A Hadoop-Based Visualization and Diagnosis Framework for Earth Science Data

Shujia Zhou[1], Xi Yang[2], Xiaowen Li[3,5], Toshihisa Matsui [4,5], Si Liu[2], Xian-He Sun[2], Weikuo Tao[5]
[1] *Northrop Grumman Information Technology, McLean, VA 22102*
[2] *Illinois Institute of Technology, Chicago, IL 60616*
[3] *Morgan State University, Baltimore, MD 21251*
[4] *University of Maryland, College Park, MD 20742*
[5] *NASA Goddard Space Flight Center Greenbelt, MD 20771*
[1] *shujia.zhou@ngc.com*
[2] *{xyang34, sliu89}@hawk.iit.edu, sun@iit.edu*
[5] *{xiaowen.li, Toshihisa.Matsui-1, wei-kuo.tao-1}@nasa.gov*

***Abstract*—With rapidly growing computing power, ultra high-resolution Earth science simulations with a long period of time are feasible. However, it is still very challenging to distribute and analyze a huge amount of simulation results, which could be over 100TB. One key reason is that typical Earth science data are represented in NetCDF, which is not supported by the popular and powerful Hadoop Distribute File System (HDFS) and consequently cannot be analyzed with tools based on HDFS. In this paper, we propose a Hadoop-based visualization and diagnosis framework for visualizing and analyzing Earth science data. It has a data model to transform data from the format of NetCDF to CSV (Comma Separated Value) that is supported by HDFS. With this model, data can be processed with the operations such as maximize, sum, and subset through HIVE and Cloudera Impala and, therefore, typical diagnoses can be performed. In addition, the framework has a technique to visualize and diagnose HDFS-resident data with the popular visualization and diagnosis tool, IDL. To speed up this process, a concurrent reader is developed to obtain HDFS-resident data. Moreover, a dynamic reader to transfer data from a parallel file system (PFS) to HDFS is developed to efficiently visualize and diagnose PFS-resident data. The cloud resolve mode simulations are used for testing and evaluating this framework.**

Keywords—Hadoop; MapReduce; Visualization; NetCDF; Cloud Resolve Model

I. INTRODUCTION

With rapidly growing computing power, a high-performance computer with hundreds of thousands of computer processors is available for performing ultra-high resolution and long-time Earth science simulations. For example, NASA simulated CO2 global transport from May 2005 to June 2007 with NASA GEO-5 model on a 7km grid. The simulation produced nearly four petabytes (million billion bytes) of data [1].

Of climate and weather simulations, one challeng is how to model cloud. A cloud-resolving model (CRM) is an atmospheric numerical model that can numerically resolve clouds and cloud systems at 0.25~5km horizontal grid spacings such as Goddard Cumulus Ensemble (GCE) [2] [3] [4] and NU WRF [5].The main advantage of the CRM is that it can allow explicit interactive processes between microphysics, radiation, turbulence, surface, and aerosols without subgrid cloud fraction, overlapping and convective parameterization. However, it is still very challenging to distribute and analyze huge amount of simulation results which could be over 100TB. For example, a 5-day simulation of 1km GCE model with the resolution of 256x256x104 produce a single output data size of ~60GB and ~7.2TB in total. With ultra high resolution, 4096x4096x104, a 2-3 day simulation produces one single-precision variable data file of ~7GB and 125TB in total with all the relevant variables.

