Scientific Research Publishing

Journal of Computer and Communications, 2017, 5, 76-85
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Data Analysis Framework for Earth System Simulation within an *In-Situ* Infrastructure

D. Wang[1*], X. Luo[2], F. Yuan[1], N. Podhorszki[3]

[1]Environmental Science Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[2]Department of Computer Science and Electric Engineering, University of Tennessee, Knoxville, TN, USA
[3]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Email: *wangd@ornl.gov

How to cite this paper: Wang, D., Luo, X., Yuan, F. and Podhorszki, N. (2017) A Data Analysis Framework for Earth System Simulation within an *In-Situ* Infrastructure. *Journal of Computer and Communications*, **5**, 76-85.
https://doi.org/10.4236/jcc.2017.514007

Received: October 20, 2017
Accepted: December 26, 2017
Published: December 29, 2017

Abstract

This paper presents a generic procedure to implement a scalable and high performance data analysis framework for large-scale scientific simulation within an *in-situ* infrastructure. It demonstrates a unique capability for global Earth system simulations using advanced computing technologies (*i.e.*, automated code analysis and instrumentation), *in-situ* infrastructure (*i.e.*, ADIOS) and big data analysis engines (*i.e.*, SciKit-learn). This paper also includes a useful case that analyzes a globe Earth System simulations with the integration of scalable *in-situ* infrastructure and advanced data processing package. The *in-situ* data analysis framework can provides new insights on scientific discoveries in multiscale modeling paradigms.

Keywords

In-Situ Data Analysis, Source Code Analysis, Data Staging, ADIOS, Earth System Model, Machine Learning, SciKit-Learn, E3SM

1. Introduction

Earth system models (ESMs) are essential approaches to understand Earth system dynamics and to project future climate scenarios. It is well known that the validation and verification of Earth system process within EMSs are quite challenging [1] [2]. Earth system scientists widely adopted post-simulation approaches to analyze results, ranging from visual exploration to transitional statistical data analysis. Along with the advances in *in-situ* infrastructure development [3] [4], high performance computing and artificial intelligence [5], real-time data analysis becomes an innovative approach to investigate Earth system simulation results [6] [7] [8].

Developing an *in-situ* data analysis platform for Earth System modeling requires several practical solutions, including 1) automated code instrumentation of large-scale code to extract appropriate data of interest; 2) efficient packing and transfer of highly customized data types; 3) optimal tuning of underlying *in-situ* infrastructure based on the unique characteristics of the applications; as well as 4) seamless integration with external data processing and machine learning packages.

In this paper, we first present design considerations and key components of a data analysis framework for Earth system simulation. We then list the computing and software environment used in our study. At last, a case study is designed to demonstrate the practical usefulness of the data analysis system and its computing performance.

2. Data Analysis Procedure and Key Components

The data analysis framework developed in our study consists of four major components: 1) source code analysis and data capture; 2) data packing and conversion tools; 3) *in-situ* infrastructure; and 4) data analysis packages. The work-flow of the framework is illustrated in Figure 1. The main functions and objectives of each component are listed in the following sections.

2.1. Source Code Analysis and Data Capture

In this step, we analyze the source code dependency using the information extracted from compilers or language parsers. The main goal is to understand the software structure and to capture internal data structure and scientific workflow of the source code. For a given function or module of interest, we use programming language parsers to analyze the source code and store the program information as an abstract syntax tree (AST). Then, we conduct recursive name resolution through the AST to capture the input and output data streams of the target function. Finally, we generate a code segment into the original source code to

