Visualization and Diagnosis of Earth Science Data through Hadoop and Spark

Shujia Zhou[1,4], Xiaowen Li[2,4], Toshihisa Matsui [3,4], Weikuo Tao[4]

[1] *Northrop Grumman Information Technology, McLean, VA 22102*

[2]*Morgan State University, Baltimore, MD 21251*

[3] *University of Maryland, College Park, MD 20742*

[4]*NASA Goddard Space Flight Center Greenbelt, MD 20771*

[1] *shujia.zhou@ngc.com*

[4]*{xiaowen.li, Toshihisa.Matsui-1, wei-kuo.tao-1}@nasa.gov*

***Abstract*— Large data (over Terabyte) are produced by ultra high-resolution Earth science simulations with a long period of time. This creates a challenge to distribute and analyze in an effective, efficient, and scalable way. 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 report a system for visualizing and analyzing Earth science data based on Hadoop and Spark. It transforms data from the format of NetCDF to CSV (Comma Separated Value) that is supported by HDFS and indexes data appropriately to save storage space as well as manipulate flexibly through HIVE, Impala, and SparkSQL. Adaptive subsetting and visualization of cloud resolve model simulation data are used to validate and demonstrate the features of this system.**

Keywords—Hadoop; Spark; MapReduce; Visualization; NetCDF

I. Introduction

With the advent of high-performance computers consisting of hundreds of thousands of computer processors, performing ultra high-resolution and long-time Earth science simulations are possible. For example, an ultra high resolution (e.g, 4096x4096x104 grid, 250-meter resolution) and one model-simulation-day cloud model simulation with NASA Goddard Cumulus Ensemble (GCE), a cloud resolving model, takes 5 wall-clock days with 4096 computer processors at the NASA NCCS Discover supercomputer. However, its single-precision output data can be about 2.5TB in total with all the relevant variables [1][2]. To simulate evloution of a weather system, a typical simulation takes 6 model simulation days, which would create 15TB data.

Analyzing large data (over Terabyte) with a desktop computer is not practical. Furthermore, distributing those large data sets through downloading from a website is problematic. If a user can obtain adequate insight through quickly visualizing and diagnosing the targeted data, movement of voluminous data can be avoided. Similar problems also exist for other Earth science simulations (e.g., NU WRF[3]).

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 implementation of MapReduce, which consists of Hadoop MapReduce and Hadoop Distributed File System (HDFS) [4]. The in-memory computational engine, Spark [5] [6], alleviates expensive disk I/O for storing intermediate result, significantly improves the performance, especially for interactive and iterative computations. Spark provides rich APIs, including MapReduce, to efficient programming. In addition, Spark supports Python as well as R.

However, the first challenge in applying Hadoop or Spark in Earth science is how to allow Hadoop or Spark 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 for further analysis.

The second challenge is to access data efficiently. The model simulation output data are often generated under High Performance Computing (HPC) environments and saved on Parallel File Systems (PFS), such as General Parallel File

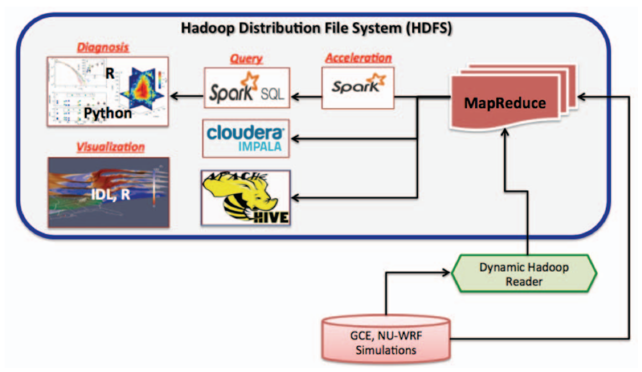

Figure 1 Illustration of a visualization and diagnosis framework realized with Hadoop as well as Spark. Data resident in a local file system, PFS or HDFS are read into memory and processed to be ready for visualization and diagnosis via MapReduce-like strategy.

