection and other manipulation back to the client-side.

By integrating into environments in this way, not only are researchers given easy access to immersive plotting, but they are also able to make use of the space as a primitive for larger compositions. An example of this is that researchers can link together the immersive space and Shiny R [54] webpages, either for using a tablet as a companion in large immersive environments to

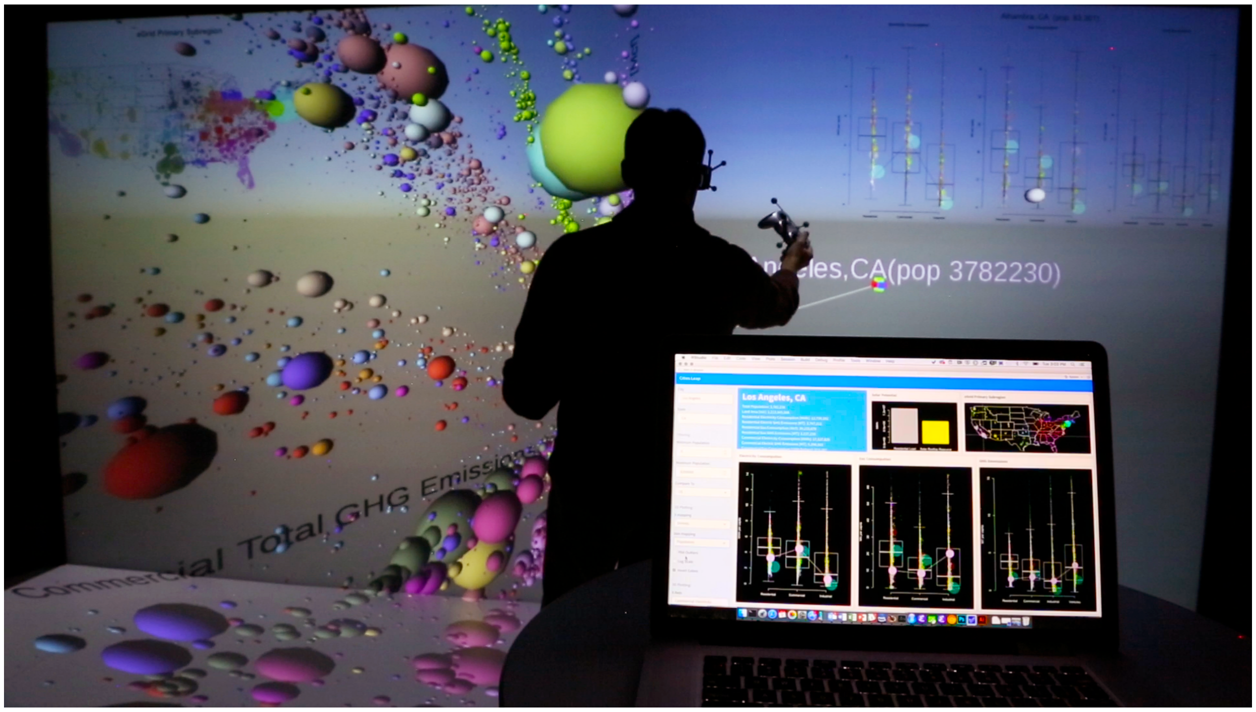


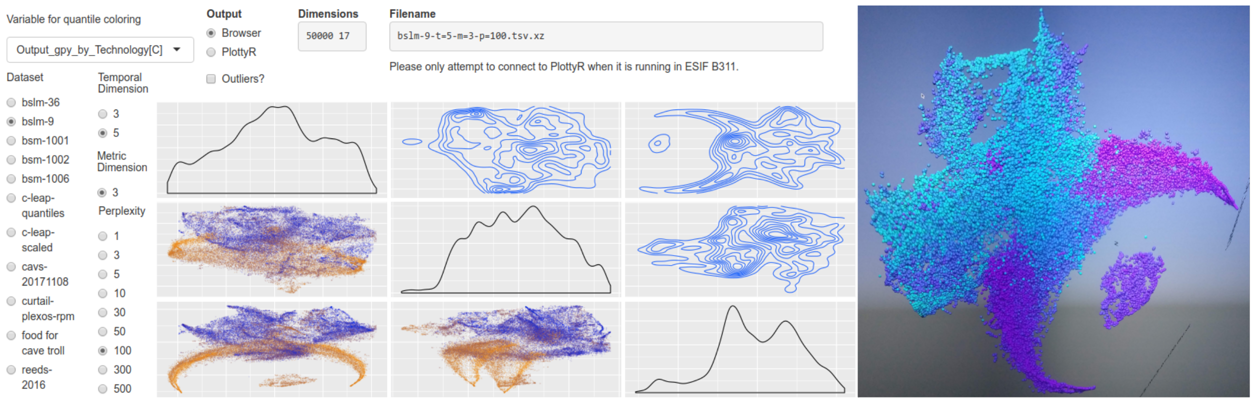
Fig. 4. Cities-LEAP analysis exploring a high-dimensional data set of city energy profiles. Foreground: real-time analysis in R-based Shiny web application. Background: immersive visualization.

furnish supplemental 2D plots and provide more accessible control over what is being seen, or for providing views for other remote analysts. In a more operational context, streaming data can be pulled in and joined with the immersive space for real-time analysis.

4 USE CASES

Using this toolset, we have achieved workflows such as synchronous, collaborative hypothesis testing where a statistician pushes data into the VR space, a scientist constructs a hypothesis by manipulating plots in that space, and the statistician applies statistical tests to the hypothesis, all in real-time. This workflow combines the power of statistical analysis with the insightfulness of rich, multidimensional data visualizations, and we have used this workflow to support both static datasets and on-demand simulations. PlottyVR has been used at NREL to meaningfully explore multidimensional time-series in as many as twenty dimensions by teams of as many as six persons, allowing collaborative identification of features, anomalies, and patterns in datasets that would be difficult and tedious to explore in 2D displays that limit collaborative interaction. Workflows combining real-time statistical analysis in R-based Shiny web applications [54] with immersive visualization by PlottyVR have been the most popular, allowing rapid, interactive statistical exploration of the output of complex dynamic simulation models. We briefly describe three examples, but PlottyVR continues to be employed regularly for varied applications at NREL.

The Cities Leading through Energy Analysis (Cities-LEAP) [64] project has developed city energy profiles for over 23,400 U.S. Cities. We combined an R-based Shiny dashboard that allowed cities to be compared on a range of metrics with immersive


Variable for quantile coloring: Output_gpy_by_Technology[C]

Output: [✓ Browser] PlottyR

Dimensions: 50000 17

Filename: bslm-9-t=5-m=3-p=100.tsv.xz

Please only attempt to connect to PlottyR when it is running in ESIF B311.

Outliers?

Dataset: bslm-36 [✓ bslm-9] bsm-1001 bsm-1002 bsm-1006 c-leap-quantiles c-leap-scaled cavs-20171108 curtail-plexos-rpm food for cave troll reeds-2016

Temporal Dimension: 3 [✓ 5]

Metric Dimension: [✓ 3]

Perplexity: 1 3 5 10 30 50 [✓ 100] 300 500

Fig. 5. R-based Shiny web application (left) and immersive visualization environment (right) for controlling and exploring self-organized maps of ensembles of simulation output [13].

3D scatter plots that revealed correlations, trends, stratification, and outliers in the data (see Figure 4). An analyst seated at the desktop would choose the metrics of interest, specifying the three axes, color, and size for the scatter plot. A second analyst in the immersive environment would identify and select cities or clusters of interest. Those selections would automatically update on the R desktop. The two analysts would iterate, hypothesizing on relationships, generating new metrics, and applying those metrics to the scatter plot to test the hypotheses.

In another application, the R program running on an analyst's laptop pushes ensembles of simulation output to the immersive environment displayed as 3D scatter plots. Working in the immersive space, analysts use a handheld tool to select regions of interest on the scatterplot and send that selection back to R on the laptop. The R program then performs Monte-Carlo filtering [56], which is a sensitivity analysis technique that infers which input variables most significantly influence whether data records fall inside versus outside of the selected region. The web application then displays (sometimes via a standard 2D projector on a wall next to the immersive environment) a ranked list of input parameters sorted in order of their influence on the output in the selected region. This enables rapid, interactive sensitivity analysis on ensembles of simulations: users can formulate, test, and discard hypotheses sequentially, gradually refining their understanding of the correlations and influences of input parameters on output results over ensembles of simulations. Interspersed with the hypothesis testing, analysts use the system to verify simulation behavior by scrutinizing input-output relationships and trends that might be present in the model.

We have deployed similar applications that combine Shiny web applications with immersive visualizations of self-organized maps (SOMs) of ensembles of high dimensional simulation output [13]. The web interface allows analysts to manipulate the parameters of the dimension-reduction algorithm and to view 2D projections of the SOMs on their laptops, but simultaneously to push the SOMs into 3D renderings in the immersive environment (see Figure 5). The user with the laptop can be in a location distant from the immersive environment. For this application, we have developed a parallel coordinates plotting WebVR client, specifically supporting multiple, simultaneous remote collaborators to view and jointly manipulate the same 3D scene of SOMs of simulation output.

5 FUTURE WORK

The most immediate improvement for the PlottyVR system is the publishing of server and client libraries to public-facing package repositories.

While PlottyVR has regularly proven useful, there are some limitations to the current implementation and protocol. First, the protocol originally was designed for a single client and a single server; while useful for a single researcher or a small team working in the same room, this has proven limiting in common situations at NREL, such as distributed analysis where many participants are not in the same locale (i.e. remote locations in the nomenclature of the taxonomy), and so is sub-optimal for multi-client or multi-server collaborative configurations. In order to share data outside of the single client and server model, the researcher must build their own methods of distributing data, updating that data, or coordinating clients or servers. Further, it was initially envisioned that only the server would be graphically based, and that there would be a library client.

In response to these limitations, NOODLES (NREL Object Oriented Data Layout and Exploration System) is a work-in-progress replacement for the protocol and data representation presently used in PlottyVR, updating the model to a single server with multiple clients. It functions closer to the scene-graph level, where a synchronized 3D scene can be shared across many clients with support for customizations for different form factors, as well as providing database-like access to the protocol to support clients[3] of any form factor (a more heterogeneous environment). Note that this would not change the interface for the user; they would still be supplied with a library to provide a low-friction path to simple plotting. Other clients can now join the session and register simple callbacks or software hooks to automatically watch for data changes from collaborators or add their own data and plots to the mix. Additional generalized primitives permit more applications to be developed beyond the domain of simple 3D plotting and statistics, all while still fulfilling the PlottyVR goals. An initial demonstration of the system can be found in [11].

ACKNOWLEDGMENTS

This work was authored in part by the National Renewable Energy Laboratory (NREL), operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

A portion of This research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

REFERENCES

[1] V. Anupam, C. Bajaj, D. Schikore, and M. Schikore. 1994-07. Distributed and collaborative visualization. *Computer* 27, 7 (1994-07), 37–43.

[2] Sriram Karthik Badam and Niklas Elmqvist. 2014-11-16. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces* (Dresden, Germany) *(ITS '14)*. Association for Computing Machinery, 109–118.

[3] Sriram Karthik Badam, Andreas Mathisen, Roman Rädle, Clemens N. Klokmose, and Niklas Elmqvist. 2019. Vistrates: A Component Model for Ubiquitous Analytics. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 586–596.

[4] Hrvoje Benko, Edward W. Ishak, and Steven Feiner. 2004. Collaborative mixed reality visualization of an archaeological excavation. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 132–140.

[3]It is up to the developer of the client library to determine how best to present this data for that platform. They can select 3D geometry, tabular representations, plots, or any mixture of these.

[5] Mark Billinghurst, Maxime Cordeil, Anastasia Bezerianos, and Todd Margolis. 2018. Collaborative Immersive Analytics. In *Immersive Analytics*, Kim Marriott, Falk Schreiber, Tim Dwyer, Karsten Klein, Nathalie Henry Riche, Takayuki Itoh, Wolfgang Stuerzlinger, and Bruce H. Thomas (Eds.). Springer International Publishing, Cham, 221–257.

[6] Mark Billinghurst, Maxime Cordeil, Anastasia Bezerianos, and Todd Margolis. 2018. Collaborative Immersive Analytics. In *Immersive Analytics*. Springer International Publishing, 221–257.

[7] Louis Borgeat, Guy Godin, Jean-François Lapointe, and Philippe Massicotte. 2004. Collaborative visualization and interaction for detailed environment models. In *10th International Conference on Virtual Systems and Multimedia*.

[8] Paul Bourke. 2008. Evaluating Second Life as a tool for collaborative scientific visualization. *Computer Games and Allied Technology* 29 (2008), 2008.

[9] Isaac Brewer, Alan M. MacEachren, Hadi Abdo, Jack Gundrum, and George Otto. 2000. Collaborative geographic visualization: Enabling shared understanding of environmental processes. In *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*. IEEE, 137–141.

[10] K. W. Brodlie, D. A. Duce, J. R. Gallop, J. P. R. B. Walton, and J. D. Wood. 2004. Distributed and Collaborative Visualization. 23, 2 (2004), 223–251.