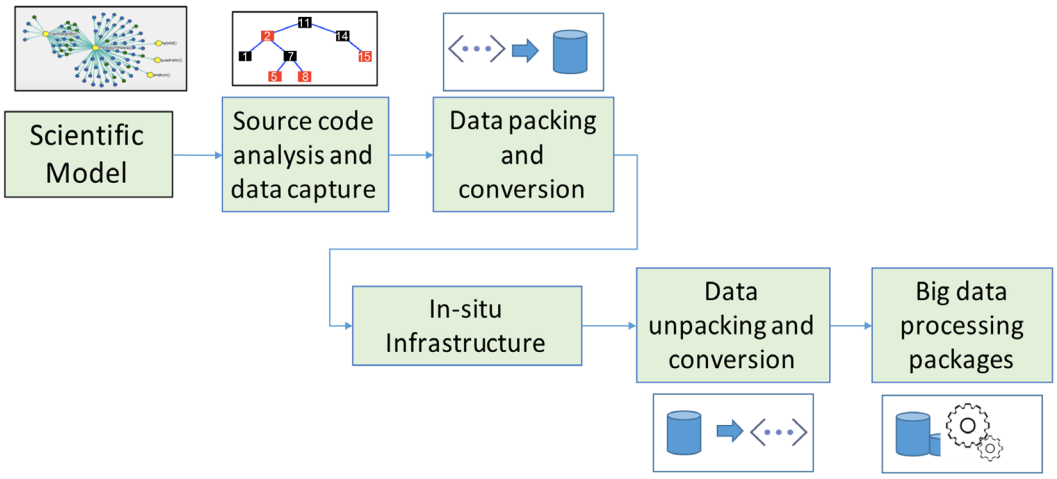

Figure 1. A procedure of *in-situ* data analysis.

package all the data of interest into a continuous memory buffer ready for *in-situ* data transfer. More detailed information on the source code analysis and data capture can be found in previous publications [9] [10].

2.2. Data Packing and Conversion Tool

As most scientific codes, Earth system models use highly customized data structures. As mentioned in the previous section, the data is packed into a continuous memory block for efficient data transfer using the underlying *in-situ* infrastructure. The internal information on the customized data structure has to be recorded as well, to reconstruct the customized data types after data transferring. It is also worth to mention that most *in-situ* infrastructures use specific data formations to facilitate its performance, therefore, it is convenient to create a software tool for the data conversions between applications and *in-situ* infrastructures.

2.3. Integration with *In-Situ* Infrastructure

By now, several *in-situ* infrastructure have emerged, including data analysis and visualization toolkits (such as ParaView [11] and VisIt [12]) and high performance data infrastructure (such as ADIOS and GLEAN). These infrastructures enable full capability of *in-situ* data processing at the benefit of large I/O cost savings and better utilization of all available resources. In our study we leverage the capabilities with the adaptive IO system developed at ORNL.

2.4. External Packages for Automated Data Analysis

There are several comprehensive packages widely used for big data analysis, including graph database and machine learning [13] [14] [15]. In this study, we use a free, python-based machine learning library (Scikit-learn) for automated data processing. Scikit-learn features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the other Python libraries.

3. Case Study: An *In-Situ* Data Analysis System for the E3SM Land Model

Energy Exascale Earth System Model (E3SM) is a national effort to address the challenging and demanding climate-change research imperatives. Within the E3SM modeling framework, E3SM Land Model (ELM) is a process-based model that represents the energy-water-biogeochemistry interactions between the atmosphere and the terrestrial landscape. We implement an *in-situ* data analysis system for the ELM and focus on key biogeophysical and biogeochemical functions.

Due to the complexity of ELM, the validation and verification of the terrestrial system process are quite challenging. Scientists routinely use post-simulation approaches to analyze results. Generating data for post-simulation earth system process investigation quickly becomes a cumbersome task once a simulation

reaches a fairly large scale with a huge amount of data and daunting input/output cost. A previous pilot effort focused on demonstrating a concept and small-scale (pointwise) prototype [8]. Our current effort is to process global simulations with the integration of scalable *in-situ* infrastructure and advanced data processing package.

3.1. Computing Platform

The platform used in this study is a Linux cluster within the Computing and Data Environment for Science (CADES) at Oak Ridge National Laboratory. The cluster has 48 nodes of Cray CS400 machines. Each node contains 2 Intel E5-2698v3 16-core (total of 32 per node), 128 GB RAM, Dual port mellanox-FDR IB and 10GbE and 250 GB local hard drive. The cluster shares a Petascale parallel file system with other clusters within CADES. External users can access these ORNL HPC clusters via two dedicated login nodes.