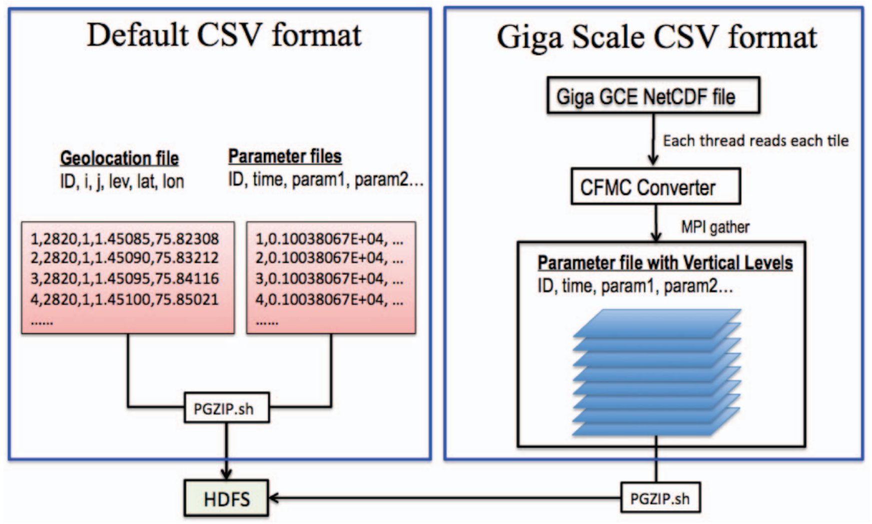

Figure 2 A parallel NetCDF-to-CSV format transformer as well as compressor: The default one supports a small domain (up to 3D grid) and the Giga one supports a large domain through treating each layer (2D grid) as a subdomain.

System (GPFS) [9], Lustre [10], and PVFS [11], which are not accessible by the Hadoop framework at the time of writing this paper. To avoid data movement from PFS to HDFS, Yang et al. [12] developed a dynamic Hadoop reader, so-called PortHadoop, to fetch user-specified data files from PFS to the memory of a Hadoop cluster.

There are few open-source Hadoop tools intending to visualize Earth science data resident in HDFS. However, visualization in Earth science applications demand more sophisticated visualization tools (e.g., 3D) 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. Yang et al. [15] have developed the methods to use MapReduce-like strategy to enable R to visualize and analyze data in a scalable way based on Hadoop as well as Spark.

In this paper, we report a visualization and diagnosis framework for Earth science data introduced in [1], as illustrated in Figure 1. This framework has been realized with Hadoop and Spark. It has (1) a data model to index data in time and geo-location, (2) a parallel transformer to change a data format from NetCDF to CSV (Comma Separated Value) [16], which is supported by HDFS. With this data model, data can be processed with basic operations such as maximize, sum, and subset through HIVE [17], Cloudera Impala [18], and advanced operations such as correlation and machine learning with Spark. In addition, HIVE SQL and Spark DataFrame/SQL allow 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 Web interface to visualize a HDFS-resident data in IDL and diagnose data in Python SPARK (i.e., PySpark) and subset HDFS-resident data with Impala, (5) a dynamic Hadoop reader to transfer data from PFS (e.g., GPFS, PVFS, CephFS) to HDFS so as to dynamically visualize and diagnose the data [12]. The cloud resolve mode simulation outputs from both GCE and NU WRF are used for testing and evaluating this framework.

The rest of paper is organized as follows. Section II introduces data model, data transformation as well as HIVE table construction. Section III presents our visualization and diagnosis tools based on Hadoop and Spark. Section IV discusses unresolved issues and Section V summarizes the results.

II. Data model, Transformer, and HIVE Tables

A. Data Model

Since NetCDF4 supports HDF and binary data format can be converted to NetCDF or CSV, we have been focusing on NetCDF [1]. As illustrated in Figure 2, our data model is as follows: (1) Convert non-NetCDF files (e.g., GCE's binary data format) to NetCDF files, (2) Write them out in a text (CSV) format and provide a file with the relevant metadata (e.g., unit, time stamp, CFMC compliant variable names) for reconstructing them into CFMC-NetCDF in the future. (CFMC-NetCDF is a specialized NetCDF format compliant with Climate and Forecast Metadata Convection [19].) A subset data file will be available in CFMC-NetCDF.

