Supporting Provenance in Climate Science Research

Brett Yasutake, Niko Simonson, Jason Woodring, Nathan Duncan, William Pfeffer,
Hazeline U. Asuncion, Munehiro Fukuda, Eric Salathe
School of Science, Technology, Engineering, and Mathematics
University of Washington Bothell
Bothell, WA, USA
{yasutake, nikouw, jman5000, njd91, wpfeffer, hazeline, mfukuda, salathe}@u.washington.edu

Abstract—**While the data produced by climate models exponentially grows in size and complexity, the ability of researchers to analyze available data lags. Existing tools for climate analysis that capture provenance are generally implemented on supercomputing clusters. Provenance is often difficult for a researcher to analyze due to its sheer volume. In contrast, our Pacific Northwest Climate Analysis (PNCA) Tracker is a lightweight, provenance-aware parallel system that allows researchers from smaller facilities to quickly develop custom analysis tools while enabling them to easily verify their datasets. This technique modularizes the captured provenance, allows researchers to customize the provenance collection, and efficiently collects provenance within a parallel and distributed environment, made possible by the use of the Multi-Agent Spatial Simulation (MASS) library. It is designed to be highly extensible by minimizing dependencies within the architecture. We demonstrate that our tool is potentially accessible to a wider range of researchers and is highly efficient compared to the commonly used climate analysis tool, Network Common Data Form (NetCDF) Operators or NCO. Finally, we discuss how provenance concepts in PNCA Tracker map to the W3C PROV.**

Keywords-data provenance; climate science; parallelization; big data.

I. INTRODUCTION

Regional and global climate models produce a vast array of model output data simulated for periods of years to centuries. The complexity and resolution of climate models is steadily increasing, incorporating more complex and realistic methods. Analysis of regional climate data can yield important discoveries that have wide-ranging impacts; however, climate data have grown to a scale that makes them difficult to analyze without sophisticated computational tools [13]. On top of this, small to medium-sized climate research labs are resource-constrained. These labs have access to limited computational infrastructure and their computational resources are generally focused on the analysis of data or running simulations, not on administering or maintaining specific technologies. Thus, these research labs must be able to quickly build one-off analytical tools (e.g., scripts or homegrown software), run them efficiently on large datasets, and capture useful, understandable provenance to support various analysis tasks. Data provenance is defined as the "origin or the history of data" [18]. Without data provenance, it is difficult to assess the results of a simulation.

Meanwhile, current provenance tools for climate research generally cater to labs equipped with high-end computing resources. As a result, current provenance support in this domain assumes access to High Performance Computing (HPC) resources [20] or to specific platforms (e.g., scientific workflows [11] or databases [7]). Without access to these resources or staff to operate these platforms, these climate researchers lack tools to capture provenance.

To bridge this gap, we created a novel technique called Pacific Northwest Climate Analysis (PNCA) Tracker, a platform-independent provenance tool which allows climate researchers to easily integrate provenance capture with their own analysis tools in a lightweight manner. PNCA Tracker leverages the easy-to-use Multi-Agent Spatial Simulation (MASS) library for parallelization [8] to enable researchers to rapidly develop analytical tools that can be immediately applied to a large body of climate model output data efficiently. PNCA Tracker provides simple provenance collection mechanisms for the overall framework, which runs and parallelizes researcher-implemented modules for the analysis of climate data.

The contributions of this paper are as follows: (1) an accessible, adaptable, and scalable technique to support data provenance in climate research, (2) a loosely-coupled and customizable tool framework that enables researchers to "plug-in" their custom analysis tool and off-the-shelf visualization tools, (3) a set of evaluations that suggest the viability of our technique, and (4) a discussion of provenance concepts in relation to the community standard W3C PROV.

This paper is organized as follows. The next section compares our technique to existing provenance collection techniques. It also provides background information on existing techniques used with PNCA Tracker. Section III provides a motivation for our technique. Section IV contains details about PNCA Tracker and its tool support is described in Section V. Section VI covers our set of evaluations. Section VII discusses how our provenance concepts map to the W3C PROV. The paper concludes with future work.

II. RELATED WORK AND BACKGROUND

This section discusses closely related techniques and background on techniques used with the PNCA Tracker.

A. Provenance Techniques for Climate Research

Many provenance collection techniques for climate science rely on access to HPC resources. For example, one

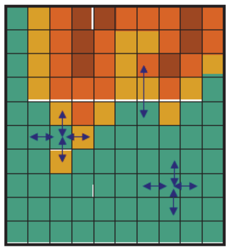


Figure 1. Visual representation of the MASS simulation space.

technique uses a high-end supercomputing cluster [20]. Milieu is a lightweight, unobtrusive provenance collection tool, but is currently implemented on the National Energy Research Scientific Computing Center's cluster [7]. Our technique does not require the availability of these types of resources.