[11] Nicholas Brunhart-Lupo. 2020. NOODLES Demo. https://www.youtube.com/watch?v=qddOHC_WHr0. Accessed: 6-24-2020.

[12] N. Brunhart-Lupo, B. W. Bush, K. Gruchalla, and S. Smith. 2016. Simulation exploration through immersive parallel planes. In *2016 Workshop on Immersive Analytics (IA)*. 19–24.

[13] Bruce Bugbee, Brian W. Bush, Kenny Gruchalla, Kristin Potter, Nicholas Brunhart-Lupo, and Venkat Krishnan. 2019. Enabling immersive engagement in energy system models with deep learning. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 12, 4 (2019), 325–337.

[14] Steve Casera, H.-H. Nageli, and Peter Kropf. 2005. A collaborative extension of a visualization system. In *First International Conference on Distributed Frameworks for Multimedia Applications*. IEEE, 176–182.

[15] Tom Chandler, Maxime Cordeil, Tobias Czauderna, Tim Dwyer, Jaroslaw Glowacki, Cagatay Goncu, Matthias Klapperstueck, Karsten Klein, Kim Marriott, Falk Schreiber, and Elliot Wilson. 2015. Immersive Analytics. In *2015 Big Data Visual Analytics (BDVA)*. 1–8.

[16] J.W. Chastine, Ying Zhu, J.C. Brooks, G.S. Owen, R.W. Harrison, and I.T. Weber. 2005-07. A collaborative multi-view virtual environment for molecular visualization and modeling. In *Coordinated and Multiple Views in Exploratory Visualization (CMV'05)*. 77–84.

[17] Yang Chen, Jamal Alsakran, Scott Barlowe, Jing Yang, and Ye Zhao. 2011. Supporting effective common ground construction in Asynchronous Collaborative Visual Analytics. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 101–110.

[18] Lisa Childers, Terry Disz, Robert Olson, Michael E. Papka, Rick Stevens, and Tushar Udeshi. 2000. *Access grid: Immersive group-to-group collaborative visualization*. Technical Report. Argonne National Lab., IL (US).

[19] Kristian Sons Christophe Mouton and Ian Grimstead. [n.d.]. Collaborative visualization: current systems and future trends. In *Proceedings of the 16th International Conference on 3D Web Technology* (2011). 101–110.

[20] Maxime Cordeil, Tim Dwyer, Karsten Klein, Bireswar Laha, Kim Marriott, and Bruce H. Thomas. 2017. Immersive Collaborative Analysis of Network Connectivity: CAVE-style or Head-Mounted Display? *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 441–450.

[21] Ciro Donalek, S. George Djorgovski, Alex Cioc, Anwell Wang, Jerry Zhang, Elizabeth Lawler, Stacy Yeh, Ashish Mahabal, Matthew Graham, and Andrew Drake. 2014. Immersive and collaborative data visualization using virtual reality platforms. In *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 609–614.

[22] Nelson Duarte Filho, Silvia Costa Botelho, Jonata Tyska Carvalho, Pedro de Botelho Marcos, Renan de Queiroz Maffei, Rodrigo Remor Oliveira, Rodrigo Ruas Oliveira, and Vinicius Alves Hax. 2010. An immersive and collaborative visualization system for digital manufacturing. *The International Journal of Advanced Manufacturing Technology* 50, 9 (2010), 1253–1261.

[23] Florent Dupont, Thierry Duval, Cedric Fleury, Julien Forest, Valérie Gouranton, Pierre Lando, Thibaut Laurent, Guillaume Lavoue, and Alban Schmutz. 2010. Collaborative scientific visualization: The COLLAVIZ framework.

[24] Sean E. Ellis and Dennis P. Groth. 2004. A collaborative annotation system for data visualization. In *Proceedings of the working conference on Advanced visual interfaces*. 411–414.

[25] Peter Galambos, Christian Weidig, Peter Baranyi, Jan C. Aurich, Bernd Hamann, and Oliver Kreylos. 2012. VirCA NET: A case study for collaboration in shared virtual space. In *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*. 273–277.

[26] I.J. Grimstead, D.W. Walker, and N.J. Avis. 2005. Collaborative visualization: a review and taxonomy. In *Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*. 61–69.

[27] Kenny Gruchalla and Nicholas Brunhart-Lupo. 2019. The Utility of Virtual Reality for Science and Engineering. In *VR Developer Gems*, William R. Sherman (Ed.). Taylor Francis, Chapter 21, 383–402.

[28] D. Guo, J. Li, H. Cao, and Y. Zhou. 2014. A collaborative large spatio-temporal data visual analytics architecture for emergence response. *IOP Conference Series: Earth and Environmental Science* 18 (2014), 012129.

[29] Andrei Hutanu, Gabrielle Allen, Stephen D. Beck, Petr Holub, Hartmut Kaiser, Archit Kulshrestha, Milos Liska, Jon MacLaren, Ludek Matyska, and Ravi Paruchuri. 2006. Distributed and collaborative visualization of large data sets using high-speed networks. *Future Generation Computer Systems* 22, 8 (2006), 1004–1010.

[30] Petra Isenberg, Niklas Elmqvist, Jean Scholtz, Daniel Cernea, Kwan-Liu Ma, and Hans Hagen. 2011. Collaborative visualization: Definition, challenges, and research agenda. *Information Visualization* 10, 4 (2011), 310–326.

[31] Timothy Jacobs and Sean Butler. 2001. Collaborative visualization for military planning. In *Java/Jini Technologies*, Vol. 4521. International Society for Optics and Photonics, 42–51.

[32] Nils Jensen, Stefan Seipel, Wolfgang Nejdl, and Stephan Olbrich. 2003. COVASE: Collaborative visualization for constructivist learning. In *Designing for Change in Networked Learning Environments*. Springer, 249–253.

[33] KyungTae Kim, Waqas Javed, Cary Williams, Niklas Elmqvist, and Pourang Irani. 2010. Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization. In *ACM International Conference on Interactive Tabletops and Surfaces*. 231–240.

[34] Matthias Klapperstuck, Tobias Czauderna, Cagatay Goncu, Jaroslaw Glowacki, Tim Dwyer, Falk Schreiber, and Kim Marriott. 2016. ContextuWall: Peer Collaboration Using (Large) Displays. In *2016 Big Data Visual Analytics (BDVA)*. 1–8.

[35] Scott Klasky, Roselyne Barreto, Ayla Kahn, Manish Parashar, Norbert Podhorszki, Steve Parker, Deborah Silver, and Mladen A. Vouk. 2008. Collaborative visualization spaces for petascale simulations. In *2008 International Symposium on Collaborative Technologies and Systems*. IEEE, 203–211.

[36] V. Ryuichi Matsukura V. Koji Koyamada, V. Yasuo Tan, and V. Yukihiro Karube V. Mitsuhiro Moriya. 2004. VizGrid: collaborative visualization grid environment for natural interaction between remote researchers. *FUJITSU Sci. Tech. J* 40, 2 (2004), 205–216.

[37] Jianping Li, Jia-Kai Chou, and Kwan-Liu Ma. 2015. High performance heterogeneous computing for collaborative visual analysis. In *SIGGRAPH Asia 2015 Visualization in High Performance Computing (SA '15)*. Association for Computing Machinery, 1–4.

[38] Thomas Ludwig, Tino Hilbert, and Volkmar Pipek. 2015. Collaborative visualization for supporting the analysis of mobile device data. In *ECSCW 2015: Proceedings of the 14th European Conference on Computer Supported Cooperative Work, 19-23 September 2015, Oslo, Norway*. Springer, 305–316.

[39] Francis T. Marchese and Natasha Brajkovska. 2007. Fostering asynchronous collaborative visualization. In *2007 11th International Conference Information Visualization (IV'07)*. IEEE, 185–190.

[40] Charles Marion and Julien Jomier. 2012-08-04. Real-time collaborative scientific WebGL visualization with WebSocket. In *Proceedings of the 17th International Conference on 3D Web Technology (Web3D '12)*. Association for Computing Machinery, 47–50.

[41] Roberto Martinez-Maldonado, Judy Kay, Simon Buckingham Shum, and Kalina Yacef. 2019. Collocated Collaboration Analytics: Principles and Dilemmas for Mining Multimodal Interaction Data. *Human-Computer Interaction* 34, 1 (Jan. 2019), 1–50.

[42] Dimitri Mavris, Patrick Biltgen, and Neil Weston. 2005. Advanced design of complex systems using the collaborative visualization environment (CoVE). In *43rd AIAA Aerospace Sciences Meeting and Exhibit*. 126.

[43] Paul Murrell. 2011. *R Graphics* (2nd ed.). CRC Press, Inc., USA.

[44] Arturo Nakasone, Helmut Prendinger, Simon Holland, Piet Hut, Jun Makino, and Ken Miura. 2009. Astrosim: Collaborative visualization of an astrophysics simulation in second life. *IEEE Computer Graphics and Applications* 29, 5 (2009), 69–81.

[45] Dorit Nevo, Saggi Nevo, Nanda Kumar, Jonas Braasch, and Kusum Mathews. 2015. Enhancing the Visualization of Big Data to Support Collaborative Decision-Making. In *2015 48th Hawaii International Conference on System Sciences*. 121–130. ISSN: 1530-1605.

[46] Jasminko Novak and Michael Wurst. 2005. Collaborative knowledge visualization for cross-community learning. In *Knowledge and Information Visualization*. Springer, 95–116.

[47] Yi Pan and Francis T. Marchese. 2004. A peer-to-peer collaborative 3D virtual environment for visualization. In *Visualization and Data Analysis 2004*, Vol. 5295. International Society for Optics and Photonics, 180–188.

[48] Alex Pang and Craig Wittenbrink. 1997. Collaborative 3D visualization with CSpray. *IEEE Computer Graphics and Applications* 2 (1997), 32–41. Publisher: IEEE.

[49] Alex Pang, Craig M. Wittenbrink, and Tom Goodman. 1995. CSpray: A collaborative scientific visualization application. In *Multimedia Computing and Networking 1995*, Vol. 2417. International Society for Optics and Photonics, 317–326.

[50] Rajeev R. Raje, Michael Boyles, and Shiaofen Fang. 1998. CEV: collaborative environment for visualization using Java RMI. *Concurrency: Practice and Experience* 10, 11 (1998), 1079–1085.

[51] D. Rantzau, U. Lang, R. Lang, H. Nebel, A. Wierse, and R. Ruehle. 1996. Collaborative and interactive visualization in a distributed high performance software environment. In *High Performance Computing for Computer Graphics and Visualisation*. Springer, 207–216.

[52] Luc Renambot, Byungil Jeong, Hyejung Hur, Andrew Johnson, and Jason Leigh. 2009. Enabling high resolution collaborative visualization in display rich virtual organizations. *Future Generation Computer Systems* 25, 2 (2009), 161–168.

[53] Nathalie Henry Riche, Kori Inkpen, John Stasko, Tom Gross, and Mary Czerwinski. 2012. Supporting asynchronous collaboration in visual analytics systems. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*. Association for Computing Machinery, 809–811.

[54] RStudio, Inc. 2013. *Easy web applications in R.* URL: http://www.rstudio.com/shiny/.

[55] So-Hyun Ryu, Hyung-Jun Kim, Jin-Sung Park, Yong-won Kwon, and Chang-Sung Jeong. 2007. Collaborative object-oriented visualization environment. *Multimedia Tools and Applications* 32, 2 (2007), 209–234.

[56] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. 2008. *Global sensitivity analysis: the primer.* John Wiley & Sons.

[57] Ali Sarvghad, Narges Mahyar, and Melanie Tory. 2009. History tools for collaborative visualization. *Collaborative Visualization on Interactive Surfaces-CoVIS'09* (2009), 21.

[58] Nikita Sawant, Chris Scharver, Jason Leigh, Andrew Johnson, Georg Reinhart, Emory Creel, Suma Batchu, Stuart Bailey, and Robert Grossman. 2000. The tele-immersive data explorer: A distributed architecture for collaborative interactive visualization of large data-sets. In *Proceedings of the Fourth International Immersive Projection Technology Workshop*. 1–16.

[59] Ralph Schroeder and Ann-Sofie Axelsson. 2006. *Avatars at work and play: Collaboration and interaction in shared virtual environments.* Vol. 34. Springer Science & Business Media.

[60] Guanghua Song, Yao Zheng, and Hao Shen. 2006. Paraview-based collaborative visualization for the grid. In *Asia-Pacific Web Conference.* Springer, 819–826.

[61] Simon Su, Vincent Perry, Nicholas Cantner, Dylan Kobayashi, and Jason Leigh. 2016-10. High-Resolution Interactive and Collaborative Data Visualization Framework for Large-Scale Data Analysis. In *2016 International Conference on Collaboration Technologies and Systems (CTS).* 275–280.

[62] Nut Taesombut, Xinran Ryan Wu, Andrew A. Chien, Atul Nayak, Bridget Smith, Debi Kilb, Thomas Im, Dane Samilo, Graham Kent, and John Orcutt. 2006. Collaborative data visualization for earth sciences with the OptIPuter. *Future Generation Computer Systems* 22, 8 (2006), 955–963.