To mitigate the risk of oversized files (e.g., over 5GB), which could not be efficiently supported by a local file system, we develop the following approaches: (1) Multiple files. One file contains geo-locations (e.g., 3D grid). Since this geo-location file is same for all the physical parameters, we only create one to save disk space. One file per 3D grid per time frame for cloud parameters (e.g., rainfall), for atmospheric dynamic parameters (e.g., upward air velocity) and for atmospheric specific parameters (e.g., tendency of air temperature due to longwave_heating), respectively; (2) One file per layer per time frame for large domains (e.g., 4096x4096x104 grid) (see Figure 2).

To save the storage space, we use Gzip to compress the data in a local file system before storing into HDFS in HIVE tables. For example, about 14.26TB of NU WRF 2500x2500x50 simulation output data can be reduced to about 1.72TB, which is close to the original data size in NetCDF.

B. *Data Transform*

A parallel data transformer has been developed to convert NU-WRF as well as GCE simulation output data from the format of NetCDF to CSV with Climate and Forecast Metadata

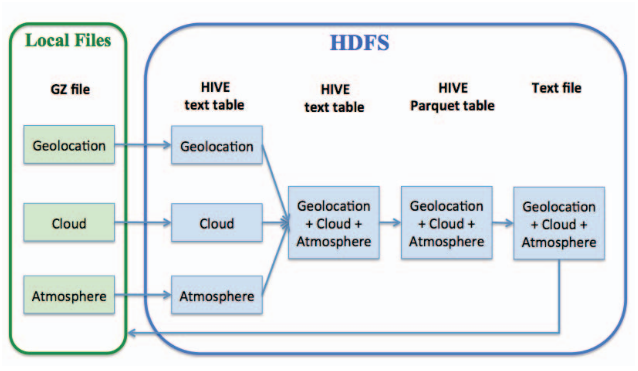

Figure 3 The data flow of building HIVE tables from gzip compressed csv files, to individual HIVE table, to HIVE tables with the desired parameters in the formats of text as well as Paraquet, to subset data files in the format of plain text.

Convection (CFMC) [19] following our data model discussed above. 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 or GCE output files, (2) one large domain. This is used in a short-time simulation with an ultra high-resolution grid.

C. *HIVE Table Construction*

For each gz compressed csv file, a HIVE table is built. Since each grid point was indexed with an ID, HIVE join operations are used to join a geo-location HIVE table with physical- parameter HIVE tables as shown in Figure 3. This is a compute-intensive operation. For each time frame, it could take half a hour for NU WRF simulation output with 2500x2500x50 grid. This approach is also applicable to the case that one file has one grid layer per time frame such as GCE 4096x4096x104 grid. In this way, a HIVE table can be built with the user-specified physical parameters such as temperature, upward air velocity and rain drop along with time and geo-location in a flexible way. To improve query performance, the Parquet data format is used. With this kind of tables, HIVE, Impala as well as SparkSQL can be used for basic analysis and subsetting along the direction of longitude, latitude, layer or time.

III. VISUALIZATION, DIAGNOSIS, AND SUBSETTING

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.

A Tomcat web server is installed on one of Hadoop data nodes so that Hadoop command-line instructions can be executed through RestAPIs. In addition, IDL is installed in this data node to visualize data from a local file system as well as HDFS. To visualize and analyze a large data set, one node in a Hadoop cluster has been augmented to have 256GB memory.

B. *Data*

In this paper, we use the cloud-resolving weather simulation outputs from (1) a NU-WRF model with an ultra high-resolution (e.g., 6 days, 2500x2500x50 grid of 2km resolution). The output is hourly and has about 20 variables. Its original output is in NetCDF. After data transformation, a cloud data file in csv and compressed with gzip has about 3GB, (2) a GCE model with a long-term simulation (e.g., 30 days, 256x256x46 grid of 1km resolution). The output is every 3 hours and has about 20 variables. Its original output is in a binary data format. For a cloud data file in csv and compressed with gzip has about 120MB.