Some techniques require the availability of specific technologies. Systems such as Milieu store data in specific databases that require native support [7]. Ultrascale Visualization Climate Data Analysis Tools (UV-CDAT) outputs application-specific VisTrails workflows [11]. These two approaches allow for very granular and highly indexed provenance that increases the value of the provenance data collected. Another technique provides provenance support to a middleware technology [15]. However, our technique does not require specific technologies to access the stored provenance, as we discuss in the next section.

Many of the software tools that support the analysis of large-scale data require some computer science background. For example, Hadoop is a well-known software package for analyzing large-scale data. HadoopProv, which supports provenance in Hadoop, also requires users to configure and monitor computing nodes in a cluster [6].

Meanwhile, SciHadoop was developed to support the array structure of data in a parallel environment [12]. Parallel NetCDF [14] and Message Passing Interface (MPI) [1] also provide parallel processing. However, these programming environments do not have built-in provenance support and, thus, must be user-customized with processor-aware code [5], which again requires non-computer scientists to write machine-aware code. Our technique can include such a processor for distributing the data analysis tasks. We chose the MASS library because it is more accessible to inter-disciplinary researchers (see Section II.C for details).

B. Background on NetCDF

The internationally accepted data format of climate science data is the Network Common Data Form (NetCDF), which is a highly efficient, array based file [3]. NetCDF itself includes portable libraries for a number of programming languages and packages for scripting languages [14]. These include C, Fortran, Java, Matlab, Python, and R. While these libraries aid in the development of tools to process data in this format, among the provenance-aware tools that are available, such as NetCDF Operators (NCO) [19], provenance collection is difficult to configure and the lack of a common output format makes the available provenance difficult to use.

C. Background on MASS Library

The MASS library addresses the semantic gap between analyzing algorithms and their actual implementation [10]. The MASS library abstracts away many of the complex and technical details of parallel programming, such as inter-process communication and shared resource management [8]. For the programmer, one simply specifies the number of computing nodes and resources to use and MASS manages the intricate parallel programming details under-the-hood.

MASS relies on inter-node and inter-thread communication, but abstracts such communication from the logical design of algorithms. In other words, if a programmer has a 100 x 200 x 47 simulation space that is running across three processors and four threads per processor, the programmer does not need to know how to divide that simulation space between the processors or threads. Barrier synchronization allows parallelization to advance in lock-step removing the need to manually handle concurrency issues.

Key to MASS's abstraction is the concept of Place and Places. A Places object is an instantiated, multi-dimensional simulation space (see Figure 1). The entire picture can be thought of as a single Places object that holds a multi-dimensional grid of Place objects. Each individual square formed by the grid within the Places object is a single Place object. A Place object can have distinct values, depicted by different colors and can communicate with other Place objects, shown by the blue arrows. Place objects are logical locations within the simulation space which can host mobile agents, or execute code themselves. In our technique, each Place maintains the entire Pacific Northwest climate data in a given time segment such as 6am, 12pm, 6pm, or 0am on a given date in a given month.

MASS does not depend on a shared memory paradigm, which is used by fork-join frameworks. Instead it uses remote computing nodes, with each node maintaining an independent memory space. When a user program calls MASS.Init( ), this function contacts ssh daemon processes, each running on a different remote process, establishes a secured TCP link, and asks each daemon process to launch a MASS-unique slave program called MProcess. Each remote MProcess manages different stripes of distributed arrays, (i.e., Places). When the user program calls MASS.Finish( ), the function kills the remote MProcesses.

III. Motivations and Requirements for Provenance Support in Climate Science

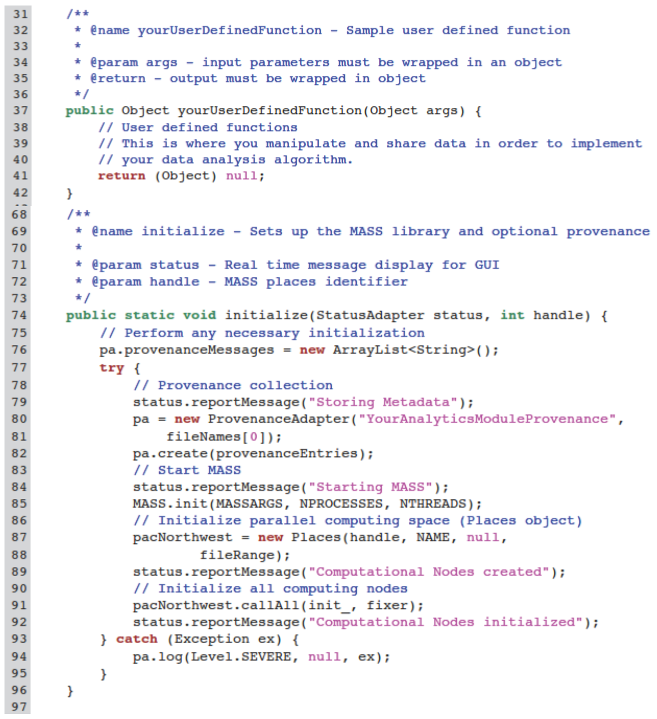
Climate researchers spend much time and effort towards peripheral, non-climate science aspects of their analyses such as file handling and manipulation. The terabyte scale of climate model data poses immediate practical issues for the individual researcher or members of small facilities [13]. Since climate researchers generally do not have extensive computer science background, they prefer to use familiar tools to perform their analyses. Thus, they are less likely to adopt tools that require much training time. On top of this, data analysis for climate researchers is highly exploratory. They usually create scripts or code for a one time analysis.

