Virtual Observation System for Earth System Model: An Application to ACME Land Model Simulations

Dali Wang, Fengming Yuan
Climate Change Science Institute
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

Yu Pei, Cindy Yao
Department of Electric Engineering
and Computer Science
University of Tennessee, Knoxville, TN 37996, USA

Benjamin Hernandez
National Center for Computational Science
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

Chad Steed
Computational Science Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA

***Abstract*—Investigating and evaluating physical-chemical-biological processes within an Earth system model (EMS) can be very challenging due to the complexity of both model design and software implementation. A virtual observation system (VOS) is presented to enable interactive observation of these processes during system simulation. Based on advance computing technologies, such as compiler-based software analysis, automatic code instrumentation, and high-performance data transport, the VOS provides run-time observation capability, in-situ data analytics for Earth system model simulation, model behavior adjustment opportunities through simulation steering. A VOS for a terrestrial land model simulation within the Accelerated Climate Modeling for Energy model is also presented to demonstrate the implementation details and system innovations.**

Keywords*—*Earth System Modeling; Accelerated Climate Modeling for Energy; In-Situ Data Analytics; Virtual Observation System; Functional Unit Testing

I. INTRODUCTION

Over the past several decades, several Earth system models (ESMs) have been developed to understand Earth system dynamics and to project future climate scenarios. Among these ESMs, the Accelerated Climate Modeling for Energy (ACME) model, funded by the US Department of Energy (DOE), is a national effort to address the challenging and demanding climate-change research imperatives. Due to the complexity of EMSs in both model design and software implementation, the validation and verification of the Earth system process with EMSs are quite challenging, especially at the scales and levels of organization wherein many relevant field measurements and experiments are made (Wang et. al., 2014a). Scientists routinely use post-simulation approaches to analyze results. These include visual exploration to detect anomalies or interesting patterns and statistical data analysis for further investigation. 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. For these reasons, an interactive, run-time simulation monitoring system, or a virtual observation system (VOS), is needed. In this paper, author first scribe key functions of a VOS and then describe its major components based on advanced computing technologies (such as compiler-based software analysis, automatic code instrumentation, and high-performance data transport). At last, for the demonstration purpose, authors present implementation details on a VOS for a terrestrial land model that is the ACME Land Model (ALM) which is a process-based model with a collection of key bio geophysical and biogeochemical functions that represent the energy-water-biogeochemical interactions between the atmosphere and the terrestrial landscape. The VOS software system for ALM provides the capabilities of real-time observation and in-situ data analytics for model simulation.

II. VIRTUAL OBSERVATION SYSTEM DESIGN

Key functions of A VOS are 1) to setup a “watch point” for a specific physical-chemical-biological function and 2) to capture the input and output data streams of a target function. Therefore, users can quantify the relationship between input and output data streams of a target function and identify variables that are can be observed at desired sampling frequencies. This information can be used to guide data collections in real world observation systems. A VOS also provides interactive tracking capability over user-selected key model variables throughout model simulation, so that users can “observe” changes in model variable values, and explore the relationship among Earth system functions (related to these user-selected model variables) over a specific spatial-temporal domain.

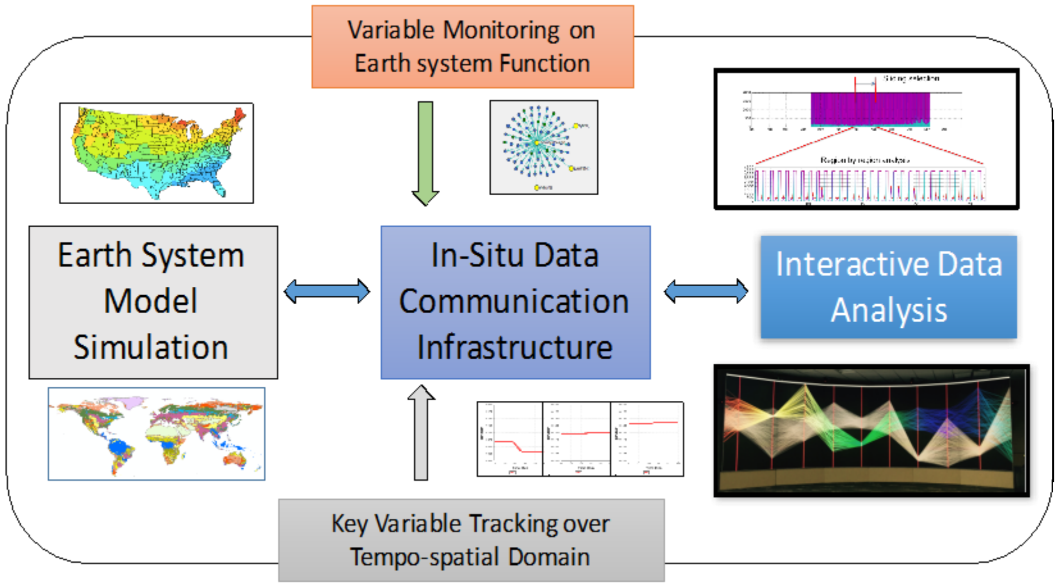
Variable Monitoring on Earth system Function
Earth System Model Simulation
In-Situ Data Communication Infrastructure
Interactive Data Analysis
Key Variable Tracking over Tempo-spatial Domain