Because of their fine resolution and complex physical processes, it is challenging for the CRM community to 1) visualize/inter-compare CRM simulations, 2) diagnose key processes for cloud-precipitation formation and intensity, and 3) evaluate against NASA's field campaign data and L1/L2 satellite data products due to large data volume and complexity of CRM's physical processes. Hence, we need to develop tools to combine those distributed/local arrays for analysis. In addition, analyzing large data (~TB) with a desktop computer is not practical. Furthermore, distributing those large data sets

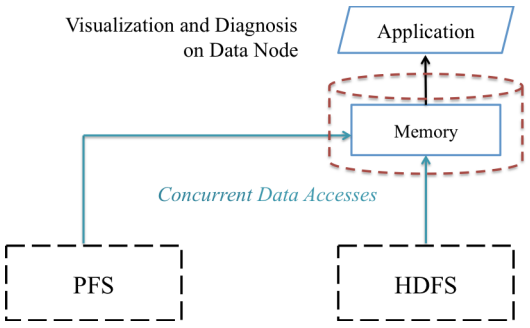

Figure 1 Illustration of Hadoop-based visualization and diagnosis framework. Data resident in PFS or HDFS are read into memory and processed to be ready for MapReduce applications and visualization and diagnosis applications.

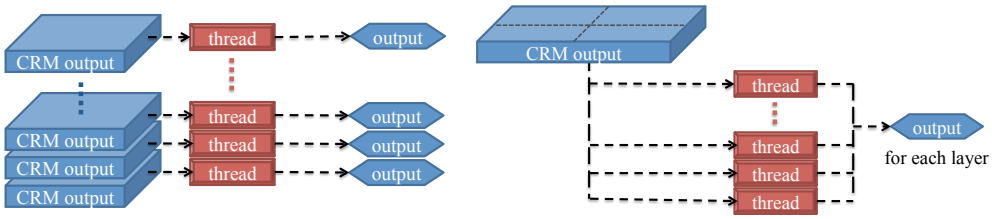

Figure 2 A parallel NetCDF to CSV format transformer: (a) Support one small domain. (b) Support one large domain that consists of subdomains.

through downloading from a website is problematic. If a user can quickly visualize and diagnose the targeted data, movement of voluminous data can be avoided. Similar problems also exist for other Earth science simulations.

MapReduce is a distributed computing framework for large-scale data analysis. It has gained growing interest in geoscience communities due to its merits of easy programming, automatic parallelism, and fault tolerance. Hadoop framework is the most popular open-source implementtion of MapReduce, which consists of Hadoop MapReduce and Hadoop Distributed File System (HDFS) [6].

However, the first challenge in applying Hadoop in Earth science is how to allow Hadoop to read climate and weather simulation as well as observation data, which are often in NetCDF [7], HDF [8], or binary format, manipulate those data sets in a flexible way, and allow a user to intelligently choose data that a user is interested in.

The second challenge is to access data efficiently. The model simulation output data often are generated under High Performance Computing (HPC) environments and saved on Parallel File Systems (PFS), such as General Parallel File System (GPFS) [9], Lustre [10], and PVFS [11], which are not accessible by the Hadoop framework. Hence, a data movement phase, transferring target data from PFS to HDFS, is inevitable. Current data movement is executed with "copy".

There are few open-source Hadoop tools intending to visualize Earth science data resident in HDFS. For example, Hue [12] is an open source Web interface for analyzing data with any Apache Hadoop. It currently has 2D XY plot and histgram plot. However, visualization in Earth science applications demand more sophisticated visualization similar to IDL [13]. At the time of writing this paper, there is no IDL support to directly access HDFS-resident data. R [14] has quite powerful visualization and statistical analysis tools. However, we find its bridge to HDFS, rhdfs [15] , cannot handle a large CSV file (~ 1GB).

In this paper, we propose a Hadoop-based visualization and diagnosis framework for Earth science data as illustrated in Figure 1. It has (1) a data model to index data, (2) a transformer to change the data format from NetCDF to CSV (Comma Separated Value) [16], which is supported by HDFS. With this data model, data can be processed with common operations such as maximize, sum, and subset through HIVE [17] and Cloudera Impala [18] and consequently typical data diagnoses can be performed. In addition, HIVE allows a user to implement User-Defined Functions (UDFs), (3) a technique to visualize and diagnose HDFS-resident data with the popular visualization and diagnosis tool, IDL, (4) a concurrent Hadoop reader to speed up the process of reading HDFS-resident data for visualization and diagnosis, (5) a dynamic Hadoop reader to transfer data from PFS to HDFS so as to dynamically visualize and diagnose the PFS-resident data. The cloud resolve mode simulation outputs are used for testing and evaluating this framework.