[63] James J. Thomas and Kristin A. Cook (Eds.). 2005. *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* National Visualization and Analytics Ctr, Los Alamitos, Calif.

[64] U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy 2018. *Cities-LEAP City Energy Profiles.* U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy. https://apps1.eere.energy.gov/sled.

[65] Andreas Wierse. 1995. Collaborative visualization based on distributed data objects. In *Workshop on Database Issues for Data Visualization.* Springer, 208–219.

[66] Jason Wood, Helen Wright, and Ken Brodie. 1997. Collaborative visualization. In *Proceedings. Visualization'97).* IEEE, 253–259.

[67] Jason Wood, Helen Wright, and Ken Brodlie. 1995. CSCV-computer supported collaborative visualization. In *Proceedings of BCS Displays Group International Conference on Visualization and Modelling.* Citeseer, 13–25.

[68] Noráin Mohd Yusoff and Siti Salwah Salim. 2015. A systematic review of shared visualisation to achieve common ground. *Journal of Visual Languages & Computing* 28 (2015).

[69] Shanyang Zhao. 2003. Toward a Taxonomy of Copresence. *Presence: Teleoperators & Virtual Environments* 12 (2003), 445–455.

[70] Björn Zimmer. 2019. *Guided Interaction and Collaborative Exploration in Heterogeneous Network Visualizations.* Ph.D. Dissertation. Linnaeus University.

SOFTWARE ENGINEERING METHODS USED FOR THE PARALLELIO (PIO) C/FORTRAN LIBRARIES

Edward Hartnett [1,2], Jim Edwards[3]

1 CIRES, University of Colorado, Boulder, CO 80309, USA
2 NOAA/ESRL/GSD, Boulder, CO 80305, USA
3 NCAR, Boulder, CO 80305, USA

ABSTRACT

The PIO C and Fortran libraries allow for high-performance I/O on HPC systems. These libraries are developed using software engineering techniques such as branch development, pull-requests, automated testing, continuous integration, portable releases which adapt to user installation conditions, and full documentation of code for users and developers. This paper details the use of software engineering techniques on the PIO software project.

1. INTRODUCTION

1.1 Background

The Parallel IO libraries (PIO) are high-level parallel I/O C/Fortran libraries for structured grid applications. PIO provides a netCDF-like API, and allows users to designate some subset of processors to perform IO. Computational code calls netCDF-like functions to read and write data, and PIO uses the IO processors to perform all necessary IO.

The ParallelIO C library contains 28 KLOC (all KLOC including comment). The Parallel IO Fortran library contains 14 KLOC. There are 21 KLOC of C tests and 29 KLOC of Fortran tests (code counts for Version 2.4.2 release of PIO.)

1.2 Motivation

The PIO libraries include a significant amount of Fortran and C code, written for multi-processing systems using the Message Passing Interface (MPI) libraries.

Managing and developing complex software, while ensuring reliability and high quality, is a challenge best attempted with the tools and techniques developed in the software engineering industry over the last few decades. The PIO team has successfully used these methods for many years, and they have enabled a small team of programmers to create and maintain a complex infrastructure tool which is being used successfully in several atmospheric models.

The PIO development team is a small group of cooperating developers from several institutions. Two to five programmers are generally active at any given time.

This paper does not represent new work by the PIO team, but describes the processes and tools we have been using, with minor variations, for several years.

2 BUILD SYSTEMS

Software systems are usually organized into files and directories, and a build system must be used to correctly build and install the software. The presence of a functional build system enables better use of tools and tests. The PIO project supports two build systems.

2.1 Autotools

PIO uses GNU tools autoconf, automake, and libtool to configure and build PIO in a standard way. This build system supports our agile software development process, and is supported by all CI tools and utilities.

This allows PIO to be installed on any Unix machine with:

```
        ./configure --enable-fortran
        make check install
```

One helpful tool provided by the autotools build system is the “distcheck” target. This causes a full tarball distribution to be created, unpacked, built, tested, and cleaned. It ensures that everything needed to successfully build the software package is included in the distribution tarball. The distcheck target is used in many of the Continuous Integration (CI) builds.

2.1.1 Use of MPE Library

The Multi-Processing Environment (MPE) provides a suite of performance and analysis tools for MPI programs (MPI Site). Use of the MPE library is only supported in the autotools build.

The user must ensure that MPE is installed on the build machine. Ensure that the MPE libraries can be found in the loader path, and the MPE convenience scripts can be found in the path, and use the configure option --enable-mpe in order to use the MPE library.

The MPE library offers four different logging methods. PIO only uses the Logging Library.

1. Tracing library - trace all MPI calls.
2. Animation library - draws amination of running program.
3. Logging library - log MPI calls, also custom logging.
4. Collective and datatype checking library - checks for argument consistency in MPI calls.

2.1.1.1 Building the I/O Stack with MPE

The entire stack of software needs to be built with mpe.

HDF5 was built like this:

```
CC='gcc' CPPFLAGS='-I/usr/local/zlib-1.2.11/include' LDFLAGS='-L/usr/local/zlib-
1.2.11/lib' LIBS='-llmpe -lmpe -lmpi -lpthread' ./configure --prefix=/usr/local/hdf5-
1.10.5_mpe_static --disable-shared --enable-parallel
```

netCDF was built like this:

```
CC='gcc' CPPFLAGS='-I/usr/local/zlib-1.2.11/include -I/usr/local/hdf5-
1.10.5_mpe_static/include' LDFLAGS='-L/usr/local/zlib-1.2.11/lib -L/usr/local/hdf5-
1.10.5_mpe_static/lib' ./configure --prefix=/usr/local/netcdf-c-4.7.0_hdf5-
1.10.5_mpe_static_nodap --disable-shared --disable-dap
```

pnetcdf was built like this:

```
CC=gcc LIBS='-llmpe -lmpe -lmpi -lpthread' ./configure --prefix=/usr/local/pnetcdf-
1.11.0_mpe --disable-shared --disable-cxx --disable-fortran
```

The PIO is built like this:

```
autoreconf -i && CC='gcc' CPPFLAGS='-I/usr/local/pnetcdf-1.11.0_mpe/include -
I/usr/local/zlib-1.2.11/include -I/usr/local/hdf5-1.10.5_mpe_static/include -
I/usr/local/netcdf-c-4.7.0_hdf5-1.10.5_mpe_static_nodap/include' LDFLAGS='-
L/usr/local/pnetcdf-1.11.0_mpe/lib -L/usr/local/zlib-1.2.11/lib -L/usr/local/hdf5-
1.10.5_mpe_static/lib -L/usr/local/netcdf-c-4.7.0_hdf5-1.10.5_mpe_static_nodap/lib'
LIBS=' -lhdf5_hl -lhdf5 -lz -ldl -lm -llmpe -lmpe -lmpi -lpthread' ./configure --disable-
shared --enable-mpe
```

Note that all the mpicc/mpecc compiler wrappers seemed unable to yield the correct results for this build, because the libraries would be listed in an incorrect order. Only by using CC=gcc, and explicitly linking to -llmpe -lmpe -lmpi -lpthread can this be made to build.

Once built in this way, any PIO program will produce a CLOG2 output file. This can be converted to SLOG2 format, and view with the Jumpshot utility.

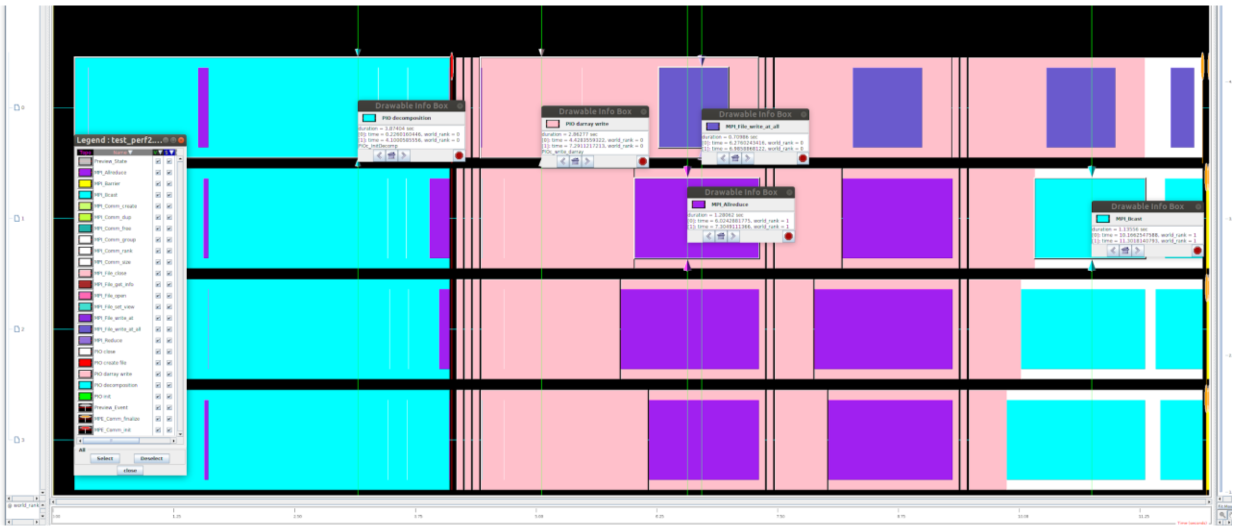


Figure 1: In this jumpshot display of the log file from test_perf2, we see 10 large records written. The large light blue box at the beginning is PIOc_InitDecomp. Then there are 10 darray writes (pink), then the files closes (white). Note that the 4th, 7th, and 10th darray write calls are far longer. (The X axis is time). These are the times the buffer filled up and was flushed to disk.

2.2 CMake

In addition to the autotools build system, a cmake based build system is also provided and maintained for PIO. The cmake-based system does not fully duplicate functionality of the autotools build, but allows the PIO C and Fortran libraries to be built and tested with cmake.

3 REPOSITORY

The PIO code is hosted in a GitHub repository. GitHub provides many graphical tools to help examine the changing code base.

3.1 Development on Branches

All development is done on branches from master.

Since the master branch always builds (enforced by the CI system), each developer always starts with a fully tested and functional version of the PIO libraries. By developing on a branch, the developer ensures that any changes cannot affect other programmers until they have been completed and subjected to review and testing, before being merged to the master branch.

The procedure for creating and starting to use a new branch is:

```
ed@mikado:~/tmp/ParallelIO$ git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.
ed@mikado:~/tmp/ParallelIO$ git pull
Already up to date.
ed@mikado:~/tmp/ParallelIO$ git branch ejh_more_docs
ed@mikado:~/tmp/ParallelIO$ git checkout ejh_more_docs
Switched to branch 'ejh_more_docs'
ed@mikado:~/tmp/ParallelIO$ git push -u origin ejh_more_docs
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'ejh_more_docs' on GitHub by visiting:
remote:      https://github.com/NCAR/ParallelIO/pull/new/ejh_more_docs
remote:
To github.com:NCAR/ParallelIO.git
```

```
 * [new branch]       ejh_more_docs -> ejh_more_docs
Branch 'ejh_more_docs' set up to track remote branch 'ejh_more_docs' from 'origin'.
```