C. *Subset*

With our data model, HDFS-resident data items with the corresponding indexes can be stored in HIVE tables and processed straightforwardly with HIVE, Impala and SparkSQL. In particular, we can perform subsetting and common diagnosis calculations such as maximize, sum, and average. SparkSQL has been adding more sophisticated functions such as correlation. For those unavailable functions, User-Defined Functions (UDFs) can be created. One typical subsetting example is to first identify a region where an interesting event such as the center of a Hurricane or a thunderstorm, and then save the data surrounding that region for further analysis with advanced methods and tools. The criterion for identifying the region can be implemented with those available functions in HIVE, Implala, SparkSQL or UDFs. For example, we can use the maximum upward air velocity, *w*. A bash script with Impala SQL commands is developed to process each time frame in a HIVE table: (1) Search the maximum *w* value, (2) Get the latitude and longitude coordinates with that maximum *w* value, (3) Subset a box of data points around that maximum *w* point. The box is constructed with *n* grid-point extension in four directions of that maximum w point, where *n* is chosen by a user. The subsetting processing with Impala takes a few minutes without any optimization. The similar subsetting process can be realized with SparkSQL. The first step is to create a DataFrame out of the HIVE table. On the DataFrame, SparkSQL can perform the subsetting through similar procedures as Impala's. In Spark1.6 under Cloudera, we found that creating a DataFrame takes little time (i.e., less than a second), however, performing SparkSQL on a Data Frame with a large data size (e.g., 500GB) needs sufficient memory. However, Impala appeared not requiring such a large memory to process the data with the same size.