Fig. 1. Major software components of a VOS, including software analysis and code instrumentation, in-situ data infrastructure, and interactive data analysis. Two typical uses of the VOS are function-specific data monitoring and variable tracking throughout the simulation

Figure 1 shows two typical uses of a VOS. First, the VOS allows users to define a specific function (an individual subroutine or a group of related subroutines) and an observation period, then the VOS collects input and output data streams of the target function and transports these data out of the simulation system for visualization and analysis. Second, the VOS helps to track specific key model variables throughout the simulation system over a user-defined period. Figure 1 also illustrates the major components of a VOS, including software analysis and code instrumentation, in-situ data communication infrastructure and interactive data analysis.

A. *Software analysis and automated instrumentation*

The main purpose of this VOS component is to collect information on software structures and workflow. Authors adopted a similar workflow procedure used in a scientific function test platform (Wang et. al., 2015, 2014b; Yao et. al., 2016). The procedure has several steps: First, authors use software dependency analysis to identify methods to reduce software dependency on parallel computing and external libraries. This step simplifies the model software dependency by using production compilers without an optimization option. Next, authors perform a compiler-assisted workflow analysis to capture the internal data structure and scientific workflow of the simulation source code. For a given function or module, authors use a programming language parser to analyzes the source code, break it into tokens, and store the program internally as an abstract syntax tree (AST). Then, authors conduct recursive name resolution through the AST to capture the input and output data streams of a target function in the simulation source code. Finally, authors instrument code segments into the source code to pack all the data of interest into a continuous memory buffer ready for in-situ data infrastructure. Since the majority of EMSs are developed in Fortran, authors are working on the integration of a kernel extraction tool (Kim, et. al., 2016), which is built on top of a Python Fortran parser, for automatic code instrumentation. The process is shown in Figure 2.

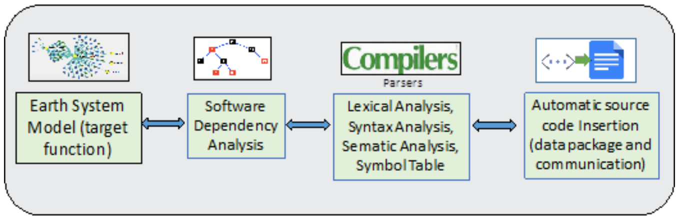
Earth System Model (target function)
Software Dependency Analysis
Compilers
Parsers
Lexical Analysis, Syntax Analysis, Sematic Analysis, Symbol Table
Automatic source code Insertion (data package and communication)

Fig. 2. General procedure for software analysis and automated instrumentation within a VOS

B. *In-situ data communication infrastructure*

The main function of this VOS component is to provide high-performance data communication capability for transferring the data of interest out of simulation system for external analysis. The data infrastructure allows users to inspect variable values in real time during model simulation. In the current effort, VOS in-situ data infrastructure is built on the Common Communication Interface (CCI) (Atchley et al, 2011). The CCI project is an open-source communication interface that aims to provide a simple and portable Application Programming Interface (API), high performance and scalability for the largest deployments, and robustness in the presence of faults. The in-situ data infrastructure consists of three segments: data generation, data staging, and data analysis (Figure 3). In the VOS, the data analysis segment first creates CCI channels to which the data generation segment (instrumented simulation code) can connect. Once the connection is established, users can then pass simulation parameters (function and variable names, time interval, and location, etc.) to instrumented simulation code. Once the simulation runs to the user-defined time interval, the instrumented simulation code packs all the relevant data into a buffer and uses CCI's Remote Memory Access (RMA) methods to send the data over the network to the data analysis segment. The data analysis segment always listens on its own CCI channel. When the data arrives, the analysis segment unpacks the data for follow-up data processing and analysis.