The rest of paper is organized as follows. Section II introduces data model as well as data transformation. Section III presents our Hadoop-based visualization and diagnosis tools and their performance. Section IV discusses unresolved issues and Section V summarizes the results.

II. Data Model and Transformer

A. Data Model

Since NetCDF4 supports HDF and binary data format can be converted to NetCDF or CSV, we will focus on NetCDF in this paper. Our data model is as follows: Read a NetCDF file, write it out in a text (CSV) format and provide a file with the relevant metadata (e.g., unit and time stamp) for reconstructing it into CFMC-NetCDF in the future. (CFMC-NetCDF is a NetCDF compliant with Climate and Forecast Metadata Convection.) A subset data file will be available in CFMC-NetCDF.

The style of one variable, one time frame per file is used for a large simulation file (e.g., 1GB for one variable per time frame) while one variable, multiple time frames per file is used for a small simulation output (e.g., 10MB for one variable per time frame). For the tools built on top of HDFS such as HIVE, Impala, or Hadoop Stream "cat" function, all files in an HDFS directory are processed as a whole. As long as a data item (e.g., temperature) is indexed (e.g., time, layer, latitude, longitude), data can be processed independent of whether they are in a file or a directory. In this way, data output from an individual subdomain (e.g., MPI subdomain) can be assembled and processed as a whole.

B. Data Transform

A parallel data transformer has been developed to convert NU-WRF/GCE simulation output data from the format of NetCDF to CSV with Climate and Forecast Metadata Convection (CFMC) [19] according to our above data model. The parallelism is implemented with MPI [20]. As shown in Figure 2, this transformer currently supports two kinds of simulation outputs: (1) one small domain. This is used in a long-time simulation with a large number of WRF/GCE output

This work is sponsored by the NASA Advanced Information Systems Technology

files, (2) one large domain consisting of subdomains. This is used in a short-time simulation with a high resolution grid.

III. Visualization and Diagnosis in Hadoop

A. Test Planforms

Our NASA NCCS Hadoop cluster has 34 nodes with InfiniBand FDR interconnect. Each node has 16 cores (not hyperthreaded) of Intel CPU E5-2670@2.6GHz, 16GB RAM, and 2TB disk. Each core has 2.6 GB memory. Cloudera 5.4.1 is used. In addition, we use 17 nodes at the IIT cluster: one master node and eight slave nodes for Hadoop and other eight nodes as OrangeFS [21] server nodes. We use OrangeFS version 2.8.6 that is the commercial version of PVFS2. Each node is equipped with two 2.3GHz Opteron quad-core processors, 8GB memory, and a 250 GB 7200 RPM SATA hard drive. All nodes are connected through one Gigabit Ethernet. Both Hadoop 1.0 and Hadoop 2.5.2 are used.

B. Data

In this paper, we use the cloud-resolving weather simulation output from a NU-WRF model with a 600x552x60 grid of 3km resolution and 48 hour simulation time. The output is hourly and has 10 variables. Its original output is in NetCDF. Each file has one variable and one time stamp. Its data format has 6 columns: row id, time stamp, layer, latitude, longitude, variable. With our parallel NetCDF-to-CSV transformer, one CSV file has 999,380,097 bytes. For one varialbe, there are 48 files. For all 10 variables, there are 480 files with about 465GB in total.

In this paper, we will report the experiment results with one variable, rain (qr), in 48 output files.

C. Visualization with IDL

IDL is a powerful and widely used visualization and diagnosis tool, especially in geoscience applications. A significant number of IDL tools have been developed by researchers and other users. So reusing those tools for visualizing and diagnosing HDFS-resident data is highly desirable. At the time of writing this paper, there is no IDL support to directly access HDFS-resident data.