3.2. Software System Implementation

The ELM used in the study comes from the newest version of E3SM code in 2017. In the past, we have developed two methods for parsing and analyzing the ELM code based on KGEN and PGI Fortran compiler [9]. In this study, we further enhanced our code parsing and analysis capability using KGEN to better handle the global, nested and customized ELM data structure. Since KGEN is written in Python, we developed our own python script to extract information from KGEN and generate FORTRAN code segments into the ELM source code for data packing. The underlying *in-situ* infrastructure used in our study is the Adaptive IO system (ADIOS) [3]. We use ADIOS with the ELM code as an external module. We developed a data writer that uses two different *in-situ* mechanisms available in ADIOS, FlexPath and Data Spaces, depending on the characteristics of specific ELM variables. On the data analysis side, we developed a data reader using the ADIOS C API and then converted it into a shared library with SWIG. This way we can access this staged reading function through other applications, such as python-based data analysis packages. In our study, Scikit-learn is used to handle the data streams from ELM simulations. Technically, we used the ADIOS reader in C to retrieve the data streams from ADIOS infrastructure, and converted the data into python arrays which can be integrated with Scikit-learn functions. We can also dump the data into disk using the native ADIOS data format and then connect to other applications, such as Visit, for further applications. The procedure and high-level implementation of our system is illustrated in **Figure 2**.

4. Analysis of Total Leaf Area Index within Global Terrestrial Ecosystem Simulations

4.1. Simulation System Configuration

In this study, we configure the global ELM simulation on a 0.5 × 0.5 degree grid.

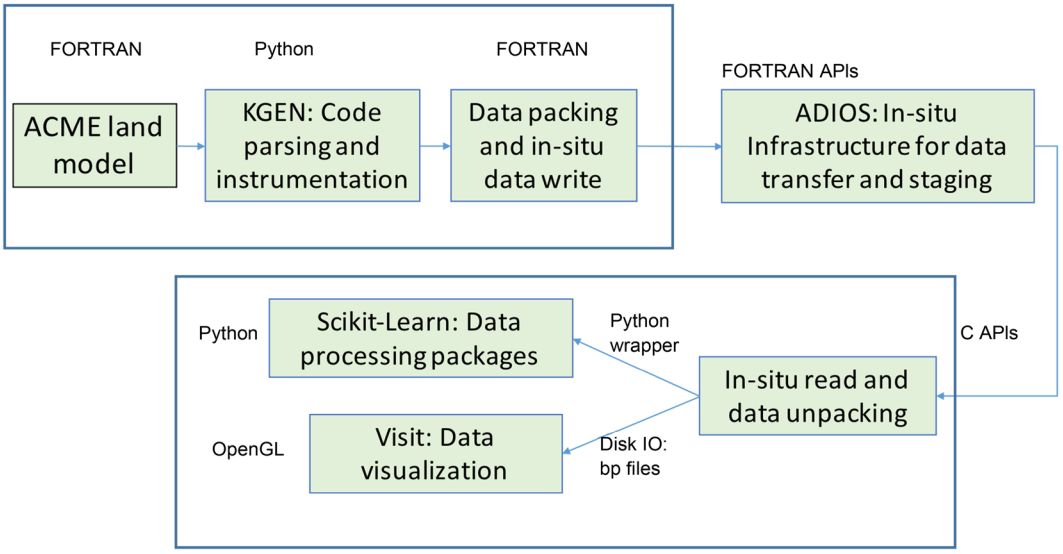

Figure 2. Software system implementation of the *in-situ* data analysis for earth system simulation.

The simulation starts at the spinup stage, driven by the climate datasets of 1920 - 1948, developed by the Climatic Research Unit in the United Kingdom. It also runs assuming a constant CO_2 and land use in 1850. For the purpose of testing our approach, the spinup only runs for 320 years, long enough to demonstrate how plants evolve in warm regions such as the tropics, without *in-situ* Data Analysis Tools plugged in. Then the simulation restarts from the end of the spinup run and continues for a transit run (1850 to 2010). The transit run is configured to simulate the historic Earth system behavior since the industrial revolution. It is one of commonly used simulation configurations for future scenario projections.