Considering that large data volume needs to be transferred into data analysis, VOS data infrastructure also includes a data staging area that allows data caching for input/output operations and low-latency data queries. The data staging area also allows users to define functions and observation periods and track key model variables over simulation period. The main purposes of data staging are: 1) reduce potential data overload in the analysis side during model simulations and 2) then enable user-based queries and maintain interactive rates. Currently, the staging area is co-located with data analysis and visualization, and acts as a temporal storage area for data processing operations (e.g., storing, loading, extraction, transformation, or querying). Figure 3 shows the VOS in-situ data infrastructure with a staging area.

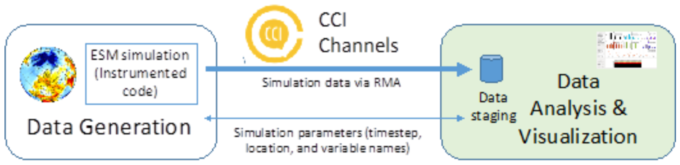
ESM simulation (Instrumented code)
Data Generation
CCI Channels
Simulation data via RMA
Simulation parameters (timestep, location, and variable names)
Data staging
Data Analysis & Visualization

Fig. 3. In-situ data infrastructure with a data staging area inside the data analysis component

C. Interactive data analysis

This VOS component provides a front end with which users can perform three main tasks: 1) choose the ecosystem functions and time interval for monitoring, 2) interactively visualize the results of predefined "watch" points throughout simulation, and 3) steer the simulation accordingly, if necessary. The data analysis component also directly communicates with the staging area to conduct query submission and data retrieval based on the user interactions.

From the technical perspective, this component contains three modules: 1) a graphic user interface (GUI) that allows users to perform these three main tasks, 2) an interactive data visualization engine that plots physical-chemical-biological interactions produced by the simulation, and 3) a communication interface with a staging area which in turn connects to the instrumented simulation code.

In the study, the GUI is built using Qt and the data visualization engine is developed using the Visualization Toolkit (VTK), which utilizes the underlying graphical processing unit (GPU) for faster rendering. Multicore CPU processors are used to handle data transfer. After receiving the buffer from CCI, the engine converts the data into vtkTable data structure for visualization. The buffering mechanism based on data staging allows users to select time steps for visualization. The visualization engine employs a client-server model, so that while the VTK server is located alongside the simulation for faster data transfer, the actual client display windows can be on any remote machine. This feature greatly increases the portability and usability of the system.

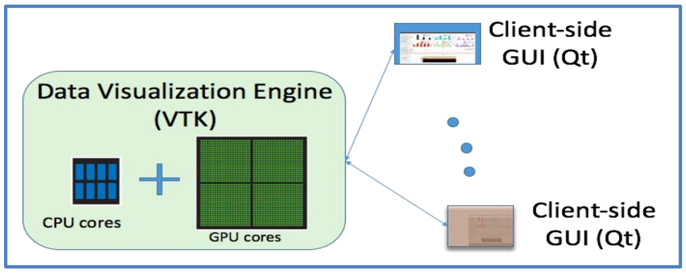

Fig. 4. Key components of VOS data visualization, which utilizes hybrid hardware and provides cross-platform GUIs

III. VOS FOR ALM: CASE DEMONSTRATION

In this section, authors demonstrate a VOS for the ALM simulation over the Next Generation Ecosystem Experiments Arctic site (NGEE-Arctic, http://ngee-arctic.ornl.gov), located at the Barrow Ecosystem Observatory (BEO) in Barrow, Alaska. In this experiment, ALM was configured as a point-mode offline simulation to investigate terrestrial ecosystem responses to specific atmospheric forcing over a single landscape grid cell at Barrow (Yuan, et. al., in preparation). For the demonstration purposes, the observation system is used to track all the variables in and out of a CNAllocation module within ALM. The CNAllocation function is developed to allocate key chemical elements (such as carbon, nitrogen and phosphorus) of a plant in a terrestrial ecosystem.