We have developed a prototype to use Java to read HDFS-resident data and save as a Java object. In addition, we developed an IDL wrapper to read data out of that Java object and pass them to an IDL code for visualization and diagnosis. Although this approach is feasible, it incurs data conversion overhead from Java to IDL. In addition, such a wrapper has to be rewritten for a different IDL application.

After investigating the existing IDL data input features, we

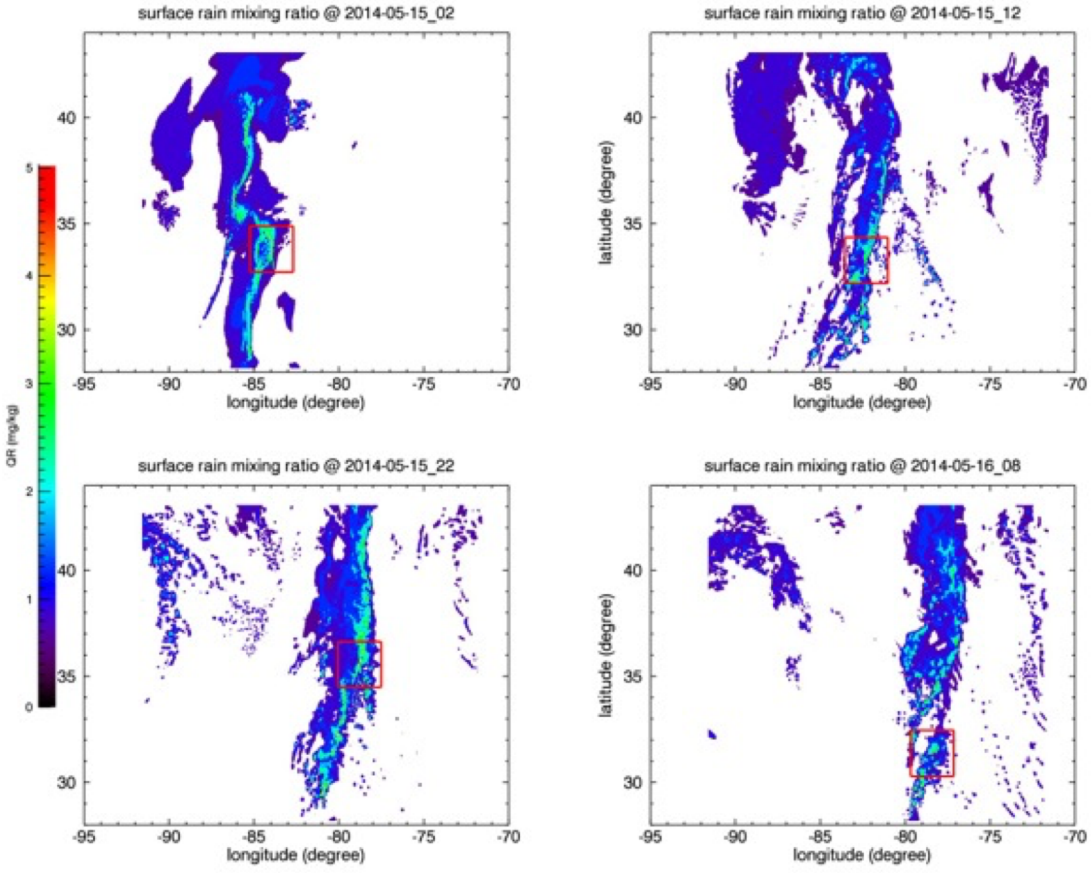

Figure 3 Illustration of visualization and subsetting. These four plots show rain (qr) of the surface layer in four different time frames. A small red box encloses the data subset around the maximum qr value. It varies among four time frames.