This work is sponsored by the NASA Advanced Information Systems Technology

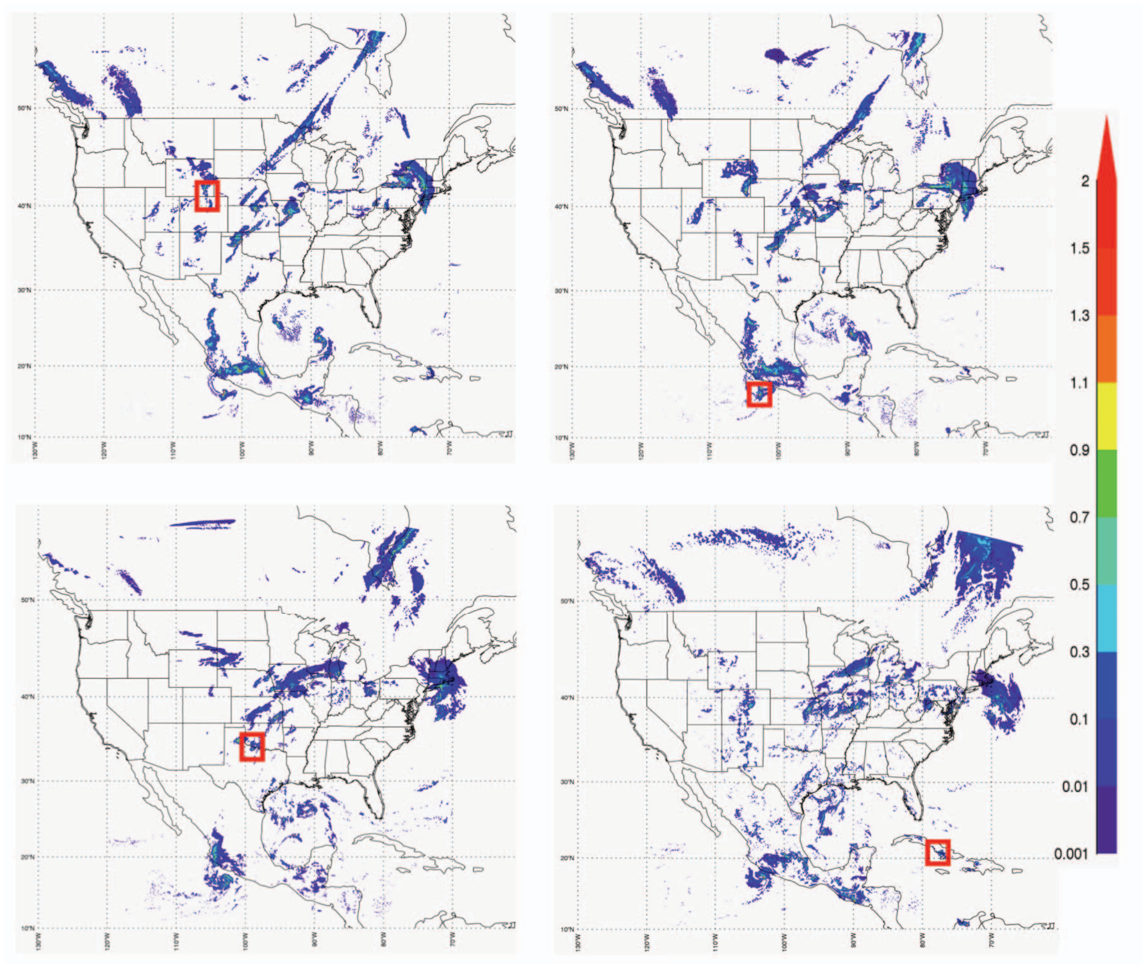

Figure 4 Illustration of 2D visualization and adaptive subsetting of NU WRF simulation outputs. These four plots show rain of the surface layer in four different time frames. A small red box encloses the data subset around the maximum rainfall rate. It varies among four time frames.

D. *Visualization*

IDL is a powerful and widely used visualization and diagnosis tool, especially in geoscience applications. It includes convenient map projections and 3D plotting packages. A significant number of sophisticated IDL tools have been developed by researchers and other users. So reusing those tools for visualizing, especially 3D, 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 solution to address this issue [1]: (1) Use Hadoop Streaming "cat" function to read HDFS-resident 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. However, the performance of this approach relies on the throughput of data that are gathered and transferred to the visualization node. Visualizing a subset data rather than the whole data set is an efficient way of utilizing this method.

As shown in Figure 4, at any given time, the weather systems generally occupy a small percentage over the US continent. If one is interested in one weather system or systems over a fixed location, subsetting the large output data can be much more efficient than downloading the whole dataset. Figure 4 gives an example of subsetting the strongest precipitation system over the domain.

In Figures 4 and 5, we illustrate our approach to adaptively subset a large data set with Hadoop Impala and visualize with IDL. To find interesting events such as storm centers in a large data set, a user can use 2D visualization of rainfall on Earth surface layer (the bottom layer in a 3D grid) for all the time frames. The algorithm to define a storm center could be simple or sophisticated. In our current case, we use the maximum rainfall. Our Impala code finds that maximum value as well as its latitude and longtitue coordidates and pass them to the IDL code to indicate the interested areas (see Figure 4). In addition, the data in the enclosed box are saved into HDFS or a local file system. Those subset data sets are further visualized in 3D and with various combinations of parameters (see Figure 5). Since the relevant physical parameters such as ice, rain, wind are stored in the same HIVE table, the parameters of interests in the subset box can be saved into one file and further visualized in various ways to investigate the detailed physical processes. For example, rain and ice are shown below and above the 20th layer (level), respectively. In addition, both updraft (upward velocity > 1 meter/second) and down draft (upward velocity < -