4.2. Total Leaf Area within Terrestrial Ecosystem Modeling

For the demonstration, we capture and analyze the values of Total Leaf Area Index (TLAI) through the half year global terrestrial ecosystem simulations and calculate the data characteristics (such as statistics and primary components) of TLAI during the real-time simulations using built-in function from Sci-Kit-Learn. Leaf area index (LAI), *i.e.* projected one-side foliage area over ground surface, is a dimensionless quantity that characterizes the vegetation foliage size and thus its function in the Earth system model to predict photosynthetic primary production, evapotranspiration, and it can be regarded as an indicator for plant growth or greenness. As such, LAI plays an essential role in theoretical production ecology. In ELM, LAI is simulated for total 17 individual vegetation types in a grid cell. Then, with the consideration of the actual vegetation coverage of each grid cell, these individual LAIs are fraction weighted to calculate total LAI (TLAI) at each grid cell on the land section of the Earth.

4.3. A Snap Shot of TLAI Map at the Beginning of Simulation

Figure 3 contains a snapshot of simulation result on the Total Leaf Area (TLAI)

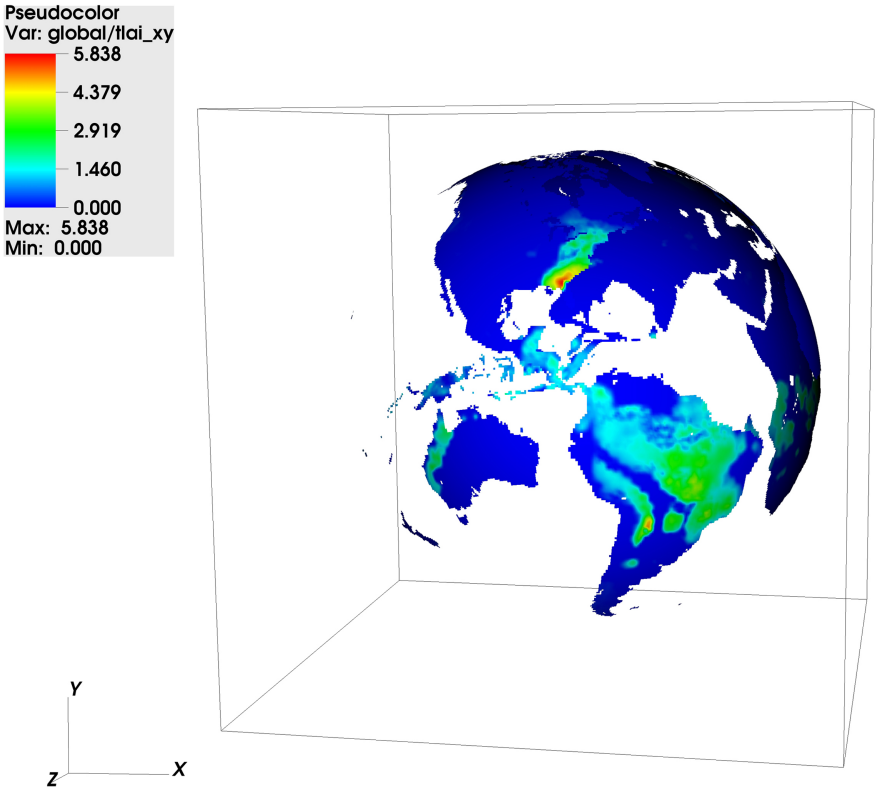

Figure 3. A snapshot of simulation result: the map of TLAI values at the beginning of global simulations.

at the very beginning of simulation (12 am GWT, January 1, 1850). The TLAI at each grid around the globe (total 360 × 720 cells) is calculated as a weighted summarization of LAI on each vegetation type (called patch) times the percentage of vegetation area within each grid cell. Obviously, TLAI values of tropical cells, such as Amazon and other low altitude, well vegetated areas, are much higher than TLAI at other places. It is also noticeable from the TLAI map that the beginning simulation is a winter time at north hemisphere, since the TLAI is low at these middle-latitude areas in the north hemisphere.