have developed a solution: (1) Use Hadoop Streaming "cat" function to read the HDFS-reside data and write into the standard output (terminal), (2) Use IDL to read this standard output and visualize and diagnose the data. In this way, we only need to modify the input function of an existing IDL code so as to read the data through the standard input. Our initial experiments show that is a general and feasible approach. For example, it takes ~14 minutes to visualize 2D data and create an animation (movie) of 48 HDFS-resident CSV files with ~48GB in total. On average, each file takes 17.5 seconds to read and visualize, of which 15 seconds is for reading each file out of HDFS. For the same visualization with Linux "cat", it takes 17 minutes at NASA Discover supercomputer where the data files is stored in IBM GPFS files sytem.

D. Subset and diagnosis

With our data model, HDFS-resident data items with the corresponding indexes can be processed straightforwardly with HIVE and Impala. In particular, we can perform subsetting and other common diagnosis calculations such as maximize, sum, and average. To illustrate this approach, we develop a HIVE script on one file with one time frame to (1) Get the maximum value of a variable (qr) of the surface layer, (2) Get the coordinate (latitude, longitude) with that maximum qr value, (3) Subset a box of data points around that maximum qr point. The box is constructed with one-degree extension in four directions of that maximum qr point. After that, we develop a bash shell script to run through all 48 files (e.g., 48 time frames). Without any optimization, it takes about 104 minutes to complete this subsetting process. To speed it up, we develop a similar code for Impala. The same subsetting processing takes about 13 minutes without any optimization. In future, we plan to optimize the performance. For example, partition on "layer" is expected to improve the performance since the target data are a subset within a layer.

One interesting observation is that it only takes a few hours for an inexperienced HIVE and Impala researcher to develop codes for this adaptive subsetting. We believe that productivity using HIVE and Impala for subsetting and common diagnosis is higher than using traditional parallel programming such as MPI.

E. Concurrent Hadoop reader

Currently we use one node in Hadoop cluster to perform visualization and analysis with IDL. IDL has multi-thread and multi-node features for handling a large domain. We will explore that feature in the future.

In the case of one visualization node in HDFS, one visualization job can have multiple tasks, and each task visualizes one file. When creating an animation (movie) of a time-series simulation output, the visualization has to perform on each file in the same order as the simulation time sequence. In addition, typical visualization is on a whole file rather than blocks stored in HDFS. Therefore reading data out of HDFS is one bottleneck for typical visualization applications.

We have developed a solution to address this issue. Figure 4 show the procedure of proposed visualization mechanism. To accelerate the read process, we deploy concurrent read to improve the read throughput. Since HDFS distributes data blocks across the multiple data nodes, the blocks of a file will be read concurrently from multiple data nodes and aggregated into a whole file. The visualization task starts only when one file has been fully read.

The number of concurrent read threads are controlled. This is because (1) memory capacity is limited; (2) access to data nodes could increase I/O interference and bursty gathering communication could cause I/O contention, which eventually leads heavy latency for a visualization task. As shown in Figure 4, after visualizing File 1, the corresponding memory resource is released for next concurrent read operation that will aggregate a whole file in memory. While File 2 is being visualized, multiple threads are reading blocks and aggregating them into File 3. 15 concurrent threads are used for measuring I/O events since 15 blocks of 64MB just covers one data file of 999,380,097 bytes. This processing flow can be pipelined. In adition, memory usage has to be tuned accordingly for optimal performance.

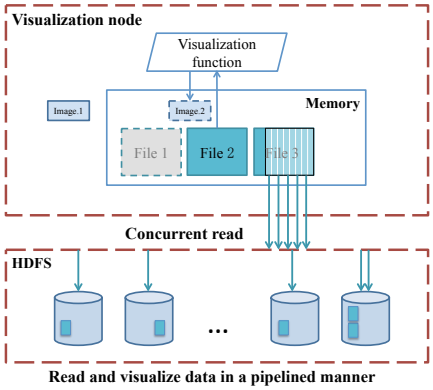

Figure 4 The process flow of visualizing time-series data files resident in HDFS. HDFS-resident data are read concurrently. Once a target file is fully aggregated, visualizing that file starts.