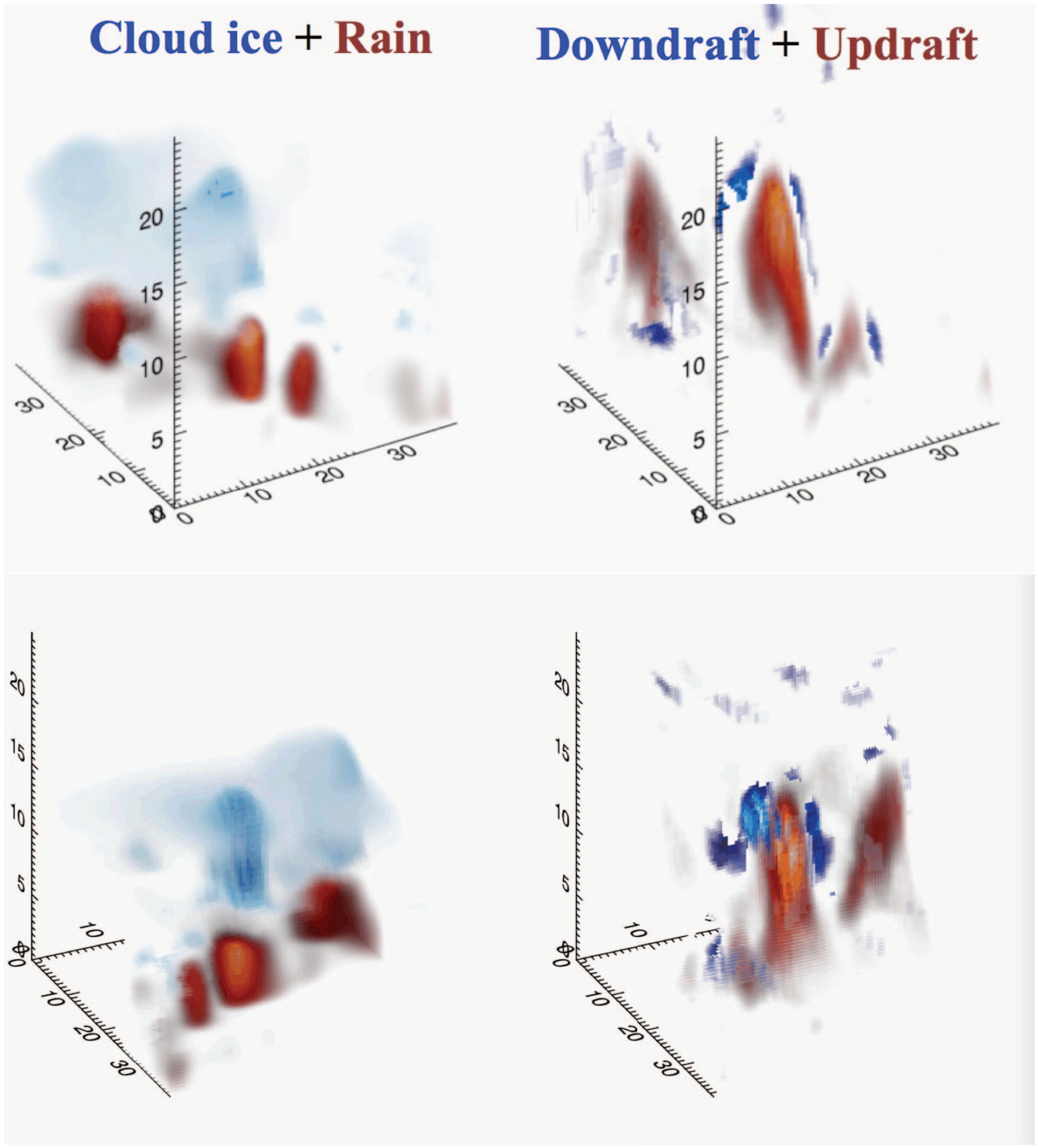

Figure 5 Illustration of 3D visualization on subset data of GCE simulation outputs. These two plots (top and bottom) show cloud ice, rain, downdraft and updraft, in one of 3D subset boxes as shown in Figure 4, visualized in two different angles. The left panel is ice in blue and rain in red. Above 0^oC degree is ice and below 0^oC degree is rain. Ice and rain are separated at the level of 20. The right panel is updraft, upward air velocity>1m/s, in red and downdraft, upward air velocity< -1m/s, in blue. They are not separated in height or level.

1 meter/second) are shown through all the layers.

We also developed methods to utilize MapReduce in Hadoop and Spark to analyze and visualize a large-size data file created in an ultra high-resolution simulation or a large number of files with R in a distributed way [15]. Those methods can be used to visualize all the layers of all the time frames, which could be very helpful for a user to search the interesting events in a comprehensive way.

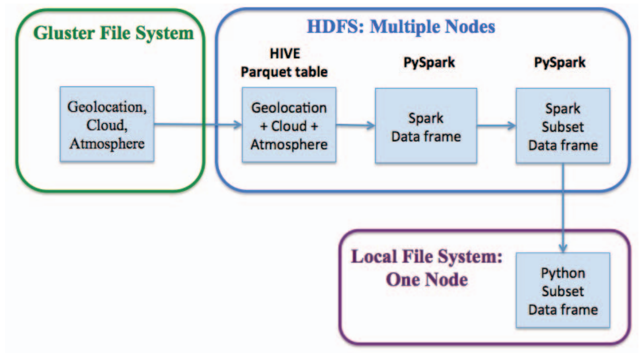

Figure 6 The data flow of an end-to-end analysis. Files in a local or remote file system are loaded to build HIVE tables with the desired parameters. A Spark Data Frame is created with a HIVE Table. On that DataFrame, analysis and subsetting are performed with Spark Python. The Data Frame with the subset data is converted to a Python Data Frame on a Spark driver node.