Thus, they will not spend time to create detailed workflows since these may not be used again in the future. Finally, there is a great need for data provenance to determine the validity of analyses. Insufficient provenance gathering creates difficulties in understanding the meaning of the results, especially when there is not enough model data preserved to easily visualize the results. In order to provide provenance support for these researchers, we posit that provenance techniques and tools must satisfy the following properties: accessibility, adaptability, and scalability.

Accessibility is a property that allows as many researchers as possible the ability to capture provenance and use it for their tasks. Accessibility implies simplicity, both in the storage and retrieval of provenance. Accessibility also implies low barrier to entry, requiring minimal computing resources and minimal tool training time.

Adaptability is a property that allows researchers to capture provenance regardless of changes in their methods or processes. The ability to accommodate changes is necessary in a highly exploratory nature of data analysis in climate research. Not only do methods or processes change, but the computing environment(s) on which researchers run their analyses are also likely to change.

Scalability is a property that is often associated with the ability to accommodate large datasets or large number of computing nodes. While these are important items to consider, we also refer to the scalability of the provenance capture (in terms of performance overhead) and understandability of the captured provenance.



```
31      /**
32       * @name yourUserDefinedFunction - Sample user defined function
33       *
34       * @param args - input parameters must be wrapped in an object
35       * @return - output must be wrapped in object
36       */
37      public Object yourUserDefinedFunction(Object args) {
38          // User defined functions
39          // This is where you manipulate and share data in order to implement
40          // your data analysis algorithm.
41          return (Object) null;
42      }
68      /**
69       * @name initialize - Sets up the MASS library and optional provenance
70       *
71       * @param status - Real time message display for GUI
72       * @param handle - MASS places identifier
73       */
74      public static void initialize(StatusAdapter status, int handle) {
75          // Perform any necessary initialization
76          pa.provenanceMessages = new ArrayList<String>();
77          try {
78              // Provenance collection
79              status.reportMessage("Storing Metadata");
80              pa = new ProvenanceAdapter("YourAnalyticsModuleProvenance",
81                  fileNames[0]);
82              pa.create(provenanceEntries);
83              // Start MASS
84              status.reportMessage("Starting MASS");
85              MASS.init(MASSARGS, NPROCESSES, NTHREADS);
86              // Initialize parallel computing space (Places object)
87              pacNorthwest = new Places(handle, NAME, null,
88                      fileRange);
89              status.reportMessage("Computational Nodes created");
90              // Initialize all computing nodes
91              pacNorthwest.callAll(init_, fixer);
92              status.reportMessage("Computational Nodes initialized");
93          } catch (Exception ex) {
94              pa.log(Level.SEVERE, null, ex);
95          }
96      }
97
```

Figure 2. A researcher implements a climate analysis calculation in the yourUserDefinedFunction method (line 37). The initialize method initializes the ProvenanceAdapter.

IV. PACIFIC NORTHWEST CLIMATE ANALYSIS TRACKER

We now discuss how PNCA Tracker achieves these required properties.

A. *Achieving Accessibility through Simple Interfaces and Modular Provenance Files*

We provide mechanisms for easy development of parallelizable analysis modules that are integrated with provenance collection and for the usage of modular provenance files which are stored in commonly used file formats.

The development of analysis modules is straightforward and caters to researchers who are familiar with scripts (e.g., Python) or high level programming languages (e.g., Java). A researcher simply needs to implement a few required functions (read and write methods) and user defined functions (in Figure 2, it is the yourUserDefinedFunction method), and to place this function in a specified location. Doing so will automatically trigger the collection of a basic set of provenance information (e.g., geographic metadata) as discussed in the next section.

In addition, we created modular provenance files to assist researchers in quickly retrieving the appropriate provenance: result-specific provenance, execution-specific provenance, and error logs. We also store these files in commonly used file formats to enable researchers to easily share them with collaborators. All the provenance files are stored in the same directory. Researchers may choose to use a file-naming convention to connect the result-specific provenance with the execution-specific provenance and error logs.