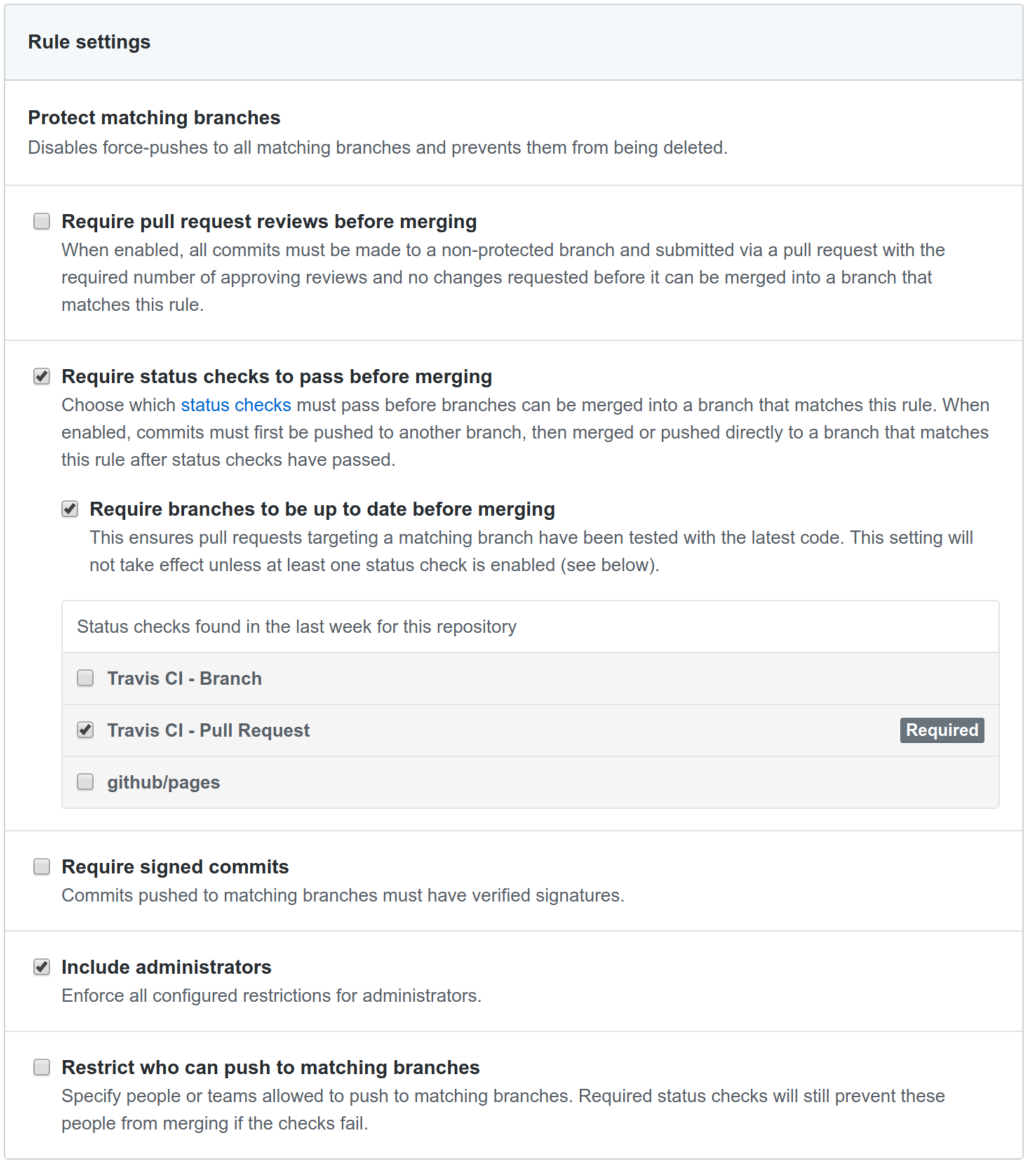
Developers then make changes to the code and commit them to their branch. The branch is tested by the developer CI system.

When the developer has completed the changes, GitHub provides a graphical interface to create the pull request (see Figure 3).

3.2 Special Protection for master Branch

We have special protections for the master branch turned on in our GitHub settings.

1. No commits to master except via Pull Request.
2. No merging of Pull Requests without passing the travis Continuous Integration (CI) system.



Rule settings

Protect matching branches
Disables force-pushes to all matching branches and prevents them from being deleted.

- [ ] **Require pull request reviews before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

- [✓] **Require status checks to pass before merging**
Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

  - [✓] **Require branches to be up to date before merging**
  This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

  Status checks found in the last week for this repository
  - [ ] Travis CI - Branch
  - [✓] Travis CI - Pull Request — Required
  - [ ] github/pages

- [ ] **Require signed commits**
Commits pushed to matching branches must have verified signatures.

- [✓] **Include administrators**
Enforce all configured restrictions for administrators.

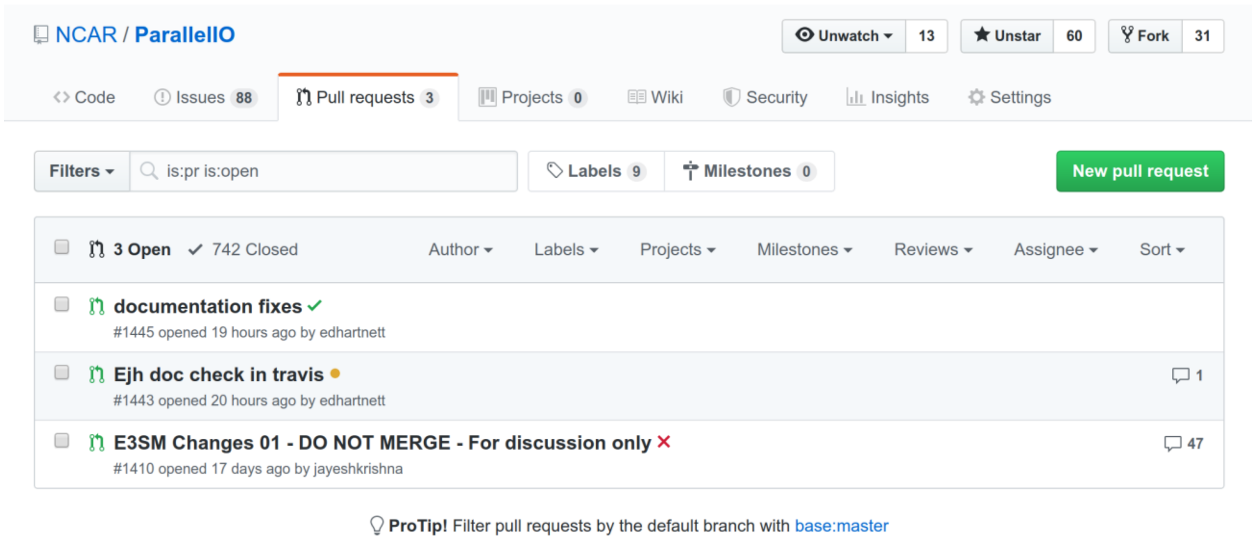
- [ ] **Restrict who can push to matching branches**
Specify people or teams allowed to push to matching branches. Required status checks will still prevent these people from merging if the checks fail.

Figure 1: GitHub settings for branch master ensure that all pull requests are subjected to testing with the Travis CI system, including the pull requests of project administrators. No exceptions are made and all pull requests must pass the Travis CI system in order to be merged with master.

Enforcing these protections ensures that all development is done on branches, and that no changes are merged to master which have not passed the Travis CI system.

3.2 Pull Requests and Code Inspection

By project custom, Pull Requests are left up for about a day before merging, to allow other project participants to review the code. In practice, most code does get reviewed by the senior developers.



documentation fixes ✓
#1445 opened 19 hours ago by edhartnett

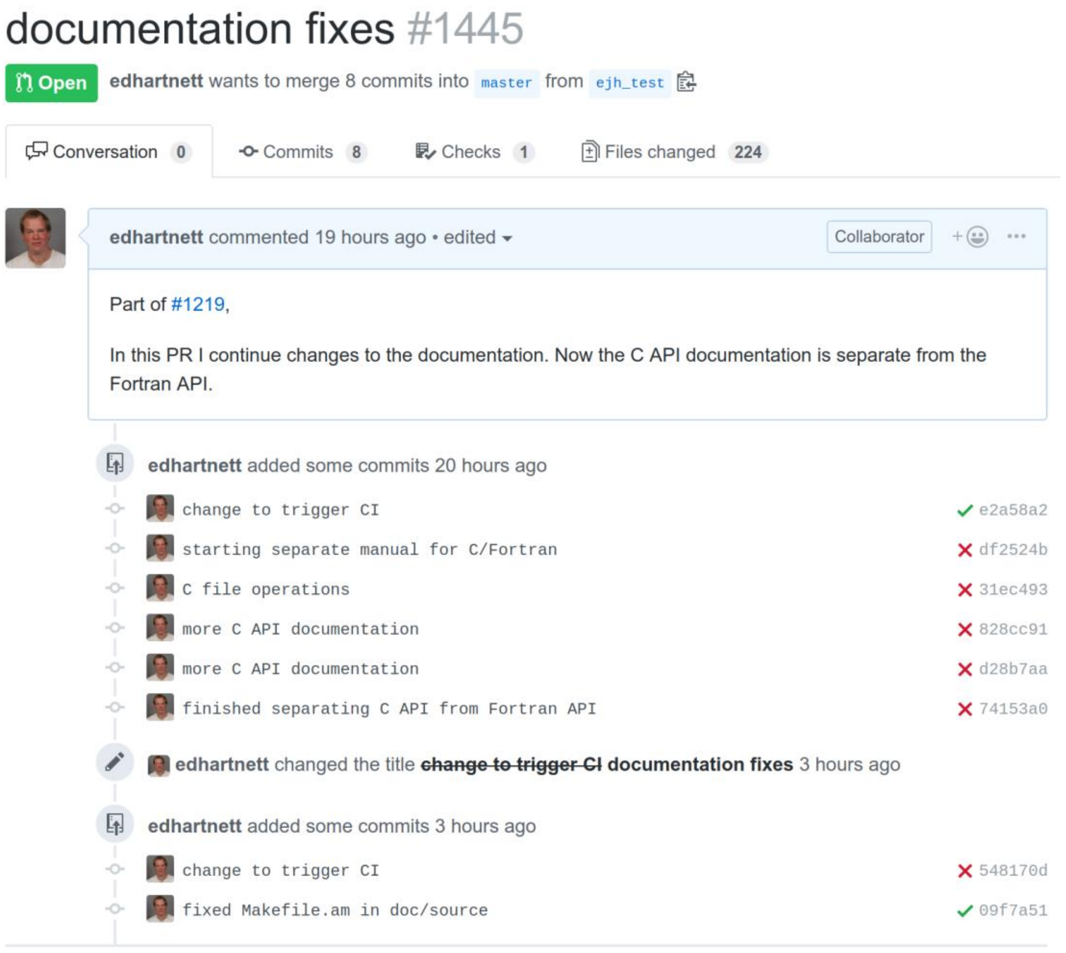
Ejh doc check in travis ●
#1443 opened 20 hours ago by edhartnett — 1

E3SM Changes 01 - DO NOT MERGE - For discussion only ✗
#1410 opened 17 days ago by jayeshkrishna — 47

ProTip! Filter pull requests by the default branch with base:master

Figure 2: The Pull Request page shows three open pull requests. The first has passed the Travis CI checks, the second is currently being tested, and the third has failed the Travis CI tests.

Each pull request includes a title and text description describing the proposed changes.



documentation fixes #1445

Open edhartnett wants to merge 8 commits into master from ejh_test

Conversation 0 Commits 8 Checks 1 Files changed 224

edhartnett commented 19 hours ago • edited Collaborator

Part of #1219,

In this PR I continue changes to the documentation. Now the C API documentation is separate from the Fortran API.

edhartnett added some commits 20 hours ago

```
change to trigger CI                              ✓ e2a58a2
starting separate manual for C/Fortran            ✗ df2524b
C file operations                                 ✗ 31ec493
more C API documentation                          ✗ 828cc91
more C API documentation                          ✗ d28b7aa
finished separating C API from Fortran API        ✗ 74153a0
```

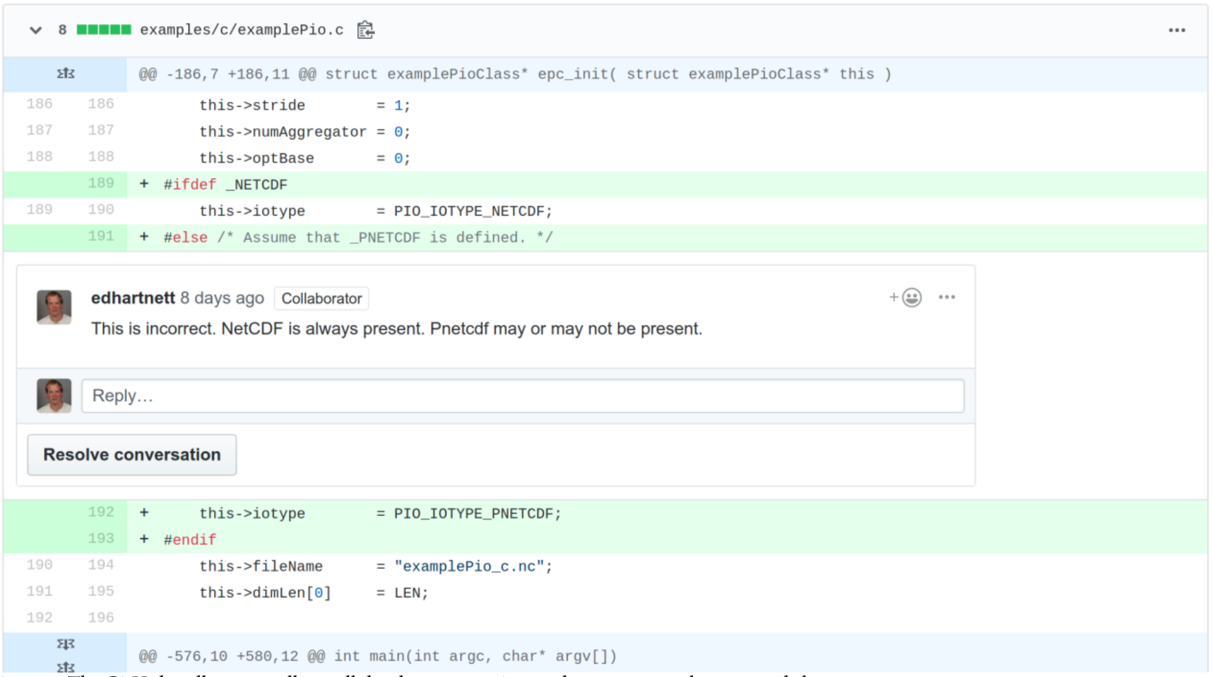
edhartnett changed the title ~~change to trigger CI~~ **documentation fixes** 3 hours ago

edhartnett added some commits 3 hours ago

```
change to trigger CI                              ✗ 548170d
fixed Makefile.am in doc/source                   ✓ 09f7a51
```

Figure 3: Each pull request includes a description of changes from the developer. This should include a link to the related GitHub issue, which is automatically provided by GitHub when the issue number is cited with a "#" first. (This PR includes a much higher than usual number of files, because it includes the built documentation, which has changed with this PR.)