The software architecture diagram of the VOS for ALM using the CNAllocation module is illustrated in Figure 5.

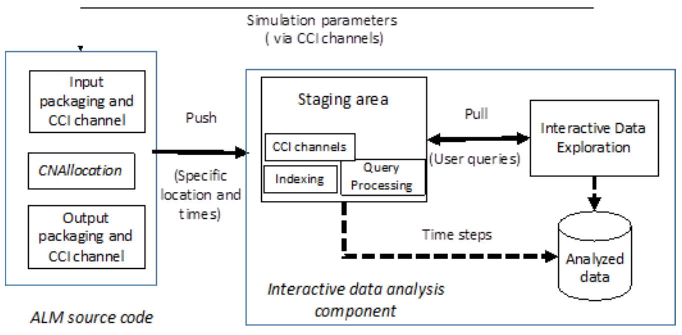
Simulation parameters (via CCI channels)
Input packaging and CCI channel
CNAllocation
Output packaging and CCI channel
ALM source code
Push
(Specific location and times)
Staging area
CCI channels
Indexing
Query Processing
Pull
(User queries)
Interactive Data Exploration
Time steps
Analyzed data
Interactive data analysis component

Fig. 5. The schematic software architecture diagram of the VOS for the ACME Land Model

As shown in Figure 5, code segments are instrumented into the source code to capture and pack the input and output data streams of the targeted module, CNAllocation. The code segments also contain functions that invoke the in-situ data communication infrastructure, including CCI channel and data buffer. The VOS has a staging area that also contains a CCI channel and data buffer. The staging area is accessible from a data exploration subcomponent. Authors first start the interactive data analysis component, which takes user-specified parameters (such as time interval, or a subset of variables) and then listens to the CCI connection requests from the simulation side. Next, authors start the instrumented ALM simulation code. When the simulation code runs to the user-defined time steps, the instrumented code packages all the relevant data into a buffer and then sends the buffer to the interactive data analysis component over the network. The data analysis component always listens on its CCI channel. When data arrive, the data analysis component unpacks the data in the staging area for follow-up data processing and analysis.

The GUI for CNAllocation data analysis and exemplar simulation data streams is illustrated in Figure 6. The first two rows show different bar plots of carbon and nitrogen allocation variables for a plant type over a specific range of time steps. The third row displays a time series from given carbon and nitrogen allocation variables; this graph allows users to track the behavior of target variables during the simulation. The fourth row includes a heat map for plotting variables having a 2D domain. Finally, the left panel shows the complete variables and time step selection.