4.4. Automated Global TLAI Clustering

This test is designed to demonstrate the automated data processing using a machine learning package, SciKit-learn. After the data of interest is extracted and transferred from the simulation, transferred data arrays are transformed into numpy arrays that can be accessed by a Python application. In our test, we use the built-in statistical functions of Scikit-learn to classify TLAI values during the simulation, so that the seasonal global vegetation greenness pattern can be recognized dynamically. **Figure 4** shows the automated clustering of TLAI values (into 9 group) at a single half-hour time-step during the simulation. The clustering was carried out by using k-means in Scikit-learn.

The high-resolution snapshot of raw model data output, like the one in **Figure 3**, usually shows too much detailed information to recognize its patterns or

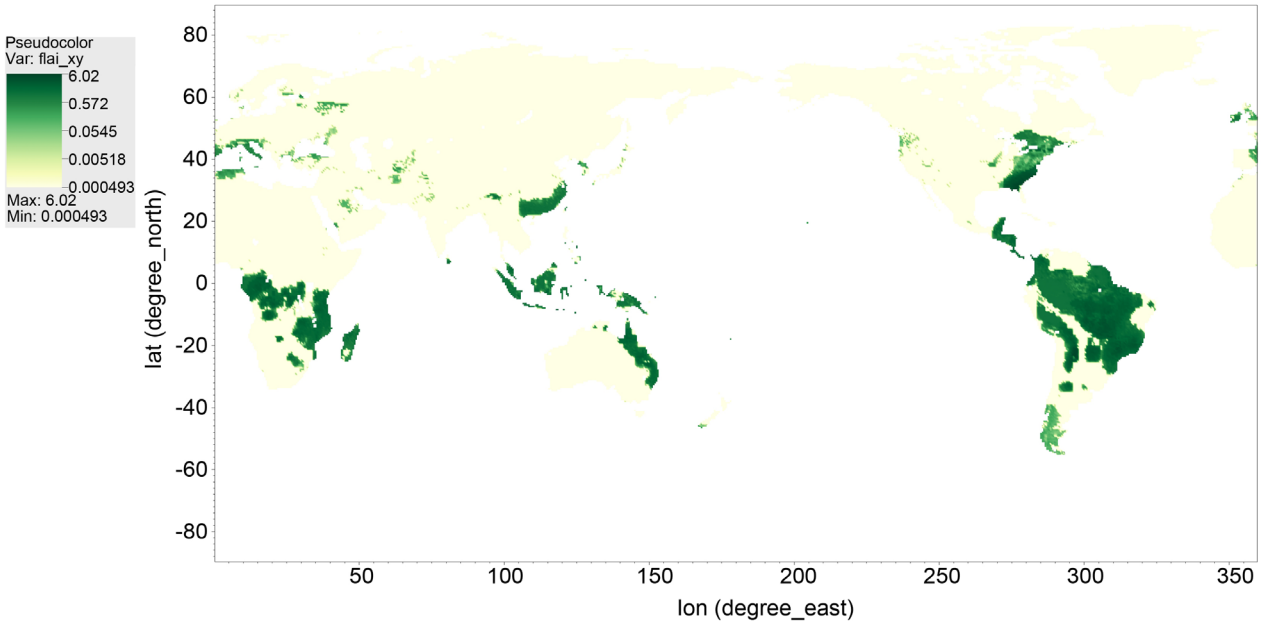

Figure 4. The spatial pattern of automated global classification of high-resolution TLAI at a single half-hour time-step during simulation. The classes are categorized by mean plus or minus fractional SD.

changes. Post data analysis, such as classification in **Figure 4**, allows for more clear spatial patterns (with less noise). This is very beneficial for modelers to understand and interpret results.

4.5. Performance Analysis

Two sets of experiments have been conducted to understand simulation performance. We first record the walltime used for data collection, packing and transfer. The pie chart of time division is illustrated in the left graph of **Figure 5**. We do not directly compare the original E3SM configuration with filesytem operations since it is just too slow to be useful in practice. The walltime of E3SM shown in **Figure 5** is used for computation only. Data transfer via ADIOS takes about 7.4%, while the data packing and convert via KGEN takes more than 25% of time. We will need to further investigate and improve the performance of data packing. In this study, we also investigate how to achieve best performance with different process placement strategies. The time of running E3SM on different configurations is illustrated in the right-side chart of **Figure 5**. The performance of the data transfer can be optimized by mapping each pair of ADIOS reader and writer on the same compute node, shown as ADIOS_SAMESOCKET, instead of on different nodes, shown as ADIOS_DIFFNODE. ADIOS_BP shows the time to write the data back into disk. Since we can save a lot of disk IO time using the *in-situ* infrastructure, both ADIOS_SAMESOCKET and ADIOS_DIFFNODE are much faster than ADIOS_BP.