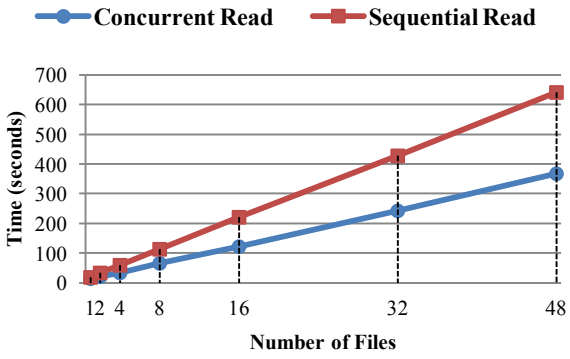

Figure 5 The performance of concurrent read. Each file is about 1GB and there are 48 files in total. The baseline is a sequential read.

The performance of concurrent read is shown in Figure 5. The experiment is conducted under eight HDFS data nodes

environment. The reader (initiated at one node) reads input file one by one but concurrently reads blocks. It takes 367 seconds and 641 seconds for concurrent reads and sequential reads, respectively. It is ~57% speedup.

F. *Dynamic Hadoop reader*

Typically high-performance computing applications output and store their data in PFS. Some simulation data files can be huge (e.g., over 100TB). In addition, only some events in those data files are sufficiently interesting for further analysis. For example, only a small region around the center of Hurricane or tornado is very important for detailed analysis. Therefore, an efficient approach is to first perform visualization and diagnosis on part of data resident in PFS to search and identify those interesting events before copying those files onto HDFS. To address this issue, we have developed a dynamic Hadoop reader, so-called PortHadoop, to fetch user-specified data files from PFS to the memory of a Hadoop cluster.

PortHadoop supports multiple PFS implementations including PVFS2 and CephFS [22]. PortHadoop does not intend to reconcile PFS and HDFS. Instead, it supports direct data access between these two file systems. PortHadoop reads PFS data directly from a (remote) PFS and stores the data onto PortHadoop memory system. Both the interface of PFS and HDFS are unchanged. To access and process a file resident in PFS, a user only needs to add "PFS:" in front of the directory where the target file is located. To achieve those goals, we have modified multiple existing components in Hadoop system and created several new components, ensured data alignment and data integrity, developed a data prefetching mechanism to further utilize the bandwidth between PFS and Hadoop, and finally developed a fault tolerant mechanism. We elaborate design and implementation details in a separate paper[23].

Visualization and diagnosis often require an entire file. Therefore, we configure each map task to process one entire file rather than a block. In our experiment, 48 files are used. Each file is about 1GB.

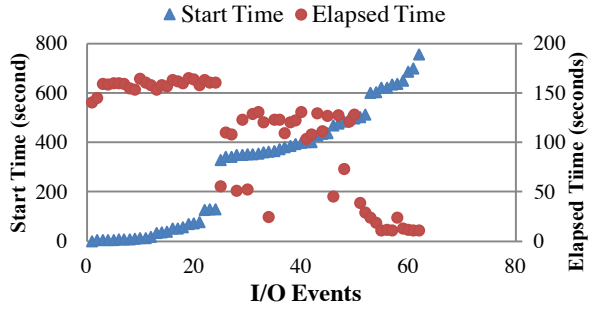

Figure 6 The performance of dynamic Hadoop reader. Data is first read directly from remote PFS and then processed with Map function of a MapReduce application (i.e., WordCount). Start time indicates the time when a read event begins. Elapsed time is the duration for a whole event.