Result-specific provenance contains information regarding the data's parent files and the processing used (e.g., steps within the analytical module). It also contains the climate model used, the date of data generation, characteristics of the output data itself, date range of the arrays, and geographic metadata (i.e., information to locate the results over a map of the Earth to allow the results to be renderable by any geographic information system (GIS) viewers). Result-specific provenance is embedded within the results file within a NetCDF metadata. By co-locating the data with its provenance, we minimize the chances of the provenance being lost or inaccessible.

Execution-specific provenance, meanwhile, contains provenance related to the MASS execution state. We store execution-specific provenance in a separate NetCDF file because a single analysis can produce hundreds of NetCDF files. This scheme avoids replicating the same provenance across all the results files, and thereby minimizes the storage requirement. Embedding provenance in NetCDF files was chosen to support easy query and retrieval [9].

Finally, we store in a text file format errors encountered during an analysis execution (i.e., error logs). These errors may be associated with the MASS environment or they may be associated with the analysis modules developed by the researchers. Similar to execution-specific provenance, the error logs are stored separately. By using a non-tool-specific file format, .txt file, the errors are viewable by any member of the research team.

B. Achieving Adaptability through Provenance Customization

To provide adaptability, we use different levels of provenance customization: no customization, semi-customization, and full customization. The capabilities of each level are folded into the next level.

1) No customization

The first level, no customization, requires no human intervention. As long as the analysis module is integrated within our provenance tool support, basic provenance collection automatically occurs. This includes exceptions and geographic metadata.

If an exception is tied to an operation involving a MASS Place, the Place is also identified. In addition to the exception messages, we also include the logical coordinates where the exception occurred. For example, if the data for climate data simulation analysis file at noon on July 18th of 1974 created an exception in the analysis, we include these temporal coordinates in our error log.

Provenance is also captured through the geographic metadata. The climate data contains a large amount of metadata stored in the header file. When a file is identified to be read, the metadata is automatically extracted. When a results file is created, this information is used to create the metadata used for the header of the results file.

Of the metadata available from the climate simulation files, some are broadly applicable for every results file. This would include all of the GIS data: results cover the same geographic range as the source data. Metadata is duplicated for every results file. Other metadata cannot be copied exactly because there are multiple entities involved, such as time-based data. When the results file of a time-series analysis is created, several entities contribute to the data, each one with a distinct time slice. Appropriately transforming the metadata to reflect this is also an activity of our provenance collection at the level of no customization.

2) Semi-customized provenance collection

The second level, semi-customized provenance collection, requires some input from researchers. For example, researchers may choose to collect detailed information regarding the MASS execution state. State information can assist users in understanding the extent of parallelization employed (e.g., scalability of their analysis modules) and the source of errors. Execution state is particularly important because debugging software in a distributed system can be difficult. State information includes number of Place objects generated, number of processes, threads per process, the different processors employed, total execution time, and the analysis module used. Some of the provenance data are taken directly from the argument parameters input to the MASS engine. Others are collected from information given to the instantiation of the Places space, such as the number of logical computational nodes.

3) Fully customized provenance collection

The third level, fully-customized provenance collection, requires more specification from researchers but provides richer metadata that can encapsulate contextual information about an analysis (e.g., hypothesis being tested). Within the user-defined analysis module, users may embed provenance information related to each analysis step or method within the module, similar to adding comments in a code where a tag or a note related to the algorithm is recorded. This provenance information is stored with the execution-specific provenance. The captured provenance is dependent on the writer of the analysis class.

Creating a fully-customized provenance collection is fairly straightforward. Users simply create a new instance of the Provenance Adapter object (see line 80, Figure 2). Once it is created, they may add provenance logs to record steps executed in the analysis.

C. Achieving Scalability through Coupling Provenance with Analysis Logic

When the operation scales to include dozens, hundreds, or even thousands of computing nodes, it is still important to collect discernible provenance. The results-specific provenance is strictly tied to the logic of the analysis. The captured execution-specific provenance is also scalable because we get an overview of the entire computation cluster and not by individual node. This is because the MASS library abstracts the physical computation from the logical model. Therefore, if we run the same analysis, going up from two nodes to two thousand nodes, we expect, other than the necessary communication overhead, a commensurate increase in efficiency without any degradation of the understandability of the provenance being collected.

V. PNCA TRACKER TOOL SUPPORT

The PNCA Tracker tool support is implemented within the PNCA Framework to facilitate parallelized analysis of climate model data. The tool is implemented in Java, to enable running on different operating systems. This section covers more details about the tool.