5. Conclusion and Suggestions

We have presented design considerations of a data analysis framework for Earth

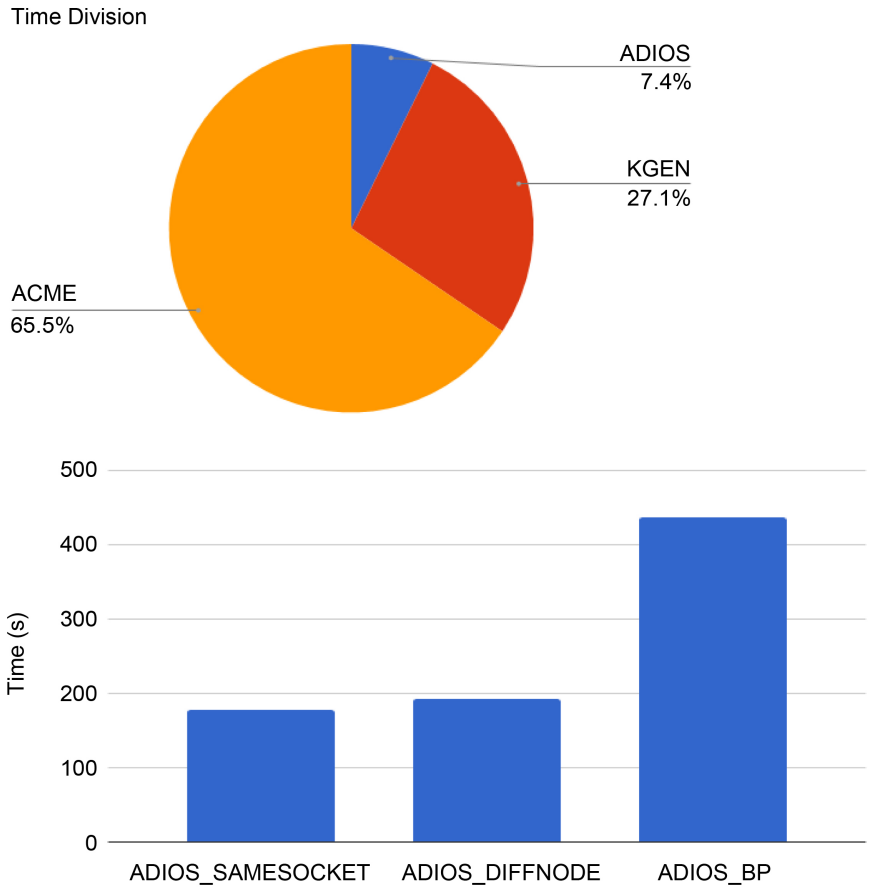

Figure 5. Performance analysis.

system simulation based on automated source code instrumentation, *in-situ* infrastructure and real-time data processing. We have designed a case study of Earth system simulation to demonstrate the practical usefulness of the data analysis system and its computing performance. With the integration of external data processing package, such as Scikit-Learn, SPACK and TensorFlow, we can easily apply novel machine learning approaches to study simulation results in on the fly. We believe the *in-situ* processing is a feasible way to investigate large scale climate simulations without intensive human interaction and it avoids the prohibitive IO cost of post-processing on high performance computing platforms. Future efforts will have two directions. We will focus on tuning the performance on high end computers, such as Titan and Summit-Dev at the National Center for Computational Science at Oak Ridge National Laboratory. We will also work to integrate our data analysis system with other external packages, such as TensorFlow, for further large-scale Ecosystem simulation data analysis on hybrid architectures. Science efforts will focus on relationship identifications between the extreme weather events and long term ecosystem behaviors.