Pull requests allow other developers to examine and comment on the proposed changes.



8 examples/c/examplePio.c

```
@@ -186,7 +186,11 @@ struct examplePioClass* epc_init( struct examplePioClass* this )
         this->stride         = 1;
         this->numAggregator  = 0;
         this->optBase        = 0;
+ #ifdef _NETCDF
         this->iotype         = PIO_IOTYPE_NETCDF;
+ #else /* Assume that _PNETCDF is defined. */
```

edhartnett 8 days ago Collaborator

This is incorrect. NetCDF is always present. Pnetcdf may or may not be present.

```
+        this->iotype         = PIO_IOTYPE_PNETCDF;
+ #endif
         this->fileName       = "examplePio_c.nc";
         this->dimLen[0]      = LEN;

@@ -576,10 +580,12 @@ int main(int argc, char* argv[])
```

Figure 4: The GitHub pull request allows all developers to review and comment on the proposed changes.

4 DEVELOPMENT PROCESS

PIO development follows an agile process which incorporates test-driven development, continuous integration, and static and dynamic code analysis.

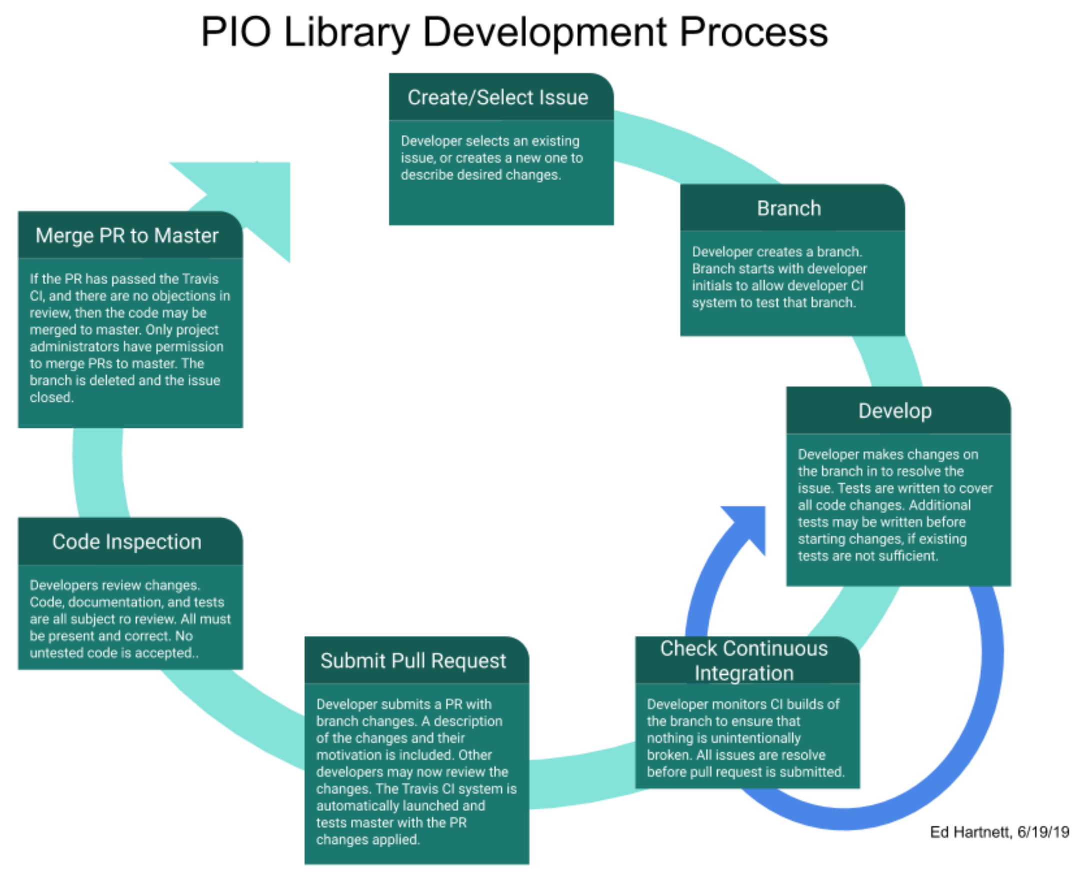

PIO Library Development Process
Create/Select Issue
Developer selects an existing issue, or creates a new one to describe desired changes.
Branch
Developer creates a branch. Branch starts with developer initials to allow developer CI system to test that branch.
Develop
Developer makes changes on the branch in to resolve the issue. Tests are written to cover all code changes. Additional tests may be written before starting changes, if existing tests are not sufficient.
Check Continuous Integration
Developer monitors CI builds of the branch to ensure that nothing is unintentionally broken. All issues are resolve before pull request is submitted.
Submit Pull Request
Developer submits a PR with branch changes. A description of the changes and their motivation is included. Other developers may now review the changes. The Travis CI system is automatically launched and tests master with the PR changes applied.
Code Inspection
Developers review changes. Code, documentation, and tests are all subject ro review. All must be present and correct. No untested code is accepted..
Merge PR to Master
If the PR has passed the Travis CI, and there are no objections in review, then the code may be merged to master. Only project administrators have permission to merge PRs to master. The branch is deleted and the issue closed.
Ed Hartnett, 6/19/19

Figure 5: The PIO development process follows the agile model. The master branch is never allowed to break, with all tests always passing before code is merged to master. The code is always ready for public release.

The development process involves the following steps:

1. Select or Create Issue - a GitHub issue describing the problem to be fixed, or the new feature to be added.
2. Developer takes branch from master.
3. Developer works on branch, writing tests and changing code.
4. Each commit to the branch is tested with Jenkins CI for many configurations. The developer monitors the Jenkins dashboard, learning quickly (within ~10 minutes) if anything has been inadvertently broken. The developer CI system also performs static and dynamic code analysis to allow the developer to target untested areas and quickly catch memory and other mistakes detectable to the CI system.
5. When developer is satisfied that a complete, correct, and well-tested body of work is ready (which includes passing all developer CI), a pull request is submitted for the branch. The Travis CI run is automatically started on the branch.
6. All developers inspect the code. Comments may result in further changes (i.e. commits to the branch). Each new commit to the branch triggers a complete Travis CI run.
7. When all developers are satisfied, the project admins merge the pull request with the master branch. Completed issues are closed with the pull request merge.

4.1 Issues

GitHub issues are used to keep track of bugs and new features. Each pull request is associated with at least one issue. Within the pull request description, associated issues are enumerated, allowing GitHub to hyper-link to the issue(s).

Minor issues may be added and resolved quickly. When resolving a larger issue, smaller issues may be discovered, entered as issues, and then resolved in pull request(s) also associated with other issues.

4.2 Branching

All development is done on branches which isolate development efforts. Branches may only be merged to master with pull requests, which enforce Travis CI testing.

4.3 Test-Driven Development

Development is test-driven, and may be Test First Development (TFD) or iterative Test Last Development (TLD).

TFD is the practice of writing test code first. On PIO this is most often used when adding new features or API calls.

Iterative TLD is similar to TFD but involves iteratively writing the tests as functionality is being developed or bugs are fixed.

4.4 Developer CI

The developer CI is invaluable for providing very rapid feedback as code is being developed. Many configurations are checked.

If the master branch changes during the development process, it must be merged into the developer branch in order to prepare an up-to-date pull request. At that time, the developer CI system will test the new master with the developing branch to ensure that the merged changes to master have not broken any tests on the development branch.

4.5 Pull-Requests

Project etiquette suggests that pull requests should be small and focused. More frequent pull requests are better than less frequent, because they decrease the feedback latency for the developer (Rahman, A.)

The PIO project often merges one or more pull requests per day (PIO GitHub Site).

When a pull request is submitted, a Travis CI job is launched on a branch which contains master merged with the proposed pull request. Pull request code is always tested by the Travis CI system before it is merged to master. If any of the checks of the Travis CI system fail, the pull request will not be merged.

4.6 Code Review

Code review is conducted with GitHub graphical tools (see Figure 4). Project admins ensure that all comments and questions are resolved before a pull request is merged.

In addition to code changes, documentation and testing are also reviewed. If testing is not adequate, the developer will be directed to add more tests.

Like many projects with few active developers, we struggle to review all code, and many pull requests from core contributors go unreviewed. For this reason, we do not require review on GitHub.

4.7 Merge to Master

Only project admins may merge to master. Pull requests are not merged to master unless all CI systems are passing, and all code review comments and questions have been resolved.

Once merged, the pull request branch is deleted. An overabundance of old branches causes problems when adding new test runs to the developer CI system. For this reason only one or two developer branches exist at any time.

5 AUTOMATED TESTING

The PIO C and Fortran libraries come with many unit tests which are run when “make check” is invoked when building the PIO libraries. The test codes are organized in three subdirectories of the tests directory:

- cunit - all C tests are in this directory.
- general - most Fortran tests are in this directory.
- performance - this directory contains the pioperf performance measurement tool.

5.1 Testing with MPI

Testing with MPI presents a challenge - jobs may be launched in different ways on different machines.

The approach used in the PIO library is to have bash shell scripts which run the tests using mpiexec. This works on all developer machines and most linux workstations with MPI installed.

PIO also provides a configure option --disable-test-runs which causes “make test” to build the tests, but not attempt to run the MPI-based tests. This allows the user to build the tests and then run them manually with whatever batch commands may be necessary.

The following bash script (tests/cunit/run_tests.sh) demonstrates how MPI tests are run. In the Makefile.am file, the test programs are listed as check_PROGRAMS, which causes them to be built, and run_tests.sh is added to TESTS, which causes the run_tests.sh script to be run when the “check” target is built.

```
#!/bin/sh
# This is a test script for PIO.
# Ed Hartnett

# Stop execution of script if error is returned.
set -e

# Stop loop if ctrl-c is pressed.
trap exit INT TERM

printf 'running PIO tests...\n'

PIO_TESTS='test_intercomm2 test_async_mpi test_spmd test_rearr test_async_simple '\
'test_async_3proc test_async_4proc test_iosystem2_simple test_iosystem2_simple2 '\
'test_iosystem2 test_iosystem3_simple test_iosystem3_simple2 test_iosystem3 test_pioc '\
'test_pioc_unlim test_pioc_putget test_pioc_fill test_darray test_darray_multi '\
'test_darray_multivar test_darray_multivar2 test_darray_multivar3 test_darray_1d '\
'test_darray_3d test_decomp_uneven test_decomps test_darray_async_simple '\
'test_darray_async test_darray_async_many test_darray_2sync test_async_multicomp '\
'test_darray_fill'

success1=true
success2=true
for TEST in $PIO_TESTS
do
    success1=false
    echo "running ${TEST}"
    mpiexec -n 4 ./${TEST} && success1=true
    if test $success1 = false; then
        break
    fi
done

PIO_TESTS_8='test_async_multi2  test_async_manyproc'

for TEST in $PIO_TESTS_8
do
    success2=false
    echo "running ${TEST}"
    mpiexec -n 8 ./${TEST} && success2=true
    if test $success2 = false; then
        break
    fi
done

# Did we succeed?
if test x$success1 = xtrue -a x$success2 = xtrue; then
    exit 0
fi
exit 1
```

6 CONTINUOUS INTEGRATION

Continuous Integration (CI) is a practice of running all tests, with all (or many) possible build variations, each time software is committed to the repository, or on some other frequent schedule (Fowler, M.).

The benefits of CI, when used in conjunction with other agile practices, have been studied. The near-immediate feedback to developers results in improved productivity and increased quality, when CI is used in conjunction with frequent commits (Rahman, Akond et. al.). CI systems are highly favored by programmers than try them (Yangyang Zhao et. al.), and result in increased issue resolution, decrease in pull request integration time, and increased frequency of project releases, as well as higher programmer confidence that the build is not being broken (M. Hilton et. al.).

6.1 Continuous Integration Systems Used by PIO

The PIO C and Fortran libraries are tested with several Continuous Integration (CI) systems. These systems perform regression testing of changes made to the code base, and ensure that the master branch always builds and passes all tests.

The different CI systems in use on PIO work together and meet distinct needs. The CI systems in use on PIO are:

- Travis - checks submitted pull requests.
- Jenkins - provides very fast feedback on many variations of the build.
- CMake - runs on HPC systems.

6.1.1. The Travis CI System

Working with GitHub, the travis CI system builds and tests each submitted pull request, before it is merged with master. The travis CI system is operated by a commercial entity which charges for private, but allows open-source projects to test for free.