A. Tool Design

The overarching architecture employed is one of adapters, based on the adapter design pattern (see Figure 3). We used this pattern to loosely-couple the major modules within our system to support extensibility. Each major module of the system is treated as an "adapter" and their

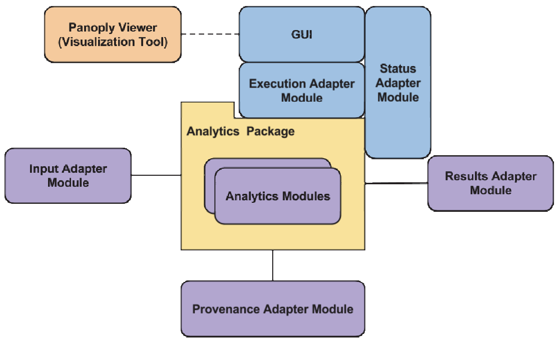


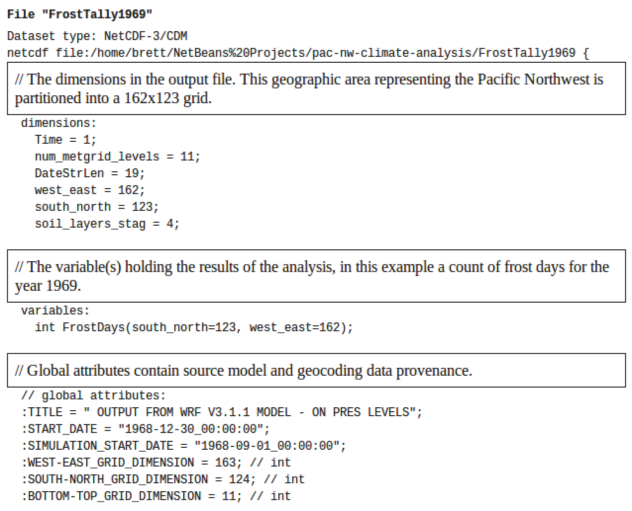
Figure 3. Design diagram of the PNCA Tracker tool support. Modules connected with lines are loosely coupled.

main purpose is to provide "dependency sinks." Changes in how each adapter operates should only result in internal changes to the adapter without affecting their interfaces to the rest of the framework.

We developed an integrated system with a graphical user interface (GUI) and several modules that can be applied to read data, record results, and supply important provenance information. These input and output tasks are able to be performed in parallel on multiple threads and processes through the MASS library, which resides within the Execution Adapter Module.

The Analytics Package contains the analysis modules developed by researchers and function as "plug-ins" to the tool framework. Users may write an unlimited number of functions to carry out an analysis. The user defined functions are performed in parallel and may contain customized instructions on provenance collection (see Figure 2). The researcher implements methods to read, write, and analyze climate simulation data (e.g., yourUserDefinedFunction() shows the minimal format of such a method). The initialize() method (line 74 in Figure 2) is called by the Execution Adapter module to allow the analysis to be parallelized by MASS.

The GUI, Status Adapter, and Execution Adapter modules are used to execute the analysis packages. The GUI allows for the selection of climate data files and specific analysis modules. The analysis modules are automatically added to the dropdown when the users save the files in the proper location. The Configure Input command allows selection of source files. The GUI also allows researchers to view the results of the analysis directly in a viewer. We currently integrated our tool with the Panoply Viewer. The Status adapter transmits execution-time status messages from the analysis module to the GUI. Most important is the Execution Adapter, which controls the parallel execution of the analysis module across multiple processes and threads using the MASS engine. The Execution Adapter is also responsible for capturing execution-specific provenance.



```
File "FrostTally1969"
Dataset type: NetCDF-3/CDM
netcdf file:/home/brett/NetBeans%20Projects/pac-nw-climate-analysis/FrostTally1969 {
```

// The dimensions in the output file. This geographic area representing the Pacific Northwest is partitioned into a 162x123 grid.

```
  dimensions:
    Time = 1;
    num_metgrid_levels = 11;
    DateStrLen = 19;
    west_east = 162;
    south_north = 123;
    soil_layers_stag = 4;
```

// The variable(s) holding the results of the analysis, in this example a count of frost days for the year 1969.

```
  variables:
    int FrostDays(south_north=123, west_east=162);
```

// Global attributes contain source model and geocoding data provenance.

```
  // global attributes:
  :TITLE = " OUTPUT FROM WRF V3.1.1 MODEL - ON PRES LEVELS";
  :START_DATE = "1968-12-30_00:00:00";
  :SIMULATION_START_DATE = "1968-09-01_00:00:00";
  :WEST-EAST_GRID_DIMENSION = 163; // int
  :SOUTH-NORTH_GRID_DIMENSION = 124; // int
  :BOTTOM-TOP_GRID_DIMENSION = 11; // int
```

Figure. 4. A partial list of metadata copied from the original source files and applied to the resulting output files.