Acknowledgements

This research was funded by the U.S. Department of Energy, Office of Science,

Biological and Environmental Research program (E3SM and TES). This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Wang, D., Schuchart, J., Janjusic, T., Winkler, F., Xu, Y. and Kartsaklis, C. (2014) Toward Better Understanding of the Community Land Model within the Earth System Modeling Framework. *Procedia Computer Science*, **29**, 1515-1524. https://doi.org/10.1016/j.procs.2014.05.137

[2] Wang, D., Xu, Y., Thornton, P., King, A., Steed, C., Gu, L. and Schuchart, J. (2014) A Functional Test Platform for the Community Land Model. *Environmental Modelling & Software*, **55**, 25-31. https://doi.org/10.1016/j.envsoft.2014.01.015

[3] Liu, Q., Logan, J., Tian, Y., Abbasi, H., Podhorszki, N., Choi, J.Y., Klasky, S., Tchoua, R., Lofstead, J., Oldfield, R. and Parashar, M. (2014) Hello Adios: The Challenges and Lessons Of Developing Leadership Class I/O Frameworks. *Concurrency and Computation: Practice and Experience*, **26**, 1453-1473. https://doi.org/10.1002/cpe.3125

[4] Vishwanath, V., Hereld, M. and Papka, M.E. (2011) Toward Simulation-Time Data Analysis and I/O Acceleration on Leadership-Class Systems. 2011 *IEEE Symposium on Large Data Analysis and Visualization* (*LDAV*), Providence, Rl, 23-24 October 2011, 9-14. https://doi.org/10.1109/LDAV.2011.6092178

[5] Russell, S.J. and Norvig, P. (2003) Artificial Intelligence: A Modern Approach. 111-114.

[6] Kress, J., Klasky, S., Podhorszki, N., Choi, J., Childs, H. and Pugmire, D. (2015) Loosely Coupled *In Situ* Visualization: A Perspective on Why It's Here to Stay. *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, Austin, TX, 15-20 November 2015, 1-6. https://doi.org/10.1145/2828612.2828623

[7] Malakar, P., Vishwanath, V., Munson, T., Knight, C., Hereld, M., Leyffer, S. and Papka, M.E. (2015) Optimal Scheduling of *In-Situ* Analysis for Large-Scale Scientific Simulations. 2015 *SC-International Conference for High Performance Computing, Networking, Storage and Analysis*, Austin, TX, 15-20 November 2015, 1-11. https://doi.org/10.1145/2807591.2807656

[8] Wang, D., Yuan, F., Hernandez, B., Pei, Y., Yao, C. and Steed, C. (2017) Virtual Observation System for Earth System Model: An Application to ACME Land Model Simulations. *International Journal of Advanced Computer Science and Applications*, **8**, 171-175. https://doi.org/10.14569/IJACSA.2017.080223

[9] Wang, D., Pei, Y., Hernandez, O., Wu, W., Yao, Z., Kim, Y., Wolfe, M. and Kitchen, R. (2017) Compiler Technologies for Understanding Legacy Scientific Code: A Case Study on an ACME Land Module. *Procedia Computer Science*, **108**, 2418-2422. https://doi.org/10.1016/j.procs.2017.05.264

[10] Wang, D., Wu, W., Janjusic, T., Xu, Y., Iversen, C., Thornton, P. and Krassovisk, M. (2015) Scientific Functional Testing Platform for Environmental Models: An Application to Community Land Model. 37*th International Workshop on Software Engineering for High Performance Computing in Science*.

[11] Ahrens, J., Geveci, B. and Law, C. (2005) Paraview: An End-User Tool for Large Data Visualization. The Visualization Handbook, 717-731.

https://doi.org/10.1016/B978-012387582-2/50038-1

[12] VISIT Team (2003) VISIT: Software the Delivers Parallel Interactive Visualization. Lawrence Livermore National Laboratory. http://visit.llnl.gov

[13] Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J. and Ghodsi, A. (2016) Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM*, **59**, 56-65. https://doi.org/10.1145/2934664

[14] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M. (2016) TensorFlow: A System for Large-Scale Machine Learning. *OSDI*, **16**, 265-283.

[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. (2011) Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830.