Eight nodes are used for PFS (OrangeFS) while other eight nodes are used for Hadoop slave nodes. Each Hadoop slave node is configured with three concurrent map tasks. OrangeFS is configured to enable Hadoop slave nodes as clients to access data resident in PFS. In this experiment, we focus on measuring the IO event time. Therefore, we only turn on Map function. With simultaneously calling Hadoop "put" command, concurrently copying 48 files from PFS to HDFS with one replication takes 61 minutes. However, it takes less than 13 minutes with our dynamic Hadoop reader for reading data from PFS and processed with Map function. On average, a map operation takes about 5 minutes for processing a file of 1GB.

Figure 6 shows the performance of our dynamic reader where I/O events are concurrent. Contention for bandwidth and interference in data accesses cause heavy latency. At the beginning, reading a file takes ~170 seconds. However, it tasks about 10 seconds after the initial stage, where only a few concurrent I/O requests to remote PFS.

Conventional MapReduce adopts backup tasks to accelerate the overall completion time for a job, namely speculative execution. When only a few tasks remain and computation resources idle, the speculative execution is triggered. In our experiment, there are about 63 I/O events rather than 48 I/O events. This is because speculative map tasks require extra I/O events. Although disabling speculative execution reduces total number of I/O events, we found that overall performance becomes worse in our dynamic reader.

IV. Discussion

We have developed a method to enable R to read a large file on HDFS with functions in rhdfs. rhdfs provides two read functions: hdfs.read() and hdfs.line.reader(), both of which have limitations on the size of a file to be read. For hdfs.read(), the default value of read size is 64KB. However, it can only read 64KB regardless of an assigned value of read size. For hdfs.line.reader(), its default value of read size is 1000 lines, which can be changed to read more lines with the available memory of JVM. Thus, through executing multiple times of hdfs.line.reader(), we can read a CSV file up to 12GB in a Hadoop node with 16GB memory. We will test this solution with multiple Hadoop nodes.

The current visualization strategy uses one visualization data node and input data are stored across data nodes. Visualization is serial. We plan to utilize the IDL features of multi-thread per node and multiple nodes to make it parallel. This becomes very important in processing a large file with over 30 GB.

In our current dynamic reader, PFS-resident data is fetched to memory of Hadoop cluster so that MapReduce application such as Word Count can process. We plan to develop interface for high-level Hadoop tools such as HIVE to facilitate processing the PFS-resident data. In addition, we also plan to add a reformat tool to enable HIVE to process NetCDF files.

Converting from NetCDF to CSV takes computing resource and uses more disk space. We will investigate the

techniques to allow Hadoop tools such as HIVE to directly process NetCDF files resident in Hadoop.

V. SUMMARY

We have developed a Hadoop-based framework for visualizing and diagnosing Earth science data. Its data model allows for transforming data from NetCDF to CSV with indexes. Consequently, Earth science data can be stored in HDFS and straightforwardly processed by Hadoop-based tools for subsetting and diagnosis. In addition, its IDL tool is able to visualize and diagnose HDFS-resident data accessed through Hadoop streaming "cat". Moreover, its concurrent Hadoop reader is shown to speed up the process of reading data from HDFS. Finally, its dynamic Hadoop reader is capable of fetching PFS-resident data and making them ready for MapReduce applications including subsetting and diagnosis.

ACKNOWLEDGMENT

We would like to thank the NASA Center for Climate Simulation (NCCS) for providing Hadoop cluster and Discover supercomputing resources and Daniel Duffy, John Thompson, Garrsion Vaughan, and Michael Bowen for sharing their insight on Hadoop technology. We also would like to thank Beau Legeer of Exelis for his help in prototyping the IDL code for reading streaming data.

Tao, Matsui, Li, Zhou and Sun are supported by NASA AIST 2014 and Yang, Liu and Sun are supported by NSF under NSF grants CNS-0751200, CCF-0937877, and CNS-1162540.

REFERENCES

[1] "NASA Web." [Online]. Available: http://www.nasa.gov/press/goddard/2014/november/nasa-computer-model-provides-a-new-portrait-of-carbon-dioxide/index.html#.VePLSd64N1L.