B. *Tool Implementation using the MASS Library*

We use the MASS library for the backbone of execution of the custom analysis modules developed by climate researchers. We use two major methods provided by the library: ExchangeAll and CallAll. ExchangeAll reveals information, providing for communication between Place locations within a Places space, and between multiple Places spaces. CallAll performs the work, running code, allowing up to all Place spaces to perform calculations. Places have an accessible, machine-unaware indexing structure that allow specific subsets of Place objects to be identified, meaning that groups of Place locations can execute code while others remain quiescent.

When researchers create their own analysis modules, these modules extend the Place object, so its methods can be run across large quantities of data in parallel. By using specific Place objects identified from their index values, data can be quickly reduced. For example, in an analysis module that calculates the number of days when the temperature dropped below freezing for each geographic cell of climate output, the multiple input files would be reduced to a single results file of the time-series analysis.

C. *Usage Scenarios*

We built our tool support with the following usage scenarios. Researchers capture provenance from an exploratory analysis so that they can retrace their steps if their analysis yields useful results. Analysis developers create a reusable set of analysis tools that research students can later run and record provenance. After analysis execution, analysis developers fine-tune their algorithms based on the MASS state information. Prior to publication of results, developers consult the results file(s) (which contain result-specific provenance), the execution-specific provenance, and error log to check their results.

VI. EVALUATION

We evaluated PNCA Tracker using provenance queries it can answer, a case study on local climate data analysis, and timing results. We also discuss limitations of our technique.

A. *Provenance Queries*

1) What input climate data files were analyzed to produce FileXYZ.nc?

This can be answered by referring to the global attributes within FileXYZ.nc. As shown in Figure 4, the three global attributes: TITLE, START_DATE, and SIMULATION_START_DATE identify the specific parent file(s) and time slice. Figure 4 also shows that the resulting output files include the necessary geographical information to allow results files to be visualized with GIS viewers. This provenance information is collected at the no customization level.

2) What analysis steps were taken to produce FileXYZ.nc?

Detailed processing steps within the analysis module, if specified by the analysis developer, are captured and saved with the execution-specific provenance.

B. *Local Climate Research Case Study*

We asked a climate impacts researcher to evaluate PNCA Tracker's usability. Using the tool, s/he was able to evaluate data from the Pacific Northwest derived from the Weather Research and Forecasting (WRF) Model.

WRF is a state-of-the-art mesoscale numerical weather prediction system designed to serve both operational forecasting and atmospheric research needs [4]. This model has been developed and used extensively in recent years for regional climate simulation [17]. WRF has been implemented as a regional climate model over the Northwest United States at 12 km grid spacing [16].

The dataset covered climate models from 1980 spanning 6 hour intervals. Each time slice was around 56 megabytes in size. The geographic space simulated consists of a 123 by 162 grid, where each cell is approximately 15 kilometers on a side. The space is extended into the third dimension through a number of atmospheric layers above the ground, and soil layers below the ground. Each cell contains 150 variables of measurement, such as temperature and pressure.

The study was run with semi-customized provenance collection, which provides metadata from the climate data examined as well as clustering information from the analysis. The provenance information enabled the researcher to verify the climate models' data.

Concerning accessibility, it took the user less than 30 minutes to understand the tool and to analyze some sample data with a predetermined calculation for Simple Precipitation Index. S/he also stated that the time spent recording provenance and analyzing the provenance was highly acceptable. S/he also commented that the tool "is fast. The parallel computing is great." Since the user was unfamiliar with Java, s/he was unable to use the full provenance customization feature of the tool.

C. *Timing Results*

With regards to scalability, we compared the performance overhead imposed by our technique with NCO (see Figure 5). The timing analysis was run on a Linux Mint 15 (Olivia) x64 with Intel Core CPU T5600 @ 1.83GHz and 2 gigabytes (GB) of random access memory (RAM). Our tests involving minimal parallelization (two threads on a single processor) nevertheless produced substantial improvements when compared to an unparallelized version of the same algorithm and running the algorithm using NCO in provenance-aware mode. A moisture flux calculation using PNCA Tracker on the dataset used in the case study took 15 seconds of real time to accomplish, including the provenance capture, while running the calculation using only one variable pair (PNCA Tracker used two) with NCO took 156 seconds of processing time, and over 15 minutes of real time to complete. The PNCA Tracker was executed at the level of "full customization," the most comprehensive provenance generation in our tool. The improvement can be attributed to the fact that PNCA Tracker first loads all the data to memory so that the calculations are highly efficient. Furthermore, the scalability that PNCA Tracker achieved due to the MASS engine's ability to distribute as well as parallelize the analysis is far superior to NCO.

We also compared the overhead imposed by customizing provenance (see Figure 6). The dataset used for the timing evaluation was an entire year of WRF climate models from 1970 (10.27 GB total file size). The timing analysis was run on a Red Hat Enterprise 5.11 (Tikanga) with Intel Xeon E5520 @ 2.27 GHz and 6GB of RAM. Three sets of 5 runs each (total of 15 runs) were performed, with the different order of runs for each set. The average runtime of a no customization run was 8.922 seconds, semi-customization was 8.910 seconds, and full customization was 9.241 seconds. This indicates that fully customizing provenance imposes minimal additional overhead compared to no customization.