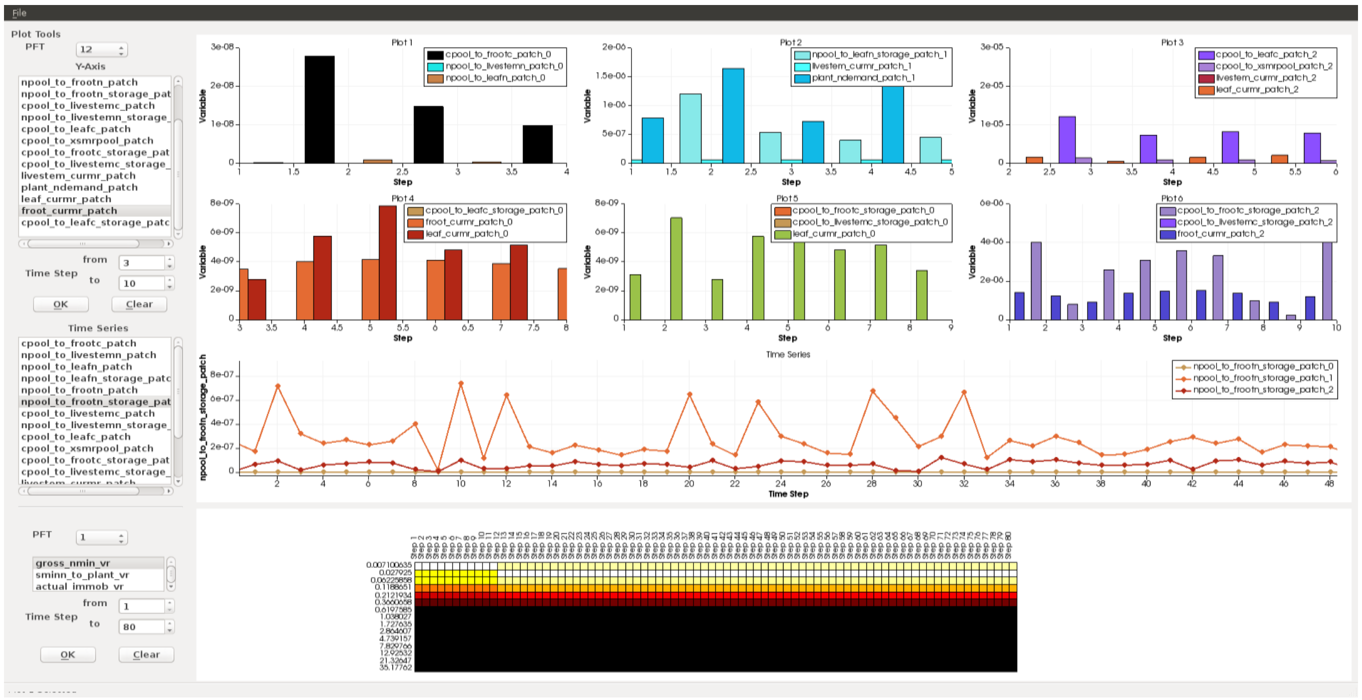

Fig. 6. GUI of the VOS data analysis of the CNAllocation functions within the ACME Land model. Users can zoom in or out to inspect different time steps or drag on any plot to highlight certain variables

IV. Conclusion

Authors have demonstrated an approach to develop a virtual observation system (VOS) for Earth system models. Authors also have implemented a VOS for the ACME Land Model using a single point-mode simulation case. By taking advantage of compiler-based software system analysis, automatic code instrumentation, and high-performance in-situ data transport, the VOS provides unique capabilities to investigate Earth system behaviors in a unique way. The VOS is designed based on non-intrusive observation principles; it preserves all the original software data flow and function calls. The VOS also allows scientists to interactively select targets of interest, such as key variables, functions, or specific break points for a simulation. Modelers can focus on investigating model behaviors without dealing with complex code instrumentation and large data handling on high-performance computing platforms. Future work will focus on two directions: 1) extending two-way communication mechanism to improve the efficiency of data collection and 2) integrating with external big data visual analysis toolkits (such as EDEN (Steed et al., 2013)) and existing advanced statistical analysis packages (such as R (Horsburgh et al., 2014) and Matlab (Pianosi et al., 2012)). The latter requires further development of data staging nodes within the system. In this extension, Dataspaces library (Docan et al., 2012) could be used to allocate and manage data staging nodes and handle push and pull operations between the VOS components, whereas Fastbit library (Wu, 2005) could be used for data indexing and query processing within these nodes, and CCI can still provide a two-way communication between simulation and analysis components to enable simulation steering.

V. Software Availability

VOS has been tested on a variety of computing environments (from desktop to high-performance computer cluster). VOS uses the software parsing and instrumentation capability developed through a functional unit testing platform for ALM (in Fortran). The functional testing platform uses compiler-based technology for software analysis and code instrumentation. The source code of the functional unit testing platform is located at a unit testing repository within bitbucket. (https://bitbucket.org/cindy387/clm85/src/cfa8d8faa43a21dcdde9b8750a9816a92477a361/?at=DEMO). Currently, the in-situ data infrastructure code is developed based on CCI libraries (in C), and is located at a CCI-in-situ repository in bitbucket.

(https://bitbucket.org/cindy387/clm85/src/83f7ade49968afef18 dd944560a343adbd6a3810/?at=In-situ). The visualization package can be found at https://bitbucket.org/benjha/dataviz-acme-land-model.

ACKNOWLEDGEMENT

This research was funded by the U.S. Department of Energy (DOE), Office of Science, Biological and Environmental Research (BER) program and Advanced Scientific Computing Research (ASCR) program, and by an ORNL LDRD project (#7409). This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory (ORNL), which is managed by UT-Battelle LLC for the Department of Energy under contract DE-AC05-00OR22725.