[2] W.-K. Tao and J. Simpson, "The Goddard cumulus ensemble model. Part I: Model description," *Terr Atmos Ocean. Sci*, vol. 4, no. 1, pp. 35–72, 1993.

[3] W.-K. Tao, "Goddard Cumulus Ensemble (GCE) model: Application for understanding precipitation processes," in *Cloud Systems, Hurricanes, and the Tropical Rainfall Measuring Mission (TRMM)*, Springer, 2003, pp. 107–138.

[4] W.-K. Tao, S. Lang, X. Zeng, X. Li, T. Matsui, K. Mohr, D. Posselt, J. Chern, C. Peters-Lidard, P. M. Norris, I.-S. Kang, I. Choi, A. Hou, K.-M. Lau, and Y.-M. Yang, "The Goddard Cumulus Ensemble model (GCE): Improvements and applications for studying precipitation processes," *Atmospheric Res.*, vol. 143, pp. 392–424, Jun. 2014.

[5] C. D. Peters-Lidard, E. M. Kemp, T. Matsui, J. A. Santanello, S. V. Kumar, J. P. Jacob, T. Clune, W.-K. Tao, M. Chin, and A. Hou, "Integrated modeling of aerosol, cloud, precipitation and land processes at satellite-resolved scales," *Environ. Model. Softw.*, vol. 67, pp. 149–159, 2015.

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010, pp. 1–10.

[7] "NetCDF." [Online]. Available: http://www.unidata.ucar.edu/software/netcdf/.

[8] M. Folk, A. Cheng, and K. Yates, "HDF5: A file format and I/O library for high performance computing applications," in *Proceedings of Supercomputing*, 1999, vol. 99, pp. 5–33.

[9] F. B. Schmuck and R. L. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *FAST*, 2002, vol. 2, p. 19.

[10] S. Donovan, G. Huizenga, A. J. Hutton, C. C. Ross, M. K. Petersen, and P. Schwan, "Lustre: Building a file system for 1000-node clusters," in *Proceedings of the Linux Symposium*, 2003.

[11] R. Ross and R. Latham, "PVFS: A Parallel File System," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, New York, NY, USA, 2006.

[12] "Hue." [Online]. Available: http://gethue.com.

[13] "IDL." [Online]. Available: http://www.exelisvis.com/ProductsServices/IDL.aspx.

[14] "R-project." [Online]. Available: https://www.r-project.org.

[15] "rhdfs." [Online]. Available: https://github.com/RevolutionAnalytics/rhdfs.

[16] "CSV." [Online]. Available: https://en.wikipedia.org/wiki/Comma-separated_values.

[17] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A Warehousing Solution over a Map-reduce Framework," *Proc VLDB Endow*, vol. 2, no. 2, pp. 1626–1629, Aug. 2009.

[18] M. Kornacker and J. Erickson, "Cloudera Impala: Real Time Queries in Apache Hadoop, For Real," *Ht Tpblog Cloudera Comblog201210cloudera-Impala-Real-Time-Queries--Apache-Hadoop--Real*, 2012.

[19] "CFMC." [Online]. Available: https://earthdata.nasa.gov/standards/climate-and-forecast-cf-metadata-conventions.

[20] M. Snir, *MPI--the Complete Reference: The MPI core*, vol. 1. MIT press, 1998.

[21] M. Moore, D. Bonnie, B. Ligon, M. Marshall, W. Ligon, N. Mills, E. Quarles, S. Sampson, S. Yang, and B. Wilson, "OrangeFS: Advancing PVFS," *FAST Poster Sess.*, 2011.

[22] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 307–320.

[23] X. Yang, N. Liu, B. Feng, X.-H. Sun, and S. Zhou, "PortHadoop: Support Direct HPC Data Processing in Hadoop," in *2015 IEEE International Conference on Big Data*, Santa Clara, California.