D. *Limitations*

There are substantial overhead requirements involved with the use of the PNCA Tracker. Each Place object is a complex data structure that takes up large amounts of memory. With a commodity laptop and 2GB of RAM, the maximum size for an analysis is about three months' worth of six-hour time slices. However, the point of the inclusion of MASS is to allow these calculations to be performed over a computer cluster, allowing for immediate scalability.

The MASS library abstracts away the parallel and distributed nature of the program execution from the developer. One significant drawback is that we are not collecting provenance on the location of execution (i.e., processing node) for a subset of Place objects. It is possible to collect this provenance information and we are currently examining how this can be efficiently achieved with minimal overhead.

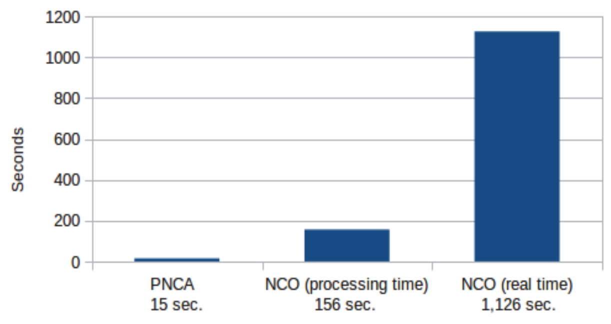


Figure. 5. PNCA Tracker shows minimal performance overhead with provenance collection compared to the provenance-aware mode of NCO.

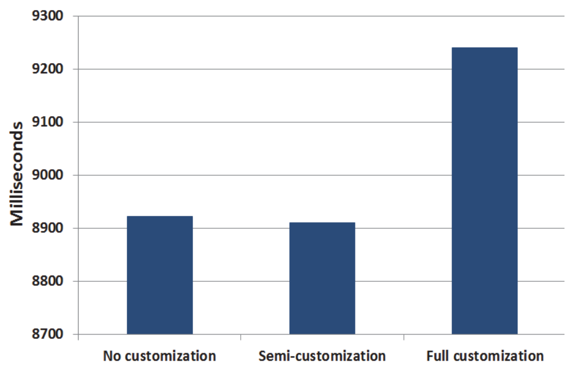


Figure. 6. PNCA Tracker shows a performance overhead of less than half a second for fully customizing provenance.

VII. Mapping Provenance Concepts to W3C PROV

W3C PROV is a community standard for representing provenance [2]. The three main concepts in W3C PROV, entities, activities, and agent, have direct mappings to concepts in PNCA Tracker, as shown in Table 1.

The entities in PNCA Tracker are the input climate data and the results files (output), both in NetCDF format. Although our area of focus is the Pacific Northwest region of the United States of America, southern Canada, and the northeastern Pacific Ocean, the PNCA Tracker can read and apply its analysis package to any properly-formatted climate data. This is due to the universal agreed-upon format for climate data in terms of variable names and data types.

The activities can be mapped to the steps within the analysis algorithms and the interactions with the tool's user interface (e.g., selection of analysis module, selection of input files).

The agent can be mapped to multiple concepts in PNCA Tracker. Researchers who create the analysis modules can be considered an agent. The MASS Environment, Execution Adapter Module, external climate data simulation programs, and result viewers can also be considered agents.

Analysis modules are unique in that they can be mapped to these three concepts depending on time and perspective. During and after an analysis module is written, it is considered an entity that an activity (writing) produced which is associated with an agent (a researcher). During a module's execution, it itself can be considered an agent by which the output files are associated. After a module's execution, the individual steps within the analysis module can be considered activities that generated output analysis files.

VIII. Conclusion and Future Work

The PNCA Tracker is an improvement over the existing state of the art for climate researchers who wish to collect provenance but have limited access to, or training with, HPC or specific platforms or technologies. PNCA Tracker is accessible to researchers with minimal computer science background, adaptable to different types of climate analysis and different operating systems, and scalable to large computing clusters. Our technique stores captured provenance in a modular fashion, to minimize access time and to minimize the storage requirements. Our technique also allows researchers to customize the provenance collection. Our set of evaluations (provenance queries, climate research case study, and timing results) indicates that PNCA Tracker is a feasible option to provenance tracking in a parallel and distributed environment. Finally, provenance concepts in PNCA Tracker can also be mapped to concepts in the W3C PROV.

A future goal is to use agent analysis. As the name implies, MASS supports the deployment of large quantities of mobile agents into its environment. Two immediate uses for such agents would be an improvement to the basic time detection algorithm and meteorological phenomena detection, such as locating atmospheric rivers or storms. We also plan to export provenance into the W3C format.