REFERENCE

[1] Atchley, S., Dillow, D., Shipman, G., Geoffray, P., Squyres, J., Bosilca G., and Minnich, R., 2011, The common communication interface (CCI) in the 19th IEEE Symposium on High Performance Interconnects (HOTI), Santa Clara, CA, August 23-25, 2011.

[2] Docan, C., Parashar, M., and Klasky, S., 2012. “DataSpaces: an interaction and coordination framework for coupled simulation workflows”. Cluster Computing 15(2): 163-181, doi: 10.1007/s10586-011-0162-y

[3] Horsburgh, J.S., Reeder, S. L., Data visualization and analysis within a Hydrologic Information System: Integrating with the R statistical computing environment, Environmental Modelling & Software, Volume 52, February 2014, Pages 51-61, ISSN 1364-8152, http://dx.doi.org/10.1016/j.envsoft.2013.10.016.

[4] Pianosi, F., Sarrazin, F., Wagener, T., A Matlab toolbox for Global Sensitivity Analysis, Environmental Modelling & Software, Volume 70, August 2015, Pages 80-85, ISSN 1364-8152, http://dx.doi.org/10.1016/j.envsoft.2015.04.009.

[5] Steed, C. A., Ricciuto, D.M., Shipman, G., Smith, B., Thornton, P.E., Wang, D., Shi, X., Williams, D. N., 2013, Big data visual analytics for exploratory earth system simulation analysis, Computers & Geosciences, Volume 61, December 2013, Pages 71-82, ISSN 0098-3004, http://dx.doi.org/10.1016/j.cageo.2013.07.025.

[6] Wang, D., Schuchart, J., Janjusic, T., Winkler F., and Xu, Y.,2014a. Toward better understanding of the Community Land Model within the Earth System Modeling Framework, in: Abramson, D; Lees, M; Krzhizhanovskaya, W., Dongarra, J; Sloot, P.M.A. (Eds.), Procedia Computer Science, 14th Annual International Conference on Computational Science, Cairns, Australia, 2014, Procedia of Computer Science, Volume 29, 2014, Pages 1515–1524, 10.1016/j.procs.2014.05.1375.

[7] Wang, D. Xu, Y., Thornton, P., King, A., Gu, L., Steed, C., Schuchart, J., 2014b, A functional testing platform for the Community Land Model, Environmental Modeling and Software, 2014, Volume 55, Pages 25-31, 10.1016/j.envsoft.2014.01.015

[8] Wang. D., Wu, W., Janjusic, T., Xu, Y., Iversen, C., Thornton, P., Krassovski, M., 2015. Scientific functional testing platform for environmental models: An application to the Community Land Model, International Workshop on Software Engineering for High Performance Computing in Science, 37th International Conference on Software Engineering, May 16-24, 2015, Florence, Italy. Doi: 10.1109/SE4HPCS.2015.10

[9] Wu, K., 2005. “FastBit: an efficient indexing technology for accelerating data-intensive science” J. Phys.: Conf. Ser. 16 556-560, doi: 10.1088/1742-6596/16/1/077

[10] Yao, Z., Jia, Y., Wang, D., Steed, C., Atchley, S., 2016, In situ data infrastructure for scientific unit testing platform1, in: Connelly, M. (Ed.), Procedia Computer Science, Volume 80, 2016, Pages 587-598, ISSN 1877-0509, http://dx.doi.org/10.1016/j.procs.2016.05.344.

[11] Youngsung Kim, Y., Dennis, J., Kerr, C., Kumar, R., Simha, A., Baker, A., Mickelson,S., 2016, KGEN: A Python tool for automated Fortran kernel generation and verification, in: Connelly, M. (Ed.), Procedia Computer Science, Volume 80, 2016, Pages 1450-1460, ISSN 1877-0509, http://dx.doi.org/10.1016/j.procs.2016.05.466.

[12] Yuan, F., Thornton, P. E., Xu, ,X., Sloan, VL., Iversen, C., Rogers, A., Yang B., and Wullschleger S. D., (2016). Modeling analysis of assimilate partitioning between storage and pools of multiple plant function types for simulating carbon cycles in Arctic coastal tundra ecosystem at Barrow, Alaska. JGR-biogeoscience (in preparation)