The Travis build is invoked automatically by GitHub whenever a pull request is submitted. It runs the tests and marks the pull request accordingly. It will be re-run whenever developers update the branch on which the pull request is based. This allows the submitter of the pull request to iterate with the CI system until the pull request passes all tests.

Travis builds and tests PIO with both the autotools and cmake build systems. Additionally the GNU address sanitizer is used, so any memory problems will cause the tests to fail. The Travis build also ensures that the doxygen documentation builds without warnings.

If programmers submit a pull request with failing tests, memory errors, or missing documentation, Travis will fail the pull request and the developer may continue to work on their branch until the problems are resolved.

The Travis build is controlled by a file in the main project directory called .travis.yml:

```
language: c
dist: trusty
sudo: false

branches:
  only:
```

```
  - master

addons:
  apt:
    sources:
    - ubuntu-toolchain-r-test
    packages:
    - pkg-config netcdf-bin libnetcdf-dev openmpi-bin libopenmpi-dev gfortran doxygen
graphviz

before_install:
  - test -n $CC && unset CC
  - test -n $FC && unset FC
  - test -n $CPPFLAGS && unset CPPFLAGS
  - test -n FCFLAGS && unset FCFLAGS

before_script:
  - export CC=mpicc
  - export FC=mpif90
  - export CPPFLAGS='-I/usr/include'
  - wget https://parallel-netcdf.github.io/Release/pnetcdf-1.11.0.tar.gz
  - tar -xzvf pnetcdf-1.11.0.tar.gz
  - ls -l
  - pushd pnetcdf-1.11.0
  - ./configure --prefix=/usr --enable-shared
  - make
  - sudo make install
  - popd
env:
  global:
    - CC=mpicc
    - FC=mpif90
    - CPPFLAGS='-I/usr/include'
    - CFLAGS='-std=c99'
    - LDFLAGS='-L/usr/lib'

script:
  - ls -l /usr/include
  - autoreconf -i
  - export CFLAGS='-std=c99  -fsanitize=address -fno-omit-frame-pointer'
  - export FFLAGS='-fsanitize=address -fno-omit-frame-pointer'
  - export FCFLAGS='-fsanitize=address -fno-omit-frame-pointer'
  - export DISTCHECK_CONFIGURE_FLAGS='--enable-fortran'
  - ./configure --enable-fortran --enable-developer-docs
  - make
  - make -j distcheck
  - rm -rf build
  - mkdir build
  - cd build
  - cmake -DPIO_HDF5_LOGGING=On -DPIO_USE_MALLOC=On -DPIO_ENABLE_LOGGING=On -
DPIO_ENABLE_TIMING=Off ..
  - make VERBOSE=1
  - make tests VERBOSE=1
  - make test VERBOSE=1
```

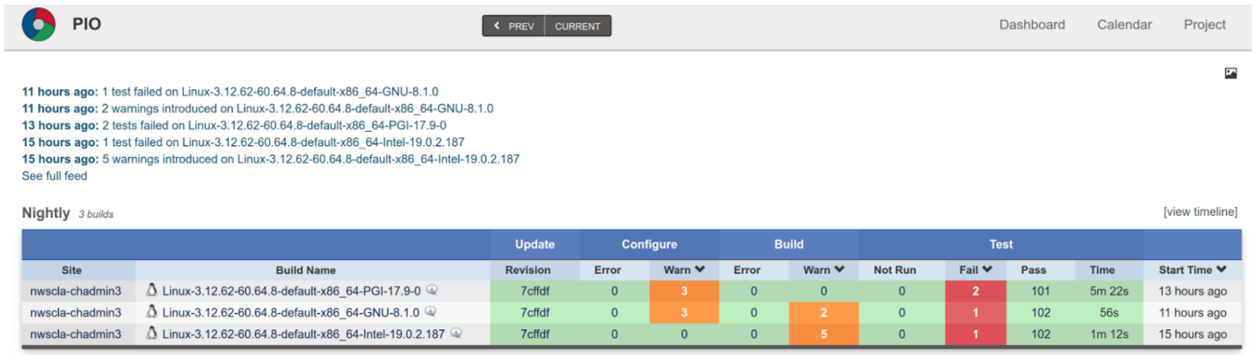
6.1.2 Jenkins

A jenkins CI server, running on a developer machine, tests every branch starting with the developer initials. This allows full CI testing of each developer branch (see Figure 7).

The Jenkins CI server polls the GitHub repository every 15 minutes for changes, or may be manually launched by the developer. If Jenkins finds any new branches that start with the developer's initials, or finds new commits to any existing branches that start with the developer initials, the full CI test suite is launched for that branch.

6.1.3 CDash

A CDash server builds (using cmake) the PIO C and Fortran libraries each night at NCAR on the Cheyenne HPC system. The results are reported on a web interface. The advantage of the CDash server for PIO is that it can execute builds on several HPC systems of interest. These systems are difficult to build on from external tools, but the CDash server is installed within the security perimeter and can build on these systems.



11 hours ago: 1 test failed on Linux-3.12.62-60.64.8-default-x86_64-GNU-8.1.0
11 hours ago: 2 warnings introduced on Linux-3.12.62-60.64.8-default-x86_64-GNU-8.1.0
13 hours ago: 2 tests failed on Linux-3.12.62-60.64.8-default-x86_64-PGI-17.9-0
15 hours ago: 1 test failed on Linux-3.12.62-60.64.8-default-x86_64-Intel-19.0.2.187
15 hours ago: 5 warnings introduced on Linux-3.12.62-60.64.8-default-x86_64-Intel-19.0.2.187
See full feed

Nightly *3 builds* [view timeline]

		Update	Configure		Build		Test				
Site	Build Name	Revision	Error	Warn	Error	Warn	Not Run	Fail	Pass	Time	Start Time
nwscla-chadmin3	Linux-3.12.62-60.64.8-default-x86_64-PGI-17.9-0	7cffdf	0	3	0	0	0	2	101	5m 22s	13 hours ago
nwscla-chadmin3	Linux-3.12.62-60.64.8-default-x86_64-GNU-8.1.0	7cffdf	0	3	0	2	0	1	102	56s	11 hours ago
nwscla-chadmin3	Linux-3.12.62-60.64.8-default-x86_64-Intel-19.0.2.187	7cffdf	0	0	0	5	0	1	102	1m 12s	15 hours ago

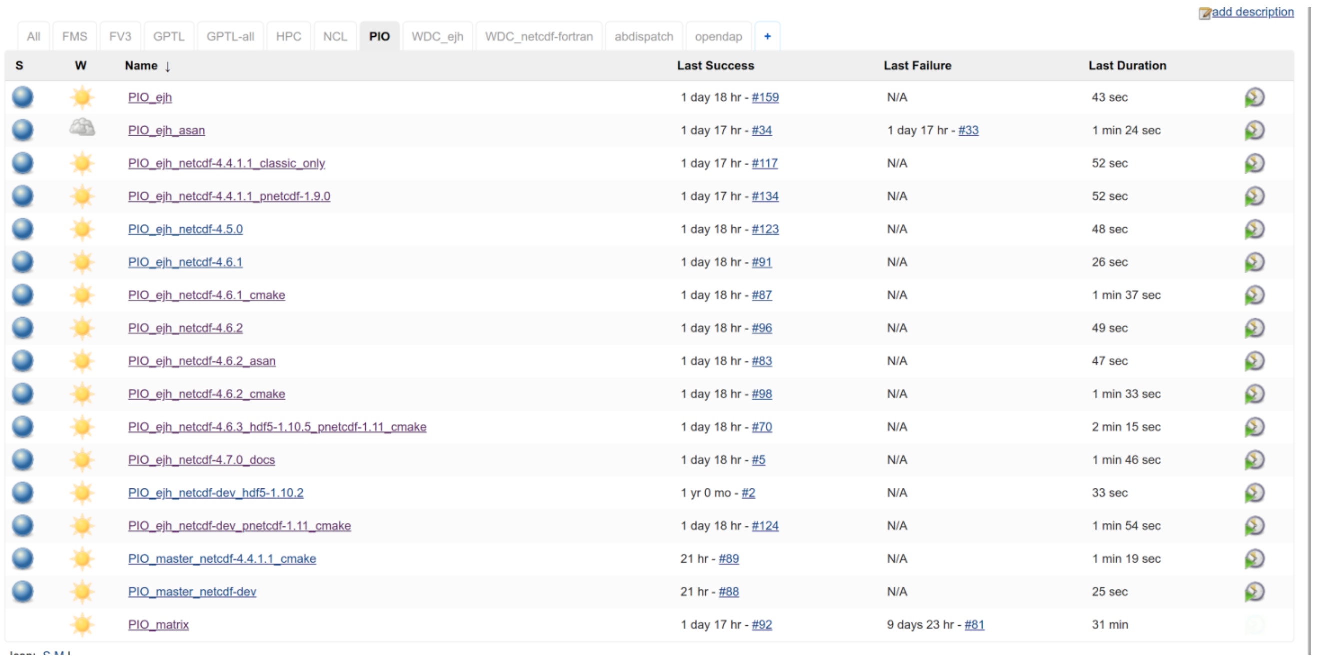
Figure 6: The CDash test system builds and tests PIO periodically. The web interface allows developers to inspect failures.

6.2 Testing Multiple Configurations

Many build configurations are possible with PIO. Some of the variants between builds include:

- different compilers or compiler versions.
- MPICH vs. OpenMPI MPI libraries.
- different versions of netCDF, HDF5, and pnetcdf.
- different combinations of configure options, such as --enable-fortran, --enable-logging, --enable-timing, --enable-docs.
- different versions of other tools used in the build (doxygen, make, etc.).
- different build systems (autotools vs. cmake).

These differences are most comprehensively handled for PIO by the Jenkins CI system. More than 20 different build jobs are launched each time the system is run, in addition to a matrix build to check different combinations of dependent library versions.



add description

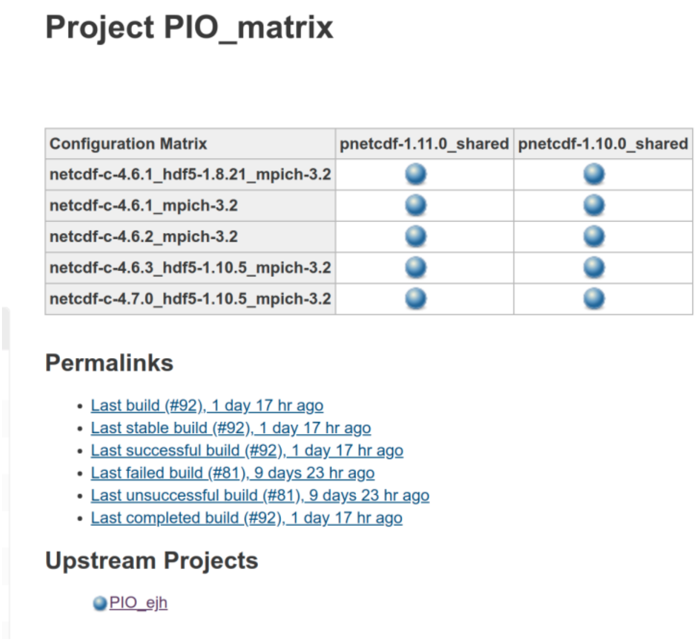
All | FMS | FV3 | GPTL | GPTL-all | HPC | NCL | PIO | WDC_ejh | WDC_netcdf-fortran | abdispatch | opendap | +

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	PIO_ejh	1 day 18 hr - #159	N/A	43 sec
●	☁	PIO_ejh_asan	1 day 17 hr - #34	1 day 17 hr - #33	1 min 24 sec
●	☀	PIO_ejh_netcdf-4.4.1.1_classic_only	1 day 17 hr - #117	N/A	52 sec
●	☀	PIO_ejh_netcdf-4.4.1.1_pnetcdf-1.9.0	1 day 17 hr - #134	N/A	52 sec
●	☀	PIO_ejh_netcdf-4.5.0	1 day 18 hr - #123	N/A	48 sec
●	☀	PIO_ejh_netcdf-4.6.1	1 day 18 hr - #91	N/A	26 sec
●	☀	PIO_ejh_netcdf-4.6.1_cmake	1 day 18 hr - #87	N/A	1 min 37 sec
●	☀	PIO_ejh_netcdf-4.6.2	1 day 18 hr - #96	N/A	49 sec
●	☀	PIO_ejh_netcdf-4.6.2_asan	1 day 18 hr - #83	N/A	47 sec
●	☀	PIO_ejh_netcdf-4.6.2_cmake	1 day 18 hr - #98	N/A	1 min 33 sec
●	☀	PIO_ejh_netcdf-4.6.3_hdf5-1.10.5_pnetcdf-1.11_cmake	1 day 18 hr - #70	N/A	2 min 15 sec
●	☀	PIO_ejh_netcdf-4.7.0_docs	1 day 18 hr - #5	N/A	1 min 46 sec
●	☀	PIO_ejh_netcdf-dev_hdf5-1.10.2	1 yr 0 mo - #2	N/A	33 sec
●	☀	PIO_ejh_netcdf-dev_pnetcdf-1.11_cmake	1 day 18 hr - #124	N/A	1 min 54 sec
●	☀	PIO_master_netcdf-4.4.1.1_cmake	21 hr - #89	N/A	1 min 19 sec
●	☀	PIO_master_netcdf-dev	21 hr - #88	N/A	25 sec
	☀	PIO_matrix	1 day 17 hr - #92	9 days 23 hr - #81	31 min

Figure 7: The Jenkins CI server runs many test configurations of PIO, each time a change is made to any of the developer's branches.