E. *Diagnosis*

For advanced diagnosis, PySpark is used. We have developed an end-to-end solution as shown in Figure 6. Files in a local file system or a remote file system such as Gluster File System are loaded into HDFS and stored in a HIVE table with the Paraquet format. A Spark Data Frame can be created from such a HIVE table. With this Data Frame, various analyses and subsettings can be performed with Python Spark/Spark SQL. Typically a user prefers its own Python code to further perform sophisticated data analysis such as correlation calculation and 3D visualization, which are often available in a local, no-Spark system. We found that a user can first collect distributed data onto a Spark driver node and convert this PySpark Data Frame to a Python Data Frame. In this approach, a user is able to carry out an end-to-end large data analysis in an interactive PySpark shell.

F. *Web interface*

We have been developing a Web portal to allow a user to subset, diagnose, visualize data and save the subset data. This is realized with a Tomcat server located in one of Hadoop data node where an IDL software installed. In this way, a user can use IDL to visualize an HDFS-resident data through Web interface. The RestAPIs are written in JAVA. IDL, PySpark, Impala commands can be invoked through the JAVA servlets.

IV. DISCUSSION

Hadoop and Spark ecosystem become popular and their tools are increasingly powerful and mature. However, their current support targets are industrial users, especially in internet, finance, and health care. Earth science users have to adopt the available tools for their applications.

For example, HDFS does not support the NetCDF file format. We have to convert data from NetCDF to CSV, which wastes computing and storage resource. Li et al. develop a method to store NetCDF or HDF files directly at HDFS and allow Hadoop tools such as HIVE and Spark SQL to directly process them such as subsetting along the time direction [21]. Biookaghazadeh et al. develop an alternative approach to enable subsetting along latitude, longitude, or time directions efficiently [22]. We plan to evaluate those approaches and optimize and integrate them into our system.

V. SUMMARY

A system for visualizing and diagnosing large Earth science data (over Terabyte) has been developed and realized with Hadoop and Spark. It transforms data from the NetCDF format to the CSV format with appropriate indexes. In this way, Earth science data can be stored and organized in HIVE tables and adaptively subset and diagnosed with Hadoop tools such as HIVE and Impala and Spark tools such as Python Spark and Spark SQL. The subset data can be further visualized with IDL and analyzed with Python in an end-to-end way.

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. Tao, Matsui, Li, Zhou are supported by NASA AIST 2014.

REFERENCES

[1] S.Zhou,X.Yang,X.Li,T.Matsui,S.Liu,X.-H.Sun,andW.-K.Tao,"A hadoop-based visualization and diagnosis framework for earth science data," in Proc. of IEEE International Conference on Big Data (IEEE BigData 2015), 2015.
[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] 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.
[4] 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.
[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." HotCloud, 2010.
[6] The Apache Software Foundation, "Spark," https://spark.apache.org.
[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] X. Yang, N. Liu, B. Feng, X.-H. Sun, and S. Zhou, “PortHadoop: Supporting Direct HPC Data Processing in Hadoop,” in 2015 IEEE International Conference on Big Data, Santa Clara, California.

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

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

[15] X. Yang, S. Liu, K. Feng, S. Zhou, X.-H. Sun, “Visualization and Adaptive Subsetting of Earth Science Data in HDFS,” in 6th IEEE International Conference on Big Data and Cloud Computing (BDCloud 2016).

[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] Z. Li, F. Hu, J. L. Schnase, D. Q. Duffy, T. Lee, M. K. Bowen, and C. Yang, “A spatiotemporal indexing approach for efficient process- ing of big array-based climate data with mapreduce,” International Journal of Geographical Information Science, pp. 1–19, 2016.

[22] S. Biookaghazadeh, Y. Xu, S. Zhou, and M. Zhao, “Kaleido: Enabling Efficient Scientific Data Processing on Big-Data Systems,” in 2016 IEEE International Conference on Big Data, Washington DC.