TABLE I. Provenance Mapping between W3C PROV and PNCA Tracker.

W3C PROV	PNCA Tracker
Entities	Climate data in NetCDF format (input) Results file(s) in NetCDF format (output) Analysis modules
Activities	Analysis module steps Interactions with the PNCA Tracker GUI
Agent	Analysis module developers Climate data simulation programs Execution Adapter Module MASS Environment Viewers (e.g., Panoply) Analysis module (execution)

Acknowledgment

The authors would like to thank the following for their active assistance and participation: Dr. William Erdly for academic project guidance, Delmar Davis for provenance collection advice, the MASS Project Team for advice and guidance concerning the MASS Library, and the student cohort of the UWB Master's Project from Summer and Autumn 2013 for peer reviews and advice. This work is based in part upon work supported by the US National Science Foundation under Grant No. ACI 1350724. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] Message passing interface (MPI) standard. http://www.mcs.anl.gov/research/projects/mpi. retrieved: Jan. 2015.

[2] PROV-Overview. http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/. retrieved: Jan. 2015.

[3] Status of standards body endorsements of NetCDF and related conventions. http://www.unidata.ucar.edu/software/netcdf/docs/standards.html. retrieved: Jan. 2015.

[4] The weather research & forecasting mode. http://www.wrf-model.org/index.php. retrieved: Jan. 2015.

[5] Message Passing Interface Forum. MPI-2, chapter Extension to the Message-Passing Interface, Chapter 9, I/O. University of Tennessee, 1997.

[6] S. Akoush, R. Sohan, and A. Hopper, "HadoopProv: towards provenance as a first class citizen in MapReduce," Workshop on the Theory and Practice of Provenance, Apr. 2013, pp. 11:1–11:4.

[7] Y.-W. Cheah, R. Canon, B. Plale, and L. Ramakrishnan, "Milieu: Lightweight and configurable big data provenance for science," Proc of Int'l Congress on Big Data, Jun. - Jul. 2013, pp. 46–53.

[8] T. Chuang and M. Fukuda, "A parallel multi-agent spatial simulation environment for cluster systems," Proc of Int'l Conf on Comp Science and Eng, Dec. 2013, pp. 143–150.

[9] D. B. Davis, H. U. Asuncion, G. M. Abdulla, and C. W. Carr, "Towards recovering provenance with Experiment Explorer," Proc of Fifth Int'l Conf on Info, Process, and Knowledge Management, Feb.-Mar. 2013, pp. 104–110.

[10] J. Emau, T. Chuang, and M. Fukuda, "A multi-process library for multi-agent and spatial simulation," Proc of Pacific Rim Conf on Communications, Computers and Signal Processing, Aug. 2011, pp. 369–375.

[11] E. Santos et al., “Designing a provenance-based climate data analysis application,” Int’l Provenance and Annotation Workshop, Jun. 2012, pp. 214–219.

[12] J. B. Buck et al., “SciHadoop: Array-based query processing in Hadoop,” Proc of Int’l Conf for High Performance Computing, Networking, Storage and Analysis, Nov. 2011, pp. 66:1–66:11.

[13] J. Gray et al., “Scientific data management in the coming decade,” SIGMOD Record, vol. 34, no. 4, Dec. 2005, pp. 34–41.

[14] J. Li et al., “Parallel NetCDF: a high-performance scientific I/O interface,” Proc. of Supercomputing, Nov. 2003, page 39.

[15] D. Goodman, “Provenance in dynamically adjusted and partitioned workflows,” Proc of the Int’l Conf on eScience, Dec. 2008, pp. 39–46.

[16] E. P. Salathe Jr, L. R. Leung, Y. Qian, and Y. Zhang, “Regional climate model projections for the State of Washington,” Climatic Change, vol. 102, no. 1-2, May 2010, pp. 51–75.

[17] L. R. Leung, Y.-H. Kuo, and J. Tribbia, “Research needs and directions of regional climate modeling using WRF and CCSM,” Bulletin of the American Meteorological Society, vol. 87, no. 12, Dec. 2006, pp. 1747–1751.

[18] R. Stevens, J. Zhao, and C. Goble, “Using provenance to manage knowledge of in silico experiments,” Briefings in Bioinformatics, vol. 8, no. 3, May 2007, pp. 183–194.

[19] C. Zender, H. Butowsky, and W. Wang, Welcome to the netCDF Operators (NCO) homepage. http://nco.sourceforge.net/. retrieved: Jan. 2015.

[20] D. Zhao, C. Shou, T. Malik, and I. Raicu, “Distributed data provenance for large-scale data-intensive computing,” Proc of Int’l Conf on Cluster Comp, Sep. 2013, pp. 1–8.