6.3 Testing With Different Third-Party Library Versions

The Jenkins CI server also runs a "matrix" build, which builds all combinations of a matrix of parameters, in this case, the different versions of pnetcdf and netCDF. This allows for continuous confirmation that no changes to the code will break PIO for the last several releases of each of pnetcdf and netCDF. The matrix build includes builds of both the autotools build system and the cmake build system.



Project PIO_matrix

Configuration Matrix	pnetcdf-1.11.0_shared	pnetcdf-1.10.0_shared
netcdf-c-4.6.1_hdf5-1.8.21_mpich-3.2	●	●
netcdf-c-4.6.1_mpich-3.2	●	●
netcdf-c-4.6.2_mpich-3.2	●	●
netcdf-c-4.6.3_hdf5-1.10.5_mpich-3.2	●	●
netcdf-c-4.7.0_hdf5-1.10.5_mpich-3.2	●	●

Permalinks

- Last build (#92), 1 day 17 hr ago
- Last stable build (#92), 1 day 17 hr ago
- Last successful build (#92), 1 day 17 hr ago
- Last failed build (#81), 9 days 23 hr ago
- Last unsuccessful build (#81), 9 days 23 hr ago
- Last completed build (#92), 1 day 17 hr ago

Upstream Projects

● PIO_ejh

Figure 8 - The Jenkins matrix build performs continuous integration of the PIO C/Fortran libraries, for all combinations of supported releases of pnetcdf and netCDF.

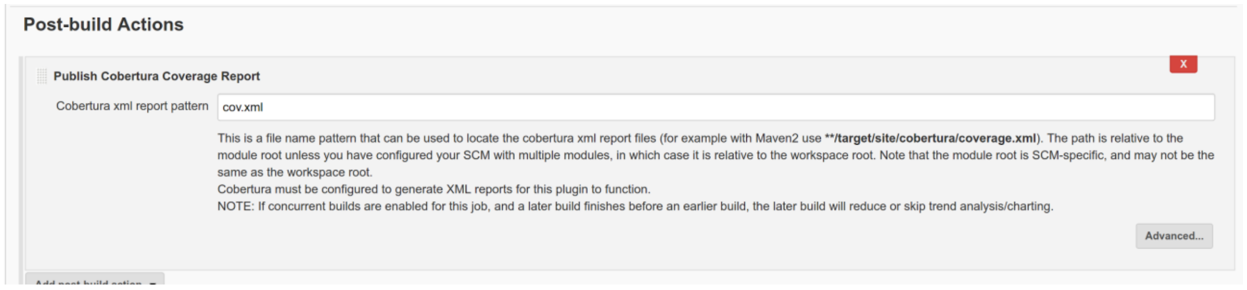
6.4 Code Test Coverage

The gcov tool is built into the GNU compilers, and allows us to monitor how much of our code is covered by tests.

The following script, run by Jenkins, generates a code coverage report, cov.xml.

```
autoreconf -i
export CC=mpicc
export FC=mpifort
export CFLAGS='-DNDEBUG -fprofile-arcs -ftest-coverage'
export LDFLAGS='-L/usr/local/netcdf-c-4.7.0_hdf5-1.10.5_mpich-3.2/lib -
L/usr/local/pnetcdf-1.11.0_shared/lib'
export CPPFLAGS='-I/usr/local/netcdf-c-4.7.0_hdf5-1.10.5_mpich-3.2/include -
I/usr/local/pnetcdf-1.11.0_shared/include -I/usr/local/netcdf-fortran-
4.4.5_c_4.6.3_mpich-3.2/include'
./configure --enable-logging --enable-fortran --enable-docs
make all check
gcovr -r . --xml-pretty>cov.xml
make -j distclean
```

This coverage report is automatically graphed by Jenkins as a “post-build action”.


Post-build Actions

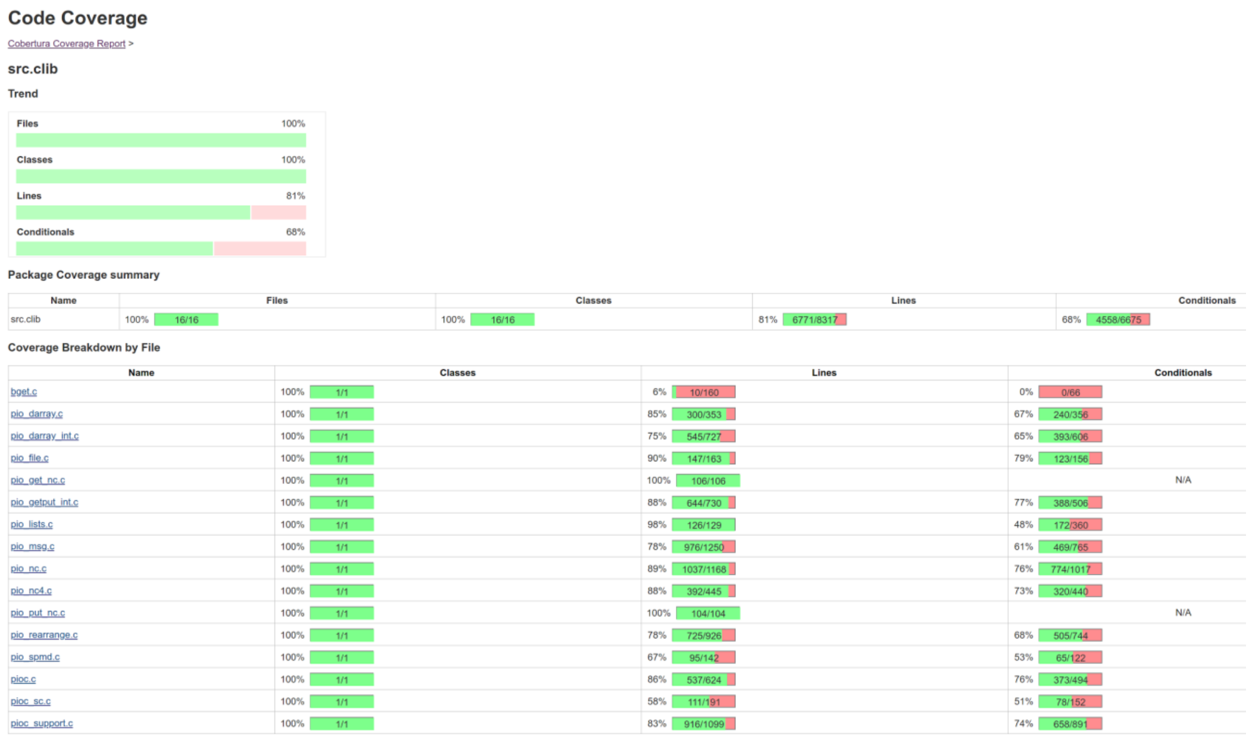
Publish Cobertura Coverage Report

Cobertura xml report pattern: cov.xml

This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use ****/target/site/cobertura/coverage.xml**). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root.
Cobertura must be configured to generate XML reports for this plugin to function.
NOTE: If concurrent builds are enabled for this job, and a later build finishes before an earlier build, the later build will reduce or skip trend analysis/charting.

Figure 9: Turning on the coverage report in the Jenkins CI system.

The coverage report provides a file-by-file coverage report.



Code Coverage

Cobertura Coverage Report >

src.clib

Trend

Files	100%
Classes	100%
Lines	81%
Conditionals	68%

Package Coverage summary

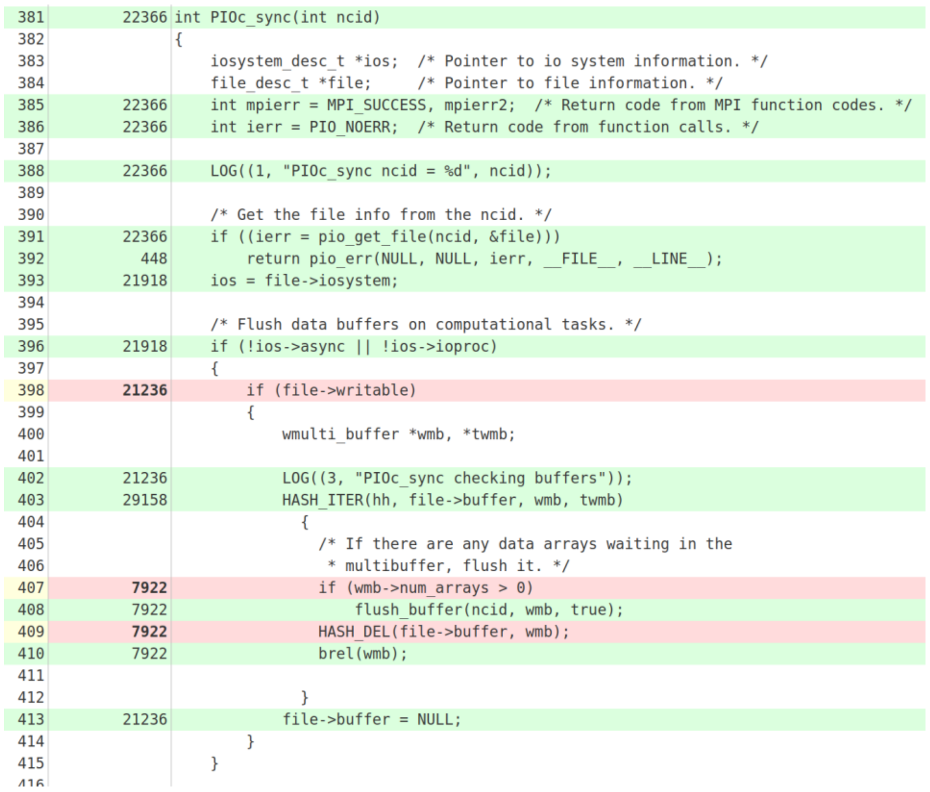
Name	Files	Classes	Lines	Conditionals
src.clib	100% 16/16	100% 16/16	81% 6771/8317	68% 4558/6675

Coverage Breakdown by File

Name	Classes	Lines	Conditionals
bget.c	100% 1/1	6% 10/160	0% 0/66
pio_darray.c	100% 1/1	85% 300/353	67% 240/356
pio_darray_int.c	100% 1/1	75% 545/727	65% 393/606
pio_file.c	100% 1/1	90% 147/163	79% 123/156
pio_get_nc.c	100% 1/1	100% 106/106	N/A
pio_getput_int.c	100% 1/1	88% 644/730	77% 388/506
pio_lists.c	100% 1/1	98% 126/129	48% 172/360
pio_msg.c	100% 1/1	78% 976/1250	61% 469/765
pio_nc.c	100% 1/1	89% 1037/1168	76% 774/1017
pio_nc4.c	100% 1/1	88% 392/445	73% 320/440
pio_put_nc.c	100% 1/1	100% 104/104	N/A
pio_rearrange.c	100% 1/1	78% 725/926	68% 505/744
pio_spmd.c	100% 1/1	67% 95/142	53% 65/122
pioc.c	100% 1/1	86% 537/624	76% 373/494
pioc_sc.c	100% 1/1	58% 111/191	51% 78/152
pioc_support.c	100% 1/1	83% 916/1099	74% 658/891

Figure 10: The code coverage report provides a test coverage for each file, in lines of code and conditionals.

Drilling down further into the file, the coverage report shows which lines have been tested, allowing us to target tests at untested code.



```
381       22366 int PIOc_sync(int ncid)
382             {
383                 iosystem_desc_t *ios;  /* Pointer to io system information. */
384                 file_desc_t *file;     /* Pointer to file information. */
385       22366     int mpierr = MPI_SUCCESS, mpierr2;  /* Return code from MPI function codes. */
386       22366     int ierr = PIO_NOERR;  /* Return code from function calls. */
387
388       22366     LOG((1, "PIOc_sync ncid = %d", ncid));
389
390                 /* Get the file info from the ncid. */
391       22366     if ((ierr = pio_get_file(ncid, &file)))
392         448         return pio_err(NULL, NULL, ierr, __FILE__, __LINE__);
393       21918     ios = file->iosystem;
394
395                 /* Flush data buffers on computational tasks. */
396       21918     if (!ios->async || !ios->ioproc)
397                 {
398       21236         if (file->writable)
399                     {
400                         wmulti_buffer *wmb, *twmb;
401
402       21236             LOG((3, "PIOc_sync checking buffers"));
403       29158             HASH_ITER(hh, file->buffer, wmb, twmb)
404                           {
405                             /* If there are any data arrays waiting in the
406                              * multibuffer, flush it. */
407        7922                 if (wmb->num_arrays > 0)
408        7922                     flush_buffer(ncid, wmb, true);
409        7922                 HASH_DEL(file->buffer, wmb);
410        7922                 brel(wmb);
411
412                           }
413       21236             file->buffer = NULL;
414                     }
415                 }
416
```

Figure 11: The code coverage report shows which lines of code have been tested (green), and which have not (red).

6.5 Memory Leak Detection

The address sanitizer tool is built into the GNU compilers. It allows the compilers to build extensive memory testing into the code, which, when run, will error on any incorrect memory operation, such as reading or writing beyond allocated limits, reusing memory that has been freed, or allocating memory and failing to free it.

Address sanitizer runs can be achieved by adding flags to the CFLAGS and FCFLAGS for the build. The following script is run by Jenkins to perform address sanitizer runs on PIO:

```
autoreconf -i
export CC=mpicc
export CFLAGS='-Wall -g -fsanitize=address -fno-omit-frame-pointer'
export FCFLAGS='-Wall -g -fsanitize=address -fno-omit-frame-pointer'
export LDFLAGS='-L/usr/local/netcdf-c-4.6.2_mpich-3.2/lib -L/usr/local/pnetcdf-
1.11.0_shared/lib'
export CPPFLAGS='-I/usr/local/netcdf-c-4.6.2_mpich-3.2/include -I/usr/local/pnetcdf-
1.11.0_shared/include -I/usr/local/netcdf-fortran-4.4.5_c_4.6.3_mpich-3.2/include'
export DISTCHECK_CONFIGURE_FLAGS=--enable-fortran
./configure
make -j check
make -j distclean
```

6.6 Compiler Warnings

Compiler warnings can be important indicators of code quality issues. The PIO code compiles warning-free with GNU compilers, with the -Wall option. Warnings are treated as bugs and fixed to maintain a warning-free build. Only in a build without warnings can developers easily see new warnings, which may identify a coding mistake before it is committed to the code base.

7 DOCUMENTATION

Documentation does not need to be built by the end user of PIO. The latest PIO documentation is available at the PIO GitHub site. As is customary, the documentation files are also included with the tarball distribution, for completeness.

The documentation for both the C and Fortran libraries is built with the doxygen tool. Doxygen allows documentation to be included in the code files as comments. When Doxygen builds the documentation, it first examines a configuration file, called the Doxyfile. Both doxygen and a graphics tool GraphViz are required to build the documentation.

The PIO documentation is best built with the autotools build system. To build the documentation, the developer should use the --enable-docs configure option. This will cause the documentation to be built by “make all”. If there are any warnings in the documentation, the build will fail. This allows the travis CI system to build documentation, and reject pull requests without correct documentation.

The configure script turns a Doxyfile.in into the Doxyfile that is used by doxygen. This allows configuration options to control how the documentation is built. In the configure.ac script we have:

```
# If building docs, process Doxyfile.in into Doxyfile.
if test "x$enable_docs" = xyes; then
   AC_SUBST([CMAKE_CURRENT_SOURCE_DIR], ["."])
   AC_SUBST([CMAKE_BINARY_DIR], [".."])
   if test "x$enable_fortran" = xno; then
      AC_MSG_ERROR([--enable-fortran is required for documentation builds.])
   fi
   AC_SUBST([FORTRAN_SRC_FILES], ["../src/flib/piodarray.f90  ../src/flib/pio.F90
../src/flib/pio_kinds.F90  ../src/flib/piolib_mod.f90  ../src/flib/pionfatt_mod_2.f90
../src/flib/pio_nf.F90  ../src/flib/pionfget_mod_2.f90  ../src/flib/pionfput_mod.f90
../src/flib/pio_support.F90  ../src/flib/pio_types.F90"])
   if test "x$enable_developer_docs" = xyes; then
      AC_SUBST([C_SRC_FILES], ["../src/clib"])
   else
      AC_SUBST([C_SRC_FILES], ["../src/clib/pio_nc.c ../src/clib/pio_nc4.c
../src/clib/pio_darray.c ../src/clib/pio_get_nc.c ../src/clib/pio_put_nc.c
../src/clib/pioc_support.c ../src/clib/pioc.c ../src/clib/pio_file.c ../src/clib/pio.h"])
   fi
   AC_CONFIG_FILES([doc/Doxyfile])
fi
```

This causes the documentation to include all C files if --enable-developer-docs is used, and just the C files that contain PIO's public API if --enable-docs is used.

Documentation may also be built by the CMake build system when CMake is invoked with the option `-DPIO_ENABLE_DOC=On`. This causes CMake to generate a Doxyfile from our Doxyfile.in, allowing us to change documentation configuration options from CMake.

Once the CMake build configuration is complete, the documentation may be built by running the command `doxygen` in the build/doc directory. This will read the Doxyfile for settings.

7.1 Documentation Hosting on GitHub

Using the github pages feature allows us to easily maintain the latest documentation on-line. In the project GitHub settings, we have selected to activate GitHub pages from the /gh-pages branch.

The following settings in the Doxyfile allow the documentation to be built in the top level docs directory:

```
OUTPUT_DIRECTORY       = ../..
HTML_OUTPUT            = docs
```

To update the documentation on GitHub, another clone of the repository is taken, and switched to the gh-pages branch. This branch contains only the built documentation. To update the documentation the existing documentation is removed, then the newly-generated documentation copied from the docs folder where it has been generated. This is then added, committed, and pushed to GitHub.

The files committed to the gh-pages branch are displayed to the user at the GitHub pages site for the project.

8 RELEASE ENGINEERING

Releases of PIO are distributed on the GitHub site as release tarballs. Each tarball contains everything needed to build with either autotools or cmake.

8.1 Steps to Release

The following steps are followed for release:

1. **Update Release Notes:** The release notes are updated with information about changes since the last release.
2. **Update Library Version**: The version information is updated in configure.ac (three places), and CMakeLists.txt. These changes are committed to the master branch.
3. **Update Shared Library Versions**: The libtool shared library version-info number is updated as indicated by the libtool documentation (http://www.gnu.org/software/libtool/manual/libtool.html#Libtool-versioning), for both the C and the Fortran library. The shared library versions can be found in the Makefile.am files in src/clib and src/flib. Each library (C and Fortran) has its own shared library version-info number. These changes are committed to the master branch.

4. **Make Distribution**: The updated master branch is checked out, and configure and make test are run, with the --enable-fortran option. After this completes, the make dist target is run to produce the distribution tarball.
5. **Upload Tarball**: The tarball is uploaded into the binaries section of the release on GitHub.
6. **Tag and Release**: GitHub is used to tag a release, with a tag like "pio_2_4_1". The new version is released on GitHub in the "releases" section.
7. **Update Documentation**: The master branch is checked out, and configured with the --enable-docs option.(Doxygen is required). The documentation is built by doing a make in the doc directory. This produces the HTML version of the documentation in the (confusingly named) docs directory. Another clone of the library is checked out and the gh-pages branch checked out. A git rm -rf * is done on the gh-pages branch to delete all files. Then the docs directory is copied there and a git add * adds the files to the gh-pages branch. This is then committed and GitHub will display it as project documentation. Note that it can take some time after pushing the changes before the web page with the documentation gets updated. (See also https://stackoverflow.com/questions/11577147/how-to-fix-http-404-on-github-pages).

   The commands to do this:
   ```
       git clone git@github.com:NCAR/ParallelIO.git p1
       cd p1/
       git checkout gh-pages
       git rm -rf *
       cp -R ../ParallelIO/docs/* .
       git add *
       git commit -m "updated docs" -a && git push
   ```
8. **Announce to Users:** The release notes are mailed to the PIO and netCDF mailing lists to notify users of a new release.
9. **Change Version for Developers:** Add "-development" to the version numbers in step 2, so that, after the release, subsequent developer builds have a version of (for example) "2.4.2-development". This indicated a version after 2.4.1, but not yet 2.4.2.
10. **Start Release Notes for Next Release:** On the GitHub releases page, draft release notes for the next release. This gives developers a place to list their changes as they work, preparing for the next release.

9 REFERENCES

A. Allain. Why Bother with Compiler Warnings. http://www.cprogramming.com/tutorial/compiler_warnings.html, accessed: 2019-06-27.

Earl T. Barr Christian Bird Peter C. Rigby Abram Hindle Daniel M. German Premkumar Devanbu, Cohesive and Isolated Development with Branches, Proceedings of the International Conference on Fundamental Approaches to Software Engineering, April 2012, Published by Springer. Retrieved on May 23, 2019 from http://earlbarr.com/publications/gitbranches.pdf

Kent Beck. 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

K. Beck, Test Driven Development: by example, 1st Edition, Addison-Wesley, Boston, 2002.

W Bissi, AGSS Neto, MCFP Emer, The effects of test driven development on internal quality, external quality and productivity: A systematic review, Information and Software Technology, 2016 - Elsevier, Retrieved on May 26, 2019 from https://www.researchgate.net/profile/Adolfo_Neto/publication/295863661_The_Effects_of_Test_Driven_Development_on_Internal_Quality_External_Quality_and_Productivity_A_systematic_review/links/57974ca208aec89db7b99c66/The-Effects-of-Test-Driven-Development-on-Internal-Quality-External-Quality-and-Productivity-A-systematic-review.pdf

Y. Bugayenko, "Continuous integration is dead," http://www.yegor256.com/2014/10/08/continuous-integration-is-dead.html, 2014, accessed: 2016-10-8.

Doxygen, http://www.doxygen.nl/

Fowler, M., "Continuous integration," http://martinfowler.com/articles/originalContinuousIntegration.html, 2000, accessed: 2016-10-8.

GitHub Pages Help, https://help.github.com/categories/github-pages-basics/

Hatton L. (1997) The T-experiments: errors in scientific software. In: Boisvert R.F. (eds) Quality of Numerical Software. IFIP Advances in Information and Communication Technology. Springer, Boston, MA.

M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig. 2016. Usage, costs, and benefits of continuous integration in open-source projects. In 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). 426–437. Retrieved on May 23, 2019 from http://cope.eecs.oregonstate.edu/papers/OpenSourceCIUsage.pdf.

Jenkins. 2017. Jenkins. https://jenkins.io/. (2017). Accessed 2019-05-21.

McIntosh, S., Kamei, Y., Adams, B. et al., An empirical study of the impact of modern code review practices on software quality, Empir Software Eng (2016) 21: 2146. https://doi.org/10.1007/s10664-015-9381-9 Retrieved on May 30, 2019 from https://www.researchgate.net/profile/Ahmed_E_Hassan/publication/276162927_An_empirical_study_of_the_impact_of_modern_code_review_practices_on_software_quality/links/578e893308ae81b4466ec98e/An-empirical-study-of-the-impact-of-modern-code-review-practices-on-software-quality.pdf.

MPE GitHub Site, https://github.com/pmodels/mpe

PIO GitHub Site, https://github.com/NCAR/ParallelIO

PIO CDash Continuous Integration Site, https://my.cdash.org/index.php?project=PIO.

Rahman, Akond & Agrawal, Amritanshu & Krishna, Rahul & Sobran, Alexander & Menzies, Tim. (2017). Continuous Integration: The Silver Bullet? Retrieved on May 23, 2019 from https://www.researchgate.net/publication/321025247_Continuous_Integration_The_Silver_Bullet.

Siniaalto, M., & Abrahamsson, P. (2007). A comparative case study on the impact of test-driven development on program design and test coverage. In Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on (pp. 275-284). IEEE. Retrieved on May 23, 2019 from https://www.researchgate.net/publication/4279048_A_Comparative_Case_Study_on_the_Impact_of_Test-Driven_Development_on_Program_Design_and_Test_Coverage

S. Stolberg, “Enabling agile testing through continuous integration,” in Agile Conference (AGILE). IEEE, 2009, pp. 369–374. Retrieved on May 23, 2019 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.848.5149&rep=rep1&type=pdf.

Chengnian Sun, Vu Le, Zhendong Su, “Finding and analyzing compiler warning defects”, ICSE '16 Proceedings of the 38th International Conference on Software Engineering, Austin, Texas — May 14 - 22, 2016 Accessed: 2019-06-27 from https://web.cs.ucdavis.edu/~su/publications/icse16-warning.pdf.

Travis CI. https://travis-ci.org/. Accessed: 2019-05-21.

B. Vasilescu, Y. Yu, H. Wang, P. T. Devanbu, and V. Filkov, “Quality and productivity outcomes relating to continuous integration in GitHub,” in Joint Meeting on Foundations of Software Engineering (ESEC/FSE). ACM, 2015, pp. 805–816. Retrieved on May 23, 2019 from https://bvasiles.github.io/papers/fse15ci.pdf.

Y. Yu, G. Yin, T. Wang, C. Yang, and H. Wang, “Determinants of pullbased development in the context of continuous integration,” Science China Information Sciences, vol. 59, no. 8, pp. 1–14, 2016. Retrieved on May 23, 2019 from http://yuyue.github.io/res/paper/pr-scis.pdf.

Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 2017. The Impact of Continuous Integration on Other Software Development Practices: A Large-Scale Empirical Study. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE 2017). ACM, New York, NY, USA. Retrieved on May 23, 2019 from https://cmustrudel.github.io/papers/ase17ci.pdf.